IBM Systems and Technology Group

Tutorial

Hardware and Software Architectures for the CELL BROADBAND ENGINE processor

Michael Day, Peter Hofstee
IBM Systems & Technology Group, Austin, Texas

CODES+ISSS Conference, September 2005

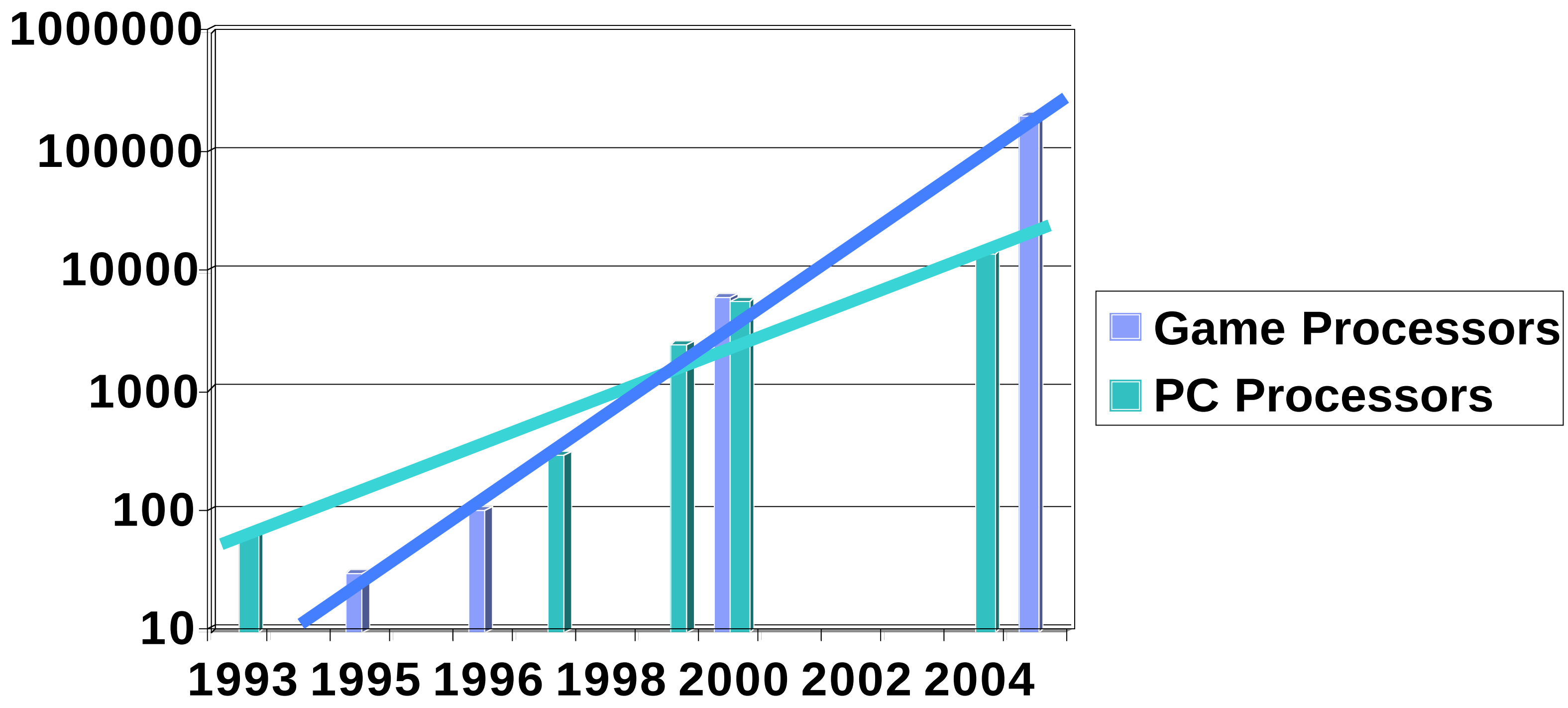
Agenda

- Trends in Processors and Systems
- Introduction to Cell
- Challenges to processor performance
- Cell Broadband Engine Architecture (CBEA)
- Cell Broadband Engine (CBE) Processor Overview
- Cell Programming Models
- Prototype Software Environment
- TRE Demo

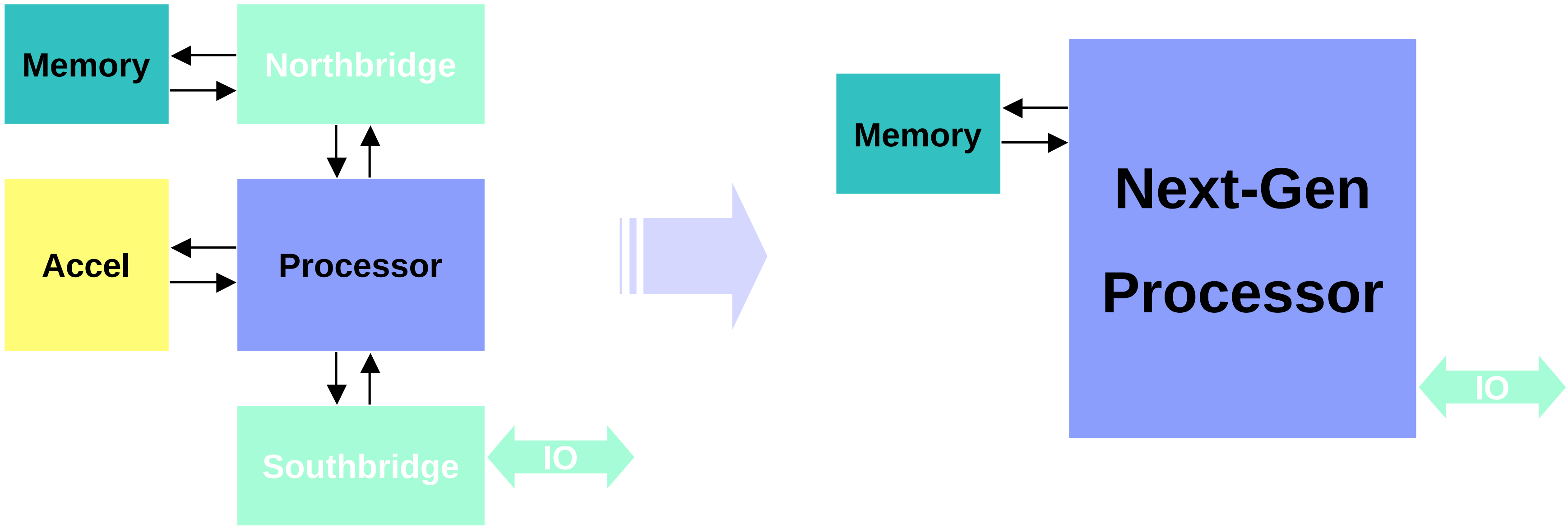
Trends in Microprocessors and Systems

Processor Performance over Time
(Game processors take the lead on media performance)

Flops (SP)



System Trends toward Integration



- Implied loss of system configuration flexibility
- Must compensate with generality of acceleration function to maintain market size.

Motivation: Cell Goals

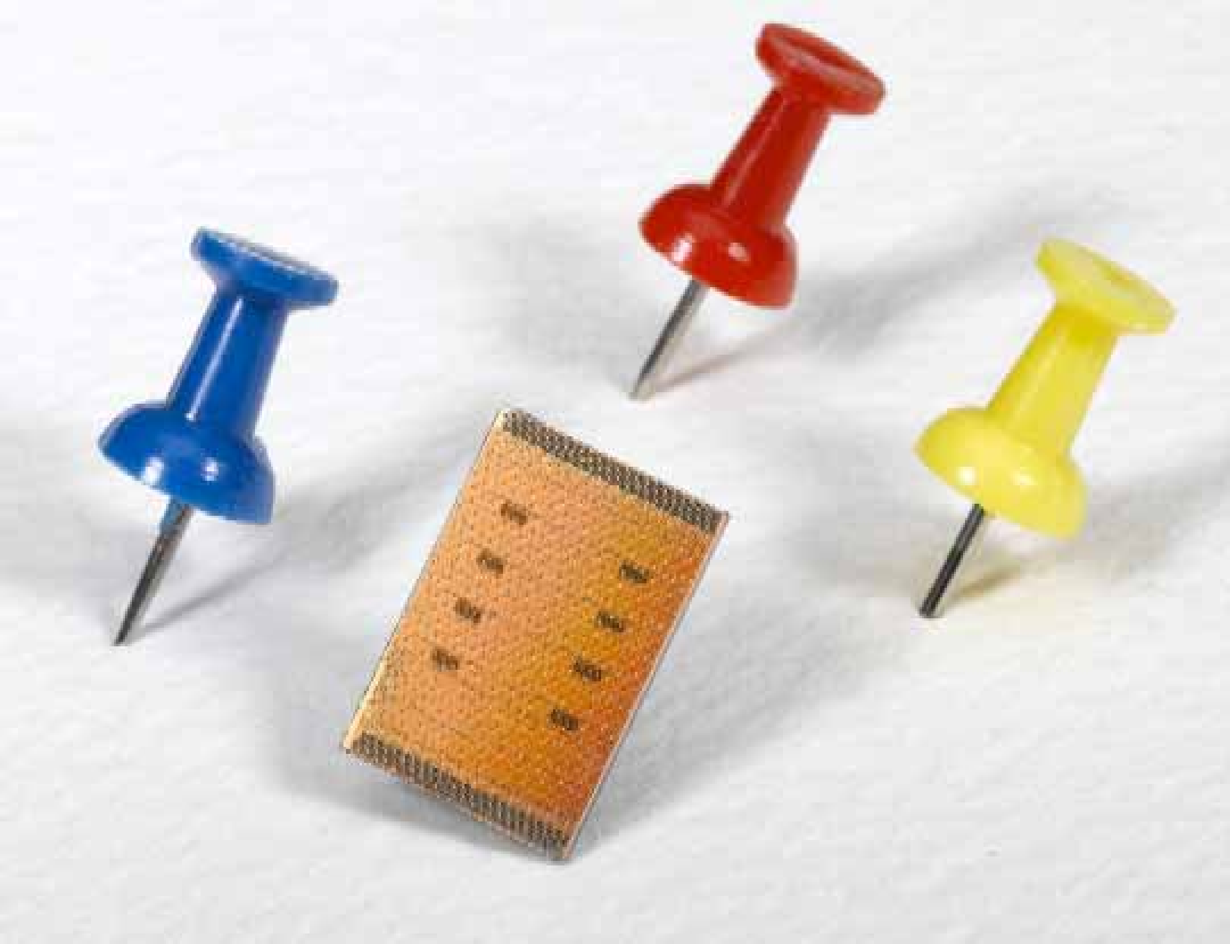
- Outstanding performance, especially on game/multimedia applications.
  - Challenges: Memory Wall, Power Wall, Frequency Wall
- Real time responsiveness to the user and the network.
  - Challenges: Real-time in an SMP environment, Security
- Applicable to a wide range of platforms.
  - Challenge: Maintain programmability while increasing performance
- Support an introduction in 2005/6.
  - Challenge: Structure innovation such that 5yr. schedule can be met

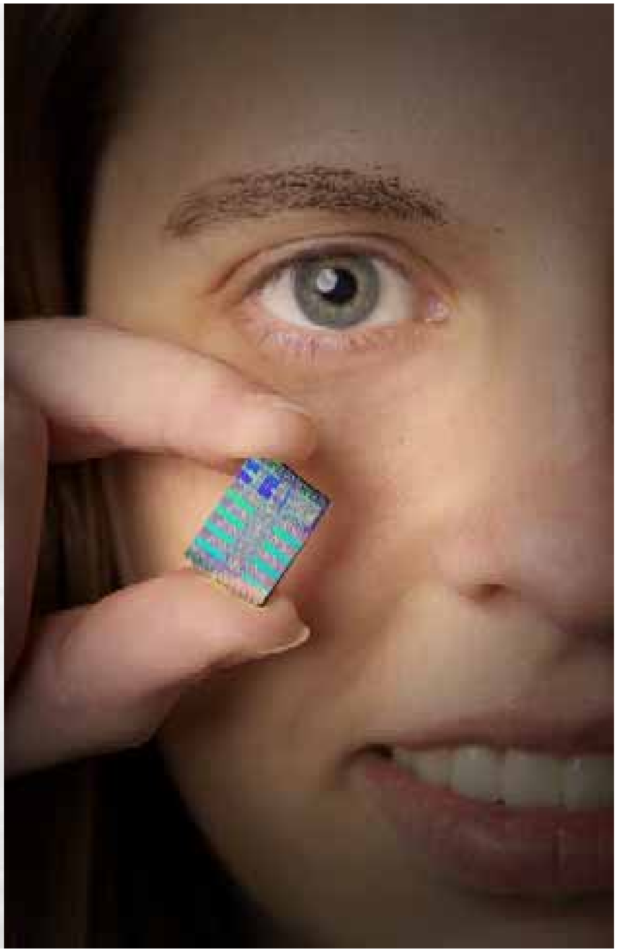
Performance Limiters in Conventional Microprocessors

- Memory Wall
  - Latency induced bandwidth limitations
- Power Wall
  - Must improve efficiency and performance equally
- Frequency Wall
  - Diminishing returns from deeper pipelines

    (can be negative if power is taken into account)

Cell





Cell History

- IBM, SCEI/Sony, Toshiba Alliance formed in 2000
- Design Center opened in March 2001
- Based in Austin, Texas
- February 7, 2005: First external technical disclosures
  - Cell Broadband Engine Architecture documentation can be found at:
    - ✓ http://www.ibm.com/developerworks/power/cell
  - Additional publications on Cell can be downloaded from:
    - ✓ http://www.ibm.com/chips/techlib/techlib.nsf/products/Cell
  - A paper on Cell in the upcoming issue of the IBM Journal of Research and Development can be found at:
    - ✓ http://www.research.ibm.com/journal/rd/494/kahle.html

SONY COMPUTER ENTERTAINMENT

SONY

TOSHIBA

IBM

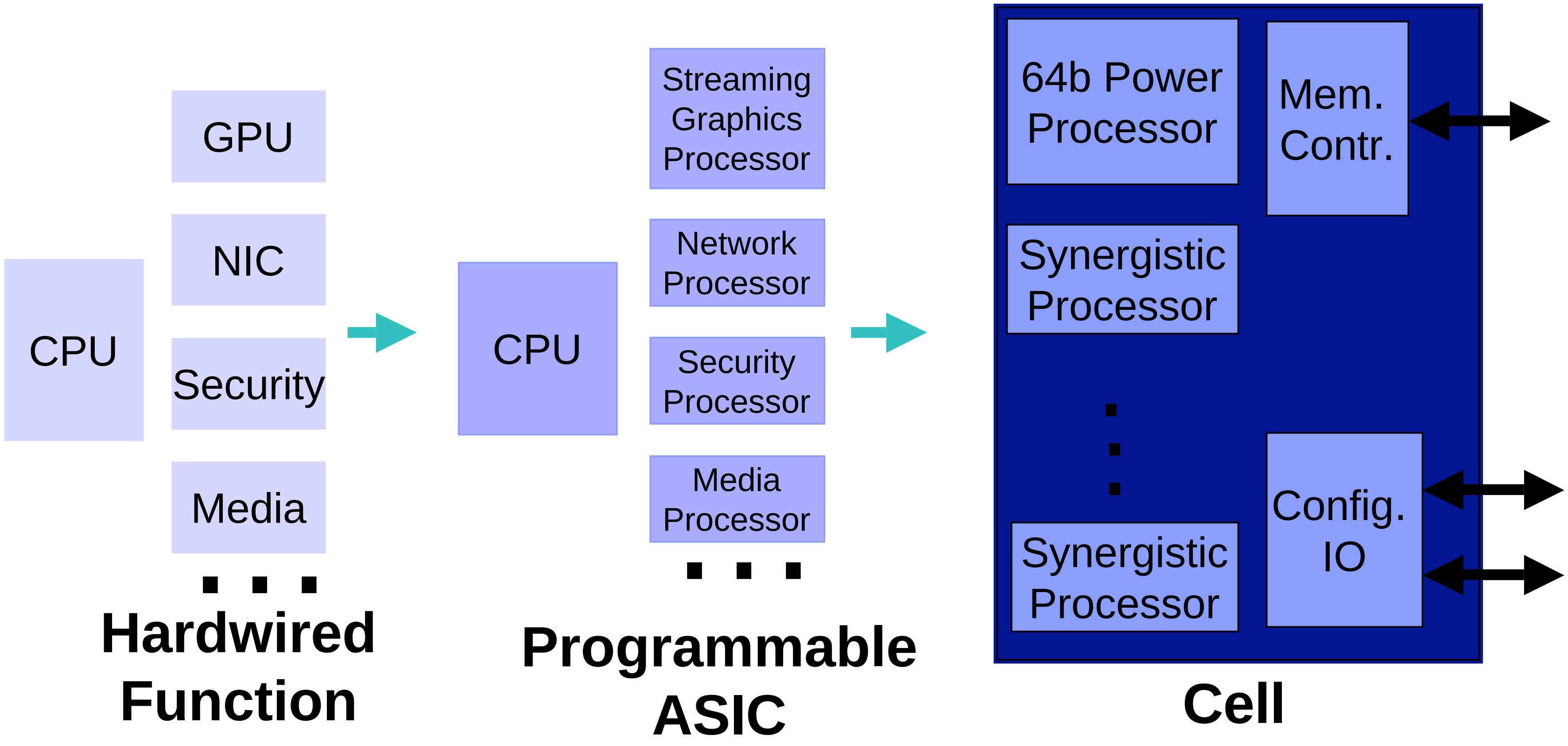
Cell Highlights

- Supercomputer on a chip
- Multi-core microprocessor (9 cores)
- 3.2 GHz clock frequency
- 10x performance for many applications
- Digital home to distributed computing

Introducing Cell

- Sets a new performance standard
  - Exploits parallelism while achieving high frequency
  - Supercomputer attributes with extreme floating point capabilities
  - Sustains high memory bandwidth with smart DMA controllers
- Designed for natural human interaction
  - Photo-realistic effects
  - Predictable real-time response
  - Virtualized resources for concurrent activities
- Designed for flexibility
  - Wide variety of application domains
  - Highly abstracted to highly exploitable programming models
  - Reconfigurable I/O interfaces
  - Autonomic power management

Microprocessor Architecture Trends

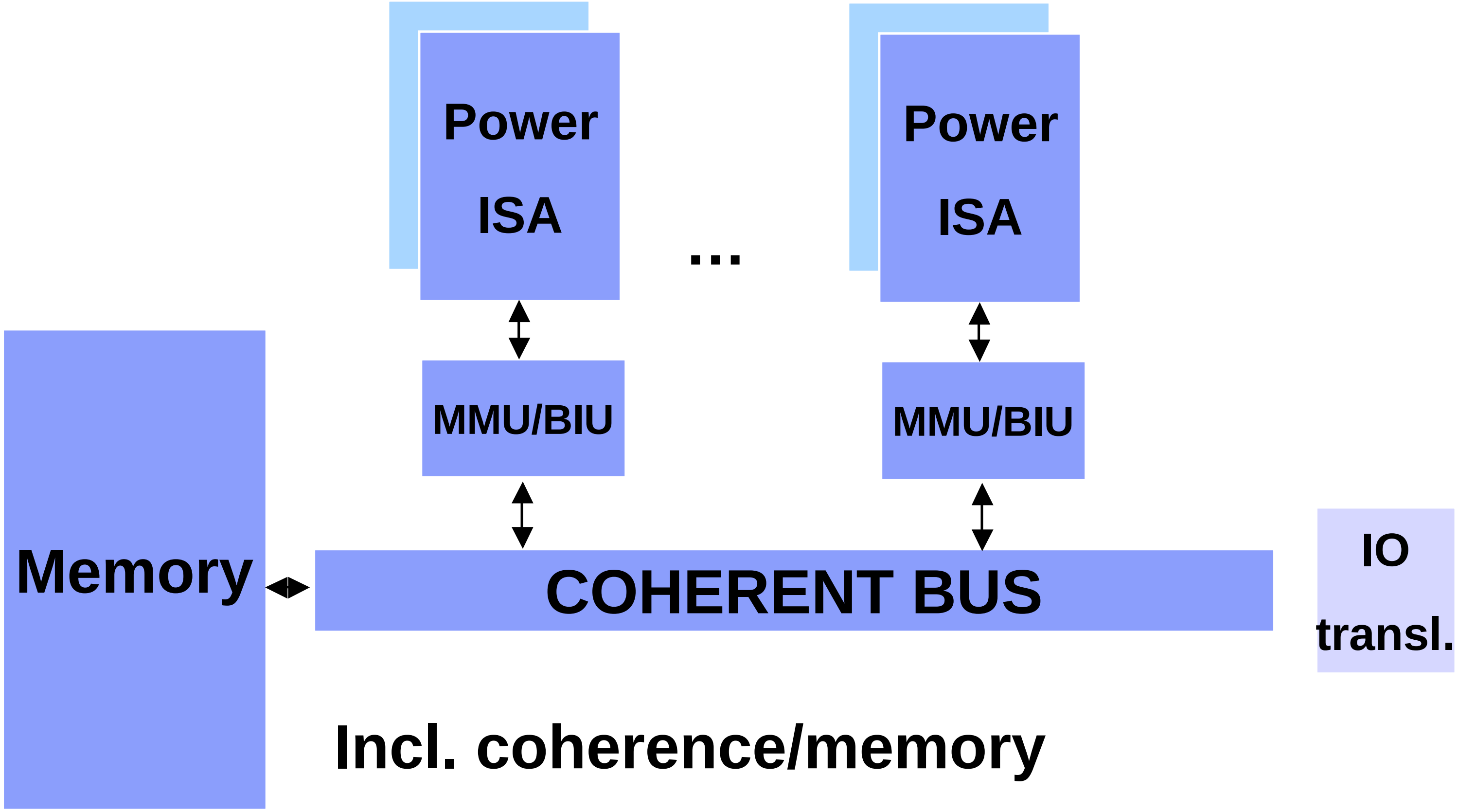


Key Attributes of Cell

- Cell is Multi-Core
  - Contains 64-bit Power Architecture TM
  - Contains 8 Synergistic Processor Elements (SPE)
- Cell is a Flexible Architecture
  - Multi-OS support (including Linux) with Virtualization technology
  - Path for OS, legacy apps, and software development
- Cell is a Broadband Architecture
  - SPE is RISC architecture with SIMD organization and Local Store
  - 128+ concurrent transactions to memory per processor
- Cell is a Real-Time Architecture
  - Resource allocation (for Bandwidth Measurement)
  - Locking Caches (via Replacement Management Tables)
- Cell is a Security Enabled Architecture
  - SPE dynamically reconfigurable as secure processors

Cell Architecture is …

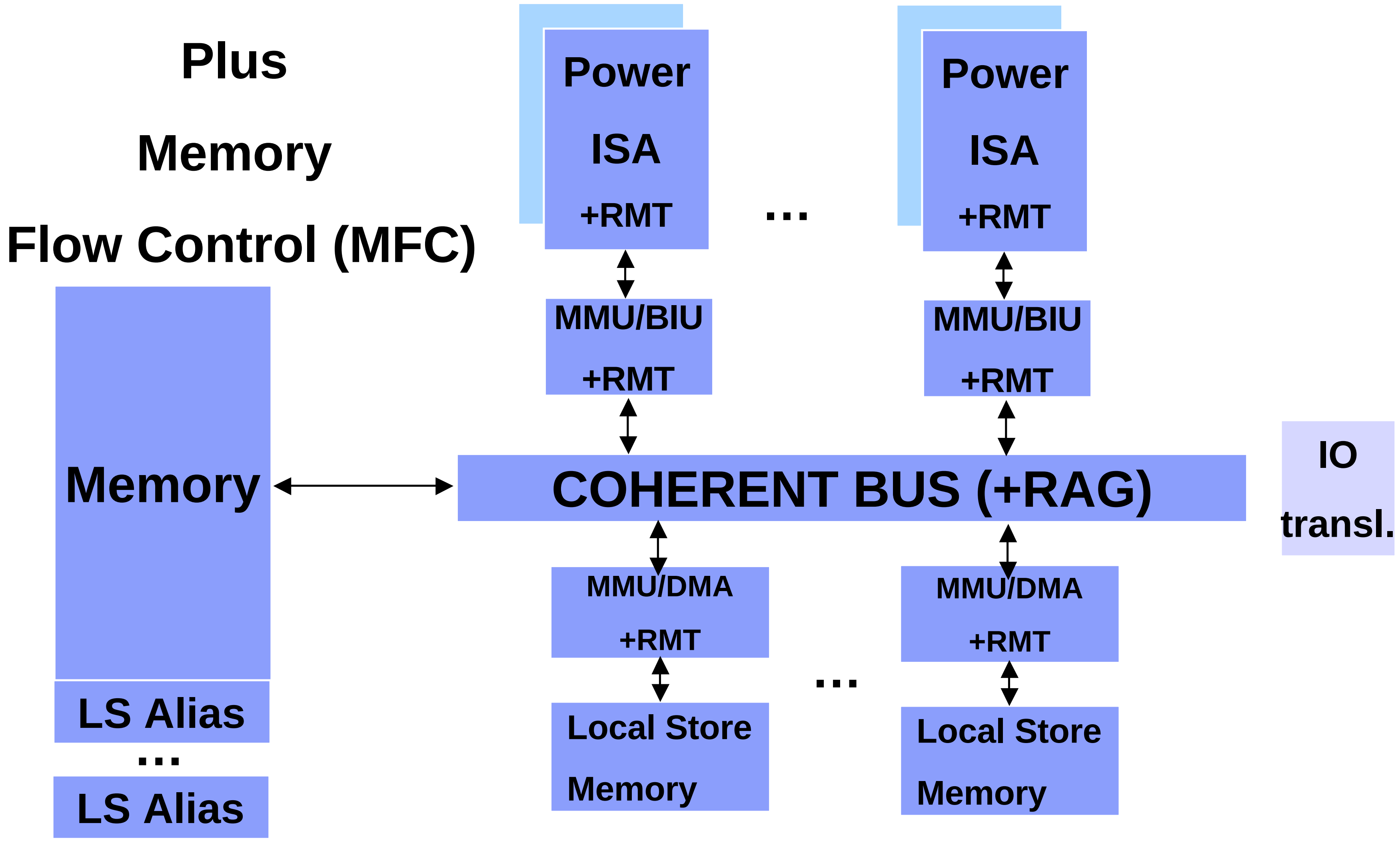
64b Power Architecture™



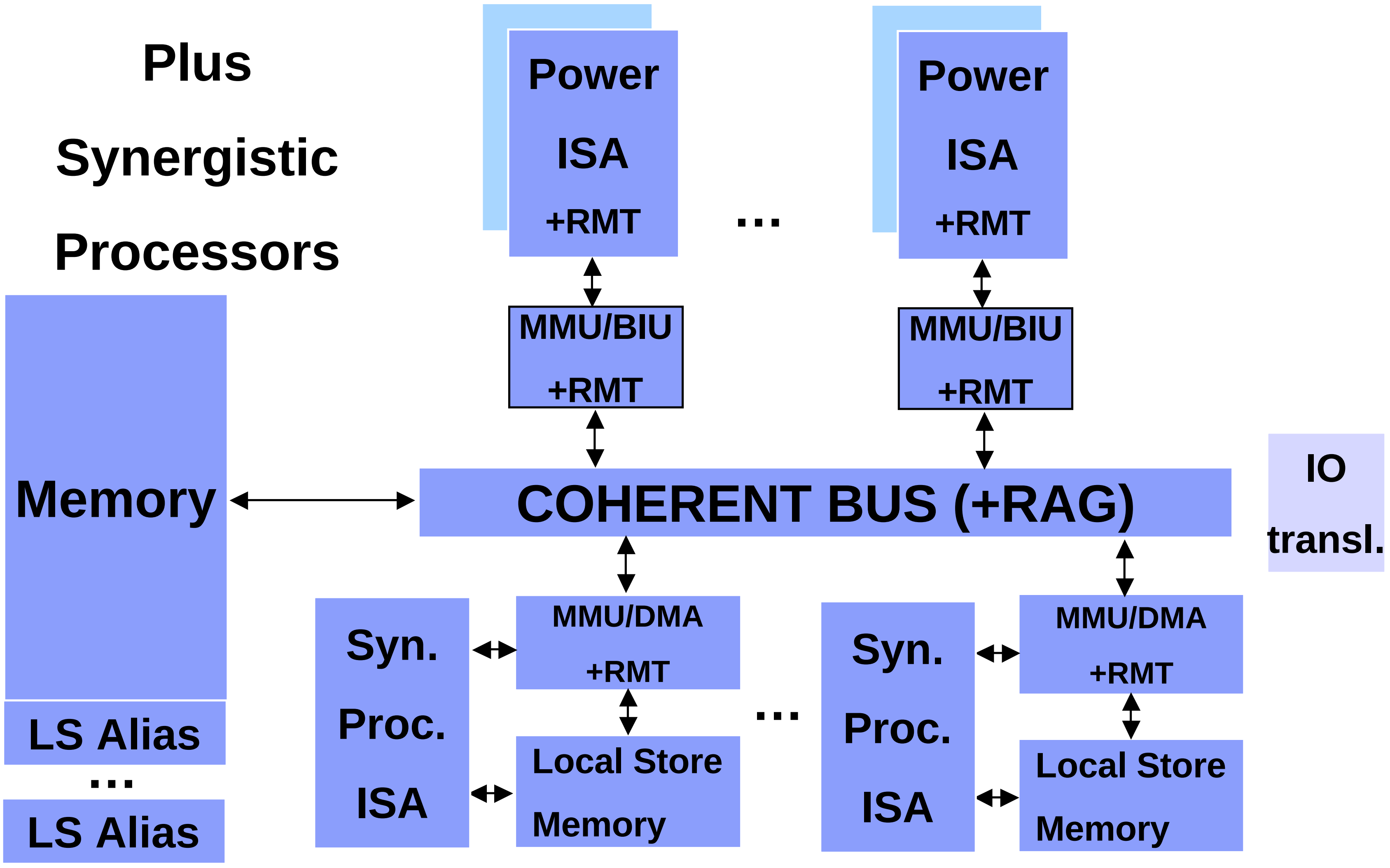
Incl. coherence/memory

compatible with 32/64b Power Arch. Applications and OS's

Cell Architecture is … 64b Power Architecture™

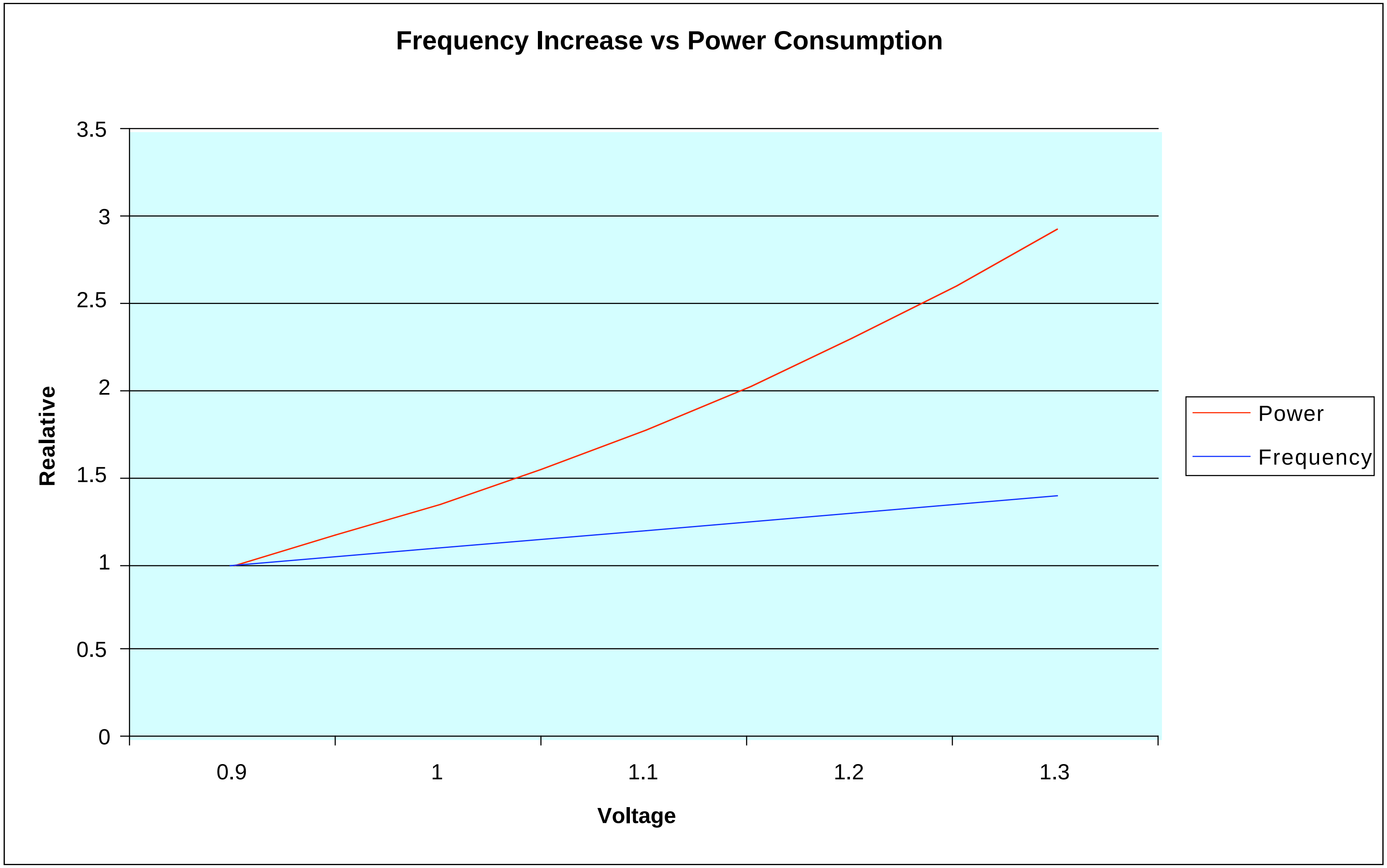


Cell Architecture is … 64b Power Architecture™+ MFC



Cell - Attacking the Performance Walls

- Multi-Core Non-Homogeneous Architecture
  - Control Plane vs. Data Plane processors
  - Attacks **Power Wall**
- 3-level Model of Memory
  - Main Memory, Local Store, Registers
  - Attacks **Memory Wall**
- Large Shared Register File & SW Controlled Branching
  - Allows deeper pipelines
  - Attacks **Frequency Wall**


Frequency Increase vs Power Consumption
3.5
3
2.5
2
1.5
1
0.5
0
Realative
0.9
1
1.1
1.2
1.3
Voltage
Power
Frequency

Power Efficient Architecture and the CellBE

- Non-Homogeneous Coherent Multi-Processor
  - Data-plane/Control-plane specialization
  - More efficient than homogeneous SMP
- 3-level model of Memory
  - Bandwidth without (inefficient) speculation
  - High-bandwidth .. Low power

Power Efficient Architecture and the SPE

- Power Efficient ISA allows Simple Control
  - Single mode architecture
  - No cache
  - Branch hint
  - Large unified register file
  - Channel Interface
- Efficient Microarchitecture
  - Single port local store
  - Extensive clock gating
- Efficient implementation
  - See Cool Chips paper by O. Takahashi et al. and T. Asano et al.

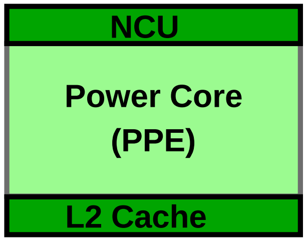
Cell Broadband Engine Components

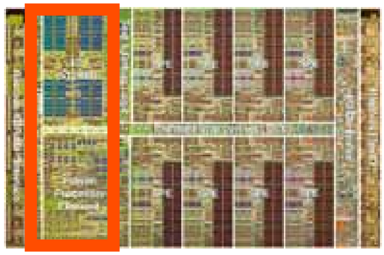
Cell Processor Components

Power Processor Element (PPE):

- •General Purpose, 64-bit RISC Processor (PowerPC AS 2.0.2)
- •2-Way Hardware Multithreaded
- •L1 : 32KB I ; 32KB D
- •L2 : 512KB
- •Coherent load/store
- •VMX
- •3+ GHz
- •Real-time Control
  - •Locking L2 Cache & TLB
  - •Bandwidth Reservation

In the Beginning
– the solitary Power Processor

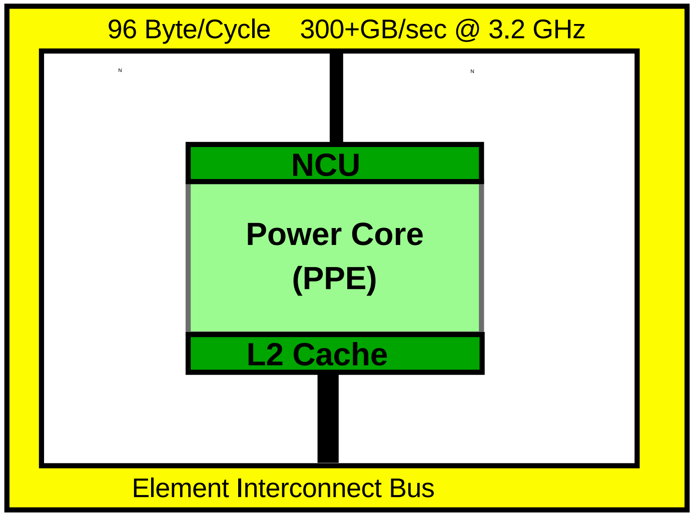


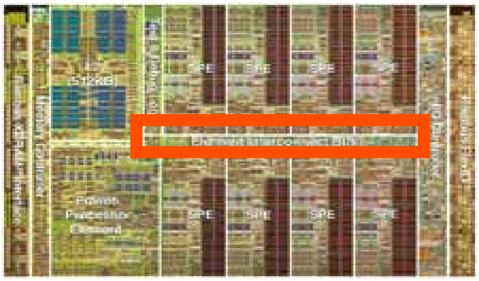


Custom Designed
– for high frequency, space and power efficiency

Cell Processor Components

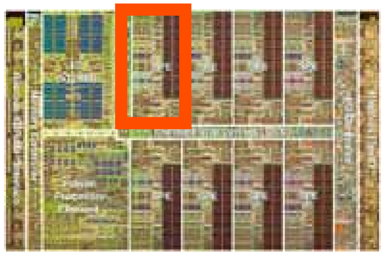
- **EIB data ring for internal communication**
  - Four 16 byte data rings, supporting multiple transfers
  - 96B/cycle peak bandwidth
  - Over 100 outstanding requests
  - 300+ GByte/sec @ 3.2 GHz

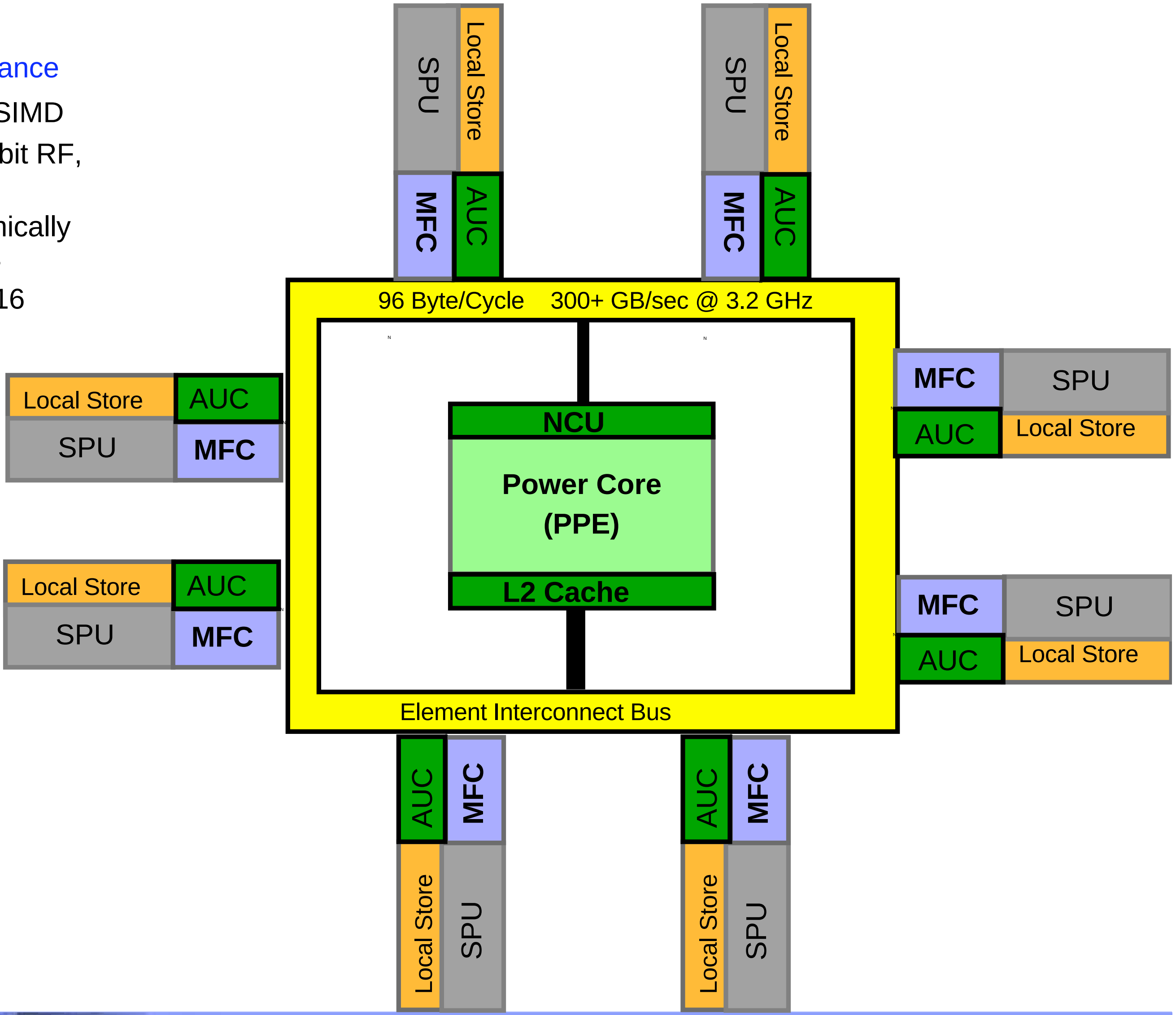




Cell Processor Components

- SPE provides computational performance
  - Dual issue, up to 8-way 32-bit SIMD
  - Dedicated resources: 128 128-bit RF, 256KB Local Store
  - Each DMA MMU can be dynamically configured to protect resources
  - Dedicated DMA engine: Up to 16 outstanding requests

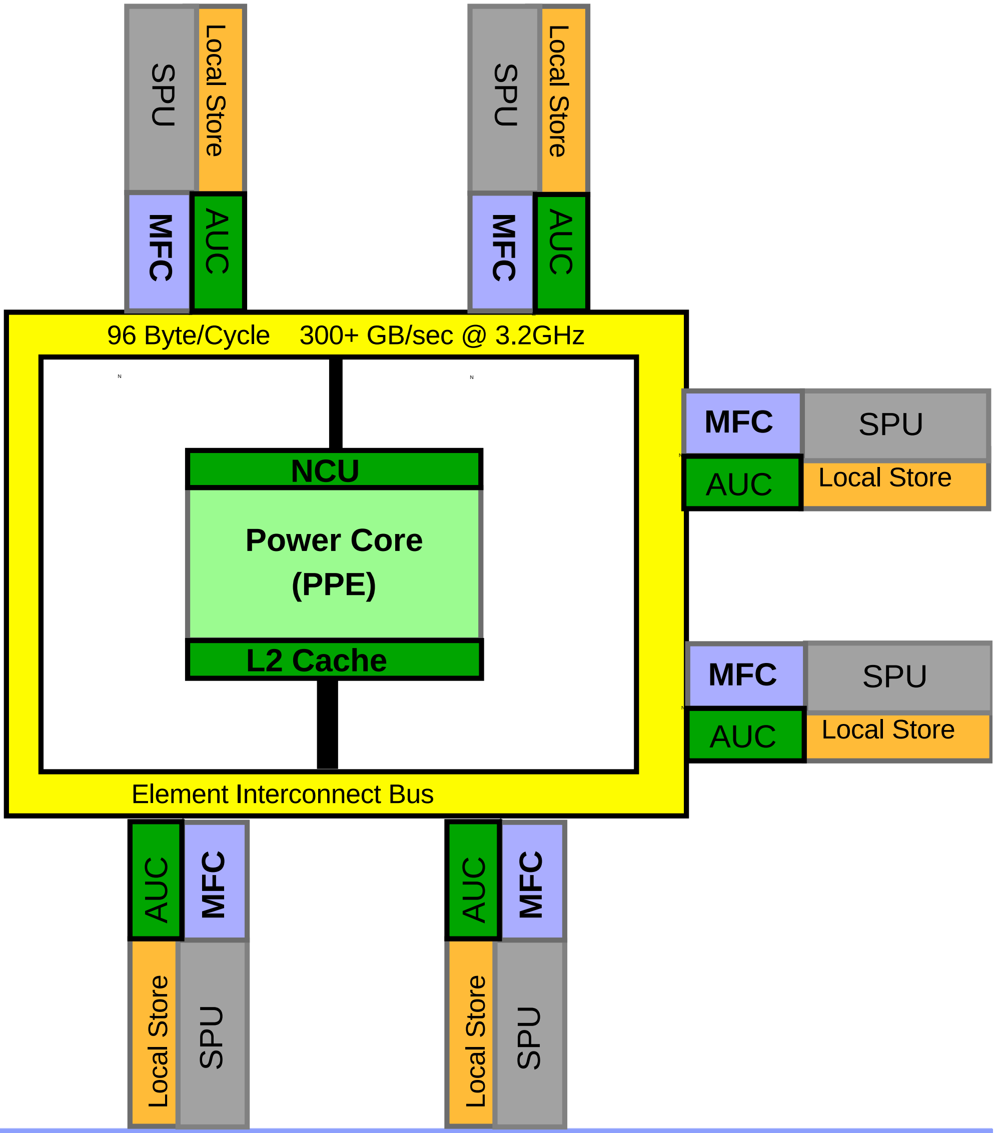


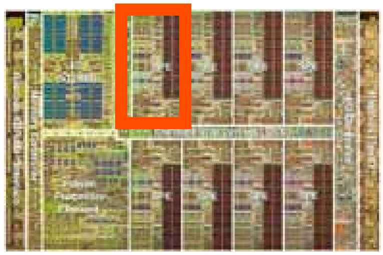


Cell Processor Components

Memory Flow Controller (MFC) and Atomic Update Cache (AUC)

- 4 line cache for shared memory atomic update primitives
- High performance DMA unit
- Local Store aliased into PPE system memory
- 16 element SPE side DMA Command Queue
- 8 element PPE side DMA Command Queue
- DMA List supports 2K entry scatter/gather
- MFC-MMU controls SPE DMA accesses
  - Compatible with PowerPC Virtual Memory architecture
  - Full memory protection for MFC DMA
  - S/W controllable from PPE MMIO
- DMA 1,2,4,8,16,128 -> 16Kbyte transfers for I/O access





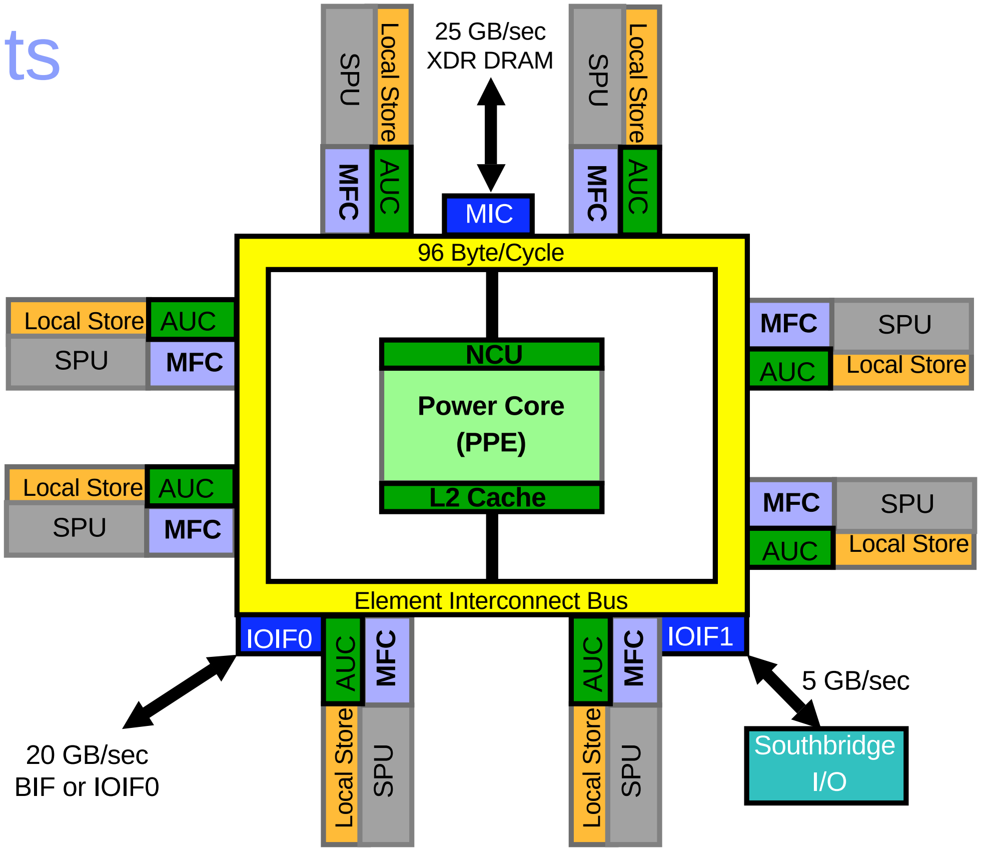
Cell Processor Components

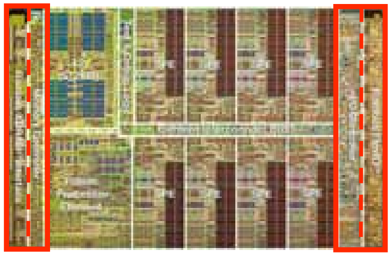
Broadband Interface Controller (BIC):

- Provides a wide connection to external devices
- Two configurable interfaces (50+GB/s @ 5Gbps)
  - Configurable number of bytes
  - Coherent (BIF) and / or I/O (IOIFx) protocols
- Supports two virtual channels per interface
- Supports multiple system configurations

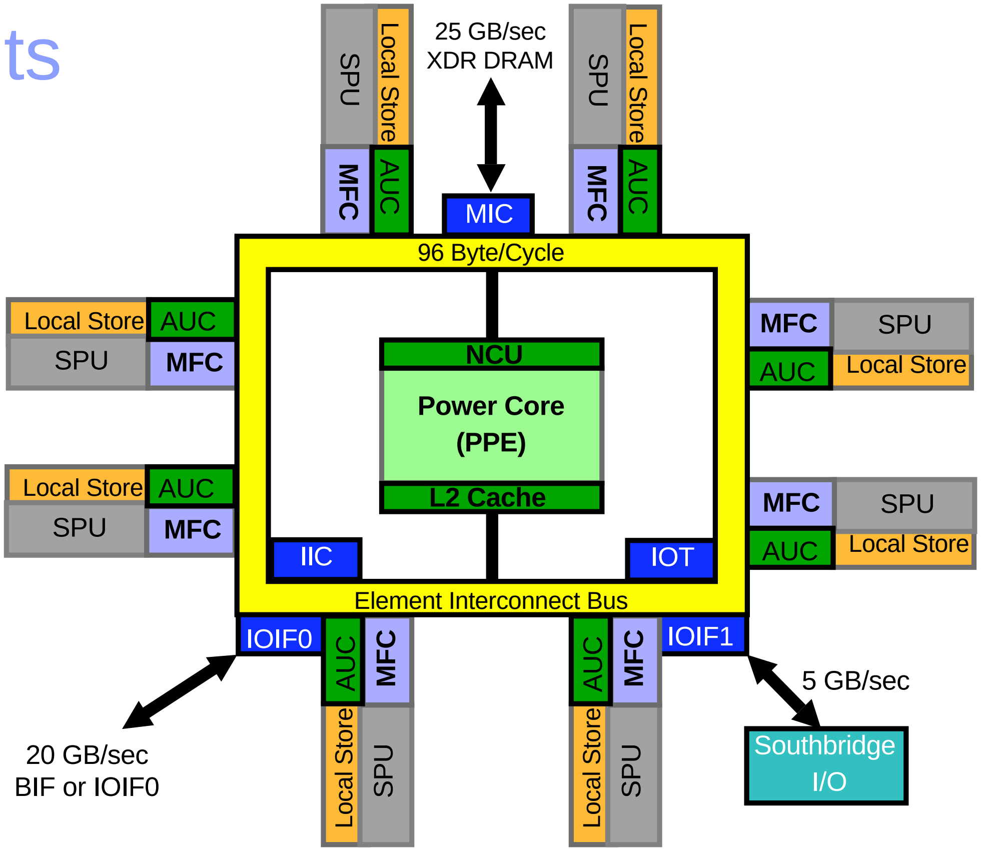
Memory Interface Controller (MIC):

- Dual XDR^{TM} controller (25.6GB/s @ 3.2Gbps)
- ECC support
- Suspend to DRAM support





Cell Processor Components

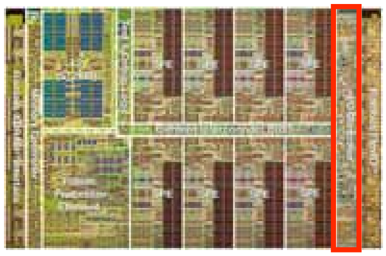


Internal Interrupt Controller (IIC)

- Handles SPE Interrupts
- Handles External Interrupts
  - From Coherent Interconnect
  - From IOIF0 or IOIF1
- Interrupt Priority Level Control
- Interrupt Generation Ports for IPI
- Duplicated for each PPE hardware thread

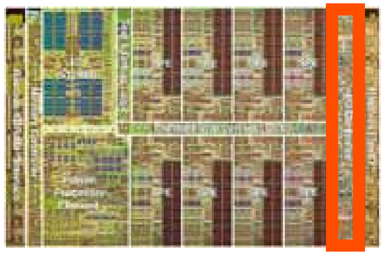
I/O Bus Master Translation (IOT)

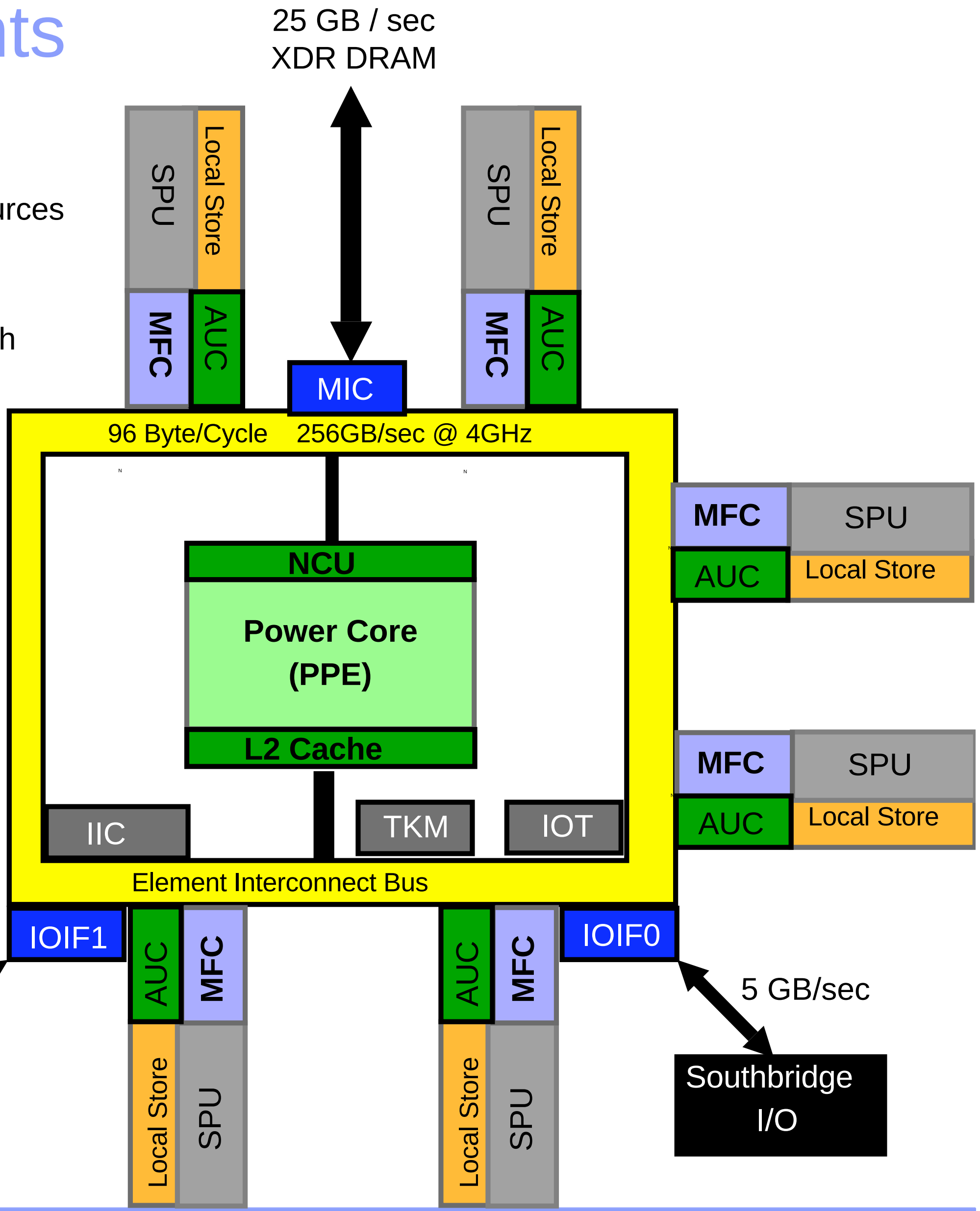
- Translates Bus Addresses to System Real Addresses
- Two Level Translation
  - I/O Segments (256 MB)
  - I/O Pages (4K, 64K, 1M, 16M byte)
- I/O Device Identifier per page for LPAR
- IOST and IOPT Cache – hardware / software managed



Cell Processor Components

- **Token Manager** – Bandwidth Reservation for shared resources
  - Optionally Enabled for RT Tasks or LPAR
  - Multiple Resource Allocation Groups
  - Generates access tokens at configurable rate for each allocation group
    - 1 per each memory bank (16 total)
    - 2 for each IOIF (4 total)
  - Requestors assigned RAG ID by OS/hypervisor
    - Each SPE
    - PPE L2 / NCU
    - IOIF 0 Bus Master
    - IOIF 1 Bus Master
  - Priority order for using another RAGs unused tokens
  - Resource overcommit warning interrupt

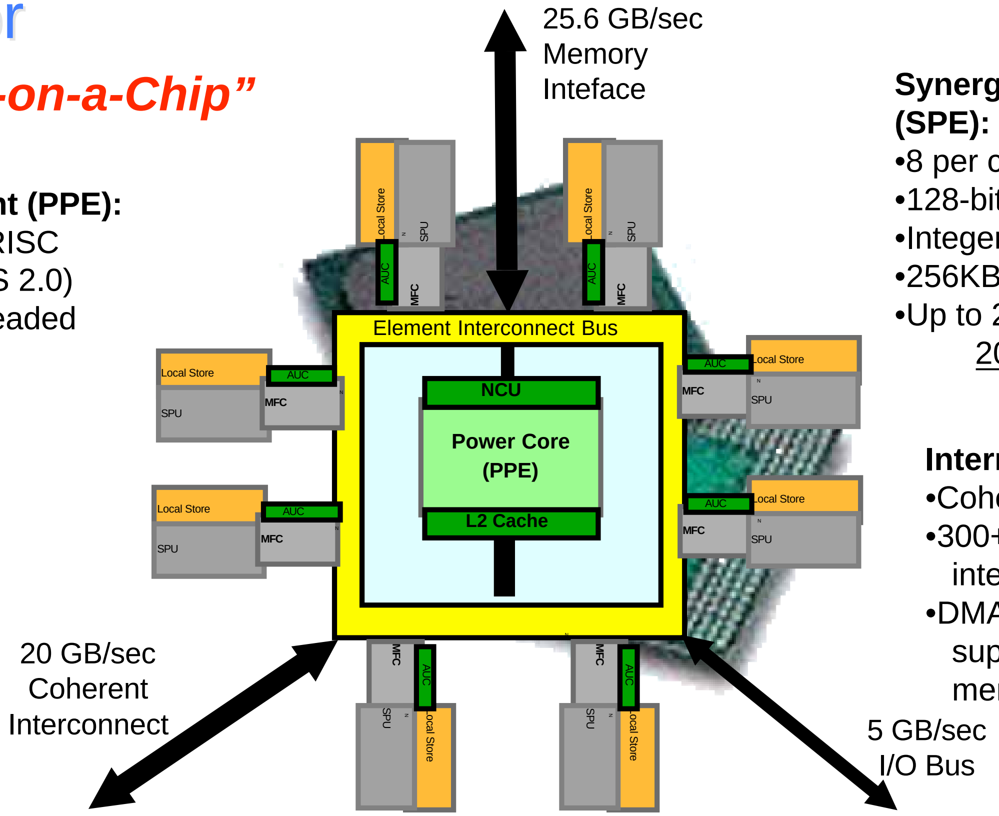




Cell Processor

"Supercomputer-on-a-Chip"

Power Processor Element (PPE):
- General Purpose, 64-bit RISC Processor (PowerPC AS 2.0)
- 2-Way Hardware Multithreaded
- L1 : 32KB I ; 32KB D
- L2 : 512KB
- Coherent load/store
- VMX
- 3.2 GHz



Synergistic Processor Elements (SPE):
- 8 per chip
- 128-bit wide SIMD Units
- Integer *and* Floating Point capable
- 256KB Local Store
- Up to 25.6 GF/s per SPE --- 200GF/s total *

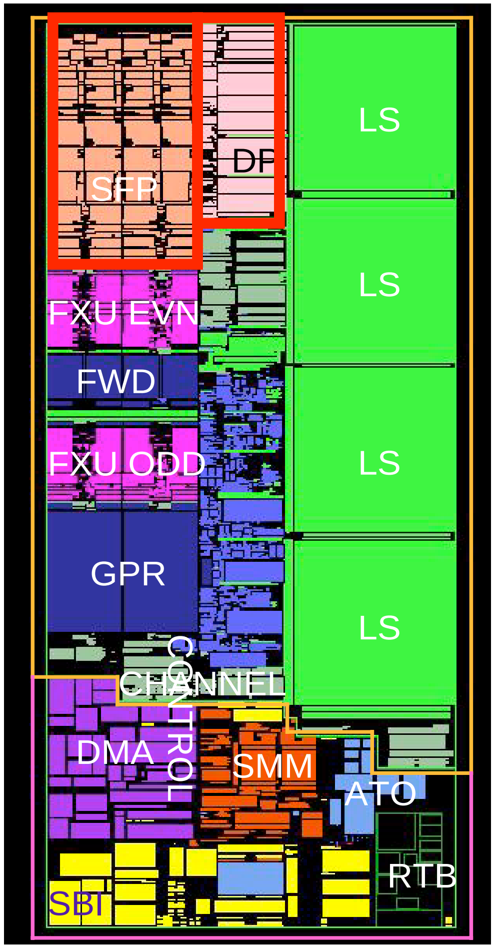
** At clock speed of 3.2GHz*

Internal Interconnect:
- Coherent ring structure
- 300+ GB/s total internal interconnect bandwidth
- DMA control to/from SPEs supports >100 outstanding memory requests

External Interconnects:
- 25.6 GB/sec BW memory interface
- 2 Configurable I/O Interfaces
  - Coherent interface (SMP)
  - Normal I/O interface (I/O & Graphics)
  - Total BW configurable between interfaces
  - Up to 35 GB/s out
  - Up to 25 GB/s in

Memory Management & Mapping
- SPE Local Store aliased into PPE system memory
- MFC/MMU controls SPE DMA accesses
  - Compatible with PowerPC Virtual Memory architecture
  - S/W controllable from PPE MMIO
- Hardware or Software TLB management
- SPE DMA access protected by MFC/MMU

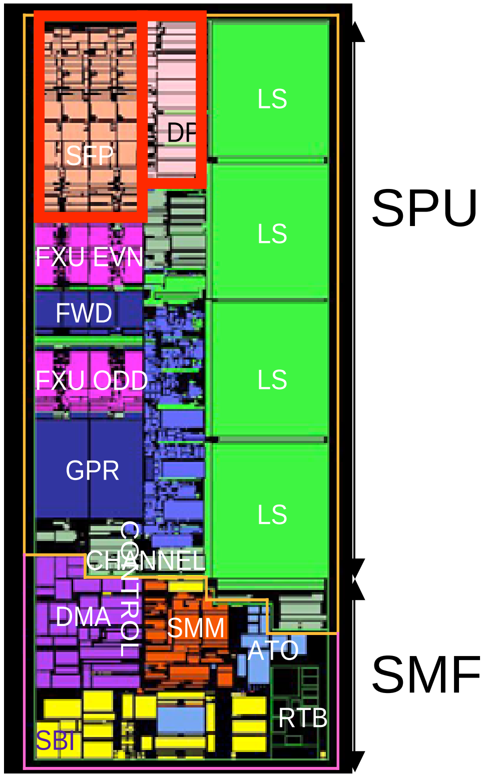
SPE Highlights



- RISC like organization
  - 32 bit fixed instructions
  - Clean design – unified Register file
- User-mode architecture
  - No translation/protection within SPU
  - DMA is full Power Arch protect/x-late
- VMX-like SIMD dataflow
  - Broad set of operations (8 / 16 / 32 Byte)
  - Graphics SP-Float
  - IEEE DP-Float
- Unified register file
  - 128 entry x 128 bit
- 256KB Local Store
  - Combined I & D
  - 16B/cycle L/S bandwidth
  - 128B/cycle DMA bandwidth

What is a Synergistic Processor? (and why is it efficient?)

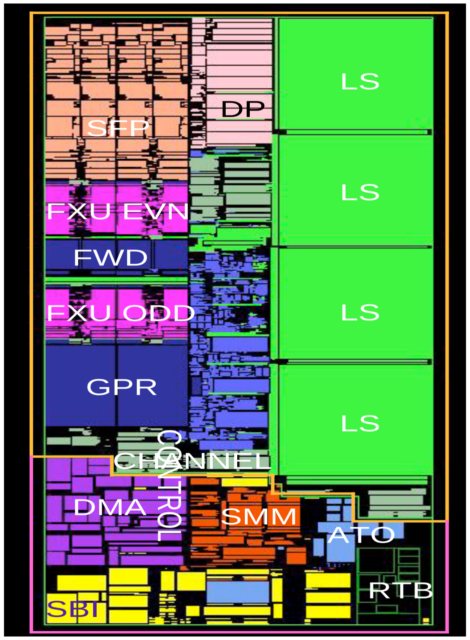
- Local Store "is" large 2nd level register file / private instruction store instead of cache
  - Asynchronous transfer (DMA) to shared memory
  - Frontal attack on the Memory Wall
- Media Unit turned into a Processor
  - Unified (large) Register File
  - 128 entry x 128 bit
- Media & Compute optimized
  - One context
  - SIMD architecture



SPE Detail

Synergistic Processor Element (SPE)

- User-mode architecture
  - No translation/protection within SPE
  - DMA is full PowerPC protect/xlate
- Direct programmer control
  - DMA/DMA-list
  - Branch hint
- VMX-like SIMD dataflow
  - Graphics SP-Float
  - No saturate arith, some byte
  - IEEE DP-Float (BlueGene-like)
- Unified register file
  - 128 entry x 128 bit
- 256KB Local Store
  - Combined I & D
  - 16B/cycle L/S bandwidth
  - 128B/cycle DMA bandwidth



SPU Units:

- Simple (FXU even)
  - Add/Compare
  - Rotate
  - Logical, Count Leading Zero
- Permute (FXU odd)
  - Permute
  - Table-lookup
- FPU (Single / Double Precision)
- Control (SCN)
  - Dual Issue, Load/Store, ECC Handling
- Channel (SSC) – Interface to MFC
- Register File (GPR/FWD)

SPE Latencies

- Simple fixed point - 2 cycles*
- Complex fixed point - 4 cycles*
- Load - 6 cycles*
  - Local store size = 256 KB
- Single-precision (ER) float - 6 cycles*
- Integer multiply - 7 cycles*
- Branch miss - 20 cycles
  - No penalty if correctly hinted
- DP (IEEE) float - 13 cycles*
  - Partially pipelined
- Enqueue DMA Command - 20 cycles*

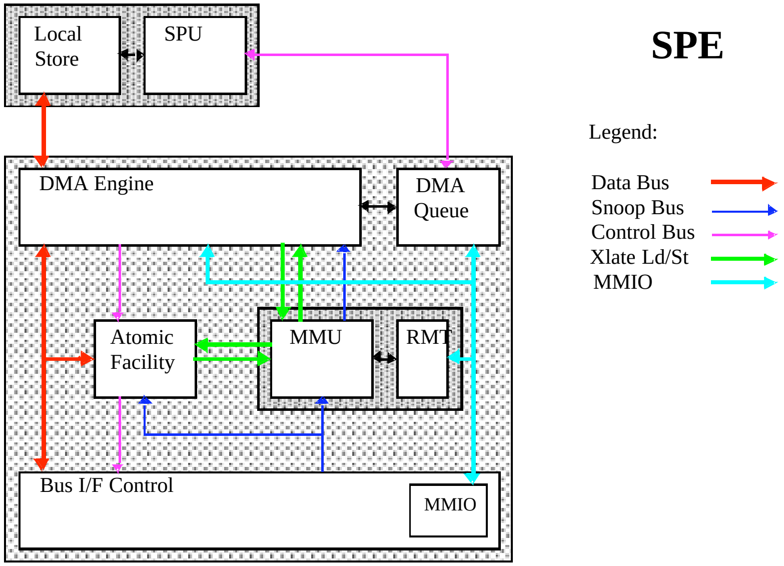
Coherent Offload Model

- DMA into and out of Local Store similar to Power core loads & stores
- Governed by Power Architecture page and segment tables for translation and protection
- Shared memory model
  - Power architecture compatible addressing
  - MMIO capabilities for SPEs
  - Local Store is mapped (alias) allowing LS to LS DMA transfers
  - DMA equivalents of locking loads & stores
  - OS management/virtualization of SPEs
    - Pre-emptive context switch is supported

MFC Detail

Memory Flow Control System

- DMA Unit
  - LS <-> LS, LS<-> Sys Memory, LS<-> I/O Transfers
  - 8 PPE-side Command Queue entries
  - 16 SPU-side Command Queue entries
- MMU similar to PowerPC MMU
  - 8 SLBs, 256 TLBs
  - multiple page sizes
  - Software/HW page table walk
  - PT/SLB misses interrupt PPE
- Atomic Cache Facility
  - 4 cache lines for atomic updates
  - 2 cache lines for cast out/MMU reload
- Up to 16 outstanding DMA requests in BIU
- Resource / Bandwidth Management Tables
  - Token Based Bus Access Management
  - TLB Locking

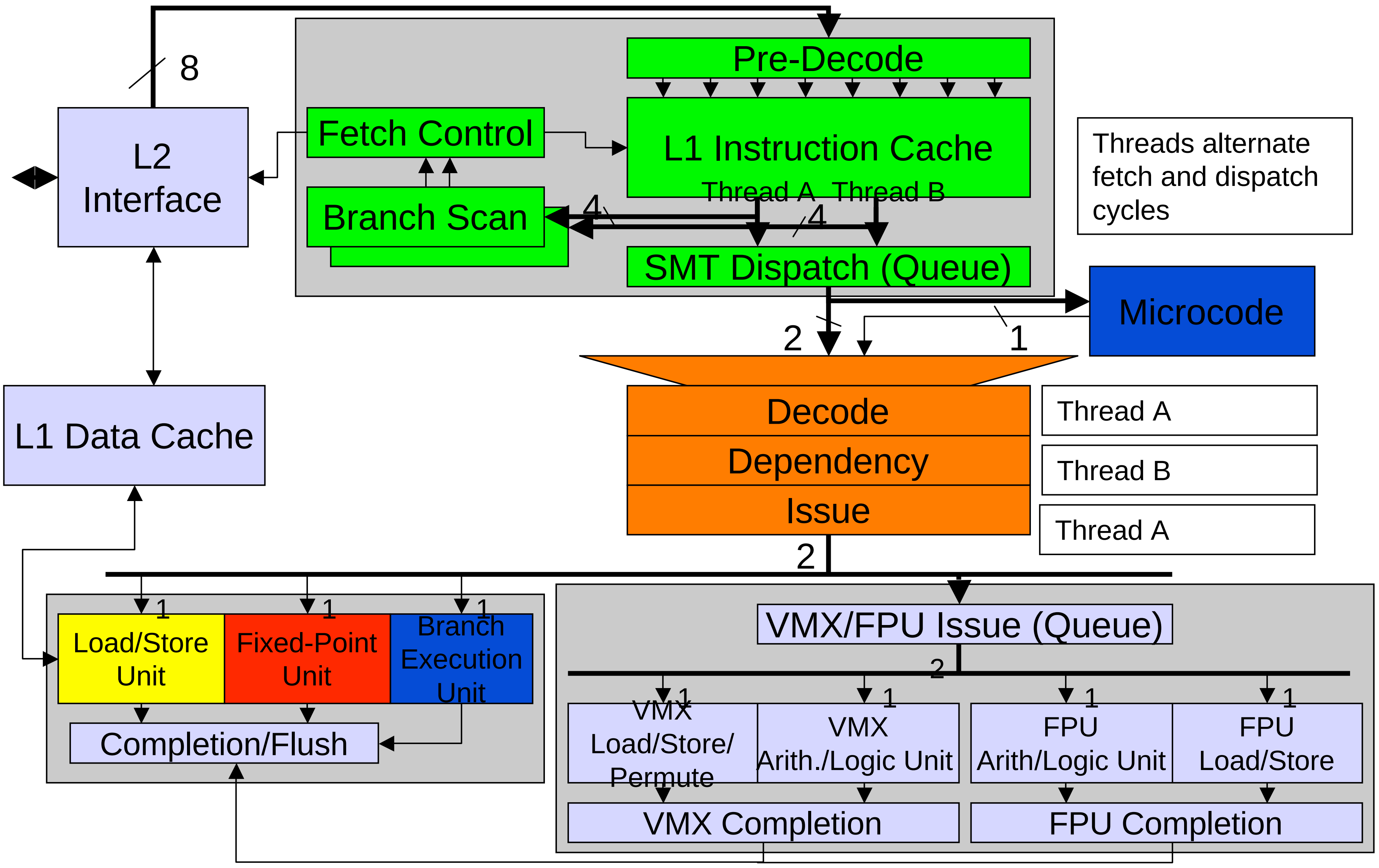


Isolation Mode Support (Security Feature)

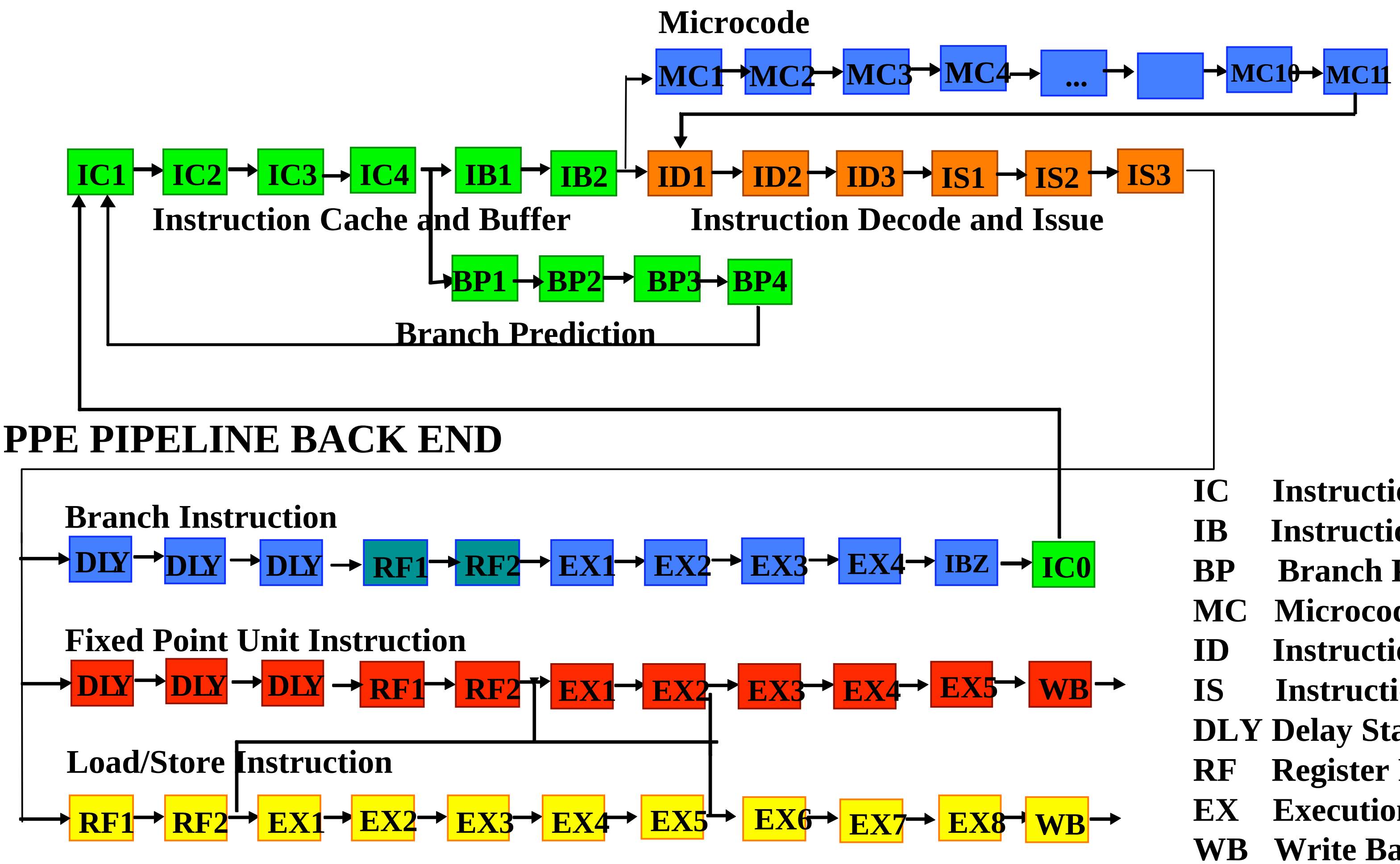
- Hardware enforced “isolation”
  - SPE and Local Store not visible (bus or jtag)
  - Small LS “untrusted area” for communication area
- Secure Boot
  - Chip Specific Key
  - Decrypt/Authenticate Boot code
- “Secure Vault” – Runtime Isolation Support
  - Isolate Load Feature
  - Isolate Exit Feature

Cell Implementation Aspects

PPE BLOCK DIAGRAM



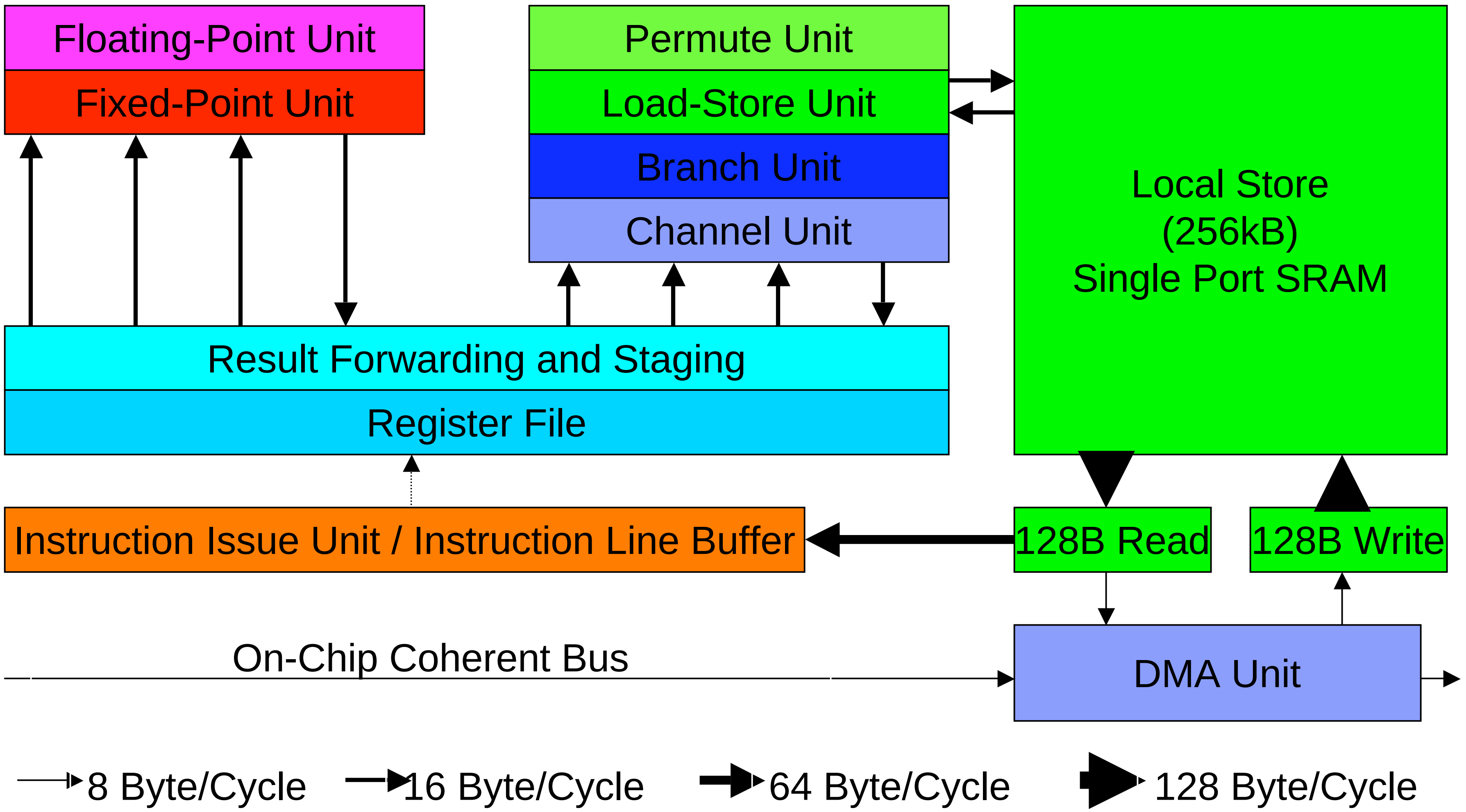
PPE PIPELINE FRONT END



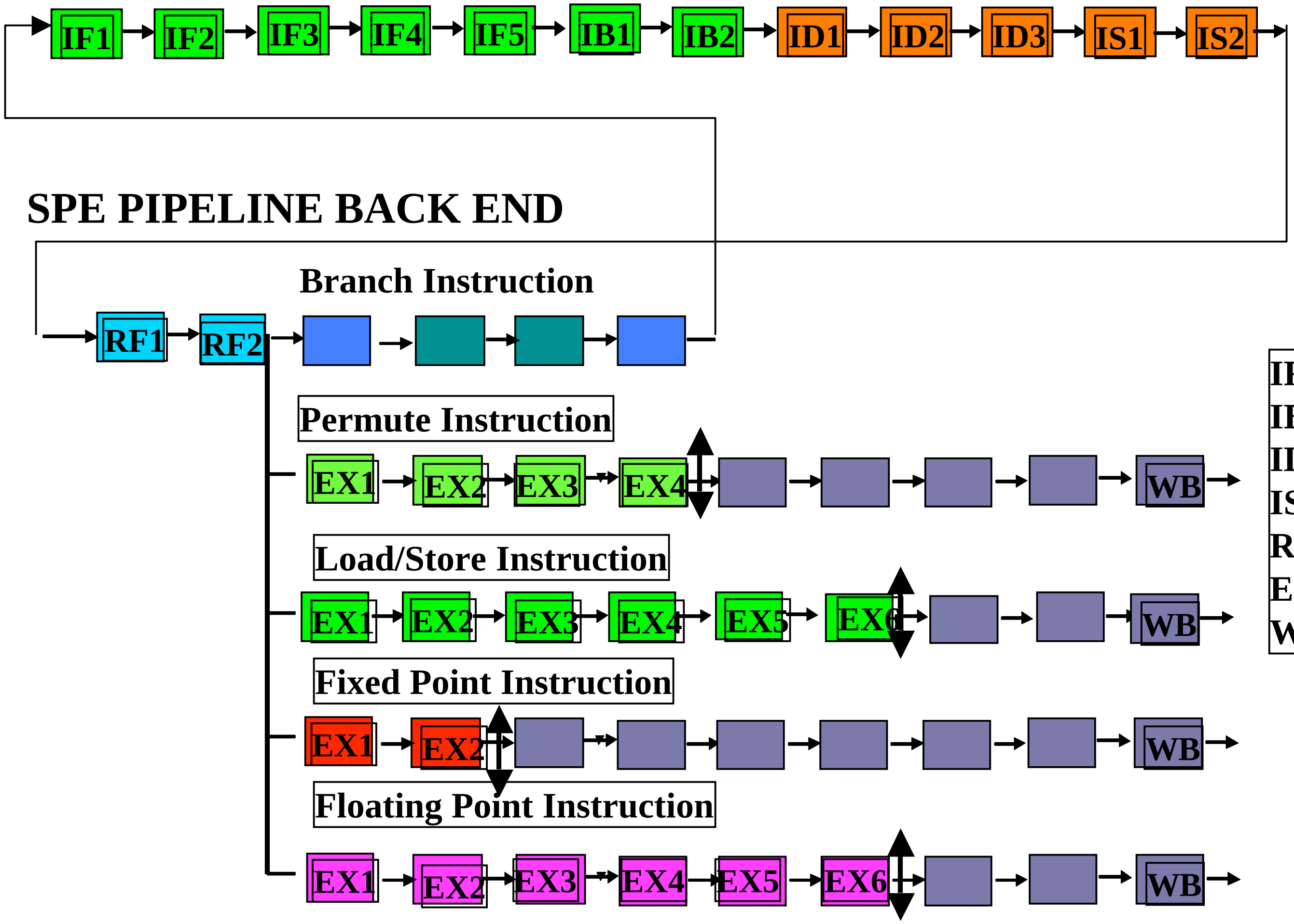
PPE PIPELINE BACK END

IC	Instruction Cache
IB	Instruction Buffer
BP	Branch Prediction
MC	Microcode
ID	Instruction Decode
IS	Instruction Issue
DLY	Delay Stage
RF	Register File Access
EX	Execution
WB	Write Back

SPE BLOCK DIAGRAM



SPU PIPELINE FRONT END

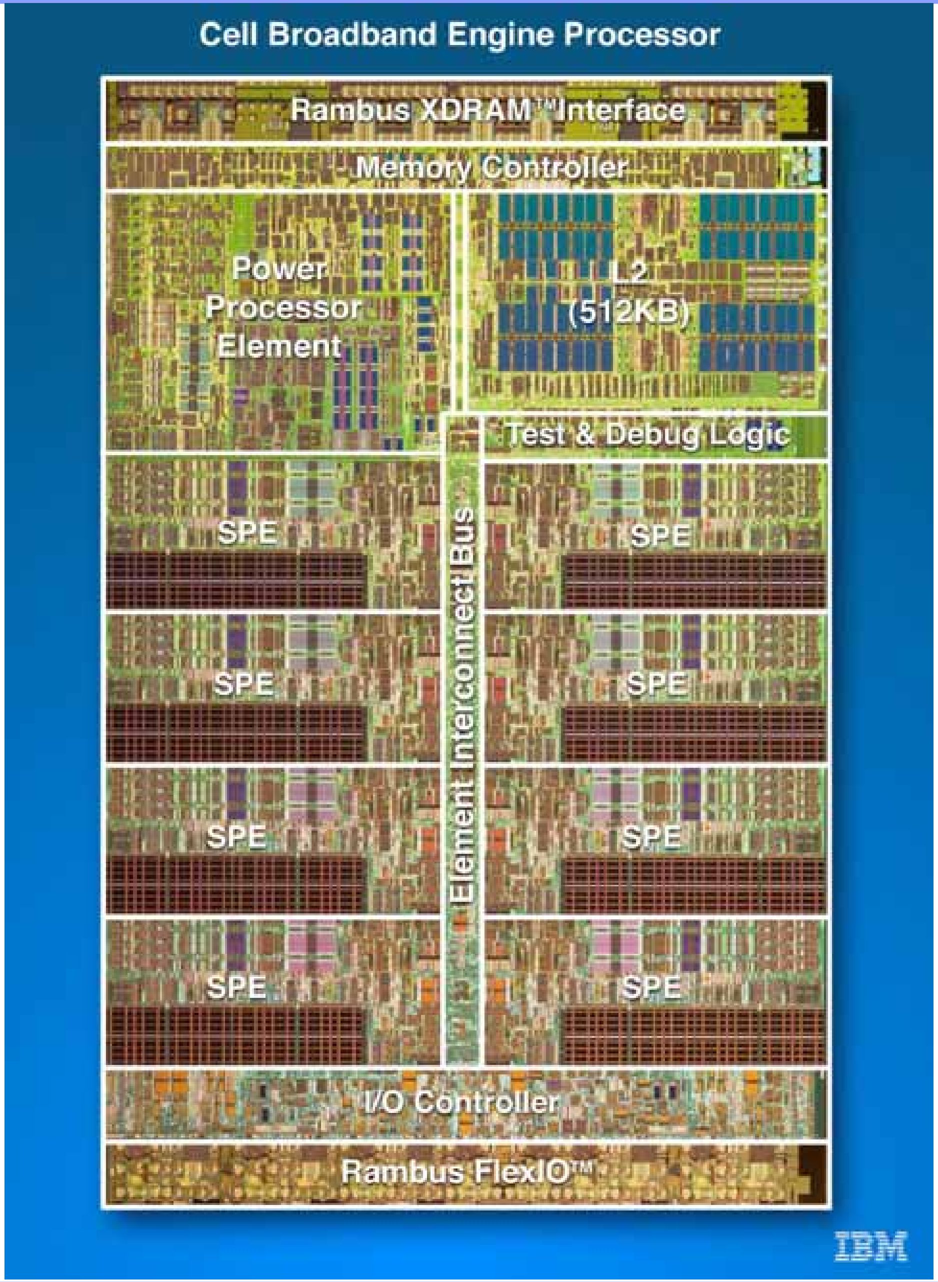


SPE PIPELINE BACK END

IF	Instruction Fetch
IB	Instruction Buffer
ID	Instruction Decode
IS	Instruction Issue
RF	Register File Access
EX	Execution
WB	Write Back

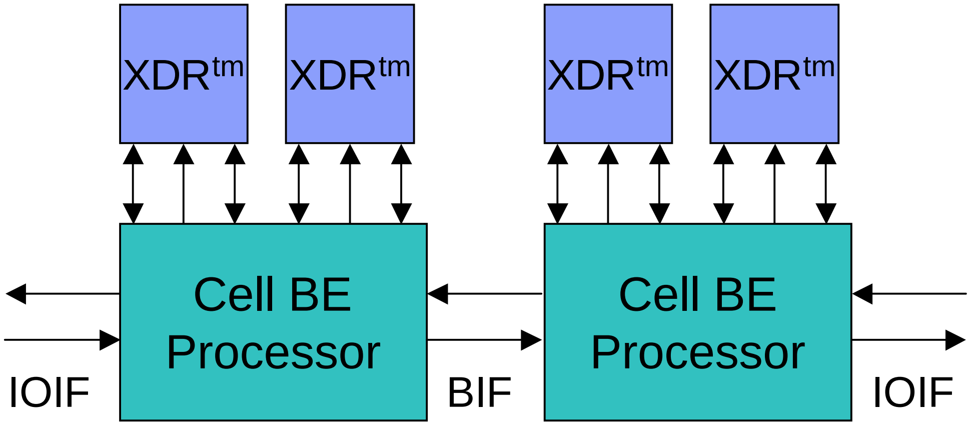
CellBE Processor

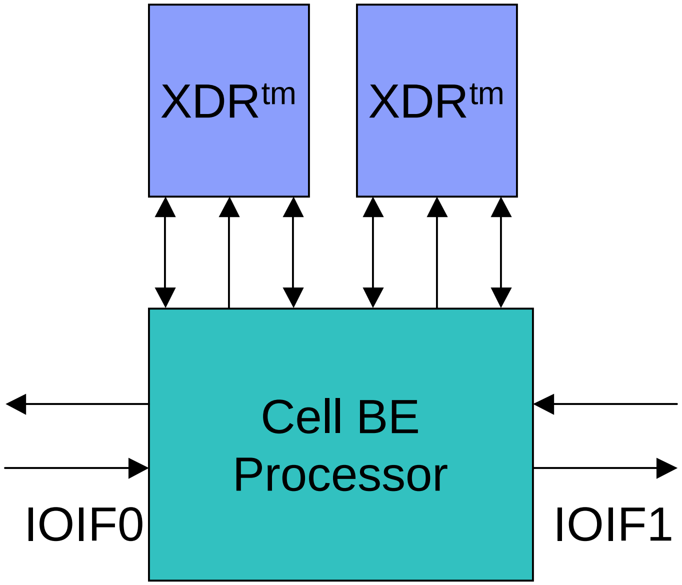
- ~250M transistors
- ~235mm2
- Top frequency >3GHz
- 9 cores, 10 threads
- > 200+ GFlops (SP) @3.2 GHz
- > 20+ GFlops (DP) @3.2 GHz
- Up to 25.6GB/s memory B/W
- Up to 50+ GB/s I/O B/W
- ~400M$(US) design investment

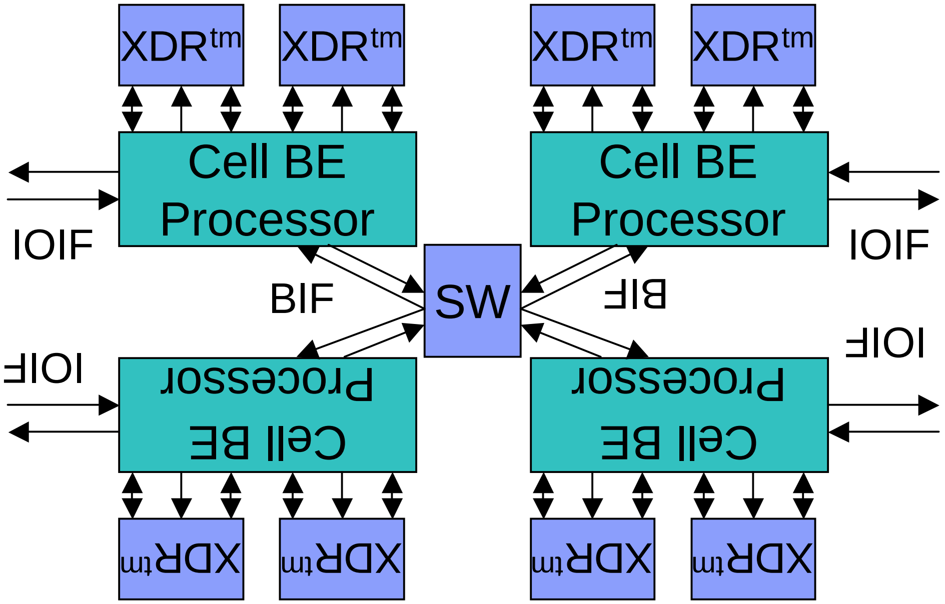


Cell Processor Can Support Many Systems

- **Game console systems**
- **Blades**
- **HDTV**
- **Home media servers**
- **Supercomputers**







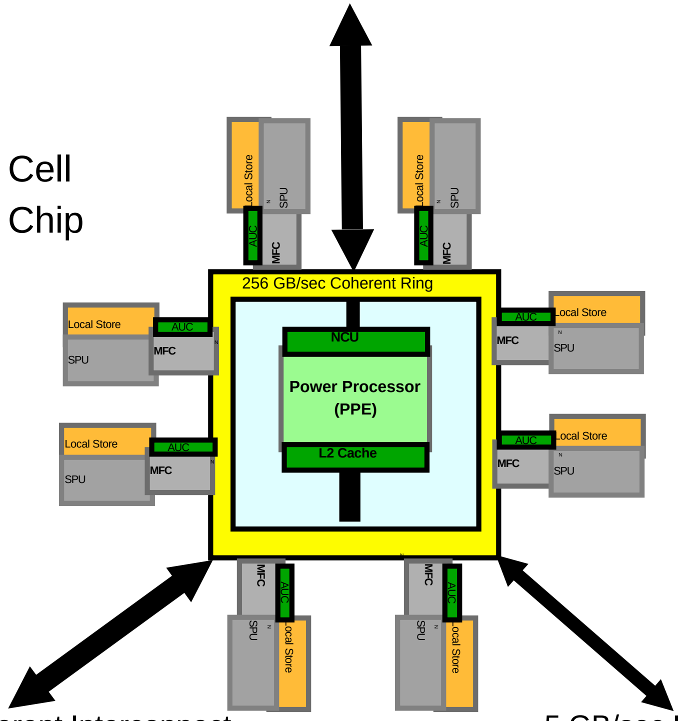
Cell BE Processor Initial Application Areas

- **Cell excels at processing of rich media content in the context of broad connectivity**
  - Digital content creation (games and movies)
  - Game playing and game serving
  - Distribution of dynamic, media rich content
  - Imaging and image processing
  - Image analysis (e.g. video surveillance)
  - Next-generation physics-based visualization
  - Video conferencing
  - Streaming applications (codecs etc.)
  - Physical simulation & science
- **Cell is an excellent match for any applications that require:**
  - Parallel processing
  - Real time processing
  - Graphics content creation or rendering
  - Pattern matching
  - High-performance SIMD capabilities

Cell Software Overview

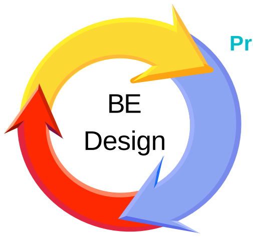
- ■Cell Programming Features
- ■Flexible Programming Models
  - ■Function Offload Model
    - ■Existing Application Accelerator
  - ■Parallel Computational Acceleration
  - ■Heterogeneous Multi-Threading Runtime

Cell Chip



20 GB/sec Coherent Interconnect

5 GB/sec I/O Bus



Raw Hardware Performance

Programmability

Programmer Productivity

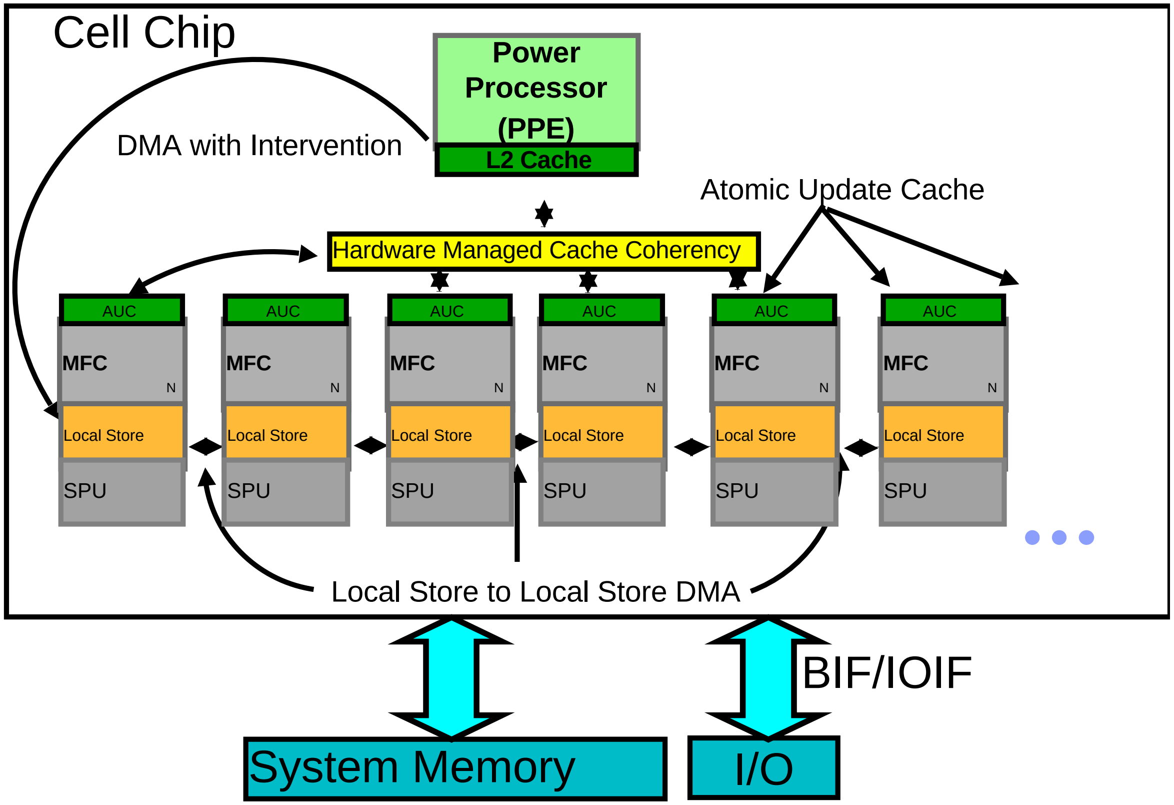
Cell Programming Characteristics

- Exploit all of Cell’s 18 asynchronous engines
  - Through function offload, or parallel computational tasks
- Decompose work into data parallel blocks
  - Independent Data parallel tasks are most efficient
- Overlap SPE compute with DMA loads and stores
  - Multibuffering , pipelining
- Reduce PPE/SPE workload ratio
  - PPE as Control / OS processor
  - SPE as heavy lifting computational engines
- Super-Linear speedups over conventional processors may be achieved
  - Through exploitation of asynchronous DMA engines

Cell Features Exploited by Software

Cell Programming Features

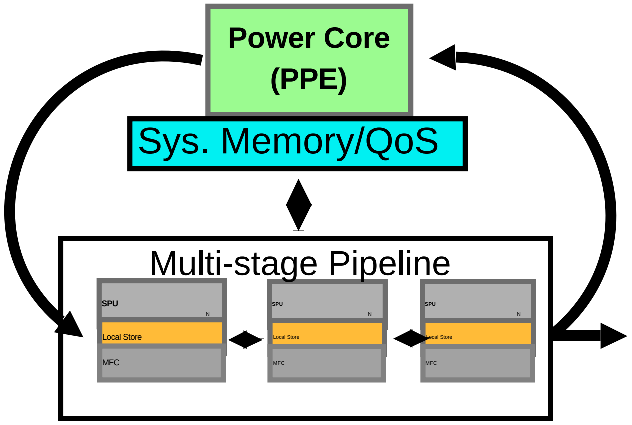
- Keeping Intermediate/Control Data on-Chip
  - MMU – ERATS, SLBs, TLBs
  - DMA from L2 cache-> LS
  - LS to LS DMA
  - Cache <-> Cache transfers (atomic update)
  - SPE Signalling Registers
  - SPE <-> PPE Mailboxes
- Resource Reservation and Allocation
  - PPE can be shared across logical partitions
  - SPEs can be assigned to logical partitions
  - SPEs independently or Group Allocated
  - Cache Replacement Management
  - TLB Replacement Management
  - Bandwidth Reservation

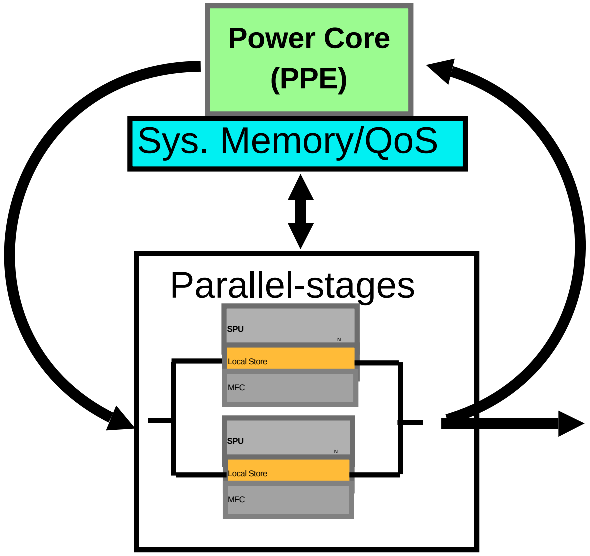


Subsystem Programming Model

Function Offload

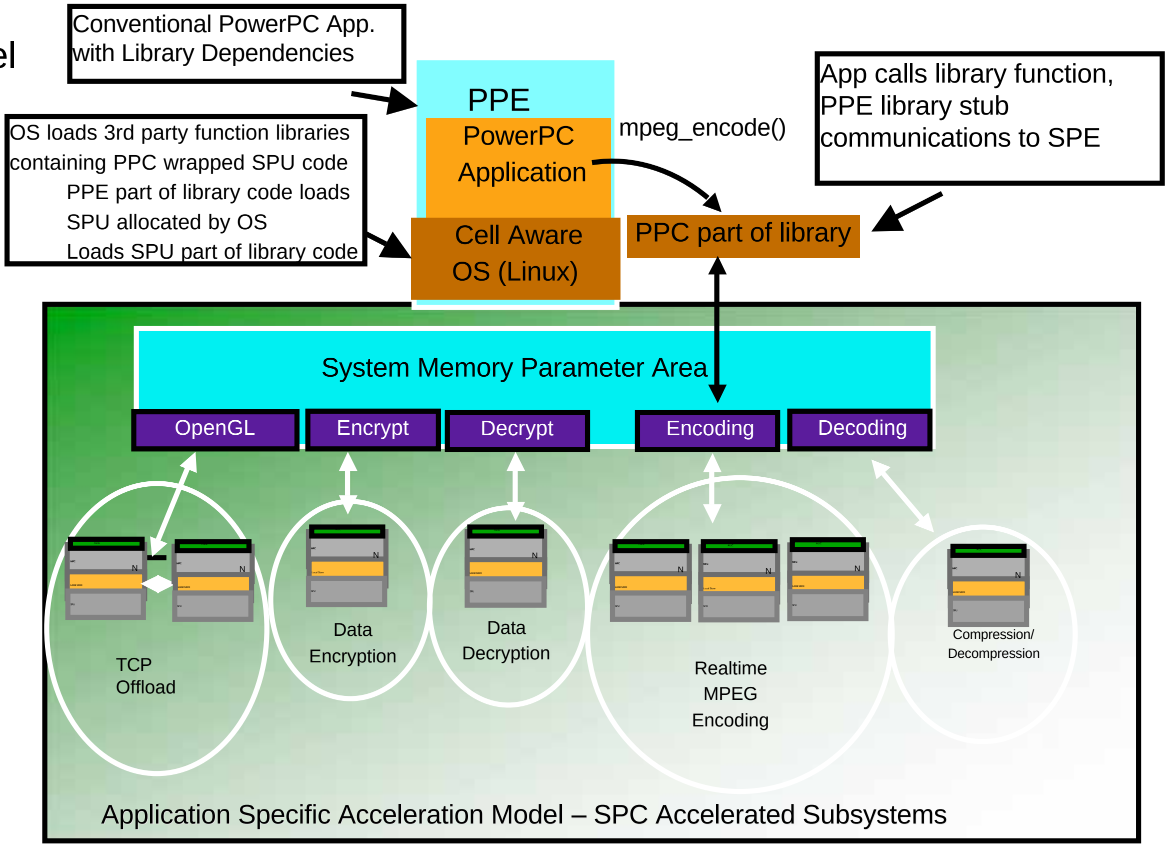
- Dedicated Function (problem/privileged subsystem)
  - Programmer writes/uses SPU "libraries"
    - ➔Graphics Pipeline
    - ➔Audio Processing
    - ➔MPEG Encoding/Decoding
    - ➔Encryption / Decryption
  - Main Application in PPE, invokes SPU bound services
    - ➔RPC Like Function Call
    - ➔I/O Device Like Interface (FIFO/ Command Queue)
  - 1 or more SPUs cooperating in subsystem
    - ➔Problem State (Application Allocated)
      - –Transcoding
      - –Realtime data transformation & streaming
      - –Graphics Processing
      - –MPEG Encoding / Decoding
      - –Physical Simulation
    - ➔Privileged State (OS Allocated)
      - –Encryption / Decryption Services
      - –Network Packet Filtering / Routing
  - Code-to-data or data-to-code pipelining possible
  - Very efficient in real-time data streaming applications





Application Specific Accelerators

- Maintains PowerPC programming model
- Similiar to IOP but integrated with main processor
  - Supports shared memory programming
  - Extremely high on-chip bandwidth
  - Scales with “conventional” processor
- SPU code provided in:
  - Application or middleware libraries
  - Operating System Services
- Does not require application rewrite
  - Specific subsystems targetted
  - Separate compilations
  - Leverage ELF
  - SPUs managed by OS
- SPE Programming localized in library
  - SPU Code vectorization required
  - Private Local Store and DMA model
    - Scheduling of code / data movement
  - Code debug and fault isolation complexity



Conventional PowerPC App. with Library Dependencies

OS loads 3rd party function libraries containing PPC wrapped SPU code
PPE part of library code loads
SPU allocated by OS
Loads SPU part of library code

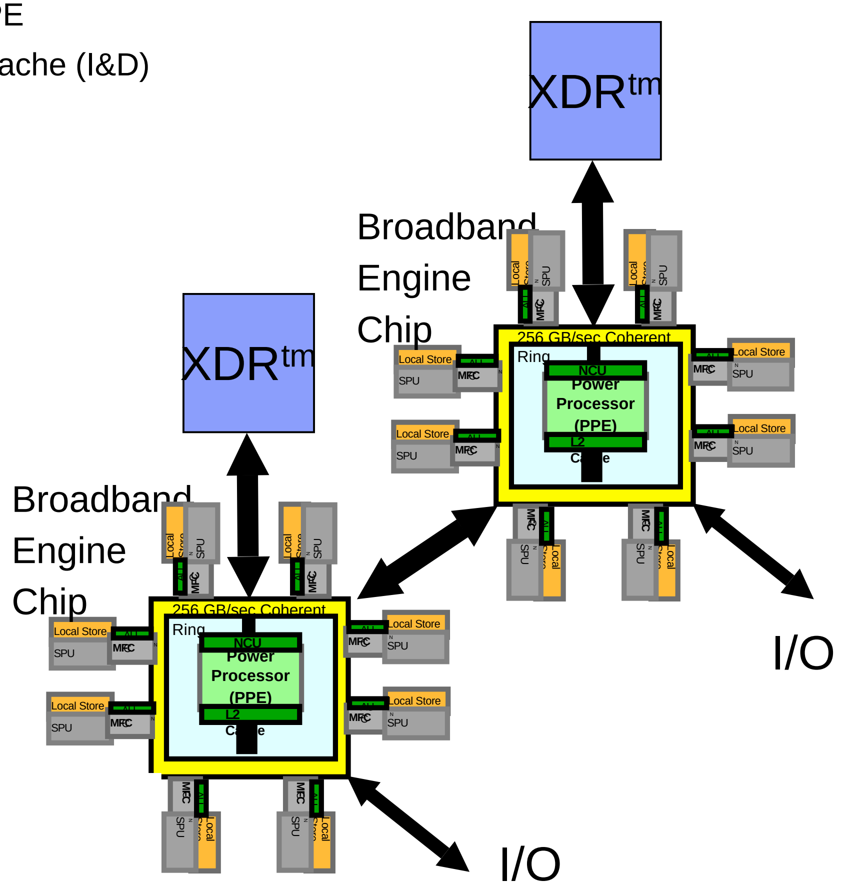
App calls library function, PPE library stub communications to SPE

Application Specific Acceleration Model – SPC Accelerated Subsystems

Parallel Computational Acceleration

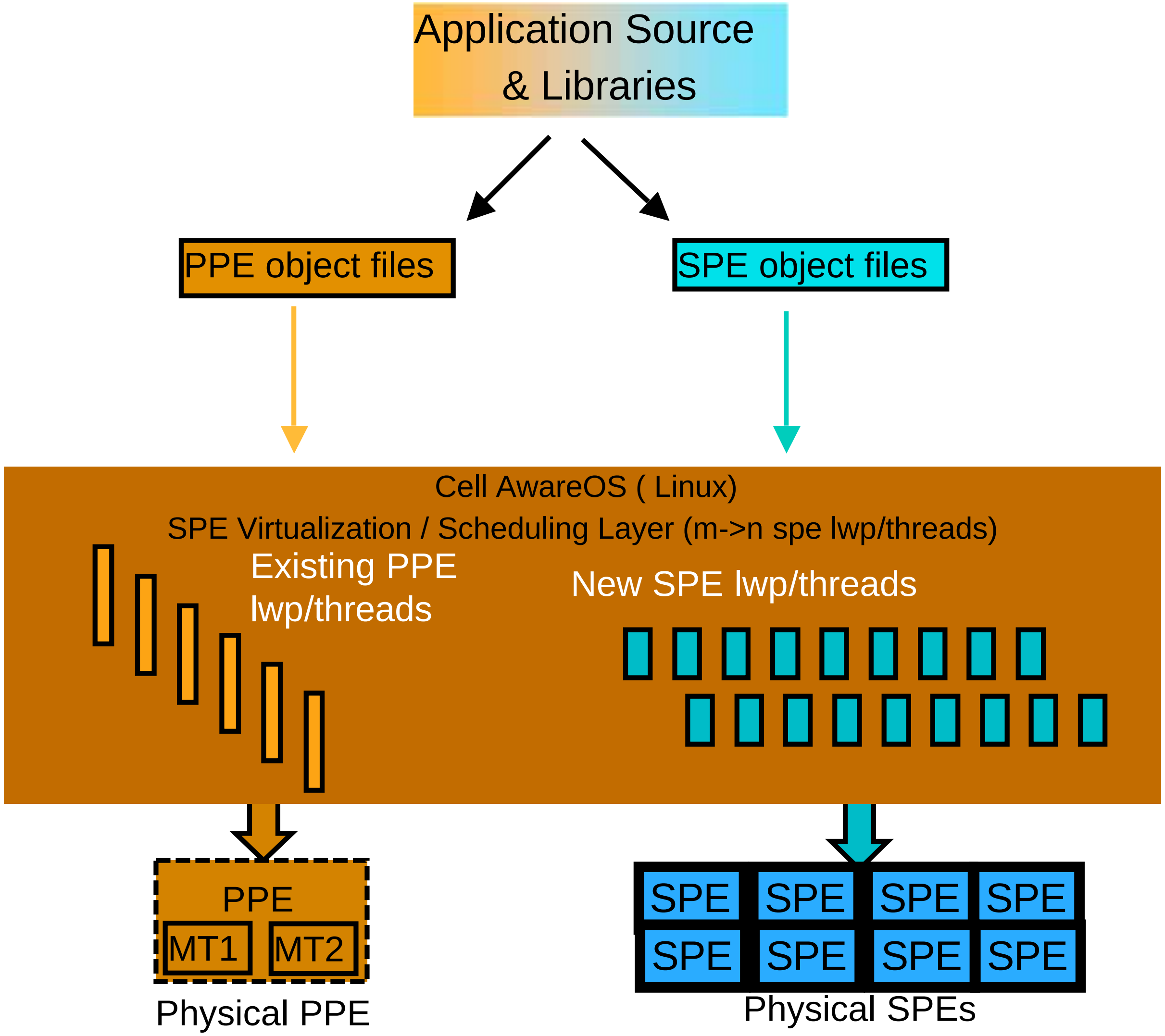
Tools and Programmer Driven

- Single Source Compiler (PPE and SPE targets)
  - Auto parallelization (treat target Cell as an Shared Memory MP)
  - Auto SIMD-ization (SIMD-vectorization) for PPE VMX and SPE
  - Compiler management of Local Store as Software managed cache (I&D)
- Optimization Options
  - OpenMP-like pragmas
  - MPI based Microtasking
  - Streaming languages
  - Vector.org SIMD intrinsics
  - Data/Code partitioning
  - Streaming / pre-specifying code/data use
    - Compiler or Programmer scheduling of DMAs
    - Compiler use of Local store as soft-cache
- IBM Research Prototype Single Source Compiler
  - C Frontend
  - XLC SPE and XLC PPE back-end
- IBM Research Prototype Parallelizing Compiler
  - UPC front-end
  - Fortran front-end
  - XLC SPE backend

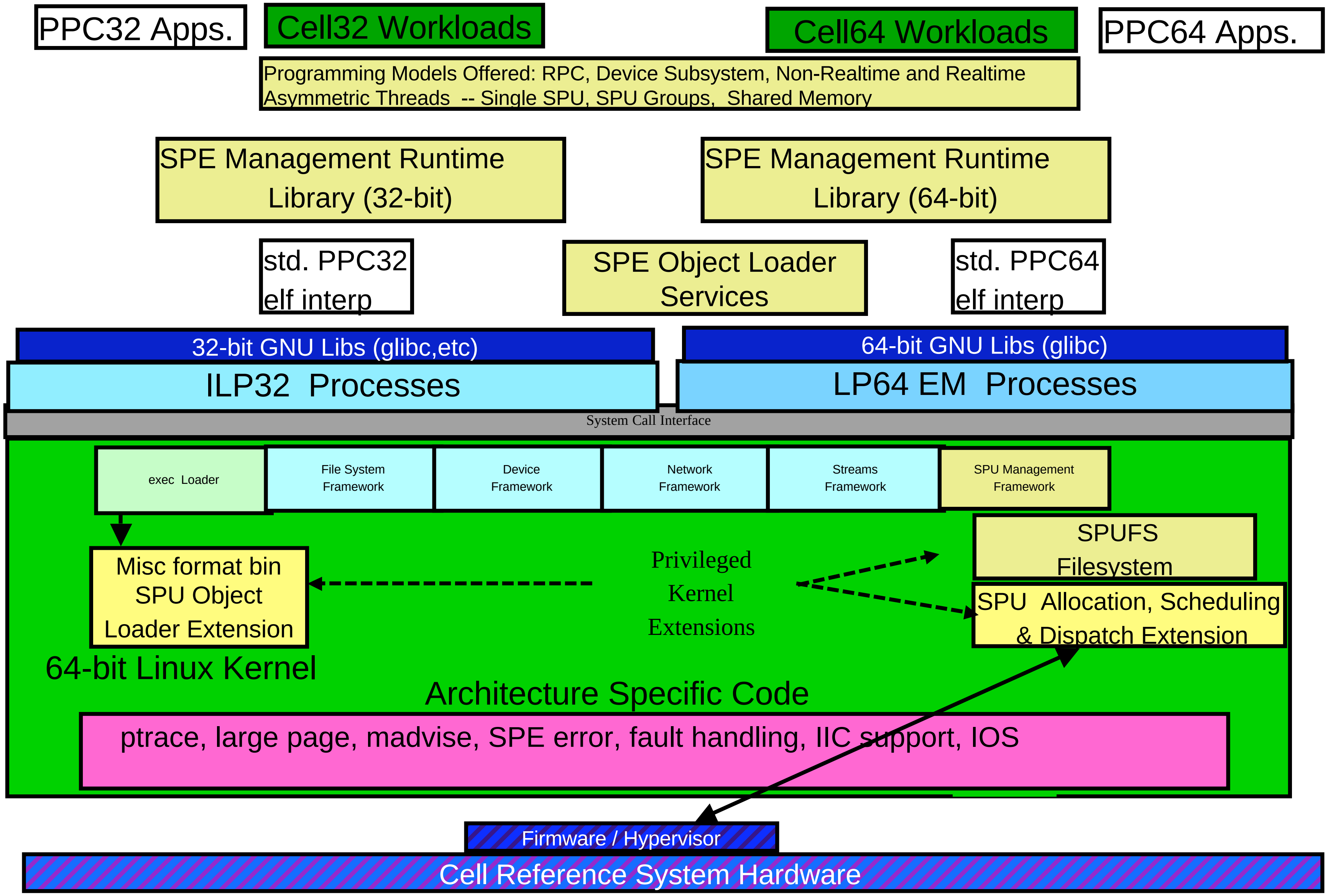


Operating System Runtime Strategy

- Heterogeneous Multi-Threading Model
  - PPE LWP/Threads
  - SPU LWP/Threads
  - SPU DMA EA = PPE Process EA Space
  - OS supports Create/Destroy SPE tasks
  - Atomic Update Primitives used for Mutex
  - SPE Context Fully Managed
    - Context Save/Restore for Debug
    - Virtualization Mode (indirect access)
    - Direct Access Mode (realtime)
  - OS assignment of SPE threads to SPUs
    - Programmer directed using affinity mask



Prototype Cell Extensions to Linux


PPC32 Apps.

Cell32 Workloads

Cell64 Workloads

PPC64 Apps.

Programming Models Offered: RPC, Device Subsystem, Non-Realtime and Realtime Asymmetric Threads -- Single SPU, SPU Groups, Shared Memory

SPE Management Runtime Library (32-bit)

SPE Management Runtime Library (64-bit)

std. PPC32 elf interp

SPE Object Loader Services

std. PPC64 elf interp

32-bit GNU Libs (glibc,etc)

64-bit GNU Libs (glibc)

ILP32 Processes

LP64 EM Processes

System Call Interface

exec Loader

File System Framework

Device Framework

Network Framework

Streams Framework

SPU Management Framework

SPUFS Filesystem

Misc format bin SPU Object Loader Extension

Privileged Kernel Extensions

SPU Allocation, Scheduling & Dispatch Extension

64-bit Linux Kernel

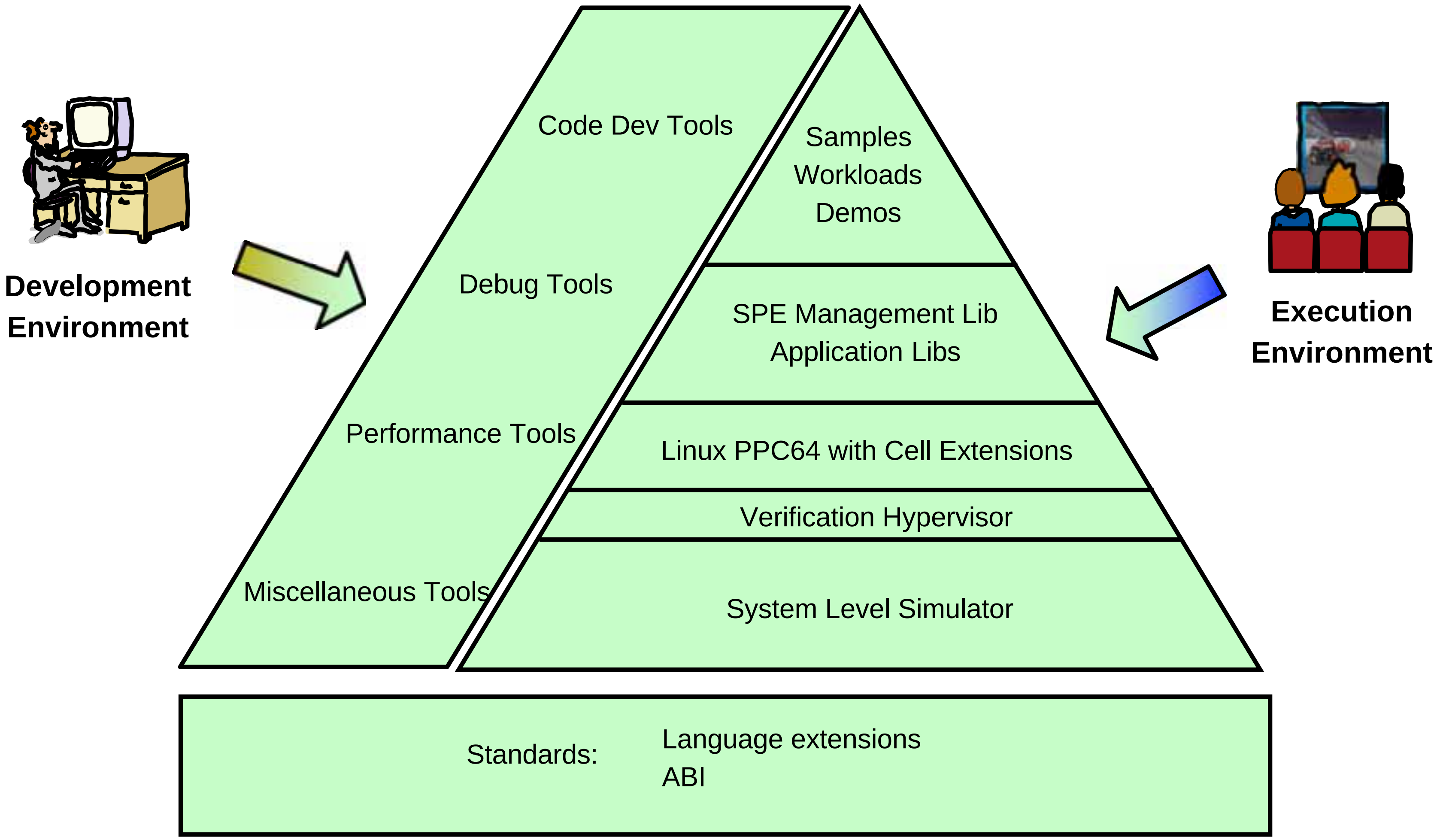
Architecture Specific Code

ptrace, large page, madvise, SPE error, fault handling, IIC support, IOS

Firmware / Hypervisor

Cell Reference System Hardware

Cell Prototype Software Environment



Development Environment

Execution Environment

Code Dev Tools

Debug Tools

Performance Tools

Miscellaneous Tools

Samples
Workloads
Demos

SPE Management Lib
Application Libs

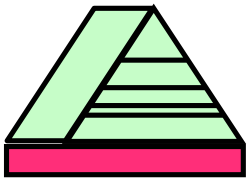
Linux PPC64 with Cell Extensions

Verification Hypervisor

System Level Simulator

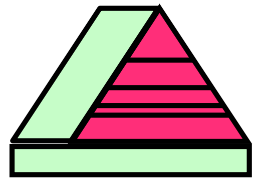
Standards: Language extensions
ABI

Cell Standards



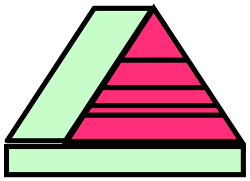
- Application Binary Interface Specifications
  - Defines such things as data types, register usage, calling conventions, and object formats to ensure compatibility of code generators and portability of code.
    - SPE ABI
    - Linux Cell ABI
- SPE C/C++ Language Extensions
  - Defines standardized data types, compiler directives, and language intrinsics used to exploit SIMD capabilities in the core.
  - Data types and Intrinsics styled to be similar to Altivec/VMX.
- SPE Assembly Language Specification

System Level Simulator



- Cell BE – full system simulator
  - Uni-Cell and multi-Cell simulation
  - User Interfaces – TCL and GUI
  - Cycle accurate SPU simulation (pipeline mode)
  - Pseudo accurate memory and MFC modes
  - Emitter facility for tracing and viewing simulation events
- Other simulators
  - spusim – standalone SPU simulator

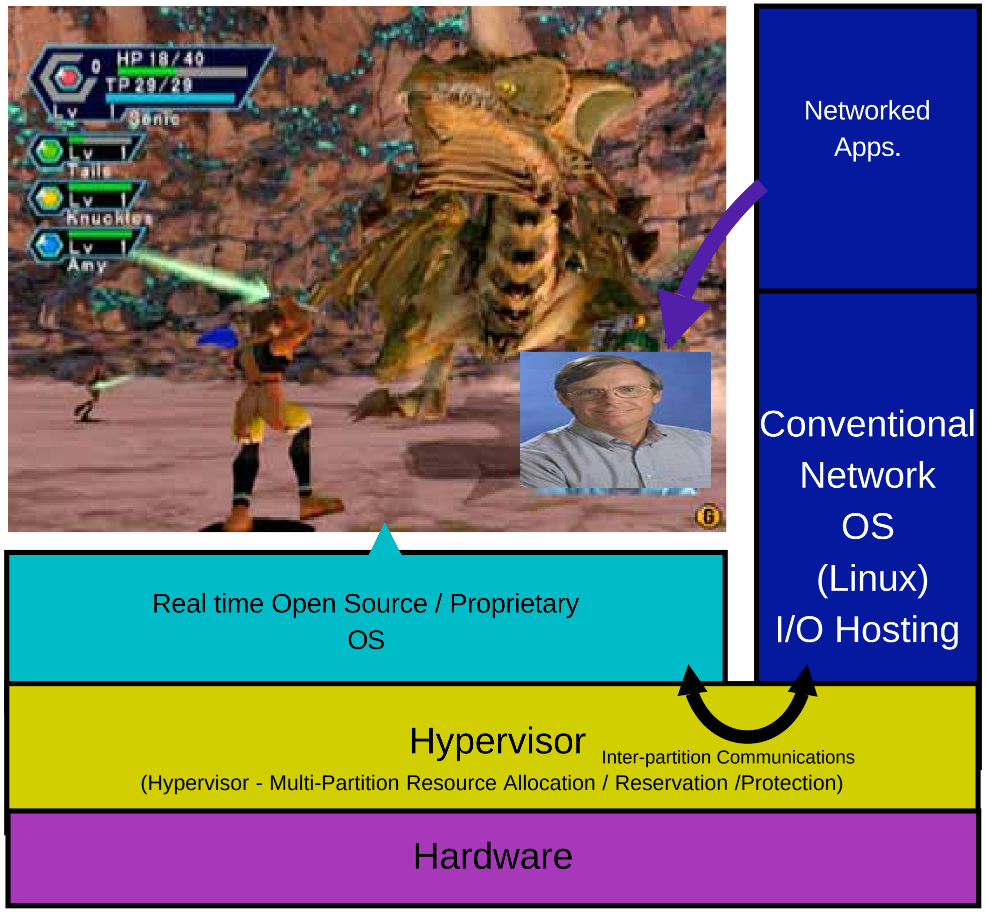
Verification Hypervisor (vHype)



Execution Environment

- Seamless Integration of Dual Environments
  - •Low overhead – small footprint
  - •Realtime Resource management
  - •SPE management
  - •Separate policy manager
  - •Pre-emptive partition switching on high priority interrupts

Logical Partitioning RTOS/Linux



Networked Apps.

Conventional Network OS (Linux) I/O Hosting

Real time Open Source / Proprietary OS

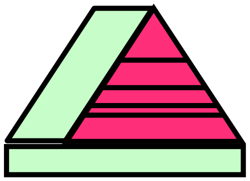
Hypervisor

Inter-partition Communications

(Hypervisor - Multi-Partition Resource Allocation / Reservation /Protection)

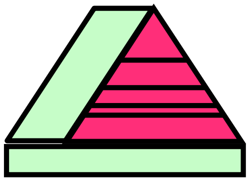
Hardware

Linux on Cell



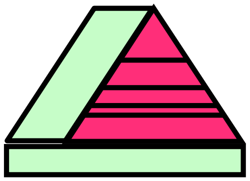
- All software in STIDC written on Linux OS
  - Started with Linux 2.4 PPC64 on Cell Simulator
    - SPEs exposed as I/O Devices (function offload model)
    - SPE DMA required pre-pinned memory
    - Inflexible programming model
- Moved to 2.6.3
  - Added heterogenous lwp/thread model – via system call – *moved to SPUFS in 2.6.13*
    - SPE thread API created (similar to pthreads library)
    - User mode direct and indirect SPE access models
    - Full pre-emptive SPE context management
    - spe_ptrace() added for gdb support
    - spe_schedule() for thread to physical SPE assignment
      currently FIFO – run to completion
  - SPE threads share address space with parent PPE process (through DMA)
    - Demand paging for SPE accesses
    - Shared hardware page table with PPE
  - SPE Error, Event and Signal handling directed to parent PPE thread
  - SPE elf objects wrapped into PPE shared objects with extended gld
    - SPE-side mini-loader
  - madvise() extended for L2 cache and TLB locking/preloading (realtime feature)
  - All patches for Cell in architecture dependent layer (subtree of PPC64)
- Publishing Initial CellBE Patches for 2.6.13 (Fall 2005 target)

SPE Management Library



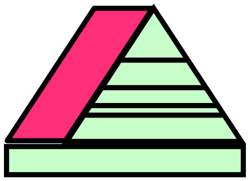
- SPEs are exposed as threads
  - SPE thread model interface is similar to POSIX threads.
  - SPE thread consists of the local store, register file, program counter, and MFC-DMA queue.
  - Associated with a single Linux task.
  - Features include:
    - **Threads** - create, groups, wait, kill, set affinity, set context
    - **Thread Queries** - get local store pointer, get problem state pointer, get affinity, get context
    - **Groups** - create, set group defaults, destroy, memory map/unmap, madvise.
    - **Group Queries** - get priority, get policy, get threads, get max threads per group, get events.
    - **SPE image files** - opening and closing
- SPE Executable
  - Standalone SPE program managed by a PPE executive.
  - Executive responsible for loading and executing SPE program. It also services “syscall” requests for I/O (eg, fopen, fwrite, fprintf) and memory requests (eg, mmap, shmat, …).

Optimized Prototype Libraries



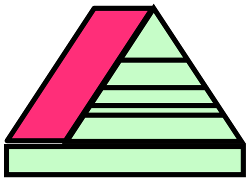
- Audio (resampling)
- Cryptographic
- Fast fourier transform
- Game math
- Image
- Large matrix
- Math
- Matrix (4x4)
- Miscellaneous
- Memory management
- Multi-precision math
- Noise & Turbulence
- Oscillator
- Parallel programming (MPI like)
- Shared memory
- SPU plugin
- SPU system call
- Surfaces / curves
- Synchronization
- Vector

Code Development Tools



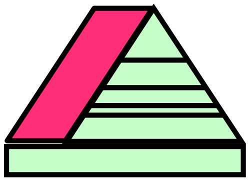
- GNU based binutils
  - gas SPE assembler
  - gld SPE ELF object linker
    - gld extensions for embedding SPE object modules in PPE executables
  - misc bin utils (ar, nm, ...) targeting SPE modules
  - hosted on Linux IA32, Linux PowerPC
- GNU based C/C++ compiler targeting SPE
  - From STI Partner
  - retargeted compiler to SPE
  - Supports common SPE Language Extensions and ABI (ELF/Dwarf2) object output
- Cell Broadband Engine Optimizing Compiler (IBM Proprietary)
  - IBM XLC C/C++ for PowerPC (Tobey)
  - IBM XLC C retargeted to SPE assembler (including vector intrinsics) - highly optimizing
  - Prototype - XLC Compiler supporting CellBE Programmer Productivity Aids
    - Single Source compilation using OpenMP like pragmas (PPE and SPE object code generated)
    - Auto-Vectorization (auto-SIMD) for SPE code
    - Auto-Parallelization across SPEs
    - UPC Front end – parallelization across SPEs
    - Local Store software managed caching model
  - Hosted on Linux
  - Executables to be available on IBM Alphaworks – Fall 2005

Debug Tools



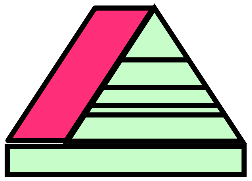
- CellBE system simulator
  - Executable availability on AlphaWorks (Fall 2005 target)
- GNU gdb
  - ptrace and spe_ptrace enabled
  - Multi-core Application source level debugger supporting PPE multithreading, SPE multithreading, interacting PPE and SPE threads
  - Three modes of debugging SPU threads
    - Attach to SPE thread
    - Launch mode – launch a new debug session for each SPE thread
    - Pass-thru mode – follow execution into SPE thread
- RISCwatch
  - Low level hardware (JTAG) debugger

Prototype Performance Tools



- pmcount
  - Tool to access to HW performance counters
- Performance inspector
  - Suite of GPL based performance analysis tools extended to support SPE threads
    - tprof – timer based analysis tool
    - ptt – per thread time
    - ai – above idle
    - post – report generator
    - a2n – address to name
- ctrace
  - Branch tracing performance monitor (under development)

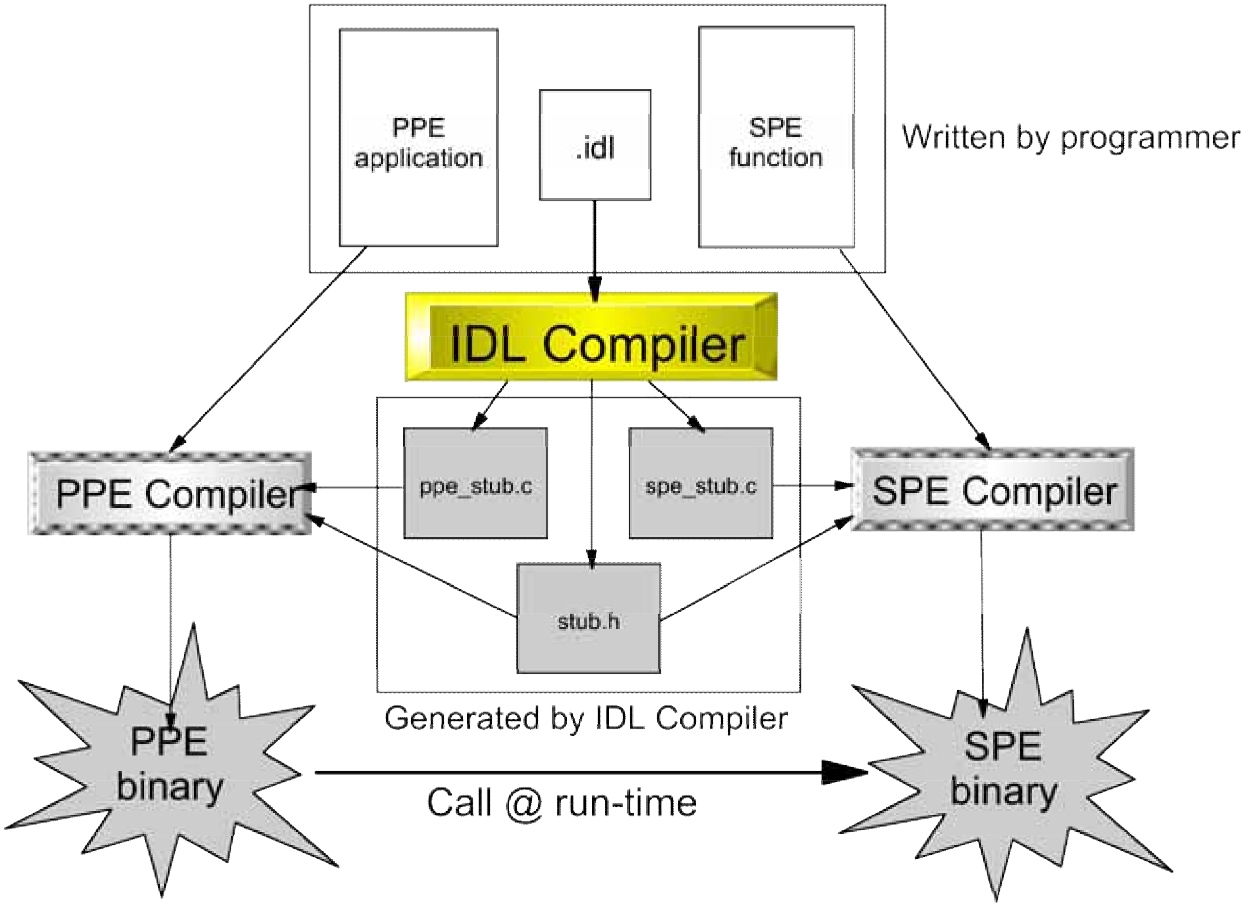
SPE Performance Tools



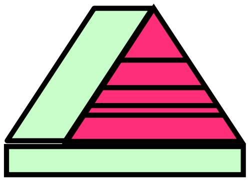
- Static analysis (spexlc_timing)
  - Annotates assembly source with instruction pipeline state
- Dynamic analysis (CellBE System Simulator)
  - Generates statistical data on SPE execution
    - Cycles, instructions, and CPI
    - Single/Dual issue rates
    - Stall statistics
    - Register usage
    - Instruction histogramming

Miscellaneous Tools – IDL Compiler

Development Environment



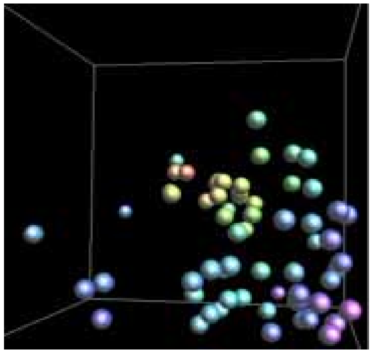
Samples / Workloads / Demos



Execution Environment

- Numerous code samples provided to demonstrate system design constructs
- Complex workloads and demos used to evaluate and demonstrate system performance

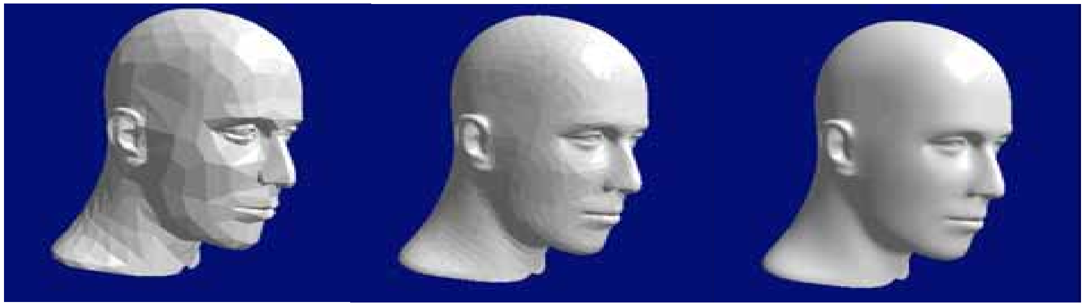
Geometry Engine



Physics Simulation

Terrain Rendering Engine

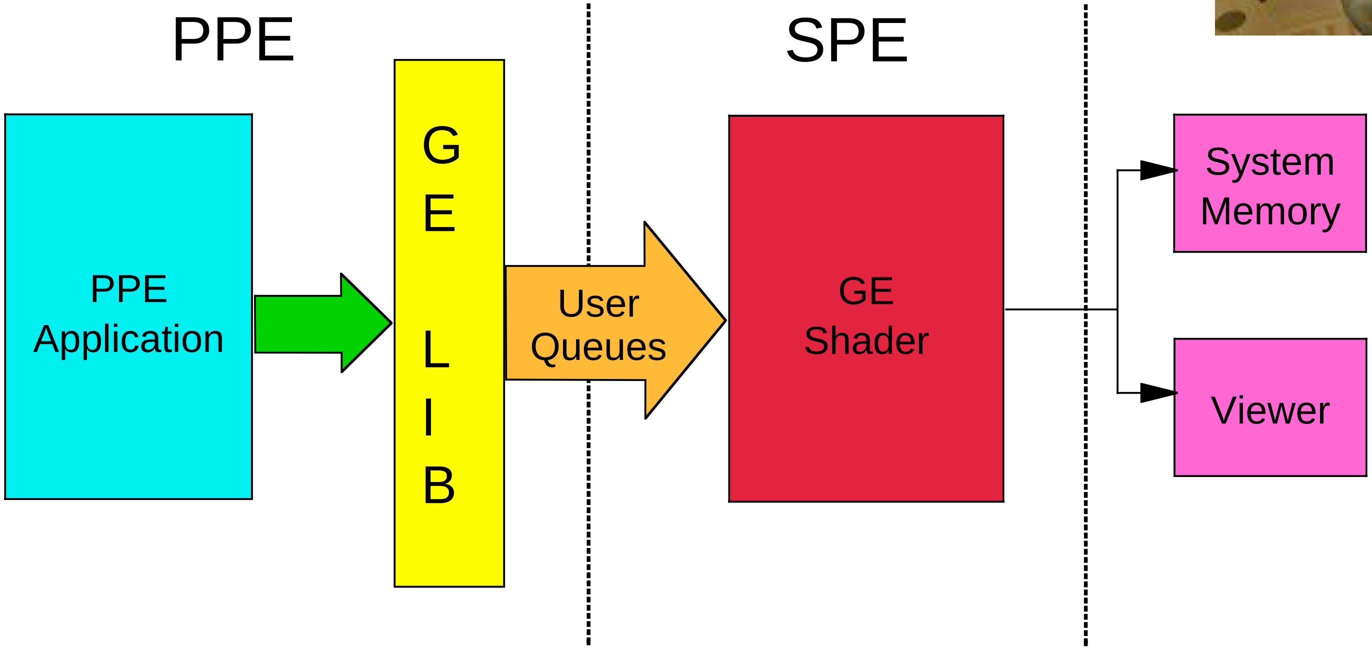
Subdivision Surfaces



Subsystem Sample – Geometry Engine

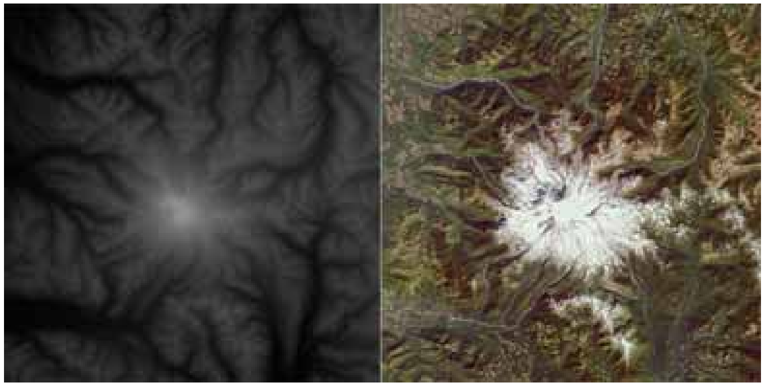
Execution Environment

- OpenGL-like geometry engine
  - Geometry processing is offloaded to compile-time configurable SPE “vertex shader”
  - *User Queue* communication model consisting of 4KB blocks for SPU command requests with command headers in SPE Mailbox FIFO

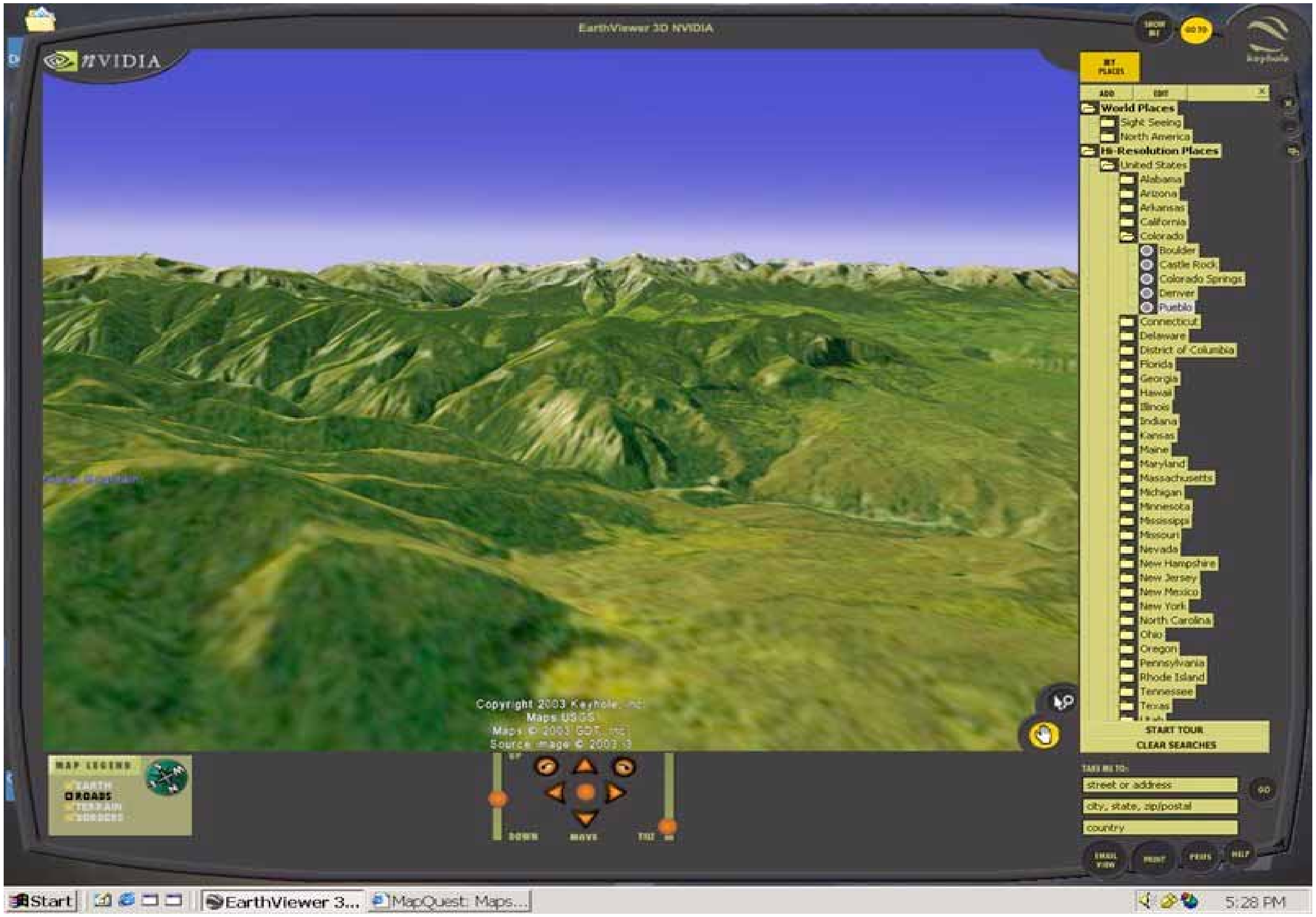


Terrain Rendering Engine Overview

- Visualization of Terrain Data Increasingly Important
  - General availability of high resolution satellite images
  - Publicly available USGS Digital Elevation Models (DEMs)
  - Mobile GPS devices (land, sea, air) + wireless networks
- Inferior Current Solutions
  - Polygonal Models (low quality, non-Real Time)
  - Requires CPU + GPU (Graphics Processing Unit)
- Superior CellBE Solution
  - Highly Compressed Height Map Models (10x less data)
  - Requires only one CellBE (No GPU)
  - High Quality Images (Multi-Sampled Raycast)
  - Fast (Real Time Animations)

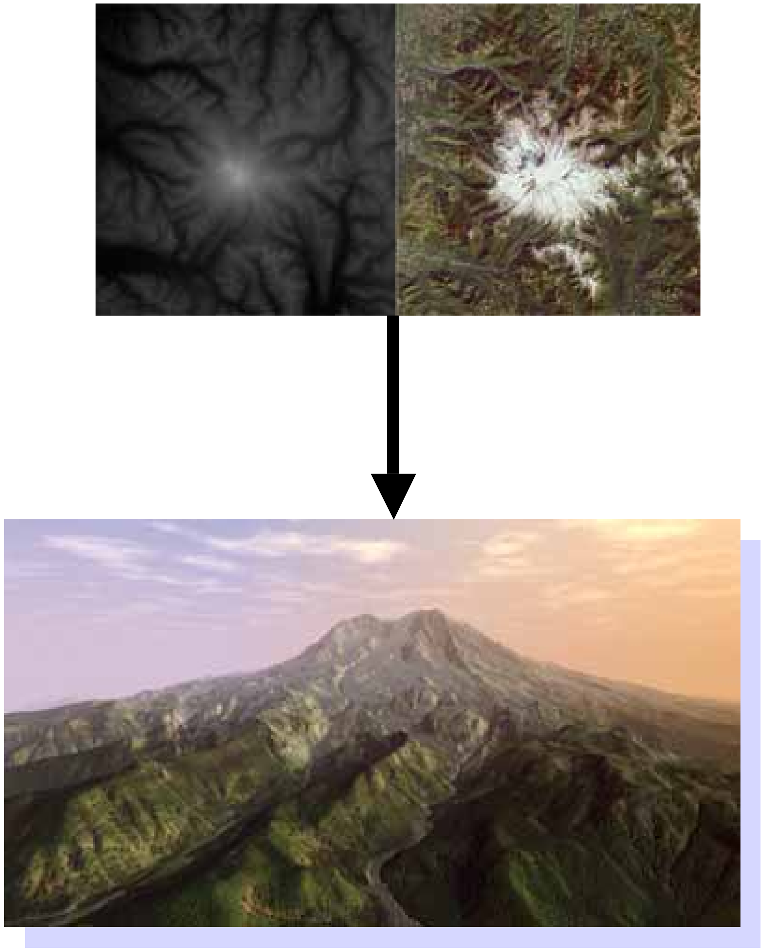


Earthviewer – Polygon Render

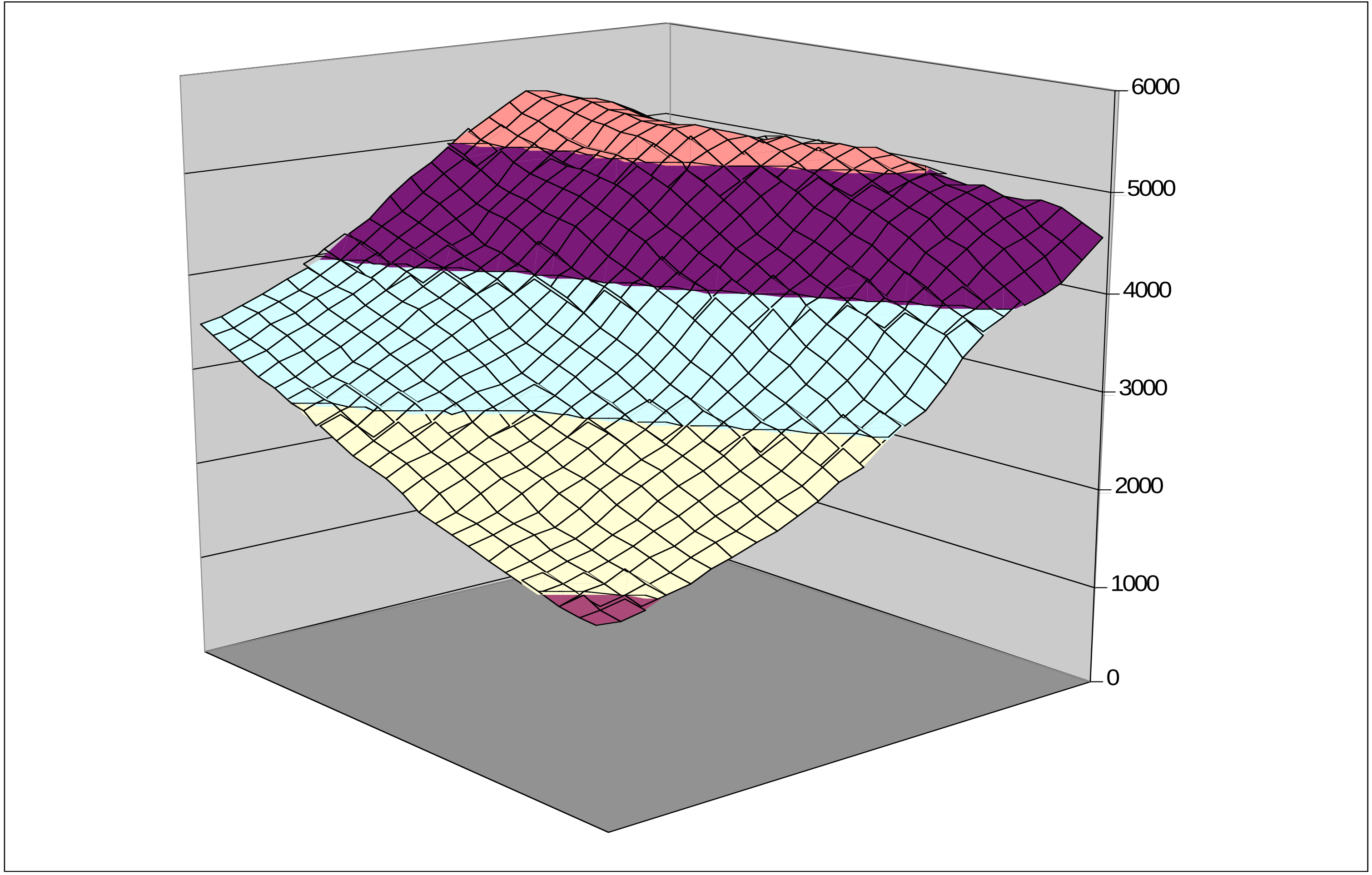


Real-Time Terrain Rendering Engine (TRE)
Cell Optimized Raycaster

- Advanced SPU shader function
- Real time rendering with only one Cell processor
  - No graphics adapter assist
  - High definition resolution
  - Ray/Terrain intersection computation
  - Texture Filtering
  - Normal computation
  - Bump map computation
  - Diffuse + Ambient lighting model
  - Perlin Noise based clouds
  - Atmosphere computation (haze, sun, halo)
  - Dynamic multi-sampling (4 – 16 samples per pixel)
  - Image based input (16 bit height + 16 bit texture)
  - 47 KB of SPU object code
- MJPEG like compression via SPU
- Performance scales linearly with number of available SPUs
- Written completely in C with intrinsics
- Client support – OpenGL workstation, wireless PDA
- User inputs – graphical, joystick, GPS/accelerometer

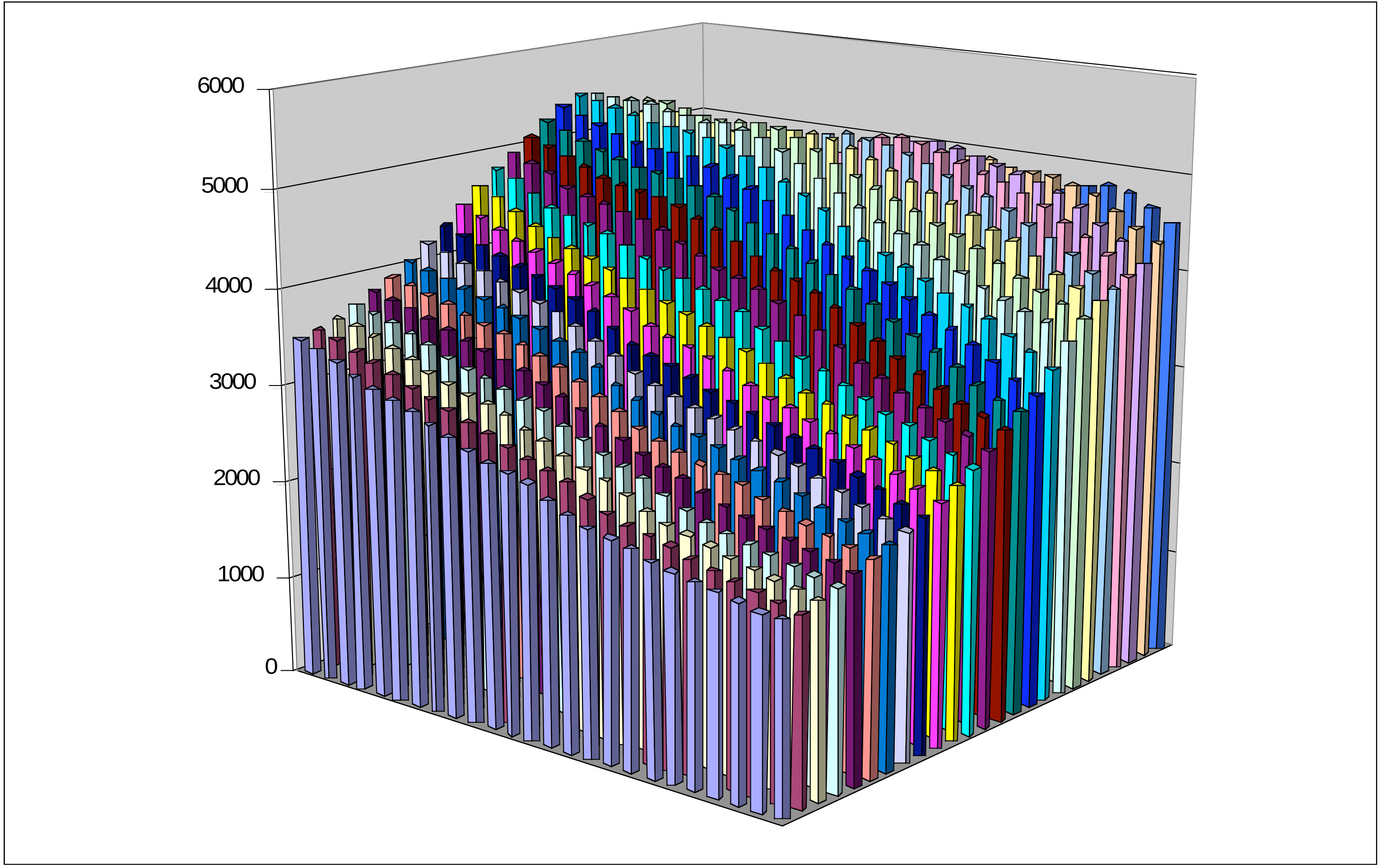


3D Surface from 25x25 Height Data



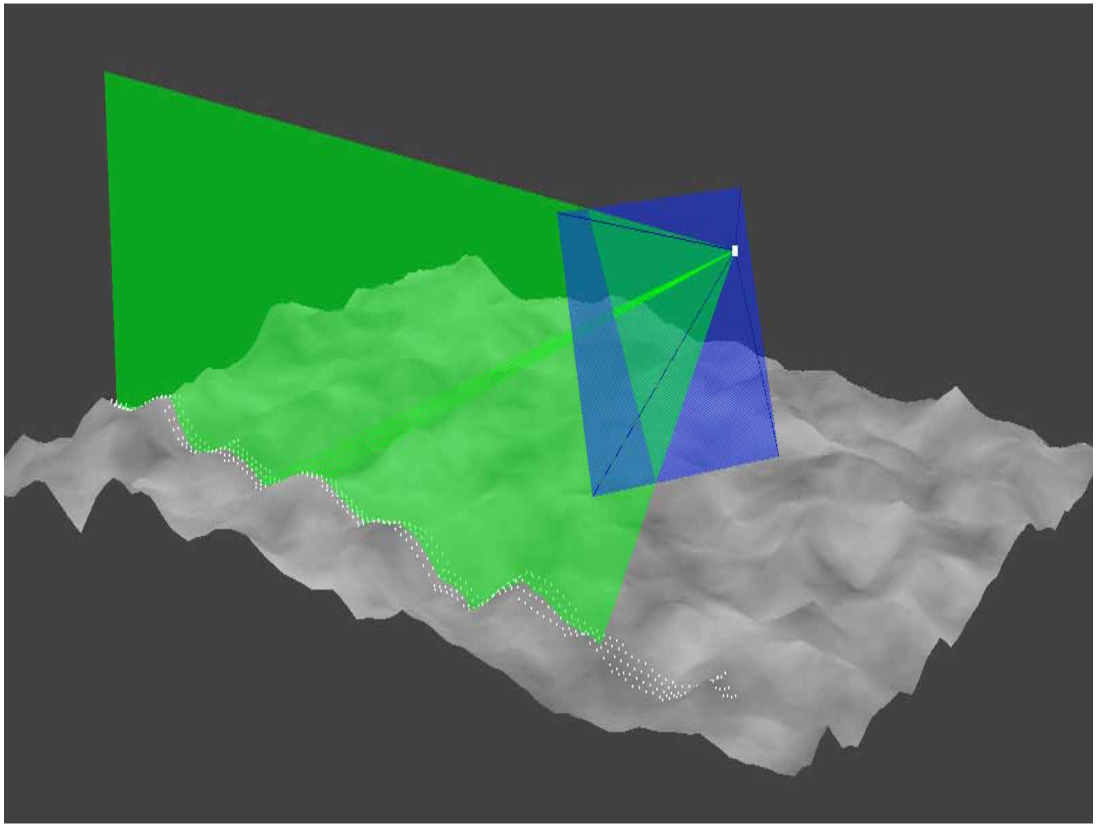
7.5 KB for just the Surface

Raw 25x25 Height Data

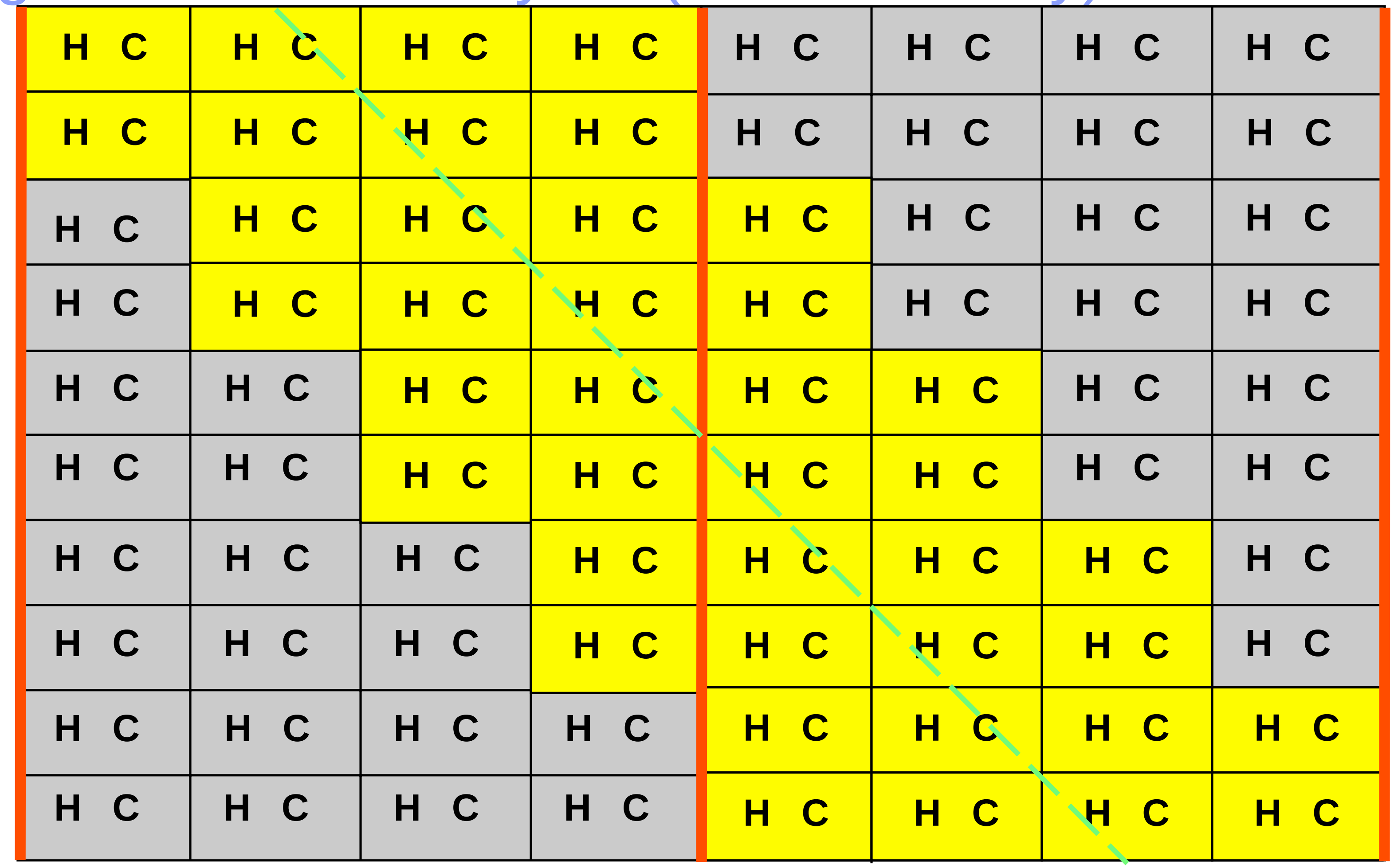


1.25 KB for Height Map (6x Compression)

Ray Casting

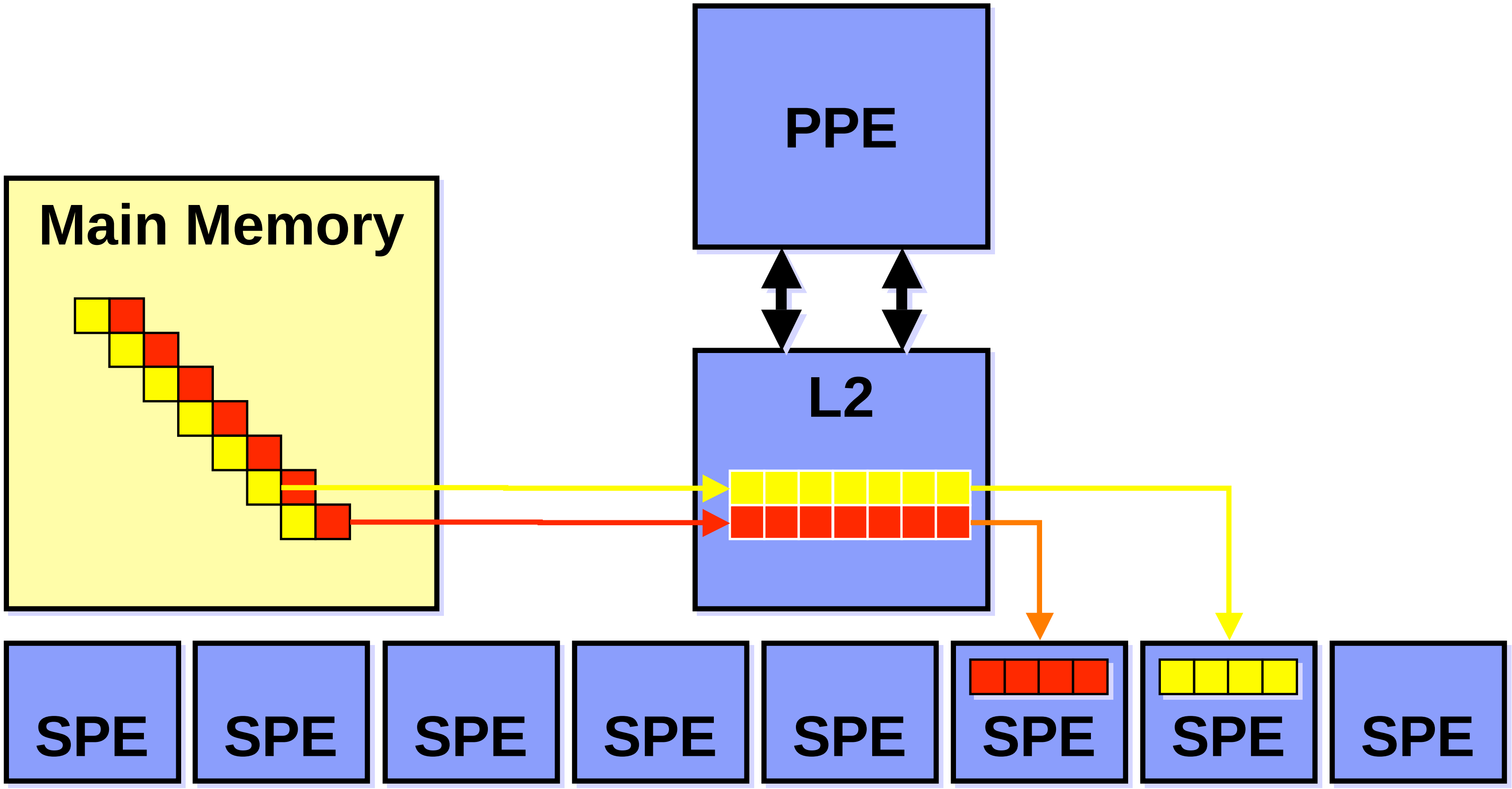


Height Color Data Layout (Main Memory)



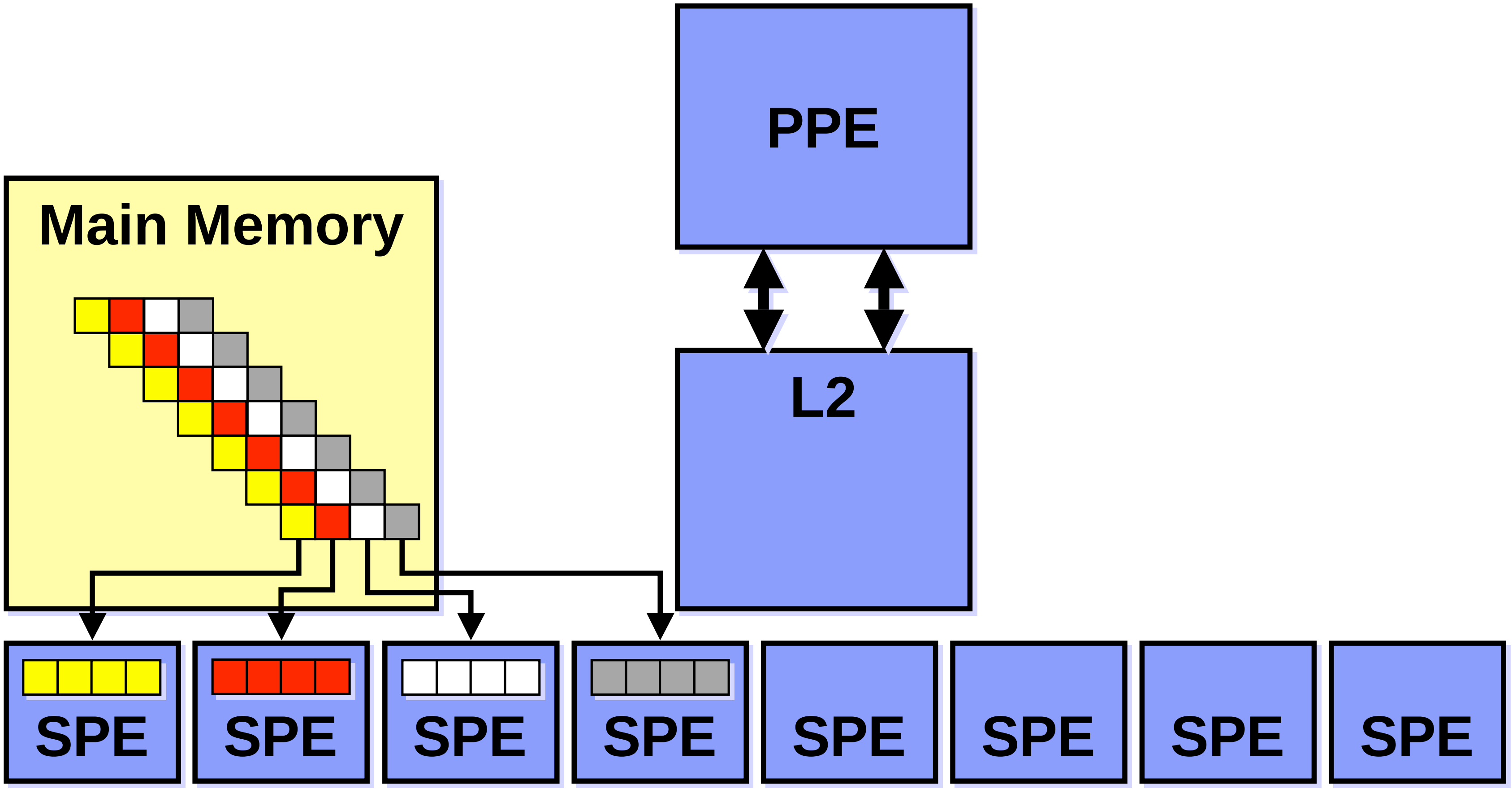
H C	H C	H C	H C	H C	H C	H C	H C
H C	H C	H C	H C	H C	H C	H C	H C
H C	H C	H C	H C	H C	H C	H C	H C
H C	H C	H C	H C	H C	H C	H C	H C
H C	H C	H C	H C	H C	H C	H C	H C
H C	H C	H C	H C	H C	H C	H C	H C
H C	H C	H C	H C	H C	H C	H C	H C
H C	H C	H C	H C	H C	H C	H C	H C
H C	H C	H C	H C	H C	H C	H C	H C
H C	H C	H C	H C	H C	H C	H C	H C

PPE Data Staging via L2



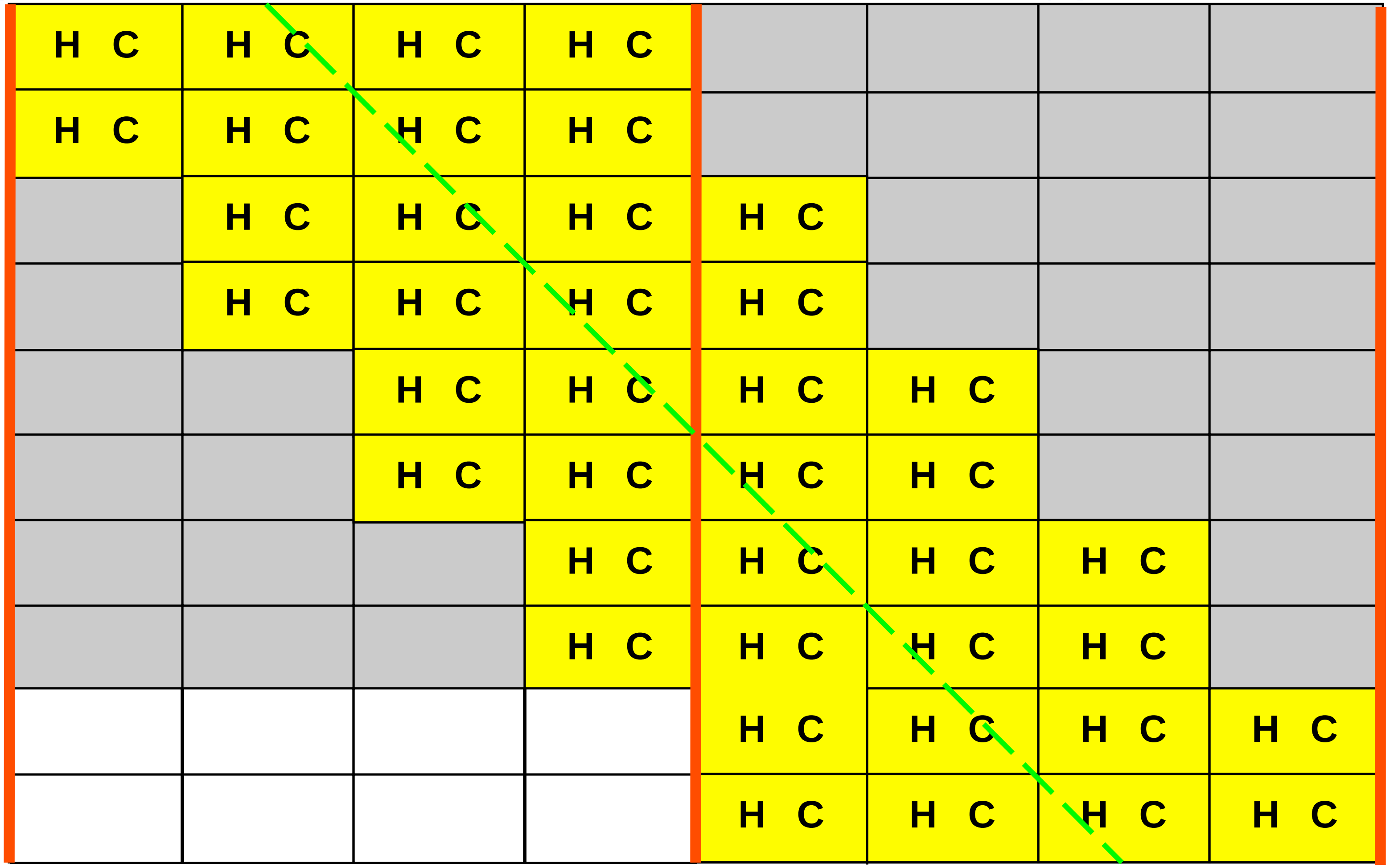
The L2 has 4 Outstanding Loads + 2 Prefetch

SPE Data Staging



16 Outstanding Loads per SPE

Height Color Data Layout (Main Memory)

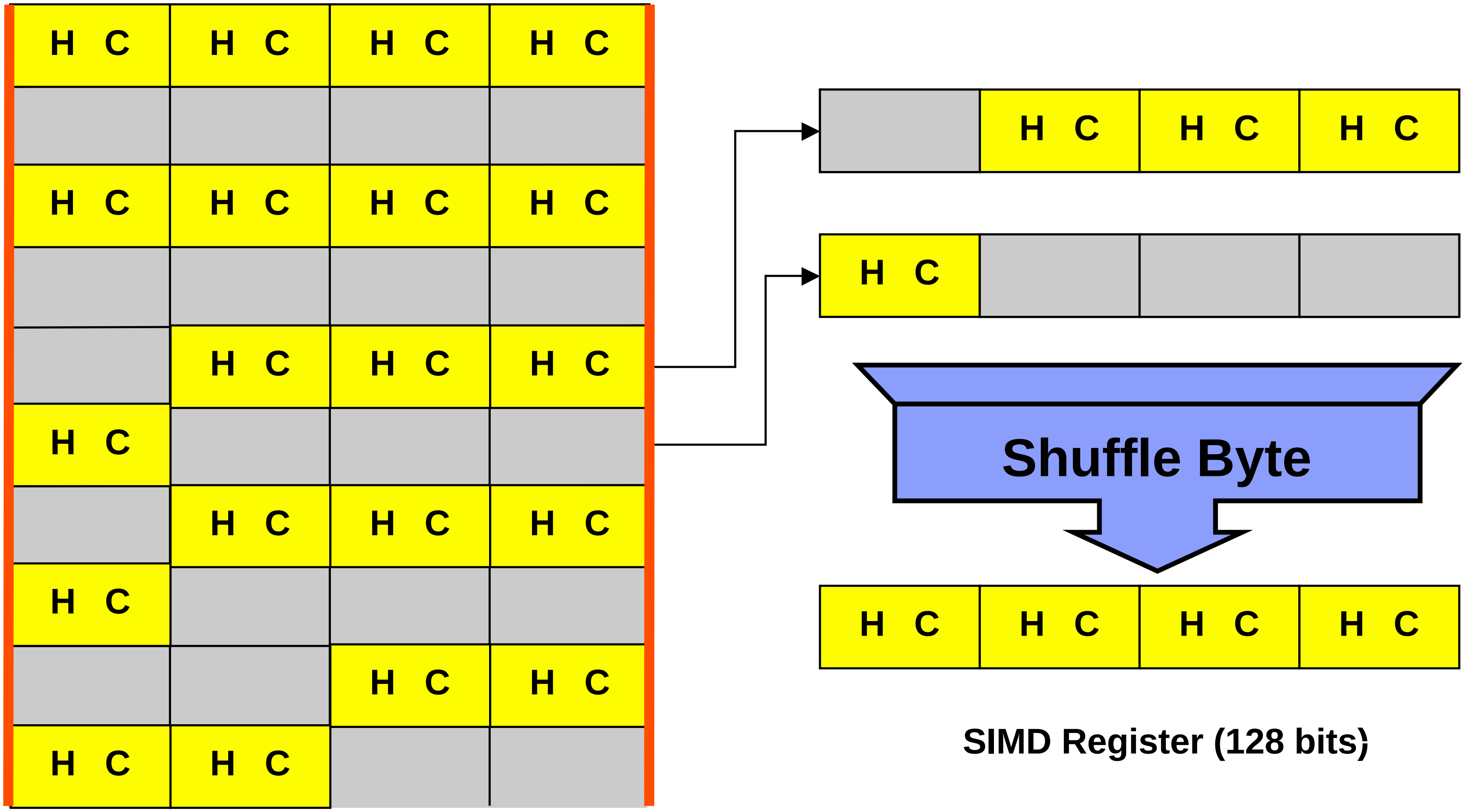


H C	H C	H C	H C				
H C	H C	H C	H C				
	H C	H C	H C	H C			
	H C	H C	H C	H C			
		H C	H C	H C	H C		
		H C	H C	H C	H C		
			H C	H C	H C	H C	
			H C	H C	H C	H C	
				H C	H C	H C	H C
				H C	H C	H C	H C

Quad Word (128 bits)

Quad Word (128 bits)

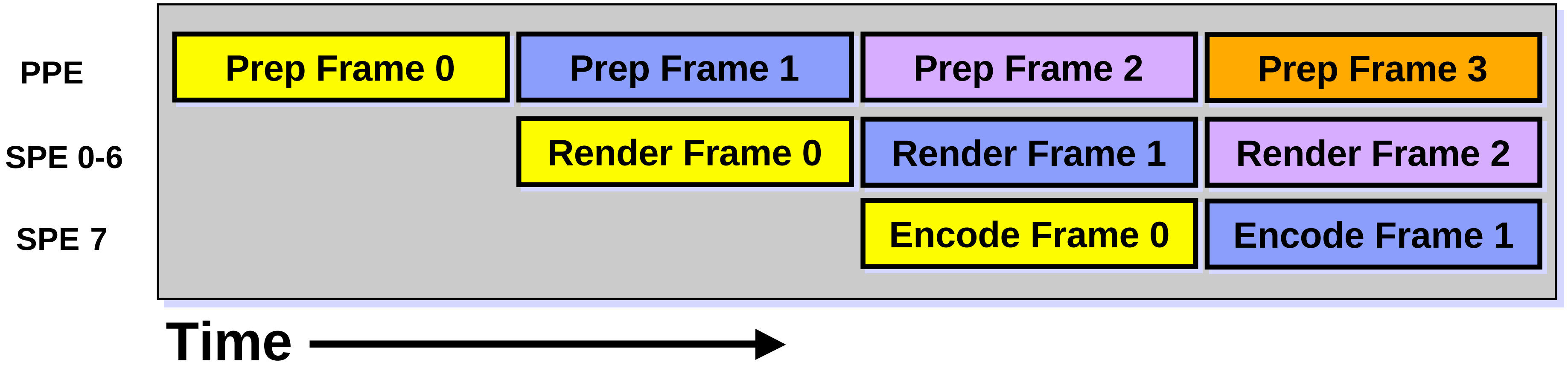
Height Color Data Layout (Local Store)



SIMD Register (128 bits)

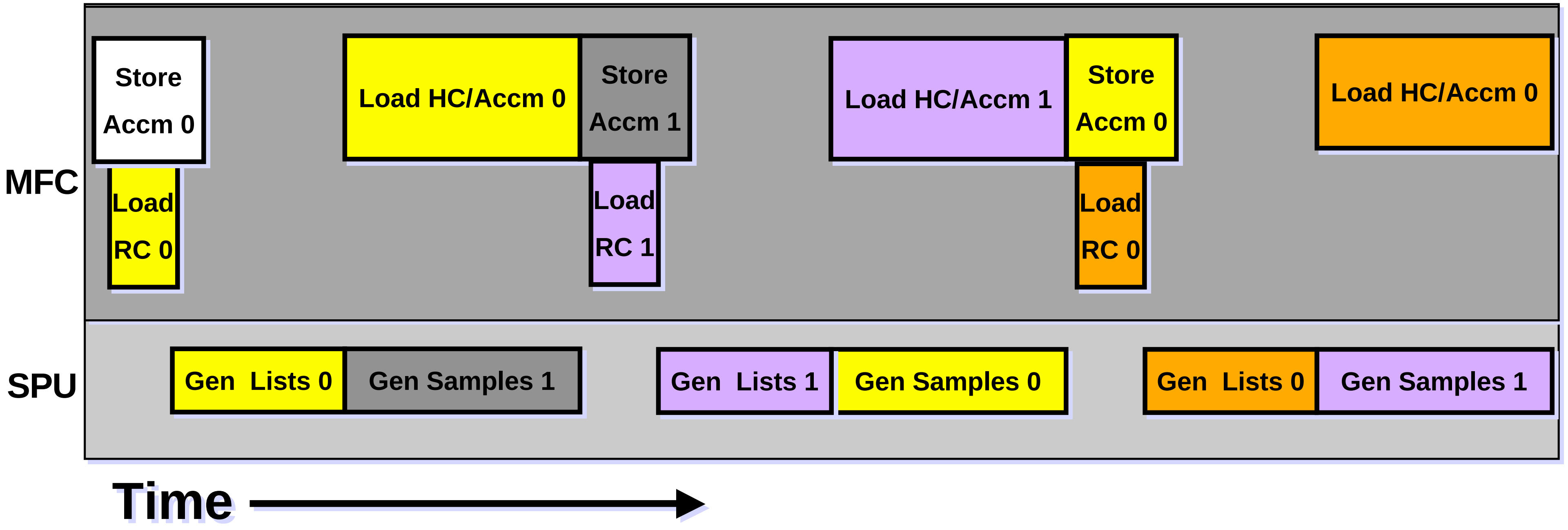
Quad Word (128 bits)

TRE Image Pipeline

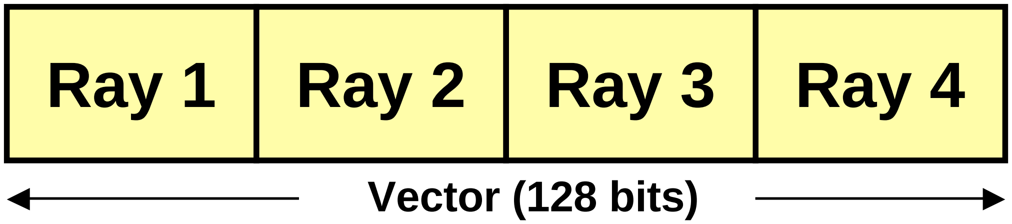


PPE	Prep Frame 0	Prep Frame 1	Prep Frame 2	Prep Frame 3
SPE 0-6		Render Frame 0	Render Frame 1	Render Frame 2
SPE 7			Encode Frame 0	Encode Frame 1

TRE SPE Render Pipeline

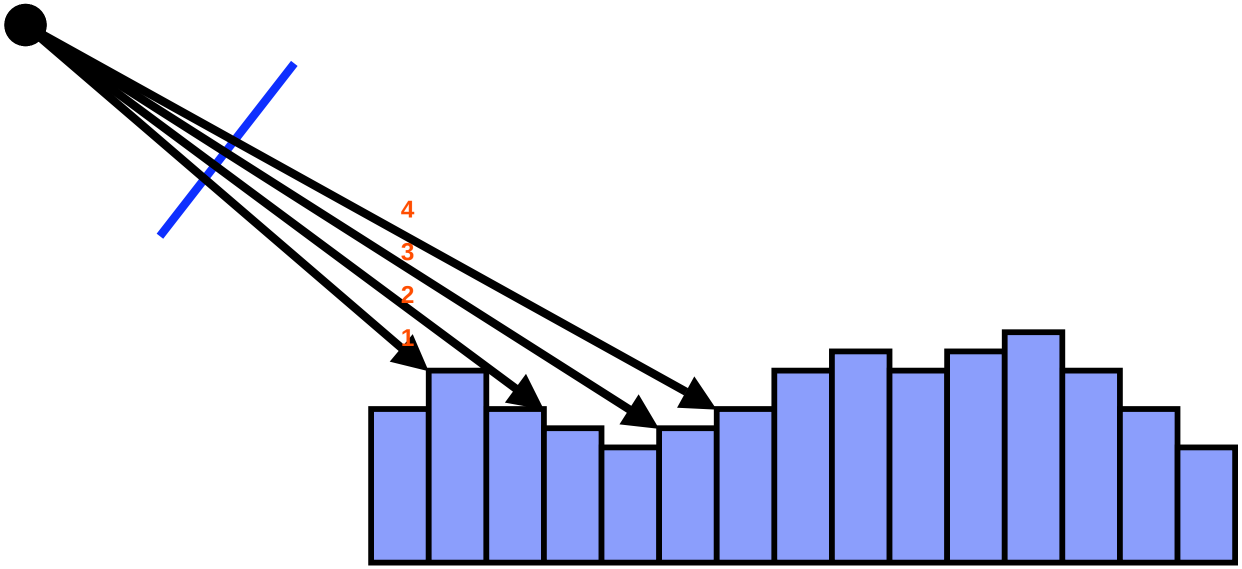


Ray-casting with SIMD

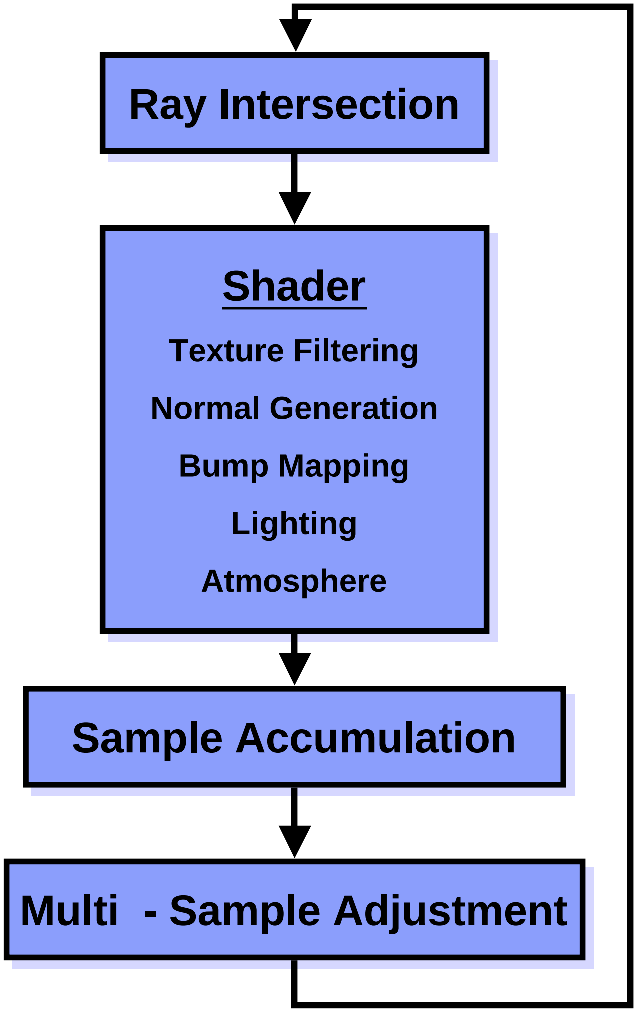


Ray 1 | Ray 2 | Ray 3 | Ray 4

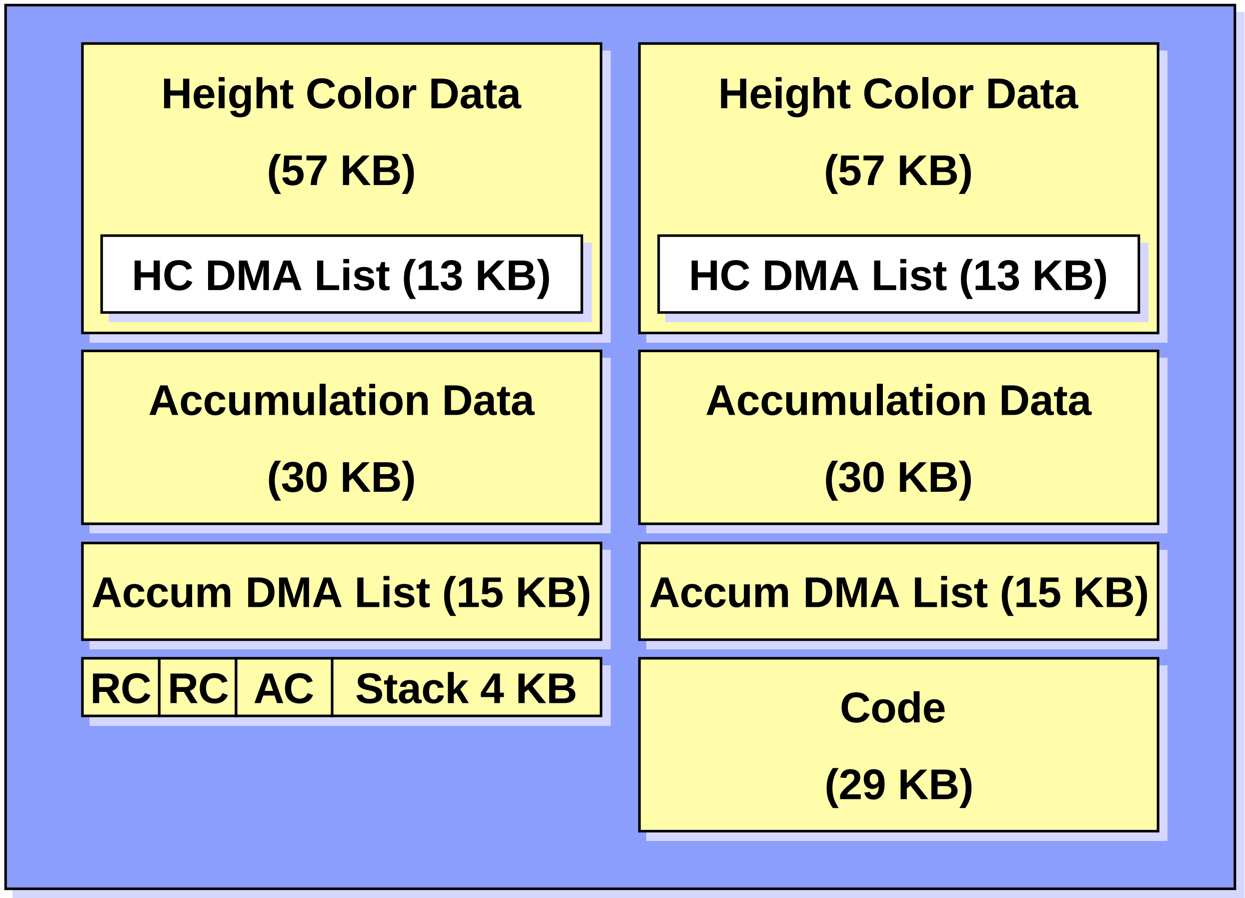
Vector (128 bits)



TRE SPE Ray Kernel



SPE Local Store Memory Layout



Height Color Data

(57 KB)

HC DMA List (13 KB)

Height Color Data

(57 KB)

HC DMA List (13 KB)

Accumulation Data

(30 KB)

Accumulation Data

(30 KB)

Accum DMA List (15 KB)

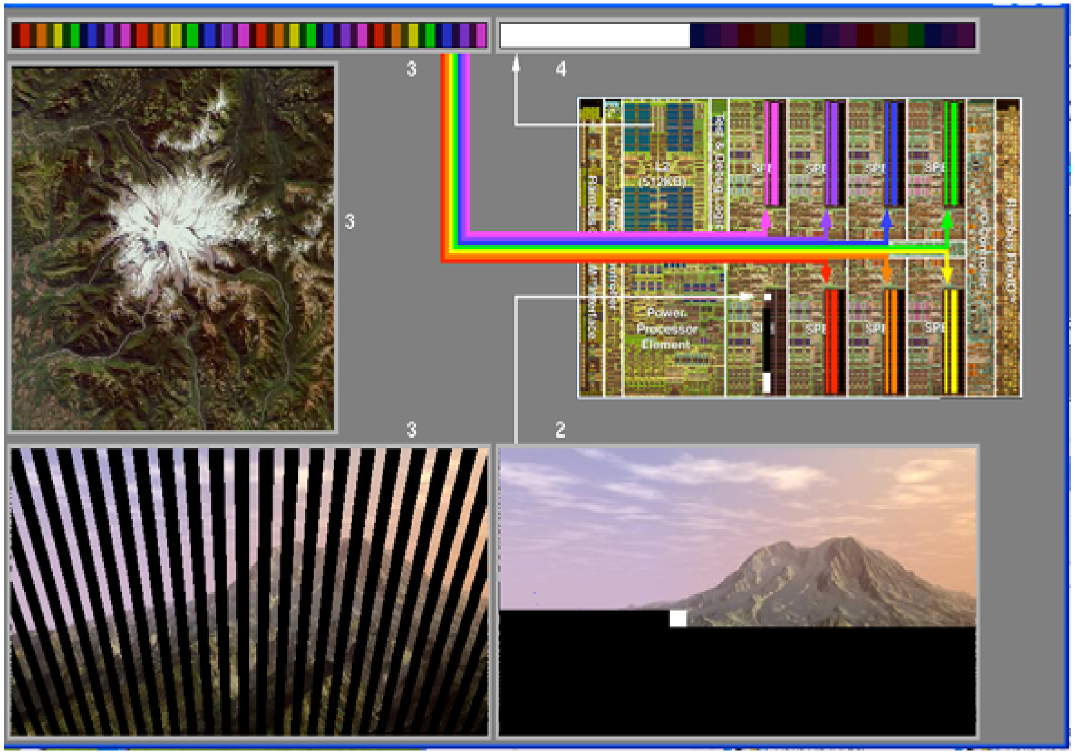
Accum DMA List (15 KB)

RC | RC | AC | Stack 4 KB

Code

(29 KB)

Chip


3
4
3
L2
(512KB)
SPE
SPE
SPE
SPE
Power
Processor
Element
SPE
SPE
SPE
3
2

Sample SPE Simulator Output

```
Performance Cycle count         96187613
Performance Instruction count   113483578 (108381008)
Performance CPI                 0.85 (0.89)

Branch instructions             1255537
Branch taken                    826730
Branch not taken                428807

Hint instructions               507711
Hint hit                        809520

Single cycle                                          59886926 ( 62.3%)
Dual cycle                                            24247041 ( 25.2%)
Nop cycle                                               133131 (  0.1%)
Stall due to branch miss                               1024965 (  1.1%)
Stall due to prefetch miss                               22394 (  0.0%)
Stall due to dependency                                9709208 ( 10.1%)
Stall due to fp resource conflict                            0 (  0.0%)
Stall due to waiting for hint target                   1163937 (  1.2%)
Stall due to dp pipeline                                     0 (  0.0%)
Channel stall cycle                                          0 (  0.0%)
SPU Initialization cycle                                     9 (  0.0%)
-----------------------------------------------------------------------
Total cycle                                           96187611 (100.0%)

The number of used registers are 128, the used ratio is 100.00
```

Austin

Mount Saint Helens

TRE 720P Performance

- 2.0 GHz Apple G5 0.6 frames/sec
  - 40% of cycles spent waiting for Memory
- 3.2 GHz Cell 30.0 frames/sec
  - 1% of cycles spent waiting for Memory
- Cell has 50x advantage

Summary

- Cell ushers in a new era of leading edge processors optimized for digital media and entertainment
- Desire for realism is driving a convergence between supercomputing and entertainment
- New levels of performance and power efficiency beyond what is achieved by PC processors
- Responsiveness to the human user and the network are key drivers for Cell
- Cell will enable entirely new classes of applications, even beyond those we contemplate today

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in applications such as implantation, life support, or other hazardous uses where malfunction could result in death, bodily injury, or catastrophic property damage. The information contained in this document does not affect or change IBM product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6351

The IBM home page is http://www.ibm.com
The IBM Microelectronics Division home page is
http://www.chips.ibm.com