ISPを使用するための ガイドライン

1998年 8 月 ver.1.01 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システム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MAX® 9000（MAX 9000Aを含む）、MAX 7000S、MAX 7000Aファミリの各デバイスは、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ボードのファンクション・テストをひとつの工程で行うことが可能となり、テスト時間とアッセンブル・コストを削減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
- 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。ユーザ・モードにおいては、これらの条件が守られていない場合でも、アルテラのデバイスは一般的に正常に動作しますが、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ことが必要です。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が維持されるように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。オシロスコープのトリガ・レベルを4.75Vおよび5.25Vに設定し、デバイスのこれらのピンにプローブを当ててVCCINTとグランド間の電圧を測定します。ここで、オシロスコープが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で、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（GND−1V）から（V_{CCINT}+1V）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。

詳細については、**1998年版データブック**に掲載されている「*Operating Requirements for Altera Devices*」（日本語版データシート「*アルテラ・デバイスの使用上の注意*」）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てしまう可能性もあります。

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

イン・システム・プログラミングで発生するほとんどの不良は、TCK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信号が遷移することによって、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。不適切なクロックの供給はステート・マシンを規定されていない不定のステートに遷移させたり、イン・システム・プログラミング不良を発生させる原因になります。

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他のすべてのJTAG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・アウトを持っています。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良モードとしては、「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。

ダウンロード・ケーブルによるプログラミング

BitBlaster™、ByteBlaster™、またはByteBlasterMV™のダウンロード・ケーブルを使用しているときに、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、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。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・プログラミングの問題の発生原因となります。

BitBlaster、ByteBlaster、またはByteBlasterMVの各ダウンロード・ケーブルの詳細については、「*BitBlaster Serial Download Cable*」、「*ByteBlaster Parallel Port Download Cable*」、「*ByteBlasterMV Parallel Port 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 7000A *注(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。
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。	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。

注：
(1)　MAX 7000Aデバイスに関する情報は暫定仕様に基づくものです。

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

MAX 7000SデバイスおよびMAX 7000Aデバイスの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を規定しておく必要があります。「*Enable JTAG Support*」のオプションをONに設定した場合は、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デバイスおよび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を最初に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。

JTAG回路をISPとBSTの実行時にイネーブルにし、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MAX 7000S、MAX 7000A、MAX 9000、MAX 9000Aの各デバイスでは、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なければなりません。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デバイスでJTAG回路を永久にディセーブルする場合と同じように、TCKをLowにしてTMSをHighにするか、TCKがHighになる前にTMSをHighにすることでJTAG回路をディセーブルすることができます。

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}の2種類の電源が使用されます。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ピンの状態によって、「test-logic-reset」のステートが保持されなかった場合に、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、信号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・マシンが不定の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、JTAG信号がフローティングのままとなります。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が行われる結果になることがあります。このため、すべてのJTAG信号は、4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チェイン内の次のデバイスに電源が投入された状態になったときでも（3.3V動作デバイスにV_{CCIO}が供給された状態）、この次のデバイスが「test-logic-reset」のステートを保っ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。したがって、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とMAX 7000Aの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の値は、「*MAX 7000 Programmable Logic Device Family*」、および「*MAX 7000A Programmable Logic Device Family*」のデータシートの中で、それぞれ示されています。

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へ大きな電流が流れ（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投入ができないような不良が発生することがあります。

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、下記の項目が上げられます。

- 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・ケーブルがパラレル・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。

BitBlaster、ByteBlaster、ByteBlasterMVの各ダウンロード・ケーブルのインストール方法については、「*BitBlaster Serial Download Cable*」、「*ByteBlaster Parallel Port Download Cable*」、「*ByteBlasterMV Parallel Port 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

TDOポートが接続されていないと、エラーが発生します。デバイスをプログラムするためには、TDOポートをチェイン内の次のデバイスと接続する必要があります。

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たエラーの発生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良が発生します。

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（.jam）またはシリアル・ベクタ・フォーマット・ファイル（.svf）を通じてターゲット・デバイスにIDCODEのインストラクションをロードします。 JamファイルまたはSVFファイルを使用してIDCODEのインストラクションをロードし、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。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。

図1　IDCODEを読み出すためのJamファイルの例

```
'This example reads the IDCODE for the second device in the chain below:
'TDI -> EPM7128S -> EPM7064S -> EPM7256S -> EPM7256S -> TDO

'The device IDCODE can be found in AN 39: IEEE 1149.1 (JTAG) Boundary-Scan Testing
'in Altera Devices

BOOLEAN expected_data[32] = BIN 10111011000000100110000011100000;

'EPM7064S IDCODE

PREIR 20;             ' This command bypasses the 2 devices after EPM7064S
POSTIR 10;            ' This command bypasses the 1st device
PREDR 2;              ' This command bypasses the 2 devices after EPM7064S
POSTDR 1;             ' This command bypasses the 1st device
STATE RESET;
IRSCAN 10, 057;
DRSCAN 32, FFFFFFFF COMPARE expected_data[0..31], FFFFFFFF, error;
IF error==0 then PRINT "IDCODE read successfully";
IF error==1 then PRINT "IDCODE not read successfully";
```

図2はIDCODEを読み出すためのSVFファイルの例を示したものです。デバイスが大きなチェインの一部となっている場合は、このファイルに適切なヘッダおよびトレイラ情報を追加する必要があります。

図2　IDCODEを読み出すためのSVFファイルの例

```
! If the device is in a chain of devices, HIR, HDR, TIR, and TDR
! statements must be added here.

STATE RESET;
SIR 10 TDI (057);
SDR 32 TDI (FFFFFFFF) TDO (<add appropriate IDCODE here>) MASK (FFFFFFFF);

! If the IDCODE is wrong, the previous line gives compare errors when executed
```

これらのファイルの例を拡張して、IDCODEのインストラクションをデバイスから読み出されるビットをチェックして、さらに詳細なエラー・チェックを行うこともできます。これらのファイルの拡張バージョンについてはISPembed@altera.com、ISPATE@altera.comまたは日本アルテラの応用技術部へお問い合わせください。Jam Playerのソース・コードと共に提供されているreadme.txtファイルにはJam Playerのポーティングに関する詳細な情報が提供されています。

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

Bytecode Jam 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ファイルのアップデートを目的にしたフィールドでのアップグレードが簡略化されます。Jam Playerのメモリとしては、ROMとダイナミック・メモリ（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、ダイナミック・メモリは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*）をご覧ください。

Jam Playerのポーティング

アルテラのJam Player（BytecodeバージョンおよびASCIIバージョン）は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、Bytecode Jam 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には、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。

- ByteBlasterまたはByteBlasterMVのダウンロード・ケーブルがプロセッサと接続されているとき、TDOの値が逆極性で読み出されている。この問題は、パラレル・ポートのレジスタからデータが反転した極性で読み出されるようになっている場合に発生します。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readme.txtの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

の読み出しのタイミングがTCKの半クロック・サイクル分遅延させ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3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ウェッブ・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3　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
```

図3　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)

```
'#
'#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;

BOOLEAN ID_9320[32]    = BIN 10111011000000000100110010010000;
BOOLEAN ID_9400[32]    = BIN 10111011000000000000001010010000;
BOOLEAN ID_9480[32]    = BIN 10111011000000000000101001010000;
BOOLEAN ID_9560[32]    = BIN 10111011000000000011010101010000;
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_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;
```

図３　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）

```
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 is what is expected to be shifted out of the instruction register
BOOLEAN expected_data[10] = BIN 0101010101;

BOOLEAN ir_data[10];
' 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;

LET COMPARE_FLAG2 = COMP_7128S_IDCODE || COMP_7128A_IDCODE || COMP_7160S_IDCODE
   || COMP_7192S_IDCODE || COMP_7256S_IDCODE || COMP_7256A_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];
```

図3　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)

```
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];

NEXT I;

' ######################### 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3　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 -
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。シリアル・ベクタ・フォーマット（.svf）、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のバージョン8.2以降のソフトウェアは、"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、今後供給される"F"コード付きのデバイスのプログラムには、これらのSVF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は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・パルス幅、イレーズ・パルス幅、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の動作が必要です。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-nails」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場合は、これを複数の小さなサイズのファイルに分割する必要があります。例えば、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・プログラム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値を設定することができます。１つのファイルに処理できない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ことが必要です。

HP 3070テスタ

アルテラのユーティリティ・プログラム、svf2pcfは、ひとつのHP PCFファイルを複数の小さなファイルに自動的に分割して、HPのテスタのメモリ容量でイン・システム・プログラミングに要求されるベクタ数をサポートできるようにします。

テスタが各ベクタ・ファイルをデバイスに供給するときに小さな遅延が発生します。このインターバルに、HPのテスタはすべてのピンをトライ・ステートにし、すべてのピンを短時間でLowへドライブしてから、次のPCFファイルの最初のベクタをドライブするようになっています。

このインターバルの期間に意図しないTCKの遷移の発生を防ぐためには、下記の２段階の対策が必要です。

- 各PCFのベクタ・ファイルはすべての信号がLowで開始され、Lowレベルで終了するようになっている必要があります。アルテラのユーティリティ・プログラム、svf2pcfはこの処理を自動的に実行します。
- TCK信号にはプルダウン抵抗を付加する必要があります。この抵抗はボード上で、５ページの「JTAG回路をISPとBSTの実行時にイネーブルし、ユーザ・モードでディセーブルする方法」に記述されている推奨事項が満足されるような値になっている必要があります。この抵抗をボード上に実装できない場合は、これをテスタ側に追加することもできます。この抵抗値はチェインに接続されるデバイスの数と各デバイスの負荷に応じて異なります。ただし、プルダウン抵抗をボード上に実装する場合は1kΩから5kΩが、テスタ側に実装する場合は500Ωから1kΩがひとつの目安となります。

まとめ

この記事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、ISPembed@altera.com、ISPATE@altera.comまたは日本アルテラの応用技術部へご連絡ください。また、アルテラのウェッブ・サイト、http://www.altera.comの中に提供されている「Altera ISP Support Program」のページを参照してください。

更新記録

このアプリケーション・ノート、AN 100のバージョン1.01は、これまでのバージョン1.0を改訂したものです。

このバージョン1.01には、４ページに示されている表１、「IEEE Std. 1149.1回路のディセーブル方法」に関する訂正された情報が含まれています。

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/

本社 Altera Corporation

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