A STUDY OF COMPUTER ARCHITECTURE TOPOLOGY SIMULATION USING NETWORK II.5, SIMSCRIPT II.5, AND MODSIM II

José L. Muñoz and Matthew J. Krzych

Naval Underwater Systems Center
New London, CT 06320

ABSTRACT

Network II.5, Simscript II.5, and ModSim II have been used extensively for the development of discrete-event models in support of a computer architecture assessment effort. With the development of over a dozen models simulating various computer system configurations, significant experience with these tools has been obtained. This paper focuses on three of these models and provides a modeling tool comparison that encompasses both functional and performance areas.

Network II.5, Simscript II.5, and ModSim II each provide the capability for assessing computer architectures although the underlying methodology behind each tool varies. Network II.5 is specifically designed for computer network simulation while Simscript II.5 and ModSim II are general purpose simulation languages with ModSim II supporting object-oriented programming. As a result, each has its own unique set of strengths and limitations that directly influence each model's ability to support modeling goals. The selection of a tool for a particular modeling effort is highly dependent upon these goals.

INTRODUCTION

Network II.5 (Network), Simscript II.5 (Simscript), and ModSim II (ModSim) have been used extensively in a computer architecture assessment effort that focused on the performance evaluation of computer architectures as applied to real-time combat systems [1]. Over a dozen models were developed from which significant experience with Network, Simscript, and ModSim was obtained. This paper focuses on three of these models and provides an assessment of the modeling tools used in support of this effort.

The principal objective of the assessment effort was to develop tool selection guidelines for computer architecture modeling efforts. This includes identifying strengths, weaknesses, and limitations of these tools. Often, the choice of using either Network, Simscript, or ModSim for a modeling effort is not apparent and in some cases, any may be used. A model's "effectiveness" including its ability to realistically model the actual system is restricted by the capabilities of the modeling tool. It is therefore critical that the appropriate tool be used.

MODEL DEVELOPMENT EFFORT

The tool assessment is based on work conducted in support of a research program whose goal was to develop a methodology and environment for the evaluation of future submarine combat systems [2]. In support of this goal, a substantial modeling effort was undertaken whose objective was to determine the effectiveness of a computer architecture to meet the requirements of a proposed combat system. As proof of principle and to serve as early prototype efforts, a sample problem with a set of candidate architectures was identified [3]. The goal of the prototype effort was to determine which of the architectures is best suited for solving the problem. In this case, the problem is to solve a NxN matrix multiply. Binary tree, mesh, and token ring topologies were chosen as the candidate architectures and are presented in Figures 1 through 3. The modeling effort goal was to determine which of these architectures is "best" suited for solving a NxN matrix multiply.

This paper provides a reference guide for the tool selection process particularly when applied to computer architecture simulation. The remainder of this paper is divided into three sections. The first section provides the foundation upon which the Network/Simscript/ModSim assessment is based and briefly describes the modeling effort used for the assessment. The second section describes the performance evaluation used to assess the run time performance of these tools. Included in this section is an assessment of the computers used to run the models. The third section documents the experiences obtained and identifies some of the strengths, weaknesses, and limitations. In addition to these sections, a brief description of these tools is provided in Appendix A.

INTRODUCTION

Network II.5 (Network), Simscript II.5 (Simscript), and ModSim II (ModSim) have been used extensively in a computer architecture assessment effort that focused on the performance evaluation of computer architectures as applied to real-time combat systems [1]. Over a dozen models were developed from which significant experience with Network, Simscript, and ModSim was obtained. This paper focuses on three of these models and provides an assessment of the modeling tools used in support of this effort.

The principal objective of the assessment effort was to develop tool selection guidelines for computer architecture modeling efforts. This includes identifying strengths, weaknesses, and limitations of these tools. Often, the choice of using either Network, Simscript, or ModSim for a modeling effort is not apparent and in some cases, any may be used. A model's "effectiveness" including its ability to realistically model the actual system is restricted by the capabilities of the modeling tool. It is therefore critical that the appropriate tool be used.

MODEL DEVELOPMENT EFFORT

The tool assessment is based on work conducted in support of a research program whose goal was to develop a methodology and environment for the evaluation of future submarine combat systems [2]. In support of this goal, a substantial modeling effort was undertaken whose objective was to determine the effectiveness of a computer architecture to meet the requirements of a proposed combat system. As proof of principle and to serve as early prototype efforts, a sample problem with a set of candidate architectures was identified [3]. The goal of the prototype effort was to determine which of the architectures is best suited for solving the problem. In this case, the problem is to solve a NxN matrix multiply. Binary tree, mesh, and token ring topologies were chosen as the candidate architectures and are presented in Figures 1 through 3. The modeling effort goal was to determine which of these architectures is "best" suited for solving a NxN matrix multiply.

This paper provides a reference guide for the tool selection process particularly when applied to computer architecture simulation. The remainder of this paper is divided into three sections. The first section provides the foundation upon which the Network/Simscript/ModSim assessment is based and briefly describes the modeling effort used for the assessment. The second section describes the performance evaluation used to assess the run time performance of these tools. Included in this section is an assessment of the computers used to run the models. The third section documents the experiences obtained and identifies some of the strengths, weaknesses, and limitations. In addition to these sections, a brief description of these tools is provided in Appendix A.

INTRODUCTION

Network II.5 (Network), Simscript II.5 (Simscript), and ModSim II (ModSim) have been used extensively in a computer architecture assessment effort that focused on the performance evaluation of computer architectures as applied to real-time combat systems [1]. Over a dozen models were developed from which significant experience with Network, Simscript, and ModSim was obtained. This paper focuses on three of these models and provides an assessment of the modeling tools used in support of this effort.

The principal objective of the assessment effort was to develop tool selection guidelines for computer architecture modeling efforts. This includes identifying strengths, weaknesses, and limitations of these tools. Often, the choice of using either Network, Simscript, or ModSim for a modeling effort is not apparent and in some cases, any may be used. A model's "effectiveness" including its ability to realistically model the actual system is restricted by the capabilities of the modeling tool. It is therefore critical that the appropriate tool be used.

MODEL DEVELOPMENT EFFORT

The tool assessment is based on work conducted in support of a research program whose goal was to develop a methodology and environment for the evaluation of future submarine combat systems [2]. In support of this goal, a substantial modeling effort was undertaken whose objective was to determine the effectiveness of a computer architecture to meet the requirements of a proposed combat system. As proof of principle and to serve as early prototype efforts, a sample problem with a set of candidate architectures was identified [3]. The goal of the prototype effort was to determine which of the architectures is best suited for solving the problem. In this case, the problem is to solve a NxN matrix multiply. Binary tree, mesh, and token ring topologies were chosen as the candidate architectures and are presented in Figures 1 through 3. The modeling effort goal was to determine which of these architectures is "best" suited for solving a NxN matrix multiply.

This paper provides a reference guide for the tool selection process particularly when applied to computer architecture simulation. The remainder of this paper is divided into three sections. The first section provides the foundation upon which the Network/Simscript/ModSim assessment is based and briefly describes the modeling effort used for the assessment. The second section describes the performance evaluation used to assess the run time performance of these tools. Included in this section is an assessment of the computers used to run the models. The third section documents the experiences obtained and identifies some of the strengths, weaknesses, and limitations. In addition to these sections, a brief description of these tools is provided in Appendix A.
Models of the binary tree, mesh, and ring were originally developed in Simscript. Soon after their development, all three models were converted to Network for validation purposes. As a sanity check, results of the Simscript models were compared to the Network results. It may be interesting to note that this comparison proved to be very insightful. Initial results from each Network/Simscript pair were substantially different. It was determined that this was a result of several misinterpretations on the use of Network features. After making the appropriate corrections, each Network/Simscript pair had near identical results.

The binary tree, mesh, and ring models were later converted to ModSim as part of a ModSim evaluation effort. As with the Network/Simscript conversion, the ModSim models had initially produced results inconsistent with their Network/Simscript counterparts. These inconsistencies were later removed.

Tables 1 through 3 provide a summary of each model's size. The Network Source File Size is defined to be the total number of lines contained within the Network "NET" file and was obtained from the NETWORK utility after the "NET" file had been loaded. Simscript and ModSim Source Lines of Code are defined as the total number of lines contained within the source file. This includes executable and preamble code and excludes input data and comments.

<table>
<thead>
<tr>
<th>NETWORK II.5</th>
<th>SIMSCRIPT II.5</th>
<th>MODSIM II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Elements</td>
<td>37</td>
<td>Resources 179</td>
</tr>
<tr>
<td>Transfer Devices</td>
<td>61</td>
<td>Routines 2</td>
</tr>
<tr>
<td>Modules</td>
<td>138</td>
<td>Processes 9</td>
</tr>
<tr>
<td>Source File Size (Lines)</td>
<td>4,039</td>
<td>Source Lines of Code 593</td>
</tr>
</tbody>
</table>

Table 2 MESH SIZING TABLE

<table>
<thead>
<tr>
<th>NETWORK II.5</th>
<th>SIMSCRIPT II.5</th>
<th>MODSIM II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Elements</td>
<td>12</td>
<td>Resources 46</td>
</tr>
<tr>
<td>Transfer Devices</td>
<td>23</td>
<td>Routines 1</td>
</tr>
<tr>
<td>Modules</td>
<td>40</td>
<td>Processes 7</td>
</tr>
<tr>
<td>Source File Size (Lines)</td>
<td>1,755</td>
<td>Source Lines of Code 327</td>
</tr>
</tbody>
</table>

Table 3 RING SIZING TABLE

For each model described above, the Network version and its Simscript and ModSim counterparts are considered functionally equivalent. That is, each version models the same system components at the same level of detail and all three versions produce all the outputs needed to support their modeling requirements. It should be noted that outside the scope of this definition differences do exist between model versions. This can be attributed to inherent differences that exist between the tools. Network provides many additional services to the user that are not supported by either Simscript or ModSim. This includes, but is not limited to: (a) user 'friendly' interface, (b) a variety of detailed output reports, (c) model 'verification' (the NETIN verify command), (d) debug tools including trace and snapshot functions, (e) hardware interconnection diagrams, and (f) timeline reports [4]. In short, Network is a modeling tool while as Simscript and ModSim are programming languages.

PERFORMANCE EVALUATION

A principal focus of the assessment effort was to evaluate tool performance. This had two main objectives the first of which was to assess the performance of available computer resources that include: (a) Vax 785, (b) uVax II, (c) Sun 3/160, (d) Sun 3/260, and (e) Cray X-MP/28. From this evaluation we sought to determine which of these computers is best suited for running Network, Simscript, and ModSim models. The second objective of the performance evaluation was to assess the performance of Network, Simscript, and ModSim. Based solely on their computer performance, we sought to determine which tool should be used for model development.

Appendix B provides a description of the computer resources used in the assessment. It should be noted that a beta version of ModSim was used and was only available on the Sun workstation. In addition, Network was not available on the Cray.

All three models identified in the previous section (binary tree, mesh, and ring) were used in the performance evaluation. The results of running their Network, Simscript, and ModSim versions on the available computers are listed below.
Computer Performance

Three metrics were used to evaluate computer performance and include: (a) CPU time, (b) storage requirements, and (c) compile and link time.

CPU Time  The amount of CPU time required to run each model is presented in Figures 4 through 6. In each figure, CPU time is represented in seconds.

Figure 4 provides the CPU times consumed by the Network versions of the binary tree, mesh, and ring models. In all cases, the Sun 3/260 provided the fastest run times while the Micro Vax provided the slowest. In general, the Sun 3/260 was nearly twice as fast as the Sun 3/160 and almost three and a half times faster than the Micro Vax.

As might be expected, the Cray provided the fastest run times for all three Simscript models as demonstrated in Figure 5. The Cray was four times faster than the next fastest computer, the Sun 3/260, and 27 times faster than the slowest, the Micro Vax.

Figure 6 provides the results of the ModSim CPU time analysis. As with the Network and Simscript results, the Sun 3/260 was nearly twice as fast as the Sun 3/160. As previously mentioned, ModSim was only available for the Suns.

Storage Requirements  Figures 7 and 8 describe the storage requirements needed for each model and were obtained from the Network.NET and the Simscript executable files. Although there were insignificant differences between computers for the Network models, the Vax/uVax provided better than 100% improvement in Simscript storage requirements over the Sun workstations and 200% improvement over the Cray.
As with the Network and Simscript models, the ModSim models performed better on the Sun 3/260 than on the Sun 3/160.

**Network II.5. Simscript II.5. and ModSim II Performance**

CPU time was the only parameter used to evaluate Network, Simscript, and ModSim performance and is presented in Figures 11 through 13. In all cases, Simscript outperformed both Network and ModSim however, there was a wide degree of variance between the run time performance of these tools. The greatest discrepancy occurred when comparing the run time performance of a Simscript model against its Network counterpart. At most, Simscript's performance was 75 times faster than Network's which occurred when running the binary tree model on the Sun 3/160 (Figure 11). This is in contrast to a two and half time improvement achieved when running the ring model on the Vax (Figure 13).

The variance in performance between Simscript and ModSim was not as great as that of Simscript and Network. As demonstrated in Figure 13, Simscript was at most five times faster than ModSim.

**Simscript Compile Time**

![Simscript Compile Time Graph](image)

**ModSim Compile Time**

![ModSim Compile Time Graph](image)

**Binary Tree Model CPU Time**

![Binary Tree Model CPU Time Graph](image)

**Mesh Model CPU Time**

![Mesh Model CPU Time Graph](image)

It should be noted that the development of a Network model does not require compilation or linking and therefore, compile and link times for the Network models have not been provided.

**Computer Performance Conclusions** Based on the results obtained from this evaluation, the Sun 3/260 and Sun 3/160 would be the better choices for running Network models. This is based on the faster run times provided by both of these computers. Storage requirement was not a factor due to the minor differences between computers.

The Cray and Sun 3/260 provided the best performance for running Simscript models although both these systems were less efficient than the Vax/uVax in storage requirements.
Network II.5, SIMSCRIPT II.5, MODSIM II

ASSESSMENT

The following assessment is based on the experiences obtained from the modeling effort described above.

Network II.5 Strengths

By providing a set of “building blocks” and a user-friendly interface, Network supports the development of software models “without programming”. As a result, “programmers” are not required for model development. This provides flexibility in staffing as well as avoiding the problems associated with programmer shortages.

Network is a modeling tool tailored for computer network analysis. The building blocks used to define a Network model closely resemble actual computer elements (e.g., storage device, processing element) that allow for a smooth transition between the real system and the model. Only a minimal effort is required in determining how to represent system elements within the model.

Because of the similarities between Network building blocks and actual system elements, Network is extremely easy to learn. In addition, all house-keeping functions such as simulation clocks and statistics gathering functions are automatically performed by Network. The user can therefore focus more of his attention on learning how to model the system and less on how to perform house-keeping tasks.

In general, Network models are maintainable. NETPLOT, a Network utility, can provide insight into a model’s construction by generating hardware and software diagrams of the model’s architecture. In addition, Network provides a standard definition of terms which is documented in their user manuals. Both the model developer and maintainer are required to use the same set of definitions.

As previously mentioned, Network provides a number of built-in functions which help reduce model development time and allow the modeler to focus his efforts on modeling the actual system.

All of the Network models were easily ported between the Vax 785, uVax II, and Sun workstations.

Simscript II.5 Strengths

Simscript is a programming language tailored for general-purpose simulation and as such, can provide greater flexibility in modeling a wider variety of systems. Simscript does not restrict either the type (e.g., factory, banking systems, etc.) or the scale and depth at which a system can be modeled.

In addition to providing flexibility, Simscript also provides the capability to make a model modifiable. By partitioning hardware, software and message constructs properly, Simscript models can be easily updated and enhanced.

Through the use of a variety of constructs supported by Simscript such as subroutines or arrays, it is possible to define a number of identical system elements within a single piece of code. This helps minimize the model size and contributes to the model's modifiability and maintainability.

As demonstrated in the previous section, Simscript models have faster run times than Network or ModSim models.

As with Network models, all Simscript models were easily ported between the Vax 785, uVax II, Sun workstations, and Cray.

Network II.5 Weaknesses

Making structural modifications to a Network model, such as changing its hardware topology or its work load, is extremely difficult and can be attributed to the lack of clear partitioning of hardware, software, and message definitions. Within a Network model, these definitions are tightly coupled with one another. As a result, modifying any single definition typically has an impact on the other definitions requiring additional updates. As an example, making modifications to a hardware definition in order to reflect changes in topology would require modifications to software and message definitions.

Network supplies a limited set of building blocks with associated semantics from which all Network models must be defined. In most cases this set is adequate, but on occasion, Network fails to provide the necessary building block. This problem is related to model complexity where the more complex the model, the more evident the problem. An example of this deficiency would be Network’s inability to support the concept of a “variable” or a subroutine.

As a result of Network’s failure to support any subroutine–like structures, most models require the duplication of code. Quite often the actual system will contain a number of elements which are functionally identical. Ideally, when modeling these elements, only one definition (or portion of code) characterizing the set should exist. Through subroutine calls or through indexing into an array, each element could be “instantiated”. Unfortunately with Network, each element must be individually defined and created. Although duplicating code is not difficult, it can be time consuming and can restrict the modifiability of a model.

As mentioned in the previous section, Network models have slow run times particularly when compared to Simscript.
Simscript II.5 Weaknesses

Simscript is a complex language which makes it very difficult to learn. This is especially true if one is not familiar with modeling languages or concepts. The approach used for developing a Simscript model differs significantly than that of other programming languages such as Fortran and requires the programmer to be familiar with such concepts as event and process scheduling. In addition, Simscript contains many options embedded within its instruction repertoire that although they may add to the program's readability, add to the complexity of the language.

Because Simscript is a programming language, programmers are required for model development. This can be a significant problem particularly if Simscript programmers are unavailable. As mentioned in the previous section, training people is more difficult and can be very time consuming.

Although Simscript documentation provides five reference books, no single book can be used for all referencing requirements. Information is scattered across the five references and can be difficult to find [5], [6], [7], [8], [9].

ModSim II Strengths

ModSim is a general purpose programming language specifically tailored to support process oriented simulation. As a result, a great deal of flexibility is available in modeling a wide variety of systems. ModSim is not restrictive in the category (eg. banking systems, war gaming, factory situations, etc.) or in the scale and the depth at which things can be modeled.

ModSim supports the object-oriented paradigm. Using this capability it is possible to define object libraries that are reusable. Using these libraries and object inheritance, one can readily modify systems and/or create new configurations. This was done with the binary tree, mesh and ring models developed for this paper. Processor and link objects were defined that were then replicated and configured into the various topologies.

The only code requiring change was the code that characterized the communication behavior and termination conditions for that particular topology. For example, a processor in the mesh is required to communicate both to its neighbor on the right and the neighbor below while the ring is only required to communicate to its neighbor to the right. Object definition and reusability, along with the associated ModSim "methods", gives ModSim a moderate complexity of use. While reusable code is possible in Simscript, it is natural, directly supported and encouraged in ModSim.

ModSim's "smart" compiler with its built-in implied "make" facility and the top-down approach to software development encouraged by object-oriented programming were found to be valuable assets to the software development effort.

Unfortunately, it was not possible to address ModSim's portability capability, i.e., the facility with which it is possible to take code from one machine such as a Sun workstation and move it over to a different machine such as a Vax. The Vax version of ModSim was not available at this time; portability is an area we expect to assess in the near future. Because ModSim generates C code that is then passed on to the native C compiler, portability is not expected to be a problem.

In every case, the ModSim models executed faster than the Network models but slower than the Simscript models (ModSim models were 2-5 times slower than Simscript models). As previously mentioned, the version of ModSim used was still in beta testing and therefore lacked the maturity of Simscript. As ModSim matures and is "fine tuned" these execution times are expected to improve. ModSim's capability of taking advantage of a parallel processing environment (still under development) should provide further improvements.

ModSim II Weaknesses

ModSim is a programming language and is therefore more difficult to learn than a tool such as Network. Object-oriented programming, supported by ModSim, is a relatively recent development. This implies that the expected number of programmers that can take full advantage of ModSim's capabilities, at least initially, will be limited.

ModSim's current documentation consists of a user's manual [10] and a tutorial [11]. These rely heavily on a user having prior experience with object oriented programming and are therefore not sufficient. It is expected that these will be augmented as the language matures.

CONCLUSIONS

Network, Simscript, and ModSim are each capable of being used for assessing the performance of an architecture. Network is specifically designed for the simulation of computer networks, while Simscript and ModSim are general purpose simulation languages with ModSim specifically designed around the object-oriented programming paradigm.

As a result, each has its own unique set of strengths and limitations that directly impact any selection criteria. The effectiveness of a model to meet its objectives will be largely determined and restricted by these characteristics. It is therefore imperative that the tool selection process consider (a) modeling objectives, (b) life-time of the model, (c) how likely are characteristics of what is being model to change, (d) personnel skills and, (e) what hardware is available. Based on the results discussed in this paper and the above criteria:

Network was found to be effective under the following circumstances:

(a) For rapid model development - Network supports a number of built-in functions such as report generators and random number generators that, when used, reduce development time and effort.

(b) When model complexity is low - The complexity of a model is restricted by the limited set of building blocks supported by Network. In general, models which require conditional logic (ex: if-then-else, while-do, etc.), especially when nested, should NOT be developed in Network. Network does not adequately support these type of constructs.

(c) When staff consists of non-programmers - Network provides a user interface that supports the user for modeling characteristics. All code is then automatically generated by Network and there by eliminating the need for programmers.

(d) When staff is inexperienced with modeling - Network is easy to learn and use.
Simscript was found to be effective under the following circumstances:

(a) When model complexity is high - Simscript has a robust syntax that supports the development of intricate models.

(b) When periodic modifications to a model are required - As a result of Simscript's flexibility, models can be coded and partitioned such that they can be easily modified and enhanced.

(c) When schedule and staff are available to support development - Developing a Simscript model is a software development effort that will require Simscript programmers and a schedule that supports the full software life-cycle.

ModSim was found to be effective under the following circumstances:

(a) When model complexity is high - ModSim is a robust language that can be used to develop a wide variety of significantly complex models at varying levels of detail. ModSim was specifically designed with large scale simulation in mind.

(b) When significant modifications to a model are required or expected - ModSim's flexibility inherent in its integrated program development approach, modular program construction and in its object-orientation all serve to make the task of model modification less burdensome.

(c) When schedule and staff are available to support development - The development of a ModSim model is a software development effort that will require ModSim programmers and a schedule that can support the full software life-cycle.

(d) When model execution time is expected to be significant - ModSim is targeted to be ported to a parallel processing environment. The extent to which ModSim can take advantage of such an environment and the degree of expected performance improvements is currently not assessable. This will, of course, depend significantly on the model and the modeling semantics used.

The assessment of simulation tools is expected to be an ongoing process. In the future, we expect to include additional languages and tools and additional selection criteria such as data visualization, animation, and distributed simulation. The intent is to provide a simulation capability that is an integral part of an architecture assessment environment.

ACKNOWLEDGEMENT

Acknowledgement is gratefully extended to the Very Massively Parallel Acoustic Architectures staff especially Ms. L. Karasevich and to Barbara Donovan and Ron Belanger of CACI, Inc.

REFERENCES


APPENDIX A

NETWORK II.5, SIMSCRIPT II.5, MODSIM II OVERVIEW

This appendix provides a brief overview of Network II.5, Simscript II.5, and ModSim II. All three of these tools are used for discrete event simulation and support the modeling and analysis of computer architectures. Network II.5 and Simscript II.5 run on a variety of computers including VAX, MICROVAX, IBM-PC, SUN WORKSTATIONS, and a number of other systems. ModSim II is currently restricted to running on SUN WORKSTATIONS and IBM-PCs.

A.1 Network II.5

Network II.5 is a computer and communications network analysis tool that supports the development of software models "without programming". This is achieved by providing the user with three individual utilities: NETIN, NETWORK, and NETPLOT. NETIN provides the user with an interface for developing a data file that characterizes the system to be modeled. The data file is used as input to NETWORK which performs the actual simulation. NETWORK produces user simulation reports and a timeline data file (optional). The timeline data file is input to NETPLOT which produces timelines of various utilizations.
The user defines the model through the use of eight Network II.5 building blocks. Each building block contains a series of attributes that are also defined by the user. These building blocks include: (a) processing elements, (b) storage devices, (c) transfer devices, (d) modules, (e) instructions, (f) files, (g) messages, and (h) statistical distribution functions.

A.2 Simscript II.5

Simscript II.5 is a procedural-oriented programming language tailored to support general-purpose simulation. The language provides a number of constructs that enables the language to be flexible and effective in modeling a wide variety of systems. In addition, Simscript supports both event-oriented and process-oriented simulation. Simscript is a true programming language in that code must be developed, compiled and linked before a model can execute. As a result, Simscript models must be developed by someone familiar with programming.

A.3 ModSim II

ModSim II is a general purpose, block structured, simulation language that provides support for object-oriented programming and discrete event simulation. It is intended to be used for large scale, process based (i.e. a simulation model written in ModSim describes a system in terms of processes) simulations developed using modular and object-oriented programming development techniques. The object-oriented capabilities of ModSim allow it to be used in a parallel processing environment.

ModSim provides a "smart" compiler that incorporates an implied 'make' facility so that only code requiring recompilation is recompiled. In this manner users are not required to have to determine what modules require recompilation as a result of modifications to another module. The version of ModSim used in this paper was a beta-test version and was therefore still in development although a great many of the language features had been defined.

APPENDIX B
COMPUTER CHARACTERISTICS

<table>
<thead>
<tr>
<th>Vax 785</th>
<th>MicroVax II</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMS v5.02</td>
<td>VMS v5.1</td>
</tr>
<tr>
<td>8 MBytes RAM</td>
<td>9 MBytes RAM</td>
</tr>
<tr>
<td>2.1 GByte Disk</td>
<td>1 GByte Disk</td>
</tr>
<tr>
<td>Network II.5 v4.01</td>
<td>Network II.5 v4.01</td>
</tr>
<tr>
<td>Simscript II.5 v5.1</td>
<td>Simscript II.5 v5.1</td>
</tr>
<tr>
<td>1.5 MIPS rating</td>
<td>1.0 MIPS rating</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUN 3/260</th>
<th>SUN 3/160</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun Unix 4.2 Rel 3.5</td>
<td>Sun Unix 4.2 Rel 3.5</td>
</tr>
<tr>
<td>16 MBytes RAM</td>
<td>16 MBytes RAM</td>
</tr>
<tr>
<td>300 MByte Disk</td>
<td>280 MByte Disk</td>
</tr>
<tr>
<td>Network II.5 v4.01</td>
<td>Network II.5 v4.01</td>
</tr>
<tr>
<td>Simscript II.5 v1.3</td>
<td>Simscript II.5 v1.3</td>
</tr>
<tr>
<td>ModSim II v0.71</td>
<td>ModSim II v0.71</td>
</tr>
<tr>
<td>4 MIPS rating</td>
<td>2 MIPS rating</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cray X-MP/28</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNICOS v4.0</td>
</tr>
<tr>
<td>2 CPUs</td>
</tr>
<tr>
<td>8 MWord (64bits/word) RAM</td>
</tr>
<tr>
<td>128 MWord Solid State Storage Device</td>
</tr>
<tr>
<td>Simscript II.5 v1.0</td>
</tr>
</tbody>
</table>
VARIATIONS OF THE PHASE-ONLY FILTER

Joseph L. Horner
Rome Air Development Center
Solid State Sciences Directorate
Hanscom AFB, MA 01731

An alternative to the classical matched filter has been shown to be useful for pattern recognition and image correlation - the phase-only filter. It exhibits extremely narrow correlation peaks with very small sidelobes, better discrimination between similar objects and 100% throughput when implemented optically. We show and discuss several alternatives derived from the phase-only filter and implementable optically on both phase or amplitude modulating real-time devices. Performance is tested through computer simulation and optical bench experimentation. Results are shown for signal-to-noise ratio and discrimination ability.