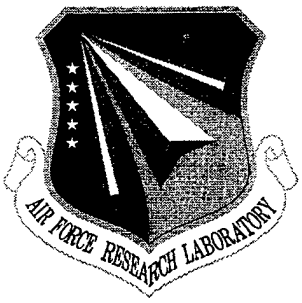
RL-TR-97-222
Final Technical Report
February 1998



IMPACT OF BISECTION BANDWIDTH CONSTRAINTS ON SOFTWARE DEVELOPMENT COSTS

University of Colorado

Sponsored by
Advanced Research Projects Agency
ARPA Order No. B667

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-97-222 has been reviewed and is approved for publication.

APPROVED:

JAMES DAVIS
Project Engineer

FOR THE DIRECTOR:

DONALD W. HANSON, Director
Surveillance & Photonics Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/SNDP, 25 Electronic Pky, Rome, NY 13441-4514. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

ALTHOUGH THIS REPORT IS BEING PUBLISHED BY AFRL, THE RESEARCH WAS ACCOMPLISHED BY THE FORMER ROME LABORATORY AND, AS SUCH, APPROVAL SIGNATURES/TITLES REFLECT APPROPRIATE AUTHORITY FOR PUBLICATION AT THAT TIME.

IMPACT OF BISECTION BANDWIDTH CONSTRAINTS ON SOFTWARE DEVELOPMENT COSTS

Wes C. Berseth
John A. Neff

Contractor: University of Colorado
Contract Number: F30602-92-2-0234
Effective Date of Contract: 9 July 1996
Contract Expiration Date: 9 April 1997
Program Code Number: 6Y10
Short Title of Work: Impact of Bisection Bandwidth Constraints on Software Development Costs

Period of Work Covered: Jul 96 - Apr 97

Principal Investigator: Wes Berseth
Phone: (303) 492-7135

AFRL Project Engineer: James Davis
Phone: (315) 330-4276

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by James Davis, Air Force Research Laboratory/SNDP, 25 Electronic Pky, Rome, NY 13441-4515.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED	
	February 1998	Final	Jul 96 - Apr 97

4. TITLE AND SUBTITLE

IMPACT OF BISECTION BANDWIDTH CONSTRAINTS ON SOFTWARE DEVELOPMENT COSTS

5. FUNDING NUMBERS

C - F30602-96-2-0234
PE - 62712E
PR - B667
TA - 00
WU - 01

6. AUTHOR(S)

Wes C. Berseth and John A. Neff

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Colorado
NSF Engineering Research Center
Campus Box 525
Boulder, CO 80309-0525

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington, VA 22203-1714

AFRL/SNDP
25 Electronic Pky
Rome NY 13441-4514

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-97-222

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: James L. Davis/SNDP/(315) 330-4276

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report discusses the costs of developing software as a function of bisection bandwidth constraints of hardware configurations. System development experts were surveyed to obtain data to help quantify effects of developing software in bandwidth limited applications. A discussion of software cost estimating models is given and a case is made for including bisection bandwidth in using these models. The advantages of free-space optical interconnects are enumerated. The report concludes that availability of high-bandwidth interconnects, especially in intensive military real-time signal processing applications, could lead to substantial savings in system development and maintenance costs.

14. SUBJECT TERMS

Software development, metrics, cost estimates

15. NUMBER OF PAGES

44

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

Abstract

Demand for additional functionality in real time embedded multiprocessor applications is steadily increasing. This, in turn, is causing software complexity and cost to increase. Consequently, software is comprising a greater proportion of the costs of applications, accounting for as much as 70-80 % of the total development cost.

The impact of hardware constraints on software productivity is recognized in existing parametric software cost estimating models. These models, developed for uniprocessors, consider the effects of processor and memory constraints, but do not consider bandwidth constraints.

Experts experienced in the development of communication constrained multiprocessor applications were surveyed to determine the impact of the interconnection network on software productivity and application development. The general impact of networks on software development can be captured by determining the impact of bisection bandwidth constraints since bisection bandwidth and connectivity are related through network topology. When bisection bandwidth is constrained, software productivity is affected as much as, or more than, when processing or memory is constrained, and can cause software development costs to double or triple. This can amount to a substantial increase in the cost to develop an application; potentially millions of dollars. The capability of the application must often be reduced and the resulting code is more complex and harder to debug. Also, the maintenance phase is affected as fixing defects, and adding functionality becomes more difficult as bisection bandwidth utilization is increased.

These results indicate significant advantages to implementing interconnection networks with high bandwidth. For large DSP applications substantial savings can be realized in both software development and maintenance costs. Greater bandwidth can increase capability by enabling the implementation of algorithms with high communication requirements, such as high resolution SAR, STAP, and advanced ATR. Also, greater bandwidth may enable efficient implementation of shared memory architectures which are easier to program than message passing systems. For these reasons it may be advantageous to invest in more expensive interconnection networks or in developing optical interconnection technologies such as free-space optics.

We view this work as providing a basis for extending existing parametric software cost estimating models to describe real time embedded multiprocessor systems. With such a model, it would be possible to perform trade analyses between major multiprocessor hardware components (processing, memory and interconnection network) to optimize system cost and performance. This would lead to substantial savings in system development and maintenance costs, better performance and easier upgrades.

Acknowledgments

This work was carried out under the auspices of the Optoelectronic Computing Systems Center at the University of Colorado and supported by contract F30602-96-2-0234, managed by Rome Laboratory/OCPC.

A number of people have contributed to this work and are listed in Appendix C. Special credit goes to Bill Wren, of Honeywell Technology Center, for valuable insights concerning software estimating models and extending these models to real time embedded multiprocessor systems, and Jim Otte, of PRICE Systems, for useful discussions on cost estimation relationships in parametric models and software effort data. We also wish to acknowledge Barbara Yoon for providing information on application domains to pursue and people to contact.

1 Introduction and Background

1.1 Software Growth in Real Time Embedded Multiprocessor DSP Applications

Radar and imaging are extremely important for military applications such as surveillance, intelligence and automatic target recognition. These applications have demanding requirements in computation and communication and often have stringent requirements on size, weight and power as well. The major trend for these applications, and embedded applications in general, is the demand to provide greater functionality while reducing costs and cycle times. This was expressed by many at the Embedded Systems Conference in San Jose in September, 1996, (e.g. Bunza, 1996). As a consequence, system design and development are becoming more complex, while users are demanding that systems be easy to work with and program. The greater functionality, which includes fault tolerance and security requirements, leads to increased complexity of the system, much of which is implemented in software (e.g. Bartow, 1995; Bunza, 1996). As a result, software is becoming the dominant component of an application and, an increasingly greater proportion of the total development cost (e.g. Bunza, 1996; Douglas, 1996). Discussions with development experts revealed that software costs are typically greater than hardware costs and often consume 70-80% of the total development cost. For large DSP applications software is generally over $1M and can be as high as $3, $4 and $15 M. A study by Eagle Design Automation revealed that the ratio of software to hardware engineers is commonly 4:1 and is steadily increasing (Geoff Bunza, pers. comm.). This trend can severely impact the US military which relies heavily on software for their capability, investing billions of dollars in software annually. This is evidenced by the fact that the military software industry employed more than 475,000 of the 2,400,000 software professionals in the United States in 1995 (Jones, 1997).

1.2 Interconnection Networks

With demands for greater functionality in real time applications increasing, connectivity and bandwidth requirements are beginning to push the limits of electronic interconnection networks. This has led to compromised topologies, e.g. Fat Tree, Mesh, n-cube, in which processors must share available links. Consequently, algorithms that require extensive communication between distant processors (e.g. 2-D FFT) experience high levels of contention and latency when the network is heavily loaded resulting in poor performance. Another problem with conventional electrical networks is a scaling limitation due to greater link sharing as processor number is increased. For example, as the number of processors of a NxN mesh is increased, the number of processors that must share each link also grows by N so bisection bandwidth does not scale. As a consequence of these, and other, problems a number of industry leaders are convinced that optics will be required for future backplanes (e.g. Lund, 1996; Bristow, 1996).

An alternative to an electrical network is an optical interconnection network. Optical networks, such as free-space, offer a number of technical advantages over electronics including freedom from capacitive loading, higher spatial and temporal bandwidth, massive parallelism without crosstalk, lower power dissipation, and immunity to electromagnetic interference. However, for practical implementation of free-space optical interconnection technology a number of barriers must be overcome. This technology is relatively immature and improvements in technical issues, such as the alignment of optical components and the efficiency of devices need to be resolved. Also, before these networks become affordable, substantial markets need to develop. However, if one considers the total cost of an application, as opposed to hardware cost only, the costs related to the constraints imposed by an electrical network may be of considerable magnitude. This would be the case if software development is made much more difficult, since software is generally the greatest cost of application development and maintenance. If the advantages of optics, including greater bandwidth and connectivity, can provide substantial savings in software, then it may be advantageous to consider investing in free-space, or other, optical networks.

1.3 Parametric Cost Models for Software Estimation

A number of parametric models have been developed for estimating software development costs for uniprocessors. With the parametric approach, cost estimating relationships (CERs), based on design, personnel, application and project parameters, are used to predict the cost of a software project. The CERs are determined from historical software data, normally based on industry averages. Accuracy can be improved by calibrating the models to specific application domains and/or to a specific company.

The first software estimation model to be developed was the Constructive Cost Model, or COCOMO (Boehm, 1981). With this model, the effort required to produce a software application in programmer months, PM, is given by the following formula,

$$PM = c \times KLOC^k, \qquad (1.1)$$

where KLOC is the number of source lines of code in thousands, k is an exponent greater than 1.0 reflecting nonlinear growth in effort with code size and c is obtained from the product of the effort multipliers. The parameters c and k are determined by the following protocol:

1) First, the nominal effort required is estimated. This is done by determining the development mode of the project; organic, semidetatched or embedded, the latter of which is used for real time signal processing applications. This provides nominal values of k and c which are used to determine the nominal effort required as a function of the 'size' of the

software, usually given in terms of lines of code. For example, the nominal equation for the embedded development mode has c = 2.8 and k = 1.20 in Eq. (1.1) (Boehm, 1984).

2) Next, the effort multipliers are determined based on 15 cost drivers. The effort multipliers determine how much the nominal estimate must be multiplied by to account for various attributes inherent in the project and development process. Effort multipliers are classified as attributes of the:

- Product, e.g. product complexity and reliability requirements,
- Computer, e.g. memory and processor constraints,
- Personnel, e.g. experience with the application or programming language, and
- Project, e.g. use of modern programming practices and development tools.

For example, the complexity multiplier can range from 0.70 for simple applications to 1.65 for highly coupled, timing dependent applications that utilize machine code (Boehm, 1984). Similarly, reliability requirements multipliers range from 0.75 to 1.40.

3) Once the effort multipliers have been determined, they are used to scale the estimate of the development effort (c in Eq. 1.1), and related project factors such as schedule length can be estimated.

This model has been found to provide estimates within 20 % of actual values, 70 % of the time (Boehm, 1981).

Recently developed software cost estimating models are similar to COCOMO, using a standard nominal equation and adjustment factors which are determined from cost drivers. For example, the Revised Intermediate COCOMO, or REVIC, is based on the COCOMO model and was calibrated with Air Force projects to reflect actual Air Force experience (REVIC).

The Parametric Review of Information for Costing and Evaluation, or PRICE S, is a commercially available software estimating tool that automates the estimating process for software development and life cycle costs (Minkiewicz and DeMarco). It was created to help managers and estimators compare current projects to previous development efforts, translating actual software size and character into estimates of cost and schedule consistent with an organization's demonstrated capabilities. This is a commonly used tool, as evidenced by its popularity among parametric cost analysts at a meeting of the International Society of Parametric Analysts in Santa Monica in January 1997, and has been accepted for use by the Air Force, Office of the Secretary of Defense and the Defense Contract Audit Agency (Anderson, 1994). Comparisons with actual costs (e.g. Air Force Avionics Laboratory) indicated this tool to be within 15% of actual costs (Minkiewicz and DeMarco).

1.4 Impact of Processing and Memory Constraints on Software Productivity

If the specifications of an application impose tight constraints on system resources, software development costs and life cycle time will be adversely affected. Software developers need to deal directly with the operating system and hardware, and therefore, must have detailed knowledge of the machine details. High level development tools are not useful, and often coding must be done in low-level languages (Minkiewicz and DeMarco).

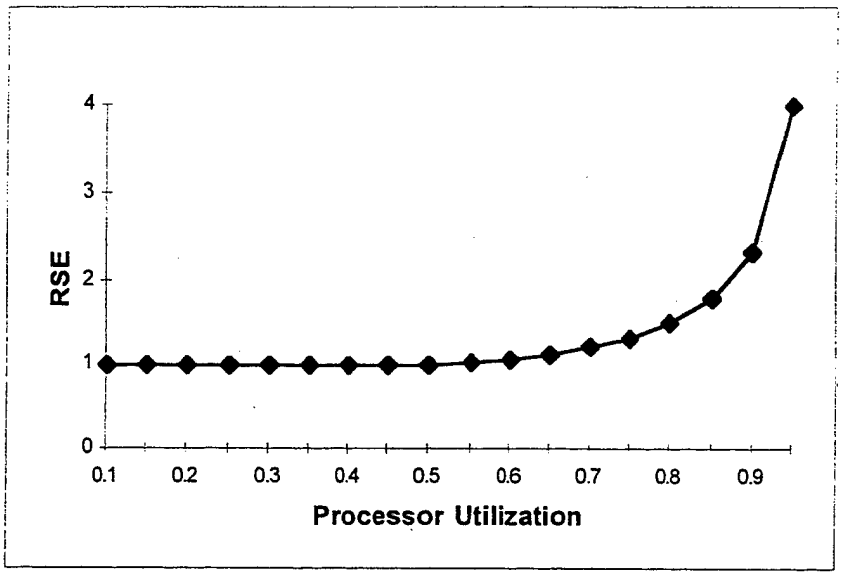


Figure 1.1 Relation between processor utilization and relative software effort, RSE, as determined from the PRICE S software estimation model (Minkiewicz and DeMarco).

The relationship between processor utilization, UTIL, and relative software effort, RSE, determined from the PRICE S model, is given by the following equation,

$$\begin{aligned} &UTIL < .5, RSE = 1.0, \\ &UTIL > .5, RSE = \frac{2}{3} + \frac{1}{6} \times \frac{1}{1 - UTIL}, \end{aligned} \tag{1.2}$$

(Minkiewicz and DeMarco) and is displayed in figure 1.1. The identical relationship is given for memory utilization and RSE. As processor utilization, defined as the fraction of

available hardware cycle time used, increases above 0.5, the required software effort increases nonlinearly. Defect rate, defined as the number of errors per thousand lines of code that are not detected until the maintenance and operations phases, is affected in a similar fashion, which will reduce the reliability of the system (Fig. 1.2).

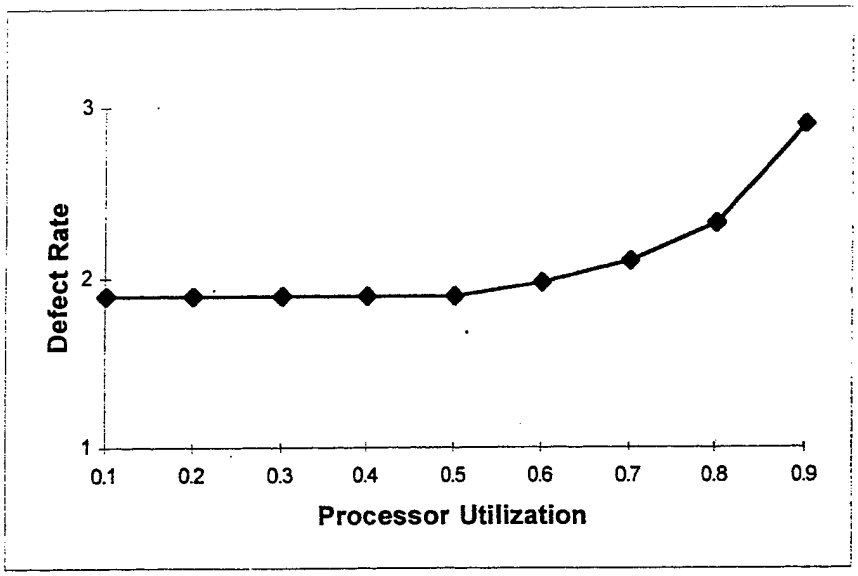


Figure 1.2 Relation between processor utilization and defect rate. Data courtesy Jim Otte, PRICE Systems.

Parametric cost estimation techniques were used to examine the effects of memory and processor constraints on the development costs of a UAV SAR processor based on the RASSP SAR benchmark developed at Lincoln Labs (Anderson, 1995). Processing (1Gflop/s) and memory requirements can just be realized with the minimal hardware of 6 Mercury MCV6 4x4m cards each with 4 40 Mhz Intel I860 processors and 16 Mbytes of DRAM. In addition, a COTS backplane-mounted processor interconnect crossbar switch, a Motorola MVME167 system controller card and a custom radar interface card is required. The resulting processor will have a memory utilization of 86% and processor utilization of 88% for a total hardware cost of $281,000 (Table 1.1). According to Anderson, these requirements are not unusual for UAV applications where size, weight and power must be minimized. However, the software costs and development time corresponding to the above memory and processor constraints are $2,360,000 and 32 months respectively, as determined by the REVIC software cost estimating model (Section 1.3). The cost was determined by estimating the code to be 8,750 uncommitted source lines of code, requiring 155 programmer-months, 152 programmer-hours /programmer-month and $100/programmer-hour. With this minimum hardware cost scenario, software development is 89% of the total development expense, $2,640,000.

	Hardware Costs	Software Costs	Total Cost	Development Time
Minimum H/W Cost	$281,000 (11)	$2,360,000 (89)	$2,640,000	32 months
Minimum Total Cost	$432,000 (32)	$911,200 (68)	$1,343,200	28 months

Table 1.1. Comparison of costs and development time for minimum hardware cost and minimum total cost scenarios. Percentages are in parentheses. Data taken from Anderson, 1995.

Minimizing the total system cost can be achieved by adding hardware to reduce the memory and processor utilization such that there is no effect on software cost. This occurs when both memory and processor utilization are below 50% (Eq. 1.1). This can be achieved by increasing the number of Mercury MCV6 4x4m cards from six to eleven with no change to the rest of the hardware. The result is an increase in hardware costs by a factor of 1.8 to $432,000 and a decrease in software costs by a factor of 2.59 to $911,200 and development time by 0.88 to 28 months. This example shows how development cost can be dominated by software and that a greater investment in hardware can reduce overall system development costs substantially. The net result is a superior product at half the cost of the minimum hardware product.

1.5 Report Overview

The above example shows how investments in processors and memory can dramatically decrease software development costs and the overall cost to develop an application. It seems likely that, in similar fashion, investments in the interconnection network can decrease the cost of developing software and hence the overall cost of an application. In this work we consider the impact of bisection bandwidth constraints on software development effort, modifications to the system to accommodate these constraints and affects on software complexity and defect rate. Our results show that bisection bandwidth constraints impact software development as much as, or more than, processor or memory constraints. Limitations in bisection bandwidth also increase software complexity and defect rates and often the capability of the system must be reduced. We discuss potential advantages of additional bandwidth, including cost savings in software development and maintenance, increased capability and the effective implementation of shared memory systems.

2 Methods

2.1 Approach

The general approach was to identify application domains in which bandwidth was constrained and determine the impact of these constraints on software development and maintenance.

To obtain an understanding of requirements, bottlenecks and development costs of real time embedded signal processing applications, experts from industrial, academic and government laboratories were contacted. An initial survey was made via mail, phone, e-mail, workshops and meetings. Visits were made to select experts to obtain information on relevant architectural and development issues. Topics of particular interest were communications and processing bottlenecks, system development processes and costs, and factors influencing software productivity such as bisection bandwidth and connectivity constraints. Considerable effort was then spent in follow up in an attempt to obtain more precise data.

2.2 Software Productivity Data

Initially, the main focus of this study was on the impact of limitations in network connectivity on software productivity and the potential of global interconnection networks in reducing software effort. It was hypothesized that DSP applications, which contain extensive global communications such as cornerturn operations, would benefit greatly from global interconnections between processors. While some DSP experts recognized that connectivity limitations increase the effort required to develop software, most identified bandwidth constraints as the major network parameter affecting programming. Also, while it seemed possible to determine a quantitative relationship between bisection bandwidth constraints and software effort, this did not seem possible for software effort and connectivity constraints. However, bisection bandwidth is related to connectivity as both depend on network topology; i.e. networks with a greater ratio of links/processor will have greater bisection bandwidth. Therefore, a relation between bisection bandwidth constraints and software effort will also capture the general affect of connectivity constraints, although we are not claiming a one to one relationship since different algorithms will be compatible to different topologies. Based on these initial results,we decided to pursue the general relationship between bisection bandwidth constraints and software productivity to capture the primary impact of the interconnection network.

During the initial survey a number of people indicated they would provide data pertaining to communication bandwidth and software productivity. However, while following up to

acquire this data, these people were often reluctant to do so. We believe this to be due to the following:

1) Software productivity data is proprietary information,
2) Data on software productivity and communication resources is rarely recorded, and
3) An individual would be required to expend considerable effort to acquire this data.

This was verified by discussions with experts in the software measurement field, including Barry Boehm of the University of Southern California, Bob Giallamboro of Mitre and Jim Otte of PRICE Systems who all claimed that software productivity data was very difficult to obtain.

In addition to contacting development experts, parametric software cost estimating models, such as COCOMO, REVIC and PRICE S were explored. While these models relate software productivity to some hardware resources (processing and memory), the impact of communication is not included explicitly. The Space and Missile Systems Center Software Database was also examined and, as with the parametric models, did not contain relevant data.

To obtain the desired information, an iterative data acquisition procedure was implemented. First, a preliminary model relating software productivity to bisection bandwidth constraints was constructed based on the results of an initial survey. This model was then sent to various experts who agreed to provide estimates, based on their experience, to test the model. Accompanying the model was a questionnaire designed to collect additional information (e.g. modifications made to algorithms, system architecture and software as a result of bandwidth limitations) and to provide a check on model estimates (Appendix A). Data was obtained by following up with phone calls when possible. This method proved to be effective in acquiring the desired data as eight people provided data for the model and eleven people answered the questionnaire. Given limited resources and time, combined with the difficulty in obtaining software data, we believe this was an effective approach to determine the general relationship between bandwidth constraints and software productivity, reflecting an industry average. Accuracy was traded for breadth in that the data includes information from many projects and from different organizations. The alternative was to collect more precise data on a very small number of programs. We believe this could have severely biased our results due to small sample size. Also, many factors contribute to software productivity, including size of project and programming language, making it difficult to compare widely differing software projects.

We report data and opinions expressed by experts in the real time embedded multiprocessor and high performance supercomputer domains, but we do not release specific information related to specific individuals or corporations in respect of the proprietary nature of this information. Data is supported with literature where possible.

3 Impact of Bandwidth Limitations on Software Productivity

A number of people have recognized that programming multiprocessors is more difficult when interconnection bandwidth is limited. In the supercomputer domain this has been expressed in the Accelerated Strategic Computing Initiative (ASCI) PathForward Project Description in which they state that interconnection performance directly impacts the ease of programming large supercomputers for high performance (ASCI, 1996). In a report prepared for the Department of Energy concerning architectural and business recommendations for achieving petaflop computing, it is stated that additional memory bandwidth and latency tolerance is required to support locality independence to make programming easier and load balancing effective (Probst, 1995). Blelloch et al. take the position that much of the difficulty in programming large parallel computers stems from a lack of appreciation of the impact of low performance interconnection on software development (Blelloch et al., 1994). They studied the NAS benchmarks, and claim that software development costs can be substantial due to multiple man-years required to obtain benchmark-specific optimizations that are often highly tuned to the particular distribution of data specified by the benchmark. This sentiment was supported by personal communications with experts in the supercommputer domain (e.g. Burton Smith of Tera Computers and Monica Lam of Stanford Computer Science Department).

The above situation was also revealed during interviews with developers of real time embedded DSP applications (e.g. Vince Zagardo and Dave Sloper of Northrop Grumman, Craig Lund and Mark Skalabrin of Mercury Computer Systems, Paul Stanton of Alacron, Bill Wren of Honeywell Technologies, Paul Storaci of Ball Aerospace, Jim Otte of PRICE Systems and many others). In a report prepared for the 'Investments in avionics and missiles software technology workshop' distributed computation with limited bandwidth was identified as one of the 'seven deadly sins' of software engineering (Shrobe, 1995).

3.1 Bisection Bandwidth and Software Productivity

To quantify the relationship between bandwidth constraints and software productivity, estimates were obtained from eight experts experienced in the development of multiprocessor applications; seven experienced in real time embedded signal processing and one in high performance supercomputing. As a measure of bandwidth constraint we use bisection bandwidth utilization (BBU), defined as the average fraction of available bisection bandwidth that is utilized during data transfers. We believe this to be a useful measure, especially for large DSP applications, since bisection bandwidth is the critical bottleneck when performing global data transfers, such as cornerturn operations, therefore reflecting the difficulty encountered by a software developer. Also, this measure is consistent with measures of hardware constraints used in existing PCE models, i.e. processor and memory utilization (Section 1.4).

As a measure of software productivity, we use relative software effort (RSE) defined as the effort required to develop software as a result of bandwidth constraints relative to the effort required if bandwidth was not constrained. A relative measure was chosen to normalize data and enable pooling from a broad range of application parameters, such as size of application and programming language, to obtain an industry average.

The relationship between BBU and RSE is given in figure 3.1 (also Appendix B) and resembles that for memory or processor utilization as given by existing PCE models (Fig. 1.1). The RSE is not affected until BBU reaches 0.3, beyond which RSE increases nonlinearly. As BBU approaches 1.0, RSE becomes extremely high, although other factors likely come into effect and the relationship is undefined. Some experts stated that as software costs begin to increase dramatically, other decisions are made. The program can be temporarily called to a halt while the organization waits until faster hardware becomes available or custom hardware may be developed. It is worth noting that three people, who did not supply detailed numbers for the model, claimed that bandwidth constraints could cause RSE to double, and two claimed it could triple.

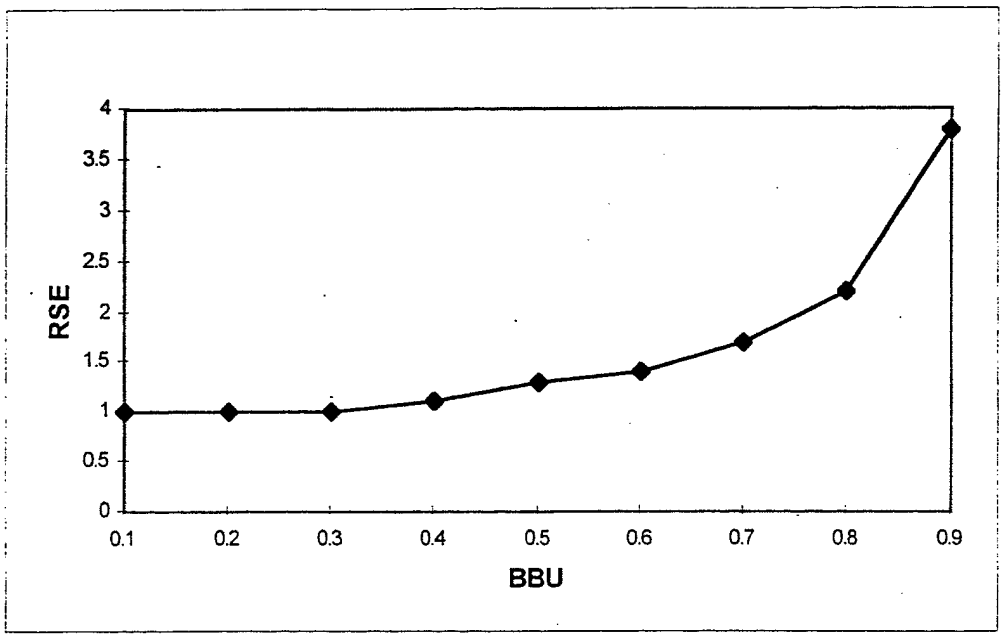


Figure 3.1 Relation between bisection bandwidth utilization (BBU) and relative software effort (RSE). See text for definitions.

Comparison to figure 1.1 indicates that BBU has a more dramatic impact on RSE than does memory or processor utilization. The adverse impact of bandwidth constraints sets in at a BBU of 0.3, compared to 0.5 for processor or memory utilization, and for BBU and processor utilization of 0.8 the RSE is 2.2 and 1.5, respectively. This was supported by the questionnaire as seven of ten claimed that bisection bandwidth constraints can affect software development as much as, or more than, memory constraints, and eight of ten claimed that software development is affected as much as, or more than, when processing is constrained.

3.2 System and Application Modifications to Reduce Communication

To overcome latency and contention problems associated with limited bandwidth, the total communication is reduced. A common strategy employed is to change the algorithm to one that is less communication intensive if the system does not possess sufficient bandwidth to run the application; i.e. replace one with global communication by one with local communication. The algorithms selected are ones where memory and computation can be added to compensate for limited communication. These changes often result in a simpler algorithm and reduced capability of the application. For example, throughput restrictions may require reducing the number of bits per pixel, thus reducing the dynamic range and limiting the capability of applications such as target recognition.

Another way to reduce communication is to reduce message size. This is done with data compression algorithms which require additional processing and memory resources. High level communication protocols, such as TCP/IP, have significant overhead, which is the time spent preparing and receiving a message, associated with them which contributes to latency. Therefore, it is common to develop low level communication protocols to reduce overhead latency. This is very difficult to do and involves coding in lower level languages.

3.3 Algorithm Mapping

Once the algorithm has been chosen it must be mapped onto the system. The mapping process involves decomposing the algorithm and distributing software tasks and data throughout the system while considering task scheduling as well. To map algorithms effectively one needs to consider memory, communication and processing as well as the operating system. This requires detailed knowledge of the system hardware, operating system and communications protocols. Unfortunately, it is impossible to predict the optimum data distribution. This procedure must be done by trial and error, and it can take considerable time to find optimal data placement.

When bandwidth is constrained, the algorithm mapping process can be much more difficult. It is common to change the way the problem is decomposed and mapped to the

architecture. One strategy is to move from data parallelism to task parallelism, which is more coarse grained and less communication intensive, and use pipelining to get concurrency. For high performance applications with size and resource restrictions, e.g. military applications, the standard approach is to make the algorithms 'fit' the available system, often resulting in reduced capabilities. A major issue to consider is contention during run time which can greatly increase the mapping effort required. Also, the task scheduling rules become more difficult when bandwidth is limited. This can require substantial effort, and some have claimed that it can comprise up to 90% of the total software effort. Applications claimed to be very difficult to map are SAR and STAP due to high communication requirements.

3.4 Bandwidth Constraints, Processor and Memory Utilization

All eleven people surveyed stated that bandwidth constraints affect processor and memory utilization. A consequence of low bandwidth is that the processors need to wait for data. A common way to solve this problem is to keep the data nearby the processor, which implies multiple copies of data and therefore additional memory. Also, reducing communication with data compression (and uncompressing) adds functionality to the system, which requires additional processing and memory resources.

3.5 Software Complexity and Defect Rate

Ten of eleven people stated that bandwidth constraints affect software complexity for the following reasons. As stated above (Section 3.3), when bandwidth is limited, one is often forced to decompose the problem by task as opposed to by data domains. Task parallelism is more asynchronous than data parallelism, which introduces a load balancing problem. This makes synchronization more difficult which also affects communication. Also, writing low level communication protocols and reducing communication is difficult (Section 3.2). The code must be made more efficient, which often requires coding in lower level languages, resulting in complex code that is hard to understand.

It is well known among software experts that error rates, in particular defect rates, are higher when software becomes more complex (e.g. Jones, 1997). This was supported in this study as six of six people claimed that bandwidth constraints affect defect rate by making code more complex. This can have a significant impact on the test and integration phase of a system (Section 3.7) as errors are more difficult to detect when code is complex, many of which go undetected until the operation and maintenance phase (Section 3.8).

3.6 Code Reuse

An obvious way to save on software development costs is to reuse code from previous applications. When asked what proportion of the code is reused when porting software to new platforms, the responses obtained were very broad depending on application domain, requirements and platform similarity. However, it was commonly expressed that a major problem in real time high performance embedded systems development is the difficulty in porting software to new target platforms. In fact, in many cases porting basically becomes a reimplementation as code that has been highly optimized for a particular machine and distribution of data cannot be reused on a different platform. Some claimed that as little as 5% of the code is reused but the most common response was between 10 to 20 %, consisting largely of utilities software which is easiest to code. Five of seven people claimed that additional bandwidth could make it easier to reuse code.

3.7 System Integration and Test

Although bandwidth constraints can affect all phases of the development cycle, including system specification, hardware/software partitioning and design, and coding, the phase most severely impacted is the integration and test phase. It is during this phase of the development cycle that the limitations are found. This can be a very long and costly phase, comprising over 50% of the total development time for a typical project and can consume 80% for human critical systems (DeMarco, 1994; Bunza, 1996).

Debugging multiprocessing systems is far more difficult than uniprocessors as one must find both logical errors as well as errors due to interactions between parts of the code. Standard real-time debugging tools run on a uniprocessor environment and don't reveal time-related interactions between code sections (Quinnell, 1996). When the code is more complex due to bandwidth constraints, it is even more difficult to find and fix errors (Section 3.5). Also, if the bandwidth limitation is not known until the hardware/software integration phase, it may be necessary to repeat the system specification, hardware/software partitioning, software design and algorithm mapping phases, which can increase the development effort substantially.

3.8 Software Maintenance

Software maintenance, which includes fixing defects and upgrading functionality, is generally the most costly phase of the life cycle (Shrobe, 1995). For example, the cost to develop 236,000 lines of code for an F16 Fighter system was $85 million, and $250 million to maintain the code (Bunza, 1996). When bandwidth is constrained the maintenance phase can be impacted in that upgrades are more difficult, and the number of

possible udgrades is limited. It is very difficult to upgrade functionality on complex systems. When part of the system is changed, the effect on the rest of the system must be determined. This requires considerable testing to ensure that the system is fully operable. For high reliability systems 75% of the time in an upgrade cycle is spent in testing and analyses (Shrobe, 1995). Eight of ten people stated that bandwidth constraints make the maintenance phase more difficult. The added complexity due to bandwidth constraints makes it more difficult to test the system (Section 3.7).

Seven of eleven people stated that adding new functionality to a previously developed application will utilize more bandwidth. The other four said it can, depending on whether the tasks are scheduled concurrently or not. If an upgrade requires bisection bandwidth utilization to increase, then software development and testing become increasingly difficult (Fig. 3.1). Also, it may be necessary to reallocate bandwidth on the original application to accommodate the new functionality which will require additional software design and development. If bandwidth is constrained in the original application and upgrades utilize additional bisection bandwidth, software development will become increasingly more difficult until upgrades are no longer possible.

4 Impact of Connectivity Constraints on Software Productivity

Additional connectivity was not perceived as being as important as additional bandwidth in making programming easier. This was indicated during both the initial survey and the questionnaire results. When asked if additional connectivity would make software development easier, the responses were as follows: yes (2), no (2), sometimes (2), probably (1), could (1), not necessarily (1), somewhat (1), don't know (1). While these results indicate that limitations in connectivity can impact software development effort, there was uncertainty in the responses and in how additional connectivity would help. The advantages given were that the additional connectivity would provide more data paths, reducing contention and latency, and task placement would be easier. Also, during general discussions with DSP development experts, some claimed that additional connectivity would make programming easier and some expressed a desire for all-to-all connectivity such as a fully connected, low latency crossbar. A network of this nature would not only minimize software development costs by providing substantial bisection bandwidth (Section 2.2), the low latency would provide performance benefits.

5 Advantages of Additional Communication Resources

5.1 Savings in Software Development and Maintenance

A common theme with embedded systems experts is that increasing bandwidth on a constrained system would make programming easier. How much easier it would be to program would depend on how close the application is to utilizing the hardware resources and how much that constraint would be reduced by adding communication resources. Some claimed that greater connectivity would also help by providing more data paths, reducing contention and latency, but a quantitative relationship could not be determined.

As an estimate of potential savings in software development costs, we use the RASP SAR benchmark example of Anderson in which the cost to develop software with no hardware constraints is $911,200 (Anderson, 1995; Table 1.1). Consider a system in which the communication requirements and hardware resources are such that developers are forced to develop software while utilizing 80% of the available bisection bandwidth. With this level of bisection bandwidth utilization the software development cost, determined from figure 3.1, will be $2,004,640, a 2.2 fold increase. This does not seem unrealistic based on discussions with experts who revealed that bisection bandwidth utilization can become higher than 0.8 and bandwidth constraints can cause software costs to double or triple (Section 3.1). If an interconnection network is implemented such that bisection bandwidth utilization is 30%, software development costs are reduced to the baseline value of $911,200, a savings of $1,093,440. Besides saving in software costs, the resulting system will have better performance and simpler more reliable code. Also, it will be easier to debug and upgrade the system during the maintenance phase, and with greater bandwidth available more upgrades can be accommodated.

5.2 Shared Memory Architectures Enabled

It is widely recognized that shared memory systems are easier to program than message passing systems. Message passing codes are difficult to parallelize, tightly coupled to existing data structures, require extensive 'tuning' to achieve optimal performance through locality and are more difficult to modify than shared memory codes (Probst, 1996). A shared memory machine, with uniform memory access, does not require the programmer to be concerned about data locality to achieve optimal performance. However, to effectively implement locality independence and make dynamic load balancing effective, a high bandwidth interconnection network is required (Probst, 1996). It has been estimated that moving from message passing to shared memory will reduce software development and maintenance costs by approximately 50%. These savings may be adequate to justify larger investments in interconnection bandwidth.

5.3 New Algorithms Implemented

If sufficient bandwidth is not available, communication intensive algorithms must be reduced or discarded. This can lead to reduced capability of the algorithms that are implemented (Section 3.2). Some of the algorithms that were claimed to be limited by bandwidth include:

1) STAP problems with high numbers of channels,
2) High resolution SAR,
3) ATR with greater than 10 models,
4) Some Kalman filters with high degrees of adding,
5) Applications with TeraFLOP requirements,
6) Closing track loops in remote sensing.

An obvious advantage of increasing bandwidth is an increase in capability as communication intensive algorithms are enabled. This could greatly increase radar, remote sensing and target detection capabilities.

6 Conclusions

The results of this study show that bisection bandwidth constraints can lead to a number of adverse consequences in the resulting application. Algorithms are modified to reduce communication, requiring processing and memory requirements to increase. This often leads to a reduced capability of the application. Accommodating limited bandwidth results in complex software that is difficult to understand and debug, and often requires coding in lower level languages. As a result of the added complexity, integration and testing, which comprise over 50% of the development time, is more difficult. The net result is that bandwidth constraints have a substantial impact on software costs. It is possible that software development cost can double or triple, consuming millions of extra dollars if bandwidth is sufficiently constrained. Also, the additional software costs accumulated during the maintenance phase, which comprise the largest portion of the total life cycle cost, can be substantial as upgrades consume additional bandwidth resources.

We expect bandwidth constraints will have a greater overall impact on software development costs with time. As military applications demand greater functionality and throughput, developers will be forced to utilize more of the available bandwidth while developing more complex software. This will likely be enhanced by the increasing disparity between advances in processing speeds and interconnection bandwidth of electrical networks; i.e. processing speeds are increasing faster than electrical network speeds (e.g. Stone, 1996).

Although connectivity was not perceived to be as important as bandwidth, we believe there are advantages to implementing a global all-to-all topology, in agreement with some of the experts interviewed. The extremely high bisection bandwidth of a network of this nature will alleviate communication constraints, thus minimizing software development costs (Section 3.1). Also, latency will be minimized which will contribute to greater performance and make it easier to implement algorithms with high global communication requirements.

Given the substantial impact of bandwidth constraints on software development and maintenance costs, there may be strategic advantages in investing in high bandwidth interconnection networks, or in developing new interconnection technologies. One such technology is free-space optics which promises to be compact and low power, which are important requirements for airborne applications. Also, free-space optics may enable a global all-to-all topology. It is impossible to predict the magnitude of investment required to move this technology into the commercial world, or the business advantages in doing so. This will depend on many factors including market sizes. However, besides providing direct savings in software costs, additional advantages can be realized by increasing bandwidth. This includes greater capability in DoD radar and imaging applications, by

enabling communication intensive algorithms, and effective implementation of shared memory systems. The greater capability may have direct impact on mission effectiveness (i.e. reduced loss of aircraft and life) and therefore national security. Effective implementation of shared memory systems, with uniform memory access, should make it easier to reuse code since it will not be locality dependent. The benefits of increased capability and shared memory implementation may, in fact, provide the biggest payoffs in the long term.

Discussions with developers of real time multiprocessors have revealed that it would be desirable to be able to perform trade analyses between major hardware components (processing, memory and interconnection network) and software productivity to optimize system performance and cost. To do so would require a software cost estimation model for real time multiprocessors that explicitly incorporates the impact of these major hardware components on software productivity. Existing models, developed for uniprocessors, consider processing and memory constraints but do not consider parameters unique to multiprocessors such as bisection bandwidth constraints or number of processors. We have found that bisection bandwidth constraints impact software development as much as, or more than, processing and memory constraints. For this reason, the importance of bisection bandwidth should not be overlooked when estimating software costs for real time embedded multiprocessors. Other factors to consider are 1) the impact of the number of processors on software effort, and 2) inter-dependencies between major hardware components since accommodating for bandwidth constraints requires increasing processor and memory requirements (Section 3.4). A tool of this nature should be extremely useful for the rapid design and prototyping of cost effective real time multiprocessing systems. It would enable tradeoff analyses to be made in the early stages of the development cycle, e.g. conceptualization, and support decisions on high level issues such as technology choices. It would also be desirable to include the maintenance phase in a real time embedded software estimating model. This would allow one to optimize for upgrades and enable trade analyses based on the entire life cycle of the system. We believe a modeling tool of this nature would produce substantial savings in costs over the life cycle of an application.

7 List of References

Anderson, J. C., 1995. "Projecting RASSP Benefits," *Proceedings of the 2nd Annual RASSP conference*, ARPA, Arlington, Virginia, pp. 65-72.

Accelerated Strategic Computing Initiative (ASCI) PathForward Project Description, December 27, 1996, http://www.llnl.gov/asci-pathforward.

Bartow, J., 1995, "Evolutionary Design of Complex Systems," *Investments in avionics and missiles software technology workshop report, ARPA/SISTO*, Software Productivity Consortium Inc. Report SPC-95068-CMC.

Blelloch, G. E., B. M. Maggs and G. L. Mile, 1994, "The Hidden Cost of Low Bandwidth Communication," *Developing a computer science agenda for high performance computing*, ACM Press, pp. 22-25.

Bristow, J., 1996, "Electrical and Optical Interconnect Issues at the Backplane /Daughtercard Interface," *1996 IEEE Workshop on interconnections within highspeed digital systems*, May 19-22, 1996, Santa Fe, NM.

Bunza, G. J., 1996, "A Journey Into Parallel Worlds: Exploring Hardware/Software Systems Integration," *Embedded Systems Conference*, San Jose, September, 1996, Miller Freeman.

DeMarco, T., 1994, "Software Integration and Test," PRICE Systems, Mt. Laurel, NJ.

Douglas, B. P., 1996 "Software Estimation and Scheduling," *Embedded Systems Conference*, San Jose, September, 1996, Miller Freeman.

Jones, C., 1997, *Applied Software Measurement: Assuring Productivity and Quality*, 2nd Ed., McGraw-Hill, New York.

Lund, C., 1996, "The Evolution of a PCI Fabric," *1996 IEEE Workshop on interconnections within highspeed digital systems*, May 19-22, 1996, Santa Fe, NM.

Minkiewicz, A. and DeMarco, T., "The PRICE Software Model," PRICE Systems.

Probst, D. K., 1995, "Architectural Visions versus Business Models: How Soon Will There Be Enabling Technologies for Petaflops Computing?" *Report prepared for Gil Weiland at DoE/DP-07 [HQ]* December 1995.

Quinnel, R. A., 1996, "Operating Systems and Development Tools Lighten the Load," *EDN*, July, 1996.

REVIC Users' Group c/o Management Consulting and Research, Inc., Oxnard, CA.

Shrobe, H., 1995, "Evolutionary Design of Complex Software," *Investments in avionics and missiles software technology workshop report, ARPA/SISTO*, Software Productivity Consortium Inc. Report SPC-95068-CMC.

Stone, H., 1996, "Keynote Presentation," *1996 IEEE Workshop on interconnections within highspeed digital systems*, May 19-22, 1996, Santa Fe, NM.

8 Appendices

Appendix A - Data Questionnaire

Below is the questionnaire as sent to development experts. It includes a preliminary software cost model, for experts to test with their estimates, and a questionnaire to obtain additional information on the consequences of bandwidth constraints and to verify estimates given for the model.

SOFTWARE/BANDWIDTH COST MODEL - BACKGROUND

Discussions with numerous engineers indicates that when communications bandwidth is constrained, software development can be severely impacted, especially in real time embedded DSP applications. As a result: 1) additional planning during the initial design stage is required, 2) coding must be done at lower levels and 3) the testing and integration phase is severely complicated.

A model relating relative software effort and development time to bandwidth constraint (bisection bandwidth utilization) is given below. Please provide estimates to test this model and answer as many questions as possible. Provide ranges if necessary. ALL INFORMATION YOU PROVIDE WILL BE KEPT CONFIDENTIAL. AVERAGES AND RANGES ONLY WILL BE RELEASED. If you would like a copy of the report when completed, provide a mailing address.

SOFTWARE COST MODEL

Bisection B/w Util	Relative S/W Effort (Model)	Relative S/W Effort (Estimate)	Relative Dev. Time (Model)	Relative Dev. Time (Estimate)
.1	1.0		1.0	
.2	1.0		1.0	
.3	1.0		1.0	
.4	1.0		1.0	
.5	1.0		1.0	
.6	1.1		1.0	
.7	1.2		1.1	
.8	1.5		1.1	
.85	1.7		1.2	
.9	2.2		1.4	
.95	3.8		1.7	

NOTE: As a measure of bandwidth constraint we use bisection bandwidth utilization, defined as the average fraction of available bisection bandwidth used during data transfers. We believe this to be a useful measure for large DSP applications since bisection bandwidth is the critical bottleneck when performing global data transfers such as cornerturn operations, therefore reflecting the difficulty encountered by a software developer. Other possible measures are 1) the ratio of computation to communication and 2) gap, defined as "the average time needed between consecutive message transmissions by any one processor in order to ensure that the network does not become overloaded".

To what domain(s) does the above data apply? E.G.: Real time embedded DSP - Military, Airborne, Space; Real time Telecommunications; MIS; Supercomputer; Shared memory, message passing.

Does the number of processors influence software productivity?

QUESTIONS

1. On average, what proportion of the development cost and effort is due to software?

2. Estimate an average cost for a relative software effort of 1.0 in dollars and lines of code, if possible.

3. When it is known that bandwidth will be limited, what changes are made to the system? E.G.: interfaces, drivers, processors, communication protocols, control, synchronization.

4. For communication intensive applications, what is it about Hardware that makes it difficult to develop software?

5. When bandwidth is limited, what changes are made to the software?

6. Do bandwidth constraints affect software complexity? Does it affect defect rate?

7. When bandwidth is constrained, does it affect software development as much as, or more than, when memory or processing is constrained?

8. Do bandwidth constraints affect CPU and memory utilization? I. E. Does one have to battle the network to get CPU and memory utilization up?

9. When bandwidth is limited, what phases in the system development cycle are impacted?

a) System specification
b) Hardware/Software partition
c) Software design
d) Algorithm mapping
e) Coding
f) Software test and integration
g) System test and integration
h) Maintenance

10. Does adding modes to a previously developed application utilize more bandwidth?

11. Would greater network connectivity help? If so, how?

12. What applications are communication intensive with stringent latency constraints?

13. What applications and/or algorithms cannot be done due to bandwidth constraints?

14. What proportion of code is reused when porting software to new platforms? Can this also be a problem when porting from the development system to the target?

15. If the bandwidth constraints could be removed, would that:

a) make coding easier,
b) simplify software development (e.g. distribution of tasks),
c) simplify the overall system (e.g. control, synchronization, protocols),
d) enable the implementation of new algorithms,
e) make it easier to reuse software
f) enable new capabilities/applications

Appendix B - Software Effort Data

Table B.1 Summary of data relating bisection bandwidth utilization (BBU) to relative software effort (RSE) for sample size (N). Note that as BBU becomes greater than 0.95 the relationship is undefined.

BBU	RSE	N
0.1	1	8
0.2	1	8
0.3	1	8
0.4	1.1	8
0.5	1.3	8
0.6	1.4	4
0.7	1.7	6
0.8	2.2	6
0.9	3.8	7
0.85	2.4	4
0.95	4.4	4

Appendix C - List of Credits

Hundreds of people were contacted during the course of this study. The following is a list of people who provided assistance throughout this project. Those who were particularly helpful are underlined.

Alacron
Sgro, Joe
Stanton, Paul

Army Research Lab
Welby, Steve

Ball Aerospace Systems
Storaci, Paul

Carnegie-Mellon University
Blelloch, Guy

Computing Devices International
Christofferson, Brett
Lienberger, Bill

Cygnus
Savoye, Robert

DARPA
Husain, Anis
Salasin, John
Shrobe, Howie
Yoon, Barbara

Eagle Design Automation
Bunza, Geoff

GDE Systems
Grucza, Jack

Georgia Tech.
Madisetti, Vijay

Harvard University
Valiant, Leslie

Honeywell Technologies
Bristow, Julian
Samson, John
Spaanenburg, Henk
Symosek, Peter
Wren, Bill

JPL
Seigel, Herb

L. A. Air Force Base
Tinkler, Shirley

Lincoln Labs
Anderson, Jim
Martinez, David
Shaw, Gary

Lockheed Martin
Kline, Bill
Pridmore, Jeff
Saultz, James

Los Alamos National Lab
McGhee, John

MCI
Reifer, Don

MCR Federal Inc.
Sherry Stukes

Mercury Computer Systems
Lund, Craig
Skalabrin, Mark
Vichniac, Gerard

MICOM
Sims, Richard

Mitre Corporation
Games, Richard
Giallamboro, Bob

NASA
Dean, Ed

Northrop Grumman
Campbell, Mark
Hand, Bruce
Harwick, Tom
Sloper, Dave
Oechsler, Tim
Zagardo, Vince

NRAD
Cottel, Dennis
Partow, Perry

Omeda Medical
Douglas, Bruce

PRICE Systems
DeMarco, Anthony
Otte, Jim
Slocum, George
Tahir, Nina

Rome Labs
Repak, Paul

SAIC
Guarino, Dave

Sanders
Potter, Stewart
Graybill, Bill

Sandia National Lab
Carlson, Richard
Hale, Art
Stalker, Terry
Williams, Robert
Yee, Mark

Stanford University
Lam, Monica
Goodman, Joseph

Sun Microsystems
Papadopoulos, Greg

TechWorks Service Group
Sanford, Walt

Tera Computer Company
Smith, Burton

University of Colorado
Jordan, Harry
Johnson, Kirk

University of Southern California
Boehm, Barry
Prasanna, Victor

Vexcel Corporation
Curlander, Jim

WindRiver Systems
Klein, Ed

DISTRIBUTION LIST

addresses	number of copies
JAMES L. DAVIS ROME LABORATORY/OCPC 25 ELECTRONIC PKY ROME NY 13441-4515	3
UNIVERSITY OF COLORADO AT BOULDER OFFICE OF CONTRACTS AND GRANTS 206 ARMORY, CAMPUS BOX 19 BOULDER CO 80309-0019	3
ROME LABORATORY/SUL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 8725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	2
ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
ROME LABORATORY/ERO ATTN: RICHARD PAYNE HANSCOM AFB, MA 01731-5000	1
ROME LABORATORY/EROC ATTN: JOSEPH P. LORENZO, JR. HANSCOM AFB, MA 01731-5000	1
ROME LABORATORY/EROP ATTN: JOSEPH L. HORNER HANSCOM AFB, MA 01731-5000	1

ROME LABORATORY/EROC 1
ATTN: RICHARD A. SOREF
HANSCOM AFB, MA 01731-5000

ROME LABORATORY/ERXE 1
ATTN: JOHN J. LARKIN
HANSCOM AFB, MA 01731-5000

ROME LABORATORY/ERDR 1
ATTN: DANIEL J. BURNS
525 BROOKS RD
ROME NY 13441-4505

ROME LABORATORY/IRAP 1
ATTN: ALBERT A. JAMBERDINO
32 HANGAR RD
ROME NY 13441-4114

ROME LABORATORY/C3BC 1
ATTN: ROBERT L. KAMINSKI
525 BROOKS RD
ROME NY 13441-4505

ROME LABORATORY/OCP 1
ATTN: MAJOR GARY D. BARMORE
25 ELECTRONIC PKY
ROME NY 13441-4515

ROME LABORATORY/OCP 1
ATTN: JOANNE L. ROSSI
25 ELECTRONIC PKY
ROME NY 13441-4515

NY PHOTONIC DEVELOPMENT CORP 1
MVCC ROME CAMPUS
UPPER FLOYD AVE
ROME, NY 13440