この翻訳版ドキュメントのメンテナンスは終了しております。

この文書には、古いコンテンツや商標が含まれている場合があります。

最新情報につきましては、次のリンクから英語版の最新資料をご確認ください。

https://www.intel.com/content/www/us/en/programmable/documentation/lit-index.html

Please take note that this document is no longer being maintained. It may contain legacy content and trademarks which may be outdated.

Please refer to English version for latest update at
https://www.intel.com/content/www/us/en/programmable/documentation/lit-index.html

アルテラ CPLD における Agilent 3070 テスタを使用したイン・システム・プログラミング

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。

AN-628-1.0 アプリケーション・ノート

このアプリケーション・ノートでは、Agilent 3070 テスト・システムを使用して、アルテラの MAX® II および MAX V デバイスのプログラミング時間を短縮する方法について説明します。また、このアプリケーションノートでは、Agilent Technologies の PLD（プログラマブル・ロジック・デバイス（PLD））ISP（イン・システム・プログラミング）ソフトウェアを使用する場合としない場合における Agilent 3070 テスタの開発フローについて説明します。

ISP は、PLD の中心的な機能であり、システム設計者およびテスト・エンジニアは、PLD プログラミングをボード・レベル・テストに統合して、コスト面で大きなメリットを享受することができます。これらのメリットには、事前にプログラムされたデバイスの在庫削減、コスト低減、取り扱い時のデバイスの損傷の減少、技術変更における柔軟性の向上などがあります。

アルテラは、他の ISP 対応のデバイスと共に Agilent 3070 テスト・システムを使用して、MAX II および MAX V デバイスをプログラムするソリューションを提供しています。

☞ Agilent 3070 テスタを使用して、MAX II および MAX V デバイスを他のファミリのデバイス・ファミリと共にプログラムする場合は、Agilent 3070 テスタでチェイン内にあるすべてのデバイスがプログラムできるように確認してください。

この章は、以下の項で構成されています。

101 Innovation Drive
San Jose, CA 95134
www.altera.com

2010 年 12 月
Altera Corporation

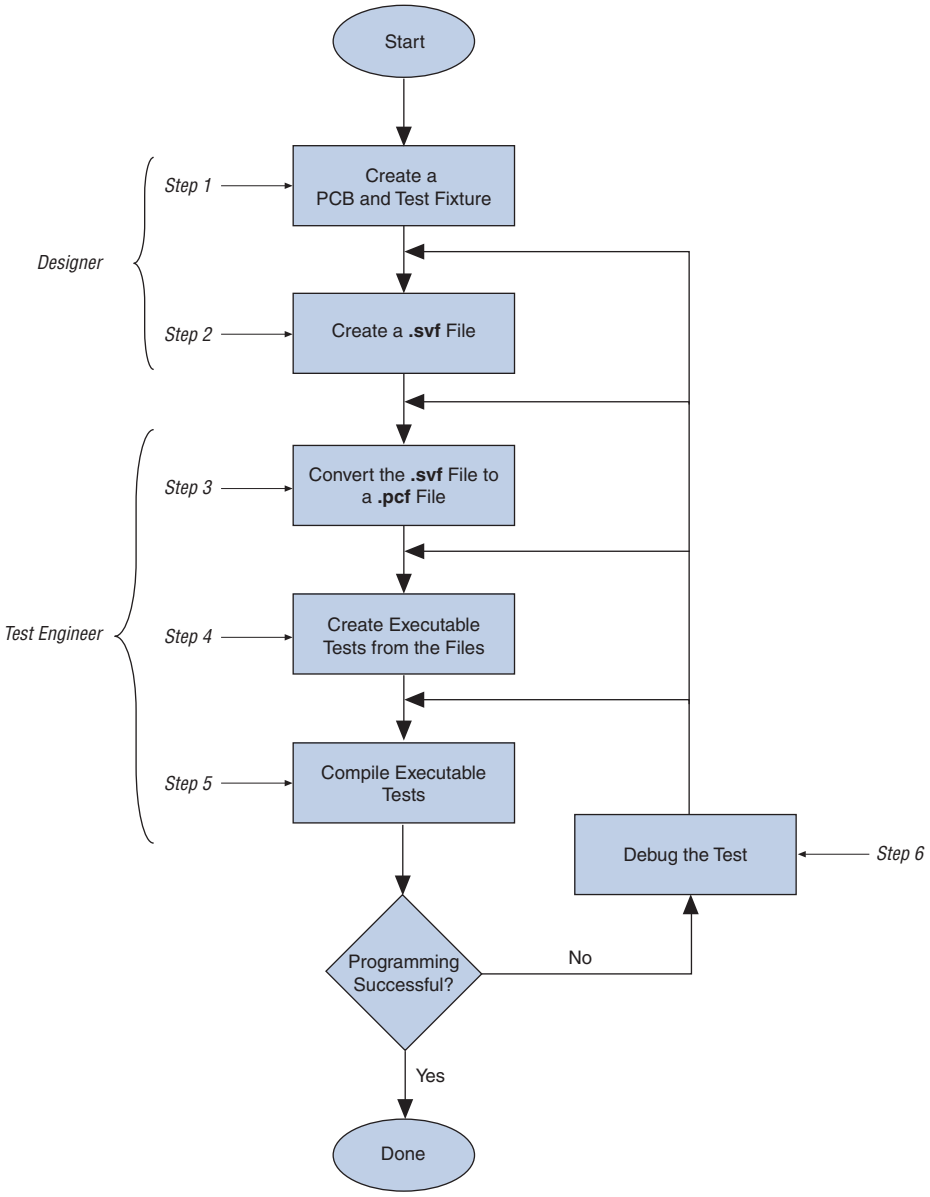
Subscribe

PLD ISP ソフトウェアを使用しない Agilent 3070 テスタの開発フロー

図 1 に、PLD ISP ソフトウェアを使用せずに Agilent 3070 テスタでデバイスをプログラムする（(Serial Vector Format「**.svf**」ファイルを使用）のに必要なステップを示します。

次の項では、図 1 に示す各ステップについて説明します。

図 1. SVF ファイルを使用した ISP に対する Agilent 3070 開発フロー（PLD ISP は不使用）



ステップ 1: PCB およびテスト治具の作成

ISP を使用する場合、最初のステップはボードのレイアウトおよびテスト治具の作成です。

PCB およびテスト治具の作成

以下の情報は、PCB デザインの問題における重要事項に関するものです。

- TCK 信号配線パターンをクロック・ツリーとして慎重に扱う必要があります。TCK は、デバイスの JTAG チェイン全体に対するクロックです。これらのデバイスは TCK 信号でエッジ・トリガされるため、この信号を高周波ノイズから保護し、良好なシグナル・インテグリティを確保する必要があります。信号が該当するデバイス・ファミリのデータシートに記載されたパルス立ち上がり時間（t_R）およびパルス立ち下がり時間（t_F）パラメータに適合することを確認してください。
- TCK 信号にプルダウン抵抗を追加します。TCK 信号は、パターン・キャプチャ・フォーマット（**.pcf**）ダウンロード間では、プルダウン抵抗を介して Low に保持する必要があります。Agilent 3070 ドライバは、テスト間ではハイ・インピーダンスになり、次の **.pcf** が適用されると短時間だけ Low にドライブするため、TCK は Low に保持しなければなりません。TCK ラインがフロートすると、プログラミング・データ・ストリームが破壊され、デバイスは正しくプログラムされません。

  .pcf ファイルのダウンロードについて詳しくは、「ステップ 2: .svf ファイルの作成」を参照してください。

- テスト治具のネイルに対して、VCC および GND テスト・アクセス・ポイント（TAP）を設けます。動作中には、PCB 動作が乱れないように、十分な TAP が必要です。TAP が足りないと、システムのノイズが増大し、JTAG スキャンが中断する可能性があります。
- システム・ノイズを低減するために、オンボード・オシレータをオフにします。プログラミング中に、オンボード・オシレータを電気的にオフにする機能が必要です。
- プログラミング中に外部抵抗を追加して、定義済みロジック・レベルに出力をプルします。

  プログラミング中に、出力ピンははトライ・ステートになり、内部ウィーク抵抗でプルアップされます。ただし、アルテラは定義済みレベルを必要とする信号は、外部抵抗を使用して外部から強制的に適切なレベルに設定することを推奨しています。

ISP 用ボード・デザインについて詳しくは、*「AN 100: In-System Programmability Guidelines」*を参照してください。

治具の作成

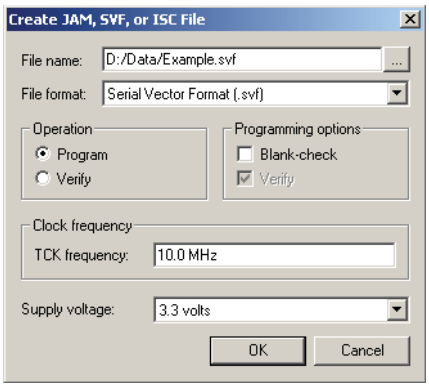
ISP を成功させるには、クリーンなインタフェースを提供するために、テスト治具内で短いワイヤを使用して、TCK の接続性を向上させます。ワイヤを長くすると、システム内部に誘導ノイズが誘発され、プログラミングが中断することがあります。TCK を接続するワイヤは 1 インチ未満にしてください。テスト治具のレイアウトと作成を管理するには、Agilent Fixture Consultant を使用します（*Agilent Board Test Family Manual* を参照）。

ステップ 2: .svf ファイルの作成

Quartus® II ソフトウェアは、1 つまたは複数のデバイスをプログラムするための **.svf** f ファイルを生成します。複数の MAX II および MAX V デバイス・ファミリをターゲットとする場合、Quartus II ソフトウェアは、デバイスを同時にプログラムするための 1 つの **.svf** ファイルを自動的に生成します。したがって、すべてのデバイスのプログラミング時間は、IEEE Std. 1149.1 JTAG チェイン内の最大の CPLD デバイスのプログラミング時間とほぼ等しくなります。

図 2 に、**.svf** ファイルの生成に使用する **Create JAM, SVF, or ISC File** ダイアログ・ボックス（File メニュー）を示します。

図 2. Create JAM, SVF, or ISC File ダイアログ・ボックス



.svf ファイルを作成する前に、Quartus II Programmer を開いて、チェイン内のすべてのデバイス用の Programmer Object File（**.pof**）をプログラマに追加します。各 **.pof** は、それぞれのターゲット・デバイスに対応します。

Create JAM, SVF, or ISC File ダイアログ・ボックスで、**TCK frequency** ボックス内の値は、TCK がテスト中に動作する周波数と一致する必要があります。テストで使用される値と異なる周波数を入力すると、プログラミングが失敗したり、プログラミング時間が極端に長くなることがあります。

また、プログラム操作および検証操作のどちらを実行するかを選択でき、さらにオプションで **Programming options** をオンにすることにより、デバイスの検証およびブランク・チェックを選択できます。アルテラは、検証ベクタを含む **.svf** ファイルの生成を推奨しています。これによって、プログラミングの失敗が識別され、限定された追加プログラミング時間が使用されます。必要な **.svf** ファイルは、プログラミング対象となるボードおよびアルテラ・デバイスのスキャン・チェイン・トポロジーに基づいて生成できます。**svf** ファイルが生成された後、テスト・エンジニアはこれを開発に使用できます。

デバイスを個別にプログラムする場合、チェイン内のアルテラ・デバイスごとに、個別に **.svf** ファイルを生成できます。チェイン内の 1 つのデバイス用に SVF ファイルを作成する場合は、そのデバイスに **.pof** を指定し、Programmer での **Add Device** を選択することで残りのデバイスは ***<none>*** に設定したままにします。これらのデバイスはプログラミング中にはバイパスされます。ターゲットとするすべてのデバイスに対する **.svf** ファイルを作成するまで、このプロセスを繰り返します。

ステップ 3: .svf ファイルを .pcf ファイルに変換

Agilent 3070 テスタで使用するには、アルテラ **svf2pcf** 変換ユーティリティで **.svf** ファイルを **.pcf** ファイルに変換します。**svf2pcf** ユーティリティは、1 つのデバイス・チェインに対して複数の **.pcf** ファイルを作成できます。このユーティリティを実行すると、ファイルごとにベクタ数を指定できます。結果として得られたファイルで使用されるメモリ容量は、データによって異なります。

Agilent 3070 デジタル・コンパイラはベクタの繰り返しパターンを検索し、ディレクトリを最適化します。さらに、テスタ・コントロール・カード上の RAM に順番を付けて、ファイルを再ロード前のベクタの最大数を適用します。コンパイル済み **.pcf** ファイル内のベクタ数は、ターゲット・デバイスのサイズと集積度によって、10 万 ～ 100 以上になります。

svf2pcf 変換ユーティリティは、アルテラ・ウェブサイトの Agilent ISP Support のページからダウンロードできます。

ステップ 4: ファイルからの実行可能テストの作成

Agilent 3070 テスタを使用して、デバイスのチェインをプログラムするためのデジタル・テストを作成するには、次の項で説明するステップに従ってください。

a. 5 ページの「ターゲット・デバイスまたはスキャン・チェインのライブラリの作成」

b. 6 ページの「Test Consultant の実行」

c. 6 ページの「デジタル・テストの作成」

d. 6 ページの「テスト用ワイヤリスト情報の作成」

e. 7 ページの「テスト・プランの修正」

ターゲット・デバイスまたはスキャン・チェインのライブラリの作成

ボード用の初回プログラム開発では、ISP バウンダリ・スキャン・チェイン・インタフェース用のセットアップ専用ノード・テスト・ライブラリを作成します。テスト・ライブラリにより、ターゲット・デバイスをプログラムするためのテスト治具に、Agilent 3070 テスタ・リソースが確実に予約されます。ボード上に 1 つのターゲット・デバイスしかなく、かつそのデバイスが分離されているバウンダリ・スキャン・チェインの一部分でない場合はピン・ライブラリを使用し、それ以外の場合はノード・ライブラリを使用します。ピン・ライブラリを使用する場合は、すべてのデバイス・ピンを記述する必要があります。テスト・ライブラリにはテスト・ベクタを含めないでください。

以下のコード例は、セットアップ専用ノード・テスト・ライブラリを示します。

```
!Setup only test for the boundary scan chain
assign  TCK  to  nodes   "TCK"! Node name for the TCK pin
assign  TMS  to  nodes   "TMS"! Node name for the TMS pin
assign  TDI  to  nodes   "TDI"! Node name for the TDI pin
assign  TDO  to  nodes   "TDO"! Node name for the TDO pin
inputs  TCK, TMS, TDI
outputs TDO
pcf order is TCK, TMS, TDI, TDO! The order is defined by the program
that
    ! generates the PCF files.
```

☞ TCK および TMS バウンダリ・スキャン・ノードを Board Consultant で CRITICAL としてマークします。このクリティカル属性は、テスト治具でのノードのワイヤ長を最小にします。

Test Consultant の実行

Test Consultant を実行して、新しいボード開発用のすべてのファイルを作成します。Test Consultant は、このセットアップ専用テスト・ライブラリを使用して実行を終了すると、正しい治具配線情報とともに実行可能テスト（ベクタなし）を作成します。このファイルをテンプレートとして使用して、実行可能テストのソース・コードを作成します。

デジタル・テストの作成

実行可能テンプレートを希望のプログラム名にコピーすることで、デバイスのプログラムに必要なデジタル・テストを作成します。例えば、**svf2pcf** が 4 つの **.pcf** ファイルを作成した場合は、デジタル・ディレクトリ内の 4 つの実行可能テスト（prog_a、prog_b、prog_c、prog_d など）にテンプレート・ファイルをコピーします。

これらのテスト名を **testorder** ファイルに追加し、以下の構文を使用してこれらに permanent マークを付けます。

```
test digital "prog_a";  permanent
test digital "prog_b";  permanent
test digital "prog_c";  permanent
test digital "prog_d";  permanent
```

テスト用ワイヤリスト情報の作成

これらの実行可能テストをコンパイルして、テストのセットアップ専用バージョン用にオブジェクト・ファイル（「テスト・プランの修正」を参照）を生成します。**Module Pin Assignment** を実行して、必要なエントリを **wirelist** ファイル内に作成します。

次に、ターゲット・デバイスをプログラムするためのベクタが含まれるように、実行可能テストを修正します。実行可能テストで include ステートメントを使用するか、ベクタをファイルにマージできます。include ステートメントには以下の構文を使用します。これは、実行可能テストの最後のステートメントでなければなりません。

```
include "pcf1"
```

.pcf ファイルは、デジタル・ディレクトリに存在し、またデジタル・ファイルでなければなりません。デジタル・ファイルが正しいディレクトリに存在するように、BT-Basic コマンド・ラインで以下のコマンドを実行します。

```
load  digital  "digital/pcf1"  |  re-save
```

また、シェル・プロンプトで chtype コマンドを使用して、ファイルの位置を確認することもできます。

```
chtype  -n6  digital/pcf1
```

各 **.pcf** ファイルについて、このステップを繰り返します。

テスト・プランの修正

以下の構文を使用して、テスト・プランにテスト・ステートメントを追加します。

```
test  "digital/prog_a" ! First program file
test  "digital/prog_b" ! Second program file
test  "digital/prog_c" ! Third program file
test  "digital/prog_d" ! Fourth program file
```

テストの実行は、**svf** ファイルが分割された順番と同じにします。例えば、**svf** ファイルが 4 つのファイル（**pcf1**、**pcf2**、**pcf3**、**pcf4**）に分割された場合、テストを分割の順番で実行しなければなりません（prog_a、prog_b、prog_c、prog_d の順）。この順序に従わなければ、デバイスは正しくプログラムできません。

ステップ 5: 実行可能テストのコンパイル

アルテラは、BT-Basic または UNIX シェルを使用したバッチ起動式コンパイルを推奨しています。BT-Basic で以下のバッチ・ファイル・コードを参照してください（ターゲット・デバイスをプログラムするための 4 つの実行可能テストとデバッグ・オブジェクト・コードの生成を仮定しています）。

```
compile  "digital/prog_a" ;  debug
compile  "digital/prog_b" ;  debug
compile  "digital/prog_c" ;  debug
compile  "digital/prog_d" ;  debug
```

後で技術的変更が発生しても対応できるように、このファイルはボード・ディレクトリに保存してください。対応するシェル・スクリプトを参照してください（D オプションでデバッグ情報を生成）。

```
dcomp  -D  digital/prog_a
dcomp  -D  digital/prog_b
dcomp  -D  digital/prog_c
dcomp  -D  digital/prog_d
```

☞ ソース・ファイルに含まれる **.pcf** ベクタ数、コントローラのタイプ、およびコントローラの負荷によっては、コンパイル時間が長くなることがあります。アルテラは、バッチ・ファイルを使用して、ISP テストのコンパイルを自動化することを推奨しています。

定義されているアルテラ・デバイスを含むバウンダリ・スキャン・チェインを使用する場合、.pcf ファイルのベクタが JTAG インタフェースに適用されている後に、定義済みのアルテラ・デバイスのみプログラムされます。

ステップ 6: テストのデバッグ

実行可能テストが作成されると、テスト・システムのデバッグが可能になります。適用されたベクタ・セットにより、デバイスのコンテンツを検証するとデバイスが正しくプログラムされていることが確認できます。プログラミング・アルゴリズムは、TDO ピンを使用してデバイスからのビットストリームをチェックします。どのベクタも予想値に一致しない場合にはテストは失敗し、以下の 2 つのうちのいずれかを示します。

- デバイス ID が予想された ID と一致しない。最初のテストの開始時に失敗する場合は、明らかにこれが原因です。
- デバイスのプログラミングが失敗した。

各ベクタを調べて失敗の原因を特定すること必要はありません。デバイスのプログラミングが失敗する場合は、以下のトラブルシューティング・ガイドラインに従ってください。

- テスト治具のプルダウン抵抗をチェックします。ボード上で TCK ピンにプルアップ抵抗を配置した可能性があります。プルダウン抵抗が大きすぎる場合、TCK ピンはロジック Low に対するデバイスのスレッショルドを超えることがあります。抵抗値を適切に調整します。入力ロジック・レベルの仕様については、該当するデバイス・ファミリのデータシートを参照してください。
- TCK ピンで過電力エラーが発生した場合は、抵抗値をチェックします。抵抗値が小さすぎるために、テスト・システムが長い間バック・ドライブできないことが原因と考えられます。
- テストの実行順序が正しいことを確認します。順序がバラバラでテストを実行すると、プログラミング情報が不正になります。また、同じテストを連続 2 回実行すると、ターゲット・デバイスが順不同になり、正しいプログラミング情報を受け入れません。
- 実際のベクタが入力ピン（TCK、TMS、および TDI）の予想値と一致するようにします。予想値が一致しない場合は、テストを再コンパイルする必要があります。
- テストにおける **.pcf** の順序のステートメントが、4 ページの「ステップ 2: .svf ファイルの作成」で生成された **.pcf** コードの順序に一致するようにします。一致しない場合は、順序を変更してテストを再コンパイルしなければなりません。
- 可能な場合は、Quartus II ソフトウェア、ByteBlaster™ II ダウンロード・ケーブル、および **.svf** ファイルの生成に使用した **.pof** ファイルを使用して、デバイスが正しくプログラムされていることを確認します。この処置は、製造時に実用的ではありませんが、テスト開発およびデバッグ中に役立ちます。
- 個々のデバイスを分離する必要がある場合、チェイン内のターゲットとするアルテラ・デバイスごとに個別の **.svf** ファイルを生成できます（4 ページの「ステップ 2: .svf ファイルの作成」を参照）。検証エラーが発生し、チェイン内の複数のアルテラ・デバイスがプログラムされる場合、このプロセスを使用してください。
- デバイスのプログラミングが失敗する場合は、ウンダリ・スキャン・チェインの定義を調べます。命令レジスタのビット数がチェイン内の各デバイスに対して正しく指定されていることを確認します。チェイン内のいずれかのデバイスに対して不正なビット数が定義されている場合、プログラミング・テストは失敗します。

テストが正しく動作すると、ボードは製造プログラミングが可能な状態になります。アルテラは、**.pcf** ファイルとオブジェクト・コードをバックアップのために保存しておくことを推奨しています。圧縮プログラムを使用して、保存するバイナリおよびファイルのサイズを最小にします。

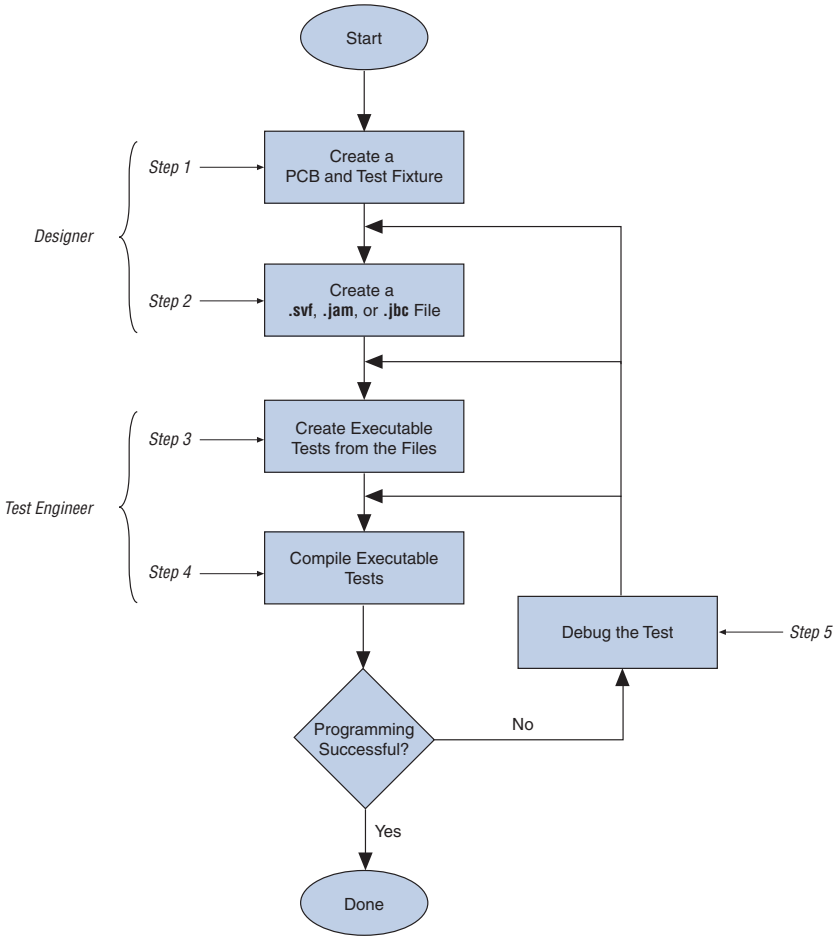
PLD ISP ソフトウェアを使用した Agilent 3070 テスタの開発フロー

PLD ISP ソフトウェアを使用した Agilent 3070 テスタによるデバイスのプログラミングは、2 ページの図 1 のステップとは多少異なります。

図 3 に、Agilent オプションの PLD ISP ソフトウェアと Agilent 3070 テスタを使用した開発フローを示します。

次のセクションでは、図 3 に示す各ステップについて説明します。

図 3. Agilent の PLD ISP ソフトウェアを使用したイ ISP に対する Agilent 3070 開発フロー



Agilent PLD ISP ソフトウェアを使用すると、デバイス・プログラミングに対する **svf2pcf** フローと比較して、以下の利点が得られます。

- テスタは、**.svf**、Jam STAPL（**.jam**）、または **.jbc** ファイル・フォーマットを直接使用した（つまり、**.pcf** や VCL に変換しない）デバイスのプログラミングをサポートできます。
- デバイスをプログラムする Agilent 3070 デジタル・テストは、1 つのファイルになります。
- デバイス・プログラミングが 1 つのテストとして実行されるため、TCK ラインと TMS ラインにプルアップ抵抗とプルダウン抵抗は必要ありません。
- デジタル・テストのソース・ファイル、およびコンパイル済みのオブジェクト・ファイルのサイズが **svf2pcf** ソリューションの場合よりも、はるかに小さくなります。
- 1 つのデジタル・テスト・ファイルのみ実行されるため、大規模な CPLD およびコンフィギュレーション・デバイスに対する実行時間が高速化されます。

Jam Byte-Code Player

Agilent の PLD ISP ソフトウェアを使用すると、Jam Byte-Code Player はテスタの Control XTP カードに実装されます。これにより、Quartus II ソフトウェアで生成された **.jbc** ファイルを使用してデバイスをプログラムすることができます。また、テスタはこれらのプログラミング用ファイルをコンパイルする JBC コンパイラを備えているため、**.jam** ファイルや **.svf** ファイルにも対応します。Jam Byte-Code Player は、Control XTP カード上のマイクロコントローラを介して実行され、それによって、ベクタのシーケンスを実行するのではなく、アルゴリズム的にベクタを適用することが可能になります。

プログラミング時間

Agilent 3070 テスタにおけるプログラミング時間は、極めて一貫しています。唯一の変数は TCK 周波数で、これはプログラミング時間に影響を及ぼします。プログラミング時間は、TCK クロック・レートの関数であるためです。クロックを高速にすると、データをデバイスにシフトする時間が短くなります。MAX II および MAX V デバイスは、最大 18 MHz の TCK クロック・レートをサポートしています。

ガイドライン

Agilent 3070 テスタをプログラミングに使用するときには、以下のガイドラインに従ってください。

- ピン・ライブラリを使用して、スタンドアロンのバウンダリ・スキャン・チェイン内のターゲット・デバイスを記述する場合は注意が必要です。アルテラは、ISP デバイスのすべての I/O ピンを双方向として記述することは推奨していません。この手法では多数のハイブリッド・カード・チャネルが使用されるため、テストの開発時に冶具のオーバーフロー・エラーが発生する原因となる可能性があります。

- テスト・ライブラリには **.pcf** ベクタを含めないでください。セットアップ専用ノード・ライブラリを使用してください。**.pcf** ベクタを含むテスト・ライブラリを作成すると、おおきなライブラリ・オブジェクト・ファイルが作成され、テスト開発時間が大幅に長くなります。このような遅延が発生するのは、統合プログラム・ジェネレータがライブラリ・オブジェクトのベクタ・セット全体を調べ、競合回避のためにベクタをコメント・アウトする必要があるか判断するからです。ライブラリ・オブジェクト・コンパイルは、実行可能コンパイルとは異なります。さらに、ライブラリ・オブジェクト・ファイルが大きいために、統合プログラム・ジェネレータが失敗することがあります。
- 時間とディスク・スペースを節約するには、プログラミング動作での検証を含む **.svf** ファイルを生成します。このプロセスでは、検証ベクタは 1 つのステップに統合されるため、テスト開発プロセスでの作業量が減少します。この統合化された検証は、プログラミング・エラーを性格にキャプチャするため、テスト・シーケンスに付加的なスタンドアロン検証を追加する必要はありません。
- 本書では、テストを生成してプログラミング用のデバイスにベクタを適用する方法を説明していますが、デバイスの機能をテストするにはバウンダリ・スキャン記述言語（BSDL）ファイルが必要です。バウンダリ・スキャン・テストまたは機能テストを実行する場合は、ピン・コンフィギュレーション情報（どのピンが入力ピン、出力ピン、双方向ピンであるかなど）を含むターゲット・デバイスのプログラム済み状態に対応する BSDL ファイルを生成します。テストの生成には、Agilent 3070 バウンダリ・スキャン・ソフトウェアを使用します。

バウンダリ・スキャン・テストのサポートについて詳しくは、*「MAX II デバイス・ハンドブック」*の *「IEEE 1149.1 (JTAG) Boundary-Scan Testing for MAX II Devices」*の章または *「MAX V デバイス・ハンドブック」*の *「JTAG Boundary-Scan Testing for MAX V Devices」*の章を参照してください。

改訂履歴

表 1 に、このアプリケーション・ノートの改訂履歴を示します。

表 1. 改訂履歴

日付	バージョン	変更内容
2010 年 12 月	1.0	初版