



IEEE RADAR Conference 2007

Parallel Processing on a Multicore Processor

Authors:

Kerry Barnes, Bill Lundgren, Jim Steed
Gedae, Inc. (Email: bill.lundgren@gedae.com)

Overview of Presentation



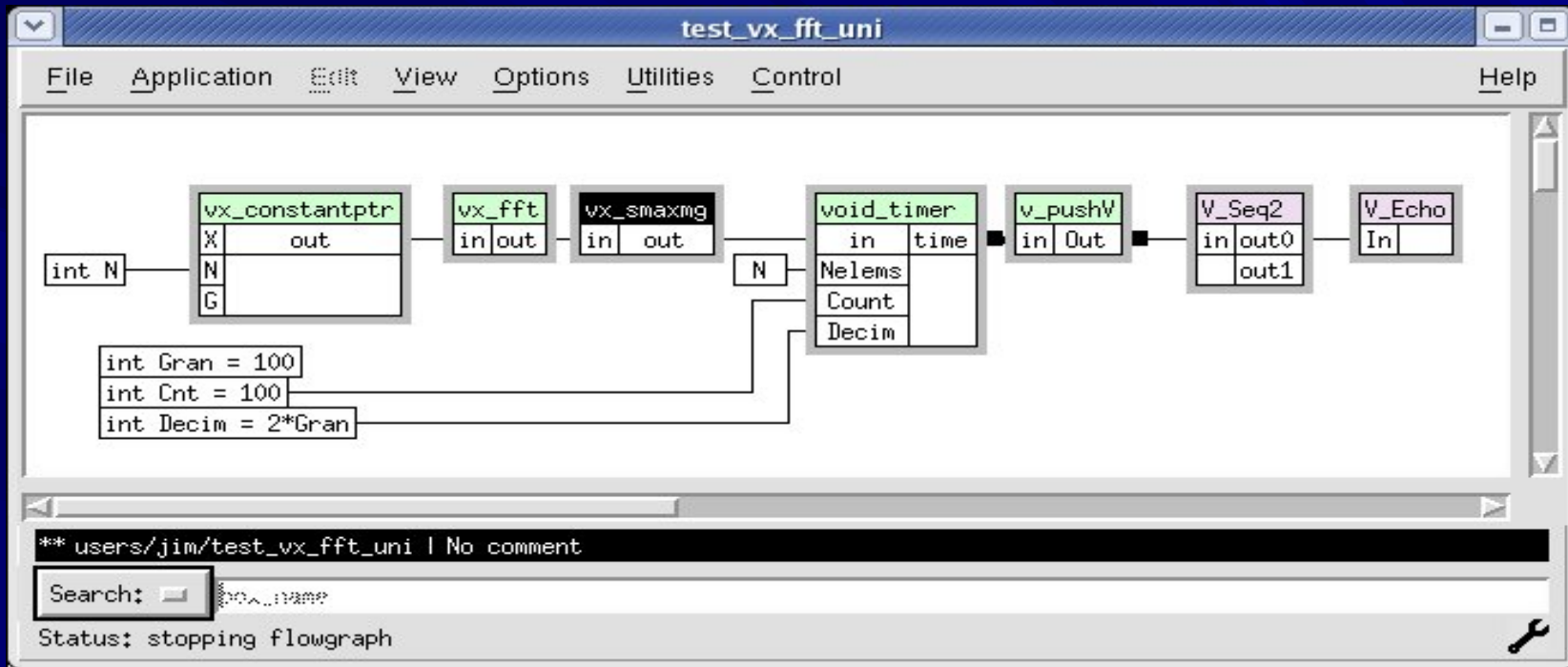
- One of the core components of RADAR processing is the matched filtering
- Frequently done using Fourier Transforms on a 2 dimensional array of data.
- Parallel execution shortens latency – round robin only increases throughput
- This paper explores those issues by showing:
 - Distribution of FFT processing on a dual core Intel processor
 - Distribution of the range and azimuth processing over 2 quad processor Mercury boards
 - Effect of data transfer methods among boards



IEEE RADAR Conference 2007

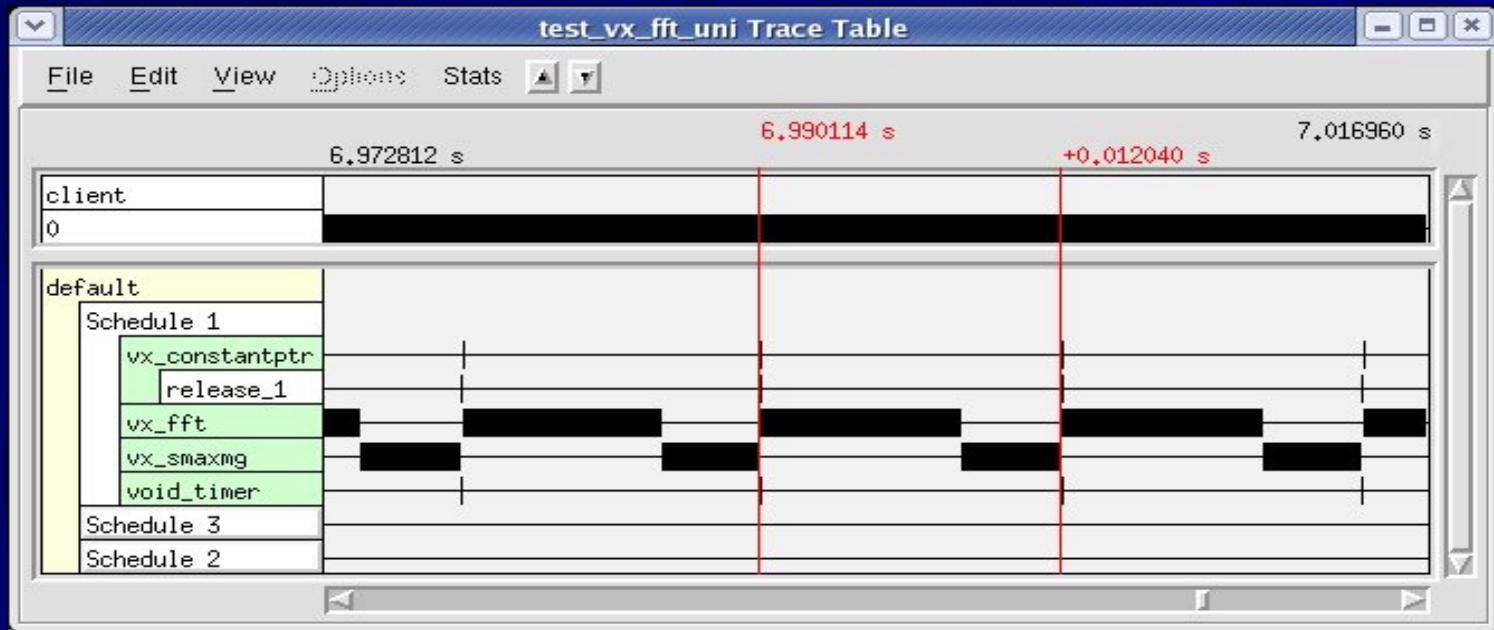
Intel Core Duo Processor Example

Block Diagram for Single Core



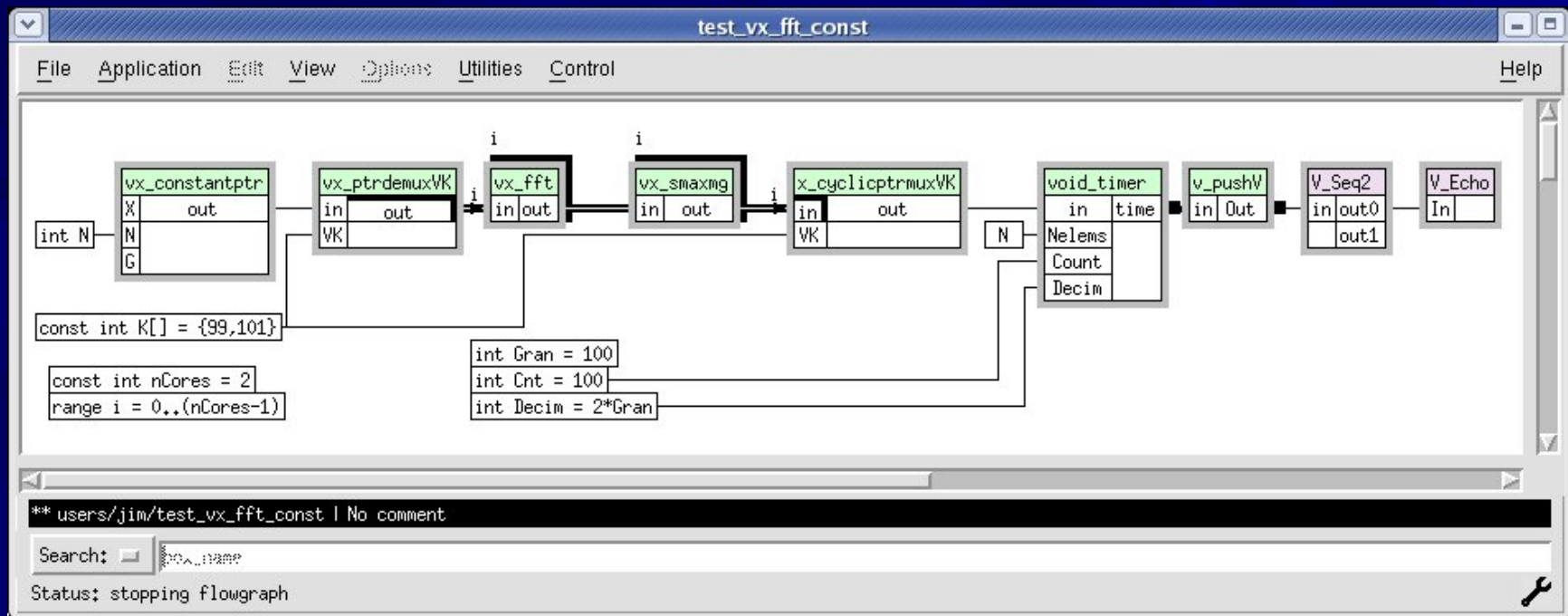
- Single core implementation requires no data partitioning
- Each timing is of 200 1K vectors

Trace Table for Single Core Implementation



- Single core implementation takes ~12ms

Block Diagram with Distribution for Multiple Cores



- Multicore implementation partitions the data through a demux operation where the partitioning is defined by $K[]$

Implementation on Dual Core



Assign boxes to partitions

Then map partitions to cores

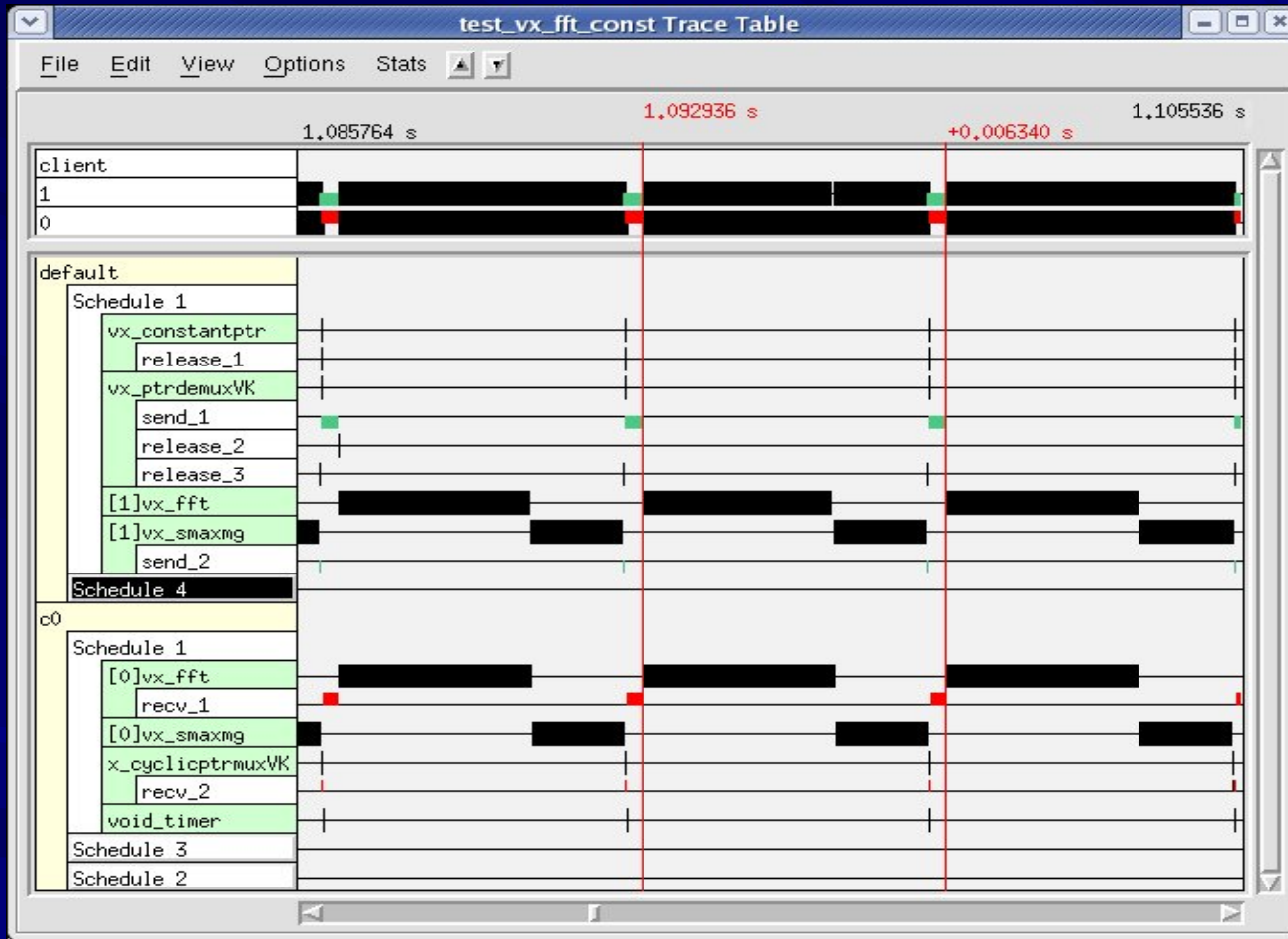
Change granularity to 1 so that each token is multiplexed separately

Name	Part	SubSched
vx_constantptr	default	
vx_ptrdemuxVK	default	
[0]vx_fft	c0 *	
[1]vx_fft	default	
[0]vx_smaxmg	c0 *	
[1]vx_smaxmg	default	
x_cyclicptrmuxVK	c0 *	
void_timer	c0 *	
v_pushV	c0 *	

Name	Gran	Mult	Nat	Gran	Max	Gran	Granularity	Priority
Schedule 1	1							
[0]vx_fft			100		100			0
recv_1			100		100			0
[0]vx_smaxmg			100		100			0
[1]vx_fft			100		100			0
[1]vx_smaxmg			100		100			0
send_2			100		100			0
void_timer		1			1			0
vx_constantptr		2			2			0
vx_ptrdemuxVK		1			1			0
send_1			100		100			0
x_cyclicptrmuxVK		2			2		1 *	0
recv_2			100		100			0
Schedule 2	1							

Name	CP	ProcNum	System Name	Trace Size	Trace MemType	Params
c0	0	*	eredhat	10000	default	
default	1	*	eredhat	10000	default	

Trace Table for Core Duo Implementation



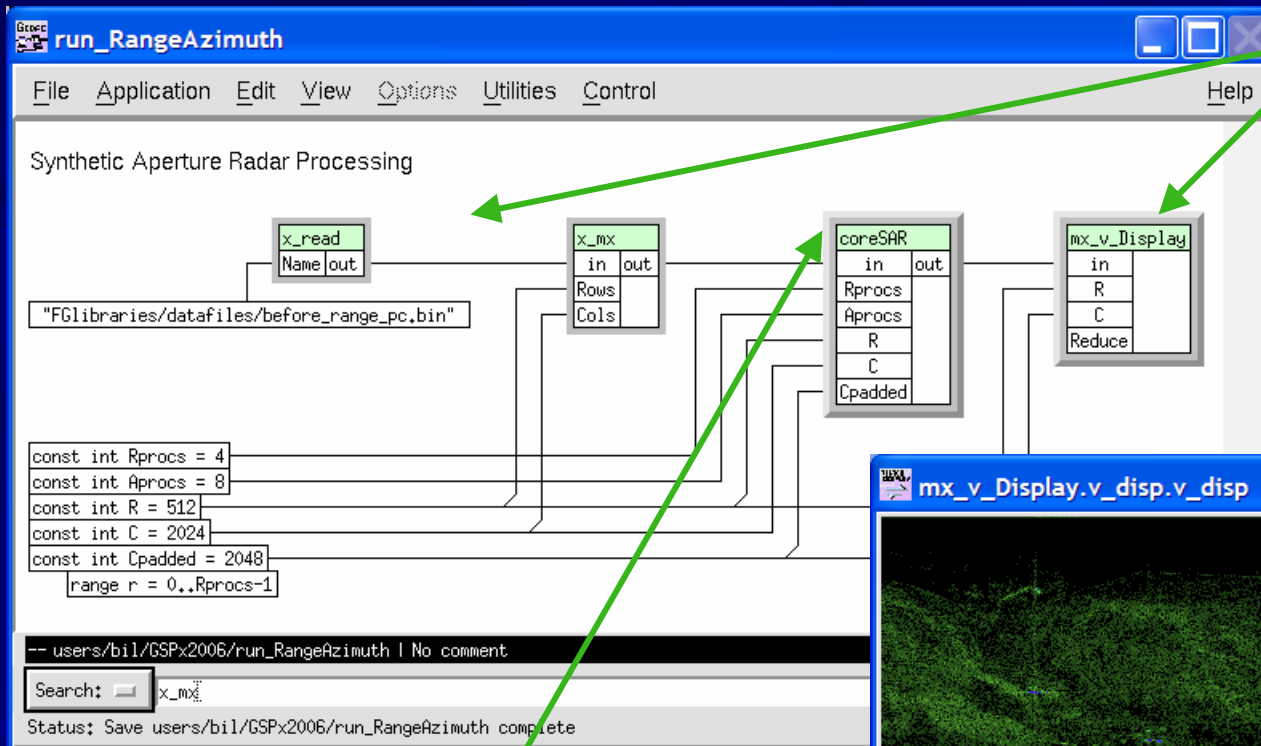
- Single core takes ~12ms
- Speed up achieved is 1.899 X



IEEE RADAR Conference 2007

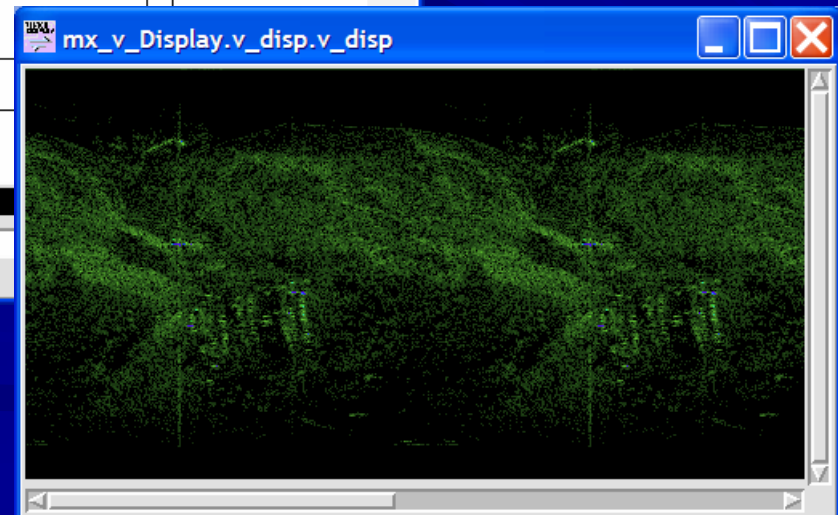
Mercury AdapDev 4 Processor Example

Top Level SAR Graph with Data Display

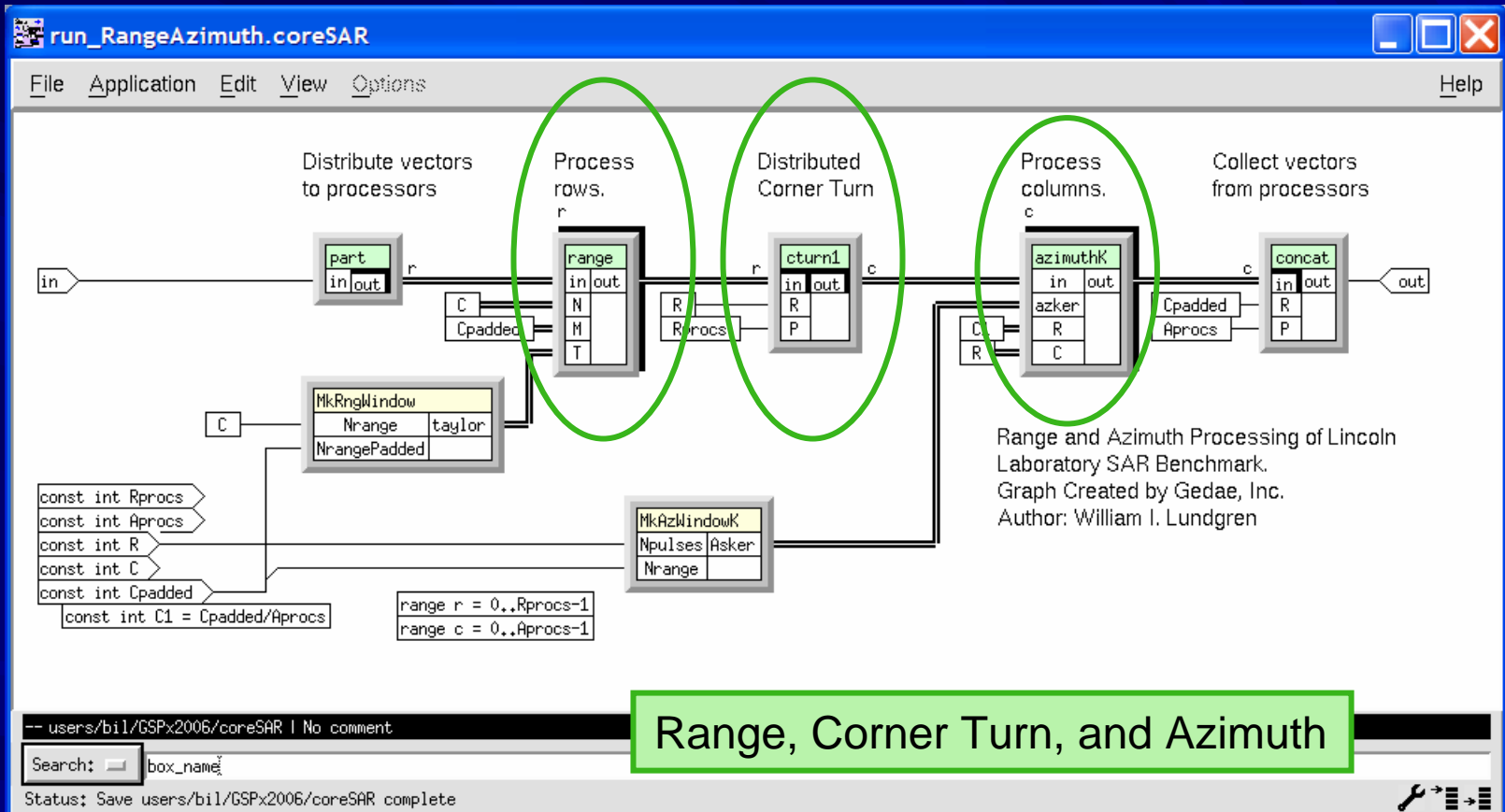


Data source and display

Accelerate range and azimuth processing and corner turn in this subgraph



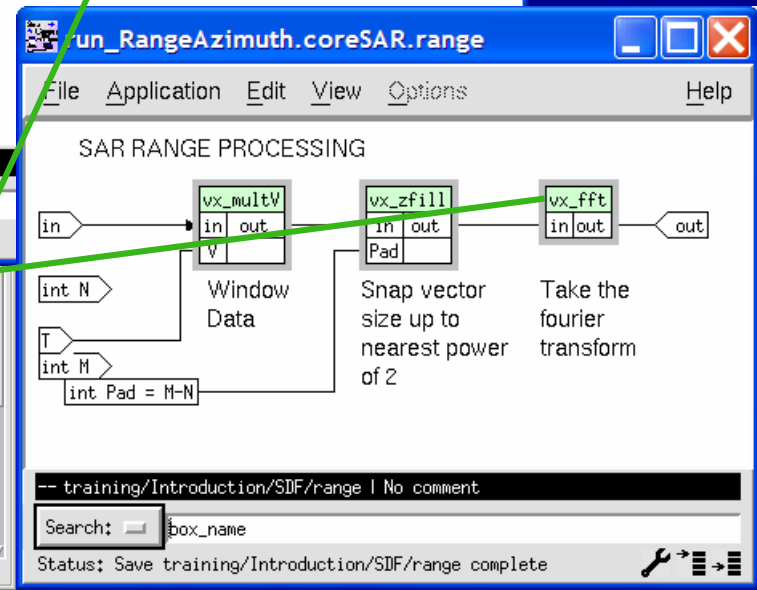
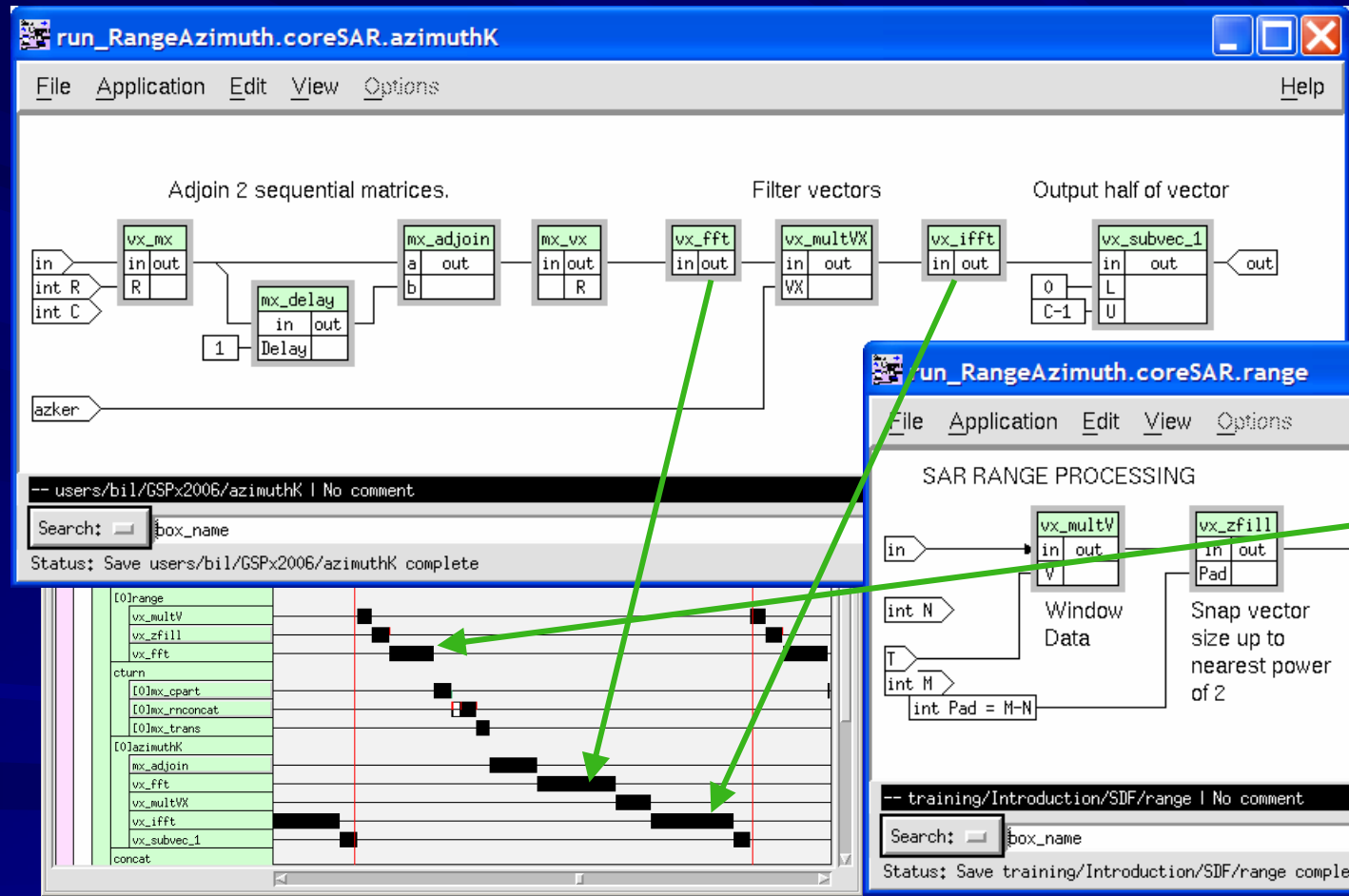
Core SAR Processing Graph



Azimuth and Range Processing Graphs



- Primary load is FFTs

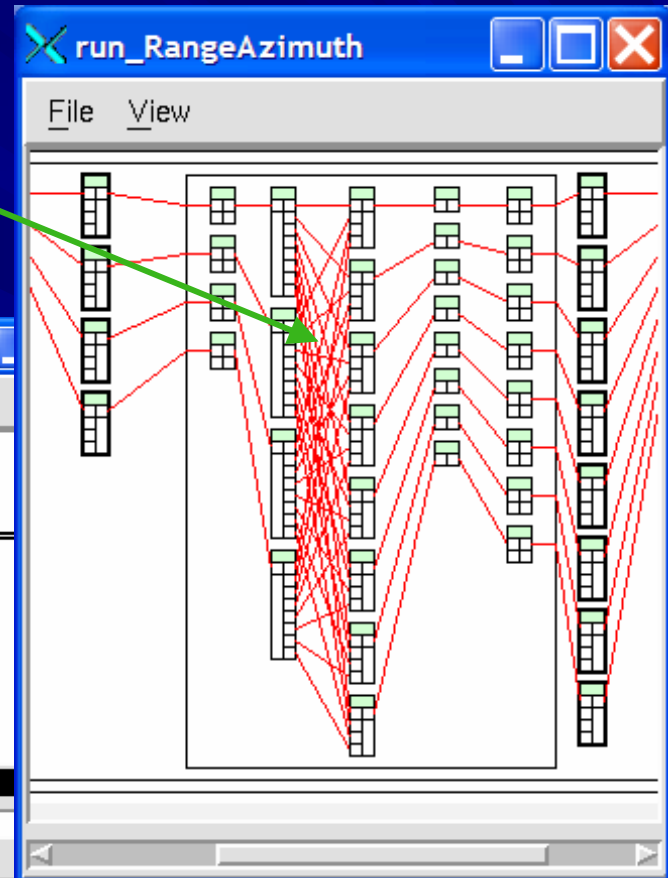
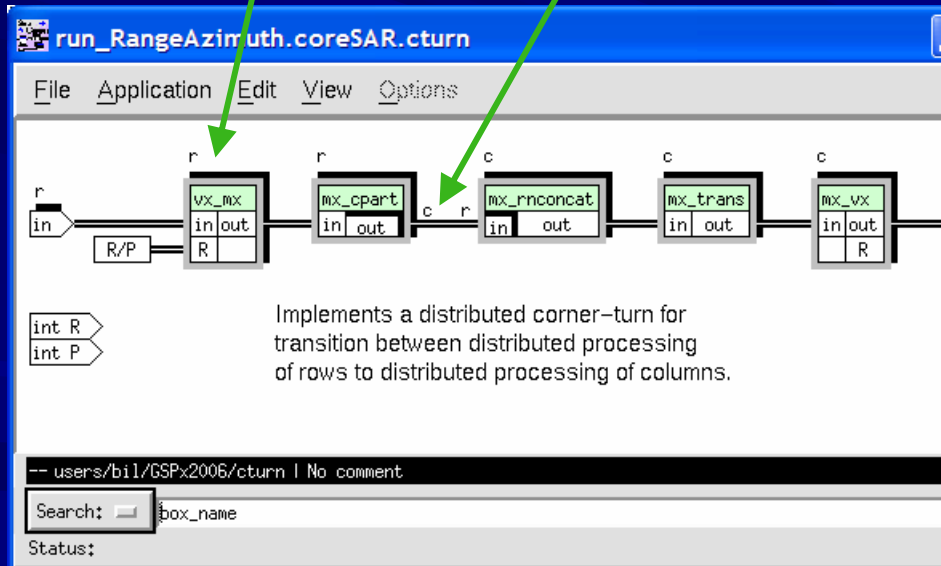


Corner Turn Processing Graph



Families, indicated by shadowing & index, represent an array of elements

Corner turn specified by connecting r-x-c family to c-x-r



Partition Graph and Map to 6 AltiVec Processors



The image shows two overlapping windows from a software application. The foreground window is titled "Group 2 Map Partition Table" and contains a table with the following data:

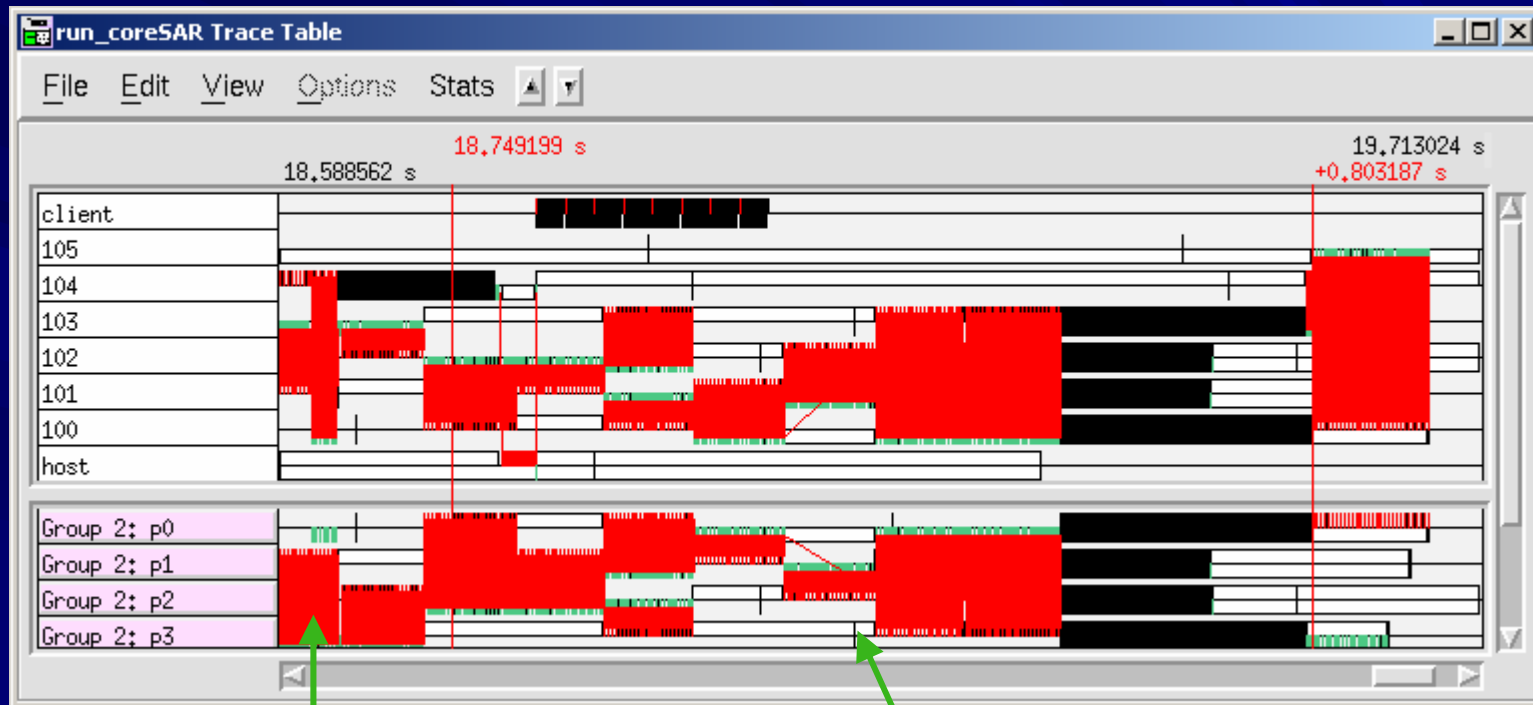
Name	CP ProcNum	System Name	Trace Size	Trace MemType	Params
h	host	host	10000	default	
p0	100 *	mcos_altivec	10000	default	
p1	101 *	mcos_altivec	10000	default	
p2	102 *	mcos_altivec	10000	default	
p3	103 *	mcos_altivec	10000	default	
sink	104 *	mcos_altivec	10000	default	
src	105 *	mcos_altivec	10000	default	

The background window is titled "Group 2 Partition Table" and shows a tree view of a partition graph. The "coreSAR" node is expanded, showing sub-nodes like "part", "cturn", and "concat". The "part" node is further expanded to show four "range" entries (p0 through p3) and four "azimuthK" entries (p0 through p3). Each "range" entry has a parameter value of "p="+\$1. Each "azimuthK" entry has a parameter value of "p="+\$1.

Two green callout boxes with arrows provide instructions:

- "Assign boxes to partitions" (arrow points to the "part" node in the partition table)
- "Then map partitions to processors" (arrow points to the "p0" through "p3" rows in the map partition table)

Results With Default Transfer Methods

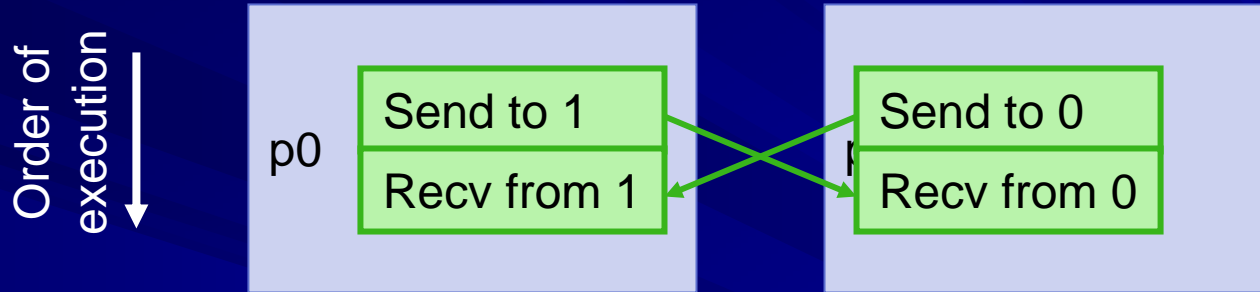


Each red line is a send/receive pair

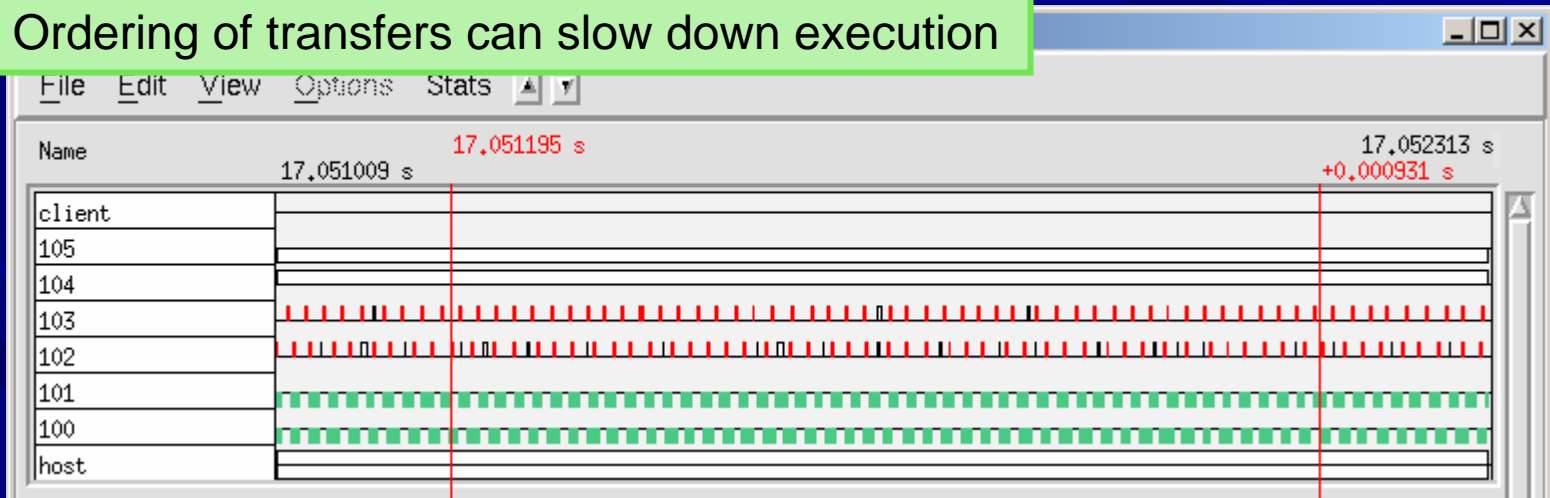
Default transfer method blocks, and communication dominates timing

Blocking Transfers Can Cause Slowdown and Deadlock

Ordering of transfers can cause classic data flow deadlock



Ordering of transfers can slow down execution



Change Transfer Methods to Nonblocking



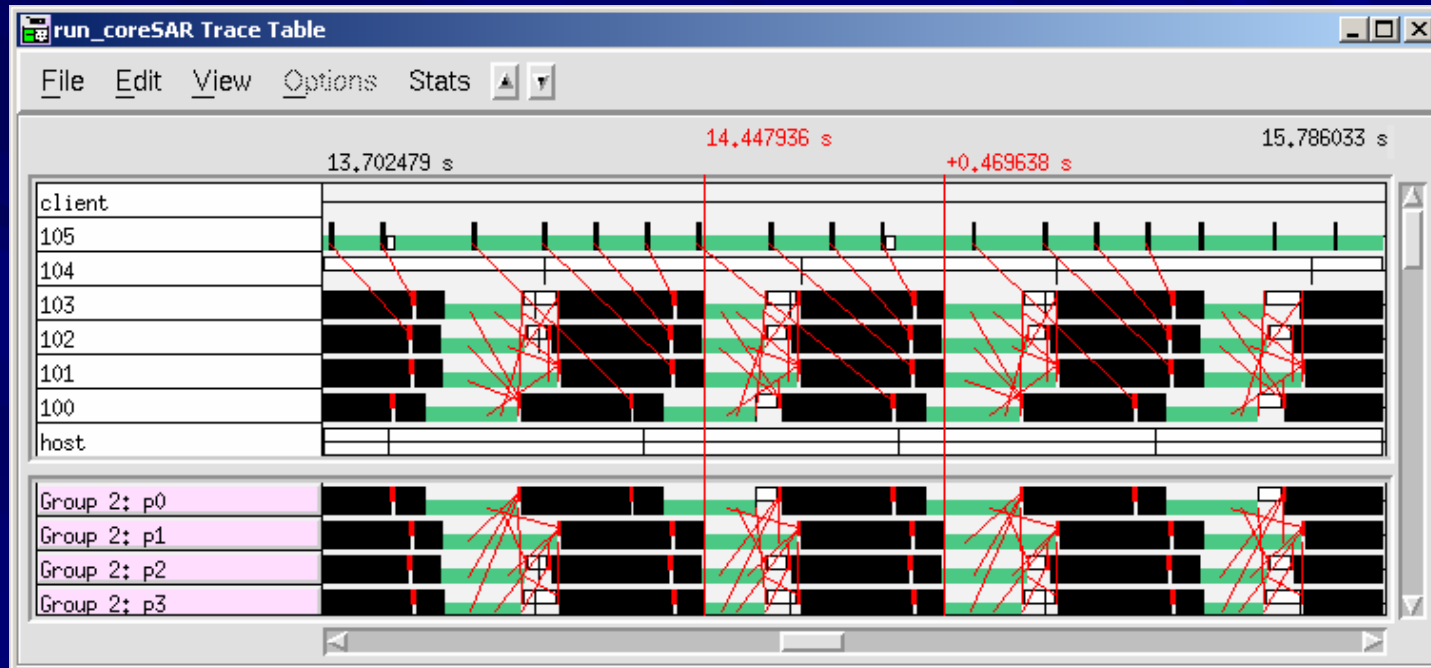
Group 2 Transfer Table

File Edit View Options Help

Name	Id	Source	Dest	Xfer Type	NBsize	Send
mx_v_Display						
m_s<in	18	104	host	mcos_active<host	nb: 65536 *	
coreSAR						
part						
cturn						
[0]mx_rnconcat<[1]in	9	101	100	stream_shared_mem *	nb: 524288 *	
[0]mx_rnconcat<[2]in	12	102	100	stream_shared_mem *	nb: 524288 *	
[0]mx_rnconcat<[3]in	15	103	100	stream_shared_mem *	nb: 524288 *	
[1]mx_rnconcat<[0]in	6	100	101	stream_shared_mem *	nb: 524288 *	
[1]mx_rnconcat<[2]in	13	102	101	stream_shared_mem *	nb: 524288 *	
[1]mx_rnconcat<[3]in	16	103	101	stream_shared_mem *	nb: 524288 *	
[2]mx_rnconcat<[0]in	7	100	102	stream_shared_mem *	nb: 524288 *	
[2]mx_rnconcat<[1]in	10	101	102	stream_shared_mem *	nb: 524288 *	
[2]mx_rnconcat<[3]in	17	103	102	stream_shared_mem *	nb: 524288 *	
[3]mx_rnconcat<[0]in	8	100	103	stream_shared_mem *	nb: 524288 *	
[3]mx_rnconcat<[1]in	11	101	103	stream_shared_mem *	nb: 524288 *	
[3]mx_rnconcat<[2]in	14	102	103	stream_shared_mem *	nb: 524288 *	
concat						

Enable nonblocking, and Gedae automatically computes sizes and creates buffers

Performance Improvement of Nonblocking Communications



- Significantly faster communication, but still slow
- CPU is processing each transfer

Change Transfer Methods to DMA

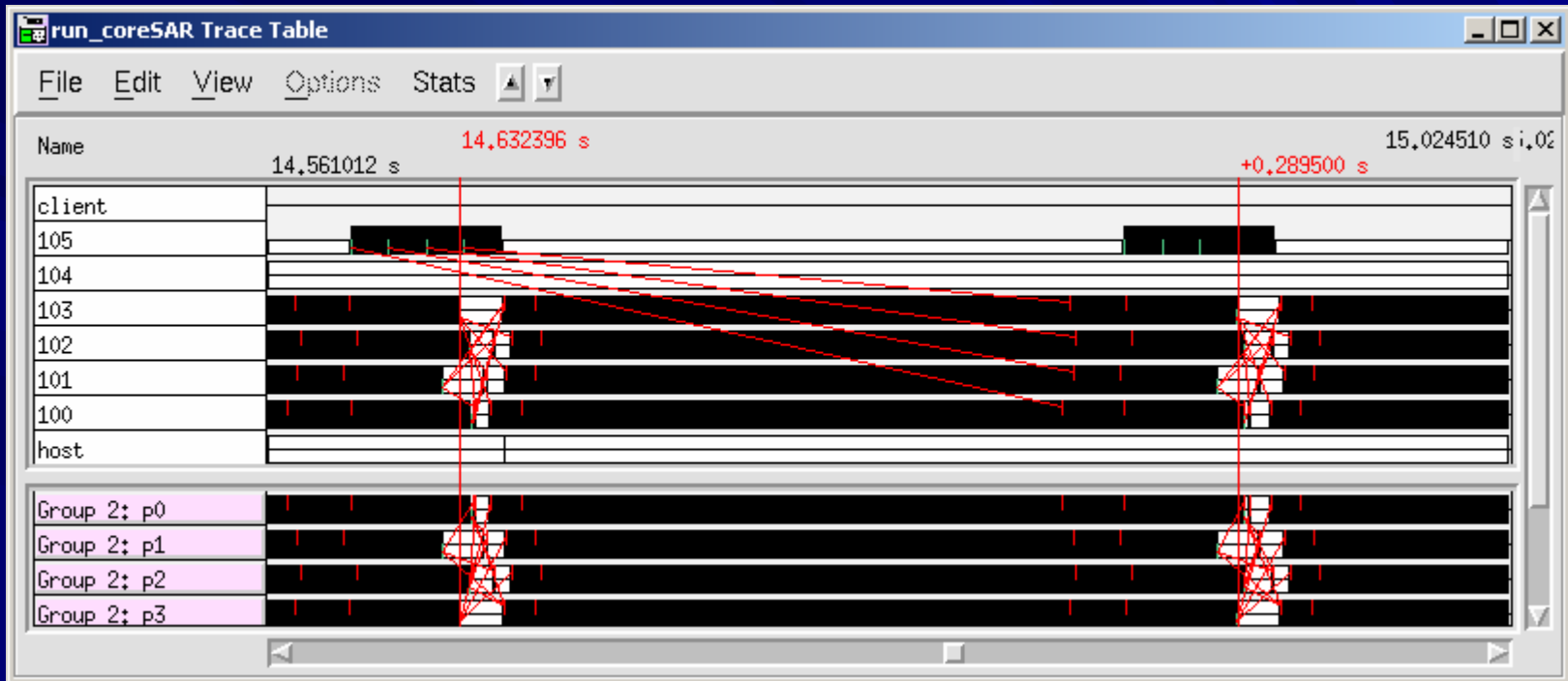
Group 2 Transfer Table

File Edit View Options Help

Name	Id	Source	Dest	Xfer Type	NBsize	Senc
mx_v_Display						
m_s<in	18	104	host	mcos_altivec>host	nb: 65536 *	
coreSAR						
part						
cturn						
[0]mx_rnconcat<[1]in	9	101	100	dsa_dx *		
[0]mx_rnconcat<[2]in	12	102	100	dsa_dx *		
[0]mx_rnconcat<[3]in	15	103	100	dsa_dx *		
[1]mx_rnconcat<[0]in	6	100	101	dsa_dx *		
[1]mx_rnconcat<[2]in	13	102	101	dsa_dx *		
[1]mx_rnconcat<[3]in	16	103	101	dsa_dx *		
[2]mx_rnconcat<[0]in	7	100	102	dsa_dx *		
[2]mx_rnconcat<[1]in	10	101	102	dsa_dx *		
[2]mx_rnconcat<[3]in	17	103	102	dsa_dx *		
[3]mx_rnconcat<[0]in	8	100	103	dsa_dx *		
[3]mx_rnconcat<[1]in	11	101	103	dsa_dx *		
[3]mx_rnconcat<[2]in	14	102	103	dsa_dx *		
concat						

Change Xfer type to Direct Schedule Access (DSA) to implement DMA transfers

Performance Improvement of DMA Communications



- DMA unit processes each transfer, not CPU
- 1.623x speedup for processing + communication

Conclusions



- Gedae provides a graphical method for describing parallel processing
- Creating a parallel implementation in Gedae for a multicore or DSP board is a simple process of partitioning and mapping the components to processors
- The Gedae Trace Table is invaluable for diagnosing problems with the parallel implementation
- Gedae allows for quick and easy experimentation with parallel multicore and embedded implementations, offering many degrees of freedom in the implementation