インテル® MAX® 10 FPGA コンフィグレーション ユーザーガイド

この翻訳版は参照用であり、翻訳版と英語版の内容に相違がある場合は、英語版が優先されるものとします。翻訳版は、資料によっては英語版の更新に対応していない場合があります。最新情報につきましては、必ず英語版の最新資料をご確認ください。

オンラインバージョン
フィードバック
UG-M10CONFIG
ID: 683865
バージョン: 2019.01.07

目次

1. インテル® MAX® 10 FPGA コンフィグレーションの概要

インテル® MAX® 10 CRAM (Configuration RAM) は、以下のコンフィグレーション・スキームを使用したコンフィグレーションが可能です。

- JTAG インターフェイスを使用する JTAG コンフィグレーション
- 内部フラッシュを使用する内部コンフィグレーション

サポートされているコンフィグレーション機能

表 1. インテル MAX 10 デバイスがサポートするコンフィグレーション・スキームとその機能

コンフィグレーション・スキーム	リモート・システム・アップグレード	圧縮	デザイン・セキュリティー	SEU の緩和
JTAG コンフィグレーション	—	—	—	あり
内部コンフィグレーション	あり	あり	あり	あり

関連する IP コア

- アルテラ・デュアル・コンフィグレーション インテル FPGA IP — リモート・システム・アップグレード機能で使用
- ユニークチップ ID インテル FPGA IP — インテル MAX 10 デバイスのチップ ID を取得

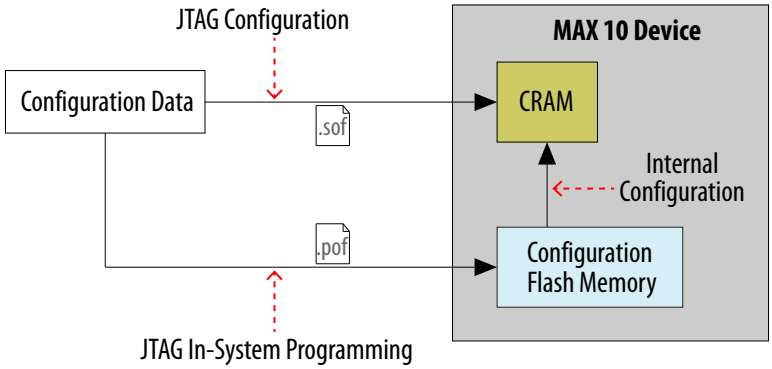
関連情報

- インテル MAX 10 FPGA のコンフィグレーション・スキームとその機能 (5 ページ)
  コンフィグレーション・スキームとその機能について情報を提供します。
- インテル MAX 10FPGA コンフィグレーション・デザインのガイドライン (32 ページ)
  コンフィグレーション・スキームとその機能の使用方法について情報を提供します。
- Unique Chip ID インテル FPGA IP コア (21 ページ)
- デュアル・コンフィグレーション インテル FPGA IP コア (19 ページ)

2. インテル MAX 10 FPGA のコンフィグレーション・スキームとその機能

2.1. コンフィグレーション・スキーム

図 -1: インテル MAX 10 デバイスの JTAG コンフィグレーションおよび内部コンフィグレーションの上位レベルの概要



2.1.1. JTAG コンフィグレーション

インテル MAX 10 デバイスでは、JTAG 命令は内部コンフィグレーション・スキームよりも優先されます。

JTAG コンフィグレーション・スキームを使用して、JTAG インターフェイスの TDI、TDO、TMS および TCK ピンを介して直接的にデバイスの CRAM をコンフィグレーションすることができます。インテル Quartus® Prime 開発ソフトウェアは SRAM オブジェクト・ファイル (**.sof**) を自動的に生成します。**.sof** は、ダウンロード・ケーブルと インテル Quartus Prime 開発ソフトウェア Programmer を使用してプログラミングが可能です。

関連情報

JTAG コンフィグレーションによるインテル MAX 10 デバイスのコンフィグレーション (33 ページ)
インテル Quartus Prime 開発ソフトウェア Programmer とダウンロード・ケーブルを使用した JTAG コンフィグレーションについて詳しい情報を提供します。

2.1.1.1. JTAG ピン

表 2. JTAG ピン

ピン	機能	説明
TDI	以下に用いるシリアル入力ピン	• TDI は TCK の立ち上がりエッジでサンプリングされます • TDI ピンは内部ウィークプルアップ抵抗を備えています

continued...

ピン	機能	説明
	• 命令 • バウンダリー・スキャン・テスト (BST) データ • プログラミング・データ	
TDO	以下に用いるシリアル出力ピン • 命令 • バウンダリー・スキャン・テスト・データ • プログラミング・データ	• TDO は TCK の立ち下がりエッジでサンプリングされます • このピンは、データがデバイスからシフトアウトされない場合、トライステートになります
TMS	TAP コントローラー・ステート・マシンの遷移を決定するコントロール信号を提供する入力ピンです	• TMS は TCK の立ち上がりエッジでサンプリングされます • TMS ピンは内部ウィークプルアップ抵抗を備えています
TCK	BST 回路へのクロック入力	—

すべての JTAG ピンは、V_{CCIO} 1B によって駆動されます。JTAG モードでは、I/O ピンは LVTTL あるいは LVCMOS の 3.3 V から 1.5 V の規格をサポートします。

関連情報

- インテル® MAX® 10 デバイス・データシート
  MAX 10 デバイスでサポートされている I/O Standard について詳しい情報を提供します。
- ガイドライン: 兼用コンフィグレーション・ピン (32 ページ)
- 兼用ピンのイネーブル (33 ページ)

2.1.2. 内部コンフィグレーション

内部コンフィグレーションを行う前に、コンフィグレーション・データをコンフィグレーション・フラッシュメモリー (CFM) にプログラミングしておく必要があります。CFM に書き込まれるコンフィグレーション・データは、プログラム・オブジェクト・ファイル (.pof) の一部になります。JTAG インシステム・プログラミング (ISP) を使用して、.pof を内部フラッシュにプログラミングします。

内部コンフィグレーション時には、インテル MAX 10 デバイスは CFM からのコンフィグレーション・データを CRAM にロードします。

2.1.2.1. 内部コンフィグレーション・モード

表 3. インテル MAX 10 の機能オプションによってサポートされる内部コンフィグレーション・モード

インテル MAX 10 の機能オプション	サポートされる内部コンフィグレーション・モード
コンパクト	• Single Compressed Image • Single Uncompressed Image
フラッシュおよびアナログ	• Dual Compressed Images • Single Compressed Image • Single Compressed Image with Memory Initialization • Single Uncompressed Image • Single Uncompressed Image with Memory Initialization

注意: Dual Compressed Images モードでは、CONFIG_SEL ピンを使用してコンフィグレーション・イメージを選択します。

関連情報

- 内部コンフィグレーションを使用するインテル MAX 10 デバイスのコンフィグレーション (37 ページ)
- リモート・システム・アップグレード (12 ページ)

2.1.2.2. コンフィグレーション・フラッシュ・メモリー

CFM は、コンフィグレーション・イメージを格納するために使用する不揮発性の内部フラッシュです。CFM は、圧縮されたコンフィグレーション・イメージを、圧縮率とインテル MAX 10 デバイスに応じて最大で 2 つまで格納することができます。デバイスで 2 つのコンフィグレーション・イメージを格納するには、圧縮率を少なくとも 30%にする必要があります。

関連情報

コンフィグレーション・フラッシュメモリーへのアクセス許可 (23 ページ)

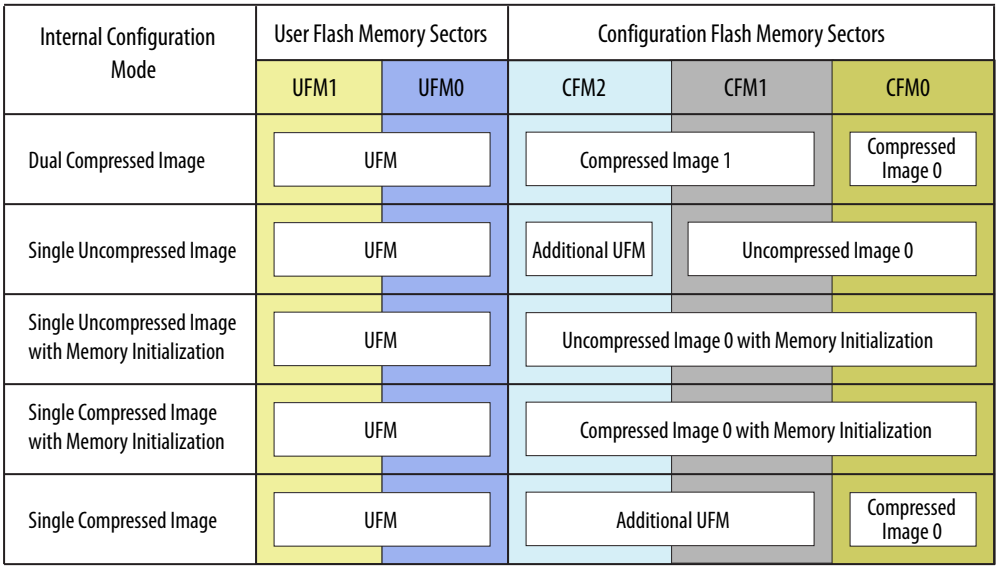
2.1.2.2.1. コンフィグレーション・フラッシュメモリーのセクター

10M02 を除くインテル MAX 10 デバイスの CFM はすべて、CFM0、CFM1、CFM2 の 3 つのセクターで構成されています。セクターは、選択した内部コンフィグレーション・モードに応じて異なる方法でプログラミングされます。

インテル MAX 10 デバイスは CFM0 のみで構成されています。シングル圧縮イメージまたはシングル非圧縮イメージを選択すると、インテル MAX 10 デバイスの CFM0 セクターは同様の方法でプログラミングされます。

図 -2: アナログ機能オプションおよびフラッシュ機能オプションを持つすべてのインテル MAX 10 デバイスのコンフィグレーション・フラッシュメモリー・セクターの使用方法

未使用の CFM1 セクターと CFM2 セクターは、ユーザー・フラッシュメモリー (UFM) として使用可能です。



関連情報

CFM と UFM のアレイサイズ
UFM と CFM のセクターサイズについて詳しい情報を提供します。

2.1.2.2.2. コンフィグレーション・フラッシュメモリーのプログラミング時間

表 4. インテル MAX 10 デバイスでのコンフィグレーション・フラッシュ・メモリーのセクター要するプログラミング時間

注意: プログラミング時間は、システム・オーバーヘッドのみを除外した JTAG インターフェイスのプログラミング時間を意味します。これはユーザーが経験する実際のプログラミング時間を示すものではありません。システム・オーバーヘッドを補償する目的で、インテル MAX 1010M04/08/16/25/40/50 デバイスではデバイス・プログラミング時にフラッシュ・パラレル・モードが活用されるよう インテル Quartus Prime Programmer が強化されています。インテル MAX 10 デバイスはフラッシュ・パラレル・モードをサポートしていないため、他のデバイスと比較してプログラミング時間が長くなります。

デバイス	インシステム・プログラミング時間 (秒)		
	CFM2	CFM1	CFM0
10M02	—	—	5.4
10M04 および 10M08	6.5	4.6	11.1
10M16	12.0	8.9	20.8
10M25	16.4	12.6	29.0
10M40 と 10M50	30.2	22.7	52.9

2.1.2.3. インシステム・プログラミング

インテル MAX 10 デバイスの CFM を含む内部フラッシュを、業界標準の IEEE 1149.1 JTAG インターフェイスを介して ISP でプログラミングすることができます。ISP は、CFM をプログラミング、消去、および検証する機能を提供します。インテル MAX 10 デバイスの JTAG 回路と ISP 命令は、IEEE-1532-2002 のプログラミング仕様に準拠しています。

インテル MAX 10 デバイスは、ISP 実行時に TDI 入力ピンを介して IEEE Std. 1532 命令、アドレス、およびデータを受信します。データは TDO 出力ピンを介してシフトアウトされ、期待されるデータと比較されます。

以下は、ISP 動作の一般的なフローです。

1. ID の確認 — あらゆるプログラミングまたは検証処理の前に JTAG ID が確認されます。JTAG ID 読み出しの所要時間は、全体的なプログラミング時間と比較した場合、比較的短時間となります。
2. ISP の開始 — I/O ピンがユーザーモードから ISP モードにスムーズに遷移するようにします。
3. セクター消去 — デバイス消去のためにアドレスと命令をシフトインし、消去パルスを適用します。
4. プログラミング — アドレス、データ、プログラミング命令をシフトインし、フラッシュセルをプログラミングするためのプログラミング・パルスを生成します。この処理を内部フラッシュセクターの各アドレスごとに繰り返します。
5. 検証 — アドレスをシフトインし、検証命令により読み出しパルスを生成し、比較用のデータをシフトアウトします。この処理を内部フラッシュのアドレスごとに繰り返します。
6. ISP の終了 — I/O ピンが ISP モードからユーザーモードにスムーズに遷移するようにします。

インテル Quartus Prime Programmer を使用して CFM をプログラミングすることもできます。

関連情報

内部フラッシュへの.pof のプログラミング (40 ページ)
インテル Quartus Prime Programmer を使用した.pof のプログラミング手順を提供します。

2.1.2.3.1. ISP クランプ

通常の ISP 動作が開始する際には、すべての I/O ピンはトライステートになります。デバイスの ISP 動作時にデバイスの I/O ピンがトライステートになることが望ましくない状況では、ISP クランプ機能が使用できます。

ISP クランプ機能を使用すると、I/O ピンをトライステート、High、Low、またはサンプリングおよび保持ステートに設定することができます。インテル Quartus Prime 開発ソフトウェアは、この設定に基づいて各 I/O ピンのバウンダリー・スキャン・レジスターにスキャンされる値を決定します。これにより、デバイス・プログラミングが進行中におけるピンがクランプされるステートが決定されます。

I/O ピンをクランプする前に、適切な値をバウンダリー・スキャン・レジスターにロードするために SAMPLE/PRELOAD JTAG 命令が最初に実行されます。適切な値をバウンダリー・スキャン・レジスターにロードした後、EXTEST 命令が実行され、I/O ピンを SAMPLE/PRELOAD 実行時にバウンダリー・スキャン・レジスターにロードされた特定の値にクランプします。

デバイスが ISP クランプモードに入った時にピンの既存のステートをサンプリングし、ピンをそのステートに保持するように選択した場合は、信号が安定したステートになることを確認しなければなりません。サンプル・セットアップ・タイムは、ダウンロード・ケーブルやソフトウェアだけでなく、TCK 周波数にも依存しており、制御不可能であるため、安定したステート信号が必要です。トグルする信号、または長期間にわたってスタティックでない信号をサンプリングした場合、キャプチャーした値は正確ではない場合があります。

関連情報

インテル Quartus Prime 開発ソフトウェアによる ISP クランプの実装 (41 ページ)

2.1.2.3.2. リアルタイム ISP

新しいデザインイメージで内部フラッシュを更新するための通常の ISP 動作では、デバイスはユーザーモードを終了し、すべての I/O ピンはトライステートになります。デバイスが新しいデザインイメージのプログラミングを完了すると、リセットし、ユーザーモードに入ります。

リアルタイム ISP 機能は、ユーザーモードで動作しながら、新しいデザインイメージで内部フラッシュを更新します。内部フラッシュの更新中は、デバイスは既存のデザインで動作を継続します。新しいデザインイメージのプログラミング処理が完了しても、デバイスはリセットされません。新しいデザインイメージによる更新は次のコンフィグレーション・サイクルから有効になります。

2.1.2.3.3. ISP 命令とリアルタイム ISP 命令

表 5. インテル MAX 10 の ISP 命令とリアルタイム ISP 命令

命令	命令コード	概要
CONFIG_IO	00 0000 1101	• JTAG テストに向けて、IOCSR を使用した JTAG ポート経由での I/O リコンフィグレーションを可能にします。これはコンフィグレーション中、もしくは終了後に実行されます。 • CONFIG_IO 命令を発行する前に、nSTATUS ピンが High になる必要があります。
PULSE_NCONFIG	00 0000 0001	物理ピンが影響されていない状態でも、nCONFIG ピンを Low にパルスすることでリコンフィグレーションのトリガーをエミュレーションします。

continued...

命令	命令コード	概要
ISC_ENABLE_HIZ (1)	10 1100 1100	• デバイスを ISP モードにし、すべての I/O ピンをトライステートにし、すべてのコアドライバー、ロジック、およびレジスターを駆動します。 • ISC_DISABLE 命令のロードおよび更新が完了するまで、デバイスは ISP モードを維持します。 • ISC_ENABLE 命令は必須の命令です。この要件は ISC_ENABLE_CLAMP 命令または ISC_ENABLE_HIZ 命令によって満たされます。
ISC_ENABLE_CLAMP (1)	10 0011 0011	• デバイスを ISP モードにし、すべての I/O ピンを強制的に JTAG バウンダリー・スキャン・レジスターの内容に従わせます。 • この命令がアクティブの場合、コアのドライバー、ロジック、およびレジスターのすべてが凍結されます。I/O ピンは、デバイスが正常に ISP モードを終了するまでクランプされたままになります。
ISC_DISABLE	10 0000 0001	• デバイスの ISP モードを終了させます。 • ISC_DISABLE 命令は、テスト動作状態またはアイドル状態で 200 µs 待機した直後に正しく完了します。
ISC_PROGRAM (2)	10 1111 0100	デバイスをイン・システム・プログラミングにセットします。プログラミングは、テスト動作状態またはアイドル状態で生じます。
ISC_NOOP (2)	10 0001 0000	• ISP モードを終了することなくデバイスを非動作モードにセットし、ISC_Default レジスターをターゲットにします。 • 以下の場合に使用します。 — 2 つ以上の ISP 準拠デバイスが ISP モードでアクセスされている場合 — かつ、デバイスのサブセットが何らかの命令を処理する間に、他のより複雑なデバイスが与えられた処理の付加的な手順を完了している場合
ISC_ADDRESS_SHIFT (2)	10 0000 0011	デバイスをフラッシュ・アドレスのロード向けに設定します。これは、フラッシュ・アドレス・レジスターである ISC_Address をターゲットにします。
ISC_ERASE (2)	10 1111 0010	• 内部フラッシュを消去するようにデバイスをセットします。 • ISC_ADDRESS_SHIFT 命令の後に発行します。
ISC_READ (2)	10 0000 0101	• 通常のユーザーバイアス状態で内部フラッシュを検証するためにデバイスを設定します。 • ISC_READ 命令は、バーストモードとして知られている、明確なアドレス指定と自動的なインクリメントをサポートします。
BGP_ENABLE	01 1001 1001	• デバイスをリアルタイム ISP モードに設定します。 • デバイスのユーザーモードを維持しつつ、内部フラッシュ・コンフィグレーション・セクターへアクセスすることを可能にします。
BGP_DISABLE	01 0110 0110	• デバイスのリアルタイム ISP モードを終了させます。 • リコンフィグレーションによりさえぎられると、デバイスは BGP_DISABLE 命令を使用してリアルタイム ISP から抜け出る必要があります。

注意: サポートされていない JTAG 命令は使用しないでください。デバイスを未知の状態にし、操作を回復するには、電源の再投入が必要になる恐れがあります。。

2.1.2.4. 初期化コンフィグレーション・ビット

初期化コンフィグレーション・ビット (ICB) はインテル MAX 10 デバイスのコンフィグレーション機能の設定を格納しています。ICB 設定は **Convert Programming File** ツールで設定可能です。

(1) ISC_ENABLE_HIZ 命令と ISC_ENABLE_CLAMP 命令はコアロジックから発行しないでください。

(2) ENABLE 命令と DISABLE 命令を除いて、すべての ISP 命令とリアルタイム ISP 命令は、デバイスが ISP モードまたはリアルタイム ISP モードでなければ、無効となります。

表 6. インテル MAX 10 デバイス向け ICB 値およびその概要

コンフィグレーション設定	説明	デフォルト状態と値
Set I/O to weak pull-up prior usermode	• 有効: デバイス・コンフィグレーション時に I/O をウィークプルアップに設定します • 無効: I/O をトライステートにします	有効
Configure device from CFM0 only.	有効: • CONFIG_SEL ピン設定がディスエーブルされます。 • デバイスは自動的にイメージ 0 をロードします。 • イメージ 0 で不具合が生じた場合にデバイスはイメージ 1 をロードしません。 無効: • 最初のイメージで不具合が生じた場合にデバイスが自動的に Secondary イメージをロードします。	無効
Use secondary image ISP data as default setting when available.	POF に含めるための ISP データを、Initial イメージまたは Secondary イメージから選択します。 • 無効: Initial イメージからの ISP データを使用します • 有効: Secondary イメージからの ISP データを使用します ISP データは、ISP 時のピン状態の情報を含んでいます。これはウィークプルアップを使用するトライステートにもでき、I/O の状態をクランプすることもできます。ISP クランプは、**Device and Pin Option** または **Pin Assignment** ツールから設定できます。	無効
Verify Protect	Verify Protect 機能を有効または無効にする	無効
Allow encrypted POF only	有効にすると、暗号化されていない .pof が使用されている場合、コンフィグレーション・エラーが生じます。	無効
JTAG Secure(3)	JTAG セキュア機能を有効または無効にします。	無効
Enable Watchdog	リモート・システム・アップグレード向けにウォッチドッグ・タイマーを有効または無効にします。	有効
Watchdog value	リモート・システム・アップグレード向けにウォッチドッグ・タイマーの値を設定します。	0x1FFF (4)

関連情報

- .pof と ICB 設定 (38 ページ)
- Verify Protect (22 ページ)
- JTAG セキュアモード (21 ページ)
- ISP 命令とリアルタイム ISP 命令 (9 ページ)
- ユーザー・ウォッチドッグ・タイマー (18 ページ)
- Convert Programming Files を使用した.pof の生成 (39 ページ)
  Convert Programming File を使用して.pof を生成する際の ICB 設定について、詳しい情報を提供します。

(3) JTAG セキュア機能は、インテル Quartus Prime 開発ソフトウェアではデフォルトでは無効になっています。このオプションを可視化するには、Convert Programming Files を使用した.pof の生成 (39 ページ)を参照してください。

(4) ウォッチドッグ・タイマーの値は使用するインテル MAX 10 デバイスによって異なります。詳しくはユーザー・ウォッチドッグ・タイマーの項を参照してください。

2.1.2.5. 内部コンフィグレーション時間

内部コンフィグレーション時間の測定は、nSTATUS 信号の立ち上がりエッジから CONF_DONE 信号の立ち上がりエッジまでになります。

表 7. インテル MAX 10 デバイスの内部コンフィグレーション時間 (非圧縮.rbf)

デバイス	内部コンフィグレーション時間 (ms)							
	暗号化なし				暗号化あり			
	メモリー初期化なし		メモリー初期化あり		メモリー初期化なし		メモリー初期化あり	
	最小値	最大値	最小値	最大値	最小値	最大値	最小値	最大値
10M02	0.3	1.7	—	—	1.7	5.4	—	—
10M04	0.6	2.7	1.0	3.4	5.0	15.0	6.8	19.6
10M08	0.6	2.7	1.0	3.4	5.0	15.0	6.8	19.6
10M16	1.1	3.7	1.4	4.5	9.3	25.3	11.7	31.5
10M25	1.0	3.7	1.3	4.4	14.0	38.1	16.9	45.7
10M40	2.6	6.9	3.2	9.8	41.5	112.1	51.7	139.6
10M50	2.6	6.9	3.2	9.8	41.5	112.1	51.7	139.6

表 8. インテル MAX 10 デバイスの内部コンフィグレーション時間 (圧縮.rbf)

圧縮率はデザインの複雑度合いよって異なります。最小値はベストケース (オリジナルの .rbf サイズの 25%) に基づいており、最大値は標準ケース (オリジナルの .rbf サイズの 70%) に基づいています。

デバイス	内部コンフィグレーション時間 (ms)			
	暗号化なし/暗号化あり			
	メモリー初期化なし		メモリー初期化あり	
	最小値	最大値	最小値	最大値
10M02	0.3	5.2	—	—
10M04	0.6	10.7	1.0	13.9
10M08	0.6	10.7	1.0	13.9
10M16	1.1	17.9	1.4	22.3
10M25	1.1	26.9	1.4	32.2
10M40	2.6	66.1	3.2	82.2
10M50	2.6	66.1	3.2	82.2

2.2. コンフィグレーション機能

2.2.1. リモート・システム・アップグレード

インテル MAX 10 デバイスは、リモート・システム・アップグレード機能をサポートしています。デュアル圧縮イメージでの内部コンフィグレーション・モードを選択した際には、リモート・システム・アップグレード機能はデフォルトで有効になります。

インテル MAX 10 デバイスのリモート・システム・アップグレード機能は、以下の機能を提供します。

- リモート・コンフィグレーションの制御
- エラー検出、エラーリカバリー、およびエラー情報の提供
- ダイレクト・アプリケーション・コンフィグレーション・イメージのサポート
- 圧縮され、暗号化された .pof のサポート

インテル MAX 10 デバイスでリモート・システム・アップグレードにアクセスするには、以下の 2 通りの方法があります。

- デュアル・コンフィグレーション インテル FPGA IP コア
- ユーザー・インターフェイス

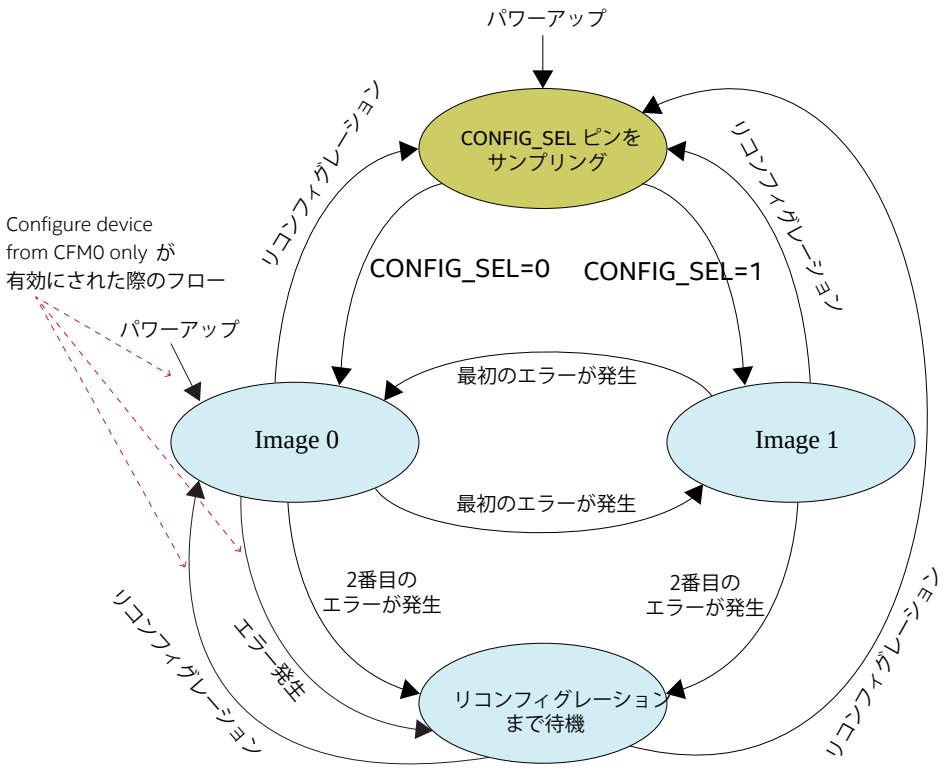
関連情報

- デュアル・コンフィグレーション インテル FPGA IP コア (19 ページ)
- ユーザーロジックを介したリモート・システム・アップグレードへのアクセス (42 ページ)
- AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor
  インテル® MAX® 10 FPGA デバイスのリモート・システム・アップグレード向けリファレンス・デザインを提供します。
- I2C Remote System Update Example
  この例では、I2C プロトコルの使用によるリモート・システム・アップグレードを示します。

2.2.1.1. リモート・システム・アップグレードのフロー

アプリケーション・コンフィグレーション・イメージであるイメージ 0 とイメージ 1 は、どちらも CFM に格納されます。インテル MAX 10 デバイスは、CFM からどちらか 1 つのアプリケーション・コンフィグレーション・イメージをロードします。

図 -3: インテル MAX 10 デバイスのリモート・システム・アップグレードのフロー

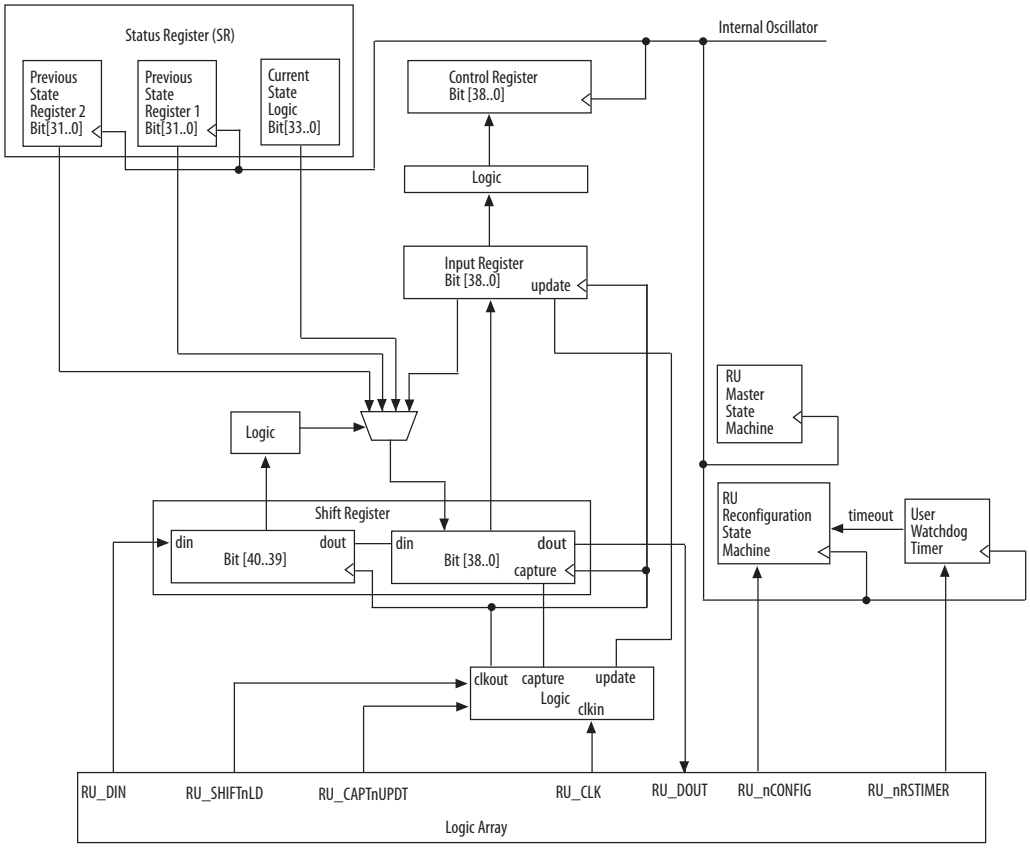


リモート・システム・アップグレード機能は、以下のシーケンスでエラーを検出します。

1. 電源投入後、デバイスは CONFIG_SEL ピンをサンプリングして、ロードするアプリケーション・コンフィグレーション・イメージを決定します。後に続くリコンフィグレーションに向けて、リモート・システム・アップグレード回路の入力レジスターによって CONFIG_SEL ピンの設定を上書きすることができます。
2. エラーが生じると、リモート・システム・アップグレード機能はもう一方のアプリケーション・コンフィグレーション・イメージをロードすることにより回復します。以下に示すエラーにより、リモート・システム・アップグレード機能はもう一方のアプリケーション・コンフィグレーション・イメージをロードします。
   - 内部 CRC エラー
   - ユーザー・ウォッチドッグ・タイマーのタイムアウト
3. 回復のコンフィグレーションが完了してデバイスがユーザーモードになると、リモート・システム・アップグレード回路を使用して、エラーの原因および障害が発生したアプリケーション・イメージを確認することができます。
4. 2 番目のエラーが発生した場合、デバイスはリコンフィグレーション・ソースを待ちます。**Auto-restart configuration after error** が有効になっていれば、デバイスはリコンフィグレーション・ソースを待つことなくリコンフィグレーションします。
5. リコンフィグレーションは、以下の操作によってトリガーされます。
   - 外部から nSTATUS を Low に駆動する
   - 外部から nCONFIG を Low に駆動する
   - RU_nCONFIG を Low に駆動する

2.2.1.2. リモート・システム・アップグレード回路

図 -4: リモート・システム・アップグレード回路



リモート・システム・アップグレード回路は、以下のように機能します。

- コンフィグレーションの現在の状態をトラッキングします
- すべてのリコンフィグレーション・ソースをモニタリングします
- アプリケーション・コンフィグレーション・イメージの設定にアクセスを提供します
- エラーが発生した場合に、デバイスをフォールバック・コンフィグレーションに戻します
- 失敗したアプリケーション・コンフィグレーション・イメージの情報へのアクセスを提供します

2.2.1.2.1. リモート・システム・アップグレード回路の信号

表 9. インテル MAX 10 デバイスのリモート・システム・アップグレード回路の信号

コア信号名	ロジック信号名	入力/出力	説明
RU_DIN	regin	入力	この信号は、RU_CLK の立ち上がりエッジのシフトレジスターにデータを書き込むために使用します。シフトレジスターへデータをロードするには、RU_SHIFTnLD をアサートします。
RU_DOUT	regout	出力	この信号は、シフトレジスターから出力データを取得するために使用します。RU_SHIFTnLD がアサートされている場合、RU_CLK の各立ち上がりエッジでデータが出力されます。

continued...

コア信号名	ロジック信号名	入力/出力	説明
RU_nRSTIMER	rsttimer	入力	• この信号は、ユーザー・ウォッチドッグ・タイマーをリセットするために使用します。この信号の立ち下がりエッジでユーザー・ウォッチドッグ・タイマーのリセットがトリガーされます。 • タイマーをリセットするには、RU_nRSTIMER 信号を少なくとも 250 ns 間パルスします。
RU_nCONFIG	rconfig	入力	この信号は、デバイスをリコンフィグレーションする目的で使用します。リモート・システム・アップグレード機能を有効にしている場合、この信号を Low に駆動するとデバイスのリコンフィグレーションがトリガーされます。
RU_CLK	clk	入力	リモート・システム・アップグレード回路のクロックです。リモート・システム・アップグレード機能を有効にしている場合、このクロックドメイン内のすべてのレジスターはユーザーモードで有効にされます。シフトレジスターと入力レジスターはポジティブ・エッジ・フリップフロップです。
RU_SHIFTnLD	shiftnld	入力	リモート・システム・アップグレード回路のモードを決定する信号を制御します。 • RU_SHIFTnLD が Low に駆動され、RU_CAPTnUPDT が Low に駆動されると、入力レジスターに RU_CLK の立ち上がりエッジでシフトレジスターの内容がロードされます。 • RU_SHIFTnLD が Low に駆動され、RU_CAPTnUPDT が High に駆動されると、シフトレジスターは RU_CLK の立ち上がりエッジで input_cs_ps モジュールから値をキャプチャします。 • RU_SHIFTnLD が High に駆動されると RU_CAPTnUPDT は無視され、シフトレジスターは RU_CLK の立ち上がりエッジごとにデータをシフトします。
RU_CAPTnUPDT	captnupdt	入力	

関連情報

インテル® MAX® 10 デバイス・データシート
リモート・システム・アップグレードのタイミング仕様について詳しい情報を提供します。

2.2.1.2.2. リモート・システム・アップグレード回路の入力制御

リモート・システム・アップグレード回路には 3 つの動作モードがあります。

- アップデート — 入力レジスターにシフトレジスターの値をロードします
- キャプチャ — シフトアウトされるデータをシフトレジスターにロードします
- シフト — ユーザーロジックにデータをシフトアウトします

表 10. リモート・システム・アップグレード回路へのコントロール入力

リモート・システム・アップグレード回路のコントロール入力				動作モード	レジスターの入力設定	
RU_SHIFTnLD	RU_CAPTnUPDT	シフト・レジスター [40]	シフト・レジスター [39]		シフト・レジスター [38:0]	入力レジスター [38:0]
0	0	Don't Care	Don't Care	アップデート	シフト・レジスター [38:0]	シフト・レジスター [38:0]
0	1	0	0	キャプチャ	現在のステート	入力レジスター [38:0]
0	1	0	1	キャプチャ	{8'b0、前回のステート・アプリケーション 1}	入力レジスター [38:0]

continued...

リモート・システム・アップグレード回路のコントロール入力				動作モード	レジスターの入力設定	
RU_SHIFTnLD	RU_CAPTnUPDT	シフト・レジスター [40]	シフト・レジスター [39]		シフト・レジスター [38:0]	入力レジスター [38:0]
0	1	1	0	キャプチャ	{8'b0、前回のステート・アプリケーション 2}	入力レジスター [38:0]
0	1	1	1	キャプチャ	入力レジスター [38:0]	入力レジスター [38:0]
1	Don't Care	Don't Care	Don't Care	シフト	{ru_din、シフトレジスター [38:1]}	入力レジスター [38:0]

以下に、リモート・システム・アップグレード回路のコントロール入力の駆動例を示します。

- RU_SHIFTnLD を High に駆動し 1'b1 にすると、シフトレジスターは RU_CLK の各立ち上がりエッジでデータをシフトし、RU_CAPTnUPDT は機能を持ちません。
- RU_SHIFTnLD と RU_CAPTnUPDT の両方を Low に駆動し 1'b0 にすると、入力レジスターに RU_CLK の立ち上がりエッジでシフトレジスターの内容がロードされます。
- RU_SHIFTnLD を Low に駆動し 1'b0 にし、RU_CAPTnUPDT を High に駆動して 1'b1 にすると、シフトレジスターは RU_DCLK の立ち上がりエッジで値をキャプチャします。

2.2.1.2.3. リモート・システム・アップグレードの入力レジスター

表 11. インテル MAX 10 デバイスのリモート・システム・アップグレードの入力レジスター

ビット	名称	説明
RCLK[38..14]	Reserved	予約 — 0 に設定します
13	ru_config_sel	• 0: コンフィグレーション・イメージ 0 をロードします • 1: コンフィグレーション・イメージ 1 をロードします このビットは ru_config_sel_overwrite が 1 にセットされている場合にのみ有効です。
12	ru_config_sel_overwrite	• 0: CONFIG_SEL ピンの上書きをディスエーブルします • 1: CONFIG_SEL ピンの上書きをイネーブルします
RCLK[11..0]	Reserved	予約 — 0 に設定

2.2.1.2.4. リモート・システム・アップグレード・ステータス・レジスター

表 12. リモート・システム・アップグレード・ステータス・レジスター — インテル MAX 10 デバイスの現在のステート・ロジック・ビット

ビット	名称	説明
33:30	msm_cs	マスター・ステート・マシン (MSM) の現在の状態
29	ru_wd_en	イネーブルされたユーザー・ウォッチドッグ・タイマーの現在の状態です。デフォルトステートはアクティブ High です。
28:0	wd_timeout_value	全 29 ビットのウォッチドッグ・タイムアウトの現在の値です。

表 13. リモート・システム・アップグレード・ステータス・レジスター — インテル MAX 10 デバイスの前回のステートビット

ビット	名称	説明
31	nCONFIG	インテル MAX 10 デバイスが以前のアプリケーション・コンフィグレーションを終了させる原因となったリコンフィグレーション・ソースを表すアクティブ High フィールドです。同時に生じた場合には、より大きな数のビットが優先されます。たとえば、nconfig と ru_nconfig が同時にトリガーされると、nconfig は ru_nconfig よりも優先されます。
30	crcerror	
29	nstatus	
28	wdtimer	
27:26	Reserved	予約 — 0 に設定
25:22	msm_cs	リコンフィグレーション・イベントが発生した際の MSM の状態です。リコンフィグレーションは、デバイスにこれまでのアプリケーション・コンフィグレーションを中断させます。
21:0	Reserved	予約 — 0 に設定

関連情報

デュアル・コンフィグレーション インテル FPGA IP コアの Avalon-MM アドレスマップ (60 ページ)

2.2.1.2.5. マスター・ステート・マシン

マスター・ステート・マシン (MSM) は現在のコンフィグレーション・モードをトラッキングし、ユーザー・ウォッチドッグ・タイマーを有効にします。

表 14. インテル MAX 10 デバイスでのリモート・システム・アップグレードのマスター・ステート・マシンの現在の状態の説明

msm_cs 値	状態の説明
0010	イメージ 0 がロードされています。
0011	アプリケーション・イメージへの差し戻しが生じた後で、イメージ 1 がロードされています。
0100	イメージ 1 がロードされています。
0101	アプリケーション・イメージへの差し戻しが生じた後で、イメージ 0 がロードされています。

2.2.1.3. ユーザー・ウォッチドッグ・タイマー

ユーザー・ウォッチドッグ・タイマーは、不完全なアプリケーションのコンフィグレーションがデバイスを無制限にストールすることを防止します。デバイスへのアプリケーション・コンフィグレーションのロードが成功した場合、動作エラーを検出する目的でタイマーを使用することができます。

カウンターは 29 ビット幅で、2^{29} の最大カウント値を有します。ユーザー・ウォッチドッグ・タイマーの値を指定する際には、上位側 12 ビットのみを指定します。タイマー設定の精度は 2^{17} サイクルです。サイクル時間は、ユーザー・ウォッチドッグ・タイマーの内部オシレーターの周波数に基づきます。カウンターおよびデバイスの内部オシレーターに基づいて、サイクル時間を 9 ms から 244 s に設定可能です。

図 -5: インテル MAX 10 デバイスのウォッチドッグ・タイマーの計算式

$$\text{Watchdog timer time-out (seconds)} = \frac{\text{Watchdog timer value (decimal)}}{\text{Watchdog timer frequency}}$$

タイマーは、アプリケーション・コンフィグレーションがユーザーモードに入るとすぐにカウントを開始します。リモート・システム・アップグレード回路は、タイマーが設定時間に達するとタイムアウト信号を生成し、ステータスレジスターを更新し、回復コンフィグレーション・イメージのロードをトリガーします。タイマーをリセットし、アプリケーションのコンフィグレーションが有効であることを確認するには、RU_NRSTIMER を少なくとも 250 ns 間パルスします。

ウォッチドッグ・タイマーを有効にすると、この設定はすべてのイメージに適用されます。また、すべてのイメージにタイマーをリセットするためのソフトロジック・コンフィグレーションを含める必要があります。アプリケーション・コンフィグレーションは、コントロール・ブロックのレジスターをリセットします。

関連情報

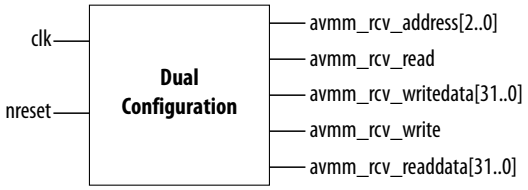
- User Watchdog Internal Circuitry Timing Specifications
  ユーザー・ウォッチドッグの周波数について詳しい情報を提供します。
- 初期化コンフィグレーション・ビット (10 ページ)

2.2.1.4. デュアル・コンフィグレーション インテル FPGA IP コア

デュアル・コンフィグレーション インテル FPGA IP コアは、Avalon-MM インターフェイスを介して次の機能を提供します。

- RU_nCONFIG をアサートしてリコンフィグレーションをトリガーします。
- ウォッチドッグ・タイマーが有効になっている場合に、RU_nRSTIMER をアサートしてウォッチドッグ・タイマーをリセットします。
- リモート・システム・アップグレード回路の入力レジスタにコンフィグレーション設定を書き込みます。
- リモート・システム・アップグレード回路から情報を読み出します。

図 -6: デュアル・コンフィグレーション インテル FPGA IP コアのブロック図



関連情報

- デュアル・コンフィグレーション インテル FPGA IP コアの Avalon-MM アドレスマップ (60 ページ)
- Avalon Interface Specifications
  デュアル・コンフィグレーション・インテル IP コアに用いる Avalon-MM インターフェイスの仕様について詳しい情報を提供します。
- デュアル・コンフィグレーション インテル FPGA IP コアのインスタンス化 (59 ページ)
- デュアル・コンフィグレーション インテル FPGA IP コアのリファレンス (60 ページ)
- リモート・システム・アップグレード (12 ページ)
- AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor
  インテル® MAX® 10 FPGA デバイスのリモート・システム・アップグレード向けリファレンス・デザインを提供します。

- I2C Remote System Update Example
  この例では、I2C プロトコルの使用によるリモート・システム・アップグレードを示します。

2.2.2. コンフィグレーション・デザイン・セキュリティー

インテル MAX 10 のデザイン・セキュリティー機能は、以下の機能をサポートしています。

- 暗号化 — 128 ビットキーの業界標準のデザイン・セキュリティー・アルゴリズムをサポートする高度暗号化規格 (AES) を内蔵しています
- チップ ID — デバイス固有の識別子
- JTAG セキュアモード — JTAG 命令のアクセスを制限します
- Verify Protect — オプションで CFM 内容のリードバックがディスエーブル可能です

2.2.2.1. AES 暗号化保護

インテル MAX 10 のデザイン・セキュリティー機能は、デザインに以下のセキュリティー保護を提供します。

- 複製に対するセキュリティー — 不揮発性キーはインテル MAX 10 デバイス内に安全に保存され、いかなるインターフェイスを介しても読み出しされることはありません。このキーなしで攻撃者が暗号化されたコンフィグレーション・イメージを復号することはできません。
- リバース・エンジニアリングに対するセキュリティー — 暗号化されたコンフィグレーション・ファイルからのリバース・エンジニアリングは、ファイルを復号しなければならないために非常に困難であり、時間を要します。
- 改ざんに対するセキュリティー — JTAG ソースと暗号化された.pof (EPOF) のみを有効にすると、インテル MAX 10 デバイスは同じキーで暗号化されたコンフィグレーション・ファイルのみを受け入れます。さらに、JTAG インターフェイスを介するコンフィグレーションはブロックされます。

関連情報

Convert Programming Files を使用した.pof の生成 (39 ページ)

2.2.2.1.1. 暗号化と復号

MAX 10 は AES での暗号化をサポートしています。プログラミング・ビットストリームはユーザーが指定した暗号化キーに基づいて暗号化されます。インテル MAX 10 デバイスでは、このキーは ICB 設定の一部であり、内部フラッシュに格納されます。したがって、キーは不揮発性ですが、ユーザーはデバイスのチップ全体の消去を行うことにより、キーをクリアーおよび削除できます。

暗号化と同時に圧縮する場合、まずコンフィグレーション・ファイルが圧縮され、次に インテル Quartus Prime 開発ソフトウェアにより暗号化されます。コンフィグレーション時には、デバイスはまずコンフィグレーション・ファイルを復号し、次に解凍します。

ヘッダーならびに I/O コンフィグレーション・シフトレジスター (IOCSR) データは暗号化されません。IOCSR チェーンがプログラミングされた後に復号ブロックがアクティブになります。復号ブロックは、コアデータとポストアンブルのみを復号します。

関連情報

利用可能な JTAG 命令 (22 ページ)

2.2.2.2. Unique Chip ID

Unique Chip ID は、以下の機能を提供します。

- 権限のないデバイスからデザインを保護するためのセキュリティー機能として、デザイン上でデバイスを識別します
- 各インテル MAX 10 デバイスに、書込み保護を備えた不揮発性 64 ビットの固有 ID を提供します

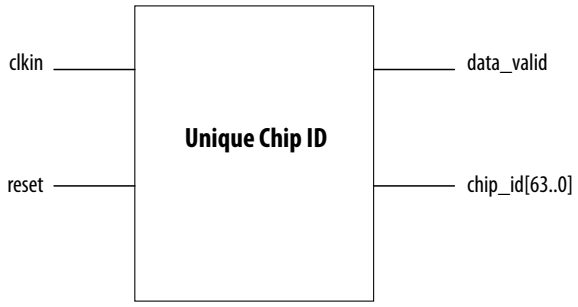
Unique Chip ID インテル FPGA IP コアを使用すれば、インテル MAX 10 デバイスのチップ ID を取得することができます。

関連情報

- Unique Chip ID インテル FPGA IP コア (58 ページ)
- Unique Chip ID インテル FPGA IP コアのポート (63 ページ)

2.2.2.2.1. Unique Chip ID インテル FPGA IP コア

図 -7: Unique Chip インテル FPGA IPIP コアのブロック図



最初の状態では、Unique Chip ID ブロックから読み出されるデータがないので data_valid 信号は Low です。clkin 入力ポートにクロック信号を供給すると、Unique Chip ID インテル FPGA IP コアが Unique Chip ID ブロックを介してデバイスのチップ ID の取得を開始します。デバイスのチップ ID を取得すると、Unique Chip ID インテル FPGA IP コアは data_valid 信号をアサートして、出力ポートにおいてチップ ID 値の読み出しの準備が整ったことを示します。

この動作は、data_valid 信号が Low の際に別のクロック信号を供給した場合にのみ繰り返されます。別のクロック信号を供給した際に data_valid 信号が High であれば、chip_id[63..0]出力がデバイスのチップ ID を保持しているために、動作が停止します。

data_valid 信号が High になるには少なくとも 67 クロックサイクルが必要です。

デバイスをリコンフィグレーションする、または Unique Chip ID インテル FPGA IP コアをリセットするまで、chip_id[63:0]出力ポートはデバイスのチップ ID の値を保持します。

2.2.2.3. JTAG セキュアモード

JTAG セキュアモードでは、必須の IEEE 1149.1 JTAG 命令のみをデバイスで使用できます。

JTAG セキュアは、Convert Programming Files で.pof を生成する際に有効にできます。JTAG セキュアモードを終了するには、UNLOCKJTAG 命令を発行します。LOCKJTAG 命令がデバイスを再び JTAG セキュアモードにします。LOCK と UNLOCK の JTAG 命令は、JTAG コアアクセスによってのみ発行できます。使用可能な命令のリストについては、表 16 (22 ページ)を参照してください。

関連情報

- 利用可能な JTAG 命令 (22 ページ)
- コンフィグレーション・フラッシュメモリーへのアクセス許可 (23 ページ)
- JTAG Secure Design Example
- Convert Programming Files を使用した.pof の生成 (39 ページ)

2.2.2.3.1. JTAG セキュアモードの命令

表 15. インテル MAX 10 デバイスの JTAG セキュアモードの命令

JTAG 命令	命令コード	概要
LOCK	10 0000 0010	• JTAG セキュアモードを有効にします • 外部ピンとコアとの両方からの、JTAG へのアクセスをブロックします
UNLOCK	10 0000 1000	JTAG セキュアモードを無効にします

2.2.2.4. Verify Protect

Verify Protect は、CFM セキュリティーを強化するためのセキュリティー機能です。**Verify Protect** を有効にすると、CFM ではプログラミングと消去の動作のみが可能になります。この機能は、CFM の内容がコピーされることを防ぎます。

インテル Quartus PrimeConvert Programming File ツールで .sof ファイルを .pof ファイルに変換する際に、**Verify Protect** 機能をオンにすることができます。

関連情報

- コンフィグレーション・フラッシュメモリーへのアクセス許可 (23 ページ)
- Convert Programming Files を使用した.pof の生成 (39 ページ)

2.2.2.5. 利用可能な JTAG 命令

表 16. JTAG セキュアモードと暗号化の設定に応じて利用可能な JTAG 命令

JTAG セキュアモード	暗号化	説明
無効	無効	すべての JTAG 命令が有効となります。
	有効	下記以外のすべての JTAG 命令が有効となります。 • CONFIGURE
有効	無効	下記以外のすべての必須ではない IEEE 1149.1 JTAG 命令が無効となります。 • SAMPLE/PRELOAD • BYPASS • EXTEST • IDCODE • UNLOCK • LOCK
	有効	

関連情報

- JTAG セキュアモード (21 ページ)
- インテル MAX 10 JTAG セキュアデザインの例 (52 ページ)
- JTAG Secure Design Example

- 暗号化と復号 (20 ページ)

2.2.2.6. コンフィグレーション・フラッシュメモリーへのアクセス許可

JTAG セキュアモードと Verify Protect 機能により、CFM 操作の許可範囲が決定されます。以下の表に、セキュリティー設定に基づいて許可される操作をリストします。

表 17. インテル MAX 10 デバイスの CFM 許可

動作	JTAG セキュアモードが無効		JTAG セキュアモードが有効	
	Verify Protect が無効	Verify Protect が有効	Verify Protect が無効	Verify Protect が有効
コアを介した ISP	不正な操作	不正な操作	不正な操作	不正な操作
JTAG ピンを介した ISP	完全なアクセス	プログラミングと消去のみ	アクセスなし	アクセスなし
コアを介したリアルタイム ISP	完全なアクセス	プログラミングと消去のみ	アクセスなし	アクセスなし
JTAG ピンを介したリアルタイム ISP	完全なアクセス	プログラミングと消去のみ	アクセスなし	アクセスなし
コアを介した UFM インターフェイス(5)	完全なアクセス	完全なアクセス	完全なアクセス	完全なアクセス

関連情報

- JTAG セキュアモード (21 ページ)
- インテル MAX 10 JTAG セキュアデザインの例 (52 ページ)
- JTAG Secure Design Example
- Verify Protect (22 ページ)
- Convert Programming Files を使用した.pof の生成 (39 ページ)

2.2.3. SEU の緩和とコンフィグレーション・エラーの検出

インテル MAX 10 デバイス内蔵の専用回路は、エラー検出巡回冗長検査 (EDCRC) 機能で構成されています。この機能は、SEU (Single Event Upset) やソフトエラーの緩和を目的として使用します。

ハード化されたオンチップ EDCRC 回路により、デバイスのフィッティングに影響を与えることなく以下の動作を行うことができます。

- コンフィグレーション中に巡回冗長検査 (CRC) エラーを自動検出します
- ユーザーモードでオプションの CRC エラー検出により SEU を特定します
- JTAG インターフェイスを介したエラー検出検証によりエラー検出をテストします

関連情報

- エラー検出機能の検証 (43 ページ)
- エラー検出の有効化 (45 ページ)
- ユーザーロジックによるエラー検出ブロックへのアクセス (45 ページ)

(5) デュアル圧縮イメージモードを選択した場合に、コアを介して UFM のインターフェイスが有効になります。

2.2.3.1. コンフィグレーション・エラーの検出

コンフィグレーション・モードでは、コンフィグレーション・データにフレームベースの CRC が格納されており、各データフレームの CRC 値を含んでいます。

インテル MAX 10 デバイスは、コンフィグレーション中に受信したデータのフレームに基づいて CRC 値を計算し、それをデータストリームのフレームの CRC 値と比較します。コンフィグレーションは、デバイスがエラーを検出するか、すべての値が計算されるまで継続します。

インテル MAX 10 デバイスでは、CRC は インテル Quartus Prime 開発ソフトウェアで計算され、コンフィグレーション・ビット・ストリームの一部としてデバイスにダウンロードされます。デバイスは、コンフィグレーション・モードの終了時に CRC を 32 ビットのストレージレジスターに格納します。

2.2.3.2. ユーザーモード・エラーの検出

SEU は、イオン化した粒子に起因する CRAM ビット状態の変化です。インテル MAX 10 デバイスはエラー検出回路を内蔵しており、CRAM セル内のデータ破損を検出します。

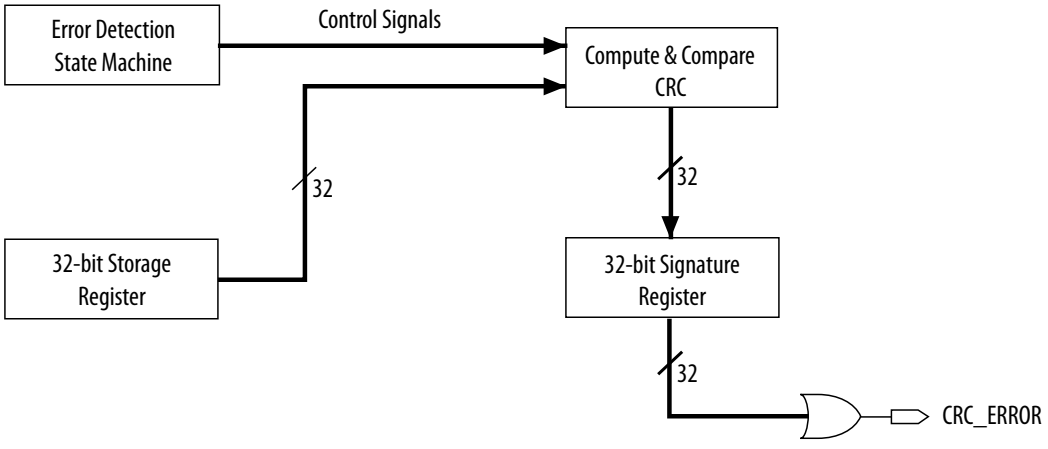
このエラー検出機能は、コンフィグレーション後の CRAM ビットの CRC を連続して計算します。デバイスコンテンツの CRC は、コンフィグレーション終了時に取得した計算済み CRC 値と比較されます。CRC 値が一致すれば、現在のコンフィグレーション CRAM ビットにエラーはありません。このエラー検出プロセスは、nCONFIG を Low に設定することによりデバイスがリセットされるまで継続されます。

インテル MAX 10 デバイスのエラー検出回路では、32 ビットの CRC IEEE Std. 802 と 32 ビットの多項式を CRC ジェネレーターとして使用しています。したがって、デバイスは 32 ビットの CRC 演算を処理します。SEU が発生していなければ、得られる 32 ビットのシグネチャー値は 0x000000 になり、結果として生じる CRC_ERROR の出力信号は 0 になります。デバイスに SEU が生じていれば、得られるシグネチャー値がゼロ以外になり、CRC_ERROR 出力信号は 1 になります。nCONFIG ピンを Low にストローブして FPGA をリコンフィグレーションするか、または、エラーを無視するかを決定する必要があります。

2.2.3.2.1. エラー検出ブロック

図 -8: エラー検出ブロック図

関係する 2 つの 32 ビット・レジスターであるシグネチャー・レジスターとストレージレジスターを含む、エラー検出ブロックのブロック図を示します。



エラー検出回路には、計算された CRC シグネチャーとあらかじめ計算された CRC 値を格納する 2 つの 32 ビット・レジスターのセットがあります。シグネチャー・レジスターがゼロ以外の値になると、CRC_ERROR ピンが High になります。

表 18. インテル MAX 10 デバイスのエラー検出レジスター

レジスター	説明
32 ビット・シグネチャー・レジスター	このレジスターには CRC シグネチャーが含まれています。シグネチャー・レジスターには、あらかじめ計算された CRC 値とユーザーモードで計算された CRC 値とを比較した結果が含まれています。エラーが検出されなければ、シグネチャー・レジスターはすべてゼロになります。シグネチャー・レジスターがゼロ以外であれば、コンフィグレーション CRAM の内容のエラーを示します。CRC_ERROR 信号はこのレジスターの内容に基づきます。
32 ビット・ストレージ・レジスター	このレジスターには、あらかじめ計算された 32 ビットの CRC シグネチャーがコンフィグレーション・ステージの終了時にロードされます。このシグネチャーは、ユーザーモード時に CRC エラーを計算するために 32 ビットの Compute and Compare CRC ブロックにロードされます。このレジスターは、CHANGE_EDREGJTAG 命令の実行時に 32 ビットのスキャンチェーンを形成します。CHANGE_EDREGJTAG 命令で、ストレージレジスターの内容を変更することができます。したがって、命令の実行によりエラーを注入することで、動作中にエラー検出 CRC 回路の機能をインシステムでチェックすることができます。CHANGE_EDREGJTAG 命令を発行しても、デバイスの動作は停止されません。

2.2.3.2.2. CHANGE_EDREG JTAG 命令

表 19. CHANGE_EDREGJTAG 命令の概要

JTAG 命令	命令コード	概要
CHANGE_EDREG	00 0001 0101	この命令は、32 ビットの CRC ストレージレジスターを TDI から TDO まで連結します。任意の計算済み CRC を CRC ストレージレジスターにロードして、エラー検出 CRC 回路の動作を CRC_ERROR ピンでテストします。

2.2.3.3. エラー検出のタイミング

インテル Quartus Prime ソフトウェアを介してエラー検出 CRC 機能が有効にされていれば、コンフィグレーションと初期化が完了してユーザーモードに入った時点でデバイスが自動的に CRC プロセスをアクティブにします。

CRC_ERROR ピンは、エラー検出回路が破損したビットを CRC 演算で検出するまで Low を維持します。ピンが High になると、次の CRC 演算の間ピンは High を維持します。このピンは前の CRC 演算を記録しません。新しい CRC 演算に破損されたビットが含まれていなければ、CRC_ERROR ピンは Low に駆動されます。エラー検出はデバイスがリセットされるまで動作します。

エラー検出回路は、周波数を最大にセットする除数を用いた内部コンフィグレーション・オシレーターによって駆動されます。CRC 演算時間はデバイスとエラー検出クロック周波数に依存します。

関連情報

エラー検出の有効化 (45 ページ)

2.2.3.3.1. エラー検出周波数

インテル Quartus Prime 開発ソフトウェアで分周係数を指定することで、より低いクロック周波数をセットできます。

表 20. インテル MAX 10 デバイスの最小および最大エラー検出周波数

デバイス	エラー検出周波数	最大エラー検出周波数 (MHz)	最小エラー検出周波数 (MHz)	n として有効な値
10M02	55 MHz/2^n から 116 MHz/2^n	58	214.8	1、2、3、4、5、6、7、8
10M04				
10M08				
10M16				
10M25				
10M40	35 MHz/2^n から 77 MHz/2^n	38.5	136.7	
10M50				

2.2.3.3.2. 巡回冗長検査の計算のタイミング

表 21. インテル MAX 10 デバイスでの巡回冗長検査の計算のタイミング

デバイス	除数値 (n = 2)	
	最小時間 (ms)	最大時間 (ms)
10M02	2	6.6
10M04	6	15.7
10M08	6	15.7
10M16	10	25.5
10M25	14	34.7
10M40	43	106.7
10M50	43	106.7

図 -9: CRC 演算式

以下の式を使用して、除数が「2」以外の CRC 演算時間を計算します。

$$CRC\ Calculation\ Time_{\ Divisor\ n} = CRC\ Calculation\ Time_{\ Divisor\ 2} \times \frac{n}{2}$$

例-1: CRC 演算の例

除数値が 256 のインテル MAX 10 デバイスでは、

除数 256 での最小 CRC 演算時間 = 10 x (256/2) = 1280 ms

2.2.3.4. CRC エラーからの回復

インテル MAX 10 デバイス内に含まれるシステムは、デバイス・リコンフィグレーションを制御する必要があります。CRC_ERROR ピンでエラーを検出した後で、システムがインテル MAX 10 デバイスをリコンフィグレーションしても支障のないタイミングで nCONFIG を Low にストローブすることにより、システムがリコンフィグレーションを行うことができます。

デバイスをリコンフィグレーションすることによりデータビットが正しい値に書き換えられると、デバイスは正しく機能します。

SEU はインテル FPGA デバイスでは一般的ではありませんが、特定の高信頼性アプリケーションでは、これらのエラーを考慮したデザインが必要な場合があります。

2.2.4. コンフィグレーション・データの圧縮

インテル MAX 10 デバイスは、圧縮されたコンフィグレーション・ビットストリームを受信し、コンフィグレーション中にリアルタイムでデータを復元することができます。この機能により CFM に格納されるコンフィグレーション・イメージのサイズを削減することができます。圧縮を実行することでデザインよってはコンフィグレーション・ファイル・サイズが少なくとも 30%削減されることがデータにより証明されます。

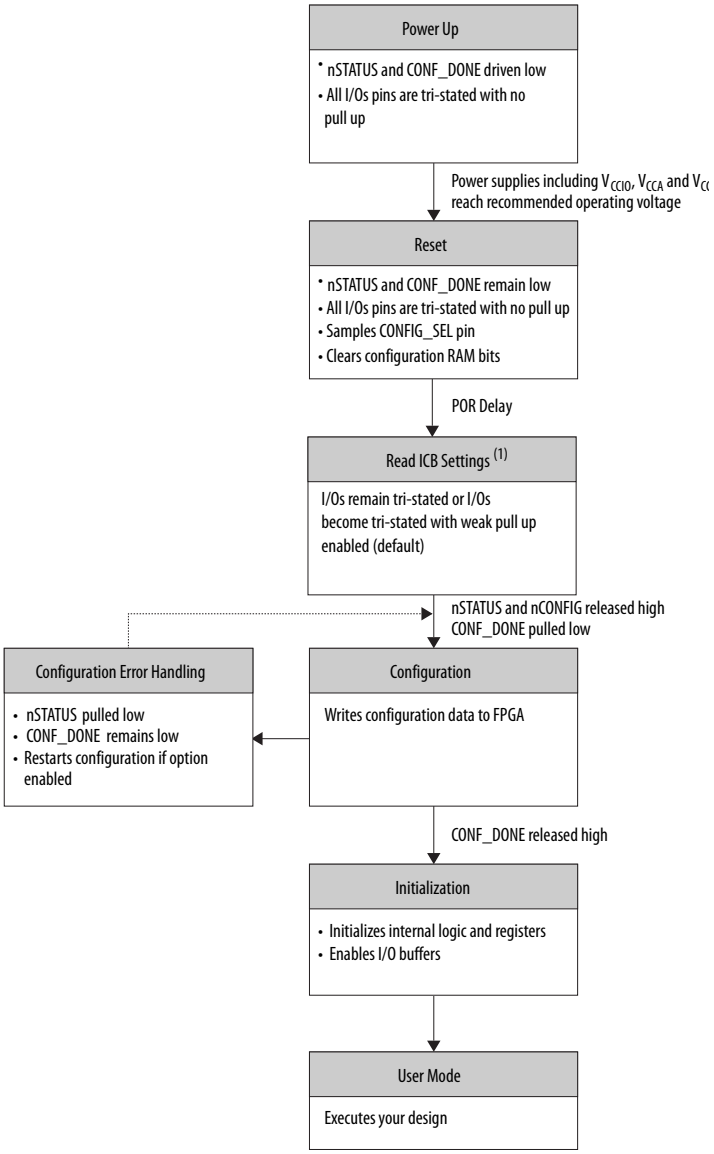
関連情報

- デザインのコンパイルの前に圧縮を有効にする場合 (47 ページ)
- デザインのコンパイル後に圧縮を有効にする場合 (48 ページ)

2.3. コンフィグレーションの詳細

2.3.1. コンフィグレーション・シーケンス

図 -10: インテル MAX 10 デバイスのコンフィグレーション・シーケンス



Note:
1. Before an Intel MAX 10 device has been configured or CFM programmed for the first time (such as a new device or a device that has been erased), the weak pull up resistors will remain off during configuration.

nCONFIG ピンを少なくとも最小 $t_{RU\_nCONFIG}$ Low パルス幅 Low に引き下げることにより、リコンフィグレーションを開始することができます。このピンが Low に引き下げられると、nSTATUS ピンと CONF_DONE ピンは Low に引き下げられ、すべての I/O ピンは ICB 設定に基づいて内部ウィークプルアップに接続またはトライステートにされます。

関連情報

Convert Programming Files を使用した.pof の生成 (39 ページ)
コンフィグレーション時にウィークプルアップをセットする方法について、詳しい情報を提供します。

2.3.1.1. パワーアップ

デバイスをパワーダウン状態からパワーアップする場合、バンク 1B (10M02 デバイスではバンク 1)、バンク 8 およびコアの V_{CCIO} に適切なレベルまで電力を加えれば、POR を終了できます。パワーアップ・ステージから抜け出た後、インテル MAX 10 デバイスは僅かな POR 遅延でコンフィグレーション・ステージに入ります。

関連情報

- インテル® MAX® 10 電源管理ユーザーガイド
  MAX® 10 デバイスの電源モードについて詳しい情報を提供します。
- インテル® MAX® 10 デバイス・データシート
  ランプアップ時間の仕様についての詳しい情報を提供します。
- インテル® MAX® 10 FPGA デバイスファミリーのピン接続ガイドライン
  コンフィグレーション・ピンの接続について詳しい情報を提供します。

2.3.1.1.1. シングル電源およびデュアル電源のインテル MAX 10 デバイスにおける POR でモニタリングされる電圧レール

コンフィグレーションを開始するには、以下の表に示すように必要な電圧を適切な電圧レベルにパワーアップする必要があります。バンク 1B (10M02 デバイスではバンク 1) とバンク 8 の V_{CCIO} は、コンフィグレーション中に 1.5V から 3.3 V の電圧にパワーアップする必要があります。

表 22. シングル電源およびデュアル電源のインテル MAX 10 デバイスでの POR でモニタリングされる電圧レール

電圧をパワーアップする際に従うべきパワーアップ・シーケンスはありません。

デバイスの電源オプション	POR にモニタリングされる電源
シングル電源	安定化された V_{CC_ONE}
	V_{CCA}
	V_{CCIO} バンク 1B [(6)]とバンク 8
デュアル電源	V_{CC}
	V_{CCA}
	V_{CCIO} バンク 1B[(6)]とバンク 8

(6) 10M02 デバイスではバンク 1 です。

2.3.1.1.2. インテル MAX 10 デバイスでモニタリングされる電源ランプ時間要件

図 -11: インテル MAX 10 デバイスでモニタリングされる電源ランプ時間要件図

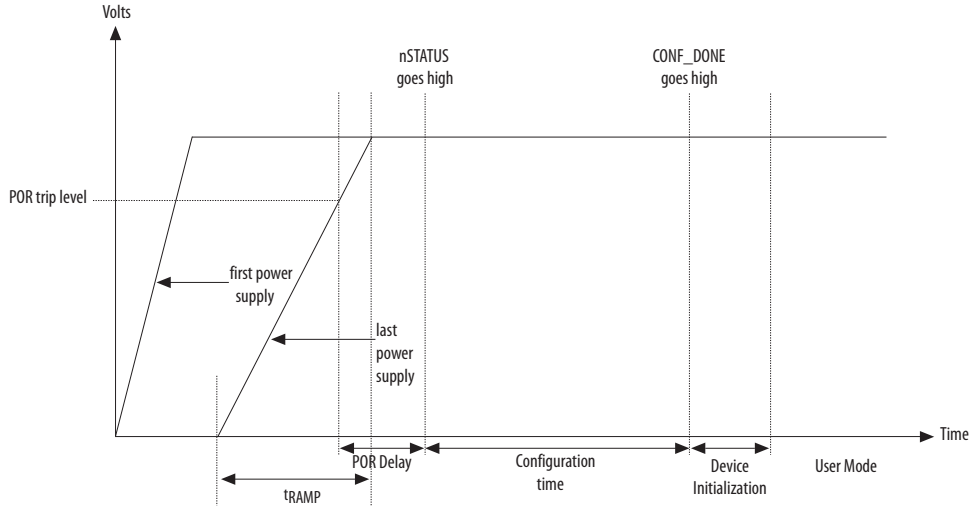


表 23. インテル MAX 10 デバイスでモニタリングされる電源ランプ時間要件

シンボル	パラメーター	最小値	最大値	単位
t_{RAMP}	電源ランプ時間(7)	— (8)	10	ms

2.3.1.2. コンフィグレーション

コンフィグレーション時に、コンフィグレーション・データは、内部フラッシュから読み出され、CRAM に書き込まれます。

2.3.1.3. コンフィグレーション・エラーの処理

自動的にコンフィグレーションを再開するには、インテル Quartus Prime 開発ソフトウェアの **Device and Pin Options** ダイアログボックスの **General** ページで **Auto-restart configuration after error** オプションをオンにします。

このオプションをオンにしない場合には、nSTATUS ピンをモニタリングすることでエラーを検出することができます。コンフィグレーションを再開するには、nCONFIG ピンを少なくとも $t_{RU\_nCONFIG}$ 期間以上 Low に引き下げます。

2.3.1.4. 初期化

CONF_DONE ピンが High になった後に、初期化シーケンスが開始します。初期化クロックソースは内部オシレーターであり、インテル MAX 10 デバイスは、適切な初期化の実行に十分なクロックサイクルを受け取ります。

---

(7) コンフィグレーションが完了する前に、すべての V_{CCIO} 電源が規定電圧に達するように確認してください。内部コンフィグレーション時間 (12 ページ)を参照してください。

(8) ランプレート要件には絶対最小値がありません。インテルは最小 t_{RAMP} を 200 µs と特性評価しました。

2.3.1.5. ユーザーモード

初期化が完了するとデザインが動作を開始します。ユーザー I/O ピンはデザインで指定されたように機能します。

2.3.2. インテル MAX 10 のコンフィグレーション・ピン

インテル MAX 10 デバイスのすべてのコンフィグレーション・ピンと JTAG ピンは兼用ピンです。コンフィグレーション・ピンは、ユーザーモードの前ではコンフィグレーション・ピンとして機能します。デバイスがユーザーモードになると、ピンはユーザー I/O ピンとして機能するか、またはコンフィグレーション・ピンのままにもできます。

表 24. インテル MAX 10 デバイスのコンフィグレーション・ピンのまとめ

すべてのピンは、V_{CCIO} バンク 1B (10M02 デバイスではバンク 1) および 8 によって駆動されます。

コンフィグレーション・ピン	入力/出力	コンフィグレーション・スキーム
CRC_ERROR	出力のみ、オープンドレイン	オプショナル、JTAG および内部コンフィグレーション
CONFIG_SEL	入力のみ	内部コンフィグレーション
DEV_CLRn	入力のみ	オプショナル、JTAG および内部コンフィグレーション
DEV_OE	入力のみ	オプショナル、JTAG および内部コンフィグレーション
CONF_DONE	双方向、オープンドレイン	JTAG および内部コンフィグレーション
nCONFIG	入力のみ	JTAG および内部コンフィグレーション
nSTATUS	双方向、オープンドレイン	JTAG および内部コンフィグレーション
JTAGEN	入力のみ	オプショナル、JTAG コンフィグレーション
TCK	入力のみ	JTAG コンフィグレーション
TDO	出力のみ	JTAG コンフィグレーション
TMS	入力のみ	JTAG コンフィグレーション
TDI	入力のみ	JTAG コンフィグレーション

関連情報

- ガイドライン: 兼用コンフィグレーション・ピン (32 ページ)
- 兼用ピンのイネーブル (33 ページ)

3. インテル MAX 10FPGA コンフィグレーション・デザインのガイドライン

3.1. 兼用コンフィグレーション・ピン

3.1.1. ガイドライン: 兼用コンフィグレーション・ピン

ユーザーモードでコンフィグレーション・ピンをユーザー I/O ピンとして使用するには、以下のガイドラインに従う必要があります。

表 25. インテル MAX 10 デバイスの兼用コンフィグレーション・ピンのガイドライン

ガイドライン	ピン
初期化時のコンフィグレーション・ピン • 外部 I/O ドライバーをトライステートにし、外部ウィーク・プルアップ抵抗をドライブします。または、(9) Intel • 外部 I/O ドライバーを使用して、ピンを外部ウィーク・プルアップ抵抗と同じ状態にドライブします。	• nCONFIG • nSTATUS • CONF_DONE
JTAG ピン • JTAGEN ピンを使用してユーザー I/O ピンと JTAG ピンの機能を交互に切り替えるのであれば、すべての JTAG ピンがシングルエンドの I/O ピンまたは電圧リファレンス形式の I/O ピンとして割り当てられている必要があります。推奨の入力バッファーはシュミットトリガー入力 • JTAG ピンのいずれかを差動 I/O ピンとして割り当てた場合、ユーザーモードで JTAG ピンが JTAG ピンとしては動作不可能です。 • JTAG プログラミング中は JTAG ピンを専用ピンとして使用する必要があり、ユーザー I/O ピンとして使用すること不可能です。 • 初期化の段階で JTAG ピンをトグルすることは不可能です。 • 初期化の前に最低 5 クロックサイクル間、テスト・アクセス・ポート (TAP) コントローラーをリセット状態にし、TDI ピンと TMS ピンを High に、TCK ピンを Low に駆動します。	• TDO • TMS • TCK • TDI

注意: JTAG ピン共有機能をイネーブルする場合は、すべての JTAG ピンをシングルエンドの I/O ピンまたは電圧リファレンス形式の I/O ピンとして割り当てます。

関連情報

- インテル® MAX® 10 FPGA デバイスファミリーのピン接続ガイドライン
  レジスターの推奨値について詳しい情報を提供します。
- インテル® MAX® 10 汎用 I/O ユーザーガイド
  シュミット・トリガー入力に関する詳細情報を提供します。
- インテル MAX 10 のコンフィグレーション・ピン (31 ページ)
- JTAG ピン (5 ページ)

(9) 外部ウィーク・プルアップ抵抗を削除するのであれば、インテルではデバイスがユーザーモードに入った後で削除することを推奨しています。

3.1.1.1. JTAG ピンの共有

表 26. インテル MAX 10 デバイスでの JTAG ピンの共有

コンフィグレーション・ステージ	JTAG ピンの共有	JTAGEN ピン	JTAG ピン (TDO、TDI、TCK、TMS)
ユーザーモード	無効	ユーザー I/O ピン	専用 JTAG ピン
	有効	Low へ駆動	ユーザー I/O ピン
		High へ駆動	専用 JTAG ピン
コンフィグレーション	Don't Care	使用しない	専用 JTAG ピン

注意: JTAG ピンを正常に動作させますためには、表 25 (32 ページ)に従ってピンを設定した上に、ピンの方向 (入力、出力、または双方向) も正しくする必要があります。

3.1.2. 兼用ピンのイネーブル

ユーザーモードで、コンフィグレーション・ピンと JTAG ピンをユーザー I/O ピンとして使用するには、インテル Quartus Prime 開発ソフトウェアで以下の操作を行う必要があります。

1. **Assignments** メニューで **Device** をクリックします。**Settings** ダイアログボックスが表示されます。
2. **Device and Pin Options** をクリックします。**Device and Pin Options** ダイアログボックスが表示されます。
3. **Device and Pin Options** ダイアログボックスのカテゴリーペインで **Configuration** を選択します。
4. Options リストで、以下の手順を実行します。
   - **Enable JTAG pin sharing** にチェックを入れます。
   - **Enable nCONFIG, nSTATUS, and CONF_DONE pins** のチェックを外します。

     注意: このオプションからチェックを外すことで、nCONFIG、nSTATUS、および CONF_DONE ピンがユーザーモードでユーザー I/O ピンとすることが可能になります。
5. **OK** をクリックします。

関連情報

- インテル MAX 10 のコンフィグレーション・ピン (31 ページ)
- JTAG ピン (5 ページ)

3.2. JTAG コンフィグレーションによるインテル MAX 10 デバイスのコンフィグレーション

インテル Quartus Prime 開発ソフトウェアは、JTAG コンフィグレーションに使用できる .sof を生成します。インテル Quartus Prime 開発ソフトウェア Programmer とダウンロード・ケーブルを使用して、インテル MAX 10 を直接コンフィグレーションすることができます。

あるいは、JAM Standard Test and Programming Language (STAPL) Format ファイル (.jam)、JAM Byte Code ファイル (.jbc)、あるいは他のサードパーティー製プログラミング・ツールを備えた Serial Vector Format (.svf) を使用することも可能です。次が実行可能です。

- これらのファイルを自動生成します。
- インテル Quartus Prime Programmer を使用してこれらのファイルを手動で変換します。

関連情報

AN 425: Using the Command-Line Jam STAPL Solution for Device Programming

3.2.1. サードパーティー製プログラミング・ツールに向けたコンフィグレーション・ファイルの自動生成

サードパーティー製プログラミング・ツール・ファイルを生成するには、以下の手順を実行します。

1. **Assignments** メニューで **Settings** をクリックします。**Settings** ダイアログボックスが表示されます。
2. **Category** リストから、**Device** を選択します。**Device** ページが表示されます。
3. **Device and Pin Options** をクリックします。
4. **Device and Pin Options** ダイアログボックスで、カテゴリー・ペインから **Configuration** を選択します。
5. 生成したいプログラミング・ファイルを選択します。

   注意: インテル Quartus Prime 開発ソフトウェアは、選択した各オプションのプログラミング・ファイルに対して次の 2 つのファイルを生成します。

   - `<project_name>.jbc` — `.sof` に相当するファイルです。このファイルは、JTAG コンフィグレーションを実行する際に使用します。
   - `<project_name>_pof.jbc` — `.pof` に相当するファイルです。内部コンフィグレーションを実行する際に使用します。
6. 設定が完了したら OK をクリックします。

3.2.2. インテル Quartus Prime Programmer を使用したサードパーティー・プログラミング・ファイルの生成

`.sof` ファイルおよび `.pof` ファイルを `.jam`、`.jbc`、あるいは `.svf` ファイルに変換するには、次の手順を実行します。

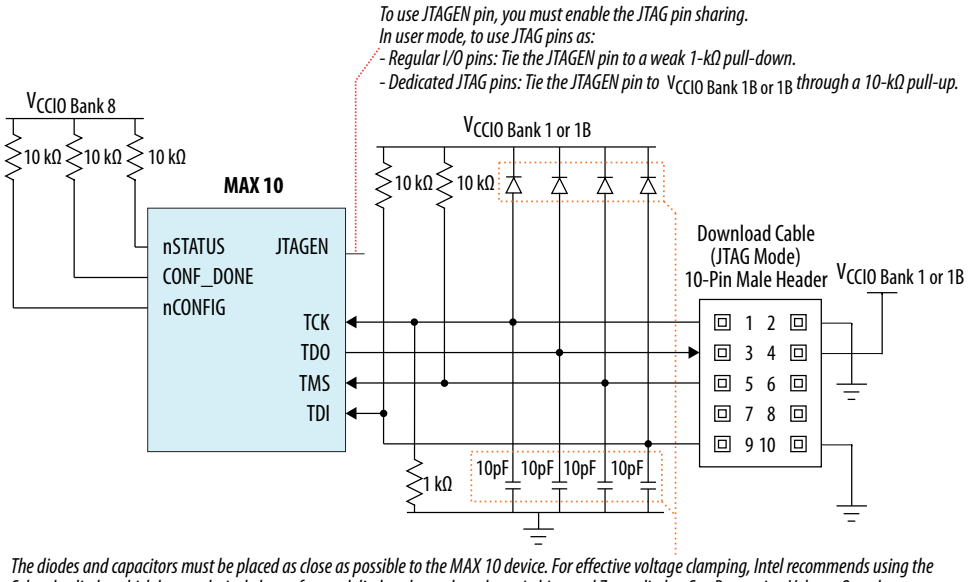
1. **Tools** メニューの **Programmer** をクリックします。
2. **Add File** をクリックして、プログラミング・ファイルを選択し、**Open** をクリックします。
3. インテル Quartus Prime Programmer メニューより **File** > **Create/Update** > **Create Jam, SVF, or ISC File** の順でクリックします。
4. **File Format** リストで、生成したいフォーマットをを選択します。

   注意: 生成されたファイル名は、それが `.sof` ファイルまたは `.pof` ファイルから変換されたかものであるかは示しません。将来の混乱を避けるために、生成されたファイルの名前を変更することを推奨しています。

3.2.3. JTAG コンフィグレーションのセットアップ

図 -12: ダウンロード・ケーブルを使用した JTAG シングル・デバイス・コンフィグレーションの接続セットアップ

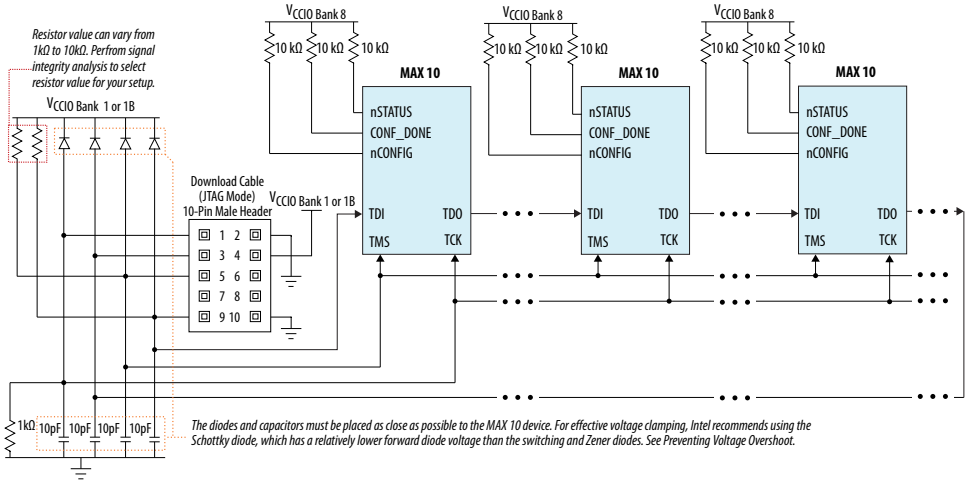
10M02 デバイスの場合、V_{CCIO} バンク 1 に接続し、それ以外のインテル MAX 10 デバイスには、V_{CCIO} バンク 1B に接続します。



The diodes and capacitors must be placed as close as possible to the MAX 10 device. For effective voltage clamping, Intel recommends using the Schottky diode, which has a relatively lower forward diode voltage than the switching and Zener diodes. See Preventing Voltage Overshoot.

図 -13: ダウンロード・ケーブルを使用した JTAG マルチデバイス・コンフィグレーションの接続セットアップ

10M02 デバイスの場合、V_{CCIO} バンク 1 に接続し、それ以外のインテル MAX 10 デバイスには、VCCIO バンク 1B に接続します。



JTAG チェーンのデバイスをコンフィグレーションするには、プログラミング・ソフトウェアは他のデバイスをバイパスモードに設定します。バイパスモードのデバイスは、単一のバイパスレジスターを介して TDI ピンから TDO ピンへプログラミング・データを伝送します。コンフィグレーション・データは 1 クロックサイクル後に TDO ピンで有効になります。

インテル Quartus Prime 開発ソフトウェアは、JTAG ポートを介してコンフィグレーション・プロセスの完了を確認するために CONF_DONE ピンを使用します。

- CONF_DONE ピンが Low の場合、コンフィグレーションが失敗したことを示します。
- CONF_DONE ピンが High の場合、コンフィグレーションが成功したことを示します。

JTAG TDI ポートを使用してコンフィグレーション・データが連続的に伝送された後、デバイスの初期化を行うために TCK ポートがクロッキングされます。

電圧オーバーシュートの防止

電圧オーバーシュートを防止するために、外部ダイオードとコンデンサーを使用する必要があります。例えば、VCCIO および JTAG ヘッダー両方向けの最大 AC 電圧が 3.9 V を超える場合です。ただし、インテルは電源が 2.5 V 以上の場合に外部ダイオードとコンデンサーを使用することを推奨します。

JTAGEN

JTAGEN ピンを使用する場合、インテルは以下の設定を推奨します。

- ユーザーモードに入り、JTAG ピンが通常の I/O ピンである場合は、JTAGEN ピンはウィークプルダウン (1 kΩ) に接続してください。
- ユーザーモードに入り、JTAG ピンが専用ピンである場合は、JTAGEN ピンはウィークプルアップ (10 kΩ) に接続してください。

注意: インテルでは、JTAG ピンの動作を変更するには、ジャンパー付きの 3 ピンヘッダーまたは他の切り替えメカニズムを使用することを推奨します。

3.2.4. JTAG コンフィグレーションでの ICB 設定

ICB 設定は、内部コンフィグレーション・スキームによる .pof のプログラミング時にデバイスにロードされます。JTAG コンフィグレーション時に使用される .sof は、CRAM のみをプログラミングしており、ICB 設定は含んでいません。インテル Quartus Prime Programmer は、以下に基づいて必要な設定を行います。

- ICB 設定を含まないデバイス — ICB 設定が内部フラッシュまたは新しいデバイスから消去されています。
- ICB 設定を含むデバイス — ICB 設定が .pof によって事前にプログラミングされています。

ICB 設定を含まないデバイス

ICB 設定を含まないデバイスでは、デフォルト値が使用されます。ただし、インテル Quartus Prime Programmer は、ウォッチドッグ・タイマー・イネーブルビットを 0 にセットすることでユーザー・ウォッチドッグ・タイマーを無効にします。この手順は、ユーザー・ウォッチドッグ・タイムアウトが原因となる不要なリコンフィグレーションが発生することを回避するためのものです。

デフォルトの ICB 設定が望ましいものではない場合、JTAG コンフィグレーション実行前に、.pof プログラミングによって望ましい ICB 設定をプログラミングします。

ICB 設定を含むデバイス

ICB 設定を含むデバイスでは、設定は内部フラッシュが消去されるまで維持されます。.map ファイルを参照することで、保存されている ICB 設定を確認することができます。JTAG コンフィグレーションは、保存されている ICB 設定に従って動作します。

事前にプログラミングされた ICB 設定が望ましいものではない場合、JTAG コンフィグレーション実行前に、.pof プログラミングによって望ましい ICB 設定をプログラミングします。

関連情報

- .pof と ICB 設定 (38 ページ)
- Verify Protect (22 ページ)
- JTAG セキュアモード (21 ページ)
- ISP 命令とリアルタイム ISP 命令 (9 ページ)
- ユーザー・ウォッチドッグ・タイマー (18 ページ)
- Convert Programming Files を使用した.pof の生成 (39 ページ)
  Convert Programming File を使用して.pof を生成する際の ICB 設定について、詳しい情報を提供します。

3.3. 内部コンフィグレーションを使用するインテル MAX 10 デバイスのコンフィグレーション

インテル MAX 10 デバイスで内部コンフィグレーション・スキームを使用するには、主に 3 つの手順があります。

1. 内部コンフィグレーション・スキームを選択します。
2. ICB 設定で.pof を生成します。
3. .pof を内部フラッシュにプログラミングします。

関連情報

- 内部コンフィグレーション・モード (6 ページ)
- リモート・システム・アップグレード (12 ページ)

3.3.1. 内部コンフィグレーション・モードの選択

コンフィグレーション・モードを選択するには、以下の手順を実行します。

1. インテル Quartus Prime 開発ソフトウェアを開き、インテル MAX 10 デバイスを使用してプロジェクトをロードします。
2. **Assignments** メニューで **Device** をクリックします。**Device** ダイアログボックスが表示されます。
3. **Device** ダイアログボックスで、**Device and Pin Options** をクリックします。**Device and Pin Options** ダイアログボックスが表示されます。
4. **Device and Pin Options** ダイアログボックスの **Configuration** タブをクリックします。
5. **Configuration Scheme** リストから **Internal Configuration** を選択します。
6. **Configuration Mode** リストから、利用可能な 5 つのコンフィグレーション・モードから 1 つのモードを選択します。インテル MAX 10 デバイスで利用可能なモードは、2 つだけです。
7. 必要に応じて **Generate compressed bitstreams** をオンにします。
8. **OK** をクリックします。

3.3.2. .pof と ICB 設定

.pof の生成および ICB の設定には 2 つの方法があります。選択した内部コンフィグレーション・モードが、その設定方法を決定します。

表 27. 内部コンフィグレーション・モードでの.pof 生成と ICB 設定の方法

内部コンフィグレーション・モード	ICB の設定	説明	.pof の生成に使用する方法
Single Compressed Image	ICB は **Device and Pin Options** で設定可能です。	インテル Quartus Prime 開発ソフトウェアは、プロジェクトのコンパイル時に .pof を自動で生成します。	自動生成 .pof (10)
Single Uncompressed Image			
Single Compressed Image with Memory Initialization.	ICB は **Convert Programming Files** タスクの際に設定可能です。	**Convert Programming Files** を使用して .pof を生成する必要があります。	**Convert Programming Files** を使用した .pof の生成
Single Uncompressed Image with Memory Initialization			
Dual Compressed Images			

3.3.2.1. 自動生成.pof

自動的に生成される .pof に ICB を設定するには、以下の手順を実行します。

1. **Assignments** メニューで **Device** をクリックします。**Device** ダイアログボックスが表示されます。
2. **Device** ダイアログボックスで、**Device and Pin Options** をクリックします。**Device and Pin Options** ダイアログボックスが表示されます。
3. **Device and Pin Option** ダイアログボックスのカテゴリーペインで **Configuration** を選択します。
4. **Device Options ...**ボタンをクリックします。
5. **Max 10 Device Options** ダイアログボックスでは、以下が設定可能です。
   a. コンフィグレーション時のユーザー I/O ウィークプルアップ
   b. Verify Protect
6. サードパーティ製プログラミング・ツールに対して設定ファイルを自動的に生成するには、カテゴリーペインから **Programming Files** を選択し、生成したいフォーマットを選択します。

   注意: インテル Quartus Prime 開発ソフトウェアは、選択したオプションのプログラミング・ファイルごとに、次に示すような 2 つのファイルを生成します。
   - <Project_name>.jbc — これは、.sof に相当するファイルです。このファイルを使用して、JTAG コンフィグレーションを実行します。
   - <project_name>_pof.jbc — これは、.sof に相当するファイルです。このファイルを使用して、内部コンフィグレーションを実行します。
7. 設定完了後、**OK** をクリックします。

(10) 自動生成の.pof は暗号化できません。Single Compressed モードと Single Uncompressed モードで暗号化機能を有効にするには、Convert Programming Files の手法を使用します。

3.3.2.2. Convert Programming Files を使用した.pof の生成

.sof ファイルを .pof ファイルに変換し、ICB を設定するには、次の手順に従います。

1. **File** メニューの **Convert Programming Files** をクリックします。
2. **Output programming file** の **Programming file type** リストから、Programmer Object File (**.pof**) を選択します。
3. **Mode** リストから **Internal Configuration** を選択します。
4. **Option/Boot Info** をクリックすると、ICB 設定をセットするための **ICB setting** ダイアログボックスが表示されます。**ICB setting** ダイアログボックスでは次の内容を設定することができます。
   a. コンフィグレーション時のユーザー I/O ウィークプルアップ
   b. CFM0 のみからのデバイス設定

      注意: この機能をイネーブルすると、デバイスは物理的な CONFIG_SEL ピンをサンプリングすることなく、常にコンフィグレーション・イメージ 0 をロードします。コンフィグレーション・イメージ 0 を正常にロードした後で、入力レジスターの config_sel_overwrite ビットを使用してコンフィグレーション・イメージを切り替えることができます。Altera Dual Configuration IP コアの入力レジスターについて、詳しくは関連情報を参照してください。

   c. 使用可能であれば 2 番目のイメージ ISP データをデフォルトとして使用してください。
   d. JTAG セキュア

      注意: TAG セキュア機能は、インテル Quartus Prime 開発ソフトウェアではデフォルトでは無効になっています。このオプションを GUI で可視化するには、key-value ペアの PGM_ENABLE_MAX10_JTAG_SECURITY=ON を使用して、テキストエディターで quartus.ini ファイルを作成し、そのファイルを次のいずれかのフォルダーに保存する必要があります。

      - プロジェクト・フォルダー
      - Windows オペレーティング・システム<Quartus installation folder>\bin64 フォルダー
      - Linux オペレーティング・システム<Quartus installation folder>/linux64 フォルダー

      注意: POF ファイルで JTAG セキュアモードが有効にされている状態で、POF が誤ったキーで暗号化さていれる場合、MAX 10 FPGA デバイスが永久にロックされてしまいます。デバイスが JTAG セキュアモードである場合、外部 JTAG のロックを解除するためには、内部 JTAG のインターフェイスをインスタンス化する必要があります。

   e. 保護を検証します。
   f. 暗号化された POF のみを許可します。
   g. デュアル・コンフィグレーション向けウォッチドッグ・タイマーとウォッチドッグ・タイマー値です。(デュアル圧縮内部イメージでコンパイルされた 2 つのデザインのための 2 ページの .sof を追加すると有効となります)
   h. ユーザー・フラッシュメモリーの設定
   i. RPD ファイルのエンディアン
5. **File name** ボックスで、作成するプログラミング・ファイルのファイル名を指定します。

6. メモリー・マップ・ファイル (.map) を生成するには、**Create Memory Map File** (自動生成される output_file.map) をオンにします。**.map** には **Option/Boot Info** オプションでセットした、ICB 設定と CFM や UFM のアドレスが含まれます。
7. ロー・プログラミング・データ (.rpd) を生成するには、**Create config data RPD** (output_file_auto.rpd を生成する) をオンにします。

   リモート・システム・アップグレードに向けて、各コンフィグレーション・フラッシュメモリーおよびユーザー・フラッシュメモリー (CFM0、CFM1、UFM) セクションの個別のロー・プログラミング・データ (.rpd) がまとめて生成されます。
8. .sof は **Input files to convert** リストから追加することができ、最大 2 つまでの .sof ファイルを追加することができます。

   リモート・システム・アップグレード用に、元のページ 0 データを .pof に保持し、ページ 1 データを新しい .sof ファイルに置き換えることも可能です。これを実行するには、.pof ファイルをページ 0 に追加し、次に .sof ページを追加し、新しい .sof ファイルをページ 1 に追加します。
9. すべての設定が完了した後で、**Generate** をクリックして関連するプログラミング・ファイルを生成します。

関連情報

- インテル® MAX® 10 ユーザー・フラッシュメモリー・ユーザーガイド
  オンチップ・フラッシュ・インテル FPGA IP コアについて詳しい情報を提供します。
- 内部コンフィグレーションでの暗号化 (50 ページ)
  さまざまな設定に基づいてロードされた内部コンフィグレーション・イメージについて、詳しい情報を提供します。

3.3.2.3. インテル Quartus Prime Programmer を使用したサードパーティー・プログラミング・ファイルの生成

.sof ファイルおよび .pof ファイルを .jam、.jbc、あるいは .svf ファイルに変換するには、次の手順を実行します。

1. **Tools** メニューの **Programmer** をクリックします。
2. **Add File** をクリックして、プログラミング・ファイルを選択し、**Open** をクリックします。
3. インテル Quartus Prime Programmer メニューより **File** > **Create/Update** > **Create Jam, SVF, or ISC File** の順でクリックします。
4. **File Format** リストで、生成したいフォーマットをを選択します。

   注意: 生成されたファイル名は、それが .sof ファイルまたは .pof ファイルから変換されたかものであるかは示しません。将来の混乱を避けるために、生成されたファイルの名前を変更することを推奨しています。

3.3.3. 内部フラッシュへの.pof のプログラミング

インテル Quartus Prime Programmer を使用すれば、JTAG インターフェイスを介して .pof を CFM へプログラミングすることができます。また、内部フラッシュの UFM 部分も インテル Quartus Prime Programmer でプログラミングすることができます。

.pof をフラッシュにプログラミングするには、以下の手順を実行します。

1. **Tools** メニューの **Programmer** をクリックします。
2. **Programmer** ウィンドウの **Hardware Setup** をクリックし、現在選択しているハードウェアのドロップダウン・リストから **USB Blaster** を選択します。
3. **Mode** リストから **JTAG** を選択します。
4. 左側のペインの **Auto Detect** ボタンをクリックします。
5. プログラミングするデバイスを選択し、**Add File** をクリックします。
6. 選択したデバイスに対してプログラミングを実行する `.pof` を選択します。
7. 内部フラッシュのプログラミングにはいくつかのオプションがあります。
   - CFM0/CFM1/CFM2 のいずれかのみをプログラミングするには、Program/Configure カラムで該当する CFM を選択します。
   - UFM のみをプログラミングするには、Program/Configure カラムで UFM を選択します。
   - CFM と UFM のみをプログラミングするには、Program/Configure カラムで CFM と UFM を選択します。

     注意: このオプションでは ICB 設定は維持されます。ただし、プログラミングを開始する前に、インテル Quartus Prime Programmer がデバイスの ICB 設定と選択された `.pof` の ICB 設定が同じであるかを確認します。ICB 設定が異なっていれば、インテル Quartus Prime Programmer が ICB 設定を上書きします。
   - ICB 設定を含む内部フラッシュ全体をプログラミングするには、Program/Configure カラムで<yourpoffile.pof>を選択します。
8. リアルタイム ISP モードを有効にするには、**Enable real-time ISP to allow background programming** をオンにします。
9. すべての設定をセットした後で、**Start** をクリックしてプログラミングを開始します。

3.4. インテル Quartus Prime 開発ソフトウェアによる ISP クランプの実装

ISP クランプを実装するには、以下を実行します。

1. ピンステートの情報 (`.ips`) ファイルを作成します。`.ips` ファイルは、デバイスが ISP クランプ動作時における、デバイスのすべてのピンのステートを定義します。既存の `.ips` ファイルを使用できます。
2. `.ips` ファイルを実行します。

注意: ターゲットするデバイスとパッケージが同じであれば、作成した `.ips` ファイルを使用して、どのようなデザインを持つデバイスもプログラミングすることができます。`.ips` ファイルは、POF ファイルとともに使用する必要があります。

関連情報

ISP クランプ (9 ページ)

3.4.1. IPS ファイルの作成

`.ips` ファイルを作成するには、以下の手順を実行します。

1. ツールバーで **Programmer** をクリックするか、**Tools** メニューの **Programmer** をクリックして、**Programmer** を開きます。
2. Programmer で **Add File** をクリックし、プログラミング・ファイル (POF、Jam、または JBC) を追加します。
3. プログラミング・ファイルをクリック (全行が強調表示されます) し、**Edit** メニューの **ISP Clamp State Editor** をクリックします。
4. **ISP Clamp State Editor** でデザイン内のピンのステートを指定します。すべてのピンはデフォルトで **tri-state** に設定されています。
5. 変更後に IPS ファイルを保存するには、**Save** をクリックします。

3.4.2. IPS ファイルの実行

ISP クランプを実行するには、以下の手順を実行します。

1. Quartus Prime **Programmer** で、デバイスにプログラミングする `.pof` を選択します。
2. `.pof` を選択し、**Add IPS File** を右クリックで選択し、**ISP CLAMP** をオンにします。

   注意: コンフィグレーション完了後、I/O クランプのスタートアップ遅延を変更することができます。**Tools** > **Options** を選択し、**Overwrite MAX10 configuration start up delay when using IO Clamp in Programmer** オプションをオンにすることで、必要に応じた遅延値が変更可能となります。
3. `Program/Configure` カラムで**.pof** を選択します。

   注意: サードパーティー・プログラミングでは、`.ips` ファイルで `.pof` ファイルから `.jam` または `.jbc` ファイルを生成することができます。
4. すべての設定をセットした後で、**Start** をクリックしてプログラミングを開始します。

3.5. ユーザーロジックを介したリモート・システム・アップグレードへのアクセス

以下の例は、インテル MAX 10 デバイスで WYSIWYG アトムの入力および出力ポートを定義する方法を示しています。

注意: WYSIWYG とは、インテル Quartus Prime 開発ソフトウェア内で Verilog Quartus Mapping ネットリストの最適化を実行するテクニックのことを指します。

```
fiftyfivenm_rublock <rublock_name>
(
    .clk(<clock source>),
    .shiftnld(<shiftnld source>),
    .captnupdt(<captnupdt source>),
    .regin(<regin input source from the core>),
    .rsttimer(<input signal to reset the watchdog timer>),
    .rconfig(<input signal to initiate configuration>),
    .regout(<data output destination to core>)
);
defparam <rublock_name>.sim_init_config = <initial configuration for simulation
only>;
defparam <rublock_name>.sim_init_watchdog_value = <initial watchdog value for
simulation only>;
defparam <rublock_name>.sim_init_config = <initial status register value for
simulation only>;
```

表 28. ポートの定義

ポート	入力/出力	定義
<rublock_name>	-	RSU ブロック固有の識別子です。記述言語の選択 (例えば Verilog、VHDL、AHDL 等) に応じて適正な識別子名が表記されます。このフィールドは必須です。
.clk(<clock source>)	入力	この信号はこのセルのクロック入力を示します。このセルの全動作はこのクロックの立ち上がりエッジに対して生じます。セルへのデータのロードであっても、セルからのデータ出力であっても、常に立ち上がりエッジで生じます。このフィールドは必須です。
.shiftnld(<shiftnld source>)	入力	この信号はリモート・システム・アップグレード・ブロックへの入力です。shiftnld = 1 の場合、データは clk の立ち上がりエッジごとに内部シフトレジスターから regout へシフトされ、また regin から内部シフトレジスターへシフトされます。このフィールドは必須です。
.captnupdt(<captnupdt source>)	入力	この信号はリモート・システム・アップグレード・ブロックへの入力です。この信号は、コンフィグレーション・モードを読み出すタイミング、またはコンフィグレーションを制御するレジスターに書き込むタイミングのプロトコルを制御します。このフィールドは必須です。
.regin(<regin input source from the core>)	入力	この信号は、コアにロードされているすべてのデータに対するリモート・システム・アップグレード・ブロックへの入力です。データは clk の立ち上がりエッジに内部レジスターにシフトされます。このフィールドは必須です。
.rsttimer(<input signal to reset the watchdog timer>)	入力	この信号は、リモート・アップデート・ブロックのウォッチドッグ・タイマーへの入力です。この信号が High の場合、ウォッチドッグ・タイマーをリセットします。このフィールドは必須です。
.rconfig(<input signal to initiate configuration>)	入力	この信号は、リモート・アップデート・ブロックのコンフィグレーション・セクションへの入力です。この信号が High の場合、リコンフィグレーションを開始します。このフィールドは必須です。
.regout(<data output destination to core>)	出力	これは1ビットの出力で、.clk の立ち上がりエッジごとに更新される内部シフトレジスターの出力です。データは制御信号に応じて出力されます。このフィールドは必須です。

関連情報

- デュアル・コンフィグレーション インテル FPGA IP コアのリファレンス (60 ページ)
- リモート・システム・アップグレード (12 ページ)
- AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor
  インテル® MAX® 10 FPGA デバイスのリモート・システム・アップグレード向けリファレンス・デザインを提供します。
- I2C Remote System Update Example
  この例では、I2C プロトコルの使用によるリモート・システム・アップグレードを示します。

3.6. エラー検出

3.6.1. エラー検出機能の検証

CRC 回路で 32 ビットの CRC ストレージレジスターを変更することによって、ソフトエラーを注入することができます。引き起こされた障害を検証した後は、同じ命令を使用して正しい値を挿入することにより、32 ビットの CRC 値を正しい CRC 値に復元することができます。既知の不正な値で更新する前に、必ず正しい値を読み出してください。

インテル MAX 10 デバイスは、ユーザーモードで CHANGE_EDREG JTAG 命令をサポートしており、これにより 32 ビットのストレージレジスターに書き込みをすることができます。**.jam** を使用すれば、テストと検証のプロセスを自動化することができます。この命令は、デバイスがユーザーモードにあるときにのみ実行することができます。この命令により、デバイスをリコンフィグレーションすることなくインシステムで CRC 機能を動的に検証可能です。その後で、CRC 回路に切り換えて、SEU に起因する実際のエラーを確認することができます。

テスト完了後に CRC エラーをクリアして元の CRC 値を復元するには、次のいずれかの方法を使用します。

- 5TCK クロックの間、TMS を High に保持して、TAP コントローラーを RESET 状態にします。
- デバイスの電源を切断し、再投入します。
- 次の手順を実行します。
  1. コンフィグレーション完了後、CHANGE_EDREG JTAG 命令を使用して計算済みの正しい CRC 値をシフトアウトし、CRC ストレージレジスターに不正な CRC 値をロードします。エラーが検出されると、CRC_ERROR ピンがアサートされます。
  2. 計算済みの正しい CRC 値をシフトインするには、CHANGE_EDREG JTAG 命令を使用します。エラー検出 CRC 回路が動作していることを示すために、CRC_ERROR ピンがディアサートされます。

例-2: JAM ファイル

```
'EDCRC_ERROR_INJECT

ACTION ERROR_INJECT = EXECUTE;
DATA DEVICE_DATA;
BOOLEAN out[32];
BOOLEAN in[32] = $02040608;    'shift in any wrong CRC value
ENDDATA;
PROCEDURE EXECUTE USES DEVICE_DATA;
BOOLEAN X = 0;
DRSTOP IDLE;
IRSTOP IDLE;
STATE IDLE;
IRSCAN 10, $015;               'shift in CHANGE_EDREG instruction
WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;
DRSCAN 32, in[31..0], CAPTURE out[31..0];
WAIT IDLE, 10 CYCLES, 50 USEC, IDLE;
PRINT " ";
PRINT "Data read out from the Storage Register: "out[31], out[30], out[29],
out[28], out[27],
out[26], out[25], out[24], out[23], out[22], out[21], out[20], out[19],
out[18], out[17], out[16], out[15], out[14], out[13], out[12], out[11],
out[10], out[9], out[8], out[7], out[6], out[5], out[4], out[3],
out[2], out[1], out[0];          'Read out correct precomputed CRC value
PRINT " ";
STATE IDLE;
EXIT 0;
ENDPROC;
```

下記のコマンドラインにより、`quartus_jli` 実行ファイルを使用して、`.jam` ファイルを動作させることが可能です。

```
quartus_jli -c<cable index> -a<action name> <filename>.jam
```

関連情報

- SEU の緩和とコンフィグレーション・エラーの検出 (23 ページ)
- AN 425: Using the Command-Line Jam STAPL Solution for Device Programming
  実行可能な quartus_jli コマンドラインについて詳しい情報を提供します。

3.6.2. エラー検出の有効化

インテル Quartus Prime 開発ソフトウェアの CRC エラー検出機能は、CRC_ERROR の出力をオプションの兼用 CRC_ERROR ピンに生成します。

CRC を使用したエラー検出機能を有効にするには、以下の手順を実行します。

1. インテル Quartus Prime 開発ソフトウェアを開き、インテル MAX 10 デバイスファミリーを使用するプロジェクトをロードします。
2. **Assignments** メニューで **Device** をクリックします。**Device** ダイアログボックスが表示されます。
3. **Device** ダイアログボックスで、**Device and Pin Options** をクリックします。**Device and Pin Options** ダイアログボックスが表示されます。
4. **Device and Pin Options** をクリックします。**Device and Pin Options** ダイアログボックスが表示されます。
5. **Device and Pin Option** ダイアログボックスのカテゴリーペインで **Error Detection CRC** を選択します。
6. **Enable Error Detection CRC_ERROR pin** をオンにします。
7. **Divide error check frequency by** フィールドに有効な除数を入力します。

   除数値は、コンフィグレーション・オシレーター出力クロックの周波数を分周します。この出力クロックは、エラー検出処理用のクロックソースとして使用されます。
8. **OK** をクリックします。

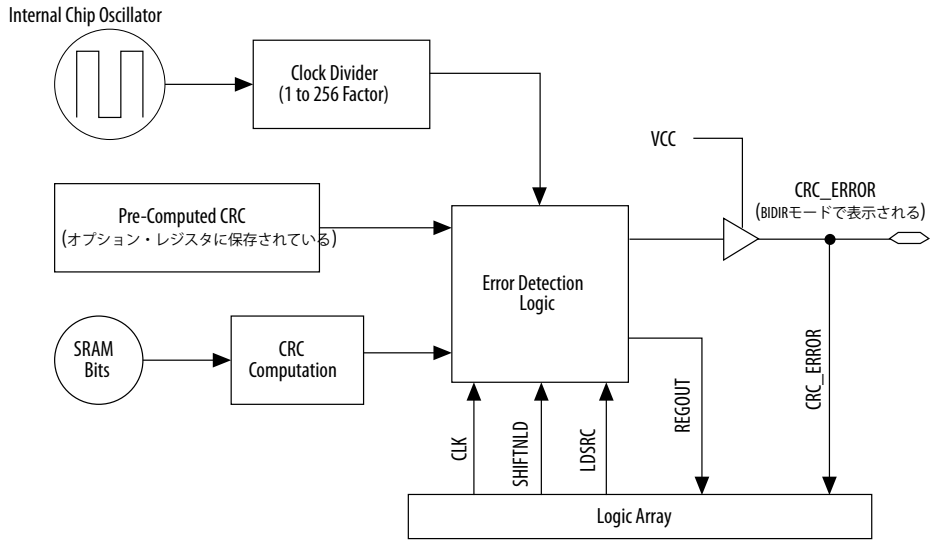
関連情報

SEU の緩和とコンフィグレーション・エラーの検出 (23 ページ)

3.6.3. ユーザーロジックによるエラー検出ブロックへのアクセス

エラー検出回路は、計算された 32 ビットの CRC シグネチャーを 32 ビットのレジスターに格納します。コアからのユーザーロジックがこのシグネチャーを読み出します。`fiftyfivenm_crcblock` プリミティブは、ユーザーロジックからエラー検出回路へのインターフェイスを確立するために使用される WYSIWYGl コンポーネントです。`fiftyfivenm_crcblock` プリミティブ・アトムは、アトムに含まれている必要がある入力ポートと出力ポートを含みます。ロジックアレイにアクセスするには、`fiftyfivenm_crcblock` WYSIWYG アトムをデザインに挿入する必要があります。.clk ポートのクロック周波数は、EDCRC ブロックのクロック周波数に従うことが推奨されます。

図 -14: インテル MAX 10 デバイスのインターフェイスを含むエラー検出ブロック図



以下の例は、インテル MAX 10 デバイスで WYSIWYG アトムの入力および出力ポートを定義する方法を示しています。

```
fiftyfivenm_crcblock <name>
(
    .clk(<ED_CLK clock source>),
    .shiftnld(<ED_SHIFTNLD source>),
    .ldsrc (<LDSRC source>),
    .crcerror(<CRCERROR_CORE out destination>),
    .regout(<output destination>)
);
defparam <crcblock_name>.oscillator_divider = <internal oscillator division (1
to 256)>;
```

表 29. ポートの定義

ポート	入力/出力	定義
<crcblock_name>	—	CRC ブロック固有の識別子であり、ここには、Verilog HDL、VHDL、AHDL といった記述言語の選択に応じて適正な識別子名が表記されます。このフィールドは必須です。
.clk(<clock source>)	入力	この信号はこのセルのクロック入力を示します。このセルの全ての動作はこのクロックの立ち上がりエッジに対して生じます。セルへのデータのロードであっても、セルからのデータ出力であっても、常に立ち上がりエッジで生じます。このポートは必須です。
.shiftnld (<shiftnld source>)	入力	この信号はエラー検出ブロックへの入力です。shiftnld=1 であれば、clk のすべての立ち上がりエッジで、データが内部シフトレジスターから regout にシフトされます。shiftnld=0 であれば、シフトレジスターはあらかじめ計算された CRC 値、またはアップデート・レジスターの内容のどちらかを ldsrc ポート入力に応じてパラレルにロードします。このポートは必須です。
.ldsrc (<ldsrc source>)	入力	この信号はエラー検出ブロックへの入力です。ldsrc=0 であれば、shiftnld=0 の際に clk の立ち上がりエッジで 32 ビットのシフトレジスターへロードするために、あらかじめ計算された CRC レジスターが選択されます。Ifldsrc=1 であれば、shiftnld=0 の際に clk の立ち上がりエッジでシフトレジスターへロードするために、シグネチャー・レジスター (CRC の計算結果) が選択されます。shiftnld=1 であればこのポートは無視されます。このポートは必須です。
.crcerror (<crcerror out destination>)	出力	この信号はセルの出力であり、これは clk ポートではなく、デバイスの内部オシレーター (100 MHz または 80 MHz 内部オシレーター) に同期されます。この信号は、SRAM ビットが反転していること、内部 CRC の計算結果があらかじめ計算された値と異なることがエラーブロックで検出

continued...

ポート	入力/出力	定義
		された場合に、自動的に High にアサートします。この信号は、出力ピンか双方向ピンのどちらかに接続する必要があります。出力ピンに接続する場合、モニタリングが可能となるのは CRC_ERROR ピンのみとなります (コアはこの出力にアクセスできません)。CRC_ERROR 信号がコアロジックによってエラー検出ロジックの読み出しに使用される場合には、この信号を BIDIR ピンに接続する必要があります。VCC に接続された oe ポートを有する $_{BIDIR}$ ピンに供給することにより、信号は間接的にコアに供給されます。
.regout (<output destination>)	出力	この信号は clk ポートと同期しているエラー検出シフトレジスターの出力であり、コアロジックにより読み出されます。各サイクルごとに 1 ビットシフトします。シフトレジスターから 32 ビットを読み出すには、clk 信号を 31 サイクルで駆動する必要があります。.regout ポートでの値は、実際の値の逆数です。

関連情報

- SEU の緩和とコンフィグレーション・エラーの検出 (23 ページ)
- エラー検出のタイミング (25 ページ)

3.7. データ圧縮の有効化

圧縮を有効にすると、インテル Quartus Prime 開発ソフトウェアが圧縮されたコンフィグレーション・データでコンフィグレーション・ファイルを生成します。

圧縮されたコンフィグレーション・ファイルは、内部コンフィグレーション・スキームでデュアル・コンフィグレーション・モードを使用するために必要です。このファイルの圧縮は、内部フラッシュメモリーで必要になる容量を削減し、インテル MAX 10 デバイスファミリーにビットストリームを送信する時間を短縮します。インテル Quartus Prime 開発ソフトウェアでは、インテル MAX 10 デバイスファミリーのビットストリームの圧縮を有効にするには、次の 2 つの方法があります。

- デザインのコンパイル前に **Compiler Settings** メニューを使用します。
- デザインのコンパイル後に **Convert Programming Files** オプションを使用します。

3.7.1. デザインのコンパイルの前に圧縮を有効にする場合

デザインのコンパイルの前に圧縮を有効にするには、以下の手順を実行します。

1. **Assignments** メニューで **Device** をクリックします。**Device** ダイアログボックスが表示されます。
2. **Device and Pin Options** をクリックします。**Device and Pin Options** ダイアログボックスが表示されます。
3. **Device and Pin Option** ダイアログボックスのカテゴリーペインで **Configuration** を選択します。
4. **Generate compressed bitstreams** をオンにします。
5. **OK** をクリックします。
6. **Settings** ダイアログボックスで **OK** をクリックします。

関連情報

コンフィグレーション・データの圧縮 (27 ページ)

3.7.2. デザインのコンパイル後に圧縮を有効にする場合

デザインのコンパイルの後に圧縮を有効にするには、以下の手順を実行します。

1. **File** メニューの **Convert Programming Files** をクリックします。
2. **Output programming file** で Programming file type プルダウンメニューから、目的のファイルタイプを選択します。
3. Programmer Object File (**.pof**) を選択した場合は、ファイルタイプの下の、Configuration Device を指定する必要があります。
4. **Input files to convert** ボックスで、**SOF Data** を選択します。
5. **Add File** をクリックし、インテル MAX 10 デバイスファミリーの**.sof** を開きます。
6. **Convert Programming Files** ダイアログボックスで、**SOF Data** に追加した**.pof** を選択し、**Properties** をクリックします。
7. **SOF Properties** ダイアログボックスで、**Compression** オプションをオンにします。

関連情報

コンフィグレーション・データの圧縮 (27 ページ)

3.8. AES の暗号化

本項では、デザイン・セキュリティーに向けた AES 暗号化の適用に関するガイドラインについて詳しく説明します。インテル MAX 10 デバイスにデザイン・セキュリティーを適用するには、2 つの手順があります。1 つ目の手順では .ekp (Encryption Key Programming) ファイルの生成します。2 つ目の手順でデバイスへの .ekp ファイルをプログラミングします。

.ekp ファイルは、プログラミングに使用するハードウェアもしくはシステムに応じて異なる形式を有します。インテル Quartus Prime 開発ソフトウェアがサポートする 3 つのファイル形式を以下に示します。

- .jbc (JAM Byte Code) ファイル
- .jam (JAM™ Standard Test and Programming Language (STAPL) Format) ファイル
- .svf (Serial Vector Format) ファイル

.ekp ファイルタイプのみ インテル Quartus Prime 開発ソフトウェアによって自動生成されます。キーのプログラミングに .jbc、.jam、および .svf ファイルが必要な場合、インテル Quartus Prime 開発ソフトウェアを使用してこれらのファイルを生成する必要があります。

注意: インテルでは、.ekp ファイルの機密性を保持することを推奨しています。

3.8.2. .ekp ファイルからの.jam/.jbc/.svf ファイルの生成

.ekp ファイルから .jam/.jbc/.svf を生成するには以下の手順を実行します。

1. **Tools** メニューで **Programmer** をクリックし、**Programmer** ダイアログボックスを開きます。
2. **Mode** リストで、プログラミング・モードとして **JTAG** を選択します。
3. **Hardware Setup** をクリックし、**Hardware Setup** ダイアログボックスを開きます。

4. **currently selected hardware list** リストでプログラミング・ハードウェアとして **USB Blaster** を選択し、**Done** をクリックします。
5. **Add File** をクリックし、**Select Programmer File** ダイアログボックスを開きます。
6. **File name** フィールドに<filename>.ekp と入力し、**Open** をクリックします。
7. 追加した.ekp ファイルを選択し、**Program/Configure** をクリックします。
8. **File** メニューで **Create/Update** にカーソルを合わせ、**Create JAM, SVF, or ISC File** をクリックします。**Create JAM, SVF, or ISC File** ダイアログボックスが表示されます。
9. **File format** フィールドで.ekp ファイルに必要なファイル形式を選択します。
   - .jam (JEDEC STAPL Format)
   - .jbc (Jam STAPL Byte Code)
   - .svf (Serial Vector Format)
10. **File name** フィールドにファイル名を入力するか、あるいはファイルを検索・選択します。
11. **OK** をクリックし、.jam、.jbc、または.svf ファイルを生成します。

3.8.3. .ekp ファイルと暗号化.pof ファイルのプログラミング

暗号化された.pof ファイルおよび.ekp ファイルのプログラミングには、2 つの手法があります。

- .ekp および.pof ファイルを個別にプログラミングします。

  注意: Allow encrypted POF only のオプションが無効にされた場合にのみ、.ekp と**.pof** ファイルを個別にプログラミングすることができます。
- .ekp を.pof に統合し、両方をまとめてプログラミングします。

3.8.3.1. .ekp ファイルと暗号化された.pof ファイルを個別にプログラミングする場合

インテル Quartus Prime 開発ソフトウェアを使用して.ekp と暗号化された.pof を個別にプログラミングするには、以下の手順を実行します。

1. インテル Quartus Prime Programmer の **Mode** リストで、プログラミング・モードとして **JTAG** を選択します。
2. **Hardware Setup** をクリックし、**Hardware Setup** ダイアログボックスを開きます。
3. **Currently selected hardware** リストでプログラミング・ハードウェアとして **USB Blaster** を選択し、**Done** をクリックします。
4. **Add File** をクリックし、**Select Programmer File** ダイアログボックスを開きます。
5. **File name** フィールドに<filename>.ekp と入力し、**Open** をクリックします。
6. 追加した.ekp ファイルを選択し、**Program/Configure** をクリックします。
7. **Start** をクリックしてキーをプログラミングします。

   注意: インテル Quartus Prime 開発ソフトウェアのメッセージ画面に、キー・プログラミング動作の成功または失敗についての情報が提供されます。一旦.ekp がプログラミングされると、.pof が別にプログラミング可能となります。.ekp によってプログラミングされた内部フラッシュにセキュリティー・キーを保持するには、続けて以下の手順を実行します。

8. 選択したデバイスに対してプログラミングを実行する.pof を選択します。
9. CFM および UFM のために更新する必要がある機能ブロックのみを子レベルでチェックします。Programmer GUI を使用している場合、動作を親レベルでチェックしないでください。
10. すべての設定をセットした後で、**Start** をクリックしてプログラミングを開始します。

3.8.3.2. .ekp を.POF に統合しプログラミングする場合

インテル Quartus Prime 開発ソフトウェアを使用して.ekp を.pof に統合し、両方をまとめてプログラミングするには、以下の手順を実行します。

1. インテル Quartus Prime Programmer の **Mode** リストで、プログラミング・モードに **JTAG** を選択します。
2. **Hardware Setup** をクリックし、**Hardware Setup** ダイアログボックスを開きます。
3. **Currently selected hardware** リストでプログラミング・ハードウェアとして **USB Blaster** を選択し、**Done** をクリックします。
4. 左側のペインの **Auto Detect** ボタンをクリックします。
5. デバイスにプログラミングする.pof を選択します。
6. <yourpoffile.pof>を選択して右クリックし、**Add EKP File** を選択して、.ekp ファイルを.pof ファイルに統合します。

   .ekp が.pof に統合した後は、統合した.pof は新しい.pof として保存可能です。新しく保存されたファイルには、.ekp の情報と統合されたオリジナル.pof が含まれます。
7. **Program/Configure** カラムで<yourpoffile.pof>を選択します。
8. すべての設定を完了した後で、**Start** をクリックしてプログラミングを開始します。

3.8.4. 内部コンフィグレーションでの暗号化

内部コンフィグレーション時に、FPGA は格納しているキーで.pof を復号化し、復号データをコンフィグレーションに使用します。コンフィグレーション中にロードされるコンフィグレーション・イメージも暗号化設定と **Configure device from CFM0 only** 設定に影響されます。

表 30. 暗号化設定、暗号化キー、および CONFIG_SEL ピン設定に基づくコンフィグレーション・イメージ結果

以下の表に **Configure device from CFM0 only** が無効にされているシナリオを示します。キー X とキー Y はデバイスおよびコンフィグレーション・イメージに含まれているセキュリティー・キーです。

コンフィグレーション・イメージ・モード	CFM0 (イメージ 0) 暗号化キー	CFM1 (イメージ 1) 暗号化キー	デバイスに格納されたキー	Allow encrypted POF only	CONFIG_SEL ピン	パワーアップ後にロードされるデザイン
シングル	非暗号化	使用不可	キーなし	無効	0	イメージ 0
シングル	非暗号化	使用不可	キーなし	無効	1	イメージ 0
シングル	非暗号化	使用不可	キー X	無効	0	イメージ 0
シングル	非暗号化	使用不可	キー X	無効	1	イメージ 0
シングル	非暗号化	使用不可	キー X	有効	0	コンフィグレーション不成功
シングル	非暗号化	使用不可	キー X	有効	1	コンフィグレーション不成功

continued...

コンフィグレーション・イメージ・モード	CFM0 (イメージ 0) 暗号化キー	CFM1 (イメージ 1) 暗号化キー	デバイスに格納されたキー	Allow encrypted POF only	CONFIG_SEL ピン	パワーアップ後にロードされるデザイン
シングル	キー X	使用不可	キーなし	有効	0	コンフィグレーション不成功
シングル	キー X	使用不可	キーなし	有効	1	コンフィグレーション不成功
シングル	キー X	使用不可	キー X	有効	0	イメージ 0
シングル	キー X	使用不可	キー X	有効	1	イメージ 0
シングル	キー X	使用不可	キー Y	有効	0	コンフィグレーション不成功
シングル	キー X	使用不可	キー Y	有効	1	コンフィグレーション不成功
デュアル	非暗号化	非暗号化	キーなし	無効	0	イメージ 0
デュアル	非暗号化	非暗号化	キーなし	無効	1	イメージ 1
デュアル	キー X	非暗号化	キーなし	無効	0	イメージ 1 (11)
デュアル	キー X	非暗号化	キーなし	無効	1	イメージ 1
デュアル	キー X	非暗号化	キー X	無効	0	イメージ 0
デュアル	キー X	非暗号化	キー X	無効	1	イメージ 1
デュアル	キー X	非暗号化	キー X	有効	0	イメージ 0
デュアル	キー X	非暗号化	キー X	有効	1	イメージ 0
デュアル	キー X	非暗号化	キー Y	有効	0	コンフィグレーション不成功
デュアル	キー X	非暗号化	キー Y	有効	1	コンフィグレーション不成功
デュアル	キー X	キー X	キーなし	有効	0	コンフィグレーション不成功
デュアル	キー X	キー X	キーなし	有効	1	コンフィグレーション不成功
デュアル	キー X	キー X	キー X	有効	0	イメージ 0
デュアル	キー X	キー X	キー X	有効	1	イメージ 1
デュアル	キー X	キー Y	キー X	有効	0	イメージ 0
デュアル	キー X	キー Y	キー X	有効	1	イメージ 0 (12)
デュアル	キー Y	キー Y	キー Y	有効	0	イメージ 0
デュアル	キー Y	キー Y	キー Y	有効	1	イメージ 1
デュアル	キー X	キー Y	キー Y	有効	0	イメージ 1(11)
デュアル	キー X	キー Y	キー Y	有効	1	イメージ 1

(11) イメージ 0 でのコンフィグレーションが失敗すると、デバイスは自動的にイメージ 1 をロードします。

(12) イメージ 1 でのコンフィグレーションが失敗すると、デバイスは自動的にイメージ 0 をロードします。

表 31. 暗号化設定と暗号化キーに基づくコンフィグレーション・イメージの結果

以下の表に **Configure device from CFM0 only** が有効の場合のシナリオを示します。

CFM0 (イメージ 0) 暗号化キー	デバイスに格納されたキー	Allow encrypted POF only	パワーアップ後にロードされるデザイン
非暗号化	キーなし	無効	イメージ 0
非暗号化	キー X	無効	イメージ 0
非暗号化	キー Y	無効	イメージ 0
非暗号化	キーなし	有効	コンフィグレーション不成功
非暗号化	キー X	有効	コンフィグレーション不成功
非暗号化	キー Y	有効	コンフィグレーション不成功
キー X	キーなし	無効	コンフィグレーション不成功
キー X	キー X	無効	イメージ 0
キー X	キー Y	無効	コンフィグレーション不成功
キー X	キーなし	有効	コンフィグレーション不成功
キー X	キー X	有効	イメージ 0
キー X	キー Y	有効	コンフィグレーション不成功
キー Y	キーなし	無効	コンフィグレーション不成功
キー Y	キー X	無効	コンフィグレーション不成功
キー Y	キー Y	無効	イメージ 0
キー Y	キーなし	有効	コンフィグレーション不成功
キー Y	キー X	有効	コンフィグレーション不成功
キー Y	キー Y	有効	イメージ 0

関連情報

Convert Programming Files を使用した.pof の生成 (39 ページ)

3.9. インテル MAX 10 JTAG セキュアデザインの例

このデザイン例では、LOCK および UNLOCK JTAG 命令を実行するための、内部 JTAG WYSIWYG アトムのインスタンス化および インテル Quartus Prime 開発ソフトウェアで実現されるユーザーロジックの例について説明します。このデザイン例は、JTAG セキュアモードが有効であるインテル MAX 10 デバイスを対象としています。

関連情報

- 利用可能な JTAG 命令 (22 ページ)
- コンフィグレーション・フラッシュメモリーへのアクセス許可 (23 ページ)
- JTAG Secure Design Example

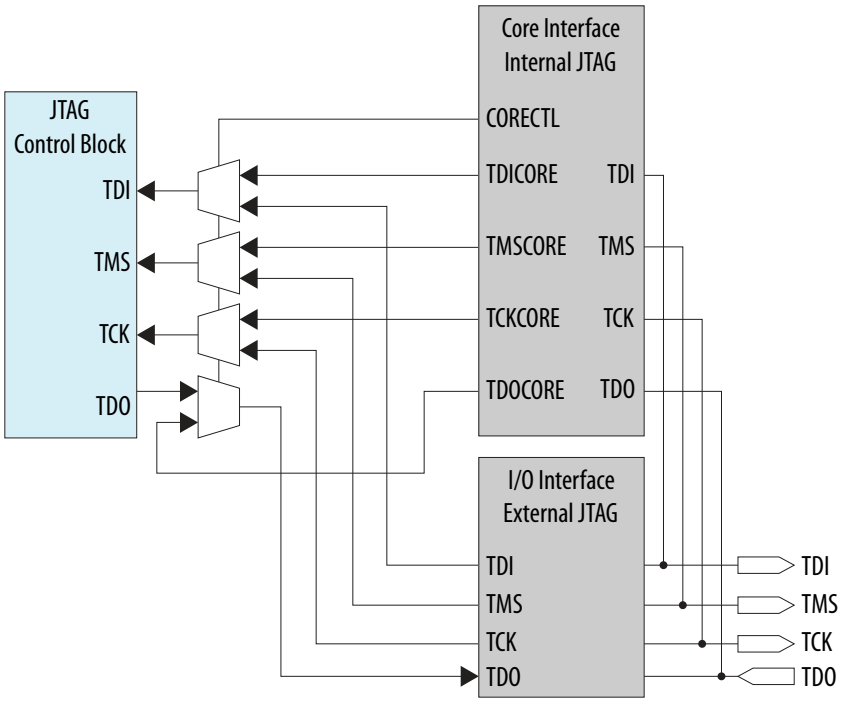
3.9.1. 内部 JTAG インターフェイス

インテル MAX 10 デバイスには、JTAG コントロール・ブロックにアクセスするために 2 つのインターフェイスがあります。

- 外部 JTAG インターフェイス — 物理的な JTAG ピン、TCK、TDI、TDO および TMS から JTAG コントロール・ブロックに接続します。
- 内部 JTAG インターフェイス — 内部 FPGA コア・ファブリックから JTAG コントロール・ブロックに接続します。

JTAG コントロール・ブロックへのアクセスに向けては、外部または内部 JTAG インターフェイスのいずれかを一度に 1 つのだけ使用することができます。外部 JTAG インターフェイスは通常、プログラミング・ケーブルを使用した JTAG コンフィグレーションに使用されます。内部 JTAG インターフェイスにアクセスするには、インテル Quartus Prime 開発ソフトウェアのデザインに WYSIWYG アトムを含める必要があります。

図 -15: 内部および外部 JTAG インターフェイスの接続



注意: インテル MAX 10 デバイスの内部 JTAG を正しく機能させるには、JTAG WYSIWYG アトム内での 4 つの JTAG 信号 (TCK、TDI、TMS および TDO) をすべて外部に配線する必要があります。インテル Quartus Prime 開発ソフトウェアは、専用 JTAG ピンに対応するポートを自動的に割り当てます。

3.9.2. 内部 JTAG ブロックアクセスの WYSIWYG アトム

以下の例は、インテル MAX 10 デバイスで WYSIWYG アトムの入力および出力ポートを定義する方法を示しています。

```
fiftyfivenm_jtag <name>
(
    .tms(),
    .tck(),
```

```
    .tdi(),
    .tdoutap(),
    .tdouser(),
    .tdicore(),
    .tmscore(),
    .tckcore(),
    .corectl(),
    .tdo(),
    .tmsutap(),
    .tckutap(),
    .tdiutap(),
    .shiftuser(),
    .clkdruser(),
    .updateuser(),
    .runidleuser(),
    .usr1user(),
    .tdocore(),
    .ntdopinena()
);
```

表 32. ポートの説明

ポート	入力/出力	機能
<name>	—	インテル MAX 10 JTAG WYSIWYG アトムの識別子で、Verilog HDL、VHDL、および AHDL などの所定の記述言語に対して適正な識別名を表します。
.corectl()	入力	コア・インターフェイスからの内部 JTAG アクセスを有効にする、JTAG コントロール・ブロックへのアクティブ High 入力です。コンフィグレーション後に FPGA がユーザーモードに入る際、このポートはデフォルトで Low となります。このポートをロジック High にすると、内部 JTAG インターフェイスが有効になり (同時に外部 JTAG インターフェイスが無効にります) 、ポートをロジック Low にすると、内部 JTAG インターフェイスが無効になります (同時に外部 JTAG インターフェイスが有効になります) 。
.tckcore()	入力	tck コア信号
.tdicore()	入力	tdi コア信号
.tmscore()	入力	tms コア信号
.tdocore()	出力	tdo コア信号
.tck()	入力	tck ピン信号
.tdi()	入力	tdi ピン信号
.tms()	入力	tms ピン信号
.tdo()	出力	tdo ピン信号
.clkdruser()	入力/出力	これらのポートは、内部 JTAG インターフェイスから JTAG セキュアモードを有効にするためは使用されないため、未接続のままにできます。
.runidleuser()		
.shiftuser()		
.tckutap()		
.tdiutap()		
.tdouser()		
.tdoutap()		
.tmsutap()		

continued...

ポート	入力/出力	機能
.updateuser()		
.usr1user()		
.ntdopinena()		

3.9.3. LOCK および UNLOCK JTAG 命令の実行

JTAG セキュアモードが有効にされている状態で、このリファレンス・デザインをインテル MAX 10 デバイスにコンフィグレーションすると、デバイスが起動後およびコンフィグレーション後に JTAG セキュアモードになります。

JTAG セキュアモードを無効にするには、ユーザーロジックの start_unlock ポートをトリガーし、UNLOCK JTAG 命令を発行します。UNLOCK JTAG 命令の発行後、デバイスは JTAG セキュアモードを終了します。JTAG セキュアモードが無効である場合、インテル MAX 10 デバイスの内部フラッシュをフルチップ消去することで、JTAG セキュアモードを永久に無効にすることができます。

ユーザーロジックの start_lock ポートは、LOCK JTAG 命令の実行をトリガーします。この命令の実行すると、インテル MAX 10 デバイスで JTAG セキュアモードが有効になります。

図 -16: LOCK または UNLOCK JTAG 命令の実行

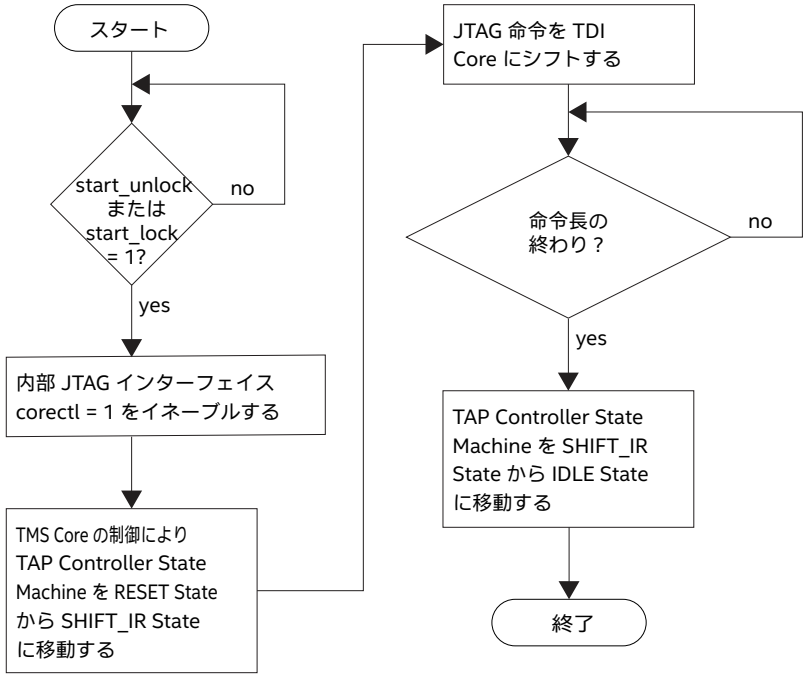


表 33. ユーザーロジックの入力および出力ポート

ポート	入力/出力	機能
clk_in	入力	ユーザーロジックのクロックソースです。ユーザーロジックの f_{MAX} は、タイミング・クロージャー解析に依存します。f_{MAX} を決定するには、タイミング制約を適用し、パスにタイミング解析を実行する必要があります。
start_lock	入力	内部 JTAG インターフェイスに LOCK JTAG 命令の実行をトリガーします。トリガーするには、パルス信号を 1 クロックサイクル以上 High にします。
start_unlock	入力	内部 JTAG インターフェイスに UNLOCKJTAG 命令の実行をトリガーします。トリガーするには、パルス信号を 1 クロックサイクル以上 High にします。
jtag_core_en_out	出力	JTAG WYSIWYG アトムへの出力です。このポートは、内部 JTAG インターフェイスを有効にする目的で、JTAG WYSIWYG アトムの corectl ポートに接続されています。
tck_out	出力	JTAG WYSIWYG アトムへの出力です。このポートは JTAG WYSIWYG アトムの tck_core ポートに接続されています。
tdi_out	出力	JTAG WYSIWYG アトムへの出力です。このポートは JTAG WYSIWYG アトムの tdi_core ポートに接続されています。
tms_out	出力	JTAG WYSIWYG アトムへの出力です。このポートは JTAG WYSIWYG アトムの tms_core ポートに接続されています。
indicator	出力	この出力ピンがロジック High の場合、LOCK または UNLOCKJTAG 命令の実行が完了したことを示します。

3.9.4. JTAG セキュアモードの検証

必須ではない JTAG 命令を実行することにより、デバイスが正しく JTAG セキュアモードに入ったか、または JTAG セキュアモードから抜け出たかを検証することができます。

注意: デバイスが JTAG セキュアモードの場合、外部 JTAG のロックを解除するには、内部 JTAG インターフェイスをインスタンス化する必要があります。

JTAG セキュアオプションを有効にすると、インテル MAX 10 デバイスはパワーアップ後に JTAG セキュアモードに入ります。デザイン例で JTAG セキュア機能を検証するには、以下の手順を実行します。

1. JTAG セキュアモードが有効にされている状態で、リファレンス・デザインの .pof ファイルをデバイスに設定します。電源の再投入後に、デバイスは JTAG セキュアモードとなります。
2. デバイスが正しくユーザーモードに入ったかを確認するには、以下のいずれかを観察します。
   - CONFDONE ピンが High のままである
   - counter_output ピンがトグルを開始する
3. 外部 JTAG ピンを使用して、PULSE_NCONFIG JTAG 命令を発行し、デバイスをリコンフィグレーションします。デザイン例に添付された pulse_ncfg.jam ファイルを使用することができます。この pulse_ncfg.jam ファイルを、quartus_jli または JAM player を通じて実行できます。以下のいずれかを観察することによって、デバイスはリコンフィグレーションしていないことを確認できます。
   - CONFDONE ピンが High でとどまっている
   - counter_output ピンが継続してトグルする

   リコンフィグレーションが成功しないということは、デバイスが現在 JTAG セキュアモードであることを意味します。
4. ユーザーロジックの start_unlock ポートをロジック High にして、UNLOCK JTAG 命令を実行します。

UNLOCK JTAG 命令の完了後に、インジケーター・ポートが High になります。

5. 外部 JTAG ピンを使用して、PULSE_NCONFIGJTAG 命令を発行し、デバイスをリコンフィグレーションします。以下のいずれかを観察することによって、デバイスが正しくリコンフィグレーションしたことを確認できます。
   - CONFDONE ピンが Low になる
   - counter_output ピンがトグルを停止する

   コンフィグレーションが成功したということは、デバイスが現在 JTAG セキュアモードではないという証左となります。

4. インテル MAX 10FPGA コンフィグレーション IP コア実装ガイド

関連情報

- Introduction to Intel FPGA IP Cores
  すべての Intel FPGA IP コアに関する基本的な情報を提供しています。これには、IP コアのパラメーター化、生成、アップグレード、シミュレーションが含まれます。
- Creating Version-Independent IP and Qsys Simulation Scripts
  ソフトウェアまたは IP のバージョンのアップグレードに対して手動更新が不要なシミュレーション・スクリプトを作成します。
- Project Management Best Practices
  プロジェクトと IP ファイルの効率的な管理と移植性について説明するガイドラインです。

4.1. Unique Chip ID インテル FPGA IP コア

本項では、Unique Chip ID インテル FPGA IP コアの実装に向けてのガイドラインについて説明します。

関連情報

- Unique Chip ID (21 ページ)
- Unique Chip ID インテル FPGA IP コアのポート (63 ページ)

4.1.1. Unique Chip ID インテル FPGA IP コアのインスタンス化

Unique Chip ID インテル FPGA IPIP コアをインスタンス化するには、以下の手順を実行します。

1. インテル Quartus Prime 開発ソフトウェアの Tools メニューで、**IP Catalog** をクリックします。
2. Library カテゴリーで、Basic Functions と Configuration and Programming を展開します。
3. **Unique Chip インテル FPGA IP** を選択し、**Add** をクリックして、使用する出力ファイル名を入力します。
4. Save IP Variation ダイアログボックスで、
   - IP バリエーション・ファイル名とディレクトリーを設定します。
   - IP バリエーション・ファイルタイプを選択します。
5. **Finish** をクリックします。

4.1.2. Unique Chip ID インテル FPGA IP コアのリセット

Unique Chip ID インテル FPGA IP コアをリセットするには、reset 信号を少なくとも 1 クロックサイクル以上 High にアサートする必要があります。reset 信号をディアサートした後、Unique Chip ID インテル FPGA IP コアはヒューズ ID ブロックからデバイスの Unique Chip ID を再度読み出します。Unique Chip ID インテル FPGA IP コアは、動作が完了すると data_valid 信号をアサートします。

4.2. デュアル・コンフィグレーション インテル FPGA IP コア

本項では、デュアル・コンフィグレーション インテル FPGA IP コアを実装するためのガイドラインについて説明します。

4.2.1. デュアル・コンフィグレーション インテル FPGA IP コアのインスタンス化

デュアル・コンフィグレーション インテル FPGA IP コアをインスタンス化するには、次の手順を実行します。

1. インテル Quartus Prime 開発ソフトウェアの Tools メニューで、**IP Catalog** をクリックします。
2. Library カテゴリーで、Basic Functions と Configuration and Programming を展開します。
3. **Dual Configuration インテル FPGA IP** を選択し、**Add** をクリックすると、IP Parameter Editor が表示されます。
4. New IP Instance ダイアログボックスで、
   - IP のトップレベル名を設定します。
   - デバイスファミリーを選択します。
   - デバイスを選択します。
5. **OK** をクリックします。

5. デュアル・コンフィグレーション インテル FPGA IP コアのリファレンス

関連情報

- デュアル・コンフィグレーション インテル FPGA IP コア (19 ページ)
- ユーザーロジックを介したリモート・システム・アップグレードへのアクセス (42 ページ)
- AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor
  インテル® MAX® 10 FPGA デバイスのリモート・システム・アップグレード向けリファレンス・デザインを提供します。
- I2C Remote System Update Example
  この例では、I2C プロトコルの使用によるリモート・システム・アップグレードを示します。

5.1. デュアル・コンフィグレーション インテル FPGA IP コアの Avalon-MM アドレスマップ

表 34. インテル MAX 10 デバイスに向けたデュアル・コンフィグレーション インテル FPGA IP コアの Avalon-MM アドレスマップ

- インテルでは、書き込み動作向けの予約ビットを 0 に設定することを推奨しています。読み出し動作では、IP コアは常に出力として 0 を生成します。
- 1 を書き込むことで、記述した任意の動作をトリガーします。
- オフセット 4、5、6、7 の任意の読み出し動作の前に、必要な動作をオフセット 2 からトリガーする必要があります。

オフセット	R/W	幅 (ビット)	説明
0	W	32	• ビット 0 — リコンフィグレーションをトリガーします。 • ビット 1 — ウォッチドッグ・タイマーをリセットします。 • ビット 31:2 — 予約済 信号は、Avalon® の同じ書き込みサイクルと同時にトリガーされます。
1	W	32	• ビット 0 — 入力レジスターに対して config_sel_overwrite をトリガーします。 • ビット 1 — 入力レジスターに config_sel を書き込みます。コンフィグレーション・イメージ 0 または 1 をロードするには 0 または 1 を設定します。 • ビット 31:2 — 予約済 busy 信号は書き込みサイクルの直後、コンフィグレーション・イメージの情報がレジスターに格納されている間に生成されます。busy 信号が High になると、処理が完了して busy 信号がディアサートされるまで、このアドレスへの書き込みは無視されます。
2	W	32	• ビット 0 — ユーザー・ウォッチドッグからの読み出し動作をトリガーします。 • ビット 1 — 前回のステート・アプリケーション 2 レジスターからの読み出し動作をトリガーします。 • ビット 2 — 前回のステート・アプリケーション 1 レジスターからの読み出し動作をトリガーします。 • ビット 3 — 入力レジスターからの読み出し動作をトリガーします。 • ビット 31:4 — 予約済 busy 信号が書き込みサイクル直後に生成されます。

continued...

オフセット	R/W	幅 (ビット)	説明
3	R	32	• ビット 0 — IP busy 信号です。 • ビット 31:1 — 予約済 busy 信号は、デュアル・コンフィグレーション インテル FPGA IP コアが書き込みまたは読み出し処理中であることを示します。この状態では、リモート・システム・アップデート・ブロック・レジスターの動作要求に向けたすべての書き込み動作は、リセットタイマーへのトリガーを除き無視されます。インテルでは、読み出しまたは書き込みプロセスをトリガーした後、この busy 信号をポーリングすることを推奨しています。
4	R	32	• Bit 11:0 — ユーザー・ウォッチドッグの値です。(13) • ビット 12 — ユーザー・ウォッチドッグの現在の状態 • ビット 16:13 — 現在の状態の msm_cs 値です。 • ビット 31:17 — 予約
5	R	32	• ビット 3:0 — *リモート・システム・アップグレード・ステータス・レジスター — MAX 10 デバイスの前回のステート・ロジック・ビット*の表からの前回のステート・アプリケーション 1 リコンフィグレーション・ソースの値です。 • ビット 7:4 — 前回のステート・アプリケーション 1 の msm_cs 値です。 • ビット 31:8 — 予約済
6	R	32	• ビット 3:0 — *リモート・システム・アップグレード・ステータス・レジスター — MAX 10 デバイスの前回のステート・ロジック・ビット*の表からの前回のステート・アプリケーション 2 リコンフィグレーション・ソースの値です。 • ビット 7:4 — 前回のステート・アプリケーション 2 の msm_cs 値です。 • ビット 31:8 — 予約済
7	R	32	• ビット 0 — 入力レジスターからの config_sel_overwrite 値です。 • ビット 1 — 入力レジスターの config_sel 値(14) • ビット 31:2 — 予約済

関連情報

- デュアル・コンフィグレーション インテル FPGA IP コア (19 ページ)
- Avalon Interface Specifications
  デュアル・コンフィグレーション・インテル IP コアに用いる Avalon-MM インターフェイスの仕様について詳しい情報を提供します。
- デュアル・コンフィグレーション インテル FPGA IP コアのインスタンス化 (59 ページ)
- リモート・システム・アップグレード・ステータス・レジスター (17 ページ)
  リモート・システム・アップグレード・ステータスレジスター — インテル® MAX® 10 デバイスの前回のステート・ロジック・ビットの表で前回のステート・アプリケーション・リコンフィグレーション・ソースについて詳しい情報を提供します。

(13) デュアル・コンフィグレーション IP コアを使用して、29 ビットのユーザー・ウォッチドッグ値のうち上位 12 ビットのみを読み出しすることができます。

(14) 入力レジスターの config_sel 読み出しのみを行います。物理的な CONFIG_SEL ピン設定には影響しません。

5.2. デュアル・コンフィグレーション インテル FPGA IP コアのパラメーター

表 35. インテル MAX 10 のデュアル・コンフィグレーション インテル FPGA IP コアのパラメーター

パラメーター	値	説明
Clock frequency	最大 80 MHz	RU_nRSTIMER 信号と RU_nCONFIG 信号をアサートするサイクル数を指定します。最大 RU_CLK が 40 MHz の場合、デュアル・コンフィグレーション インテル FPGA IP コアの動作制限は、ハードウェアの制限の 2 倍に等しい最大 80 MHz となります。これはデュアル・コンフィグレーション インテル FPGA IP コアが入力周波数の半分のレートで RU_CLK が生成されることが原因です。

6. Unique Chip ID インテル FPGA IP コアのリファレンス

6.1. Unique Chip ID インテル FPGA IP コアのポート

表 36. Unique Chip ID インテル FPGA IP コアのポート

ポート	入力/出力	幅 (ビット)	説明
clkin	入力	1	• Unique Chip ID ブロックにクロック信号を供給します。サポートされる最大周波数は 100 MHz です。 • クロック信号を供給すると、IP コアが Unique Chip ID の値を読み出して、その値を chip_id 出力ポートに送信します。
reset	入力	1	• reset 信号を少なくとも 1 クロックサイクル以上 High にアサートすると、IP コアがリセットされます。 • chip_id[63:0]出力ポートは、デバイスをリコンフィグレーションするか、IP コアをリセットするまで Unique Chip ID の値を保持します。
data_valid	出力	1	• Unique Chip ID が取得の準備が整っていることを示します。この信号が Low の場合、IP コアは初期状態であるか、またはヒューズ ID からデータをロード中です。 • IP コアがこの信号をアサートした後であれば、データは chip_id[63..0]出力ポートで取得するにあたっての準備が整っています。
chip_id	出力	64	• 対応するヒューズ ID の位置に応じた Unique Chip ID を示します。データは IP コアが data_valid 信号をアサートした後にのみ有効です。 • 起動時の値は、0 にリセットされます。

7. MAX 10 FPGA コンフィグレーション・ユーザーガイド 改訂履歴

ドキュメント・バージョン	変更内容
2019.01.07	• 以下のトピックの手順を更新しました。 — *兼用ピンのイネーブル* — *内部コンフィグレーション・モードの選択* — *自動生成.pof* — *Convert Programming Files を使用した.pof の生成* — *内部フラッシュへの.pof のプログラミング* — *エラー検出の有効化* — *デザインのコンパイルの前に圧縮を有効にする場合* — *デザインのコンパイル後に圧縮を有効にする場合* • *Convert Programming Files を使用した.pof の生成*の手順 4b の注を更新しました。 • *ユーザーロジックを介したリモート・システム・アップグレードへのアクセス*に注を追加しました。 • インテルへのブランド変更に伴い、次の IP コア名を変更しました。 — 「アルテラ・デュアル・コンフィグレーション IP コア」から「デュアル・コンフィグレーション・インテル FPGA IP コア」に変更しました。 — 「アルテラ Unique Chip ID IP コア」から「Unique Chip ID Intel FPGA IP コア」に変更しました。
2018.10.29	• 表「*インテル MAX 10 デバイス向け ICB 値およびその概要*」の脚注の JTAG セキュア機能を更新しました。 • *ユーザー・ウォッチドッグ・タイマー*の説明を更新しました。 • *Convert Programming Files を使用した.pof の生成*の JTAG セキュアオプションについての注を更新しました。 • *.ekp ファイルおよび暗号化コンフィグレーション・ファイルの生成*の手順 5 の注を更新しました。 • *兼用ピンのイネーブル*の手順 4 に注を追加しました。 • 図「*インテル MAX 10 デバイスのコンフィグレーション・シーケンス*を更新し、Read ICB Settings ブロックおよび Read ICB Settings ブロックへの注を追加しました。 • 図「*インテル MAX 10 デバイスの兼用コンフィグレーション・ピンのガイドライン*」で、JTAG ピンについてのガイドラインを更新しました。 • 図「*ダウンロード・ケーブルを使用した JTAG シングル・デバイス・コンフィグレーションの接続セットアップ*」を更新しました。
2018.06.01	表「*インテル MAX 10 デバイスの最小および最大エラー検出周波数*」に、n の有効値として 1 を追加しました。
2018.02.12	サードパーティー製プログラミング・ツール・ファイル（.jbc、.jam、および.svf）の生成に手順を追加しました。

日付	バージョン	変更内容
2017 年 7 月	2017.07.20	• 図「*インテル MAX 10 デバイスの JTAG コンフィグレーションおよび内部コンフィグレーションの上位レベルの概要*」のコンフィグレーション・フラッシュ・メモリーに向けた CFM 用語を更新しました。 • バウンダリー・スキャン・テストに BST の定義を追加しました。
2017 年 6 月	2017.06.15	*エラー検出機能の検証*のセクションで CRC エラーをクリアし、元の CRC 値を復元する方法を更新しました。
2017 年 4 月	2017.04.06	ユーザー・インターフェイスの更新に伴い、*Auto-recofigure from secondary image when initial image fails (enabled by default)*オプションを *Configure device from CFM0 only* に変更しました。
2017 年 2 月	2017.02.21	ブランド名をインテルに変更しました。

continued...

日付	バージョン	変更内容
2016 年 10 月	2016.10.31	• *電圧オーバーシュートの防止*の概要を更新しました。 • 図「*ダウンロード・ケーブルを使用した JTAG シングル・デバイス・コンフィグレーションの接続セットアップ*」と図「*ダウンロード・ケーブルを使用した JTAG マルチデバイス・コンフィグレーションの接続セットアップ*」の注を更新しました。 • ISP クランプ機能を実装するための手順を追加しました。 • UFM セクターを追加するために図「*アナログ機能オプションおよびフラッシュ機能オプションでのすべてのインテル MAX 10 デバイスのコンフィグレーション・フラッシュメモリー・セクターの使用方法*」を更新しました。
2016 年 5 月	2016.05.13	• インテル Quartus Prime GUI を反映するために標準 POR のインスタンスを低速の POR に変更しました。 • t_{CFG} を $t_{RU\_nCONFIG}$ に更新しました。 • *.ekp ファイルと暗号化された.pof ファイルを個別にプログラミングする*において、手順 8 のファイルタイプを .ekp から .pof に修正しました。 • 表「*インテル MAX 10 デバイス向け ICB 値およびその概要*」の Use secondary image ISP data as default setting when available についての記述を修正しました。 • CFM のプログラミング時間を修正しました。 • JTAG ピンの共有を使用する際の JTAG ピンの要件に関する注を追加しました。 • *JTAG ピンの共有をガイドライン: 兼用コンフィグレーション・ピン*の下に移植しました。 • 「コンフィグレーション RAM ビットをクリア」を Power Up ステートから Reset ステートに移動したことに伴い、コンフィグレーション・シーケンスの図を更新しました。 • <crcblock_name>に向けたエラー検出ポートの入力/出力を、入力からなしに修正しました。 • ユーザー・インターフェイスとポートの定義を介したリモート・システム・アップグレード・アクセス例を追加しました。 • *エラー検出周波数*および*巡回冗長検査の計算のタイミング*の暫定用語を削除しました。 • 図「*ダウンロード・ケーブルを使用した JTAG マルチデバイス・コンフィグレーションの接続セットアップ*」を追加しました。 • 図「*ダウンロード・ケーブルを使用した JTAG シングル・デバイス・コンフィグレーションの接続セットアップ*」を更新しました。 • 新しい JTAG セキュアデザイン例を追加しました。 • リモート・システム・アップグレードの項題から「デュアル・イメージ・コンフィグレーション」を削除しました。 • 表「*インテル MAX 10 デバイスでモニタリングされる電源ランプ時間要件*」を更新しました。 • *内部コンフィグレーション時間*を追加しました。 • インスタント・オン機能を削除しました。 • 図「*アナログ機能オプションおよびフラッシュ機能オプションでのすべての MAX 10 デバイスのコンフィグレーション・フラッシュメモリー・セクターの使用方法*」で追加の UFM に向けたユーザー・フラッシュ・メモリーを更新しました。
2015 年 12 月	2015.12.14	• *Set I/O to weak pull-up prior usermode* オプションの ICB 設定の概要を更新し、ウィークプルアップがコンフィグレーション時にイネーブルされることを明記しました。 • *ユーザーインターフェイスを介したリモート・システム・アップデート・ブロックへのアクセス*を削除しました。 • エラー検出の WYSIWYG アトムでの入出力ポートの定義を追加しました。 • リコンフィグレーションの際に ICB ビット設定に基づく I/O ピンの状態を更新しました。
2015 年 11 月	2015.11.02	• JTAG コンフィグレーション向けの JRunner サポートおよび AN 414 へのリンクを削除しました。 • デバイスの機能オプションごとの内部コンフィグレーション・モード・サポートの違いを表で更新しました。 • 内容の重複により、圧縮されたコンフィグレーション・イメージの最大数の表を削除しました。

continued...

日付	バージョン	変更内容
		• Quartus Prime 15.1 のアップデートに伴い、MAX 10 デバイスの初期化コンフィグレーション・ビットの設定および説明を更新しました。 • Quartus Prime 15.1 のアップデートに伴い、**Enable JTAG pin sharing** および **Enable nCONFIG, nSTATUS, and CONF_DONE pins** を更新しました。 • ISP クランプ機能についての情報を追加しました。 • Raw Programming Data (.rpd) 生成手順の情報を更新しました。 • 項題を「*コンフィグレーション・フラッシュメモリーの合計プログラミング時間*」から「*コンフィグレーション・フラッシュメモリーのプログラミング時間*」に変更しました。 • 表題を「*インテル MAX 10 デバイスでのコンフィグレーション・フラッシュメモリーのセクターに対する合計プログラミング時間*」から「*インテル MAX 10 デバイスでのコンフィグレーション・フラッシュメモリーのセクターに対するプログラミング時間*」に変更しました。 • 表「*インテル MAX 10 デバイスでのコンフィグレーション・フラッシュメモリーのセクターに対するプログラミング時間*」に注を追加しました。 • 内部 JTAG インターフェイスおよびユーザーインターフェイスを介した内部 JTAG ブロックへのアクセスに関する情報を追加しました。 • インテル MAX 10 JTAG セキュアデザイン例を追加しました。
2015 年 6 月	2015.06.15	• *アルテラ・デュアル・コンフィグレーション IP コアの参考資料* と *デュアル圧縮イメージでのリモート・システム・アップグレード* の関連情報に AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor へのリンクを追加しました。 • 表「*インテル MAX 10 デバイスのリモート・システム・アップグレード回路の信号*」で RU_nRSTIMER にパルス保持の要件を追加しました。 • 表「*インテル MAX 10 デバイスのアルテラ・デュアル・コンフィグレーション IP コアの Avalon-MM アドレスマップ*」の関連情報として、表「*リモート・システム・アップグレード・ステータスレジスター — インテル MAX 10 デバイスの前回のステートビット*」のリンクを追加しました。
2015 年 5 月	2015.05.04	• 表「MAX 10 デバイスの初期化コンフィグレーション・ビット」を再編し、コンフィグレーションの設定の項目を更新しました。 • 図「MAX 10 デバイスの内部コンフィグレーションの概要」に JTAG コンフィグレーションを追加し、図を「コンフィグレーション・スキーム」の項に移動しました。 • 「MAX 10 デバイスの初期化コンフィグレーション・ビット」の表に、関連するコンフィグレーション設定を説明するリンクを追加しました。 • 表「MAX 10 デバイスの初期化コンフィグレーション・ビット」において、デフォルトのウォッチドッグ・タイマー値を 16 進数から 10 進数の値に更新しました。 • 表「MAX 10 デバイスの初期化コンフィグレーション・ビット」で ISP データの説明を更新しました。 • 「ユーザー・ウォッチドッグ・タイマー」の項にウォッチドッグ・タイマーの計算式を追加しました。 • 「User Watchdog Internal Circuitry Timing Specifications」 へのリンクを追加しました。 • 表「MAX 10 デバイスの初期化コンフィグレーション・ビット」に JTAG セキュアがデフォルトで無効にされており、有効にするにはアルテラのサポートが必要であることを示す注を追加しました。 • 最小および最大 CRC 演算時間の除数を 2 に更新しました。 • リモート・システム・アップグレードのフローの図を更新しました。 • 表「内部コンフィグレーションでの暗号化」に「キー」の用語を追加し、イメージ 1 とイメージ 2 をそれぞれイメージ 0 とイメージ 1 に変更しました。 • 表「内部コンフィグレーションでの暗号化」に、片方のコンフィグレーションが失敗するともう片方が自動的にロードされることを示す注を追加しました。 • 除数が 2 以外の場合の最小および最大 CRC 演算時間の計算式を追加しました。 • JTAG セキュアがオンになっている場合の注意を追加しました。 • 特定の内部コンフィグレーション・モード向けに自動的に生成される.pof に関する情報を追加しました。

continued...

日付	バージョン	変更内容
		• Device and Pin Options と Convert Programming Files を用いる.pof と ICB の設定を追加しました。 • 「概要」にコンフィグレーション RAM (CRAM) を追加しました。 • 編集上の修正を行いました。
2014 年 12 月	2014.12.15	• BOOT_SEL ピンの名称を CONFIG_SEL ピンに変更しました。 • アルテラ IP コア名を Dual Boot IP コアからアルテラ Dual Configuration IP コアに更新しました。 • ICB の要素としての AES 暗号化キーについての情報を追加しました。 • 暗号化機能のガイドラインを追加しました。 • 14.1 リリースで使用可能になった ICB 設定オプションを追加しました。 • 14.1 リリースで使用可能になった CFM プログラミング機能でのプログラマーのオプションを追加しました。
2014 年 9 月	2014.09.22	初版