THE PHYSICAL DESIGN OF RPM

Jaeheon Jeong, Yongho Song and Michel Dubois

CENG 97-26

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089-2562
(213)740-4475
Fax: (213) 740-7290
{jaeheonj, yongho, dubois}@paris.usc.edu

December 1997

THE PHYSICAL DESIGN OF RPM

Abstract

RPM is a hardware emulator for multiprocessors which executes parallel applications with big data set size and produces performance statistics on the fly. In contrast with other hardware prototyping projects, RPM should sustain production level of reliability and flexible features to investigate various architectures.

We have built several revisions of RPM PCBs. The final revision of RPM PCBs has excellent reliability and incorporates new features which broaden our emulation space and facilitate the maintenance of the system.

This report describes the brief history of the revisions of RPM Printed Circuit Boards (PCB) from the initial version to the final version as well as the new features added to the final version.

1. INTRODUCTION

Table 1 shows the list of our PCB versions. The date on PCB is the completion date of the CAD work provided by EZFAB service at ISI. We have assembled 23 PCBs in total. For v1994.1, only two PCBs were assembled and one bare PCB was initially used as reference for the PCB design. Later the bare PCB was used to build the Futurebus+ interface card which connects an HP logic analyzer to the Futurebus+ backplane and provides an easy setup for the testing of the Futurebus+ interface design. Currently only v1997 boards are operational.

Version name	Date on PCB	# of PCBs Fabricated	Remarks
v1994.1	06/04/94	3	Initial design
v1994.2	06/04/94	9	For the site visit and major testing and debugging.
v1996	11/13/96	2	Trial version of final version
v1997	03/13/97	9	Final version

Table 1 RPM PCBs

2. RPM PCBs

2.1. v1994.1 PCB

This was a trial version of our initial design to filter out most of major problems before we built v1994.2 boards. The detail architecture can be found in [1][2]. Sparc processor could access each level of memory and remote memory via the Futurebus+ without coherence activities. Input/output devices such as serial ports and SCSI bus were operational. With this version, it was possible to test only part of the design thoroughly. Many problems were discovered in the part of the design not entirely covered by our simulation including some components such as LIFE chip and DRAM controller for which we did not have a simulation. At the early stage of debugging, two fatal errors were caused by the CAD work. The footprint layouts of Xilinx FPGA and LIFE chip were generated from the bottom view instead the top view. Additionally, SRAM modules could not be plugged into the sockets. We fixed these errors by mounting the sockets for FPGAs and the LIFE chip on the solder side and by adding piggyback boards for the SRAM modules. Because of the tight schedule, we decided to proceed building nine v1994.2 boards without complete debugging and testing.

2.2. v1994.2 PCB

In parallel with debugging v1994.1, we built 9 boards. We cleaned up the outstanding problems discovered in v1994.1 at the time as we started the CAD work for v1994.2. The architecture and the design remained as same as the initial design. However the PCB layout changed slightly and we added the second serial port to initiate the communication through the SCSI bus to the host since RPM is configured as a SCSI target device which cannot initiate SCSI transaction. The first serial port was used as a console port. Figure 1 and Figure 2 show a v1994.2 PCB and RPM with v1994.2 boards, respectively.

In the Summer of 1995, a limited number of RPM nodes were operational and we could run the SPLASH benchmark programs. Unfortunately, we could not run 9 boards in parallel because each board had different level of reliability. Some boards were working steadily while others needed frequent troubleshooting. At one point 8 boards were running together but later we were no longer able to run 8 boards in parallel. In July 1996, we had completed the debugging and testing successfully, but, because of the severe reliability problem we concluded that we will need a new revision with enough reliability to continue the project.

Several problems impeded the debugging and testing of this version. First, the PCB layout and design were poor. Many components were misplaced and overall placement was sparse. As a result, the length of many traces was unduly long and the design was plagued by usual electrical problems such as ground bounce, undershoots and overshoots. These problems were fixed by adding serial terminators or altering the timing.

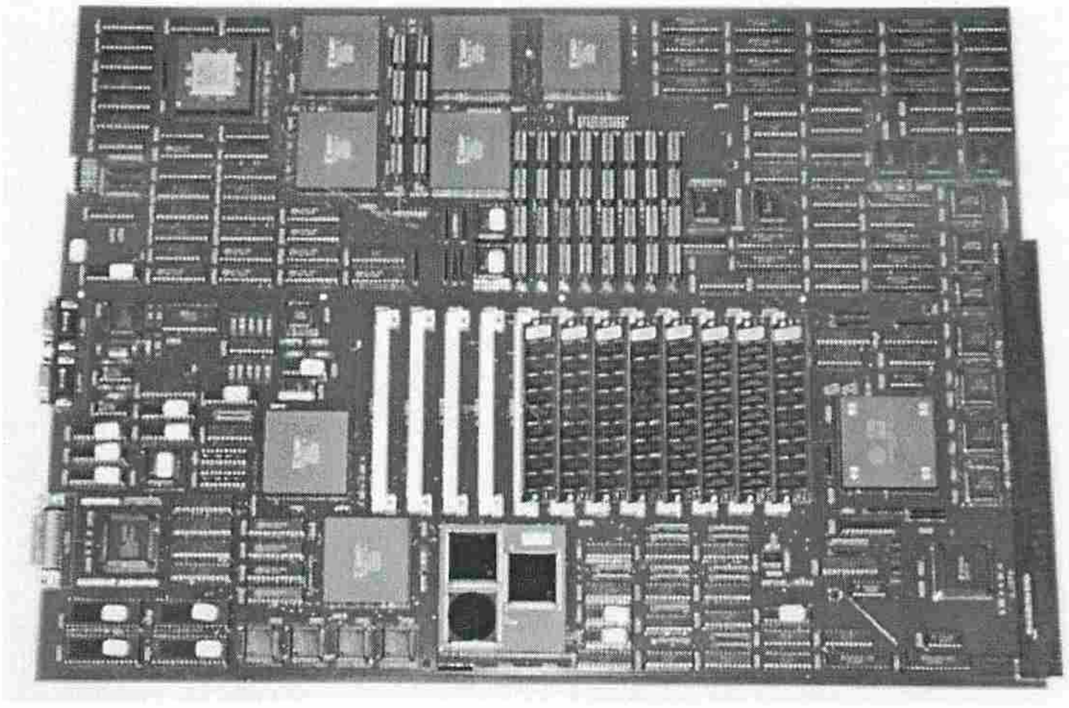

Figure 1 v1994.2 PCB

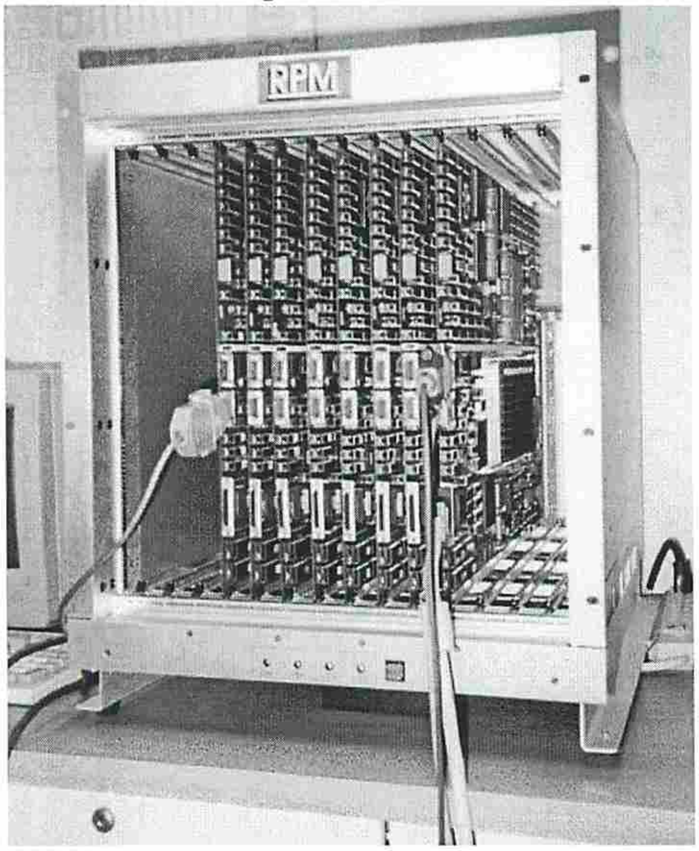

Figure 2 RPM with v1994.2 boards

Second, we experienced a serious problem with PCB fabrication and component selection. To meet the tight PCB design requirement of the Futurebus+ interface logic and to facilitate its testability, we decided to use PLCC (Plastic Lead Chip Carrier) type BTL (Backplane Transceiver Logic) transceivers and to mount PLCC surface mount sockets on both side of the PCB. However, we found out that the solder-mask for via holes was missing after the assembled PCBs were returned from the assembly house and this created many shorts between surface mount leads and via holes under BTL transceivers. This was the main source for the poor reliability of the boards and we spent a lot of time fixing the shorts until we developed our Futurebus+ interface board. In some cases, we were unable to fix the shorts or missing contacts. Also the quality of the Futurebus+ connectors was questionable.

Third, the design inherently had many isolated address and data buses. There were conflicts and oscillations due to metastability on the buses because the control timing on buffers between buses was less than marginal and buses were not properly terminated. The design of the internal bus was not really practical and consisted of too many components.

Finally, we could not invent nice features to help testing when we ran a program on more than two nodes. For example, among all the bugs and fixes, the system with more than three boards hung indefinitely when we ran our test program because we violated a simple fact that the LIFE chip cannot queue any additional packets in its internal FIFO until the completion of the current transaction. It was hard to track down which message was lost especially when the reliability of each board was questionable. This version cannot be mixed with other versions and is no longer operational since we took the parts to build v1997 PCBs. Two good boards of this version have been used in the port of Solaris since September 1995.

2.3. v1996 PCB

This is a trial version of v1997 before re-building the complete set of boards. The design is the same as v1997. We had finished the debugging and testing in February 1997.

3. v1997 PCB

As described earlier, the main goal of this version is to achieve solid reliability and to enhance or to add some features which were not provided in v1994.2. Figure 3 shows a v1997 PCB and Figure 4 shows the system with v1997 boards.

3.1. Reliability

In terms of reliability, we assessed that we should have same level of reliability and production quality as sustained by most commercial computer products since we will execute applications with big data set on top of our operating system which could run easily up to several days.

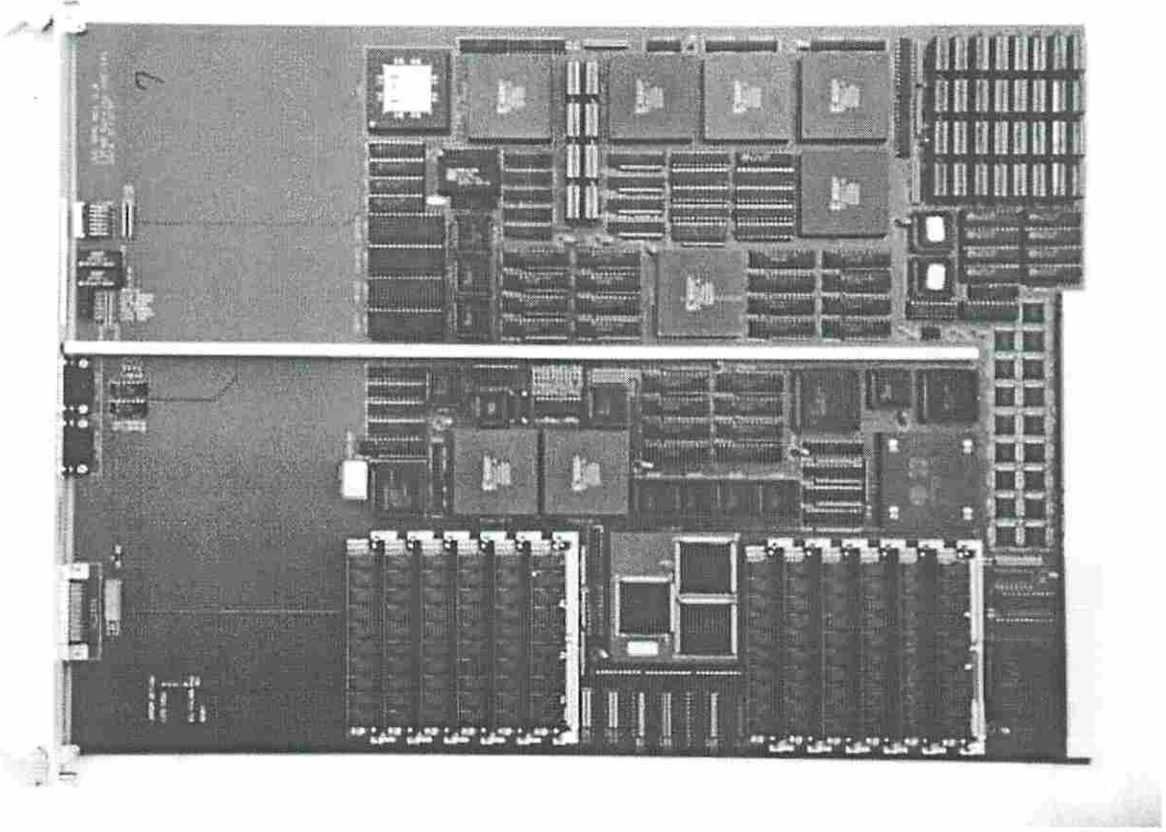

Figure 3 v1997 PCB

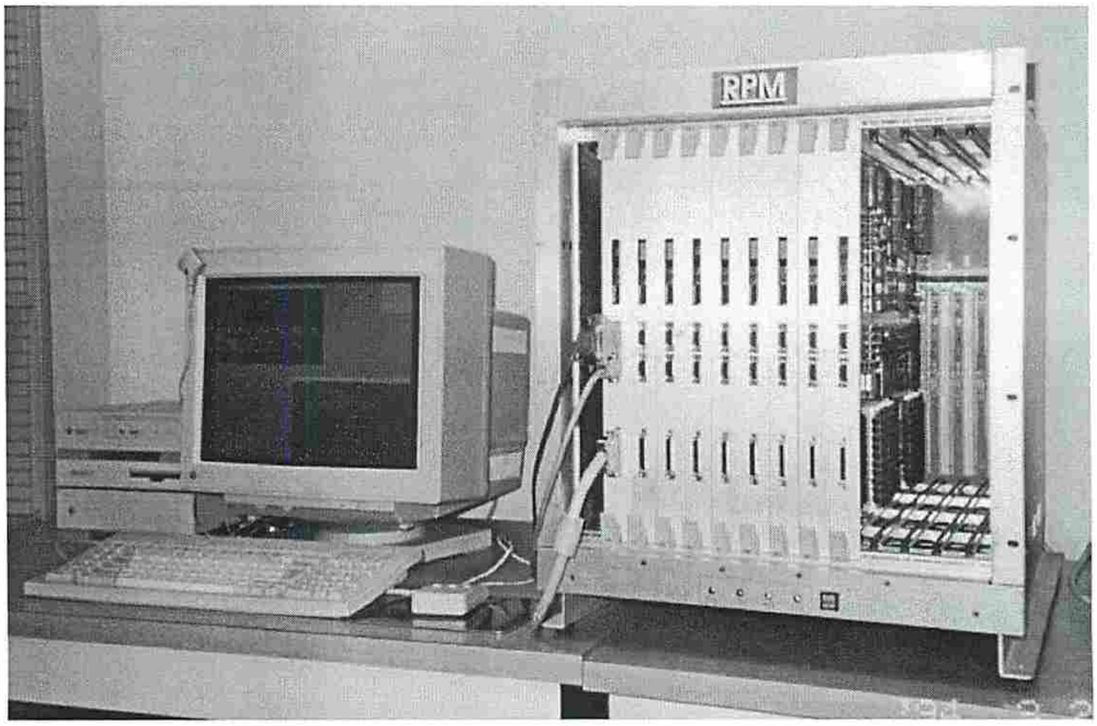

Figure 4 RPM with v1997 boards

With the limited resources available to us, it was impractical to execute various stress testing and to verify that our systems reaches the required reliability. Therefore we intuitively set our acceptance criteria as follows: RPM must pass diagnostic programs for seven days without single error in ambient condition, i.e., in our lab and we have passed the testing successfully.

First, to reach the required reliability we reduced the component count by removing and merging components. This includes migrating from low density programmable logic devices to high density ones such as MACH PLDs from AMD, removing redundant buffers, compacting the designs and replacing the internal bus with a Xilinx X4013 FPGA and a high density PLD, MACH230. We made the active PCB area as small as possible. We used Surface Mounting Technology (SMT) type decoupling capacitors which were mounted on the soldering side. Because of cost constraints, we could not replace some components such as DRAM and SRAM modules. After several iterations of preplacement with Powerview, we reached a nicely distributed placement of the design. Also we isolated the ground plane of the BTL transceivers from the digital ground plane to enhance their noise immunity.

Second, to improve the assembly quality, we changed the packaging of all BTL transceivers for the Futurebus+ from PLCC type to compact SMT type and put solder masks on the PCB.

Item	Before	After
Timer and RTC	- Short timer interval - No real timer clock	- Wide range of timer interval - Real time clock with low software overhead
SCSI	- No external buffer - High software overhead for read operation	- 4 KBytes external FIFO - Pseudo DMA relieves processor overhead
On-board status indicators	- None	- 7 LEDs show the status of FPGAs, LIFE chip and processor blocking
Board ID setting	- Futurebus+ backplane jumpers	- Board ID switch on the front panel of each board
FPGA programming interface	- TTL transceivers with no output controls	-BTL transceivers with output control
Hardware barrier synchronization	- Open drain by TTL transceiver (slow transition time)	- Wired-OR function of BTL transceivers (Fast transition time)
Internal bus and network interface control	- Discrete components - Not programmable	- one X4013 FPGA and one PLD - Flexible design

Table 2 New Features of v1997

The resulting design uses 20 less components and occupies 30% less area than v1994.2 even after adding the new features. The board size is unchanged because we had to keep same form factor to utilize the same card cage. Additionally we replaced the Futurebus+ connectors by connectors with low insertion force from AMP. To improve the physical strength of the boards, we made the PCB thicker than the previous versions and added custom front panels to insert the boards into the card cage.

3.2. New Features

Table 2 summarizes seven new features of v1997. Each feature is described in the following. We have modified the timer and have added a real time clock to meet the requirement of operating systems. Previously the timer interrupt interval could be up to 500ms but we found that this was not long enough when we scale down the emulated clock to less than 1 Mhz. Now the timer interval can be set up to 8 seconds. The real time clock keeps running even when the system is off and the processor can read the real time with minimum overhead.

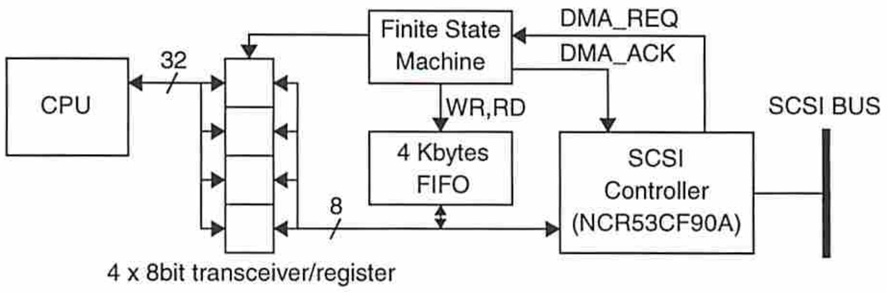

Figure 5 Block diagram of SCSI

Figure 5 shows the block diagram of the new SCSI design. To reduce the processor overhead for SCSI operations we have added 4 kbytes of external FIFO and an automatic data path multiplexor. To send a block to the SCSI bus, the processor simply writes a 32-bit word to the transceiver/register unit and then triggers the finite state machine to generate four consecutive 8-bit writes to the external FIFO. Once the external FIFO is filled up with a block to send, the processor sets the transfer count with the size of a block and sends a DMA write command to the SCSI controller. Then the SCSI controller sends a DMA request to the finite state machine and the finite state machine pumps the data from the external FIFO by asserting DMA acknowledge until the transfer counter reaches zero and the SCSI controller generates an interrupt to the processor. To receive a block from the SCSI bus, we simply set the transfer count and send a DMA read command to the SCSI controller. The SCSI controller generates an interrupt upon the completion of the transfer in cooperation with the finite state machine when a received block is in the external FIFO. The SCSI controller has a dedicated clock oscillator and its performance is less sensitive to the processor clock. The transfer rate between the external FIFO and the SCSI bus is about 3 MBytes/sec in asynchronous SCSI bus mode.

Seven LED indicators show the status of the processor, FPGAs and LIFE chip. This simple feature is very useful to "see" the status of RPM. The first LED shows whether the processor is blocked or not. If this LED goes off, the current memory access is incomplete for some reasons. Four LEDs are connected to the error signals of MC1 (first level cache controller), MC2 (second level cache controller), MC3 (memory controller) and NIC (network interface chip). Each LED goes off if the controller enters unreachable states. Finally the next two LEDs show the status of the LIFE chip.

Each board must be assigned a unique board identification number which affects the runtime configuration of the system such as the memory mapping and the role of each node. In the previous revisions, unique IDs were assigned by setting jumpers in each slot of the Futurebus+ backplane and the node with board ID zero was the master node and was also in charge of FPGA programming and IO operations. Therefore, to replace the master node by another, we had to move the new board in the preassigned slot. To improve flexibility, we have put two rotary switches on the front panel; one for the board ID and one for setting the role of each board.

The FPGA programming interface was not reliable and not flexible in the maximum configuration since TTL transceivers without output control were used. These have been replaced by BTL transceivers and its direction control which is set by the rotary switch on the front panel. Also the transition time of hardware barrier synchronization signals has been improved so that the transition can be captured in one processor cycle.

The internal bus has been replaced by an eighth FPGA called NIC and a PLD to improve its reliability. Figure 6 shows the block diagram of NIC. The NIC consists of three identical FIFO controls, data path and NIC controller. The FIFO control logic counts the exact number of messages in the bidirectional FIFO and transfers the messages between NIC and external buses autonomically. The data path is made of three sets of 2-to-1 multiplexors and the controller routes the messages with given priority when the data path is free. The PLD (not shown in Figure 6) is used to restore the message for accessing the internal registers of the LIFE chip.

Beyond its improved reliability, the NIC allows us to modify the message format easily. In the previous design, the format of the message header was fixed and the design of the internal bus was hardwired to the header format. For example, one bit in the fixed location of the message header tells whether the message is for MC or SLCC. The message type field is also fixed and this prevent us from adding new message types.

The NIC also allows us to extend our emulation space to message passing systems or virtual shared memory systems [3]. In these cases, the NIC is a communication assist such that it can issue DMA requests to the memory controller without processor intervention or generate interrupts to the processor upon receiving a message from the network. The message is deposited either in a specified region of main memory or in the internal registers of NIC depending on the type of message. To avoid potential deadlock situations caused by our architecture of the internal bus, we may need to limit the size of the incoming FIFO from the SLC bus to one so that processor can access the NIC anytime without being blocked by previous messages in the FIFO.

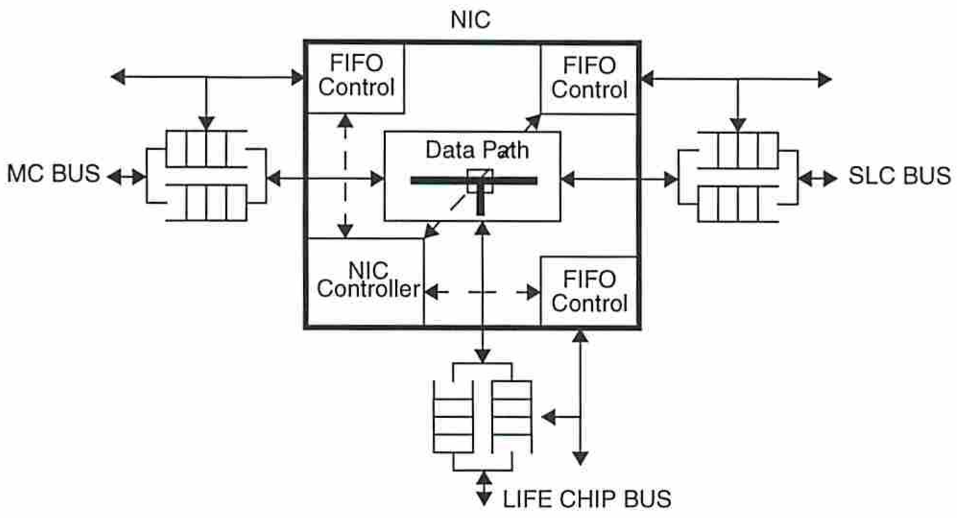

Figure 6 Block diagram of NIC

Acknowledgments. Funding for this work has been provided by the National Science Foundation under Grants MIP-9223812 and MIP-9633542. Besides the authors, several individuals have contributed to the project. In particular, we want to thank Per Stenström from Lund University (Sweden) and Massoud Pedram from EE-Systems, U.S.C. Luiz Barroso, Jacqueline Chame, Koray Oner, Adrian Moga, Sasan Iman, and Krishnan Ramamurthy participated in the design of the RPM emulator. Through their University Program several companies helped reduce the cost of the hardware and software needed for the project. These companies are Advanced Micro Devices, Synopsys, Viewlogic, Axil Workstations and Xilinx. Finally, John Granacki from ISI offered the services of EZFAB, which is part of the ARPA-sponsored Systems Assembly Project.

4. References

[1] Luiz Barroso, Sasan Iman, Jaeheon Jeong, Koray Oner, Krishnan Ramamurthy and Michel Dubois., "RPM: A Rapid Prototyping Engine for Multiprocessor Systems," IEEE Computer, Vol. 28, No. 2, February 1995, pp. 26-34.

[2] Koray Oner et el., "The Design of RPM: An FPGA-Based Multiprocessor Emulator," Proc. Third ACM Int'l Symp. Field-Programmable Gate Arrays, ACM Press, New York, 1995.

[3] David Culler et el., "Parallel Computer Architecture: Chapter 5, Large-Scale Distributed-Memory Multiprocessors," Draft, Morgan Kaufmann Publishers, 1996.