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

NEWS

2004

NEUHEITEN ZUR EMBEDDED WORLD 2004

In 25 Jahren zum Weltmarktführer - ein Rückblick auf die Firmengeschichte

Die Gründungsjahre

Gegründet wurde die Ein-Mann-Firma 1979 von Lothar Lauterbach, der sich zunächst mit Auftragsentwicklungen für andere Firmen seine Brötchen verdiente. Für diese Tätigkeiten benötigte er einen Debugger. Weil nichts Gutes und Günstiges auf dem Markt war, wurde kurzerhand mit einer eigenen Debugger-Entwicklung begonnen. So kam es zum ersten eigenen Softwareprodukt, dem ***TRACE80 Debugger*** *für den Z80, lauffähig auf CPM-Systemen. 1982 stieß Stephan Lauterbach, der jüngere Bruder zur Firma, erst als Student im Praktikum, bald aber hauptberuflich mit Nebentätigkeit Student. Zu dieser Zeit entstand auch der erste echte* ***In-Circuit-Emulator TRACE80*** *für den Z80. Das Layout für die Platinen wurde in Klebetechnik mit Kreppband im Verhältnis 10:1 erarbeitet. 1984 wurden dann die ersten Mitarbeiter eingestellt und man bezog zwei Räume als Untermieter bei der Firma MBB in Ottobrunn bei München. Mit insgesamt 5 Personen machte man sich nun an die Weiterentwicklung, Produktion und Vermarktung der Tools. Das erste CAD-System wurde angeschafft, die Emulatoren auch für den 8085, den NSC800 und den HD64180 angepasst. Auch die passenden Hostrechner wurden von Lauterbach angeboten: Der Desktop-Rechner Commander und das tragbare Modell SCOUT, quasi ein Vorläufer der ersten Laptops. Beide basierten auf dem Mikroprozessor Z80 und dem damals gängigen Betriebssystem CPM. Zur Hannover-Messe 1985 wurden die Produkte dann erstmals einem breiten Publikum vorgestellt. Der Hochsprachendebugger für C und PLM sorgte damals für höchstes Aufsehen. TRACE80 entwickelte sich schnell zum Verkaufsschlager und brachte in den Jahren 1985/86 einen jährlichen Umsatz von ca. 4 Millionen DM ein.*

Die nächste Herausforderung

Damit war die Basis für den nächsten Meilenstein gelegt: die Entwicklung eines universellen In-Circuit-Emulators. Dieser Emulator sollte schrittweise

möglichst viele 8- bis 32-Bit Mikrocontroller unterstützen und gleichzeitig bahnbrechende, neue technische Möglichkeiten bieten. Dies war der Startschuss für die Entwicklung des ***TRACE32-ICE*** *für den legendären MC68000 von Motorola. Zur Verstärkung der Entwicklung wurden 1986 weitere Mitarbeiter eingestellt und zusätzlich eine Doppelhaushälfte in Neubiberg bei München angemietet. Stephan Lauterbach entwickelte im „Kinderzimmer" eine erste flexible Window-Oberfläche unter CPM und MS-DOS. Lothar Lauterbach saß im „Schlafzimmer" und dachte über ein universelles Hardwarekonzept nach.*

LAUTERBACH

Weitere Informationen im Internet: www.lauterbach.com

Nun sollten die schwierigsten, aber auch spannendsten Zeiten der Firmengeschichte kommen. Schwierig, weil sich die Entwicklung des neuen Emulators länger hinzog als geplant. Und zudem begab sich Lauterbach in ein Marktsegment, wo man sich etablierten Wettbewerbern mit großen Namen stellen musste.

TRACE32-ICE etabliert sich

Stephan Lauterbach

Im Herbst 1987 zog man in das eigene Gebäude am jetzigen Standort in Hofolding um. Um dieses zu finanzieren, war die Firma nun praktisch zum Erfolg gezwungen. Es folgten die magersten Jahre in der Unternehmensgeschichte. Unter diesen schwierigen Bedingungen konnten jedoch Schritt für Schritt Kunden gegen die „großen" Anbieter gewonnen werden. Lauterbach hatte nur ein einziges Verkaufsargument: ***die bessere Technik!*** *Argumente, wie „lange erprobt", „viele Referenzkunden" oder „etablierter Hersteller" waren der Konkurrenz vorbehalten. All diese Attribute treffen zwar inzwischen auch auf Lauterbach zu, Technik und Qualität sind aber bis heute immer noch die wichtigsten Unternehmensziele.*

Nach dieser „Durststrecke" entwickelte sich TRACE32-ICE Anfang der 90er Jahre schließlich auch zu einem Verkaufsschlager. Mit herausragender Technik und Qualität, sowie dem raschen Bereitstellen von neuen Emulatormodulen für neue Mikrocontrollerderivate konnte das Vertrauen der Kunden nun endgültig gewonnen werden.

Lothar Lauterbach

On-Chip-Debugging mit TRACE32-ICD und PowerTools

1994 stellte Herr Lauterbach wieder eine wichtige Weiche für den jetzigen Erfolg der Firma. Bis zu diesem Zeitpunkt lebte die Firma ausschließlich von dem Verkauf der klassischen In-Circuit-Emulatoren. In diesem Jahr wurde der erste BDM-Debugger für den 68332 entwickelt. Dies war die Geburtsstunde der In-Circuit Debugger ***TRACE32-ICD****. Es folgten Debugger für viele weitere Mikrocontroller und eine hervorragende Bilanz in Hinblick auf die verkauften Stückzahlen: 2001 konnte die Schwelle von 10.000 ausgelieferten TRACE32-ICD Debuggern überschritten werden. Für das Jahr 2004 rechnet Lauterbach mit über 25.000 ausgelieferten Systemen. Die Umsatzsteigerungen in den letzten Jahren lagen im Schnitt bei über 25%.*

Weltweit präsent

Für die Entwicklung der Firma war natürlich die Internationalisierung ein wichtiger Punkt. Anfang der 90er Jahre gewann Lauterbach Vertriebspartner in Europa und USA, Mitte der 90er dann auch im asiatischen Raum. 1995 wurde dann in den USA die erste eigene Niederlassung gegründet. Weitere Niederlassungen folgten: 2001 in England, 2002 in Japan. Zusammen mit seinen weltweiten Vertriebspartnern ist die Fa. Lauterbach somit hervorragend positioniert, um die Herausforderungen der nächsten Jahre zu meistern.

Alle Lauterbach Produkte auf einen Blick:
Emulatoren - TRACE80 (1984), TRACE32-ICE (1991), TRACE32-FIRE (1997),
Debugger - TRACE32-ICD (1995), PowerTrace (2001), PowerIntegrator (2003)

25-jähriges Jubiläum

Liebe Kunden, wir würden uns freuen, Sie dieses Jahr auf der embedded world 2004 in Nürnberg begrüßen zu können, um mit uns gemeinsam das Firmenjubiläum zu feiern und natürlich auch, um Sie über unsere neuesten Produkte zu informieren!

DIE PRODUKTLINIE

TRACE32-ICD

Schwerpunkt DSPs bei den Debuggern

Bei der Unterstützung neuer Prozessorarchitekturen durch den In-Circuit Debugger TRACE32-ICD standen digitale Signalprozessoren 2003 klar im Mittelpunkt.

LSI Logic	**ZSP**	sofort
Motorola	**DSP56800E**	sofort
Texas Instruments	**TMS320C55x** **TMS320C55x im OMAP**	sofort sofort
CEVA	**alle lizenzierten Cores mit JAM JTAG Interface** **TeakLite** **Teak** **XpertTeak**	 sofort Q1/04 Q1/04
StarCore LLC **Motorola**	**alle lizenzierten Cores** **MSC8101** **MSC8102**	 Q1/04 Q2/04

Tabelle 1: Neu unterstützte DSP-Prozessorarchitekturen

Datenintensive Anwendungen, wie etwa Bildübertragung oder Internet-Browser in Mobiltelefonen, erfordern immer leistungsfähigere Prozessorarchitekturen. Die hierbei anfallenden großen Datenmengen werden in der Regel effizient und schnell von digitalen Signalprozessoren verarbeitet. Für Embedded-Designs in der Kommunikationsindustrie lassen sich nun aktuell zwei Trends beobachten:

- *Es werden Multicore-SoCs eingesetzt, bei denen ein RISC Prozessor die Systemsteuerung und mehrere Signalprozessoren die Datenverarbeitung übernehmen.*
- *Für eine leistungsfähige Datenverarbeitung werden mehrere Signalprozessoren auf einem SoC integriert.*

Mit der Unterstützung für den OMAP von Texas Instruments, der einen TMS320C55x DSP und einen ARM925/926 Core enthält, ermöglichte Lauterbach erstmals zum Jahresbeginn 2003 das Debugging eines RISC und eines DSP mit einer gemeinsamen Debugger-Hardware. Beide Cores können dabei synchron gestartet und angehalten werden.

Aufgrund steigender Nachfrage hat Lauterbach eine Reihe weiterer DSPs in sein Produktspektrum aufgenommen. Dabei werden DSPs sowohl als eigenständige Mikrocontroller als auch als lizenzierte Cores, die meist in ein Multicore-SoC integriert sind, unterstützt. Siehe dazu auch Tabelle 1.

Aktuell wird an der Entwicklung eines Debuggers für die MSC81xx-Architektur von Motorola gearbeitet. Diese Architektur basiert auf einem lizenzierten DSP der Firma StarCore LLC. Zunächst wird der MSC8101 unterstützt. Mit dem Support für den MSC8102 wird Lauterbach zum ersten Mal eine Debug-Umgebung anbieten, in der ein Prozessor getestet werden kann, der vier DSP-Cores enthält.

Neue JTAG Debugger

ARM	**ARM10 Core** **ARM11 Core**	sofort sofort
IBM	**PPC750FX, PPC750GX**	sofort
Philips	**LPC21xx** **(ARM7TDMI basiert)**	sofort
Motorola	**MCS08** **MPC5200** **MPC55xx** **MPC74xx** **MPC85xx**	sofort sofort sofort sofort sofort

Tabelle 2: Neu unterstützte Standard-Prozessorarchitekturen

Auch 2003 hat Lauterbach die Anzahl der Standard-Prozessorarchitekturen, die vom In-Circuit Debugger TRACE32-ICD unterstützt werden, erweitert.

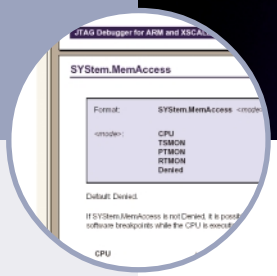
Für den ARM10 und ARM11 ist auch der ETM-Trace bereits lieferbar. Zum Tracen des Programm- und Datenflusses des PowerQUICC III MPC8560 werden für den PowerIntegrator bereits Support Packages angeboten.

Neue NEXUS Debugger

Motorola	**MAC71xx** **MPC5500**	sofort sofort

Tabelle 3: Neu unterstützte NEXUS-Debugger

Lauterbach vervollständigt mit dem NEXUS-Support für den MAC71xx/MPC5500 seine Unterstützung für Motorola Prozessoren mit NEXUS-Technologie und baut so seine führende Marktposition auf diesem Sektor weiter aus.

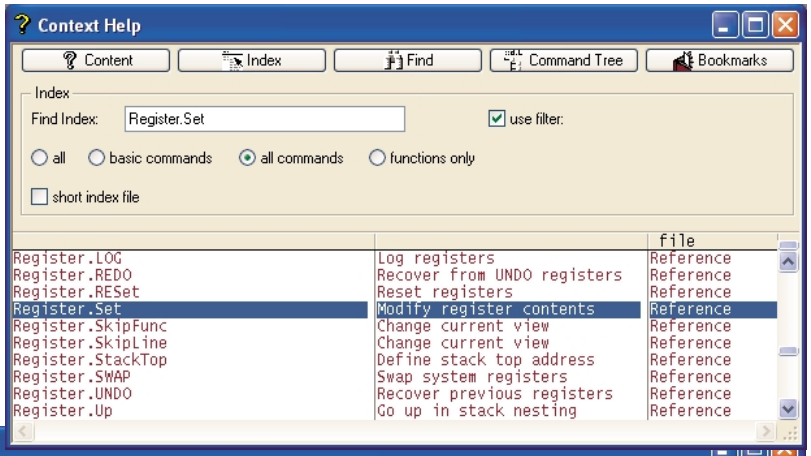


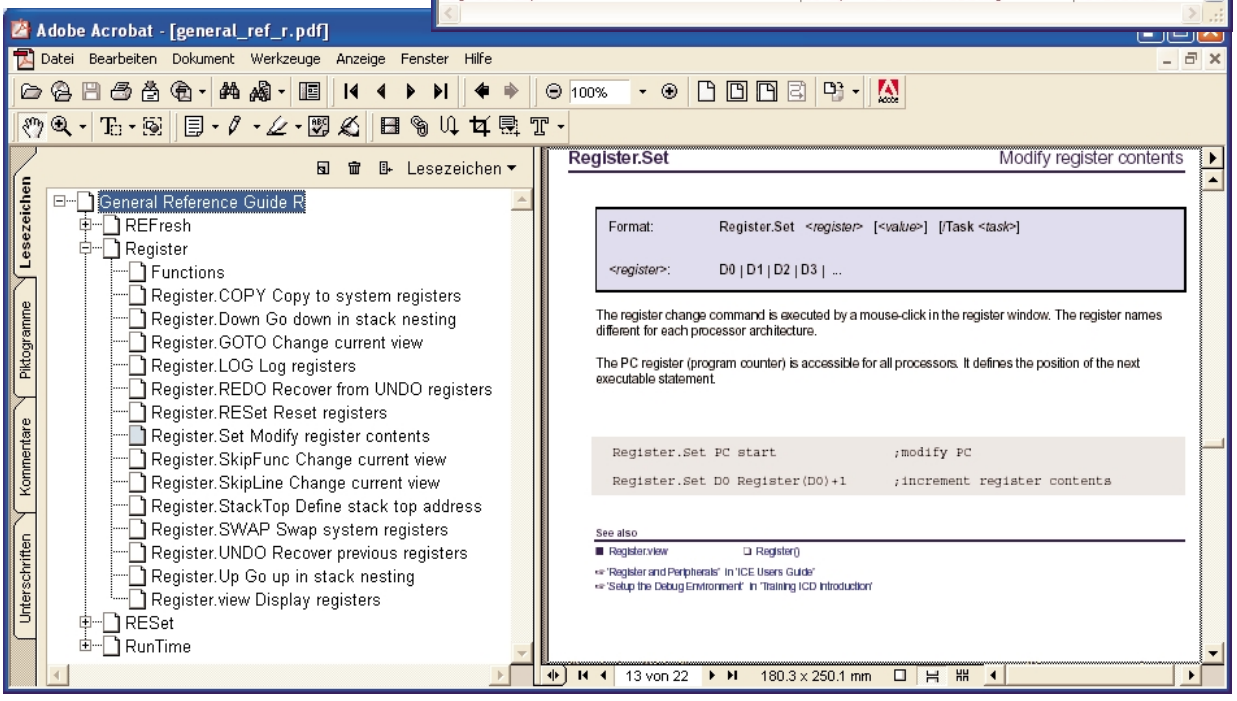
Neue Online-Hilfe

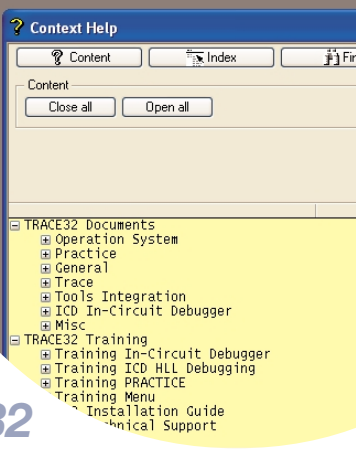
*Auf der **embedded world 2004** in Nürnberg wird Lauterbach seine neue PDF-basierte Online-Hilfe vorstellen. Die wesentlichen Neuerungen sind:*

- *Die Online-Hilfe ist gezielt auf das eingesetzte Entwicklungswerkzeug, die Prozessorarchitektur des Zielsystems und das verwendete RTOS abgestimmt. Selbstverständlich lässt sich die gesamte Online-Hilfe auf Wunsch auch ungefiltert einsetzen.*
- *Ein vollständiges Tooltip-Text-System gibt auf einfache Art wichtige Hinweise zu den Eingabefenstern und Kommandos.*
- *TRACE32-Neueinsteiger haben die Möglichkeit zunächst mit einer Basis-Hilfe zu arbeiten. So gewinnen sie schnell einen Überblick über die wichtigsten Funktionen ihres Entwicklungswerkzeugs.*
- *Die vollständigen Schulungsunterlagen sind in die Online-Hilfe integriert, so dass sich jeder TRACE32-Benutzer anhand von Beispielen gezielt in einzelne Systemfunktionen einarbeiten kann.*
- *Die einzelnen Teile der Online-Hilfe sind über Querverweise vernetzt. Über diese Vernetzung erhält der TRACE32-Anwender schnell eine Übersicht über ähnliche oder ergänzende Kommandos.*
- *Nach einem automatischen Update-Check können die aktuellen Hilfedateien über das Internet nachgeladen werden.*

Selbstverständlich bietet die neue PDF-basierte Online-Hilfe das gleiche „Look-and-Feel" für Windows, LINUX und alle unterstützten Workstations.







Debuggen von „Userland“ Prozessen mit TRACE32

Applikations-Debugging unter Linux, QNX und WindowsCE

User-Prozesse mit einem Hardware-Debugger zu testen ist sowohl für den Debugger, als auch für den Entwickler eine echte Herausforderung. Wird eine Applikation z.B. in Linux gestartet, so wird zunächst ein Prozess erzeugt, die entsprechende Applikation vom Betriebssystem geladen und gelinkt, und schließlich automatisch gestartet. Da der Debugger (und der Entwickler) vor dem Starten der Applikation nicht weiß, wohin der Prozess geladen wird, können auch im Voraus keine Symbole geladen werden. Ohne Symbolinformation lassen sich aber auch keine Breakpoints setzen. Die Applikation läuft also ohne Kontrolle an und eventuell schon auf den ersten Fehler. Zusätzlich zu diesem Problem kommt noch hinzu, dass die Betriebssysteme beim Erzeugen und Starten des Prozesses die MMU neu programmieren. Auch diese Umprogrammierung der MMU muss vom Debugger erkannt werden, um den neuen Prozess dann richtig debuggen zu können.

Bis vor kurzem war eine aufwändige manuelle oder Skript-gesteuerte Befehlssequenz notwendig, um trotz dieser Widrigkeiten einen Prozess debuggen zu können.

1.) Zunächst musste im Kernel des Betriebssystems ein Breakpoint gesetzt werden, der dafür sorgte, dass bei jeder Erzeugung eines Prozesses angehalten wurde.

2.) Nachdem die Programmausführung an dem Breakpoint angehalten hatte, musste geprüft werden, ob der gerade erzeugte Prozess derjenige ist, den man debuggen wollte. Wenn nicht, dann konnte das Programm ohne weitere Aktionen wieder gestartet werden.

3.) Wurde der gewünschte Prozess erzeugt, dann konnte der Debugger zu diesem Zeitpunkt ermitteln, wohin der Prozess geladen wurden. Das ermöglichte dann, die passende Symbolinformation in den Debugger zu laden und einen Breakpoint auf main() zu setzen.

4.) Ein „Go" sollte nun den Prozess starten und bei main() stehen bleiben. Nun musste noch die MMU des Prozesses gescannt werden. Anschließend konnte dann das Debugging des Prozesses beginnen.

*Diese Methode funktioniert zwar, ist aber recht unhandlich. Um das Debuggen von Prozessen zu vereinfachen, hat Lauterbach nun ein **Process Watch System** für verschiedene Betriebssysteme und Prozessoren implementiert. Dieses System erkennt, ob ein zum Debugging anstehender Prozess im System vorhanden ist, gerade gestartet wird oder schon beendet ist. Es lädt und löscht automatisch die jeweiligen Symbolinformationen und die MMU-Tabellen. Der Anwender muss nur noch den Namen der gewünschten Prozesse angeben und ist von allen weiteren Aktionen entlastet.*

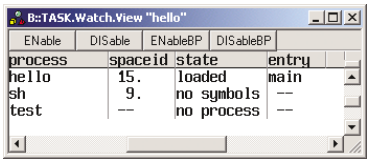


Bild 2: Das Process Watch System überwacht 3 Prozesse

Das Process Watch System kann bis zu zehn Prozesse beobachten. Es wird empfohlen, sich auf eine wesentlich geringere Zahl zu beschränken, da sonst die Reaktionsgeschwindigkeit des Debuggers leidet. Wie im Bild 1 dargestellt, kann durch einfache Anwahl im Menü (im Beispiel Linux) das Process Watch System mit einem zu beobachtenden Prozess aktiviert werden. In der gleichen Weise ist ein Hinzufügen weiterer Prozesse möglich. Bild 2 zeigt ein aktiviertes Process Watch System mit drei Prozessen.

*So arbeitet nun das **Process Watch System**: Wird der Debugger angehalten (und nur dann), sucht der Debugger das Target nach den Prozessen ab. Wird ein entsprechender Prozess gefunden, lädt der Debugger automatisch die Symbolinformation und die entsprechenden MMU-Einträge. Ist der Prozess im Target nicht vorhanden, dann werden evtl. schon geladene Symbole und MMU-Tabellen wieder aus dem Debugger gelöscht. Hat sich die Lade-Adresse des Prozesses geändert (weil er beendet und erneut gestartet wurde), dann erkennt dies der Debugger und veranlasst ein erneutes Laden der Symbole und Scannen der MMU, um den neuen Bedingungen zu entsprechen.*

Falls erforderlich, kann der Debugger das Erzeugen eines neuen Prozesses erkennen und diesen an seinem Einsprungpunkt bei main() anhalten. Damit ist ein Debuggen

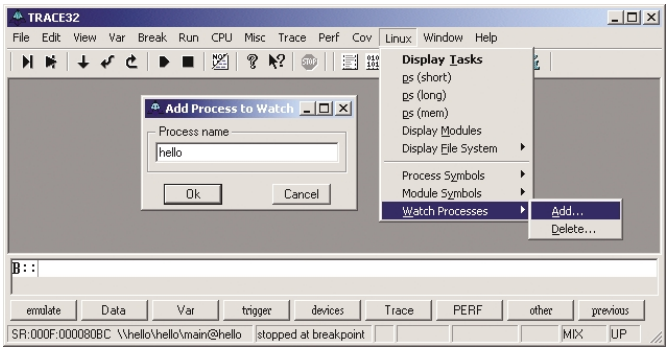
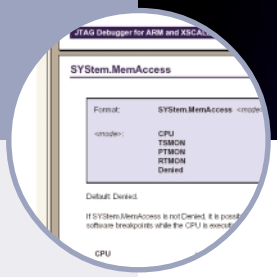


Bild 1: Aktivierung des Process Watch Systems



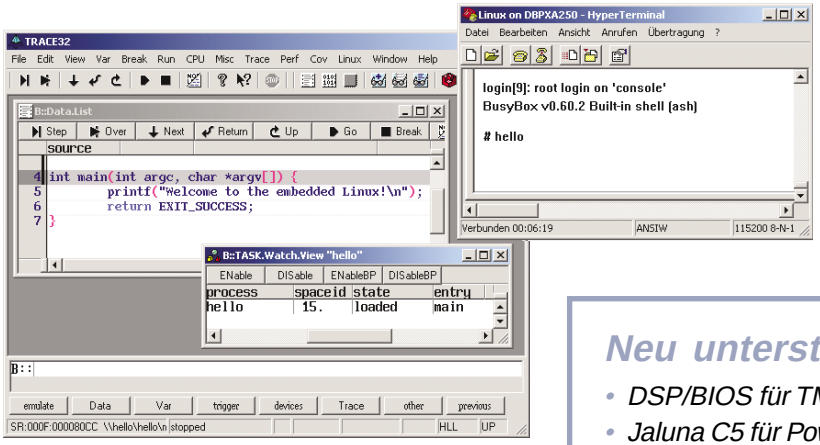


Bild 3: Start eines Prozesses

des Prozesses von Anfang an möglich. Da diese Funktionalität zur Erkennung der Prozess-Erzeugung einen Breakpoint verwendet, der das Echtzeitverhalten des Targets beeinflusst, lässt sie sich auch abschalten. Dann wird zwar nicht mehr automatisch beim Erzeugen des Prozesses angehalten, die Symbole werden nach einem manuellen Break aber dennoch entsprechend geladen.

Bild 3 zeigt die Situation, nachdem der Prozess in der Konsole gestartet wurde. Der Debugger hat auf main() den Prozess angehalten. Nach Beendigung des Prozesses werden die Symbole (beim nächsten Halt) dann wieder aus dem Debugger entfernt (Bild 4). Wird der Prozess erneut gestartet, dann hält der Debugger erneut auf main(), und der Entwickler kann einen neuen Debug-Zyklus abarbeiten.

*Mit dem **Process Watch System** von TRACE32 ist somit ein komfortables Debuggen auch von User-Prozessen möglich. Durch eine integrierte Unterscheidung der User-MMU-Tabellen wird das Debugging mehrerer Prozesse gleichzeitig unterstützt. Da TRACE32 auch Bootstrap, Kernel, Interrupt-Routinen, Module und Libraries debuggen kann, wird damit das Testen der gesamten Target-Software ohne Einschränkungen ermöglicht.*

*Das **Process Watch System** steht bereits auf diversen Prozessoren für Linux, QNX und WindowsCE zur Verfügung.*

Neu unterstützte Echtzeitkerne

- *DSP/BIOS für TMS320C55x und OMAP*
- *Jaluna C5 für PowerPC*
- *OS21 für ST40*
- *OSEck für TMS320C55x und OMAP*
- *RTXC Quadros für ColdFire, geplant auch für ARM und StarCore*
- *ThreadX geplant nun auch für StarCore*
- *Windows CE.net für ARM und XScale, geplant auch für SH4 und MIPS*

3rd Party Tool Integration

- *Windows CE Platform Builder eXDI, geplant*

Monitor Lösungen

- *OSE SDM Run Mode Debugger für PowerPC*
- *GDB front end debugger für Linux, geplant*

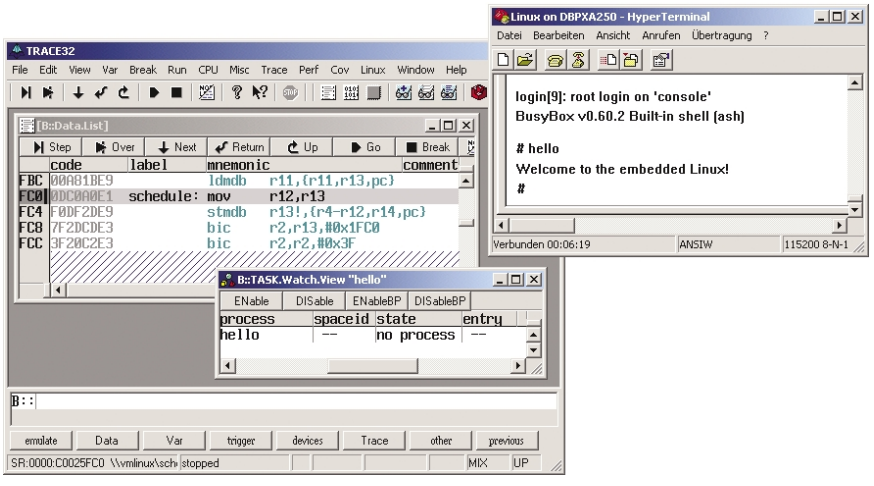


Bild 4: Prozess beendet

Tipps zur Nutzung der ARM-ETM

In vielen Kundengesprächen werden wir immer wieder gebeten, Tipps zur optimalen Dimensionierung und Nutzung der ARM-ETM zu geben. Dieser Artikel versucht nun, die wichtigsten Hinweise zum Einsatz der ETM für den ARM7 und den ARM9 zusammenzufassen, sowie einen Ausblick auf den ARM11 zu geben.

Die ETM (Embedded Trace Macrocell) ist ein On-Chip Echtzeit-Tracemodul für ARM-Architekturen. Dieses Tracemodul bietet folgende Funktionalität:

- *Ausgabe des gesamten Programm- und Datenflusses über ein spezielles Traceport*
- *Trigger und Filter für die Steuerung dieser Ausgabe*

Mit TRACE32-PowerTrace von Lauterbach, einem Entwicklungswerkzeug bestehend aus einem JTAG Debugger und einem 64 MFrames tiefen Tracespeicher, kann diese Traceausgabe aufgezeichnet und ausgewertet werden.

Der Traceport

Die ETM definiert zunächst einmal einen physikalischen Traceport. Siehe dazu Bild 1.

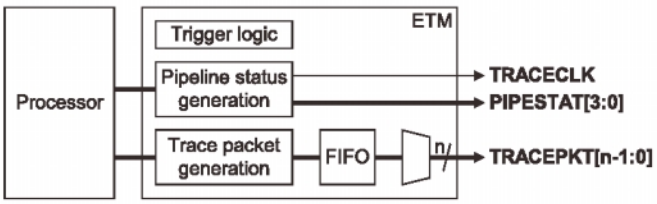


Bild 1: Der Traceport der ETM für ARM7/ARM9

TRACECLK: *In der Regel arbeitet die ETM mit der gleichen Frequenz wie der ARM-Prozessor.*

PIPESTAT: *Über die sogenannten Pipeline-Status-Pins (3 Pins für den ARM7 bzw. 4 Pins für den ARM9) gibt die ETM pro Clockzyklus aus, was in der Instruction-Pipeline des Prozessors passiert (z.B. Instruction taken, branch taken etc.)*

TRACEPKT: *Über TRACEPKT (4-/8- oder 16-Pins) wird auf jeden Fall der Programmfluss sichtbar gemacht. Dazu wird für jeden indirekten Sprung die komprimierte Sprungzieladresse in einem Tracepaket ausgegeben. Der Begriff „indirekter Sprung" ist hier sehr weit gefasst und meint alle Programmverzweigungen, bei denen das Sprungziel erst zur Programmlaufzeit feststeht (beispielsweise Laden des PCs aus einem Register, berechnete Sprungbefehle, Exceptions etc.). Der gesamte Programmfluss kann auf Basis dieser Information unter Zuhilfenahme des Quellcodes rekonstruiert werden. Für Datenzugriffe kann der ETM-Nutzer selbst entscheiden, ob er nur die Datenadresse, nur den Dateninhalt oder beides am Traceport sichtbar machen möchte.*

Während 4 Pins für TRACEPKT durchaus ausreichend sind, um den Programmfluss auszugeben, ist es offensichtlich, dass vor allem die Ausgabe des Datenflusses einen so schmalen Traceport schnell überlastet. Dies ist besonders dann der Fall, wenn mehrere Datenzugriffe kurz hintereinander durchgeführt werden. Um eine Überlastung des Traceports zu verhindern, ist TRACEPKT ein FIFO-Speicher vorgeschaltet. Dieser dient als Zwischenspeicher für alle Tracepakete, die nicht sofort über TRACEPKT ausgegeben werden können.

Dimensionierung der ETM

ARM Limited bietet nun eine Reihe von Standardkonfigurationen für die ARM-ETM an. Aus unserer Erfahrung heraus hat sich eine Portbreite von 8 oder 16 Pins für TRACEPKT, sowie eine FIFO-Größe von mindestens 45 Bytes bewährt (mediumplus- oder large-sized ETM). Für kleiner dimensionierte ETMs muss man vor allem bei der Ausgabe des Datenflusses mit Einschränkungen rechnen, da es hier sehr schnell zu einem FIFO-Überlauf kommen kann.

FIFO-Überlauf

Ein FIFO-Überlauf entsteht immer dann, wenn so viele Tracepakete zur Ausgabe anstehen, dass der FIFO, der TRACEPKT vorgeschalten ist, diese nicht mehr alle zwischenspeichern kann. Dadurch können natürlich wichtige Tracepakete verloren gehen. Um diese sehr unerwünschte Situation zu vermeiden, gibt es zwei Strategien.

1.) FIFOFULL-Logik

Die meisten ARM-ETM Standardkonfigurationen enthalten eine FIFOFULL-Logik. Mit Hilfe dieser Logik ist es möglich, die Programmausführung immer dann anzuhalten (STALL), wenn ein FIFO-Überlauf droht. Diese Strategie hat jedoch den Nachteil, dass sie die Laufzeit des Programms verlängert. Hinzu kommt, dass die ARM-ETM selbst keine Möglichkeit bietet, später bei der Traceauswertung festzustellen, wie hoch der Laufzeitverlust tatsächlich war.

2.) Reduktion der Tracepakete

Eine andere Strategie ist, einen FIFO-Überlauf dadurch zu verhindern, dass man vor allem die Tracepakete für die Datenzugriffe reduziert. Bereits eine Reduktion der Ausgabe der Datenzugriffe auf Schreibzyklen macht einen

FIFO-Überlauf wesentlich unwahrscheinlicher. Möchte man jedoch zur effizienten Fehlersuche den gesamten Datenfluss sehen, empfiehlt Lauterbach die Ausgabe der Tracepakete für die Datenzugriffe auf die Lesezyklen zu reduzieren. TRACE32-PowerTrace kann nämlich durch spezielle Algorithmen sämtliche Schreibzyklen auf Basis der Lesezyklen rekonstruieren. Siehe dazu auch
http://www.lauterbach.com/cts.html

Sollten diese Strategien zur Reduktion der Tracepakete noch nicht ausreichen, um einen FIFO-Überlauf zu vermeiden, kann man noch auf die Filter der ARM-ETM zurückgreifen. Besonders empfehlenswert ist das Herausfiltern des Stackbereichs aus dem Datenfluss. Denn vor allem die Stackoperationen bei Funktionswechseln führen oft zu einem FIFO-Überlauf. Eine weitere sehr naheliegende Idee ist, die Tracefilter so zu programmieren, dass nur noch Datenzugriffe auf ausgewählte Variablen/Speicherbereiche sichtbar gemacht werden.

Smart Trace

Selbst eine größere Anzahl von Tracelücken durch FIFO-Überlauf ist beim Einsatz von TRACE32-PowerTrace eher unproblematisch. Ein spezielles Verfahren - SmartTrace - erlaubt ein Füllen der meisten Tracelücken. Einzig Exceptions und Lesezugriffe auf Ports können nicht immer vollständig rekonstruiert werden. Siehe dazu auch Bild 2 sowie
http://www.lauterbach.com/smarttrace.html

Tipps für die Zielhardware

Um eine optimale Aufzeichnung der Traceinformationen durch TRACE32-PowerTrace zu erreichen, sollten folgende Hinweise für das Design der Zielhardware beachtet werden:

- *möglichst kurze und gleich lange Leitungen für die ETM-Signale*
- *3 ns Setup- und 1 ns Hold-Zeit*
- *das Zielsystem sollte eine 50 Ohm Parallel-Terminierung treiben können*

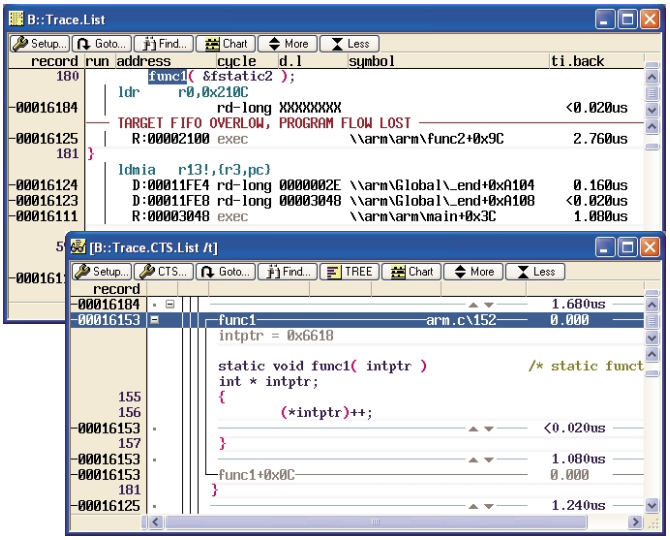


Bild 2: Füllen einer Tracelücke durch Rekonstruktion des Programm- und Datenflusses mit SmartTrace

ETM für ARM11

Die ETM für den ARM11 bietet entscheidende Neuerungen.

1.) Höhere Datenkompression

Die Datenmenge für die Ausgabe des Programmflusses wurde um den Faktor 7, die Datenmenge für die Ausgabe des Datenflusses wurde um durchschnittlich 25% reduziert.

2.) Entkopplung des Traceclocks vom Prozessorclock

Die Traceinformation, im speziellen die Pipeline-Status-Information wird nicht mehr synchron zum Prozessorclock ausgegeben. Stattdessen ist die Pipeline-Status-Information als Header in den Tracepaketen enthalten.

3.) Unterdrückung der Datenpakete bei FIFO-Überlauf

Statt bei einem drohenden FIFO-Überlauf die Programmausführung anzuhalten, wird hier die Ausgabe des Datenflusses unterdrückt.

Nachdem die ersten Projekte mit dem ARM11 bei unseren Kunden gerade erst anlaufen, können wir für diese ETM-Version heute noch keine Empfehlungen aussprechen.

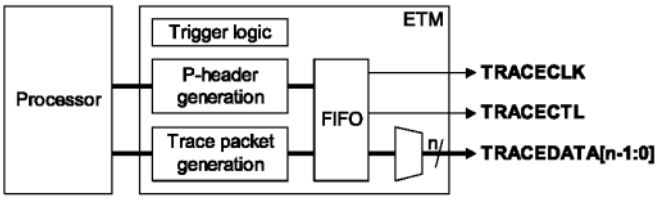
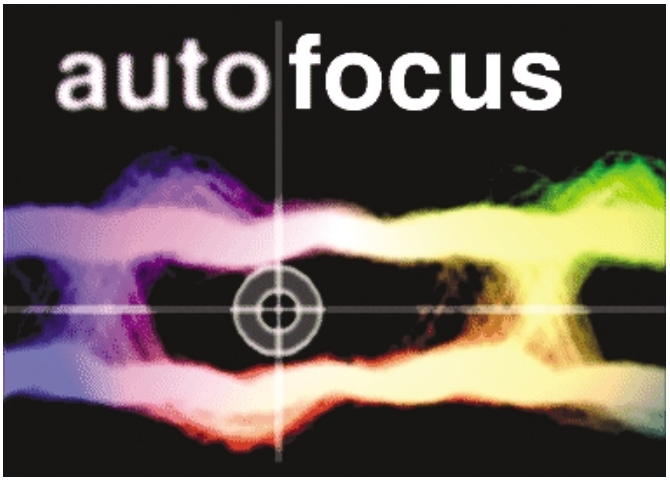


Bild 3: Der Traceport der ETM für ARM11

Optimaler Abtastzeitpunkt für Tracekanäle



Mikrocontroller, die die Traceinformation mit einer Frequenz von mehr als 200 MHz über den Traceport ausgeben, sind heute keine Seltenheit mehr. Das Zeitfenster für die Abtastung der gültigen Daten wird so immer kleiner. Kleine Laufzeitunterschiede zwischen den Tracekanälen oder andere geringfügige Störungen auf der Zielhardware werden so leicht zur Fehlerquelle für die Traceaufzeichnung. Eine Feinabstimmung des Abtastzeitpunkts gewinnt dadurch immer mehr an Bedeutung. Aus diesem Grund wird Lauterbach Mitte 2004 eine neue Generation von Preprozessoren und NEXUS-Adaptern auf den Markt bringen, die über eine Autofocus-Funktionalität verfügen. Autofocus unterstützt die automatische Einstellung des optimalen Abtastzeitpunkts durch eine FPGA-basierte Hardware.

Eine der Hauptaufgaben eines Preprozessors/NEXUS-Adapters ist es, die über den Traceport des Mikrocontrollers ausgegebenen Traceinformationen vom Zielsystem abzugreifen und in den Tracespeicher zu übertragen. Die Qualität der Tracesignale kann bei hohen Frequenzen durch vielerlei Faktoren beeinträchtigt werden. Dazu gehören: Leitungslänge, Resonanzen, Übersprechen von anderen Signalen, Toleranzen der Bauteile etc. Wenn der Abtastzeitpunkt für die Tracedaten dann nicht optimal gewählt ist, erhöht sich die Fehlerrate bei der Aufzeichnung des Programm- und Datenflusses.

Bisher wurde bei den meisten Preprozessoren/NEXUS-Adaptern lediglich die Referenzspannung automatisch eingestellt. Zur Verringerung der Fehlerrate konnte die Referenzspannung manuell nachjustiert werden; zur Verbesserung der Signalamplitude war das Einschalten einer Terminierungsspannung möglich. Für die bei hohen Frequenzen erforderliche Feinabstimmung des Abtastzeitpunkts ist jedoch ein wesentlich komplexeres Verfahren notwendig.

Die Feinanpassung des Abtastzeitpunkts erfolgt dabei in zwei Schritten:

1. Hardwareabgleich

Sobald der Mikrocontroller über den Traceport Daten ausgibt, ermitteln der Preprozessor/NEXUS-Adapter mittels einer FPGA-basierten Hardware folgendes:

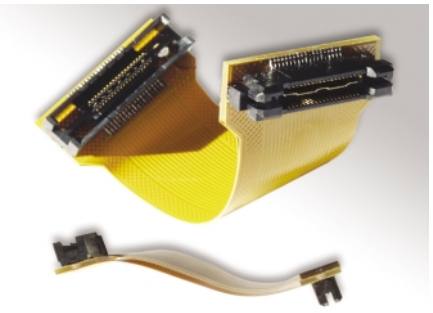
- *die ideale Terminierungsspannung*
- *die ideale Referenzspannung separat für alle Clock- und Datenkanäle*
- *das ideale Clock-Delay*
- *sowie das ideale Delay für jede einzelne Datenleitung*

2. Test der Tracekanäle

Zum Test der Tracekanäle muss der Mikrocontroller anschließend über den Traceport ein festes Datenmuster ausgeben. Diese Daten werden dann vom Preprozessor/NEXUS Adapter in den Tracespeicher übertragen und können anschließend von der TRACE32 Software verifiziert werden. So lässt sich nach der automatischen Einstellung des idealen Abtastzeitpunkts überprüfen, ob alle Tracekanäle auf dem Zielsystem fehlerfrei arbeiten. Bei Bedarf kann die Feinabstimmung des Abtastzeitpunkts für einzelne Tracekanäle nachjustiert werden.

Der erste NEXUS-Adapter, der mit einer Autofocus-Funktionalität ausgestattet ist, wird der NEXUS-Adapter für den Super10 sein. Der Preprozessor für die ARM-ETM wird dann folgen.

Flex-Extensions



Mictor-Flex-Extension, 55 mm, vertikal auf vertikal

Für die verschiedenen Preprozessoren/NEXUS Adapter bietet Lauterbach seit Mitte 2003 sogenannte Flex-Extensions an. Diese Verlängerungen garantieren auch bei hohen Frequenzen eine gute Übertragungsqualität zwischen dem Zielsystem und dem TRACE32 Entwicklungswerkzeug.

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

Complex-Trigger-Einheit

Wer sich bei der Software-Integration mit komplexen Fehlerszenarien oder sporadisch auftretenden Systemabstürzen herumschlagen muss, weiß die Vorteile einer leistungsfähigen Triggereinheit zu schätzen. Dazu gehören:

- *strategische, planvolle Fehlersuche*
- *gezieltes Anhalten der Traceaufzeichnung oder der Programmausführung*
- *optimale Ausnutzung des Tracespeichers*
- *gezielte Aufzeichnung und Auswertung der interessanten Informationen*
- *verbesserte Laufzeitüberwachung des Gesamtsystems*

Natürlich erfordert der effiziente Einsatz einer Triggereinheit gute Kenntnisse der Hard- und Software des Zielsystems, um daraus geeignete Triggerstrategien zu entwickeln. Für die investierte Arbeit wird man jedoch immer mit einer erheblichen Zeitersparnis belohnt.

Um den Entwickler beim Auffinden schwieriger Systemfehler besser zu unterstützen, sind die folgenden Lauterbach Entwicklungswerkzeuge seit November 2003 mit einer sogenannten Complex-Trigger-Einheit ausgestattet:

- ***PowerTrace***
- ***PowerProbe***
- ***PowerIntegrator***

Im folgenden sollen die Trigger- und Filterfunktionen dieser Triggereinheit für die verschiedenen Familienmitglieder der PowerTools-Familie kurz vorgestellt werden.

PowerTrace

Obwohl die Complex-Trigger-Einheit ein fester Bestandteil der PowerTrace-Hardware ist, ist eine spezielle Softwareanpassung für die einzelnen Prozessorarchitekturen notwendig.

1. Bustrace-Anpassungen

Am einfachsten anzupassen sind natürlich Architekturen, die über den externen Adress- und Datenbus den gesamten Programm- und Datenfluss sichtbar machen. Ein Beispiel für eine solche Architektur ist der EGOLD von Infineon. Sollen beispielsweise nur die Zyklen einer bestimmten Funktion oder nur die Schreibzugriffe auf eine bestimmte Variable aufgezeichnet werden, kann die am Adressbus sichtbare Adresse direkt als Eingangsereignis für die Complex-Trigger-Einheit verwendet werden. Siehe dazu auch Bild 1.

2. FlowTrace-Anpassungen

Prozessorarchitekturen, die den Programm- und Datenfluss über einen speziellen Traceport ausgeben, arbeiten meist mit stark komprimierten Tracedaten. Diese Kompression ermöglicht es, auch bei hohen Frequenzen den gesamten Programm- und Datenfluss über einen schmalen Traceport sichtbar zu machen.

Um diese stark komprimierten Tracedaten für die Complex-Trigger-Einheit verwenden zu können, ist eine vollständige Rekonstruktion der Programm- und Datenadressen zur Programmlaufzeit notwendig. Aktuell kann nur für die SH4-Familie von Hitachi eine solche Laufzeit-Rekonstruktion problemlos durchgeführt werden. Für alle anderen Architekturen bleibt man wie bisher auf die vom Halbleiterhersteller im Prozessor implementierten Filter- und Triggermöglichkeiten angewiesen.

3. NEXUS-Anpassungen

Für Architekturen, die über eine NEXUS-Schnittstelle verfügen, ist eine Laufzeit-Rekonstruktion der Programmadressen nicht möglich. Eine Rekonstruktion der Datenadressen kann jedoch in der Regel durchgeführt werden.

Das NEXUS-Protokoll bietet aber eine Reihe von Messagetypen, die sich gut als Eingangsereignisse für die Complex-Trigger-Einheit nutzen lassen:

- *Watchpoint-Hit-Messages*
- *Ownership-Trace-Messages*
- *Hardware-Event-Messages etc.*

Über eine Watchpoint-Hit-Message kann die NEXUS-Schnittstelle beispielsweise anzeigen, dass eine bestimmte Funktion einen definierten Datenwert auf eine Variable geschrieben hat. Die Watchpoint-Hit-Message kann dann

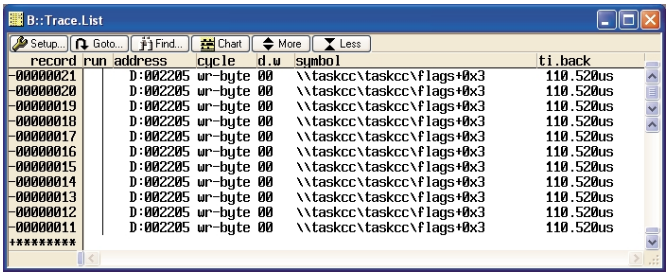


Bild 1: Selektiver Datentrace für einen Bustrace (EGOLD von Infineon)

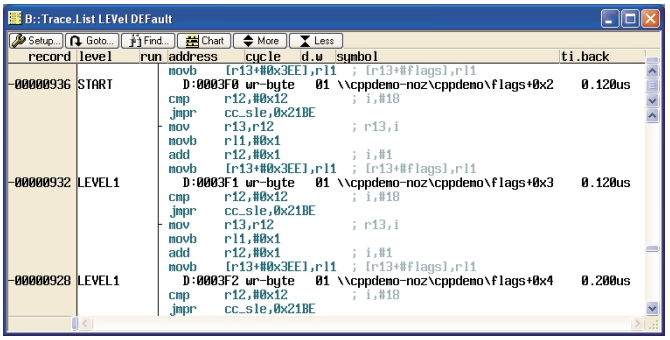


Bild 2: Wechsel der Triggerebene nach Schreiben von 1 auf flags[2] (NEXUS Super10 von STMicroelectronics)

von der Complex-Trigger-Einheit dazu verwendet werden, noch eine bestimmte Anzahl von Zyklen im Trace aufzuzeichnen und anschließend das Programm zu stoppen. Ein Beispiel für den Einsatz der Complex-Trigger-Einheit für NEXUS zeigt Bild 2.

Nachdem eine Anpassung für die Prozessorarchitektur auf dem PowerTrace durchgeführt wurde, bietet die Complex-Trigger-Einheit folgende Möglichkeiten:

- *Filtern der Traceinformation*
- *Definition von komplexen Triggerbedingungen, die zum Abbruch der Traceaufzeichnung oder der Programmausführung führen können*
- *Zeitüberwachung einzelner Programmabschnitte*
- *Generierung eines Triggersignals für ein externes Gerät*

Die Complex-Trigger-Einheit verfügt dabei über folgende Ressourcen:

- *mindestens 2 Adress-/Datenkomparatoren*
- *3 x 45-Bit Zeit- und Ereigniszähler*
- *4 Triggerebenen*
- *2 Trigger-Flags*
- *2 Trace-Marken*

PowerProbe

Mit der PowerProbe bietet Lauterbach die Möglichkeit, Portleitungen, Kommunikationsschnittstellen sowie externe Interrupts in die Debug-Umgebung zu integrieren. Dabei werden 64 Eingangskanäle bis 100 MHz bzw. 16 Eingangskanäle bis 400 MHz unterstützt. Die Tracetiefe beträgt 256K Einträge.
Einzelne Eingangskanäle oder zu einem Triggerwort (max. 64-Bit) zusammengefasste Eingangskanäle können nun als Eingangsereignis für die Complex-Trigger-Einheit verwendet werden. Zusammen mit drei 45-Bit Zeit-/Ereigniszählern, 4 Triggerebenen etc. können nun komplexe Triggerbedingungen definiert werden, die:

- *die Traceinformation filtern*
- *die Aufzeichnung in den Tracespeicher abbrechen*
- *den Patterngenerator starten*
- *Triggersignale für externe Geräte generieren*
- *oder dafür sorgen, dass der Debugger die Programmausführung stoppt*

Bild 3 zeigt ein Beispiel, in dem die Zustände der Eingangskanäle x.0-x.2 nur dann im Tracespeicher aufgezeichnet werden, solange der Eingangskanal x.3 aktiv ist.

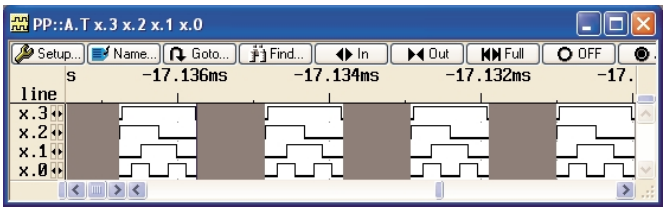


Bild 3: Selektive Aufzeichnung von x.0-x.2, solange x.3 aktiv ist

PowerIntegrator

Die Hauptmotivation für die Markteinführung des PowerIntegrators im Sommer 2003 war die Idee, die PowerTools-Produktfamilie um einen universellen Bustrace zu erweitern. Der PowerIntegrator kann dazu über Standard-Probes an das Zielsystem angeschlossen werden. Gleichzeitig bieten sogenannte Support-Packages die Möglichkeit, die aufge-

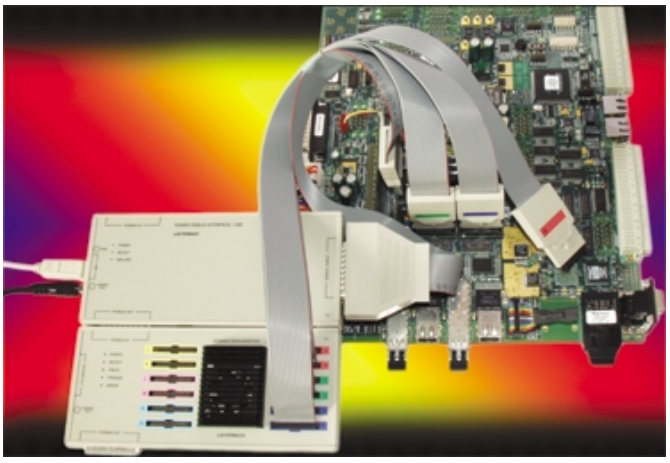


Bild 4: In-Circuit Debugger TRACE32-ICD und PowerIntegrator an einem PowerQUICC III Evaluation Board

TRACE32-ICD
PowerLine
die komplette Umgebung
für System-on-Chip Designs

Weitere Informationen im Internet: www.lauterbach.com

zeichneten Eingangssignale einfach und schnell in einen Programm- und Datenfluss-Trace zu übersetzen. Der so aufbereitete Traceinhalt kann dann gemeinsam mit der Debug-Information dargestellt und ausgewertet werden.

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***

Die Abtastschwelle für die Signale kann dabei frei in einem Bereich von 0 .. 5 V gewählt werden.

Die Tracespeichertiefe des PowerIntegrator beträgt 512 KFrames. Jeder Eintrag im Tracespeicher wird mit einem Zeitstempel versehen. Der Zeitstempel hat eine Breite von 48 Bit und arbeitet mit einer Auflösung von 4 ns.

Ähnlich wie bei der PowerProbe können einzelne Datenkanäle oder zu einem Triggerwort (max. 204 Bit) zusammengefasste Datenkanäle als Eingangsereignis für die Complex-Trigger-Einheit verwendet werden. Die wichtigsten Triggerworte beim Einsatz des Power-Integrators sind sicherlich der Adress- und Datenbus. Beide Triggerworte sind bereits durch das Support-Package vordefiniert und stehen so direkt zur Verfügung.

Als Beispiel für den Einsatz der Complex-Trigger-Einheit beim Arbeiten mit dem PowerIntegrator soll die Zeitüberwachung einer Funktion dienen. Für die Einsprung- und Aussprungadresse der Funktion wird je ein Triggerwort definiert. Durch Filterung werden im Tracespeicher nur noch die Einsprung- und Aussprungadresse der Funktion aufgezeichnet. Das Ergebnis kann dann vom Power-Integrator auch statistisch ausgewertet werden. Siehe dazu auch Bild 5.

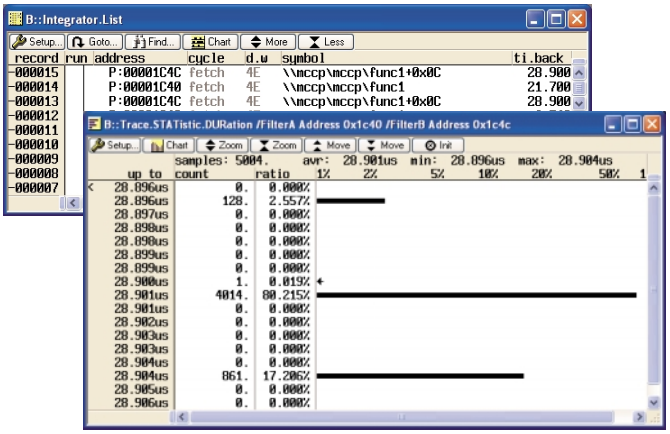


Bild 5: Zeitüberwachung einer Funktion durch selektive Aufzeichnung der Funktionseintritte und Austritte

Bitte schicken Sie mir Informationsmaterial zu:

Wir setzen folgende Prozessoren ein:

_____

Ich interessiere mich für folgende Entwicklungswerkzeuge:

- ☐ ***TRACE32-ICD***
- ☐ ***TRACE32-PowerTrace***
- ☐ ***TRACE32-PowerIntegrator***

Absender:

Name _____

Firma _____

Adresse _____

_____

Telefon _____

Email _____

LAUTERBACH

FAX: ++49 8104 8943-187