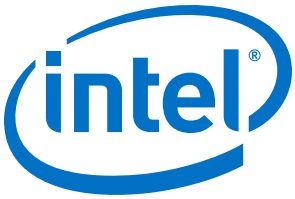


AN 539: インテル® FPGA デバイスにおける CRC を使用したエラー検出および回復のテスト方法

更新情報
フィードバック

AN-539 | 2019.08.09
最新版をウェブからダウンロード: PDF | HTML

目次

1. インテル® FPGA デバイスにおける CRC を使用したエラー検出および回復のテスト方法

本アプリケーション・ノートでは、Arria® II、Stratix® III、Stratix IV、Arria V、Cyclone® V、および Stratix V デバイスの強化されたエラー検出巡回冗長検査 (CRC) 機能の使用方法について説明します。また、サポートされているデバイスでこの機能を検証する際に使用可能なテスト方法についても説明します。Arria V、Cyclone V、および Stratix V デバイスはまた、エラー訂正機能をサポートしています。

FPGA コンフィグレーションの実行時に、ビットストリームが外部デバイスから FPGA に転送されると、エラー検出 CRC 機能はコンフィグレーション・ビットストリームの破損を検出します。ユーザーモードでは、エラー検出 CRC 機能は SEU (シングル・イベント・アップセット) を検出し、エラーのタイプおよびその位置を特定します。さらに、Arria V、Cyclone V、および Stratix V デバイスは、ユーザーモードで検出されたエラーを訂正する機能である内部スクラブもサポートしています。

関連情報

- SEU Mitigation in Arria II Devices
- SEU Mitigation in the Cyclone III Device Family
- SEU Mitigation in Stratix III Devices
- SEU Mitigation in Stratix IV Devices
- SEU Mitigation for Arria V Devices
- SEU Mitigation for Cyclone V Devices
- SEU Mitigation for Stratix V Devices
- SEU Mitigation in Intel Cyclone 10 LP Devices

1.1. 機能の説明

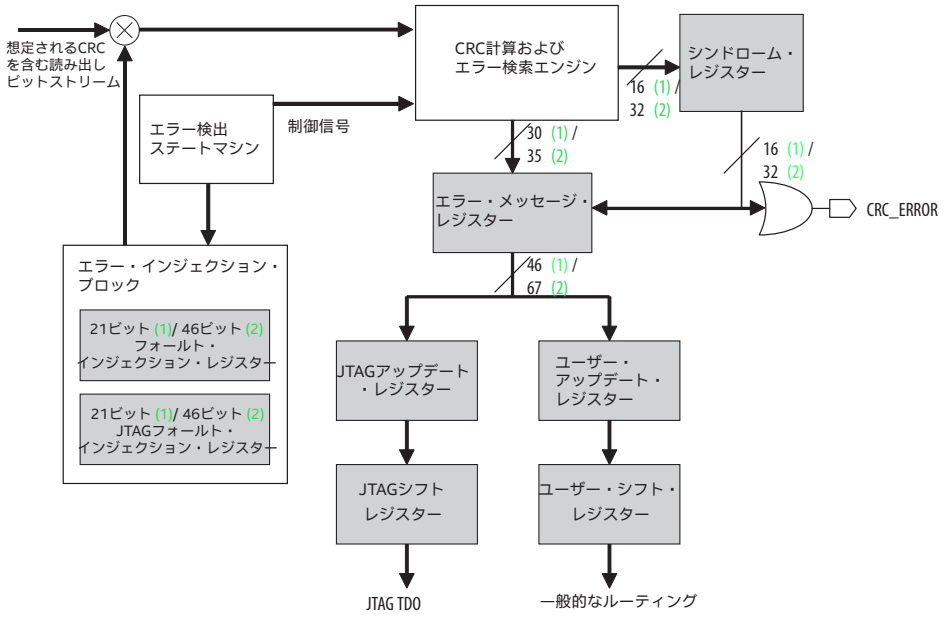
エラー検出 CRC 機能を使用すると、コンフィグレーション・モードおよびユーザーモードにおいて FPGA コンフィグレーション・ビットのエラーを検出することが可能です。

CRC のフィールド幅は、Arria II、Stratix III、および Stratix IV デバイスでは 16 ビットであり、Arria V、Cyclone V、および Stratix V デバイスでは 32 ビットです。

図 -1: エラー検出のブロック図

次の図は、エラー検出回路、シンドローム・レジスター、およびエラー・インジェクション・ブロックを表しています。

1. Arria II、Stratix III、および Stratix IV デバイスにのみ適用されます
2. Arria V、Cyclone V、および Stratix V デバイスにのみ適用されます



レジスター	説明
シンドローム・レジスター	このレジスターには、エラー検出検証サイクルにおける現在のフレームの CRC シグネチャーが含まれています。CRC_ERROR 信号は、このレジスターの内容から生成されます。
エラー・メッセージ・レジスター (EMR)	このレジスターには、エラータイプ、エラーの位置、および実際の症状に関する情報が含まれています。報告されるエラータイプおよびエラーの位置は、シングルビット・エラーおよびダブル隣接ビットエラーです。Arria V、Cyclone V、および Stratix V デバイスの場合、報告されるエラータイプおよびエラーの位置は、シングル、ダブル、トリプルおよびクアドルプルの隣接ビットエラーです。その他のエラータイプの位置のビットは、EMR では特定されません。このレジスターの内容は SHIFT_EDERROR_REG JTAG 命令でシフト可能です。またはコア・インターフェイスを介してコアへシフト可能です。エラー・メッセージ・レジスターの詳細については、関連情報を参照してください。
JTAG アップデート・レジスター	このレジスターは、EMR レジスターの内容が確定した 1 サイクル後に EMR の内容で自動的に更新されます。JTAG アップデート・レジスターには、JTAG シフトレジスターにサンプリングされる前にアサートされる必要のあるクロックイネーブルが含まれます。この要件により、JTAG アップデート・レジスターへの EMR の内容の書き込みと、JTAG シフトレジスターによるその内容の読み出しが同時に発生しないことが保証されます。
ユーザー・アップデート・レジスター	このレジスターは、EMR レジスターの内容が確定した 1 サイクル後に EMR の内容で自動的に更新されます。ユーザー・アップデート・レジスターには、ユーザー・シフト・レジスターにサンプリングされる前にアサートされる必要のあるクロックイネーブルが含まれます。この要件により、ユーザー・アップデート・レジスターへの EMR の内容の書き込みと、ユーザー・シフト・レジスターによるその内容の読み出しが同時に発生しないことが保証されます。
JTAG シフトレジスター	このレジスターには JTAG インターフェイスからアクセス可能で、SHIFT_EDERROR_REG JTAG 命令によって JTAG アップデート・レジスターの内容をサンプリングし、読み出すことができます。
ユーザー・シフト・レジスター	このレジスターにはコアロジックからアクセス可能で、ユーザーロジックによってユーザー・アップデート・レジスターの内容をサンプリングし、読み出すことができます。
JTAG フォールト・インジェクション・レジスター	このレジスターは、EDERROR_INJECT JTAG 命令により完全に制御されます。このレジスターは、ビットストリーム内に希望するエラー・インジェクションの情報を保持します。
フォールト・インジェクション・レジスター	JTAG フォールト・インジェクション・レジスターがアップデートされる際、JTAG フォールト・インジェクション・レジスターの内容がこのフォールト・インジェクション・レジスターにロードされます。

1.1.1. コンフィグレーション・エラーの検出

インテル® Quartus® Prime 開発ソフトウェアはコンフィグレーション・ビットストリームを生成する際に、各データフレームに対して CRC 値も計算します。コンフィグレーション・ビットストリーム内のデータフレーム数とフレーム長は、各デバイスで異なります。そのため、コンフィグレーション・ビットストリームには、ビットストリーム内のデータフレーム数に応じて複数の CRC 値が含まれている場合があります。

コンフィグレーション中にデータフレームが FPGA にロードされると、あらかじめ計算された CRC 値が FPGA 内の CRC 回路にシフトされます。同時に、FPGA 内の CRC エンジンは、受信したデータフレームの CRC 値を計算し、その CRC 値をコンフィグレーション RAM (CRAM) に格納します。あらかじめ計算された CRC 値は、CRC エンジンによって計算された CRC 値と比較されます。CRC 値が一致しない場合、nSTATUS は Low に設定され、コンフィグレーション・エラーを通知します。

コンフィグレーション・エラー検出機能の能力は、コンフィグレーション・ビットストリームを変更する、もしくはコンフィグレーション中にコンフィグレーション・ビットストリームを意図的に破壊することにより、テストすることができます。

1.1.2. ユーザーモードでのエラー検出

ユーザーモード中に、コンフィグレーションされた CRAM ビットの内容がソフトエラーの影響を受けることがあります。ユーザーモードのエラー検出では、ソフトエラーのタイプを判断し、影響を受けているビットの位置を特定することができます。

コンフィグレーション・サイクルの完了後、コンフィグレーションされた CRAM ビットの CRC 値がエラー検出機能によって計算され、その結果があらかじめ計算された CRC 値と比較されます。この 2 つの CRC 値が一致する場合、結果の CRC シグネチャーは 0 に設定され、エラーが検出されなかったことを示します。このエラー検出プロセスは、nCONFIG が Low に設定されてデバイスがリセットされるまで継続します。

CRC エラーが発生した場合、結果のシグネチャーは 0 以外の値になり、CRC_ERROR ピンが High に設定されてエラーを通知します。エラーが発生すると、エラー検出ステートマシン内の検索エンジンはエラータイプとエラーの位置を特定します。フレーム内のすべてのタイプの CRC エラーが検出可能です。検索エンジンの結果は EMR に格納されます。このレジスターの内容は、JTAG 命令またはコア・インターフェイス・ロジックを介してシフトアウトすることができます。エラー検出ブロックの実行中に、この内容をシフトアウトすることができます。

1.1.3. エラー検出ピン

エラー検出機能をイネーブルする場合、CRC_ERROR ピンを専用の出力ピンとして使用します。それ以外の場合は、CRC_ERROR ピンをユーザー I/O ピンとして使用します。このピンを専用の出力ピンとして使用する場合、このピンのアクティブ High 信号は、エラー検出回路がコンフィグレーションされた CRAM ビット内でエラーを検出したことを示します。WYSIWYG 機能をイネーブルする場合、CRC エラー出力は CRC_ERROR ピンへの専用のパスとなります。このピンはまた、インテル Quartus Prime 開発ソフトウェアでオプションをイネーブルすることによりオープンドレイン出力として使用することも可能です。このピンをオープンドレイン出力として使用することで、電圧平準化の利点が得られます。

注意: ユーザーモードのエラー検出では、CRC_ERROR ピンのみを使用します。

関連情報

- SEU Mitigation in Arria II Devices
- Enabling the error detection block in Quartus II software for Cyclone III Device Family

- Enabling the error detection block in Quartus II software for Stratix III Devices
- Enabling the error detection block in Quartus II software for Stratix IV Devices
- Enabling the error detection block in Intel Quartus Prime software for Arria V Devices
- Enabling the error detection block in Intel Quartus Prime software for Cyclone V Devices
- Enabling the error detection block in Intel Quartus Prime software for Stratix V Devices
- SEU Mitigation in Intel Cyclone 10 LP Devices

1.1.4. エラー・メッセージ・レジスター (EMR)

EMR には、エラータイプ、エラー位置、および実際の症状についての情報が含まれています。このレジスターの幅は Arria II、Stratix III、および Stratix IV デバイスでは 46 ビットで、Arria V、Cyclone V、および Stratix V デバイスでは 67 ビットです。表 2 (7 ページ) と表 3 (7 ページ) は報告されるエラーのタイプと位置を表しています。EMR は他のエラータイプに対して位置ビットを特定しません。

エラーの位置は、列のフレーム番号、フレーム内のバイト位置、およびバイト内のビット位置で構成されています。このレジスターの内容は、SHIFT_EDERROR_REG JTAG 命令でシフトアウトすることが可能です。またはコア・インターフェイスを介してコアへシフトアウト可能です。

表 1. SHIFT_EDERROR_REG JTAG 命令

JTAG 命令	命令コード	説明
SHIFT_EDERROR_REG	00 0001 0111	JTAG 命令は、TDI ピンと TDO ピンの間のエラー検出ブロックの JTAG ピンに EMR を接続します。

エラーが発生すると、EMR の内容が更新されます。EMR の内容は、次のエラーメッセージで上書きされる前に転送する必要があります。2 つの EMR の更新の間の最小間隔時間はデバイスによって異なります。EMR の最小更新間隔の詳細については、以下の関連情報を参照してください。

注意: エラー検出の周波数を制御することで、エラー検出プロセスの速度を落とし、EMR の読み出しに十分な時間を確保することができます。

関連情報

- 外部ホストを使用した EMR のアンロード (12 ページ)
- CRC Timing specifications for Arria II Devices
- CRC Timing specifications for Cyclone III Device Family
- CRC Timing specifications for Stratix III Devices
- CRC Timing specifications for Stratix IV Devices
- CRC Timing specifications for Arria V Devices
- CRC Timing specifications for Cyclone V Devices
- CRC Timing specifications for Stratix V Devices
- SEU Mitigation in Intel Cyclone 10 LP Devices

1.1.4.1. Arria II、Stratix III、および Stratix IV デバイスの EMR

図 -2: Arria II、Stratix III、および Stratix IV デバイスの EMR の内容

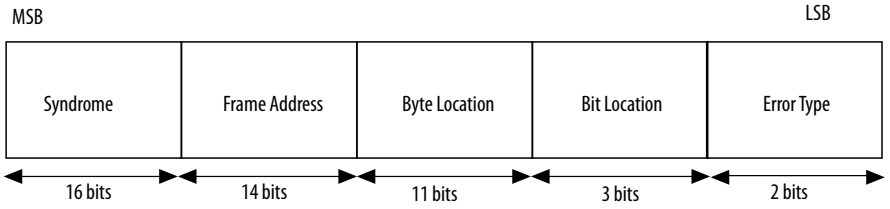


表 2. Arria II、Stratix III、および Stratix IV デバイスの EMR で表現されるエラータイプ

エラータイプ		説明
ビット 1	ビット 0	
0	0	CRC エラーは特定されていません。
0	1	シングルエラーの位置が特定されました。
1	0	ダブル隣接ビットエラーの位置が特定されました。
1	1	3 つ以上のエラーまたは 2 つの隣接しないエラーがあります。

1.1.4.2. Arria V、Cyclone V、および Stratix V デバイスの EMR

図 -3: Arria V、Cyclone V、および Stratix V デバイスの EMR の内容

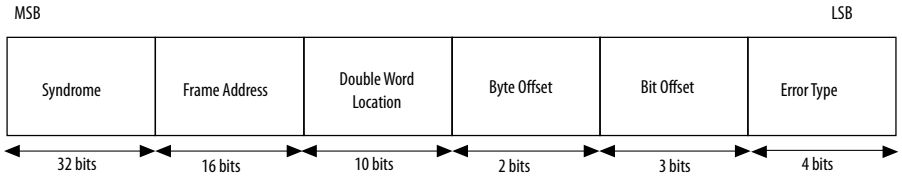


表 3. Arria V、Cyclone V、および Stratix V デバイスの EMR で表現されるエラータイプ

エラータイプ				説明
ビット 4	ビット 3	ビット 1	ビット 0	
0	0	0	0	CRC エラーは特定されていません。
0	0	0	1	シングルエラーの位置が特定されました。
0	0	1	0	ダブル隣接ビットエラーの位置が特定されました。
1	1	1	1	訂正不可能なエラーです。
その他				無効

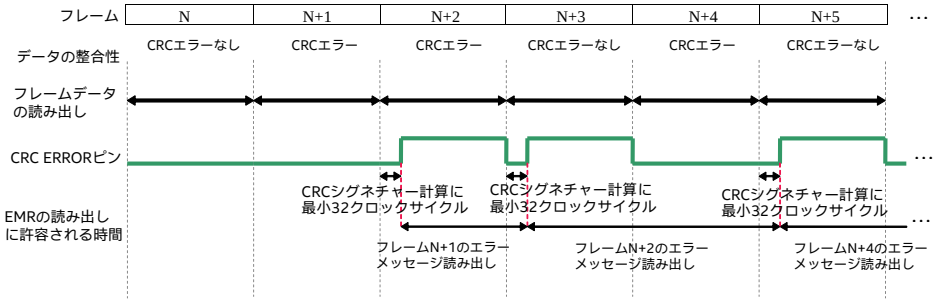
1.1.5. エラー検出のタイミング

CRC_ERROR ピンは、CRC シグネチャーの計算中にかならず少なくとも 32 クロックサイクル Low に設定されます。エラーが発生すると、CRC_ERROR ピンは EMR の更新、あるいは 32 サイクルの経過のいずれか遅い方で High に設定されます。したがって、CRC_ERROR ピンの立ち上がりエッジで EMR の

内容のアンロードを開始することができます。このサイクルの最後に、CRC_ERROR ピンは再度、最小 32 クロックサイクルの間 Low に設定されます。次のフレームにもエラーが含まれている場合、CRC_ERROR ピンは再び High に設定され、EMR は新しいエラー情報で更新されます。

図 -4: CRC_ERROR ピンの動作例

次の図は、データフレーム内でエラーが発生した場合の CRC_ERROR ピンの動作を表す一例です。



エラー検出プロセスの速度は、インテル Quartus Prime 開発ソフトウェアでクロック周波数の分周係数を設定することにより制御することができます。除数は 2 のべき乗であり、この場合の n は 1 から 8 です。除数の範囲は 2 から 256 です。

図 -5: エラー検出周波数の計算式

$$\text{エラー検出の周波数} = \frac{\text{内部オシレーターの周波数}}{N}$$

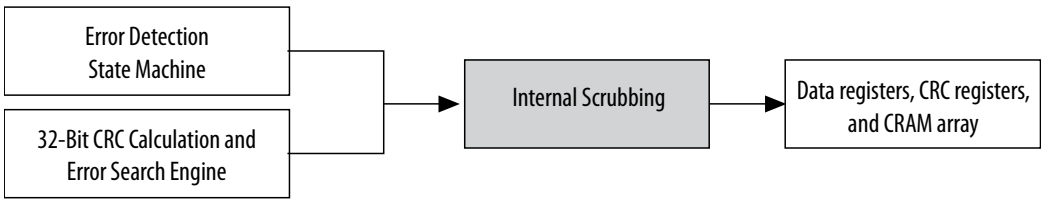
関連情報

- CRC Timing specifications for Arria II Devices
- CRC Timing specifications for Cyclone III Device Family
- CRC Timing specifications for Stratix III Devices
- CRC Timing specifications for Stratix IV Devices
- CRC Timing specifications for Arria V Devices
- CRC Timing specifications for Cyclone V Devices
- CRC Timing specifications for Stratix V Devices
- SEU Mitigation in Intel Cyclone 10 LP Devices

1.2. エラー訂正

エラー検出機能に加え、Arria V、Cyclone V、および Stratix V デバイスは内部スクラブもサポートしています。内部スクラブとは、ソフトエラーを内部で訂正する機能です。これは、フレームごとに実行されます。内部スクラブは、デバイスをリコンフィグレーションすることなくエラーを訂正します。内部スクラブはユーザーモードで動作します。インテル Quartus Prime 開発ソフトウェアで内部スクラブ機能を有効にするには、**Device & Pin Options** ダイアログボックスの **Error Detection CRC** ページで、Enable internal scrubbing オプションをオンにします。

図 -6: 内部スクラブ機能



1.3. エラー検出 CRC 機能の使用

CRC エラーの位置を特定するには、EMR をアンロードします。EMR の内容は、JTAG 命令またはコア・インターフェイスを介してシフトすることができます。

1.3.1. ユーザーロジックを使用したエラー検出

この項では、ユーザーロジックを介してエラー検出機能を使用する方法をデザイン例で説明します。

関連情報

- Design Example for Arria II, Stratix III, and Stratix IV Devices
- Design Example for Arria V, Cyclone V, and Stratix V Devices

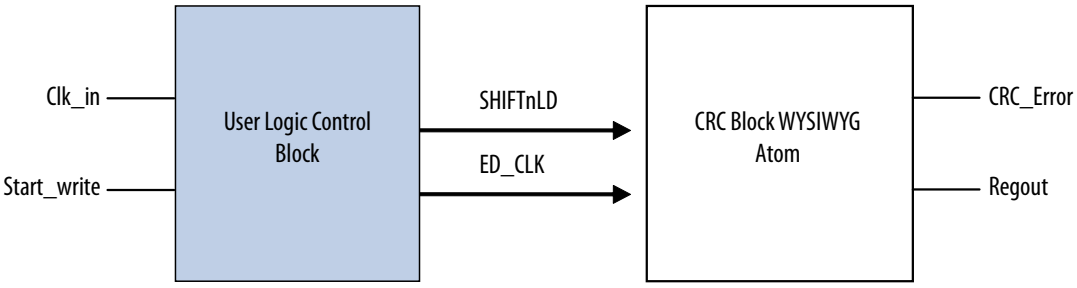
1.3.1.1. CRC_ERROR 出力信号を使用した CRC エラーの検出

CRC エラーを検出するには、CRC_ERROR 出力信号を使用します。EMR の内容をアンロードする前に、CRC 計算で CRC エラーが検出されたかどうかを確認します。これを実行するには、WYSIWYG アトムの `crcerror` ポートを専用の CRC_ERROR ピンまたは任意のユーザー I/O にルーティングします。ユーザー I/O に `crcerror` ポートをルーティングするには、`crcerror` ポートとユーザー I/O の間に D フリップフロップ (DFF) を挿入する必要があります。

1.3.1.2. ユーザーロジックを使用した EMR のアンロード

ユーザーロジックを使用して EMR をアンロードするには、WYSIWYG アトムを使用してユーザーロジックとエラー検出回路の間にインターフェイスを確立します。WYSIWYG アトムは、エラー検出ブロックへのアクセスを提供します。さらに、WYSIWYG アトムへのアクセスに必要となるコントロール信号を制御するユーザーロジック制御ブロックをデザインする必要があります。

図 -7: ユーザーロジック制御ブロックと WYSIWYG アトム間のインターフェイス



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手法です。

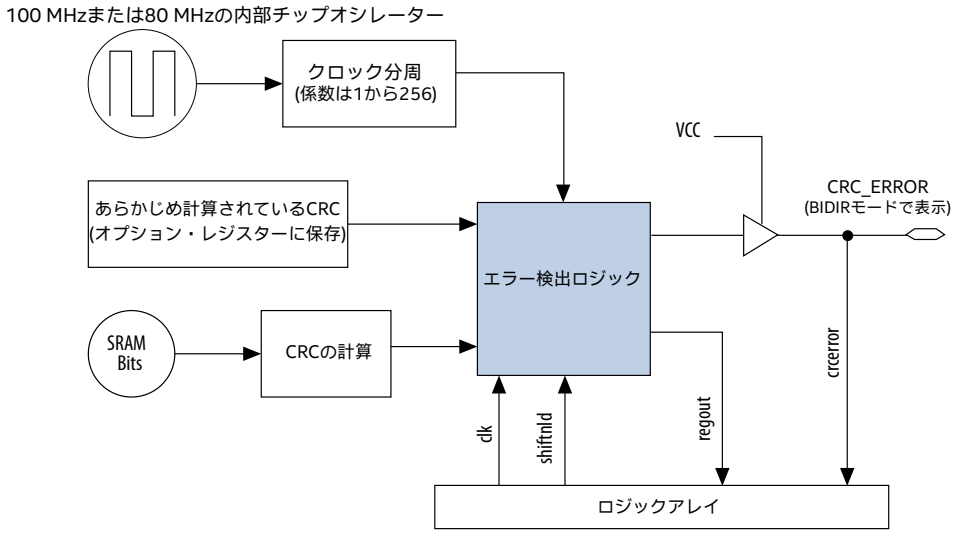
1.3.1.3. ユーザーロジックを使用したエラー検出ブロックへのアクセス

<device>_crcblock WYSIWYG コンポーネントを使用し、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します。<device>_crcblock プリミティブ・アトムには、アトムに組み込まなければならない入力および出力ポートが含まれています。ロジックアレイにアクセスするには、<device>_crcblock WYSIWYG アトムをデザインに挿入します。

EMR で提供される情報を使用してソフトエラーを検出することはできません。代わりに、ソフトエラーの影響を受けない CRC_ERROR 出力信号で提供される情報を使用します。

図 -8: ユーザーロジックでのエラー検出のブロック図

次の図は、デザインでイネーブルする必要のあるエラー検出機能および WYSIWYG アトムを示しています。



注意: インテル Quartus Prime 開発ソフトウェア・バージョン 8.0 SP1 またはそれ以前のバージョンでは、デザインに <device>_crcblock WYSIWYG アトムを含める場合、インテル Quartus Prime 開発ソフトウェアの Device & Pin Options ダイアログボックスでエラー検出 CRC 機能をイネーブルする必要があります。

<device>_crcblock WYSIWYG アトムをイネーブルするには、各デバイスに応じてアトムの名前を付けます。例えば、Stratix III デバイスの WYSIWYG アトムには stratixiii_crcblock、Arria II デバイスの WYSIWYG アトムには arriaii_crcblock と命名します。

表 4. CRC ブロックの入力および出力ポート

次の表に、WYSIWYG アトムに組み込む必要のある入力および出力ポートを示します。

ポート	入力/出力	定義
<crcblock_name>	入力	CRC ブロックに対する一意の識別子で、Verilog HDL、VHDL、Altera Hardware Description Language (AHDL) などの特定の記述言語の正当な識別子名を表します。このフィールドは必須です。
.clk(<clock source>)	入力	このセルのクロック入力を指定します。このセルの動作はすべてこのクロックの立ち上がりエッジに対して発生します。セルへのデータのロードまたはセルからのデータのロードにかかわらず、動作は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。

continued...

ポート	入力/出力	定義
.shiftnld (<shiftnld source>)	入力	• エラー検出ブロックへの入力です。 • shiftnld=1 の場合、ユーザー・シフト・レジスターは clk ポートの各立ち上がりエッジでデータを regout ポートにシフトします。 • shiftnld=0 の場合、ユーザー・シフト・レジスターはユーザー・アップデート・レジスターの内容をパラレルロードします。このポートは必須です。 • この入力は、ユーザー・アップデート・レジスターのクロックイネーブルを 2 サイクルの EDCLK 後にディアサートするようトリガーします。ED_SHIFTNLD 信号を Low に駆動した後、ED_CLK 信号をクロックする前に少なくとも 2 サイクルの EDCLK 間待機します。
.crcerror (<crcerror out destination>)	出力	• 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) に同期しているセルの出力です。SRAM ビットが反転し、内部の CRC 計算であらかじめ計算されている値と異なる値が示されたことをエラー検出ブロックが検出した場合、この出力は自動的に High にアサートされます。 • この信号は、出力ピンまたは双方向ピンに接続します。この出力信号を出力ピンに接続する場合、CRC_ERROR ピンのみがモニタリング可能となります (コアロジックはこの出力にアクセスできません)。コアロジックがエラー検出ロジックの読み出しに CRC_ERROR 信号を使用する場合、この信号は BIDIR ピンに接続します。信号は、V_{CC} に接続された出力イネーブルポート (oe) を有する BIDIR ピンを供給することで、コアに間接的に供給されます。 • CRC_ERROR ピンにルーティングされる信号は、コアにもルーティングされます。
.regout(<output destination>)	出力	• clk ポートに同期するユーザー・シフト・レジスターの出力で、コアロジックによって読み出されます。 • これは各サイクルで 1 ビットシフトし、LSB-first の形式に従います。

例-1: Stratix III デバイスにおける WYSIWYG アトムの入力ポートと出力ポートの例

次の例は、Stratix III デバイスにおける WYSIWYG アトムの入力ポートと出力ポートを表しています。

```
stratixiii_crcblock <crcblock_name>
             (
             .clk(<clock source>),
             .shiftnld(<shiftnld source>),
             .crcerror(<crcerror out destination>),
             .regout(<output destination>)
             );
             defparam crc_wysiwyg_atom.oscillator_divider = 2
```

注意: crc_wysiwyg_atom.oscillator_divider パラメーターは、<device>_crcblock WYSIWYG アトムのエラー検出周波数のクロック分周器を表しています。

関連情報

エラー検出のタイミング (7 ページ)

1.3.1.4. ユーザーロジック制御ブロック

ユーザーロジック制御ブロックを定義する必要があります。このアプリケーション・ノートでは、デザイン例が提供されています。このデザイン例では、ユーザーロジック制御ブロックが WYSIWYG アトムの入力ポートを制御し、ユーザー・アップデート・レジスターの内容を読み出します。ユーザー・アップデート・レジスターは、EMR の内容で更新されます。ユーザー・アップデート・レジスターの内容を読み出すには、次の手順を実行します。

1. SHIFTnLD 信号を Low に駆動します。
2. CLK_in を少なくとも 2 サイクル待機します。
3. 1 つの立ち上がりエッジに CLK_in を 1 サイクルクロックし、ユーザー・アップデート・レジスターの内容をユーザー・シフト・レジスターにロードします。
4. SHIFTnLD 信号を High に駆動します。
5. CLK_in を 29 サイクルクロックし、30 ビットのエラー位置情報を読み出します。
6. CLK_in をさらに 16 サイクルクロックし、エラーの症状を読み出します。

1.3.1.5. ユーザーロジック制御ブロックの信号

表 5. ユーザーロジック制御ブロックの信号

信号名	入力/出力	説明
clk_in	入力	ユーザーロジック制御ブロックへのクロックソースです。
Start_write	入力	この入力は、ユーザーロジック制御ブロックが SHIFTnLD 出力信号および ED_CLK 出力信号の駆動を開始するようトリガーします。この入力が High になると、ユーザーロジック制御ブロックは、ユーザー・アップデート・レジスターをアンロードするメカニズムの実行を開始します。EMR は crcerror ポートの立ち上がりエッジでアンロードされるため、この入力は通常、WYSIWYG アトムからの crcerror 出力ポートに接続されます。
SHIFTnLD	出力	WYSIWYG アトムへの出力です。この信号は、WYSIWYG アトムの shiftnld ポートを駆動するために使用します。
ED_CLK	出力	WYSIWYG アトムへの出力クロックです。このクロックは、WYSIWYG アトムへのクロックソースとして使用します。このクロックは、WYSIWYG アトムの clk ポートに接続されます。このクロックのソースは、clk_in 入力信号から得られます。

1.3.2. 外部ホストを使用したエラー検出

この章では、外部ホストを介したエラー検出機能の使用方法をデザイン例で説明します。

関連情報

- Design Example for Arria II, Stratix III, and Stratix IV Devices
- Design Example for Arria V, Cyclone V, and Stratix V Devices

1.3.2.1. CRC_ERROR 出力信号を使用した CRC エラーの検出

EMR の内容をアンロードする前に、CRC_ERROR 出力信号を使用して、CRC 計算において CRC エラーが検出されていないかを確認します。これを実行するには、インテル Quartus Prime 開発ソフトウェアの **Device & Pin Options** ダイアログボックスでエラー検出 CRC 機能をイネーブルにし、専用の CRC_ERROR ピンを観察します。

1.3.2.2. 外部ホストを使用した EMR のアンロード

JTAG ポートなどの外部ホストを使用して EMR の内容をアンロードするには、SHIFT_EDERROR_REG JTAG 命令を使用します。この JTAG 命令は、TDI ピンと TDO ピンの間のエラー検出ブロックの JTAG ピンに EMR を接続します。CRC_ERROR が High になればいつでもこの命令を実行することができます。

EMR の内容は、このレジスターが次の CRC エラー情報によって上書きされる前にアンロードする必要があります。

次の例は、EMR の内容をアンロードする SHIFT_EDERROR_REG JTAG 命令の実行に使用される.jam™ STAPL (Standard Test and Programming Language) 形式ファイル (Jam) を示しています。

例-2: Arria II、Stratix III、および Stratix IV デバイスにおいて EMR の内容をアンロードする.jam ファイルの例

```
ACTION UNLOAD_EMR = EXECUTE;

        DATA EMR_DATA;

        BOOLEAN out[46];

        BOOLEAN in[46]=$3FFFFFFFFFFF;

        ENDDATA;

        PROCEDURE EXECUTE USES EMR_DATA;

        DRSTOP IDLE;

        IRSTOP IDLE;

        STATE IDLE;

        IRSCAN 10, $017;

        WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;

        DRSCAN 46, in[45..0], CAPTURE out[45..0];

        WAIT IDLE, 10 CYCLES, 25 USEC, IDLE;

        PRINT " ";

        PRINT "Data read out from the EMR Register: ", out[45], out[44],
out[43], out[42], out[41], out[40], out[39], out[38], out[37], out[36],
out[35], out[34], out[33], out[32], 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];

        PRINT " ";

        PRINT "Syndrome:", out[45], out[44], out[43], out[42], out[41],
out[40], out[39], out[38], out[37], out[36], out[35], out[34], out[33],
out[32], out[31], out[30];

        PRINT "Frame Address:", 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];

        PRINT "Byte Location:", out[15], out[14], out[13], out[12], out[11],
out[10], out[9], out[8], out[7], out[6], out[5];

        PRINT "Bit Location:", out[4], out[3], out[2];

        PRINT "Error Type:", out[1], out[0];

        STATE IDLE;

        EXIT 0;

        ENDPROC;
```

例-3: Arria V、Cyclone V、および Stratix V デバイスにおいて EMR の内容をアンロードする .jam ファイルの例

```
ACTION UNLOAD_EMR = EXECUTE;

        DATA EMR_DATA;

        BOOLEAN out[67];

        BOOLEAN in[67]=$7FFFFFFFFFFFFFFF;

        INTEGER i;

        ENDDATA;

        PROCEDURE EXECUTE USES EMR_DATA;

        DRSTOP IDLE;

        IRSTOP IDLE;

        STATE IDLE;

        IRSCAN 10, $017;

        WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;

        DRSCAN 67, in[66..0], CAPTURE out[66..0];

        WAIT IDLE, 10 CYCLES, 25 USEC, IDLE;

        PRINT " ";

        PRINT "Data read out from the ";

        PRINT "EMR_Register ::", out[66], out[65], out[64], out[63], out[62],
out[61], out[60], out[59], out[58], out[57], out[56], out[55], out[54],
out[53], out[52], out[51], out[50], out[49], out[48], out[47], out[46],
out[45], out[44], out[43], out[42], out[41], out[40], out[39], out[38],
out[37], out[36], out[35]," ", out[34], out[33], out[32], 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];

        'PRINT " ";

        PRINT "Syndrome ::", out[66], out[65], out[64], out[63], out[62],
out[61], out[60], out[59], out[58], out[57], out[56], out[55], out[54],
out[53], out[52], out[51], out[50], out[49], out[48], out[47], out[46],
out[45], out[44], out[43], out[42], out[41], out[40], out[39], out[38],
out[37], out[36], out[35];

        PRINT "Frame Address ::", out[34], out[33], out[32], 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];

        PRINT "Double Word Location ::", out[18], out[17], out[16], out[15],
out[14], out[13], out[12], out[11], out[10], out[9];

        PRINT "Byte Offset ::", out[8], out[7];

        PRINT "Bit Offset ::", out[6], out[5], out[4];

        PRINT "Error Type ::", out[3], out[2], out[1], out[0];

        STATE IDLE;
```

```
        EXIT 0;

        ENDPROC;
```

関連情報

- エラー・メッセージ・レジスター (EMR) (6 ページ)
- インテル Quartus Prime Jam ツールでの.jam ファイルの実行 (24 ページ)
- マルチデバイス JTAG チェーンでの使用に向けたシングルデバイス用の.jam ファイルの変更 (23 ページ)

1.4. エラー・インジェクション

エラー検出ブロックをテストする目的で、意図的にエラーを注入することができます。このエラー注入手法により、デザインの検証およびシステム障害の許容特性評価を行います。EDERROR_INJECT JTAG 命令を使用することで、シングルエラー、ダブルエラー、またはダブル隣接エラーをコンフィグレーション・メモリーに注入することが可能になります。

EDERROR_INJECT JTAG 命令は、内部スクラブをテストするようにはデザインされていません。内部スクラブ機能を検証する場合は、インテル Quartus Prime Fault Injection Debugger を使用することをインテルでは推奨しています。

表 6. EDERROR_INJECT JTAG 命令

JTAG 命令	命令コード	説明
EDERROR_INJECT	00 0001 0101	この命令は、エラーの注入に使用される 21 ビットの JTAG フォールト・インジェクション・レジスターを制御します。

注意: Stratix V III デバイスの場合、EDERROR_INJECT JTAG 命令は 50 MHz のエラー検出周波数でのみ実行可能です。詳細については関連情報を参照してください。

関連情報

- SEU Mitigation in Stratix III Devices
- アルテラ・フォールト・インジェクション IP コアのユーザーガイド
  Fault Injection Debugger および、それを使用してエラーを注入、記録、およびクリアする方法について説明します。

1.4.1. フォールト・インジェクション・レジスター

EDERROR_INJECT JTAG 命令は、JTAG フォールト・インジェクション・レジスターの内容を制御します。レジスターは、コンフィグレーション・メモリーに注入するエラーの情報を保有します。

エラーの位置は、21 ビットの JTAG フォールト・インジェクション・レジスター (Arria V、Cyclone V、および Stratix V デバイスでは 46 ビット) にいつでもスキャンすることができます。JTAG フォールト・インジェクション・レジスターの内容は、最後と最初のデータフレームの処理中にフォールト・インジェクション・レジスターにロードされます。つまり、エラーを注入できるのはコンフィグレーション・データの最初のフレームのみですが、エラー情報はいつでもモニタリング可能です。レジスターは、コンフィグレーション・メモリーに注入するエラーの情報を保有します。

注意: フォールト・インジェクション・レジスターの内容は、JTAG フォールト・インジェクション・レジスターからフォールト・インジェクション・レジスターに 0 ビットがロードされる際にクリアされます。

1.4.1.1. Arria II、Stratix III、および Stratix IV デバイスにおけるフォールト・インジェクション

図 -9: Arria II、Stratix III、および Stratix IV デバイスにおけるフォールト・インジェクション・レジスター

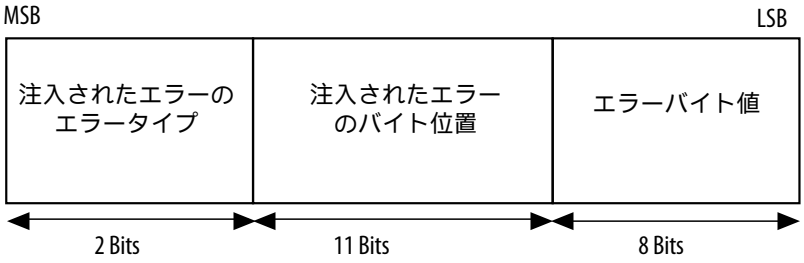


表 7. Arria II、Stratix III、および Stratix IV デバイスの EMR に注入されるエラーのタイプ

エラータイプ		説明
ビット 20	ビット 19	
0	0	エラー・インジェクションなし
0	1	シングル・バイト・エラー・インジェクション
1	0	ダブル隣接エラー・インジェクション
1	1	無効なエラー・インジェクション

1.4.1.2. Arria V、Cyclone V、および Stratix V デバイスにおけるフォールト・インジェクション

図 -10: Arria V、Cyclone V、および Stratix V デバイスにおけるフォールト・インジェクション・レジスター

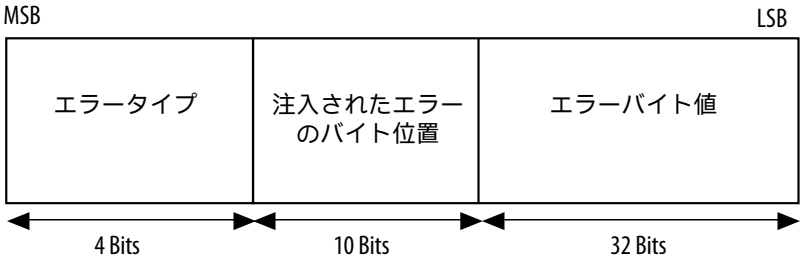


表 8. Arria V、Cyclone V、および Stratix V デバイスの EMR に注入されるエラーのタイプ

エラータイプ				説明
ビット 45	ビット 44	ビット 43	ビット 42	
0	0	0	0	エラー・インジェクションなし
0	0	0	1	シングル・バイト・エラー・インジェクション
0	0	1	0	ダブル隣接エラー・インジェクション
その他				無効なエラー・インジェクション

1.4.2. EDERROR_INJECT JTAG 命令を使用したエラー・インジェクション

この章では、テストプロセスについて説明します。.jam ファイルを使用してテストプロセスを自動化すると、デバイスをリコンフィグレーションすることなくエラー検出ブロックの機能をテストおよび検証することができます。EDERROR_INJECT JTAG 命令を使用して JTAG フォールト・インジェクション・レジスターの内容を制御し、JTAG フォールト・インジェクション・レジスターで指定されている最初のデータフレーム内の特定の位置にあるリードバック・ビットを反転させます。

注意: インテルでは、CRC_ERROR ピンの動作を観察し、推奨されている方法で EMR の内容をアンロードすることで、エラーが正確に注入されたかを確認することを推奨しています。

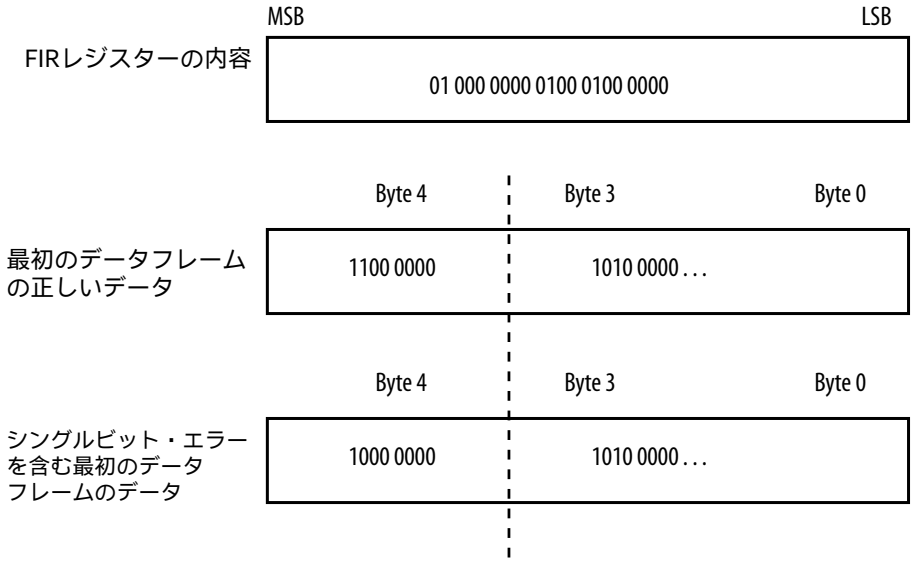
1.4.2.1. Arria II、Stratix III、および Stratix IV におけるシングルビット・エラーの注入

シングルビット・エラーを最初のデータフレーム内の 5 番目のバイトの 7 番目のビットに注入する場合、注入されるエラーの情報を含むデータをフォールト・インジェクション・レジスターにシフトインする必要があります。

次の図は、フォールト・インジェクション・レジスターにシフトするバイナリー・ビット・シーケンスの 0 1000 0000 0100 0100 0000 を表しています。フォールト・インジェクション・レジスターのビット 19 とビット 20 は 01 であるため、シングルビット・エラーが注入されます。注入されるエラーのバイト位置は 000 0000 0100 として定義されているため、シングルビット・エラーはデータフレームの 5 番目のバイトに注入されます。エラーバイト位置の値は 0100 0000 であり、5 番目のバイトデータの 7 番目のビットが、シングルビット・エラーが注入される特定の位置であることを示しています。

図 -11: Arria II、Stratix III、および Stratix IV デバイスにおけるシングルビット・エラーの注入

次の図は、エラー・インジェクションのメカニズムを示しています。



次の例は、Arria II、Stratix III、および Stratix IV デバイスにおいて最初のデータフレームにシングルビット・エラーを注入する際に、EDERROR_INJECT JTAG 命令で実行する必要のある .jam ファイルを示しています。

例-4: Arria II、Stratix III、および Stratix IV デバイスにおいてシングルビット・エラーを注入する .jam ファイルの例

```
ACTION ERROR_INJECT = EXECUTE;

        DATA DEVICE_DATA;

        BOOLEAN out[21];

        BOOLEAN in[21] = $080440;

        ENDDATA;

        PROCEDURE EXECUTE USES DEVICE_DATA;

        BOOLEAN X = 0;

        DRSTOP IDLE;

        IRSTOP IDLE;

        STATE IDLE;

        IRSCAN 10, $015;

        WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;

        DRSCAN 21, in[20..0], CAPTURE out[20..0];      'shift out the previous
content from fault injection register

        WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;

        DRSCAN 21, in[20..0], CAPTURE out[20..0];    'shift out current content
from fault injection register

        WAIT IDLE, 10 CYCLES, 25 USEC, IDLE;

        PRINT " ";

        PRINT "Data read out from the FIR Register: ", 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];

        PRINT " ";

        PRINT "Error Type:", out[20], out[19];

        PRINT " ";

        PRINT "Byte Location:", out[18], out[17], out[16], out[15], out[14],
out[13], out[12], out[11], out[10], out[9], out[8];

        PRINT " ";

        PRINT "Error Byte Value:", out[7], out[6], out[5], out[4], out[3],
out[2], out[1], out[0];

        STATE IDLE;

        EXIT 0;

        ENDPROC;
```

関連情報

- インテル Quartus Prime Jam ツールでの.jam ファイルの実行 (24 ページ)

- マルチデバイス JTAG チェーンでの使用に向けたシングルデバイス用の.jam ファイルの変更 (23 ページ)

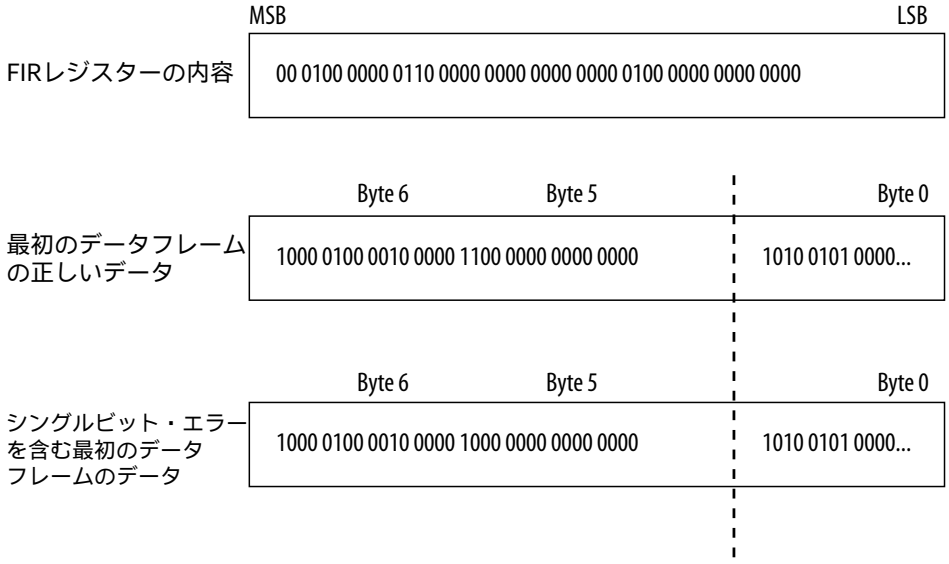
1.4.2.2. Arria V、Stratix V、および Cyclone V デバイスにおけるシングルビット・エラーの注入

シングルビット・エラーを最初のデータフレーム内の 7 番目のバイトの 15 番目のビットに注入する場合、注入されるエラーの情報を含むデータをフォールト・インジェクション・レジスターにシフトインする必要があります。

次の図は、フォールト・インジェクション・レジスターにシフトするバイナリー・ビット・シーケンスの 00 0100 0000 0110 0000 0000 0000 0000 0100 0000 0000 0000 を表しています。フォールト・インジェクション・レジスターのビット 45、44、43、42 が 0001 であるため、シングルビット・エラーが注入されます。注入されるエラーのバイト位置は 00 0000 0110 として定義されているため、シングルビット・エラーはデータフレームの 7 番目のバイトに注入されます。エラーバイト位置の値は 0000 0000 0000 000 0100 0000 0000 0000 であり、7 番目のバイトデータの 15 番目のビットが、シングルビット・エラーが注入される特定の位置であることを示しています。

図 -12: Arria V、Stratix V、および Cyclone V デバイスにおけるシングルビット・エラーの注入

次の図は、エラー・インジェクションのメカニズムを示しています。



次の例は、Arria V、Stratix V、および Cyclone V デバイスにおいて最初のデータフレームにシングルビット・エラーを注入する際に、EDERROR_INJECT JTAG 命令で実行する必要のある .jam ファイルを示しています。

例-5: Arria V、Stratix V、および Cyclone V デバイスにおいてシングルビット・エラーを注入する .jam ファイルの例

```
ACTION ERROR_INJECT = EXECUTE;

        DATA DEVICE_DATA;

        BOOLEAN out[46];

        BOOLEAN in[46] = $040600004000;

        ENDDATA;
```

```
        PROCEDURE EXECUTE USES DEVICE_DATA;

        BOOLEAN X = 0;

        DRSTOP IDLE;

        IRSTOP IDLE;

        STATE IDLE;

        IRSCAN 10, $015;

        WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;

        DRSCAN 46, in[45..0], CAPTURE out[45..0];     ‘shift out the previous
content from fault injection register

        WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;

        DRSCAN 46, in[45..0], CAPTURE out[45..0];    ‘shift out current content
from fault injection register

        WAIT IDLE, 10 CYCLES, 50 USEC, IDLE;

        PRINT " "; PRINT "Data read out from the FIR Register: ", out[45],
out[44], out[43], out[42], " ", out[41], out[40], out[39], out[38], out[37],
out[36], out[35], out[34], out[33], out[32], " ", 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];

        PRINT " ";

        PRINT "Error Type:", out[45], out[44], out[43], out[42];

        PRINT " ";

        PRINT "Byte Location:", out[41], out[40], out[39], out[38], out[37],
out[36], out[35], out[34], out[33], out[32];

        PRINT " ";

        PRINT "Error Byte Value:", 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];

        STATE IDLE;

        EXIT 0;

        ENDPROC;
```

関連情報

- インテル Quartus Prime Jam ツールでの.jam ファイルの実行 (24 ページ)
- マルチデバイス JTAG チェーンでの使用に向けたシングルデバイス用の.jam ファイルの変更 (23 ページ)

1.4.3. フォールト・インジェクション・レジスターのクリア

テストプロセスの完了後、フォールト・インジェクション・レジスターの内容をクリアする、もしくは nCONFIG 信号を Low にしてリコンフィグレーションを開始することで、注入したエラーをディスエーブルします。フォールト・インジェクション・レジスターの内容をクリアするには、レジスターにすべて 0 のデータをスキャンします。

次の例は、フォールト・インジェクション・レジスターの内容のクリアに使用される .jam ファイルを表しています。

例-6: Arria II、Stratix III、および Stratix IV デバイスにおいてフォールト・インジェクション・レジスターの内容をクリアする .jam ファイルの例

```
ACTION ERROR_INJECT_DISABLE = EXECUTE;

        DATA DEVICE_DATA;

        BOOLEAN out[21];

        BOOLEAN in[21] = $000000;

        ENDDATA;

        PROCEDURE EXECUTE USES DEVICE_DATA;

        BOOLEAN X = 0;

        DRSTOP IDLE;

        IRSTOP IDLE;

        STATE IDLE;

        IRSCAN 10, $015;

        WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;

        DRSCAN 21, in[20..0], CAPTURE out[20..0];    'shift out the previous
content from fault injection register

        WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;

        DRSCAN 21, in[20..0], CAPTURE out[20..0];    'shift out current content
from fault injection register

        WAIT IDLE, 10 CYCLES, 25 USEC, IDLE;

        PRINT " ";

        PRINT "Data read out from the FIR Register: ", 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];

        PRINT " ";

        PRINT "Error Type:", out[20], out[19];

        PRINT " ";

        PRINT "Byte Location:", out[18], out[17], out[16], out[15], out[14],
out[13], out[12], out[11], out[10], out[9], out[8];

        PRINT " ";

        PRINT "Error Byte Value:", out[7], out[6], out[5], out[4], out[3],
out[2], out[1], out[0];
```

```
        STATE IDLE;

        EXIT 0;

        ENDPROC;
```

例-7: Arria V、Stratix V、および Cyclone V デバイスにおいてフォールト・インジェクション・レジスターの内容をクリアする .jam ファイルの例

```
ACTION ERROR_INJECT_DISABLE = EXECUTE;

        DATA DEVICE_DATA;

        BOOLEAN out[46];

        BOOLEAN in[46] = $000000000000;

        ENDDATA;

        PROCEDURE EXECUTE USES DEVICE_DATA;

        BOOLEAN X = 0;

        DRSTOP IDLE;

        IRSTOP IDLE;

        STATE IDLE;

        IRSCAN 10, $015;

        WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;

        DRSCAN 46, in[45..0], CAPTURE out[45..0];    'shift out the previous
content from fault injection register

        WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;

        DRSCAN 46, in[45..0], CAPTURE out[45..0];    'shift out current content
from fault injection register

        WAIT IDLE, 10 CYCLES, 50 USEC, IDLE;

        PRINT " ";

        PRINT "Data read out from the FIR Register: ", out[45], out[44],
out[43], out[42], " ", out[41], out[40], out[39], out[38], out[37], out[36],
out[35], out[34], out[33], out[32], " ", 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];

        PRINT " ";

        PRINT "Error Type:", out[45], out[44], out[43], out[42];

        PRINT " ";

        PRINT "Byte Location:", out[41], out[40], out[39], out[38], out[37],
out[36], out[35], out[34], out[33], out[32];

        PRINT " ";

        PRINT "Error Byte Value:", 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];

        STATE IDLE;

        EXIT 0;

        ENDPROC;
```

注意: インテルでは、エラー検出のテストと検証後、デバイスのリコンフィグレーションを行うことを推奨しています。

関連情報

- インテル Quartus Prime Jam ツールでの.jam ファイルの実行 (24 ページ)
- マルチデバイス JTAG チェーンでの使用に向けたシングルデバイス用の.jam ファイルの変更 (23 ページ)

1.5. マルチデバイス JTAG チェーンでの使用に向けたシングルデバイス用の .jam ファイルの変更

このドキュメントの .jam ファイルのコードは、シングルデバイスの JTAG チェーンを対象にしています。これらのコードをマルチデバイスの JTAG チェーンで使用するには、チェーン内の .jam ファイルが対象とするデバイス以外のデバイスの命令レジスター (IR) およびデータレジスター (DR) の長さを追加します。

1. JTAG チェーン内の他のデバイスすべての命令レジスター長を確認します。
   - IR の長さ
     - インテル FPGA および CPLD デバイス: 10
     - インテル SoC FPGA デバイス内のハードウェア・プロセッサー・システム (HPS): 4
   - デバイスの DR の長さ: 1
2. .jam ファイルのコード内で PROCEDURE EXECUTE ラインを特定し、それに続く新しいラインに次の手順で示されているコードを追加します。
3. チェーン内で対象デバイスの前にデバイスがある場合は、次のコードを追加します。

```
POSTIR <total IR length before the target device>;
POSTDR <total DR length before the target device>;
```

4. チェーン内で対象デバイスの後にデバイスがある場合は、次のコードを追加します。

```
PREIR <total IR length after the target device>;
PREDR <total DR length after the target device>;
```

例-8: JTAG チェーンで対象デバイスの前または後に他のデバイスが存在する場合

それぞれのチェーン例において、PROCEDURE EXECUTE ラインの後にコードを追加します。

- ダウンロード・ケーブル TDI → 他のデバイス 1 (IR=10) → 対象デバイス → ダウンロード・ケーブル TDO

```
POSTIR 10;
POSTDR 1;
```

- ダウンロード・ケーブル TDI → 対象デバイス → 他のデバイス 1 (IR=10) → ダウンロード・ケーブル TDO

```
PREIR 10;
PREDR 1;
```

- ダウンロード・ケーブル TDI → 対象デバイス → 他のデバイス 1 (IR=10) → 他のデバイス 2 (IR=10) → ダウンロード・ケーブル TDO

```
PREIR 20;
PREDR 2;
```

- ダウンロード・ケーブル TDI → 他のデバイス 1 (IR=4) → 対象デバイス → 他のデバイス 2 (IR=10) → ダウンロード・ケーブル TDO

```
POSTIR 4;
POSTDR 1;
PREIR 10;
PREDR 1;
```

関連情報

- 外部ホストを使用した EMR のアンロード (12 ページ)
- Arria II、Stratix III、および Stratix IV におけるシングルビット・エラーの注入 (17 ページ)
- Arria V、Stratix V、および Cyclone V デバイスにおけるシングルビット・エラーの注入 (19 ページ)
- フォールト・インジェクション・レジスターのクリア (21 ページ)

1.6. インテル Quartus Prime Jam ツールでの .jam ファイルの実行

.jam ファイルは、コマンドラインの インテル Quartus Prime Jam ツールである quartus_jli を使用して実行することができます。

1. 関連情報に記載されている項目の該当する例から .jam コードをコピーします。
2. テキストファイルにコードを貼り付け、ファイルを *<filename>*.jam として保存します。
3. ダウンロード・ケーブルのインデックス番号を特定するには、コマンドラインで次のコマンドを実行します。

   ```
   quartus_jli -n
   ```

4. .jam ファイルを実行するには、コマンドラインで次のコマンドを実行します。

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

   例: `quartus_jli -a error_inject -c 2 errortest.jam`

関連情報

- Arria II、Stratix III、および Stratix IV におけるシングルビット・エラーの注入 (17 ページ)
- Arria V、Stratix V、および Cyclone V デバイスにおけるシングルビット・エラーの注入 (19 ページ)
- フォールト・インジェクション・レジスターのクリア (21 ページ)
- 外部ホストを使用した EMR のアンロード (12 ページ)

1.7. AN 539: インテル® FPGA デバイスにおける CRC を使用したエラー検出および回復のテスト方法改訂履歴

ドキュメント・バージョン	変更内容
2019.08.09	• エラー・インジェクションに関する内容を更新し、EDERROR_INJECT は内部スクラブ機能のテストにはデザインされていないことを明確にしました。代わりに、Fault Injection Debugger を使用します。 • フォールト・インジェクション・レジスターのクリアに関する内容を更新し、これまで示されていたコード例は Arria II、Stratix III、および Stratix IV デバイスに向けたものであることを明確にし、Arria V、Stratix V、および Cyclone V デバイスに向けたコード例を追加しました。 • .jam ファイルを変更してマルチデバイスの JTAG チェーンで使用する方法を追加しました。 • quartus_jli コマンドを使用して .jam ファイルを実行する方法を追加しました。
2018.05.07	• ブランド名をインテルに変更しました。 • 次の例を更新しました。 — Arria II、Stratix III、および Stratix IV デバイスにおいてシングルビット・エラーを注入する .jam ファイルの例 — Arria V、Cyclone V、および Stratix V デバイスにおいてシングルビット・エラーを注入する .jam ファイルの例 — フォールト・インジェクション・レジスターの内容をクリアする .jam ファイルの例

日付	バージョン	変更内容
2017 年 6 月	2017.06.14	例 5 を修正しました。
2016 年 8 月	2016.08.03	• フォールト・インジェクション・レジスターのサブセクションのタイトルを訂正しました。
2015 年 8 月	2015.08.20	• *フォールト・インジェクション・レジスター*で編集ミスを修正しました。 • 関連リンクの説明を修正しました。 • 成熟した製品のアプリケーション・ノートである AN357 へのリンクを削除しました。
2015 年 5 月	2015.05.04	• Arria V、Cyclone V、および Stratix V デバイスにおいてシングルビット・エラーを注入する.jam ファイルの例を、二重引用符を訂正し、35 行目にセミコロンを追加することで更新しました。 • テンプレートを更新しました。
2014 年 12 月	2014.12.15	• 5 ページの図 3 を更新しました。
2014 年 4 月	3.0	• 11 ページの表 5 を更新し、.clk(<clock source>) ポートの動作をクロックの立ち下がりエッジから立ち上がりエッジに訂正しました。 • Arria V および Cyclone V をサポートするように内容を更新しました。 • 5 ページの図 3 を更新しました。 • 16 ページに例 3 を追加しました。
2011 年 6 月	2.0	• 図 3、図 10、図 12 を追加しました。 • 表 4 および表 9 を追加しました。 • 例 5 を追加しました。 • 「エラー訂正」の章を追加しました。

continued...

日付	バージョン	変更内容
		• 図 1、図 2、図 9、および図 11 を更新しました。 • 表 1、表 3、表 8 を更新しました。 • 「機能の説明」、「コンフィグレーション・エラーの検出」、「ユーザーモードでのエラー検出」、「エラー・メッセージ・レジスター」、「フォールト・インジェクション・レジスター」、「EDERROR_INJECT JTAG 命令を使用したエラー・インジェクション」の章を更新しました。 • 編集上の軽微な変更を行いました。
2009 年 4 月	1.1	• 「概要」、「EDERROR_INJECT JTAG 命令を使用したエラー・インジェクション」、「エラー・インジェクション」、「エラー検出ピン」の章を更新しました。 • 表 4 および表 8 を更新しました。
2008 年 12 月	1.0	初版