1.Classification *INPE-COM.4/RPE*
C.D.U.: 681.326.32

2.Period

4.Distribution Criterion

internal [ ]

external [*X*]

3.Key Words (selected by the author)

MICROPROGRAMMING
COMPUTER AIDED DESIGN
EMULATOR SYSTEM
MONITOR PROGRAM

5.Report No.
INPE-2050-RPE/299

6.Date
May, 1981

7.Revised by
[illegible]
Eduardo W. Bergamini

8. Title and Sub-title

EMMMAC - A COMPUTER AIDED MICROCONTROL MEMORY EMULATOR

9.Authorized by
[illegible]
Nelson de Jesus Parada
Director

10.Sector *DEE/DEL* Code

11.No. of Copies *09*

12.Authorship *Marcos Antônio Cardoso Cruz*
Paulo Farias Santos Amaral

14.No. of Pages *13*

15.Price

13.Signature of first author [illegible]

16.Summary/Notes

This paper describes a microprogramming support system composed by a special interface and a monitor program. The Emulator System allows the user to load, test, modify and debug a microprogram in a real time environment, during design phase of a microprogrammed device without PROM burning. It can also aid in the diagnosis and maintenance of the final prototype. The "EMMAC" is a computer aided design tool, and a version of it is running at Digital and Analog Systems Laboratory of INPE.

17.Remarks

Submitted for presentation in the International Conference on CAD/CAM (IFIP/SUCESU) to be held in São Paulo Oct. 21-23, 1981

EMMAC - A COMPUTER AIDED MICROCONTROL MEMORY EMULATOR

Marcos Antônio Cardoso Cruz

Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq
Instituto de Pesquisas Espaciais - INPE
12200 - São José dos Campos, SP, Brazil

Paulo Farias Santos Amaral

Universidade Federal do Espirito Santo
Departamento de Engenharia Elétrica
Campos de Goiabeiras
29000 - Vitória, ES, Brazil

ABSTRACT

This paper describes a microprogramming support system composed by a special interface and a monitor program. The Emulator System allows the user to load, test, modify and debug a microprogram in a real time environment, during design phase of a microprogrammed device without PROM burning. It can also aid in the diagnosis and maintenance of the final prototype. The "EMMAC" is a computer aided design tool, and a version of it is running at Digital and Analog Systems Laboratory of INPE.

1. INTRODUCTION

Since its introduction by Wilkes in 1951, microprogrammed logic has been largely employed. Presently, various smart components are available to its implementation.

A more general microprogrammed structure is depicted in Figure 1. Basically, the structure is composed by three major parts: sequencer, next address logic and microcontrol storage.

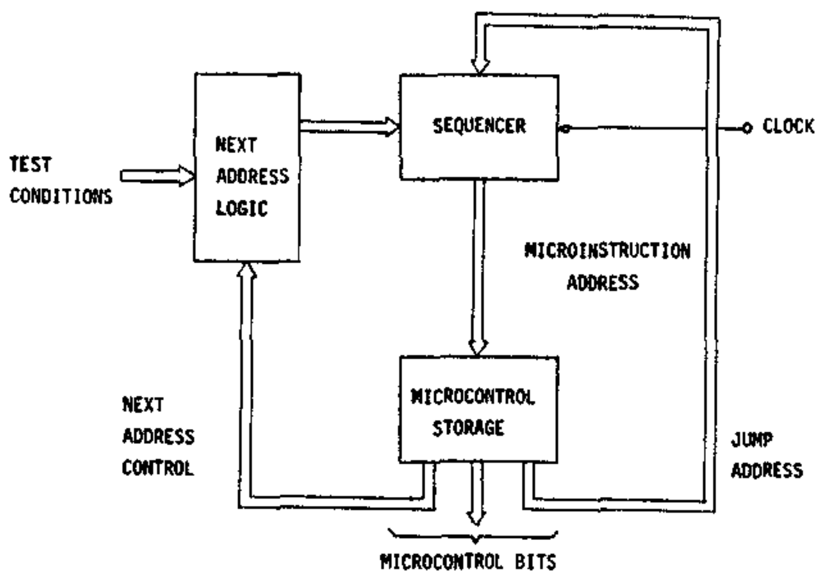

Fig. 1 - General structure of microprogrammed logic

The sequencer gives the microinstruction address to be executed at current cycle, which maps the microcontrol storage and furnishes microcontrol bits. The microcontrol storage contains the microprogram, coded at 1's and 0's. Next address logic, under microprogram control and combined with test conditions, gives information about next microinstruction address to sequencer.

The advantage of using microprogrammed logic instead of wired logic results in a more organized and flexible structure, obtainable with the first option. Many design alterations may be easily carried out just changing the microprogram contents.

After development phase, the microprogram is recorded into nonvolatile memory, such as a programmable read only memory.

However, a problem arises during the microprogram test phase, where loading the microinstructions into the microcontrol storage is needed. After this step, if the microprogram needs to be changed, then new PROM's will have to be programmed, which may be prohibitive.

Therefore, a microprogram debugging tool, which doesn't compromise the PROM burning and incorporates the major features of a simulator program, is highly desirable.

2. MICROPROGRAM DEBUGGING TOOL

The most commonly used tools for microprogram tests are:

1) Simulator programs running in a host computer;

2) Hardware emulators.

The advantage of simulator program over the hardware emulator is that facilities such as those for execution control, microinstruction sequencing and verification of register contents can be easily implemented.

On the other hand, a simulator program presents limitations when more complex LSI integrated circuits are employed in the system, the need for a special new program for each new system to be simulated, besides the simulation of the microprogram logic does not validate, necessarily, a real time simulation.

The outstanding characteristic of the emulation approach is the real time testing. In this case, the system under test is subjected to more realistic conditions, such as gates delays, races, fan-in, fan-out and others.

An usefull tool for microprogram validation is a randon access memory based emulator, that is, a device that replaces the microcontrol memory during microprogram debugging, while permitting microinstruction changes.

A somewhat primitive tool which permits the emulator realization is an alterable programmable memory module, made of EPROM's or EAROM's devices. It has inconveniences such as that of memory reprograming, which is not very practical. Furthermore, the system under testing may be forced to run at a lower clock rate, due to the slow access time of this type of memory.

An advantage of the RAM based emulation approach is that it eliminates the inconveniences of the EPROM/EAROM alternative. The complexity of RAM based solution is increasead over the other one, due to additional resources which are needed to manipulate the volatile memory.

A smart emulator can be made using a computer to support the machine-man interaction, therefore reducing the work load with the emulator.

The emulation model developed at INPE (Instituto de Pesqui_ sas Espaciais) supports a microprogram debugging tool and may also be employed in the maintenance and fault diagnosis of microprogrammed equipments.

Maintenance and fault diagnosis of microprogrammed equipments are very interesting features which aid the equipment troubleshoots.

The EMMAC (Emulador de Memórias de Microcontrole Auxiliado por Computador) is made of two major parts:

1) An Emulator System which implements physical emulation;
2) An attached computer resident Monitor Program which provides resources for manipulating the Emulator System.

A block diagram of EMMAC Emulator System and its configuration with a host computer may be seen at Figures 2 and 3, respectively.

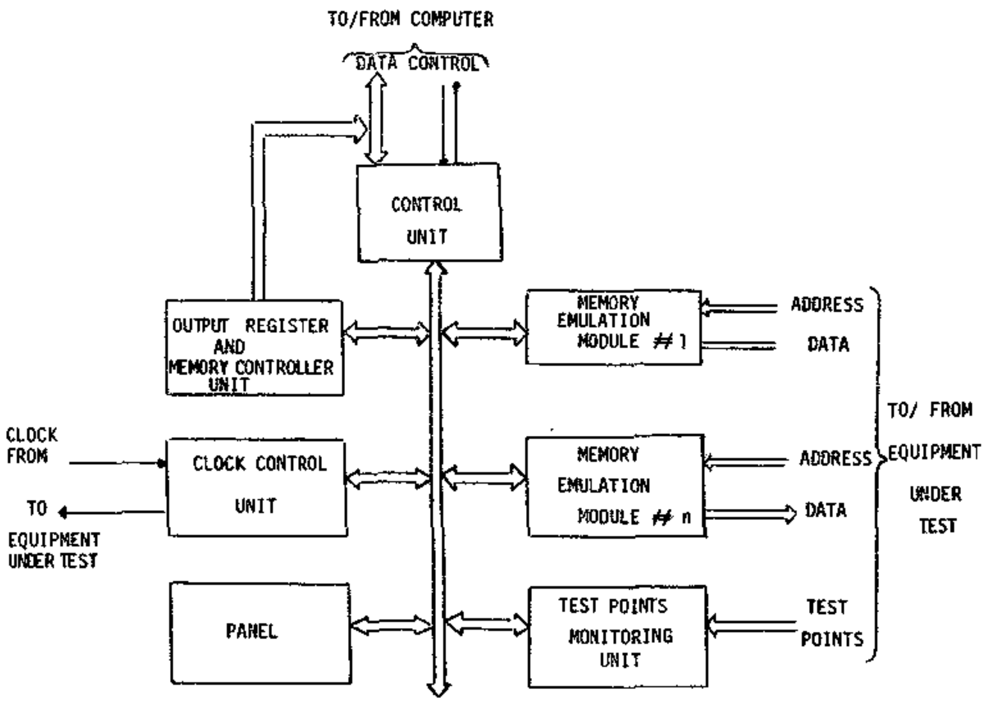
TO/FROM COMPUTER
DATA
CONTROL
CONTROL UNIT
OUTPUT REGISTER AND MEMORY CONTROLLER UNIT
MEMORY EMULATION MODULE #1
ADDRESS
DATA
CLOCK FROM
TO
EQUIPMENT UNDER TEST
CLOCK CONTROL UNIT
MEMORY EMULATION MODULE # n
ADDRESS
DATA
TO/ FROM EQUIPMENT UNDER TEST
PANEL
TEST POINTS MONITORING UNIT
TEST POINTS

Fig. 2 - Schematic diagram of the Emulator System

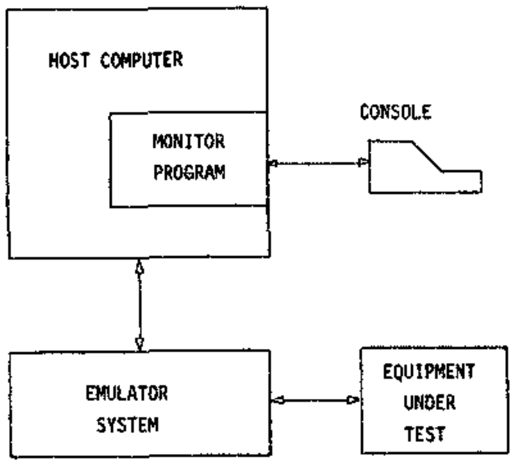

Fig. 3 - EMMAC's system configuration

3. HARDWARE OF THE EMMAC

The Emulator System may be implemented with a variable number of random access memory emulation modules. The limit number would be 2^{12} modules which is too large for practical needs. In fact, the most common microcontrol memories have up to 160 bits in word lenght wiht up to 512 microinstruction words in depth.

EMMAC has a set of memory modules that are shared by the Emulator System and the equipment under test, althought with exclusive access, that is, only one system can address the memories, at a time.

The access of the memories is controlled by the Emulator, which can perform read/write operations, while the equipment under test can do only read operations.

With this approach, EMMAC has capability of loading and altering the emulation modules, besides reading over them. For the equipment under test, the emulation modules have a performance similar to that expected by the PROM memories.

Since each memory emulation module has its own address bus, it can be used not only for microcontrol storage emulation, but it can also emulate PROM memory with different functions such as mapping memory, constant table and others.

The only restriction related to the allocation of memory modules is that the module number one must be used as microcontrol memory in the case of microprogram storage emulation, because its address bus is seen as microinstruction address when the system is operating in conditional mode.

EMMAC can operate in four distinct emulation modes:

1) Step by step;
2) Conditionally, until a given microinstruction address has been reached;
3) Conditionally, until a given number of clock pulses have been counted;
4) Free running mode.

The operating mode is selected by user through Monitor Program, during the preparation phase of the emulation section. After test conditions have been set up, the emulation will start with a monitor command and will run with the time base of the equipment under test clock. The execution of emulation can be interrupted by the operator at any time.

Besides the cited features, the Emulator System provides means of monitoring up to 32 test points defined by the user over the equipment under test. The sampling rate of these test points can be made at the same rate or at a submultiple rate of the equipment clock,programable through the host computer, which monitors the system.

The control unit itself was microprogrammed, and controls all the Emulator System. It receives instructions from the computer and executes them sequentially.

Each instruction cycle is initiated when the computer sends an instruction word to signal it. After the execution cycle, the control unit signals the computer indicating that the current instruction has been completed.

The instruction set which can be executed by EMMAC includes read/write into memory emulation module, read of test points and control instructions. These control instructions are used to specify emulation conditions, such as the operating mode, test condition and test points sampling rate.

Finally, the panel provides an alternative way to control the Emulator System and replaces the computer when EMMAC is at local mode. In this mode, instructions can be entered in the Emulator System with the panel displaying the results. The EMMAC panel is eventually used only for system maintenance.

4. SOFTWARE OF THE EMMAC

The software that controls the Emulator System is called Monitor Program and is composed by a command set, which aids the user during the operation of the system.

The commands have not the same interpretation of a common programming language, like ALGOL or FORTRAN.

Each command of the Monitor is analyzed and executed in an interpretative manner, and the execution of the commands is independent among them.

The command set is divided into five classes, according to the function to be carried out, namely: memory, execution, monitoring, operating mode and control.

The memory class is composed by "ARMAZENE" (Store), "COPIE" (Copy), "TROQUE" (Exchange), "MUDE" IChange), "ESCREVA" (Write) and "LEIA" (Read) commands. Execution of all these commands implies access the memory modules and the use of the peripheral equipments for data input/output.

The execution class is directly related to the execution of another command and includes the "PARE" (Stop) and "SIGA" (Go)commands, besides the "EXECUTE" (Execute) command for test running.

The monitoring class consists of two commands: "MONITORE" (Monitor) and "DIAGNOSE" (Diagnosis). "MONITORE" is used to acquire data from test points, while "DIAGNOSE" compares data previously stored from "MONITORE" command with new data obtained from the test points.

"RELÕGIO" (Clock), "INICIE" (Initialize) and "FREQUÊNCIA" (Frequency) commands are available at operating mode class. "RELÕGIO" is used to select the operating mode of the Emulator System (step by step, free running, etc). The "FREQUÊNCIA" command determines the monitoring sampling rate, while "INICIE" command sets initial conditions for the beginning of the emulation during a given number of clock pulses.

Finally, the control class has the following commands: "ABORTE" (Abort), "LISTE" (List), "TERMINAL" (Terminal), "PFITA"(Magnetic Tape positioning) and "FIM" (Exit). "ABORTE" command is used to terminate a running command such as "EXECUTE", while "FIM" terminates the monitor program and deviates the control back to the host computer operating system.

The "TERMINAL" command defines a listing terminal, and "LISTE" command is used to make the output of results into the peripheral determinated by "TERMINAL". "PFITA" chooses a better allocation for the files in the magnetic tape.

It is believed that, with this command set, it is possible to make a good automated use of peripheral resources of a host computer. However, new commands may be implemented for another specific computing system.

5. THE IMPLEMENTATION OF THE PROTOTYPE

The prototype of EMMAC was implemented with the aid of a Hewlett-Packard 21MX-E minicomputer, coupled to an HP Microcircuit Interface.

The Microcircuit is a 16 bit general purpose interface, with two lines for handshake with the peripheral equipment. These control lines can be programmed within an universal driver, supported by operating system of the minicomputer.

The peripherals available with the HP 21MX-E minicomputer of the Digital and Analog Systems Laboratory of INPE are:

1) Dual cartdrige disc (1 fixed + 1 removable);
2) System console;
3) Dual 800 BPI magnetic tape;
4) Paper punch and reader;
5) 200 LPM line printer;
6) Remote terminal.

The central processing unit is a 16 bit word HP2113B unit, with 64 K words of semiconductor memory.

The Monitor Program of EMMAC was written with HP ALGOL language, resulting in 14 K words of object code and utilizing the operating system's routines package. These routines were used for peripheral input/output and for control operations.

The prototype of EMMAC is running under the Real Time Executive - version IV A operating system in a multiprogramming environment.

The emulation test is performed by replacing the PROM's of equipment under test by EMMAC's memory emulation cables, and by connecting the tested equipment oscillator output to EMMAC's clock input. The Emulator System clock output is connected back to the tested equipment clock driver.

Through the computer console, the user can run the monitor program and define the emulation conditions for it. Afterwards the microprogram is loaded into the emulator memories from some available input peripheral.

When all test requirements have been set-up, the user will enter an "EXECUTE" command to start the emulation. Results can be listed or stored in any output terminal as previously defined, and new commands can be entered in an interactive operation's manner.

Diagnosis and maintenance can be performed by comparing the results of the test points of the equipment being troubleshooted with the data collected from an unfailing reference model of equipment being monitoried, or with data of expected results entered by the user. In this way, EMMAC aids to localize failures in the equipment under test.

6. CONCLUSIONS

The prototype of EMMAC was implemented with capability for emulating 512 x 128 bit microprograms. Presently, the emulating memories actually used with the Emulator System have a 450 nsec access time, but they will be replaced by a 30 nsec access time memories in the near future.

The results obtained with the prototype were very good, and it is believed that no problem will arise with the memories replacement.

The only restriction related to the prototype of EMMAC is the emulation memory module depth. However, another approach may be considered by treating the emulation memory module as a cache memory. With this approach, an additional hardware for page selection must be added and, thereby, we can implement large microprograms with minimal hardware expansion.

ACKNOWLEDGEMENTS

We thank Dr. Eduardo Whitaker Bergamini for his suggestions and encouragement.

BIBLIOGRAPHIC REFERENCES

AMARAL, P.F.S. *Emulador de memorias de microcontrole auxiliado por computador*. São Josē dos Campos, INPE, maio, 1979. (INPE-1489-TDL/009).

CRUZ, M.A.C. *EMMAC - sistema emulador*. São Josē dos Campos, INPE (a ser publicado).