

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。

MI151016-1.4

### はじめに

イン・システム・プログラミングは、プログラマブル・ロジック・デバイス (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、Agilent 3070 システムの既存システム・フローにイン・システム・プログラマビリティ (ISP) を統合するソフトウェアおよびデバイスのサポートを提供しています。この章では、Agilent 3070 テスト・システムを使用して、アルテラの MAX® II 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。

この章は、以下の項で構成されています。

- 15-1 ページの「Agilent 3070 用の新しい PLD 製品」
- 15-1 ページの「デバイス・サポート」
- 15-2 ページの「PLD ISP ソフトウェアを使用しない Agilent 3070 開発フロー」
- 15-11 ページの「PLD ISP ソフトウェアを使用した Agilent 3070 開発フロー」
- 15-13 ページの「プログラミング時間」
- 15-14 ページの「ガイドライン」

### Agilent 3070 用の新しい PLD 製品

Agilent 3070 テスタのメーカーである Agilent Technologies は、PLD プログラミング問題の解決をサポートする新しい PLD ISP ソフトウェアを発表しました。この新製品を使用することで得られる利点については、この章の後半で説明します。

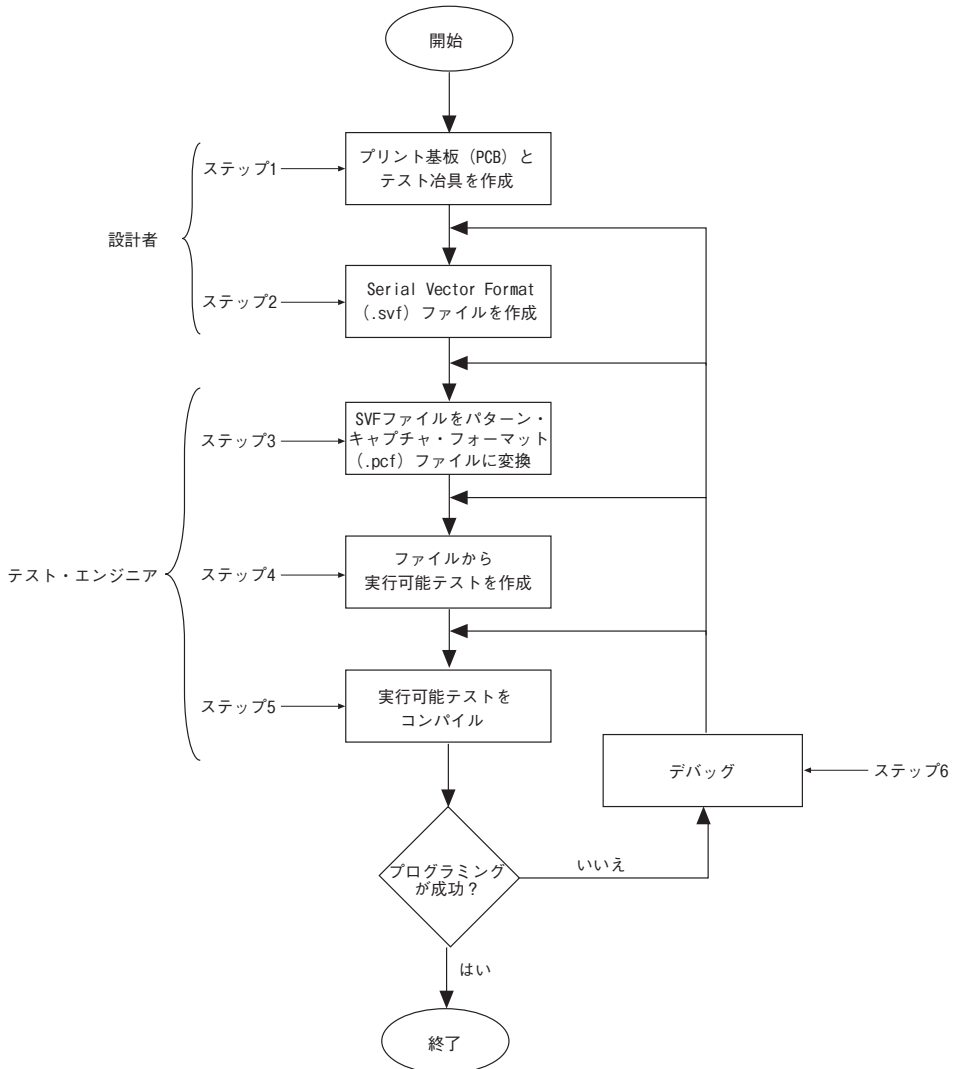
### デバイス・ サポート

Agilent 3070 テスタを使用して、MAX II デバイスを他のファミリのデバイスと共にプログラムする場合は、このテスタでチェーン内のすべてのデバイスをプログラムできるようにする必要があります。

## PLD ISP ソフトウェア を使用しない Agilent 3070 開発フロー

Agilent の PLD ISP ソフトウェアを使用しないで、Agilent 3070 テスタ (Serial Vector Format (.svf) ファイルを使用) でデバイスをプログラムするには、以下のステップに従う必要があります。図 15-1 を参照してください。

図 15-1. SVF ファイルを使用したイン・システム・プログラミングに対する 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

イン・システム・プログラミングを成功させるための最初のステップは、テスト開発を始める前に、ボードを適切にレイアウトしてからテスト治具を作成することです。

### PCB 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 シグナル・トレースは、クロック・ツリーと同様に慎重に扱う必要があります。TCK は、デバイスの JTAG (Joint Test Action Group) 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ことが不可欠です。信号がデバイス・データシートで規定される  $t_R$  および  $t_F$  パラメータに適合するようにしてください。
- TCK にプルダウン抵抗を追加します。TCK 信号は、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。パターン・キャプチャ・フォーマット (PCF) ダウンロードについて詳しくは、「[ステップ 2: Serial Vector Format ファイルの作成](#)」を参照してください。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。
- 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・ポイントを設けます。動作中には、PCB 動作が乱れないように、十分なアクセス・ポイントが必要です。アクセス・ポイントが足りないと、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。
- 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、システム・ノイズを低減するために、オンボード・オシレータを電氣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。



出力ピンはプログラミング中にはトライ・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、「MAX II デバイス・ハンドブック」の「MAX II デバイスのイン・システム・プログラマビリティ・ガイドライン」の章を参照してください。

## 治具の作成

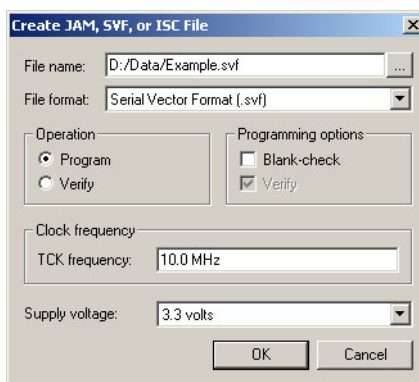
イン・システム・プログラミングを成功させるには、テスト治具とターゲット・ボードの間にクリーンなインターフェースを提供することが不可欠です。クリーンなインターフェースを提供するには、テスト治具内で短いワイヤを使用して、TCK の接続性を向上させます。ワイヤを長くすると、システム内部に誘導ノイズが誘発され、プログラミングが中断することがあります。TCK を接続するワイヤは 1 インチ未満にしてください。テスト治具のレイアウトと作成を管理するには、Agilent Fixture Consultant を使用します (Agilent Board Test Family Manual を参照)。

## ステップ 2: Serial Vector Format 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 ファイルを生成します。複数の MAX II CPLD ファミリ・デバイス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 15-2 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 15-2. 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、チェー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。実際の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プログラミング・オプション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・チェーン・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ると、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必要がある場合、チェーン内のアルテラ・デバイスごとに、個別に SVF ファイルを生成できます。チェーン内の 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、残りのデバイスは <none> に設定したままにします。これは、Programmer で **Add Device** を選択して、実行できます。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 ファイルの PCF ファイルへの変換

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 ファイルに変換する必要があります。**svf2pcf** ユーティリティは、1 つのデバイス・チェーンに対して複数の 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 ([www.altera.co.jp](http://www.altera.co.jp)) の Agilent ISP Support 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、以下のステップを実行する必要があります。

1. 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を作成する。
2. Test Consultant を実行する。
3. デジタル・テストを作成する。
4. テスト用のワイヤリスト情報を作成する。
5. テスト・プランを修正する。

### 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バウンダリ・スキャン・チェインの一部でない (分離されている) 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して、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らないことに注意してください。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**) に分割された順番に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。

アルテラ・デバイスを含むバウンダリ・スキャン・チェーンが定義され、PCF ベクタが JTAG インタフェースに適用されている場合は、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。

多数のベクタが検証されるため、各ベクタを調べて失敗の原因を特定することは実用的ではありません。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。デザイン・エンジニアが、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ュホー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。

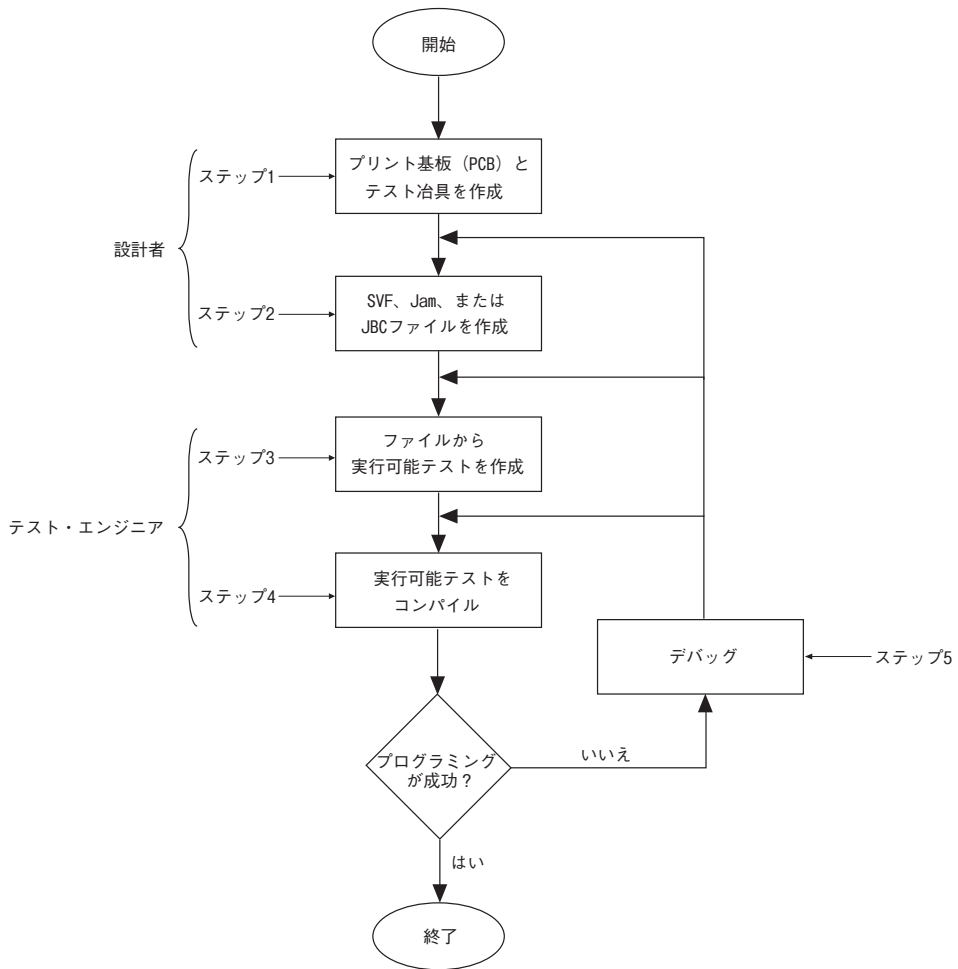
- テストの実行順序が正しいことを確認します。順序がバラバラでテストを実行すると、プログラミング情報が不正になります。また、同じテストを連続2回実行すると、ターゲット・デバイスが順不同になり、正しいプログラミング情報を受け入れません。
- 実際のベクタが入力ピン (TCK、TMS、および TDI) の予想値と一致するようにします。予想値が一致しない場合は、テストを再コンパイルする必要があります。
- テストにおける pcf order ステートメントが、15-5 ページの「[ステップ 2: Serial Vector Format ファイルの作成](#)」で生成された PCF コードの順序に一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 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、チェー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。SVF ファイルを生成するプロセスは、15-5 ページの「[ステップ 2: Serial Vector Format ファイルの作成](#)」で説明されています。検証エラーが発生し、チェー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は、このプロセスが役立ちます。
- 上記のいずれの手順を実行しても問題が解決しない場合は、バウンダリ・スキャン・チェーンの定義を調べます。命令レジスタのビット数がチェー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。チェー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。

テストがスムーズに動作すると、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 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、[図 15-1](#) のステップとは多少異なります。[図 15-3](#) 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。

図 15-3. Agilent の PLD ISP ソフトウェアを使用したイン・システム・プログラミングに対する 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、以下の利点を得られます。

- テスタは、SVF、Jam STAPL、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・プログラミングは全体として1つのテストとして実行されるため、テストの治具で 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。

Agilent の PLD ISP ソフトウェアを使用すると、Jam Byte-Code Player はテストの Control XTP カードに実装されます。これによって、ユーザは Quartus II から直接作成された 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可能になります。また、テスト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、それによってユーザは、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。Jam Byte-Code Player は、デバイスのプログラミングおよび消去パルス幅レジスタを読み出し、これらの値をプログラミングおよび消去アルゴリズムで使用します。

## プログラミング 時間

Agilent 3070 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。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。プログラミング時間は、TCK クロック・レートの関数です。MAX II 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・チェーン内のターゲット・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・チャンネルが使用されるため、テストの開発時に治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 (IPG) 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、IPG 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の実行が必要な場合は、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・テストに対するアルテラ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「MAX II デバイスの IEEE 1149.1 (JTAG) バウンダリ・スキャン・テスト」の章を参照してください。

## まとめ

アルテラは、Agilent 3070 テスト・システムを使用して、すべての MAX II 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。すべての MAX II デバイスは、その他の ISP 対応デバイスと組み合わせてプログラムできます。ソフトウェアおよびデバイス・サポートにより、Agilent 3070 ユーザはコスト削減と生産性の向上を実現できます。

## 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。

- 「MAX II デバイス・ハンドブック」の「MAX II デバイスの IEEE 1149.1 (JTAG) バウンダリ・スキャン・テスト」の章
- 「MAX II デバイス・ハンドブック」の「MAX II デバイスのイン・システム・プログラマビリティ・ガイドライン」の章

## 改訂履歴

表 15-1 に、本資料の改訂履歴を示します。

表 15-1. 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 12 月 v1.4	「参考資料」の項を追加。	—
2006 年 12 月 v1.3	改訂履歴を追加。	—
2005 年 6 月 v1.2	「プログラミング時間」の項のテキスト修正（25 MHz を 18 MHz に変更）。	—
2005 年 1 月 v1.1	16 章から変更。内容の変更はなし。	—

