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

増田 豊* 橋本 昌宜* 尾上 孝雄*
*大阪大学 大学院情報科学研究科 情報システム工学専攻

チップの動作検証時に電気的タイミング故障のデバッグが課題となっている．電気的故障は電源ノイズ，温度変化等の動的変動要因に依存し，発生条件が複雑で発生から検出までの時間が長いという問題がある．C 言語ベースで実装可能な高速故障検出手法として EDM (Error Detection Mechanisms) 変換がある．EDM では入力プログラムをブロック単位で複製し，複製したブロックの実行結果を定期的に比較する．電気的故障に適用するには，複製前の故障を再現し，複製ブロックに同一の故障が発生しない必要がある．本稿では，動的変動要因の中でも特に電源ノイズに着目し，EDM の電源ノイズ起因タイミング故障に対する有効性を評価する．評価により，EDM の故障再現は対象プログラムによって大きく変化し，故障検出時間は全体の 70 % で 1000 サイクル以内であるという結果を得た．

Performance Evaluation of Software-based Quick Error Detection Technique for Localizing Electrical Failures due to Dynamic Power Supply Noise

Yutaka Masuda* Masanori Hashimoto* Takao Onoye*
*Dept. Information Systems Engineering, Graduate School of Information Science and Technology, Osaka University

Localizing electrical timing bug is one of the most time-consuming tasks in post-silicon validation. For quickly detecting bugs, we focus on a C-language-based error detection technique called EDM (Error Detection Mechanisms), which decomposes the program into blocks, duplicate each block and inserts check instructions for every pair of original-and-duplicated blocks. To successfully detect electrical timing bugs in the original program, two conditions must be satisfied. In this paper, we experimentally evaluate the error detection performance of EDM by investigating whether these two conditions are satisfied under dynamic power supply noise. Experimental results show that error reproduction rate by EDM changes significantly by executing program, and in no-masked samples, error detection latency by EDM is shorter than 1,000 cycles in the 70 % of samples.

1 序論

近年の回路の微細化，動作周波数の高速化に伴い，電気的タイミング故障が大きな問題となっている．電気的故障の発生要因として，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などの動的変動要因が挙げられる [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，設計時に電気的タイミング故障の発生状況を正確に予期することは困難である．その結果，予測できていなかった電気的故障が製造後チップの動作検証時に観測される．

チップの動作検証では，幅広いテストパターンが様々な動作環境と組み合わせて実行される．動作検証時に予期しないシステムの異常動作が観測されると，回路動作の解析が行われる．この解析では，(1) 故障発生の認識，(2) 故障発生箇所 (ALU やキャッシュコントローラなど)，故障発生時刻の把握，(3) 故障発生条件の特定が行われる。ここで，解析にかかる時間・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の異常動作により認識される．故障発生を検出する方法として，End-result-check が挙げられる．End-result-check では，プログラムの実行結果をあらかじめ用意した正解値と比較することで故障の発生をチェックする．故障認識後は故障発生箇所の特定が必要であるが，この工程が大きな課題となっている．故障発生箇所の特定をが困難な原因として，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) が非常に長い (図 1) ことが挙げられる．故障検出時間は数十億サイクル以上に及ぶこともあり [3]，このような長時間経過後に故障発生箇所の

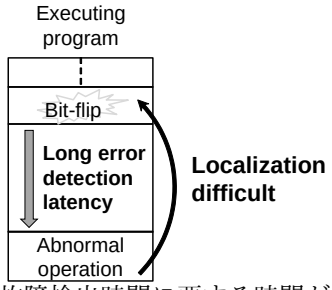

図 1: 故障検出時間に要する時間が長い問題

特定を行う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がしばしば用いられるが，記録可能な命令数が限られており，上記のような長い検出時間の間，実行命令を記録し続けることは出来ない．このため，故障検出時間の削減が強く求められている．

故障発生位置特定手法として，IFRA(Instruction Footprint Recording and Analysis) が提案されている．IFRA では，故障の予兆によりすばやく検出し，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]．このアサーションベースの手法では，いつ，どこにアサーションを挿入するかが非常に重要である．しかし，ハードウェア資源への制約は厳しく，限られたハードウェア資源を用いて故障を素早く検出するのは容易ではない．以上より，ソフトウェアベースで実装可能な高速故障検出手法に注目が集まっている．

ソフトウェアベースの高速故障検出手法として，QED (Quick Error Detection) 変換 [3, 7, 8] がある．QED では入力プログラムに対するソフトウェアベースの多様な変換を行い，故障検出用プログラムを生成する．入力プログラムとしては，ランダム命令テスト，アーキテクチャ特有の機能を検証するためのテスト，オペレーティング・システムやゲームなどのエンドユーザ・アプリケーションなどが挙げられる．QED は入力プログラムをブロックに分割し，各ブロックを複製し，オリジナルブロックの後に追加する．全てのオリジナル - 複製ブロックの組に対して，QED はブロック内の中間結果を比較するためのチェック命令を挿入する．上記の定期的なチェックにより，QED は故障検出時間を大幅に改善している．[3] では，プログラム実行中にランダムに発生した 0-1 反転に対して，10^6 倍以上の故障検出時間の改善を実験的に確認している．しかしながら，Gao らの研究で対象としたランダムな 0-1 反転は，電気的故障の発生モデルと一致していないため，電気的故障に対して QED が機能するか調査する必要

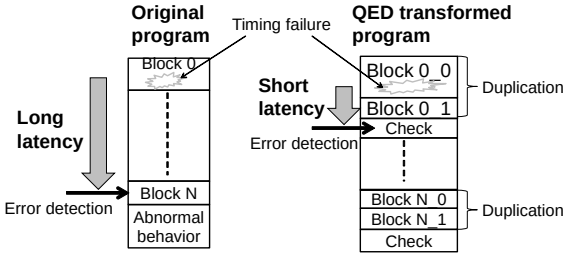

図 2: QED 変換による故障検出

がある．

本研究では，電気的故障の高速検出・故障発生位置特定に対して QED が有効かを議論する．電源ノイズが電気的故障の主な要因であると想定し，電源ノイズにより発生する電気的故障に対して，QED が機能するか実験的に評価する．

本論文の構成は以下の通りである．2 章では，QED の変換方法を紹介し，QED が電気的故障の発生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．最後に，4 章で結論を述べる．

2 QED による電気的故障の発生箇所特定

本章では QED の変換方法について紹介し，QED がプログラム実行時に発生した故障を検出するための必要条件を議論する．

2.1 QED 変換

QED はプログラム実行時に発生した故障を素早く検出するために考案された，ソフトウェアベース変換手法であり，変換手法として，EDM (Error Detection Mechanism) [9] や EDDI (Error Detection by Duplicated Instructions for Validation) [10,11] 等の手法を含む．

QED 変換，および故障検出を図 2 に示す．QED では，最初にオリジナルプログラムを複数のブロックに分割し，それぞれのブロックを複製する．全てのオリジナル - 複製ブロックの組に対して，QED はチェック命令を挿入してオリジナル - 複製ブロック間の実行結果を比較する．従って，QED 変換後プログラムでは全てのブロック組において，オリジナルブロック，複製ブロック，そしてチェック命令の順に実行する．QED では，プログラム実行時に発生した故障を早期検出するために，定期的なチェック命令の挿入を行っている．上記の変換はアセンブリレベル，もしくは C/C++ レベルで実装可能である．

チェック命令により，プログラム実行中に発生した故障を検出するには，オリジナル - 複製ブロック間で命令の実行条件に差を持たせることが非常に重要である．もし，オリジナル - 複製ブロックの実行条件が完全に同一ならば，全く同じエラーが両ブロッ

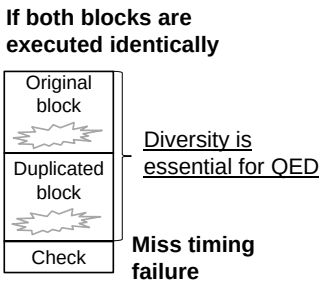

図 3: 故障検出に必要な実行条件の多様性

クに発生し，チェック命令が機能しない (図 3)．両ブロックに同時に故障を起こさないため，例えば，オリジナルブロックと複製ブロックで使用するメモリ空間，汎用レジスタを分割することが考えられる．両ブロックの実行命令において，例えばメモリ・レジスタへのアクセス時間に差が出るため，同じ故障が両ブロックに発生する可能性を削減できる．

2.2 電気的故障の発生位置特定の必要条件

本節では QED が電気的故障の発生箇所を特定するために必要な条件について述べる．電気的故障発生箇所を正しく求めるために，以下の 2 つの条件を満足する必要がある (図 4).

- 条件 1: QED 変換後プログラムは，オリジナルプログラムの故障発生状況を再現する必要がある．すなわち，オリジナルプログラム実行時に発生した故障は QED 変換後プログラム実行時にも発生しなければならない．
- 条件 2: QED はオリジナル - 複製ブロック間の両方に故障が発生しないよう，両ブロック間の実行条件に十分な差を与えなければならない．

条件 1 はオリジナルプログラム実行時の故障発生箇所，発生原因を求めるために必要である．もし QED がオリジナルプログラム実行時の故障を再現できなければ，故障発生箇所の特定は不可能である．また，QED 変換によって新たに故障が発生する場合も考えられる．QED によってオリジナルプログラムの故障発生状況が変化する理由は，プログラム実行時のプロセッサ内消費電流が QED 変換前後で変動し，その結果，プログラム実行時の電源ノイズも． QED 変換前後で変動するためである．

条件 2 は，QED 変換後プログラムのチェックが機能するために必要である．そのために，オリジナル - 複製ブロック実行時の動的変動要因に差を与え，実行条件に多様性を持たせることが必要である．

一方，条件 1 に関しては，オリジナル - QED 変換後プログラム実行時の動的変動要因の差が大きければ大きい程，成立が難しい．例えば，オリジナルプログラム実行時と QED 変換後プログラム実行時ではチップ内電源ノイズが異なるため，オリジナルプログラム実行時に発生していた故障が QED 変換後は発生しなくなる事例や，逆に QED 変換により新たに故障が発生する事例が起こりうる．

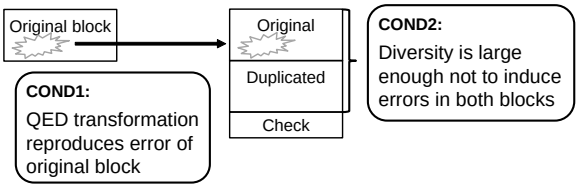

図 4: 電気的故障の発生箇所を特定するために必要な 2 条件

表 1: EDM が実行サイクル，キャッシュミス数に与える影響

	execution cycles		inst. cache misses		data cache misses	
	original	EDM	original	EDM	original	EDM
dijkstra	24512 (1.00)	73513 (2.99)	45 (1.00)	150 (3.33)	11 (1.00)	20 (1.82)
sha	30757 (1.00)	120487 (3.92)	44 (1.00)	213 (4.84)	25 (1.00)	52 (2.08)
crc	22887 (1.00)	62713 (2.74)	51 (1.00)	63 (1.23)	47 (1.00)	65 (1.38)

オリジナル - EDM 変換後プログラムにおける実行サイクル数，およびキャッシュミス数

3 EDM 変換に対する実験的評価

本章では，EDM 変換が条件 1，2 を満足するか実験的に評価する．本実験では電源ノイズが電気的故障を引き起こす主な要因であると想定している．

3.1 実験環境

本実験は東芝 MeP プロセッサをを対象として，65nm ライブラリを用いて設計したレイアウト済みの設計を評価に用いた．

実行プログラムとして，Mi_bench [12] に含まれる 3 つの C 言語プログラム dijkstra, sha, crc を選択した．次に，EDM 変換フィルタを作成し，各入力プログラムから EDM 変換後プログラムを生成した．オリジナル - 複製ブロックでは，全く同じ入力データをそれぞれ異なるメモリ空間に格納し，各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する．表 1 に EDM による実行サイクル数，キャッシュミスの増加を示す．EDM により，実行サイクル数が約 3 〜 4 倍に増加していることが分かる．一方，キャッシュミスは，dijkstra, sha と比較して，crc における増加割合が低い．

次に，オリジナルプログラム，EDM 変換後プログラム実行時に発生する電気的故障を論理シミュレーションで発生させた．本実験で使用する論理シミュレーションでは RT(Register Transfer) レベル，ゲートレベルの 2 種類の MeP 回路記述を用いた．RT レベルはゲート遅延が無いため，動作周波数，電源ノイズに関わらず，タイミング故障は発生しない．一方，ゲートレベルでは回路記述にタイミング情報を含むため，入力電源ノイズ，動作周波数によっては，

タイミング制約を満たせず，故障が発生する可能性がある．本研究では，電源ノイズに応じてゲート通過時の遅延量が変動する論理シミュレーションモデル [13] を用いて，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を評価した．RT レベルとゲートレベルシミュレーション実行時のフリップフロップの値を比較することで，タイミング故障がいつ，どこで発生したか知ることが出来る．このシミュレーションモデルについて，次節で説明する．

本稿では，オリジナルプログラム，EDM 変換プログラム間の故障発生状況の比較を以下の条件で計 300 回実行した．

まず，様々なばらつきを持つ仮想 MeP チップを用意した．各ゲートの遅延時間が正規分布に従って変動し，その標準偏差を平均遅延時間の 25 % として仮想チップを 10 個生成した．パス毎に遅延特性に変化が生じるため，条件 1，2 に影響を与える可能性がある．次に，プリント基板やパッケージ，チップ内電源網の設計や選定によって電源ノイズは大きく異なるそこで，10 種類の電源等価回路を準備し，電源ノイズ評価に用いた．電源等価回路は次節で説明する．これらの 10 × 10 = 100 種類のチップ・電源網の組み合わせに対して，dijkstra, sha，crc の 3 種類のプログラムを実行し，全ての (チップ，パッケージ，実行プログラム) 組に対して，タイミング故障が起きる最大のクロックサイクル時間を求めた．

本実験では，プログラム実行時に最初に発生する故障に着目し，その故障に対して条件 1 が満足されているか評価した．ここで，EDM 変換時でオリジナルプログラムをブロックに分割した時，それぞれのブロックに対して先頭から番号を付けた．これにより，故障発生箇所のブロック番号を比較することで，条件 1 が満足されているか評価出来る． 条件 2 は，EDM 変換後プログラム実行時に発生した電気的故障が，最終的にチェック命令により検出されているか評価した．このとき EDM 変換後プログラム実行時に発生した故障に対して，チェック命令が機能しないことがある．そこで，故障検出に失敗した場合を，以下の 2 つのケースに分類した．(1) Error masking : 故障は発生したが，その故障がメモリ・汎用レジスタには伝播せず，実行結果が正しいままプログラムが終了し，EDM 変換後プログラムのチェック機構が機能しない事例． (2)Fault Silent Violation : 故障は発生し，実行結果も誤っているが，EDM のチェック命令が機能しない事例．本実験では，RT レベルシミュレーションの実行結果と比較して，故障検出されなかった際の分類を行った．

3.2 電気的故障再現用論理シミュレーション

本稿では，プログラム実行時に発生する電気的故障を評価するため，プログラム実行時の電源ノイズに応じて，プロセッサ内のゲート通過時遅延量が動的に変動するゲートレベル論理シミュレーションモデル [13] を用いた．

電源ノイズに応じて変動するゲート遅延を実現するため，入力電源ノイズ値に相当する信号を用意し，その信号値によって遅延が変動するゲート素子を作成した．電源ノイズとゲート遅延の関係は事前に回路シミュレーションで求めた．このゲート素子を論理シミュレーションに用いることで，入力電源ノイズに応じて回路内で発生する電気的故障が評価できる．

この電気的故障評価論理シミュレーションモデルを実施するためには，プログラム実行時の電源ノイズ波形を入力する必要がある．この電源ノイズ波形を取得を以下の 2 段階のアプローチで行った．

まず，オリジナル - EDM 変換後プログラムを MeP プロセッサで実行したときの消費電流を求めた．次に，取得した電流変動を図 5 のチップ・パッケージ等価回路に与え，プログラム実行時の電源ノイズを取得した．標準電源電圧は 1.2 V とし，いずれもトランジスタレベルの回路シミュレータで評価した．

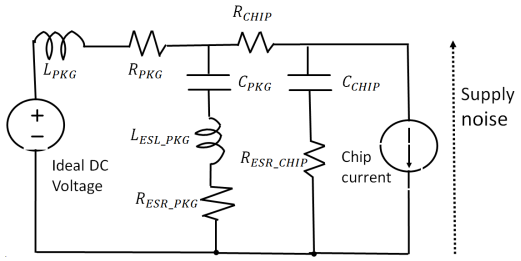

図 5: 電源ノイズ評価用電源分配網等価回路

IR ドロップ・ノイズや，LC 共振ノイズといった，多様な電源ノイズに対する評価を行うため，電源分配網等価回路のパラメータを変化させた．L_{PKG} を 100pH から 4nH まで，R_{PKG} を 0.25 Ω から 10 Ω まで, R_{CHIP} を 200 mΩ から 2Ω まで, C_{PKG} を 2.35 nF から 2.35 pF まで C_{CHIP} を 75 pF から 300 pF までで変化させた. $R_{\mathrm{ESR\_CHIP}}$, $R_{\mathrm{ESR\_PKG}}$，$L_{\mathrm{ESR\_PKG}}$ はそれぞれ 1.34 Ω, 3 Ω，100 nH とした．取得した LC 共振ノイズが顕著な波形の例を図 6 に， IR ドロップノイズが顕著な波形の例を図 7 に，それぞれ示す．

3.3 評価結果

3.3.1 条件 1

図 8 にオリジナル - EDM 変換後の dijkstra, sha で発生した電気的故障発生箇所の再現度を，図 9 に

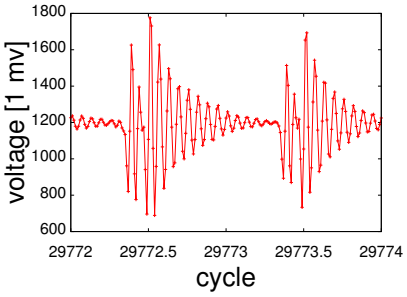

図 6: LC 共振ノイズが顕著なノイズ波形の例

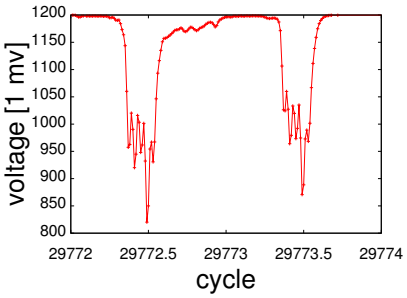

図 7: IR ドロップノイズが顕著なノイズ波形の例

crc の電気的故障発生箇所の再現度を示す．ここで故障発生箇所の再現度は，オリジナル - EDM 変換後プログラム実行時の故障発生ブロックの番号の差であらわす．もしブロック番号の差が 0 であれば，EDM 変換がオリジナルプログラム実行時に発生した電気的故障を再現している．図 8， 9 より crc と dijkstra, sha の間で再現度の分布に大きな差が見られた．crc では，60% が条件 1 を満たしており，再現度の低い例は，約 20% であった．この結果からは，EDM 変換後プログラムが，オリジナルプログラムの故障をかなり再現できている．一方，dijkstra, sha に関しては，条件 1 を満たしている例は無く，特に sha では，再現度の低い例がほとんどであった．以上より，「EDM が条件 1 を満足するか」という命題に，実行プログラムが寄与する影響は非常に大きいという結果が得られた．

3.3.2 条件 2

評価結果を図 10，11 に示す．EDM 変換後プログラムで発生した電気的故障 300 パターンの内，245 パターンはマスクされた．一方，残りの 55 パターンではタイミング故障が実行結果に影響を与えた．この 55 パターン全てに対して，故障検出時間を評価した結果，EDM は 55 パターン全てで，発生した電気的故障を検出した．つまり本実験で試行した 300 パターンでは，実行結果が誤っていて，かつ検出に失敗したケースは無かった．次に，故障検出時間に

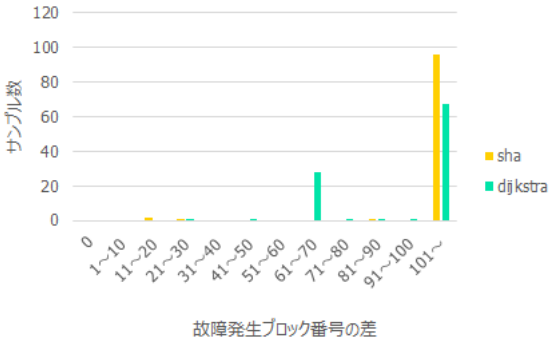

図 8: オリジナル - EDM 変換後プログラム実行時の故障発生箇所のずれ (条件 1，dijkstra, sha)

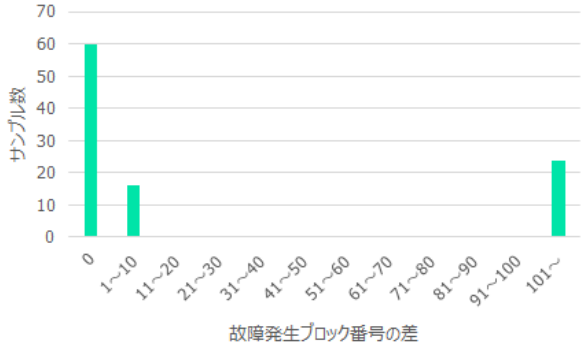

図 9: オリジナル - EDM 変換後プログラム実行時の故障発生箇所のずれ (条件 1，crc)

ついて述べる．全体では，故障検出された内の 約 70 % が，1000 サイクル以内に検出された．プログラム別に見ると，crc では検出されたサンプル全てで故障検出時間が 1000 サイクル以内であった．一方，dijkstra，sha では，故障検出時間が非常に長い場合もあり，EDM 変換後プログラムにおいて電気的故障の検出が遅れた顕著な例は，全体で 25% 程度であった．

4 結論

本稿では，QED 変換から，C 言語ベースの変換手法である EDM に注目し，EDM 変換の電気的故障検出に対する性能評価を行った．まず，故障検出実現に必要な要件を，2 つの条件に整理し，実際にその条件が満足されるか評価した．条件 1: EDM がオリジナルプログラムの故障を再現可能か，という条件に関して，3 種類の実行プログラムのうち，1 つではよく満足されたが，残りの 2 つでは全く満たされなかった．条件 2 : EDM 変換後プログラムにおいて，オリジナル-複製ブロック間で実行条件の差が十分であり，その結果チェック機構が機能するか，という条件に関しては，故障検出されず実行結果が誤っているケースは存在しなかった．故障検出された場合の約 70 % で 1000 サイクル以内に検出された一方，約 25 % で故障検出時間が長いという

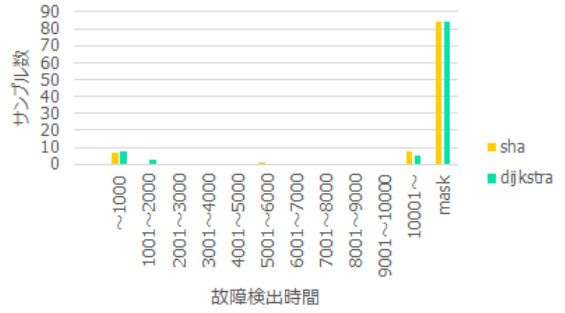

図 10: 故障検出時間 (条件 2，dijkstra, sha)

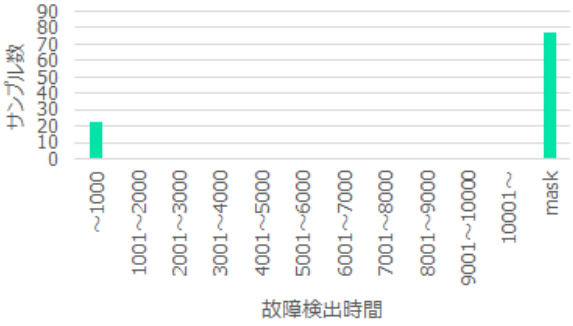

図 11: 故障検出時間 (条件 2，crc)

結果が得られた．

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] T. Hong, Y. Li, S.-B. Park, D. Mui, D. Lin, Z.A. Kaleq, N. Hakim, H. Naeimi, D. S. Gardner and S. Mitra, ”QED: Quick Error Detection tests for effective post-silicon validation,” *Proc. ITC*, pp.1–10, 2010.

[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 and 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] D. Lin, T. Hong, Y. Li, F. Fallah, D.S. Gardner, N. Hakim and S. Mitra, ”Overcoming post-silicon validation challenges through Quick Error Detection (QED),” *Proc. DATE*, pp.320–325, 2013.

[8] D. Lin, T. Hong, F. Fallah, N. Hakim and S. Mitra, ”Quick detection of difficult bugs for effective post-silicon validation,” *Proc. DAC*, pp.561–566, 2012.

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

[10] N. Oh, P.P. Shirvani and E.J. McCluskey, ”Error detection by duplicated instructions in super-scalar processors,” *IEEE Trans. Reliability*, vol. 51, no. 1, pp.63–75, Mar 2002.

[11] N. Oh, S Mitra and E.J. McCluskey, ”ED4I: error detection by diverse data and duplicated instructions,” *IEEE Trans. Computers*, vol. 51, no. 2, pp.180–199, Feb 2002.

[12] 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.

[13] M. Ueno, M. Hashimoto, T. Onoye, “Trace-Based Fault Localization with Supply Voltage Sensor,” *Proc. TAU*, pp.77–81, 2014.