THE DESIGN AND CONSTRUCTION OF A LARGE HIGHLY PARALLEL RESEARCH COMPUTER (U) MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE C RIEGER ET AL. 30 APR 83

UNCLASSIFIED AFOSR-TR-84-0505 AFOSR-80-0270 F/G 9/2 NL
1. ABSTRACT

Design and construction of several prototype parallel processing computers has been completed. These computers, all versions of the ZMOB parallel processing system, led up to the 256 processor ZMOB still undergoing integration and testing. Operating system and support software were developed for the ZMOB architecture, including a Zmob simulator, a system for graphical display of communications activity, and several software debugging testbeds.
The Design and Construction of a Large Highly Parallel Research Computer

Design and construction of several prototype parallel processing computers has been completed. These computers, all versions of the ZMOB parallel processing system, led up to the 256 processor ZMOB still undergoing integration and testing. Operating system and support software were developed for the ZMOB architecture, including a ZMOB simulator, a system for graphical display of communications activity, and several software debugging testbeds.
2. Hardware

The ZMOB computer has up to 256 Z80 computers connected on an intelligent high-speed ring bus called the "conveyer belt". It has been described extensively elsewhere [Rieger 81] [Rieger 79] [Rieger et al 80] [Rieger 81] [Rieger et al 81]. Each Z80 has 64k bytes of memory, a parallel port, a serial port, and a floating point chip in addition to its conveyer belt connection. Initial versions of ZMOB will be completely dependent on a host computer for downloading of programs and for mass storage. Later versions may be self-sufficient in the area of mass storage, communications, and software.

Individual Z80's with associated hardware are called "moblets". A moblet consists of a Z80 and a "mailstop" or "mailbox". Mailboxes are tied together on the "belt". The mailstops associated with host communications are "Vaxstops".

2.1. Status

The ZMOB basic design has been checked in two-moblet versions both wire-wrapped and on printed circuit boards. Multiple processors can run independently, as many as necessary (up to 16 have been tested). A serious glitch was corrected in the processor card in January 1983, and since then there have been no processor failures.

Hardware debugging focused on communications between the processors. The four components of the communication system--the master clock, the clock cables and backplane, the backplane interconnect cables, and the mailstop boards--have each undergone several revisions. The clock circuit required tuning to provide optimal timing between the clock and index pulses. Several versions of clock cable were tried, including coax, twisted pair, and shielded twisted pair. The clock cables are critical because they must run for several yards to distribute clock to all the processors. Deglitching capacitors on the backplane have eliminated some noise, and pull-up resistors on the mailstop boards have been adjusted for better noise tolerance. A logic analyzer was an invaluable tool for debugging multiple mailstop signals.

Communication is reliable in a single 8 processor backplane. This was achieved approximately April 1, 1983.
2.2. Software

There are three main projects with applications for ZMOB. These are the Computer Vision project with Professors Rosenfeld and Davis [Kushner et al 82], the Distributed Numerical Computation project with Professors Stewart and O’Leary, and the Distributed Problem Solving project with Professor Minker. Each plans to use ZMOB to explore parallel processing algorithms in their specific domains, and each faces the problem of operating system support for coordinating the processes and accessing the powerful ZMOB hardware.

These three projects need similar basic operating system utilities [Bane et al 81] [Trigg 81] [Rieger et al 81]. The projects’ implementation strategies have been driven by the ZMOB conveyer belt hardware design but none of the projects envisions working at the machine code level at which the conveyer belt can be directly accessed. Each requires high-level language access to the full conveyer belt capability.

Common system utilities have been developed for use by all three projects. The following tools have been completed and are fully documented on-line in the Computer Science Department’s Vax computer:

1. gcc - Run Whitesmiths C compiler for Z80/ZMOB
2. belt - primitive routines for ZMOB mail stops
3. io1 - IO library for ZMOB
4. zfasl - Load a ZMOB program at 9600 baud
5. zgo - transfer a program to the ZMOB, and go
6. zload - load the ZMOB’s memory

Other working software for which final documentation is still under preparation includes:

(1) Z80 Simulator. This emulates the basic Z80 processor without the ZMOB environment. It was the first emulator to be completed and provided initial experience with the Z80 environment. It is a sufficiently detailed emulation to permit running CP/M and prolog, a capability much used by the distributed problem-solving group.

(2) Communications Simulator. This emulates multiprocessing and use of the mailstop and conveyer belt environment but executes Vax C code rather than Z80 machine language. It performs multiprocessing using basic Unix capabilities. The interface to the conveyer belt uses the same high level language calls that will be available on the moblets. Checkout has been via an implementation of a solution
to the dining philosophers problem.

(3) Communications Activity Display. This system accepts a stream of input describing the belt activity and displays on a color graphics display the resultant belt status. The input stream is now generated by the communications simulator (see above) and will eventually be generated by the 'monitor mail stop' on the actual ZMOB (see below). Figure 1 shows the relationship of the simulator and the display during the dining philosophers problem. An example of the activity display during the dining philosophers emulation is shown in figure 2.

(4) System Debugging Harness. This permits debugging new low level system software for inclusion in ZMOB. It allows the tester to manually create conditions which in actual practice would be quite rare, thus testing the robustness of the system. It consists of three parts: (a) the system software under test, (b) the moblet simulation system, and (c) the user interface display connected to both (a) and (b). The system software under test believes that it is running within a moblet, as emulated by the moblet simulation system. In fact, however, all attempts at communication are simply displayed to the user via (c). The user can then specify any arbitrary response on the part of the hardware, including timing relationships. The right half of the user's display is reserved for interaction about attempts at moblet communication. The left half of the display can be used by the system under test to display internal status of interest to the user.

(5) Multiple-window Communications Interface. This is an adaptation of the maryland window shell [Weiser et al 83] to problems of communications with the ZMOB. The window shell allows arbitrarily sized and positioned rectangular windows on a single CRT screen, each functionally equivalent to an independent terminal. A typical application is to open several windows and run a communication process to a different moblet in each window.
Zmob High-Level Simulator

Figure 1.

Zmob (simulated)

Vax
Figure 2.
REFERENCES

[Bane et al 81]

[Kushner et al 82]

[Rieger et al 81]
Rieger, Chuck, Bane, Robert, Stanfill, Craig, Trigg, Pandy, and Weiser, Mark, Three ZMOB Papers for the CAPAIDM Hot Springs Conference, Univ. of Md., CSD, TR1099 September 1981.

[Rieger 79]

[Rieger et al 80]

[Rieger 81]
Rieger, C., ZMOB: Hardware From a Users Viewpoint, University of Maryland CS TR-1042, 1981.

[Rieger et al 81]

[Rieger 81]

[Trigg 81]

[Weiser et al 83]