From Prototyping to Emulation:
The StarT (*T) Era
(1992-1999)

Derek Chiou
(Dataflow-StarT-Synthesis Era occupant)
The University of Texas at Austin
Machine Building In CSG
1992-1999

- *T
  - 88110MP-based
- StarT-NG (Next Generation)
  - PowerPC 620-based
- StarT-Voyager
  - PowerPC 604-based
- StarT-X, StarT-Jr
  - x86 PCI-based
- Moving forward: RAMP
Dataflow Machines Looked Impractical

- Monsoon worked well, but
  - IBM RS/6000 donated at the same time was about as fast as 8 node Monsoon machine

- Could we leverage commercial processors?
*T: Integrated Building Blocks for Parallel Computing

Greg Papadopoulos, Andy Boughton, Robert Greiner, Michael J. Beckerle

MIT and Motorola
**T: Motorola 88110MP**

- Integrates NIU onto Motorola 88110 core
  - A functional unit
- Send/Receive instructions to access NIU
  - Use general-purpose registers
- Asymmetric message passing performance
  - Dual issue means 4 read ports, 2 write ports
- Motorola was doing the implementation
  - Many visits to Phoenix
- We grumbled
  - 6 cycles to send a message, 12 cycles to receive????
  - Monsoon was much better
And Then Arvind Has a Meeting

- And comes back with some news
- IBM/Motorola/Apple alliance
  - Out goes 88110
  - In comes PowerPC
- \textit{Re-*T}?
- PowerPC 620 selected as base processor
  - Not yet implemented, very aggressive 64b processor
- StarT-NG was born
StarT-NG: Delivering Seamless Parallel Computing

Derek Chiou, Boon S. Ang, Robert Greiner, Arvind, James Hoe, Michael J. Beckerle, James E. Hicks, and Andy Boughton

MIT and Motorola

http://www.csg.lcs.mit.edu:8001/StarT-NG
StarT-NG

- A parallel machine providing
  - Low-latency, high-bandwidth message passing
    » Extremely low overhead
    » User-level
    » Time and space shared network
  - coherent shared memory test-bed
    » Software implemented, configurable
    » Extremely simple hardware

- Used aggressive, next-gen commercial systems
  - PowerPC 620-based SMPs
  - AIX 4.1
A StarT-NG Site

Modification
- Arctic Switch Fabric
- L3 Cache Coherent Interconnect
- Memory
- I/O
- ACD

Original
- L2 $ NIU
- 620
- 620
- 620
- 620

SMU

Andy Boughton, MIT
Arctic Switch Fabric

- 32-leaf full-bandwidth fat tree
  - 200MB/sec/direction
- Differential ECL links to endpoints
- Modular, scalable design
8-Site StarT-NG
Network Interface Unit (NIU)

- 620 provides a *coprocessor interface* to L2
  - accesses to specific region of memory go directly to L2 coprocessor
    » bypass L2 cache interface
  - still cacheable within L1, if desired

- NIU attached to L2 coprocessor interface
NIU Implementation

4-32 msg buffers (4KB each)
Custom ASIC

FPGA

Dual-Ported Buffer

L2 Cache

Arctic Network

Arctic 200MB/sec/direction

L3 bus

Cache interface
128 bits @ 1/2 processor clock

Attempted Full Performance

Derek Chiou
Prototyping to Emulation
Address Capture Device (ACD)

- Allows an SMP 620 (sP) to service bus ops
  - Support shared memory
- ACD is simple hardware on L3 bus
  - “captures” global memory bus transactions
- sP communicates with ACD over L3 bus
  - Reads captured accesses to global address
  - Services requests using message passing
  - Writes back returned cache-lines to ACD
  - depends on out-of-order 620 bus
- If not needed, sP becomes an aP
ACD Example

Modification

Original

Arctic Switch Fabric

Cache Coherent Interconnect

Memory

I/O

SMU

Derek Chiou

Prototyping to Emulation
Status *(from EuroPar 95 talk)*

- Hardware & Software design completed
  - implementations in progress
- Hardware will be available soon after the 620 SMP is available
Then, in 1996, Arvind has a meeting

- PowerPC 620 indefinitely delayed
  - Look for another processor

- Lesson to current grad students
  - Don’t let Arvind go to meetings

- PowerPC 604e chosen
  - Available off the shelf
The StarT-Voyager Parallel System

Derek Chiou, Boon S. Ang, Dan Rosenband, Mike Ehrlich, Larry Rudolph, Arvind,
MIT Laboratory for Computer Science
**StarT-Voyager**

- **Scalable SMP cluster**
  - IBM 604e-based SMP building blocks
  - Custom Network Endpoint Subsystem (NES) connects SMP to network via memory bus

- **Intended Research**
  - network sharing
  - communication mechanisms
  - architecture
  - system and application software
Network Endpoint Subsystem
Why Share Network?

- Single network
- Different Services
  - message passing (MP)
  - coherence protocol
  - file system….
- Multiple processors/node
  - multiple network jobs
  - multiple services/processor
StarT-Voyager Network Sharing

Application

Infinite Queues

Gateway/Translation

Network
Multiple Queues

- Fixed number hdw queues
- Service Processor (sP) emulates infinite queues
  - sP controls/uses NES
- Critical queues use hdw queues (resident), others emulated by sP (non-resident)
- Application oblivious
  - switch queues without app knowledge or support (VM)
- Synchronization
- Flow control
Virtualized Destination

message

<table>
<thead>
<tr>
<th>Head</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>vdest</td>
<td></td>
</tr>
</tbody>
</table>

Tx Translation

Rx message queues

RxQ Translation

destination node

virtual receive queue

virtual receive queue

**Makes Migration Easy!**
Memory with Weird Semantics: Message Passing Mechanisms

- **Four mechanisms**
  - Basic Message
  - Express Message
  - Tag-on Message
  - DMA
- **512 msg queues**
  - 16 resident
- **Protected user-level access**
  - Multi-tasking (space / time)
  - No strict gang scheduling required
Express Messages

- For small messages, e.g. Acks:
  - Payload: 32 + 5 bits
- Uncached access to message queues
- Advantages:
  - Avoid weak memory model’s SYNC
  - No coherence maintenance for msg queue space
S-COMA Shared Memory

- Global mem mapped to local physical mem
  - Page granularity allocation
  - cache-line granularity protection
- Accesses to global mem snooped by NES
  - legal access completes against local RAM
  - illegal access passed to sP for servicing
    » aP bus operation retried until sP fixes
S-COMA Hardware Support

- NES hw snoops part of physical memory
- \( F(\text{Bus Operation, HAL State}) \rightarrow \text{Action} \)
  - Proceed
  - Proceed \& Forward to sP
  - Retry \& Forward to sP
- sP only entity that can modify HAL state
  - simplicity at slight restriction on functionality
Accessing S-COMA Memory
Implementation

- It worked!

- NESChip implemented in Chip Express technology
  - laser-cut gate array prototyping (1 week)

- TxU/RxU implemented in FPGA’s

- Buffers implemented by dual-ported SRAM’s and FIFO’s

- Implemented by students and staff
StarT-X/StarT-Jr

James Hoe, Mike Ehrlich
StarT-X: A Real Success

Heterogeneous Network of Workstations
StarT-X PCI-Arctic network interface
Integrated network processor
StarT-Hyades Cluster

- **Our system**
  - 16 2-way Pentium-II SMPs running Linux
  - Fast Ethernet (LAN)
  - Even faster system area network (StarT-X)
  - Owned by a single research group

- **Application: MITgcmUV**
  - Coupled atmosphere and ocean simulation for climate research
  - Traditionally relied on shared Big Irons
## Application Performance

<table>
<thead>
<tr>
<th>Processor Count</th>
<th>Machine</th>
<th>Sustained Performance (Gflop/s)</th>
<th>Normalized Performance 1-proc C90</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hyades</td>
<td>0.054</td>
<td>&lt;0.1</td>
</tr>
<tr>
<td>16</td>
<td>Hyades</td>
<td>0.9</td>
<td>1.5</td>
</tr>
<tr>
<td>32</td>
<td>Hyades</td>
<td>1.8</td>
<td>3.0</td>
</tr>
<tr>
<td>1</td>
<td>Cray C90</td>
<td>0.6</td>
<td>1.0</td>
</tr>
<tr>
<td>4</td>
<td>Cray C90</td>
<td>2.2</td>
<td>3.7</td>
</tr>
<tr>
<td>1</td>
<td>NEC-SX4</td>
<td>0.7</td>
<td>1.2</td>
</tr>
<tr>
<td>4</td>
<td>NEC-SX4</td>
<td>2.7</td>
<td>4.5</td>
</tr>
</tbody>
</table>
Modern Day:  
**RAMP: MPP on FPGAs**

- Goal 1000-CPU system for $100K early next year  
  - *Not intended to be prototype*
- $\approx 16$ CPUs will fit in Field Programmable Gate Array (FPGA)  
  - Need about 64 FPGAs  
  - $\approx 8$ 32-bit simple “soft core” RISC at 100MHz in 2004 (Virtex-II)
- HW research community shares logic design (“gate shareware”) to create out-of-the-box, MPP  
  - Use off-the-shelf processor IP (simple processors, $\sim 150$MHz)
  - RAMPants: **Arvind** (MIT), Krste Asanovíc (MIT), Derek Chiou (Texas), James Hoe (CMU), Christos Kozyrakis (Stanford), Shih-Lien Lu (Intel), Mark Oskin (Washington), David Patterson (Berkeley, Co-PI), Jan Rabaey (Berkeley), and John Wawrzynek (Berkeley, PI)
- “Research Accelerator for Multiple Processors”
RAMP-White Reference Platform

- Very flexible shared memory platform
  - Different components/policies/parameters
- Uses StarT-Voyager-like bus retry
- 3 Phase Approach:
  » Phase 1: Incoherent global shared memory
    - All accesses to main memory
    - No caches
  » Phase 2: Snoopy-based coherency over a ring
    - Adds coherent cache
  » Phase 3: Directory-based coherency over network
    - Adds directory
RAMP-White Phases

- Processor
- Memory Controller (MC)
- Network Interface (NIU)
- Intersection Unit (IU)
- IO & Platform Devices

Ring Network
Intersection Unit (in Bluespec)
Conclusions

- Ideas recycle
  - RAMP-White \(\approx\) StarT-Voyager

- Don’t be too implementation-ambitious
  - Matching industry is impossible
  - Balance between implementation effort and accuracy

- Delicate balance between rolling your own and depending on others
  - Reuse whatever you can (Arctic)

- Thanks Arvind!
  - Using what I learned in grad school daily
  - Bluespec