AD-A257 415

53.301-298

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 30, 1992	3. REPORT TYPE AND DATES COVERED Quarterly Technical Report, 7/92 -9/92

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
An MCM/Chip Concurrent Engineering Validation 1992 Third Quarter Technical Report	MDA972-92-C-0022

6. AUTHOR(S)

Dr. Hector Moreno

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Microelectronics and Computer Technology Corporation 3500 West Balcones Center Dr. Austin, TX 78759	HVE- 232-92

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
DARPA/DSO, Dr. H. Lee Buchanan 3701 N. Fairfax Dr. Arlington, VA 22203-1714	

11. SUPPLEMENTARY NOTES

DTIC
ELECTE
OCT 3 0 1992
S E D

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
No restrictions	

13. ABSTRACT (Maximum 200 words)

A software link was established between three commercially available Multi-Chip Module design systems: Allegro, EDGE and Finesse. The link was implemented through a database system based on the ROSE system developed under the sponsorship of the DICE program. The code was written in C++ and uses various methods to feed the information in and obtain it out of the design systems: IGES for Allegro, SKILL for EDGE and dfile for Finesse.

The DDR2 (Digital Drop Receiver, version 2) multi-chip module from Harris has been entered into the system and routed. and the information transferred to all the designers through the ROSE database.

14. SUBJECT TERMS		15. NUMBER OF PAGES 23
Concurrent Engineering, Multi-Chip Module, Computer-Aided Design		16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

92 10 29 013

92-28483

53-85

MCC TECHNICAL REPORT
HVE-232-92

An MCM/Chip Concurrent Engineering Validation, 1992, 3rd Quarter Report

Accesion For		
NTIS CRA&I		☒
DTIC TAB		☐
Unannounced		☐
Justification		
By		
Dist. ibution /		
Availability Codes		
Dist	Avail and / or Special	
A-1		

DTIC QUALITY INSPECTED

MCC High Value Electronics Division
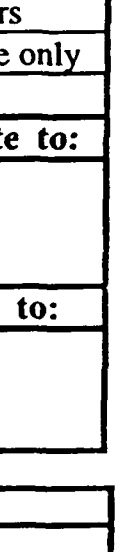
Document No. HVE-232-92
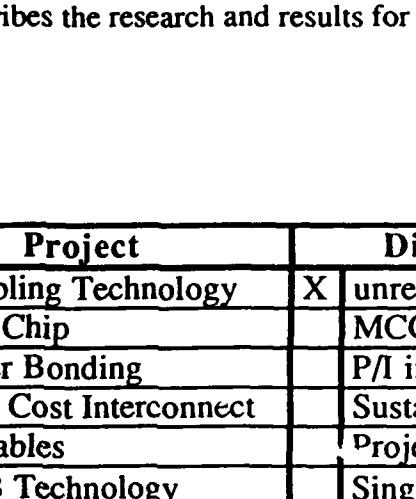
Author(s)   Hector Moreno

Title   An MCM/Chip Concurrent Engineering Validation, 1992, 3rd Quarter Report

Abstract   This report to DARPA describes the research and results for the 3rd quarter of the DICE contract.

Prior P/I
Documents

Contents		Project	Distribution	
X	Non proprietary	Enabling Technology	X	unrestricted
	MCC confidential	Flip Chip		MCC internal only
	HVE proprietary	Laser Bonding		P/I internal only
	Project proprietary	Low Cost Interconnect		Sustaining members
	Single client proprietary	Portables		Project members
Quality		TAB Technology		Single client use only
	draft	Open Systems - HW		
	external review	Open Systems - SW	Do not distribute to:	
X	finished	Govt. Open Systems		
External Publication		RwoH		
		PRI		
		X Government Contract	Also distribute to:	
		MSDA		
		Power Sources		

Position	Name	Signature	Date
author	Hector Moreno	*(signature)*	10 / 22 /92
project leader	Hector Moreno	*(signature)*	10 / 22 /92
reviewed by	Larry Smith	*(signature)*	10 / 22 /92
project manager	Don Orr	*(signature)*	22 / /92
program director	Dennis Herrell	*(signature)*	10 / 22 /92

MCC

Microelectronics and Computer Technology Corporation

An MCM/Chip Concurrent Engineering Validation

1992 Third Quarter Technical Report

Dr. Hector Moreno
High Value Electronics Division
Austin, TX 78727
September 30, 1992

Trademarks

Table of Contents

Summary

This report presents the technical progress during the third quarter of 1992, of the effort on Phase 1 of a project at the Microelectronics and Computer Technology Corporation (MCC) supported by the Defense Advanced Research Projects Agency (DARPA).

The principal project objective is to create and validate a concurrent engineering environment for the design of Multi-Chip Modules (MCMs).

On Phase 1, the focus is on physical design of MCMs. The objective in this phase is to integrate four different MCM layout systems: EDGE, Finesse, Allegro 6.0 and AutoCad. The principal technical problem addressed in the third quarter 1992 was the integration of the Allegro layout system from Cadence Design Systems to the ROSE concurrent engineering object-oriented data manager. This problem was solved through the development of a software system that links through the use of IGES (Initial Graphics Exchange Specification) objects and files.

In addition, Harris Electronic Design Automation (Harris EDA), subcontractor in this project, implemented a link between their MCM layout system, called Finesse, and the same database managed by ROSE. This brings to three the number of MCM CAD systems successfully integrated.

1. Project Objectives

1.1. Phase 1

To create a concurrent engineering environment suitable for the physical design of Multi-Chip Modules (MCMs) by integrating several different physical layout Computer-Aided Design (CAD) tools through data sharing.

To validate the use of ROSE as a concurrent engineering data manager .

To validate the concurrent engineering approach to MCM design.

To utilize the concurrent engineering environment to design MCMs.

1.2. Phase 2

Repeat the Phase 1 objectives but extending them individually to each of the following areas:

MCM modeling.

MCM simulation.

Design for testability.

This phase will begin upon funding release.

2. Progress Overview

2.1. Technical progress achieved

During the months of June through September of 1992, the following work was done:

The Finesse MCM CAD design system was integrated to the concurrent engineering environment defined by the layout schema which had been created during the second quarter of 1992. That schema was modified to take into consideration certain necessary design characteristics of interest to systems like Finesse and Allegro.

The Allegro software was integrated as well by making use of its IGES (Initial Graphics Exchange Specification) input/output interface.

We have started integration work between AutoCad (a layout design system package from AutoDesk, Inc.) and our concurrent engineering environment.

The parts definition and netlist for the DDR2 Multi-Chip Module (MCM) were received from Harris in ROSE data form and were the subject of an initial test for our Concurrent Engineering experiment.

We have accomplished everything we had planned for this period.

3. Technical Problems

3.1. Problems addressed

During the period covered by this review, we have focused exclusively on Phase 1 objectives. The principal technical areas that have been addressed are the following:

The improvement of the definition of the entities that are necessary for the description of the MCM layout data,

The integration of the Allegro CAD system through its IGES input/output interface.

The integration of the Finesse CAD system.

The implementation of a UNIX server to allow the Concurrent Design team to work efficiently in the integrated CAD environment.

4. General Methodology

4.1. Layout Information Model

In the last reporting period, we noted that an information model for the layout entities needed for MCM layout design had been implemented in the EXPRESS language by the definition of a schema. As the work progressed in the period covered by this report, it became apparent that we needed to make some adjustments to that schema. The main areas were those concerning arrays, ellipses and donuts.

There is a certain problem with the use of arrays in systems that keep connectivity information and that perform routing. Generally speaking, a netlist, which is a list of connections that need to be implemented, does not make reference to any type of arrays of basic elements, but directly addresses the elements themselves. Put another way, if we have an array of say, eight identical memory chips, the netlist will still refer to the individual chips and their pins and will not make any assumptions as to the fact we may like to arrange those eight chips in some geometric pattern, which we designate as an array. The array is a geometric implementation and is independent of the fact that the electrical connection still has to be made. Therefore, most if not all commercial CAD systems that perform routing will need to have arrays "exploded" or "flattened" for their routers to perform their work. Therefore, although we have technically speaking the array entity in our schema, it is necessary in practice to curtail its use if we are going to integrate CAD systems that do not make use of them, or know how to handle them.

As for the ellipse and donut entities, we found out that most systems do not handle them as such and that if a system handles any of them, it will be necessary to write them out of it as polygons that approximate them, for the other systems to accept them. Therefore, they have been dropped from the schema. However, most systems do handle circles, and we have added a circle entity.
The revised schema that implements these entities is listed as Appendix A of this report.

4.2. Allegro IGES link

In order to link the Allegro system it was useful to exploit its IGES input/output interface. Allegro is a system which is not very open from the input point of view. Today, there are essentially only two ways to get data into Allegro coming from external sources: one is to write a "script" which is essentially a series of interactive Allegro commands that one then has to replay in the Allegro environment to achieve the creation of a database, and the other is , as explained above, through its supported IGES interface. IGES, in turn, is a graphics exchange specification, which implies it has its limitations. It is hard to pass along electrical connectivity, for example. Fortunately, Allegro also allows for reading a netlist through another file-reading mechanism, called the NETIN command.

In this endeavor we have been fortunate to have access and use of the STEP Tools, Inc. IGES translation tool [ST],[NCGA], which is comprised by a large collection of ROSE object classes which have been defined through an EXPRESS schema to hold and manipulate the information stored by every standard IGES entity. In addition , STEP Tools has supported us in this project by adding quite a few non-standard IGES entities which have been defined by the Allegro developers in order to accommodate the many other properties that they found necessary to have for their I/O interface. IGES allows its users to define new entities which may be necessary in a certain range of entity numbers, although abusing this is clearly not recommended.

From this cursory discussion it should be apparent that an IGES file which is perfectly legal for Allegro will not necessarily be meaningful or even accepted by other systems, and indeed, we have found out that IGES files produced by Allegro can not be successfully read by AutoCad and vice versa. It turns out that when people talk about their system accepting IGES they are talking about their particular IGES dialect and not necessarily the standard form.

In any case, we found that even considering these limitations, Allegro's version of IGES was the only reasonable way to integrate it to the other systems, as the scripts route is not advisable, because it is impractical and exposed to many dangers due to the continuously evolving user interface, the appearance of new system features and the many variants that can be found even among users of a given system version. For example, consider the menus implementation, which can be user-customized. A script where one is opening a menu and clicking on "Show Element" could, in another menu customization be "Show Pins" and situations such as that one will happen often enough to disqualify the scripts option.

We have defined a mapping between the database objects expressed in Appendix A and the STEP Tools-defined IGES ROSE objects. Once the mapping was established, we have written C++ programs that implement the map in a bi-directional fashion. We can then go back and forth between the two versions of the objects. If one has the IGES ROSE version of the objects, one then uses a STEP Tools program to output the information as an IGES text file. Allegro accepts it and creates the Allegro "board" database which can then be edited by it. On the way out from Allegro, an IGES text file is created and STEP Tools has a corresponding program that reads it and creates the persistent IGES ROSE objects. Our software then can map them into the layout objects the other integrated systems accept.

4.3. Finesse link

The Finesse system has been linked to the ROSE database through an intermediate text file form called a "dfile", which is a standard textual database description form that Finesse provides its users, and which can be read back to recreate it (the Finesse database). Our original intention was to integrate Finesse in a direct fashion, without intermediate text forms such as the dfile, since Harris owns the source of Finesse. However, it was determined by Harris Electronic Design Automation that such a route would at this point take more time than that allowed by our project.

This link has been developed very much like the Cadence link that was reported in the previous report [PR].

4.4. Unix server

In order to integrate the users of the concurrent engineering environment that we are creating we have implemented a mechanism by which the Unix system will automatically update and distribute the changes to the ROSE database. Essentially, every user has his/her usual working environment, but in addition has two extra directories: One, labeled incoming and the other outgoing. If the user wants to update the ROSE database, he saves (using his CAD's system save command) to the outgoing directory (in addition to wherever he usually saves his CAD's system database). Once the CAD system finishes saving, the directory is "moved" or renamed to a specific location which the Unix server periodically inspects. If the server finds data there, it copies it to its own "incoming" directory, where it performs all necessary manipulations to change it into the standard ROSE form and from there translate it to all other integrated systems. This frees the user and his workstation from that overhead, as the server normally runs independently in another machine, with enough memory, disk and CPU power, typically a SPARCStation 2 with 50+ Megabytes of memory and, depending on the size of the database an number of designs being managed, enough storage space.

Once the translations are complete, the server copies the ROSE database to an official directory where it is kept, and a backup copy of the previous data is made. The translations are distributed by the server to all registered users according to their recorded CAD system type, and copies placed in their "incoming" directories. Users are registered in a special file that the server reads when it is initially invoked. The users inspect periodically their incoming directories and can update their designs with the new distributed

information. After they accept the changes, they can "move" their incoming directory to another location, and are ready to receive any new updates.

If they don't accept or agree with the changes, they need to communicate with the Concurrent design team members, discuss why the change is not acceptable and come to a consensus. If the change does stand, nothing is done. If the change is not accepted, then the old data is still in backup, or else the user that initiated the criticism on the change can submit his own data again to the server, and the change will be superceded.

This server mechanism has been implemented in this fashion as an interim solution. It would be useful to have ROSE implement both a database version control and a design checking/locking mechanism. Once these features are available, the server can invoke them on behalf of the user that is submitting the change or requesting a specific piece of information or version of the data.

4.5. DDR2 parts and netlist

The module we will design concurrently utilizing the system we are building is known as DDR2 (Digital Drop Receiver, version 2) and is proprietary to Harris Corp. The parts have been designed using Finesse and the netlist has been entered into the Finesse system. From there , that information has been made available to the other CAD systems through the ROSE database and the software we have built during this project.

5. Technical Results

5.1. Cadence Edge-Allegro-Finesse CAD systems integrated

During this reporting period, the principal technical results were that a bi-directional link was established and demonstrated between each of the Cadence Edge, Allegro, Finesse CAD systems and the ROSE central database The fundamentals of how this links work have been described above. The MCC information model was improved

The layout schema we have utilized is included in this report as Appendix A.

Images taken from the EDGE and Allegro systems are included in this report to illustrate the capabilities obtained so far.

5.2. Initial Concurrent Design Experiment

Figure 1 shows a screendump of a Sun Workstation where the initial data for the DDR2 module was entered. The system is Finesse, and the parts were created there and the netlist read into that system. Using our Finesse-ROSE interface, the initial database was created. It was transferred to both Cadence Edge and Allegro systems.

Figure 2 shows the design in the Cadence design system, and Figure 3 the initial design in the Allegro system. After being extracted from the ROSE database in text form, the netlist was read into the Allegro system through a NETIN command which which was fed to a blank Multi-Chip Module design. The physical design data for the DDR2 parts placement is then automatically created in that environment.

The next step was to route the module using Allegro's interconnect software. The results are shown in Figure 4.

The design was then written out and automatically saved in ROSE form. The ROSE database was translated back into the Cadence environment and the routed module is shown in the Cadence environment in Figure 5.
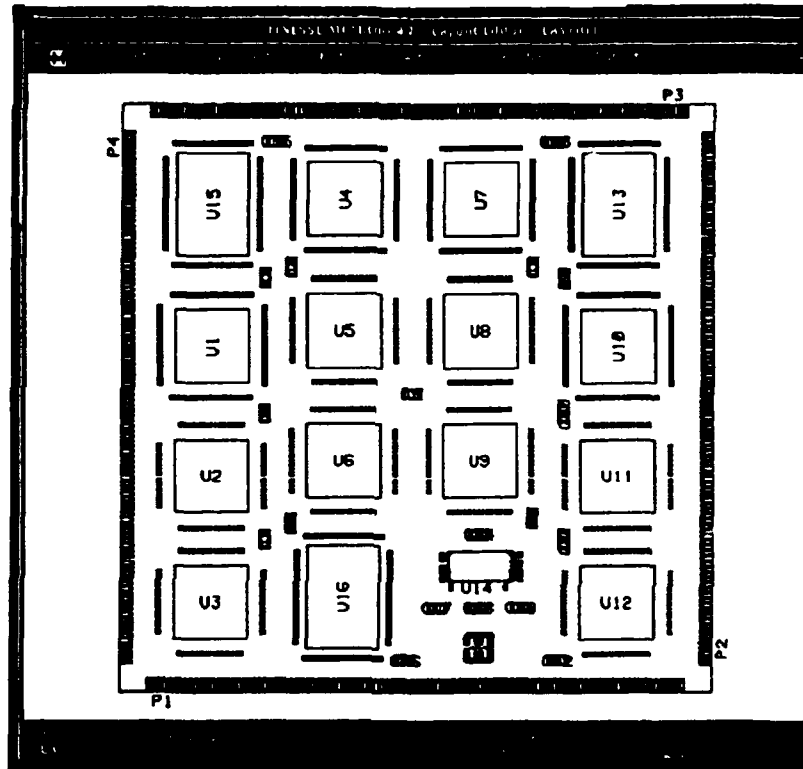
Figure 1. The Finesse environment with the initial DDR2 Parts placement. Displayed are the bond pad layer, the reference designator layer and the outline layers.

Figure 2. The DDR2 module parts placed in the Cadence EDGE system. Only the bond pad layer
is shown.
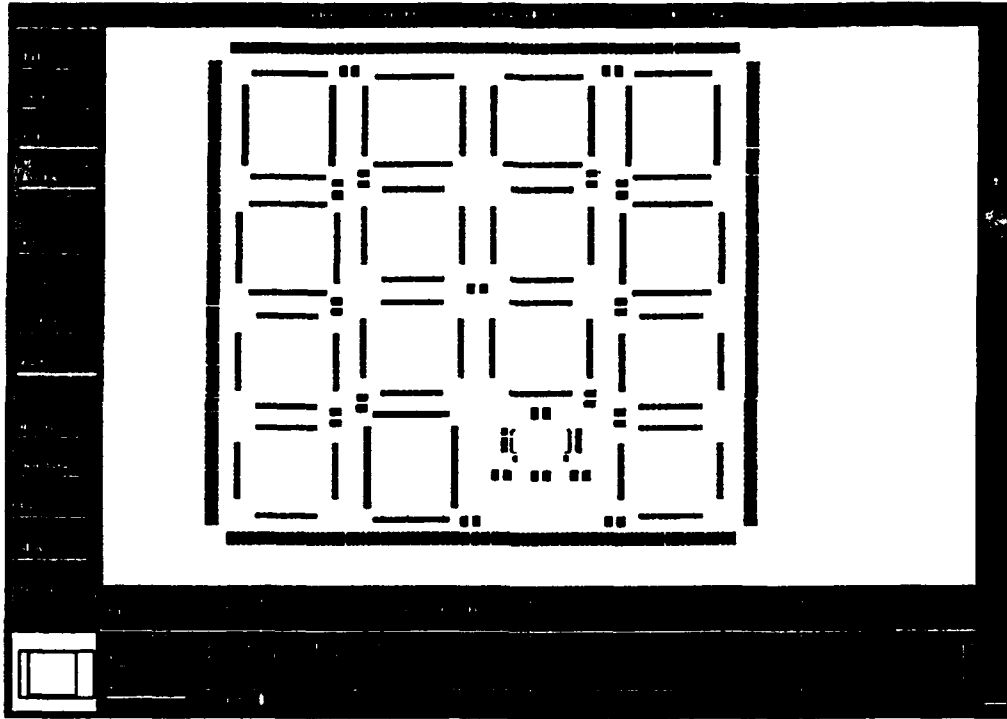
Figure 3. The same data as in Figure 2 , now in Allegro.
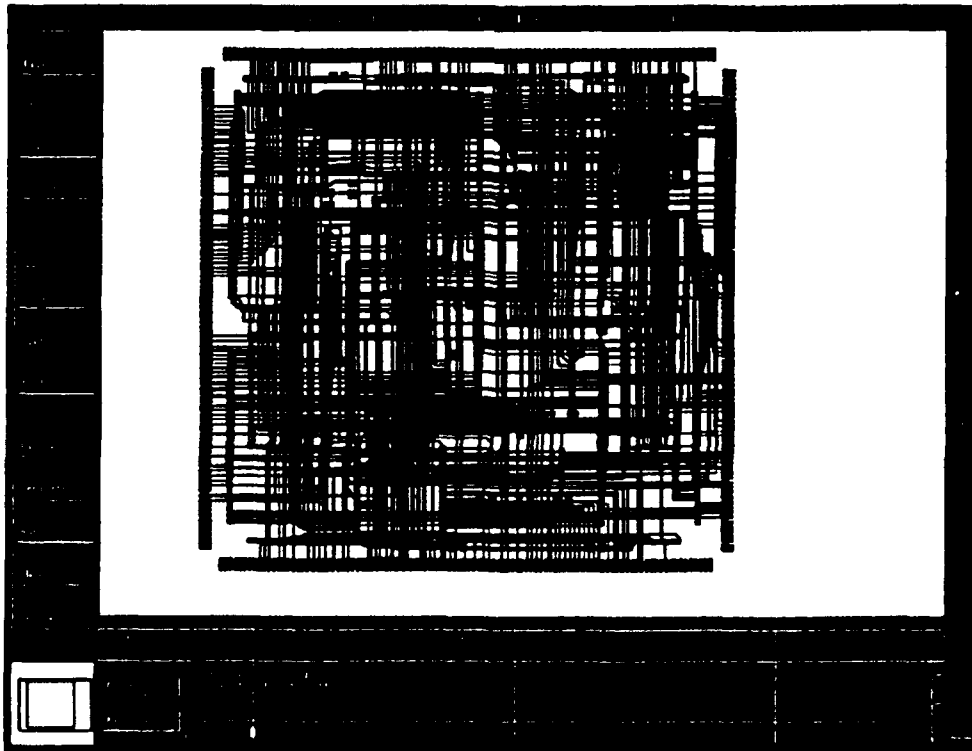
Figure 4. The DDR2 after being routed with Allegro, in that environment.

Figure 5. The routed DDR2 after transfer to Cadence Edge through the ROSE database.

6. Conclusions

6.1. Phase 1 - 1992 Third quarter

We have successfully integrated three of the anticipated four MCM layout systems to the ROSE data manager. This represents 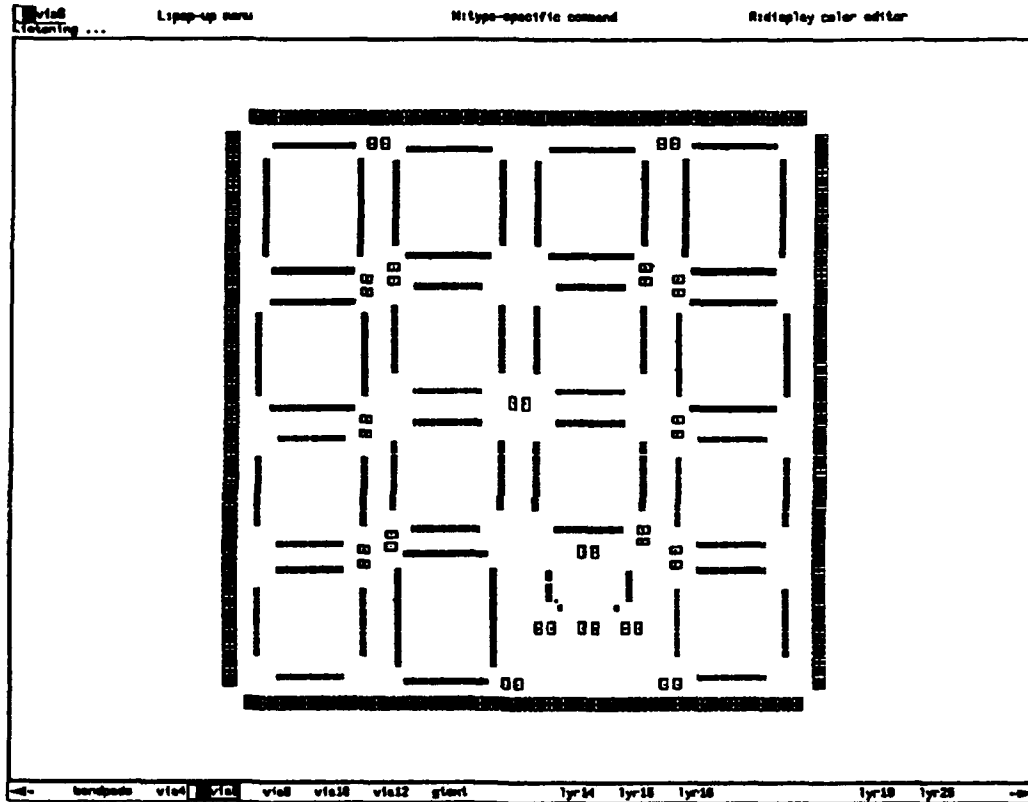substantial progress in the implementation of a concurrent engineering environment for the design of MCMs at MCC. We have also established a strategy and a methodology suitable for the concurrent design effort, which we will utilize and validate during the design activity portion of this project.

We are now in a position to check the performance and throughput of the ROSE system.

References

[PR] Hector Moreno, An MCM/Chip Concurrent Engineering Validation, 1992 Second Quarter Technical Report, MCC Report # HVE-171-92 (1992).

[NCGA] **IGES documentation can be obtained from the National Computer Graphics Association, IGES/PDES Organization, 2722 Merrilee Dr., Suite 200, Fairfax, VA 22031, (703)698-9600**

[ST] "Tutorial Manual for the ROSE++ data manager" and "Reference Manual for the ROSE++ data manager", STEP Tools Inc., Rensselaer Technology Park, Troy, NY, 12180, (518) 276-6751

Appendix A: A Layout Information Model, V. 1.1

```
(*••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••*)
(*                                    *)
(* Copyright 1992, Microelectronics and Computer Technology Corporation   *)
(*              All rights reserved              *)
(*                                    *)
(*••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••*)
SCHEMA layout;


ENTITY Property;
        name: STRING;
        value: STRING;
END_ENTITY;


ENTITY Layer
        SUBTYPE OF (LayoutObj);
        lyr: INTEGER;
END_ENTITY;


ENTITY Point
        SUBTYPE OF (LayoutObj);
        x: REAL;
        y: REAL;
END_ENTITY;


ENTITY BBx
        SUBTYPE OF (LayoutObj);
        ll: Point;
        ur: Point;
END_ENTITY;


ENTITY Line
        SUBTYPE OF (LayoutObj);
        lyr: Layer;
        nPath: INTEGER;
        path: LIST OF Point;
END_ENTITY;


ENTITY Path
        SUBTYPE OF (LayoutObj);
        beginExt: REAL;
        endExt: REAL;
        lyr: Layer;
```

```
        netNum: INTEGER;
        nPath: INTEGER;
        path: LIST OF Point;
        pathShape: STRING;
        width: REAL;
END_ENTITY;


ENTITY Rectangle
        SUBTYPE OF (LayoutObj);
        bBox: BBx;
        lyr: Layer;
END_ENTITY;


ENTITY Polygon
        SUBTYPE OF (LayoutObj);
        lyr: Layer;
        nPath: INTEGER;
        path: LIST OF Point;
END_ENTITY;


ENTITY Circle
        SUBTYPE OF (LayoutObj);
        center: Point;
        radius: REAL;
        lyr: Layer;
END_ENTITY;


ENTITY Label
        SUBTYPE OF (LayoutObj);
        draftingP: BOOLEAN;
        font: STRING;
        height: REAL;
        justify: STRING;
        labelType: STRING;
        lyr: Layer;
        orient: STRING;
        angle: REAL;
        theLabel: STRING;
        xy: Point;
END_ENTITY;


ENTITY Cell
        SUBTYPE OF (LayoutObj);
        blockName: STRING;
        cellType: STRING;
        objList: LIST OF LayoutObj;
END_ENTITY;


ENTITY Instance
        SUBTYPE OF (LayoutObj);
        master: STRING;
        blockRef: STRING;
        orient: STRING;
        xy: Point;
END_ENTITY;
```

```
ENTITY Mosaic
        SUBTYPE OF (LayoutObj);
        columns: INTEGER;
        rows: INTEGER;
        master: STRING;
        uX: REAL;
        uY: REAL;
        xy: Point;
        orient: STRING;
END_ENTITY;


ENTITY EIPin
        SUBTYPE OF (LayoutObj);
        pinName: STRING;
        pinNum: INTEGER;
        padRef: STRING;
        pkgRef: STRING;
END_ENTITY;


ENTITY Net
        SUBTYPE OF (LayoutObj);
        netName: STRING;
        netNum: INTEGER;
        elPinList: LIST OF EIPin;
END_ENTITY;


ENTITY LayoutObj
        SUPERTYPE OF (ONEOF (Point, Layer, Circle, Line, Path, Polygon,
                            Rectangle, Label, Cell, Instance, EIPin, Net));
        selfIdent: STRING;
        prop: LIST OF Property;
END_ENTITY;


END_SCHEMA;
```

Acknowledgements

Team Members

The efforts of Shaune Stark from MCC are acknowledged. He has written the SKILL procedures to interface to the Cadence EDGE database and contributed to the definition of the architecture and operation of the system, as well as the implementation of the Unix server.

Chuck Gannon from Harris Electric Design Automation has implemented the Finesse - ROSE interface.

Deborah Cobb from MCC has collaborated with her expertise with the Allegro system and its IGES characteristics. She has also collaborated in extracting device characteristics out of the ROSE database for the Allegro NETIN command.

Carroll Vance from MCC has implemented netlist extraction and creation for the Allegro system.

Christopher Hudnall from MCC has been helpful with all matters related to the installation of ROSE at MCC, as well as the UNIX support and server implementation.

External Support

It is a pleasure to acknowledge the support of Martin Hardwick, Alok Mehta, Blair Downie (STEP Tool's IGES translation toolkit) and in general of the employees of STEP Tools, Inc., in all matters regarding ROSE.