NEWS 2018

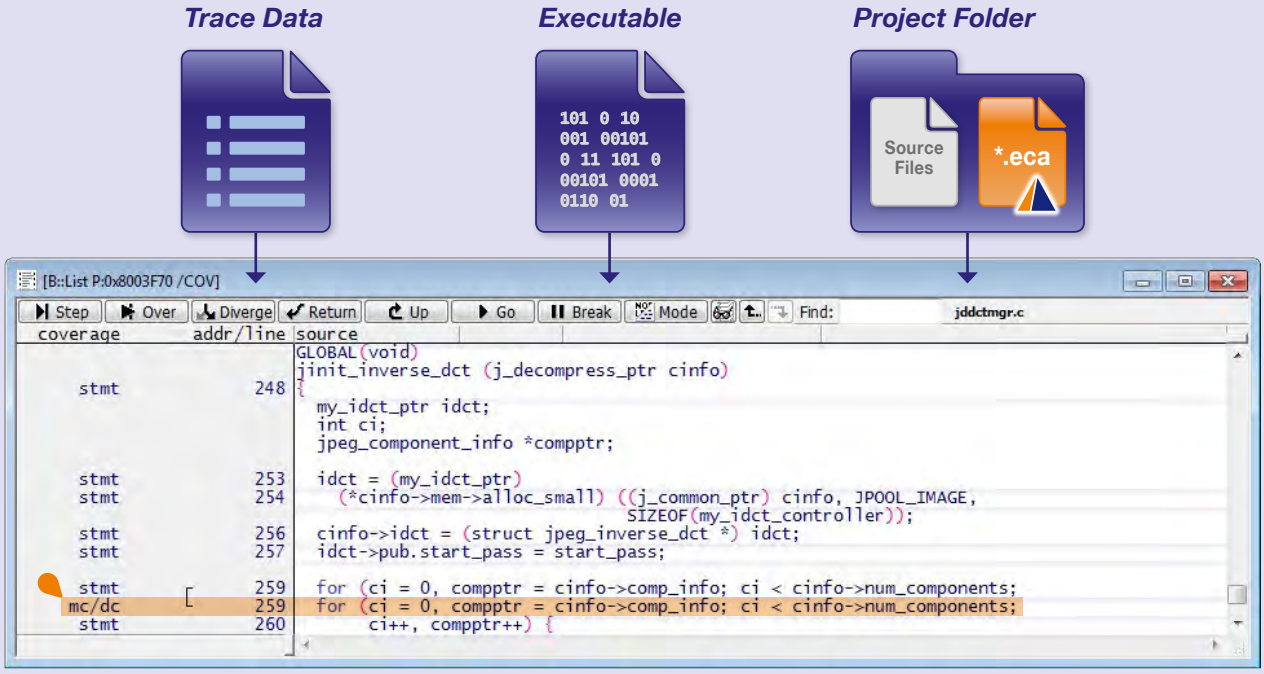
Deutsche Ausgabe

INHALT

LAUTERBACH
DEVELOPMENT TOOLS

Tracebasierte MCDC Coverage

MCDC Coverage



Auf der embedded world 2018 in Nürnberg präsentiert Lauterbach erstmals seine tracebasierte MCDC-Coverage. Damit unterstützt TRACE32 jetzt alle wichtigen Code-Coverage-Metriken auf Quellcodeebene.

Definitions According to DO-178C [3]

Statement Coverage: Every statement in the program has been invoked at least once.

Decision Coverage: Every point of entry and exit in the program has been invoked at least once and every decision in the program has taken on all possible outcomes at least once.

Modified Condition/Decision Coverage: Every point of entry and exit in the program has been invoked at least once, every condition in a decision in the program has taken all possible outcomes at least once, every decision in the program has taken all possible outcomes at least once, and each condition in a decision has been shown to independently affect that decision's outcome. A condition is shown to independently affect a decision's outcome by: (1) varying just that condition while holding fixed all other possible conditions, or (2) varying just that condition while holding fixed all other possible conditions that could affect the outcome.

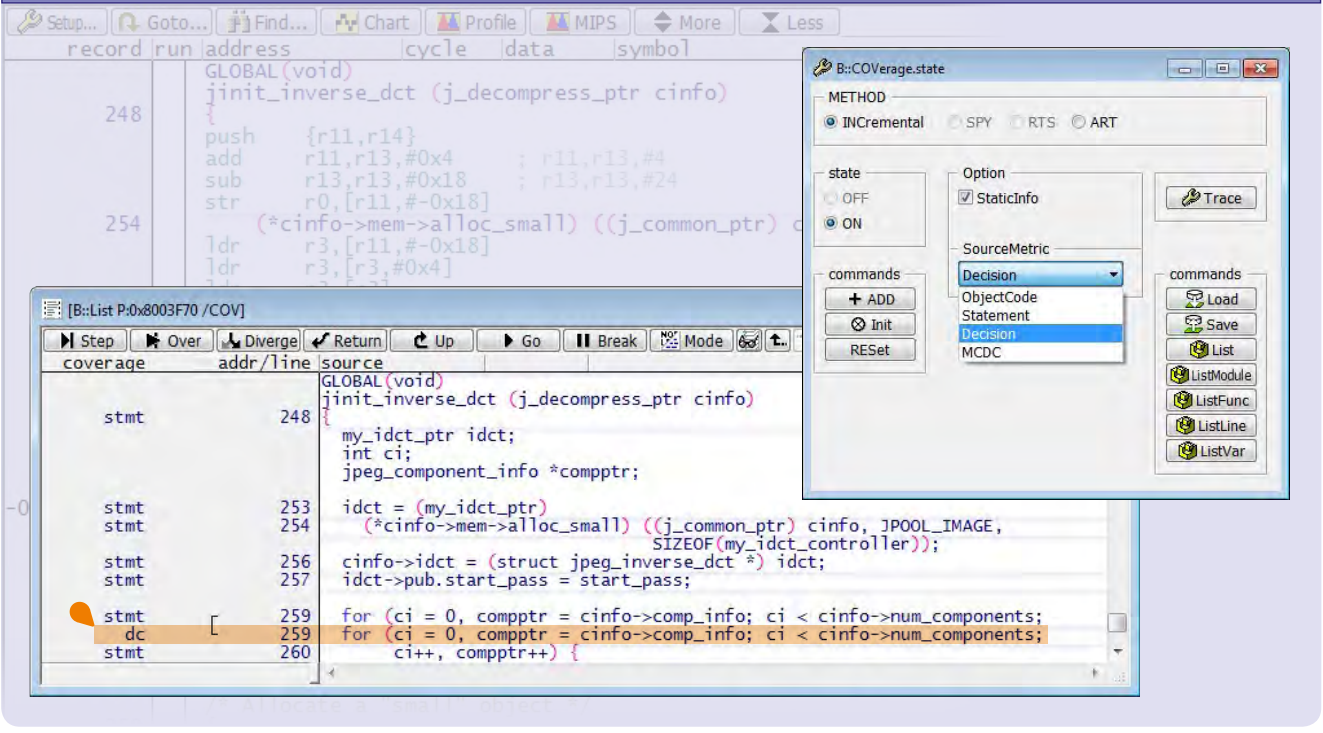
Bei der Entwicklung von sicherheitskritischen Systemen werden Code-Coverage-Verfahren eingesetzt, um nachzuweisen, dass die Software gründlich und umfassend getestet wurde. Normen für die Softwareentwicklung, wie ISO 26262 und DO-178C, haben dazu geführt, dass der Nachweis der Codeabdeckung heutzutage in vielen Projekten verbindlich gefordert wird.

Tracebasierte Coverage

Als Anbieter von Echtzeit-Tracetools kann Lauterbach hier sein ganzes Potential entfalten und tracebasierte Code-Coverage-Messungen anbieten, die komplett ohne Codeinstrumentierung auskommen. Bei der tracebasierten Coverage liegen die Informationen zur Programmausführung zunächst auf Objektcodeebene vor. Deshalb sind folgende Coverage-Metriken einfach nachweisbar:

- **Object Statement Coverage** weist nach, dass jede Assemblerinstruktion während des Tests mindestens einmal ausgeführt wurde.
- **Object Branch Coverage** weist nach, dass jeder bedingte Sprung mindestens einmal genommen (taken) sowie mindestens einmal nicht genommen (not taken) wurde.

Decision Coverage



Code-Coverage-Messungen auf Objektcodeebene gehören seit jeher zum Leistungsumfang aller TRACE32 Tracetools. Seit Februar 2017 unterstützt Lauterbach mit Statement- und Decision-Coverage nun auch den Nachweis auf Quellcodeebene (siehe Bild „Decision Coverage“ oben auf dieser Seite). Viele Kunden möchten ihre TRACE32 Tracetools nun auch dazu benutzen, die oft geforderten MCDC-Coverage-Messungen durchzuführen.

MCDC Code Coverage

Lange war man der Meinung, dass der Nachweis von *Object Branch Coverage* ein vollwertiger Ersatz für die Messung von MCDC-Coverage auf Quellcodeebene sei. In der Luftfahrtindustrie hat sich das Commercial Aviation Safety Team (CAST) aber dieser Auffassung klar entgegengestellt (siehe [1]).

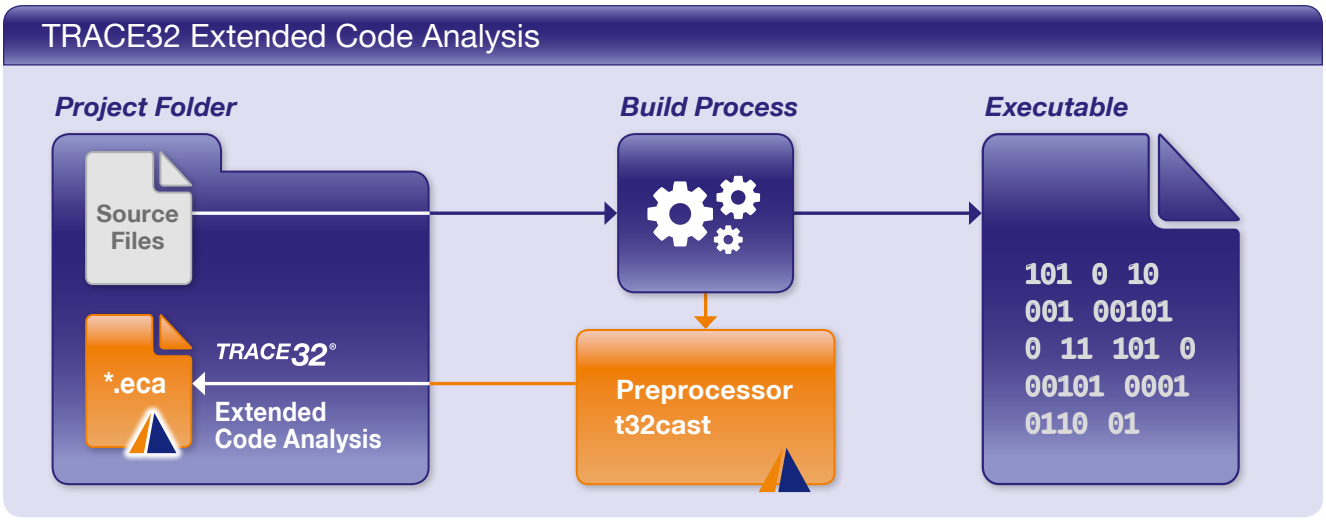
Aktuell wird deswegen bei der Messung von MCDC-Coverage fast ausschließlich mit Codeinstrumentierung gearbeitet. Lauterbach hat sich dagegen zum Ziel gesetzt, eine tracebasierte MCDC-Coverage anzubieten, die ohne Instrumentierung auskommt. Aus diesem Grund haben wir uns intensiv mit den Veröffentlichungen zu diesem Thema beschäftigt. Auch wenn die Veröffentlichungen unterschiedliche Lösungsansätze vorstellen, gibt es dennoch grundlegende Gemeinsamkeiten.

1. Damit MCDC-Coverage auf Basis der Tracedaten nachweisbar ist, müssen der Aufbau und die Position einer Entscheidung (Decision) im Quellcode bekannt sein.
2. Gleichzeitig muss jede Bedingung (Condition) im Quellcode auf Objektcodeebene durch einen bedingten Sprung bzw. eine bedingte Instruktion realisiert sein und nicht etwa durch eine arithmetische Repräsentation der Bedingung.
3. Bei der Durchführung der MCDC-Coverage-Analyse, müssen der Aufbau und die Position einer Entscheidung (Decision) im Objektcode bekannt sein.

Für die Implementierung einer MCDC-Coverage-Messung ohne Codeinstrumentierung werden gemäß Anforderung 1 also zusätzliche Informationen über die Quellcodestruktur benötigt, die aktuell nicht Teil der vom Compiler generierten Debuginformation sind. Außerdem ist sicherzustellen, dass der Compiler den Objektcode passend zu Anforderung 2 erstellt.

AdaCore

MCDC-Coverage kann mit einer Traceaufzeichnung



einfach nachgewiesen werden, wenn der Compiler und die MCDC-Coverage-Analyse durchführende Software vom selben Hersteller kommen. Der Compiler kann hier die neu benötigten Informationen über den Aufbau und die Position jeder Entscheidung im Quellcode in die Debuginformation integrieren. Eine solche Lösung bietet die Firma AdaCore (siehe [2]). AdaCore stellt zudem eine Target-Emulation zur Generierung der Tracedaten bereit.

Für Kunden, die auch für ihre finale Implementierung auf der Target-Hardware eine MCDC-Coverage nachweisen müssen, bietet AdaCore eine Schnittstelle, die es erlaubt, die mit TRACE32 aufgezeichneten Tracedaten für die Analyse zu importieren.

t32cast

Seit Januar 2018 können Lauterbach Kunden nun auch in TRACE32 eine MCDC-Coverage auf Basis einer Echtzeit-Traceaufzeichnung nachweisen. Lauterbach bietet dazu das Kommandozeilentool t32cast, das den C/C++ Quellcode analysiert. Als Ergebnis werden die für die Durchführung der MCDC-Coverage-Messung notwendigen Informationen über den Aufbau der Entscheidungen für jede Quellcodedatei erzeugt. Der Buildprozess ist also so anzupassen, dass diese Informationen generiert werden (siehe Bild „TRACE32 Extended Code Analysis“ oben auf dieser Seite). Das Kommandozeilentool t32cast ist vom Compiler unabhängig und einfach in vorhandene Buildumgebungen integrierbar.

Sobald der Anwender in TRACE32 die MCDC-Coverage-Messung startet, lädt TRACE32 die benötigten .eca-Dateien automatisch. Anschließend kann TRACE32 mit Hilfe der Debuginformation die Abbildung der Entscheidungen auf den Objektcode ermitteln.

Der Anwender muss bei diesem Vorgehen jedoch selbst sicherstellen, dass sein Compiler jede Bedingung im Quellcode durch einen bedingten Sprung bzw. eine bedingte Instruktion auf Objektcodeebene realisiert, z.B. durch Abschalten von Optimierungen.

Fazit

Die TRACE32 MCDC-Coverage kann unabhängig vom Compiler und der Prozessorarchitektur verwendet werden. Wer auch bei der Implementierung sicherheitskritischer Systeme nicht auf optimierten Code verzichten kann, kommt nicht darum herum, die Optimierung für die tracebasierte MCDC-Coverage zurückzuschrauben.

Grundsätzlich wäre es zu begrüßen, wenn sich die Compiler zukünftig so konfigurieren ließen, dass unabhängig von der gewählten Optimierungsstufe, Entscheidungen auf Objektcodeebene ausschließlich mit bedingten Sprüngen bzw. bedingten Instruktionen abgebildet werden und für alle übrigen Bereiche die ausgewählten Optimierungen durchgeführt werden.

Referenzen

[1] CAST-17 Position Paper (2003, January). Structural Coverage of Object Code
[2] Comar, C., Guitton, J., Hainque, O., & Quinot, T. (2012, May). Formalization and comparison of MCDC and object branch coverage criteria. In ERTS (Embedded Real Time Software and Systems Conference).
[3] RTCA Inc. (2011, December) RTCA/DO-178C Software Considerations in Airborne Systems and Equipment Certification

Code Coverage Live

Real-Time Processing of Trace Data

Real-time Profiling (RTS) nennt Lauterbach den Tracemode, bei dem die Tracedaten bereits zur Aufzeichnungszeit auf den Host übertragen und dort sofort analysiert werden. Damit ist es möglich, die Ergebnisse der Code-Coverage-Analyse live am Bildschirm mitzuverfolgen. Ende 2017 wurde der RTS-Mode, der bereits seit 2009 für Arm-ETMv3 genutzt werden kann, auf weitere Traceprotokolle ausgedehnt. Eine Übersicht über alle nun unterstützten Traceprotokolle zeigt die Tabelle „RTS - Supported Trace Protocols“.

Rahmenbedingungen

Für alle unterstützten Traceprotokolle gelten dabei die folgenden Rahmenbedingungen:

1. Zum Dekodieren der Tracedaten wird der Programmcode benötigt. Da das Auslesen des Codes aus dem Speicher zur Programmlaufzeit zu langsam wäre, muss dieser vor dem Start der Liveanalyse in TRACE32 geladen werden. Das bedeutet, eine Liveanalyse ist nur für statische Programme durchführbar.
2. Die Liveanalyse der Tracedaten funktioniert nur dann, wenn die durchschnittliche Datenrate am Traceport die maximale Übertragungsrate zum Hostrechner nicht überschreitet. Da alle aktuellen TRACE32 Tracetools über eine USB3-Schnittstelle verfügen, hat sich die max. Übertragungsrate zum Hostrechner auf 180 MByte/s erhöht. Dies ist eine Verdreifachung gegenüber den Datenraten aus dem Jahr 2009.
3. Die Liveanalyse der Tracedaten lässt sich zurzeit für Single-Core und SMP-Multicore Traceströme durchführen. Die Implementierung für AMP-Multicore Traceströme ist noch in der Entwicklung.
4. Die Live-Code-Coverage-Messung kann für die Metriken *Object Statement Coverage* und *Object Branch Coverage* sowie für Statement- und Decision-Coverage eingesetzt werden.

In der Regel werden die Tracedaten, nachdem sie analysiert wurden, nicht mehr benötigt. TRACE32 bietet jedoch die Option, die Tracedaten parallel zur Auswertung in einem sogenannten *Streaming File* abzuspeichern. Damit besteht die Möglichkeit, die Tracedaten auch nach der Code-Coverage-Messung noch einmal im Detail zu überprüfen. Wie bei einer klassischen TRACE32 Code-Coverage-Messung lassen sich umfassende Testreports erstellen. Dafür steht seit Ende 2017 ein neuer Dialog in TRACE32 mit vereinfachter Benutzerführung zur Verfügung.

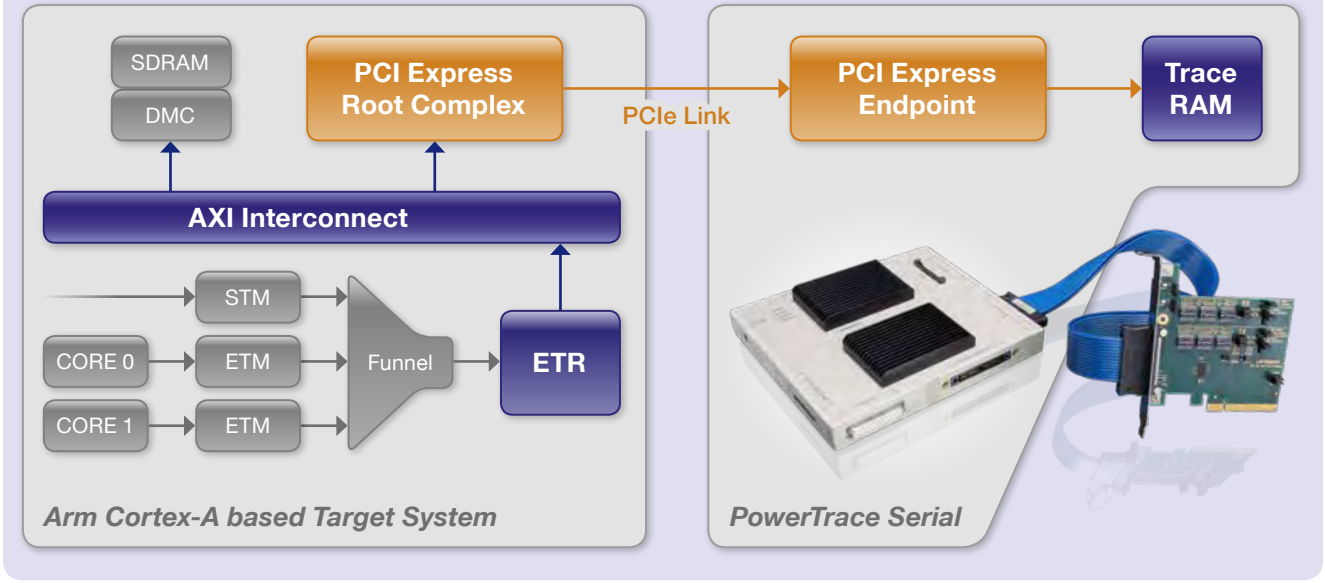
RTS – Supported Trace Protocols

- ETM v3 on Arm / Cortex®
- PTM on Arm / Cortex®
- ETM v4 on Arm / Cortex®
- MCDS on Infineon TriCore™
- Nexus on NXP MPC5xxx / STM SPC5xx
- Nexus on NXP PPC QorIQ®

others on request

Tracen über PCI Express

TRACE32 PowerTrace Serial as PCIe Endpoint



Verfügt ein Chip über eine PCIe-Schnittstelle, lässt sich diese nutzen, um Tracedaten mit einem externen Tracetool aufzuzeichnen und zu analysieren. Erste Lauterbach Kunden setzen diese Tracetechnik bereits für Arm Cortex-A® bzw. NXP Power Architecture-basierte QorIQ® Prozessoren ein.

Tracetool als PCIe Endpoint

Zunächst soll kurz skizziert werden, wie das Tracen über PCI Express funktioniert. Das Tracetool TRACE32 PowerTrace Serial ist so konzipiert, dass es als PCIe-Endpoint arbeiten kann. Als solcher muss das Tracetool bereits an das Target angeschlossen sein und laufen, bevor die Software (z.B. der Boot Loader) die Endpoints enumeriert und konfiguriert. Während der Konfiguration wird jedem Endpoint ein Adressbereich im Systemspeicher zugeordnet. Daten für diesen Endpoint können dann einfach dorthin geschrieben werden.

Nun muss die Traceinfrastruktur des Target-Systems so eingestellt werden, dass sie die Tracedaten auf den Adressbereich schreibt, der dem Endpoint TRACE32 PowerTrace Serial zugeordnet ist. Dies ist für die allermeisten Prozessoren machbar. Anschließend kann mit dem Tracen begonnen werden.

Zusammenfassung

Tracen über PCIe ist also einfach zu haben. Für Target-Systeme ohne dedizierte Traceschnittstelle bietet diese Technik eine sehr gute Möglichkeit, große Mengen von Tracedaten aufzuzeichnen.

Zum Abschluss noch einmal die wichtigsten Unterschiede zum Tracen über eine dedizierte Schnittstelle:

- Mit der Traceaufzeichnung kann erst begonnen werden, wenn die Target-Software den *PCIe Root Complex* konfiguriert hat. Dies fällt in den Aufgabenbereich des Betriebssystems.
- Gleichzeitig ist das Tracen über PCIe streng genommen nicht mehr „non-intrusive“. Es benötigt einen Bereich des Systemspeichers. Und der Trace konkurriert mit anderen Endpoints um die Bandbreite des PCIe-Busses.

Characteristics

Variable data rate
- Gen1 250 MByte/s per lane
- Gen2 500 MByte/s per lane
- Gen3 984 MByte/s per lane

Variable port width (1, 2, 4 or 8 lanes)

Full-size card adapter available,
mini-PCIe card adapter planned

4 GigaByte of trace memory,
trace decoding for all standard protocols

Nahtloser Umstieg von Wind River auf TRACE32

Da Wind River seit 2014 keine JTAG-Debugger mehr anbietet, steigen immer mehr Firmen für die Pflege und Weiterentwicklung ihrer Produkte auf TRACE32 um. In enger Zusammenarbeit mit Wind River hat Lauterbach seine TRACE32-Software systematisch so erweitert, dass Anwender wie gewohnt weiterarbeiten können, sobald ein paar grundlegende Anpassungen abgeschlossen sind.

Support für alle Prozessoren

Die meisten Umsteiger setzen Prozessoren aus der Power Architecture™ Familie ein. Da Lauterbach bereits seit 1997 die Unterstützung für diese Architektur in seinem Produktportfolio hat, können sich die Umsteiger auf bewährte Debug-Lösungen verlassen.

Skriptkonverter

Bevor man mit dem Debuggen loslegen kann, muss der Programmcode in das Flash programmiert werden. Dafür wird in TRACE32 ein Skript verwendet. Die Einstellung der Konfigurationsregister des Prozessors, insbesondere die Konfiguration des SDRAMs, nimmt innerhalb des Skripts einen wichtigen Platz ein. Bei Wind River wurden die notwendigen Einstellungen durch ein sogenanntes „Register Configuration File“ festgelegt. Für die Umstellung dieses Files auf ein TRACE32-Skript stellt Lauterbach einen speziellen Konverter zur Verfügung. Beim professionellen Einsatz eines Debuggers entstehen über die Jahre immer auch eine Vielzahl von Testskripten. Auch für die Portierung komplexer Testskripte bietet Lauterbach selbstverständlich einen Skriptkonverter an.

Integration der Wind River Workbench

Nach langjähriger, oftmals intensiver Nutzung eines Debuggers ist man mit dem Bedienkonzept und den Debug-Möglichkeiten so vertraut, dass ein notwendiger Umstieg auf ein anderes Tool großes Unbehagen auslöst. Aus diesem Grund hat Lauterbach seine Debugger bereits 2015 so erweitert, dass sie auch als TCF-Agent arbeiten können. Seitdem ist es möglich, die Wind River Workbench als Debug-IDE und den TRACE32 Debugger als Debug-Backend zu verwenden.

Support für alle Wind River Produkte

In den allermeisten Projekten werden auch Wind River Softwareprodukte eingesetzt, beispielsweise Wind River VxWorks, Wind River Linux oder der Wind River Hypervisor. Umfassende Debug-Möglichkeiten für diese Produkte waren beim Einsatz eines Wind River Debuggers natürlich eine Selbstverständlichkeit. Aber auch Lauterbach ist auf diesem Gebiet ein erfahrener Toolanbieter. TRACE32 unterstützt das Debugging von Wind River VxWorks bereits seit 1996. Ähnliches gilt seit dem Jahr 2000 auch für Wind River Linux. Inzwischen bietet TRACE32 sogar Support für Wind River Produkte, die die Wind River Workbench nicht mehr unterstützt. Dies umfasst VxWorks 7, VxWorks 653 v3 sowie Wind River Hypervisor v3 und folgende.

TRACE32 Debugger für RISC-V

JTAG Debugger

Seit November 2017 liefert Lauterbach seinen neuen RISC-V Debugger aus. Die ersten nun unterstützten Chips sind der E31 Core Complex (32-Bit) und der E51 (64-Bit) Core Complex von SiFive.

RISC-V ist eine *Open Instruction Set Architecture* (ISA), die auf den etablierten RISC-Prinzipien basiert und unter der Führung der RISC-V Foundation (https://riscv.org) organisiert, spezifiziert und weiterentwickelt wird. Zunächst für die akademische Forschung entwickelt, ist RISC-V dabei, sich auf dem Embedded-Markt zu etablieren. Aus diesem Grund ist ein professioneller Hardware-Debugger unverzichtbar.

Die Lauterbach-Implementierung für den RISC-V Debugger basiert auf der Open-Source-Spezifikation „RISC-V External Debug Support“, welche die RISC-V Foundation voraussichtlich im Jahr 2018 endgültig verabschieden wird. Zielsetzung der Spezifikation ist ein flexibles Halt-Mode-Debugging. Jeder Hardware-Thread eines RISC-V Cores soll ab dem Reset zu debuggen sein.

Damit RISC-V Prozessoren mit TRACE32 zu debuggen sind, müssen sie aktuell über das JTAG DTM (*Debug Transport Module*) verfügen. Da das DTM als ein unabhängiges und somit austauschbares Modul spezifiziert ist, besteht für Chiphersteller die Möglichkeit, den Zugriff auf das sogenannte *Debug Module* über eine andere Kommunikationsschnittstelle zu realisieren. Die große Erfahrung mit den verschiedensten Debug-Kommunikationsschnittstellen macht Lauterbach zum kompetenten Partner bei der Umsetzung.

Über das *Debug Module* hat der TRACE32 Debugger Zugriff auf alle Standard-Debugfunktionen des Prozessors. Diese sind als *Debug Register* und sogenannte *Abstract Commands* Teil der Spezifikation. Darüber hinaus ermöglicht es die Spezifikation auch proprietäre Debug-Funktionen zu designen.

Der RISC-V Debugger unterstützt bereits verschiedene Standard-ISA-Erweiterungen, wie etwa komprimierte Instruktionen oder Floating-Point-Instruktionen sowie kundenspezifische ISA-Erweiterungen.