XAPP538 (v1.0) 2012 年 4 月 4 日

優先エッセンシャル ビットを使用したソフト エラーの軽減

著者 : Robert Le

概要

コンフィギュレーション メモリでソフト エラーが生じると、デザインが破損する可能性があります。しかし、ザイリンクス FPGA へロードされたデザインの正常動作は、通常コンフィギュレーション メモリ セルのわずかな割合にしか関与しません。このアプリケーション ノートでは、ユーザー デザインで特定の階層領域を定義し、ISE® デザイン ツール (13.4 またはそれ以降) を使用して定義したユーザー ロジックに関連した優先エッセンシャル ビットを特定する方法について説明します。さらに、優先エッセンシャル ビットと LogiCORE™ IP の Soft Error Mitigation (SEM) Controller を併用して、ザイリンクスの 7 シリーズおよび Virtex®-6 FPGA のコンフィギュレーション メモリのソフト エラーを検出および訂正する方法についても解説します。この方法により、FIT (Failure In Time : 故障率) が低減し、デザインの有用性が向上します。ここでは、付属するリファレンス デザインを使用してデザイン フローを説明します。

はじめに

コンフィギュレーション メモリは、幅の広いスタティック RAM のように多数のフレームで構成されています。大半のデバイス ファミリではエラー訂正コード (ECC) で各フレームが保護され、すべてのデバイス ファミリでは巡回冗長検査 (CRC) で全フレームが保護されます。これら 2 つの機能は相補的に作用します。つまり、CRC はエラー検出に優れており、ECC はエラーの位置を高い精度で特定します。FPGA 内のすべてのコンフィギャラブル リソースは、1 つまたは複数のコンフィギュレーション メモリ ビットで定義されています。

ソフト エラーとは、電離放射線によってコンフィギュレーション メモリ ビットのステートに意図しない変更が起こることです。ソフト エラーでクリティカル ビットが変更されると、デザインの機能が破損する可能性があります (ここでは、ステートが変更されると機能不具合を起こすビットをクリティカル ビットという)。ザイリンクス FPGA へロードされたデザインは、通常一部のコンフィギャラブル メモリ セルのみ使用します。使用しないセルのメモリ ビットは、デザインにとって重要ではありません。したがって、ソフト エラーの検出および訂正は、デザイン回路に関わるコンフィギュレーション ビットのみを対象にすればよいことになります (ここでは、デザイン回路に関わるビットをエッセンシャル ビットといい、デバイスのコンフィギュレーション ビットの一部である)。

ザイリンクスのエッセンシャル ビット テクノロジは、コンフィギュレーション ビットの中からエッセンシャル ビットを識別するアルゴリズムを使用します。エッセンシャル ビットが反転すると、デザイン回路は変更されますが、デザインの機能に影響を及ぼさない場合もあります。ザイリンクスのエッセンシャル ビット テクノロジは、FIT を大幅に低減し、デザインの有用性を高めるのに効果的な方法を提供するために開発されました。このテクノロジにより、システムをオフライン状態にする致命的な不具合のように反転ビットをすべて処理する必要がなくなるため、システムの可用性が向上します。さらに、より高い信頼性基準も達成できます。

ザイリンクスは、エッセンシャル ビット テクノロジの効果を高めるために、エッセンシャル ビット リストに優先フィルターをかける方法を提供しています。これを利用して、デザインに定義した特定の階層領域に基づいてエッセンシャル ビット ファイルに優先フィルターをかけることができます。この方法でフィルタリングしたビットを優先エッセンシャル ビットと呼びます。

クリティカル ビットを判断するのはシステム設計者のみ可能です。デザインの完全なクリティカル ビット リストを生成するには、デザインのコンフィギュレーション ビットを 1 つずつ移動させながら正しいデザイン動作を検証しなければならないため、莫大な時間を要します。このためには、指定した領域

がデザインの正常動作にとってエッセンシャルか非エッセンシャル (必須か否か) に分類する必要があります (つまり、1 つの領域にエッセンシャル ビットと非エッセンシャル ビットは混在しない)。たとえば、デザインの鍵となるステート マシンは、優先フィルターでデザインの正常動作にエッセンシャルと判断され、デバッグ専用モジュールは、デザインの正常動作には非エッセンシャルと分類されます。

図 1 に、デバイス コンフィギュレーション ビット、エッセンシャル ビット、優先エッセンシャル ビット、およびクリティカル ビットの関係を示します。

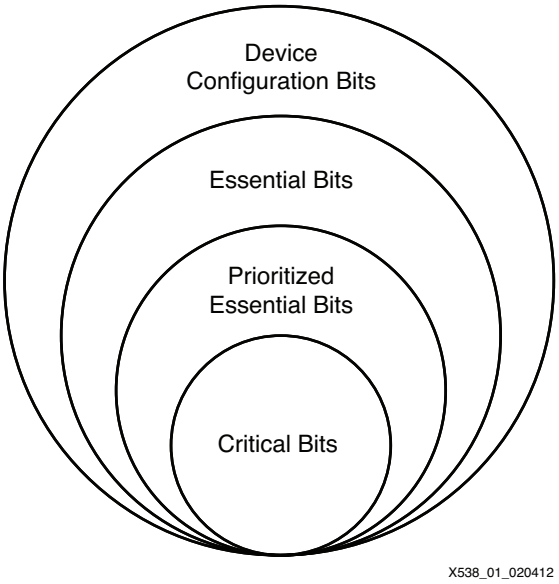

図 1 : デバイス コンフィギュレーション ビット、エッセンシャル ビット、優先エッセンシャル ビット、およびクリティカル ビットの関係

SEM Controller は、ザイリンクス FPGA のコンフィギュレーション メモリのソフト エラーを検出および訂正する、自動コンフィギュレーションで検証済みのソリューションです。SEM Controller はソフト エラーを防ぐことはできませんが、ソフト エラーによるシステム レベルの影響を適切に管理する手段を提供します。エラーに適切に対処することで、デザインの信頼性と可用性が向上し、システムの管理コストおよびダウンタイム コストを削減できます。SEM Controller は、オプションとしてエラー分類機能もサポートします。この機能は、ルックアップ テーブルを用いてソフト エラー イベントによってエッセンシャルなコンフィギュレーション メモリ ビットが変更されたかどうかを判断します。この機能を活用することで、実際の FIT が低減します。使用の際には、エッセンシャル ビット ファイルを含むルックアップ テーブルを格納するための外部記憶装置が必要です。このファイルを使用して、コンフィギュレーション ビットがエッセンシャルまたは非エッセンシャルであるかを判断します。優先エッセンシャル ビットを使用する場合は、ルックアップ テーブルの内容が優先エッセンシャル ビット ファイルで置き換えられます。このデザイン フローに関しては、ISE デザイン ツールでこのファイルを生成する方法と共に後に詳しく説明します。

優先エッセンシャル ビットの生成および使用

図 2 に、優先エッセンシャル ビットの生成および使用の大まかなフローを示します。

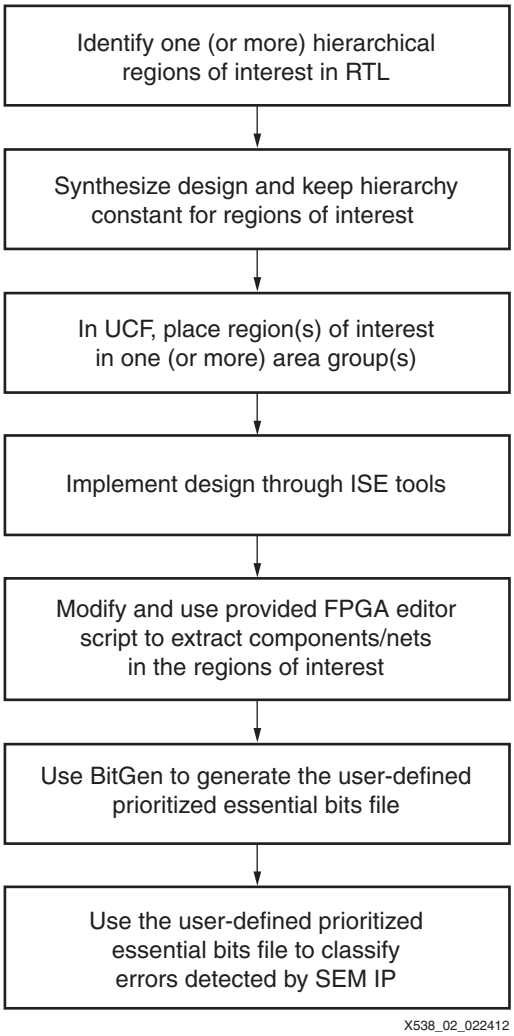

図 2 : 優先エッセンシャル ビットの生成および使用

次に、このデザイン フローの詳細を示します。

RTL の階層領域の特定およびデザインの合成

RTL レベルで、デザインの動作にとってクリティカルと判断した階層領域を 1 つまたは複数指定し、これらの領域を限定してモジュールを分離するように RTL を分割します。この手順は、フローの中で最も難しく重要です。この指定および分割作業は、カスタマー デザインやシングル イベント アップセット (SEU) 軽減の要件によって異なります。

クリティカルな領域と非クリティカルな領域間に明確な境界を保持するには、RTL で階層維持 (KEEP_HIERARCHY) 制約を適用します。この制約は、RTL デザインで指定された階層ブロック (VHDL エンティティ、Verilog モジュール) へ適用されます。合成中に階層を維持する場合は、ISE ツールで Keep Hierarchy を使用してインプリメンテーション中の階層を維持するようにします。

図 3 で、エンティティ /モジュール I2 に Keep Hierarchy が設定されている場合、次のようになります。

- 階層 I2 は、最終ネットリストに保持される
- 階層 I4 と I5 は、階層 I2 内でフラット化される
- 階層 I1、I3、I6、および I7 もフラット化される

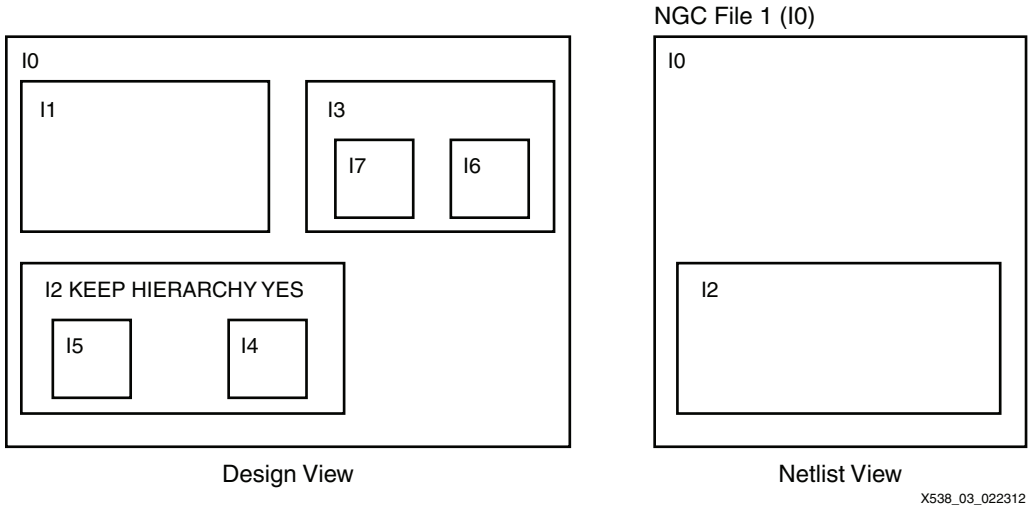

図 3 : Keep Hierarchy を設定した場合

XST (Xilinx Synthesis Tool) での階層維持構文は次のとおりです。

```
VHDL Syntax:
attribute keep_hierarchy : string ;
attribute keep_hierarchy of instant_name: label is "yes"
Verilog Syntax :
(* keep_hierarchy = "yes" *)
```

リファレンス デザインの階層は次のとおりです。

```
sem_example
 |
 +- sem_core
 |
 +- sem_cfg
 |
 +- sem_mon
 |    |
 |    +- sem_mon_fifo
 |    |
 |    +- sem_mon_piso (region of interest)
 |    |
 |    +- sem_mon_sipo (region of interest)
 |
 +- sem_ext
 |    |
 |    +- sem_ext_byte
 |
 +- sem_hid
 |    |
 |    +- chipscope_icon
 |    |
 |    +- chipscope_vio
 |
 \- BUFG (unisim)
```

リファレンス デザインでは、デザイン フローを説明するために、sem_mon_piso および sem_mon_sipo が指定領域の例として使用されています。これらのモジュールの Verilog および VHDL ファイルは、sem_mon_piso.v/vhd および sem_mon_sipo.v/vhd です。

Keep Hierarchy は、次のように適用されています。

```
Verilog:
module sem_mon (
  …..

  );

  (* keep_hierarchy = "yes" *)
  sem_mon_piso example_mon_piso (
    ……

    );

  (* keep_hierarchy = "yes" *)
  sem_mon_sipo example_mon_sipo (

     ……

    );

 ……
endmodule


VHDL:
entity sem_mon is
port (

  ……

  );
end entity sem_mon;
architecture xilinx of sem_mon is
  ……
attribute keep_hierarchy : string;
attribute keep_hierarchy of example_mon_piso : label is "yes";
attribute keep_hierarchy of example_mon_sipo : label is "yes";
begin
  ……
example_mon_piso : sem_mon_piso
  port map (
     …….

    );
example_mon_sipo : sem_mon_sipo
  port map (
     …….

    );
……
end architecture xilinx;
```

指定した階層領域に対してエリア制約を作成する

ユーザーは、指定した階層領域を 1 つまたは複数の AREA_GROUP へ配置します。エリア グループは、マップ、パック、配置配線用にデザインを物理的領域に分割して、デバイスの特定領域にロジックを配置する基本的な手段です。エリア グループによって、指定した領域がその他の領域から分離されます。

UCF (ユーザー制約ファイル) のエリア グループ構文は次のとおりです。

```
INST "X" AREA_GROUP = groupname;
AREA_GROUP "groupname" RANGE = range,
AREA_GROUP "groupname" GROUP = CLOSED;
AREA_GROUP "groupname" PLACE = OPEN;
```

X には INST 名が入り、これは RTL で定義されている階層領域の名前と一致させます。groupname は、パッカーで関連付けられてプレーサーでエリア グループに配置される論理ブロックの名前です。

ツールで関係のないロジックが 1 つのコンポーネントへパックされないようにするには、GROUP = CLOSED とした AREA_GROUP を使用する必要があります。これにより、論理デザインのエレメントを物理デザイン内で名前によって識別できるようになります。AREA_GROUP に対して PLACE = OPEN とすると、そのエリアに関係のないコンポーネントが配置されます。

UCF の AREA_GROUP 制約の構文例は次のとおりです。

```
## Area Constraints on hierarchical region of interest:
INST "example_mon/example_mon_piso/*" AREA_GROUP = "ESSENTIAL_LOGIC" ;
INST "example_mon/example_mon_sipo/*" AREA_GROUP = "ESSENTIAL_LOGIC" ;
AREA_GROUP "ESSENTIAL_LOGIC" RANGE = SLICE_X48Y100:SLICE_X51Y149 ;
## Prohibit addition of unrelated logic into this group...
AREA_GROUP "ESSENTIAL_LOGIC" GROUP = CLOSED ;
## Allow placement of unrelated components in the range...
AREA_GROUP "ESSENTIAL_LOGIC" PLACE = OPEN ;
```

UCF の作成後、ISE デザイン ツールでデザインをインプリメントし、配置配線ネットリストを生成します。提供されているサンプル インプリメンテーション スクリプト (`implement.sh` または `implement.bat`) は、インプリメンテーション ツールでコマンド ラインを使用してデザインを実行するフローを示しています。

FPGA Editor スクリプトを変更して特定領域のコンポーネント/ネットを抽出する

一般的なインプリメンテーション フローでデザインを実行後、ユーザーは提供されている FPGA Editor スクリプト (`essential.scr`) を変更できます。スクリプトを開いて、特定領域のコンポーネントおよびネットを選択し、これらのリストを完成させてファイルを閉じます。

リファレンス デザインと共に提供される FPGA Editor スクリプトの内容は次のとおりです。

```
clear
select -k comp *example_mon/example_mon_piso/*
select -k comp *example_mon/example_mon_sipo/*
select -k net *example_mon/example_mon_piso/*
select -k net *example_mon/example_mon_sipo/*
list
clear
exit
```

このスクリプトでは、特定領域のコンポーネントおよびネットが選択されています。リファレンス デザインでは、コンポーネントおよびネットは次のようになります。

- `*example_mon/example_mon_piso/*`
- `*example_mon/example_mon_sipo/*`

ユーザーは、スクリプトを変更してユーザー デザインの特定の階層領域を使用します。FPGA Editor スクリプトを実行するコマンドは次のとおりです。

fpga_edline routed.ncd mapped.pcf -p essential.scr

上記の例を実行した場合の出力として、fpga_edline ログ ファイル (routed_fpga_edline.log) が生成されます。このファイルは長いため、ここでは省略した例を示します。

```
#Xilinx FPGA Editor Command Log File
#
#------------------------------
        #Reading routed.ncd...
       #   "sem_example" is an NCD, version 3.2, device xc7k325t, package
fbg900, speed -1
        #Building chip graphics...
        #Loading speed info...
        #1
setattr main edit-mode no-logic-changes
        #2
playback essential.scr
        #clear
        #select -k comp *example_mon/example_mon_piso/*
        #select -k comp *example_mon/example_mon_sipo/*
        #select -k net *example_mon/example_mon_piso/*
        #select -k net *example_mon/example_mon_sipo/*
        #list
        #Comp "example_mon/example_mon_piso/bit_select<1>"
        #Comp "example_mon/example_mon_piso/piso_out"
        #Comp "example_mon/example_mon_piso/tx_start"
        #Comp "example_mon/example_mon_sipo/delay_line<149>"
        #Comp "example_mon/example_mon_sipo/Mshreg_valid_delay_151_0"
        #Comp "example_mon/example_mon_sipo/valid_delay<151>"
        #Net "example_mon/example_mon_piso/hot_delay<15>"
        #Net "example_mon/example_mon_piso/tx_stop
        #Net "example_mon/example_mon_piso/hot_delay<0>"
        #Net "example_mon/example_mon_piso/en_16_x_baud_tx_bit_AND_5_o"
        #Net "example_mon/example_mon_piso/bit_select<2>"
        #Net "example_mon/example_mon_piso/bit_select<1>"
        #clear
        #exit
```

BitGen を使用して優先エッセンシャル ビット ファイルを生成する

次のコマンド ライン オプションを使用して、BitGen を呼び出します。

-g essentialbits:yes

-g essentialbitsfilter:{none|mask|enable}

-g essentialbitsfilterfile:<filename>

- **essentialbitsfilter** を **enable** とした場合、BitGen は、**essentialbitsfilterfile** オプションで指定した名前と一致するデザイン エンティティに関連するエッセンシャル ビットを生成します。リファレンス デザインでは、このオプションを使用しています。
- **essentialbitsfilter** を **mask** とした場合、生成されるエッセンシャル ビットには **essentialbitsfilterfile** オプションで指定した名前と一致するデザイン エンティティに関連するビットは含まれません。
- **essentialbitsfilter** を **none** とした場合、生成されるエッセンシャル ビットにはすべてのデザインに関連するビットが含まれます。

-g essentialbitsfilterfile で使用される **filename** は、「FPGA Editor スクリプトを変更して特定領域のコンポーネント/ネットを抽出する」で説明した FPGA Editor スクリプトで生成される FPGA edline ログ ファイルです。BitGen は、FPGA edline ログファイル (手動で追加/削除できるが、

ユーザーが変更する必要はない) を受け取り、EBD ファイルを生成します。生成された EBD ファイルには、優先エッセンシャル ビットとして識別されたビットが含まれます。

優先エッセンシャル ビットと共に SEM Controller を使用する

SEM Controller は、オプションとしてエラー分類機能をサポートします。SEM Controller でソフト エラーが検出された位置が特定されると、シリアル ペリフェラル インターフェイス (SPI) フラッシュ メモリに格納されている生成済みのエッセンシャル ビット データでエラーの位置がルックアップ (確認) されます。この分類機能を使用することで、ユーザーはソフト エラーがエッセンシャル ビットであるかどうかを判断できます。ザイリンクスでは、SPI フラッシュ インターフェイスを含み、SEM Controller を使用したサンプル デザインを提供しています。優先エッセンシャル ビットを使用する場合は、ルックアップ テーブルの内容が優先エッセンシャル ビット ファイルで置き換えられます。

このオプションを有効にするには、SEM Controller の **enable_classification** パラメーターを **true** に設定します。提供されているリファレンス デザインでは core.xco ファイルを使用し、パラメーターは次のように設定されています。

```
# BEGIN Parameters
CSET clock_freq=66.0
CSET component_name=sem_core
CSET correction_method=repair
CSET enable_classification=true
CSET enable_correction=true
CSET enable_injection=true
CSET injection_shim=chipscope
CSET retrieval_shim=spi_flash_x1
# END Parameters
```

SEM Controller は、リファレンス デザインの implement ディレクトリにある generate_cores.sh または generate_cores.bat スクリプトを使用して生成されます。CORE Generator™ ツールと core.xco ファイルを用いて SEM Controller を生成する場合のコマンド ラインは、次のとおりです。

```
coregen -p coregen.cgp -b sem_core.xco
```

一般的なデザインで SEM Controller コアを生成するには、CORE Generator ツールの GUI を使用する方法が最も簡単です。コアのコンフィギュレーション、使用方法、および SPI フラッシュ インターフェイスの詳細は、『LogiCORE IP Soft Error Mitigation Controller 3.1 ユーザー ガイド』[参照 1] を参照してください。

SPI フラッシュ インターフェイスを含むサンプル デザインは、SEM Controller と共に生成されます。同時に TCL スクリプト (makedata.tcl) も生成されます。このスクリプトは BitGen 出力ファイルを処理し、次の 3 つの出力ファイルを生成します。

- Intel HEX 形式のデータ ファイル (MCS) - SPI フラッシュ デバイスのプログラム用
- 生のバイナリ データ ファイル (BIN) - SPI フラッシュ デバイスのプログラム用
- 初期化ファイル (VMF) - SPI フラッシュ シミュレーション モデルのロード用

これらの出力ファイルは、次のコマンド ラインで生成されます。

```
xtclsh makedata.tcl -ebd routed.ebd datafile
```

リファレンス デザイン

このアプリケーション ノートのリファレンス デザインは、次のサイトからダウンロードできます。

https://secure.xilinx.com/webreg/clickthrough.do?cid=183797

表 1 に、リファレンス デザインの詳細を示します。

表 1 : リファレンス デザインの詳細

パラメーター	説明
一般情報	
開発者	Robert Le
ターゲット デバイス (ステッピング レベル、ES、プロダクション、スピード グレード)	Virtex-6 FPGA、 7 シリーズ FPGA
ソース コードの提供	はい
ソース コードの形式	VHDL、Verilog
既存のリファレンス デザイン、アプリケーション ノート、サードパーティ、CORE Generator ツールからデザインへのコード/IP の使用	SEM_V3_1 またはそれ以降
シミュレーション	
機能シミュレーションの実施	N/A
タイミング シミュレーションの実施	N/A
機能およびタイミング シミュレーションでのテストベンチの利用	N/A
テストベンチの形式	N/A
使用したシミュレータ	N/A
SPICE/IBIS シミュレーションの実施	N/A
インプリメンテーション	
使用した合成ソフトウェア ツール	XST、ISE® Design Suite 13.4
使用したインプリメンテーション ソフトウェア ツール	ISE Design Suite 13.4
スタティック タイミング解析の実施	はい
ハードウェア検証	
ハードウェア検証の実施	いいえ

リファレンス デザインは、生成された IP SEM Controller v3.1 のサンプル デザインに基づいています。リファレンス デザインのモジュールは、SEM Controller のサンプル デザインのモジュールと機能的に同じです。デザイン フローを説明するために、モジュール sem_mon_piso および sem_mon_sipo に Keep Hierarchy 属性を適応し、UCF でこれら 2 つのモジュールにエリア グループ制約を追加しています。リファレンス デザインでは、これらのモジュールを指定領域の例として使用し、優先エッセンシャル ビット ファイルを生成しています。

このリファレンス デザインをインプリメントする場合は、リファレンス デザインの `implement` ディレクトリにある `implement.bat` または `implement.sh` スクリプトを使用してください。

リファレンス デザインの構造は、次のとおりです。

- XAPP538/<Kintex-7 または Virtex-6>/<Verilog または VHDL>/
  - `implement` : インプリメンテーション スクリプト
  - `reference_design_source` : リファレンス デザイン用のソース コードおよび UCF
  - `sem_core` : SEM Controller コア
  - `coregen.cgp` : CORE Generator ツール用のプロジェクト ファイル

SEM Controller のパラメーターは次のとおりです。

- Component name : sem_core
- Error injection : Enabled
- Error injection shim : ChipScope
- Error correction : Enabled
- Error correction method : Repair
- Error classification : Enable
- Controller clock frequency : 66MHz (Kintex-7 デバイス) または 100MHz (Virtex-6 デバイス)

まとめ

このアプリケーション ノートでは、FIT を低減してデザインの可用性を向上させる ISE デザイン ツールのデザイン フローについて説明しました。このデザイン フローによって、影響を受けたデザイン部分に応じてシステムが適切に対応できるようになり、デザインの可用性が向上します。このフローを使用した場合、多くの反転ビットが意識的に無視されますが、これはデザインの動作に影響を与えないため、実際の FIT は大幅に低減されます。

リソース

このアプリケーション ノートは、次の補足資料を参照しています。

1. UG764 :『LogiCORE IP Soft Error Mitigation Controller 3.1 ユーザー ガイド』
2. UG360 :『Virtex-6 FPGA コンフィギュレーション ユーザーガイド』
3. UG470 :『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	変更内容
2012 年 4 月 4 日	1.0	初版リリース

Notice of Disclaimer

Automotive Applications Disclaimer

XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS APPLICATIONS RELATED TO: (I) THE DEPLOYMENT OF AIRBAGS, (II) CONTROL OF A VEHICLE, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR, OR (III) USES THAT COULD LEAD TO DEATH OR PERSONAL INJURY. CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN SUCH APPLICATIONS.

本資料は英語版 (v1.0) を翻訳したもので、内容に相違が生じる場合には原文を優先します。
資料によっては英語版の更新に対応していないものがあります。
日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com までお知らせください。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。