Chapter 1

Introduction

Part I
Vue d'Ensemble de l'Architecture du Processeur Cell

Le processeur Cell est la première implémentation de l'architecture *Cell Broadband Engine* (CBEA), qui est entièrement compatible avec l'architecture *PowerPC* 64-bit. Ce processeur à été initialement conçu pour la console de jeux *PlayStation 3* mais ses performances hors normes ont trés vite fait de lui un bon candidat pour d'autres domaines d'applications qui requièrent une grande puissance de calcul, comme le traitement du signal et des images.

Le processeur Cell est une machine multi-coeur hétérogène, capable d'effectuer une quantité de calcul en virgule flottante considérable, sur des données occupant une large bande-passante. Il est composé d'un processeur 64-bit appelé *Power Processor Element* (PPE), huit coprocesseurs spécialisés appelés *Synergistic Processor Element* (SPE), un contrôleur mémoire haute-vitesse et une interface de bus à large bande-passante. Le tout intégré sur une seule et même puce.

Le PPE et les SPEs communiquent au travers d'un bus interne trés rapide appelé *Element Interconnect Bus* (EIB). Si l'on considère une fréquence d'horloge de 3.2 Ghz, le processeur Cell peut atteindre théoriquement une performance crête de 204.8 Gflop/s en simple-précision et 14.6 Gflop/s en double-précision.

Le bus interne supporte une bande passante qui peut aller jusqu'à 204.8 Gbytes/s pour les transferts internes à la puce (impliquant le PPE, les SPEs, la mémoire et les contrôleurs I/O). le contrôleur mémoire: *Memory Interface Controller* (MIC) fournit une bande-passante de 25.6 Gbytes/s vers la mémoire principale. Le contrôleur I/O quand à lui fournit 25 Gbytes/s en entrée et 35 Gbytes/s en sortie.

Le rôle du PPE est celui d'un chef d'orchestre, il prend en charge l'OS (*Operating System*) et coordonne les SPEs. Au niveau de l'architetcure c'est un *PowerPC* 64-bit classique avec une extension SIMD (VMX), un cache L1 de 32 KB (données et instructions) et un cache L2 de 512 KB. C'est un processeur à exécution dans l'ordre (in-order execution processor), il suporte le dual-issue (parallélisme d'instuctions) ainsi que le multi-threading d'ordre 2 (parallélisme de tâches).

Chaque SPE est composé de d'une SPU (*Synergistic Processesor Unit*) et d'un MFC (*Memory Flow Controller*). Le MFC contient à son tour un contrôleur DMA, une unité de gestion de la

mémoire (MMU), une unité interface de bus, et une unité atomique pour la synchronisation avec les autres SPEs et le PPE. Le SPU est un processeur de type RISC avec un jeu d'instructions et une microarchitecture conçus pour les applications flot de données haute-performance ou de calcul intensif. Le SPU inclut une mémoire locale de 256 KB qui contient les données et les instructions. Le SPU ne peut pas accéder directement à la mémoire principale mais par le biais de commandes DMA via le MFC qui permettent de lire et d'écrire dans la mémoire principale. Les deux unités MFC et SPU sont indépendantes ce qui permet l'exécution des tâches de calcul et de transferts en parallèle sur le SPE.

Il n'existe pas de mécanisme hardware tel que la mémoire cache pour la gestion des mémoires locales, et celles-ci doivent être gérées par le software. Le MFC effectue des commandes DMA pour transférer entre la mémoire centrale et les mémoires locales. Les instructions DMA pointent des emplacements de la mémoire centrales en utilisant des adresses virtuelles compatibles *PowerPC*. Les commandes DMA peuvent transférer des données à partir de n'importe quel emplacement lié au bus d'interconnexion (mémoire principale, la mémoire locale d'un autre SPE, ou un périphérique I/O). Des transferts SPE vers SPE en parallèle sont faisables à raison de 16 bytes par cycle d'horloge SPE, tandis que la bande-passante de la mémoire centrale est de 25.6 Gbytes/s pour le processeur entier.

Chaque SPU contient 128 registres SIMD de taille 128-bits. Cette quantité importante de registres facilite l'ordonnancement efficace des instructions ainsi que d'autres optimisations comme le déroulage de boucle (loop-unrolling).

Toutes les instructions SIMD sont des instructions que le pipeline peut exécuter à 4 granularités: 16x des entiers 8-bit, 8x des entiers 16-bit, 4x des entiers 32-bits ou des flottants simple-précision. Le processeur SPU est un processeur à exécution dans l'ordre (in-order-execution processor), il possède deux pipelines d'instructions connus sous les dénominations pair (*even*) et impair (*odd*).

Les instructions flottantes et entières sont dans le pipeline *even* alors que le reste est dans le pipeline *odd*. Chaque SPU peut lancer et compléter jusqu'à deux instructions par cycle, une par pipeline. Cette limite théorique peut être atteinte pour un large eventail d'applications. Toutes les instructions flottantes double-précision peuvent être lancées en un cycle d'horloge du processeur. Par contre les instructions flottantes double-précision ne sont pipelinées que partiellement, il en résulte un débit d'exécution moindre (deux instructions double-précision tous les 7 cycles d'horloge SPU).

Si l'on prend une instruction de multiplication accumulation en flottant simple-précision (qui comptent pour deux opérations) les 8 SPEs peuvent exécuter un total de 64 opérations par cycle.

INSERER FIGURE DU CELL

Le PPE: Power Processor Element

Le PPE est un processeur 64-bit compatible avec l'architecture Power, optimis au niveau de l'efficacit énergétique. La profondeur de pipeline du PPE est de 23 tages, chiffres qui peut paratre faible par rapport au prcdentes architectures surtout quand on sait que la dure de l'tage t rduite d'un facteur 2. Le PPE est une architecture dual-issue (deux instructions peuvent tre lances par cycle) qui ne reordonance pas dynamiquement les instructions l'excution (excution dans l'ordre). Le processeur entrelace des instructions provenant de deux threads de calcul diffrents pour optimiser l'utilisation de la fentre d'excution. Les instructions arithmtiques simples s'excutent et fournissent leur rsultat en deux cycles. Les instructions de chargements (loads) s'excutent galement en deux cycles. Une instruction flottante en double prcision s'excute en 10 cycles. Le PPE supporte une hirarchie conventionnelle de caches avec un cache L1 (de niveau 1) donnes en instructions de 32-KB, et un cache L2 de 512-KB.
Le processeur fournit deux threads d'excution simultans et peut tre vu comme un processeur double-coeur avec un flot de donnes partag, ceci donne l'impression au logiciel d'avoir deux units de traitement distinctes. Certains registres sont dupliqus mais pas les caches qui sont partags par les deux threads.
Le processeur est compos de trois units l'unit d'instructions (UI) responsable du chargement, dcodage, branchements, excution et compltion des instructions. Une unit d'excution des oprations en arithmtique point-fixe (XU) qui est galement responsable des instructions load/store. Et enfin l'unit VSU qui excute les instructions en virgule flottante ainsi que les instructions vectorielles. Les instructions SIMD dans le PPE sont celle des anciennes gnrations de PowerPC 970 et effectuent des oprations sur des registres 128-bit de donnes qui donnent un paralllisme de 2, 4, 8 ou 16, selon le type de donnes considr.

Les SPE (Synergistic Processing Element)

Le SPE contient un jeux d'instructions nouveau mais qui n'est autre qu'une version réduite du jeux d'instructions SIMD VMX (Altivec), mais qui est optimise au niveau de consommation d'nergie et des performances pour les applications de calcul intensif et de multimdia. Le SPE contient une mmoire locale de 256 KB (scrathpad) qui est une mmoire de donnes et d'instructions. Les donnes et les instructions sont transfres de la mmoire centrale vers cette mmoire prive au travers de commande DMA synchrones et cohrentes qui sont excuts par le MFC (Memory Flow

Controler) qui est prsent dans chaque SPE. Chaque SPE peut supporter jusqu' 16 commandes DMA en suspens. L'unit DMA peut tre programme de trois manires diffrente: 1) avec des instructions sur le SPE qui insrent des commandes DMA dans la file d'attente; 2) Par la programmation de transferts permettant de faire des accs sur des zones non contigus de la mmoire au travers d'une liste de DMA; 3) Par l'insertion d'une commande DMA dans la file d'attente d'un autre processeur par les commandes de DMA-write. Afin de faciliter la programmation et de permettre des transferts entre SPEs les mmoire locales sont mappes en mmoire centrale. La prsence des mmoires locales introduit un autre niveau dans la hirarchie mmoire au dessus des registres. Les temps d'accs de ces mmoires sont de l'ordre du cycle ce qui en fait de bon candidats pour rduire la latence d'accs la mmoire centrale qui est de l'ordre de 1000 cycles, d'autant plus que le fait que le contrleur DMA soit indpendant de l'unit donne un niveau de paralllisme supplmentaire. La prsence de ces mmoires prives permet diffrents modles de programmation qui peuvent tre appliqus au processeur Cell.

Le Bus Interne (Element Iterconnect Bus)

Le bus interne du processeur permet de relier les units de traitement PPE, SPE la fois entre elles, la mmoire centrale ainsi qu' une sortie externe. Le bus contient des chemins de donnes diffrents de ceux des requtes. Les lments autour du bus sont connects par des liaison point–point et un arbitre de bus est responsable de la rception des commandes et de leur diffusion vers les units. Le bus est constitu de 4 anneaux d'une largeur de 16-octets deux fonctionnent dans le sens d'une aiguille d'une montre et les deux autres dans le sens inverse. Chaque anneau peux potentiellement grer 3 transferts en parallle si toutefois leurs chemins ne se croisent pas. Le EIB opre une frquence qui est la moiti de celle du processeur, chaque unit du bus peut simultanment envoyer et recevoir 16 octets par cycle d'horloge du bus.

Part II
Modèles de Programmation pour le Processeur Cell

1.1 RapidMind

RapidMind[citer paper Rapidmind] est un modèle de programmation du processeur Cell, il relève du modèle de programmation *stream programming* et s'apparente à un langage de programmatio enfoui dans C++. Il est basé sur la bibliothèque template C++ et une librarie de runtime qui effectué la génération dynamique de code. La librairie tepmplate permet l'invocation permet l'invocation de code SPE à l'intrieur du code PPE, avec l'ensemble du code SPE écrit en template.
La librairie template de **RapidMind** fournit un ensemble de types de données, des macros de contrôle, des opérations de réduction et des fonctions communes qui permettent à la librairie de runtime de capturer une représentation du code SPE (retained code). Les types de données ont été spécialement conçus pour exprimer de manière simple les opérations SIMD et les exposer facilement à la librairie runtime. Le runtime à son tour extraits le parallélisme à partir de ces opérations en vectorisant le code et en divisant les calculs sur les tableaux et les vecteurs sur les différents SPEs. Il peut également effectuer des optimisations de boucle comme les détéction des invariants de boucle. **RapidMind** assigne des tâches aux SPEs dynamiquementet peut effectuer des optimisations à plus haut niveau comme la superposition des calculs et des transferts qui permet de masquer la latence de ces derniers. Enfin le modèle de calcul est un modèle SPMD, il diffère du modèle SIMD du fait que les programme peuvent contenir du flot de contrôle et que ce modèle puisse gérer une certaine forme de parallélisme de tâches même si étant initialement un modèle data-parallel.

1.1.1 Modèle de Programmation et Interface

L'interface est basée sur trois types C++ principaux: **`Value<N,T>`**, **`Array<D,T>`** et **`Program`**, tous sont des conteneurs, les deux premiers pour les données et le dernier pour les opérations. Le calcul parallèle est efféctué soit en applicant des **`Program`** sur des **`Array`** pour créer de nouveaux **`Array`**, ou en applicant une opération collective parallèle qui peut être paramétrée par un objet **`Program`** comme la réduction par exemple.
A première vue les types **`Value`** et **`Array`** ne sont pas une grande nouveauté. En effet, tout développeur C++ a pour habitude d'utiliser les types N-tuples pour exprimer le calcul numérique sur des vecteurs, et le type **`Array`** est une manière usuelle d'encapsuler la vérification des valeurs limites (boundary checking). Toutefois ces types constituent une interface pour une machine parallèle puissante basée sur la génération dynamique de code. Ceci est rendu possible grâce au type **`Program`** qui est la principale innovation du modèle de programmation **RapidMind**. Un mode d'exécution symbolique *retained* est utiliser pour collecter dynamiquement des opérations arbitraires sur les **`Value`** et les **`Array`** dans les objets **`Program`**.

le type `Value`

le type **`Value<N,T>`** est un N-tuple, les instances de ce type contiennent N valeurs de type T, ou T peut être un type numérique de base (un flottant simple ou double précision ou tout autre entier), les flottants 16-bits sont également supportés. Des notations courtes existent pour certaines tailles usuelle comme le **`Value4f`** pour un quarter de floats ou **`Value3ub`** pour un triplet d'entiers 8-bits non signés.
Les opérations arithmétiques standard et les opérations logiques sont surchargés pour les types tuples et opèrent composante par composante. Les fonctions de la bibliothèque standard sont également supportées, comme les fonctions trigonométriques et logarithmiques. En plus des opérations arithmétiques, des opérations de réorganisation des données on été ajoutées au type value: ces opérations permettent la duplication d'une composante ou ou le réordonnancement des composantes. Par exemple, si **`a`** est une valeur de type **`Value<4, float>`** qui représente une couleur RGBA, a(2,1,0) est l'inverse représentant le triplet BGR.
Les calculs sont exprimés en utilisant les tuples de **`Value`** et les opérateurs sur ces types peuvent être utilisés directement pour exprimer du parallélisme SWAR (SIMD Within A Register). Les constructions C++ de modularité tels que namespaces, classes et fonctions.

le type `Array`

Le type **`Array<D,T>`** est également un conteneur de données. Ce qui le distingue du type **`Value`** est le fait qu'il peut être multidimensionnel et de taille variable. L'entier D représente la dimmensionnalité (1,2 ou 3), le type T est le type des éléments du conteneur. Le type des éléments et pour le moment restreint aux instances du type **`Value<N,T>`**.
Les instances du type