Broadband Processor Architecture

A novel SIMD architecture for the Cell heterogeneous chip-multiprocessor

Michael Gschwind, Peter Hofstee,
Brian Flachs, Martin Hopkins,
Yukio Watanabe, Takeshi Yamazaki

Acknowledgements

- **Cell is the result of a partnership between SCEI/Sony, Toshiba, and IBM**
- **Cell represents the work of more than 400 people starting in 2000 and a design investment of about $400M**

Cell Design Goals

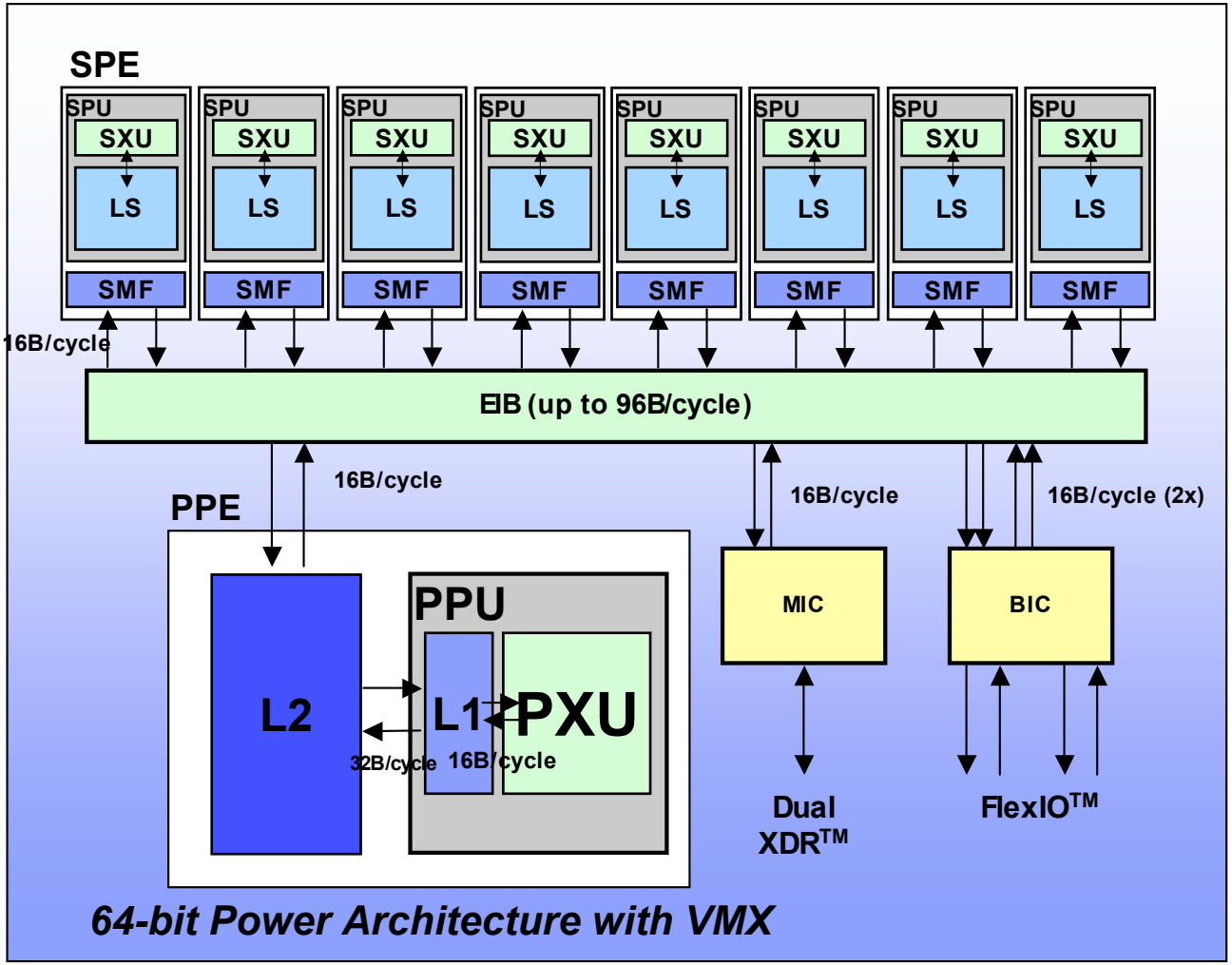
- **Provide the platform for the future of computing**
  - 10× performance of desktop systems shipping in 2005
  - Ability to reach 1 TF with a 4-node Cell system
- **Computing density as main challenge**
  - Dramatically increase performance per X
    - X = Area, Power, Volume, Cost,…
- **Single core designs offer diminishing returns on investment**
  - In power, area, design complexity and verification cost
- **Exploit application parallelism to provide a quantum leap in performance**

Shifting the Balance of Power with Cell

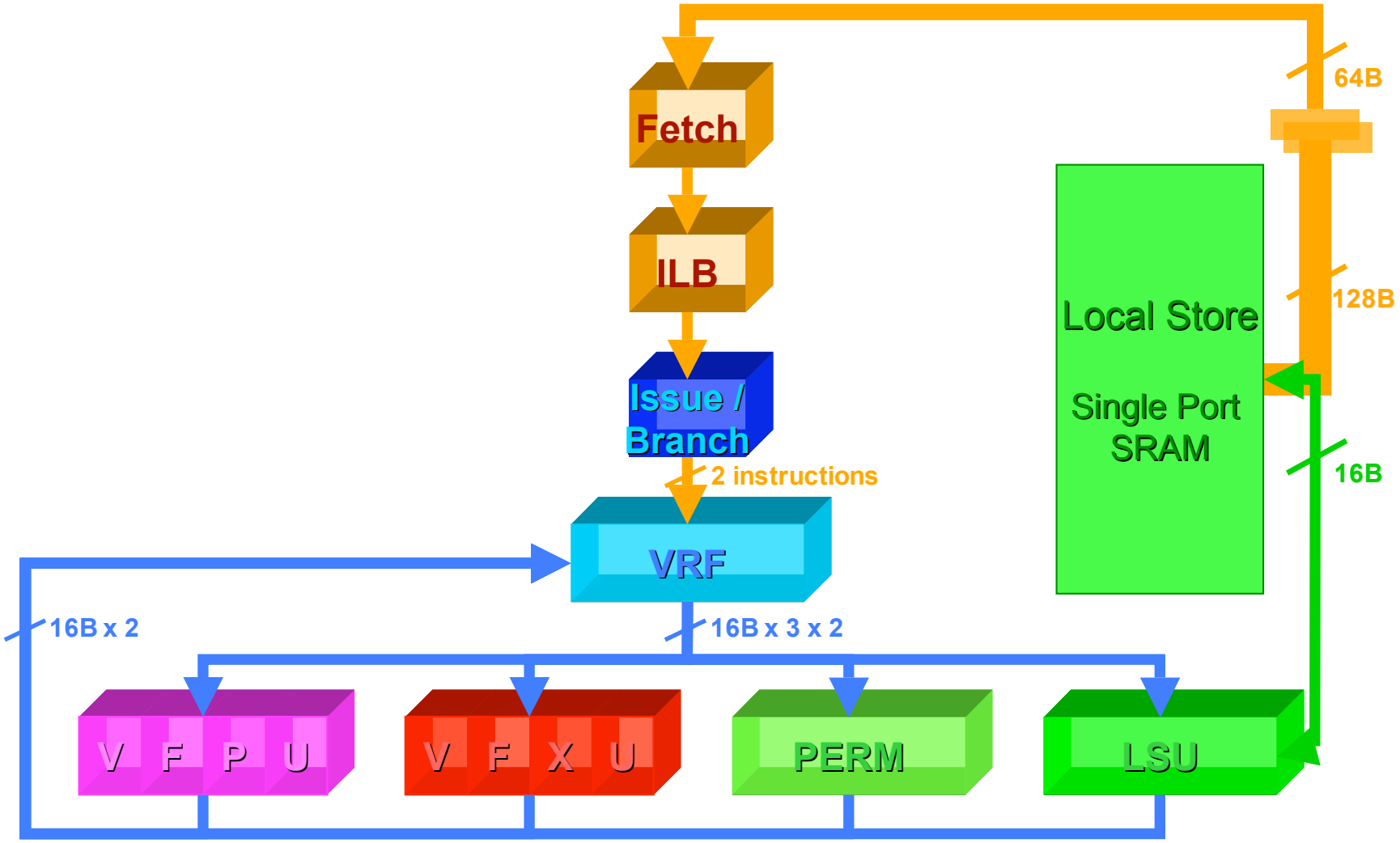
- **Today's architectures are built on a 40 year old data model**
  - Efficiency as defined in 1964
  - Big overhead per data operation
  - Data parallelism added as an after-thought
- **Cell provides parallelism at all levels of system abstraction**
  - Thread-level parallelism → multi-core design approach
  - Instruction-level parallelism → statically scheduled & power aware
  - Data parallelism → data-parallel instructions
- **Data processor instead of control system**

Cell Features

- **Heterogeneous multi-core 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 (SMF)
    - Data movement and synchronization
    - Interface to high-performance Element Interconnect Bus



Powering Cell – the Synergistic Processor Unit



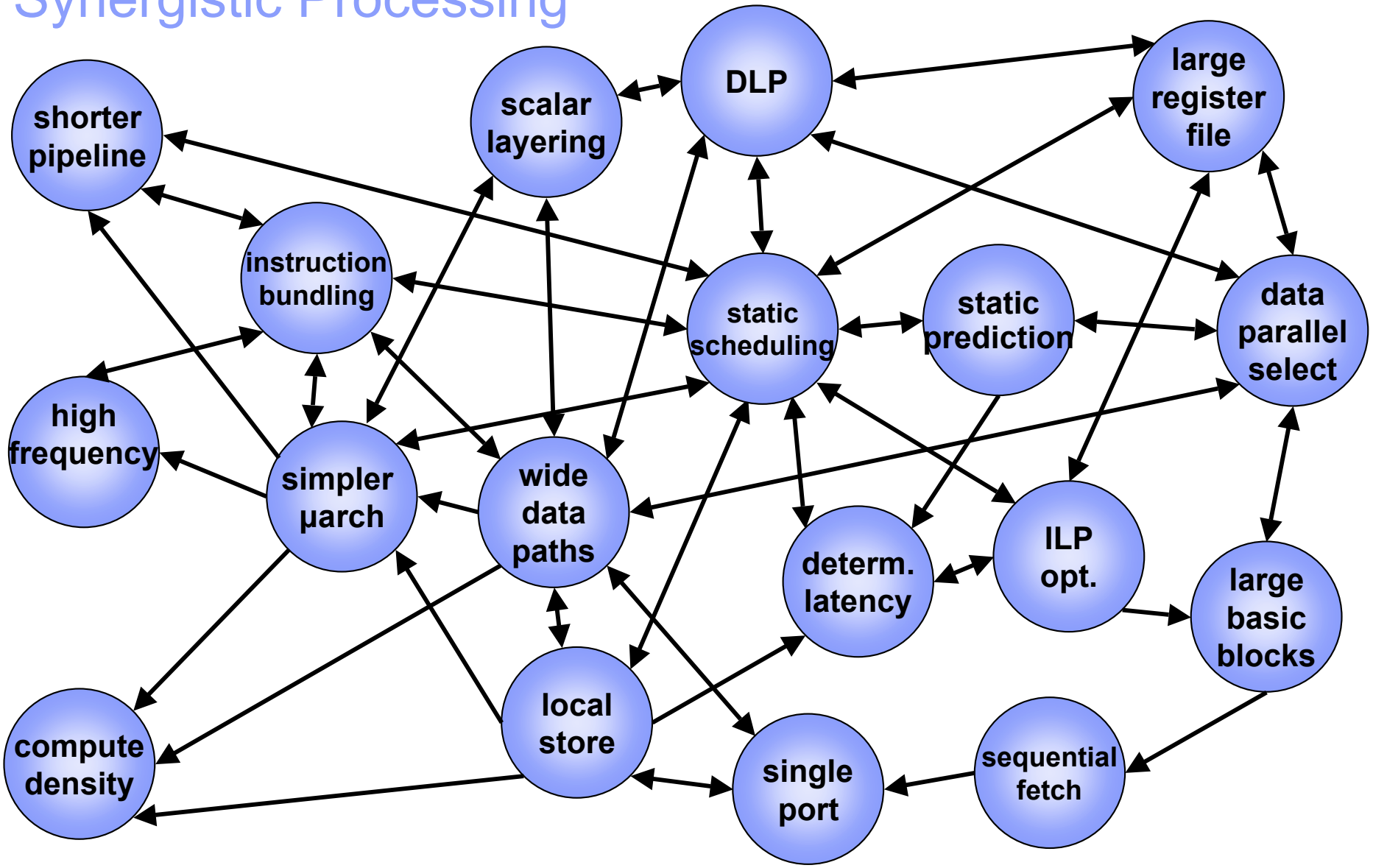
Density Computing in SPEs

- **Today, execution units only fraction of core area and power**
  - Bigger fraction goes to other functions
    - Address translation and privilege levels
    - Instruction reordering
    - Register renaming
    - Cache hierarchy
- **Cell changes this ratio to increase performance per area and power**
  - Architectural focus on data processing
    - Wide datapaths
    - More and wide architectural registers
    - Data privatization and single level processor-local store
    - All code executes in a single (user) privilege level
    - Static scheduling

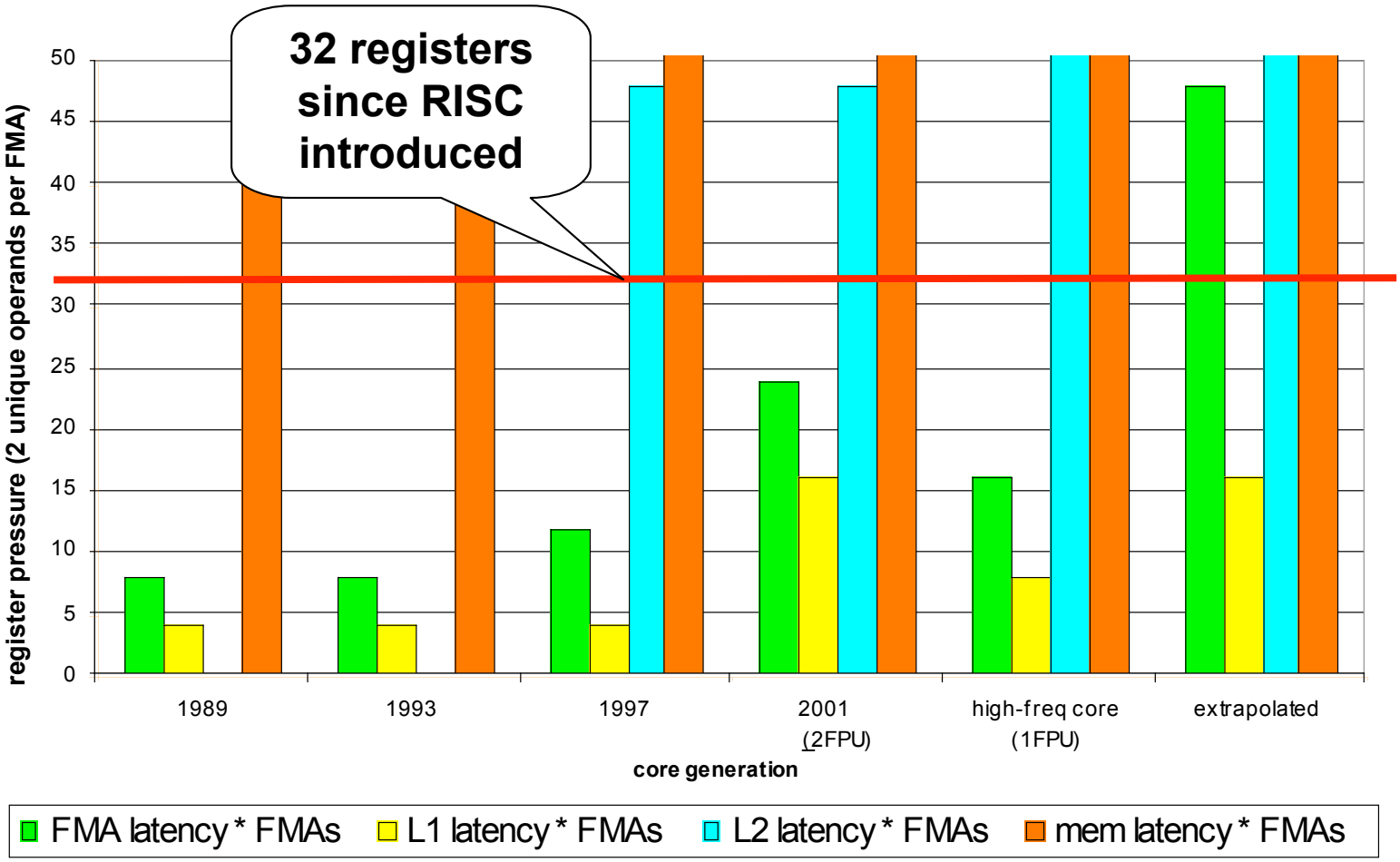
Streamlined Architecture

- **Architectural focus on simplicity**
  - Aids achievable operating frequency
  - Optimize circuits for common performance case
  - Compiler aids in layering traditional hardware functions
  - Leverage 20 years of architecture research
- **Focus on statically scheduled data parallelism**
  - Focus on data parallel instructions
    - No separate scalar execution units
    - Scalar operations mapped onto data parallel dataflow
  - Exploit wide data paths
    - data processing
    - instruction fetch
  - Address impediments to static scheduling
    - Large register set
    - Reduce latencies by eliminating non-essential functionality

Synergistic Processing



Architected Registers have Lagged Behind Latency



Restoring the Architectural Register Balance

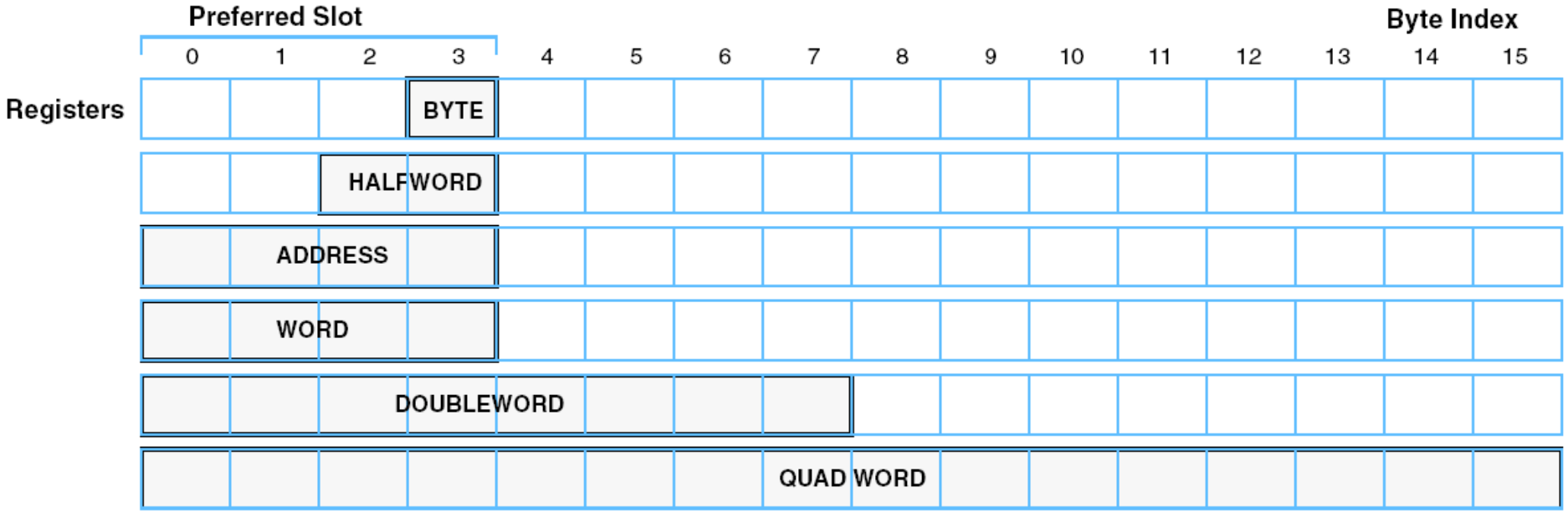
- **Unified 128 entry, 128b wide register file**
  - Used as 128x1, 64x2, 32x4, 16x8, 8x16, 1x128
  - Emphasis on 32x4 integer and FP arithmetic
  - Register file also stores addresses and condition values
- **Unified register file gives programmers more flexibility in resource allocation**
  - Stores all data types
    - address, integer, SP/DP FP, Boolean
  - Stores scalar and vector SIMD data
- **Large register file facilitates static scheduling and loop optimization for latency hiding**

Pervasive Data Parallel Computing (PDPC)

- **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



PDPC Memory Access Architecture

- **Data memory interface optimized for quadword access**
  - Simplified alignment network
  - Always access 128 bits of data
- **Processing scalar data**
  - Extracted using rotate
  - Stored with read-modify-write sequence
- **Four address formats**
  - register + displacement, register-indexed
  - PC-relative, absolute address
- **Local Store Limit Register (LSLR)**
  - Address mask register to define maximum address range
  - Addresses wrap at LSLR value

PDPC Branch Architecture

- **Register-indirect target address in preferred slot of general purpose register**
  - Indirect branch (function return)
  - Indirect prepare-to-branch
- **Link address deposited in preferred slot of specified general purpose register**
  - Register 0 ($ra) used as return register by software convention

A Statically Scheduled PDPC Architecture

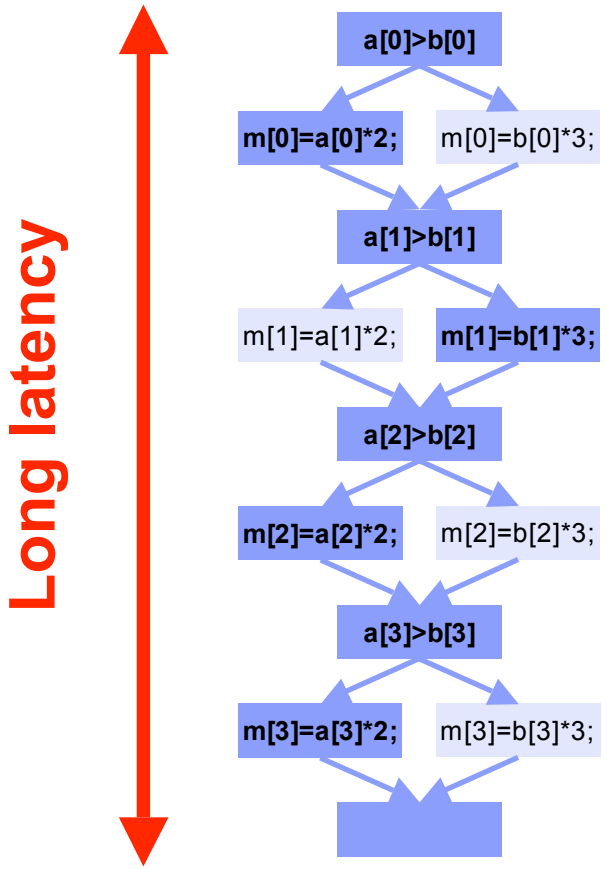
- **Bundle concept**
  - Sequential semantics
  - Static scheduling key ingredient to great performance
  - Strong performance bias to properly aligned instructions
    - Align instructions to map on simplified issue routing logic
    - Maximize fetch group utilization
- **Branch and control architecture**
  - Static branch prediction with Prepare-to-branch instruction
  - Simplify implementations, improve utility of sequential fetch
- **Explicit inline instruction fetch**
  - To avoid fetch starvation during bursts of high-priority load/store traffic
  - Compiler manages 256KB unified, single ported memory

Data Selection Architecture

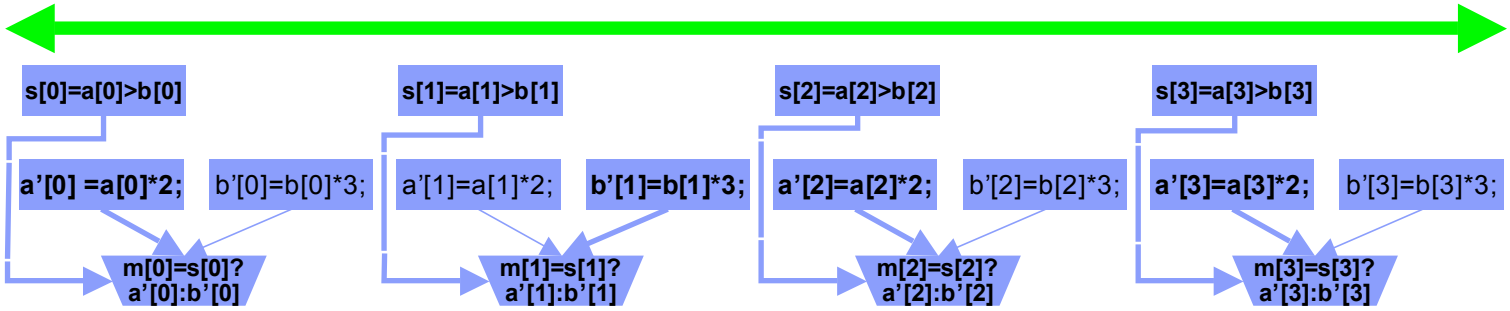
- **Branch is scalar, select is data parallel**
  - Branch inherently inefficient
- **No exception → safe to compute both paths through conditional assignment**
- **Use data parallel compute flow**
  - Data parallel if-conversion
  - Select instruction independently selects result for each slot
  - Compare instructions generate type-specific mask fields

Data Parallel Select Operation

```
for (i =0; i< VL; i++)
  if (a[i]>b[i])
    m[i] = a[i]*2;
  else
    m[i] = b[i]*3;
```



Exploit data parallelism



Compare and Data Selection Architecture

- **Only limited number of compares**
  - Compare for equality
    - CEQ
  - Compare for ordering
    - CGT, CGTL
  - Other conditions can be derived
    - operand order, inverted condition
- **Generates mask of data type width for use with select**
  - Data-parallel conditional operation

Integer Data Processing for Advanced Media Computing

- **Ensure data fidelity**
  - Avoid accumulated saturation error in content creation
  - Emphasis on data quality first, quantity second
  - Concentrate on 16b and 32b integer data
  - Concentrate on modulo arithmetic
    - Consistent with common C/C++/Java semantics
- **Selected byte operations to maximize performance on key operations**
  - Encryption/decryption performance
    - Security in electronic transactions will be key to enable future business models
  - Motion estimation
    - Video encoding

Reconciling Data Fidelity and Compact Data Representation

- **Use wide data types for intermediate results in computation**
  - Avoid saturation of intermediate results
  - Prevent loss of dynamic range
- **Pack-and-saturate operation to store data in dense memory format**
  - Keep data footprint in memory compact
  - Pack and saturate *once* for bulk data storage in memory
- **Saturating computation only important for low-cost content *rendering* devices**
  - No/less accumulation of error across rendering steps
  - *e.g.,* MPEG synchronize with I-frame

Floating Point Processing in the SPE

- **Support standard IEEE FP and graphics-optimized arithmetic**
  - Floating-point naturally saturating
  - Always use IEEE data layout
- **Graphics oriented SP-FP using IEEE compatible data layout**
  - Extended range / Simplified format
    - Improved estimate instructions to reduce “banding”
  - Media workloads impose realtime processing requirements
    - No floating point exceptions
  - Graphics oriented SP-FP mode added to Cell Power Processor Element
- **DP-FP follows IEEE conventions**

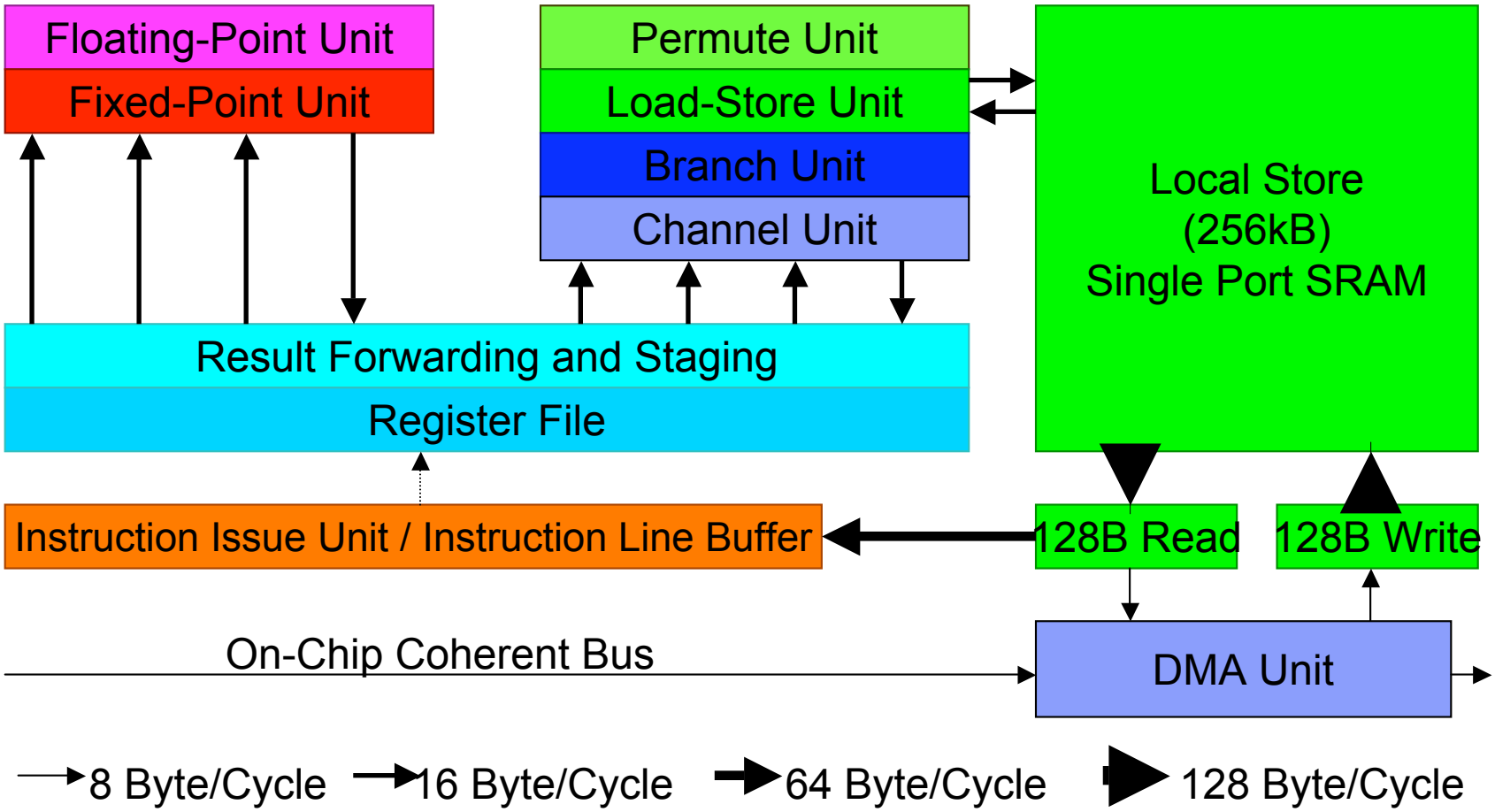
Cell: a Synergistic System Architecture

- **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
- **SPE optimized for efficient data processing**
  - SPEs share Cell system functions provided by Power Architecture
  - SMF implements interface to memory
    - Copy in/copy out to local storage
- **Power Architecture provides system functions**
  - Virtualization
  - Address translation and protection
  - External exception handling
- **EIB** (Element Interconnect Bus) **integrates system as data transport hub**

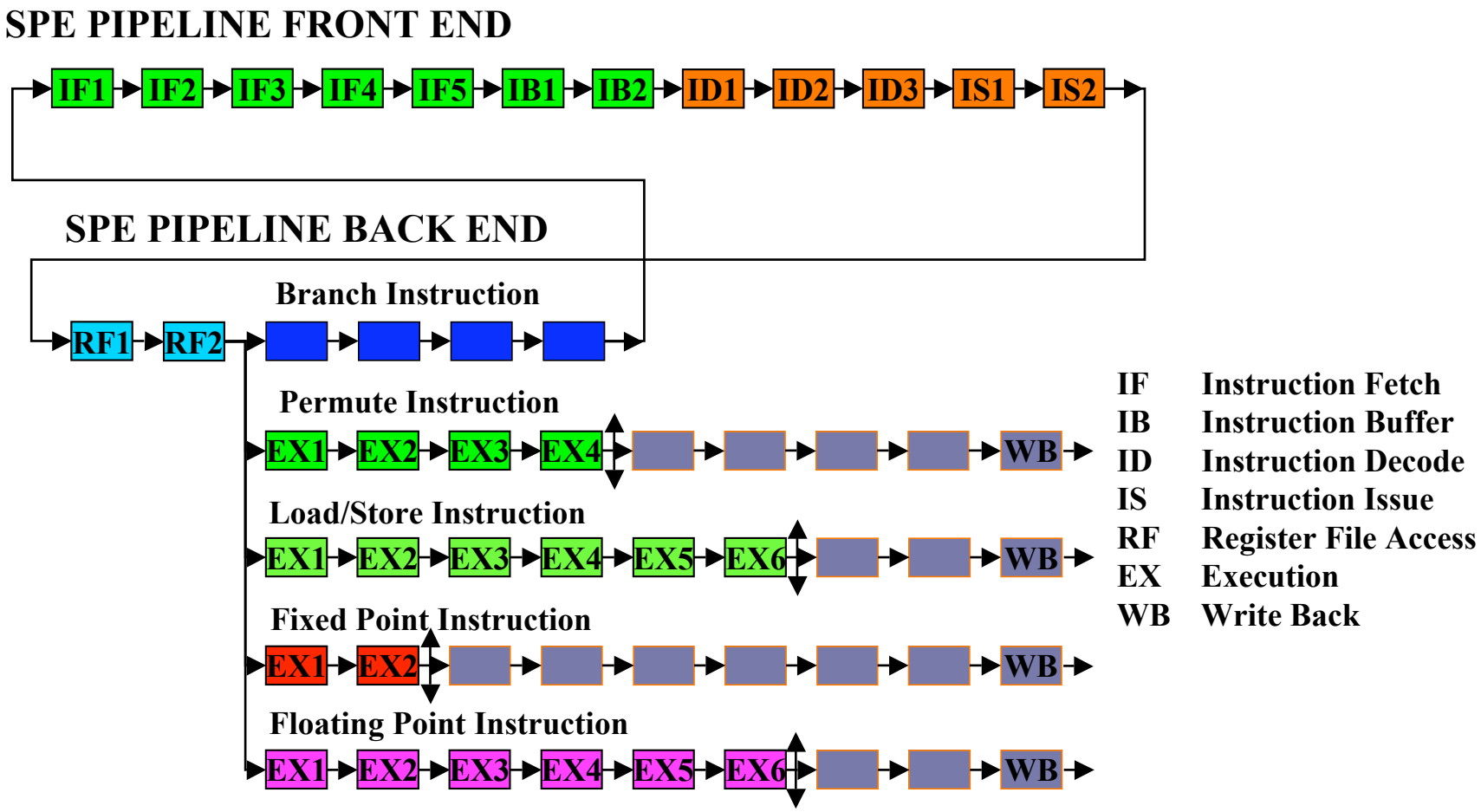
Compiling for Cell

- **The lesson of "RISC computing"**
  - Architecture provides fast, streamlined primitives to compiler
  - Compiler uses primitives to implement higher-level idioms
  - If the compiler can't target it → do not include in architecture
- **Compiler focus throughout project**
  - Prototype compiler soon after first proposal
  - Cell compiler team has made significant advances in
    - Automatic SIMD code generation
    - Automatic parallelization
    - Data privatization

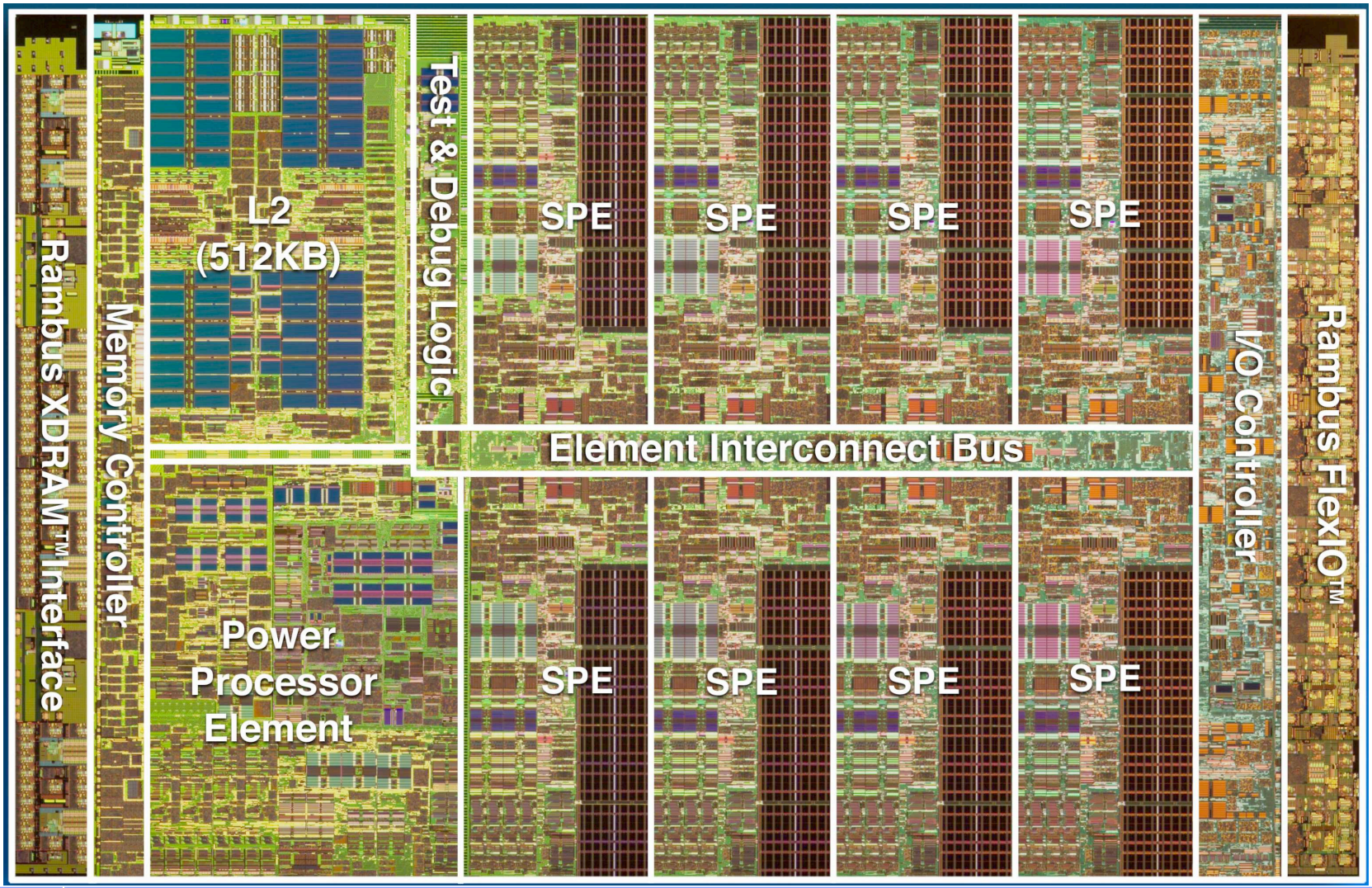
SPE Block Diagram



SPE Pipeline

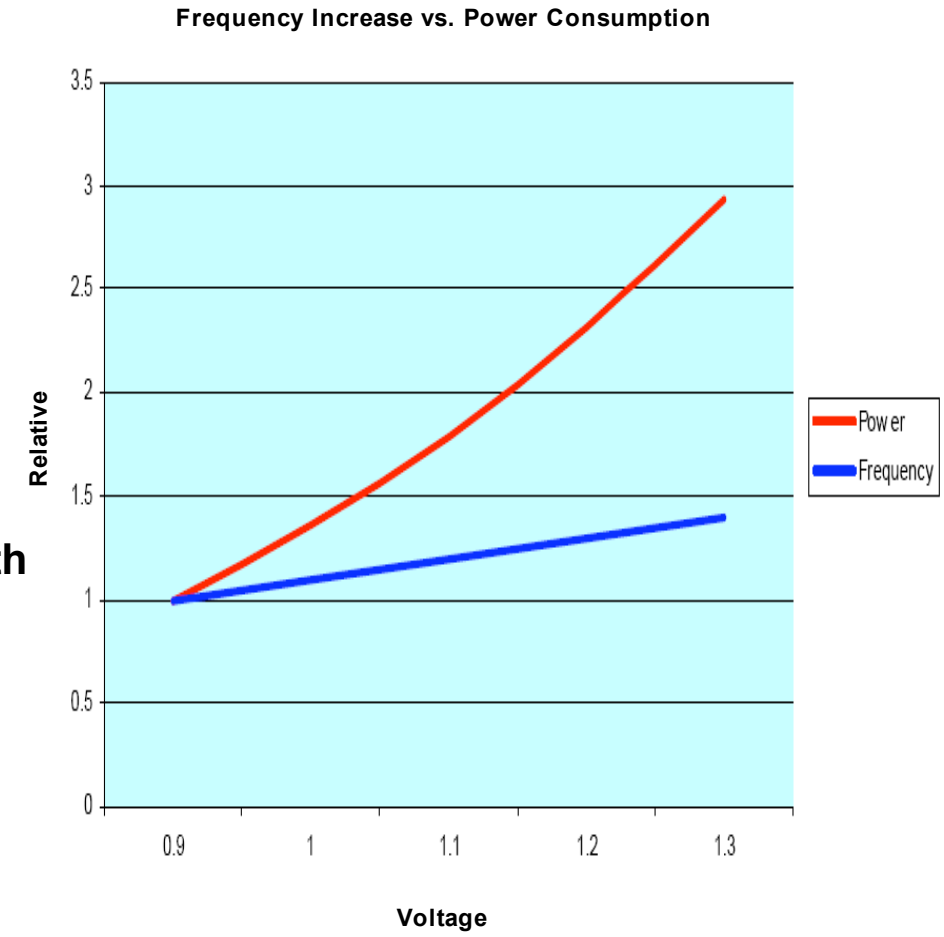


Cell Broadband Engine



Cell Implementation Characteristics

- **241M transistors**
- **235mm^2**
- **Design operates across wide frequency range**
  - Optimize for power & yield
- **> 200 GFlops (SP) @3.2GHz**
- **> 20 GFlops (DP) @3.2GHz**
- **Up to 25.6 GB/s memory bandwidth**
- **Up to 75 GB/s I/O bandwidth**
- **100+ simultaneous bus transactions**
  - 16+8 entry DMA queue per SPE



Conclusion

- **Single chip heterogeneous multicore system takes chip performance to a new level**
  - By exploiting thread level parallelism
  - By exploiting instruction level parallelism
  - By exploiting data level parallelism
- **Novel pervasive vector-centric architecture**
  - Data parallelism at the center
  - Redefines high-performance architecture
- **SPE compute engine offers high compute density**
  - Power-efficient
  - Area-efficient
- **Cell delivers unprecedented supercomputer power for consumer applications**

Additional Information

- **Additional details on the SPE architecture and the Cell implementation can be found at**
  - http://www-306.ibm.com/chips/techlib/techlib.nsf/products/Cell
  - http://www.research.ibm.com/cell