



Running Applications on the Cell Broadband Engine™

Authors:

Kerry Barnes, Bill Lundgren, Jim Steed

Gedae, Inc. (Email:
bill.lundgren@gedae.com)

Overview



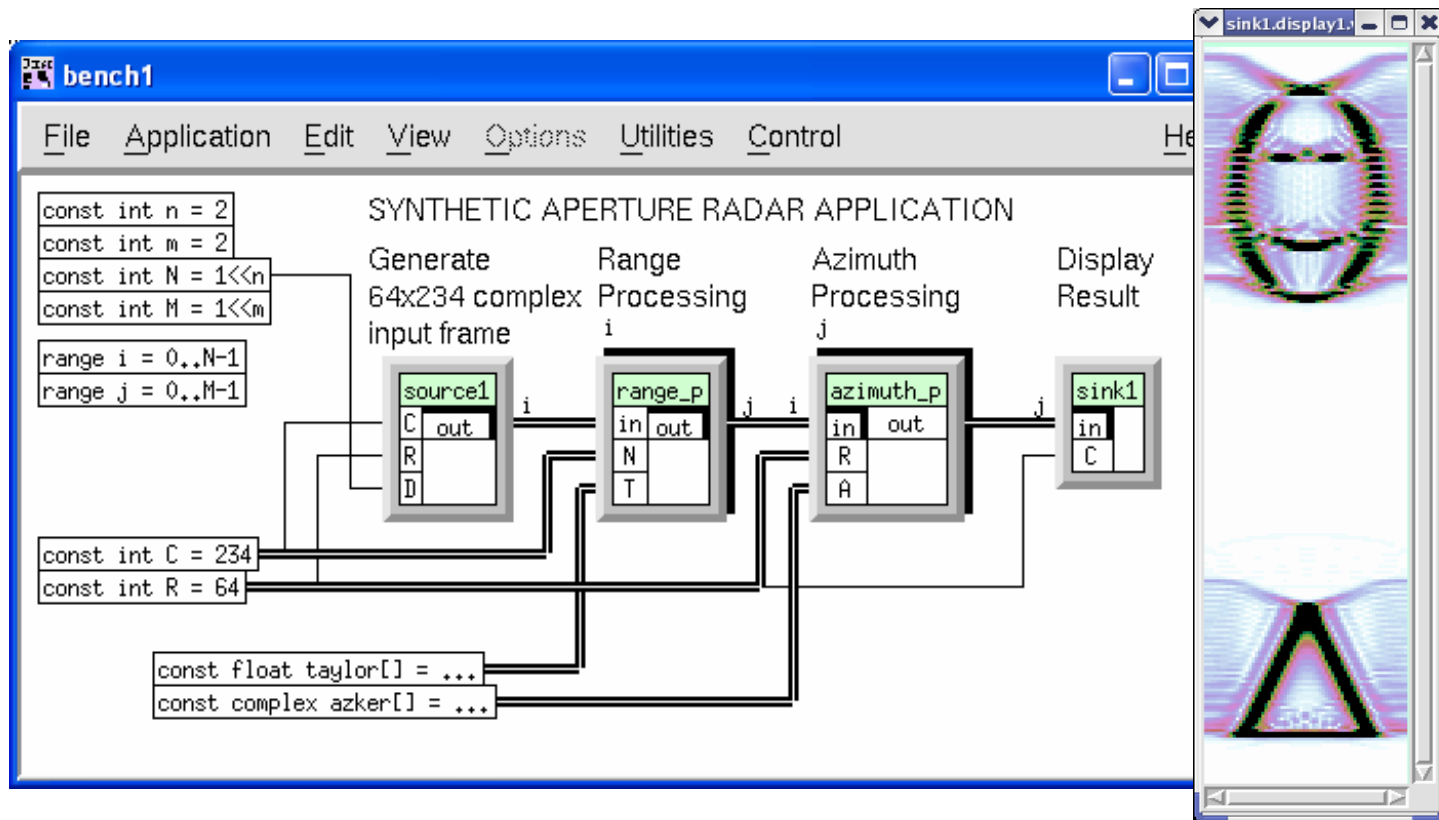
- Small SAR Benchmark – demonstration on Playstation 3
 - Fits on Cell Broadband Engine (Cell/B.E.) with “out of the box” Gedae thread scheduler
 - Range and Azimuth strip-mined on SPEs
 - Centralized transpose on PPE
 - Fits on Cell/B.E. with hand crafted distributed transpose
 - First stage of Gedae thread scheduler optimization
 - Manual memory management between local SPE memory and PPE bulk memory
 - Demonstration on Playstation 3

Overview (continued)

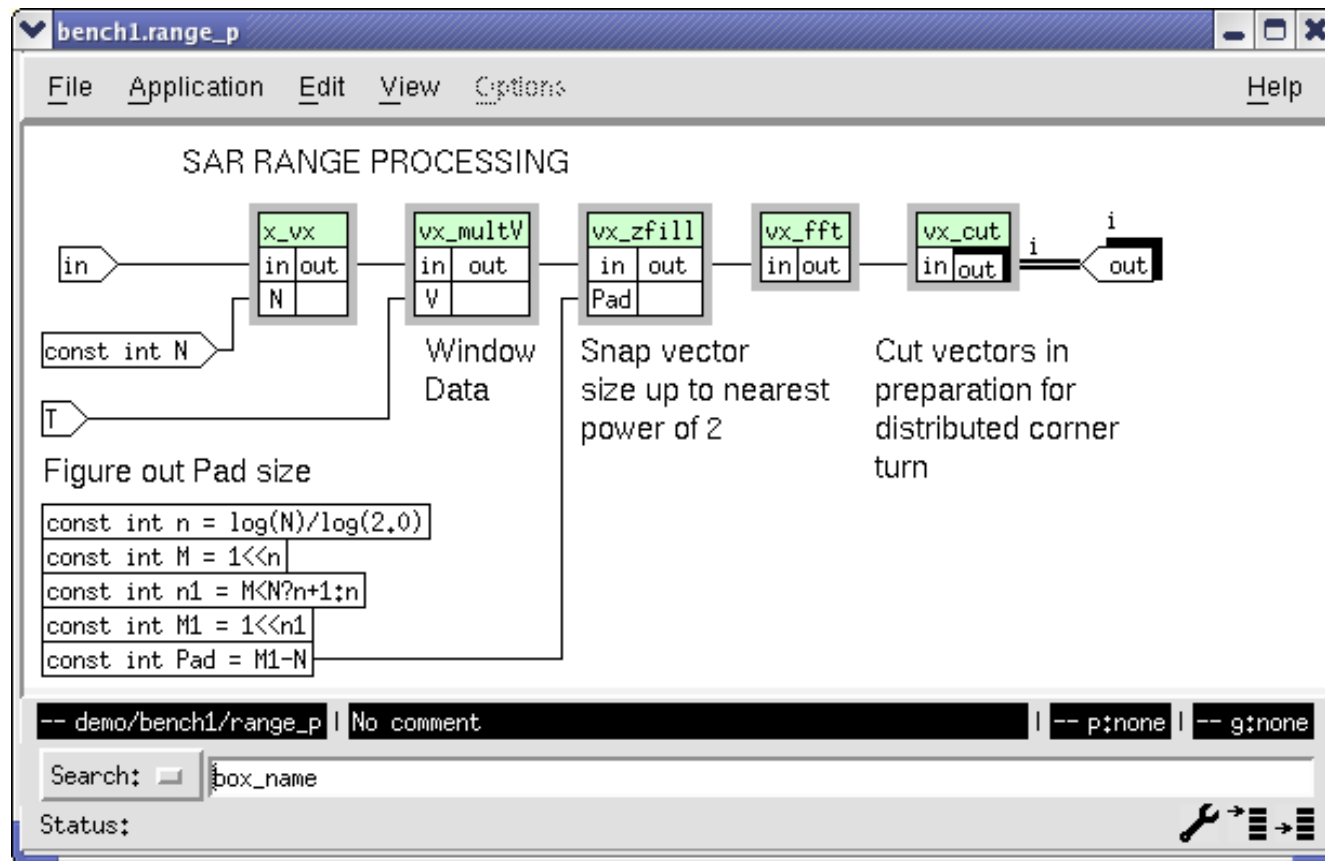


- Large SAR Benchmark
 - Demonstrates efficiency of phase 1 release of Gedae-CBE (Gedae for Cell/B.E.)
 - Fits on Cell/B.E. SPEs with optimization of thread scheduler
 - Uses hand crafted distributed transpose
- Automation roadmap

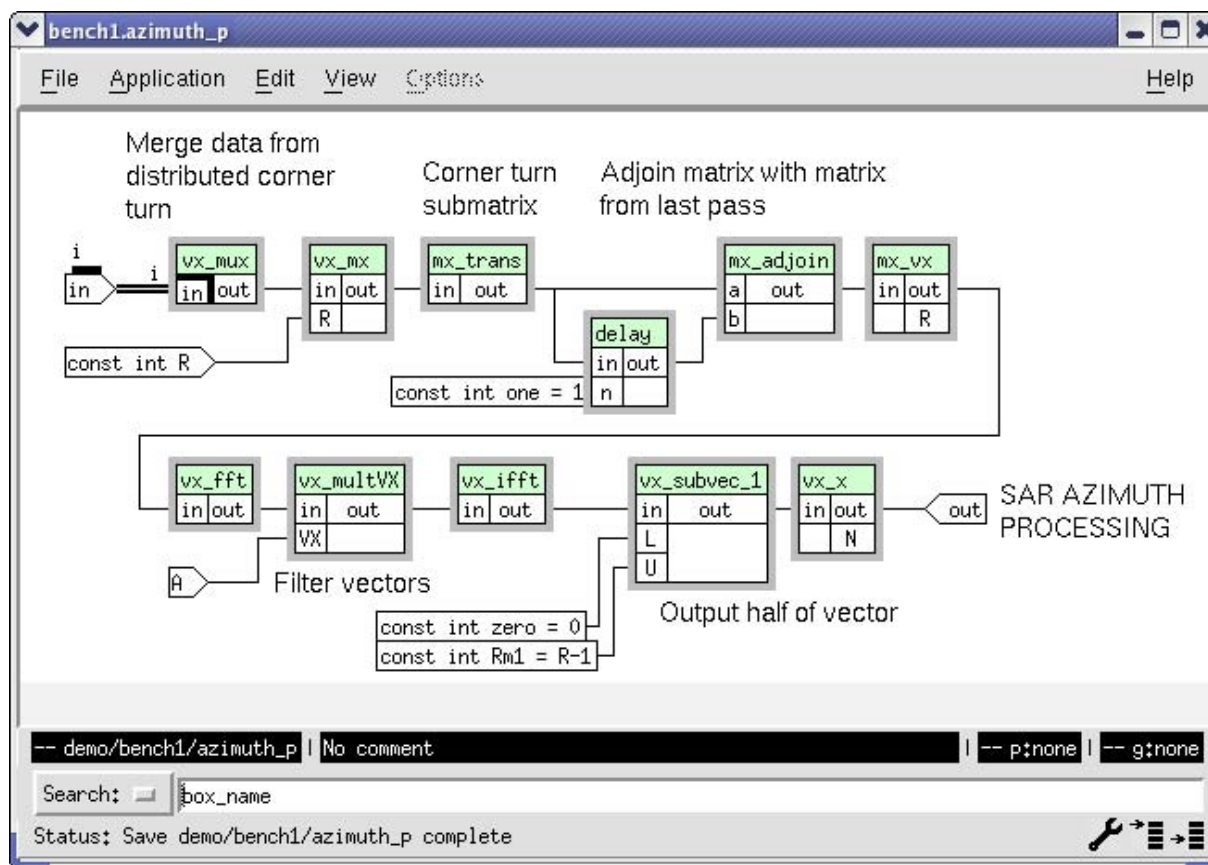
First Test Case: Small SAR



First Test Case: Small and Large SAR



First Test Case: Small and Large SAR

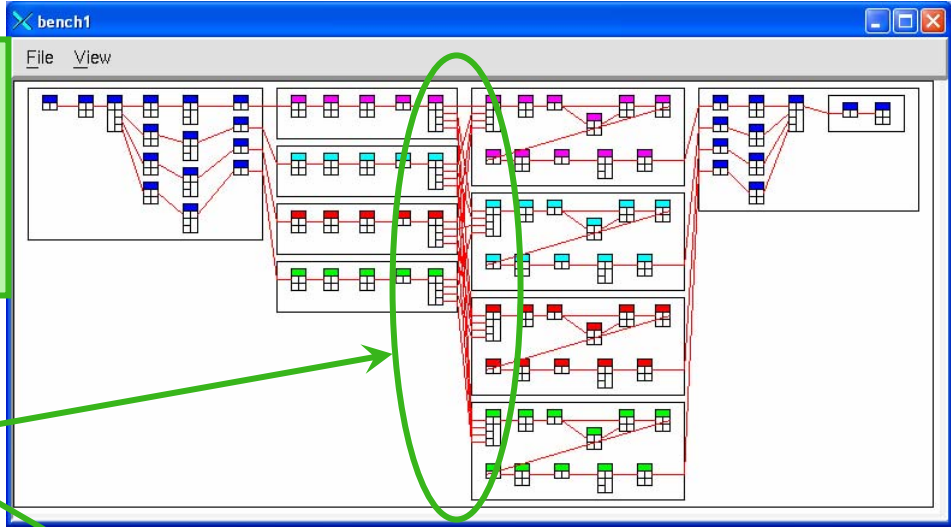


Execution of Sar Algorithm on Quad Altivec – Distributed Transpose

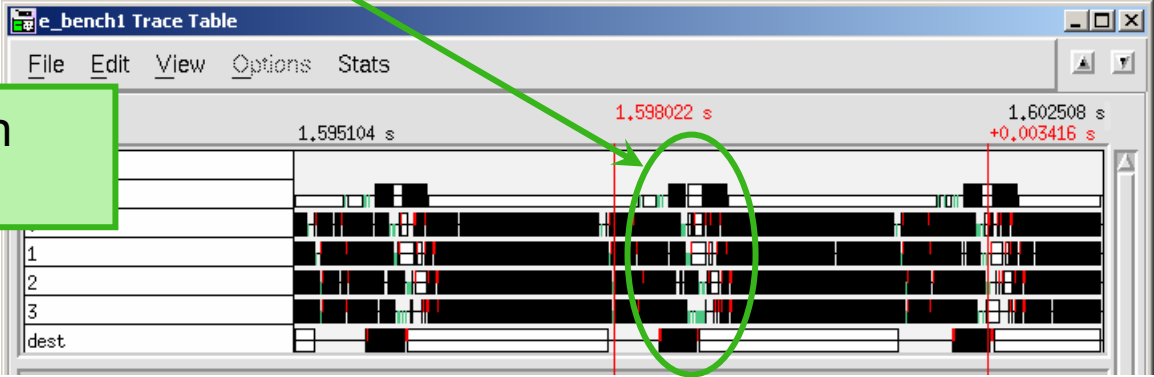


Flattened
Graph
Showing
Partitioning

Distributed
Transpose



Execution
Timeline



Small SAR Implementation Issues



- Range data size is 64×256 complex \rightarrow 128 Kbytes
 - Strip mining reduces the size of the data by processing 1 row (or N rows) at a time
- Corner turn and adjoin is 256 Kbytes so must be either performed from bulk (PPE) memory or distributed
 - First iteration has centralized transpose and adjoin implemented on PPE
- Azimuth data is 128×256 complex \rightarrow 256 Kbytes
 - Must be strip mined to fit into memory

Implementation Settings



Group 2 Partition Table

File Edit View Options

Name	Part	SubSched
[0]range_p	0	1
[1]range_p	1	1
[2]range_p	2	1
[3]range_p	3	1
[0]azimuth_p		
vx_mux	host *	1 *
vx_mx	host *	
mx_trans	host *	
delay	host *	
mx_adjoin	host *	
mx_vx	host *	
vx_fft	0 *	2 *
vx_multVX	0 *	2 *
vx_ifft	0 *	2 *
vx_subvec_1	0 *	2 *
vx_x	host *	2 *
[1]azimuth_p		
[2]azimuth_p		
[3]azimuth_p		

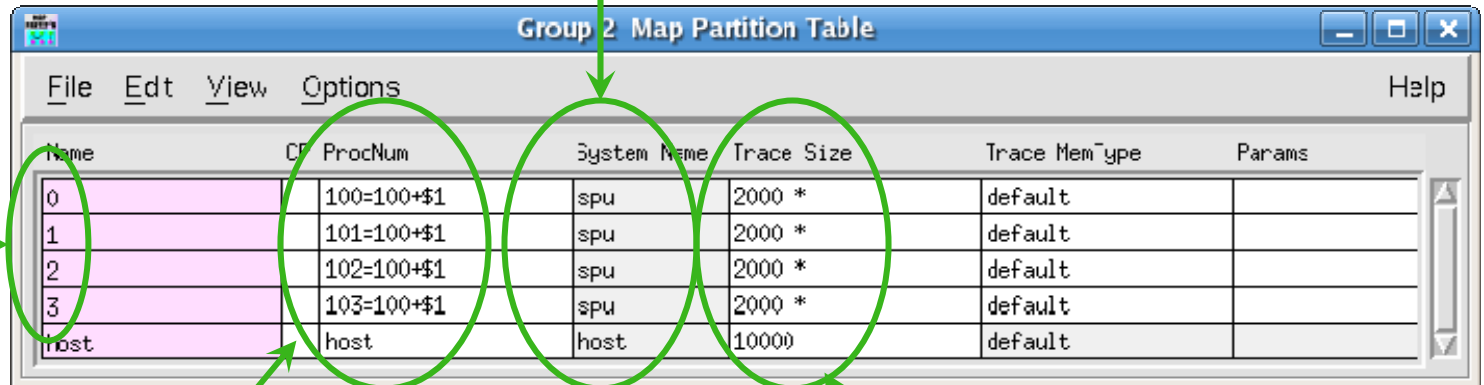
Partitioning the functionality specifies how the application is to be broken up for distribution to processors or cores.

Subscheduling is Gedae's technique for strip mining

Each partition is given an ASCII name. The host partition will run on the PPE.

Mapping of Partitions to Processors

Logical processors are either SPU's or PPU's



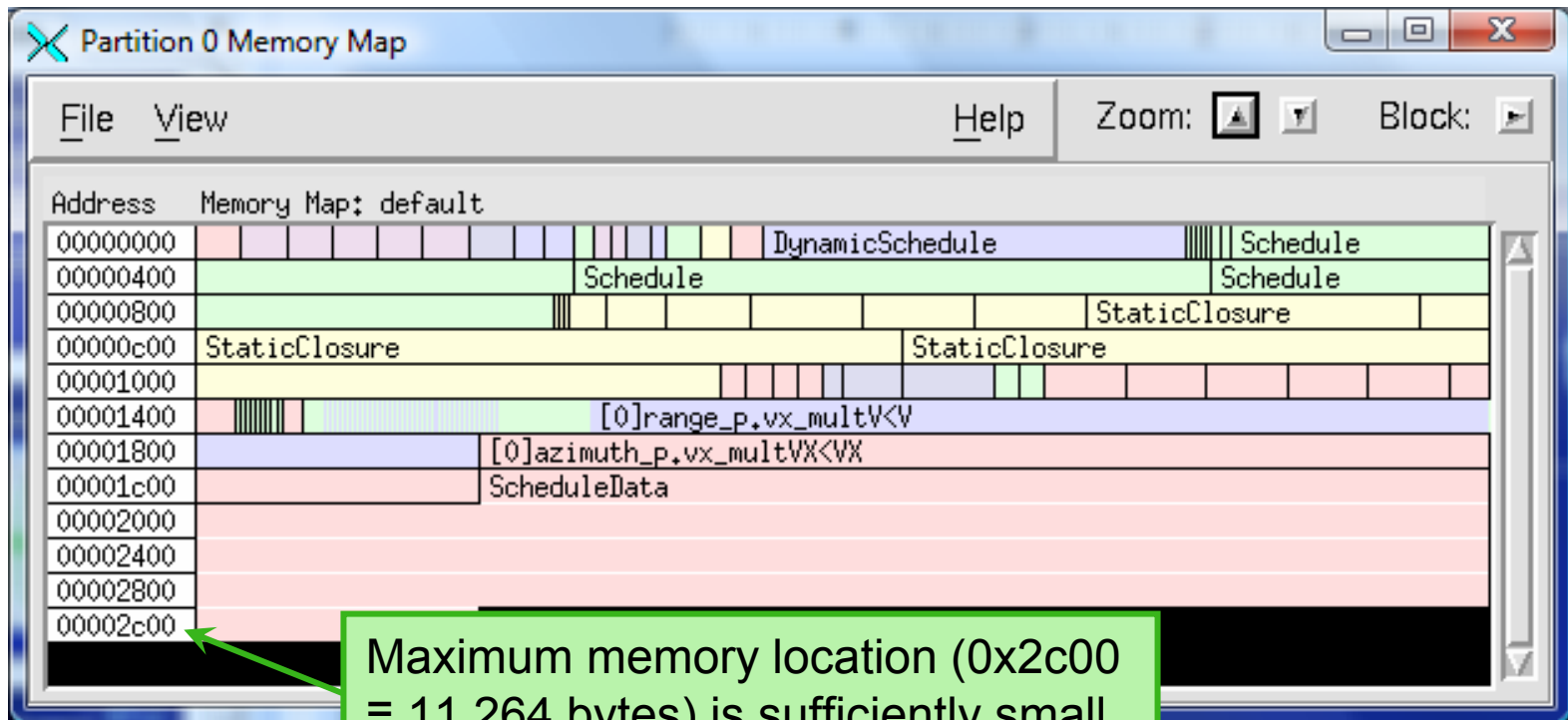
Name	CP	ProcNum	System Name	Trace Size	Trace Mem_type	Params
0		100=100+\$1	spu	2000 *	default	
1		101=100+\$1	spu	2000 *	default	
2		102=100+\$1	spu	2000 *	default	
3		103=100+\$1	spu	2000 *	default	
host		host	host	10000	default	

\$1

Partitions are mapped to logical processors using an equation with variable \$1

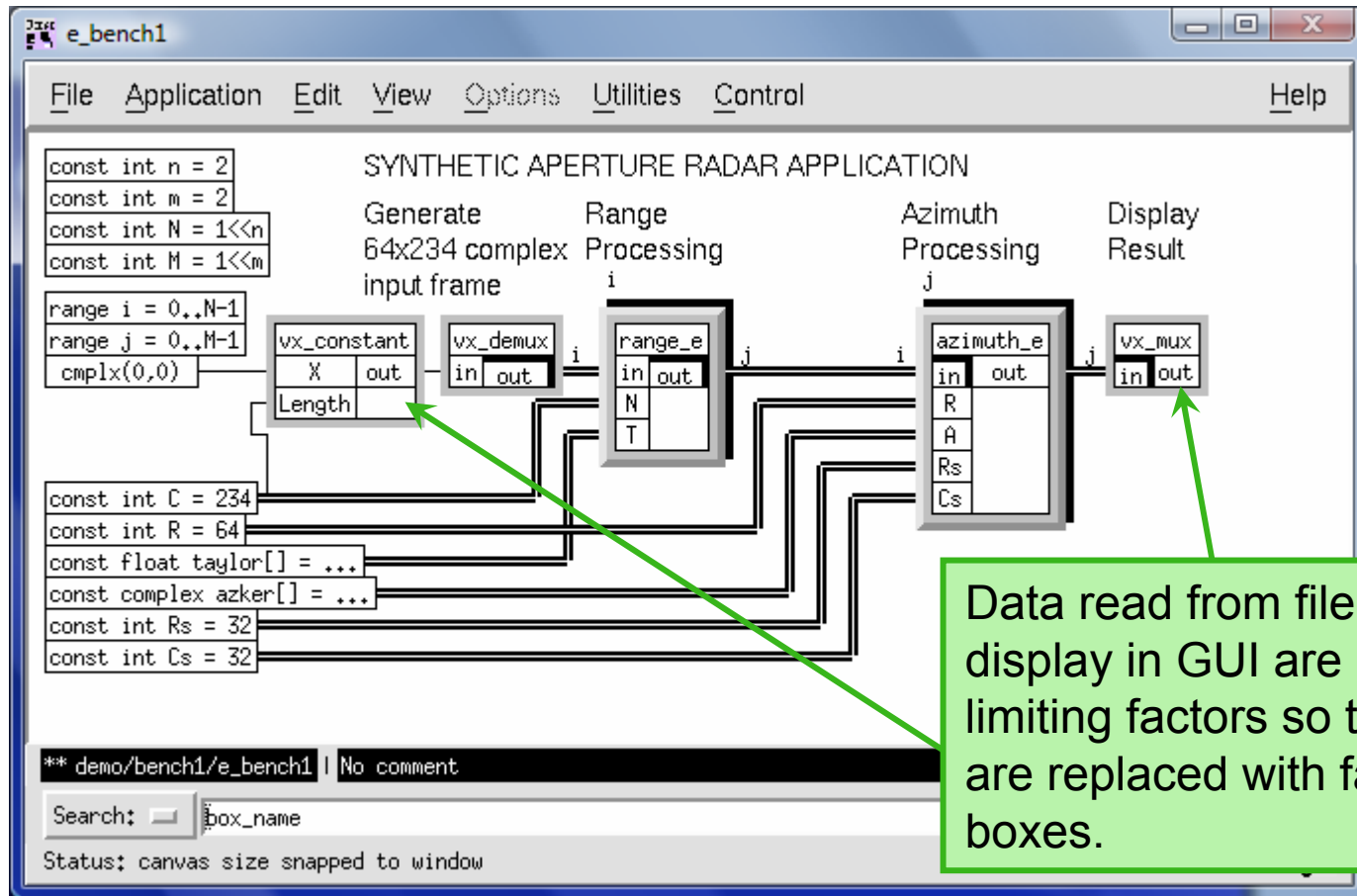
Trace buffer size is set to 2000 to conserve memory.

Memory Map Shows SPU Data Memory Foot Print

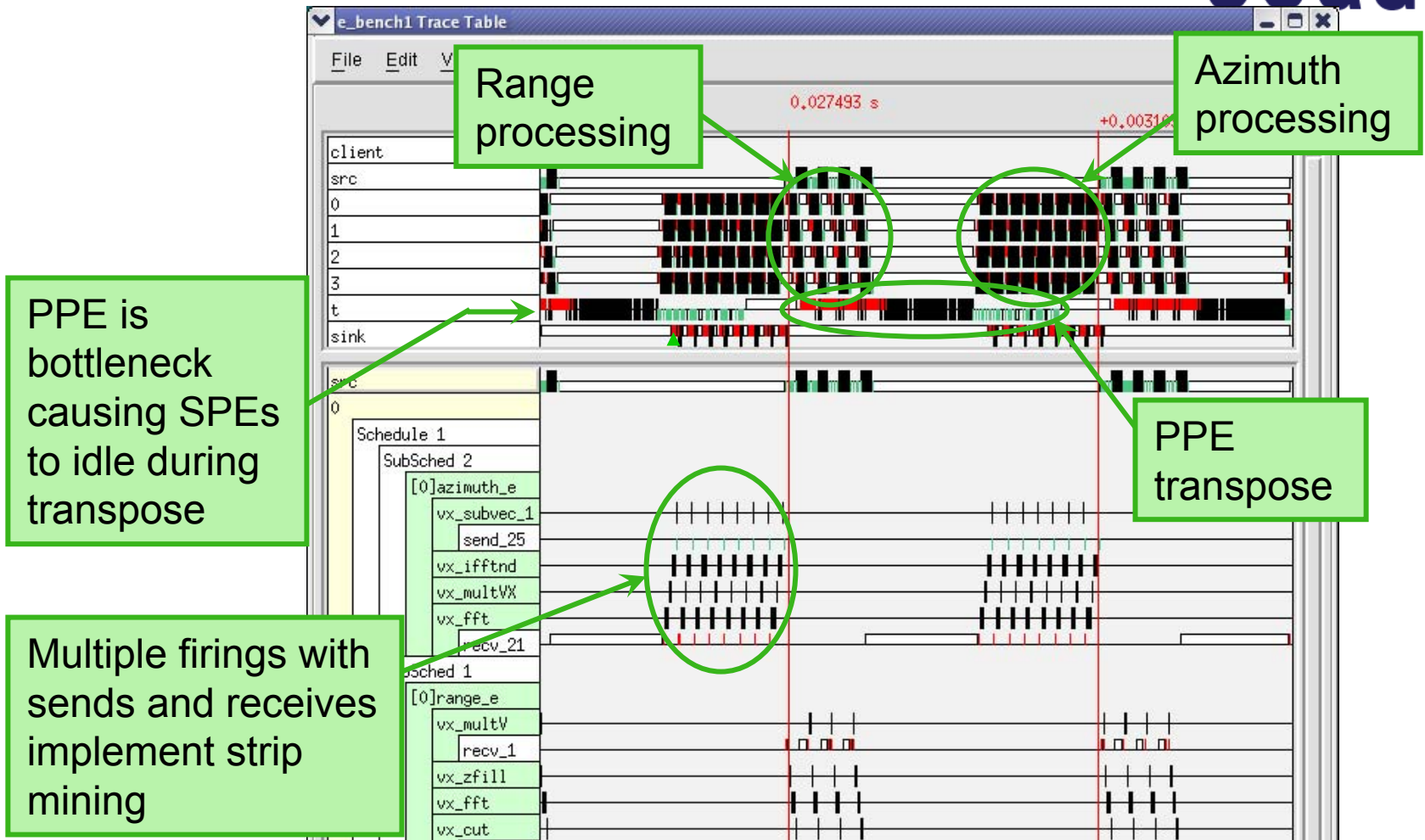


Maximum memory location (0x2c00 ≡ 11,264 bytes) is sufficiently small to easily fit in SPE memory

Graph Modified to Measure Throughput



Execution of SAR Algorithm on Cell – Centralized Transpose



Implementing Distributed Transpose

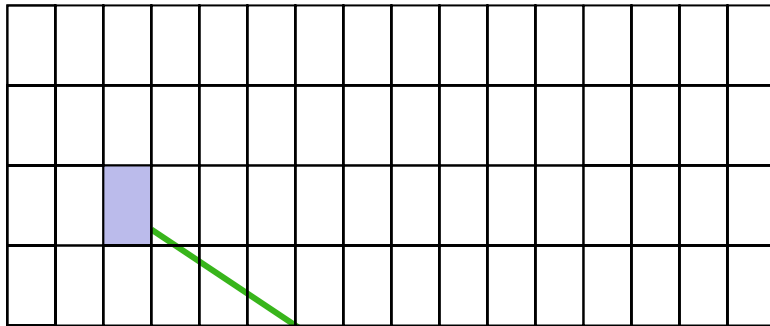


- Distributed transpose requires movement of data from bulk memory to SPE transposing and then inserting back into bulk memory
 - The implementation issues are illustrated on the next chart

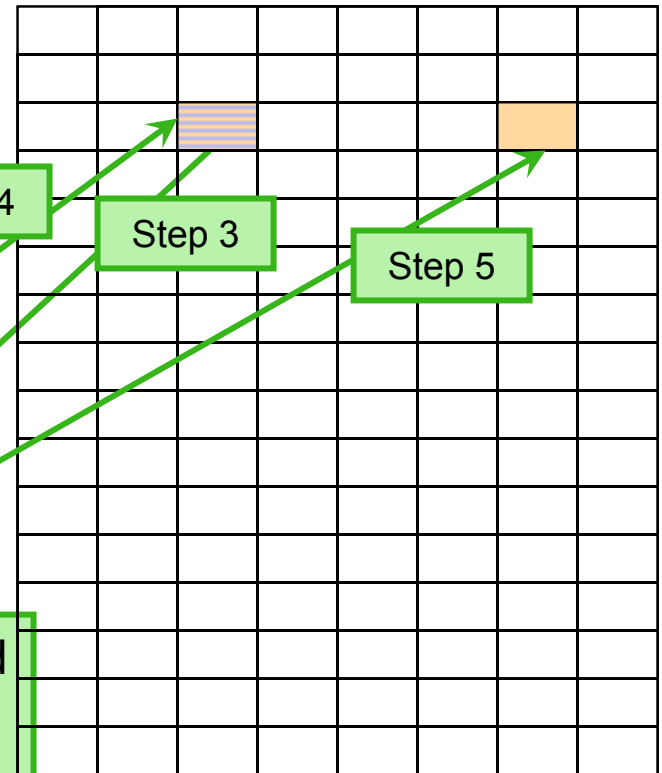
Mapping of Data to Processor for Range and Azimuth Processing



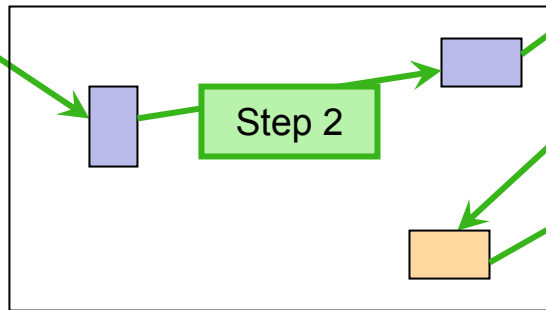
Input array in bulk memory



Output Array



Step 1



Step 2

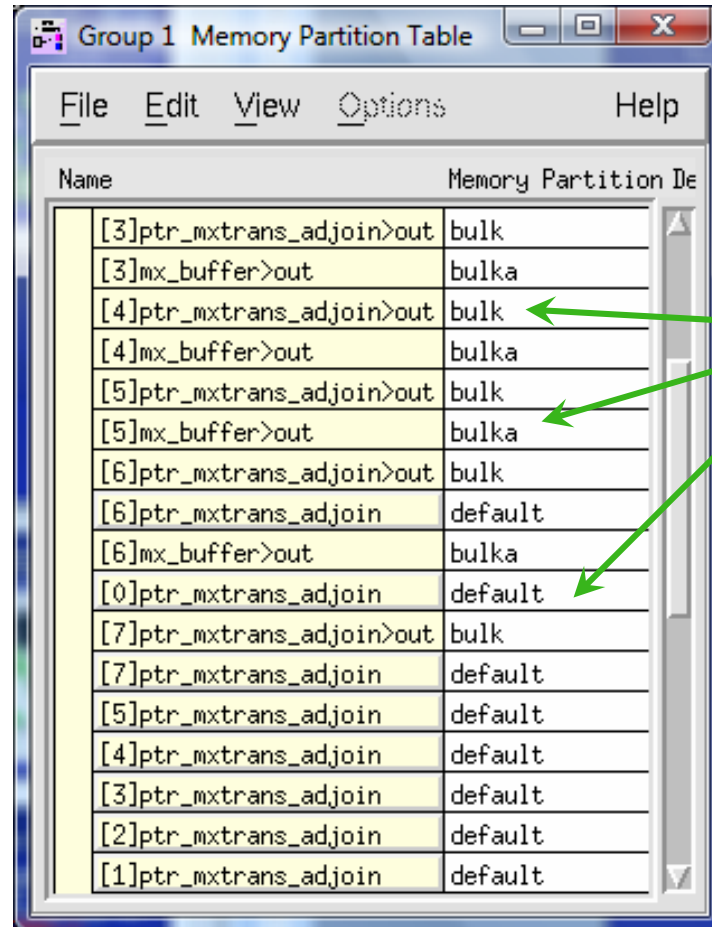
Step 4

Step 3

Step 5

Distributed transpose and adjoin are combined to minimize data movement over bus. This operation is repeated for every sub-rectangle

Mapping Buffers to Memory



The screenshot shows a window titled "Group 1 Memory Partition Table" with a menu bar containing "File", "Edit", "View", "Options", and "Help". The main content is a table with two columns: "Name" and "Memory Partition De". The table lists various memory partitions and their corresponding memory types.

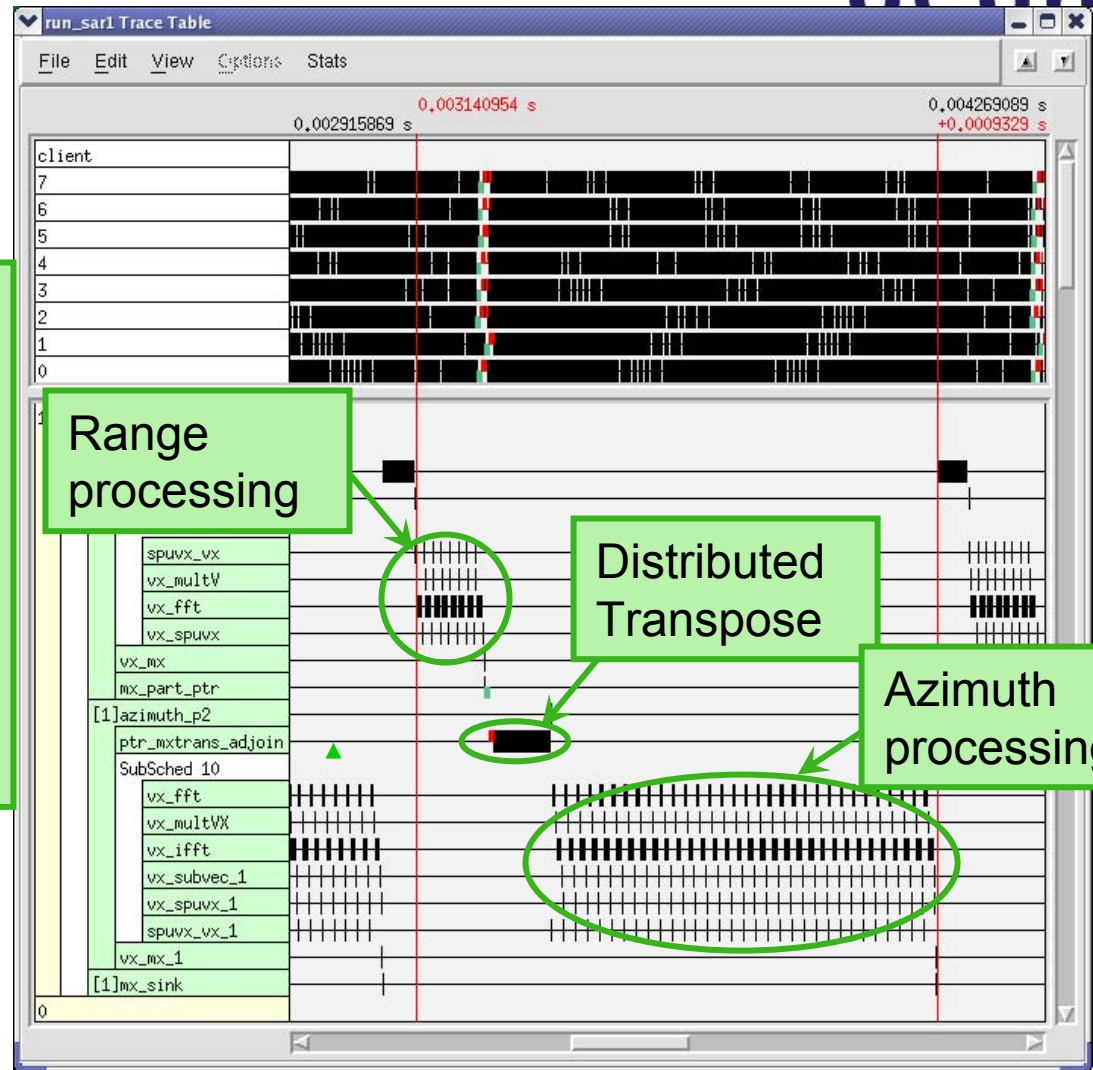
Name	Memory Partition De
[3]ptr_mxtrans_adjoin>out	bulk
[3]mx_buffer>out	bulka
[4]ptr_mxtrans_adjoin>out	bulk
[4]mx_buffer>out	bulka
[5]ptr_mxtrans_adjoin>out	bulk
[5]mx_buffer>out	bulka
[6]ptr_mxtrans_adjoin>out	bulk
[6]ptr_mxtrans_adjoin	default
[6]mx_buffer>out	bulka
[0]ptr_mxtrans_adjoin	default
[7]ptr_mxtrans_adjoin>out	bulk
[7]ptr_mxtrans_adjoin	default
[5]ptr_mxtrans_adjoin	default
[4]ptr_mxtrans_adjoin	default
[3]ptr_mxtrans_adjoin	default
[2]ptr_mxtrans_adjoin	default
[1]ptr_mxtrans_adjoin	default

Buffers are mapped to bulk or SPE memory as appropriate

Execution of SAR Algorithm on Cell Processor – Distributed Transpose



Large SAR fits on 8 SPE system with sub-scheduled range and azimuth processing and distributed transpose. Processing is very dense. Transpose is a modes percentage of the time.



Summary of Preliminary Results



- Cell timings to be announced at lecture
- Comparison of small SAR throughput
 - Cell/B.E. with PPE only _____ Hz
 - Cell/B.E. with distributed transpose _____ Hz
 - Cell/B.E. with bulk memory transpose _____ Hz
- Comparison of large SAR throughput
 - 500 Mhz Quad AltiVec Board 3 Hz
 - 3.2 Ghz Cell/B.E. _____ Hz
- Additional optimizations to be implemented
 - Overlap processing and data movement
 - Decrease function overhead

Gedae Roadmap for Cell/B.E. Support

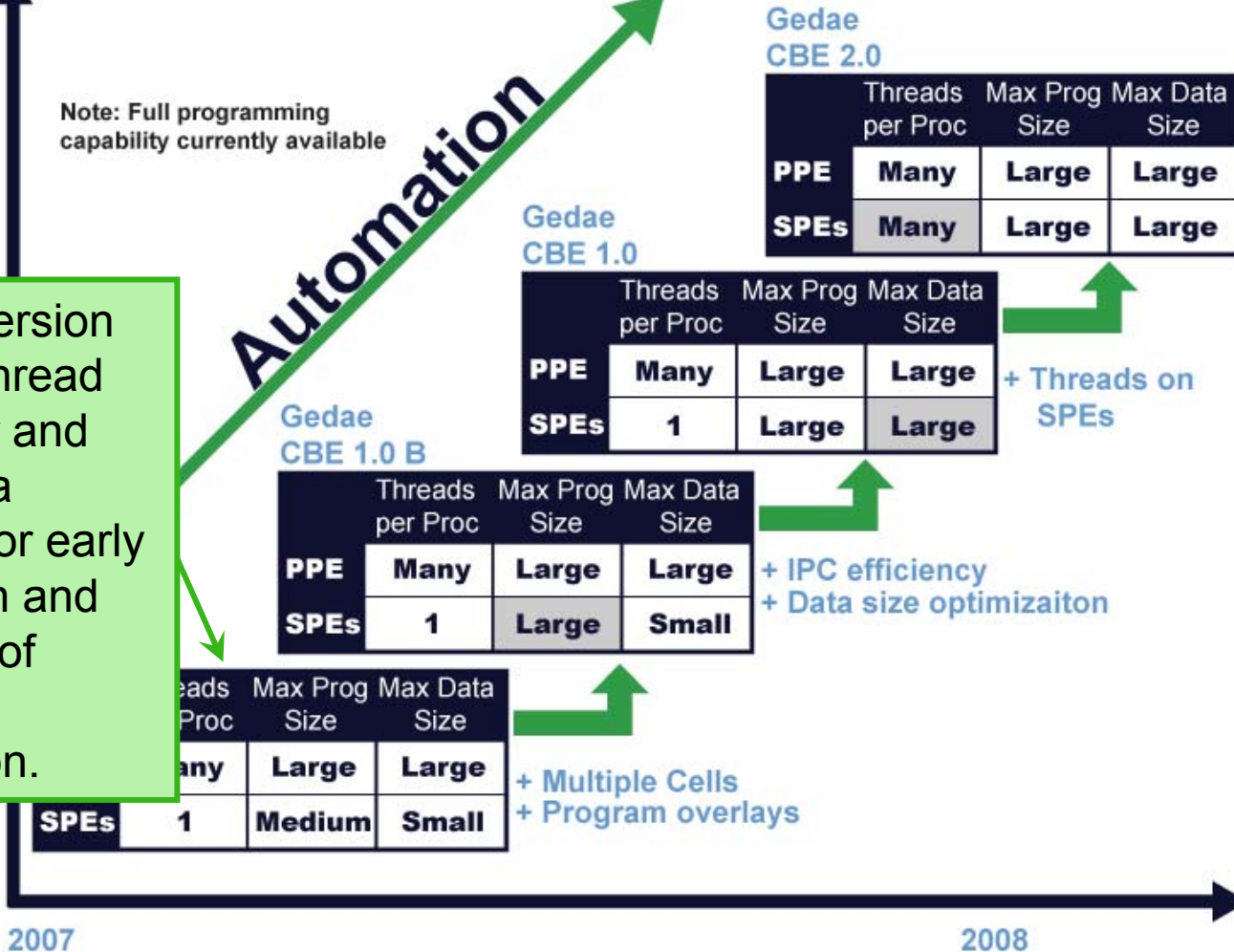


Capability

Note: Full programming capability currently available

Automation

Current version runs full thread scheduler and provides a platform for early evaluation and evolution of compiler automation.



Demonstration



- Small SAR Benchmark Demonstration on Playstation 3
 - Part 1
 - “out of the box” Gedae compiler and thread scheduler
 - Range and azimuth strip-mined on SPEs
 - Centralized transpose on PPE
 - Part 2
 - First stage of Gedae thread scheduler optimization
 - Hand crafted distributed transpose with memory management between local SPE memory and PPE bulk memory
 - Manual implementation demonstrates feasibility of adding automation to compiler

Conclusion



- Cell/B.E. and Gedae combination works
 - The Cell/B.E. multi-core technology delivers an innovative and powerful microprocessor architecture for new levels of energy efficiency and performance, while
 - Gedae provides an equally powerful application development environment

