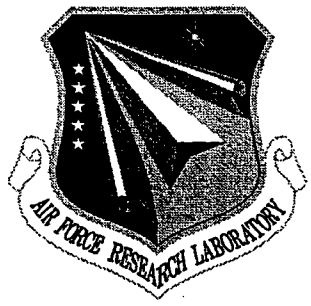
AFRL-IF-RS-TR-1998-35
Final Technical Report
April 1998



DIMINISHING MANUFACTURING SOURCES (DMS) STUDIES FOR JOINT TACTICAL INFORMATION DISTRIBUTION SYSTEMS (JTIDS)

Synectics Corporation

Kathy Martin and Jeff Volp

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

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

AFRL-IF-RS-TR-1998-35 has been reviewed and is approved for publication.

APPROVED: Daniel F. Fayette

DANIEL F. FAYETTE
Project Engineer

FOR THE DIRECTOR: Northrup Fowler

NORTHRUP FOWLER, III, Technical Advisor
Information Technology Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFT, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1998	Final Aug 96 - Sep 97

4. TITLE AND SUBTITLE

DIMINISHING MANUFACTURING SOURCES (DMS) STUDIES FOR JOINT TACTICAL INFORMATION DISTRIBUTION SYSTEMS (JTIDS)

5. FUNDING NUMBERS

C - F30602-95-D-0028/0005
PE - 64754F
PR - 2982
TA - QE
WU - 07

6. AUTHOR(S)

Kathy Martin and Jeff Volp

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

PRIME:
Synectics Corporation
111 East Chestnut Street
Rome NY 13440

SUB:
The Charles Stark Draper Laboratory, Inc.
555 Technology Square
Cambridge MA 02139

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/IFT
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-1998-35

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Daniel F. Fayette/IFT/(315) 330-2151

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The purpose of this study was to provide a feasibility assessment and Rough Order of Magnitude (ROM) cost of using a 3rd party's approach for eliminating microcircuit part obsolescence on a printed wiring board, which was designed by a cognizant JTIDS contractor. The results of this study are to be used in supporting future acquisition strategies (e.g., organic vs contractor repair) for JTIDS life cycle sustainment. As a follow-up/sequel to the initial studies, the overall benefits, risks and impacts associated with the use of Very High Speed Integrated Circuit Hardware Description Language (VHDL) as a Diminishing Manufacturing Sources (DMS) alternative for the chosen JTIDS board or other military applications was evaluated. Although a ROM cost was provided for one of the options developed it is not contained within this final report but was provided to the JTIDS Program Office for their use.

14. SUBJECT TERMS

VHDL, Diminishing Manufacturing Sources

15. NUMBER OF PAGES

56

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	UL

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The purpose of this study was to provide a feasibility assessment and Rough Order of Magnitude (ROM) cost of using a 3rd party's approach for eliminating microcircuit part obsolescence on a printed wiring board, which was designed by a cognizant JTIDS contractor. The results of this study are to be used in supporting future acquisition strategies (e.g., organic vs. contractor repair) for JTIDS life cycle sustainment. As a follow-up/sequel to the initial studies, the overall benefits, risks and impacts associated with the use of Very High Speed Integrated Circuit Hardware Description Language (VHDL) as a Diminishing Manufacturing Sources (DMS) alternative for the chosen JTIDS board or other military applications was evaluated. Although a ROM cost was provided for one of the options developed it is not contained within this final report but was provided to the JTIDS Program Office for their use.

BACKGROUND

JTTDS. The Joint Tactical Information Distribution System (JTIDS)is an advanced radio system which provides information distribution, position, location, and identification capabilities in an integrated form for application to military operations. These capabilities result from the ability of the system to distribute information at high rates, encrypted in such a way as to provide security, and with sufficient jam resistance to yield high reliability communications in hostile electromagnetic environments. The concept of JTIDS was developed in 1975 with the design for the now-existing production terminals consisting of 1980's technology which has contributed to the ever increasing problem of obsolescence within JTIDS.

Terminals. There are four basic terminals, each designed for a specific set of users, that will provide a JTIDS capability, the two most commonly known are the Class 2/2H and 2M terminals. The TB/IF card is resident in the 2M terminal which is a down-sized variant of the Class 2/2H and is intended for use in Army ground applications.

The Class 2 terminal was developed for small platform JTIDS users, principally aircraft and mobile ground units. The terminal provides an Interface Unit (IU) which will tailor the terminal to specific host platforms. A High Power Amplifier Group (HPAG) call be added to increase the transmit power for increased capability; this configuration is referred to as the 2H terminal. The Class 2M terminal is fully inter-operable with the Class 2/2H terminals. Similar to the Class 2 F- 15 terminal, it also has a bilingual capability. It does not have a TACAN or a voice capability; and the number of receivers is reduced from eight (8) to two (2), which results in a small degradation in anti-jam performance.

Time Base/Intermediate Frequency (TB/IF) Circuit Card Assembly. The Time Base Intermediate Frequency (TB/IF) processor circuit card assembly was chosen as the candidate board for this study because a comparison cost for eliminating obsolescence on the board as a result of 1980's technology had already been established by the original design contractor.

TIME BASE/INTERMEDIATE FREQUENCY (TB/IF) CIRCUIT CARD ASSEMBLY STUDY

The scope of this task was to assess the impact of parts obsolescence on the current JTIDS system design, review options to replace obsolete parts, and provide a recommended approach and estimated cost for the selected options.

In this post military specification era, manufacturers have been discontinuing production of unprofitable military grade components. As technology continues to evolve, the lifetime of commercial semiconductor devices is decreasing. In a quest to maximize profit, manufacturers are discontinuing devices that are no longer significant contributors to their bottom line. Consequently, the industry is faced with an accelerated rate of Diminishing Manufacturing Sources (DMS). This is an area where contractors such as Draper Laboratory can draw on their experience to provide alternate solutions to the DMS problem.

Additional JTIDS TB / IF Processor cards were to be fabricated. Draper was provided with a list of 18 components that were either unavailable, or had limited availability. A workaround for those devices was necessary before the board could be fabricated. Because this JTIDS receiver was expected to remain in service for only a limited time, a cost effective means of supporting this board for an additional five years was desired.

The Time Base / IF Processor is comprised of a digital circuit card and an RF circuit card bonded to a center heatsink. Draper began by building a database from the schematics that contained device type, part number, and schematic location. Using this database, it was easy to locate each of the problem components, and determine how it was used in the circuit. Most of the problem components were on the digital circuit card.

Several of the devices were older Transistor Transistor Logic (TTL) digital logic devices, for which suitable substitutions were available. The remainder were all Emitter Coupled Logic (ECL) devices spread over several schematic pages. We were able to identify a functional block containing a majority of the ECL circuitry. We examined each of the interfaces with ECL functional block, and found that virtually all interfaces were with TTL digital logic through level translators. We focused our efforts on the ECL devices, and identified three basic approaches to resolve the DMS issue for this board:

1) Replace unavailable devices with functionally similar devices. This could involve additional testing to qualify commercial or industrial components for this application. It could also involve purchasing devices from a company that specializes in producing obsolete devices. The printed circuit layout would have been modified to accommodate any devices that did not have footprints matching the original devices. The layout would have been kept as close to the original as possible to minimize risk. The goal of this approach was to minimize changes. The risk was that the ECL devices still being used could cause supportability problems in the future.

2) Replace unavailable devices with functionally identical programmable gate array(s). In this approach, circuitry comprised of obsolete ECL devices would have been duplicated within one or more modem high-speed programmable gate array(s). This approach was feasible because virtually all affected ECL used level translators to interface with TTL levels at both inputs and outputs. Modern high-speed programmable gate arrays should have sufficient speed to replace the ECL Circuitry after the level translator delays are added. Any ECL and TTL circuitry that could not be integrated into a programmable gate array would have been redesigned using currently available devices of similar function. The printed circuit layout would have been modified but the overall layout would have been kept as close to the original as possible, taking advantage of the reduced parts count. Because all identified problem devices would have been engineered out, the board should have been more easily supported than with option one.

3) The third approach was to redesign the entire card using currently available devices. Only devices expected to have a long lifespan would have been utilized. The functional and electrical characteristics of the card would have been retained, and the device count reduced. The layout would have been simplified by taking advantage of the reduced parts count. This approach had the highest cost and design risk. It required complete understanding of the functional and I/O characteristics of the board. Using devices expected to have a long lifetime, the board should have remained supportable into the future. However, as in the other options, there was no guarantee that all devices would remain available.

Given the limited anticipated life span for this board, we do not believe that a complete redesign at this point was the cost-effective solution. To conserve the funds available for this task, we did not expend the effort to calculate a ROM for a complete redesign. Nor did we generate an estimate for option one because in the time available, we could not identify the costs of developing alternative sources for the obsolete components. We submitted a cost estimate for option two, which we believed was the most cost-effective solution of supporting the Time Base / IF Processor for another five years.

ASSUMPTIONS

1) All parts except the 18 listed were available. We contacted manufacturers of the Source Control Drawing (SCD) devices, and did confirm that all SCD parts were still available.

2) All SCD parts except amplifiers were estimated to cost $1000 each. SCD amplifiers were estimated at $100 each. In the time available, we were only able to obtain quotes for three SCD part types. We estimated the cost of other SCD devices from

those quotes. We did not include any cost for lot qualification testing for the prototype build.

3) There are now two programmable logic devices on this card. We assumed the programming files would be available for these devices.

4) The cost of parts and fabrication for one prototype board was included under the engineering effort.

5) Test equipment specified in the ATP for this card would be available for testing the prototype. Three trips to the test facility were projected. The first trip was to become familiar with the test procedure and equipment (using an existing board). The second trip was to perform a functional test on the prototype board. The third trip was projected to resolve any problems encountered during the first functional test of the prototype board.

6) No environmental qualification would be needed due to similarity to previous board.

The results of this effort were provided to the JTIDS Program Office at ESC, Hanscom AFB, MA. With the exception of the developmental cost effort, a copy of these view graphs are contained in Appendix A.

VHDL ASSESSMENT

Subsequent to the study on this particular board, the question came up on how VHDL could help solve the DMS problem. Not enough resources were left to investigate this issue directly for the JTIDS project. However, Draper previously performed a corporate sponsored research project that used VHDL to recapture the design of an existing Trident Missile circuit board. This is very similar to what is needed for the JTIDS DMS problem, and the final report for that Draper CSR project is included as Appendix B.

APPENDICES

Appendix A - Viewgraph Results of IF/TB Processor Board Study

Appendix B - Draper CSR VHDL Study

APPENDIX A

VIEWGRAPH RESULTS OF IF/TB PROCESSOR BOARD STUDY

JTIDS TIME BASE/IF PROCESSOR

Diminishing Manufacturing Resources Study

- Background
- Board Characteristics
- Redesign/remanufacture alternatives
- Estimated effort
- Recommendations

Background

- Time Base/IF Processor is a circuit card in the JTIDS Terminal
- Identified 18 components either unavailable or limited availability
- Original design early 1980s
- Terminals expected to be in inventory until 2003
- Board replacements and repairs are necessary to support fielded LRUs

DMS Survey

- 18 parts identified as either obsolete or difficult to procure
  - 13 are high speed emitter coupled logic (ECL)
  - 5 are TTL
- Form, fit, funtion replacements have been found for the TTL parts (issue of temperature range needs to be resolved)
- Other than the 18 parts, all other parts are available

Remanufacture Options

- Procure obsolete parts from sources specializing in replacement devices of DMS items
- Replace 18 parts with functional equivalents in current technology
  - Replacements are not form, fit, function (13 ECL devices)
    - Requires new board layout
    - No parts reduction
- Apply new technology to integrate functionality of ECL devices
  - Most of the parts are functionally grouped
  - New technology (gate array) available
  - Requires board redesign
    - reduced parts count
    - lower packaging density
- Redesign entire board using current technology
  - Complete board redesign
  - Significant reduction in parts count

Comparison of alternatives

	Parts substitution	*ECL replacement*	*ECL redesign*	*Complete redesign*
Design cost	L	M	M	H
Design risk	L	L	M	H
Parts count	H	H	M	L
Schedule	L	M	M	H
Sustainment	H	M	M	L
Relative failure rate	H	H	M	L

L-Low
M-Medium
H-High

Redesign Ground Rules

- ECL Redesign - replace 13 ECL devices with current technology
- SCD parts (Source control drawing) are available
- Program files for two programmable devices required
- Testing limited to functionality and integration with terminal
- No environmental requalification due to similarity with previous design

Redesign Effort

- Electrical
  - Circuit Analysis
  - RF and Logic redesign
- Pakaging
  - Schematic capture in CAD
  - PC board layout
  - Frame, covers, and EMI dividers
  - fixtures
- Board Protoype
  - Materials
  - Fabrication
  - Inspection
- Test
  - Board Level
  - Terminal integration

Recommendation

- Redesign ECL area to reduce dependence on obsolete parts
- Retain board level FFF
- If expected lifetime is increased, to 10+ years, reevaluate complete redesign with current technology to prevent a second iteration in 5-7 years.

APPENDIX B

DRAPER CSR VHDL STUDY

Appendix I: CSDL VHDL CSR Task

The Company Sponsored Research (CSR) project documented in the attached report was undertaken back in 1992 in an effort to integrate new design tools and methodology into the design environment at CSDL. For the purpose of exercising new simulation and logic synthesis tools, a portion of an existing electronics assembly was selected as a demonstration vehicle, and retargeted to a Field Programmable Gate Array (FPGA). (Reference objective no. 1 in the attached CSR report). The work accomplished under objective 1 of the CSR task resulted in a detailed illustrative example which shows how an old design can be captured and retargeted to an FPGA. It is proposed that this CSR demonstration vehicle may be of value as a guideline in applying this methodology to the retargeting of a selected portion of the JTIDS Time Base / IF Processor board.

Using this approach, the target design is captured in VHDL, a technology-independent representation which can be synthesized to a variety of FPGA and ASIC devices. In addition, once the design is captured in VHDL it has the potential for retargeting again at a later date if the FPGA is no longer available. It also opens up the possibility for easy inclusion of minor design changes. In general, it may make sense to fold all of the digital logic on a given module, excluding processors and bus drivers, into an FPGA.

This appendix includes the following items:

- ❑ CSR final report
- ❑ Memo summarizing results of CSR objective no. 1.
- ❑ Schematic diagram of the MSI/TTL circuit selected for retargeting
- ❑ Synthesizeable VHDL code
- ❑ Logic synthesis script, includes design constraints such as size, and clock freq.
- ❑ Block diagram of simulation test-bench
- ❑ Presentation viewgraph which summarizes FPGA design flow

Due to the fact that this task was undertaken back in 1992, the VHDL is a Viewlogic specific subset of the language, and there were some minor wrinkles in the VHDL compatibility between simulation and synthesis tools. The tools have matured considerably since that time, and these are no longer issues. It should also be noted that following this CSR task, this exact methodology has been used successfully on a number of FPGA designs at CSDL.

As a further note, it has been my experience that creating a VHDL simulation model of an existing system is an effective method for reverse-engineering digital electronics, given that the original design team is no longer available for consultation. Obviously, it is necessary that the individual(s) responsible for the re-targeting effort have experience with the VHDL language, as well as familiarity with and access to simulation and synthesis tools.

CSR FINAL REPORT

ADVANCED DIGITAL ASIC DESIGN CAPABILITIES
CSR Project No. C95
David McGorty

PROBLEM

With ASICs now available which contain 100K and more gates, increasingly large digital functions can be implemented on a single chip. It is becoming widely recognized that traditional design methods are no longer adequate in managing the size and complexity of such designs. Traditional methods involve technology-specific gate-level design, accomplished in a piece-meal fashion, with system integration done last. Large and complex ASIC designs require that a new methodology be developed.

Over the past three years VHDL has emerged as an industry standard Hardware Description Language, and is being supported by an increasing number of synthesis and simulation tools. In addition, tools are now available for system-level simulation using a composite of gate-level schematics, behavioral HDL models, and "hardware" models. VHDL and the supporting tools enable a new design methodology, sometimes referred to as "top-down", in which system verification occurs before design implementation. Hardware Description Languages make it possible to create technology-independent models of the design at a level of abstraction higher than the gate-level schematic.

Why is this project needed? Although these tools are extremely powerful, simply being trained in the use of these tools does not mean that one can successfully create complex digital designs. The purpose of this project includes not only acquiring an understanding of the tools but more importantly developing a workable methodology for doing a real design.

OBJECTIVES

The overall CSR task was split up into two major objectives:

1) behavioral VHDL capture of an existing design and re-implementation into a current ASIC technology using logic synthesis. This portion of the project would involve demonstrating the design process all the way from the VHDL models to a real FPGA working in a system.

2) conceptualization of a new design using VHDL

In keeping with the "top-down" design philosophy, our approach for both objectives involved the following guidelines:

1) The chip functionality should be described and entered entirely in VHDL, with the VHDL code being the design "master" or "source", rather than an intermediate representation.

2) The exact same VHDL code should be used for both design-verification simulation and logic synthesis.

3) Synthesis of the VHDL code into gates, and finally into an FPGA "burn-file", should be automated to the extent possible. That is, there should be no hand-editing of any schematic diagram or intermediate format text file during the process. It should be noted that this goal, along with number 2 above, becomes an issue due to the fact that software tools from different companies are involved in various steps of the process.

4) The VHDL description should be completely technology independent. That is, the same behavioral description code should be usable for synthesis to multiple ASIC technologies.

Regarding VHDL coding style, it is intended that the code should be written at the highest level of abstraction accepted by the logic synthesis tool, and should represent accepted good software programming practices.

By completing and demonstrating an FPGA design using these guidelines, it was intended that a specific methodology would be developed, along with the required infra-structure of libraries and support systems, which could then be applied to funded projects. This objective has been achieved with unqualified success.

PROGRESS

The following is a summary of accomplishments for objective no. 1.

Module Selection

The Trident MK6 CLU3 module was chosen as the existing design targeted for reimplementation. This module contains the clock sequencing and memory control logic for the Mission Processor (MP). An MP module was preferred since a partial RTL model of the MP already existed from another project. Also, the CLU3 module contains "glue" logic, and a state-machine which runs at a clock frequency of 12 MHz. During the original MK6 development, the logic on this module had to be carefully optimized in order that it would run at the required frequency. It is this type of digital function which can best take advantage of the strengths of synthesis tools.

Design Validation

In order to create a simulation test-bed for the CLU3 module, an existing VHDL RTL model of the Mission Processor was used as a starting point. This MP block diagram had to be re-partitioned such that the block for the CLU3 module exactly matched the MK6 module boundary. Further, the test-bed had to be expanded such that all of the CLU3 module functions were exercised in the test-bed. Once the test-bed had been set up properly, the CLU3 VHDL code debugging process began.

Since the MP module interprets op-codes, assembled MP programs were used as the test stimulus. The test-bed as a whole was verified using the MP "Rapid Functional Test", a selftest stored in MP program ROM. In addition, for a quick CLU3 module debug simulation, an MP test program was written which exercised as much as possible of the CLU3 module within a few dozen instructions.

The VHDL models were debugged using these test programs similar to the way in which one would debug real hardware. MP activity was determined by observing values on the address bus, data bus, and other key signals, displayed as waveforms. When the test program terminated at the end of the "pass" branch in the test program, the VHDL model was considered to be correct.

Writing VHDL for Synthesis

Once the CLU3 VHDL code was considered to be functionally correct, it was then used as input to the logic synthesizer. Maintaining two separate versions of the VHDL could defeat the purpose of the simulation, thus the VHDL was written such that the exact same code used for simulation could also be used as input for the logic synthesis. This was not trivial since our simulation and synthesis tools were purchased from different companies and supported different sub-sets of the VHDL language.

Libraries

Both the logic synthesis and gate-level simulation required gate/macro libraries for the Actel FPGA. The Synopsys synthesis library and the Viewlogic simulation library were purchased from Actel.

Synthesis Process

Synthesis requires two types of input, a VHDL description of the design functionality, and a set of constraints on the design. Area, maximum clock frequency, and propagation delay of specific paths are examples of such constraints. The output is a gate/macro level schematic using the Actel library. This gate-level schematic was validated in two different ways. First, static timing analysis reports generated by Synopsys were reviewed in order to make sure that the circuit would run at speed with adequate margin. Second, the gate-level schematic was translated into Viewlogic format using EDIF and substituted back into the same simulation test-bed in place of the VHDL code. The same MP test program was run with the gate-level CLU3 schematic, verifying that the schematic was functionally correct. In going through this process a number of issues were discovered and resolved. Several passes through the synthesis, simulation, and timing analysis process were required.

State-machine Optimization

As the default, the synthesizer generates state-machines with a "register" implementation. Based on static timing analysis reports, this implementation was too slow to run at the required clocks frequency. Using the Synopsys finite state machine optimization feature, the state machine portion of the circuit was extracted from the schematic, and then optimized separately from the rest of the circuit. The optimizer changed the state machine implementation from a register to what Synopsys refers to as "one-hot". The one-hot style uses one flip flop for each state, and only one flip flop can be set in a given state. This configuration creates more logic (chip area) but allows the circuit to run at a faster clock frequency. With a "one-hot" state machine, the timing requirements were met with adequate margin.

Synthesis "Scripts"

All synthesis commands were run from a Synopsys "script". This script contains the sequence of commands which control the synthesis process, identifying libraries, include files, synthesis variables, and specifying the constraints imposed on the synthesis process. With each iteration, the script was modified as required, and then re-run to produce a new schematic.

Once the synthesis was complete, UNIX scripts were used to run the commands for translating the schematic into the required format and for programming the FPGA. The set of scripts serves as a complete record of the process used to create a specific gate-level circuit, and allows the FPGA programming file to be automatically reproduced from the VHDL.

Functional and Timing Verification

For functional verification, the schematic created by synthesis was substituted back into the MP test-bed simulation model in place of the CLU3 behavioral VHDL. It was exercised with the same simulation used in debugging the VHDL.

All of the following methods were used in checking for timing problems:

- Synopsys static timing analysis reports
- Viewlogic functional simulation with unit delays
- Actel static timing analysis reports
- Viewlogic simulation with delays back-annotated from post-route Actel data

Several synthesis iterations were performed using feedback from the Synopsys static timing reports and unit-delay functional simulations. No timing problems were uncovered later by the Actel static timing analysis or back-annotated simulation.

FPGA Programming

Based on static timing analysis reports it was determined that the Actel ACT1 family, which is less expensive, would be too slow, and that a device from the ACT2 family was required. In addition, chip area reports indicated that the 1240 device was the correct size for this design. Using the Actel package installed on the SUN IPX workstation, a 1240 FPGA device was programmed.

Device Verification

A Hardware model for the CLU3 FPGA device was created using the LM1000 Hardware modeler and exercised using the same test-bed simulation, with the hardware model substituted in place of the gate-level schematic. No problems were encountered at this stage.

Demonstration in System

Finally, the FPGA was installed on a special adapter module and substituted into the EA in place of the existing CLU3 module. The EA was then given its complete set of factory acceptance tests. All tests which were expected to pass did pass, using the first and only FPGA device programmed. This served to demonstrate that the entire design process was successful.

Progress under Objective No. 2

In order to explore this methodology in the conceptualization of a new design, we attempted to choose a function for which a real application may exist. The function chosen was a clock synchronization and data exchange mechanism for a fault-tolerant network of redundant processors. It was the goal that this function be implemented on a single ASIC (per fault-containment region) and work with various different processor architectures.

For a fault-tolerant configuration of processors running in lock-step (clock deterministic) there must be a mechanism for fault tolerant synchronization of the processor clocks such that each processor gets the same number of clock pulses over a specified interval. In addition, there must be a mechanism for data exchange between the redundant processor copies. The data exchange mechanism allows all processors to be given identical initial states, and provides a mechanism for all processors to receive identical copies of input data.

Using the AIPS (Advanced Information Processing System) design as a starting point, a behavioral VHDL model of fault tolerant processors and communicators was created, and interconnected in a triplex configuration. Once the basic AIPS concept had been captured, it was the intent that the following features be incorporated into the design:

- Serial transmission links for data exchange
- Multiple concurrent serial data exchanges

- Synchronization of interrupts using data exchange path
- Triplex processor network with four fault-containment regions

This portion of the project was originally planned for DFY 1993 but was not approved. At the completion of DFY92, the AIPS baseline design has been modeled in VHDL and verified in simulation. This simulation model consists of hardware models of the SBR9000 processor, interconnected in a triplex configuration with complete working VHDL models of the data and clock communicators and interstages.

MEMO SUMMARIZING CSR OBJECTIVE NO.1 RESULTS

MEMO

Memo No: ETC:92:30
FBM-378-92

To: J. Cate SP234

From: Wayne Wilson/Dave McGorty

Date: June 4, 1992

Subject: **DRAPER CORPORATE FUNDED RESEARCH PROJECT #95 - CLU3 Module to One CHIP**

Copies: Distribution

Here is a very quick review of the CSDL research effort that went into miniaturizing the CLU3 module (63 SSI DRAPA parts) onto one field programmable CGA:

1) The CLU3 module is the central timing generator for all memory sequencing to PWM, SRAM and PROM for the Mission Processor. It runs directly off the 1 2MHZ clock and is critical to timing margin in the Mission Processor.

2) The new CLU3 module chip design was captured in the VHDL hardware description language and simulated in a VHDL model of the Mission Processor to insure functional correctness of the new CLU3 CGA

3) Next the VHDL description of the CLU3 chip was transformed into gates by automatic logic synthesis using the Synopsys synthesis CAE tools. Synthesis was an iterative process performed to optimize the chip for speed and timing margin prior to fabrication.

4) Once the actual gate level model of the synthesized CLU3 chip was found to meet the timing performance requirements needed by the Mission Processor, the CLU3 chip was fabricated at CSDL using an ACTEL (4,000 gate capacity) field programmable CGA.

The Charles Stark Draper Laboratory, Inc.
555 Technology Square, Cambridge, Massachusetts 02139 3563
Telephone: 617 258-1000

5) With CLU3 chip in hand, a further verification of the part was performed prior to installing it in the EA. The physical device was plugged into CSDL's hardware modeler and then resimulated. Timing path information was extracted from the physical chip to guarantee correct performance in the EA.

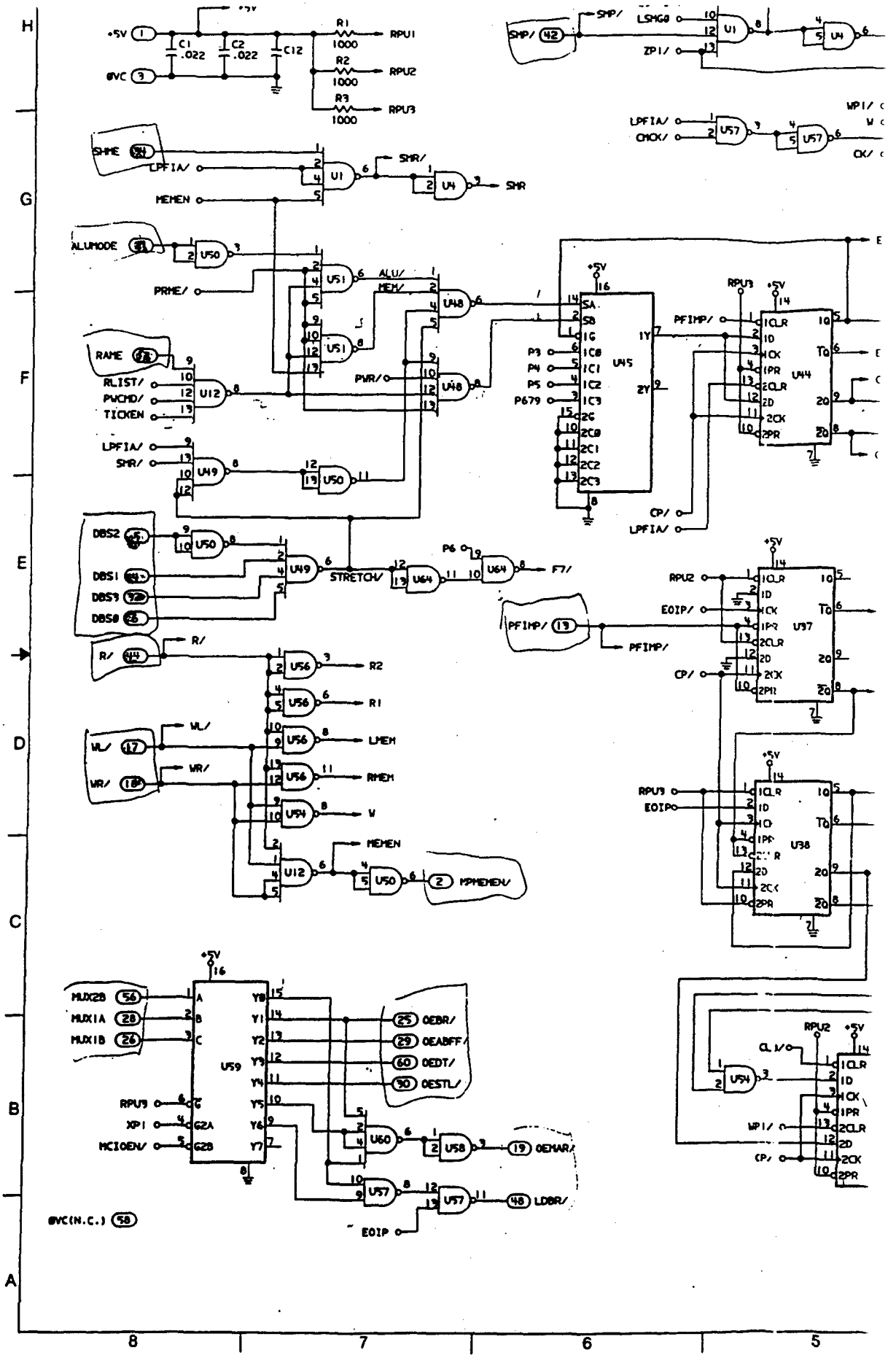
6) At this point the device was installed on a blank module fixture (Exhibit 1. Miniaturized CLU3 Module) and, on an extender, plugged into EA E7.

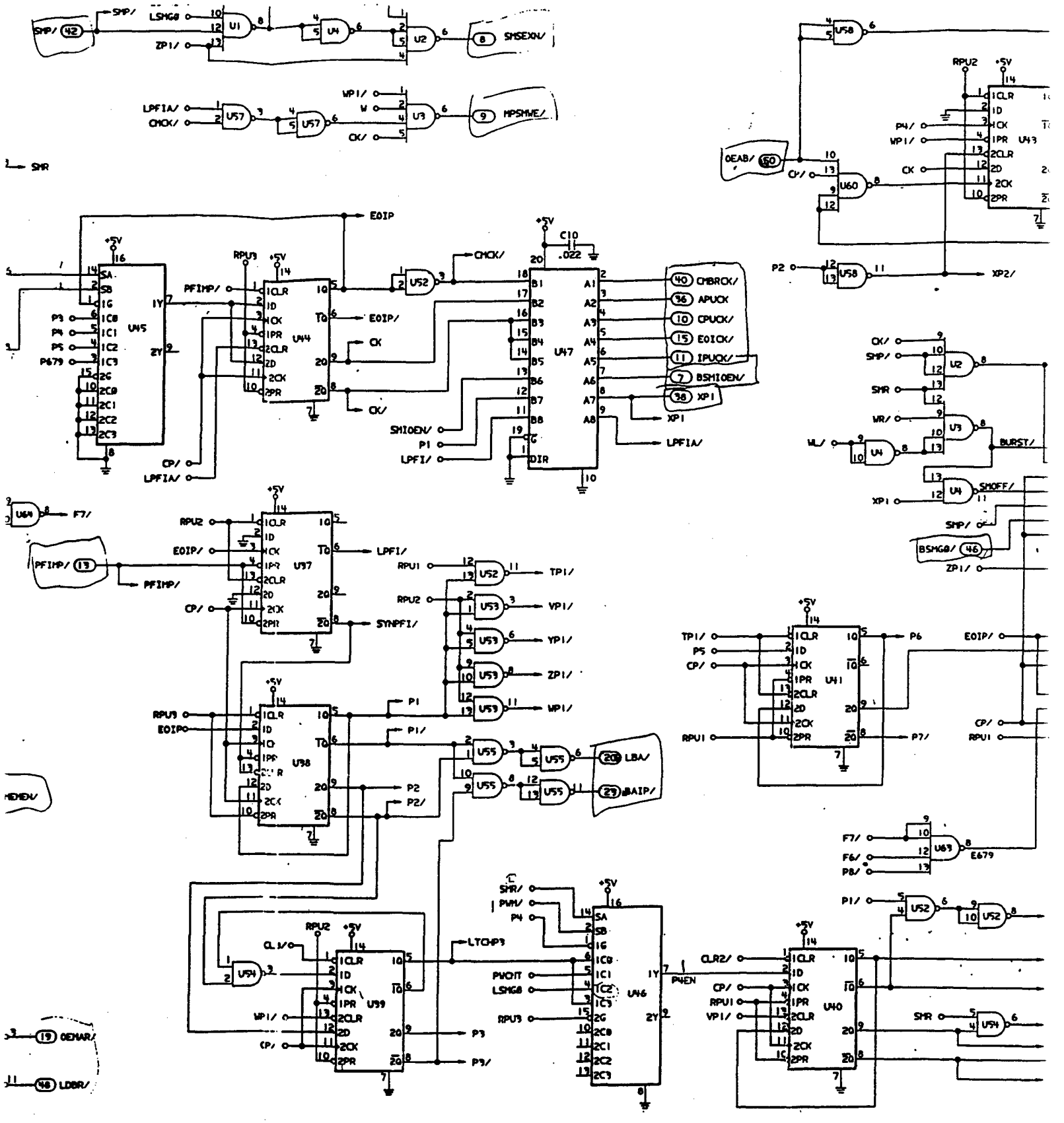
7) RFT ran successfully and a full set of CETs was run on TREATS successfully (with one minor oversight on a PFI timing signal which is fully understood and easily correctable) without modification to first silicon !

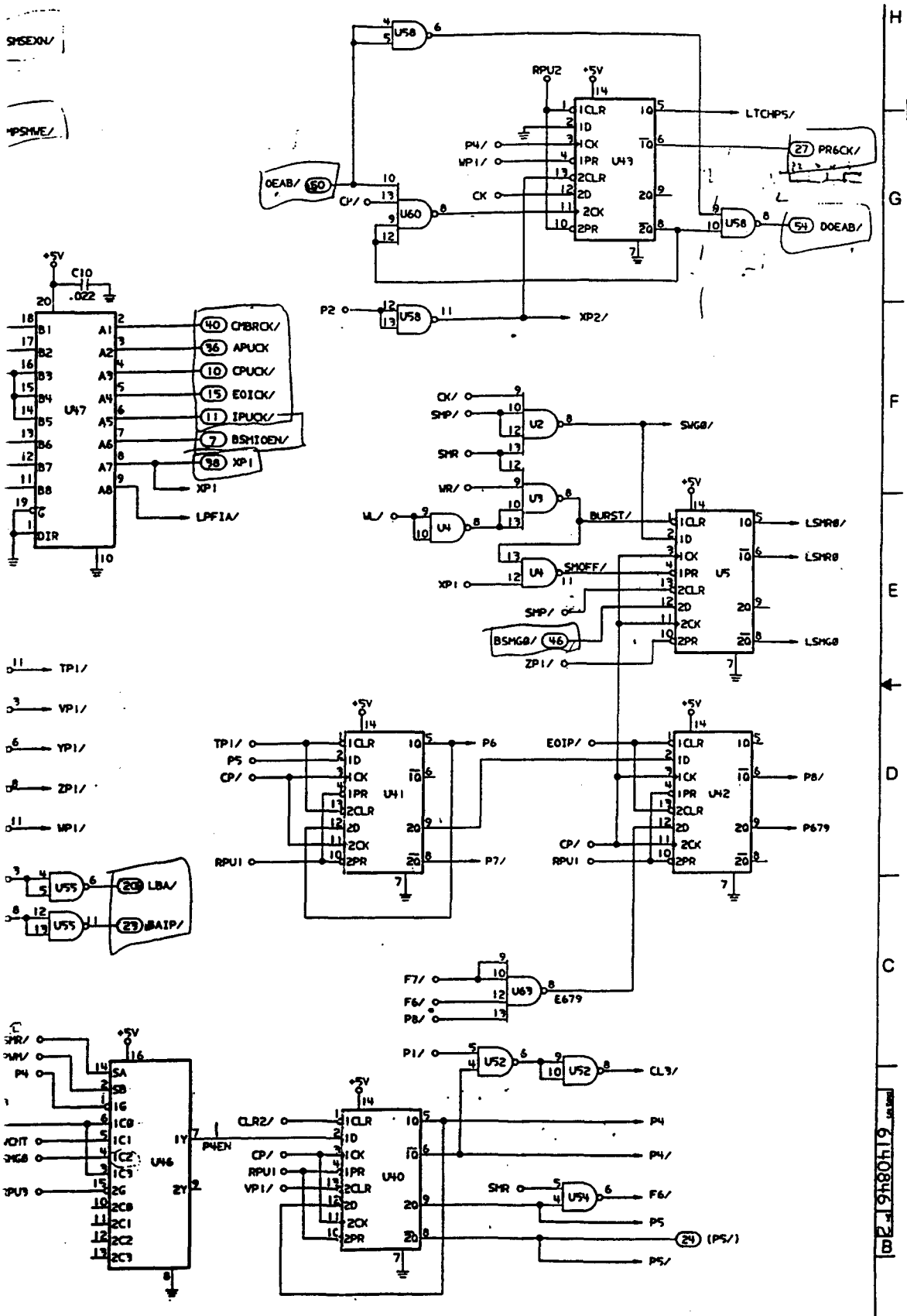
8) The module was further run on the MTS station, but because of the margin improvement on the new chip, the MTS vectors would need to be adjusted for successful module verification.

In conclusion, we believe we have demonstrated the ability to capture the existing MK6 digital circuits in VHDL, an IEEE standard form of design capture, that allows for future fabrication to any new military or commercial technology. I think you can conceive of how this method could facilitate extending the design of MK6 beyond its expected life cycle with the infusion of a future technology. As well, there may be some application to a non-nuclear strike capability (a less expensive non-hardened version of the Guidance System) while preserving existing test equipment and diagnostic software.

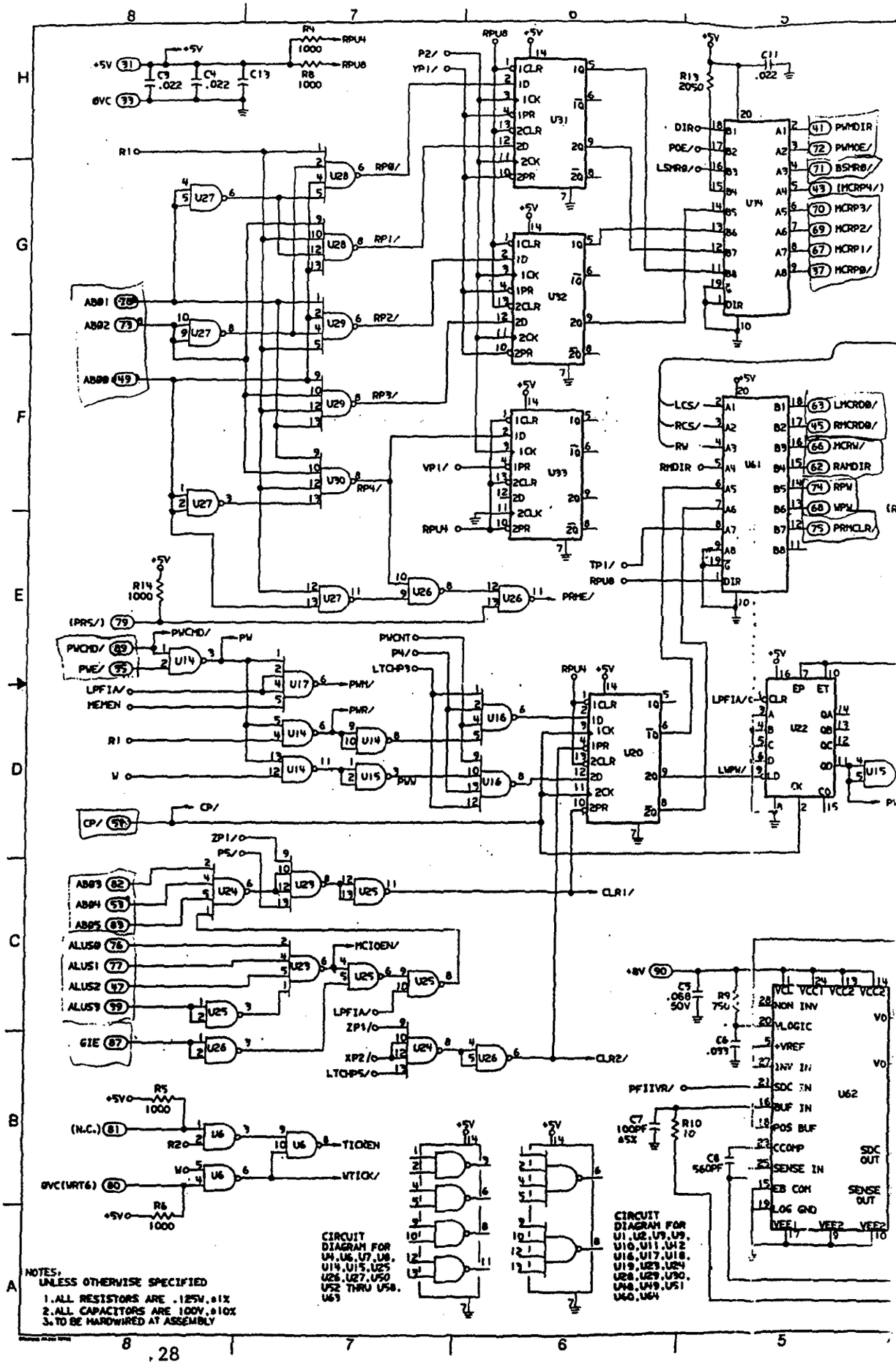
DEMONSTRATION CIRCUIT TTL SCHEMATIC


+5V
C1 .022
C2 .022
C12
R1 1000
R2 1000
R3 1000
RPU1
RPU2
RPU3
SHME
LPFIA/
MEMEN
SHR/
SHR
ALUMODE
PRME/
ALU/
MEM/
RAME
RLIST/
PWCMD/
TICKEN
PWR/
P3
P4
P5
P679
U45
1Y
2Y
PFIMP/
U44
CP/
LPFIA/
DBS2
DBS1
DBS3
DBS0
STRETCH/
P6
F7/
U64
R/
R2
R1
WL/
WR/
LMEM
RMEM
W
MEMEN
MPMEMEN/
EOIP/
PFIMP
U37
U38
MUX2B
MUX1A
MUX1B
U59
RPU3
XPI
MCIOEN/
OEBR/
OEABFF/
OEDT/
OESTL/
OEMAR/
LDBR/
@VC(N.C.)
EOIP
CLK/
WPI/
CK/
8
7
6
5


SMP/
LSMG0
ZP1/
U1
U4
U2
SMSEXN/
LPF1A/
CMCK/
U57
WP1/
W
CK/
U3
MPSMME/
SMR
EOIP
+5V
C10
.022
RPU3
PFIMP/
U45
U44
U47
CMCK/
EOIP/
CK
CP/
LPF1A/
SMIOEN/
P1
LPF1/
DIR
CMBROK/
APUCK
CPUCK/
EOICK/
IPUCK/
BSMIOEN/
XP1
LPF1A/
RPU2
U58
OEAB/
U60
P4/
WP1/
P2
XP2/
WR/
WL/
U64
F7/
U37
EOIP/
PFIMP/
SYNPFI/
RPU1
U52
U53
TP1/
VP1/
YP1/
ZP1/
WP1/
RPU0
U38
P1
P1/
P2
P2/
U55
LBA/
BAIP/
U54
U59
LTCHP3
SMR/
PWM/
P4
U46
PWOUT
LSMG0
RPU3
P3
P3/
CLR2/
P4EN
U40
VP1/
U41
TP1/
P5
P6
P7/
BURST/
SMOFF/
BSMG0/
E679
F6/
P8/
OEMAR/
LDBR/



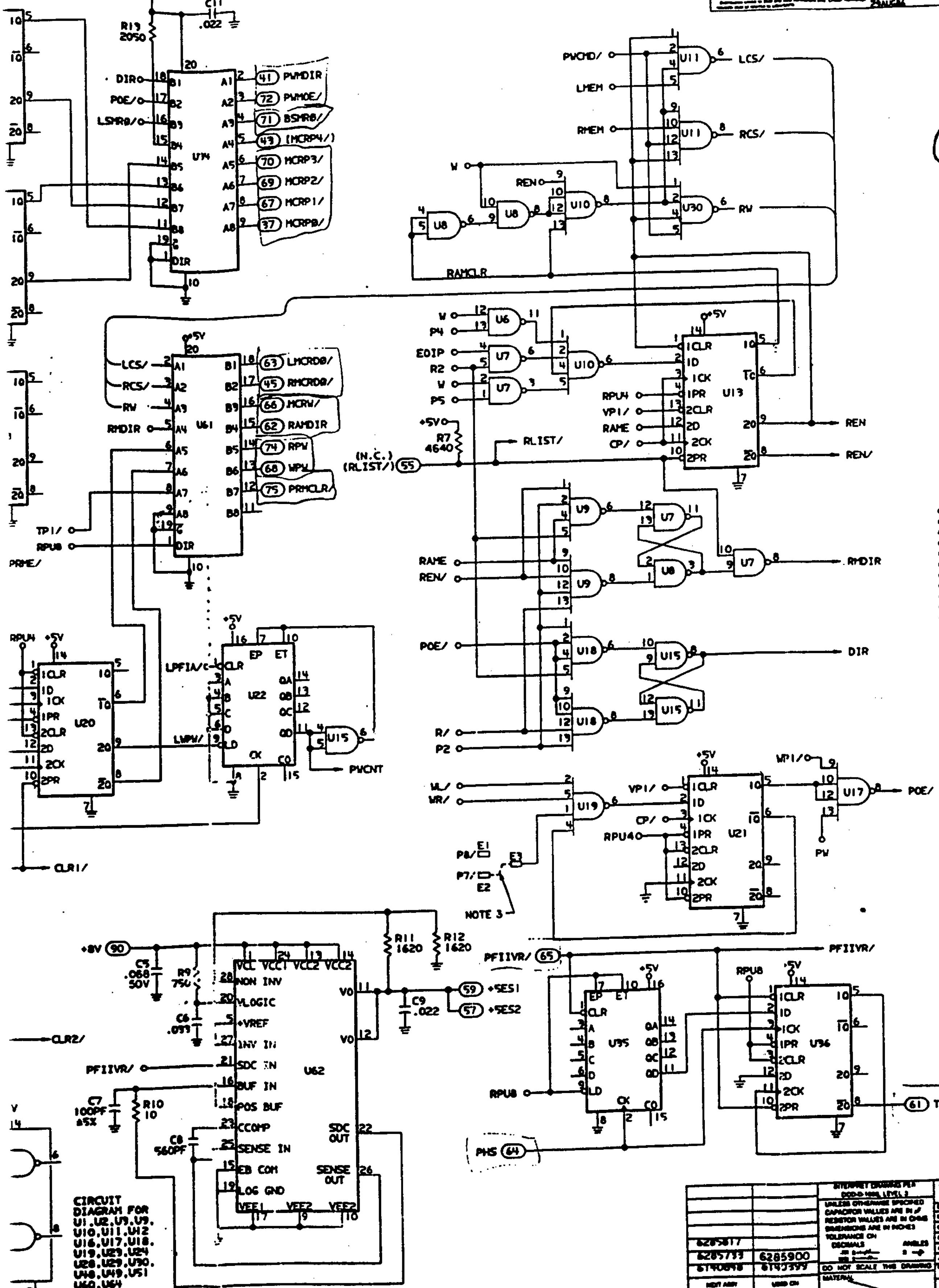
(3)


NOTES:
UNLESS OTHERVISE SPECIFIED
1.ALL RESISTORS ARE .125W,±1%
2.ALL CAPACITORS ARE 100V,±10%
3.TO BE HARDWIRED AT ASSEMBLY
CIRCUIT DIAGRAM FOR U4,U6,U7,U8, U14,U15,U25 U26,U27,U50 U52 THRU U58, U63
CIRCUIT DIAGRAM FOR U1,U2,U9,U9, U10,U11,U12 U16,U17,U18, U19,U23,U24 U28,U29,U30, U48,U49,U51 U60,U64

①

–	RELEASED PER RD 117526
A	AUTHENTICATED PER RD 117756
B	REVISED PER RD 117587
C	REVISED PER RD 117698
D	REVISED PER RD 117733
E	REVISED PER RD 117812

CLU3 Schematic



REF DES	PART NO.
R1	5936388-461
R2	-461
R3	-461
R4	-461
R5	-461
R6	-461
R7	-493
R8	-461
R9	-455
R10	-401
R11	-471
R12	-472
R13	-476
R14	5936388-461
C1	5936311-33
C2	-33
C3	-33
C4	-33
C5	-43
C6	5936311-36
C7	5936310-13
C8	5936311-9
C9	-33
C10	-33
C11	5936311-33
C12	NOT USED
C13	NOT USED

REF DES
U1
U2
U3
U4
U5
U6
U7
U8
U9
U10
U11
U12
U13
U14
U15
U16
U17
U18
U19
U20
U21
U22
U23
U24
U25
U26
U27
U28
U29
U30
U31
U32
U33
U34
U35
U36
U37
U38
U39
U40
U41
U42
U43
U44
U45
U46
U47
U48
U49
U50
U51
U52
U53
U54
U55
U56
U57
U58
U59
U60
U61
U62
U63
U64

ALTERN. REF DES
C1 THRU C4
C6
C7
C8
C9
C10
C11
C5

6285817
6285733 6285900
6140848 6140399
NEXT ASSY USED ON
APPLICATION

DO NOT SCALE THIS DRAWING

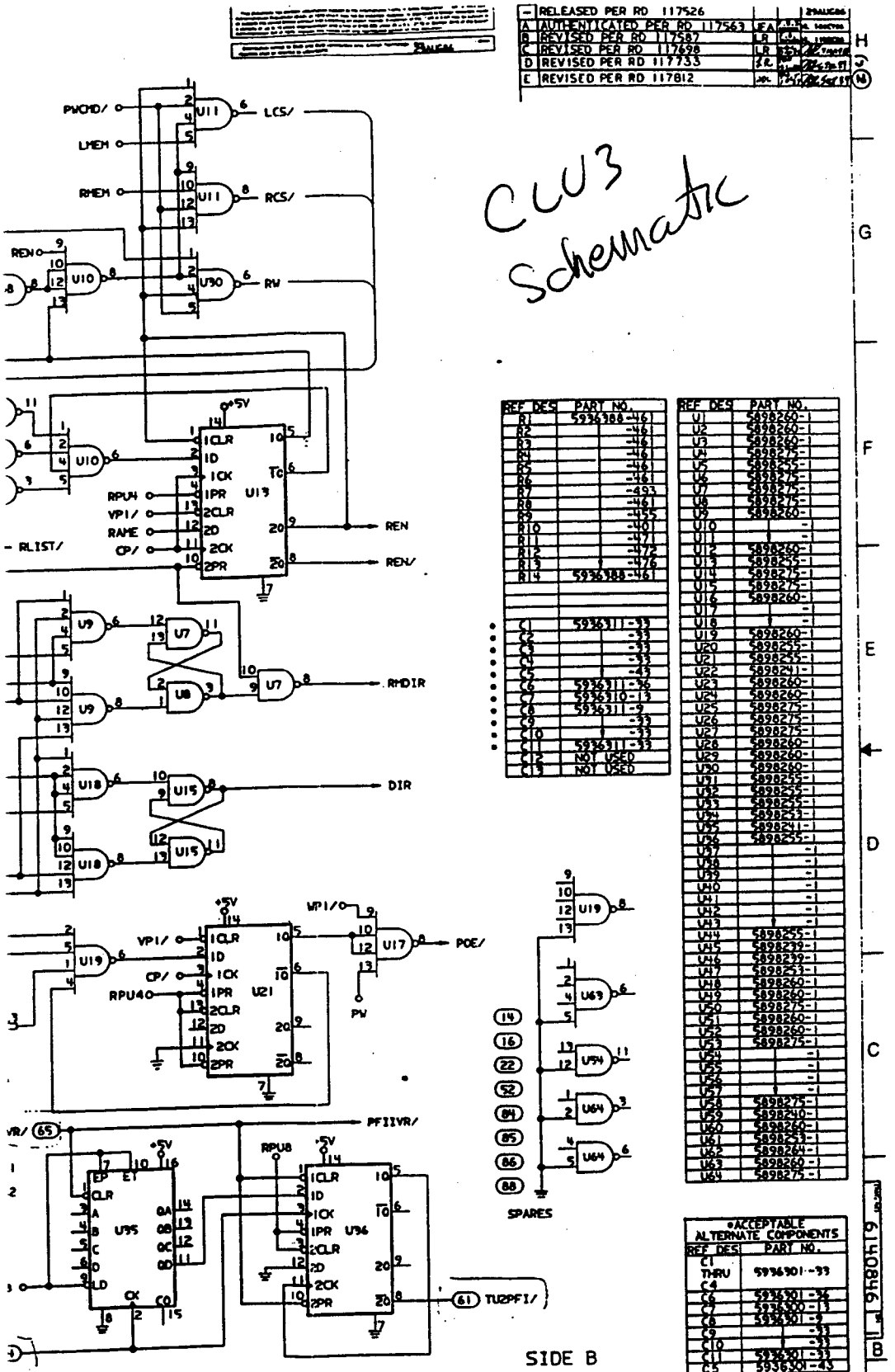
SCHEM CL E

E 53711

SCALE NONE

6 5 4 3 2

②



–	RELEASED PER RD 117526		
A	AUTHENTICATED PER RD 117563	LEA	
B	REVISED PER RD 117587	LR	
C	REVISED PER RD 117698	LR	
D	REVISED PER RD 117733		
E	REVISED PER RD 117812		

REF DES	PART NO.
R1	5936388-46
R2	-46
R3	-46
R4	-46
R5	-46
R6	-46
R7	-493
R8	-461
R9	-455
R10	-401
R11	-471
R12	-472
R13	-476
R14	5936388-461
C1	5936311-33
C2	-33
C3	-33
C4	-33
C5	-43
C6	5936311-36
C7	5936310-13
C8	5936311-9
C9	-33
C10	-33
C11	5936311-33
C12	NOT USED
C13	NOT USED

REF DES	PART NO.
U1	5898260-1
U2	5898260-1
U3	5898260-1
U4	5898275-1
U5	5898255-1
U6	5898275-1
U7	5898275-1
U8	5898275-1
U9	5898260-1
U10	-1
U11	-1
U12	5898260-1
U13	5898255-1
U14	5898275-1
U15	5898275-1
U16	5898260-1
U17	-1
U18	-1
U19	5898260-1
U20	5898253-1
U21	5898255-1
U22	5898241-1
U23	5898260-1
U24	5898260-1
U25	5898275-1
U26	5898275-1
U27	5898275-1
U28	5898260-1
U29	5898260-1
U30	5898260-1
U31	5898255-1
U32	5898255-1
U33	5898255-1
U34	5898253-1
U35	5898241-1
U36	5898255-1
U37	-1
U38	-1
U39	-1
U40	-1
U41	-1
U42	-1
U43	-1
U44	5898255-1
U45	5898239-1
U46	5898239-1
U47	5898253-1
U48	5898260-1
U49	5898260-1
U50	5898275-1
U51	5898260-1
U52	5898260-1
U53	5898275-1
U54	-1
U55	-1
U56	-1
U57	-1
U58	5898275-1
U59	5898240-1
U60	5898260-1
U61	5898253-1
U62	5898264-1
U63	5898260-1
U64	5898275-1

ACCEPTABLE ALTERNATE COMPONENTS	
REF DES	PART NO.
C1 THRU C4	5936901-33
C6	5936901-36
C7	5936900-13
C8	5936901-9
C9	-33
C10	-33
C11	5936901-33
C5	5936901-43

6285817	
6285753	6285900
6140848	6143399

DO NOT SCALE THIS DRAWING

SCHEMATIC
CLU3
EA

SIZE	FSCM NO	DRAWING NO	REV
E	53711	6140846	E

SCALE NONE SHEET 1 OF 2

(3)

SYNTHESIZABLE VHDL MODEL OF DEMONSTRATION CIRCUIT

```
------------------------------------------------------------------
-- Clock Sequencer and Memory Control Logic (clu3 module)
------------------------------------------------------------------
-- This is a revised version of the original CLU3 model written for
-- the MP.  It now represents the actual MK6 module I/O and is
-- compatible with Synopsys VHDL Compiler.
--              D. McGorty      12/6/91
-- 1/14/92 Revised AB contention logic to eliminate gated clock DJM
-- convert to Synopsys format by removing all "--@@" and ";--$$"
--              djm 1/21/92
------------------------------------------------------------------
--
--@@ use WORK.TYPES.ALL;
--@@ use WORK.VLMATH.ALL;
-- synopsys translate_off
entity clu3synth is
-- synopsys translate_on

--@@ entity csr1 is
        port(
                phs,
                pfiivrn,
                alus3,
                alus2,
                alus1,
                alus0,
                smpn,
                oeabn,
                mux1b,
                mux1a,
                mux2b,
                dbs3,
                dbs2,
                dbs1,
                dbs0,
                cpn,
                pwcmdn,
                alumode,
                wrn,
                wln,
                rn,
                bsmg0n,
                gie,
                pfimpn,
                shme,
                rame,
                pwen,
                ab05,
                ab04,
                ab03,
                ab02,
                ab01,
                ab00:           in vlbit;

                cmbrckn,
                apuck,
                cpuckn,
                eoickn,
                ipuckn,
                baipn,
                lban:           out vlbit;
                xp1:            inout vlbit;
                doeabn,
                lmcrd0n,
```

```
                    rmcrd0n,
                    mcrwn,
                    ramdir,
                    pr6ckn,
                    mpmemenn:       out vlbit;
                    mcrp0n,
                    pwmdir:         out vlbit;
                    pwmoen:         inout vlbit;
                    rpw:            out vlbit;
                    wpw:            inout vlbit;
                    oebrn,
                    oeabffn,
                    oedtn,
                    oestln,
                    oemarn,
                    ldbrn,
                    smsexnn,
                    mpsmwen,
                    prmclrn,
                    bsmioenn,
                    mcrp1n,
                    mcrp2n,
                    mcrp3n,
                    tu2pfin,
                    bsmr0n:         out vlbit);

-- synopsys translate_off
end clu3synth;
-- synopsys translate_on
--@@ end csr1;

-- synopsys translate_off
architecture behavior of clu3synth is
-- synopsys translate_on
--@@ architecture behavior of csr1 is

type osc_tick is (reset,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10);

type cycle_type is (startup_cycle,ram_read,ram_write,pwm_read,pwm_write,prom_read,
                    shm_cycle,register_transfer,cpu_alu,stretched);

signal  xp2,
        force_pwm,
        master_reset,
        clocked_reset,
        pwm_busy,
        mcio_enable,
        sync_reset,
        ram_address,
        ram_enable,
        write_strobe,
        pwm_address,
        shm_address,
        memory_cycle,
        shm_request,
        shm_grant,
        shm_lockout,
        eoip,
        eoipn,
        read,write,
        write_left,
        write_right,
        burstn,
        register_clk,
```

```
        latched_shm_request,
        latched_shm_grant,
        shm_enable:             vlbit;

signal  dbs:            vlbit_1d(0 to 3);

-- synopsys translate_off
constant reg_delay:     time := 10 ns;
-- synopsys translate_on

begin

        read <= not rn;
        write <= not wrn or not wln;
        memory_cycle <= read or write;
        write_left <= not wln and rn;
        write_right <= not wrn and rn;
        pwm_address <= not pwen;
        ram_address <= rame;
        shm_address <= shme;
        shm_request <= shm_address and memory_cycle and not clocked_reset;
        shm_lockout <= not smpn;
        shm_grant   <= not bsmg0n;
        mcio_enable <= (not alus0) and alus1 and alus2 and alus3;
        shm_enable <=  latched_shm_request and latched_shm_grant
                         and not shm_lockout and not xp1;
        dbs <= dbs0 & dbs1 & dbs2 & dbs3;
        force_pwm <= not pwcmdn;
        master_reset <= not pfimpn;
        burstn <= not (write_left and not write_right and shm_request);

-------------------------------------------------------
Latched_PFI:process(master_reset,eoipn)
begin
        If master_reset = '1' then
           clocked_reset <= '1'  ;

        --@@ elsif eoipn'event and eoipn = '1' then
        -- synopsys translate_off
        elsif pchanging(eoipn) and eoipn = '1' then
        -- synopsys translate_on

           clocked_reset <= '0'  ;
        end if;
end process;
-------------------------------------------------------
SYNPFI:process(master_reset,cpn)
begin
        If master_reset = '1' then
           sync_reset <='1'  ;

        -- synopsys translate_off
        elsif pchanging(cpn) and cpn = '1' then
        -- synopsys translate_on
        --@@ elsif cpn'event and cpn = '1' then

           sync_reset <= '0'  ;
        end if;
end process SYNPFI;
-------------------------------------------------------
pwm_holdoff:process(cpn,wpw,clocked_reset)
variable wait_count: integer ;--$$ range 0 to 8;
begin
        if clocked_reset = '1' then
```

```
            wait_count := 1;
            pwm_busy <= '1' after reg_delay;

        -- synopsys translate_off
        elsif pchanging(cpn) and cpn = '1' then
        -- synopsys translate_on
        --@@ elsif cpn'event and cpn = '1' then

            if wpw = '1' then
                wait_count := 1;
            elsif wait_count >= 8 then
                wait_count := wait_count;
            else
                wait_count := wait_count + 1 ;
            end if;

            if wait_count < 8 then
                pwm_busy <= '1' after reg_delay;
            else
                pwm_busy <= '0' after reg_delay;
            end if;

        end if;

end process;
-------------------------------------------------------
MUXDECODE:process(xp1,mcio_enable,mux2b,mux1a,mux1b,eoip)
variable temp8: vlbit_1d(0 to 7);
begin
        temp8 := ("00000000");
        If xp1 = '0' and mcio_enable = '1' then
            temp8(v1d2int(mux1b & mux1a & mux2b)) := '1';
        else temp8(v1d2int(mux1b & mux1a & mux2b)) := '0';
        end if;

        oebrn   <=   not temp8(1);
        oeabffn <=   not temp8(2);
        oedtn   <=   not temp8(3);
        oestln  <=   not temp8(4);
        oemarn  <=   not ( temp8(1) or temp8(5) or temp8(0) );
        ldbrn   <=   not (( temp8(0) or temp8(6) ) and eoip);

end process MUXDECODE;
-------------------------------------------------------
-- ab transceiver bus contention logic
oeab_delay:process(oeabn,cpn)
begin
        if oeabn = '1' then
                doeabn <= '1' after reg_delay;

        -- synopsys translate_off
        elsif pchanging(cpn) and cpn = '1' then
        -- synopsys translate_on
        --@@ elsif cpn'event and cpn = '1' then

            if xp1 = '1' then
                doeabn <= '0' after reg_delay;
            end if;
        end if;

end process;
-------------------------------------------------------
Shared_Mem_Request:process(shm_request,burstn,cpn,xp1,shm_lockout)
begin
```

```
        if burstn = '0' then
            latched_shm_request <= '1';

        -- synopsys translate_off
        elsif pchanging(cpn) and cpn = '1' then
        -- synopsys translate_on
        --@@ elsif cpn'event and cpn = '1' then

            latched_shm_request <=  not eoip and not shm_lockout
                                        and not xp1 and shm_request;
        end if;
end process;
-------------------------------------------------------
Shared_Mem_Grant:process(xp1,shm_lockout,cpn)
begin
        if xp1 = '1' then
            latched_shm_grant <= '0' after reg_delay;

        -- synopsys translate_off
        elsif pchanging(cpn) and cpn = '1' then
        -- synopsys translate_on
        --@@ elsif cpn'event and cpn = '1' then

            latched_shm_grant <= shm_grant or shm_lockout;
        end if;
end process;
-------------------------------------------------------
PTU2PFIN:process(phs,pfiivrn)
variable count: integer ;--$$ range 0 to 9;
begin
        If pfiivrn ='0' then
            count:=0;
            tu2pfin<='0';

        --@@ elsif phs'event and phs = '1' then
        -- synopsys translate_off
        elsif pchanging(phs) and phs = '1' then
        -- synopsys translate_on

            if count = 9 then
                tu2pfin <= '1';
            else
                count := count + 1;
            end if;
        end If;
end process PTU2PFIN;
-------------------------------------------------------
cycle_state:process(sync_reset,cpn)

variable end_of_cycle   :       boolean;
variable next_state,
         current_state  :       osc_tick;
variable current_cycle  :       cycle_type;
variable bank           :       integer ;--$$ range 0 to 3;
variable write_protect  :       boolean;

begin

        if sync_reset = '1' then
            mcrp0n      <= '1' after reg_delay;
            mcrp1n      <= '1' after reg_delay;
            mcrp2n      <= '1' after reg_delay;
            mcrp3n      <= '1' after reg_delay;
            lmcrd0n     <= '1' after reg_delay;
```

```
        rmcrd0n     <= '1' after reg_delay;
        wpw         <= '0' after reg_delay;
        rpw         <= '0' after reg_delay;
        xp1         <= '1' after reg_delay;
        lban        <= '0' after reg_delay;
        baipn       <= '0' after reg_delay;
        pr6ckn      <= '0' after reg_delay;
        pwmoen      <= '1' after reg_delay;
        xp2     <= '0' after reg_delay;
        eoip    <= '0' after reg_delay;
        register_clk <= '0' after reg_delay;
        end_of_cycle := false;
        write_protect := false;
        write_strobe <= '0' after reg_delay;
        bank := 0;
        current_cycle := startup_cycle;
        ram_enable <= '0' after reg_delay;
        current_state := reset;
        next_state := p1;

-- synopsys translate_off
elsif pchanging(cpn) and cpn = '1' then
-- synopsys translate_on
--@@ elsif cpn'event and cpn = '1' then

        current_state := next_state;
        end_of_cycle := false;
        write_protect := false;

        case current_state is
            when p1 =>
                next_state := p2;

                mcrp0n    <= '1' after reg_delay;
                mcrp1n    <= '1' after reg_delay;
                mcrp2n    <= '1' after reg_delay;
                mcrp3n    <= '1' after reg_delay;
                lmcrd0n   <= '1' after reg_delay;
                rmcrd0n   <= '1' after reg_delay;
                wpw       <= '0' after reg_delay;
                xp1       <= '1' after reg_delay;
                lban      <= '0' after reg_delay;
                baipn     <= '0' after reg_delay;
                pr6ckn    <= '0' after reg_delay;
                pwmoen    <= '1' after reg_delay;
                ram_enable <= '0' after reg_delay;

            when p2 =>
                next_state := p3;

                xp2       <= '1' after reg_delay;
                xp1       <= '0' after reg_delay;
                baipn     <= '1' after reg_delay;
                rpw       <= '0' after reg_delay;

            when p3 =>
                xp2 <= '0' after reg_delay;
                if clocked_reset = '1' then
                    current_cycle := startup_cycle;

                elsif pwm_address = '1' or force_pwm = '1' then
                    if read = '1' and write = '0' then
                        current_cycle := pwm_read;
                    elsif write = '1' then
```

```
            current_cycle := pwm_write;
            pwmoen <= '0' after reg_delay;
        else current_cycle := register_transfer;
        end if;

    elsif ram_address = '1' and read = '1' and write = '0' then
        current_cycle := ram_read;
        lmcrd0n <= '0' after reg_delay;
        rmcrd0n <= '0' after reg_delay;

    elsif ram_address = '1' and write = '1' then
        current_cycle := ram_write;
        lmcrd0n      <= not write_left  after reg_delay;
        rmcrd0n      <= not write_right after reg_delay;
        write_strobe <= '1'             after reg_delay;

    elsif ab00 = '1' and read = '1' then
        current_cycle := prom_read;
        bank := vld2int( ab01 & ab02 );
        case bank is
            when 0 => mcrp0n <= '0' after reg_delay;
            when 1 => mcrp1n <= '0' after reg_delay;
            when 2 => mcrp2n <= '0' after reg_delay;
            when 3 => mcrp3n <= '0' after reg_delay;
            when others => null;
        end case;

    elsif (shme = '1' and memory_cycle = '1') then
        current_cycle := shm_cycle;

    elsif dbs = vlbit_1d'("1101") then
        current_cycle := stretched;

    elsif alumode = '0' then
        current_cycle := cpu_alu;

    else
        current_cycle := register_transfer;
    end if;

    ram_enable <= ram_address after reg_delay;

    lban        <= '1' after reg_delay;
    baipn       <= '0' after reg_delay;

    if (current_cycle = pwm_read or current_cycle = pwm_write)
                and pwm_busy = '1' then
        next_state := current_state;

    elsif (current_cycle = shm_cycle) and latched_shm_grant = '0' the
n
        next_state := current_state;

    else        next_state := p4;
    end if;

when p4 =>
    if current_cycle = register_transfer then
        end_of_cycle := true;
        next_state := p1;
    else
        next_state := p5;
    end if;
```

```
            if current_cycle = pwm_read then
                rpw <= '1' after reg_delay;
            end if;

            if ab03 & ab04 & ab05 = vlbit_1d'("111") and mcio_enable = '0' an
d gie = '0'
                        and clocked_reset = '0' then
                write_protect := true;
            end if;

            if current_cycle = pwm_write and write_protect = false then
                wpw <= '1' after reg_delay;
            end if;
            baipn <= '1' after reg_delay;

        when p5 =>
            if current_cycle = cpu_alu or current_cycle = ram_write
                        or current_cycle = pwm_write then
                end_of_cycle := true;
                next_state := p1;
            else
                next_state := p6;
            end if;

            wpw          <= '0' after reg_delay;
            rpw          <= '0' after reg_delay;
            write_strobe <= '0' after reg_delay;
            pr6ckn       <= '1' after reg_delay;

        when p6 =>
            if current_cycle = prom_read or current_cycle = ram_read then
                end_of_cycle := true;
                next_state := p1;
            else
                next_state := p7;
            end if;

        when p7 =>
            if current_cycle = shm_cycle then
                next_state := p1;
                end_of_cycle := true;
            else
                next_state := p8;
            end if;

        when p8 =>
            if current_cycle = stretched then
                next_state := p1;
                end_of_cycle := true;
            else
                next_state := p9;
                if current_cycle = pwm_read then
                    pwmoen <= '0' after reg_delay;
                end if;
            end if;

        when p9 =>
            next_state := p10;

        when p10 =>
            next_state := p1;
            end_of_cycle := true;

        when reset =>
```

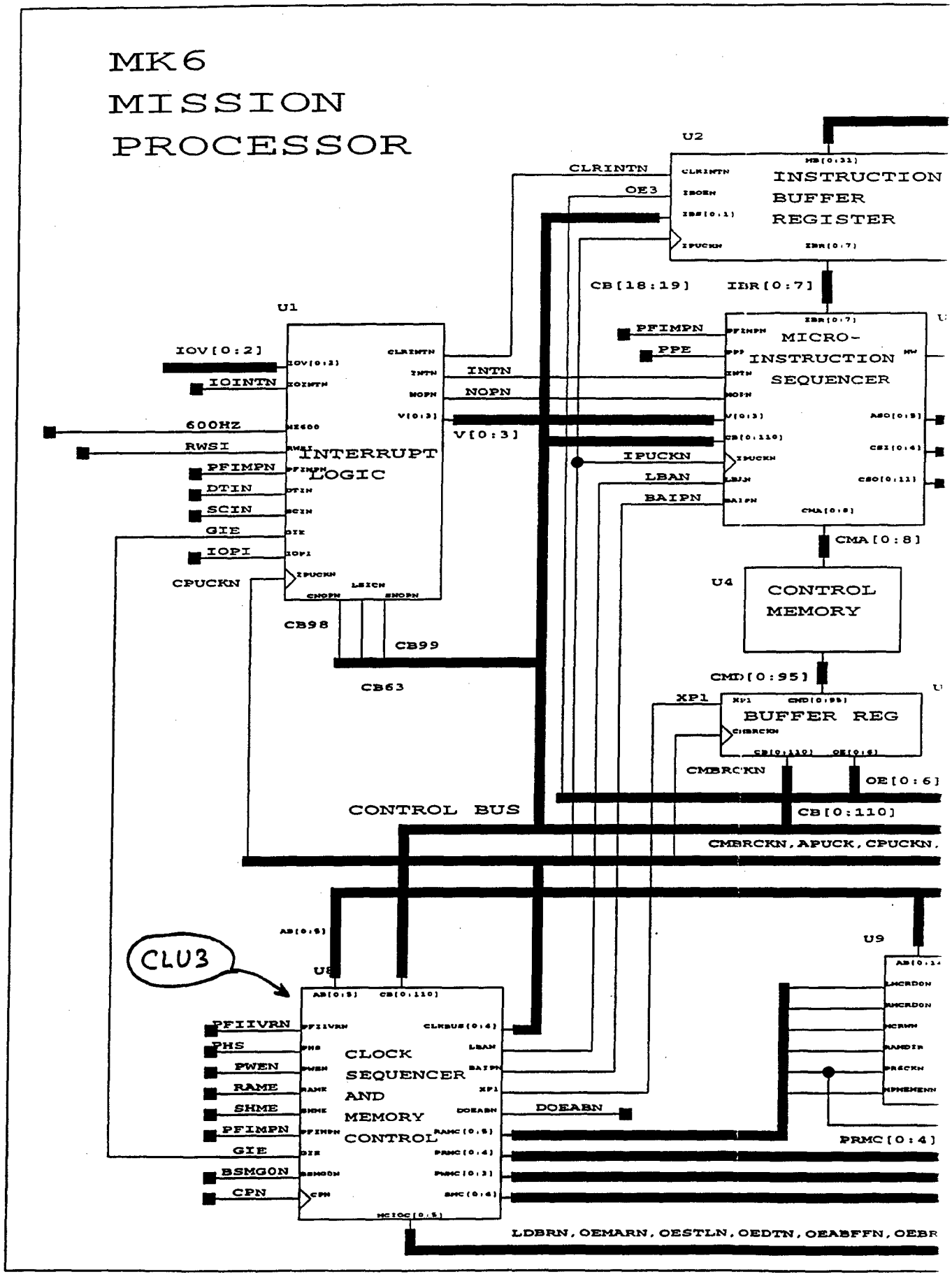
```
                next_state := p1;

            end case;

            if end_of_cycle then
                eoip <= '1' after reg_delay;
                if current_cycle /= startup_cycle then
                    register_clk <= '1' after reg_delay;
                end if;
            else
                eoip <= '0' after reg_delay;
                register_clk <= '0' after reg_delay;
            end if;
        end if;

end process cycle_state;
-------------------------------------
pwm_direction:process(pwmoen,xp2,read)
begin
        if xp2 = '1' and pwmoen = '1' then
            pwmdir <= read after reg_delay;
        end if;
end process;
-------------------------------------
ram_direction:process(ram_address,read,xp2,ram_enable)
begin
        if xp2 = '1' and ram_address = '1' and ram_enable = '0' then
            ramdir <= read after reg_delay;
        end if;
end process;
-------------------------------------

        apuck    <= register_clk;
        cmbrckn  <= not eoip;
        eoipn    <= not eoip;
        ipuckn   <= not register_clk;
        eoickn   <= not register_clk;
        cpuckn   <= not register_clk;
        prmclrn  <= not xp1;
        mpmemenn <= not memory_cycle;
        mcrwn    <= not write_strobe;
        bsmioenn <= not shm_enable;
        bsmr0n   <= not latched_shm_request;
        smsexnn  <= not (shm_enable and read and not xp1);
        mpsmwen  <= not (not xp1 and write and eoipn and (not clocked_reset and eoipn
       ));

end behavior;
-------------------------------------
```

LOGIC SYNTHESIS SCRIPT (SYNOPSYS)

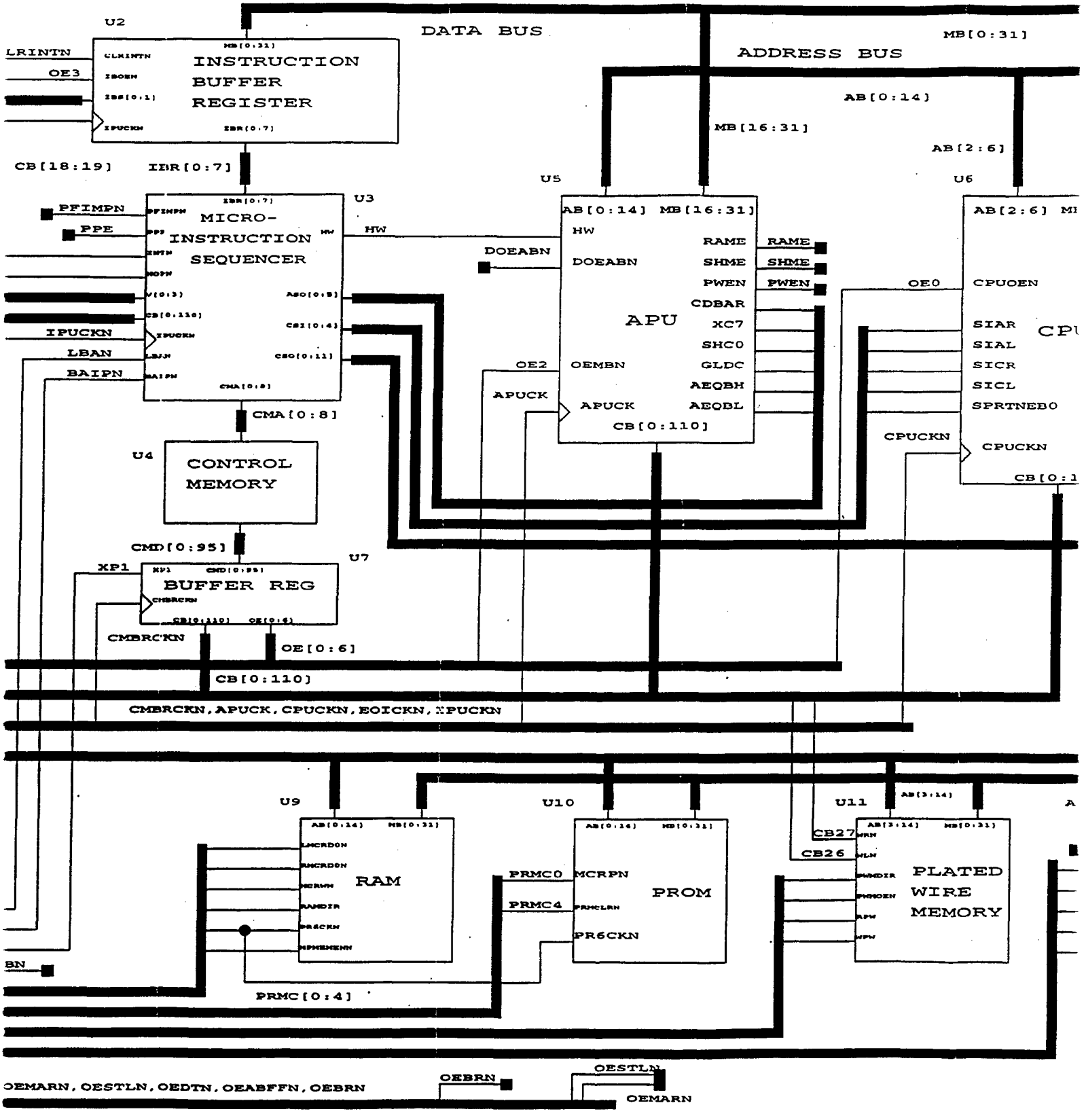
```
/*  SYNOPSYS 2.0 SCRIPT FOR CREATING CLU3 DESIGN ON ACTEL 1240 FPGA
    D. MCGORTY  1/22/92                                          */

designer = "Dave McGorty ";
company = "C.S. Draper Labs";
default_schematic_options = "-size D"

/*  READ PACKAGE FILES WHICH RECOGNIZE VIEWLOGIC VHDL FUNCTIONS          */

hdlin_files = "(~/synop/vlbit.vhd ~/synop/vlmath.vhd )"

/*  SET DEFAULT TO ACTEL ACT2 LIBRARY                          */

search_path = ". /external/draper/actel/lib /external/synopsys/libraries "
link_library = "/external/draper/actel/lib/act2_20.db "
target_library = "/external/draper/actel/lib/act2_20.db "
symbol_library = "/external/draper/actel/lib/act2.sdb "

/*  DONT USE NEGATIVE EDGE TRIGGERED D-FLOPS.  THIS PREVENTS THE COMPILER FROM
    ADDING GATES TO BUFFER OUTPUT OF THE CLKBUF BUFFER.  SYNOPSYS DOES NOT
    RECOGNIZE THE FACT THAT ONLY SPECIAL GATES CAN INPUT THE CLKBUF OUTPUT.   */

dont_use { act2/DF1B, act2/DF1C, act2/DFC1A, act2/DFC1D, act2/DFC1G }

/*  SET UP REQUIRED EDIF VARIABLES FOR EVENTUAL READING BY VIEWLOGIC EDIF2VL2
    EDIF READER                                                  */

edifout_no_array = "true"
edifout_netlist_only="false"
edifout_external="true"
edifout_power_and_ground_representation = "cell";

/*  REMOVE SPECIAL COMMENT STRINGS IN THE VIEWLOGIC VHDL TO ENABLE
    SYNOPSYS-SPECIFIC STATEMENTS                                 */

sh trans -x ~/synop/vl2synop.dic -i ~/workview/mp/behv/clu3synth.vhd -o temp_synth.vh
      d

/*  READ/COMPILE VHDL FOR DESIGN MINUS I/O BUFFERS                */

read -format vhdl temp_synth.vhd
create_schematic -size D -no_schematic -no_symbol_view -no_hierarchy_view -gen_databa
      se
compile -no_map
create_schematic -size D -no_schematic -no_symbol_view -no_hierarchy_view -gen_databa
      se

/*  EXTRACT AS A FINITE STATE MACHINE THE REGISTER CORRESPONDING TO THE "NEXT_STATE"
    STATE VARIABLE AND ITS ASSOCIATED LOGIC. USE ONE-HOT STYLE FOR MAX SPEED    */

set_fsm_minimize true
set_fsm_encoding_style one_hot
set_fsm_state_vector {next_state_reg[0],next_state_reg[1],next_state_reg[2],next_stat
      e_reg[3]}
group -fsm -design FSM1
create_schematic -size D -no_schematic -no_symbol_view -no_hierarchy_view -gen_databa
      se
current_design = FSM1
create_schematic -size D -no_schematic -no_symbol_view -no_hierarchy_view -gen_databa
      se
extract
create_schematic -size D -no_schematic -no_symbol_view -no_hierarchy_view -gen_databa
      se
```

```
set_fsm_minimize true
reduce_fsm

/*  CONSTRAIN COMPILER FOR MAX CLOCK FREQUENCY                    */

create_clock cpn
max_period 20 cpn

/*  ALLOW INFINITE FANOUT TO PREVENT CLOCK BUFFERING              */

set_max_fanout 200 current_design
compile
create_schematic -size D -no_schematic -no_symbol_view -no_hierarchy_view
create_schematic -size D -no_bus

/*  FLATTEN FINITE STATE MACHINE INTO REST OF DESIGN              */

current_design = csr1
create_schematic -size D -no_schematic -no_symbol_view -no_hierarchy_view
create_clock cpn
max_period 20 cpn
compile -ungroup_all
create_schematic -size D -no_schematic -no_symbol_view -no_hierarchy_view
create_schematic -size D -no_bus
set_arrival -max -50 pfimpn
set_arrival -max -50 clocked_reset_reg/QN
set_arrival -max -50 sync_reset_reg/QN
set_arrival -rise 0 cpn
set_arrival -fall 10 cpn
current_design = csr1

/*  ACTEL RECOMMENDS MAXIMUM FANOUT OF 10 FOR ALL EXCEPT CLKBUF   */

set_max_fanout 10 current_design
dont_touch cpn
create_schematic -size D -no_schematic -no_symbol_view -no_hierarchy_view

/*  TRY AS HARD AS POSSIBLE TO DO A GOOD JOB MAPPING              */

compile -map_effort high
create_schematic -size D -no_bus

/*  READ VHDL MODEL WITH I/O BUFFERS INSTANTIATED, AND FLATTEN    */

read -f vhdl -/workview/csrclu3/clu3io.vhd
compile -incremental_mapping -ungroup_all
create_schematic -size D -no_bus
sh rm temp_synth.vhd

/*  WRITE EDIF SCHEMATIC FORMAT                                   */

write -f edif -o clu3io.edif

/*  SAVE AS SYNOPSYS DB FORMAT                                    */

write -f db -output temp_clu3io.db
```

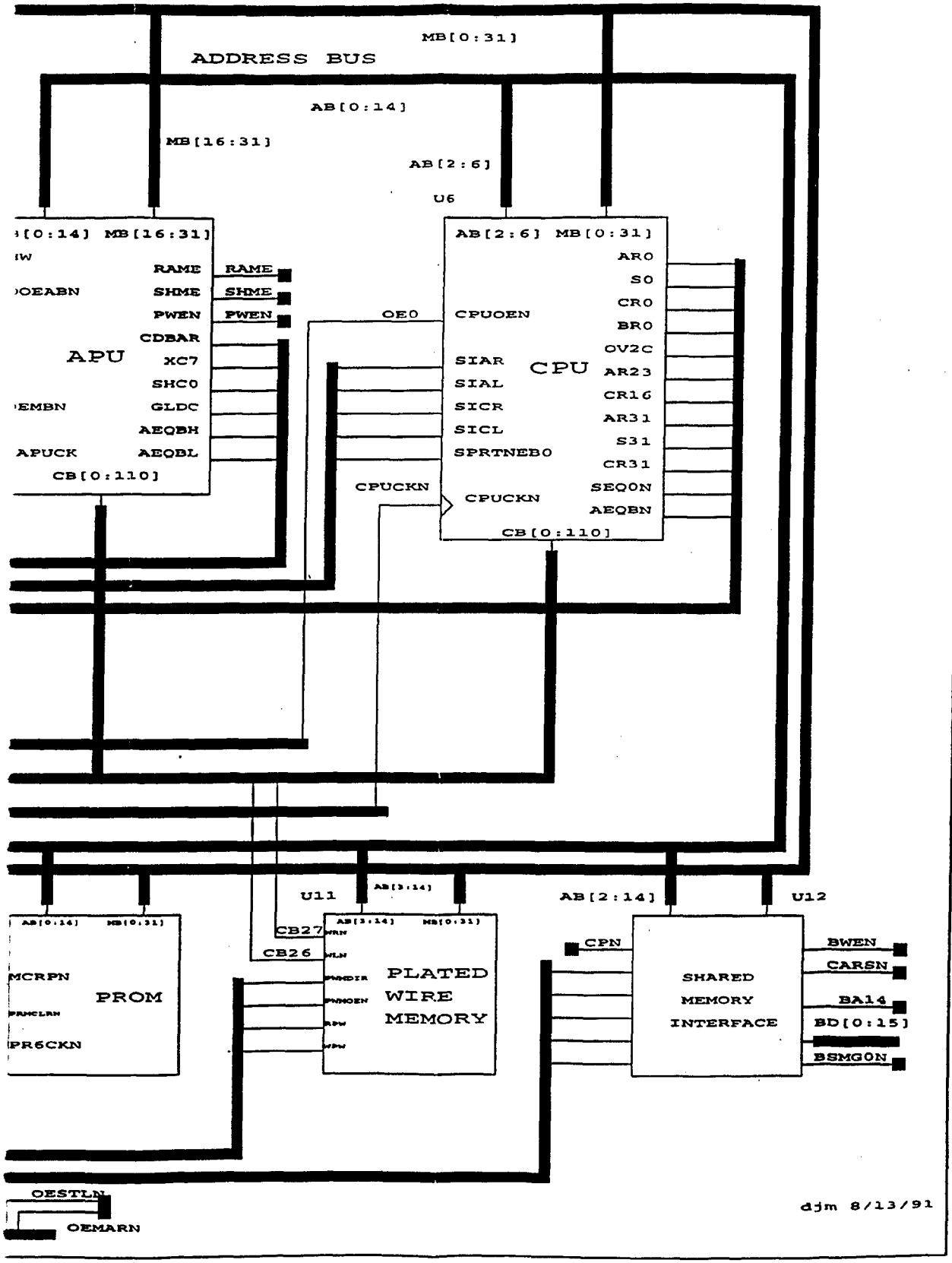
SIMULATION TEST BENCH TOP-LEVEL DIAGRAM


MK6
MISSION
PROCESSOR
U2
INSTRUCTION
BUFFER
REGISTER
CLRINTN
OE3
CB[18:19]
IBR[0:7]
U1
INTERRUPT
LOGIC
IOV[0:2]
IOINTN
600HZ
RWSI
PFIMPN
DTIN
SCIN
GIE
IOPI
CPUCKN
INTN
NOPN
V[0:3]
CB98
CB99
CB63
MICRO-
INSTRUCTION
SEQUENCER
PPE
IPUCKN
LBAN
BAIPN
CMA[0:8]
U4
CONTROL
MEMORY
CMD[0:95]
XP1
BUFFER REG
CMBRCKN
OE[0:6]
CB[0:110]
CONTROL BUS
CMBRCKN, APUCK, CPUCKN,
CLU3
U8
U9
CLOCK
SEQUENCER
AND
MEMORY
CONTROL
PFIIVRN
PHS
PWEN
RAME
SHME
BSMGON
CPN
DOEABN
PRMC[0:4]
LDBRN, OEMARN, OESTLN, OEDTN, OEABFFN, OEBR

Simulation Test Bench



Simulation Test Bench



(3)

FPGA DESIGN FLOW

Design Flow

Write VHDL models for ASIC and test-bed

Functional Simulation

Logic Synthesis

Static Timing Analysis

Gate-Level Simulation

FPGA Programming

FPGA Verification in Hardware Modeler

DRAPER LABORATORY