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 It may contain legacy content and trademarks which may be outdated.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

15. FPGA メモリおよび定数のインシステム・アップデート

QII53012-7.2.0

はじめに

FPGA のデザインはますます集積度が高くなり、複雑になってきています。設計者および検証エンジニアが、問題を素早く識別、テスト、および解決するため、デバイス上にプログラムされたデザインへアクセスする必要性はますます増えています。Quartus® II ソフトウェアのメモリおよび定数のインシステム・アップデート機能は、Joint Test Action Group（JTAG）インタフェースを介して、インシステムでの FPGA メモリおよび定数への読み出しおよび書き込みアクセスを提供し、エンド・システムで FPGA が動作している間に、FPGA 内のメモリ内容への変更をより簡単にテストできるようにします。

この章では、Quartus II In-System Memory Content Editor を FPGA デザインおよび検証フローの一部として使用する方法を説明します。

この章は、以下の項で構成されています。

概要

プログラムされたデバイスのメモリおよび定数の読み出しおよびアップデート機能により、ユーザ・デザインを十分に把握し、コントロールすることができます。Quartus II In-System Memory Content Editor を使用して、デバイスのメモリおよび定数にアクセスできます。この機能を SignalTap® II エンベデッド・ロジック・アナライザと共に使用すると、デザインをハードウェア・ラボでデバッグするのに必要な可視性が得られます。

SignalTap II エンベデッド・ロジック・アナライザについて詳しくは、「Quartus II ハンドブック Volume 3」の「SignalTap II エンベデッド・ロジック・アナライザを使用したデザインのデバッグ」の章を参照してください。

メモリおよび定数からデータを読み出す機能を備えているため、問題の発生源を素早く特定することができます。また、書き込み機能も備えており、期待される正しいデータを書き込むことによって、機能的な問題をバイパスすることができます。例えば、メモリのパリティ・ビットが正しくない場合は、In-System Content Editor を使用して RAM に正しい

パリティ・ビット値を書き込むことができ、それによってシステムは動作の継続が可能です。また、RAM に不正なパリティ・ビット値を意図的に書き込んで、デザインのエラー処理機能をチェックできます。

デバイスのメガファンクションのサポート

以下の表に、現在 Quartus II ソフトウェアでサポートされるデバイスとメモリおよび定数のタイプを示します。表 15–1 に、MegaWizard® Plug-In Manager および In-System Memory Content Editor によってサポートされるメモリのタイプを示します。

表 15–1. MegaWizard Plug-In Manager のサポート

インストールされたプラグ・イン・カテゴリ	メガファンクション名
ゲート	LPM_CONSTANT
メモリ・コンパイラ	RAM: 1-PORT, ROM: 1-PORT
ストレージ	ALTSYNCRAM, LPM_RAM_DQ, LPM_ROM

表 15–2 に、Stratix® シリーズ、Arria™ GX、Cyclone® シリーズ、APEX™ II、APEX 20K、および Mercury™ デバイス・ファミリに対するメモリおよび定数のインシステム・アップデートのサポートを示します。

表 15–2. サポートされるメガファンクション

メガファンクション	Arria GX / Stratix シリーズ			Cyclone シリーズ	APEX II	APEX 20K	Mercury
	M512 ブロック	M4K ブロック	MegaRAM ブロック				
LPM_CONSTANT	読み出し / 書き込み	読み出し / 書き込み	読み出し / 書き込み	読み出し / 書き込み	読み出し / 書き込み	読み出し / 書き込み	読み出し / 書き込み
LPM_ROM	書き込み	読み出し / 書き込み	N/A	読み出し / 書き込み	読み出し / 書き込み	書き込み	読み出し / 書き込み
LPM_RAM_DQ	N/A	読み出し / 書き込み	読み出し / 書き込み	読み出し / 書き込み	読み出し / 書き込み	N/A (1)	読み出し / 書き込み
ALTSYNCRAM (ROM)	書き込み	読み出し / 書き込み	N/A	読み出し / 書き込み	N/A	N/A	N/A
ALTSYNCRAM（シングル・ポート RAM モード）	N/A	読み出し / 書き込み	読み出し / 書き込み	読み出し / 書き込み	N/A	N/A	N/A

表 15–2 の注：

(1) このシングル・ポート RAM には、書き込み専用モードのみ適用できます。読み出し専用モードでは、LPM_RAM_DQ の代わりに LPM_ROM を使用します。

デザインでのメモリ定数のインシステム・アップデートの使用

メモリおよび定数のインシステム・アップデート機能を使用するには、以下のステップが必要です。

1. アクセスするメモリと定数を確認します。
2. メモリと定数を実行時に修正可能になるように設定します。
3. フル・コンパイルを実行します。
4. デバイスをプログラムします。
5. In-System Memory Content Editor を起動します。

インシステムで修正可能なメモリ定数の作成

メモリまたは定数がシステム動作時に修正可能と設定した場合、Quartus II ソフトウェアはデフォルトの実装を変更します。シングル・ポート RAM はデュアル・ポート RAM に変換され、定数はルックアップ・テーブル（LUT）の代わりにレジスタに実装されます。これらの変更により、デザインの機能を変えることなく実行時の修正が可能です。実行時に修正可能なメガファンクションのリストについては、表 15–1 を参照してください。

メモリまたは定数を修正可能にするには、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。
2. 新しいメガファンクションを作成する場合は、**Create a new custom megafunction variation** を選択します。既存のメガファンクションがある場合は、**Edit an existing custom megafunction variation** を選択します。
3. デザインに要求される特性に基づいてメガファンクションに対して必要な変更を加え、**Allow In-System Memory Content Editor to capture and update content independently of the system clock** をオンにし、**Instance ID** テキスト・ボックスに値を入力します。これらのパラメータは、インシステム・アップデートをサポートするメガファンクションに対するウィザードの最終ページにあります。

   
   インスタンス ID は、メガファンクションを他のインシステム・メモリおよび定数から区別するのに使用される 4 文字の文字列です。

4. **Finish** をクリックします。
5. Processing メニューで **Start Compilation** をクリックします。

メモリまたは定数メガファンクションを、VHDL または Verilog HDL のポートとパラメータを直接使用してインスタンス化する場合は、以下に示すとおり lpm_hint パラメータを追加または修正します。

VHDL コードでは、以下を追加します。

```
lpm_hint => "ENABLE_RUNTIME_MOD = YES,
  INSTANCE_NAME = <インスタンス名>";
```

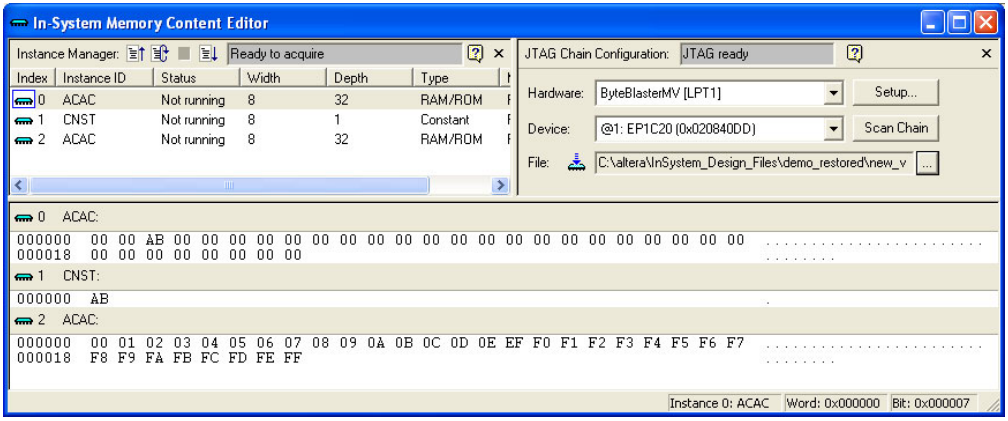
Verilog HDL コードでは、以下を追加します。

```
defparam <メガファンクション・インスタンス名>.lpm_hint =
  "ENABLE_RUNTIME_MOD = YES,
  INSTANCE_NAME = <インスタンス名>";
```

In-System Memory Content Editor の実行

In-System Memory Content Editor は、Instance Manager、JTAG Chain Configuration、および Hex Editor に分けられます（図 15–1）。

図 15–1. In-System Memory Content Editor



Instance Manager は、FPGA デバイス内の実行時に修正可能なすべてのメモリおよび定数を表示します。JTAG チェイン・コンフィギュレーション・セクションでは、FPGA をプログラムしたり、チェイン内でアップデートするアルテラ・デバイスを選択することができます。

In-System Memory Content Editor を使用するときに、プロジェクトを開く必要はありません。In-System Memory Content Editor は、JTAG チェインをスキャンし、JTAG チェイン・コンフィギュレーション・セクションで選択した特定のデバイスにクエリを送ることによって、実行時にコンフィギュレーション可能なメモリおよび定数のすべてのインスタンスを取得します。

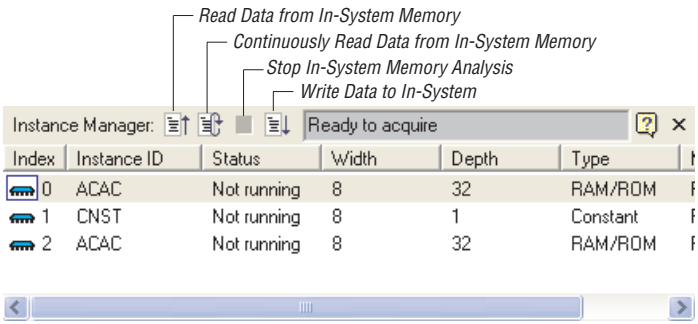
各 In-System Memory Content Editor は、1 つのデバイスのインシステム・メモリおよび定数にアクセスできます。JTAG チェインで複数のデバイスがインシステムのコンフィギュレーション可能なメモリまたは定数を持っている場合、Quartus II ソフトウェア内で複数の In-System Memory Content Editor を起動して、各デバイスのメモリおよび定数にアクセスできます。

Instance Manager

JTAG チェインをスキャンして、デザイン内の実行時に修正可能なすべてのメモリおよび定数のリストで、Instance Manager をアップデートします。Instance Manager は、リスト内の各エレメントのインデックス、インスタンス、ステータス、幅、深さ、タイプ、およびモードを表示します。

図 15–2 に示すとおり、Instance Manager を使用して、インシステム・メモリを読み出したり書き込むことができます。

図 15–2. Instance Manager のコントロール



Instance Manager には、以下のボタンが用意されています。

- **Read data from In-System Memory**— システム・クロックに関係なくデバイスからデータを読み出し、それを Hex Editor に表示します。

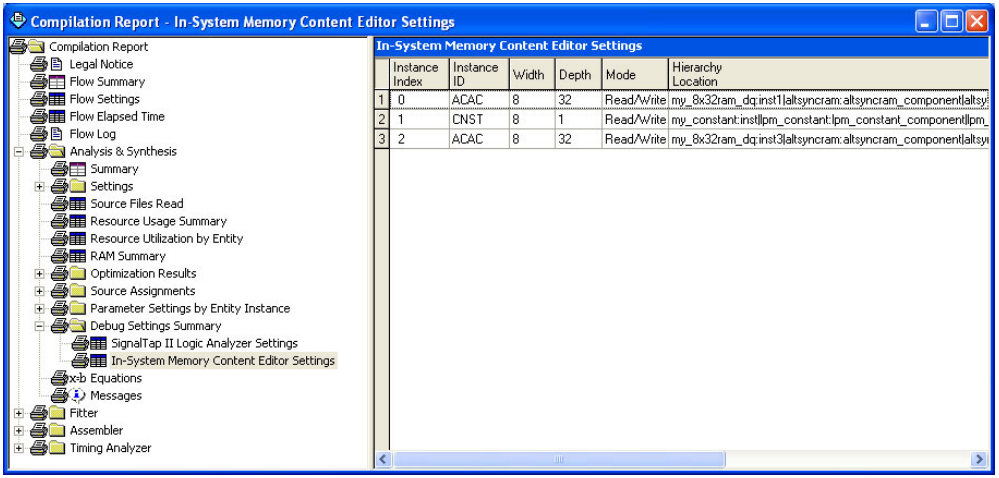
- **Continuously Read Data from In-System Memory**— デバイスから非同期でデータを継続的に読み出し、それを Hex Editor に表示します。
- **Stop In-System Memory Analysis**— 現在の読み出しまたは書き込み動作を停止します。
- **Write Data to In-System Memory**—Hex Editor にあるデータを非同期でデバイスに書き込みます。

Instance Manager で使用可能なボタンに加えて、Instance Manager または Hex Editor の Processing メニューまたは右ボタンのポップアップ・メニューからコマンドを選択することによって、データを読み出したり書き込んだりすることができます。

各インスタンスのステータスは、Instance Manager の各エントリの横にも表示されます。ステータスはインスタンスが **Not running**、**Offloading data**、または **Updating Data** のいずれであるかを示します。ヘルス・モニタはエディタのステータスに関する有用な情報を提供します。

Quartus II ソフトウェアは、各インシステム・メモリおよび定数に異なるインデックス番号を割り当てて、同じメモリまたは定数機能の複数のインスタンスを識別します。コンパイル・レポートの **In-System Memory Content Editor Setting** セクションを表示して、インデックスと対応するインスタンス ID を一致させます（図 15–3）。

図 15–3. コンパイル・レポートの In-System Memory Content Editor Setting セクション



Hex Editor に表示されているデータの編集

エディタに値を直接入力するか、あるいはメモリ・ファイルをインポートすることによって、インシステム・メモリおよび定数から読み出し、Hex Editor に表示されているデータを編集することができます。

Hex Editor に表示されているデータを修正するには、エディタのある位置をクリックして入力するか、新しいデータをペーストします。新しいデータは青で表示され、修正されたデータが FPGA に書き込まれていないことを示しています。Edit メニューで、Value を選択して、**Fill with 0's**、**Fill with 1's**、**Fill with Random Values**、または **Custom Fills** をクリックし、データ・ブロックを選択することによってデータ・ブロックをアップデートします。

メモリ・ファイルのインポート、エクスポート

メモリ・ファイルのインポートおよびエクスポートによって、インシステム・メモリを新しいメモリ・イメージで素早くアップデートし、将来の使用および解析のためにデータを記録します。

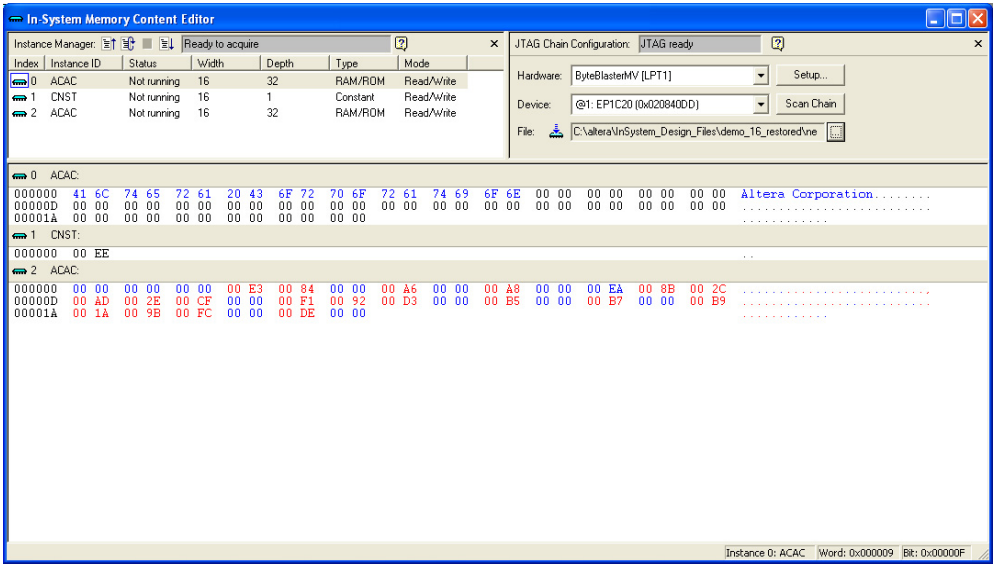
Edit メニューの **Import Data from File** をクリックしてメモリ・ファイルをインポートするか、インスタンス・マネージャからインシステム・メモリまたは定数を選択します。16 進（インテル・フォーマット）ファイル（**.hex**）またはメモリ初期化ファイル（**.mif**）フォーマットのメモリ・ファイルのみインポートできます。

Edit メニューの **Export Data to File** をクリックして、Hex Editor に表示されたデータをメモリ・ファイルにエクスポートするか、インスタンス・マネージャからインシステム・メモリまたは定数を選択します。データを **.hex**、**.mif**、Verilog Value Change Dump ファイル（**.vcd**）、または RAM 初期化ファイル（**.mif**）フォーマットにエクスポートすることができます。

Hex Editor でのメモリ定数の表示

インシステム・メモリまたは定数の各インスタンスについて、Hex Editor はデータを 16 進表現および ASCII 文字（ワード・サイズが 8 ビットの倍数の場合）で表示します。16 進数の配置はメモリの寸法によって決まります。例えば、ワード幅が 16 ビットの場合、Hex Editor はデータをバイト列を含むワード列で表示します。（図 15–4）

図 15–4. Hex Editor を使用した 16 ビット・メモリ・ワードの編集



プリント不能な ASCII 文字はピリオド（.）で表されます。読み出しおよび書き込みを実行すると、データの色が変化します。黒で表示されるデータは、Hex Editor のデータがデバイスから読み出されたデータと同じであることを示しています。Hex Editor のデータの色が赤に変わった場合は、それまで Hex Editor に表示されていたデータとデバイスから読み出されたデータが異なります。

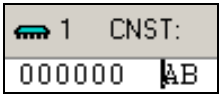
データを解析すると、カーソルとステータス・バーを使用して、メモリ内の正確な位置を素早く特定することができます。ステータス・バーは In-System Memory Content Editor の最下部にあり、選択したインスタンス名、ワード位置、およびビット・オフセットを表示します（図 15–5）。

図 15–5. In-System Memory Content Editor のステータス・バー

ビット・オフセットはワード内のカーソルのビット位置です。以下の例では、1 ワードは 8 ビット幅に設定されています。

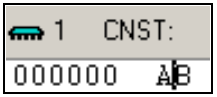
カーソルが図 15–6 に示す位置にある場合、ワード位置は 0×0000、ビット位置は 0×0007 です。

図 15–6. Hex Editor のカーソルはビット 0×0007 に置かれます。



カーソルが図 15–7 に示す位置にある場合、ワード位置は 0×0000 のままですが、ビット位置は 0×0003 です。

図 15–7. Hex Editor のカーソルはビット 0×0003 に置かれます。



スクリプトのサポート

In-System Memory Content Editor は、コマンド・プロンプトに入力された Tcl スクリプトまたは Tcl コマンドによるメモリ内容の読み出しと書き込みをサポートします。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 r
```

「Quartus II Scripting Reference 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。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。

In-System Memory Content Editor で一般的に使用されるコマンドは、次のとおりです。

- メモリからの読み出し :
  ```
  read_content_from_memory
  [-content_in_hex]
  -instance_index <インスタンス・インデックス>
  -start_address <開始アドレス>
  -word_count <ワード・カウント>
  ```
- メモリへの書き込み :
  ```
  write_content_to_memory
  ```
- ファイルへのメモリ内容の保存 :
  ```
  save_content_from_memory_to_file
  ```
- ファイルへのメモリ内容の保存 :
  ```
  update_content_to_memory_from_file
  ```

コマンド・オプションの説明とスクリプト例については、Tcl API Help ブラウザおよび「Quartus II Scripting Reference Manual」を参照してください。

In-System Memory Content Editor を使用したデバイスのプログラミング

デザインを変更した場合、In-System Memory Content Editor からデバイスをプログラムすることができます。デバイスをプログラムするには、以下のステップに従います。

1. Tools メニューの **In-System Memory Content Editor** をクリックします。
2. In-System Memory Content Editor の **JTAG Chain Configuration** パネルで、修正可能なメモリと定数を含む SRAM オブジェクト・ファイル（**.sof**）を選択します。
3. **Scan Chain** をクリックします。
4. **Device** リストで、プログラムするデバイスを選択します。
5. **Program Device** をクリックします。

例 : SignalTap II Embedded Logic Analyzer での In-System Memory Content Editor の使用

以下のシナリオでは、SignalTap II エンベデッド・ロジック・アナライザで、メモリおよび定数のインシステム・アップデート機能を使用して、インシステムでデザインを効率的にデバッグする方法を説明しています。In-System Memory Content Editor と SignalTap II エンベデッド・ロジック・アナライザは両方とも JTAG 通信インタフェースを使用しますが、それらは同時に使用できます。

FPGA デザインの完了後、FIR フィルタ・デザインの特性が期待通りでないことが判明しました。

1. 問題の原因を特定するには、すべての FIR フィルタ係数をインシステムで修正可能になるように変更し、SignalTap II エンベデッド・ロジック・アナライザをインスタンス化します。

2. SignalTap II エンベデッド・ロジック・アナライザを使用し、内部デザイン・ノードでタップおよびトリガすると、FIR フィルタが期待されるカットオフ周波数の外側で動作していることが分かります。

3. **In-System Memory Content Editor** を使用して、FIR フィルタ係数が適正かどうかをチェックします。各係数を読み出したときに、係数の 1 つが誤っていることに気づきました。

4. 係数はインシステムで修正可能なので、**In-System Memory Content Editor** を使用して、係数を正しいデータでアップデートします。

このシナリオでは、In-System Memory Content Editor と SignalTap II エンベデッド・ロジック・アナライザの両方を使用して、問題の原因を素早く特定することができます。また、デザインのソース・ファイルを修正する前に、係数値を変更することによって、デバイスの機能性を検証することも可能です。

この例の延長として、係数を In-System Memory Content Editor で修正して、FIR フィルタの特性（フィルタの減衰、遷移帯域幅、カットオフ周波数、ウィンドウ関数など）を変更することがあります。

まとめ

インシステムでのメモリおよび定数のアップデート機能は、ハードウェア・ラボで効率的なデバッグを行うためにデバイスへのアクセスを提供します。SignalTap II エンベデッド・ロジック・アナライザでメモリおよび定数のインシステム・メモリ・アップデートを使用して、アルテラ FPGA の可視性を最大限に高めます。デバイスの内部ロジックへの可視性およびアクセスを向上させることによって、デザインまたはその実装の問題をより迅速に特定して、解決することができます。

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 2」の「Command-Line Scripting」の章
- 「Quartus II ハンドブック Volume 3」の「Design Debugging Using the SignalTap II Embedded Logic Analyzer」の章
- 「Quartus II Scripting Reference Manual」
- 「Quartus II ハンドブック Volume 2」の「Tcl Scripting」の章

改訂履歴

表 15–3 に、本資料の改訂履歴を示します。

表 15–3. 改訂履歴

日付 & バージョン	変更内容	概要
2007 年 10 月 v7.2.0	15–12 ページの「参考資料」を再編成。	—
2007 年 5 月 v7.1.0	• 15–9 ページに「スクリプト・サポート」の項を追加。 • 15–12 ページに「参考資料」の項を追加。	Quartus II バージョン 7.1 のために更新。
2007 年 3 月 v7.0.0	15–2 ページにリストされる Cyclone III デバイスのサポートを追加。	—
2006 年 11 月 v6.1.0	• 改訂履歴を追加。 • 表 15–2 を更新。	Stratix III のサポートの情報を追加。
2006 年 5 月 v6.0.0	Quartus II ソフトウェア・バージョン 6.0.0 のためのマイナー・アップデート。	—
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。 • バージョン 5.0 の 12 章を 13 章に変更。	—
2005年5月v5.0.0	• バージョン 4.2 の Vol 3 のセクション V を 12 章に変更。	—
2004年12月v1.2	• 11 章を 12 章に変更。 • 表を更新。 • `lpm_hint` パラメータの Verilog コードを修正。 • “Making Changes” セグメントを、Hex Editor の「Editing Data Displayed in the Hex Editor」と「Importing and Exporting Memory Files」セグメントに再編成。Edit value メニューを追加。 • 以下の例を追加。SignalTap II エンベデッド・ロジック・アナライザでの In-System Memory Content Editor の使用。	—
2004 年 8 月 v1.1	誤字・脱字のマイナー修正。	—
2004 年 6 月 v1.0	初版。	—