NEWS 2011

LAUTERBACH
DEVELOPMENT TOOLS

DEBUGGER, REAL-TIME TRACE, LOGIC ANALYZER

常に先を行く製品開発

このスローガンの下、ローターバッハは過去30年以上にわたって、組み込み業界向けのツールを開発してきました。とりわけ新しいデバッグ技術の分野において、ローターバッハは世界的なリーダーであり、その動向を左右する存在です。

こうした取り組みの成果により、ローターバッハは、大手半導体メーカー各社から高い評価を得るに至りました。長年にわたって、新技術の開発・実装に携わるメーカー各社は、積極的にローターバッハと共同作業を進めてきました。この共同作業から画期的なアイデアが生まれ、最先端の製品に生かされています。

加えて、ローターバッハは、お客様の声に真摯に耳を傾けることを旨としています。当社のTRACE32をご使用のお客様からいただいた要望や提案は、貴重な声として、ローターバッハの製品開発に寄与しています。多くの事例で、お客様からの提案は即座に実現され、デバッガの次回リリースバージョンに反映されています。

このような先駆者の視点から、ローターバッハが現在注目しているトレンドと、今後普及が進むと予想される技術について紹介します。

Androidのデバッグ

Androidのデバックは、特に重要なトピックです。携帯電話向けアプリケーションでは、特定のアーキテクチャに依存しない、仮想マシン（VM）を想定したコードによる記述が増えています。GoogleのAndroidとそのDalvik　VMが広く普及した結果、アプリケーション、仮想マシン、オペレーティングシステム、および基盤となるハードウェアの相互作用によってのみ発生する複雑なエラーがデバッグの対象になっています。このような複雑なエラーをデバッグするには、Javaアプリケーションから Linuxのハードウェアドライバに至るまで、ソフトウェアレイヤー全体に透過性が求められます。

いくつかの携帯電話メーカーからの依頼を受けて、ローターバッハは2010年半ばに、VMデバッグ対応APIの開発に着手しました。この開発では、Androidをリファレンスプラットフォームとして使用しています。その目的は、オープンソース型およびクローズドソース型のVMのプロバ »

コンテンツ

イダ各社がTRACE32によるデバッグに製品を適合させられるよう、オープンなインタフェースを提供することです。VMデバッグ対応と開発の現状の詳細については、6ページの記事「VMデバッグ対応API」を参照してください。

エネルギープロファイリング

地球温暖化対策や「環境に優しい」電子システムの必要性が強調される中で、組み込みシステムのエネルギー消費量の測定に関心が寄せられています。最近の技術雑誌には必ずと言ってよいほど、バッテリー駆動の機器や低消費電力のマイクロコントローラに関する様々な記事が掲載されています。この分野の新技術に対して、技術革新賞が贈られるケースが増えています。

ただし、携帯電話市場では、待ち受け時間と使用可能時間が常に重要なトピックになっています。携帯電話には、長年にわたって、様々なエネルギー削減対策が実装されてきましたが、このような対策が有効に機能するのは、組み込みシステムを制御するソフトウェアで一貫して、ハードウェアのエネルギー節約機能全般が使用される場合に限定されています。

2006年の初頭から、ローターバッハのツールは、組み込みシステムにおけるソフトウェアと電力消費の相互作用をシンプルに比較および解析できる測定装置をサポートしてきました。同様の技術は、2010年半ばから、TRACE32 CombiProbeでも使用できるようになりました。「CombiProbeによるエネルギープロファイリング」の詳細については、11ページを参照してください。

マルチコアデバッグ

マルチコアチップが組み込みシステムに採用されるようになってから10年が経過しました。ローターバッハも2001年からマルチコアシステム向けのデバッガを投入してきましたが、マルチコアのデバッグは現在でも重要なトピックです。内部システムの動作をさらに可視化したいという要望を受けて、チップのデバッグインフラストラクチャに新しいトレースセルを統合する取り組みが進められています。

元々、トレース情報は個別のコアに対してのみ生成されていましたが、現在では、様々なトレースソースが用意されています。

a)チップ-内部バスの転送を可視化するトレースソース:

- AMBA AHB Trace Macrocell(HTM)を実装したARM CoreSight
- Infineon社から提供されるTriCore用のSystem Peripheral Bus(SPB)とLocal Memory Bus(LMB)を実装したMCDS
- Texas Instruments社から提供されるチップ用のRAM Trace Port
- NEXUS Power Architecture用のDMAおよびFlexRayトレース

b)特殊なインタラプトトレースなど、チップ内部IP(Intellectual Property)のトレース情報を生成するトレースソース

c)ソフトウェア生成されたトレース情報の出力を可能にするトレースソース:

- ARM CoreSight用のInstrumentation Trace Macrocell(ITM)
- ARM CoreSight用のSystem Trace Macrocell(STM)

TRACE32デバッガの開発は継続しており、新しいトレースソースへの対応が進められると同時に、設定の簡素化や提供情報の広範な解析が実現しています。

シリアルトレースポート

内部チッププロセスの可視化によって提供されるトレースデータの量が増加したことに伴い、複雑なマルチコアチップや高性能なプロセッサに対応する広い帯域幅に加えて、高速なトレースポートも求められています。

このようなニーズを受けて、チップメーカーはここ数年の重要な技術革新の成果として、シリアルトレースポートを開発しました。ハードディスクメーカーは長年、PCの高速データ交換用にシリアルインタフェースを採用してきましたが、2008年に初めて、この技術を流用して、ARM社のHigh Speed Serial Trace Port(HSSTP)経由でトレース情報をエクスポートしました。同時期に、ローターバッハはシリアルトレースポートに対応するトレースツールの提供を開始しました。

その間、シリアルトレースインタフェースを備えたプロセッサファミリーが他にも登場しています。この領域における開発の現状については、9ページの記事「シリアルトレースポートの用途拡大」を参照してください。

トレースメモリの容量拡大

高速なトレースインタフェースはデータ転送速度も高速化した結果、さらに大容量のトレースメモリが必要になっています。大容量のトレースメモリがなければ、トラブルシューティング用に十分な大きさのプログラムセクションを »

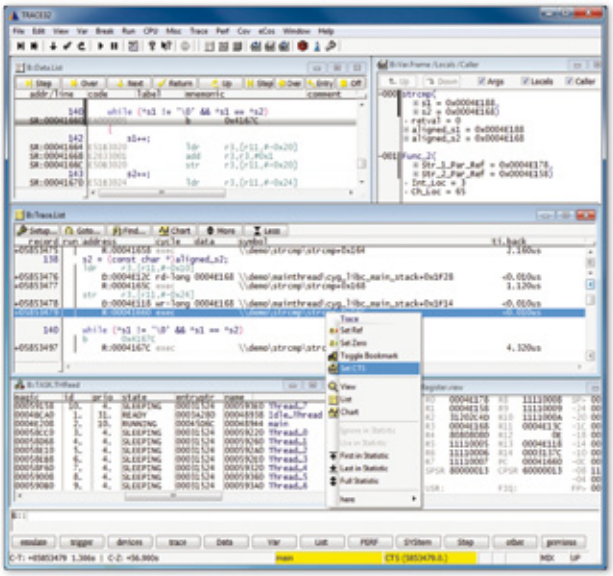

図1:　トレースベースのデバッグを高速に実行するには4GBものトレースメモリ解析が必要

取得して、組み込みシステムの時間的な挙動を解析することはできません。
ただし、トレースメモリの容量拡大に意味があるのは、トレース情報の高速処理に必要なインフラストラクチャを使用できる場合に限定されます。この点は特に、トレースベースのデバッグなど、要件の厳しいトレース解析機能に当てはまります（図1を参照）。SDRAMチップの容量増加、高速なPC、GB　Ethernetインタフェースが実現したことにより、ローターバッハは2007年に、4GBのメモリを搭載したトレースツール、PowerTrace　IIの提供を開始しました。

2008年半ばに、ローターバッハはトレース記録および解析の新しい手法である、Real-time Streamingの開発に着手しました。この開発は、長時間のコードカバレッジ解析、広範なシステム実行時解析、散発的なエラーを特定するためのトレース記録時間の延長といった、お客様のニーズを受けて進められました。

Real-time　Streamingの新機能としては、記録中のトレースデータがホストに転送されるという点が挙げられます。ホストで受信されたトレース情報は、そのまま解析されます。オプションとして、解析中のトレース情報をハードディスクに保存することも可能です。

Real-time　Streamingが機能するのは、トレースデータの処理ステップがすべて最適な速度で実行される場合に限定されます。この点は、転送と解析に加えて、ハードディスクに保存されているファイル内のトレース情報をシステム的に検索する場合にも当てはまります。

従来型のトレースでも、新しい速度最適化機能の多くは利用できます。たとえば、トレースデータ圧縮機能（Real-time　Streaming用に開発）を従来型のトレース用にも実装する計画があります。トレースデータ圧縮機能の詳細については、10ページを参照してください。

トレースベースのデバッグ

トレースベースのデバッグ（別名、Context Tracking System（CTS））では、トレース対象のプログラムセクションの再デバッグが可能になります。具体的な方法として、TRACE32は、そのPowerView　GUIで各トレースレコードのターゲットシステムの状態を再構成できます。この再構成の対象としては、レジスタやメモリの内容、変数の状態、ソースおよびタスクのリスト、スタックフレームなどが挙げられます。

トレースベースのデバッグの開始点を選択した後に、すべてのデバッグコマンドが使用可能になります。各コマンドは、再構成されたトレース記録に基づいて、TRACE32によって実行されます。ステップを戻す操作や、関数の開始時点に戻る操作も実行できるので、トレースベースのデバッグ機能を利用している多くのユーザーに役立つと考えられます。

トレースベースのデバッグでは、他にも有用な一連の機能が用意されています。

- 全ローカル変数を含む、高水準言語でのトレース表示
- 実行時解析と関数呼び出しツリー
- トレースギャップの再構成（トレースポートから出力可能な量以上のトレースデータが生成された場合の対応）

www.lauterbach.com/cts.html

展望

現在のトレンドに加えて、デバッグ技術の領域では、様々な新しい技術の開発が進められています。2011年のニュースレターに一通り目を通していただければ、お客様のプロジェクトに役立つ新しい発見があるのではないかと考えます。5月2～5日にサンノゼで開催されるESC　Silicon Valleyや、年間を通して米国内で開催されるその他の展示会でも、当社は新しい技術の成果をいくつか披露させていただく予定です。

新サポートプロセッサ

New derivatives	
Actel	LA-7844 (Cortex-M) • A2F060, A2F200, A2F500
AppliedMicro	LA-7723 (PPC400) • APM80186, APM821x1 • APM86290 LA-7752 (PPC44x) • PPC460SX
ARM	LA-7843 (Cortex-A/R) • Cortex-A15 • Cortex-A15 MPCore LA-7844 (Cortex-M) • Cortex-M4 • SC000, SC300
Atmel	LA-7844 (Cortex-M) • AT91SAM3S, AT91SAM3N LA-3779 (AVR32) • AT32UC3A / B / C / D / L
Broadcom	LA-7760 (MIPS32) • BCM3549 / 35230 / 4748 • BCM5354 / 5358 / 5331X • BCM6816 / 6328 / 6369 • BCM7407 / 7413 / 7420
Cavium	LA-7761 (MIPS64) • CN63XX
Ceva	LA-3711 (CEVA-X) • CEVA-X1643, CEVA-XC
Cortus	LA-3778 (APS) • APS3 / B / BS / S
Cypress	LA-7844 (Cortex-M) • PSoC5
Faraday	LA-7742 (ARM9) • FA726TE
Freescale	LA-7736 (MCS12X) • MCS9S12GC / GN / Q LA-7732 (ColdFire) • MCF5301x, MCF5441x LA-7845 (StarCore) • MSC8156 LA-7742 (ARM9) • i.MX28 LA-7843 (Cortex-A/R) • i.MX53 LA-7844 (Cortex-M) • Kinetis
Freescale (Cont.)	LA-7753 (MPC55xx/56xx) • MPC5602D / P • MPC564XA / B / C / S • MPC567XF / R LA-7729 (PowerQUICC II) • MPC830X LA-7764 (PowerQUICC III) • P10xx, P20xx, P40xx • P3041 (2H/2011) • P5010, P5020 (2H/2011)
Fujitsu	LA-7844 (Cortex-M) • FM3
Infineon	LA-7756 (TriCore) • TC1182, TC1184 • TC1782, TC1782ED • TC1784, TC1784ED • TC1791, TC1791ED • TC1793, TC1793ED • TC1798, TC1798ED LA-7759 (XC2000/C166S V2) • XC22xxH / I / L / U • XC23xxC / D / E / S • XC27x2 / x3 / x7 / x8 • XE16xFH / FU / FL
Intel®	LA-3776 (Atom™/x86) • E6xx, Z6xx, N470 • Core i3 / i5 / i7, Core2 Duo
Lantiq	LA-7760 (MIPS32) • XWAY xRX200
LSI	LA-7765 (ARM11) • StarPro2612, StarPro2716 LA-7845 (StarCore) • StarPro2612, StarPro2716
Marvell	LA-7742 (ARM9) • 88F6282, 88F6283, 88F6321 • 88F6322, 88F6323 LA-7765 (ARM11) • 88AP510-V6 LA-7843 (Cortex-A/R) • 88AP510-V7
MIPS	LA-7760 (MIPS32) • MIPS M14K, MIPS M14KC
Netlogic	LA-7761 (MIPS64) • XLR, XLS
NXP	LA-7844 (Cortex-M) • LPC11xx • EM773

New derivatives	
Ralink	LA-7760 (MIPS32) • RT3052, RT3662
Renesas	LA-3777 (78K0R/RL78) • 78K0R/Hx3/Lx3/Ix3 • 78F804x, 78F805x • RL78/G12, RL78/G13 LA-3786 (RX) • RX610/6108/621/62N/630
STMicro-electronics	LA-7753 (MPC55xx/56xx) • SPC560D/P, SPC56APxx • SPC564Axx, SPC56ELxx LA-7844 (Cortex-M) • STM32F100, STM32L15x
ST-Ericsson	LA-7843 (Cortex-A/R) • DB5500, DB8500
Tensilica	LA-3760 (Xtensa) • LX3

Texas Instruments	LA-3713 (MSP430) • MSP430xG461x • MSP430x20x1/x2/x3 LA-7742 (ARM9) • AM1707/1808/1810 LA-7843 (Cortex-A/R) • OMAP36xx LA-7838 (TMS320C6x00) • OMAP36xx
Toshiba	LA-7742 (ARM9) • TMPA900, TMPA910 LA-7844 (Cortex-M) • TMPM330, TMPM370
Trident	LA-7760 (MIPS32) • HiDTV PRO-QX
Wintegra	LA-7760 (MIPS32) • WinPath3, WinPath3-SL
Zoran	LA-7760 (MIPS32) • COACH 12

Fast Modelsの仮想ターゲットに対応するトレース

ローターバッハは、2010年11月からARM Fast Modelsのトレースをサポートしています。

最初のハードウェアプロトタイプを待ってからソフトウェア開発に着手するという事態を回避するため、多くの事例で、ハードウェアをソフトウェア的に再現したモデルが使用されています。ARM社は、ARMベースの設計に対応するプログラミングモデルのソフトウェアパッケージとして、Fast Modelsを顧客に提供しています。

2008年以降、ローターバッハはCADIインタフェース上のFast Modelsのデバッグをサポートしてきました。そして現在、バージョン5.1のFast Models向けに導入されたModel Trace Interfaceをサポートするに至りました。トレース情報を適切に準備し、仮想ターゲットのバッファに格納する目的で、デバッガメーカーは個別のトレースプラグインをロードできます。図2に、TRACE32およびFast Modelsの相互作用の概要を示します。仮想ターゲットのデバッグの詳細については、次のURLを参照してください。

www.lauterbach.com/frontend.html

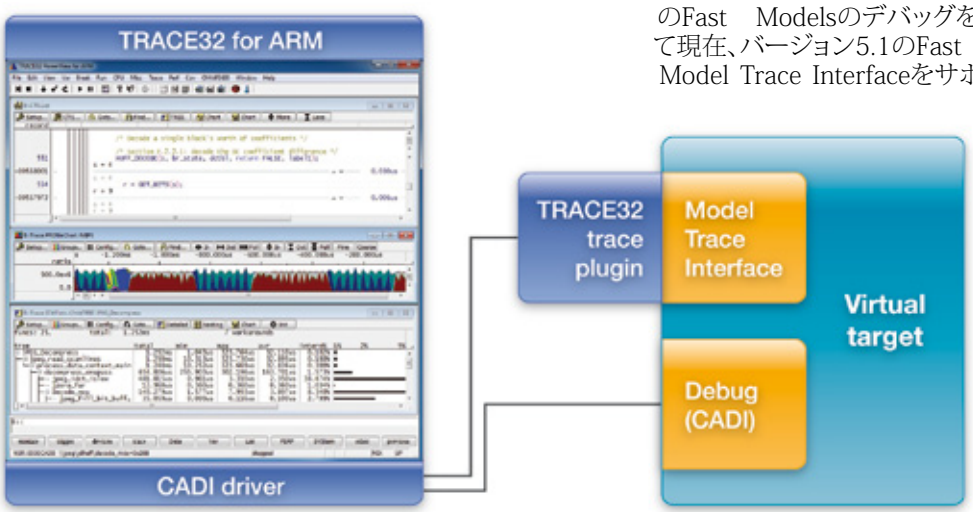

図2: TRACE32による仮想ターゲットのデバッグおよびトレース

VMデバッグ対応API

2006年以降、ローターバッハはJ2ME CLDC、J2ME CDC、KaffeといったJava仮想マシン向けJavaアプリケーションのデバッグをサポートしてきました。仮想マシンの普及に伴い、そのプロバイダの数は増え続けています。現在提供されている仮想マシンの中には、オープンソース型以外のものもあります。VMプロバイダ各社とその顧客による柔軟なVMのデバッグを可能にするため、ローターバッハは2010年半ばからソリューションに取り組んでいます。

ARMコア向けに実装されたAndroid Dalvik Virtual Machineは、停止モードデバッグに対応するVM APIの開発時のリファレンスモデルとして採用されています。

2種類のデバッグ環境

開発者の視点では、Androidは、次のコンポーネントで構成されるオープンソース型のソフトウェアスタックです（図3を参照）。

- ハードウェアドライバを備えたLinuxカーネル
- Dalvik Virtual Machineと一連のライブラリ（従来型のJavaコアライブラリ、Android固有のライブラリ、C/C++のライブラリなど）を備えたAndroid Runtime
- Javaとそのサポート対象のApplication Frameworkでプログラムされたアプリケーション

Android向けのソフトウェアは、様々な言語で記述されています。

- Linuxカーネル、一部のライブラリ、およびDalvik Virtual Machineのコードは、C、C++、またはアセンブラで記述されています。

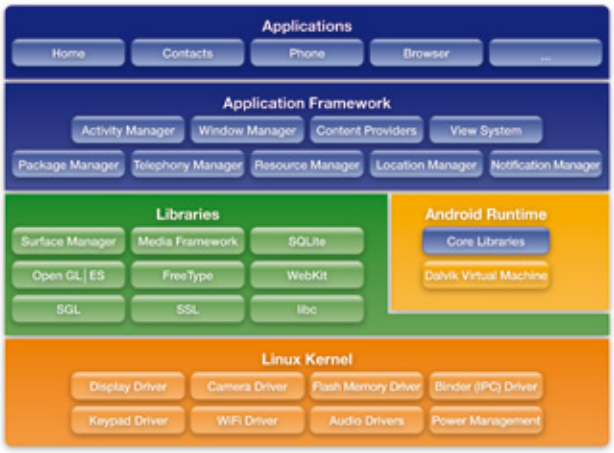

図3: オープンソース型のAndroidソフトウェアスタック

- VMアプリケーションとそのサポート対象のアプリケーションフレームワークは、Javaでプログラムされています。

コードの各ブロックのテストは、専用の独立したデバッグ環境で行われます。

C/C++およびアセンブラコードのデバッグ

C/C++およびアセンブラでコードが記述されたAndroidのパーツは、停止モードのJTAGインタフェースを介してターゲットハードウェア上でデバッグできます。停止モードデバッグでは、TRACE32デバッガはAndroidハードウェアプラットフォームのプロセッサと直接通信します（図4を参照）。

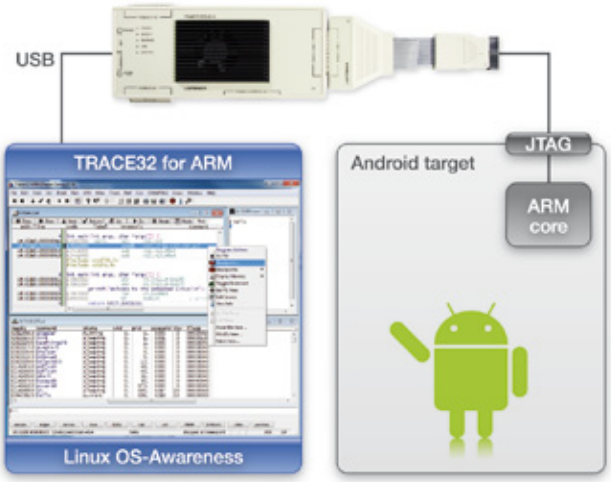

図4: 停止モードデバッグでは、デバッガはAndroidハードウェアプラットフォームのプロセッサと直接通信する

停止モードデバッグの特徴としては、デバッグ目的でプロセッサを停止したときに、Androidシステム全体が停止することが挙げられます。

停止モードデバッグには、次に示すように大きな利点があります。

- 必要になるのは、デバッガとプロセッサ間の機能しているJTAG通信のみ。
- ターゲット側にデバッグサーバーを必要としないので、テストリリース版のソフトウェアに適している。
- リアルタイム条件でテストを実行できるので、特定の条件でのみ発生する問題に対しても、効率的なトラブルシューティングが可能になる。

現在のところ、停止モードデバッグでは、Dalvik VM上のアプリケーションなど、VMアプリケーションのデバッグをサポートしていません。したがって、ソフトウェアレイヤー全体を対象とする透過的なデバッグはまだ実現していません。

Javaコードのデバッグ

Android向けのJavaコードは通常、Eclipseに統合されたAndroid Development Tools(ADT)を使用してテストします。ホスト側のAndroid Debug Bridge(adb)サーバーは、USBまたはEthernetを介して、ターゲット側のadbデーモンと通信します(図5)。

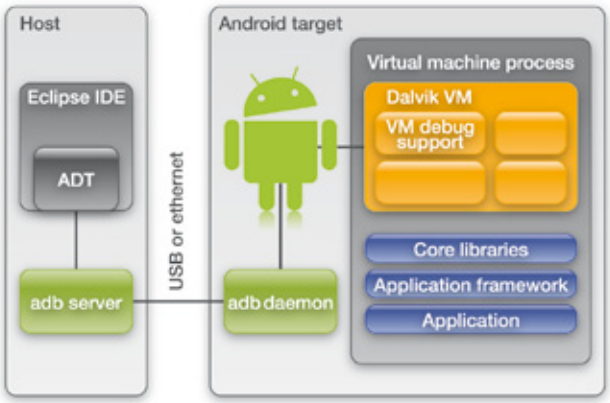

図5: Javaコードのデバッグに使用する、Eclipseに統合されたAndroid Development Tools(ADT)

ADTを使用してデバッグを実施する場合には、VMアプリケーションがデバッグ用に特殊な方法でコンパイルされており、Androidのデバッグサポート機能(adbデーモン)がハードウェアプラットフォーム上で動作していることが前提条件になります。

ADTを使用すると、Javaコードをスムーズにデバッグできます。ただし、次に示すように、ADTが有効に機能しない事例もいくつかあります。

- リリースコードで最初に発生したエラー
- Javaアプリケーションと、C/C++またはLinuxハードウェアドライバで提供されるサービスとの対話処理が行われたときに最初に発生したエラー
- デバッグ中に、adbサーバーとadbデーモンの間の通信が途切れた場合

VM対応の停止モードデバッグ

JavaアプリケーションからLinuxハードウェアドライバに至るまで、Androidシステムのテストをリアルタイム条件で実行できるようにするため、ローターバッハは現在、VMデバッグ対応機能を停止モードデバッグ機能に追加しています。

JTAGデバッガは、Androidハードウェアプラットフォームのプロセッサと直接通信します。したがって、プロセッサの停止後、JTAGデバッガはあらゆるシステム情報にアクセスできることになります。JTAGデバッガに採用されている高度な技術により、適切な情報を検索して、ビット単位やバイト単位の情報から抽象化した内容をユーザーが理解できる形式に変換することが可能です。

TRACE32ユーザーの利用できる抽象化レベルとしては、複数の仮想アドレス空間上でのオペレーティングシステムソフトウェアのデバッグに対応するオプションが挙げられます。別の抽象化レベルとしては、これまでのオペレーティングシステムのデバッグとは独立した、Javaのデバッグが挙げられます。

VM自体がオペレーティングシステムプロセス内でインスタンス化されている、AndroidのようなシステムのVM上で動作しているアプリケーションをデバッグするには、オペレーティングシステムのデバッグとJavaのデバッグを連携させる必要があります。この複雑な構成を実装するため、ローターバッハは拡張性に配慮した新しいオープンソリューションを開発しています。

オープンソリューション

今後、ローターバッハが提供する停止モードデバッグ機能では、次の抽象化レベルをサポートする予定です。

- 高水準言語のデバッグ
- ターゲットOS対応デバッグ
- VM対応デバッグ

高級言語のデバッグ機能はTRACE32ソフトウェアの固定コンポーネントであり、シンボルおよびデバッグ情報がロードされたプログラム向けに構成されています。 »

Dalvik Virtual Machine

Dalvikは、Androidで使用される仮想マシンの名称です。Dalvik Virtual Machineは、Javaによって生成されたバイトコードを実行するプロセッサのソフトウェアモデルです。仮想マシンを使用することで、プロセッサに依存しないソフトウェアの記述が可能になります。新しいハードウェアプラットフォームに移行する場合にも、仮想マシンを移植するだけで済みます。VMに対してコンパイルされたソフトウェアは、そのVMの移植先のプラットフォームでもそのまま動作できます。

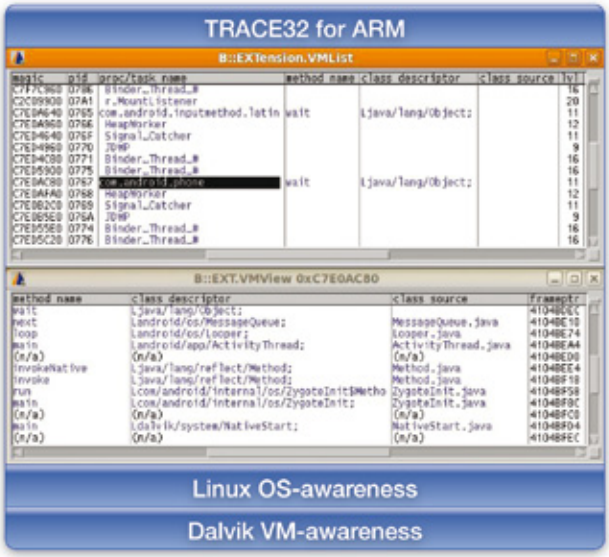

図6: リファレンス実装では、Linux OS対応とDalvik VM対応機能をTRACE32にロードする必要がある.

ターゲットOSデバッグ対応機能は常に、TRACE32ユーザー側で設定する必要があります。一般的なオペレーティングシステムを網羅した設定例が個別に用意されています。RTOS APIには、専用オペレーティングシステム向けにカスタマイズされたオプションが用意されています。

VMデバッグ対応機能は、J2ME CLDC、J2ME CDC、Kaffe向けのTRACE32ソフトウェアの固定コンポーネントです。その他の仮想マシンについては、個別にVM APIに適合させる必要があります。一般的なAndroid Dalvik VM向けに、すぐに使用できる設定が用意されています。

オペレーティングシステムと仮想マシンの両方について、オープンソリューションを採用することで、クローズドソース型のVMのプロバイダ各社は自社製品向けにTRACE32 VM対応機能を設定して顧客に提供できるようになります。

リファレンス実装

JavaアプリケーションからLinuxハードウェアドライバに至るまで、ARMベースのAndroidターゲット全体のデバッグを可能にするため、TRACE32は次の機能拡張を必要としています(図6を参照)。

- 1998年からローターバッハが提供しているLinux OS対応機能。
- 弊社ホームページからダウンロード可能なDalvik VM対応機能。本機能は、実プラットフォームに応じて設定する必要があります。

www.lauterbach.com/vmandroid.html

この成果として、現在実行中のJavaアプリケーションを識別してリスト表示(図6のEXTension.VMList)することが可能になっています。また、選択したJavaアプリケーションのVMスタックを解析して確認(図6のEXTension.VMView)することも可能になっています。

計画されている次のステップでは、VMで現在実行されているソースコードを表示します。この開発の目的は、最新のデバッガ機能全般による、VMアプリケーション向けの停止モードデバッグへの対応です。

新サポートRTOS	
DSP/BIOS for ARM	Q2/2011
OSEK/ORTI SMP	Q2/2011
T-Kernel for ARM	available
Windows Embedded Compact 7 for ARM	available
µC/OS-III for ARM	available

機能拡張と新RTOSバージョン

- TRACE32スクリプトがTimesys組み込みLinuxに対応しました。
- OSEK/ORTIにより、NEXUS所有権トレースメッセージがタスクの変化に応じて生成されます。これにより、NEXUSでデータトレースメッセージが生成されない場合でも、TRACE32でMPC55xx/MPC56xxを対象とするタスク対応の実行時測定が可能になります。

次のバージョンへの対応が実施および計画されています。

- OSEck 4.0
- QNX 6.5.0
- Symbian^3 for ARM
- Symbian^4 planned for Q1/2011
- Windows CE6 for Atom™

シリアルトレースポートの用途拡大

「より速く、より高く、より強く」という標語はスポーツの分野だけでなく、マイクロエレクトロニクスの分野でも基本方針になっています。クロック速度の高速化とプロセスステップの並列化がさらに進められたことで、数十年にわたって、驚くほどコンスタントに処理速度の向上が実現しています。トレース情報の伝送についても同様に、設計者はこの標語に従って取り組んでいます。

トレースインタフェースは、プロセッサがその内部プロセスの詳細なオペレーション情報を提供するためのインタフェースであり、情報量の爆発的な増加への対策が施されてきました。組み込みシステムの開発者の視点では、この重要な情報なしに開発を進めることは全く困難であり、あらゆる面でトレースインタフェースのデータスループットを向上させる取り組みが実施されてきました。クロック周波数の増加と、トレースポートのバス幅の拡張は、長年にわたって、データ量の増加に対応する効率的な方法でした。

ただし、このような方法は、代償を伴います。トレースポートの拡張によって貴重なパッケージピンが占有されるだけでなく、クロック周波数が増加して信号品質が低下することによってトレースバスからの全信号に対して補正処理が必要になります。Auto-Focus技術の高度なアルゴリズムを活用することで、ローターバッハは高周波数のトレース信号を誤差なく記録できる方法を実現しています。

プロセッサのアーキテクチャで並列化による処理速度の向上と構成の複雑化が進むことによって、トレースインタフェースでも、他の領域で採用実績がある高速なデータ転送方式の採用が始まっています。高速シリアル転送方式は、SATA、Fibre Channel、PCI Express、およびUSB3.0（SuperSpeed USB）で採用されています。各方式のデータの転送速度は非常に高速であり、差動データ用の信号線がいくつか必要になるという欠点が気にならないほどと言えます。

チップ上に高速シリアルインタフェースを統合する場合はコストがかさみ、最初のうちは統合によって問題が発生する可能性があります。一例として、I/Oパッドの処理速度を高速化する必要があります。ただし、ギガヘルツ範囲のシリアルインタフェースの実装事例が増えたことで、蓄積された知識・情報に基づいて、シリアルトレースポートに伴う問題の多くは解決できるようになりました。

2008年に、ARMはこの技術をHigh Speed Serial Trace Port（HSSTP）によって実装しました。この実装はほどなく、AMCC社のTitan、Freescale社のQorIQプロセッサP4040およびP4080、Marvell社のSETM3といった各社のプロセッサでも採用されるようになりました。

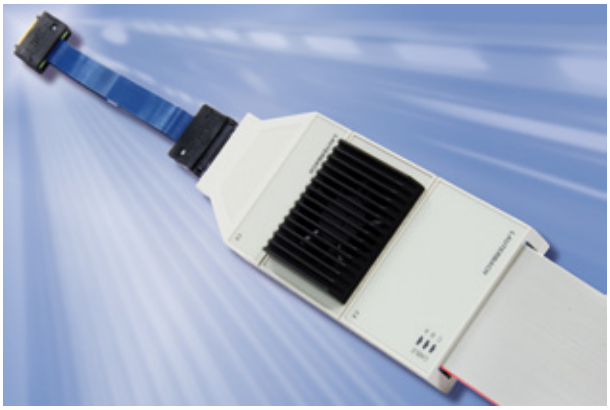

図7: ファームウェアとソフトウェアの対応に続キ、汎用ハードウェアではシリアルトレースインタフェースの様々なプロトコルをサポート

ローターバッハは2008年に、シリアルトレース用のハードウェアインタフェースを設計しました。Auroraプロトコルに基づいて汎用プリプロセッサが開発されました。他のプロトコルを記録する場合には、ファームウェアとソフトウェアの変更のみで対応できます。つまり、シリアルトレースプロトコルのバリエーションが今後増えた場合でも、当社のシステムは、すでに対応の準備が整っているということです。.

サポートされているシリアルトレースポート		
AMCC	APM83290 Program flow	2009
ARM-HSSTP	ETMv3, PTM, CoreSight ETMv3, CoreSight PTM Program flow, Data flow and Context-ID	2008
Freescale	NEXUS QorIQ P4040 and P4080 Branch Trace and Owner-ship Trace Messages, Data Write Messages	2010
Marvell-SETM3	CoreSight ETMv3 Program flow, Data flow and Context-ID	2009

Real-time Streamingの転送速度の高速化

「Real-time Streaming」は、記録中のトレースデータをホストに転送して、即座に解析することを意味します。この処理に伴い、特にCPUに負荷がかかるアプリケーションや、マルチコアシステムでは、トレースツールからホストに大量のデータを転送する必要があります。TRACE32では、このようなアプリケーションシナリオに対応するため、トレースデータをホストに転送する前に、トレースツール（PowerTrace II）で圧縮します。この機能は、2010年12月から、TRACE32ソフトウェアによってサポートされています。

Real-time Streamingは現在、ARM向けトレースプロトコルのETMv3およびPTMで実装されています。

ハードウェア圧縮

ホストへの最大転送速度は依然として、Real-time Streamingのボトルネックになっています。トレースツールとホストをピアツーピアGB Ethernetインタフェースで接続しても、有効な転送速度は500MBit/s程度に過ぎません。トレースポートの全データをロスなくホストに転送できるだけの最大転送速度を実現する必要があります。

転送対象の実際のデータ量を試算するには、Real-time Streamingの諸条件を把握しておく必要があります。

1. Real-time Streamingの主な用途としては、コードカバレッジと実行時測定が挙げられます。この両方の機能は、エクスポートされた情報がプログラムトレース情報だけの場合でも十分に対応できます。高精度な実行時測定値を得る目的で、サイクル精度のトレース機能を有効にできます。

2. トレースポートの平均負荷を考慮するだけで、実際に必要なデータ速度を試算できます。トレースポートのピーク負荷はPowerTrace IIによって捕捉され、大規模なFIFO（最大4GB）として認識されます。図8に、Cortexコアのトレースポートにおける平均/最大負荷の概要を示します。実際の負荷は、Cortexコア上で動作するアプリケーションによって決まります。

FPGAベースのハードウェア圧縮機能をPowerTrace IIに実装することで、ホストへの転送速度は3.2GBit/sに向上しました。

純粋な長時間トレース

ローターバッハは、Real-time Streaming処理中に、トレースデータが解析されると同時にハードディスクに保存される状況を長時間トレースとして認識しています。

Nexusなどの他のトレースプロトコルに長時間トレース機能を提供するため、ローターバッハは現在、同時解析を伴わない、ハードディスクへの純粋なストリーミング記録機能を提供しています。これにより、64ビット版のホストオペレーティングシステムで、最大1テラフレームのトレースの記録が可能になります。

Real-time Streamingと長時間トレースの詳細については、Lauterbachのホームページ（下記URL）にアクセスしてください。
www.lauterbach.com/tracesinks.html

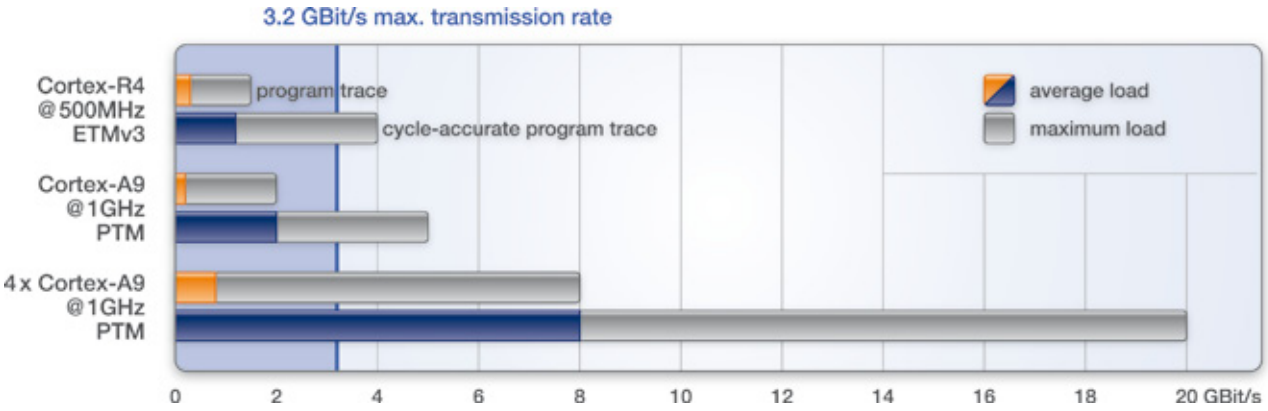

図8: 通常、転送速度3.2GBit/sは記録中のプログラムトレース情報をホストに転送する場合に十分な速度である.

CombiProbeによるエネルギープロファイリング

TRACE32 CombiProbeは、アプリケーションで使用されるエネルギーの測定用途にも対応できるようになりました。

次のような解析が可能です。

- プロセッサ上で実行されているコードに直接リンクする形で、最大3つの測定点での電流/電圧プロファイルを表示可能です。
- システム全体のエネルギー消費量を個別の関数ごとに解析可能です。

「プログラムのどの部分がエネルギーを最も消費しているのか」、「プログラム修正によって、組み込みシステムのエネルギー要件にどのような影響が現れるのか」。CombiProbeを使用することで、このような疑問点の答えが得られるようになりました。

プログラムの各ポイントでのエネルギー消費量を判別するには、次のような測定データを収集する必要があります。

- プロセッサのトレースポートからエクスポートされるプログラムフロー。
- ターゲットハードウェアの適切な測定点で測定された電流および電圧プロファイル。

最大3つのパワードメインの電流および電圧の変化について、TRACE32 Analog ProbeをCombiProbeに接続することで識別できるようになりました。

すべての測定データにはCombiProbeのグローバルタイマーによってタイムスタンプが付加されるので、実行されたプログラムコードと電力消費量やシステムの電圧プロファイルとの直接的な関連性を簡単に確認できます。

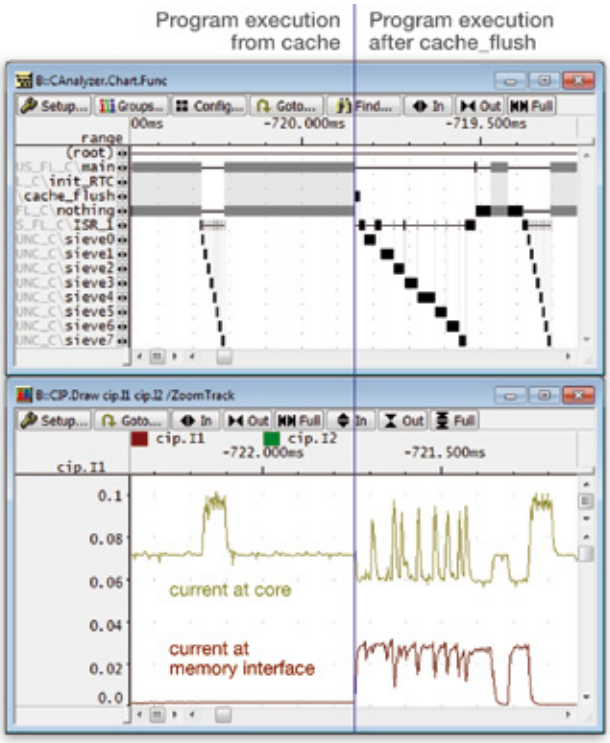

図9: キャッシュから実行されていないプログラムセクションは処理時間が長くなり、電力消費量が大きくなる

図9に、プログラムセクションがキャッシュからではなく、外部メモリから実行された場合の状況を示します。この場合、必要な処理時間が長くなるだけでなく、外部メモリで使用する電力消費量も増加します。

図10に、統計解析形式のエネルギー消費量の情報を示します。

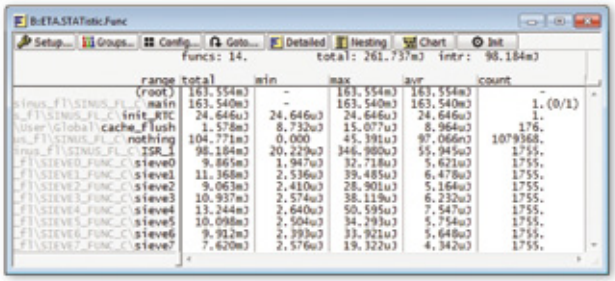

図.10 :個別の関数の最小、最大、平均のエネルギー消費量.

CombiProbe

CombiProbeは、128MBのトレースメモリを搭載したデバッグケーブルです。CombiProbeは、4ビットトレースポートを備えたプロセッサ専用に開発されました。プログラムフローの記録については現在、次のトレースプロトコルがサポートされています。

- 連続モードのARM-ETMv3（ARM社）
- PIC32向けIFLOW Trace（Microchip社）
- X-GOLD102およびX-GOLD110向けMCDS Trace（Infineon社）

www.lauterbach.com/cobstm.html

SMPプロファイリング

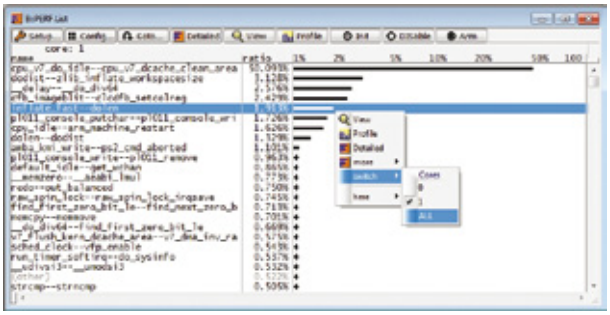

図11: サンプルベースのプロファイリングにより、実行時間全体に占める個別のコードセクションの比率をパーセント単位で表示したもの。結果は、SMPシステムの個別のコア（この例では、core 1）と全コアの合計という両形式で表示可能

サンプルベースのプロファイリングは、2010年に全面的に見直されました。重要な新技術としては、新しい運用概念に基づく測定手法、サンプリングレートのセルフキャリブレーション、SMPシステムの機能拡張などが挙げられます。

関数のSMPプロファイリング

2010年10月から、SMPシステムのプロファイリングデータの収集が可能になりました。関数レベルのプロファイルを作成するため、TRACE32では、個別のコアのプログラムカウンタをサイクルごとに読み取って、データベースに保存します。保存されたプロファイル情報は、個別のコアおよび全コアの合計として表示できます。

多くのチップには、プロセッサの実行中にプログラムカウンタを読み取る機能が用意されています。この機能により、次のアーキテクチャでは、リアルタイムに測定を実施できます。

- **ARM / Cortex:** ARM11 MPCore, Cortex-A5 MPCore, Cortex-A9 MPCore, Cortex-A15 MPCore

世界の支社

- 日本
- ドイツ
- フランス
- イギリス
- イタリア
- 中国
- アメリカ

その他の国々でも経験豊富な
販売代理店が対応させていただきます

サンプルベースのプロファイリング

サンプルベースのプロファイリングでは、プログラムカウンタか、現在のタスクのIDを格納している変数が定期的に読み取られます。この情報に基づいて、実行時間全体に占める関数またはタスクの比率がパーセント単位で表示されます。

Symmetrical Multiprocessing (SMP)

同じ複数のコアで構成されているマルチコアチップは、SMPシステムとして構成できます。SMPオペレーティングシステムは、（プログラム実行前ではなく）プログラム実行時に、保留中のプロセス（タスク）を動的に個別のコアに分配します。SMPシステムのデバッグ時には、すべてのコアを監視するための、単一のTRACE32インスタンスが開きます。

- **MIPS32:** MIPS34K, MIPS1004K
- **MIPS64:** Broadcom BCM7420

チップのデバッグロジックによる情報の読み取りに伴って負荷がかかる場合は、個別のコアを定期的に停止して、情報を取得する必要があります。

タスクのSMPプロファイリング

タスクプロファイルを作成するため、個別のコアのタスクIDをサイクルごとにメモリから読み取る必要があります。多くのチップには、プログラムの実行時に、物理メモリを読み取る機能が用意されています。オンチップのデバッグロジックで、この機能をサポートしている場合は、リアルタイムに測定を実行できます。

- **ARM / Cortex:** ARM11 MPCore, Cortex-A5 MPCore, Cortex-A9 MPCore, Cortex-A15 MPCore
- **Power Architecture:** MPC8641D, MPC8572, QorIQ

この機能をサポートしていない場合は、個別のコアを一時的に停止し、必要なデータを読み取る必要があります。

連絡先更新のお願い

アドレスの変更があった場合、またはメーリングリストからの退会をご希望の場合には、info@lauterbach.co.jp まで電子メールをお送りください。.