Deutschland Lauterbach GmbH
Frankreich Lauterbach S.A.R.L.
Großbritannien Lauterbach Ltd.
Italien Lauterbach Srl
USA Lauterbach Inc.
China Suzhou Lauterbach Technologies Co. Ltd.
Japan Lauterbach Japan Ltd.

Debugging mit ARM-CoreSight

Am Beispiel von ARM-CoreSight sollen die Debug- und Tracekonzepte für heterogene Multicore-Prozessoren vorgestellt werden.

Um die vielfältigen Aufgaben innerhalb eines embedded Systems zu bearbeiten, werden heute oftmals Prozessoren eingesetzt, die verschiedenartige Cores beinhalten. Damit sich ein solches System gut debuggen lässt, müssen zwei Voraussetzungen erfüllt sein:

1. Der Multicore-Prozessor muss über eine geeignete *On-Chip Debug/Trace Logic* verfügen.

2. Die Entwicklungsumgebung muss das Debugging der einzelnen Cores und des Gesamtsystems mit intelligenten Test- und Analysefunktionen unterstützen.

Dieser Artikel beschreibt, wie die TRACE32-Entwicklungsumgebung im Zusammenspiel mit der *On-Chip Debug/Trace Technology CoreSight* diese Anforderungen meistert.

Was ist CoreSight?

CoreSight heißt die *On-Chip Debug/Trace Technology*, die ARM speziell für Multicore-Prozessoren zur Verfügung stellt. CoreSight ist jedoch nicht als ein fester Logik-Block konzipiert, sondern stellt wie ein Baukasten verschiedene Komponenten zur Verfügung. Der Designer des Multicore-Prozessors kann so selbst den Leistungsumfang für das Debugging und Tracing bestimmen.

CoreSight bietet einen großen Gestaltungsfreiraum. Um die passenden Debug- und Tracemöglichkeiten auf dem Prozessor zu integrieren, ist oft das Spezialwissen des Toolherstellers gefragt. Seit mehreren Jahren beraten unsere Experten weltweit Kunden während der Designphase des Prozessors zu diesem Thema.

Das CoreSight-Baukastenkonzept wirkt sich natürlich auch auf das eingesetzte Entwicklungswerkzeug aus. Kennt dieses den Prozessor und seine CoreSight-Komponenten, ist alles ganz einfach. Für neue Prozessoren erfordert das Baukastenkonzept jedoch eine hohe Flexibilität des Werkzeugs. Die CoreSight-Konfigurationsinformationen lassen sich zwar aus dem Prozessor auslesen, dennoch ist es oft notwendig, Implementierungsdetails mit dem Designer des Prozessors abzuklären.

Für die folgenden Ausführungen wurde ein heterogener Multicore-Prozessor bestehend aus den RISC-Cores ARM11 und Cortex-A sowie einem Ceva-X DSP gewählt.

CoreSight-Debug

Für Prozessoren mit CoreSight erfolgt das Debugging aller Cores über eine gemeinsame JTAG-Schnittstelle. Eine Entwicklungsumgebung für unseren Beispielprozessor umfasst folgende TRACE32-Produkte (siehe Bild 1):

- Ein universelles PowerDebug Modul, das über eine USB- bzw. Ethernet-Schnittstelle an den Hostrechner angeschlossen wird

- Ein Debug-Kabel mit Lizenzen für die Architekturen ARM11, Cortex-A und Ceva-X

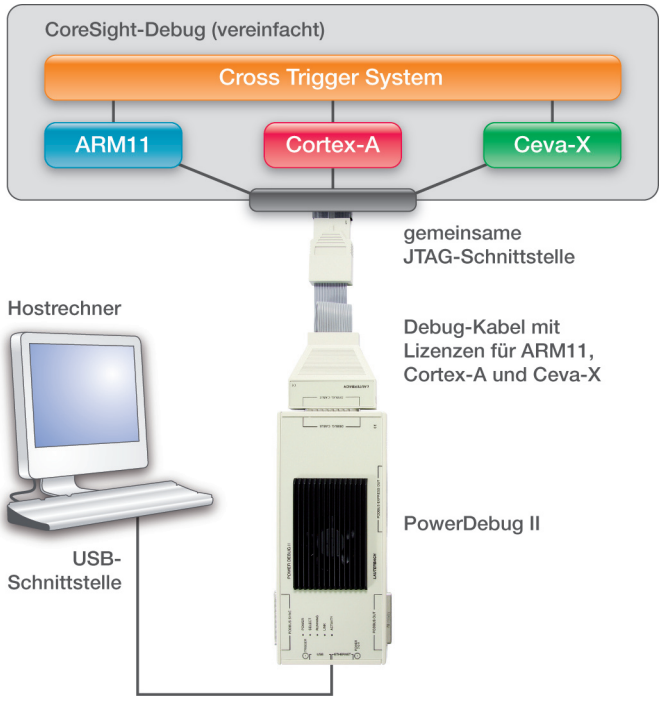

Bild 1: Eine TRACE32-Entwicklungsumgebung für CoreSight-Debug

In heterogenen Multicore-Prozessoren arbeiten die einzelnen Cores in der Regel relativ unabhängig voneinander an ihren Aufgaben. So ist es sinnvoll für das Debugging jedes Cores eine eigene TRACE32-Instanz zu starten (siehe Bild 2 auf der nächsten Seite).

Um das Zusammenspiel der Cores zu testen, muss jedoch auch die Möglichkeit für ein Core-übergreifendes Debugging bestehen. Dafür stellt CoreSight ein *Cross Trigger System* zur Verfügung, welches das synchrone Debugging aller Cores ermöglicht: Hält ein Core an einem Breakpoint, werden die anderen Cores synchron dazu »

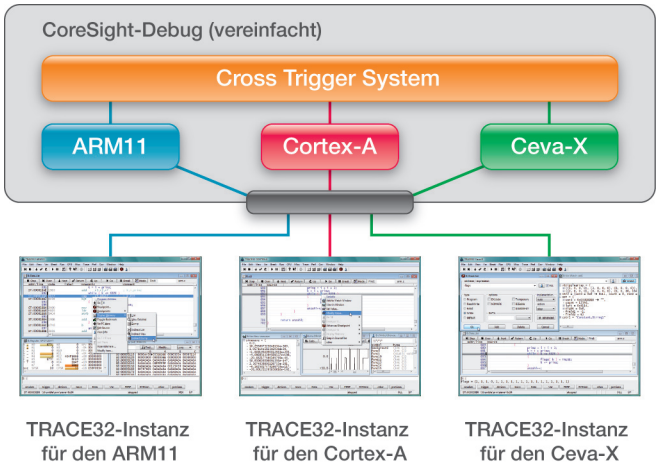

Bild 2: Für das Debugging jedes Cores wird eine eigene TRACE32-Instanz gestartet

TRACE32-Features für CoreSight-Debug

- Flexible Unterstützung für Multicore-Prozessoren mit CoreSight; Lauterbach bietet Debugger für alle ARM-/Cortex-Cores sowie eine Vielzahl von DSPs
- Debugging über die JTAG-Schnittstelle und den *Serial Wire Debug Port*
- Laufzeit-Zugriff auf den physikalischen Speicher und die Peripherieregister
- Synchrones Debugging aller Cores und der Peripherie
- Der *Power Down Mode* eines Cores hat keine Auswirkungen auf das Debugging der übrigen Cores

gestoppt. Auf diese Weise lässt sich der Kontext der einzelnen Cores an einer ausgewählten Programmstelle gut visualisieren.

Über diese Basisfunktion für das Multicore-Debugging hinaus, können abhängig von der CoreSight-Konfiguration weitere nützliche Debug-Funktionen von TRACE32 bereitgestellt werden. Eine Zusammenfassung aller TRACE32-Features für CoreSight-Debug zeigt der Kasten rechts.

CoreSight-Trace

Für die Traceinformation aller Cores steht ebenfalls eine gemeinsame Schnittstelle zur Verfügung. Unter CoreSight kann jedem Core eine Komponente zur Generierung von Traceinformationen zugeordnet sein. Für unseren Beispielprozessor sind dies die Komponenten:

- ARM ETM für den ARM11 und den Cortex-A
- Ceva-X ETM für den Ceva-X (siehe auch Bild 3)

Jede Tracekomponente generiert Informationen darüber, welche Instruktionen ihr Core ausgeführt hat und welche Datenzugriffe durchgeführt wurden. Um diese Traceinformation an der gemeinsamen Schnittstelle zur Verfügung zu stellen, fasst der so genannte *Funnel* die Tracedaten zu einem einzigen Datenstrom zusammen. Dieser wird dann entweder über einen Traceport ausgegeben oder in einem On-Chip Tracespeicher abgelegt.

Off-Chip Traceport

Über 18 Prozessorpins, 16 Pins für die eigentliche Traceinformation und zwei Kontrollsignale, können die Tracedaten aller Cores für ein externes Tracetool ausgegeben werden.

Für die off-chip Aufzeichnung und Auswertung durch TRACE32 muss die Entwicklungsumgebung aus Bild 1 um folgende Produkte ergänzt werden (siehe Bild 4):

- Ein universelles PowerTrace II Modul, das den bis zu 4 GByte großen Tracespeicher zur Verfügung stellt.
- Einen *Preprocessor AutoFocus II* für das Abgreifen der Tracedaten am Traceport. Dabei muss der *Preprocessor AutoFocus II* Tracelizenzen für die ARM ETM und die Ceva-X ETM enthalten. »

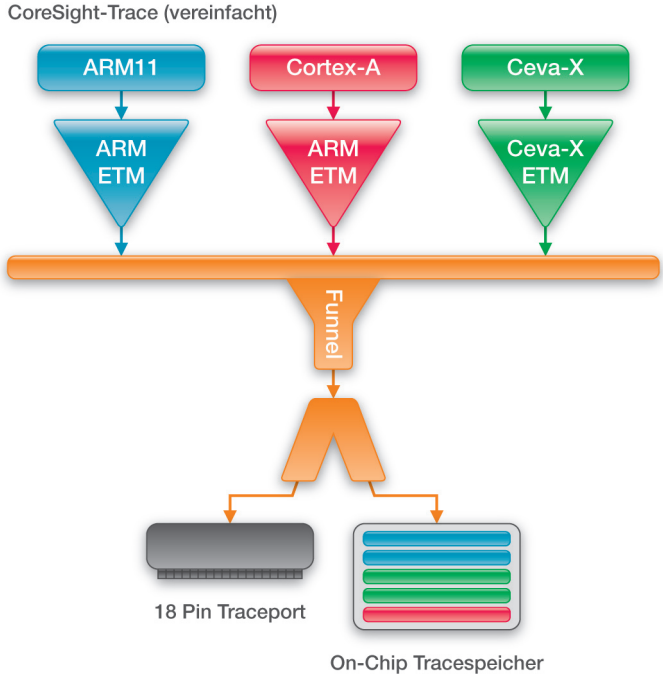

Bild 3: Jeder Core generiert seine eigene Traceinformation

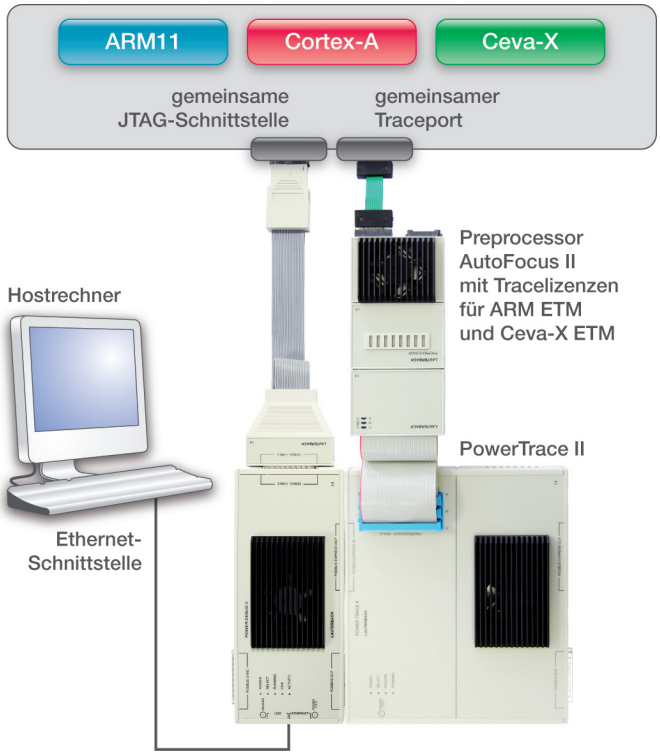

Bild 4: Eine TRACE32-Entwicklungsumgebung für CoreSight-Debug und CoreSight-Trace

On-Chip Tracespeicher ETB

Eine Pin-sparende Alternative zum Traceport bildet der On-Chip Tracespeicher, der unter CoreSight ETB – *Embedded Trace Buffer* – heißt. Sein Fassungsvermögen ist aber im Vergleich zu einem externen Tracetool wesentlich kleiner, in der Regel nur 2-8 KByte.

Werden die Tracedaten im ETB abgespeichert und anschließend über die JTAG-Schnittstelle ausgelesen, muss das Debug-Kabel aus Bild 1 noch Tracelizenzen für den ARM ETB und den Ceva-X ETB enthalten.

Traceauswertung

Nach der Aufzeichnung kann der Entwickler die Tracedaten für jeden einzelnen Core darstellen und analysieren. Dazu liest jede TRACE32-Instanz ihre Tracedaten aus dem gemeinsamen Tracespeicher aus (siehe Bild 5).

Zur Analyse des Zusammenspiels der Cores lässt sich die Tracedarstellung so konfigurieren, dass die Traceeinträge aller Cores in einen direkten zeitlichen Zusammenhang gesetzt werden können. Wird beispielsweise ein Traceeintrag in der ARM11-Instanz ausgewählt, markieren die beiden anderen TRACE32-Instanzen die Instruktion, die der ihnen zugeordnete Core zur selben Zeit ausgeführt hat.

Abhängig von der CoreSight-Konfiguration können weitere Tracefunktionen von TRACE32 angeboten werden. Eine Zusammenfassung finden Sie im Kasten unten.

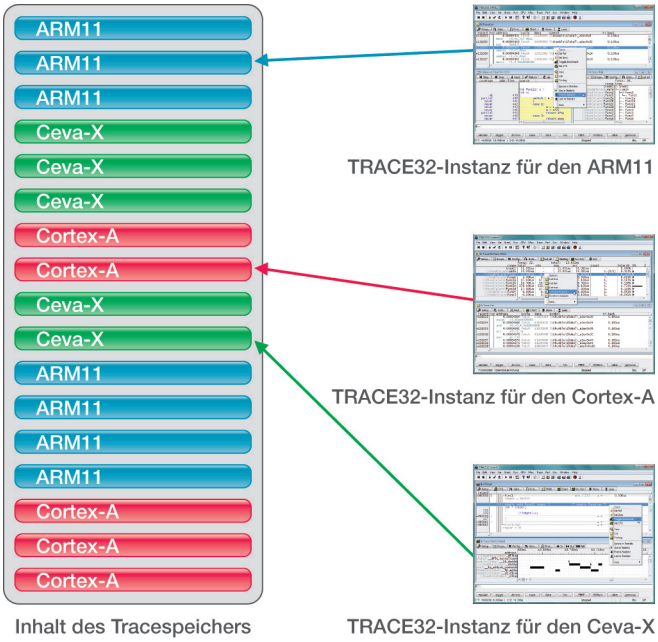

Bild 5: Jede TRACE32-Instanz stellt die Tracedaten ihres Cores dar

TRACE32-Features für CoreSight-Trace

- Flexible Unterstützung für Multicore-Prozessoren mit CoreSight; TRACE32 unterstützt die Auswertung von Traceinformationen für die ARM ETM und eine Vielzahl von DSP ETMs
- Tracen von Buszyklen des AMBA AHB-Busses
- Tracen von Daten, welche die Anwendung mit Hilfe der *Instrumentation Trace Macrocell* (ITM) ausgibt
- Ausgabe der Tracedaten über einen Traceport oder Abspeicherung in dem On-Chip Tracespeicher
- Die einzelnen Komponenten zur Tracedatengenerierung können sich gegenseitig über das *Cross Trigger System* anstoßen
- Zeitkorrelierte Visualisierung der Tracedaten für die einzelnen Cores
- Code-Coverage und umfassende Laufzeitanalysen