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

NEWS

2003

NEUHEITEN ZUR EMBEDDED WORLD 2003

Weitere Informationen im Internet: www.lauterbach.com

Optimistische Aussichten für das Jahr 2003

Umsatzsteigerung

2002 war mit einer Umsatzsteigerung von 20 % ein sehr erfolgreiches Jahr für die Firma Lauterbach. Wann immer wir mit Kunden oder Branchenkollegen darüber sprachen, schauten wir in erstaunte Gesichter. Umsatzsteigerungen im Jahr 2002? Ist so etwas möglich in einem so schwierigen Umfeld? Für die meisten Firmen im High-Tech-Bereich ist das Jahr 2002 nicht so gut gelaufen und der Markt für Entwicklungswerkzeuge ist letztes Jahr wahrscheinlich nicht gewachsen. Einiges spricht dafür, dass Lauterbach also Anteile am Gesamtmarkt dazugewonnen hat. So konnte also auch in 2002 unsere Marktposition wieder gestärkt werden. Ganz deutlich war das in Großbritannien zu erkennen, wo die neue Niederlassung der Firma Lauterbach deutliche Umsatzsteigerungen verbuchte und schon im ersten Jahr profitabel wurde.

Ob das Jahr 2003 wieder ein erfolgreiches Jahr werden wird, kann natürlich niemand vorhersagen, aber Lauterbach geht mit den besten Voraussetzungen in das neue Jahr.

Eigene Niederlassung in Japan

Dass Japan ein großer und wichtiger Markt ist, weiß man. Dass es für ausländische Firmen ein sehr verschlossener und schwieriger Markt ist, weiß man auch. Umso wichtiger ist es daher, dort mit einer eigenen Niederlassung vertreten zu sein, wenn man seine Produkte dort erfolgreich vermarkten möchte. Lauterbach ist es gelungen, japanische Mitarbeiter mit großer Erfahrung im Embedded Markt für sich zu gewinnen, mit denen wir in 2003 eine Niederlassung aufbauen werden. Dies ist ein wichtiger Schritt, um die weltweite Präsenz der Firma weiter zu verstärken.

Verstärktes Entwicklungsteam

Ende 2002, Anfang 2003 konnten für unseren Hauptsitz in Hofolding mehrere neue Mitarbeiter gewonnen werden. Sie verstärken die Entwicklungsabteilung insbesondere bei VDHL-Designs in

TRACE32-Produkten und bei der Weiterentwicklung der JTAG-Tools. So kann man also zuversichtlich sein, dass alle geplanten Projekte für 2003 zügig und planmäßig umgesetzt werden können.

Neue Produkte auf der embedded world 2003

Wie sie auf den nächsten Seiten feststellen werden, gibt es viele neue Produkte und Ideen bei Lauterbach. Die Schwerpunkte bilden hierbei sicher das Thema Multicore-Debugging und ein erweitertes Angebot an Debuggern für DSPs.

*Ein brandneues Produkt in diesem Zusammenhang, über das wir hier wegen Vertraulichkeitsvereinbarungen noch nicht schreiben dürfen, hoffen wir Ihnen schon auf der **embedded world 2003** in Nürnberg vorstellen zu können. Wir laden Sie recht herzlich ein, uns auf unserem Messestand zu besuchen, um ein angenehmes und informatives Gespräch mit Ihnen zu führen.*

LAUTERBACH

Multicore-Debugging

Das Debuggen von System-on-Chip Bausteinen, die mehrere Cores enthalten, stellt neue Anforderungen an die Entwicklungswerkzeuge. Ideal wäre es, wenn sich Chip-Designer und Toolhersteller schnell über geeignete Techniken verständigen, die dem Anwender bei freier Wahl der Core-Kombination eine schnelle Verfügbarkeit hochwertiger Entwicklungswerkzeuge garantieren.

Für Embedded Designs werden heute zunehmend anwendungsspezifische ASICs eingesetzt. Um eine optimale Funktionalität und Performance zu erreichen, werden dabei immer häufiger mehrere Cores zu einem System-on-Chip (SoC) integriert. Weit verbreitet ist das Zusammengehen eines RISC-Prozessors mit einem DSP. Ein Beispiel für eine solche Core-Kombination ist der S-GOLD von Infineon, bei dem sich ARM926EJ-S und OAK DSP auf einem Chip befinden.

Der Test und die Integrationsphase solcher Multicore-SoC Designs stellen neue Anforderungen an Mikroprozessor-Entwicklungswerkzeuge. Aus der Vielzahl der Anforderungen greift dieser Artikel einen Schwerpunkt heraus: die Ansteuerung mehrerer Cores über eine gemeinsame Debugschnittstelle. Es wird versucht die Problemstellung zu erläutern und Lösungsansätze vorzustellen.

Ansteuerung mehrerer Cores über eine Debugschnittstelle

Zum Debuggen stellen die meisten Prozessoren heute eine On-Chip Debuglogik zur Verfügung. Die am häufigsten eingesetzte Schnittstelle zur Ansteuerung der Debuglogik ist die JTAG-Schnittstelle.

Bei Multicore-SoC Designs wird, um Pins und damit Kosten zu sparen, nicht für jeden Core eine eigene JTAG-Schnittstelle bereitgestellt, sondern eine gemeinsame JTAG-Schnittstelle muss zum Debuggen aller Cores genügen. Damit wird die Frage aufgeworfen, wie der Debugger das Ansprechen mehrerer Cores über die gemeinsame JTAG-Schnittstelle realisiert und wie ein synchrones Debuggen umgesetzt werden kann.

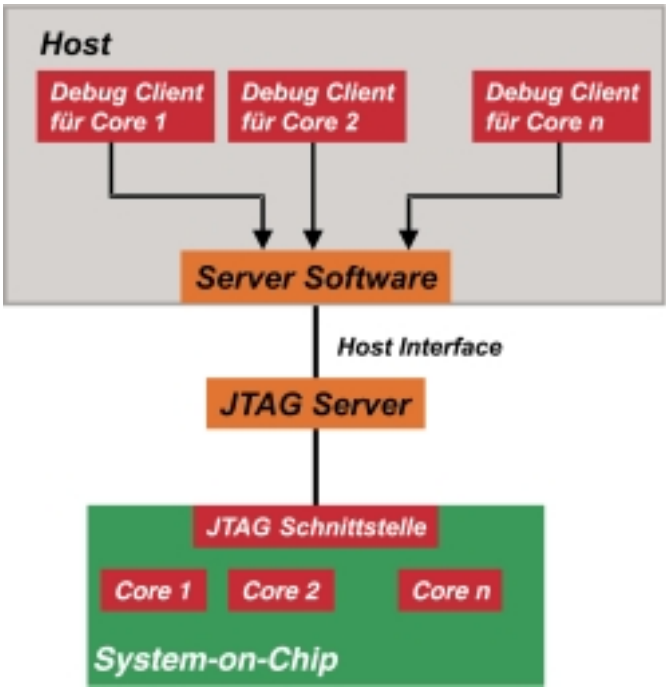


Bild 1: JTAG-Serverlösung

1. Ein Debugger unterstützt alle Cores auf dem SoC

Diese Lösung ist auf den ersten Blick die Ideallösung für den Entwickler. Hauptvorteil hierbei ist: ein Entwicklungswerkzeug für alle Cores und damit die Einarbeitung in nur eine Oberfläche und Produktphilosophie.

Dieser Ansatz ist für SoCs, die in großen Stückzahlen gefertigt und für viele Entwicklungen eingesetzt werden, sicher der Beste. Ein Beispiel für eine solche Realisierung ist der TriCore (Infineon) Debugger von Lauterbach, der sowohl das Debuggen des Hauptcores als auch des Peripheral Control Processors (PCP) ermöglicht.

Dem entgegen steht der Trend zum individuellen SoC, der nur für eine Produktentwicklung eingesetzt wird. Schon für das nächste Produkt wird ein neues SoC geschnitzt, das eine noch optimalere Performance und Funktionalität erreicht. Will man zudem noch volle Freiheit bei der Auswahl der integrierten Cores, muss man einen Toolhersteller an der Hand haben, der alle eingesetzten Architekturen optimal bedient. Natürlich sollte der Debugger auch bereits voll ausgetestet zur Verfügung stehen, wenn der erste Chip dem Waver entsprungen ist.

Ganz offensichtlich ist diese Lösung zwar optimal, die Chancen für eine kostengünstige, qualitativ hochwertige und vor allem schnelle Realisierung sind aber eher gering.

2. Einsatz eines JTAG-Servers

Für die meisten Cores stehen heute bereits voll ausgereifte Debugger zur Verfügung. Warum schließt man also die vorhandenen Debugger nicht an einen JTAG-Server an, der sich um das Ansprechen der einzelnen Cores über die gemeinsame JTAG-Schnittstelle kümmert?

Eine solche Realisierung kann man sich wie folgt vorstellen: der gemeinsamen JTAG-Schnittstelle wird eine Server Hardware vorgeschaltet. Diese Hardware wird von einer Server-Software auf dem Host gesteuert. Der einzelne Debug-Client schickt nun per Remote Procedure Call seine Kommunikationsanforderung an den Server. Dieser gewährleistet den exklusiven Zugriff auf die gemeinsame JTAG-Schnittstelle und leitet die Kommunikationsanforderung an den einzelnen Core weiter. Bild 1 zeigt das Grundprinzip einer solchen Serverlösung. Der Debugger benutzt hier statt einer eigenen JTAG-Hardware also eine Software-Schnittstelle zum JTAG-Server.

Auch dieser, auf den ersten Blick sehr elegante Ansatz, bietet keine vollständige Lösung des Problems. Zwar gibt es bereits heute von einigen Halbleiterherstellern Serverlösungen für Core-Gruppierungen aus ihrem Produktangebot, ein Serverstandard, der eine herstellerübergreifende Lösung anbietet, ist jedoch nicht in Sicht.

Die JTAG-Serverlösung erweist sich zudem als wenig flexibel, wenn einzelne Halbleiterhersteller die JTAG-Signale ihres Cores erweitert haben, um zusätzliche Debug-Funktionalität anzubieten. Für eine Multicore-Entwicklungsumgebung wäre es beispielsweise wünschenswert, wenn jeder Core über ein Stop Request Signal verfügen würde, das ein sofortiges Anhalten des Cores ermöglicht. Zudem könnte über ein Stop Indication Signal angezeigt werden, dass ein Core angehalten hat. Beide Signale ließen sich ideal für ein synchrones Anhalten aller Cores in einer Multicore-Anwendung verwenden.

Bei freier Auswahl einer optimalen Core-Kombination kann es schnell eine Anzahl zusätzlicher Signale geben. Man könnte nun diese Signale einfach unter den Tisch fallen lassen und auf die zusätzliche Funktionalität verzichten. Oft sind diese Signale jedoch notwendig oder zumindest sehr hilfreich und sollten schon deshalb bedient werden. Der JTAG-Server müsste also doch wieder anwendungsspezifisch angepasst und getestet werden.

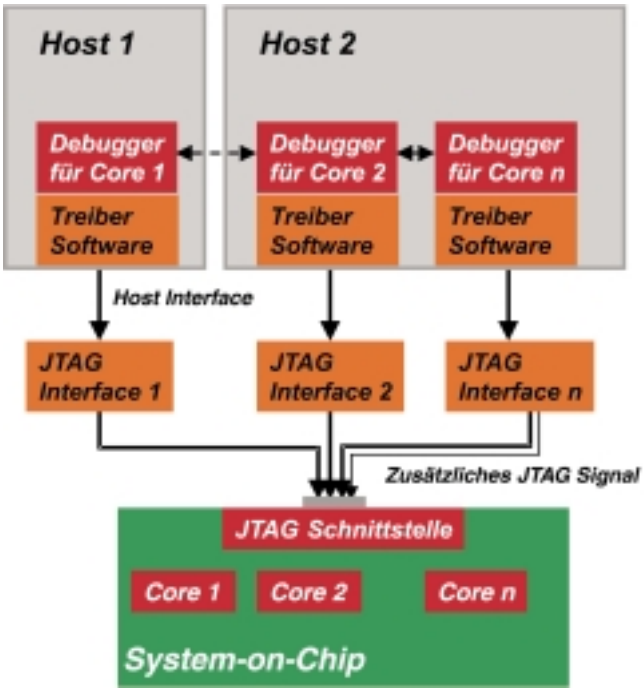


Bild 2: Komplett unabhängige Debugger an einer gemeinsamen JTAG-Schnittstelle

Als weiterer Knackpunkt speziell für Echtzeitanwendungen kommen noch die großen Datenmengen hinzu, die vom JTAG-Server abgewickelt werden müssen. Die Reaktionszeit des einzelnen Cores wird dadurch entsprechend verlangsamt. Um den Echtzeitanforderungen dennoch gerecht zu werden, müssen zeitkritische Funktionen in die Hardware gepackt werden, wodurch wir wieder beim anwendungsspezifischen Server gelandet sind.

Eine Serverlösung ist also vor allem dann optimal, wenn alle verwendeten Cores vom gleichen Halbleiterhersteller kommen und dieser gleich den JTAG-Server und die dazu passenden Debugger mitliefert. Anwendungsspezifische Server hingegen erfordern Absprachen zwischen den Herstellern der eingesetzten Debugger, die sich durch die Konkurrenzsituation meist als schwierig erweisen und zudem zeit- und kostenintensiv sind. Hinzu kommt, dass für jede neue Core-Kombination erneut eine Anpassung des Servers notwendig wird.

3. Komplett unabhängige Debugger an der gemeinsamen JTAG-Schnittstelle

Dieser Ansatz stellt eine einfache und offene Lösung dar, die es erlaubt, die optimale Core-Kombination frei für die

eigene Anwendung auszuwählen. Bei der Entwicklung und Integration kann dabei auf verfügbare, voll ausgereifte Debugger zurückgegriffen werden, ohne dass eine anwendungsspezifische Anpassung der Entwicklungsumgebung notwendig ist.

Die Grundidee ist hier, dass für jeden Core ein unabhängiger Debugger an die gemeinsame JTAG-Schnittstelle angeschlossen wird. Damit dieser Ansatz funktioniert, muss jeder Debugger seinen JTAG-Treiber abschalten, wenn er keine Daten mit seinem Core austauscht. Bild 2 auf Seite 3 zeigt diesen Ansatz.

Da nun mehrere unabhängige Debugger die gleiche JTAG-Schnittstelle benutzen, muss man sicherstellen, dass zu jedem Zeitpunkt nur ein Debugger auf die Schnittstelle treibt. Dies lässt sich automatisieren, indem die Debug-Tasks auf dem Host über den Einsatz eines Semaphors regeln, wer den exklusiven Zugriff auf die JTAG-Schnittstelle erhält. Alternativ wäre auch eine Art Hardware-Semaphor denkbar.

Das synchrone Starten und Stoppen aller Cores kann einfach und effektiv über eine spezielle Logik auf dem SoC gelöst werden. Für ein synchrones Stoppen beispielsweise kann jeder Core mit zwei zusätzlichen Signalen ausgestattet werden:

- *Ein Stop Request Signal, das ein sofortiges Anhalten des Cores ermöglicht.*
- *Ein Stop Indication Signal, das anzeigt, dass der Core angehalten hat.*

Diese Signale könnten dann auf dem SoC über eine Matrix verknüpft werden. Jeder Core kann dann beispielsweise über das Setzen eines memory-mapped Kontrollregister einstellen, ob er beim Anhalten eines anderen Cores ebenfalls stoppen oder weiterlaufen möchte. Bild 3 zeigt eine solche Matrix.

Die Matrix ließe sich auch einfach für die Peripherien der einzelnen Cores erweitern. Hauptvorteil dieser Lösung ist eine sehr hohe Synchronisation beim Anhalten und eine vom Chip-Designer bereitgestellte einheitliche Schnittstelle für die Debuggerhersteller.

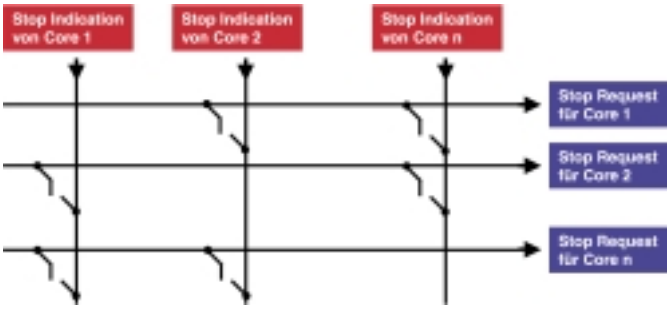


Bild 3: Matrix zum synchronen Anhalten mehrerer Cores

Der Ansatz mehrere unabhängige Debugger an einer gemeinsamen JTAG-Schnittstelle zu betreiben, bietet ein relativ einfaches Verfahren für Chip-Designer und Toolhersteller. Wenn ein paar Grundvoraussetzungen geschaffen werden, können beliebige Cores in das SoC integriert und die dazugehörige Entwicklungsumgebung frei gewählt werden.

Zusammenfassend ergeben sich hier folgende Vorteile:

- *Für diesen Lösungsansatz sind keine zusätzlichen Absprachen zwischen den Toolherstellern notwendig. Leistungsfähige Entwicklungswerkzeuge stehen sofort zur Verfügung.*
- *Die vorgestellte Lösung erfordert keine anwendungsspezifischen Anpassungen seitens der Toolhersteller. Es können Standardtools eingesetzt werden, die in anderen Projekten wieder verwendbar sind.*
- *Der offensichtliche Nachteil, dass für jeden Core ein komplettes Debugsystem angeschafft werden muss, erweist sich auf den zweiten Blick eher als Vorteil, da während der Entwicklungsphase sowieso meist nur ein Core ausgetestet wird und dabei die Debugger einzeln verwendet werden. Lediglich in der Integrationsphase, beim Austesten der Aktionen zwischen den Cores, kommen alle Debugger gemeinsam zum Einsatz.*
- *Lauterbach unterstützt zudem mit seinen Debuggern schon heute eine breite Palette von Cores. Bei einem Einsatz dieser Debugger für Multicore-SoC Designs ist daher eine einheitliche Bedieneroberfläche und eine gemeinsame Produktphilosophie bereits gegeben.*

Solange das Multicore-Debugging noch in den Kinderschuhen steckt und kein allgemeiner Standard festgesetzt wird, betrachtet Lauterbach diese dritte Lösung als die zweckmäßigste. Die Debugger von Lauterbach sind bereits für den Betrieb an einer gemeinsamen JTAG-Schnittstelle ausgerüstet.

Multicore-Debugging mit TRACE32-ICD/PowerTrace

Debugger-Hardware

Zum Debuggen mehrerer Cores über eine gemeinsame JTAG-Schnittstelle können einfach mehrere TRACE32-ICD/PowerTrace aneinander gesteckt und über eine gemeinsame Schnittstelle an den Hostrechner angeschlossen werden.

Bild 1: Zwei unabhängige TRACE32-ICD Debugger, die ein SoC mit zwei Cores über eine gemeinsame JTAG-Schnittstelle debuggen

Jeder TRACE32-ICD/PowerTrace verfügt dabei über ein eigenes JTAG-Kabel. Für den Anschluss der JTAG-Kabel gibt es zwei Möglichkeiten:

- *Auf dem Zielsystem ist für jedes JTAG-Kabel ein eigener Stecker vorgesehen.*
- *Auf dem Zielsystem ist nur ein JTAG-Stecker vorhanden. Dann muss ein entsprechender Adapter verwendet werden, der Steckplätze für alle JTAG-Kabel bereitstellt. Eine solche Lösung zeigt Bild 1.*

Selbstverständlich ist auch das Hinzufügen der passenden Tracemodule möglich.

Konfiguration

Was muss der Anwender nun bei der Konfiguration seines TRACE32-ICD Debuggers beachten?

Entscheidend für die Konfiguration ist, wie die Cores im SoC angeordnet sind. Die nun folgende Beschreibung bezieht sich auf eine einfache, jedoch häufig eingesetzte Anordnung: Die JTAG-Ports der einzelnen Cores sind seriell angeordnet (daisy chained), wie im Bild 2 dargestellt.

Die erste Frage ist: Wie spricht der einzelne Debugger seinen Core an?

Laut JTAG-Konvention gibt es die Möglichkeit, Sequenzen für die Scan Chain zu erzeugen, die dafür sorgen, dass einzelne TAP Controller sich neutral verhalten (BYPASS). Um gezielt einen Core anzusprechen, muss der Debugger also so konfiguriert werden, dass beim Generieren der Scan Chain für alle TAP Controller vor und nach diesem Core BYPASS-Sequenzen erzeugt werden (siehe IRPOST, IRPRE, DRPOST und DRPRE im Bild 3).

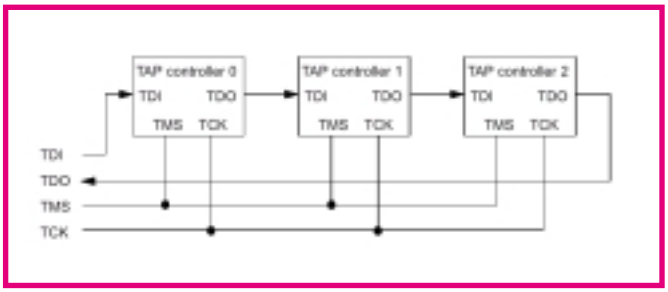


Bild 2: Serielle Anordnung der JTAG-Ports (daisy chained)

Die zweite Frage ist: Wie verhält sich der einzelne Debugger, wenn er keinen Zugriff auf den JTAG-Port benötigt?

Jeder Debugger, der die gemeinsame JTAG-Schnittstelle aktuell nicht benutzt, schaltet seine Leitungen in den Tristate Mode. Damit der Debugger bei der nächsten Aktivierung der JTAG-Leitungen definiert aufsetzen kann, muss ein geeigneter Zustand des TAP Controllers gewählt werden (TAPState). Der Debugger setzt den TAP Controller vor dem Aktivieren des Tristate Modes in diesen Zustand und erwartet natürlich, dass sich der TAP Controller bei einer erneuten Aktivierung der JTAG-Leitungen immer noch in diesem Zustand befindet.

Damit im Tristate Mode keine Leitungen offen sind und es nicht zu einem ungewollten Verstellen des TAP Controllers kommt, müssen die wichtigsten JTAG-Leitungen mit Widerständen abgeschlossen werden.

Zum Abschluss der Konfiguration muss nun noch festgelegt werden, welcher Debugger der Master über die Reset-Leitungen des SoCs ist (Slave).

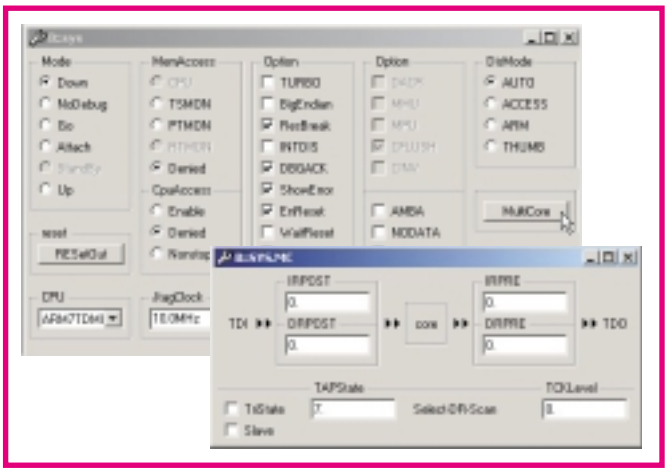
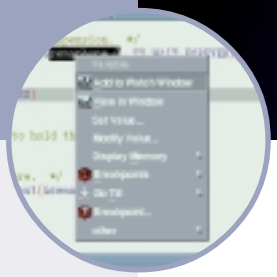


Bild 3: Konfiguration eines Debuggers, der mit anderen Debuggern eine gemeinsame JTAG-Schnittstelle benutzt

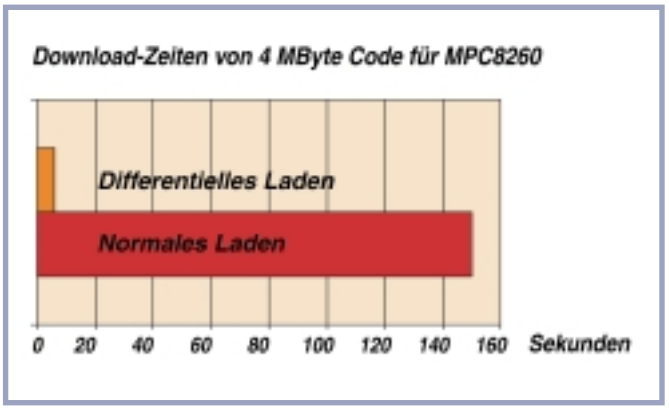


Differentielles Laden

Lauterbach führt für das Laden großer Programme neue Verfahren ein, die es ermöglichen, nur noch die Programmteile in das Entwicklungswerkzeug zu laden, die sich geändert haben. Da es sich dabei im allgemeinen nur um einen Bruchteil des Gesamtprogrammes handelt, wird so eine Reduktion der Download-Zeiten um den Faktor 30 erreicht.

Ein typischer Softwareentwicklungszyklus besteht aus Debuggen, Fehlerbeheben, Neucompilieren, Neuladen und wieder Debuggen. Werden große Programme auf einem Prozessor getestet, der auf Grund der Architektur der On-Chip Debugging Schnittstelle kein schnelles Laden erlaubt, verbringt der Entwickler viel Zeit mit Warten.

Um speziell die Wartezeiten beim Laden zu verkürzen, bietet Lauterbach nun die Möglichkeit, nur noch die Teile des Programmes neu zu laden, die sich auch geändert haben. Im Prinzip funktioniert das differentielle Laden wie folgt: Der Entwickler kann bei der Eingabe des Lade-Kommandos optional das differentielle Laden einschalten. Die Ladesoftware des

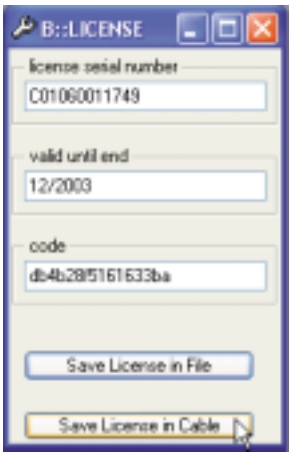


Entwicklungswerkzeuges ermittelt dann die geänderten Programmteile, komprimiert diese und lädt die komprimierten Daten über die On-Chip Debugging Schnittstelle in einen vordefinierten Speicherbereich. Anschließend startet die Ladesoftware einen Target-Agenten, der die Daten dekomprimiert und dafür sorgt, dass sie in die entsprechenden Speicherbereiche geschrieben werden.

Differentielles Laden wird vor allem für die folgenden Architekturen empfohlen: PPC7xx, MPC82xx, JANUS und ARM mit RTCK. Mehr Informationen zum differentiellen Laden finden Sie unter:

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

Software-Lizenzschlüssel



Um die Verwaltung der Software-Lizenzschlüssel zu vereinfachen, gibt es nun für alle TRACE32-ICD ab Dezember 2002 die Möglichkeit, den Lizenzschlüssel direkt im Debug-Kabel abzuspeichern.

Das einzelne Debug-Kabel kann nun problemlos auf einem Notebook oder in einer Testumgebung verwendet werden, in der kein Zugriff auf ein Lizenzfile möglich ist.

Neu unterstützte Echtzeitkerne

- ***QNX von QNX Systems Ltd.***
  für ARM, PowerPC und XScale
- ***µCLINUX***
  für 68k, ARM und ColdFire
- ***eCOS von Red Hat***
  für PowerPC
- ***Remote Debugging für OSE von OSE Systems***
  für ARM
- ***LINUX***
  für ARM und XScale

Eine Auflistung aller von TRACE32-PowerView unterstützten Echtzeitkerne finden Sie unter:

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

DIE PRODUKTLINIE

TRACE32-ICD

Neue Debugger

Lauterbach wird auch 2003 die Anzahl der Prozessoren, die vom In-Circuit Debugger TRACE32-ICD unterstützt werden, erweitern. Den Schwerpunkt bilden neben den ARM-Cores und den neuen PowerPC-Derivaten in diesem Jahr DSPs, die vor allem in Multicore-Designs eingesetzt werden.

ROM Monitor für ARM

Seit November 2002 bietet Lauterbach jetzt auch einen ROM Monitor für alle ARM-Cores und für den XScale von Intel an.

Der ROM Monitor kann entweder zusammen mit einem FLASH-Simulator oder über die serielle Schnittstelle betrieben werden. Damit steht nun auch ein TRACE32-ICD Debugger für alle ARM-Cores ohne JTAG-Schnittstelle zur Verfügung, wie beispielsweise für den ARM8 und den StrongARM.

Die Download-Zeiten für große Anwenderprogramme können durch Verwendung des differentiellen Ladens optimiert werden. Siehe dazu auch Seite 6.

Lizenzierung der seriellen ROM Monitore

Für die Lizenzierung der seriellen ROM Monitore steht seit November 2002 jetzt auch ein USB-Dongle zur Verfügung. Damit wird eine Lösung angeboten, die gegenüber der bisherigen Lizenzierung über die Ethernet-Adresse des Hosts mehr Flexibilität erlaubt.

ARM	**ARM9EJ** **ARM10 Core** **ARM11 Core**	sofort Q1/2003 Q3/2003
Avalent	**Janus 2**	sofort
DSP Group	**TeakLite/OAK**	Q2/2003
Hitachi	**H8S2319, H8S2329, H8S2339**	sofort
Infineon	**S-GOLD (ARM926EJ-S und OAK)**	sofort
Motorola	**MPC5500** **MGT5100** **MGT5200** **MPC750FX/CX/CXE** **MPC74xx** **MPC8265/MPC8264** **MPC8540/MPC8560** **MPC827x** **MPC828x** **MCS12C32 inklusive On-Chip Trace**	Q3/2003 sofort Q2/2003 Q1/2003 Q1/2003 Q1/2003 Q2/2003 Q2/2003 Q2/2003 sofort
STMicro-electronics	**Super10**	sofort

Software-Breakpoints im FLASH

Um den Debug-Komfort für die Toolfamilien TRACE32-ICD und PowerTools zu verbessern, hat Lauterbach die FLASH-Programmierung so erweitert, dass es nun möglich ist, Software-Breakpoints ins FLASH zu setzen und Code im FLASH zu patchen.

Die meisten Prozessoren, die das Debuggen über eine On-Chip Debugging Schnittstelle ermöglichen, verfügen nur über eine begrenzte Anzahl von On-Chip Breakpoints. Entwickler, die ihre Software im FLASH testen wollen, stoßen dadurch sehr schnell an eine Grenze.

Deshalb bietet Lauterbach seit Mitte 2002 auch die Möglichkeit, Software-Breakpoints in das FLASH zu setzen. Um den Zeitaufwand für die Neuprogrammierung zu minimieren, unterstützt die Software TRACE32-PowerView nun die volle Sektorierung aller FLASH-Typen. Durch intelligente Strategien wird zudem versucht, die Häufigkeit der FLASH-Neuprogrammierung zu reduzieren.

Durch die volle Sektorierung aller FLASH-Typen ist es nun außerdem möglich, mit geringem Zeitaufwand einen Patch im FLASH-Code durchzuführen.

Beide Features stehen inzwischen für alle internen und externen FLASHs zur Verfügung und können auch einfach für die Target basierende FLASH-Programmierung eingesetzt werden.

PowerIntegrator

Mit dem PowerIntegrator bringt Lauterbach zur embedded world 2003 einen Logic-Analyzer auf den Markt, den der Anwender einfach in seine TRACE32-Entwicklungsumgebung integrieren kann, um während des Debuggens alle Busaktivitäten und Portleitungen auf seinem Zielsystem zu überwachen.

Damit der PowerIntegrator schnell zum Einsatz kommen kann, bietet Lauterbach zusätzlich sogenannte Support-Packages an:

- ***Board-Support-Packages*** *konfigurieren den PowerIntegrator für den Einsatz auf Standard Evaluation Boards.*
- ***Bus-Support-Packages*** *konfigurieren den PowerIntegrator für spezielle Busprotokolle wie beispielsweise SDRAM Busse, Double Data Rate Busse etc.*

Der PowerIntegrator verfügt über 204 frei konfigurierbare Datenkanäle, von denen 12 wahlweise auch als Clockeingänge verwendet werden können. Über aktive Probes sind folgende Abtastraten möglich:

- ***500 MHz im Timing-Mode***
- ***200 MHz im State-Mode***

Zum Abtasten der Signale stehen zwei verschiedene Probes zur Verfügung:

- ***Mictor-Probe*** *mit 32 Datenkanälen und 2 Clock-/Datenkanälen*
- ***Standard-Probe*** *mit 16 Datenkanälen und 1 Clock-/Datenkanal*

Die Abtastschwelle für die Signale kann dabei frei in einem Bereich von 0 .. 4 V gewählt werden.

Die Tracespeichertiefe des PowerIntegrator beträgt 256 KFrames. Jeder Eintrag im Tracespeicher wird mit einem Zeitstempel versehen. Der Zeitstempel hat eine Breite von 48 Bits und arbeitet mit einer Auflösung von 8 ns.

Wie der PowerIntegrator arbeitet, soll nun an einem Beispiel kurz skizziert werden. Zur Veranschaulichung dient die Aufzeichnung des Programm- und Datenflusses über einen SDRAM Bus. Dazu sind folgende Schritte notwendig:

PowerIntegrator am Zielsystem

1. Definition, welche Signale den Adress- bzw. Datenbus bilden, sowie Konfiguration der Steuersignale

Für diese Definition bietet der PowerIntegrator ein universelles NAME-Kommando, das erlaubt, einzelne Signale zu gruppieren und ihnen logische Namen zuzuordnen.

2. Vorfilterung der Zyklen am SDRAM Bus

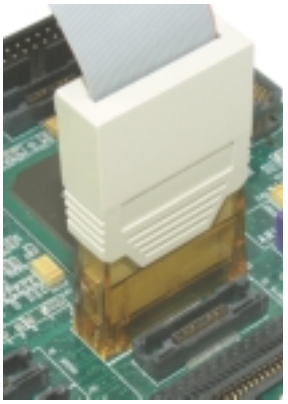
Um die Tracetiefe des PowerIntegrators optimal auszunutzen, ist es sinnvoll, aus den Aktivitäten am SDRAM Bus nur die Signale in den Tracespeicher zu übertragen, die für die spätere Generierung des Programm- und Datenflusses verwertbar sind. Dies kann durch eine geeignete Einstellung der Transient-Detection für die Steuersignale des SDRAM Busses erreicht werden.

3. Generieren des Programm- und Datenflusses mit Hilfe eines geeigneten Skriptes

Aus den im Tracespeicher aufgezeichneten, vorgefilterten Informationen muss nun noch der Programm- und Datenfluss generiert werden. Dazu wird über eine Skriptsprache definiert, welche Steuersignale für einen Fetch-, Read- bzw. Write-Zyklus aktiv sein müssen. Die Bildschirmabzüge auf Seite 9 zeigen dieses Vorgehen.

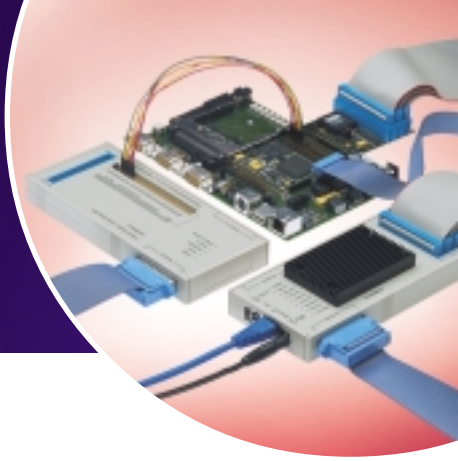
Arbeitet der PowerIntegrator zusammen mit einem Debugger, kann die Dateninformation der Fetch-Zyklen noch disassembliert werden. Die Darstellung der dazugehörigen Hochsprachen-information ist ebenfalls möglich.

Da das Erstellen des Skriptes zur Generierung des Programm- und Datenflusses aus den Tracedaten meist aufwendig ist und eine gute Kenntnis des Busprotokolls voraussetzt, bietet



Mictor-Probe

TRACE32-POWERTOOLS

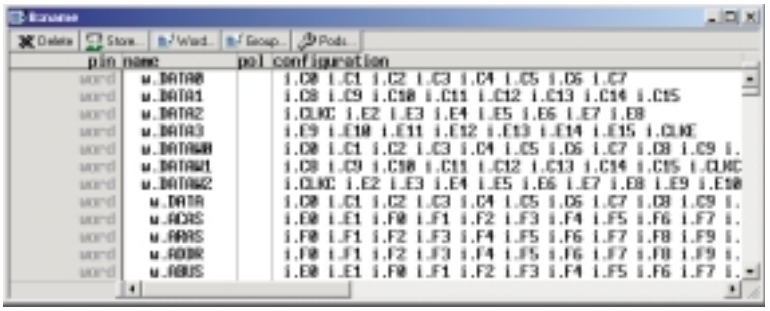


Lauterbach zu der skizzierten, eher Software basierenden Lösung die folgende Alternative:

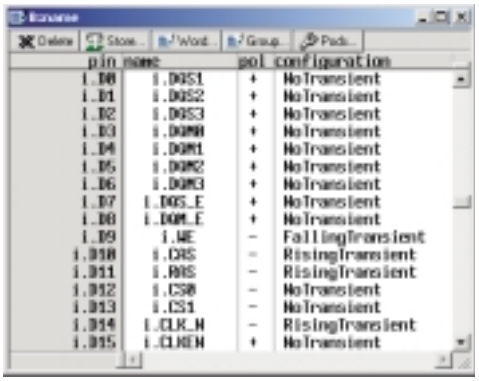
Durch den Einsatz eines SDRAM Bus-Support-Packages müssen die vorher beschriebenen drei Schritte nicht mehr vom Anwender selbst durchgeführt werden. Stattdessen wird die gesamte Intelligenz zur Generierung des Programm- und Datenflusses in ein FPGA geladen. Die Logik im FPGA generiert dann aus den Aktivitäten am SDRAM Bus physikalisch den Adress- und Datenbus, sowie die Steuersignale.

Der Einsatz eines Support-Package bietet dem Anwender folgende Vorteile:

- *Der PowerIntegrator ist bereits vollständig für die gewünschte Aufzeichnung konfiguriert.*
- *Die Filterung der Buszyklen ist für das Busprotokoll optimiert.*
- *Da der Adress- und Datenbus, sowie die Steuersignale physikalisch generiert werden, ist es nun auch möglich eine komplexe Echtzeit-Triggerung anzubieten. Die Triggerung erlaubt es, gezielt Zyklen aus dem Programm- und Datenfluss herauszufiltern oder den Prozessor unter einer definierten Triggerbedingung anzuhalten.*



Festlegung der Signale für den Adress- und Datenbus, sowie der Steuersignale



Vorfilterung der Zyklen am SDRAM Bus durch günstige Auswahl der Transient-Detection

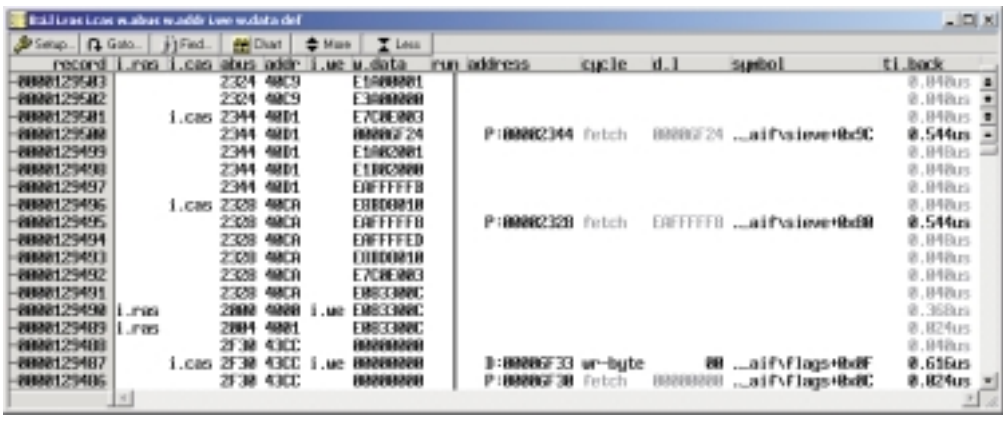
Mit der Markteinführung des PowerIntegrators hat sich Lauterbach das Ziel gesetzt, einen universellen Trace anzubieten.

*Durch den eher **Software basierenden Ansatz**, bei dem neben den Busaktivitäten selbstverständlich auch alle Portleitungen mit Hilfe eines Skriptes aufbereitet werden können, steht eine sehr flexible und offene Lösung zur Verfügung.*

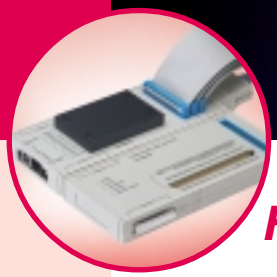
*Bei dem **FPGA basierenden Ansatz**, der durch die Support-Packages angeboten wird, steht eher die schnelle Verfügbarkeit einer effizienten Aufbereitung der Bus- und Portinformationen im Vordergrund.*

Für das erste Halbjahr 2003 sind Board-Support-Packages für Standard Evaluation Boards der Architekturen ARM, MIPS und XScale geplant. Eine Anpassung für den PowerQUICC III wird durchgeführt, sobald das erste Evaluation Board auf dem Markt ist.

Für den SDRAM Bus wird es das erste Bus-Support-Package geben. Weitere Busarchitekturen werden folgen. Die weitere Verfügbarkeit von Support-Packages wird sich dann direkt aus dem Kundeninteresse entwickeln.



Programm- und Datenfluss generiert aus den Aufzeichnungen der Aktivitäten am SDRAM Bus



Hardware basierendes Code Coverage

Welche Vorteile bringt Hardware basierendes Code Coverage gegenüber dem bisher möglichen Trace basierenden Code Coverage?

Während das Trace basierende Code Coverage durch die Tracegröße beschränkt ist und damit immer nur für einen kurzen Überwachungszeitraum durchgeführt werden kann, ist das Hardware basierende Code Coverage auch bei Langzeittests über große Programmbereiche einsetzbar.

Was ist Hardware basierendes Code Coverage?

Seit Oktober 2002 kann ein Hardware basierendes Code Coverage mit dem PowerTrace durchgeführt werden. Für diese Funktionalität ist im PowerTrace ein eigener Speicher vorhanden, aus dem 2 Bits pro Instruktion gemappt werden können. Mit diesen beiden Bits kann markiert werden:

	executed	branch taken
never	0	0
taken	0	1
not taken	1	0
ok	1	1

Markierung von Sprungbefehlen für das Code Coverage

- *ob eine Instruktion vom Programm ausgeführt wurde.*
- *ob ein Sprung durchgeführt wurde oder nicht.*

Für welche Prozessoren ist Hardware basierendes Code Coverage möglich?

Hardware basierendes Code Coverage ist aktuell für alle Prozessoren mit ARM-ETM (ARM7/9 basierte Cores), für den SH4 von Hitachi und den ST40 von STMicroelectronics möglich. Für weitere Architekturen wird eine Realisierung geprüft.

Grundvoraussetzung ist, dass der Prozessor so konfiguriert werden kann, dass die Traceinformation über den Programmfluss zur Laufzeit die Sprungziele sowohl der direkten, als auch der indirekten Sprünge enthält. Da die Sprungziel-Information meist kodiert und erst zeitversetzt verfügbar ist, wird die Traceinformation von einem FPGA weiterverarbeitet, um die Markierungen für die einzelnen Instruktionen zur Programmlaufzeit setzen zu können.

Wieviele Instruktionen können maximal überwacht werden?

Der PowerTrace stellt Speicher für 4 x 2 Millionen Instruktionen zur Verfügung. Dieser Speicher muss explizit gemappt werden. Der maximale physikalische Adressbereich für den ein Hardware basierendes Code Coverage durchgeführt werden kann, hängt dann im Wesentlichen von der minimalen Breite einer einzelnen Instruktion ab.

Bei einer Instruktionsbreite von 32-Bit können beispielsweise 4 x 8 MByte analysiert werden.

USB 2.0

Ab März 2003 wird das Power Debug Module/Ethernet mit einer USB 2.0 Schnittstelle ausgeliefert.

Mit einer Übertragungsrate von 480 Mbps ermöglicht USB 2.0 ein ebenso schnelles Laden wie eine 100 MBit Ethernet-Schnittstelle.

Da USB 2.0 voll aufwärts- und abwärts-kompatibel zu USB 1.x ist, kann das neue PowerDebug Module/Ethernet selbstverständlich auch an älteren PCs benutzt werden.

Neues zur PowerProbe

Die PowerProbe kann nun so konfiguriert werden, dass es möglich ist, damit einen Programm- und Datenfluss Trace aufzuzeichnen.

Dazu sind folgende Schritte notwendig:

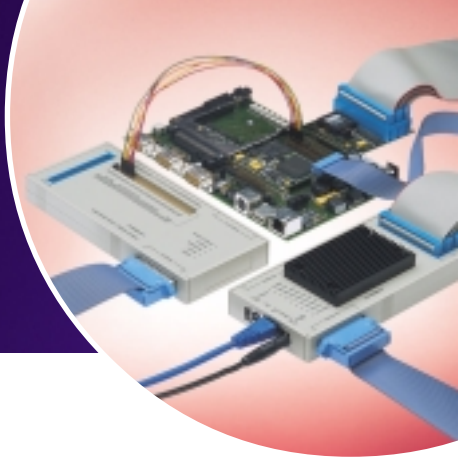
1.) Definition, welche Signale den Adress- bzw. den Datenbus bilden, sowie Konfiguration der Steuersignale.

2.) Generierung des Programm- und Datenflusses mit Hilfe eines Skriptes, das festlegt, welche Steuersignale für einen Fetch-, Read- und Write-Zyklus aktiv sein müssen.

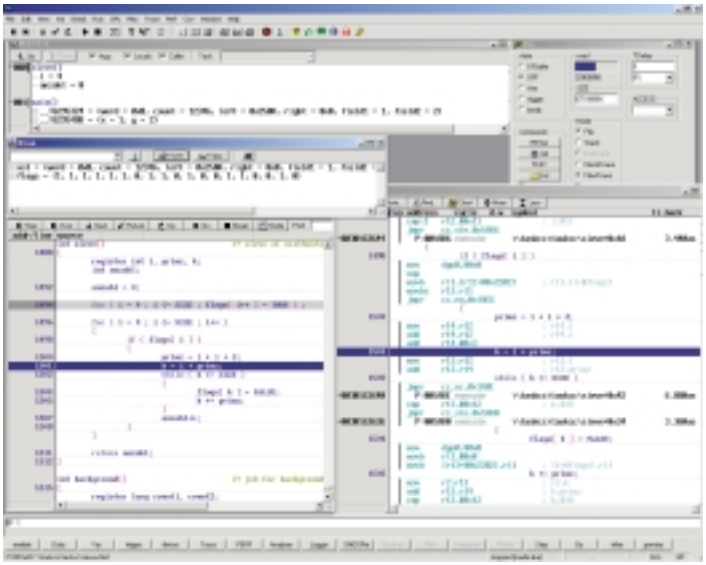
Arbeitet die PowerProbe mit einem Debugger zusammen bzw. ist die passende Symbol- und Debug-Information geladen, kann die Dateninformation der einzelnen Fetch-Zyklen disassembliert werden. Eine Zuordnung der Hochsprachen-Information ist dann natürlich möglich.

DIE PRODUKTLINIE

TRACE32-POWERTOOLS



NEXUS für Super10



Nachdem bereits seit Oktober 2002 der JTAG Debugger für den Super10 von STMicroelectronics verfügbar ist, wurde zum Jahresanfang 2003 die Entwicklung des NEXUS Debuggers abgeschlossen.

JTAG-Super10 *bietet ein komfortables Assembler- und Hochsprachen-Debugging für alle Standard-Compiler. Selbstverständlich werden dabei auch alle On-Chip Break- und Triggermöglichkeiten unterstützt.*

NEXUS-Super10 *bietet neben dem Debugging auch die Aufzeichnung des Programm- und Datenflusses. Da NEXUS-Super10 auf dem leistungsstarken PowerTrace realisiert ist, stehen neben einer umfassenden Traceauswertung auch Funktionen zur Laufzeitmessung, zur Performance Analyse und zum Code Coverage zur Verfügung. Der erste unterstützte Prozessor der Super10 Familie ist der ST10R303.*

Neue Gehäusetechnik für TRACE32

Höhere Frequenzen und moderne Tracetechniken erfordern eine ständige Weiterentwicklung unserer Produkte. Deshalb werden ab März 2003 folgende TRACE32-Produkte mit einer neuen Gehäusetechnik ausgeliefert.

Standardisierter Dongle mit steckbarem Debug-Kabel

- *prozessorspezifische Debug-Kabel*
- *prozessorspezifische Preprozessoren*
- *prozessorspezifische Nexus-Adapter*

Debug-Kabel

Durch einen größeren, standardisierten Dongle wurde mehr Platz für die immer komplexer werdende Logik zur Ansteuerung der On-Chip Debugging Schnittstelle geschaffen.

Das Debug-Kabel ist bei der neuen Gehäusetechnik nicht mehr fest an den Dongle angelötet, sondern einfach steckbar. Dies ermöglicht, soweit es die Frequenz der On-Chip Debugging Schnittstelle erlaubt, mehr Flexibilität bei der Kabellänge.

Preprozessoren und Nexus-Adapter

Zum besseren Schutz vor Verschmutzung und zur Vermeidung von Kurzschlüssen mit dem Zielsystem sind die Preprozessoren und Nexus-Adapter nun in ein Gehäuse eingebaut.

Um einen komfortableren Anschluß an das Zielsystem zu ermöglichen, werden für die meisten Preprozessoren und Nexus-Adapter zusätzlich Flex-Extensions angeboten.

Preprozessor im Gehäuse mit Flex-Extension zum komfortableren Anschluss an das Zielsystem

TRACE32-FIRE
der superschnelle, voll integrierte Risc-Emulator

Weitere Informationen im Internet: www.lauterbach.com

Aktuelles zum TRACE32-FIRE

Die Unterstützung von 16-Bit FLASH-Mikrokontrollern durch TRACE32-FIRE wird auch 2003 weiter ausgebaut.

512 KFrames Tracespeicher

Ab März 2003 ist der TRACE32-FIRE auch mit einem 512 KFrames tiefen Tracespeicher lieferbar.

H8S von Hitachi

Die Unterstützung der H8S Familie durch TRACE32-FIRE wird auch 2003 weiter ausgebaut. Geplant sind für das Frühjahr 2003 der H8S/2612 und der H8S/2678.

H8 von Hitachi

Seit September 2002 kann der RISC Emulator TRACE32-FIRE nun auch für die Emulation der H8 Familie von Hitachi eingesetzt werden. Folgende Prozessoren werden aktuell unterstützt:

- *H8/3006/3007/3008*
- *H8/3044/3045/3046/3047/3048*
- *H8/3052*
- *H8/3060/3061/3062/3064/3065/3066/3067/3068*

XC16x von Infineon

Der Support für die XC16x Familie von Infineon wurde 2002 um neue Derivate erweitert. Die folgenden Prozessoren können jetzt emuliert werden: XC161CJ, XC161CS und XC164CS.

Selbstverständlich plant Lauterbach für das Jahr 2003 die Anpassung des In-Circuit Debuggers TRACE32-ICD und des RISC Emulators TRACE32-FIRE für alle neuen Familienmitglieder.

MCS12 von Motorola

Die Unterstützung für die Mitglieder der 68HC12 und MCS12 Familie wurde auch 2002 weiter vervollständigt.

Neu hinzu kamen Module für den 68HC912DT128 und MCS12DB128.

ST10F276 von STMicroelectronics

Ab sofort unterstützt der RISC Emulator TRACE32-FIRE für die ST10 Familie auch den ST10F276.

Der ST10F276 verfügt über ein größeres On-Chip FLASH als der ST10F269 und ist deshalb für viele Entwicklungen sehr attraktiv. Kunden, die bereits einen TRACE32-FIRE für den ST10F269 im Einsatz haben, können diesen Emulator problemlos bei Lauterbach für den ST10F276 umrüsten lassen.

Bitte schicken Sie mir Informationsmaterial zu:

Wir setzen folgende Prozessoren ein:

_____

Ich interessiere mich für folgende Entwicklungswerkzeuge:

- ☐ ***TRACE32-ICD***
- ☐ ***TRACE32-PowerTools***
- ☐ ***TRACE32-PowerIntegrator***
- ☐ ***TRACE32-FIRE***

Absender:

Name _____

Firma _____

Adresse _____

_____

Telefon _____

Email _____

LAUTERBACH
FAX: ++49 8104 8943-187