intellitech

The Intelligent Use of Technology

Computer-Aided Engineering Tools for Spacecraft Multi-Microprocessor Design

EXECUTIVE SUMMARY

Computer-Aided Engineering Tools for Spacecraft Multi-Microprocessor Design

EXECUTIVE SUMMARY

May 1982

Authors: Dr. S.A. Mahmoud
Dr. C. Laferriere
Mr. J.G. Ouimet
Mr. W.T. Brown

Approved by: Dr. S.A. Mahmoud

Government of Canada | Gouvernement du Canada

Department of Communications

DOC CONTRACTOR REPORT DOC-CR-SP -82-043

DEPARTMENT OF COMMUNICATIONS - OTTAWA - CANADA

SPACE PROGRAM

TITLE: Computer-Aided Engineering Tools For Spacecraft Multi-Microprocessor Design - Executive Summary

AUTHOR(S): S.A. Mahmoud
C. Laferriere
J. Ouimet
W. Brown

ISSUED BY CONTRACTOR AS REPORT NO: None

PREPARED BY: Intellitech Canada Ltd.
352 MacLaren St.
Ottawa, Ontario
K2P 0M6

DEPARTMENT OF SUPPLY AND SERVICES CONTRACT NO: 3ER.36100-1-0273
SN: OER81-03151

DOC SCIENTIFIC AUTHORITY: R.A. Millar

CLASSIFICATION: Unclassified

DATE: May 1982

Preface

This work was performed for the Department of Communications, Communications Research Centre under DSS Contract No. OER81-03151, entitled "Computer-Aided Engineering Tools for Spacecraft Multi-Microprocessor Design", from September 15, 1981 to March 31, 1982. This report is one of the following four contract deliverables:

1. Executive Summary

2. Report #1 - Review of Multiprocessor Systems and their Spacecraft Applications.

3. Report #2 - A Survey of Computer-aided Engineering (CAE) Tools for the Design and Simulation of Multiprocessor Systems.

4. Report #3 - The Definition and Specification of an Integrated Set of CAE Tools for Spacecraft Multiprocessor System Design.

Acknowledgement

The study team gratefully acknowledges the technical guidance of Mr. R.A. Millar of the Communications Research Centre. His knowledge and experience in the field of computer simulation of spacecraft systems have contributed to the quality of the work and provided a constant source of encouragement to the study team.

As well, the study team wishes to thank Mr. J.M. Savoie, also from CRC, for his fruitful discussions and critical reviews.

EXECUTIVE SUMMARY

1.0 Introduction

Interest in multiprocessor and distributed intelligence computer systems have increased substantially in recent years. This interest has been fostered by the availability of microprocessors with ever increasing performance-price ratios and the expected emergence of monolithic systems with still higher capabilities in the near future.

Advances in LSI and VLSI semi-conductor technology have significantly reduced computer hardware weight, power consumption and cost. It is now feasible and practical to employ multiprocessor systems on spacecraft in order to increase the reliability, extend mission duration and satisfy increasingly more computational demand during the mission.

The development of multiprocessor and distributed intelligence computer systems and their utilization in various applications have been impeded by the lack of an appropriate theoretical base. The control of systems containing large number of processors is not well understood. While considerable work has been done recently to develop a theoretical base, it seems unlikely that this work will have significant impact on practical system design in the near future. The difficulties encountered in developing a theoretical base are attributed to the large number of interrelated design variables and decisions, many of which depend on rapidly advancing hardware and software technologies.

Practical design methodologies of multiprocessors can be facilitated and enhanced by the availability of computer-assisted engineering (CAE) tools. Such CAE tools support the skill level of the designer, provide insight into the attributes of alternative architectures, allow evaluation of the performance of these architectures and support the development, simulation and testing of actual multiprocessor systems.

More specifically, computer-aided engineering tools are required to simulate alternate hardware configurations, evaluate the software implications of selecting a particular hardware configuration, perform required hardware-software tradeoffs, establish that the specified hardware and software are compatible and that overall system performance requirements are met. All of these must be done at an early stage in the design process, before the software is coded and the hardware is constructed.

In the absence of such computer-aided engineering tools, it is difficult for the designer to assess and evaluate system performance adequately before constructing a breadboard prototype, developing its software, and testing the resulting system. At this late stage in the design process, discovered inadequacies and inconsistencies are expensive and time-consuming to correct and often require significant redesign. With the appropriate CAE tools, the chances of this happening at such a late stage in the design process are minimized.

The use of these CAE tools in the area of software design and development for actual multiprocessors provides significant

advantages. In current practice, the task of translating the functional requirements of the system into software modules (written in a given high level language) is usually left up to the individual designer. The task is often performed based on ad hoc (informal) techniques which depend on the designer's skill and background. In the absence of formal tools and techniques that can ensure the correctness of the software and its compliance with requirements, several errors that accumulate throughout the design and development stages are discovered only at later stages. Correcting these errors often involves massive changes in the software design, thus prolonging the project's development cycle and increasing its associated cost. The availability of tools for software verification and validation, and the utilization of these tools at high levels of the design will enforce good software programming practice and uniform documentation procedures. It will thus minimize the probability of major errors appearing at later stages of the design and development process.

2.0 Contract Objectives

The objective of this contract is twofold:

(1) to examine, through a detailed survey, existing CAE tools for multiprocessor systems design and,

(2) to investigate ways of enhancing and augmenting these tools to form an integrated set which can be used at all design levels.

3.0 Reports Delivered

Results of the study are documented in three reports which accompany this Executive Summary.

The first report, entitled "Review of Multiprocessor Systems and their Spacecraft Applications", examines the basic technological issues involved in the design of software and hardware architectures of multiprocesors and their applicability in general to meet the requirements of spacecraft on-board processing. The advantages of using multiprocessor systems for spacecraft applications have been identified in terms of reliability enhancement, flexibility in meeting increasing computational demand and extending mission duration.

The second report, entitled "A Summary of Computer-Aided Engineering Tools for the Design and Simulation of Multiprocessor Systems" presents a survey of existing CAE design tools for multiprocessors, covering all design layers including the requirement specifications phase. Surveyed tools were broadly classified into four categories according to the main function of the tool and the design level at which the tool is primarily used. The advantages and disadvantages of available tools in each category were identified according to a specified set of evaluation criteria.

The third report, entitled "The Definition and Specification of an Integrated Set of CAE Tools for Spacecraft Multiprocessor System Design" presents, in detail, our proposal to enhance and augment existing tools in two aspects. The first involves the development of a high level functional component description tool,

based on the ADA programming language, to bridge the gap between the high level requirement specification phase and the relatively low design level at which the system architecture is selected and simulated. The second aspect involves the development of performance analysis tools which can be used in evaluating system reliability and resource utilization. The performance analysis tools will augment existing special design languages which can be used to simulate the system architecture and the software/hardware structure.

4.0 Technical Summary

Unique definitions for the terms "multiprocessor" and "multiprocessor networks" are noticeably lacking in current related literature. To avoid any ambiguity that may arise as a result of this, we define, for the purpose of this study, the multiprocessor system to be a multiplicity of microprocessors that are physically and logically interconnected to form a single system in which overall executive control is exercised through the cooperation of decentralized system elements.

Moreover, the scope of multiprocessor systems examined in this study can be defined in terms of a set of characteristics considered to be pertinent in spacecraft applications:

- The microprocessors forming the system, as well as all other system elements, co-exist in the same locality (i.e., no telecommunications lines are used since the elements are not geographically separated),
- The microprocessors and other system elements are interconnected according to one of alternative structures (uni or multi-bus, a loop or ring connection, a matrix switch, etc.),
- Conceptually, a single executive manages all of the system's physical and logical resources in an integrated fashion. The control logic and data structures are replicated among a number of processors or memories,

- The number of processors to be interconnected is relatively small,
- Redundancy in the hardware is assumed through the use of identical spares, which along with other fault recovery mechanisms constitute what is known as "fault-tolerant" architectures.

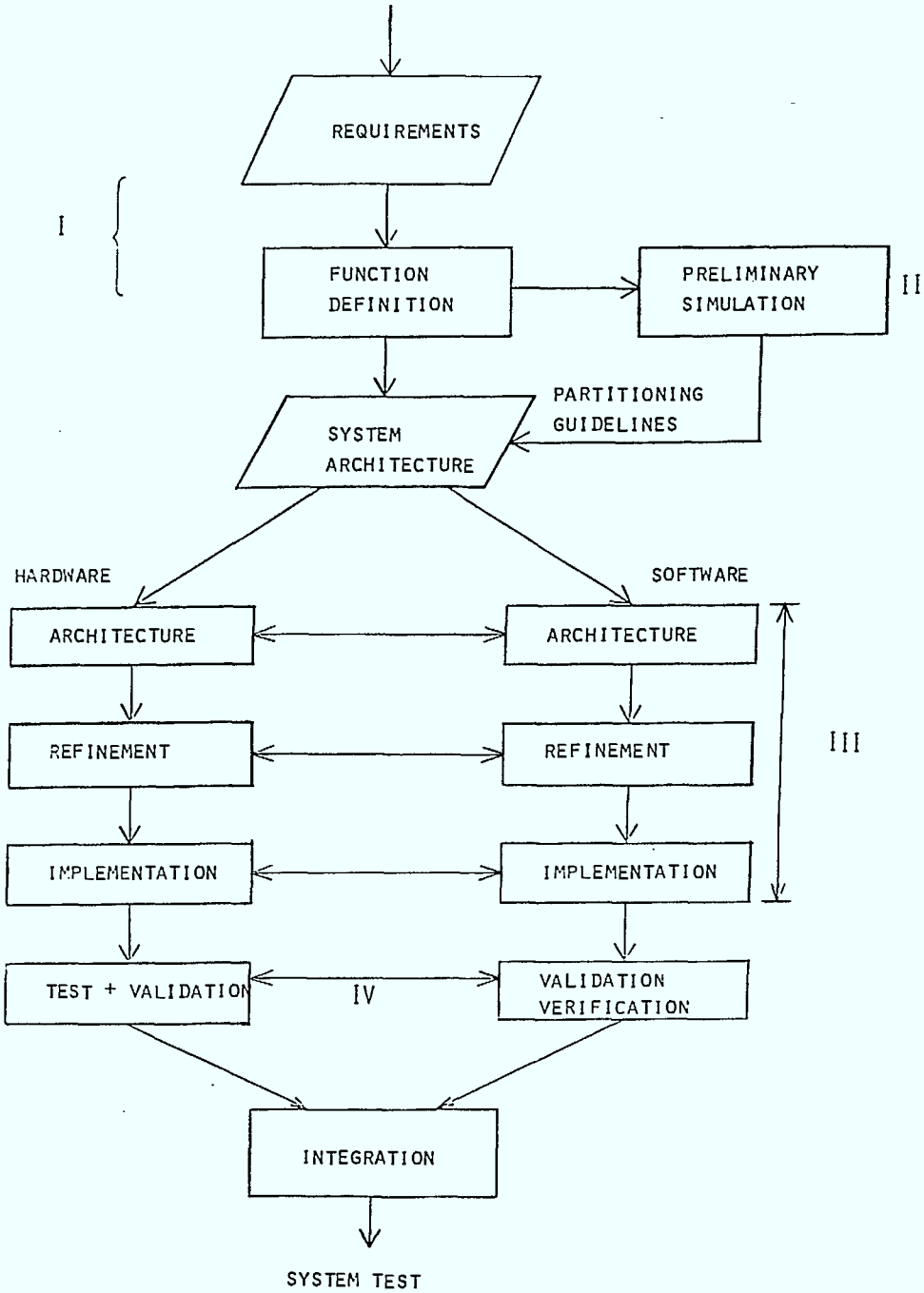
To understand the role, scope and utility of multiprocessor design tools in various design phases, it is essential to describe the various design steps followed in a general top-down development process of a multiprocessor system. Figure 1 illustrates the design steps followed from the early requirements definition step until a complete system prototype is assembled and tested.

The first step involves the specification of system requirements and is followed by a description of the functional components of the system which are considered necessary to satisfy the requirements. The high level description of the functional components is then translated into an intermediate design stage which involves the selection of a system architecture and its representation in the form of a system model. Preliminary simulation is usually conducted to examine alternative architectures and to provide partitioning guidelines which define the boundaries of the hardware and the software components of the system. Two design paths are then followed concurrently.

The first path involves determining the architecture of the software which is then defined in a series of top-down design steps. The software is then coded and verified using simulators to

Figure 1

DESIGN STEPS OF MULTIMICROPROCESSORS



determine its correctness and its compatibility with the hardware. The second path involves determining the architecture of the hardware which is then refined in a series of top-down design steps. The hardware architecture is then simulated to determine its compatibility with the software. Following simulation, the hardware is implemented in a prototype which can be tested and validated. The hardware and software subsystems are then integrated and the resultant system is tested.

Evaluation of existing tools can be conducted based on the utility and usefulness of the tool with respect to each design step shown in Figure 1. The following basic criteria are used to characterize and evaluate the tools covered by the survey.

I. Ability to Specify Functions:

Function definition follows the requirements specification phase and is conducted at the early stage of the design prior to the selection of the system architecture.

II. Simulation Capabilities:

Simulation is conducted at two different stages: (1) early (preliminary) simulation, conducted to select a subset of feasible architectures which will be examined in detail later, and (2) complete simulation, conducted to bind the software and the hardware structure descriptions and test their compatibility following the refinement of the description of each structure.

III. Support of Top-down Design Methodology:

Because of the complexity of the system architecture, it is always convenient to conduct the design refinement process by detailing each component separately while maintaining the consistency of the design by specifying the interconnection between the components at each refinement level. In the software area, the support of a top-down development approach means that units or components can be specified, compiled and tested separately.

IV. Verification Capabitilies:

Verification capabilities exist in the form of certain mechanisms (constructs) in the programming language (tool) which can be used to verify the correctness of the executable code. It is the responsibility of the user to define, in a mathematical form, all the conditions that correspond to correct execution. These conditions can be used either as the basis for a complete symbolic simulation as performed by verification systems or as run-time checks.

V. Support and Compatibility:

The degree of support a tool is given by the computer industry and major government users (e.g. Department of National Defence) is important since it determines the level of attention, commitment and future development effort to enhance the language and to support its evolution.

It is possible to classify existing tools into four broad categories based on their essential characteristics and the main design stage at which the tool is most useful. These categories are:

(1) Specification Languages

Examples of such languages include:

- SADT/SAINT (System Analysis and Design Techniques, developed by SofTech and the SAINT simulator developed by the U.S. Air Force)
- REVS (the Requirements Engineering and Validation System, developed by TRW Defense and Space Systems)
- RPS (the Requirements Processing System, developed by GTE Laboratories)

These languages are used primarily as tools at the requirement and function defenition phases. Their basic characteristics with respect to the evaluation criteria described earlier are summarized in Table 1.

(2) Special Verification Languages

Examples of such languages include:

- GYPSY
- AFFIRM
- EUCLID
- SPECIAL (HDM)

The primary purpose of these languages is to develop

verifiable software code. Special verification mechanisms are embedded in the language and can be used to verify the executable code. The basic characteristics of these languages, with respect to the evaluation criteria described earlier, are summarized in Table 2.

(3) General Procedural Languages

The most promising among these languages is the recently introduced programming language ADA*. The primary objective of this language is to allow the development of software code in modular fashion. The modules can be specified, written and compiled separately. A complex system can be formed as a combination of these modules. The modular feature of this language, coupled with its flexible concurrency mechanisms, makes it an attractive development tool for multiprocessors. In addition, the language is strongly supported by the computer industry and the U.S.A. Department of Defense. The basic characteristics of such a development tool, with respect to the evaluation criteria described earlier, are summarized in Table 3.

(4) Special Design Languages

Examples of these languages include:

- AIDE (ArchItecture Design Environment, Bell Laboratories

* ADA is a trademark of the U.S. Department of Defense.

- N.mPc. (implemented at Case Western Reserve University)
- SABLE (Stanford University)

Such languages are generally used as tools at the architecture design level and below. They are particularly useful in simulating the hardware/software details and interactions. Design consistency and completeness can thus be checked out and analyzed prior to the implementation of a hardware prototype. The basic characteristics of such special design languages, with respect to the evaluation criteria described earlier, are summarized in Table 4.

Table 1

S P E C I F I C A T I O N L A N G U A G E S

Criteria	Comments
I. Ability to Specify Functions	- Very powerful requirements definition
II. Simulation Capabilities	- Limited simulation (except for SADT/SAINT)
III. Support of Top-down Methodology	- Has no translation (interface) to lower levels
IV. Verification Capabilities	- No verification
V. Support and Compability	- Strongly supported by industry

Table 2

SPECIAL VERIFICATION LANGUAGES

Criteria	Comments
I. Ability to Specify Functions	- Not requirements oriented - Support mathematical system specification
II. Simulation Capabilities	- Limited simulation (at architecture level and above) - Restrictive concurrency
III. Support of Top-down Methodology	- Translation to lower levels possible - Separate compilation units not supported
IV. Verification Capabilities	- Verification possible - Considerable user expertise required
V. Support and Compability	- Current use is limited to academic institutions & research

Table 3

GENERAL PROCEDURAL LANGUAGES

(Ex. ADA)

Criteria	Comments
I. Ability to Specify Functions	- Not requirements oriented - Useful in function definition - Support flexible concurrency
II. Simulation Capabilities	- Simulation possible at architecture level and above - Can be used to provide hardware/software partitioning guidelines
III. Support of Top-down Methodology	- Separate compilation facilities
IV. Verification Capabilities	- Limited verification (ex. type checking, range checking, procedure call checks, ...)
V. Support and Compability	- Considerable support from U.S. DOD and computer industry

Table 4

S P E C I A L D E S I G N L A N G U A G E S

Criteria	Comments
I. Ability to Specify Functions	- Not used above architecture level - Description at architecture level and below
II. Simulation Capabilities	- Simulation possible at architecture level and below - Simulation allows consistent description of hardware (structure) and software (behaviour) - Limited performance evaluation possible
III. Support of Top-down Methodology	- System can be decomposed into components
IV. Verification Capabilities	- No verification capability
V. Support and Compability	- SABLE/ADLIB and N.mPc. are available commercially - use so far has been limited to research institutions

5.0 Conclusions

Two main conclusions are drawn from the study conducted here:

1. Existing special design languages, such as SABLE/ADLIB and N.mPc., are extremely useful in simulating system architecture and in binding the software and hardware detailed structures, and in testing the compatibility of these structures. The utility of these languages can be enhanced by the addition of special performance analysis routines in the resource allocation and utilization area and in the reliability analysis area.

2. A high level procedural language is needed with the following characteristics:

- Support function definition and description of system architecture.
- Allow preliminary simulation in order to select feasible architectures and specify each architecture in terms of its software and hardware components.
- Be translatable into lower level target machine code.
- Has verification capability.
- Has considerable support from government and industry.
- Support top-down program development.

The programming language ADA appears to satisfy most of the above requirements. However, the main drawback of the language is in its complexity, since the existence of many flexible constructs makes it difficult to verify those programs that attempt to utilize the full power of the language. By imposing certain constraints on the use of some language mechanisms, it is hoped that a compromise

is reached such that certain critical program sections can be verified. The verification issue in ADA will likely be resolved to some extent as a result of the massive research and development effort currently underway in various research institutions.

6.0 Follow-up Work

6.1 Proposed Short Term Work (1982-1983)

The following is a summary of the work proposed for the short term (1982-1983):

- Acquire special design languages (N.mPc., SABLE/ADLIB).
- Install on computer facility at Communications Research Centre.
- Examine top-down design capability through specific design examples.
- Assess limitations (performance evaluation).
- Determine required interfaces to high level procedural language.

6.2 Proposed Intermediate Term Work (1983-1984)

The following is a summary of the work proposed for the intermediate term (1983-1984):

- Use of ADA as a tool at the function definition level and for architecture description and simulation.
- Definition of appropriate constructs to describe data flow and model system behaviour (concurrency, synchronization, multitasking, etc.)
- Introduce certain restrictions to allow verification of critical program sections.
- Define and implement interfaces to special design languages at the architecture level and below.
- Examine utility through specific design examples.

intellitech
Intellitech Canada Ltd.
352 MacLaren Street,
Ottawa, Ontario.
K2P 0M6
(613)235-5126