5. Data Parallel Architectures

Basics

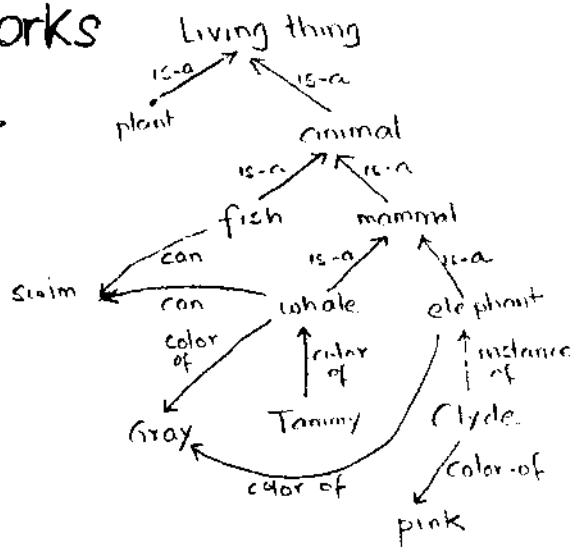
- Data parallelism vs. Control parallelism
  - Data parallelism: Concurrency arises from executing essentially the same code on large number of objects.
  - Control parallelism: Concurrency arises from executing different threads of control in parallel.
- Hypothesis: Applications that successfully use massively parallel machines will mostly exploit data parallelism.
- Data parallel model was originally linked with SIMD machines.
- Machines: Maspar MP-1, Thinking Machines CM-2 and CM-5.

The Connection Machine

- **Original motivation was semantic nets (Hillis, AI Memo 646, MIT, 1981)**

- **A SIMD machine with very simple processors to do marker-passing algs.**

semantic networks
one PE per node



All animals that can swim and are gray?

- **Was made by Thinking Machines Corp. in Boston**

- **Primary use is for data parallel computing (scientific, database, ... applns)**

CM-2 System Organization

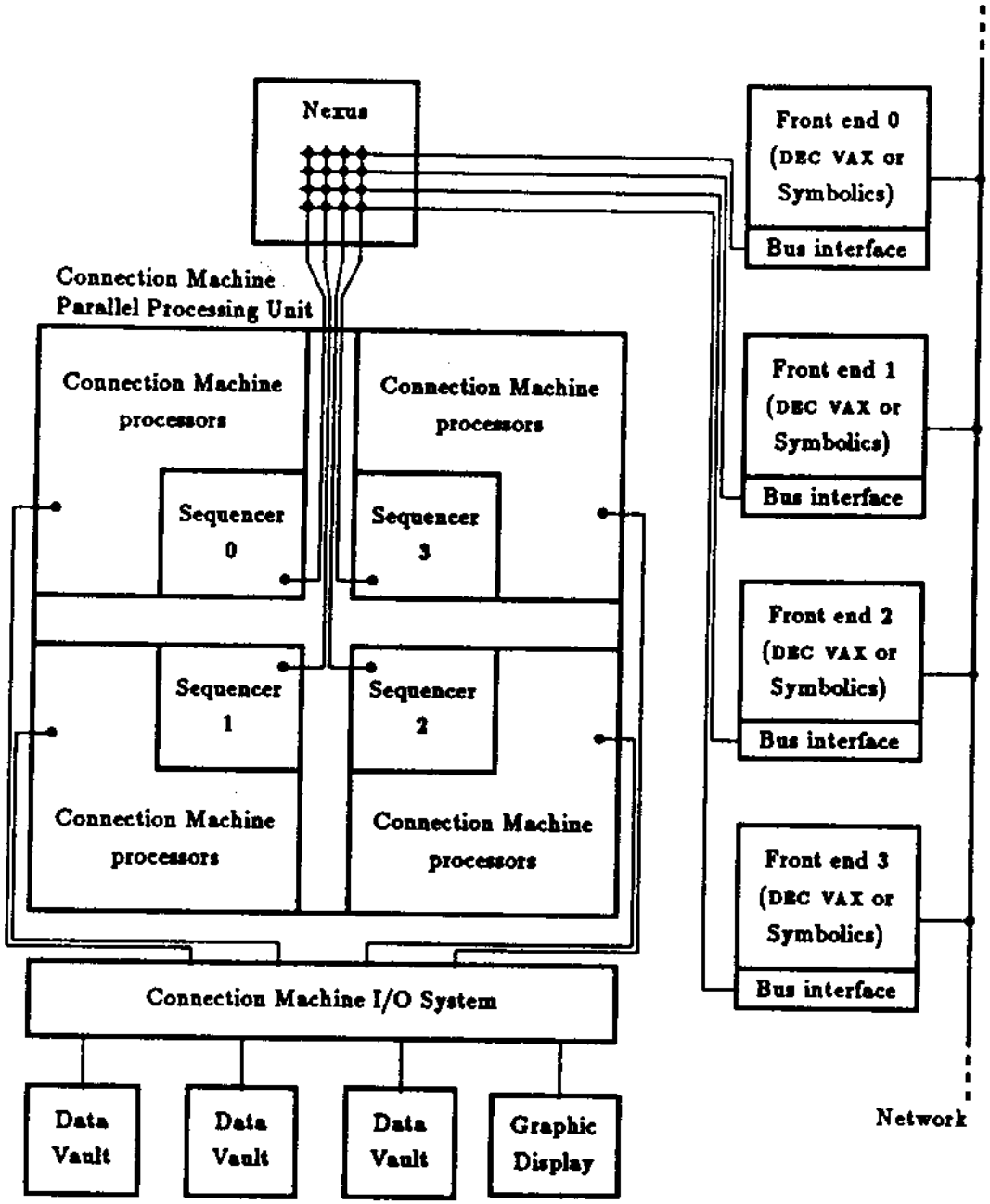
- Major components:
  - Parallel Processing Unit (PPU):
    - 16-64K bit-serial PEs, each with ~8Kbytes of memory; optionally 512-2K FPUs. (≈ 20μs for 32-bit add ⇒ 64k PE → 3000 MIPS)
    - hypercube interconnection network with support for combining
    - 1-4 sequencers; I/O controllers; ...
  - 1-4 Front-End Computers (FECs):
    - user application ran on FEC.
    - PPU was a peripheral device to which instructions were broadcast
  - Sophisticated I/O system:
    - for every 8K PEs, 256-bit wide I/O channel
    - data vault (39 disks: 32 data, ECC) for high-performance disk I/O
    - graphics support

Communication

- A general router: All PEs may concurrently send/receive information from another PE → processes send messages to each other

- NEWS: nearest-neighbor communication
  programmable: 256 x 256 torus, or 16-ary 4 cube...
  communication faster since instruction need not specify an address

→ underlying network is a 2-ary 12-cube on which both mech. are implemented



With Sequencers:

CM viewed as 4 independent slightly smaller CMs (does not affect pgms as they use virtual PEs)

The Paris Language

- Instruction Set Architecture of the PPU

- Provides 2 powerful abstractions:
  - a much richer instruction set (discussed later)
  - notion of virtual processors (VPs)
    - VPs are independent of # of PPs on machine
    - if VPs > PPs, then #-of-VPs/#-of-PPs VPs mapped on each PP
      - system transparently splits memory per PP; does routing; ...

VP given at init time

- Notion of current context in SIMD machines
  - a context-flag in each PE identifies those participating in computation
  - the current context is used to execute conditional statements

The Instruction Set

- **Most instructions are memory-to-memory 2/3 operand instructions**
  (CM-S-add3 : $value_1 + value_2 \rightarrow value_3$ all in mem)
- **Standard set of logical and arithmetic/logic scalar instructions**
  - **e.g., CM-s-add-constant-2, move**
  - **~ 26 microseconds for 32-bit add on 64K PEs**

- **Intra-processor vector instructions (vector within each PE)**
  - **e.g., f-vector-dot-product, ...**
    f-matrix-multiply
- **Inter-processor vector instructions**
  - **each PE has only one element of the vector**
  - **three kinds: global reductions; regular scans; segmented scans**

S-add signed
U-add unsigned
F-add float

Examples of Scans

- **global-s-add: reduction operator that returns the sum of all elements of the vector**

- **s-add-scan: A parallel-prefix operation that replaces each item with the reduction of all vector elements preceeding it (300us on 64K 32-bit ints)**

X:	3	5	2	3	1	4	2	7	3	2	1
scan-X:	3	8	10	13	14	18	20	27	30	32	33

- **segmented-s-add-scan: parallel-prefix done on segments of an array**

X:	3	5	2	3	1	4	2	7	3	2	1
segment	0	0	0	1	0	0	0	0	1	0	0
seg-scan-X:	3	8	10	3	4	8	10	17	3	5	6

Interprocessor Communication

- Communication within cartesian-grid (NEWS net) much much cheaper than global communication

- Global communication:
  - fetch/store: assume that only one person storing to any given destn.
  - get/send: multiple PEs may request from or send to a given destn
    - the network does combining in these cases
    - example: send-with-s-max (only max value stored at destn)

if all 64K PEs send a message, takes somewhere between 260-820 μs to complete

- Data transfer between front-end and PPU

Floating Point on CM-2

- **Floating point was added as an after thought, so somewhat clumsy**
- **Every set of 32 PEs has one pipelined floating point unit**
- **CM-2 in this case appears as a 512-2K PE machine**
- Each PE contains: bit-serial ALU + associated latches
  64 K bits of bit-addressable memory
  4 1-bit flag regs
  float pt HW
  router interface, I/O interface

Languages

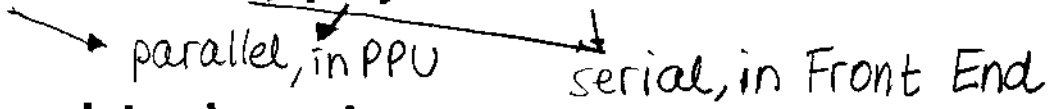
- Data parallel programs on CM-2 associate a PE with each data element and operate on data in parallel using a sequential stream of instructions broadcast to all PEs.

- Since a sequential stream of instructions is used
  - normal control constructs of conventional languages suffice
  - only few additional twists to the semantics needed

- Languages on CM-2: C*, Fortran, *Lisp, CM-Lisp, ...

Issues for Data Parallel Languages

- Establishing parallel data structures:
  - C*: domain vs. struct; poly vs. mono



- Linkages among data elements:
  - C*: using pointers or array subscripts

- Operations on parallel data structures:
  - operations on parallel data structures automatically in parallel on PPU
  - operations on serial data done sequentially on FEC
  - operations on mixed data turn into reductions or broadcasts

- Conditionals:
  - For if construct, both then-part and else-part are evaluated
  - For while construct, wait until all PEs exit the construct

```
domain employee {
    double salary;
    employee_type type;
    char *name;
    int knowledge;
};

domain part {
    int part_number;
    double price;
    vendor *supplier;
    char *description;
};

domain book {
    char *title, *ISBN;
    int content;
    employee *owner;
};

domain employee Fred;              /* Processor 0 */
domain employee Sally;             /* Processor 1 */
domain part grommet;               /* Processor 2 */
domain employee George;            /* Processor 3 */
domain part wing_nut;              /* Processor 4 */
domain book my_novel;              /* Processor 5 */
domain employee programmer[20];    /* Processors 6-25 */
```

```
mono int total;
poly int salary;
poly extern float coefficients[10];
mono int *poly x;   /* A poly pointer
                       to a mono integer. */
poly static struct foo x[20];
poly auto double all_the_day;
```

Maspar MP-1

- Later design than CM-2
  - more extensive use of VLSI (thus denser and lower cost)
    - upto 16K PEs; 4-bit PEs; 32 PEs on a chip
  - explicit x net and global net
    - x-net operations can be pipelined → Neighborhood mesh
    - global net is circuit switched multistage network → Crossbar
  - RISC philosophy
    - register-based instruction set (40 32-bit regs per PE)
    - VPs implemented in software by compiler
    - Floating point and integer performance better matched

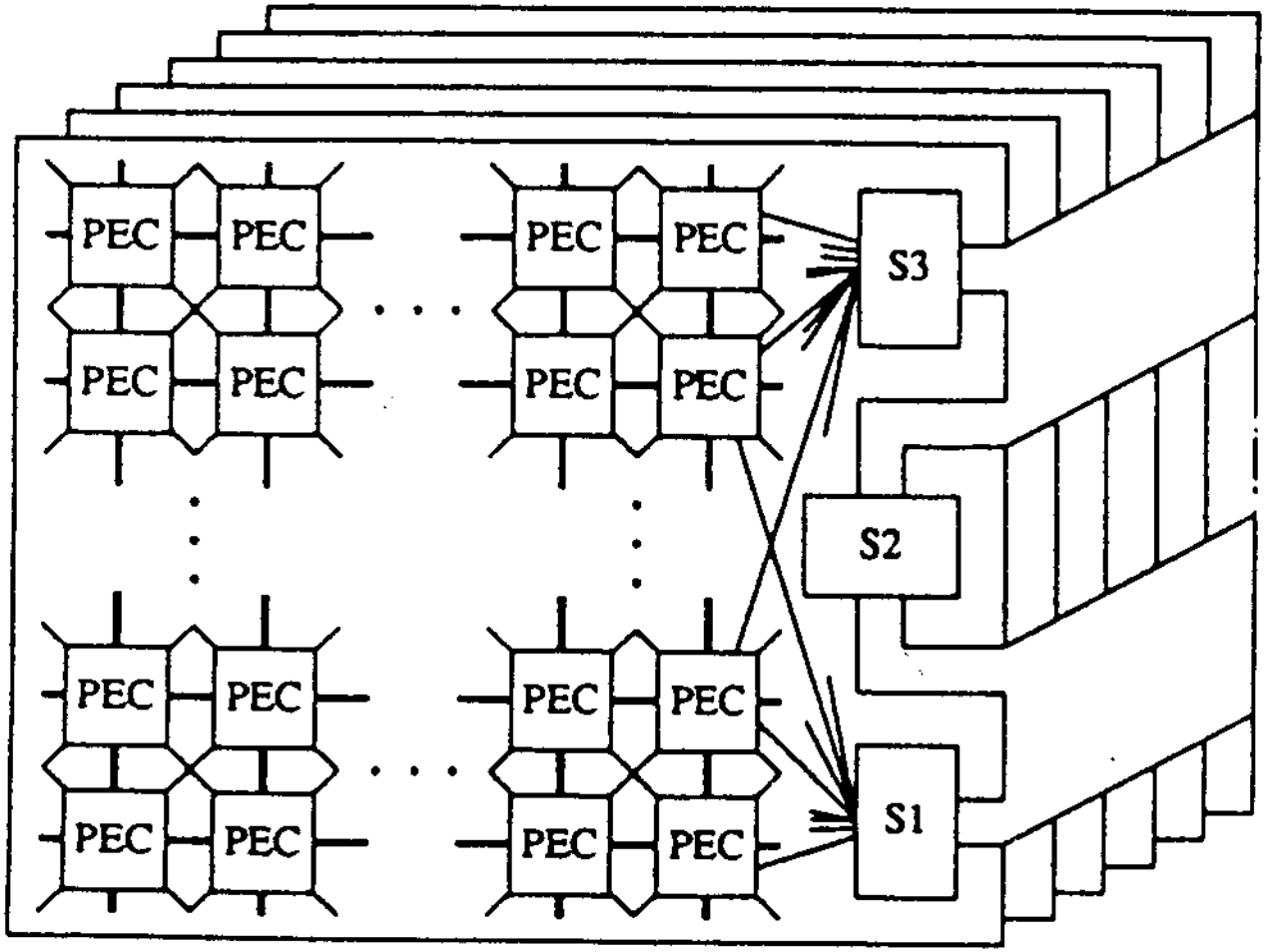


Figure 1. Array of PE Clusters

Weaknesses of CM-2 and MP-1

- If mostly used for floating pt. computations, why 1-bit or 4-bit PEs
- No good support for multiprogramming
- SIMD is not a very general-purpose model