Uma Visão Geral da Arquitetura *Cell*

Wellington Mariusso (010086)
Instituto de Computação
UNICAMP
wellington.mariusso@ic.unicamp.br

RESUMO

A arquitetura *Cell* foi desenvolvida a partir de uma proposta das empresas Sony e Toshiba feita à IBM para a criação de uma tecnologia que permitisse alto desempenho de processamento sem o ônus do alto consumo de energia e preços elevados. O resultado disso foi o desenvolvimento conjunto do *Cell*, ou CBEA (*Cell Broadband Engine Architecture*), uma arquitetura *multicore* heterogênea, otimizada para demandas como entretenimento, criptografia e computação científica.

Este texto tem por objetivo apresentar as principais características da arquitetura, proporcionando uma visão geral dos pontos em que ela difere das arquiteturas convencionais.

Palavras Chave

arquitetura de computadores, Cell, multicore

1. INTRODUÇÃO

A arquitetura *Cell*, conhecida formalmente como CBEA *(Cell Broadband Engine Architecture)* [2], foi desenvolvida em conjunto pela IBM, Sony e Toshiba com o objetivo inicial de usar *chips* deste tipo como a base para o novo console de entretenimento da Sony, o *Playstation 3* [4]. Aplicações deste tipo demandam alto poder de processamento, mas não podem ter custo elevado, o que diminuiria muito o possível mercado consumidor.

Para poder suprir estas duas necessidades, associada à necessidade de evitar consumo de energia muito elevado, a alternativa foi criar uma arquitetura *multicore* heterogênea [1]. O fato de o chip ser *multicore*, permite que um alto desempenho seja alcançado através do uso de paralelismo, sem a necessidade de custear várias vezes o encapsulamento do chip – o que teria de ser feito no caso de se usar uma arquitetura paralela usando processadores completamente independentes. Um *chip Cell* típico, como os que serão usados pela Sony em seu console, possui nove processadores capazes de atuar em paralelo, o que permite um bom desempenho.

Para diminuir as necessidades de consumo de energia e também de custo, optou-se por uma arquitetura paralela, porém contendo processadores com características diferentes para realizar tarefas diferenciadas. A arquitetura define que todo *chip Cell* deve ter dois tipos de processadores diferentes: os PPEs *(PowerPC Processor Element)*, que é um processador de propósito geral e os SPUs *(Synergistic Processor Unit)*, que são processadores mais especializados, mas não define exatamente quantos processadores de cada tipo devem estar presentes – exige apenas que pelo menos um de cada tipo esteja no *chip* [2] . Os SPUs também são chamados de SPEs *(SPU Element)*. Em um *chip* típico, há apenas um PPE e oito SPEs. Na verdade, não há a necessidade de que todos os componentes estejam fisicamente no mesmo *chip*, mas em geral essa é uma características importante para a garantia de baixo custo e de performance.

O PPE é um processador PowerPC de 64 bits, de propósito geral. Isso garante que um *Cell* seja capaz de executar aplicações genéricas, sem compilação especial, desde que tenham sido desenvolvidas para PowerPC.

O SPU é um processador mais especializado, menor, com um conjunto de instruções diferente do conjunto de instruções do PowerPC. Este processador permite instruções do tipo SIMD *(Single Instruction, Multiple Data)* e opera com valores de até 128 bits. Este processadores têm por objetivo trabalhar com aplicações que dependam fortemente de manipulação numérica, como criptografia, computação científica e aplicações de mídia (áudio, vídeo, etc.).

Dessa forma, os SPUs atuam como co-processadores para os PPEs, que distribuem as tarefas computacionalmente pesadas. Esta decisão permite um melhor aproveitamento da área ocupada pelo *chip*, dado que os SPUs são menores e mais simples.

Além disso, a arquitetura define a existência de um mecanismo rápido de acesso à memória e uma interface rápida para a realização de entrada e saída de dados.

Interconectando os componentes do *Cell*, há um barramento chamado EIB *(Element Interconnect BUS)*, que transmite até 96 bytes por ciclo.

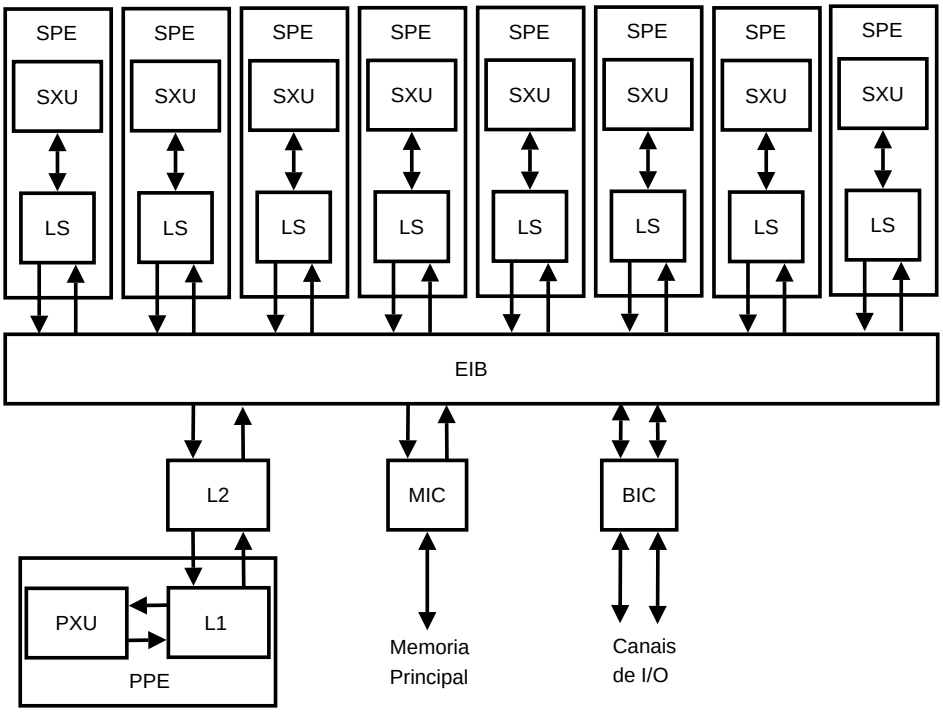

Figura 1: Visão Geral do *Cell*

As características de cada um destes componentes serão discutidas com mais detalhes posteriormente.

2. PRINCIPAIS COMPONENTES

Nesta seção, são apresentados os principais componentes de arquitetura *Cell* e suas características mais importantes.

2.1 PowerPC Processor Element (PPE)

A arquitetura CBEA define que todo sistema compatível com *Cell* deve ter, pelo menos, um processador do tipo PPE.

Este processador deve ser um processador de 64 bits, compatível com a família *PowerPC*, mas deve ter algumas características diferenciadas. É necessário, por exemplo, que o processador tenha um conjunto de instruções relativo às extensões multimídia, permitindo que ele também seja usado para execução de programas fortemente dependente de processamento de dados. Portanto, o PPE também possui um pequeno conjunto de instruções SIMD.

O PPE ainda pode ser dividido em duas partes: o cache L2 e a PXU *(PowerPC eXecution Unit)*.

Além disso, o PPE presente no *Cell* deve ter uma implementação que permita a operação em freqüências altas, além de ter características de baixo consumo de energia, mesmo que para isso algumas funcionalidades dos *PowerPCs* mais novos não estejam presentes. Boa parte das características do PPE não são parte da especificação CBEA, sendo dependentes de implementação.

No caso do *Cell* típico, que será usado pela Sony nos consoles de entretenimento, muitas destas características podem ser observadas e são interessantes para exemplificar o potencial desta unidade. As características do PPE que seguem são referentes a esta implementação específica.

O cache L2, por exemplo, foi reduzido em relação ao que existe nos processadores *PowerPC* recentes. O cache L2 possui apenas 512KB, já que é comum o cache ocupar uma área muito grande do *chip*. Dessa forma, pode-se poupar área para que pudessem ser colocados outros processadores no mesmo *chip*. Essa decisão foi de certa forma contrabalanceada pelo uso de memórias externas rápidas, do tipo XDR da RAMBUS. O cache é L1 é de 32KB, e é dividido em um cache para dados e outro cache para instruções.

A PXU desta implementação é capaz de executar instruções de duas threads simultaneamente, mas não reordena instruções – uma involução se compararmos aos *PowerPCs* mais novos, que já possuem esta característica. A PXU pode ser subdivida em três componentes ainda menores, mas ainda com funcionalidade bem definida. A IU *(*Instruction Unit), a XU *(Fixed Point Execution Unit)* e a VSU *(Vector Scalar Unit)*.

A IU é responsável pelo controle do PPE e cuida de tarefas como *fetch* de instruções, decodificação e *branch prediction*. A cada ciclo, 4 instruções sofrem *fetch* e duas delas são processadas pela IU. A tabela usada para *branch prediction* possui 4 KB de tamanho e possui um *pipeline* de 23 estágios.

A XU é responsável pela execução das instruções que envolvem números inteiros e de instruções do tipo *Load* e *Store*. A XU possui, para cada thread, 32 registradores de propósito geral de 64 bits e a unidade de *Load* e *Store*, da qual o cache L1 de dados faz parte, tem características não bloqueantes. Isso significa que, enquanto o cache está processando uma

requisição à memória por conta de um *cache miss*, pode receber outras requisições. Na prática é possível, portanto, observar um *cache hit* ao mesmo tempo que há o processamento do *cache miss*.

A VSU é responsável pela execução de instruções SIMD e também atua como unidade de ponto flutuante. A unidade de ponto flutuante da VSU também possui um banco de 32 registradores de 64 bits por thread, tem um pipeline interno de 10 estágios e é compatível com as especificações da IEEE. A unidade vetorial da VSU processa vetores de 32 entradas de 128 bits por thread e todas as instruções para esta unidade são instruções do tipo SIMD de 128 bits.

2.2 Synergistic Processing Element (SPE)

Os processadores do tipo SPE são bem diferentes dos processadores do tipo PPE. O objetivo principal de tais processadores é garantir alto desempenho para aplicações que processam dados de maneira intensiva, tais como operações para tratamento de áudio, vídeo, criptografia e cálculos para simulação de fenômenos.

2.2.1 Local Store

Um das maiores barreiras para alcançar altos níveis de performance é o gargalo de acesso à memória. O custo de fazer acessos constantes à memória é muito grande e deve ser evitado sempre que possível[5].

Para tentar minimizar tais problemas, os SPEs têm memória própria e não têm acesso direto à memória principal do sistema. Essa memória é uma memória rápida (tão rápida quanto memória usada em sistemas de cache) e é usada para guardar tanto instruções quanto dados. A implementação dessas memória tenta minimizar o número de transistores necessários para guardar os dados, para melhorar o aproveitamento da área do *chip*.

A comunicação com a memória principal é feita por DMA e é responsabilidade de uma unidade interna de gerência de memória, chamada MFC *(Memory Flow Controller)*. As transferência de dados com a memória principal podem ser iniciadas tanto pelos PPEs quanto pelos SPEs e é possível, através de um esquema de filas, executar simultaneamente múltiplos comandos de DMA.

Dessa forma, os dados necessários para executar uma tarefa em um SPE em geral estão sempre presentes no *Local Store* e foram pré carregados pelo PPE antes de entregar a tarefa. Isso minimiza bastante os acessos (principalmente os acessos aleatórios) à memória principal, garantindo uma performance muito boa em relação às arquiteturas convencionais.

Em uma implementação típica do *Cell*, o Local Store ocupa de 85% a 90% da área total do SPE, o que equivale a 256KB de memória rápida e local.

O *Local Store* provê dois canais de acesso: um canal mais largo, de 128 bytes, é usada para interface com o sistema de DMA. Um canal mais enxuto, de 128 bits, é usado para interfacear com a unidade de execução do SPE.

Como o *Local Store* é uma unidade rápida de memória, não há a necessidade de memória cache para os SPEs. Mesmo assim, a arquitetura permita que haja um cache entre o *Local Store* e o MFC, para evitar chamadas à memória principal desnecessárias.

2.2.2 A unidade de execução – SXU

A unidade de execução de instruções do SPE é chamada de SXU *(SPE eXecution Unit)* e é otimizada para trabalhar com dados em unidades de 128 bits. Apenas uma única thread é executada por vez, mas é possível executar mais de umas instrução por ciclo, dependendo dos tipos de instrução. Caso seja uma instrução de acesso a memória e uma instrução de operação com dados, é possível que ambas sejam executadas simultaneamente. A SXU não realiza reordenação de instruções, deixando esta tarefa para o compilador e o conjunto de instruções é novo, desenhado especificamente para o *Cell*, para permitir o aproveitamento máximo das funcionalidades implementadas.

A SXU tem um banco de 128 registradores de 128 bits cada, o que permite que o compilador tenha muitas opções para reorganizar as instruções de forma a minimizar possíveis perdas com latência. Instruções com inteiros são executadas em dois ciclos e instruções do tipo *Load* e de ponto flutuante de precisão simples são executadas em apenas 6 ciclos.

Essa performance para execução de operações com ponto flutuante é alcançada facilmente porque a unidade de ponto flutuante não é compatível com o padrão IEEE, e portanto algumas simplificações foram feitas. Isso permitiu que uma unidade menor, mais simples e mais rápida fosse criada.

Os projetistas do *Cell* argumentam que este não é um problema sério [1], visto que para a maioria das aplicações a que a arquitetura está destinada, uma pequena falha de arredondamento provocada por problemas na normalização não implicaria em perdas consideráveis. Em um vídeo, por exemplo, não seria tão problemático sem um pixel parece ligeiramente mais claro ou mais escuro. Quando um personagem em um jogo realiza um salto, não teria problema se o do cálculo da física envolvida provocasse um deslocamento aquém do possível para o personagem, em dadas situações. Caso haja a necessidade de tratar com mais seriedade os dados de ponto flutuante, deve-se avaliar se não seria melhor usar precisão dupla (que é tratada com mais cuidado pela SXU) ou então usar o próprio PPE para realizar tais cálculos.

O conjunto de instruções da SXU é SIMD [3]– o que significa que uma única instrução é capaz de processar muitos dados simultaneamente. As instruções podem tratar os dados de várias formas diferentes, contato que no total tenham sido processados 128 bits. É possível, com uma única instrução SIMD, realizar operações com:

- 16 inteiros de 8 bits
- 8 inteiros de 16 bits
- 4 inteiros de 32 bits
- 4 valores de ponto flutuante com precisão simples
- 2 valores de ponto flutuante com precisão dupla

É importante observar que, diferentemente do PPE, o SPE não possui características de *branch prediction*, pois uma unidade deste tipo traria uma complexidade adicional que não é desejada para este elemento. Por outro lado, é possível que o programador (ou compilador) possa indicar, através de instruções especiais, qual é o destino mais provável de um *branch*. Dessa forma, quando a SXU detecta que um *branch* se aproxima, realiza o pré-*fetch* das instruções que estão no suposto endereço de destino do *branch*.

2.2.3 Restrições

O SPE não é um processador de propósito geral como o SPE e portanto possui algumas restrições. A principal delas é o fato de não ter acesso direto à memória principal – todos os acessos à memória principal têm de ser feitos através do controlador de memória, por comandos DMA. O SPE também não sabe distinguir entre diferentes modos do processador, como modo usuário e modo privilegiado. É impossível ao SPE acessar recursos de controle do sistema, como tabelas de página por exemplo.

Por conta destas restrições, grande parte das tarefas realizadas pelo sistema operacional devem ser executadas pelos PPEs.

3. SOFTWARE

Como a arquitetura *Cell* é razoavelmente diferente do que vem sendo trabalhado com as arquiteturas convencionais, é importante que os softwares tenham que ser desenvolvidos com características especiais.

Embora qualquer aplicação compatível com *PowerPC* possa ser executada no *Cell* sem a necessidade de alterações, é de suma importância que software tenha que ser desenvolvido especificamente para a arquitetura *Cell* para que todo o potencial seja aproveitado.

Essa iniciativa deve começar pela criação de novos compiladores. Paralelamente ao desenvolvimento do *Cell*, uma equipe da IBM trabalhou na criação de compiladores otimizados, mas ainda há muito o que caminhar neste sentido.

Como vimos, tanto os SPEs quanto os PPEs foram projetados imaginando que boa parte das tarefas de otimização fossem realizadas pelo processador. Por exemplo: não há reordenação de instruções em nenhum dos processadores. No caso do PPE, não há nem ao menos *branch prediction*.

Além disso, é necessário que os programas para essa nova arquitetura sejam criados levando em consideração suas características. Os programas devem ser feitos de forma que possam ser divididos em tarefas menores e independentes, que podem ser divididas entres os vários SPEs. Deve-se também programar imaginando que existe um conjunto de instruções SIMD que deve ser usado para maximizar performance. Por outro lado, é necessário desenvolver software capaz de escalonar de maneira eficientes tais tarefas para os SPEs corretos, bem como garantir que os dados necessários estejam presentes no *Local Store* do SPE.

Freqüência de *clock*	4+ GHz
Pico de performance (precisão simples)	256 GFlops
Pico de performance (precisão dupla)	26 GFlops
Cache L2	512 KB
Local Store (total)	4 MB
Área	221 mm^2
Transistores	240 milhões

Tabela 1: Estatísticas relacionadas ao *Cell*

4. PERFORMANCE E ESTATÍSTICAS

Para termos uma idéia do que um chip *Cell* é capaz de fazer, é interessante apresentarmos alguns dados. Os dados que estão sendo apresentados na Tabela 1 são referentes à implementação típica do *Cell*, que será usada no Sony *Playstation 3*.

5. CONCLUSÃO

A arquitetura *Cell*, desenvolvida em conjunto pela Sony, Toshiba e IBM é uma arquitetura *multicore* de alto desempenho, que tenta minimizar custo e gasto com energia.

Atualmente a arquitetura já tem um mercado potencial gigantesco, que são os consumidores de console *Playstation* da Sony, mas o potencial do *Cell* permite que ele ainda seja introduzido em várias áreas diferentes.

Para garantir o sucesso da plataforma, porém, é imprescindível que seja desenvolvido software para maximizar o uso da arquitetura.

6. REFERÊNCIAS

[1] E. Altman, P. Capek, et al. The cell architecture – innovation matters. IBM, Sony, Toshiba, ISSCC, 2005.

[2] IBM. *Cell Broadband Engine Architecture – Version 1.0*. IBM, Sony, Toshiba, 2005.

[3] IBM. *Synergistic Processor Unit Instruction Set Architecture – Version 1.0*. IBM, Sony, Toshiba, 2005.

[4] J. A. Kahle, M. N. Day, H. P. Hofstee, et al. Introduction to the cell processor. *IBM J. Res & Dev.*, 49(4/5):589–604, July/September 2005.

[5] W. Wulf and S. McKee. Hitting the memory wall: Implications of the obvious. *ACM Computer Architecture News*, 23(1):20–24, 1995.