An Object Oriented Framework for an Associative Model of Parallel Computation

M. Scherger, J. Potter, and J. Baker
{mscherge, potter, jbaker}@cs.kent.edu
Department of Mathematics and Computer Science
Kent State University
Kent, Ohio 44242

Abstract

An object oriented description and framework of the Multiple ASsociative Computing (MASC) model of parallel computation is presented. This description identifies MASC objects and specifies various object and inter-object relationships, dependencies, and behaviors. This was achieved by describing various views of the MASC model by using many of the UML structural and behavioral diagrams. This object oriented framework has been highly useful in designing an implementation of a runtime environment for the MASC model. Also the object oriented framework has been highly effective for further parallel modeling techniques, comparisons to other parallel models, MASC parallel system software research, and MASC algorithm development.

Keywords: parallel models, object oriented, parallel architectures.

1 Introduction

Abstract models of parallel computation are important vehicles for the design and development of parallel architectures, algorithms, and programming languages [4]. As research continues in the development of these abstract models of parallel computation, so do advances in the implementation a model using both software and hardware. To assist in the design and development of models of parallel computation, often a more detailed model is needed in describe the static and dynamic components of the model.

This paper presents an object oriented description and framework for an associative model of parallel computation. This object oriented framework is useful for the development of parallel system software, such as compilers, interpreters, and runtime environments. Also the object oriented framework is useful in determining predictability parameters reflected in the model.

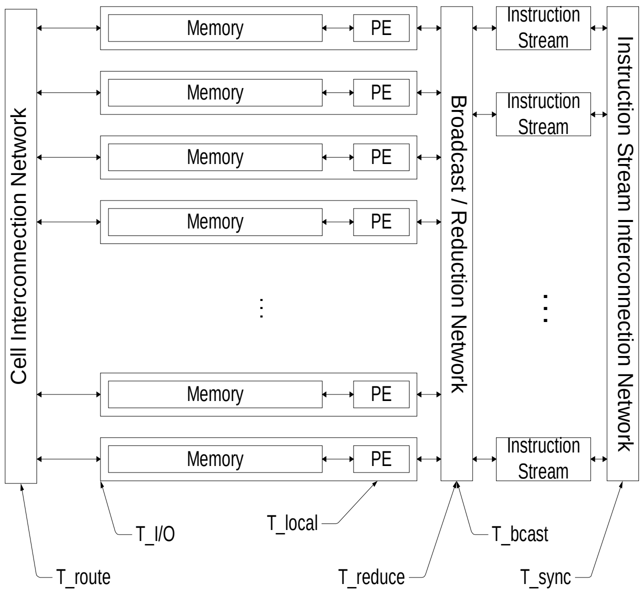
The MASC (for Multiple ASsociative Computing) model for parallel computation supports a generalized version of an associative style of computing that has been in use since the introduction of the associative SIMD computers (STARAN, MPP) in the early 1970s and 1980s. The MASC model includes the well-known data parallel-programming paradigm and extends this paradigm to a complete computational model. The associative features of the model allow data in the local memories of the processors (PE's) to be located by content rather than by address. A complete description of MASC (a renaming of the original ASC model to emphasize multiple instruction streams) can be found in [6] and [7].

The original description in [7], however, provides a conceptual view of the principal components and basic component interactions of the MASC model. This conceptual description of MASC is primarily used to introduce the model and not for research and development. By creating an object framework to describe MASC transforms this conceptual description into one that is not only object oriented, but provides several views for further MASC research in algorithm development and predictability, system software design and implementation, and hardware simulation and modeling. This object-oriented description of MASC details the principal objects and inter-object behaviors by using class inheritance, structural, and behavior diagrams. This common set of object and behaviors in the model allow MASC application developers, MASC system software designers and MASC hardware designers to establish a common reference point for interfacing system software to hardware components. Also the MASC object oriented framework provides a reference for hardware and algorithmic predictability analysis.

The remainder of this paper will first give a conceptual description of the MASC model of parallel computation and also present the MASC predictability parameters used in algorithm predictability analysis. Next, an overview and research motivation of the MASC object model is discussed. The basic objects of the MASC model are presented and organized into classes for basic structural modeling. Once the structural elements are defined, the object interactions are presented for a discussion of model predictability.

2 The MASC Model

The following is a conceptual description of the MASC model of parallel computation. As shown in figure 1, the MASC model consists of an array of processor-memory pairs called cells and an array of instruction streams.



Cell Interconnection Network
Memory
PE
Broadcast / Reduction Network
Instruction Stream
Instruction Stream Interconnection Network
T_I/O
T_local
T_bcast
T_route
T_reduce
T_sync

Figure 1: Conceptual view of the MASC model of parallel computation.

A MASC machine with *n* cells and *j* instruction streams is denoted as *MASC(n, j)*. It is expected that the number of instruction stream processors be much less than the number of cells. The model also includes three virtual networks:

1. A cell network used for cell-to-cell communication. This network is used for the parallel movement of data between cells. This network could be a linear array, mesh, hypercube, or a dynamic interconnection network.
2. A broadcast/reduction network used for communication between an instruction stream and a set of cells. This network is also capable of performing common reduction operations.
3. An instruction stream network used for inter-instruction stream communication.

Cells can receive their next set of instructions to execute from the instruction stream broadcast network. Cells can send and receive messages to each other using some communication pattern via the cell network. Each instruction stream processor is also connected to two interconnection networks. An instruction stream processor broadcasts instructions to the cells using the instruction stream broadcast network. The instruction streams also may need to communicate and may do so using the instruction stream network. Any of these networks may be virtual and be simulated by whatever network is present.

MASC provides one or more instruction streams. Each is assigned to a unique dynamic partition of cells. This allows a task that is being executed in a data parallel fashion to be partitioned into two or more tasks using of control parallelism. The multiple IS's supported by the MASC model allows for greater efficiency, flexibility, and reconfigurability than is possible with only one instruction stream. While SIMD architectures can execute data parallel programs very efficiently and normally can obtain near linear speedup, data parallel programs in many applications are not completely data parallel and contain several non-trivial regions where significant branching occurs.

In these regions, only a subset of traditional SIMD processors can be active at the same time. With MASC, control parallelism can be used to execute these different branches simultaneously. Other MASC properties include:

- The cells of the MASC model consist of a processing element (PE) and local memory. The accumulated memory of the MASC model consists of an array of cells. There is no shared memory between cells.
- Each instruction stream is a processor with a bus or broadcast/reduction network to all cells. Each cell listens to only one instruction stream and initially, all cells listen to the same instruction stream. The cells can switch to another instruction stream in response to commands from the current instruction stream.
- An active cell executes the commands it receives from its instruction stream, while an inactive cell

listens to but does not execute the command from its instruction stream. Each instruction stream has the ability to unconditionally activate all cells listening to it.

- Cells without further work are called idle cells and are assigned to a specified instruction stream, which among other tasks manages the idle cells.

- The average time for a cell to send a message through the cell network to another cell is characterized by the parameter t_{route}. Each cell also can read or write a word to an I/O channel. The maximum time for a cell to execute a command is given by the parameter t_{local}. The time to perform a broadcast of either data or instructions is given by the predictability parameter t_{bcast}. The time to perform a reduction operation is given by the predictability parameter t_{reduce}. The time for a cell to perform this I/O transfer is characterized by the parameter $t_{i/o}$. The time to perform instruction stream synchronization is characterized by the parameter t_{synch}.

- An instruction stream can instruct its active cells to perform an associative search in time $t_{bcast} + t_{local} + t_{reduce}$. Successful cells are called *responders*, while unsuccessful cells are called *non-responders*.

- The instruction stream can activate either the set of responders or the set of non-responders. It can also restore the previous set of active cells in $t_{bcast} + t_{local}$ time.

- Each instruction stream has the ability to select an arbitrary responder from the set of active cells in $t_{bcast} + t_{local}$ time.

- An active instruction stream can compute the *OR, AND, GLB,* or *LUB* of a set of values in all active cells in t_{reduce} time [3].

- An idle cell can be dynamically allocated to an instruction stream in $t_{synch} + t_{bcast}$ time.

3 MASC Object Model

This section will present an overview and research motivations for the MASC Object Model (MASCOM). The MASC Object Model is a set of parallel associative model component descriptions (classes) and object behaviors (messages). MASCOM was designed to be a reference of the abstract model for continuing MASC design, development, and implementation research. A MASCOM design goal is that the framework is to be portable when implemented across different classes of parallel computing architectures. Another design goal is that the framework should reflect the costs of implementing (overhead) MASC in a particular runtime environment for algorithm and performance predictability.

For an implementation of hardware and system software components of an abstract model of parallel computation progresses, hardware designers and software developers can now have discussions using a common set of classes and objects. Consider the model diagram shown in figure 2. When discussing a “cell” in the MASC model, a hardware designer may view a cell as a FPGA or ASIC component containing an ALU and a small memory. A system software engineer may discuss the interactions of a “cell” in terms of dynamic activation, or controlling which instruction stream to listen. The end result is that different types of developers now have a common reference point to perform research and development while maintaining a common dialog with researchers in other areas.

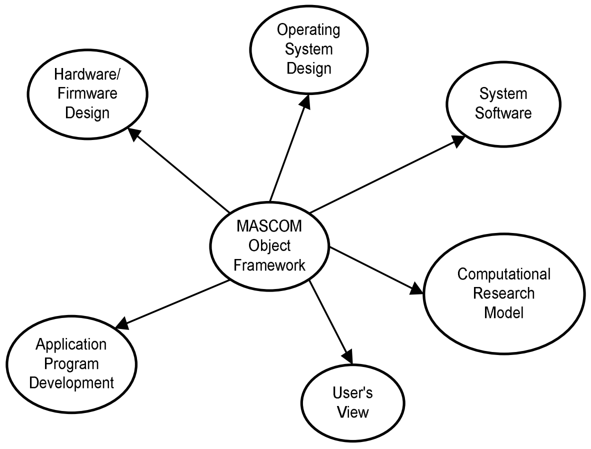

Operating System Design
Hardware/ Firmware Design
System Software
MASCOM Object Framework
Computational Research Model
Application Program Development
User's View

Figure 2: MASCOM framework viewpoints.

4 MASCOM Class Structure

The structural classes of MASCOM provide the model with a foundation of classes, objects, aggregations, and inheritance. The fundamental base classes of the MASCOM model are shown in figure 3.

At the abstract parallel model level, parallel data is stored in *fields*. Beginning with the memory in a cell, the basic class for storage element is the field class. The field class is an abstract base class in which other concrete field types are derived (integer fields, string fields, real fields, Boolean fields, etc).

To manage the fields, the field manager class maintains the collection of fields. The purpose of this collection is to provide the memory addressing capability for the cell and instruction stream classes.

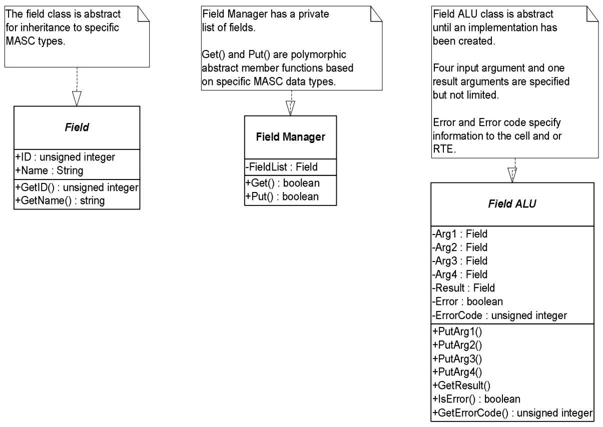

The field class is abstract for inheritance to specific MASC types.
Field
+ID : unsigned integer
+Name : String
+GetID() : unsigned integer
+GetName() : string
Field Manager has a private list of fields.
Get() and Put() are polymorphic abstract member functions based on specific MASC data types.
Field Manager
-FieldList : Field
+Get() : boolean
+Put() : boolean
Field ALU class is abstract until an implementation has been created.
Four input argument and one result arguments are specified but not limited.
Error and Error code specify information to the cell and or RTE.
Field ALU
-Arg1 : Field
-Arg2 : Field
-Arg3 : Field
-Arg4 : Field
-Result : Field
-Error : boolean
-ErrorCode : unsigned integer
+PutArg1()
+PutArg2()
+PutArg3()
+PutArg4()
+GetResult()
+IsError() : boolean
+GetErrorCode() : unsigned integer

Figure 3: Fundamental base classes of MASCOM.

The field manager shown in figure 4 is identified with a cell; however, the functionality of the Field Manager could be associated with an instruction stream if the cell does not have any memory addressing capabilities.

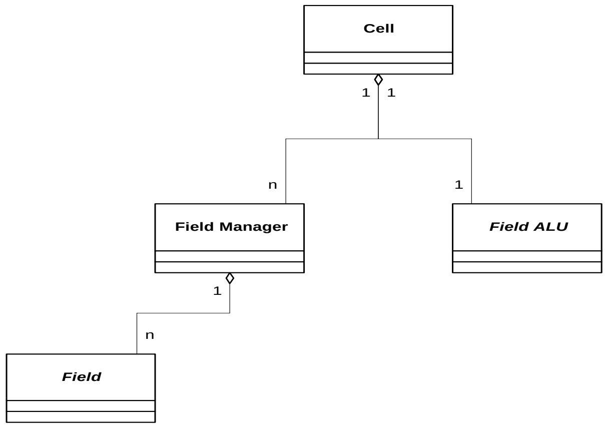


Figure 4: MASCOM cell aggregation diagram.

Each cell also has a *field ALU class* capable of performing basic arithmetic and logical operations on fields. The functionality of the field ALU class is not specified to allow for different types of processing elements to be "plugged-in" the MASC model. The cardinality of to cell to a Field ALU is 1:1. The field ALU can be implemented as a singleton pattern to reflect that when a parallel model of computation (or implementation of a model) is supporting virtual parallelism, there are fewer physical processing elements than data to be processed. As illustrated in figure 5, allowing a virtual cell manager class to maintain a collection of field managers supports virtual parallelism.

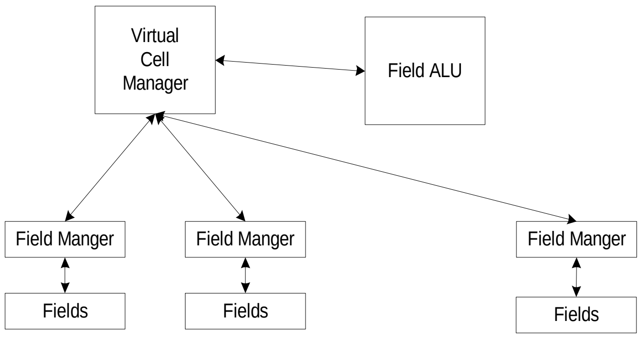


Figure 5: MASCOM Virtual cell organization.

A MASCOM *instruction stream class* has the same basic components and functionality of a MASC cell; i.e. it has the capability of performing computations on local (scalar) fields and communicates with other instruction streams. Since instruction streams must also be able to broadcast instructions to its partition of cells, and perform reductions from a partition of cells. Therefore, it is natural for the properties and functionalities of the instruction stream class to be derived from a cell class. Since the properties of an instruction stream can be derived from a cell, it is natural for an instruction stream to inherit the properties and functionality of a cell and this is illustrated in figure 6.

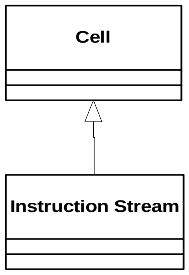


Figure 6: MASCOM Instruction stream / cell inheritance.

An *interconnection network class* is a class used for communication between cells and/or instruction streams. This is an abstract base class to allow for different interconnection network class implementations to be plug-in compatible with the existing MASC architecture. Thus, it will be possible to design and specify different types of networks used in MASC for different deployments of the model. For example, the cell network could be implemented using a grid mesh network, the broadcast reduction network could be implemented using a bus based network, while the instruction stream network could be implemented as a type of intelligent shared memory with basic network functionality.

Now that the basic structural components are defined, the MASCOM aggregation diagram of MASC is illustrated in figure 7.

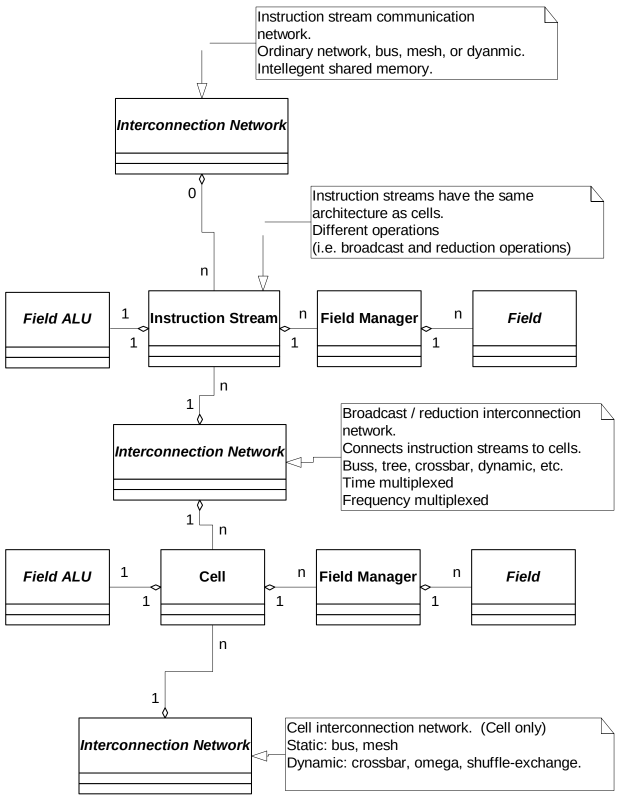

Instruction stream communication network.
Ordinary network, bus, mesh, or dyanmic.
Intellegent shared memory.
Interconnection Network
0
Instruction streams have the same architecture as cells.
Different operations
(i.e. broadcast and reduction operations)
n
Field ALU
1
1
Instruction Stream
n
1
Field Manager
n
1
Field
n
1
Interconnection Network
Broadcast / reduction interconnection network.
Connects instruction streams to cells.
Buss, tree, crossbar, dynamic, etc.
Time multiplexed
Frequency multiplexed
1
n
Field ALU
1
1
Cell
n
1
Field Manager
n
1
Field
n
1
Interconnection Network
Cell interconnection network. (Cell only)
Static: bus, mesh
Dynamic: crossbar, omega, shuffle-exchange.

Figure 7: MASCOM class structure and aggregation diagram.

The top Interconnection Network class is used for an object to allow for Instruction Stream communication and synchronization. The aggregation of an Instruction Stream is identical to that of a Cell, each allowing for Fields, Field Managers, and Field ALU classes. The middle Interconnection Network object is used for instruction stream broadcast and cell reductions. Finally the lower Interconnection Network class is used for inter-cell communication.

5 MASCOM Object Interactions and Predictability

The MASCOM object interactions and behavioral diagrams define the predictability in the MASC model [5]. The MASC predictability parameters can be illustrated using sequence diagrams; four are presented in this paper. The T_{route} is a measurement of cell-to-cell communication. Since the communication network and protocol are not specified however, the sequence diagram in figure 8 illustrates that the time T_{route} is bounded by the completion of all sends from one cell to another cell (sends to multiple cells are acknowledged by the * symbol).

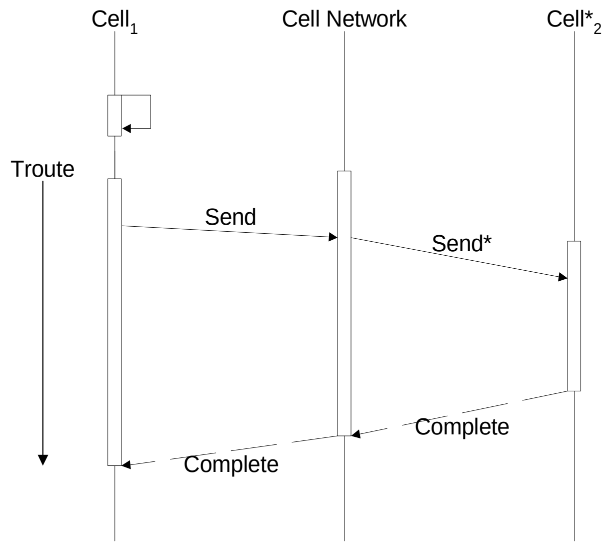


Figure 8: MASCOM predictability parameter T_{route} sequence diagram.

The next predictability parameter in the MASC model is T_{bcast}. This parameter measures the maximum time for performing a broadcast of data or instructions from an instruction stream to a cell or partition of the set of cells. Again, figure 9 illustrates that the time T_{bcast} is bounded by the completion of the broadcast to the interconnection object and the completion of all sends from the network to a partition of cells.

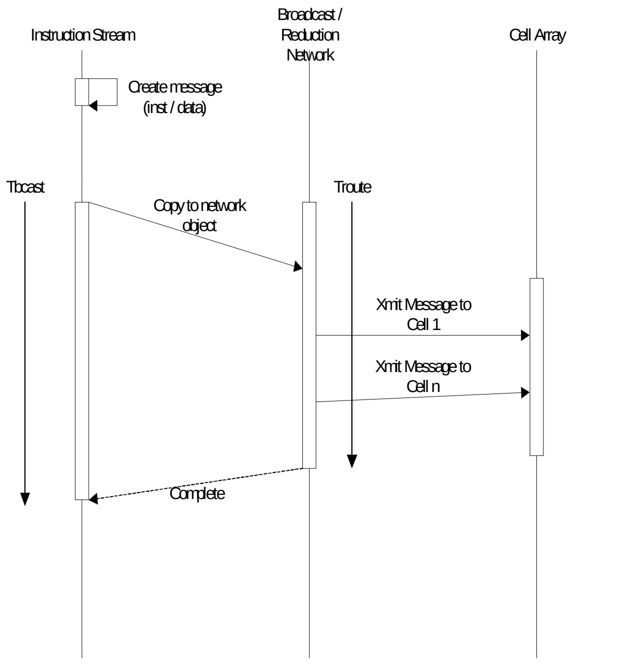


Figure 9: MASCOM predictability parameter T_{bcast} sequence diagram.

Another sequence diagram is for the MASC predictability parameter T_{local} that measures the maximum time a local cell operation requires. This is illustrated in figure 10. The time T_{local} is bounded by the time from when an instruction stream broadcasts the instruction to the cell to the time the instruction sequence is complete. Since not all instructions are arithmetic, the ALU request and complete sequence is optional. This sequence diagram also considers that an instruction stream may broadcast a series of instructions to a cell to execute instead of broadcasting each instruction individually.

A feature of parallel associative computing is to perform an associative search and process all the responding cells. This is a typical operation performed often using a basic parallel selection programming structure along with program iteration [10]. The basic associative search-process-retrieve cycle is illustrated in figure 11. Note that "Cell*" is used to illustrate that all cells of an instruction stream are used. The instruction stream would first instruct all cells to perform a parallel search by broadcasting the datum and fields to search. Once complete, the first responder cell is identified, selected, and processed. This is repeated iteratively for the remaining responding cells.

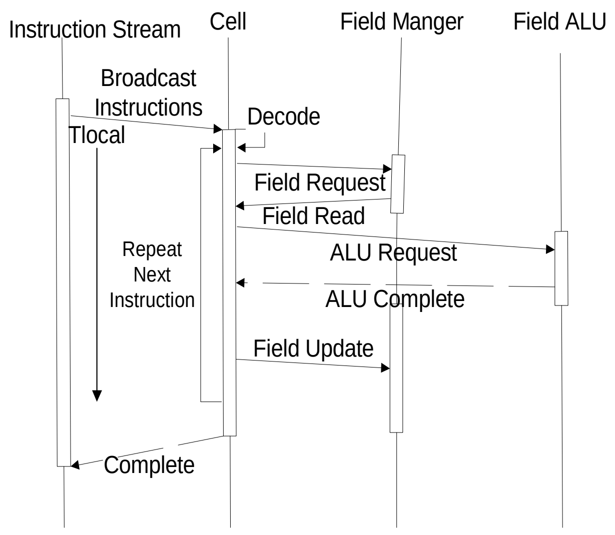
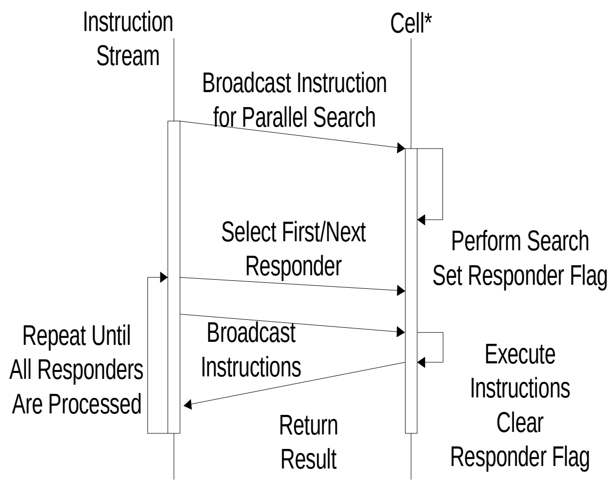


Figure 10: MASCOM predictability parameter T_{local} sequence diagram.



Instruction Stream
Cell*
Broadcast Instruction for Parallel Search
Select First/Next Responder
Perform Search Set Responder Flag
Repeat Until All Responders Are Processed
Broadcast Instructions
Execute Instructions Clear Responder Flag
Return Result

Figure 11: MASC description of the associative search-process-retrieve cycle.

6 Conclusions

This paper has presented an object oriented description and software framework for the MASC model of parallel computation. This framework provides a basis for many types of hardware and software developers to communicate and share ideas and concepts in an object oriented manner. The MASC model is a model of parallel computation that supports an associative style of parallel computation that allows memory to be addressed by content rather than by address. By using the various MASCOM structural and behavioral diagrams and views, the same model can be used for various disciplines of parallel computing research such as parallel architecture, parallel algorithm development, and implementations of parallel runtime environments. The class and structural diagrams of MASCOM provided the foundation classes and class aggregations. The object interactions and behavioral diagrams of MASCOM provided the responsibilities and requirements of the classes in the model. The object interactions led to the development of various sequence diagrams for computational predictability. The development of the MASC object model was instrumental in identifying duplicate responsibilities and classes among the various components of MASCOM; such as clarifying that an instruction stream and a cell have the same basic functionality, which is not illustrated in the MASC conceptual diagram.

The future of the MASC object model is currently being used as a development reference for implementing the MASC model using a cluster of workstations. However implementing MASC on other parallel computing hardware can use this same description as a reference. The MASCOM description can also be used as a common reference for comparing MASC with other models of parallel computation.

7 References

[1] Grady Booch, James Rumbaugh, and Ivar Jacobson, The Unified Modeling Language User Guide, Addison Wesley, Reading, MA, 1999.

[2] Bruce Powel Douglass, Real-Time UML: Developing Efficient Objects for Embedded Systems, Addison Wesley, Reading, MA, 1998.

[3] Falkoff, A. D., “Algorithms for Parallel-Search Memories”, Journal of the ACM, Vol. 9, Issue 4 (October 1962), pp. 488-511.

[4] Todd Heywood and Claudia Leopold, "Models of Parallelism", Abstract Machine Models for Highly Parallel Computers, John R. Davy and Peter M. Dew, Eds., Oxford Science Publications, Oxford, England, 1995, pp. 1-16.

[5] Jin, Mingxian, Johnnie Baker, and Kenneth Batcher, “Timings for Associative Operations on the MASC Model”, Proceedings of the 15th International Parallel and Distributed Processing Symposium (Workshop in Massively Parallel Processing), abstract on page 193, full text on CDROM, April 2001.

[6] Potter, Jerry L., Associative Computing: A Programming Paradigm for Massively Parallel Computers, Plennum Press, New York, NY, 1992.

[7] Potter, Jerry, Johnnie Baker, Stephen Scott, Arvind Bansal, Chokchai Leangsuksun, and Chandra Asthagiri, “ASC: An Associative Computing Paradigm”, IEEE Computer, Nov., 1994, pp. 19-25.

[8] Scherger, Michael C. Johnnie Baker, and Jerry Potter, “On Using the UML to Describe the MASC Model of Parallel Computation”, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Las Vegas, NV, June 2000, pp. 2639-2645.

[9] Scherger, Michael C. , Johnnie Baker, and Jerry Potter, “Using the UML to Describe the BSP Model of Parallel Computation”, Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2002), Las Vegas, NV, June 2002, pp. 578-583.

[10] Wu, Meiduo, Robert Walker, and Jerry Potter, “Implementing Associative Search and Responder Resolution”, Proceedings of the 16th International Parallel and Distributed Processing Symposium (Workshop in Massively Parallel Processing), abstract on page 246, full text on CDROM, April 2002.