

CMSC 611: Advanced Computer Architecture

Parallel Computation

Parallel Computers

- Definition: “A parallel computer is a collection of processing elements that cooperate and communicate to solve large problems fast.”
 - Almasi and Gottlieb, Highly Parallel Computing, 1989
- Parallel machines are expected to have a bigger role in the future since:
 - Power & heat problems grow for big/fast processors
 - Even with double transistors, how do you use them to make a single processor faster?
 - CPUs are being developed with an increasing number of cores

Questions about parallel computers:

- How large a collection?
- How powerful are processing elements?
- How do they cooperate and communicate?
- How are data transmitted?
- What type of interconnection?
- What are HW and SW primitives for programmers?
- Does it translate into performance?

Level of Parallelism

- Bit-level parallelism
 - ALU parallelism: 1-bit, 4-bits, 8-bit, ...
- Instruction-level parallelism (ILP)
 - Pipelining, Superscalar, VLIW, Out-of-Order execution
- **Process/Thread-level parallelism**
 - Divide job into parallel tasks
- Job-level parallelism
 - Independent jobs on one computer system

Applications

- Scientific Computing
 - Nearly Unlimited Demand (Grand Challenge):
 - Successes in some real industries:
 - Petroleum: reservoir modeling
 - Automotive: crash simulation, drag analysis, engine
 - Aeronautics: airflow analysis, engine, structural mechanics
 - Pharmaceuticals: molecular modeling

App	Perf (GFLOPS)	Memory (GB)
48 hour weather	0.1	0.1
72 hour weather	3	1
Pharmaceutical design	100	10
Global Change, Genome	1000	1000

Commercial Applications

- Transaction processing
- File servers
- Electronic CAD simulation
- Large WWW servers
- WWW search engines
- Graphics
 - Graphics hardware
 - Render Farms

Framework

- Extend traditional computer architecture with a communication architecture
 - abstractions (HW/SW interface)
 - organizational structure to realize abstraction efficiently
- Programming Model:
 - Multiprogramming: lots of jobs, no communication
 - Shared address space: communicate via memory
 - Message passing: send and receive messages
 - Data Parallel: operate on several data sets simultaneously and then exchange information globally and simultaneously (shared or message)

Communication Abstraction

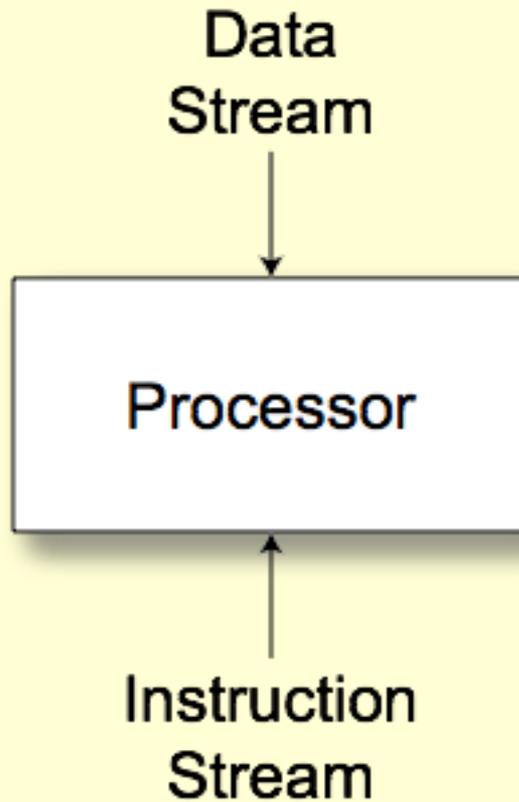
- Shared address space:
 - e.g., load, store, atomic swap
- Message passing:
 - e.g., send, receive library calls
- Debate over this topic (ease of programming, scaling)
 - many hardware designs 1:1 programming model

Taxonomy of Parallel Architecture

- Flynn Categories
 - **SISD** (Single Instruction Single Data)
 - **MISD** (Multiple Instruction Single Data)
 - **SIMD** (Single Instruction Multiple Data)
 - **MIMD** (Multiple Instruction Multiple Data)

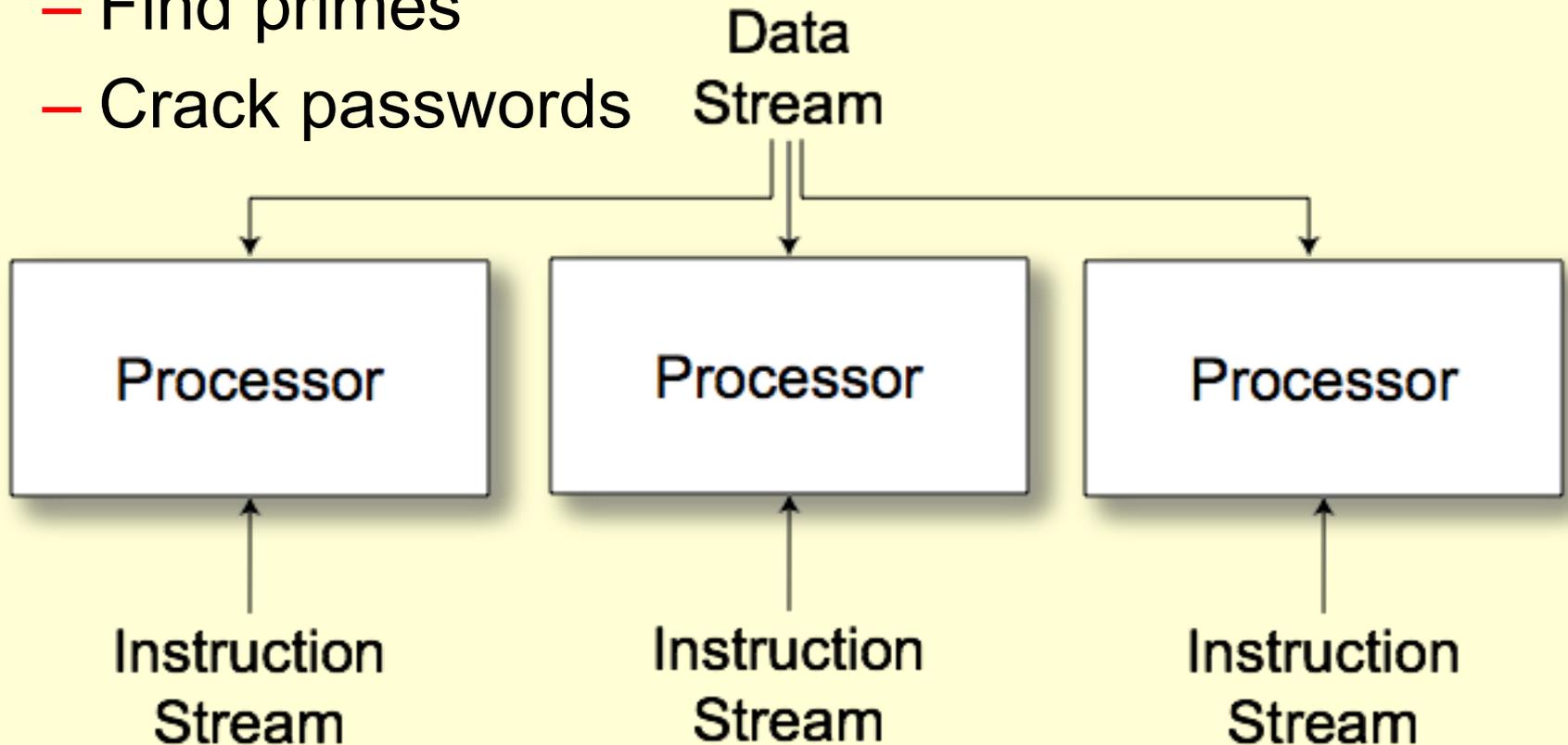
SISD

- Uniprocessor



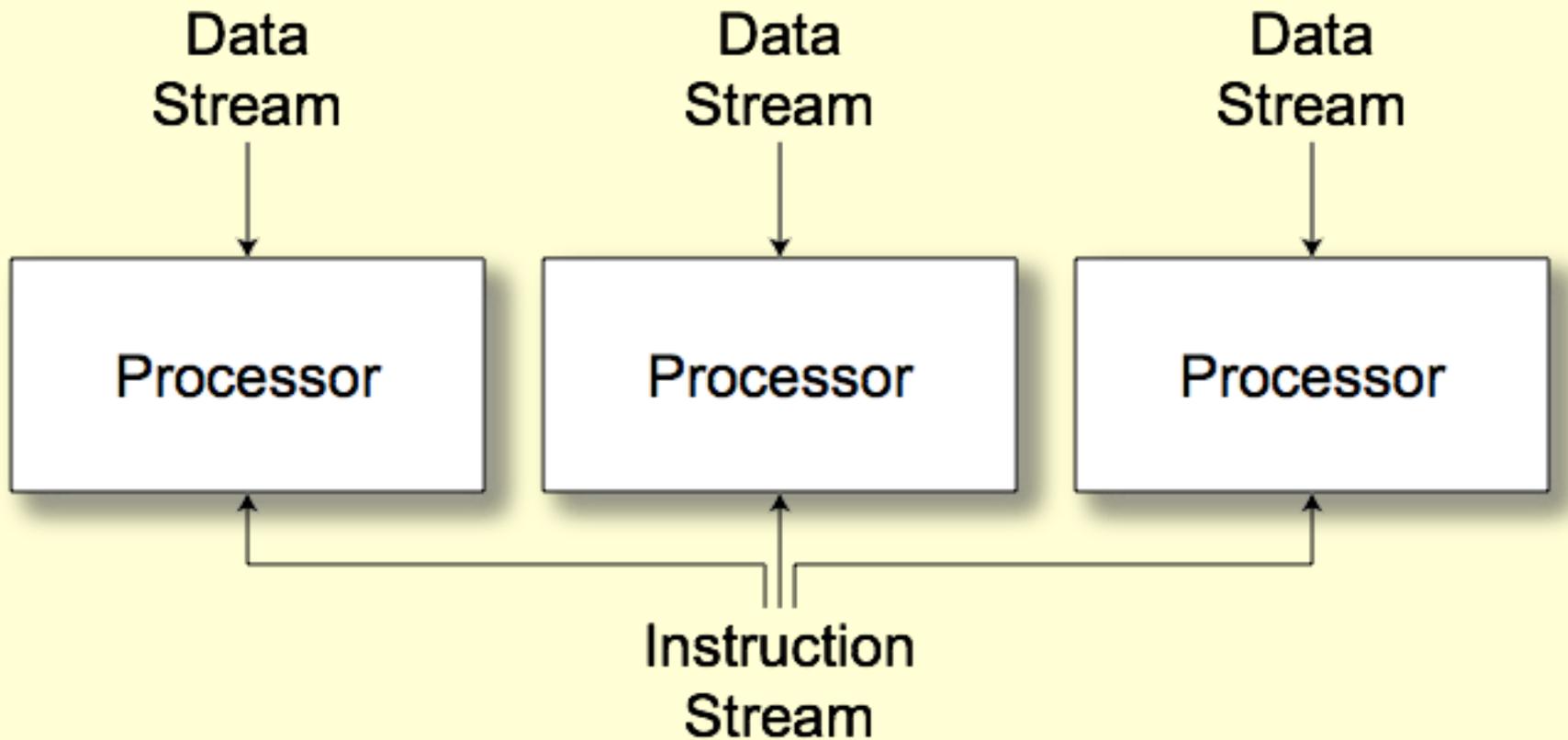
MISD

- No commercial examples
- Different operations to a single data set
 - Find primes
 - Crack passwords



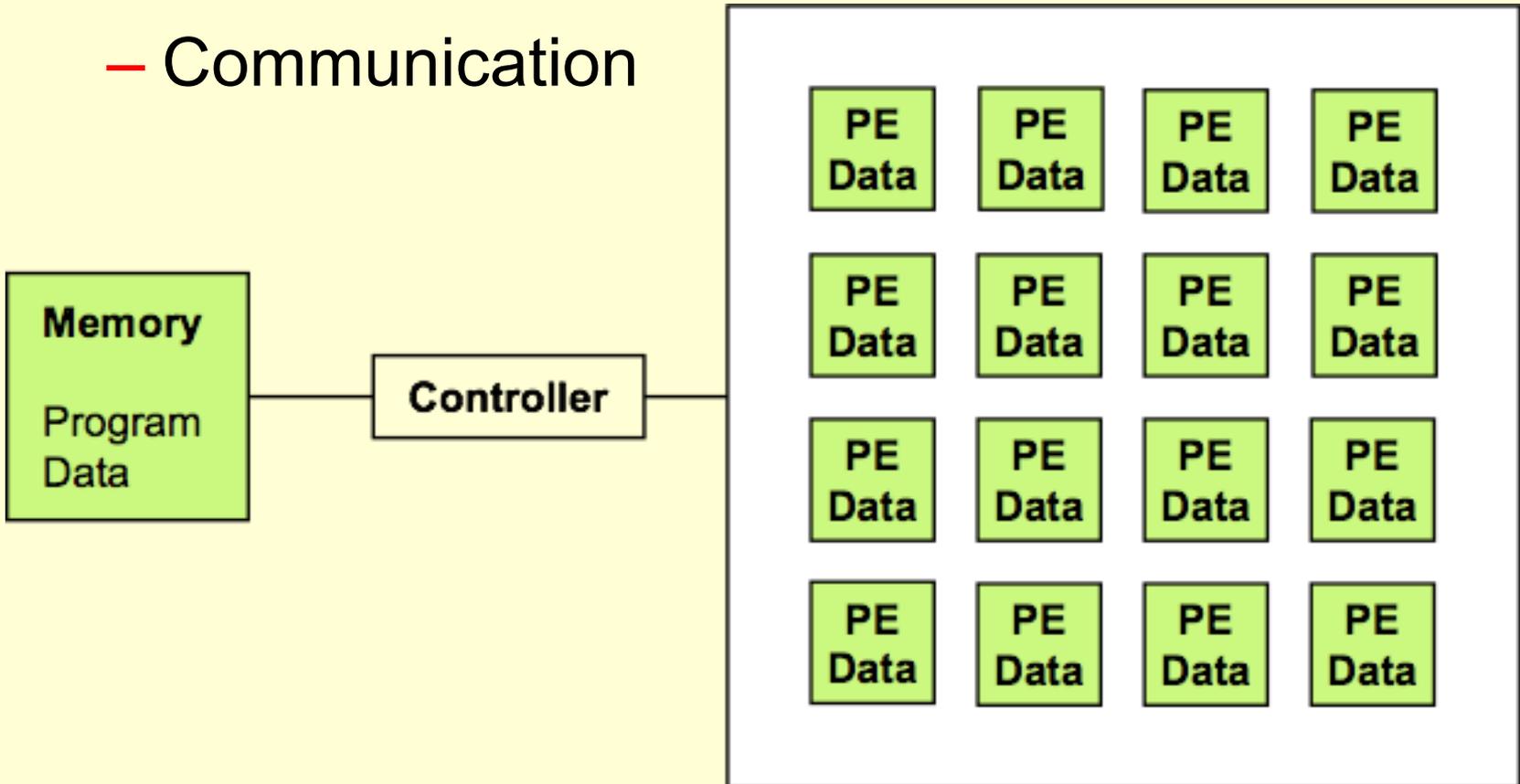
SIMD

- Vector/Array computers



SIMD Arrays

- Performance keys
 - Utilization
 - Communication

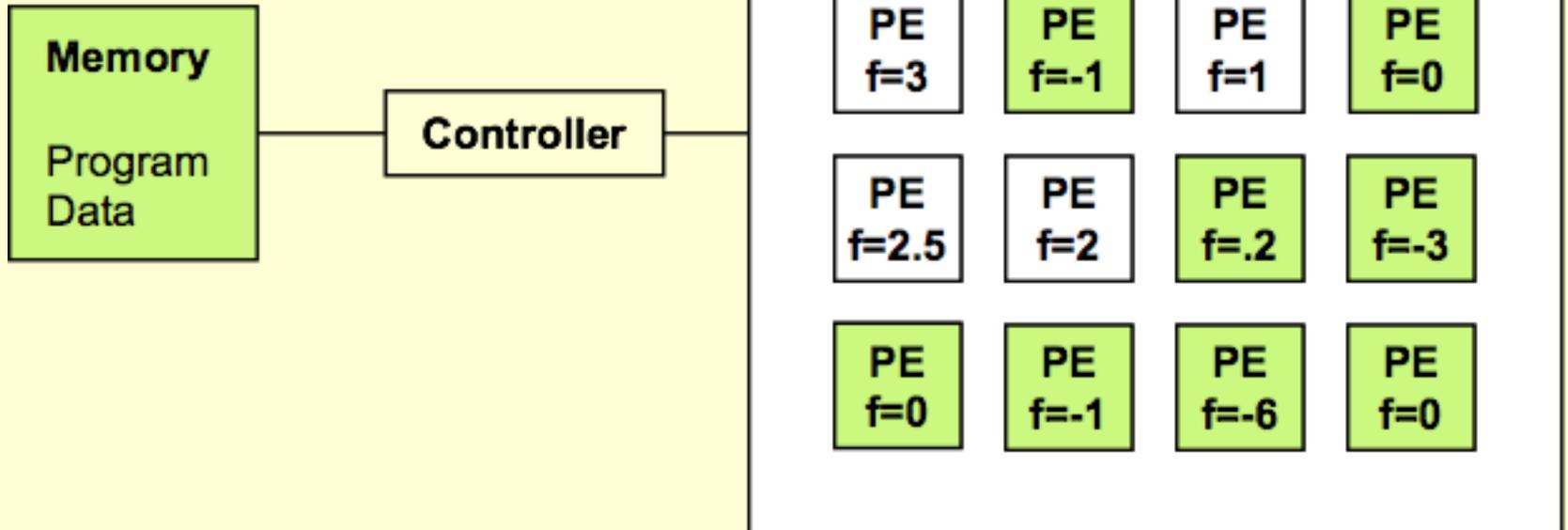


Data Parallel Model

- Operations performed in parallel on each element of a large regular data structure, such as an array
 - One Control Processor broadcast to many processing elements (PE) with condition flag per PE so that can skip
- For distributed memory architecture data is distributed among memories
 - Data parallel model requires fast global synchronization
 - Data parallel programming languages lay out data to processor
 - Vector processors have similar ISAs, but no data placement restriction

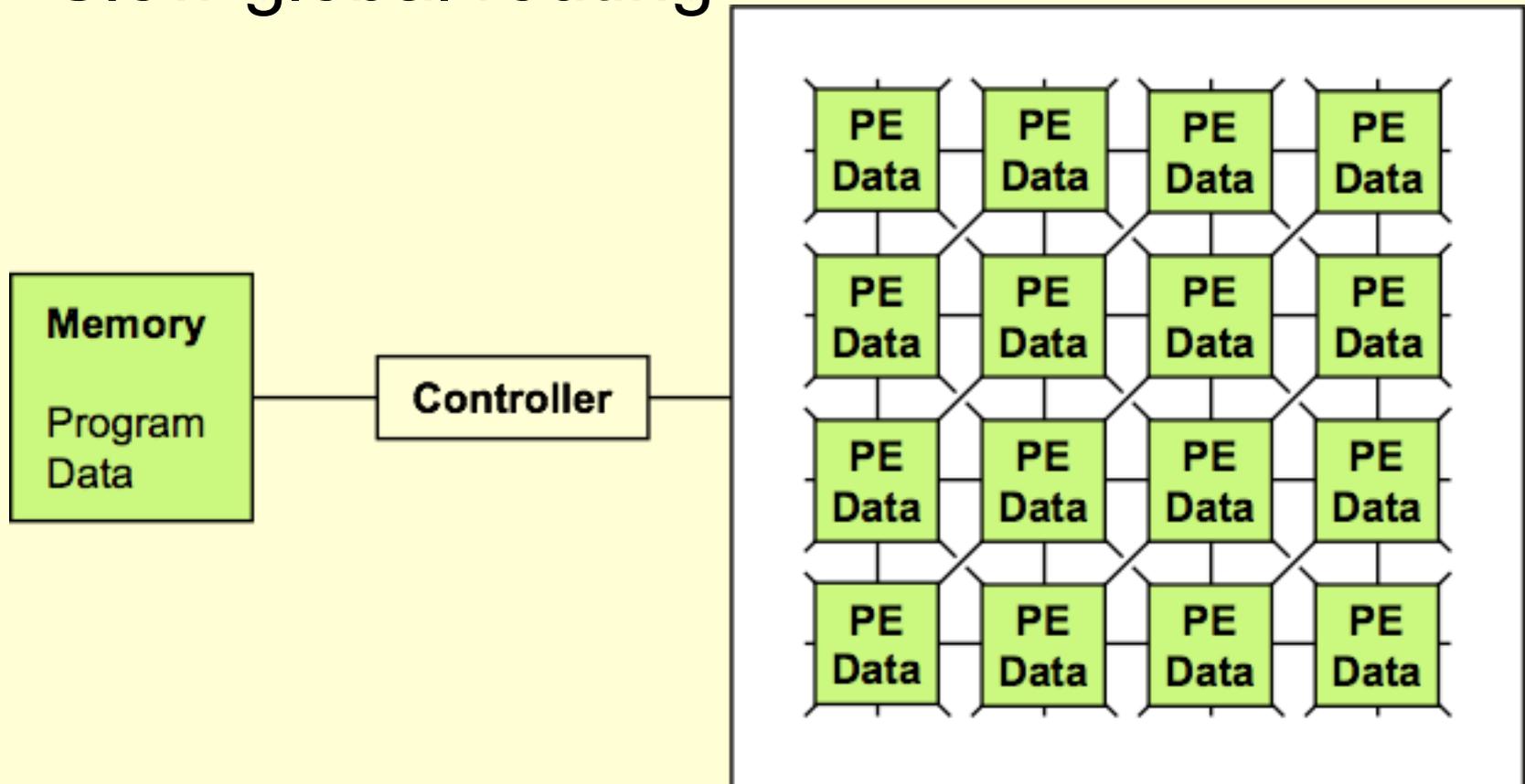
SIMD Utilization

- Conditional Execution
 - PE Enable
 - if ($f < .5$) {...}
 - Global enable check
 - while ($t > 0$) {...}



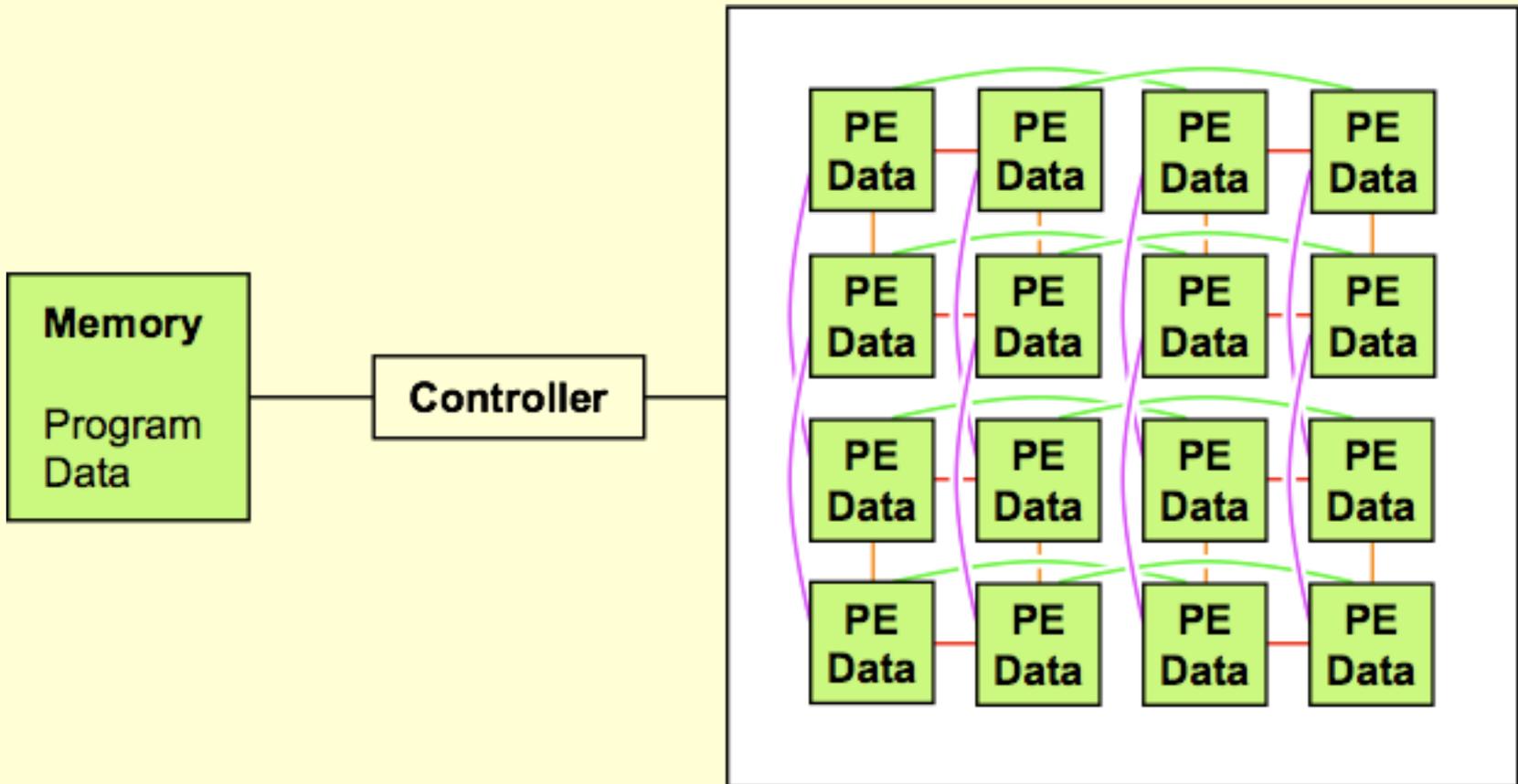
Communication: MasPar MP1

- Fast local X-net
- Slow global routing



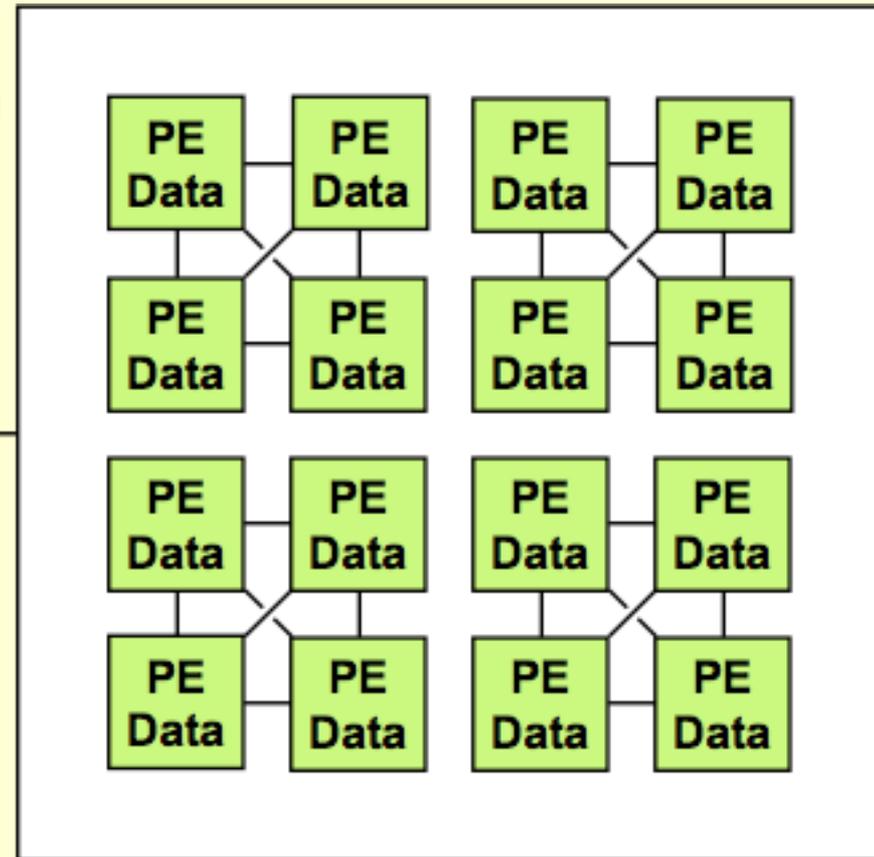
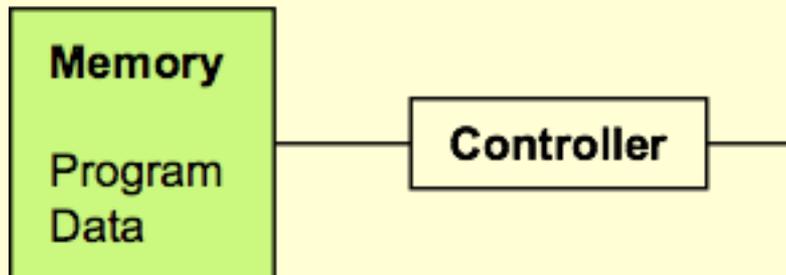
Communication: CM2

- Hypercube local routing
- Wormhole global routing



Communication: PixelFlow

- Dense connections within block
 - Single swizzle operation collects one word from each PE in block
 - Designed for antialiasing
 - NO inter-block connection
 - NO global routing



Data Parallel Languages

- SIMD programming
 - PE point of view
 - Data: shared or per-PE
 - What data is distributed?
 - What is shared over PE subset
 - What data is broadcast with instruction stream?
 - Data layout: shape $[256][256]d$;
 - Communication primitives
 - Higher-level operations
 - Scan/Prefix sum: $[i]r = \sum_{j \leq i} [j]d$
 - $1, 1, 2, 3, 4 \rightarrow 1, 1+1=2, 2+2=4, 4+3=7, 7+4=11$

Single Program Multiple Data

- Many problems do not map well to SIMD
 - Better utilization from MIMD or ILP
- Data parallel model \Rightarrow Single Program Multiple Data (SPMD) model
 - All processors execute identical program
 - Same program for SIMD, SISD or MIMD
 - Compiler handles mapping to architecture