Um olhar sobre a Arquitetura CELL

Trabalho desenvolvido
pelos alunos:

Ubiratan F. Soares
Ulisses F. Soares
Paulo Ricardo Chagas

São Carlos,
18 de novembro de 2008

Agenda para esta apresentação

- Objetivos
- Introdução à Arquitetura CELL
- Por dentro da Arquitetura CELL
- Benchmark : O CELL contra seus concorrentes
- Programando o CELL : Modelos e possibilidades
- O Futuro da Arquitetura
- Conclusões e Agradecimentos
- Referências

Objetivos

Objetivos dessa Apresentação

- Apresentar a Arquitetura CELL e seus aspectos do ponto de vista de inovação em hardware, traçando paralelos com arquiteturas concorrentes atualmente no mercado;
- Apresentar um Benchmark consistente para arquitetura, comparando o desempenho em relação à outras plataformas;
- Apresentar modelos de programação para a arquitetura, bem como uma introdução aos aspectos do desenvolvimento de software para o CELL;
- Debater se a arquitetura CELL é viável no “mainstream” ou é somente voltada para nichos específicos de mercado e apresentar futuras aplicações para a Arquitetura;

Introdução à Arquitetura CELL

Contexto Histórico

- Inicialmente, a arquitetura foi concebida para o novo console da Sony – Playstation 3 – em meados de 2000;
- Desenvolvido pelo consórcio STI (**S**ony – **T**oshiba – **I**BM) no período de 2000 à 2005[1];
- Custo de projeto não declarado, mas estimado não oficialmente casa de bilhões de dólares, antes do lançamento dos primeiro protótipo[2];

Objetivos Iniciais do STI Group

- Segundo Kahle [3], os objetivos do STI ao conceber o CELL eram:
  1) Melhoria de performance, especialmente no que condiz a aplicações de jogos e multimídia;
  2) Compromisso de tempo real com as aplicações do usuário e de rede;
  3) Aplicabilidade para uma grande variedade de plataformas, desde PDA's até servidores de alto desempenho;

Barreiras para a melhoria de Performance

- Dentre os principais desafios, destacaram-se:
  - ✓ Contornar o problema de “memory wall”[3], através de uma organização que permitisse que mais largura de banda de memória fosse utilizada de maneira efetiva;
  - ✓ Encontrar mecanismos que permitissem aumentar a eficiência do consumo de energia em paralelo ao aumento de performance;
  - ✓ Desenvolver uma microarquitetura que contornasse os problemas de rendimentos decrescentes associados aos aumentos da frequência de Clock do processador e profundidade de Pipeline;

Compromisso com o Usuário e a Rede

- Um dos objetivos do projeto CELL foi conceber um processador que fornecesse a melhor experiência possível para o usuário final, além da melhor resposta possível às interfaces de comunicação;
- Isso implica em um robusto suporte à aplicações de tempo real, em suporte a concorrência de sistemas operacionais *Real-Time* e *Non-Real-Time*, além de otimizações no tratamento de tarefas orientadas à comunicação, visando a escabilidade da arquitetura;

Aplicabilidade para diversas Plataformas

- Inicialmente, o projeto CELL foi concebido para atender as necessidades da próxima geração de sistemas de mídia e jogos;
- Contudo, a arquitetura foi projetada para ter vida muito além de sua primeira geração.
- Para isso, busca-se ampliar o alcance do uso do CELL Broadband Architecture, através de iniciativas como a adoção da arquitetura em plataformas cada vez mais distintas, envolvimento da comunidade open-source no desenvolvimento de software, dentre diversas outras;

Por dentro da Arquietura CELL

Desempenho com Simplicidade

- Nos últimos anos, verificou-se nos processadores o aumento das caches e da profundidade do pipeline de instruções, a incorporação de hardware para execução Out of Order (OOO), hardware robusto para previsão de desvio, dentre outras técnicas para alcançar novos patamares de desempenho;
- Todas essas técnicas porém, angariam complexidade, em especial no consumo de energia e na manutenção do aproveitamento do potencial de computação de cada CPU em função da adoção dessas técnicas;
- A Arquitetura CELL busca obter poder de processamento através de técnicas e soluções mais simples, juntamente a uma aboardagem RISC, diferente das abordagens híbridas usualmente utilizadas pelos CPUs comuns;

O que é a Arquitetura CELL ?

- Uma arquitetura de alta performance, puramente RISC, escalável, heterogênea, que traz uma série de rupturas com as abordagens dos atuais processadores visando o desempenho:
  - ✓ Preferência por maior largura de Banda de Memória em detrimento à latências de memória menores;
  - ✓ Abolição de hardware dedicado para execução de instruções OOO(Out-of-Order) ;
  - ✓ Hardware mais simples para previsão de desvio, com presença somente em um dos núcleos;

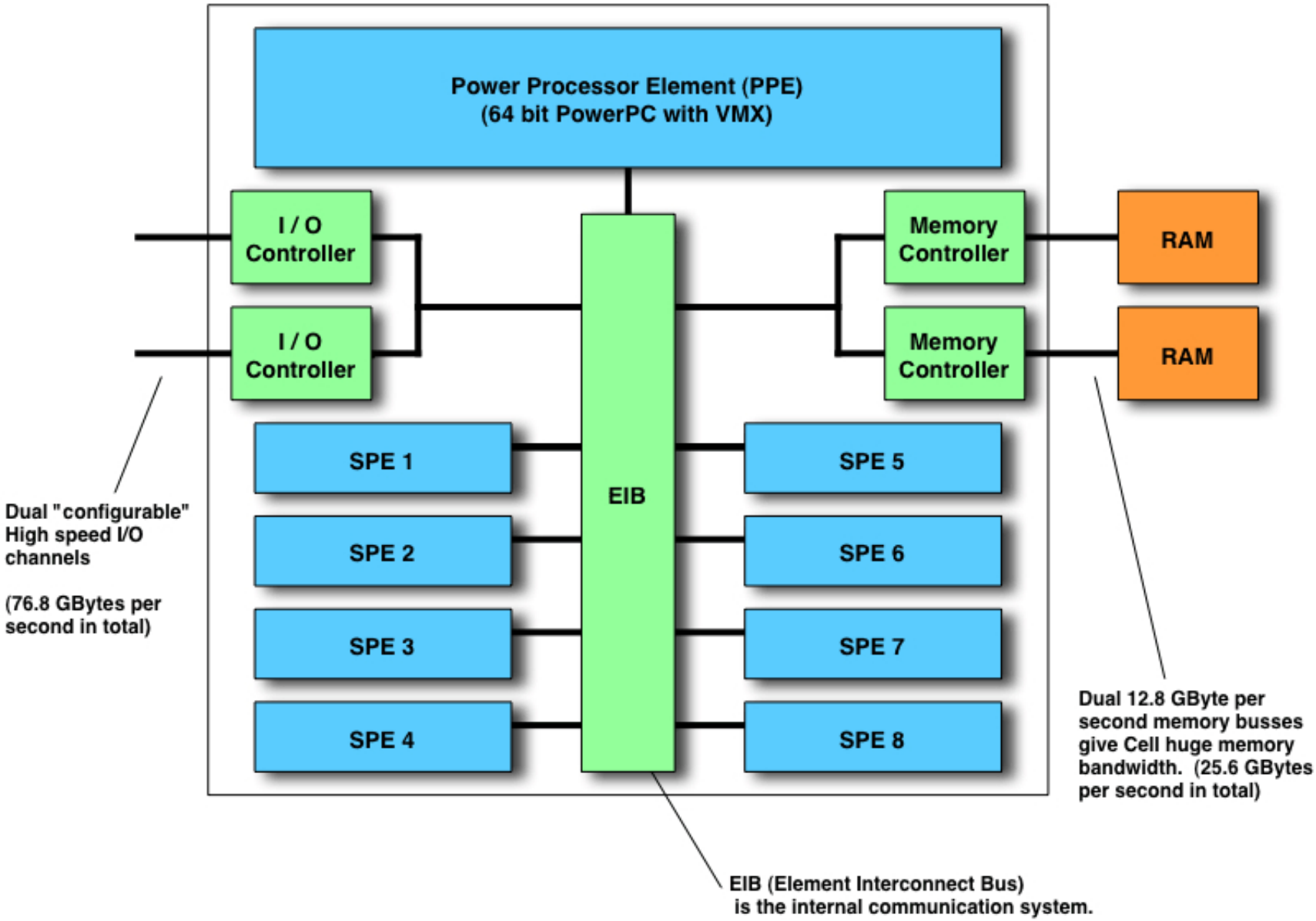
O que é a Arquitetura CELL?

- Adoção de armazenamento local de dados e instruções ao invés de memórias Cache maiores;
- Encurtamento do pipeline de instruções, associado ao aumento da frequência de Clock;
- Design para operar em altas frequências de Clock, com uma menor voltagem nos núcleos visando eficência no consumo de energia;

Especificações do Multiprocessador CELL

- O multiprocessador CELL é formado pelos seguintes elementos:
  - 1 Power Processor Element (PPE);
  - 8 Synergistic Processor Elements (SPEs);
  - Element Interconnect Bus (EIB);
  - 2 Controladores de memória Rambus XDR;
  - Broadband Interface Rambus FlexIO;
  - Frequência típica de operação @ 4 GHz

Diagrama de Blocos do Processador CELL



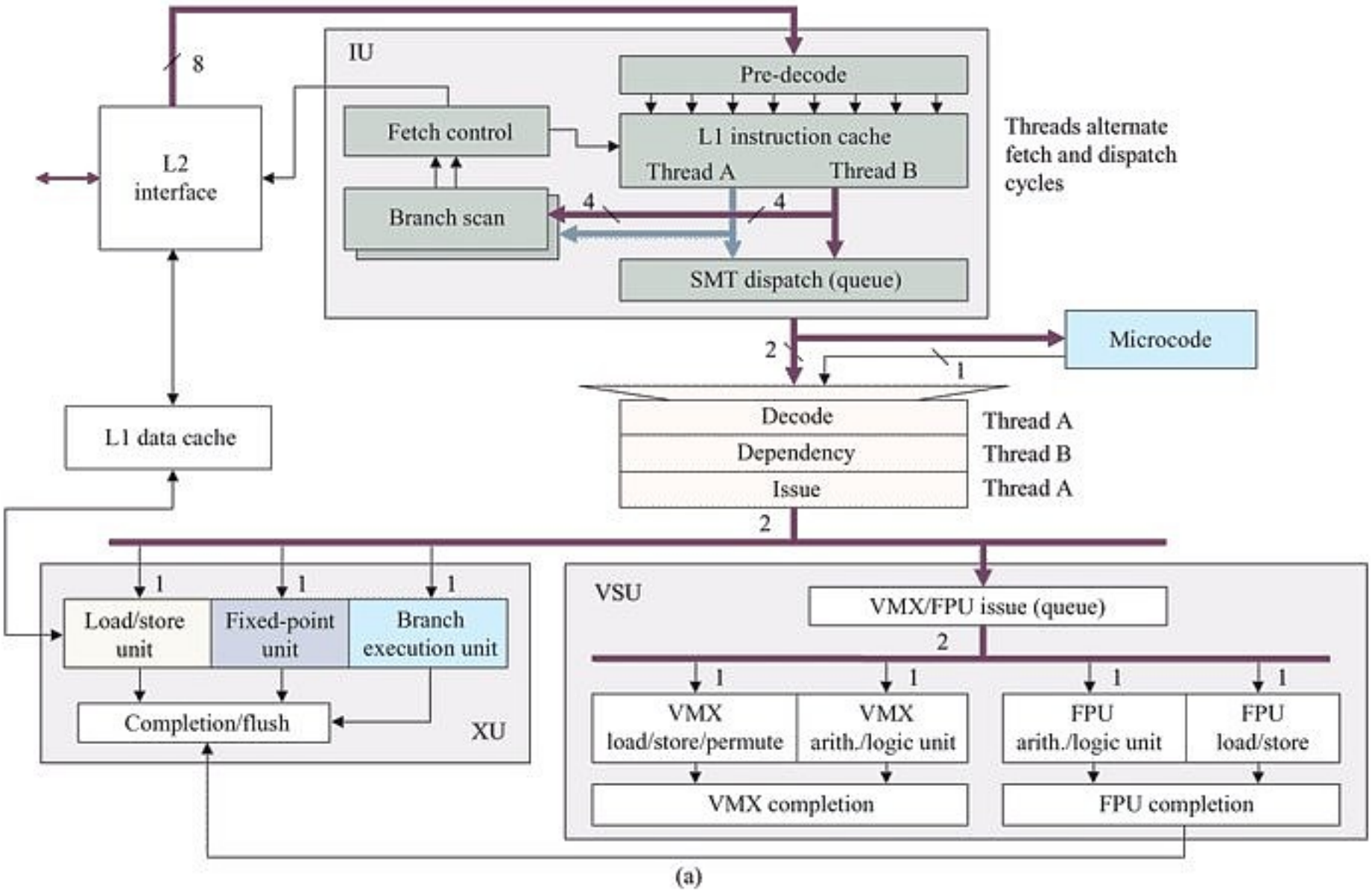
PPE – Power Processor Element

- Suporta as Arquiteturas POWER e PowerPC, o que possibilita um bom ponto de entrada para o desenvolvimento de software e compatibilidade com sistemas operacionais já existentes;
- Possui uma hierarquia de memória cache tradicional, com uma cache L1 de dados e instruções de 32KB, uma unidade de cache L2 de 512 KB;
- Capaz de chavear 2 threads de hardware (uma a cada ciclo) e executar duas instruções por ciclo de relógio, além de possuir interface de virtualização;
- Possui suporte a instruções vetoriais AltVec, presentes nos processadores PowerPC;

PPE – Power Processor Element

- O núcleo Power é subdividido em três unidades distintas:
  - IU (Instruction Unit);
  - VSU (Vector Scalar Unit) ;
  - XU (Execution Unit);
- Capaz de executar instruções de ponto fixo em somente 2 ciclos de relógio, e instruções de ponto flutuante em 7 ciclos de relógio;

PPE – Power Processor Element



(a)

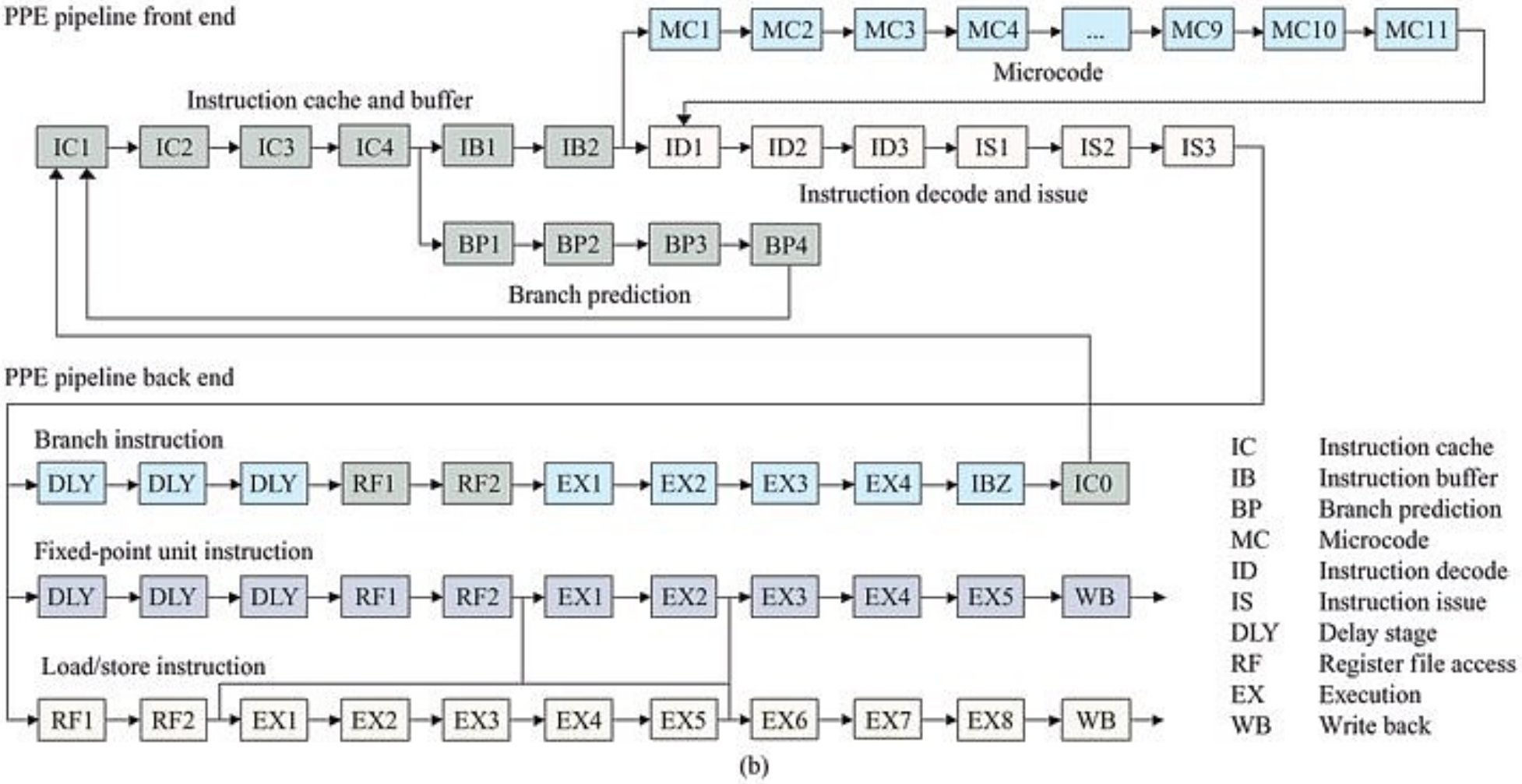
PPE – Instruction Unit (IU)

- A Unidade de Instrução é responsável pela busca de instruções, decodificação, desvio e despacho de instruções entre outras tarefas;
- Capaz de buscar 4 instruções por ciclo por thread, podendo despachar até duas instruções por ciclo;
- Quase todas as combinações dual-issued são possíveis, exceto duas instruções para a mesma unidade de execução, além de algumas instruções vetoriais[3];

PPE – XU e VSU

- A Unidade de Execução é responsável por todas as instruções de ponto fixo e todas as instruções do tipo Load/ Store;
- A Unidade de Vetor Escalar é responsável por todas as instruções de vetorização e de ponto flutuante;
- É compatível com o conjunto de instruções vetorias AltVec, sendo que há um banco de registradores de 128 bits por thread, e todas as instruções são SIMD em modalidades variáveis de 2 x 64-Bits a 128 x 1-Bit;

PPE – Pipeline de Instruções

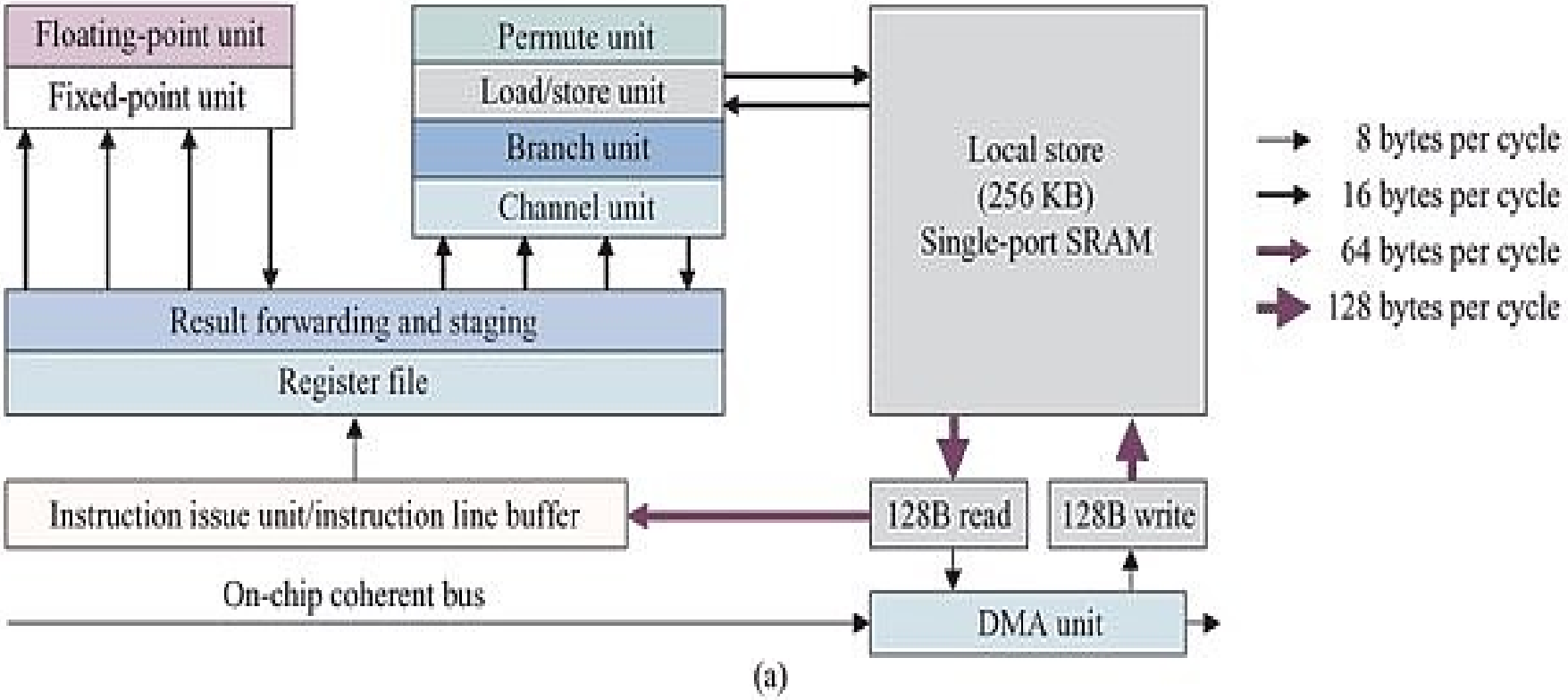


(b)

SPE – Synergistic Processing Element

- O SPE implementa um novo conjunto de instruções otimizado para grande poder computacional e alta performance em aplicações de mídia e aplicações computacionalmente intensas;
- Os SPEs podem ser vistos como “aceleradores de código”, com o elemento PPE distribuindo tarefas e coordenando a ação conjunta dos processadores;[2]
- Cada SPE é um processador de propósito geral, porém mais simples do que os convencionais por não possuirem cache e hardware para previsão de desvio;

SPE – Synergistic Processing Element



SPE – Synergistic Processing Element

- Cada SPE é formado por 4 componentes internos:
  - Execution Unit;
  - Instruction Issue Unit/Instruction Line Buffer;
  - Local Store;
  - DMA Unit;
- Instruções de ponto fixo executam em 2 ciclos; instruções de ponto flutuante simples em 6 ciclos. Instruções SIMD também são suportadas;

SPE - Execution Unit

- A Execution Unit do SPE é organizada em cima de um fluxo de dados de 128 bits;
- Há 128 registradores de 128 bits cada, e as instruções são SIMD variando nos modos
  - 2 x 64-Bits,
  - 4 x 32-Bits,
  - 8 x 16-Bits,
  - 16 x 8-Bits,
  - 128 x 1-Bit;
- É capaz de executar uma instrução de ponto-fixo/float e uma instrução de load/store/branch simultaneamente;

SPE - Local Store

- O SPE não possui memória cache!!!!
- Em sistemas multiprocessadores, o uso de memórias cache implica em custos adicionais para manter a coerência entre a memória do sistema e as caches dos elementos processadores; tal custo facilmente ofusca o desempenho obtido com o uso da hierarquia de memória cache em si;[4]
- Ao invés disso, cada SPE contém uma unidade de memória próprio denominada de Local Store, de 256KB;

SPE - Local Store

- A diferenças entre os Local Stores e as caches não residem somente na complexidade de se manter consistência e coerência de dados entre caches distribuídas!
- Os Local Stores podem manter sua alta taxa de transferências de dados durante rajadas maiores que as das caches atuais, mesmo que as caches possam oferecer picos de transferência mais altos;
- No intuito de minimizar o problema da ausência de hardware para branch predictions, os SPEs também fornecem instruções para o programador/ compilador para “advinhar” quando haverá um possível desvio, que são executadas por uma unidade dedicada!!!
- Dessa maneira, é possível diminuir a penalidade por desvios no pipeline de instruções;

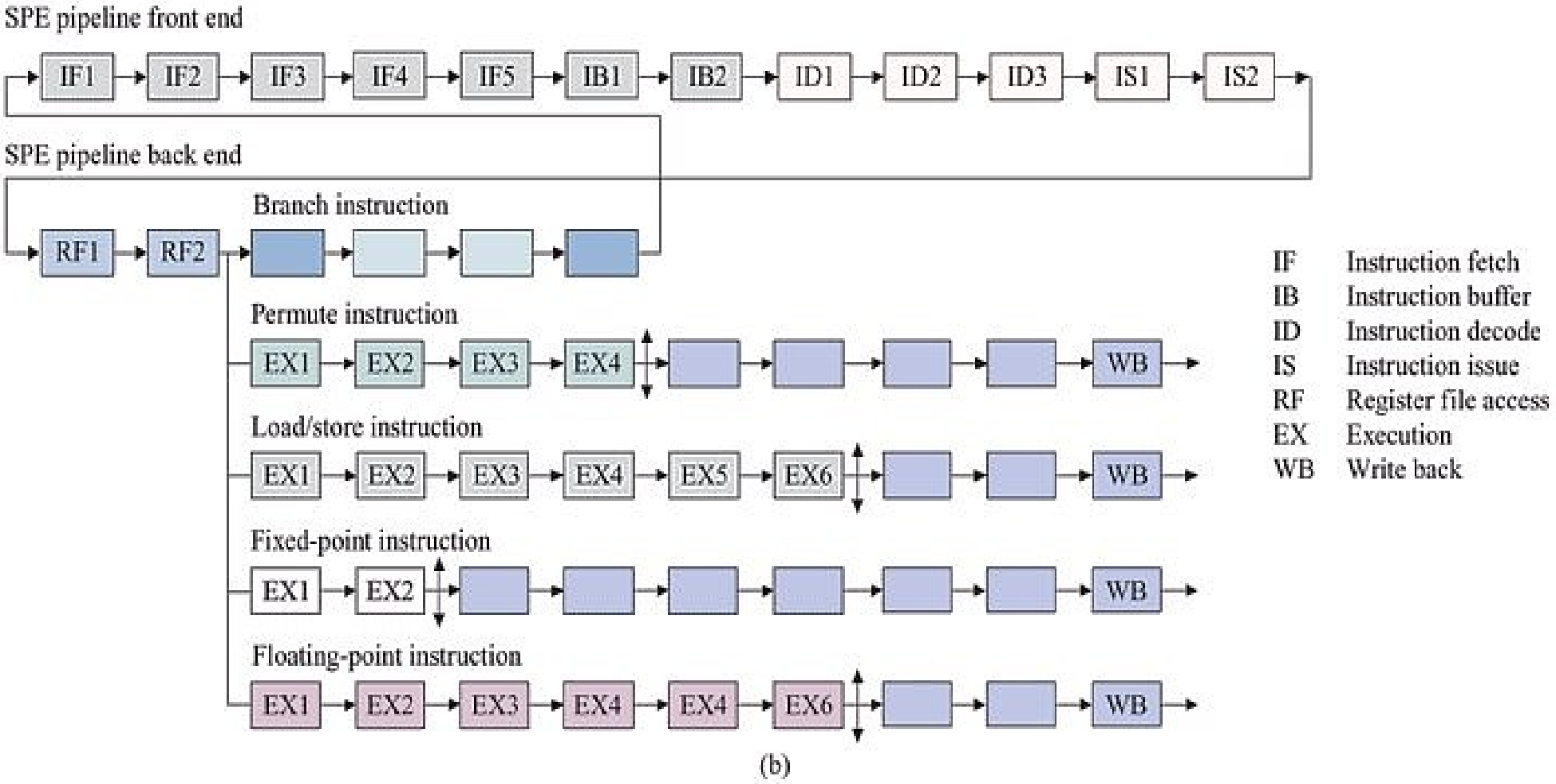
SPE – Local Store

- Nos SPEs, toda e qualquer transferência entre a memória do sistema e uma local store é feita através de uma unidade DMA, associada a cada SPE;
- Essa unidade de DMA assíncrona é programável por software; as instruções do programação DMA podem ser acumuladas, provendo acesso constante aos Local Stores a cada 8 ciclos de relógio[3];
- Adicionalmente cada SPE contém também mecanismos de proteção à memória, fornecidos por um MFC (Memory Flow Controller) associado uma MMU (Memory Management Unit);

SPE – Local Store

- O Local Store provê um excelente mecanismo para combater o problema do “Memory Wall”, uma vez podem ser armazenados dados e instruções de programa;
- Mesmo o custo adicional de alguns ciclos para preparar uma transação DMA será irrelevante perto do overhead do DMA em si; contudo, o ganho está na velocidade com que os Local Stores podem ser abastecidos com dados (0.5 Tb/s!!!);
- O Local Store, e os SPEs certamente representam um dos maiores desafios e ao mesmo tempo a maior fonte de desempenho para compiladores e programadores!

SPE – Pipeline de Instruções

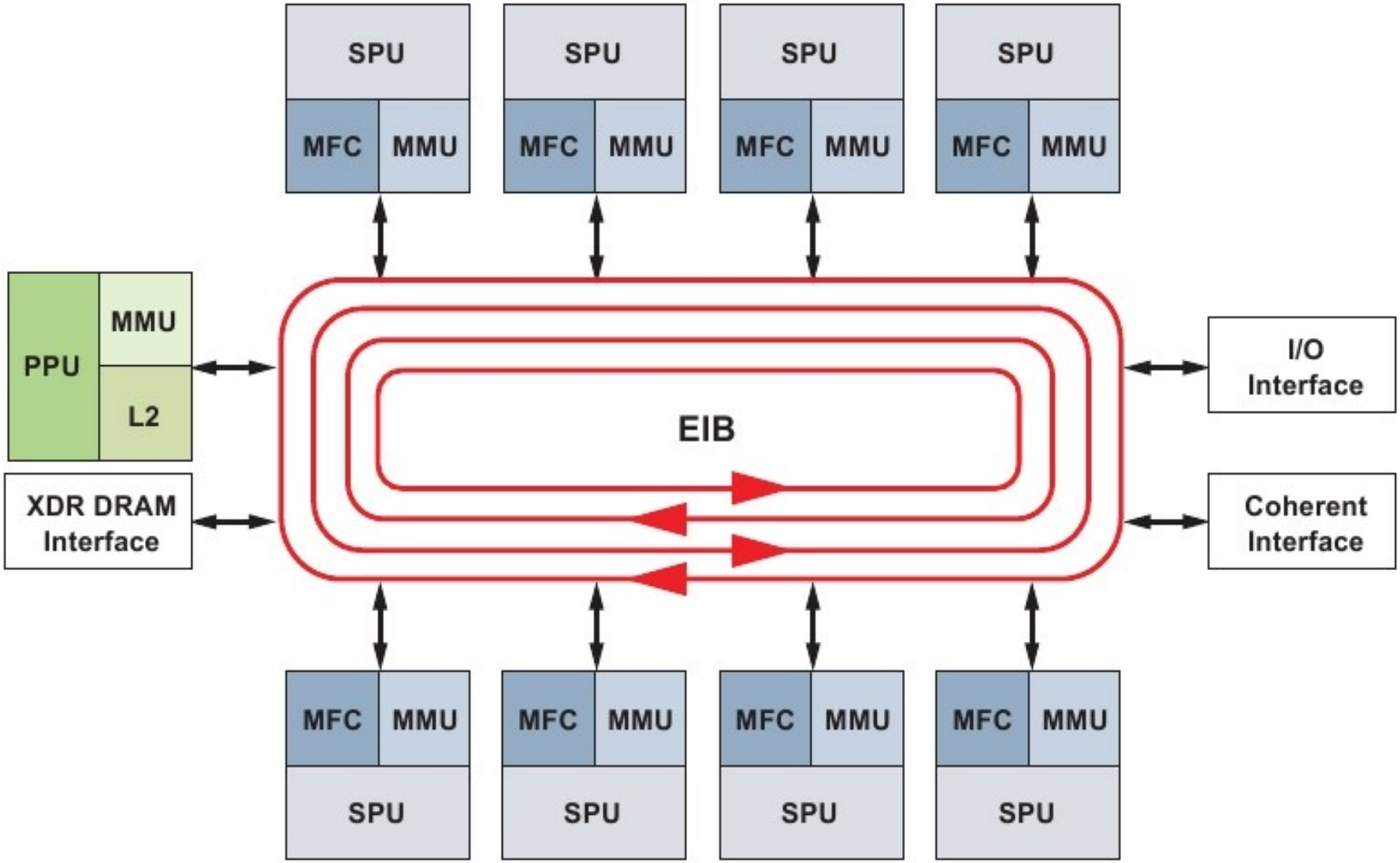


(b)

EIB – Element Interconnection BUS

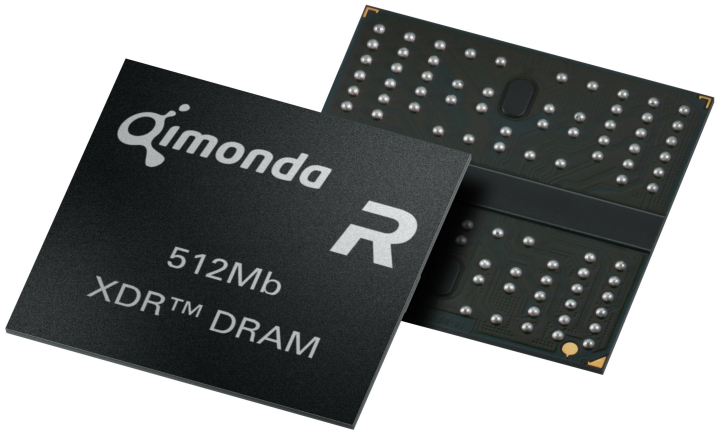
- Substituto ao barramento de 1024 bits do projeto original do CELL;
- Consiste em 4 anéis circulares (token rings) de 16 bytes cada, rodando a uma frequência de relógio equivalente à metade do Clock da CPU;
- Permite 3 transferências simultâneas, o que resulta em um pico teórico de transferência de dados de 384 Gigabytes/s!!!!
- Juntamente com o mecanismo de DMA, é o grande responsável por fornecer o fluxo de dados responsável pela alta performance visada pela arquitetura;[2]

EIB – Element Interconnection BUS



Rambus XDR – A memória para o CELL

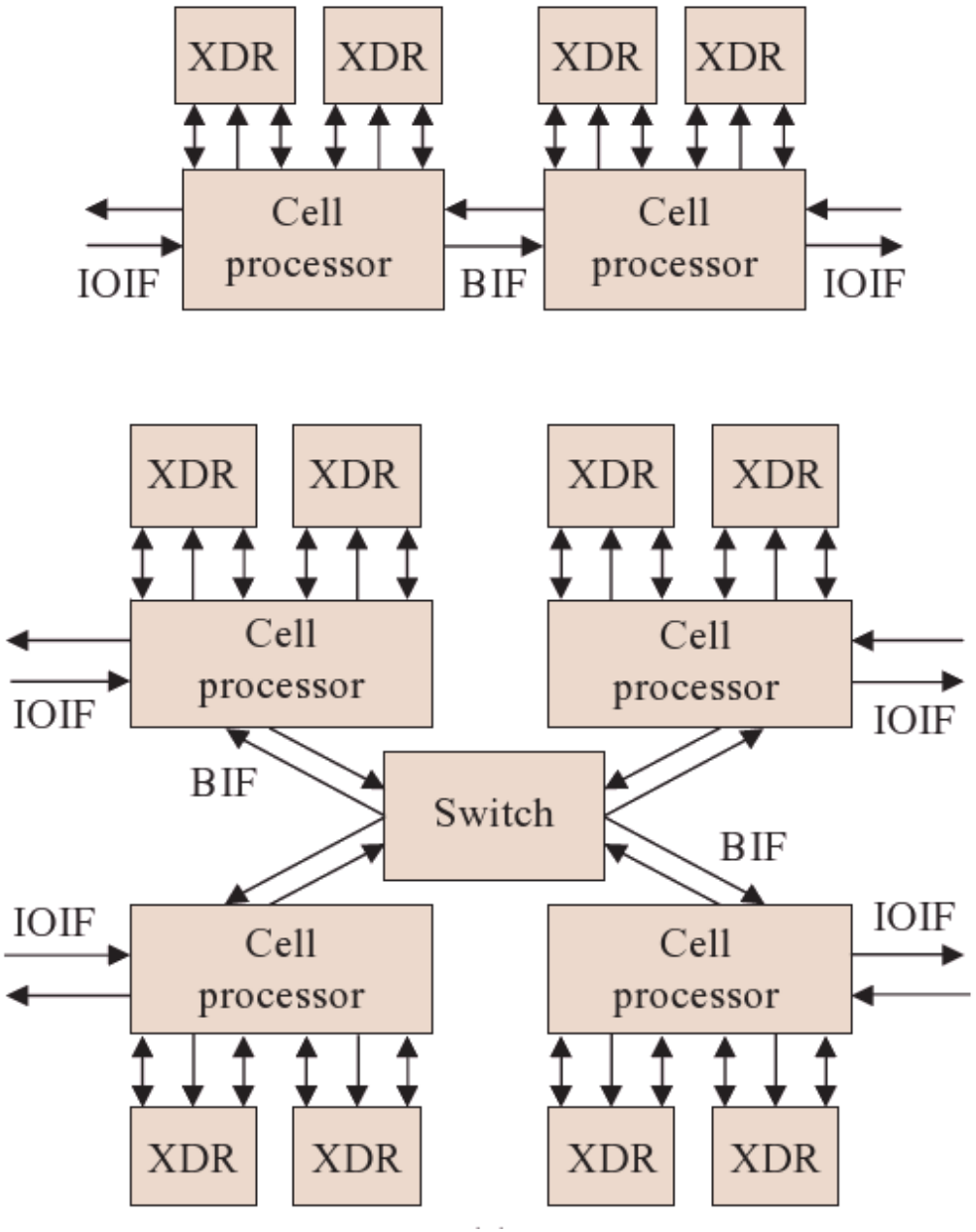
- Memória caracterizada por grande largura de banda na transferência de dados;
- Clock inicial de 400 Mhz, com expansão para até 1066 Mhz no futuro;
- A arquitetura CELL adota chips XDR que oferecem 12.6 GB/s de dados em cada canal de 32 bits, e portanto, largura de banda nominal total de 25.6 GB/s;



A BroadBand Interface (BFI)

- Barramento I/O de alto desempenho e alta largura de banda, baseado na tecnologia Rambus RRAC;
- Permite varias configurações, como um único chip com duas interfaces I/O, ou uma organização de processador duplo-chip coerente (“glueless”) que não requer switch para comunicação;[3]
- Pode ser utilizada em configurações mais amplas como em canais de comunicação inter-CELLs interligados por um switch.
- A BFI, juntamente com o processamento heterogêneo PPE/SPE são os grandes pontos fortes referentes à escalabilidade da arquitetura;

A BroadBand Interface



Benchmark :
O CELL contra seus concorrentes

- No intuito de avaliar a arquitetura, foi escolhido um benchmark que extraísse o máximo do potencial dos processadores envolvidos;
- O teste escolhido está descrito no artigo "*Lattice Boltzman Simulation Optimaztion on Leading Multicore Plataforms*"[5], publicado em abril de 2008 por pesquisadores da Universidade de Berkeley e escolhido como a melhor publicação voltada para aplicações pelo IPDPS;

Introdução

- O benchmark consiste em implementar o método LBMHD – um algoritmo para simulação de turbulência em meios magneto-hidro-dinâmicos homogêneos, isotrópicos e dissipativos;
- Apesar da técnica LBMHD ser numericamente intensa, a performance obtida nas implementações em geral tende a ser pobre, dada a complexidade das estruturas de dados envolvidas e dos padrões de memórias utilizados;

Introdução

- A implementação do método LBMHD foi testada em diversos hardwares multicores diferentes, sendo que o método foi ajustado em cada caso para se extrair o desempenho máximo de cada plataforma;
- As plataformas testadas são descritas a seguir:

Benchmark – Intel Itanium2

- Processador VLIW In-Order;
- Fetch de 6 instruções por ciclo;
- Pico de processamento de instruções de ponto-flutuante(dupla precisão) de 5.2 GFlops/s;
- Plataforma: **Madson3M**, dual-socket, clock @1.3GHz, dotada de cache de dados L1 de 256kb(x2), 3 Mb de cache L2(x2), com 8.5 Gb/s de largura de banda, memória do sistema de 4Gb;

Benchmark – Intel CoreQuad Clovertown

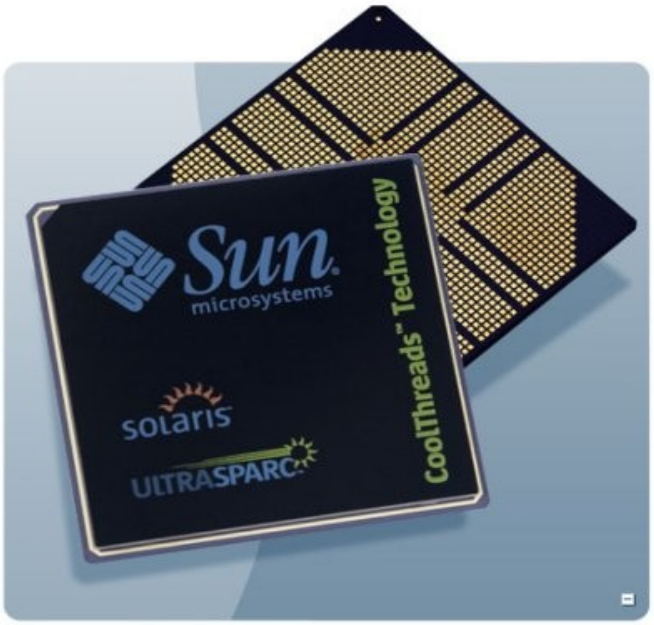
- Composto de 2 chips XeonDuo encapsulados em um único módulo multichip(MCM);
- Cada chip é baseado na arquitetura Core2Duo, buscando e decodificando 4 instruções por ciclo;
- Pico de execução DP de 9.3 Gflop/s por chip;
- Plataforma : **Dell PowerEdge 1950,** dual -socket, 32 Kb L1(x4), 4Mb L2(x4), clock @ 2.33 Ghz, largura de banda de 21.33Gb/s(leitura) e 10.66Gb/s(escrita), memória total de16Gb;

Benchmark – AMD Opteron X2

- Capaz de buscar e decodificar três instruções x86 por ciclo, além de operar operações de ponto-flutuante em dupla precisão em um pico de 4.4 Gflop/s;
- Plataforma : **SunFire X2200 M2,** dual-socket @ 2.2 GHz, tecnologia HT, 64Kb L1(x4), 1Mb L2 (x4), largura de banda de 21.33 Gb/s em dois canais DDR2-667, memória total de 16 Gb;

Benchmark – Sun Niagara2

- Arquitetura UltraSparc, composto de 8 núcleos, cada um capaz de chavear 8 threads de hardware!!
- Unidade de ponto flutuante(FPU) compartilhada entre os 8 cores, com pico de execução de DP's de 1.40 GFlops/core ;
- Plataforma : **Sun UltraSparc T5120**, single socket @ 1.4 GHz, 8Kb L1(x8), 4Mb L2(x1); largura de banda de 42.66 Gb/s(leitura), memória total compatível de 64 Gb;



Benchmark – STI CELL

- Plataforma utilizada :
- **QS20 Blade**, dual-socket, clock @ 3.2 GHz, 2 Gb de memória total (1Gb/socket), largura de banda de 51.2 Gb/s, DMA, SIMD, processamento DP a 1.8 Gflop/s para cada SPE, ;

Observações sobre as implementações

- A estrutura de dados básica usada na implementação é dada ao lado, com cada ponteiro associado a uma malha de tamanho N^3, que corresponde a uma ***lattice*** do espaço tridimensional;
- Dessa maneira, a entrada N determina o esforço computacional envolvido e a precisão do resultado;

```
Struct{
        //Macroscopical Quantities

        double  *Density;
        double  *Momentum;
        double  *Magnetic;

        // Distribuitions

        double *MomentumDist;
        double *MagneticDist;

}
```

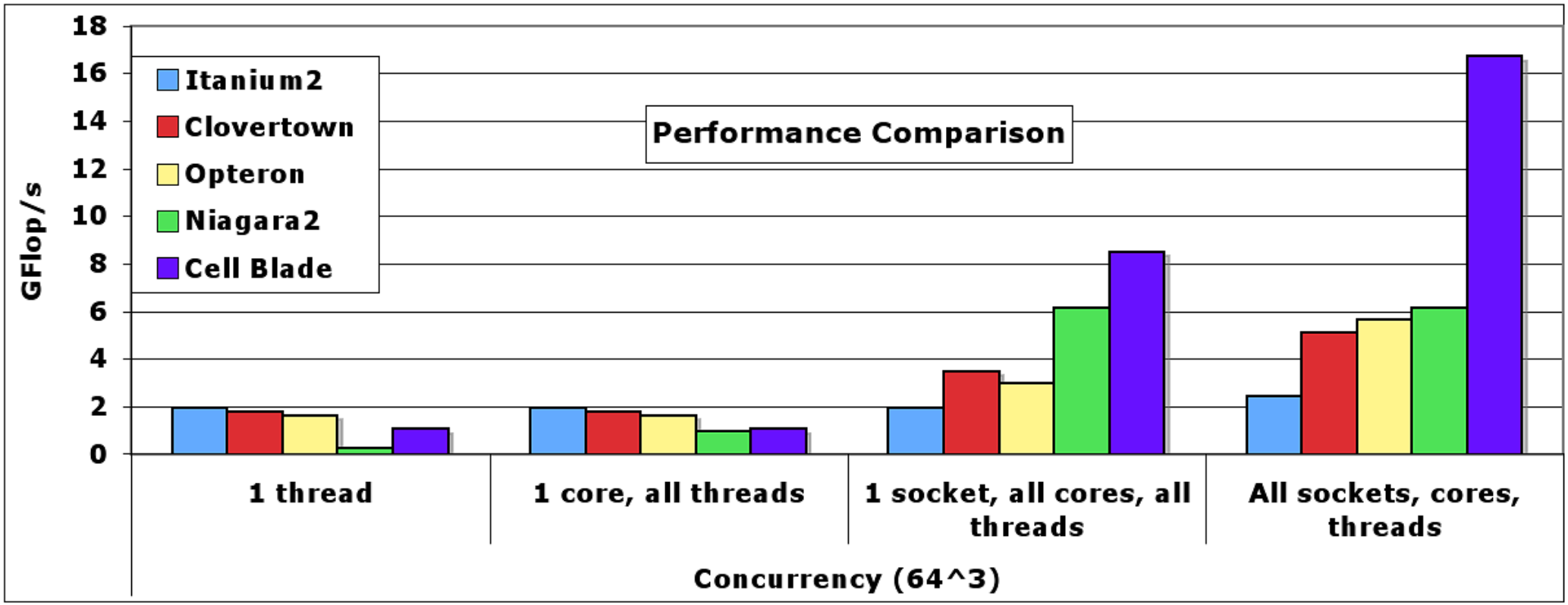
Observações sobre as implementações:

- Para otimizar o método LBMHD entre as diversas plataformas, foi criado um gerador de código PERL , capaz de produzir códigos C “auto-tunados”, multithreaded de acordo com técnicas que visam o aumento de performance em cada caso:
  - Bloqueio de TLB,
  - Reordenação de instruções,
  - “SIMDzação”;
- O paralelismo por threads foi implementado segundo o padrão POSIX(pthreads);
- Apenas na plataforma CELL foi utilizada a biblioteca libspe1.0, o acarretou em trabalho extra na codificação;

Tabela Comparativa e Dados Adicionais

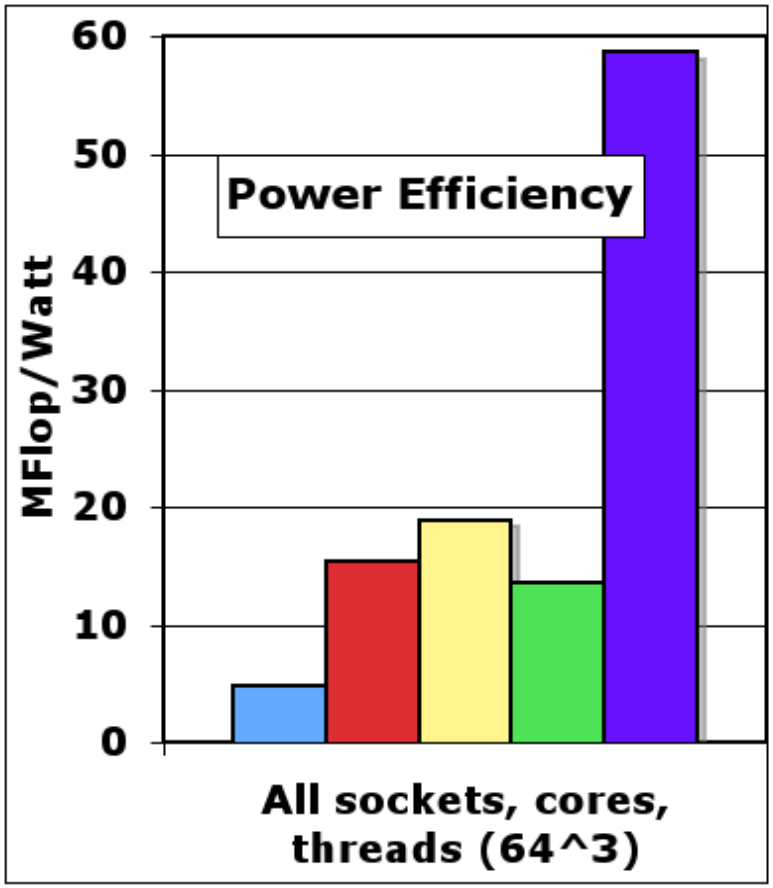
PLATAFORMA	Itanium2	Clovertown	Opteron X2	Niagara2	CELL
Tipo	VLIW	Superscalar(OOO)	Superscalar (OOO)	Multi Thread	Multi Thread SIMD
Sockets	2	2	2	1	2
Cores / Socket	2	4	2	8	8 (+1)
Clock(GHz)	1.3	2.33	2.2	1.4	3.2
DP (Gflop/s)	5.2	9.3	4.4	1.4	1.8(SPE)
Cache L1 / Core	256 Kb	32 Kb	64 Kb	8 Kb	* / 32Kb
Cache L2	3Mb(x2)	4Mb(x4)	1Mb(x4)	4 Mb(shared)	*/ 512Kb
Bandwidth (Gb/s)	8.5	21.33(r) / 10.66(w)	21.33	42.66(r) / 21.33(w)	51.2 (25.6 x2)
Threading	pthreads	pthreads	pthreads	pthreads	Libspe1.0
Compilador	Icc9.1	Icc 10.0	Gcc 4.1.2	Gcc 4.0.4	Xlc 8.2

Resultados – Vazão Aferida em GFlops/s



Resultados – Consumo de energia

- Ainda mais impressionante é o consumo de energia aferido, que indica uma relação próxima de 1/10 entre CELL e Itanium2 e quase 1/4 entre o CELL Blade e a plataforma Intel Clovertown!



Benchmark – Conclusões

- Apesar de um processamento mais “pobre” dos SPE's em operações de ponto flutuante em comparação aos processadores x86, a arquitetura CELL mostra-se superior à todas as demais concorrentes quando o cenário envolve todos os núcleos do processador;
- O benchmark mostra também a superioridade uma arquitetura heterôgenea baseada em um projeto simples porém consistente, em relação a uma arquitetura homogênea improvisada por uma promessa desempenho, sem compromisso real de escalabilidade;

Benchmark – Conclusões

- A escabilidade que é visada através da BIF pode ser de fato comprovada nesse benchmark;
- Os autores citam que o grande fator para o desempenho díspar da arquitetura CELL é o controle explícito por software da transferência entre os Local Stores dos SPEs e a memória do sistema;
- Contudo, confirma-se o fato de que o alto desempenho não vém sem um trabalho extra de programação, uma vez que não há pleno suporte dos compiladores para as otimizações abordadas e os ambientes para desenvolvimento de software ainda são precoces;

Programando o CELL : Modelos e Possibilidades

Modelos de Programação e Programabilidade

- Kahle[3] ressalta, conforme já visto, que o grande desafio para os programadores/compiladores é fato dos SPE possuírem Local Stores e que o software deve controlar as transações entre esse armazenamento e a memória do sistema;
- Modelos para software que visem o desempenho devem lídar com esse fato, bem como a natureza SIMD de determinados fluxos de dados;
- Adicionalmente, deve-se lembrar que cada SPE é single-thread, permitindo somente um contexto de execução : dessa maneira, a thread em execução ou será do usuário ou do núcleo do Sistema Operacional;

Modelos de Programação e Programabilidade

- A rica variedade de possibilidades para comunicação entre núcleos, assim como as facilidades para a movimentação de dados resultam em diversos modelos de programação para o CELL que visam extrair performance;
- O suporte a esses modelos em uma linguagem de programação de alto nível (atualmente C e Fortran) foi uma das metas no projeto da arquitetura;

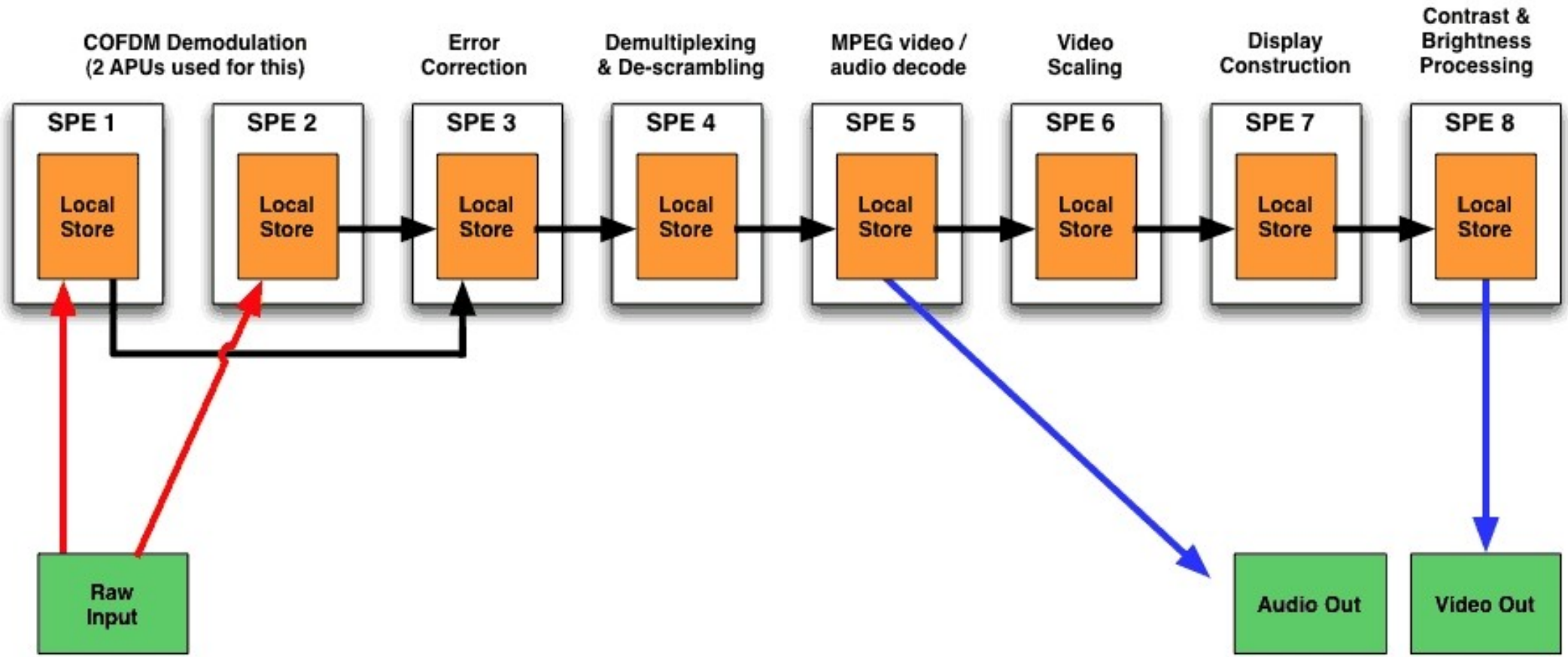
O Modelo “Function Offload”

- Modelo mais simples para programação do CELL;
- Os SPEs são utilizados como aceleradores para certos tipos de funções, consideradas críticas;
- Ao invés de sobrecarregar o elemento PPE com a execução de certas funções de biblioteca, essas funções são descarregadas para um ou mais SPEs, através de desmembramento da biblioteca dado o fluxo de execução no PPE ou SPE, seguido otimização e recompilação desses pedaços visando os SPEs;
- O maior desafio para um compilador que procure adotar esse modelo é identificar quais funções são específicas de serem ou não descarregadas para os SPEs no processo de compilação;

O Modelo Streaming

- Um dos modelos mais fortes para aplicações multimídia;
- Consiste em gerar um processamento em fluxo, com cada SPE atuando como verdadeiro pipeline serial ou paralelo para um determinado segmento do fluxo de dados;
- Nesse modelo, em geral o elemento PPE atua como o controlador de fluxo, enquanto os SPES atuam como os processadores do fluxo de dados;

O Modelo Streaming – Um exemplo



Programação – Aspectos Adicionais

- Enquanto não há um bom suporte dos compiladores na extração de desempenho da arquitetura, essa tarefa fica a cargo do programador;
- O CELL não irá “magicamente” acelerar nenhuma aplicação; o desempenho apenas vem, por hora, com um considerável esforço adicional do programador no desenvolvimento;
- Eichenberger[] destaca que futuros compiladores otimizados para a arquitetura CELL devem fazer uso adequado de uma série de técnicas, como otimizações de desvio baseadas nas instruções especiais dos SPEs, enfoque em modelos de programação, extração eficiente de paralelização SIMD e abstração para modelo multiprocessado com memória compartilhada;

O Futuro da Arquitetura

É o CELL nicho ou “mainstream”???

- A reposta mais conveniente para essa pergunta é : “*A resposta é para hoje ou amanhã?*”;
- Definitivamente, a arquitetura tem todos os requisitos para migrar das aplicações multimídia atuais para aplicações de propósito mais geral;
- Sendo uma arquitetura nova e ainda sem o suporte adequado para o desenvolvimento de aplicações, a diversificação dos segmentos que podem se beneficiar do poder do CELL depende dos resultados obtidos com as aplicações atuais, além do desenvolvimento de toda uma nova tecnologia de compiladores e ferramentas para auxiliar no desenvolvimento de software;
- Outros fatores determinarão um maior interesse pela arquitetura, como a mudança para tecnologia de encapsulamento a 45nm, prevista para 2009, que trará uma terceira geração de processadores CELL 40% mais eficientes no consumo de energia e mais baratos para o consumidor final;

Suporte de Sistemas Operacionais

- Sendo o elemento PPE compatível com a arquitetura PowerPC, ao menos duas distribuições Linux estão disponíveis para serem utilizadas como base para o desenvolvimento : Fedora nas plataformas QSBlade e YellowDog (Fedora Based) para PS3;
- O YellowDog é a única distribuição Linux incluída por default no CELL SDKv3.0;[1]

A Busca pela Aplicabilidade

- A intenção do consórcio STI é cada vez mais ampliar o uso da arquitetura para as mais diversas aplicações:
- Para 2009, Toshiba lança no mercado o notebook Qosmio g55-q802, que inclue um hardware CELL dedicado para processamento de vídeo digital ;
- IBM, por sua vez, possue sua plataforma QSBlade e prevê para o futuro seu primeiro Mainframe baseado na arquitetura CELL;

Pontos Críticos para um futuro Desktop CELL

- A viabilidade comercial de qualquer hardware voltado para computadores pessoais está atrelada à compatibilidade com os Sistemas Windows (fato);
- Esse ponto é determinante para a adoção da arquitetura como opção para PC's Desktops;
- As outras empresas do grupo STI – Sony e Toshiba – utilizam Sistemas Windows em suas linhas de computadores (notebooks), o que torna a adoção do CELL nesse produtos um ponto delicado na coesão do consórcio;
- Uma futura expectativa para computadores pessoais CELL está no fim da aventura “Intel Inside” da Apple - com contrato previsto para o fim de 2010 - e uma possível reaproximação IBM/ Apple, dada a experiência anterior dessa na plataforma PowerPC;

Arquitetura CELL – Futuros Horizontes

- Com o advento da terceira e quarta gerações do CELL nas tecnologias de 45nm e 32nm, a proliferação de hardware embarcado CELL-based tende a crescer potencialmente!
- A título de exemplo, um acordo entre a IBM e a empresa Mercury Computer Systems garante a aplicação futura do CELL em sistemas embarcados diversos, como processadores de imagens médicas, equipamento militares, sensores sismiológicos, telecomunicações, dentre outros;
- Sony e Toshiba, por sua vez, tendem também a embarcar hardware Cell em seus produtos tradicionais, como TV's de alta definição, Microsystems, Máquinas Fotográficas, Celulares, aparelhos Blue-ray, dentre diversos outros;
- Para o futuro, podemos esperar ainda mais aplicações compatíveis com os recursos oferecidos pela arquitetura CELL;

Conclusões e Agradecimentos

Os Resultados Extraídos desse Trabalho

- Dentre os principais resultados obtidos com esse trabalho, destacam-se:
  - ✓ O aprendizado de que a arquitetura CELL não é voltada somente para nichos de mercado e terá vida longa após sua primeira geração, dados seus atributos;
  - ✓ A verificação de que arquiteturas tipo RISC tendem a ser superiores à arquiteturas CISC ou híbridas, e que abordagens mais simples, inteligentes e inovadoras podem ser a saída quando as técnicas tradicionais deixam a desejar na relação complexidade/performance dados os gargalos tecnológicos;
  - ✓ A verificação prática dos fundamentos teóricos adquiridos nas disciplinas de Organzição e Arquitetura de Computadores em um contexto absolutamente atual;

Considerações Finais

- Uma constante na Ciência da Computação é sempre "evolução sobre revolução", no sentido de que novas características são adicionadas à práticas consolidadas e tidas como eficientes, em uma busca incremental e constante por melhor performance;
- A arquitetura CELL representa uma revolução, no sentido em que rompe com muitas das principais técnicas e tendências para ganho de performance adotadas nas arquiteturas atuais. Para alguns, as consequências desse rompimento representam uma ameaça; para outros, uma oportunidade.
- *"O assunto mais importante do mundo pode ser simplificado até ao ponto em que todos possam apreciá-lo e compreendê-lo. Isso é - ou deveria ser - a mais elevada forma de arte" (Charles Darwin)*

Agradecimentos

- Gostaríamos de agradecer :
  - Ao professores Eduardo Simões, Fabrício Simeoni e Paulo Sérgio Lopes pelos esclarecimentos e assistência na confecção desse trabalho;
  - Aos alunos Ricardo, Luís Paulo, Valter, Rafael e William que gentilmente cederam o tema dessa apresentação para nosso grupo;
  - Aos presentes que cederam seu tempo e paciência na apreciação desse trabalho;

Referências

Referências

- www.wikipedia.org;
- **Blachford, Nicholas** – *CELL Architecture Explained* http://www.blachford.info/computer/Cell/Cell0_v2.html(2005);
- **Kahle, J.A.;Day, M.N; Jonhs, C.R; Shippy, D.** – *Introduction to the Cell Multiprocessor* - IBM Journal of Research and Devolpement, vol 49(2005);
- **Tanenbaum, A.S.** - *Distributed Systems*(2004);
- **Williams, Samuel; Carter, Jonathan; Oliker Leonard; Yelick, Katherine** – *Lattice Boltzmann Simulation Optimization on Leading Multicore Plataforms* - IDPDS Papers(2008);
- **Tanenbaum, A. S.** - *Structured Computer Organization* (2002);
- **Stallings, William** – *Computer Architecture and Organization* (2002);

Referências

- *The Cell Project at IBM Research*
  http://www.research.ibm.com/cell/
- *The IBM Cell Broadband Engine Tecnology*
  http://www-03.ibm.com/technology/cell/
- **Preto, F.A.; Maranesi, L.; Santos, T. C.** - *O processador Cell* – UNICAMP(2006);
- Sony Computer Entertainment Inc. - *Cell Broadband EngineTM Architecture, Version 1.0* (2005);
- **Eichenberger, A; O'Brien K; Wu, P.; et al** – *Optimizing Compiler for a Cell Processor* (2005);

Dúvidas????

- Essa apresentação, assim como algumas das referências estarão em breve disponíveis em:
- www.icmc.usp.br/~ubiratan
- Maiores esclarecimentos, email-us!!!

OBRIGADO!!!!