A SOFTWARE IMPLEMENTATION OF A CYCLE PRECISION SIMULATOR OF A MULTIPLE ASSOCIATIVE MODEL

Wittaya Chantamas and Johnnie W. Baker
Kent State University
Department of Computer Science
Kent State University, Kent, OHIO 44242 USA
wchantam@cs.kent.edu and jbaker@cs.kent.edu

ABSTRACT

The Multiple Associative Computing (MASC) parallel model is a generalization model of an Associative Computing (ASC) parallel model designed to support multiple ASC data parallel threads by using control parallelism. The MASC model is designed to combine the advantages of both Single Instruction Stream Multiple Data Streams (SIMD) and Multiple Instruction Streams Multiple Data Streams (MIMD) models. Here is the first time that a complete description of MASC model has been implemented (in software) true to its original description. A cycle precision simulator is built to demonstrate the performance of MASC on various multithreaded algorithms. The simulator is a software prototype for the model with sufficient software details to allow it to be converted into a hardware prototype of the model. If a reasonable limit for the number of threads simultaneously supported is assumed, the resulting hardware design is not only easily to implement, but can easily support a huge number of processing units and is a excellent candidate architecture for supporting large scale (e.g., terascale and petascale) computing. Experimental results shows that, when processing large-scale instances using multiple workers, the algorithm executed by the MASC model using a static task assignment scheme provides strong scaling with constant time overhead.

KEY WORDS

Associative Computing, Joint Data and Control Parallelism, Model Simulation, Large Scale Computing

1. INTRODUCTION

The MASC model is a multi-SIMD model that uses control parallelism to coordinate the interactions of data parallel threads and supports "associative SIMD" execution of each of its threads. The ASC model is basically a SIMD parallel computer that has been enhanced (in hardware) so it can support a few basic reductions and operations in constant time and is more efficient and easier to program than SIMD. These constant time operations not only simplify the programming and the process of evaluating the complexity of algorithms but are extremely useful in parallel database operations and for applications such as air traffic control (which involves large dynamic databases). Each of the concurrent ASC executions of a task is performed by one of the MASC instruction streams (ISs) and the processors currently listening to this IS. In addition, the predictability of SIMD computers, which allows the worst case time to be calculated very accurately, is also an important feature of this model and is very important for real-time applications with critical deadlines. In fact, the ASC model was motivated by the STARAN associative SIMD parallel computer, which was designed by Kenneth Batcher and built by Goodyear Aerospace in the early 1970's for the air traffic control problem. A second generation version of the STARAN (the ASPRO) was used extensively by the Navy for an air defense system type application.

The original definition of MASC in ACM Communications in 1992 and subsequent publications provided a detailed specification of all aspects of the model other than information about how the communications and interactions between the multiple instruction streams can be supported and controlled. This paper provides a simulator that completely satisfies the original MASC model description and provides details about how MASC can support the instruction streams interactions by using a structure control scheme that is easy to implement. This approach allows MASC to preserve the properties of the ASC model it extends such as the predictable running time of programs. In a sense, this paper provides a completion of the MASC definition by providing an example of a more detailed MASC description that satisfies all of the original MASC requirements and is architecturally buildable. It can provide a showcase example of a more detailed description of MASC that is both simple and preserves all the desired properties of ASC.

Although a hardware prototype multithreaded associative SIMD (an alternative version of ASC) had been developed using FPGA by Schaffer [12] in 2007, no hardware

prototype of the MASC model has been developed yet. This is the first time that the MASC model has been completely implemented on a platform true to the original MASC description. Moreover, this paper provides a major extension of earlier work of Chantamas [3, 4], where their focus were to introduce the concept of using the manager-worker instruction stream paradigm to control interactions and communications between the ISs and an alternative method to produce MASC object codes directly from an ASC program (using the ASC programming language) for the MASC model. This paper completes the work on the MASC model with the manager-worker paradigm by presenting a complete description of using the manager-worker enhancement introduced in [3]. Additionally, an implementation of a new cycle precision MASC simulator to run MASC programs is provided. While the techniques used in this paper have been used with asynchronous systems, not much attention has been given to implementing synchronous systems that execute multiple data parallel processes concurrently, using SIMD computations to execute each of the data parallel processes. Coordinating and managing the communication and interaction between these SIMD processes in an efficient way and so that SIMD and ASC features (including predictability of execution time) are preserved is nontrivial.

This paper organizes into five main sections. Section one is this introduction. Section two provides the description of a MASC model consisting of manager and worker ISs. Section three describes the cycle precision software simulator. Section four provides the example multithreaded algorithm for the MASC simulator and its results. Section five concludes the work presented in this paper.

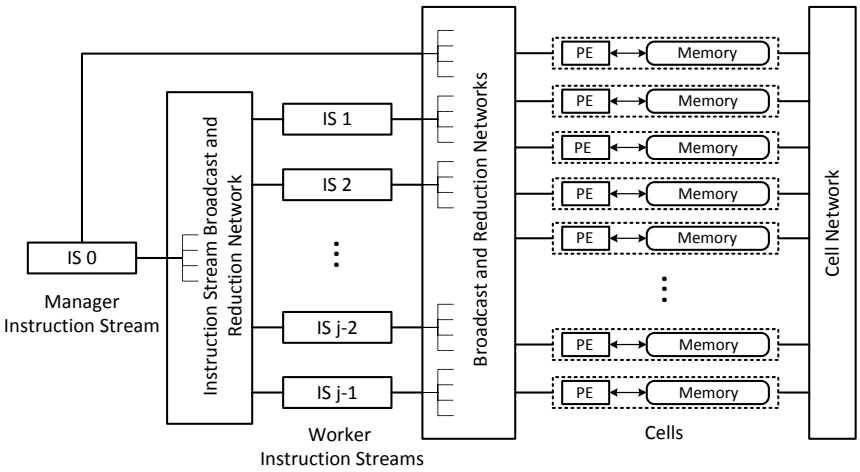

Figure 1. A MASC model using the manager-worker paradigm

2. THE MASC MODEL

As shown in Figure 1, a MASC model using the manager-worker paradigm consists of an array of processing elements (PEs), a number of ISs (one manager and a several workers), one broadcast and reduction network for each IS, and another broadcast and reduction network serving as the IS network. A MASC model with n PEs and m ISs is denoted as MASC(n,m).

All PEs are identical and are very simple, i.e., basically ALUs. Each PE, paired with its row of memory or local memory, is called a cell. The terms PEs and cells are, often, used interchangeably. Normally, a record of a set of data is stored in the memory of each cell. When the number of records is greater than the number of cells available, two or more records will be folded into one cell. The experimental results of both scenarios will be shown in section 4. Moreover, each cell had a mask register, usually a 1 bit register. The mask register indicates whether that cell is a responder (currently active) or not.

Historically, an instruction streams for a SIMD is called a control unit or a front end. Similar to a control unit of a SIMD computer, an instruction stream is a processor and able to fetch, decode, and broadcast instructions to its PEs. The number of ISs is expected to be considerably smaller than the number of PEs and corresponds to the number of SIMD threads that can be active at the same time. Both ISs and PEs have unique ID numbers and each knows its number. An IS may broadcast a value to PEs or read a value from a PE or PEs grouped under it.

A MASC model may have three types of networks, namely, a cell network for cell communications, an instruction stream network for instruction stream communications, and broadcast and reduction networks for communication between instruction streams and their sets of PEs. A cell network is an optional to the model as it has been shown by Trahan [13] that with or without cell network, the power of

the MASC model remains unchanged. The broadcast and reduction network is essential to the MASC model. It may be implemented using separate network circuits or sharing the same network circuit for both broadcasting and global reduction operations. In practice, the network can be constructed as a tree-structured set of resolver circuits as shown in Figure 2. Further details are given in [7].

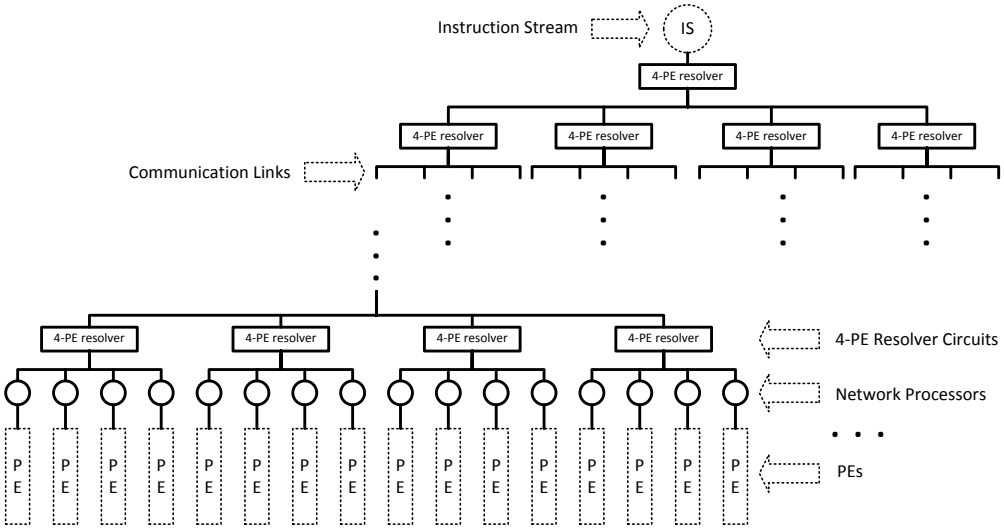

Figure 2. A broadcast and reduction network with an IS and PEs attached

The MASC model possesses certain constant time global properties such as constant time broadcasting, constant time global Maximum/Minimum and AND/OR reduction operations, and constant time associative search. These constant time global properties depend largely on the use of a broadcast and reduction network and the constant time timing was justified by Jin [7]. With these properties, the MASC model is not only able to solve a wide range of problems effectively [1][5] but also can provide solutions to problems in special areas such as real-time air traffic control in an extremely efficient manner, using worst case analysis to ensure that all deadlines are met [8]. These constant time operations not only simplify the programming and the process of evaluating the complexity of algorithms but are extremely useful in parallel database operations and for applications such as air traffic control. Each of the concurrent ASC executions of a task is performed by one of the MASC IS and the processors currently listening to this IS. In addition, the predictability of SIMD computers, which allows the worst case time to be calculated very accurately, is also an important feature of this model and is very important for real-time applications with critical deadlines. An associative language that supports the ASC model (also called ASC) has been implemented on a number of SIMD platforms by Potter [10, 11]. However a languages similar to C* designed for the Connection Machine [9] or C^n for ClearSpeed [15] can also support associative computing.

In [13], relationships between the MASC model and other computational models such as Reconfigurable Multiple Bus Machine (RMBM), Reconfigurable Mesh (RM), and the PRAM models have been established. Related models can be placed into two groups based on their power as follows.

- **Group 1:** ASC, MASC, Basic-RMBM, Segmenting-RMBM, PRAM, Basic-RM.
- **Group 2:** Fused-RMBM, Extended-RMBM, RM, Linear-RM.

Since all models in the same group have the same power and any model in Group 2 is more powerful that any model in Group 1, the MASC model has been shown to be powerful as ASC, Basic-RMBM, Segmenting-RMBM, COMMON CRCW PRAM, and Basic-RM models.

3. THE CYCLE PRECISION MASC SOFTWARE SIMULATOR

A cycle precision software simulator is built as a Win32 console application using C++ language running on a PC to allow the user to evaluate the efficiency of MASC on executing the algorithm on data of varying sizes and with a various number of ISs. A MASC C++ library was developed so MASC functions can be called from the library when a user wants to execute a MASC programs. The main MASC functions are global (AND/OR and MIN/MAX) reductions, an associative search operation, any-responder operation, and lastly, pick-one operation. More details of these functions are described in section 3.1.

This simulator is able to provide the exact number of operational steps the model requires to execute a given program. When a MASC program is executed, the number of operational steps taken by an algorithm is determined by counting the number of steps (the number of steps executing the task and the number of steps required during task synchronizations, if any) of the longest execution path of the algorithm. A basic operation (within the word length) such as arithmetic or logical, broadcast or reduction, and memory accessing operation is assumed to cost one operational step. Similarly, a complex operation consisting of j basic operations is assumed to cost j operational steps.

A parallel version of Floyd-Warshall all-pairs shortest path algorithm is used in section 4 to demonstrate the performance of the MASC model using a static task assignment scheme.

3.1 Simulating the MASC Properties

In contrast to a number of other parallel models and similar to the ASC model, the MASC model possesses certain constant time global properties such as constant time broadcasting, constant time global reduction operations, and constant time associative search. This section describes how these operations can be done in the simulator.

- **Global (AND/OR and MIN/MAX) Reductions:** The MASC model supports constant time global bitwise AND/OR reduction and Maximum/Minimum operations. For each group of 4 PEs, data are sent to its 4-PE resolver circuit. The resolver circuit does a reduction (AND/OR, MIN/MAX) operation and propagates a value to its next level resolver circuit. Next level 4-PE resolvers continue reducing values and propagate the values back up until the IS gets the final reduction result. Since we treated the whole broadcast and reduction network's gate delay as a constant time operation as justified by Jin [7] and each 4-PE resolver circuit does an operation in constant time, the MASC model supports a constant time global reduction operation.
- **An Associative Search Operation:** This operation can be performed as follows. An IS broadcasts an instruction to its PEs to execute a conditional expression. If a PE satisfies the condition, it sets its mask bit to 1. Otherwise, it resets its mask bit to 0. Since each step takes constant time, the associative search operation is a constant time operation
- **Any-Responder Operation:** This operation is usually performed after an associative search operation. An IS does a global OR reduction on mask bits of its PEs. If the returned result of the reduction is 1, then there is a responder. Otherwise, there is no responder. Since a global OR reduction takes constant time, the Any-Responder operation is a constant time operation.
- **Pick-One Operation:** This operation is usually performed after the previous Any-Responder operation returned 1 as the result. An IS does a global MAX (MIN) reduction on PE ID of its PEs, whose mask bits are 1. The returned result is the ID of a PE that will be selected. Later, the instruction stream may instruct that PE to reset its mask bit to 0 in order to avoid picking the PE again next round. Since a global MAX (MIN) reduction is a constant time operation, a Pick-One operation is also a constant time operation.

```
// Check for any responder by performing   // a Boolean OR
reduction of mask registers // of active PEs
bool MASC::AnyResponder(int t_id)
{
  bool found = false;

  found =
     BoolOrReduction(t_id, mask_register);

  return found;
}
```

Figure 3. A sample code of the any-responder function

3.2 Simulating the MASC Instruction Streams

The manager IS (or manager) can be simulated using 5-execution phase simulation cycles. The 5-execution phase consists of Finished, Fork, Assign, Join, and Termination phases. During a simulation cycle, some phases may be skipped but at least one phase must be simulated.

- **Finished:** The manager collects finished tasks from workers, if there is any finished task.
- **Fork:** The manager forks children tasks from prior finished parent tasks, if there is a fork task.
- **Assign:** The manager assigns new tasks from the task pool to idle workers, if there is a task and an idle worker.
- **Join:** The manager joins finished children tasks into one combined task, if there are to-be-joined tasks waiting.
- **Termination:** The manager checks for a terminal state. The program will be terminated if all of these conditions are true: the task pool is empty, no task is waiting to be forked or joined, and all workers are idle. Otherwise, the simulation starts at the finished phase again.

Worker ISs (or workers) can be simulated using 3-execution states. The 3 states are Ready, Busy, and Finish.

- **Ready:** This is the initial state for all workers. At this state, no PE is associated with a worker. The worker is idle and waits for the manager to assign it a task.

- **Busy:** A worker changes its state from Ready to Busy after the manager has assigned it a task. At this state, a task—a set of instructions along with a group of PEs—is assigned to the worker. In a rare case, the group of PEs may be an empty set. Nevertheless, the worker executes the assigned task following the flow of the program.
- **Finish:** After the worker has finished the assigned task and switched its PEs back to the manager, it changes its state from Busy to Finish. At this state, its PEs are no longer associated with the worker. After the manager has collected the finished task, the worker changes its state from Finish to Ready.

4. EXAMPLE MULTITHREADED MASC ALGORITHM

This section discuses the MASC Floyd-Warshall algorithm and its results first. The MASC model uses a static task assignment scheme to execute this algorithm. In this static task assignment scheme, assignments of tasks to instruction streams can be done simultaneously using a constant number of broadcasts to PEs and workers by the manager. Up to n concurrent tasks can be assigned to n worker instruction streams at a time for an input graph G with n vertices. The task assignment cost remains constant regardless of the number of assigned tasks generated by the algorithm.

Note that, not all algorithms can be used static task assignment scheme. The important characteristics of an algorithm to be used a static task assignment scheme are, first, the computation time per task (a partition of PEs and instructions) is constant and, second, the number of tasks is static for a given problem size. Mapping of problem tasks in the algorithm to instruction streams are predetermined (cannot be changed during runtime) and done statically.

The first set of results is from non fixed size MASC. Each cell will always contain only one record of a set of data in the memory. So, the input size of 32x32-matrix requires a MASC with 1024 PEs. The second set of results is from a fixed size MASC(64PEs, 1+8ISs). Two or more records will be folded into one cell, when the number of records is greater than the number of cells available. For example, in the case of 16x16-matrix input, a 2x2 or 4 records are folded into one cell.

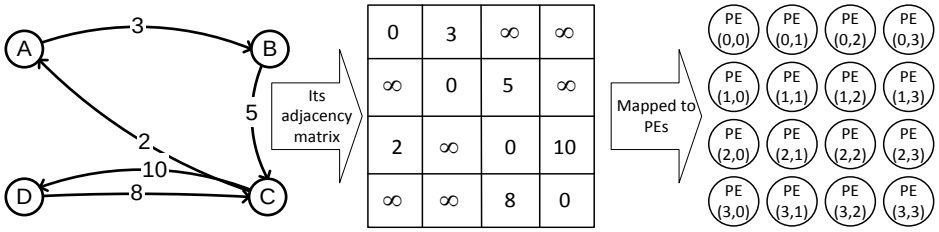

Figure 4. The adjacency matrix of a 4-vertex input graph is divided into 4^2 elements and mapped to 16 PEs

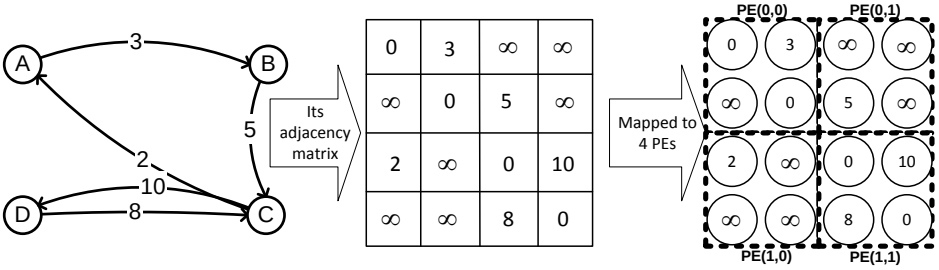

Figure 5. The same adjacency matrix is divided into 4^2 elements and mapped to 4 PEs

4.1 The MASC Floyd-Warshall All-pairs Shortest Path Algorithm and Its Performances on the Simulator

The Floyd-Warshall algorithm is an algorithm to find shortest paths between every pairs of vertices in a weighted directed graph purposed by Floyd [6]. The algorithm is based on a theorem by Warshall [14], which described how to compute a transitive closure of boolean matrices.

The algorithm solves the all-pairs shortest path problem by transforming a slightly modified adjacency matrix for the graph into a matrix whose (i, j) entry contains the shortest distance from v_i to v_j for all pairs of vertices. The slightly modified adjacency matrix used in this process has the weight $d(v_i, v_j)$ of the edge from vertices v_i to v_j in its (i, j) entry. In addition, $d(v_i, v_j)$ is set to 0 if i = j and set to ∞ if there is no edge between v_i and v_j.

The adjacency matrix of the input graph has to be altered with path estimates between identical vertices set to 0 and estimates between two vertices not jointed by an edge set to ∞ and becomes the input matrix A_0 consisting of its first approximation of path length using edge lengths. For a non fixed size MASC model, A_0 is divided into n^2 elements and each PE is responsible for an element of the matrix as shown in Figure 4. For a fixed size MASC model, A_0 is divided into n^2 elements and each PE is responsible for k^2 elements of the matrix as shown in Figure 5.

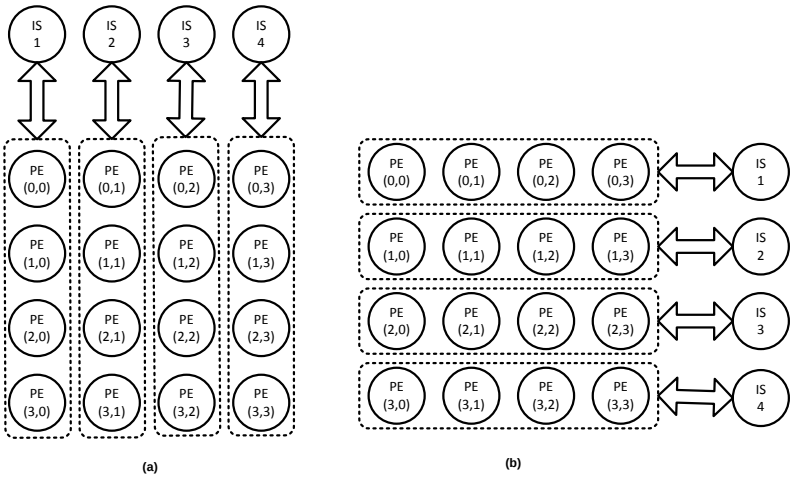

Figure 6. (a) Column-wise block-striped and (b) row-wise block-striped decompositions of the matrix

During the course of execution, PEs are partitioned into either row-wise or column-wise block-striped decompositions as shown in Figure 6. In general, each PE partition is controlled by a worker. Otherwise, two or more PE partitions may be assigned to each worker equally.

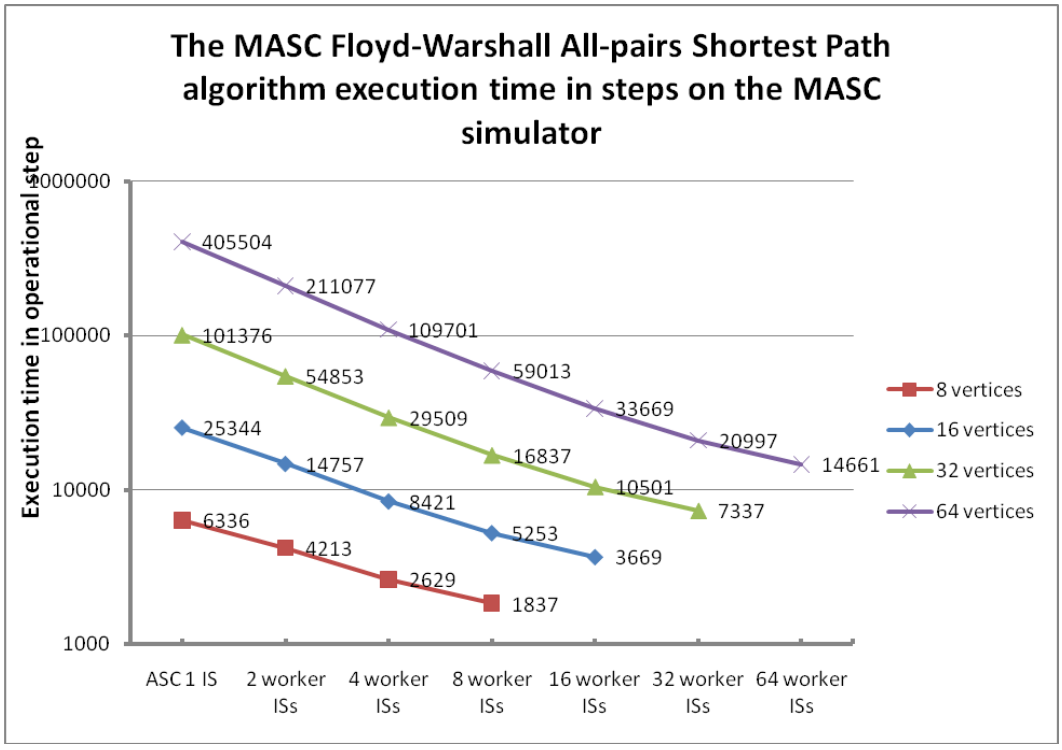

Figure 7. Results of the MASC implementation of the modified Floyd-Warshall algorithm on the simulator using non-fixed size MASC with 64, 256, 1024, and 4096 PEs on the input size of 64, 256, 1024, and 4096, respectively

4.2 The Performance of the MASC Floyd-Warshall Algorithm on the Simulator

The parallel performance is evaluated using scale-free graphs (R-MAT) [2] that represents unstructured data with the following parameters; a=0.17, b=0.55, c=0.18, and d=0.10 where a + b + c + d = 1 and using up to m workers for a graph with m vertices. In the first part of the experiment, a MASC(n^2 PEs, 1+m ISs) is used and the stripe size is n/m where n and m are powers of 2.

In this problem, the number of operational steps executed is independent to the input. Two different input sets (with the same size), when are executed on the same size MASC model, require the same number of operational steps. As shown in Figure 7, a bigger MASC (more worker instruction streams) will execute the MASC Floyd-Warshall algorithm (for the given input R-MAT graph) faster than a smaller MASC model does. When the problem size gets twice as big, i.e., from a 8-vertex R-MAT graph to a 16-vertex R-MAT graph, the execution time of an ASC quadruple, while the execution of a MASC does not quadruple. When using the maximum number of worker instruction streams allowed, the execution of a MASC increases about double. From this observation, one can conclude that a MASC model scales better to this Floyd-Warshall algorithm than an ASC model does. Also, using more workers will execute the algorithm (for a given input R-MAT graph) faster than using fewer workers and lowers the worker utilization (each worker doing less work).

Unfortunately, the maximum number of usable worker instruction streams is limited to |V|. In this implementation of the algorithm, using more than n = |V| worker instruction streams will not reduce the execution time since the maximum worker instruction stream tasks available is |V| tasks.

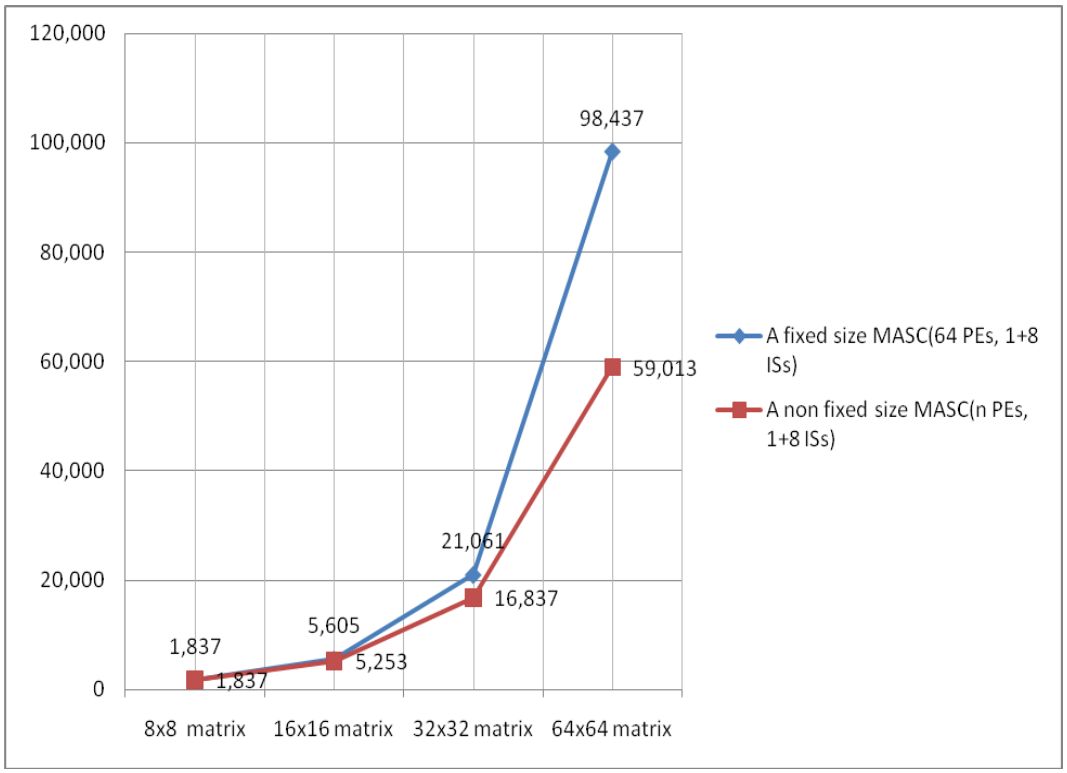

Figure 8. Results of the MASC implementation of the modified Floyd-Warshall algorithm on the simulator using a fixed size MASC(64 PEs, 1+8 ISs) comparing to those of a non-fixed size MASC(n PEs, 1+8 ISs) on the input size of 64, 256, 1024, and 4096, respectively

For a fiexed size MASC, folding of records in a cell increases the sequential execution portion of the program (each PE applies updat to each of its records one at a time). As shown in Figure 8, the more records are in a single cell, the slower the execution time when comparing with a MASC model with no folding of records. In reality, a fixed size system, not a non fixed size, is more practical and probably the one we have or will build. Modern supercomputers nowadays can be built using numerous numbers of processors from a few thousands cores to hundreds of thousand cores, i.e., the Jaguar system at Oak Ridge National Laboratory contains about 224 thousand of AMD Opteron cores and the JUGENE at Forschungszentrum Juelich (Germany) contains about 294

thousand PowerPC cores [16]. It is highly possible for someone to build a Multi-SIMD system with 256 thousand of processors since SIMD processors are much less complex than those CPU cores used in many of the systems in the Top 500 List. For this modified Floyd-Warshall algorithm, one processor of this 256K PE SIMD system is only taking care of 32 by 32 elements of the adjacency matrix of 2^{28} vertex graph.

5. CONCLUSION

We have successfully developed a software implementation of a MASC model that is true to MASC's original description using a cycle precision simulator. The simulator shows the ability of the MASC model with the manager-worker instruction stream paradigm to address a graph problem such as all-pairs shortest path problem using a static task assignment scheme. It can be concluded from the results that problems that can use a static task assignment technique perform very well using the MASC model and benefit from using the MASC model instead of the ASC model, which is a strict SIMD model. In particular, when processing large-scale instances using multiple workers, this algorithmic solution shows strong scaling with constant time overhead on this massively multithreaded problem. As a result, this algorithm scales better on the MASC model than on the ASC model.

Moreover, a MASC model using the manager-worker instruction stream paradigm combines the advantages of the SIMD and MIMD models by using control parallelism to support multiple ASC threads while maintaining both the scalability and predictability of the SIMD model with the improved flexibility. By using the manager-worker paradigm, the MASC model supports a large class of algorithms for both simple and massively multithreaded problems with better efficiency than a strict SIMD model, but results in some thread synchronization overheads.

REFERENCES

[1] M. Atwah, J. W. Baker, and S. Akl, An Associative Implementation of Classical Convex Hull Algorithm, *Proc. 8th IASTED Intl. Conf. on Parallel and Distributed Computing Systems*, Chicago, IL, October 1996. 435-438.

[2] D. Chakrabarti, Y. Zhan, and C. Faloutsos, R-MAT: A Recursive Model for Graph Mining, *Proc. fourth SIAM Intl. Conf. on Data Mining (SDM)*, Orlando, FL, April 2004.

[3] W. Chantamas, J. W. Baker, A Multiple Associative Model to Support Branches in Data Parallel Applications using the Manager-Worker Paradigm, *Proc. 19th IEEE Intl. Parallel and Distributed Processing Symposium (Workshop 14)*, Denver, CO, 2005, vol. 15, p. 266b.

[4] W. Chantamas, J. Baker, and M. Scherger, Compiler Extension of the ASC Language to Support Multiple Instruction Streams in the MASC Model Using the Manager-worker Paradigm, *Proc. PDPTA 2006*, Las Vegas, NV, June 2006, Volume 1. CSREA Press 2006.

[5] M. C. Esenwein and J. Baker, VLDC String Matching for Associative Computing and Multiple Broadcast Mesh, *Proc. the IASTED Intl. Conf. on Parallel and Distributed Computing and Systems*, Barcelona, Spain, 1997, 69-74.

[6] R. W. Floyd, Algorithm 97 (Shortest Path), *Communications of the ACM, 5(6):345*, 1962.

[7] M. Jin, J. Baker, and K. Batcher, Timing for Associative Operations on the MASC Model, *Proc. 15th IEEE Intl. Parallel and Distributed Processing Symposium (Workshop in Massively Parallel Processing)*, San Francisco, CA, 2001, 193.

[8] W. Meilander, J. Baker, and M. Jin, Importance of SIMD Computation Reconsidered, *Proc. 17th IEEE Intl. Parallel and Distributed Processing Symposium (Workshop on Massively Parallel Processing)*, Nice, France, 2003, 266a.

[9] J. Palmer, and, G.L. Steele, Jr., Connection Machine Model CM-5 System Overview, *Proc. Fourth Symposium on the Frontiers of Massively Parallel Computation*, McLean, VA, 1992, 474-483.

[10] J. Potter, *Associative computing: a programming paradigm for massively parallel computer* (Plenum Press, New York, 1992).

[11] J. Potter, J. W. Baker, S. Scott, A. Bansal, C. Leangsuksun, and C. Asthagiri, ASC: An Associative-Computing Paradigm, *Computer (27)*, 1994, 19-25.

[12] K. Schaffer and R. Walker, A Prototype Multithreaded Associative SIMD Processor, *Proc. 21st 19th IEEE Intl. Parallel and Distributed Processing Symposium (Workshop on Advances in Parallel and Distributed Computing Models)*, Long Beach, CA, 2007, 228.

[13] J. L. Trahan, M. Jin, W. Chantamas, and J. Baker, Relating the power of the Multiple Associative Computing (MASC) model to that of reconfigurable bus-based models, *Journal of Parallel and Distributed Computing, 70*, 2010, 458-466.

[14] S. Warshall, A Theorem on Boolean Matrices, *Journal of the ACM, 9(1)*, 1962, 11-12.

[15] http://www.clearspeed.com.

[16] http://www.top500.org.