Bitonic Sort Summary

- To create a bitonic sequence:
  - Sort the two halves of the sequence by any method, so that one half is ascending, the other descending.

- To sort a sequence known to be bitonic:
  - Compare-exchange elements $j$ and $j+n/2$, for $j = 0, 1, 2, ..., n/2-1$
  - It can be shown that the two halves are bitonic, and all of one half is $\leq$ all of the other.
  - Therefore, sort them as bitonic sequences.

Bitonic Sort Recursion

S = sorter
B = sorter for bitonic sequence

Exercise

- Comment on the pipelinability of bitonic sorting
- Analyze the time taken for bitonic sorting

Other facets of sorting networks

- The number of comparators for odd-even merging is $O(n \log^2 n)$, which is typical of the most accessible constructions.
- An upper bound of $O(n \log n)$ has been shown, but the constants are large (in the 1000's).

Systolic Arrays

- This is a form of **pipelining**, sometimes in more than one dimension.
- The term “systolic” was first used in this context by H.T. Kung, then at CMU; it refers to the “pumping” action of a heart.
- Machines have been constructed based on this principle, notable the iWARP, fabricated by Intel.

Systolic Matrix Multiplication

- Processors are arranged in a 2-D grid
- Each processor accumulates one element of the product
- The elements of the matrices to be multiplied are “pumped through” the array.
Systolic Matrix Multiplication
Illustrated with two 3x3 matrices

alignments in time
rows of a

Systolic Matrix Multiplication
Illustrated with two 3x3 matrices

alignments in time
b 0, 0
b 0, 1
b 0, 2
b 1, 0
b 1, 1
b 1, 2
b 2, 0
b 2, 1
b 2, 2

Systolic Matrix Multiplication
Illustrated with two 3x3 matrices

alignments in time

Systolic Matrix Multiplication
Illustrated with two 3x3 matrices

alignments in time

Systolic Matrix Multiplication
Illustrated with two 3x3 matrices

alignments in time

Systolic Matrix Multiplication
Illustrated with two 3x3 matrices

alignments in time
A Related Algorithm: Cannon’s Method

- Let's take another view of systolic multiplication: Consider the rows and columns of the matrices to be multiplied as strips that are slide past each other.
- The strips are staggered so that the correct elements are multiplied at each time step.
Fourth step
columns of b (inverted)
rows of a (reversed)

Fifth step
columns of b (inverted)
rows of a (reversed)

First step
columns of b (inverted)
rows of a (reversed)

Cannon’s Method
- Rather than have some processors idle,
  wrap the array rows and columns so that
  every processor is doing something on each
  step.
- In other words, rather than feeding in the
  elements, they are rotated around,
  starting in an initially staggered position as in
  the systolic model.
- We also change the order of products slightly,
  to make it correspond to more natural storage
  by rows and columns.

Cannon Variation
Note that the a diagonal is in the left column and the b diagonal is in the top row.

Application of Cannon's Technique
- Consider matrix multiplication of 2 n x n
  matrices on a distributed memory machine,
  on say, n² processing elements.
- An obvious way to compute is to think of the
  PE’s as a matrix, with each computing one
  element of the product.
- We would send each row of the matrix to n
  processors and each column to n.
- In effect, in the obvious way, each matrix
  is stored a total of n times.
Obvious Matrix Multiply

Columns of \( b \) distributed to each PE in column.

Rows of \( a \) distributed to each PE in row.

Row x Column on respective PEs.

Cannon’s Method

- Cannon’s method avoids storing each matrix \( n \) times, instead **cycling** (“piping”) the elements through the PE array.
- (It is sometimes called the “pipe-roll” method.)
- The problem is that this cycling is typically too fine-grain to be useful for element-by-element multiply.

Partitioned Multiplication

- Partitioned multiplication divides the matrices into **blocks**.
- It can be shown that multiplying the individual blocks as if elements of matrices themselves gives the matrix product.

Block Multiplication

- The blocks are aligned initially as the elements were in our description.
- At each step, entire blocks are transmitted down and to the left of neighboring PE’s.
- Memory space is conserved.

Cannon’s Method is Fine for Block Multiplication

- Analyze the running time for the block version of Cannon’s method for two \( n \times n \) matrices on \( p \) processors, using \( t_{\text{comp}} \) as the unit operation time and \( t_{\text{comm}} \) as the unit communication time and \( t_{\text{start}} \) as the per-message latency .
- Assume that any pair of processors can communicate in parallel.
- Each block is \((n/\sqrt{p}) \times (n/\sqrt{p})\).
Fox’s Algorithm

- Also for block matrix multiplication, it has a resemblance to Cannon’s algorithm.
- The difference is that on each cycle:
  - A row block is broadcast to every other processor in the row.
  - The column blocks are rolled cyclically.

```
Step 1
a00 * b00 a00 * b01 a00 * b02
a11 * b10 a11 * b11 a11 * b12
a22 * b20 a22 * b21 a22 * b22

Step 2
a01 * b10 a01 * b11 a01 * b12
a12 * b20 a12 * b21 a12 * b22
a20 * b00 a20 * b01 a20 * b02

Step 3
a02 * b20 a02 * b21 a02 * b22
a10 * b00 a10 * b01 a10 * b02
a21 * b10 a21 * b11 a21 * b12
```

Fox vs. Cannon

Synchronous Computations

PP Chapter 6

Barriers

- Mentioned earlier
- Synchronize all of a group of processes
- Used in both distributed and shared-memory
- Issue: Implementation & cost

Counter Method for Barriers

- One-phase version
  - Use for distributed-memory
  - Each processor sends a message to the others when barrier reached.
  - When each processor has received a message from all others, the processors pass the barrier
Counter Method for Barriers

- Two-phase version
  - Use for shared-memory
  - Each processor sends a message to the master process.
  - When the master has received a message from all others, it sends messages to each indicating they can pass the barrier.
  - Easily implemented with blocking receives, or semaphores (one per processor).

Tree Barrier

- Processors are organized as a tree, with each sending to its parent.
- Fan-in phase: When the root of the tree receives messages from both children, the barrier is complete.
- Fan-out phase: Messages are then sent down the tree in the reverse direction, and processes pass the barrier upon receipt.

Butterfly Barrier

- Essentially a fan-in tree for each processor, with some sharing toward the leaves.
- Advantage is that no separate fan-out phases required.

Butterfly Barrier

- These can be accomplished along with a barrier:
  - Reduce according to binary operator (esp. good for tree or butterfly barrier)
  - All-to-all broadcast

Barrier Bonuses

- To implement a barrier, it is only necessary to increment a count (shared memory) or send a couple of messages per process.
- These are communications with null content.
- By adding content to messages, barriers can have added utility.