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

2004－SLDM－116 (20)
2004／10／22

SoCデバッグ容易化技術とその応用

兒玉 征之† 佐圓 真† 西本 順一‡ 荒川 文男†

†株式会社日立製作所 〒185-8601 東京都国分寺市東恋ヶ窪 1-280
‡株式会社ルネサステクノロジ 〒187-8588 東京都小平市上水本町 5-20-1

E-mail: †{t-kodama, m-saen, arakawa}@crl.hitachi.co.jp, ‡nishimoto.junichi@renesas.com

あらまし 大規模IPを多数搭載したSoCにおいては，CPUコア中心のデバッグ機能では，十分なデバッグ支援ができなくなりつつある．特に，情報系と制御系を融合したシステムでは，リアルタイム処理を継続しながら他の機能をデバッグできるようにすることが求められる．また，システム全体の動作を観測しながら，性能の最適化を可能にすることも求められる．本稿では，リアルタイム処理中のデバッグ機能をと，SoCを統合的に扱うための観測機能のモジュール化を，SuperHアーキテクチャのサンプルチップに適用して効果を確認した．

キーワード デバッグ，リアルタイム処理，性能最適化

SoC debug architecture and applications

Tomoyuki KODAMA† Makoto SAEN† Junichi NISHIMOTO‡ and Fumio ARAKAWA†

†Hitachi, Ltd., 1-280 Higashi-koigakubo, Kokubunji-shi, Tokyo, 185-8601 Japan
‡Renesas Technology Corp., 5-20-1 Jousuihoncho, Kodaira-shi, Tokyo, 187-8588 Japan

E-mail: †{t-kodama, m-saen, arakawa}@crl.hitachi.co.jp, ‡nishimoto.junichi@renesas.com

Abstract Debugging tools analyzing CPU core mainly are inadequacy for latest SoC, which have many IPs. Especially debugging tools with real time operations or system performance analysis techniques for optimization are indicated. Therefor we develop techniques to coexistence between debug and real time operations and to modularize circuits for performance analysis.

Keyword Debug, Real time operation, Performance optimization

1. はじめに

現在，既存の組み込み向けLSIでは，アクセラレータや各種インタフェースのような多くのIPを搭載するために，低電力化や微細化が進められている．このような大規模SoCは，システム開発者にとって，巨大なブラックボックスとなっている．その結果，システムの問題を解析し，最適化するために長い開発期間を必要とするようになっている．これは特にシステム最適化にタイミング制約を含む場合，より開発を困難にする傾向がある[1-4].

加えて，高度ナビゲーションやテレマティクスといった情報処理を実現するSoCでは，通信，位置検出，機器制御といったリアルタイム制御と情報処理の連携が必要となる．このような情報系と制御系を融合したシステムでは，リアルタイム処理を継続しながら他の機能をデバッグできるようにすることが求められる．

2. 大規模SoC搭載システムの開発支援

従来のデバッグツールは，SoCに占めるCPUコアの割合が高いため，ブレークやトレースといったデバッグ支援機能も，CPUコアが中心であった[5]．しかし，現在は搭載IPが増え，CPUコア中心のデバッグ支援機能では不十分となってきている．私たちは，リアルタイム処理とデバッグの共存および，多様なSoCを柔軟に解析可能とする技術，またその解析結果からシステム性能を改善する手法を開発し，SuperH^{TM} CPUコアを備えたSoCに適応した[6-8].

2.1. ブレークとリアルタイム処理の共存

ソフトウェアのデバッグでは，ブレークを発生させ，CPUの処理に対し一時停止を行う．このブレークは，割込などによる処理の切り替えも停止するため，デバ

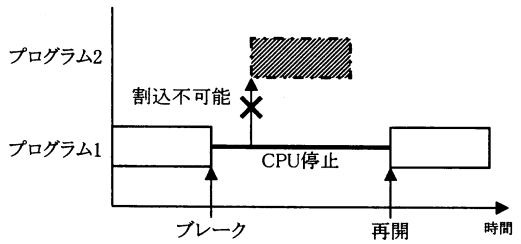
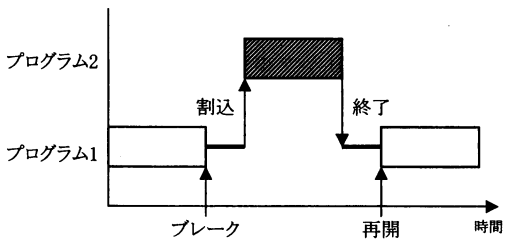


図1 想定しない実行中断

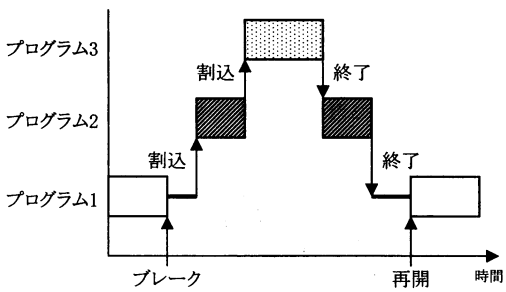
ッグ対象となっているプログラム(プログラム 1)だけでなく,IPからの割込処理といった他のプログラム(プログラム 2)に対して，想定しない実行中断を引き起こす(図 1). このため，通信や機器制御などのリアルタイム処理を伴うシステムでは，ブレークの使用により，システム全体が破綻し，デバッグ自体を継続できない場合がある.

例えば，プリンタ制御用 SoC には、焼損を防ぐために，プリンタヘッドの通電による温度上昇を検出し，一定温度以上になると割込を発生させ，通電を停止する機構がある．このプリンタの紙送り制御を実機デバッグしている際，割込を処理できないと，プリンタヘッドの温度制御ができず，焼損してしまう.

このため，CPU がブレークによる一時停止状態であっても，他のプログラムを実行し，再度一時停止状態へ遷移可能とする必要がある（図 2(a)). しかし，実行するプログラムはブレークによる一時停止状態を認識していないため，プログラム終了による一時停止状態への遷移はハードウェアで行う必要がある．更に，図 2(b)のように多重に例外又は割込が発生する場合，プログラム 3 は終了時にプログラム 2 を再開し，プログラム 2 は終了時に一時停止状態へ遷移する必要がある．このため，ハードウェアが一時停止状態の中断を記憶すると共に，各プログラム終了時に一時停止状態へ遷移すべきかどうかを判断することにより，一時停止状態からの多重の例外及び割込受付を可能とした．この結果，従来困難であったブレークとリアルタイム処理の共存を実現した.



(a) CPU停止状態からの再開と停止



(b) CPU停止状態からの多重割込

図 2 停止状態からの処理再開

2.2. SoC 内部の観測実現

大規模 SoC では，CPU コア以外に，要求仕様にあわせて様々なアクセラレータやバス，インタフェースが搭載される．これら IP の観測もデバッグ支援機能として必要である．また，SoC の構成にあわせて柔軟な構造をとることが可能にする必要がある.

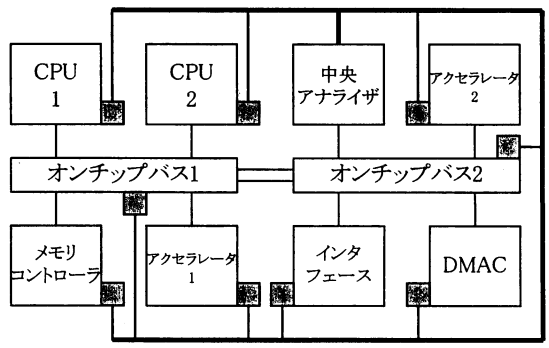
これに対し，μアナライザおよび，中央アナライザにより SoC 内部情報の観測を可能とする．μアナライザは，SoC の一部分を観測する機能を持ち，SoC の各所に挿入する．これらμアナライザは，挿入による面積増加を最小にするため，挿入箇所の観測機能と，中央アナライザとのデータ転送および制御用の接続機能のみを持つ．中央アナライザは，各μアナライザと接続することで，観測情報の収集と同期制御を行い，SoC 全体を観測することを可能とする(図 3).

2.3. システム性能最適化

システム開発では，個々のプログラムの動作の検証以外に，システム性能の最適化が行われる．このシステム性能の最適化において，意外な性能劣化が見られることがある．システムの性能低下の主な原因は，SoC 上で複数実行されているプロセスが特定のリソースに関して競合を起こすことにある．このリソース競合の多くは,(課題 1)各プロセスのリソース使用率の最適化と,(課題 2)リソース利用のタイミングの最適化により改善可能である.

これに対し，前述の SoC 内部観測を用いることで，一定期間内における各プロセスのリソース利用期間を容易に知ることができ，より少ない試行回数で最適化が可能となる．また，このリソース利用期間を得るために，周期的なトリガを生成する機能も搭載する．ここで得られた情報を元に，図 4 に示すようなグラフを作成できる．グラフには，各プロセスにおいて，リソースをどの程度使用するか，またリソースがどの程度使用されているかを示している.

システム性能最適化の例を図 4 に示す．この例では，



□ : μアナライザ

図 3 SoC 内部観測の実現

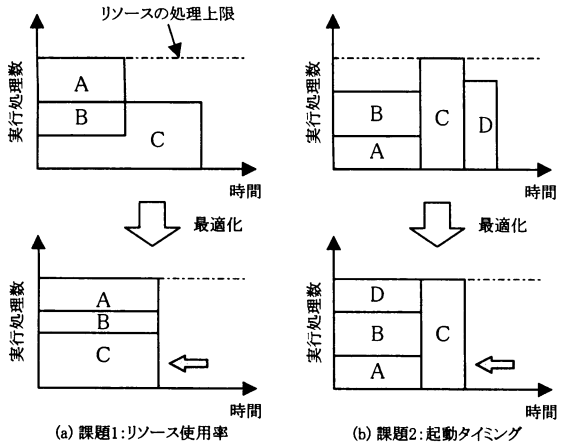


図 4 システム性能最適化

プロセス A，B，C，D が，処理能力にある上限を持ったリソースを使用して実行される．

図 4(a)に示すのは，(課題 1)の最適化例である．最適化前は，各プロセスによるリソース使用率が均一ではない．よって，各プロセスによるリソース使用率を調整することで，最適化できる．

図 4(b)に示すのは，(課題 2)の最適化例である．最適化前は，リソース使用のタイミングが非効率的であることを示す．ここでは，プロセス D の起動タイミングを変更することにより，最適化できる．

システム性能は，上記のような SoC 内部の観測により最適化が行える．

3. インプリメンテーション

3.1. CPU 状態遷移制御

CPU 状態遷移制御回路を図 5 に示す．CPU の状態遷移は，状態遷移制御回路で決定する．状態遷移制御回路は，ブレークを検出すると CPU を停止状態に遷移させる．この状態で，例外，割込を検出すると，CPU の処理を再開させると同時に，CPU 状態フラグにブレークによる停止状態から例外，割込により処理を再開していることを示すフラグをセットする．このフラグがセットされているとき，ハンドラの終了命令の実行完了と戻り先状態から，停止状態への遷移か，CPU の処理を継続するかを判定する．停止状態への遷移と判定した場合は，CPU を停止状態へ遷移させる．

3.2. アナライザの構成

μアナライザは図 6 に示すように，制御レジスタブロック，トリガ生成ブロック，動作制御ブロックの 3 つのブロックからなる．トリガ生成ブロックは，接続された対象を観測するタイミングを検知する．このトリガは，CPU に対するブレークと同じように，制御レ

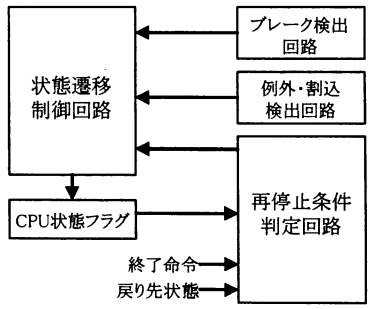


図 5 CPU 状態遷移制御回路のブロック図

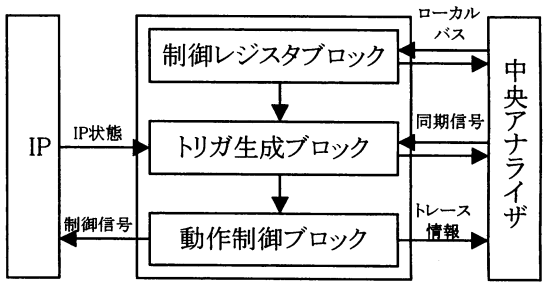


図 6 μアナライザのブロック図

ジスタの設定値と IP の状態を比較することにより生成する．動作制御ブロックは，このトリガを検知すると，システム性能最適化に必要なトレース情報を生成する．このμアナライザは，他のμアナライザからのトリガを自分のトリガ生成要因とする機能と，他のμアナライザに対して自分のトリガを同期信号として送信する機能を備えている．この機能により，複数の IP を同時に観測したり，連続的に観測することができる．

中央アナライザは，各μアナライザから送信する同期信号の配信先選択機能と，トレース情報のバッファ機能およびμアナライザと同様の機能を持つ．

また，各μアナライザの同期信号やトレース情報信号は，システムの同期制御やトレース情報収集のために中央アナライザに接続する．

3.3. SoC への適用方法

μアナライザの挿入箇所は，SoC 内部の観測性と，各 IP の特長により，次の 2 つのタイプに分類できる．1 つは IP の内部信号観測用の挿入タイプ，もう 1 つはインタフェース信号観測用の外付タイプである．

(1) 挿入タイプμアナライザ

このタイプは，汎用 CPU などの IP 内部で競合が発生する場合や，システム開発者がプログラムの改良によって競合を軽減できる場合に最適である．このような場合は，μアナライザは IP 内部に挿入するべきである．

(2) 外付タイプμアナライザ

多くのアクセラレータや，インタフェースはこのタイプのμアナライザを適用できる．IP の多くは，ハードマクロであったり，内部の解析工数が大きい場合があり，IP 内部へのμアナライザ挿入が困難な場合多い．このような場合は，IP が接続されるオンチップバスとの間に挿入することで，観測が可能となる．

4. 評価

4.1. システム性能最適化

SoC 観測によるシステム性能最適化を，5 つの処理からなるマルチメディア処理(カメラからの入力，ブレンド，MPEG エンコード，フォーマット変換，表示)のシミュレーションにより評価した．評価対象の SoC は，これら 5 つのマルチメディア処理用 IP と，IP を制御する CPU，これら IP と CPU 用のμアナライザと中央アナライザを搭載する．

評価用のマルチメディア処理は，最初に CPU から各 IP の設定を行った後，パイプライン処理される．このパイプライン処理中に，各 IP がメモリを使用する際，競合が発生し，システム性能を低下させる．

システム状態の計測は次のように行う．

- CPU 用μアナライザは，画像フレーム処理の開始と終了タイミングを検出し，他のμアナライザと中央アナライザに対して，同期信号を送信する．
- 各 IP 用μアナライザは，同期信号の周期にあわせ，各 IP が利用するメモリ帯域幅を計測し，中央アナライザへ送信する．
- 中央アナライザは，同期信号を使うことによって，周期あたりの各 IP の利用するメモリ帯域を出力する．

実際の最適化手順を図 7 に示す．ここで示す手順は，図 4(a)の例に相当し，処理時間を短縮することが，システム性能向上につながる．

図 7(a)は，最適化前のもっとも単純なバス調停を行った結果である．この設定では，最初にバスにリクエストを出す IP がアクセス権を得る．そのため，バスを効率的に利用することができず，未使用部分が発生してしまう．また，MPEG エンコードに非常に大きなバス帯域を使用していることがわかる．

各処理に要するバス帯域を計算し，バス調停の比率に反映させた結果を図 7(b)に示す．この最適化により，システム性能は 9%改善できた．しかし，依然メモリ帯域の未使用部分が残っていることがわかる．

さらにバス調停の比率を調整した結果を図 7(c)に示す．システムの性能は 17%改善され，十分に最適化できたことがわかる．

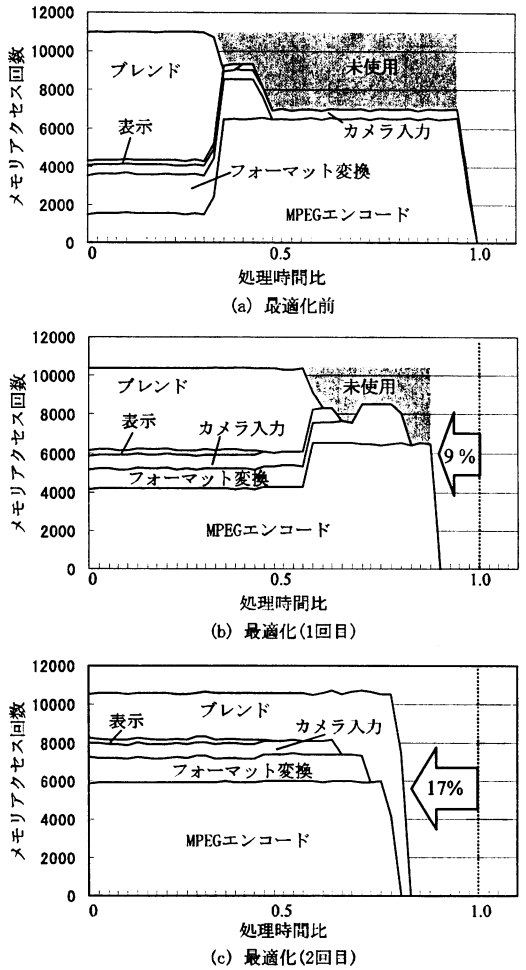


図 7 システム性能改善試行

このように，わずかな試行回数でシステム性能の最適化が可能となった．

4.2. チップ面積

チップ面積への影響は，サンプルチップへの実装により確認した(図 8)．評価に用いたチップは，CPU コアおよびいくつかのアクセラレータを搭載している．また，μアナライザは，CPU コアおよび，オンチップバスに適用した．CPU コア用μアナライザは挿入タイプ，オンチップバス用μアナライザは外付タイプである．

CPU 状態遷移回路と，オンチップバス用μアナライザ，中央アナライザによる面積増加は 1%程度であり，影響はほとんどない．

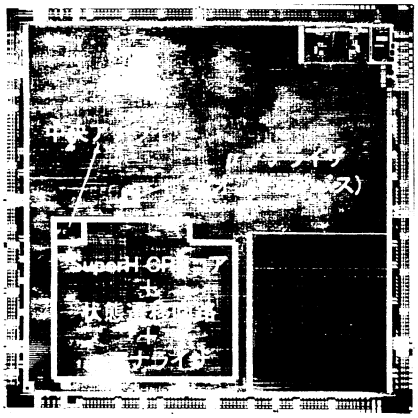


図 8 サンプルチップ写真

5. 結論

ブレークによる CPU の停止に対し，CPU の停止状態を記録，評価することで，CPU の処理再開，再停止状態への移行を可能とした．これにより，リアルタイム処理を行うシステムでも，システムの破綻を招くことなくブレークを行えるようになった．

また，SoC 全体を観測可能にすることで，システム性能最適化を最小の試行回数で実現可能とした．また，チップ面積への影響も非常に小さい．これにより，システム開発期間の短縮を実現した．

文　献

[1] D. Kirovski et al., "Improving the observability and controllability of datapaths for emulation-based debugging," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol.18(no.11), pp.1529-1541, Nov. 1999.

[2] L. Benini et al., "Virtual in-circuit emulation for timing accurate system prototyping," ASIC/SOC Conference, 2002. 15th Annual IEEE International, pp.49-53, New York, USA, Sep. 2002.

[3] Y.-T. S. Li et al., "Performance analysis of embedded software using implicit path enumeration," IEEE Trans. Computer-Aided Design, vol.16(no.12), pp.456-461, Dec. 1997.

[4] C. Brandolese et al., "Source-level execution time estimation of C programs," in Proc. International Workshop Hardware/Software Codesign, pp.98-103, Colorado, USA, Apr. 2001.

[5] P. Ching et al., "An in-circuit emulator for TMS320C25," IEEE Trans. Education, vol.37(no.1), pp.51-56, Feb. 1994.

[6] F. Arakawa et al., "An embedded processor core for consumer appliances with 2.8GFLOPS and 36M polygon/s FPU," ISSCC Dig. Tech. Papers, Feb. 2004.

[7] T. Kamei et al., "A resume-standby application processor for 3G cellular phones," ISSCC Dig. Tech. Papers, Feb. 2004.

[8] M. Saen et al., "Transparent SOC: On-chip Analyzing Techniques and Implementation for Embedded Processor , " IEEE International SOC Conference, pp.51-54, Sep. 2004.