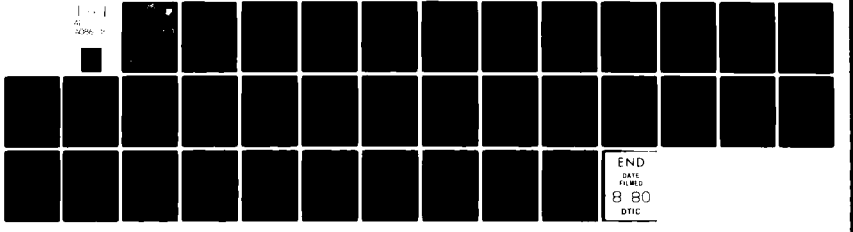
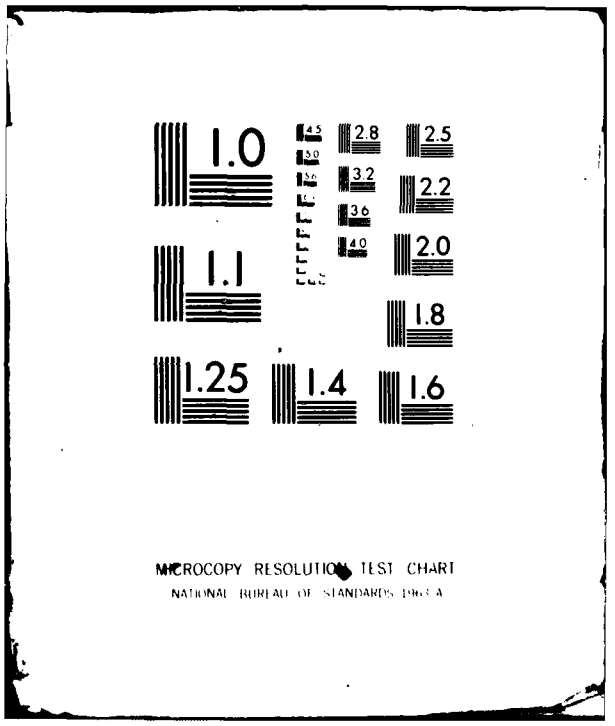
AD-A086 245 HARRIS CORP MELBOURNE FL GOVERNMENT COMMUNICATION SY--ETC F/G 9/2
MULTIPLE MICROPROCESSOR SYSTEM (MMS) DESIGN STUDY. VOLUME III.(U)
MAR 80 F30602-78-C-0114

UNCLASSIFIED RADC-TR-80-33-VOL-3 NL



END
DATE FILMED
8 80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

LEVEL III

AD A086245

RADC-TR-80-33, Vol III (of four)
Final Technical Report
March 1980

MULTIPLE MICROPROCESSOR SYSTEM (MMS) DESIGN STUDY

Harris Corporation

Government Communications Systems Division

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

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

RADC-TR-80-33, Volume III (of four) has been reviewed and is approved for publication.

APPROVED: Michael A. Troutman

MICHAEL A. TROUTMAN, 1Lt, USAF
Project Engineer

APPROVED: Wendall C. Bauman

WENDALL C. BAUMAN, Colonel, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISCA) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) TR-80-33-VOL-3

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: (18) RADC-TR-80-33, Vol III (of four)	2. GOVT ACCESSION NO.: AD-A086245
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): (6) MULTIPLE MICROPROCESSOR SYSTEM (MMS) DESIGN STUDY, Volume III.	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report
	6. PERFORMING ORG. REPORT NUMBER: N/A
7. AUTHOR(s): Harris Corporation, Government Communications Systems Division	8. CONTRACT OR GRANT NUMBER(s): (15) F30602-78-C-0114
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Harris Corporation, Government Communications Systems Division, P O Box 37, Melbourne FL 32901	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: (16) 63728F, 25290104 (17) 01
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (ISCA), Griffiss AFB NY 13441 (12) 34	12. REPORT DATE: March 1980
	13. NUMBER OF PAGES: 36
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same (9) Final Technical Rept.	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited. (11) May 80

411214

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: 1Lt Michael A. Troutman (ISCA)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

multiple microprocessors	microprocessor
emulation	distributed architecture
computer architecture	performance measurement
architecture evaluation	total system design

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This is a design for a Multiple Microprocessor System (MMS) to be built as part of the System Architecture Evaluation Facility (SAEF) being developed at RADC. The MMS was designed to be used in the modeling of a wide range of multiprocessor configurations for the purpose of evaluating their suitability to unique Air Force data processing requirements.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 411214

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

INTRODUCTION

This volume, Volume III, of the SAEF FINAL REPORT contains appendicies referenced in the MMS analysis and PMS discussion of Volume I. This volume also contains an appendix which discusses hardware changes that could be implemented to reduce hardware cost.

APPENDIX A

PERFORMANCE MONITORING SUBSYSTEM

In an effort to make the Performance Monitoring Subsystem (PMS) a clearly defined module this paper will be broken into 3 groups: goals/features; implementation; and a summary. The primary purpose for outlining the goals of the PMS is to put some reasonable bounds on the implementation. The summary is presented to help clarify the suggested method and illustrate how it meets the goals of the PMS.

1.0 PMS GOALS/FEATURES

The key feature of the PMS is a fully integrated monitoring of the functions of the target system (TS) that is being emulated and specified by the user.

1.1 PMS Features

Interactive debug

- single step
- trace
- traps
- breakpoints
- examine
- continue

Software and hardware evaluation and verification

- monitoring of I/O accesses
- monitoring of memory accesses
- monitoring of shared resource use
- monitoring of register accesses

- monitoring of instruction execution
- dumping of contents of
  - registers (includes the PC)
  - memory locations
- dumping data and memory addresses associated with a particular instruction
- traps that allow control to be returned to the user for intervention such as modification to the monitoring capability
- dynamic changes in monitoring

1.2 User Interaction

The two primary users on which the MMS relies are the microcode writer and the TS user. In general these users are not the same person. Since it seems unlikely that both users can be alleviated of any PMS burden some reasonable limit needs to exist. The following goals have been set on what the duties of these users should be:

- the microcode writer cannot be expected to be aware of what the final TS user will want to monitor and therefore will not be burdened with additional coding for each line of emulation microcode that is written.
- the microcoder is, however, aware of the fundamental workings of the microcode and can be expected to keep up with the placement of registers, instructions, etc.

- the TS user is responsible for deciding which instructions, registers, etc. should be monitored.
- the TS user specifies points of interest before or during the run, but does not insert any hooks or in any way modify the microcode.

1.3 Overhead

Overhead is the affect on emulation efficiency caused by halts in pseudo-time. Since emulation efficiency is the ratio of pseudo-time to wall time, any operation that causes pseudo-time to lag behind wall time will cause a degradation in the efficiency of the emulated system. When a processor is checking PMS events or dumping data it must halt its pseudo-time causing that processor to fall behind and possibly cause a system halt. All of the overhead caused by performance monitoring cannot be eliminated, but PMS overhead should be proportional to the amount of performance data that is being collected. The following are the goals placed on the overhead caused by the PMS:

- with no PMS data collection (PMS turned off) there will be no overhead.
- when PMS is turned on there will be no additional time required to decide if an item is of interest.
- with PMS turned on and item of interest detected the overhead concerned with dumping

of the data will be proportional to the amount of data being collected.

2.0 PMS Implementation

This major paragraph concerns itself with the actual implementation of the PMS. The major categories that it breaks down into are:

- the manner in which the microcode will be written
- hardware requirements
- monitoring that will be handled outside the PE
- overhead
- user interaction

2.1 Microcode Structure

The implementation of PMS event detection in a PE is closely tied with the instruction decoding method chosen for the PE. The decoding process will be highlighted here for explanatory purposes, and is discussed in further detail in the PE conceptual design description contained in the main body of the report.

Instruction decoding is implemented within the PE with a CASE statement in a manner similar to that used in the SMITE HDL. The unique feature of the MMS implementation is that the CASE statement is implemented in a single (possibly two) micro instruction. The hardware required for this implementation consists of a high speed shift and mask unit to extract a bit field from the instruction, a concatenation unit to append additional bits to the selected bit field, and

an internal memory (IM) addressed by the bits obtained from the concatentation unit. For instruction decoding the IM will contain the address in control store for each microcode routine callable from the CASE statement. This address will be routed to the control store sequencer for execution during the next micro instruction. A flow diagram of this hardware process is shown in Figure 1. Note that a unique IM location is used for each step of the instruction decoding (op code cracking) process. The IM contains the starting address of a control store routine to execute each decoded action (i.e., instruction). Additional width will be added to the IM for PMS purposes, as described later. This will take advantage of the fact that a unique IM location is used for each decoded instruction.

Decoding of register select fields within a micro-instruction is done in a manner similar to that used for instruction decode. A flow diagram of this process is shown in Figure 2. In this case IM is used to emulate target system CPU registers so that data out from the IM is the contents of an emulated CPU register. This data is routed to the ALU or other destinations as needed in the instruction execution process. Note again that a unique IM location is used for each target system register that is being emulated including the program counter register.

The IM within the PE is also used for an additional purpose. The PE executive (PEXE) provides the link between instruction emulation code and I/O emulation code, and also between I/O emulation code and micro interrupts such as hardware

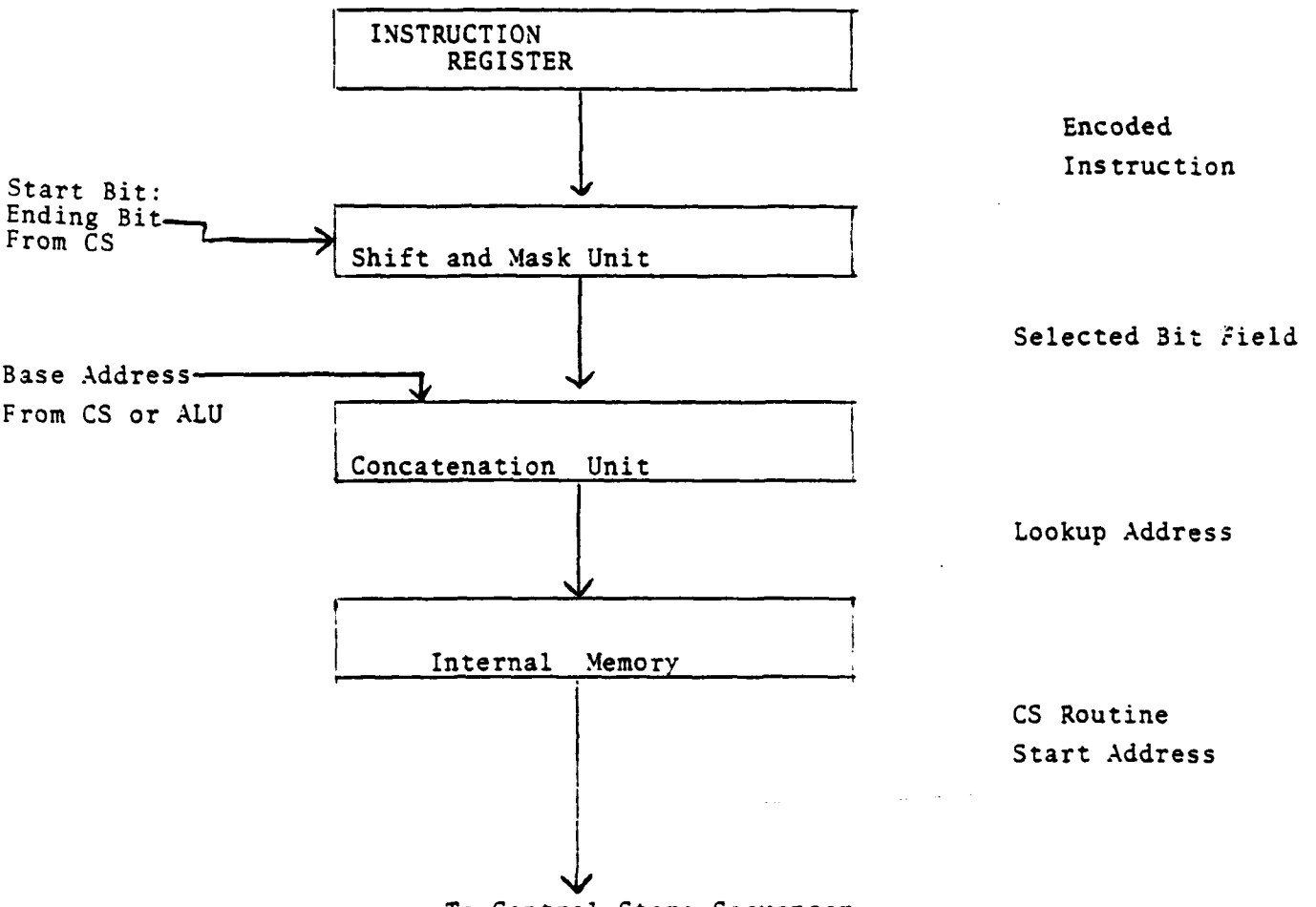


Figure 1 -- Hardware Flow Diagram

Instruction Decode

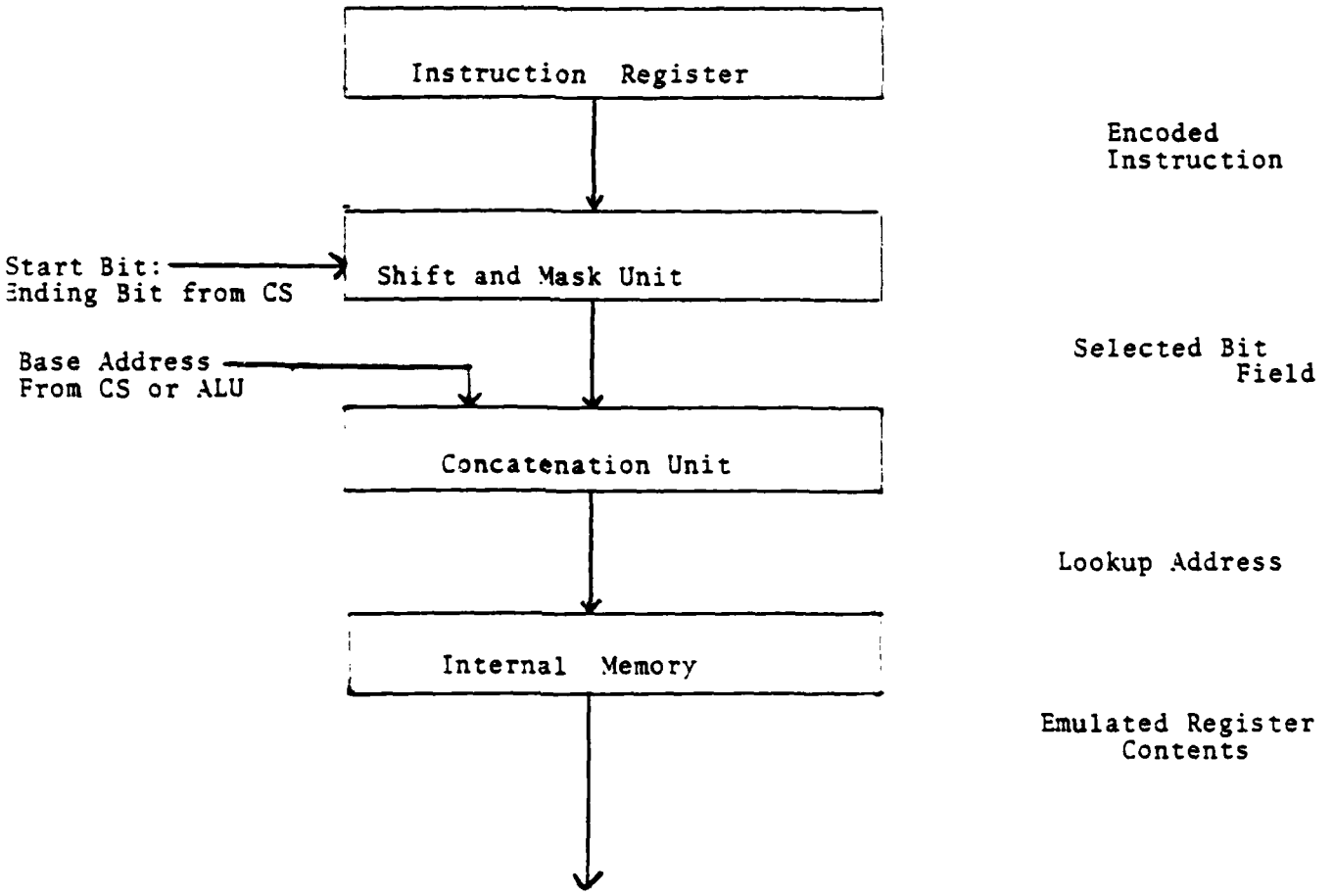


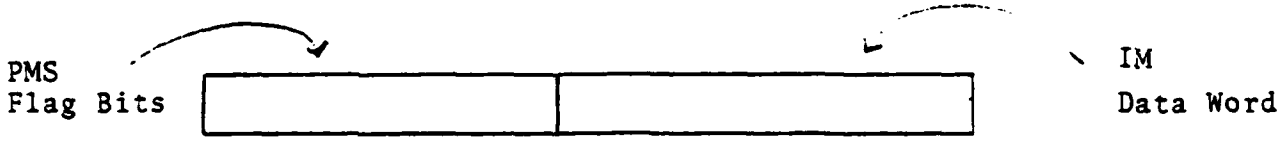
Figure 2 - Hardware Flow Diagram

Register Field Decode

timers and IPC's. PEXE provides these links by routing of I/O access and micro interrupts to appropriate entry points within I/O emulation code. This routing will be performed by table lookup within PEXE. The table lookup will be performed through the IM/ In this case the lookup address will be provided by the ALU and the IM will contain entry points to I/O emulation code which will be routed to the control store sequencer for execution. Note again that a unique internal memory location is used for each entry point into I/O emulation code. These entry points correspond to: I/O access by the emulated target system; micro interrupt requests by an I/O device; micro interrupt grants, and similar I/O activity.

2.2 Hardware Requirements

Some additional bits will be added to the IM to turn PMS on or off. The additional hardware is primarily a small group of "flag bits" that indicate such things as "Is PMS being monitored for this item?"; "Should any resulting data be dumped?" or "Should the PC be collected?". At this point the number of these bits is not determined, see example below for clarity.



The ability to do this type of performance monitoring requires the additional bits in the IM word and thus requires additional hardware. The benefits to PMS that can be achieved using this method (which will be discussed shortly) are sub-

stantial reasons for the additions.

The actual PMS "checking" will cause no overhead because the address that is accessed in the IM will be read whether PMS is activated or not. If the "PMS collection flag" is set an interrupt to PEXE occurs and PEXE executes the dumping of the associated event code. If associated data is to be dumped that is handled by PEXE also. The overhead for doing these things is small. The actual "check" for PMS enabled is done in hardware and does not require any routing to PEXE.

2.3 PMS Detection Handled by the Memory Interface and Shared Resource Controller

Two particular kinds of monitoring will be handled differently. The most abcious monitoring point that has not been discussed is memory access, this will be handled in the memory interface (MI). When a memory access is sent to the MI a check is done to decide which block of memory is being referenced. The decision is made in the manner "is the referenced location greater than or equal to memory location X and less than or equal to memory location Y?" All of memory will be broken into segments of locations such as "A to B", "C to D", etc. The user will decide the starting and ending locations of each block and if the two locations are set equal then the block represents a particular location. Along with the beginning and ending addresses of each block are also a group of PMS flag bits just as they were in the IM of the PE. One bit indicates that accesses to the memory block (or location) are important to the PMS. If this bit is set an interrupt to PEXE occurs and the rest of the operation is handled as it was for

monitoring done in the IM.

The monitoring of shared resources is handled in the shared resource controller (SRC). The SRC is a PE itself with an IM as in the cases mentioned earlier. The arbitration schemes are implemented in microcode and reside in the control store of this PE. The routing addresses to these arbitration schemes are located in the IM. Additional bits at each IM location act as PMS flags. They are checked in the same manner as in the other PE's. The PE used as the SRC has its own PEXE which is interrupted when a monitoring point is encountered. The entire operation is identical to the one for all the other PE's.

2.4 Overhead

In this implementation, there is no overhead concerned with the "checking" for a PMS point of interest. A flag is set in the IM that is checked by hardware when that location is read. If the flag is not set no degradation occurs in emulation efficiency. This is a very important point because it implies that PMS can be activated and assuming none of the PMS events occur there is no degradation of emulation efficiency. This is appropriate in many cases when the user is only concerned with specific events and not all occurrences during the run. For example, the user wants to monitor "out port 7" but not all "out port" instructions. The checks are invisible to the user, so PMS does not effect the run unless one of these events occur, at which point the user is willing to give up the time for that collection.

There is some overhead involved in the dumping of "associated information", such as the contents of a memory location, the contents of the PC, a status dump, or a trace of the execution of that instruction. The last item mentioned here needs more explanation. The trace may involve only a few "special" instructions. For example, when a "test and set" instruction is executed the data and memory accesses involved in that instruction are important. One PMS flag is set aside to indicate a need to dump all memory and register accesses with their contents and any other actions that occurred during the execution of a particular instruction. This trace is limited to the steps required to execute one particular instruction. The user is not burdened with dynamic changes in collection in order to get a partial trace, nor does the user have to decide what is needed to retrieve the necessary information.

2.5 User Interaction

The user is broken into two groups, the microcode writer and the TS user. The TS user "builds" a system with the previously written microcode modules. Since the TS user knows what needs to be monitored none of this decision burden should be placed on the microcode writer.

In this implementation the microcode writer keeps a table of the locations in the IM where registers, instructions, etc., reside. Since the microcoder made the decision as to where the routing address would reside in IM, this bookkeeping is a reasonable chore for him to perform. This table, which is necessary for each microcode module, serves as documentation

for debug and modification purposes and as the reference table for the PMS processor to use when locating events for monitoring.

The TS user follows a different scenario in support of the PMS. The TS user indicates which instructions, register accesses, etc., are important and then specifies them with the interactive PMS definition language. The PMS processor takes the list of events to be monitored, finds these events in IM and sets the appropriate PMS flags.

3.0 Summary of PMS Implementation

The purpose of this section is to give a concise look at the implementation suggested here and to show how the features and goals are fulfilled. The PMS that has been suggested is a part of the IM of each PE including the SRC and also a part of the MI. All PMS monitoring is done in hardware while the decisions concerning what data should be collected and the actual data dumping are handled in PEXE, i.e., software. The microcode contains no modifications for PMS and the microcode writer never has to know if PMS will be used. The TS user indicates events of interest for the PMS but never deals with the microcode or the hardware.

All of the PMS monitoring features are handled using this technique. I/O monitoring, instruction monitoring, and register access monitoring are handled in the PE, shared resource monitoring is handled in the SRC, and memory access monitoring is done in the MI. When a PMS monitoring point is detected an interrupt to PEXE will occur for the actual dumping of data. This is the only time the PE will have to stop its

pseudo-time, i.e. the dumping of data will be the only reason foe efficiency degradation. When the interrupt occurs PEXE will check all of the PMS flags and decide what data to dump. Typically this data will be readily available at the time of the interrupt.

Take the example of a regsiter access being monitored and the associated data being collected is the contents of the register and the PC. When the register is accessed the code indicating this will be available in the IM in one of the following ways; a separate group of bits along with the PMS flags or just the address in IM. The contents of the register is available in the IM and the PC is easily retrieved out of the IM also. The ease of finding the data contrasts to searching lists of "possible data". The amount of time necessary for the data dump will depend on how many pieces of data are requested.

A trap represents another interrupt to PEXE which allows the user to stop the system, modify it or collect data, and restart without the initialization burden. The user could enter the debug mode and single step or trace the next few operations in an effort ot clearly view system operation.

The trap is a reasonable place for the user to re-direct the data collection. This is the TS user who had to make all the monitoring decisions before the run. Although several events and/or times could be set up as triggers for dynamic changes of data collection, the TS user had to decide

what new data to collect or to stop collecting before the run. Since this user is not familiar with the microcode or the MMS hardware, he can use the "trap" feature to allow him to look at various registers or memory locations and make a better decision as to which events are important to the PMS.

This implementation provides complete monitoring capabilities by allowing the TS user to monitor any event with no limitation on the number of events. This contrasts to allowing the user to set a specified number of flags (a register) each representing a particular event (monitor word). Although the idea is virtually the same number of flags has grown from 64 or even 128 to a number that is equal to all possible events.

Since all of the monitoring features are covered within the PE, the SRC, or the MI, there is no need for the microcoder to write any additional code for PMS implementation. The PMS processor used the microcoder's documentation to provide the TS user monitoring capability within the TS. The TS user will see fluctuations in the emulation efficiency according to the amount of performance data that is collected. Monitoring points that are not accessed will not show up in emulation efficiency. This contrasts to using software for event detection. The overhead is almost constant with software detection because an interrupt to the executive must occur each time a PE operation takes place. After the interrupt, a list of events being monitored must be compared with the event that just occurred. The

problem with overhead occurs when the lists contain events that seldom happen but must be checked when any event occurs.

The PMS implementation discussed here provides the features stated in section 1.0. Overhead is proportional to the amount of performance data being collected. The microcoder must handle documentation but does not have to write additional code or decide what will be important for the PMS. The TS user does not need familiarity with the microcode or the hardware. The PMS is flexible in that all events can be monitored, the monitoring can be changed during a run, and associated data can be collected with any event that occurs.

APPENDIX B

DEFINITION OF PROCESSOR IN ANALYSIS

A processor is defined according to the following parameters:

NIPS - Average number of instructions per second executed by the machine.

BS - Bit size, width of data.

MS - The total memory space that a processor addresses for a specific application.

IIO - The frequency of I/O per instruction.

IIPC - The frequency at which interprocessor communications are performed per instruction.

IMA - The average number of memory accesses per instruction.

ESIMP - The average number of micro cycles necessary to emulate the decode, address calculation, and arithmetic given the rudimentary PE parameters.

IOWF - The portion of I/O that is local.

MAWF - The portion of memory that is local.

ISMP - Variables which indicate whether the processor has shared memory or not.

IIOC - The portion of I/O that can be handled by the IOC alone.

IMMIO - The portion of I/O that is handled by block transfer from a disk to memory and then by memory mapping in the PE.

APPENDIX C

PARAMETER ESTIMATIONS

PARAMETER	DISTRIBUTION	MEAN	TWO STD DEV	RANGE LOW	RANGE HIGH
NP	Normal	25	15	1	64
MS	Normal	64K	30K	500	48MBytes
BS	Discrete	P(8)=.3 P(16)=.6 P(32)=.09 P(64)=.01			
NIPS	4 Normal				
	BS=8	350K	200K	100K	600K
	BS=16	600K	400K	200K	1400K
	BS=32	1500K	400K	800K	2500K
	BS=64	1500K	400K	800K	2500K
IIPC	Log Normal	10^{-3}	Factor of Ten	10^{-5}	10^{-1}
	(Log_{10} Values)	-3	1	-5	-1
IIO	Log Normal	10^{-2}	Factor of Ten	10^{-4}	10^{0}
	(Log_{10} Normal)	-2	1	-4	0
IMA	Discrete				
	BS=8	2.68			
	BS=16	2.02			
	BS=32	1.36			
	BS=64	1.36			
PUE	Discrete	1			
IOWF	Discrete	0.95			
ISMP	Discrete	0-No Shared Memory 1-Shared Memory Processor			
MAWF	Binomial Approx.	0.9	0.1	0	1

APPENDIX D

FUNDAMENTAL CALCULATIONS

MEMORY TRANSFER RATE

$$MTR = PUE \times \sum_{i=1}^{NP} IMA_i \times NIPS_i$$

SHARED RESOURCE REQUEST RATE

$$SRRR - PUE \times \sum_{i=1}^{NP} ISRR_i \times NIPS_i$$

$$ISRR_i = IIPC_i + IIO_i (1 - IOWF) + IMA_i (ISMP_i) (1 - MAWF_i)$$

INPUT/OUTPUT TRANSFER RATE

$$IOTR = PUE \times \sum_{i=1}^{NP} IIO_i \times NIPS_i$$

INTERPROCESSOR MESSAGE RATE

$$IMR = PUE \times \sum_{i=1}^{NP} IIPC_i \times NIPS_i$$

APPENDIX E

DEFINITION OF AN MMS CONFIGURATION

The following parameters specify MMS:

MCT - The cycle time associated with sequencing through microinstructions.

ALUWD - The number of bits wide the ALU and associated registers in the IEU are.

BWD - The width of bits of the system data bus and the width of the memory.

LMS - The size in bits of the local memory associated with each PE.

MCT - The cycle time available in the memory itself.

MI - The delay time needed in the memory interface to translate and output the correct system address.

SCRMAX - The average maximum rate at which the SRC can process shared resource requests.

IOPMAX - The average maximum rate at which the IOP can process shared I/O requests and environmental data calculations.

BTRMAX - The maximum rate at which the bus may be run.

APPENDIX F

MMS CONFIGURATIONS

CT	ALUWD	BWD	LMS	MCT	MI	SRCMAX	IOPMAX	BTRMAX		
200ns	16	16	64K X 16 1,048,576	700ns	700ns	300K	10K	10M	100ns	
200ns	16	16	64K X 16	200ns	300ns	300K	10K	10m	100ns	A
100ns	16	16	64K X 16	700ns	300ns	300K	10K	10m	100ns	B
200ns	16	32	32K X 32	700ns	300ns	300K	10K	10m	100ns	C
200ns	16	16	64K X 16	700ns	300ns	300K	10K	5m	200ns	D
200ns	32	32	32K X 32	700ns	300ns	300K	1K	5m	200ns	E
200ns	16	32	32K X 32	700ns	300ns	300K	1K	5m	200ns	F
200ns	16	16	64K X 16	700ns	300ns	300K	1K	5m	200ns	G
200ns	8	16	64K X 16	700ns	300ns	300K	1K	5m	200ns	H
200ns	8	8	128K X 8	700ns	300ns	300K	1K	5m	200ns	I

APPENDIX G

GLOSSARY OF ACRONYMS AND ABBREVIATIONS

BC	Broadcast Controller
BCC	Broadcast Communication
BIU	Bus Interface Unit
CPU	Central Processing Unit
CS	Control Store
EE	Emulation Engine
EES	Emulation Engine Support
ES	Environmental Simulation
ESMAU	Emulated Shared Memory Arbitration Unit
FCP	Facilities Control Processor
GASP IV	General Activity Simualtion Program - Version IV
HOL	Higher Order Language
IEU	Instruction Execution Unit
I/O	Input/Output
IOC	Input/Output Controller
IOP	Input/Output Processor
IPC	Interprocessor Communications
IM	Internal Memory
LPA	Local Pseudo-time Accumulator
MI	Memory Interface
MMS	Multimicroprocessor System
MPT	Master Pseudo-time
OCS	Operating and Control System
PE	Processing Element
PEXE	Processing Element Executive
PMS	Performance Monitor Processor
PMS	Performance Monitor System
PMSI	Performance Monitoring System Interface
PRICE	Name for RCA's parameter cost-modeling system for computer hardware and software cost estimates
PRIM	Programming Research Instrument
PT	Pseudotime
RALU	Register, Arithmetic and Logic Unit
SAEF	System Architecture Evaluation Facility
SBS	Synchronous Busing Structure
SMITE	Software Machine Implementation Tool Using Emulation
SRC	Shared Resource Controller
TAC	Time Alignment Controller
TS	Target System
TSD	Total System Design

Appendix H

1.0 MMS DESIGN MODIFICATIONS FOR COST REDUCTIONS

This section describes hardware design modification of the MMS for the purpose of hardware cost reduction. The changes described in this section do not apply to the MMS hardware specification document.

1.1 Redesign Goals and Their Effects on the MMS

The primary goal of the MMS redesign is to reduce the hardware cost of the MMS. One way to reduce hardware costs is to reduce the number of dips required in each component of the MMS. The major sacrifice to be made by reducing hardware is emulation performance.

The redesign of MMS is based on the following 3 constraints:

- Sacrifice of emulation performance should result in minimal degradation of the flexibility and ease of use of the MMS.
- Removal or reduction of all MMS components that have the sole purpose of enhancing emulation performance.
- Set the upper limit for the dip count of the PE to 300 dips.

The primary components affected by the MMS redesign are:

- The PE
- The bussing structure
- The FCP interfaces to the MMS

In the following sections is a detailed discussion of the redesign of the various components and the overall affect on the MMS is provided.

1.2 Design Changes

This section describes changes in the bussing structure, modifications to the FCP interfaces, and the complete redesign of the PE. Also provided are discussions on E-code, PMS, and the microcode writer.

1.2.1 Modification of the Processing Element

The major goal of redesigning the PE is to reduce the dip count of the PE from 820 dips to approximately 300 dips. The achievement of this goal is possible, but only with vast changes in the design concept of the PE and considerable degradation of emulation performance.

The PE encompasses several components which are affected by the redesign. These components and the design changes required are as follows:

- The IEU will be directed toward a more vertically microcoded machine, but will still be approximately 64 bits wide.
- Control store will no longer be down laodable.
- The user will write emulators in E-code instead of microcode.

- The local memory will be under explicit control of the PE.
- The BIU will only contain a single receive port.
- The functions of the MI and PMS will be performed in microcode.

1.2.1.1 E-Code Languages

The major changes in the PE will involve the shift from hardware to firmware. The E-code language will represent the crux of the change. E-code will be the only language to be emulated by the microcode in the PE. A target system (TS) emulator writer will not have the capability to modify the microcode written by the vendor for emulation of E-code.

E-code will encompass such instructions as:

- Decode - Extracts arbitrarily sized field from TS instruction and branches accordingly.
- Translate - Finds address in TS address space and branches to code that handles the access of this address.
- Broadcast - Forces a 16-bit word to be sent to an arbitrary number of other PE's in the system.
- Talk - Sends an IPC to a specific PE in the MMS
- Arith - Executes one of sixteen arithmetic operations on any two registers in the system.
- Shift - Executes one of thirty different shift operations on any register in the system.
- Fetch I - Fetches a new TS instruction.
- Fetch R - Fetches a TS register.

- TS Codes - Transfers the specified condition codes into the TS condition code register.

E-Code will also have definition type literals that define the TSP. (i.e. Bit size = 16, Address byte type).

1.2.1.1.1 Effects of E-code on the MMS

The hardware emulation of exactly one language (E-Code) will decrease the amount of hardware necessary in the PE. All instruction decode logic, for example, will be specific to the E-code format which will be designed very rigidly to reduce the amount of decode necessary. The complexity of the microword and the bulk of microcode written for emulation will be reduced. This will decrease the width and length of control store needed. It will be important to spend the time required to make the microcode structured and efficient since the microcode will only be written once.

Loss of flexibility is the negative aspect of using E-code. By using a specific language the emulation writer must follow a more rigid structure as defined by the choosen language. This may result in the E-code writer loosing the freedom of implementing a process or function in the exact manner desired. The amount of flexibility lost by the emulation writer is completely dependent on how well the E-code package is designed.

A possible advantage of E-code is an increase in the ease of use of the PE for the emulation writer. By judiciously choosing the E-code instruction set, the emulation writer may require considerably less time to write an E-code emulation package than a microcode emulation package.

1.2.1.2 Local Memory Modifications

The local memory (LM) will no longer be a completely independent unit. All LM accesses will be made through the IEU and under explicit control of the IEU. This concept will eliminate the need for the arbitration logic between the LM, IEU, and BIU and also eliminate additional hardware to control read-modify-write accesses to the LM.

In addition to being used for storage of the user's application code, the LM will also store the E-code package developed by the emulation writer. By multitasking the LM the Internal Memory (IM) will no longer be required. By utilizing the LM in this manner performance will decline significantly. This decline will be due to memory cycle time of the LM (700ns) being significantly longer than the memory cycle time of control store (200ns).

1.2.1.3 Microcode Modifications

The microcode package which must be written by the vendor will consist of the E-code emulation package and PEXE. Since this microcode package will be unmodifiable by the user the microcode will be storied in roms in each PE. The only access the user and E-code writer will have into PEXE will be through MMS interrupts and calls to PEXE from E-code.

By making the microcode package firmware significant cost reduction can be achieved. The use of Roms allows for the control store to be designed using fewer and cheaper parts.

Since the emulation writer will no longer be required to write microcode, control store down load hardware along with the microcode debug package and associated debug hardware will no longer be required.

1.2.1.4 PMS Modifications

The PMS of the PE will be handled by firmware rather than with hardware as in the original design. PMS will'key' on certain E-code instructions to determine when monitoring must be carried out. For instance, only when the FETCHER, or STORER instruction is executed will a check be made for a register access. Although the check is performed at each register access and additional time is spent by the microcode to do this check, the PMS monitoring will still be transparent to the emulation writer. Therefore at the expense of performance, another cost saving will be realized without loosing the PMS transparency ideal. PMS checks for memory access, op code execution, I/O access, or any shared access will likewise be keyed on their own specific instructions.

1.2.1.5 MI Modifications

The MI as a separate hardware entity will be eliminated. The functions performed by the hardware will be readily picked up by firmware.

The memory translation function performed by the MI is potentially a time consuming process that must be frequently executed. For this reason the moving of the MI from hardware to firmware can be expected to produce a significant reduction in the performance of the PE.

1.2.1.6 Effects of the PE Redesign on Other MMS Components

In the effort to reduce the dip count of the entire PE, dip count reduction must also be performed on the BIU. The The major change in the BIU is the removal of the multiple receive ports for IPC's, memory communications, and broadcast communications. These 3 ports can be replaced by a single general purpose port.

The major effect of this modification on the MMS is the removal of the broadcast mode as currently defined in the MMS hardware design. With the removal of the broadcast port in the BIU the segment broadcast controllers are no longer required.

Some additional BIU component reduction is obtained from the reduction of performance of the PE. The reduction of performance results in a lower bus speed requirement which in-turn simplifies the control circuitry in the BIU.

The effects of the PE redesign on the components mentioned above are discussed in more detail in the following sections.

1.2.2 Modifications of the MMS Bus Structure

The reduction of performance in the PE directly relates to hardware cost reductions in the bussing structure. Since the time between bus accesses for each PE will become longer, the bus speed can be reduced. The new bus transfer rate is estimated to be about 1.5 to 2.0 mega transfers per second.

The slowing of the bus allows for the bus cycle time to be increased; which allows for modification of the bus protocol. Instead of requiring 2 bus cycles to complete the handshaking process, a longer bus cycle allows for all the handshaking to be completed in a single bus cycle.

The modification of the bus speed and protocol results in hardware reduction in both the BIU and the Segment Bus Controller.

In order to reduce the total system cost the multi-segment design of the MMS can be replaced by a single segment design. However, by going to a single segment design the max-

imum number of PE's must be reduced from 217 to 32. Additional PE's could possibly be added in the future but only at the expense of designing the Inter-Segment Bus Interface Unit and some possible bus modifications.

1.2.3 Modifications of Broadcast Communications

The removal of the broadcast port from the BIU renders the segment broadcast controller useless. However, broadcast communications can still be performed with reduced efficiency.

One possible senario for broadcast communications is as follows. A PE needing to send a broadcast communications could stop the TAC; a message to the desired PE's, and release the TAC. When the TAC is released each PE, which received a broadcast message, could begin execution of the broadcast simultaneously.

However, the sending of broadcast communications in this manner can result in significant reduction in performance.

1.2.4 Modification of the FCP Interfaces

The FCP interfaces consist of high performance hardware designed to aid in PMS monitoring, I/O emulation, and FCP to MMS communications. The combining of the FCP, PMS, and IOP interfaces into a single general purpose interface can result in a significant hardware savings. This interface could be a PE or specially designed hardware.

The single interface concept can be expected to reduce system performance, but not significantly. Loss of performance is expected to be minimal since the FCP, PMS, and IOP functions can almost be considered mutually exclusive events.

When FCP communications are active the MMS is in the initialization or debug mode. When IOP communications are active the MMS is in the run mode. PMS communications cannot be considered mutually exclusive of IOP communications. However, the assumption PMS is based on is that when the user is collecting PMS DATA; emulation efficiency is considered unimportant.

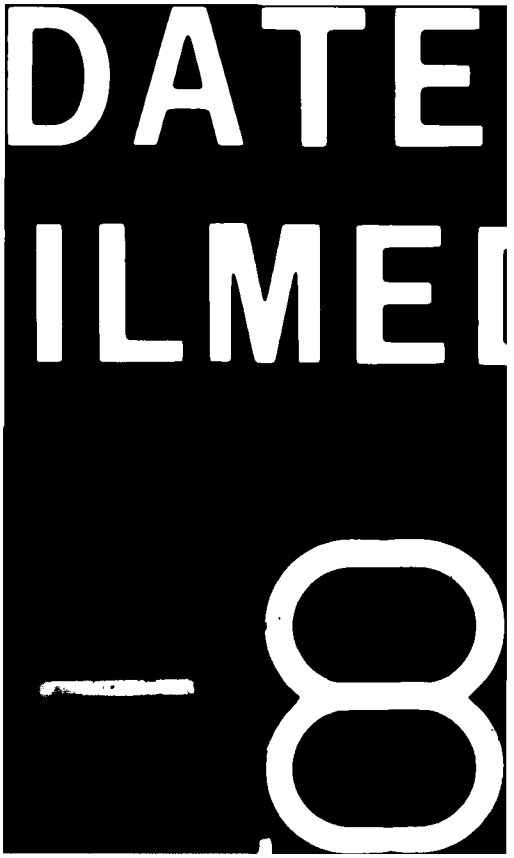
1.2.5 Conclusion

Overall the modifications discussed in this appendix are expected to reduce hardware cost by 50%. The reduction of the number of PE's to 32 or 16, is expected to reduce hardware cost by 10% to 20% respectively. The loss of flexibility is assumed negligible, while the improved ease of use obtained through E-code may prove to be a significant bonus.

It is important to note that this discussion of hardware modifications does not reflect the cost of MMS software (excluding E-code and PEXE) or the effects these changes have on the software.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.


DATE
ILME
8