Comment exécuter des applications temps réel sur Linux ?

Le déterminisme est une exigence cruciale pour les systèmes temps réel. Alors qu'il existe plusieurs processeurs sur le marché qui peuvent soit faire tourner Linux, soit exécuter du code de façon déterministe, il n'en existe quasiment aucun qui peut offrir ces deux fonctionnalités simultanément. Avec la puce-système PolarFire et des cœurs RISC-V, les applications Linux et temps réel dur peuvent coexister de manière cohérente en tirant profit d'un sous-système mémoire flexible unique. Explications de Microchip.

AUTEUR

Tim Morin, directeur du marketing stratégique, Microchip Technology.

Si le terme « système temps réel » peut paraître intriguant, il définit en fait tout simplement un système qui exécute les commandes de façon déterministe sur une base périodique. Le déterminisme est un prérequis majeur pour les systèmes temps réel car ces derniers contrôlent généralement des machines. Vous ne voudriez pas que le foret d'un tour à commande numérique se déplace d'un point A à un point B en 10 millisecondes (ms) le mardi, et qu'il réalise la même opération en 20 ms le mercredi. De la même manière, un système de commande de vol doit répondre exactement de la même façon aux ordres d'un pilote d'avion, à tout moment, et ce quelles que soient les conditions.

En conséquence, qu'est-ce qu'un système déterministe ? La figure 1 illustre le concept. Le code dont l'exécution est critique au niveau temporel est géré par la routine de service d'interruption. Les interruptions interviennent de manière périodique et le temps d'exécution de ce code est déterministe. Si ce n'est pas le cas, le résultat est visible sur la figure 2 avec des mises à jour générées par le matériel qui arrivent de manière aléatoire.

Parallèlement il existe un besoin d'apporter la richesse de l'environnement Linux et de l'écosystème de middleware associé aux systèmes contrôlés de façon matérielle. Dans l'absolu, Linux requiert une unité de gestion de la mémoire MMU (Memory Management Unit) qui sert à virtualiser la mémoire physique pour le développeur d'applications. Les processeurs qui embarquent une unité MMU intègrent également au moins une mémoire cache de niveau L1, et dans la plupart des cas un cache L2. Les caches et le déterminisme sont orthogonaux l'un par rapport à l'autre si l'on peut dire, comme on peut le voir sur la figure 3. On peut voir que les défauts des caches L1 ou L2 entraînent une gigue dans l'exécution, ce qui génère un retard au niveau du pipeline d'exécution pendant que les lignes de cache se remplissent. Les caches de plus grande capacité peuvent certes réduire la fréquence des défauts de cache mais ils ne les suppriment pas complètement.

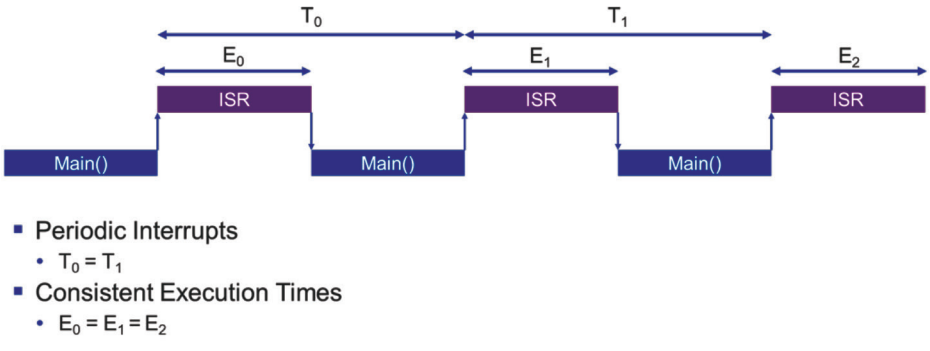
Sur certains processeurs qui peuvent faire tourner Linux, la gigue d'exécution est également affectée par le prédicteur de branchement qui sert à augmenter les performances au niveau de l'application. Mais il existe des situations où les prédictions de branchement s'avèrent fausses, ce qui conduit à une vidange du pipeline et donc à un comportement d'exécution non déterministe. Les tables d'historique de branchement qui sont utilisés par les prédicteurs durant une routine de service d'interruption (ISR) sont par ailleurs liées à l'historique d'exécution du code principal de l'application, et non à l'historique d'exécution de l'ISR. Dès lors le temps d'exécution varie d'une ISR à l'autre, principalement en raison de la vidange du pipeline au sein de l'ISR. Une manière de contrer ce phénomène est de permettre aux utilisateurs de désactiver le prédicteur de branchement. Le développeur d'applications peut ainsi contrôler le moment et la manière dont est appliqué le déterminisme dans le système. Pour mettre en œuvre largement le déterminisme dans toute l'application, il est possible de désactiver complètement les prédicteurs de branchement, avec en corollaire une diminution des performances.

L'architecture RISC-V du SoC FPGA PolarFire

Sur le marché, il existe différents types de processeurs. Certains peuvent faire tourner Linux mais ils n'ont pas été conçus pour exécuter du code de façon déterministe. A l'inverse, d'autres sont parfaitement capables d'exécuter du code de

1 EXEMPLE D'UN ENVIRONNEMENT D'EXÉCUTION DÉTERMINISTE

Le code dont l'exécution est critique au niveau temporel est géré par la routine de service d'interruption (ISR). Les interruptions interviennent de manière périodique et le temps d'exécution de ce code est déterministe.



façon déterministe mais ne peuvent pas faire tourner Linux. Ne serait-il pas intéressant pourtant de disposer, dans votre boîte à outils embarquée, d'une architecture qui puisse faire les deux à la fois ? L'architecture FPGA pour puce-système à cœur de processeur RISC-V récemment dévoilée par Microchip pour le SoC PolarFire en est justement capable.

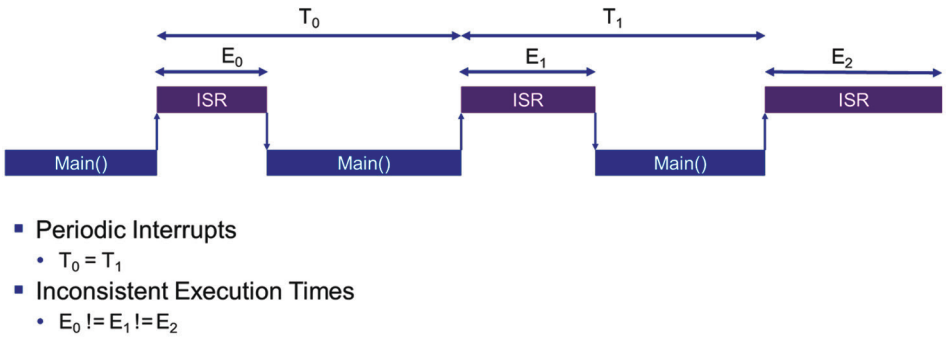
La figure 4 présente quatre cœurs de processeur 64 bits RISC-V RV64GC capables de faire tourner Linux, et un cœur (RV64IMAC) qui n'en est pas capable, du simple fait qu'il ne dispose pas de MMU. Il existe d'autres différences dans les jeux d'instructions du RV64IMAC et du RV64GC. Le RV64GC comprend une unité de calcul à virgule flottante en double précision, ce qui n'est pas le cas du RV64IMAC. Pour augmenter le niveau de déterminisme au sein de l'architecture, l'utilisateur peut désactiver le prédicteur de branchement sur n'importe quel cœur, soit avant la mise sous tension, soit pendant une ISR. Le déterminisme peut aussi être amélioré en optant pour des pipelines d'instructions « dans l'ordre » (in-order) en lieu et place de pipelines d'instructions « dans le désordre » (out-of-order). Un bénéfice supplémentaire des pipelines in-order est leur immunité aux cyberattaques Spectre et Meltdown.

Le sous-système mémoire du SoC PolarFire

Jusqu'à présent, nous n'avons parlé que du déterminisme en lien avec les cœurs de processeur. Au-delà du déterminisme, l'autre aspect important à évoquer est le sous-système mémoire dans les SoC PolarFire, là où est concrètement exécuté le code. Tout d'abord, la totalité de l'espace mémoire dans un SoC PolarFire est cohérente. La cohérence se définit comme suit. Toute mémoire qui possède des données dupliquées plusieurs fois est gérée par un gestionnaire de cohérence, et toute mémoire qui contient uniquement une copie unique de données est de par sa nature cohérente, puisqu'aucune autre copie n'existe dans la hiérarchie mémoire. Le SoC PolarFire possède trois sous-systèmes mémoire : L1, L2 et L3. Le sous-système mémoire L3 intègre un contrôleur 36 bits durci LPDDR3/LPDDR4 et DDR3/DDR4. Les 4 bits supplémentaires servent à ajouter le contrôle SECDED au sous-système mémoire L3 externe.

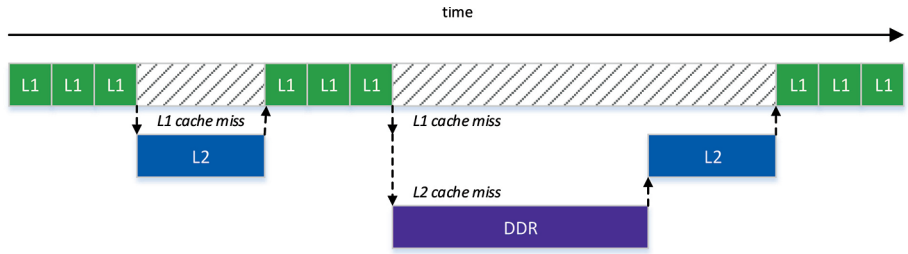
2 EXEMPLE D'UN ENVIRONNEMENT D'EXÉCUTION NON DÉTERMINISTE

Ici, à la différence de la situation de la figure 1, les durées d'exécution des routines de service d'interruption sont variables.



3 PROBLÈMES DE DÉTERMISME DANS LES SYSTÈMES À MMU AVEC DEUX NIVEAUX DE CACHE

Les défauts des caches L1 ou L2 entraînent une gigue dans l'exécution, ce qui génère un retard au niveau du pipeline d'exécution pendant que les lignes de cache se remplissent, affectant ainsi le déterminisme.

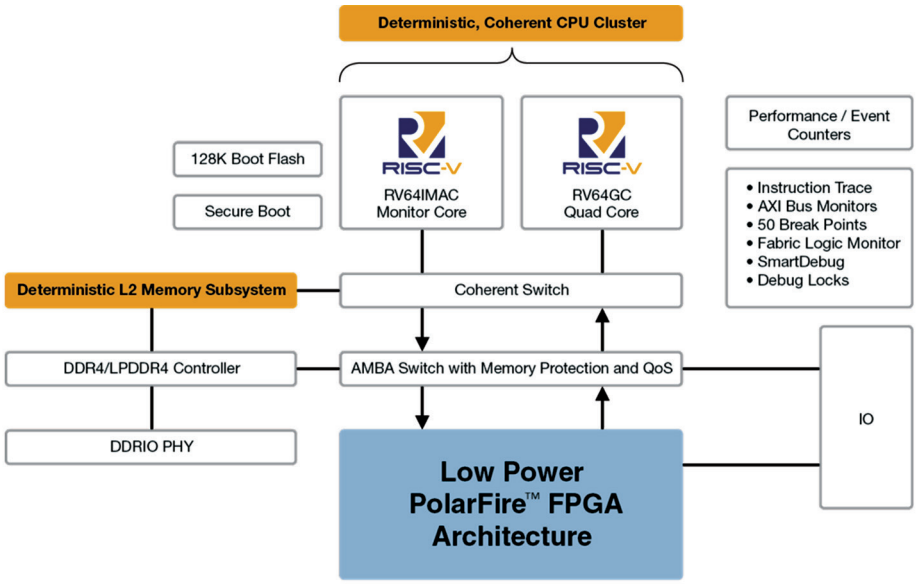


Le sous-système mémoire L1

Les quatre cœurs d'application RV64GC possèdent une mémoire cache associative par ensembles à 8 voies pour les instructions (32 Ko I$TIM) et une mémoire cache associative par ensembles à 8 voies pour les données (32 Ko D$TIM). I$ équivaut à un cache d'instructions et TIM signifie « Tightly Integrated Memory » (mémoire étroitement intégrée). Les I$TIM et D$TIM peuvent être confi-

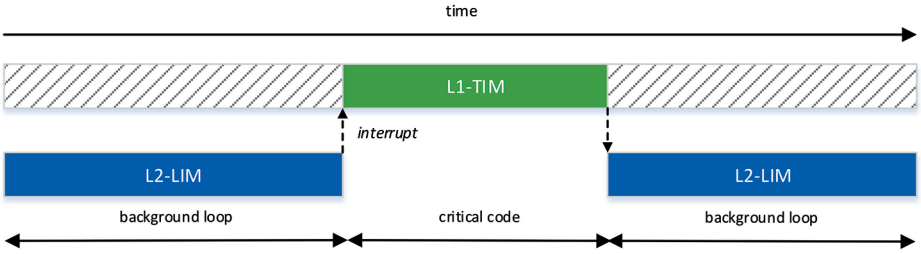
4 ARCHITECTURE DE LA PUCE-SYSTÈME SOC POLARFIRE AVEC GRAPPE DE CŒURS RISC-V

Le SoC PolarFire présente ici quatre cœurs de processeur 64 bits RISC-V RV64GC capables de faire tourner Linux, et un cœur (RV64IMAC) qui n'en est pas capable, du simple fait qu'il ne dispose pas de MMU.



5 EXÉCUTION DÉTERMINISTE AVEC MÉMOIRES LIM ET TIM

Cette figure illustre un système déterministe lorsque le sous-système mémoire L2 est configuré comme mémoire LIM (Loosely Integrated Memory) et que les sous-systèmes mémoire L1 sont configurés comme mémoires TIM (Tightly Integrated Memory).

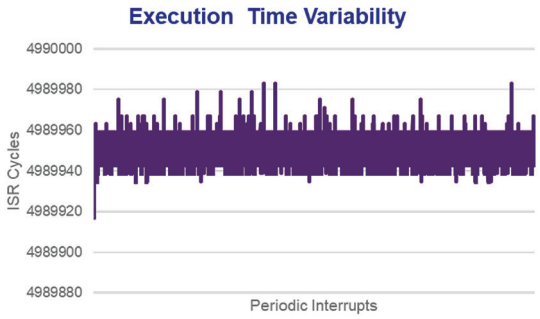


gurés par l'utilisateur à la condition qu'il y ait toujours une voie de cache pour I$TIM et D$TIM.

Le cœur de contrôle du RV64IMAC possède une mémoire cache associative par ensembles à deux voies de 16 Ko pour les instructions (I$TIM) et de 8 Ko pour les données (DTIM). La DTIM est une mémoire bloc-notes de données à partir de laquelle le code peut être exécuté. Toutes les mémoires TIM L1 offrent un accès déterministe à faible latence et possèdent une fonction de correction des erreurs sur un seul bit et détection des erreurs sur deux bits (SECDED).

6 EFFETS DU PRÉDICTEUR DE BRANCHEMENT SUR LE DÉTERMINISME

Cette figure montre une application en cours d'exécution lorsque les mémoires L1 et L2 sont respectivement configurées comme mémoires TIM et LIM. L'axe horizontal indique les interruptions et l'axe vertical indique la durée du cycle dans une ISR. Comme on peut le voir, la durée d'exécution des ISR varie dans le temps.



Le sous-système mémoire L2

Le sous-système mémoire L2 d'une capacité de 2 Mo possède une fonction SECDED et peut être configuré selon trois modes différents : une mémoire cache associative par ensembles à 16 voies, une mémoire faiblement intégrée (LIM, Loosely Integrated Memory) ou une mémoire bloc-notes. La mémoire LIM peut être attachée à un processeur particulier et peut être divisée en plusieurs voies de cache ; en d'autres termes, les LIM peuvent être constituées de plusieurs morceaux (voies) de 128 Ko et affectées en accès exclusif à un processeur. Configuré en tant que LIM, le sous-système de mémoire L2 fournit un accès déterministe au cœur auquel il est attaché et est cohérent, puisqu'aucune autre copie n'est partagée avec les sous-systèmes de mémoire L1 et L3. La mémoire LIM est idéale pour l'exécution de code déterministe dans l'application principale comme dans les ISR. La figure 5 illustre un système déterministe lorsque le sous-système mémoire L2 est configuré comme mémoire LIM et que les sous-systèmes mémoire L1 sont configurés comme mémoires TIM.

Malheureusement, à cause des erreurs de prédiction des prédicteurs de branchement, la variabilité du temps d'exécution des routines ISR existe toujours, y compris si la mémoire L2 est configurée comme mémoire LIM. La figure 6 montre une application en cours d'exécution lorsque les mémoires L1 et L2 sont respectivement configurées comme mémoires TIM et LIM. L'axe horizontal indique les interruptions et l'axe vertical indique la durée du cycle dans une ISR. Comme on peut le voir, la durée d'exécution des ISR varie dans le temps. Le déterminisme que l'on recherche peut être trouvé en désactivant les prédicteurs de branchement.

Tout comme la mémoire LIM, la mémoire bloc-notes peut être configurée en morceaux (voies) de 128 ko et affectée à des cœurs de processeur. La mémoire bloc-notes est idéale comme ressource de mémoire partagée entre le processeur exécutant le code à partir de la LIM d'une part, et les processeurs exécutant le code à partir des sous-systèmes mémoire L1/L2 et L3 (typiquement Linux). Si l'application du RV64IMAC écrit des données dans la mémoire bloc-notes, et que le contenu de cet emplacement de mémoire est dupliqué ailleurs dans le sous-système mémoire L1, alors le gestionnaire de cohérence garantira la cohérence. De cette façon, une application temps réel peut partager des données de façon cohérente avec une application fonctionnant dans un espace utilisateur ou sur Linux.

La figure 7 est l'une des configurations possibles du sous-système microprocesseur du SoC PolarFire. Dans cette configuration, le RV64IMAC gère la fonction temps réel (s'exécutant à partir de la mémoire LIM) tandis que les RV64GC font tourner Linux. Si votre fonction temps réel nécessite des calculs à virgule flottante, le RV64GC peut être utilisé à cet effet car les prédicteurs de branchement peuvent être désactivés, et le sous-système mémoire L1 peut être configuré comme une mémoire TIM. ■

7 ARCHITECTURE ASSURANT UN PASSAGE DE MESSAGE COHÉRENT

Voilà l'une des configurations possibles du sous-système microprocesseur du SoC PolarFire. Dans cette configuration, le RV64IMAC gère la fonction temps réel (s'exécutant à partir de la mémoire LIM) tandis que les cœurs RV64GC font tourner Linux.