NEWS 2011

LAUTERBACH DEVELOPMENT TOOLS

DEBUGGER, REAL-TIME TRACE, LOGIC ANALYZER

Immer ein paar Schritte voraus

Nach diesem Motto entwickelt Lauterbach seit über 30 Jahren Entwicklungswerkzeuge für die Embedded Industrie. Meist ist Lauterbach dabei seiner Zeit weit voraus und setzt früh Trends.

So haben wir uns die Anerkennung aller großen Halbleiterhersteller erworben. Eine enge Kooperation mit Lauterbach bei der Entwicklung und Umsetzung von neuen Techniken wird deshalb seit Jahren gesucht und geschätzt. Durch diese Zusammenarbeit konnten wegweisende Ideen in fortschrittliche Produkte umgesetzt werden.

Zudem setzt Lauterbach auf eine kompromisslose Kundenorientierung. Wünsche und Anregungen der TRACE32-Nutzer sind für die Produktentwicklung ein wertvoller Beitrag. In vielen Fällen werden Vorschläge direkt umgesetzt und sind schon in der nächsten Release-Version allgemein verfügbar.

Welche Trends sieht Lauterbach aktuell? Welche neuen Technologien sind gerade dabei, sich im Markt zu etablieren?

Android-Debugging

Ein wichtiges Thema ist sicher Android-Debugging. Applikationen für Mobiltelefone werden zunehmend architekturunabhängig für virtuelle Maschinen (VM) geschrieben. Google's Android und die dazugehörige Dalvik VM sind hier sehr verbreitet. Komplexe Fehler, die erst durch das Zusammenspiel von Anwendung, virtueller Maschine, Betriebssystem und darunterliegender Hardware entstehen, müssen natürlich auch zu debuggen sein. Transparenz durch alle Software-Schichten, von der Java-Anwendung bis hinunter zu den Linux Hardware-Treibern ist hier notwendig.

Auf Wunsch einiger Mobilfunkhersteller startete Lauterbach Mitte 2010 mit der Entwicklung einer API für eine *VM Debugging Awareness*. Android wird dabei als Referenzplattform verwendet. Ziel ist eine offene Schnittstelle, die es Anbietern von Open- und Closed-Source VMs erlaubt, ihre Produkte für ein Debugging mit TRACE32 anzupassen. Informationen zur *VM Debugging Awareness* »

INHALT

und zum aktuellen Stand der Entwicklung finden Sie im Artikel „API für *VM Debugging Awareness*“ auf Seite 6.

Energy Profiling

Ein durch Green-IT und Klimaerwärmung in den Fokus gerücktes, aber aus Lauterbach-Sicht schon etwas älteres Thema, ist die Energiemessung für Embedded Systeme. In jeder Fachzeitschrift finden sich aktuell eine Vielzahl von Artikeln zum Thema batteriebetriebene Geräte und Low-Power-Microcontroller. Preise für Innovationen werden bevorzugt für neue Technologien in diesem Bereich vergeben.

Standby- bzw. Betriebszeiten waren aber beispielsweise im Mobilfunkmarkt von Anfang an ein Thema. Seit Jahren werden dort umfangreiche Maßnahmen zur Reduktion des Energieverbrauchs durchgesetzt. Diese Maßnahmen greifen jedoch nur dann, wenn die Software, die ein Embedded System steuert, alle Energiesparmöglichkeiten der Hardware konsequent nutzt.

Seit Jahresbeginn 2006 unterstützt Lauterbach bereits Messanordnungen, die es erlauben, den Zusammenhang zwischen Software und Strom-/Leistungsaufnahme in einem Embedded System einfach zu visualisieren und zu bewerten. Seit Mitte 2010 steht dieses Feature jetzt auch für die TRACE32-CombiProbe zur Verfügung. Alles zum Thema „Energy Profiling mit der CombiProbe“ finden Sie auf Seite 11.

Multicore-Debugging

Obwohl Multicore-Chips in Embedded Systemen seit zehn Jahren eingesetzt werden und Lauterbach bereits seit 2001 Debugger dafür anbietet, hat dieses Thema immer noch hohe Dynamik. Aktuell sorgt der Wunsch nach mehr Sichtbarkeit interner Abläufe für die Integration immer neuer Tracezellen in die Debug-Infrastruktur der Chips.

Während anfänglich nur Traceinformationen für die einzelnen Cores generiert wurden, gibt es heute eine Vielzahl weiterer Tracequellen.

a) Tracequellen, die Transfers auf den chipinternen Bussen sichtbar machen:

- ARM CoreSight mit der AMBA AHB Trace Macrocell (HTM)
- MCDS mit dem System Peripheral Bus (SPB) und dem Local Memory Bus (LMB) für den TriCore von Infineon
- RAM Trace Port für Chips von Texas Instruments
- DMA- und FlexRay-Trace für NEXUS Power Architecture

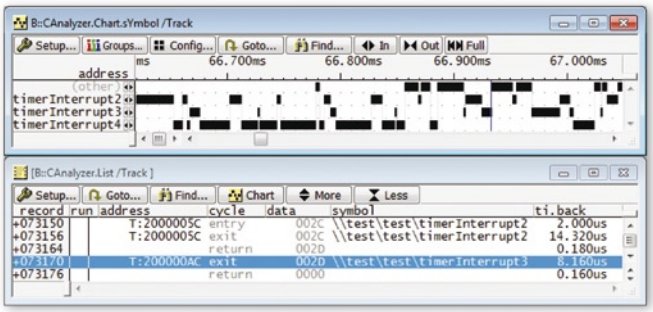

Bild 1: Interrupttrace beim Cortex-M3.

b) Tracequellen, die Traceinformationen für chipinterne IP (Intellectual Property) generieren, wie etwa spezielle Interrupttraces (siehe Bild 1).

c) Tracequellen, die die Ausgabe von Software-generierter Traceinformation erlauben, wie beispielsweise:

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

Diese neuen Tracequellen erfordern natürlich die ständige Weiterentwicklung des TRACE32-Bedienkonzepts. Wichtig ist eine einfache Konfigurierbarkeit der Tracezellen, sowie eine gute Visualisierung der von ihnen erzeugten Informationen.

Serielle Traceports

Für die Sichtbarkeit aller chipinternen Abläufe benötigen komplexe Multicore-Chips und schnelle Hochleistungsprozessoren natürlich immer mehr Bandbreite und damit immer schnellere Traceports.

Chiphersteller haben daher in den letzten Jahren als wichtige Neuerung schnelle serielle Traceschnittstellen entwickelt. Festplattenhersteller, die bereits seit Jahren serielle Schnittstellen für den high-speed Datenaustausch zum PC verwenden, nutzten 2008 erstmals diese Technik, um auch Traceinformation über ARM´s *High Speed Serial Trace Port* (HSSTP) auszugeben. Zeitgleich brachte Lauterbach Tracetools für diese Technologie auf den Markt.

Inzwischen gibt es weitere Chips mit seriellen Traceschnittstellen. Über die aktuellen Entwicklungen in diesem Bereich informiert Sie der Artikel auf Seite 9.

Größere Tracespeicher

Schnelle Traceschnittstellen mit ihren hohen Datenraten erfordern zwangsläufig größere Tracespeicher. Nur so lässt sich ein ausreichend großer Programmausschnitt »

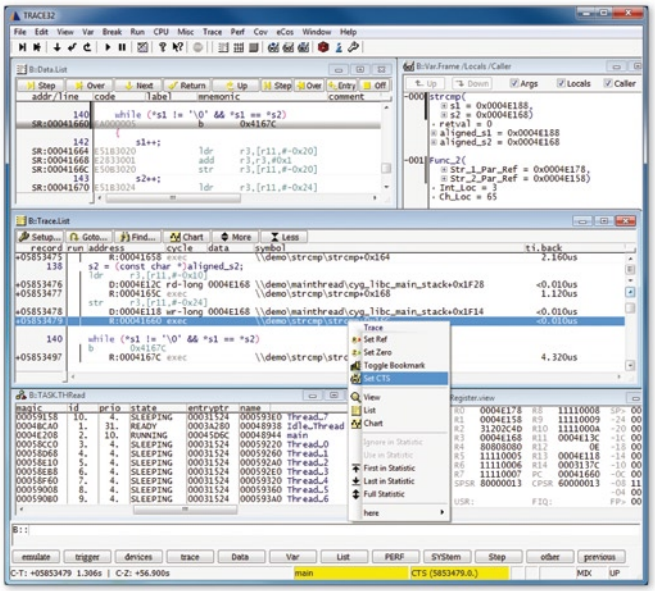

Bild 2: Daten- und rechenintensive Traceanalysefunktionen wie das Trace-Based Debugging können auch bei einem 4 GB großen Tracespeicher schnell durchgeführt werden.

für die Fehlersuche und die Analyse des Zeitverhaltens für ein Embedded System erfassen.

Immer größere Tracespeicher anzubieten macht jedoch nur dann Sinn, wenn auch die Infrastruktur für eine zügige Verarbeitung der Traceinformationen vorhanden ist. Dies gilt insbesondere für daten- und rechenintensive Traceanalysefunktionen wie das *Trace-Based Debugging* (siehe Bild 2). Sinkende Preise bei SDRAM-Speichern, schnelle PCs und Gigabit-Ethernet-Schnittstellen ermöglichten es Lauterbach, das Tracetool PowerTrace II mit 4 GB Speicher 2007 auf den Markt zu bringen.

Mitte 2008 startete Lauterbach dann mit der Entwicklung eines neuen Verfahrens für die Traceaufzeichnung und Analyse, dem so genannten *Real-Time Streaming*. Getrieben wurde diese Entwicklung von dem Kundenwunsch für die Code-Abdeckungsanalyse, für umfassende Systemlaufzeitanalysen und für die Untersuchung selten auftretender Fehler die Traceaufzeichnungszeit wesentlich zu erhöhen.

Neu beim *Real-Time Streaming* ist, dass die Tracedaten bereits zur Aufzeichnungszeit auf den Hostrechner übertragen werden. Sobald die Traceinformationen dort angekommen sind, werden diese sofort ausgewertet. Optional kann die Traceinformation parallel zur Analyse auch noch auf der Festplatte gespeichert werden.

Real-Time Streaming funktioniert nur dann, wenn alle Verarbeitungsschritte für die Tracedaten geschwindigkeitsoptimiert ablaufen. Das betrifft Übertragung und Analyse, sowie die gezielte Suche nach Traceinformation in einer auf der Festplatte gespeicherten Datei.

Trace-Based Debugging

Trace-Based Debugging (auch CTS = Context Tracking System) erlaubt es, den im Tracespeicher aufgezeichneten Programmabschnitt in TRACE32-PowerView nachträglich noch einmal schrittweise zu debuggen. Dabei wird die Tatsache genutzt, dass TRACE32 den Zustand des Zielsystems für jeden einzelnen Aufzeichnungszeitpunkt rekonstruieren kann. Zustand des Zielsystems bedeutet: Register- und Speicherinhalte, Variablenzustände, Source- und Task-Listing, Stack-Frame und vieles mehr.

Nachdem ein Startpunkt für das Trace-Based Debugging ausgewählt wurde, sind alle Debug-Kommandos nutzbar. Diese werden von TRACE32 in Analogie zur Traceaufzeichnung abgearbeitet. Viele Nutzer des Trace-Based Debugging schätzen es, dass man auch rückwärts steppen und zum Funktionsanfang zurückkehren kann.

Trace-Based Debugging bietet darüber hinaus eine Reihe weiterer nützlicher Funktionen:

- Tracedarstellung in Hochsprache mit allen lokalen Variablen
- Laufzeitanalysen und Funktionsaufrufbaum
- Schließen von Tracelücken, die dadurch entstanden sind, dass mehr Tracedaten generiert wurden als der Traceport ausgeben kann

www.lauterbach.com/cts.html

Von vielen der dazu neu entwickelten Optimierungen profitiert auch das konventionelle Tracing. So ist beispielsweise geplant, die für das *Real-Time Streaming* entwickelte Kompression der Tracedaten auch dann zu verwenden, wenn die Daten erst nach Beendigung der Aufzeichnung auf den Host übertragen werden. Ausführliche Informationen zu diesem Thema finden Sie auf Seite 10.

Ausblick

Neben den aktuellen Trends gibt es natürlich eine Vielzahl von Weiterentwicklungen. Beim Durchblättern unseres Newsletters für 2011 werden Sie sicher das eine oder andere für Ihr Projekt entdecken. Einiges davon werden wir Ihnen vom 1. bis 3. März 2011 auf der *embedded world* in Nürnberg live präsentieren.

Besuchen Sie uns: Halle 10, Stand 325.

embedded world 2011
Exhibition&Conference
... it's a smarter world

Neu unterstützte Prozessoren

Neue Derivate

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 (Forts.)	**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

Neue Derivate	
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

Tracing für Virtual Targets in Fast Models

Seit November 2010 unterstützt Lauterbach das Tracing für ARM Fast Models.

Um mit dem Start der Software-Entwicklung nicht auf die ersten Hardware-Prototypen warten zu müssen, werden oft Software-Modelle der Hardware verwendet. Mit Fast Models bietet ARM seinen Kunden ein Software-Paket für die Programmierung von Modellen für ARM-basierte Designs.

Seit 2008 unterstützt Lauterbach bereits das Debugging von Fast Models über die CADI-Schnittstelle. Neu ist nun eine Unterstützung für das *Model Trace Interface*, das mit Version 5.1 für Fast Models eingeführt wurde. Um die Traceinformationen passend aufzubereiten und im virtuellen Target zwischenzuspeichern, gibt es für Debugger-Hersteller die Möglichkeit, ein eigenes Trace-Plugin zu laden. Bild 3 zeigt das Zusammenspiel von TRACE32 und Fast Models in einer Übersicht.

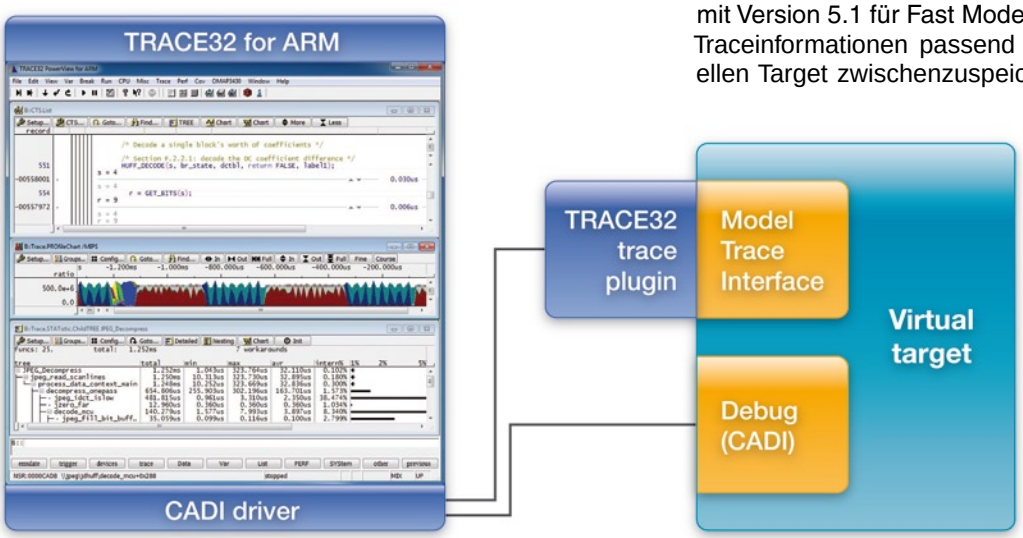

Bild 3: TRACE32 unterstützt das Debugging und Tracing von virtuellen Targets.

Ausführliche Informationen zum Thema Debugging von virtuellen Targets finden Sie unter:

www.lauterbach.com/frontend.html

API für VM Debugging Awareness

Seit Mitte 2006 unterstützt Lauterbach das Debugging von Java-Anwendungen für die Java Virtual Machines J2ME CLDC, J2ME CDC und Kaffe. Da sich virtuelle Maschinen einer wachsenden Beliebtheit erfreuen, steigt auch die Zahl der Anbieter. Und längst sind nicht mehr alle virtuellen Maschinen Open-Source. Um VM-Anbietern und ihren Kunden die Möglichkeit zu geben, das Debugging schnell und flexibel für ihre VM anzupassen, arbeitet Lauterbach seit Mitte 2010 an einer offenen Lösung.

Als Referenz für die Entwicklung einer VM API für das Stop-Mode Debugging wird die für ARM Cores implementierte Android *Dalvik Virtual Machine* verwendet.

Zwei Debug-Welten

Für alle, die mit Android nicht vertraut sind, zunächst eine kleine Einführung. Aus Entwicklersicht ist Android ein Open-Source Software-Stack, der aus folgenden Komponenten besteht (siehe Bild 4):

- Ein Linux Kernel mit seinen Hardware-Treibern.
- Die *Android Runtime* mit der *Dalvik Virtual Machine* und einer Reihe von Bibliotheken: klassische Java Core Libraries, Android-spezifische Libraries, in C/C++ erstellte Libraries.
- In Java programmierte Anwendungen und das sie unterstützende *Application Framework*.

Software für Android wird in verschiedenen Programmiersprachen geschrieben:

- Der Linux Kernel, einige Libraries und die *Dalvik Virtual Machine* werden in C bzw. C++ oder Assembler codiert.

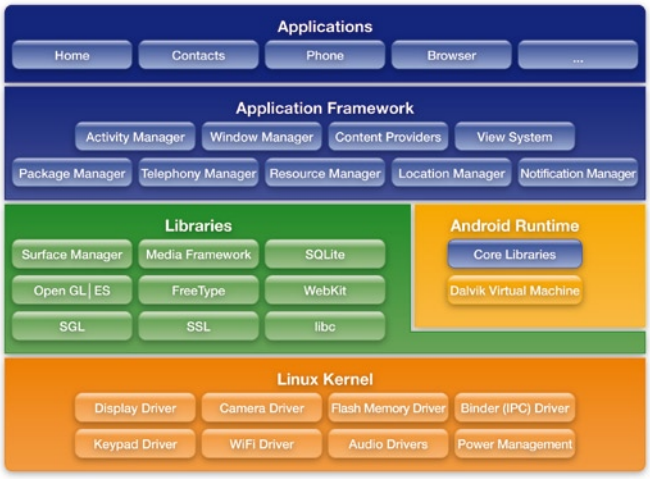

Bild 4: Der Open-Source Android Software-Stack.

- VM-Anwendungen und das sie unterstützende *Application Framework* werden in Java programmiert.

Getestet wird der jeweilige Code in seiner eigenen Debug-Welt.

Debugging von C/C++ und Assembler Code

Den in C/C++ und Assembler codierten Teil von Android kann man hardwarenah über die JTAG-Schnittstelle im Stop-Mode debuggen. Lauterbachs TRACE32 kommuniziert dazu direkt mit dem Prozessor der Android Hardware-Plattform (siehe Bild 5).

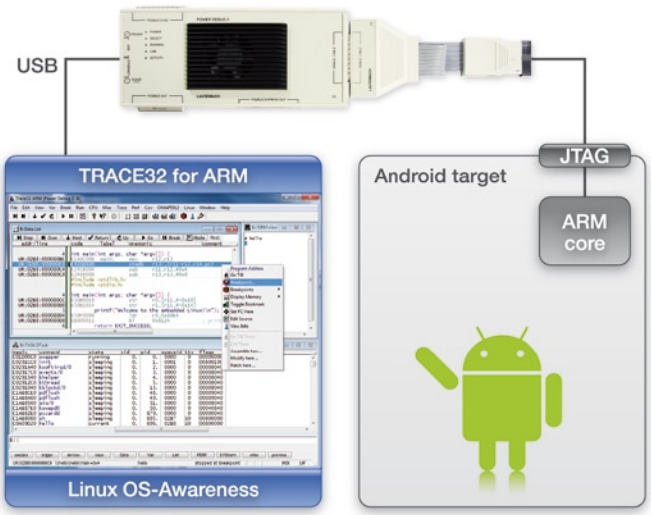

Bild 5: Beim Stop-Mode Debugging kommuniziert der Debugger direkt mit dem Prozessor auf der Android Hardware-Plattform.

Charakteristisch für das Stop-Mode Debugging ist, dass immer dann, wenn der Prozessor für das Debugging angehalten wird, das gesamte Android System stoppt.

Stop-Mode Debugging hat einige große Stärken:

- Es benötigt nur die funktionierende JTAG-Kommunikation zwischen Debugger und Prozessor.
- Stop-Mode Debugging benötigt keinen *Debug Server* auf dem Target und eignet sich deshalb auch gut für das Testen von Release-Software.
- Stop-Mode Debugging erlaubt ein Testen unter realen Zeitbedingungen und ermöglicht darum eine effiziente Fehlersuche für Probleme, die erst unter Echtzeitbedingungen auftreten. »

Stop-Mode Debugging unterstützt im Augenblick jedoch noch nicht das Debugging von VM-Anwendungen, z.B. auf der Dalvik VM. Daher ist ein transparentes Debugging durch alle Software-Schichten bisher nicht möglich.

Debugging von Java-Code

Java-Code für Android wird üblicherweise mit den in Eclipse integrierten *Android Development Tools* (ADT) getestet. Der *adb server* – adb steht für Android Debug Bridge – auf dem Host kommuniziert dazu über USB oder Ethernet mit dem *adb daemon* auf dem Target (Bild 6).

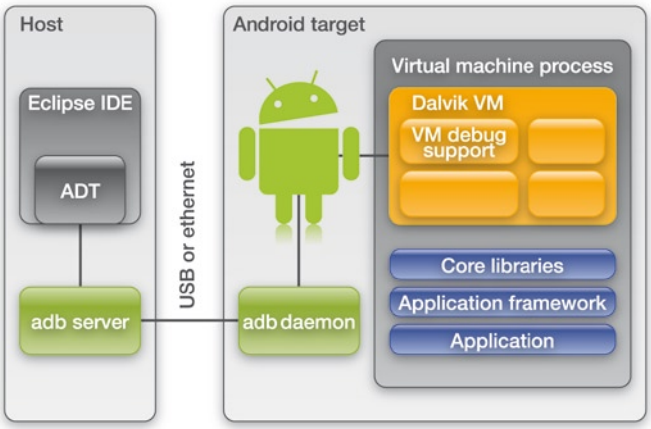

Bild 6: Die in Eclipse integrierten Android Development Tools (ADT) zum Debuggen von Java-Code.

Voraussetzung für das Testen mit ADT sind speziell für Debugging kompilierte VM-Anwendungen und eine Deklaration der Debug-Erlaubnis im *Application Manifest*. Zudem muss Debugging für die Hardware-Plattform generell freigeschaltet sein.

Das Debugging des Java-Codes mit ADT ist recht komfortabel. Es gibt jedoch Fälle, in denen man mit ADT nicht weiterkommt. Dies sind:

- Fehler, die erst mit dem Release-Code auftreten.
- Fehler, die erst dann auftreten, wenn die Java-Anwendung mit einem in C/C++ angebotenen Dienst oder einem Linux Hardware-Treiber interagiert.
- Debugging nach dem Zusammenbruch der Kommunikation zwischen *adb server* und *adb daemon*.

VM Aware Stop-Mode Debugging

Um ein durchgängiges Testen eines Android Systems von der Java-Anwendung bis hinunter zum Linux Hardware-Treiber unter Realzeitbedingungen zu ermöglichen, erweitert Lauterbach aktuell das Stop-Mode Debugging um eine *VM Debugging Awareness*.

Der JTAG-Debugger kommuniziert direkt mit dem Prozessor auf der Android Hardware-Plattform. Daher kann der Debugger nach dem Anhalten des Prozessors auf alle Systeminformationen zugreifen. Die „hohe Kunst“ für den Debugger besteht nun darin, die richtigen Informationen zu finden und für den Nutzer leicht verständlich, von Bits und Bytes abstrahiert, darzustellen.

Eine Abstraktionsebene bot bisher dem TRACE32-Nutzer die Möglichkeit, Betriebssystem-Software, ggf. über mehrere virtuelle Adressräume hinweg, zu debuggen. Eine andere, bisher vom Betriebssystem-Debugging unabhängige Abstraktionsebene, stellt das Java-Debugging dar.

Um in Systemen wie Android auf VMs laufende Anwendungen zu debuggen, wobei die VMs ihrerseits in Betriebssystem-Prozessen instanziiert sind, muss Betriebssystem- und Java-Debugging nun kombiniert werden. Zur Umsetzung dieser neuen Komplexität entwickelt Lauterbach eine neue, offene und leicht erweiterbare Lösung.

Die offene Lösung

Das Stop-Mode Debugging von Lauterbach unterstützt zukünftig folgende Abstraktionsebenen:

- Hochsprachen-Debugging
- Target-OS Debugging Awareness
- VM Debugging Awareness

Das **Hochsprachen-Debugging** ist fester Bestandteil der TRACE32-Software und wird mit dem Laden der Symbol- und Debug-Informationen für ein Programm passend konfiguriert.

Die **Target-OS Debugging Awareness** muss immer vom TRACE32-Nutzer konfiguriert werden. Dies ist einfach möglich, da es für alle gängigen Betriebssysteme ferti- »

Dalvik Virtual Machine

Dalvik ist der Name der in Android benutzten virtuellen Maschine. Die *Dalvik Virtual Machine* ist ein in Software programmiertes Modell eines Prozessors, das einen von Java abgeleiteten Byte-Code ausführt. Virtuelle Maschinen erlauben es, prozessorunabhängige Software zu schreiben. Beim Umstieg auf eine neue Hardware-Plattform muss lediglich die virtuelle Maschine portiert werden.

Für VM kompilierte Software läuft automatisch auf jeder Plattform, auf die diese VM portiert ist.

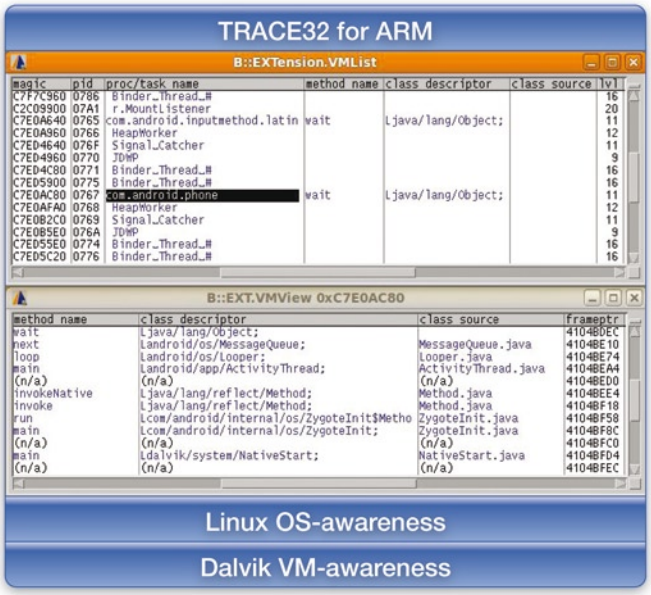

Bild 7: Für die Referenzimplementierung muss die Linux OS-Awareness und die Dalvik VM-Awareness in TRACE32 geladen werden.

ge Konfigurationen gibt. Für proprietäre Betriebssysteme bietet die RTOS API eine offene Möglichkeit zur Anpassung.

Die **VM Debugging Awareness** ist für J2ME CLDC, J2ME CDC und Kaffe direkter Bestandteil der TRACE32-Software. Alle anderen virtuellen Maschinen können mit der VM API individuell angepasst werden. Für die sehr populäre Android Dalvik VM steht, ähnlich wie für die *Target-OS Debugging Awareness,* eine fertige Konfiguration zur Verfügung.

Die offene Lösung, sowohl für das Betriebssystem als auch für die virtuelle Maschine, erlaubt es auch Anbietern von Closed-Source VMs, eine TRACE32 VM-Awareness für ihr Produkt zu erstellen und ihren Kunden anzubieten.

Die Referenzimplementierung

Um auf einem ARM-basierten Android Target von der Java-Anwendung bis hinunter zu den Linux Hardware Treibern durchgängig debuggen zu können, benötigt TRACE32 folgende Erweiterungen (siehe Bild 7):

- Eine Linux OS-Awareness, die von Lauterbach bereits seit 1998 angeboten wird.
- Eine VM-Awareness für Dalvik, die von der Lauterbach Web-Seite heruntergeladen werden kann. Diese muss lediglich für die benutzte Plattform konfiguriert werden.

www.lauterbach.com/vmandroid.html

Aktuell ist es möglich, alle Java-Anwendungen, die gerade ausgeführt werden, zu ermitteln und aufzulisten (EXTension.VMList im Bild 7). Ebenso lässt sich der VM-Stack für eine ausgewählte Java-Anwendung analysieren und darstellen (EXTension.VMView im Bild 7). Als nächster Schritt ist geplant, den von der VM gerade ausgeführten Quellcode anzuzeigen. Das Ziel der Entwicklung ist natürlich das Stop-Mode Debugging für VM-Anwendungen mit allen Funktionen eines modernen Debuggers.

Neu unterstützte RTOS	
DSP/BIOS für ARM	Q2/2011
OSEK/ORTI SMP	Q2/2011
T-Kernel für ARM	verfügbar
VDK für Blackfin	verfügbar
Windows Embedded Compact 7 für ARM	verfügbar
µC/OS-III für ARM	verfügbar

Erweiterungen und neue RTOS-Versionen

- TRACE32-Skripte wurden für *Timesys embedded Linux* angepasst.
- OSEK/ORTI sorgt nun dafür, dass bei Taskwechseln *NEXUS Ownership Trace Messages* erzeugt werden. Dadurch lassen sich mit TRACE32 Task-Aware-Laufzeitmessungen für den MPC55xx/MPC56xx durchführen, auch wenn NEXUS keine *Data Trace Messages* erzeugt.

Folgende Versionsanpassungen wurden bereits durchgeführt bzw. sind geplant:

- OSEck 4.0
- QNX 6.5.0
- Symbian^3 für ARM,
- Symbian^4 geplant für Q1/2011
- Windows CE6 für Atom™

Serieller Traceport auf dem Vormarsch

Schneller, höher, stärker! Das ist nicht nur das Motto bei vielen Sportarten. Auch in der Mikroelektronik scheint dieser Wahlspruch zum Prinzip erhoben. Immer höhere Taktraten und eine stärkere Parallelisierung der Verarbeitungsschritte haben eine seit Jahrzehnten erstaunlich konstante Steigerung der Verarbeitungsgeschwindigkeit ermöglicht. Kein Wunder also, dass man bisher auch bei der Übertragung von Traceinformationen auf diese Strategie gesetzt hat.

Gerade die Traceschnittstelle, über die die Prozessoren wichtige Informationen zu ihren inneren Abläufen preisgeben, musste mit der wachsenden Informationsflut mithalten. Auf wichtige Informationen aus dem Innersten des Prozessors zu verzichten ist für viele Entwickler von Embedded Systemen undenkbar. Daher wurden vielfältige Anstrengungen unternommen, den Datendurchsatz der Traceschnittstelle zu erhöhen. Die Steigerung der Taktraten und eine größere Busbreite am Traceport waren bisher probate Mittel für höhere Datenmengen.

Diese Maßnahmen haben jedoch auch ihren Preis. Nicht nur, dass ein breiter Traceport viele „heiß begehrte“ Prozessorpins belegt, Laufzeitunterschiede und schlechtere Signalqualität bei höheren Taktraten müssen über alle Signale des Tracebusses ausgeglichen werden. Nur durch die aufwändigen schaltungstechnischen Maßnahmen und die ausgeklügelten Algorithmen seiner AutoFocus-Technologie konnte Lauterbach eine fehlerfreie Aufzeichnung hochfrequenter Tracesignale sicherstellen.

Doch während in der Prozessorarchitektur weiter parallelisiert wird, setzt man bei den Traceschnittstellen auf eine Technik, die bereits seit längerem in anderen Bereichen zum Datentransfer genutzt wird. Bei SATA, Fibre Channel, PCI Express und USB3.0 wird eine serielle high-speed Übertragung verwendet. Der Nachteil nur sehr weniger differentieller Datenleitungen wird durch extrem hohe Datenraten mehr als wettgemacht.

Die Integration von hochfrequenten seriellen Schnittstellen auf dem Chip ist teuer und bringt zunächst auch einige Probleme mit sich. So müssen beispielsweise die I/O-Pads mit viel höherer Geschwindigkeit betrieben werden. Doch mit zunehmender Erfahrung bei der Realisierung verschiedener serieller Schnittstellen im Gigaherzbereich, nutzt man nun dieses Wissen auch mehr und mehr für die seriellen Traceports.

Im Jahr 2008 hatte ARM bereits mit seinem *High Speed Serial Trace Port* – kurz HSSTP – die erste Implementierung dieser Technik umgesetzt. Kurz darauf folgten AMCC mit dem Titan, Freescale mit den QorIQ Prozessoren P4040 und P4080, sowie Marvell mit der SETM3.

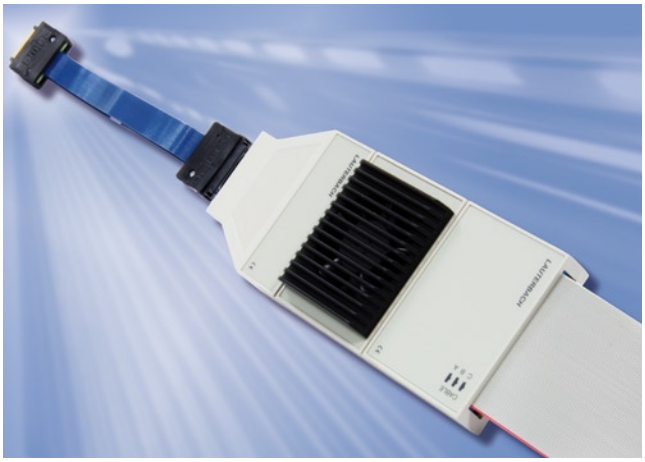

Bild 8: Eine universelle Hardware unterstützt nach Firm- und Softwareanpassung die unterschiedlichsten Protokolle serieller Traceschnittstellen.

Lauterbach war bereits von Anfang an mit einer Hardware für den seriellen Trace mit dabei. Basierend auf dem Aurora-Protokoll wurde ein universeller Präprozessor entwickelt. Für die Aufzeichnung der unterschiedlichen Protokolle muss lediglich die Firm- und Software angepasst werden. So ist man auf weitere Varianten serieller Traceprotokolle bestens vorbereitet.

Für die hohen Datenraten der seriellen Traceschnittstellen steht ein bis zu 4 GByte großer Tracespeicher zur Verfügung.

Unterstützte Serielle Traceports

AMCC	APM83290 Programmfluss	2009
ARM-HSSTP	ETMv3, PTM, CoreSight ETMv3, CoreSight PTM Programmfluss, Datenfluss und Context-ID	2008
Freescale	NEXUS QorIQ P4040 und P4080 Branch Trace und Ownership Trace Messages, Data Write Messages	2010
Marvell-SETM3	CoreSight ETMv3 Programmfluss, Datenfluss und Context-ID	2009

Höhere Übertragungsrate für Real-Time Streaming

Real-Time Streaming ist die Technik, die Tracedaten bereits zur Aufzeichnungszeit auf den Host zu übertragen und dort sofort zu analysieren. Dies erfordert die Übertragung großer Datenmengen vom Tracetool zum Host, insbesondere für rechenintensive Anwendungen und Multicore-Systeme. Um TRACE32 für diese Anwendungsszenarien „fit" zu machen, werden die Tracedaten jetzt vor der Übertragung zum Host im Tracetool PowerTrace II komprimiert. Dieses Feature wird seit Dezember 2010 von der TRACE32-Software unterstützt.

Real-Time Streaming ist aktuell für die ARM-Traceprotokolle ETMv3 und PTM implementiert.

Hardware-Kompression

Die maximale Übertragungsrate zum Hostrechner ist nach wie vor der Engpass für das *Real-Time Streaming*. Selbst mit einer Peer-to-Peer GBit Ethernet-Schnittstelle zwischen Tracetool und Host erreicht man aktuell nur etwa 500 MBit/s netto. Diese maximale Übertragungsrate muss ausreichen, um alle am Traceport ausgegebenen Daten verlustfrei zum Hostrechner zu übertragen.

Um die tatsächlich zu übertragende Datenmenge realistisch abschätzen zu können, ist es wichtig, die Randbedingungen des *Real-Time Streaming* zu kennen:

1. Die Hauptanwendungen für das *Real-Time Streaming* sind Code-Coverage und Laufzeit-Messungen. Für beide Funktionen ist es ausreichend, wenn nur Programmfluss-Informationen ausgegeben werden. Um die Laufzeitmessungen zu präzisieren, kann *Cycle-Accurate Tracing* aktiviert werden.
2. Für eine realistische Abschätzung der notwendigen Datenrate genügt es, lediglich die durchschnittliche Last am Traceport zu betrachten. Lastspitzen am Traceport werden vom PowerTrace II abgefangen, der als großer FIFO (bis zu 4 GB) betrachtet werden kann.

Bild 9 zeigt eine Übersicht der durchschnittlichen bzw. maximalen Last am Traceport für Cortex-Cores. Die auf dem Cortex-Core laufende Anwendung bestimmt letztlich immer die tatsächliche Last.

Durch die Implementierung einer FPGA-basierten Hardware-Kompression im PowerTrace II konnte die Übertragungsrate zum Host auf 3,2 GBit/s gesteigert werden.

Reiner Langzeit-Trace

Werden beim *Real-Time Streaming* die Tracedaten auf dem Host nicht nur analysiert, sondern auch auf der Festplatte abgespeichert, benutzt Lauterbach dafür den Begriff „Langzeit-Trace".

Um auch für andere Traceprotokolle, wie etwa Nexus, einen Langzeit-Trace anbieten zu können, wird Lauterbach 2011 ein reines Streaming auf die Festplatte ohne gleichzeitige Analyse anbieten. Damit ist für ein 64-Bit Host-Betriebssystem eine Traceaufzeichnung von bis zu 1 Tera-Frames möglich.

Ausführliche Informationen zum Thema *Real-Time Streaming* und Langzeit-Trace finden Sie auf der Lauterbach-Homepage unter:

www.lauterbach.com/streaming.html

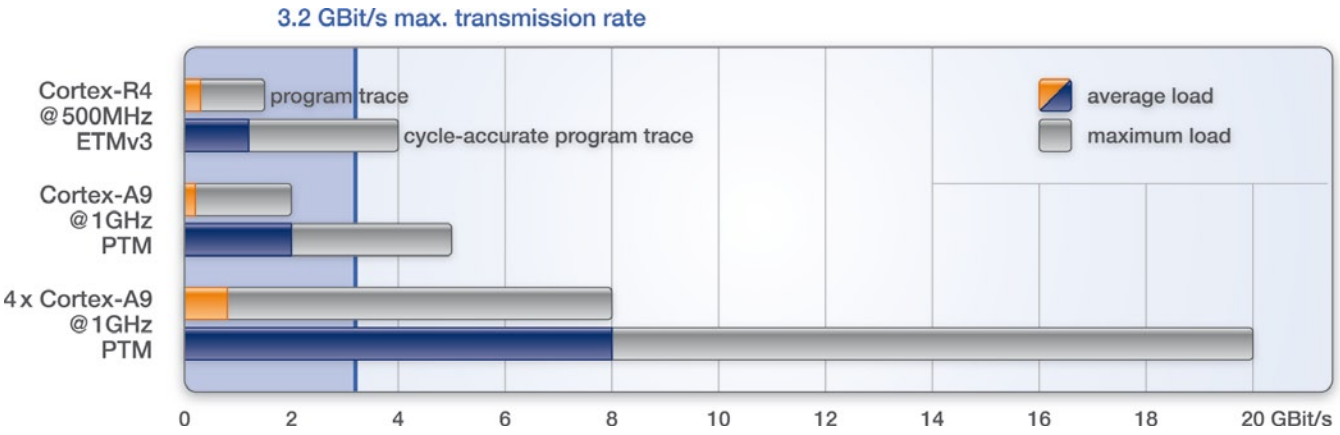

Bild 9: Eine Übertragungsrate von 3,2 GBit/s reicht in der Regel aus, um die Programmfluss-Informationen zur Aufzeichnungszeit auf den Host zu übertragen.

Energy Profiling mit der CombiProbe

Wer bereits eine TRACE32-CombiProbe im Einsatz hat, kann diese ab sofort auch für die Energiemessung seiner Anwendungen verwenden.

Folgende Auswertungen sind möglich:

- Der Strom-/Spannungsverlauf an bis zu drei Messpunkten lässt sich in direktem Zusammenhang zu dem vom Prozessor ausgeführten Programm visualisieren.
- Der Energieverbrauch des Gesamtsystems kann für die einzelnen Funktionen aufgeschlüsselt werden.

Welcher Programmteil verursacht den größten Energieverbrauch? Welchen Einfluss hat eine Programmänderung auf den Energiebedarf eines Embedded Systems? Genau diese Fragen lassen sich jetzt auch mit der CombiProbe untersuchen.

Um den Energieverbrauch für jeden Punkt des Programms zu ermitteln, müssen folgende Messdaten erfasst werden:

- Der Programmablauf, der am Traceport des Prozessors ausgegeben wird.
- Der Strom- und Spannungsverlauf, der an geeigneten Messpunkten auf der Zielhardware abgegriffen wird.

Der Strom- und Spannungsverlauf für bis zu drei *Power Domains* lässt sich ab sofort durch den Anschluss einer *TRACE32 Analog Probe* an die CombiProbe ermitteln.

Da alle Messdaten über den globalen Timer der CombiProbe einen Zeitstempel erhalten, lässt sich der zeitliche Zusammenhang zwischen ausgeführtem Programmcode und der Stromaufnahme sowie dem Spannungsverlauf im System schnell und einfach darstellen.

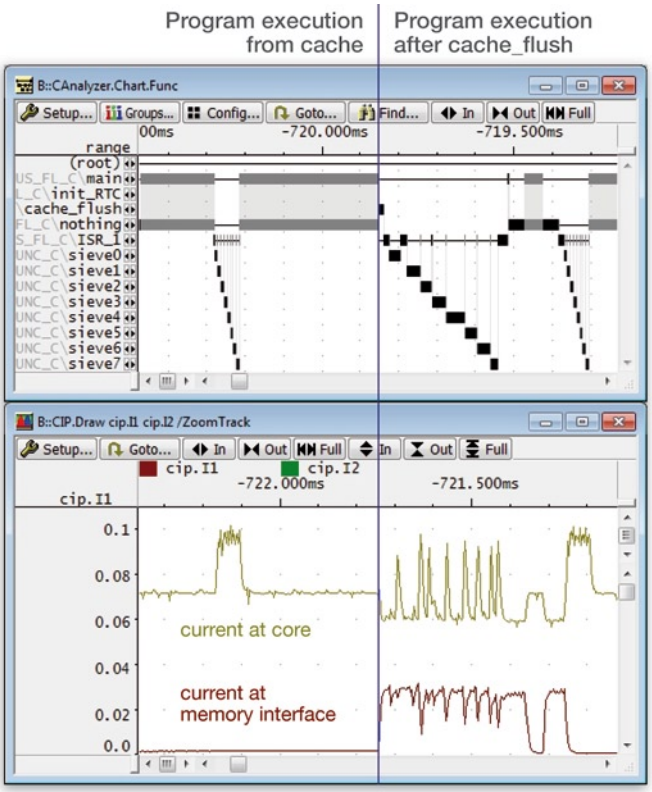

Bild 10: Ein Programmabschnitt, der nicht aus dem Cache läuft, benötigt mehr Zeit und verbraucht mehr Strom.

Bild 10 zeigt, dass ein Programmabschnitt der nicht aus dem Cache läuft, nicht nur eine wesentlich längere Abarbeitungszeit benötigt, sondern auch einen erhöhten Stromverbrauch am externen Speicher nach sich zieht.

Bild 11 zeigt den Energieverbrauch als statistische Auswertung. Dargestellt wird der minimale, maximale und durchschnittliche Energieverbrauch einzelner Funktionen.

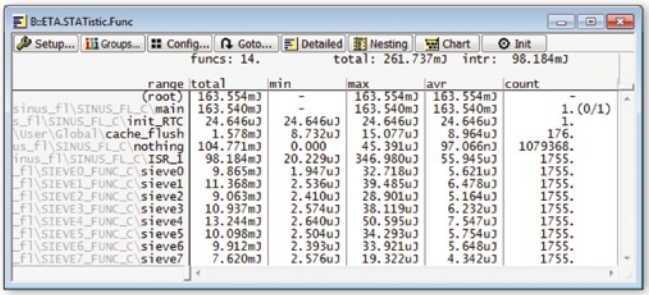

Bild 11: Der minimale, maximale und durchschnittliche Energieverbrauch einzelner Funktionen.

CombiProbe

Die CombiProbe ist ein Debug-Kabel, das zusätzlich einen 128 MByte großen Tracespeicher enthält. Die CombiProbe wurde speziell für Prozessoren mit einem 4-Bit breiten Traceport entwickelt. Die Aufzeichnung des Programmflusses wird aktuell für die folgenden Traceprotokolle unterstützt:

- ARM-ETMv3 im Continuous Mode (ARM)
- IFLOW Trace für PIC32 (Microchip)
- MCDS Trace für X-GOLD102 und X-GOLD110 (Infineon)

www.lauterbach.com/cobstm.html

Expertenforum – Im Austausch mit dem Anwender

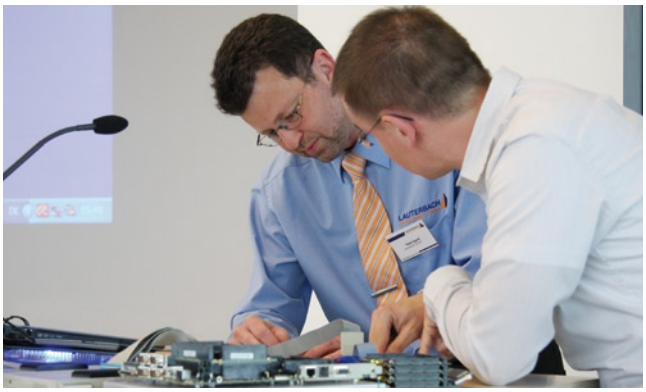

Bild 12: Technische Details werden im Fachgespräch geklärt.

Rückblick – Expertenforum ARM

Kunden und Interessenten hatten im vergangenen Jahr gleich zweimal die Möglichkeit, mit den TRACE32-Experten über das Debugging von ARM-Cores zu diskutieren. Da das Expertenforum am 5. Mai 2010 schnell ausgebucht war, entschlossen wir uns kurzfristig für einen Folgetermin am 24. und 25. November. Dieser wurde in englischer Sprache durchgeführt, um auch Kunden aus dem europäischen Ausland eine Teilnahme zu ermöglichen.

In beiden Veranstaltungen konnten sich die Teilnehmer über sämtliche Aspekte zum Debugging und Tracing von ARM-Cores informieren. Unsere Experten lieferten tiefe Einblicke in das Zusammenspiel der einzelnen Komponenten sowie zu speziellen Möglichkeiten der unterschiedlichen Cores. Das Themenspektrum reichte von den Anbindungsmöglichkeiten an die unterschiedlichen Debug- und Traceports bis hin zur optimalen Auswertung der Tracedaten.

Einen Schwerpunkt bildete die Betrachtung von Multicore-Systemen. Am Beispiel von Linux als Target-Betriebssystem wurden unterschiedliche Anforderungen aufgezeigt, die ein modernes Debugging-System bei heutigen Anwendungen bewältigen muss. Abgerundet wurde dieser Vortrag mit Ausblicken zum Thema Debugging von Android.

Ein besonderes Highlight stellte der Gastvortrag von Ian Johnson (ARM Ltd.) dar, der Details rund um die ARM-Prozessoren präsentierte, sowie einen interessanten Überblick zur zukünftigen Entwicklung gab.

Ausblick – Expertenforum Automotive

Das positive Feedback aller bisherigen Teilnehmer und die starke Nachfrage nach weiteren Terminen dieser Art haben uns gezeigt, wie wichtig eine Veranstaltung ist, bei der sich die Anwender und Entwickler unserer Debug-Systeme unmittelbar austauschen können.

Expertenforum Automotive am 18. Mai 2011

TRACE32 Product Overview

Development of AUTOSAR Systems / Components with TRACE32

TriCore Track	Power Architecture Track
• News • Concurrent Debugging of TriCore and PCP • MCDS / Trigger Language OCTL • Profiling • Q&A with TRACE32 Experts	• News • Multicore Debugging / Tracing • NEXUS-based Profiling • Q&A with TRACE32 Experts

Daher werden wir auch 2011 wieder Expertenforen veranstalten. Am 18. Mai 2011 wird sich alles um die spezifischen Debug-Lösungen im Automotive-Umfeld drehen. Es werden unter anderem die Prozessor-Familien TriCore von Infineon, MPC55xx/MPC56xx von Freescale sowie SPC56xx von ST genauer beleuchtet. Ein Schwerpunkt wird AUTOSAR sein. Einen Themenüberblick gibt die obige Agenda. Genauere Informationen stellen wir zeitnah auf unserer Homepage bereit.

BENACHRICHTIGEN SIE UNS

Falls sich Ihre Adresse geändert hat oder Sie kein Mailing mehr von uns erhalten möchten, schicken Sie bitte eine E-Mail an info@lauterbach.com.

WELTWEITE NIEDERLASSUNGEN

- **Deutschland**
- Frankreich
- Großbritannien
- Italien
- USA
- China
- Japan

In allen anderen Ländern vertreten durch kompetente Partner