IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

2024 年 11 月 21 日に受理、2024 年 12 月 16 日に承認、発行日は 2024 年 12 月 20 日、現在のバージョンの日付は 2025 年 1 月 7 日。

デジタルオブジェクト識別子 10.1109/ACCESS.2024.3520877

RESEARCH ARTICLE

フェイルセーフロジック設計戦略 最新のFPGAアーキテクチャ

プリヤ•A•バクタ [1]、（IEEE学生会員）、ジム•プラスクエリック [1]、
ANDREW SUCHANEK 1ニューメキシ [2]、（IEEEシニア会員）、TOM J. MANNOS [2]
コ大学電気•コンピュータ工学部、アルバカーキ、NM 87131、米国2サンディア国立研究所、アルバカーキ、NM 87123、米
国連絡先著者: Jim Plusquellic (jplusq@unm.edu)

この研究は、研究所主導研究開発（LDRD）を通じて研究開発によって部分的に支援されました。
サンディア国立研究所のプログラムは、助成金 LDRD 225963 の下、また一部はサンディア国立研究所によって運営されており、契約 DE-NA0003525 の下、ハニー ウェル インターナショナル社の 100% 所有子
会社である National Technology & Engineering Solutions of Sandia LLC (NTESS) によって、米国エネルギー省の国家核安全保障局 (DOE/NNSA) のために管理および運営されている多目的研究所です。

要約：フェイルセーフコンピューティングとは、障害発生時に非動作安全状態に戻るコンピューティングシステムを指します。本稿では、フィールドプログラマブルゲートアレイ（FPGA）に対するシングルイベントアップセット（SEU）およびフォールトインジェクション攻撃の緩和策として回路レベルの手法を検証し、暗号化アルゴリズムのフェイルセーフモニターとしての有効性を分析します。ルックアップテーブル（LUT）やプログラマブルインターコネクトポイント（PIP）などのFPGAプリミティブを介したフォールトの影響の伝播を、オープンソースツールを使用して作成されたFPGAアーキテクチャ内で評価し、FPGA上でのフォールトインジェクション実験によって検証しました。分析の結果、再構成可能なアーキテクチャには、同等のフェイルセーフな特定用途向け集積回路（ASIC）よりも多くの脆弱性が存在することが明らかになりました。そのため、冗長回路とチェックロジックのより精巧なネットワークが必要になります。FPGAのLUT内で配線を構成し、ロジック機能を指定するコンフィギュレーションメモリビット（CMB）は、追加のフォールト状態とフォールト伝播パスを導入することで、フェイルセーフ設計戦略を複雑化させます。リソース効率の高いフェイルセーフ回路設計手法として、再構成可能システムにおけるフェイルセーフのための設計設計（DEFCON）を提案する。DEFCONの利点と限界について、シミュレーションおよびFPGAハードウェアで実施されたフォールトインジェクション実験の文脈で説明する。

索引用語DEFCON、比較による複製、フェイルセーフ、フォールト インジェクション攻撃、シングル イベント アップセット、フォールト トレランス、動的部分再構成。

I. はじめに

FPGAは、宇宙線によって引き起こされるシングルイベントアップセット（SEU）や、フォールトインジェクション攻撃を介した悪意のある攻撃の影響を受けやすい。自然発生的なSEUは、重要なシステム機能を実行するデバイスの信頼性を脅かす。

一方、攻撃者は暗号鍵などの機密情報を盗む目的で意図的にSEUを導入し、実装された設計へのアクセスをロック解除するために障害を注入する可能性があります。これらの信頼性の問題に対処し、FPGAのデータ整合性を保護するには、効果的な組み込み信頼性およびセキュリティ監視機能が不可欠です。

この原稿の査読を担当した副編集者と
出版を承認したのはジャン•ドメニコ•リカルドだった。

FPGA設計に統合できます。本研究では、DEFCON（DEsign for Fail-safe in reCONfigurable systems）と呼ばれるフェイルセーフ回路技術を提案します。これは、障害を検出し、障害発生時にシステムを安全な動作状態に戻すように設計されています。

フェイルセーフシステムは、設計の速度、電力、面積を最適化しながら、冗長性と自己診断機能を組み込む必要があります。障害検出時のデフォルトの動作は、アラームを鳴らし、システムを停止し、出力を安全な動作状態にすることです。関連する例として、車両の自動運転機能を実装する制御システムがあります。このシステムでは、障害検出により自動運転システムが停止し、車両を停止するか、人間のドライバーに制御が引き継がれます。複雑なシステムでは、

システム アーキテクチャ、障害の検出、アラームの信号、およびシステムを安全な状態に戻すための緩和プロセスでは、ほぼ常に冗長コンポーネントの挿入が必要になります。

フェイルセーフパラダイムの目標は、フォールトトレラントパラダイムの目標とは異なります。フォールトトレランスとは、単一の障害によってシステム内の一部のコンポーネントが故障した場合でも、システムが完全な機能で動作を継続できる能力を指します。この基準を満たすフォールトトレラントシステムは、しばしばフェイルオペレーショナルと呼ばれます[1]。最も一般的なフォールトトレランス技術は、システムを三重化した三重モジュラー冗長性（TMR）を用い、多数決メカニズムを用いて、デバイスの出力を、一致する冗長コピーのうち2つが生成した値に強制的に設定し、一致しない冗長コピーは無視します。この場合、システムは障害が発生しても正常に動作を継続できます。例えば、TMRは航空機の飛行制御システムの重要なコンポーネントの障害に対する保護を提供するために一般的に使用され、3つのコピーのうち1つが故障しても航空機は事故なく運航できます。

対照的に、フェイルセーフ設計手法は、原子力発電所や核兵器の制御システムなど、故障に対する許容度がゼロで、高度な故障回復メカニズムが必要となるシステムに典型的に採用されます。この場合、フェイルセーフシステムのデフォルトの動作は、警報を鳴らし、システムを安全な状態に戻すことです。これにより、既知の安全な開始状態から是正措置を実行でき、システム動作の確実かつ正確な復旧が保証されます。

冗長性は、フェイルセーフ設計において障害検出にも利用されます。しかし、緩和機能は通常、別のコントローラに委任され、検出のみが必要なため、比較を伴う複製（DWC）と呼ばれる、よりリソース効率の高い手法を使用できます[2]、 [3]。TMRとは対照的に冗長コピーの数を2つに減らし、多数決コンポーネントを排除することで、フットプリントが小さくなり、フェイルセーフシステムの復元力が向上します。フェイルセーフシステムの緩和コンポーネントはフットプリントのサイズを大きくするため、この利点が部分的に相殺されることに注意してください。ただし、限界的には、緩和コンポーネント（例えばハードウェアリセット）が非常に小さい場合があり、その場合、フェイルセーフシステムは同等のTMRシステムと比較して小型になります。

残念ながら、緩和戦略によって実行されるアクションはアプリケーションに依存しているため、DWC ベース システムと TMR ベース システムの全体的なサイズに関して結論を導き出すことは困難です。

本研究の目標は、FPGAを用いたフェイルセーフシステム設計の有効性を実証し、検出回路と警報発生回路の構造を最適化することで、実装面積を最小化することです。FPGAではASIC実装に比べて実装がはるかに困難ですが、新しい技術の登場（例えば、新しい量子暗号アルゴリズムが新たな標準として採用されるなど）に合わせて現場でアップデートできるという利点は、FPGA上でフェイルセーフシステムを設計するためのソリューションを見つけるための大きな推進力となります。

提案された DEFCON 技術の主な特徴は、1 つの FPGA LUT に 2 つの XOR チェッカー回路をインスタンス化し、2 つのコピーのうち 1 つを無効にする可能性がある障害があっても、両方の XOR チェッカー ゲートが適切に機能することです。

XORチェッカーゲートは、DWCやTMRなどの冗長性ベースの技術において、監視対象の機能ユニット出力の2つの冗長コピーが同一かどうかを判定するためのコンパレータとして頻繁に使用されています。XORチェッカーネットワーク自体で発生する障害に対処するため、機能ユニットで監視される出力ビットのペアごとに、XORチェッカーの2つのコピーが挿入されます。したがって、XORチェッカー回路コンポーネントはフェイルセーフシステムの障害検出回路において主要なコンポーネントであり、これらのビット単位のチェッカーのサイズを大幅に削減しながら、障害に対する独立性を維持するように設計された技術は、重要な貢献を果たします。

フェイルセーフ システムに関連する特性と要件により、自動化された回路設計フローを使用して面積のオーバーヘッドを最小限に抑えながら設計を実装することが困難になります。
本稿では、オープンソースのFPGA合成ツールであるOpenFPGA [4], [5]を用いて、FPGAにおける回路レベルのフェイルセーフ設計について検討する。再構成を可能にするFPGAプログラミング機能は、冗長コンポーネントの独立性を確保することを困難にする。例えば、未使用のプログラマブルリソースを介した障害伝播の可能性を考慮する必要がある。さらに、構成状態を乱すような障害メカニズムは、設計の論理機能や配線特性を変化させる可能性があるため、監視回路の自己チェック機能はより複雑になる。

まず、オープンソースのFPGA設計ツールを使用して、故障の注入と伝播を調査します。故障伝播経路の解析には、基盤となる再プログラム可能な回路構造に関する詳細な知識が必要であり、そのような情報は商用FPGAのエンドユーザーには提供されていないためです。次に、DEFCON技術をZynq 7010 FPGAに実装し、故障から保護する機能として、高度暗号化規格（AES）アルゴリズムのDWCバージョンと統合します。シミュレーションとハードウェア実験の両方の結果を示し、DEFCON回路の利点と限界を示します。本研究の貢献は以下のように要約されます。

- オープンソース FPGA と商用 Xilinx Zynq アーキテクチャの両方で DEFCON と呼ばれるフェイルセーフ DWC 技術を実装および分析します。監視対象の機能ユニットの出力と DEFCON 回路自体の両方が、単一の障害発生に対して個別に保護されます。
- XORチェッカー回路のリソース最適化バージョン。冗長XORゲートの両方のコピーをFPGA上の1つのルックアップテーブル（LUT）にインスタンス化することで、チェッカー回路に関連するオーバーヘッドを大幅に削減します。このXORチェッカー構築技術は、各LUT内に実装された2つのXORゲートで使用される2組の入力の故障独立性を保証します。

- クアッド冗長性と呼ばれる階層型冗長方式。これにより、6入力LUT実装の冗長XORゲートが実現され、DEFCON回路自体のコンポーネントに障害が発生した場合でも、システムの回復と継続的な動作をサポートします。DEFCONモニター内で発生する障害は重大ではないとみなされるため、即座に修復する緩和戦略が提案されています。
- シミュレーションとハードウェア実験の結果から、DEFCON技術のリソース使用率、有効性、限界が明らかになりました。さらに、最も関連性の高いフェイルセーフ技術のFPGA実装を実施し、その結果をDEFCONの結果と比較しました。

シミュレーションとハードウェア実験で使用したフォールト インジェクション戦略では、FPGA のコンフィギュレーション メモリ ビット (CMB) にのみフォールトが導入され、機能ユニットで使用されるフリップフロップ (FF) にはフォールトが導入されないことを指摘しておきます。これは、機能ユニットのステート マシンまたはデータ パスのメモリ コンポーネントに格納された値を変更するフォールトは、冗長コピーの 1 つによって計算された結果を変更し、DEFCON モニタによって常に検出されるためです。一方、CMB で発生するフォールトは、機能ユニットの回路実装特性を変更する可能性があり、後述するように予期しない動作を引き起こす可能性があります。したがって、CMB フォールトは、あらゆるタイプの FPGA ベースのフェイルセーフ回路設計戦略において評価する必要がある重要なコンポーネントです。

II. 背景

FPGAにおけるフェイルセーフ技術を評価する際には、オーバーヘッドコストと検出能力という2つの重要な指標が考慮されます。FPGAモジュール内の冗長性は、エラー検出コードアルゴリズムと比較して、構成メモリ内の障害検出において優れた性能を示し、オーバーヘッドも小さいことが示されています[6]。 [6]の研究では、AESアルゴリズムの置換ボックス（S-Box）で評価された5つの障害検出戦略が示されており、特に重複比較（DWC）戦略と三重モジュラー冗長（TMR）戦略が注目されています。予想通り、DWC方式は保護されていない設計に比べて約2倍のリソースを使用し、TMRではリソース使用率が約3倍に増加します。著者らは、オーバーヘッドと堅牢性の点で、DWCが全体的に優れた障害検出技術であると結論付けています。

[2]と[3]では、FPGAのフェイルセーフシステム設計に適したDWC手法が提案されており、フリップフロップ（FF）、ブロックRAM（BRAM）、およびコンフィギュレーションメモリに格納されたビット値の状態の変動を検出する。この手法では、機能ユニットとアラーム信号を複製し、複製された信号成分（例えば、プライマリ出力と設計のフィードバックパス内の出力）の差異をフラグ付けする自己チェック型コンパレータを組み込んでいる。このフェイルセーフ回路設計手法（以降BYU法と呼ぶ）は、DEFCONに最も関連性の高い手法である。我々は、BYU メソッドを FPGA 上で実行し、セクションVで比較分析を示します。

フェイルセーフ動作を目標とした公開技術は非常に限られています。ここでは、FPGAベースのフォールトトレランス技術の中で、最も関連性の高い技術について説明します。前述のように、フォールトトレランス手法はフェイルオペレーショナル（故障動作型）と特徴付けられ、提案される回路設計技術の目的は、故障が発生してもシステムの動作を維持することです。

FPGAベースの手法では、故障マスキング技術を用いて故障システムの平均故障時間（MTTR）を延長しようとするものがいくつかある（例えば[7]および[8]）。 DEFCONと同様に、Leeら[7]は重複した論理関数を単一のLUTにエンコードし、両方のLUT出力を使用することを提案している。

2 つの複製された出力は、それぞれ 0 対 1 および 1 対 0 のシングル イベント アップセット (SEU) の障害をマスクする手段として、下流の LUT で AND または OR されます。
整数線形計画法は、故障率を最小化するための最適な複製および符号化方式を最終的に決定するために提案されている。 [8]では、コンフィギュラブルロジックブロック内に埋め込まれたキャリーチェーンコンポーネントをさらに活用する関連方式が提案されている。ここでは、ロジック関数は2つのサブ関数に分解され、キャリーインをそれぞれ0と1に制御することで、ANDゲートまたはORゲートを使用してキャリーチェーン内で結合される。この手法は、 [7]のように下流のANDおよびOR関数が必要となる場合とは異なり、配置配線の変更を必要としないため、配線の混雑が軽減される。DEFCONとは異なり、これらの手法はどちらもLUTロジック関数を符号化してLUTへの入力における故障独立性を確保しようとはしない。

[9]の著者らは、配線リソース用のCMBがFPGA内のCMB総数の90%を占めており、そのためSEUに対して非常に脆弱であると判断しました。

さらに、ルーティングCMBを混乱させるSEUの約10%は、TMRの単一故障仮定に違反するため、TMRでは修正できないPIP上の複数のショート故障およびオープン故障を生成することを発見しました。彼らは、単一のPIP故障による複数のエラーがTMRの有効性に影響を与えるのを防ぐ、信頼性重視の配置配線アルゴリズムを提案しています。

[10]の著者らは、FPGA LUTとその相互接続におけるSEUの特性を明らかにしている。LUT SEUは特定のセルが選択された場合にのみ障害を引き起こすが、コンフィギュレーションロジックブロック（CLB）はCLB内配線が完全に接続され、MUXベースであるため、SEUによって無関係な信号が選択され、障害が発生する可能性がある。CLB間配線は通常、双方向パストランジスタを介して相互接続されるが、SEU切断（オープン）障害は通常、一時的なスタックアット0/スタックアット1障害としてモデル化され、下流ゲートのクローバー電流を防ぐためにハイまたはローにプルダウンされる。

SEUショート故障は隣接する2本の配線をブリッジする故障で、2つのドライバが反対の値を入力した際に下流ゲートがそれらを異なる方法で解釈するため、特性評価が最も複雑です。ファンアウトするネットはモデルをさらに複雑にし、複数の故障を注入する機会をもたらします。

現代のFPGAは非常に大きなコンフィギュレーションビットストリームを持っているため、トラバーサルスクラビング技術を用いたエラー検出にかかる時間が長くなります。 [11]では、DWCで保護された重要な回路とコンフィギュレーションフレーム間のマッピングを利用する高速スクラビング技術が提案されています。

[12]の研究では、商用FPGAにおけるハードウェアエミュレーションによるフォールトインジェクション技術の課題が提示されている。
ハードウェア障害エミュレーション技術は、ビットストリーム内の各ビットを反転させて復元する必要があるため、網羅的かつ時間がかかります。さらに、デバイスがロックアップし、電源を完全に入れ直す必要がある場合もあります。著者らは、再構成可能なSoCデバイスを用いて、高速化されたオープンソースの障害注入プラットフォーム「ParFlip」を構築する手法を提示します。

[13]では、フォールトインジェクションベースの対策が提案されている。
著者らは、AESエンジンの重要なコンポーネントをフォールトインジェクション攻撃から監視するためのパリティベースの手法を提案する。フォールトは各バイトに挿入され、パリティベースの手法を用いて暗号化の各演算およびラウンド後に監視される。AESエンジンの各演算の出力は、XORロジックを用いてパリティ出力と比較され、差異があればフラグが付けられる。この手法は、各ラウンド内の各計算ステップの後に出力を評価することで広範なフォールトカバレッジを確保し、単一のフォールトに対する堅牢な検出を可能にする。

[14]では、別のハードウェア冗長性ベースの故障検出戦略が提示されています。この研究では、面積効率とオーバーヘッドコストを維持しながら、SHA-512ハッシュアルゴリズムのフォールトインジェクション攻撃に対するセキュリティを強化する故障検出戦略を提案しています。このアプローチでは、DWCメソッドを使用していますが、機能ユニットの出力全体を複製するのではなく、オーバーヘッドを最小限に抑えるために重要なコンポーネントのみに焦点を当てています。さらに、SHA-512操作を異なる時間に繰り返し比較する通常の時間冗長性と、後続のステップで結果を減算することで正確性を検証する逆時間冗長性の両方を使用して時間冗長性が適用されます。提案されたスキームは、Xilinx Virtex-II Pro FPGAでテストされ、私たちの研究と同様に、最大20の故障ビットを入力データにランダムに導入して故障検出スキームを評価する故障注入シミュレーションが実行され、99.99%の故障カバレッジを達成しました。結果は、周波数が 1.39% 減少し、面積が 2.94% 増加し、スループット保持率が 98.61% であることを示しました。

A. フェイルセーフ、フォールトトレランス、フォールトインジェクション攻撃への統一的なアプローチ

FPGAフレームワーク内でフェイルセーフシステム設計、フォールトトレランス、フォールトインジェクション攻撃を個別に扱うために設計された方法論は、統一されたアプローチを用いて3つの領域すべてに対処できるように範囲を拡張することができます。例えば、これらの領域はすべてSEUに関係しており、それが自然に発生するか、電源電圧やクロックのグリッチによって誘発されるか、電磁放射や

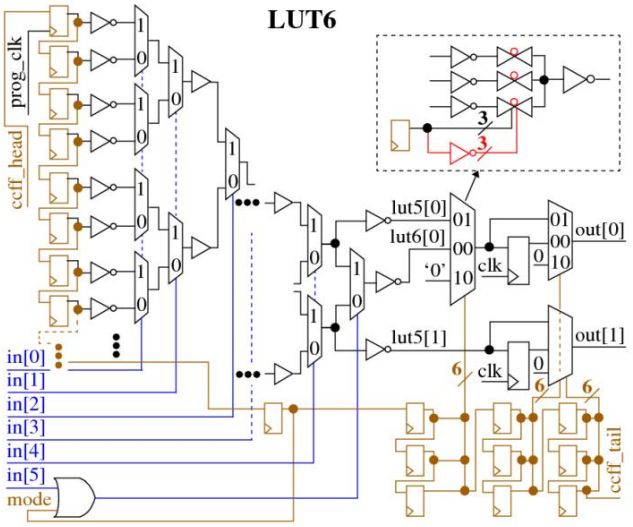

図 1. CLB 内にレジスタ ロジックを備えた OpenFPGA 6 入力 LUT。

レーザー。フェイルセーフ攻撃やフォールトインジェクション攻撃の場合、アラームを発するDWC方式が適切です。いずれの場合もシステムの動作を停止する必要があり、その後、非動作状態を維持し、システムを再起動し、当局に警告するなどの是正措置を講じる必要があるためです。

フォールトトレランスではシステムが継続的に動作することが必要であるが、障害状態にフラグを立てるDWC技術はここでも使用することができ、例えば、未使用の冗長コピーをダウンタイムを最小限に抑える手段として利用できるようにするなどである[15]。
これらのシナリオのいずれにおいても、耐性を向上したり対策を講じたりする技術は、元の設計の速度、消費電力、面積への影響を最小限に抑え、保護回路自体で発生する混乱の影響を最小限に抑えるために、フットプリントを小さく抑える必要があります。DEFCONの主な目標は、FPGA設計においてこの目標を達成することです。

III. OPENFPGAフレームワーク

本稿では、シミュレーションベースの実験と解析[4]、 [5]のためのFPGAアーキテクチャを作成するためにOpenFPGA合成フレームワークを使用しています。Pythonベースのツールフローは、XMLベースのアーキテクチャ記述を使用して、CLB、LUT、FF、配線アーキテクチャ、I/O、およびカスタムハードワイヤードIPブロックの詳細を指定します。OpenFPGAは回路記述を受け取り、回路要素を収容するのに十分な大きさのFPGAアーキテクチャを作成します。OpenFPGA内では、Verilog-to-routing（VTR）CADツール[16]を使用して、Verilogネットリストと回路記述を実装するプログラミングビットストリームを生成します。その後、ネットリストは、標準セルライブラリベースの配置配線（PNR）CADツールフローを使用してレイアウトに処理できます。

OpenFPGA合成ツールは幅広いコンポーネントをサポートしていますが、提案するDEF-CON技術の開発では、CLB、配線コンポーネント、およびI/Oのみを使用します。具体的な実装特性は、

故障の影響と故障伝播を完全に評価するには、LUTとローカル配線ネットワークを含むCLBと、グローバル配線ネットワークを定義するPIPが必要です。このセクションでは、ディストリビューションに付属するFPGA構成テストケースの例を用いて、OpenFPGAによって作成された回路構造について説明します。テストケースには、1つのCLBに収められた10個の6入力LUT、4つの接続およびスイッチボックス、そして32個のI/Oが含まれています。

6入力LUT（LUT6）の回路図を図1に示します。
プログラミング ビットストリームは、コンフィギュレーション チェーン (cc _head) を使用したロジック関数で左側の FF の列と、茶色で強調表示されたその他の要素を構成します。
ルックアップテーブルは、トランスミッションゲート型の2対1 MUXのシーケンスを用いて実装され、MUXの2段ごとに増幅バッファ列が挿入されています。LUT6は、out[0]にlut5またはlut6の上位機能、out[1]にlut5の下位機能を提供するようにプログラムできます。2つの出力は、オプションでレジスタリングできます。

LUT6の上部と下部にある2対1 MUXの2つのシーケンスに関連付けられた円錐形の回路構造により、共有入力を除いてout[0]とout[1]出力は構造的に独立します。これは、提案されたDEFCONスキームで活用される特性です。

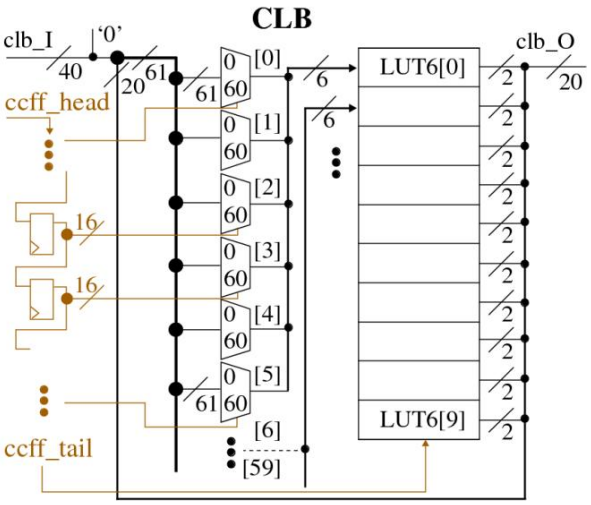

図 2. OpenFPGA によって作成された、10 個の LUT6 とローカル ルーティングを備えた CLB アーキテクチャ。

CLBアーキテクチャを図2に示します。構成可能な配線ネットワークにより、20個のLUT6出力のいずれかを、CLB内の60個のLUT6入力のいずれかにローカルに接続できます。LUT6入力を駆動する61対1のMUXは、他のCLBおよびI/Oからの40個の外部入力と定数「0」を構成オプションとして追加します。CLB内のすべての配線はファンアウトフリーであり、回路構造において再収束ファンアウトのインスタンスは存在しません。再収束ファンアウトとは、入力が論理ゲートネットワークにファンアウトし、出力に近いゲート入力で下流に再収束する回路トポロジを指します。再収束ファンアウトがないため、フェイルセーフ回路の設計の複雑さが軽減されます。

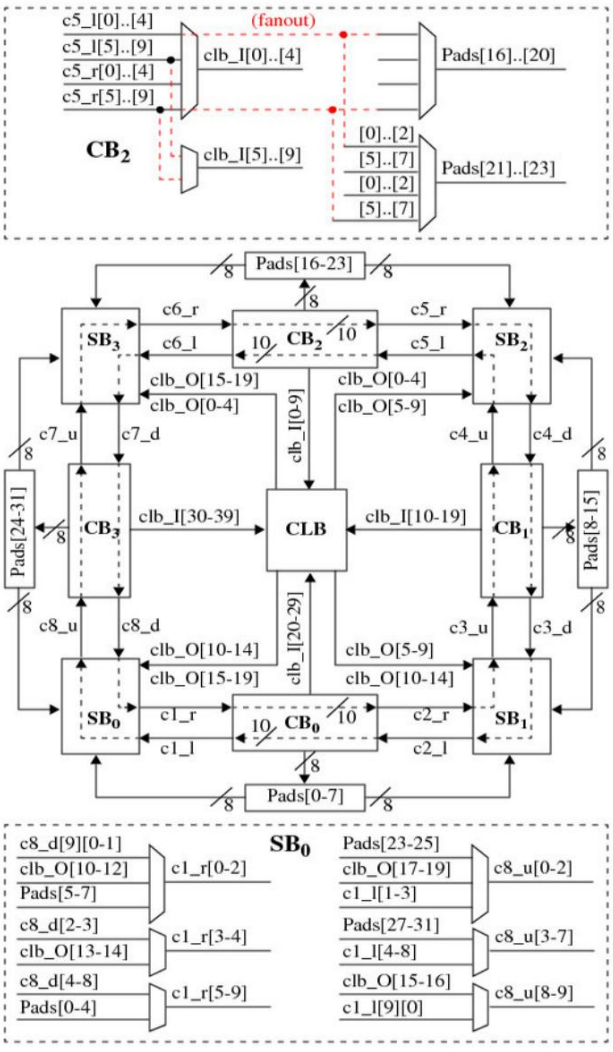

図 3. OpenFPGA によって作成されたルーティング アーキテクチャ。

シミュレーションで使用したFPGAアーキテクチャを図3に示します。この図は、4つの接続ブロック（CB0、3）とスイッチブロック（SB0、3）に囲まれた単一のCLBを示しています。図中のCB2とSB0のMUXの詳細は、信号ファンアウトがCBに実装されていること、そしてCBとSBはCLB内で提供される完全に相互接続された配線ネットワークとは対照的に、配線オプションの一部しか提供していないことを示しています。I/Oブロックには、CBから駆動される出力パッドと、SBを介して内部配線ネットワークに入る入力パッドが含まれています。MUXへの選択入力を制御するコンフィギュレーションチェーンと、MUXの実装詳細は、図を分かりやすくするために省略しています。

これらのアーキテクチャの詳細は、障害の影響を理解する上で重要であり、次のセクションの例で説明します。

A. DEFCONの実装
オープンソースFPGA

このセクションでは、DEFCON回路設計とFPGAへの実装について述べる。その有効性を評価する。

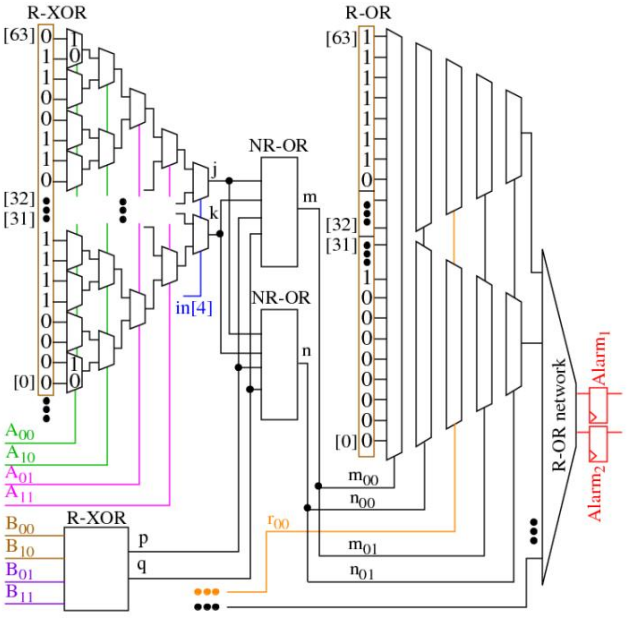

図4.提案されたDEFCONフェイルセーフ回路設計。

シミュレーション実験を用いてフェイルセーフ動作を提供すること、すなわち、保護対象の機能ユニットの出力およびモニタ自体の故障を検出できるかどうかを検証します。この設計は、1クロックサイクル以上アクティブなスタックアット故障およびCMB故障を確実に検出し、そのような場合には、故障発生後最大2クロックサイクルでアラーム信号がアサートされることを保証します。

1) XORチェッカーコンポーネント提案する

DEFCON回路の回路図を図4に示します。DWC回路は、冗長2入力XORゲート（R-XOR）のセットで構成され、各ゲートは冗長信号ペアA00とA10を監視するように設計されています。A00とA10は、 DEFCONが保護するように設計された信号を表す冗長機能ユニットペアの同じ1ビット出力に接続されます（図示せず）。機能ユニットは、例えば、車両内の冗長電子制御ユニット（ECU）ペアであり、出力信号はアンチロックブレーキシステムや運転支援機能への制御信号である可能性があります。

入力A01とA11は、機能ユニットの出力に再び複製された、冗長化された2つの機能ユニット出力です。この4重化ベースの複製戦略の根拠については、以下のセクションで説明します。

図4の左上にあるR-XOR LUT6は、2入力XORゲートの特殊バージョンを実装するようにプログラムされています。前述のように、MUXネットワークの構造は、図1のout[0]信号とout[1]信号（ここではjとkと表記）に対して2つの独立したロジックコーンを定義しますが、LUT6は通常、LUTの入力の故障依存性や独立性に関係なくプログラムされます。例えば、 LUT6内のA00に故障が発生した場合、LUT6のCMBを特別な方法でプログラムしない限り、両方のロジックコーンが影響を受けます。

特定の入力が両方のロジックコーンに影響を与えないようにするため、LUT6のCMBは、2入力XOR真理値表関数が複数回複製されるようにプログラムされています。ここでは、CMBの上位32ビットは、2入力XOR関数の出力パターン「0110」を繰り返します。最初の8つのセルのみが表示されていますが、実際にはこのパターンは上位32ビットのCMBにわたって「0110 0110 0110 0110 0110 0110 0110 0110」として8回複製されています。CMBの下位32ビットも2入力XORを実装するようにプログラムされていますが、各真理値表出力値は連続するCMBセルに4回ローカルに複製されます。 CMB の下位半分の完全なビット シーケンスは、「0000 1111 1111 0000 0000 1111 1111 0000」です。

この特定の構成ビット パターンにより、図に示す 4 つの入力に対する 2 つのロジック コーン出力の依存性がなくなり、1 つの LUT で完全な自己チェック コンパレータ (つまり、監視対象の重複する機能ユニット出力信号ごとに 1 つの LUT) を実装できるようになります。ここでは、入力A00とA10のみが上部コーンの機能出力値に影響し、入力A01とA11のみが下部コーンの出力に影響します。たとえば、{A00、 A10}が ''01'' で{A01、 A11}が ''01'' (つまり、機能ユニット出力で障害が発生し、 A00で 2 つ目の 1 スタック障害が発生する(上部コーンの入力が ''11'' になったため上部コーンが無効になる) 場合、 A01とA11 が独立した XOR 関数の入力を駆動するため、下部コーンはkに '1' を伝播し続けます。

これは、下側コーンのプログラミングビットパターンが、 A00およびA10入力に関連付けられた4つの選択すべてに「1」を複製し、これらの入力がA01およびA11入力で選択された論理値とは無関係になるためです。したがって、この障害をマスクすることはできません。

また、提案された冗長化方式はCMBで発生する障害も検出することに注意してください。例えば、CMBがいずれかのロジックコーンのXORパターンシーケンスの1つで「1」から「0」に反転し、監視対象の機能ユニット出力の1つで2つ目の障害が発生した場合、冗長化されたロジックコーンの1つが常にその障害を検出し、 j出力またはk出力のいずれかに「1」を生成します。

単一故障の仮定の下では、フェイルセーフ特性を維持しながら、代替プログラミングパターンが可能です。例えば、上半分のパターン「0110 xxxx xxxx 0110 0110 xxxx xxxx 0110」も有効です。ここで、 xは「don't care」を表します。これは、 A01とA11に故障がないと仮定すると、パターン「00」と「11」のみが考えられるためです。A00またはA10に故障が発生した場合、 A01とA11によって2つのCMB領域のうち最初または最後の領域が選択されます。ただし、完全に複製されたXORパターンを指定することの利点は、入力ペアごとに1つずつ、合計2つの故障を検出できることです。

図1に示すように、 lut5[0]信号は追加の3対1 MUXを通過します。in [5]およびmode信号のDEFCON構成は図1から「1」であり、3対1 MUXの選択入力では「01」です。in [5]またはmode入力信号によって駆動されるORゲートの出力のスタックアット0障害は、3対1 MUX構成によってマスクされ、無害化されます。3対1 MUX選択信号上のCMB障害は、in[5]およびmode入力信号が「00」であるため、障害ケースでは無視されます。

信号割り当ては、''01'' と ''00'' の両方の入力で上位ロジックコーンの出力を選択します。一方、CMB 障害によって ''10'' 入力でハードワイヤードの '0' が選択された場合、その障害は上位コーンでマスクされます。ただし、下位コーンは監視対象の機能ユニットの出力で発生した障害を検出します。つまり、このシナリオでは 2 番目のチェッカー障害のみがマスクされます。

LUT6の2つの出力は、図1の右端に示すように、オプションでレジスタに記録されます。2対1選択MUXのフェイルセーフ設定は0で、lut5信号はレジスタをバイパスします。出力MUXに関連付けられたCMBに障害が発生し、 out[0]またはout[1]のいずれかが強制的にレジスタに記録された場合、アラーム信号は1クロックサイクル遅延されます。ただし、前述のマスキングシナリオと同様に、障害のない出力は常に機能ユニットの障害を即座に検出します。

2) 警報信号圧縮ネットワークR-XORゲートからの出力信号
は、非冗長OR（NR-OR)ゲートを用いてOR演算されます。図4の中央には、NR-ORゲートとして構成された2つのLUTが示されています。

R-XORゲートとは異なり、NR-ORゲートの冗長入力は2つの異なるLUTに配置されています。この冗長化により、最大3つのR-XOR出力信号の出力ペアを2つのLUTのみで結合（または収集)することができ、3対2の圧縮率を実現します。

対照的に、図4の右端に示されている OR ゲートの冗長バージョン (R-OR) は、 2.5 対 2 の圧縮率を実現するために、2.5 個の R-XOR ゲート出力ペアのみを監視できます。
これは、R-ORバージョンではlut5出力が両方とも使用されるため、6つの入力のうち5つしか使用できないためです。したがって、NR-ORバージョンはシステム全体のオーバーヘッドを削減します。ただし、どちらのバージョンでも、LUT内部で2つ目の障害（ノードのスタックやCMBビット反転など)が発生した場合でも、機能障害が発生した場合は、アラーム出力信号の少なくとも1つがアサートされることが保証されています。

図4の右側に示すR-ORゲートのCMBプログラミングシーケンスは、 R-XORゲートとは2つの点で異なります。1つ目はORロジック機能を実装していること、2つ目は5番目の入力を処理するように設定されていることです。上側のコーンには「11111110 111111110 111111110 11111110」というシーケンスがプログラムされ、下側のコーンには「11111111 11111111 11111111 00000000」というシーケンスがプログラムされています。図には両方のシーケンスの一部が示されています。これらのシーケンスは、2つの故障に依存しない機能、つまり上側のコーンに3入力ORゲート、下側のコーンに2入力ORゲートを実装しています。

NR-OR および R-OR ネットワーク コンポーネントは、3 対 2 および 2.5 対 2 の圧縮機能の階層を定義します。これにより、最終的にすべての R-XOR 冗長出力が 2 つのアラーム信号 (Alarm1および Alarm2)に結合されます(図 4 の右端を参照)。階層内のゲート数とレベル数は、監視される機能ユニット出力信号の数と、アラーム信号の圧縮に使用される NR-OR ゲートと R-OR ゲートの数によって異なります。

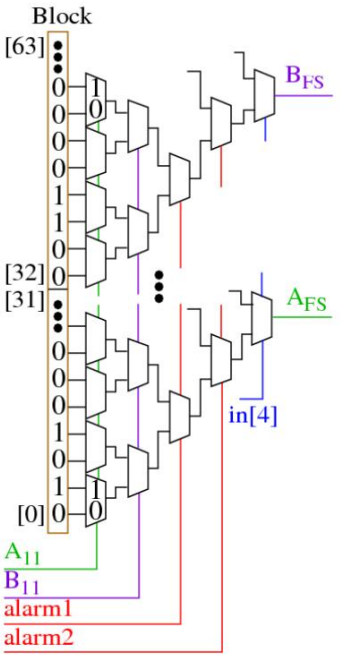

図5. DEFCON機能ユニット出力ブロッキング設計。

3) DEFCONにおける出力信号制御DEFCONには、障害が
検出され、アラーム信号のいずれかまたは両方がアサートされると、機能ユニットの出力信号を強制的にフェイルセーフ状態にするメカニズムが組み込まれています。以下では、機能ユニット出力のフェイルセーフ状態は「0」であると仮定しますが、任意のビットパターンが可能です。

2つの機能ユニット出力AとBの出力ブロッキング回路を図5に示す。LUT6 （ブロックと表記)は、ここでもデュアルLUT6出力を利用する。冗長機能ユニット出力信号の1つのコピーは、2つのアラーム信号とともにLUT6入力に送られる。真理値表関数は、両方のアラームが「0」の場合にはA11とB11の現在の状態を渡し、それ以外の場合にはlut5出力を「0」に強制するようにコード化されている。[4]でラベル付けされた入力は、後述する4重冗長方式で使用される。

結果のセクションで示すように、機能ユニット出力で 1 つの障害が発生し、2 つ目の障害が DEFCON モニターで発生するという二重障害状態では、DEFCON が機能ユニット出力信号をブロックできないケースが少数あります。特に、シミュレーション実験では、アラームの 1 つまたは両方がアサートされているにもかかわらず、機能ユニット出力が非フェイルセーフ値 (実験では「1」) で固定される DEFCON モニター障害が少数あることを確認しました。同様に、どちらのアラームもアサートされていない場合でも、フェイルセーフ出力信号 (例: AFS) にエラーがある場合がいくつか発生します。残念ながら、この 2 番目のシナリオでは、ブロック ゲートの完全冗長バージョンを追加しても、どの冗長出力が正しい機能ユニット出力値を表しているかを判断できません。

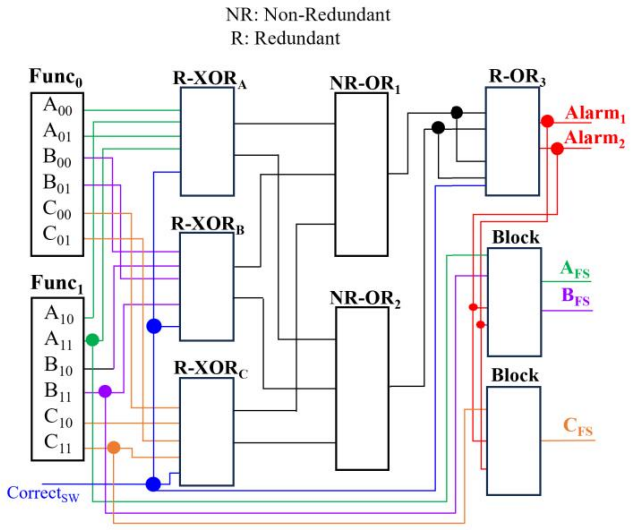

図6. 3つの機能ユニット出力信号ペアのフェイルセーフチェックによる検証のシミュレーションモデル。

4) DEFCON の 4 重冗長性[2]および[3]で提案されているアーキテクチャに対する本提案アーキテクチャの顕著な利点は、各 R-XOR LUT 内に存在する 4 重冗長性です。4 重冗長性により、DEFCON は実行時に DEFCON モニターの R-XOR LUT で発生する可能性のある SEU を軽減できます。たとえば、アラーム信号の 1 つだけがアサートされている場合、これは DEFCON モニター自体の CMB で障害が発生したことを示しています。図4および5の [4] でラベル付けされた信号をアサートして、2 つの LUT サブ機能のそれぞれで冗長コピーに切り替えることによって、つまり LUT6 の上部コーンの CMB ビット {16} ～ {31} と下部コーンの {48} ～ {63} に切り替えることによって、CMB 障害が発生したかどうかを確認できます。

アサートされたアラーム信号が 0 に戻ると、提案されたフェイルセーフ システムは完全なフェイルセーフ検出機能を備えて動作を継続し、CMB の修理を後まで延期することができます。

B. シミュレーション実験のセットアップ

本解析で考慮する故障は、CMBビットのアップセットである。PIPのスタックオープン故障およびスタッククローズ故障は、PIPを制御するCMBのスタックアット故障と同等であることに注意する必要がある[17]。独立した配線セグメント間のブリッジ故障（PIP経由で接続できないもの）は、本解析では考慮しない。さらに、OpenFPGAアーキテクチャのPIPコンポーネントは単方向であるため、双方向PIPの使用に伴う複雑な故障伝播挙動を考慮する必要がない[11]。

シミュレーション実験で使用したテスト回路構成を図6に示す。機能ユニットの2つの冗長コピーからの出力は、 Func0とFunc1とラベル付けされている。DEFCON回路構成は、機能ユニットからの3つの1ビット冗長出力信号を監視するために構築されている。冗長出力信号A、B 、 Cは、各信号はFPGAの入力で複製され、FPGA内の経路における単一障害点を排除します。12個の信号はすべて、図3のOpenFPGAアーキテクチャに示すように、I/Oパッドを介してFPGAに入ります。これは、FPGAを2つの同期されたマイクロプロセッサ（それぞれ独立したデバイス）のボードレベルモニタとして使用するシステムアーキテクチャ、または機能ユニットが組み込まれたアーキテクチャをモデル化します。

FPGA アーキテクチャ自体の中で。

DEFCON 実装では、CLB 内に含まれる 10 個の LUT アレイから 6 個の LUT を利用し (図 3 を参照)、前述の CMB シーケンスを使用して 3 つの R-XOR ゲート、2 つの NR-OR ゲート、および 1 つの R-OR ゲートを作成します。図6 に示すように、2 つの追加 LUT は「ブロック」ゲートを実装するために使用されます。CLB、SB、および CB 内のルーティング MUX は、図に示すように接続フレームワークを作成するようにプログラムされます。CMB チェーンの長さは 2,562 ビットで、そのうち 740 ビットは CLB 内の 10 個の LUT を構成するために、960 ビットは CLB 内のローカル ルーティング MUX を構成するために、32 ビットは I/O パッドの方向を定義するために、400 ビットは SB MUX を構成するために、430 ビットは CB MUX を構成するために使用されています。 2つのアラーム信号（ alarm1/2)はDEFCONモニターの出力を表し、 AFS 、 BFS 、 CFSは機能ユニットのフェイルセーフ出力を表します。アラーム信号は、キルスイッチまたは回復手順を実装するモジュールにルーティングできます。

図 3 に示す OpenFPGA アーキテクチャの構造ネットリスト表現で障害をシミュレートすることにより、DEFCON のフェイルセーフ特性を分析します。障害は、2 つの方法のいずれかでシミュレーション モデルに導入されます。

まず、冗長出力のペアに反対の値を割り当てることによって、機能ユニットの出力に障害が作成されます。たとえば、 {A00、 A01、 A10、 A11} = ''0011'' (複製されたワイヤ、たとえば{A00、 A01}には常に同じ値が割り当てられることに注意してください)。

出力ペアに同じ値が割り当てられる無故障割り当ては8つあり、ペアの出力の一方にのみ反対の値が割り当てられる故障割り当ては24つあります。つまり、機能ユニット内で単一故障モデルが想定されています。シミュレーションモデルのこれらの信号入力コンポーネントをFuncUnitと呼びます。

DEFCONモニター自体にも障害が導入されます。FuncUnitテストシナリオごとに、2,562個のCMBビットのうち1つを反転させることで、CMBチェーンに1つずつ同時に障害が導入されます。シミュレーションモデルのこれらのコンポーネントを指すために、 FPGAUnitという用語を使用します。したがって、障害シミュレーション実験は、 FuncUnitとFPGAUnitにそれぞれ1つの障害を導入する二重障害モデルに基づいて実行されます。シミュレーション結果には、この二重障害モデルにおける検出数、つまりアラーム信号のアサーション数が報告されます。

C. シミュレーション実験の課題

FPGA設計における故障シミュレーションの課題としてよく知られているのは、組み合わせループの発生です。これは、故障によって同じ組み合わせ論理ネットワークの出力と入力が接続される際に発生します。FPGAの配線アーキテクチャの柔軟性により、

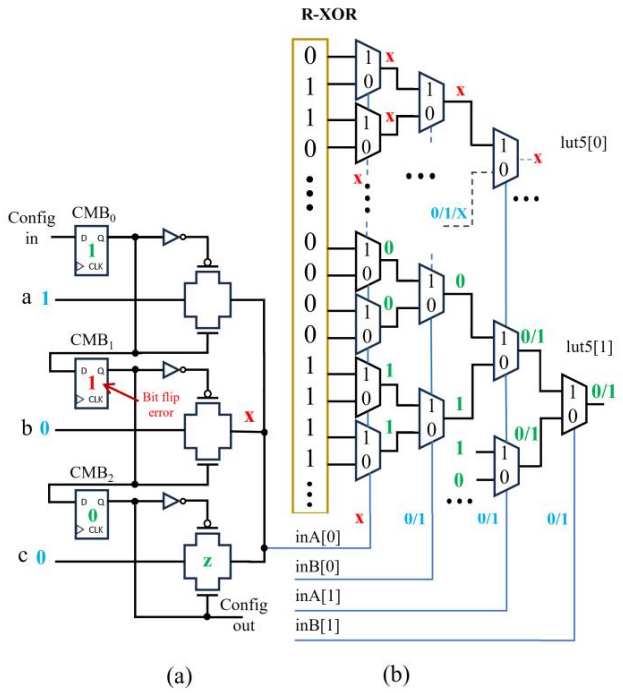

図7. (a) CMBビット反転により2つのトランスミッションゲートがMUX出力を駆動するワンホットMUX実装の例。入力aは1を、入力bは0を駆動する。シミュレータはMUX出力の論理値を判定できず、xを割り当てる。(b) LUT入力inA[0]がxによって駆動され、上部コーン出力(lut5[0])にxが出力されるテストシナリオ。DEFCON冗長化スキームは、入力inA[1]とinB[1]を用いて下部コーンの論理関数を評価し、出力(lut5[1])に明確に定義された0または1を生成する。

これは可能です。組み合わせループを生成する故障はシミュレータが結果を生成できないため、解析ではシミュレーション対象故障セットから除外されます。シミュレーション失敗の数は、次のセクションに示す結果表の別の列に報告されます。

ワンホット•パスゲートMUXの実装は、FPGAの故障シミュレーションにおける2つ目の課題です。FPGA内の配線構成は、CLB、CB、およびSB内のMUXを使用して実装されます。そのため、配線ネットワーク内の様々な構成オプションを実装するには、多数のMUXが必要になります。ワンホット•パスゲートMUXの実装は面積効率に優れていますが、故障発生時には望ましくない特性を示し、短絡や出力ノードのフローティング状態を引き起こす可能性があります。

図7の左側は、 MUXのパスゲート実装でのみ発生する状態を示しています。ここでは、CMB1セル内のビット反転により2番目のパスゲートが有効になり、入力aとbの間に短絡が発生します。ハードウェアでは、出力ノードの論理値は2つのパスゲートの抵抗比によって定義されます。例えば、電源電圧が1.0Vの場合、出力ノードの論理値は0.5Vになります。シミュレーションツールは、図に示すようにx （未定義）の論理値を割り当てることで、この短絡状態を処理します。2つ目の障害シナリオとして、CMBビット反転によってすべてのパスゲート（図示せず）が無効になり、出力がハードウェアでは、ノードが未知の値に浮動している可能性があります。この場合、シミュレータはz （高インピーダンス）の論理値を割り当てます。どちらのシナリオでも、パスゲートMUXは、フォールトトレラントおよびフェイルセーフシステム設計において、ゲート実装よりも多くの課題をもたらします。

幸いなことに、DEFCONで提案されている冗長化方式は、どちらのタイプの障害状態にも関わらず正常に動作します。図7の右側は、 inA[0] LUT入力がxで駆動されたときのLUT出力lut5[0]とlut5[1]の動作を示しています。上段のコーンの出力もxです。これは、初段のMUXへの選択入力が未知の論理値で駆動され、それらのMUXへのCMB入力が異なる値を持つためです。一方、下段のコーンは障害にもかかわらず正常に動作し、lut5[1]出力に明確に定義された「0」または「1」を生成します。これは、inputsinA[1]とinB[1]の値によって決定されます。

これは、第 1 ステージ (および第 2 ステージ) MUX 入力がすべて同じ入力値で駆動され、障害が発生した入力inA[0] の実際の値が無関係になるためです。

残念ながら、シミュレータはout5[1]の値が適切に定義されているかどうかを判断できず、 lut5[0]とlut5[1]の両方にxを保守的に割り当てます。シミュレータはxをネットリストの残りのコンポーネントに伝播させ、図4のalarm1とalarm2信号に多数のxを割り当てます。

この問題は、シミュレーション モデルにアサーションを追加することで対処します。アサーションは、DEFCON デザインの 6 つの LUT の入力値を監視し、図7に示す条件と同様の条件が満たされた場合に、 lut5[0] またはlut5[1] に 0 または 1 を再割り当てします。たとえば、 inA[1] とinB[1] が同じ論理値の場合は lut5[1] に 0 が再割り当てされ、それ以外の場合は 1 が再割り当てされるため、 xの割り当ては不要になります。これらのアサーションを追加することで、下流のアラーム信号へのxの割り当て数を大幅に削減できました。たとえば、 {A00/01、 A10/11} = ''0011'' で、 BxとCxに ''0000'' が割り当てられた FuncUnit シミュレーションでは、アサーションがない場合のalarm1へのxの割り当て数は 257 ですが、アサーションを含めると 35 に削減されます。

D. シミュレーション結果

前述の通り、シミュレーション実験は二重故障モデルを用いて実施されます。このモデルでは、 FuncUnitとFPGAUnitに同時に単一故障が導入されます。アラーム信号と機能ユニット出力の結果を、以下の2つのサブセクションでそれぞれ示します。

1) 警報信号分析警報信号分析は、

機能ユニット出力の障害が正常に検出された回数に焦点を当てています。表1には、32個のFuncUnitテストケースシナリオのうち4つのシナリオの結果が示されています。残りのテストケースシナリオの結果も、ここに示したものとほぼ同様です。

「Fault-Free」というラベルの付いたFuncUnitテストケースは、機能ユニット出力への一貫した割り当てを参照します。たとえば、 A00 / 01とA10/11は「0000」または「1111」に設定されますが、「Fault X」テストケースは、一貫性のない割り当てを参照します。たとえば、

表1. DEFCONの障害検出結果。

FuncUnit	At least one Alarm	$\mathbf{Alarm_1}$	$\mathbf{Alarm_2}$	Both alarms x	Missed	Loops	Correctable	Max
Fault-Free	77	69	65	12	-	0	6	2562
Fault A	2562	2511	2513	0	0	4	-	2562
Fault B	2562	2510	2512	0	0	4	-	2562
Fault C	2562	2502	2505	0	0	3	-	2562

表2.ブロックされた出力の結果。

FuncUnit	Alarms Asserted	Total Blocked	Not Blocked A	Not Blocked B	Not Blocked C	Total	Undefined x
Fault-Free	89	2534	8	7	13	28	903
Fault A	2562	2555	2	2	3	7	251
Fault B	2562	2555	2	2	3	7	223
Fault C	2562	2555	2	2	3	7	231

「0011」と「1100」です。各行には3つの機能出力を持つ8つの構成が可能ですが、ここでは1つの構成のみの結果を示します。例えば、 {A00/01、A10/11、 B00/01、 B10/11 、C00/01、 C10/11} = それぞれ「000000000000」、「001100000000」、「000000110000」、「000000000011」です。

表1の列の数字は、CMBの各ビットを1つずつ反転させた、2,562回のCMB障害挿入実験全体の結果を表しています。列2は、少なくとも1つのアラームがアサートされた回数を示し、列3と列4は、それぞれアラーム1とアラーム2がアサートされた回数を示しています。

列5は、両方の警報信号が未定義（x)である障害の数をカウントします。列6 （Missed)は、両方の警報信号を無効化する障害の数をカウントします。これには、列5で未定義としてカウントされた障害も含まれます。

7列目は、組み合わせループが原因でシミュレーションが失敗したテストケースの数を表しています。9列目は、図6のCorrectsw信号をアサートすることで修正可能な障害の数を示しています。

Fault-Freeの結果によると、アラーム信号のいずれかまたは両方がアサートされる障害はわずか77件でした。これらのアラーム状態はFuncUnitの障害ではなく、 DEFCONモニター自体の障害に関連しているため、障害アサートの数が少ないことは望ましい特性です。

これらの検出可能な障害のうち、 Correctsw はR-XOR ゲート内で発生する 6 つの障害を即座に修復できます。

3 行目から 5 行目に示されている障害A/B/Cテストの結果は、DEFCON モニターの有効性を反映しています。Missed列は、DEFCON モニターで 2 番目の障害が発生しているにもかかわらず、DEFCON モニターがFuncUnit障害を検出できなかったテスト ケースを示しています。ここで、「失敗」はどちらのアラーム信号もアサートされていないと定義されます。DEFCON 冗長化スキームは、すべてのテスト シナリオで 2,562 個の CMB 障害をすべて検出することができ、どちらの CMB 障害によっても両方のアラームが「x」に設定されることはありませんでした。さらに、各アラーム信号がアサートされる回数も多く、多くの障害によって両方のアラーム信号がアサートされていることを示しています。少数のテスト ケースでは組み合わせループが発生し、結果は不明ですが、DEFCON は二重障害モデルの下で機能ユニット障害をフラグ付けするという目的を達成しています。

興味深いことに、図 7 を参照して説明した解決にもかかわらず、Fault-Free テスト シナリオで両方のアラームが未定義となる回数は 12 回です。12ケースすべての根本原因は、OR ゲートの支配的な値に関連しています。

ほぼすべてのフォールトフリー テスト ケースでは、3 つの OR ゲートへの入力は 0 とx です。論理 0 は OR ゲートの非優位値であるため、すべての入力が 0 とx の場合、OR ゲートの出力はx になります。NR-OR ゲートも出力が 1 つしかなく、どちらかの NR-OR 出力にx が発生すると、両方のアラームにxが伝搬されるため、R-OR ゲートほど堅牢ではありません。これを防ぐために、R-OR の入力を再配線する (つまり、各 NR-OR ゲートの両方の冗長入力を R-OR 内の同じ冗長 OR ゲートに接続する) ことは可能ですが、これを行うと、一部の障害で組み合わせループが発生します。さらに、機能ユニットで障害が発生した場合、OR の優位値は 1 であるため、両方のアラームのxは少なくとも 1 つのアラームで直ちに 1 に解決されるため、これらのケースは多少意味がありません。

2) 機能ユニット出力信号の分析このセクションでは、前述の単一障害と二重障害の両方のシナリオで機能ユニット出力信号をブロックする際の DEFCON の有効性を分析します。

フェイルセーフ出力値として「0」を使用し、4つの異なるテストシナリオを用いてOpenFPGA設計をシミュレートしました。「Fault-Free」 FuncUnit テストケースでは、機能ユニット出力への一貫した割り当てを参照しますが、この解析ではすべて「1」を使用します。つまり、 {A00/01、 A10/11、 B00/01 、B10/11、 C00/01、 C10/11} = 「111111111111」です。「Fault X」テストケースでは、それぞれ「001111111111」、「111100111111」、「111111110011」という不整合な割り当てを参照します。

表 2 の列に示されているデータは、左から右の順に、 FuncUnitテスト シナリオ、一方または両方のアラームがアサートされた回数、機能出力(AFS 、 BFS 、 CFS) が正常にブロックされた障害シミュレーション ケースの数、DEFCON が機能ユニット出力のブロックに失敗した回数、テスト ケースの中で失敗したブロックの合計数、および出力が未定義であるテスト ケースの数を示しています。

フォールトフリーの結果は、失敗したブロックと不確定な状態が最も多く表示されますが、

機能ユニット出力に障害がないため、これらのケースは無害です。ここでは、どちらのアラームもアサートされていないが、1つ以上の機能ユニット出力にエラーがあるケースを観察します。2行目から4行目、および4列目から6列目に示されている結果は、DEFCON が失敗するケースを表しています。ただし、DEFCON が失敗する二重障害テストケースの数は2と3と少ないです。ここでは、1つまたは両方のアラームがアサートされていますが、DEFCON は1つ以上の機能ユニット出力信号をブロックできません。残念ながら、これらの行では不確定値の数がかなり多いため、スキームの全体的な有効性を評価することは不可能です。図示されていませんが、機能ユニット出力のみに障害がある場合、DEFCON はすべての機能ユニット出力をブロックすることに成功しています。

E. 資源利用

冗長（R)LUTと非冗長（NR)LUTのインスタンス化を組み合わせることで、DEFCONはコンパクトなアーキテクチャを実現しています。監視対象となる機能ユニットの出力ペアごとに1つのLUTが必要です。NR-ORゲートは3対2の圧縮率を提供し、最大3つのR-XORゲートからの入力を受け取り、2組の出力信号を生成します。

R-ORゲートは上流のアラーム信号の圧縮にも使用できますが、前述の通り、圧縮率は2.5対2とやや低くなるため、NR-ORゲートの代わりに使用する場合はより大きなネットワークが必要になります。ただし、R-ORゲートはデュアル出力であるため、 FPGAユニットの障害に対する耐性はより高くなります。

DEFCONモニタのLUT使用率はnlognで上限が決まります。例えば、機能ユニットの出力数を3から6に倍増すると、X-ORゲートとNR-ORゲートの数も倍増し、R-ORゲートの数は1から3に3倍になります。ここでは、R-ORゲートが最終圧縮段で使用されると仮定し、LUT使用率は6から13に増加します。アラーム信号圧縮ネットワークの全段でNR-ORを使用することで、リソース使用率をさらに削減できます。

IV. DEFCONのFPGA実験評価

このセクションでは、ザイリンクスZynq 7010 FPGAにDEFCONを実装し、その有効性を解析する。DEFCONモニターの機能ユニットとして、Advanced Encryption Standard（AES)アルゴリズムを採用した。AESアルゴリズムのオープンソースVerilog HDL記述は[18]から取得し、ザイリンクスVivadoを用いて合成した。AESアルゴリズムを複製し、DEFCONを組み込んだ設計を構築した。AESエンジンは128ビットキーで暗号化を行うように構成されている。

実験設計のブロック図を図8 に示します。プロセッサ側 (PS) には ARM Cortex A9 が含まれており、これは Linux オペレーティング システムでフォールト インジェクション マネージャ (FIM) として機能する C プログラムを実行します。

DEFCON を備えた複製された AES エンジンの障害のないバージョンは、SoC のプログラマブル ロジック (PL) 側の動的部分再構成 (DPR) 領域への実行時プログラミング用の部分ビット ストリームに合成されます。

部分的なビットストリームはVivadoを使用して作成され、転送されます。

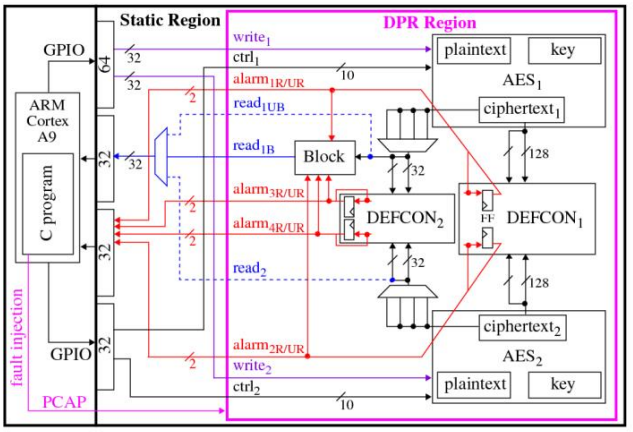

図 8. Xilinx Zynq 7010 SoC デバイスのプロセッサ、スタティック、および DPR プログラマブル ロジック領域のブロック図。C プログラムは PCAP インターフェイスを介して DPR 領域を再プログラムし、各実験のプログラミング ビットストリームに 1 ビットの反転を導入します。AES エンジンの 2 つのコピーが、2 つの DEFCON モニターとともに DPR 領域に配置および配線されます。DEFCON1 は、暗号化の各ラウンドで更新される 2 つの 128 ビット内部データパス レジスタを比較し、DEFCON2 は32 ビットの読み出し回路を監視します。ブロッキング回路 (Block) は、 4 つの登録済みアラーム信号 (alarmxR) のいずれかがアサートされると、 read1B バスに 0 を強制します。スタティック デザインは、4 つの GPIO レジスタを使用して、プロセッサから DPR 領域への制御信号とデータ信号を配線します。

Avnet ZYBO Z7-10ボード上のフラッシュメモリカードに[19]、 [20]。

図8 に示すように、2 つの AES 実装で使用される DPR 領域リソースは完全に独立しており、個別の制御信号(ctrl1とctrl2)、 32 ビット データ入力(write1とwrite2) 、およびデータ出力(read1Bとread2)バスを備えています。

同様に、AES の各コピーには、個別のキー、プレーン テキスト、および暗号文のラウンド レジスタがあります。制御信号とデータ入力バスは、Vivado がこれらの信号に対して配線削減の最適化を実行しないように、別々の GPIO チャネルに接続されています。データ出力バスは、暗号化後にread1B、 read2 、およびread1UB暗号文出力を読み取れるように MUX で構成されています。ここで、添え字のBとUB は、それぞれブロックと非ブロックを表します。read2とread1UBは、挿入された障害の影響を判断するのに役立つ情報としてのみ使用され、DEFCON のフィールド バージョンには含まれません。同様に、4 つの登録済みアラーム信号(alarmxR)と 4 つの未登録アラーム信号(alarmxUR)は、通常は 2 つの登録済みアラーム信号に削減されますが、障害動作の分類に役立つように別々に配線されています。

図9は、この設計のVivado実装ビューを示しています。マゼンタ色の四角形で囲まれた静的領域（左)とpblock領域（右)の両方が含まれています。Xilinx Vivadoでは、pblock構造を使用して領域を再構成可能として指定します。AES1 、 AES2 、 DEFCONという注釈は、これらの要素が配置配線される一般的な領域のみを示しています。実際には、これらのコンポーネントは互いに明確な境界なく絡み合っています。静的領域から動的領域への配線、およびその逆の配線はすべて1回のみ実行するように制約されています。

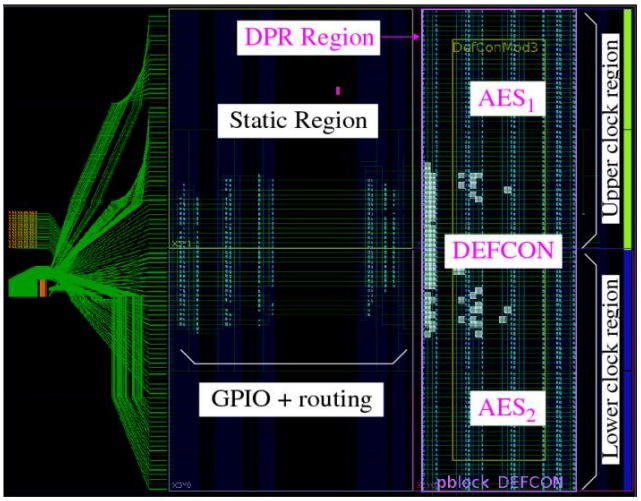

図9. Xilinx Zynq 7010の実験設計。AESエンジンとDEFCONモニターの2つのコピーが、右側のマゼンタ色の四角で囲まれたダイナミック・パーシャル・リコンフィギュレーション（DPR）領域に配置されています。左側の静的設計には、プロセッサ上で実行されるCプログラムがプログラマブルロジックに制御とデータを送るためのGPIOレジスタと、これらの信号のファンアウトを実装するためのMUXのみが含まれています。これらの信号は、別々のインターフェースを介してDPR領域に配線されます。

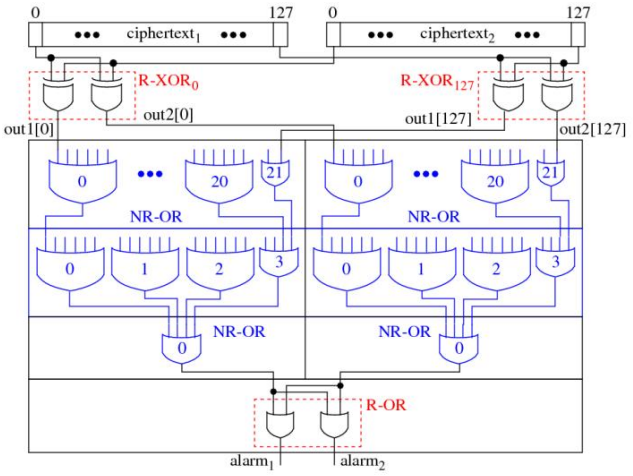

図10. 2つのAESエンジンからの2つの128ビット暗号文ラウンドレジスタを監視するDEFCON1の回路図。使用されるリソースには、128個のR-XOR LUT、54個のNR-OR LUT、および1個のR-OR LUTが含まれます。alarm1およびalarm2信号は、10ラウンドのいずれかで不一致が発生した場合に「1」を格納するレジスタ（図示せず）に接続されます。

制御信号やデータ信号はDPR領域に配線されておらず、その後再び出力されません。これは、これらの信号がデバイス上のI/Oパッドを介して配線されるモデルを表しています。これは、これらの信号がGPIOを介してSoCのプロセッサ側に配線される私たちの設定とは対照的です。設計のサイズにより、デバイスの右側にある上位クロック領域と下位クロック領域の両方にまたがるDPR領域が必要です。各領域には1,554,592ビットのプログラミングデータがあり、3,109,184ビットの障害エミュレーション実験（FEE）が生成されます。私たちのFIMは1秒あたり約6.6回の障害エミュレーション実験を実行でき、2つのクロック領域それぞれですべての障害エミュレーション実験を完了するには約2.7日かかります。

ハードウェア設計に 2 つの DEFCON チェッカー回路が追加され、図 8 では DEFCON1とDEFCON2とラベル付けされています。
DEFCON1 は2 つの AES エンジンからの 128 ビット暗号文ラウンドレジスタのエラーチェックを行い、 DEFCON2 は2 つの 32 ビット読み出しバスのエラーチェックを行います。DEFCON1 のアラームは、 10回の暗号化ラウンド中に発生した比較ミスをアラーム状態として記録し、暗号化操作の終了時に読み出して保存できるように登録されます。同様に、暗号文の読み出し操作は 32 ビットの読み出しを 4 サイクル必要とするため、 DEFCON2もアラーム信号を登録します。登録されたアラーム信号(alarmxR)と、未登録のアラームワイヤの終了状態(alarmxUR)は、暗号化と読み出し操作が完了した後、または障害により完了できなかった場合はタイムアウト後に、C プログラムによって読み取られます。 DEFCONが現場に導入された場合、アラームがトリガーされると、暗号化および／または読み出し処理が直ちに停止し、出力がフェイルセーフ状態に強制され、その後、復旧のための緩和処理が開始されます。当社のテストおよび評価では、暗号化と読み出しは、システムの状態とは無関係に完了するようにしています。

アラーム信号を受信し、 read1Bチャネルを使用して暗号文1を読み出し、read2チャネルを使用して暗号文2とブロック解除された暗号文1をread1UBチャネルを使用してそれぞれ読み出します。暗号文の3つのコピーはすべて、8つのアラーム信号とともにファイルに保存され、オフライン解析に使用されます。

A. DEFCONモニター設計

DEFCONモニターは、OpenFPGA実験の図4に示したものと同じ方法で構築されていますが、監視対象ビット数がDEFCON1では128ビット、DEFCON2では32ビットに増加しています。また、NR-ORツリー回路の深さは、 DEFCON1では4レベル、DEFCON2では3レベルに増加しています。DEFCON1の回路図を図10に示します。実装に必要なリソースは、R-XOR LUT 128個、NR-OR LUT 54個、およびR-OR LUT 1個です。

DEFCON2には、R-XOR LUTが32個、NR-OR LUTが14個、R-OR LUTが1個必要です。両方のDEFCONモニターは合計230個のLUTを使用します。

ブロック回路は、図11に示すように冗長な非アクティブ化信号で構成されています。2つの NOR ゲート出力は 32 個の AND ゲート LUT のセットにルーティングされ、 read1UB 32 ビット バスから出力信号read1B への通過をゲートします。
登録されたアラーム信号のいずれかがアサートされると、 read1B出力は強制的に 0 になります (フェイルセーフ出力値として 0 を使用しますが、任意の割り当てが可能です)。

B. テストプロセス

FIM によってソフトウェアで実行されるアクションは、図12のブロック図に示されています。以下に、障害エミュレーション プロセスをまとめます。

- FIMは、障害のない部分ビットストリームを
  メモリアレイ。

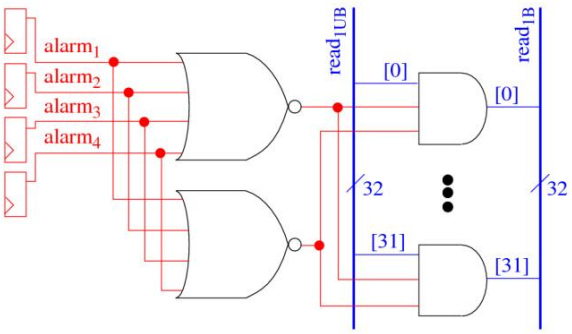

図11.登録されたアラームのいずれかがアサートされると、暗号文1の32ビットチャンクを強制的に0にするブロック回路の回路図。ブロック信号は複製され、障害発生時の非アクティブ化に対する耐性を高めます。リソースは2つのNOR LUTと32個のAND LUTで構成されています。

- DPR 領域は障害のない部分ビットストリームでプログラムされ、GPIO 制御信号を使用してキーとプレーンテキストがロードされ、AES エンジンが 128 ビット暗号化用に構成され、両方の AES エンジンが同時に起動されて障害のない暗号化操作が実行されます。

  エラーのない暗号文が取得され、後続の FEE で使用するためにメモリに保存されます。
- 障害エミュレーションループが実行され、Type 2パケットのベースアドレスから部分ビットストリームに1ビット反転エラーが導入されます[21]。 • 障害のある部分ビットストリームは、FIM Cプログラムからのシステムコールを使用してPCAPインターフェースに書き込まれます。 • GPIO制御レジスタは、キーとプレーンテキストをロードし、AESエンジンを128ビット暗号化用に設定し、AESエンジンをロックステップで起動するために再び使用されます。 • AES完了信号は、完了のために監視されます。

  アサートされると、データのアンロード手順が開始されます。
  1 ミリ秒のタイムアウトにより、障害によって AES エンジンが done 信号をアサートできない場合でも処理を続行できます。 • データのアンロードは、暗号文レジスタのどの 32 ビット チャックをread1UBおよびread2データ バスに転送するかを示すアドレス信号と開始信号を AES エンジンに発行することから構成されます。暗号文は、FIM によって GPIO データ レジスタから取得され、ファイルに保存されます。前述のように、データ アンロード プロセスでは、障害の発生した回路の動作を判断するために役立つ 3 つの異なるソースから暗号文を取得します。 • 8つのアラーム ビット値も読み取られ、ファイルに保存されます。correctsw が '0' に設定されたアラーム ステータス値が最初に読み取られ、次にcorrectsw制御信号がアサートされた後に再度読み取られます。alarmxURビットが '0' に戻ると、障害は修復可能に分類されます。
- 部分ビットストリームのタイプ2データパケット領域に挿入可能な各障害に対して、障害エミュレーションループが繰り返されます。このプロセスは、DPR領域の上位クロック領域と下位クロック領域の両方に対して繰り返されます。

C. 実験結果

このセクションでは、4つのザイリンクスZynq 7010 SoCから収集したデータを分析します。主な目的は、良性の故障数、アラームによって検出された故障数、および見逃された故障数を特定することです。OpenFPGA解析と図7に示した解析から、スイッチボックスで使用されているトランスミッションゲートMUX構造に導入された故障が、短絡および開放状態を引き起こす可能性があります。Zynqデバイスレイアウトの実装詳細は入手できませんが、ハードウェアで同様の状態が発生する可能性があると予想されます。デバイス間で結果を比較することでこれを確認し、差異はダイ内のプロセスばらつきに起因すると考えています。このばらつきは、 [22]で示された結論と同様に、短絡および開放故障の動作をデバイス依存にする役割を果たします。

まず、SoCのプロセッサシステム（PS）側とプログラマブルロジック（PL）側を用いてフェイルセーフシステムを実装する実験セットアップにおける重要な制限事項について説明します。最初の制限事項は、DEFCONがチェッカー自体の後段で情報を伝達する回路コンポーネントの故障を検出できないことです。例えば、図8に示すように、32ビットのread1UBバスがブロック回路を通過し、 read1BがDPR領域からスタティック領域に渡る地点までの配線で発生する故障は検出されません。

冗長アラーム信号とブロック回路自体内の冗長実装により、ブロック回路のいくつかの種類の障害に対する耐性が向上しますが、AES エンジンの両方のインスタンスで暗号文が正しい場合でも、静的領域への送信時にデータが破損しないことを保証できません。

この問題を解決する直感的な方法としては、 DEFCON2をブロック回路の出力に接続することが考えられます。しかし、これを行うには、破損データが設計の静的部分（または実際のフィールド版DEFCONのI/Oパッド）に到達するのを防ぐため、より複雑なパイプライン構造が必要になります。例えば、 read1Bデータバスで配線障害が発生した場合、 DEFCON2はクロックの次の立ち上がりエッジでalarm3Rをアサートし、ブロック回路はread1Bデータバスにフェイルセーフ値を生成するように強制します。しかし、静的領域へのインターフェースの下流に配置されたread1Bデータバスのパイプラインレジスタは、破損したデータをキャプチャしたばかりなので、リセットする必要があります。パイプラインレジスタに緊急非同期リセット信号を搭載することで、レジスタをフェイルセーフ値に強制的に設定できます。しかし、この戦略は、パイプラインレジスタへの配線およびそれ以降の配線で発生する可能性のある障害に対してフェイルセーフではありません。 2つ目の戦略としては、パイプラインレジスタを省くために、レジスタ未登録のalarm3UR信号をブロック回路への入力として利用するという方法がありますが、この設計では組み合わせループが発生し、発振の可能性が生じます。最善の解決策は、ブロッキング回路の配置と配線を静的領域（またはI/Oパッド）へのインターフェースに直接固定し、下流への配線を排除することですが、そのためには特別な制約を作成する必要があります。

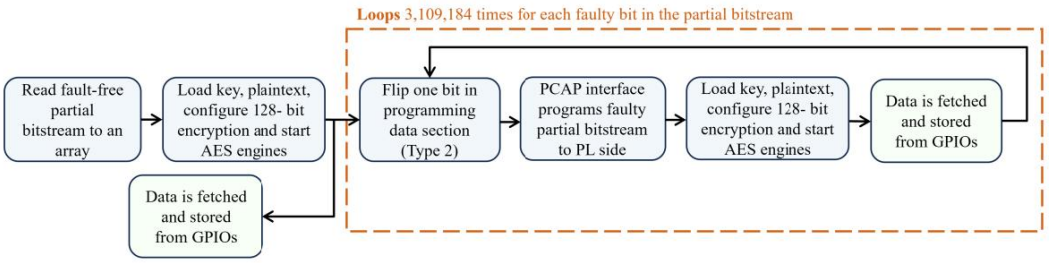

図 12.フォールト インジェクション マネージャ (FIM) によって実行されるアクションを示すソフトウェア フロー ダイアグラム。

表 3.最初の Zynq 7010 デバイスの実験結果 (合計 3,109,184 個の障害)。

Status	Top Alarms	Bottom Alarms	Both Regions	Correctable
Benign	1,412,356	1,399,009	2,811,365	-
Active	142,236	155,572	297,808	-
At least one Alarm	141,925	153,986	295,911	324
False Positives	222	475	697	-
Missed	311	1,586	1,897	-
Untestable	0	11	11	-

ファイル。今後の作業では、このタイプのカスタマイズされたソリューションによって実現可能な改善点を調査します。

単一障害モデルでは、AESエンジンの1つに不具合を引き起こす障害は、常にアラーム1とアラーム2の両方をアサートします。ただし、まれにAESエンジン内の配線とアラームの1つが短絡する場合を除きます。その場合、アラームの1つが無効になる可能性があります。いずれの場合も、FIMは障害の検出を記録します。したがって、AESエンジン内で発生したすべての単一障害が検出されます。

これは、単一の障害によって最大2本の配線が短絡する可能性があると想定しており、その場合、2本の配線が両方のAESエンジンから出ている同一の配線ではないことも想定しています。同一の配線が2本、同じスイッチボックスを通過する可能性はありますが、極めて低いと考えられます。

後ほど示すように、私たちの設計ではそのような事例は発生しませんでした。

しかし、どちらのAESエンジンにもエラーが発生していないにもかかわらず、1つ以上のアラームがアサートされるケースがいくつかあります。以下、これを誤検知と呼びます。OpenFPGAの解析によると、最も一般的な誤検知シナリオは、監視配線の配線、またはDEFCON LUTのCMBの異常によって、障害がDEFCONモニター自体に影響を与える場合に発生します。correctswを使用すれば、LUT CMBで発生した障害を即座に修復できますが、配線障害の場合はスクラビング操作、つまりDPR領域の再プログラミングが必要になります。

別のシナリオでは、障害によってアラームの1つがVCCに短絡する可能性を考慮しています。後者のシナリオは、登録済みアラームと未登録アラームの4つのうち1つだけがアサートされるため、この設定では区別できます。他のシナリオも考えられますが、結果を示す表を参照して説明します。

FIMは最初のテストとしてエラーのないエミュレーションを実行し、正しい暗号文を配列に保存します。エラーのない暗号文は、 read1B (ブロック回路が直列に接続された暗号文チャネル)から読み取られた暗号文と比較されます。

不一致が発生し、8 つのalarmxフラグのいずれも設定されていない場合、障害はミスとして分類されます。私たちの実験では、観測されたミスはすべて、DEFCON2 回路の後のルーティングまたは LUT に影響する障害で発生しています。read1UB および read2チャネルから読み取られた暗号文が正しい暗号文であることを確認し、AES エンジンの障害を除外することでこれを検証しました。さらに、すべてのケースで、 read1Bチャネルから読み取られた暗号文には、正しい暗号文の 32 ビット チャンクが表示されます。対照的に、どちらかの AES エンジンが障害の影響を受けている場合は、AES ラウンド変換によって、アバランシェ効果により、正しい暗号文の一致する 32 ビット チャンクがすべて削除されます。DEFCON のフィールド化バージョンでは正しい暗号文がわからないため、これらの比較操作は、この研究では診断目的でのみ実行されることに注意してください。

結果の考察のため、8 つのアラーム信号を 4 ビットずつ 2 つのグループに分け、最初のグループをアラーム 1 と 2 に、2 番目のグループをアラーム 3 と 4 にそれぞれ対応させます。アラーム信号に対応するビット値は 0 から F までの 16 進数値としてコード化され、各 16 進数の上位 2 ビットは未登録アラーム信号に対応し、下位 2 ビットは登録済みアラーム信号に対応します。たとえば、アラーム ステータス コード 3 は(alarm1UR、alarm2UR、 alarm1R、 alarm2R) = 0011 とデコードされます。これは、登録済みアラーム信号は障害を検出したが、未登録アラーム ワイヤはアサートされていないため障害を検出できなかったことを示します。実際の DEFCON フィールド システムでは、登録済みアラームのみが使用されることに注意してください。未登録アラームは、この作業では診断情報としてのみ使用されます。

表3の左端の列には登録された警報信号の状態カテゴリーがリストされており、2列目と3列目には

表4.ブロッキング結果

Total Alarms	Full Block	Partial Block	Missed	False Positive
297,808	266,020	3,541	1,897	25,266

DPR 地域の上位半分と下位半分の各ステータス カテゴリに分類される FEE の数。

列 4 には、列 2 と 3 の行ごとの合計が表示されます。列 5 には、少なくとも 1 つの登録済みアラーム信号をアサートするセットから修正可能な障害のみがカウントされます。

まず、上半分と下半分にはそれぞれ1,554,592 CMBがあるため、実行された故障エミュレーション実験の総数は3,109,184回であることがわかります。Vivadoのレポートによると、AESインスタンスとDEFCON回路はLUTリソースの76.1%とFFリソースの34.0%を使用しています。最も重要なステータス条件は表の行に示されており、以下のように説明されています。

•良性: これらの障害は、正しい暗号文を持ち、アラームが設定されていないものとして特徴付けられます。設計で使用されるリソースの使用率がかなり高いにもかかわらず、ほとんどの CMB は設計を実装するために使用されていないため、これらの CMB に挿入された障害は機能動作に影響を与えません。これらの結果は、平均して特定の設計で CMB の 10% のみが使用されていることを明らかにした以前の研究と一致しています[23]。 •アクティブ: これらの障害は、障害が AES エンジンの 1 つに影響するか、読み取り回路が破損しているために、暗号文の 1 つが破損しているものとして特徴付けられます。アクティブな障害に対応する CMB は、以前の研究 (および Xilinx による) では必須コンフィギュレーション ビットと呼ばれています[22]。 •少なくとも 1 つのアラーム: これらの障害は、DEFCON モニターの 1 つまたは両方によって検出されるものとして特徴付けられます。 •誤検知: これらのDEFCONモニター障害は、暗号文に影響を与えないものの、1つ以上のアラーム信号を発生させるものです。これらの障害は「少なくとも1つのアラーム」クラスにもカウントされることに注意してください。 •見逃し: これらの障害は、 read1B暗号文を破損させるものの、アラーム信号は発生しないものとして特徴付けられます。

•テスト不可能: これらの障害は、クロック ネットワークに影響を与え、FPGA をロックし、暗号文とアラーム結果を取得できなくなるという特徴があります。

以下の結論は、表3の表の結果と暗号文および警報信号の全セットの観察に基づいて導き出されます。

•障害検出能力: DEFCON 技術は、上部領域の活性障害の 99.78%、下部領域の活性障害の 98.98% を検出できます。

•ミスフォールト: 3つのチャネルすべてからの暗号文を使用して、すべてのミスケースはDEFCON2の下流のルーティングとLUTの障害によって発生したことを確認しました。つまり、AESエンジンは正しい計算を行いました。

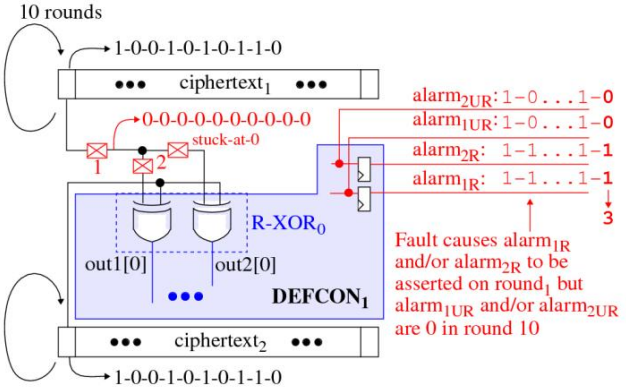

図 13.オープン障害モデルと、登録済みおよび未登録のアラーム信号への影響。

暗号文は生成されましたが、 read1Bチャネルを通じて正しく送信されませんでした。

•訂正可能な障害: FEEで観測された訂正可能な障害の数は324で、これは2つのDEFCONモニターで使用されるR-XORゲートとR-ORゲートの数と一致しています。つまり、2*128 + 2*32 = 320 R-XORと2*2 R-NORです（前述のように、各冗長LUTゲートでは2つの障害が訂正可能です）。 •登録されたアラーム信号のみをアサートする障害（アラームコード = 3） :これらの障害は、暗号文をDEFCONチェッカーにルーティングする際に発生します。このアラーム状態を引き起こす可能性のある障害の例は次のとおりです。

図13に示します。この図では、赤い四角でスタック•アット•ゼロ障害のセットが注釈されています。スタック•アット•ゼロ障害は、CMB障害によって開回路が作成され、おそらく弱いプルダウンによって浮動ネットが論理「0」に強制されるときに発生します。ここでは、両方のAESエンジンが暗号化操作の10ラウンドを実行するときに生成される可能性のあるビット値のシーケンスの例を示しており、「1001010110」と表記されています。この例では、最初のラウンドの上位（左端）ビット位置に「1」があります。スタック•アット•ゼロ障害により、R-XORは最上位の暗号文ラウンドレジスタから「1」を受信できなくなり、どのスタック•アット•ゼロ障害がアクティブになっているかに応じて、 out1[0]またはout2[0]のいずれかにアラームがアサートされます。これにより、 R-ORコレクタゲートへの入力のファンアウトにより、レジスタに記録されたアラーム信号alarmxRの両方に「1」が記録されます（図10参照）。暗号化エンジンの処理が完了すると、レジスタ内の暗号文の上位ビットは「0」になります。

これにより、未登録のalarmxUR信号は「0」に戻ります。これは、障害値と実際の値が一致するためです。この場合のアラームコードは「0011」となり、未登録アラームが両方ともデアサートされ、登録済みアラーム信号が両方ともアサートされていることを示します。 •登録済みアラームと未登録アラームの両方がアサートされる障害（アラームコード = F） :これは最も一般的なエラーコードであり、AES実装の1つにおいて、障害がルーティングまたはLUTネットワークに影響を与えた場合に発生します。

欠陥により、1ラウンド以上で暗号文が破損する

AES 暗号化操作が実行され、両方の DEFCON 回路に登録済みアラームと未登録アラームの両方が設定されます。

- R-OR真理値表の障害（アラームコード = 5またはA） :登録済みアラーム信号と未登録アラーム信号のいずれか一方のみがアサートされます。このケースは、障害が次のいずれかに影響した場合に発生します。
  R-OR LUT内の選択された「0」ビット。ここで観察されるのは
  暗号文は 3 つの読み出しチャネルすべてで正しいです。
  この状態は、設計内にR-ORゲートが2つ（DEFCONモニターごとに1つ）しか存在しないため、正確に4回発生します。図14に、この障害の影響を示します。CMB内の1つの障害（赤で強調表示）は、アラーム信号の1つにのみ影響を及ぼします（LUTの上部MUXネットワークとアラーム信号は示されていません）。

  この場合、青色で強調表示されているcorrectsw信号をアサートすると、LUT の冗長真理値表エンコーディングの上位 16 ビットに切り替えることで、障害を即座に修復できます。

- 非決定性 :FPGAおよびトップまたはボトム領域において、20件未満の少数の障害が発生し、すべての読み出しチャネルで常に誤った暗号文が生成されました。そのため、アラームはいずれもアサートされませんでした。さらに、この障害が発生したすべてのFEEで、暗号文は全く同じでした。

  また、この異常な状態を引き起こすFEEは非決定論的であることも判明しました。つまり、FEEを2回再実行すると、3つのチャネルすべてで暗号文にエラーがなくなります。この異常な動作の根本原因は依然として不明です。私たちの推測では、SDカードから部分的なビットストリームを読み取り、PS側からPCAPインターフェースにストリーミングするシステムコマンドが時折誤動作し、PL側を何らかの障害状態に陥らせることが原因と考えられます。これらのFEEを再実行するプロセスにより、すべてのインスタンスでエラーが解消されたため、表3ではこれらのエラーを「良性」に分類しています。今後の研究では、根本原因をさらに調査する予定です。

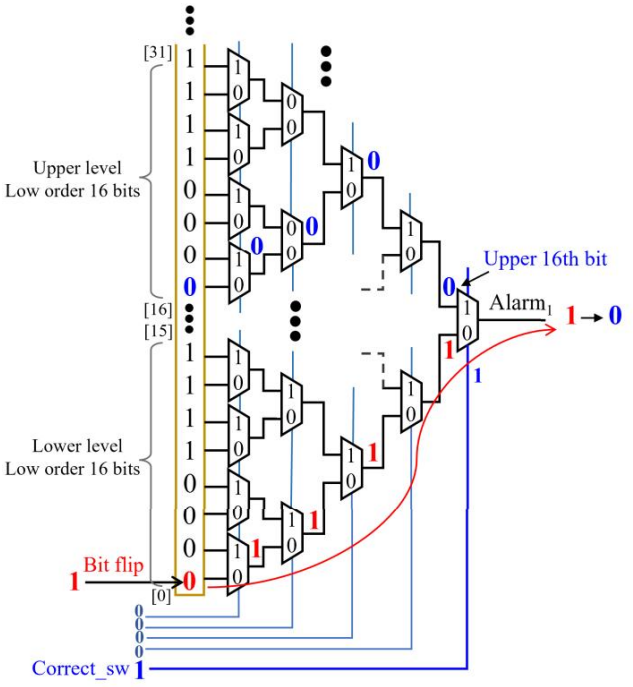

図14. Correct_sw信号を使用した修復機能を示すDEFCON R-ORゲート。

ブロック回路の結果は、表4 に示すように、完全ブロック、部分ブロック、ミス、および誤検知の 4 つのカテゴリに分割されます。アラームのいずれかがアサートされると、ブロック回路は暗号文をブロックし、すべてのバイトをゼロにします。ミス列で示されるブロック障害の数は、表3 のミス行で示されるミスしたアラームアサーションの数と正確に対応します。前述のように、これらのミスは、ルーティングのブロックコンポーネントの後、および DEFCON2 の下流の LUT で発生する障害に起因します。1つ以上のアラームがアサートされ、障害が確実に検出されたにもかかわらず、暗号文の一部のみがブロックされているケースも確認されています。これは、障害が読み出し MUX 構造で発生した場合に発生し、読み出し操作中に MUX への障害のある入力コンポーネントにアクセスされるまで、 DEFCON2 によって障害が検出されません。

表4の「偽陽性」列に挙げられている障害は、暗号文が正しいにもかかわらず、ブロッキング回路が読み取りを妨げていることを示しています。これらのケースは、

DEFCON1またはDEFCON2コンポーネントの配線と LUT で障害が発生する実験。

1) FPGA依存の障害動作表3に示すFPGAの1つに関する結果は、他の3つのFPGAから収集したデータを用いた結果とは異なります。すべてのFPGAで同じ設計が使用されているため、観察された差異は配線ネットワークに短絡を引き起こす障害によるものと考えられます。短絡ノードを駆動するLUTとバッファの相対的な駆動力はプロセスばらつきの影響によって変動し、下流のLUTによって解釈される論理値に非決定性が生じます。

例として、図15は2つのLUT（ LUT1とLUT2)が、それぞれ反対の論理値を持つ2つのネットを駆動している様子を示しています。このケースの障害により、スイッチボックス内の2つの出力ネット間のNチャネルスイッチが有効化されます。配線とスイッチボックスMUXは分圧回路を構成し、下流のLUT（ LUT3とLUT4)の入力に現れる電圧値は、それぞれ論理「0」と論理「1」の電圧値よりも大きく、小さくなります。この例では下流の電圧値として0.4Vと0.6Vを示していますが、実際の値はR1、 R2 、 R3の相対抵抗値に依存します。

レシーバLUTは、誤った入力電圧を論理値として解釈する役割を担っており、正しい出力値を生成する場合とそうでない場合があります。また、LUT内のパスゲートMUXネットワークによって追加の短絡が発生し、 LUT3とLUT4の出力電圧が下流に不確定な論理レベルを伝播する可能性もあります。

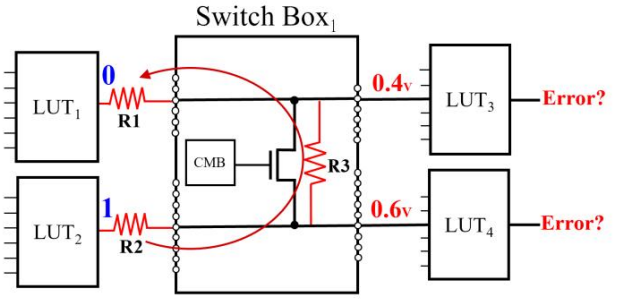

図15. FE実験の故障モデル。この故障は、2つの独立したネット間の接続を有効にすることで、2つのドライバLUT（LUT1とLUT2）間に短絡を発生させます。この例では、故障によりNチャネルトランジスタのゲートを駆動するCMBが0から1に反転します。

表 5.参照 FPGA と比較した 3 つの追加 FPGA の障害動作で観測された差異の数。

Device	Alarm	Detection	Ciphertext	Same	Total
$FPGA_1$	24,227	21,762	22,369	285,497	309,854
$FPGA_2$	20,934	18,773	19,240	286,983	308,040
$FPGA_3$	25,652	22,801	23,411	285,230	311,039

実験に使用した市販のFPGAにおいてこれらの条件が存在するかどうかは検証できませんが、4つのFPGA間で観測された障害挙動の差異から、FEEのサブセットに何らかの競合が存在することが確認できます。このような事例の数は表5に示されています。表5では、最初の列にリストされている3つの追加FPGAデバイスそれぞれについて、アラーム値と暗号文の差異を、上記の基準FPGAの値と比較しています。ここでは、上位領域と下位領域の両方で観測された差異の合計を報告しています。2番目の列「アラーム」は、2つのFPGAに登録されている4つのアラーム値に何らかの差異が存在する障害の数を示しています。「検出」列は、2番目の列のうち、一方のFPGAが障害を検出し（少なくとも1つのアラームがアサートされている）、もう一方のFPGAが検出しない（アラームがアサートされていない）障害のみをカウントします。したがって、これらの障害により、障害がアクティブである（例えば、配線ネットワークにショートが発生する）にもかかわらず、一部のFPGAは正常に機能し、正しい暗号文を生成することができます。

4列目は、暗号文が異なる障害の数をカウントします。ここでも、暗号文の相違は、障害の影響がFPGAに依存していることを示しています。5列目「Same」は、同じ障害動作を引き起こす障害の数をカウントします。つまり、誤った暗号文とアラームステータス値が一致する障害、特に両方のアラームがゼロ以外で値が等しい障害です。6列目「Total」は、アクティブな障害の大部分が同一の障害動作を引き起こしていることを示しています。ここで「Total」は、誤った暗号文を生成する、または1つ以上のアラームをアサートするなど、何らかのエラーを生成する障害の数を表しています。

BYU対デフコン

このセクションでは、DEF-CON の設計と McMurtrey らが提案した DWC 技術を分析および比較します。

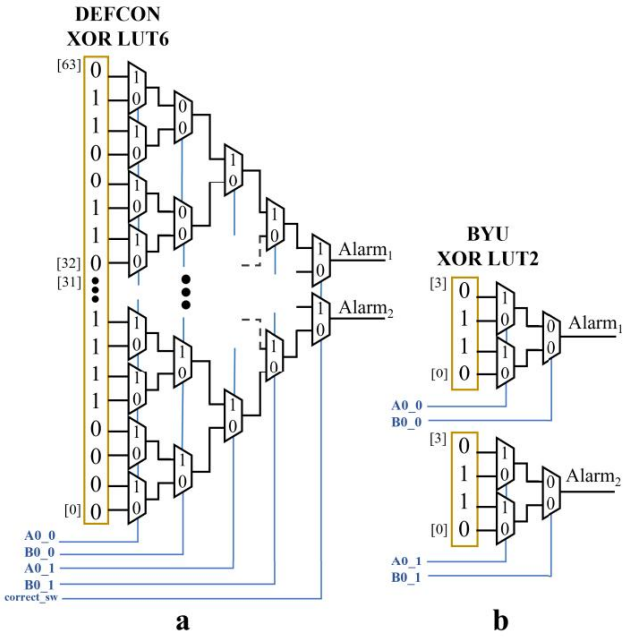

図 16. DEFCON と BYU XOR LUT 実装の比較。
BYU 設計では、2 つの LUT2 ゲートを使用して、A0_0、B0_0、A0_1、B0_1 というラベルの付いた冗長出力ビットを比較します。これは、DEFCON では 1 つの LUT のみを使用して実行されます。

表6. DEFCONとBYU LUTの使用率。

Design	LUTs
BYU	680
DEFCON	183

表7. BYU vs DEFCONの実験的故障分析（故障総数3,109,184個）。

Status	DEFCON	BYU
Benign	2,811,365	2,796,923
Active	297,808	312,243
At least one Alarm	295,911	310,381
False Positives	697	650
Missed	1,897	1,862
Untestable	11	18

[3]および両設計から得られた実験結果。ここでは、 [3]の設計をBYU 回路と呼ぶ。BYU 技術は、128 ビット AES エンジンからの各冗長出力ビットを比較するために、2 つの 2 入力 XOR LUT を使用する。これは DEFCON で使用される LUT の 2 倍である。DEFCON と BYU の回路構造をそれぞれ図16(a)と(b)に示す。DEFCON と同様に、エラー信号は 4 入力 OR LUT のバイナリツリーを使用して収集され、2 つの最終アラーム信号を駆動する。

各手法のリソース使用率は表6 に示されています。ここでは、DEFCON 設計は BYU 設計で使用される 680 個の LUT に対して合計 183 個の LUT を使用する点で、BYU 設計よりも大きな利点があることがわかります。

正確に数えることは難しいが、BYUの設計で使用されるルーティングリソースの数も

BYU 設計では追加の XOR LUT が必要となるため、ほぼ同じ割合で大きくなります。

表7は、両設計の故障検出能力を比較したものです。予想通り、DEFCON設計では良性故障の数がわずかに多く、活性故障の数がわずかに少なくなっています。これは、DEFCON監視回路で使用されるリソースが少ないためです。誤検知、未検出、テスト不能に分類される故障の数は両設計でほぼ同数ですが、このわずかな差はVivado配置配線ツールによる最適化によるものと考えられます。

したがって、DEFCON に提案された XOR パッキング戦略には、障害検出機能に関する欠点はありません。

VI. 結論

本稿では、FPGA上に実装されたコンパクトでフェイルセーフなモニターを提案する。このモニターは、モニター自体に発生する 2 番目の障害に対して堅牢でありながら、2 つの冗長機能ユニットの出力の違いを検出できる。

この手法を DEFCON (DEsign for Fail-safe in reCONfigurable systems) と呼んでいます。DEFCON では、単一の LUT で 2 つの独立した機能を提供できる新しい冗長化方式が提案されています。LUT 内の構成メモリ ビットの特別なエンコードにより、1 つの入力セットの障害に対応しながら、別の入力セットに委任された指定機能を正しく実行できます。LUT への 5 番目の入力により 4 重冗長化が可能になり、LUT 自体に障害が発生した場合でも、LUT は別の構成メモリ ビット セットに瞬時に切り替えて正常に動作を継続できます。シミュレーション結果では、監視対象の機能ユニットに障害が発生した場合、および FPGA で DEFCON モニターを実装している構成メモリ ビットで同時 (デュアル) 障害が発生した場合に、DEFCON が正しくアラームをアサートできることが示されています。

さらに、ほぼすべての二重故障テストシナリオにおいて、DEFCONはすべての機能ユニット出力をブロックし、フェイルセーフ値を強制することができます。DEFCON回路はFPGAセット上のハードウェアで検証され、リソース利用率と検出能力の点で最も類似した技術と比較されています。

謝辞

この著作物はNTESSの従業員によって執筆されました。この著作物に関する権利、所有権、および利益はNTESSではなく従業員が所有し、その内容について責任を負います。

本著作物に明示されている主観的な見解や意見は、必ずしも米国政府の見解を反映するものではありません。発行者は、米国政府が、本著作物の出版形態を米国政府の目的のために出版または複製し、または他者に出版または複製を許可する非独占的、支払済み、取消不能、かつ世界的なライセンスを保持していることを承認します。エネルギー省（DOE）は、エネルギー省公開アクセス計画に基づき、連邦政府が支援する研究成果への一般公開を行います。

参考文献

[1] K. Szurman、Z. Kotasek、「ソフトコアプロセッサNEO430用実行時再構成可能フォールトトレラントアーキテクチャ」、 Proc. IEEE 22nd Int.
Symp. Design Diag. Electron. Circuits Syst. (DDECS)、 2019年4月、pp.1–4。
[2] J. Johnson、W. Howes、M. Wirthlin、DL McMurtrey、M. Caffrey、P. Graham、およびK. Morgan、「FPGAベースの設計におけるオンラインエラー検出のための比較による複製の使用」、 Proc.
IEEE Aerosp. Conf.、 2008年3月、pp. 1-11。

[3] DL McMurtrey、「FPGAベースの設計におけるオンラインエラー検出のための比較による複製の使用」、博士論文、Dept. Elect.
ブリガムヤング大学コンピュータ工学科、プロボ、ユタ州、米国、2006年。
[4] X. Tang、E. Giacomin、A. Alacchi、B. Chauviere、およびP.-E. Gaillardon、「OpenFPGA：カスタマイズ可能なFPGAの迅速なプロトタイピングを可能にするオープンソースフレームワーク」、 Proc.
29th Int. Conf. Field Program. Log. Appl.
(FPL)、 2019 年 9 月、367 ～ 374 ページ。
[5] X. Tang、E. Giacomin、B. Chauviere、A. Alacchi、およびP.-E. Gaillardon、「OpenFPGA：カスタマイズ可能なFPGAのアジャイルプロトタイピングのためのオープンソースフレームワーク」、 IEEE
Micro、 vol. 40、no. 4、pp. 41-48、2020年7月。
[6] F. BenevenutiとFL Kastensmidt、「SRAMベースのFPGAにおける障害攻撃検出の評価」、 Proc.
18th IEEE Latin Amer. Test Symp.
(LATS)、 2017年3月、pp.1-6。
[7] J.-Y. Lee、Y. Hu、R. Majumdar、L. He、およびM. Li、「デュアル出力LUTによるフォールトトレラントな再合成」、 Proc. 15th Asia South Pacific Design Autom. Conf. (ASP-DAC)、 2010年1月、pp.
325–330。
[8] J.-Y. Lee、Z. Feng、L. He、「FPGAにおける堅牢性のためのインプレース分解」、 Proc. IEEE/ACM
Int. Conf. Comput.-Aided Design (ICCAD)、 2010年11月、pp. 143-148。

[9] L. SterponeとM. Violante、「SRAMベースFPGA向けの信頼性重視の新しい配置配線アルゴリズム」、 IEEE Trans. Comput.、 vol. 55、no. 6、pp. 732-744、2006年6月。

[10] N. Jing、J.-Y. Lee、Z. Feng、W. He、Z. Mao、およびL. He、「SRAMベースのFPGAアーキテクチャと合成アルゴリズムのSEU故障評価と特性」、 ACM Trans. Design Autom. Electron. Syst.、 vol. 18、no. 1、pp. 1–18、2013年1月。

[11] S. Zheng、H. You、G. He、Q. Wang、T. Si、J. Jiang、J. Jin、およびN. Jing、「SRAMベースのFPGAにおけるSEU軽減のための高速スクラビング技術」、Proc . IEEE Int. Symp. Circuits Syst. (ISCAS)、
2019年5月、pp.1～5。

[12] ST FlemingとD. Thomas、「FPGA構成障害の並列注入」、 Proc. Int. Conf. Field-Program.
Technol. (FPT)、 2018年12月、198-205頁。

[13] H. Mestiri、N. Benhadjyoussef、M. Machhout、R. Tourki、「障害検出対策を備えたAESのFPGA実装」、 Proc.
内部。会議コントロール、決定。情報テクノロジー。 (CoDIT)、 2013 年 5 月、264 ～ 270 ページ。
[14] F. Kahri、H. Mestiri、B. Bouallegue、M. Machhout、「セキュアハッシュアルゴリズムSHA-512の効率的な障害検出方式」、 Proc. Int.
グリーンエネルギー会議システム（GECS）、 2017年3月、pp.1-5。
[15] MN Shaker、AH Madian、MB Abdelhalim、SH Amer、AS Emara、およびHH Amer、「FPGAスイッチマトリックスのオープンフォールトがフォールト検出メカニズムに与える影響」、 Proc. 28th Int.
Conf. Microelectron. (ICM)、 2016年12月、pp. 233-236。

[16] J. Luu、J. Goeders、M. Wainberg、A. Somerville、T. Yu、K. Nasartschuk、M. Nasr、S. Wang、T. Liu、N. Ahmed、KB Kent、J. Anderson、J. Rose、およびV. Betz、「VTR 7.0：FPGA向けの次世代アーキテクチャおよびCADシステム」、 ACM Trans. Reconfigurable Technol. Syst.、 vol. 7、no. 2、
pp. 1–30、2014年7月。

[17] J. Yao、B. Dixon、C. Stroud、V. Nelson、「Virtex-4 FPGAのグローバル配線リソースのシステムレベル組み込みセルフテスト」、 Proc. 41st Southeastern Symp. Syst. Theory、 2009年3月、29～
32ページ。
[18] M. Ehabと I. Nashaat、「Verilog HDLによる高度暗号化規格（AES128、AES192、AES256）の暗号化と復号の実装」、2022年。[オンライン]。入手先：https://github.com/michaelehab/AES-Verilog

[19] Zybo Z7 リファレンス マニュアル、 Digilent、プルマン、ワシントン州、米国、2024 年。
[20] Zybo Z7リファレンスリンク、 Digilent、プルマン、ワシントン州、米国、2024年。
[21] 7シリーズFPGAS構成、ユーザーガイド、ザイリンクス、サンノゼ、カリフォルニア州、米国、
2024年。
[22] C. Fibich、M. Horauer、およびR. Obermaisser、「Artix-7およびiCE40 FPGAにおけるシステマティックフォールトインジェクション結果のデバイスおよび温度依存性」、 Proc. Design、Autom. Test
Eur. Conf. Exhib. (DATE)、 2021年2月、pp.1600–1605。

[23] NA Harward、MR Gardiner、LW Hsiao、MJ Wirthlin、「フォールトインジェクションによるソフトプロセッサのソフトエラー感度の推定」、 Proc. IEEE 23rd Annu. Int. Symp. Field-Program.
Custom Comput. Mach.、 2015年5月、pp. 143–150。

PRIYA A. BHAKTA （IEEE学生会員)は、ニューメキシコ大学アルバカーキ校でコンピュータ工学の学士号（2022年)と修士号（2023年)を取得し、現在は同大学でコンピュータ工学の博士号取得を目指しています。彼女の研究は、回路レベルのフェイルセーフ設計を通じて、重要システムにおけるフィールドプログラマブルゲートアレイ（FPGA)の信頼性向上に重点を置いています。彼女の研究分野は、集積回路ロジックです。設計および組み込みシステム。

ANDREW (DREW) SUCHANEK (IEEE シニア会員) は、米国アーカンソー州フェイエットビルのアーカンソー大学で、応用数学とコンピュータエンジニアリングの二重学士号、コンピュータエンジニアリングの修士号と博士号をそれぞれ 2013 年、2013 年、2017 年、2018 年に取得しました。

2017年より、米国ニューメキシコ州アルバカーキにあるサンディア国立研究所に勤務し、現在は技術スタッフの主任メンバーを務めています。様々な重要システム向けASIC開発を主導し、FPGA上に実装されたASICおよび回路のフェイルセーフ性を保証する自動化手法の開発に取り組んでいます。研究分野は、耐放射線性ICおよびメモリ、安全で信頼性の高いFPGA、非同期デジタル回路などです。

ジム・プラスクエリックは、ピッツバーグ大学でコンピュータサイエンスの修士号と博士号を取得しました。現在はニューメキシコ大学で電気・コンピュータ工学の教授を務めています。2012年と2017年には、ハードウェア指向のセキュリティと信頼に関するシンポジウム（HOST)の共同設立と貢献により、IEEEコンピュータソサエティより「Outstanding Contribution Award」を受賞しました。

TOM J. MANNOS は、ユタ大学で電気工学の学士号を取得し、ニューメキシコ大学で電気工学の修士号を取得しました。

彼は現在、サンディア国立研究所の集積回路設計者として、組み込み FPGA セキュリティ、セキュリティ障害分析、超伝導エレクトロニクスの研究を行っています。