

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。

H51025-1.2

### はじめに

Quartus®II ソフトウェアには、コマンドライン実行コマンドのセットが含まれており、その多くがインタラクティブ Tcl シェルをサポートしています。Tcl シェルを使用し、Quartus®II ウィンドウ・ベース GUI を使用しないで、FPGA または HardCopy® デザイン操作を実行できます。

この章では、インタラクティブ Tcl シェルを使用したスクリプト・ベースの HardCopy II デザインのための Tcl 操作の概要を説明します。この章では、以下の内容について説明します。

- Quartus II ソフトウェアでの Tcl スクリプティング機能の概要
- HardCopy II デザイン・フロー
- ロケーションおよびタイミング制約の適用
- 合成、HardCopy II デザインの配置配線、および Stratix®II プロトタイプ
- デザイン検証および解析

### Quartus II ソフトウェア での Tcl サポート

Quartus II ソフトウェアは、以下をはじめとする Tcl コマンドおよびスクリプトを実行するためのさまざまな方法を提供します。

- Tcl Console ウィンドウ
- Tcl Scripts ダイアログ・ボックス
- コマンドライン処理
- インタラクティブ Tcl シェル

Tcl Console ウィンドウおよび **Tcl Scripts** ダイアログ・ボックスは、両方とも Quartus II GUI 内で動作し、ここでは説明していません。代わりに、この章では Quartus II コマンドライン実行コマンドで使用可能なインタラクティブ Tcl シェルに的を絞っています。



コマンドライン処理およびバッチファイル、メイクファイル、およびスクリプトでの Quartus II コマンドライン実行コマンドの使用については詳しくは、「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。



Quartus II Tcl 実装について詳しくは、「Quartus II ハンドブック」の「Tcl Reference Manual」および「Tcl Scripting」の章を参照してください。

## インタラクティブ Tcl シェル

多数の Quartus II 実行コマンドをユーザ・インタフェースとして、インタラクティブ Tcl シェルで実行できます。これらの実行コマンドを表 6-1 に示します。インタラクティブ Tcl シェルは Tcl バージョン 8.4 をサポートしています。

コマンド名	説明
quartus_sh	基本的な Tcl インタプリタ・シェル。アサインメント指定、コンパイル操作、およびネイティブ・オペレーティング・システム・コマンドのサポート。詳しくは、「Quartus II Scripting Reference Manual」の「Command-Line Executables」の項にある quartus_sh を参照してください。
quartus_sta	Quartus II TimeQuest タイミング・アナライザ・エンジンは、デザインおよびタイミング解析 Tcl コマンドのタイミング・グラフの構築をサポートします。詳しくは、「Quartus II Scripting Reference Manual」の「Command-Line Executables」の項にある quartus_sta を参照してください。
quartus_tan	Quartus II クラシック・タイミング・アナライザ・エンジンは、デザインおよびタイミング解析 Tcl コマンドのタイミング・グラフの構築をサポートします。詳しくは、「Quartus II Scripting Reference Manual」の「Command-Line Executables」の項にある quartus_tan を参照してください。
quartus_cdb	Quartus II データベース・インタフェース実行コマンド。HardCopy II デザインに対する LogicLock、バック・アノテーション、FPGA と HardCopy の比較などのデザイン・データベースに関連する操作をサポート。詳しくは、「Quartus II Scripting Reference Manual」の「Command-Line Executables」の項にある quartus_cdb を参照してください。
quartus_sim	Quartus II シミュレータ詳しくは、「Quartus II Scripting Reference Manual」の「Command-Line Executables」の項にある quartus_sim を参照してください。

コマンドライン実行コマンドのインタラクティブ Tcl シェルは、コマンドライン・スイッチ `-s` を使用して起動することができます。例えば、基本的な Quartus シェルを実行するには、コマンド・プロンプトで `quartus_sh -s` を入力します。

```
% quartus_sh -s
Info:
*****
Info: Running Quartus II Shell
Info:
*****
Info: The Quartus II Shell supports all TCL commands in addition
Info: to Quartus II Tcl commands. All unrecognized commands are
Info: assumed to be external and are run using Tcl's "exec"
Info: command.
Info: - Type "exit" to exit.
Info: - Type "help" to view a list of Quartus II Tcl packages.
Info: - Type "help -pkg <package name>" to view a list of Tcl commands
Info: available for the specified Quartus II Tcl package.
Info: - Type "help -tcl" to get an overview on Quartus II Tcl usages.
Info:
*****
tcl>
```

Quartus II の Tcl 実装は、Quartus II 操作を実行するためのカスタム Tcl プロシージャを提供します。これらのプロシージャは、機能ごとに Tcl パッケージに分かれています。表 6-2 に、これらの Tcl パッケージと供給状況を示します。一部のパッケージは、実行コマンドが呼び出されるとデフォルトでロードされます。その他のパッケージは、それぞれの Tcl プロシージャが使用される前に明示的にロードする必要があります。特定のパッケージをロードするには、`load_package Tcl` プロシージャを使用します。例えば、`quartus_sh` シェルにフロー・パッケージをロードするには、以下の Tcl 文を実行します。

```
tcl> load_package flow
```


 すべての実行コマンドが、すべての Tcl パッケージをサポートしているとは限らない点に注意してください。

表 6-2. Quartus II 実行コマンドでの Tcl パッケージのサポート (1 / 2)		
コマンド名	サポートされる Tcl パッケージ	デフォルトでロードされるか？
quartus_sta	device	ロードされる
	misc	ロードされる
	flow	ロードされない
	project	ロードされる
	report	ロードされる
	sdc	ロードされる
	sta	ロードされる
quartus_sh	device	ロードされる
	flow	ロードされない
	misc	ロードされる
	project	ロードされる
	report	ロードされない
quartus_tan	advanced_timing	ロードされない
	device	ロードされない
	flow	ロードされない
	logiclock	ロードされない
	Misc	ロードされる
	project	ロードされる
	report	ロードされない
	timing	ロードされる
timing_report	ロードされない	
quartus_cdb	backannotate	ロードされない
	chip_editor	ロードされない
	device	ロードされる
	flow	ロードされない
	logiclock	ロードされない
	misc	ロードされる
	project	ロードされる
	report	ロードされない

表 6-2. Quartus II 実行コマンドでの Tcl パッケージのサポート (2 / 2)

コマンド名	サポートされる Tcl パッケージ	デフォルトでロードされるか？
quartus_sim	device	ロードされる
	flow	ロードされない
	misc	ロードされる
	project	ロードされる
	report	ロードされる
	simulator	ロードされる

表 6-2 で引用された各 Tcl パッケージの簡単な説明を表 6-3 に示します。



ロードされる Tcl パッケージを見つけるには、コマンド `quartus_???` `--tcl_eval help` を使用します。例：  
`quartus_sta --tcl_eval help.`

表 6-3. Quartus II の Tcl パッケージの説明

Tcl パッケージ	説明
advanced_timing	タイミング・ネットリストを横断し、タイミング・モードに関する情報を取得する。
backannotate	アサインメントをバック・アノテートする。
chip_editor	Chip Editor でリソースの使用および配線を識別、変更する。
database_manager	バージョン互換のデータベース・ファイルを管理する。
device	デバイス・データベースからデバイスおよびファミリー情報を取得する。
flow	プロジェクトのコンパイル、コマンドライン実行コマンドおよび他の共通フローの実行。
logiclock	LogicLock 領域の作成と管理。
misc	その他のタスクの実行。
project	プロジェクトおよびリビジョンの作成と管理、およびタイミング・アサインメントを含むプロジェクト・アサインメントの作成。
report	レポート・テーブルからの情報の取得とカスタム・レポートの作成。
simulator	シミュレーションの設定と実行。
stp	SignalTap® II アナライザの操作。
timing	遅延情報を持つタイミング・ネットリストのアノテート、タイミング・パスの計算およびレポート。
timing_report	タイミング・パスのリスト。

Quartus II コマンドライン実行コマンドおよび Tcl シェルは、Microsoft Windows、Linux、および Unix プラットフォームを含むすべての Quartus II オペレーティング・システムでサポートされています。



Quartus II Tcl パッケージとそれらで使用可能な Tcl プロシージャについて詳しくは、「Quartus II Scripting Reference Manual」の「Tcl Packages and Commands」の章を参照してください。

## コマンドライン処理

インタラクティブ Tcl シェルに加えて、Quartus II コマンドライン実行コマンドは、Tcl スクリプトおよびコマンドを実行するためのコマンドライン・スイッチをサポートしています。これらのスイッチと併用すれば、コマンドライン実行コマンドは完了時に終了します。コマンドライン実行コマンドは、特定の Quartus II 動作を実行するためのスイッチも提供します。例えば、以下の c シェル・スクリプトは、トップレベルのデザイン・ファイルとエンティティ名を引数として取り込み、HardCopy II デザイン・フロー全体を通して実行します。

```
#!/bin/csh
quartus_sh --flow compile %1
quartus_cdb %1 --create_companion=%1_hcii
quartus_sh --flow compile %1 -c %1_hcii
quartus_cdb --compare=%1_hcii %1 -c %1
```

この例は、HardCopy II デザイン・フローを実行するための、おそらく最もシンプルな方法を示しています。プロジェクトに対して、デザイン I/O、ロケーション、およびタイミング制約を開発および適用した場合、これらの制約はスクリプトの実行中にインクルードされます。



Quartus II の実行コマンドおよびコマンドラインのオプションについて詳しくは、「Quartus II Scripting Reference Manual」の「Command-Line Executables」の章および「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。

## HardCopy II デザイン・ フロー

Quartus II ソフトウェアは、HardCopy II を最初に設計する場合のフロー、および Stratix II を最初に設計する場合のフローの両方をサポートします。Stratix II を最初に設計する場合のフローでは、以下を実行する必要があります。

- Stratix II FPGA プロトタイプのコmpایل
- Stratix II FPGA プロトタイプの検証
- プロトタイプ・デザインの HardCopy II デザインへの移行
- HardCopy II デザインのコmpایل
- アルテラ・デザイン・センタへの HardCopy II ファイルの転送

HardCopy II を最初に設計する場合のフローは似ていますが、HardCopy II ターゲット・デバイスのコンパイルから開始します。HardCopy II のコンパイルが正常に完了したら、デザインは Stratix II ターゲットに移行されます。

Quartus II ソフトウェアの HardCopy II デザイン・フローを図 6-1 に示します。デザインを開始するには、Stratix II FPGA プロトタイプに対する新しいプロジェクトおよびリビジョンを作成します。Quartus II 設定を I/O アサイメントおよびタイミング制約とともに適用します。Stratix II プロトタイプ・リビジョンをコンパイル（合成、配置配線、およびアセンブリ）し、完全なレイアウトをタイミング・クロージャおよびエラーなしで生成します。ここで、必要な追加の機能およびタイミング検証を実行し、プロトタイプをハードウェアに実装し検証することができます。

FPGA プロトタイプが検証されたら、HardCopy II デザインをコンパイルできます。FPGA プロトタイプの HardCopy II コンパニオン・リビジョンの作成から始めます。

1. FPGA プロトタイプの HardCopy II コンパニオン・リビジョンを作成します。すべてのデザイン設定と制約は、新しいコンパニオン・リビジョンに自動的に移行されます。
2. HardCopy II リビジョンをコンパイルします。コンパイルが実行されると、Design Assistant がエラーをチェックします。コンパイルが完了したら、エラーを修正して、Quartus II レポートに現れる障害を解決しなければなりません。
3. HardCopy II コンパニオン・リビジョン比較ツールを実行して、HardCopy II のデザインを FPGA プロトタイプと比較します。比較ツールは 2 つのリビジョン間の構造的な等価性と一貫性をチェックします。
4. 不一致がなければ、アルテラのデザイン・センタに HardCopy II デザイン・ファイルを転送する準備を行うことができます。


 Quartus II ソフトウェアでのデザイン検証に加えて、フローでは Synopsys 社の PrimeTime を使用してスタティック・タイミング解析 (STA) を実行するのに必要なファイルを生成することができます。

図 6-1. HardCopy II デザイン・フロー

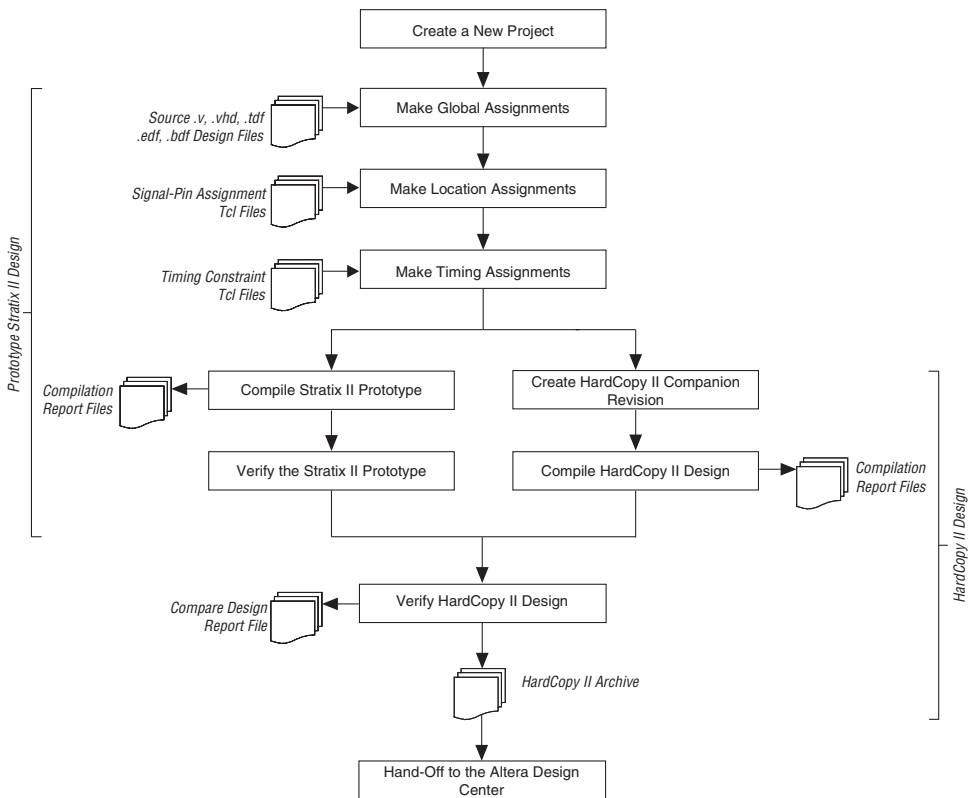


図 6-1 のデザイン・フローは、Stratix II FPGA プロトタイプ・デザインで始まり、このデザインを HardCopy II デバイス・ターゲットに移行するか、または HardCopy II ターゲットで最初に設計し、このデザインを FPGA プロトタイプ作成のために Stratix II ターゲットに移行します。両方のケースのデザイン・フローを図 6-1 に示します。



Quartus II GUI を使用して HardCopy II デザインを完成させるための HardCopy II デザイン・フローおよび代替方法については、「Quartus II ハンドブック」の「Quartus II Support for HardCopy II Devices」の章または「HardCopy シリーズ・ハンドブック Volume 1」の「HardCopy II Design Considerations」の章を参照してください。

以下の項では、図 6-1 に示すフローの各ステップを記述し、インタラクティブ Tcl シェルを使用してどのように各ステップを完了するかを説明します。

## 新規プロジェクトの作成

Quartus II ソフトウェアの FPGA および HardCopy デザインの両方とも、プロジェクトの使用を中心に展開します。新規デザインに取りかかる前に、プロジェクトを作成する必要があります。プロジェクトには、ソース・デザイン・ファイル (RTL および回路図)、Quartus II ツール設定、およびピン位置およびタイミング制約のセットが含まれています。プロジェクトにはデザインに対する多くの異なるリビジョンを含めることができますが、各リビジョンは独自のデザイン制約、ターゲット・デバイス設定、Quartus II ソフトウェア設定の組み合わせを持つことができます。プロジェクトは明示的に開いてから、他の操作を実行しなければなりません。別のプロジェクトまたはリビジョンに切り替えるには、現在のプロジェクトを閉じる必要があります。

この項では、Tcl コマンドを使用してプロジェクトの管理に関連する各種操作を詳細に説明します。

### Stratix II プロトタイプ・プロジェクトの作成

新しい Stratix II プロトタイプのプロジェクトを作成するには、**project\_new** Tcl コマンドを使用します。このコマンドの構文は、次のとおりです。

```
tcl> project_new [-family <family>] [-overwrite] \  
    [-part <part>] [-revision <revision_name>] \  
    <project_name>
```

このコマンドに必要な引数はプロジェクト名 *<project name>* だけです。ただし、この時点でもターゲット・デバイス・ファミリー、パーツ・コード、リビジョン名を指定できます。デフォルトでは、リビジョン名はプロジェクト名と同じです。デバイス・ファミリーとパーツ・コードは、後で **set\_global\_assignment** コマンドを使用して設定することができます。例えば、*demo\_design* というデフォルトのリビジョン名を持つ *demo\_design* と呼ぶプロジェクト、および未指定のターゲット・デバイス・ファミリーまたはパーツを作成するには、次の Tcl コマンドを実行します。

```
tcl> project_new demo_design
```

新しいプロジェクトを作成すると、カレント・ディレクトリに Quartus 設定ファイル (QSF) と Quartus II プロジェクト・ファイル (QPF) が作成されます。加えて、Quartus II データベース・ファイルを格納するのに

使用される **db** サブディレクトリが作成されます。demo\_design プロジェクト例の場合、プロジェクト・ディレクトリに以下のファイルが作成されます。

```
demo_design.qpf
demo_design.qsf
db/
    demo_design.db_info
```


## プロジェクトのオープン

**project\_new** コマンドを使用すると、作成されたプロジェクトが自動的に開きます。将来の Quartus II セッションで、またはプロジェクトを開じた場合は、**project\_open** コマンドでプロジェクトを開かなければなりません。**project\_open** コマンドの構文は、次のとおりです。

```
tcl> project_open [-current_revision] \
    [-revision <revision_name>] <project_name>
```

例えば、プロジェクト demo\_design のデフォルトのリビジョンを開くには、次の Tcl コマンドを実行します。

```
tcl> project_open demo_design
```

 プロジェクトの Stratix II リビジョンと HardCopy II リビジョンに、一貫性のある名前を付けると実践的です。これによって、リビジョンを簡単に区別できます。例えば、リビジョンに *projectname\_fpga* と *projectname\_hcii* を付けると、Stratix II リビジョンと HardCopy II リビジョンを簡単に区別できます。

## プロジェクトを閉じる

Quartus II プロジェクト・セッションを終了する前に、**project\_close** コマンドを使用して Quartus II プロジェクトを閉じてください。これにより、プロジェクトに行った変更は Quartus II QSF ファイルに書き込まれます。**project\_close** コマンドの構文は、次のとおりです。

```
tcl> project_close [-dont_export_assignments]
```

## 新規プロジェクトのスクリプト例

以下のスクリプトは、Tclコマンドを使用して、リビジョン名 `demo_design_fpga` を持つプロジェクト `demo_design` と呼ぶプロジェクトを開く、および閉じる方法を示しています。プロジェクトが存在しない場合は作成されます。このスクリプトでは、**project\_exists** および **project\_open** Tcl コマンドを利用しています。

```
## プロジェクトを開く、および閉じるための Tcl スクリプト例
```

```
## プロジェクト demo_design を開く。このプロジェクトが存在しない場合は、  
## 作成する。  
if [is_project_open] project_close  
if [project_exists demo_design] {  
    project_open demo_design -revision demo_design_fpga  
} else {  
    project_new demo_design -revision demo_design_fpga  
}
```

```
## 他の Tcl コマンドをここに含める。...
```

```
## プロジェクト demo_design を閉じて、設定の変更を  
## demo_design.qsf に書き込む。  
project_close
```

```
## スクリプトの終わり
```



上記およびその他の有用なプロジェクト関連コマンドについて詳しくは、「Quartus II Scripting Reference Manual」の「Tcl Packages and Commands」の章にある「Project」の項を参照してください。

## グローバル・アサインメントの作成

### HardCopy II デザインの初期化

HardCopy II デザインの場合、Quartus II プロジェクトが作成された後、以下の主な操作が必要です。

- デザイン・ソース・ファイル (Verilog、VHDL、AHDL、EDIF、および BDF ファイル) を指定します。
- Stratix II プロトタイプ・ターゲット・ファミリおよびデバイス名を指定します。
- HardCopy II コンパニオン・リビジョンおよびマイグレーション・デバイスを指定します。
- Design Assistant を有効にします。
- 推奨される HardCopy II を Quartus II ツール設定固有のものにします。

これらに加えて、この時点で、合成や配置配線などのツールに影響を与えるその他のプロジェクト設定を行うことができます。

上記の操作は、**set\_global\_assignment** コマンドを使用して実行されます。このコマンドの構文は、次のとおりです。

```
tcl> set_global_assignment [-comment<comment>] \  
    [-disable] [-entity <entity_name>] -name <name> \  
    [-remove] [-section_id <section_id>] <value>
```

**set\_global\_assignment** コマンドの最も重要なパラメータは、**<name>** と **<value>** です。**<name>** 引数は設定する Quartus II のグローバル変数を指定し、**<value>** はその変数に割り当てられた新しい値です。

HardCopy II デザインを初期化する際のステップの 1 つは、Design Assistant をオンにすることです。GUI で実行するとき、Design Assistant はデザインの Stratix II フェーズと HardCopy II フェーズの両方を実行するためのビジュアル・チェックリストを提供します。初めてのユーザにとって、これが HardCopy II プロジェクトを正常に完了するための強力なガイドとなります。

HardCopy II プロジェクトの主なグローバル変数は、表 6-4 にリストされています。

表 6-4. 主な HardCopy II デザイン設定	
グローバル変数名 <name>	値の説明 <value>
VERILOG_FILE	Verilog ファイル名
VHDL_FILE	VHDL ファイル名
AHDL_FILE	アルテラ HDL ファイル名
EDIF_FILE	EDIF ファイル名
BDF_FILE	アルテラ回路図ファイル名
FAMILY	デバイス・ファミリ名。例: Stratix II
DEVICE	プロトタイプ FPGA のターゲット・デバイス名
TOP_LEVEL_ENTITY	トップレベルのデザイン・エンティティ名またはモジュール名
DEVICE_TECHNOLOGY_MIGRATION_LIST	HardCopy II のターゲット・デバイス名
COMPANION_REVISION	HardCopy II のデザイン・リビジョン名
ENABLE_DRC_SETTINGS	Design Assistant をオンにする。
USE_TIMEQUEST_TIMING_ANALYZER	TimeQuest をデフォルトのタイミング・アナライザとして設定 <ON>
SDC_FILE	TimeQuest 制約のファイル <constraint_file.sdc>.
クラシック・タイミング・アナライザを使用するときには、以下の設定のみ必要です。クラシック・タイミング・アナライザの使用は推奨されません。	
REPORT_IO_PATHS_SEPARATELY	入力および出力の最小および最大タイミング結果に対する個別レポート・パネルを作成します。
FLOW_ENABLE_TIMING_CONSTRAINT_CHECK	タイミング制約の完全性がチェックされます (すべてのクロック・ドメインの制約と最小および最大制約がすべての I/O パスに対して設定される)。
DO_COMBINED_ANALYSIS	タイミング解析は、それぞれ高速および低速動作条件、ベスト・ケースおよびワースト・ケースのタイミングに対して実行されます。
IGNORE_CLOCK_SETTINGS	これはオフにする必要があります。
ENABLE_RECOVERY_REMOVAL_ANALYSIS	非同期コントロールおよびリセット信号でのリカバリおよびリムーバルのタイミングを検証します。
ENABLE_CLOCK_LATENCY	クロック・レイテンシがタイミング解析に含まれ、クロック挿入のタイミングおよびクロック・スキューが評価されます。

DEVICE および DEVICE\_TECHNOLOGY\_MIGRATION\_LIST 変数は、Stratix II プロトタイプ・デザインおよび HardCopy II デザインに使用される部分です。移行を可能にするには、選択された Stratix II プロトタイプ・デバイスが選択された HardCopy II デバイスと互換性がなければなりません。これらのデバイスの有効な組み合わせは、表 6-5 にリストされています。

DEVICE\_TECHNOLOGY\_MIGRATION\_LIST 変数の場合、表 6-5 にリストされる HardCopy II のパーツ名が使用されます。DEVICE 変数の場合、Stratix II のパーツ名にはそのパーツのスピード・グレードが含まれます。スピード・グレードは、工業用 (I) またはコマーシャル (C)、およびスピード・インジケータ (番号 3、4、または 5) を示す 2 文字のコードです。例えば、-4 の一般用パーツは、2 文字のスピード・グレード C4 を使用して示されます。この 2 文字のスピード・グレードは、Stratix II のパーツ名に付加され、DEVICE 変数の値文字列を形成します。

表 6-5. HardCopy II に対する Stratix II プロトタイプ・オプション (1 / 2)	
HardCopy II	Stratix II プロトタイプ・パーツ
HC210F484C HC210W484C	EP2S30F484C3 EP2S30F484C4 EP2S30F484C5 EP2S30F484I4
	EP2S60F484C3 EP2S60F484C4 EP2S60F484C5 EP2S60F484I4
	EP2S90H484C4 EP2S90H484C5
HC220F672C	EP2S60F672C3 EP2S60F672C4 EP2S60F672C5 EP2S60F672I4
HC220F780C	EP2S90F780C4 EP2S90F780C5
	EP2S130F780C4 EP2S130F780C5

表 6-5. HardCopy II に対する Stratix II プロトタイプ・オプション (2/2)	
HardCopy II	Stratix II プロトタイプ・パーツ
HC230F1020C	EP2S90F1020C3 EP2S90F1020C4 EP2S90F1020C5 EP2S90F1020I4
	EP2S130F1020C3 EP2S130F1020C4 EP2S130F1020C5 EP2S130F1020I4
	EP2S180F1020C3 EP2S180F1020C4 EP2S180F1020C5 EP2S180F1020I4
HC240I020C	EP2S180F1020C3 EP2S180F1020C4 EP2S180F1020C5 EP2S180F1020I4
HC240F1508C	EP2S180F1508C3 EP2S180F1508C4 EP2S180F1508C5 EP2S180F1508I4

以下の 2 つの Tcl コマンドは、DEVICE および DEVICE\_TECHNOLOGY\_MIGRATION\_LIST 変数の設定を示します。

```
tcl> set_global_assignment -name DEVICE EP2S90F1020C4
tcl> set_global_assignment -name \
    DEVICE_TECHNOLOGY_MIGRATION_LIST HC230F1020C
```

## Design Assistant

ENABLE\_DRC\_SETTINGS グローバル変数をオンにして、デザイン・プロセスの初めに Design Assistant をオンにしなければなりません。

```
tcl> set_global_assignment \  
-name ENABLE_DRC_SETTINGS ON
```

Design Assistant は、プロトタイプ Stratix II および HardCopy II デザイン・フローの両方の各ステップと並行して動作します。Design Assistant がオンのとき、Quartus II ソフトウェアはチェックを行って、プロジェクトがすべての HardCopy II デザイン・ルールおよびデザイン要件の下でフルにコンパイルされるようにします。



Design Assistant について詳しくは、「HardCopy シリーズ・ハンドブック Volume 1」の「Design Guidelines for HardCopy II Devices」の章、および「Quartus II ハンドブック」の「Quartus Support for HardCopy II Devices」の章を参照してください。

## グローバル・アサインメント作成のための Tcl スクリプト例

以下の Tcl スクリプト例は、HardCopy II プロジェクトに対するグローバル制約の適用を示します。

```
## HardCopy II デザインのためのグローバル・アサインメント・スクリプト例
## このスクリプトは EP2S90 Stratix II
## プロトタイプ FPGA ターゲットおよび HC230 HardCopy II ターゲットの設定を適用する。

## ソース・デザイン・ファイルの設定
## =====
set_global_assignment -name VERILOG_FILE demo_design.v
set_global_assignment -name VERILOG_FILE example_ram.v

## Stratix II プロトタイプ FPGA ターゲットの設定
## =====
set_global_assignment -name FAMILY "Stratix II"
set_global_assignment -name DEVICE EP2S90F1020C4
set_global_assignment -name TOP_LEVEL_ENTITY demo_design

## HardCopy II コンパニオン・リビジョンおよびターゲットの設定
## =====
set_global_assignment -name COMPANION_REVISION_NAME \
                      demo_design_hardcopyii
set_global_assignment -name DEVICE_TECHNOLOGY_MIGRATION_LIST HC230F1020

## HardCopy II に要求される Design Assistant のアサインメントおよび設定
## =====
set_global_assignment -name ENABLE_DRC_SETTINGS ON
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 1
set_global_assignment -name REPORT_IO_PATHS_SEPARATELY ON

## 以下のアサインメントはクラシック・タイミング・アナライザのみに対するもので、
## TimeQuest では使用されない。
## =====
set_global_assignment -name FLOW_ENABLE_TIMING_CONSTRAINT_CHECK ON
set_global_assignment -name DO_COMBINED_ANALYSIS ON
set_global_assignment -name IGNORE_CLOCK_SETTINGS OFF

set_global_assignment -name ENABLE_RECOVERY_REMOVAL_ANALYSIS ON
set_global_assignment -name ENABLE_CLOCK_LATENCY ON

## スクリプトの終わり
```

## I/O アサインメン トの作成

プログラマブル I/O セルの使用と、特定のピンおよびパッケージに対するそれらのセルの可用性は複雑なルールで支配されているため、アルテラではピン・プランニングツールおよび Quartus II GUI のアサインメント・エディタを使用して I/O アサインメントを実行することを強くお勧めしています。これらのツールにより、各ピンおよび I/O セルに関するすべてのルールが正しく適用されます。Quartus II GUI は、すべての I/O アサインメントと仕様を含む Tcl スクリプトをエクスポートできます。I/O アサインメントは、ここでは簡単に説明するにとどめます。



Quartus II アサインメント・エディタおよびピン・プランナ・ツールを使用した I/O の位置およびタイプ・アサインメントについて詳しくは、「Quartus II ハンドブック Volume 2」の「アサインメント・エディタ」の章を参照してください。

この項では、I/O 仕様は以下の 2 つのパートで検討されます。

- ピン・アサインメント
- I/O タイプ・アサインメント

### ピン・アサインメント

デザイン I/O 信号は、**set\_location\_assignment** コマンドを使用してパッケージ・ボールに割り当てられます。このコマンドの構文は、次のとおりです。

```
tcl> set_location_assignment [-comment <comment>] \  
    [-disable] [-remove] -to <destination> <value>
```

ここでは、<destination> はパッケージ・ボール名、<value> はデザイン I/O 信号名です。BGA パッケージと FBGA パッケージについては、ボール名は PIN\_<coordinate> の形式になります。例えば、デザイン I/O 信号 data\_out[15] をパッケージボール AL17 に割り当てる場合は、次のようにします。

```
tcl> set_location_assignment -to PIN_AL17 data_out[15]
```

### I/O タイプとパラメータの設定

I/O タイプとパラメータの仕様については、**set\_instance\_assignment** コマンドが使用されます。このコマンドの構文は、次のとおりです。

```
tcl> set_instance_assignment [-comment <comment>] \  
    [-disable] [-entity <entity_name>] \  
    [-from <source>] -name <name> [-remove] \  
    [-section_id <section_id>] \  
    \
```

```
[-to <destination>] <value>
```

アサインメント名 <name> は、I/O 仕様が適用されることを示すために、IO\_STANDARD に設定します。対応する I/O 信号は、-to <destination> で指定されます。引数 <destination> は、レベルや規格など I/O タイプに関する詳細を提供する文字列です。表 6-6 に、HardCopy II デバイスでサポートされる I/O 規格に対応する文字列を示します。

I/O タイプまたは <name>	説明
LVTTTL	LVTTTL I/O
LVC MOS	LVC MOS I/O
"3.3-V PCI"	3.3-V PCI I/O
"3.3-V PCI-X"	3.3-V PCI X I/O
"1.5 V"	1.5-V I/O
"1.8 V"	1.8-V I/O
"2.5 V"	2.5-V I/O
"1.5-V HSTL CLASS I"	QDR II SRAM 1.5-V I/O
"1.5-V HSTL CLASS II"	QDR II SRAM 1.5-V I/O
"1.8-V HSTL CLASS I"	QDR II SRAM/RLDRAM II 1.8-V I/O
"1.8-V HSTL CLASS II"	QDR II SRAM/RLDRAM II 1.8-V I/O
"DIFFERENTIAL 1.5-V HSTL CLASS I"	メモリ・クロック・インタフェース
"DIFFERENTIAL 1.5-V HSTL CLASS II"	メモリ・クロック・インタフェース
"DIFFERENTIAL 1.8-V HSTL CLASS I"	メモリ・クロック・インタフェース
"DIFFERENTIAL 1.8-V HSTL CLASS II"	メモリ・クロック・インタフェース
"DIFFERENTIAL 1.8-V SSTL CLASS I"	DDR2 SDRAM
"DIFFERENTIAL 1.8-V SSTL CLASS II"	DDR2 SDRAM
"DIFFERENTIAL SSTL-2"	DDR SDRAM
"DIFFERENTIAL 2.5-V SSTL CLASS II"	DDR SDRAM
"SSTL-18 CLASS I"	DDR2 SDRAM
"SSTL-18 CLASS II"	DDR2 SDRAM
"SSTL-2 CLASS I"	DDR SDRAM
"SSTL-2 CLASS II"	DDR SDRAM
LVDS	2.5-V 差動シグナリング
HYPERTRANSPORT	2.5-V 差動シグナリング
LVPCL	差動

`set_instance_assignment` コマンドを使用して、他の多くの I/O パラメータを指定できます。より一般的なパラメータの一部を、表 6-7 に示しています。

<name> 設定	<value> 設定	説明
<code>weak_pull_up_resistor</code>	on	ピンにウィーク・プルアップ抵抗を実装。
<code>output_pin_load</code>	整数	出力ピンまたは双方向ピンの容量性負荷。単位 pF。
<code>fast_output_register</code>	on	I/O セルまたは隣接 LAB に、高速出力レジスタを実装。
<code>fast_output_enable_register</code>	on	I/Oセルまたは/および隣接LABに、高速出力イネーブル・レジスタを実装。
<code>fast_input_register</code>	on	I/O セルまたは隣接 LAB に、高速入力レジスタを実装。
<code>current_strength_new</code>	2 mA, 4 mA, 8 mA, 10 mA, 12 mA, 16 mA, 18 mA, 20 mA, 24 mA minimum_current または maximum_current	出力ピンまたは双方向ピンのドライブ強度。
<code>stratixii_termination</code>	差動 “キャリブレーション付き直列 25Ω” “キャリブレーションなし直列 25Ω” “キャリブレーション付き直列 50Ω” “キャリブレーションなし直列 50Ω”	I/Oピン用On-Chip Termination (またはインピーダンス・マッチング)。



HardCopy II デバイスの I/O 可用性について詳しくは、「HardCopy シリーズ・ハンドブック Volume 1」の「I/O の構造と機能」の項を参照してください。

## I/O アサインメントのスク립ト例

以下の Tcl スクリプト例では、いくつかの異なる I/O 制約を指定しています。

```
## シグナル・ボール・アサインメント
set_location_assignment PIN_AH5 -to addr_out[0]
set_location_assignment PIN_AH6 -to addr_out[1]
set_location_assignment PIN_AJ5 -to data_in[0]
set_location_assignment PIN_AJ6 -to data_in[1]
set_location_assignment PIN_AJ32 -to resetn
set_location_assignment PIN_AM17 -to ref_clk

# I/O タイプとパラメータのアサインメント
set_instance_assignment -name IO_STANDARD "1.5-V HSTL CLASS II" -to addr_out[0]
set_instance_assignment -name IO_STANDARD "1.5-V HSTL CLASS II" -to addr_out[1]
set_instance_assignment -name IO_STANDARD "1.5-V HSTL CLASS II" -to data_in[0]
set_instance_assignment -name IO_STANDARD "1.5-V HSTL CLASS II" -to data_in[1]
set_instance_assignment -name IO_STANDARD LVDS -to resetn
set_instance_assignment -name IO_STANDARD LVCMOS -to ref_clk

set_instance_assignment -name fast_input_register on -to data_in[0]
set_instance_assignment -name fast_input_register on -to data_in[1]
set_instance_assignment -name fast_output_register on -to addr_out[0]
set_instance_assignment -name fast_output_register on -to addr_out[1]

set_instance_assignment -name output_pin_load 10 -to addr_out[0]
set_instance_assignment -name output_pin_load 10 -to addr_out[1]
set_instance_assignment -name current_strength_new 16mA -to addr_out[0]
set_instance_assignment -name stratixii_termination "キャリブレーションなしの直列 25Ω" \
-to data_in[1]
```

## タイミング 制約の 割り当て

### デザイン・タイミング制約のプランニング

タイミング制約は、Quartus II ソフトウェアでコンパイルされるデザインが特定のタイミング要件に適合することを保証します。FPGA をターゲットにした場合は、タイミング制約の全セットを適用するのではなく、プロトタイプ・システムでタイミング問題が発生したときに、これらを修正する方法を選択することができます。ただし HardCopy デバイスは、リコンフィギュレーションを使用して変更し、タイミング問題を修正することはできないため、デザインを完全に制約することが非常に重要です。デザインを完全に制約しなければ、プロトタイプの Stratix II FPGA と HardCopy II デバイス間でタイミング特性が大きく異なることとなります。デザインを完全に制約することによって、アルテラは Stratix II FPGA と HardCopy II デバイスの両方でユーザのタイミング仕様への完全準拠を保証できます。

HardCopy II デザインに対する最低限のタイミング制約は、次のようになります。

- 各クロック・ドメインに対するクロック設定 ( $F_{MAX}$ )
- 非同期リセット信号とコントロールI/O信号を含む、すべてのI/Oパスに対する最小および最大遅延

また、以下に対応するようにタイミング制約を開発するのがよいデザイン方法です。

- 特定のクロスクロック・ドメイン・タイミング要求
- フォルス・パス
- マルチサイクル・パス

TimeQuest では、タイミング制約は TimeQuest SDC 形式で記述され、SDC ファイルから読み込まれます。ファイルの一例は、*demo\_design.sdc* です。6-32 ページの「TimeQuest の使用」を参照してください。

クラシック・タイミング・アナライザでは、タイミング制約は専用の Tcl コマンドを使用し、**set\_instance\_assignment** コマンドでタイミング固有の属性を割り当てることによって適用されます。

この項では、Tcl コマンドを使用したタイミング制約開発の概要を示します。



タイミング制約について詳しくは、「Quartus II ハンドブック Volume 3」の「タイミング解析」の項を参照してください。

## システム・クロックの指定

適用すべき最も基本的な制約で、各クロック・ドメインのクロックが記述されます。各クロックには通常、次のようなパラメータが指定されます。

- クロック期間
- レイテンシ (LATE\_CLOCK\_LATENCY/EARLY\_CLOCK\_LATENCY アサインメント)
- 不確実性 (**set\_clock\_uncertainty** コマンド)

**set\_clock\_uncertainty** コマンドで指定されるクロックの不確実性は、ジッタを含むクロック期間にける不確実性をモデル化しており、しばしばターゲットのクロック周波数に少しのマージンを導入するために使用されます。以下の制約の例は、*clk\_a* と *clk\_b* の 2 つのクロック・

ドメインを持つデザインのクロック定義を示しています。この場合、両方のクロックとも 100 MHz で動作しますが、クロック・レイテンシとスキューは異なります。

```
## クロック clk_a と clk_b を定義する TimeQuest SDC 制約の例
create_clock -period 10.0 -name clk_a [get_ports clk_a]
set_clock_latency -source -late 3.0 clk_a
set_clock_latency -source -early 2.0 clk_a
set_clock_uncertainty -to clk_a 0.25

create_clock -period 10.0 -name clk_b [get_ports clk_b]
set_clock_latency -source -late 4.0 clk_b
set_clock_latency -source -early 3.0 clk_b
set_clock_uncertainty -to clk_b 0.25
```

## 入力/出力タイミング

システム・クロック・パラメータは、各クロック・ドメイン内のレジスタ間パスに対するセットアップおよびホールド・タイミングを定義します。I/O タイミング・パラメータは、I/O とレジスタ間、およびレジスタと I/O 間の記述に使用されます。

**set\_input\_delay** 制約は、チップ外部のソースから入力ピンまでの遅延を、定義済みのクロックを基準にして指定するのに使用されます。このコマンドの構文は、次のとおりです。

```
set_input_delay \
  -clock <clock name> \
  [-clock_fall] \
  [-rise | -fall] \
  [-max | -min] \
  [-add_delay] \
  [-reference_pin <pin or port>] \
  <delay value> \
  <port pin list>
```

<clock name> 引数は、遅延の基準クロックを指定します。<port pin list> 引数は、デザインのトップレベルの入力信号であり、<delay value> は外部遅延です。外部遅延は、-clock\_fall 引数が指定されていない場合、<clock> の正（立ち上がり）エッジから測定されます。-min 引数と -max 引数は、<delay value> がそれぞれ最小外部遅延または最大外部遅延かどうかを指定するのに使用されます。

**set\_output\_delay** 制約は、出力ピンから外部接続先までの遅延をクロックを基準にして指定することを除いて、**set\_input\_delay** 制約に類似しています。

```

set_output_delay \
  -clock <clock name> \
  [-clock_fall] \
  [-rise | -fall] \
  [-max | -min] \
  [-add_delay] \
  [-reference_pin <pin or port>] \
  <delay value> \
  <port pin list>

```

一例として、次の Tcl スクリプトは 2 つの I/O 信号の入力および出力の最小遅延と最大遅延を指定します。入力 data\_in[0] の最小および最大外部遅延は、それぞれ 3 ns と 7 ns です。出力 data\_out[0] の最小および最大外部遅延は、それぞれ 4 ns と 8 ns です。data\_in[0] の外部入力遅延は、クロック ref\_clk の正エッジを基準にし、data\_out[0] の外部出力遅延はクロック ref\_clk の負エッジを基準にします。

```

#set_input_delay と set_output_delay を使用して I/O タイミングを設定する Td スクリプト
set_input_delay -clock ref_clk -max 7.0 [get_ports data_in[0]]
set_input_delay -clock ref_clk -min 3.0 [get_ports data_in[0]]
set_output_delay -clock ref_clk -max 8.0 [get_ports data_out[0]]
set_output_delay -clock ref_clk -min 4.0 [get_ports data_out[0]]

```

## タイミング例外の作成

タイミングの例外は、クロック設定および I/O タイミング設定で対応されないタイミング制約を修正するために使用されます。最も一般的なタイミング例外はマルチサイクル・パスとフォルス・パスです。

TimeQuest では、マルチサイクル・パスは **set\_multicycle\_path** 制約を使用して記述されます。この制約の構文は、次のとおりです。

```
set_multicycle_path [-setup] [-hold] [-start]
```

クラシック・タイミング・アナライザでは、マルチサイクル・パスは **set\_multicycle\_assignment** コマンドを使用して記述されます。このコマンドの構文は、次のとおりです。

```

tcl> set_multicycle_assignment [-comment <comment>] \
  [-disable] [-end] [-from <from_list>] \
  [-hold] [-remove] [-setup] [-start] \
  [-to <to_list>] <path_multiplier>

```

いずれのタイミング・アナライザでも、マルチサイクル・アサインメントは、最大サイクル数を指定するには `-setup` 引数を使用し、パスの最小サイクル数を指定するには `-hold` 引数を使用して行われます。

フォルス・パスは、タイミング最適化または解析操作に指定できないパスを記述します。Quartus II ソフトウェアでは、フォルス・パスを記述する方法がいくつかあります。デフォルトにより、クラシック・タイミング・アナライザでは、双方向 I/O の出力側から入力側へのフィードバック、メモリ内の read-while-write パス、およびクロック・ドメイン・パスは、最適化またはタイミング解析中はタイミングが計られません。デフォルトにより、Time Quest ではクロック・ドメイン・パスのタイミングが計られます。



これらのデフォルト設定を変更するには、「Quartus II ハンドブック Volume 1」の「Quartus II による HardCopy シリーズ・デバイスのサポート」の章にある「タイミング設定」の項を参照してください。

TimeQuest では、**set\_false\_path** 制約は、タイミング最適化または解析に含めることができないパスを記述する場合に使用されます。この制約の構文は、次のとおりです。

```
tcl> set_false_path \
    [-from <from list>] \
    [-to <to list>] \
    [-thru <thru list>]
```

クラシック・タイミング・アナライザでは、フォルス・パスを制御する最も一般的なコマンドは、**set\_timing\_cut\_assignment** コマンドです。このコマンドの構文は、次のとおりです。

```
tcl> set_timing_cut_assignment \
    [-comment <comment>] \
    [-disable] \
    [-from <from_pin_list>] \
    [-remove] \
    [-to <to_pin_list>]
```

<from\_pin\_list> のノードと <to\_pin\_list> のノード間のすべてのパスは、タイミング最適化および解析操作から除外されます。

## TimeQuest SDC 制約の例

```
# タイミング・アサインメント
# =====
create_clock -period 10.0ns -name ref_clk ref_clk

set_clock_latency -late 3 ref_clk
set_clock_latency -early 2 ref_clk
set_clock_uncertainty -hold -to ref_clk 0.250ns
set_clock_uncertainty -setup -to ref_clk 0.250ns

# bus data_in[1:0] に対する 6ns (max) & 2ns (min) の入力遅延
```




```

[-check_ios] \
[-check_netlist] \
[-compile] \
[-compile_and_simulate] \
[-early_timing_estimate] \
[-eco] [-export_database] \
[-fast_model] \
[-generate_functional_sim_netlist] \
[-import_database]

```

プロトタイプ Stratix II および HardCopy II デザインに関連するスイッチを、表 6-8 に示します。

スイッチ	説明
analysis_and_elaboration	ターゲットのアルテラ・テクノロジーへの合成およびマッピングを実行
attempt_similar_placement	Attempt Similar Placement を実行
check_ios	I/O アサインメントを検証
check_netlist	ネットリストの構文チェックを実行
compile	Quartus II コンパイル・フローを実行
compile_and_simulate	コンパイルでは、シミュレーションも実行
early_timing_estimate	Early Timing Estimate を実行
eco	フィッタ ECO コンパイルを実行
export_database	バージョン互換のデータベースをエクスポート
fast_model	タイミング解析（高速モデル解析）を実行
generate_functional_sim_netlist	シミュレーション・ネットリストを生成
import_database	バージョン互換のデータベースをインポート

 execute\_flow コマンドの HardCopy スイッチは、HardCopy II デザインではなく、HardCopy Stratix デザインに使用されることに注意してください。

execute\_flow コマンドを実行する最も簡単な方法は、-compile スイッチを使用することです。

```
tcl> execute_flow -compile
```

このように **execute\_flow** コマンドを実行すると、Quartus II コンパイル・フローの以下の 4 ステージが、各ステージにデフォルト設定を使用して実行されます。

- 解析と合成
- フィッタ
- タイミング解析
- アセンブラ

デザイン・アシスタントおよびタイミング制約チェックは、Quartus II 設定ファイルで有効になっている場合に実行されます。

I/O アサインメントをチェックして、ダウンストリームのコンパイル動作での問題を回避する必要があります。これを行うには、execute\_flow コンパイルを以下の3つのステップに分割する必要があります。

1. tcl>execute\_flow-analysis\_and\_elaboration
2. tcl>execute\_flow-check\_ios
3. tcl>execute\_flow -compile

明確さと簡潔さを目的として、ここで示す Tcl フラグメントにはエラー・チェックが組み込まれていないことに注意してください。ただし、デザインを進める過程で成功を確認するコードを Tcl スクリプトに組み入れるのもよい方法です。execute\_flow プロシージャの場合、戻り値を Tcl catch コマンドで使用して成功または失敗を処理することができます。次の例は、これを行うための1つのオプションを示しています。

```
# コンパイルが成功したかどうか、および
# 個別メッセージを出力するかどうかを判断する。
if {[catch {execute_flow -compile} result]} {
    puts "\nResult: $result\n"
    puts
    "ERROR: Compilation failed. See report files.\n"
} else {
    puts "\nINFO: Compilation was successful.\n"
}
}
```



**execute\_flow** コマンドについて詳しくは、「Quartus II Scripting Reference Manual」の「Tcl Packages and Commands」の章の命令の説明を参照してください。

## HardCopy II デザインの コンパイル

Stratix II FPGA プロトタイプ・デザインがコンパイルされ、検証されると、HardCopy II バージョンのデザインをコンパイルできます。これは次の2段階のプロセスで行われます。

1. HardCopy II コンパニオン・リビジョンを作成します。

## 2. HardCopy II コンパニオン・リビジョンをコンパイルします。

HardCopy IIバージョンのデザインを作成するには、**execute\_hardcopyii** Tcl コマンドを、**-create\_companion** オプションを指定して実行します。

```
tcl> execute_hardcopyii -create_companion demo_design_hcii
```

このコマンドは、HardCopy II リビジョンのデータベースを初期化し、新しい QSF ファイルを作成し（この例では、**demo\_design\_hcii.qsf**）、Stratix II FPGA リビジョンのすべての制約が確実に移植されるようにします。


次に、Quartus II プロジェクトの現在作業中の HardCopy II リビジョンが HardCopy II リビジョンに変更され、デザインが HardCopy II デバイス・ターゲットに対してコンパイルされます。

```
tcl> set_current_revision demo_design_hcii
tcl> execute_flow -compile
```

プロトタイプ Stratix II リビジョンと同様に、実行される各ツールのプロジェクト・ディレクトリにレポート・ファイルが生成されます。

## レポート・ ファイルの 理解

**execute\_flow** コマンドは、プロジェクト・ディレクトリに多数のレポート・ファイルを生成します。これらのファイルは、コンパイル中にコンソールに表示されるメッセージを要約し、デザインに関する追加情報を提供します。各レポート・ファイルの名前は、**<revision><tool short name>.summary** と **<revision><tool short name>.rpt** の形式をとります。ここで、**<revision>** は現在のデザインのリビジョン名です。**.summary** ファイルには、メッセージの簡単な要約と、ツールによる結果が格納され、**.rpt** ファイルにはより詳細なメッセージと情報が格納されます。HardCopy II プロジェクトの場合、2 組のレポート・ファイルが生成されます。1 つは Stratix II プロトコル FPGA リビジョンのファイル、もう 1 つは HardCopy II リビジョンのファイルです。表 6-9 に、さまざまなレポート・ファイルを説明しています。

 Tcl report パッケージは、Quartus II のフィッタおよびタイミング解析エンジンに関連するレポート・ファイルをカスタマイズおよび管理するための強力なプロシージャの集合を提供します。



レポート・ファイルのカスタマイズと管理については、「Quartus II Tcl Reference Manual」の「Tcl Packages and Commands」のレポートの項を参照してください。

表 6-9. Stratix II コンパイル・レポート・ファイルの説明

スイッチ	ツール	説明
<revision>.map.rpt	Analysis & Synthesis	合成設定、ソース・ファイル、メッセージ、リソース利用。
<revision>.map.eqn	Analysis & Synthesis	実装等式およびデバイス・リソースのインスタンス化。
<revision>.fit.rpt	フィッタ	フィッタ設定、レイアウト最適化、リソース、ピン配置、およびメッセージ。
<revision>.fit.eqn	フィッタ	実装された等式およびフィッティング後のデバイス・リソースのインスタンス化。
<revision>.drc.rpt	デザイン・アシスタント	デザイン・ルール設定、違反、およびメッセージ。
<revision>.upc.rpt	Timing Constraint Checker	制約の適用範囲情報。
<revision>.asm.rpt	アセンブラ	アセンブラ設定、.pof 出力ファイルおよび.sof 出力ファイル・オプション、およびメッセージ。
<revision>.rec.rpt	コンパニオン・リビジョンの比較	HardCopy II リビジョンと Stratix II プロトタイプ・デザインの構造の比較に関するステータス・レポート。
<revision>.flow.rpt	フロー	フロー内の各ツールのリソース要約と実行時間。このレポートはフロー内のツールが完了すると更新されます。
<revision>.sta.rpt	TimeQuest	TimeQuest タイミング解析レポート。

## FPGA リビジョンと HardCopy リビジョンの 比較

HardCopy II プロジェクトをアルテラ・デザイン・センターに提出する前に、Stratix II プロトタイプの FPGA リビジョンと照合してチェックする必要があります。これを行うには、quartus\_sh シェルから `-compare` オプションを指定した `execute_hardcopyii` Tcl コマンドを実行します。

```
tcl> execute_hardcopyii -compare
```

このコマンドを実行すると、プロジェクト・ディレクトリにレポート・ファイルとサマリ・ファイルが生成されます。これらのファイルは、<revision\_name>.rec.rpt および <revision\_name>.rec.summary と呼ばれます。このコマンドは、次の項目が HardCopy II のデザイン・ルールに準拠し、HardCopy II リビジョンと Stratix II リビジョンが矛盾しないことを確認します。

- ソース・デザイン・ファイルとデバイス・ネットリスト・ファイル
- ユーザ・クロック・アサインメント
- タイミング制約（アサインメント）
- I/O 位置アサインメントとタイプ・アサインメント
- PLL パラメータ

- メモリ実装パラメータ
- DSP 実装パラメータ
- グローバル・リソース・プロパティ
- 使用される他のすべてのデバイス・リソースのプロパティ

比較でのエラーまたは失敗はすべて、**.rec** レポート・ファイルで報告されます。**.rec** ファイルの一例を以下に示します。この例では、HardCopy II コンパニオン・リビジョン比較サマリ・テーブルのデザイン比較チェックはすべて合格のマークが付けられ、Quartus II ソフトウェアの HardCopy II デザインが終了し、アルテラ・デザイン・センタのバックエンド・エンジニアリング・チームにハンドオフする準備ができたことを示します。

比較サマリに示されたすべての失敗を解決しなければ、デザインを進めることはできません。

```
HardCopy II Companion Revision Comparison report for demo_design_hardcopyii
Wed Sep 20 15:30:07 2006
Version 6.0 Build 202 06/20/2006 Service Pack 1 SJ Full Version
```

```
-----
; Table of Contents ;
-----
```

1. Legal Notice
2. HardCopy II Companion Revision Comparison Summary
3. Atom Netlist Comparison Summary
4. DSP Information
5. HardCopy II Companion Revision Comparison Messages

```
+-----+
; HardCopy II Companion Revision Comparison Summary ;
+-----+
; HardCopy II Companion Revision Comparison Status ; Analyzed - Wed Sep 20 15:29:55 2006 ;
; Quartus II Version ; 6.0 Build 202 06/20/2006 SP 1 SJ Full Version ;
; Revision Name demo_dsign_hardcopyii ;
; Top-level Entity Name ; demo_design ;
; Family ; Stratix II ;
; Compare Status ; Passed (14/14) ;
; Source Files Compared ; Passed (121/121) ;
; Assignments Compared ; Passed ;
; User Clocks Compared ; Passed (0/0) ;
; Resource Counts Compared ; Passed (5/5) ;
; I/O Structure Compared ; Passed (130/130) ;
; Package Pins Compared ; Passed (1020/1020) ;
; PLL Structure Compared ; Passed (1/1) ;
; PLL Clocks Compared ; Passed (2/2) ;
; Timing Constraints Compared ; Passed (3/3) ;
; RAM Information Compared ; Passed (10/10) ;
; DSP Information Compared ; Passed (100/100) ;
; Global Resources Compared ; Passed (8/8) ;
; Atom Compared ; Passed (335084/335084) ;
; Atom Netlist Compared ; Passed (1/1) ;
+-----+
```

## スタティック・タイミン 解析の実行

**Quartus II ソフトウェアのスタティック・タイミング解析**  
Stratix II プロトタイプと HardCopy II リビジョンに対して行われたグローバル・アサインメントにより、高速および低速動作条件の両方に対してスタティック・タイミング解析 (STA) が実行され、セットアップとホールドの両方のタイミングが検証されます。

### TimeQuest の使用

タイミング解析は、次の 2 つのうちいずれかの方法でコンパイル・プロセスから独立して実行できます。

1. **execute\_module -tool sta** Tcl コマンドを使用して、基本 Quartus シェル `quartus_sh` 内から `quartus_sta` のタイミング解析 Tcl スクリプトを実行します。
2. `quartus_sta` インタラクティブ Tcl シェルを独立して実行し、Tcl プロンプトで Tcl コマンドおよびスクリプトを実行します。

### クラシック・タイミング・アナライザの使用

タイミング解析は、次の 2 つのうちいずれかの方法でコンパイル・プロセスから独立して実行できます。

1. **execute\_module -tool tan** Tcl コマンドを使用して、基本 Quartus シェル `quartus_sh` 内から `quartus_tan` のタイミング解析 Tcl スクリプトを実行します。
2. `quartus_tan` インタラクティブ Tcl シェルを独立して実行し、Tcl プロンプトで Tcl コマンドおよびスクリプトを実行します。



Quartus II ソフトウェアでのスタティック・タイミング解析の実行について詳しくは、「Quartus II ハンドブック」の「タイミング解析」の項を参照してください。



スタティック・タイミング解析に関連する Tcl コマンドについては、「Quartus II Scripting Reference Manual」の「Tcl Packages and Commands」にある「タイミング」の項を参照してください。

### PrimeTime でのスタティック・タイミング解析

Quartus II ソフトウェアは、Synopsys 社の PrimeTime で STA を実行するのに必要なファイルも生成できます。以下の例では、Tcl コマンドは Quartus II ソフトウェアに、STA の PrimeTime ファイルを生成するように指示しています。

```
## Tcl Script to Generate PrimeTime STA File Output
execute_module -tool sta -args --tq2pt
execute_module -tool eda -args "--tool primetime --format verilog --timing_analysis"
```

Quartus II ソフトウェアで生成されたファイルは、プロジェクト・ディレクトリのサブディレクトリ内に編成されます。例えば、Stratix II プロトタイプ・デザイン (demo\_design) をコンパイルした後、次の verilog (.vo) SDF (.sdo) と PrimeTime Tcl スクリプト (.tcl) が、プロジェクト・ディレクトリに作成されます。

```
timing\
    primetime\
        demo_design_v.sdo
        demo_design.pt.tcl
        demo_design.collections.sdc
        demo_design.constraints.sdc
```

Tcl スクリプトには、Quartus II ソフトウェアのコンパイル中に適用されるすべてのタイミング制約が含まれます。

## HardCopy II の Tcl スクリプト例

以下のスクリプトでは、ここまでで説明した Tcl のアイデアを quartus\_shTcl シェル用のトップレベル Tcl スクリプトにまとめています。このスクリプトは、demo\_design と呼ばれる HardCopy II デザインを実装します。demo\_design と呼ばれる新しいプロジェクトの作成から始まり、Stratix II FPGA プロトタイプのコンパイル、HardCopy II コンパニオン・リビジョンの作成、コンパニオン・リビジョンのコンパイルが行われます。最後に、リビジョン比較ツールが実行されて、両方のリビジョンに一貫性があることが検証されます。

この例では、グローバル、ピン、およびタイミング・アサインメント・スクリプトが、Tcl source コマンドを使用してトップレベル・スクリプトに読み込まれます。ソース・スクリプトは、トップレベル・スクリプト・リストに従ってリストされます。

### トップレベル・スクリプト demo\_design.tcl の例

```
## demo_design.tcl
## Top-level script for executing a HardCopy II design in quartus_sh -s
load_package flow

## Open or create the Stratix II FPGA prototype revision
if [is_project_open] project_close
if {[project_exists demo_design]} {
    project_open demo_design

} else {
    project_new demo_design
}
```

```
## Apply global design settings
source global_assignments.tcl

## Apply I/O assignments
source pin_assignments.tcl

## Apply FPGA timing constraints
source timing_assignments.tcl

## Compile the Stratix II FPGA prototype design
execute_flow -compile

# #Create and switch to the HardCopy II target revision
execute_hardcopyii -create_companion demo_design_hcii
project_close
project_open demo_design -revision demo_design_hcii

## Compile the HardCopy II design revision
execute_flow -compile

## Check the HardCopy II revision and make sure it matches the FPGA
## design
execute_hardcopyii -compare

## Generate a HardCopy II Handoff Report
execute_hardcopyii -handoff_report

## Archive the HardCopy II Handoff Files into
## the file named "demo_design_hcii_handoff.qar"
execute_hardcopyii -archive demo_design_hcii_handoff.qar

## Quit quartus_sh -s
qexit

## End of demo_design.tcl
```

### Global Assignments Script global\_assignments.tcl

ト ッ プ レ ベ ル・ス ク リ プ ト の demo\_design.tcl の global\_assignments.tcl スクリプト・ソースは、HardCopy II プロジェクトのグローバル変数、ターゲット・デバイス、リビジョン名を準備します。

```
## global_assignments.tcl

## Source Design File Settings
## =====
set_global_assignment -name VERILOG_FILE demo_design.v
set_global_assignment -name VERILOG_FILE example_ram.v

## Constraint File Settings for TimeQuest
## =====
set_global_assignment -name USE_TIMEQUEST_TIMING_ANALYZER ON
```

```

set_global_assignment -name SDC_FILE demo_design.sdc

## Stratix II Prototype FPGA Target Settings
## =====
set_global_assignment -name FAMILY "Stratix II"
set_global_assignment -name DEVICE EP2S90F1020C4
set_global_assignment -name TOP_LEVEL_ENTITY demo_design

## HardCopy II Companion Revision and Target Settings
## =====
set_global_assignment -name COMPANION_REVISION_NAME \
demo_design_hardcopyii
set_global_assignment -name DEVICE_TECHNOLOGY_MIGRATION_LIST HC230F1020

## Design Assistant Assignments and Settings Required for HardCopy II
## =====
set_global_assignment -name ENABLE_DRC_SETTINGS ON
set_global_assignment -name ERROR_CHECK_FREQUENCY_DIVISOR 1
set_global_assignment -name REPORT_IO_PATHS_SEPARATELY ON

## The following assignments are Classic Timing Analyzer only and
## are not used by TimeQuest.
## =====
set_global_assignment -name FLOW_ENABLE_TIMING_CONSTRAINT_CHECK ON
set_global_assignment -name DO_COMBINED_ANALYSIS ON
set_global_assignment -name IGNORE_CLOCK_SETTINGS OFF
set_global_assignment -name ENABLE_RECOVERY_REMOVAL_ANALYSIS ON
set_global_assignment -name ENABLE_CLOCK_LATENCY ON

## End of global_assignments.tcl

```

## ピン・アサインメント・スクリプト **pin\_assignments.tcl**

トップレベル・スクリプト `demo_design.tcl` から実行される `pin_assignments.tcl` スクリプトは、トップレベル・デザイン信号とボール・アサインメントの制約と I/O パラメータを指定します。

```

## pin_assignments.tcl
set_location_assignment PIN_AH5 -to addr_out[0]
set_location_assignment PIN_AH6 -to addr_out[1]
set_location_assignment PIN_AJ5 -to data_in[0]
set_location_assignment PIN_AJ6 -to data_in[1]
set_location_assignment PIN_AJ32 -to reseth
set_location_assignment PIN_AM17 -to ref_clk

## I/O Type and Parameter Assignments
set_instance_assignment -name IO_STANDARD "1.5-V HSTL CLASS II" -to addr_out[0]
set_instance_assignment -name IO_STANDARD "1.5-V HSTL CLASS II" -to addr_out[1]
set_instance_assignment -name IO_STANDARD "1.5-V HSTL CLASS II" -to data_in[0]
set_instance_assignment -name IO_STANDARD "1.5-V HSTL CLASS II" -to data_in[1]
set_instance_assignment -name IO_STANDARD LVDS -to reseth
set_instance_assignment -name IO_STANDARD LVCMOS -to ref_clk

set_instance_assignment -name fast_input_register on -to data_in[0]

```

```

set_instance_assignment -name fast_input_register on -to data_in[1]
set_instance_assignment -name fast_output_register on -to addr_out[0]
set_instance_assignment -name fast_output_register on -to addr_out[1]

set_instance_assignment -name output_pin_load 10 -to addr_out[0]
set_instance_assignment -name output_pin_load 10 -to addr_out[1]

## End of pin_assignments.tcl

```

## TimeQuest 制約ファイル demo\_design.sdc

TimeQuest は SDC ファイル *demo\_design.sdc* を読み込み、システム・クロック `ref_clk` のタイミング制約、および I/O からコアへのタイミング仕様を適用します。

```

## constraints.sdc
create_clock -period 10.0 MHz -name ref_clk [get_ports ref_clk]

set_clock_latency -late 3 ref_clk
set_clock_latency -early 2 ref_clk
set_clock_uncertainty -hold -to ref_clk 0.250
set_clock_uncertainty -setup -to ref_clk 0.250

# Input delay of 6ns (max) & 2ns (min) for bus data_in[1:0]
set_input_delay -clock ref_clk -max 6 [get_ports data_in]
set_input_delay -clock ref_clk -min 2 [get_ports data_in]
# Output delay of 6ns (max) & 2ns (min) for bus data_out[1:0]
set_output_delay -clock ref_clk -max 6 [get_ports data_out]
set_output_delay -clock ref_clk -min 2 [get_ports data_out]

# Don't care about timing on the resetn net. Set as false path
set_false_path -from [get_ports resetn]
## End of timing_assignments.tcl

```

## タイミング・アサインメント・スクリプト timing\_assignments.tcl

クラシック・タイミング・アナライザを使用する場合、`timing_assignments.tcl` スクリプトはトップレベル・スクリプト `demo_design.tcl` から実行されます。このスクリプトは、システム・クロック `ref_clk` のタイミング制約、および I/O からコアのタイミング仕様を適用します。

```

## timing_assignments.tcl
create_base_clock -fmax 10.0ns -target ref_clk ref_clk

set_instance_assignment -name LATE_CLOCK_LATENCY 3ns -to ref_clk
set_instance_assignment -name EARLY_CLOCK_LATENCY 2ns -to ref_clk
set_clock_uncertainty -hold -to ref_clk 0.250ns
set_clock_uncertainty -setup -to ref_clk 0.250ns

```

```
# Input delay of 6ns (max) & 2ns (min) for bus data_in[1:0]
set_input_delay -clk_ref ref_clk -max -to data_in 6.0ns
set_input_delay -clk_ref ref_clk -min -to data_in 2.0ns
# Output delay of 6ns (max) & 2ns (min) for bus data_out[1:0]
set_output_delay -clk_ref ref_clk -max -to data_out 6.0ns
set_output_delay -clk_ref ref_clk -min -to data_out 2.0ns

# Don't care about timing on the resetn net.Set as false path
set_timing_cut_assignment -from resetn
## End of timing_assignments.tcl
```

## まとめ

この章では、Quartus II インタラクティブ Tcl シェルを使用した HardCopy II デバイスのためのスクリプト・ベースのデザインについて説明しました。このアプローチは、リモート端末での Quartus II の実行、デザイン・フローの自動化、あるいはスクリプト環境での操作が快適な場合など、特定の状況での GUI ベース・デザインに代わるデザイン方法を提供します。

## 改訂履歴

表 6-10 に、本資料の改訂履歴を示します。

表 6-10. 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 6 月 v1.2	テキストのマイナーな編集。	
2006 年 12 月 v1.1	<p>Quartus II ソフトウェア・バージョン 6.1.0 のための更新</p> <ul style="list-style-type: none"> <li>● Quartus II ソフトウェア・バージョン 6.1.0 で新たに利用できるようになった Tcl コマンドラインの実行ファイルに関する情報、および HardCopy II デザインのタイミング解析で使用するための推奨事項が追加されました。</li> <li>● 図 6-1 を更新。</li> <li>● 表 6-1、表 6-2、および表 6-3 を更新。</li> <li>● 変更履歴を追加。</li> </ul>	Quartus II ソフトウェア・バージョン 6.1 リリースの変更に対応した章の中規模なアップデート。
2006 年 3 月	15 章でリリース、内容の変更なし。	
2005 年 10 月 v 1.0	「HardCopy II デバイスのためのスクリプト・ベースのデザイン」の初版。	

