動的トレースを用いたアセンブリ命令レベルの割込み処理カバレッジ解析

肥塚 真由子[1,a)]　黒田 亮[1,b)]　松崎 秀則[1,c)]　渡邊 竜明[2,d)]

概要：組込みシステムにおいて，割込み処理はリアルタイム OS 等で要求される高速応答性のための重要な技術である．割込み処理によって優先度の高いタスク等に直ちに対応することが可能となる一方で，割り込み処理の制御が不適切な場合には予期しない不具合の要因ともなる．割込み処理に起因する不具合は，ソフトウェアの静的解析では検出・再現できないケースも多く，デバッグコストを増大させる原因となっている．本研究では，割込み処理に着目したテストの品質の向上のために，テスト対象となるシステムを実際に動作させた際の動的トレースログを用いることで，アセンブリ命令レベルの割込み処理のカバレッジを解析し，これを可視化するツールの提案を行う．

1. はじめに

近年の各種産業の IT 化に伴い，様々な電気・電子機器に内蔵される組込みシステムの重要性が高まってきており，ハードウェアとソフトウェア共に高い性能が求められている．特に車載向け組込みシステムでは機能的な性能に加えてリアルタイム性などの時間的な制約がある [1]．時間的な制約を満たすためには，高速な応答性を必要とするタスクを優先的に動作させる機能が必要となる．割込み処理は，優先度の高いタスクの動作要求あった場合に，動作中のタスクを一時停止させて優先度の高いタスクを先に動作させる機能であり，組込みシステムにおける高速応答性を確保するのに重要な役割を果たしている [2]．

割込み処理は組込みシステムにおいて重要な技術であるが，ソフトウェアとハードウェアの協調動作によって実現するため，ソフトウェアの静的解析のみでは検出されないハードウェアを含めたシステムとしての不具合が発生する可能性がある．そこで，実際の組込み開発における割込み処理を原因とする不具合に関する調査を行った．割込み処理が多く活用されるシステムの 1 つである車載系組み込みドライバのシステム開発を調査対象として，バグ件数とバグ分類，デバッグに要した工数を集計した．その結果，割込み処理の不具合解消にかかる工数が，割込み処理以外を原因とするバグのデバッグにかかる工数よりも多いという調査結果が得られた．割込み処理によるバグを未然に防ぐためには，割込み処理が動作するテストが十分になされていることをシステムレベルで検証する必要があり，そのためには割込み処理に着目したカバレッジ分析ツールが必要である．そこで，本報告では動的トレースを解析することで，割り込み処理テストのカバレッジをシステムレベルで検証するツールの提案を行う．

[1] 株式会社東芝 技術統括部 研究開発センター
コンピュータアーキテクチャ・セキュリティラボラトリー
[2] 株式会社東芝　ストレージ&デバイスソリューション社
システム・ソフトウェア推進センター
a) mayuko.koezuka@toshiba.co.jp
b) akira.kuroda3@toshiba.co.jp
c) hidenori.matsuzaki@toshiba.co.jp
d) tatsuaki.watanabe@toshiba.co.jp

2. 提案手法

提案手法では，動的トレースとアセンブリコードを入力として，割込み処理のカバレッジ率を算出する手法について述べる．システムの検証テストにおいてカバレッジは大きな役割を果たしているが，システムの挙動は複雑なものが多く，システムの状態を十分にカバーするためにはシステムの挙動をどこまで正確に捉えることができるかという点が課題となる．システムでは割込み処理の発生箇所が無限に存在しうるために，テストケースも無限に増大してしまう [3]．割込み処理が発生可能な箇所全てをテストすることは不可能であるため，割込み処理に起因する不具合を検出するための網羅的な挙動の検証を保障しながら割込み処理の発生箇所を縮減する必要がある [4]．そこで，本提案では，ソフトウェアのアセンブリ命令レベルで割込み処理を解析することで割込み発生箇所の縮減を実現し，アセンブリ命令レベルでの割込み処理のカバレッジを算出する手法の提案を行う．

割込み処理のカバレッジ算出には，アセンブリ命令レベ

ルでの割込み処理の発生箇所の抽出が必要とされる．アセンブリコードは各アセンブリ命令の PC（プログラムカウンタ）とアセンブリ命令のニーモニックによって構成されており，動的トレースはアセンブリコードで示された命令コードの実行順序が示されている．各アセンブリ命令は固有の PC が与えられているため，動的トレースではアセンブリ命令の正確な実行順序を得ることが可能となる．

さらに，正確なカバレッジ率を算出するためには，動的トレースを用いてアセンブリコードの各命令におけるクリティカルセクションを判定する必要がある．割込み処理はシステム実行時におけるいずれのタイミングでも発生する可能性があるため，共有資源の保護や割込み処理よりも優先度の高いタスクが存在する場合には，割込み処理の発生をを禁止するクリティカルセクションの設定が必要不可欠な技術となる．また，割込み処理のカバレッジを算出するためには，割込み処理の発生箇所を集計した値とクリティカルセクションを除いた割込み処理が発生可能な区間を判定することが必要であり，クリティカルセクションを抽出することでより正確に割込み処理発生可能区間を判定することが可能となる．そこで，動的トレースとクリティカルセクションの制御関数の PC から割込み処理発生可能区間に分類される命令コードの判定を行う．

このように，アセンブリ命令レベルで割込み処理の発生位置と割込み処理発生可能区間を判定することで，割込み処理のテスト検証者が必要とする正確なカバレッジを算出することが可能となる．

3. 割込み処理カバレッジ解析ツール

本章では，割込み処理の発生位置情報とカバレッジ情報を可視化するツールを提案し，実アプリケーションの動的トレースを用いてツールの有用性を示す．

ツールの構築にあたり，トレース情報を効率的に解析・可視化するためのプラットフォームである Polyspector™ をベースとして用いた [5]．提案ツールは関数毎の割込み処理のカバレッジ率の可視化だけでなく，関数毎のさらに詳細なアセンブリ命令レベルのでの割込み処理の発生箇所の可視化を実現している．なお，入力となる動的トレースは ARM 社の CPU コアを割り込み処理解析のターゲットとして採用し，ARM® 内部の実行命令の情報を表す ETM トレース（Embedded Trace Macrocell）情報を動的トレースとして用いるものとする [6]．また，割込み処理を示す関数もしくはアセンブリ命令とクリティカルセクションの制御を行う関数もしくはアセンブリ命令は，ツールのユーザーによって事前に設定される．

図.1 では，提案ツールにおいて重要な一部機能を示されている．上部グラフに関数毎の割込み処理のカバレッジ率が示されており，カバレッジ率が 78.9% とされた関数について，上部グラフに示されたある関数の詳細が下部グラフ

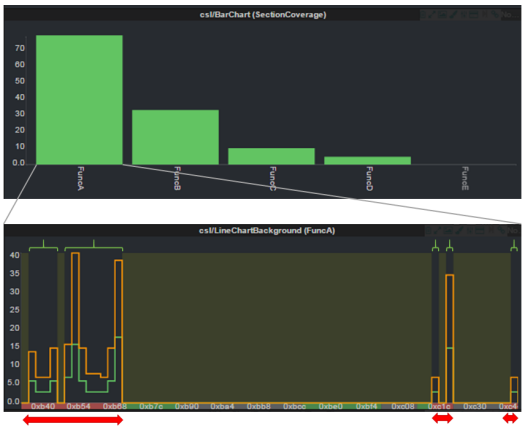

図 1　割込みカバレッジ率と割込み発生位置

に表示されている．下部グラフは，横軸を PC，縦軸を割込み処理の発生回数として 2 つの ETM トレースの結果がグラフ表示されている．下部グラフでは割込み可能区間に属する命令が 19 個存在し，そのうち割込み処理が 15 命令で発生していることが確認できるため，上部グラフにおいて割込み処理のカバレッジ率が 78.9% と表示される．このように提案手法を用いることで，割込み処理の正確なカバレッジ率とカバレッジ算出に関連する詳細な情報をアセンブリ命令レベルで表示するツールが実現の実現が可能となった．

4. まとめ

組み込みシステムにおいて重要な割込み処理に着目し，割込み処理のテスト検証のためのツールの提案を行った．提案ツールでは，割込み処理のテストに必要とされる割込み処理のカバレッジ率の表示に加えて，関数毎にアセンブリ命令レベルの割込み処理の発生回数やセクションの分類も実現した．このツールを用いることによって，開発において多くのデバッグコストを必要とする割込み処理のテストのクオリティが向上すると考えられる．

参考文献

[1] A. Burns and A. J. Wellings, *Real-time systems and programming languages*. Addison-Wesley, 2010, vol. 2097.
[2] J. Sanchez and M. P. Canton, *Embedded systems circuits and programming*. CRC Press, 2012.
[3] B. Marick *et al.*, “How to misuse code coverage,” in *Proceedings of the 16th International Conference on Testing Computer Software*, 1999, pp. 16–18.
[4] C. Kaner *et al.*, “Measurement issues and software testing,” 2001.
[5] “Toshiba Develops SSD Simulation and Analysis Platform for Design Optimization,” https://www.toshiba.co.jp/rdc/rd/detail_e/e1506_01.html, 2015.
[6] “Embedded trace macrocell architecture specification,” *Arm Limited*, 2007.