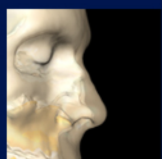
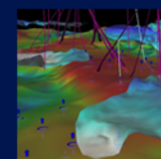
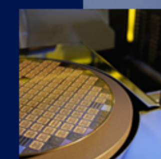
Computer Systems, Inc.
MERCURY
Challenges Drive Innovation





Performance of a Multicore Matrix Multiplication Library

High Performance Embedded Computing Conference – September 19, 2007

Frank Lauginiger, Robert Cooper, Jon Greene, Michael Pepe, Myra Prelle, Yael Steinsaltz

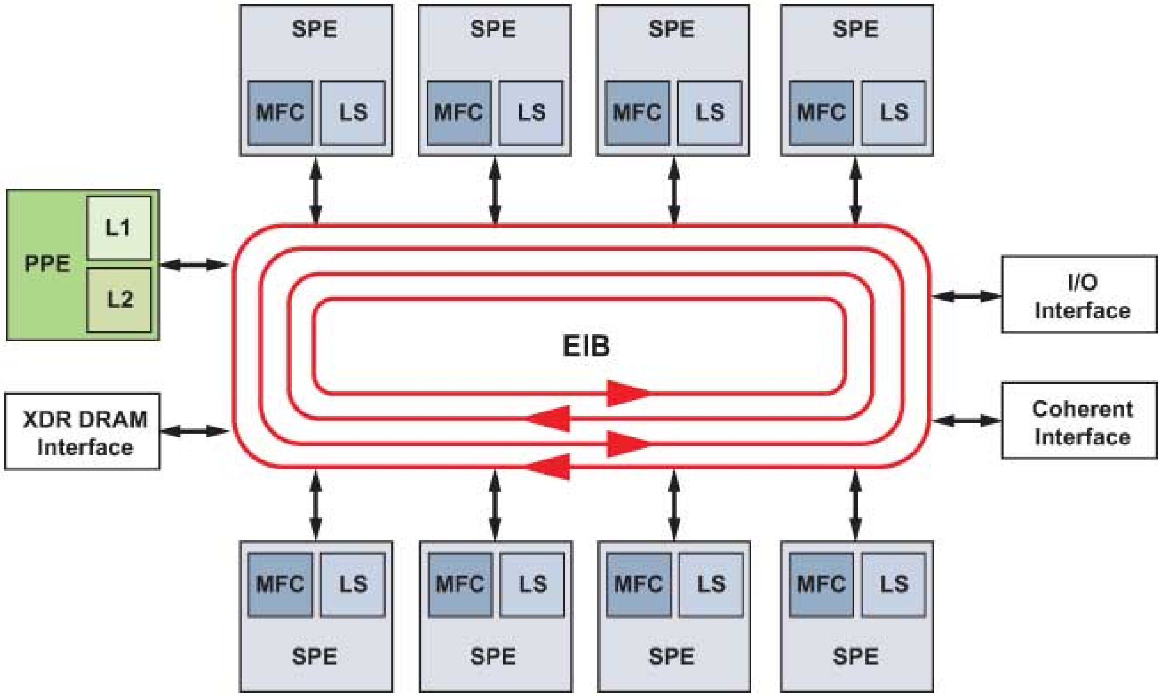
Multicore Programmers Need Help

- **Parallel programming is decades old...**
- **But parallel programming is new to most programmers (and programs)**
- **Libraries optimized across multiple cores are quickest route to performance**

Outline

- **Cell processor refresher**
- **Multicore matrix multiplication library**
  - Design and performance
- **Bonus slides on multicore FFT library performance**
- **Where a multicore offload library fits into the bigger picture of multicore programming**

Cell BE Processor Block Diagram



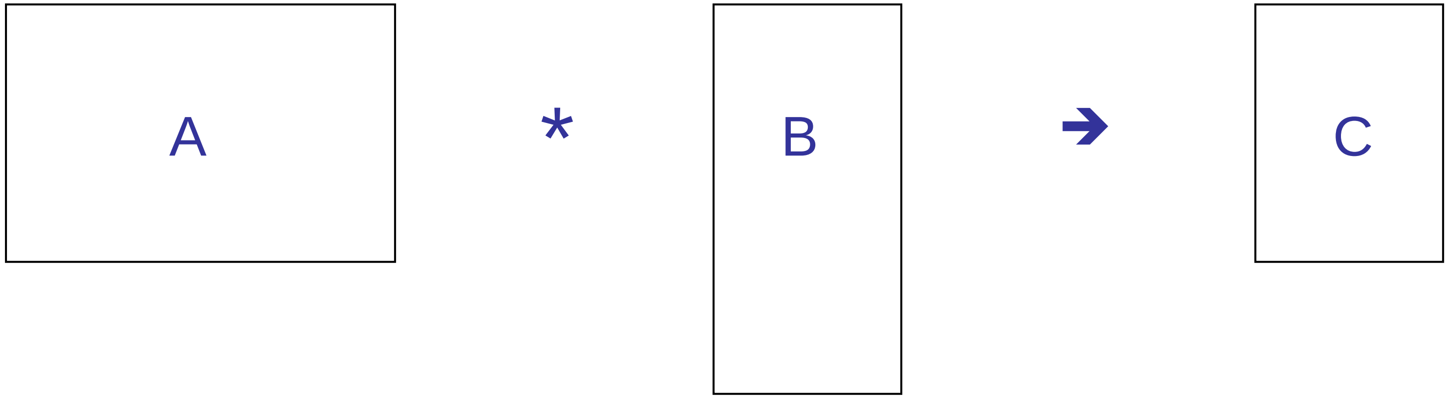
- **Cell BE processor boasts nine processors on a single die**
  - 1 Power processor
  - Up to 8 vector processors
- **Computational performance**
  - 205 GFLOPS @ 3.2 GHz
  - 410 GOPS @ 3.2 GHz
- **High-speed data ring connects everything**
  - 205 GB/s maximum sustained bandwidth
- **High-performance chip interfaces**
  - 25.6 GB/s XDR main memory bandwidth

Programming the Cell Processor

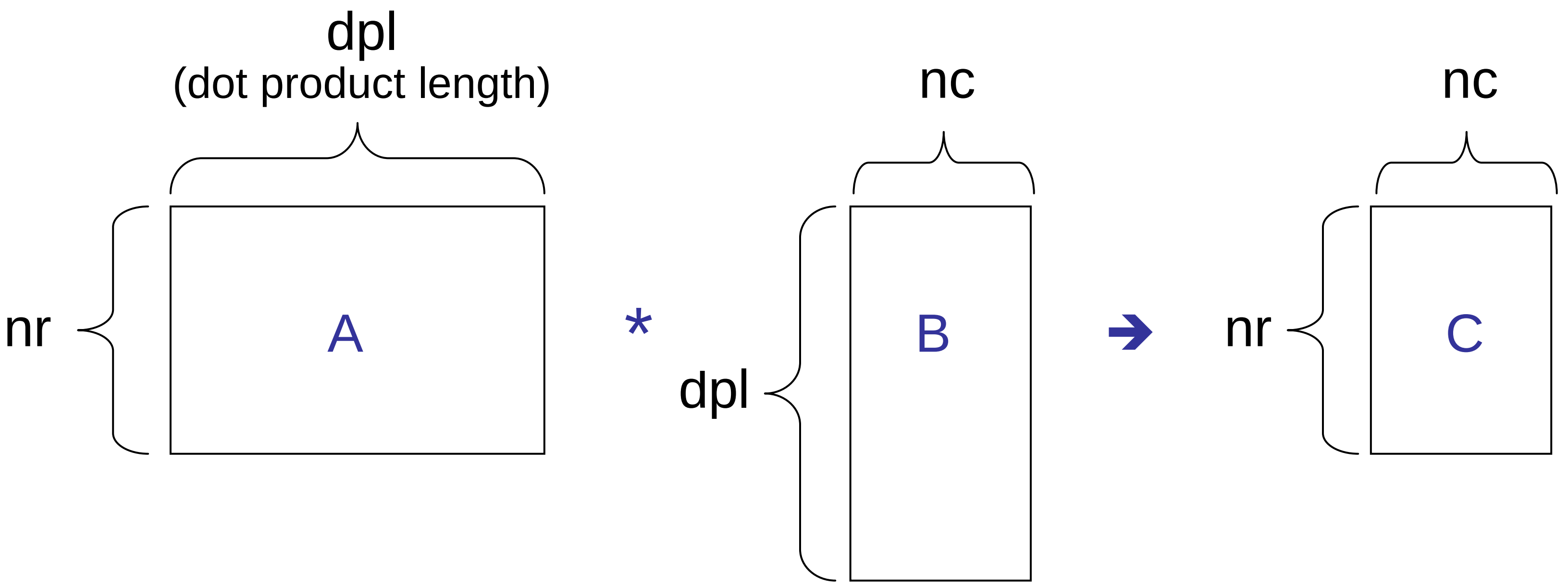
Mercury Computer Systems, Inc.
Challenges Drive Innovation

- **Easiest aspects of programming the Cell processor**
  - Very deterministic SPE performance
  - Generous ring bandwidth
  - Standards-compliant Power core
- **Getting top performance**
  - 256KB SPE local store for code and data:
    - Minimize code
    - Decompose algorithm to work on chunks that fit in local store
  - Explicit DMAs of code and data between local store and main memory
    - Performance best with 128 byte aligned data in granularity of 128 bytes
  - 128 bit vector engine:
    - Vectorize inner loops
  - Design data decomposition that:
    - Optimizes DMA alignment constraints and
    - Presents data in chunks that can be processed in parallel by vector engine

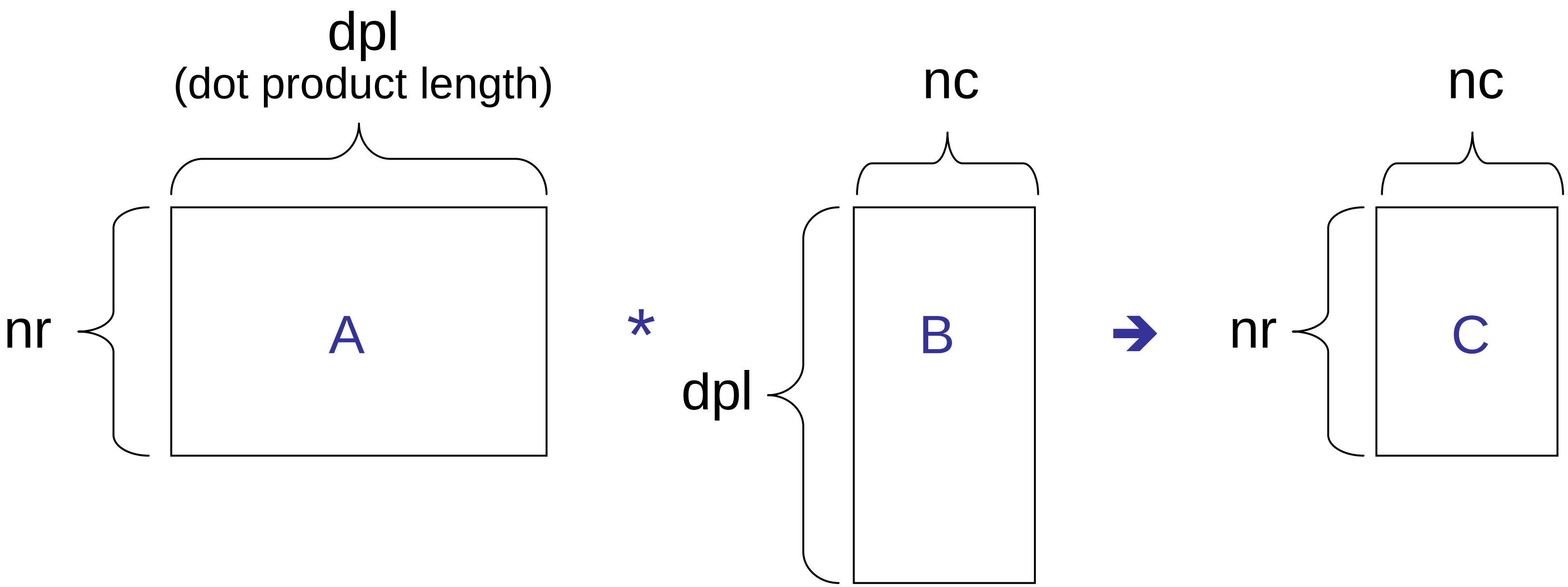
Multicore Matrix Multiplication Library



Multicore Matrix Multiplication Library



Multicore Matrix Multiplication Library



- **Supports**
  - Rectangular matrices
  - Sizes in increments of 32 row or columns
  - Optional accumulation C = C + A * B
  - Optional pre-transposition of A or B or both
  - Selectable parallelism (number of SPEs)
- **Part of MultiCore SAL (Scientific Algorithm Library)**

Matrix Multiplication Implementation

- **Different algorithm mappings for different matrix sizes**
- **Rest of talk covers sizes between 32 and 1024 rows or columns**

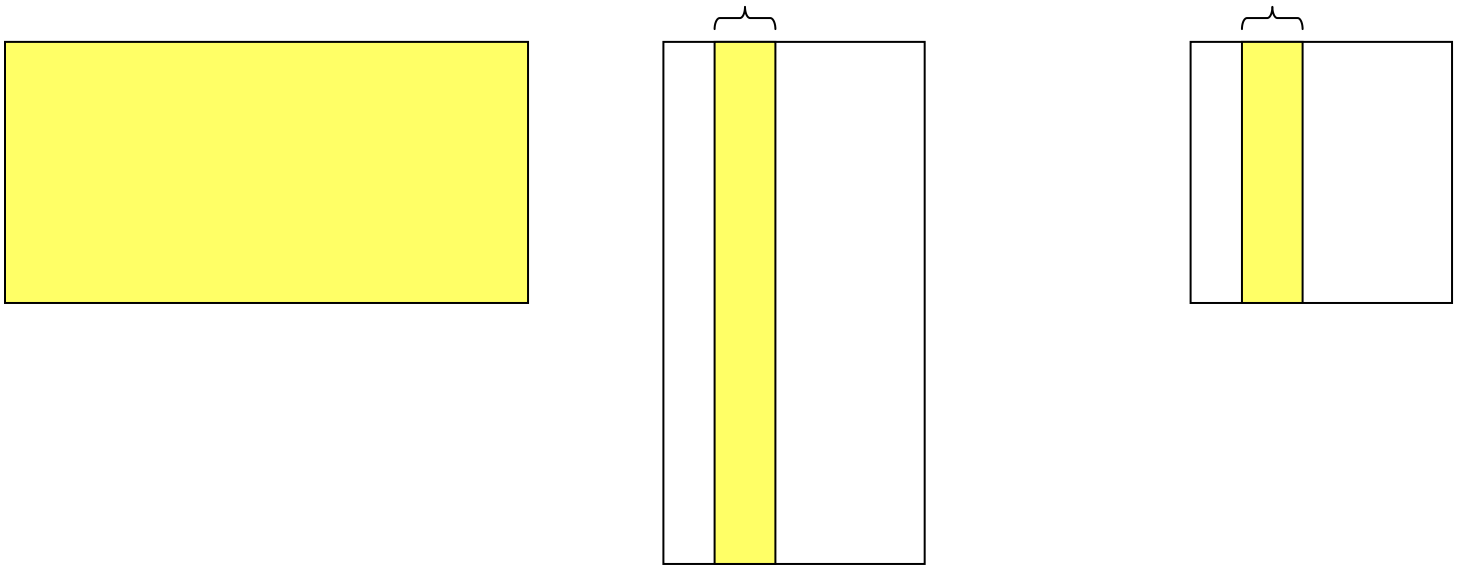
Problem Decomposition

A [nr=512,dpl=1024] **x** B [dpl=1024,nc=512] ➔ C [nr=512,nc=512]

Each SPE processes entire matrix A

Each SPE processes *nc/p* = 64 column partition of matrix B

Each SPE computes *nc/p* = 64 column partition of matrix C



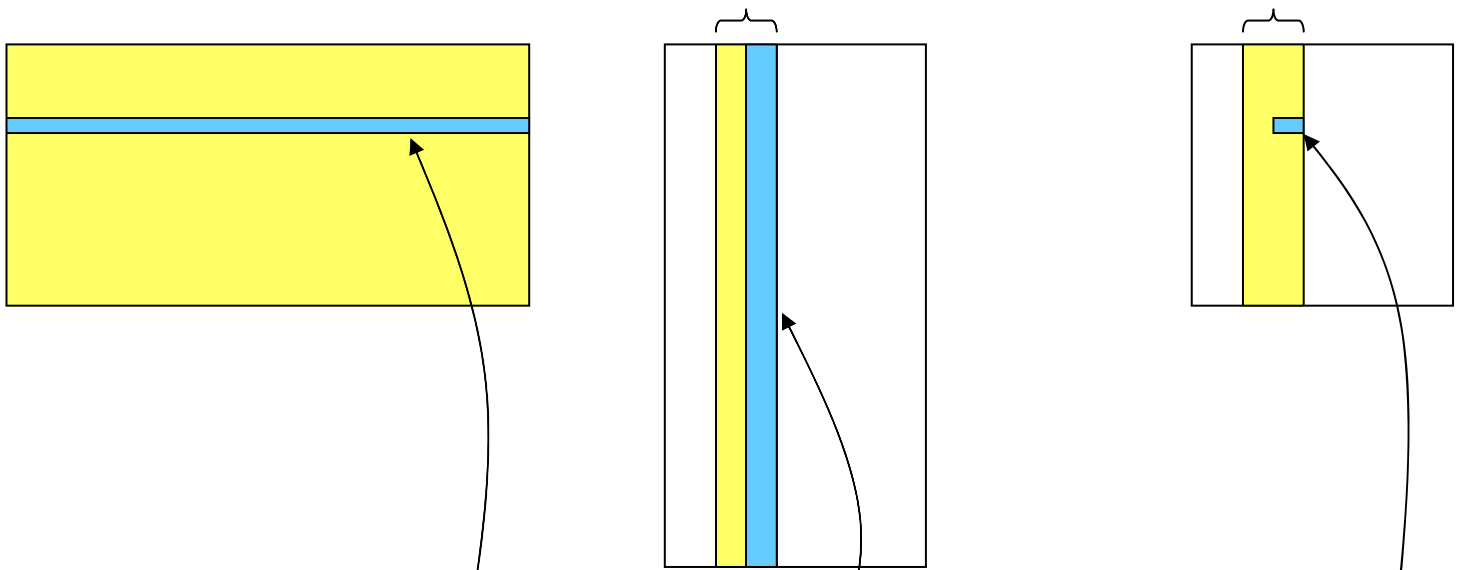
Problem Decomposition

A [nr=512,dpl=1024] **x** B [dpl=1024,nc=512] ➔ C [nr=512,nc=512]

Each SPE processes entire matrix A

Each SPE processes *nc/p* = 64 column partition of matrix B

Each SPE computes *nc/p* = 64 column partition of matrix C



Inner loop multiplies

8 x *dpl* element tile from A

with *dpl* x 32 tile from B

to produce 8 x 32 tile of C

Why These Sizes?

Computer Systems, Inc.
MERCURY
Challenges Drive Innovation

- **Chose to store entire dot product in an SPE**
  - Want to maximize dot product length for efficient inner loop
  - But also want to process enough columns at once to make strided transfers of B and C tiles efficient
    - 32 columns: 128 byte DMAs
  - Multiple columns also make vectorization easier

- **Local store usage**
  - Two A buffers: 2 * 8 * 1024 * 4 bytes/float = 64K
  - B buffer: 32 * 1024 * 4 bytes/float = 128K
  - C buffer: 8 * 32 * 4 bytes/float = 1K
  - Total: 193K

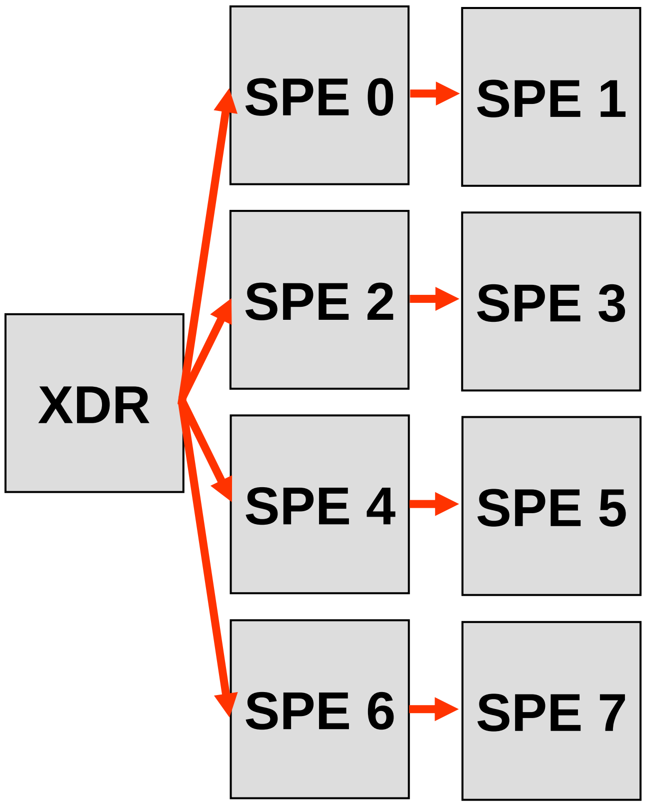
Streaming Matrix A into Local Store

Computer Systems, Inc. MERCURY Challenges Drive Innovation

- **Each SPE reads all of matrix A eight rows at a time**
  - XDR bandwidth can be the bottleneck

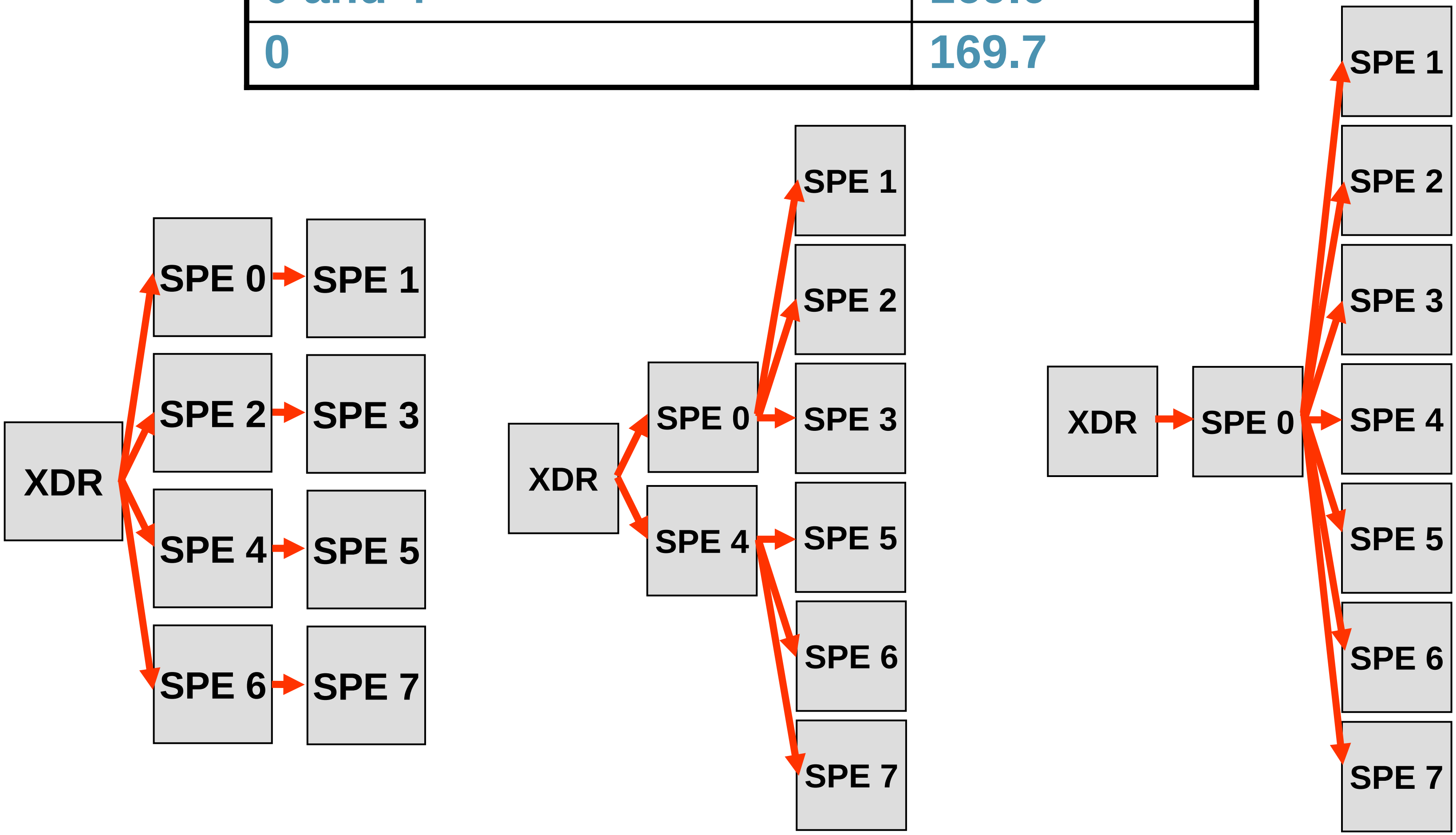
Streaming Matrix A into Local Store

- **Each SPE reads all of matrix A eight rows at a time**
  - XDR bandwidth can be the bottleneck
- **Idea: Some SPEs stream data to other SPEs**



What's the Best Streaming Strategy?

SPEs pulling A from XDR	GFLOPS
0 through 7	159.0
0, 2, 4 and 6	170.2
0 and 4	169.0
0	169.7



Matrix Multiply Library Performance

Matrix Dimensions			GFLOPS	Efficiency
nr	nc	dpl		
512	512	512	148.9	73%
512	512	1024	161.8	79%
768	768	768	162.8	79%
1024	1024	1024	170.4	83%

- **As part of a library, implementation must satisfy additional goals beside raw performance**
  - Flexibility in data size and organization
  - Options such as accumulation
  - Selectable parallelism
  - Compatible calling sequence

Performance Comparison (GFLOPS)

Matrix Dimensions			Mercury (row major)	IBM SDK (block layout)		Hackenberg (row major)
nr	nc	dpl	Reported	Measured	Reported	Reported
512	512	512	149	174*	201	70
512	512	1024	162			
768	768	768	163			125
1024	1024	1024	170	174	201	150

- **IBM SDK 2.1 matrix multiplication example**
  http://www.ibm.com/developerworks/power/library/pa-cellperf/
  - Square matrices, power of two sizes
  - Block layout only
  - * 174 GFLOPS for 512x512 achieved only for 1000 iterations of same matrix
- **Daniel Hackenberg, TU Dresden, May 2007**
  http://www.fz-juelich.de/zam/datapool/cell/Performance_Measurements_on_Cell.pdf
  - Square matrices, size increments of 64
  - Row major and block layout
  - Accumulation option

More MC-SAL Performance: Large 2D FFTs

Computer Systems, Inc. MERCURY Challenges Drive Innovation

- MC-SAL API called from PPE
- Each FFT performed in parallel on up to 8 SPEs
- Each FFT is too large to fit in the aggregate of the SPE local stores, but small enough that the row and column FFTs fit within local store

# columns \ # rows	64	128	256	512	1024
128					43
256				44	51
512			45	52	58
1024		44	52	59	63
2048	44	52	59	63	67
4096	51	55	63	67	61

MC-SAL 2D FFT performance (GFLOPS) on 8 SPEs (called from PPE, data starts and ends in XDR)

MC-SAL Performance: Streaming Small FFTs

- MC-SAL API performs a batch of 1D FFTs
- Each FFT executed on a single SPE
- Up to 8 SPEs used in parallel

N	GFLOPS
512	58
1024	65
2048	70
4096	77

Performance for 1000 FFTs using 8 SPEs (called from PPE, data starts and ends in XDR)

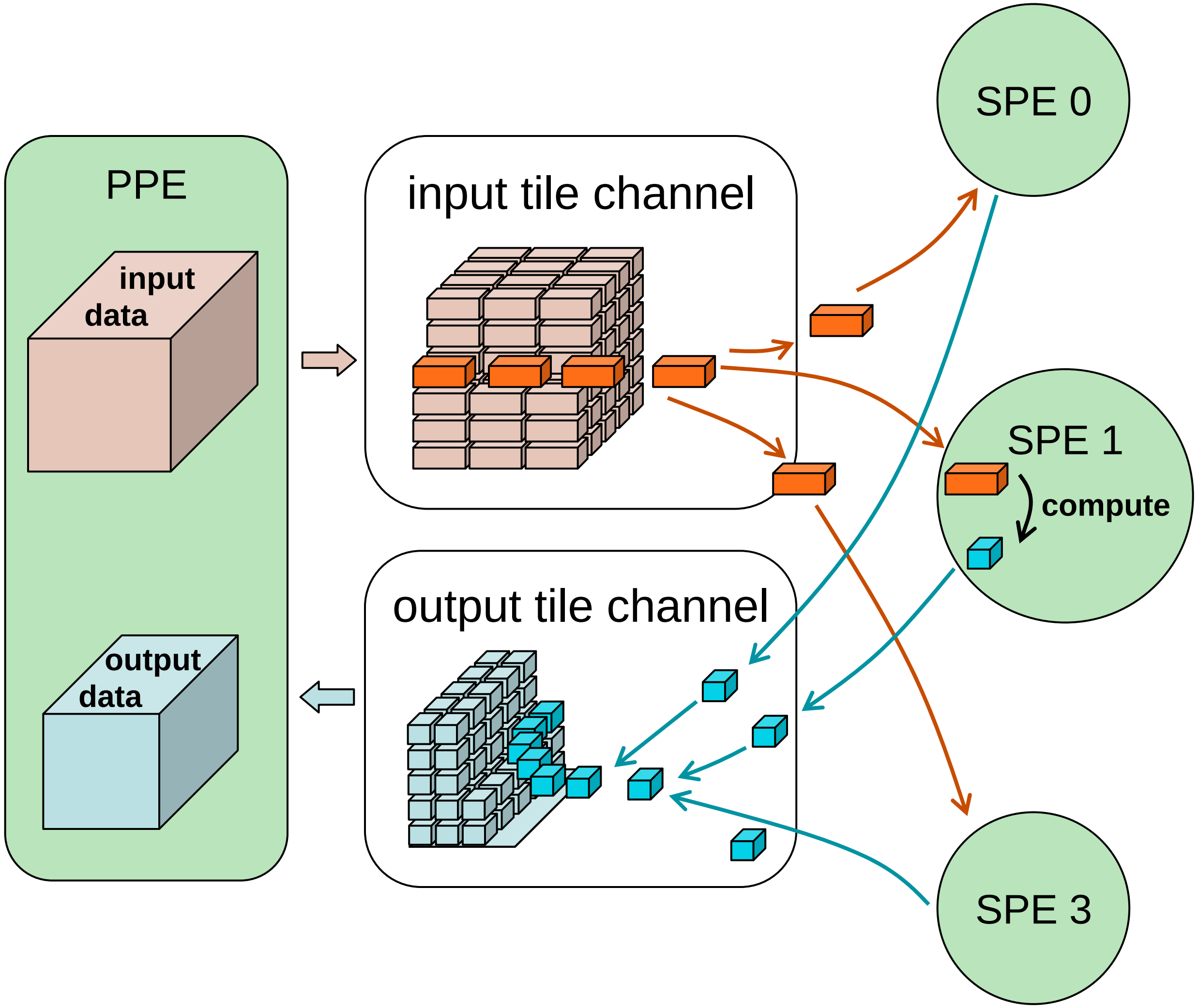
Migrating Applications to Multicore

1. **Compile/run application on general-purpose single core (Cell's PPE)**
2. **Introduce function-offload model**
    - Replace compute-intensive calls with calls to offload library (MultiCore-SAL)
3. **For further improvement, selectively develop custom offload functions to replace offload library calls**
    - E.g., fuse functions on SPE to reduce number of SPE-XDR transfers
    - Use SPE-local library (SPE-SAL) and data movement middleware (MultiCore Framework)

MultiCore Scientific Algorithm Library

- **Large FFTs, fast convolutions and matrix operations**
- **Batch operations for smaller sizes**
- **Also compatible with single-core SAL API**
- **Compatible with MultiCore Framework**
  - For explicit data movement and SPE computation
  - Example or template data-flow code provided for common algorithms
  - User can insert appropriate math (SPE-SAL)

MultiCore Framework Data Movement



Summary

- **Demonstrated superb performance for matrix multiplication on Cell processor**
- **Function offload libraries provide easiest path to good performance on multicore processors**
  - No new languages to learn
  - Also provide portability between diverse multicore architectures
- **Need ability to develop custom offload functions to extract maximum performance**