MASC
The Multiple Associative Computing Model

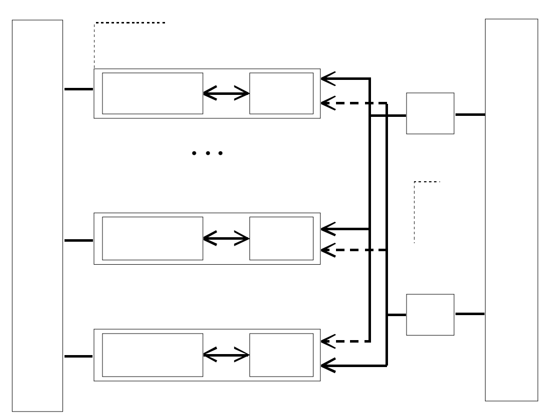
Johnnie Baker, Jerry Potter, Robert Walker
Kent State University
(http://www.mcs.kent.edu/~parallel/)

OVERVIEW

- Introduction
  - Motivation for the MASC model
  - The MASC and ASC Models
  - Languages Designed for the ASC Model
  - Some ASC Algorithms and Programs
- ASC and MASC Algorithm Examples
  - ASC version of Prim's MST Algorithm
  - ASC version of QUICKHULL
  - MASC version of QUICKHULL.
- Simulations involving MASC (Overview)
  - Background History and Basics
  - Overview of PRAM Simulations
  - Overview of Enhanced Mesh Simulations
  - General Conclusions

Motivation For MASC Model

- The STARAN Computer (Goodyear Aerospace, early 1970's) provided an architectural model for associative computing.
- MASC provides a 'definition' for associative computing.
- Associative computing extends the data parallel paradigm to a complete computational model.
- Provides a platform for developing and comparing associative, MSIMD (Multiple SIMD) type programs.
- MASC is studied locally as a computational model (Baker), programming model (Potter), and architectural model (Baker, Potter, & Walker).
- Provides a practical model that supports massive parallelism.
- Model can also support intermediate parallel applications (e.g., multimedia computation, interactive graphics) using on-chip technology.
- Model addresses fact that most parallel applications are data parallel in nature, but contain several regions where significant branching occurs.
  - Normally, at most eight active sub-branches.
- Provides a hybrid data-parallel, control-parallel model that can be compared to other parallel models.



- Basic Components
  - An array of cells, each consisting of a PE and its local memory
  - An interconnection network between the cells
  - One or more instruction streams (ISs)
  - An IS communications network
- MASC is a MSIMD model that supports
  - both data and control parallelism
  - associative programming.
- MASC(n, j) is a MASC model with n PEs and j ISs

Basic Properties of MASC

- Instruction Streams or ISs
  - A processor with a bus to each cell
  - Each IS has a copy of the program and can broadcast instructions to cells in unit time
  - NOTE: MASC(n,1) is called ASC
- Cell Properties
  - Each cell consists of a PE and its local memory
  - All cells listen to only one IS
  - Cells can switch ISs in unit time, based on a data test.
  - A cell can be active, inactive, or idle
    - *Inactive* cells listen but do not execute IS commands
    - *Idle* cells contain no useful data and are available for reassignment
- Responder Processing
  - An IS can detect if a data test is satisfied by any of its cells (each called a **responder**) in constant time
  - An IS can select (or **pick one**) arbitrary responder in constant time.
  - Justified by implementations using a resolver

- Constant Time Global Operations (across PEs with a common IS)
  - Logical OR and AND of binary values
  - Maximum and minimum of numbers
  - Associative searches (see next slide)
- Communications
  - There are three real or virtual networks
    - PE communications network
    - IS broadcast/reduction network
    - IS communications network
  - Communications can be supported by various techniques
    - actual networks such as 2D mesh
    - bus networks
    - shared memory
- Control Features
  - PEs, ISs, and Networks operate synchronously, using the same clock
  - Control Parallelism used to coordinate the multiple ISs.

Reference: An Associative Computing Paradigm, IEEE Computer, Nov. 1994, Potter, Baker, et al., pg 19-26. (Note: MASC is called ASC in this article.)

The Associative Search

IS

	Make	Color	Year	Model	Price	On lot		Busy-idle
PE1	Dodge	red	1994			1		1
PE2						0		0
PE3	Ford	blue	1996			1		1
PE4	Ford	white	1998			0		1
PE5						0		0
PE6						0		0
PE7	Subaru	red	1997			1		1

Characteristics of Associative Programming

- Consistent use of data parallel programming
- Consistent use of global associative searching & responder processing
- Regular use of the constant time global reduction operations: **AND, OR, MAX, MIN**
- Data movement using IS bus broadcasts and IS **fork** and **join** operations to minimize the use of the PE network.
- Tabular representation of data
- Use of searching instead of sorting
- Use of searching instead of pointers
- Use of searching instead of ordering provided by linked lists, stacks, queues
- Promotes an intuitive type of programming that promotes high productivity
- Uses structure codes (i.e., numeric representation) to represent data structures such as trees, graphs, embedded lists, and matrices.
  - See Nov. 1994 IEEE Computer article.
  - Also, see Associative Computing by Potter

Languages Designed for MASC

- ASC was designed by Jerry Potter for MASC(n,1)
  - Based on C and Pascal
  - Initially designed as a parallel language.
  - Avoids compromises required to extend an existing sequential language
    - E.g., avoids unneeded sequential constructs such as pointers
  - Implemented on several SIMD computers
    - Goodyear Aerospace's STARAN
    - Goodyear/Loral's ASPRO
    - Thinking Machine's CM-2
    - WaveTracer
- ACE is a higher level language that uses natural language syntax; e.g., plurals, pronouns.
- Anglish is an ACE variant that uses an English-like grammar.
- An OOPs version of ASC for MASC(n,k) is planned (by Potter and his students)
- **Language Refs:** www.mcs.kent.edu/~potter/ and Jerry Potter, Associative Computing - A Programming Paradigm for Massively Parallel Computers, Plenum Publishing Company, 1992

Algorithms and Programs Implemented in ASC

- **A wide range of algorithms implemented in ASC <u>without use of PE network</u>**
  - Graph Algorithms
    - minimal spanning tree
    - shortest path
    - connected components
  - Computational Geometry Algorithms
    - convex hull algorithms (Jarvis March, Quickhull, Graham Scan, etc)
    - Dynamic hull algorithms
  - String Matching Algorithms
    - all exact substring matches
    - all exact matches with "don't care" (i.e., wild card) characters.
  - Algorithms for NP-complete problems
    - traveling salesperson
    - 2-D knapsack.
  - Data Base Management Software
    - associative data base
    - relational data base

(Cont) ASC Algorithms and Programs

- A Two Pass Compiler for ASC
  - first pass
  - optimization phase
- Two Rule-Based Inference Engines
  - OPS-5 interpreter
  - PPL (Parallel Production Language interpreter)
- A Context Sensitive Language Interpreter
  - (OPS-5 variables force context sensitivity)
- An associative PROLOG interpreter

- **Numerous Programs in ASC <u>using a PE network</u>**
  - 2-D Knapsack Algorithm using a 1-D mesh
  - Image Processing algorithms using 1-D mesh
  - FFT using Flip Network
  - Matrix Multiplication using 1-D mesh
  - An Air Traffic Control Program using Flip Network
    - Demonstrated using live data at Knoxville in mid 70's.

Preliminaries for MST Algorithm

- Next, a "data structure" level presentation of Prim's algorithm for the MST is given.
- The data structure used is illustrated in the example in Figure 6 on slide 15.
  - Figure 6 is from the basic paper in Nov. 1994 IEEE Computer (see slide 6).
- There are two types of variables for the ASC model, namely
  - the parallel variables (i.e., ones for the PEs)
  - the scalar variables (ie., the ones for the control unit).
  - Scalar variables are essentially global variables.
    - Can replace each with a parallel variable.
- In order to distinguish between them, the parallel variables names end with a "$" symbol.
- Each step in this algorithm is constant.
- One MST edge is selected during each pass through the loop in this algorithm.
- Since a spanning tree has n-1 edges, the running time of this algorithm is O(n).
- Since the sequential running time of the Prim MST algorithm is $O(n^2)$ and this time is optimal, this parallel implementation is cost-optimal.

Algorithm: ASC-MSP-PRIM(root)

- Initially assign any node to *root*.
- All processors set
  - *candidate$* to "waiting"
  - *current-best$* to ∞
  - the *candidate* field for the *root* node to "no"
- All processors whose distance *d* from their node to *root* node is finite do
  - Set their *candidate$* field to "yes
  - Set their *parent$* field to *root*.
  - Set *current_best$ = d*.
- While the *candidate* field of some processor is "yes",
  - Restrict the active processors to those responding and (for these processors) do
    - Compute the minimum value x of *current_best$*.
    - Restrict the active processors to those with *current_best$ = x* and do
      - pick an active processor, say one with node *y*.
        - » Set the *candidate$* value of this processor to "no"
      - Set the scalar variable *next-node* to *y*.

      - If the value *z* in the *next_node* field of a processor is less than *current_best$,* then
        - » Set *current_best$* to *z*.
      - Set *parent$* to *next_node*
- For all processors, if *candidate$* is "waiting" and the distance of its node from *next_node* is finite, then
  - Set *candidate$* to "yes"
  - Set *parent$* to *next-node*
  - Set *current_best$* to the distance of its node from *next_node.*

COMMENTS:

- Figure 6 on the next slide shows the data structure used in the preceding ASC algorithm for MST
- Next slide is from the Nov 1994 IEEE Computer paper referenced earlier.
  - This slide also gives a compact, data-structures level pseudo-code description for this algorithm
    - Pseudo-code illustrates Potter's use of pronouns (e.g., *them*)
    - The mindex function returns the index of a processor holding the minimal value.
  - This MST pseudo-code is much simpler than data-structure level sequential MST pseudo-codes (e.g., Sara Baase's algorithm textbook).

Slides from Maher's Work Go Here

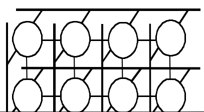
- First slide of Figure 6 in the IEEE Computer article on associative minimal spanning tree goes here. (Don't number this slide, as it would be slide 15.
- Next use slides 15 - 23 from my general presentations (prepared by Maher) called "An Associative Model of Computation". It is in latex and in directory ~jbaker/slides/matwah in UNIX directory.
- I am adding blank slides 16-23 to keep numbering correct.
- Work starting with slide 24 on simulations between enhanced meshes and MASC in dissertation work of Mingxian Jin.

Previous MASC Simulation

- MASC Simulation of PRAM
  - MASC(n,j) can simulate priority CRCW PRAM(n,m) in O(min{n/j, m/j}) with high probability.
  - MASC(n,1) [or ASC] can simulate priority CRCW with a constant number of global memory locations in constant time
    - This result is stronger than it first appears
    - Some CRCW algorithms only require a constant nr of global memory locations
  - A reverse simulation of MASC by Combining CRCW PRAM result will be in the dissertation of Mingxian Jin
- Self-simulation of MASC
  - Provides an efficient algorithm for MASC to efficiently simulate a larger MASC - with more PEs and/or ISs.
  - Establishes that MASC is highly scalable
  - MASC(n,j) can simulate MASC(N,J) in O(N/n + J) extra time and O(N/n + J) extra memory.

[redacted]nhanced Mesh, MMB

- [redacted]nes are basic mesh models
[redacted]n fixed or reconfigurable buses
- [redacted]e PE on a bus can broadcast to
[redacted]PEs during one step.
- [redacted]ed bus example:
- [redacted]multiple broadcasting (MMB)
- [redacted]D mesh
- [redacted]lumn bus enhancements
- [redacted]can occur along only row or column
[redacted]ot both) in one step



N

[redacted]

Simulation of MMB with MASC

- Since both models have identical 2D meshes, these do not need to be simulated
- Since the power of PEs in respective models are identical, their local computations are not simulated
- To simulate a MMB row broadcast on the MASC,
  - All PEs switch to their assigned row IS
  - The IS for each row checks to see if there is a PE that wishes to broadcast
  - If true, the IS broadcasts this value to all of its PEs (i.e., the ones on its assigned row).
- Simulation of a MMB column broadcast is similar
- The running time is O(1)
- There are examples that show the MASC model is **strictly more powerful** than the MMB model

Theorem 1.

- MASC(n, j) with a 2-D mesh is strictly more powerful than a $\sqrt{n}\times\sqrt{n}$ MMB for $j = \Omega(\sqrt{n})$.

- An algorithm for a $\sqrt{n}\times\sqrt{n}$ MMB can be executed on MASC(n, j) with $j=\Omega(\sqrt{n})$ and a 2-D mesh with a running time at least fast as the MMB time.

Simulation of MASC by MMB

- PE(1,1) stores a copy of the program and simulates the $\sqrt{n}$ ISs sequentially.
- Each instruction stream command or datum is first sent by P(1,1) to the PEs in the first column.
- Next, the PEs in the first column broadcast this command or datum along the rows to all PEs.
- Each MMB processor uses two registers, *channel* and *status*, to decide whether or not to execute the current instruction.
  - *channel* records which IS the processor is assigned to
  - *status* records whether PE is active, inactive, etc
- The simulation of $\sqrt{n}$ simultaneous broadcasts of ISs takes O($\sqrt{n}$) time.
- A local computation, memory access, or a data movement along local links are identical in the two models and require O(1) time.
- The execution of a global reduction operator **OR**, **AND**, **MAX**, **MIN** takes O($n^{1/6}$) using an optimal MMB algorithm.
- Since the global reduction operators may be computed for O($\sqrt{n}$) ISs, an upper bound is O($\sqrt{n}\times n^{1/6}$) or O($n^{2/3}$).

Theorem 3.

- MASC(n, $\sqrt{n}$) with a 2-D mesh can be simulated by a $\sqrt{n}\times\sqrt{n}$ MMB in O($n^{2/3}$) time with O($\sqrt{n}$) extra memory