

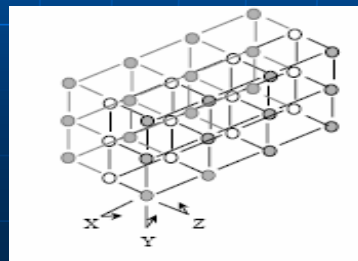
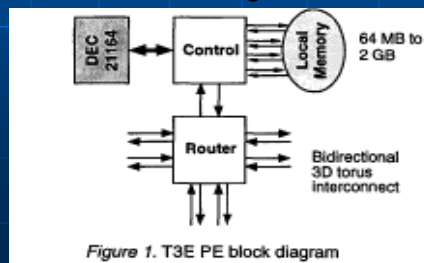
# Synchronization and Communication in the T3E Multiprocessor

Steven L. Scott  
Cray Research, Inc

Presented by  
Hari Sivaramakrishnan

## T3E Features

- Distributed shared memory system
  - Up to 2GB memory per processor
  - DEC Alpha 21164 processor
  - **Shell** – control and router chips, memory



## T3E Features

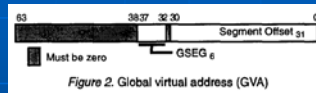
- Buffering
  - Buffers can detect multiple interleaved streams
- Local memory cached
  - No onboard cache
  - External backmap to maintain data consistency
- E-Registers
  - 512 user + 128 system
  - Remote communication and synchronization
  - Highly pipelined
  - Extend the processor's physical address space

## Global Communication

- Operations performed on E-Registers
  - Direct loads, stores between E-registers and processor registers
  - Global operations (message passing, synchronization, remote loads)
- Global references
  - Global Virtual Address (GVA)

# Address Translation

- Global Virtual Address (GVA)



- Virtual PE number

- Centrifuge
  - Mask, index, base

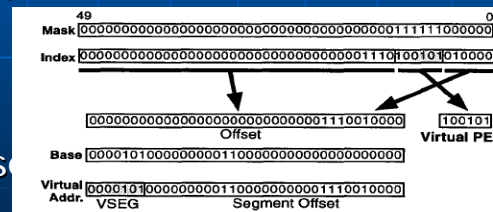
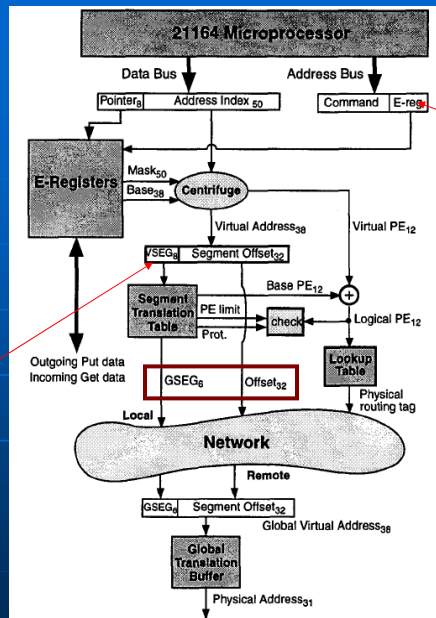


Figure 3. HW centrifuge operation example: Array interleaved by cache line over 64 PEs



Source or destination

Should be only 6 bits, not 8

# Get and Put operations

- Reads and writes to global E-Registers
  - Single word or a vector
- Flags on each register for synchronization
  - Empty
  - Full
  - Memory to memory copy through E-registers
    - Does not touch processor bus
  - No RAW hazards
- Highly pipelined
  - 256 bytes in 26.7ns can be issued
  - Large number of E-registers
  - Max transfer rate = 480MB/s between two nodes

# Atomic Memory Operations

- T3D used dedicated SWAP registers
- T3E uses memory locations

Atomic Operation (operands)	Description
Fetch_&_Inc (none)	Add one to memory location and return original memory contents.
Fetch_&_Add (addend)	Add integer addend to memory location and return original memory contents.
Compare_&_Swap (comperand, swaperand)	If comperand equals contents of memory, then store swaperand into memory. Return original contents of memory.
Masked_Swap (mask, swaperand)	For each bit set in mask, store corresponding bit of swaperand into memory. Return original contents of memory.

Universal construct

Table 1. Atomic Memory Operations

## How to perform an AMO?

- Operands written to E-registers
- Store to I/O space to trigger operation
- Atomic Memory Operation packet sent to particular memory location
- Result returned to E-Register specified on the address line
- Most AMOs need a read-modify-write of RAM
  - 11 sysclocks at 147ns per clock
  - 8M operations per second
- High bandwidth fetch\_and\_inc served out of buffer at memory controller for each node

## Messages

- T3D
  - Single hardware message queue for user and system messages
  - Every message generates an interrupt
- T3E
  - Arbitrary number of message queues
  - Mapped to memory
  - Queue max size = 128 MB. Message size = 64 bytes
- Message notification
  - Always interrupt
  - Never interrupt (polling)
  - Interrupt on a threshold
- Message passing and shared memory integration

# Message Queue Control Word

- Descriptor for a message queue
- Messages rejected when queue is full
- If message insertion creates a segmentation violation, **nack** is returned

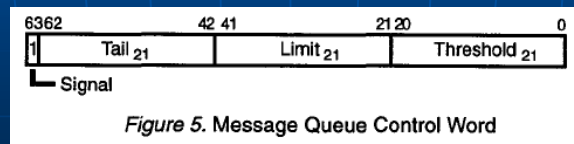


Figure 5. Message Queue Control Word

# Sending Messages

- Messages assembled in an aligned block of 8 E-Registers
- Sent to address of MOCW
- MOCW updates and message storage are atomic
- E-Registers status is set to empty on send
  - If message accepted, changed to full
  - If message rejected, changed to full-send-rejected

# Barrier/Eureka Synchronization

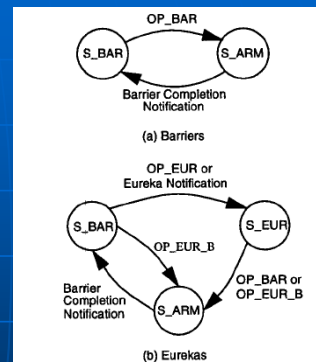
- Barrier
  - Wait for **all** processors to signal an event
- Eureka
  - Wait for **some** processor to signal an event
- Barrier/Eureka Synchronization units
  - 32 BSUs
  - Memory-mapped
  - Set of processors given access to a BSU

State	Description
S_EUR	A eureka event came
S_EUR_I	A eureka came, interrupt signalled
S_ARM	Barrier is armed
S_ARM_I	Barrier is armed, an interrupt will occur on completion
S_BAR	Barrier just completed
S_BAR_I	Barrier just completed, interrupt signalled

Table 2. Barrier/Eureka Synchronization Unit States

Operation	Description
OP_EUR	Send eureka
OP_INT	Set to interrupt when a eureka event occurs
OP_BAR	Arm Barrier
OP_BAR_I	Arm Barrier, interrupt on completion
OP_EUR_B	Send eureka and arm barrier

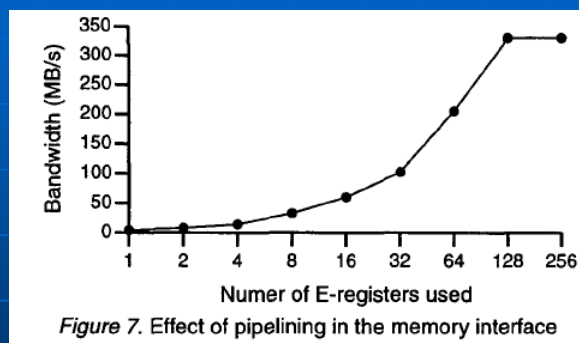
Table 3. Barrier/Eureka Synchronization Unit Operations



## Barrier/Eureka Trees

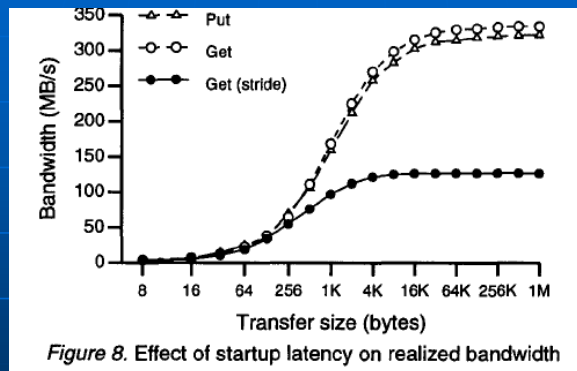
- Barrier/Eureka network embedded in torus interconnect
  - Keeps latency lower than a remote reference
- Network router has a register for each BSU
  - Node can be configured as internal in BSU tree
  - Information about which of six network directions is the parent
- Eureka and Barrier notifications are sent to the parent nodes
- Completions are broadcast hierarchically

## Performance – E-Registers



Bandwidth increases with the number of E-registers used. Ultimately, at 128 registers, the E-register control logic becomes a bottleneck.

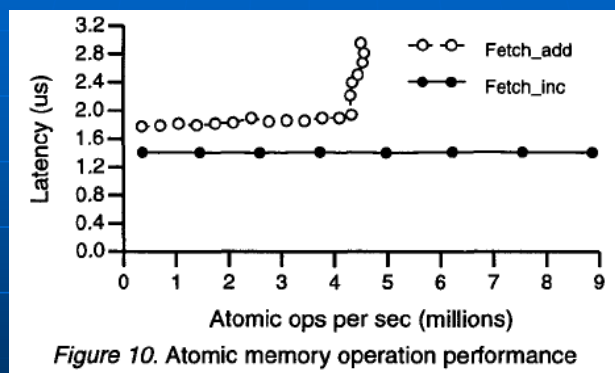
# Performance – Startup Latency



Stride 10

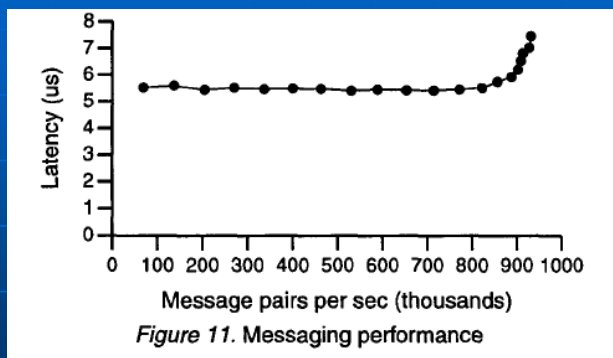
When transfer size is small, startup time becomes significant. This fades away as the number of bytes transmitted increases to a big enough number.

# Performance - AMO



Fetch\_add has bigger packet sizes, and fetch\_inc has buffering at memories

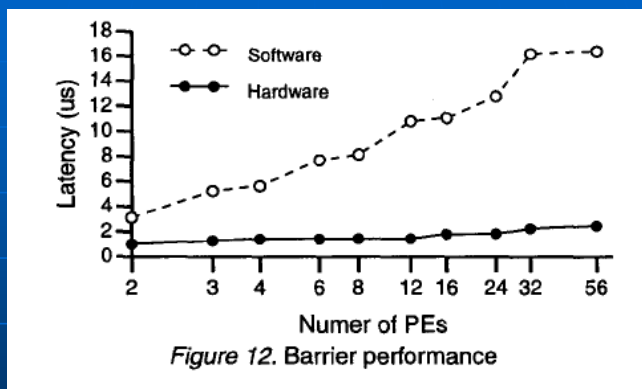
## Performance - Messaging



Processors 1 through 15 exchange messages with processor 0.

Maximum exchange rate = 932M/s

## Performance - Barriers



Average time to perform a global barrier over 50 consecutive barriers

## Summary of the T3E

- E-Registers
  - Extend memory address space
  - Pipelining of memory references
  - Used in communication and synchronization
- Messaging implemented as shared memory at user level
- A range of atomic synchronization constructs
- Hardware barrier outperforms software barrier by a factor of 7
  - Free because packets over normal interconnect are used
- Remote memory access has a greater startup penalty in T3E than in the T3D

## Questions?

