TRACE32®
IN-CIRCUIT-EMULATOREN & DEBUGGER
ONE SYSTEM FITS ALL!

NEWS

2005

NEUHEITEN ZUR EMBEDDED WORLD 2005

Lauterbach setzt weiter auf Expansion

Wieder einmal konnte Lauterbach ein Geschäftsjahr als das erfolgreichste in der Firmengeschichte abschließen. Das Umsatzplus im Jahr 2004 betrug erneut 30%. Hierauf stützt sich unsere selbstbewusste Planung für das neue Jahr. Gleich zu Beginn gibt es auch schon wieder Neuigkeiten zu melden: seit dem 1. Januar 2005 unterhält Lauterbach ein eigenes Büro in Italien (siehe dazu auch Seite 12).

Was haben wir uns für das Jahr 2005 vorgenommen?

Obwohl nach unserer Auffassung für Entwicklungssysteme in naher Zukunft allgemein nur moderate Zuwächse zu erwarten sind, haben wir uns zum Ziel gesetzt, auch 2005 weiter stark zu wachsen. Eine Analyse unserer Märkte in den verschiedenen Weltregionen zeigt starke regionale Unterschiede. Dementsprechend werden wir für 2005 unsere strategischen Schwerpunkte setzen.

Nordamerika und Europa

In Nordamerika und Europa, wo Lauterbach schon seit vielen Jahren sehr gut etabliert ist, gehen wir von einem soliden aber moderaten Wachstum aus. Hier wird die technologische Weiterentwicklung unserer Produkte ausschlaggebend für den Erfolg in 2005 sein.

Asien

Asien wird nach unserer Einschätzung für Toolhersteller in nächster Zeit wohl die Region mit dem größten Wachstumspotential sein. Unsere positiven Erwartungen haben im wesentlichen zwei Gründe: Zum einen wird in Asien und speziell in China ein starkes allgemeines Wirtschaftswachstum erwartet. Zum anderen beobachten wir schon seit einiger Zeit, dass immer mehr Firmen aus dem Westen insbesondere aus Europa nun auch Software- und Hardware-Entwicklungsprojekte nach Asien verlagern, vornehmlich nach Indien und China.

Natürlich sind auch in Asien regionale Unterschiede zu berücksichtigen. In Japan etwa war der Markt für Entwicklungssysteme bei allgemeinem Nullwachstum schon unter lokalen Toolanbietern verteilt. Lauterbach konnte jedoch Marktanteile gewinnen und schon im zweiten Jahr seit der Gründung der eigenen Niederlassung in Yokohama arbeitet diese profitabel.

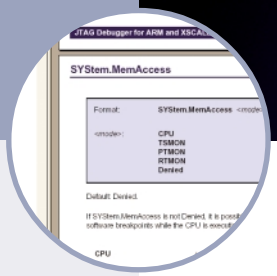
Anders ist die allgemeine Situation in China, dort werden jetzt die Weichen für die nächsten Jahre gestellt. Als europäische Firma können wir uns dort durch einen Technologievorsprung, hohe Qualität und die schnelle Unterstützung neuer Prozessoren sicher etablieren. Aber wir rechnen dort in Zukunft vermehrt mit lokalen Mitbewerbern, die ihren Produktionskostenvorteil zu niedrigen Angebotspreisen nutzen werden.

Während Lauterbach in Japan, Korea und Indien schon jetzt sehr stark präsent ist, wollen wir 2005 eine Niederlassung in China aufbauen.

*Aber nun zurück nach Europa und Deutschland. Auf der bevorstehenden **embedded world** in Nürnberg haben Sie Gelegenheit unsere Innovationskraft und die Leistungsfähigkeit der TRACE32-Power-Tools kennen zu lernen. Informationen dazu finden Sie auf den nächsten Seiten. Wir würden uns freuen, Sie auf unserem Messestand begrüßen zu dürfen.*

LAUTERBACH

Weitere Informationen im Internet: www.lauterbach.com



CFI Flash-Programmierung

Ab Mai 2005 wird Lauterbach eine automatische Generierung der Flash-Deklaration über das Common Flash Interface (CFI) unterstützen.

Um ein Flash zu programmieren, waren bisher umfassende Kenntnisse über den genauen Typ und die Organisation des Flash-Bausteins notwendig. Durch den speziellen Query-Modus CFI konformer Flash-Bausteine lassen sich alle für die Flash-Programmierung benötigten Parameter automatisch erfragen. Mit diesen Daten kann TRACE32 zukünftig automatisch eine Flash-Deklaration erstellen. Der Anwender muss dazu nur noch die beiden folgenden Parameter angeben:

- *die Startadresse des Flash-Bausteins.*
- *die Datenbus-Breite, über die der Flash-Baustein an den Mikrocontroller angeschlossen ist.*

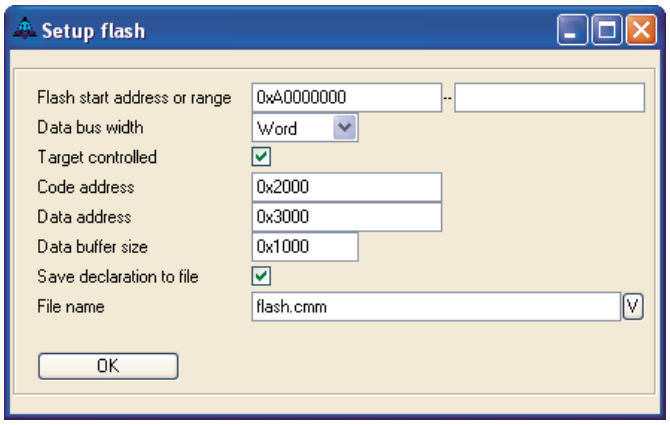
Selbstverständlich lässt sich die so generierte Flash-Deklaration in ein ablauffähiges Skript konvertieren.

TRACE32 wird CFI für die beiden folgenden Flash-Programmiermethoden unterstützen:

***TRACE32 Tool basierte Flash-Programmierung:** Bei dieser Programmiermethode wird die Flash-Programmierung vollständig über die TRACE32-Software abgewickelt, dadurch werden keine extra Ressourcen auf dem Zielsystem benötigt. Dieses Verfahren ist sehr gut für den Einstieg geeignet, da es keine zusätzliche Parametrisierung erfordert.*

***Target kontrollierte Flash-Programmierung:** Die Stärke dieses Ansatzes liegt darin, dass der für die Flash-Programmierung verwendete Algorithmus auf dem Target läuft und somit wesentlich weniger Kommunikation mit der TRACE32-Software auf dem Host erforderlich ist. Target kontrolliertes Flash-Programmieren ist deshalb etwa um den Faktor 20 schneller als die Tool basierte Methode. Dieser Ansatz erfordert die Deklaration von RAM Bereichen auf dem Zielsystem für:*

- *den Flash-Algorithmus*
- *den Datenaustausch zwischen der TRACE32-Software und dem Flash-Algorithmus*



Immer wenn eine Operation auf dem Flash-Baustein durchgeführt werden soll, lädt TRACE32 den Flash-Algorithmus in das RAM des Zielsystems und führt ihn dort aus.

Die Puffergröße legt fest, wie viele Daten mit einem Aufruf des Flash-Algorithmus programmiert werden.

Die Binär-Files der Flash-Algorithmen für die meisten Prozessorarchitekturen und Flash-Bausteine werden von Lauterbach auf der TRACE32 Software-CD zur Verfügung gestellt.

Eine aktuelle Liste der von TRACE32 unterstützten Flash-Bausteine finden Sie unter:
http://www.lauterbach.com/ylist.html

Neu unterstützte Echtzeitkerne

- ***LINUX 2.6** für ARM, MIPS, PowerPC, SH4 und XScale*
- ***QNX 6.3** (QNX Software Systems) für ARM, PowerPC, SH4 und XScale*
- ***Quadros** (Quadros Inc.) für TMS320C55x*
- ***Symbian OS V8.0a** und **Symbian OS EKA2** (Symbian) für ARM*
- ***ThreadX** (Express Logic) für StarCore*
- ***Windows CE** (Microsoft) für SH4*
- ***Windows CE 4.x** (Microsoft) eXDI Driver für ARM, SH4 und XScale*
- ***Windows CE 5.0** (Microsoft) für ARM, SH4 und XScale*
- ***µIPLUS** (Accelerated Technology) für ARM und PowerPC*

Cache-Analyse

Lauterbach bietet für sein Mikroprozessor-Entwicklungswerkzeug TRACE32-PowerTrace seit Oktober 2004 eine Cache-Analyse an.

Ein Cache ist ein kleiner, schneller Speicher, der meist direkt in den Mikrocontroller integriert ist. Damit der Mikrocontroller nicht jede Programminstruktion bzw. jeden Datenwert aus dem langsamen externen Speicher holen muss, werden einzelne Programmteile bzw. einzelne Daten im Cache zwischengespeichert. Da der Cache aber nur ein kleiner Speicher (4 KB bis 128 KB) ist, kann immer nur ein Teil der benötigten Programminstruktionen bzw. Daten im Cache gehalten werden. Unterschiedliche Teile des Programms konkurrieren also um einen Platz im Cache und verdrängen sich gegenseitig. Dabei bestimmen die Speicheradresse und die Cachearchitektur, wer einen Platz im Cache erhält und wer dazu aus dem Cache verdrängt wird. Jedes Verdrängen bedeutet jedoch, dass die verdrängten Cacheinhalte wieder in den Cache geladen werden müssen, wenn das Programm sie erneut benötigt. Die dazu notwendigen Zugriffe auf den externen Speicher sollen aber aus den folgenden Gründen möglichst vermieden werden:

- *Aufgrund der aktuellen Zugriffszeiten für externe Speicher kann davon ausgegangen werden, dass die Bereitstellung einer Instruktion bzw. eines Datenwertes aus dem externen Speicher in der Regel 10- bis 30-mal langsamer erfolgt, als wenn sich das Datum bereits im Cache befindet. Jeder Zugriff auf den externen Speicher verlangsamt also die Programmausführung um ein Vielfaches.*

  Die tatsächlichen Zugriffzeiten auf die einzelnen Cachespeicher sowie auf den externen Speicher des Zielsystems kann TRACE32 über ein Speicher-Benchmarking ermitteln (siehe Bild 1).

B::d.benchmark

max size: 0x200000 dhrystones/sec: 197430 tolerance:

	block read	block write	block copy	random read	random write	random copy	latency near
IC	494.4MB/s	-	-	-	-	-	-
DC	493.5MB/s	494.4MB/s	232.1MB/s	123.3MHz	123.7MHz	30.9MHz	16.168ns
L2							
L3							
MEM	85.83MB/s	98.13MB/s	39.76MB/s	4.821MHz	12.96MHz	3.339MHz	172.615ns

Bild 1: Mit einem Benchmark-Testprogramm ermittelt TRACE32 die Zugriffsraten für alle Cachespeicher und für den externen Speicher

- *Bei jedem Zugriff auf den externen Speicher steigt der Stromverbrauch des Gesamtsystems sofort erheblich an. Dies ist vor allem für tragbare, batteriebetriebene Geräte ein großer Nachteil.*

Zugriffe auf den externen Speicher lassen sich vor allem dadurch reduzieren, dass unnötige Verdrängungen aus dem Cache vermieden werden. Unnötige Verdrängung entsteht immer dann, wenn viele Programminstruktionen bzw. Daten um einzelne, wenige Cachezeilen konkurrieren, während andere Cachezeilen kaum oder gar nicht genutzt werden (siehe Bild 2).

B::CTS.CACHE.ListLine DC

Params... Config... Addresses Functions Variables Lines

POS: 0.

address	cached	hits		misses	unknown	victims
DC:00000820	304536.	190261.	62.475%	114273.	2.	114273.
DC:000008C0	266765.	152492.	57.163%	114271.	2.	114271.
DC:000007E0	266579.	152337.	57.145%	114240.	2.	114240.
DC:000007F0	266570.	152375.	57.161%	114193.	2.	114193.
DC:00000800	190658.	76474.	40.110%	114182.	2.	114182.
DC:00000810	266548.	189996.	71.280%	76550.	2.	76550.
DC:00000870	116103.	114895.	98.959%	1206.	2.	1206.
DC:00000840	152751.	151863.	99.418%	886.	2.	886.
DC:00000830	114857.	114178.	99.408%	677.	2.	677.
DC:00000770	77021.	76774.	99.679%	245.	2.	245.
DC:00000E80	77199.	76954.	99.682%	243.	2.	243.
DC:00000890	39135.	38897.	99.391%	236.	2.	236.
DC:00000090	1484.	1248.	84.097%	234.	2.	234.
DC:000000B0	1459.	1224.	83.893%	233.	2.	233.
DC:000007D0	76850.	76616.	99.695%	232.	2.	232.
DC:000009A0	76825.	76591.	99.695%	232.	2.	232.
DC:00000BF0	39046.	38815.	99.408%	229.	2.	229.

Bild 2: Viele Speicheradressen konkurrieren um einige wenige Cachezeilen, während andere Cachezeilen kaum genutzt werden

Die Effizienz des Cache lässt sich optimieren, indem man über die Relocation Information im Linker Command-File dafür sorgt, dass Funktionen und Daten so im Speicher platziert werden, dass es zu einer möglichst gleich verteilten Konkurrenz um alle Cachezeilen kommt (siehe Bild 3).

B::CTS.CACHE.ListLine DC

Params... Config... Addresses Functions Variables Lines

POS: 0.

address	cached	hits		misses	unknown	victims
DC:00000770	77021.	76774.	99.679%	245.	2.	245.
DC:00000E80	77199.	76954.	99.682%	243.	2.	243.
DC:00000890	39135.	38897.	99.391%	236.	2.	236.
DC:00000090	1484.	1248.	84.097%	234.	2.	234.
DC:000000B0	1459.	1224.	83.893%	233.	2.	233.
DC:000007D0	76850.	76616.	99.695%	232.	2.	232.
DC:000009A0	76825.	76591.	99.695%	232.	2.	232.
DC:00000BF0	39046.	38815.	99.408%	229.	2.	229.
DC:00000D20	1501.	1270.	84.610%	229.	2.	229.
DC:00000D0	1184.	954.	80.574%	228.	2.	228.
DC:00000E70	152809.	152579.	99.849%	228.	2.	228.
DC:00000BD0	1096.	868.	79.197%	226.	2.	226.
DC:00000B80	2457.	2231.	90.801%	224.	2.	224.
DC:00000C0	1253.	1028.	82.043%	223.	2.	223.
DC:00000A0	1458.	1234.	84.636%	222.	2.	222.
DC:00000A00	1119.	895.	79.982%	222.	2.	222.
DC:00000F40	1079.	855.	79.240%	222.	2.	222.

Bild 3: Die Konkurrenz um die Cachezeilen ist gleich verteilt

Mit seiner Cache-Analyse bietet Lauterbach dem Entwickler nun ein Analyse-Werkzeug, dass ihm hilft unnötige Verdrängungen aufzuspüren und die Cache-Effizienz so zu optimieren.

Die Durchführung

Bevor die Cache-Analyse im Folgenden anhand eines Beispiels beschrieben wird, einige Begriffsdefinitionen:

***Cache Hit:** Eine vom Mikrocontroller benötigte Programminstruktion bzw. ein Datenwert befindet sich im Cache.*

Cache Miss: *Eine vom Mikrocontroller benötigte Programminstruktion bzw. ein Datenwert befindet sich nicht im Cache und muss deshalb aus dem externen Speicher geladen werden.*

Cache Victim: *Eine Programminstruktion bzw. ein Datenwert wurde aus dem Cache verdrängt, um nach einem Cache Miss Platz für die benötigte Programminstruktion bzw. den Datenwert zu schaffen.*

Cachezeile: *Der Cache ist zeilenweise organisiert. Eine Cachezeile umfasst 8, 16 oder 32 Byte als ein Vielfaches der Busbreite des Mikrocontrollers. Bei einem Cache Miss werden die neuen Inhalte blockweise in die Cachezeile geladen (Burst Zugriffe auf den externen Speicher).*

Um mit einem TRACE32-PowerTrace für eine bestimmte Systemfunktion (z.B. eine Bildkompression in einem Handy) eine Cache-Analyse durchzuführen, sind folgende Schritte notwendig:

1. Vordefinition der Cache-Struktur

Die Cache-Struktur des im Zielsystem verwendeten Mikrocontrollers wird von der TRACE32-Software automatisch definiert.

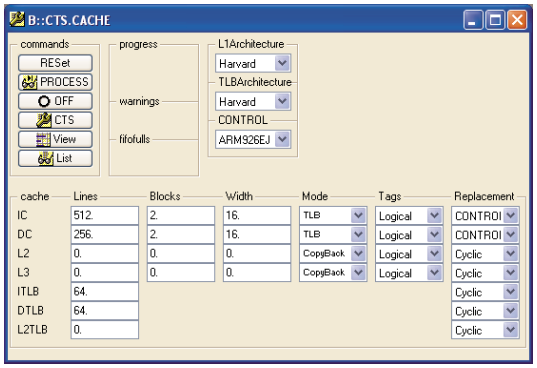


Bild 4: Automatische Definition der Cache-Struktur für den eingesetzten Mikrocontroller

Dabei werden alle auf dem Markt gängigen Cache-Strukturen und Cache-Hierarchien, also auch Level 2 / Level 3 Caches, unterstützt. Insbesondere erlaubt TRACE32 auch Cache-Strukturen, die durch eine Memory Protection Unit (MPU) bzw. eine Memory Management Unit (MMU) konfiguriert werden (siehe Bild 4).

2. Aufzeichnung des Programm- und Datenflusses dieser Funktion in den Tracespeicher

Die Basis für die Cache-Analyse bildet der im Tracespeicher aufgezeichnete Programm- und Datenfluss. TRACE32-PowerTrace stellt einen 128 MFrame großen Tracespeicher zur Verfügung, der beispielsweise für die ARM-ETM eine Aufzeichnung von bis zu 10 Sekunden der Programmlaufzeit erlaubt.

3. Durchführung der Cache-Analyse

Die Cache-Analyse ermittelt zunächst die Cache Hit/Cache Miss/Cache Victim Rate für den aufgezeichneten Programm- und Datenfluss basierend auf der definierten Cachestruktur (siehe Bild 5).

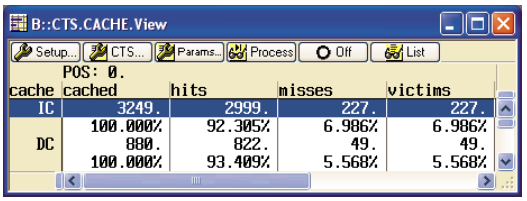


Bild 5: Analyse der Cache Hit / Cache Miss / Cache Victim Rate für den Instruction Cache (IC) und den Daten Cache (DC)

Um diese Information zur Reduktion der Cache Victims zu nutzen, wird folgende Detailinformation benötigt:

- *Für welche Cachezeilen ist die Cache Victim Rate besonders hoch?*
- *Welche Instruktionen konkurrieren um eine solche Cachezeile?*
- *Gibt es Cachezeilen die nicht bzw. kaum genutzt sind?*

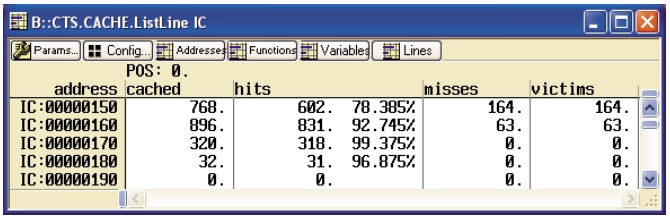


Bild 6: Analyse der Cache Hit / Cache Miss / Cache Victim Rate für die einzelnen Zeilen des Instruction Cache

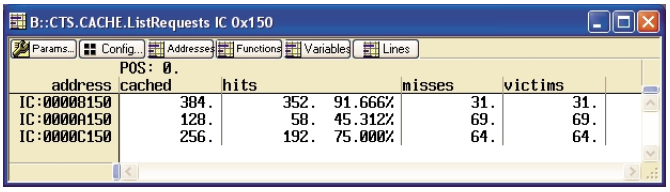


Bild 7: Auflistung der Programmadressen, die um die Cachezeile 0x15 des Instruction Cache konkurrieren

Die Analyse zeigt in Bild 7, dass die Programminstruktionen auf Adresse 0x8150, 0xA150 und 0xC150 um die Cachezeile 0x15 konkurrieren. Da im obigen Beispiel ein Cache verwendet wurde, bei dem jede Cachezeile 2-fach vorhanden ist (2-fach assoziativer Cache), kommt es bei einer Konkurrenz von drei Programmadressen um eine Cachezeile immer wieder zu Verdrängungen und damit zu Cache Victims. Die Cache Line 0x19 ist dagegen vollkommen ungenutzt. Durch eine Adressverschiebung einer der konkurrierenden Programminstruktionen auf die Endadresse 0x190 kann die Cache Line 0x15 so entlastet werden, dass keine unnötigen Cache Victims mehr entstehen.

Der Zusammenhang zwischen Cache-Effizienz und Programmlaufzeit

Wie im vorhergehenden Abschnitt beschrieben, beruht die Cache-Analyse auf dem im Tracespeicher aufgezeichneten Programm- und Datenfluss. Da die Einträge im Tracespeicher immer mit einem Zeitstempel versehen sind, kann der Traceinhalt auch als Basis für Laufzeitmessungen verwendet werden. Damit bietet sich der große Vorteil, dass der direkte Zusammenhang zwischen Programmlaufzeit und Cache-Effizienz mit TRACE32-PowerTrace mess- und verifizierbar ist.

Wie aus Bild 6 ersichtlich, hat die analysierte Funktion für die Cachezeile 0x15 ein Cache Hit Rate von 78% sowie eine Cache Victim Rate von 22%.

Die hohe Verdrängungsrate macht sich natürlich auch unmittelbar in der Laufzeit der Funktion bemerkbar. Die durchschnittliche Funktionslaufzeit der Funktion (sievebad) beträgt 77,6 us (siehe Bild 9).

Durch eine Adressverschiebung einer der konkurrierenden Programminstruktionen auf die Endadresse 0x190 kann die Cachezeile 0x15 entlastet werden. Da nun nur noch zwei Adressen um diese Cachezeile konkurrieren, kommt es zu keiner Verdrängung mehr. Eine Analyse der optimierten Funktion (sievegood) ergibt sofort ein wesentlich besseres Laufzeitverhalten. Die Funktionslaufzeit beträgt im Schnitt nur noch 7,9 us (siehe Bild 10).

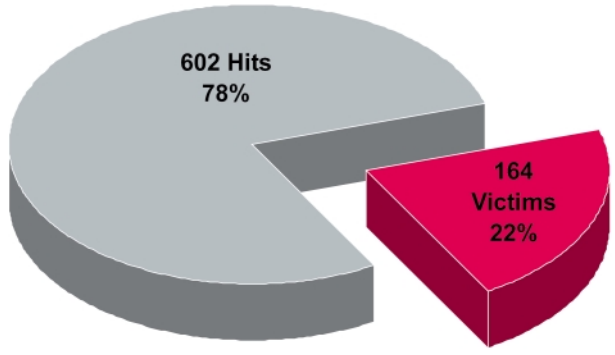


Bild 8: Graphische Analyse der Cache Hit/Cache Victim Rate für die Cachezeile 0x15.

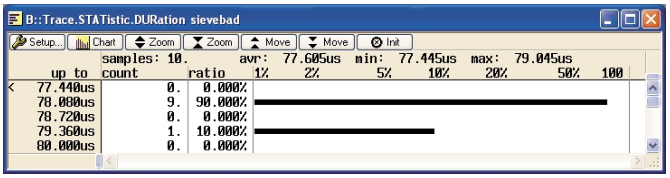


Bild 9: Laufzeitanalyse der Funktion sievebad mit TRACE32-PowerTrace

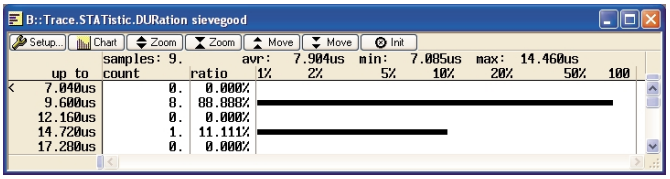


Bild 10: Laufzeitanalyse der Funktion sievegood mit TRACE32-PowerTrace

Durch die Kombination aus Cache-Analyse und Laufzeitmessung ermöglicht TRACE32-PowerTrace unmittelbar nachzuprüfen, ob eine durchgeführte Optimierung der Cache-Effizienz auch wirklich zu der erwarteten Laufzeitverbesserung führt. Umgekehrt lässt sich natürlich auch einfach untersuchen, ob sich Codeoptimierungen ungünstig auf die Cache-Effizienz auswirken und damit keine Verbesserung der Programmlaufzeit zur Folge haben.

Zusammenfassung

Mit seiner neuen Cache-Analyse bietet Lauterbach den Entwicklern von Embedded Designs folgende Möglichkeiten:

- *Optimierung des Laufzeitverhalten ihres Programms*
- *Reduktion des Stromverbrauchs im Gesamtsystem*

Die Cache-Analyse ist offen für alle Mikrocontroller, Cache-Architekturen und Cache-Hierarchien. Die für den Mikrocontroller vordefinierte Cache-Struktur kann bei Bedarf auch geändert werden. So lässt sich beispielsweise ausprobieren, ob eine andere Cache-Struktur generell zu einer besseren Cache-Effizienz und damit zu kürzeren Programmlaufzeiten führen würde. Das Ergebnis kann dann für zukünftige Designs berücksichtigt werden.

Die Cache-Analyse wurde in enger Kooperation mit führenden Mobilfunk-Herstellern entwickelt. Die Firma Lauterbach stellt damit erneut ihre Vorreiterrolle auf dem Entwicklungswerkzeuge-Markt unter Beweis.

Mehrfach-Lizenzen im Debug Kabel

Seit Jahresbeginn 2004 kann ein TRACE32 Debug Kabel mehrere Lizenzen enthalten. Im Folgenden soll nun eine kurze Übersicht über die Einsatzmöglichkeiten von Mehrfach-Lizenzen und ihre Handhabung gegeben werden.

Zur Verwaltung der Mehrfach-Lizenzen ist im Debug Kabel eine Lizenz-Matrix (4 Spalten und 3 Reihen) abgelegt. Mehrfach-Lizenzen wurden für folgende Zwecke eingeführt:

Neue Prozessorgenerationen

Ein Kunde, der in seinem aktuellen Projekt beispielsweise ein TRACE32 Debug Kabel für den ARM9 im Einsatz hat, kann dieses Debug Kabel mit einer so genannten eXtension License für das Debugging eines ARM11 aufrüsten.

ARM7	ARM9	ARM11	Cortex

Beispiel für Mehrfach-Lizenzen innerhalb der ARM Architektur

Eine solche Aufrüstung innerhalb der gleichen Prozessorarchitektur ist immer dann möglich, wenn die neuen Prozessorgeneration physikalisch die gleiche Debug-Schnittstelle verwendet.

Komplexer Onchip-Tracespeicher

Einige Prozessorarchitekturen bieten große Onchip-Tracespeicher, die dann meist auch über komplexe Filter und Triggermöglichkeiten verfügen. Diese Onchip-Tracespeicher werden in der Regel über die Debug-Schnittstelle konfiguriert und ausgelesen. Zur Freischaltung der Tracekonfiguration sowie der Traceauswertung durch die TRACE32-Software ist eine eXtension License notwendig.

ARM11	ETB Trace Support		

Für die Konfiguration des Embedded Trace Buffer (kurz ETB) und die Auswertung des Traceinhalts wird eine eXtension License benötigt

Neben dem Embedded Trace Buffer (ETB) von ARM erfordern folgende komplexe Onchip-Tracespeicher eine eXtension License.

- *EJTAG Onchip-Trace für die MIPS Architektur*
- *Onchip-Trace des TriCore TC1796ED von Infineon*

Multicore Debugging

Der Test und die Systemintegration von Multicore-Designs kann ab Frühjahr 2005 mit einer gemeinsamen Debugger-Hardware durchgeführt werden. Ein Debug Kabel bedient dabei die gemeinsame Onchip-Debug-Schnittstelle aller Cores. Siehe dazu auch Seite 7.

- ***Multicore-Designs mit identischen Cores***

  Für Multicore-Designs, die aus identischen Cores aufgebaut sind, ist eine Lizenz für die Prozessorarchitektur sowie eine Multicore-Lizenz erforderlich. So genügt beispielsweise für das Debugging des MSC8102 von Motorola, der vier StarCore DSPs enthält, ein Debug Kabel, das eine StarCore- und eine Multicore-Lizenz enthält.

StarCore	Multicore Lizenz		

Beispiel für Mehrfach-Lizenzen für ein Multicore-Design mit identischen Cores, hier für ein Design aus vier StarCore DSPs

- ***Multicore-Designs mit unterschiedlichen Cores***

  Für Multicore-Designs, bei denen ein RISC Prozessor die Systemsteuerung und ein Signalprozessor die Datenverarbeitung übernimmt, müssen im Debug Kabel Lizenzen für beide Architekturen eingetragen sein. Die Lizenz für den DSP wird dabei als Additional License bezeichnet.

ARM9			
Teak/TeakLite für JAM Interface			

Beispiel für Mehrfach-Lizenzen für ein Multicore-Design mit unterschiedlichen Cores, hier für ein Design aus ARM9 und TeakLite DSP

Grundsätzlich gilt: *Das Multicore Debugging ist immer dann freigeschaltet, wenn das Debug Kabel mehr als eine Lizenz enthält.*

Neue Konzepte für das Multicore Debugging

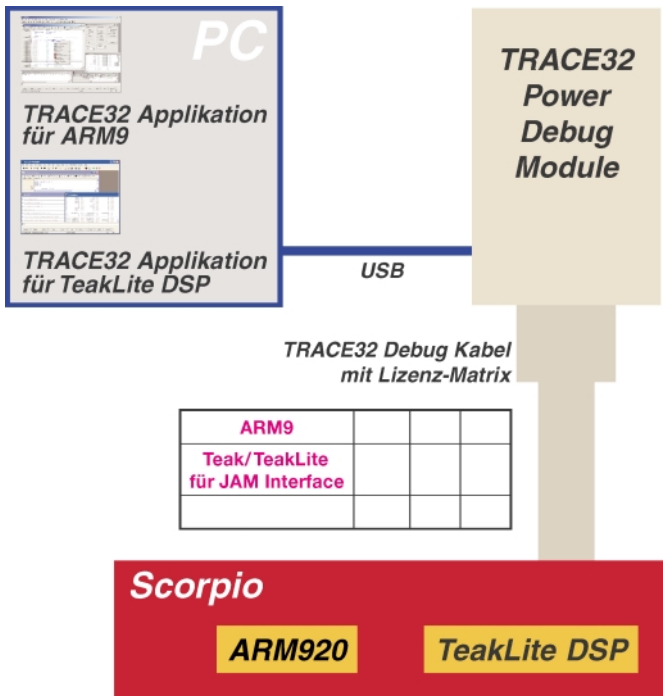
Seit 3 Jahren unterstützt Lauterbach mit seinen In-Circuit Debuggern TRACE32-ICD das Debugging von Multicore SoCs. Aufgrund der Erfahrungen in zahlreichen Kundenprojekten wurde das Lauterbach Multicore Debugging Konzept nun weiterentwickelt.

Um eine optimale Funktionalität und Performance zu erreichen, werden immer häufiger mehrere Cores zu einem System-on-Chip (SoC) integriert. Weit verbreitet ist das Zusammengehen eines RISC Prozessors mit einem oder mehreren DSPs, aber auch andere Kombinationen kommen zum Einsatz. Bei Multicore SoC Designs wird in der Regel, um Pins und Kosten zu sparen, nur eine Debug-Schnittstelle für alle Cores zur Verfügung gestellt.

Ansteuerung mehrerer Cores über eine gemeinsame Debug-Schnittstelle

Mit dem neuen Konzept genügt für das Debugging aller Cores eines SoC ein Power Debug Module und ein Debug Kabel. Dabei muss das Debug Kabel Lizenzen für alle eingesetzten Cores bzw. eine Multicore-Lizenz enthalten.

Für das Debugging jedes Cores läuft auf dem Host eine separate TRACE32 Applikation. Die gemeinsame TRACE32-Systemsoftware im Power Debug Module sorgt nun dafür, dass die von einer Applikation abgesetzten Debug-Kommandos an den zugehörigen Core weitergeleitet werden.



Debugging des ARM920 und des TeakLite DSP im Scorpio über eine gemeinsame TRACE32-ICD Hardware

Zur Veranschaulichung dieses Konzepts zwei Beispiele:

Scorpio: *Der Scorpio von Samsung ist ein SoC, der einen ARM920 und einen TeakLite DSP enthält. Für das gleichzeitige Debugging beider Cores muss das Debug Kabel eine Lizenz für den ARM9 und eine Lizenz für den TeakLite enthalten (siehe Bild).*

MSC8102: *Der MSC8102 ist ein SoC von Motorola, der vier StarCore DSPs enthält. Für das gleichzeitige Debugging aller vier DSPs genügt es, dass im Debug Kabel eine Lizenz für den StarCore sowie eine Multicore-Lizenz eingetragen ist.*

Start-/Stopp-Synchronisation

Ein wichtiges Thema beim gleichzeitigen Debugging mehrerer Cores ist natürlich die Start- und Stopp-Synchronisation. Beides funktioniert nur dann, wenn sowohl das synchrone Starten und als auch das synchrone Stoppen vom SoC unterstützt wird. Sind die einzelnen Cores beispielsweise über einen Breakswitch miteinander gekoppelt, lässt sich ein nahezu clocksynchrones Anhalten aller Cores realisieren. Ein programmierbarer Breakswitch erlaubt dem Anwender zudem selbst zu konfigurieren, welche Cores des SoC sich gegenseitig synchron stoppen sollen. Unterstützt der SoC selbst kein synchrones Anhalten und gibt es auch im Zielsystem physikalisch keine Möglichkeit, dass sich die Cores gegenseitig stoppen, kann auch der Debugger nur eine zeitnahe Stopp-Synchronisation realisieren. Gleiches gilt auch für die Start-Synchronisation: Lassen sich die einzelnen Cores des SoC über ihre Debugging-Logik gemeinsam starten (beispielsweise über ein spezielles JTAG Kommando), kann der Debugger eine Start-Synchronisation realisieren. Andernfalls kann er nur versuchen, die einzelnen Cores schnellstmöglich nacheinander zu starten.

Gemeinsamer Traceport

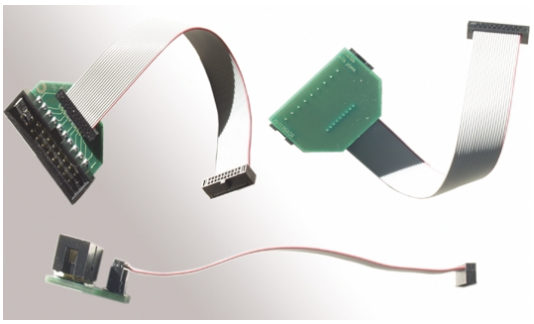
Viele Entwickler wollen auch bei einem Multicore SoC Design nicht auf die Vorteile eines Echtzeit-Traces verzichten. Natürlich wird auch hier aus Kostengründen an den Pins für den Traceport gespart. Inzwischen gibt es die ersten SoC Designs, bei denen sich mehrere Cores einen Traceport teilen. In den aktuellen Designs kann der Entwickler einstellen, welcher Core den Traceport exklusiv nutzen darf. Im Gespräch sind aber auch SoCs bei denen mehrere Cores gleichzeitig einen Traceport bedienen (z.B. im CoreSight Konzept von ARM oder bei NEXUS).

Aktuelles zu den TRACE32-ICD Debuggern

Halfsize Adapter

Seit Jahresbeginn 2004 bietet Lauterbach so genannte Halfsize Adapter an.

Die Stecker an den TRACE32 Debug Kabeln erfordern in der Regel Standardpfostenleisten (100 mil Pinabstand, 100 mil Pinreihenabstand) auf dem Zielsystem. Die neuen Halfsize Adapter erlauben nun auch Pfostenleisten mit 50 mil Pinabstand und 50 mil Pinreihenabstand zu verwenden. Damit lässt sich der Platzbedarf für den Debugger-Anschluss auf dem Zielsystem auf etwa 1/4 reduzieren.



Halfsize Adapter für die Debug Kabel

Halfsize Adapter stehen für alle 14-, 16- und 20-poligen Debug Kabel zur Verfügung. Die Länge des Verbindungskabels beträgt 10 cm.

http://www.lauterbach.com/adhalfsize.html

Optimierung der Download-Zeiten

Die Download-Zeiten konnten für die folgenden Prozessor-Familien erheblich verbessert werden:

- *Für den **PowerQUICC III** lässt sich bei einer JTAG-Clock von 50 MHz eine Download-Rate von 4,3 MByte/s erreichen.*
- *Für den **MPC55xx** lässt sich bei einer Core-Clock von 120 MHz und einer JTAG-Clock von 40 MHz eine Download-Rate von 2,2 MByte/s erreichen.*
- *Für den **ARM11** lässt sich bei einer JTAG Clock von 50 MHz eine Download-Rate von 3 MByte/s erzielen.*

Die schnelleren Download-Zeiten wurden durch Software-Optimierungen und durch eine verbesserte Hardware des Debug Kabels möglich.

Neue unterstützte Prozessoren

Altera	NIOS II	sofort
AMCC	PPC440EP	sofort
ARM	Cortex-M Cortex-A Cortex-R	Q2/05 Q3/05 Q3/05
CEVA	CEVA-X 1620	Q1/05
freescale	S12X MPC521x MPC7448 MPC83xx MPC8541/MPC8555 MPC8548	sofort Q1/05 Q1/05 Q1/05 sofort Q2/05
IBM	PPC970FX STB02500	Q2/05 Q1/05
Intel	IXP23xx (XScale)	Q1/05
Motorola	MSC711x (StarCore)	sofort
NEC	V4133 VR55xx V850 (N-Wire)	Q1/05 Q1/05 sofort
Renesas	M32R	Q1/05
ST	Nomadic (ARM, MMDSP) NEXUS-MMDSP	Q1/05 Q1/05
Texas Instruments	TMS320C2x TMS320C54x TMS320C6x	Q2/05 Q1/05 sofort
XILINX	Virtex-II Pro (PPC405) als Single- und Multi-Core Design	sofort
ZSP	ZSP500	Q2/05

Tabelle der neu unterstützten Prozessoren

Aktuelles für die ARM Architektur

Neue JTAG Clock Modi für ARM Prozessoren

Lauterbach unterstützt für seine ARM-Debugger einige neue JTAG Clock Modi. Diese neuen JTAG Clock Modi ermöglichen es, die Debug-Schnittstelle mit wesentlich höheren Frequenzen zu takten.

Höhere JTAG Frequenzen erlauben beispielsweise:

- *Ein schnelleres Download von Code/Daten in den Zielsystemspeicher.*
- *Ein schnelleres Upload der Daten aus dem Onchip Tracespeicher (ETB) auf das Hostsystem.*

Für alle ARM Designs, bei denen der JTAG Clock unabhängig von der CPU Clock laufen kann, war bisher eine JTAG Frequenz von max. 25 MHz möglich. Es gibt nun folgende neue JTAG Clock Modi:

CTCK: *Mit dem neuen Debug Kabel, das seit Jahresbeginn 2004 ausgeliefert wird, kann dieser neue JTAG Clock Mode verwendet werden. Er erlaubt die JTAG Clock auf eine Frequenz von bis zu 35 MHz hochzusetzen.*

CRTCK: *Ist TCK auf dem Zielsystem direkt oder über einen Treiber auf RTCK zurückgeführt, kann der JTAG Clock Mode CRTCK eingestellt werden. Dieser Mode ermöglicht eine JTAG Clock von bis zu 50 MHz.*

Synthetische ARM Cores (ARMxxx-S) müssen mit RTCK als JTAG Clock arbeiten. RTCK muss immer dann verwendet werden, wenn eine Synchronisation zwischen der JTAG Clock und der CPU Clock erforderlich ist. Mit dem JTAG Clock Modus RTCK arbeitet die JTAG Clock mit höchstens einem Sechstel der CPU Clock. Die max. mögliche Frequenz beträgt hier 20 MHz.

ARTCK: *Der neue JTAG Clock Mode ARTCK verwendet nun einen speziellen Beschleunigungsmodus und ermöglicht so eine JTAG Clock, die maximal halb so schnell wie die CPU Clock ist. Bestenfalls können so 50 MHz erreicht werden.*

Unterstützung für NEON Technologie

Mit seiner NEON Technologie bietet ARM einen umfangreichen Befehlssatz für Embedded Designs, deren Schwerpunkt im Bereich Multimedia und Signalverarbeitung liegt. Sobald der erste ARM Prozessor mit NEON Technologie verfügbar ist, wird diese Technik auch von Lauterbach unterstützt werden.

Support für CoreSight

CoreSight ist der neue ARM-Standard für das Debuggen und Tracen von Multicore SoCs. CoreSight umfasst viele neue Konzepte wie eine Single-Wire Debug-Schnittstelle, Variablen-Monitoring während des Programmlaufs, gleichzeitiges Tracen von verschiedenen Cores, Prozessorbussen und Peripheriekomponenten etc. Selbstverständlich wird Lauterbach all diese neuen Konzepte mit seinen Entwicklungswerkzeugen TRACE32-ICD und PowerTrace unterstützen.

Neuer Debugger für V850E/V850ES Core

Lauterbach unterstützt ab sofort mit seinem Debugger TRACE32-ICD und seinem PowerTrace Entwicklungswerkzeug alle NEC Prozessoren mit V850E und V850ES Core.

Über die N-Wire Schnittstelle des V850 bietet der Debugger TRACE32-ICD ein komfortables Assembler- und Hochsprachen-Debugging für alle Standard-Compiler. Selbstverständlich werden dabei auch alle Onchip Break- und Triggermöglichkeiten unterstützt.

TRACE32-PowerTrace ermöglicht für alle Derivate mit erweitertem N-Wire Interface die Aufzeichnung des Programm- und Datenflusses in einen 512 MByte großen Tracespeicher. Diese Laufzeitinformation erlaubt eine strategische Fehlersuche sowie umfassende Performance-Analysen.

Bestimmte Typen der V850 Familie verfügen zudem über ein so genanntes NBD Interface. Über dieses Interface kann

Speicher ausgelesen bzw. beschrieben werden, während das Programm in Echtzeit läuft. Für diese Funktionalität bietet Lauterbach eine kleine Zusatz-Hardware an (siehe Bild).

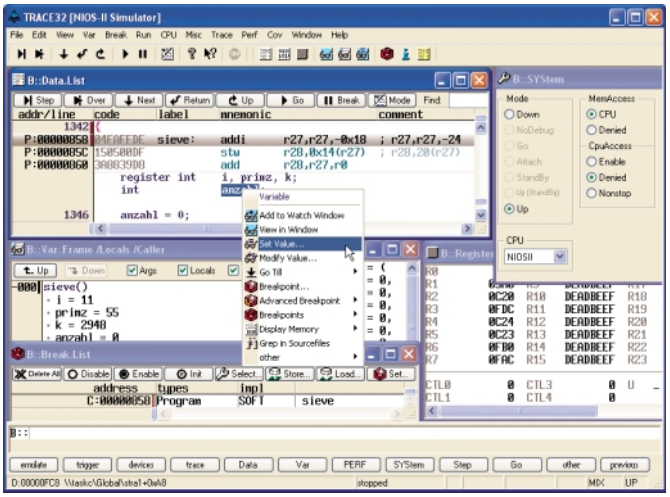
Debugger für NIOS II

Lauterbach bietet zum Jahresbeginn 2005 einen Debugger für den Softcore NIOS II von Altera an.

NIOS II ist ein 16-/32-Bit RISC Softcore. Zusammen mit anwendungsspezifischen Peripheriekomponenten kann mit NIOS II einfach ein SOPC (System On a Programmable Chip) entworfen werden. Dabei ist auch die Integration einer Onchip Debugging- und Trace-Logik möglich.

Die Onchip Debugging-Logik erlaubt ein komfortables Debuggen auf Assembler- und Hochsprachenebene für den GNUPro C/C++ Compiler. Da die Onchip Debug-Schnittstelle den gleichen 10-poligen Stecker wie der Altera-Byteblaster benutzt, unterstützt Lauterbach auch die Umprogrammierung des SOPC mit dem Debugger TRACE32-ICD.

Die Integration von Trace-Logik in den SOPC erlaubt das Sichtbarmachen des Programm- und Datenflusses über einen 19 Pin breiten Traceport. Nach der Aufzeichnung dieser



Information in den 512 MByte großen Tracespeicher von TRACE32-PowerTrace sind eine planvolle Fehlersuche, umfassende Laufzeitmessungen sowie eine Codeabdeckungs-Analyse möglich.

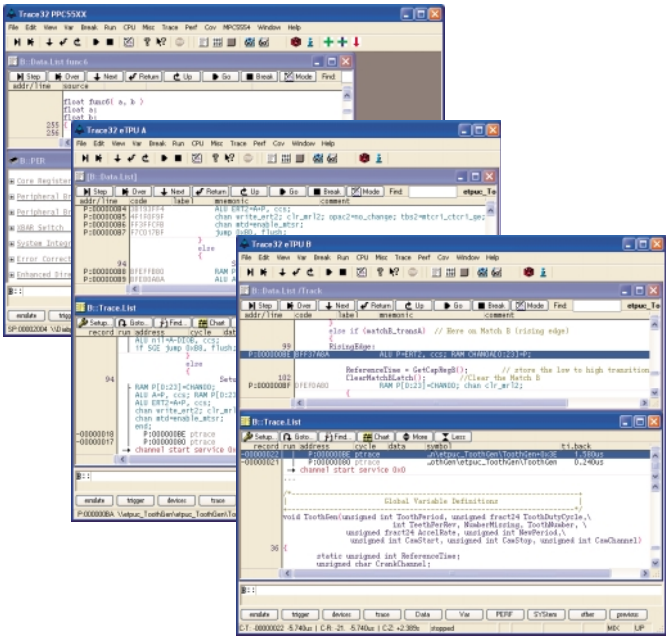
Da ein SOPC auch mehrere NIOS II Softcores enthalten kann, unterstützt Lauterbach von Anfang an auch das Multicore Debugging für diese Architektur.

eTPU Debugger für MPC55xx

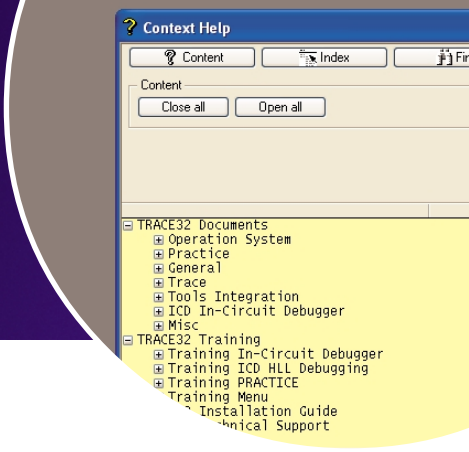
Seit Oktober 2004 unterstützt Lauterbach mit seinen Entwicklungswerkzeugen das Debugging und Tracen für die beiden eTPUs der MPC55xx-Familie von freescale.

Die Grundidee für das Testen der eTPUs ist folgende: Sowohl für den Core, als auch für jede eTPU wird eine eigene Applikation gestartet. Somit kann jede Applikation ihr eigenes Programm laden und debuggen, sowie ihre eigenen NEXUS-Messages darstellen. Der Core, der DMA-Controller sowie beide eTPUs können NEXUS-Messages erzeugen, die über einen gemeinsamen Traceport ausgegeben werden.

Lauterbach realisiert hier für den MPC55xx erstmals einen Multi-Source Trace. Aufgabe der TRACE32-Systemsoftware ist es vor allem, die Tracepakete an die einzelnen Applikationen zu verteilen, sowie dafür zu sorgen, dass der Entwickler eine übersichtliche Darstellung des zeitlichen Zusammenhangs zwischen Core- und eTPU-Tracedaten erhält.



Für das Debugging und Tracen von Core / DMA, eTPUA und eTPUB wird je eine eigene Applikation gestartet



Lauterbach APIs

Lauterbach stellt mit seinen APIs Software-Schnittstellen zur Verfügung, die es erlauben:

- *ein TRACE32-Entwicklungswerkzeug durch eine externe Applikation zu steuern.*
- *den Funktionsumfang eines TRACE32-Entwicklungswerkzeugs mittels einer externen Applikation zu erweitern.*

Im folgenden werden die einzelnen TRACE32 APIs kurz vorgestellt.

TRACE32 API

Die TRACE32 API ist die Standard-API zur Steuerung eines TRACE32-Entwicklungswerkzeugs durch eine externe Applikation. Folgende funktionsspezifische APIs sind Teil der TRACE32 API:

- *Die **Visual Basic API** stellt eine vorcompilierte DLL zur Verfügung, die es ermöglicht, ein TRACE32-Entwicklungswerkzeug über ein Visual Basic Programm zu steuern.*
- *Die **FDX API** stellt C-Bibliotheken bereit, um einer externen Applikation, die auf dem Host läuft, einen sehr schnellen Datenaustausch mit der Applikation auf dem Zielsystem zu ermöglichen. Das TRACE32-Entwicklungswerkzeug dient dabei als Kommunikationsschnittstelle.*
- *Die **JTAG API** stellt C-Bibliotheken zur Verfügung, die einer externen Applikation erlauben über eine TRACE32-ICD Debugger Hardware direkt mit einem JTAG TAP-Controller im Zielsystem zu kommunizieren. Damit kann man ohne eigenes Hardware-Interface beispielsweise Boundary Scan Testprogramme oder Basisfunktionen eines Debuggers für einen weiteren Core in der JTAG Kette des Mikrocontrollers implementieren.*

Die folgenden APIs stellen eigenständige Software-Schnittstellen zur Verfügung:

Simulator API

Die Simulator API erlaubt es, für die von Lauterbach angebotenen Instruction Set Simulatoren Timer, externe Kommunikationsschnittstellen und andere Peripheriekomponenten zu definieren. So kann der Instruction Set Simulator auch für den Test von Programmen eingesetzt werden, die Timer bedienen oder über Kommunikationsschnittstellen Daten austauschen.

Communication API

Für eine ganze Reihe von Mikrocontroller bietet Lauterbach ROM Monitor-basierte Debugger an. Diese Debugger kommunizieren mit dem im Zielsystem installierten ROM Monitor über eine RS232 Schnittstelle bzw. über die Schnittstelle einer Lauterbach EPROM Simulator Hardware.

Die Communication API erlaubt nun, einen ROM Monitor-basierten Debugger durch eine externe Applikation so zu erweitern, dass er auch über andere Kommunikationsschnittstellen wie beispielsweise Ethernet oder CAN betrieben werden kann.

Kernel Awareness API

TRACE32-Entwicklungswerkzeuge unterstützen standardmäßig die meisten auf dem Markt gängigen Echtzeit-Betriebssysteme. Hauptfunktionen dieser Unterstützung sind:

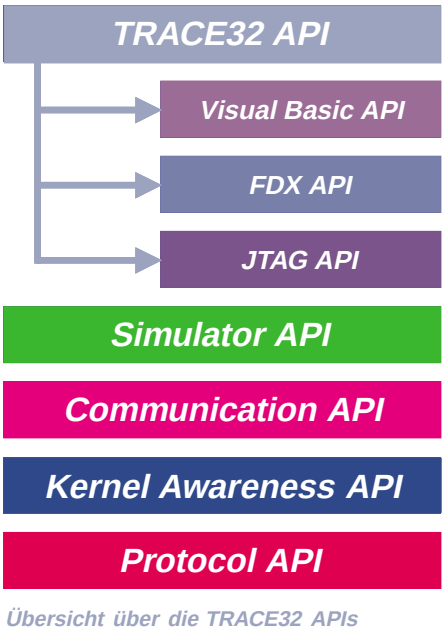
- *Anzeige von Betriebssystem-Ressourcen während des Debugging*
- *Umfassende Möglichkeiten zur Analyse des Laufzeitverhaltens der einzelnen Tasks*

Die Kernel Awareness API erlaubt den Nutzern eines TRACE32-Entwicklungswerkzeugs diese Betriebssystemunterstützung auch an eigene, proprietäre Echtzeit-Betriebssysteme anzupassen.

Protocol API

Die TRACE32-Produkte PowerProbe und PowerIntegrator ermöglichen die Aufzeichnung des Zeitverhaltens beliebiger Kommunikationsschnittstellen sowie beliebiger Busprotokolle. Die Auswertung und Darstellung wichtiger Kommunikations- und Busprotokolle wird von der TRACE32-Software bereits unterstützt.

Die Protocol API ermöglicht die Umsetzung von Roh-Tracedaten in höhere Protokolle über eine externe Applikation auch dann, wenn die TRACE32 Software noch keine Anpassung für das Kommunikations- bzw. Busprotokoll bietet. So kann das im Trace aufgezeichnete Zeitverhalten intuitiv und schnell analysiert werden.



Übersicht über die TRACE32 APIs

TRACE32-FIRE
der superschnelle, voll integrierte Risc-Emulator

In-Circuit Emulator für die S12X-Familie

Nachdem bereits seit August 2004 der BDM-Debugger für die S12X-Familie von freescale lieferbar ist, wurde zum Jahresende 2004 auch die Entwicklung für den In-Circuit Emulator TRACE32-FIRE abgeschlossen.

Der In-Circuit Emulator bietet gegenüber dem BDM-Debugger folgende entscheidende Vorteile:

Emulationsspeicher: *Das Onchip-Flash wird durch Emulation-RAM ersetzt. Damit lässt sich das zu testende Programm schneller laden, gleichzeitig stehen dadurch beliebig viele Software-Breakpoints für das Debugging zur Verfügung.*

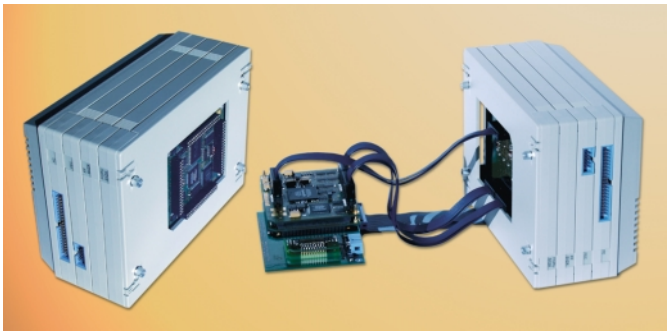
Zusätzliche Read-/Write-Breakpoints: *Der In-Circuit Emulator kann eigene Ressourcen für das Setzen von Read-/Write-Breakpoints zuschalten. So kann die Programmausführung beim Lesen bzw. Beschreiben von Speicher/Variablen angehalten werden. Zwei dieser Read-/Write-Breakpoints lassen sich zudem mit einem Datenwert verknüpfen.*

Code Coverage/Variablen Analyse: *Durch das Zuschalten eigener Ressourcen bietet der In-Circuit Emulator eine Codeabdeckungsanalyse sowie eine umfassende Analyse der Schreib-/Lesezugriffe auf Speicher und Variablen.*

512 K Frames Tracespeicher: *Beim Einsatz des In-Circuit Emulators werden die 64 Einträge des Onchip-Tracespeichers nicht mehr für den S12X-Core benötigt. Die Onchip-Tracefunktionen stehen ausschließlich für das Arbeiten am XGATE-Coprozessor zur Verfügung.*

Port Analyzer: *Lauterbach bietet für all seine Emulatoren so genannte Port Analyzer an. Ein Port Analyzer bietet die Möglichkeit die Signale sämtlicher Portleitungen ohne zusätzliche Kontaktierung aufzuzeichnen und einen direkten zeitlichen Zusammenhang zum Programmablauf herzustellen.*

Der FIRE Emulator für die S12X-Familie bietet außerdem eine verbesserte Adaption an das Zielsystem an (siehe Bild).



Bei Bedarf kann das S12X-Coremodule durch Verwendung von Mictor Flex Extensions vom FIRE-Grundsystem getrennt werden. So wird die Adaption an die Zielhardware erleichtert.

Neue Niederlassung in Italien

Mit Jahresbeginn 2005 eröffnete Lauterbach seine neue Niederlassung in Italien. Durch Lauterbach Srl, mit Sitz in Mailand, stärkt Lauterbach seine Position auf dem italienischen Entwicklungswerkzeugemarkt.

Geschäftsführer von Lauterbach Srl ist Maurizio Menegotto, der eine langjährige Erfahrung im Vertrieb und der technischen Betreuung von Lauterbach-Produkten mitbringt.

www.lauterbach.it

Benachrichtigen Sie uns:

Falls wir Sie aus unserer Mailing-Liste streichen sollen, schicken Sie bitte eine E-Mail an:
info@lauterbach.com

LAUTERBACH

Weitere Informationen im Internet: www.lauterbach.com