6.189 IAP 2007

Lecture 2

Introduction to the Cell Processor

Michael Perrone (mpp@us.ibm.com)

Class Agenda

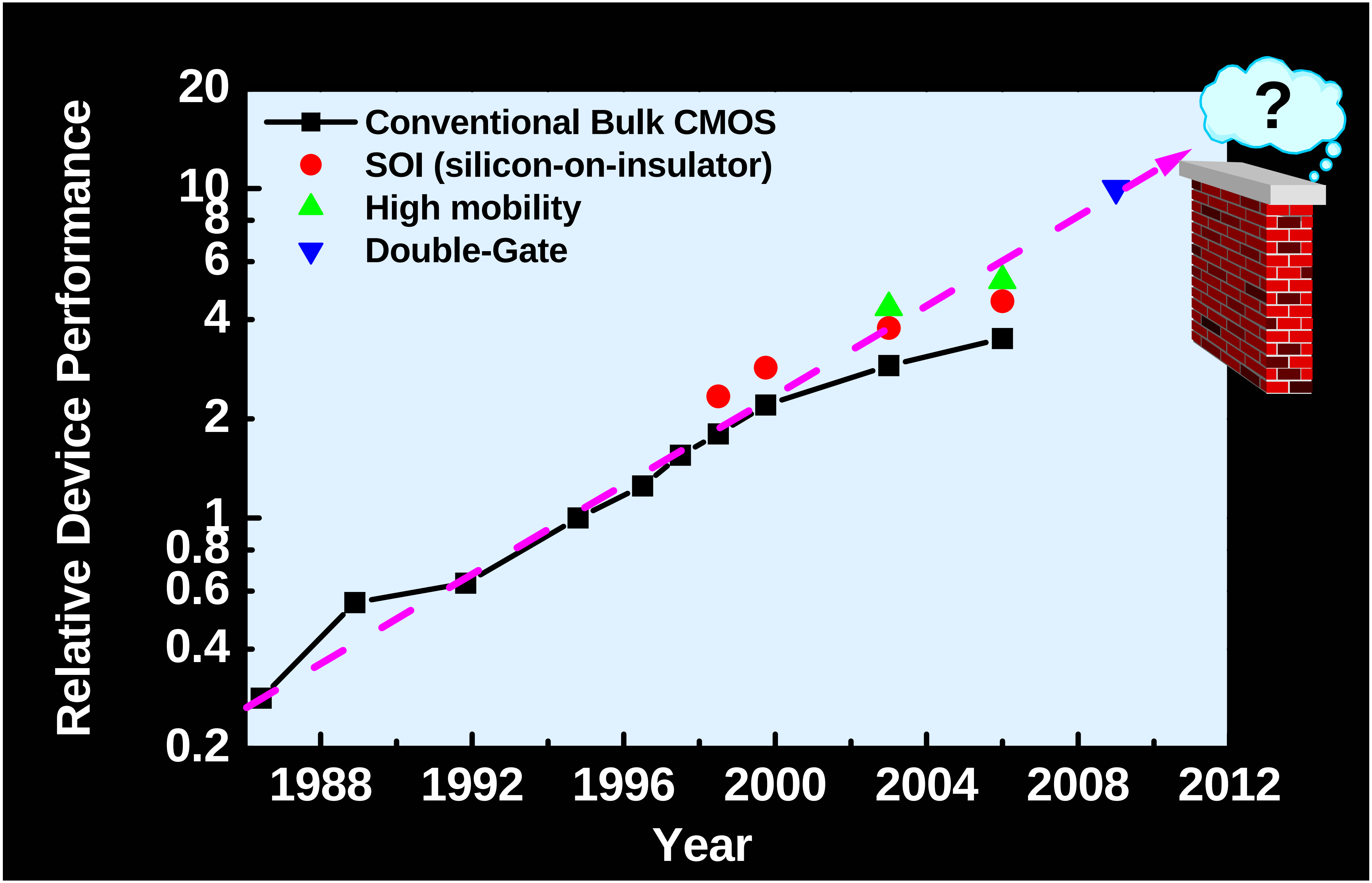
- Motivation for multicore chip design
- Cell basic design concept
- Cell hardware overview
  - Cell highlights
  - Cell processor
  - Cell processor components
- Cell performance characteristics
- Cell application affinity
- Cell software overview
  - Cell software environment
  - Development tools
  - Cell system simulator
  - Optimized libraries
- Cell software development considerations
- Cell blade

6.189 IAP 2007

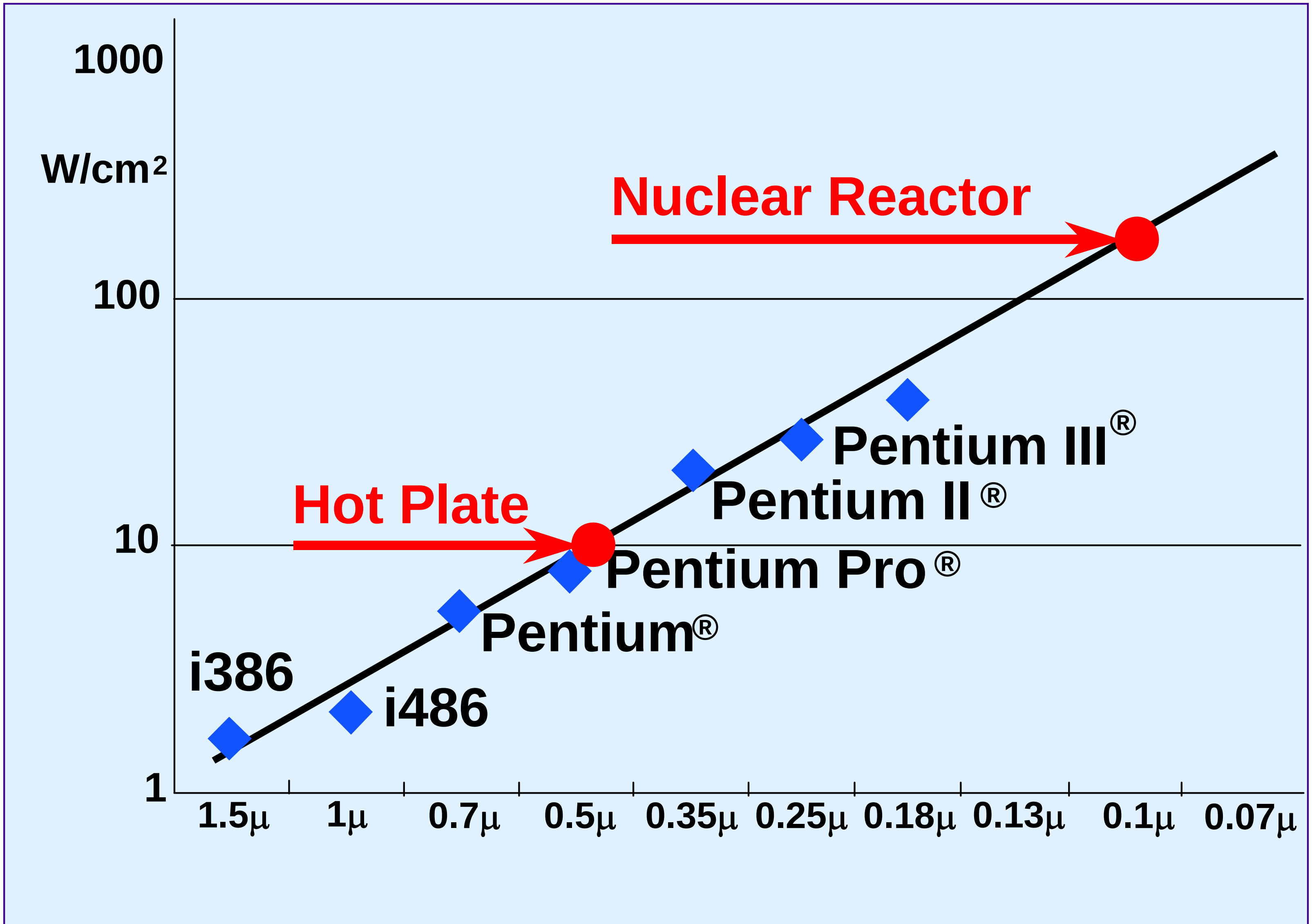
Lecture 2

Where have all the gigahertz gone?

Technology Scaling – We’ve hit the wall

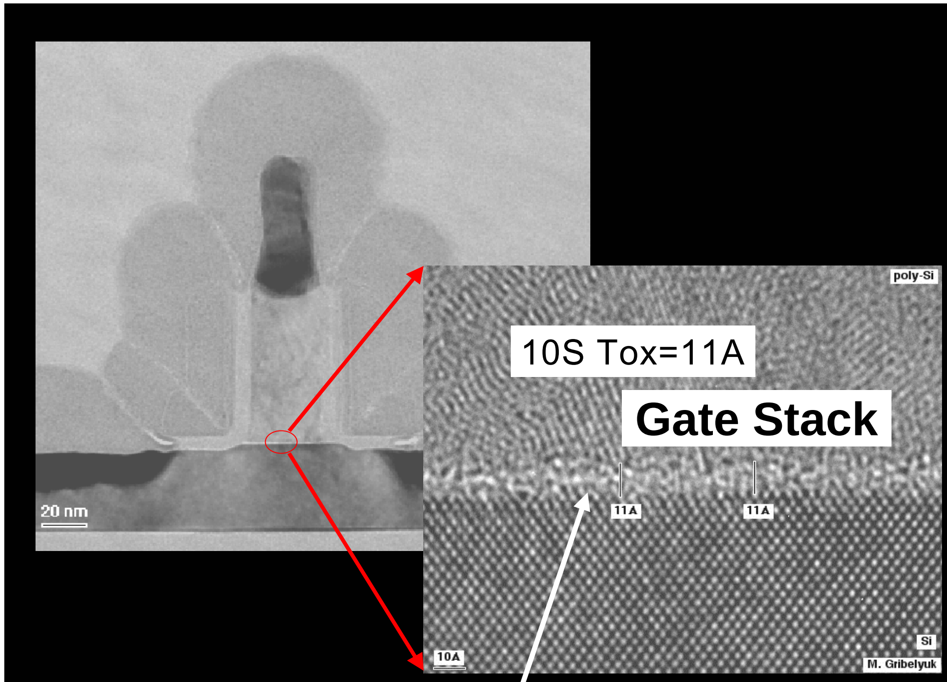


Power Density – The fundamental problem

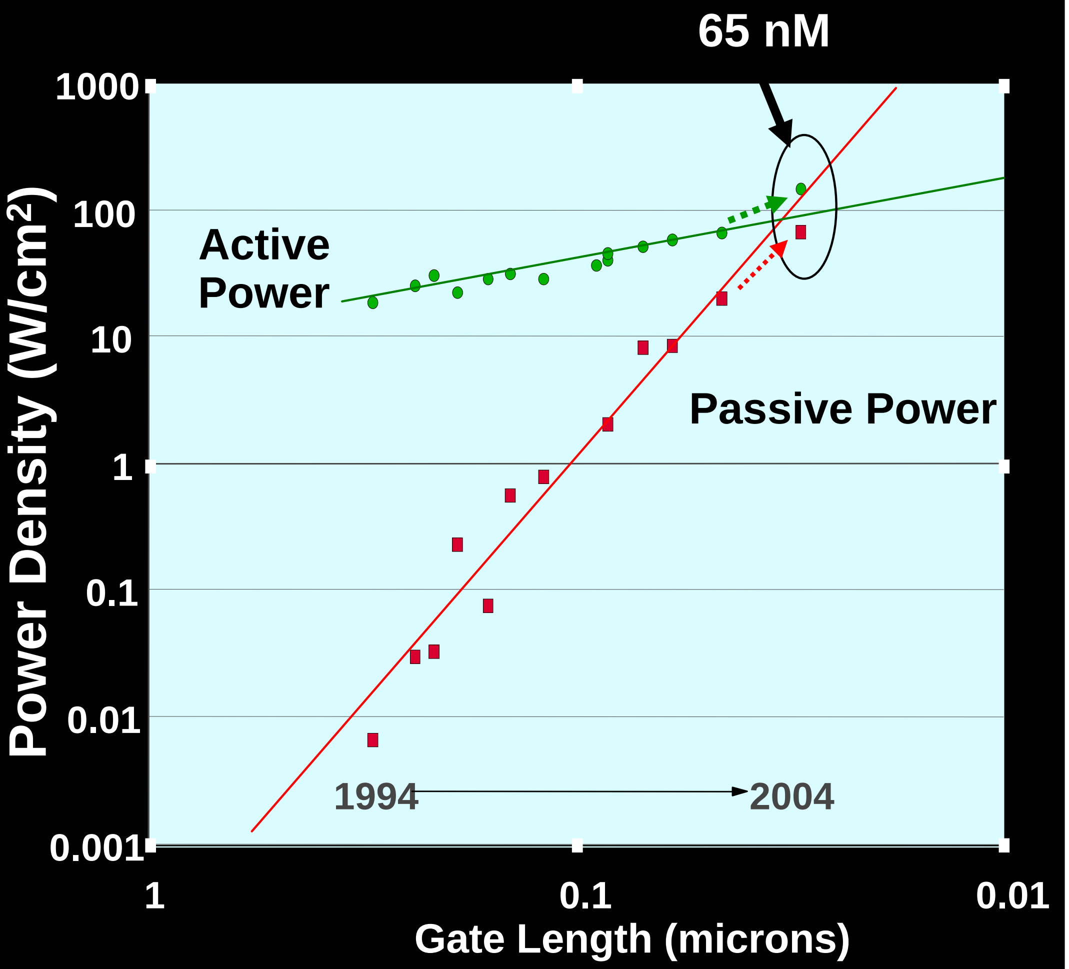


Source: Fred Pollack, Intel. New Microprocessor Challenges in the Coming Generations of CMOS Technologies, Micro32

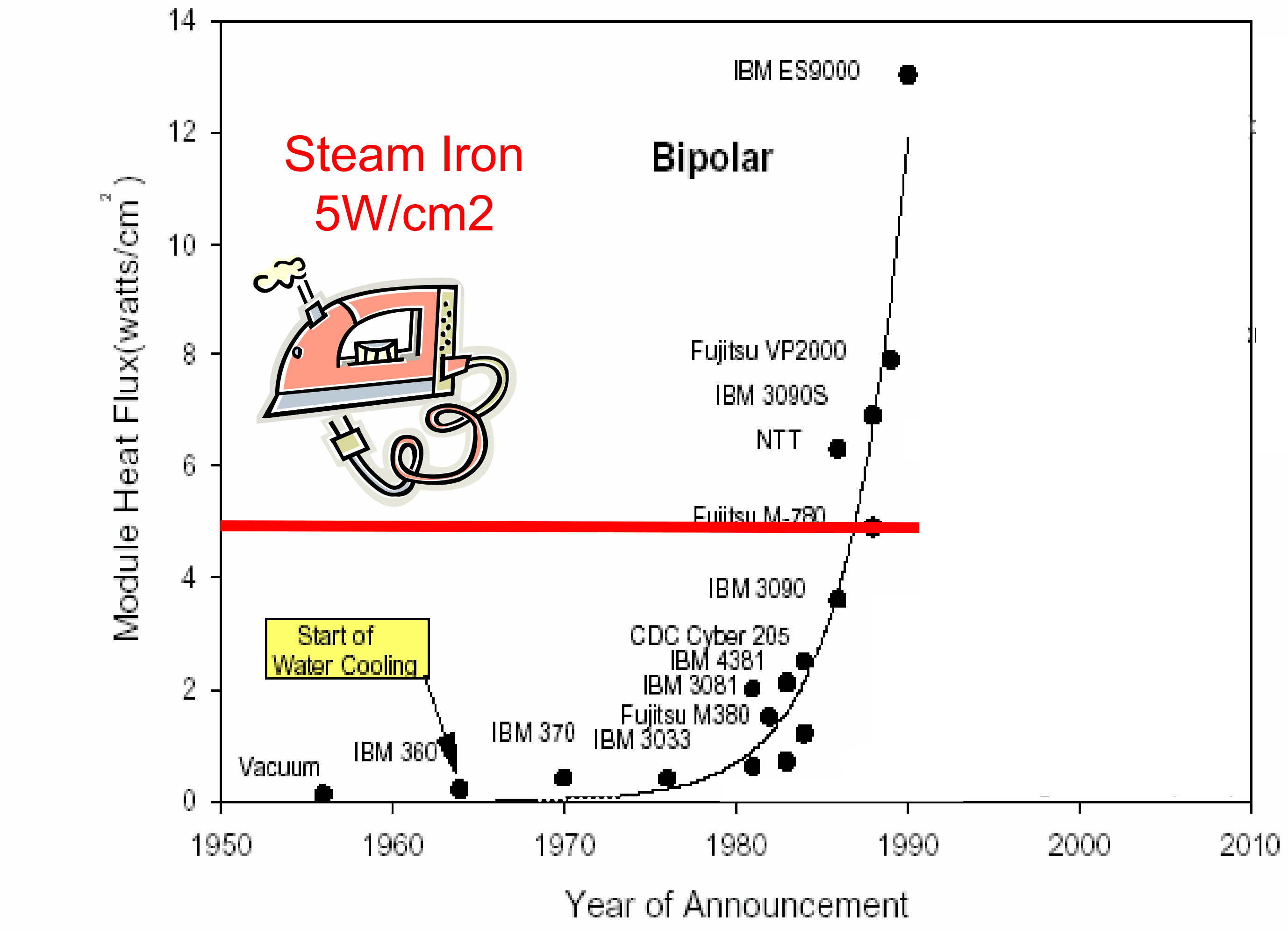
What’s Causing The Problem?



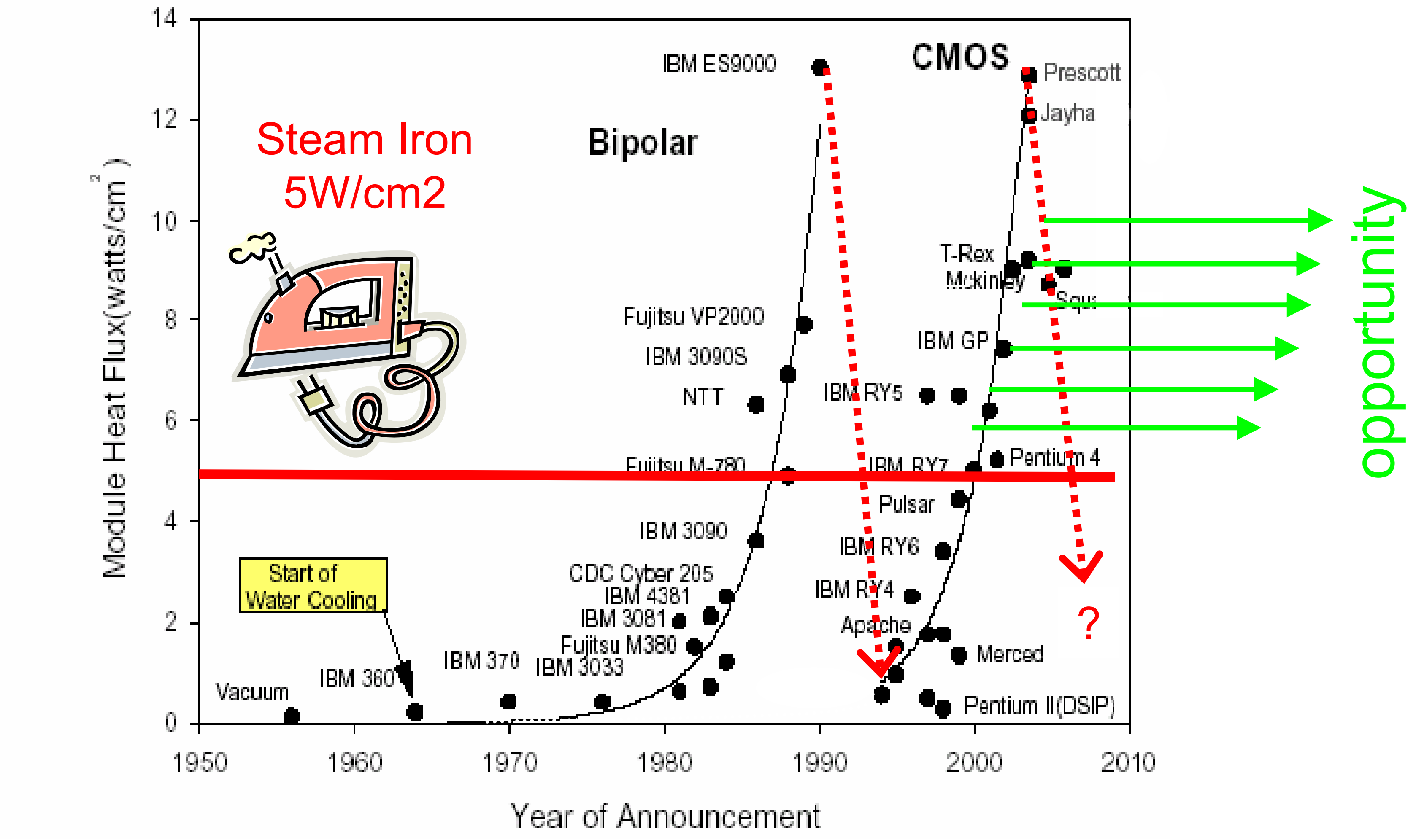
Gate dielectric approaching
a fundamental limit
(a few atomic layers)



Has This Ever Happened Before?



Has This Ever Happened Before?

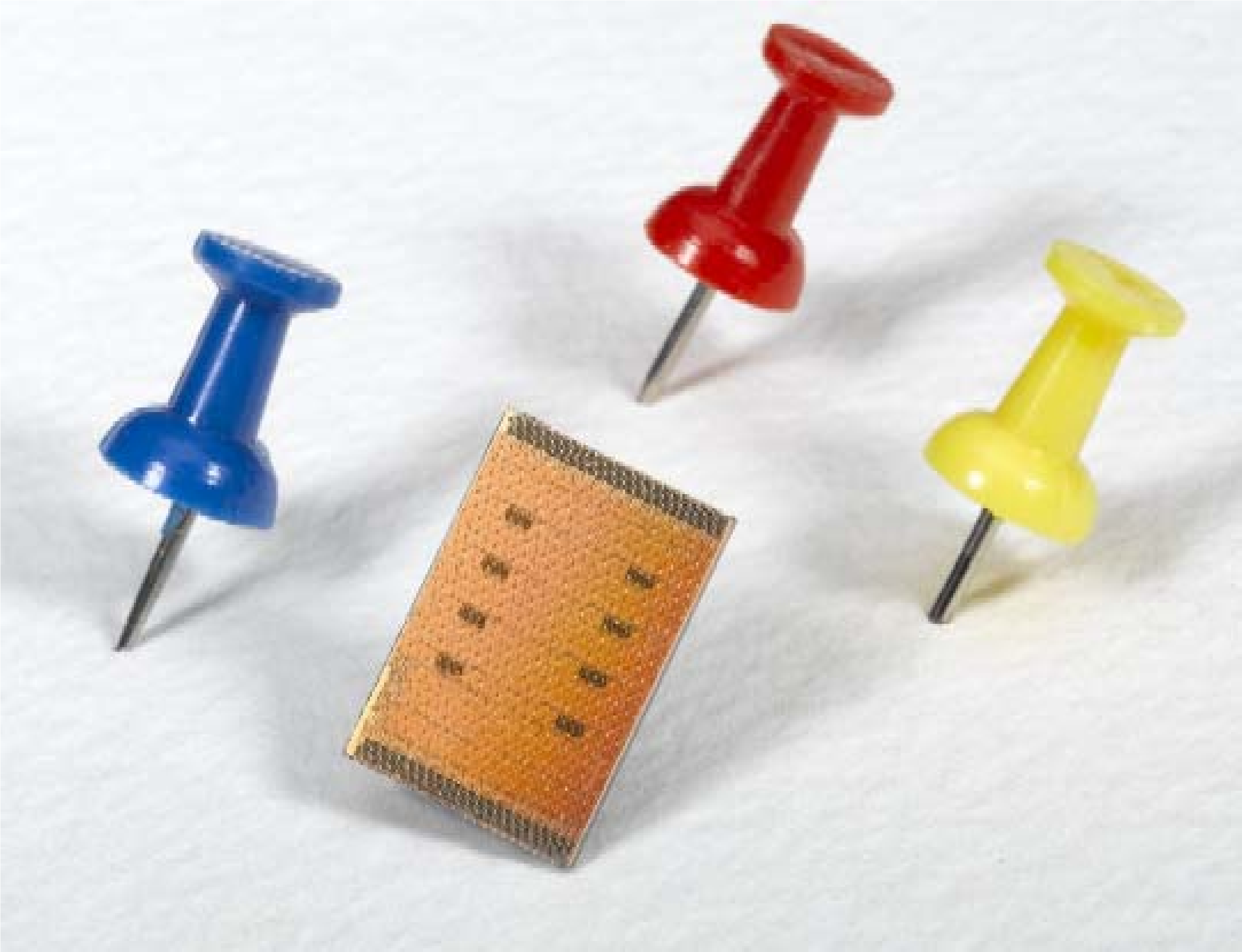


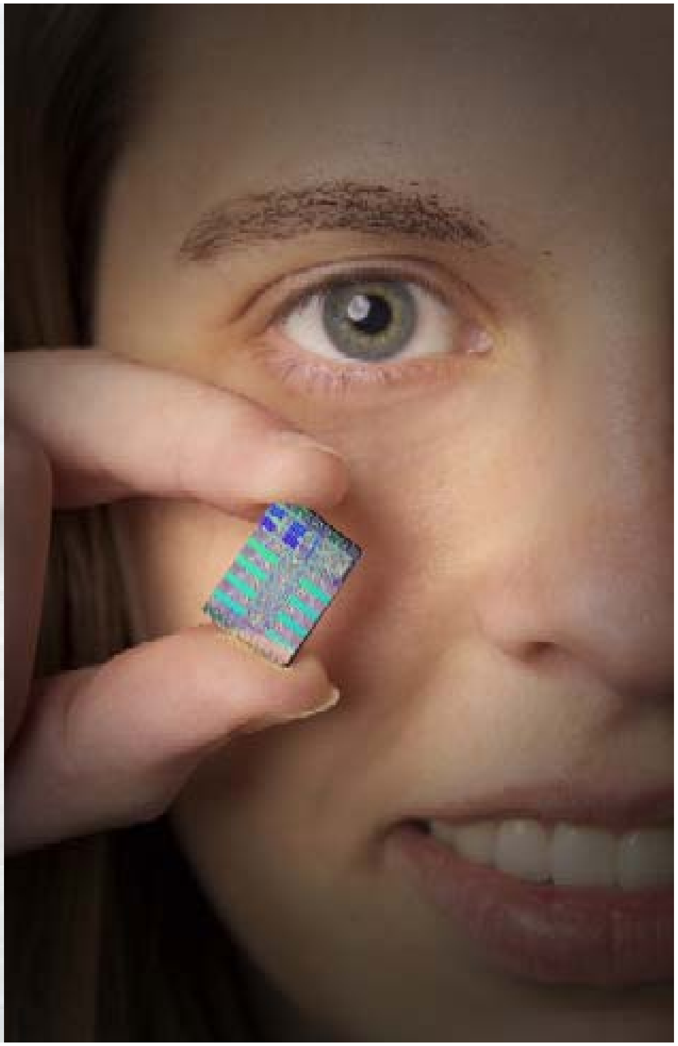
6.189 IAP 2007

Lecture 2

The Multicore Approach

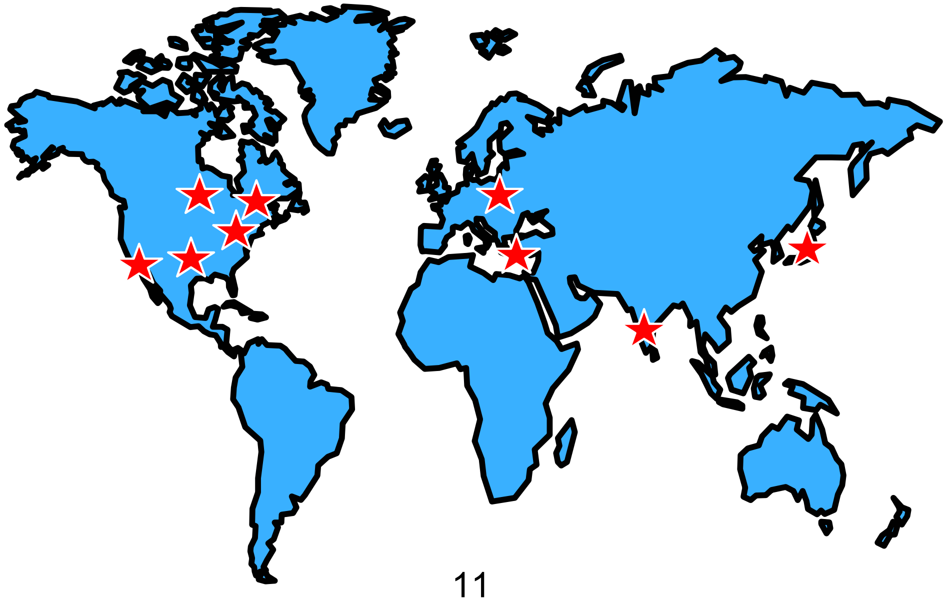
Cell





Cell History

- IBM, SCEI/Sony, Toshiba Alliance formed in 2000
- Design Center opened in March 2001
  - Based in Austin, Texas
- Single Cell BE operational Spring 2004
- 2-way SMP operational Summer 2004
- February 7, 2005: First technical disclosures
- October 6, 2005: Mercury Announces Cell Blade
- November 9, 2005: Open Source SDK & Simulator Published
- November 14, 2005: Mercury Announces Turismo Cell Offering
- February 8, 2006 IBM Announced Cell Blade



6.189 IAP 2007

Lecture 2

Cell Basic Design Concept

Cell Basic Concept

- Compatibility with 64b Power Architecture™
  - Builds on and leverages IBM investment and community
- Increased efficiency and performance
  - Attacks on the “Power Wall”
    - Non Homogenous Coherent Multiprocessor
    - High design frequency @ a low operating voltage with advanced power management
  - Attacks on the “Memory Wall”
    - Streaming DMA architecture
    - 3-level Memory Model: Main Storage, Local Storage, Register Files
  - Attacks on the “Frequency Wall”
    - Highly optimized implementation
    - Large shared register files and software controlled branching to allow deeper pipelines
- Interface between user and networked world
  - Image rich information, virtual reality
  - Flexibility and security
- Multi-OS support, including RTOS / non-RTOS
  - Combine real-time and non-real time worlds

Cell Design Goals

- Cell is an accelerator extension to Power
  - Built on a Power ecosystem
  - Used best know system practices for processor design
- 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
  - Virtual trusted computing environment for security

Cell Synergy

- Cell is not a collection of different processors, but a synergistic whole
  - Operation paradigms, data formats and semantics consistent
  - Share address translation and memory protection model

- PPE for operating systems and program control

- SPE optimized for efficient data processing
  - SPEs share Cell system functions provided by Power Architecture
  - MFC implements interface to memory
    - Copy in/copy out to local storage

- PowerPC provides system functions
  - Virtualization
  - Address translation and protection
  - External exception handling

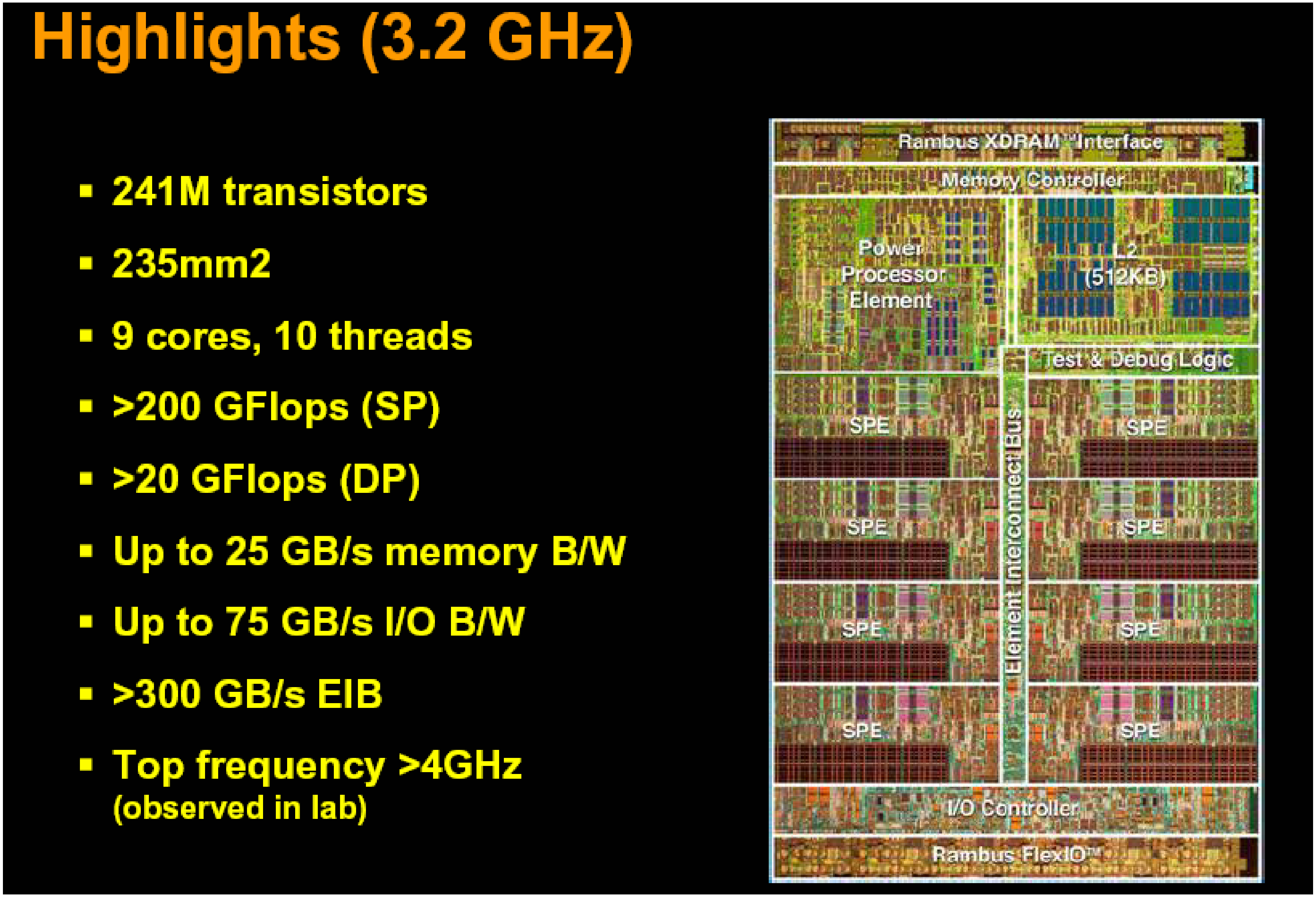
- EIB integrates system as data transport hub

6.189 IAP 2007

Lecture 2

Cell Hardware Components

Cell Chip

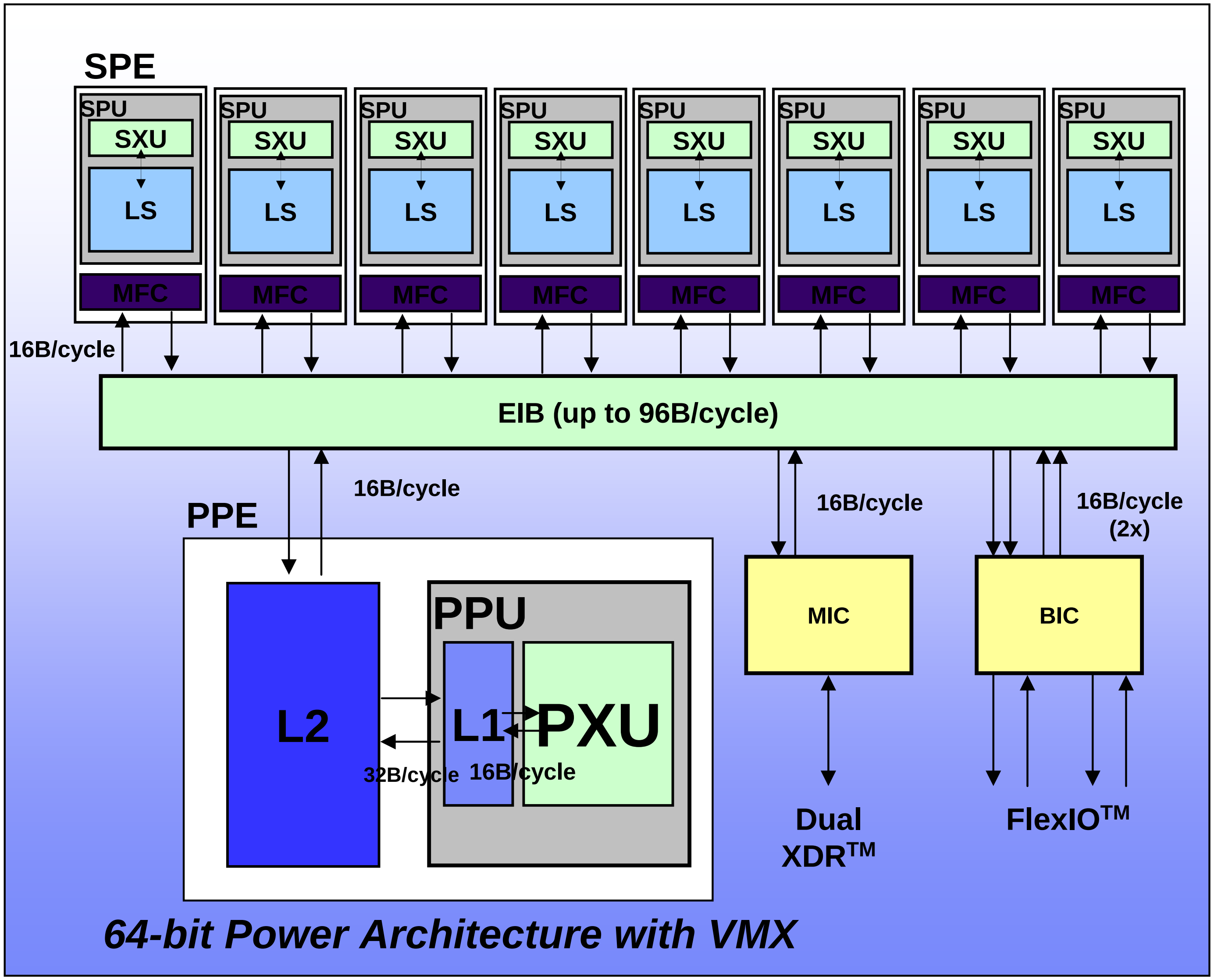


Highlights (3.2 GHz)

- 241M transistors
- 235mm2
- 9 cores, 10 threads
- >200 GFlops (SP)
- >20 GFlops (DP)
- Up to 25 GB/s memory B/W
- Up to 75 GB/s I/O B/W
- >300 GB/s EIB
- Top frequency >4GHz
  (observed in lab)

Cell Features

- Heterogeneous multicore system architecture
  - Power Processor Element for control tasks
  - Synergistic Processor Elements for data-intensive processing
- Synergistic Processor Element (SPE) consists of
  - Synergistic Processor Unit (SPU)
  - Synergistic Memory Flow Control (MFC)
    - Data movement and synchronization
    - Interface to high-performance Element Interconnect Bus

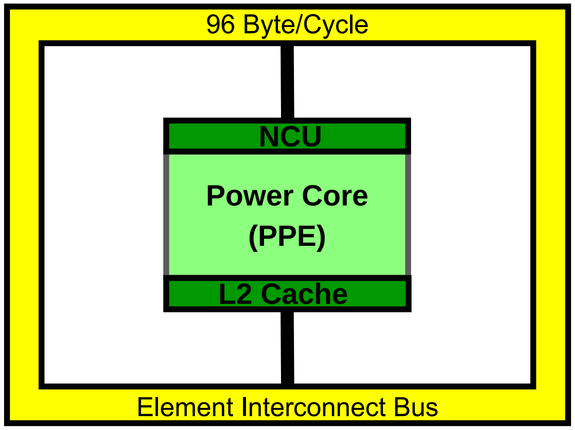


Cell Processor Components (1)

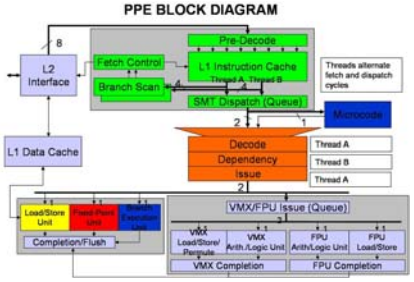
- Power Processor Element (PPE):
  - General purpose, 64-bit RISC processor (PowerPC AS 2.0.2)
  - 2-Way hardware multithreaded
  - L1 : 32KB I ; 32KB D
  - L2 : 512KB
  - Coherent load / store
  - VMX-32
  - Realtime Controls
    - Locking L2 Cache & TLB
    - Software / hardware managed TLB
    - Bandwidth / Resource Reservation
    - Mediated Interrupts

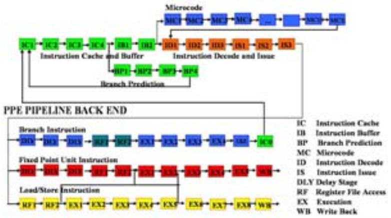
- Element Interconnect Bus (EIB):
  - Four 16 byte data rings supporting multiple simultaneous transfers per ring
  - 96Bytes/cycle peak bandwidth
  - Over 100 outstanding requests

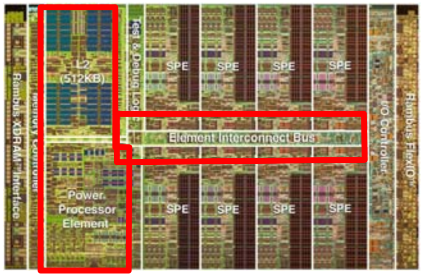
In the Beginning
– the solitary Power Processor



Custom Designed
– for high frequency, space, and power efficiency



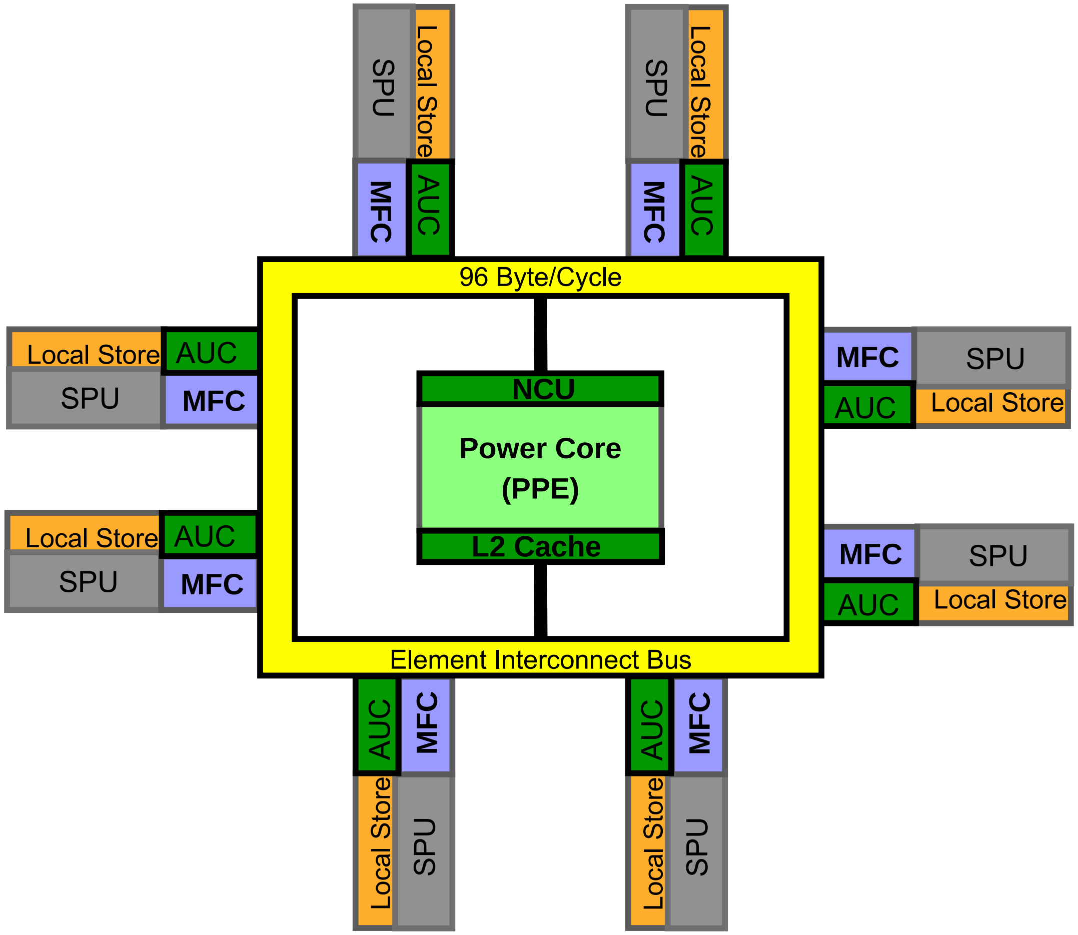


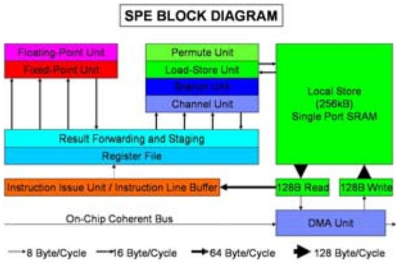


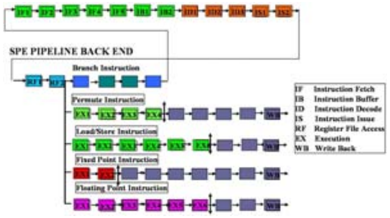
Cell Processor Components (2)

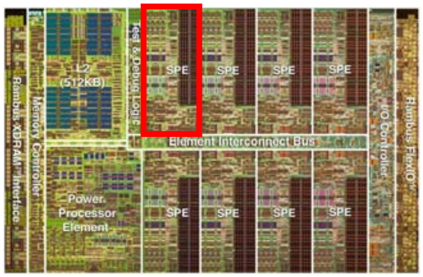
- Synergistic Processor Element (SPE):
  - Provides the computational performance
  - Simple RISC User Mode Architecture
    - Dual issue VMX-like
    - Graphics SP-Float
    - IEEE DP-Float
  - Dedicated resources: unified 128x128-bit RF, 256KB Local Store
  - Dedicated DMA engine: Up to 16 outstanding requests

- Memory Management & Mapping
  - SPE Local Store aliased into PPE system memory
  - MFC/MMU controls / protects SPE DMA accesses
    - Compatible with PowerPC Virtual Memory Architecture
    - SW controllable using PPE MMIO
  - DMA 1,2,4,8,16,128 -> 16Kbyte transfers for I/O access
  - Two queues for DMA commands: Proxy & SPU





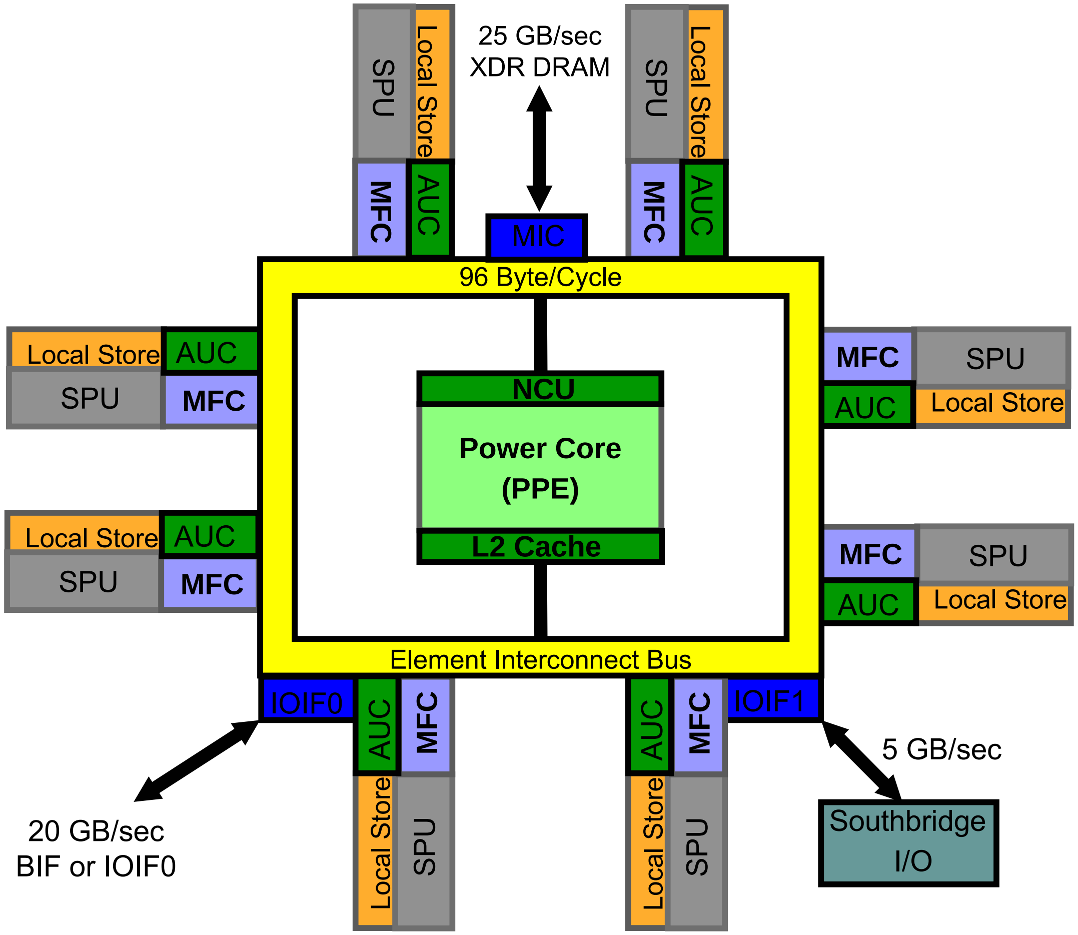


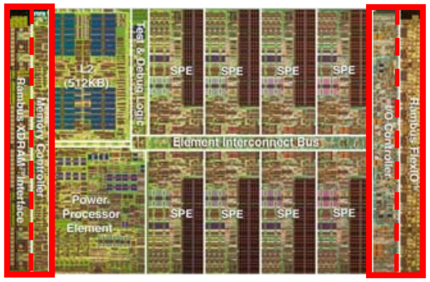


Cell Processor Components (3)

- Broadband Interface Controller (BIC):
  - Provides a wide connection to external devices
  - Two configurable interfaces (60GB/s @ 5Gbps)
    - Configurable number of bytes
    - Coherent (BIF) and / or I/O (IOIFx) protocols
  - Supports two virtual channels per interface
  - Supports multiple system configurations

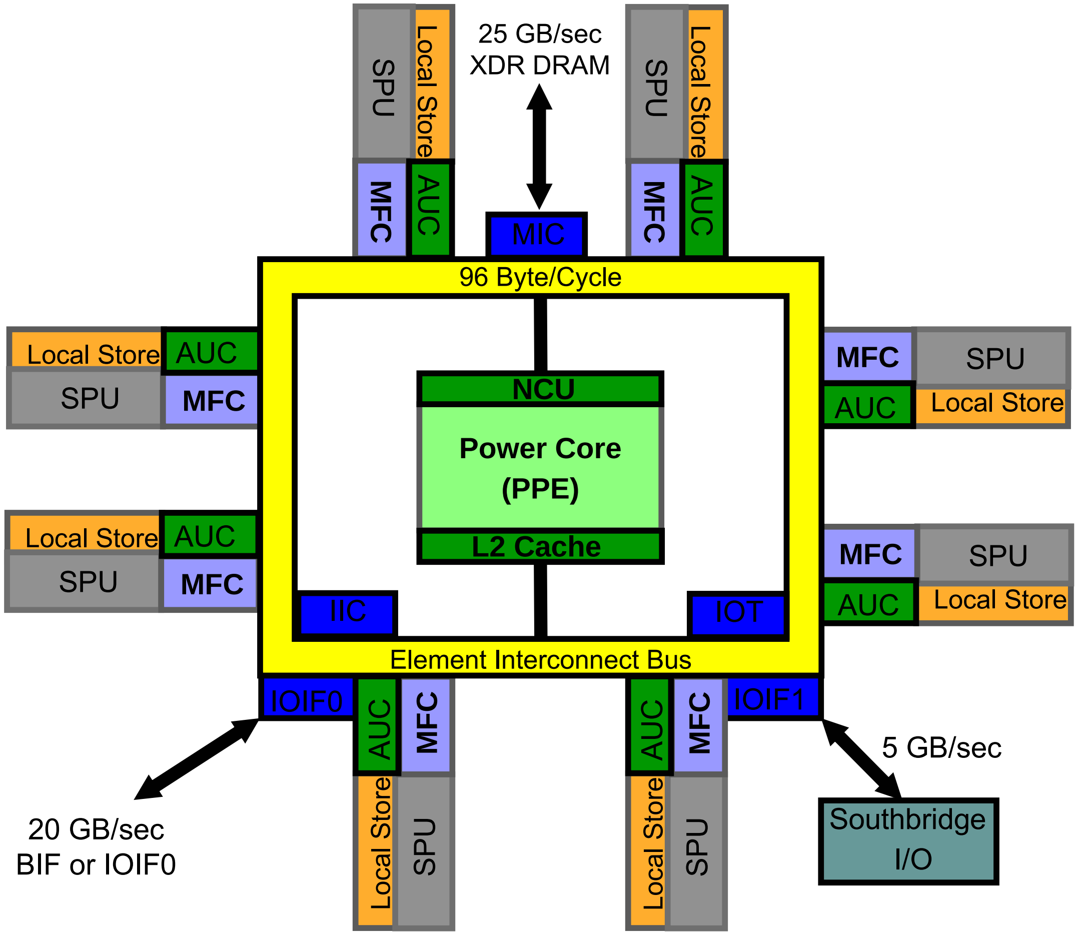
- Broadband Interface Controller (BIC):
  - Provides a wide connection to external devices
  - Two configurable interfaces (60GB/s @ 5Gbps)
    - Configurable number of bytes
    - Coherent (BIF) and / or I/O (IOIFx) protocols
  - Supports two virtual channels per interface
  - Supports multiple system configurations

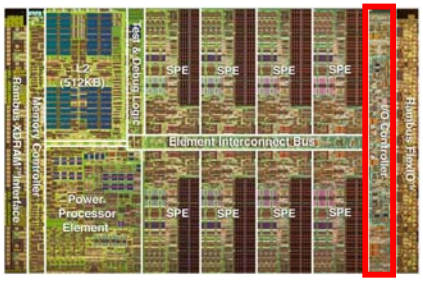




Cell Processor Components (4)

- 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





6.189 IAP 2007

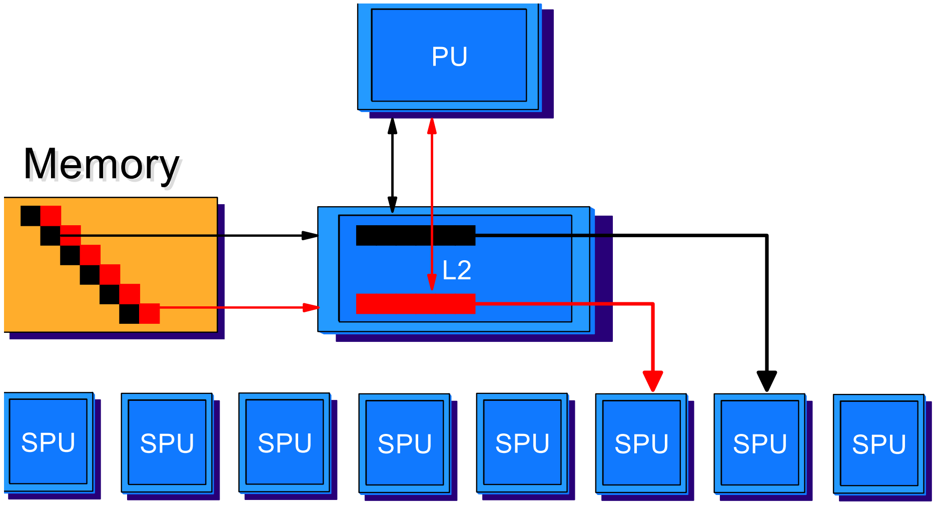
Lecture 2

Cell Performance Characteristics

Why Cell Processor Is So Fast?

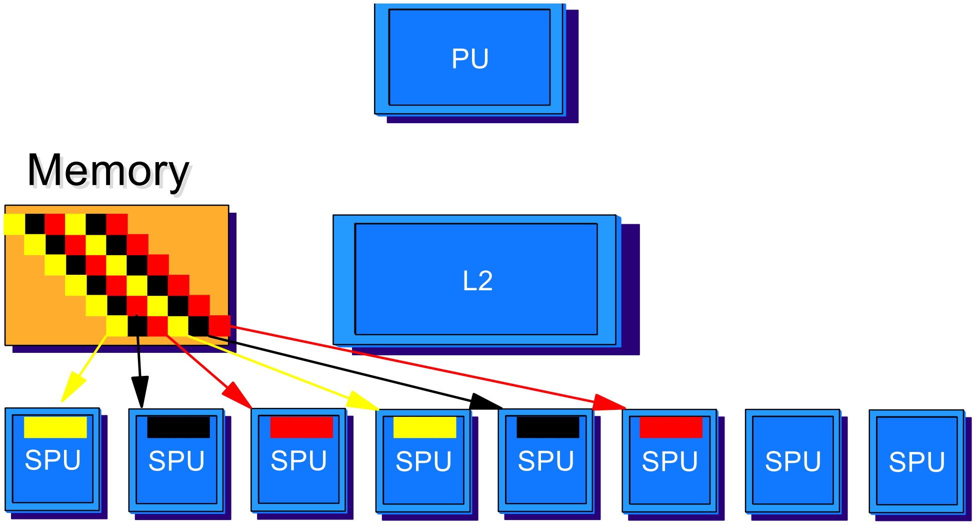
- Key Architectural Reasons
  - Parallel processing inside chip
  - Fully parallelized and concurrent operations
  - Functional offloading
  - High frequency design
  - High bandwidth for memory and IO accesses
  - Fine tuning for data transfer

Staging



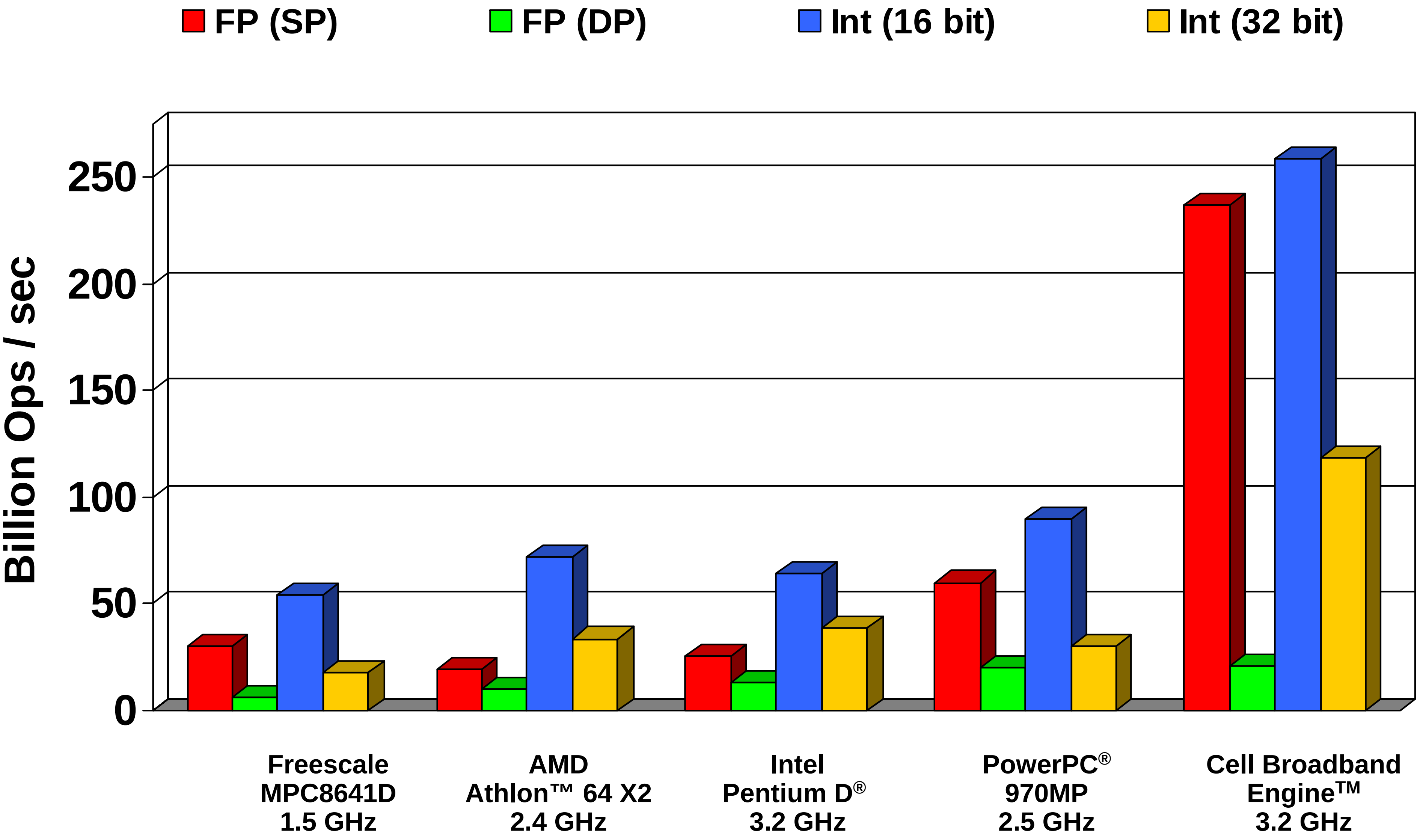
L2 - 4 outstanding loads + 2 prefetch

Data



SPU - 16 outstanding loads per SPU

Theoretical Peak Operations



Cell BE Performance

- BE can outperform a P4/SSE2 at same clock rate by 3 to 18x (assuming linear scaling) in various types of application workloads

Type	Algorithm	3 GHz GPP	3 GHz BE	BE Perf Advantage
HPC	Matrix Multiplication (S.P.)	25 Gflops	190 GFlops (8SPEs)	8x
	Linpack (S.P.)	18 GFlops (IA32)	150 GFlops (BE)	8x
	Linpack (D.P.)	6 GFlops (IA32)	12 GFLops (BE)	2x
bioinformatic	smith-waterman	570 Mcups (IA32)	420 Mcups (per SPE)	6x
graphics	transform-light	160 MVPS (G5/VMX)	240 MVPS (per SPE)	12x
	TRE	1.6 fps (G5/VMX)	24 fps (BE)	15x
security	AES	1.1 Gbps (IA32)	2Gbps (per SPE)	14x
	TDES	0.12 Gbps (IA32)	0.16 Gbps (per SPE)	10x
	MD-5	2.68 Gbps (IA32)	2.3 Gbps (per SPE)	6x
	SHA-1	0.85 Gbps (IA32)	1.98 Gbps (per SPE)	18x
communication	EEMBC	501 Telemark (1.4GHz mpc7447)	770 Telemark (per SPE)	12x
video processing	mpeg2 decoder (sdtv)	200 fps (IA32)	290 fps (per SPE)	12x

Key Performance Characteristics

- Cell's performance is about an order of magnitude better than GPP for media and other applications that can take advantage of its SIMD capability
  - Performance of its simple PPE is comparable to a traditional GPP performance
  - its each SPE is able to perform mostly the same as, or better than, a GPP with SIMD running at the same frequency
  - key performance advantage comes from its 8 de-coupled SPE SIMD engines with dedicated resources including large register files and DMA channels

- Cell can cover a wide range of application space with its capabilities in
  - Floating point operations
  - Integer operations
  - Data streaming / throughput support
  - Real-time support

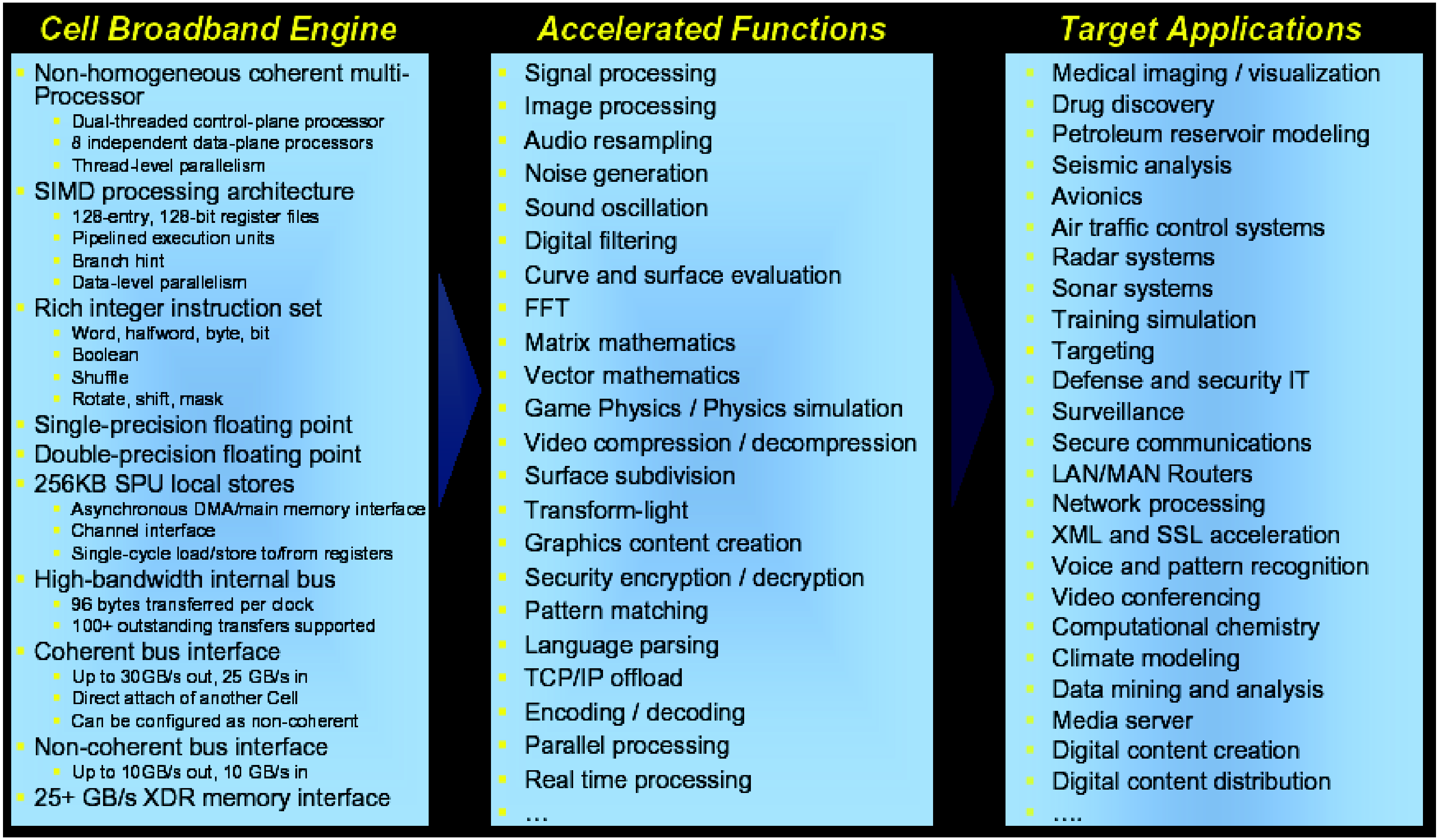
- Cell microarchitecture features are exposed to not only its compilers but also its applications
  - Performance gains from tuning compilers and applications can be significant
  - Tools/simulators are provided to assist in performance optimization efforts

6.189 IAP 2007

Lecture 2

Cell Application Affinity

Cell Application Affinity – Target Applications



Cell Broadband Engine

- Non-homogeneous coherent multi-Processor
  - Dual-threaded control-plane processor
  - 8 independent data-plane processors
  - Thread-level parallelism
- SIMD processing architecture
  - 128-entry, 128-bit register files
  - Pipelined execution units
  - Branch hint
  - Data-level parallelism
- Rich integer instruction set
  - Word, halfword, byte, bit
  - Boolean
  - Shuffle
  - Rotate, shift, mask
- Single-precision floating point
- Double-precision floating point
- 256KB SPU local stores
  - Asynchronous DMA/main memory interface
  - Channel interface
  - Single-cycle load/store to/from registers
- High-bandwidth internal bus
  - 96 bytes transferred per clock
  - 100+ outstanding transfers supported
- Coherent bus interface
  - Up to 30GB/s out, 25 GB/s in
  - Direct attach of another Cell
  - Can be configured as non-coherent
- Non-coherent bus interface
  - Up to 10GB/s out, 10 GB/s in
- 25+ GB/s XDR memory interface

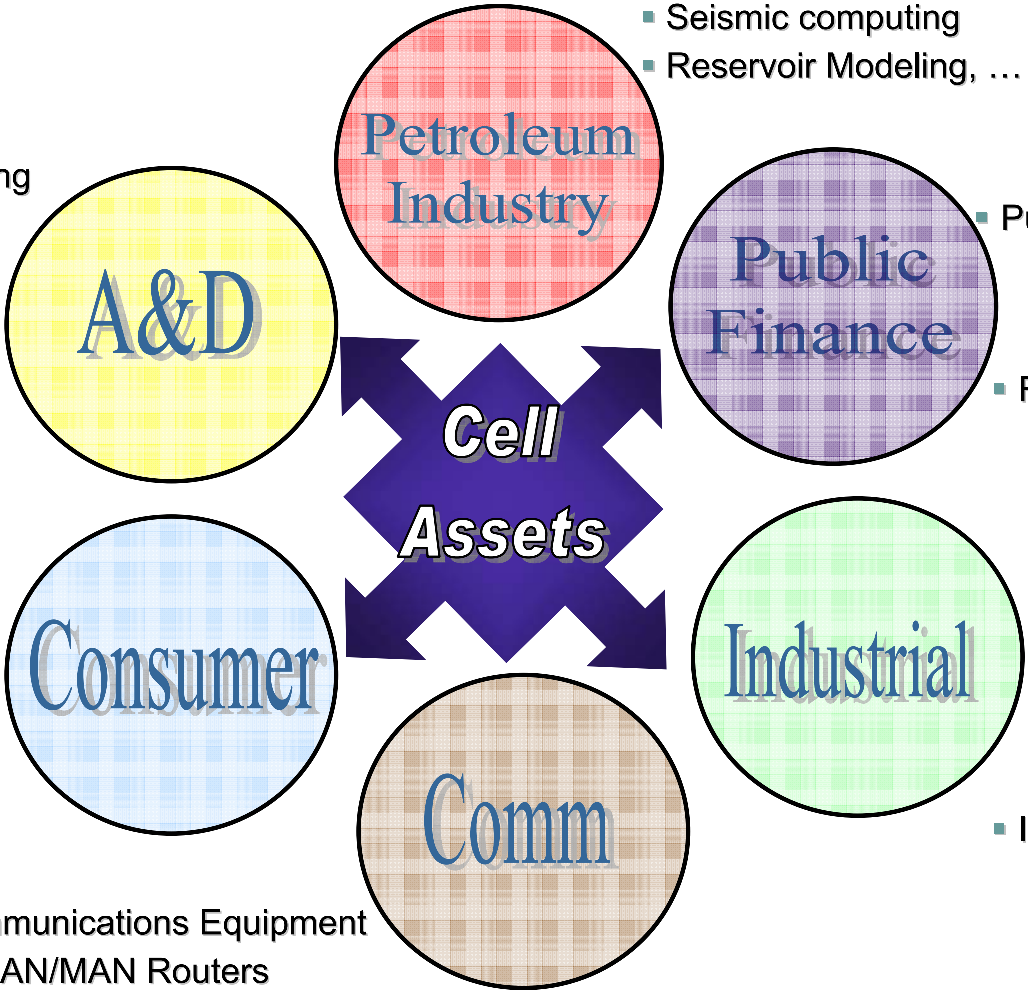
Accelerated Functions

- Signal processing
- Image processing
- Audio resampling
- Noise generation
- Sound oscillation
- Digital filtering
- Curve and surface evaluation
- FFT
- Matrix mathematics
- Vector mathematics
- Game Physics / Physics simulation
- Video compression / decompression
- Surface subdivision
- Transform-light
- Graphics content creation
- Security encryption / decryption
- Pattern matching
- Language parsing
- TCP/IP offload
- Encoding / decoding
- Parallel processing
- Real time processing
- …

Target Applications

- Medical imaging / visualization
- Drug discovery
- Petroleum reservoir modeling
- Seismic analysis
- Avionics
- Air traffic control systems
- Radar systems
- Sonar systems
- Training simulation
- Targeting
- Defense and security IT
- Surveillance
- Secure communications
- LAN/MAN Routers
- Network processing
- XML and SSL acceleration
- Voice and pattern recognition
- Video conferencing
- Computational chemistry
- Climate modeling
- Data mining and analysis
- Media server
- Digital content creation
- Digital content distribution
- ….

Cell Application Affinity – Target Industry Sectors



- Petroleum Industry
  - Seismic computing
  - Reservoir Modeling, …

- Aerospace & Defense
  - Signal & Image Processing
  - Security, Surveillance
  - Simulation & Training, …

- Public Sector / Gov't & Higher Educ.
  - Signal & Image Processing
  - Computational Chemistry, …

- Finance
  - Trade modeling

- Consumer / Digital Media
  - Digital Content Creation
  - Media Platform
  - Video Surveillance, …

- Medical Imaging
  - CT Scan
  - Ultrasound, …

- Industrial
  - Semiconductor / LCD
  - Video Conference

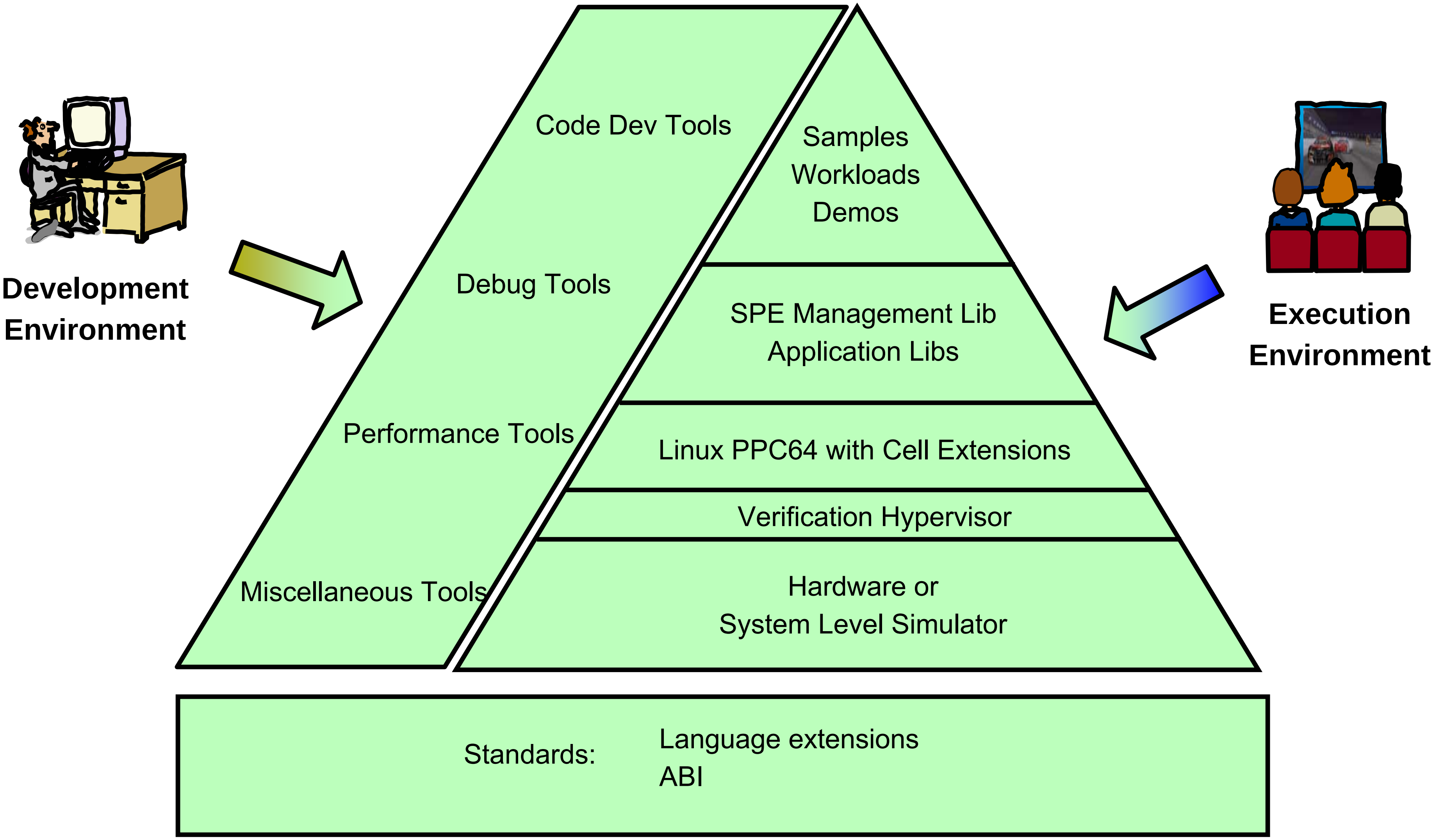
- Communications Equipment
  - LAN/MAN Routers
  - Access
  - Converged Networks
  - Security, …

6.189 IAP 2007

Lecture 2

Cell Software Environment

Cell Software Environment



Development Environment

Execution Environment

Code Dev Tools

Debug Tools

Performance Tools

Miscellaneous Tools

Samples
Workloads
Demos

SPE Management Lib
Application Libs

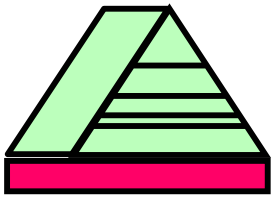
Linux PPC64 with Cell Extensions

Verification Hypervisor

Hardware or
System Level Simulator

Standards: Language extensions
ABI

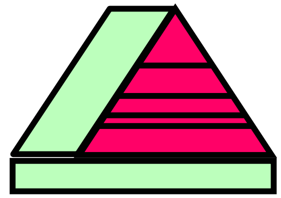
CBE Standards



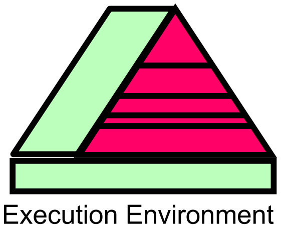
- Application Binary Interface Specifications
  - Defines such things as data types, register usage, calling conventions, and object formats to ensure compatibility of code generators and portability of code
    - SPE ABI
    - Linux for CBE Reference Implementation ABI
- SPE C/C++ Language Extensions
  - 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

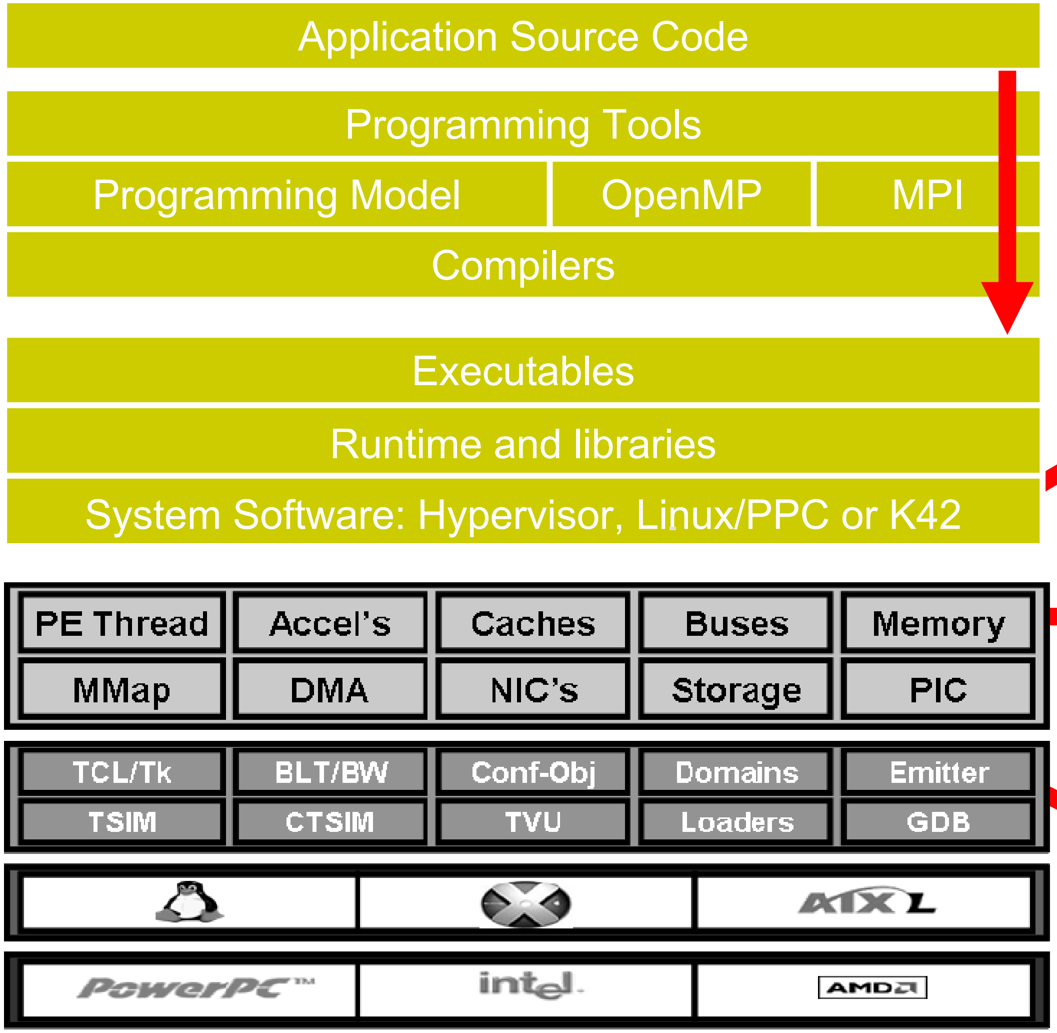
System Level Simulator

- Cell BE – full system simulator
  - Uni-Cell and multi-Cell simulation
  - User Interfaces – TCL and GUI
  - Cycle accurate SPU simulation (pipeline mode)
  - Emitter facility for tracing and viewing simulation events



SW Stack in Simulation


Execution Environment


Application Source Code

Programming Tools

Programming Model | OpenMP | MPI

Compilers

Executables

Runtime and libraries

System Software: Hypervisor, Linux/PPC or K42

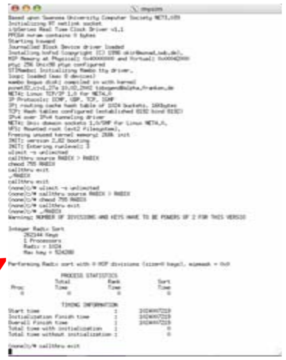
PE Thread | Accel's | Caches | Buses | Memory

MMap | DMA | NIC's | Storage | PIC

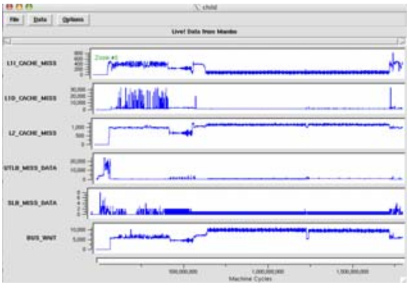
TCL/Tk | BLT/BW | Conf-Obj | Domains | Emitter

TSIM | CTSIM | TVU | Loaders | GDB

CellSim: *Simulation of hardware*

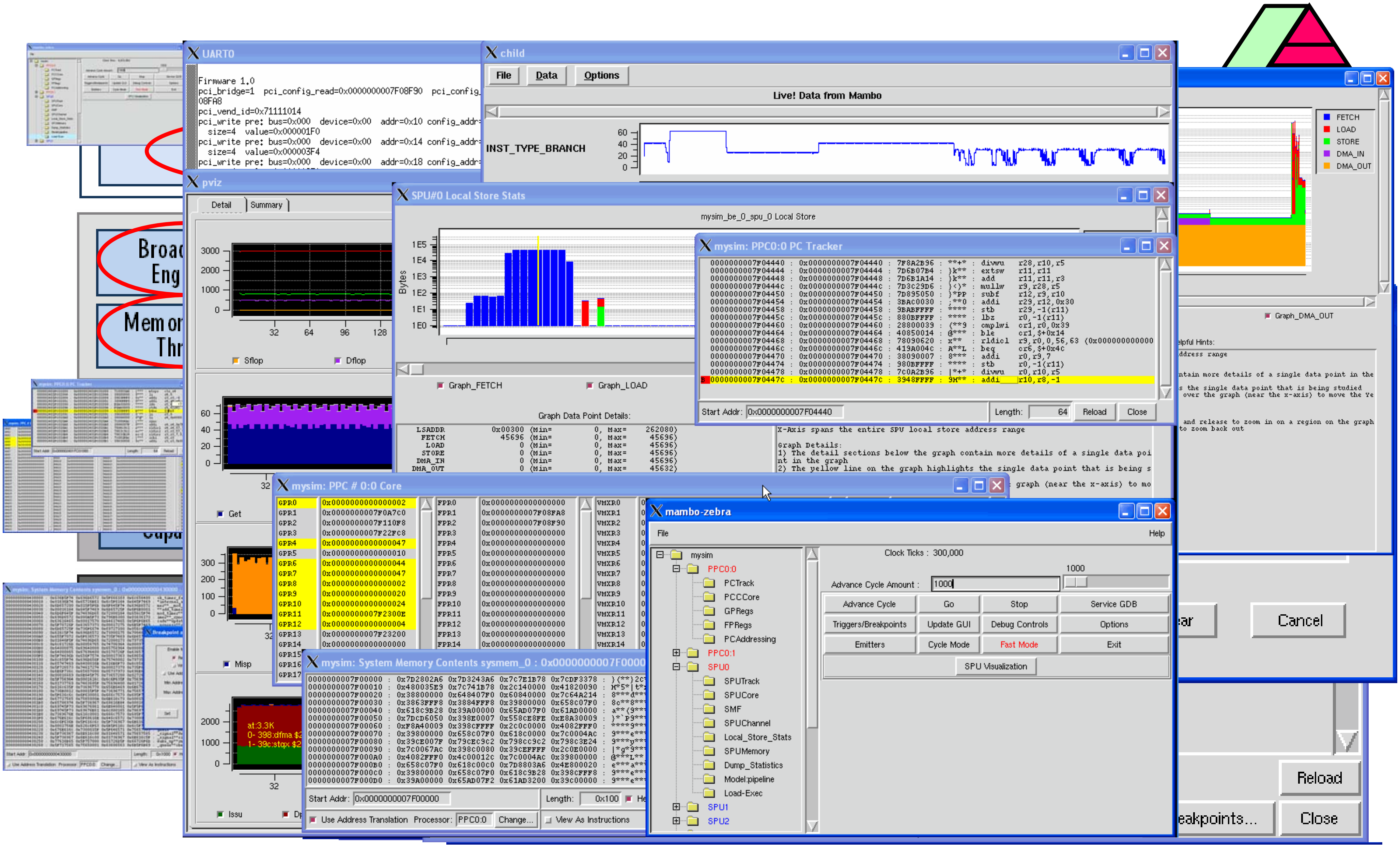


```
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
i/pSeries Real Time Clock Driver v1.1
PPC64 nvram contains 0 bytes
Starting kswapd
Journalled Block Device driver loaded
Installing knfsd (copyright (C) 1996 okir@monad.swb.de).
RTAS Memory at Physical: 0x40000000 and Virtual: 0x00042000
pty: 256 Unix98 ptys configured
STIMambo: Initializing Mambo tty driver.
loop: loaded (max 8 devices)
mambo bogus disk: compiled in with kernel
pcnet32.c:v1.27a 10.02.2002 tsbogend@alpha.franken.de
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
IP: routing cache hash table of 1024 buckets, 16Kbytes
TCP: Hash tables configured (established 8192 bind 8192)
IPv4 over IPv4 tunneling driver
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
VFS: Mounted root (ext2 filesystem).
Freeing unused kernel memory: 288k init
INIT: version 2.82 booting
INIT: Entering runlevel: 3
ulimit -s unlimited
callthru source RADIX > RADIX
chmod 755 RADIX
callthru exit
./RADIX
callthru exit
(none):/# ulimit -s unlimited
(none):/# callthru source RADIX > RADIX
(none):/# chmod 755 RADIX
(none):/# callthru exit
(none):/# ./RADIX
Warning: NUMBER OF DIVISIONS AND KEYS HAVE TO BE POWERS OF 2 FOR THIS VERSIO

Integer Radix Sort
     262144 Keys
     1 Processors
     Radix = 1024
     Max key = 524288

Performing Radix sort with 0 MSP divisions (size=0 keys), mipmask = 0x0

              PROCESS STATISTICS
             Total          Rank          Sort
 Proc         Time          Time          Time
   0            0             0             0

              TIMING INFORMATION
Start time                         :      1024697219
Initialization finish time         :      1024697219
Overall finish time                :      1024697219
Total time with initialization     :               0
Total time without initialization  :               0

(none):/# callthru exit
```



Traces

Cell Simulator Debugging Environment



UART0

```
Firmware 1.0
pci_bridge=1  pci_config_read=0x000000007F08F90  pci_config_
08FA8
pci_vend_id=0x71111014
pci_write pre: bus=0x000  device=0x00  addr=0x10 config_addr=
  size=4  value=0x000001F0
pci_write pre: bus=0x000  device=0x00  addr=0x14 config_addr=
  size=4  value=0x000003F4
pci_write pre: bus=0x000  device=0x00  addr=0x18 config_addr=
```

child

pviz

```
at:3.3K
0- 398:dfma $2
1- 39c:stqx $2
```

SPU#0 Local Store Stats

Graph Data Point Details:

```
  LSADDR        0x00300 (Min=        0,  Max=    262080)
   FETCH          45696 (Min=        0,  Max=     45696)
    LOAD              0 (Min=        0,  Max=     45696)
   STORE              0 (Min=        0,  Max=     45696)
  DMA_IN              0 (Min=        0,  Max=     45696)
 DMA_OUT              0 (Min=        0,  Max=     45632)
```

```
X-Axis spans the entire SPU local store address range

Graph Details:
1) The detail sections below the graph contain more details of a single data poi
nt in the graph
2) The yellow line on the graph highlights the single data point that is being s
```

mysim: PPC0:0 PC Tracker

```
 000000007F04440 : 0x000000007F04440 : 7F8A2B96 : **+* : divwu   r28,r10,r5
 000000007F04444 : 0x000000007F04444 : 7D6B07B4 : }k** : extsw   r11,r11
 000000007F04448 : 0x000000007F04448 : 7D6B1A14 : }k** : add     r11,r11,r3
 000000007F0444c : 0x000000007F0444c : 7D3c29D6 : }<)* : mullw   r9,r28,r5
 000000007F04450 : 0x000000007F04450 : 7D895050 : }*PP : subf    r12,r9,r10
 000000007F04454 : 0x000000007F04454 : 3BAC0030 : ;**0 : addi    r29,r12,0x30
 000000007F04458 : 0x000000007F04458 : 9BABFFFF : **** : stb     r29,-1(r11)
 000000007F0445c : 0x000000007F0445c : 880BFFFF : **** : lbz     r0,-1(r11)
 000000007F04460 : 0x000000007F04460 : 28800039 : (**9 : cmplwi  cr1,r0,0x39
 000000007F04464 : 0x000000007F04464 : 40850014 : @*** : ble     cr1,$+0x14
 000000007F04468 : 0x000000007F04468 : 78090620 : x**  : rldicl  r9,r0,0,56,63 (0x000000000000
 000000007F0446c : 0x000000007F0446c : 419A004c : A**L : beq     cr6,$+0x4c
 000000007F04470 : 0x000000007F04470 : 38090007 : 8*** : addi    r0,r9,7
 000000007F04474 : 0x000000007F04474 : 980BFFFF : **** : stb     r0,-1(r11)
 000000007F04478 : 0x000000007F04478 : 7c0A2B96 : |*+* : divwu   r0,r10,r5
B000000007F0447c : 0x000000007F0447c : 3948FFFF : 9H** : addi    r10,r8,-1
```

Start Addr: 0x000000007F04440 Length: 64

mysim: PPC # 0:0 Core

```
GPR0   0x0000000000000002   FPR0   0x0000000000000000   VMXR0
GPR1   0x000000007F0A7c0    FPR1   0x000000007F08FA8    VMXR1
GPR2   0x000000007F110F8    FPR2   0x000000007F08F90    VMXR2
GPR3   0x000000007F22Fc8    FPR3   0x0000000000000000   VMXR3
GPR4   0x0000000000000047   FPR4   0x0000000000000000   VMXR4
GPR5   0x0000000000000010   FPR5   0x0000000000000000   VMXR5
GPR6   0x0000000000000044   FPR6   0x0000000000000000   VMXR6
GPR7   0x0000000000000047   FPR7   0x0000000000000000   VMXR7
GPR8   0x0000000000000002   FPR8   0x0000000000000000   VMXR8
GPR9   0x0000000000000020   FPR9   0x0000000000000000   VMXR9
GPR10  0x0000000000000024   FPR10  0x0000000000000000   VMXR10
GPR11  0x000000007F2300E    FPR11  0x0000000000000000   VMXR11
GPR12  0x0000000000000004   FPR12  0x0000000000000000   VMXR12
GPR13  0x000000007F23200    FPR13  0x0000000000000000   VMXR13
GPR14  0x0000000000000000   FPR14  0x0000000000000000   VMXR14
GPR15
GPR16
GPR17
```

mysim: System Memory Contents sysmem_0 : 0x000000007F0000

```
000000007F00000 : 0x7D2802A6 0x7D3243A6 0x7c7E1B78 0x7CDF3378 : }(**}2c
000000007F00010 : 0x480035E9 0x7c741B78 0x2c140000 0x41820090 : H*5*|t*
000000007F00020 : 0x38800000 0x648407F0 0x60840000 0x7c64A214 : 8***d**
000000007F00030 : 0x3863FFF8 0x3884FFF8 0x39800000 0x658c07F0 : 8c**8**
000000007F00040 : 0x618c9B28 0x39A00000 0x65AD07F0 0x61AD0000 : a**(9**
000000007F00050 : 0x7DCD6050 0x398E0007 0x558CE8FE 0xE8A30009 : }*`P9**
000000007F00060 : 0xF8A40009 0x398cFFFF 0x2c0c0000 0x4082FFF0 : ****9**
000000007F00070 : 0x39800000 0x658c07F0 0x618c0000 0x7c0004Ac : 9***e**
000000007F00080 : 0x39cE007F 0x79cEc9c2 0x798cc9c2 0x798c3E24 : 9***y**
000000007F00090 : 0x7c0067Ac 0x398c0080 0x39cEFFFF 0x2c0E0000 : |*g*9**
000000007F000A0 : 0x4082FFF0 0x4c00012c 0x7c0004Ac 0x39800000 : @***L**
000000007F000B0 : 0x658c07F0 0x618c00c0 0x7D8803A6 0x4E800020 : e***a**
000000007F000c0 : 0x39800000 0x658c07F0 0x618c9B28 0x398cFFF8 : 9***e**
000000007F000D0 : 0x39A00000 0x65AD07F2 0x61AD3200 0x39c00000 : 9***e**
```

Start Addr: 0x000000007F00000 Length: 0x100

☑ Use Address Translation Processor: PPC0:0

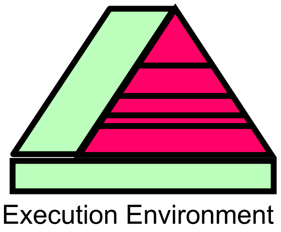
☐ View As Instructions

mambo-zebra

Clock Ticks : 300,000

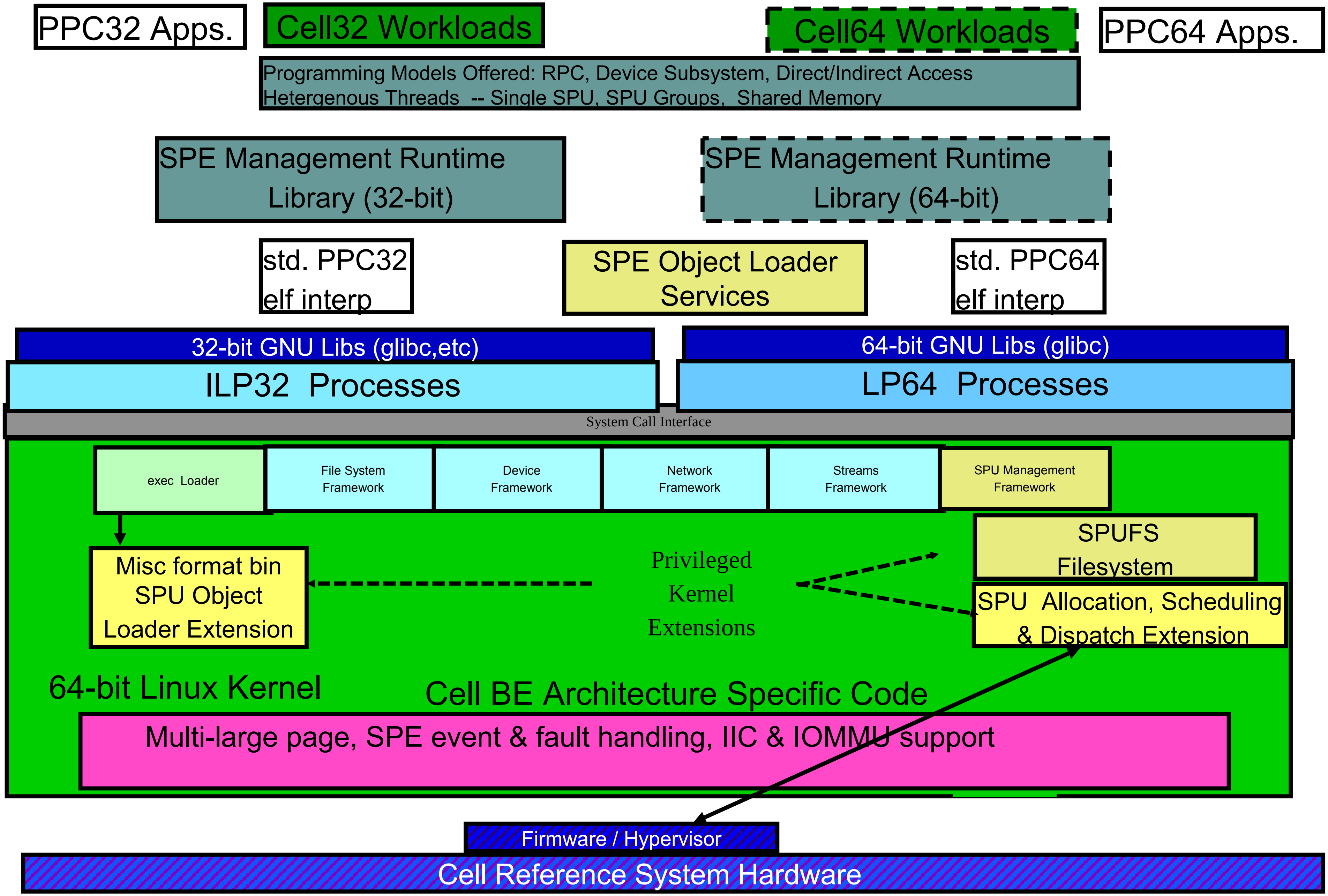
Advance Cycle Amount : 1000

Linux on CBE



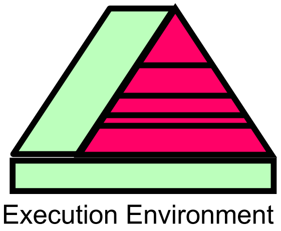
- Provided as patched to the 2.6.15 PPC64 Kernel
  - Added heterogeneous lwp/thread model
    - SPE thread API created (similar to pthreads library)
    - User mode direct and indirect SPE access models
    - Full pre-emptive SPE context management
    - spe_ptrace() added for gdb support
    - spe_schedule() 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
  - PPE proxy thread allocated for each SPE thread to:
    - Provide a single namespace for both PPE and SPE threads
    - Assist in SPE initiated C99 and POSIX-1 library services
  - SPE Error, Event and Signal handling directed to parent PPE thread
  - SPE elf objects wrapped into PPE shared objects with extended gld
  - All patches for Cell in architecture dependent layer (subtree of PPC64)

CBE Extensions to Linux



PPC32 Apps.
Cell32 Workloads
Cell64 Workloads
PPC64 Apps.
Programming Models Offered: RPC, Device Subsystem, Direct/Indirect Access
Hetergenous 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 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
Cell BE Architecture Specific Code
Multi-large page, SPE event & fault handling, IIC & IOMMU support
Firmware / Hypervisor
Cell Reference System Hardware

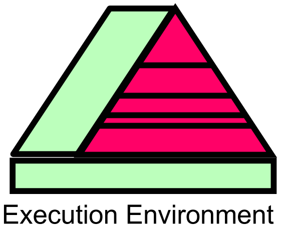
SPE Management Library



- 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 area 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

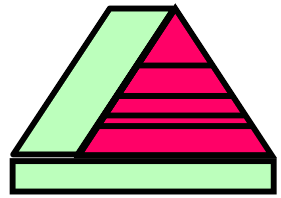
- SPE Executable
  - Standalone SPE program managed by a PPE executive
  - Executive responsible for loading and executing SPE program
    - It also services assisted requests for I/O (eg, fopen, fwrite, fprintf) and memory requests (eg, mmap, shmat, …)

Optimized SPE and Multimedia Extension Libraries



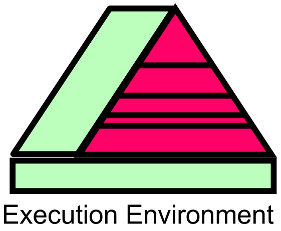
- Standard SPE C library subset
  - optimized SPE C99 functions including stdlib c lib, math and etc.
  - subset of POSIX.1 Functions – PPE assisted
- Audio resample - resampling audio signals
- FFT - 1D and 2D fft functions
- gmath - mathematic functions optimized for gaming environment
- image - convolution functions
- intrinsics - generic intrinsic conversion functions
- large-matrix - functions performing large matrix operations
- matrix - basic matrix operations
- mpm - multi-precision math functions
- noise - noise generation functions
- oscillator - basic sound generation functions
- sim – simulator only function including print, profile checkpoint, socket I/O, etc …
- surface - a set of bezier curve and surface functions
- sync - synchronization library
- vector - vector operation functions

Sample Source



- cesof - the samples for the CBE embedded SPU object format usage
- spu_clean - cleans SPU register and local store
- spu_entry - sample SPU entry function (crt0)
- spu_interrupt - SPU first level interrupt handler sample
- spulet - direct invocation of a spu program from Linux shell
- sync
- simpleDMA / DMA
- tutorial - example source code from the tutorial
- SDK test suite

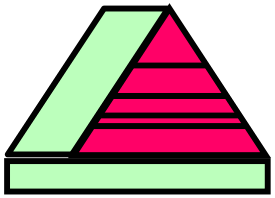
Workloads



- FFT16M – optimized 16 M point complex FFT
- Oscillator - audio signal generator
- Matrix Multiply – matrix multiplication workload
- VSE_subdiv - variable sharpness subdivision algorithm

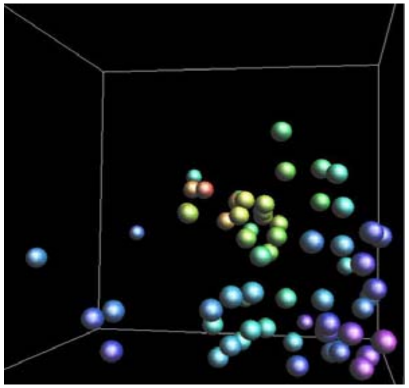
Bringup Workloads / Demos

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

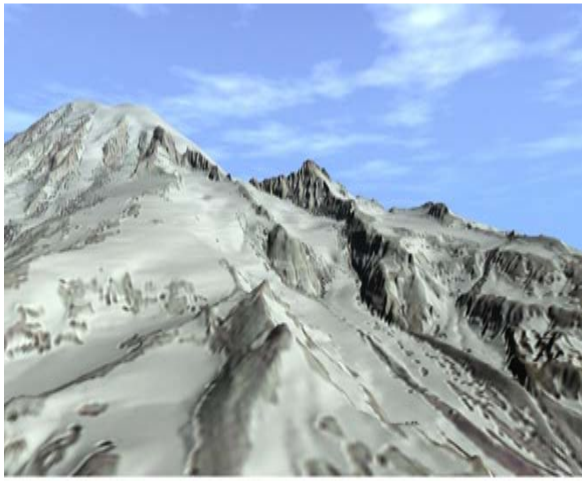


Execution Environment

Geometry Engine

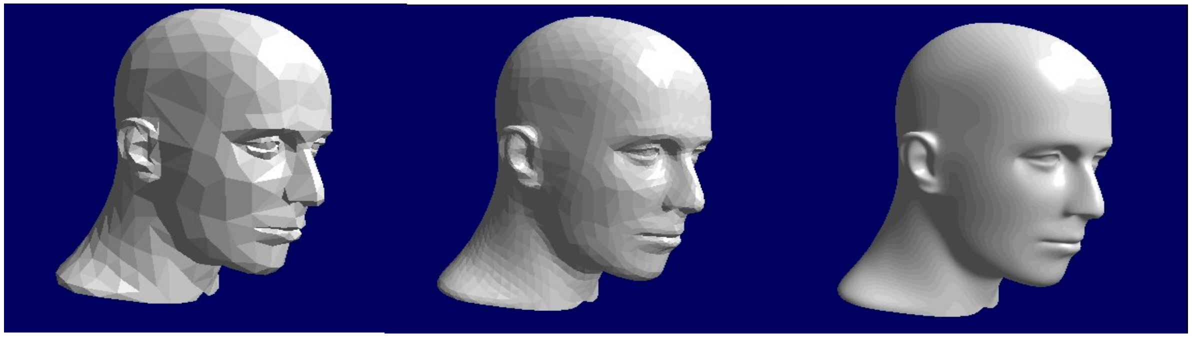


Physics Simulation



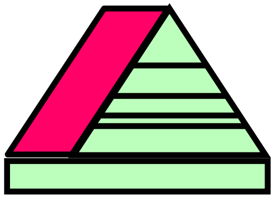
Terrain Rendering Engine

Subdivision Surfaces



Code Development Tools

- GNU based binutils
  - From Sony Computer Entertainment
  - gas SPE assembler
  - gld SPE ELF object linker
    - ppu-embedspu script for embedding SPE object modules in PPE executables
  - Miscellaneous bin utils (ar, nm, ...) targeting SPE modules

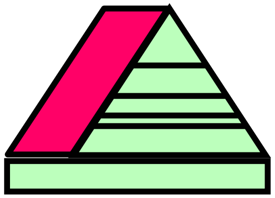


Development Environment

- GNU based C/C++ compiler targeting SPE
  - From Sony Computer Entertainment
  - Retargeted compiler to SPE
  - Supports common SPE Language Extensions and ABI (ELF/Dwarf2)

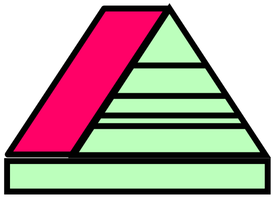
- Cell Broadband Engine Optimizing Compiler (executable)
  - IBM XLC C/C++ for PowerPC (Tobey)
  - IBM XLC C retargeted to SPE assembler (including vector intrinsics)
    - Highly optimizing
  - Prototype CBE Programmer Productivity Aids
    - Auto-Vectorization (auto-SIMD) for SPE and PPE Multimedia Extension code
  - Timing Analysis Tool

Bringup Debug Tools



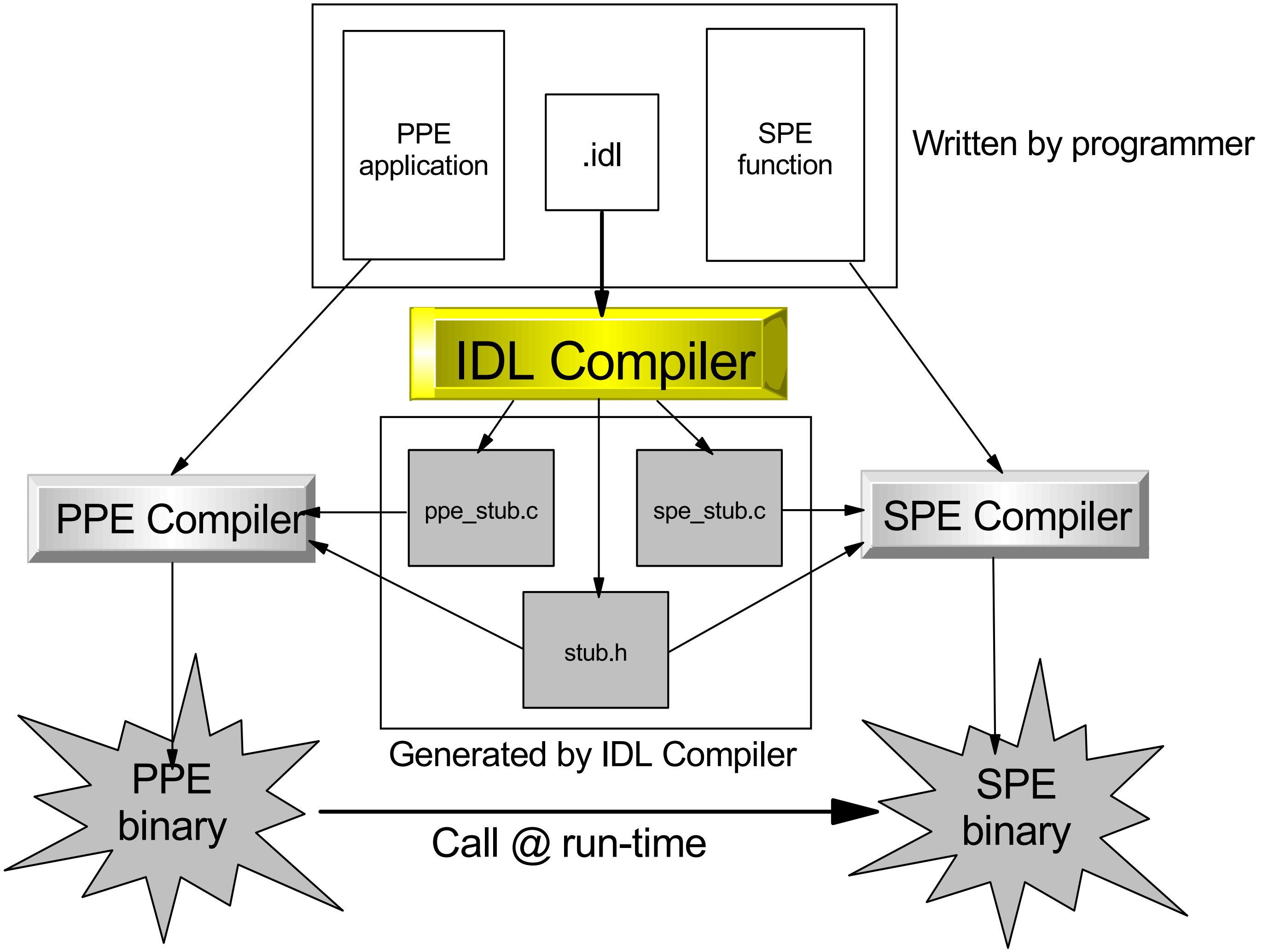
- GNU gdb
  - Multicore Application source level debugger supporting
    - PPE multithreading
    - SPE multithreading
    - Interacting PPE and SPE threads
  - Three modes of debugging SPU threads
    - Standalone SPE debugging
    - Attach to SPE thread
      - Thread ID output when SPU_DEBUG_START=1

SPE Performance Tools (executables)



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

Miscellaneous Tools – IDL Compiler



6.189 IAP 2007

Lecture 2

Cell Software Development Considerations

CELL Software Design Considerations

- Four Levels of Parallelism
  - Blade Level: Two Cell processors per blade
  - Chip Level: 9 cores run independent tasks
  - Instruction level: Dual issue pipelines on each SPE
  - Register level: Native SIMD on SPE and PPE VMX
- 256KB local store per SPE: **data + code + stack**
- Communication
  - DMA and Bus bandwidth
    - DMA granularity – 128 bytes
    - DMA bandwidth among LS and System memory
  - Traffic control
    - Exploit computational complexity and data locality to lower data traffic requirement
  - Shared memory / Message passing abstraction overhead
  - Synchronization
  - DMA latency handling

Typical CELL Software Development Flow

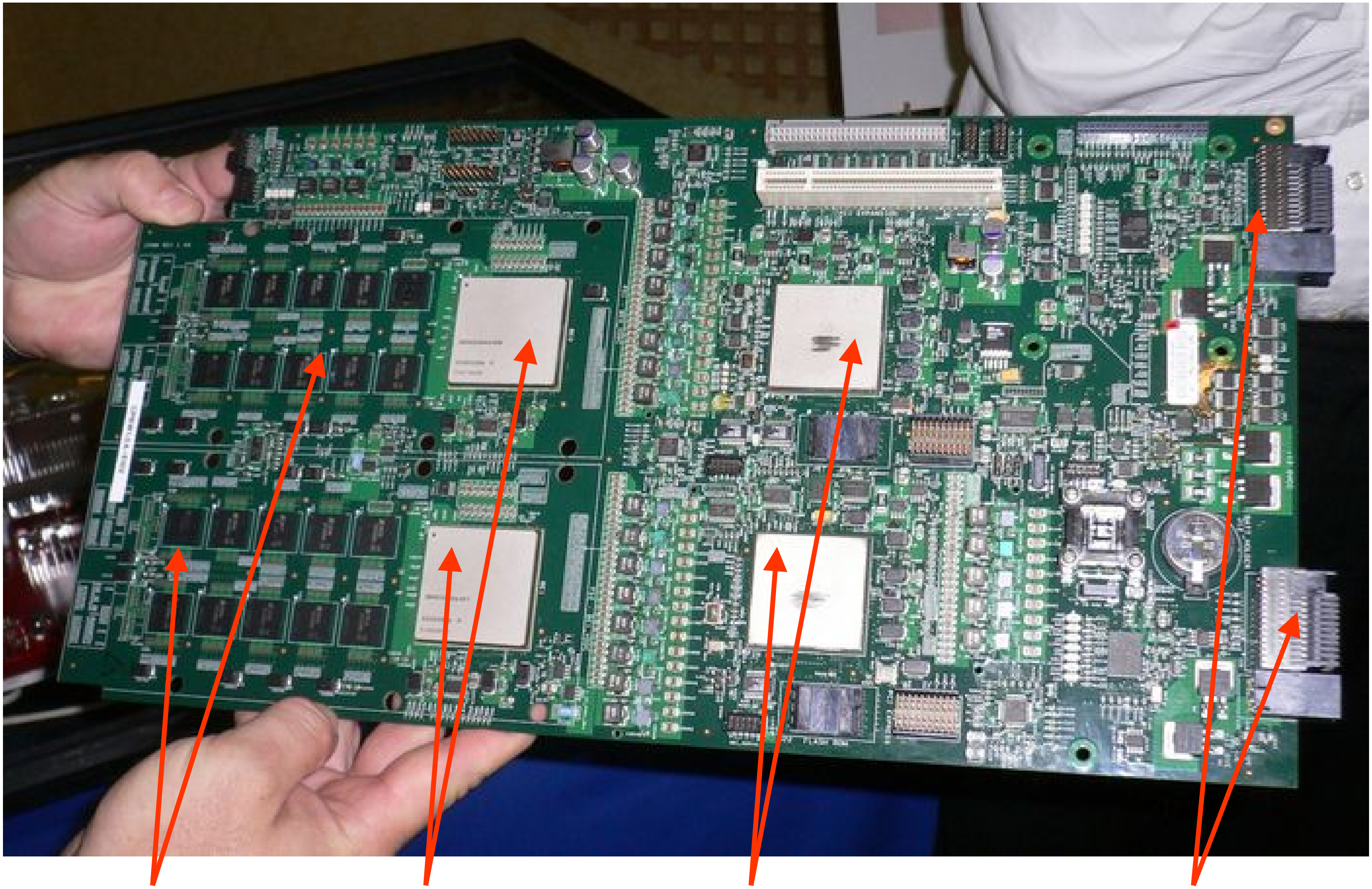
- Algorithm complexity study
- Data layout/locality and Data flow analysis
- Experimental partitioning and mapping of the algorithm and program structure to the architecture
- Develop PPE Control, PPE Scalar code
- Develop PPE Control, partitioned SPE scalar code
  - Communication, synchronization, latency handling
- Transform SPE scalar code to SPE SIMD code
- Re-balance the computation / data movement
- Other optimization considerations
  - PPE SIMD, system bottleneck, load balance

6.189 IAP 2007

Lecture 2

Cell Blade

The First Generation Cell Blade



1GB XDR Memory

Cell Processors

IO Controllers

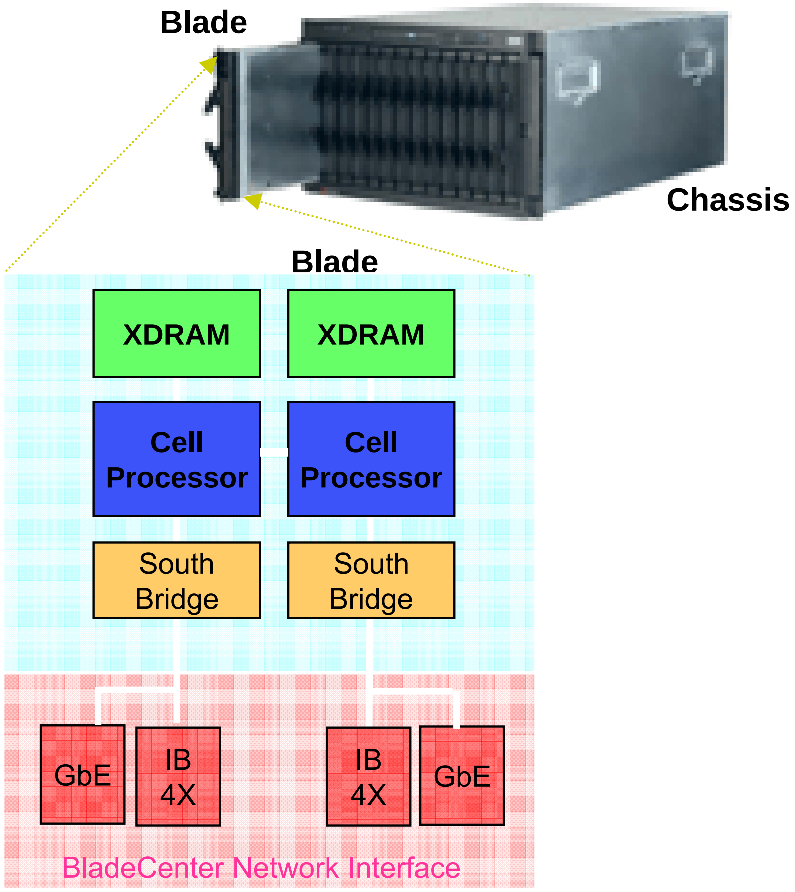
IBM Blade Center interface

Cell Blade Overview

- Blade
  - Two Cell BE Processors
  - 1GB XDRAM
  - BladeCenter Interface (Based on IBM JS20)

- Chassis
  - Standard IBM BladeCenter form factor with:
    - 7 Blades (for 2 slots each) with full performance
    - 2 switches (1Gb Ethernet) with 4 external ports each
  - Updated Management Module Firmware.
  - External Infiniband Switches with optional FC ports

- Typical Configuration (available today from E&TS)
  - eServer 25U Rack
  - 7U Chassis with Cell BE Blades, OpenPower 710
  - Nortel GbE switch
  - GCC C/C++ (Barcelona) or XLC Compiler for Cell (alphaworks)
  - SDK Kit on http://www-128.ibm.com/developerworks/power/cell/



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

Special Notices

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client's credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

All prices shown are IBM's United States suggested list prices and are subject to change without notice; reseller prices may vary.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Many of the features described in this document are operating system dependent and may not be available on Linux. For more information, please check: http://www.ibm.com/systems/p/software/whitepapers/linux_overview.html

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.

Special Notices (Cont.) -- Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States and/or other countries: alphaWorks, BladeCenter, Blue Gene, ClusterProven, developerWorks, e business(logo), e(logo)business, e(logo)server, IBM, IBM(logo), ibm.com, IBM Business Partner (logo), IntelliStation, MediaStreamer, Micro Channel, NUMA-Q, PartnerWorld, PowerPC, PowerPC(logo), pSeries, TotalStorage, xSeries; Advanced Micro-Partitioning, eServer, Micro-Partitioning, NUMACenter, On Demand Business logo, OpenPower, POWER, Power Architecture, Power Everywhere, Power Family, Power PC, PowerPC Architecture, POWER5, POWER5+, POWER6, POWER6+, Redbooks, System p, System p5, System Storage, VideoCharger, Virtualization Engine.

A full list of U.S. trademarks owned by IBM may be found at: http://www.ibm.com/legal/copytrade.shtml.

Cell Broadband Engine and Cell Broadband Engine Architecture are trademarks of Sony Computer Entertainment, Inc. in the United States, other countries, or both.
Rambus is a registered trademark of Rambus, Inc.
XDR and FlexIO are trademarks of Rambus, Inc.
UNIX is a registered trademark in the United States, other countries or both.
Linux is a trademark of Linus Torvalds in the United States, other countries or both.
Fedora is a trademark of Redhat, Inc.
Microsoft, Windows, Windows NT and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries or both.
Intel, Intel Xeon, Itanium and Pentium are trademarks or registered trademarks of Intel Corporation in the United States and/or other countries.
AMD Opteron is a trademark of Advanced Micro Devices, Inc.
Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States and/or other countries.
TPC-C and TPC-H are trademarks of the Transaction Performance Processing Council (TPPC).
SPECint, SPECfp, SPECjbb, SPECweb, SPECjAppServer, SPEC OMP, SPECviewperf, SPECapc, SPEChpc, SPECjvm, SPECmail, SPECimap and SPECsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).
AltiVec is a trademark of Freescale Semiconductor, Inc.
PCI-X and PCI Express are registered trademarks of PCI SIG.
InfiniBand™ is a trademark the InfiniBand® Trade Association
Other company, product and service names may be trademarks or service marks of others.

Revised July 23, 2006

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

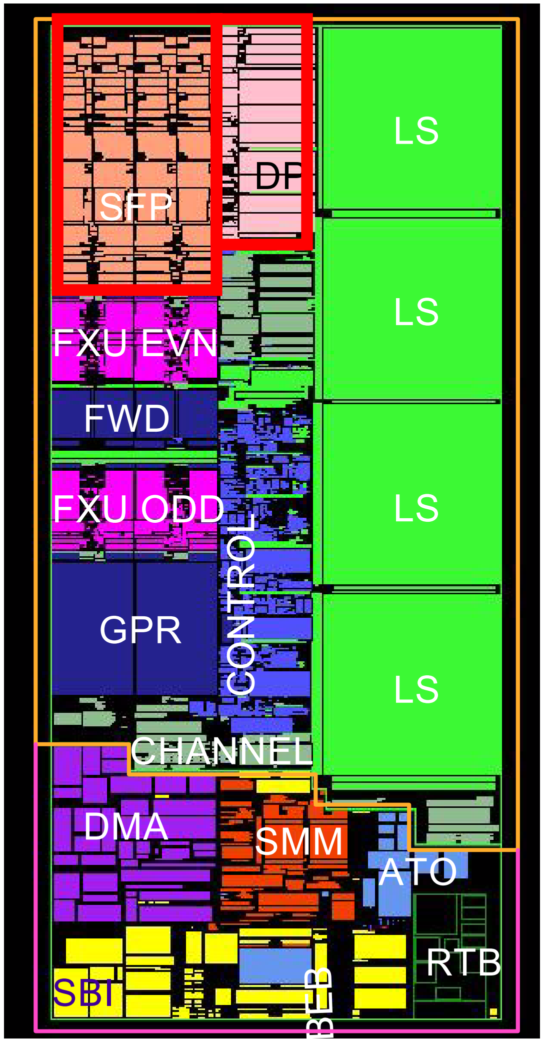
The IBM home page is http://www.ibm.com
The IBM Microelectronics Division home page is
http://www.chips.ibm.com

6.189 IAP 2007

Lecture 2

Backup Slides

SPE Highlights

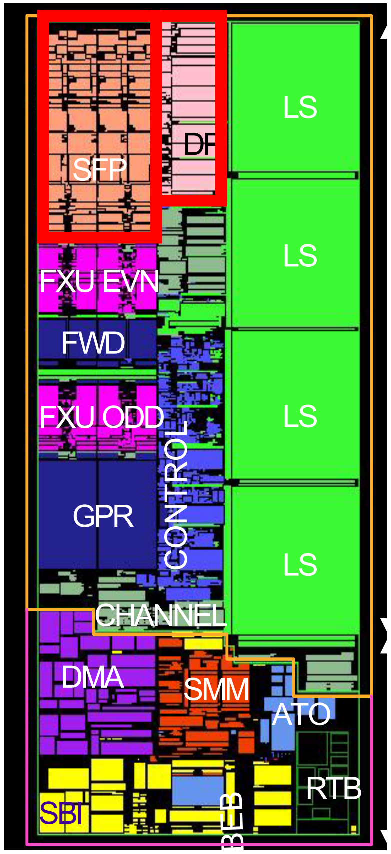


14.5mm^2 (90nm SOI)

- 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

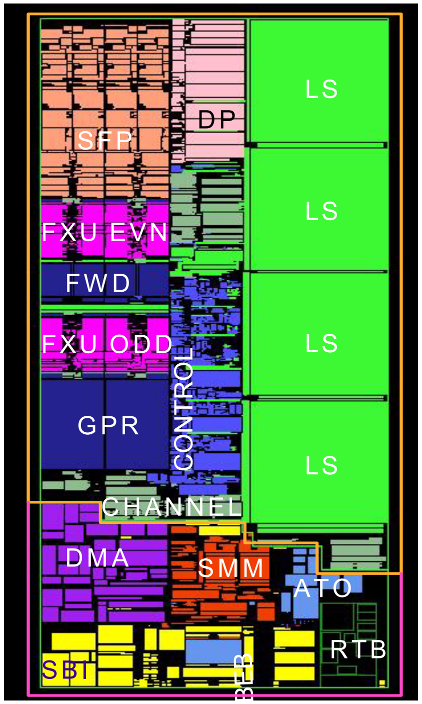
What is a Synergistic Processor? (and why is it efficient?)

- 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



SPU Details

- 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
- Memory Flow Control (MFC)

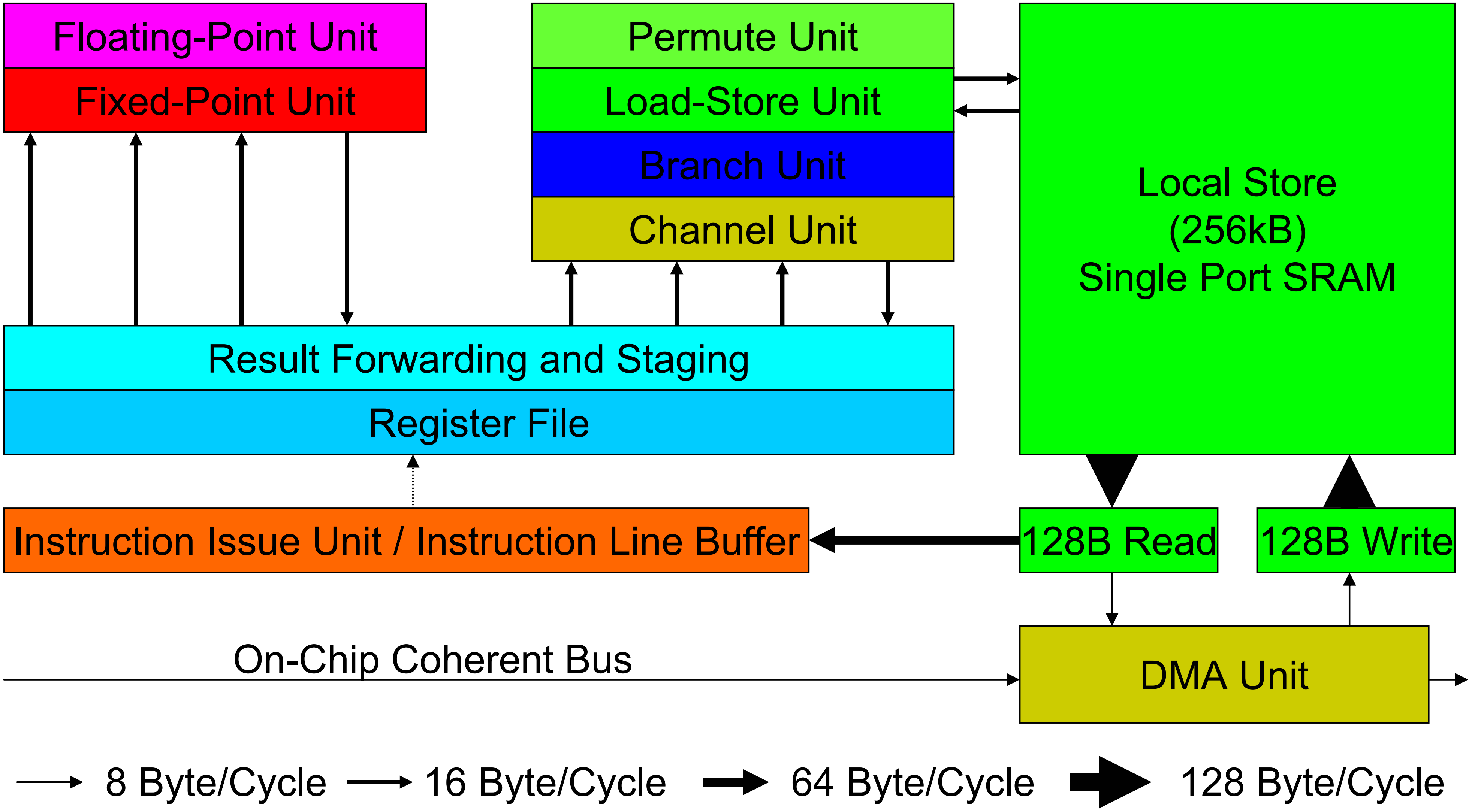


- SPU Units
  - Simple (FXU even)
    - Add/Compare
    - Rotate
    - Logical, Count Leading Zero
  - Permute (FXU odd)
    - Permute
    - Table-lookup
  - FPU (Single / Double Precision)
  - Control (SCN)
    - Dual Issue, Load/Store, ECC Handling
  - Channel (SSC) – Interface to MFC
  - Register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

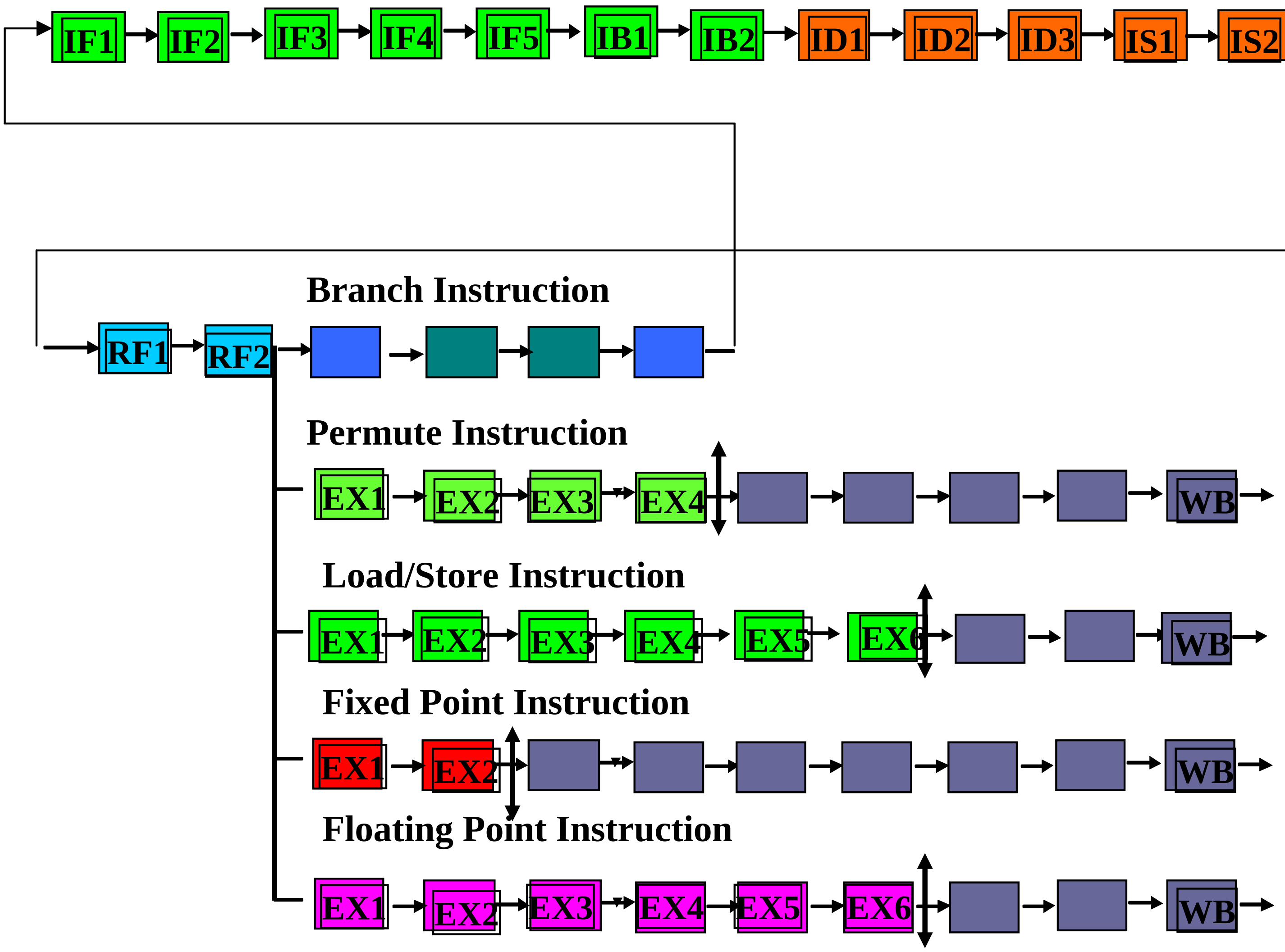
- SPU Latencies

Simple fixed point	- 2 cycles*
Complex fixed point	- 4 cycles*
Load	- 6 cycles*
Single-precision (ER) float	- 6 cycles*
Integer multiply	- 7 cycles*
Branch miss (no penalty for correct hint)	- 20 cycles
DP (IEEE) float (partially pipelined)	- 13 cycles*
Enqueue DMA Command	- 20 cycles*

SPE Block Diagram

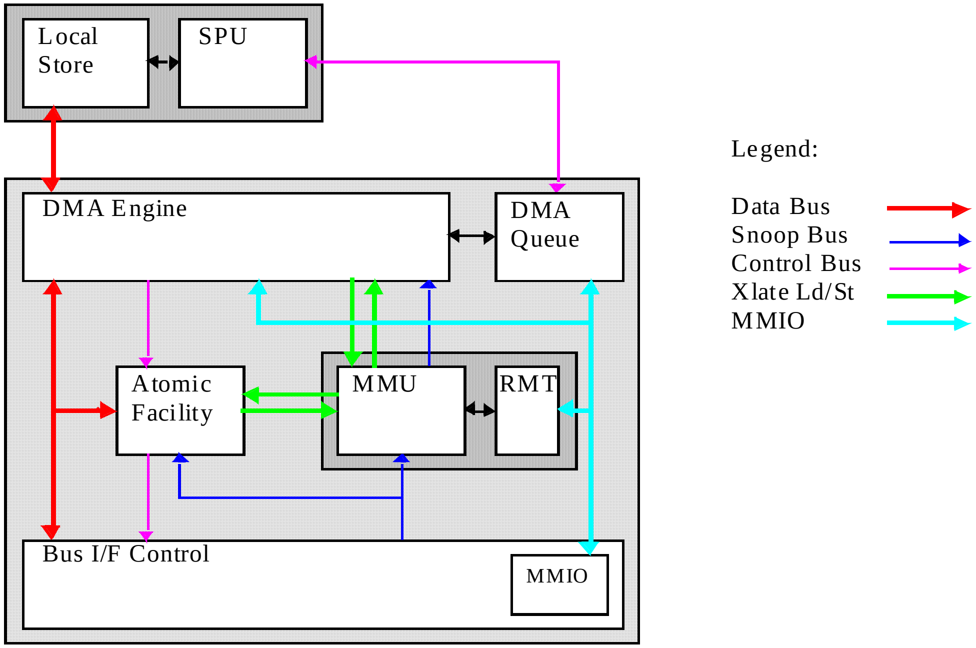


SXU Pipeline



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

MFC Detail



- Isolation Mode Support (Security Feature)
- Hardware enforced “isolation”
  - SPU 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

- 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
  - 4K, 64K, 1M, 16M 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

Per SPE Resources (PPE Side)

Problem State

4K Physical Page Boundary

8 Entry MFC Command Queue Interface
DMA Command and Queue Status
DMA Tag Status Query Mask
DMA Tag Status
32 bit Mailbox Status and Data from SPU
32 bit Mailbox Status and Data to SPU
4 deep FIFO
Signal Notification 1
Signal Notification 2
SPU Run Control
SPU Next Program Counter
SPU Execution Status

4K Physical Page Boundary

Optionally Mapped 256K Local Store

Privileged 1 State (OS)

4K Physical Page Boundary

SPU Privileged Control
SPU Channel Counter Initialize
SPU Channel Data Initialize
SPU Signal Notification Control
SPU Decrementer Status & Control
MFC DMA Control
MFC Context Save / Restore Registers
SLB Management Registers

4K Physical Page Boundary

Optionally Mapped 256K Local Store

Privileged 2 State (OS or Hypervisor)

4K Physical Page Boundary

SPU Master Run Control
SPU ID
SPU ECC Control
SPU ECC Status
SPU ECC Address
SPU 32 bit PU Interrupt Mailbox
MFC Interrupt Mask
MFC Interrupt Status
MFC DMA Privileged Control
MFC Command Error Register
MFC Command Translation Fault Register
MFC SDR (PT Anchor)
MFC ACCR (Address Compare)
MFC DSSR (DSI Status)
MFC DAR (DSI Address)
MFC LPID (logical partition ID)
MFC TLB Management Registers

Per SPE Resources (SPU Side)

SPU Direct Access Resources

- 128 - 128 bit GPRs
- External Event Status (Channel 0)
  - Decrementer Event
  - Tag Status Update Event
  - DMA Queue Vacancy Event
  - SPU Incoming Mailbox Event
  - Signal 1 Notification Event
  - Signal 2 Notification Event
  - Reservation Lost Event
- External Event Mask (Channel 1)
- External Event Acknowledgement (Channel 2)
- Signal Notification 1 (Channel 3)
- Signal Notificaiton 2 (Channel 4)
- Set Decrementer Count (Channel 7)
- Read Decrementer Count (Channel 8)
- 16 Entry MFC Command Queue Interface (Channels 16-21)
- DMA Tag Group Query Mask (Channel 22)
- Request Tag Status Update (Channel 23)
  - Immediate
  - Conditional - ALL
  - Conditional - ANY
- Read DMA Tag Group Status (Channel 24)
- DMA List Stall and Notify Tag Status (Channel 25)
- DMA List Stall and Notify Tag Acknowledgement (Channel 26)
- Lock Line Command Status (Channel 27)
- Outgoing Mailbox to PU (Channel 28)
- Incoming Mailbox from PU (Channel 29)
- Outgoing Interrupt Mailbox to PU (Channel 30)

SPU Indirect Access Resources
(via EA Addressed DMA)

- System Memory
- Memory Mapped I/O
- This SPU Local Store
- Other SPU Local Store
- Other SPU Signal Registers
- Atomic Update (Cacheable Memory)

Memory Flow Controller Commands

DMA Commands

Put - Transfer from Local Store to EA space

Puts - Transfer and Start SPU execution

Putr - Put Result - (Arch. Scarf into L2)

Putl - Put using DMA List in Local Store

Putrl - Put Result using DMA List in LS (Arch)

Get - Transfer from EA Space to Local Store

Gets - Transfer and Start SPU execution

Getl - Get using DMA List in Local Store

Sndsig - Send Signal to SPU

Command Modifiers: <**f,b**>

f: Embedded Tag Specific Fence

Command will not start until all previous commands in same tag group have completed

b: Embedded Tag Specific Barrier

Command and all subsiquent commands in same tag group will not start until previous commands in same tag group have completed

SL1 Cache Management Commands

sdcrt - Data cache region touch (DMA Get hint)

sdcrtst - Data cache region touch for store (DMA Put hint)

sdcrz - Data cache region zero

sdcrs - Data cache region store

sdcrf - Data cache region flush

Command Parameters

LSA - Local Store Address (32 bit)

EA - Effective Address (32 or 64 bit)

TS - Transfer Size (16 bytes to 16K bytes)

LS - DMA List Size (8 bytes to 16 K bytes)

TG - Tag Group(5 bit)

CL - Cache Management / Bandwidth Class

Synchronization Commands

Lockline (Atomic Update) Commands:

getllar - DMA 128 bytes from EA to LS and set Reservation

putllc - Conditionally DMA 128 bytes from LS to EA

putlluc - Unconditionally DMA 128 bytes from LS to EA

barrier - all previous commands complete before subsiquent commands are started

mfcsync - Results of all previous commands in Tag group are remotely visible

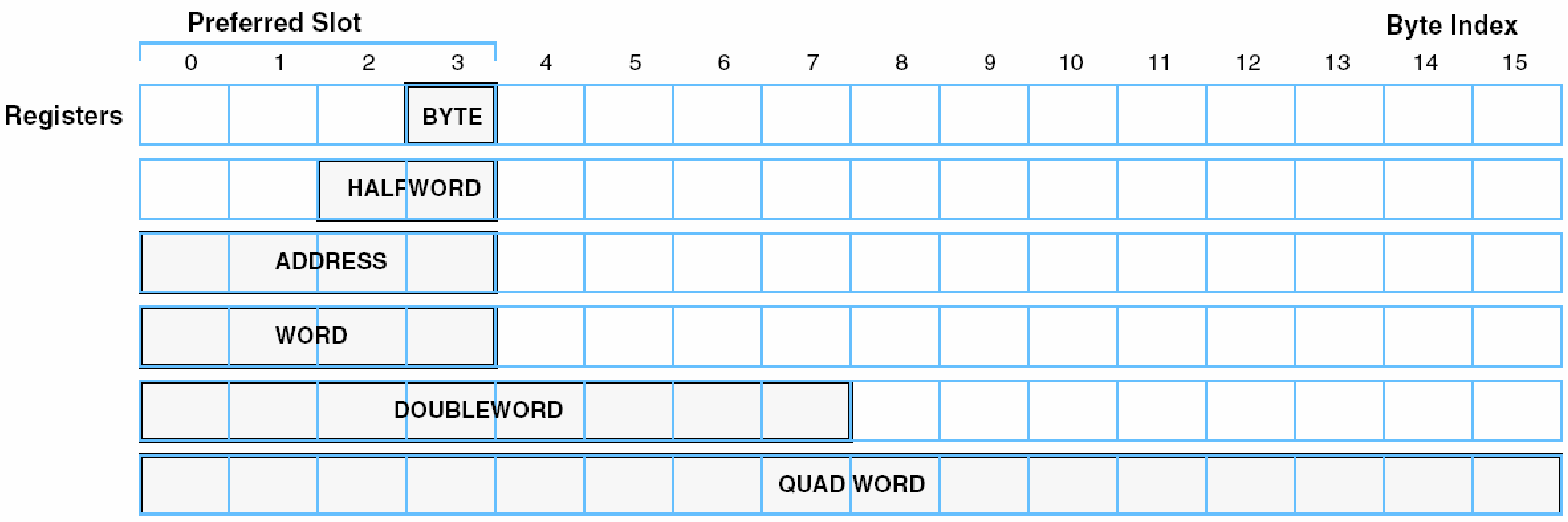
mfceieio - Results of all preceding Puts commands in same group visible with respect to succeeding Get commands

SPE Structure

- Scalar processing supported on data-parallel substrate
  - All instructions are data parallel and operate on vectors of elements
  - Scalar operation defined by instruction use, not opcode
    - Vector instruction form used to perform operation
- Preferred slot paradigm
  - Scalar arguments to instructions found in “preferred slot”
  - Computation can be performed in any slot

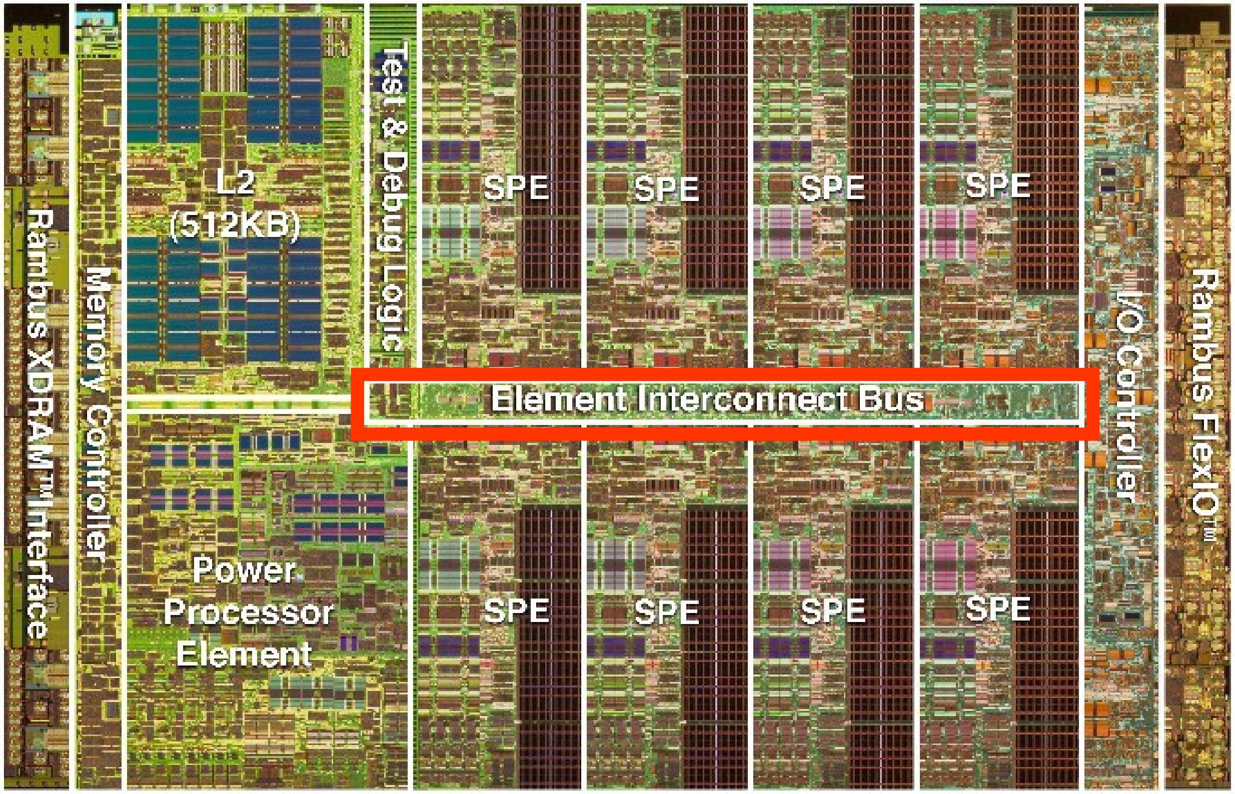
Register Scalar Data Layout

- Preferred slot in bytes 0-3
  - By convention for procedure interfaces
  - Used by instructions expecting scalar data
    - Addresses, branch conditions, generate controls for insert



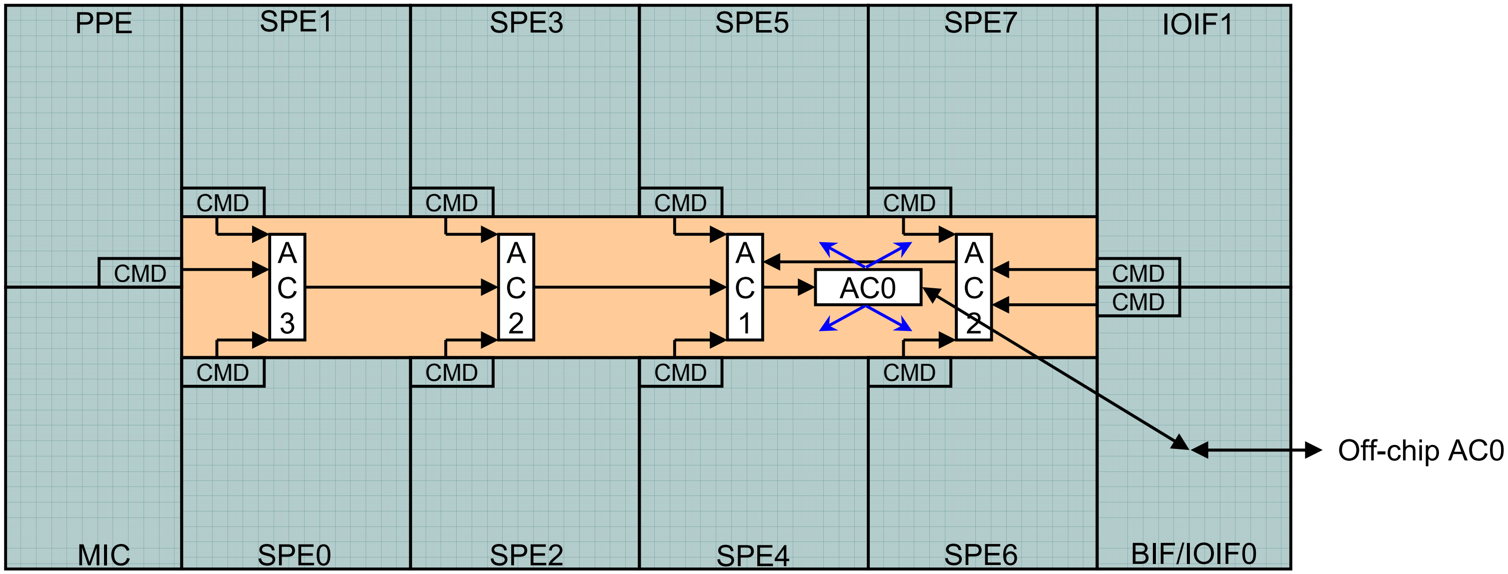
Element Interconnect Bus

- EIB data ring for internal communication
  - Four 16 byte data rings, supporting multiple transfers
  - 96B/cycle peak bandwidth
  - Over 100 outstanding requests



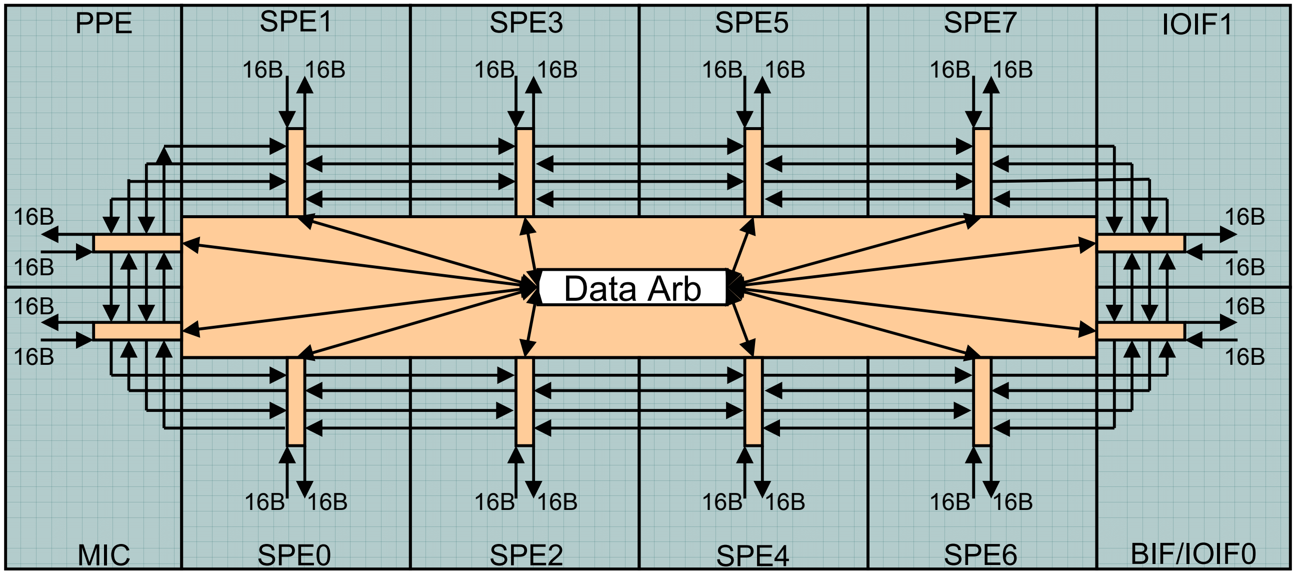
Element Interconnect Bus – Command Topology

- “Address Concentrator” tree structure minimizes wiring resources
- Single serial command reflection point (AC0)
- Address collision detection and prevention
- Fully pipelined
- Content –aware round robin arbitration
- Credit-based flow control



Element Interconnect Bus – Data Topology

- Four 16B data rings connecting 12 bus elements
  - Two clockwise / Two counter-clockwise
- Physically overlaps all processor elements
- Central arbiter supports up to three concurrent transfers per data ring
  - Two stage, dual round robin arbiter
- Each element port simultaneously supports 16B in and 16B out data path
  - Ring topology is transparent to element data interface



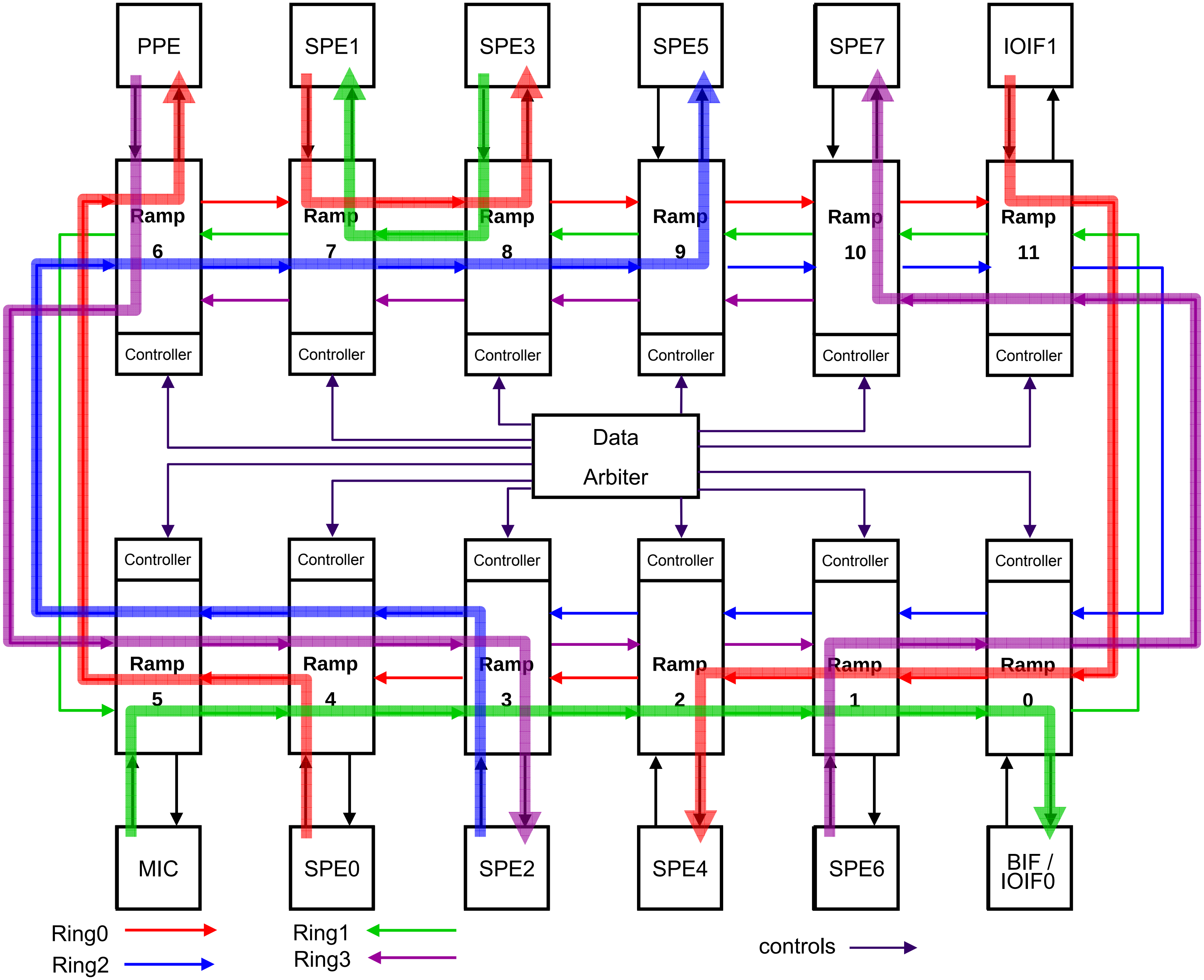
Internal Bandwidth Capability

- Each EIB Bus data port supports 25.6GBytes/sec* in each direction
- The EIB Command Bus streams commands fast enough to support 102.4 GB/sec for coherent commands, and 204.8 GB/sec for non-coherent commands.
- The EIB data rings can sustain 204.8GB/sec for certain workloads, with transient rates as high as 307.2GB/sec between bus units

Despite all that available bandwidth…

* The above numbers assume a 3.2GHz core frequency – internal bandwidth scales with core frequency

Example of Eight Concurrent Transactions


PPE
SPE1
SPE3
SPE5
SPE7
IOIF1
Ramp 6
Ramp 7
Ramp 8
Ramp 9
Ramp 10
Ramp 11
Controller
Data Arbiter
Controller
Ramp 5
Ramp 4
Ramp 3
Ramp 2
Ramp 1
Ramp 0
MIC
SPE0
SPE2
SPE4
SPE6
BIF / IOIF0
Ring0
Ring2
Ring1
Ring3
controls