ISPを使用するための ガイドライン

1999年 5 月 ver. 3 Application Note 100

イントロダクション

「Time-to-Market」の要求が高まると共に、開発や製造で問題を発生させないようにした最新のシステム・レベル製品が求められるようになっています。イン・システム・プログラマビリティ（ISP）機能を持ったプログラマブル・ロジック・デバイス（PLD）は、開発期間の短縮、フィールドでのアップグレードの実現、製造工程の簡略化、在庫コストの低減、プリント基板（PCB）のテスト機能の改善に役立ちます。アルテラのISP対応デバイスは、IEEE Std. 1149.1 のJTAG（Joint Test Action Group）インタフェースを使用してプログラミングおよび再プログラミングすることができます。JTAGインタフェースを使用することによって、デバイスのプログラミングとPCBのファンクション・テストをひとつの工程で行うことが可能となり、テスト時間の短縮とアッセンブル・コストの低減を実現することができます。このアプリケーション・ノートは、ISPを使用したデザインで守られる必要があるガイドラインを、以下の項目ごとに解説したものです。

- 一般的なISPに対するガイドライン
- IEEE Std.1149.1の信号
- シーケンシャル・プログラミングとコンカレント・プログラミング
- ISPのトラブルシューティングに関するガイドライン
- エンベデッド・プロセッサによるISP
- イン・サーキット・テスタによるISP

一般的なISPに対するガイドライン

このセクションでは、ISP対応デバイスのデザインを行う上で役立つガイドラインを提供します。これらのガイドラインは、個々のデザインの実現方法とは関係なく適用される必要があります。

デバイスの動作条件

アルテラの各デバイスには、適切な動作を行うために必要となる動作条件やパラメータの値が規定されています。イン・システム・プログラミングの実行時においては、これらの条件は必ず守られている必要があります。イン・システム・プログラミングの実行時に、これらの動作条件が守られなかった場合には、プログラミング不良が発生したり、デバイスに不正なパターンがプログラムされる結果となります。

V_{CCISP} 電圧

アルテラのすべてのISP対応デバイスには、V_{CCISP}と呼ばれる規格が設定されています。デバイスのEEPROMセルが正しくプログラムされるようにするためには、イン・システム・プログラミング時にこのV_{CCISP}をVCCINTピンの電圧レベルに保っておく必要があります(V_{CCINT}=V_{CCISP})。V_{CCISP}の規格は一般用および工業用温度範囲の製品の双方に適用されます。

イン・システム・プログラミング時の消費電力はユーザ・モードにおける消費電力を超えることがあるため、イン・システム・プログラミングの設定条件を調整して双方のモードにおいて正しい電圧レベルを維持する必要があります。アルテラは、オシロスコープを使用してデバイスのVCCINTピンでV_{CCISP}の電圧レベルをチェックすることを推奨します。まず最初に、オシロスコープのトリガ・レベルを該当するデバイス・ファミリのデータシートの推奨動作条件に記載されているV_{CC}の最小値に設定します。そして、デバイスのこれらのピンにプローブを当て、VCCINTとグランド間の電圧を測定します。次に、オシロスコープのトリガ・レベルを推奨動作条件の表に記載されているV_{CC}の最大値に設定し、このテストを繰り返します。オシロスコープがいずれかの電圧でトリガされた場合は、プログラミングの設定条件を調整する必要があります。

入力電圧

各デバイス・ファミリのデータシートでは、絶対最大定格と推奨動作条件の表の中にデバイスの入力電圧が規定されています。絶対最大定格の表の中で規定されている入力電圧範囲はデバイスが永久に破壊されてしまう危険性のある限界の電圧を示しています。例えば、MAX® 9000デバイスの最大入力電圧は7.0V、最小入力電圧は−2.0Vとなっています。

推奨動作条件の表では、デバイスが安全に動作する電圧範囲が規定されています。すべてのデバイスは（ground−1.0V）から（V_{CCINT}+1.0V）の範囲の入力電圧、最高100mAまでの入力電流で安全に動作します。ただし、イン・システム・プログラミング時に、すべてのピンの遷移でグランドのアンダシュート、またはV_{CC}のオーバシュートが発生しないようにする必要があります。一般的に、オーバシュートの問題は、イン・システム・プログラミング時にもトグルするようになっているフリー・ランニングのクロックやデータ・バスで発生する可能性があります。1.0Vを超えるオーバシュートが発生しているピンには、シリーズ・ターミネーションの対策が必要です。

推奨動作とターミネーション方法の詳細については、「*Operating Requirements for Altera Devices*」（日本語版データシート「アルテラ・デバイス使用上の注意」)およびアプリケーション・ノート、AN 75「*High-Speed Board Designs*」をご覧ください。

イン・システム・プログラミングのインタラプト

部分的にプログラムされたデバイスは予想できない動作を行う可能性があるため、アルテラはプログラミングのプロセスを中断させることを推奨していません。部分的にプログラムされたデバイスは信号のコンフリクトを起こす原因となることがあり、これによってデバイスが永久に破壊され、ボード上で適切に動作している他のデバイスに影響を与える可能性もあります。

MultiVolt対応デバイスと電源投入シーケンス

イン・システム・プログラミングやバウンダリ・スキャン・テストの実行時にJTAG回路を正常に動作させるためには、JTAGチェイン内のすべてのデバイスが同じステートになっている必要があります。したがって、複数の電源電圧が使用されているシステムでは、チェイン内のすべてのデバイスに完全に電源が投入されるまで、JTAG回路を「test-logic-reset」のステートに保っておく必要があります。複数の電源電圧が使用されているシステムでは、すべての電源の電圧レベルを同時に規定値までに上げることが不可能なため、特にこのプロセスが重要になります。

MultiVolt™機能をサポートしているアルテラのデバイスには、V_{CCINT}とV_{CCIO}で２種類の電源を使用できます。V_{CCINT}はJTAG回路に電源を供給し、V_{CCIO}はTDOを含むすべてのピンの出力ドライバに電源を供給します。したがって、これらのデバイスに２種類の電源電圧が使用されている場合は、双方の電源が規定の電圧に達するまで、JTAG回路が「test-logic-reset」のステートを保っている必要があります。 JTAG回路がこの「test-logic-reset」のステートを維持していない場合には、イン・システム・プログラミングでエラーが発生する可能性があります。

V_{CCIO}よりも先にV_{CCINT}を供給した場合

V_{CCIO}ピンより先にV_{CCINT}ピンに電源が供給されると、JTAG回路はアクティブとなりますが、出力をドライブすることはできません。このため、TCKの変化でステート・マシンが意図しないJTAGステートに遷移してしまう可能性があります。TMSとTCKがV_{CCIO}に接続されていて、V_{CCIO}に電源が供給されていない状態では、これらのJTAG信号がフローティングの状態となります。これらのフローティングの値が原因となって、デバイスが意図しないJTAGステートに遷移し、最終的にV_{CCIO}が投入されたときに不適切な動作を行う可能性があります。このため、すべてのJTAG信号は、6ページの「IEEE Std. 1149.1回路のディセーブル方法」で解説されているように、ディセーブルされた状態になっていなければなりません。

V_{CCINT}よりも先にV_{CCIO}を供給した場合

V_{CCINT}よりも先にV_{CCIO}に電源が供給されると、JTAG回路はアクティブとならず、TDOはトライ・ステートとなります。この場合、JTAG回路はアクティブとなっていませんが、 JTAGチェインの次のデバイスにV_{CCIO}の配線パターンを通じて電源が供給されるようになっている場合には、このデバイ

スのJTAG回路が「test-logic-reset」のステートを維持している必要があります。すべてのデバイスのTMSとTCKの信号は共通に接続されているため、チェイン内のすべてのデバイスがディセーブルされている必要があります。このため、TCKをLowにすることによって、JTAGピンをディセーブルする必要があります。

イン・システム・プログラミング時にトライ・ステートになるI/Oピン

すべてのデバイスのI/Oピンは、イン・システム・プログラミングの実行時にトライ・ステートになります。MAX 7000S、MAX 7000A、MAX 7000AE、MAX 7000B、およびMAX 3000Aの各デバイスには弱いプルアップ抵抗が内蔵されています。このプルアップ抵抗は、未使用のI/Oピンに対する外部のプルアップ抵抗を不要にするためのものです。このプルアップ抵抗の値は、各デバイス・ファミリのデータシートの中に示されています。

イン・システム・プログラミングの実行時に特定の値が要求される信号には（出力イネーブルやチップ・イネーブルなどの信号）、適切なプルアップ抵抗またはプルダウン抵抗を追加する必要があります。プルアップ抵抗またはプルダウン抵抗が追加されなかった場合に、イン・システム・プログラミング時にデバイスに大きな電流が流れ（ボード上でのコンフリクトによって発生）、「unrecognized device」や「verify error」などのイン・システム・プログラミング不良が発生したり、イン・システム・プログラミング後に電源の投入ができないような不具合が発生することがあります。

IEEE Std. 1149.1 の信号

このセクションでは、イン・システム・プログラミングにおけるIEEE Std. 1149.1（JTAG）インタフェース信号に対するガイドラインについて解説します。

TCK信号

イン・システム・プログラミングで発生するほとんどの不具合は、TCK信号にノイズが乗っていることが原因となっています。立ち上がりまたは立ち下がりエッジにノイズを含んだTCK信号が入力されることによって、IEEE Std. 1149.1のテスト・アクセス・ポート（TAP）コントローラに不適切なクロックが与えられる可能性があります。このような不適切なクロックが供給されると、ステート・マシンが不定のステートに遷移したり、イン・システム・プログラミングで不具合が発生する可能性があります。

さらに、TCK信号はJTAGチェイン内にパラレルに接続されているすべてのIEEE Std. 1149.1対応デバイスをドライブする必要があるため、大きなファン・アウトを持つ可能性があります。したがって、大きなファン・アウトを持つユーザ・モードのクロック信号と同じように、TCKに対しても波形が正常に保たれるようなクロック・ツリーの設計が必要となります。不適切なクロック波形によって発生する代表的な不具合モードとしては、「Invalid ID」のエラー・メッセージ、ブランク・チェック・エラー、ベリファイ・エラーの発生などがあります。

アルテラはTCK信号を抵抗を介してLowレベルにプルダウンしておくことを推奨します。このときの抵抗値としては、ボード上のデバイス数と消費される電流に応じて、1kΩから5kΩ程度となります。

高速で遷移するTCKのエッジがボードのインダクタンスと結合して、オーバシュートの問題を発生させる原因になることがあります。このような状態が発生している場合は、配線パターンのインダクタンスを減少させるか、スルー・レートの低いTTL（Transistor-to-Transistor Logic）のドライバを使用してスイッチング・レートを低下させる必要があります。アルテラでは、抵抗とキャパシタによる（RC）ネットワークを使用してエッジ・レートを低下させる方法はデバイスの入力に対する規格を超える可能性があるため推奨していません。ほとんどの場合は、ドライバ専用チップを使用することで、エッジ・レートが遅くなりすぎる問題を避けることができます。アルテラは、電源投入後にグリッジが発生しないドライバ・チップの使用を推奨します。

ダウンロード・ケーブルによるプログラミング

MasterBlaster™、ByteBlasterMV™、ByteBlaster™または BitBlaster™ のダウンロード・ケーブルを使用しているときや、JTAGチェイン内に３個以上のデバイスが含まれている場合は、アルテラはチェインにバッファを追加することを推奨します。この場合は、ノイズの発生を最小に抑えるため、低速遷移タイプのバッファを選択する必要があります。

ダウンロード・ケーブルを延長する必要があるときは、ダウンロード・ケーブルに標準的なPC用パラレル・ポート・ケーブル、またはシリアル・ポート・ケーブルを接続することができます。ただし、ダウンロード・ケーブルの10ピン・ヘッダ側でケーブルを延長することはできません。このヘッダ側でのケーブルの延長は、ノイズやイン・システム・プログラミングで問題を発生させる原因となります。

MasterBlaster、ByteBlasterMV、ByteBlaster、およびBitBlasterの各ダウンロード・ケーブルの詳細については、"*MasterBlaster Serial/USB Communications Cable* "、"*ByteBlasterMV Parallel Port Download Cable* "、"*ByteBlaster Parallel Port Download Cable* "、"*BitBlaster Serial Download Cable* "の各データシートを参照してください。

IEEE Std. 1149.1回路のディセーブル方法

ISPまたはバウンダリ・スキャン・テスト（BST）回路を使用しないデザインに対して、アルテラはIEEE Std. 1149.1の回路をディセーブルしておくことを推奨します。表１は、IEEE Std. 1149.1の回路を使用しないときのディセーブル方法を示したものです。

表１　IEEE Std. 1149.1回路のディセーブル方法

デバイス	永久にディセーブル	ISPおよびBST時にイネーブルにし、ユーザ・モード時にディセーブル
MAX 7000S MAX 7000B *(1)* MAX 7000A *(1)* MAX 7000AE *(1)* MAX 3000A *(1)*	MAX+PLUS® II で *Enable JTAG Support*のオプションをOFF に設定する。	TMS を High にプルアップして TCK を Low にプルダウンするか、TCKがHighになる前にTMSをHighにプルアップする。*(2)*
MAX 9000 MAX 9000A	TMSをHighにプルアップして TCK を Low にプルダウンするか、TCK が High になる前に TMS を High にプルアップする。*(2)*	TMS を High にプルアップして TCK を Low にプルダウンするか、TCKがHighになる前にTMSをHighにプルアップする。*(2)*

注：
(1) MAX 7000B、MAX 7000A、MAX 7000AE、MAX 3000Aの各デバイスに関する情報は暫定仕様に基づくものです。
(2) 標準的な抵抗値は1kΩから5kΩです。この値はボード上のデバイス数や消費される電流によって異なります。

JTAG回路を永久にディセーブルする方法（MAX 7000S、MAX 7000B、MAX 7000A、MAX 7000AE、およびMAX 3000Aデバイスの場合）

MAX 7000S、MAX 7000B、MAX 7000A、MAX 7000AE、およびMAX 3000AのJTAGピンは、JTAGポートまたはI/Oピンのいずれかとして使用することができます。このため、MAX+PLUS IIでデザインをコンパイルする前に、*Enable JTAG Support*のオプションをONまたはOFFに設定することによって、これらのピンをどのように使用するか規定しておく必要があります。 *Enable JTAG Support*のオプションをONに設定すると、これらのピンがイン・システム・プログラミングおよびバウンダリ・スキャン・テストの実行時にJTAGポートとして動作します。また、*Enable JTAG Support*のオプションをOFFに設定した場合は、これらのピンがI/Oピンとして動作し、イン・システム・プログラミングやバウンダリ・スキャン・テストが実行できなくなります。

MAX+PLUS IIを使用してJTAG回路をディセーブルする方法についての詳細は、MAX+PLUS IIのヘルプ機能で "Classic & MAX Global Project Device Options Dialog Box"または"Classic & MAX Individual Device Options Dialog Box"の項目をサーチして確認してください。

JTAG回路を永久にディセーブルする方法（MAX 9000およびMAX 9000Aデバイスの場合）

MAX 9000およびMAX 9000AデバイスはJTAG専用のピンと回路を持っているため、JTAG回路が常時イネーブルとなっています。このため、ISPやBST回路を使用する予定がない場合は、JTAGピンから回路をディセーブルしておくことができます。JTAGの仕様では、JTAG回路をディセーブルするときはTMSをHighにすると規定されていますが、TCKについては何も規定されていません。アルテラは、この場合にTMSをHighにTCKをLowにしておくことを推奨します。TCKをLowにしておくことによって、電源投入時のシーケンスでもTCKに立ち上がりエッジが発生することがなくなります。

TCKをHighレベルにすることもできますが、TCKがHighになる前にTMSをHighにしておく必要があります。TMSを最初にHighにしておくことによって、TCKの立ち上がりエッジまたはエッジでJTAG回路が「test-logic-reset」のステートから抜けてしまう問題を避けることができます。

ISPとBSTの実行時にJTAG回路をイネーブルにし、ユーザ・モードでディセーブルする方法

イン・システム・プログラミングまたはバウンダリ・スキャン・テストにJTAG回路を使用するアルテラのISP対応デバイスでは、JTAG回路をISPとBSTの実行時にイネーブルにし、それ以外のときにディセーブルしておく必要があります。JTAG回路の動作は、JTAGピンを通じてコントロールすることができます。MAX 9000およびMAX 9000AデバイスのJTAG回路を永久にディセーブルする場合は、TCKをLowにしてTMSをHighにするか、TCKがHighになる前にTMSをHighにします。

異なる電圧レベルで動作するデバイスへの対策

JTAGチェイン内のデバイスが異なる電圧レベルで動作している場合は、デバイスの出力電圧の規格が次のデバイスの入力電圧の規格に適合している必要があります。デバイスがこの条件を満たしていない場合は、レベル・シフタのような回路を追加して電圧レベルを変更しなければなりません。例えば、5.0Vデバイスが2.5Vデバイスをドライブしている場合は、5.0Vデバイスの出力電圧が2.5Vデバイスの入力電圧規格を満足していなければなりません。

JTAGチェイン内のすべてのデバイスは相互に接続されており、チェイン内のデバイスを正常にプログラムするためには、最初のデバイスのTDO出力が次に接続されているデバイスのTDIの入力電圧規格を満足するようになっている必要があります。

アルテラのすべてのISP対応デバイスにはMultiVolt I/O機能が提供されており、これらのデバイスと電源電圧レベルの異なるシステムとのインタフェースが可能になっています。5.0V動作のすべてのMultiVoltデバイスには、3.3Vまたは5.0VのI/O動作を設定することができます。また、3.3V動作のす

べてのMultiVoltデバイスには、2.5V、3.3Vまたは5.0VのI/O動作を設定することができます。

シーケンシル・プログラミングとコンカレト・プログラミング

このセクションでは、複数のデバイスを１個ずつプログラムするシーケンシャル・プログラミングと同時にプログラムするコンカレント・プログラミングについて解説します。シーケンシャル・プログラミングとコンカレント・プログラミングの詳細については、*Product Information Bulletin 26* (*Concurrent Programming through the JTAG Interface for MAX Devices*) を参照してください。

シーケンシャル・プログラミング

シーケンシャル・プログラミングは、チェインに接続された複数のデバイスを１度に１個ずつプログラムする方法です。チェイン内の最初のデバイスのプログラムが完了すると、次のデバイスがプログラムされます。このシーケンスは、JTAGチェイン内の指定されたすべてのデバイスがプログラムされるまで継続されます。デバイスがプログラムされると、このデバイスがJTAGのBYPASS命令を使用してデータを次のデバイスに転送します。ただし、JTAG BYPASS命令がロードされたデバイスが通常のユーザ・モードで動作するように指定することもできます。

コンカレント・プログラミング

コンカレント・プログラミングは、同一ファミリのデバイスをパラレルにプログラミングする方法です。複数のデバイスをプログラミングするのに要する時間は、大容量のEEPROMやFLASHセルにデータを「焼き込む」のに必要な時間よりも僅かに長くなります。このため、シーケンシャル・プログラミングの場合よりもプログラミング時間が大幅に短縮されます。データをシフトさせるクロックのレートを高速化することで、さらに時間を短縮することもできます。シリアル・ポートのバンド幅には制限があるため、データ転送にシリアル・ポートではなくパラレル・ポートを使用することによって、さらに時間を大幅に短縮することができます。FLEX® 10KデバイスはSRAMをベースにしたデバイスのため、データを「焼き込む」必要がなく、シリアルのコンフィギュレーションだけがサポートされています。

シーケンシャル・プログラミングとコンカレント・プログラミングの選択

プログラマ・オブジェクト・ファイル(.pof）とMasterBlaster、ByteBlasterMV、ByteBlasterまたはBitBlasterダウンロード・ケーブルを使用したプログラミングには、シーケンシャル・プログラミングが自動的に選択されます。Jam™ファイル（.jam）またはシリアル・ベクタ・フォーマット（.svf）ファイルが使用された場合は、デバイスが下記の順番でプログラムまたはコンフィギュレーションされます。

1. FLEX 10Kデバイスがシーケンシャルに
2. APEX™ 20Kデバイスがシーケンシャルに
3. MAX 7000SとMAX 7000Aデバイスがコンカレントに
4. MAX 7000AEとMAX 3000Aデバイスがコンカレントに
5. EPC2デバイスがシーケンシャルに
6. MAX 9000デバイスがコンカレントに

JamまたはSVFが各デバイスごとに個別に作成されている場合は、これらのファイルでシーケンシャル・プログラミングが実行できます。この方法では、MAX+PLUS IIのProgrammerでConfigureのボタンがクリックされるまで、FLEXデバイスとAPEXデバイスのコンフィギュレーションが開始されません。

異なるモードのデバイス

チェイン内のデバイスがプログラム中のときに他のデバイスが動作可能になると、エラーが発生する可能性があります。このため、MAX 7000S、MAX 7000A、MAX 7000AE、MAX 7000B、MAX 3000Aの各デバイスには、チェイン内のすべてのデバイスがイン・システム・プログラミングを完了するまでデバイスが通常の動作に入るのを防ぐために特別なISP命令が使用されます。このモードでは、これらのデバイスがバウンダリ・スキャン・データを同期転送し、同一ファミリの他のすべてのデバイスがプログラミングを完了するのを待ちます。これによって、これらのすべてのデバイスが同時に動作を開始するようになります。APEX 20K、FLEX 10K、MAX 9000、MAX 9000Aの各デバイスでは、このモードは現在サポートされていません。これらのデバイスは、すべてのデバイス・ファミリのプログラムまたはコンフィギュレーションが完了するまで、プログラミング・ソフトウェアによりトライ・ステート・モードを保持するようになります。

ISPのトラブルシューティングに関するガイドライン

このセクションでは、ISPに関連した問題に対するトラブルシューティングに役立つ情報を提供します。

Invalid ID、Unrecognized Deviceのエラー・メッセージ

イン・システム・プログラミングの実行時に最初に行われるのが、デバイスのシリコンIDのチェックです。このシリコンIDが一致しないと、「Invalid ID or Unrecognized Device」のエラー・メッセージが生成されます。このエラーの発生原因としては、下記の項目が挙げられます。

- ダウンロード・ケーブルが正しく接続されていない
- TDOが接続されていない
- JTAGチェインが不完全
- TCK信号のノイズ
- Jam Playerの不適切なポーティング

ダウンロード・ケーブルが正しく接続されていない

ダウンロード・ケーブルが正しいポートに正しく接続されていなかったり、ボード側から電源が供給されていなかった場合は、エラー・メッセージが表示されます。

MasterBlaster、ByteBlasterMV、ByteBlaster、BitBlasterの各ダウンロード・ケーブルの詳細は、「*MasterBlaster Serial/USB Communication Cable*」、「*ByteBlasterMV Parallel Port Download Cable*」、「*ByteBlaster Parallel Port Download Cable*」「*BitBlaster Serial Download Cable*」の各データシートで確認してください。

TDOが接続されていない

デバイスのTDOポートがチェインに接続されていないと、エラーが発生します。イン・システム・プログラミング時には、データがJTAGチェイン内の各デバイスのJTAGピンを通じてシフト・インおよびシフト・アウトされなければなりません。このため、デバイスのTDOポートはチェイン内の次のデバイスのTDIポートと接続され、最後のデバイスのTDOポートがダウンロード・ケーブルのTDOポートと接続されるようになっている必要があります。

JTAGチェインが不完全

JTAGチェインが完全になっていない場合も、エラーが発生します。不完全なJTAGチェインが原因となってエラーが発生していないかをチェックするときは、オシロスコープを使用してチェイン内の各デバイスから出力されるベクタを観測します。イン・システム・プログラミング時に各デバイスのTDOポートがトグルしていない場合は、JTAGチェインが不完全な状態となっています。

TCK信号のノイズ

TCK信号のノイズは、イン・システム・プログラミングのエラーを発生させるもっとも代表的な理由となっています。TCKの立ち上がりエッジまたは立ち下がりエッジでのノイズを含んだ遷移は、IEEE Std. 1149.1のTAPコントローラに不正なクロックが供給される原因となり、ステート・マシンが不定のステートに遷移して、イン・システム・プログラミングで不具合が発生します。TCK信号のノイズ対策については、4ページの「TCK信号」を参照してください。

Jam Playerの不適切なポーティング

Jam Playerが使用されるプラットフォームに適切にポーティングされなかった場合もエラーが発生します。Jam Playerがエラーの原因になっているかどうかをチェックする場合は、Jamファイルを通じてターゲット・デバイスにIDCODE命令をロードします。 Jamファイルを使用してIDCODEの命令をロードし、IDCODEをシフト・アウトさせることができます。このテストにより、JTAGチェインが正しく設定されているか、JTAGチェインへのリードとライトの動作を適切に実行できるかを判断することができます。14ページの図１は、IDCODEを読み出すためのJamファイルの例を示したものです。

トラブルシューティングに関するヒント

このセクションでは、ISPに関連した問題を解決するために役立つ追加情報を提供します。

JTAGチェインの接続状態を検証する方法

イン・システム・プログラミングを正常に開始させるためには、JTAGチェイン内のデバイスの数がMAX+PLUS II ソフトウェアからレポートされた数と一致していなければなりません。下記の手順は、JTAGチェインが適切に接続されていることを検証するためのひとつの簡単な方法を示したものです。

1. MAX+PLUS II のProgrammerで、Multi-Device JTAG Chain Setupを選択します。

2. Multi-Device JTAG Chain Setupのダイアログ・ボックスで、Detect JTAG Chain Infoのボタンをクリックします。MAX+PLUS IIソフトウェアはJTAGチェイン内で発見されたデバイスの数をレポートします。

イン・システム・プログラミング時におけるボードのV_{CC}レベルをチェックする

オシロスコープを使用して、JTAGチェイン上のV_{CCINT}信号をモニタし、トリガ・レベルを各デバイス・ファミリのデータシートの推奨動作条件に記載されている最小のV_{CC}電圧に設定してください。イン・システム・プログラミング時にトリガが発生する場合は、デバイスに使用中の電源から供給されているよりもさらに大きな電流が必要になっている可能性があります。このような場合は、使用中の電源をさらに高電流の供給が可能なものに交換してみてください。

電源投入時の問題

電源投入時に過大な電圧や電流がI/Oピンに与えられると、JTAGチェイン内のデバイスがラッチアップ状態になる可能性があります。熱くなっているデバイスがないかチェックしてください。熱くなっているデバイスがあったときは、そのデバイスがラッチアップ状態になった可能性があり、ダメージを受けていることが考えられます。このような場合は、各電圧ソースをチェックして、デバイスに過大な電圧や電流が与えられないようにしてください。そして、影響を受けたデバイスを交換して、再度プログラミングを実行してください。

JTAGピン上のランダム信号

通常の動作モードでは、デバイスのTAPコントローラが「test-logic-reset」のステートになっていなければなりません。デバイスをこのステートに戻す必要があるときは、TMS信号をHighにしてTCKクロックを6回与えます。これでデバイスへ正常に電源が投入されれば、TCK信号にさらに大きな値のプルダウン抵抗を接続する必要があります。

ソフトウェアに関連した問題

イン・システム・プログラミング時のエラーは、場合によってはMAX+PLUS IIソフトウェアに関連していることもあります。ソフトウェアに関連したすべての問題は、アルテラのwebサイト（http://www.altera.com）にある、Altera Technical Support（Atlas^{SM}）のセクションに掲示されています。 Atlasのデータベースをサーチして、イン・システム・プログラミングでの障害が発生するソフトウェアの問題に関する情報を確認してください。

エンベデッド・プロセッサによるISP

このセクションでは、プログラミング／テスト用言語であるJamとエンベデッド・プロセッサを使用してISP対応デバイスをプログラムするときのガイドラインについて解説します。

プロセッサとメモリに対する要求

Jam Byte-Code Playerは8ビット以上のプロセッサをサポートしており、ASCII Jam Playerは16ビット以上のプロセッサをサポートしています。Jam Playerは予測可能な方法でメモリを使用するため、Jamファイルの更新を目的にしたフィールドでのアップデートが簡略化されます。Jam Playerのメモリとしては、ROMとRAMの双方が使用されます。ROMはJam Playerのバイナリ・コードとJamファイルのストアに使用され、RAMはJam Playerがコールされたときに使用されます。

Jam Playerに必要となるRAMとROMの最大容量を推定する方法については、アプリケーション・ノート、AN 88（*Using the Jam Language for ISP via an Embedded Processor*、日本語版「エンベッデッド・プロセッサによるISPにJam言語を使用する方法」）をご覧ください。

Jam Player**のポーティング**

アルテラのJam Player（Byte-CodeバージョンおよびASCIIバージョン）は、１つのPCパラレル・ポートを使用して動作します。Jam Playerを使用するプロセッサへのポーティングは、ASCII Jam Playerに対してはjamstub.cのファイルを、Jam Byte-Code Playerに対してはjbistub.cのファイルを修正するだけで行えます。他のすべてのファイルは、変更の必要がありません。

Jam Playerが正しくポーティングされていなかった場合は、Unrecognized Deviceのエラーが発生します。このエラーが発生するもっとも代表的な原因を以下に示します。

- Jam Playerのポーティング後に、TDOの値が逆極性で読み出されている。この問題は、Jam PlayerのデフォルトI/OコードがPCパラレル・ポートの使用を想定しているときに発生する可能性があります。この問題を解決する方法については、*In-System Programmability CD-ROM*に収録されているJam Playerのreadme.txtファイルを参照してください。
- TMSとTDIの信号がTCKの立ち上がりエッジに正しく同期しているが、TCKの立ち下がりエッジまで出力が変化しない。この状態では、出力値の読み出しタイミングがTCKの半クロック・サイクル分遅延される原因となっています。TDOの遷移をTCKの立ち上がりエッジで読み込むようになっている場合は、データが１クロック分オフセットされた状態で出力されることになります。
- アルテラは、出力の遷移をクロックに同期させるためにレジスタを使用することを推奨します。また、プロセッサによっては、出力信号の同期化をはかるためにデータ・ポートにレジスタを使用しているものがあります。例えば、PCのパラレル・ポートに対するリードとライトの動作は、レジスタに対するリードとライトによって実現されています。ただし、JTAGチェインに対するリードおよびライト動作を行う場合には、使用されるレジスタの数を正確に把握しておく必要があります。これらの使用されるレジスタの数が正しくカウントされていないと、期待値の出力が遅れたり、進んでしまう結果となります。

Jamファイルを使用して、Jam Playerが正しくポーティングされているかどうかをチェックすることが可能です。図１は、前に述べた３種類のポーティングの問題をデバッグするときに役立つJamファイルの例を示したものです。このサンプル・ファイルはアルテラのwebサイト、http://www.altera.comのliteratureのページからダウンロードすることができます。

図1　ポーティングの問題をデバッグするためのJamファイルの例（1/5）

```
NOTE JAM_VERSION "1.1 ";
NOTE DESIGN "IDCODE.jam version 1.4 4/28/98";
'################################################################################
'#This Jam File compares the IDCODE read from a JTAG chain with the
'#expected IDCODE. There are 5 parameters that can be set when executing
'#this code.
'#
'#COMP_IDCODE_[device #]=1, for example -dCOMP_IDCODE_9400=1
'#compares the IDCODE with an EPM9400 IDCODE.
'#PRE_IR=[IR_LENGTH] is the length of the instruction registers you want
'#to bypass after the target device. The default is 0, so if your
'#JTAG length is 1, you don't need to enter a value.
'#POST_IR=[IR_LENGTH] is the length of the instruction registers you
'#want to bypass before the target device. The default is 0, so if
'#your JTAG length is 1, you don't need to enter a value.
'#PRE_DR=[DR_LENGTH] is the length of the data registers you want
'#to bypass after the target device. The default is 0, so if your
'#JTAG length is 1, you don't need to enter a value.
'#POST_DR=[DR_LENGTH] is the length of the data registers you want
'#to bypass before the target device. The default is 0, so if your
'#JTAG length is 1, you don't need to enter a value.
'#Example: This example reads the IDCODE out of the second device in the
'#chain below:
'#
'#TDI -> EPM7128S -> EPM7064S -> EPM7256S -> EPM7256S -> TDO
'#
'#In this example, the IDCODE is compared to the EPM7064S IDCODE. If the JTAG
'#chain is set up properly, the IDCODEs should match.
'# C:\> jam -dCOMP_IDCODE_7064S=1 -dPRE_IR=20 -dPOST_IR=10 -dPRE_DR=2
'#-dPOST_DR=1 -p378 IDCODE.jam
'#
'#
'# Example: This example reads the IDCODE of a single device JTAG chain
'# and compares it to an EPM9480 IDCODE:
'#
'# C:\> jam -dCOMP_IDCODE_9480=1 -p378 IDCODE.jam
'################################################################################

'########################## Initialization ########################

BOOLEAN read_data[32];
BOOLEAN I_IDCODE[10] = BIN 1001101000;
BOOLEAN I_ONES[10] = BIN 1111111111;
BOOLEAN ONES_DATA[32]= HEX FFFFFFFF;
```

図1　ポーティングの問題をデバッグするためのJamファイルの例（2/5）

```
BOOLEAN ID_9320[32]          = BIN 10111011000000000100110010010000;
BOOLEAN ID_9400[32]          = BIN 10111011000000000000001010010000;
BOOLEAN ID_9480[32]          = BIN 10111011000000000001001010010000;
BOOLEAN ID_9560[32]          = BIN 10111011000000000110101010010000;
BOOLEAN ID_7032S[32]         = BIN 10111011000001001100000011100000;
BOOLEAN ID_7064S[32]         = BIN 10111011000000100110000011100000;
BOOLEAN ID_7128S[32]         = BIN 10111011000000010100100011100000;
BOOLEAN ID_7128A[32]         = BIN 10111011000000010100100011100000;
BOOLEAN ID_7160S[32]         = BIN 10111011000000000110100011100000;
BOOLEAN ID_7192S[32]         = BIN 10111011000001001001100011100000;
BOOLEAN ID_7256S[32]         = BIN 10111011000001101010010011100000;
BOOLEAN ID_7256A[32]         = BIN 10111011000001101010010011100000;

BOOLEAN COMP_9320_IDCODE     = 0;
BOOLEAN COMP_9400_IDCODE     = 0;
BOOLEAN COMP_9480_IDCODE     = 0;
BOOLEAN COMP_9560_IDCODE     = 0;
BOOLEAN COMP_7032S_IDCODE    = 0;
BOOLEAN COMP_7064S_IDCODE    = 0;
BOOLEAN COMP_7096S_IDCODE    = 0;
BOOLEAN COMP_7128S_IDCODE    = 0;
BOOLEAN COMP_7128A_IDCODE    = 0;
BOOLEAN COMP_7160S_IDCODE    = 0;
BOOLEAN COMP_7192S_IDCODE    = 0;
BOOLEAN COMP_7256S_IDCODE    = 0;
BOOLEAN COMP_7256A_IDCODE    = 0;
BOOLEAN COMP_7032AE_IDCODE   = 0;
BOOLEAN COMP_7064AE_IDCODE   = 0;
BOOLEAN COMP_7128AE_IDCODE   = 0;
BOOLEAN COMP_7256AE_IDCODE   = 0;
BOOLEAN COMP_7512AE_IDCODE   = 0;
INTEGER PRE_IR       = 0;
INTEGER PRE_DR       = 0;
INTEGER POST_IR      = 0;
INTEGER POST_DR      = 0;

BOOLEAN SET_ID_EXPECTED[32];
BOOLEAN COMPARE_FLAG1  = 0;
BOOLEAN COMPARE_FLAG2  = 0;
BOOLEAN COMPARE_FLAG   = 0;

' This information is what is expected to be shifted out of the instruction
' register

BOOLEAN expected_data[10] = BIN 0101010101;
BOOLEAN ir_data[10];
```

図1　ポーティングの問題をデバッグするためのJamファイルの例（3/5）

```
' These values default to 0, so if you have a single device JTAG chain, you do
' not have to set these values.

PREIR PRE_IR;
POSTIR POST_IR;
PREDR PRE_DR;
POSTDR POST_DR;

INTEGER i;

' ########################## Determine Action ########################

LET COMPARE_FLAG1= COMP_9320_IDCODE || COMP_9400_IDCODE || COMP_9480_IDCODE ||
  COMP_9560_IDCODE || COMP_7032S_IDCODE || COMP_7064S_IDCODE ||
  COMP_7096S_IDCODE || COMP_7032AE_IDCODE || COMP_7064AE_IDCODE ||
  COMP_7128AE_IDCODE;

LET COMPARE_FLAG2 = COMP_7128S_IDCODE || COMP_7128A_IDCODE ||COMP_7160S_IDCODE
  || COMP_7192S_IDCODE || COMP_7256S_IDCODE || COMP_7256A_IDCODE ||
  COMP_7256AE_IDCODE || COMP_7512AE_IDCODE;

LET COMPARE_FLAG = COMPARE_FLAG1 || COMPARE_FLAG2;
IF COMPARE_FLAG != 1 THEN GOTO NO_OP;

FOR i=0 to 31;
IF COMP_9320_IDCODE   == 1 THEN LET SET_ID_EXPECTED[i] = ID_9320[i];
IF COMP_9400_IDCODE   == 1 THEN LET SET_ID_EXPECTED[i] = ID_9400[i];
IF COMP_9480_IDCODE   == 1 THEN LET SET_ID_EXPECTED[i] = ID_9480[i];
IF COMP_9560_IDCODE   == 1 THEN LET SET_ID_EXPECTED[i] = ID_9560[i];
IF COMP_7032S_IDCODE  == 1 THEN LET SET_ID_EXPECTED[i] = ID_7032S[i];
IF COMP_7064S_IDCODE  == 1 THEN LET SET_ID_EXPECTED[i] = ID_7064S[i];
IF COMP_7128S_IDCODE  == 1 THEN LET SET_ID_EXPECTED[i] = ID_7128S[i];
IF COMP_7128A_IDCODE  == 1 THEN LET SET_ID_EXPECTED[i] = ID_7128A[i];
IF COMP_7160S_IDCODE  == 1 THEN LET SET_ID_EXPECTED[i] = ID_7160S[i];
IF COMP_7192S_IDCODE  == 1 THEN LET SET_ID_EXPECTED[i] = ID_7192S[i];
IF COMP_7256S_IDCODE  == 1 THEN LET SET_ID_EXPECTED[i] = ID_7256S[i];
IF COMP_7256A_IDCODE  == 1 THEN LET SET_ID_EXPECTED[i] = ID_7256A[i];
IF COMP_7032AE_IDCODE == 1 THEN LET SET_ID_EXPECTED[i] = ID_7032AE[i];
IF COMP_7064AE_IDCODE == 1 THEN LET SET_ID_EXPECTED[i] = ID_7064AE[i];
IF COMP_7128AE_IDCODE == 1 THEN LET SET_ID_EXPECTED[i] = ID_7128AE[i];
IF COMP_7256AE_IDCODE == 1 THEN LET SET_ID_EXPECTED[i] = ID_7256AE[i];
IF COMP_7512AE_IDCODE == 1 THEN LET SET_ID_EXPECTED[i] = ID_7512AE[i];


NEXT I;
```

図1　ポーティングの問題をデバッグするためのJamファイルの例（4/5）

```
' ######################### Actual Loading ########################

IRSTOP IRPAUSE;
STATE RESET;
IRSCAN 10, I_IDCODE[0..9], CAPTURE ir_data[0..9];
STATE IDLE;

DRSCAN 32, ONES_DATA[0..31], CAPTURE read_data[0..31];

' ######################### Printing ########################

PRINT "EXPECTED IRSCAN : 1010101010";
PRINT "ACTUAL IRSCAN: ",ir_data[0], ir_data[1], ir_data[2], ir_data[3],
  ir_data[4], ir_data[5], ir_data[6], ir_data[7], ir_data[8], ir_data[9];

PRINT "";PRINT "EXPECTED IDCODE : ", SET_ID_EXPECTED[0], SET_ID_EXPECTED[1],
  SET_ID_EXPECTED[2], SET_ID_EXPECTED[3], SET_ID_EXPECTED[4],
  SET_ID_EXPECTED[5], SET_ID_EXPECTED[6], SET_ID_EXPECTED[7],
  SET_ID_EXPECTED[8], SET_ID_EXPECTED[9], SET_ID_EXPECTED[10],
  SET_ID_EXPECTED[11], SET_ID_EXPECTED[12], SET_ID_EXPECTED[13],
  SET_ID_EXPECTED[14], SET_ID_EXPECTED[15], SET_ID_EXPECTED[16],
  SET_ID_EXPECTED[17], SET_ID_EXPECTED[18], SET_ID_EXPECTED[19],
  SET_ID_EXPECTED[20], SET_ID_EXPECTED[21], SET_ID_EXPECTED[22],
  SET_ID_EXPECTED[23], SET_ID_EXPECTED[24], SET_ID_EXPECTED[25],
  SET_ID_EXPECTED[26], SET_ID_EXPECTED[27], SET_ID_EXPECTED[28],
  SET_ID_EXPECTED[29], SET_ID_EXPECTED[30], SET_ID_EXPECTED[31];

PRINT "ACTUAL IDCODE : ", READ_DATA[0], READ_DATA[1], READ_DATA[2],
  READ_DATA[3], READ_DATA[4], READ_DATA[5], READ_DATA[6], READ_DATA[7],
  READ_DATA[8], READ_DATA[9], READ_DATA[10], READ_DATA[11], READ_DATA[12],
  READ_DATA[13], READ_DATA[14], READ_DATA[15], READ_DATA[16], READ_DATA[17],
  READ_DATA[18], READ_DATA[19], READ_DATA[20], READ_DATA[21], READ_DATA[22],
  READ_DATA[23], READ_DATA[24], READ_DATA[25], READ_DATA[26], READ_DATA[27],
  READ_DATA[28], READ_DATA[29], READ_DATA[30], READ_DATA[31];

GOTO END;
```

図1　ポーティングの問題をデバッグするためのJamファイルの例（5/5）

```
' ######################### If no parameters are set ########################

NO_OP: PRINT "jam [-d<var=val>] [-p<port>] [-s<port>] IDCODE.jam";
PRINT "-d : initialize variable to specified value";
PRINT "-p : parallel port number or address <for ByteBlaster>";
PRINT "-s : serial port name <for BitBlaster>";
PRINT " ";
PRINT "Example: To compare IDCODE of the 4th device in a chain of 5 Altera ";
PRINT "devices with EPM7192S IDCODE";
PRINT " ";
PRINT "jam -dCOMP_7192S_IDCODE=1 -dPRE_IR=10 -dPOST_IR=30 -dPRE_DR=1";
PRINT "dPOST_DR=3 -p378 IDCODE.jam";
PRINT " ";

END:

EXIT 0;
```

インサーキット・テスタによるISP

このセクションでは、インサーキット・テスタを使用したISP対応デバイスのプログラミングに関連する事項について解説します。

“F” デバイスの使用とNon-“F” デバイスの使用

MAXデバイスは、固定アルゴリズム（"F"コード付きのデバイス用）またはブランチを必要とするアルゴリズム（"F"コードの付いていないデバイス用）のいずれかでプログラムされます。SVFや、Hewlett-Packard社のパターン・キャプチャ・フォーマット(.pcf）、DTS、ASCなど、ほとんどのインサーキット・テスタのファイル・フォーマットは、固定または固有のアルゴリズムにのみ対応したものとなっており、ブランチのない1種類のみのアルゴリズムだけをサポートしています。MAX+PLUS IIソフトウェアは、"F"コード付きのデバイス用のSVFファイルを生成することができます。SVFファイルのアルゴリズムは固定になっているため、"F"コード付きのデバイスのプログラムに、常にこのファイルが使用できるようになっています。

アルテラは、インサーキット・テスタによる"F"コードなしのデバイスのプログラミングを推奨しておりません。"F"コードの付いていないデバイスには、デバイスから読み込まれるプログラミング・パルス時間、イレーズ・パルス時間、製造メーカのシリコンIDの3種類の変数に応じたブランチ動作が必要です。これら3種類の変数は、アルテラのすべての"F"コードの付いていないデバイスにプログラムされています。"F"コード付きのデバイスのみを使用することで、これらの変数が変更されている場合でも問題の発生を防ぐことができます。

ファイルあたりの最大ベクタ数

テスト・ポイントにピンを接触させて試験を行う「bed of nail」タイプのイン・サーキット・テスタでは、一般的にイン・システム・プログラミングのために非常に多数のベクタが必要となります。このファイル・サイズがテスタに提供されているメモリ容量よりも大きくなるときは、ファイルを複数の小さなサイズに分割する必要があります。アルテラが提供しているsvf2pcfのユーティリティ・プログラムは１つのSVFファイルを複数の小さなファイルに自動的に分割します。また、このユーティリティを使用することによって、ファイルあたりの最大ベクタ数、またはそのデフォルトを設定することができます。１つのファイルに多数のベクタが含まれていた場合は、エラー・メッセージが表示されます。このようなエラーが表示されたときは、ファイルあたりのベクタ数を減少させる必要があります。

プルアップおよびプルダウン抵抗

テスタによっては、多くの信号にプルダウン、またはプルアップ抵抗を付加する必要があります。詳細は、各イン・サーキット・テスタのベンダにコンタクトして確認してください。

まとめ

このアプリケーション・ノートに掲載されている情報は、アルテラの開発経験やユーザで発生した問題点を解決した実例などをベースにしたものとなっています。イン・システム・プログラミングに関連した問題を解決する必要がある場合は、日本アルテラの応用技術部へご相談ください。また、アルテラのwebサイト、http://www.altera.comにあるAtlasのセクションに掲示されているデータベースも確認してください。

日本アルテラ株式会社

〒163-0436
東京都新宿区西新宿2-1-1
新宿三井ビル私書箱261号
TEL. 03-3340-9480　FAX. 03-3340-9487
http://www.altera.com/japan/
E-mail: japan@altera.com

本社 Altera Corporation

101 Innovation Drive,
San Jose, CA 95134
TEL : (408) 544-7000
http://www.altera.com