Cell Broadband Engine

Chip Multiprocessing and the Cell Broadband Engine

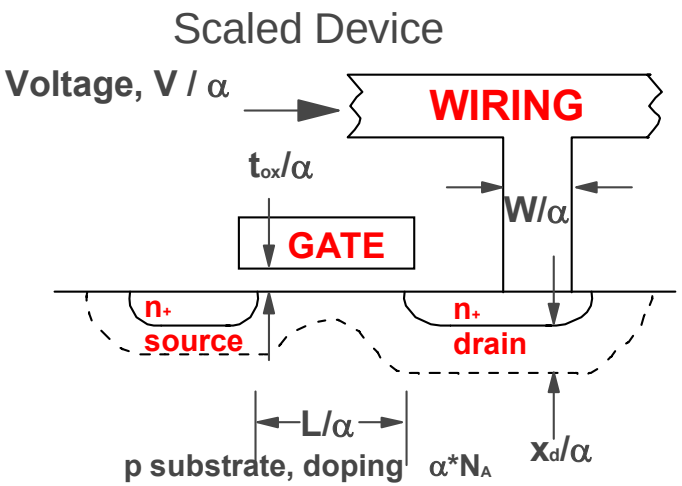
Michael Gschwind

ACM Computing Frontiers 2006

Computer Architecture at the turn of the millenium

- **"The end of architecture" was being proclaimed**
  - Frequency scaling as performance driver
  - State of the art microprocessors
    - multiple instruction issue
    - out of order architecture
    - register renaming
    - deep pipelines
- **Little or no focus on compilers**
  - Questions about the need for compiler research
- **Academic papers focused on microarchitecture tweaks**

The age of frequency scaling



SCALING:

Voltage:	V/α
Oxide:	t_{ox}/α
Wire width:	W/α
Gate width:	L/α
Diffusion:	x_d/α
Substrate:	$\alpha * N_A$

RESULTS:

Higher Density: $\sim\alpha^2$

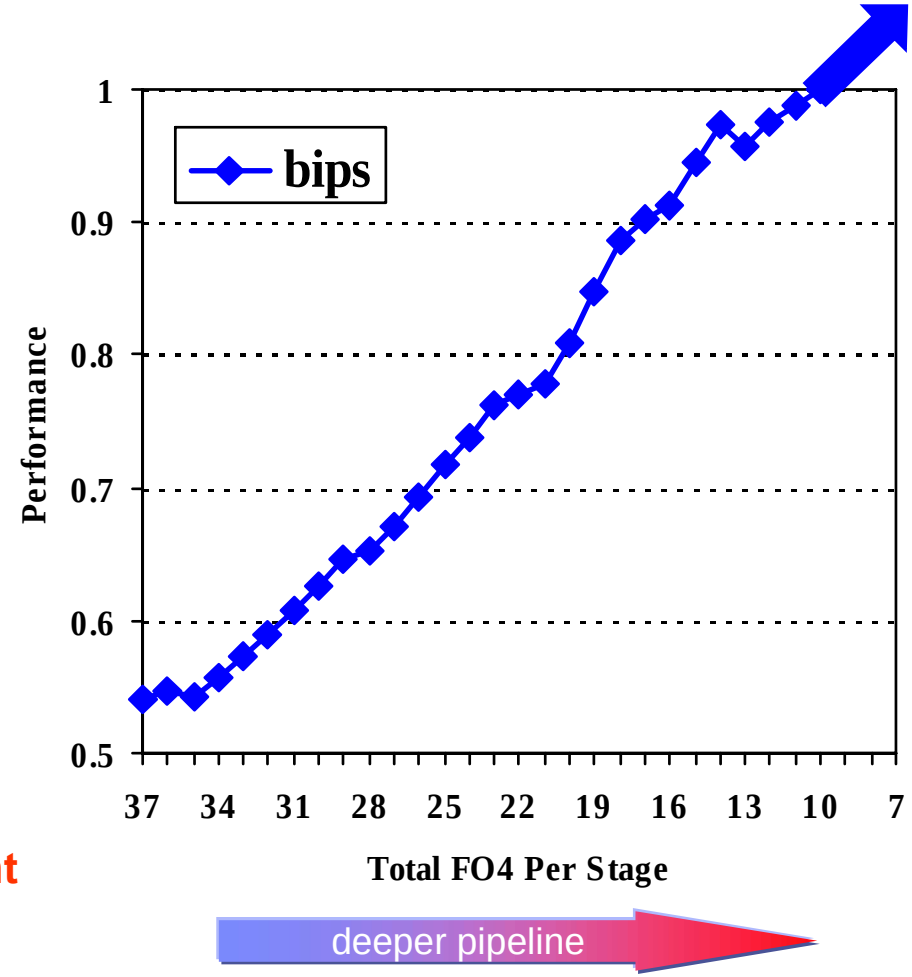
Higher Speed: $\sim\alpha$

Power/ckt: $\sim 1/\alpha^2$

Power Density:

~Constant

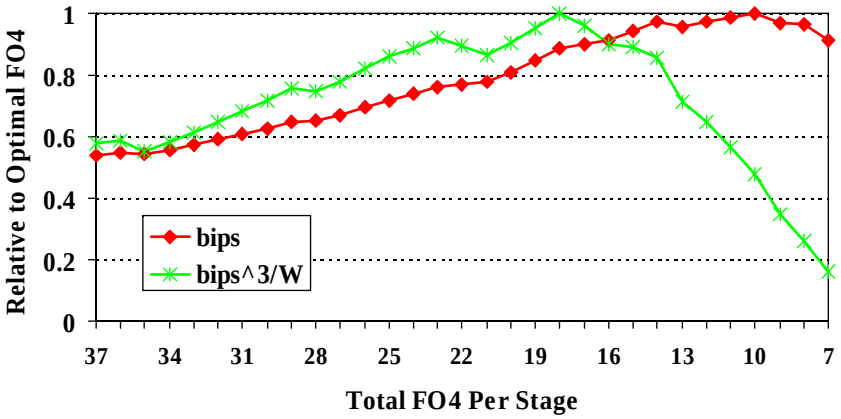
Source: Dennard et al., JSSC 1974.

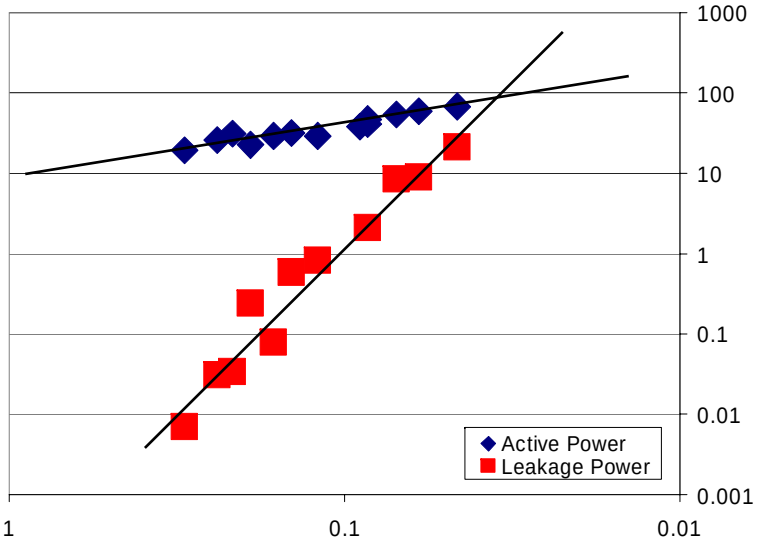


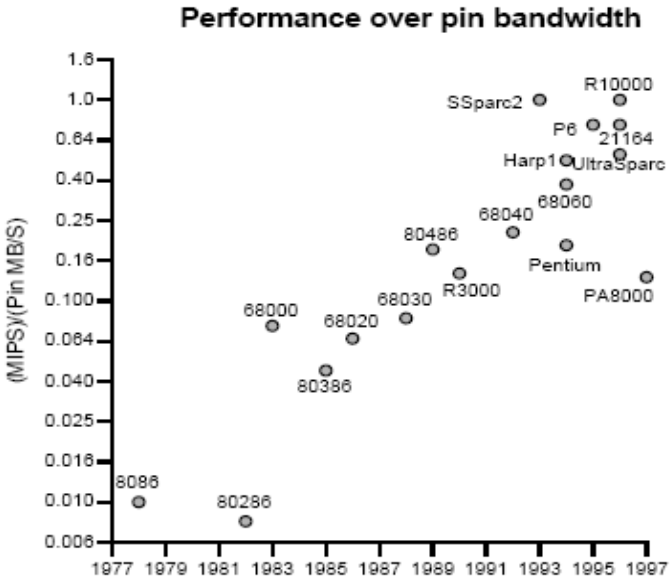
Frequency scaling

- **A trusted standby**
  - Frequency as the tide that raises all boats
  - Increased performance across all applications
  - Kept the industry going for a decade
- **Massive investment to keep going**
  - Equipment cost
  - Power increase
  - Manufacturing variability
  - New materials

… but reality was closing in!

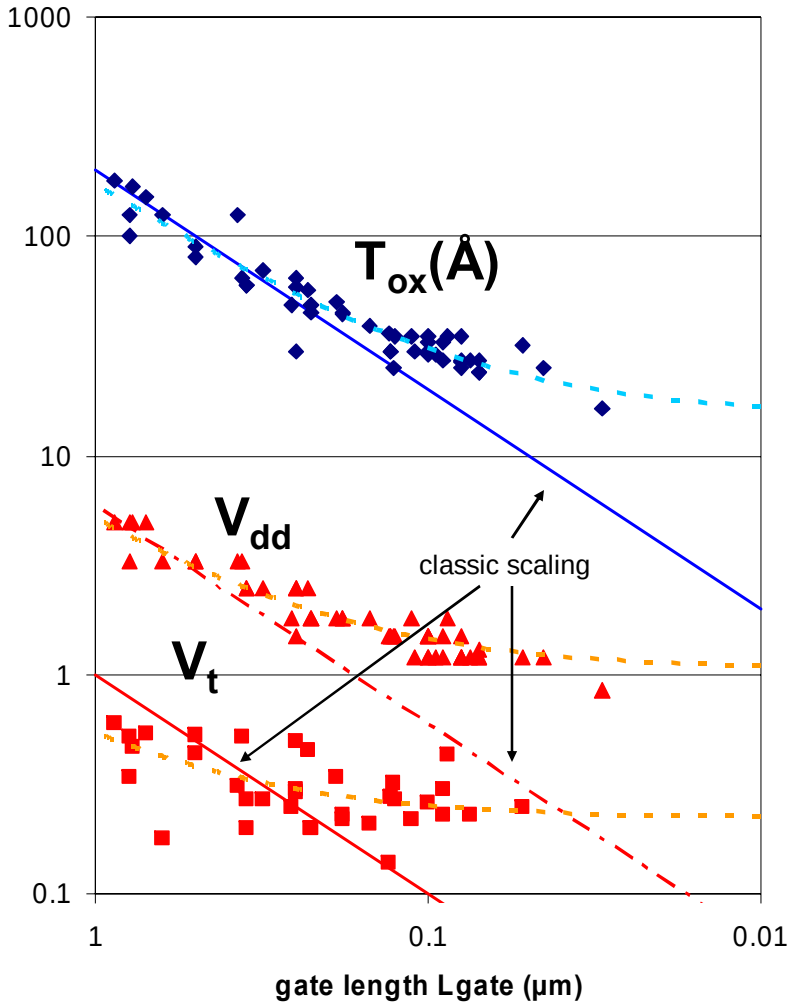




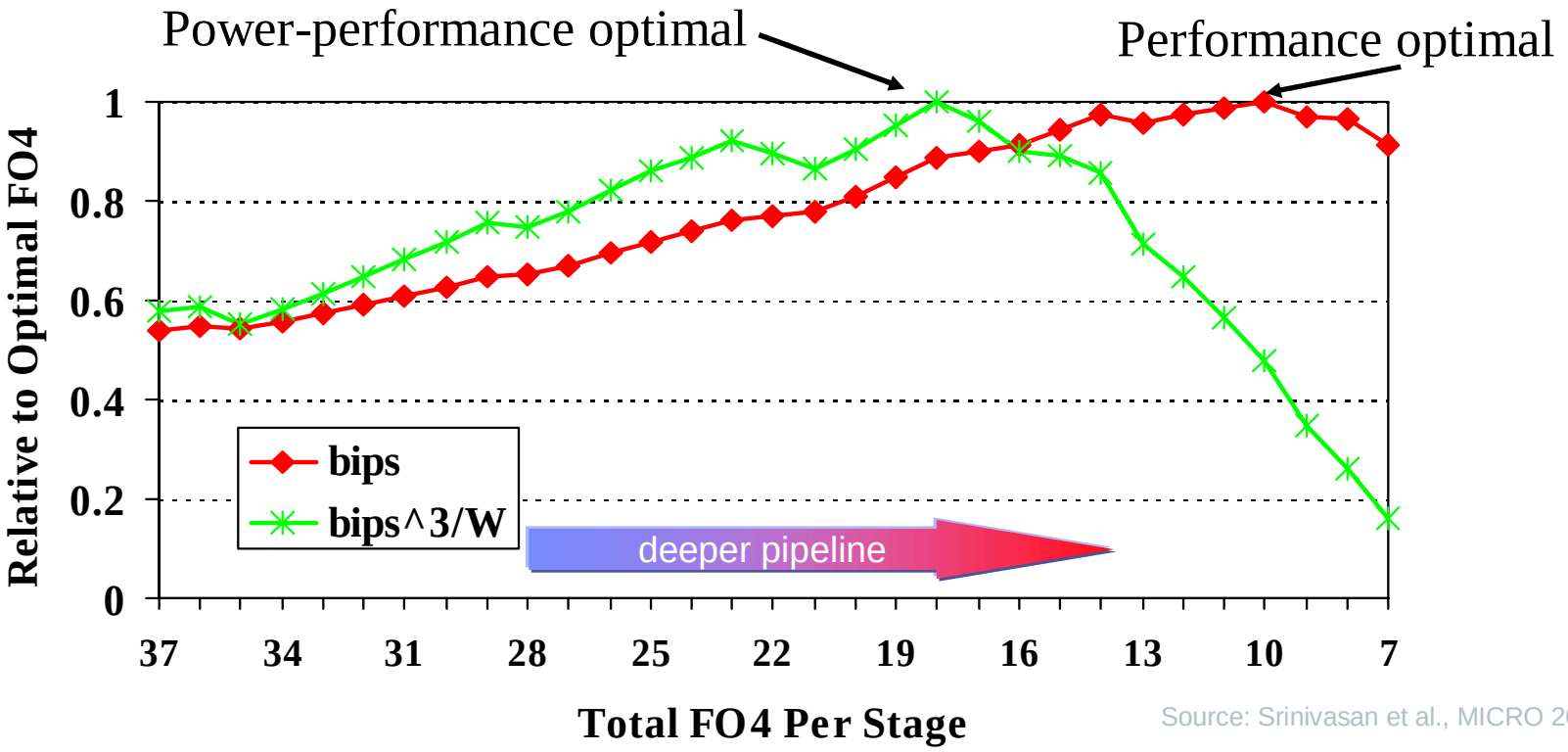


Power crisis

- **The power crisis is not "natural"**
  - Created by deviating from ideal scaling theory
  - V_{dd} and V_t not scaled by α
    - additional performance with increased voltage
- **Laws of physics**
  - $P_{chip} = n_{transistors} * P_{transistor}$
- **Marginal performance gain per transistor low**
  - significant power increase
  - decreased power/performance efficiency



The power inefficiency of deep pipelining



Source: Srinivasan et al., MICRO 2002

- **Deep pipelining increases number of latches and switching rate**

  ⇒ power increases with at least f^2

- **Latch insertion delay limits gains of pipelining**

Hitting the memory wall

- **Latency gap**
  - Memory speedup lags behind processor speedup
  - limits ILP
- **Chip I/O bandwidth gap**
  - Less bandwidth per MIPS
- **Latency gap as application bandwidth gap**

usable bandwidth

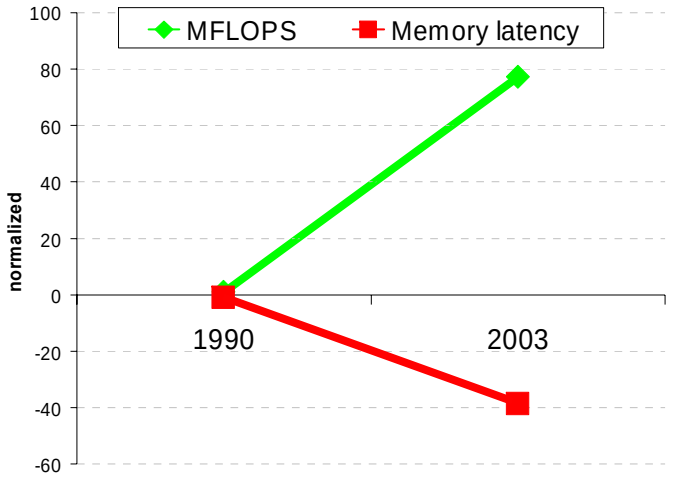
avg. request size

$$BW = \frac{SZ * requests}{latency}$$

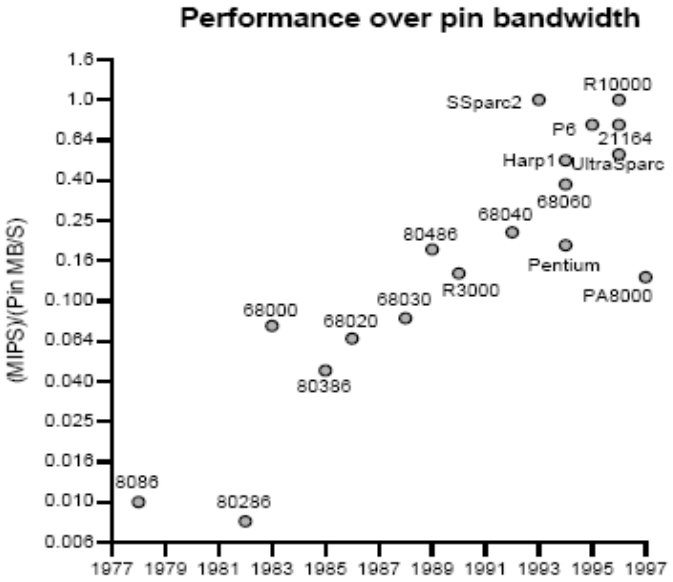
roundtrip latency

no. of inflight requests

  - Typically (much) less than chip I/O bandwidth



Source: McKee, Computing Frontiers 2004



Source: Burger et al., ISCA 1996

Our Y2K challenge

- **10× performance of desktop systems**
- **1 TeraFlop / second with a four-node configuration**
- **1 Byte bandwidth per 1 Flop**
  - "golden rule for balanced supercomputer design"
- **scalable design across a range of design points**
- **mass-produced and low cost**

Cell Design Goals

- **Provide the platform for the future of computing**
  - 10× performance of desktop systems shipping in 2005
- **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

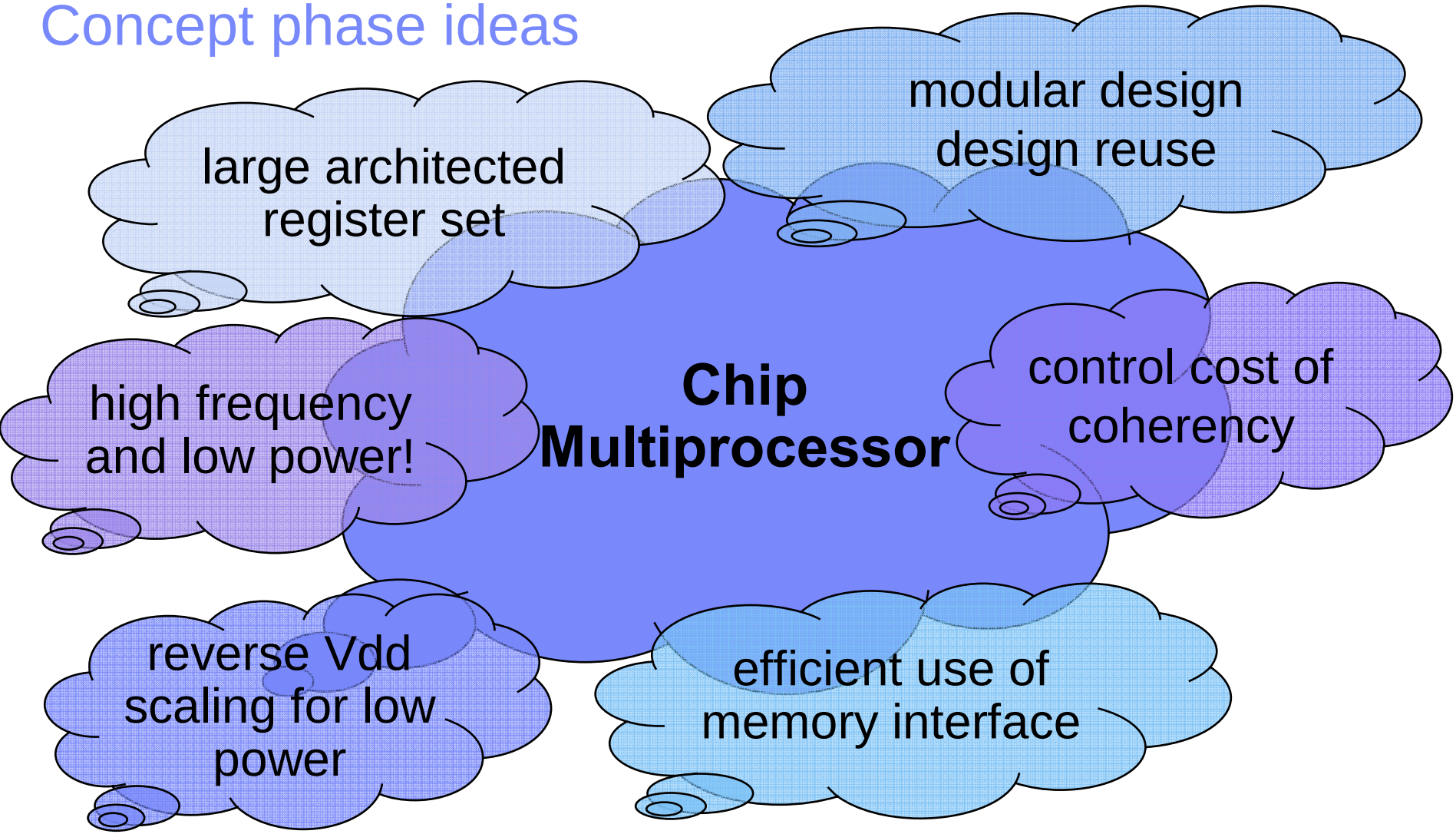
Necessity as the mother of invention

- **Increase power-performance efficiency**
  - Simple designs are more efficient in terms of power and area
- **Increase memory subsystem efficiency**
  - Increasing data transaction size
  - Increase number of concurrently outstanding transactions
  - ⇒ Exploit larger fraction of chip I/O bandwidth
- **Use CMOS density scaling**
  - Exploit density instead of frequency scaling to deliver increased aggregate performance
- **Use compilers to extract parallelism from application**
  - Exploit application parallelism to translate aggregate performance to application performance

Exploit application-level parallelism

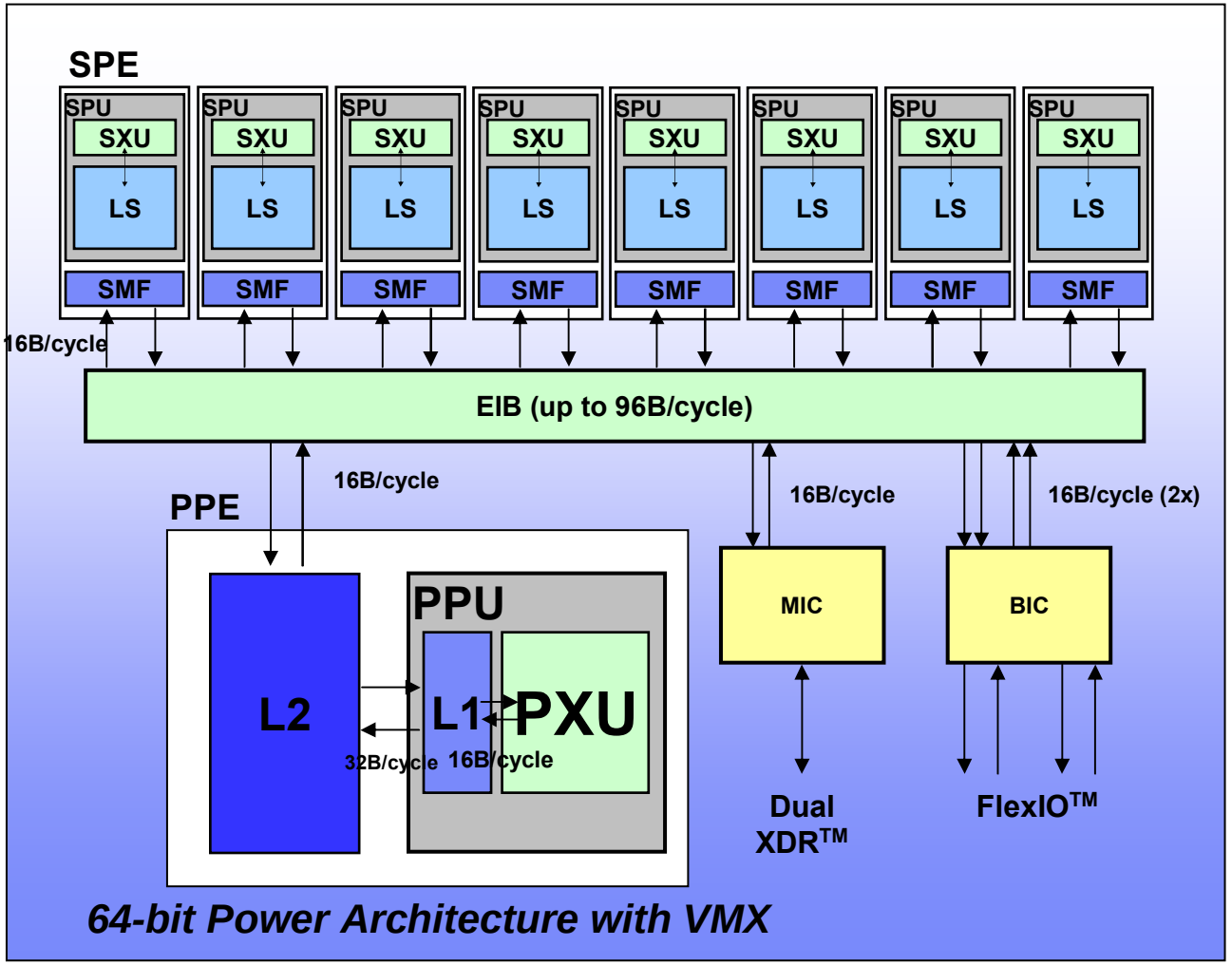
- **Data-level parallelism**
  - SIMD parallelism improves performance with little overhead
- **Thread-level parallelism**
  - Exploit application threads with multi-core design approach
- **Memory-level parallelism**
  - Improve memory access efficiency by increasing number of parallel memory transactions
- **Compute-transfer parallelism**
  - Transfer data in parallel to execution by exploiting application knowledge

Concept phase ideas



Cell Architecture

- **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)

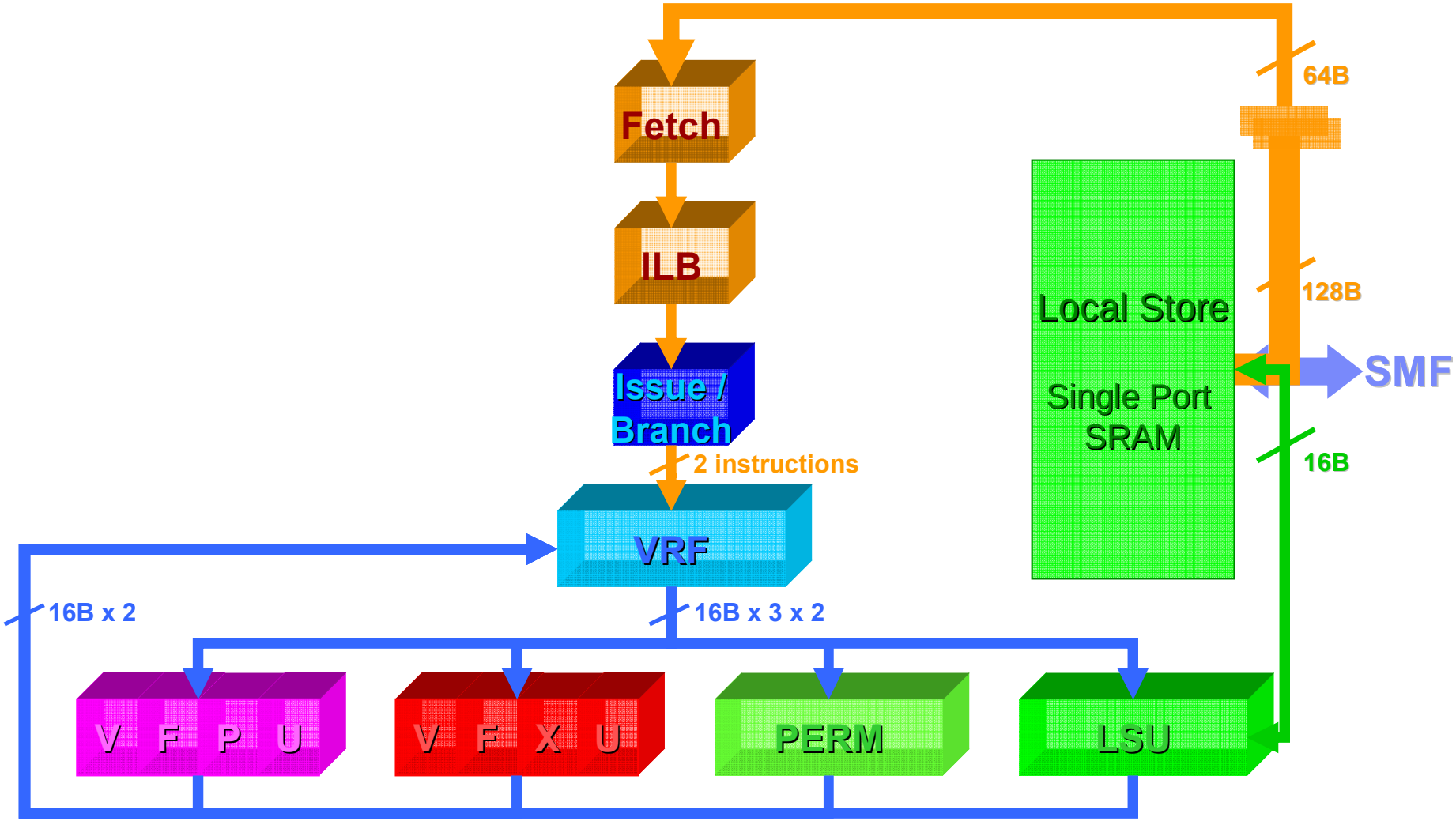


Shifting the Balance of Power with Cell Broadband Engine

- **Data processor instead of control system**
  - Control-centric code stable over time
  - Big growth in data processing needs
    - Modeling
    - Games
    - Digital media
    - Scientific applications

- **Today's architectures are built on a 40 year old data model**
  - Efficiency as defined in 1964
  - Big overhead per data operation
  - Parallelism added as an after-thought

Powering Cell – the Synergistic Processor Unit



Source: Gschwind et al., Hot Chips 17, 2005

Density Computing in SPEs

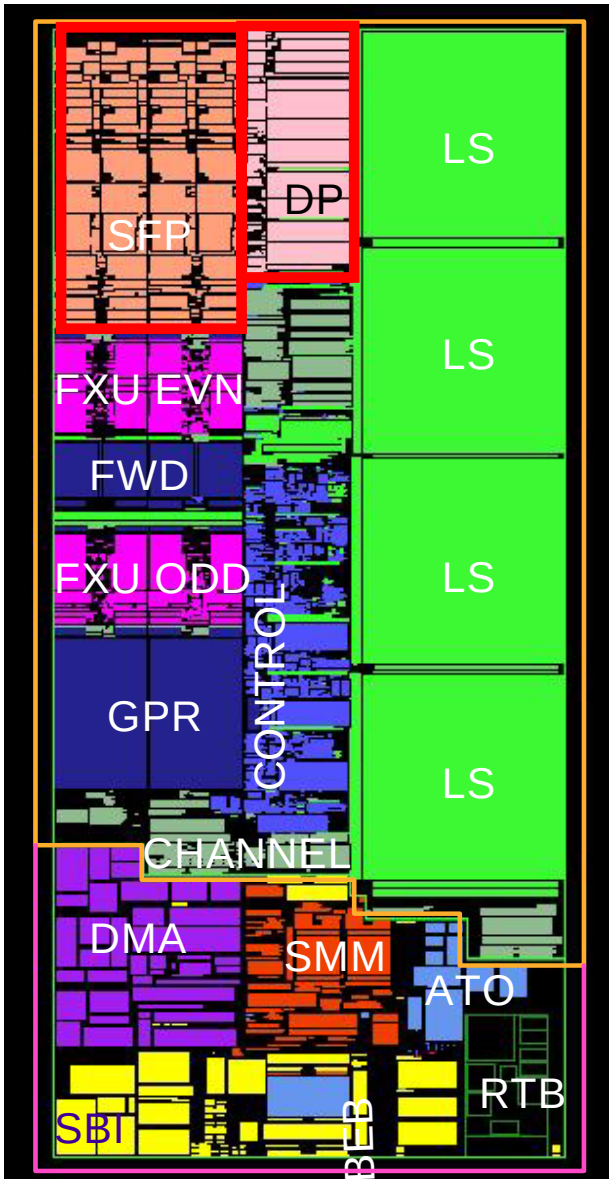
- **Today, execution units only fraction of core area and power**
  - Bigger fraction goes to other functions
    - Address translation and privilege levels
    - Instruction reordering
    - Register renaming
    - Cache hierarchy
- **Cell changes this ratio to increase performance per area and power**
  - 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

SPE Highlights

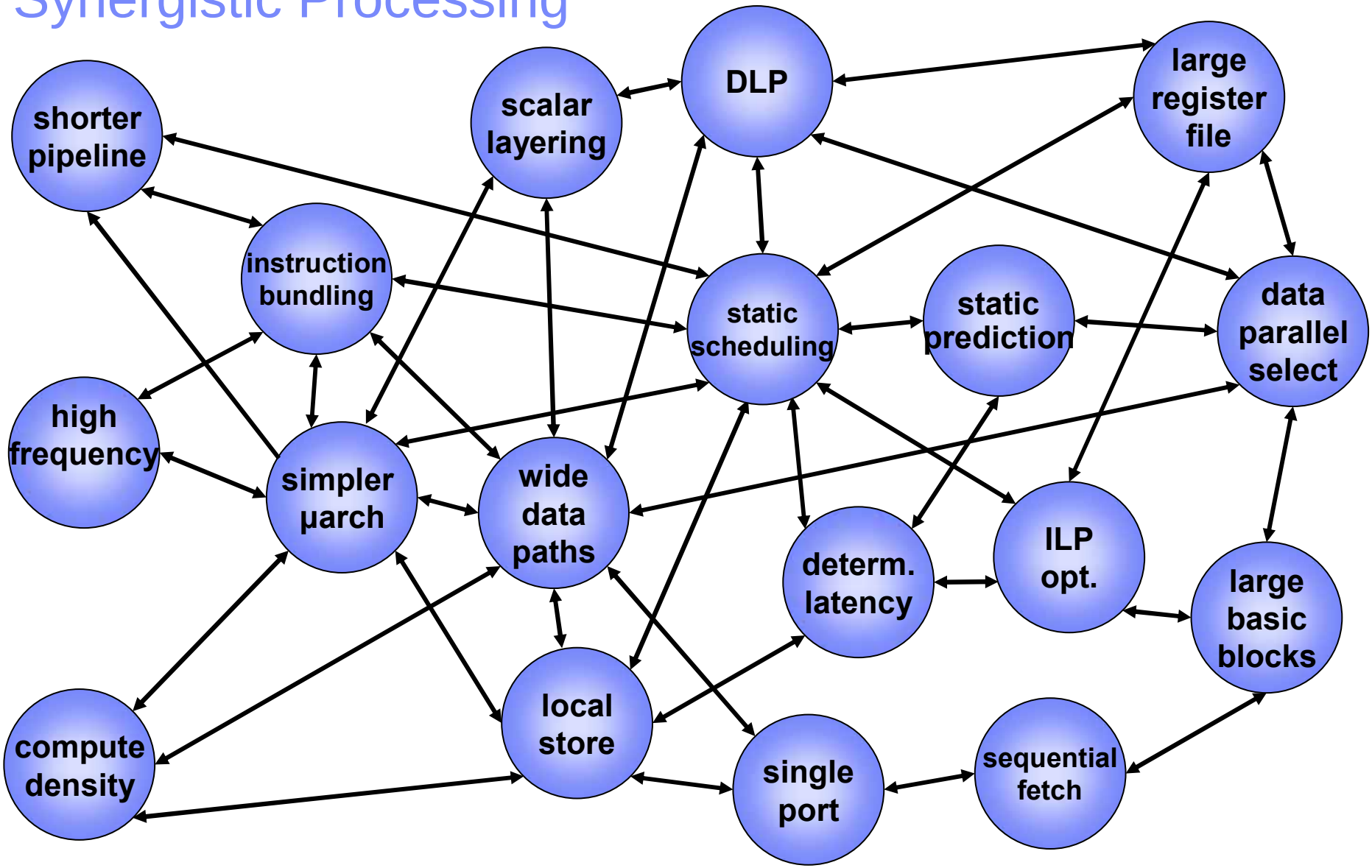
- **RISC organization**
  - 32 bit fixed instructions
  - Load/store architecture
  - Unified register file
- **User-mode architecture**
  - No page translation within SPU
- **SIMD dataflow**
  - Broad set of operations (8, 16, 32, 64 Bit)
  - Graphics SP-Float
  - IEEE DP-Float
- **256KB Local Store**
  - Combined I & D
- **DMA block transfer**
  - using Power Architecture memory translation



14.5mm² (90nm SOI)

Source: Kahle, Spring Processor Forum 2005

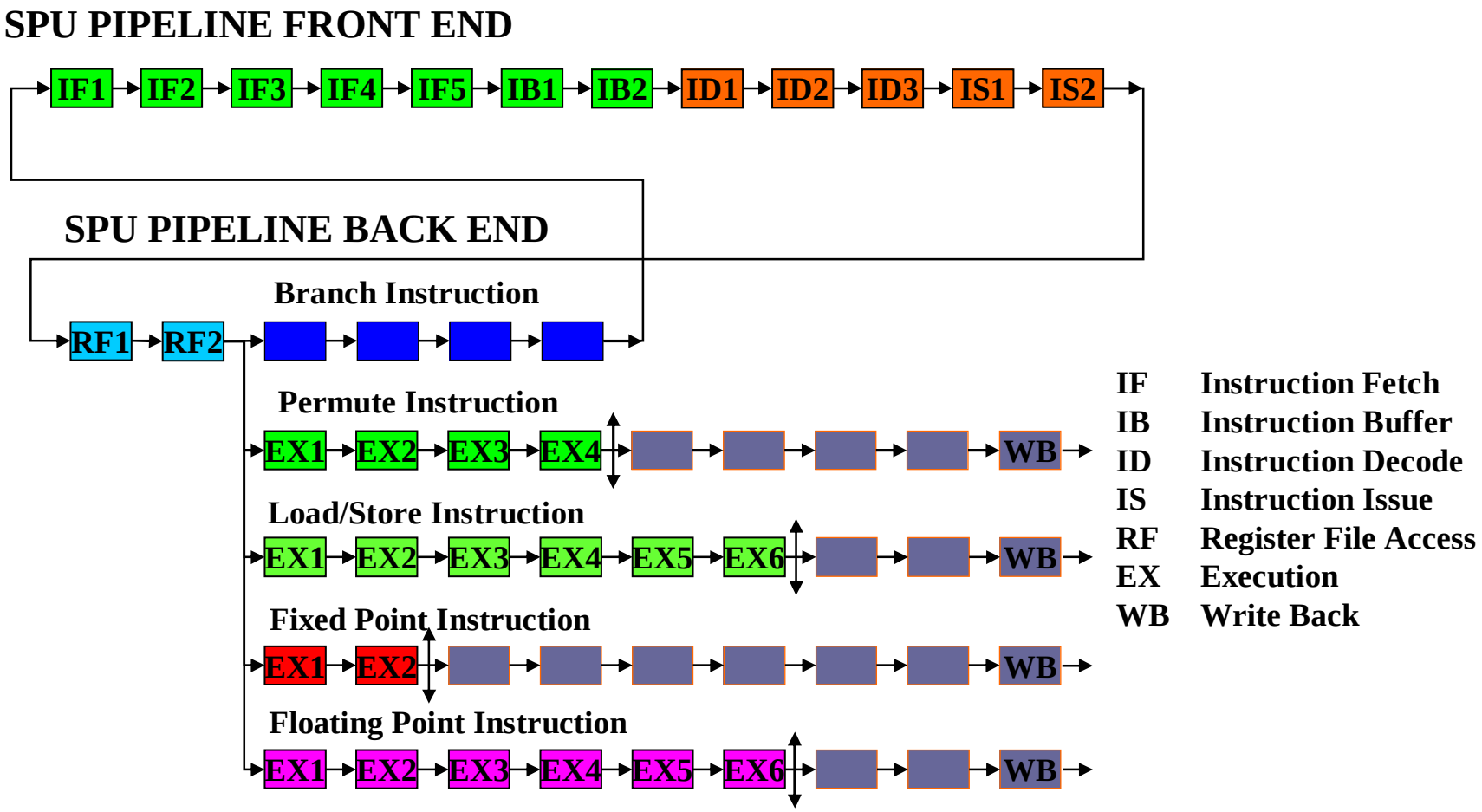
Synergistic Processing



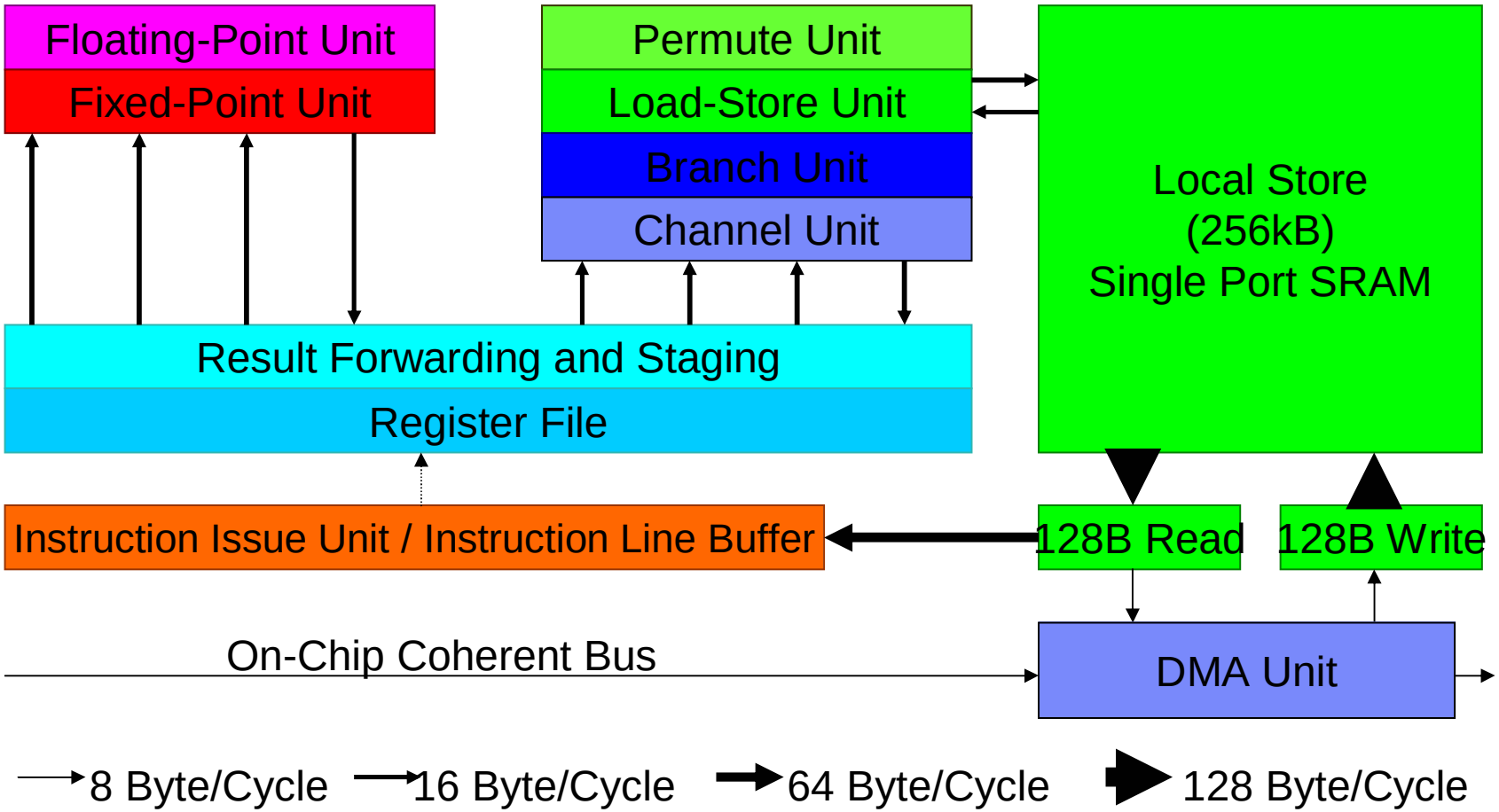
Efficient data-sharing between scalar and SIMD processing

- **Legacy architectures separate scalar and SIMD processing**
  - Data sharing between SIMD and scalar processing units expensive
    - Transfer penalty between register files
  - Defeats data-parallel performance improvement in many scenarios
- **Unified register file facilitates data sharing for efficient exploitation of data parallelism**
  - Allow exploitation of data parallelism without data transfer penalty
    - Data-parallelism *always* an improvement

SPU Pipeline

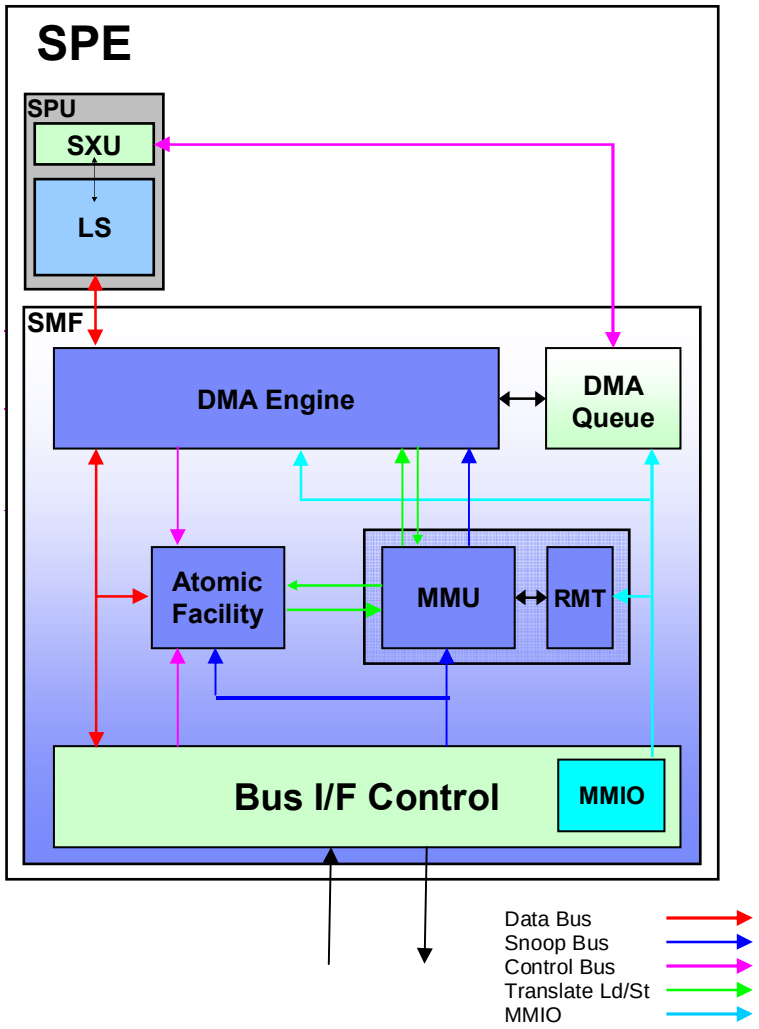


SPE Block Diagram



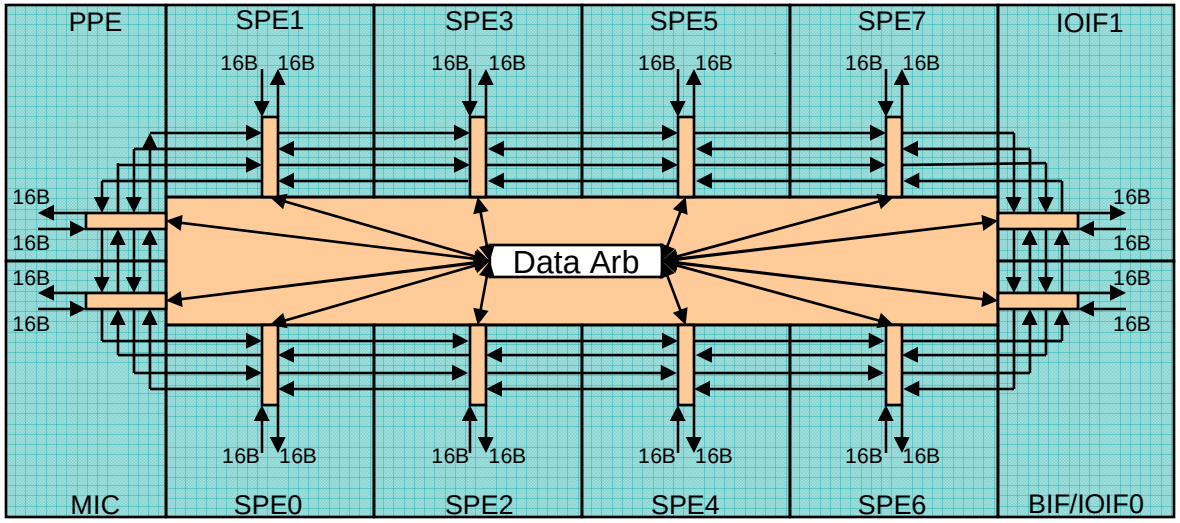
Synergistic Memory Flow Control

- **SMF implements memory management and mapping**
- **SMF operates in parallel to SPU**
  - Independent compute and transfer
  - Command interface from SPU
    - DMA queue decouples SMF & SPU
    - MMIO-interface for remote nodes
- **Block transfer between system memory and local store**
- **SPE programs reference system memory using user-level *effective address space***
  - Ease of data sharing
  - Local store to local store transfers
  - Protection



Synergistic Memory Flow Control

- **Bus Interface**
  - Up to 16 outstanding DMA requests
  - Requests up to 16KByte
  - Token-based Bus Access Management
- **Element Interconnect Bus**
  - Four 16 byte data rings
  - Multiple concurrent transfers
  - 96B/cycle peak bandwidth
  - Over 100 outstanding requests
  - 200+ GByte/s @ 3.2+ GHz



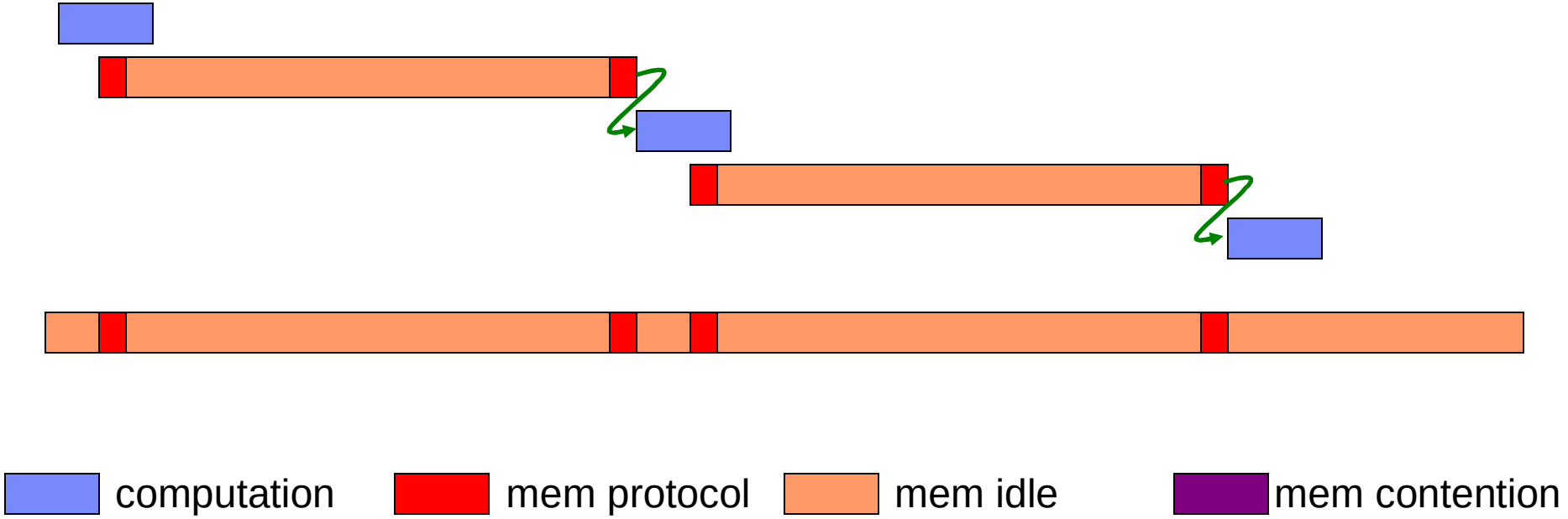
Source: Clark et al., Hot Chips 17, 2005

Memory efficiency as key to application performance

- **Greatest challenge in translating peak into app performance**
  - Peak Flops useless without way to feed data
- **Cache miss provides too little data too late**
  - Inefficient for streaming / bulk data processing
  - Initiates transfer when transfer results are already needed
  - Application-controlled data fetch avoids not-on-time data delivery
- **SMF is a better way to look at an old problem**
  - Fetch data blocks based on algorithm
  - Blocks reflect application data structures

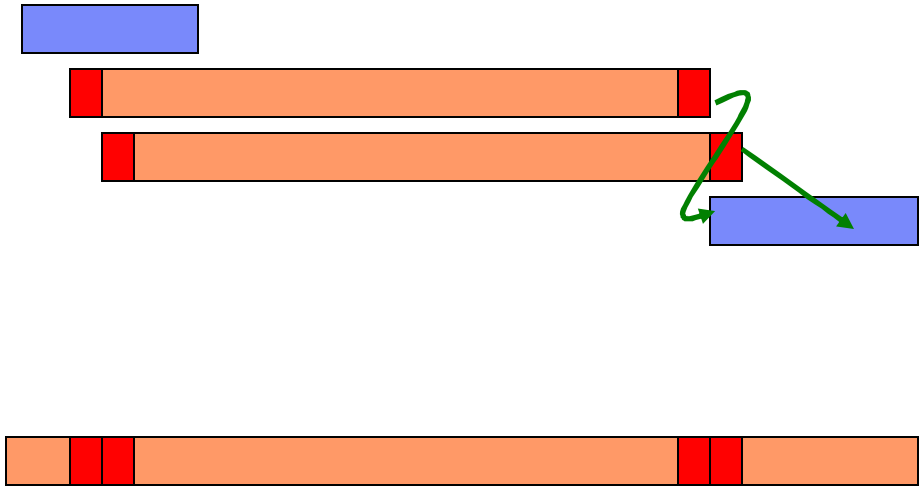
Traditional memory usage

- **Long latency memory access operations exposed**
- **Cannot overlap 500+ cycles with computation**
- **Memory access latency severely impacts application performance**



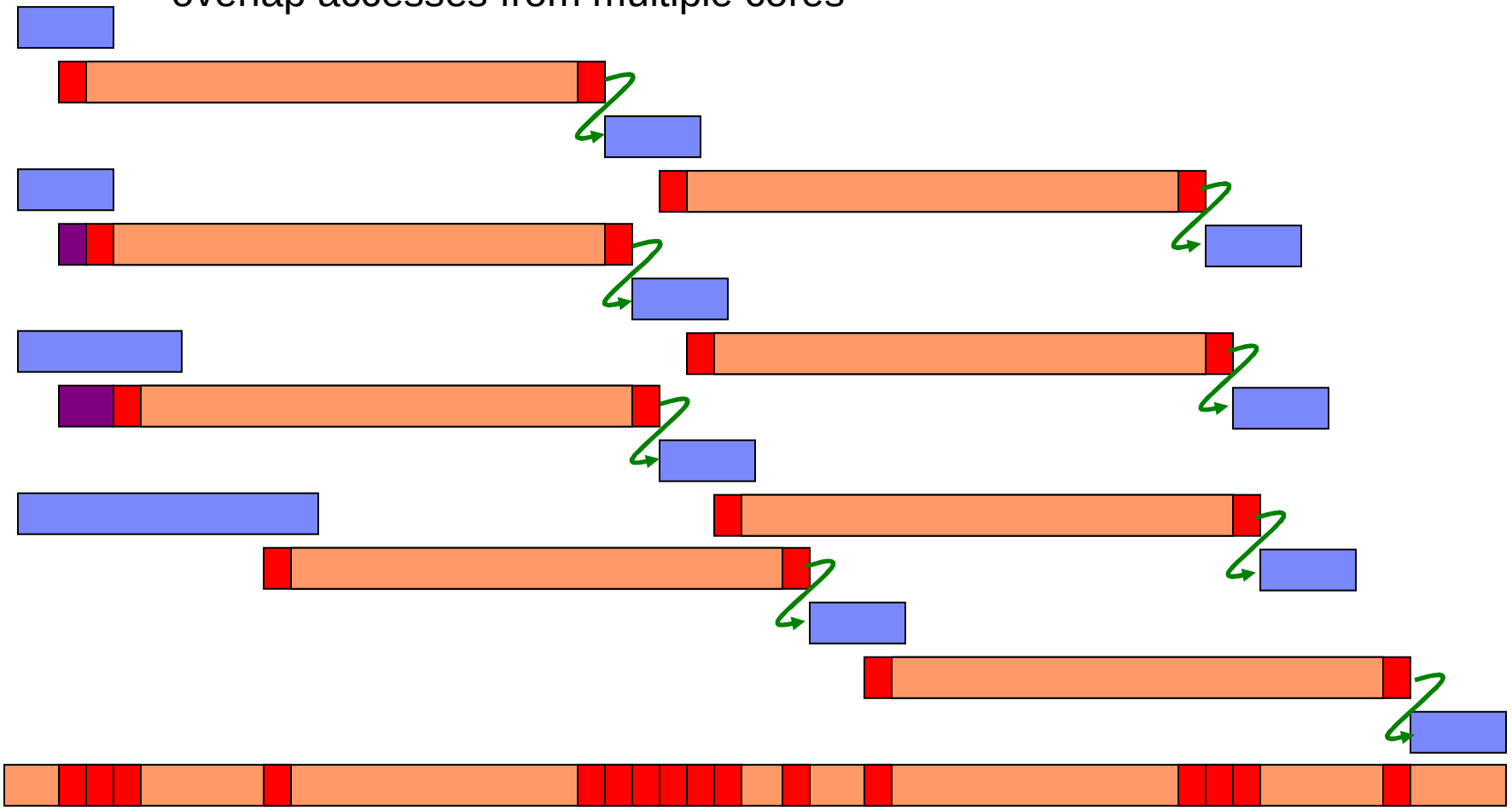
Exploiting memory-level parallelism

- **Reduce performance impact of memory accesses with concurrent access**
- **Carefully scheduled memory accesses in numeric code**
- **Out-of-order execution increases chance to discover more concurrent accesses**
  - Overlapping 500 cycle latency with computation using OoO illusory
- **Bandwidth limited by queue size and roundtrip latency**



Exploiting MLP with Chip Multiprocessing

- **More threads ➔ more memory level parallelism**
  - overlap accesses from multiple cores

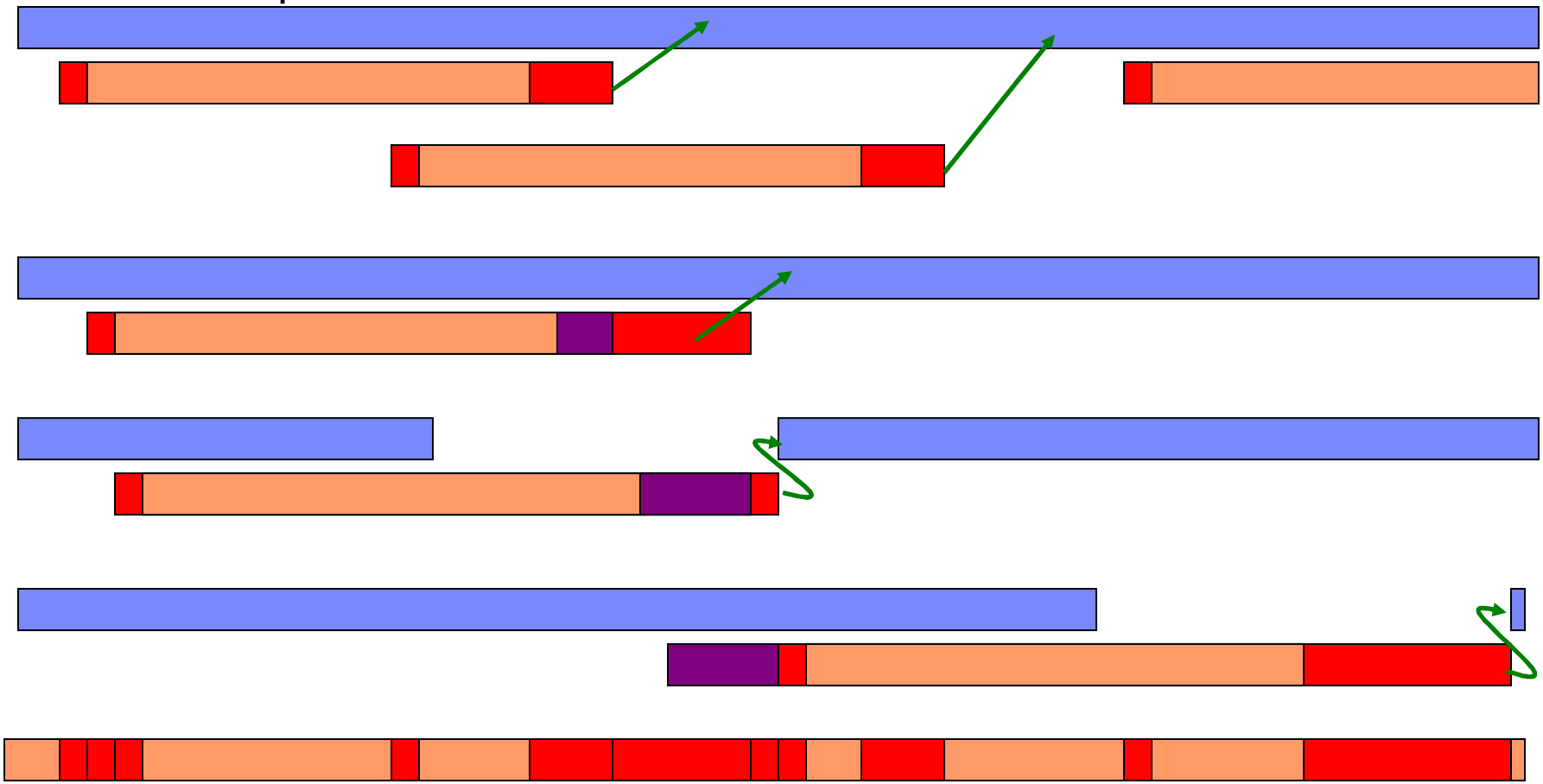


A new form of parallelism: CTP

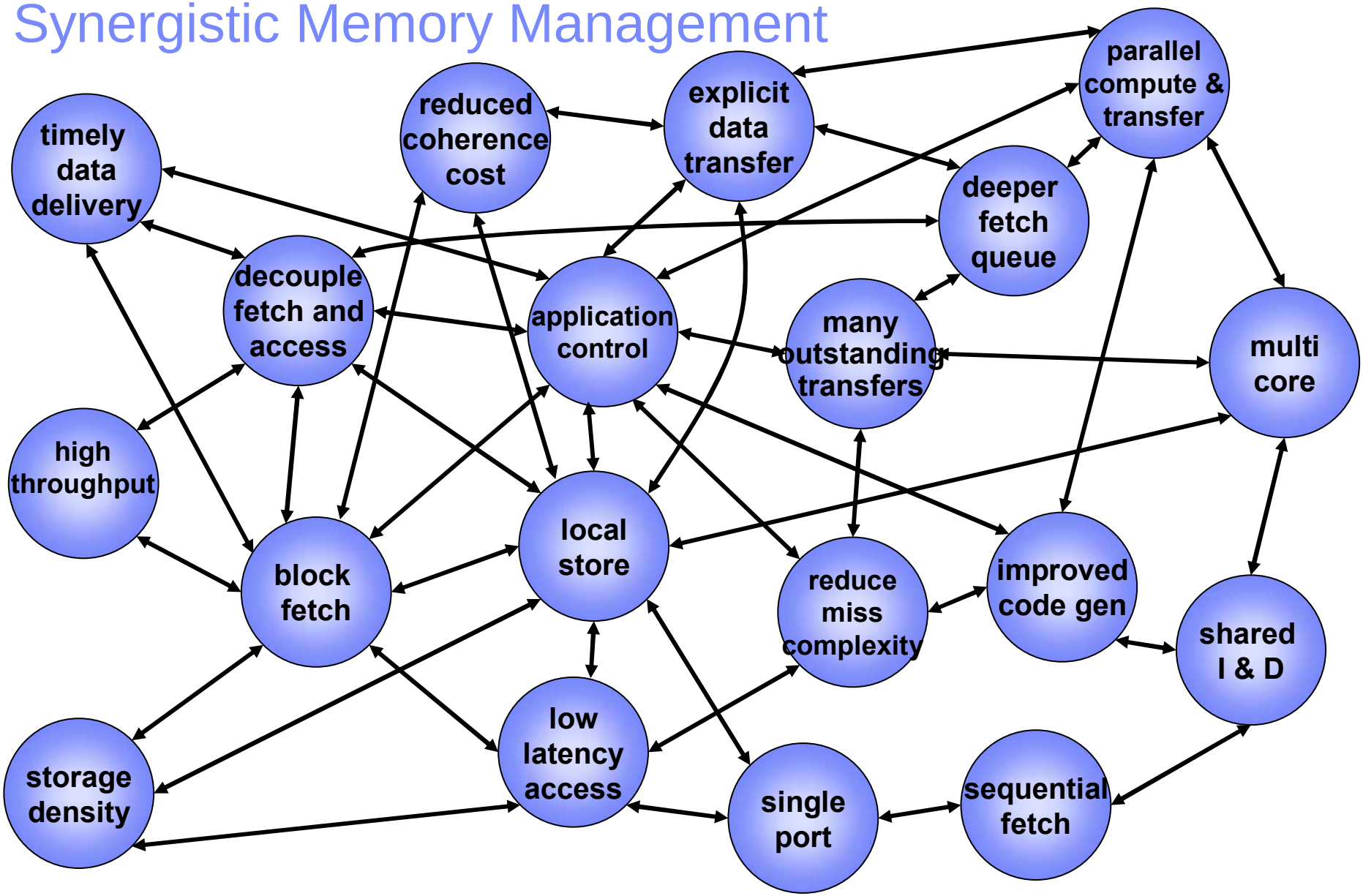
- **Compute-transfer parallelism**
  - Concurrent execution of compute and transfer increases efficiency
    - Avoid costly and unnecessary serialization
  - Application thread has two threads of control
    - SPU ➔ computation thread
    - SMF ➔ transfer thread
- **Optimize memory access and data transfer at the application level**
  - Exploit programmer and application knowledge about data access patterns

Memory access in Cell with MLP and CTP

- **Super-linear performance gains observed**
  - decouple data fetch and use

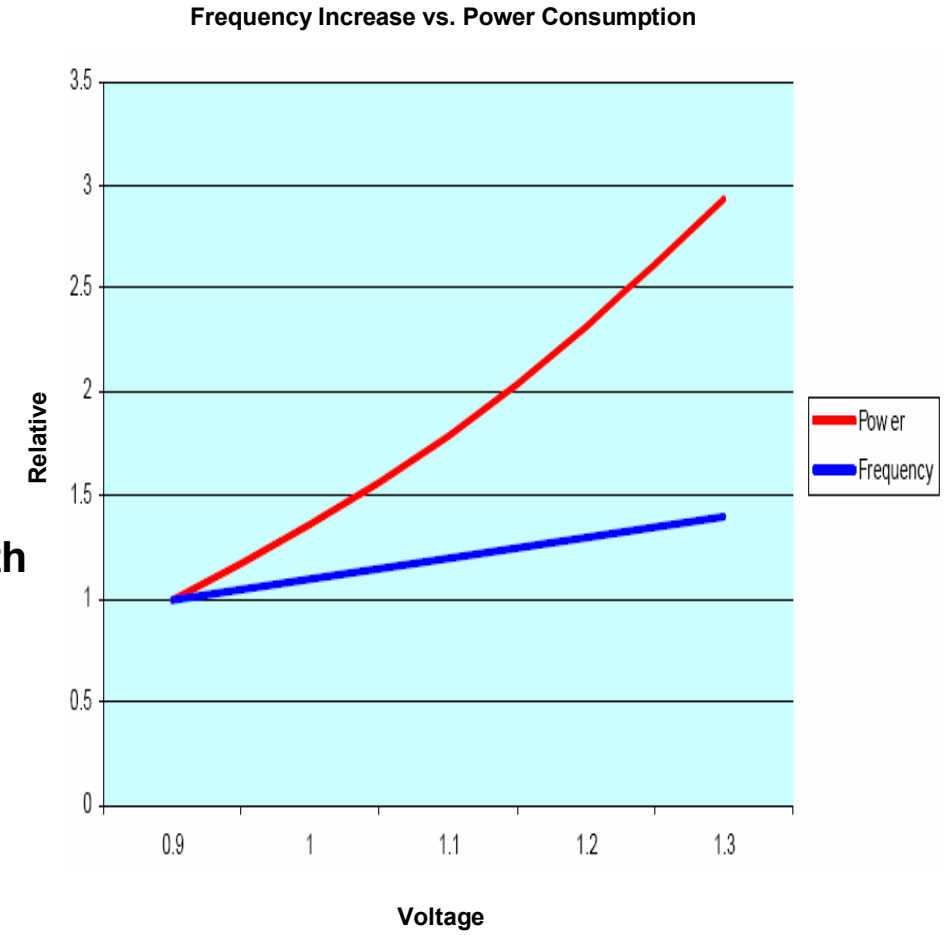


Synergistic Memory Management



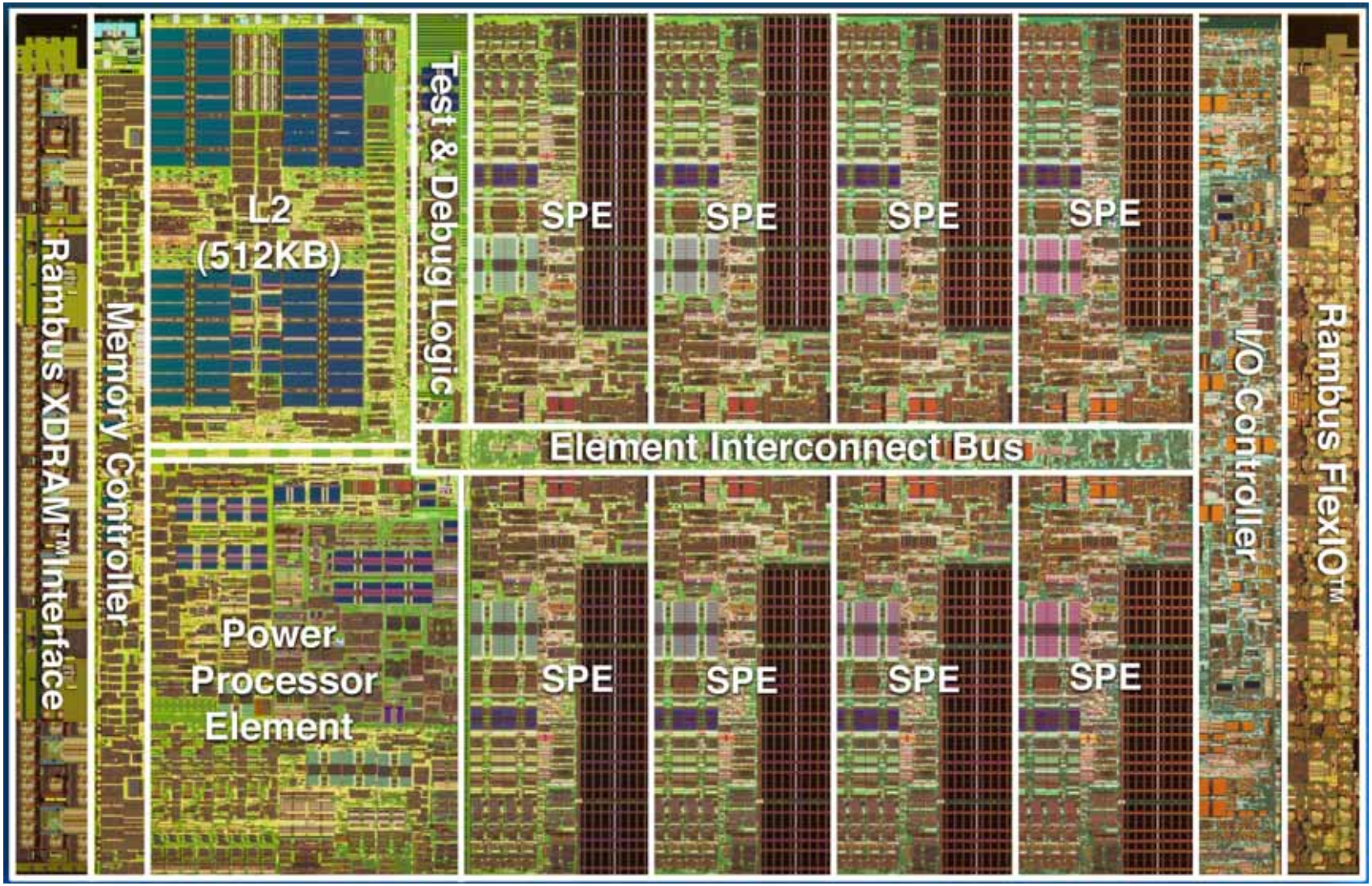
Cell BE Implementation Characteristics

- **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



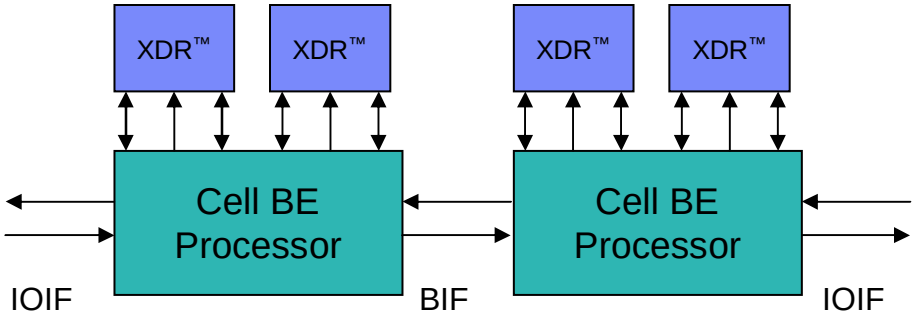
Source: Kahle, Spring Processor Forum 2005

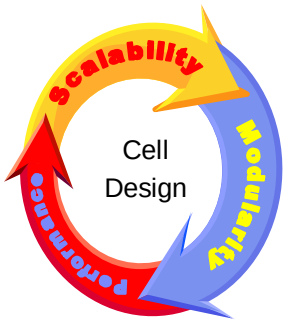
Cell Broadband Engine

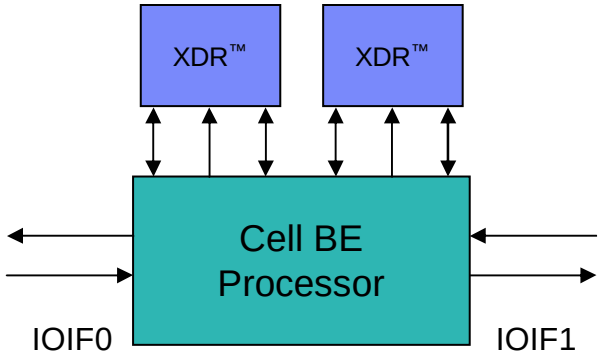


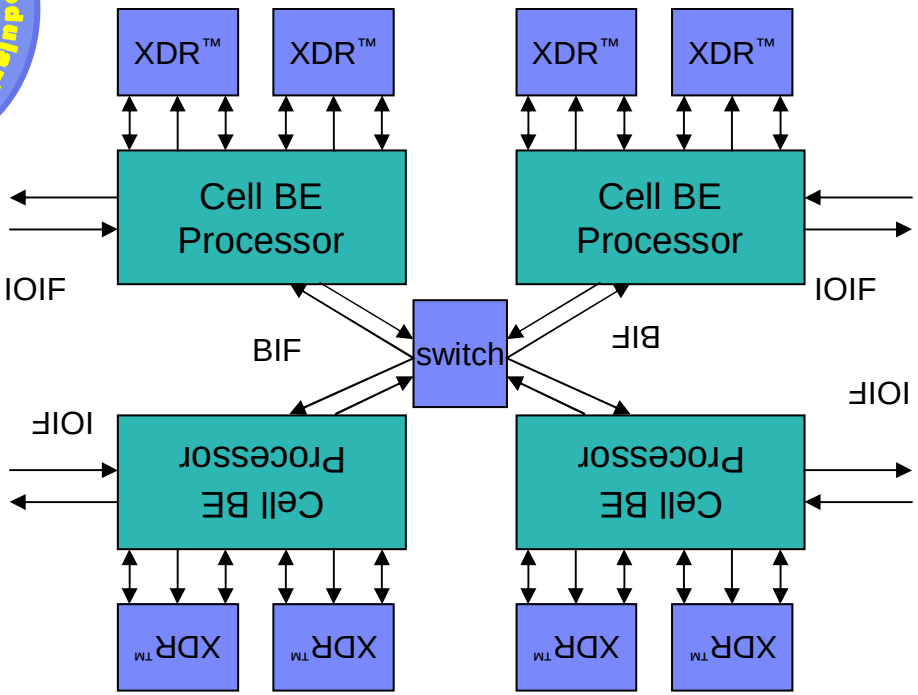
System configurations

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



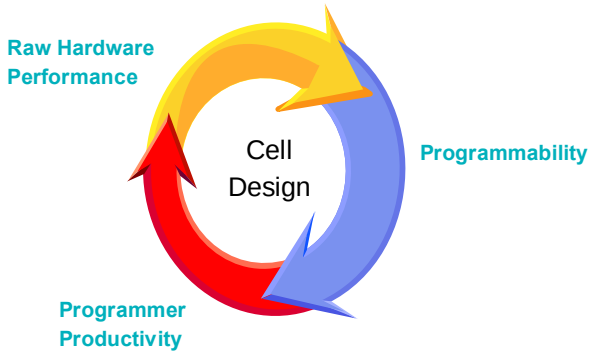






Compiling for the Cell Broadband Engine

- **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

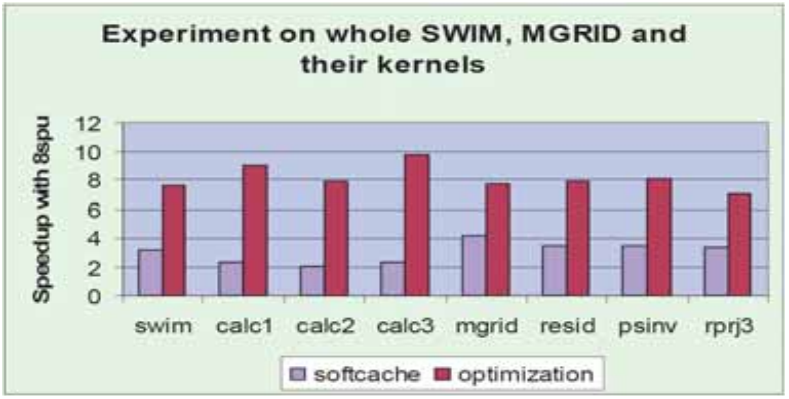


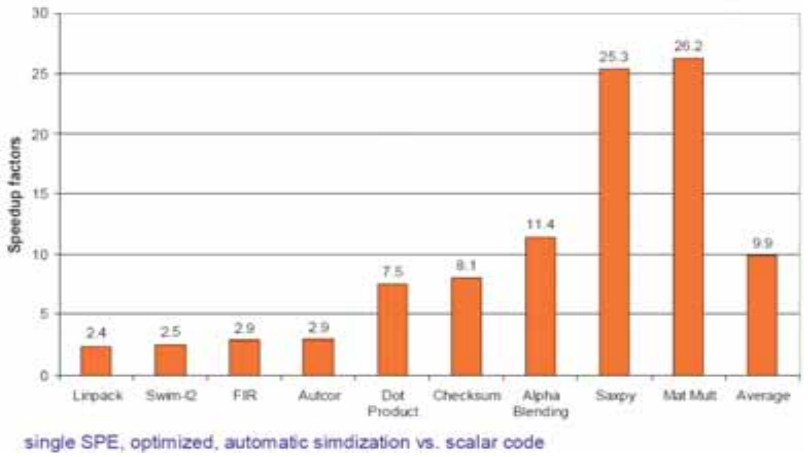
Single source automatic parallelization

- **Single source compiler**
  - Address legacy code and programmer productivity
  - Partition a single source program into SPE and PPE functions
- **Automatic parallelization**
  - Across multiple threads
  - Vectorization to exploit SIMD units
- **Compiler managed local store**
  - Data and code blocking and tiling
  - “Software cache” managed by compiler
- **Range of programmer input**
  - Fully automatic
  - Programmer-guided
    - pragmas, intrinsics, OpenMP directives
- **Prototype developed in IBM Research**

Compiler-driven performance

- **Code partitioning**
  - Thread-level parallelism
  - Handle local store size
  - “Outlining” into SPE overlays
- **Data partitioning**
  - Data access sequencing
  - Double buffering
  - “Software cache”
- **Data parallelization**
  - SIMD vectorization
  - Handling of data with unknown alignment





Source: Eichenberger et al., PACT 2005

Raising the bar with parallelism…

- **Data-parallelism and static ILP in both core types**
  - Results in low overhead per operation
- **Multithreaded programming is key to great Cell BE performance**
  - Exploit application parallelism with 9 cores
  - Regardless of whether code exploits DLP & ILP
  - Challenge regardless of homogeneity/heterogeneity
- **Leverage parallelism between data processing and data transfer**
  - A new level of parallelism exploiting bulk data transfer
  - Simultaneous processing on SPUs and data transfer on SMFs
  - Offers superlinear gains beyond MIPS-scaling

… while understanding the trade-offs

- **Uniprocessor efficiency is actually low**
  - Gelsinger's Law captures historic (performance) efficiency
    - 1.4x performance for 2x transistors
  - Marginal uni-processor (performance) efficiency is 40% (or lower!)
    - And power efficiency is even worse
- **The "true" bar is marginal uniprocessor efficiency**
  - A multiprocessor "only" has to beat a uniprocessor to be the better solution
  - Many low-hanging fruit to be picked in multithreading applications
    - Embarrassing application parallelism which has not been exploited

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
- **Element Interconnect Bus integrates system as data transport hub**

A new wave of innovation

- **CMPs everywhere**
  - same ISA CMPs
  - different ISA CMPs
  - homogeneous CMP
  - heterogeneous CMP
  - thread-rich CMPs
  - GPUs as compute engine
- **Compilers and software must and will follow…**
- **Novel systems**
  - Power4
  - BOA
  - Piranha
  - Cyclops
  - Niagara
  - BlueGene
  - Power5
  - Xbox360
  - Cell BE

Conclusion

- **Established performance drivers are running out of steam**
- **Need new approach to build high performance system**
  - Application and system-centric optimization
  - Chip multiprocessing
  - Understand and manage memory access
  - Need compilers to orchestrate data and instruction flow
- **Need new push on understanding workload behavior**
- **Need renewed focus on compilation technology**
- **An exciting journey is ahead of us!**