社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－SLDM－134／2008－EMB－8（26）
2008 / 3 / 28

マルチコアSoCの高度な観測を可能とするプログラマブルなデバッグ支援ハードウェアの開発

鈴木 紀章† 酒井 淳嗣† 鳥居 淳†

†NEC システムIPコア研究所 〒229-1198 神奈川県相模原市下九沢 1120
E-mail: †{n-suzuki@ha.jp.nec.com, jsakai@bc.jp.nec.com, s-torii@ay.jp.nec.com}

あらまし 組み込み機器で問題が増大している観測が困難なバグに対処し，開発効率を向上させるため，高度な条件を用いてフレキシブルにマルチコアSoC各部を観測可能とするデバッグ支援ハードウェアを開発した．デバッグ支援ハードウェアは，SoC各部の観測プローブと，高度な観測機能の実現や観測条件の柔軟な変更を可能とするデバッグプロセッサから構成される．これらの特徴を持つデバッグ支援ハードウェアを，NECエレクトロニクスの携帯電話用SoCであるMedity M2へ適用した．この結果，チップの1%以下の回路規模で，既存手法では確認が困難なバグ事象に対する観測性の改善により，開発効率を向上可能であることが確認できた．

キーワード マルチコア，デバッグ

Design of a Programmable Debug Support Mechanism for Efficient Analysis in Multi-core SoC

Noriaki SUZUKI† Junji SAKAI† and Sunao TORII†

† System IP Core Research Laboratory ,NEC 1120, Shimokuzawa, Sagamihara, Kanagawa, 229-1198 Japan
E-mail: †{n-suzuki@ha.jp.nec.com, jsakai@bc.jp.nec.com, s-torii@ay.jp.nec.com}

Abstract To improve the bug observability in the embedded systems, we have developed a Debug Support Mechanism that helps Multi-core SoC users to trace the precise behavior of each SoC blocks with user-supplied complex trigger conditions. Our Debug Support Mechanism is composed of observation modules for each target block in the SoC, and a Debug Processor that brings much flexibility in observation functions and conditions. We have applied this Debug Support Mechanism to the Medity M2, a one-chip mobile terminal SoC by NEC Electronics. As a result, we have achieved the improvement of the bug observability by our Debug Support Mechanism. The die area occupied by our Mechanism is less than 1% of the logic part in the Medity M2, and we can safely say that the impact to the die area is small enough.

Keyword Multicore, Debug

1. はじめに

旧来より，組み込み機器のソフトウェアデバッグには，CPUデバッガや，CPU実行トレースが主に用いられてきている．

CPUデバッガは，SoCが搭載するCPUコアのJTAG-ICE機能を用いてCPUの実行制御を行う．これは，ブレイクポイントによるCPUの実行停止，ステップ実行などが活用可能である．ただしCPUの停止を行うことができないリアルタイムシステムなど，適用できないケースが存在する．また，適用が可能なケースであっても，バグ発生までに長時間を要するケースなど，CPUデバッガで逐一追跡する事が困難な事象も存在する．

CPU実行トレースは，CPUのプログラム実行経路，レジスタ値の変遷など，CPUの実行履歴を取得することが可能である．CPUデバッガで確認が困難なケースでは，CPUの実行トレースによりバグ発生までの実行履歴を取得し，解析する手法が有効となる．代表的なものとして，MIPS社のPDtrace™[1]，ARM社のETM(Embedded Trace Macrocells™)[2]などが挙げられる．

一方で近年の組み込み機器への要求の高度化に伴い，マルチコア構成をとるSoCの製品適用が進みつつある．CPUコアを複数搭載したもの，DSPを搭載したもの，更に3Dグラフィックに代表されるハードウェアアクセラレータを搭載したものなど，SoCは複雑化の一途を辿る．

このようなマルチコアSoCを対象とした場合，CPUデバッガやCPU実行トレースには限界がある．例えばCPUデバッガで特定のCPUコアを停止させた場合，その他のコアが動き続けることによりシステムが不正な動作をする．またCPUコアの停止を伴わないCPU

実行トレースを用いた場合でも，特に複数コア間の連携の観測や，複数コアの動作タイミングの確認など，適用できない事象が多数存在する．

そこで近年では，MIPS Technologies 社の OCI®(On Chip Instrumentation)[3], ARM 社の CoreSight™[4]など，マルチコア SoC の観測性を向上する取り組みも進められている．これらの技術では，コア間の連携ブレイク機能や連携トレース機能など，マルチコア SoC に向けた機能の強化が図られている．

MIPS 社の OCI が備える特徴として，多くのバスや CPU コアに対応していることが挙げられる．MIPS コアはもちろんのこと，Altera®社の Nios® Embedded Processor，AMD 社の Geode™ GX and LX processor をはじめ，その他のコアにも対応している．バスは AMBA，OCP，SiliconBackplane などに対応している．

ARM 社の CoreSight は，1 本のピンで SoC から CPU 実行トレース情報を引き出すシングル・ワイヤ・デバッグ・ポートを備える．近年の SoC では必要 I/O ピン本数が増大し，デバッグ用に I/O ピンを確保する事が難しくなる中，少ない I/O ピン本数で情報出力が可能である．

しかし，これまで提案されているマルチコア SoC に向けたデバッグ技術には二つの課題がある．

一つ目の課題は，トレース情報の取りこぼしの問題である．一般に SoC からのトレース情報出力にはデバッグ出力ポートを用いる．デバッグ出力ポートのバンド幅は限られているため，トレース情報を圧縮する仕組みの導入や，トレース情報に優先度付けを行う取り組みが行われている．しかし，SoC 内部の各コアやバスの動作周波数，扱うデータの bit 幅は以前に比べて大幅に高まっており，これら複数を同時に観測したトレース情報を全て出力することは不可能である．従って必要なトレース情報を取りこぼしてしまう可能性が高まり，これを防ぐためにはデバッグ出力ポートのバンド幅に見合うよう観測対象を制限しなければならない．

二つ目の課題は，測定期間の問題である．一般に，デバッグ出力ポートのバンド幅を超える高速なトレース情報を扱うために，SoC 内部にトレース情報バッファを備える手法が用いられている．しかし大規模なトレース情報バッファを用いると SoC のコストが増大することから，バッファ容量は非常に限られている事が一般的である．従って高速な観測対象のトレース情報を扱える期間は短く制限されており，有効なデバッグ情報が取得できない場合がある．

そこで本研究では，これらの課題を解決するため，高度な条件を用いてフレキシブルに SoC 各部を観測可能とするデバッグ支援ハードウェアを開発した．

本稿では，デバッグ支援ハードウェアの構成及び実 SoC への適用結果，利用方法について述べる．以下 2 章ではデバッグ支援ハードウェアの構成について述べる．3 章では SoC への適用事例の評価について述べ，4 章でデバッグ支援ハードウェアの利用方法について述べる．最後に 5 章で本稿をまとめる．

2. デバッグ支援ハードウェア

2.1. 思想

概念上の思想

既存技術の二つの課題は，いずれもデバッグ出力ポートのバンド幅の制限により引き起こされる．極めて大きいバンド幅があれば，トレース情報の取りこぼしも発生せず，SoC 内部のトレース情報バッファを備える必要もない．しかし現実的には，デバッグ用に割り当てられるピン数は限られており，SoC 内部の動作周波数向上やコア数増加のトレンドからも十分なバンド幅を期待することはできない．

一方で本当にデバッグに必要な情報はごく一部である事が多い．デバッグの際，しばしば大量のトレース情報を取得して保存しておき，後で解析する手法が取られる．しかしデバッグに必要な情報は，不具合発生時の近傍のトレース情報のみであることや，散発するイベント周辺のごく一部のトレース情報が全て揃っていればよいなど，限られていることが多い．

このようなケースにおいて，有効なトレース情報を抽出することができれば，膨大な量のトレース情報を全て出力する必要はなくなる．

そこで，デバッグ支援ハードウェアでは，観測対象は数多く確保したうえで，SoC 内部でトレース情報に相当するイベントデータを解析し，データ量の少ないデバッグ情報へと処理したうえで SoC 外部に出力する．

この概念を図 1 に示す．まず観測可能なイベントを多数確保するために，マルチコア SoC の各部に観測プローブを導入する．観測プローブは，イベントの発生を検出すると，イベントデータを生成し，イベントデータ処理部に送る．イベントデータ処理部は，複数の

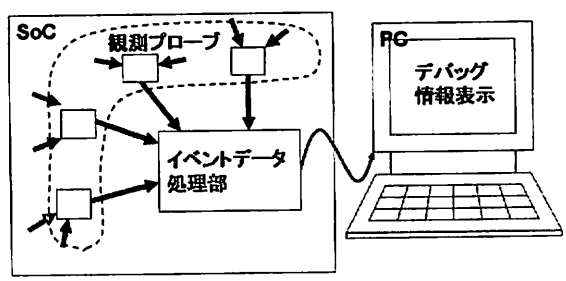

図 1 デバッグ支援ハードウェアの概念

観測プローブからイベントデータを収集し，データ量が少ない有効なデバッグ情報へと再構築する内部処理を行う．そして再構築されたデバッグ情報はSoC外部に送信され，開発者はそれをホストPC上で表示することで分析を行う．

実装上の思想

デバッグ支援ハードウェアは，SoCへの実装上の課題として回路規模と配置配線の二点を考慮した構造をとる事を基本とする．

まず，回路規模は十分小さい必要がある．一般に，SoC全体の回路規模のうち，デバッグ用回路の割合が大きいことは，SoC本来の機能を構成する回路規模に対してコストアップが大きい事を意味し，受け入れられない．

次に，配置配線が困難となる事態を避けるよう注意する必要がある．デバッグ支援ハードウェアは，SoC各部の観測プローブからの情報をイベントデータ処理部に集めて処理する事を基本とするため，配線集中により配線不能となる問題，長距離配線による遅延の問題に注意した構成をとる必要がある．

2.2. 全体構成

2.1節で述べた思想を実現するために，デバッグ支援ハードウェアは，主にデバッグプロセッサ，観測プローブ，観測プローブ制御ユニット及び内部メモリからなる構成をとる．この全体構成を図2に示す．図2において，デバッグプロセッサ，観測プローブ制御ユニット，内部メモリが前出のイベントデータ処理部に相当する．

各部の概要は次のとおりである．まずデバッグプロセッサは各観測プローブで検出したイベントデータを処理し，有効なデバッグ情報に加工して外部出力を行う．観測プローブはSoC各部でのイベントを検出して観測プローブ制御ユニットにイベントデータを送信する処理を行う．観測プローブには，割り込み信号などの信号変化を観測する信号観測プローブと，バスアクセスを観測するバス観測プローブがある．観測プローブ制御ユニットは，各観測プローブの観測条件設定や，観測の開始，停止などの制御を行う．内部メモリには，

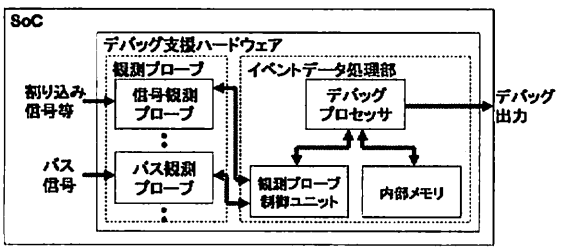

図2 デバッグ支援ハードウェアの全体構成

デバッグプロセッサ用の命令及びデータを格納する．

2.3. デバッグプロセッサ

デバッグ支援ハードウェアにデバッグプロセッサを導入するにあたって，小規模なプロセッサの回路規模と命令/データメモリ容量でイベントデータ処理が可能な専用小型プロセッサを開発した．

デバッグプロセッサに求められる要件は二つある．一つ目の要件として，デバッグ支援ハードウェアの回路規模は極力小さい必要があることから，デバッグプロセッサもデバッグ支援ハードウェアの小型化を実現できる構成である必要がある．一方で二つ目の要件として，bit幅が大きいイベントデータを扱う必要がある．

これらの要件を既存のプロセッサコアで満足することは難しい．既存の4bit，8bitマイコンのコアを用いた場合，回路規模は小さく抑えられる．しかしbit幅が大きいイベントデータは分割して扱う必要があり，コアの動作周波数は低く設計されているため，十分な性能を得ることは困難である．一方で大きなbit幅を扱える32bitマイコンのコアは，デバッグ支援ハードウェアの回路規模を許容不能なほど増大させてしまう．

そこでデバッグプロセッサとして，専用小型プロセッサを開発する方針とした．

まずレジスタ幅は32bitとした．デバッグプロセッサが扱う観測イベントデータのうち典型的なものとして，バスアクセスのイベントデータが挙げられる．ここで，特にバスアクセスのイベントデータのうち，32bitのアドレス値を用いて様々な判断を行う処置が頻繁に発生することから，レジスタ幅を32bitで実装する方針とした．

一方で命令幅は16bitの構成とした．命令bit幅を狭く取ることで命令コード容量を小さく抑える事が可能となり，命令コード格納用の内部メモリ容量を削減することも可能になる．しかし，命令bit幅が狭くなりすぎると，32bitの即値を扱う場合に必要な命令数が多大になる問題が生じる．そこでデバッグプロセッサでは，32bit即値を数命令で扱うことが可能な16bitの命令幅を採用した．

また，命令セットはプロセッサ自体の回路規模を削減するため，種類を必要最低限のものに絞った．これは，それぞれ単純な加減算，ビット演算，シフト演算，分岐命令，ロードストア命令のみとした．

2.4. 観測プローブ

デバッグ支援ハードウェア用の観測プローブとして，観測プローブ側でイベントデータを通知する状況を限定する，信号観測プローブとバス観測プローブを開発した．

観測プローブを開発するにあたって，注意すべき課題が二つある．

一つ目の課題は，処理するイベントデータの情報量の問題である．複数の観測プローブからのイベントデータは一旦観測プローブ制御ユニットに集められ，デバッグプロセッサが読み出して内部処理を行う．このため観測点の信号をそのまま接続する方式では，デバッグプロセッサが常にチェックする必要がある．この場合，単一の観測プローブであっても変化が高速である場合処理が間に合わず，複数の観測プローブを対象とする事は不可能となる．

二つ目の課題は，配置配線の問題である．イベントデータは，複数の観測プローブで検出されて観測プローブ制御ユニットに集まる．ここで，観測点の信号をそのまま全て観測プローブ制御ユニットに接続すると，配線混雑が発生し，配置配線が困難となる．また，チップ周辺部の高速な信号変化をそのまま配線で集める場合，多数の長距離配線が発生することから，配線遅延や配線コストの問題により配置配線が困難となる．

これら二つの課題は，いずれも観測プローブから送られてくるイベントデータのデータレートに起因する．データレートを低減することができれば，デバッグプロセッサによる内部処理が可能になる．更に観測プローブから観測プローブ制御ユニットへの配線を集約することや配線遅延を緩和することが可能となり，配置配線の問題に対処することができる．

このため，観測プローブ側でイベントデータを通知する状況を限定することでデータレートの低減と配線の集約が可能な構成を基本とし，信号観測プローブとバス観測プローブを開発した．

信号観測プローブは，信号変化を検出した場合のみイベントデータを観測プローブ制御ユニットに送る．信号変化が起きた場合のみ通知する手法を用いることで，観測プローブから出力する必要があるイベントデータのデータレートを減少させる．

更に，信号観測プローブでは通知対象とする信号変化を限定する手法も用いる．通知対象はデバッグプロセッサから設定可能なよう構成し，状況に応じて絞り込み内容を変更することで，デバッグプロセッサの内部処理能力に余裕を持たせる効果がある．割り込み信号のように単一の信号で構成される場合，信号の立ち上がり，または立ち下がりエッジに検出を限定して通知する方法を取ることがある．複数 bit からなる制御信号の場合，信号が特定の bit パターンになった事を検出して通知することも可能である．

また，データレートが減少したことを活用して，必要に応じて複数信号を符号化して信号観測プローブからの配線本数を縮減する．

バス観測プローブは，バスアクセスの成立を検出した場合のみイベントデータを観測プローブ制御ユニットに送る．バスアクセスが成立した場合のみ通知することで，バス観測プローブから出力する必要があるイベントデータのデータレートを減少させる．更に，バスの制御信号はバス観測プローブ側で処置が完結することから，必然的に観測プローブから観測プローブ制御ユニットへの配線本数が削減される．

また，バス観測プローブでは通知対象とするバスアクセスを限定する手法も用いる．通知対象は，信号観測プローブ同様デバッグプロセッサから設定可能にする．限定方法として，アドレスやデータのビットパターンを用いて限定する方法，バースト/シングルアクセスを区別する方法，リード/ライトアクセスを区別する方法などがある．

また，信号観測プローブ同様，必要に応じて複数信号を符号化してバス観測プローブからの配線本数を縮減する方法を活用する．

2.5. 観測プローブ制御ユニット

観測プローブで検出したイベントデータの収集と，観測プローブの制御を行う観測プローブ制御ユニットを開発した．この構成を図 3 に示す．

収集されたイベントデータを一時的に保存するため，観測プローブ制御ユニットは内部にイベントデータバッファを備える．イベントデータバッファを備えることにより，一時的にデバッグプロセッサの処理容量を超えるイベントデータが到着しても，取りこぼしなく処理することが可能となる．

また，観測プローブ制御ユニットは，観測プローブの制御を行うため，観測プローブ用の設定レジスタを備える．設定レジスタはデバッグプロセッサから読み書きが可能で，観測プローブのイベント絞り込み条件の設定，イベント検出の開始，停止などを設定することができる．

イベントデータ収集から内部処理までの流れは次のようになる．まずそれぞれの観測プローブがイベントを検出すると，イベントデータは一旦イベントデータバッファに保存される．この時，観測プローブからのイベントデータは，シリアライズして収集される．イベントデータバッファはデバッグプロセッサから直接読み出すことが可能で，デバッグプロセッサは読み出したイベントデータを用いて必要情報の抽出と有効なデバッグ情報への再構築を行う．この際デバッグプロセッサはイベントデータバッファ溢れが起こらないよう，観測プローブの絞り込み条件を設定し，必要に応じて条件変更を繰り返しながら戦略的に不要データの排除と有効なデバッグ情報への再構築を行う．

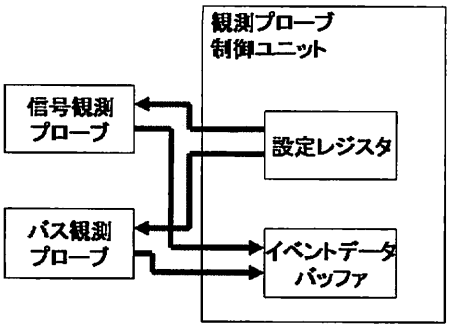

図 3 観測プローブ制御ユニットの構成

3. デバッグ支援ハードウェアの適用と評価

これらの特徴を持つデバッグ支援ハードウェアを，NEC エレクトロニクスの携帯電話用 SoC である Medity™ M2[5]へ適用した．

Medity M2 は，アプリケーションおよびモデム機能を 1 チップに集積した SoC である．Medity M2 の諸元を表 1 に示す．

また，Medity M2 のバス構成と，デバッグ支援ハードウェアの観測プローブの設置状況を図 4 に，デバッグ支援ハードウェアの諸元を表 2 に示す．観測プローブは，バス観測プローブとして AXI バスと AHB バス対応のプローブを，信号観測プローブとして割り込み信号と省電力制御信号のプローブを導入した．バス観測プローブは，DRAM 及び NOR Flash へのアクセス観測，及び APB バスを経由するハードウェアモジュールの制御，設定レジスタへのアクセス観測が可能な構成とした．

このような構成をとるデバッグ支援ハードウェアを Medity M2 に導入した結果，チップの 1%以下の回路規模で，実 SoC 上に実現可能である事が確認できた．

回路規模は内部メモリを含めて 100Kgate となった．Medity M2 のロジック部の回路規模は 15Mgate である

表 1 Medity M2 の諸元

アプリケーション CPU	ARM1176JZF-S™ 500MHz
モデム CPU	ARM1156T2-S™ 250MHz
システムバス	AXI 64bit 166MHz
DRAM	MobileDDR-SDRAM 32bit 166MHz
モデム	WCDMA, GSM/GPRS, HSDPA
LCD 解像度	WVGA
ビデオコーデック	H.264, D1, 30fps
ペリフェラル	USB2.0HS, CCP, SD, IrDA, SPI, etc
ゲート数	Logic: 15Mgate, SRAM: 12Mbit
テクノロジ	65nm, 7Cu, 1.2V

ことから，デバッグ支援ハードウェアの内部メモリを含む回路規模はロジック部の 0.7%程度に相当する．ここで Medity M2 の SRAM 部の面積も考慮するとデバッグ支援ハードウェアの占める割合は更に低下することから，コストにほとんど影響がない規模で実現可能なことが分かった．

更に，配置配線の問題の発生も見られなかった．このため，提案したデバッグ支援ハードウェアの構成が実 SoC へ十分適用可能である事も確認できた．

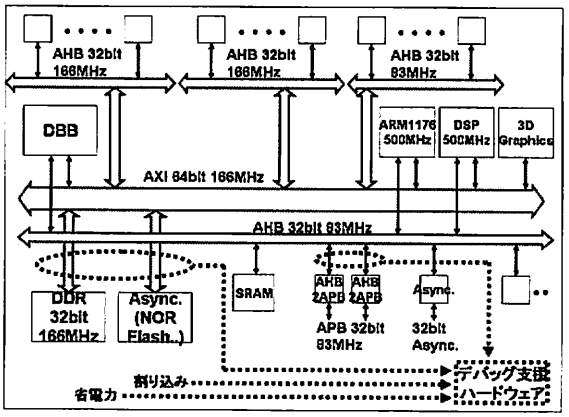

図 4 Medity M2 のバス構成と観測プローブ

表 2 デバッグ支援ハードウェアの諸元

デバッグプロセッサ	83MHz
内部メモリ	1Kbyte
バス観測プローブ	AXI 166MHz (DRAM, NOR Flash) AHB 83MHz (レジスタ)
信号観測プローブ	32bit(割り込み信号観測用) 32bit(省電力信号観測用)

4. デバッグ支援ハードウェアの利用

本章では，これまで述べたデバッグ支援ハードウェアを利用して実際のデバッグを行う具体事例について述べる．

4.1. 規定動作からの逸脱の観測

ごく稀に発生する不具合を対象とする場合，しばしば原因の追求に大きな労力を必要とする．ここで組み込み機器において，一定周期で規定の動作を繰り返す処理が頻繁に見られる．不具合はこの規定の動作から逸脱することで発生することがある．

この時従来技術であれば，周期動作のトレース情報を取得することで観測を行うが，効果的な情報を得られない場合も多い．一例として頻繁にアクセスされる共有メモリ領域を観測対象とする場合，デバッグ出力

ポートのバンド幅制約によりトレース情報をすべて取得できないケースが発生する．発生頻度が低いものを観測対象とする場合は長時間トレース情報を取得し続ける方法も用いられるが，バグと無関係なトレース情報が多く効率的でない．

そこでデバッグ支援ハードウェアを用いた手法では，規定動作からの逸脱を検出した場合にのみその周辺のイベントデータを保存する手法を用いる．まずバスや割り込み信号など，観測対象からイベントデータの取得を続ける．デバッグプロセッサは取得したイベントデータが規定の動作に沿ったものであるか随時確認する．ここで規定の動作を続けている間はイベントデータを随時破棄する．規定の動作からの逸脱をデバッグプロセッサが検出した場合，前後の一定期間のイベントデータをそのまま保存し，以後のイベントデータの取得を停止する．

このように，デバッグプロセッサによって異常点を検出してその周辺のイベントデータを保存する方法により，異常点周辺の詳細なイベントデータを取得することができる．

4.2. アクセスホットスポット解析

性能未達バグに対処するために，アクセスのホットスポット解析が行われる．

この時，従来技術ではバスのアクセストレースを出力し，後で統計処理する手法が有効な手段になり得るが，適用できないケースがある．特にマルチコアなどの複数バスマスタを対象とした場合，全コアのトレース情報を出力するのはデバッグ出力ポートのバンド幅の問題から困難なことが多い．

そこでデバッグ支援ハードウェアを用いた手法では，バスアクセスのイベントデータを，アドレス範囲の区画ごとに分類し，アクセス頻度テーブルとして保存する．まずアクセスホットスポット解析を行いたいアドレス範囲全体を対象に，バス観測プローブからイベントデータ取得を続ける．デバッグプロセッサは随時イベントデータを読み出す．この時デバッグプロセッサは，イベントデータに含まれるアクセスのアドレス情報からアドレス範囲の区画のハッシュ値を計算し，該当するアクセス頻度テーブルの該当する区画の値を増加させる．この処理を一定期間続ける．

この方法を用いることで，アクセス頻度が高いアドレス領域の数値が大きく，頻度が低い領域の数値が小さいアクセス頻度テーブルが作成でき，アクセスのホットスポットを確認することができる．

また，同様の方式を用いて，単一コアのキャッシュミスが発生しやすい領域を検出するために使用することもできる．

5. まとめ

組み込み機器で問題が増大している観測が困難なバグに対処し，開発効率を向上させるため，高度な条件を用いてフレキシブルにマルチコアSoC各部を観測可能とするデバッグ支援ハードウェアを開発した．

デバッグ支援ハードウェアは，SoC各部の観測モジュールと，高度な観測機能や観測条件の柔軟な変更を実現するデバッグプロセッサから構成される．

これらの特徴を持つデバッグ支援ハードウェアを，NEC エレクトロニクスの携帯電話用 SoC である Medity M2 へ適用した．

回路規模が小さいデバッグプロセッサを専用設計し，内部メモリ容量の縮減も図った結果，回路規模は内部メモリを含めて 100Kgate となった．Medity M2 のロジック部の回路規模は 15Mgate であることから，デバッグ支援ハードウェアの内部メモリを含む回路規模はロジック部の 0.7%程度に相当する．結果的に，チップの 1%以下の回路規模で，既存手法では確認が困難であった不具合事象の観測性を向上可能であることが確認できた．

また，観測プローブ側でイベントデータを通知する状況を限定することで，データレートの低減による配線本数の縮減と配線遅延制約緩和が可能な構成をとった．その結果配置配線の問題は発生せず，提案したデバッグ支援ハードウェアの構成が実SoCへ十分適用可能であることも確認できた．

一方で，デバッグ支援ハードウェアの能力を生かすためにはデバッグプロセッサのプログラミングが必要である．このため容易にデバッグプロセッサを活用できるツール環境整備が必要であり，今後の課題である．

文　献

[1] MIPS Technologies, Inc., A Brief Summary of the Benefits of MIPS PDtrace™, http://www.fs2.com/summaryofmipsbenefits.html

[2] ARM ltd., Embedded Trace Macrocell http://www.arm.com/products/solutions/ETM.html

[3] Neal Stollon, “マルチコアのデバッグをいかにより良く行うか,” Japan Multicore Expo 2006, 一日目, pp.12-1-12-12, Oct.2006.

[4] ARM ltd., “CoreSight™ 技術,” http://www.jp.arm.com/products/soc_infraip/coresight.html

[5] Shuichi Kunie, Takefumi Hiraga, Tatsuya Tokue, Sunao Torii, Taku Ohsawa, “Low power architecture and design techniques for mobile handset LSI Medity™ M2,” Proc. The 13th Asia and South Pacific Design Automation Conference., no.8D-4, pp.748-753, Seoul, Korea, Jan.2008.