電源ノイズ起因タイミング故障のデバッグにおける C 言語ベース故障検出手法の有効性評価

増田　豊[1,a)]　橋本　昌宜[1,b)]　尾上　孝雄[1]

概要：チップの動作検証時に発生した故障のデバッグを目的としたソフトウェアベース故障検出手法が提案されており, C 言語実装可能な一手法として EDM (Error Detection Mechanisms) 変換がある. EDM の電気的故障への有効性を評価するため, 我々は二つのシナリオを考えた. (1) オリジナルプログラム実行時に発生した電気的故障をデバッグしたい, (2) プロセッサに潜在する電気的故障を出来るだけ知りたい. 本研究では EDM を電気的故障高速検出用に変形し, 両シナリオにおいて電源ノイズ起因タイミング故障に有効か実験的に評価した. 評価により, EDM を用いてオリジナルプログラム実行時の故障をデバッグすることは困難であるが, 潜在的な電気的故障の発見には有用であるという結果を得た.

Performance Evaluation of Software-based Error Detection Mechanisms for Localizing Electrical Timing Failures under Dynamic Supply Noise

Masuda Yutaka[1,a)]　Hashimoto Masanori[1,b)]　Onoye Takao[1]

Abstract: For facilitating error localization, software-based error detection techniques have been proposed and EDM (error detection mechanisms) transformation is one of these techniques. To discuss the effectiveness of EDM for electrical bug localization, two scenarios are considered; (1) localizing an electrical bug occurred in the original program, and (2) localizing as many potential bugs as possible. We experimentally evaluated the error detection performance in these two scenarios under dynamic power supply noise. Experimental results show that the EDM transformation customized for quick error detection cannot locate electrical bugs in the original program in the first scenario, but it is useful for finding potential bugs in the second scenario.

1. 概要

近年の回路の微細化に伴い，論理的に正しい回路において発生する電気的タイミング故障が大きな問題となっている．電気的故障は，予期しない電源ノイズ，局所的な温度変化，クロストークノイズ等の動的変動要因により起こる [1]．動的変動要因はプログラム実行時の回路状態や動作環境に応じて変動するため，設計時に電気的故障の発生状況を正確に予期することは困難である．従って，予期しない電気的故障が製造後検証時に観測される．

チップの製造後検証では，幅広いテストパターンが様々な動作条件において与えられる．動作検証時に予期しない異常動作が観測されると，回路動作が解析され，(1) 故障発生の認識，(2) 故障箇所 (ALU やキャッシュ等)，故障時刻の把握，(3) 故障条件の特定が行われる。ここで，解析に要する時間・コストの大半は (1) と (2) で占められており [2]，これらの労力を削減することが強く求められている．

故障の発生は，システム・クラッシュ，セグメンテーション・フォルト，不当な命令コードによる例外処理等から認識される．故障認識後は発生箇所の特定が必要だが，本ステップが大きな課題となっている．故障発生から異常動作の検出までにかかる時間 (故障検出時間) は非常に長く，数 10 億サイクル以上に及ぶこともある [3]．このような長時間経過後に故障発生箇所を特定することは非常に困難である．例えば，製造後チップのデバッグではトレースバッファを用いることがあるが，記録可能な命令数が限られているため，実行命令を上記のような長い間記録し続けることは現実的ではない．従って，故障検出時間の削減は製造後チップのデバッグにおいて非常に重要である．故障発生位置特

[1] 大阪大学大学院情報科学研究科
Dept. Information Systems Engineering, Graduate School of Information Science and Technology, Osaka University
[a)] masuda.yutaka@ist.osaka-u.ac.jp
[b)] hasimoto@ist.osaka-u.ac.jp

定手法として，Park らにより IFRA(Instruction Footprint Recordingand Analysis) が提案されている．IFRA では，故障の予兆を元にすばやく検出し，追加ハードウェアを用いて回路動作履歴を解析することにより，故障発生箇所の特定を行う [4]．別の方法として，ハードウェアの追加により実装されるアサーションベースの故障検出手法も提案されている [5], [6]．この手法では，いつ，どこにアサーションを挿入するかが，ハードウェア面積に関する制約と効率的な故障検出の両立において非常に重要である [7]．

ソフトウェアベースで高速検出性を付加する手法として QED 変換がある [3]．QED は入力プログラムをアセンブリレベルで変換し，実行データ，命令に故障検出性を加える． [3] では，特定の論理故障に対する故障検出時間の 10^6 倍以上の改善 (10 億サイクル⇒数 1000 サイクル) を実験的に確認している．このような短い故障検出時間であれば，製造後チップの動作検証を劇的に効率化できる．

他のソフトウェアベース故障検出手法として，EDM (Error Detection Mechanisms) 変換が挙げられる [8]．EDM は元々，ソフトエラーの検出率向上を目的として提案されている．[8] ではデータメモリへランダムに挿入した 0-1 反転への 90% 以上の検出率，[9] ではレジスタへ挿入した 0-1 反転への 80% 以上の検出率を実験的に確認している．EDM は高級言語 (C，C++ 等) で記述された入力プログラム内の使用変数やデータを複製し，変数読み出し時に複製前後で値を比較することで故障検出性能を向上する．EDM の大きな利点は，動作検証を行うハードウェアに依存せず，高水準言語レベルで実装できる点である．ここで，変数読み出し時のチェックでは故障検出時間が長くなる事があるため，本研究では故障高速検出用にチェック箇所を変更した EDM も実装し，両 EDM を用いて評価を行う．

先行研究ではソフトエラーを対象としている一方，電気的故障に対して有効かは未検討である．本稿では EDM が電気的故障の発生位置特定に対して有効かという問題に着目する．ここで，電源ノイズが電気的故障の主要因であると仮定し，プログラム実行時電源ノイズにより起こる電気的故障に対して，EDM が機能するか実験的に評価する．そのために，製造後検証での EDM 使用シナリオを 2 つ考える．(1) オリジナルプログラム実行時に発生した電気的故障をデバッグしたい，(2) プロセッサに潜在する電気的故障を出来るだけ知りたい．一つ目のシナリオではオリジナルプログラムの故障再現，高速検出の 2 条件を満足する必要がある．一方シナリオ 2 では高速検出のみ満足すればよい．本研究では，上記 2 シナリオにおいて EDM が機能するか評価し，電気的故障への有効性を議論する．

本論文の構成は以下の通りである．2 章では，EDM の変換方法を紹介し，電気的故障の位置を特定するための必要条件を説明する．3 章では，プログラム実行時の電源ノイズにより発生する電気的故障に対して，EDM 変換が機能するか実験的に評価する．4 章では EDM の製造後検証での有効性を議論する．最後に， 5 章で結論を述べる．

2. EDM による電気的故障の発生位置特定

本章では EDM の変換方法について紹介し，製造後検証での EDM 使用シナリオを議論する．さらに，両シナリオにおいて EDM が機能するための必要条件を説明する．

2.1 EDM 変換

EDM 変換の例を図 1 に示す．以後，オリジナルの EDM [8] を EDM-O (EDM-Original) と呼ぶ．EDM-O はまず，入力プログラムをブロックに分割し，各ブロックを複製後，オリジナルブロックの後に追加する．ブロックの大きさは変数読み出し間隔と等しい．次に，全てのオリジナル - 複製ブロックの組に対して，ブロック内の実行結果を比較するチェック命令を挿入する．このチェックはメモリやレジスタで起こる 0-1 反転を検出し，ソフトエラーの検出率を上げるために，変数読み出し後に挿入している．

EDM-O はソフトエラー検出に有効である一方，高速故障検出用に改善の余地がある．図 2 左に一例を示す．変数 $a0$ への書き込み時に故障が発生し，その後 $a0$ が長時間読み出されないとする．この場合，変数書き込み後のチェックであれば故障検出時間を改善できる (図 2 右)．そこで本研究では，EDM-O に加えて，書き込み命令時にチェックを行う EDM-L (EDM for short Latency) を実装する．

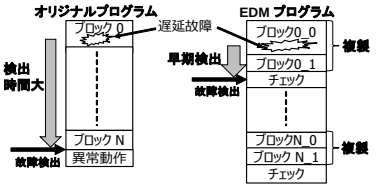

図 1 EDM プログラムによる故障検出例.

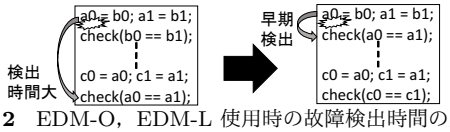

図 2 EDM-O，EDM-L 使用時の故障検出時間の差.

2.2 EDM 使用シナリオ，および位置特定への必要条件

本節では製造後検証での EDM 使用シナリオを整理し，各シナリオで EDM が満足するべき必要条件を議論する．以下の 2 シナリオについて考える．

シナリオ 1: オリジナルプログラム実行時に発生した電気的故障をデバッグしたい．

シナリオ 2: プロセッサに潜在する電気的故障を出来るだけ知りたい．

まず，シナリオ 1 での必要条件を精査した．EDM は以下の 2 条件を同時に満足する必要がある (図 3)．

条件 1: オリジナルプログラム実行時の故障を再現する．

条件 2: 動的変動要因に十分差があり，オリジナル-複製ブロック間で同じ故障が再現しない．

条件 1 はオリジナルプログラム実行時の故障発生要因を求めるために必要である．故障箇所を再現するために，

EDM はオリジナルプログラムと同様の振る舞いをしなければならない．もし EDM がオリジナルプログラムの故障を再現できなければ，故障要因の特定は不可能である．

条件 2 は EDM のチェック機構が働くために必要である．もしオリジナル-複製両ブロックで同じ故障が起これば，チェック命令は機能しない．条件 2 では，電源ノイズ等の動的変動要因により，オリジナル-複製ブロック実行時の動作条件に差がつくことが期待できる．

一方，条件 1 に関しては，オリジナル-EDM 間の動的変動要因の差により成立が難しくなる．例えば，電源ノイズは実行命令や使用メモリによって変化するため，オリジナル，EDM プログラム実行時のチップ内電源ノイズは異なる．従って，オリジナルプログラムのみ，もしくは EDM プログラム実行時のみ故障が起こるケースが考えられる．

以上より，シナリオ 1 では条件 1，条件 2 を満足する必要がある．しかし，先行研究 [8]，[9] では条件 1，2 に着目しておらず，EDM が両条件を満足するか明らかでない．次章では，動的に変動する電源ノイズにより起こる電気的故障に対して，両条件が成立するか実験的に評価する．

シナリオ 2 では潜在的な故障を多く発見したいので，オリジナルプログラムの故障再現は不要である．つまり，条件 2 のみ満足すればよく，条件 1 は必要ない．シナリオ 2 の必要条件は，シナリオ 1 の必要条件の部分集合であるため，シナリオ 1 への評価はシナリオ 2 にも有効である．

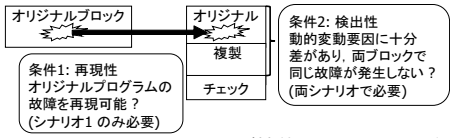

図 3 シナリオ 1 で EDM が機能するための 2 条件.

3. EDM の有効性評価

本章では EDM がシナリオ 1，シナリオ 2 において有効に機能するか実験的に評価する．電源ノイズが電気的故障の主要因であると想定し，EDM が電源ノイズや電源ノイズ起因タイミング故障に与える影響を再現する．[10] では，本評価の予備実験，およびその評価結果を述べている．

3.1 評価環境

本評価では 65nm プロセスで設計・製造された東芝 MeP プロセッサを用いる．実行プログラムとして，Mi bench [14] から 3 つの C プログラム dijkstra, sha, crc を選択する．次に，各プログラムから EDM プログラムを生成する．

EDM 変換では，2 種類のチェック命令を挿入する [8]; (1) データチェック，(2) 命令順序チェック．データチェックでは各変数 v を $v0$，$v1$ と複製し，EDM-O では，$v0$ ($v1$) 読み出し時に，EDM-L では，$v0$ ($v1$) 書き込み時に両変数値を比較する．命令順序チェックは，実行順序の不正な変更の検出が目的であり，以下のように挿入される．

まず，EDM はプログラムを基本ブロック (非分岐命令列) に分割し，命令実行順と合致するよう各基本ブロックに番号を付ける．次に，if, else if，while 等の分岐/ループ命令をチェックする．各分岐/ループの条件式評価が正しく実行されているかチェックするため，各分岐/ループ内部に，それぞれの条件式と真偽が逆になる条件式を挿入する．最後に関数呼び出し/return 文 に対するチェックを行う．各関数毎に異なる値を割り当て，関数呼び出し時に割り当て値を確認し，不正に関数内に飛んでいないか検査する．本章では，全変数を複製し，全データチェック，命令順序チェックを挿入する full-EDM を使用する．複製，チェック挿入頻度を変更した EDM については次章で議論する．

オリジナル- 複製ブロックでは，全く同じ入力データを異なるメモリ空間に格納し，それぞれのブロック内命令では，対応するメモリアドレスにアクセスしている．表 1 に EDM-L による実行サイクル数，キャッシュミス増加の様子を示す．実行サイクルは 3 〜 4 倍になり，それに伴い命令キャッシュミスも同程度増加している．全変数を複製しているため，データキャッシュミスも約 2 倍となっている．

オリジナル，full-EDM-L プログラム間での故障箇所の比較は以下の環境で計 300 回行う．実行プログラムとして MiBench[11] から dijkstra, sha, crc の 3 つを選択した．製造ばらつきの影響で，同一製造プロセスで作成されたチップ間でも異なる遅延特性を持つ．この点を踏まえて，10 種類の異なる遅延特性を持つ MeP プロセッサを用意する．それぞれの遅延特性は，標準の遅延情報ファイルに正規分布に従う 25% のばらつきを与えることで生成する．さらに，製造後チップに応じてパッケージの設計は大きく変動する．本評価では，10 種類のパッケージを，異なる 10 種類の電源電圧網を設計することにより実装する．この電源電圧網のモデルは，次節で紹介する．同様に，full-EDM-O を用いた評価も 300 回行う．

本実験ではプログラム実行時の最初に発生した故障に注目し，その発生位置の比較により条件 1 が満足されたか判別する．ここで，タイミング故障を起こす最大周期を 2ps 単位で調査する．EDM 変換時にプログラムをブロックに分割し，番号付けするため，ブロック番号の差を故障位置の差とする．この差が 0 であれば，タイミング故障は同ブロックで起こっており，条件 1 が成立しているとみなす．条件 2 は，EDM 内のチェックにヒットしたサイクルを取得し，故障検出時間を算出して評価する．EDM が故障検出に失敗した場合は，最終結果が正しいか (masked error)，誤っているか (silient error) に結果を分類する．

表 1 EDM が実行サイクル，キャッシュミス数に与える影響.

	実行サイクル数		命令キャッシュミス数		データキャッシュミス数	
	orig.	EDM	orig.	EDM	orig.	EDM
dijkstra	24512 (1.00)	69838 (2.85)	45 (1.00)	161 (3.58)	11 (1.00)	20 (1.82)
sha	30757 (1.00)	97831 (3.18)	44 (1.00)	167 (3.80)	25 (1.00)	42 (1.68)
crc	19975 (1.00)	57252 (2.87)	9 (1.00)	29 (3.22)	35 (1.00)	71 (2.03)

括弧内の値は full-EDM-L の値をオリジナルで割ったもの.

3.2 電源ノイズ考慮論理シミュレーション

本研究では，電源ノイズ考慮論理シミュレーションモデ

ル [12] を用いて，オリジナル，EDM プログラム実行時の電源ノイズ起因タイミング故障を再現する．[12] では RT レベル，ゲートレベルの MeP 回路記述の両方でプログラムを同時実行し，毎サイクル全フリップフロップ値の比較を行い，タイミング故障の発生箇所，時刻を取得する．各ゲート遅延は入力電源電圧値に応じて変動する．

上記の論理シミュレーション実行時に電源ノイズを入力するため，以下の 2 段階でプログラム実行時電源ノイズを取得する．まず，トランジスタレベルシミュレータを用いて，プログラム実行時の MeP 内消費電流を取得する．

次に，取得した電源電流変動を図 4 の電源分配網に与え，電源ノイズに変換する．基準バイアスは 1.0V である．電源ノイズはパッケージ方法に大きく依存するため，本評価では 10 種類のパッケージ設計と，対応する電源分配網を用意した．電源分配網のパラメータ設定を以下に示す．

本論文では，パッケージキャパシタの寄生容量 C_{PKG}，$R_{\mathrm{ESR\_PKG}}$，$L_{\mathrm{ESL\_PKG}}$ と，オンチップキャパシタ C_{CHIP} を変動させる．他の 5 つのパラメータは以下のように固定する．L_{BOARD}=0.1 nH，R_{BOARD}=5 mΩ，L_{BOND}= 0.3 nH，R_{CHIP}=0.1 Ω，$R_{\mathrm{ESR\_CHIP}}$= 0.3 Ω．パッケージキャパシタに対して，5 種類の設定を用意した．(1) 配置しない，(2)NPO を 1 つ配置，(3)X7R を 1 つ配置，(4)NPO 10 個を並列配置，(5)X7R 10 個を並列配置．ここで，NPO と X7R は商用でよく使用されるセラミックキャパシタであり [13]，C_{PKG}，$R_{\mathrm{ESR\_PKG}}$，$L_{\mathrm{ESL\_PKG}}$ 組はそれぞれ (100pF, 0.3Ω, 0.6nH) と (1nF, 0.6Ω, 0.6nH) である [13], [14]．オンチップキャパシタ C_{CHIP} は，3.5nF と 0.3nF の 2 種類を用意した．上記の 10 (=5×2) 種類の電源分配網を用いて電源ノイズを取得した．ノイズ波形の例を図 5， 6 に示す．

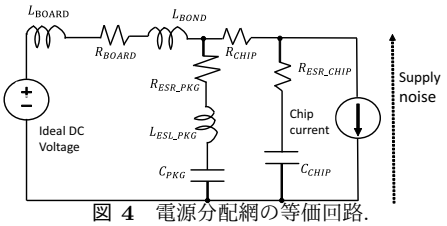

図 4　電源分配網の等価回路．

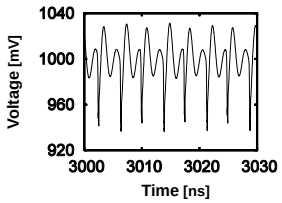

図 5　誘導性の強い波形例．

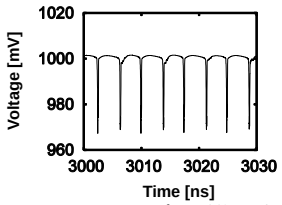

図 6　IR ドロップの顕著な波形例．

3.3　評価結果

図 7 に条件 1，条件 2 の満足割合を示す．オリジナルプログラムで発生した電気的故障 600 サンプルに対して，EDM は条件 1，2 を同時に満足せず，シナリオ 1 において機能しなかった．また条件 1，条件 2 共に満足しない例が 75 % 以上もあった．full-EDM-L と full-EDM-O を比較すると，条件 1/条件 2 満足割合に差があることが分かる．以降で，条件 1，条件 2 に対する結果の詳細を示す．

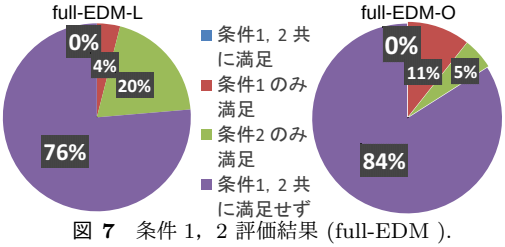

図 7　条件 1，2 評価結果 (full-EDM)．

3.3.1　条件 1

図 8 にオリジナル，full-EDM での故障箇所の差を示す．故障ブロック番号の差が小さいほど故障箇所が近く，0 であれば再現に成功している．EDM-L では， crc の 10%，dijkstra の 2 % で条件 1 が満足された一方，sha では全く再現されなかった．EDM-O では，crc で 30% 以上の故障が再現された一方，sha，dijkstra では条件 1 を満足する例は存在しなかった．全体では，EDM-L，EDM-O の故障再現率はそれぞれ 4%，11% であった．故障再現率が低い主な要因として，以下の 2 つが挙げられる．

一つ目の要因は，EDM によるプログラム実行時電源ノイズの変動である．これは EDM の複製とチェック挿入により，命令列や，メモリ，汎用レジスタの占有状況が変動するためである．図 9 に，オリジナル，full-EDM dijkstra プログラムで同一命令を実行した際のノイズ波形を示す．電源ノイズが一致していないことが分かる．図 10 に，crc オリジナルプログラムで実行している mov 命令の，1 サイクル内最低電圧の分布を示す．同じ mov 命令であっても，最低電圧が 941 mV から 947 mV と変動している．このノイズ波形の変動により故障再現が妨げられている．

二つ目の要因は，EDM により実行プログラムが長くなる点である (表 1)．実行プログラムが長くなることで，オリジナルプログラム実行時に発生しなかった故障が新たに出現しやすくなる．さらに，複製とチェック挿入により，プログラム内の命令構成も変化する．図 11 に，sha オリジナル，sha-full-EDM-L プログラムでの構成命令内訳を記載する．EDM の複製による使用変数の増加に伴う lw 命令の増加や，チェック命令挿入による beq 命令の増加が顕著である．上記のような実行命令列の変動は，プロセッサの動作を変えるだけでなく，電源ノイズの差も大きくするため，その結果，故障の再現が困難となっている．

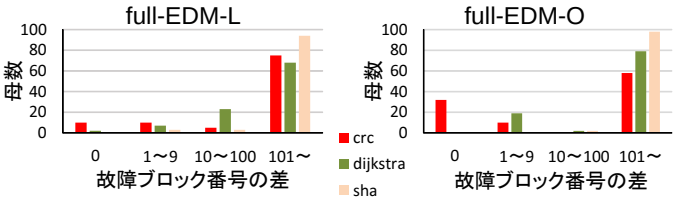

図 8　条件 1 評価結果．各プログラムにおける母数は 100.

3.3.2　条件 2

図 12 に，masked error，silent error，および検出された故障の内訳を示す．EDM-L では，77% の故障がマスクされ，2% が silent error であった一方，EDM-O ではマスク

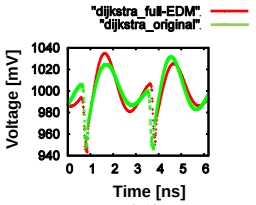

図 9　EDM 変換前後での電源ノイズ比較.

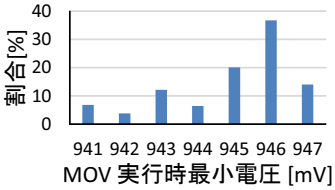

図 10　MOV 命令実行時の最小電圧分布 (crc-original).

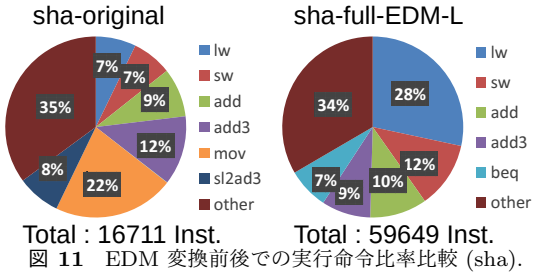

図 11　EDM 変換前後での実行命令比率比較 (sha).

率が 87%，silent error が 7% であった．

マスクされなかった故障の早期 (1000 サイクル以内) 検出割合に着目すると，EDM-O が 30% であるのに対して，EDM-L では 86% もある．実行結果に影響を及ぼす電気的故障に対して EDM-L の検出性能は高い．換言すれば，シナリオ 2 において EDM は有効である．故障検出時間が長いケースに関しては，最初に発生した故障を見逃し，2 番目以降の故障を検出する傾向を観測した．一方で，EDM-O の故障検出性能は高くない．EDM-L よりも silent error の割合が多く，故障検出時間が長い．製造後検証で起こった故障を早期検出する目的では，明らかに EDM-L の方が EDM-O よりも優れている．

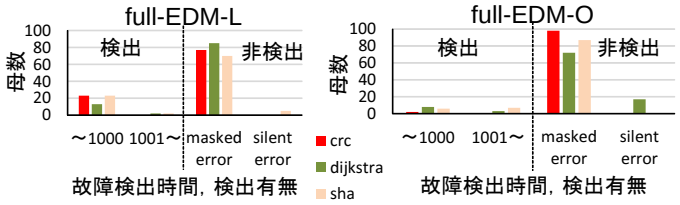

図 12　条件 2 評価結果．各プログラムにおける母数は 100.

4. EDM の 製造後検証における有効性評価

前章より，シナリオ 1 では EDM-L，EDM-O 共に機能せず，シナリオ 2 では EDM-L が有効であるという結果を得た．ここでの問題は，EDM が実行プログラムを変えることにより、条件 1 の満足割合が著しく低下する点である．そこで，ここからは EDM による複製，チェック挿入量を削減し，製造後検証において有効か検討する．

チェック頻度はプログラムの変更量を制御する主要素である．この頻度を操作することで故障再現性，検出性能が変動すると予想できる．例えば，チェック頻度を上げれば検出性能が向上する一方，命令列と電源ノイズの差が大きくなり故障再現性が劣化すると考えられる．このトレードオフは，QED [3] の inst_min，inst_max に相当する．

本章では，EDM-L のチェック挿入頻度を変えて，シナリオ 1，シナリオ 2 での有効性を評価する．以下の 2 つの EDM-L 変換を dijkstra，crc，sha に対して実行する．

- less-data-check-EDM：プログラム内の使用変数をランダムに選択し，その変数にのみ複製とデータチェックを行う．dijkstra では 17 変数 3 つ，crc では 15 変数 2 つ，sha では 10 変数 2 つ をそれぞれ選択した．命令順序チェックは全て挿入する．
- no-code-check-EDM：全変数を複製し，全てのデータチェックを挿入する．命令順序チェックは挿入しない．

ここで，前章使用した full-EDM は，複製，チェック量が最も多いため，チェック削減後 EDM のみ実装している．表 2 に，チェック削減前後 EDM の挿入チェック数をまとめた．less-data-check-EDM-L では 29% ～ 51%，no-code-check-EDM-L では 41% ～ 46% 挿入チェック数が削減されている．表 1，2 より，full-EDM-L では 15～35 サイクルに 1 回チェックが実行されている．

表 2　EDM-L によるチェック命令挿入数.

	full-EDM	less-data-check-EDM	no-code-check-EDM
dijkstra	1950 (1.00)	1237 (0.63)	1061 (0.54)
sha	5015 (1.00)	2450 (0.49)	2939 (0.59)
crc	3017 (1.00)	2145 (0.71)	1637 (0.54)

括弧内の値は full-EDM-L の値で割ったもの.

4.1 シナリオ 1

図 13 に，less-data-check-EDM-L，no-code-check-EDM-L の評価結果を示す．full-EDM-L と同様，各 EDM-L に対して 300 回ずつ評価した．両 EDM-L において，条件 1，2 を同時に満足するケースは極めて少なく，less-data-check-EDM-L で 0%，no-code-check-EDM-L で 2% であった．

図 7，13 より，less-data-check-EDM-L で条件 1 満足割合がわずかに改善している (4%⇒ 6%)．図 14 に sha-no-code-check-EDM-L の実行命令比率を示す．図 11 と比べると，オリジナルプログラムの命令比率に近づいている．一方，条件 2 満足割合は full-EDM-L から大きく劣化している．条件 2 評価結果については次節で説明する．

以上より，条件 1，条件 2 の満足割合はチェック挿入頻度とトレードオフの関係にあり，条件 1，条件 2 満足割合を同時に改善することは容易ではない．さらに，チェック挿入頻度の削減による条件 1 の改善よりも条件 2 の劣化が顕著であった．以上より，このトレードオフからシナリオ 1 に最適な故障挿入頻度を検出することは困難である．

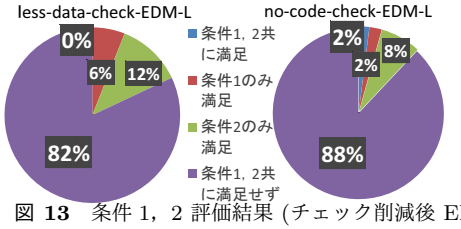

図 13　条件 1，2 評価結果 (チェック削減後 EDM-L).

4.2 シナリオ 2

図 7，13 より，早期故障検出率は less-data-check-EDM-L で 21% から 12% に，no-code-check-EDM-L で 21% から 10% に劣化している．しかし，no-code-check-EDM-L

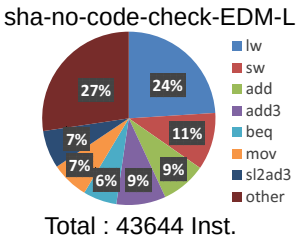

図 14　実行命令比率 (sha-no-code-check-EDM-L).

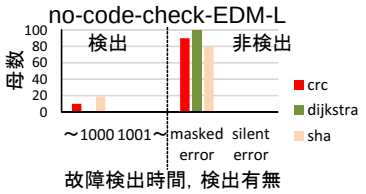

図 15　条件 2 評価結果. (no-code-check-EDM-L).

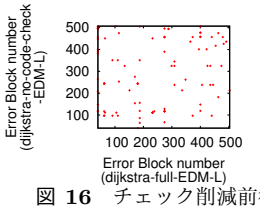

図 16　チェック削減前後 EDM での故障ブロック比較.

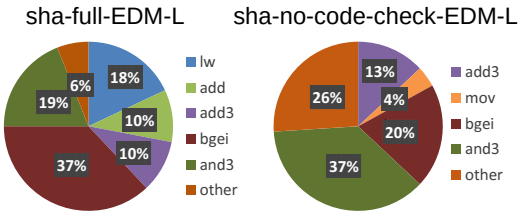

図 17　チェック削減前後 EDM での故障時命令比較.

では実行結果を変える故障への検出性能は高く，検出率は 96.6% であった (図 15)．従って，no-code-check-EDM-L も full-EDM-L と同様シナリオ 2 において有効である．

もし full-EDM-L と no-code-check-EDM-L が異なる潜在的な故障を発見できれば，両 EDM を使用することで製造後検証をより効率よく出来る．そこで，この両 EDM で検出した故障を比較した．図 16 に，dijkstra-full-EDM-L と dijkstra-no-code-check-EDM-L 間の故障ブロック番号の比較図を示す．分布は広がっており，両 EDM の故障発生状況が異なることが分かる．図 17 に，sha-full-EDM-L, sha-no-code-check-EDM-L での故障発生時の命令を示す．両 EDM で，故障発生時の実行命令は大きく異なる．この結果から，両 EDM では異なる故障が起こり，検出されている．以上より，EDM は製造後検証用の様々なプログラムを生成でき，潜在的な故障の位置特定を効率化できる．

5. 結論

本研究は C 言語実装可能な故障検出手法である EDM 変換に着目し，EDM が電源ノイズ起因タイミング故障に対して有効か実験的に評価した．EDM の電気的故障への有効性を評価するため，2 つの使用シナリオを設定した; オリジナルプログラム実行時に発生した電気的故障をデバッグしたい (シナリオ 1)，プロセッサに潜在する電気的故障を出来るだけ知りたい (シナリオ 2)．本研究では，EDM-O, EDM-L が両シナリオに対して有効か評価した．ここで，EDM-O は先行研究で提案された元々の EDM で，EDM-L は故障高速検出用に実装した EDM である．評価結果より，EDM-L，EDM-O の両方とも，シナリオ 1 では有効に機能しなかった．これは，EDM により実行命令列，電源ノイズが変動し，オリジナルプログラム実行時に発生した故障が再現されにくいことに起因する．一方，シナリオ 2 では，EDM-L が実行結果を変える故障の 85% 以上を検出し，潜在的な故障の発見に有用であるという結果を得た．チェック挿入頻度の変更により，故障発生条件の異なる様々な EDM プログラムを生成でき，これらを製造後検証時に用いることで，潜在的な故障の発見を効率化できる．

謝辞

本研究の一部は STARC との共同研究による．

参考文献

[1] P. Patra, “On the cusp of a validation wall,” *Design & Test of Computers*, vol. 24, no. 2, pp.193–196, June 2007.

[2] D. Josephson, “The good, the bad, and the ugly of silicon debug,” *Proc. DAC*, pp.3–6, 2006.

[3] D. Lin, T. Hong, Y. Li, S. Eswaran, S. Kumar, F. Fallah, N. Hakim, D.-S. Gardner, and S. Mitra, “Effective Post-Silicon Validation of System-on-Chips Using Quick Error Detection,” *IEEE Trans. CAD* , vol. 33, no. 10, pp.1573–1590, Oct. 2014

[4] S.-B. Park, T. Hong, and S. Mitra, “Post-silicon bug localization in processors using instruction footprint recording and analysis (IFRA),” *IEEE Trans. CAD*, vol. 28, no. 10, pp.1545–1558, Oct. 2009.

[5] A.A. Bayazit, S. Malik, “Complementary use of runtime validation and model checking,” *Proc. ICCAD*, pp.1052–1059, 2005.

[6] M. Boule, Z. Zilic, “Incorporating efficient assertion checkers into hardware emulation,” *Proc. ICCD*, pp.221–228, 2005.

[7] S. Mitra, S. A. Seshia, and N. Nicolici, “Post-silicon validation opportunities, challenges and recent advances,” *Proc. DAC*, pp.12–17, 2010.

[8] M. Rebaudengo, M.S. Reorda, M. Torchiano, and M. Violante, “Soft-error detection through software fault-tolerance techniques,” *Proc. DFT*, pp.210–218, 1999.

[9] N. Nicolescu, R. Velazco, “Detecting soft errors by a purely software approach: method, tools and experimental results,” *Proc. DATE*, pp.57–62, 2003.

[10] 増田豊，橋本昌宜，尾上孝雄, “電源ノイズ起因電気的故障を対象としたソフトウェアベース高速エラー検出手法の性能評価,” DA シンポジウム, 2014.

[11] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown, “MiBench: A free, commercially representative embedded benchmark suite,” *Proc. Workload Characterization*, pp.3–14, 2001.

[12] M. Ueno, M. Hashimoto, and T. Onoye, “Real-time On-chip Supply Voltage Sensor and Its Applicatoin to Trace-based Timing Error Localization,” *Proc. IOLTS*, 2015.

[13] L.D. Smith, R.E. Anderson, D.W. Forehand, T.J. Pelc, and T. Roy, “Power distribution system design methodology and capacitor selection for modern CMOS technology,” *IEEE Trans. Advanced Packaging*, vol.22, no.3, pp.284–291, Aug 1999.

[14] T. Roy,L. Smith, and J. Prymak, “ESR and ESL of ceramic capacitor applied to decoupling applications,” *Proc. EPEP*, pp.213–216, 1998.