

AD-A281 574



AD-A281 574
Final Technical Report
200 1534

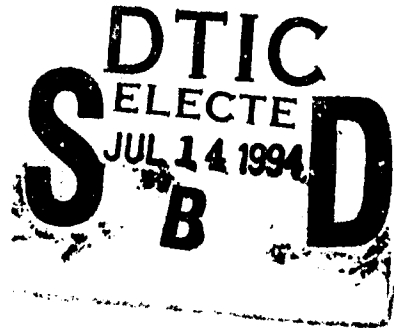


(1)

AN ARCHITECTURE FOR INTELLIGENT MULTICHIP MODULE RELIABILITY ANALYSIS

University of Massachusetts, Amherst

Daniel D. Corkill



APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

94-21577



4606

DTIC QUALITY INSPECTED 8

**Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York**

94 7 12 2 54

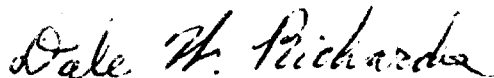
This report has been reviewed by the Rome Laboratory Public Affairs Office (PAO) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

Although this report references the following limited document, no limited information has been extracted.

"Thermal Stress Analyses of Integrated Circuits Using Finite Element Methods," Apr 84, distribution limited to USGO agencies and their contractors.

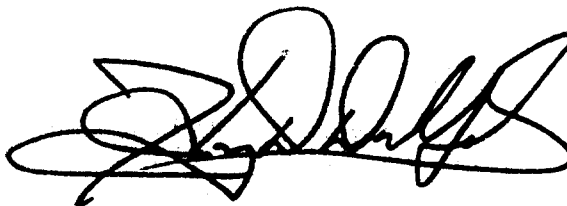
RL-TR-94-71 has been reviewed and is approved for publication.

APPROVED:



DALE W. RICHARDS
Project Engineer

FOR THE COMMANDER



HARVEY D. DAHLJELM, Colonel
Director, Reliability
Electromagnetics & Reliability Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (ERSR) Griffiss AFB NY 13441. 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.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE April 1994		3. REPORT TYPE AND DATES COVERED Final Mar 92 - Aug 93	
4. TITLE AND SUBTITLE AN ARCHITECTURE FOR INTELLIGENT MULTICHIP MODULE RELIABILITY ANALYSIS				5. FUNDING NUMBERS C - F30602-92-C-0028 PE - 62702F PR - 2338 TA - 02 WU - 5X	
6. AUTHOR(S) Daniel D. Corkill					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Massachusetts, Amherst Box 36010, OGCA, Munson Hall Amherst MA 01003-6010				8. PERFORMING ORGANIZATION REPORT NUMBER FTR-528226	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory (ERSR) 525 Brooks Rd Griffiss AFB NY 13441-4505				10. SPONSORING/MONITORING AGENCY REPORT NUMBER RL-TR-94-71	
11. SUPPLEMENTARY NOTES Rome Laboratory Project Engineer: Dale W. Richards/ERSR/(315) 330-3476					
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This report describes the development of a computer-based modeling and analysis system called the "Intelligent Multichip Module (MCM) Analyzer" (IMCMA). IMCMA is a blackboard-based software tool that automatically performs finite-element and knowledge-based analysis of MCM designs to rapidly assess their reliability. In IMCMA, modeling effort and expert-operator requirements have been reduced through: (1) use of high-level representation of devices as the interface between the designer and the analysis tools; (2) capturing the expertise of experienced design analysts in an intelligent assistant for use by less experienced designers. Nine knowledge sources (KSs) were completed that take a high-level device specification through model generation and simplification, finite-element generation, and thermal analysis. These KSs include stand-alone FORTRAN finite-element generation codes that have been integrated into IMCMA by using the generic blackboard framework, GBB. This initial IMCMA prototype quickly produces a thermal analysis when given a high-level MCM design description.					
14. SUBJECT TERMS Multichip Module, MCM, Blackboard, Artificial Intelligence, Reliability, Thermal Analysis, Design, Finite Element Analysis, FFA, Modeling				15. NUMBER OF PAGES 48	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED		18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED		19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	
20. LIMITATION OF ABSTRACT UL					

Executive Summary

This report describes the results of our one-year effort in developing an intelligent, automated analysis system that can be used to provide rapid reliability assessments of multichip microelectronic module designs. This effort is part of a cooperative effort between the Design Analysis Branch at Rome Laboratory and the Mechanical Engineering and Computer Science Departments at the University of Massachusetts in developing a powerful computer-based modeling and analysis system called the *Intelligent Multichip Module Analyzer* (IMCMA). The IMCMA system is a blackboard-based software tool that automatically applies finite-element and knowledge-based analysis to rapidly assess the reliability of microelectronic multichip module (MCM) designs. The software bases its evaluation on environmentally and operationally induced failure mechanisms. It is useful in assessing the reliability of advanced microelectronic devices that have no historical reliability data.

Extensive understanding of the analysis and the failure prediction of advanced electronics is being incorporated into IMCMA. Rome Laboratory is the Air Force's center of expertise for the reliability assessment of advanced electronics and is experienced in the design, construction, and analysis of MCMs, especially in the areas of MCM failure mechanisms and reliability assessment. The Mechanical Engineering Department at UMass provides significant expertise in finite-element modeling and analysis and the Computer Science Department provides expertise in knowledge-based problem solving, problem-solving representations, and control.

Early detection of design-related reliability problems can significantly decrease the development, manufacture, and testing costs of MCMs. This potential savings is offset by the labor-intensive nature of modeling as a means of detecting design-related reliability problems. Modeling an MCM requires significant expertise to build and, if necessary, remodel critical regions of an initial finite-element model. Typically, it may take a senior design analyst several weeks to construct and analyze an initial computer-based model of a microelectronic device. An expert design analyst must decide how to model the individual components, how to represent them so they can be used by analysis tools (such as an automated mesh generator), and how to interpret the results. The numerical results from the initial model will often indicate critical regions of the device that must be remodeled in order to obtain an accurate analysis of the mechanical stresses of these regions.

Current analysis tools lack high-level models of microelectronic devices and loading conditions. Based on low-level geometric representations, these tools require a skilled expert to translate the high-level model of the device into representations that the analysis tools can utilize. Conversion between different representations used in each tool is also often left to the designer.

In the IMCMA system, modeling effort and expert-operator requirements have been reduced through two main techniques:

- Use of a high-level representation of devices as the interface between the designer and the analysis tools
- Capturing the expertise of experienced design analysts in an intelligent assistant to be made available to less experienced designers

Increasing the productivity of design experts significantly reduces the development effort, lead time, and cost associated with new MCMs.

To date, nine knowledge sources (KSs) have been completed that take a high-level device specification through model simplification, finite-element generation, and thermal analysis. These KSs include the use of stand-alone FORTRAN finite-element generation codes that have been integrated into IMCMA by using the generic blackboard framework, GBB. This initial portion of the IMCMA system can produce a thermal analysis of a high-level MCM design description in a few minutes.

In this report, we describe our findings and progress.

Accession For		<input checked="" type="checkbox"/>	<input type="checkbox"/>
NTIS GRA&I		<input type="checkbox"/>	<input type="checkbox"/>
DTIC TAB			
Unannounced			
Justification			
By			
Distribution/			
Availability Codes			
Dist	Avail and/or		
A-1	Special		

Contents

1 Introduction	1
2 The IMCMA Architecture	2
The IMCMA Blackboards	3
IMCMA Knowledge Sources	5
3 Knowledge Source Descriptions	5
The Input-Model KS	5
The Adjust-Model KS	9
The Complete-Model KS	10
The Find-Symmetry KS	12
The Generate-Mapmesh-Regions KS	12
The Generate-2D-Mesh KS	13
The Extrude-Component KS	16
The Combine-3D-Meshes KS	17
The Analyze-3D-Mesh KS	18
4 Lessons Learned	21
Interfacing Issues	21
Avoiding Private Information	22
The Importance of Rapid Prototyping	23
Problems Using the Sandia Tools	23
5 Summary of Accomplishments	24
6 References	25
A Device Definition Files	27
B Running IMCMA	31
C IMCMA Trace Output	33
D Glossary	38

Acknowledgments

The IMCMA project is a cooperative effort involving a number of individuals from Rome Laboratory and the Mechanical Engineering and Computer Science Departments at the University of Massachusetts at Amherst. Without the friendly and open interchange of ideas and expertise among everyone involved with the IMCMA effort, the research described in this report would not have been accomplished.

From the Rome Laboratory Design Analysis Branch, Dale Richards, Doug Holzhauer, Mark Stoklosa, Peter Rocci, and Paul Yaworsky provided expertise in microelectronic devices and finite-element-based reliability assessment.

From the UMass Mechanical Engineering Department, Professor Ian Grosse and his students, Prasanna Katragadda and Anagha Jog, developed the FEECAP finite-element analysis code integrated into IMCMA as part of this effort, performed experiments on the appropriateness of paving versus map-mesh finite-element generation, determined how to best obtain insulating chip sides using the Sandia tools, and provided considerable insight into the effective use of the Sandia software meshing tools. The Mechanical Engineering Department also provided the Research Associate for this project, Venkat Manakkal, who developed the Common Lisp interfaces to the Sandia Exodus binary files.

Sandia National Laboratory made available the FORTRAN-based finite-element meshing tools (FASTQ, GJOIN, and GEN3D) that were used as knowledge sources in the IMCMA prototype. Ray Meyers and Greg Sjaardema of Sandia assisted with the software licensing and provided technical support during this effort.

Finally, the emphasis of this one-year effort was on integrating a number of diverse tools and expertise into the prototype IMCMA system. Our work was made substantially easier by the project organization and management provided by Doug Holzhauer and Dale Richards. The success of this effort is directly tied to their success in providing a project environment where individuals separated by both background and physical distance could collaborate effectively.

1 Introduction

Early detection of design-related reliability problems can significantly decrease the development, manufacture, and testing costs of multichip microelectronic modules (MCMs). For example, the Design Analysis Branch (ERSD) of the Electromagnetic Reliability Directorate at Rome Laboratory has successfully used the finite-element method for reliability assessment of MCM components to predict failure modes and assess their mechanical reliability [1,2]. Detailed simulation studies of microelectronic devices has been used to successfully predict the location and magnitude of critical thermal and physical stresses [3,4,5,6,7,8]. These studies can be used to predict the reliability and failure modes of the device.

Detecting design-related reliability problems using detailed simulation studies is labor intensive, and requires significant expertise to build and, if necessary, remodel critical regions of an initial finite-element model of MCM devices on the computer. Typically, it may take an engineer several weeks to construct and analyze an initial model of a microelectronic device on the computer. The numerical results from the initial model will often indicate critical regions of the device which must be remodeled in order to obtain an accurate model of these regions. For example, remeshing is needed to resolve meshing problems due to violation of geometric transitioning constraints or due to basic limitations of the mesh generator. Increasing the productivity of design experts will significantly reduce the development effort, lead time, and cost associated with new MCMs.

Today, an expert designer must decide how to model the individual components, how to represent them so they can be used by analysis tools (such as an automated mesh generator), how to interpret the results and potential failure of these tools, and how to reformulate the model for further analysis, if needed. Current analysis tools lack high-level models of microelectronic devices and loading conditions. Based on low-level geometric representations, these tools require a skilled expert to translate the high-level model of the device into representations that the analysis tools can utilize. Similarly, the expert must relate the detailed analysis results back to the high-level model, understanding the implications of the detailed results to overall device reliability. Conversion between the many different representations used in the various analysis tools is also often left to the designer.

Modeling effort and expert operator requirements can be reduced through several techniques:

- Use of a high-level representation of devices as the interface between the designer and the analysis tools. The designer would define the device in terms of its components and the analysis system would use these high-level definitions to develop a detailed representation of the device. For example, a MCM might consist of a number of uniform rectangular chips and capacitors mounted on the substrate in a symmetrical pattern. The designer would specify the chips, capacitors, and

An Architecture for Intelligent Multichip Module Reliability Analysis

the pattern, and the system would develop all the detailed submodels of critical features of the device.

- By capturing the expertise of experienced designers in an intelligent assistant for MCM device analysis, much of the labor-intensive aspects of reliability analysis can be automated and made available to less experienced designers [9]. Instead of the expert designer deciding how to use the tools to effectively model devices, the analysis system would develop and implement a modeling strategy for analyzing the device. The system would monitor the accuracy of the modeling process, detecting modeling problems such as idealizations resulting in singularities in the finite-element solution, violations of mesh transitioning constraints, and poorly structured meshes. Once detected, the system would reformulate the model or remesh until an acceptable model is developed.

These techniques form the basis of the IMCMA system.

2 The IMCMA Architecture

A blackboard system, based on the blackboard problem-solving paradigm [10,11], was used as the basis for the IMCMA architecture. A blackboard system performs problem solving by using three basic components (Figure 1):

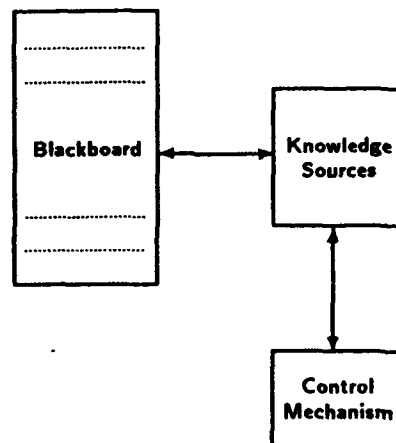


Figure 1: Basic Components of the Blackboard Model

- A blackboard, that is a global database containing input data, partial solutions, and other data that are in various problem-solving states.
- Knowledge sources (KSs) which are independent modules that contain the knowledge needed to solve the problem, and that can be widely diverse in representation and in inference techniques. KS modularity facilitates application development and simplifies maintenance and enhancement.

An Architecture for Intelligent Multichip Module Reliability Analysis

- A control mechanism, that is separate from the individual KSs and that makes dynamic decisions about which KS is to be executed next.

The power of the blackboard approach lies in its ability to:

- Organize problem-solving knowledge into multiple independent knowledge modules
- Allow the knowledge in each module to be represented differently
- Allow different problem-solving techniques in each knowledge module
- Flexibly apply knowledge modules in the most appropriate way to efficiently solve the problem

The blackboard model offers a powerful problem-solving architecture that is suitable when:

- Many diverse, specialized knowledge representations are needed. KSs can be developed in the most appropriate representation for the data they are to handle.
- An integration framework is needed that allows for heterogeneous problem-solving representations and expertise.
- The application uses large-grained modularity for design, implementation, and maintenance.
- The application involves many developers.
- Uncertain knowledge or limited data inhibits absolute determination of a solution.
- Multilevel reasoning or flexible, dynamic control of problem-solving activities is required in an application.

The IMCMA prototype was implemented using Blackboard Technology Group, Inc.'s GBB™ framework, a toolkit for rapidly developing and delivering high-performance blackboard applications. As will be discussed, GBB allowed us to quickly change problem representations and approaches during prototype development. This proved crucial to the success of this one-year prototype effort.

The IMCMA Blackboards

The blackboards in the IMCMA prototype are organized into the hierarchy shown in Table 1. There are three groupings of blackboards in IMCMA: MODEL, LIBRARY, and CONTROL-SHELLS.

The MODEL blackboards contain the information used in the current IMCMA prototype. These blackboards are subdivided into 2D-MODEL, 3D-MODEL, and MATERIAL blackboards containing blackboards used for 2D modeling, 3D modeling, and material

GBB is a trademark of Blackboard Technology Group, Inc.

An Architecture for Intelligent Multichip Module Reliability Analysis

MODEL

2D-MODEL

COMPONENT-2D-SURFACES	11 Units:	(11 COMPONENT-2D-SURFACE)
COMPONENT-2D-LINES	44 Units:	(44 COMPONENT-2D-LINE)
COMPONENT-2D-POINTS	43 Units:	(43 COMPONENT-2D-POINT)
MAPMESH-2D-REGIONS	272 Units:	(272 MAPMESH-2D-REGION)
MAPMESH-2D-LINES	577 Units:	(577 MAPMESH-2D-LINE)
MAPMESH-2D-POINTS	306 Units:	(306 MAPMESH-2D-POINT)
2D-ELEMENTS	1088 Units:	(1088 2D-ELEMENT)
2D-NODES	1155 Units:	(1155 2D-NODE)

3D-MODEL

COMPONENTS	22 Units:	(10 PRESCRIBED-FLUX-SURFACE, 1 PRESCRIBED-TEMPERATURE-SURFACE, 11 COMPONENT)
COMPONENT-3D-LINES	132 Units:	(132 COMPONENT-3D-LINE)
COMPONENT-3D-POINTS	88 Units:	(88 COMPONENT-3D-POINT)
COMPONENT-3D-SURFACES	66 Units:	(66 COMPONENT-3D-SURFACE)
3D-ELEMENTS	1352 Units:	(1352 3D-ELEMENT)
3D-NODES	2704 Units:	(2704 3D-NODE)
MATERIALS	2 Units:	(2 MATERIAL)

LIBRARY [0]

COMPONENT-LIBRARY	Empty
COMPONENT-INSTANCE-LIBRARY	Empty
MATERIAL-LIBRARY	Empty

CONTROL-SHELL [0]

KSS	9 Units:	(9 KS)
KSAS	19 Units:	(19 KSA)

Table 1: IMCMA Blackboard (analyzing test device)

descriptions, respectively. The various component, mapmesh, and finite-element blackboards and objects contained in these blackboards will be discussed as part of the KS descriptions presented later in this report.

The LIBRARY blackboards are for a planned standard component and material library facility that has not been implemented as part of this effort.

The CONTROL-SHELLS blackboards are part of the KS control facilities supplied by GBB.

The particular blackboard and object hierarchy used in the IMCMA system was designed for conceptual and representational convenience. The GBB framework used to build IMCMA allows developers to select from a wide range of blackboard and object structures (such as a single blackboard for all objects, a separate blackboard for each class of object, and so on). These decisions do not effect performance, and we chose the IMCMA hierarchy based on our sense of naturalness.

Table 1 also indicates the names and counts of the blackboard objects created by a high-level analysis of a fictitious MCM device. Although based on real technology,

An Architecture for Intelligent Multichip Module Reliability Analysis

this "Test MCM" device does not physically exist and was created solely for purposes of demonstration, debugging, and illustration. The Test MCM device will be used throughout this report.

IMCMA Knowledge Sources

Nine KSs and KS interfaces were developed for the IMCMA system under this contract (Table 2).

<code>input-model-ks</code>	GBB	Reads a device specification file and builds the component and material objects specified therein
<code>adjust-model-ks</code>	GBB	Simplifies the geometry of the device to facilitate rapid analysis
<code>find-symmetry-ks</code>	GBB/CLIPS	Identifies 2D (XY) geometric symmetries in the device
<code>complete-model-ks</code>	GBB	Builds the component objects that are based on the adjusted model
<code>generate-mapmesh-regions-ks</code>	GBB	Generates the coarsest possible 2D mesh for the adjusted device
<code>generate-2d-mesh-ks</code>	GBB/FORTRAN	Generates a 2D mesh from the mapmesh regions and mesh density specifications
<code>extrude-component-ks</code>	GBB/FORTRAN	Edits the 2D mesh for a specific component and extrudes that component into a 3D mesh
<code>combine-3d-meshes-ks</code>	GBB/FORTRAN	Combines all the component 3D meshes into a single 3D mesh
<code>analyze-3d-mesh-ks</code>	GBB/FORTRAN	Analyzes the combined 3D mesh

Table 2: IMCMA Knowledge Sources (developed or interfaced under this contract)

Processing proceeds among these KSs as shown in Figure 2. These KSs will be described in detail in the following sections.

3 Knowledge Source Descriptions

The Input-Model KS

The `input-model-ks` KS is triggered when the IMCMA system begins its operation. For example, once the IMCMA system has been loaded, the form:

```
(run-imcma "test-example" :xy-adjust 1.0 :z-adjust .2)
```

An Architecture for Intelligent Multichip Module Reliability Analysis

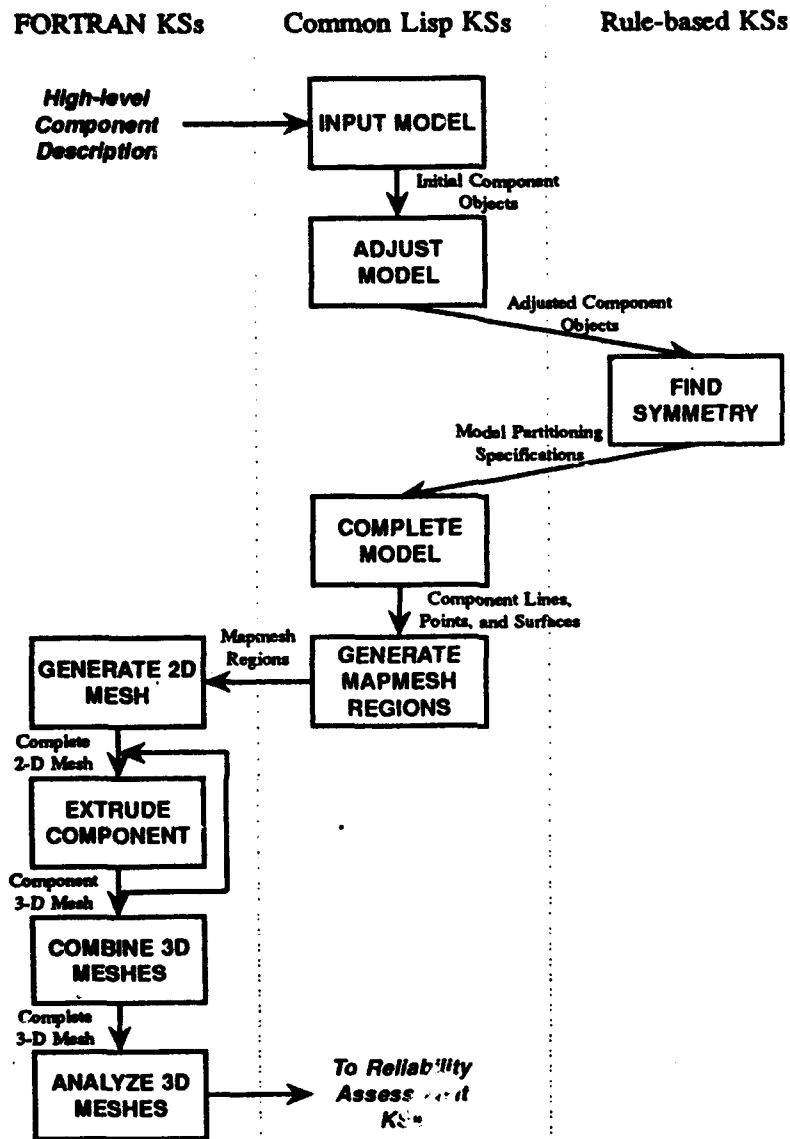


Figure 2: IMCMA KS Processing

will invoke IMCMA on the device specified in the device specification file named `test-example.lisp`. The device specification file contains a high-level geometric description of the device and its components, descriptions of the materials used in the device, and thermal and static loading information. (An example device specification file is shown in Section A of the Appendices.) These specifications are supplied using the following forms:

An Architecture for Intelligent Multichip Module Reliability Analysis

define-device *name* &*key* :*size* :*title* :*filename* [Macro]

This macro should appear once at the start of the device specification file. It specifies that the device being defined is named *name*. The *:size* argument specifies the overall size (bounding rectangular prism) of the device. The *:title* argument specifies a title to be included in all ASCII files generated by IMCMA (the default title is the value of *name*). The *filename* argument specifies a root filename to be used by IMCMA in generating all intermediate file names (the default value is also *name*).

Here is an example of **define-device**:

```
(define-device "Test MCM"  
:filename "test"  
:size (40.64 40.64 1.27))
```

defmaterial *name* &*key* :*e* :*nu* :*tk* :*max-error-tolerance* :*alpha* [Macro]
: *beta* :*reference-temperature*

This macro is used to define each material used in the device. These macros should appear after the **define-device** macro, but before any components are defined. The *:e*, *:nu*, *:tk*, *:alpha*, *:beta*, and *:reference-temperature* arguments specify material properties. The *:min-error-tolerance* and *:max-error-tolerance* arguments specify FEECAP analysis requirement values (see KS analyze-3d-mesh-ks).

Here is an example of **defmaterial**:

```
(defmaterial :alumina  
:e .54e8  
:nu .22  
:min-error-tolerance .1  
:max-error-tolerance .1  
:tk (50 55 60)  
:alpha (.2 .3 .4)  
:beta 90  
:reference-temperature 30)
```

defcomponent *name type* &*key* :*x* :*y* :*z* :*size* :*components* [Macro]
: *material* :*operating-range* :*xy-alignment*
: *point-static-loads* :*point-heat-sources*
: *prescribed-flux-surfaces*
: *prescribed-convection-surfaces*
: *prescribed-temperature-surfaces*
: *power-dissipation-surfaces* :*power-dissipation*
: *failure-mode*

This macro is used to define each component used in the device. These macros should appear after the **define-device** and **defmaterial** macros. The *:x*, *:y*, and *:z* arguments specify the component's position relative to the device coordinates (the

An Architecture for Intelligent Multichip Module Reliability Analysis

argument `:xy-alignment` specifies whether the `:x` and `:y`, values correspond to the lower-left of the component (the default) or the center of the component (specified by `:centered`). The `:size` argument specifies the length, width and height of the component. The `:components` argument specifies subcomponents. The `:material` argument specifies the component material. The arguments:

- `:operating-range`
- `:xy-alignment`
- `:point-static-loads`
- `:point-heat-sources`
- `:prescribed-flux-surfaces`
- `:prescribed-convection-surfaces`
- `:prescribed-temperature-surfaces`
- `:power-dissipation-surfaces`
- `:power-dissipation`

specify operating characteristics of the component. The `:failure-mode` argument is not used in the current prototype implementation.

Here are two examples of `defcomponent`:

```
(defcomponent :SUBSTRATE-1 :substrate
: size (40.64 40.64 1.27)
: prescribed-temperature-surfaces ((:bottom 30))
: material :alumina)
```

```
(defcomponent :CHIP-1 :chip
: size (13.0010 5.9890 .516)
: x (+ (/ 40.64 2.0) 10.1451)
: y (+ (/ 40.62 2.0) -8.4118)
: z (+ 1.27 -.516)
: xy-alignment :centered
: power-dissipation .1
: material :silicon)
```

The overall design for IMCMA includes a component and material library facility for maintaining a library of often used components and materials. This library is not operational in the prototype IMCMA system and all material and component information must be supplied in the device specification file.

The Lisp-based input-model-ks KS performs the following activities:

- Reads the supplied device specification file and creates the defined material and component objects on the `model` blackboard.

The component surface, line, and point objects and the prescribed convection, temperature, power-dissipation, and flux surface objects are not created until the

An Architecture for Intelligent Multichip Module Reliability Analysis

components are adjusted for modeling (by the adjust-model-ks KS described below).

- If the IMCMA graphics module has been loaded, the graphics are activated and the graphic display is initialized to show the plan view (xy) and side view (yz) of the device (Figure 3).

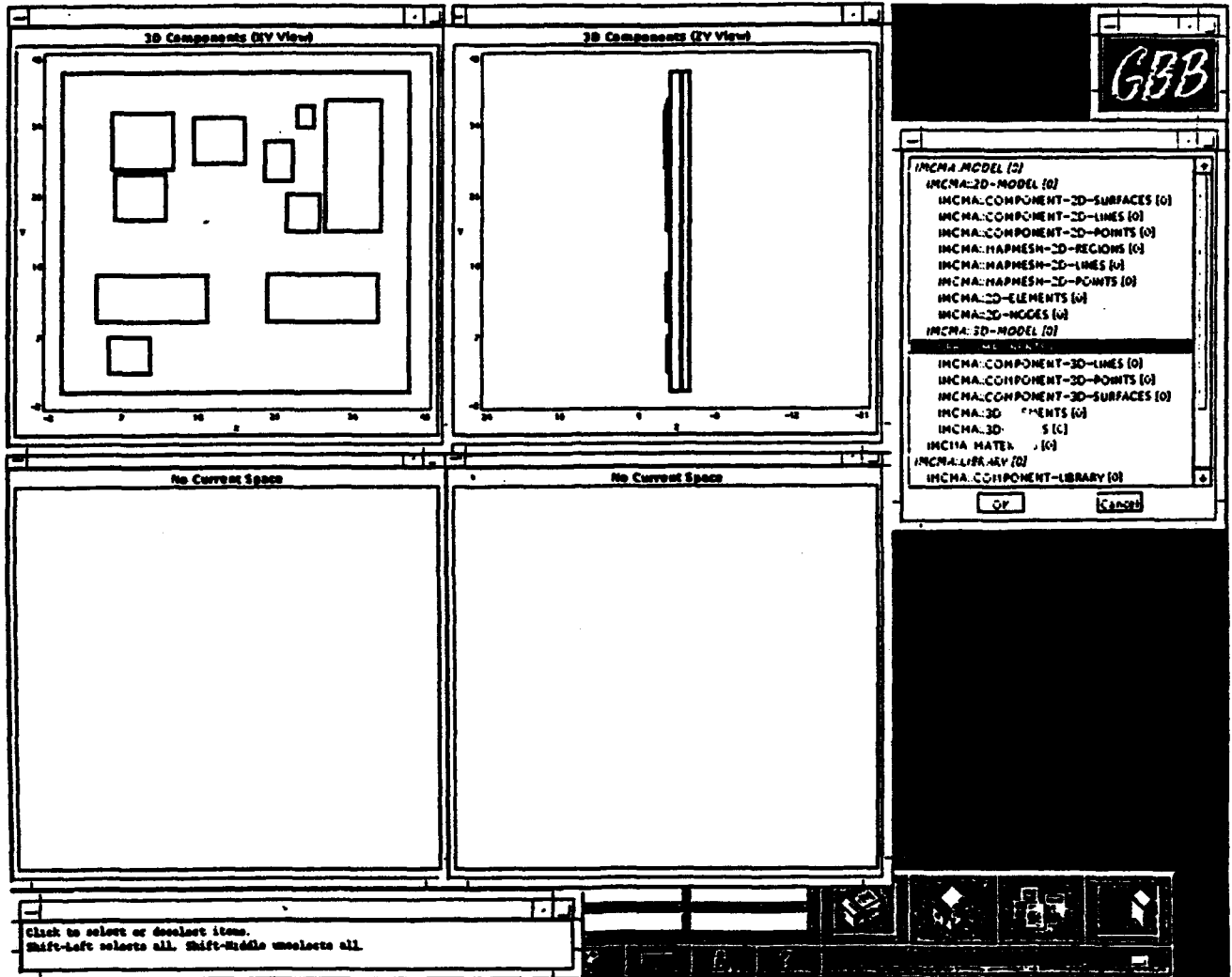


Figure 3: GBB Graphics Showing Test MCM Plan View

The Adjust-Model KS

This KS simplifies the geometry of the device to facilitate rapid analysis by adjusting the modeled position and size of components to reduce the number of finite elements necessary to model the device. Given xy-adjustment and z-adjustment parameters, the modeled physical positions of components are shifted to improve component alignment in the xy and z dimensions. This Lisp-based KS performs the following activities:

An Architecture for Intelligent Multichip Module Reliability Analysis

1. Checks the specified `:xy-adjust-distance` and `:z-adjust-distance` parameters to insure that they are not greater than half the minimum component xy-distance and z-height, respectively. Adjustment parameters greater than these maximums can cause components to "disappear" from the simplified model. If either of these parameters are too large, the user is presented with a continuable error.
2. Adjusts the xy coordinates of the components within the `:xy-adjust-distance` bounds. The x and y dimensions are adjusted separately. For each dimension, the lowest coordinate (the device exterior) remains unchanged, and other coordinates within the `:xy-adjust-distance` bounds are shifted to match this lowest coordinate. Similarly, coordinates within the adjustment bounds of the highest coordinate (again, the device exterior) are shifted to match it. Finally, interior points are shifted to match one another as long as none are shifted beyond the adjustment bounds.

This simple adjustment algorithm works reasonably well, but shifts some components more than is necessary (by not shifting others far enough). A better adjustment algorithm was written, but not fully tested, as part of this contract.¹

3. Adjusts the z coordinates of the components within the `:z-adjust-distance` bounds. As with the xy adjustment, the simple adjustment algorithm is used to merge values. Since x values are rarely widely distributed over the height of the device, the simple adjustment algorithm produces good adjustments for typical devices.

The original component object includes both the adjusted xyz extent of the component (as changed above) and the original xyz extent. The GBB graphics displays used in IMCMA show both the original extent (using thin lines) and the adjusted extent (using thick lines), as shown in Figure 4.

The Complete-Model KS

This KS builds the component objects, based on the adjusted model produced by the `adjust-model-ks` KS. Recall that we delayed making the component surfaces, lines, and points and the prescribed convection, temperature, power-dissipation, and flux surfaces until the components were adjusted for modeling. This Lisp-based KS performs the following activities:

1. Creates 2D line, point, and surface and 3D line, point, and surface objects on the blackboard, linked appropriately.
2. Creates the prescribed convection, temperature, power-dissipation, and flux surface objects and links them to the appropriate component objects. These surfaces

¹That code is commented out in the prototype system source file.

An Architecture for Intelligent Multichip Module Reliability Analysis

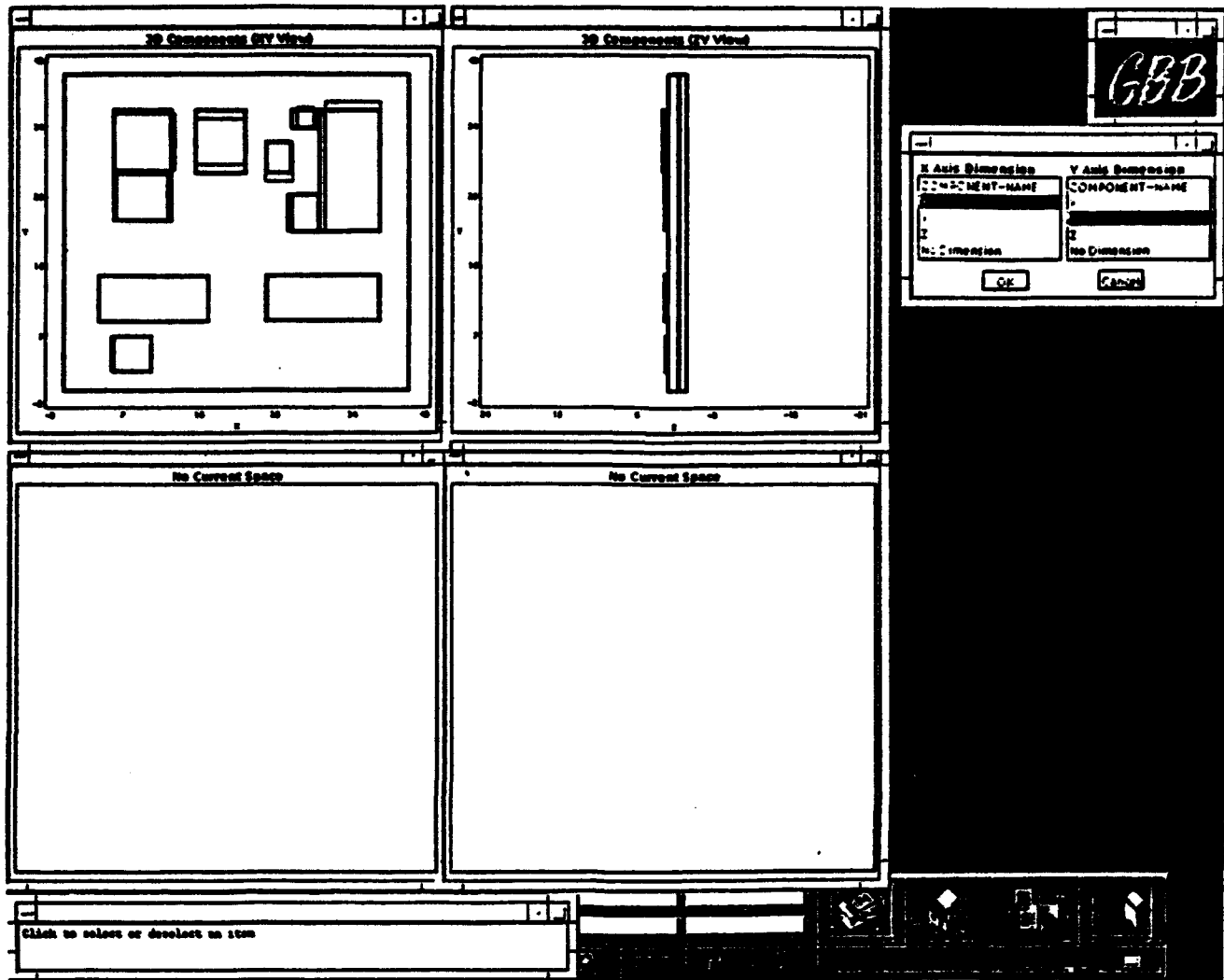


Figure 4: GBB Graphics Showing Test MCM Plan View (Adjusted Components)

are shown in black on the IMCMA graphics component display. (Point-heat-source and point-static-load objects are created by the combine-3d-meshes-ks KS later in the analysis.)

3. For each component, finds any adjacent chip components, linking the component objects as appropriate. An additional extrusion amount for adjacent chips is computed using a 4-color algorithm and stored with the chip objects. (This value is used during extrusion to effectively insulate chip sides from adjacent components; see the description of the extrude-component KS for details.)
4. Signals a generate-mapmesh-regions event (to trigger mapmesh region generation).
5. Signals a combine-3d-meshes event (to eventually combine the 3D component meshes, once they are generated).

The Find-Symmetry KS

The core of this CLIPS-based KS was written by Peter Rocci of Rome Laboratory. Its job is to identify 2D (xy) symmetry in the device. The Lisp-based interface to Peter's CLIPS code was written under this contract and performs the following activities:

1. Creates an ASCII input file for the CLIPS ruleset containing:
 - 2D region descriptions for each component 2D surface.
 - 2D point descriptions for each component.
2. Invokes the CLIPS ruleset on the ASCII input file.

The Lisp-based code to incorporate the results of this KS in further IMCMA processing was not written under this contract, as none of the example devices were symmetric at the highest modeling level.

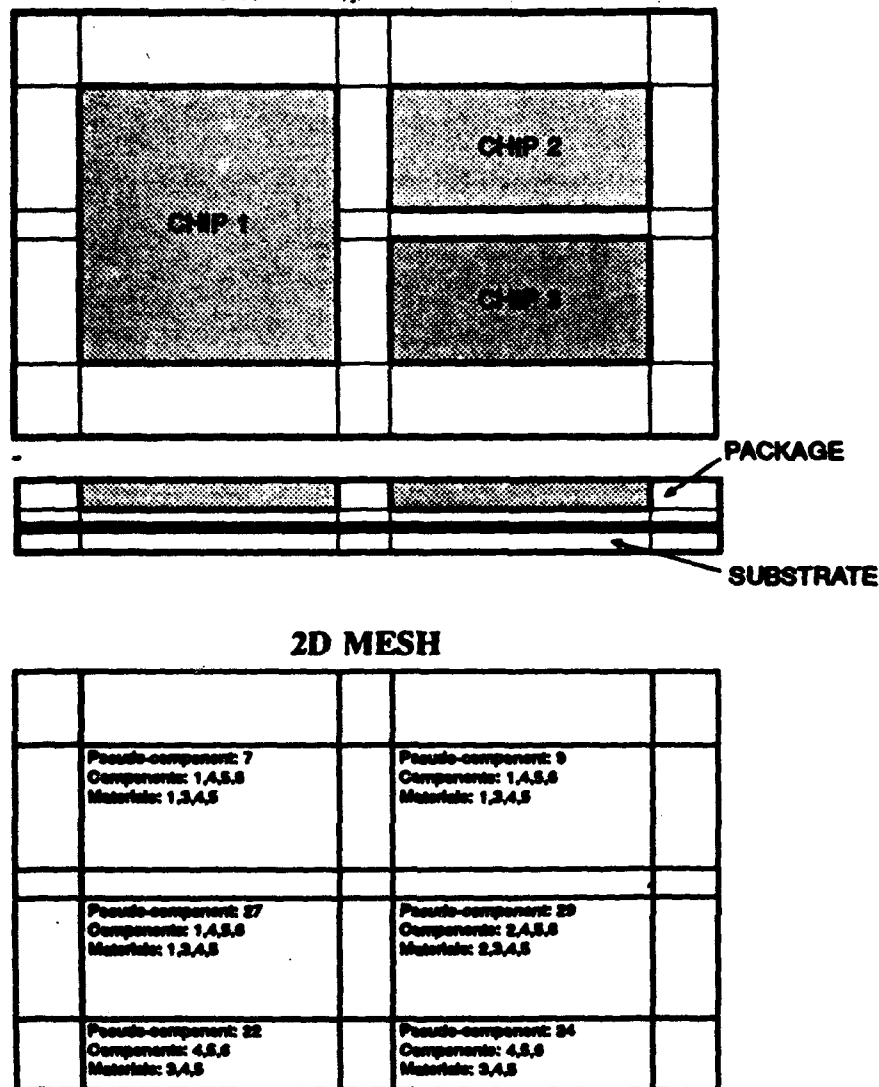
The Generate-Mapmesh-Regions KS

This KS generates 2D mapmesh regions based on the complete model produced by `complete-model-ks`. Mapmesh regions are the largest 2D "elements" that can be used to analyze the adjusted device model. The 2D mapmesh regions can be subdivided to create a more detailed 2D mesh (by the `generate-2d-mesh-ks` KS, described below).

This Lisp-based KS performs the following activities:

1. Given a list of xy transitions generated by the `adjust-model` KS, objects representing 2D mapmesh points, lines, and regions are created on the blackboard and linked to one another and to the component objects they represent.
2. "Pseudo-component" numbers are generated for each mapmesh 2D region. A pseudo-component number represents the set of component numbers that are associated with the mapmesh 2D region. As shown in Figure 5, a single 2D mapmesh region represents a number of different components in the full 3D model. The Sandia finite-element tools used to generate the 2D and 3D mesh require a numbering scheme in order to perform operations on the generated 2D and 3D meshes. Pseudo-components encode the information needed to appropriately edit the 2D mesh for 3D extrusion. The pseudo-component numbers are translated to actual component numbers (by the `extrude-component-ks`), and eventually to material numbers (by the `analyze-3d-mesh-ks`) later in the analysis process.
3. Changes the graphic display to include showing the 2D mapmesh regions (Figure 6).

An Architecture for Intelligent Multichip Module Reliability Analysis



A simple device and associated 2D mesh showing pseudo-component, component, and material numbers for selected 2D regions.

Figure 5: Relationship Between Pseudo-components, Components, and Materials

The Generate-2D-Mesh KS

This KS is called to generate a 2D mesh from the mapmesh regions created by generate-mapmesh-regions-ks.

1. Creates an ASCII input file for FASTQ containing:
 - The 2D point descriptions for the mapmesh regions.

An Architecture for Intelligent Multichip Module Reliability Analysis

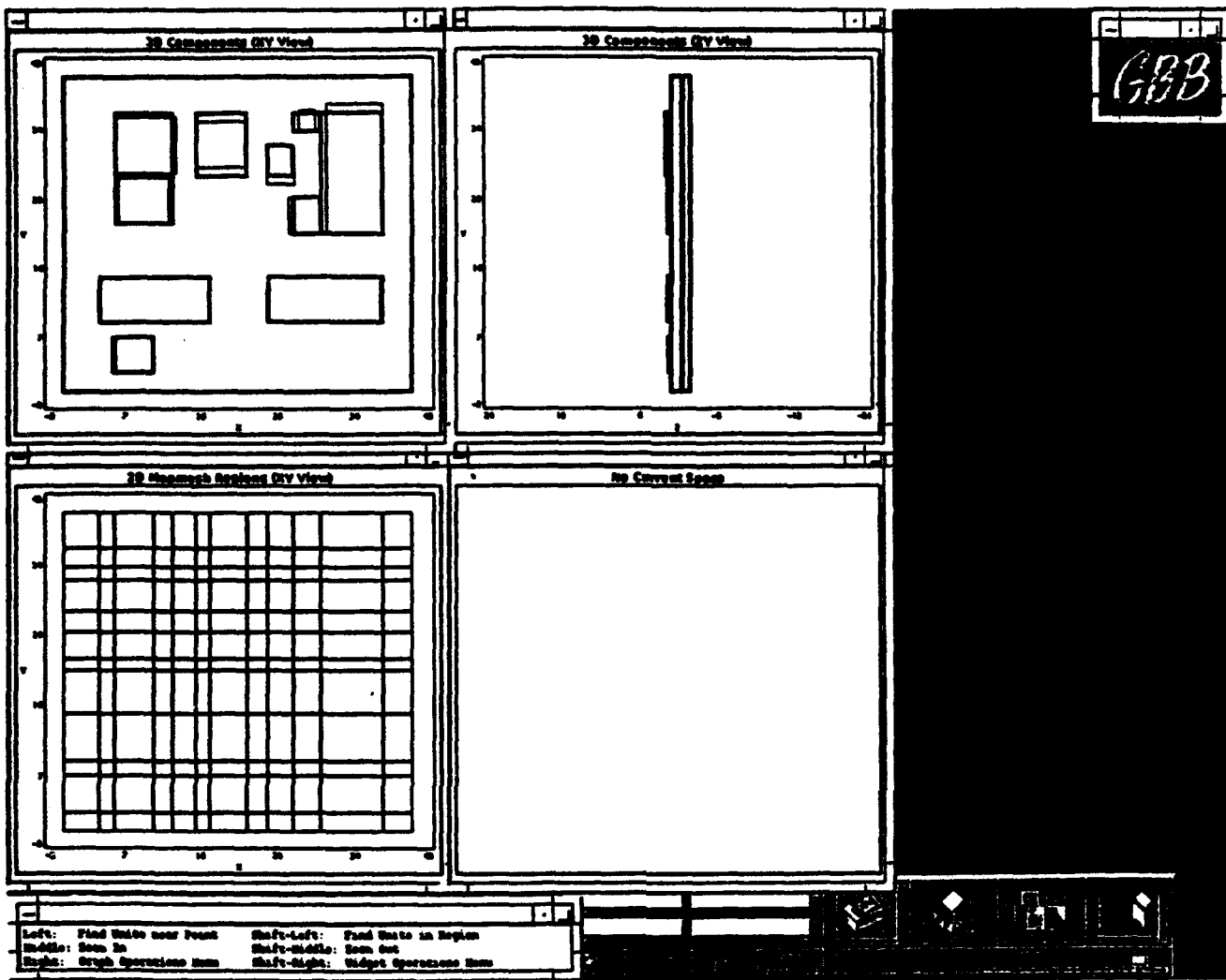


Figure 6: GBB Graphics Showing Test MCM Mapmesh Regions

- The 2D line descriptions for the mapmesh regions, which include the number of elements to create along the line and the element-size ratio associated with the line.²
- The 2D region descriptions for the mapmesh regions. These descriptions are sorted by region number within pseudo-component number to insure that FASTQ generates a 2D-mesh Exodus file with internal block numbers that match the pseudo-component numbers. (The need for this will be discussed in the description of the extrude-component KS.)

2. Invokes the FORTRAN FASTQ code on the ASCII input file.

²The number of elements along the line in the prototype is supplied by the parameter :number-of-elements when IMCMA is invoked. This will be replaced by a more intelligent mechanism being developed by Dale Richards at Rome Laboratory.

An Architecture for Intelligent Multichip Module Reliability Analysis

3. Reads the binary 2D-mesh Exodus file created by FASTQ to extract the 2D-node and 2D-element descriptions.³ A blackboard object for each 2D-node and 2D-element is created, the 2D-nodes are linked to the appropriate 2D-elements and the 2D-elements are linked to the appropriate component objects. An ordered list of the 2D-element's 2D-nodes is also stored with the 2D-element object to maintain the lower-left, upper-left, upper-right, lower-right node relationships to the element.
4. Changes the graphic display to show the 2D mesh (Figure 7).

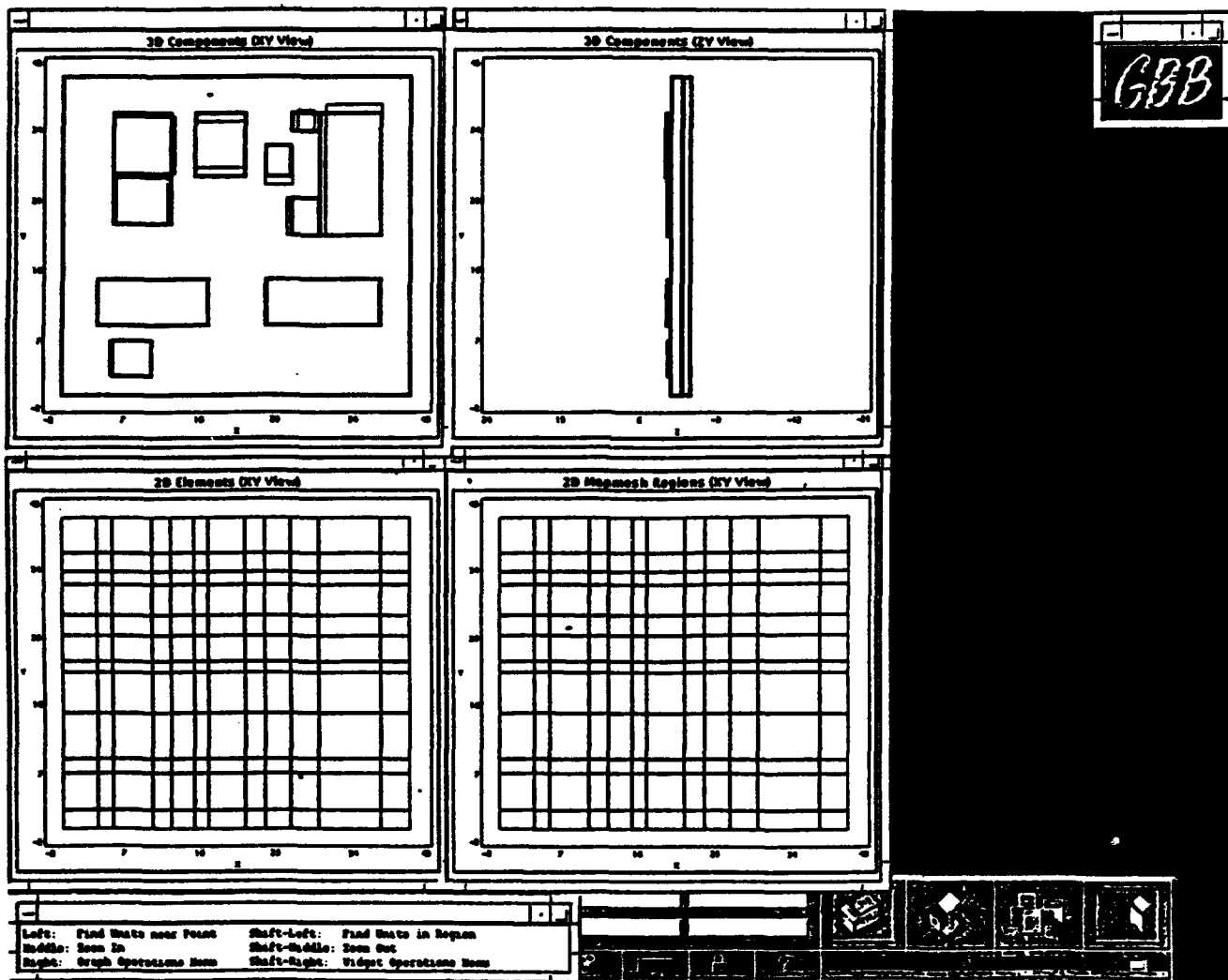


Figure 7: GBB Graphics Showing Test MCM 2D Mesh

³The 2D-mesh Exodus file reader was written by graduate research assistant, Venkat Manakkal.

The Extrude-Component KS

This KS is called once for each component in the device. Its job is to edit the 2D mesh produced by the generate-2d-mesh-ks KS to contain only those 2D elements appropriate for the component. It then extrudes the edited 2D mesh in the Z dimension to produce a 3D mesh of the component. As will be described below, chip components and non-rectangular components complicate this basic process.

The core of the extrude-component KS are two FORTRAN-based codes developed by Sandia National Laboratory: GJOIN (used for editing the 2D-mesh) and GEN3D (used to perform the extrusion). The Lisp-based interface to these codes written under this contract performs the following activities:

1. Determines any well locations associated with the component, based on chip placement. To eliminate the need to explicitly describe wells in the high-level, device-description file, chip components that overlap other components are assumed to be in wells of the same size as the overlap. This KS uses simple geometric reasoning to identify these wells and records them as part of the component object.
2. Determines the z-layers associated with this component. A component can be extruded in a single layer unless it has wells. When a component has wells, it must be extruded as a number of layers, based on the well depths, with each layer extruded separately. In the prototype implementation of this KS, some components are extruded with more layers than is required due to the use of simple geometric-reasoning heuristics in this step.
3. Performs the following on each layer of the component:
 - Determines which components are associated with this layer and which components within the xy spatial extent of this component must be deleted (such as a chip representing a well).
 - Determines the pseudo-components that are to be retained and the pseudo-components that must be deleted, given the above component analysis.
 - Determines how to renumber the retained pseudo-component block numbers to the actual component block number for the component.
 - Invokes GJOIN to edit the 2D-mesh file to delete the pseudo-component blocks and to renumber the blocks to the actual component number. Due to limitations in the Sandia codes, this requires that the target component block number must be in the retained pseudo-component block numbers. If it is not, the target block number (assigned to one of the pseudo-component blocks to be deleted) must be exchanged with one of the pseudo-component block numbers to be retained prior to the deletion. This code is quite cumbersome, due to the need to account for pseudo-component block numbers, component block numbers, and GJOIN's internal block numbers required for exchanging block numbers. The name of the edited 2D-mesh Exodus file is passed onto the next step.

An Architecture for Intelligent Multichip Module Reliability Analysis

- Invokes GEN3D to extrude the component layer. Chips receive special treatment to insure that their sides are insulated (decoupled) from adjacent chips or component wells. The Sandia codes do not provide a direct means of decoupling nodes, so an alternative technique was devised. Chips are extruded a slight additional amount (a multiple of the value of `*gjoin-merge-distance*`, based on a four-coloring algorithm), so that the nodes above the chip bottom do not coincide with the nodes of adjacent chips or wells.⁴ This technique causes problems, however, if a component is immediately above a chip (since the additional extrusion distance will cause the topmost chip elements to overlap the bottom elements of the component on top of the chip).
- The name of the 3D Exodus file created by GEN3D is saved with the component.

The Combine-3D-Meshes KS

This KS combines the sets of extruded 3D-meshes for all components that were produced by `extrude-component-ks`. The core of this KS is the FORTRAN-based GJOIN code developed by Sandia National Laboratory. The Lisp-based interface to these codes written under this contract performs the following activities:

1. Determines what is required to change all component-block numbers to material-block numbers as required by the FEECAP code used in the `analyze-3d-mesh-ks` KS. These changes must be done before the component-layer 3D-mesh files are merged into a single 3D-mesh file. As was the case with the `extrude-component-ks`, limitations in the Sandia codes require that the target material block number must be one of the component block numbers to be assigned to that material. If it is not, the target component block numbers must be exchanged to make this so. Since the component block numbers to be exchanged cannot already be intended for use as a material block number, all exchanges must be considered as a group. Again, this code is quite cumbersome, due to the need to account for material block numbers, component block numbers, and GJOIN's internal block numbers required for exchanging block numbers.

Once these exchanges have been determined, the FORTRAN GJOIN code is invoked on each of the component layer 3D-mesh Exodus files to perform the exchanges.

2. Invokes the the FORTRAN GJOIN code one last time to combine the individual component-layer 3D-mesh Exodus files into a single 3D-mesh Exodus file. The node merging threshold in GJOIN is set to be less than `*gjoin-merge-distance*` during this operation.

⁴This approach requires a setting of the node merging threshold in GJOIN of less than `*gjoin-merge-distance*`.

An Architecture for Intelligent Multichip Module Reliability Analysis

- 3. Reads the merged binary 3D-mesh Exodus file created by GJOIN (in the above step) to extract the 3D-node and 3D-element descriptions.⁵ A blackboard object for each 3D-node and 3D-element is created, the 3D-nodes are linked to the appropriate 3D-elements and the 3D-elements are linked to the appropriate component objects. An ordered list of the 3D-element's 3D-nodes is also stored with the 3D-element object to maintain the node relationships to the element (used for face number determination in the next step and in the analyze-3d-mesh-ks KS).**
- 4. Determines which 3D-element objects are on the top surface (meaning that they have no other 3D-elements directly above them).**
- 5. Links the component objects to the appropriate 3D-element objects and computes the 3D-element's "q" value from the proportional volume of the 3D-element to the component and the total power dissipation of the component.**
- 6. Links the prescribed-convection, prescribed-flux, prescribed-temperature, and power-dissipation surface objects to the appropriate 3D-element objects.**
- 7. Creates and links the point-heat-source and point-static-load objects to the appropriate 3D-element objects.**
- 8. Changes the graphic display to show the 3D mesh (Figure 8).**
- 9. Signals an analyze-3d-mesh-event with the .exo file.**

The Analyze-3D-Mesh KS

This KS analyzes the combined 3D mesh produced by combine-3d-meshes-ks. The core of this KS is the FORTRAN-based FEECAP program developed by Ian Grosse and his students in the Mechanical Engineering Department at the University of Massachusetts. The Lisp-based interface to FEECAP written under this contract performs the following activities:

- 1. Creates an ASCII input file for FEECAP containing:**
 - 3D-node descriptions containing for each node: X, Y, and Z constraint values, whether there are prescribed surface temperatures associated with the node, nodal temperature, and the node's X, Y, and Z location.**
 - material descriptions containing for each material used in the device: E, NU, error-tolerance, Exodus-file block number, TKr, TKs, TKz, beta-value, Alpha-r, Alpha-s, Alpha-z, and reference temperature.**
 - 3D-element descriptions containing for each element: the nodal connectivity, material number, and Q value.**
 - convection data containing for each element with a prescribed convection surface: the element number, face number, h, and ambient temperature.**

⁵The 3D-mesh Exodus file reader was written by graduate research assistant, Venkat Manakkal.

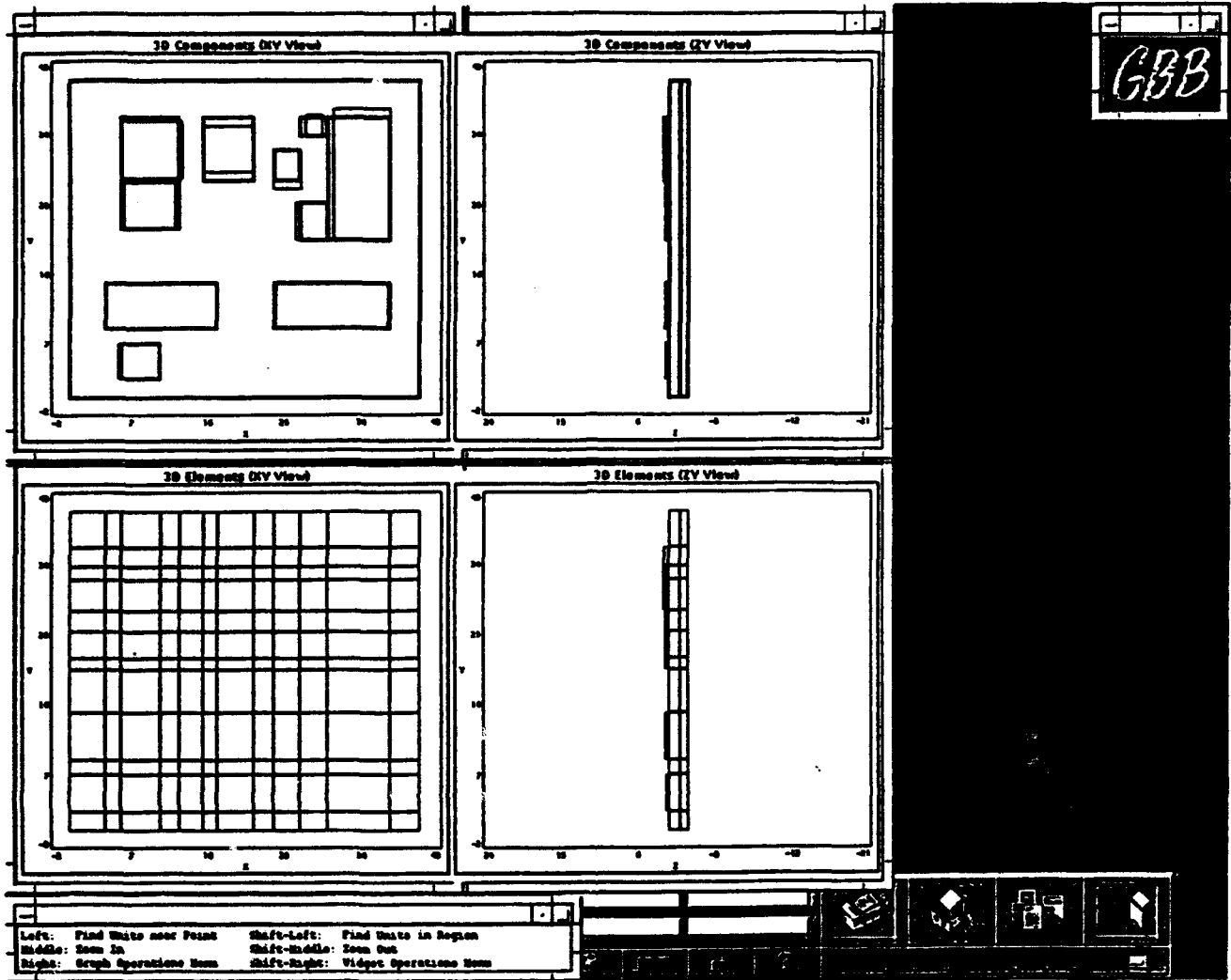


Figure 8: GBB Graphics Showing Test MCM 3D Mesh

- flux data containing for each element with a prescribed flux surface: the element number, face number, and flux value.
- point-heat-source data containing for each node with a prescribed point-heat source: the node number, and value.
- point-static-load data containing for each node with a prescribed point static load: the node number, and value.

2. Invokes the FORTRAN FEECAP code on the ASCII input file.

3. Reads the binary 3D-mesh Exodus file created by FEECAP to extract the analysis results appended onto the end of the mesh data.⁶ At present, these results include the nodal temperature, which is stored with each 3D-node object on the

⁶The Exodus-file analysis results reader was written by graduate research assistant, Venkat Manakkal.

An Architecture for Intelligent Multichip Module Reliability Analysis

blackboard, and the element error ratio which is stored with each 3D-element object.

4. Computes average and top-surface temperature values for each 3D-element using the nodal temperature of the element's 3D-nodes.
5. Determines the maximum and minimum nodal, element-average, and element-top-surface temperatures and the maximum and minimum element error ratios.
6. Changes the graphic display to show false-colored 3D-element top-surface temperatures (Figure 9).

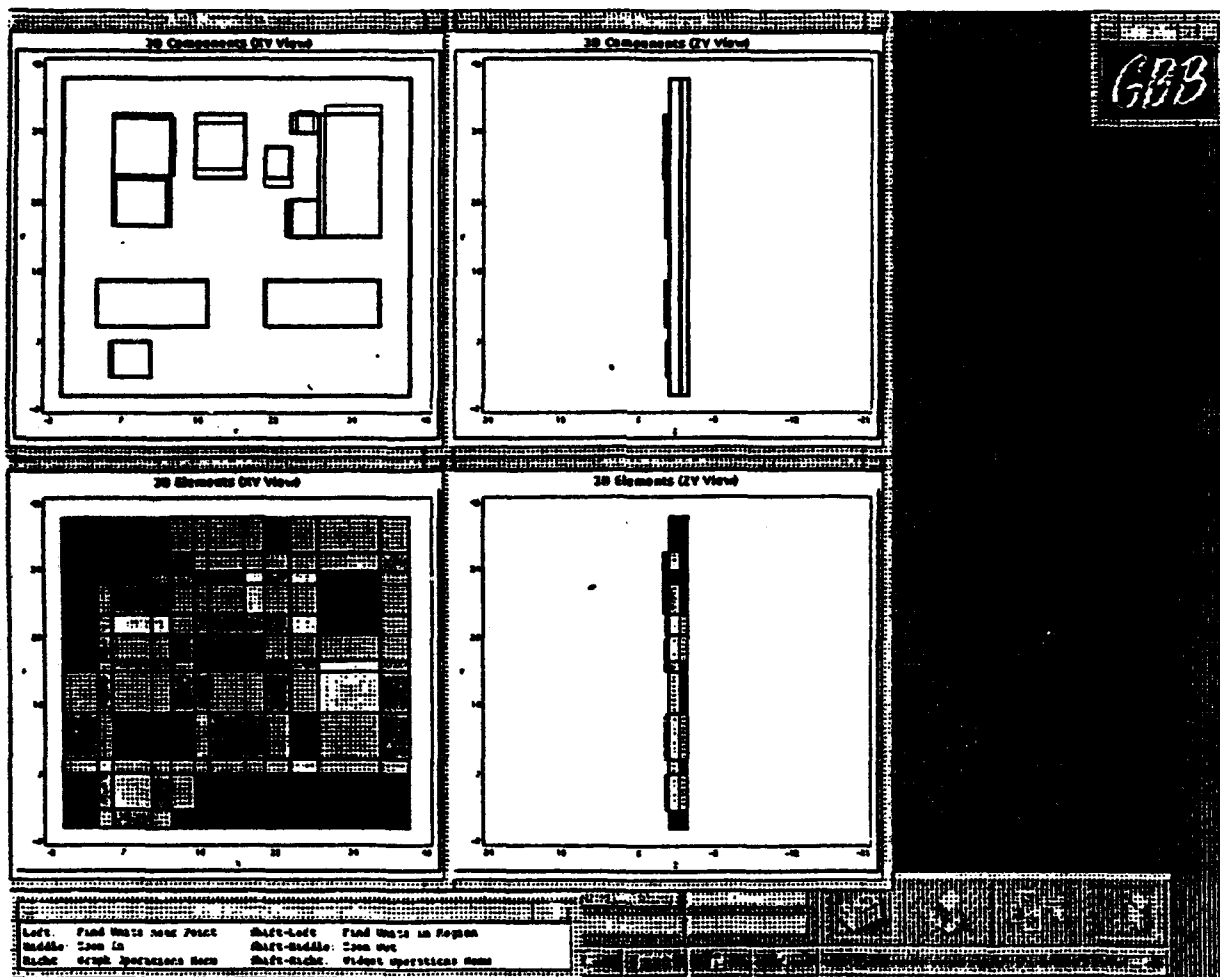


Figure 9: GBB Graphics Showing Test MCM 3D Mesh with Thermal Analysis

The analyze-3d-mesh-ks KS is the last KS to execute in the current IMCMA prototype. Once it completes, the user is free to interact with the objects stored on the blackboard by using GBB's interactive graphics facilities.

4 Lessons Learned

A number of system-integration lessons were learned during this one-year effort. In this section, we present some important observations that are applicable to other system-integration efforts.

Interfacing Issues

Integrating together independently developed tools as KSs in a blackboard architecture is made substantially easier by using the blackboard approach. However, the details of interfacing existing tools is complicated by assumptions built into these tools. Below, we briefly mention some of the interfacing issues that we encountered during IMCMA prototype development.

Interactive tools

The Sandia tools were designed to be used interactively by a human operator. These tools do not have an application program interface (API), and because we were to use the Sandia tools without modification, this required the interfaces to these tools to emulate the human operator commands by passing commands to shell scripts running the tools.

A major difficulty with these interfaces was the need to model the indexes assigned to blocks of elements by the Sandia tools. Normally, the operator is shown the indexes needed to manipulate the blocks, but the interactive interfaces written to make the Sandia tools robust to human operator errors prevented us from reading the index values. Instead, we were required to discover and model the index assignment algorithm used by the Sandia tools to enable the proper commands to be issued.

In short, the effort the Sandia-tool developers placed on building an effective human interface became a liability when these stand-alone tools were embedded as components in a larger integrated application.

Numeric tolerance

One surprise in this integration effort was the sensitivity of the various tools to numeric precision. In particular, GBB represents floating point values using double precision, while the FORTRAN-based tools use a mixture of single and double-precision. An example of the problems stemming from this difference is determining which 3d-element objects (produced by FORTRAN codes) correspond to which 3d component objects (created within GBB). Some 3d-elements extended slightly outside the component, while other neighboring 3d-elements incorrectly extended inside the component.

An Architecture for Intelligent Multichip Module Reliability Analysis

Initially we attempted to handle this issue by rounding when returning single-precision values from FORTRAN to GBB. This proved impractical and we adopted a strategy of adding a numeric tolerance to GBB's retrieval operations.

The flip side of numeric precision was used to isolate the chip-side elements from adjacent elements. Since the location specifications for the chips were all produced by GBB, identical values undergo the same losses in precision and remain identical inside the FORTRAN tools. This allowed us to use a very tight numeric tolerance inside the FORTRAN codes to keep slightly shifted node positions from being merged during 3d mesh merging.

Node ordering

Another interfacing incompatibility we encountered was the different node-ordering conventions used by the Sandia FORTRAN tools and the FEECAP FORTRAN program. The KS interfaces to these codes performed the translation from one node ordering to the other as part of their activities.

Avoiding Private Information

In the preliminary IMCMA design, detailed finite-element information was to be passed among the Sandia and FEECAP codes as private data. The IMCMA blackboard would contain high-level summary information, but detailed information would not be placed on the blackboard in the interest of avoiding the transfer of large amounts of data between the FORTRAN codes and IMCMA.

We soon learned that it would be critical that the detailed information generated during thermal and stress analysis would be needed by KSs involved in submodeling and reliability assessment. These KSs would need to be able to quickly determine where the hottest elements or the greatest deflections were located within a particular component or within the entire device. This meant that the results of the various KSs should not be retained as private, but must be placed on the blackboard so that all KSs can have access to them. Therefore, we placed individual elemental and nodal data objects onto the blackboard. This allows KSs to use the advanced retrieval capabilities of the underlying GBB architecture in performing categorization and analysis activities.

We would recommend taking a hard look at situations where private data is exchanged among KSs. Unless use of the blackboard is prohibitive for some reason, the increased flexibility and availability of placing the data on the blackboard is likely to result in significant advantages in overall system operation.

The Importance of Rapid Prototyping

During the one-year effort, the IMCMA system underwent a number of major redesigns. These redesigns were inevitable, and should be viewed as an important aspect of this form of research.

Initially, it appeared that the one-year IMCMA prototype effort was a relatively straightforward implementation effort. We would be automating the use of existing Sandia codes, using a meshing technique that appeared appropriate for MCM analysis. Yet, we were in for surprises that no amount of additional specification would have detected.

For example:

- The paving mesh-generation technique proved unable to handle realistic MCM devices, and so IMCMA was changed to support the map-mesh technique.
- The 3D-mesh extrusion process was changed several times as different techniques for isolating the sides of chips from well sides were explored. (Cross-sectional slices were replaced by individual-component extrusions to support tapered sides, and eventually, slightly expanded chip extrusions were used to force mismatched 3D-node positions on the chip sides.)
- The entire Sandia-tool interface was changed several times as unforeseen restrictions in the Sandia tools were discovered.

The use of a flexible and adaptable blackboard architecture allowed us to rapidly implement these changes as the prototype development effort progressed. In building such a unique prototype as IMCMA, it would have been impossible to have developed standard requirements and specifications prior to implementation. Our "evolutionary" system-development approach served us well in producing a prototype IMCMA system that, although different from our initial design in a number of aspects, is an effective analysis tool.

Problems Using the Sandia Tools

A requirement of the initial IMCMA prototype system was that it use unmodified, existing tools wherever possible. The FORTRAN-based Sandia finite-element tools (FASTQ, GEN3D, GJOIN, etc.) were selected for use in the IMCMA prototype due to their capabilities and availability at no cost to the government.

Although availability of these Sandia FORTRAN codes early in the prototype IMCMA development effort helped in demonstrating IMCMA's ability to generate effective finite-element meshes for various MCMs, reliance on these codes is a significant liability. The following problems are associated with the use of the Sandia codes:

- The Sandia codes are not well-integrated into IMCMA. One of the requirements of the IMCMA system was that any off-the-shelf codes used in the

An Architecture for Intelligent Multichip Module Reliability Analysis

system would be used without modification. The Sandia tools were not designed with an application-program interface (API) and had to be integrated into IMCMA as stand-alone programs. For use as IMCMA KSs, this required that the codes use ASCII text files and piped commands to receive input information, and that IMCMA read FORTRAN-generated binary output files. As a result, approximately one-half of the total processing time required for an initial analysis of an MCM device is expended in these interfaces.

- **The Sandia codes add minor capabilities.** Because a significant amount of reasoning about how to generate an appropriate 3D mesh is already performed within IMCMA, the Sandia codes perform little functionality beyond the book-keeping required to generate a 3D mesh from detailed specifications. Developing GBB-based IMCMA KSs to directly generate the 3D mesh from these specifications would be relatively straightforward.
- **The Sandia codes are inflexible.** A significant amount of code was added to IMCMA to work within the limits of the Sandia codes. Because of limitations in the ways the Sandia codes can be used, representations of "pseudo-components," components, and materials must all be used at appropriate times during the mesh-generation process. At this point, it is estimated that the amount of the additional code required to generate and translate among these representations is about as much as the amount of code that would be required to provide the Sandia-code functionality directly within IMCMA.
- **The Sandia codes add a system-administrative burden.** The Sandia codes are large and require an extensive library of routines to install, build, and maintain. Unlike the GBB-based IMCMA code, porting these codes to a new platform requires significant additional effort. Elimination of the Sandia codes would make IMCMA much more self contained and portable.
- **The Sandia codes are not universally available.** The Sandia codes are freely available for governmental and educational users only. They are not readily available to commercial users. This makes the IMCMA system unavailable for evaluation by many of the manufacturers and developers of multichip modules.

Performing the activities of the Sandia codes directly in IMCMA KSs would resolve all these problems, resulting in an IMCMA system that is smaller, faster, more portable, more available, and more self contained.

5 Summary of Accomplishments

In summary, the IMCMA prototype system now includes nine operational KSs that take a high-level device specification through model simplification, finite-element generation, and thermal analysis. These KSs include the use of stand-alone FORTRAN finite-element generation codes that have been integrated into IMCMA by using GBB. This

initial portion of the IMCMA system can produce a thermal analysis of a high-level MCM design description in a few minutes.

We are excited by our progress to date, although much remains to attain all the goals of the IMCMA approach. The three teams of individuals from the two UMass Departments and from Rome Laboratory that were assembled by Doug Holzhauser and Dale Richards successfully combined their diverse backgrounds, expertise, and viewpoints to produce an effective prototype IMCMA system. Our one-year effort would not have been so successful without these individuals and the management provided by Rome Laboratory.

In work being performed in related contracts, KSs for submodeling and reliability assessment, KSs for interactively creating high-level device specifications, facilities for managing libraries of standard components and materials, and facilities for displaying the analysis results to the operator are now being developed. Once on line, these new KSs and facilities will allow rapid differential comparison of the reliability of alternative MCM designs early in the design process.

6 References

- [1] W. J. Bocchi, J. A. Collins, and D. J. Holzhauser. Thermal stress analysis of integrated circuits using finite element methods. Technical Report RADC-TR-84-100, Rome Air Development Center, Rome, NY, April 1984.
- [2] W. J. Bocchi. Finite element modeling and thermal simulations of transistor integrated circuits. Technical Report RADC-TR-89-176, Rome Air Development Center, Rome, NY, October 1989.
- [3] J. H. Lau, D. W. Rice, and P. A. Avery. Elastoplastic analysis of surface mount solder joints. *IEEE Transactions on Components, Hybrids and Manufacturing Technology*, CHMT-10(3):346-357, September 1986.
- [4] P. A. Engel and C. K. Lim. Stress analysis in electronic packaging. *Finite Element Analysis and Design*, 4(1):9-18, June 1988.
- [5] J. H. Lau and C.G. Harkins. Thermal stress analysis of SOIC packages and interconnections. *IEEE Transactions on Components, Hybrids and Manufacturing Technology*, 11(4):380-389, 1988.
- [6] J. C. Glaser and M. P. Juairé. Thermal and structural analysis of a PLCC device for surface mount processes. *Journal of Electronic Packaging*, 3(3):172-178, September 1989.
- [7] S. Kawai. Structural design of plastic ic packages. *JSME International Journal, Series 1—Solid Mechanics, Strength of Materials*, 32(3):320-330, July 1989.
- [8] J. H. Lau. Thermal stress analysis of SMT PQFP packages and interconnections. *Journal of Electronic Packaging*, 3(1):2-8, March 1989.

An Architecture for Intelligent Multichip Module Reliability Analysis

- [9] **George Turklyyah and Stevern J. Fenves.** Feasibility study of a knowledge-based finite-element modeling assistant. Final report, Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, February 1988.
- [10] **Lee D. Erman, Frederick Hayes-Roth, Victor R. Lesser, and D. Raj Reddy.** The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, 12(2):213-253, June 1980.
- [11] **Daniel D. Corkill.** Introducing blackboard systems. *AI Expert*, 6(9):40-47, September 1991.

Appendices

A Device Definition Files

Device and material specifications are read from an IMCMA device definition file by the input-model-ks KS. An example device definition file for the Test MCM example (with the chips located on top of the substrate) is included below. Although based on real technology, this description does not physically exist and was created solely for purposes of demonstration, debugging, and illustration.

```

;;; -*- Mode:COMMON-LISP; Package:IMCMA; Base:10 -*-
;;; -*- File: SLATE:DIS$DISK:[GDB.REPORTS]GET.LISP -*-
;;; -*- Edited-By: Cork -*-
;;; -*- Last-Edit: Friday, September 24, 1993 13:19:36 -*-
;;; -*- Machine: GRANITE (Explorer II, Microcode 489) -*-

;;; *****
;;; *****
;;; *
;;; *
;;; *
;;; *
;;; *****
;;; *****
;;;
;;;
;;; Written by: Dan Corkill
;;;             COINS, UMass Amherst
;;; Modified by: Prasanna Katragadda
;;;             NE, UMASS, 04/01/93
;;; * * * * *
;;;
;;; 11-16-92 File created.
;;;
;;; * * * * *

(in-package "IMCMA")

;; This must come first:

(define-device "Test MCM"
  :filename "test"
  :size (40.64 40.64 1.27))

;; Materials come next:

(defmaterial :silicon
  :min-error-tolerance .1
  :max-error-tolerance .1
  :tk (0.1256 0.1256 0.1256)
  :alpha (0.233e-05 0.233e-05 0.233e-05)

```

An Architecture for Intelligent Multichip Module Reliability Analysis

```
:beta 0
:reference-temperature 0)

(defmaterial :Al203
:min-error-tolerance .1
:max-error-tolerance .1
:tk (0.025 0.025 0.025)
:alpha (0.8e-05 0.8e-05 0.8e-05)
:beta 0
:reference-temperature 0)

;; Now the device:

;; The aluminum-oxide carrier:

(defcomponent :SUBSTRATE-1 :substrate
:size (40.64 40.64 1.27)
:prescribed-temperature-surfaces ( (:bottom 30))
:material :Al203)

;; The Chips:

(defcomponent :CHIP-1 :chip
:size (13.0010 5.9890 .516)
:x (+ (/ 40.64 2.0) 10.1451)
:y (+ (/ 40.62 2.0) -8.4118)
:z (+ 1.27)
:xy-alignment :centered
:prescribed-flux-surfaces ( (:top 0.08))
:material :silicon)

(defcomponent :CHIP-2 :chip
:size (6.0000 6.0000 .516)
:x (+ (/ 40.64 2.0) -1.8409)
:y (+ (/ 40.62 2.0) 11.6080)
:z (+ 1.27)
:xy-alignment :centered
:prescribed-flux-surfaces ( (:top 0.08))
:material :silicon)

(defcomponent :CHIP-3 :chip
:size (12.9887 5.9890 .516)
:x (+ (/ 40.64 2.0) -9.6919)
:y (+ (/ 40.62 2.0) -8.4363)
:z (+ 1.27)
:xy-alignment :centered
:prescribed-flux-surfaces ( (:top 0.08))
:material :silicon)

(defcomponent :CHIP-4 :chip
:size (6.0000 6.0000 .516)
:x (+ (/ 40.64 2.0) -11.0530)
:y (+ (/ 40.62 2.0) 4.4870)
```

An Architecture for Intelligent Multichip Module Reliability Analysis

```
:z (+ 1.27)
:xy-alignment :centered
:prescribed-flux-surfaces ((:top 0.08))
:material :silicon)
```

```
(defcomponent :CHIP-5 :chip
:size (7.2430 7.2440 .669)
:x (+ (/ 40.64 2.0) -10.8227)
:y (+ (/ 40.62 2.0) 11.6080)
:z (+ 1.27)
:xy-alignment :centered
:prescribed-flux-surfaces ((:top 0.08))
:material :silicon)
```

```
(defcomponent :CHIP-6 :chip
:size (6.7090 16.7800 .429)
:x (+ (/ 40.64 2.0) 13.6987)
:y (+ (/ 40.62 2.0) 8.4359)
:z (+ 1.27)
:xy-alignment :centered
:prescribed-flux-surfaces ((:top 0.08))
:material :silicon)
```

```
(defcomponent :CHIP-7 :chip
:size (4.9090 4.6160 .361)
:x (+ (/ 40.64 2.0) -12.3240)
:y (+ (/ 40.62 2.0) -15.6350)
:z (+ 1.27)
:xy-alignment :centered
:prescribed-flux-surfaces ((:top 0.08))
:material :silicon)
```

```
(defcomponent :CHIP-8 :chip
:size (3.8500 4.9500 .480)
:x (+ (/ 40.64 2.0) 7.8190)
:y (+ (/ 40.62 2.0) 2.4290)
:z (+ 1.27)
:xy-alignment :centered
:prescribed-flux-surfaces ((:top 0.08))
:material :silicon)
```

```
(defcomponent :CHIP-9 :chip
:size (3.4190 5.1230 .264)
:x (+ (/ 40.64 2.0) 5.0069)
:y (+ (/ 40.62 2.0) 9.0850)
:z (+ 1.27)
:xy-alignment :centered
:prescribed-flux-surfaces ((:top 0.08))
:material :silicon)
```

```
(defcomponent :CHIP-10 :chip
:size (2.0090 2.8490 .567)
:x (+ (/ 40.64 2.0) 8.0830)
```

An Architecture for Intelligent Multichip Module Reliability Analysis

```
:y (+ (/ 40.62 2.0) 14.7080)  
:z (+ 1.27)  
:xy-alignment :centered  
:prescribed-flux-surfaces ((:top 0.08))  
:material :silicon)
```

```
;;; -----  
;;; End of File  
;;; -----
```

B Running IMCMA

This appendix describes how to run the IMCMA system.⁷ A device is analyzed by executing the function `run-ipcma`, described below.

```
run-ipcma device-description-filename &key :xy-adjust :z-adjust      [Function]
      :tolerance :analysis-type :display-mode
      :retain-intermediate-files :delete-all-files
      :break-on-errors :invoke-sandia-tools :load-mesh
      :save-mesh :edit-mesh
      :pause-on-display-mode-changes
      :default-number-of-elements
      :top-surface-temperatures :describe-operations
      :show-tool-commands :skip-toolfile-generation
      :no-tools :skip-2d-mesh-reading
```

This function is used to invoke the IMCMA system on a specified device-description file. The following keyword arguments can be specified:

Keyword	Default	Description
<code>xy-adjust</code>	0	Limits the XY adjustment of components during model simplification
<code>z-adjust</code>	0	Limits the XY adjustment of components during model simplification
<code>tolerance</code>	.001	The matching tolerance used within the GBB portion of IMCMA
<code>analysis-type</code>	:thermal	The analysis type (one of :thermal, :static, or :both)
<code>display-mode</code>	nil	Overrides the dynamic graphics display mode changes with a specified value (one of :components, :mapmesh-regions, :2d-elements, :3d-elements, or :3d-nodes)
<code>retain-intermediate-files</code>	nil	Controls the retention of all intermediate files generated by the GBB portion of IMCMA
<code>delete-all-files</code>	nil	Controls the deletion of all non-intermediate files generated by the GBB portion of IMCMA

⁷This appendix assumes the IMCMA system has been correctly installed and loaded into GBB.

An Architecture for Intelligent Multichip Module Reliability Analysis

Keyword	Default	Description
break-on-errors	nil	Controls signalling of an error versus a warning on problems detected by the GBB portion of IMCMA
invoke-sandia-tools	t	Controls the invocation of SANDIA and UMass tools
load-mesh	nil	Loads a saved mesh when <code>:invoke-sandia-tools</code> is nil (one of <code>:2d-mesh</code> , <code>:3d-mesh</code> , or <code>t</code>)
save-mesh	nil	Saves the generated mesh (one of <code>:2d-mesh</code> , <code>:3d-mesh</code> , or <code>t</code>)
edit-mesh	nil	Pauses IMCMA processing until continued by the operator (useful for manual inspection and editing of intermediate files)
pause-on-display-mode-changes	nil	Pauses IMCMA processing on all automatic graphics display mode changes (useful when interacting with the graphics at various stages of IMCMA processing)
default-number-of-elements	1	The default number of elements for each mapmesh region
top-surface-temperatures	t	Specifies whether the 3D-element display shows top-surface or element-average temperatures
describe-operations	t	Changes GBB event logging to brief descriptions of IMCMA processing
show-tool-commands	nil	Shows the output piped from IMCMA to Sandia and UMass tools
skip-toolfile-generation	nil	Skips the generation of the FASTQ ".fsq" input file and the FEECAP ".fee" input file
no-tools	nil	A shorthand for setting the arguments <code>:invoke-sandia-tools</code> to nil, <code>:skip-toolfile-generation</code> to t, and <code>:load-mesh</code> to t
skip-2d-mesh-reading	nil	Skips reading the 2D mesh data produced by FASTQ into IMCMA

C IMCMA Trace Output

As the IMCMA system is operating, it describes what is occurring to the user in the form of brief trace messages. An example of the trace output for the Test MCM example (with the chips located on top of the substrate) is included below. This example does not show the output generated by any of the external tools (FASTQ, GEN3D, GJOIN, FEECAP, or CLIPS).

```
> (run-imcma "get" :xy-adjust 1 :z-adjust .132)
```

```
:: Starting IMCMA Analysis:
```

```
:: Executing INPUT-MODEL-KS:
```

```
; Loading /usr/users/dis/cork/imcma/get.lisp.
```

```
:: Defining device Test MCM.
```

```
:: Instantiating the library and model blackboards.
```

```
:: Defining material SILICON.
```

```
:: Defining material AL203.
```

```
:: Defining component SUBSTRATE-1 (SUBSTRATE).
```

```
:: Defining component CHIP-1 (CHIP).
```

```
:: Defining component CHIP-2 (CHIP).
```

```
:: Defining component CHIP-3 (CHIP).
```

```
:: Defining component CHIP-4 (CHIP).
```

```
:: Defining component CHIP-5 (CHIP).
```

```
:: Defining component CHIP-6 (CHIP).
```

```
:: Defining component CHIP-7 (CHIP).
```

```
:: Defining component CHIP-8 (CHIP).
```

```
:: Defining component CHIP-9 (CHIP).
```

```
:: Defining component CHIP-10 (CHIP).
```

```
:: Shifting X from 6.2669992 to 5.904249.
```

```
:: Shifting X from 5.8758 to 5.904249.
```

```
:: Shifting X from 5.541499 to 5.904249.
```

```
:: Shifting X from 13.1188 to 12.6929.
```

```
:: Shifting X from 12.266999 to 12.6929.
```

```
:: Shifting X from 23.9646 to 23.791.
```

```
:: Shifting X from 23.617401 to 23.791.
```

```
:: Shifting X from 27.3985 to 26.806252.
```

```
:: Shifting X from 27.036402 to 26.806252.
```

```
:: Shifting X from 26.214 to 26.806252.
```

```
:: Shifting X from 30.6642 to 30.03585.
```

```
:: Shifting X from 30.064001 to 30.03585.
```

```
:: Shifting X from 29.407501 to 30.03585.
```

```
:: Shifting X from 37.3732 to 37.1694.
```

```
:: Shifting X from 36.9656 to 37.1694.
```


An Architecture for Intelligent Multichip Module Reliability Analysis

```
:: Shifting Y from 8.903699 to 8.891449.
:: Shifting Y from 8.879199 to 8.891449.
:: Shifting Y from 14.892698 to 14.880448.
:: Shifting Y from 14.868198 to 14.880448.
:: Shifting Y from 20.3559 to 20.309948.
:: Shifting Y from 20.263998 to 20.309948.
:: Shifting Y from 28.918 to 27.87575.
:: Shifting Y from 28.296 to 27.87575.
:: Shifting Y from 27.796999 to 27.87575.
:: Shifting Y from 26.8335 to 27.87575.
:: Shifting Y from 37.135902 to 36.02695.
:: Shifting Y from 36.442497 to 36.02695.
:: Shifting Y from 35.54 to 36.02695.
:: Shifting Y from 34.918 to 36.02695.

:: Shifting Z from 1.8369999 to 1.939.
:: Shifting Z from 1.786 to 1.939.
:: Shifting Z from 1.75 to 1.939.
:: Shifting Z from 1.699 to 1.939.
:: Shifting Z from 1.631 to 1.939.
:: Shifting Z from 1.5339999 to 1.939.

:: Executing COMPLETE-MODEL-KS:
:: Making 2D and 3D points, lines, and surfaces for component SUBSTRATE-1.
:: Making 2D and 3D points, lines, and surfaces for component CHIP-1.
:: Setting chip component 2 extrusion delta to 1
:: Making 2D and 3D points, lines, and surfaces for component CHIP-2.
:: Setting chip component 3 extrusion delta to 1
:: Making 2D and 3D points, lines, and surfaces for component CHIP-3.
:: Setting chip component 4 extrusion delta to 1
:: Making 2D and 3D points, lines, and surfaces for component CHIP-4.
:: Setting chip component 5 extrusion delta to 1
:: Making 2D and 3D points, lines, and surfaces for component CHIP-5.
:: Setting chip component 6 extrusion delta to 2
:: Making 2D and 3D points, lines, and surfaces for component CHIP-6.
:: Setting chip component 7 extrusion delta to 1
:: Making 2D and 3D points, lines, and surfaces for component CHIP-7.
:: Setting chip component 8 extrusion delta to 1
:: Making 2D and 3D points, lines, and surfaces for component CHIP-8.
:: Setting chip component 9 extrusion delta to 2
:: Making 2D and 3D points, lines, and surfaces for component CHIP-9.
:: Setting chip component 10 extrusion delta to 1
:: Making 2D and 3D points, lines, and surfaces for component CHIP-10.
:: Setting chip component 11 extrusion delta to 2

:: Executing GENERATE-MAPMESH-REGIONS-KS:
:: Creating mapmesh points, lines, and regions.....
:: Matching mapmesh regions with components.
:: Generating pseudo-component numbers (11 pseudo-components).

:: Executing FIND-SYMMETRY-KS:
:: Invoking CLIPS symmetry rules.
```

An Architecture for Intelligent Multichip Module Reliability Analysis

```
;; Executing GENERATE-2D-MESH-KS:
;;   Writing the FASTQ input file.
;;     Writing 2D point descriptions (169 points).
;;     Writing 2D line descriptions (312 lines).
;;     Writing 2D region descriptions (144 regions).
;;   Invoking FASTQ to generate 2D mesh.
;;   Reading the 2D mesh data.
;;     Reading 2D nodes (169 nodes).
;;     Reading 2D elements (144 elements).

;; Executing EXTRUDE-COMPONENT-KS:
;;   Extruding component SUBSTRATE-1 (1).
;;   Invoking GJOIN to edit 2D mesh for component SUBSTRATE-1 from from 0 to 1.27.
;;     Converting pseudo-component blocks 11 10 9 8 7 6 5 4 3 2 1 to component block 1.
;;   Invoking GEN3D to extrude component SUBSTRATE-1 from 0 to 1.27.

;; Executing EXTRUDE-COMPONENT-KS:
;;   Extruding component CHIP-1 (2).
;;   Invoking GJOIN to edit 2D mesh for component CHIP-1 from from 1.27 to 1.939.
;;     Converting pseudo-component block 3 to component block 2.
;;     Exchanging pseudo-component block 2 with 3.
;;   Invoking GEN3D to extrude component CHIP-1 from 1.27 to 1.939.

;; Executing EXTRUDE-COMPONENT-KS:
;;   Extruding component CHIP-2 (3).
;;   Invoking GJOIN to edit 2D mesh for component CHIP-2 from from 1.27 to 1.939.
;;     Converting pseudo-component block 9 to component block 3.
;;     Exchanging pseudo-component block 3 with 9.
;;   Invoking GEN3D to extrude component CHIP-2 from 1.27 to 1.939.

;; Executing EXTRUDE-COMPONENT-KS:
;;   Extruding component CHIP-3 (4).
;;   Invoking GJOIN to edit 2D mesh for component CHIP-3 from from 1.27 to 1.939.
;;   Invoking GEN3D to extrude component CHIP-3 from 1.27 to 1.939.

;; Executing EXTRUDE-COMPONENT-KS:
;;   Extruding component CHIP-4 (5).
;;   Invoking GJOIN to edit 2D mesh for component CHIP-4 from from 1.27 to 1.939.
;;     Converting pseudo-component block 7 to component block 5.
;;     Exchanging pseudo-component block 5 with 7.
;;   Invoking GEN3D to extrude component CHIP-4 from 1.27 to 1.939.

;; Executing EXTRUDE-COMPONENT-KS:
;;   Extruding component CHIP-5 (6).
;;   Invoking GJOIN to edit 2D mesh for component CHIP-5 from from 1.27 to 1.939.
;;     Converting pseudo-component block 10 to component block 6.
;;     Exchanging pseudo-component block 6 with 10.
;;   Invoking GEN3D to extrude component CHIP-5 from 1.27 to 1.939.

;; Executing EXTRUDE-COMPONENT-KS:
;;   Extruding component CHIP-6 (7).
;;   Invoking GJOIN to edit 2D mesh for component CHIP-6 from from 1.27 to 1.939.
;;     Converting pseudo-component block 5 to component block 7.
;;     Exchanging pseudo-component block 7 with 5.
;;   Invoking GEN3D to extrude component CHIP-6 from 1.27 to 1.939.
```

An Architecture for Intelligent Multichip Module Reliability Analysis

```
:: Executing EXTRUDE-COMPONENT-KS:
:: Extruding component CHIP-7 (8).
:: Invoking GJOIN to edit 2D mesh for component CHIP-7 from from 1.27 to 1.939.
:: Converting pseudo-component block 2 to component block 8.
:: Exchanging pseudo-component block 8 with 2.
:: Invoking GEN3D to extrude component CHIP-7 from 1.27 to 1.939.

:: Executing EXTRUDE-COMPONENT-KS:
:: Extruding component CHIP-8 (9).
:: Invoking GJOIN to edit 2D mesh for component CHIP-8 from from 1.27 to 1.939.
:: Converting pseudo-component block 6 to component block 9.
:: Exchanging pseudo-component block 9 with 6.
:: Invoking GEN3D to extrude component CHIP-8 from 1.27 to 1.939.

:: Executing EXTRUDE-COMPONENT-KS:
:: Extruding component CHIP-9 (10).
:: Invoking GJOIN to edit 2D mesh for component CHIP-9 from from 1.27 to 1.939.
:: Converting pseudo-component block 8 to component block 10.
:: Exchanging pseudo-component block 10 with 8.
:: Invoking GEN3D to extrude component CHIP-9 from 1.27 to 1.939.

:: Executing EXTRUDE-COMPONENT-KS:
:: Extruding component CHIP-10 (11).
:: Invoking GJOIN to edit 2D mesh for component CHIP-10 from from 1.27 to 1.939.
:: Invoking GEN3D to extrude component CHIP-10 from 1.27 to 1.939.

:: Executing COMBINE-3D-MESHES-KS:
:: Invoking GJOIN to merge extruded component meshes.
:: Invoking GJOIN to convert component numbers to material numbers.
:: Exchanging component block 1 with 2.
:: Reading the 3D mesh data.
:: Reading 3D nodes (423 nodes).
:: Reading 3D elements (179 elements).
:: Determining 3D element positions (144 top elements).
:: Linking components to 3D elements.
:: Component :CHIP-10 (1 element).
:: Component :CHIP-9 (1 element).
:: Component :CHIP-8 (2 elements).
:: Component :CHIP-7 (1 element).
:: Component :CHIP-4 (4 elements).
:: Component :CHIP-2 (6 elements).
:: Component :CHIP-5 (6 elements).
:: Component :CHIP-3 (5 elements).
:: Component :CHIP-1 (3 elements).
:: Component :CHIP-6 (6 elements).
:: Component :SUBSTRATE-1 (144 elements).
:: Linking prescribed convection surfaces to 3D elements (0 links).
:: Linking prescribed flux surfaces to 3D elements (35 links).
:: Linking power dissipation surfaces to 3D elements (0 links).
:: Linking point heat sources to 3D elements (0 links).
:: Linking point static loads to 3D elements (0 links).
:: Linking prescribed temperature surfaces to 3D elements (0 links).
```

An Architecture for Intelligent Multichip Module Reliability Analysis

```
:: Executing ANALYZE-3D-MESH-KS:
::   Writing FEECAP input file.
::     Writing nodal data.
::     Writing material data.
::     Writing element data.
::     Writing convention data.
::     Writing elemental flux data.
::     Writing thermal load data.
::     Writing static load data.
::   Invoking FEECAP.
::   Reading FEECAP analysis results.
::     Reading Exodus Header
::     Reading Coordinates
::     Reading Map Data
::     Reading Element Blocks
::     Reading Node Sets
::     Skipping 15 integers before QA header.
::     Reading Analysis Results...
::     Reading nodal temperatures (423 read).
::     Reading element error-ratios (179 read).
::   Nodal temperatures: 29.000841..33.882454
::   Maximum element-error-ratio: 6.043649
::   Computing element temperatures (Top: 29.675283..33.598553  Avg: 29.837643..33.399624).
nil
```

D Glossary

Blackboard (GBB)	A space or another GBB blackboard
GBB	Product trademark for Blackboard Technology Group's generic blackboard framework
CLIPS	Forward-chaining production rule system developed by NASA
Exodus file	Sandia finite-element mesh tool interchange file
FASTQ	Sandia finite-element mesh generator
FEECAP	UMass finite-element analysis package
GEN3D	Sandia 2d-to-3d finite-element mesh extruder
GJOIN	Sandia finite-element mesh file merging utility
IMCMA	Intelligent Multichip Module Analyst system
KS	Knowledge source
KSA	An activation of a knowledge source
Link	A bidirectional relationship between two GBB blackboard units
MCM	Multichip module
Space (GBB)	A blackboard level or plane
UMass	University of Massachusetts
Unit (GBB)	A blackboard object

**MISSION
OF
ROME LABORATORY**

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

- a. Conducts vigorous research, development and test programs in all applicable technologies;
- b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;
- c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;
- d. Promotes transfer of technology to the private sector;
- e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.