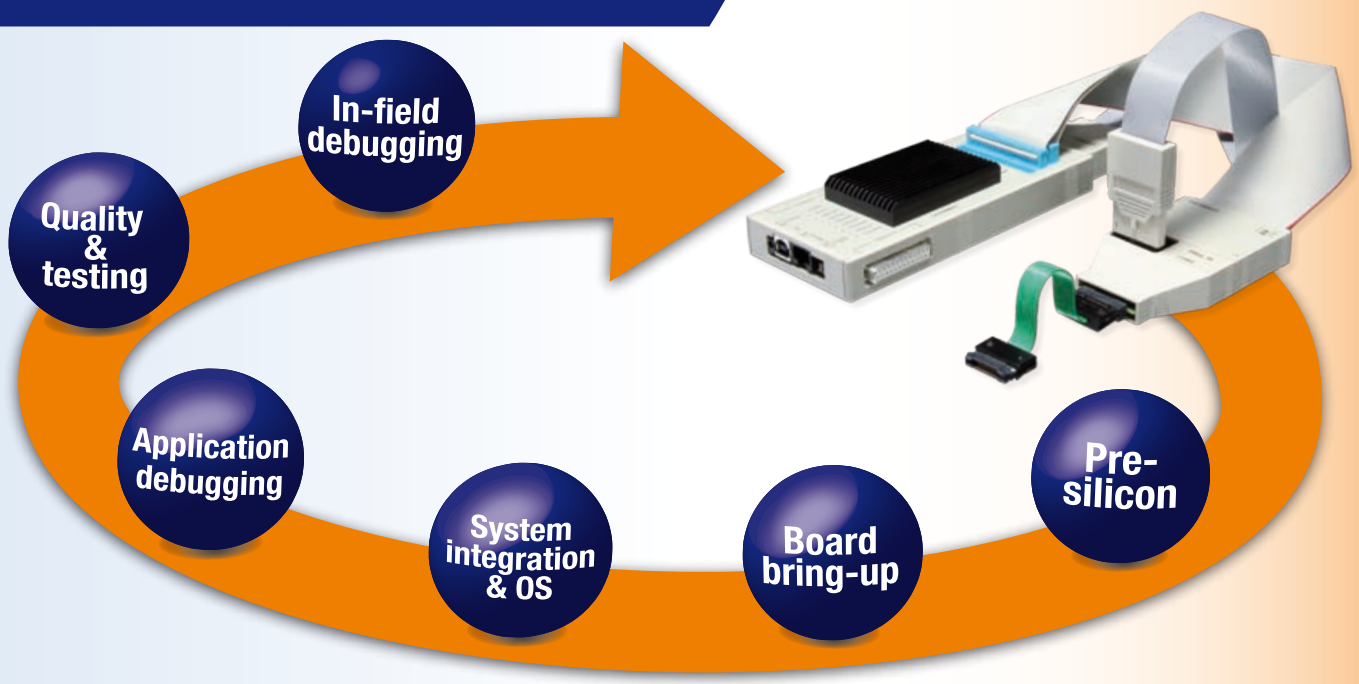
NEWS 2012

LAUTERBACH
DEVELOPMENT TOOLS

DÉBUGGEUR, TRACE TEMPS-RÉEL, ANALYSEUR LOGIQUE



Un débuggeur pour toutes les phases de votre projet

Les systèmes embarqués deviennent de plus en plus complexes, et les dates de livraison de plus en plus courtes. Afin de respecter les délais, les entreprises misent sur les outils de debugge et de trace qui accompagnent les développeurs, dans toutes les phases du projet.

TRACE32, l'outil de debugge et de trace de Lauterbach, a fait ses preuves comme le compagnon de développement par excellence.
La même logique d'utilisation, une interface conviviale et des scripts extensibles, sont des élements qui facilitent l'adaptation à l'outil. Il est très commun pour des ingénieurs d'exploiter la puissance du TRACE32 tout au long de leur carrière.

Qu'est-ce qui distingue alors TRACE32 ?

- Un outil de développement basé sur le matériel et le logiciel
- Un support précoce de tous les nouveaux processeurs
- Support d'un très large panel de processeurs
- Des fonctions étendues de test et d'analyse
- Une intégration parfaite dans la chaîne de développement
- Une continuité de la logique d'utilisation, de la convivialité et du langage de script

Outils matériels et logiciels

Le cœur de métier de Lauterbach est essentiellement le développement d'outils matériels de debugge et de trace. Cependant, Lauterbach offre également depuis plus d'une vingtaine d'année des analyseurs logiques qui s'intègrent parfaitement aux outils de debugge et de trace.

Un scénario d'application typique d'un tel analyseur logique est présenté dans l'article « Vérification des signaux JTAG » en page 6.

»

CONTENU

Grâce à la puissance croissante des ordinateurs, la simulation et validation pré-silicone des processeurs et SoCs se fait de plus en plus sur des cibles virtuelles. Afin de couvrir les besoins de cette phase de développement, Lauterbach fourni également des solutions de simulation purement logicielles.

Validation pré-silicone

Pour les fabricants de semi-conducteurs, il est important de valider leurs nouveaux processeurs ou SoCs avant production. Lors de cette validation, plusieurs composants sont intensivement testés comme l'interface JTAG, le noyau et l'interaction entre le noyau et les périphériques.

Pour cela, des émulateurs pré-silicone, comme le palladium ou bien des prototypes FPGA, sont habituellement connectés à l'outil de debugge et de trace TRACE32, généralement via une interface JTAG. Malgré tout, le système fonctionnera beaucoup plus lentement comparé à un processeur réel.

Aujourd'hui, il est possible d'effectuer les premières validations des modèles Verilog ou bien SystemC directement sur PC ou sur station de travail. Pour une telle validation, les outils matériels de debugge ne peuvent être utilisés. C'est pourquoi, Lauterbach offre depuis 2011 une solution Verilog Back-End qui simule les signaux de l'interface JTAG (voir figure 1).

L'utilisation des outils TRACE32 dans la validation pré-silicone représente une partie importante dans le support en avance de phase des futurs processeurs et SoCs :

- Des outils testés sont disponibles avant même que le premier silicone ne quitte l'usine.
- Une expertise sur les nouveaux processeurs/SoCs est mise au service de nos clients.
- Des scripts de démarrage pour le débuggeur TRACE32 sont déjà disponibles.

Les cibles virtuelles

Aujourd'hui, les cibles virtuelles sont de plus en plus utilisées et permettent de démarrer les développements logiciels très tôt avant la disponibilité des premiers prototypes. Dès que la cible virtuelle existe, le debugge des pilotes, du système d'exploitation et des applications commence.

Pour le debugge et la trace, la plupart des cibles virtuelles disposent de leur propre API. Si ce n'est pas le cas, l'API standard MCD peut toujours être utilisées (**http://www.lauterbach.com/mcd_api.html**). De nombreux projets utilisent désormais des systèmes multi-cœurs, par conséquent, Lauterbach a étendu en 2011 le debugge multi-cœurs sur des cibles virtuelles.

Plus de 60 architectures supportées

Lauterbach fourni des outils de développement pour tous les processeurs utilisés dans le domaine de l'embarqué. En fait, Lauterbach est le seul fournisseur du marché supportant toutes ces architectures. Les outils TRACE32 peuvent debugger toutes les combinaisons multi-cœur possibles, que ce soit des contrôleurs standards, des DSPs, des processeurs softcores ou bien des noyaux configurables.
La liste des nouveaux processeurs et systèmes multi-cœurs supportés en 2011 est en page 4.

Les fonctions de test et d'analyse

Chaque étape du projet exige ses propres fonctions de test et d'analyse.

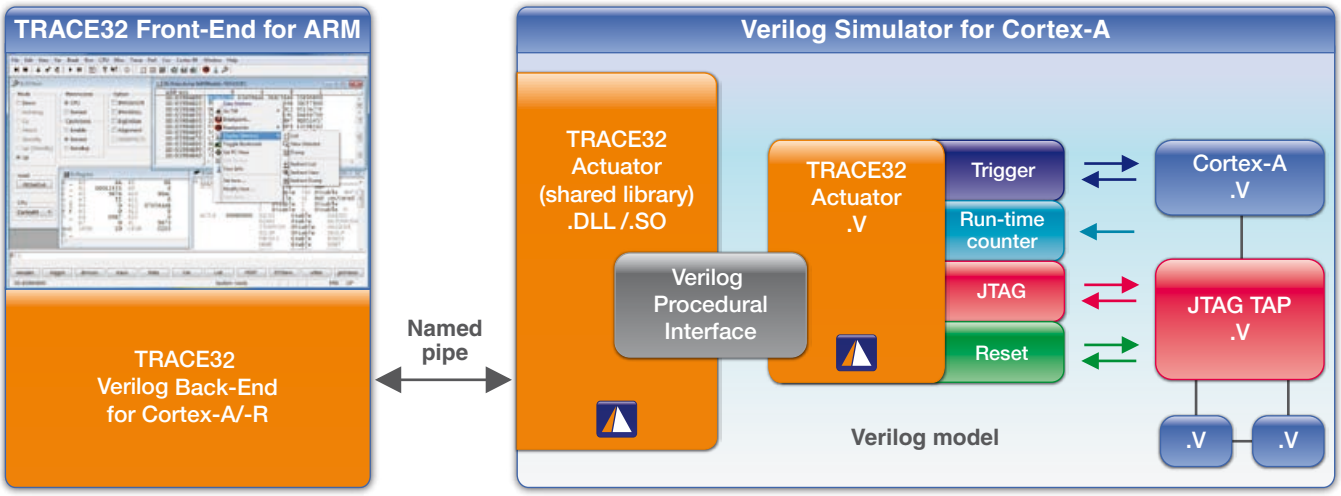


Fig. 1: TRACE32 Verilog Back.End.

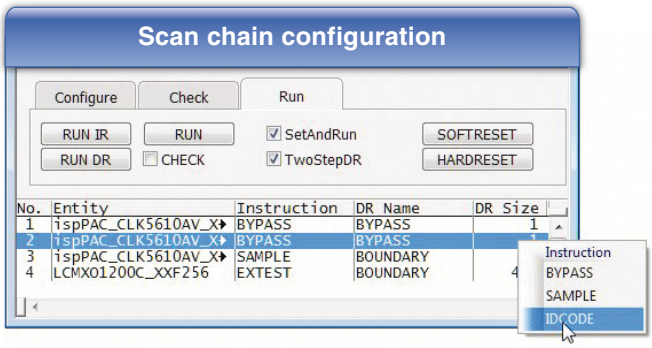


Fig. 2: TRACE32 offre plusieurs commandes pour la technique Boundary-Scan.

TRACE32, offre une multitude de commandes et de menus. Les commandes Boundary-Scan (voir figure 2) et les commandes pour la manipulation des signaux JTAG représentent des exemples concrets de commandes bas niveau.

Pendant la phase de test qualité, les fonctions TRACE32 de plus haut niveaux aident l'utilisateur dans sa tâche. Il s'agit typiquement ici de commandes avancées pour l'évaluation des données de trace. On peut citer par exemples la mesure des temps d'exécution de fonctions, l'analyse de la consommation en d'énergie ainsi que l'analyse de couverture de code.

Depuis le début de l'année 2011, Lauterbach offre pour toutes les architectures processeurs la possibilité de diffuser, lors de l'enregistrement, les informations de trace vers le système hôte. Il est possible de cette manière de collecter beaucoup plus de données. Ceci simplifie considérablement l'analyse qualitative. Voir l'article « Couverture de code - Simplifiée » en page 7.

Intégration au développement

Le logiciel TRACE32 est conçu d'une manière ouverte ce qui lui permet d'interagir sans difficulté avec tous les composants habituels d'un design embarqué. Ceci inclut :

- Plusieurs systèmes d'exploitation hôte
- Différents langages de programmation et différents compilateurs
- Une multitude de systèmes d'exploitation cible
- Des machines virtuelles comme la *VM Dalvik d'Android*

L'API ouverte de TRACE32 permet également une interaction transparente avec une variété d'outils tiers comme celui d'Eclipse, les langages de programmation graphique et les outils d'analyse externes. Dans ce contexte, il y a eu quelques nouveautés en 2011.

Prism, l'environnement de développement et d'analyse multi-cœurs de la société écossaise CriticalBlue, aide les développeurs à exploiter la totalité du potentiel de leurs systèmes multi-cœurs. Il permet en effet d'essayer plusieurs stratégies de parallélisassions sans modifier le code. Une fois la stratégie optimale déterminée, la parallélisassions peut être alors effectuée. Depuis Juillet 2011, Lauterbach offre la possibilité d'enregistrer les informations de trace en format Prism présentant par la suite les données de base nécessaires à une parallélisassion.

L'article « Une simulation très proche de la réalité » en page 14 montre une autre nouveauté, à savoir, l'intégration entre MATLAB Simulink® et TRACE32.

Une continuité assurée

Il est essentiel pour Lauterbach d'assurer de longues périodes de transition, lors d'une migration vers une nouvelle technologie.
A partir de Mai 2012, Lauterbach fournira une nouvelle version de son interface graphique TRACE32 basée sur QT (figure 3). La bibliothèque QT sera utilisée dans le but de fournir une interface standard pour Linux, Mac OS X et d'autres systèmes d'exploitation.
Lauterbach continuera le support de TRACE32 basé sur l'ancienne bibliothèque graphique « Motif », de telle sorte que si une transition doit être faite, elle se fera plus facilement dans le temps.

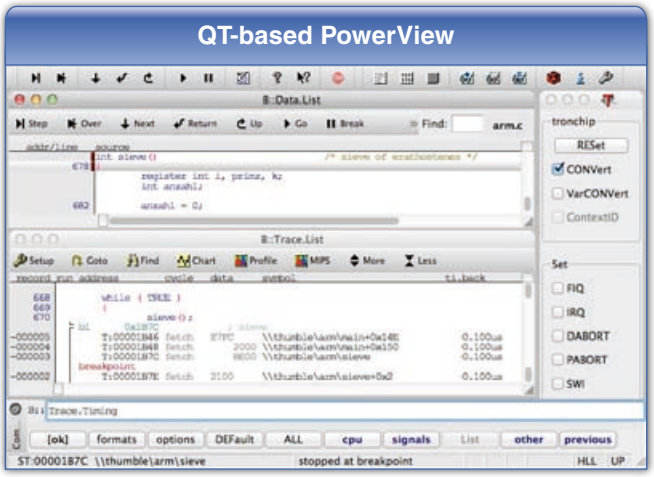


Fig. 3: La nouvelle interface graphique basée sur QT pour Linux, Mac OS X et d'autres systèmes d'exploitation.

En feuilletant notre lettre d'information 2012, vous serez probablement intéressés par certaines nouveautés qui pourraient concerner votre projet. Nous présenterons plusieurs d'entre elles du 03 au 05 Avril au salon RTS Embedded Systems, Paris, Porte de Versailles.

Venez-nous rendre visite:

Hall 7.1 Stand C2

Nouveaux processeurs supportés

Nouveaux dérivés

Altera	**Cortex-A/-R** • FPGA avec Hardcore Cortex A9 MPCore **MIPS32** • MP32
AppliedMicro	**PPC44x** • 86290/491/791 Q2/2012
ARM	**Cortex-A/-R** • Cortex-A7/Cortex-A7 MPCore • Cortex-A15 • Cortex-A15 MPCore • Cortex-R5/Cortex-R5 MPCore • Cortex-R7/Cortex-R7 MPCore
Beyond Semiconductor	**Beyond** • BA22
Broadcom	**MIPS32** • BCM35230 • BCM63168, BCM63268 • BCM7231, BCM7358
Cavium	**MIPS64** • CN61XX/CN62XX/CN66XX • CN67XX/CN68XX
Ceva	**CEVA-X** • CEVA-XC
CSR	**ARM11** • QUATRO 4500
Cypress	**ARM9** • EZ-USB FX3
Energy Micro	**Cortex-M** • Giant Gecko
Freescale	**MCS08** • S08LG **MCS12X** • MC9S12VR, MC9S12XS • MM912F634 **Cortex-A/-R** • i.MX 6 Series **MPC55xx/56xx** • MPC5604E, MPC5675K, • MPC5676R **QorIQ 32-Bit** • P1010, P1020 • P2040, P2041 • P3041, P4040, P4080 • PSC9131
Freescale (cont.)	**QorIQ 64-Bit** • P5010, P5020
Fujitsu	**Cortex-A/-R** • MB9DF126, MB9EF126
IBM	**PPC44x** • 476FP Q2/2012
Ikanos	**MIPS32** • Fusiv Vx185
Infineon	**TriCore** • TriCore Multi-Core Architecture
Intel®	**Atom™/x86** • Atom D2500, Atom N550 • Core i3/i5/i7 2nd Generation
Lantiq	**MIPS32** • XWAY xRX100
LSI	**PPC44x** • ACP344x Q2/2012
Marvell	**ARM9** • 88E7251 **ARM11** • 88AP610-V6, MV78460-V6 **Cortex-A/-R** • 88AP610-V7, MV78460-V7
Nuvoton	**Cortex-M** • NuMicro
NXP	**Cortex-M** • LPC12xx **Beyond** • JN5148
Qualcomm	**MIPS32** • AR7242 **Cortex-A/-R** • Krait
Renesas	**V850** • V850E2/Fx4: 70F3548..66 70F4000..70F4011 • V850E2/Fx4-L: 70F3570..89 • V850E2/Px4: 70F3503/05 70F3507/08/09 **78K0R/RL78** • 78K0R/Kx3-C/L • RL78/G14, RL78/G1A • RL78/F12, RL78/I1A **H8SX** • H8SX1725

Nouveaux dérivés

Renesas (cont.)	**SH** • SH708x avec AUD/Onchip-Trace • SH7147
Samsung	**ARM7** • S3F4 **Cortex-A/-R** • S5PV310 **Cortex-M** • S3FM, S3FN
ST-Ericsson	**Cortex-A/-R** • A9500, A9540, M7400 **MMDSP** • A9500, A9540
STMicro-electronics	**MPC55xx/56xx** • SPC56A80, SPC56HK **Cortex-M** • STM32F2xx, STM32F4xx
Synopsys	**ARC** • ARC EM4, ARC EM6
Tensilica	**Xtensa** • BSP3, LX4, SSP16
Texas Instruments	**MSP430** • CC430Fxxx, MSP430FR5xxx • MSP430x1xx..MSP430x6xx **ARM9** • AM38xx • OMAP4460/4470 • TMS320C6A81xx • TMS320DM81xx **Cortex-A/-R** • AM335x, AM38xx • OMAP4460/4470/543x • RM48L950 • TMS320C6A81xx • TMS320DM81xx • TMS570LS3xxx **Cortex-M** • AM335x • OMAP4460/4470/543x • TMS470MFxxx **TMS320C28X** • TMS320C28346/F28069 **TMS320C6x00** • OMAP4460/4470/543x • TMS320C6A81xx • TMS320DM81xx • TMS320TCI6616/18
Xilinx	**Cortex-A/-R** • Zynq7000

La trace Nexus enfin disponible sur les petits boîtiers

La cellule de trace Nexus, intégrée entre autre sur les familles Freescale MPC560xB/C ainsi que sur les familles SPC560B/C de chez ST, permet de produire une trace instructions exécutées par le processeur. Si un système d'exploitation est utilisé, la trace Nexus permettra aussi de visualiser le basculement des tâches.

Afin de donner la possibilité à un outil de trace externe comme TRACE32 d'enregistrer ces données, le micro-contrôleur doit disposer d'une interface de trace off-chip. Toutefois, les contrôleurs de la famille X560XB/C ne disposent pas dans leurs boîtiers standards de ce type d'interface. Il est néanmoins possible de ne pas se priver de ces précieuses informations de trace, grâce à de nouveaux boîtiers au format BGA 208 broches, Silicium compatibles. Ces boîtiers disposent désormais d'un port NEXUS de 4 broches MDO (*Message Data Out*).

Lauterbach propose depuis mi-2011 des adaptateurs permettant de remplacer le processeur sur votre cible, par sa version qui dispose de l'interface de trace Nexus. En effet, il est désormais possible d'utiliser un support de type « Tokyo Eletech » sur la cible permettant ainsi l'utilisation alternativement du processeur d'origine ou bien de l'adaptateur avec l'interface NEXUS.

L'adaptateur est composé des éléments suivants :

- Un processeur MPC560x dans sa version BGA 208-broches possédant un connecteur Mictor et exploitant l'interface Nexus sur lequel viendra se connecter l'outil TRACE32 (en bleu sur la figure 4).
- Un support de type « Tokyo Eletech »

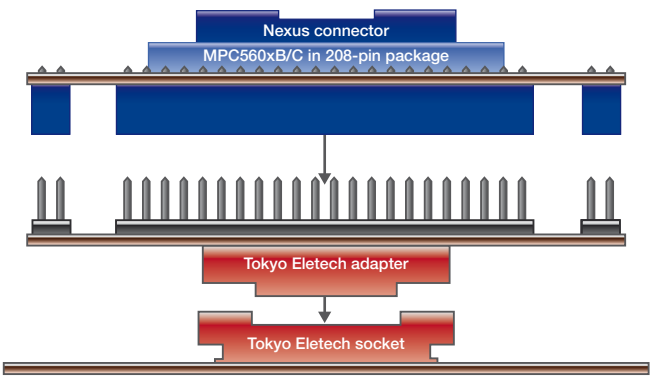


Fig. 4: L' adaptateur MPC560x permet de connecter le kit de développement disposant d'une interface Nexus, à la place du CPU d'origine.

Vérification des signaux JTAG

Le PowerTrace II inclut un analyseur logique. Il est livré avec une sonde 17 voies permettant d'enregistrer 17 signaux logiques avec une fréquence d'échantillonnage jusqu'à 200 MHz.

Grâce à sa mémoire de 1024 K, il permet par exemple de tester les signaux JTAG lors de la validation pré-silicone ainsi que lors de la mise en œuvre des cibles d'évaluation (voir figures 6 et 7).

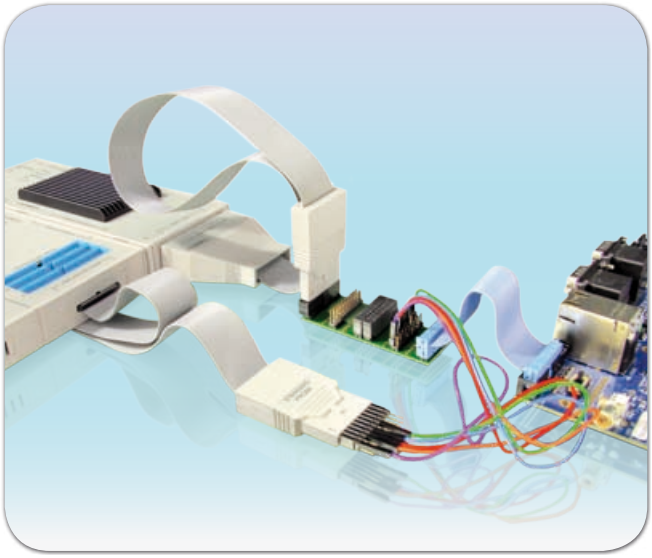


Fig. 5: L'analyseur logique Iprobe intégré dans la sonde PowerTrace II.

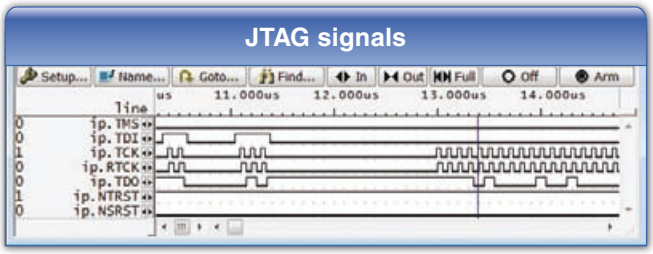


Fig. 6: Les signaux JTAG enregistrés.

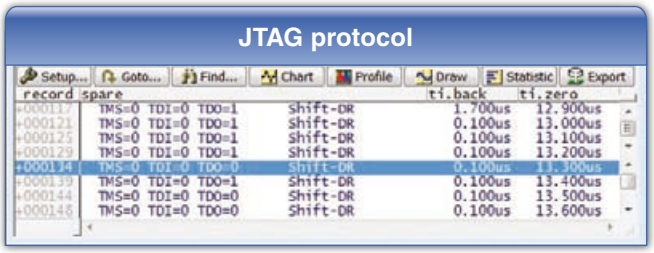


Fig. 7: L'affichage du protocole JTAG.

Nouveaux RTOS supportés

Les versions RTOS suivantes ont été intégrées :

- eCos 3.0
- embOS 3.80
- FreeRTOS v7
- Linux v3.0
- MQX 3.6
- RTEMS 4.10
- SMX v4

- Le contenu du *tracelogger de QNX* peut être affiché en utilisant la QNX OS-Awareness. Les commandes « LOGGER » de TRACE32 permettent une représentation graphique du basculement des tâches.

- L'awareness QNX a été adaptée pour permettre l'utilisation d'exécutable « *position-indépendant* ».

Nouveaux RTOS supportés	
µC/OS-II pour Andes	disponible
µC/OS-II pour Blackfin	disponible
Elektrobit tresos (OSEK/ORTI)	disponible
Erika (OSEK/ORTI)	disponible
FreeRTOS pour AVR32	disponible
Linux pour Beyond	prévu
MQX pour ARC	disponible

OSEK/ORTI SMP	prévu
PikeOS	disponible
PXROS-HR pour TriCore	disponible
PXROS-HR Run Mode Debugging	disponible
RTEMS pour Nios II	disponible
Sciopta 2.x	disponible
SYS/BIOS pour ARM	disponible
VxWorks SMP	disponible

Couverture de code – Simplifiée

Depuis le mois de Mars 2011, TRACE32 est capable de transmettre « à la volée » les informations de trace vers le PC. Ceci implique une grande quantité d'informations sur l'exécution code et permet une amélioration significative sur l'analyse de la couverture de code.

Couverture de code basée sur la trace

L'assurance qualité de produits ayant une notion de sécurité requière parfois strictement une preuve de *couverture de code exécuté* et de *couverture conditionnelle*.

- La **couverture de code exécuté** prouve que chaque ligne du code a été réellement exécutée ou pas.
- La **couverture conditionnelle** prouve que pour chaque instruction conditionnelle, chacune des branches « vraies » ou « fausses » ont été exécutées au moins une fois.

Pour beaucoup de systèmes embarqués, le code, hautement optimisé, doit obligatoirement être testé en temps réel. L'instrumentation du code ou bien le changement du comportement temporel du système ne sont pas autorisés.

Pour répondre à ces exigences, le processeur/SoC utilisé doit remplir les conditions suivantes :

1. Les processeurs utilisés doivent disposer d'une logique de trace (figure 8) produisant au moins une trace de code. En plus, des informations supplémentaires sur le basculement des tâches ainsi que sur les accès mémoire effectués.

2. Le processeur/SoC doit disposer d'un port de trace avec suffisamment de bande passante pour que l'outil externe puisse récupérer, sans perte, les données produites.

La méthode de mesure classique

L'analyse de couverture de code par TRACE32 a été réalisée jusqu'à présent en suivant les étapes suivantes :

1. Démarrer le programme puis l'arrêter automatiquement quand la mémoire de trace est totalement remplie.
2. Transférer le contenu de la mémoire de trace vers une base de données.
3. Redémarrer le programme.

Pour chaque étape de mesure, le nombre de données saisies est limité par la taille de la mémoire de l'outil de trace.

Si nécessaire, les résultats de la couverture de code peuvent être, après chaque étape intermédiaire, vérifiés.

La nouveauté : Le « *Streaming* »

En utilisant la méthode *Streaming*, les données de trace sont transférées « à la volée » vers le PC pendant l'exécution CPU. Par la suite, toutes les informations nécessaires peuvent être récupérées en **une seule étape de mesure**. Les données transférées sont alors enregistrées dans un fichier temporaire sur le disque dur. Pour éviter un débordement du disque dur par les données de trace, TRACE32 arrête la diffusion des données si la mémoire restante sur le PC ne dépasse pas 1 Go.

Pour pouvoir utiliser la méthode *Streaming*, les exigences techniques suivantes doivent être remplies :

- Un PC ainsi qu'une exécutable TRACE32 64-bit
- Une interface rapide entre l'outil de trace et le PC
- Une configuration optimale de la source et de l'outil de trace

»

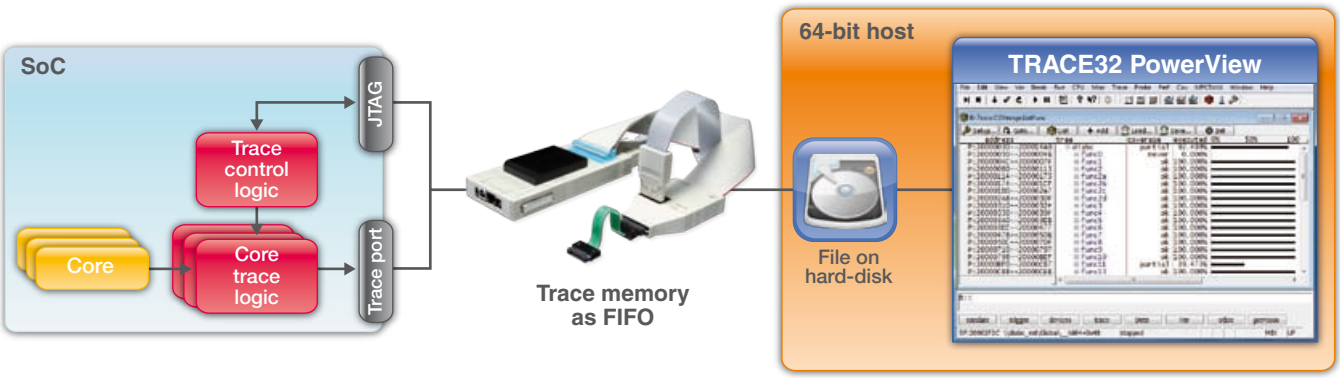


Fig. 8: Jusqu'à 1To de données de trace peuvent être diffusées vers le PC afin d'être utilisées pour l'analyse de couverture de code.

Une interface hôte rapide

La quantité de données de trace exportées via l'interface de trace off-chip dépend de la configuration matérielle du système cible. Le nombre de processeurs, le nombre de broches de trace, la fréquence ainsi que le protocole de trace utilisée sont des paramètres importants. Pour l'architecture ARM, le protocole de trace ARM PTM est beaucoup plus compact que le protocole ARM ETMv3 (figure 9).

Un autre facteur important est le logiciel embarqué. Un programme qui effectue de nombreux branchements et qui trouve, la plupart du temps les données dans le cache, produit beaucoup plus de données de trace par seconde qu'un programme qui exécute beaucoup d'instructions séquentielles et qui doit attendre que les données soient disponibles.

La quantité de données est dans tous les cas importante. La méthode du *Streaming* ne fonctionne que si toutes les données de trace sont transférées sans perte vers le PC donc, si la vitesse de transfert entre l'outil de trace et le PC est suffisante. L'interface Ethernet Gigabit est recommandée sur la sonde PowerTrace II de Lauterbach.
Dans le but de réduire la quantité de données produites, il est possible de programmer la logique de trace d'une façon que seules les informations nécessaires à l'analyse de couverture de code sont générées. Vous trouverez ici deux exemples :

ETM/PTM : configuration optimale

Pour les architectures ARM/Cortex, il existe deux différentes implémentations de la logique de trace à savoir l'ETM et PTM. Il est possible de configurer l'ETM pour avoir seulement une trace instruction exécutée. La trace des accès mémoire (Read/Write) effectués n'est pas nécessaire pour l'analyse de couverture de code. La PTM inclut seulement une trace de code donc cette configuration n'est pas nécessaire.

Dans les deux cas, les paquets de la trace contiennent seulement les adresses virtuelles des instructions exécutées. Si un système d'exploitation est présent sur la cible,

PowerTrace vs. PowerTrace II

Les outils de trace TRACE32 sont disponibles en deux variantes avec différentes caractéristiques :

PowerTrace

- Mémoire de trace : 256 ou bien 512 Mo
- USB 2.x et 100 MBit Ethernet
- Débit de transfert maximal : 80 MBit/s
- Compression logicielle des données de trace (facteur 3)
- Interface mémoire avec 100 MHz

PowerTrace II

- Mémoire de trace : 1/2/4 Go
- USB 2.x et 1 GBit Ethernet
- Débit de transfert maximal : 500 MBit/s
- Compression matérielle des données de trace pour ETMv3 et PTM (facteur 6)
- Interface mémoire avec 233 MHz

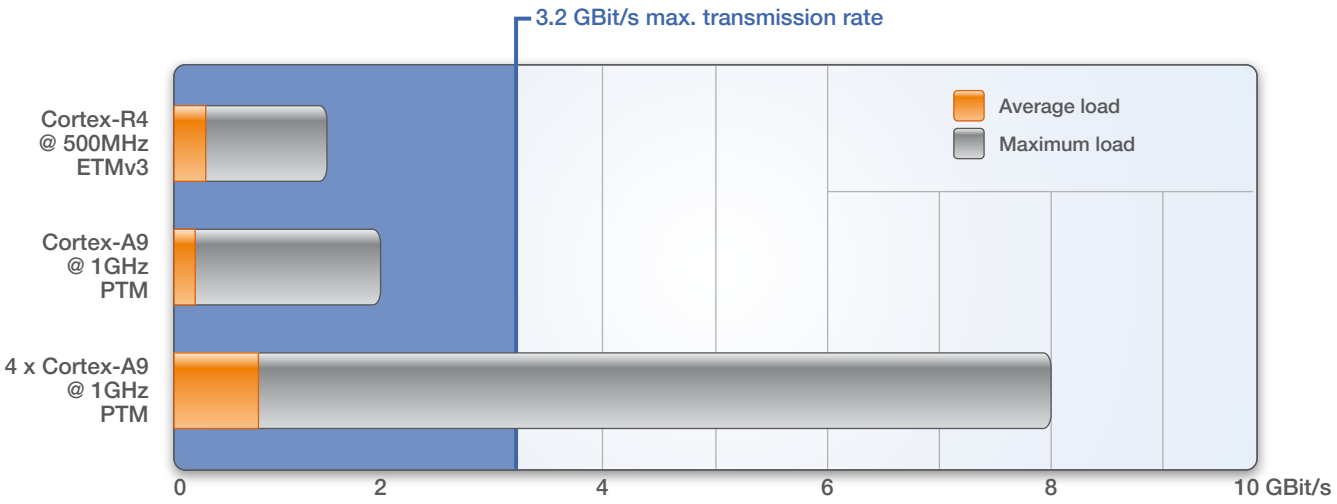


Fig. 11: Un débit de transfert de 3,2 GBits/s est généralement suffisant pour diffuser toutes les informations sur l'exécution du programme vers le PC.

il n'est pas possible de déterminer l'adresse physique à partir de l'adresse virtuelle puisque la même adresse virtuelle peut correspondre à plusieurs adresses physiques. Dans ce cas, la source de trace doit être configurée pour produire des informations supplémentaires sur les espaces d'adressage virtuels. Il existe également d'autres méthodes pour encore réduire la quantité des données de trace :

- Les informations temporelles ne sont pas nécessaires pour l'analyse de la couverture de code. Il est par conséquent recommandé de configurer l'outil de trace TRACE32 de manière à ce que les données soient envoyées vers le PC sans time-stamps. De cette façon, il est possible de réduire la quantité de données d'un tiers.
- La sonde PowerTrace II offre une compression matérielle des données de trace basée sur un FPGA. En utilisant cette compression, il est possible d'atteindre un débit de transmission de 3,2 Gbits/s. La figure 9 montre que cette vitesse de transfert est généralement suffisante pour diffuser sans aucune perte les informations produites par l'ETM/PTM vers le PC.

Nexus : Configuration optimale

Le standard Nexus est implémenté sur les familles de processeurs MPC5xxx/SPC5xx. Pour la couverture de code, une cellule de trace Nexus classe 2 est nécessaire. Les données de trace sont particulièrement compactes si la méthode « *Branch History Messaging* » est utilisée. Grâce à cela, il sera possible de réduire les données de trace par dix.
Actuellement, seulement la sonde PowerTrace II inclut le support de la méthode *Streaming* pour les cellules de trace Nexus.

Bien sûr, le *Streaming* fonctionne aussi pour tous les autres processeurs/SoC supportés par TRACE32.

Couverture de code processeurs multi-cœurs

En utilisant TRACE32, il est également possible d'effectuer une analyse de couverture de code pour les systèmes multi-cœurs. Il est ici nécessaire de déterminer si une instruction a été exécutée ou pas. Il n'est pas important de savoir quel processeur a exécuté le code.

La figure 10 montre l'analyse de couverture de code d'un système SMP composé de deux Cortex-A9 MPCores.

Une instruction conditionnelle toujours fausse est marquée ici en jaune avec le commentaire « not exec ». Cette analyse de couverture de code détaillée montre le nombre d'exécutions de chaque instruction ou branche conditionnelle.

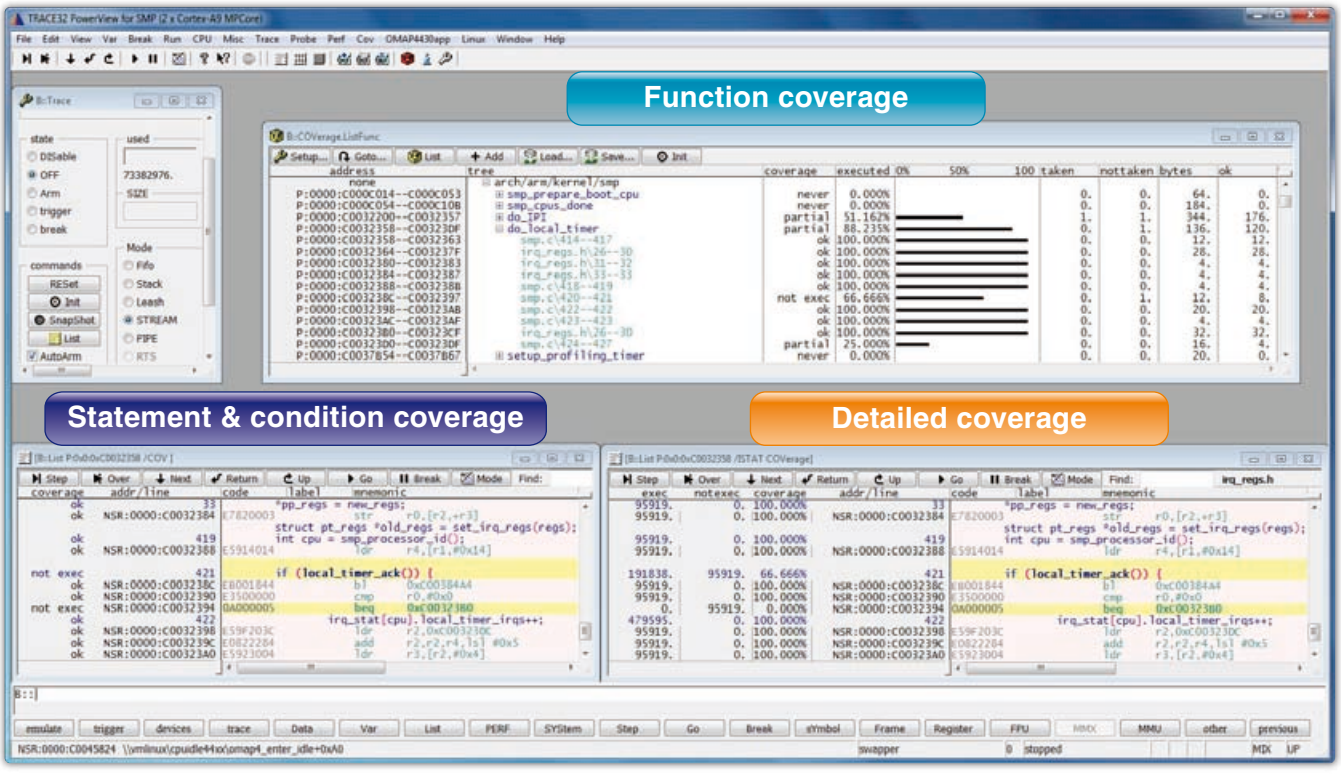


Fig. 10: Analyse de couverture de code pour un système SMP.

CoreSight : *Trace Memory Controller*

Le nouveau contrôleur de mémoire de trace (*Trace Memory Controller*, TMC) intégré dans CoreSight, offre aux designers de SoCs une plus grande liberté dans la conception de l'infrastructure de trace. TRACE32 inclut déjà le support des premiers designs utilisant TMC.

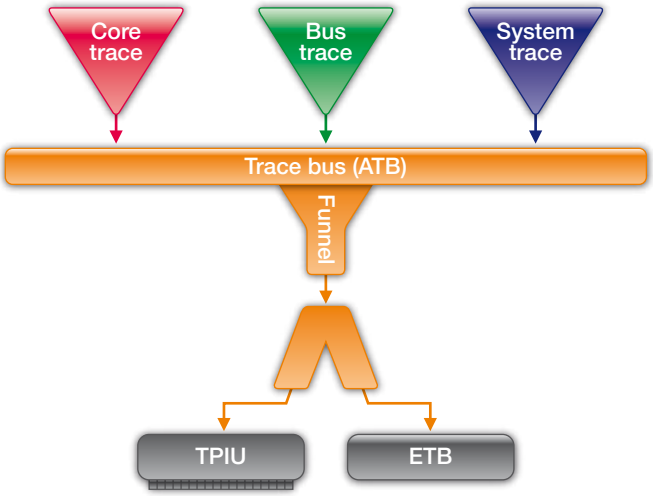


Fig. 11: Les données provenant de différentes macro-cellules de trace sont assemblées en un seul flux grâce à un *Funnel*.

La technologie CoreSight inclut des macro-cellules de trace fournissant des informations sur le fonctionnement interne du SoC. On peut distinguer trois types de technologies :

- **Les macro-cellules de trace associées à un processeur** produisent une trace des instructions exécutées par ce processeur. Selon la configuration interne de la cellule de trace, des informations supplémentaires sur le basculement des tâches ainsi que sur les accès mémoire effectués sont parfois également disponibles.
- **Les macro-cellules de trace assignées à un bus** permettent de visualiser les transferts de données sur un bus interne du CPU.
- **Les macro-cellules de trace système** nécessitent une instrumentation du code et produisent des données de diagnostique ainsi qu'une trace en fonction de déclencheurs matériels (*system event tracing*).

Les données provenant de ces différentes sources de trace sont assemblées en un seul flux de données grâce à un composant Coresight appelé *Funnel* (Figure 9). Ces données sont par la suite soit enregistrées dans une mémoire (ETB), soit exportées à travers un port de trace (TPIU) pour être alors enregistrées par un outil de trace externe. Dans le cas de systèmes multi-cœurs complexes possédant plusieurs macro-cellules de trace, les IPs (*Intellectual Property*) de trace actuellement implémentées par la technologie CoreSight arrivent parfois à leurs limites.

- **ETB:** La mémoire de trace « on-chip » est souvent trop petite pour enregistrer suffisamment de données nécessaires à l'analyse ultérieure. Les tailles typiques de la mémoire ETB varient entre 4 et 16 Ko.
- **TPIU:** Un état système particulier peut entrainer une quantité de trace générée trop importante par rapport à ce que le port de trace permet de transmettre. La technologie CoreSight est conçue de manière à ce que les données de trace soient uniquement extraites des macro-cellules si le TPIU peut les exporter. Si les données générées restent trop longtemps dans la macro-cellule de trace, la FIFO locale sera débordée et des données peuvent être perdues.

La nouvelle technologie de trace de CoreSight, *Trace Memory Controller*, présente des solutions à ces deux scénarios.

ARM CoreSight

ARM fournit avec la technologie CoreSight un vaste ensemble de blocs IP permettant aux concepteurs de SoCs de développer une infrastructure de debugge et de trace sur mesure.

Une seule interface de debugge suffit pour contrôler et coordonner tous les cœurs du SoC et pour accéder à toutes les mémoires du système.

Une interface de trace est aussi suffisante pour fournir des informations sur le fonctionnement interne du SoC et ce sans violation du comportement temps réel du système.

TMC comme *Embedded Trace Buffer*

Afin de pouvoir enregistrer encore plus de données de trace, les fondeurs sont théoriquement capables de

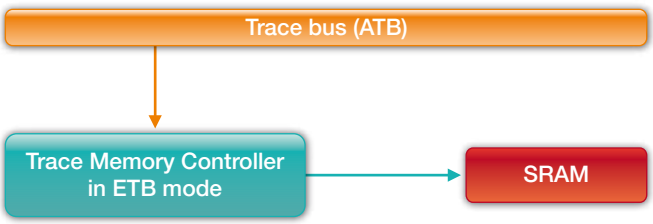


Fig. 12: Le TMC fournit en mode ETB une mémoire de trace sur puce maximale de 4 Go.

connecter une mémoire SRAM au maximum de 4 Go, par le biais du TMC (figure 12).

TMC comme *Embedded Trace FIFO*

L'analyse de la trace nous montre que la bande passante de la plupart des ports de trace est généralement suffisante pour transmettre le flux de données exporté par le TPIU. Une surcharge et donc une perte de données survient uniquement à cause de pic de charge.

Le TMC est alors utilisé comme une FIFO de trace embarquée (*Embedded Trace FIFO: ETF*) permettant de compenser les pics de charge (figure 13). La taille de la ETF - généralement entre 512 octets et 4 Go - peut-être définie pendant la conception du système. Cela permet de supprimer totalement ces pertes de données.

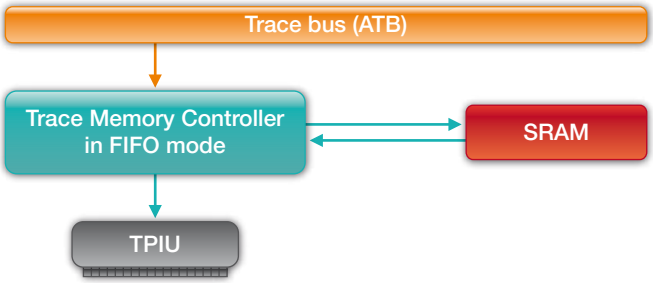


Fig. 13: En mode FIFO, le TMC agit comme une mémoire temporaire pour compenser les pics de charge au niveau du TPIU.

Ces deux intégrations du TMC dans l'infrastructure de trace du SoC sont de simples exemples. Le bloc IP du TMC peut naturellement être intégré dans le système CoreSight de façon plus flexible et plus complexe.

Extensions TRACE32

Lauterbach devait bien sûr adapter le logiciel TRACE32 afin d'inclure la configuration et la manipulation du TMC. Cela est particulièrement valable quand le TMC est intégré dans le SoC en utilisant une nouvelle méthode pas encore incluse dans TRACE32. L'utilisateur du logiciel TRACE32 doit désormais simplement spécifier l'adresse de base du module TMC et peut par la suite utiliser toutes les fonctions de trace habituelles.

TMC: routeur liaison haute vitesse

La possibilité de ne plus utiliser un port de trace dédié fait partie depuis longtemps des discussions au sein de la communauté embarquée. Désormais de bons arguments plaident en sa faveur. Pour la première fois, il est désormais possible de connecter les traces CoreSight directement à une interface standard haute vitesse comme l'USB ou l'Ethernet très souvent disponibles sur les

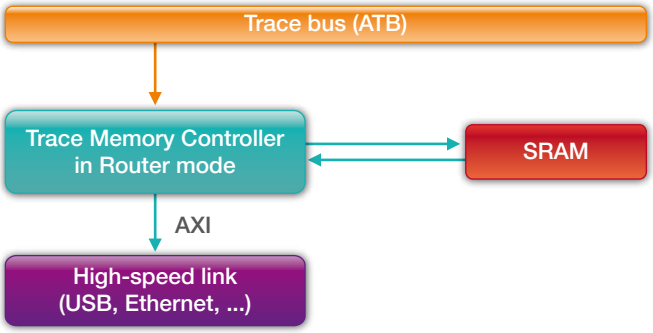


Fig. 14: En mode routeur, le TMC transmet les données de trace à une interface standard haute vitesse pour être exportées.

produits actuels. Idéalement, l'interface utilisée est partagée entre l'outil de trace externe et les autres systèmes connectés.

Le TMC est utilisé ici comme un routeur de trace embarqué (*Embedded Trace Router*). Il a la tâche, au sein du SoC, de transmettre les données de trace à travers un bus AXI vers l'IP de l'interface haute vitesse pour être par la suite exportées (figure 14).

Un tel export de données nécessite des outils de trace complètement nouveaux. Lauterbach est déjà en contact étroit avec plusieurs grands fabricants de semi-conducteurs afin de développer des outils appropriés à cette nouvelle technologie.

Les fonctionnalités CoreSight de TRACE32

- Lauterbach offre un débuggeur pour tous les processeurs intégrables dans le système CoreSight entre autres les processeurs ARM-/Cortex, une multitude de DSPs ainsi que plusieurs noyaux configurables.
- Support AMP (*asymmetric multiprocessing*) et SMP (*symmetric multiprocessing*)
- Debugge JTAG ou en utilisant le « *Serial Wire Debug* » à deux broches.
- Un debugge synchrone de tous les processeurs
- Support de la *Cross Trigger Matrix de CoreSight*
- Support de tous les types de macro-cellules de trace (ETM, PTM, HTM, ITM, STM ...)
- Des outils pour les ports de trace parallèle et série
- Trace des systèmes multi-cœurs
- Analyse de performance détaillée des systèmes multi-cœurs

Cortex-M3/-M4 : Analyse précise de la trace

Le debugge, l'amélioration de la performance ou bien la couverture du code d'un système embarqué peuvent être effectués d'une manière précise et rapide si une analyse optimisée de la trace est disponible. Lauterbach a intégré en 2011 de nouvelles techniques afin d'offrir une analyse optimale de la trace pour les processeurs Cortex-M3/-M4.

Fusion de données ETM et ITM

Pour les processeurs Cortex-M3/-M4, les informations de la trace peuvent être produites à partir de deux sources différentes (voir figure 17).

- **ETMv3:** produit une trace instruction exécutée.
- **ITM:** produit des informations sur les accès mémoire (*Read/Write*) en utilisant l'unité *Data Watchpoint and Trigger Unit* (DWT).

Les paquets de trace provenant de l'ITM et décrivant les accès mémoire contiennent les informations suivantes : L'adresse et la valeur de la donnée et le type de l'accès (lecture/écriture), le compteur programme.

En évaluant le compteur programme, il est possible d'intégrer parfaitement les accès (lecture/écriture) aux données dans la trace instruction, ce qui simplifie considérablement la recherche d'erreur (voir figure 15). Si les accès mémoire sont corrélés à la trace instruction, il est possible par exemple de trouver facilement si des données incorrectes ont été écrites ou pas sur une adresse mémoire donnée.

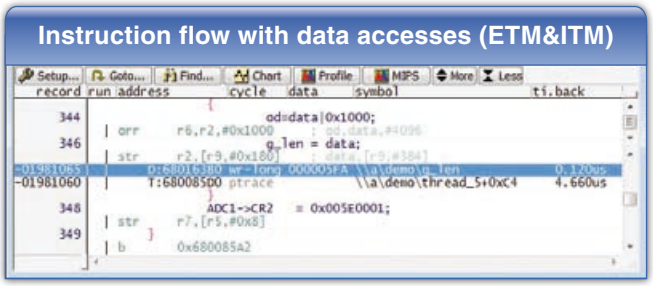


Fig. 15: En fusionnant les données de la ETM et de la ITM, il est possible de parfaitement intégrer les accès mémoire dans la trace d'exécution code.

Trace RTOS sur cibles

Si un système d'exploitation est exécuté sur une cible Cortex-M3/M4, les informations sur le basculement des tâches sont indispensables pour l'analyse de la trace. Il

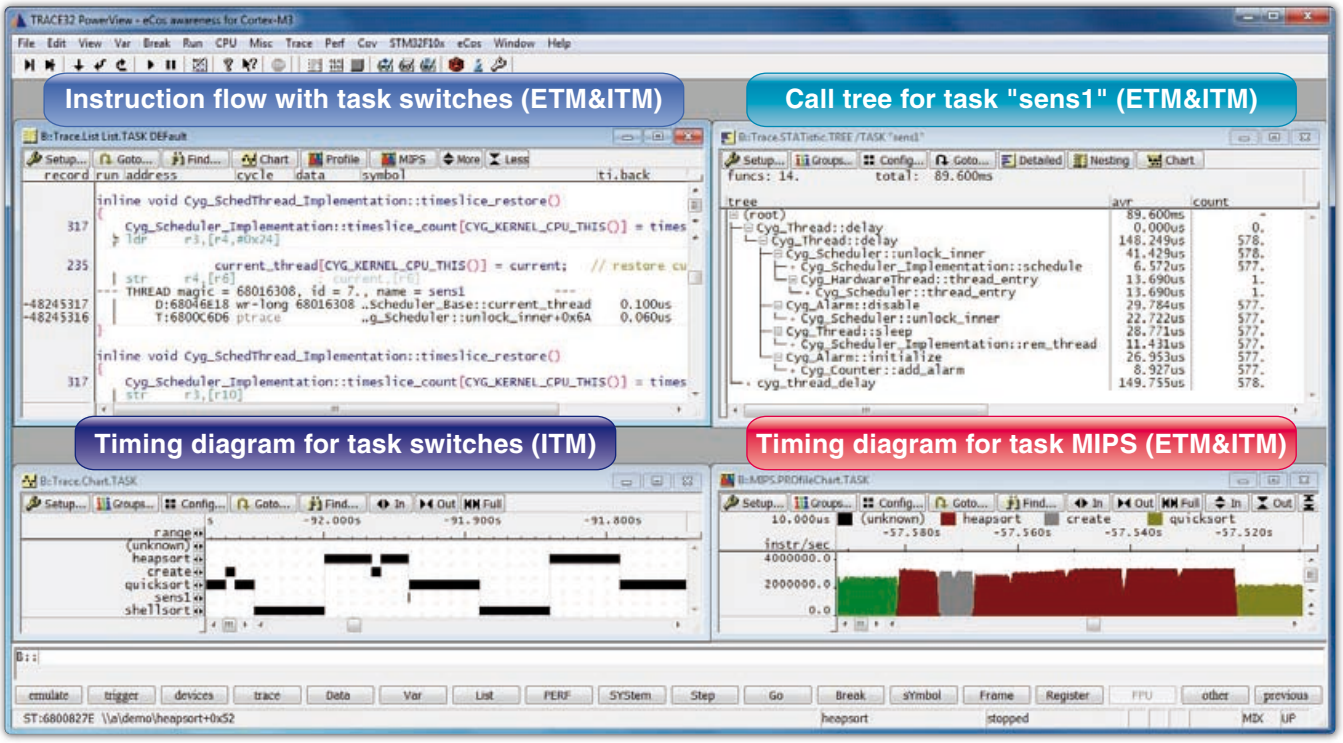


Fig. 16: Évaluation de trace étendue pour le système d'exploitation eCos utilisant les données de l'ETM et de l'ITM.

est possible d'obtenir ces informations par le biais de l'ITM en traçant les accès mémoire en écriture de la variable contenant l'identifiant de la tâche actuelle. De cette manière, on peut intégrer le basculement de tâche dans la trace d'exécution du code. Ceci améliore les fonctions d'affichage de la trace (figure 16). Cette fonctionnalité constitue aussi la base de l'analyse des temps d'exécution (figure 16).

Trois modes d'enregistrement

Lauterbach offre trois modes pour enregistrer les informations de trace produites par les processeurs Cortex-M3/-M4 :

- **Mode FIFO :** Enregistrement des informations dans la mémoire de la sonde CombiProbe (128 Mo)
- **STREAM-Modus :** Transmission « à la volée » des informations vers le disque dur du PC pendant l'exécution CPU
- **Analyse de performance dynamique :** Les informations transmises vers le PC sont directement analysées.

Pour les deux premiers modes d'enregistrement, les informations de trace sont d'abord transférées. L'analyse de ces informations n'est effectuée qu'en fin de transfert.

Chaque mode d'enregistrement a ses avantages. Le mode FIFO est le plus utilisé puisqu'il est rapide et généralement suffisant pour la recherche d'erreur et l'analyse des temps d'exécution.

L'ETMv3 implémentée sur les processeurs Cortex-M3/-M4 ne dispose ni de déclencheur ni de filtre de trace. Il n'est donc pas possible de configurer la source de trace afin de tracer uniquement une partie du code.
Par ailleurs, il est souvent nécessaire d'enregistrer les informations de trace pendant des durées relativement longues. Dans ce cas, le mode STREAM est le meilleur choix. Toutefois, ce mode soumet l'environnement de debugge à des exigences plus importantes :

- La grande quantité de données générée par ce mode requiert une exécutable TRACE32 64-bits.
- La vitesse de transfert entre la sonde CombiProbe et le PC doit être suffisante pour un transfert sans perte de données. La mémoire de la sonde CombiProbe (128 Mo) amortit les pics de charge du TPIU.

L'analyse de performance dynamique est particulièrement appropriée pour la couverture de code. En effet, il est possible de suivre la couverture de code en temps réel, les résultats étant visibles à la volée. Pour les lignes de code qui n'ont pas été complètement exécutées (en jaune sur la figure 17), le système embarqué peut être stimulé jusqu'à ce que ces lignes obtiennent la marque « ok ».

Fig. 17: Le profilage temps-réel permet de suivre l'analyse de couverture de code en direct sur l'écran.

Une simulation très proche de la réalité

Il est courant de nos jours d'effectuer des simulations et des vérifications avant l'arrivée du prototype. C'est pourquoi les logiciels de développement MATLAB® et Simulink® font partie intégrante des solutions de développement. On peut économiser beaucoup de temps et d'effort si les boucles de contrôle sont testées par la simulation. Mais une fois l'algorithme de contrôle optimal déterminé, comment l'appliquer sur le matériel ?

Pour cela, le logiciel Simulink offre la possibilité de générer automatiquement le code. Mais peut-on être certain que le code créé, se comportera de la même façon sur le matériel que lors de la simulation ?

Démarche de vérification

Au sein de l'université technique de Munich, au département des systèmes de vol dynamique, une solution intéressante a été présentée au cours du développement d'un système de contrôle de vol pour un Diamond DA42 (voir figure 20).

Après la création des algorithmes de contrôle et la validation des tests fonctionnels avec Simulink, le code source embarqué a été généré à partir des blocs de contrôle grâce a l'« *Embedded Coder* ». Le code généré a été par la suite chargé et fonctionnellement testé sur le processeur en utilisant un débuggeur TRACE32.

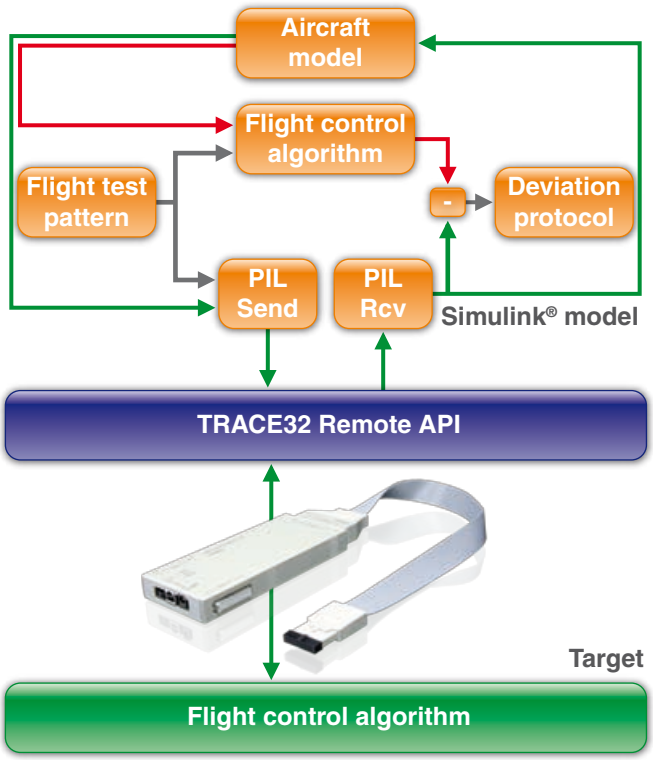


Fig. 18: Équilibrage du cas réel (en vert) et de la simulation (en rouge).

Afin de déterminer l'écart entre la simulation (en rouge) et le cas réel (en vert), mais surtout afin de confirmer la précision numérique du matériel de contrôle, une simulation *Processor-In-the-Loop* (PIL) a été choisie (voir figure 18). La simulation PIL se base essentiellement sur les blocs Simulink « *PIL Send* » et « *PIL Receive* » spécialement développés lors de ce projet. Ces blocs étaient conçus pour assurer la communication entre Simulink et l'interface de programmation de TRACE32 « *TRACE32 Remote API* ».

Sur le matériel cible, l'algorithme de contrôle de vol exécute chaque itération par étape de calcul. Les paramètres d'entrée nécessaires sont fournis par le modèle Simulink. Les valeurs de réglage calculées sont retournées au modèle Simulink et alimentent le modèle.

En parallèle, l'algorithme de contrôle de vol analyse ces mêmes valeurs de réglage. La différence entre les deux systèmes nous permet de comparer les résultats.
Les tests ont donné une différence absolue de 10^{-13}. En utilisant cette approche, la concordance a été montrée de façon simple et élégante.

Pour plus d'information concernant ce projet, veuillez consulter l'adresse suivante :
www.lauterbach.com/intsimulink.html

TRACE32: Intégration Simulink®

Lauterbach présentera plusieurs nouveautés - aussi une intégration plus complète entre Simulink et le débuggeur TRACE32 - du 03 au 05 Avril au salon RTS Embedded Systems, Paris, Porte de Versailles.

Pour cette intégration, TRACE32 utilise la propriété suivante de la génération de code : pour chaque bloc Simulink, le code généré commence par une ligne de commentaire qui contient le nom et le chemin du modèle. Ces lignes de commentaires sont disponibles dans TRACE32, après le chargement du code généré. Elles permettent de déterminer de manière simple les lignes du code source correspondantes à chaque bloc Simulink.

Navigation de Simulink® vers TRACE32

Plusieurs menus TRACE32 sont intégrés dans Simulink (*Simulink Customizations Menus*). Ces menus peuvent

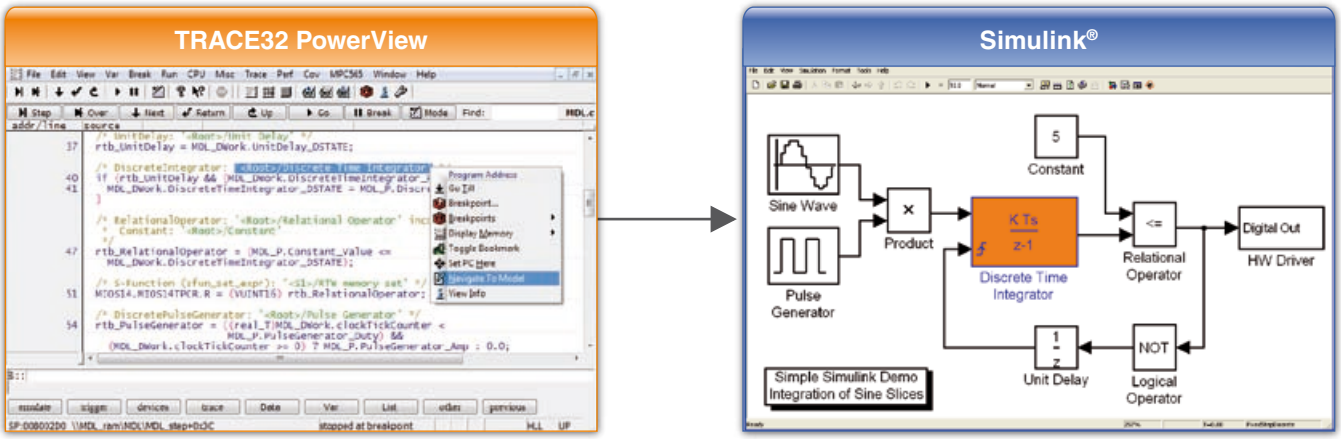


Fig. 19: Marquage du bloc Simulink pour une ligne de code source sélectionnée.

être utilisés pour contrôler le débuggeur TRACE32 à partir de Simulink.

Les fonctions suivantes sont disponibles depuis Simulink :

- Affichage du bloc de code sur TRACE32
- Affichage des signaux sur TRACE32 en utilisant les fenêtres « *Watch* »
- Chargement du build Simulink dans le débuggeur TRACE32
- Placer des points d'arrêt de bloc et de programme à partir de l'interface Simulink
- Placer et gérer les points d'arrêt de signaux et de données
- Démarrer et arrêter l'exécution du programme sur la cible

Navigation de TRACE32 vers Simulink®

A partir du débuggeur TRACE32, il est possible de marquer un bloc Simulink correspondant au code source sélectionné (figure 19).

Perspectives

La version 2012a de Simulink permettra à Lauterbach d'utiliser les améliorations de l'interface de programmation *rtiostream API* de Simulink afin d'intégrer une simulation PIL, Data Logging ainsi que le réglage des paramètres.

MATLAB® et Simulink® sont des marques déposées de The MathWorks, Inc.

Fig. 20: Avion de test du type Diamond DA42 (source: www.diamond-air.at).

Debugge du BIOS UEFI avec TRACE32

Une nouvelle extension TRACE32 pour les processeurs Atom™ permet d'effectuer un debugge détaillé du BIOS UEFI d'Insyde's H2O.

UEFI est le successeur du BIOS traditionnel utilisé par les PCs. Il agit comme une interface entre le firmware et le système d'exploitation gérant le processus de démarrage, de la mise sous tension jusqu'à la prise du contrôle par le système d'exploitation. L'UEFI effectue différentes tâches bien précises (figure 21).

Grâce au JTAG, TRACE32 permet de debugger le système cible à partir du vecteur de reset. Pour chaque phase du processus de Boot, TRACE32 permet à l'utilisateur d'afficher les informations spécifiques à l'UEFI.
TRACE32 met également à la disposition de l'utilisateur une série de scriptes et de fonctions permettant un debugge dynamique des pilotes chargés et cela dès leur première instruction. Pour plus de détails concernant l'extension UEFI :
www.lauterbach.com/uefi.html.

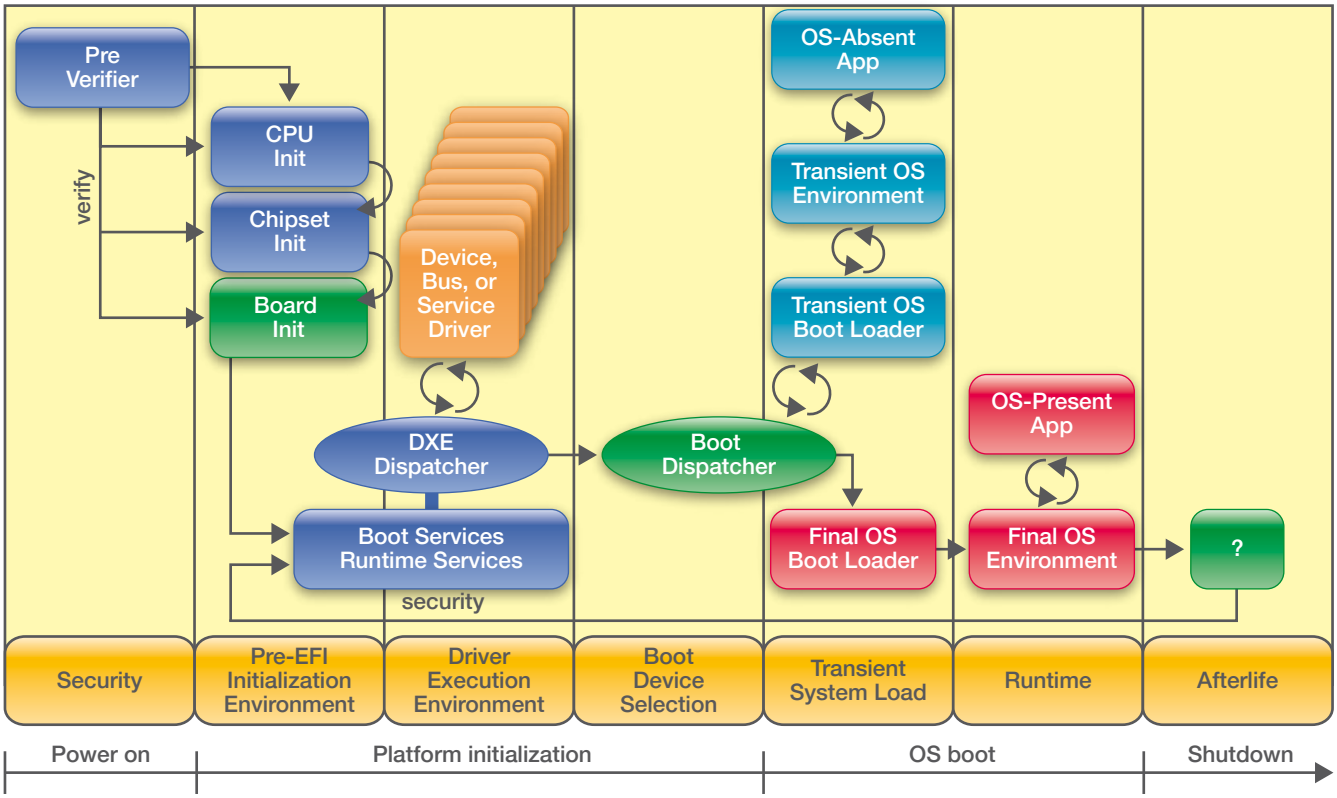


Fig. 21: Différentes phases du démarrage avec le système UEFI.

FILIALES À TRAVERS LE MONDE

- France
- Allemagne
- Grande-Bretagne
- Italie
- États-Unis
- Chine
- Japon

Et représentés par nos distributeurs dans tous les autres pays.

NEWSLETTER

Si vous souhaitez dorénavant recevoir notre Newsletter par voie postale, veuillez nous communiquer vos coordonnées à l'adresse suivante :
info_fr@lauterbach.com