Machine and Network Models

• Memoryless serial and parallel machines
  — logic & algebraic circuits (ring, field ops)

• Serial machines: RAM & TM
  — memory hierarchies

• Parallel machines with memory
  — fine- vs coarse-grained computers
  — PRAM – p RAMs with shared memory

Network and VLSI Models

Loosely coupled computers:

NAND gate:

Nanotechnology

— Nanowires & nanotubes for connections
— Switches at intersection of nanowires
— Access from micro level

h-hot addressing
Performance Metrics

- logical and algebraic circuits
  - circuit size and depth
- RAM and TM
  - time (no. steps) & space (no. locations)
- Parallel machines
  - time, no. processors, & space
- Memory hierarchies
  - I/O time vs primary storage space
- Parallel machines
  - time, no. processors, & space

Networked Computers

- Such computers are modeled by a graph. Vertices represent processors and edges denote connections between processors.
- Examples of important networks:
  - trees, linear and multidimensional arrays. Examples of the latter coming.

Flynn’s Taxonomy

- Michael Flynn classified computers as
  - SISD (single instruction, single data)
    Single thread of control accessing one datum on each time step
  - SIMD (single instruction, multiple data)
  - MISD (multiple instruction, single data)
  - MIMD (multiple instruction, mult data)
    Multiple threads of control accessing multiple data on each time step
- The data parallel model realizes the SIMD style of programming
  - instructions vary from operations on vectors to ops on bits but with one thread of control

Crude Bounds on Parallel Computing

- Amdahl’s Law If a fraction $f$ of a (legacy) program’s execution time on a serial machine is parallelizable, the speedup $S$ achievable by the program on a $p$-processor RAM satisfies the following inequality.

$$ S \leq \frac{1}{(1-f) + \frac{f}{p}} $$

Thus, if $f = 90\%$ (which is large), $S \leq 10$, which is small even if $p$ is infinite. Note that efficient parallel programs are generally very different from efficient serial ones.

- Brent’s Principle provides a technique to convert an inherently parallel problem to an efficient parallel algorithm with good speedup. (See book.)
Multidimensional Mesh Models

- Matrix-vector multiplication on a 1D systolic array. On each cycle $S_i$ is equal to the sum of $S_{i+1}$ and the product of $x_i$ with the vertical input.

- Systolic arrays – cells operate in synch, as shown above.

- 2D array processors have connections along NSEW axes. Toroidal connections possible.

- Higher dimensional meshes possible

- How can cells be numbered in an array?

![Matrix Multiplication on a 2D Array]

- As two values enter a cell they are multiplied and added to the current value which is 0 initially. Why does this algorithm correctly compute the matrix product?

- This algorithm multiplies two $n \times n$ matrices in $4n-2$ steps. Why? How many steps are needed to compute the last of the results and then deliver the results to the output?

Sorting on a 1D Array

- Bubble sort compares adjacent elements and "bubbles up" the largest elements.

- This can be implemented on a linear array whose elements are numbered $1, 2, \ldots, n$ when $n$ is even by alternating the following operations:
  - a) comparing and swapping, if necessary, $i$ and $i+1$ for $i$ odd
  - b) comparing and swapping, if necessary, $i$ and $i+1$ for $i$ even

- The textbook gives a proof that this procedure correctly sorts.

Data Storage in Nanoarrays

- Two methods of addressing arrays:
  - 1-hot: individual nanowires controllable
  - $h$-hot

- Goal: enter 1s in all 0s array efficiently

- Two storage methods:
  - write subarrays of 1s
  - write subarrays of 1s or 0s

- Results:
  - NP-hard to enter data in min # steps and to approximate minimum
  - Problems, such as $n \times n$ diagonal, need $n$ steps for 1st method, log $n$ for 2nd
  - Not known if problem is NP-hard with second storage method