NEWS 2018

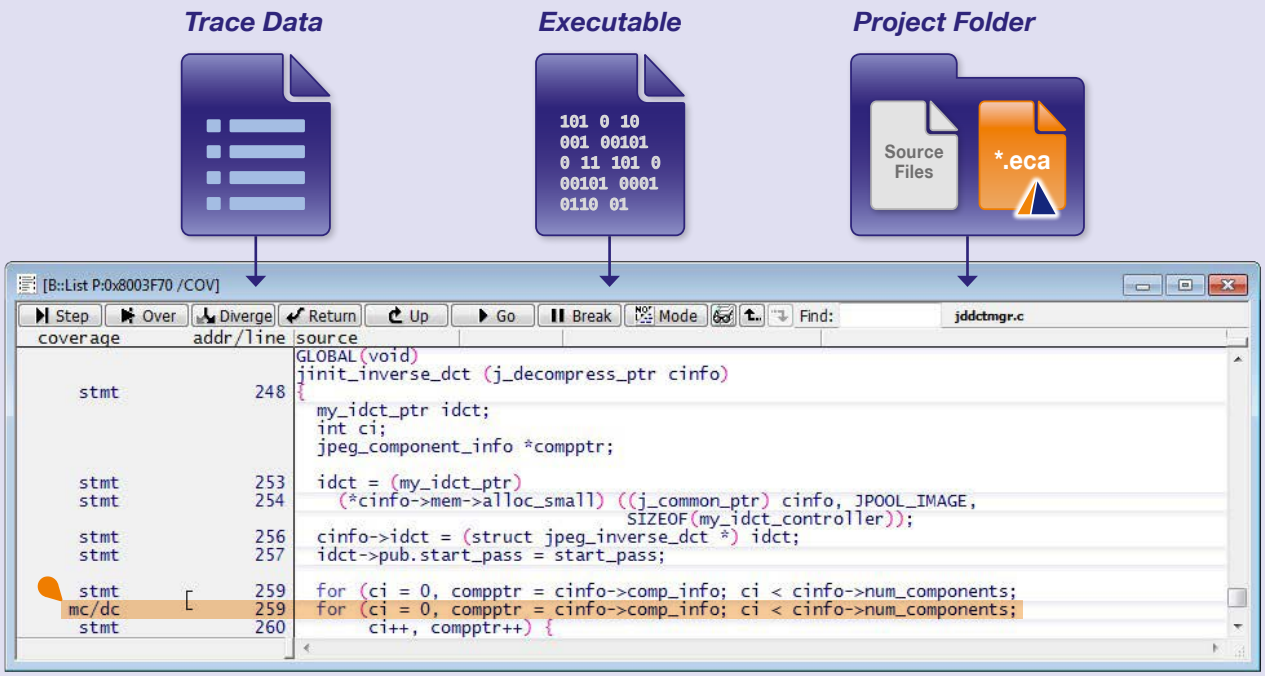
Edizione italiana

INDICE

LAUTERBACH
DEVELOPMENT TOOLS

Copertura di codice MCDC basata sul trace

MCDC Coverage



A marzo 2018 Lauterbach rivelerà la sua soluzione per l'analisi di copertura MCDC del codice basata sul trace. Già ora TRACE32 supporta a livello di codice sorgente tutte le metriche più importanti di copertura del codice.

Definizioni secondo DO-178C [3]

Statement Coverage: ciascuna istruzione del programma è stata eseguita almeno una volta.

Decision Coverage: ciascun punto di entrata e uscita del programma è stato eseguito almeno una volta e ciascuna decisione del programma ha raggiunto tutti i possibili risultati almeno una volta.

Modified Condition/Decision Coverage: ciascun punto di entrata e uscita del programma è stato eseguito almeno una volta, ciascuna condizione di una decisione del programma ha raggiunto tutti i possibili risultati almeno una volta, ciascuna decisione del programma ha raggiunto tutti i possibili risultati almeno una volta, e si è mostrato che ciascuna condizione di una decisione influenza in modo indipendente il risultato di quella decisione. Si mostra che una condizione influenza in modo indipendente il risultato di una decisione: (1) variando solo quella condizione e mantenendo fisse tutte le altre possibili condizioni oppure (2) variando solo quella condizione e mantenendo fisse tutte le altre possibili condizioni che potrebbero influenzare il risultato.

Nello sviluppo di sistemi safety-critical l'analisi di copertura del codice viene svolta per verificare che il software sia stato testato in modo completo e accurato. Gli standard di sviluppo software, come per esempio ISO 26262 e DO-178C, concordano nell'affermare che la verifica di copertura del codice è una parte indispensabile del ciclo di sviluppo.

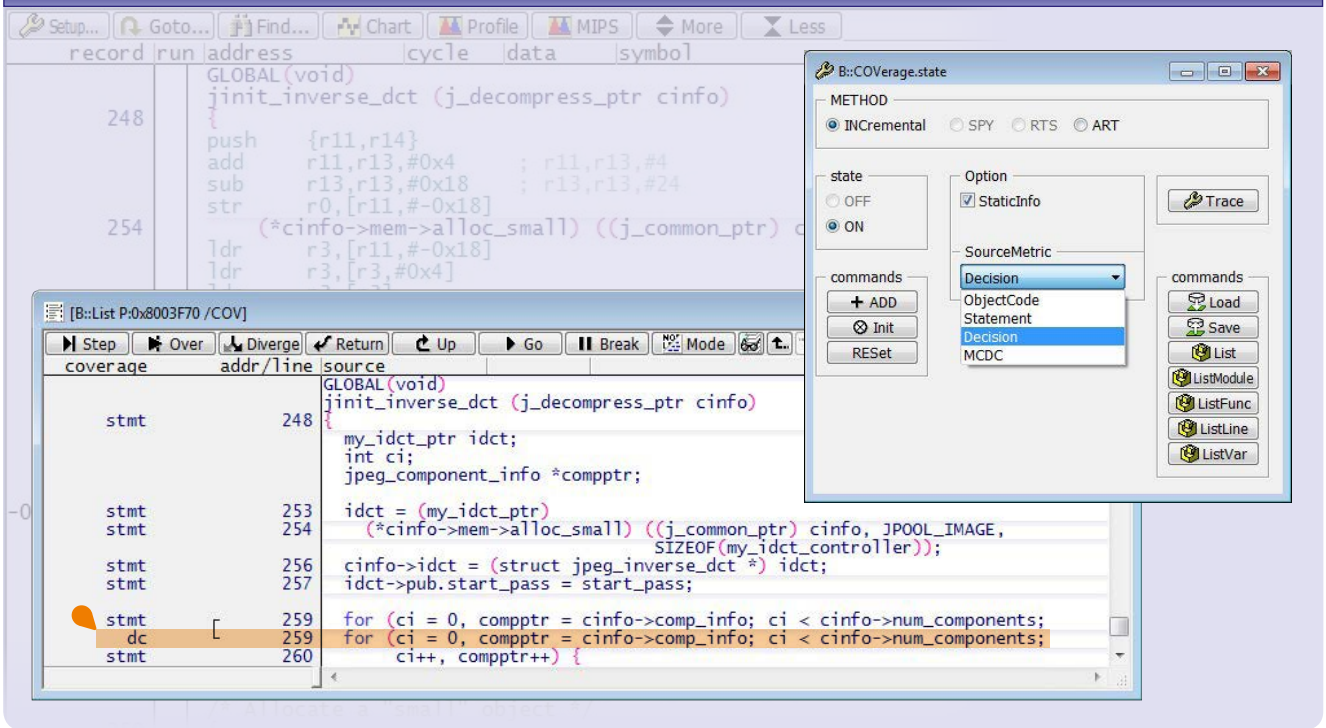
Copertura di codice basata sul trace

In quanto fornitore di sistemi di trace in tempo reale, Lauterbach offre misure di copertura del codice basate sul trace che non richiedono l'instrumentazione del codice da testare. Le informazioni sul flusso di esecuzione del programma sono tracciate direttamente a livello di codice oggetto. Questo permette di verificare facilmente le seguenti metriche di copertura:

- **Object statement coverage** – verifica che ogni istruzione assembler è stata eseguita almeno una volta durante il test.
- **Object branch coverage** – verifica che ogni salto condizionato è stato eseguito almeno una volta ed è stato "non eseguito" almeno una volta.

Le misure di copertura di codice a livello di codice oggetto sono da sempre tra le funzionalità fornite da

Decision Coverage



tutti i sistemi di trace TRACE32. Con l'aggiunta dello "statement coverage" e del "decision coverage" a febbraio 2017, i sistemi Lauterbach permettono anche la verifica della copertura a livello di codice sorgente (vedere la figura "Decision Coverage" in alto in questa pagina). Ora molti clienti vogliono anche usare i loro sistemi di trace TRACE32 per effettuare le sempre più richieste misure di copertura MCDC.

Copertura di codice MCDC

Per molto tempo si è pensato che la verifica di "object branch coverage" fosse un sostituto adeguato per la misura di copertura MCDC a livello di codice sorgente. Tuttavia nell'industria aeronautica il CAST (Commercial Aviation Safety Team) ha sostenuto una chiara opposizione su questo punto (vedere [1]).

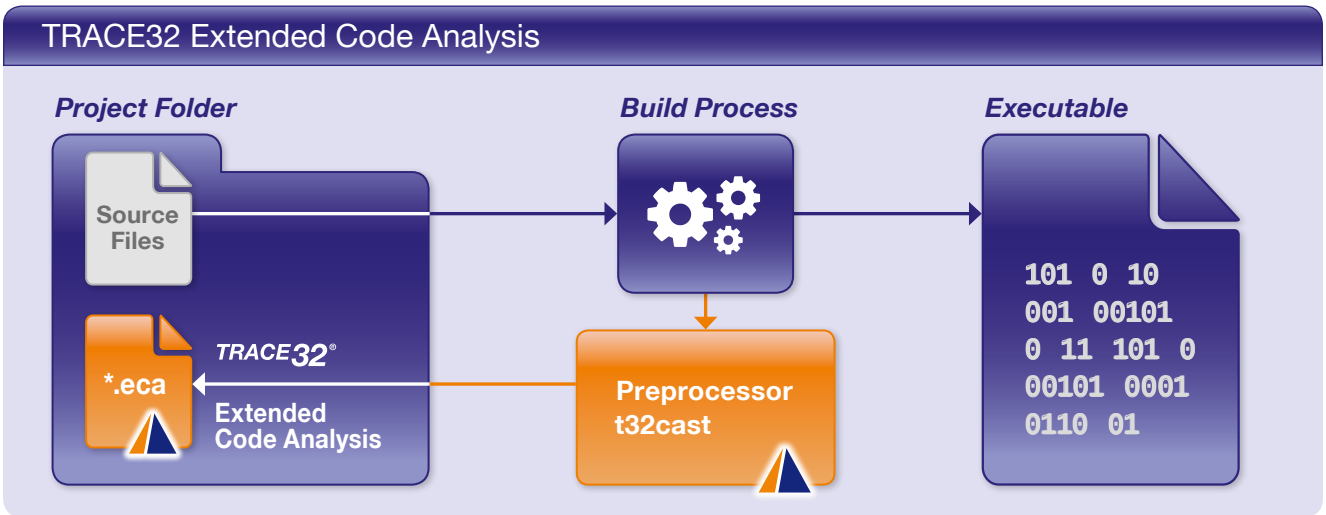
Attualmente quasi tutti si basano sull'instrumentazione del codice sorgente per riuscire a ottenere dati di copertura MCDC. Gli ingegneri in Lauterbach hanno invece raggiunto l'obiettivo di fornire una copertura MCDC senza il bisogno di alterare in alcun modo il codice da testare. A questo riguardo sono stati analizzati molti documenti divulgativi e varie pubblicazioni. Ognuno considera vincente il proprio approccio ma tutti hanno in comune fra loro alcuni punti fondamentali:

1. Per garantire che la copertura MCDC possa essere verificata partendo dai dati di trace, devono essere note sia la struttura sia la posizione di una decisione all'interno del codice sorgente.
2. Allo stesso tempo ogni condizione nel codice sorgente deve essere rappresentata a livello di codice oggetto da un salto condizionato o da una istruzione condizionale, non da una rappresentazione aritmetica della condizione.
3. Quando si esegue l'analisi di copertura MCDC, devono essere note la struttura e la posizione di una decisione all'interno del codice oggetto.

Per poter implementare una misura di copertura MCDC senza instrumentazione di codice, in accordo con il requisito #1 sono necessarie informazioni aggiuntive sulla struttura del codice sorgente. Al momento ciò non fa parte delle informazioni di debug generate dal compilatore. Inoltre è necessario che il compilatore produca codice oggetto in modo tale da soddisfare il requisito #2.

AdaCore

La copertura MCDC può essere ottenuta facilmente



da un trace registrato se il compilatore e il software che esegue l'analisi di copertura MCDC provengono dallo stesso fornitore. Il compilatore può includere nelle informazioni di debug i nuovi contenuti informativi richiesti sulla struttura e posizione di ogni decisione nel codice sorgente. L'azienda AdaCore (vedere [2]) offre tale soluzione. Inoltre AdaCore offre anche una soluzione di emulazione del target per generare i dati di trace.

Per i clienti che devono anche verificare la copertura MCDC nell'implementazione finale su un target hardware, AdaCore espone un'interfaccia che permette di importare i dati di trace registrati con TRACE32 per l'analisi richiesta.

t32cast

Da marzo 2018 i clienti di Lauterbach possono anche verificare la copertura MCDC in TRACE32 partendo da una registrazione del trace in tempo reale. A tal scopo Lauterbach ha realizzato il tool a linea di comando "t32cast" per l'analisi del codice sorgente C/C++. Per ogni file di codice sorgente viene così generata l'informazione sulla struttura delle decisioni, necessaria per poter eseguire le misure di copertura MCDC. Ne consegue che il processo di generazione del codice deve essere riorganizzato per permettere che tale informazione venga prodotta (vedere la figura "TRACE32 Extended Code Analysis" qui sopra). Il tool a linea di comando t32cast è indipendente dal compilatore usato e può essere facilmente integrato negli ambienti di compilazione già esistenti.

Non appena l'utente avvia le misure di copertura MCDC in TRACE32, TRACE32 carica automaticamente tutti i file con estensione .eca richiesti, prodotti dal tool t32cast. Grazie a questi file TRACE32 è in grado di mappare sul codice oggetto le decisioni a livello di codice sorgente, con l'aiuto delle informazioni di debug.

In ogni caso, durante questo processo, l'utente deve assicurare che il compilatore utilizzato rappresenti ogni condizione nel codice sorgente con un salto condizionato o con una istruzione condizionale a livello di codice oggetto, ad esempio disabilitando le ottimizzazioni.

Conclusioni

La copertura MCDC in TRACE32 può essere usata indipendentemente dal compilatore e dall'architettura del processore. Per coloro che non possono prescindere dalle ottimizzazioni del codice, anche quando si implementano sistemi safety-critical, non c'è altra scelta che ridurre l'ottimizzazione quando si esegue la copertura MCDC basata sul trace.

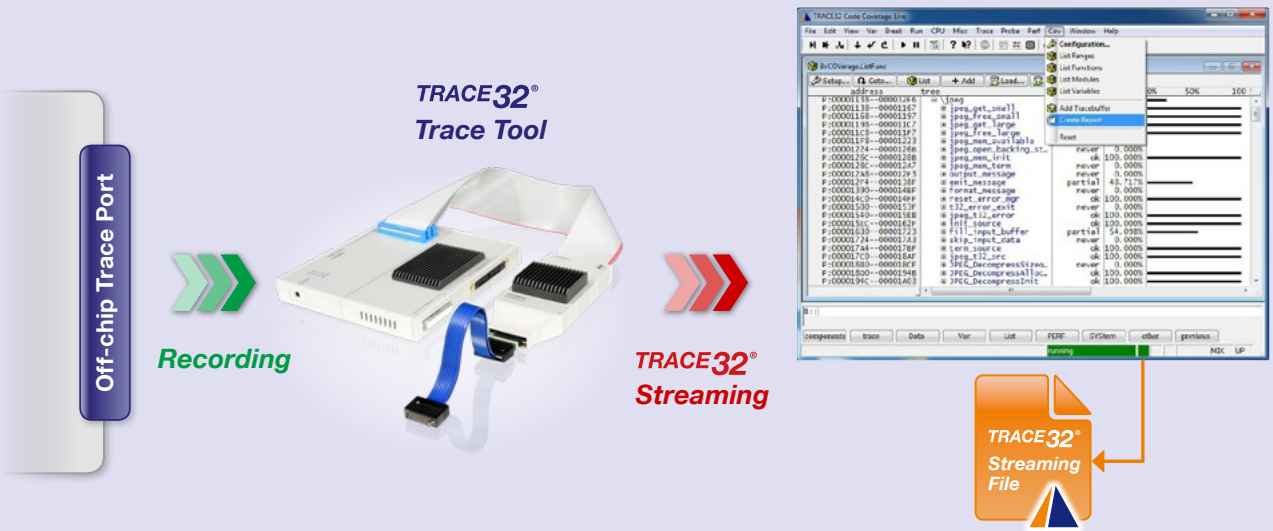
In linea di principio sarebbe molto utile se in futuro i compilatori potessero essere configurati in modo tale da tradurre le decisioni, a livello di codice oggetto, solamente in salti condizionati o istruzioni condizionali, indipendentemente dalle ottimizzazioni impostate, e che tali ottimizzazioni venissero interamente eseguite in tutte le parti rimanenti.

Riferimenti

[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

Copertura di codice in tempo reale

Real-Time Processing of Trace Data



Real-time Profiling (RTS) è il nome che Lauterbach ha dato alla modalità di trace in cui i dati di trace sono trasferiti all'host e analizzati immediatamente. In questo modo è possibile seguire in tempo reale sullo schermo i risultati dell'analisi di copertura di codice. Alla fine del 2017 il metodo RTS, già disponibile per ARM ETMv3 fin dal 2009, è stato portato anche su altri protocolli di trace. La tabella "RTS – Protocolli di Trace supportati" descrive brevemente tutti i protocolli di trace attualmente supportati.

Prerequisiti

I seguenti prerequisiti si applicano a tutti i protocolli di trace supportati:

1. Per decodificare i dati di trace è necessario il codice del programma. Considerando che sarebbe troppo lento leggere il codice dalla memoria mentre il programma è in esecuzione, il codice deve essere caricato in TRACE32 prima di iniziare l'analisi in tempo reale. Ciò significa che un'analisi in tempo reale può essere svolta solo per programmi statici.

2. L'analisi in tempo reale dei dati di trace funziona solo se la velocità media dei dati sulla porta trace non supera la massima velocità di trasmissione dei dati verso host computer. Considerando che i sistemi TRACE32 attuali sono equipaggiati con un'interfaccia USB3, la massima velocità di trasmissione dei dati verso host computer è di circa 180 MByte/s. La velocità è triplicata rispetto a quella che era disponibile nel 2009.

3. Ad oggi l'analisi in tempo reale dei dati di trace può essere eseguita su flussi trace provenienti da un singolo core o in configurazioni multicore SMP. L'implementazione per il caso multicore AMP è attualmente ancora in fase di sviluppo.

4. Le misure di copertura di codice in tempo reale possono essere usate per le metriche Object Statement Coverage e Object Branch Coverage, ma anche per Statement Coverage e Decision Coverage.

Di solito i dati di trace non servono più dopo essere stati analizzati. TRACE32 permette comunque di salvarli mentre l'analisi è in corso, in un cosiddetto file di streaming. In questo modo i dati di trace possono essere nuovamente verificati, in modo accurato, anche al termine delle misure di copertura di codice. Come per le misure di copertura di codice tradizionali offerte da TRACE32, anche il metodo RTS consente di generare dei report di test complessivi.

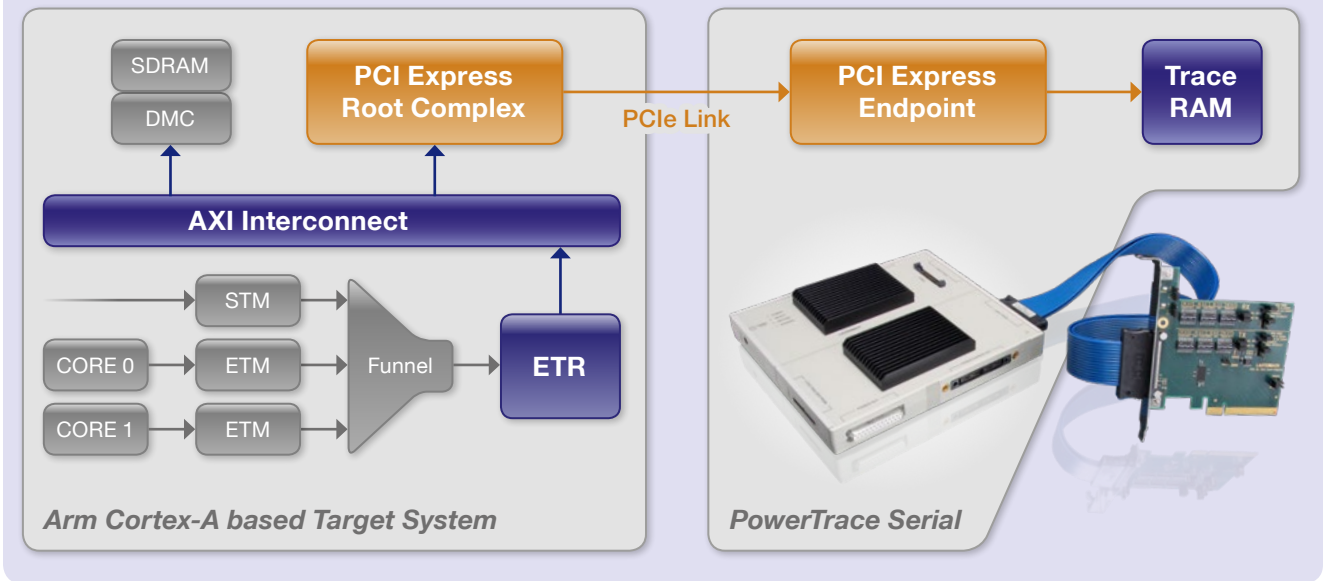
RTS – Protocolli di Trace supportati

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

altri su richiesta

Tracing via PCI Express

TRACE32 PowerTrace Serial as PCIe Endpoint



Nei chip equipaggiati con un'interfaccia PCIe, attraverso tale interfaccia è possibile registrare e analizzare dati di trace con un sistema di trace esterno. Alcuni clienti Lauterbach hanno già provato questa tecnica di trace su processori ARM Cortex-A® o QorIQ® di NXP basati su Power Architecture.

Il tool di trace come endpoint PCIe

Anzitutto occorre chiarire come funziona il tracing via PCI Express. Il sistema di trace, TRACE32 PowerTrace Seriale, è stato progettato in modo da poter operare come endpoint PCIe. Per questo motivo il sistema di trace deve essere collegato al target e già avviato prima che il software (per esempio il bootloader) inizi l'enumerazione e la configurazione degli endpoint.

Caratteristiche

Data rate variabile

Gen1 250 Mbyte/s per lane
Gen2 500 Mbyte/s per lane
Gen3 984 Mbyte/s per lane

Larghezza di porta variabile (1, 2, 4 o 8 lanes)

Scheda di adattamento full-size disponibile, scheda di adattamento mini-PCIe pianificata

4 GigaByte di memoria trace, decodifica trace per tutti i protocolli standard

Nel corso di questa configurazione, per ogni endpoint viene allocato un intervallo di memoria di sistema. Da questo momento in poi, basta semplicemente scrivere a queste locazioni i dati relativi all'endpoint.

L'infrastruttura di trace del sistema target deve essere configurata per scrivere i dati di trace nell'intervallo di memoria allocato all'endpoint di TRACE32 PowerTrace Seriale. Ciò si può fare sulla maggior parte dei processori. A questo punto il trace può iniziare.

Conclusioni

Tracciare via PCIe è facile. Per i sistemi target privi di un'interfaccia di trace dedicata, questa tecnica costituisce un metodo eccellente per registrare grandi quantità di dati di trace. Le principali differenze fra il tracciamento tramite un'interfaccia esterna come PCIe, e una porta di trace dedicata, possono essere così riassunte:

- La registrazione del trace può iniziare solo dopo che il software del target ha configurato il Root Complex PCIe. Questo è un compito assegnato al sistema operativo.
- Allo stesso tempo, strettamente parlando il tracing via PCIe non è più "non intrusivo". Richiede infatti una porzione della memoria di sistema e, inoltre, il trace deve competere con altri endpoint per l'utilizzo del bus PCI.

Passare senza problemi da Wind River a TRACE32

Fin dal 2014 Wind River ha smesso di offrire i suoi debugger JTAG e gli utenti interessati stanno migrando su TRACE32 per la manutenzione e ulteriori sviluppi dei propri prodotti. In stretta collaborazione con Wind River, Lauterbach ha sistematicamente migliorato il suo SW TRACE32 e ogni necessaria modifica viene subito implementata così che gli utenti possano continuare a lavorare come hanno sempre fatto.

Supporto per tutti i processori

Molti clienti che stanno passando a TRACE32 utilizzano processori della famiglia Power Architecture™. Il supporto per questa architettura esiste nel portafoglio prodotti Lauterbach sin dal 1997, e quindi i clienti che migrano su TRACE32 possono avvalersi di soluzioni di debug ben collaudate.

Un converter per gli script

Prima di poter iniziare il debug, solitamente il codice eseguibile viene programmato nella flash del target. TRACE32 lo fa con uno script. Il setup dei registri di configurazione del processore, in particolare per l'SDRAM, è una componente molto importante dello script. Wind River ha definito il setup necessario in un file chiamato "Register Configuration File". Lauterbach fornisce uno speciale convertitore per tradurre questo file in uno script TRACE32. In molti anni di lavoro con debugger professionali, i clienti hanno spesso accumulato un gran numero di script di test per scopi diversi. Lauterbach offre un convertitore di script che permette di trasferire in TRACE32 questi script di test così complessi.

Wind River Workbench

Dopo aver usato per molti anni uno stesso debugger ben testato e collaudato, per molti progettisti passare a un nuovo debugger può costituire motivo di preoccupazione. Per questo Lauterbach nel 2015 ha migliorato i suoi debugger per operare come "TCF agent". Ora è possibile usare Wind River Workbench come IDE di debug e il debugger TRACE32 come backend di debug.

Supporto per tutti i prodotti Wind River

In molti progetti vengono anche usati prodotti software di Wind River, per esempio: Wind River VxWorks, Wind River Linux o Wind River Hypervisor.
Soluzioni complete di debug per questi prodotti erano ovviamente disponibili quando si usava un debugger Wind River. Lauterbach è un fornitore di lunga esperienza in quest'area e TRACE32 supportava già dal 1996 il debug di Wind River VxWorks. Nel 2000 è stato aggiunto il supporto completo per Wind River Linux. Attualmente TRACE32 offre anche supporto per prodotti Wind River che non sono più supportati neanche da Wind River Workbench, in particolare: VxWorks 7, VxWorks 653 v3 e Wind River Hypervisor v3.

Debugger TRACE32 per RISC-V

JTAG Debugger

A novembre 2017 Lauterbach ha lanciato il suo nuovo debugger RISC-V. I primi chip attualmente supportati sono i Core Complex E31 (32-bit) ed E51 (64-bit) di SiFive.

RISC-V è un'architettura aperta di set di istruzioni (ISA, Instruction Set Architecture) basata sui ben collaudati principi RISC; organizzata, specificata e infine sviluppata sotto la direzione del RISC-V Foundation (https://riscv.org). Definita inizialmente per ricerche in ambito accademico, ora RISC-V si sta sempre più affermando nel mercato embedded, rendendo indispensabile la presenza di un debugger hardware professionale.

L'implementazione Lauterbach del debugger RISC-V si basa sulla specifica open source "RISC-V External Debug Support", che si ritiene verrà adottata nel 2018 da RISC-V Foundation. Obiettivo di questa specifica è definire una modalità di debug di tipo "halt mode" flessibile. Ogni thread hardware di un core RISC-V potrà essere debuggato direttamente dal reset.

Per poter effettuare il debug di processori RISC-V con TRACE32, attualmente i processori devono essere equipaggiati con un JTAG DTM (Debug Transport Module). Il DTM è un modulo indipendente e rimpiazzabile, lasciando così liberi i produttori di chip di implementare l'accesso al cosiddetto "Debug Module" attraverso una diversa interfaccia di comunicazione. Grazie alla sua ampia esperienza con le diverse interfacce di comunicazione di debug, Lauterbach può svolgere un ruolo di collaborazione competente nell'implementazione.

Tramite il Debug Module, il debugger TRACE32 può accedere a tutte le funzioni di debug standard del processore. Queste funzioni sono definite nella specifica, come i registri di debug e i cosiddetti Comandi Astratti. Inoltre la specifica permette anche di progettare funzioni di debug proprietarie. Il debugger RISC-V supporta già diverse estensioni ISA standard come, per esempio, le istruzioni compresse o quelle per virgola mobile, ma supporta anche estensioni ISA specifiche del cliente.

I.P.