

この資料は英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。こちらの日本語版は参考用としてご利用ください。設計の際には、最新の英語版で内容をご確認ください。

QI153009-7.1.0

はじめに

デザインのサイズおよび複雑さの驚異的な拡大により、デザインの検証は今日の FPGA システムで重大な障壁となっています。内部信号へのアクセスの制限、複雑な FPGA パッケージ、および PCB の電気ノイズのため、デザイン・サイクルの中でデザインのデバッグが最も困難なプロセスとなっています。デザイン・サイクル・タイムの 50% 以上がデザインのデバッグと検証に費やされることもあります。デザインのデバッグを支援するために、アルテラはデザインを FPGA デバイス上でフル・スピードで動作させながら、外部 I/O ピンを使用することなく内部信号の動作を検査できるソリューションを提供します。

SignalTap® II エンベデッド・ロジック・アナライザは、拡張性があり使いやすく、Quartus® II ソフトウェア・サブスクリプションに含まれています。このロジック・アナライザは、外部装置を使用しないでデザインの内部信号の状態を精査することによって、FPGA デザインのデバッグを支援します。カスタム・トリガ条件ロジックを定義して、精度を向上させ、問題を特定する能力を改善します。SignalTap II エンベデッド・ロジック・アナライザは、デザインの内部ノードまたは I/O ピンの状態をキャプチャするために外部プローブやデザイン・ファイルへの変更を必要としません。キャプチャしたすべての信号データは、設計者がデータを読み出して解析できるようになるまでデバイス・メモリに保存されます。

SignalTap II エンベデッド・ロジック・アナライザは、SOPC (system-on-a-programmable-chip) または FPGA デザインでリアルタイムに信号動作をキャプチャおよび表示する、次世代のシステム・レベル・デバッグ・ツールです。SignalTap II エンベデッド・ロジック・アナライザは、最大数のチャンネル、最大のサンプル容量、およびプログラマブル・ロジック市場におけるエンベデッド・ロジック・アナライザの中で最も速いクロック速度をサポートします。図 13-1 に、SignalTap II エンベデッド・ロジック・アナライザを構成するコンポーネントのブロック図を示します。

図 13-1. SignalTap II ロジック・アナライザのブロック図 (1)

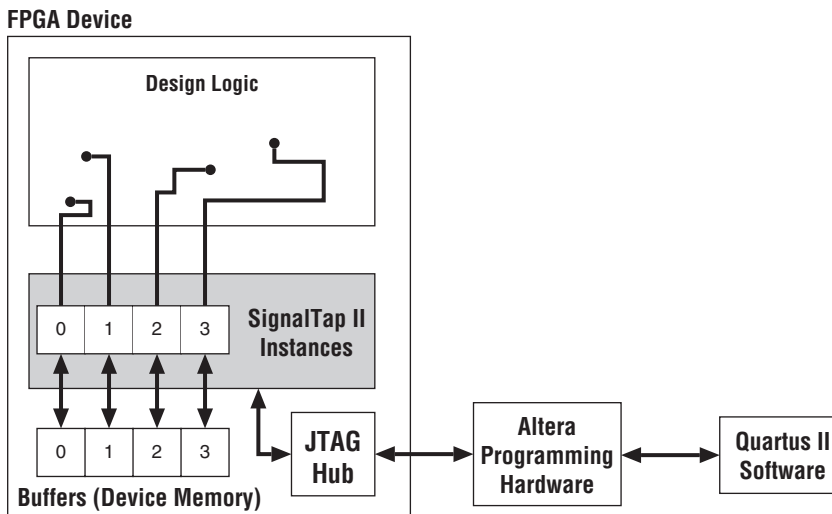


図 13-1 の注 :

- (1) このブロック図では、Quartus II のインクリメンタル・コンパイル機能を使用して、SignalTap II ロジック・アナライザを個別のデザイン・パーティションとしてデザインと共にコンパイルしたものと仮定しています。これは Quartus II ソフトウェアでの新規プロジェクトのデフォルト設定です。インクリメンタル・コンパイルがディセーブルまたは使用されていない場合、SignalTap II ロジックはデザインに統合されます。SignalTap II ロジック・アナライザでのインクリメンタル・コンパイルの使用について詳しくは、13-43 ページの「SignalTap II インクリメンタル・コンパイルを使用したコンパイルの高速化」を参照してください。

この章では、通常のデバイス動作中に、外部ラボ装置を使用しないで FPGA デザインをデバッグする方法について説明します。SignalTap II エンベデッド ロジック・アナライザは従来の外部ロジック・アナライザに類似しているため、外部ロジック・アナライザの動作に関する知識があると役に立ちますが、必ずしも必要ではありません。SignalTap II ロジック・アナライザに変更を加える際に高速コンパイル時間を活かすために、Quartus II インクリメンタル・コンパイル機能の知識が役立ちます。



Quartus II インクリメンタル・コンパイル機能の使用について詳しくは、「Quartus II ハンドブック」の「階層ベースおよびチーム・ベースのデザインのためのインクリメンタル・コンパイル」の章を参照してください。

ハードウェアおよびソフトウェア要件

SignalTap II エンベデッド・ロジック・アナライザによるロジック解析には、以下のコンポーネントが必要です。

- Quartus II デザイン・ソフトウェア
または
Quartus II Web Edition (TalkBack 機能がイネーブルされた状態)
または
SignalTap II ロジック・アナライザのスタンドアロン・ソフトウェア
- ダウンロード / アップロード・ケーブル
- テスト対象のデバイスへの JTAG 接続を備えたアルテラの開発キットまたはユーザ・デザイン・ボード

キャプチャされたデータはデバイスのメモリ・ブロックに格納された後、EthernetBlaster や USB Blaster™ などの JTAG 通信ケーブルを介して Quartus II ソフトウェアにダウンロードされ、波形表示されます。表 13-1 に SignalTap II エンベデッド・ロジック・アナライザの特長と利点を示します。

特長	利点
1 つのデバイス内で複数のロジック・アナライザを使用可能	デザイン内の複数のクロック・ドメインから同時にデータのキャプチャが可能
1 つの JTAG チェイン内の複数デバイスで、複数のロジック・アナライザを使用可能	1 つの JTAG チェイン内の複数デバイスから同時にデータのキャプチャが可能
プラグインをサポート	Nios® II エンベデッド・プロセッサなどの IP に対するノード、トリガ、および信号モニターを簡単に指定可能
アナライザ・インスタンスごとに最大 10 レベルの基本または拡張トリガ条件を設定可能	より複雑なデータ・キャプチャ・コマンドをロジック・アナライザに送信し、精度の向上と問題の切り分けを達成可能
パワーアップ・トリガ	手動によるロジック・アナライザの起動前ではなくデバイス・プログラミング後に発生するトリガで信号データのキャプチャが可能
インクリメンタル・コンパイル	SignalTap II ロジック・アナライザでモニタする信号およびトリガをフル・コンパイルを実行しないで変更することによって時間を節約
柔軟なバッファ・モード	イベントの起動タイミングを基準として、サンプリングする範囲をトリガごとに個別指定して、データをより的確にキャプチャすることが可能
MATLAB とインクルードされた MEX ファンクションとの統合	SignalTap II ロジック・アナライザでキャプチャしたデータを MATLAB 整数マトリックスに取得
各デバイスで最大 1,024 チャンネルをサポート	多数の信号およびワイド・バス構造をサンプリング

特長	利点
各デバイスで最大 128,000 のサンプリングが可能	各チャンネルで大容量サンプル・セットのキャプチャが可能
高速クロック周波数	最大 270 MHz でサンプリング・データを収集
リソースの推定	SignalTap II エンベデッド・ロジック・アナライザ・コンフィギュレーションで使用されるロジックおよびメモリ・デバイス・リソースを推定
追加コストが不要	SignalTap II ロジック・アナライザは、Quartus II サブスクリプションおよび Quartus II Web Edition (TalkBack 機能がイネーブルされた状態) で使用可能

SignalTap II ロジック・アナライザは以下のデバイスをサポートしています。

- Stratix® III
- Stratix II
- Stratix II GX
- Stratix
- Stratix GX
- HardCopy® II
- HardCopy Stratix
- Cyclone® III
- Cyclone II
- Cyclone
- APEX™ II
- APEX 20KE
- APEX 20KC
- APEX 20K
- MAX® II
- Excaliber™ ARM
- Mercury™

オン・チップ・ デバッグ・ ツールの比較

Quartus II ソフトウェアは、デバイス・プログラミング後の FPGA デザインのデバッグを支援するさまざまな方法を提供します。SignalTap II ロジック・アナライザ、SignalProbe™、およびロジック・アナライザ・インタフェース (LAI) には類似機能がいくつかありますが、それぞれ利点を持っています。デバッグの状況によっては、どのツールが最適か、あるいは複数のツールを使用する必要があるのかどうかの判断が困難な場合があります。表 13-2 では、これらのツール間で一般的なデバッグ機能を比較し、機能ごとに使用するのに最適なツールを推奨します。

表 13-2. 一般的なデバッグ機能に推奨されるオン・チップ・デバッグ・ツール (1) (1 / 2)

特長	Signal Probe	ロジック・アナライザ・インタフェース (LAI)	SignalTap II エンベデッド・アナライザ	説明
大きなサンプル容量	N/A	√	—	LAI と共に使用する外部ロジック・アナライザは容量が大きいバッファを搭載し、SignalTap II ロジック・アナライザより多くのキャプチャ・データを格納できます。SignalProbe でデータをキャプチャまたは格納することはできません。
デバッグの容易さ タイミング問題	N/A	√	—	LAI と共に使用する外部ロジック・アナライザでは、タイミング・モードにアクセスして組み合わせたデータ・ストリームをデバッグできます。
ロジック・デザインへの影響を最小化	√	√(2)	√(2)	LAI はデザインに最小限のロジックを追加し、必要なデバイス・リソースを低減します。SignalTap II ロジック・アナライザは、インクリメンタル・コンパイルを使用して個別のデザイン・パーティションとして設定された場合、デザインにほとんど影響を与えません。SignalProbe はノードをピンにインクリメンタルに配線し、デザインにはまったく影響を与えません。
短時間でのコンパイルおよびリコンパイル時間	√	√(2)	√(2)	SignalProbe では、以前に予約したピンにインクリメンタルに信号を配線し、非常にわずかな再コンパイル時間でソース信号の選択を変更することができます。SignalTap II ロジック・アナライザおよびLAI では、インクリメンタル・コンパイルを活用して独自のデザイン・パーティションを修正し、再コンパイル時間を短縮できます。
トリガ機能	N/A	√	—	SignalTap II ロジック・アナライザでは拡張トリガ機能を使用できますが、LAI と共に使用する場合、多くのトリガ・オプションは外部ロジック・アナライザでしか選択できません。
I/O 使用率	—	—	√	SignalTap II ロジック・アナライザでは、追加出力ピンは必要ありません。LAI およびSignalProbe は両方とも I/O ピン・アサインメントが必要です。

特長	Signal Probe	ロジック・アナライザ・インタフェース (LAI)	SignalTap II エンベデッド・アナライザ	説明
収集速度	N/A	—	√	SignalTap II ロジック・アナライザは 200 MHz 以上の速度でデータを収集することができます。LAI と共に使用される外部ロジック・アナライザでも同じ収集速度を達成できますが、シグナル・インテグリティ問題によって制限される場合があります。
JTAG 接続が必要	√	—	—	SignalTap II ロジック・アナライザまたは LAI を使用した FPGA デザインの場合、Quartus II ソフトウェアを実行するホストにアクティブ JTAG 接続が必要です。SignalProbe はデバッグのためのホストは必要ありません。
外部装置	—	—	√	SignalTap II ロジック・アナライザのロジックは、完全にプログラムされた FPGA デバイスの内部に組み込まれます。Quartus II ソフトウェアまたはスタンドアロン SignalTap II ソフトウェアを実行するホストからの JTAG 接続以外に装置は必要ありません。SignalProbe および LAI では、マルチメータ、オシロコプ、ロジック・アナライザなどの外部デバッグ装置を使用する必要があります。

表 13-2 の注：

- (1) √ はその機能に推奨される最適なツールを示します。— はツールをその機能に使用できるが、必ずしも最良の結果をもたらさないことを示します。N/A はその機能は選択したツールに適用できないことを示します。
- (2) インクリメンタル・コンパイルでの使用

SignalTap II ロジック・アナライザに必要なロジック・リソースを追加しないで、外部装置を使用して信号をモニタする場合、Quartus II ソフトウェアで使用できる他のツールの使用を検討してください。ロジック・アナライザ・インタフェース (LAI) を使用した場合、多数の信号の多重化バンクでは一部のピンしか表示できませんが、信号は SignalProbe を使用した ECO 変更の一環として予約 I/O ピンに迅速に配線できます。

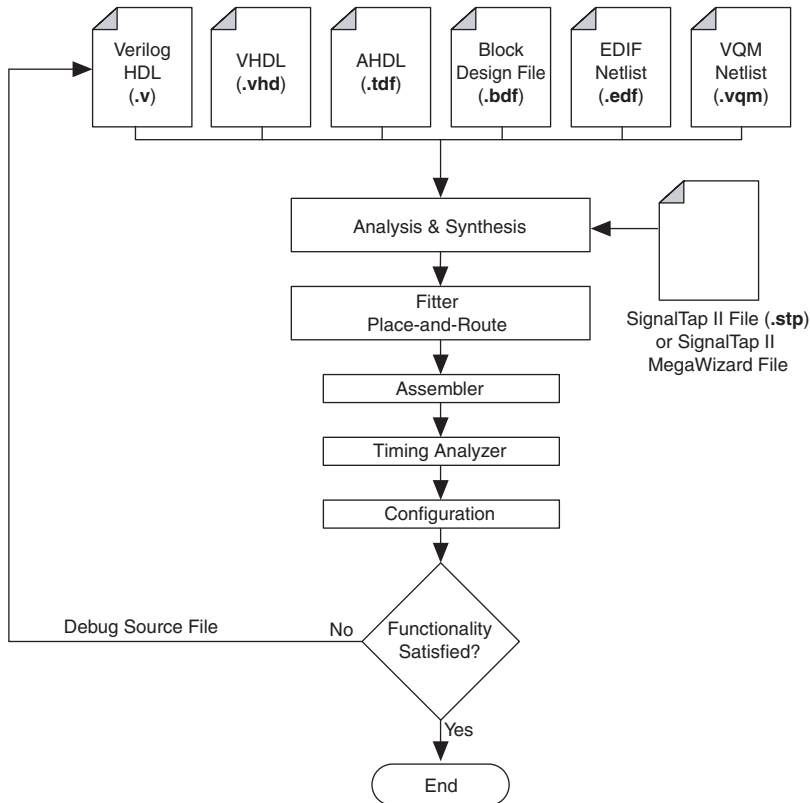


これらのツールの使用について詳しくは、「Quartus II ハンドブック Volume 3」の「Quick Design Debugging Using SignalProbe」および「In-System Debugging Using External Logic Analyzers」の章を参照してください。

SignalTap II ロジック・ アナライザ を使用した デザイン・ フロー

図 13-2 に、デザインに SignalTap II ロジック・アナライザを使用した標準的な FPGA デザイン・フロー全体を示します。SignalTap II ファイル (.stp) がプロジェクトに追加されイネーブルされるか、または MegaWizard® Plug-in Manager で作成した SignalTap II HDL ファンクションがデザインでインスタンス化されます。図は、最初に SignalTap II ロジック・アナライザをデザインに追加するところから、最後のデバイスのコンフィギュレーション、テスト、およびデバッグまでのフローを示しています。

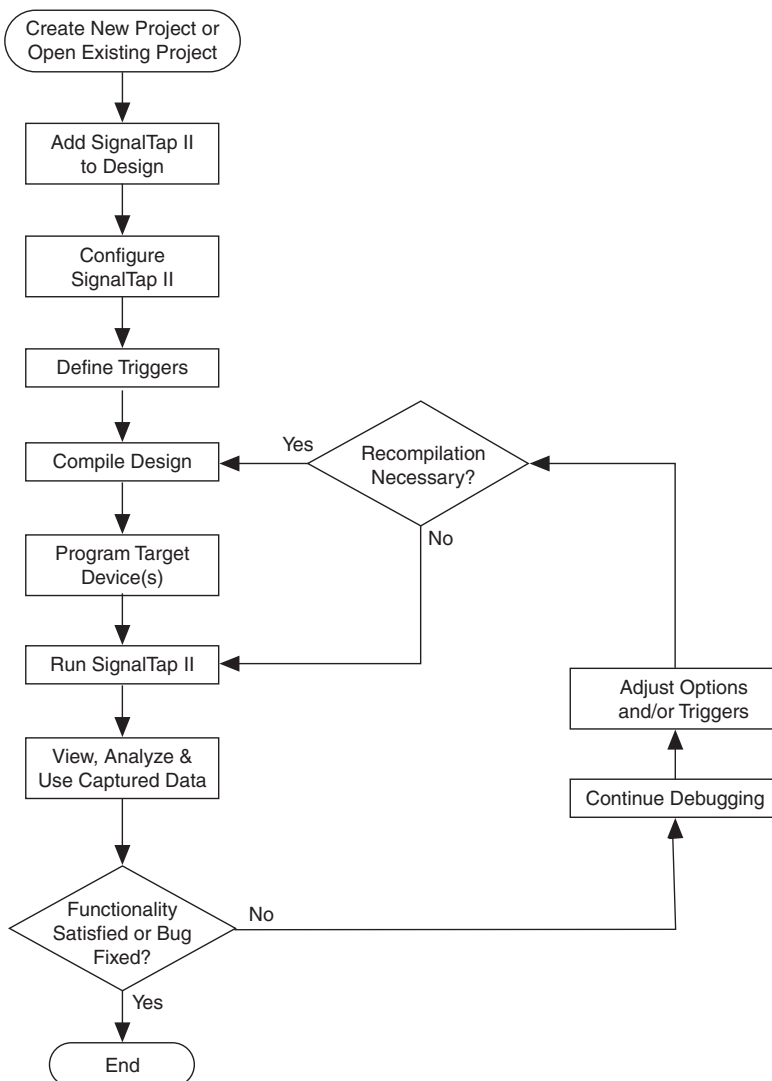
図 13-2. SignalTap II FPGA デザイン & デバッグ・フロー



SignalTap II ロジック・アナライザ のタスク・ フロー

SignalTap II ロジック・アナライザを使用してデザインをデバッグするには、ロジック・アナライザの追加、コンフィグレーション、および実行などの多くの作業を実行します。図 13-3 に、デザインのデバッグを完了するための作業の標準的なフローを示します。各作業の詳細については、この章の該当する部分を参照してください。

図 13-3. SignalTap II ロジック・アナライザのタスク・フロー



デザインへの SignalTap II ロジック・アナライザの追加

MegaWizard Plug-In Manager を使用して SignalTap II ファイルを作成するか、またはロジック・アナライザのパラメータ化された HDL インスタンス表現を作成します。複数のクロック・ドメインを同時にモニタする場合、デザインにロジック・アナライザのインスタンスを追加することができますが、デバイスで使用可能なリソースによってのみ制限されます。

SignalTap II ロジック・アナライザのコンフィギュレーション

SignalTap II ロジック・アナライザをデザインに追加したら、それをコンフィギュレーションして必要な信号をモニタします。信号を手動で追加するか Nios II plug-in などのプラグインを使用して、特定の IP に関連する信号の全セットを素早く追加することができます。また、サイズ、データのキャプチャおよび格納方法、およびメモリ・タイプ選択が可能なデバイス内のバッファで使用するデバイス・メモリ・タイプなど、データ・キャプチャ・バッファの設定を指定できます。

トリガの定義

SignalTap II ロジック・アナライザは、動作中に継続的にデータをキャプチャします。特定の信号データをキャプチャおよび格納するには、トリガをセットアップしてロジック・アナライザがデータのキャプチャを停止する条件を指示します。SignalTap II ロジック・アナライザでは、1 つの信号の立ち上がりエッジなどの単純なものから、信号グループ、追加ロジック、および複数条件などを含む非常に複雑なものまで、幅広いランタイム・トリガを定義することができます。パワーアップ・トリガは、コンフィギュレーション後にデバイスがユーザ・モードに入った直後に発生するトリガ・イベントからデータをキャプチャする能力を提供します。

デザインのコンパイル

SignalTap II ファイルをコンフィギュレーションし、トリガを定義したら、プロジェクトを通常どおりコンパイルして、ロジック・アナライザをデザインに含めます。デバッグ時には頻繁にモニタする信号ノードを変更したり、トリガ設定を調整する必要があるため、コンパイル時間を短縮するために Quartus II ソフトウェアのインクリメンタル・コンパイルに加え、SignalTap II ロジック・アナライザに組み込まれたインクリメンタル・コンパイル機能を使用することが推奨されます。

ターゲット・デバイスのプログラミング

SignalTap II ロジック・アナライザを使用してデザインをデバッグする場合、Quartus II Programmer を使用しないで SignalTap II ファイルから直接ターゲット・デバイスをプログラムすることができます。また、異なるデザインを使用して複数のデバイスをプログラムし、それらを同時にデバッグすることも可能です。

SignalTap II ロジック・アナライザの実行

通常のデバイス動作では、JTAG 接続を介してロジック・アナライザを制御し、データ・キャプチャを開始するためのトリガ条件の検索開始タイミングを指定します。ランタイムまたはパワーアップ・トリガでは、キャプチャしたデータをオン・チップ・バッファから読み込んで SignalTap II ファイルに転送して解析します。

キャプチャしたデータの表示、解析、使用

データをキャプチャし SignalTap II ファイルに読み込んだら、それを解析したりデバッグ・プロセスで使用することができます。手動またはプラグインを使用してモニター・テーブルをセットアップでき、簡単にキャプチャした信号データを読み込んで解釈できるようになります。SignalTap II ノード・リストの検索機能を使用して、Quartus II ソフトウェアの他のツールで問題のあるノードの位置を迅速に検出し、デバッグをスピードアップすることができます。後で解析するためにキャプチャしたデータを保存するか、共有およびさらなる研究のために他のフォーマットに変換します。

デザインへの SignalTap II ロジック・アナライザの追加

SignalTap II ロジック・アナライザはターゲット・デバイスのロジックに実装されるため、デザイン自体の一部として FPGA デザインに追加する必要があります。SignalTap II ロジック・アナライザを生成し、デザインに追加してデバッグを行う方法が 2 つあります。1 つは SignalTap II ファイル (.stp) を作成し、SignalTap II エディタを使用してロジック・アナライザの詳細をコンフィギュレーションする方法です。もう 1 つは、MegaWizard Plug-in Manager で SignalTap II ファイルを作成およびコンフィギュレーションし、デザインでインスタンス化する方法です。

SignalTap II ファイルの作成およびイネーブル

エンベデッド・ロジック・アナライザを作成するには、既存の SignalTap II ファイルを使用するか新しくファイルを作成することができます。ファイルを作成または選択したら、それが使用されているプロジェクトでイネーブルする必要があります。

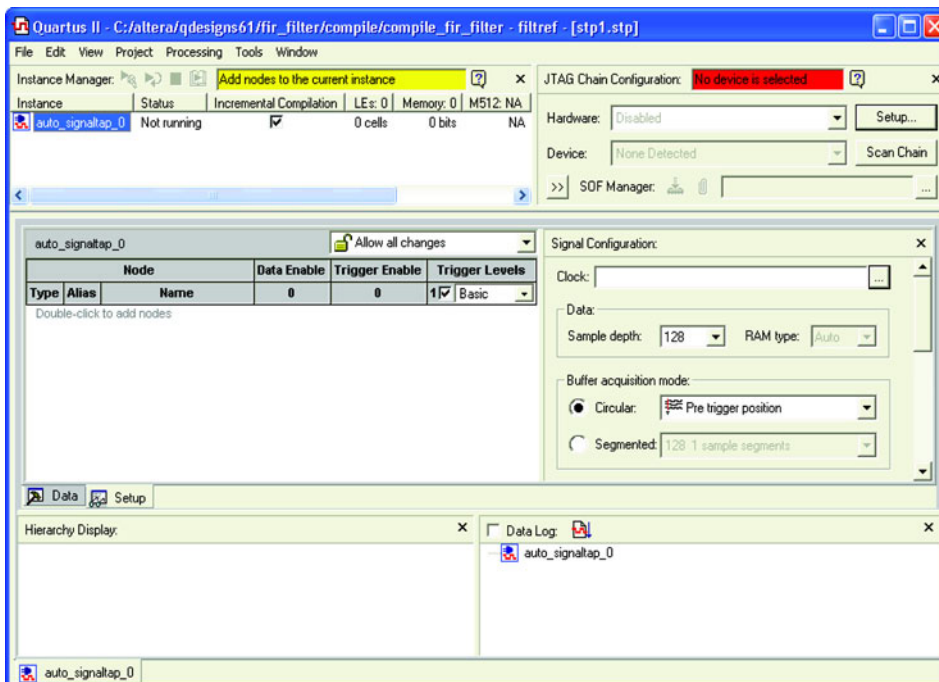
SignalTap II ファイルの作成

SignalTap II ファイルには、SignalTap II ロジック・アナライザの設定と表示および解析するためにキャプチャしたデータが含まれます。新しい SignalTap II ファイルを作成するには、以下のステップを実行します。

1. File メニューの **New** をクリックします。
2. **New** ダイアログ・ボックスの **Other Files** タブをクリックし、**SignalTap II File** を選択します。
3. **OK** をクリックします。

既にプロジェクトと関連付けられている既存の SignalTap II ファイルを開くには、Tools メニューの **SignalTap II Logic Analyzer** をクリックします。また、現在のプロジェクトに SignalTap II ファイルが存在しない場合は、この方法で新しい SignalTap II ファイルを作成するか、File メニューの **Open** をクリックして SignalTap II file (図 13-4) を選択して既存のファイルを開くことができます。

図 13-4. SignalTap II エディタ



現在のプロジェクトに対する SignalTap II ファイルのイネーブルおよびディセーブル

新しい SignalTap II ファイルを保存するたびに、Quartus II ソフトウェアは現在のプロジェクトに対してそのファイルをイネーブルするかどうかを確認します。しかし、以下のステップを実行して、このファイルを手動で追加したり、選択した SignalTap II ファイルを変更したり、ロジック・アナライザを完全にディセーブルすることができます。

1. Assignments メニューの **Settings** をクリックします。Settings ダイアログ・ボックスが表示されます。
2. **Category** リストで、**SignalTap II Logic Analyzer** を選択します。SignalTap II Logic Analyzer ページが表示されます。
3. **Enable SignalTap II Logic Analyzer** をオンにします。このオプションをオフにするとロジック・アナライザはディセーブルされ、デザインから完全に削除されます。

4. **SignalTap II File name** ボックスに、デザインに含める SignalTap II ファイルの名前を入力するか、参照してファイル名を選択します。
5. **OK** をクリックします。

MegaWizard Plug-In Manager を使用したエンベデッド・ロジック・アナライザの作成

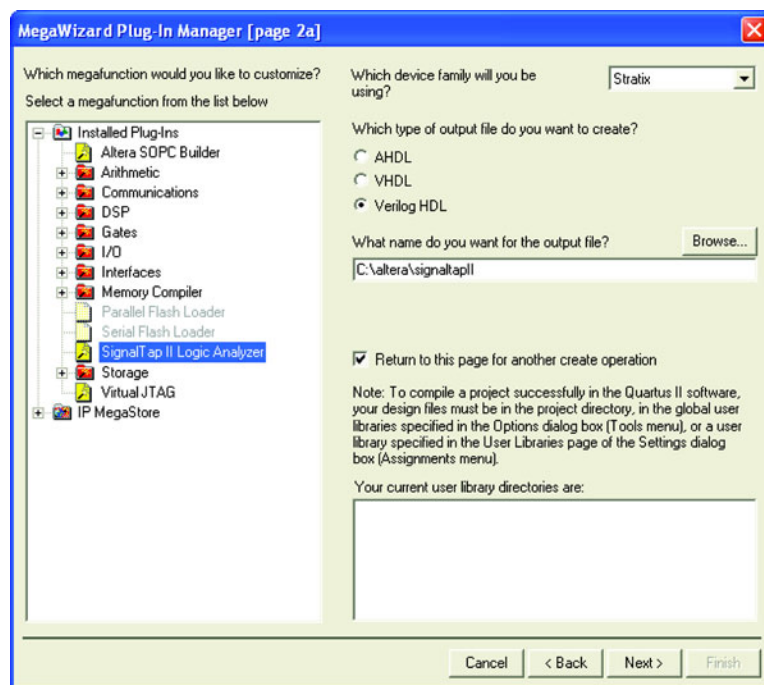
MegaWizard Plug-In Manager を使用して SignalTap II ロジック・アナライザ・インスタンスを作成することもできます。MegaWizard Plug-In Manager はデザインでインスタンス化する HDL ファイルを生成します。また、HDL で MegaWizard Plug-In Manager ファイルをインスタンス化するハイブリッド手法を使用して、13-10 ページの「[SignalTap II ファイルの作成およびイネーブル](#)」で説明されている方法を使用することもできます。

MegaWizard Plug-In Manager を使用した HDL 表現の作成

Quartus II ソフトウェアにより、MegaWizard Plug-In Manager を使用して簡単に SignalTap II ロジック・アナライザを作成することができます。SignalTap II メガファンクションを実装するには、以下のステップを実行します。

1. Tools メニューの **MegaWizard Plug-In Manager** をクリックします。**MegaWizard Plug-In Manager** ダイアログ・ボックスが表示されます。
2. **Create a new custom megafunction variation** を選択します。
3. **Next** をクリックします。
4. **Installed Plug-Ins** リストで、**SignalTap II** ロジック・アナライザを選択します。出力ファイルの種類を選択し、希望の SignalTap II メガファンクションの名前を入力します。出力ファイル (図 13-5) の種類として、AHDL (.tdf)、VHDL (.vhd)、または Verilog HDL (.v) を選択することができます。

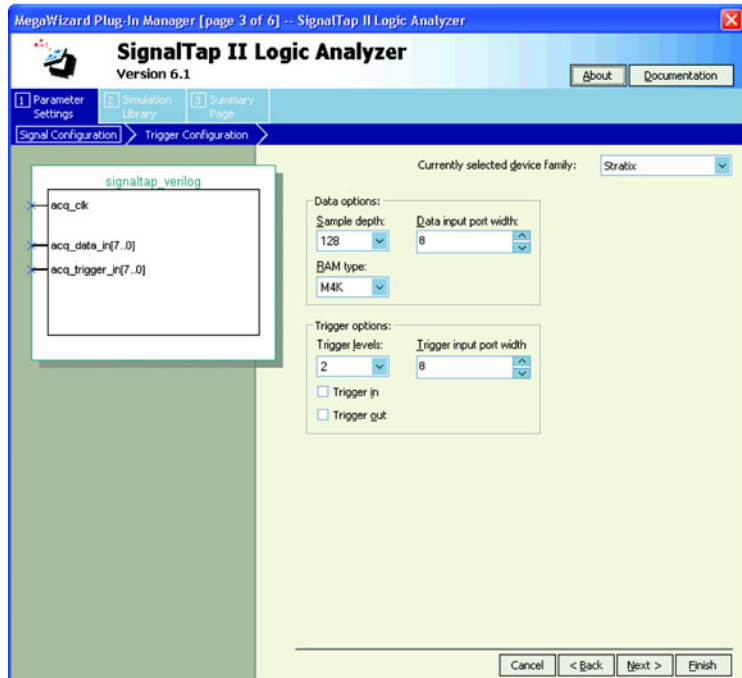
図 13-5. MegaWizard Plug In Manager での SignalTap II ロジック・アナライザの作成



5. **Next** をクリックします。
6. **Sample depth**、**RAM Type**、**Data input port width**、**Trigger levels**、**Trigger input port width**、および外部 **Trigger in** または **Trigger out** (図 13-6) をイネーブ爾するかどうかを指定して、アナライザをコンフィギュレーションします。

これらの設定について詳しくは、この章の 13-20 ページの「SignalTap II ロジック・アナライザのコンフィギュレーション」および 13-31 ページの「トリガの定義」を参照してください。

図 13-6.ロジック・アナライザ・パラメータの選択

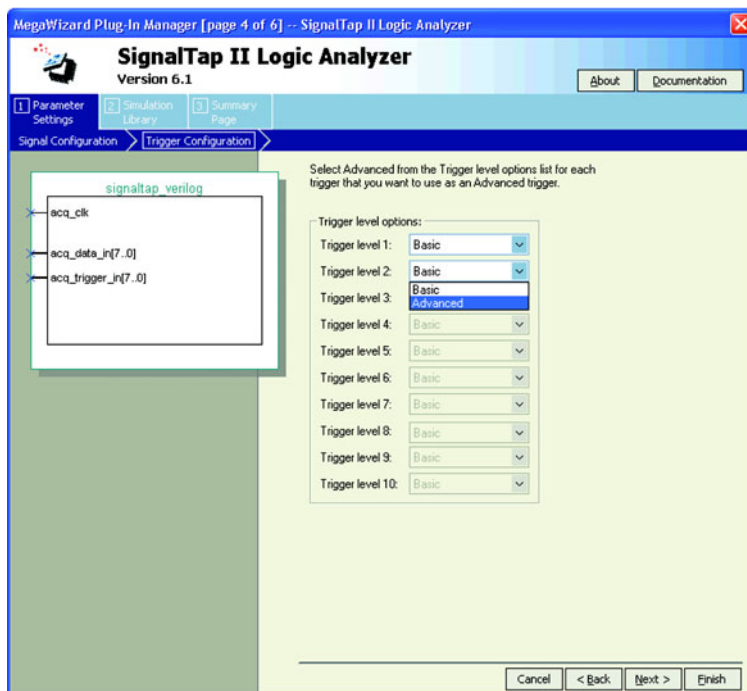


7. **Next** をクリックします。
8. **Basic** または **Advanced** (図 13-7) を選択して、**Trigger level** オプションを設定します。トリガ・レベルで **Advanced** を選択した場合、MegaWizard Plug-In Manager の次のページで **Advanced Trigger Condition Editor** が表示されます。トリガ入力ポート幅に指定した信号数を使用して、拡張トリガ式をコンフィギュレーションすることができます。



MegaWizard Plug-In Manager を使用してパワーアップ・トリガを定義することはできません。SignalTap II ファイルを使用してこれを行う方法について詳しくは、13-31ページの「トリガの定義」を参照してください。

図 13-7. MegaWizard Basic および拡張トリガ・オプション



9. MegaWizard Plug-In Manager の最後のページで、作成する追加のファイルを選択し、**Finish** をクリックして SignalTap II ロジック・アナライザの HDL 表現を作成します。

MegaWizard Plug-In Manager のコンフィギュレーション設定オプションについて詳しくは、13-20 ページの「SignalTap II ロジック・アナライザのコンフィギュレーション」を参照してください。トリガの定義について詳しくは、13-31 ページの「トリガの定義」を参照してください。

SignalTap II メガファンクション・ポート

表 13-3 に、SignalTap II メガファンクション・ポートに関する情報を記載しています。



このメガファンクションのポートおよびパラメータの最新情報は、Quartus II Help を参照してください。

ポート名	タイプ	必要	説明
acq_data_in	入力	なし	この信号セットは、SignalTap II ロジック・アナライザでモニタする信号セットを表します。
acq_trigger_in	入力	なし	この信号セットは、アナライザをトリガするのに使用する信号セットを表します。
acq_clk	入力	○	このポートは、SignalTap II ロジック・アナライザがデータのキャプチャするのに使用するサンプリング・クロックを表します。
trigger_in	入力	なし	この信号は、SignalTap II ロジック・アナライザをトリガするために使用されます。
trigger_out	出力	なし	この信号はトリガ・イベントが発生するとイネーブルされます。

HDL での SignalTap II ロジック・アナライザのインスタンス化

HDL でのロジック・アナライザのインスタンス化は、デザインにおける他の Verilog HDL または VHDL メガファンクションのインスタンス化に類似しています。MegaWizard Plug-In Manager で生成されたファイルのコードをデザインに追加し、デザインの信号を適切な SignalTap II メガファンクション・ポートにマップします。デザインで最大 127 のアナライザ、または FPGA に物理的にフィットする数のアナライザをインスタンス化することができます。HDL ファイルで SignalTap II ファイルをインスタンス化したら、ロジック・アナライザがターゲット FPGA にフィットするように Quartus II プロジェクトをコンパイルします。

データをキャプチャおよび表示するには、SignalTap II HDL 出力ファイルから SignalTap II ファイルを作成する必要があります。これを行うには、File メニューの Create/Update を選択して、**Create SignalTap II File from Design Instance(s)** をクリックします。

デザインまたは SignalTap II インスタンスを変更する場合、このコマンドを使用して SignalTap II ファイルを再作成またはアップデートします。これにより、SignalTap II ファイルは常にデザインの SignalTap II インスタンスと互換性があります。SignalTap II ファイルがデザインの

SignalTap II インスタンスと互換性がない場合、SignalTap II ロジック・アナライザをデバイスにプログラムすると制御できなくなる可能性があります。

SignalTap II ファイルとプログラムされた SignalTap II インスタンスの互換性について詳しくは、13-47 ページの「ターゲット・デバイスのプログラミング」を参照してください。

1 個の FPGA への複数のアナライザの実装

SignalTap II ロジック・アナライザには、1 個の FPGA デバイスでの複数のロジック・アナライザのサポートが含まれています。この機能により、デザインの各クロック・ドメインに対して固有のロジック・アナライザを作成できます。アナライザの複数のインスタンスを SignalTap II ファイルに追加すると、それに比例してリソースの使用率が増加します。

この機能により、複数のクロック・ドメインのデバッグに加えて、同じ SignalTap II の設定を同じクロック・ドメインの信号グループに適用することができます。例えば、あるクロック・ドメインに 64K のサンプル容量を使用する必要がある信号セットがあり、また同じクロック・ドメインに 1K のサンプル容量を必要とする別の信号セットがある場合は、これらの要件を満たす 2 つのインスタンスを生成できます。

複数のアナライザを作成するには、Edit メニューの **Create Instance** をクリックするか、または Instance Manager ウィンドウ内で右クリックして、**Create Instance** をクリックします。

SignalTap II ロジック・アナライザの各インスタンスは、個別にコンフィギュレーションすることができます。Instance Manager のコンフィギュレーションに使用できるアクティブなインスタンスのアイコンは、カラーで表示され、青色のボックスで囲まれます。別のインスタンスをコンフィギュレーションするには、Instance Manager で別のインスタンスのアイコンまたはインスタンス名をダブル・クリックします。

SignalTap II ロジック・アナライザで使用する FPGA リソースのモニタ

SignalTap II ロジック・アナライザは、ロジック・リソースと各 SignalTap II ロジック・アナライザが使用するメモリ量を計算するリソース使用推定機能を内蔵しています。ロジック・アナライザの各インスタンスのリソース使用率および SignalTap II Editor の Instance Manager セクションのカラムで使用される全リソースを表示することができます。この機能は、デバイス・リソースに制限があり、かつ SignalTap II ロジック・アナライザが使用するデバイス・リソースを知る必要がある場合に役立ちます。リソース使用推定機能がレポートする値は、実際のリソース使用率とは 5% 程度異なる場合があります。

表 13-4 に、リストされているデバイスに対する信号幅およびサンプル容量あたりの、SignalTap II ロジック・アナライザ M4K メモリ・ブロック・リソースの使用率を示します。

信号 (幅)	サンプル (容量)			
	256	512	2,048	8,192
8	< 1	1	4	16
16	1	2	8	32
32	2	4	16	64
64	4	8	32	128
256	16	32	128	512

表 13-4 の注 :

- (1) SignalTap II ロジック・アナライザをコンフィギュレーションする場合、Instance Manager はコンフィギュレーションの実装に必要なメモリ・ビットおよびロジック・エレメントの推定値をレポートします。

SignalTap II ロジック・ アナライザの コンフィギュ レーション

SignalTap II ファイルは、ロジック・アナライザのインスタンスをコンフィギュレーションするための多くのオプションを提供します。設定のいくつかは従来の外部ロジック・アナライザのものに類似しています。その他の設定は、エンベデッド・ロジック・アナライザのコンフィギュレーションが必要なため、SignalTap II ロジック・アナライザ固有のものであります。これらの設定により、デザインをデバッグするのに役立つ方法でロジック・アナライザをコンフィギュレーションすることができます。



いくつかの設定は、パワーアップ・トリガ条件ではなくランタイム・トリガ条件を表示しているときにのみ調整できます。パワーアップ・トリガおよびその他のトリガ条件の表示について詳しくは、13-36 ページの「パワーアップ・トリガの作成」を参照してください。

アクイジション・クロックの割り当て

SignalTap II ロジック・アナライザによるデータ収集を制御するために、クロック信号を割り当てる必要があります。ロジック・アナライザは、アクイジション・クロックの立ち上がりエッジでデータをサンプリングします。デザイン内のどの信号でもアクイジション・クロックとして使用できます。しかし、最良の結果を得るために、アルテラではデータ収集にグローバルのゲートなしクロックの使用を推奨しています。ゲート付きクロックをアクイジション・クロックとして使用した場合、デザインの動作を正確に反映しない予期しないデータが発生する可能性があります。Quartus II クラシック・タイミング・アナライザは、デザインを実行できる最大アクイジション・クロック周波数を表示します。

アクイジション・クロックを割り当てるには、以下のステップを実行します。

1. SignalTap II Logic Analyzer ウィンドウの **Setup** タブをクリックします。
2. Signal Configuration ペインの **Clock** フィールドの横にある **Browse** をクリックします。Node Finder ダイアログ・ボックスが表示されます。
3. インクリメンタル・コンパイルを使用する場合は、**Filter** リストで **SignalTap II: post-fitting** を選択します。
または
SignalTap II インクリメンタル・コンパイル機能をディセーブルしている場合は、**Filter** リストで **SignalTap II: pre-synthesis** を選択します。



インクリメンタル・コンパイルを使用しない場合は、**SignalTap II: post-fitting** フィルタに表示されるクロック信号を挿入することはできません。その逆もいえます。インクリメンタル・コンパイルを使用する場合は、**SignalTap II: pre-synthesis** フィルタに表示されるクロック信号を使用することはできません。SignalTap II でのインクリメンタル・コンパイルの使用については、[13-42 ページの「デザインのコンパイル」](#)を参照してください。

4. **Named** フィールドで、サンプル・クロックとして使用するノードの正確な名前を入力するか、またはノード名の一部およびワイルドカード文字を使用してノードを検索します。
5. ノードの検索を開始するには、**List** をクリックします。
6. **Nodes Found** リストで、デザインのグローバル・クロック信号を表すノードを選択します。
7. 「>」をクリックするか、ノード名をダブル・クリックして、選択したノード名を **Selected Nodes** リストに追加します。
8. **OK** をクリックします。これで、このノードが SignalTap II エディタでのアクイジション・クロックとして指定されます。

SignalTap II エディタでアクイジション・クロックを割り当てない場合、Quartus II ソフトウェアは `auto_stp_external_clk` という名前のクロック・ピンを自動的に生成します。

このピンに対してデザインとは独立したピン・アサインメントを作成する必要があります。デザインのクロック信号がアクイジション・クロックをドライブしなければなりません。




ピンへの信号の割り当てについては、「[Quartus II ハンドブック Volume 2](#)」の「[I/O Management](#)」の章を参照してください。

SignalTap II ファイルへの信号の追加

ロジック・アナライザのコンフィギュレーション中、SignalTap II ファイルのノード・リストに信号を追加して、デザイン内でモニタする信号を選択します。選択した信号はトリガの定義にも使用します。SignalTap II ファイルに、次の2種類の信号を割り当てることができます。

- 合成前 — 合成前信号は、合成の最適化が行われる前ではなく、デザイン・エラボレーション後に生成されます。この信号セットは RTL (レジスタ転送レベル) 信号を反映しなければなりません。
- フィットティング後 — フィットティング後信号はフィジカル・シンセシスの最適化および配置配線後に生成されます。

 インクリメンタル・コンパイルを使用しない場合、SignalTap II ファイルには合成前信号のみ追加します。これは、デザインに変更を加えた後に新しいノードを素早く追加したい場合に役立ちます。これを行うには、Processing メニューの **Start Analysis & Elaboration** をクリックします。

Analysis & Elaboration が成功すると、無効な信号は赤色で表示されますので、SignalTap II ファイルを正しく動作させるには、これらの信号を削除する必要があります。これは、例えば SignalTap II ファイルに合成前信号を追加してから、インクリメンタル・コンパイルをイネーブルした場合に発生します。SignalTap II インクリメンタル・コンパイルではフィットティング後信号のみ使用されるため、フィットティング後信号が存在しない場合には合成前信号は無効です。また、SignalTap II Health Monitor は SignalTap II ファイルに無効なノード名が存在している場合も通知します。

I/O ピンに割り当てられた信号は、フィットティング後のノードとして表示される場合のみ直接タップできます。I/O ピンの合成前信号をタップすることはできません。しかし、合成前の出力ピンを直接ドライブするロジックをタップすることはできます。合成前の入力ピンの場合、信号が割り当てられた後にそのピンに対して生成された COMBOUT 信号または REGOUT 信号をタップします。

SignalTap II ロジック・アナライザでのインクリメンタル・コンパイルの使用について詳しくは、13-43 ページの「[SignalTap II インクリメンタル・コンパイルを使用したコンパイルの高速化](#)」を参照してください。

信号の保持

多くの RTL 信号は合成および配置配線のプロセス中に最適化されます。これにより、フィットティング後信号の名前が RTL 名とは大幅に異なるため、デザインのデバッグ時に問題が生じる可能性があります。また、インクリメンタル・コンパイルを使用している場合にも問題が発生することがあります。インクリメンタル・コンパイルを使用する場合は、フィットティング後信号のみ SignalTap II ロジック・アナライザに追加できるため、モニタしたい RTL 信号がフィットティング後に生成されず、使用できない可能性があります。この問題を回避するために、合成属性を使用して合成および配置配線中に信号を保持することができます。Quartus II ソフトウェアはこれらの合成属性が検出すると、指定された信号の最適

化は実行しないで、強制的にフィッティング後のネットリストに存在させます。ただし、これを行うとリソースの使用率が増加したり、タイミング性能が低下する場合があります。使用できる属性は次の2つです。

- **Keep**—この属性によって組み合わせ信号が削除されないようにします。
- **Preserve**—この属性はレジスタが削除されないようにします。



これらの属性の使用について詳しくは、「Quartus II ハンドブック Volume 1」の「Quartus II Integrated Synthesis」の章を参照してください。

Nios II CPU などの IP コアまたはその他の暗号化された IP をデバッグする場合は、SignalTap II ロジック・アナライザでのデバッグで利用できるよう、コアのノードを保持しなければならないこともあります。これは、プラグインを使用して特定の IP 用の信号グループを追加する場合に必要がよくあります。これを行うには、Assignments メニューの **Settings** をクリックします。Category リストで、**Analysis & Synthesis Settings** を選択します。**Create debugging nodes for IP cores** をオンにして、これらのノードを SignalTap II ロジック・アナライザで利用できるようにします。

データ信号の割り当て

データ信号を割り当てるには、以下のステップを実行します。

1. 解析およびエラーポレーション、または解析および合成を実行するかデザインをコンパイルします。
2. SignalTap II Logic Analyzer ウィンドウの **Setup** タブをクリックします。
3. SignalTap II エディタのノード・リスト内でダブル・クリックして、**Node Finder** ダイアログ・ボックスを開きます。
4. **Fitter** リストで、**SignalTap II: pre-synthesis** または **SignalTap II: post-fitting** を選択します。これらのフィルタのいずれかに表示された信号のみ SignalTap II ノード・リストに追加することができます。信号はその他のフィルタから選択することはできません。



SignalTap II インクリメンタル・コンパイル機能を使用する場合、フィッティング後のノードのみノード・リストに追加することができます。

5. **Named** フィールドで、ノード名を入力するかノード名の一部およびワイルドカード文字を入力して特定のノードを検索します。ノード名の検索を開始するには、**List** をクリックします。
6. **Nodes Found** リストで、**SignalTap II** ファイルに追加するノードまたはバスを選択します。
7. 「>」をクリックするかノード名をダブル・クリックして、選択したノード名を **Selected Nodes** リストに追加します。
8. 選択したノードを **SignalTap II** ファイルに挿入するには、**OK** をクリックします。**SignalTap II** ロジック・アナライザのデフォルト・カラー設定では、リストの合成前の信号は黒色、フィッティング後の信号は青色で表示されます。



また、信号を **Node Finder** ダイアログ・ボックスから **SignalTap II** ファイルにドラッグ・アンド・ドロップすることもできます。

ノード・リスト信号の使用オプション

信号をノード・リストに追加すると、ロジック・アナライザでの信号の使用方法を指定するオプションを選択できます。**SignalTap II** ファイルのノード・リストの信号に対する **Trigger Enable** をディセーブルすると、アナライザをトリガする信号の機能をオフにすることができます。このオプションは、キャプチャした信号のデータのみ表示し、その信号をトリガの一部として使用しない場合に役立ちます。

Data Enable カラムをディセーブルすると、信号のデータの表示機能をオフにすることができます。このオプションは、信号でトリガするがその信号のデータを表示しない場合に役立ちます。

ノード・リストの信号を使用した **SignalTap II** トリガ条件の生成について詳しくは、[13-31 ページの「トリガの定義」](#)を参照してください。

タップ不能信号

デザインのフィッティング後の信号には、**Node Finder** ダイアログ・ボックスの **SignalTap II: post-fitting** フィルタにないものもあります。以下の信号タイプはタップできません。

- キャリー・チェーンの一部である信号 — ロジック・エレメントのキャリー・アウト (cout0 または cout1) 信号はタップできません。アーキテクチャの制約のため、キャリー・アウト信号は別のロジック・エレメント (LE) のキャリー・インにのみ供給することができます。
- PLL (Phase-Locked Loop) clkout—PLL の出力クロックはタップできません。アーキテクチャの制約のため、clkout 信号はレジスタのクロック・ポートにのみ供給することができます。
- JTAG 信号—JTAG コントロール (TCK, TDI, TDO, および TMS) はタップできません。
- altgxb メガファンクション —altgxb インスタンス化のポートは直接タップできません。
- LVDS—シリアルライザ/デシリアルイザ (SERDES) ブロックからのデータ出力はタップできません。

プラグインによる信号の追加

Node Finder により個別のまたはグループ化された信号を追加する代わりに、プラグインを使用して特定タイプの IP に関連する信号グループを追加することができます。SignalTap II ロジック・アナライザには、Nios II プロセッサ用のプラグインが既にインストールされています。プラグインは信号を簡単に追加する機能以外に、トリガの作成とデータ表示に役立つ設計済みモニター・テーブル、キャプチャしたデータのコードの逆アセンブル機能など、他の多数の機能も提供します。


例えば、以下に示すように、Nios II プラグインは **Setup** タブで 1 つのモニター・テーブル、**Data** タブで 2 つのテーブルを作成します。

- **Nios II Instruction** (**Setup** タブ) — 選択した命令アドレスでのトリガに必要なすべての信号をキャプチャします。
- **Nios II Instance Address** (**Data** タブ) — 実行された命令のアドレスを 16 進数、またはオプションの実行およびリンク可能形式 (.elf) ファイルで定義される場合はプログラミング・シンボル名として表示します。
- **Nios II Disassembly** (**Data** タブ) — 対応するアドレスから逆アセンブルしたコードを表示します。

プラグインのその他の機能については、13-31 ページの「トリガの定義」および 13-54 ページの「キャプチャしたデータの表示、解析、および使用」を参照してください。

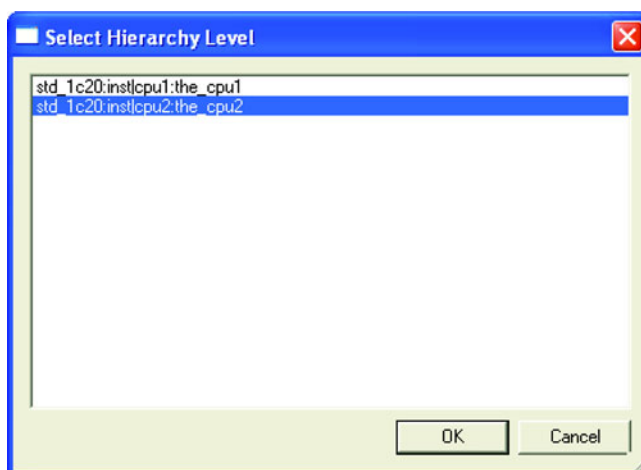
プラグインを使用して信号を SignalTap II ファイルに追加するには、デザインで解析およびエラーボレーションを実行した後で以下のステップを実行します。

1. ノード・リスト内で右クリックします。**Add Nodes with Plug-In** サブメニューで、含まれている **Nios II** など、使用するプラグイン名をクリックします。


 選択したプラグインの IP (Intellectual Property) がデザインに存在しない場合、選択したプラグインを使用できないことを知らせるメッセージが表示されます。

2. デザインの IP 階層を示す **Select Hierarchy Level** ダイアログ・ボックスが表示されます (図 13-8)。プラグインでモニタする信号が含まれている **IP** を選択して、**OK** をクリックします。

図 13-8.IP 階層の選択



3. プラグインのすべての信号が利用可能な場合は、選択したプラグインに応じてダイアログ・ボックスが表示されるため、プラグインに対して使用可能なオプションを設定できます。Nios II プラグインでは、オプションで Nios II 統合開発環境 (IDE) ソフトウェア・デザインからプログラム・シンボルを含む実行およびリンク可能形式 (.elf) ファイルを選択することができます。必要に応じて選択したプラグインのオプションを設定し、**OK** をクリックします。

 必要なすべての信号が利用可能なことを確認するには、Quartus II Analysis & Synthesis の設定の **Create debugging nodes for IP cores** をオンにします。

プラグインに含まれるすべての信号がノード・リストに追加されます。

サンプル容量の指定

サンプル容量とは、キャプチャ・データ・バッファで各信号に対してキャプチャおよび格納されるサンプルの数です。サンプル容量を設定するには、**Sample Depth** リストに格納する希望サンプル数を選択します。サンプル容量の範囲は 0 ~ 128K です。

デバイス・メモリ・リソースが制限されている場合、選択したサンプル・バッファ・サイズではデザインをコンパイルできない場合があります。サンプル容量を少なくしてリソースの使用率を低減します。

特定の RAM タイプへのデータのキャプチャ

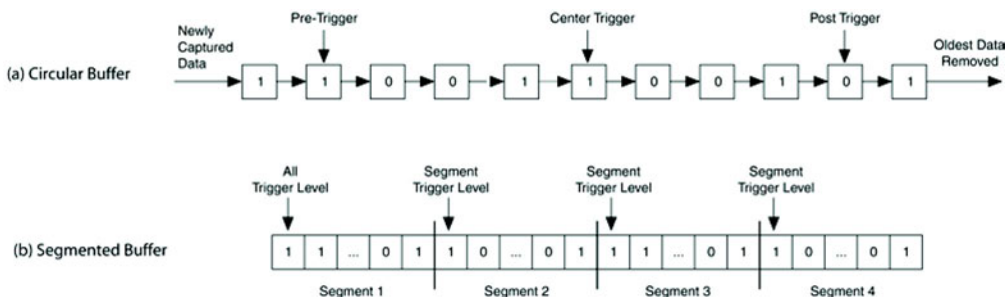
デバイスで SignalTap II ロジック・アナライザを使用する場合、オプションにより収集データを格納する RAM タイプを指定できます。RAM を選択すると、デザインの特定のメモリ・ブロックを保持し、メモリの別の領域を SignalTap II でのデータ収集に割り当てることができます。例えば、デザインでシステム・キャッシュなどの大容量バッファリングを必要とするアプリケーションを実装する場合、このアプリケーションを M-RAM ブロックに配置し、残りの M512 または M4K ブロックを SignalTap II でのデータ収集に使用できます。

SignalTap II バッファに使用する RAM タイプを選択するには、**RAM type** リストから選択します。この機能は、収集したデータ (SignalTap II Resource Estimator でレポートされる) が FPGA で選択したメモリ・タイプの使用可能なメモリ容量以下の場合に使用します。

バッファ収集モードの選択

SignalTap II ロジック・アナライザのバッファ収集タイプ選択機能により、キャプチャしたデータ・バッファの編成方法を選択し、SignalTap II でのデータ収集に必要なメモリ容量を潜在的に低減することができます。トリガ発生前後にキャプチャするデータ容量を指定する循環バッファ、または周期的に発生するトリガをキャプチャするのに役立つセグメント・バッファのいずれかを選択できます。図 13-9 に、この 2 つのバッファ・タイプの違いを示します。

図 13-9. SignalTap II ロジック・アナライザで使用するバッファ・タイプの比較



循環バッファ

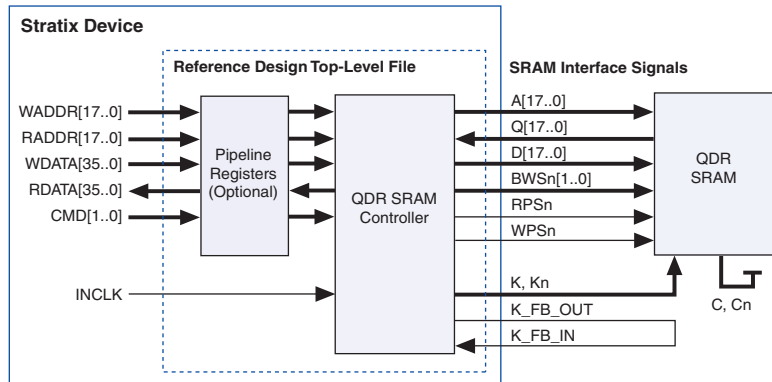
循環バッファ (図 13-9 (a)) は、SignalTap II ロジック・アナライザで使用するデフォルトのバッファ・タイプです。ロジック・アナライザの動作中、データはバッファが満杯になるまでバッファに格納され、満杯になると新しいデータが古いデータに置き換わります。指定したトリガ・イベントが発生するまでこれが継続されます。トリガ・イベントが発生すると、ロジック・アナライザはトリガ・イベント後も SignalTap II ファイルの Signal Configuration ペインの循環バッファ・トリガ位置の設定に基づき、バッファが満杯になるまでデータのキャプチャを継続します。リストから設定を選択し、データの大部分をトリガ発生前 (**Post trigger position**) または発生後 (**Pre trigger position**) にキャプチャするか、またはトリガ位置をデータの中央に (**Center trigger position**) するかを選択します。また、ロジック・アナライザが停止するまでデータ・キャプチャの継続を選択することもできます。

詳しくは、13-39 ページの「トリガ位置の指定」を参照してください。

セグメント・バッファ

セグメント・バッファ (図 13-9 (b)) は、バッファを複数の同一サイズの独立したセグメントで編成されています。このバッファ構造のタイプにより、あまり頻繁に発生しない周期的イベントを含むシステムをより簡単にデバッグすることができます。図 13-10 に、このタイプのバッファ・システム例を示します。

図 13-10. 周期的イベントを生成するシステムの例



SignalTap II ロジック・アナライザは、図 13-10 に示すデザインの機能を検証して、SRAM コントローラに確実に正しいデータが書き込まれるようにします。SignalTap II ロジック・アナライザのバッファ収集機能により、H'0F0F0F0F が RADDR ポートに送信されたときに RDATA ポートをモニタすることができます。SignalTap II ロジック・アナライザを再度動作させることなく、SRAM デバイスからの複数のリード・トランザクションをモニタできます。バッファ収集機能によって、メモリをセグメント化して、割り当てられたメモリを無駄にすることなく、同じイベントを複数回キャプチャできます。キャプチャされるサイクル数は、**Signal Configuration** 設定に指定したセグメント数によって決まります。

バッファ収集をイネーブルおよびコンフィギュレーションするには、SignalTap II エディタで **Segmented** を選択し、使用するセグメント数を選択します。この例では、64 サンプル・セグメントを選択することにより、RADDR 信号が H'0F0F0F0F のときに、64 リード・サイクルをキャプチャできます。



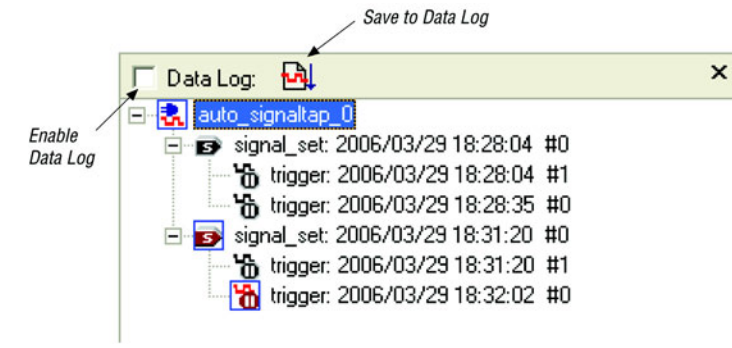
バッファ収集モードについて詳しくは、Quartus II Help の「Setting the Buffer Acquisition Mode」を参照してください。

複数の SignalTap II ファイルおよび コンフィギュレーションの管理

1 つのデザインに複数の SignalTap II ファイルが存在する場合もあります。各ファイルには潜在的に異なるモニタ信号グループがあります。これらの信号グループにより、デザイン内の異なるブロックをデバッグできます。また、各信号グループは異なるトリガ条件セットの定義にも使用できます。各 SignalTap II ファイルの他に、関連するプログラミングファイル (SRAM オブジェクト・ファイル (SOF)) もあります。選択した SignalTap II ファイルの設定は、デバイスがプログラムされたときにロジック・アナライザが正しく動作するために、関連する SOF ファイルの SignalTap II ロジック・デザインと一致しなければなりません。すべての SignalTap II ファイルとそれらの関連設定およびプログラミング・ファイルの管理は、困難を伴う作業です。これらを管理するために、**Data Log** 機能と **SOF Manager** を使用できます。

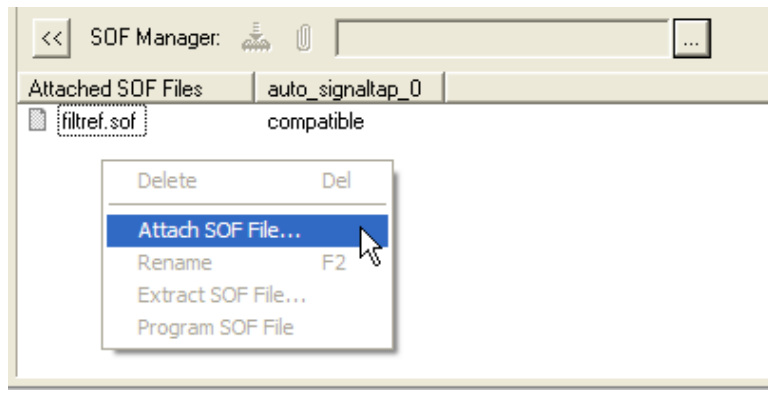
Data Log を使用して、1 つの SignalTap II ファイルに複数の SignalTap II コンフィギュレーションを格納できます。図 13-11 に、1 つの SignalTap II ファイルに複数のトリガ条件を持つ 2 つの信号セット・コンフィギュレーションを示します。アクティブ・コンフィギュレーション間でトグルするには、**Data Log** のエントリをダブル・クリックします。異なるコンフィギュレーション間でトグルすると、SignalTap II ファイルの **Setup** タブで信号リストとトリガ条件が変更されます。SignalTap II ファイルに表示されるアクティブ・コンフィギュレーションは、**Data log** で信号セットの周囲を囲む青色の四角形で示されます。データ・ログにコンフィギュレーションを保存するには、Edit メニューの **Save to Data Log** をクリックするか Data Log の一番上にある **Save to Data Log** ボタンをクリックします。

図 13-11. データ・ログ



SOF Manager を使用して、複数の SOF を 1 つの SignalTap II ファイルに埋め込むことができます。SOF を SignalTap II ファイルに埋め込むことにより、関連する SOF を個別ファイルとして含める必要なく、SignalTap II ファイルを同じコンピュータまたはネットワーク上の別のコンピュータに移動できます。新しい SOF を SignalTap II ファイルに埋め込むには、SOF Manager で右クリックし、**Attach SOF File** (図 13-12) をクリックします。

図 13-12. SOF Manager



Data Log でコンフィギュレーションを切り替えると、特定のコンフィギュレーションと互換性のある SOF を抽出し、SignalTap II ロジック・アナライザでプログラマウを使用して、新しい SOF を FPGA にダウンロードすることができます。このようにして、SignalTap II ファイルのコンフィギュレーションを常にターゲット・デバイスにプログラムされたデザインと一致させることができます。

トリガの定義

必要なとき必要なデータをキャプチャするには、モニタしている信号がそのデータを表示する条件を指定する必要があります。SignalTap II ロジック・アナライザでは、これらの条件を従来の外部ロジック・アナライザやオシロスコープでの呼び名と同様に、トリガと呼びます。デバッグに役立つように、さまざまなタイプのトリガを生成するためのオプションが多数あります。

基本トリガの生成

使用可能な最もシンプルな種類のトリガが基本トリガです。このトリガは、SignalTap II エディタのノード・リストにある **Trigger Levels** カラムの一番上のリストから選択します。トリガ・タイプを **Basic** に設定した場合、SignalTap II ファイルに追加した各信号のトリガ・パターンを設定する必要があります。トリガ・パターンを設定するには、**Trigger Levels** カラムで右クリックし、希望のパターンをクリックします。トリガ・パターンは以下の条件のいずれかに設定できます。

- Don't Care
- Low
- High
- Falling Edge
- Rising Edge
- Either Edge

バスの場合、パターンをバイナリ形式で入力するか、右クリックして **Insert Value** を選択し、他の形式でパターンを入力できます。関連するニーマニック・テーブルを持つ SignalTap II ファイルに追加される信号の場合は、右クリックしてテーブルのエントリを選択して、定義済みのトリガ条件を設定できます。

ニーマニック・テーブルの作成および使用について詳しくは、[13-54 ページの「キャプチャしたデータの表示、解析、および使用」](#)および [Quartus II Help](#) を参照してください。

特定のプラグインを使用して追加された信号の場合、定義済みニーマニック・テーブルのエントリを使用して簡単に基本トリガを生成できます。例えば、Nios II IDE デザインから実行可能なソフトウェア (.elf) ファイルを指定した場合は、Nios II プラグインを使用して、Nios II コードからファンクション名を入力することができます。Nios II 命令アドレスが指定されたコードのファンクション名のアドレスと一致すると、ロジック・アナライザがトリガします。

あるトリガ・レベルに対してすべての信号の論理 AND が TRUE の場合、データのキャプチャが停止し、データはバッファに保存されます。

拡張トリガの生成

SignalTap II トリガ・タイプを SignalTap II エディタのノード・リスト内の **Trigger Levels** カラムの一番上にある **Advanced** に設定すると、**Advanced Trigger** という新しいタブ名が表示されるため、シンプルな GUI を使用して複雑なトリガ式を作成できます。Advanced Trigger Configuration Editor 配置した演算子をダブル・クリックするか右クリックして Properties を選択し、演算子の設定をコンフィギュレーションします。表 13-5 に使用できる演算子を示します。

演算子の名前	タイプ
Less Than	比較
Less Than or Equal To	比較
Equality	比較
Inequality	比較
Greater Than	比較
Greater Than or Equal To	比較
Logical NOT (論理 NOT)	論理
Logical AND (論理 AND)	論理
Logical OR (論理 OR)	論理
Logical XOR (論理 XOR)	論理
Reduction AND (リダクション AND)	リダクション
Reduction OR (リダクション OR)	リダクション
Reduction XOR (リダクション XOR)	リダクション
Left Shift	シフト
Right Shift	シフト
Bitwise Complement	ビット単位
Bitwise AND (ビット AND)	ビット単位
Bitwise OR (ビット OR)	ビット単位
Bitwise XOR (ビット XOR)	ビット単位
エッジおよびレベル検出器	信号検出
連続カウンタ	カウンタ
ステート・カウンタ	カウンタ
イベント・カウンタ	カウンタ

表 13-5 の注：

(1) これらの演算子の詳細は、Quartus II Help を参照してください。

特定の演算子の設定には、実行時にコンフィギュレーションできるものがあります。これにより、演算子のタイプを変更したりデザインをリコンパイルすることなく、演算子の他の設定を調整することができます。演算子記号の背景が白い演算子設定は、デザインをリコンパイルしないで変更することができます。

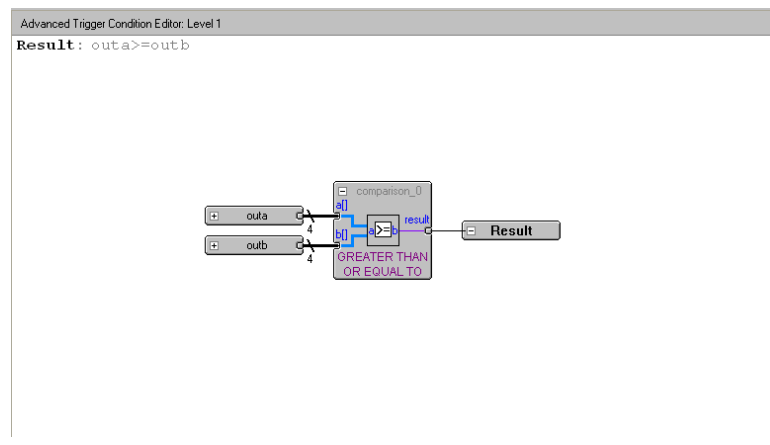
多数のオブジェクトを **Advanced Trigger Condition** エディタに追加すると、作業領域が乱雑になり読みにくくなります。拡張トリガ条件の作成中にオブジェクトの編成を維持するには、メニューを右クリックして **Arrange All Objects** を選択します。また、**Zoom-Out** コマンドを使用して、より多くのオブジェクトを **Advanced Trigger Condition** エディタのウィンドウに収めることができます。

拡張トリガ式の例

以下の例は、拡張トリガの使用方法を示します。

- バス **outa** が **outb** より大きいか等しいとき、トリガします (図 13-13)。

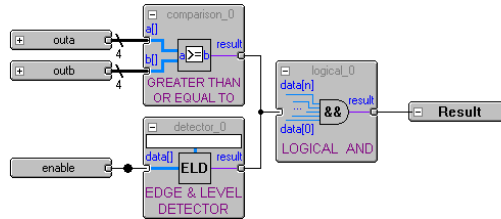
図 13-13. バス **outa** が **outb** より大きいか等しい



- バス **outa** がバス **outb** より大きいか等しく、かつイネーブル信号に立ち上がりエッジがある場合にトリガします (図 13-14)。

図 13-14. イネーブル信号に立ち上がりエッジがある

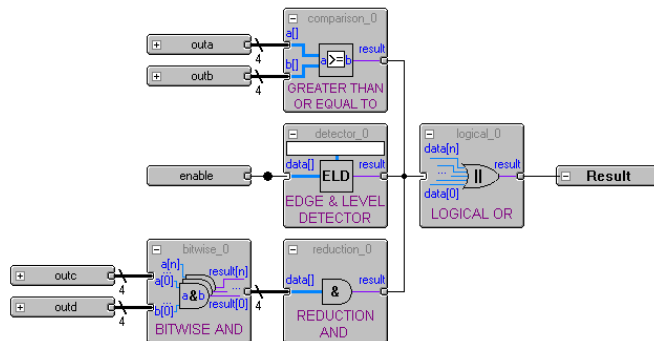
```
Result: outa>=outb&&ELD({enable})
```



- バス outa がバス outb より大きいか等しい、あるいはイネーブル信号に立ち上がりエッジがある場合にトリガします。または、ビット単位 AND 演算がバス outc とバス outd の間で実行され、その演算の結果の全ビットが 1 と等しい場合はトリガします (図 13-15)。

図 13-15. ビット単位 AND 演算

```
Result: outa>=outb|ELD({enable})||(&outc&outd)
```



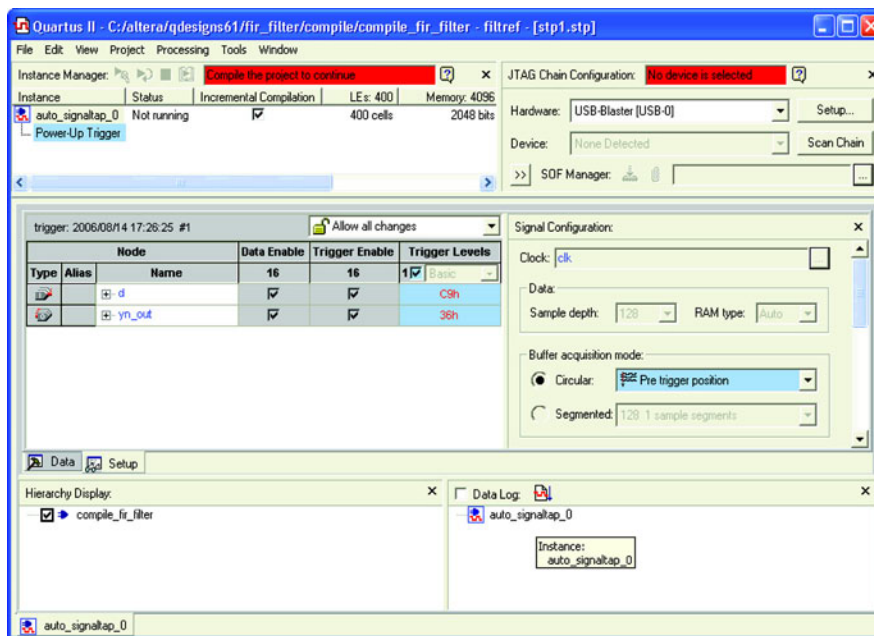
パワーアップ・トリガの作成

通常、SignalTap II ロジック・アナライザは、通常のデバイス動作中に発生するイベントでトリガするように使用されます。ターゲット・デバイスが完全に起動し、デバイスの JTAG 接続が使用可能な状態になると手動で解析を開始します。しかし、FPGA のパワーオンまたはリセット直後のデバイス初期化中に発生するトリガ・イベントをキャプチャしたい場合があります。SignalTap II パワーアップ・トリガ機能により、デバイス・プログラミング後にロジック・アナライザを手動で起動する前に発生するトリガからデータをキャプチャすることができます。

パワーアップ・トリガのイネーブル

SignalTap II Instance Manager で、別のパワーアップ・トリガをロジック・アナライザの各インスタンスに追加することができます。ロジック・アナライザのインスタンスでパワーアップ・トリガをイネーブルするには、インスタンスを右クリックして、**Enable Power-Up Trigger** をクリックするか、インスタンスを選択して Edit メニューの **Enable Power-Up Trigger** をクリックします。パワーアップ・トリガをディセーブルするには、同じ場所にある **Disable Power-Up Trigger** をクリックします。**Power-Up Trigger** は、ノード・リストで設定したデフォルト・トリガ条件を持つ選択したインスタンスの名前に基づく子インスタンスとして表示されます。図 13-16 に、パワーアップ・トリガがイネーブルされているときの SignalTap II エディタを示します。

図 13-16. パワーアップ・トリガがイネーブルされているときの SignalTap II エディタ



パワーアップおよびランタイム・トリガ条件の管理と コンフィギュレーション

ロジック・アナライザのインスタンスに対してパワーアップ・トリガがイネーブルされたら、ランタイムと呼ばれる通常のトリガと同じ方法で、それに対する基本および拡張トリガ条件を作成します。ランタイム・トリガ条件は白色のままですが、調整可能なパワーアップ・トリガ条件は水色で表示されます。各インスタンスにはパワーアップ・トリガとランタイム・トリガの2セットのトリガ条件があるため、色分けして区別することができます。パワーアップ・トリガとランタイム・トリガのトリガ条件を切り替えるには、Instance Manager でインスタンス名またはパワーアップ・トリガをダブル・クリックします。

パワーアップ・トリガのトリガ条件を設定するときは、ランタイム・トリガの条件に変更を加える場合に通常コンパイルを必要としない変更のみ行うことができます。信号の追加、削除、または基本トリガと拡張トリガの切り替えなど、フル・コンパイルを必要とする変更を行うことはできません。これらのアクションを実行するには、ランタイム・トリガ条件に切り替えます。しかし、パワーアップ・トリガ条件を変更する

には、常に SignalTap II ロジック・アナライザをリコンパイルする必要があり、同様の変更をランタイム・トリガに加えた場合にリコンパイルが不要な場合でも必要です。

ランタイム・トリガまたはパワーアップ・トリガのトリガ条件を作成または変更する場合、これらの条件を他のトリガにコピーすることができます。これにより、パワーアップ時および動作中に同じトリガを簡単に検索できるようになります。これを行うには、Instance Manager でインスタンス名またはパワーアップ・トリガ名を右クリックして、**Duplicate Trigger** をクリックするか、インスタンス名またはパワーアップ・トリガ名を選択して、Edit メニューの **Duplicate Trigger** をクリックします。

パワーアップ・トリガをイネーブルした状態で SignalTap II ロジック・アナライザのインスタンスを実行する方法については、13-51ページの「[パワーアップ・トリガの実行](#)」を参照してください。

複数のトリガ・レベルの使用

複数のトリガ・レベル機能で、作成したトリガ条件を正確に制御することができます。この機能により、ロジック・アナライザでより複雑なデータ・キャプチャ・コマンドを使用して、精度を向上させ問題を分離することができます。最大 10 レベルのトリガ条件を設定できます。

SignalTap II ロジック・アナライザは、最初にトリガ・レベル 1 に関連するトリガ・パターンを評価します。トリガ・レベル 1 の式が TRUE の場合、ロジック・アナライザはトリガ・レベル 2 の式を評価します。このプロセスは、すべてのトリガ・レベルが処理され、最後のトリガ・レベルが TRUE になるまで継続します。基本トリガ、拡張トリガ、または両方を組み合わせた複数のトリガ・レベル機能を使用できます。

ランタイム・トリガ条件を表示する場合、**Trigger Levels** リストで必要な数のトリガ・レベルを選択します。パワーアップ・トリガを表示する場合は、トリガ・レベル数を調整することはできません。この変更を行うには、ランタイム・トリガ条件に切り替えます。ランタイム・トリガとパワーアップ・トリガ間のトリガ・レベル数は常に同じです。ランタイム・トリガまたはパワーアップ・トリガで余分なトリガ・レベルが必要ない場合は、トリガ・レベルをディセーブルします。トリガ・レベルをディセーブルするには、ノード・リストのカラム上部のチェック・ボックスをクリックして、希望の **Trigger Level** カラムをオフにします。

トリガ位置の指定

循環バッファ収集モードを使用する場合、トリガ・イベントの前後に取得するデータ量を指定できます。ランタイム・トリガとパワーアップ・トリガでトリガ位置を個別に設定できます。以下の比率のいずれかを選択して、希望のトリガ前データ / トリガ後データの比率を選択します。

- **Pre**— トリガ後 (12% トリガ前、88% トリガ後) に発生した信号アクティビティを保存します。
- **Center**— 50%トリガ前データおよび50%トリガ後のデータを保存します。
- **Post**— トリガ前 (88% トリガ前、12% トリガ後) に発生した信号アクティビティを保存します。
- **Continuous**— 信号アクティビティを (手動で停止するまで) 保存し続けます。

外部トリガの使用

外部ソースから SignalTap II ロジック・アナライザのトリガを可能にするトリガ入力を作成できます。外部トリガ入力はトリガ・レベル 10 と同様に動作します。外部トリガは、他のコンフィギュレーションしたトリガ・レベルが評価される前に評価し、結果が TRUE でなければなりません。アナライザは外部デバイスまたは他の SignalTap II インスタンスをトリガする信号も供給できます。これらの機能により、外部ロジック解析装置を内部ロジック・アナライザに同期させることができます。パワーアップ・トリガでは外部トリガ機能を使用することができますが、関連するランタイム・トリガと同じソースまたはターゲット信号を使用する必要があります。

Trigger In

Trigger In を使用するには、以下のステップを実行します。

1. SignalTap II エディタの **Setup** タブをクリックします。
2. パワーアップ・トリガがイネーブルされている場合、ランタイム・トリガ条件が表示されていることを確認します。
3. Signal Configuration ペインの **Trigger In** をオンにします。
4. **Pattern** リストで、トリガ・イベントとして使用する条件を選択します。これをランタイム・トリガまたはパワーアップ・トリガで別々に設定できます。

5. **Trigger In** ペインの **Source** フィールド横の **Browse** をクリックします (13-42 ページの図 13-18 を参照してください)。 **Node Finder** ダイアログ・ボックスが表示されます。
6. **Node Finder** ダイアログ・ボックスで、**Trigger In** ソースをドライブする信号 (入力ピンまたは内部信号) を選択して、**OK** をクリックします。

Source フィールドに新しい信号名を入力すると、ピン・プランナまたはアサインメント・エディタで入力ピンに割り当てることができる新しいノードが作成されます。**Source** フィールドを空白のままにした場合は、`auto_stp_trigger_in_<SignalTap インスタンス番号>` の形式でデフォルト名が入力されます。

Trigger Out

Trigger Out を使用するには、以下のステップを実行します。

1. **SignalTap II** エディタの **Setup** タブをクリックします。
2. パワーアップ・トリガがイネーブルされている場合、ランタイム・トリガ条件が表示されていることを確認します。
3. **Signal Configuration** ペインの **Trigger Out** をオンにします。
4. **Level** リストで、トリガ・イベントが発生していることを示す条件を選択します。これをランタイム・トリガまたはパワーアップ・トリガで別々に設定できます。
5. **Target** フィールドに、新しい信号名を入力します。ピン・プランナまたはアサインメント・エディタで出力ピンに割り当てた新しいノード名が作成されます。

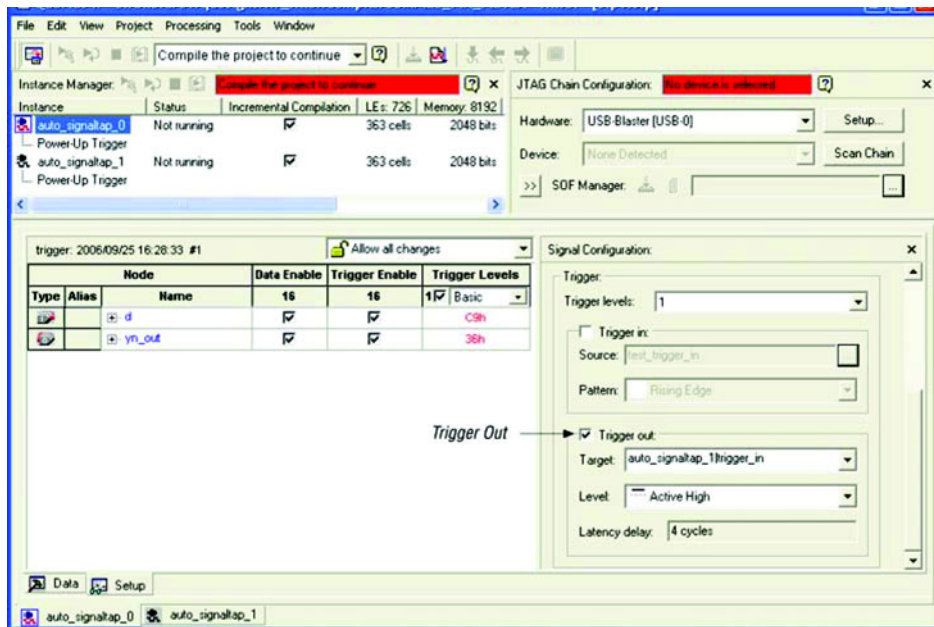
Target フィールドを空白のままにした場合は、`auto_stp_trigger_out_<SignalTap インスタンス番号>` の形式でデフォルト名が入力されます。ロジック・アナライザがトリガすると、指定したレベルの信号が新しいノードに割り当てたピンに出力されます。

あるアナライザの Trigger Out を別のアナライザの Trigger In として使用する

SignalTap II ロジック・アナライザでは、最新機能の 1 つとしてあるアナライザの Trigger Out を別のアナライザへの Trigger In として使用できます。この機能により、複数のクロック・ドメインで発生するイベントを同期させデバッグすることができます。

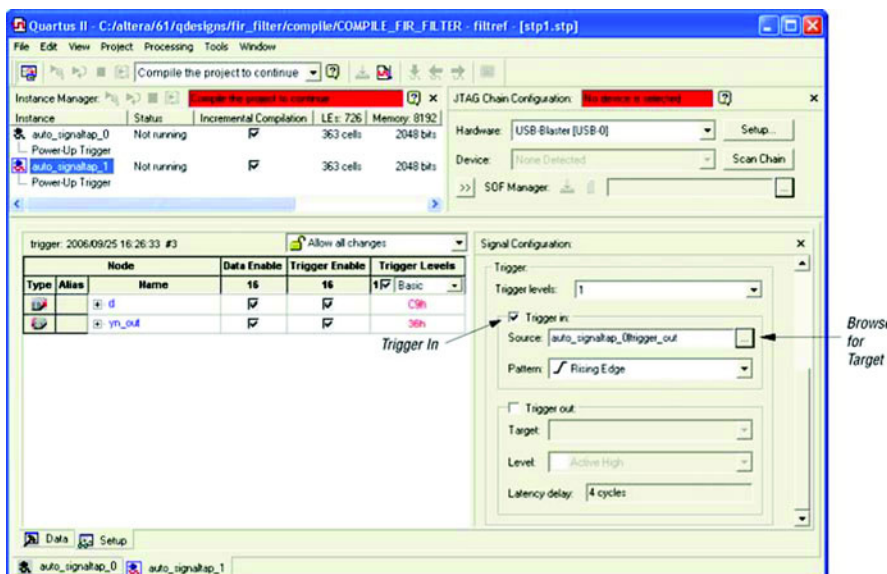
これを実行するには、最初にソース・ロジック・アナライザ・インスタンスの **Trigger Out** をイネーブルします。Trigger out **Target** リストで、ターゲットとするロジック・アナライザのインスタンスを選択します。例えば、auto_signaltap_0 という名前のインスタンスで auto_signaltap_1 をトリガする場合、リストから auto_signaltap_1|trigger_in を選択します (図 13-17)。

図 13-17. Trigger Out 信号のイネーブル



ターゲットとするロジック・アナライザのインスタンスの **Trigger In** を自動的にイネーブルし、ソース・ロジック・アナライザのインスタンスからの **Trigger Out** 信号を **Trigger In Source** フィールドに入力します。この例では、auto_signaltap_0 は auto_signaltap_1 をターゲットにしています。auto_signaltap_1 の **Trigger In Source** フィールドには auto_signaltap_0|trigger_out が自動的に入力されます (図 13-18)。

図 13-18. Trigger In 信号のイネーブル



デザインの コンパイル

SignalTap II ファイルをプロジェクトに追加すると、SignalTap II ロジック・アナライザはデザインの一部になります。このため、プロジェクトを SignalTap II ロジックを含めてコンパイルし、ロジック・アナライザを制御するのに使用する JTAG 接続をイネーブルする必要があります。外部ロジック・アナライザを使用してデバッグを行う場合、モニタする信号やトリガ条件を変更しなければならぬこともよくあります。SignalTap II ロジック・アナライザを使用するとき、これらのタイプの調整はリコンパイル時間になるため、Quartus II ソフトウェアのインクリメンタル・コンパイルと併せて、SignalTap II インクリメンタル・コンパイル機能を使用してリコンパイル時間を短縮することができます。

SignalTap II インクリメンタル・コンパイルを使用したコンパイルの高速化

SignalTap II ファイルを使用してデザインをコンパイルする場合、`sld_signaltap`および`sld_hub`エンティティがコンパイル階層に自動的に追加されます。これら 2 つのエンティティは SignalTap II ロジック・アナライザの主要コンポーネントで、動作に必要なトリガ・ロジックと JTAG インタフェースを提供します。

SignalTap II インクリメンタル・コンパイルでは、オリジナル・デザインの合成およびフィッティングの結果を保持し、オリジナル・ソース・コードをリコンパイルすることなく、デザインに SignalTap II ロジック・アナライザを追加することができます。また、この機能は SignalTap II ファイルのコンフィギュレーションを変更する場合にも役立ちます。例えば、バッファのサンプル容量またはメモリ・タイプを、変更後にフル・コンパイルを実行することなく修正できます。変更を反映させるには、独自のデザイン・パーティションとしてコンフィギュレーションされた SignalTap II ロジック・アナライザのみリコンパイルします。

SignalTap II インクリメンタル・コンパイルを使用するには、最初にデザインに対してフル・インクリメンタル・コンパイルをイネーブルし（まだイネーブルされていない場合）、必要に応じてデザイン・パーティションを割り当てます。Quartus II ソフトウェアでは、新規プロジェクトに対してはインクリメンタル・コンパイルがデフォルト設定なので、新規プロジェクトではデザイン・パーティションをすぐに確立することができます。ただし、SignalTap II インクリメンタル・コンパイル機能を使用するために、デザイン・パーティションを作成する必要はありません。フル・インクリメンタル・コンパイルを使用するようにデザインをセットアップしたら、SignalTap II ファイルで SignalTap II インクリメンタル・コンパイル機能をイネーブルします。これによって、SignalTap II ロジック・アナライザが独自の個別デザイン・パーティションに変換されます。

デザインでのインクリメンタル・コンパイルのイネーブル

インクリメンタル・コンパイルをイネーブルする（まだ、イネーブルされていない場合）には、以下のステップを実行します。

1. Assignments メニューの **Design Partitions** ウィンドウをクリックします。
2. **Incremental Compilation** リストで、**Full Incremental Compilation** を選択します。

3. 必要の場合はユーザ定義のパーティションを作成し、**Netlist Type** をすべてのパーティションに対して **Post-fit** に設定します。
4. **Processing** メニューの **Start Compilation** をクリックするか、またはツールバーの **Start Compilation** をクリックします。

プロジェクトが一度で完全にコンパイルされ、作成したデザイン・パーティションが確立されます。デザインに対して **SignalTap II** ファイルをすでにイネーブルしていても、**SignalTap II** ファイルでインクリメンタル・コンパイルをイネーブルしていない場合は、**SignalTap II** 設定が変更されているため、以降のリコンパイルでインクリメンタル・コンパイルによるコンパイル時間の短縮を実現できません。次の項で説明しますが、**SignalTap II** ロジック・アナライザが個別のデザイン・パーティションとして確立されるまでインクリメンタル・コンパイル機能を使用することはできません。



インクリメンタル・コンパイルのコンフィギュレーションおよび実行について詳しくは、「Quartus II ハンドブック Volume 1」の「Quartus II Incremental Compilation for Hierarchical & Team-Based Design」の章を参照してください。

SignalTap II インクリメンタル・コンパイルのイネーブル

SignalTap II インクリメンタル・コンパイルをイネーブルするには、以下のステップを実行します。

1. **SignalTap II** ファイルの各インスタンスは、**Instance Manager** の **Incremental Compilation** をオンにして、それぞれを個々のデザイン・パーティションに変換します。

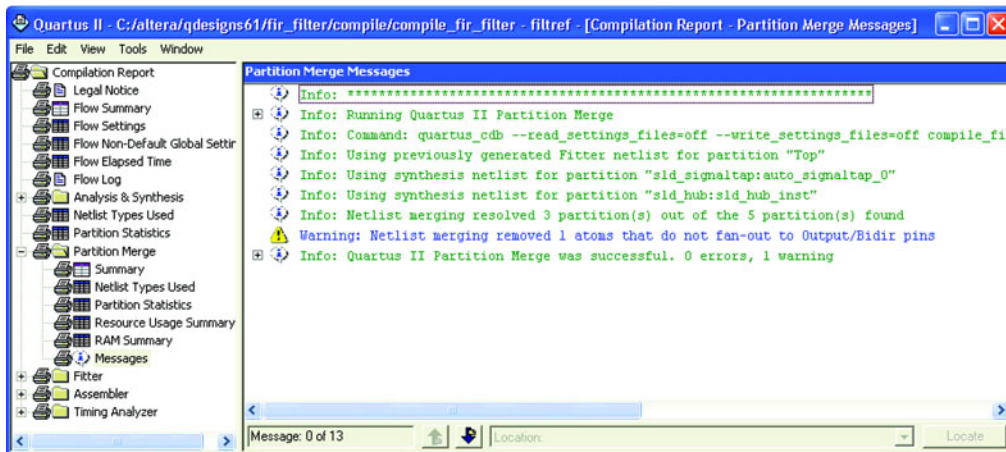


インクリメンタル・コンパイルをイネーブルすると、既存のすべての合成前信号が対応するフィッティング後信号に置き換わります。**SignalTap II** インクリメンタル・コンパイル機能は、フィッティング後信号に対してのみ使用できます。対応するフィッティング後信号がないか、または合成属性付きで保持されていない合成前信号が無効になり、信号名の色が赤に変化してそれが示されます。無効な信号をノード・リストから削除しないと、コンパイルを正しく実行できません。

2. 他の必要な **SignalTap II** のフィッティング後ノードを **SignalTap II** ファイルに追加します。
3. **Processing** メニューの **Start Compilation** をクリックするか、またはツールバーの **Start Compilation** をクリックします。

オリジナルのデザインが変更されていないことを確認するには、Compilation Report の Partition Merge セクションのメッセージを調べます。図 13-19 に、表示されるメッセージの例を示します。

図 13-19. コンパイル・レポート・メッセージ



デザイン・パーティションにリコンパイルを必要とする変更を行っていない場合、SignalTap II のデザイン・パーティションのみリコンパイルされます。それ以降 SignalTap II ファイルのみ変更する場合、SignalTap II デザイン・パーティションのみリコンパイルして、リコンパイル時間を短縮します。

インクリメンタル・コンパイルを使用しないコンパイル

SignalTap II ファイルをコンフィギュレーションし、トリガを定義したら、プロジェクトをコンパイルして SignalTap II ロジック・アナライザを取り込みます。Processing メニューの **Start Compilation** をクリックするか、またはツールバーの **Start Compilation** をクリックしてコンパイルを開始します。SignalTap II の設定にリコンパイルを必要とする変更を行った場合は、プロジェクト全体をリコンパイルする必要があります。

表 13-6 に、プロジェクトおよび SignalTap II ロジック・アナライザに適用可能なインクリメンタル・コンパイル設定をまとめます。

Quartus II の インクリメンタル・ コンパイル	SignalTap II の インクリメンタル・ コンパイル	動作
ディセーブル	ディセーブル	インクリメンタル・コンパイルなし SignalTap II の設定にリコンパイルを必要とする変更を行った場合、プロジェクト全体をリコンパイルしなければなりません。
ディセーブル	イネーブル	プロジェクトのコンパイルが失敗します。プロジェクトのフル・インクリメンタル・コンパイルをイネーブルして続行します。
イネーブル	ディセーブル	プロジェクトは最初に正常にコンパイルされ、SignalTap II ロジック・アナライザ用ロジックを組み込むデザイン・パーティションを確立します。これ以降コンパイルは正常に実行されますが、デザイン・パーティションをリコンパイルして SignalTap II 設定の変更を反映させる必要があります。アルテラでは、SignalTap II インクリメンタル・コンパイルをオンにして、リコンパイル時間を短縮することを推奨しています。
イネーブル	イネーブル	フル・インクリメンタル・コンパイル。SignalTap II 設定にリコンパイルを必要とする変更を行った場合、SignalTap II デザイン・パーティションのみリコンパイルしなければなりません。

リコンパイルを必要とする変更の防止

SignalTap II ファイルをコンフィギュレーションして、通常はリコンパイルを必要とする変更を防止することができます。これを行うには、**Setup** タブのノード・リストの上部からロック・モードを選択します。インクリメンタル・コンパイルを使用するか否かに係わらず、トリガ条件の変更のみ許可することでコンフィギュレーションをロックすることができます。



ロック・モードの使用について詳しくは、Quartus II Help を参照してください。

SignalTap II ロジック・アナライザでのタイミングの保持

機能の検証に加え、タイミング・クロージャはデザインを完了するうえで最も重要なプロセスの1つです。SignalTap II ロジック・アナライザでインクリメンタル・コンパイルを使用しないでプロジェクトをコンパイルする場合、既存のデザインに IP を追加します。したがって、デザインの既存の配置、配線、およびタイミングが影響を受ける可能性があります。SignalTap II ロジック・アナライザのデザインへの影響を最小限に抑えるために、アルテラではプロジェクトにインクリメンタル・コンパイルの使用を推奨しています。新規デザインでは、インクリメンタル・コンパイルがデフォルト設定であり、既存のデザインで簡単にイネールおよびコンフィギュレーションすることができます。独自のデザイン・パーティションで SignalTap II ロジック・アナライザを使用する場合、デザインにはほとんどまたはまったく影響はありません。デザインにインクリメンタル・コンパイルを使用できない場合、SignalTap II ロジック・アナライザをインサートする前にデザインをバック・アノテートします。バック・アノテーションにより、ロジック・アナライザがデザインの性能に与える影響を最小限に抑えることができます。

デザインのバック・アノテーションの他に、以下の手法を使用してタイミングを維持することができます。

- SignalTap II ファイルへのクリティカル・パス信号追加を回避
- Trigger Enable が選択された信号数を少なくします。デフォルトでは、SignalTap II ファイルに追加するすべての信号で Trigger Enable がオンになっています。トリガとして使用しない信号の Trigger Enable はオフにします。
- SignalTap II ファイルに追加する組み合わせ信号数を少なくし、可能な場合はレジスタを追加します。
- デザインの各クロックに対して f_{MAX} 制約を指定します。
- SignalTap II ロジック・アナライザでコンパイルを行う前にデザインをバック・アノテートします。



SignalTap II ロジック・アナライザでのタイミング保持の例については、「Quartus II ハンドブック Volume 2」の「エリアおよびタイミングの最適化」の章を参照してください。

ターゲット・ デバイスのプ ログラミング

SignalTap II ロジック・アナライザを含めて、プロジェクトをコンパイルした後は、FPGA ターゲット・デバイスをコンフィギュレーションする必要があります。SignalTap II ロジック・アナライザをデバッグに使用するときは、Quartus II Programmer ではなく SignalTap II ファイルからデバイスをコンフィギュレーションすることができます。SignalTap II

ファイルからコンフィギュレーションを行うので、複数の SignalTap II ファイルを開いて、複数のデバイスをプログラムし、複数のデザインを同時にデバッグすることができます。

SignalTap II ファイルの設定は、デバイスをプログラムするのに使用するプログラミング (SOF) ファイルと互換性がなければなりません。ロジック・アナライザの設定 (キャプチャ・バッファのサイズやモニタまたはトリガに選択された信号など) が、ターゲット・デバイスがプログラムされる方法と一致するときには、SignalTap II ファイルは SOF と互換性があると見なされます。ファイルに互換性がない場合でも、デバイスをプログラムすることはできますが、SignalTap II エディタからロジック・アナライザを実行または制御することはできません。

プログラミングの互換性を確保するために、必ず直近のコンパイルで生成された最新の SOF を使用してデバイスをプログラムします。

デバッグ・セッションを開始する前は、SignalTap II ファイルの設定に対してプロジェクトのリコンパイルを必要とする変更を行わないようにします。Instance Manager 上部の SignalTap II ステータス・ディスプレイをチェックして、変更した設定にデザインのリコンパイルが必要かどうか確認し、新しい SOF を生成することができます。これにより、変更を取り消してリコンパイルを不要にする機会が与えられます。このような変更を防止するには、SignalTap II ファイルでロック・モードをイネーブルします。

単一デバイスのプログラミング


SignalTap II ロジック・アナライザで使用するために単一デバイスをコンフィギュレーションするには、以下のステップを実行します。

1. SignalTap II エディタの **JTAG Chain Configuration** ペインで、**Hardware** リストのデバイスとの通信に使用する接続を選択します。通信ケーブルをリストに追加する必要がある場合は、**Setup** をクリックして接続をコンフィギュレーションします。
2. **JTAG Chain Configuration** ペインで **Browse** をクリックして、SOF file that includes the compatible SignalTap II Logic Analyzer を選択します。
3. **Scan Chain** をクリックします。
4. **Device** リストで、デザインをダウンロードするデバイスを選択します。
5. **Program Device** アイコンをクリックします。

複数のデバイスのプログラミングおよび複数のデザインのデバッグ

Quartus II ソフトウェアの 1 つのインスタンスを使用して同時に複数のデザインをデバッグするには、以下のステップを実行します。

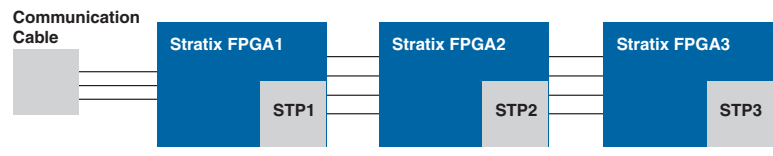
1. SignalTap II ファイルを含む各プロジェクトを作成・設定して、コンパイルします。
2. 各 SignalTap II ファイルを開きます。

 SignalTap II ファイルを開くために、Quartus II プロジェクトを開く必要はありません。

3. **JTAG Chain Configuration** パネル・コントロールを使用して、各 SignalTap II ファイルでターゲット・デバイスを選択します。
4. 各 FPGA をプログラムします。
5. 各アナライザを個別に実行します。

図 13-20 に、JTAG チェインおよび関連する SignalTap II ファイルを示します。

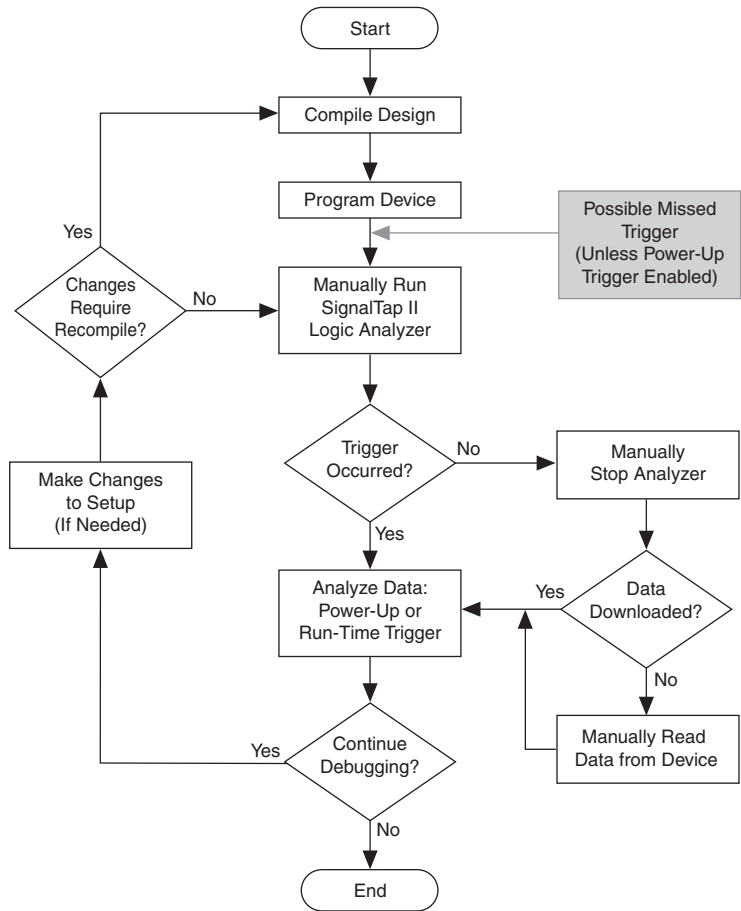
図 13-20. JTAG チェイン



SignalTap II ロジック・ アナライザの 実行

SignalTap II ロジック・アナライザを含むデザインでデバイスをコンフィギュレーションした後で、外部ロジック・アナライザを使用する場合と同様の方法でデバッグ操作を実行することができます。解析を開始してロジック・アナライザを「準備」します。トリガ・イベントが発生すると、キャプチャしたデータがデバイスのメモリ・バッファに保存され、JTAG 接続を介して SignalTap II ファイルに転送されます。また、トリガ・イベントを発生させないで、現在バッファにあるキャプチャしたデータを表示可能な「強制トリガ」と同等機能を実行することも可能です。図 13-21 に、SignalTap II ロジック・アナライザの操作方法のフローを示します。フローチャートは、パワーアップおよびランタイム・トリガ・イベントの発生箇所とこれらのイベントからキャプチャしたデータが解析に利用可能になるタイミングを示します。

図 13-21. パワーアップおよびランタイム・トリガ・イベントのフローチャート



Instance Manager の SignalTap II ツールバーには、アナライザを動作させるための 4 つのオプションがあります。

- **Run Analysis**—SignalTap II ロジック・アナライザは、トリガ・イベントが発生するまで動作します。トリガ・イベントが発生し、アクイジション・バッファがいっぱいになるとモニタおよびデータ・キャプチャが停止します。
- **AutoRun Analysis**—SignalTap II ロジック・アナライザは、**Stop Analysis** ボタンがクリックされるまでデータのキャプチャを継続し、すべてのトリガ・イベント条件を無視します。
- **Stop Analysis**—SignalTap II の解析が停止します。トリガ・イベントが発生していない場合、収集したデータは自動的に表示されません。
- **Read Data**—キャプチャしたデータが表示されます。このボタンは、トリガ・イベントが発生していないときでも取得したデータを表示するのに役立ちます。

パワーアップ・トリガの実行

SignalTap II ロジック・アナライザのインスタンスに対してパワーアップ・トリガをイネーブルおよびセットアップした場合、デバイス・コンフィギュレーション後にトリガ・イベントが発生した場合は、すでにキャプチャしたデータが表示できる状態になっていることがあります。キャプチャしたデータをダウンロードするか、またはパワーアップ・トリガが動作中かどうかを確認するには、Instance Manager の **Run Analysis** をクリックします。パワーアップ・トリガが発生すると、ロジック・アナライザは直ちに停止し、デバイスからキャプチャしたデータがダウンロードされます。このデータは SignalTap II エディタの **Data** タブに表示できます。パワーアップ・トリガが発生せずに、キャプチャしたデータがダウンロードされない場合、ロジック・アナライザは動作し続けます。パワーアップ・トリガ・イベントが発生するのを待つか、または **Stop Analysis** をクリックしてロジック・アナライザを停止させることができます。

ランタイム・トリガの実行

デバイスをコンフィギュレーションし、ランタイム・トリガに基づいてデータ・サンプルをキャプチャした後、SignalTap II ロジック・アナライザを手動で準備して動作させることができます。パワーアップ・トリガがイネーブルされていない場合は、すぐにこれを行うことができます。パワーアップ・トリガがイネーブルされている場合は、パワーアップ・トリガ・データがデバイスからダウンロードされた後か、またはパワーアップ・トリガ・イベントが発生しなかったためロジック・アナライザを停止させた後でこれを行うことができます。SignalTap II エディタの **Run Analysis** をクリックして、トリガ・イベントのモニタを開始しま

す。必要なインスタンスをすべて選択してから、ツールバーの **Run Analysis** をクリックすると、複数の SignalTap II インスタンスを同時に起動できます。

ロジック・アナライザを手動で停止させない場合、トリガ・イベントが TRUE になるとデータのキャプチャが開始されます。これが起こると、キャプチャしたデータがバッファからダウンロードされます。SignalTap II エディタの **Data** タブでデータを表示することができます。

強制トリガの実行

外部ロジック・アナライザやオシロスコープを使用するとき、トリガ・イベントをセットアップしたりイベントの発生を待たずに、信号の現在の状態を確認したい場合があります。設定したトリガ条件に関係なくテスト装置に強制的にデータをキャプチャさせるので、これを「強制トリガ」操作と呼びます。SignalTap II ロジック・アナライザでは、アナライザを起動して直ぐにデータをキャプチャするか、必要ときにデータをキャプチャするかを選択できます。

アナライザを起動した直後にデータをキャプチャするには、ノード・リストで各 **Trigger Level** カラムをオフにして、すべてのトリガ・レベルでトリガ条件をディセーブルします。この操作ではリコンパイルは必要はありません。Instance Manager で **Run Analysis** をクリックします。SignalTap II ロジック・アナライザは直ちにデータをトリガ、キャプチャし、SignalTap II エディタの **Data** タブにダウンロードします。データが自動的にダウンロードされない場合は、Instance Manager の **Read Data** をクリックします。

データを手動でキャプチャするタイミングを選択する場合は、トリガ条件をディセーブルする必要はありません。**Autorun Analysis** をクリックしてロジック・アナライザを起動し、**Stop Analysis** をクリックしてデータをキャプチャします。データが自動的に SignalTap II エディタの **Data** タブにダウンロードされない場合は、**Read Data** をクリックします。

最後に、トリガ・イベントが発生した後、データを手動でキャプチャするよう選択できます。これは、トリガ・イベントを発生させ、その後にトリガ・イベントそのものではなく信号に関するデータをキャプチャしたい場合に役立ちます。これを実行するには、**Buffer acquisition mode** を **Circular** および **Continuous** に設定し、**Run Analysis** をクリックします。トリガ・イベントが発生すると、SignalTap II Health Monitor の状態が **Acquiring post-trigger data** として表示されますが、ロジック・アナライザは停止しません。データのキャプチャおよびダウンロードを行う場合は、**Stop Analysis** をクリックします。データが自動的にダウンロードされない場合は、**Read Data** をクリックします。

SignalTap II ステータス・メッセージ

表 13-7 に、データの収集中およびその前後に Instance Manager の SignalTap II Health Monitor に表示されることがあるテキスト・メッセージについて説明します。これらのメッセージにより、ロジック・アナライザの状態またはロジック・アナライザが実行している処理を知ることができます。

メッセージ	説明
Not running	SignalTap II ロジック・アナライザは動作していません。デバイスに接続されていないかデバイスがコンフィギュレーションされていません。
(Power-Up Trigger) Waiting for clock (1)	SignalTap II ロジック・アナライザはランタイムまたはパワーアップ・トリガ収集を実行中で、クロック信号の遷移を待っています。
Acquiring (Power-Up) pre-trigger data (1)	トリガ条件が評価されていません。循環バッファ収集モードが選択されている場合は、データのフル・バッファが収集されます。
Trigger In conditions met	Trigger In 条件が発生しました。SignalTap II ロジック・アナライザは最初のトリガ・レベル条件の発生を待っています。これは Trigger In を指定した場合に表示される可能性があります。
Waiting for (Power-up) trigger (1)	SignalTap II ロジック・アナライザはトリガ・イベントの発生を待っています。
Trigger level <x> met	トリガ・レベル x の条件が発生しました。SignalTap II ロジック・アナライザはレベル x+1 で指定した条件の発生を待っています。
Acquiring (power-up) post-trigger data (1)	全部のトリガ・イベントが発生しました。SignalTap II ロジック・アナライザはトリガ発生後のデータを取得します。トリガ発生後に収集されるデータ量は、循環バッファ収集モード選択時のユーザ定義値 12%、50%、および 88% です。
Offload acquired (Power-Up) data (1)	JTAG チェインを介して Quartus II ソフトウェアにデータを送信中です。
Ready to acquire	SignalTap II ロジック・アナライザはユーザがアナライザを動作可能な状態にするのを待っています。

表 13-7 の注：

- (1) このメッセージは、ランタイム・トリガ・イベントとパワーアップ・トリガ・イベントの両方に表示できます。パワーアップ・トリガを参照するときは、括弧で囲まれたテキストが追加されます。



セグメント収集モードでは、トリガ前およびトリガ後は適用されません。

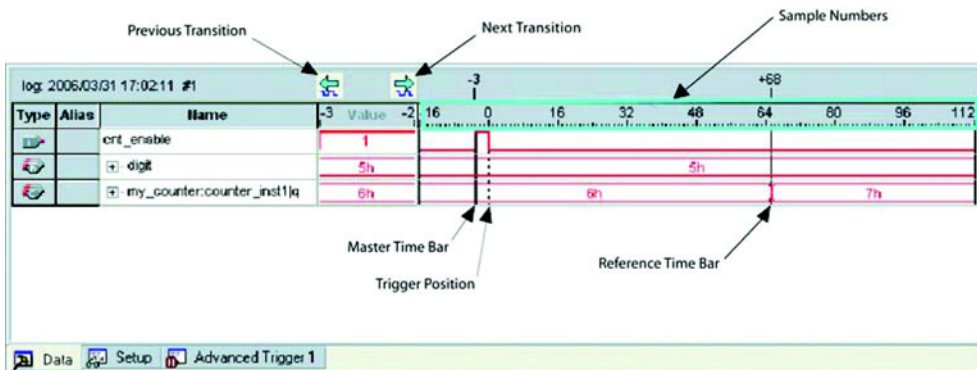
キャプチャしたデータの表示、解析、および使用

トリガ・イベントが発生した後またはデータを手動でキャプチャした後で、SignalTap II のインタフェースを使用してデータを調査し、結果をデザインのデバッグに使用することができます。SignalTap II ロジック・アナライザはこれを容易に行うためのさまざまな機能を提供します。

キャプチャしたデータの表示

キャプチャした SignalTap II データを SignalTap II ファイルの **Data** タブで表示することができます (図 13-22)。Data タブの各ロウはキャプチャした 1 つの信号またはバスのデータを表示します。バスを拡大してバス上の各信号のデータを表示することができます。データ波形をクリックするとキャプチャしたデータ・サンプルが拡大され、右クリックで縮小されます。

図 13-22. キャプチャした SignalTap II データ



キャプチャしたデータを表示する場合、2 つのイベント間の時間間隔が分かる则便利です。タイム・バーにより、システムでキャプチャしたデータの 2 つのサンプル間のクロック・サイクル数が分かります。タイム・バーには、次の 2 つのタイプがあります。

- **Master Time Bar**— マスタ・タイム・バーのラベルは、位置の絶対時間を太字で表示します。マスタ・タイム・バーは、Data タブにある太い黒線です。キャプチャしたデータにはマスタ・タイム・バーが 1 つだけあります。
- **Reference Time Bar**— リファレンス・タイム・バーのラベルは、マスタ・タイム・バーを基準にした時間を表示します。作成できるリファレンス・タイム・バーの数には制限がありません。

マスタ・タイム・バー位置に対する信号遷移を検出するには、**Next Transition** ボタンまたは **Previous Transition** ボタンのいずれかを使用します。これにより、マスタ・タイム・バーを選択した信号または信号グループの次の遷移または前の遷移に揃えることができます。この機能は、サンプル容量が非常に大きく、信号トグル・レートが非常に低い場合に大いに役立ちます。

ビット・パターン用ニーモニックの作成

ニーモニック・テーブル機能では、バスなどのビット・パターンのセットに対して、意味のある名前を割り当てることができます。ニーモニック・テーブルを作成するには、SignalTap II ファイルの **Setup** または **Data** タブで右クリックし、**Mnemonic Table Setup** をクリックします。ニーモニック・テーブルは、ビット・パターン・セットを入力し、各パターンを表すラベルを指定して作成します。ニーモニック・テーブルを作成した後、それを信号グループに割り当てます。ニーモニック・テーブルを割り当てするには、グループを右クリックして、**Bus Display Format** をクリックし、希望のニーモニック・テーブルを選択します。

テーブルで作成するラベルは、**Setup** および **Data** タブでさまざまな方法で使用されます。**Setup** タブで、任意の **Trigger Levels** カラムのエントリを右クリックし、信号グループに割り当てたテーブルからラベルを選択することによって、意味のある名前を持つ基本トリガを作成することができます。**Data** タブでは、キャプチャしたデータが割り当てられたニーモニック・テーブルに含まれるビット・パターンと一致する場合は、信号グループのデータが適切なラベルに置き換えられ、期待したデータ・パターンが発生したことを容易に確認できます。

プラグインを使用した自動ニーモニック

プラグインを使用して信号を SignalTap II ファイルに追加する場合、追加する信号のニーモニック・テーブルが自動的に作成され、プラグインで定義された信号に割り当てられます。これらのニーモニック・テーブルを手動でイネーブルする必要がある場合は、信号名または信号グループ名を右クリックします。**Bus Display Format** サブメニューで、プラグインと一致するニーモニック・テーブルの名前をクリックします。

例えば、Nios II プラグインではコードを実行すると、デザインの信号アクティビティを容易にモニタできるようになります。ロジック・アナライザを ELF ファイルからのデータに基づいて Nios II コードのファンクション名でトリガするようにセットアップした場合、[13-56 ページの図 13-23](#)を参照してくださいに示すとおり、トリガ・サンプルで **Disassembly** 信号グループの対応する逆アセンブル・コードと併せて、**Instance Address** 信

号グループのファンクション名を表示することができます。トリガ周辺のキャプチャしたデータ・サンプルは、トリガ・ファンクション名からのオフセット・アドレスとして参照されます。

図 13-23. Nios II プラグインを使用する場合の Data タブ

Type	Alias	Name	37	Value	38	48	49	50	51	52
PC		...Nios II Inst Address		alt_main+0x8		<empty>		alt_main+0xc		<empty>
DIS		...Nios II Disassembly		mov fp, sp		<empty>		movi r2, 2		<empty>

デザインでのノードの検索

SignalTap II ロジック・アナライザを使用してデザインでのバグの発生源を検索する場合、ノード検索機能を使用して Quartus II ソフトウェアにあるツールの多くやデザイン・ファイルでその信号を検索できます。これによって、問題の発生源を素早く見つけ、デザインを変更して問題点を是正することができます。Quartus II ソフトウェア・ツールまたはデザイン・ファイルの 1 つで SignalTap II ロジック・アナライザからの信号を検索するには、SignalTap II ファイルで信号を右クリックし、**Locate in <tool name>** をクリックします。以下の場所のいずれかでノード・リストからの信号を検索することができます。

- Assignment Editor
- Pin Planner
- タイミング・クロージャ・フロアプラン
- Chip Planner
- Resource Property Editor
- Technology Map Viewer
- RTL Viewer
- デザイン・ファイル



これらのツールの使用について詳しくは、「Quartus II ハンドブック」の該当する章を参照してください。

キャプチャしたデータの保存

データ・ログは、キャプチャしたデータの履歴とデータのキャプチャに使用したトリガを示します。アナライザはデータを取得してログに保存し、波形として表示します。ロジック・アナライザがオートラン・モードのときにトリガ・イベントが複数回発生すると、トリガが発生するごとにキャプチャしたデータが個別エントリとしてデータ・ログに保存されます。これにより、各トリガ・イベントでキャプチャしたデータを簡単に遡って確認することができます。ログのデフォルト名はデータを取得した時刻に基づいて付けられます。アルテラでは、データ・ログ名をより意味のある名前に変更することを推奨しています。

ログは階層型に編成され、キャプチャしたデータの類似ログがトリガ・セットでグループ化されます。**Data Log** ペインが閉じている場合、**View** メニューの **Data Log** を選択して再度開きます。データのロギングをイネーブルするには、**Data Log** の **Enable data log** をオンにします (図 13-11)。あるトリガ・セットのデータ・ログを呼び出してアクティブにするには、リストでデータ・ログ名をダブル・クリックします。

データ・ログ機能は、さまざまなトリガ条件セットおよび信号コンフィギュレーション・セットを編成するのに役立ちます。[13-30 ページの「複数の SignalTap II ファイルおよび コンフィギュレーションの管理」](#) を参照してください。

キャプチャしたデータの他のファイル・フォーマットへの変換

キャプチャしたデータは、以下のファイル・フォーマットでエクスポートできます。この中には他の EDA シミュレーション・ツールで使用できるものもあります。

- カンマ区切り値ファイル (.csv)
- テーブル・ファイル (.tbl)
- 値変更ダンプ・ファイル (.vcd)
- ベクタ波形ファイル (.vwf)
- グラフィックス・フォーマット・ファイル (.jpg, .bmp)

SignalTap II ロジック・アナライザでキャプチャしたデータをエクスポートするには、**Flie** メニューの **Export** をクリックして、**File Name**、**Export Format**、および **Clock Period** を指定します。

SignalTap II リスト・ファイルの作成

キャプチャしたデータは SignalTap II リスト・ファイルで表示することもできます。SignalTap II リスト・ファイルは、ロジック・アナライザでトリガ・イベントに対してキャプチャしたすべてのデータをリストするテキスト・ファイルです。リスト・ファイルの各ロウは、バッファ内の1つのキャプチャしたサンプルに対応しています。カラムはキャプチャした各信号の値またはそのサンプルの信号グループに対応しています。キャプチャしたデータ用にニモニック・テーブルが作成されている場合、リストの値はテーブルのマッチング・エントリに置き換えられます。これは、インストラクション・コード・デイスアセンブリを含むプラグインを使用する場合は特に役立ちます。トリガ・イベントと同じ期間内にインストラクション・コードが実行された順序をすぐに確認できます。SignalTap II リスト・ファイルを作成するには、File メニューの **Create/Update** を選択して、**Create SignalTap II List File** をクリックします。

その他の機能

SignalTap II ロジック・アナライザには、必ずしもタスク・フローの特定のタスクに属さない他の機能が数多くあります。

SignalTap II MATLAB MEX ファンクションを使用したデータのキャプチャ

DSP デザインに MATLAB を使用する場合、MATLAB MEX ファンクション `alt_signaltap_run` を呼び出して Quartus II ソフトウェアに組み込み、SignalTap II ロジック・アナライザから取得したデータを直接 MATLAB 環境のマトリックスに取り込むことができます。MEX ファンクションをループで繰り返し使用する場合、Quartus II ソフトウェア環境で SignalTap II を使用するときには、同じ時間でできる限り多くのデータ取得を実行することができます。



SignalTap II MATLAB MEX ファンクションは、Windows バージョンの Quartus II ソフトウェアでのみ使用できます。MATLAB リリース 14 オリジナル・バージョン7以降に対応しています。

Quartus II ソフトウェアと MATLAB 環境をセットアップして、SignalTap II を使用してデータ収集を実行するには、以下のステップを実行します。

1. Quartus II ソフトウェアで SignalTap II ファイルを作成します。

2. SignalTap II エディタの **Data** タブのノード・リストで、信号および信号グループを MATLAB マトリックスで表示する順に編成します。インポートしたマトリックスの各ロウは、1 つの SignalTap II の収集サンプルを表し、各カラムは信号または信号グループを **Data** タブで編成した順序で表します。



MEX ファンクションを使用して SignalTap II ロジック・アナライザから取得して MATLAB 環境に転送した信号グループは、32 個の信号幅に制限されています。MEX ファンクションをバスまたは33以上の信号からなる信号グループで使用する場合は、信号グループを 32 個の信号制限を越えない小さなグループに分割します。

3. SignalTap II ファイルを保存して、デザインをコンパイルします。デバイスをプログラムして SignalTap II ロジック・アナライザを実行し、トリガ条件と信号収集が正しく行われているか確認します。
4. MATLAB 環境では、以下のコマンドで Quartus II バイナリ・ディレクトリをパスに追加します。

```
addpath <Quartus install directory>\win ←
```

MATLAB で演算子を付けずに `alt_signaltap_run` と入力して、MEX ファンクションのヘルプ・ファイルを表示することができます。

MATLAB 環境で MEX ファンクションを使用してデバイスへの JTAG 接続を開き、SignalTap II ロジック・アナライザを使用してデータを取得します。データを取得した後、接続を閉じる必要があります。

JTAG 接続を開いて、キャプチャしたデータを直接 `stp` という MATLAB マトリックスに取り込むには、以下のコマンドを使用します。

```
stp = alt_signaltap_run('<stp filename>', ('signed'|'unsigned')[, '<instance names>'], '/<signalset name>', '<trigger name>']; ←
```

データをキャプチャする場合は、<stp filename> が使用する SignalTap II ファイルの名前になります。このファイルは MEX ファンクションを使用するのに必要です。他の MEX ファンクションのオプションは表 13-8 で定義されます。

オプション	使用法	説明
signed unsigned	'signed' 'unsigned'	signed オプションは、信号グループのデータを符号付き 2 の補数に変換します。SignalTap II の Data タブで定義されるグループの最上位ビット (MSB) は符号ビットです。 unsigned オプションは、データを符号なし整数として保持します。デフォルトは signed です。
<instance name>	'auto_signaltap_0'	複数のインスタンスを定義する場合は、SignalTap II インスタンスを指定します。デフォルトは SignalTap II ファイルの最初のインスタンス auto_signaltap_0. です。
<signal set name> <trigger name>	'my_signalset' 'my_trigger'	複数のコンフィギュレーションが SignalTap II ファイルに存在する場合、SignalTap II データ・ログからの信号セットとトリガを指定します。デフォルトはファイル内のアクティブ信号セットおよびトリガです。

冗長モードをイネーブルまたはディセーブルして、ロジック・アナライザがデータを収集しているときのロジック・アナライザのステータスを確認することができます。冗長モードをイネーブル / ディセーブルするには、以下のコマンドを使用します。

```
alt_signaltap_run('VERBOSE_ON'); ←
alt_signaltap_run('VERBOSE_OFF'); ←
```

データを取得した後、JTAG 接続を閉じる必要があります。コマンドを閉じるには、以下のコマンドを使用します。

```
alt_signaltap_run('END_CONNECTION'); ←
```



MATLAB の MEX ファンクションの使用について詳しくは、MATLAB Help を参照してください。

ラボ環境での SignalTap II の使用

SignalTap II ロジック・アナライザのスタンドアロン・バージョンをインストールすることができます。このバージョンは、完全な Quartus II のインストールに必要な必要条件を満足するワークステーションがないラボ環境において、または Quartus II ソフトウェアのフル・インストールに必要なライセンスがない場合に特に有用です。SignalTap II ロジック・アナライザのスタンドアロン・バージョンは、Quartus II スタンドアロン Programmer に付属しており、アルテラ・ウェブサイト www.altera.co.jp のダウンロード・ページから入手できます。

SignalTap II ロジック・アナライザを使用したリモート・デバッグ

SignalTap II ロジック・アナライザを使用して、遠隔地の PC に接続されているデバイス上で動作するデザインをデバッグすることができます。

リモート・デバッグ・セッションを実行するには、以下のセットアップが必要です。

- ローカル PC にインストールされた Quartus II ソフトウェア
- リモート PC にインストールされたスタンドアロン SignalTap II ロジック・アナライザまたはフル・バージョンの Quartus II ソフトウェア
- 遠隔地にある PCB 上のデバイスに接続されたプログラミング・ハードウェア
- TCP/IP プロトコル接続

装置のセットアップ

遠隔地の PC では、スタンドアロン・バージョンの SignalTap II ロジック・アナライザまたはフル・バージョンの Quartus II ソフトウェアをインストールします。このリモート・コンピュータには、EthernetBlaster や USB-Blaster などのアルテラのプログラミング・ハードウェアが接続されていなければなりません。

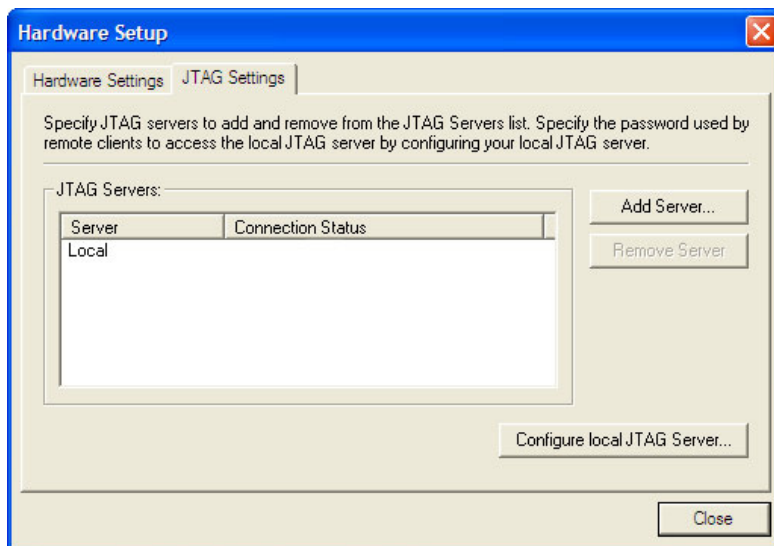
ローカル PC では、フル・バージョンの Quartus II ソフトウェアをインストールします。このローカル PC は、TCP/IP プロトコルを使用する LAN を介して、リモート PC に接続しなければなりません。

リモート PC でのソフトウェアのセットアップ

遠隔地の PC でソフトウェアをセットアップするには、以下のステップを実行します。

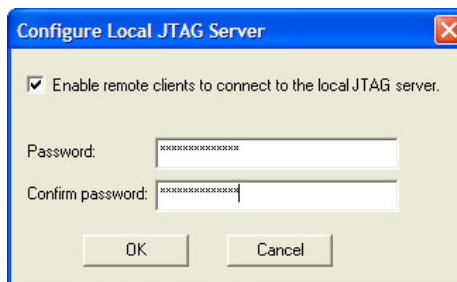
1. Quartus II programmer で、**Hardware Setup** をクリックします。
2. **JTAG Settings** タブをクリックします (13-62 ページの図 13-24 を参照してください)。

図 13-24. リモート PC での JTAG のコンフィギュレーション



3. **Configure local JTAG Server** をクリックします。
4. **Configure Local JTAG Server** ダイアログ・ボックス (図 13-25) で、**Enable remote clients to connect to the local JTAG server**, をオンにして、パスワード・ボックスにパスワードを入力します。**Confirm Password**ボックスにパスワードを再入力し、**OK**をクリックします。

図 13-25. リモートでのローカル JTAG サーバのコンフィギュレーション

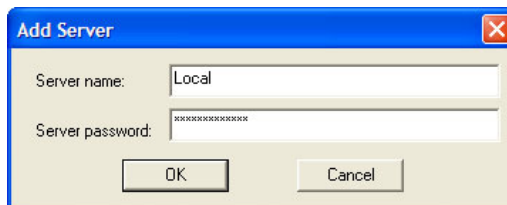


ローカル PC でのソフトウェアのセットアップ

ローカル PC でソフトウェアをセットアップするには、以下のステップを実行します。

1. Quartus II Programmer を起動します。
2. **Hardware Setup** をクリックします。
3. **JTAG settings** タブで、**Add server** をクリックします。
4. **Add Server** ダイアログ・ボックス (図 13-26) で、使用するネットワーク名またはサーバの IP アドレスとリモート PC で作成した JTAG サーバのパスワードを入力します。

図 13-26. Add Server ダイアログ・ボックス



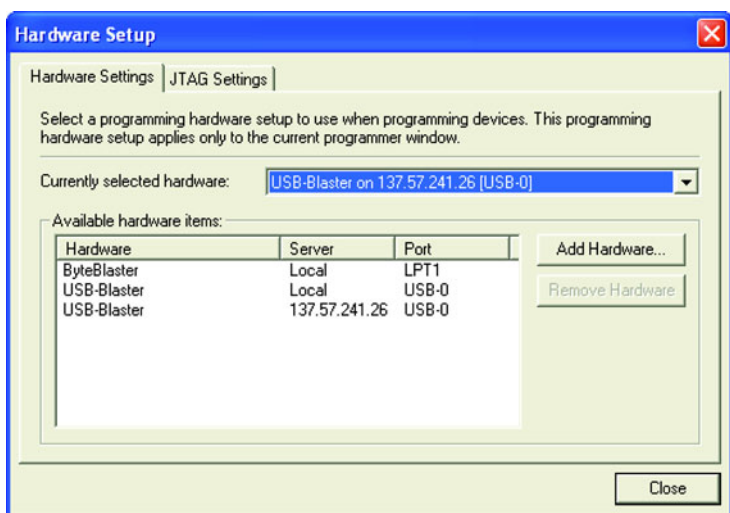
5. **OK** をクリックします。

ローカル PC での SignalTap II のセットアップ

リモート PC のハードウェアに接続するには、以下のステップを実行します。

1. **Hardware Settings** タブをクリックして、リモート PC のハードウェアを選択します (図 13-27)。

図 13-27. リモート PC でのハードウェアの選択



2. **Close** をクリックします。

これで、リモート PC に接続されたデバイスのロジック・アナライザを、ローカル PC に直接接続した場合と同じように制御することができます。

SignalTap II スクリプ ティング・ サポート

この章で説明する手順の実行と設定は、Tcl スクリプトで行うことができます。また、一部の手順はコマンド・プロンプトでも実行できます。スクリプティング・コマンド・オプションについて詳しくは、**Quartus II Command-Line** および **Tcl API Help** ブラウザを参照してください。この Help ブラウザを使用するには、コマンド・プロンプトで次のコマンドを入力します。

```
quartus_sh --qhelp ←
```



「**Quartus II Software Command-Line Operation & TCL Scripting Manual**」には、同じ情報が PDF 形式で付属しています。Tcl スクリプトについて詳しくは、「**Quartus II ハンドブック Volume 2**」の「Tcl スクリプト」の章を参照してください。Quartus II ソフトウェアにおける設定および制約について詳しくは、「**Quartus II Settings File Reference Manual**」を参照してください。コマンド・ライン・スクリプトについて詳しくは、「**Quartus II ハンドブック Volume 2**」の「**Command-Line Scripting**」の章を参照してください。

SignalTap II コマンド・ライン・オプション

SignalTap II ロジック・アナライザでコマンド・プロンプトを使用してデザインをコンパイルするには、`quartus_stp` コマンドを使用します。[表 13-9](#)に、実行コマンドの使用方法をより深く理解するためのオプションを示します。

オプション	使用方法	説明
stp_file	<code>quartus_stp --stp_file <stp_filename></code>	指定した SignalTap II ファイルを、Quartus II Settings File (QSF) の <code>USE_SIGNALTAP_FILE</code> に割り当てます。
enable	<code>quartus_stp --enable</code>	QSF の指定した SignalTap II ファイルへのアサインメントを作成し、 <code>ENABLE_SIGNALTAP</code> を ON に変更します。次にプロジェクトをコンパイルするとき、SignalTap II ロジック・アナライザがデザインに組み込まれます。QSF で SignalTap II ファイルを指定しない場合は、 <code>--stp_file</code> オプションを使用しなければなりません。 <code>--enable</code> オプションを省略した場合は、QSF の <code>ENABLE_SIGNALTAP</code> の現在値が使用されません。

表 13-9. SignalTap II コマンド・ライン・オプション (2 / 2)		
オプション	使用方法	説明
disable	quartus_stp --disable	QSF から SignalTap II ファイル・リファレンスを削除して、ENABLE_SIGNALTAP を OFF に変更します。次にデザインをコンパイルすると、SignalTap II ロジック・アナライザはデザイン・データベースから削除されます。--disable オプションを省略した場合は、QSF の ENABLE_SIGNALTAP の現在値が使用されます。
create_signaltap_hdl_file	quartus_stp --create_signaltap_hdl_file	MegaWizard Plug-in Manager で作成した SignalTap II ロジック・アナライザ・メガファンクションによって生成されたデザインでの、SignalTap II インスタンスを表す SignalTap II ファイルを作成します。このファイルは最後のコンパイルに基づきます。SignalTap II ファイルを適切に作成するには、--stp_file オプションを使用しなければなりません。これは Quartus II ソフトウェアの Create SignalTap II File from Design Instance(s) コマンドに似ています。

以下の例は、コマンド・ラインで SignalTap II ロジック・アナライザを使用してデザインをコンパイルする方法を示しています。

```
quartus_stp filtref --stp_file stp1.stp --enable ←
quartus_map filtref --source=filtref.bdf --family=CYCLONE ←
quartus_fit filtref --part=EP1C12Q240C6 --fmax=80MHz --tsu=8ns ←
quartus_tan filtref ←
quartus_asm filtref ←
```

quartus_stp --stp_file stp1.stp --enable コマンドは、QSF 変数を作成し、Quartus II ソフトウェアにデザインと共に **stp1.stp** ファイルをコンパイルするよう指示します。

MegaWizard Plug-In Manager で SignalTap II ロジック・アナライザのインスタンスを構築した後、以下のコマンド例を使用して、新しい SignalTap II ファイルを作成します。

```
quartus_stp filtref --create_signaltap_hdl_file --stp_file stp1.stp ←
```



その他のコマンド・ライン実行コマンドについて詳しくは、「Quartus II ハンドブック Volume 2」の「Command-Line Scripting」の章を参照してください。

SignalTap II Tcl コマンド

quartus_stp 実行コマンドは、Quartus II GUI を実行しないでデータ・キャプチャを可能にする Tcl インタフェースをサポートします。GUI の Tcl コンソール内から SignalTap II Tcl コマンドを実行することはできません。これらは、**quartus_stp** 実行コマンドでコマンド・ラインから実行しなければなりません。SignalTap II Tcl コマンドを持つ Tcl ファイルを実行するには、以下のコマンドを使用します。

```
quartus_stp -t <Tcl file> ←
```

表 13-10 に、SignalTap II で使用可能な Tcl コマンドを示します。

表 13-10.SignalTap II Tcl コマンド (1 / 2)		
コマンド	引数	説明
open_session	-name <stp_filename>	指定した SignalTap II ファイルを開きます。キャプチャしたデータはすべてこのファイルに保存されます。
run	-instance <instance_name> -signal_set <signal_set> (オプション) -trigger <trigger_name> (オプション) -data_log <data_log_name> (オプション) -timeout <seconds> (オプション)	アナライザを起動します。このコマンドには、アナライザを正しく起動するために必要なすべての引数を続けなければなりません。オプションで作成するデータ・ログの名前を指定できます。トリガ条件が満たされない場合は、タイムアウト値を指定してアナライザを停止させることができます。
run_multiple_start	なし	run コマンド・セットの開始を定義します。このコマンドは、複数のデータ収集のインスタンスを同時に起動するときに使用します。このコマンドは、データ収集を指定する run コマンド・セットの前に追加します。このコマンドは、run_multiple_end コマンドと共に使用します。run_multiple_end コマンドが含まれていない場合、run コマンドは実行されません。
run_multiple_end	なし	run コマンド・セットの終了を定義します。このコマンドは、複数のデータ収集のインスタンスを同時に起動するときに使用します。このコマンドは、run_commands セットの後に追加します。

表 13-10.SignalTap II Tcl コマンド (2 / 2)		
コマンド	引数	説明
stop	なし	データ収集を停止します。
close_session	なし	現在開いている SignalTap II ファイルを閉じます。SignalTap II ファイルを閉じた後でアナライザを実行することはできません。



SignalTap II Tcl コマンドについて詳しくは、Quartus II Help を参照してください。

以下の例は、継続的にデータをキャプチャするのに使用するスクリプトから抜粋したものです。トリガ条件が満たされると、データがキャプチャされデータ・ログに保存されます。

```
#opens signaltap session
open_session -name stp1.stp
#start acquisition of instance auto_signaltap_0 and
#auto_signaltap_1 at the same time
#calling run_multiple_end will start all instances
#run after run_multiple_start call
run_multiple_start
run -instance auto_signaltap_0 -signal_set signal_set_1 -trigger /
trigger_1 -data_log log_1 -timeout 5
run -instance auto_signaltap_1 -signal_set signal_set_1 -trigger /
trigger_1 -data_log log_1 -timeout 5
run_multiple_end
#close signaltap session
close_session
```

スクリプトが終了すると、データをキャプチャするのに使用する SignalTap II ファイルを開いて、データ・ログの内容を調べます。

SignalTap II ロジック・ アナライザ を使用する デザイン例

アルテラのアプリケーション・ノート、「AN 323: Using SignalTap II Embedded Logic Analyzers in SOPC Builder Systems」は、SignalTap II ロジック・アナライザを使用して SOPC Builder で生成されたシステム・モジュール内の信号をモニタする方法を説明しています。この例のシステムは、Nios プロセッサ、ダイレクト・メモリ・アクセス (DMA) コントローラ、オンチップ・メモリ、および外部 SDRAM メモリへのインタフェースなど、多くのコンポーネントを搭載しています。この例では、Nios プロセッサはオンチップ・メモリから簡単な C プログラムを実行し、ボタンが押されるまで待機します。ボタンが押されると、プロセッサは DMA 転送を開始し、ユーザは SignalTap II ロジック・アナライザを使用して解析します。



この例について詳しくは、アルテラ・ウェブサイト (www.altera.co.jp) の資料ページの「AN 323: Using SignalTap II Embedded Logic Analyzers in SOPC Builder Systems」を参照してください。

まとめ

FPGA 業界が絶えず技術的進歩を達成する中で、旧態依然とした手法に代えて生産性を最大限に高める新たな技術を導入する必要があります。SignalTap II ロジック・アナライザは、専用テスト装置の多くの欠点を克服しながら、従来のロジック・アナライザの持つ利点を継承しています。このバージョンの SignalTap II ロジック・アナライザは、FPGA の内部信号をキャプチャおよび解析可能な多数の新しい革新的機能を備えており、ユーザは非常に短時間でデザイン問題の原因を特定できます。

参考資料

この章では以下のドキュメントを参照しています。

- 「Quartus II ハンドブック Volume 3」の「Quick Design Debugging Using SignalProbe」の章
- 「Quartus II ハンドブック Volume 3」の「In-System Debugging Using External Logic Analyzers」の章
- 「Quartus II ハンドブック Volume 2」の「I/O Management」の章
- 「Quartus II ハンドブック Volume 1」の「Quartus II Integrated Synthesis」の章
- 「Quartus II ハンドブック Volume 1」の「Quartus II Incremental Compilation for Hierarchical & Team-Based Design」の章
- 「Quartus II ハンドブック Volume 2」の「Area & Timing Optimization」の章
- 「Quartus II ハンドブック Volume 2」の「Tcl Scripting」の章
- 「Quartus II Settings File Reference Manual」
- 「Quartus II ハンドブック Volume 2」の「Command-Line Scripting」の章
- 「AN 323: Using SignalTap II Embedded Logic Analyzers in SOPC Builder System」

改訂履歴

表 13-11 に、この章の改訂履歴を示します。

表 13-11.改訂履歴		
日付 & ドキュメント・バージョン	変更内容	概要
2007 年 5 月 v7.1.0	13-69 ページの「参考資料」を追加。	—
2007 年 3 月 v7.0.0	13-4 ページのリストに Cyclone III デバイスのサポートを追加	—
2006 年 11 月 v6.1.0	Quartus II ソフトウェア・バージョン 6.1 のための更新 <ul style="list-style-type: none"> ● 図 13-4、13-11、13-16、13-17、13-18 を更新。新規の図 13-23 を追加。 ● その他の変更 ● インクリメンタル配線に関する情報を削除（機能が削除されたため） ● インクリメンタル・コンパイルの使用に関する詳細な情報を追加 ● Nios II プラグインの使用に関する詳細な情報を追加 ● SignalTap II ファイル /SOF の互換性に関する詳細な情報を追加 ● Trigger in/out を使用してあるロジック・アナライザを別のロジック・アナライザでトリガする手法を更新。 	Quartus II ソフトウェア・バージョン 6.1 のための更新
2006 年 5 月 v6.0.0	Quartus II ソフトウェア・バージョン 6.0 のための更新	—
2005 年 10 月 v5.1.0	Quartus II ソフトウェア・バージョン 5.1 のための更新	—
2005 年 5 月 v5.0.0	<ul style="list-style-type: none"> ● 情報を更新。 ● 図を更新。 ● Quartus II ソフトウェア・バージョン 5.0 の新機能。 	—
2004 年 12 月 v1.0	初版	—