Arquitetura do Processador Cell Broadband Engine

Kleber Sacilotto de Souza
RA: 024249

RESUMO

Este trabalho apresenta uma introdução à arquitetura do processador Cell Broadband Engine (Cell BE). Essa arquitetura foi densenvolvida em conjunto pelas empresas Sony, Toshiba e IBM e apresenta um novo conceito de arquitetura multicore. O Cell BE possui um núcleo chamado de POWER Processing Element (PPE) com suporte a duas threads e oito núcleos chamados de Synergistic Processing Elements (SPEs). Também são apresentados algumas aplicações desta arquitetura e alguns dados sobre seu desempenho.

Categories and Subject Descriptors

D.1.2 [**Computer Systems Organization**]: Processor Architecture – Multiple Data Stream Architectures (Multiprocessors).

General Terms

Performance, Design.

Keywords

STI, Cell, Cell BE, Cell BEA, CBEA, arquitetura, multiprocessadores, supercomputadores

1. INTRODUÇÃO

O processador Cell Broadband Engine (Cell BE ou somente Cell), é a primeira implementação da Cell Broadband Engine Architecture (CBEA) [3], que é uma arquitetura desenvolvida em conjunto pela Sony, Toshiba e IBM (também conhecido por STI) com o objetivo de prover processamento de alto-desempenho com baixo consumo de energia e com uma boa relação custo/benefício para uma ampla gama de aplicações [6]. O Cell preenche o abismo entre processadores de propósito geral e processadores de alto-desempenho especializados. Enquanto o objetivo dos processadores de propósito geral é alcançar um bom desempenho numa vasta gama de aplicações, e o objetivo de um hardware especializado é obter o melhor desempenho em uma única aplicação, o objetivo do Cell é obter alto desempenho em workloads críticos para jogos, multimídia, aplicações que exigem uma grande largura de banda[11] e aplicações científicas.

O Cell BE inclui um POWER Processing Element (PPE) e oito Synergistic Processing Elements (SPEs). A arquitetura Cell BE foi desenvolvida para ser usada por uma ampla variedade de modelos de programação e permite que as tarefas sejam divididas entre o PPE e os oito SPEs.

O design do Cell BE foi focado em melhorar as relações desempenho/área e desempenho/consumo de energia. Esses objetivos foram alcançados basicamente na utilização de núcleos poderosos, porém simples, que fazem um uso mais eficiente da área com menos dissipação de potência. Suportados por um barramento com uma larga banda de dados, estes núcleos podem trabalhar tanto independentemente como em conjunto. Com um suporte a um número grande de acessos simultâneos dos núcleos a memória, a largura de banda da memória também pode ser usada mais eficientemente. A filosofia do design do Cell BE é de algum modo similar à tendência de ter vários núcleos de propósito geral no mesmo chip, entretanto no Cell BE os núcleos são apenas muito mais simples, mas poderosos.

2. ARQUITETURA DO CELL BE

2.1 Visão Geral

O Cell BE implementa um multiprocessador em um único chip com nove processadores operando em um sistema de memória compartilhado e coerente. Ele inclui um POWER Processing Element (PPE) de propósito geral de 64-bits e oito Synergistic Processing Elements (SPEs) interconectados por um barramento de alta velocidade, coerente em nível de memória, chamado de Element Interconnect Bus (EIB). Tanto o PPE quanto os SPEs são arquiteturas RISC com instruções de formato fixo de 32 bits. Os endereços de memória do sistema são representados tanto para o PPE quanto para os SPEs em 64 bits que podem endereçar teoricamente 2^{64} bytes, embora na prática nem todos esses bits são implementados em hardware. A figura 1 mostra a uma visão em alto nível da implementação do Cell.

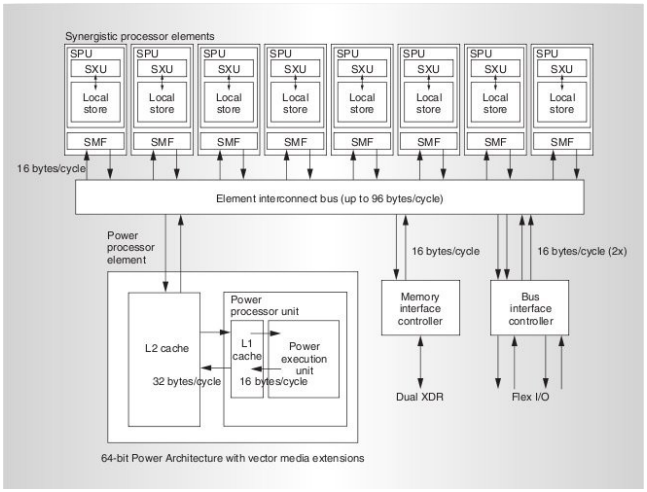

Figura 1. Implementação do processador Cell BE

Um die do processador Cell BE possui aproximadamente 241 milhões de transistores, numa área de 235mm^2. A freqüencia mais comum utilizada é de 3,2GHz, porém freqüencias maiores

que 4GHz já foram alcançadas [5]. A figura 2 mostra a foto de um die do processador Cell BE onde os elementos principais estão identificados.

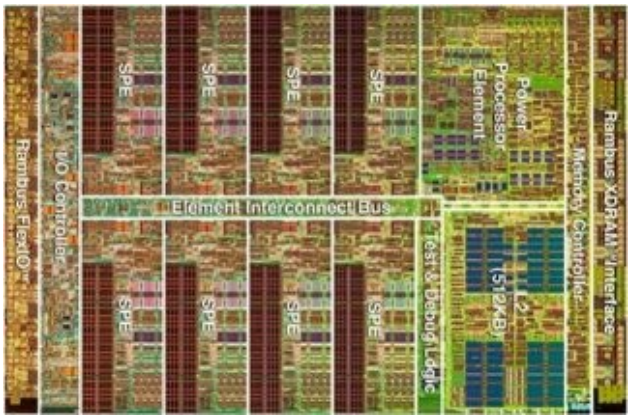

Figura 2. Foto de um die do processador Cell BE

2.2 O Power Processing Element (PPE)

O PPE consiste em um POWER Processing Unit (PPU) conectado a uma cache L2. O PPE é o processador principal do Cell BE e é responsável em executar o sistema operacional e coordenar os SPEs. O PPE é desenvolvido baseado na arquitetura Power de 64 bits da IBM com extensões vetoriais de multimídia (VMX, ou Vector Multi-media Extension) para operações SIMD (single instruction, multiple data). Ele é completamente compatível com a especificação da Arquitetura Power de 64 bits e pode executar sistemas operacionais e aplicações de 32 ou 64 bits. Além disso, o PPE inclui uma unidade de gerenciamento de memória (MMU, ou memory management unit) e uma unidade AltiVec que possui um pipeline completo para operações em ponto flutuante de precisão simples (o AltiVec não suporta vetores de ponto flutuante de precisão dupla).

O PPE possuiu dois níveis de memória cache. O nível 1 (L1) possui duas cachês de 32KB, uma para instruções a outra para dados, e o nível 2 (L2) possui uma cache unificada de 512KB (instruções e dados). O tamanho da linha de cache é de 128 bytes.

O PPU é um processador de execução em ordem, despacho duplo e com suporte a duas threads. Um diagrama do pipeline do PPU é mostrado na figura 3.

O PPE pode fazer o fetch de quatro instruções por vez, e despachar duas. Para melhorar o desempenho do seu pipeline em ordem, o PPE utiliza pipelines com execução atrasada e permite execução fora de ordem limitada de instruções de load. Isso permite o PPE obter algumas vantagens da execução fora de ordem sem aumentar significativamente sua complexidade.

O PPE acessa a memória principal com instruções de load e store que movem os dados entre a memória principal e o conjunto de registradores privado, sendo que este conteúdo pode ser armazenado em cache. O método de acesso à memória do PPE é semelhante às tecnologias convencionais de processadores. Os SPEs, diferentemente do PPE, precisam utilizar comandos de DMA para acessar a memória principal. Mais detalhes sobre este método serão mostrados na próxima seção.

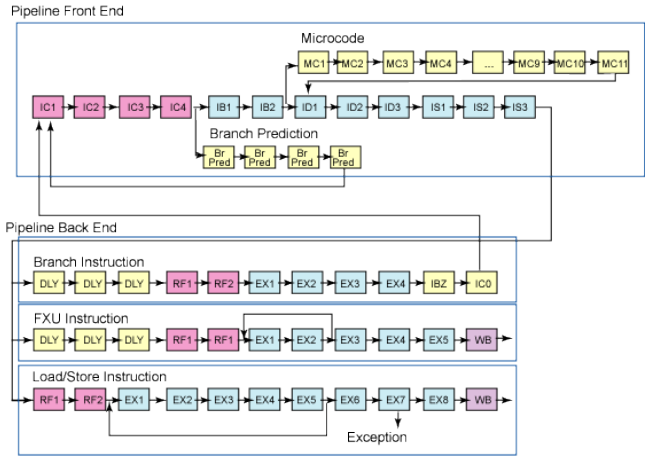

Figura 3. Pipeline do PPU

2.3 Os Synergistic Processing Elements (SPEs)

O SPE tem um design modular que consiste em uma Synergistic Processing Unit (SPU) e um Memory Flow Controller (MFC). Um SPU é um elemento de computação com suporte a SIMD e 256KB de armazenamento local dedicado (LS, ou Local Store), que possui um espaço de endereçamento de 32 bits [11]. O MFC contém um controlador DMA com uma MMU associada, assim como uma Unidade Atômica (Atomic Unit) para tratar das operações de sincronização com outras SPUs e com a PPU.

O PPE é mais competente que os SPEs em tarefas intensivas em controle e mais rápido na troca de tarefas. Os SPEs são mais competentes em tarefas de computação intensiva e mais lentos na troca de tarefas. Entretanto, todos os elementos de processamento são capazes de realizar ambos os tipos de funções. Essa especialização é um fator significante na melhoria em ordem de magnitude no desempenho, na área ocupada pelo chip e na eficiência no consumo de energia que o Cell BE alcança em relação aos processadores para PC convencionais

Uma SPU é um processador de execução em ordem, com despacho duplo e com um banco de registradores com 128 registradores de 128 bits usados tanto para operações de ponto flutuante quanto operações de inteiros. A SPU opera diretamente sobre as instruções e os dados em seu armazenamento local dedicado, e conta com uma interface para acessar a memória principal e outros armazenamentos locais. Esta interface, que é o MFC, executa independentemente da SPU e é capaz de traduzir endereços e realizar transferências DMA enquanto a SPU continua a execução do programa.

O suporte SIMD das SPUs podem realizar operações sobre dezoito inteiros de 8 bits, oito inteiros de 16 bits, quatro inteiros de 32 bits, ou quatro números em ponto flutuante de precisão simples por ciclo. A 3,2GHz, cada SPU é capaz de realizar até 51.2 bilhões de operações com inteiros de 8 bits ou 25.6GFLOPs com precisão simples. A figura 4 mostra as principais unidades funcionais de uma SPU: (1) uma unidade de ponto flutuante para multiplicação de inteiros e de ponto flutuante de precisão simples

e dupla; (2) uma unidade de ponto fixo par para aritmética, operações lógicas, e deslocamento de palavras; (3) uma unidade de ponto fixo ímpar para permutações, embaralhamentos, e rotação de quatro palavras; (4) uma unidade de controle para seqüenciamento de instruções e execução de desvios; (5) uma unidade de aramazenamento local para loads e stores; também fornece instruções para a unidade de controle; (6) uma unidade de transporte canal/DMA que é responsável em controlar as entradas e saídas através do MFC.

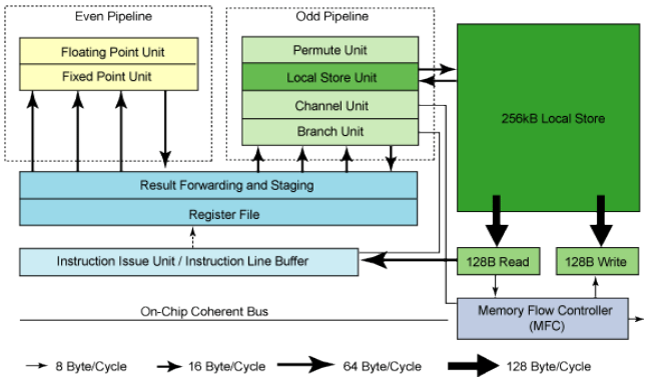

Figura 4. Unidades funcionais da SPU

Como mostra a figura 4, cada unidade funcional é atribuída a um dos dois pipelines de execução. As unidades de ponto flutuante e de ponto fixo estão no pipeline par enquanto que o resto das unidades funcionais está no pipeline ímpar. A SPU pode despachar e completar até duas instruções por ciclo, uma em cada um dos pipelines de execução. Um despacho duplo ocorre quando um grupo de instruções de um fetch tem duas instruções que podem ser despachadas, uma que é executada por uma unidade no pipeline par e a outra por uma unidade no pipeline ímpar.

Há três tipos de fetch de instruções: flush-initiaded fetches, inline fetches e hint fetches. Para fazer o fetch de uma instrução, a lógica de fetch de instruções lê 32 instruções de uma vez e armazena no seu buffer de instruções (ILB, ou instruction line buffer), do qual duas instruções por vez são enviadas à unidade lógica. Quando os operandos estiverem prontos, a lógica de despacho envia as instruções para as unidades funcionais para execução. Os pipelines das unidades funcionais variam de dois a sete ciclos. Instruções de hint fazem o preload de intruções no ILB.

Os SPEs são elementos de processamento independentes, cada um executando seus próprios programas ou threads individuais. Cada SPE tem acesso completo à memória compartilhada, incluindo o espaço de memória dedicado ao memory-mapped I/O (MMIO) implementado pelas unidades de DMA. Há uma dependência múltipla entre o PPE e os SPEs. Os SPEs dependem do PPE para executar o sistema operacional, e, em muitos casos, a thread de alto nível que controla a aplicação. O PPE depende dos SPEs para prover a maior parte do processamento responsável pelo desempenho da aplicação.

Os SPEs acessam a memória principal com comandos DMA que movem dados e instruções entre a memória principal e o armazenamento local (LS). O fetch das instruções e as instruções de load e store de um SPE acessam seu LS privado ao invés da memória principal compartilhada, e o LS não tem um cache associado. A organização do armazenamento em três níveis (banco de registradores, LS e memória principal), com transferências DMA assíncronas entre o LS e a memória principal, é uma mudança radical da arquitetura convencional e dos modelos de programação. Esta organização explicitamente paraleliza a computação com a transferência de dados e instruções que alimentam a computação e o armazenamento do resultado da computação em uma memória principal.

Uma das motivações para essa mudança radical é a latência das memórias, medida em ciclos de processador, que tem crescido em ordem de centena de vezes entre os anos 1980 e 2000. O resultado é que o desempenho das aplicações é, na maioria dos casos, limitado pela latência da memória ao invés do poder de processamento ou largura de banda. Quando um programa seqüencial sendo executado em uma arquitetura convencional executa uma instrução de load que dá miss na cache, a execução do programa pode ser interrompida por centenas de ciclos. (Técnicas como threading em hardware podem tentar esconder esses stalls, mas isso não ajuda no caso de aplicações que possuem somente uma thread.) Comparados com essa penalidade, os poucos ciclos que levam para configurar uma transferência DMA para um SPE é uma penalidade bem menor, especialmente considerando o fato de os controladores DMA de cada um dos SPEs podem gerenciar até 16 transferências DMA simultaneamente. A antecipação eficiente da necessidade de DMA pode fornecer uma entrega just-in-time de dados, o que pode reduzir ou até eliminar completamente os stalls. Processadores convencionais, até mesmo com uma larga e custosa especulação, conseguem obter, no melhor caso, apenas alguns acessos simultâneos à memória.

Um dos métodos de transferência DMA suporta uma lista, como uma lista *scatter-gatter*, de transferências DMA que é construída no armazenamento local do SPE. Portanto, o controlador DMA do SPE pode processar a lista de maneira assíncrona enquanto o SPE realiza operações em dados transferidos anteriormente. As transferências DMA podem ser iniciadas e controladas pelo SPE que está fornecendo ou recebendo os dados, ou alguns casos, pelo PPE ou outro SPE.

2.4 O Element Interconnect Bus (EIB)

O Element Interconnect Bus (EIB) no Cell BE permite a comunicação entre o PPE, os SPEs, a memória principal localizada externamente ao chip, e I/O externo (veja figura 5). O EIB consiste em um barramento de endereço e quatro anéis de dados de 16 bytes de largura, dos quais dois os dados são transmitidos em sentido horário, e dois em sentido anti-horário. Cada anel pode potencialmente permitir até três transmissões de dados concorrentes desde que seus caminhos não se sobreponham. O EIB opera a metade da velocidade do processador.

Cada requisitante do EIB começa com um número inicial pequeno de créditos de comandos para enviar requisições ao barramento. O número de créditos é o tamanho do buffer de comandos dentro do EIB para aquele requisitante em particular. Um crédito de comando é utilizado para cada requisição ao barramento. Quando um slot se abre no buffer de comandos assim que uma requisição anterior progride no pipeline de requisições do EIB, o EIB devolve o crédito ao requisitante.

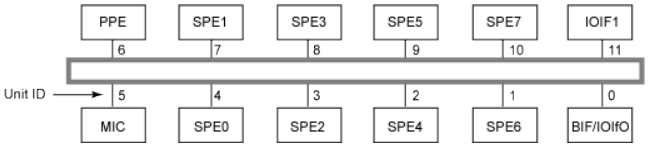

Figura 5. EIB e seus elementos

Quando um requisitante precisa de um anel de dados, ele envia uma única requisição para o árbitro do barramento de dados do EIB. O árbitro arbitra entre os múltiplos requisitantes e decide qual anel de dados é concedido a qual requisitante e quando. Para o controlador de memória é dada a maior prioridade para prevenir stall de dados no requisitante da leitura dos dados, enquanto todos os outros são tratados igualmente com uma prioridade em *round-robin*. Qualquer requisitante do barramento pode usar qualquer um dos quatro anéis de dados para enviar ou receber dados. O árbitro de dados não concede um anel de dados a um requisitante se a transferência cruzar mais do que a metade da volta do anel para chegar ao seu destino, ou se interferir em outra transferência de dados que esteja em progresso no momento.

Cada unidade conectada ao EIB pode enviar e receber simultaneamente 16B de dados a cada ciclo do barramento. A largura de banda máxima de todo o EIB é limitada a taxa máxima a qual endereços são monitorados através de todas as unidades no sistema, a qual é um por ciclo do barramento. Como cada requisição pode potencialmente transmitir até 128B de dados, o pico de largura de banda no EIB a 3,2GHz é 128Bx1,6GHz = 204,8GB/s.

A largura de banda sustentada pelo EIB será menor do que o pico de largura de banda devido a vários fatores: a distância relativa entre o destino e a origem, o potencial de novas transmissões interferirem em aquelas que já estejam em progresso, o número de processadores Cell BE no sistema, se as transmissões de dados são para ou da memória ou entre armazenamentos locais nos SPEs, e a eficiência do árbitro de dados.

Larguras de banda reduzidas podem resultar quando: (1) todos os requisitantes acessam o mesmo destino ao mesmo tempo, como memória no mesmo armazenamento local; (2) todas as transmissões são para o mesmo destino e deixam dois dos quatros anéis ociosos; (3) há um grande número de transferências parciais de linhas de cache diminuindo a eficiência do barramento; e (4) todas as transferências de dados passam por 6 unidades até chegarem ao seu destino, inibindo as unidades que estão no caminho de utilizarem o mesmo anel [3].

2.5 O Subsistema de Memória

O Memory Flow Controller (MFC) é o mecanismo de transferência de dados. Ele provê o método primário para transferência de dados, proteção, e sincronização entre a memória principal e o armazenamento local associado, ou entre o armazenamento local associado a outro armazenamento local. Um comando MFC descreve a transferência a ser executada. Um dos objetivos arquiteturais principais do MFC é realizar estas transferências de maneira mais rápida possível, deste modo maximizando o throughput geral do processador.

Comandos que transferem dados são chamados de comandos MFC DMA. Estes comandos são convertidos em transferências DMA entre o domínio do armazenamento local e o domínio do armazenamento principal. Cada MFC pode suportar múltiplas transferências DMA ao mesmo tempo e pode manter e processar múltiplos comandos MFC. Para alcançar isso, o MFC mantém e processa filas de comandos MFC. Cada MFC provê uma fila para a SPU associada (fila de comandos MFC SPU) e uma fila para outros processadores e dispositivos (fila de Proxy de comandos MFC). Logicamente, um conjunto de filas MFC é sempre associada a cada SPU em um processador Cell BE [9].

O Memory Interface Controller (MIC) do chip do Cell BE é conectado à memória externa RAMBUS XDR através de dois canais XIO que operam a uma freqüência máxima efetiva de 3,2GHz (400 MHz, Octal Data Rate). Ambos os canais RAMBUS podem ter oito bancos operando concorrentemente e um tamanho máximo de 256MB, em um total de 512MB.

O MIC possui filas separadas de requisições de leitura e escrita para cada canal XIO operando independentemente. Para cada canal, o árbitro do MIC alterna o despacho entre filas de leituras e escritas após o mínimo de oito despachos de cada fila ou até que a fila fique vazia, o que for mais curto. Às requisições de leitura de alta prioridade são dadas prioridade sobre leituras e escritas.

Escritas de 16B ou mais, mas com menos de 128B, pode ser escritas diretamente na memória usando uma operação masked-write, mas escritas menores que 16B precisam de uma operação read-modify-write. Devido ao número pequeno de buffers para operação read-modify-write, a parte de leitura da operação read-modify-write é dada alta prioridade sobre leituras normais, enquanto que a parte de escrita é dada prioridade sobre as escritas normais.

2.6 Flexible I/O Controller

Há sete links RAMBUS RRAC FlexIO para transmissão e cinco para recepção e possuem uma largura de 1B cada. Esses links podem ser configurados como duas interfaces lógicas. Com os links FlexIO operando a 5GHz, a interface de I/O provê uma largura de banda bruta de 35GB/s de saída e 25GB/s de entrada. Uma configuração típica pode ter uma interface de I/O configurada com uma largura de banda bruta de 30GB/s de saída e 20GB/s de entrada; e a outra interface de I/O com largura de banda bruta de 5GB/s de saída e 5GB/s de entrada.

Dados e comandos enviados para a interface de I/O são enviados como pacotes. Além do comando, resposta, e dados, cada pacote pode carregar informações tais como tag de dados, tamanho dos

dados, id do comando, e informações de controle de fluxo, além de outras informações. Devido a esses overheads, e tempos de chegada de comandos e dados não ótimos, a largura de banda efetiva nas duas interfaces é tipicamente menor, e varia de 50% a 80% da largura de banda bruta.

3. PROGRAMANDO PARA O CELL BE

3.1 Conjunto de Instruções

O conjunto de instruções para o PPE é uma versão entendida do conjunto de instrução da Arquitetura PowerPC. As extensões consistem em extensões multimídia vector/SIMD, algumas poucas mudanças e adições às instruções da Arquitetura PowerPC, e funções intrínsecas C/C++ para as extensões multimídia vector/SIMD.

O conjunto de instruções para os SPEs é um conjunto de instruções SIMD novo, o *Synergistic Processor Unit Instructions Set Architecture*, acompanhado de funções intrínsecas C/C++. Ele também possui um conjunto único de comandos para gerenciar transferências DMA, eventos externos, troca de mensagens intra-processador, e outras funções. O conjunto de instruções para os SPEs é similar ao das extensões multimídias vector/SIMD do PPE, no sentido que eles operam sobre vetores SIMD. Entretanto, os dois conjuntos de instruções vetoriais são distintos. Programas desenvolvidos para o PPE e para os SPEs são geralmente compilados por compiladores diferentes, gerando código para dois conjuntos de instruções completamente diferentes.

3.2 Domínios de Armazenamento e Interfaces

O processador Cell BE possui dois tipos de domínios de armazenamento: um domínio principal de armazenamento e oito domínios LS das SPEs, como mostrado na figura 6. Além disso, cada SPE possui uma interface de canal para comunicação entre o seu SPU e o seu MFC.

O domínio principal de armazenamento, o qual é todo o espaço de endereços efetivo, por ser configurado por software sendo executado em modo privilegiado no PPE para ser compartilhado por todos os elementos do processador e dispositivos memory-mapped do sistema. O estado de um MFC é acessado por sua SPU associada através da interface de canal. Este estado também pode ser acessado pelo PPE e outros dispositivos, incluindo outros SPEs, por meio dos registradores de MMIO do MFC no espaço de armazenamento principal. O LS de uma SPU pode também ser acessada pelo PPE e outros dispositivos através do espaço de armazenamento principal de uma maneira não coerente. O PPE acessa o espaço de armazenamento principal através do seu PowerPC processor storage subsystem (PPSS).

Cada MFC possui uma unidade de synergistic memory management (SMM) que realiza o processamento de informações de tradução de endereços e de permissão de acesso que são fornecidas pelo sistema operacional sendo executado no PPE. Para processar um endereço efetivo fornecido por um comando DMA, o SMM usa essencialmente o mesmo mecanismo de tradução de endereços e proteção é que utilizado pela unidade de gerenciamento de memória (MMU) no PPSS do PPE. Portanto, transferências DMA são coerentes com relação ao armazenamento do sistema, e os atributos do armazenamento do sistema são controlados pelas tabelas de página e segmento da Arquitetura PowerPC.

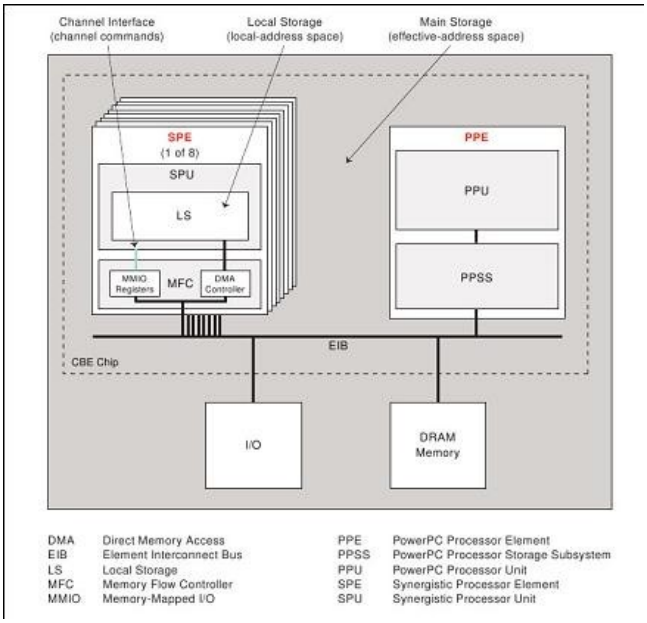

Figura 6. Domínios de armazenamento

3.3 Ambiente de Execução

O PPE executa aplicações e sistemas operacionais desenvolvidos para a Arquitetura PowerPC, o que inclui instruções da Arquitetura PowerPC e instruções da extensão multimídia vector/ SIMD. Para utilizar todas as características do processador Cell BE, o PPE precisa executar um sistema operacional que suporta todas essas características, como multiprocessamento utilizando os SPEs, acesso as operações da extensão multimídia vector/SIMD, o controlador de interrupções do Cell BE, e todas as outras funções fornecidas pelo processador Cell BE.

Uma thread principal sendo executada no PPE pode interagir diretamente com uma thread em um SPE através do LS do SPE. Ela pode interagir indiretamente através do espaço de armazenamento principal. A thread do PPE pode também se comunicar através de uma caixa de mensagens e sinalização de eventos que são implementados em hardware.

O sistema operacional define o mecanismo e as políticas para selecionar um SPE disponível para agendar a execução de uma thread SPU. O sistema operacional também é responsável pelo carregamento em tempo de execução, passagem de parâmetros aos programas sendo executados no SPE, notificação de erros e eventos do SPE, e suporte a depuração.

3.4 Modelos de Programação

Devido a flexibilidade natural do Cell BE, há várias possibilidades para a utilização de seus recursos. Nas próximas seções veremos alguns dos possíveis modelos [10].

3.4.1 *Fila de tarefas*

O PPE mantém uma fila de tarefas em memória, agenda as tarefas para serem executadas nos SPEs e monitora o progresso das tarefas. Cada SPE roda um "mini kernel" cujo papel é buscar uma tarefa na fila de tarefas, executá-la, e sincronizar com o PPE.

3.4.2 *Multi-tarefa gerenciada pelos SPEs*

O kernel e o agendamento é distribuído entre os SPEs. A sincronização é feita através do uso de exclusão mútua e semáforos, como em um sistema operacional convencional. Tarefas prontas para serem executadas por um SPE são mantidas em um *pool*. Os SPEs utilizam memória compartilhada para organizar a comunicação entre SPEs.

3.4.3 *Processamento em fluxo*

Neste modelo cada SPE executa um programa distinto. Os dados chegam em um fluxo de entrada e são enviados para os SPEs. Quando um SPE termina o processamento, os dados de saída são enviados a um buffer de fluxo de saída.

3.5 Exemplo

O exemplo a seguir mostra um simples exemplo de um "Hello world!" onde um programa principal, que é executado no PPE, cria uma thread para cada SPU (executada pelo Código 1) e espera o retorno de cada uma delas (Código 2) [7].

Este exemplo utiliza a libspe (SPE Runtime Management Library) que é uma biblioteca C/C++ que fornece uma API (Application Programming Interface) para aplicações acessarem os SPEs do Cell BE [1].

Código 1: Código para o SPE

```
#include <stdio.h>

int main(unsigned long long speid,
         unsigned long long argp,
         unsigned long long envp)
{
       printf("Hello world (0x%llx)\n",
              speid);
       return 0;
}
```

4. DESEMPENHO

A Tabela 1 mostra um resumo dos dados de desempenho obtidos em [3]. Para uma vasta gama de aplicações, o Cell BE possui desempenho igual ou significativamente melhor que um processador de propósito geral (GPP, General Purpose Processor). Para aplicações que podem tirar vantagem do pipeline SIMD das SPUs e do paralelismo no nivel de thread da PPU os resultados serão similares.

Código 2: Código para o PPE

```
#include <stdio.h>
#include <libspe.h>
#include <sys/wait.h>
extern spe_program_handle_t hello_spu;

int main (void)
{
       speid_t speid[8];
       int status[8];
       int i;
       for (i=0;i<8;i++)
              speid[i]  =  spe_create_thread
(0, &hello_spu, NULL, NULL, -1, 0);
       for (i=0;i<8;i++)
       {
              spe_wait(speid[i],
&status[i], 0);
              printf ("status = %d\n",
              WEXITSTATUS(status[i]));
       }
       return 0;
}
```

Tabela 1. Desempenho do Cell comparado a um processador de propósito geral

Algorithm	3.2 GHz GPP	3.2 GHz Cell	Cell Perf Advantage
Matrix Multiplication (S.P.)	24 Gflops (w/SIMD)	200 GFlops* (8SPEs)	8x
Linpack (S.P.)	16 GFlops (w/SIMD)	156 GFlops* (8SPEs)	9x
Linpack (D.P.): 1kx1k matrix	7.2 GFlops (IA32/SSE3)	9.67 GFLops* (8SPEs)	1.3x
Transform-light	170 MVPS (G5/VMX)	256 MVPS** (per SPE)	12x
TRE	1 fps (G5/VMX)	30 fps* (Cell)	30x
AES encryp. 128-bit key	1.03 Gbps	2.06Gbps** (per SPE)	16x
AES decryp. 128-bit key	1.04 Gbps	1.5Gbps** (per SPE)	11x
TDES	0.12 Gbps	0.16 Gbps** (per SPE)	10x
DES	0.43 Gbps	0.49 Gbps** (per SPE)	9x
SHA-1	0.85 Gbps	1.98 Gbps** (per SPE)	18x
mpeg2 decoder (CIF)	----	1267 fps* (per SPE)	--
mpeg2 decoder (SDTV)	354 fps (IA32)	365 fps** (per SPE)	8x
mpeg2 decoder (HDTV)	----	73 fps* (per SPE)	--

Notas: * Medidas por hardware ** Restultados de simulações

5. APLICAÇÕES

Nesta seção veremos algumas das possíveis aplicações do processador Cell BE.

5.1 Processamento de vídeo

Algumas empresas possuem planos de lançar adaptadores PCI-E baseados no Cell para decodificação de vídeo em tempo real.

5.2 Blade Server

Em 2007 a IBM lançou o Blade Server QS21 que gera 1,02 Giga FLOPS por watt, com picos de desempenho de aproximadamente 460 GFLOPS, o que o faz uma das plataformas mais eficientes em consumo de energia da atualidade.

5.3 Console de videogames

O console de videogame da Sony PlayStation 3 contém a primeira aplicação do Cell BE a entrar em produção, com clock de 3,2GHz e contendo sete dos oito SPEs operacionais, para aumentar o número de processadores aproveitados no processo de produção. Apenas seis destes sete SPEs são acessíveis aos desenvolvedores, sendo que um é reservado pelo sistema operacional.

5.4 Supercomputadores

O mais recente supercomputador da IBM, o IBM Roadrunner, é um híbrido processadores de propósito geral CISC Opteron e processadores Cell. Em junho de 2008 este sistema entrou como número 1 da lista Top 500 como o primeiro computador a rodar a velocidades de petaFLOPS, conseguindo sustentar uma velocidade de 1,026 petaFLOPS durante a execução do benchmark Linkpack.

Os supercomputadores que utilizam os processadores Cell também são eficientes no consumo de energia. Dentre as 10 primeiras posições da lista dos 500 supercomputadores mais eficientes em consumo de energia, as 7 primeiras são ocupadas por sistemas que utilizam processadores Cell [4].

5.5 Computação em Cluster

Clusters de consoles PlayStation 3 podem ser uma alternativa a sistemas mais caros e sofisticados baseados em Blade Servers, como mostrado em [2].

6. PowerXCell 8i

Em 2008 a IBM anunciou uma versão revisada do Cell BE chamada de PowerXCell 8i. O PowerXCell 8i suporta até 16GB de memória principal além de um aumento substancial no desempenho de operações de ponto-flutuante de precisão dupla alcançando um pico de 12,8GFLOPS por SPE e mais de 100GFLOPS utilizando todos os núcleos [8]. O supercomputador da IBM, Roadrunner, consiste em 12.240 processadores PowerXCell 8i, além de 6.562 processadores AMD Opteron.

7. REFERÊNCIAS

[1] Arevalo, A., Matinata, R. M., Pandian, M., Peri, E., Kurtis, R., Thomas, F. e Almond, C. 2008. Programming the Cell Broadband Engine™ Architecture: Examples and Best Practices
http://www.redbooks.ibm.com/redbooks/pdfs/sg247575.pdf

[2] Buttari, A., Luszczek, P., Kurzak, J., Dongarra, J., e Bosilca, G. 2007. A Rough Guide to Scientific Computing On the PlayStation 3
http://www.netlib.org/utk/people/JackDongarra/PAPERS/scop3.pdf

[3] Chen, T., Raghavan, R., Dale, J. e Iwata, E. Cell Broadband Engine Architecture and its first implementation
http://www.ibm.com/developerworks/power/library/pa-cellperf/?S_TACT=105AGX16&S_CMP=LP
19/06/2009

[4] The Green500 List
http://www.green500.org/lists/2008/11/list.php
25/09/2009

[5] IBM - Cell Broadband Engine Overview
http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/topic/com.ibm.iea.cbe/cbe/1.0/Overview/L1T1H1_02_CellOverview.pdf
25/06/2009

[6] IBM - The Cell project at IBM Research
http://www.research.ibm.com/cell/
20/06/2009

[7] IBM - Hands-on: The Hello World Cell Program, PPE vs SPE
http://www.cc.gatech.edu/~bader/CellProgramming.html
22/06/2009

[8] IBM - PowerXCell 8i processor product brief
http://www-03.ibm.com/technology/resources/technology_cell_pdf_PowerXCell_PB_7May2008_pub.pdf
25/06/2009

[9] Komornicki, Dr. A, Mullen-Schulz, G. e Landon, D. 2009. Roadrunner: Hardware and Software Overview.
http://www.redbooks.ibm.com/redpapers/pdfs/redp4477.pdf

[10] Mallinson, D. e DeLoura, M. 2005. CELL: A New Platform for Digital Entertainment
http://www.research.scea.com/research/html/CellGDC05/index.html
22/06/2009

[11] Sony - Synergistic Processor Unit Instruction Set Architecture, Version 1.2
http://cell.scei.co.jp/pdf/SPU_ISA_v12.pdf