ECE 669

Parallel Computer Architecture

Lecture 11

Static Routing Architectures

Outline

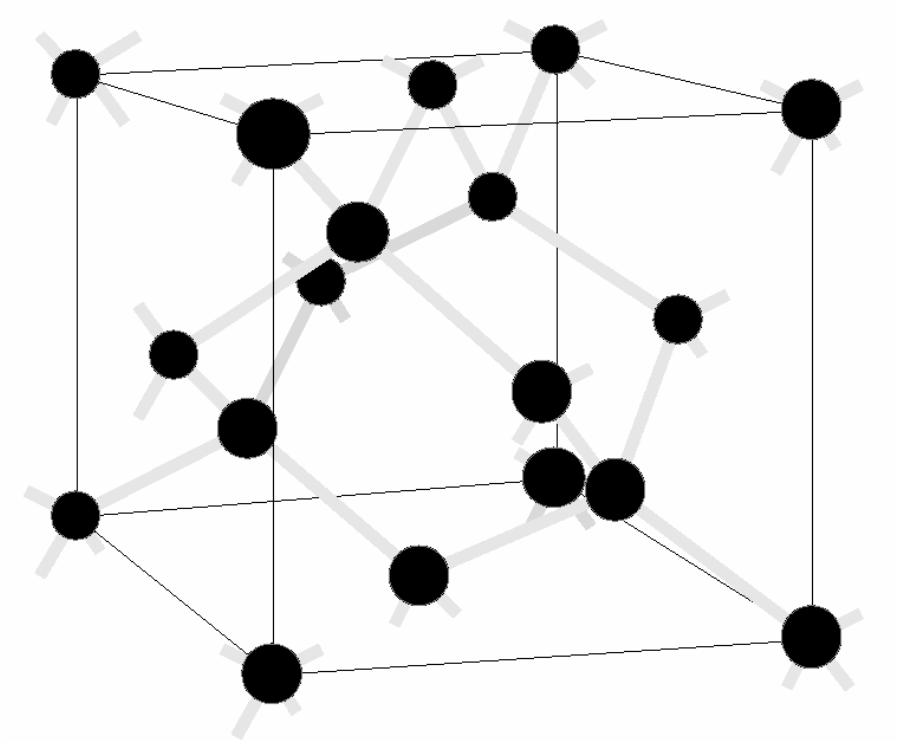
- **Programming Models**
  - Data Parallel
  - Shared Memory
  - Message Passing
- **Communication requirements**
  - Examining the network
  - Available bandwidth
  - Run-time versus compile-time
- **Models of communication**

Communication Approaches

- Circuit switched
- Store and Forward
  - On-line (dynamic routing)
  - Off-line (static routing)
- Special purpose architectures created for static routing
- Schedule all communication at compile time
- Can lead to faster overall communication (no headers)
- Can reduce congestion
- Doesn’t handle data dependency well

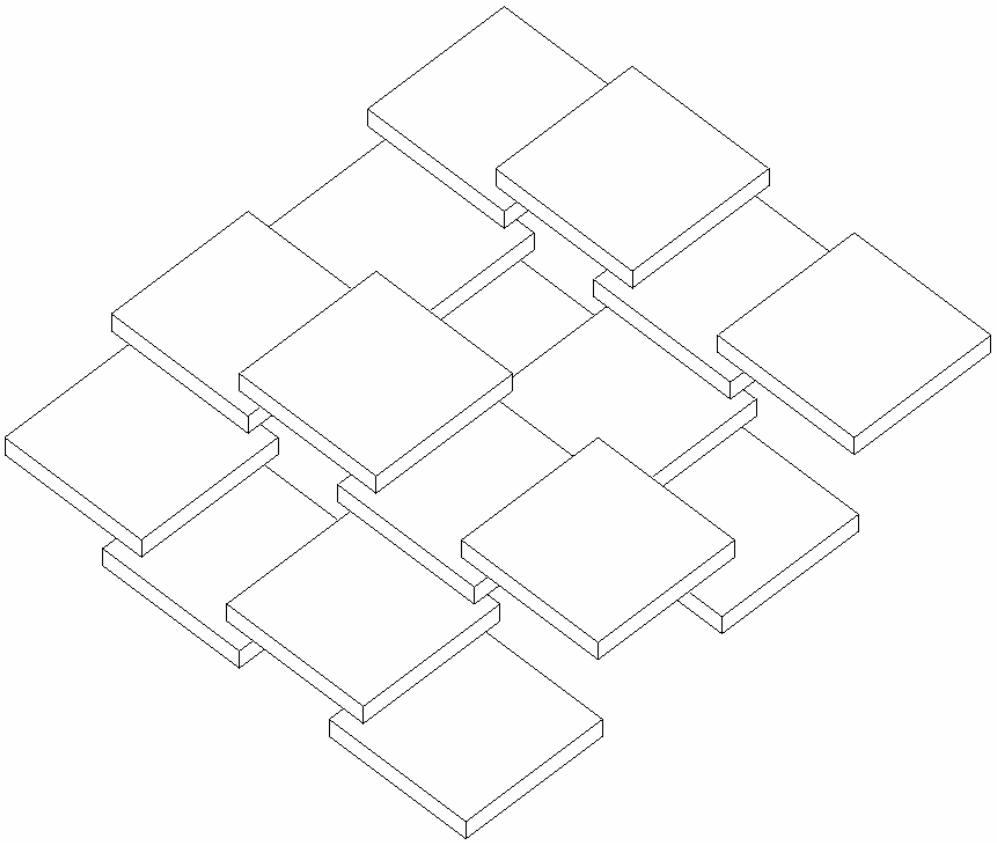
Interconnection Topology

- **Diamond lattice has desirable structure**
- **Each node has four neighbors**
- **Space filling – nodes can be packed close together**
- **Can embed other topologies**



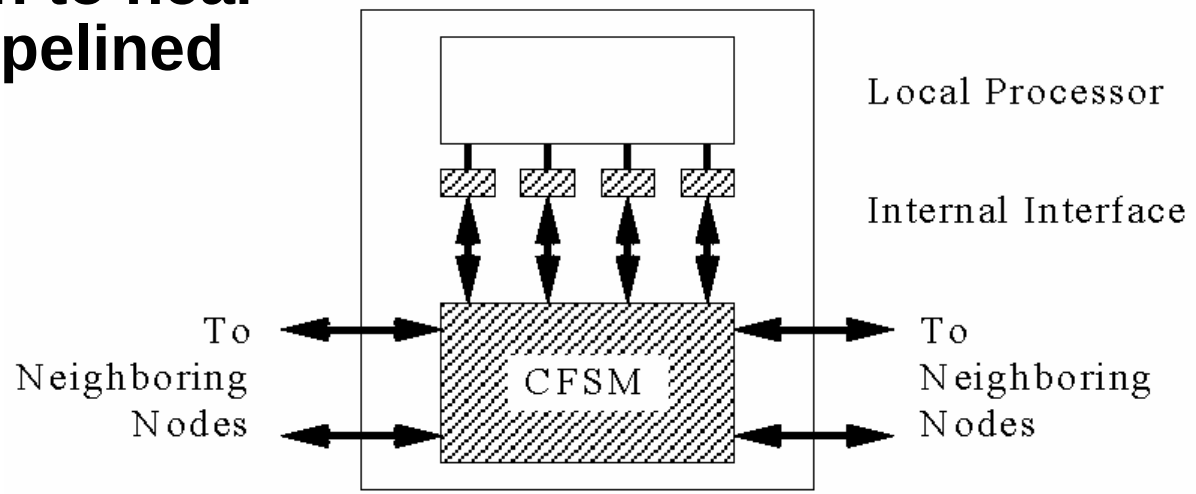
Interconnection Topology

- **Need to implement in three dimensions**
- **Bottom and top of circuit boards have connectors**
- **A node can *configure* its neighbors**



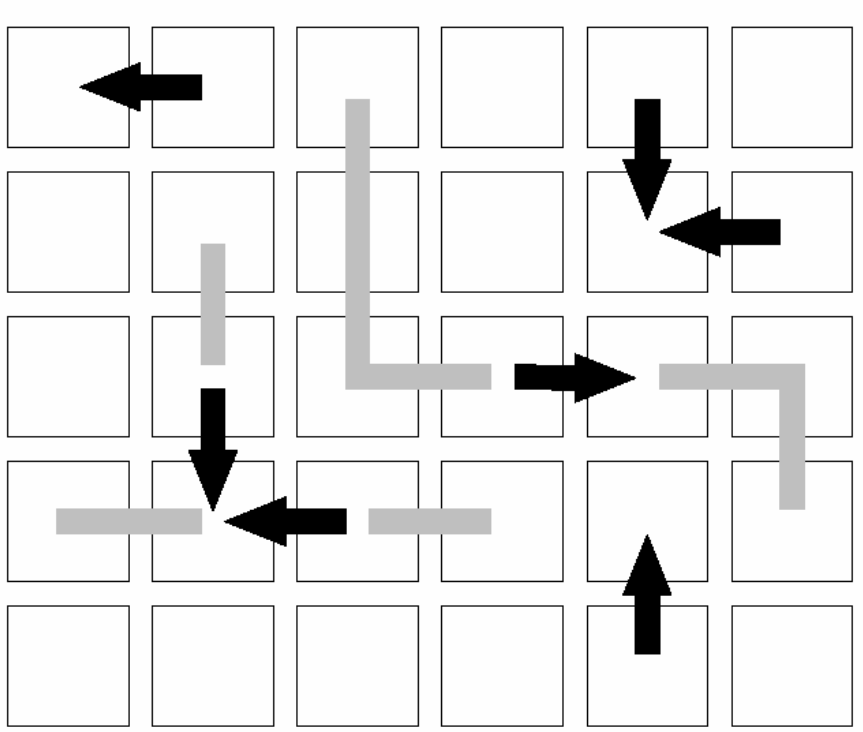
Communication Finite State Machine

- **Each node has a processing part and a communications part**
- **Interface to local processor is a FIFO**
- **Communication to near-neighbors is pipelined**



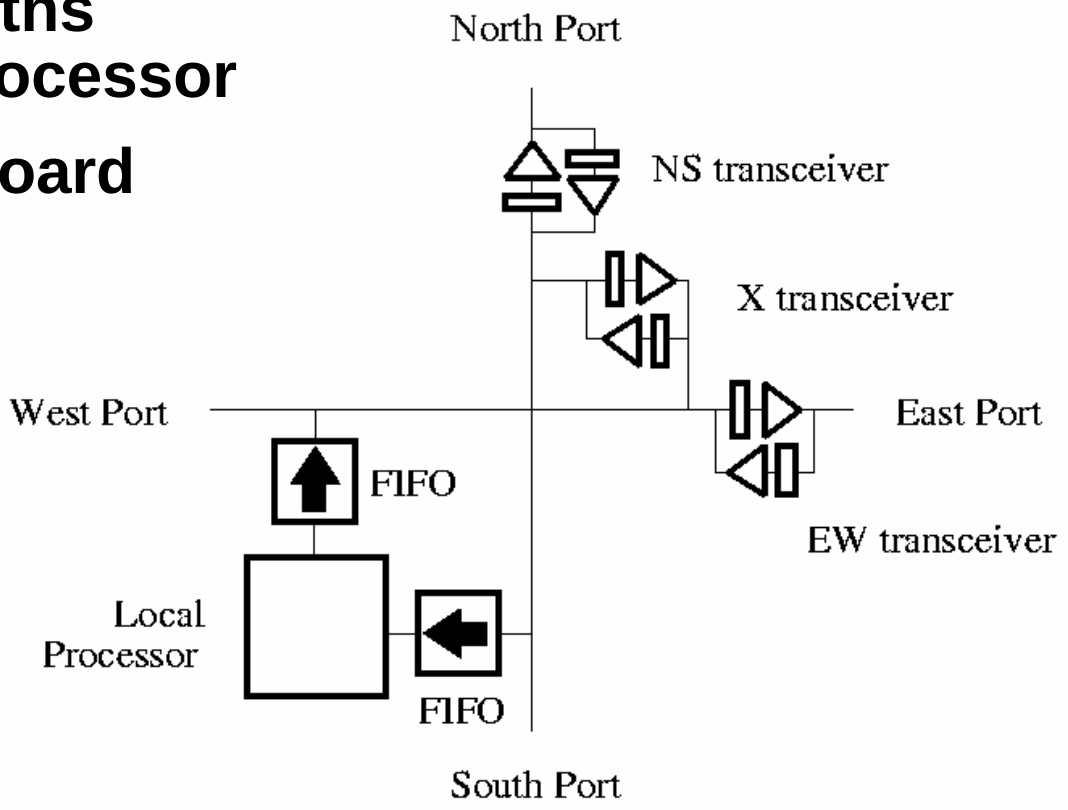
Statically Programmed Communication

- Data transferred one node in one cycle
- Inter-processor path may require multiple cycles
- Heavy arrows represent local transfers
- Grey arrows represent non-local transfers

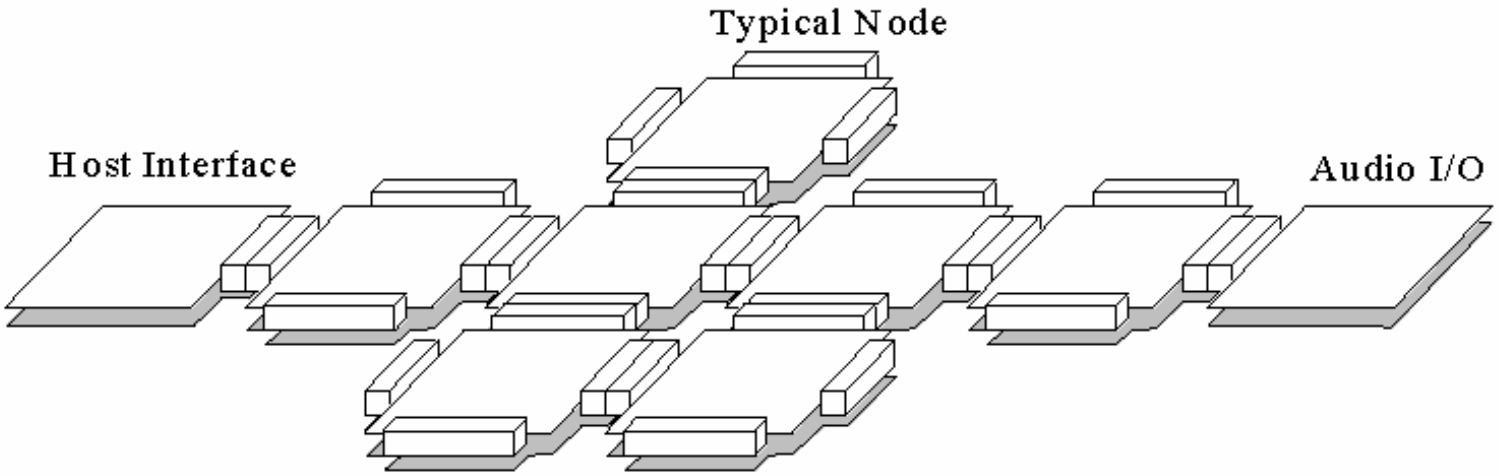


Prototype NuMesh Node - CFSM

- **Transceivers used to buffer inter-node data**
- **FIFOs buffer paths to/from local processor**
- **One node per board**

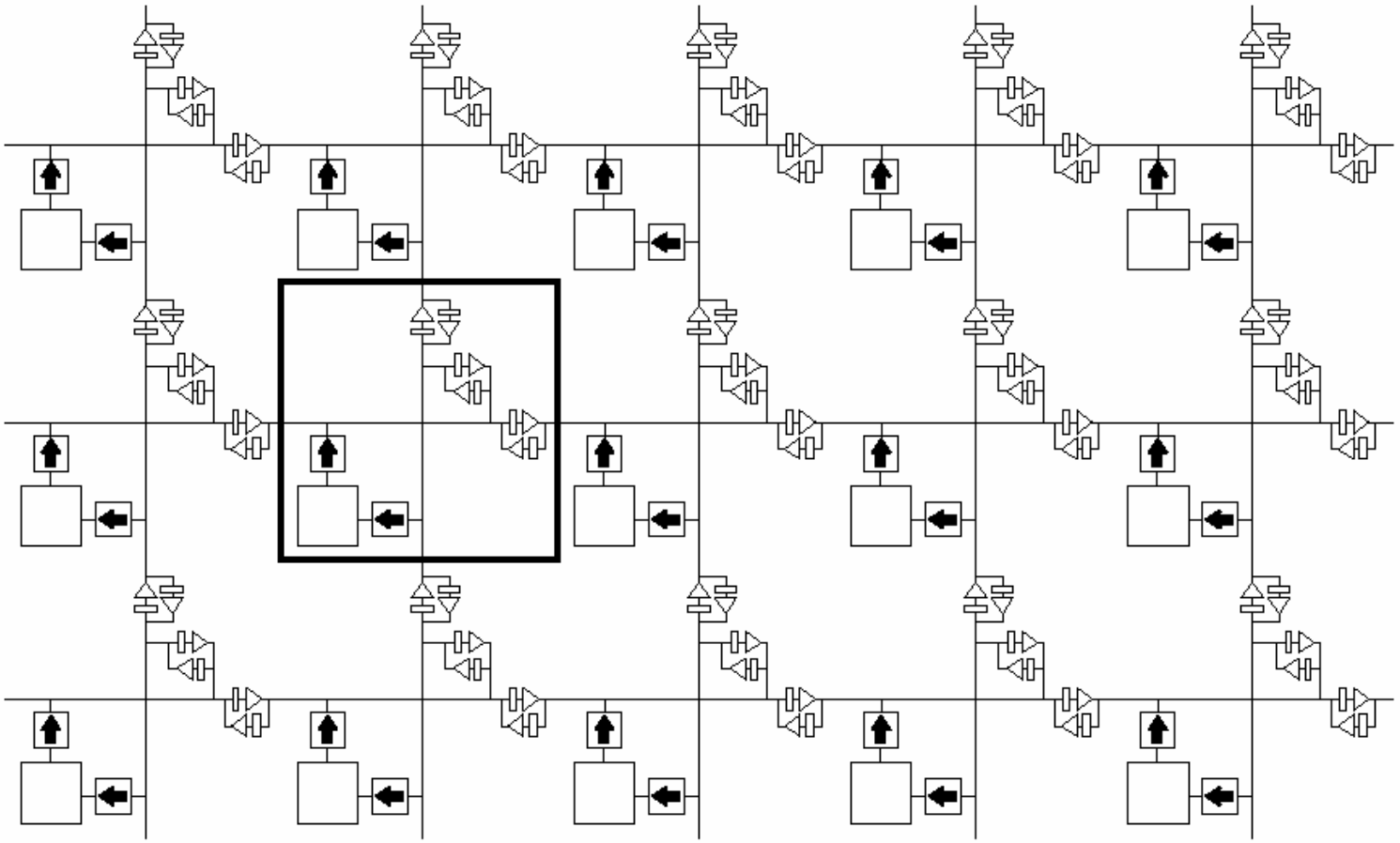


Prototype NuMesh System



- **Initial topology was a mesh**
- **Some nodes in the mesh could be unpopulated**
- **Special-purpose nodes could be populated along the system periphery**

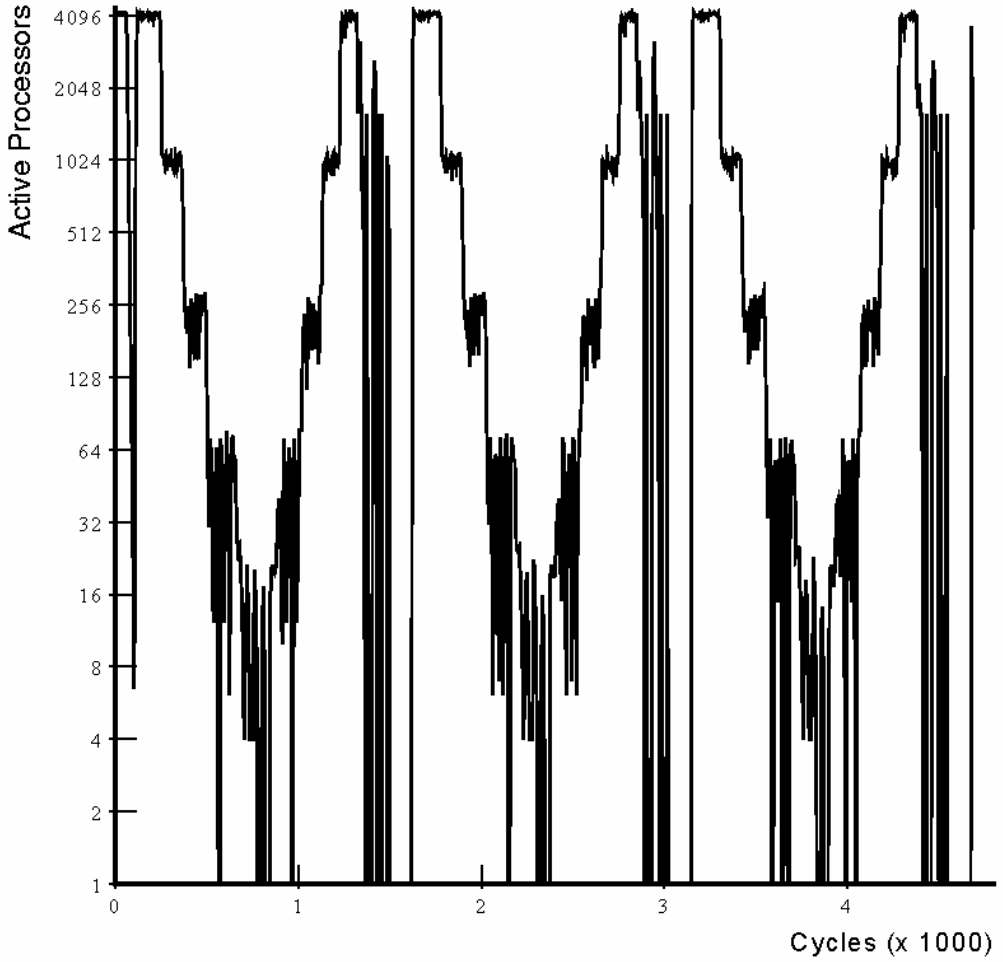
NuMesh Parallelization



- System appears like a two dimensional pipeline
- FIFOs allow processor to run at different speeds
- Rational clocking allows clocks to be distributed

NuMesh Multigrid Results

- **Multigrid is hierarchical**
- **Processor utilization indicates periodic reduced activity**
- **All communication is scheduled statically**



NuMesh Summary

- Communication determined at compile time
- Fast near-neighbor communication
- Diamond lattice provides routing benefits
- Appropriate for applications like multi-grid

Key Issues

- **Communication**
  - **Broadcast, near neighbor, tree**
- **Synchronization**
  - **Producer-consumer, barrier, locks**
- **Partitioning**
  - **Grain-size - Division of work - What to run as thread**
    - Mapping - Where to run
- **Scheduling**
  - **When to run**

° **Various computing styles differ in how the above are supported:**

- **Whether hardware support is provided**
- **Whether programmer deals with it**
- **Whether it is ignored**

° **Key: Previous machines focused heavily on hardware -once software enters the picture, distinctions become hard to make**

Historically

- **Build the machine - (paper wt.?)**
- **Low level programming --- some use**
- **Better abstractions --- much better**
  - **All programming**
  - **Low-level performance hacks**
  - **Body of theory**
    - (Low-level machine style pervades every higher level, even theory!)
- **Low-level machine organization clearly visible 'exploited' at higher levels!**
- **Sometimes machines evolve**
  application .....> machine
  (or language)

Another more common evolutionary approach...

° **Language** **Machine**

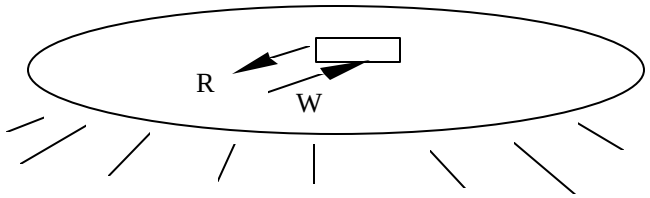
° **Fortran, C, ...**

Shared memory

$$a[i, j] = b[i, j]$$

- **View: a, b "reside" somewhere**
- **Perform operations and store values back**
- **Notion of 'location'**
- **Specify ops that can go on in parallel**

° **Algorithmic model PRAM**



PRAMS

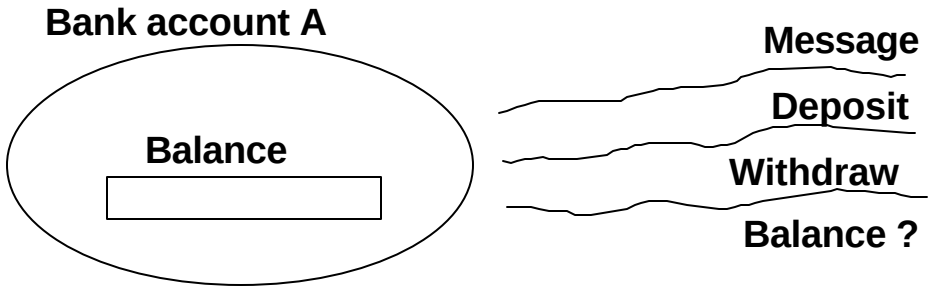
-CRCW
-CREW
-EREW

Variants **Processes**

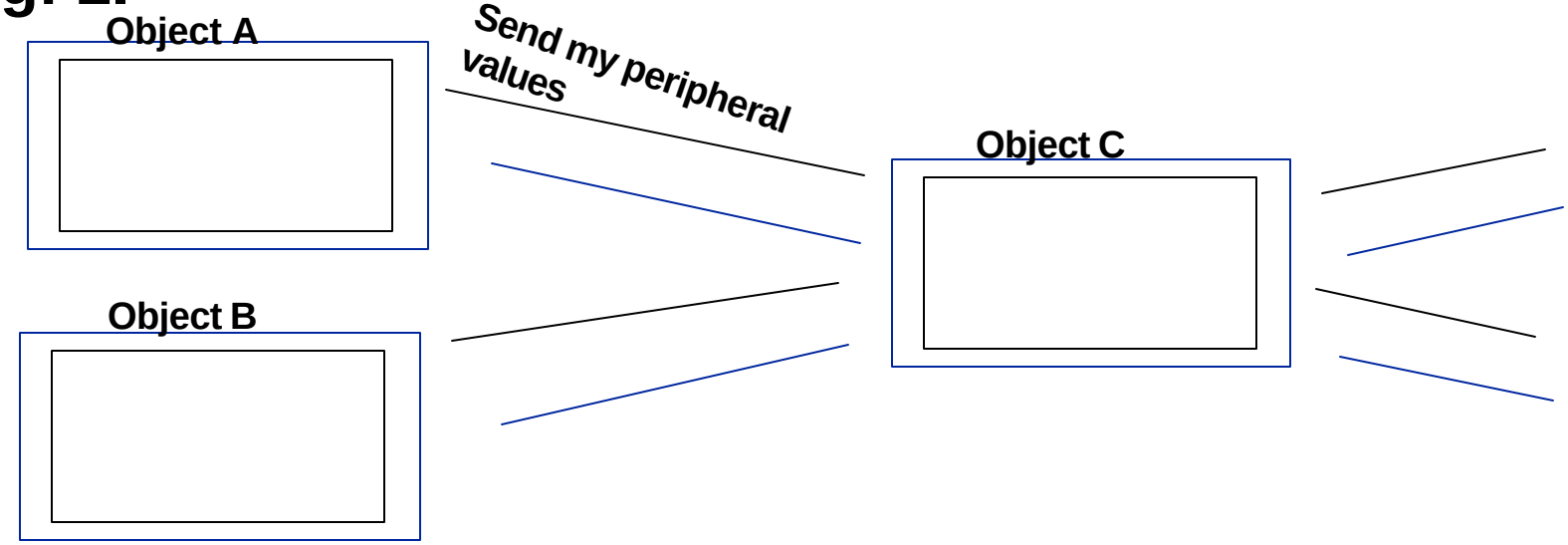
– Multiple simultaneous R,W
– Exclusive writes only
– Exclusive R & W

Object-oriented Programming
Smalltalk, variants of Scheme, C++

- **Message-Passing Machines**
- **Eg: 1.**

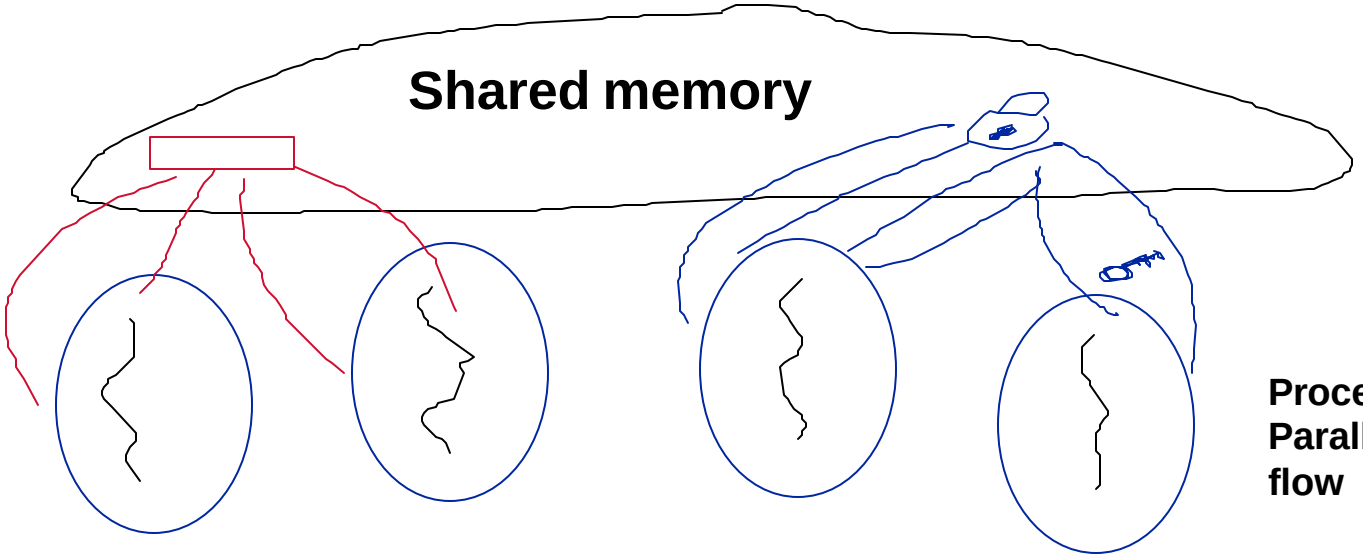


- **Eg: 2.**



- **Jacobi Relaxation**

Shared-memory style



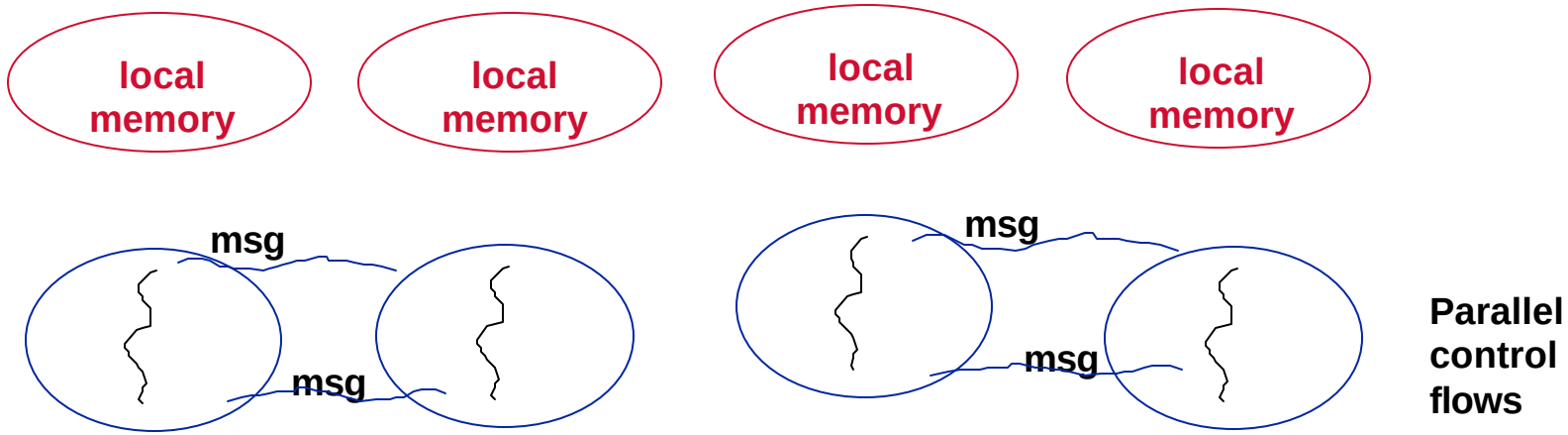
Processes - Parallel control flow

Communication via memory

Synchronization via memory

- **Partitioning - User - Coarse-fine**
- **Scheduling - System - Dynamic**

Message-passing style



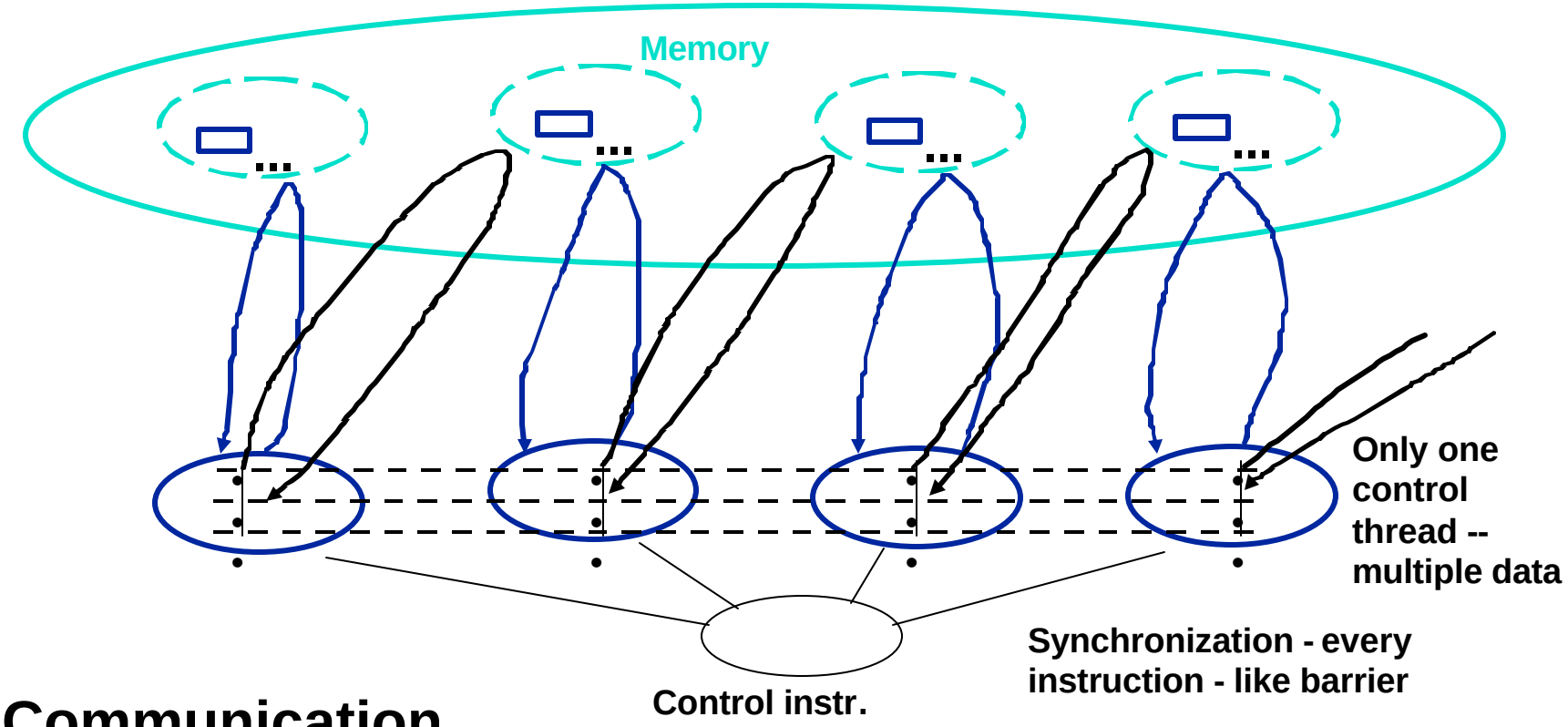
Communication via messages

Synchronization via messages

Partioning: User -- coarse

Scheduling: System -- dynamic

Data Parallel

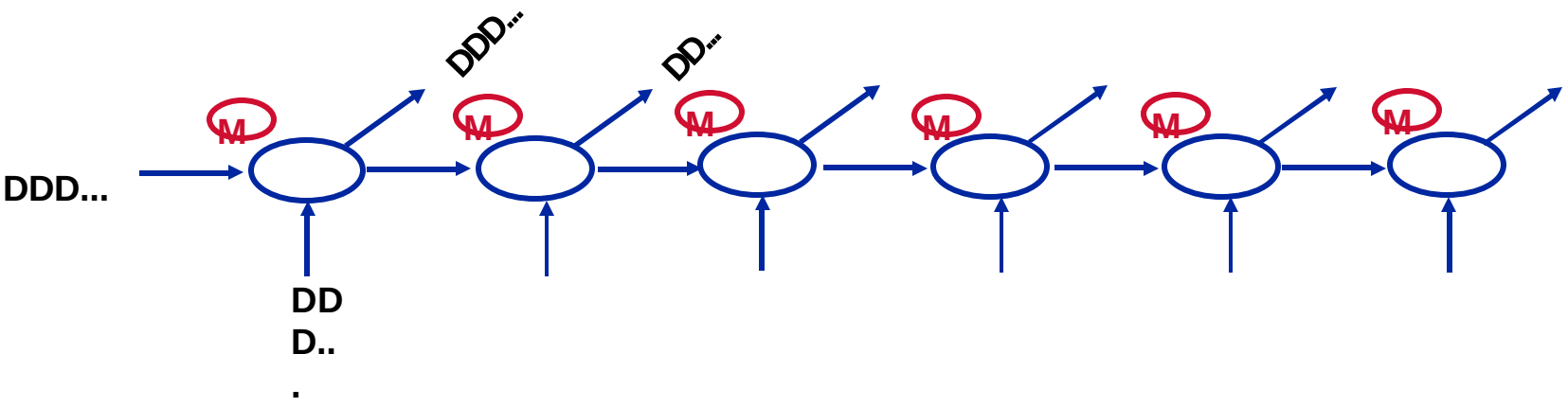


Communication

Partitioning: **Fine-grain - System**

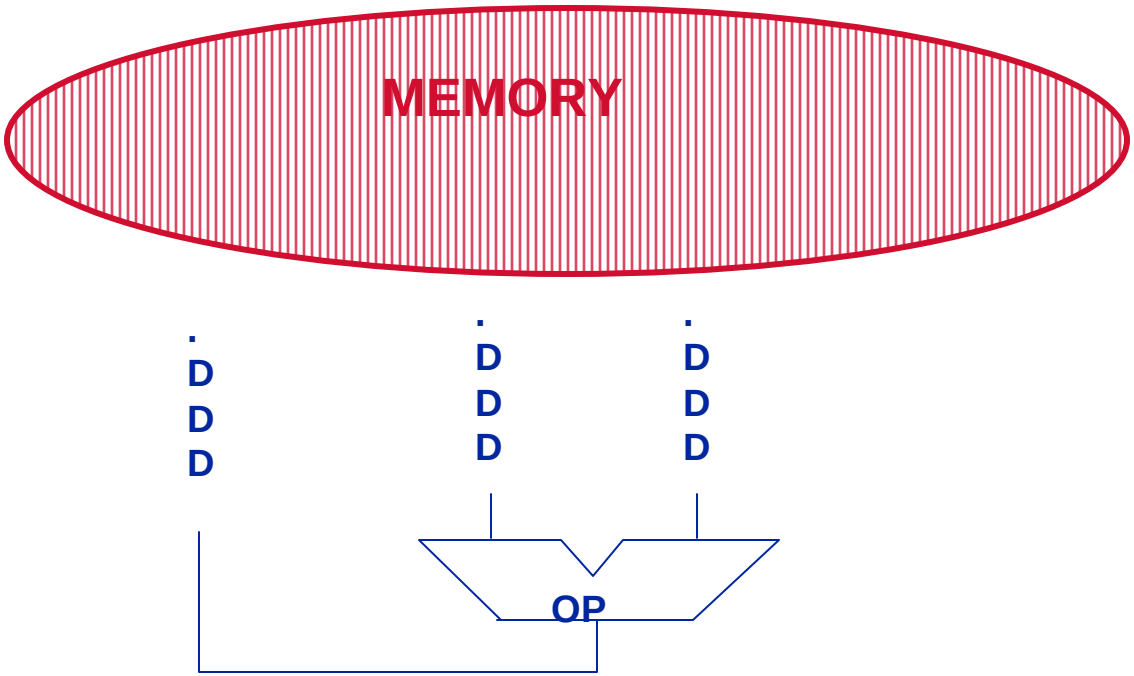
Scheduling: **User - Static**

Systolic



- **Communication:**
  - Data values
- **Synchronization:**
  - Completely static (none)
  - Pre-compiled

Vector



- Similar to data parallel
  - Only 1 processor (chaining?)
  - But exploits data parallelism