NEWS 2018

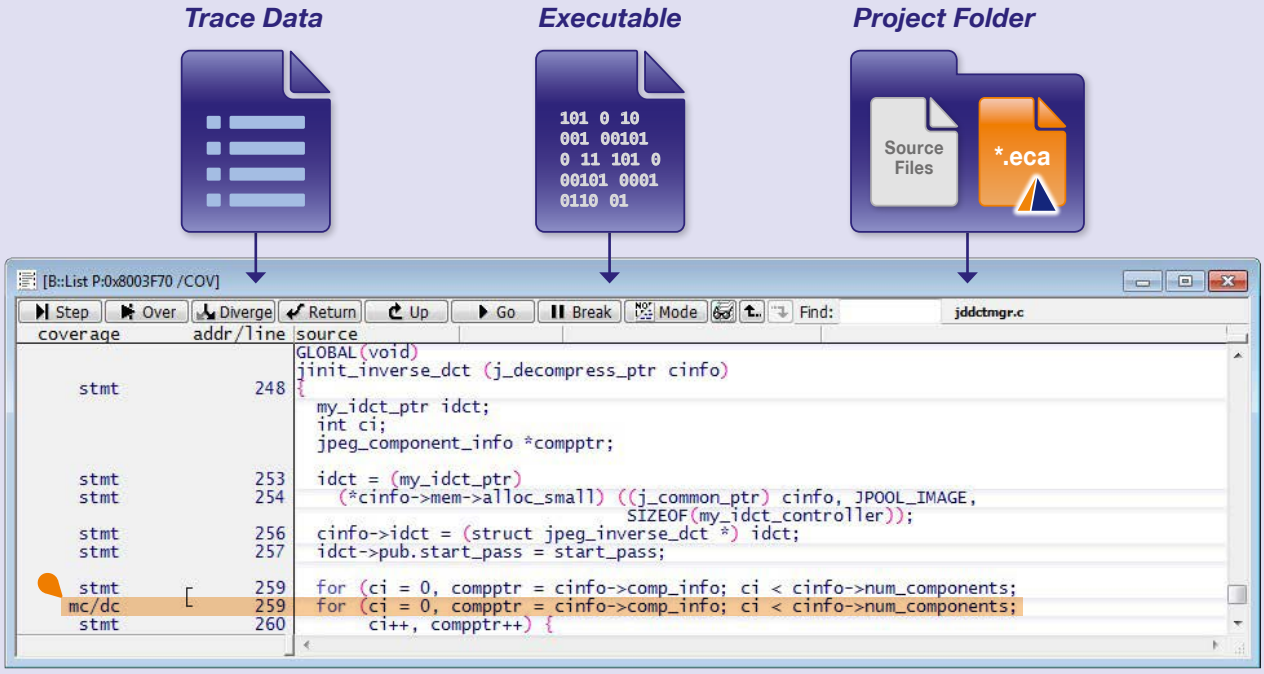
Edition Française

Sommaire

LAUTERBACH
DEVELOPMENT TOOLS

Couverture MCDC basée sur la Trace

Couverture MCDC



En Mars 2018, Lauterbach dévoilera sa propre couverture de code MCDC basée sur la trace. TRACE32 supporte désormais toutes les méthodes importantes de couverture de code au niveau source.

Définitions selon DO-178C [3]

Couverture des instructions : Chaque instruction du programme à été exécutée au moins une fois.

Couverture décisionnel : chaque point d'entrée et de sortie a été exécuté au moins une fois et chaque décision essaie toutes les issues possibles.

couverture condition / décision modifiée : Chaque point d'entrée et de sortie du programme a été exécuté au moins une fois, chaque condition dans la décision a utilisé toutes les sorties possibles au moins une fois, et toutes les conditions dans une décision ont affecté indépendamment la décision. Une condition affecte indépendamment la décision si : (1) modifier cette condition en maintenant fixe toutes les autres, ou (2) modifier cette condition en maintenant fixe toutes les autres conditions qui pourraient modifier la sortie.

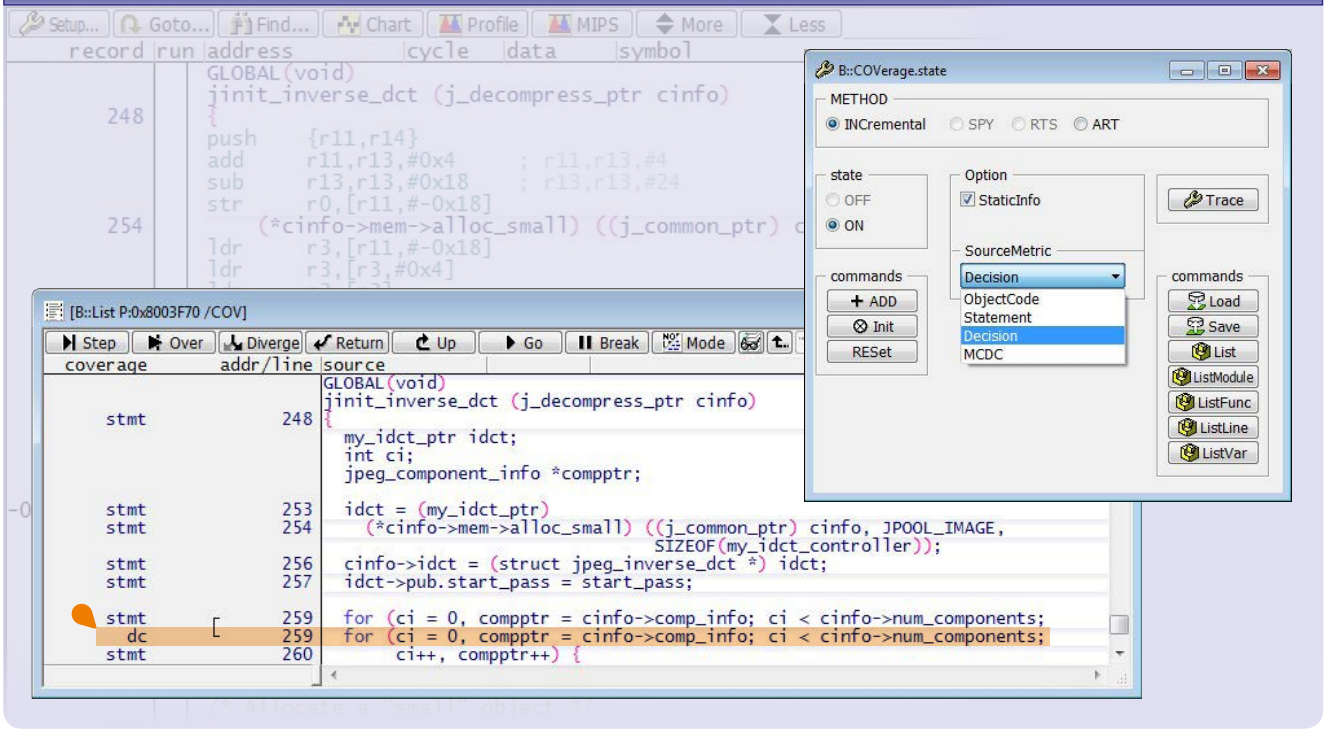
Pendant le développement de systèmes à sécurité critique, le processus de couverture de code est utilisé pour démontrer que le logiciel a été testé intégralement et de manière compréhensive. Les standards de développements logiciels comme l'ISO 26262 et la DO-178C stipulent que le test de couverture de code est obligatoire dans le cycle de développement.

Couverture de code basée sur la Trace

En tant que fournisseur d'outils temps réel, Lauterbach propose de la couverture de code basée sur la trace instruction qui ne nécessite absolument aucune instrumentation du code source. Les informations de l'exécution du programme sont d'abord tracées au niveau code objet. Cela permet de démontrer facilement les mesures de couverture suivantes :

- La **couverture objet instruction** (Statement Coverage) démontre que chaque instruction assembleur a été exécutée au moins une fois durant le test.
- La **couverture objet des branchements** (Branch Coverage) démontre que chaque saut conditionnel a été, ou bien n'a pas été pris au moins une fois.

Couverture Décisionnelle



Enregistrer la couverture de code au niveau objet a toujours fait partie des fonctionnalités TRACE32 sur tous les outils de trace. Avec désormais l'ajout de la couverture conditionnelle et décisionnelle en février 2017, les outils Lauterbach permettent de prouver la couverture au niveau code source. (Voir image « Couverture décisionnelle » tout en haut de cette page)

Couverture de Code MCDC

Les développeurs ont souvent pensé que les tests de couverture de code au niveau branchement pouvaient remplacer une mesure de couverture de code MCDC au niveau source. Cependant, l'industrie avionique de la Commercial Aviation Safety Team (CAST) a montré son opposition très claire (se référer à [1]).

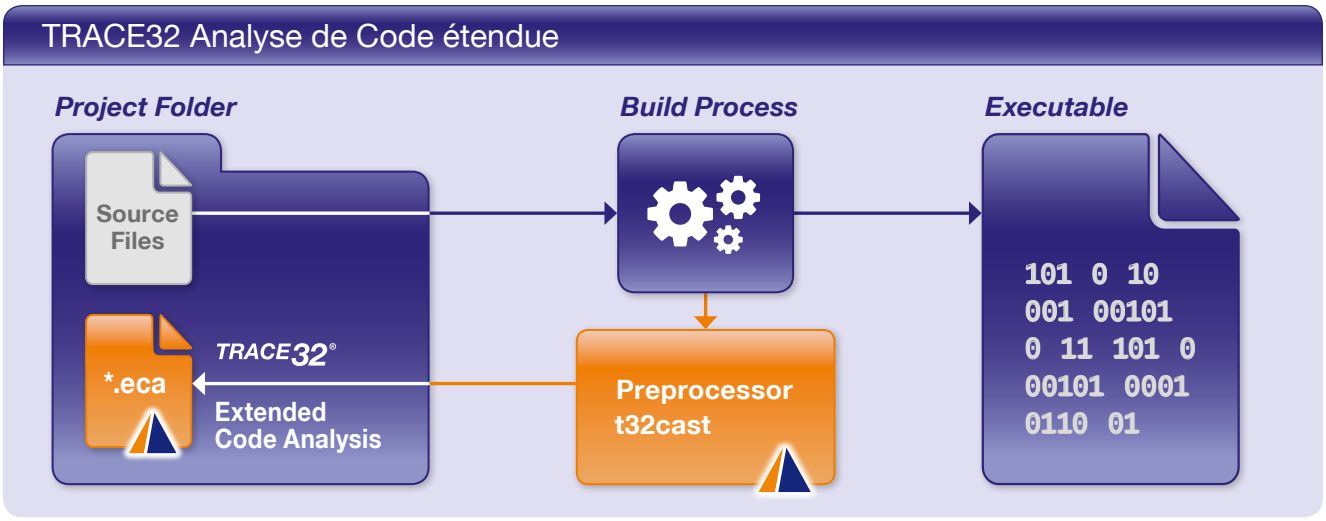
Aujourd'hui, il est courant d'être obligé d'instrumenter son code source dans le but de pouvoir générer les données de couverture de code MCDC. Les ingénieurs Lauterbach ont réussi à permettre la génération d'une couverture de code MCDC sans avoir besoin de modifier une seule ligne de code source, suite à de nombreuses recherches dans diverses publications et autres white Paper sur le sujet. Chaque expert a son approche dédiée, cependant à travers chacun on note de fondamentales similitudes :

1. Dans le but d'assurer que la couverture de code MCDC peut être validée grâce à la trace instruction, la structure et la position du décisionnel dans le code source doivent être connues.
2. Dans le même temps, chaque condition dans le code source doit être représentée au niveau code objet par un saut conditionnel ou une instruction conditionnelle et non par une représentation arithmétique de la condition.
3. Lorsque l'analyse MCDC de la couverture de code est générée, la structure et la position de la décision dans le code objet doivent être connues.

Pour implémenter une mesure de couverture de code MCDC sans instrumenter le code, tout en respectant les exigences #1, des informations supplémentaires sont nécessaires sur la structure du code source. Actuellement cela ne fait pas partie des informations de debugge standard générées par le compilateur. De plus, il faut s'assurer aussi que le compilateur génère un code objet respectant les exigences #2.

AdaCore

La couverture de code MCDC peut être validée assez facilement, en utilisant une trace instruction.



C'est possible, si le compilateur et l'application sur laquelle l'analyse de couverture MCDC est effectuée proviennent du même prestataire. Le compilateur peut ainsi inclure les nouvelles informations nécessaires sur la structure et la position de chaque décision du code source dans les informations de debugge. La société AdaCore (voir [2]) offre ce type de solution. En plus de cela, Adacore offre aussi une solution d'émulation d'une cible générant les données de trace.

Pour les clients ayant besoin de valider leur couverture MCDC sur l'implantation finale de leur cible, AdaCore offre une interface qui permet d'importer les données de trace, enregistrées par TRACE32, pour l'analyse demandée.

t32cast

Depuis mars 2018, les clients Lauterbach peuvent aussi valider leur couverture de code MCDC dans TRACE32, basée sur l'enregistrement d'une trace temps réel. Pour ce faire, Lauterbach propose un outil de commande en ligne « t32cast » qui analyse le code source C/C++. Il génère comme résultat, toutes les informations sur la structure de décision nécessaires à la couverture de code MCDC et cela pour chaque fichier source. Par conséquent, le process de build doit être ajusté pour intégrer la génération de ces informations (se référer à l'image en haut de la page « TRACE32 Analyse de Code étendue »). L'outil de commande en ligne « t32cast » est indépendant du compilateur et peut donc être inséré dans un process de build très facilement.

Dès que l'utilisateur démarrera une mesure MCDC, TRACE32 va automatiquement charger tous les fichiers *.eca générés par l'outil t32cast. Ensuite, TRACE32 sera capable de mapper les décisions sur le code source via le code objet grâce aux informations de debugge.

Cependant, pendant ce processus, l'utilisateur doit s'assurer que le compilateur sélectionné représente chaque condition dans le code source par un saut conditionnel ou une instruction conditionnelle au niveau du code objet, c.f. par désactivation des optimisations.

Conclusion

La couverture de code MCDC TRACE32 peut être utilisée indépendamment du compilateur et de l'architecture du processeur. Pour ceux qui ne peuvent pas abandonner les options d'optimisation du code, même dans l'implémentation des systèmes de sécurité critique, il n'y a aucune possibilité de réduire l'optimisation pour les mesures de trace MCDC.

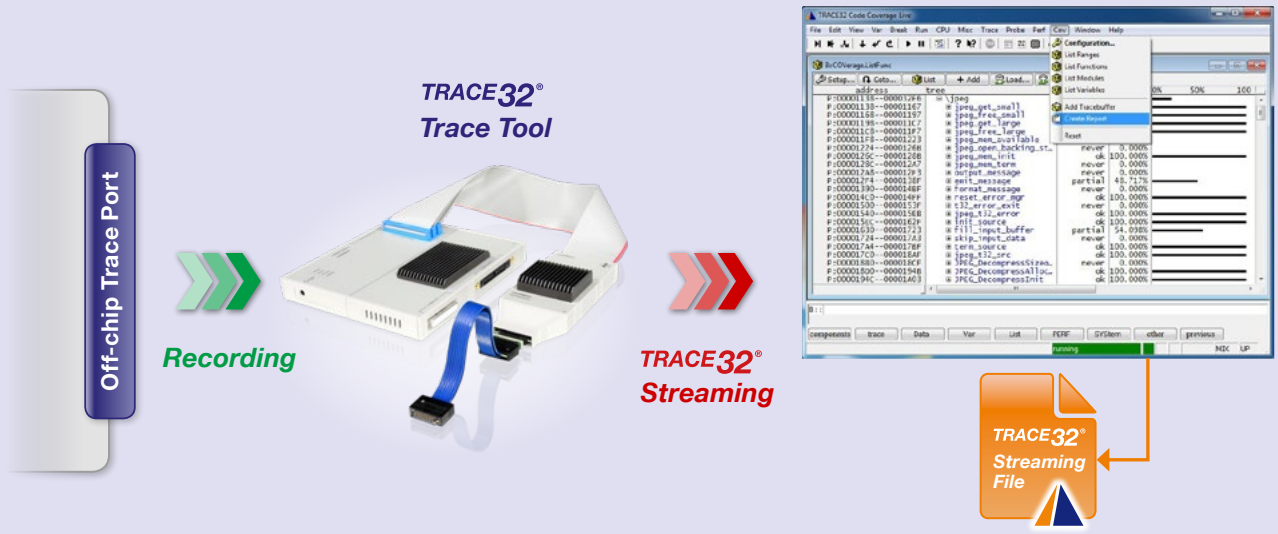
En principe, il serait très appréciable si, dans l'avenir, les compilateurs pourraient être configurés, au niveau du code objet, de façon à ce que les décisions soient traduites seulement en sauts conditionnels ou instructions conditionnelles. Ceci indépendamment du niveau d'optimisation sélectionné. Ces optimisations pourraient être utilisables sur la totalité de la mémoire.

Références

[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

Couverture de Code à la volée

Trace analysé à la volée



Real-time Profiling (RTS) est le nom donné par Lauterbach au mode d'enregistrement des données de trace à la volée vers le pc « host » et leurs analyses immédiates. Ceci permet un suivi en direct à l'écran des résultats d'analyse de la couverture de code. Fin 2017, le mode RTS, qui peut être déjà utilisé pour l'Arm-ETMv3 depuis 2009, a été déployé sur plusieurs protocoles de trace. Le tableau « RTS - Supported Trace Protocols » montre un aperçu de tous les protocoles de trace supportés.

Conditions de base

Les conditions de base ci-dessous s'appliquent à tous les protocoles de trace supportés :

1. L'accès au code programme est nécessaire pour décoder une trace data. Sinon, il serait très lent de lire le code directement dans la mémoire pendant l'exécution du code. Le code doit être chargé dans TRACE32 avant le début de l'analyse à la volée. Ce qui signifie qu'une analyse à la volée peut être lancée uniquement pour les programmes statiques en mémoire.
2. L'analyse à la volée des données de trace fonctionne uniquement lorsque la bande passante des données issue du port de trace n'excède pas la bande passante maximum de transfert au PC « host ». Sachant que les outils de trace actuel TRACE32 sont équipés d'une interface USB3 avec une bande passante maximum d'environ 180 MByte/s. La bande passante a triplé, comparé à ce qui était disponible en 2009.
3. L'analyse à la volée des données de trace peut actuellement être interprétée sur un seul cœur mais aussi pour une session de trace Multicoeur SMP. L'implémentation sur un système AMP de trace Multicoeur est actuellement en cours de développement.
4. L'analyse de la couverture de code à la volée, peut être utilisée sur la couverture des instructions et la couverture des branches mais aussi pour la couverture décisionnelle.

Généralement, les données de traces ne sont pas nécessaires après avoir été traitées. Néanmoins, TRACE32 propose une option permettant de sauvegarder les données de trace pendant l'analyse dans un fichier streaming. Cela permet de pouvoir vérifier les données de trace à postériori, de manière plus détaillée, même si vous avez déjà terminé votre analyse de couverture de code. Comme une analyse de couverture de code classique dans TRACE32, le mode RTS vous permettra de créer un rapport de test compréhensif.

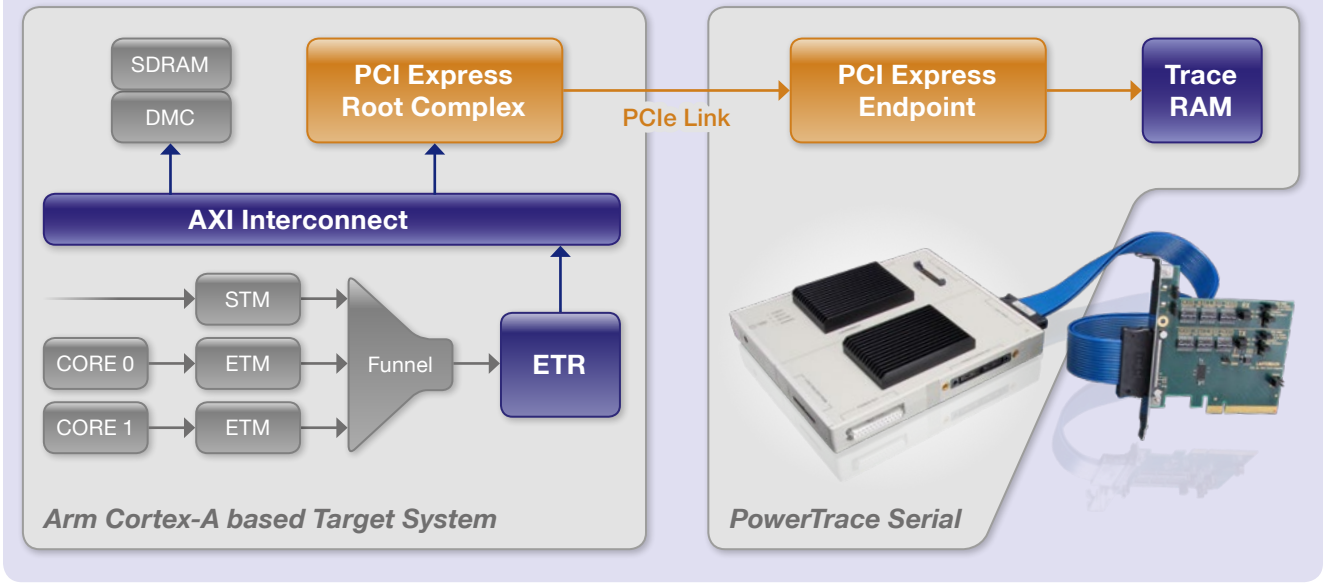
RTS – Protocoles de trace supportés

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

autres protocoles sur demande

Tracer sur PCI Express

TRACE32 PowerTrace Serial en PCIe Endoint



Lorsqu'un composant est équipé d'une interface PCIe, il est possible de l'utiliser pour enregistrer et analyser les données de trace grâce à un outil externe de trace. Plusieurs clients Lauterbach l'ont adopté en avance de phase, sur des architectures Arm Cortex-A® ou bien sur des processeurs NXP Power QorIQ®.

L'outil de Trace vu comme un PCIe EndPoint

Tout d'abord, essayons de voir comment fonctionne la trace à travers le bus PCI Express. L'outil de Trace, TRACE32 PowerTrace Serial, a été développé de façon à être reconnu comme un EndPoint PCIe. Ainsi, l'outil de trace devra être connecté à la cible et devra déjà être en fonctionnement avant que l'application (le boot loader) ne démarre l'énumération et la configuration des Endpoints. Pendant cette configuration, chaque endpoint se verra alloué une zone d'adressage dans la mémoire système. Ensuite, les données de cet Endpoint, pourront être simplement écrites dans cette allocation.

L'infrastructure de trace sur cible devra être configurée pour permettre l'écriture des données de trace dans la zone de mémoire préalablement allouée à l'Endpoint TRACE32 PowerTraceSerial. C'est actuellement une fonctionnalité disponible sur une grande majorité de processeurs. Après cela, la trace peut démarrer.

Caractéristiques

Bande passante variable
- Gen1 250 MByte/s par Lan
- Gen2 500 MByte/s par Lan
- Gen3 984 MByte/s par Lan

Nombres de port variable (1, 2, 4 or 8 Lan)

Adaptateur grande taille disponible
carte mini PCIe en prévision

4Gygaoctet de mémoire de trace
décodage de tous les protocoles standard

Sommaire

Tracer grâce au bus PICe est facile à mettre en œuvre. Pour des cibles ne possédant aucune interface de trace dédiée, cette technique est une excellente méthode pour enregistrer de grandes quantités de données de trace. Les principales différences entre la trace sur PCIe et l'utilisation d'un port de trace dédié peuvent se résumer comme suit :

- L'enregistrement de la trace ne peut démarrer qu'après l'initialisation du Root Complex PCIe par l'application.
- En même temps, tracer sur PCIe n'est plus, au sens strict du terme, « non intrusif ». Cela nécessite une partie de la mémoire système. De plus, la trace est en compétition avec les autres endpoint pour la bande passante du bus PCIe.

Transition en douceur Wind River / TRACE32

Depuis 2014, Wind River ne fournit plus de débuggeur JTAG. Les utilisateurs existant basculent de plus en plus vers TRACE32 pour la maintenance ou les nouveaux développements de leurs produits. En relation étroite avec Wind River, Lauterbach a fait évoluer régulièrement son logiciel TRACE32 ainsi, tous les ajustements sont implémentés de manière à assurer aux utilisateurs une parfaite continuité dans leur façon de travailler.

Support de tous les procésseurs

La plupart des clients qui basculent sur nos solutions TRACE32, utilisent des processeurs de la famille Power Architecture™. Le support de cette architecture fait partie des produits Lauterbach depuis 1997. Les clients basculant sur TRACE32 peuvent donc s'appuyer sur une solution de debugge très solide.

Conversion de scripts

Avant de pouvoir commencer une session de debugge, l'application est normalement programmée sur flash cible. TRACE32 utilise des scripts pour cela. La configuration des registres processeurs et en particulier la SDRAM, est une composante importante du script. Wind River définissait ce type de configuration dans un fichier appelé « Register Configuration File ». Lauterbach propose un convertisseur, permettant de traduire ce fichier en script TRACE32. A travers des années de debugge professionnel, les clients ont accumulé un grand nombre de scripts de test, sur différents sujets. Lauterbach propose un convertisseur de script, permettant aussi de transférer ces scripts complexes vers TRACE32.

Workbench Wind River

Basculer vers un nouveau débuggeur après avoir utilisé le même système pendant des années peut être un moment difficile à passer pour un ingénieur. C'est pour cette raison qu'en 2015, Lauterbach a fait évoluer ces débuggeurs pour fonctionner avec l'agent TCF. Il est désormais possible d'utiliser le Workbench Wind River comme IDE de debugge avec TRACE32 actif en backend.

Support de tous les produits Wind River

De très nombreux produits intègrent les logiciels Wind River, par exemple : Wind River VxWorks, Wind River Linux ou Wind River Hyperviseur. Les débuggeurs Wind River proposaient une solution de debugge avancé pour ces produits. Lauterbach est un fabriquant d'outils expérimenté dans ce domaine et TRACE32 supporte le débugge de VxWorks Wind River depuis 1996. Le support complet de Linux Wind River fut ajouté en 2000. Aujourd'hui, TRACE32 supporte toujours les produits Wind River qui ne sont plus supportés par le Workbench Wind River. Incluant, Vxworks 7, Vxworks 653 v3 ainsi que Wind River Hypervisor v3.

Débuggeur TRACE32 pour RISC-V

JTAG Débuggeur

Lauterbach a lancé son nouveau débuggeur RISC-V en Novembre 2017. Le premier composant désormais supporté est le cœur Complex E31 (32 bit) et le nouveau cœur Complex E51 (64-bit) par SiFive.

RISC-V est une architecture avec son jeu d'instructions open source (ISA), basée sur les principes établis RISC. Organisée, spécifiée et développée sous la direction de la fondation RISC-V (https://riscv.org). Initialement mise en œuvre pour un institut de recherche, RISC-V arrive désormais sur le marché de l'embarqué, et nécessite un débuggeur matériel professionnel.

L'implémentation du débuggeur Lauterbach pour RISC-V est basée sur les spécifications open source "RISC-V External Debug Support" qui devraient être adoptées par la fondation RISC-V en 2018. L'objectif de cette spécification est un débugge avec mode d'arrêt flexible. Chaque tâche matérielle d'un cœur RISC-V pourrait être débuggée directement à partir du reset.

Pour être en mesure de debugger avec TRACE32 les processeurs RISC-V, ceux-ci doivent être équipés obligatoirement d'un JTAG DTM (Debug Transport Module). Le DTM est un module indépendant et remplaçable, permettant au fabriquant de processeur d'avoir la liberté d'implémenter un accès au DTM à travers différentes interfaces de communication. Grâce à sa grande expérience avec les différentes interfaces de communication pour le débugge, Lauterbach est le partenaire compétant pour cette implémentation.

Grâce au module de débugge DTM, le débuggeur TRACE32 accède à toutes les fonctionnalités standard de débugge. Cela fait partie des spécifications comme les registres de débugge et de ce que l'on appelle les « Abstract Commands ». De plus, les spécifications permettent aussi de développer ses propres fonctionnalités de débugge. Le Débuggeur RISC-V supporte déjà plusieurs extensions standard ISA, comme par exemple les instructions compressées ou les instructions floating-point mais aussi des extensions ISA, client spécifique.