THE RISC PROCESSOR MODULE FOR FASTBUS COMPUTATION APPLICATIONS

The Results of a Phase II SBIR Grant

Scientific Systems International, Ltd.*

Work performed under the auspices
of the U.S. Department of Energy
SBIR Contract DE-AC02-88ER-80599

*3491 B Trinity Drive
Los Alamos, New Mexico
(505) 662-6712

TABLE OF CONTENTS

THE RISC PROCESSOR MODULE
FOR
FASTBUS COMPUTATION APPLICATIONS
FINAL TECHNICAL REPORT

1.0 Overview

The FASTBUS system specification for high-energy physics and other data-system applications anticipates the use of multiple, high-performance processor modules for data control and event reduction associated with experiments in the physical sciences. Existing processor designs will be unable to cope with the projected data-reduction and event-handling requirements of the complex experiments planned for the next generation of particle accelerators. Data-handling strategies for experimental physics are evolving from systems based upon a single central computer to those with arrays of high-speed, sophisticated, front-end processing elements. The advent of accelerators such as LEP and LHC, and beyond, is forcing the architecture of these processors toward the simpler RISC designs to enhance both speed and the software-development issue.

This report describes the prototype development of a FASTBUS RISC Processor Module (FRPM) for use as a standard processing element in FASTBUS data-acquisition systems under a Phase II SBIR grant through the U. S. Department of Energy, Division of Energy Research. The FRPM hosts a reduced instruction set computer--the SPARCengine-2 by Sun Microcomputer Systems, Inc.--capable of executing 4.2 million floating point instructions per second with a clock of up to 40 MHz.. The prototype FRPM supports a port to the FASTBUS crate segment by way of a standard-logic interface. The FRPM processor operates under a commercially available real-time operating system, and application software can be developed on workstation and mainframe computer systems.

We further cover the chronology of the Phase II work, a discussion of the objectives, and our experiences with an ASIC manufacturer in attempting to complete the fabrication of a chip implementing the FASTBUS Master Interface (FMI).

The appendices of this report contain the functional description and design details of the FRPM and of the FMI chip design.

2.0 Chronology of the Contract

The SBIR Phase II portion of the Department of Energy contract DE-AC02-88ER8059—The RISC Processor Module for FASTBUS Computation Applications—FRPM was approved to begin on May 19, 1989; a letter to our technical contract administrator requested that effort on Phase II begin on September 1, 1989. Design effort began at that time on both the FASTBUS Master Interface (FMI) and the processor support logic. Our re-evaluation of processors for the FRPM was completed in late May 1990 and the development work continues with the Sun Microsystems "SPARCengine" processor in place of the BIT chipset. Development of the FMI gate array design was delayed during our search for an alternate ASIC manufacturer, which concluded in late December 1990. We selected Applied Microcircuits Corporation (AMCC) to be the fabrication house for the FMI gate array, allowing development work to continue and an extension to the contract through December 31, 1991 was granted due to time slips associated with both the ASIC and the RISC processor. A further request for extension to December 31, 1992 was granted in order to complete the FMI gate array portion of the contract, which must be in place before the remainder of the contract work can be completed. AMCC, the ASIC prototype manufacturer, canceled the contract with SSI in late October 1993, which prompted a request for modification of the original contract and extension through December 31, 1993 to complete a new Scope of Work. Following the submission of the design package for a gate-array design in another Phase II effort (87ER80454), AMCC terminated our prototype fabrication contracts for the that array and for the FASTBUS Master Interface (FMI) design. In actual fact, AMCC unilaterally canceled our contract and refunded most of the NRE paid ($25,400.00) without giving us an opportunity to respond. The scope of work of the contract was modified to reflect the changes forces upon us by the ASIC manufacturer's failure to carry our their part of our agreement. The modified FRPM design supports a standard-logic interface (completed in July 1993) from the SparcEngine to the FASTBUS. Control of the FRPM is accomplished by software control. The prototype FRPM was assembled and hardware/software tested during the final 3-month period of the contract. As of 12/31/93, all items in the agreed-upon modification to the contract Scope of Work were completed.

2.1 Discussion of Objectives

Because of the unilateral action by the ASIC manufacturer (discussed in 2.0 and 2.2), we were forced to propose a modified set of objectives for the Phase II effort in December 1992. We discuss these objectives below in relation of the completed prototype FASTBUS module.

6.2.1 Develop an FRPM prototype design based on the conceptual design from the Phase I study and the effort through 12/92 in Phase II. Select the processor structure and design interface support logic to the FASTBUS system using standard logic elements for prototype module.

This modified objective was completed on 6/30/93. It reflected the fact that the processing element originally selected never made it to market and was dropped by Sun Microsystems as a potential processor for their SPARCengines. Following this, we selected the current SPARCengine as the FRPM computing element. The standard-logic interface was dictated by the cancellation of our FMI ASIC development effort by Advanced Micro Circuits Corporation in late October 1992. The modified specification and the original specification for the FRPM are contained in Appendix A.

6.2.2 Expand the behavioral model developed in Phase I for use as a operational standard in the simulation of the prototype FMI design. Develop a comprehensive set of test vectors for the FMI model.

We eliminated this objective in the modified scope of work, however the behavioral model and test vector set were actually completed by the end of the FMI design. The specification for the FMI is contained in Appendix C; the FMI design contained in Appendix E was derived from the specification and a comprehensive behavioral model, and was in the process of being tested against test vectors that had been developed with the model when the agreement was canceled.

6.2.3 Prepare the preliminary operations and testing manual for the FRPM.

The operations and testing manual was competed with the final testing of the prototype FRPM on 12/30/93 and is contained in Appendix A along with the FRPM specifications.

6.2.4 Select and evaluate potential sources of a real-time operating system software package.

The modification of the scope of work following the ASIC manufacturer cancellation of our FMI development agreement had no detrimental effect on this objective other that delaying the completion somewhat. Several suppliers of real-time operating system software were investigated during

the work on this objective. We were looking for a potential manufacturer who was familiar to the experimental physics community while having a product that would address the various important aspects of real-time systems. These were multitasking capability, effective scheduling of tasks to be run, context switching with fast response, software settable priority levels, semaphore signaling between tasks, and interrupt handling that was fast and deterministic. Other attributes, such as compatibility with the new POSIX standard, UNIX structure, and networking support. We found that the VxWorks product by Wind River Systems, Inc. meet these requirements and was selected for evaluation. This objective was completed with the final testing of the FRPM on 12/30/93.

6.2.5 Evaluate potential sources of application-software tools.

This objective was eliminated when we dropped the original BIT processor in favor of the SUN SPARCengine (see 2.0). The SUN development tools available would quite well fulfill this the requirements of this objective with no further investigating—we adopted the SUN tools and other commercially available tools from 3rd. party manufacturers for this part of the development.

6.2.6 Select the ASIC vendor and negotiate a design agreement to allow CAE workstation design, capture, simulation, vendor place/route, test-vector simulation, and prototype-part fabrication. Convert the expanded FMI model to an ASIC and PAL design using CAE libraries. Compare the prototype design to the behavioral model using the test vectors generated in 6.2.2.

This objective was eliminated when the ASIC manufacturer canceled our FMI development agreement, however, we had selected and negotiated the required design agreement that would allow the specified tasks; we had converted the behavioral model into an ASIC and PAL design and were in the process of testing this design when the agreement was canceled. Appendix D discusses the simulation techniques used during this part of the FRPM development.

6.2.7 Prepare printed-circuit negatives for construction of a FASTBUS RISC Processor Module in multilayer technology. Prepare two multilayer printed-wiring boards for further evaluation.

The standard-interface FRPM design was converted to multilayer printed-wiring boards by 9/30.93 by a local layout company and a printed-circuit fabrication facility in Phoenix, AZ. Fabrication problems determined during check-out of the boards required a second pass through fabrication to complete this objective.

6.2.8 Assemble, test, and evaluate the printed-circuit prototype modules of the FRPM. Prepare the final operations and testing manual.

The multilayer printed-wiring boards were assembled and tested during the final 3 months of the Phase II effort. We were able to interface the SUN SPARCengine SBus I/O to the FRPM by was of a Motorola SBus interface chip on an SBus I/O board. The Motorola chip allows a variety of timing and I/O signal characteristics to be controlled directly by software. The FRPM, containing the SPARCengine, SBus I/O board, and the standard-logic interface was powered in a FASTBUS crate in our FASTBUS test stand. In addition, the SUN Open Boot Prom, resident in the SPARCengine, contains a full FORTH interpreter for I/O testing. With the help of FORTH code, we were able to fully test the FRPM interface and characterize the design for evaluation, while the real-time software package (VxWorks by Wind River Systems) was tested with the aid of a SUN server/workstation connected to the FRPM by an Ethernet link. Our design proved to be quite robust and required only a few changes to have a fully functioning system. Final development of the commercial version of FRPM will continue in Phase III of the contract to be funded by Scientific Systems International, Ltd.

2.2 Actions by ASIC Manufacturers

Early in the Phase II effort, we had selected Raytheon Company, Semiconductor Division as the ASIC manufacturer based upon another Phase II (87ER80454) selection. Raytheon was unable to complete our 80454 agreement in December 1991 and we evaluated, and selected Applied MicroCircuits Corporation (AMCC) as the ASIC manufacturer for both Phase II ASIC developments (80454 and 80599). In late October 1992, AMCC elected to shutdown their BiCMOS product that we were using and unilaterally canceled our FMI development agreement, thus making it impossible to complete the FMI ASIC development and prompting the modification of the Phase II work in an attempt to complete a version of the FRPM.

While the semiconductor companies were initially very anxious to accept our designs, we found that internal problems with the technologies and the business outlook for ASICs in general made small business, and especially small business serving the scientific community, expendable. Our recommendation, based upon these devastating experiences, is that the physics community and small technology-businesses serving this community should be very wary of attempting to use any of the faster, more complicated integrated-circuit technologies.

Within the physics community, there have been some successes with CMOS custom and semicustom designs, and one example of a successful small ECL ASIC, but large (>20,000 gate) ECL or BiCMOS designs are rare to non-existent. Unless the potential market is very large (e.g., 100,000 units per year fabricated), even the government does not have a good chance of being able to complete a design. We further found, in the case of Raytheon Company, that the tools available to the company were out of date and quite antiquated. Their eventual exit from the ASIC business could be a reason why their design tools were poor. In the case of AMCC, the design tools were adequate, but the company seemed to be unable to meet specifications with the BiCMOS technology and were actually moving to a bipolar ECL technology when they canceled our design agreement. In both cases, the small size of the potential FASTBUS market was a deciding factor to drop our agreements.

2.3 Organization of the Report

The prototype FRPM is fully tested, evaluated, and operational and the designs for this device are contained in Appendix A (specifications and Operations Manual), Appendix B (Standard-logic TTL interface schematics), and Appendix F (advertising flyer for the FRPM distributed at the 1993 IEEE Nuclear Science Symposium). For reference, the FMI design is also contained in Appendix C (specifications), Appendix D (simulation techniques), and Appendix E (schematics for the AMCC BiCMOS array design).

APPENDIX A

FRPM SPECIFICATIONS

&

OPERATIONS MANUAL

INTERFACE DETAILS AND OPERATIONS MANUAL

SBus to FASTBUS MASTER INTERFACE for FRPM

GENERAL

Physically, the FRPM consists of a SUN Microsystems Inc. "SPARCengine" processor mounted in the front section of a 3-wide FASTBUS module, leaving the remaining rear 1/3 of the FASTBUS board-space available for FASTBUS master interface logic. Additional hardware space may be gained by taking advantage of the 3-wide module board-space behind the SPARCengine.

Through SPARCengine connectors, the user may add temporary SCSI devices and hard or floppy disc controllers for test, diagnostic or configuration purposes. Communication with the FRPM is via the on-board SPARCengine Ethernet port to other system processors. Alternatively, limited communications with the FRPM can be accomplished via the FASTBUS port. The VxWorks Real Time Operating System (RTOS) from Wind River Systems has been ported to the FRPM to allow easier software development and operation within an experimental environment.

The interface from a SUN SPARCengine to the FASTBUS data system master interface hardware is accomplished by a Motorola MC92005 SBus Slave Interface Controller (SLIC) chip. The chip, which interfaces SBus to a multi-ported private bus, or PBus, is a 160 pin quad flat pack CMOS ASIC produced by Motorola Inc. and meets the Rev. B.0 specification for the SBus interface standard (IEEE P1496). The interface is presently contained on a DAWN VME Products SLIC Evaluation Board with ribbon cable connection to the FRPM board.

INTERFACE PHILOSOPHY

Interface to FASTBUS is accomplished by **primitive operations** rather that **channel control** operations. Arbitration, Address and Data Cycle functions are handled by separate I/O operations. All time-out functions are cpu implemented, as are the basic block transfer and pipe-line transfer operations. The interface is not autonomous; there are no buffer or list memories available other than two 16 word, 32-bit registers in the SLIC interface for program use if desired. Interrupts are used with SS $\neq$ 0, SR, AK = 1, and DK(t), if enabled. Other information is available in the interface status register. The philosophy of the interface is "simple", with all data and control available to the programmer, in contrast to a more autonomous device with little to no data/control available.

INTERFACE DETAILS

The SLIC chip provides 6 interface ports with separate selects and one FCODE prom port (required by SBus spec). Internal chip control registers are handled by the last available port; each port has a 128kbytes of address space available if needed.
A single-wide SBus interface board--containing the SLIC chip, the FCODE prom and ribbon cable connectors--will mount in a SPARCengine SBus slot. Ribbon cable from the SBus board connects 60 port 1 PBus interface signals, with commons, to the master interface logic board. The PBus connection path is kept short to minimize reflections and delay electrical problems; the SBus connection path is kept **very** short and is localized to the SLIC chip only.

The FRPM master interface logic is controlled by port 1 only, register addresses 0-5 as described below:

Port 1,Register 0--Status Register:		R/W	R/O
Bits 0-3	CSR #0, bits 0, 1, 2, 14	*	
Bits 4-7	CSR #9, bits 4, 5, 6, 7	*	
Bit 8	I/O Reset-clears GK & Data cycle stuff		*
Bit 9	RB-holds GK as long as RB=1	*	
Bit 10	RB from FB		*
Bit 11	BH from FB		*
Bit 13	SS ≠ 0 interrupt enable/disable	* IRQ0	
Bit 14	SR interrupt enable/disable	* IRQ1	
Bit 15	AK =1/DK(t) interrupt enable/disable		*
IRQ2			
Bit 16	SBus interrupt enable/disable	*	
Bits 17	Clear IRQ0 (0 clears IRQ)		w/o
Bit 18	Clear IRQ1 (1 sets IRQ for test)		w/o
Bit 19	Clear IRQ2 (no readback)		w/o
Bit 20	Set PE = 1 on data cycles	*	
Bits 21-23	SS(0:2)-current cycle		*
Bit 24	WT-current cycle		*
Bit 25	AK-current cycle		*
Bit 26	DK-current cycle		*
Bit 27	SR-from FASTBUS		*
Bit 28	Parity Error gated with PE on current read cycle *		
Bit 29	Slave port active		*
Bit 30	Data cycle in progress		*
Bit 31	Master interface is FASTBUS current master		*

Port 1, Register 1--Arbitration Control:

Bits 0-7	CSR #8--Bits 0-5 = AL(0:5), Bit 6 = PRIA, Bit 7 = IAI
Bit 8-13	0
Bit 15	1 = arbitrate, 0 = drop GK and current master

FASTBUS rules for arbitration hold. The FRPM must have arbitrated before executing a FB primitive operation control. GK is held until mastership is dropped. The holding register does not clear with SBUS or I/O Reset commands and must be written to clear. PRIA is priority access and IAI is assured access, as described in the FASTBUS Standard specification.

Port 1, Register 2--Address Cycle Control

Bits 0-2	MS(0:2)
Bit 3	EG--1=> Port 1, Reg 3 = GA, otherwise =logical address
Bit 4	Parity odd = 1, even = 0 (testing purposes)
Bit 5	
Bit 6	AK-from FASTBUS (R/O)
Bit 7	Address cycle = 1, Drop AS & terminate = 0

Time-outs done by cpu as are status checks. Poll Status Register for AK, or use AK = 1 interrupt to determine if the address cycle was responded to by the FASTBUS slave addressed. The FRPM must be current master before register will accept write command.

Port 1, Register 3--Address Register

Bits 0 - 31 FASTBUS address 0 - 31

This register is R/W by PBus. Driver software must load Register 3 prior to executing Address Cycle Control, as last address written is held. Interpretation of Register 3 contents is based on the EG bit in Register 2. If EG is set to 1, the contents of Register 3 is a geographical address. If EG is set to 0, the contents of Register 3 is a logical address.

Port 1, Register 4--Data Cycle Control

Bits 0-2	MS(0:2)
Bit 3	RD
Bit 4	Parity odd = 1, Parity even = 0 (testing purposes)
Bit 5	
Bit 6	DK-from FASTBUS (R/O)
Bit 7	Data Cycle =1. Terminate = 0

The written value of MS determines the action of bit 7. For MS=0 or 2, the data cycle will terminate when bit 7 -> 0. For MS=1 or 3, DS will be toggled when bit 7 -> 0 or 1. To complete a BT sequence MS must be set to 0 when bit 7 -> 0. DK polled in Status Register, register 4, or use DK(t) interrupt. Data time-outs are fielded by the cpu. BT and PL control by cpu. FRPM must be current master before register will accept write command.

Port 1, Register 5--Data Register

Bits 0-31 AD(0:31)

R/W by PBus, R/W by FBus. Register 5 must be loaded before a FB write operation, and must be read before the next read operation. An SBus write of REG 5 followed by an SBus read of REG 5 will read the holding register. Following a FASTBUS read cycle, an SBus REG 5 read will read the latched FASTBUS AD lines, and will do so until the next SBus write of REG 5 in preparation for a FASTBUS write operation. Thus, REGISTER 5 is made up of 2 physical registers, one to hold write data and one to accept read data. Which register to read by an SBus operation is dictated by the FASTBUS operation just completed.

Slave Port

The FRPM contains a minimal slave port that may be addressed only geographically to CSR space only by another FASTBUS master, or by the FRPM for testing purposes, for example. The slave port will accept NTA = 0, 1, 8, and 9. A bad NTA will generate SS=7, as will a parity error. MS=1 for address operations is the only code accepted, and MS=0 or 2; other MS codes will result in SS=6 returned to the controlling master.

The information returned by a CSR0 read are bits 0, 1, 2, and 14. Bits 0, 1, 2, 14, 16, 17, 18 and 30 can be written, as described in the FASTBUS Standard specification. The FRPM module ID is 10D6 (hex) contained in the upper 16 bits of the 32 bit word returned by a CSR0 read. The contents of REGISTER 1 are read with a CSR8 read is executed and CSR9-4 to 7 is read with a CSR9 read.

FASTBUS RISC PROCESSOR MODULE SPECIFICATIONS AND FUNCTIONAL REQUIRMENTS

ORIGINAL DOCUMENTATION PRIOR TO MODIFICATION OF SCOPE-OF-WORK

1. Introduction

The FASTBUS RISC Processor Module (FRPM) shall be a single or double width FASTBUS module supporting a RISC processor architecture. The Module shall also support conventional input-output (I/O) devices including ethernet for local area networks (LAN), RS-232 ports for data terminals, small computer system interface (SCSI) ports for disk storage, and several front-panel signals for general input and output control purposes.

The FRPM shall also be both a FASTBUS master and slave device, and control two FASTBUS I/O ports. One shall connect to the crate segment and the other to the auxiliary connector port.

1.1. Scope

This document identifies the functional capabilities and interface features of the FRPM. These requirements, resulting from studies performed during the Phase I research for the FASTBUS RISC Processor Module, represent compromises made between the set of all desirable features for FASTBUS processor modules, and the set of feasible implementations. While these requirements are not inflexible, they are established only after thorough research to direct and limit subsequent design and implementation.

1.2. Functional Summary

1.2.1. Processor

The RISC processor shall be a 32-bit ECL device. It shall provide full arithmetic operations for integer and floating point operations using an internal capability or an external floating point coprocessor. It shall be compatible with a memory mapped architecture for I/O and I/O control. The processor shall provide external interrupts, and shall expedite rapid context switching through use of many switchable internal register files. Performance shall exceed 40 MIPS on a scale where a VAX 11/780[1] is rated as a 1 MIP machine.

---

[1] Digital Equipment Corporation

1.2.2. FASTBUS Master Interface

The FASTBUS Master Interface (FMI) shall implement a bus and a processor interface. The bus interface shall allow direct connection to the FASTBUS crate segment, and shall support all the protocol defined by IEEE 960 for master devices. The processor interface shall provide a single 64-bit path to the FMI that may be used in two modes. Seven other ancillary signals shall be provided to select the interface operation and provide timing controls. For the first mode of control, the interface path shall enable the processor to control each operation executed by the FMI. In this slaved mode, the FMI is totally dependent on the processor for control. The processor shall optionally supply data embedded in the control stream. For the second mode for control of the FMI, the interface path allows the FMI to fetch equivalent control and data information from a dedicated memory through a direct memory access (DMA) controller built into the FMI. The latter mode of operation permits the FMI to operate autonomously once triggered, and provides a high performance FASTBUS sequencer controlled by predefined lists of primitive FASTBUS operations stored in shared processor memory. Both modes shall use the list sequencer to deposit input data in processor memory.

1.3. Assumptions and Constraints

The address space for FASTBUS is potentially so large that the FRPM shall not attempt to map FASTBUS address space into the processor address space. An arbitrary FASTBUS module may contain 2^{32} addresses for each of data space and control space.

The memory and memory interface architecture shall not prevent the FRPM from executing at the maximum speed permitted by the FASTBUS specification, IEEE 960. Therefore, a built-in list sequencer shall be provided within the FMI to achieve satisfactory performance. Furthermore, the associated memory architecture should be optimized to minimize contention between the processor and the DMA.

All addresses generated by the FRPM shall be absolute addresses referencing physical memory. No address translation shall be provided.

2. Applicable Documents

2.1. FASTBUS Documents

2.1.1. Hardware

IEEE Standard FASTBUS Modular High-Speed Data Acquisition and Control System, ANSI/IEEE Std 960-1986.

2.1.2. Software

FASTBUS Standard Routines, U. S. NIM Committee, May 1988.

2.2. FMI Requirements Specification

"FASTBUS Master Interface Specification and Functional Requirements," Appendix B of this document.

3. Functional Requirements

3.1. Processor Requirements

3.1.1. RISC Engine

3.1.1.1. Architecture

3.1.1.1.1. Harvard

To relieve congestion at the memory interface, processors supporting the Harvard architecture (separate data and instruction memories) shall be preferred.

3.1.1.1.2. Instruction/Data Cache

To reduce the number of memory fetches and optimize processor performance, processors supporting on-chip caching shall be preferred.

The FRPM shall implement memory with high-speed static RAM, and shall not implement any on-board cache.

3.1.1.2. Integer Operations

Integer operations shall be an integral part of the RISC processor or shall be available as a standard option within the family of microprocessor support chips. Addition, subtraction, multiplication, and division shall be supported. Multiplication shall produce a 64-bit product. Division shall use a 32-bit divisor and a 64-bit dividend to produce a 32-bit quotient.

3.1.1.3. Floating Point Operations

Floating point operations shall be an integral part of the RISC processor or shall be available as a standard option within the family of microprocessor support chips. Both 32-bit and 64-bit floating point operands shall be supported for addition, subtraction, multiplication, and division operations.

3.1.1.4. Optional Coprocessors

The processor shall support an optional application specific coprocessor. This coprocessor may effectively extend the instruction set of the processor to include functions that are optimized for the application.

3.1.1.5. Register Sets

The processor shall have a minimum of 32 general purpose registers. Processors with larger numbers of registers and with register management algorithms optimized for rapid context switching and subroutining are preferred.

3.1.1.6. Interrupts

The processor or floating point coprocessor shall generate internal interrupts for errors including integer overflow, divide by zero, exponent overflow, and exponent underflow.

Two external interrupts shall be connected from the front panel to the processor, and three interrupts shall be connected from the FMI to the processor.

3.1.1.7. Performance

3.1.1.7.1. Absolute

Performance shall exceed 40 MIPS on a scale where a VAX 11/780 is rated as a 1 MIPS machine.

3.1.1.7.2. Interrupt Response Time

The processor shall responsd to an interrupt in ≤ 10 μs if the processor is not currently active at the same or higher interrupt level.

3.1.1.7.3. Growth Path

The processor shall be selected from a family of processors that has planned growth plan which includes a 100 MIPS ECL processor.

3.1.2. Memory

3.1.2.1. Architecture

3.1.2.1.1. Memory Size

The FRPM program memory shall contain two megabytes of static random access semiconductor storage (SRAM). All memory shall be directly addressable from the controlling processor.

3.1.2.1.2. Expansion

The program memory shall be expandable to at least four megabytes, with no limitations on the direct addressability of the memory.

3.1.2.1.3. Word Size

The program memory word size shall be 32 bits, or greater. Additional data-path width shall be in increments of 8 bits.

3.1.2.1.4. Configuration

The program memory configuration is dictated by the semiconductor industry, however, no limitation shall be placed on the number of memory cells contained within a single memory integrated circuit. Basic configurations of 256 K x 1, 1 M x 1, etc., are equally acceptable.

3.1.2.2. Technology

The process technology used for the SRAM devices shall match that of the controlling processor in speed and signaling levels.

3.1.2.3. Access Time

The access time of the program memory shall match the cycle time of the controlling processor, within the limits of available SRAM devices.

3.1.2.4. Management

Memory management shall be limited to address-bounds checking for application programs. The controlling processor shall be signaled on detection of an out-of-bounds condition during memory accesses.

3.1.2.5. FMI List-Sequencer Memory

The FMI list-sequencer memory shall be a subset of the FRPM program memory containing at least 1/2 Mbyte storage capacity. The memory section shall be accessed by both the controlling processor and the FASTBUS interface.

There shall be a list-sequencer memory associated with each FMI.

3.1.2.6. Table Memory

The table, or calibration data-base memory, shall consist of dynamic random access semiconductor memory devices (DRAM) configured as 1 M x 1 or 4 M x 1 integrated circuits. The memory size shall be 1 megabyte, or greater, and the word size shall be identical to that of the program memory.

3.1.2.7. Multiple Access

The program memory, and list-sequencer memories associated with each FMI, shall be multiply accessed. The processor address space shall contain all memory in a contiguous block; the processor shall be allowed read/write access to the list-sequencer memory for FMI control code loading and data transfer while the FMI data channel controller shall be allowed read/write access of the list-sequencer memory. Control of the multiple port access to the contiguous block of memory shall be under control of a separate data-channel interface.

3.2. Interface Requirements

3.2.1. SCSI

The FRPM shall provide a Small Computer System interface (SCSI) to service optional disk drives and other peripheral devices. This interface shall provide direct memory access between the external SCSI devices and the FRPM memory.

3.2.2. Ethernet

The FRPM shall provide an ethernet interface to service inter-processor communication. This interface shall provide direct memory access between the external ethernet nodes and the FRPM memory.

3.2.3. RS-232

The FRPM shall provide two RS-232 interfaces to service data terminals and/or inter-processor communication. This interface shall provide memory-mapped control and data transfer between the processor and the RS-232 controller integrated circuit.

3.2.4. Timers

For support of real-time operations, the FRPM shall provide four 32-bit timers with time increments 1 ms and 100ns.

3.2.5. Memory Data Channel Interface

A separate memory data channel controller shall provide contention resolution during accesses to the program/list-sequencer memory to allow both the controlling processor and FMI controller access to memory. The data channel controller shall be independent of the controlling processor and the FMI controllers.

3.2.6. FMI for Crate and Cable Segments

The FRPM shall control two FASTBUS I/O ports. One shall connect to the crate segment and the other to the auxiliary connector. With an auxiliary adapter card, the latter port may be connected to a cable segment or a custom bus through a protocol converter. Program control for each FASTBUS port shall be possible through category B and category I FASTBUS standard routines.

The FASTBUS Master Interface (FMI) shall drive each FASTBUS port. The FMI enables the processor to initiate arbitration, address, and data cycles, as well as other control functions. In addition the processor may initiate a series of FASTBUS operations and control functions, directed by a user-specified list contained in the list-sequencer memory.

3.2.7. General Purpose Input and Output Signals

A set of eight differential general purpose input signals and eight differential general purpose output signals shall be available outside the FRPM module. The voltage level shall be compatible with 10K ECL technology. The output signals shall be able to drive a 25 ohm load continuously.

3.3. Special Requirements

3.3.1. Form Factor

The FRPM module shall be a single or a double width FASTBUS module, as specified in IEEE-960.

3.3.2. Front Panel Indicators

The FRPM shall meet the specifications of IEEE-960 for each master/slave front panel indicators. In addition, there shall be a red LED indicating the operating status of the controlling processor.

3.3.3. Front Panel Connectors

The FRPM front panel shall contain a multi-pin connector for the general purpose input and output signals specified in Section 3.2.7.

3.3.4. Operating Systems

3.3.4.1. Real-time

The operating system shall support real-time functions. Interrupts servicer shall allow interrupts from the FMIs or from the front panel to trigger tasks executing on the processor. The time during which interrupts are inhibited during the interrupt routine shall be 100 us.

The operating system shall support external timers for task scheduling.

3.3.4.2. Multi-tasking

The operating system shall support the creation, termination, and sequential execution of multiple tasks. A task shall be able to suspend its execution or the execution of another task until an I/O event occurs, a timer event occurs, or until resumed by another task. A task shall be able to terminate itself or another task. A task shall be able to create a new task.

3.3.4.3. Input/Output

The operating system shall support standard I/O devices including ethernet with TCP/IP, SCSI for disks, and RS-232 for terminals.

3.3.4.4. Commercial Availability

The operating system for the FRPM shall be sold and supported by a commercial vendor in business for two or more years.

3.3.5. Development Tools

3.3.5.1. Compilers

Compilers for assembly language, Fortran, and C shall be available to generate code for the target processor. These software packages may either run on a support computer based upon a different processor family (cross compiler) or upon the same processor family (native compiler). Each compiler shall translate source files to object files.

3.3.5.2. Libraries

Software libraries shall be available to interface with operating systems functions and mathematical operations.

3.3.5.3. Linkers

Linkers shall be available to combine object files and libraries into executable images. The executable images shall be downline-loadable into the target processor.

3.3.5.4. Debuggers

Symbolic source code debuggers shall be available. The debuggers shall allow interactive placement of breakpoints and interactive variable examination and modification. The debugger may run on the support computer or on the target processor itself.

3.3.5.5. Workstations

A processor that is used as the computer engine for a commercial workstation shall be preferred if the workstation can be used for some code development and documentation generation.

3.3.5.6. Commercial Availability

The development tools for the FRPM shall be sold and supported by a commercial vendor in business for two or more years.

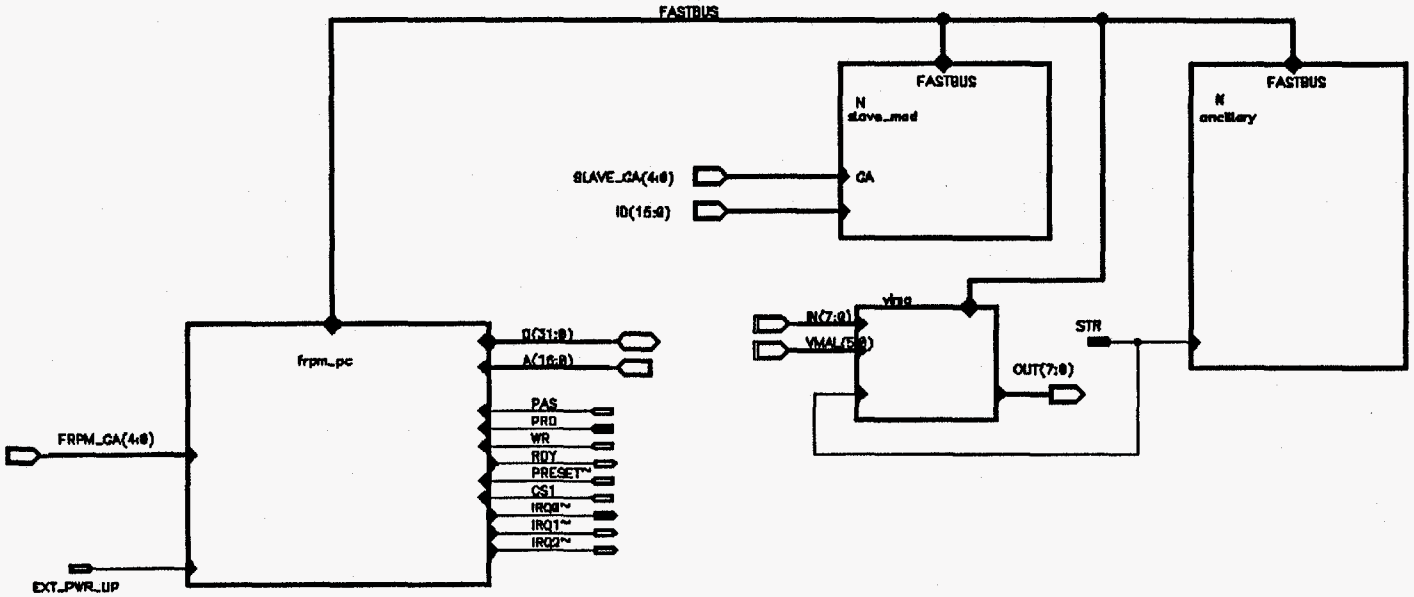
APPENDIX B

FRPM SCHEMATICS

TTL LOGIC INTERFACE

FRPM STANDARD-LOGIC (TTL) INTERFACE

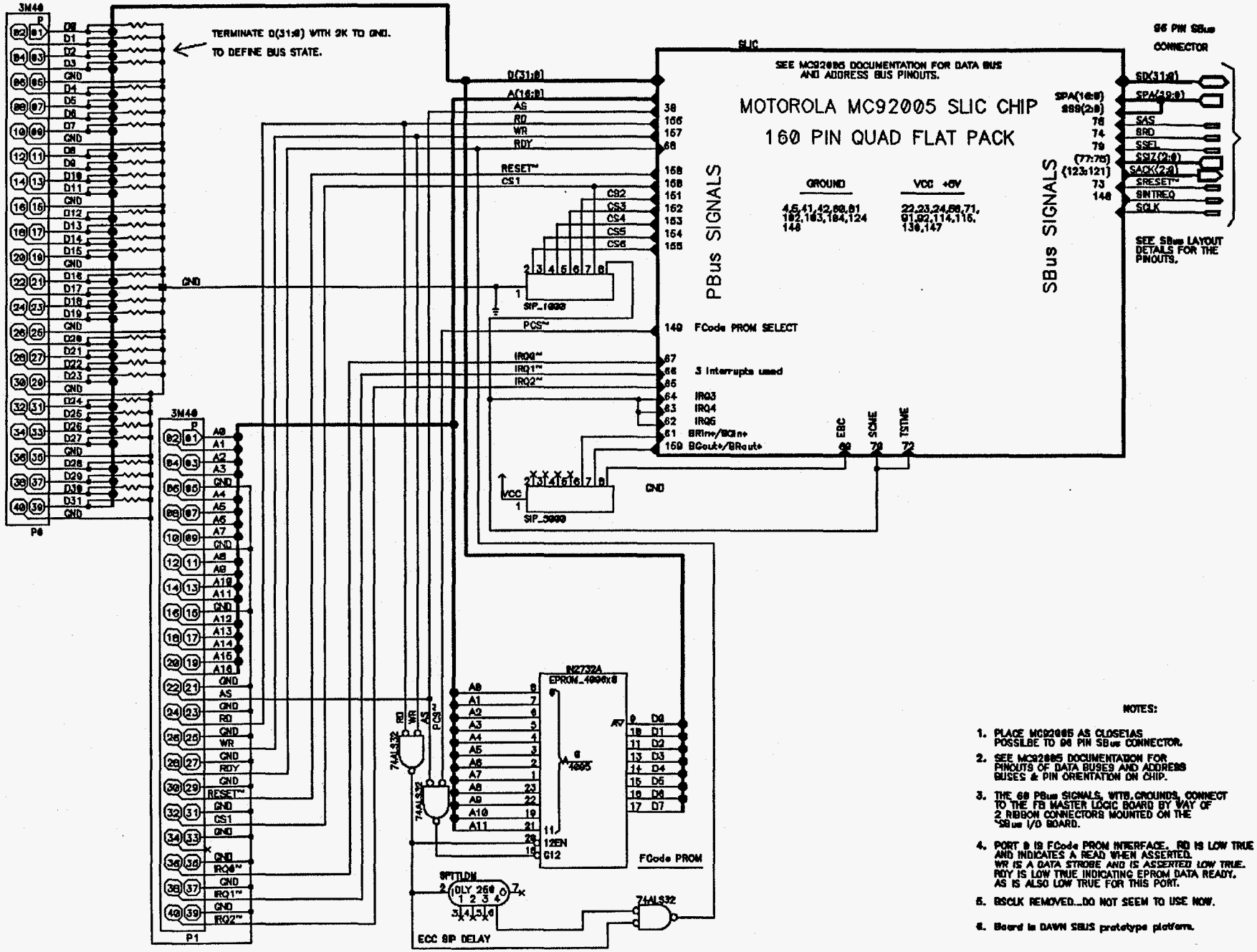
The schematic contained on page 24 is the simulation block diagram for the FRPM, while the schematic on page 25 is the SBus interface logic used to couple the SPARCengine interface bus to the FRPM master logic. Pages 26 through 38 contain the actual master logic interface that was completed and operationally tested during the Phase II effort. Prior to commiting the design to printed-circuit wiring layout, the entire design was simulated to test the operation of the interface.



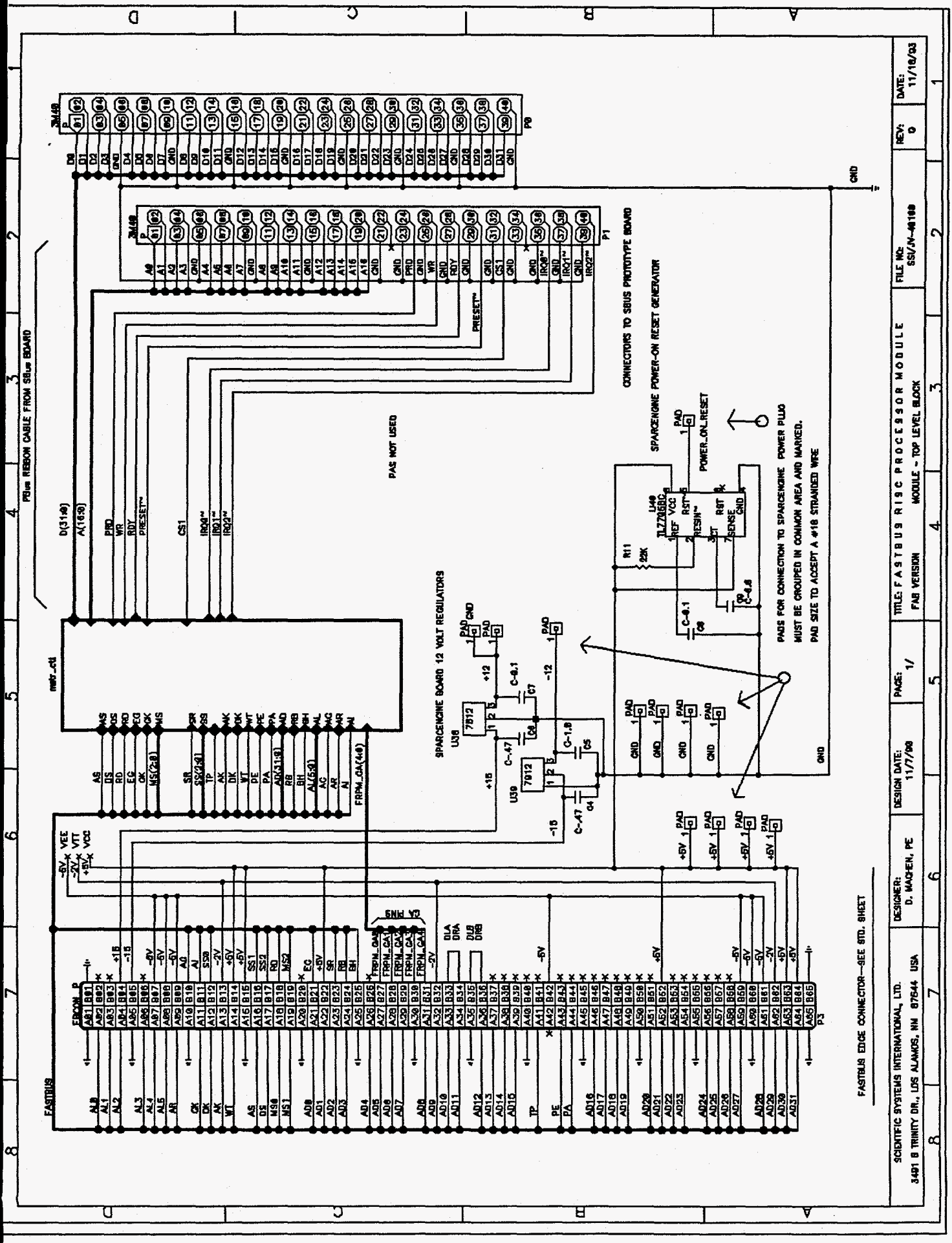
NOTES:

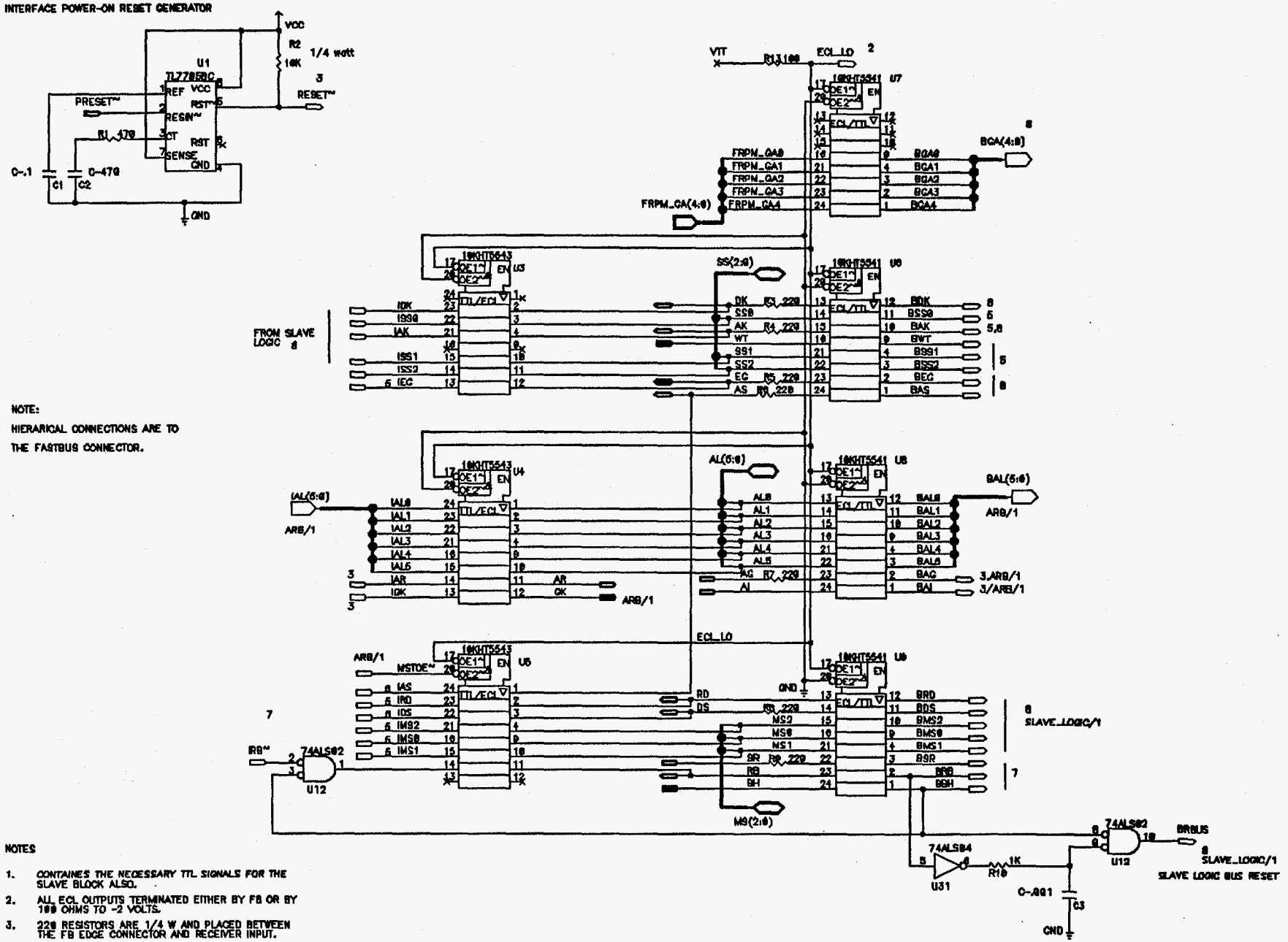
1. THE FRPM WILL HAVE THE CAPABILITY OF INTERFACING BOTH THE CRATE SEGMENT AND THE CABLE SEGMENT. THE INTERFACE TO THE SE IS BY SBUS MOTHER-BOARD PLUGGED INTO AN APPROPRIATE SBUS CONNECTOR.
2. EACH FASTBUS INTERFACE IS SEPARATE WITH NO COUPLING BETWEEN THE TWO EXCEPT VIA THE SE PROCESSOR.
3. THIS SHEET FOR TESTING PURPOSES. FAB STARTS AT FRPM_PC

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 10/4/90	PAGE: 1/1	TITLE: FASTBUS RISC PROCESSOR MODULE SIMULATION MODEL-TOP LEVEL - FRPM	FILE NO: SSI/JV-40168-1	REV: F	DATE: 4/13/93



SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 11/7/90	PAGE:	TITLE: FASTBUS R ISC PROCESSOR MODULE SBus BOARD - FRPM	FILE NO: SSI/JV-48116	REV: L	DATE: 6/3/93


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: D. MACHEN, PE
DESIGN DATE: 11/7/96
PAGE: 1/
TITLE: FASTBUS RISC PROCESSOR MODULE
FAB VERSION
MODULE - TOP LEVEL BLOCK
FILE NO: SSI/JV-40188
REV: 0
DATE: 11/18/03
FASTBUS EDGE CONNECTOR--SEE STD. SHEET
PBus RIBBON CABLE FROM SBus BOARD
PAS NOT USED
CONNECTORS TO SBUS PROTOTYPE BOARD
SPARCENGINE POWER-ON RESET GENERATION
SPARCENGINE BOARD 12 VOLT REGULATORS
PADS FOR CONNECTION TO SPARCENGINE POWER PLUG
MUST BE GROUPED IN COMMON AREA AND MARKED.
PAD SIZE TO ACCEPT A #18 STRANDED WIRE
FASTBUS
P3
P0
P1
mstr_ctl
U36
U39
U40
7812
TL7705BC
R11
22K
POWER_ON_RESET
PRESET~
GND
+5V
-5V
+12
-12
+15
-15
VEE
VTT
VCC



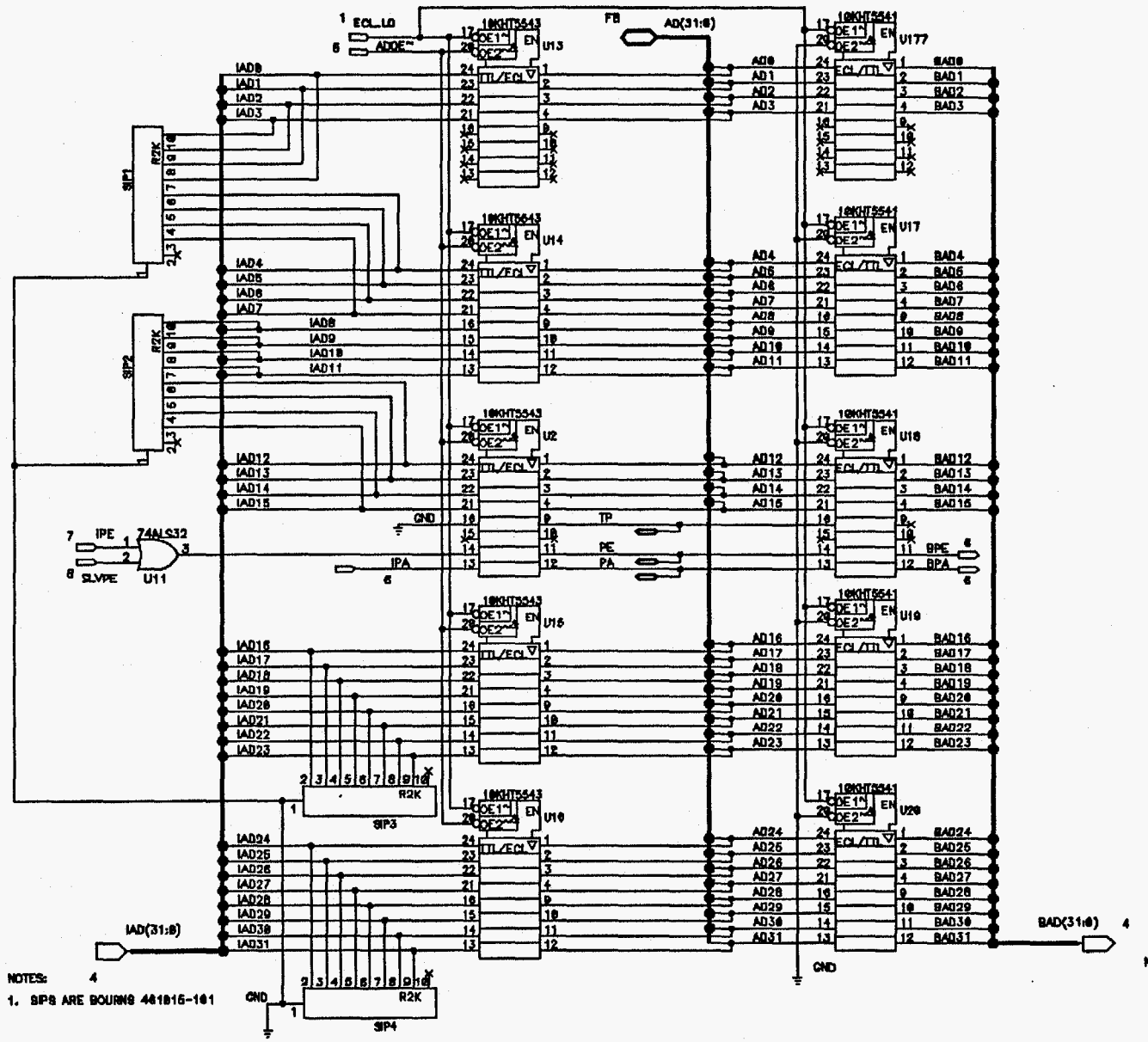
NOTE:

HIERARICAL CONNECTIONS ARE TO THE FASTBUS CONNECTOR.

NOTES

1. CONTAINES THE NECESSARY TTL SIGNALS FOR THE SLAVE BLOCK ALSO.
2. ALL ECL OUTPUTS TERMINATED EITHER BY F8 OR BY 180 OHMS TO -2 VOLTS.
3. 220 RESISTORS ARE 1/4 W AND PLACED BETWEEN THE F8 EDGE CONNECTOR AND RECEIVER INPUT.
4. -2 VOLTS IS VTT BUS

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 3/8/93	PAGE: 1/	TITLE: FASTBUS RISC PROCESSOR MODULE MASTER CONTROL LOGIC NON-ASIC VERSION			FILE NO: SSI/JV-46381	REV: 1	DATE: 8/16/93
8	7	6	5	4	3	2	1		



NOTES: 4

1. SIPS ARE BOURNS 4618S-101

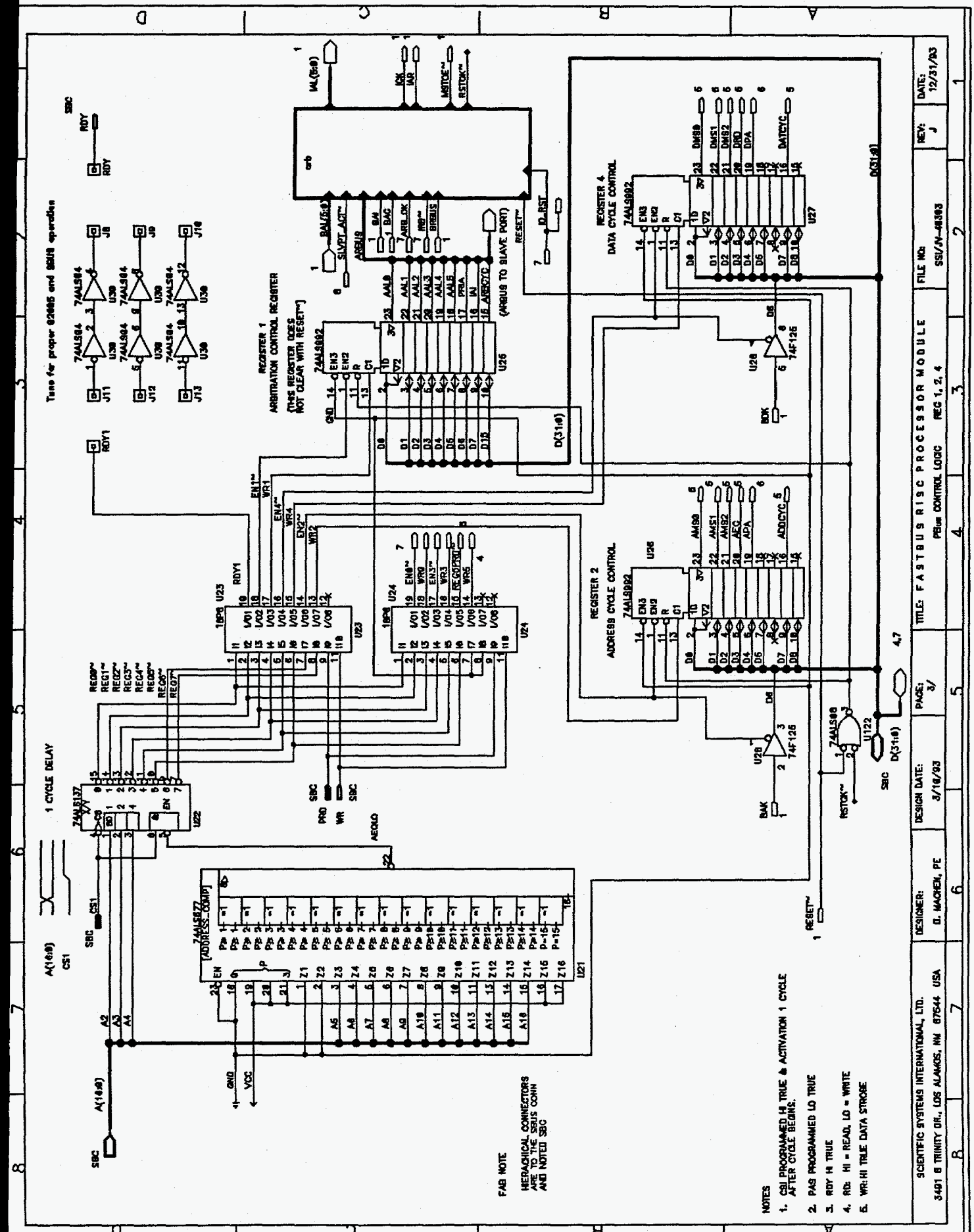
2. MAY SWAP SIP PINS AS NECESSARY DURING LAYOUT.

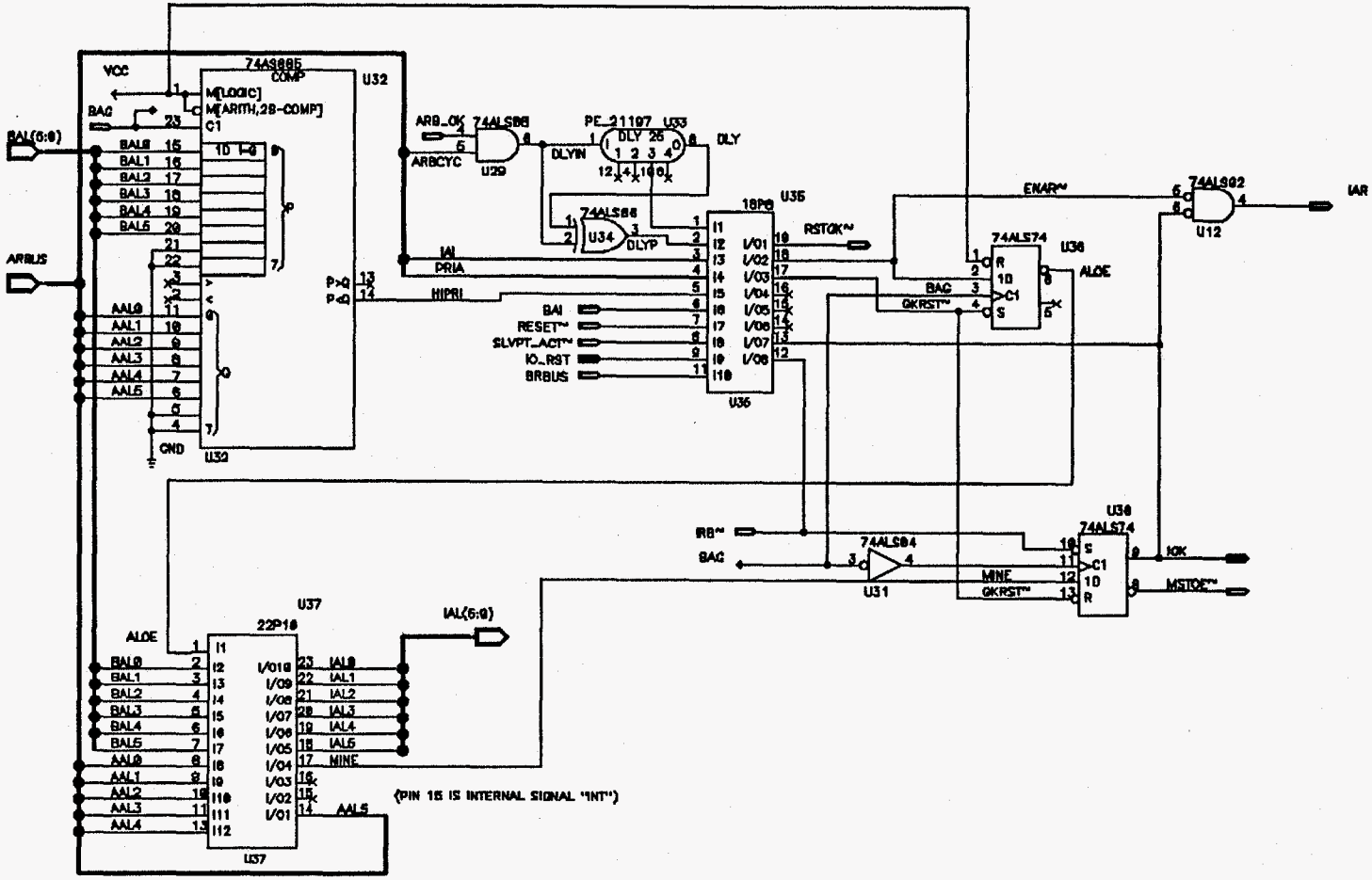
NOTES 4

1. ADOE** / ADOE MUST HAVE AN ENABLE TERM FROM THE SLAVE PORT.

2. ALLOW BAD TO MONITOR AD AT ALL TIMES.

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 3/19/93	PAGE: 2/	TITLE: FASTBUS RISC PROCESSOR MODULE AD BUS BUFFERING	FILE NO: SSI/JV-48302	REV: H	DATE: 11/18/93


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: D. MADDEN, PE
DESIGN DATE: 3/10/93
PAGE: 3/
TITLE: FASTBUS RISC PROCESSOR MODULE
PBus CONTROL LOGIC REG 1, 2, 4
FILE NO: SSI/JV-46393
REV: J
DATE: 12/31/93
REGISTER 1
ARBITRATION CONTROL REGISTER
(THIS REGISTER DOES NOT CLEAR WITH RESET~)
REGISTER 2
ADDRESS CYCLE CONTROL
REGISTER 4
DATA CYCLE CONTROL
1 CYCLE DELAY
NOTES
1. CSI PROGRAMMED H TRUE & ACTIVATION 1 CYCLE AFTER CYCLE BEGINS.
2. PAS PROGRAMMED LO TRUE
3. RDY H TRUE
4. RD: HI = READ, LO = WRITE
5. WR: HI TRUE DATA STROBE
FAB NOTE
HIERARCHICAL CONNECTORS ARE TO THE SBUS CONN AND NOTED SBC



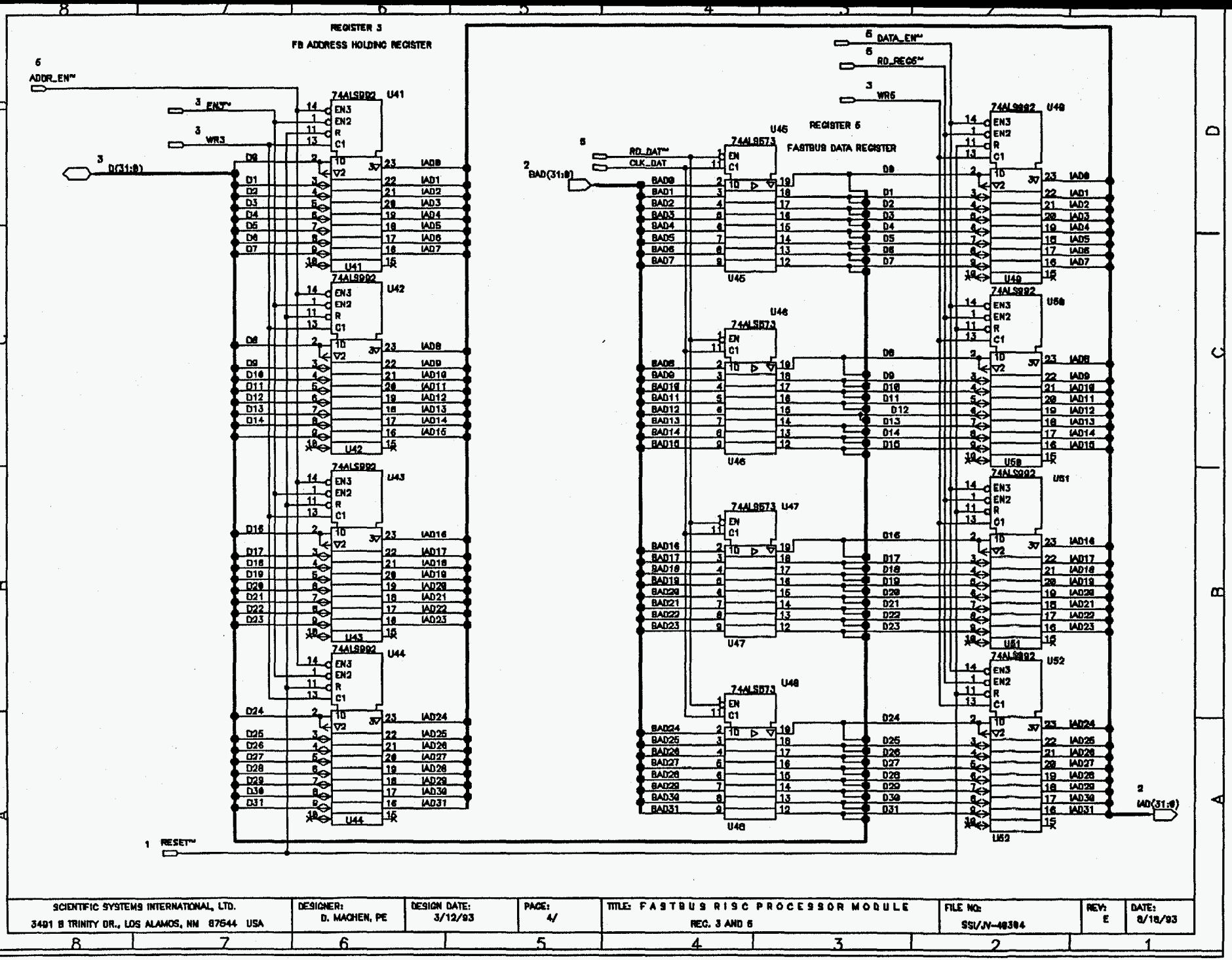
NOTES:

1. ALL EXTERNAL SIGNALS SOURCE/SINK FROM MSTR_CTL/3.DRW DESTINATION FROM SHOWN FROM 3.DRW.

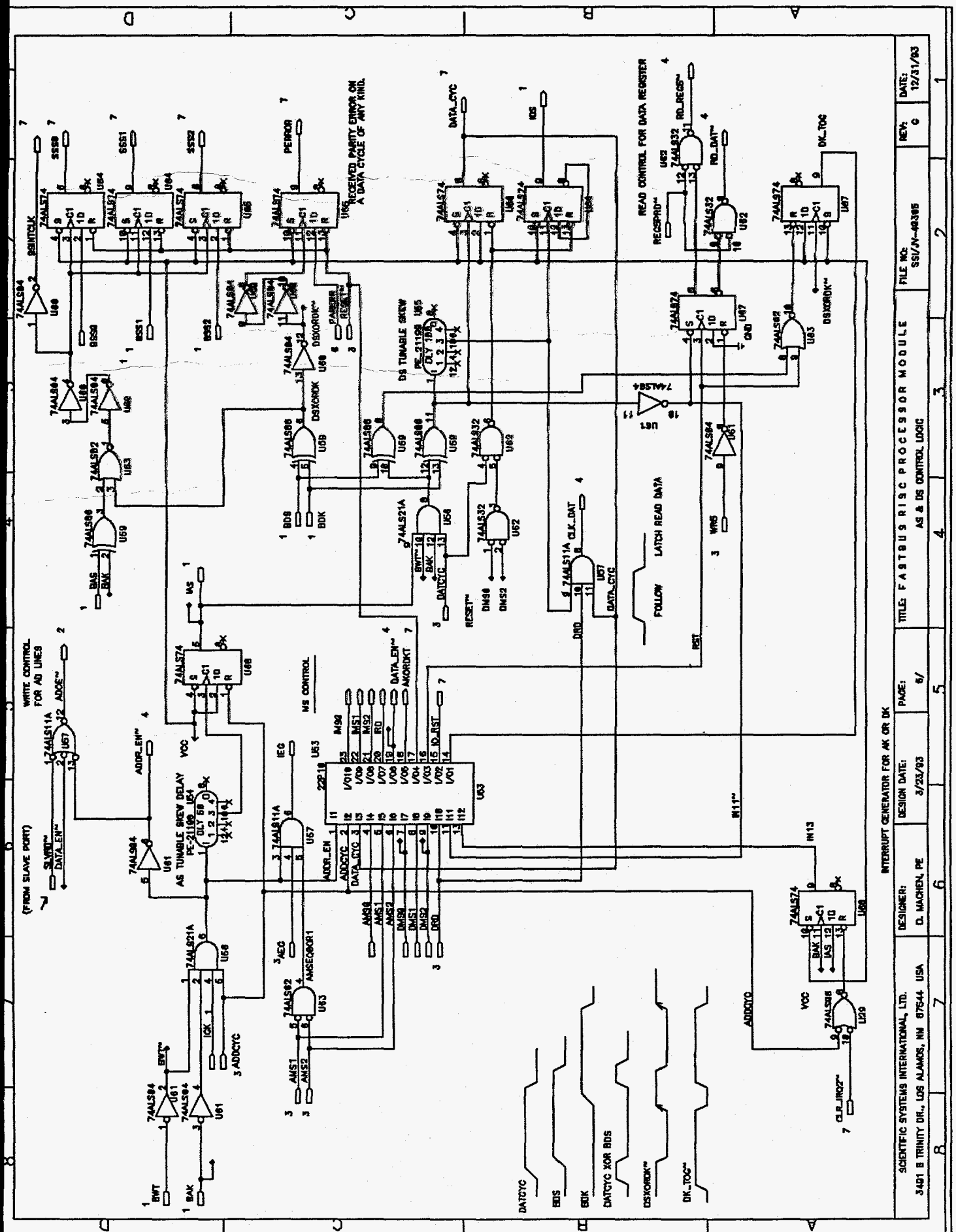
NOTES

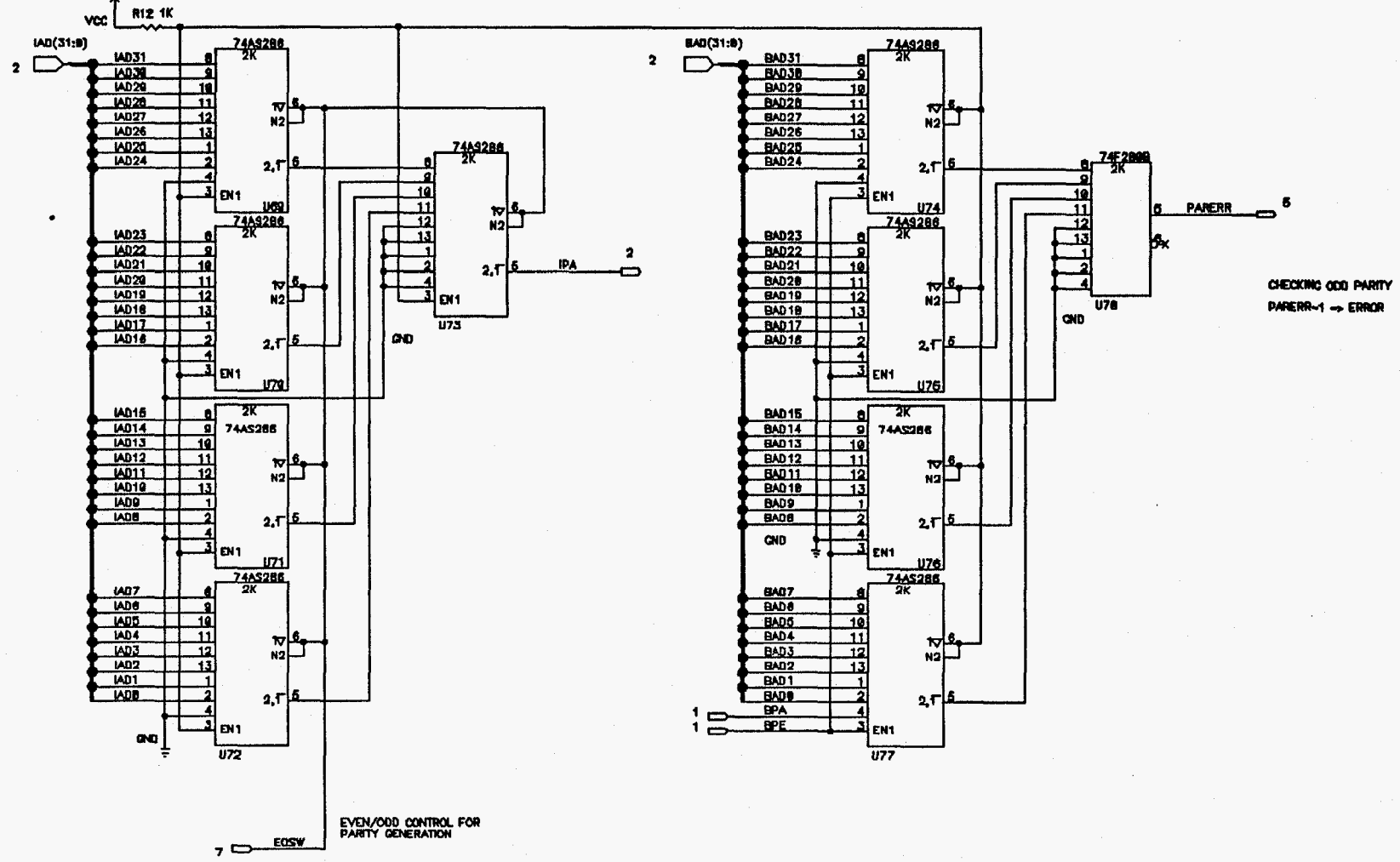
1. ARBITRATION LOGIC INTO A PAL
2. F86 AND F04 REPLACED BY AS810 IF AVAILABLE & NOT OFFERED IN PAL
3. IF THE SLAVE PORT IS ACTIVE PRIOR TO MASTER REQUESTING ARBITRATION, PREVENT ARBITRATION AND FORCE FRPM TO LOOK AT REG 0 STATUS.

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 3/16/93	PAGE: 1/	TITLE: FASTBUS RISC PROCESSOR MODULE ARBITRATION LOGIC	FILE NO: SSI/JV-48401	REV: J	DATE: 12/1/93



SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3421 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 3/12/93	PAGE: 4/	TITLE: FASTBUS RISC PROCESSOR MODULE REG. 3 AND 6	FILE NO: SSI/N-40384	REV: E	DATE: 8/18/93


INTERRUPT GENERATOR FOR AK OR DK
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: D. MACHEN, PE
DESIGN DATE: 3/23/93
PAGE: 6/
TITLE: FASTBUS RISC PROCESSOR MODULE
AS & DS CONTROL LOGIC
FILE NO: SSI/AN-49305
REV: C
DATE: 12/31/93
(FROM SLAVE PORT)
WRITE CONTROL FOR AD LINES
AS TUNABLE SKEW DELAY
DS TUNABLE SKEW
MS CONTROL
LATCH READ DATA
FOLLOW
RESET
READ CONTROL FOR DATA REGISTER
RECEIVED PARITY ERROR ON A DATA CYCLE OF ANY KIND
DATCYC
BDS
BDK
DATCYC XOR BDS
DSXORDK
DK_TOG



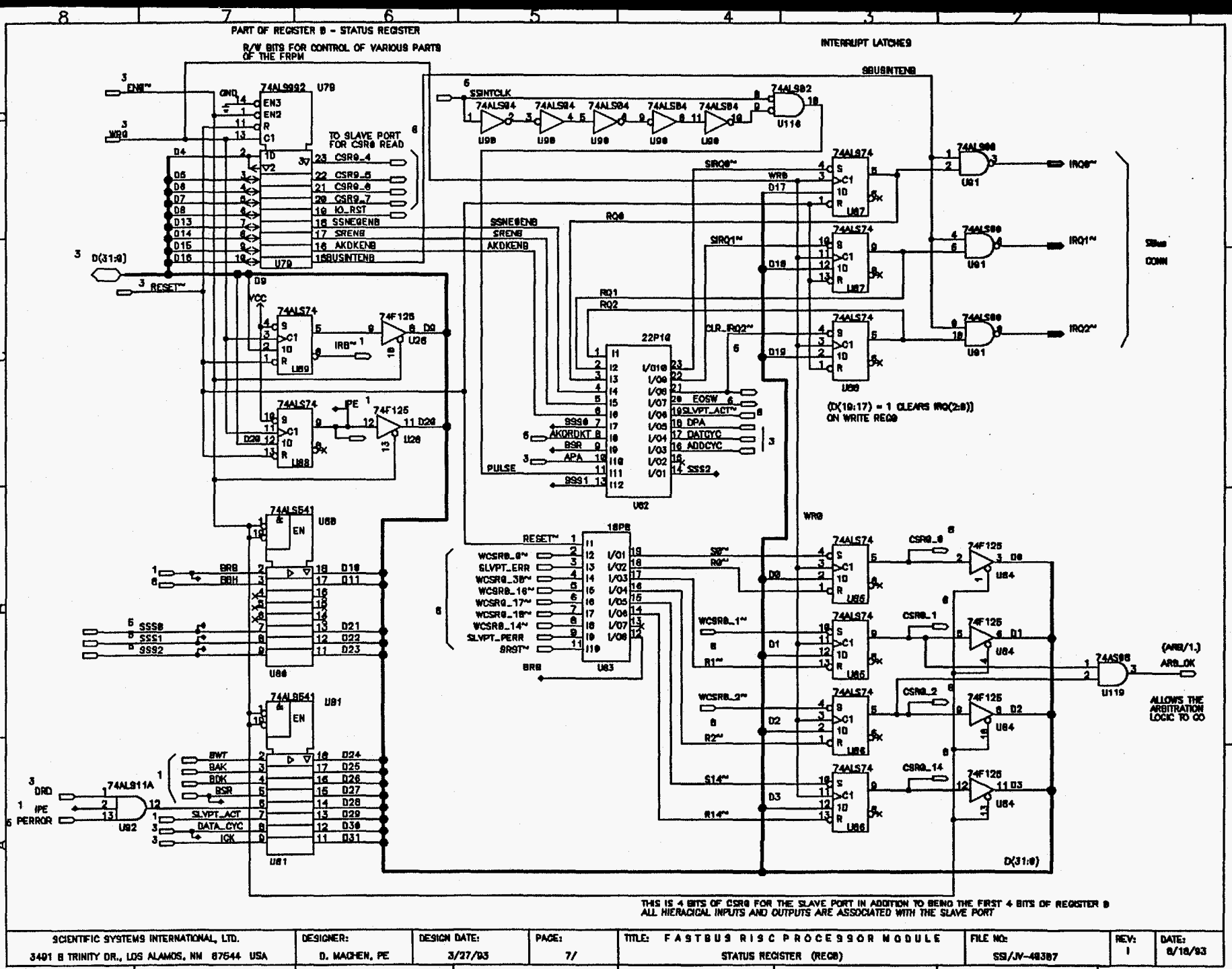
NOTES

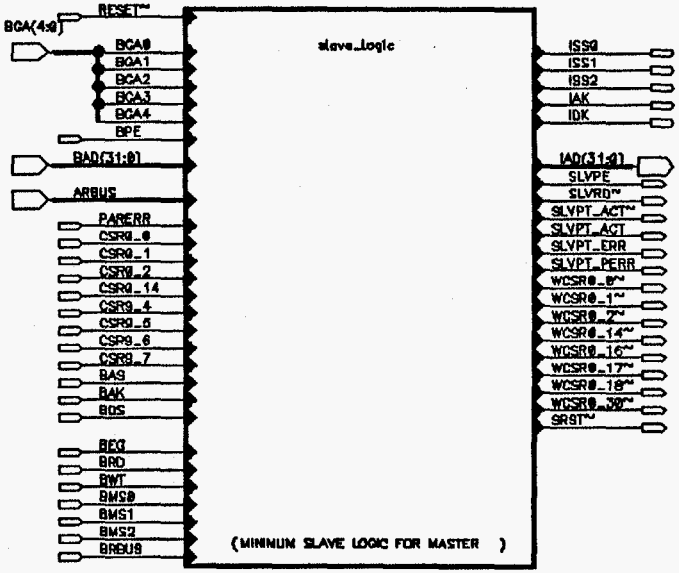
1. 9-13 NS TO CORRECT VALUE FOR PARERR

APA/DPA: LO-GENERATE EVEN PARITY
HI-GENERATE ODD PARITY

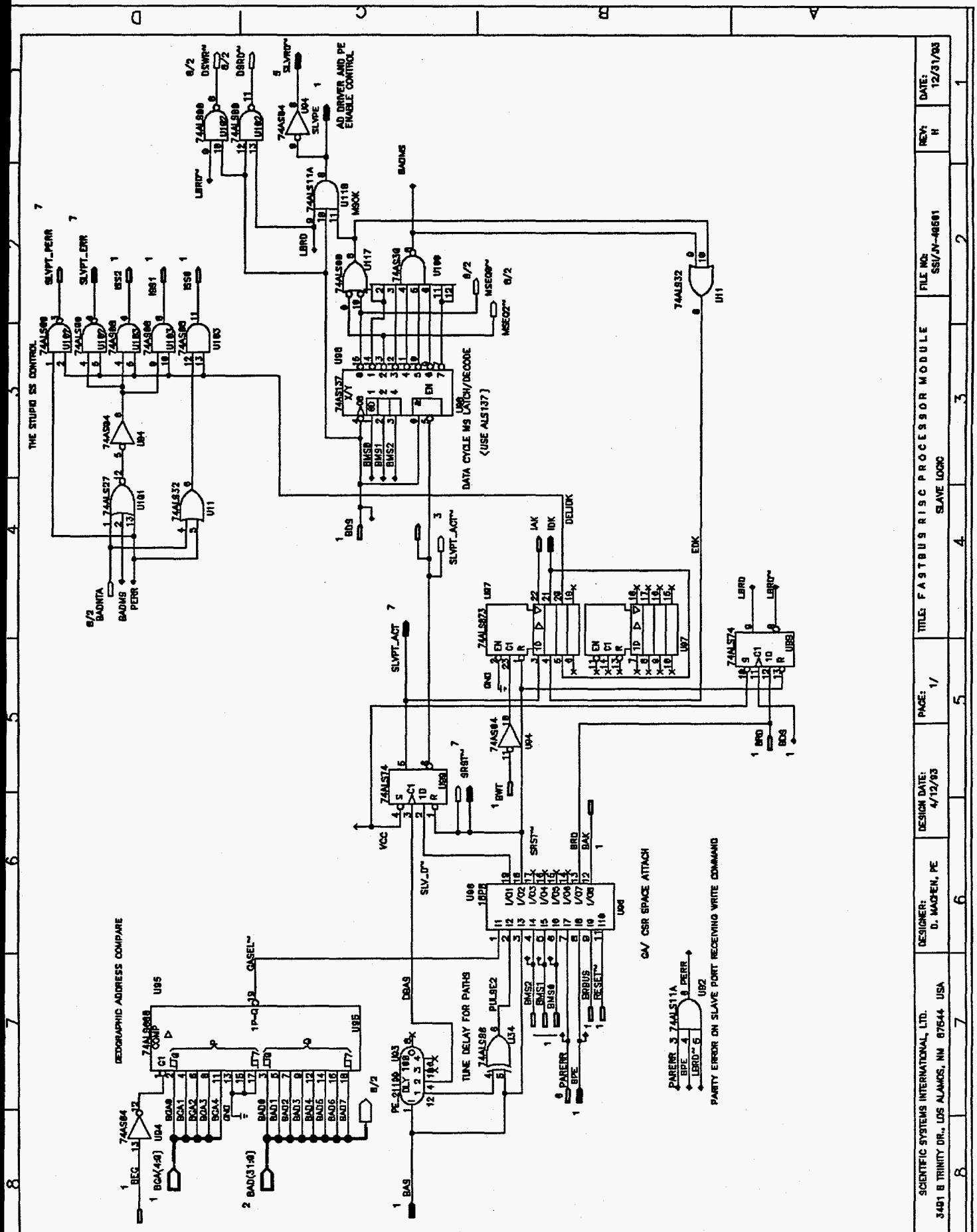
SLVPT_ACT*: LO = GENERATE ODD PARITY
NORMALLY HIGH WHEN MASTER OPERATING

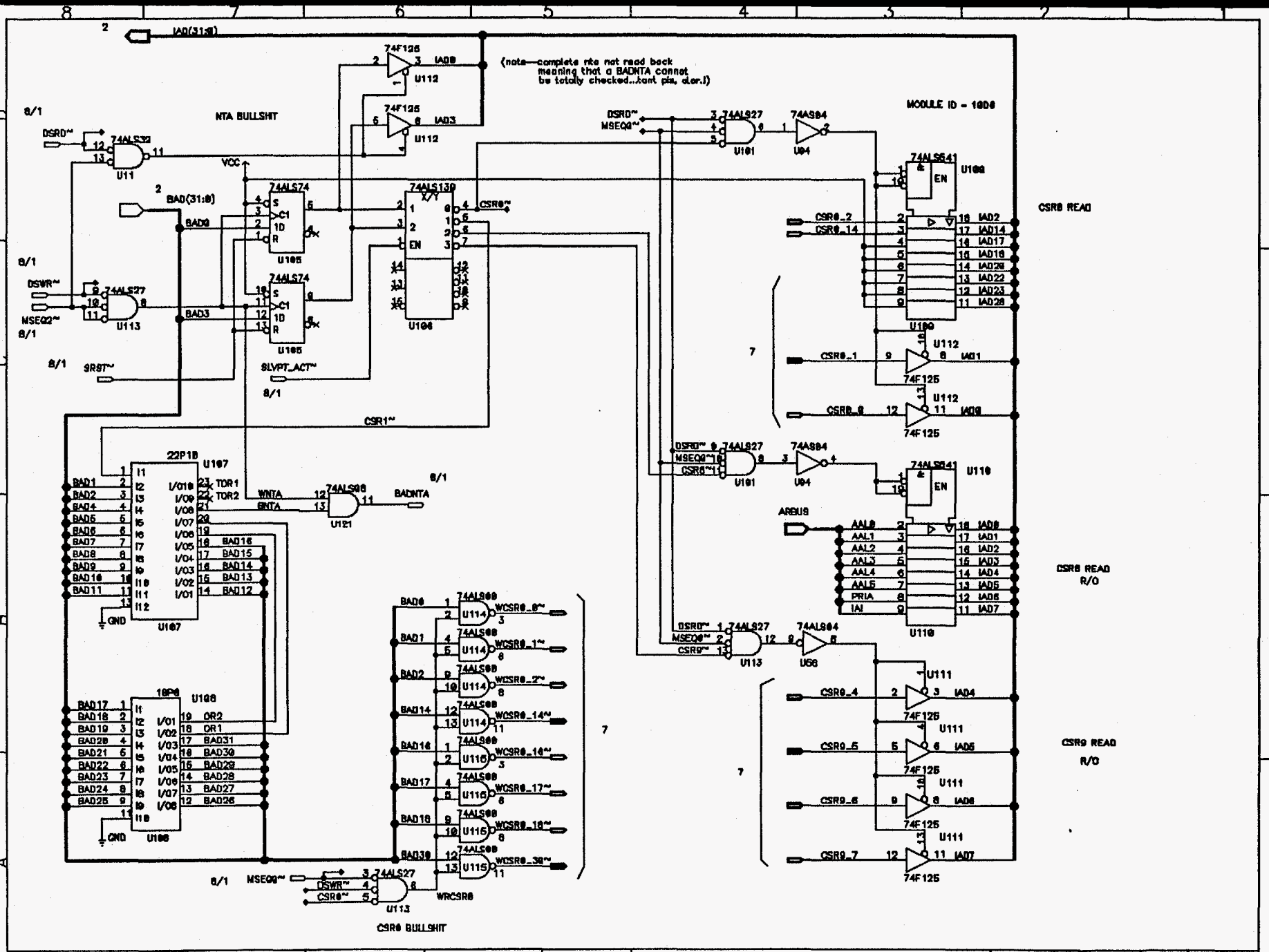
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3421 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 3/23/93	PAGE: 6/	TITLE: FASTBUS RISC PROCESSOR MODULE JPARITY CIRCUITS	FILE NO: SSI/JV-40386	REV: E	DATE: 6/16/93


PART OF REGISTER 9 - STATUS REGISTER
R/W BITS FOR CONTROL OF VARIOUS PARTS OF THE FRPM
INTERRUPT LATCHES
TO SLAVE PORT FOR CSR8 READ
(D(19:17) = 1 CLEARS IRQ(2:0)) ON WRITE REG9
ALLOWS THE ARBITRATION LOGIC TO GO
THIS IS 4 BITS OF CSR9 FOR THE SLAVE PORT IN ADDITION TO BEING THE FIRST 4 BITS OF REGISTER 9 ALL HIERACICAL INPUTS AND OUTPUTS ARE ASSOCIATED WITH THE SLAVE PORT
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: D. MACHEN, PE
DESIGN DATE: 3/27/93
PAGE: 7/
TITLE: FASTBUS RISC PROCESSOR MODULE
STATUS REGISTER (REG9)
FILE NO: SSI/JV-49387
REV: 1
DATE: 6/16/93

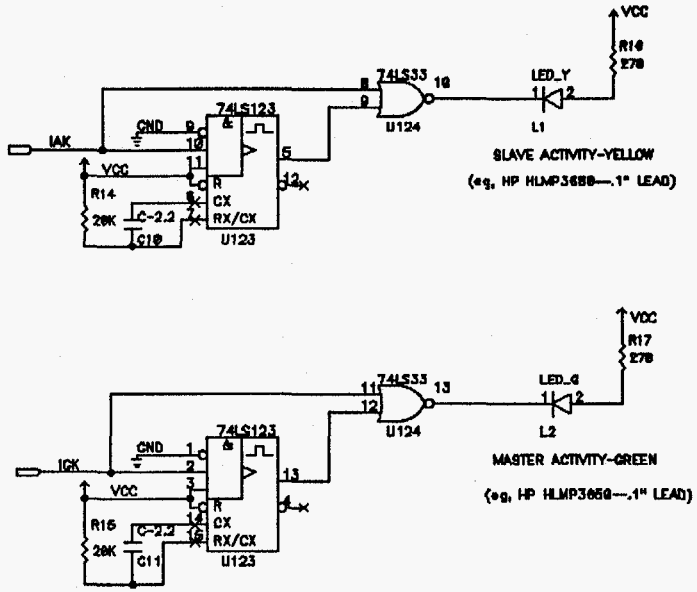


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 4/12/93	PAGE: 8/	TITLE: FASTBUS RISC PROCESSOR MODULE SLAVE LOGIC BLOCK	FILE NO: SSI/JV-40308	REV: 0	DATE: 6/18/93


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: D. MADSEN, PE
DESIGN DATE: 4/12/93
PAGE: 1/
TITLE: FASTBUS RISC PROCESSOR MODULE
SLAVE LOGIC
FILE NO: SSI/AV-40501
REV: H
DATE: 12/31/93
THE STUPID SS CONTROL
DATA CYCLE MS LATCH/DECODE
(USE ALS137)
AD DRIVER AND PE ENABLE CONTROL
MASK
GEOGRAPHIC ADDRESS COMPARE
TUNE DELAY FOR PATHS
CA/ CSR SPACE ATTACH
PARITY ERROR ON SLAVE PORT RECEIVING WRITE COMMAND



SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 4/16/93	PAGE: 2/2	TITLE: FASTBUS RISC PROCESSOR MODULE (GENERAL BULLSHIT) SLAVE LOGIC —CSR READS AND WRITES	FILE NO: SSI/JV-4858Z	REV: F	DATE: 12/1/93



SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 12/1/93	PAGE: 9/	TITLE: FASTBUS RISC PROCESSOR MODULE FRONT PANEL LED DRIVERS	FILE NO: SSI/JV-40300	REV: A	DATE: 12/2/93

APPENDIX C

FASTBS MASTER INTERFACE (FMI)

GATE-ARRAY SPECIFICATIONS

FASTBUS MASTER INTERFACE SPECIFICATION AND FUNCTIONAL REQUIREMENTS

ORIGINAL DOCUMENTATION PRIOR TO MODIFICATION OF THE SCOPE-OF-WORK

1. Introduction

The FASTBUS Master Interface (FMI) shall be a single-chip ASIC implementation of protocols defined by IEEE 960 for FASTBUS master devices. In addition the FMI shall include complete slave logic to perform all slave functions. Thus, modules acting primarily as master devices and necessarily as slave devices, will require only a single FASTBUS interface chip.

The FMI shall be controlled through a processor port. The port enables the processor to initiate arbitration, address, and data cycles, as well as other control functions described below. In addition the processor may initiate a series of FASTBUS operations and control functions, directed by a user-specified list contained in a memory and shared between the processor and FMI.

1.1. Scope

This document identifies the functional capabilities and interface features of the FMI. These requirements, resulting from studies performed during the Phase I research for the FASTBUS RISC Processor Module, represent compromises made between the set of all desirable features for FASTBUS masters, and the set of feasible implementations. While these requirements are not inflexible, they are established only after thorough research to direct and limit subsequent design and implementation.

1.2. Interface Summary

The FMI shall implement a bus and a processor interface. The bus interface shall allow direct connection to the FASTBUS crate segment, and shall support all the protocol defined by IEEE 960 for master devices. The processor interface shall provide a single 64-bit path to the FMI that may be used in two modes. Seven other ancillary signals shall be provided to select the interface operation and provide timing controls. For the first mode of control, the interface path shall enable the processor to control each operation executed by the FMI. In this slaved mode, the FMI is totally dependent on the processor for control. The processor shall optionally supply data embedded in the control stream. For the second mode for control of

the FMI, the interface path allows the FMI to fetch equivalent control and data information through a direct memory access (DMA) controller built into the FMI. The latter mode of operation permits the FMI to operate autonomously once triggered, and provides a high performance FASTBUS sequencer controlled by predefined lists of primitive FASTBUS operations stored in shared processor memory. Both modes shall use the list sequencer to deposit input data in memory.

1.3. Functional Summary

The FMI shall provide direct FASTBUS I/O for a controlling processor, list driven FASTBUS sequences with data driven to/from memory via a list sequencer, flexible buffer management options, interrupts for exception handling, and primitive FASTBUS slave operations. Direct FASTBUS I/O shall allow the processor to execute simple FASTBUS operations such as arbitration cycles, address cycles, data cycles, and embedded control functions. The control functions allow the processor to alter the prevailing FASTBUS operating environment by changing the parameters set that characterizes the state and mode of the interface. Full FASTBUS transactions may be constructed with combinations of arbitration, address, and data cycles.

FASTBUS errors shall be reported as exceptions. When sequencing through a predefined list in memory, execution shall move data to/from user memory until the list terminates or an unexpected SS code is encountered. With each FASTBUS cycle, the user shall specify a filter that defines which SS codes or timeouts shall generate exceptions. The FMI response to an exception is handled in one of two ways: the list execution promptly terminates, or faults are saved in an exception list that is passed to the user upon completion of the FASTBUS operation list.

When processing FASTBUS operation lists, data references shall have three alternate mechanisms. First, the reference shall be immediate, i.e. the data itself is passed to the FMI from the memory. This immediate data shall be allowed for write operations only. Second, the reference shall provide an address for the data. Data shall be read/written at this address. Third, the reference shall provide the address of a data structure that manages a stack which shall provide/accept data. The mode of access shall be specified by the FASTBUS operation list.

The FMI shall support interrupts of two different forms: interrupt messages and service requests. The FMI shall support up to 16 interrupt receiver blocks, the maximum defined by IEEE 960.

1.4. Assumptions and Constraints

The memory and memory interface architecture shall not prevent the FMI from executing at the maximum speed permitted by the FASTBUS specification, IEEE 960. Therefore, a built-in list sequencer shall be provided within the FMI to achieve satisfactory performance. Furthermore, the associated memory architecture should be optimized to minimize contention between the processor and the DMA.

The FMI shall be a memory-mapped I/O device. As such, the 64-bit data bus and 32-bit address bus shall be common to both processor access and list sequencer access to memory.

All addresses presented to the FMI from either the processor or the list sequencer shall be absolute addresses referencing physical memory. No address translation shall be provided.

2. Applicable Documents

2.1. FASTBUS Documents

2.1.1. Hardware

IEEE Standard FASTBUS Modular High-Speed Data Acquisition and Control System, ANSI/IEEE Std 960-1986.

2.1.2. Software

FASTBUS Standard Routines, U. S. NIM Committee, May 1988.

3. Requirements

3.1. Interface Requirements

3.1.1. FASTBUS

Electrical characteristics of the FMI FASTBUS protocol pins shall conform with limits defined by the bus standard IEEE 960, such that the FMI may be directly connected to the FASTBUS crate segment. To drive the FASTBUS cable segment, these same signals shall be buffered with appropriate current-mode drivers.

Protocol for the FASTBUS signals shall also conform to the requirements of IEEE 960. All address and data cycle modes shall be supported including those known as advanced modes (first proposed in Los Alamos in August, 1987). The advanced modes allow data transfers to take place on each and every DS(t) except those specifically identified as bus cleanup cycles.

3.1.2. Processor-directed I/O

If not processing a FASTBUS operation list, the FMI shall accept a processor-directed I/O sequence to alter the state or mode of the FMI and/or execute one or more FASTBUS primitive operations. Each command shall consist of one 64-bit word received by the FMI from the processor. Upon receipt of the command, the FMI shall initiate an asynchronous FASTBUS operation, or a synchronous non-FASTBUS operation. In either case, the processor shall await completion of the command before proceeding to the next command.
Processor-directed I/O shall be used to initiate list-directed I/O. Once initiated, the list-directed I/O shall have priority over processor-directed I/O. By checking a 32-bit FMI status register, the processor shall be able to determine that list-directed I/O is in progress and that attempts to initiate any other I/O shall be denied. Details of the internal formats for commands, data, and status for the FMI are included in Section 3.3.1.

3.1.3. List-directed I/O

List-directed I/O shall be initiated by processor-directed I/O. After initialization, the built-in list sequencer shall provide 64-bit word memory access for both control and data transfer. The behavior of the FMI shall be virtually identical for identical sequences of FMI commands independent of whether the sequence is received directly from the processor, or indirectly from the list sequencer fetching/storing 64-bit data words and commands.
Once initiated, list-directed I/O runs to completion, i.e. until the list terminates successfully, or the list prematurely terminates with an exception. The FMI shall notify the processor of list completion by issuing an interrupt.
Details of the internal formats for commands and data are included in Section 3.3.1.

3.1.4. Configurability

The FMI chip shall be configurable with option enable pins permitting selection of various features. Separate pins shall be provided to enable use of internal registers CSR0, CSR3, CSR7, CSR8, CSR9, CSR1D, CSR1E, and CSR1F. With a set a five pins, a user shall be able to specify the width of the internal address field for logical addressing. Another option shall allow a user to select either SS = 1 or WT = 1 slave responses for internal busy conditions.

3.1.5. Interface Signals

3.1.5.1. Processor Interface Signals

All processor interface signals shall be used by the FASTBUS interface for access to data memory.

Signal	Description
ADDRESS(31:0)	Address for internal memory mapped control registers accepted by FMI during processor-directed I/O operation. Address for command or data access asserted by FMI during list-directed I/O operation. Contents of NTA register during FASTBUS access to user application memory.
BUS_GRANT	Set by user application to indicate FMI has control of processor bus
BUS_REQUEST	Set by FMI to obtain control of processor bus.
BUS_WRITE	Set to indicate write operation on processor bus. Reset for read operation.
DATA_ACK	DATA_ACK(t) asserted by memory when data accepted/asserted.
DATA_IN(63:0)	64-bit wide input data/command bus. FASTBUS interface uses bits 31 to 0 only.
DATA_OUT(31:0)	32-bit wide output data bus.
DATA_REQUEST	DATA_REQUEST(t) asserted by FMI for memory to accept/assert data.
DATA_TYPE(1:0)	Selects data type on processor bus. For null operation, set = 0. For 32-bit transfer, set = 1. For 64-bit transfer, set = 2.
INTERRUPT_EXCEPT	Issued to the processor when an exception has been issued.
INTERRUPT_FIM	Issued to the processor when a FASTBUS interrupt message has been received in one of 16 message receiver blocks.
INTERRUPT_SR	Issued to the processor when a service request has been received.

3.1.5.2. FASTBUS Interface Signals

All 60 FASTBUS signals shall be connected to the FMI. These are listed in IEEE 960, and are not repeated here.

Signal	Description

ADDRESS_CYCLE Asserted by FMI when module is initially addressed but reset once data cycles AK

ATTACH Command/status for slave attachment. Asserted by FMI to indicate module is attached. Asserted by user to force attachment at address time.

BLOCK_MODE Asserted by FMI for either block or pipeline mode active.

BUSY Asserted by user signal indicate that a request has been received while the module is busy.

CLEAR_DATA Asserted by FMI during CSR0 load if bit 31 is asserted.

CLEAR_ERROR Asserted by FMI during CSR0 load if bit 16 is asserted.

CSR0_EN User signal indicating that CSR0 bits 0 to 5, bit 14, and bit 15 are supported on-board the FMI chip.

CSR0_LOAD Asserted by FMI for load command of CSR0.

CSR0_READ Asserted by FMI to indicate read command for CSR0.

CSR3_EN User signal indicating that CSR3 internal to the FMI chip is used and that logical addressing is possible.

CSR7_EN User signal indicating that CSR7 internal to the FMI chip is used and that case 2 (class N) broadcast addressing is possible.

CSR_SPACE Asserted by FMI if CSR space is attached.
EOB Asserted by user when end of block for block transfer or pipeline operation.

LA_MODE_EN User signal indicating that advanced features of the FASTBUS protocol defined in the Los Alamos understanding are enabled.

LOGICAL_ADDR_EN Asserted by FMI when logical addressing is enabled with CSR0<1>.

LOGICAL_WIDTH(4:0) User signals defining the width of the internal address.

NOT_VALID Asserted by user to indicate the NTA address is not valid.

NTA_CARRY_IN Asserted by user to indicate whether the NTA will advance when incremented.

NULL_CYCLE Asserted by FMI following address connect-disconnect without a data cycle.

PARITY_ERROR Asserted by FMI when the FASTBUS data bus has wrong parity during data write cycles. User may use this signal to define SET_ERROR_FLAG and REJECT. Latched until CLEAR_ERROR is sensed.

POWER_UP Asserted by user when power has been applied to the user module.

RESET_BUS Deglitched reset bus, RB.

READ Asserted by FMI to indicate read data cycle.

REJECT Asserted by user to indicate that bad data or bad addresses will result in rejected data.

RESET Asserted by FMI during CSR0 load if bit 30 is set or during periods when the user asserts POWER_UP.

RESPONSE_CTL_IN Signal from user indicating that FMI may safely generate either AK or DK.

RESPONSE_CTL_OUT Signal to user indicating that module needs to generate either AK or DK. User logic may simply delay this signal and return the delayed version as RESPONSE_CTL_IN, or may implement more complex logic.

RUNNING Asserted when module is enabled to run with CSR0<2> = 1.

SET_ERROR_FLAG Asserted by user when error to be reported through CSR0<0>.

SET_HALT Asserted by user when run mode latched by CSR0<2> is to be cleared.

SET_SR Asserted by user when SR is desired.
SET_SS3 Asserted by user when SS = 3 is to be generated.

SKEW_IN Asserted by user when skew time has elapsed. Generally this is SKEW_OUT passed through a fixed delay.

SKEW_OUT Asserted when skew time interval required.

SR_FLAG Asserted by FMI following receipt of SET_SR from user. Reset when CSR0 loaded and bit 21 is set. Independent of the state of CSR0<4>.

TIMER_IN Asserted by user to provide time intervals counted by master timers.

TRANSFER_DONEDK-like handshake from user. Generated for all data cycles, even those internal to the FMI chip. Prerequisite for RESPONSE_CTL_OUT.

TRANSFER_ENABLE Asserted by FMI to indicate that the user must either accept or assert data.

TRANSFER_REQUEST DS-like handshake to user. Request may be read, write, or cleanup cycle in either CSR or data space. User logic may simply delay this signal and return the delayed version as TRANSFER_DONE, or may implement more complex logic.
WT_EN User signal which if set high will generate WT when BUSY asserted until TRANSFER_DONE indicates completion of transfer.

3.2. Functional requirements

The FMI functions are divided into four categories: segment driver, service request receiver, segment salve, and interrupt receiver. Of these, the segment driver functions are far more complex and diverse. These functions and the comparatively simple function of the service request receiver are identified under master functions

Section 3.2.1. The segment slave and interrupt receiver functions are identified as slave functions in Section 3.2.2.

3.2.1. Master Functions

The FMI shall support all mandatory IEEE 960 master specifications. The FMI shall generate bus arbitration cycles and geographical, logical, and broadcast address cycles. Supported data cycles shall include single word, secondary address, block, and pipeline cycles.

3.2.1.1. FASTBUS Cycles

The FMI shall provide building blocks from which arbitrary FASTBUS transactions may be constructed. These primitive operations shall consist of arbitration, address, and data cycles. Each 64-bit word from the processor or list sequencer may specify one such cycle. An operation code shall determine which cycle shall be executed, and the remaining bits shall supply other required parameters. The state of AS and GK following each cycle shall be considered an attribute of the primitive operation (see Sections 3.2.1.1.3.8 and 3.2.1.1.3.9).

3.2.1.1.1. Arbitration Cycle

An arbitration cycle shall be executed when the FMI receives the arbitrate cycle command from the processor or list sequencer and GK = 0. The format for the arbitrate command shall be as follows:
Arbitration cycle command

Bits	Value	Description of field
63:62	1	Operation code
61:56	0	
55	0,1	Hold GK after arbitration cycle
54	0,1	Arbitration timeout filter
53:8	X	Unused
7	0,1	Assured access enable
6	0,1	Prioritized access enable
5:0	1..63	Arbitration level

Bits 7 to 0 shall be loaded into the corresponding bits of CSR8. At the discretion of the user, it shall be possible to perform arbitration cycles without executing any address or data cycles. Failure to successfully arbitrate for the bus shall result in continued reties until arbitration timeout occurs as specified in Section 3.3.1.3.1. The arbitration timeout filter defined by bit 54 shall determine whether an exception is generated or not.

3.2.1.1.2. Address Cycle

An address cycle shall be executed when the FMI receives the address cycle command from the processor or list sequencer, GK = 1, and the FMI is the bus master. The format for the address command shall be as follows:

Address cycle command

Bits	Value	Description
63:62	2	Operation code
61:59	0..7	MS code for address cycle
58	0,1	Generate EG
57	0,1	Generate parity
56	0,1	Hold AS after address cycle
55	0,1	Hold GK after address cycle
54	0,1	Enable advanced MS codes
53:47	X	SS response exception filter
46:44	X	Timeout exception filter
43	0,1	Parity error from slave exception filter
42:34	X	Unused
33:32	0..2	Data reference mechanism
31:0	X	Data reference

The MS code and EG for the resulting address cycle shall be specified by bits 61:59 and bit 58, respectively.
At the discretion of the user, it shall be possible to perform address cycles without executing any data cycles.

3.2.1.1.3. Data Cycle

A data cycle shall be executed when the FMI receives the data cycle command from the processor or list sequencer, AK = 1, and the FMI is the bus master. The format for the data command shall be as follows:

Data cycle command

Bits	Value	Description
63:62	3	Operation code
61:59	0..7	MS code for data cycle
58	0,1	RD selector
57	0,1	Generate parity
56	0,1	Hold AS after data cycle
55	0,1	Hold GK after data cycle
54	0,1	Enable advanced MS codes
53:47	X	SS response exception filter
46:44	X	Timeout exception filter
43	0,1	Parity error from slave exception filter
42:34	X	Unused
33:32	0..2	Data reference mechanism
31:0	X	Data reference

The MS code and RD for the resulting address cycle shall be specified by bits 61:59 and bit 58, respectively.

Data cycles shall be initiated at DS(t) and terminated at DK(t). For read cycles (RD = 1), the FMI shall latch the slave supplied data at data cycle termination. For write cycles (RD = 0), the FMI shall assert data on the AD bus before cycle initiation as required by IEEE 960.

Data cycles function shall be sensitive to the phase of DS at DS(t) and to the MS mode at DS(t). The following table defines the function for each possibility:

Data cycle type

MS	DS(u)	DS(d)
0	Single word	Bus cleanup
1	Block	Block
2	Load NTA(PA) for RD = 0 Read NTA for RD = 1	Bus cleanup
3	Pipeline	Pipeline
4	Bus cleanup	Single word
5	Protective buffer	Protective buffer
6	Bus cleanup	Load NTA(PA) for RD = 0 Read NTA for RD = 1
7	Load NTA(PA~) for RD = 0 Read CSR0 for RD = 1	Load NTA(PA~) for RD = 0 Read CSR0 for RD = 1

References to NTA(PA) denote a space switch to that space selected at primary address time, coupled with load of the NTA register. Reference to NTA(PA~) is similar except that space selected is opposite to that selected at primary address time. If the advanced MS codes are disabled, then MS codes 4-7 are reserved and the FMI shall not use them.

3.2.1.1.3.1. Single Word Cycle

The single word cycle shall always be selected by the data cycle command with MS field set to 0. If the advanced MS codes are enabled, then the FMI shall optimize FASTBUS performance by issuing either MS = 0 or MS = 4.

3.2.1.1.3.2. Secondary Address Cycle for NTA(PA)

The secondary address cycle for the space selected at address time shall always be selected by the data cycle command with MS field set to 2. If the advanced MS codes are enabled, then the FMI shall optimize FASTBUS performance by issuing either MS = 2 or MS = 6. Read secondary address commands shall not be sensitive to the space selected at address time and shall not change the selected space.

3.2.1.1.3.3. Secondary Address Cycle for NTA(PA~)

The secondary address cycle for the space opposite to that selected at address time shall be selected by the data cycle command with MS field set to 7 and with RD = 0. If the advanced MS codes are not enabled, then the FMI shall issue an exception indicating an illegal command.

The read secondary address command shall not be sensitive to the space selected at address time. MS = 2 shall be used to read the secondary address register. MS = 7 with RD = 1 shall have a separate function (see Section 3.2.1.1.3.4).

3.2.1.1.3.4. CSR0 Read Cycle

The CSR0 read cycle shall be selected by the data cycle command with MS field set to 7 and with RD = 1. If the advanced MS codes are not enabled, then the FMI shall issue an exception indicating an illegal command.

The read CSR0 command shall not be sensitive to the space selected at address time and shall not change the selected space.

The implementation of this advanced mode CSR0 read cycle shall in no way interfere with standard technique for reading CSR0 using a primary address cycle to CSR space, a secondary address write cycle with value zero, and finishing with a single word cycle to read.

3.2.1.1.3.5. Block or Pipeline Cycle

Block transfers or pipeline transfers shall include one additional 64-bit word to define the block length for the transfer. The format for the block length shall be as follows:

Block length

Bits	Value	Description
63	0,1	Enable NTA post increment
62	0,1	Enable use of blocklets
61:48	0..16K	Pipelined transfer data cycle clock time
47:32	0..64K	Blocklet size
31:0		Block length in 32-bit words

The block/pipeline command shall not be executed until the block length is received.

Block transfer cycles shall be selected by the data cycle command with MS field set to 1.

Pipeline transfer cycles shall be selected by the data cycle command with MS field set to 3. The pipelined transfer data cycle clock time

shall be the count of TIMER_IN(u) transitions allowed during each data cycle.
The block or pipeline command shall resolve the reference address according to the specified reference mechanism and then copy the address and block length into internal DMA registers.

3.2.1.1.3.6. Protective Buffer Cycle
The protective buffer cycle shall be selected by the data cycle command with MS field set to 5. If the advanced MS codes are not enabled, then the FMI shall issue an exception indicating an illegal command.

3.2.1.1.3.7. Bus cleanup Cycle
A bus cleanup cycle shall be generated when a read cycle is followed by a write data cycle.

3.2.1.1.3.7.1. Explicit
The bus cleanup cycle shall be selected by the data cycle command with MS field set to 4. If the advanced MS codes are enabled, then the FMI shall optimize FASTBUS performance by selecting either MS = 4 or MS = 0. If the advanced MS codes are not enabled, then the FMI shall issue an exception indicating an illegal command.

3.2.1.1.3.7.2. Implicit
A bus cleanup cycle shall be inserted at DS(d) after a data cycle command that begins on DS(u) and enables the advanced MS codes if it is followed by a data cycle command that disables the advanced MS codes.
If the data cycle command does not enable the advanced MS codes, a bus cleanup cycle shall be generated following each single word or secondary address cycle.

3.2.1.1.3.8. AS Release
During all cycle commands following an primary address cycle, AS = 1 shall be maintained if bit 56 of the cycle command is set. AS = 0 shall be maintained after a cycle where bit 56 of the cycle command is reset.
Once AS(d) occurs, the FMI shall maintain AS = 0 until another primary address cycle successfully completes independent of the state of bit 56 of the cycle command.

3.2.1.1.3.9. Bus Release

During all cycle commands following arbitration, GK = 1 shall be maintained if bit 55 of the cycle command is set. GK = 0 shall be maintained after a cycle where bit 55 of the cycle command is reset. If GK = 0 is currently maintained, GK may be switched to GK = 1 at any cycle command assuming that the FMI remains the current bus master and that bit 55 of the cycle command is set.

3.2.1.1.3.10. Generate Parity

For address cycles and data write cycles, bit 57 shall enable parity generation. If set, parity, PA, shall be generated, and PE = 1. If reset, then PE = 0 and PA = 0.

3.2.1.1.3.11. Enable Advanced MS Codes

For address and data cycles, bit 54 shall enable use of the advanced MS codes. If bit 54 is set, then the advanced MS codes shall be enabled. This document does not yet specify the use of advanced modes for address cycles. See section 3.2.1.1.3 for the use of advanced modes during data cycles.

3.2.1.2. Control Operations

A control operation shall be executed when the FMI receives a command from the processor or list sequencer, and the format for the command is as follows:

Control command

Bits	Value	Description
63:62	0	Operation code
61:54	0..255	Control function code
53:0	X	Control parameters

The control function code field shall provide the means to decode additional commands other than FASTBUS cycle commands.

3.2.1.2.1. Reset Bus Command

The reset bus command shall be executed when the FMI receives a command from the processor or list sequencer, and the format for the command is as follows:

Reset bus command

Bits	Value	Description
63:54	1	Function control code
53:0	X	Unused

The reset bus command shall set RB = 1, and after an interval defined in IEEE 960, shall reset RB = 0. The FMI shall not be required to be bus master for the reset bus operation.

3.2.1.2.2. Write Parameter Command

The write parameter command shall be executed when the FMI receives a command from the processor or list sequencer, and the format for the command is as follows:

Reset bus command

Bits	Value	Description
63:54	2	Function control code
53:46	0..255	Base register offset address
46:34	X	Unused
33:32	0..2	Data reference mechanism
31:0	X	Data reference

The write parameter command shall write the parameter from the location specified from the location specified through the data reference mechanism to the FMI internal register specified by the base register offset address. See Section 3.3.1 for a list of the displacements.

3.2.1.2.3. Read Parameter Command

The read parameter command shall be executed when the FMI receives a command from the processor or list sequencer, and the format for the command is as follows:
Reset bus command

Bits	Value	Description
63:54	3	Function control code
53:46	0..255	Base register offset address
46:34	X	Unused
33:32	0..2	Data reference mechanism
31:0	X	Data reference

The read parameter command shall copy the parameter specified by the base register offset address to the location specified through the data reference mechanism. See Section 3.3.1 for a list of the displacements. An immediate reference shall be illegal for the read parameter command. If attempted, an exception shall indicate an illegal command.

3.2.1.2.4. Write FASTBUS Lines Command

The FMI could have an extremely primitive capability to write the 60 FASTBUS protocol lines enabling operation in a mode often useful for debugging hardware. It remains to determine whether this level of operation is desirable for the FMI.

3.2.1.2.5. Read FASTBUS Lines Command

The FMI could have an extremely primitive capability to read the 60 FASTBUS protocol lines enabling operation in a mode often useful for debugging hardware. It remains to determine whether this level of operation is desirable for the FMI.

3.2.1.2.6. Execute List Command

The execute list command shall be executed when the FMI receives a command from the processor and the format for the command is as follows:

Execute list command

Bits	Value	Description
63:54	8	Function control code
53:32	X	List length in 32-bit words
31:0	X	List address

The execute list command shall include one additional 64-bit word to define the storage for the status history. The format for the status history storage shall be as follows:

Status history storage

Bits	Value	Description
63:32		Block length in 32-bit words
31:0		Storage address

The execute list command shall not be executed until the status history storage is received.
The execute list command shall copy the list address, list length, status history storage address, and status history storage length into internal DMA registers, and initiate the list sequencer for list-directed operation. The processor interface shall be disabled for commands until the list execution either terminates or aborts with an exception.
If the execute list command is encountered during list-directed execution, the command shall be treated as a null operation.

3.2.1.2.7. Terminate List Command

The terminate list command shall be executed when the FMI receives a command from the list sequencer, and the format for the command is as follows:

Terminate list command

Bits	Value	Description
63:54	9	Function control code
53:0	X	Unused

The terminate list command shall terminate list sequencer activity, and enable the processor interface for commands. If received from the processor, this command shall be treated as a null operation.

3.2.1.2.8. Reset Segment Driver Command

The reset segment driver command shall be executed when the FMI receives a command from the processor or list sequencer, and the format for the command is as follows:

Reset segment driver command

Bits	Value	Description
63:54	10	Function control code
53:0	X	Unused

The reset segment driver command shall act as a programmable reset command. The segment driver shall reset its internal state to that obtained after power-up initialization. Any list sequencer activity shall terminate and the processor interface shall be enabled to receive commands.

3.2.1.2.9. Reset Interrupt Receiver Command

The reset interrupt receiver command shall be executed when the FMI receives a command from the processor or list sequencer, and the format for the command is as follows:

Reset segment receiver command

Bits	Value	Description
63:54	11	Function control code
53:0	X	Unused

The reset interrupt receiver command shall act as a programmable reset command. The interrupt receiver shall reset its internal state to that obtained after power-up initialization.

3.2.1.2.10. Enable/Disable Interrupt Receiver Command
The enable/disable interrupt receiver command shall be executed when the FMI receives a command from the processor or list sequencer, and the format for the command is as follows:
Enable/disable interrupt receiver command

Bits	Value	Description
63:54	12	Function control code
53	0,1	Set to enable, reset to disable
52:0	X	Unused

The enable/disable interrupt receiver command shall either enable or disable interrupt receiver messages depending on whether bit 53 is set or reset, respectively.

3.2.1.2.11. Reset Service Request Receiver
The reset service request receiver command shall be executed when the FMI receives a command from the processor or list sequencer, and the format for the command is as follows:
Reset service request receiver command

Bits	Value	Description
63:54	13	Function control code
53:0	X	Unused

The reset service request receiver command shall act as a programmable reset command. The service request receiver shall reset its internal state to that obtained after power-up initialization.

3.2.1.2.12. Enable/Disable Service Request Receiver Command
The enable/disable service request receiver command shall be executed when the FMI receives a command from the processor or list sequencer, and the format for the command is as follows:
Enable/disable service request receiver command

Bits	Value	Description
63:54	14	Function control code
53	0,1	Set to enable, reset to disable
52:0	X	Unused

The enable/disable service request receiver command shall either enable or disable service requests depending on whether bit 53 is set or reset, respectively.

3.2.1.3. List Processing

3.2.1.3.1. Flow Control

List-directed processing shall be initiated under processor control by performing the execute list command. Thereafter, the list sequencer shall supply commands in sequence starting with the address specified by the execute list command. No branching or looping shall be supported. List execution ends with the execution of the terminate list command or with an occurrence of an exception. At this time, control shall revert to the processor.

3.2.1.3.2. Data Control

During list-directed processing, the list sequencer shall fetch/store data transferred by each FASTBUS cycle or control operation.

3.2.1.4. Data Access Mechanisms

Address and data cycles shall reference data by one of three address mechanisms: immediate, address reference, or stack reference. Parameter read and write operations shall also use these same mechanisms.

3.2.1.4.1. Immediate

The immediate mechanism shall be restricted to write operations, i.e. address cycles and write data cycles. This mechanism shall supply a copy of the data value to be used in the FASTBUS cycle. No additional memory fetch shall be required.

3.2.1.4.2. Address Reference

The address reference mechanism shall supply the address of data or a buffer to use for data transfer. Data shall be transferred to/from the FMI with additional memory access cycles that follow the fetch for the command.

3.2.1.4.3. Stack Reference

The stack reference mechanism shall supply the address of a memory resident control structure that maintains the current access address for a buffer area and the address of the end of the buffer. The current access address shall be revised after each command referencing the stack. Stack overflow shall cause an exception. The user application shall ensure that the stack control structure is initialized prior to use of the stack reference mechanism.

3.2.1.5. Direct Memory Access (DMA) Controller

3.2.1.5.1. Memory Management

The FMI shall not internally support any memory management algorithms. All address visible to the FMI shall be absolute addresses in memory. The user application shall be responsible for any address translations that may be required prior to invoking either processor-directed or list-directed I/O.

3.2.1.5.2. Transfer Address and Transfer Count

The list sequencer shall maintain a pair of registers for each of the three types of memory storage structures required by the FMI. These register pairs shall consist of the next transfer address (TA) and the transfer count (TC). The TA shall be the byte address of the first word to transfer. The TC shall be the number of bytes to transfer. Each DMA transfer shall cause TA and the TC to be post-incremented.

The TA/TC registers shall be defined prior to DMA utilization to avoid erroneous operation. When the execute list command is performed, the TA and TC for list access and for status history shall be initialized. Each time a block or pipeline transfer cycle command is performed, the TA and TC for block data access shall be initialized.

3.2.1.5.3. Use with Stack Data Reference Mechanism

When the stack reference mechanism is used, the control structure at the pointer address shall be accessed, the current transfer address extracted, and the transfer address deposited in the appropriate TA register. The block length shall be added to the current transfer address and compared with the end-of-stack address to verify adequate capacity in the stack. If space is insufficient, then the FMI shall generate the insufficient stack space exception.

3.2.1.6. Exceptions

The FMI shall notify the processor that an error has occurred by issuing an exception. The interrupt protocol shall have the FMI first set the interrupt signal. The interrupt shall remain pending until the processor sets the acknowledge interrupt signal. Then the FMI shall reset the interrupt signal. If the FMI has another interrupt pending, it shall set the interrupt signal only after the processor has reset the interrupt acknowledge signal.

When the FMI sets the interrupt signal, it shall latch an encoded interrupt source identifier in the status register. The status register shall record the error source generating the exception, so that processor software shall be able to identify that source. The status register is discussed further in Section 3.3.1.2. The encoded values are as follows:

Encoded exception sources

Encoded value	Interrupt source
0	No interrupt
1	SS = 1 during address cycle
2	SS = 2 during address cycle
3	SS = 3 during address cycle
4	SS = 4 during address cycle
5	SS = 5 during address cycle
6	SS = 6 during address cycle
7	SS = 7 during address cycle
8	Timeout during address cycle
9	SS = 1 during data cycle
A	SS = 2 during data cycle
B	SS = 3 during data cycle
C	SS = 4 during data cycle
D	SS = 5 during data cycle
E	SS = 6 during data cycle
F	SS = 7 during data cycle
10	Timeout during data cycle
11	Parity error from slave during data cycle read
12	Arbitration timeout
13	WT timeout
14	Overflow
15	Insufficient stack space for transfer allocation
16	Illegal command
17-3F	Reserved

3.2.1.6.1. FASTBUS Cycles

Each FASTBUS cycle shall specify a filter that defines what SS-response codes, timeouts, etc. shall generate exceptions. If list-directed I/O is in progress, then an exception shall always terminate execution of the list. At the discretion of the user application, the filter may be set not to issue an exception for a non-zero SS-response or timeout. Independent of exception generation, each non-zero SS-response or timeout shall transfer the current list address of the FMI

command cycle to the current location in the status history buffer. The FMI shall then deposit the interrupt source in the next location in the status history buffer. Both transfers shall be made under DMA control.
If processor-directed I/O is in progress, no exceptions shall be ignored and nothing shall be transferred into the status history buffer. The processor shall be responsible to check the status register for errors.

3.2.1.6.1.1. SS Response Codes
If not masked by the SS-response filter, all non-zero SS codes shall generate exceptions. For SS = 1 on address or data cycles, the exception shall be delayed until the operation has been retried the number of times specified in the retry count register (see Section 3.3.1.3.1). The SS-response filter shall have the following format within the command:

SS-response exception filter

Bit	Value	Function
53	0,1	Enable SS = 7 exception
52	0,1	Enable SS = 6 exception
51	0,1	Enable SS = 5 exception
50	0,1	Enable SS = 4 exception
49	0,1	Enable SS = 3 exception
48	0,1	Enable SS = 2 exception
47	0,1	Enable SS = 1 exception

3.2.1.6.1.2. Timeout

If not masked by the timeout response filter, either WT, AK, or DK timeout shall generate an exception. The timeout filter shall have the following format within the command:

Timeout exception filter

Bit	Value	Function
46	0,1	Enable WT timeout
45	0,1	Enable AK timeout
44	0,1	Enable DK timeout

3.2.1.6.1.3. Parity Error from Slave
If not masked by the slave parity error response filter, a parity error on a data cycle read operation shall generate an exception.

3.2.1.6.1.4. Arbitration Timeout
If not masked by the arbitration timeout filter, arbitration timeout shall generate an exception.

3.2.1.6.2. Overflow
Overflow shall occur whenever the list sequencer attempts to transfer another command, data word, or status history and the corresponding length register has already been counted to zero. Overflow shall occur whenever the list sequencer attempts to store beyond the end-of-stack while referenced with the stack addressing mechanism.
Overflow shall always generate an exception.

3.2.1.6.3. Illegal Command

3.2.1.6.3.1. MS Code Violation
It shall be illegal to send the FMI a command with the advanced MS codes not enabled and with MS > 3. Such commands shall always generate an exception.

3.2.1.6.3.2. Immediate Read
It shall be illegal to execute any read data command cycle with an immediate data reference mechanism. Such commands shall always generate an exception.

3.2.1.7. Service Request Receiver
Upon receiving SR(u), the FMI shall issue a SR interrupt to the processor. The FMI shall not attempt to service the request itself, but merely pass it along to the processor. It shall be the responsibility of the processor to block further SR interrupts until the source of the request has been identified, serviced, and cleared. SR interrupts shall be independent of activity in the segment driver or the interrupt message receiver.

3.2.2. Slave Functions
The FMI shall support all mandatory IEEE 960 slave specifications. The FMI shall handle geographical, logical, and broadcast addressing using on-chip registers in conjunction with the latter two modes. Supported data cycles shall include random, secondary address, block, and pipeline access. The application independent bits of CSR0 shall be implemented on-chip.

3.2.2.1. Address Cycles

The FMI shall support all IEEE 960 addressing modes, and CSR registers required for those modes. While the geographical address mode has no special requirements beyond supplying the geographical address, GA, to the FMI, the logical address mode requires an on-chip CSR3 register, the on-chip bit CSR0<1>, and the bit count for the range of the internal address for the module using five encoded pins on the FMI, LOGICAL_WIDTH(4:0). The broadcast address mode requires an on-chip CSR7 (class N broadcast) register, and an external pin, CONNECT_CASE_8, for T-pin control for the user specified case 8 broadcast mode.

The FMI shall provide the user application with address status indicators. A single external pin, ATTACHED, shall indicate when the FMI is the attached slave. When attached, the FMI shall further supply a 4-bit encoded value, ATTACH_MODE(3:0), representing the address mode at connect time. The table below lists the connect mode along with the encoded connect values. If the connect mode indicates that the connection results from a broadcast address case 8, which is user defined, the user application shall set CONNECT_CASE_8 to direct whether the connection will be established.

Address connect codes for the FMI. Code shall be determined at primary address time.

Attach Code	Primary address connection
0	Geographical address
1	Logical address
8	General broadcast (case 1)
9	Class N broadcast (case 2)
10	Broadcast case 3
11	Broadcast case 3a
12	Broadcast case 4
13	Broadcast case 5
14	Broadcast case 6
15	Broadcast case 8 (user defined)

Independent of addressing mode, the FMI shall initiate address connections by generating the encoded attach code, a signal reflecting the type of space addressed (either CSR or data space), a signal to indicate that an address cycle is in progress, a 32-bit wide address bus, ADDRESS(31:0), with the contents of the NTA register, and signal to trigger the user-application timing control, RESPONSE_CTL_OUT. The user logic shall validate the address cycle, generate any controls

for generation of non-zero SS response, and upon seeing the trigger, return a handshake signal, RESPONSE_CTL_IN, to initiate the AK response. Note that for broadcast addresses, the handshake shall be asserted as for other address cycles, but the AK shall be suppressed internally by the FMI. See the interface Section 3.1.6.2 for additional detail concerning required timing delays for RESPONSE_CTL_OUT and RESPONSE_CTL_IN handshake signals.
Both CSR3 and CSR7 shall be implemented in the FMI to provide the maximum flexibility for the user application. CSR3 shall be a full 32-bit wide register. CSR7 shall be only 16 bits wide as required by the IEEE 960 specification. The registers shall be separately enabled/disabled for internal implementation on the FMI by strapping FMI option pins.

3.2.2.2. Data Cycles

The FMI shall consider an independent data cycle in progress anytime DS and DK differ. At DS(t) the data cycle shall begin. For write operations, data shall flow from the AD pins to the FMI internal bus to the buffered data bus, BAD, and in the opposite direction for read functions. On-chip registers shall be transparent during write data cycles and assert data throughout read cycles. To reduce susceptibility to noise, the RD signal shall be latched during data cycles and passed to the user application as READ. Similarly, the MS lines shall be internally latched and used to decode user-application signals BLOCK_MODE and NTA_ACCESS. BLOCK_MODE shall be asserted if either block (MS = 1) or pipeline (MS = 3) mode is invoked, and NTA_ACCESS shall be asserted if secondary address mode (MS = 2) is referenced.
Timing for off-chip accesses shall be provided through interface signals TRANSFER_REQUEST, TRANSFER_DONE, and TRANSFER_ENABLE. At DS(t) time, signal TRANSFER_REQUEST shall be driven to match DS. For all off-chip accesses TRANSFER_ENABLE shall be set. Upon seeing TRANSFER_REQUEST(t), the user application shall initiate logic to access either CSR or data space memory. When the data has been accepted or asserted as directed by READ, the user application shall drive TRANSFER_DONE to match TRANSFER_REQUEST.
If the data cycle accessed on-chip storage, TRANSFER_REQUEST shall be processed as above, but TRANSFER_ENABLE shall remain reset throughout the data cycle. The user application shall be responsible for generating the TRANSFER_DONE synchronization handshake.
The FMI shall generate its internal DK(t) upon receipt of TRANSFER_DONE(t). At this time, the protective buffer shall drive

the AD bus if a read cycle is in progress. At the same time, the NTA register shall advance if a block or pipeline mode operation is active. Then a RESPONSE_CTL_OUT/RESPONSE_CTL_IN handshake analogous to that performed during the address cycle shall follow to ensure a valid SS response is available. Should data cycles require different timing from address cycles, ADDRESS_CYCLE shall enable discrimination. After RESPONSE_CTL_IN is asserted by the user application, the FMI shall generate DK(t). The data cycle shall end at this point. The SS response and any read data shall remain asserted until any subsequent DS(t) or AS(d).
The user application shall control whether the NTA advances during block or pipeline mode accesses with the signal NTA_CARRY_IN. If NTA_CARRY_IN is set, then the NTA shall advance normally. If reset, the NTA shall hold its value unchanged.

3.2.2.3. Slave Status Response

Slave status (SS) response shall be controlled by the user application through use of five interface signals. All are applicable for data cycles, while only two control address cycles. These two signals common to address and data cycles are BUSY and NOT_VALID. An asserted BUSY signal shall indicate the user application is unable to complete the requested cycle at the current time, and either WT is set or SS = 1 shall be generated according to whether WT_EN is enabled or disabled, respectively. The NOT_VALID signal shall indicate the validity of the current NTA address. For data cycles, NOT_VALID may also mean that the data is invalid.
Other SS response control signals reserved for data cycles include REJECT, EOB, and SET_SS3. When NOT_VALID is asserted, REJECT set or reset shall control issuance of either SS = 6 or SS = 7, respectively. The EOB signal shall be generated by the user application during block or pipeline transfers when the last word in the block is encountered. By itself, the EOB signal shall not generate SS = 2. However, if block or pipeline access continues with a data cycle immediately following the data cycle that asserted EOB, then SS = 2 shall be generated. Otherwise, SS = 2 generation shall be disarmed until another EOB is observed. The response SS = 3 shall be generated when SET_SS3 is asserted.
In the event that two or more of the SS response control lines are set at once, some controls shall have priority over others. Listed from highest priority, the signals are SET_SS3, BUSY, EOB delayed from immediately preceding block or pipeline data cycle if any, and NOT_VALID. REJECT is subordinate to NOT_VALID, and has meaning only when NOT_VALID is asserted.

3.2.2.4. Interrupt Receiver

The FMI shall monitor single word, block and pipeline write cycles to all interrupt-receiving CSR blocks. Upon detecting AS(d) after slave access to any interrupt receiver block, the FMI shall issue a FASTBUS interrupt message (FIM) interrupt to the processor. The FMI shall mark the interrupt receiver block as busy, and return SS = 1 on any subsequent slave access to the interrupt receiver block until the processor has serviced and cleared the interrupt.
FIM interrupts shall be independent of activity in the segment driver or the service request receiver.

3.2.3. Register Visibility

3.2.3.1. From Processor Interface

The processor interface for the FMI shall have access to interface registers, a portion of CSR space, and processor memory.

3.2.3.2. From list sequencer Interface

The processor interface for the FMI shall have access to interface registers, a portion of CSR space, and processor memory.

3.2.3.3. From FASTBUS

The FASTBUS slave interface for the FMI shall have access to all CSR and data space. No access shall be provided to interface registers.

3.2.4. Contention Resolution

3.2.4.1. Processor vs. Slave Interface

When the processor is addressing the FMI and the FMI is not FASTBUS master, then it is possible that a FASTBUS slave data cycle could conflict with the processor bus activity. In this case, the slave shall have priority, and the processor shall wait.

3.2.4.2. Processor-Directed vs. List-Directed I/O

When the list sequencer is processing a list of FMI commands, the processor may require interface status information. In this case, the processor shall be given priority to read the status register. No other access shall be permitted.

3.3. Internal Structure Requirements

3.3.1. Memory-Mapped Registers

All processor/list interface registers shall be memory mapped. The base address register shall be defined for the FMI through a serial input initialization procedure. Displacements from the base address register are specified for each of the internal registers in the table below:

Byte displacements for the interface register set.

Displacement	Register
0	Command
8	Status
12	Retry count
14	Arbitration timeout
16	List address
20	List length
24	Block data address
28	Block data length
32	History status address
36	History status length
40	CSR8 - Arbitration level register
41	CSR9 - Timer control register
42	Reserved
43	CSR1D - Wait timer

44 CSR1E - Address timer
45 CSR1F - Data timer

3.3.1.1. Command Register

The command register shall receive 64-bit values from the DATA_IN bus. Regardless of whether loaded by the processor or the list sequencer, a write to the command register shall result in command execution. If bits 62 and 63 are both zero, then the FMI shall execute a control operation as specified by Section 3.2.1.2. Otherwise, a FASTBUS cycle shall be executed as specified in Section 3.2.1.1.
A read of the command register shall not be supported.

3.3.1.2. Status Register

The status register shall report to the processor the current state of the FMI interface. The 32-bit status register shall be asserted of the DATA_OUT bus during read operations that address the status register. A write of the status register shall not be supported.
The format for the status register shall be as follows:
Status register

Bit	Description
31	List processing in progress
30	FASTBUS slave access in progress
29	Timeout on last processor-directed cycle
28	Parity error from slave on last cycle
27:25	SS code on last processor-directed cycle
24	FIM interrupt requesting
23	SR interrupt requesting
22	Exception interrupt requesting
21	FIM interrupt enabled
20	SR interrupt enabled
19	Exception interrupt enabled
18:16	Reserved
15:12	FIM receiver block for requesting interrupt
11:6	Reserved
5:0	Encoded value for last exception generated

3.3.1.3. FASTBUS Parameters

The following FASTBUS parameters shall affect all FASTBUS users. Consequently, reasonable system wide parameters shall be established and loaded during system initialization. Thereafter, these parameters should be regarded as constants. Both read and write access for these parameters shall be supported.

3.3.1.3.1. Retry Count

The retry count shall specify the maximum number of times the FMI attempts to complete an address or data cycle when the attached slave returns SS = 1. When this number of retries has been exhausted, the FMI shall issue an exception if enabled and not masked.
The retry count shall be saved in a 16-bit register.

3.3.1.3.2. Arbitration Timeout

The arbitration timeout shall define the maximum count of TIMER_IN(u) transitions allowed without obtaining bus mastership before asserting the arbitration timeout exception.
The arbitration timeout shall be saved in a 16-bit register.

3.3.1.4. Next List Address Register

The next list address (NLA) register shall point to the address in memory that holds the next command for list-directed I/O. The NLA shall be initially loaded by the execute list command, and shall be incremented by 8 following each command fetch.
The NLA register shall be saved in a 32-bit register.

3.3.1.5. List Length Register

The list length register shall hold the number of bytes remaining in the command list. The list length shall be initially loaded by the execute list command, and shall be decremented by 8 following each fetch command. The list processing shall automatically terminate without exception when a command completes with the list length counted down to zero.
The list length register shall be saved in a 32-bit register.

3.3.1.6. Next Block Address Register

The next block address (NBA) register shall point to the address in memory that accepts/asserts the data for the next FASTBUS transfer during a block or pipeline transfer. The NBA shall be initially loaded by the block or pipeline cycle command, and shall be incremented by 4 following each command fetch.
The NBA register shall be saved in a 32-bit register.

3.3.1.7. Block Length Register

The block length register shall hold the number of bytes remaining in the block transfer operation. The block length shall be initially loaded by the block or pipeline cycle command, and shall be decremented by 4 following each transfer operation. The block or pipeline cycle shall automatically terminate with an overflow

exception when a transfer completes with block list length counted down to zero.
The block length register shall be saved in a 32-bit register.

3.3.1.8. Next Status History Address Register

The next status history address (NSHA) register shall point to the address in memory that accepts the slave response code for the next SS 0 generated during FASTBUS address or data cycles. The NSHA shall be used only during list-directed I/O. The NSHA shall be initially loaded by the execute list command, and shall be incremented by 8 following each cycle with SS 0.
Each cycle with SS 0 shall store 8 bytes of data in the status history. The 32-bit address of the offending command shall be first written to the status history, followed by the block length register packed with the 3-bit SS code. This second word shall have the following format:
Packed block length/SS code

Bits	Contents
31:3	Bits 30:2 from the block length register
2:0	SS code for the offending FASTBUS cycle

If the offending cycle is not a block or pipeline cycle, then bits 31:3 shall be zero.
The NSHA register shall be saved in a 32-bit register.

3.3.1.9. Status History Length Register

The status history length register shall hold the number of bytes remaining in the status history buffer. The status history length shall be initially loaded by the execute list command, and shall be decremented by 8 following each fetch command. The list processing shall automatically terminate with an overflow exception when a command generates SS 0 with the status history length counted down to zero.
The list length register shall be saved in a 32-bit register.

3.3.2. Instruction Format

3.3.2.1. One-Word Instruction

All commands except the block/pipeline cycle and execute list command shall use the one-word instruction format. The one-word instruction format shall consist of a 64-bit word fetched from the DATA_IN bus. For details see Sections 3.2.1.1 and 3.2.1.2.
The instruction shall be latched into the command register.

3.3.2.2. Two-Word Instruction

The two-word shall be used for the block/pipeline cycle and for the execute list command. The first word shall be analogous to the one-word instruction. The second word shall specify additional parameters. For the block/pipeline cycle command, the second word shall define the block length, blocklet enable, blocklet size if applicable, and NTA post increment enable. For the execute list command, the second word shall define history status buffer and its length.
The second word shall not overwrite the command register.

3.3.3. FASTBUS Interface

3.3.3.1. CSR Registers

The FASTBUS parameters for CSR9, CSR1D, CSR1E, and CSR1F shall affect all FASTBUS users. Consequently, reasonable system wide parameters shall be established and loaded during system initialization. Thereafter, these parameters should be regarded as constants. Both read and write access for these parameters shall be supported for the processor and FASTBUS slave interfaces.

3.3.3.1.1. CSR0

All user-application independent bits defined for CSR0 shall be supported on board the FMI. This set consists of the error flag, logical address enable, run/halt, module allocated, SR enable, SR flag, parity error, and module active (bits 0,1,2,3,4,5, 14, and 15, respectively). Different functions shall be supported for each of the bits. Toggle operations shall not be supported for internal FMI bits. The error flag read from bit 0 shall be the logical OR of the user-application signal SET_ERROR_FLAG and an internal latch that shall be written to simulate an error. The latch shall be cleared by CLEAR_ERROR, which is discussed below.

Bit 1 for control of logical addressing shall be set/reset according to IEEE 960. The status of bit 1 shall be reported to the user application with signal LOGICAL_ADDR_EN.
Bit 2 for run/halt control shall be set/reset according to IEEE 960. The status of bit 2 shall be reported to the user application with signal RUNNING. If in the run state, the user application shall change the status to halt by asserting the SET_HALT line.
Bit 3 for recording module allocation status shall be set/reset according to IEEE 960. The status of bit 3 shall not be reported to the user application.
Bit 4 for enabling SR interrupts shall be set/reset according to IEEE 960. The status of bit 4 shall not be reported to the user application.
Bit 5 for recording pending SR interrupts shall be set/reset according to IEEE 960. The status of bit 5 shall be reported to the user application with signal SR_FLAG. If SR_FLAG is reset, the user application shall request service pending by asserting SET_SR. Upon seeing SR_FLAG asserted, the user application shall reset SET_SR, and make no further SR interrupt requests until SR_FLAG is reset.
Bit 14 for reporting FASTBUS parity errors shall be set/reset according to IEEE 960. The status of bit 14 shall be reported to the user application with signal PARITY_ERROR. If PARITY_ERROR is asserted, the user application shall decide whether the error is reportable to CSR0<0> and whether the data shall be accepted or rejected. Thus signals SET_ERROR_FLAG and REJECT, respectively, shall be defined in response to PARITY_ERROR.
Bit 15 shall pass the status of the ACTIVE line supplied by the user application.
Since significant parts of CSR0 are defined by the user application, signals CSR0_LOAD and CSR0_READ shall be provided by the FMI to facilitate the application. The user application shall assert at least the device ID while CSR0_READ is asserted, and may also assert any other device-dependent bits. User-application bits corresponding to bits defined in the FMI internally shall be ignored.
When asserted during CSR0_LOAD operations, bits 16, 30, and 31 shall have special meaning. Under these conditions, assertion of bit 16, 30, and 31 shall generate signal CLEAR_ERROR, RESET, and CLEAR_DATA, respectively. These signals shall be used internally and shall also be externally supplied to the user application. Internally, CLEAR_ERROR shall reset bit 0, simulate error, and bit 14, parity error. Externally, it shall clear all sources for errors contributing to SET_ERROR_FLAG.

The scope of RESET and CLEAR_DATA is defined in IEEE 960 Section 8.18. As supplied from the FMI, RESET shall also include a term asserted when when power is applied to the module, *i.e.*

RESET = AD<30> * CSR0_LOAD + POWER_UP

CLEAR_DATA shall be derived from AD<31>*CSR0_LOAD alone.
The CSR0 shall also be affected by the RESET_BUS signal, which is supplied by the user application when the bus is halted and and reset bus is executed. Logically RESET_BUS = (BH*RB), where denotes that the term is integrated as required by the specification. Both logical addressing enable and run/halt, bits 1 and 2, respectively, shall be reset when RESET_BUS is present. This signal also shall ensure AK and DK are not asserted by unconditionally disconnecting the module.
The internal CSR0 register shall be separately enabled/disabled for implementation on the FMI by strapping CSR0_EN.

3.3.3.1.2. CSR3 - Logical Address Register
CSR3 shall be implemented internal to the FMI as a 32-bit register. It shall satisfy all requirements of IEEE 960, and shall be accessible only as a FASTBUS slave.

3.3.3.1.3. CSR7 - Class N Broadcast Register
CSR7 shall be implemented internal to the FMI as a 16-bit register. It shall satisfy all requirements of IEEE 960, and shall be accessible only as a FASTBUS slave.

3.3.3.1.4. CSR8 - Arbitration Level Register
CSR8 shall be implemented internal to the FMI as an 8-bit register. It shall satisfy all requirements of IEEE 960, and shall be accessible from the processor and FASTBUS slave interfaces.

3.3.3.1.5. CSR9 - Timer Control Register
CSR9 shall be implemented internal to the FMI as a 3-bit register. It shall satisfy all requirements of IEEE 960, and shall be accessible from the processor and FASTBUS slave interfaces.

3.3.3.1.6. CSR1D - Wait Timer
CSR1D shall be implemented internal to the FMI as an 8-bit register. It shall satisfy all requirements of IEEE 960, and shall be accessible from the processor and FASTBUS slave interfaces.
CSR1D shall define the maximum count of TIMER_IN(u) transitions allowed with WT = 1 before asserting the wait timeout exception.

3.3.3.1.7. CSR1E - Address Timer

CSR1E shall be implemented internal to the FMI as an 8-bit register. It shall satisfy all requirements of IEEE 960, and shall be accessible from the processor and FASTBUS slave interfaces.
CSR1E shall define the maximum count of TIMER_IN(u) transitions allowed with AS = 1 and AK = 0 before asserting the address timeout exception.

3.3.3.1.8. CSR1F - Data Timer

CSR1F shall be implemented internal to the FMI as an 8-bit register. It shall satisfy all requirements of IEEE 960, and shall be accessible from the processor and FASTBUS slave interfaces.
CSR1F shall define the maximum count of TIMER_IN(u) transitions allowed with DS-DK before asserting the data timeout exception.

3.3.3.1.9. CSR100-1FF

The FMI shall support up to 16 external interrupt receiver blocks as defined in IEEE 960. If an interrupt receiver block is attached as a slave, then at AS(d) the FMI shall issue a FIM interrupt and latch in the status register the number of the accessed receiver block.

3.3.3.2. Data Space

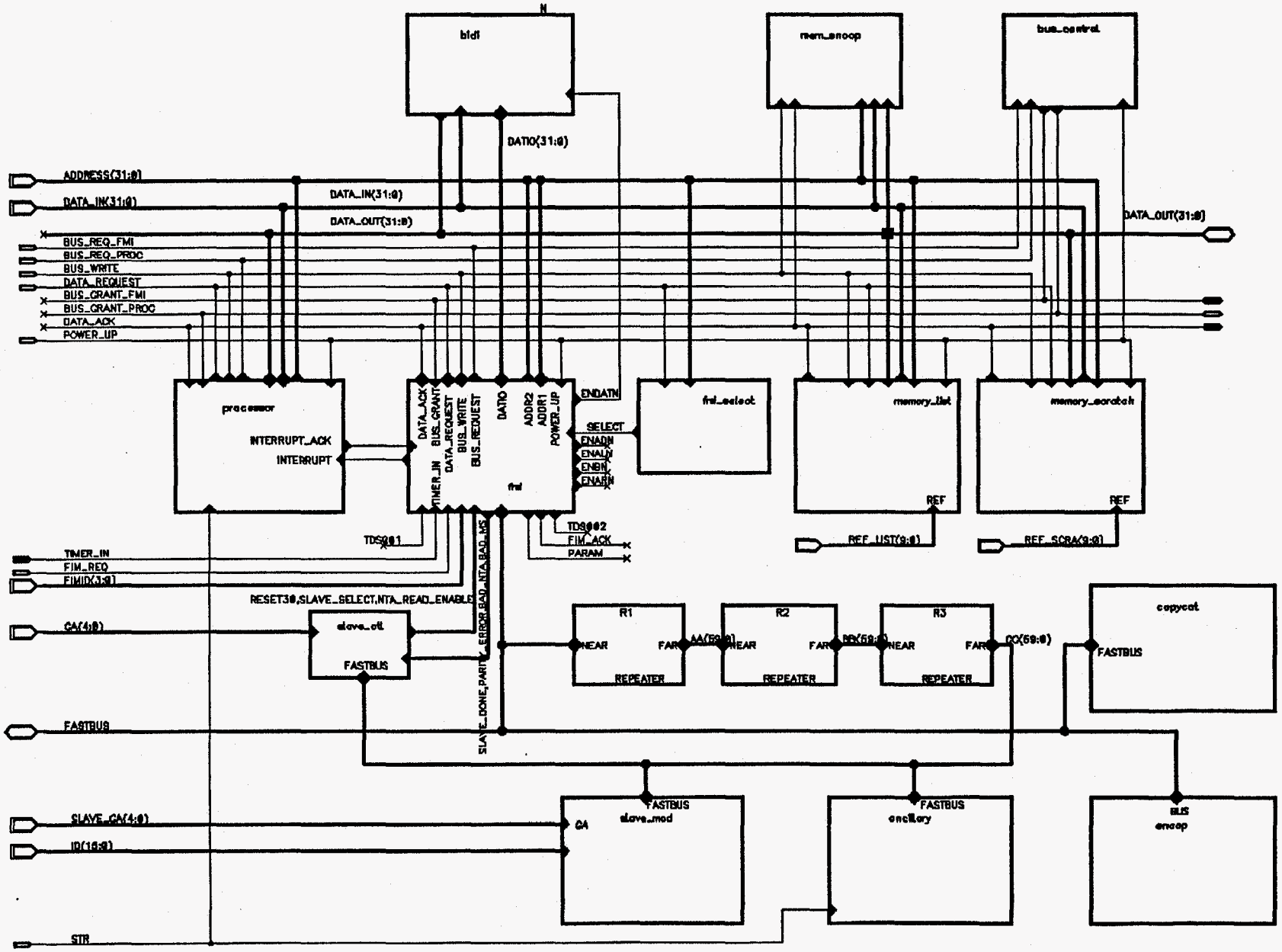
All data space shall be external to the FMI.

APPENDIX D

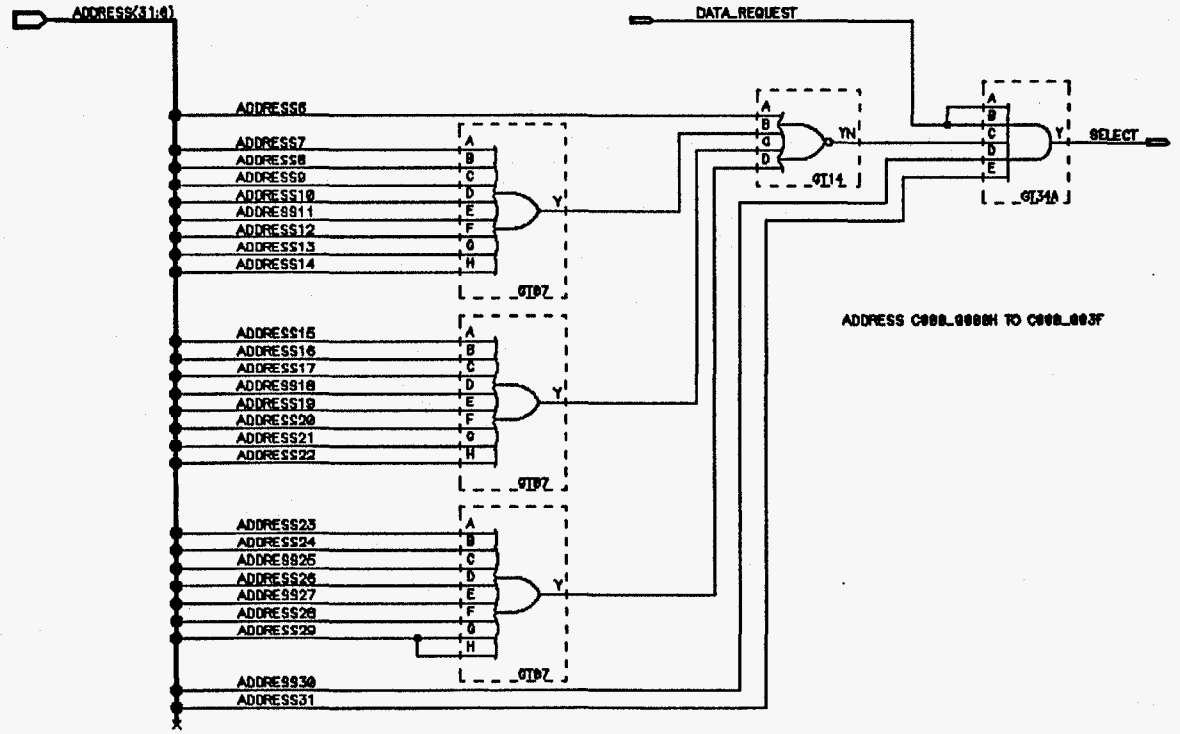
FMI SIMULATION TECHNIQUES

SIMULATION OF THE FMI GATE ARRAY

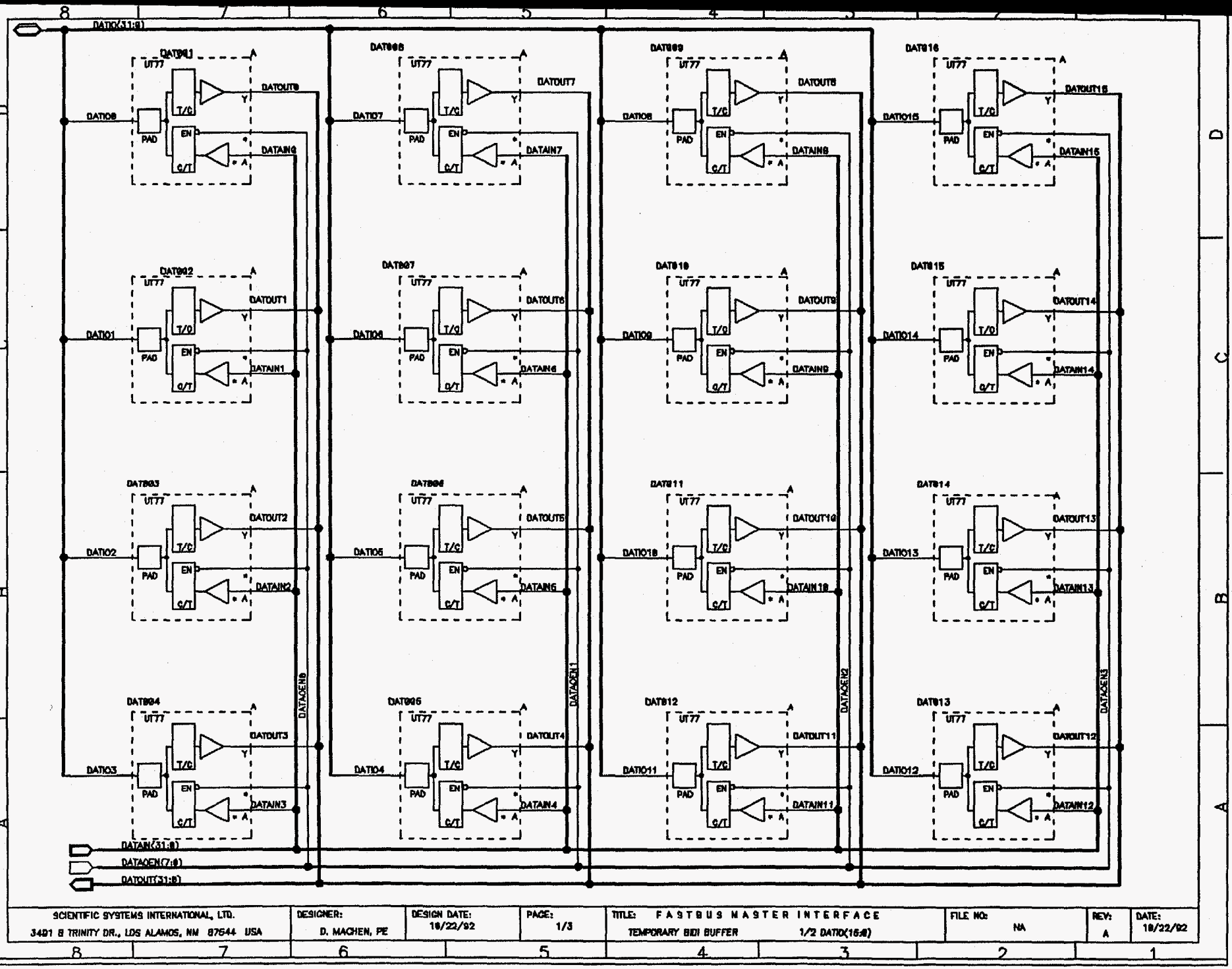
The schematics contained on pages 79-85 represent the "top" simulation level for the FMI gate-array design. Simulation in IC design is even more important than standard-logic design, as you cannot probe the circuit under test and modify the prototype to correct a subtile timing error, for example. The design must be correct the first time in order to not incure additional NRE charges required to rerun a corrected design through the prototype fabrication process. Page79 shows a number of blocks that were constructed with a behavioral modeling language to operate and test the master controller design in a FASTBUS environment. Some of the blocks are represented by logic schematics on pages 80-85, but the remainder are software models. The test-vector set that was to have been submitted to AMCC with the completed design was built using the simulation environment and the FMI design and were actual FASTBUS operations rather than arbiturary patterns applied to the inputs of the design.



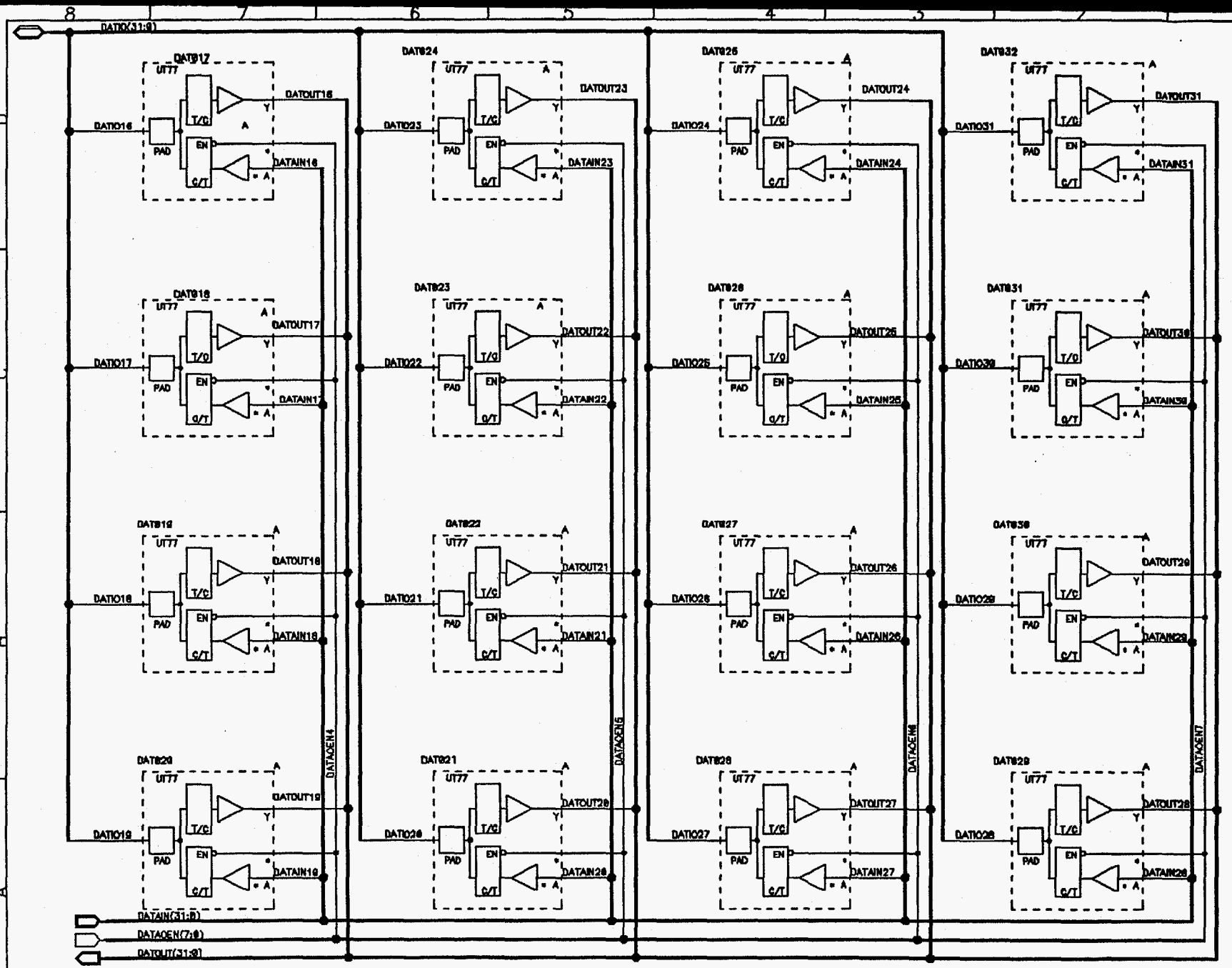
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: DRM RON	DESIGN DATE: 5/5/90	PAGE: 1/1	TITLE: FASTBUS RISC PROCESSOR MODULE SIMULATION BLOCK 1			FILE NO: SSI/JV-SIMFMI1	REV: B	DATE: 2/6/91
8	7	6	5	4	3	2	1		



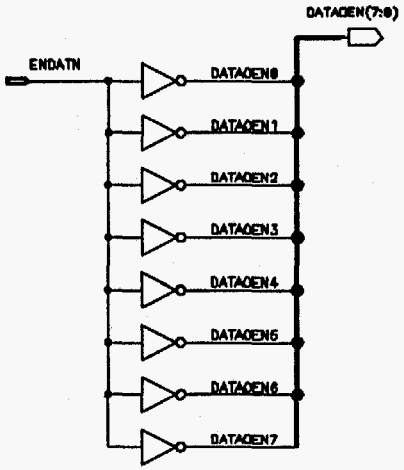
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: RON . DRM	DESIGN DATE: 1/13/98	PAGE: 1/1	TITLE: FASTBUS RISC PROCESSOR MODULE SIMULATION HARDWARE	FILE NO: SSI/JV-SIMHDW1	REV: C	DATE: 2/6/91



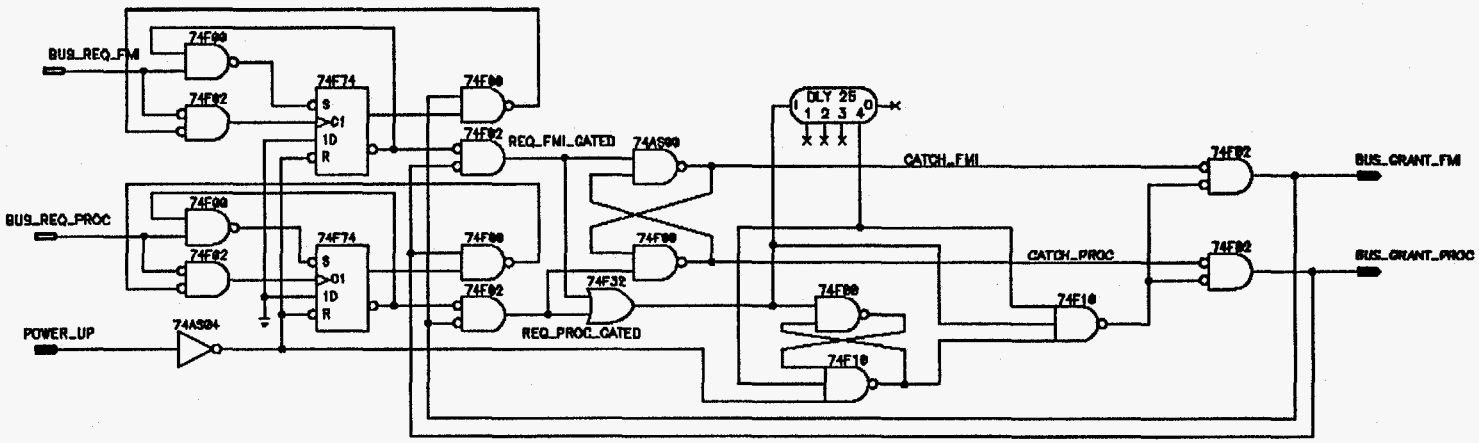
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3601 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 10/22/92	PAGE: 1/3	TITLE: FASTBUS MASTER INTERFACE TEMPORARY BIDI BUFFER 1/2 DATIO(15:0)	FILE NO: NA	REV: A	DATE: 10/22/92



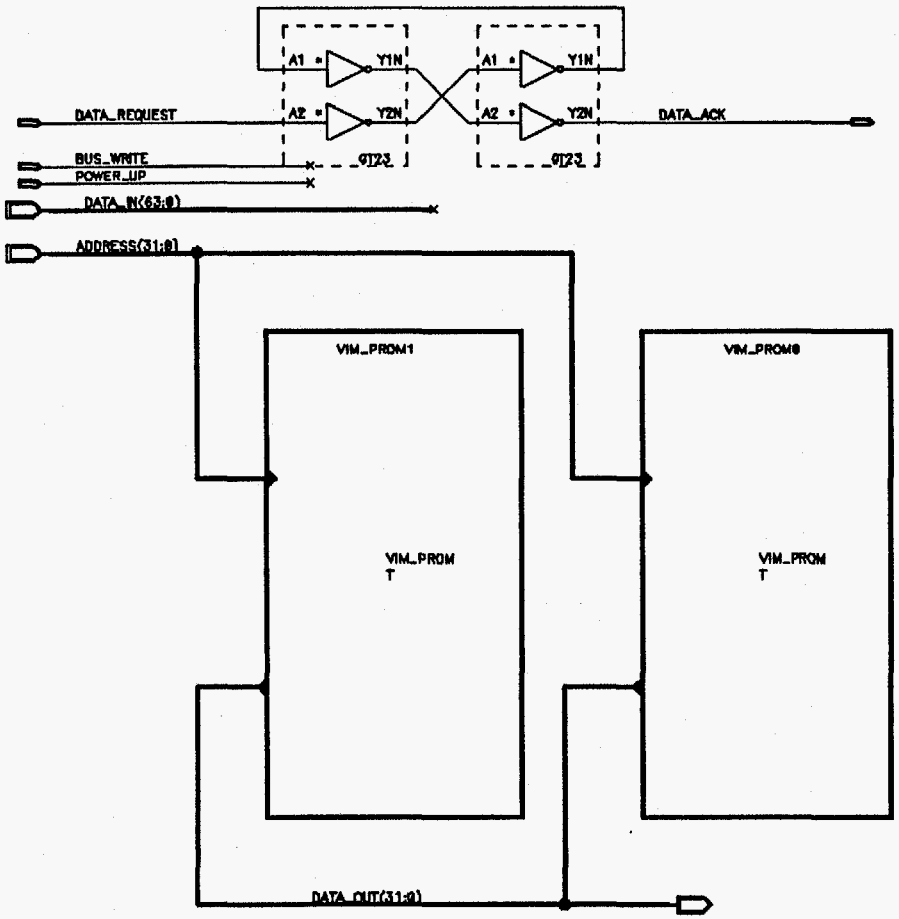
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 10/22/92	PAGE: 2/3	TITLE: FASTBUS MASTER INTERFACE TEMPORARY BIDIR BUFFER 1/2 DATIO(31:16) BUS	FILE NO: NA	REV: A	DATE: 10/22/92



SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3421 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 10/22/02	PAGE: 3/3	TITLE: TEMPORARY BIDI BUFFER CONTROLS	FILE NO: NA	REV: A	DATE: 10/22/02



SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 4/10/91	PAGE: 1/1	TITLE: FASTBUS RISC PROCESSOR MODULE FBUS ABITRATION CONTROL -FOR MODULE SIM			FILE NO: SSI/JV-40128	REV: B	DATE: 4/12/91

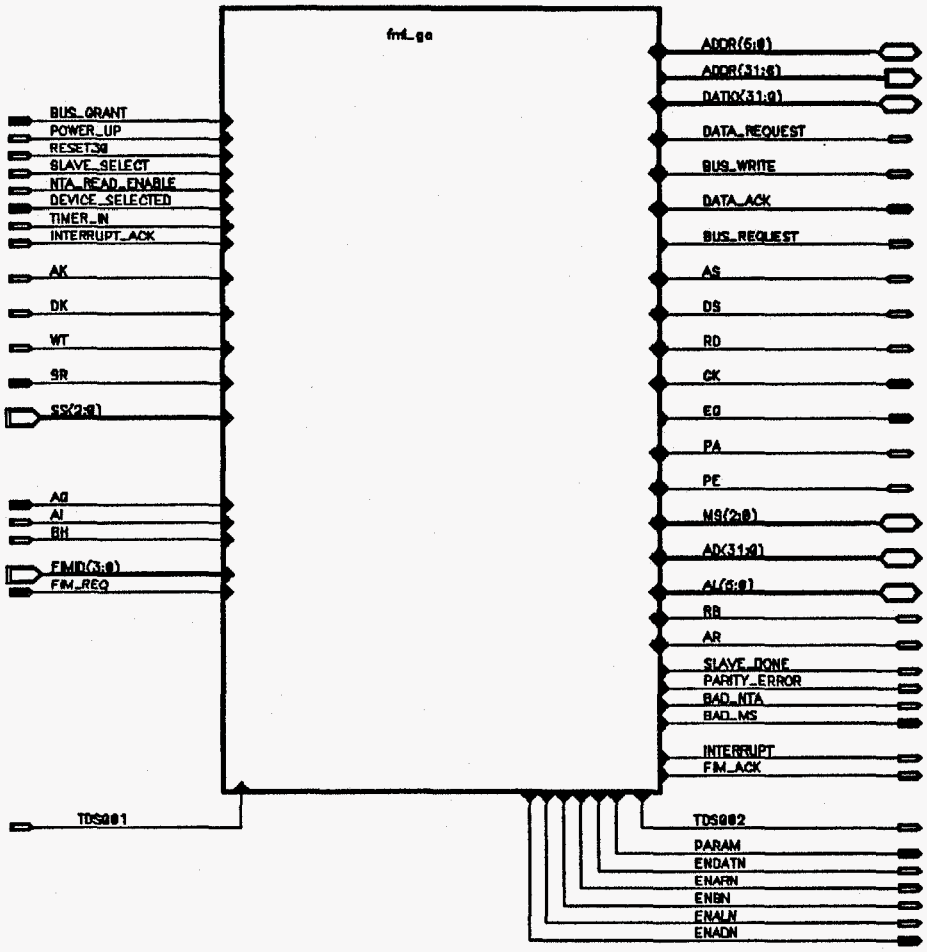


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: RON , DRM	DESIGN DATE: 6/6/98	PAGE: 1/1	TITLE: FASTBUS RISC PROCESSOR MODULE SIMULATION MEMORY	FILE NO: SSI/JV-SIMMEM	REV: B	DATE: 12/21/98

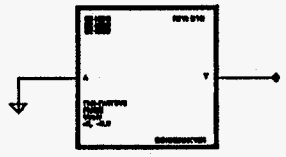
APPENDIX E

FMI SCHEMATICS

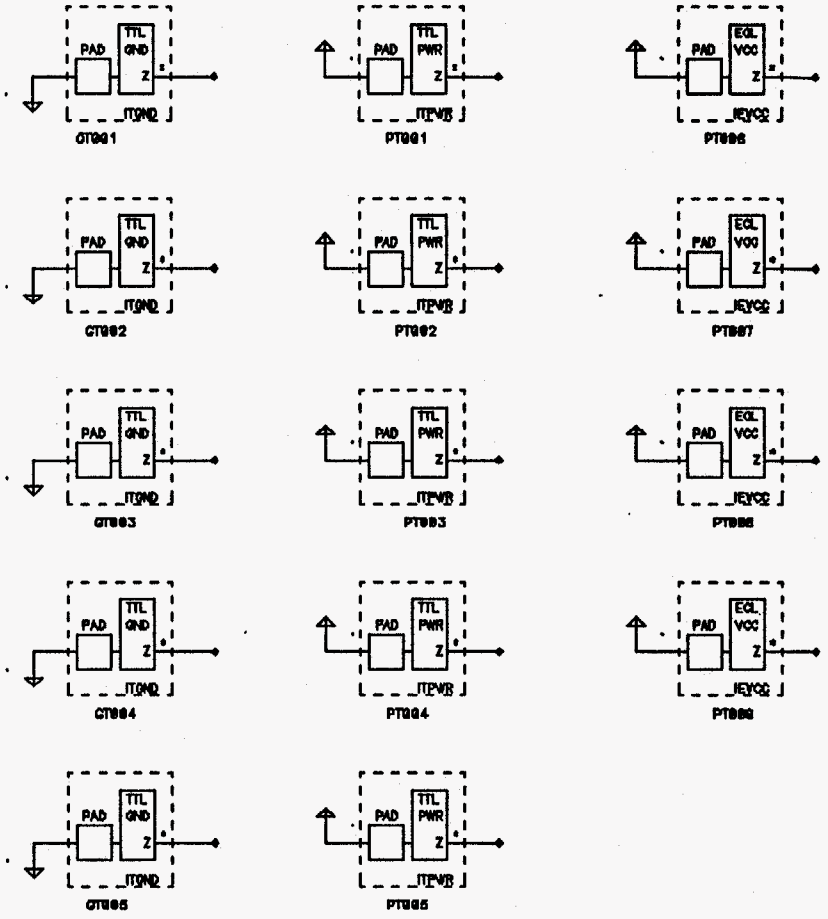
AMCC BICMOS ARRAY

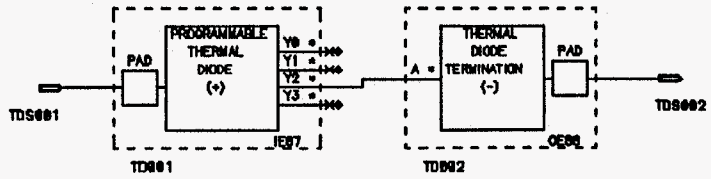


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 12/19/98	PAGE: 1/1	TITLE: FASTBUS MASTER INTERFACE TOP LEVEL BLOCK	FILE NO: SSI/JV-30000	REV: D	DATE: 10/19/92



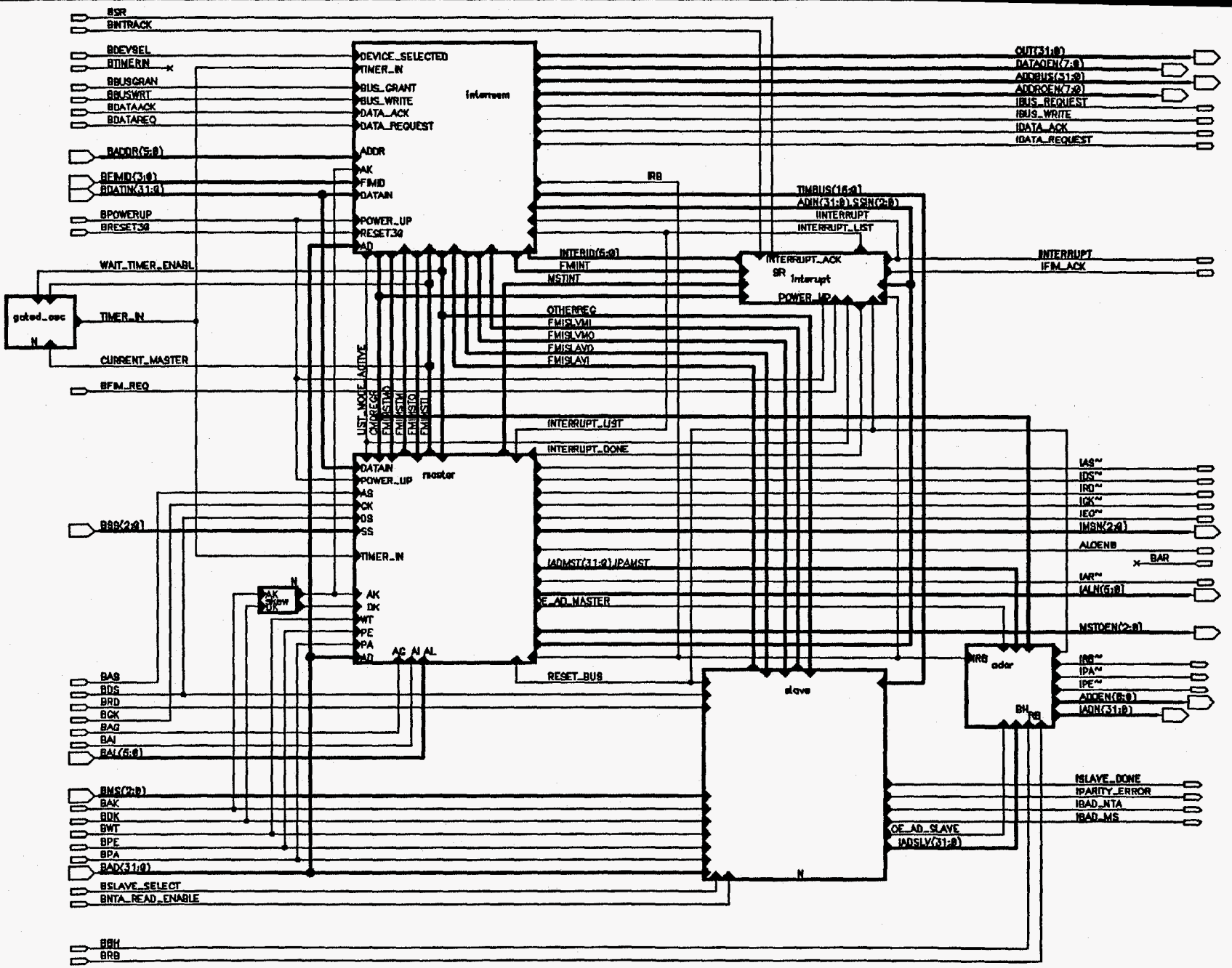
AMCC 024288 CHIP MACRO





SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 12/17/98	PAGE: 1/ 9	TITLE: FASTBUS MASTER INTERFACE AMCC CHIP MACRO & ADDED PWR/GND	FILE NO: SSI/JV-30001	REV: B	DATE: 6/1/91

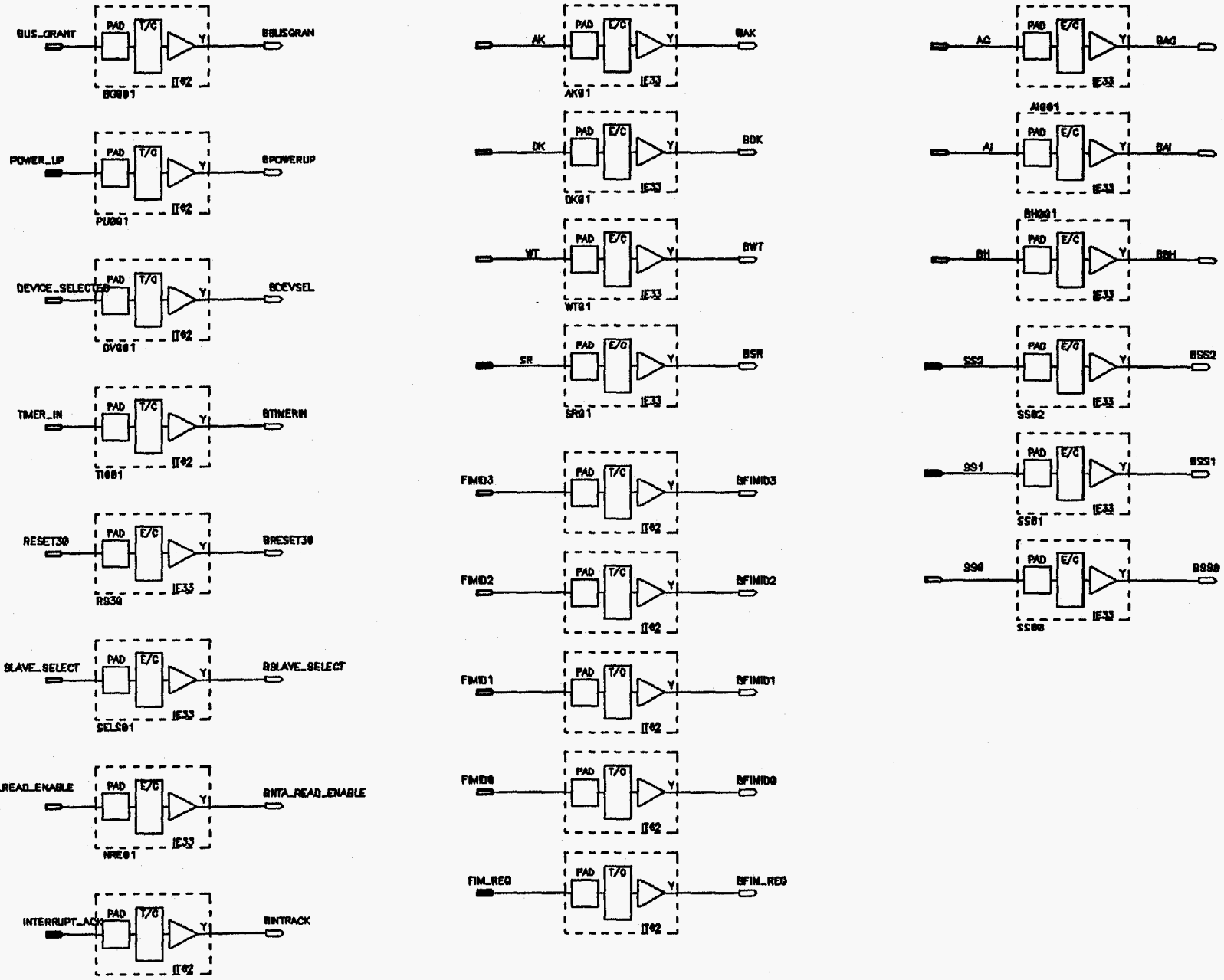
88



SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: DRM R.D. NELSON	DESIGN DATE: 11NOV89	PAGE: 2/9	TITLE: FASTBUS MASTER INTERFACE FMI BLOCKS	FILE NO: SSI/JV-38882	REV: F	DATE: 2/6/91

8 7 6 5 4 3 2 1

68

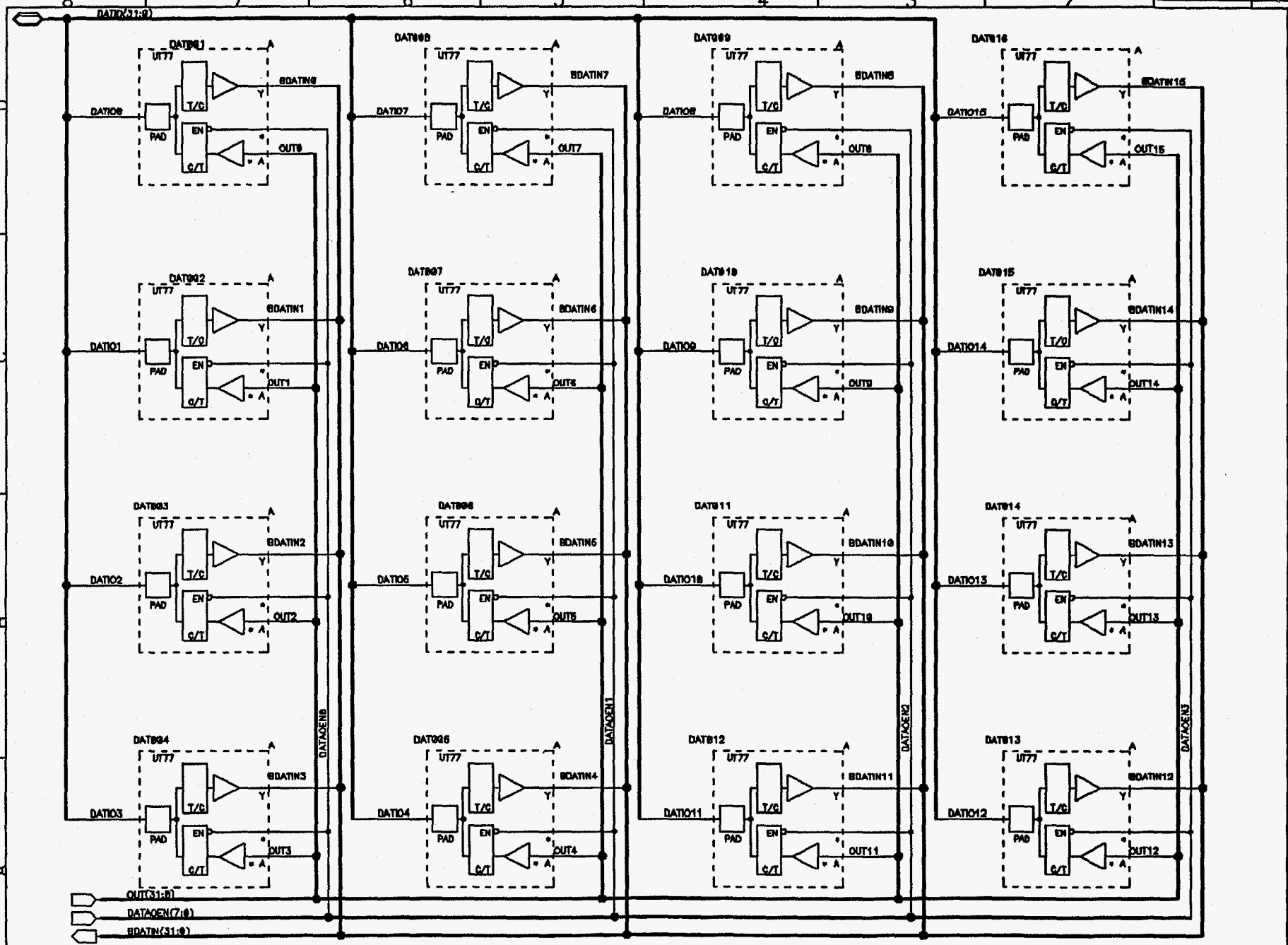


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: RON, DRM	DESIGN DATE: 1/13/90	PAGE: 3/ 9	TITLE: FASTBUS MASTER INTERFACE INPUT PADS/BUFFERS	FILE NO: SSI/JV-30003	REV: E	DATE: 8/26/91

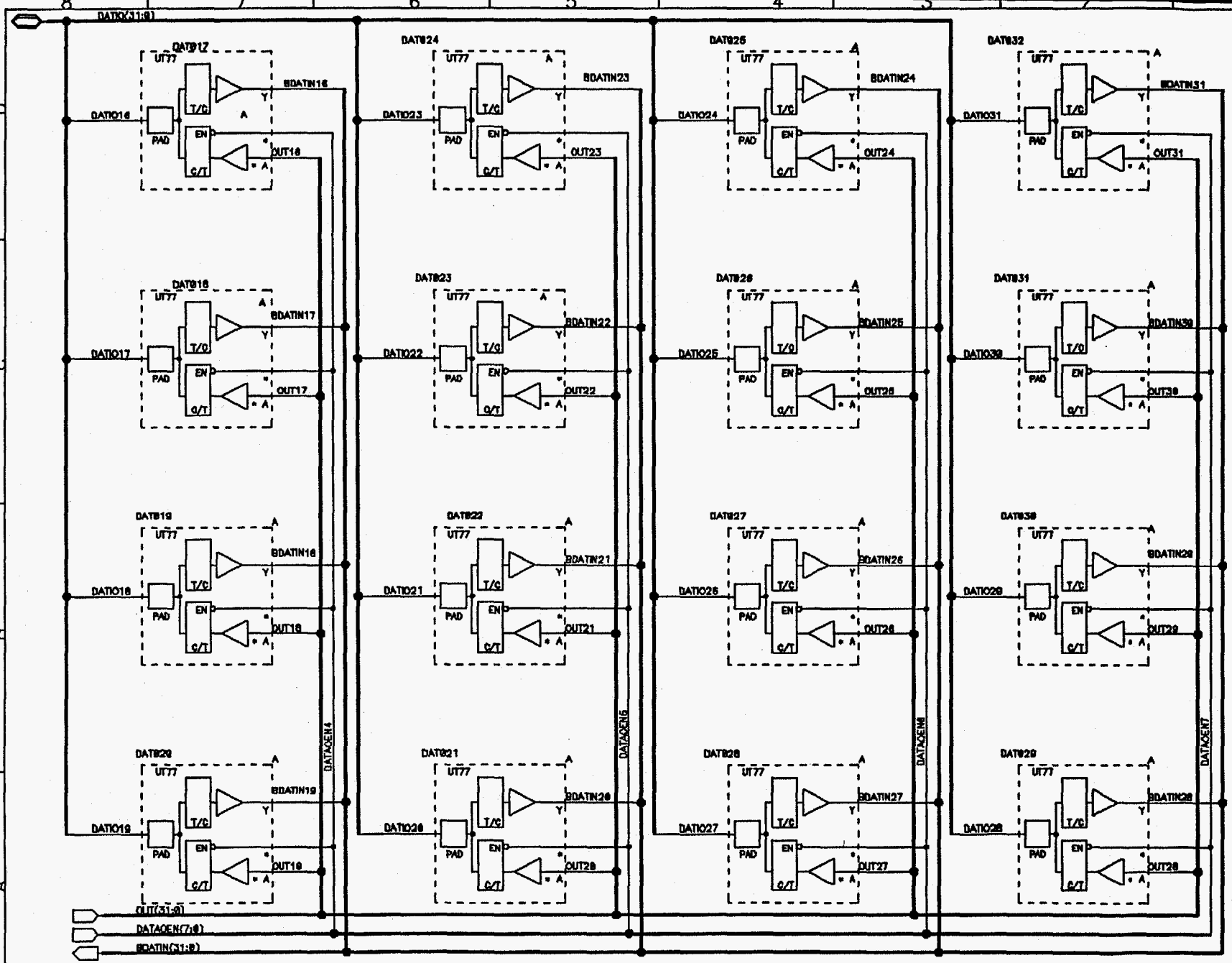
8 7 6 5 4 3 2 1

A B C D

06

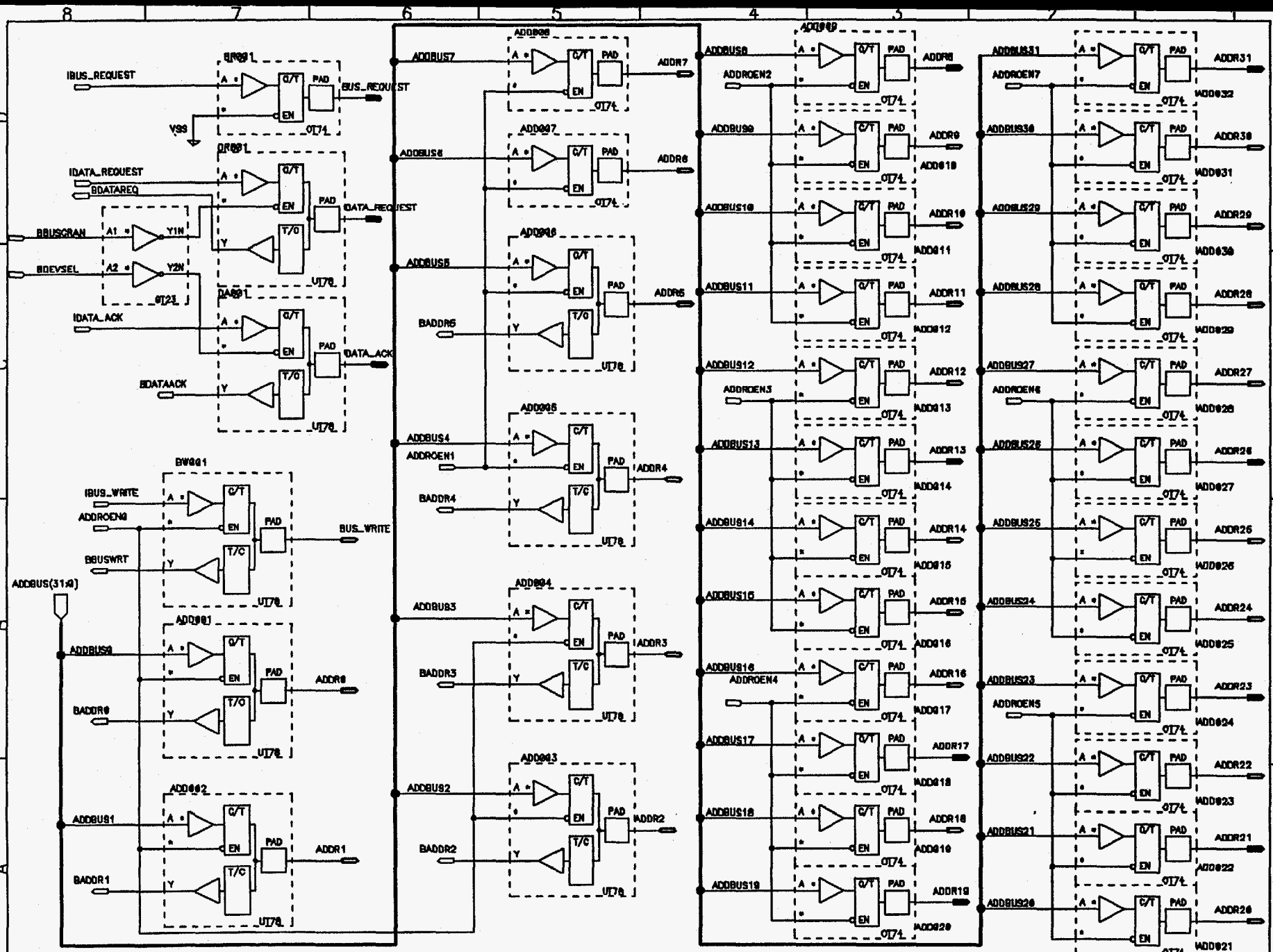


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3481 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: RON D. MACHEN, PE	DESIGN DATE: 10/19/92	PAGE: 6/9	TITLE: FASTBUS MASTER INTERFACE BDI PADS 1/2 DATIO(15:8)	FILE NO: SSI/JV-38666	REV: A	DATE: 10/19/92

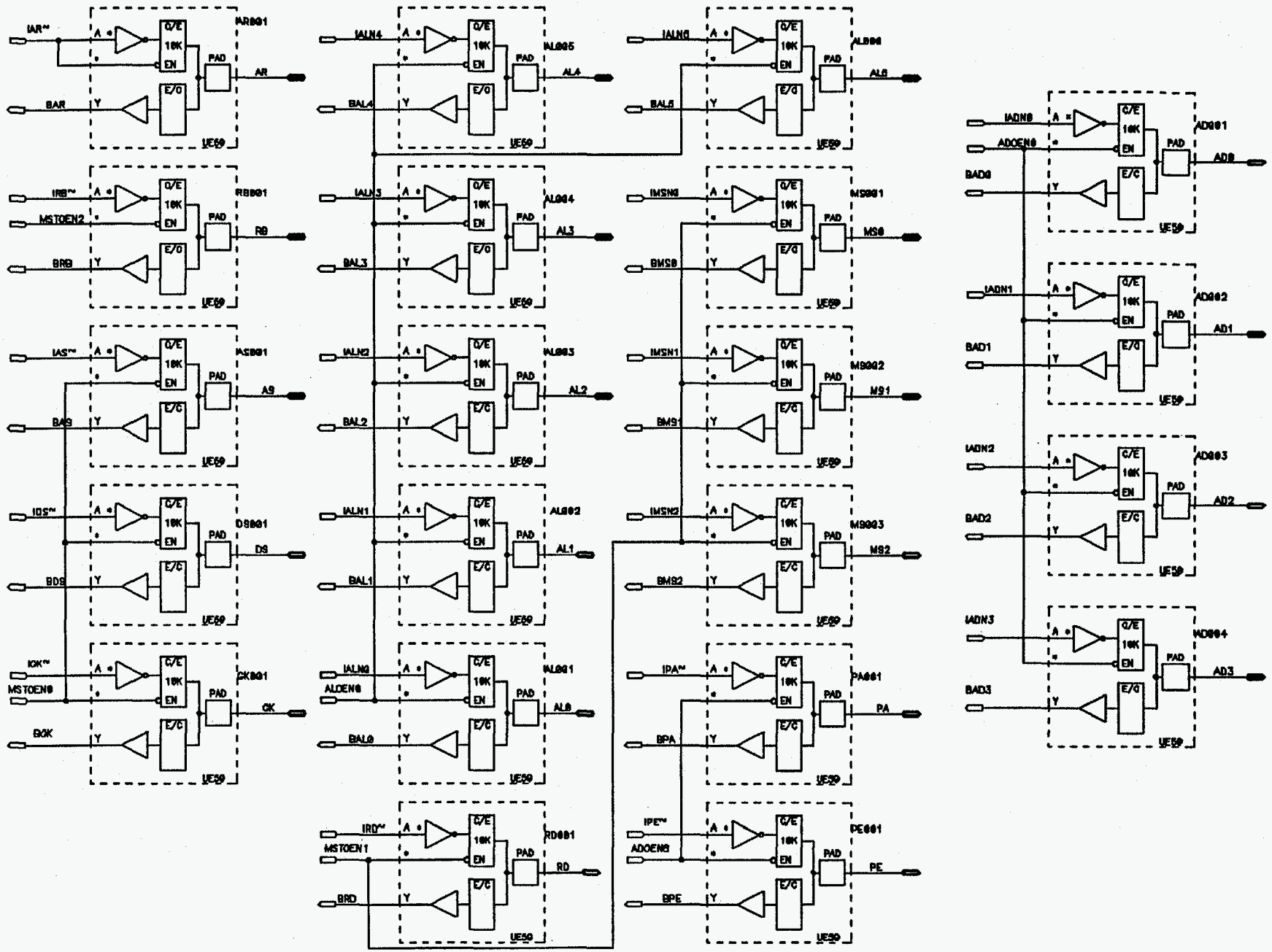


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 10/19/92	PAGE: 6/9	TITLE: FASTBUS MASTER INTERFACE 80I PADS 1/2 DATIO(31:16) BUS	FILE NO: SSI/JV-30006	REV: A	DATE: 10/19/92

92

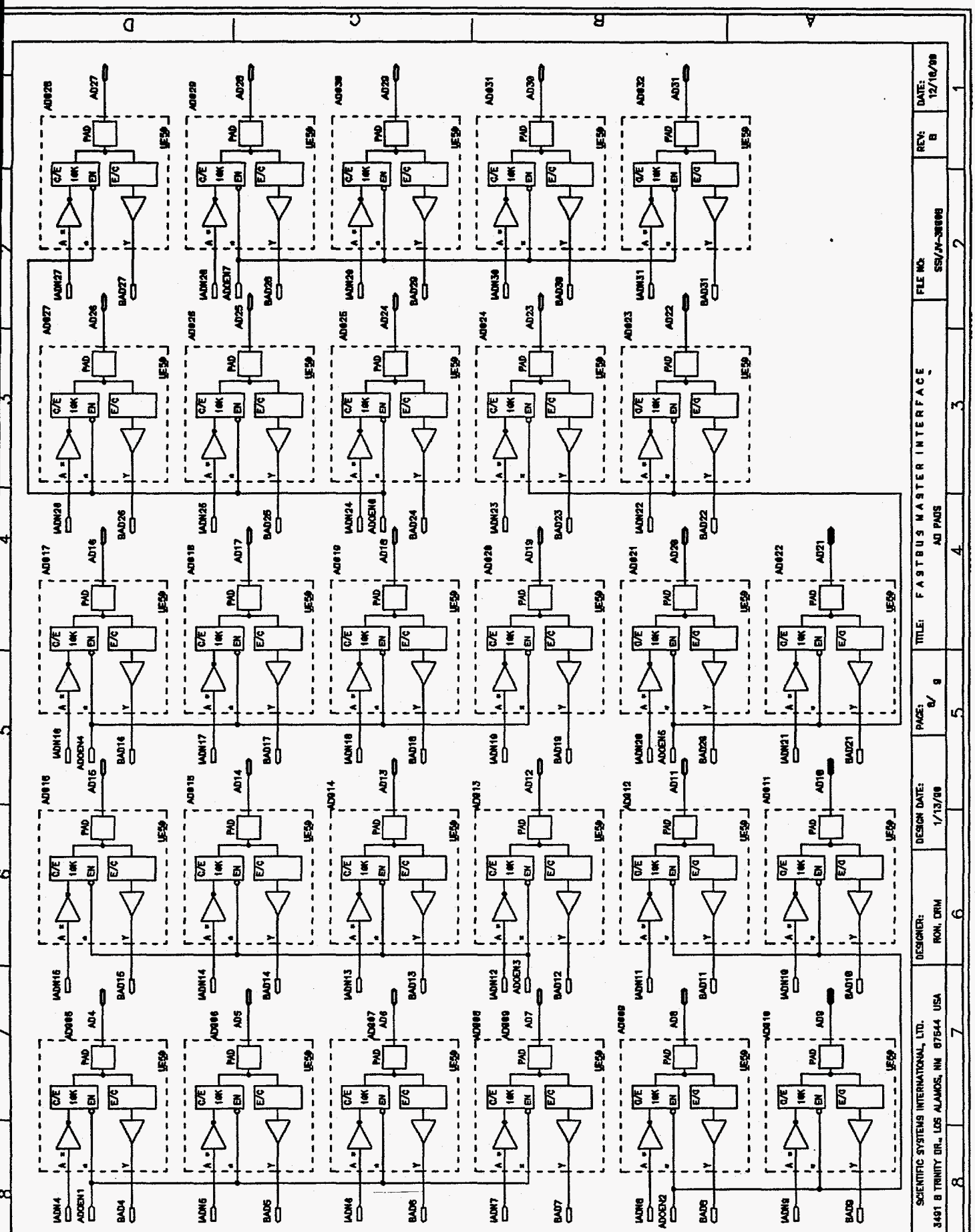


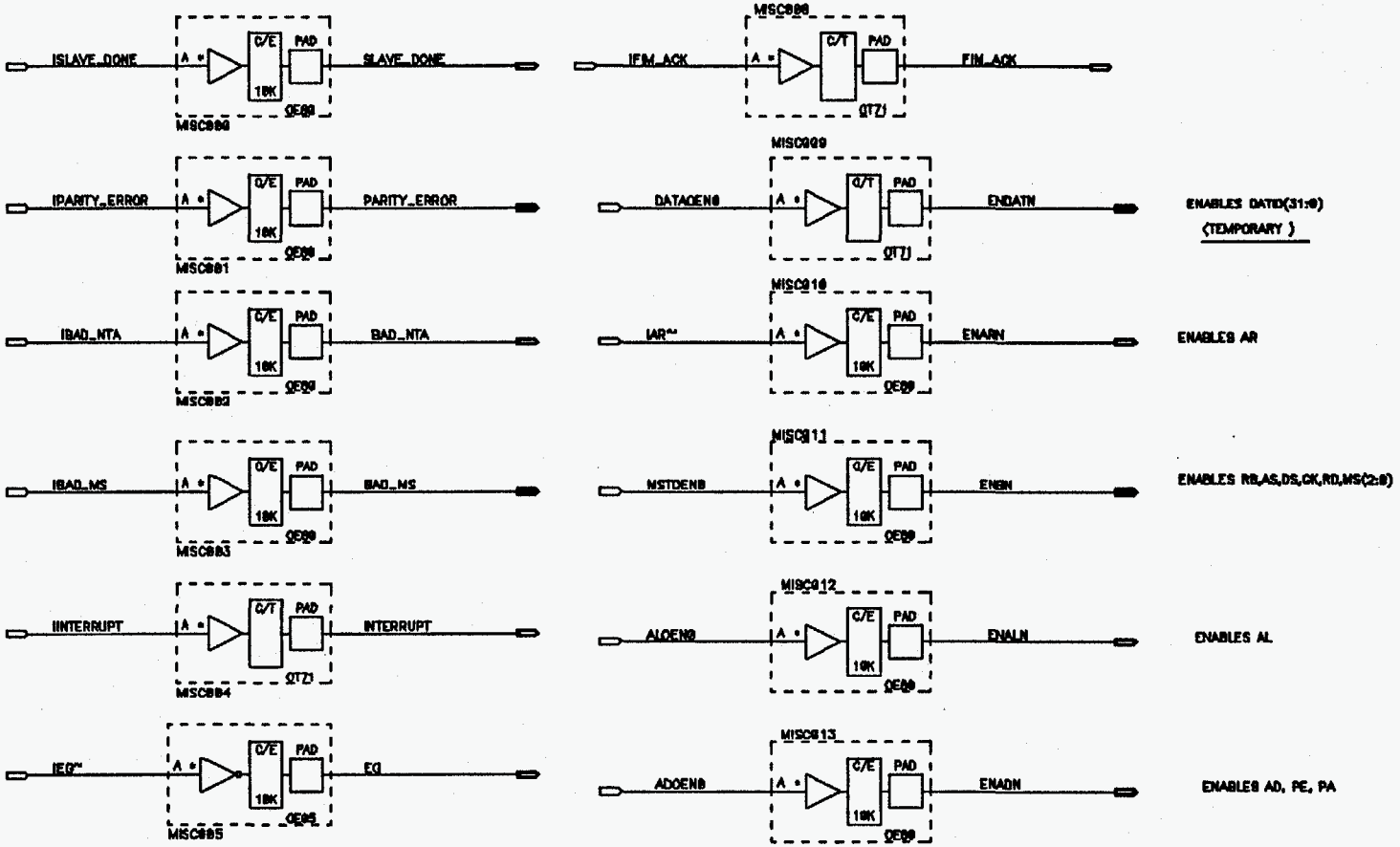
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: RON, DRM	DESIGN DATE: 1/13/98	PAGE: 5/9	TITLE: FASTBUS MASTER INTERFACE OUTPUT PADS-TTL	FILE NO: SSI/JV-30006	REV: C	DATE: 2/6/91



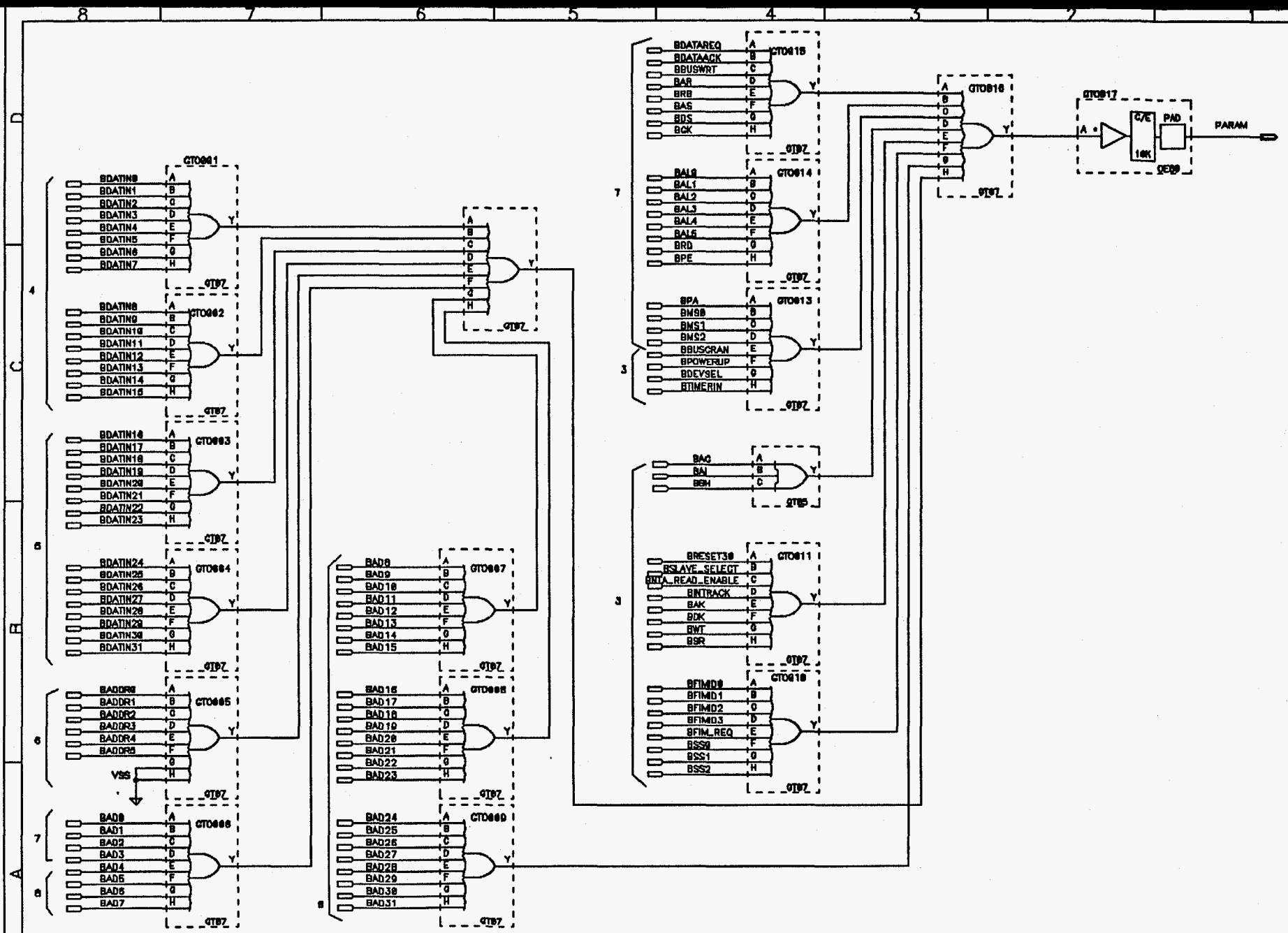
(TWO REMAINING TTL OUTPUT PADS)

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: RON, DRM	DESIGN DATE: 1/13/99	PAGE: 7/9	TITLE: FASTBUS MASTER INTERFACE OUTPUT PADS - ECL	FILE NO: SSI/JV-30007	REV: B	DATE: 12/16/99

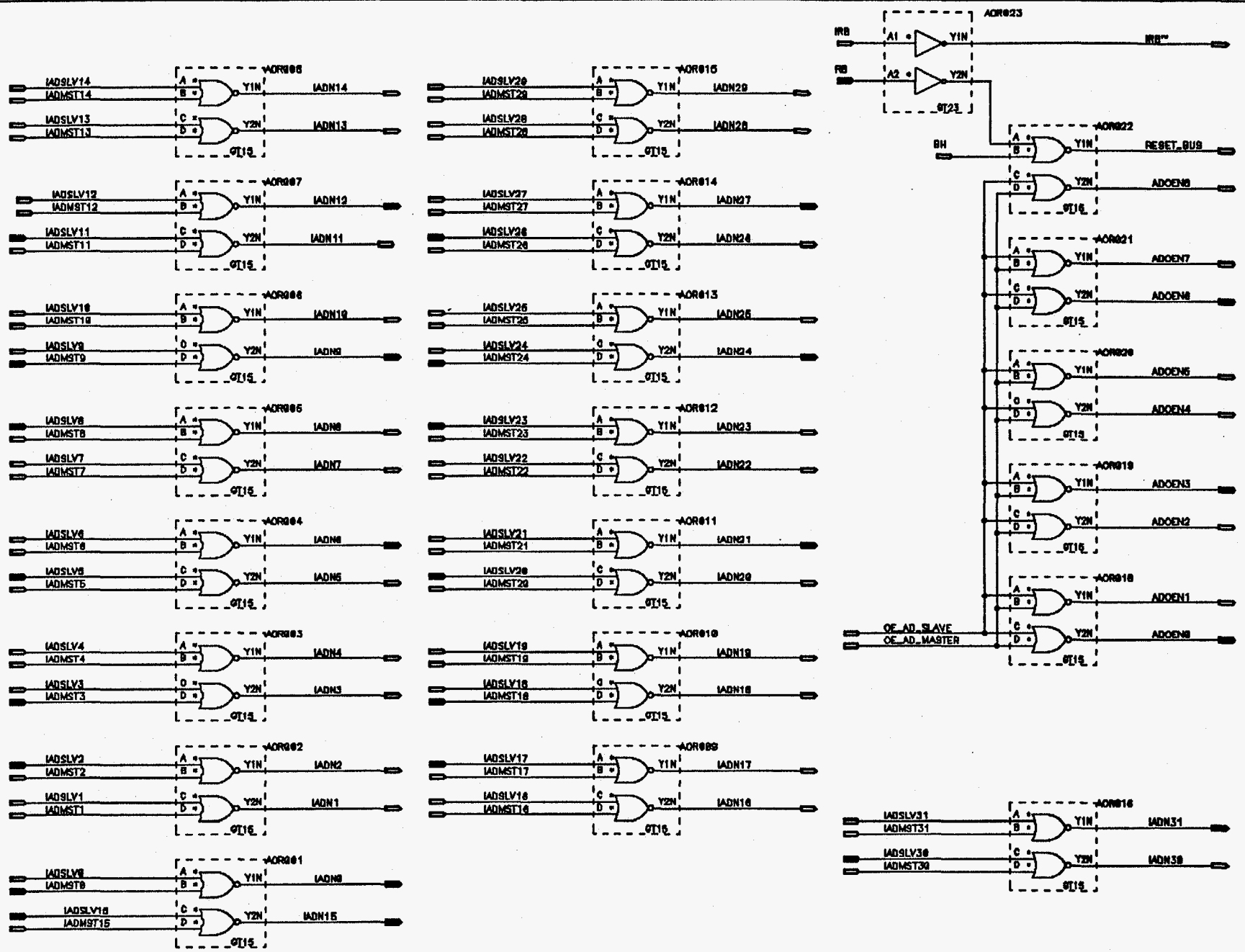

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3481 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: RON. DRM
DESIGN DATE: 1/13/98
PAGE: 8/ 8
TITLE: FASTBUS MASTER INTERFACE
AD PADS
FILE NO: SSI/M-38888
REV: B
DATE: 12/16/98
PAD
C/E
E/G
10K
EN
UE59



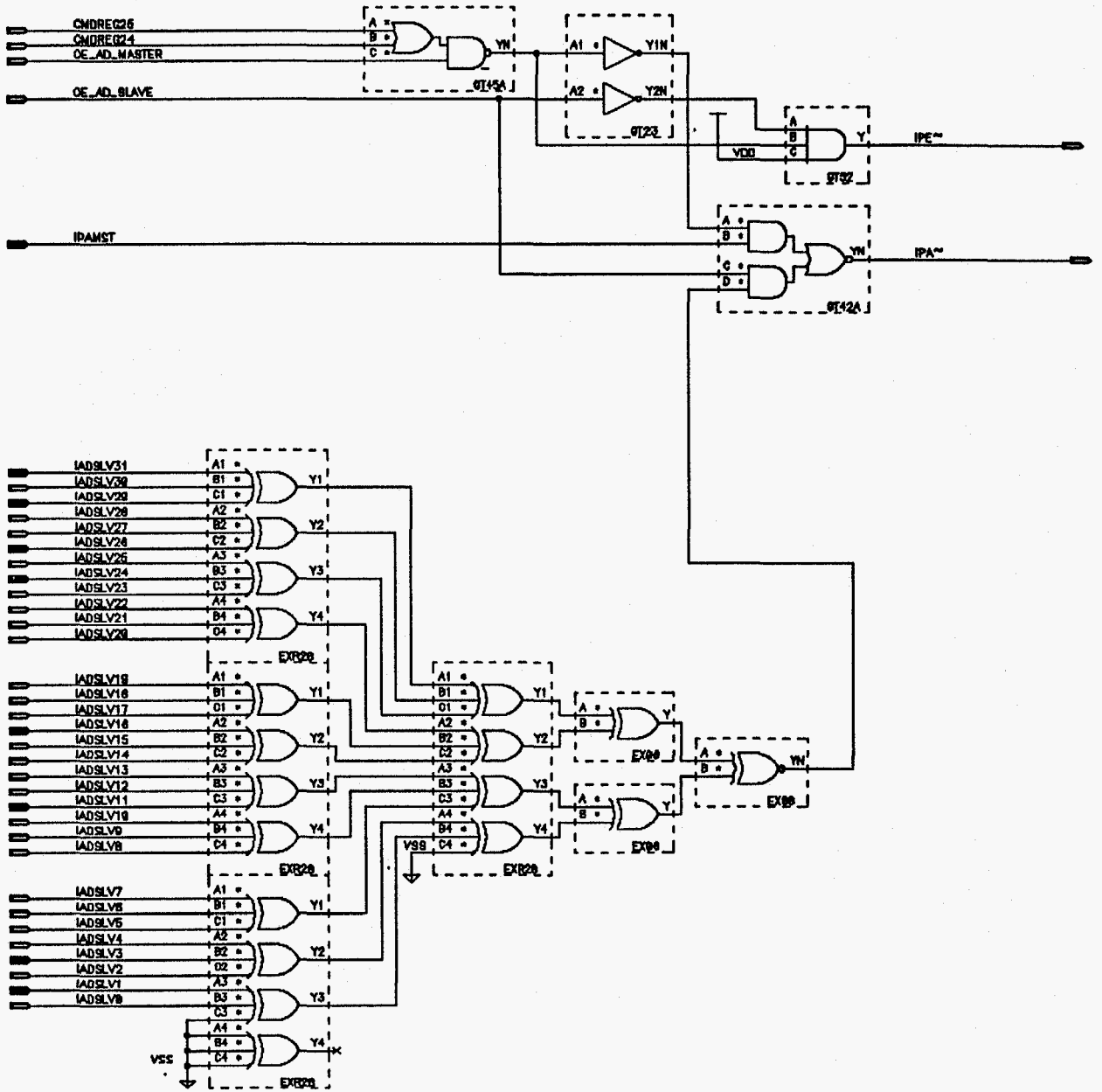
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: RON,DRM	DESIGN DATE: 8/1/91	PAGE: 9/9	TITLE: FASTBUS MASTER INTERFACE MISC OUTPUT PADS, ECL AND TTL	FILE NO: SSI/JV-30009	REV: C	DATE: 10/22/92



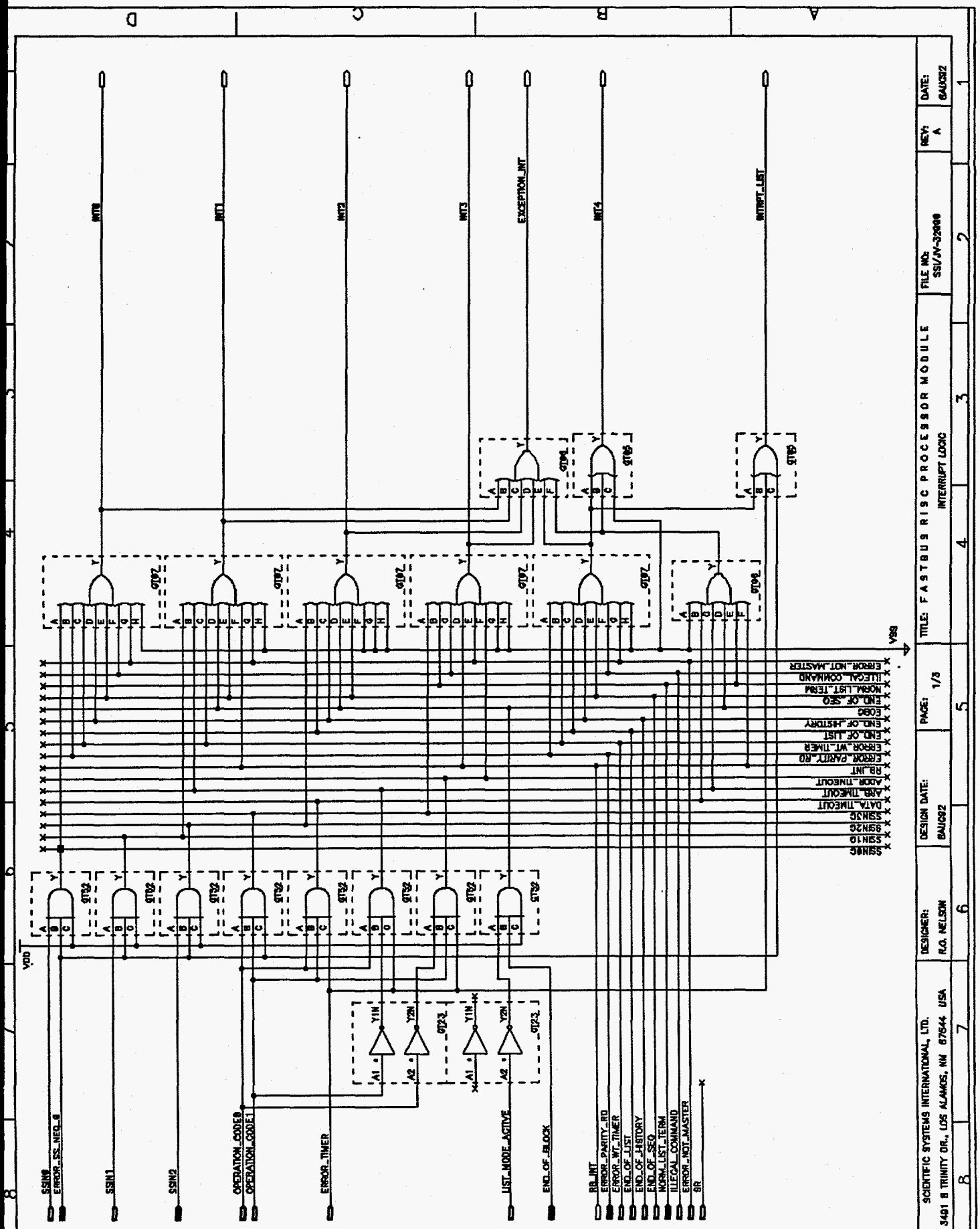
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: D. MACHEN, PE	DESIGN DATE: 10/19/92	PAGE: 10/10	TITLE: FASTBUS MASTER INTERFACE AMCC PARAMETRIC GATE TREE	FILE NO: SSI/JV-30010	REV: B	DATE: 10/26/92



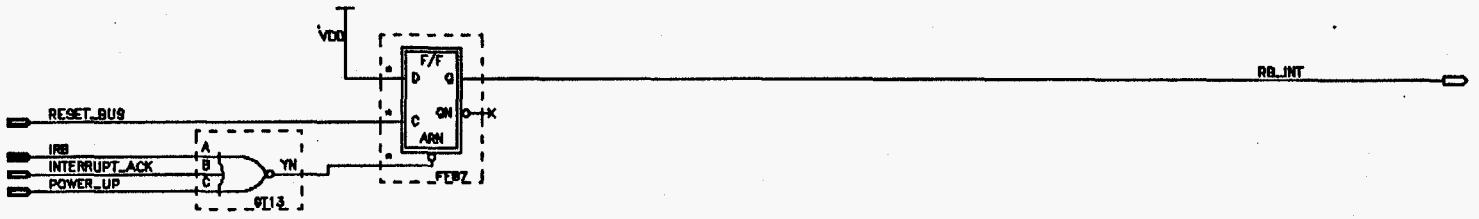
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3421 E TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: DRM R.O. NLESON	DESIGN DATE: 6MAY98	PAGE: 1/2	TITLE: FASTBUS MASTER INTERFACE MASTER/SLAVE OUTPUT AD BUS MERGE	FILE NO: SSI/JV-31600	REV: B	DATE: 12/28/99

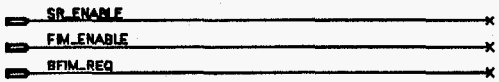


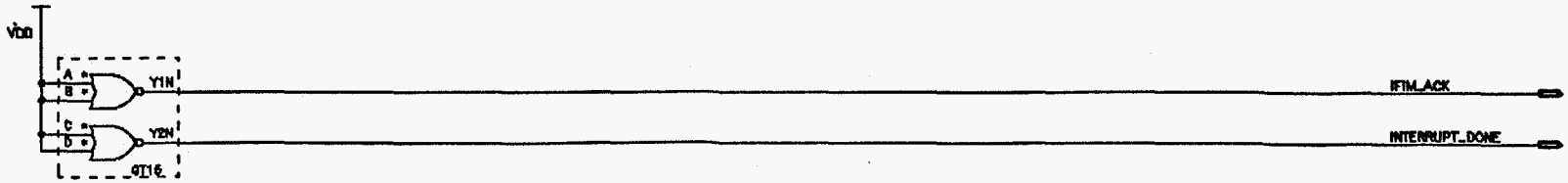
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 17OCT92	PAGE: 2/2	TITLE: FASTBUS RISC PROCESSOR MODULE PARITY GENERATION LOGIC	FILE NO: SSI/JV-31681	REV: A	DATE: 17OCT92	
8	7	6	5	4	3	2	1	


INT0
INT1
INT2
INT3
EXCEPTION_INT
INT4
INTRPT_LIST
SSIN0
SSIN1
SSIN2
SSIN3
SSIN4
SSIN5
SSIN6
SSIN7
DATA_TIMEOUT
ADR_TIMEOUT
ADDR_TIMEOUT
RB_INT
ERROR_PARITY_RD
ERROR_WT_TIMER
END_OF_LIST
END_OF_HISTORY
END_OF_SEQ
NORM_LIST_TERM
ILLEGAL_COMMAND
ERROR_NOT_MASTER
ERROR_NOT_MASTER
ILLEGAL_COMMAND
NORM_LIST_TERM
EOBC
END_OF_SEQ
END_OF_LIST
END_OF_HISTORY
ERROR_WT_TIMER
ERROR_PARITY_RD
RB_INT
ADDR_TIMEOUT
ADR_TIMEOUT
DATA_TIMEOUT
SSIN5
SSIN4
SSIN3
SSIN2
SSIN1
SSIN0
OPERATION_CODE0
OPERATION_CODE1
ERROR_TIMER
LIST_MODE_ACTIVE
END_OF_BLOCK
RB_INT
ERROR_PARITY_RD
ERROR_WT_TIMER
END_OF_LIST
END_OF_HISTORY
END_OF_SEQ
NORM_LIST_TERM
ILLEGAL_COMMAND
ERROR_NOT_MASTER
BR
VDD
VSS
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 8AUG92
PAGE: 1/3
TITLE: FASTBUS RISC PROCESSOR MODULE
INTERRUPT LOGIC
FILE NO: SSI/HV-32000
REV: A
DATE: 8AUG92

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3481 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: BAUG92	PAGE: 2/3	TITLE: FASTBUS RISC PROCESSOR MODULE INTERRUPT LOGIC	FILE NO: SSI/JV-32001	REV: A	DATE: BAUG92

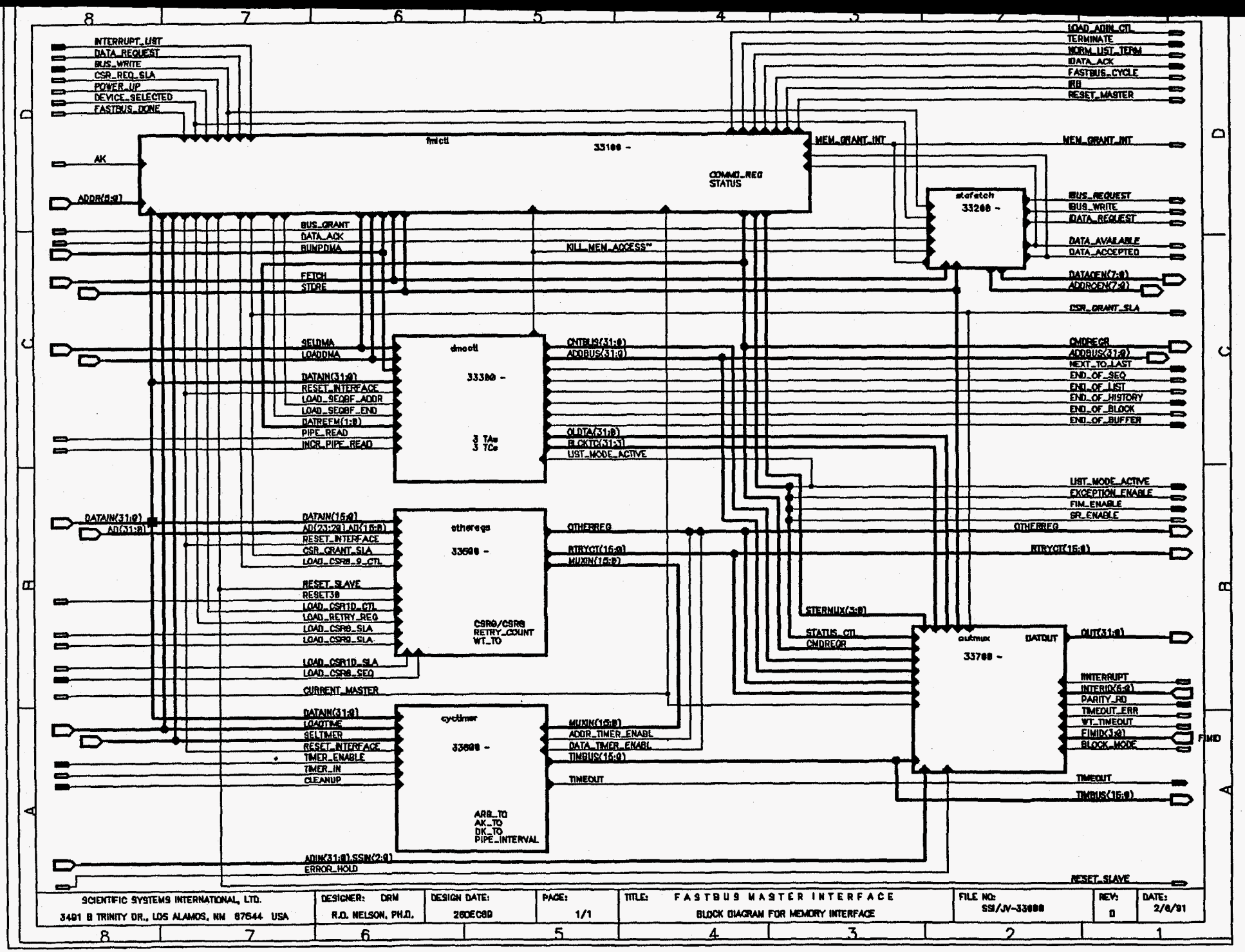




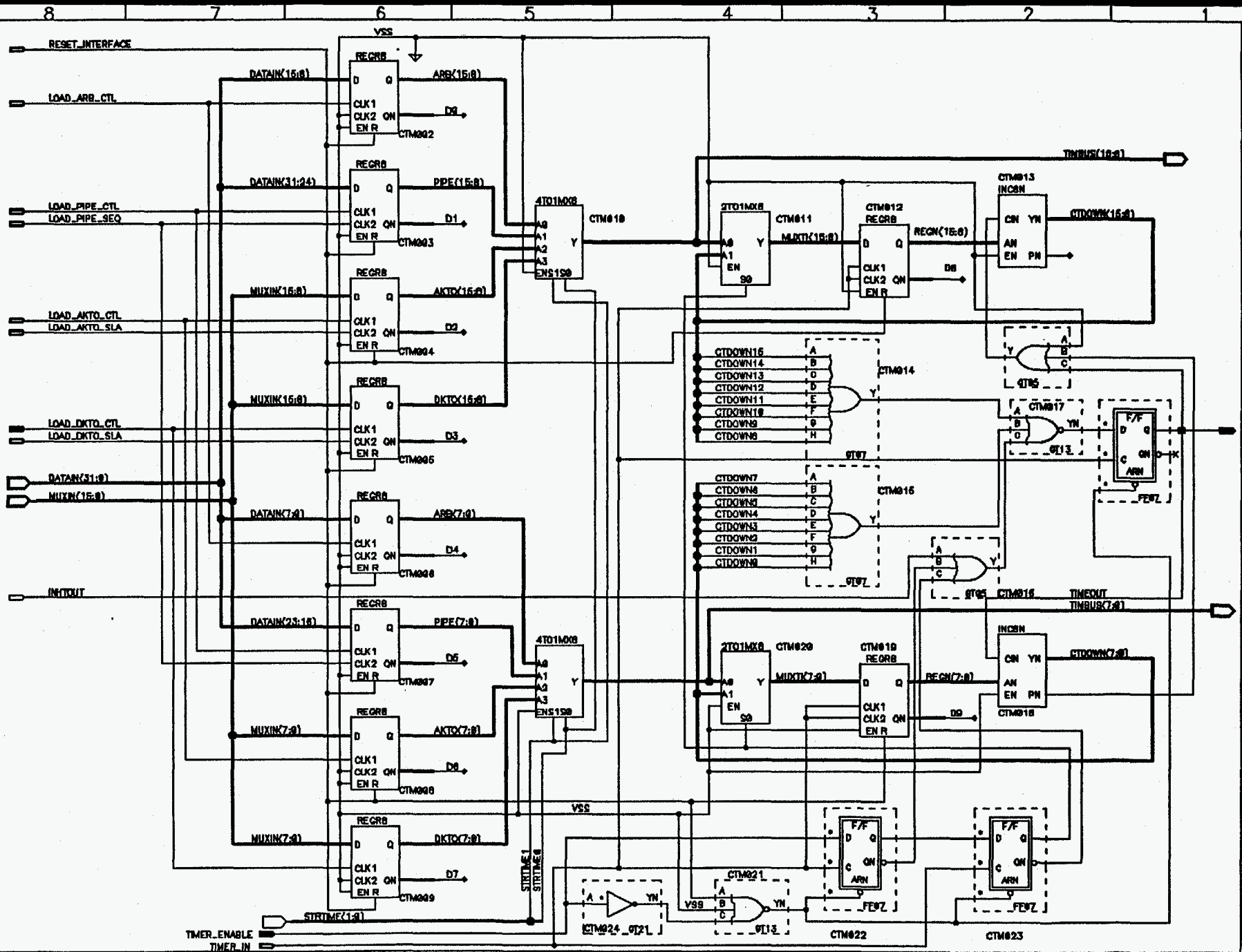


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 8AUG92	PAGE: 3/3	TITLE: FASTBUS RISC PROCESSOR MODULE FASTBUS INTERRUPT MESSAGE (FIM) LOGIC	FILE NO: SSI/JV-32002	REV: A	DATE: 8AUG92

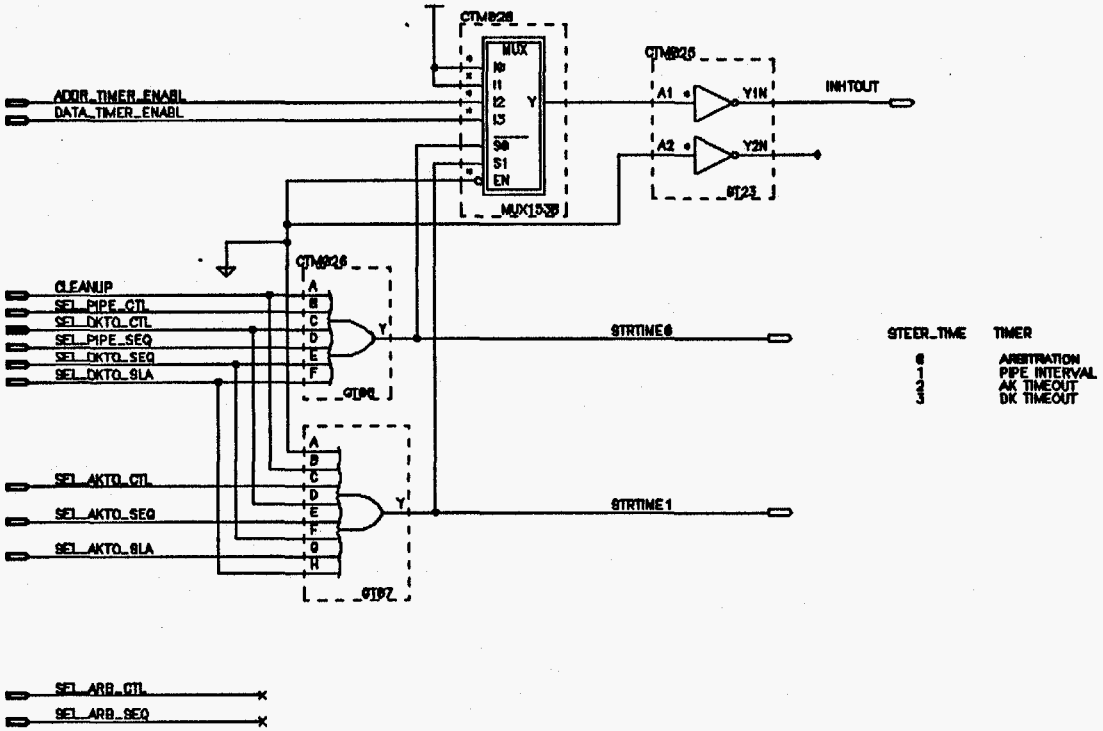
8 7 6 5 4 3 2 1



SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: DRM R.D. NELSON, PH.D.	DESIGN DATE: 26DEC89	PAGE: 1/1	TITLE: FASTBUS MASTER INTERFACE BLOCK DIAGRAM FOR MEMORY INTERFACE	FILE NO: SSI/JV-33000	REV: 0	DATE: 2/6/91

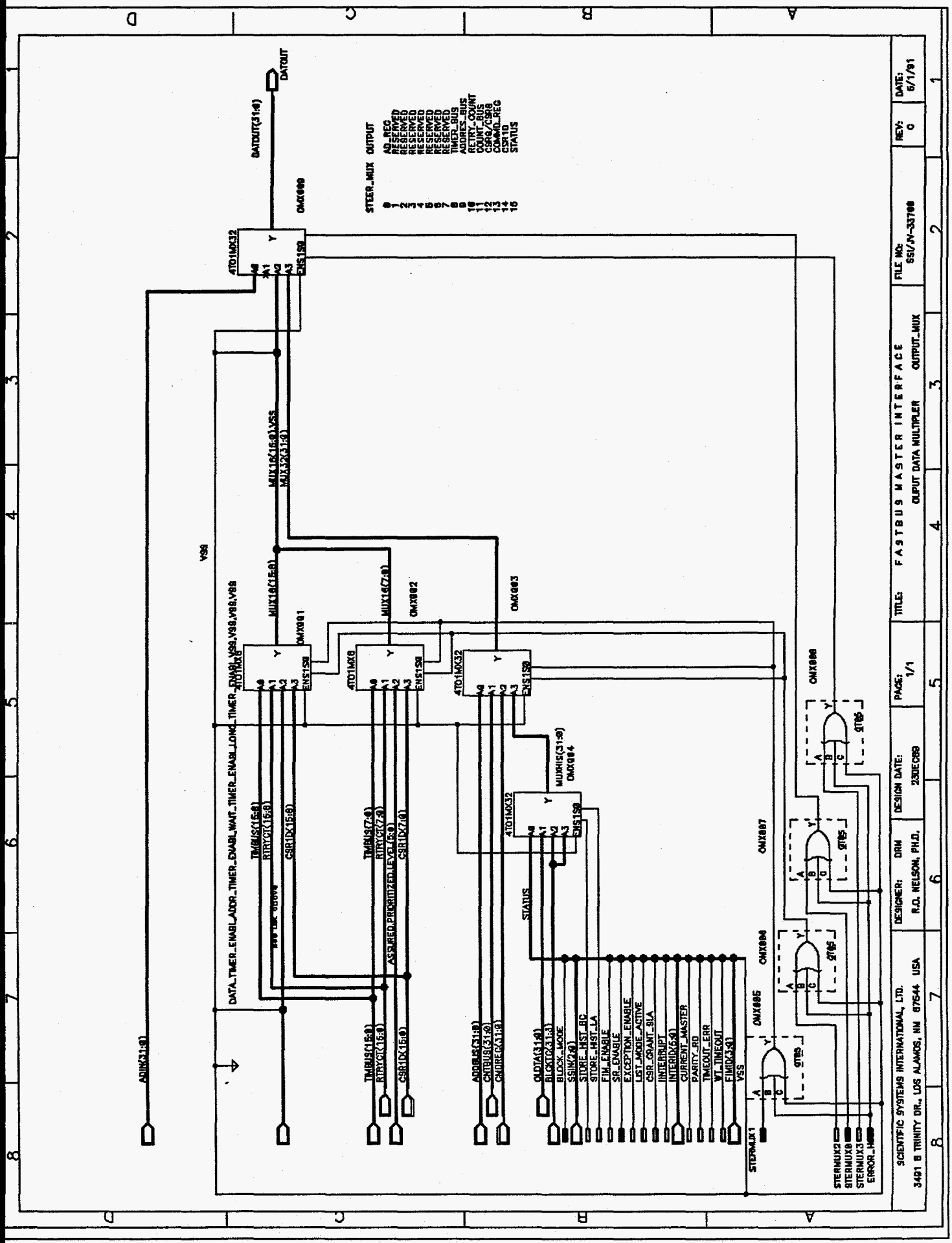


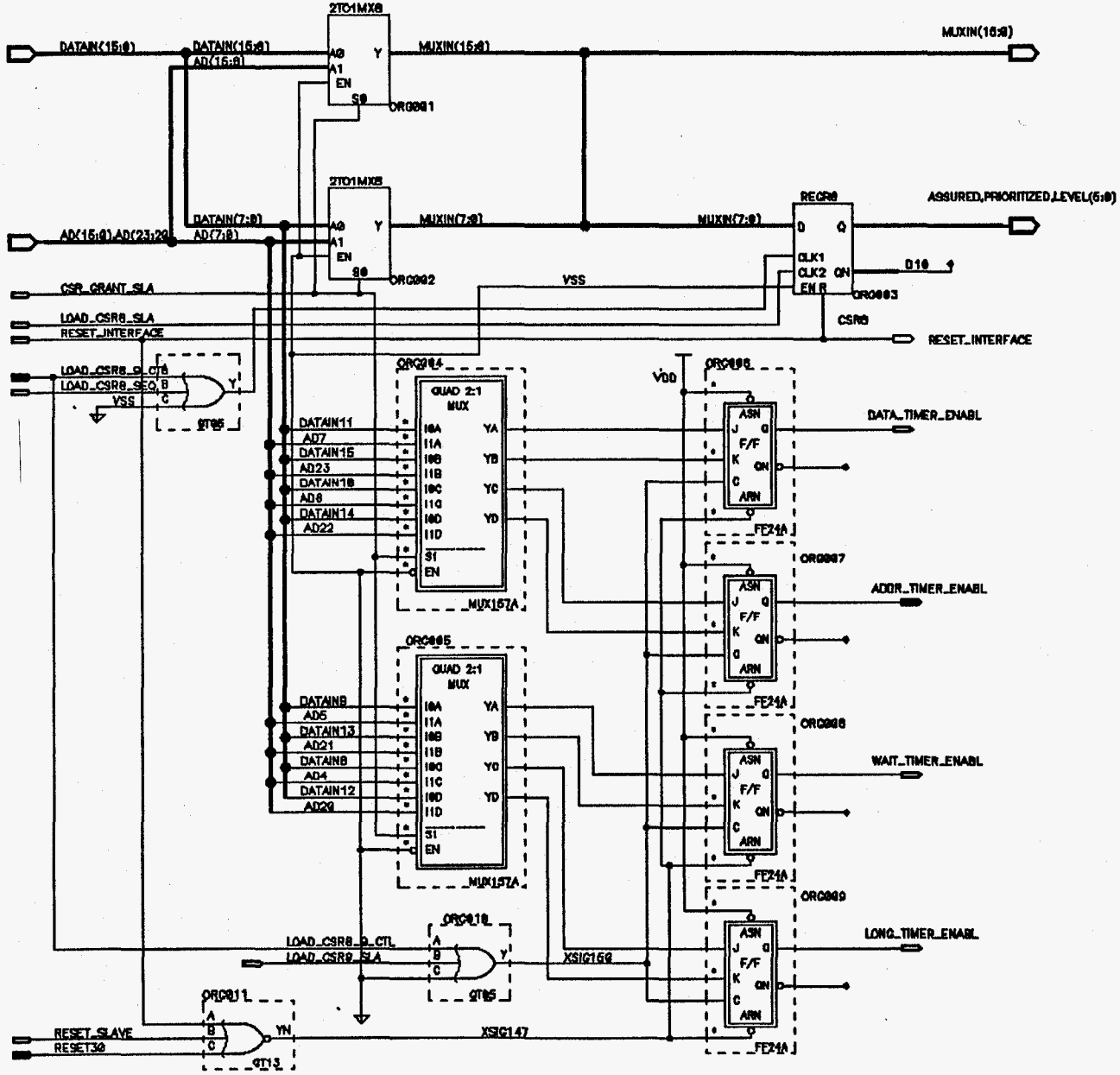
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3481 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: DRM R.O. NELSON, PH.D.	DESIGN DATE: 12DEC89	PAGE: 1/2	TITLE: FASTBUS MASTER INTERFACE TIMER REGISTERS & MUX CYCLE TIMERS	FILE NO: SSI/JV-33000	REV: D	DATE: 6/1/91



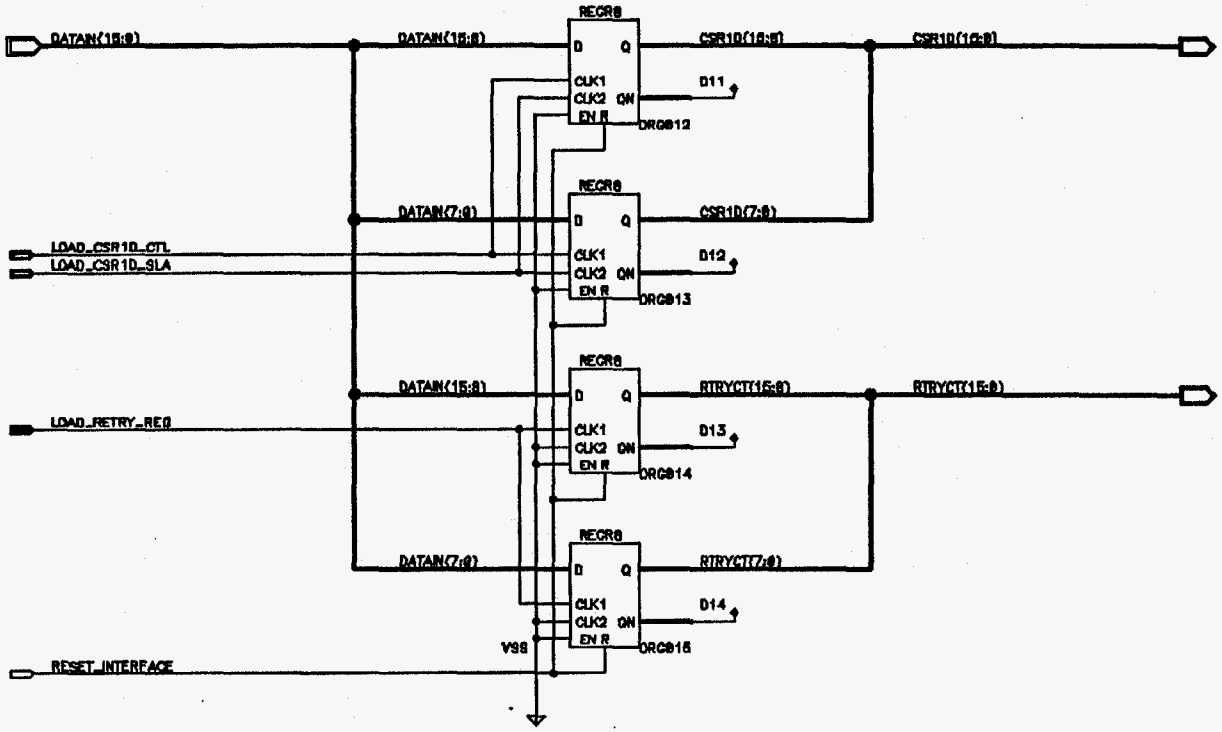
STEER_TIME	TIMER
0	ARBITRATION
1	PIPE INTERVAL
2	AK TIMEOUT
3	DK TIMEOUT

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: DRM RON	DESIGN DATE: 12/27/89	PAGE: 2/ 2	TITLE: FASTBUS MASTER INTERFACE TIMER CONTROLS	FILE NO: SSI/JV-33601	REV: B	DATE: 12/31/98

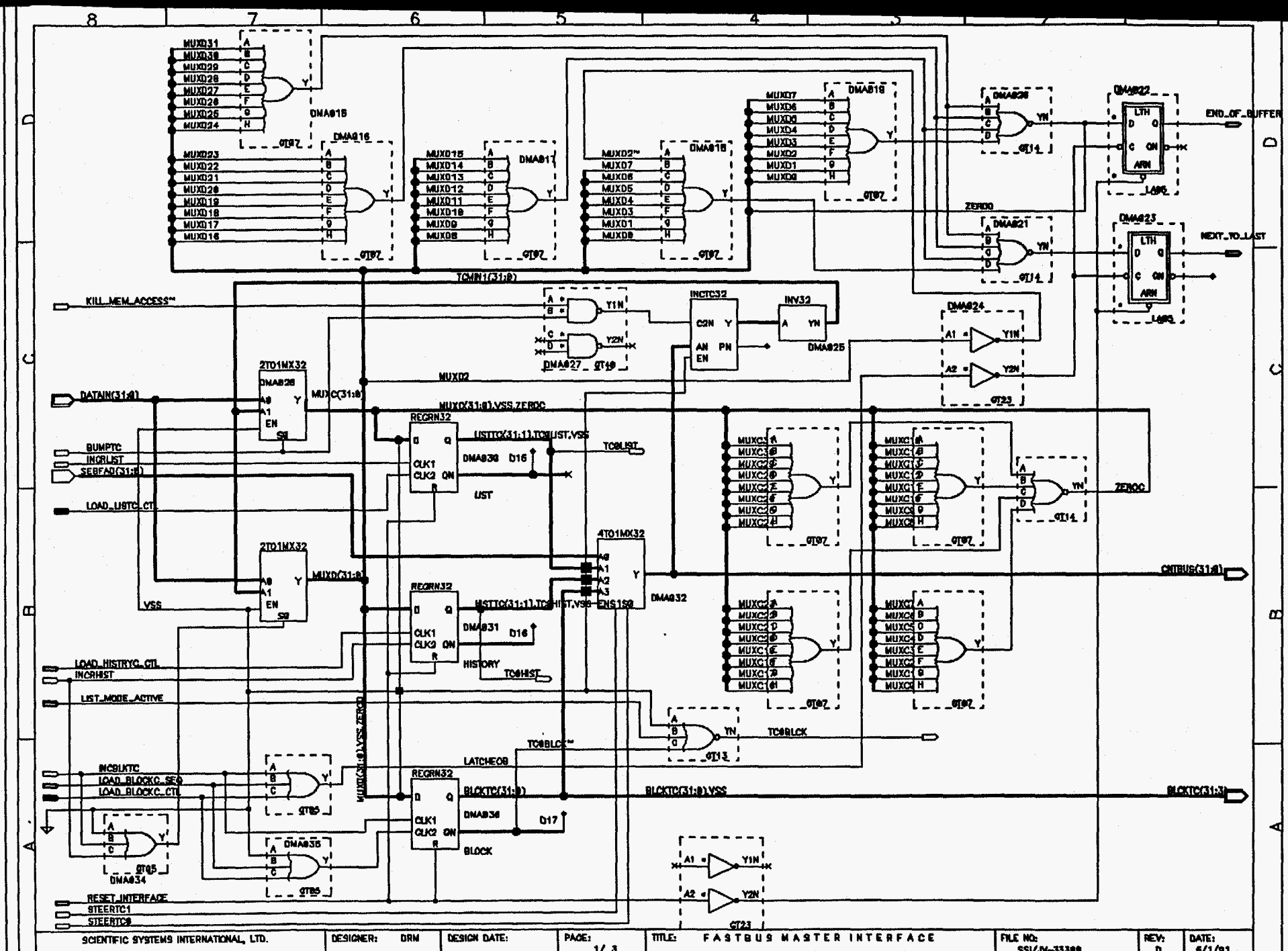

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: DRM
R.O. NELSON, PH.D.
DESIGN DATE:
23DEC89
PAGE: 1/1
TITLE: FASTBUS MASTER INTERFACE
OUTPUT DATA MULTIPLEX OUTPUT_MUX
FILE NO: SSI/JV-33708
REV: 0
DATE: 6/1/91
DATOUT
DATOUT(31:0)
OMX000
4TO1MX32
STEER_MUX OUTPUT
0 AD_REG
1 RESERVED
2 RESERVED
3 RESERVED
4 RESERVED
5 RESERVED
6 RESERVED
7 RESERVED
8 TMBUS_BUS
9 ADDRES_BUS
10 RETRY_COUNT
11 COUNT_BUS
12 CSR9/CSR8
13 COMMD_REG
14 CSR10
15 STATUS
VSS
MUX16(15:0),VSS
MUX32(31:0)
OMX001
MUX16(15:0)
MUX16(7:0)
OMX002
OMX003
DATA_TIMER_ENABL_ADDR_TIMER_ENABL_WAIT_TIMER_ENABL_LONG_TIMER_ENABL,V99,V98,V98,V98
TMBUS(15:0)
RTRYCT(15:0)
CSR1DX(15:0)
TMBUS(7:0)
RTRYCT(7:0)
ASSURED_PRIORITY_DELAY(7:0)
CSR1DX(7:0)
ADDBUS(31:0)
CNTBUS(31:0)
CNTADRE(A1:0)
QUOTA(31:0)
BLKCTR(31:0)
MUXHIS(31:0)
OMX004
STATUS
BLOCK_MODE
SSIN(2:0)
STORE_HST_ABC
STORE_HST_LA
FIM_ENABLE
SR_ENABLE
EXCEPTION_ENABLE
LIST_MODE_ACTIVE
CSR_GRANT_SLA
INTERRUPT
INTERID(6:0)
CURRENT_MASTER
PARITY_RD
TMEOUT_ERR
WT_TIMEOUT
FIMDX(3:0)
VSS
OMX005
OMX006
OMX007
OMX008
STBUF
STERMUX1
STERMUX2
STERMUX3
ERROR_H

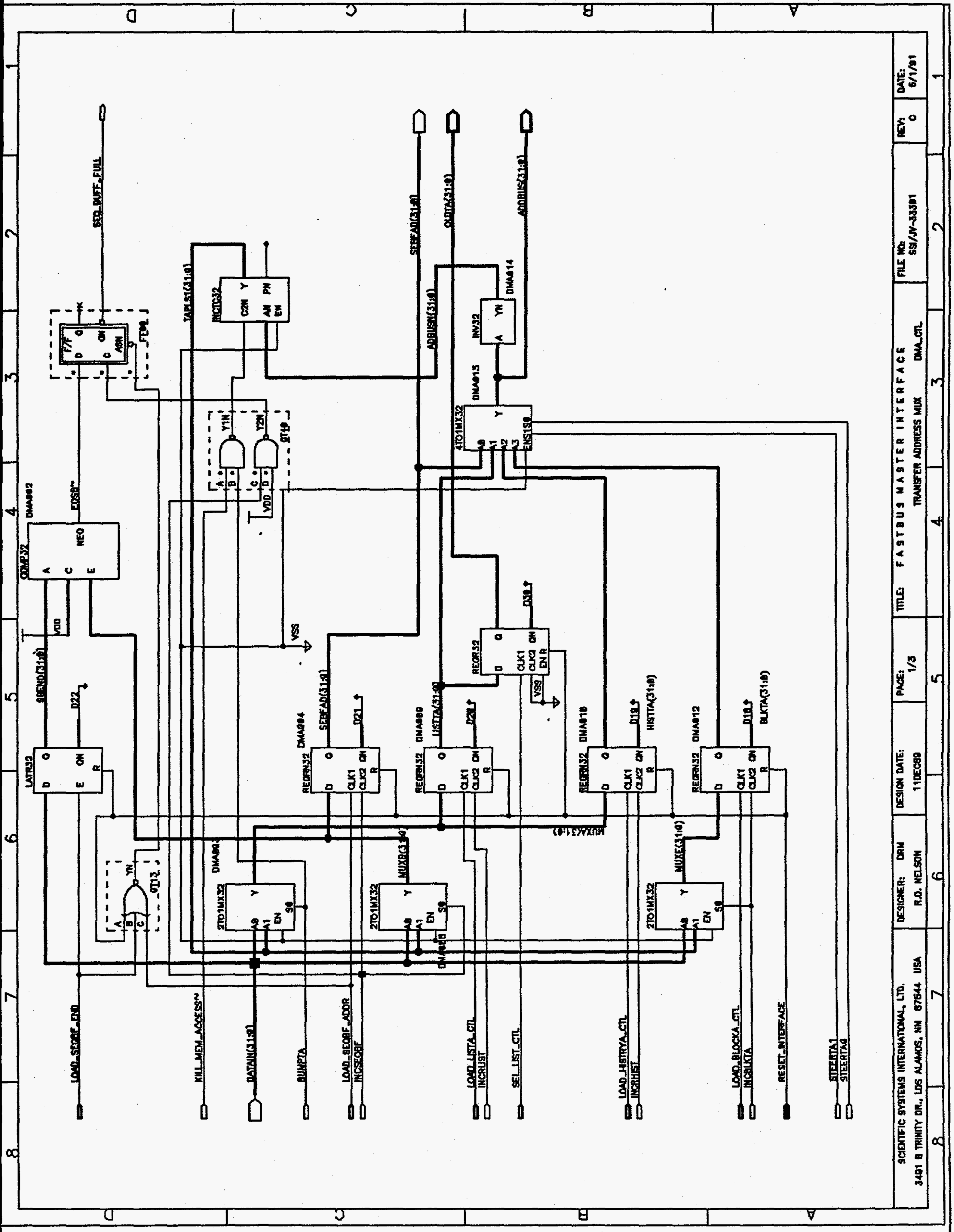


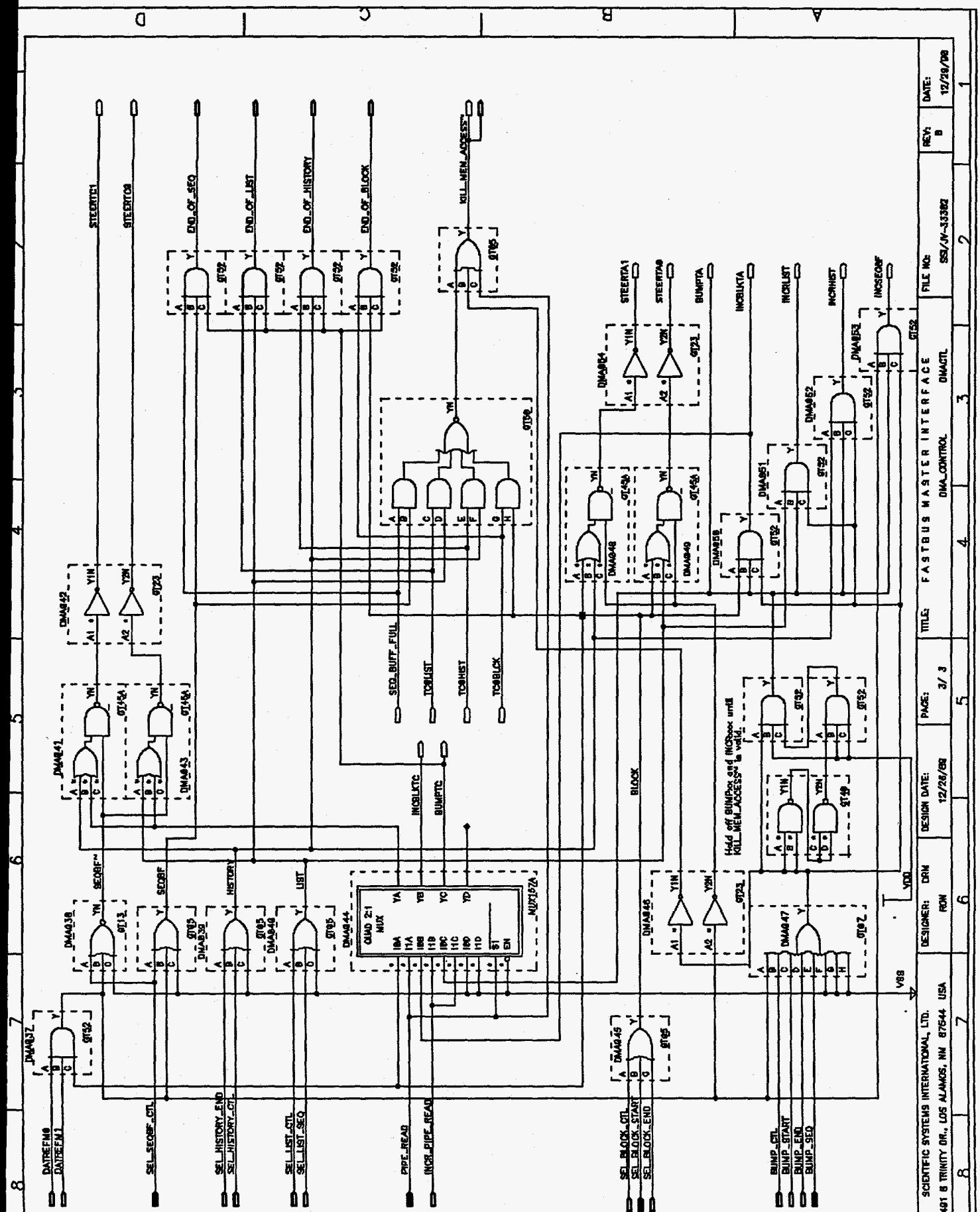
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: DRM R.D. NELSON, PH.D.	DESIGN DATE: 23DEC89	PAGE: 1/2	TITLE: FASTBUS MASTER INTERFACE CSR8 & CSR9	FILE NO: SSI/JV-33588	REV: D	DATE: 6/1/91

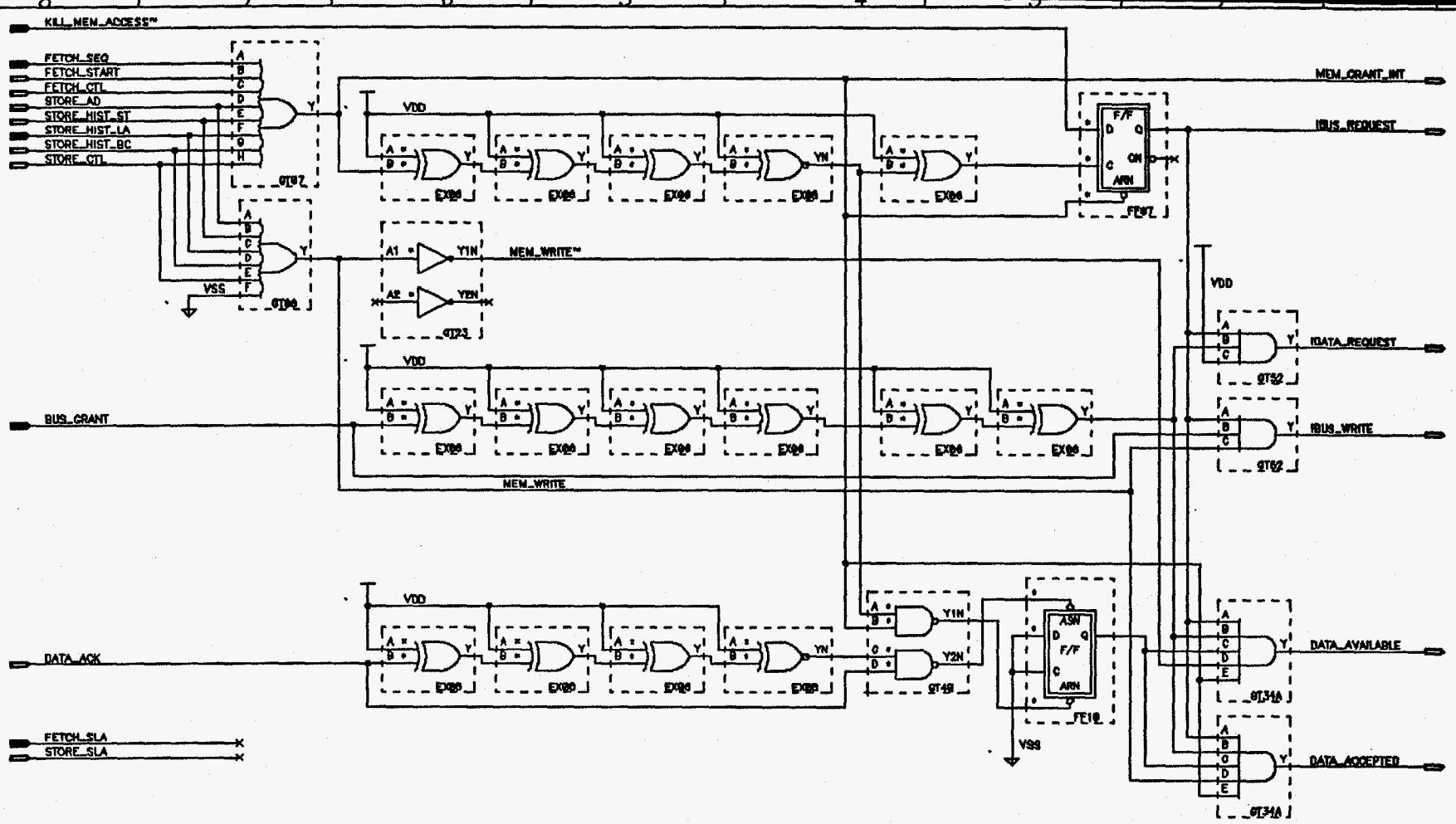


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: DRM R.D. NELSON, PH.D.	DESIGN DATE: 23DEC89	PAGE: 2/2	TITLE: FASTBUS MASTER INTERFACE WAIT TIMER REG & RETRY COUNT REG	FILE NO: SSI/JV-33581	REV: D	DATE: 6/1/91

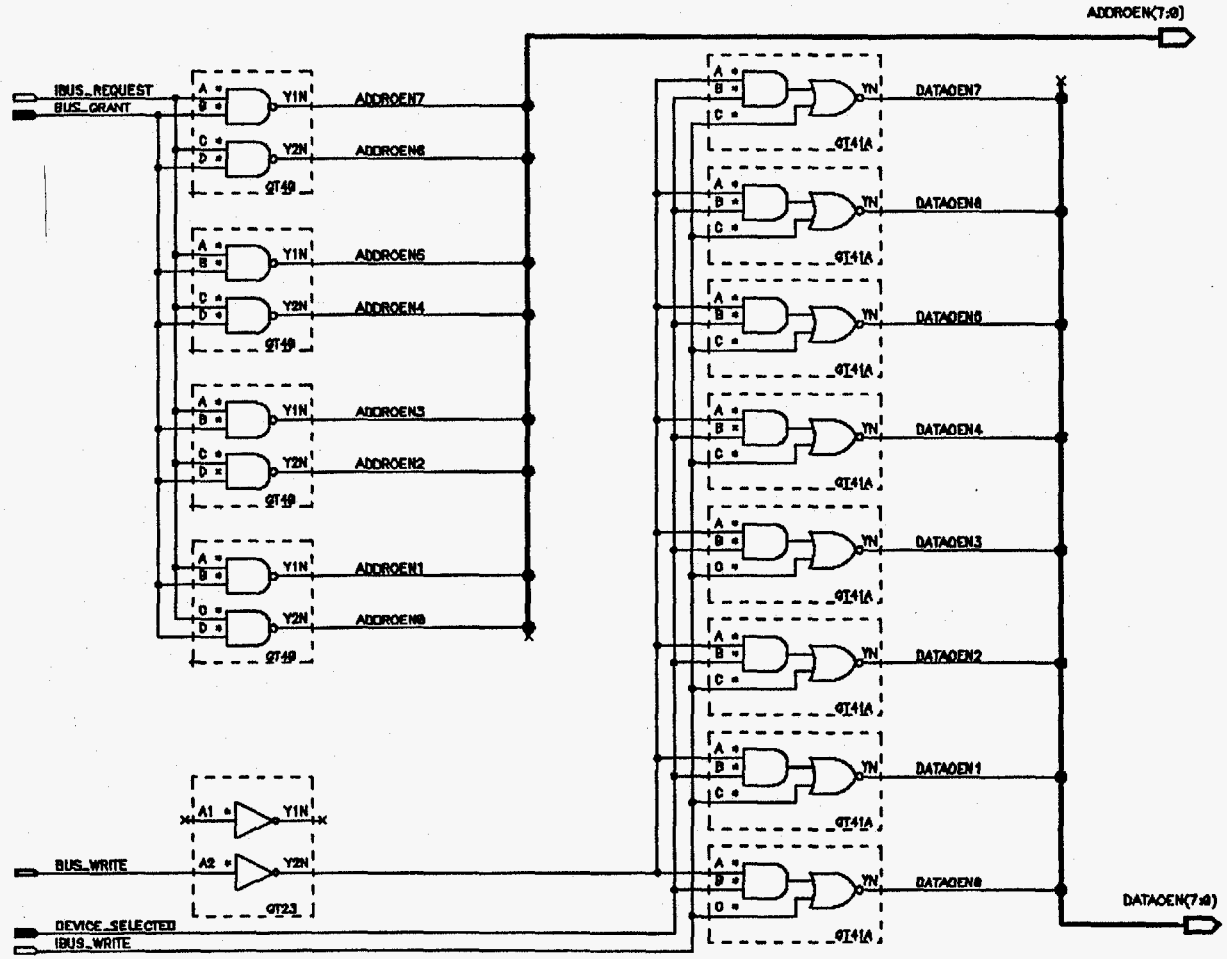

KILL_MEM_ACCESS**
DATAIN(31:0)
BUMPTC
INCRLIST
SEGFAD(31:0)
LOAD_LISTC_CTL
LOAD_HISTRYC_CTL
INCRHIST
LIST_MODE_ACTIVE
INCBLKTC
LOAD_BLOCKC_SEG
LOAD_BLOCKC_CTL
RESET_INTERFACE
STEERTC1
STEERTC0
VSS
TCMIN1(31:0)
MUXD2
MUXC(31:0)
MUXD(31:0)
MUXC(31:0),VSS,ZERO0
MUXD(31:0),VSS,ZERO0
LISTTC(31:1),TC0LIST,VSS
HISTTC(31:1),TC0HIST,VSS
BLOCKTC(31:0),VSS
TC0LIST
TC0HIST
TC0BLCK
TC0BLCK**
CNTBUS(31:0)
BLOCKTC(31:0)
ZERO0
END_OF_BUFFER
NEXT_TO_LAST
LATCHEOB
LIST
HISTORY
BLOCK
2TO1MX32
4TO1MX32
REGRN32
INCTC32
INV32
DMA826
DMA827
DMA828
DMA829
DMA830
DMA831
DMA832
DMA833
DMA834
DMA835
DMA836
DMA837
DMA838
DMA839
DMA840
DMA841
DMA842
DMA843
DMA844
DMA845
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3481 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: DRM
R.O. NELSON
DESIGN DATE:
110EC89
PAGE:
1/ 3
TITLE: FASTBUS MASTER INTERFACE
TRANSFER COUNT MUX DMACTL
FILE NO:
SSI/JV-33388
REV:
D
DATE:
6/1/91


FASTBUS MASTER INTERFACE
TRANSFER ADDRESS MUX
DMA_CTL
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER:
R.D. NELSON
DESIGN DATE:
11DEC89
PAGE:
1/3
REV:
0
DATE:
6/1/91

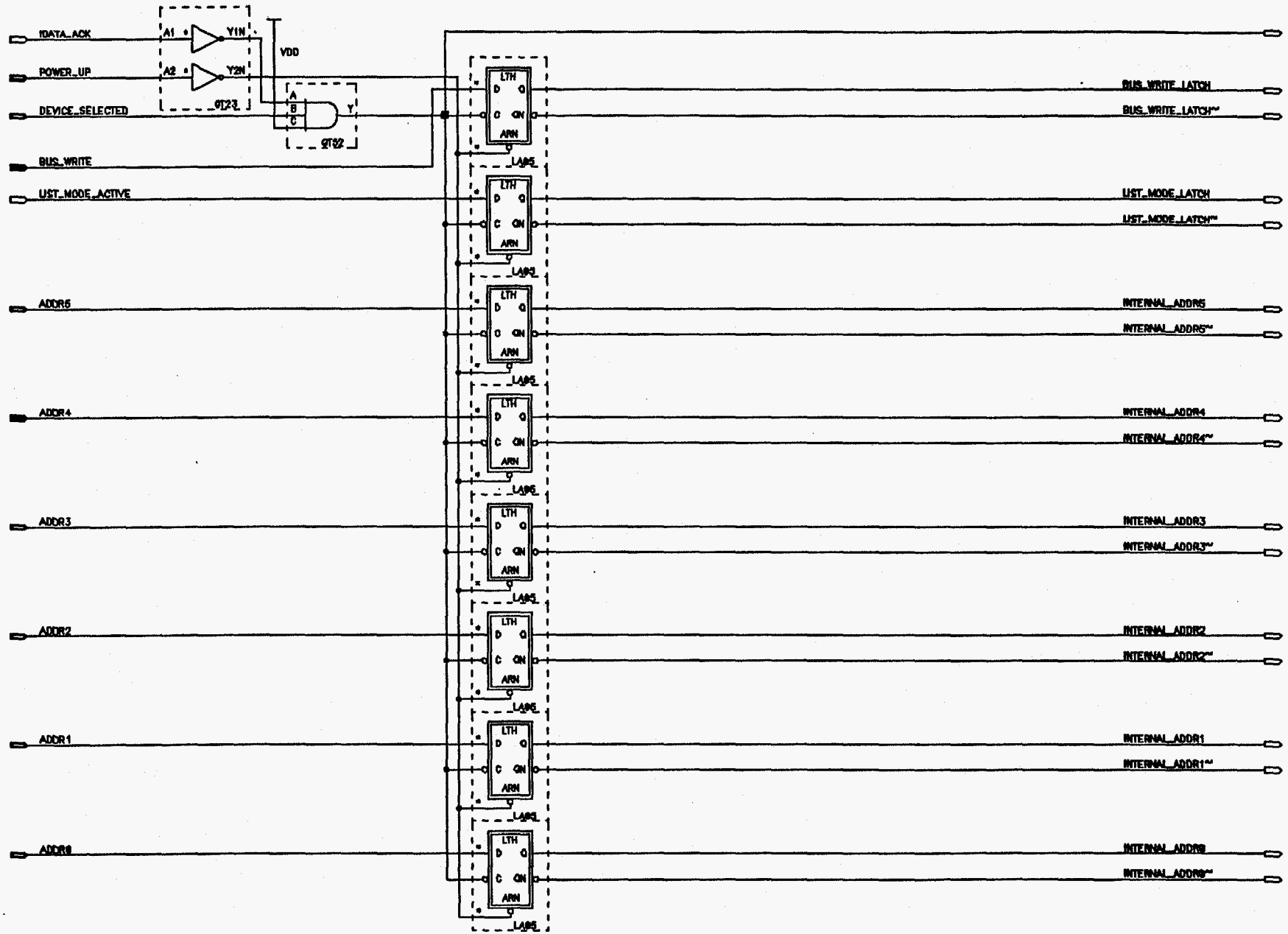

TITLE: FASTBUS MASTER INTERFACE
DMA_CONTROL
DMACTL
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: DRW
RDW
DESIGN DATE: 12/28/88
PAGE: 3/ 3
FILE NO: SSI/JN-33382
REV: B
DATE: 12/29/88
STEERTA1
STEERTA0
END_OF_SEQ
END_OF_LIST
END_OF_HISTORY
END_OF_BLOCK
KILL_MEM_ACCESS*
SEQ_BUFF_FULL
TOBLIST
TOBHIST
TOBBLOCK
INCBLKTC
BUMPTC
BLOCK
SEQBF*
SEQBF
HISTORY
LIST
QUAD 2:1 MUX
MUXTSTA
Hold off BUMPxx and INCRxxx until KILL_MEM_ACCESS* is valid.
DATREFN0
DATREFN1
SEL_SEQBF_CTL
SEL_HISTORY_END
SEL_HISTORY_CTL
SEL_LIST_CTL
SEL_LIST_SEQ
PIPE_READ
INCR_PIPE_READ
SEL_BLOCK_CTL
SEL_BLOCK_START
SEL_BLOCK_END
BUMP_CTL
BUMP_START
BUMP_END
BUMP_SEQ
STEERTA1
STEERTA0
BUMPTA
INCBLKTA
INCLIST
INCHIST
INCSEQBF



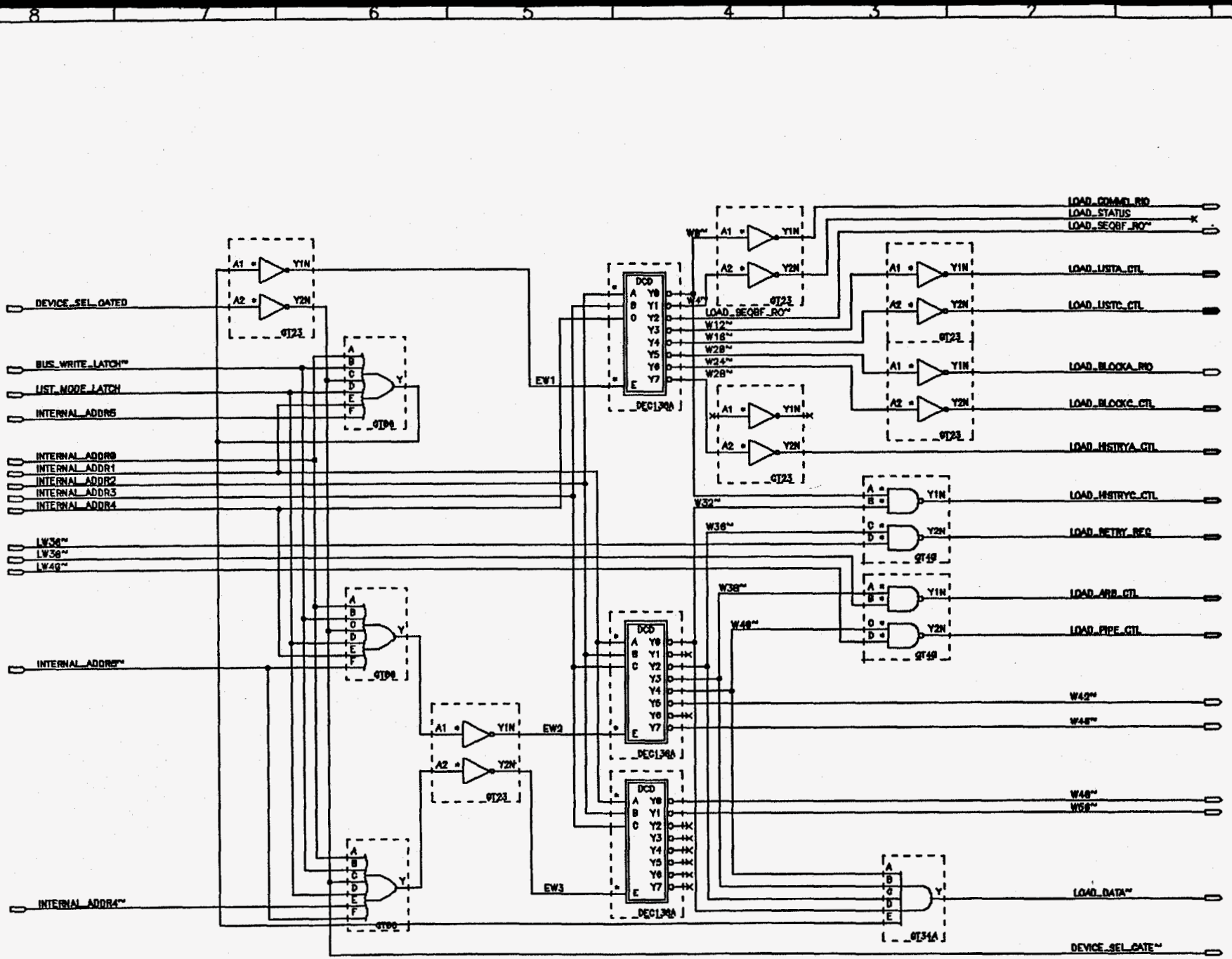
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 1AUG92	PAGE: 1/2	TITLE: FASTBUS RISC PROCESSOR MODULE MEMORY STORE AND FETCH INTERFACE SIGNALS	FILE NO: SSI/JV-34000	REV: A	DATE: 1AUG92



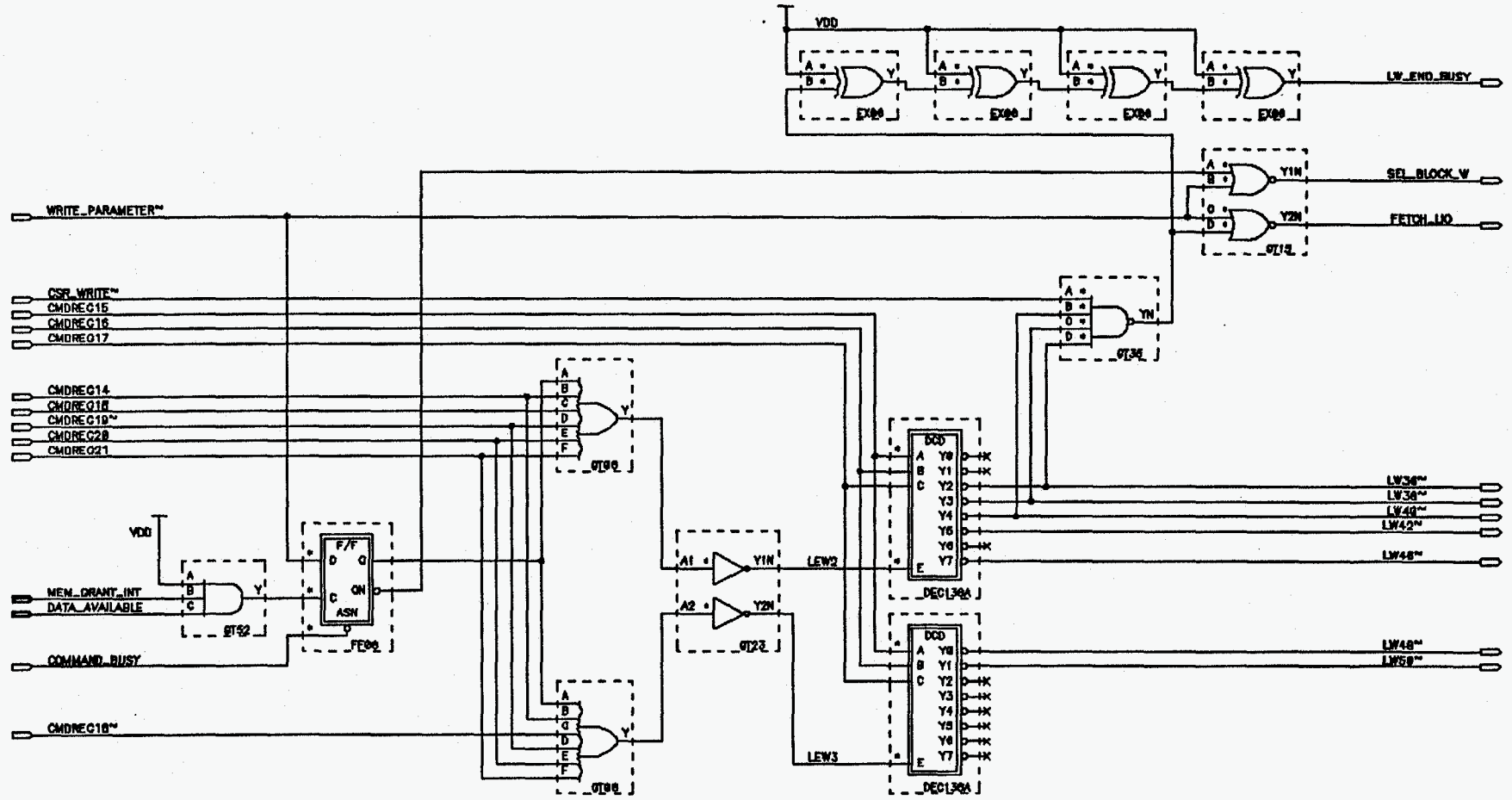
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 29AUG92	PAGE: 2/2	TITLE: FASTBUS RISC PROCESSOR MODULE DIRECT MEMORY MAPPED READ/WRITE HANDSHAKE	FILE NO: SSI/JV-34881	REV: A	DATE: 29AUG92



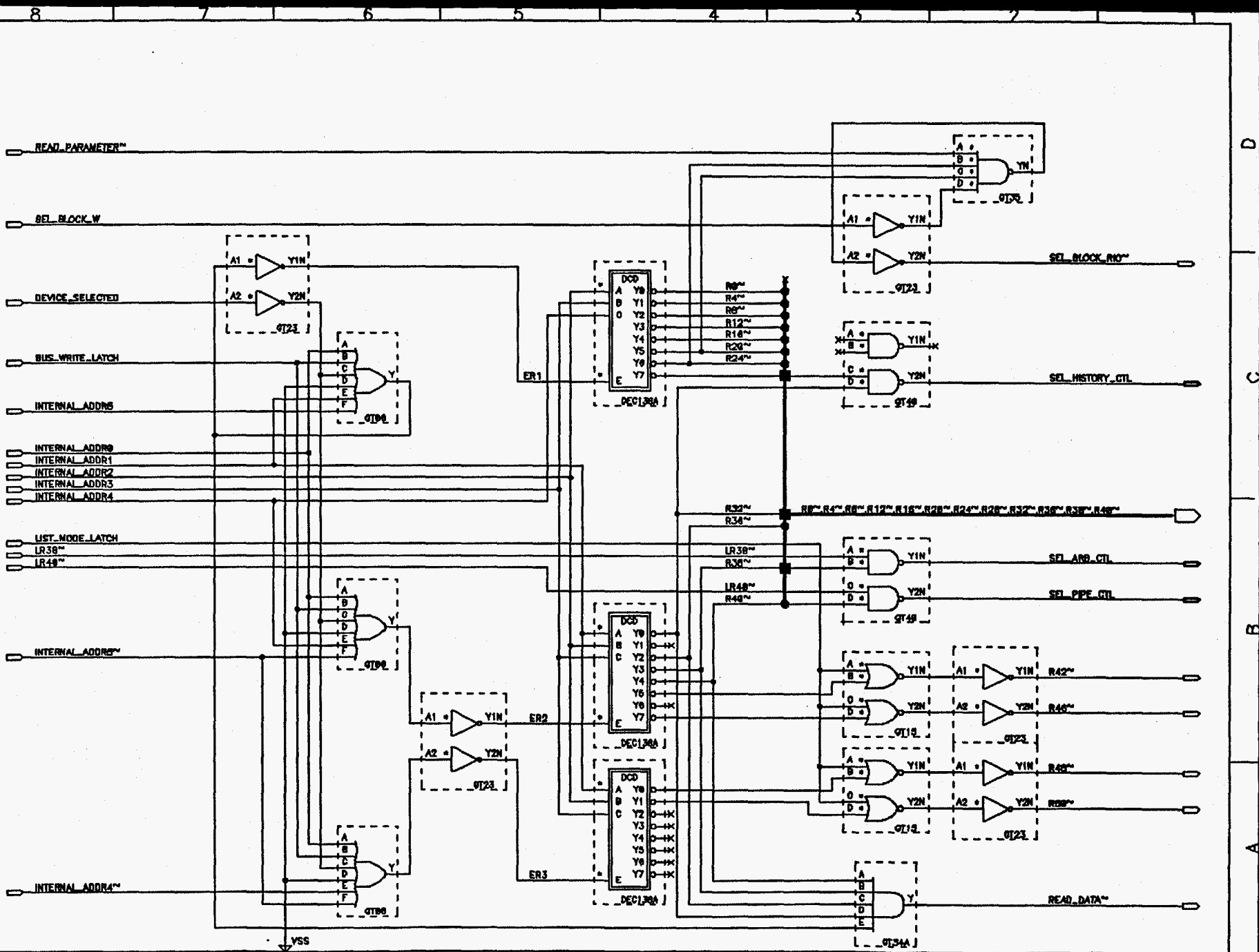
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3421 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: RON	DESIGN DATE: 8 AUG 92	PAGE: 1/19	TITLE: FASTBUS RISC PROCESSOR MODULE FMI CONTROL LOGIC	FILE NO: SSI/JV-35888	REV: A	DATE: 8/8/92



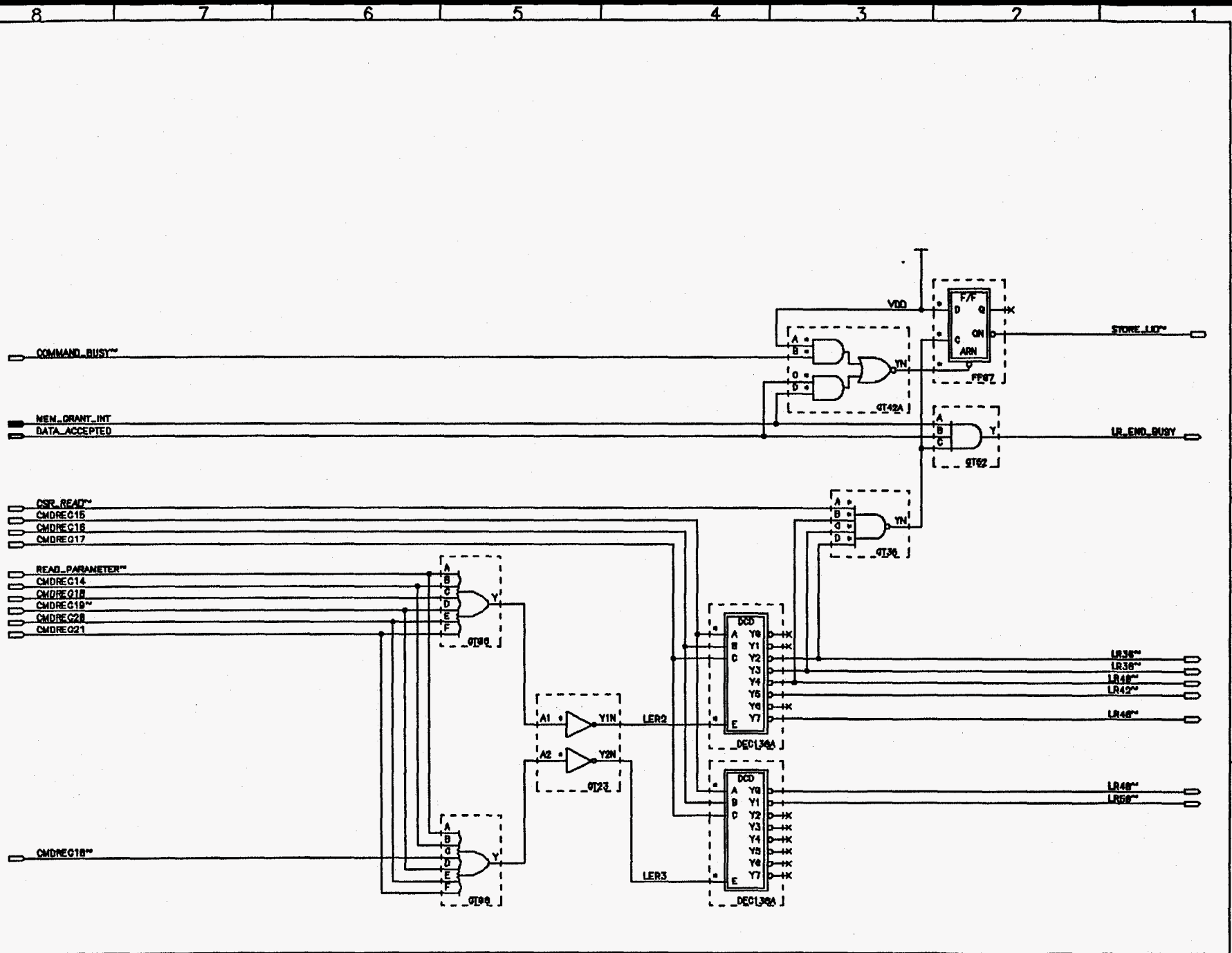
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 29AUG92	PAGE: 2/10	TITLE: FASTBUS RISC PROCESSOR MODULE DIRECT MEMORY MAPPED WRITE DECODE	FILE NO: SSI/JV-36001	REV: A	DATE: 29AUG92



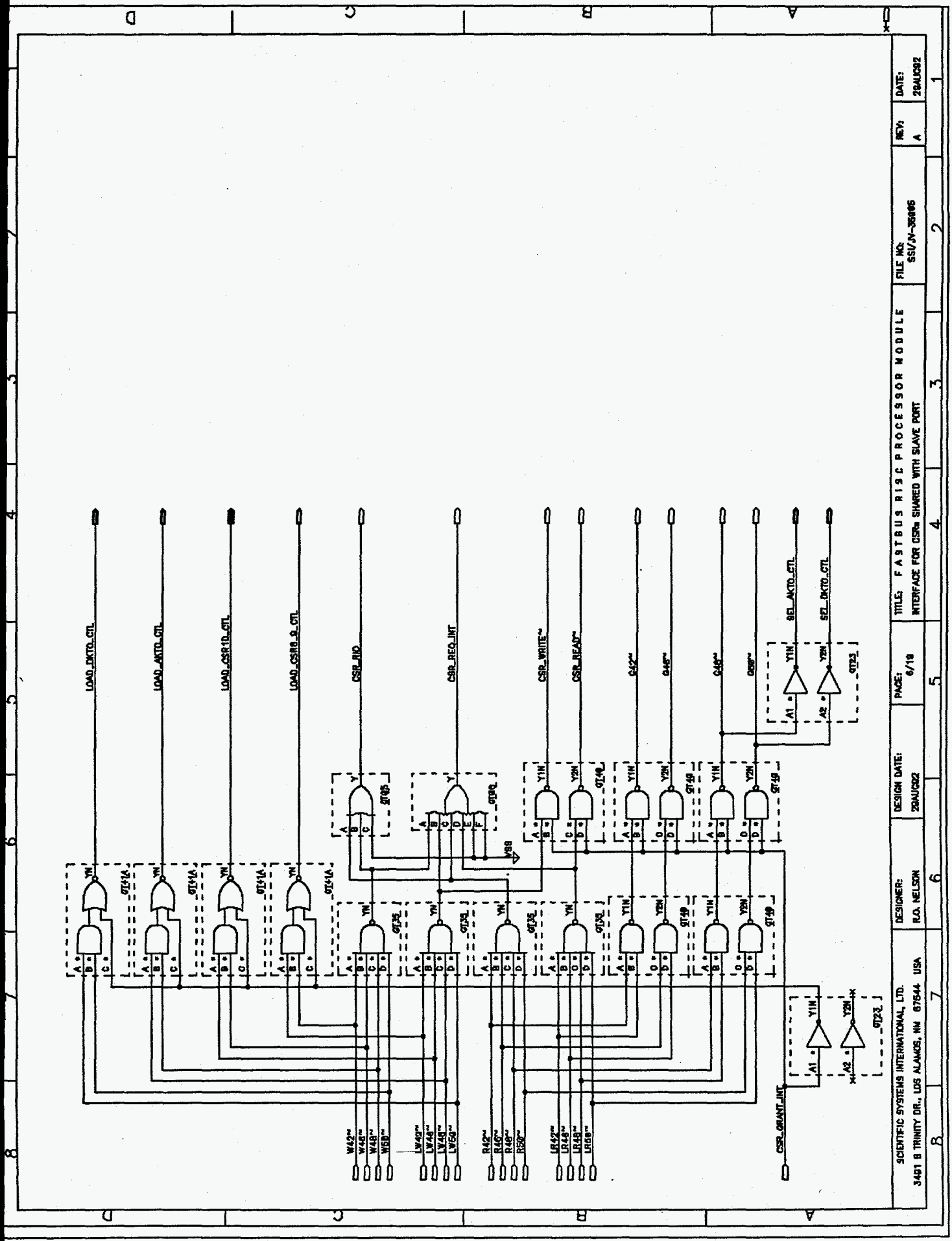
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 29AUG92	PAGE: 3/19	TITLE: FASTBUS RISC PROCESSOR MODULE UST DRIVEN WRITE PARAMETER DECODING			FILE NO: SSI/JV-36002	REV: A	DATE: 29AUG92

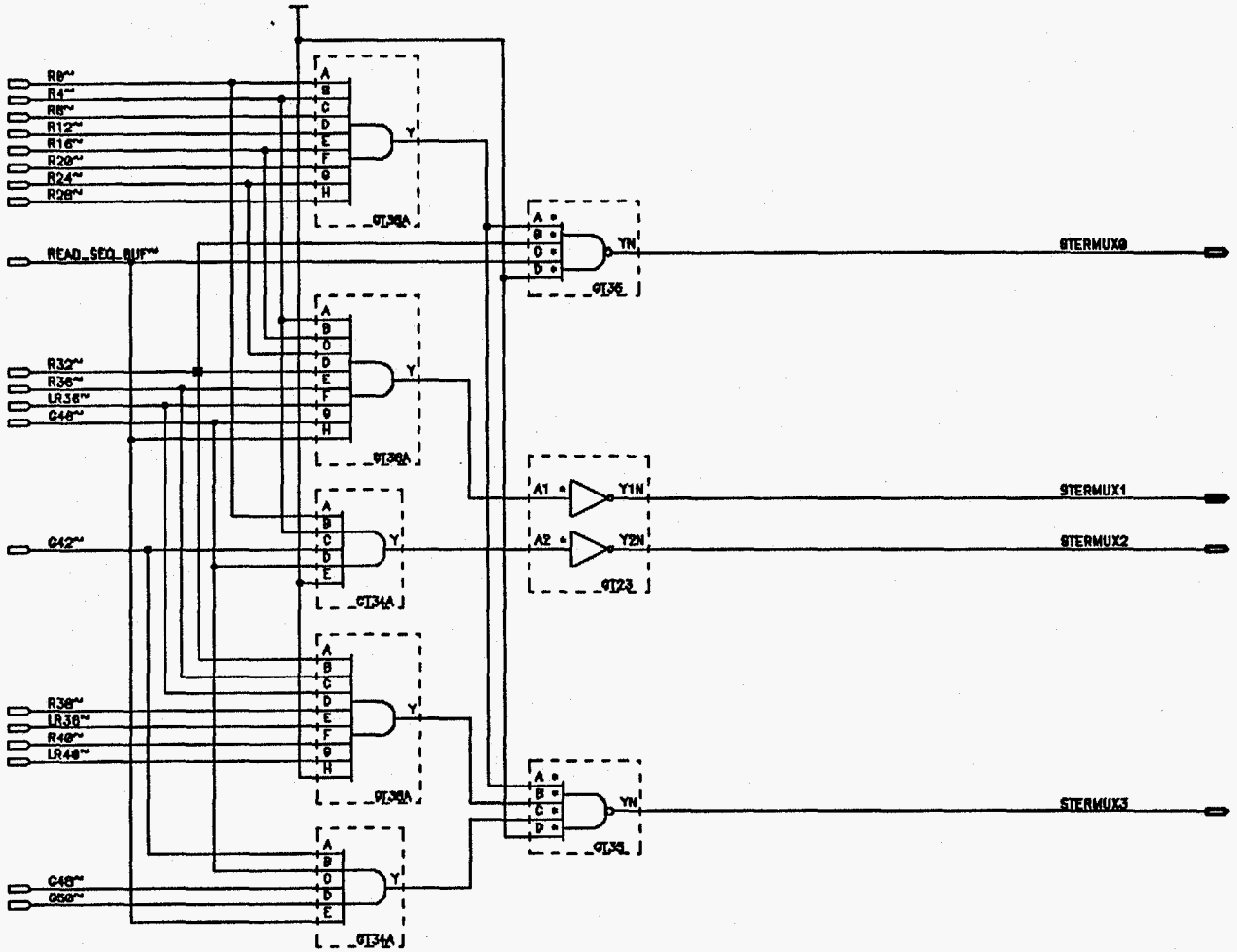


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 29AUG92	PAGE: 4/19	TITLE: FASTBUS RISC PROCESSOR MODULE DIRECT MEMORY MAPPED READ DECODE	FILE NO: SSI/JV-36003	REV: A	DATE: 29AUG92

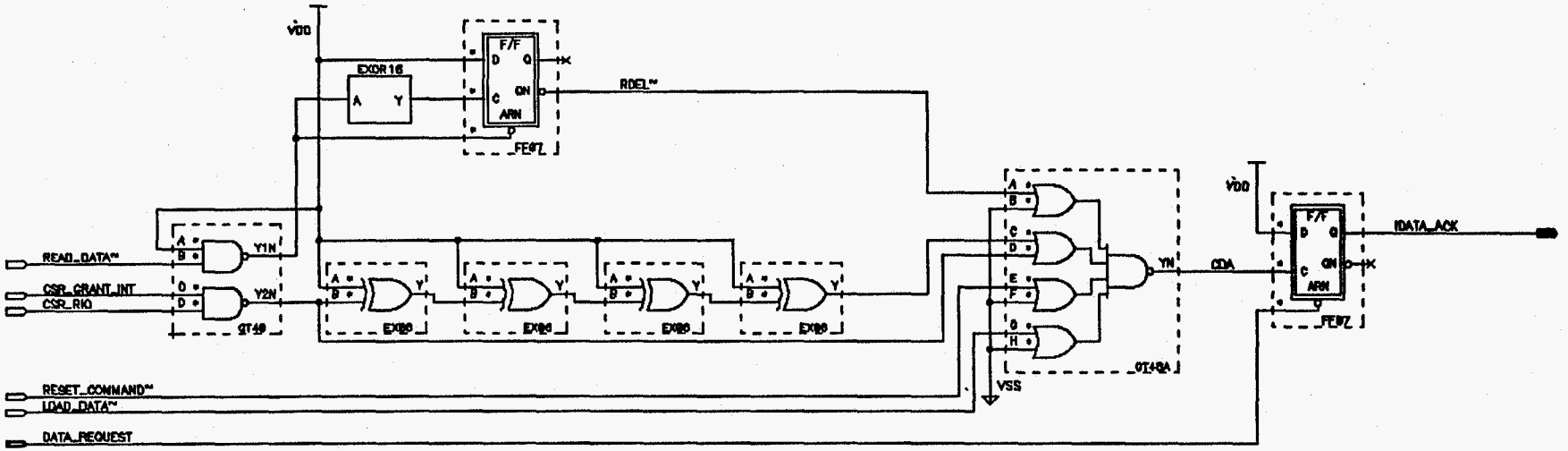


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 29AUG92	PAGE: 6/19	TITLE: FASTBUS RISC PROCESSOR MODULE LIST DRIVEN READ PARAMETER DECODING	FILE NO: SSI/JV-36084	REV: A	DATE: 29AUG92

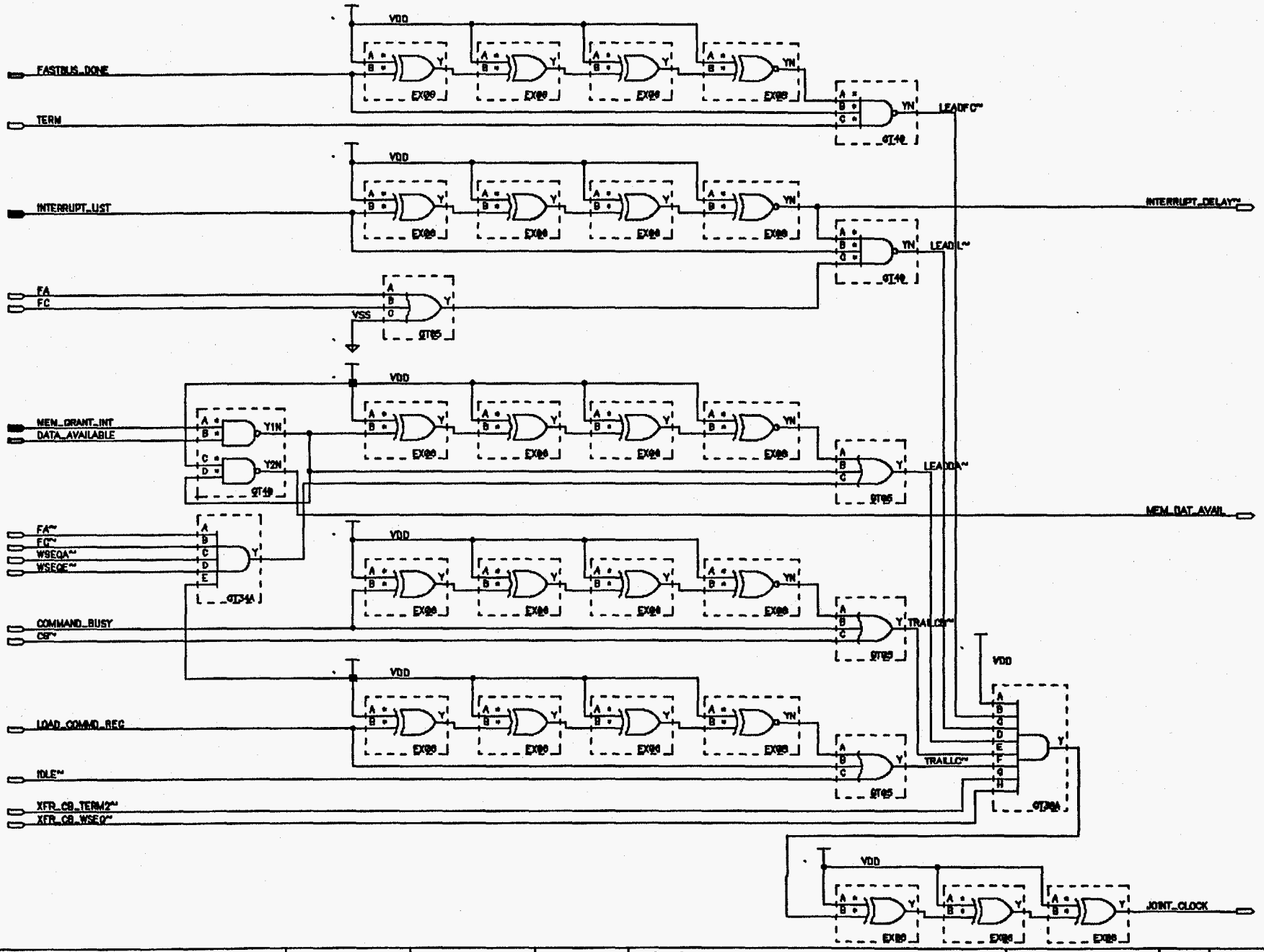

LOAD_DXTO_CTL
LOAD_AXTO_CTL
LOAD_CSR10_CTL
LOAD_CSR8_9_CTL
CSR_RID
CSR_REQ_INT
CSR_WRITE~
CSR_READ~
SEL_AXTO_CTL
SEL_DXTO_CTL
CSR_GRANT_INT
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 29AUG92
PAGE: 6/19
TITLE: FASTBUS RISC PROCESSOR MODULE
INTERFACE FOR CSRs SHARED WITH SLAVE PORT
FILE NO: SSI/JV-35005
REV: A
DATE: 29AUG92



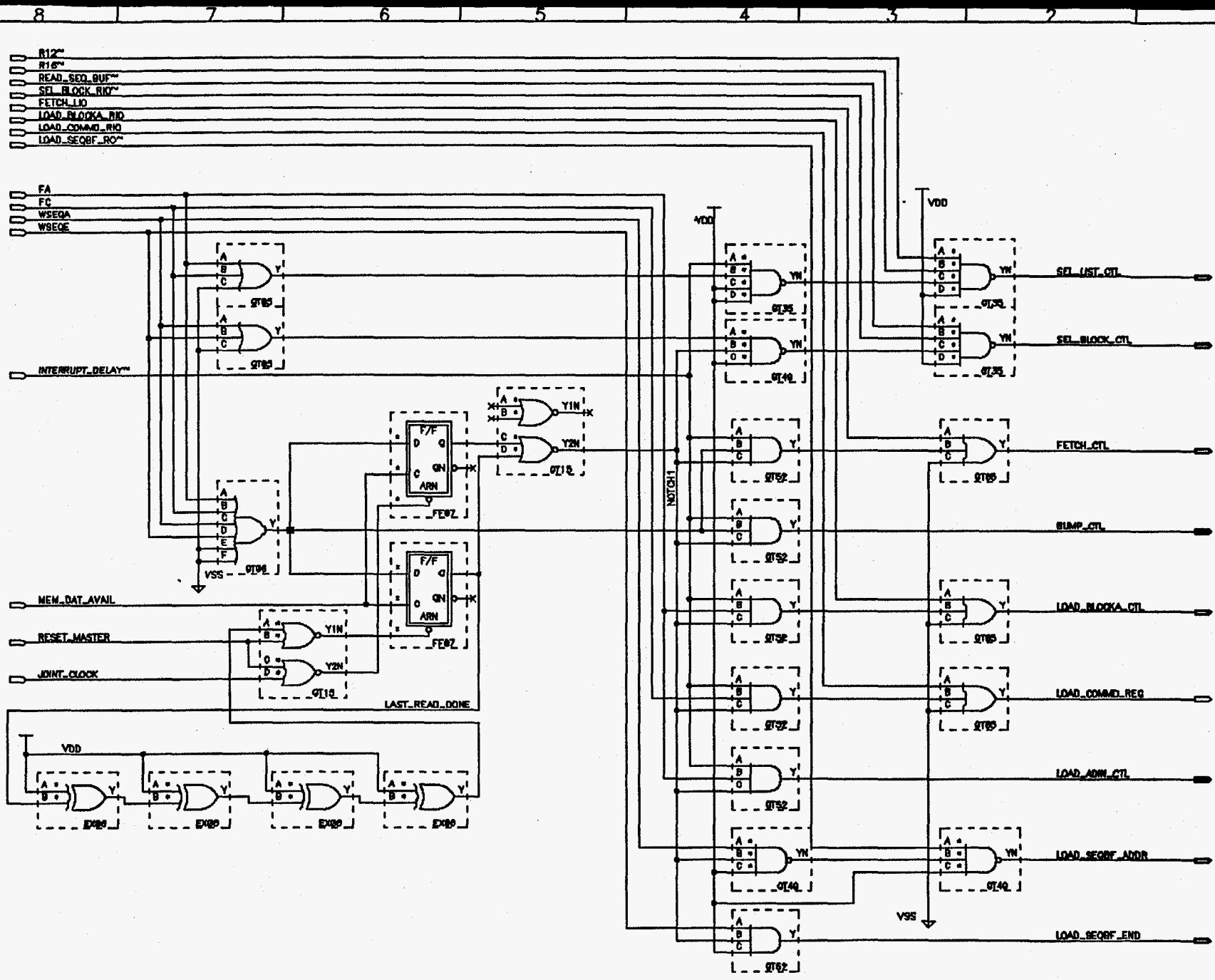
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 29AUG92	PAGE: 7/19	TITLE: FASTBUS RISC PROCESSOR MODULE ENCODING FOR REGISTER READ STEERING	FILE NO: SSI/JV-36666	REV: A	DATE: 29AUG92



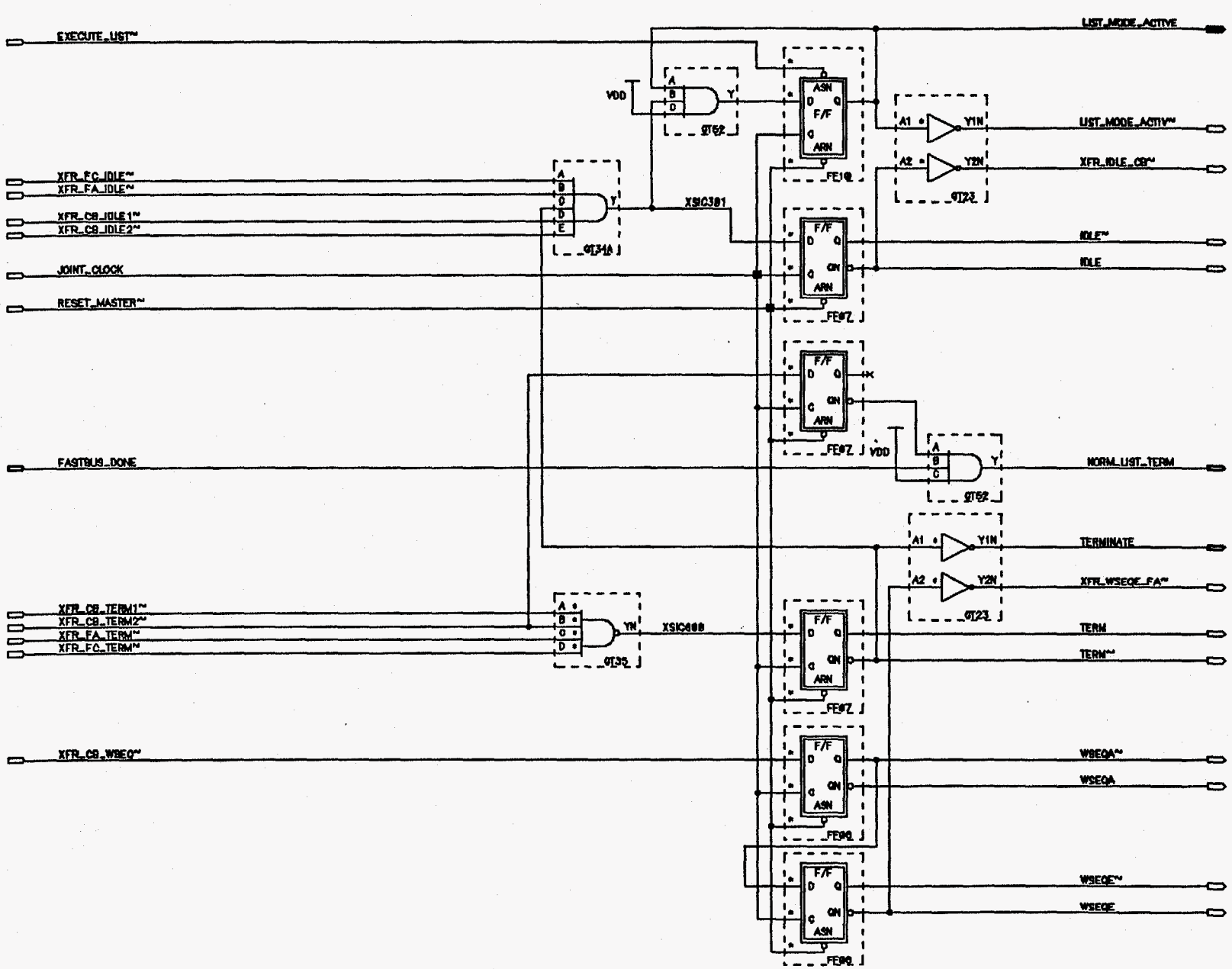
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 29AUG92	PAGE: 6/19	TITLE: FASTBUS RISC PROCESSOR MODULE DIRECT MEMORY MAPPED READ/WRITE HANDSHAKE	FILE NO: SSI/JV-36007	REV: A	DATE: 29AUG92



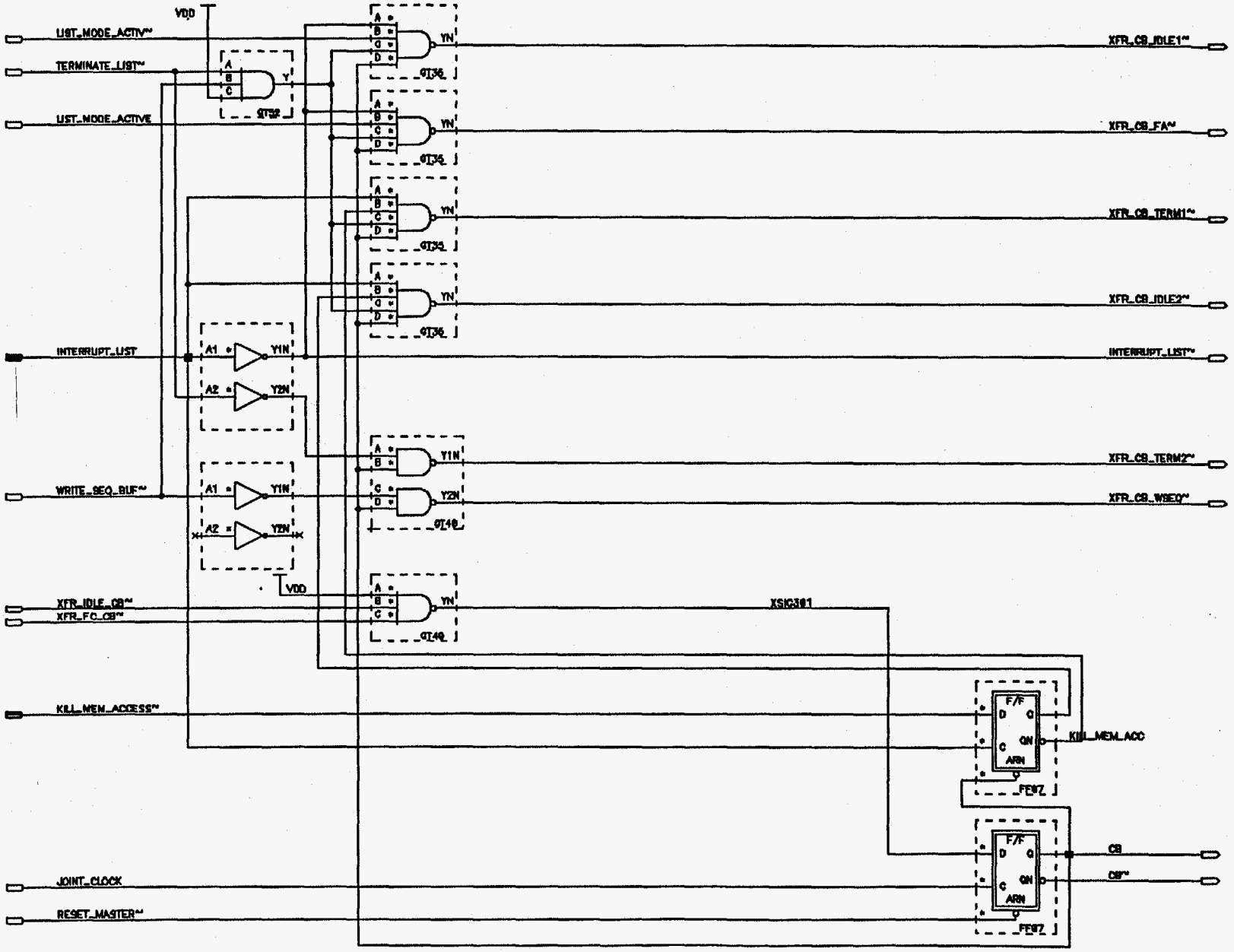
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 14SEP92	PAGE: 9/19	TITLE: FASTBUS RISC PROCESSOR MODULE TRIGGER EVENTS FOR LIST SEQUENCER	FILE NO: SSI/JV-36888	REV: A	DATE: 14SEP92



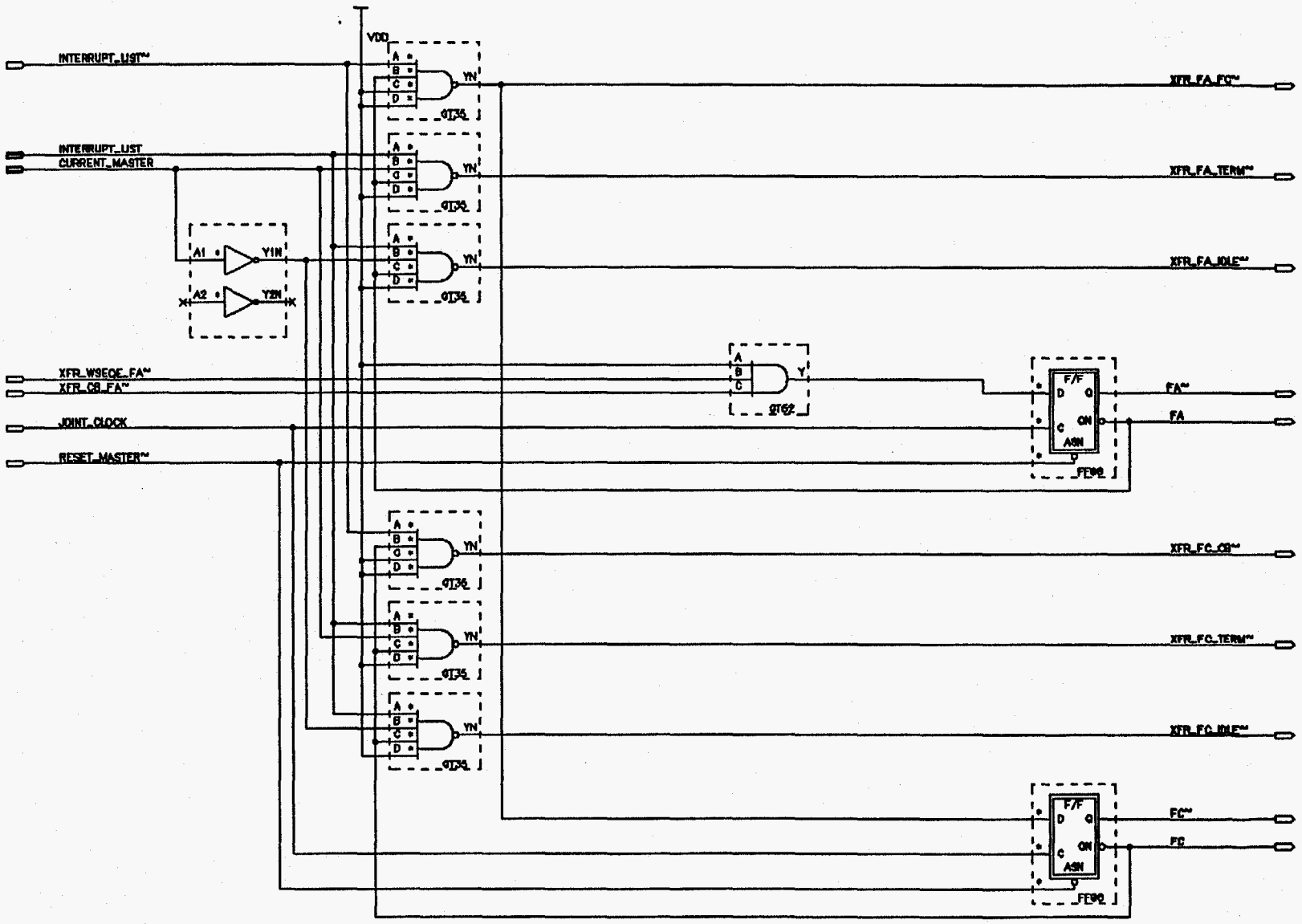
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 14SEP92	PAGE: 10/19	TITLE: FASTBUS RISC PROCESSOR MODULE MEMORY INTERFACE SIGNALS	FILE NO: SSI/JV-35000	REV: A	DATE: 14SEP92



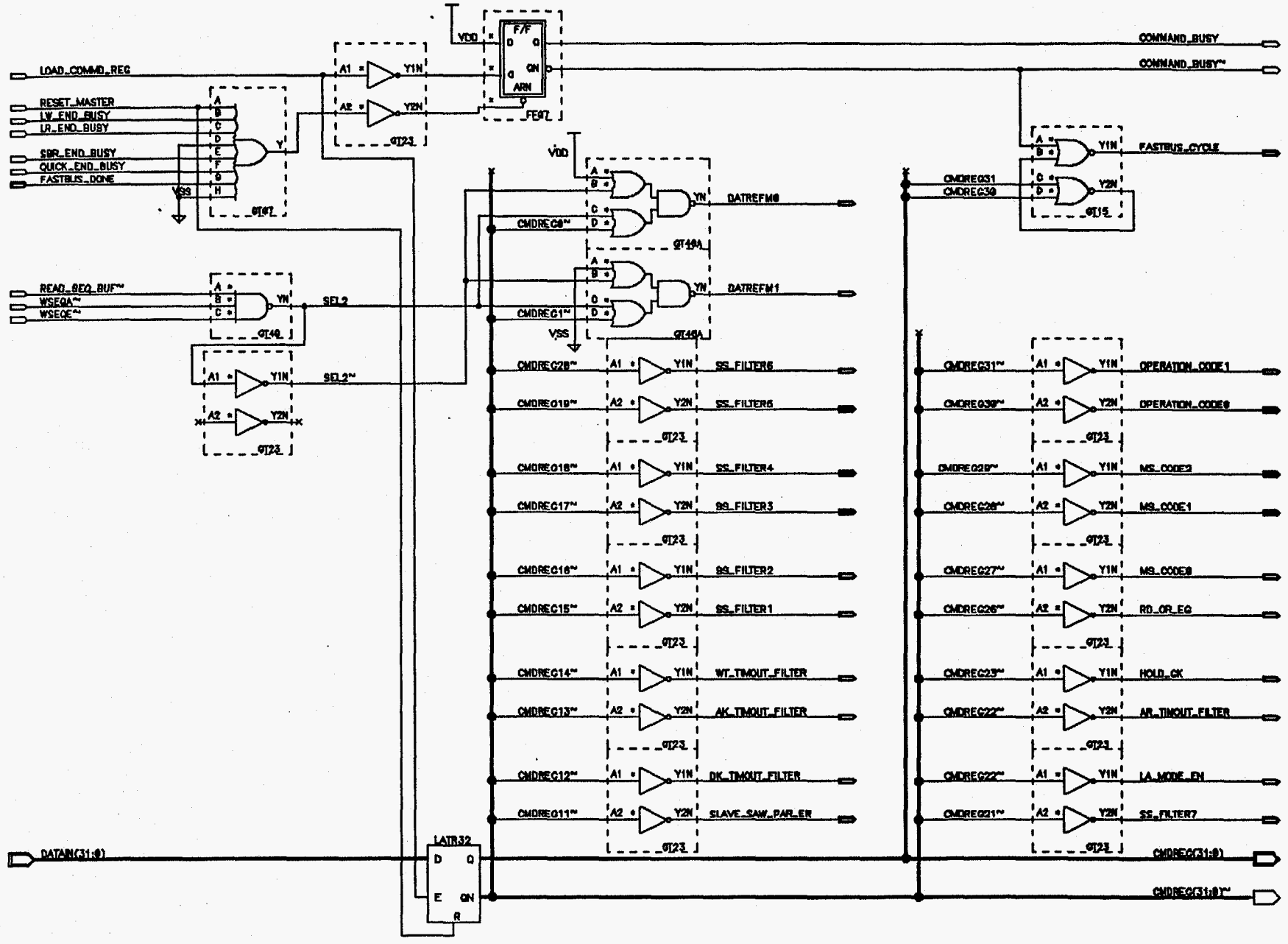
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 20SEP92	PAGE: 11/19	TITLE: FASTBUS RISC PROCESSOR MODULE IDLE AND TERMINATE STATES AND WRITE SEQUENTIAL BUFFER STAYES	FILE NO: SSI/JV-358010	REV: A	DATE: 20SEP92



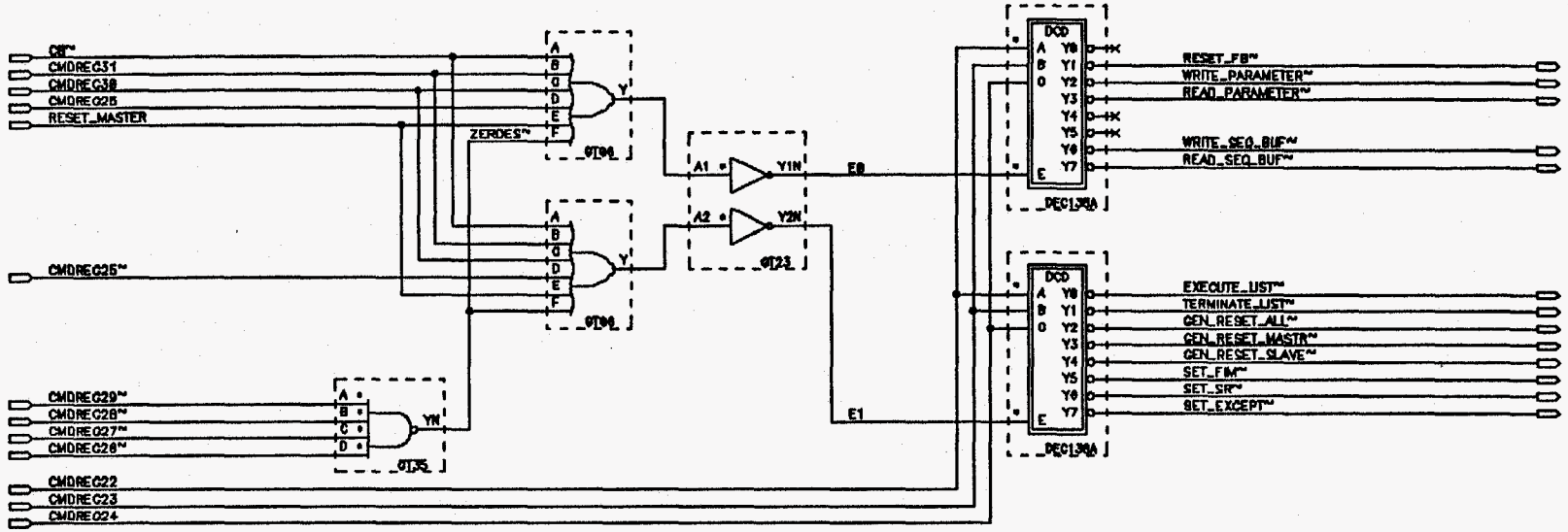
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.G. NELSON	DESIGN DATE: 20SEP92	PAGE: 12/19	TITLE: FASTBUS RISC PROCESSOR MODULE COMMD_BUSY STATE	FILE NO: SSI/JV-360811	REV: A	DATE: 20SEP92



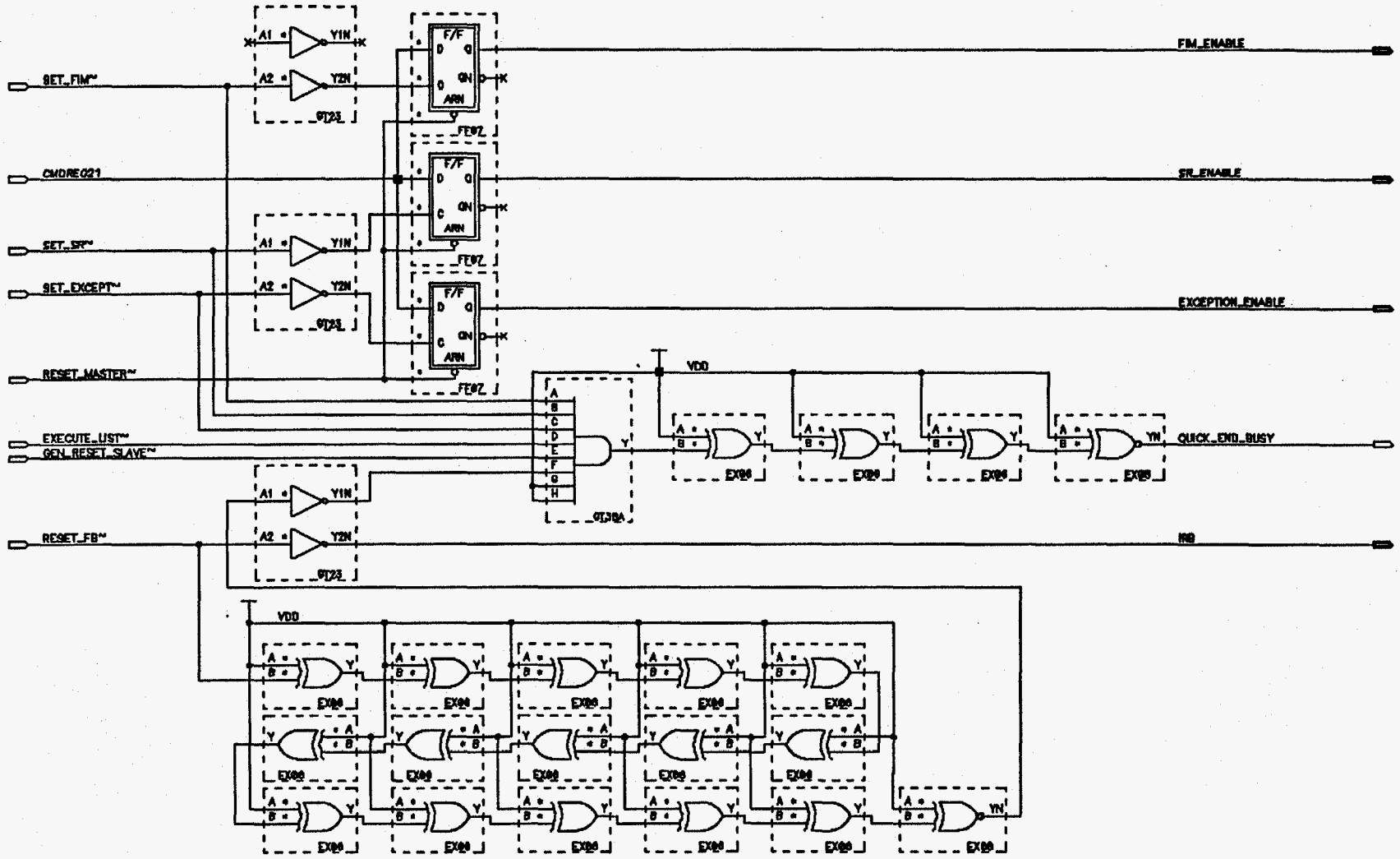
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 28SEP92	PAGE: 13/19	TITLE: FASTBUS RISC PROCESSOR MODULE FETCH ADDRESS AND FETCH COMMAND STATES	FILE NO: SSI/JV-358812	REV: A	DATE: 28SEP92



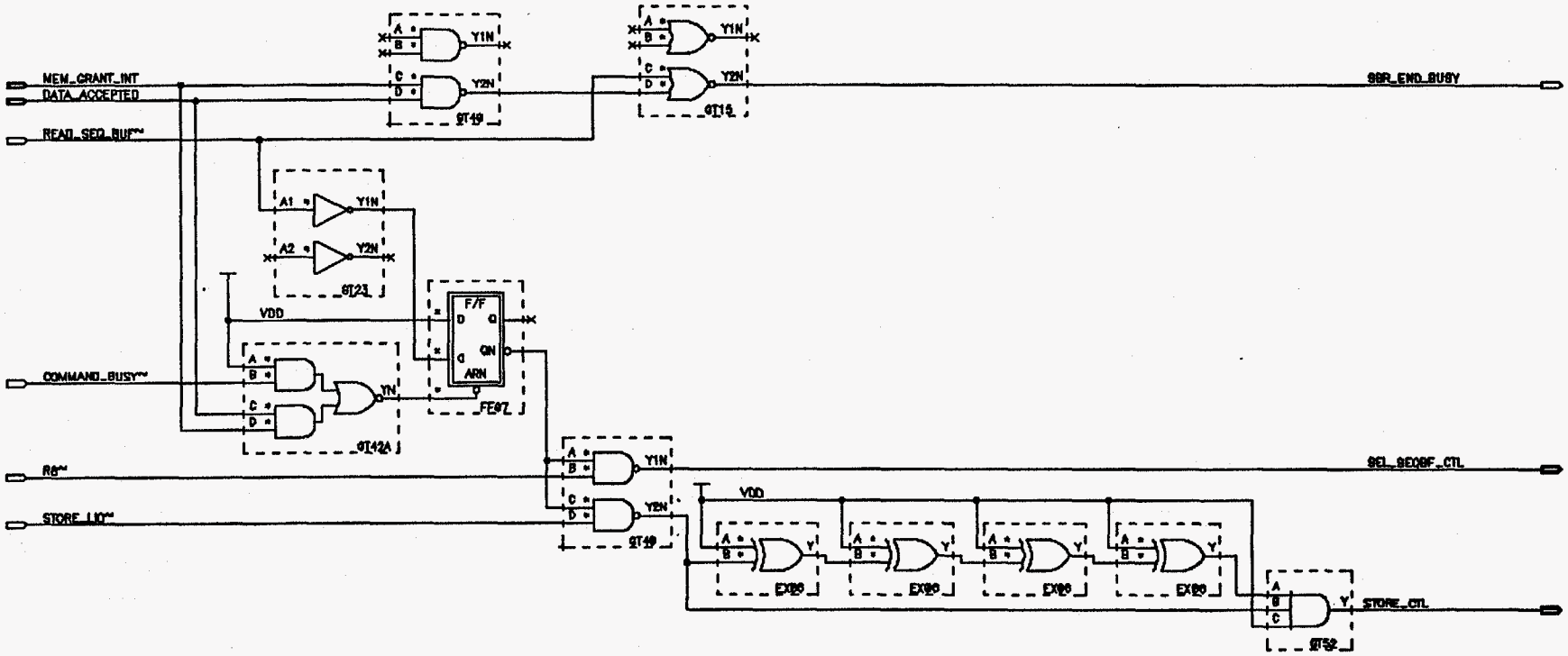
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 26SEP92	PAGE: 14/19	TITLE: FASTBUS RISC PROCESSOR MODULE COMMAND REGISTER	FILE NO: SSI/JV-368013	REV: A	DATE: 26SEP92



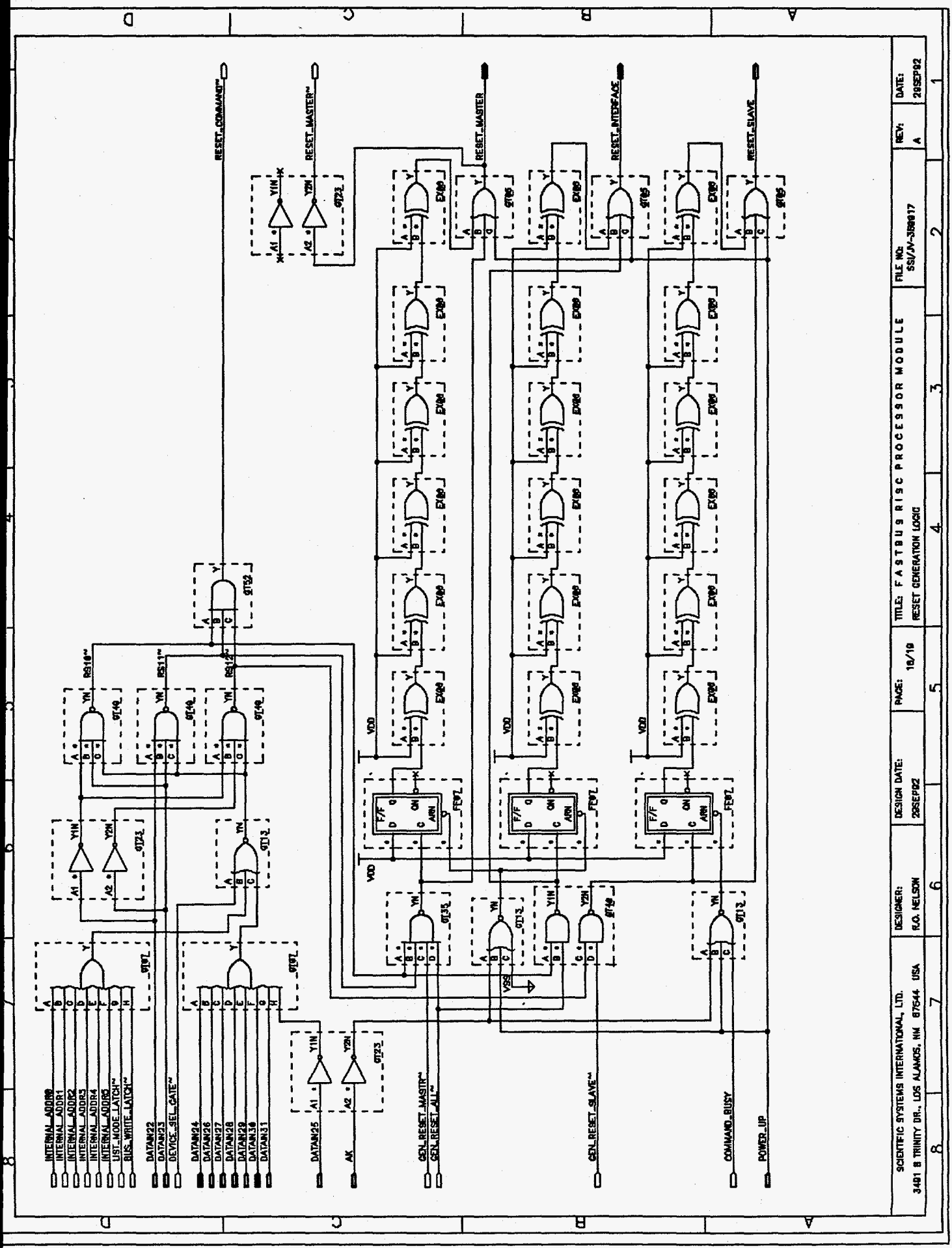
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 26SEP92	PAGE: 15/19	TITLE: FASTBUS RISC PROCESSOR MODULE COMMAND DECODE	FILE NO: SSI/JV-388814	REV: A	DATE: 26EP92

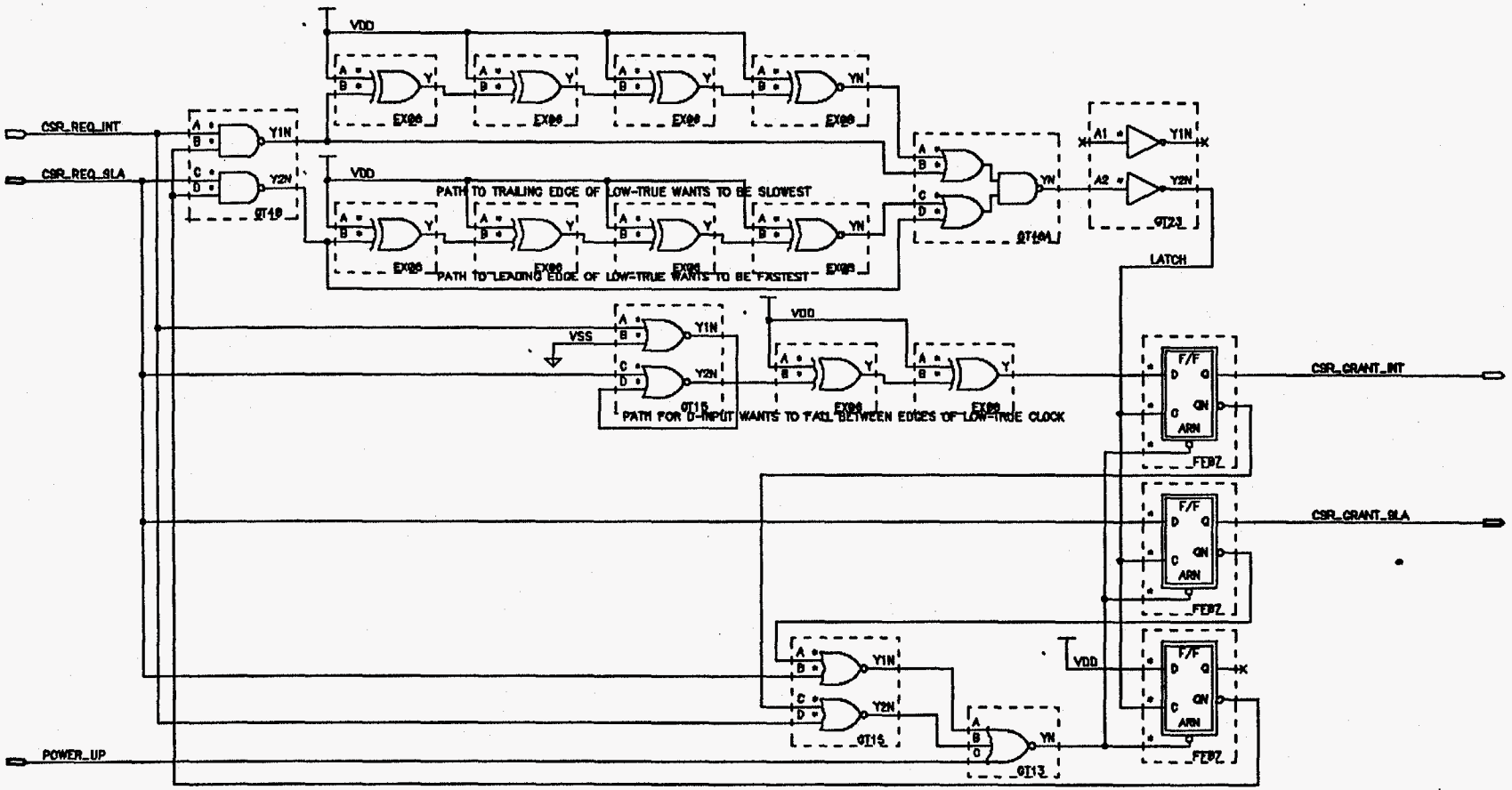


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 26SEP92	PAGE: 14/19	TITLE: FASTBUS RISC PROCESSOR MODULE MISCELLANEOUS COMMANDS	FILE NO: SSI/JV-350018	REV: A	DATE: 26SEP92	

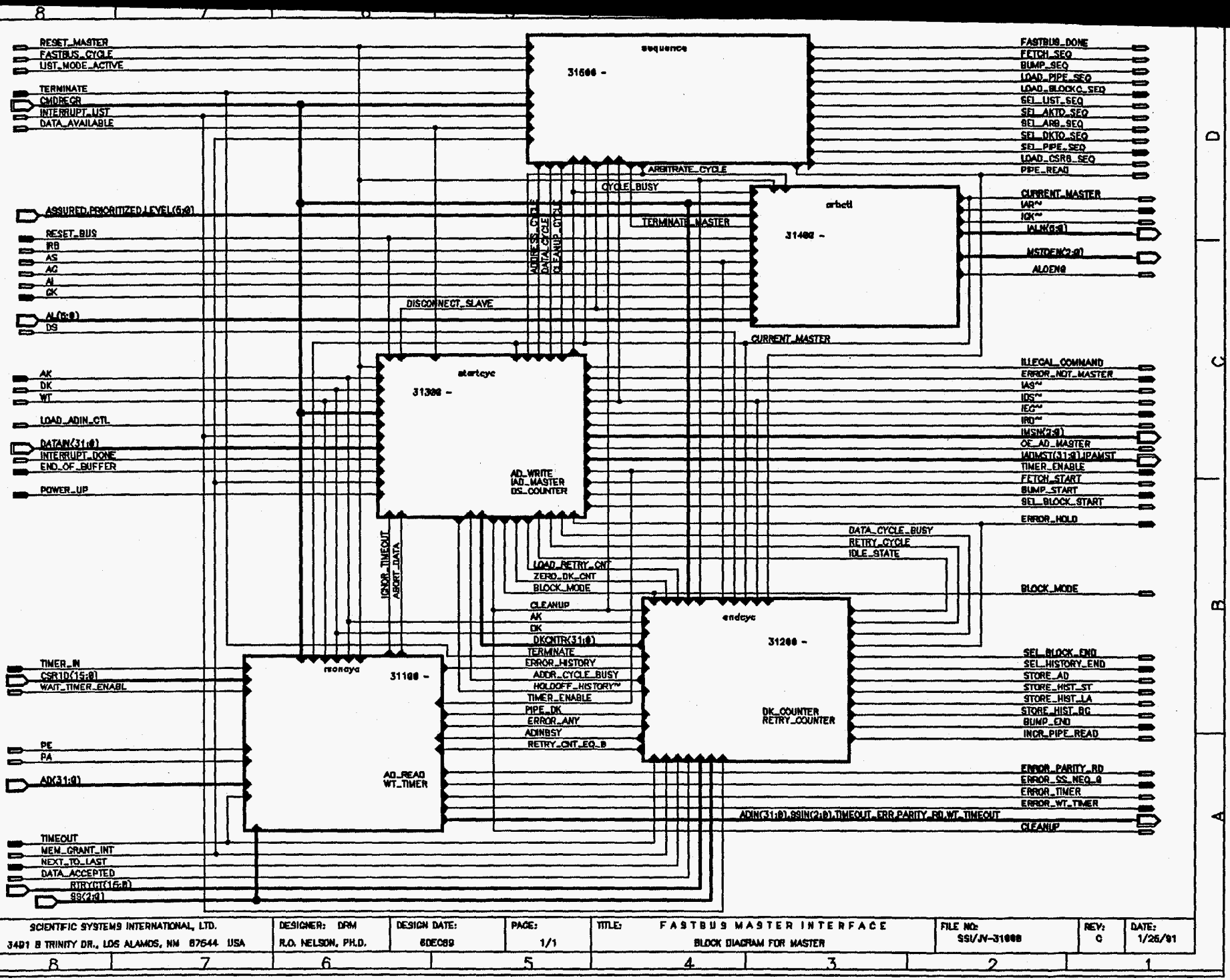


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3481 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 29SEP92	PAGE: 17/19	TITLE: FASTBUS RISC PROCESSOR MODULE READ SEQUENTIAL BUFFER	FILE NO: SSI/JV-358016	REV: A	DATE: 29SEP92

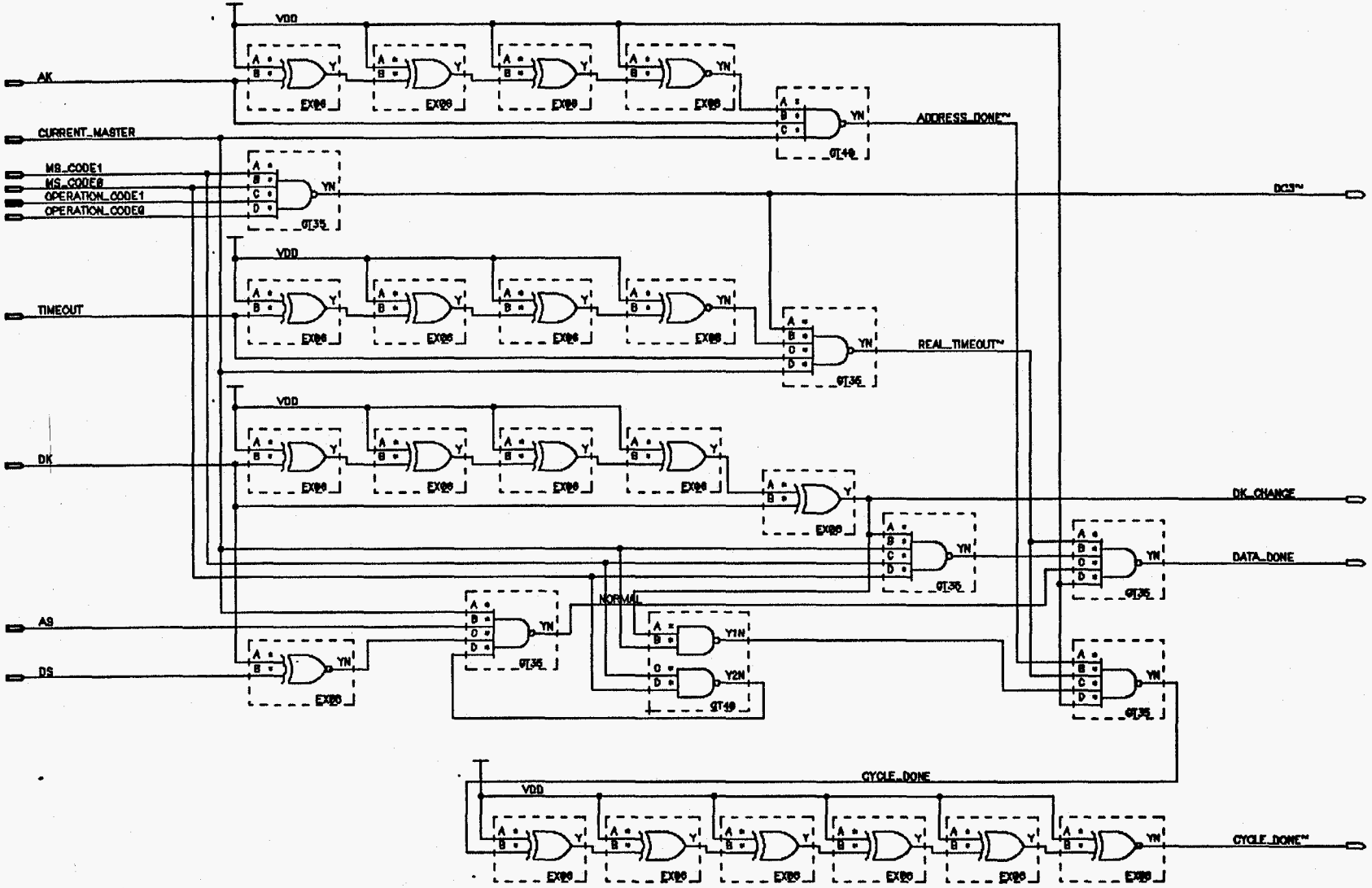

RESET_COMMAND*
RESET_MASTER*
RESET_MASTER
RESET_INTERFACE
RESET_SLAVE
INTERNAL_ADDR0
INTERNAL_ADDR1
INTERNAL_ADDR2
INTERNAL_ADDR3
INTERNAL_ADDR4
INTERNAL_ADDR5
LIST_MODE_LATCH*
BUS_WRITE_LATCH*
DATAIN22
DATAIN23
DEVICE_SEL_GATE*
DATAIN24
DATAIN26
DATAIN27
DATAIN28
DATAIN29
DATAIN30
DATAIN31
DATAIN25
AK
GEN_RESET_MASTR*
GEN_RESET_ALL*
GEN_RESET_SLAVE*
COMMAND_BUSY
POWER_UP
VDD
VSS
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 29SEP92
PAGE: 10/10
TITLE: FASTBUS RISC PROCESSOR MODULE
RESET GENERATION LOGIC
FILE NO: SSI/AV-3B0017
REV: A
DATE: 29SEP92



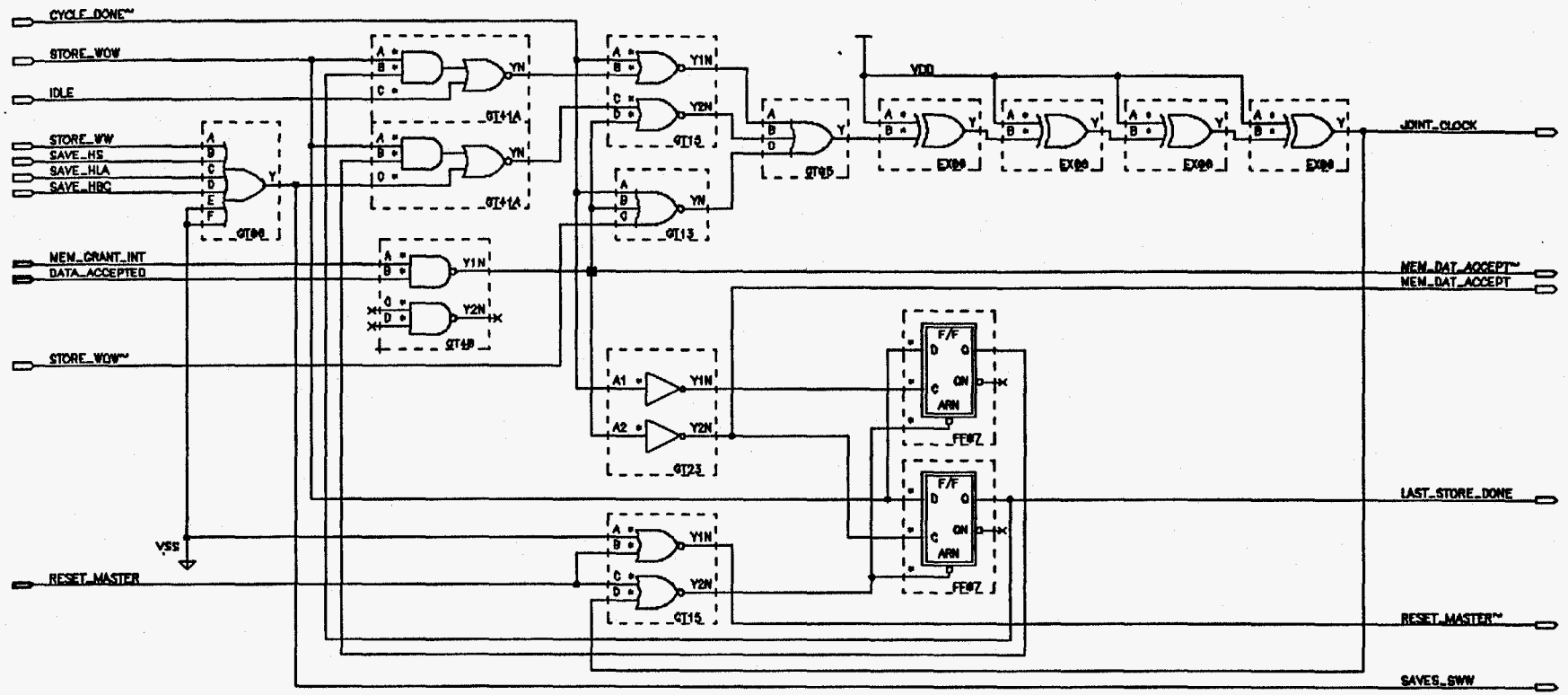
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 26SEP92	PAGE: 10/10	TITLE: FASTBUS RISC PROCESSOR MODULE CSR ACCESS CONTENTION RESOLUTION	FILE NO: SSI/JV-350018	REV: A	DATE: 26SEP92



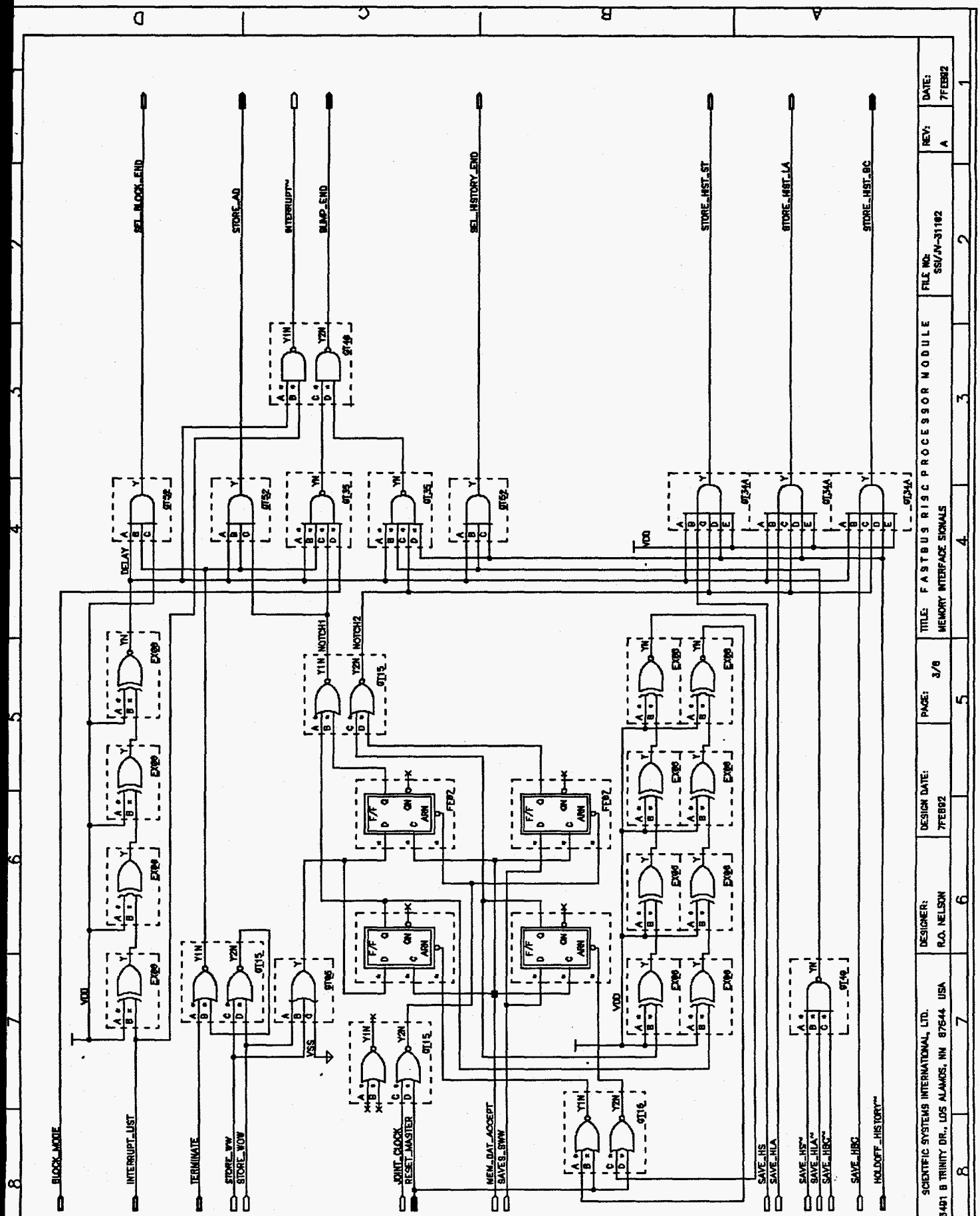
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.	DESIGNER: DRM	DESIGN DATE:	PAGE:	TITLE: FASTBUS MASTER INTERFACE	FILE NO:	REV:	DATE:
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	R.O. NELSON, PH.D.	6DEC89	1/1	BLOCK DIAGRAM FOR MASTER	SSI/JV-31666	C	1/26/91

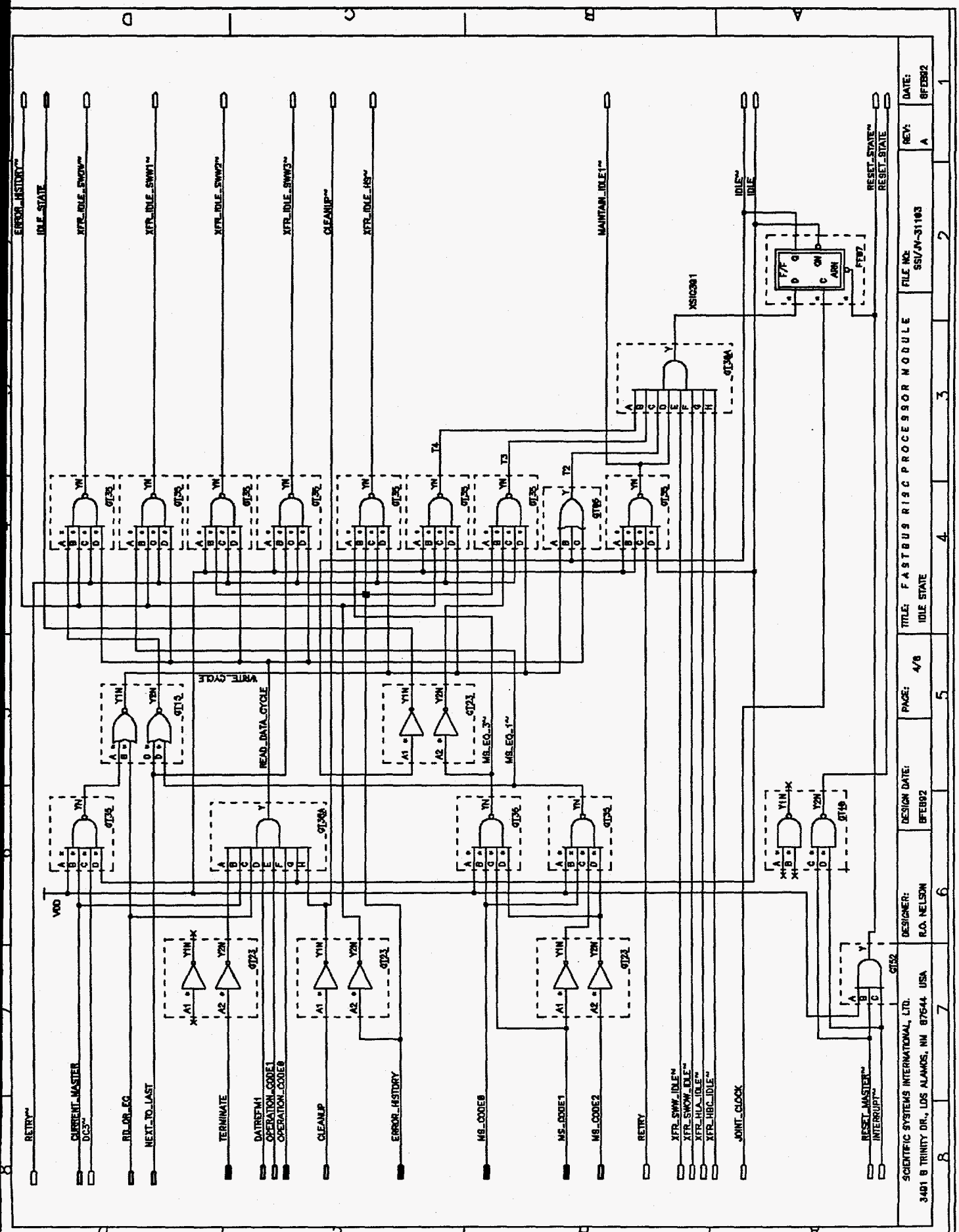


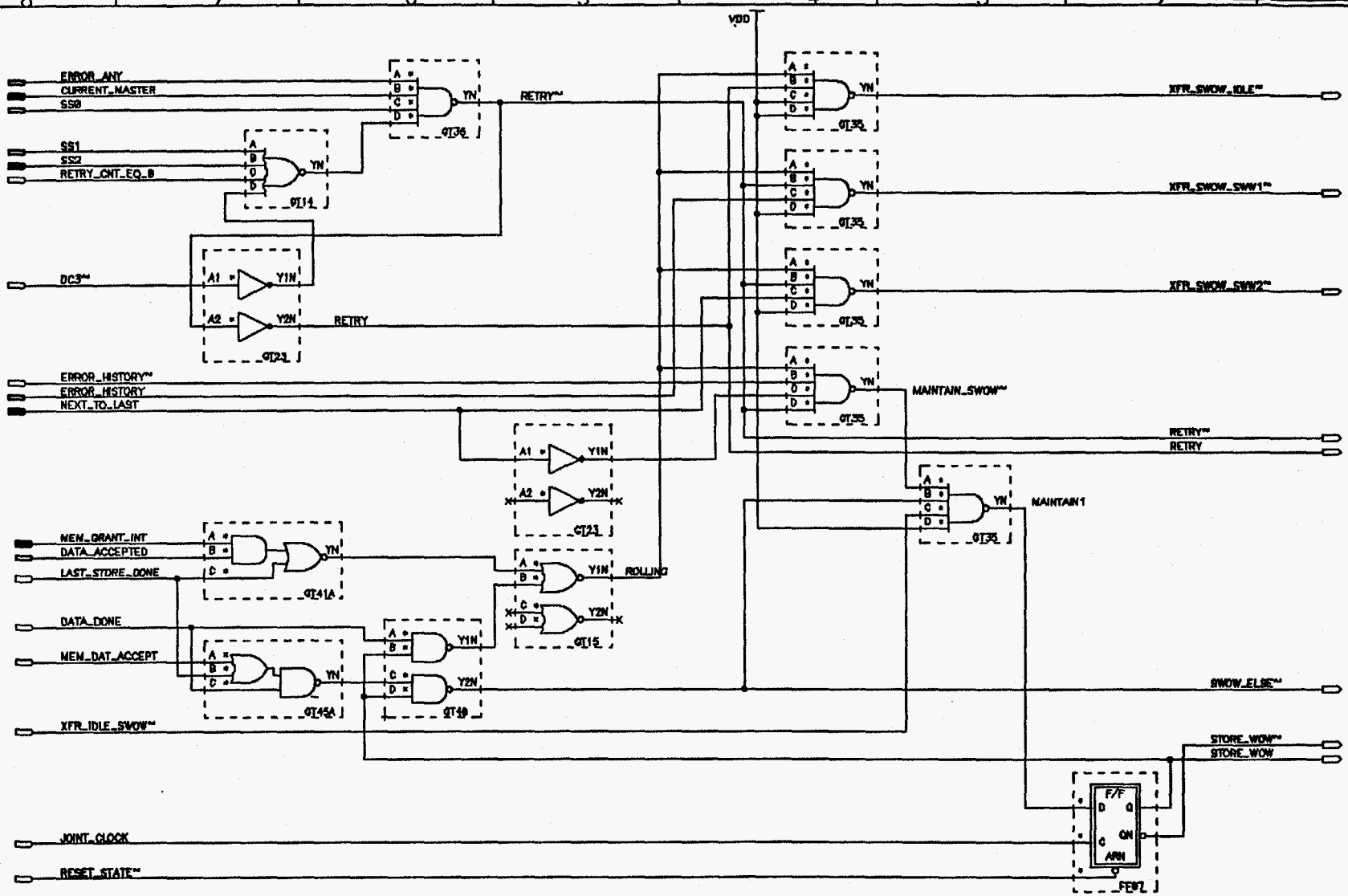
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 6FEB92	PAGE: 1/8	TITLE: FASTBUS RISC PROCESSOR MODULE TRIGGER EVENTS FOR END OF CYCLE	FILE NO: SSI/JV-31100	REV: A	DATE: 6FEB92



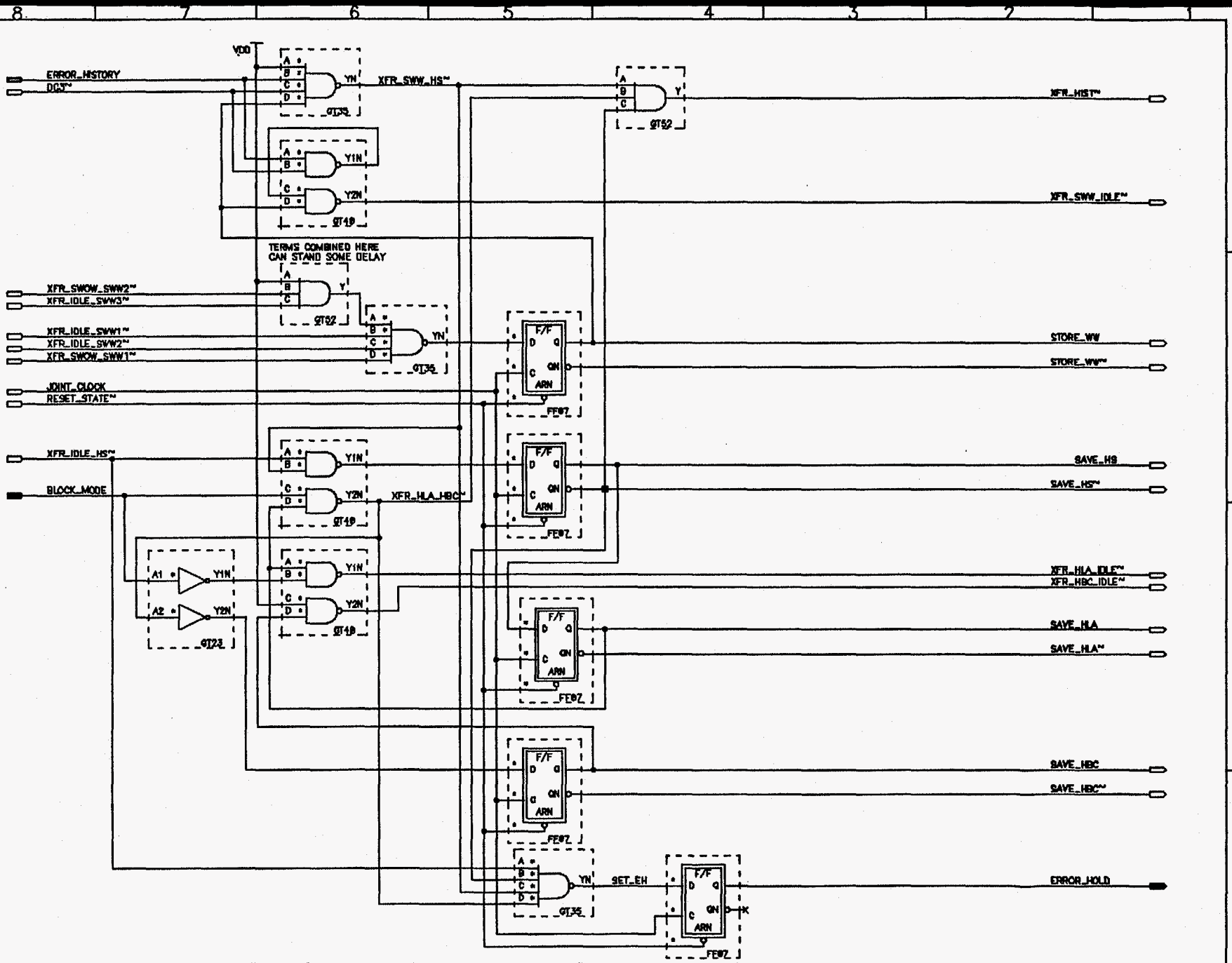
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 28MAY92	PAGE: 2/8	TITLE: FASTBUS RISC PROCESSOR MODULE CLOCK LOGIC	FILE NO: SSI/JV-31101	REV: A	DATE: 28MAY92


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3481 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 7FEB92
PAGE: 3/8
TITLE: FASTBUS RISC PROCESSOR MODULE
MEMORY INTERFACE SIGNALS
FILE NO: SSI/JN-31102
REV: A
DATE: 7FEB92
SEL_BLOCK_END
STORE_AD
INTERRUPT~
BUMP_END
SEL_HISTORY_END
STORE_HST_ST
STORE_HST_LA
STORE_HST_BC
DELAY
NOTCH1
NOTCH2
VDD
VSS
BLOCK_MODE
INTERRUPT_UST
TERMINATE
STORE_RW
STORE_WOW
JOINT_CLOCK
RESET_MASTER
MEM_DAT_ACCEPT
SAVES_SNW
SAVE_HS
SAVE_HLA
SAVE_HS~
SAVE_HLA~
SAVE_HBC~
SAVE_HBC
HOLDOFF_HISTORY~

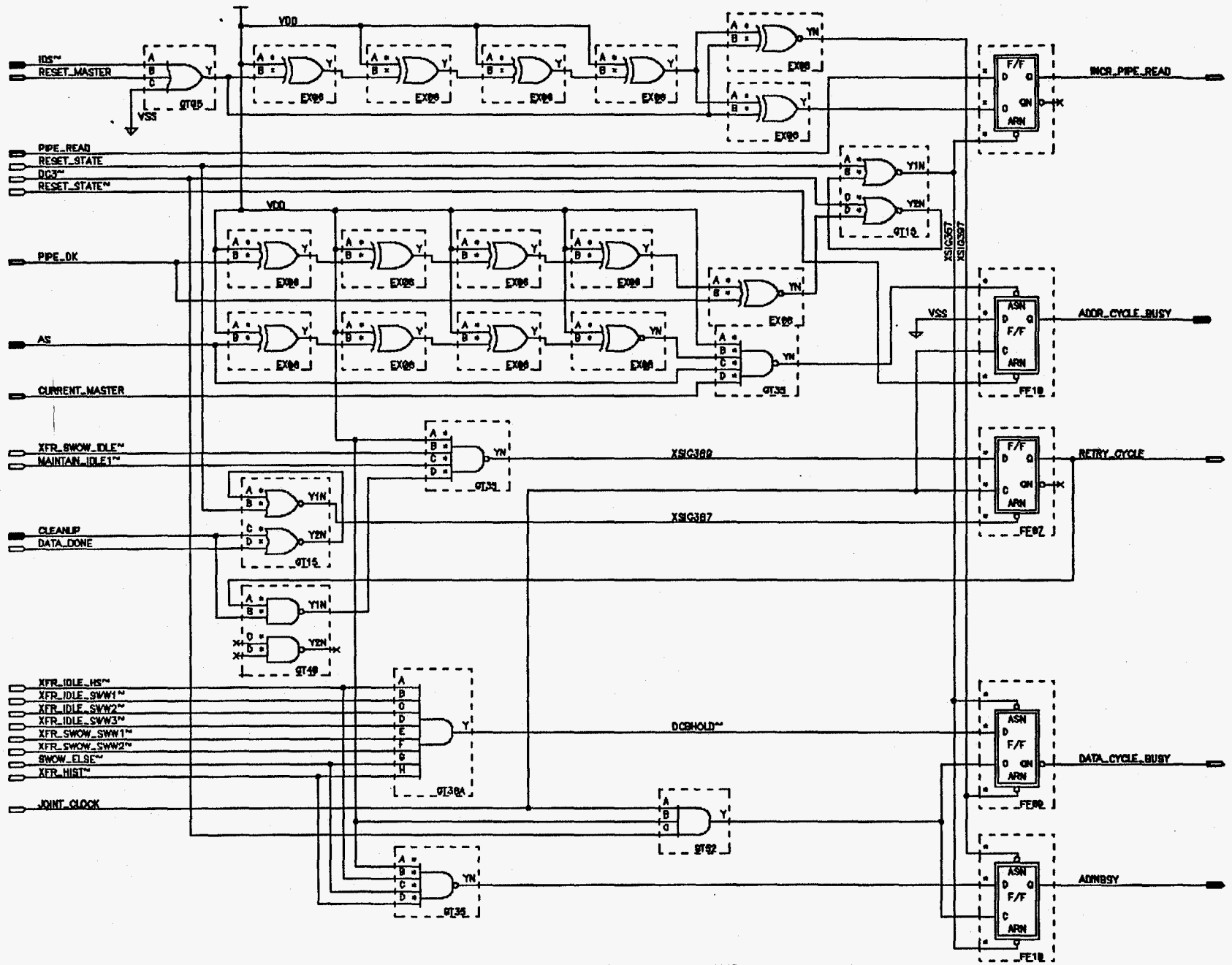

TITLE: FASTBUS RISC PROCESSOR MODULE
IDLE STATE
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 6FEB92
PAGE: 4/8
FILE NO: SSI/JN-31103
REV: A
DATE: 6FEB92
ERROR_HISTORY~
IDLE_STATE
XFR_IDLE_SWW0~
XFR_IDLE_SWW1~
XFR_IDLE_SWW2~
XFR_IDLE_SWW3~
CLEANUP~
XFR_IDLE_MS~
MAINTAIN_IDLE1~
IDLE~
RESET_STATE~
WRITE_CYCLE
READ_DATA_CYCLE
MS_EQ_3~
MS_EQ_1~
VDD
RETRY~
CURRENT_MASTER
DC3~
RD_OR_EQ
NEXT_TO_LAST
TERMINATE
DATREF.M1
OPERATION_CODE1
OPERATION_CODE0
CLEANUP
ERROR_HISTORY
MS_CODE0
MS_CODE1
MS_CODE2
RETRY
XFR_SWW_IDLE~
XFR_SWOW_IDLE~
XFR_HLA_IDLE~
XFR_HBC_IDLE~
JOINT_CLOCK
RESET_MASTER~
INTERRUPT~



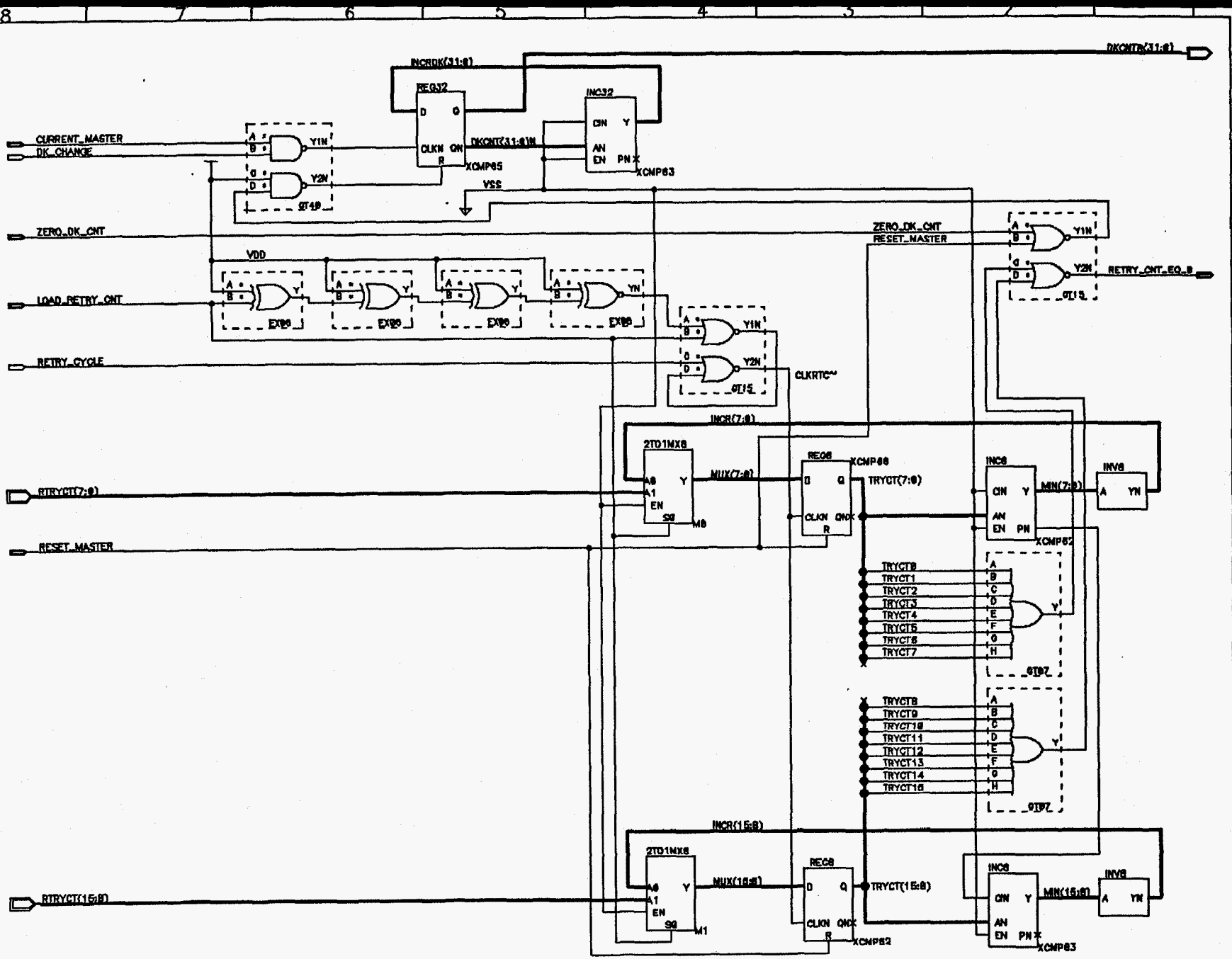
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 8FEB92	PAGE: 6/8	TITLE: FASTBUS RISC PROCESSOR MODULE STORE WITHOUT WAIT STATE	FILE NO: SSI/JV-31164	REV: A	DATE: 8FEB92



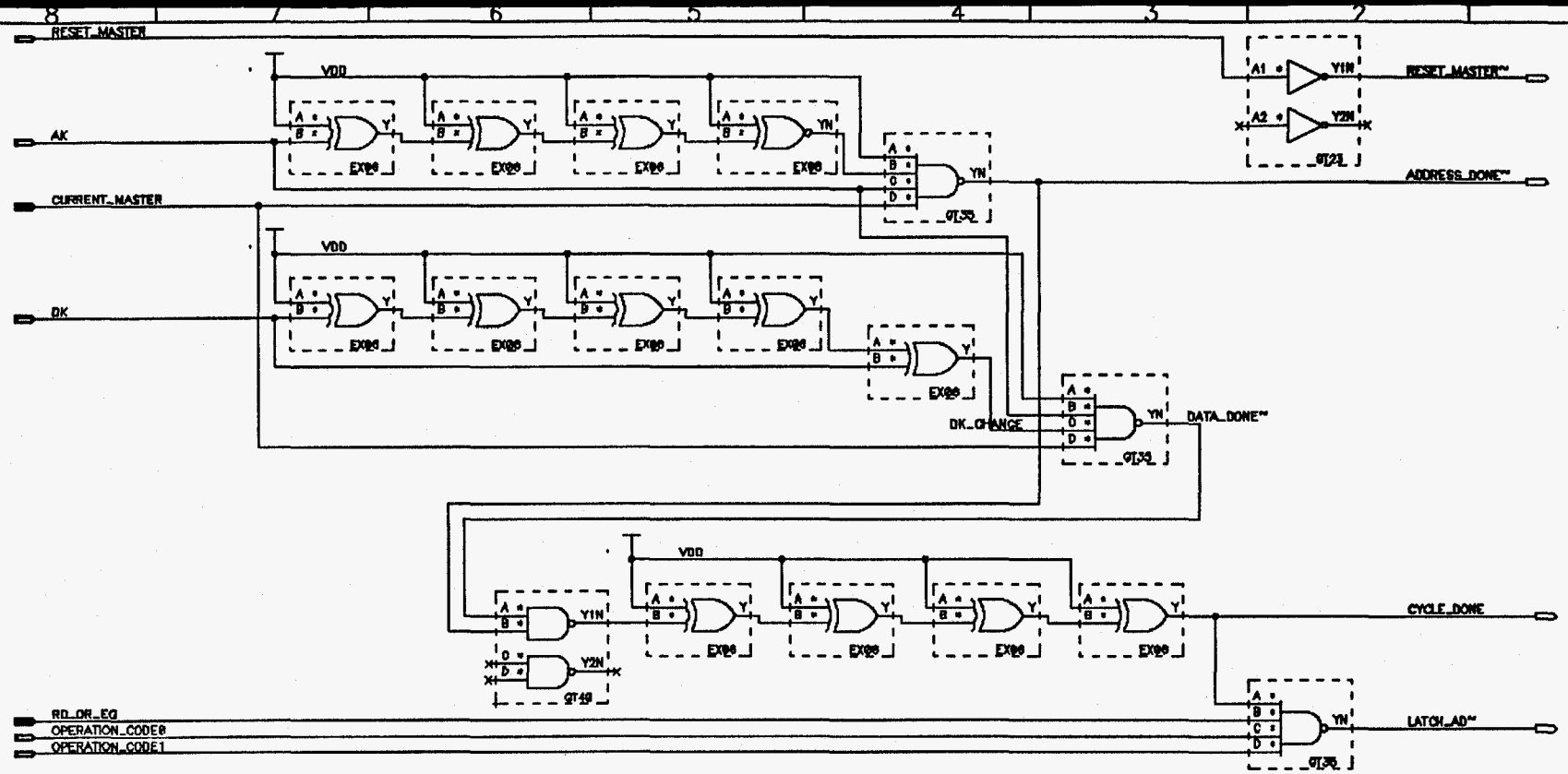
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 12FEB92	PAGE: 6/8	TITLE: FASTBUS RISC PROCESSOR MODULE STORE WITH WAIT, SAVE HISTORY STATUS, SAVE HISTORY LIST ADDRESS,, AND SAVE HISTORY BLOCK COUNT STATES	FILE NO: SSI/JV-31186	REV: A	DATE: 12FEB92



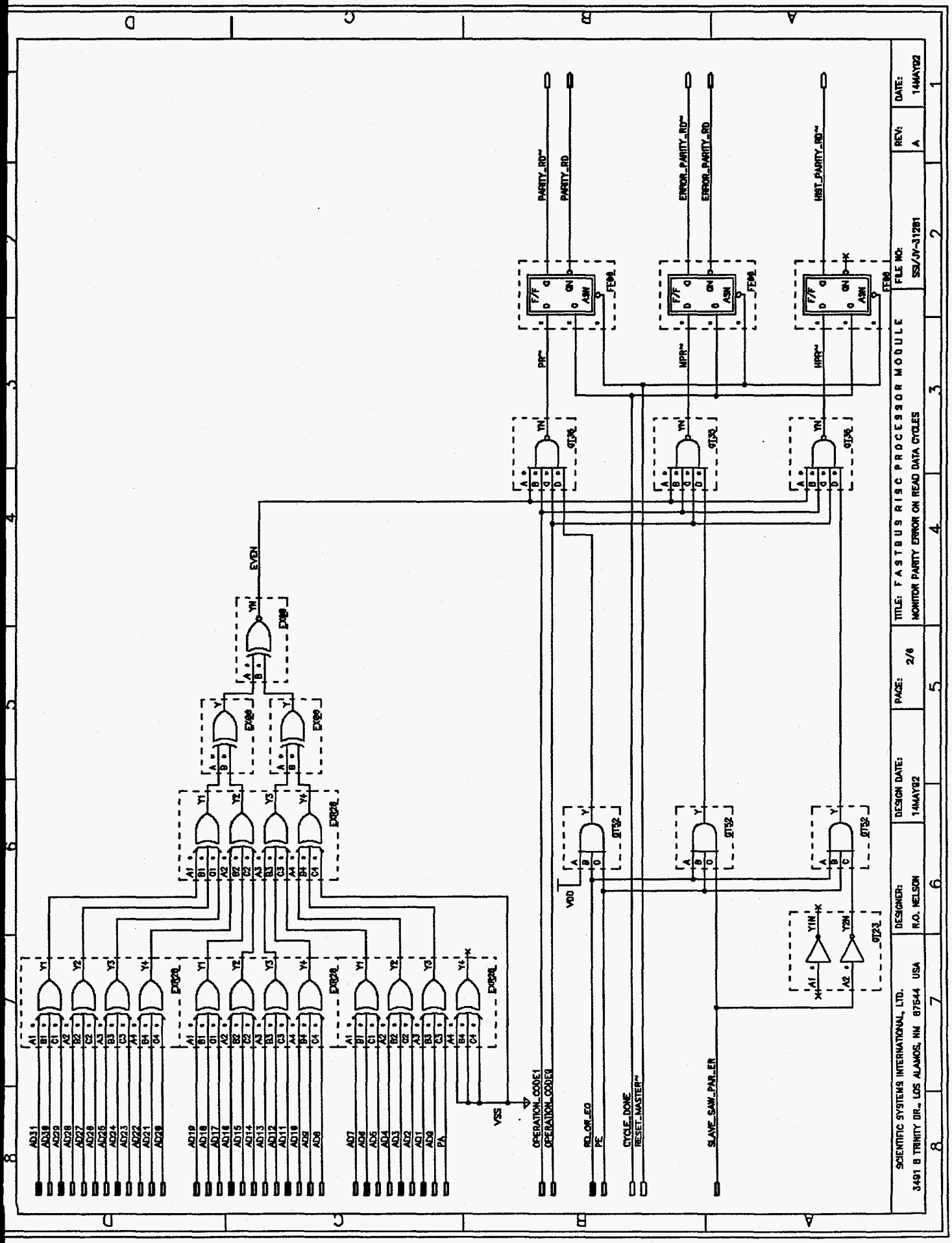
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 17FEB92	PAGE: 7/8	TITLE: FASTBUS RISC PROCESSOR MODULE STATE CONTROLLED INTERFACE SIGNALS	FILE NO: SSI/JV-31186	REV: A	DATE: 17FEB92

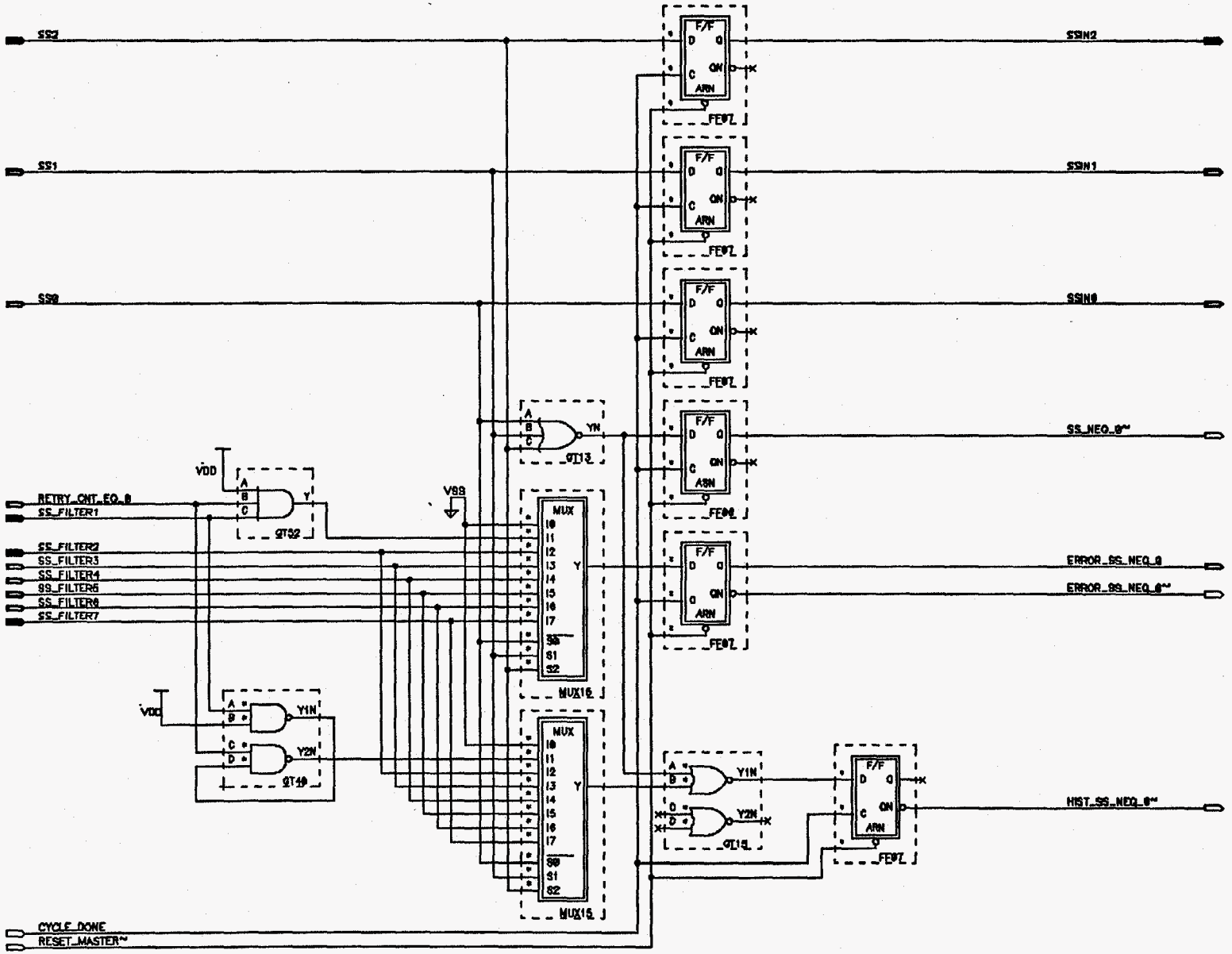


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: RON	DESIGN DATE: 2/18/92	PAGE: 8/8	TITLE: FASTBUS RISC PROCESSOR MODULE BT RETRY COUNTER		FILE NO: SSI/JV-321107	REV: B	DATE: 2/18/92

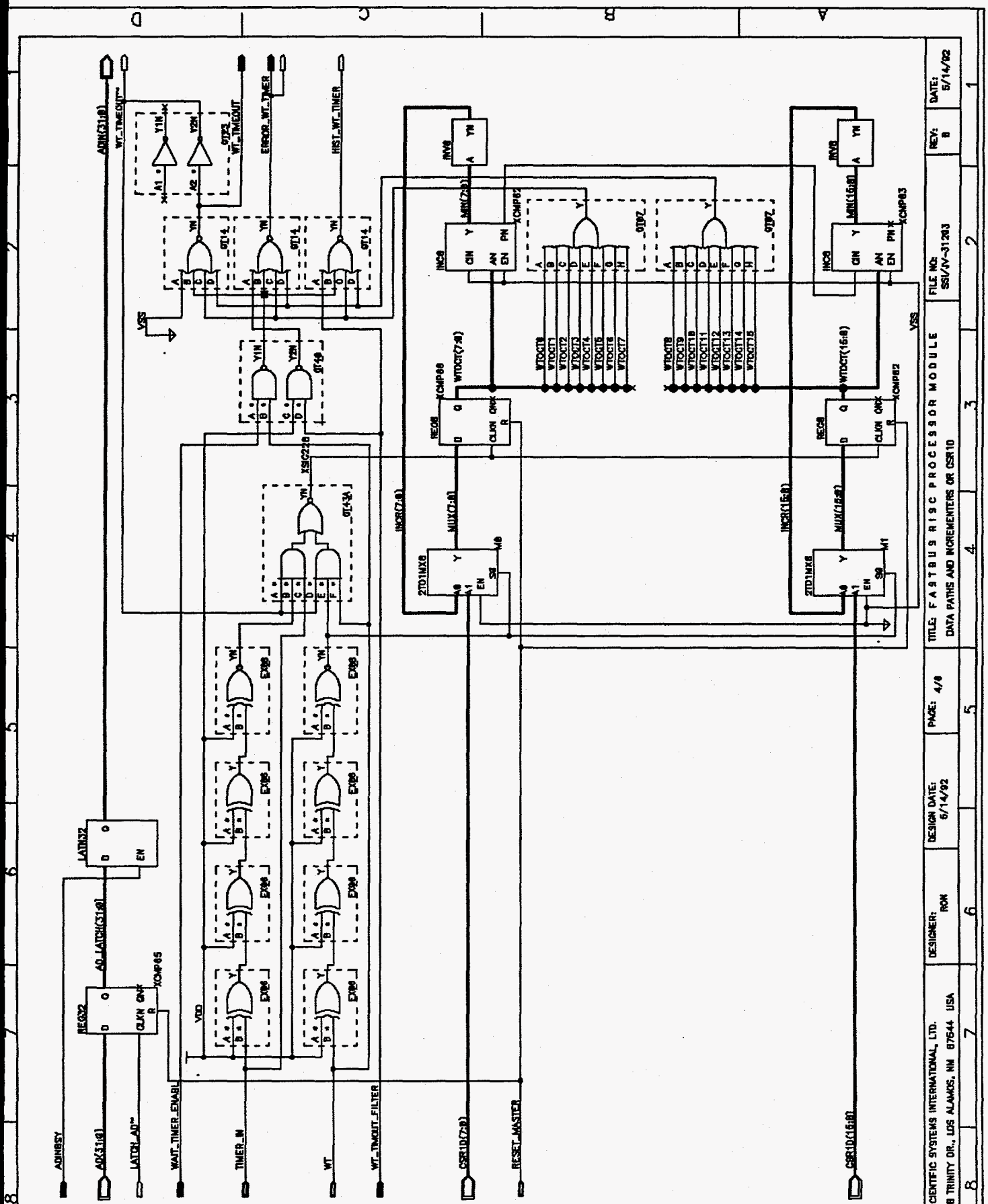


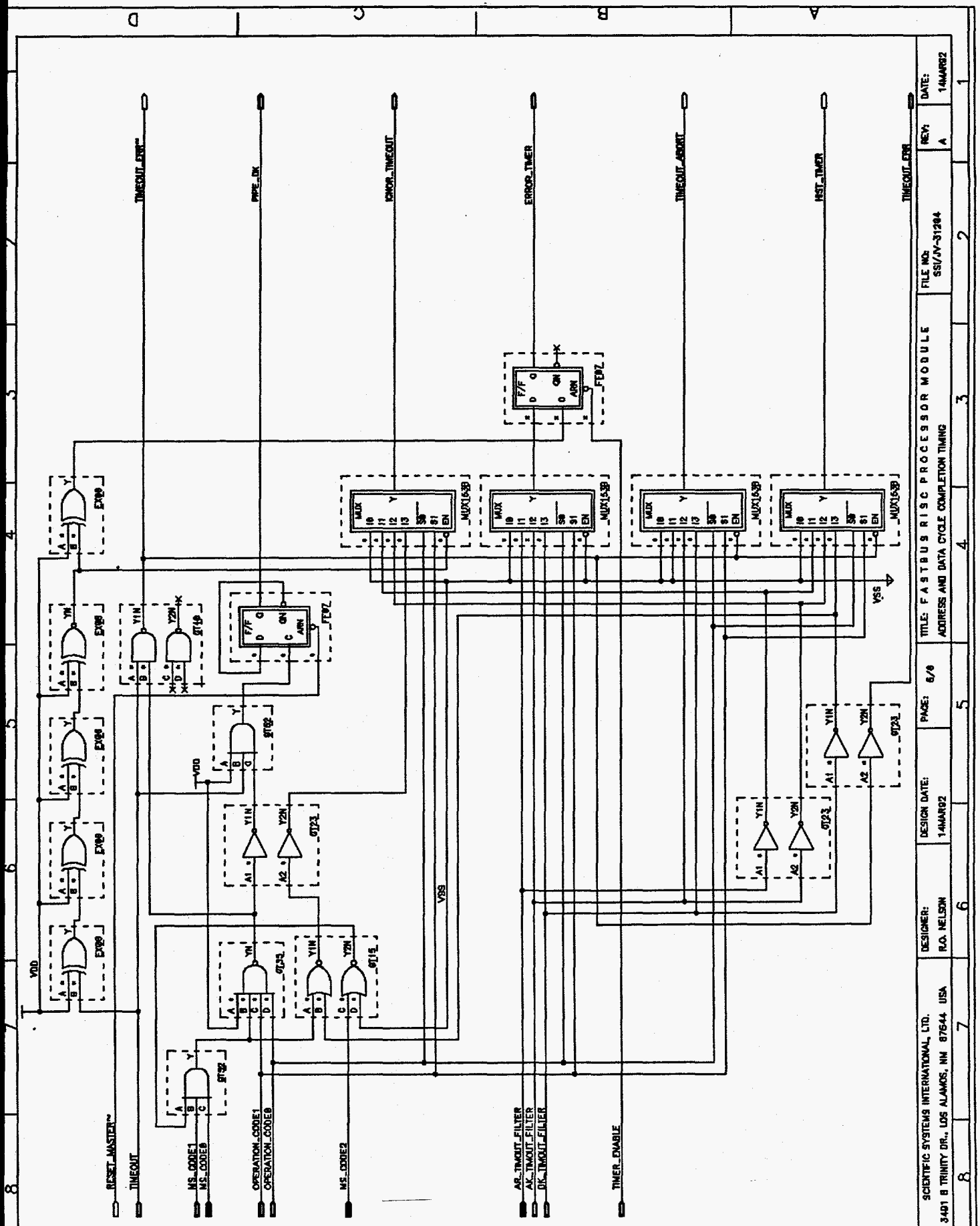
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 14MAR92	PAGE: 1/6	TITLE: FASTBUS RISC PROCESSOR MODULE ADDRESS AND DATA CYCLE COMPLETION TIMING	FILE NO: SSI/JV-31200	REV: A	DATE: 14MAR92

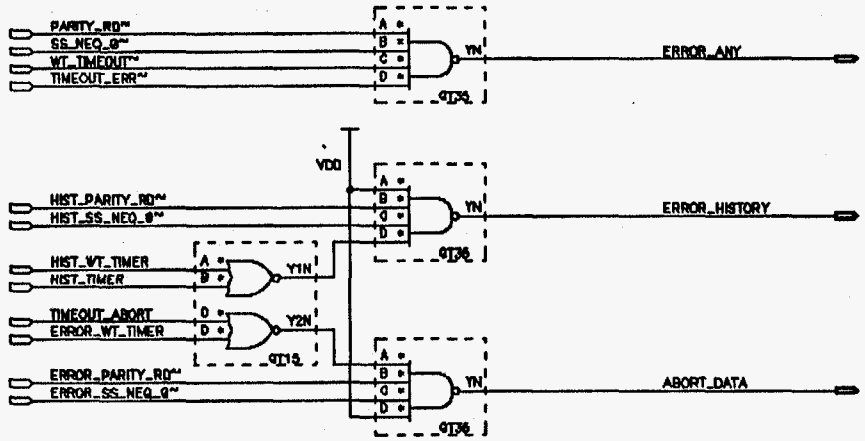

TITLE: FASTBUS RISC PROCESSOR MODULE
MONITOR PARITY ERROR ON READ DATA CYCLES
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 14MAY92
PAGE: 2/6
FILE NO: SSI/JV-31281
REV: A
DATE: 14MAY92
PARITY_RD**
PARITY_RD
ERROR_PARITY_RD**
ERROR_PARITY_RD
HST_PARITY_RD**
PPR**
MPR**
HPR**
EVEN
VDD
VSS
OPERATION_CODE1
OPERATION_CODE0
RD_OR_ED
PE
CYCLE_DONE
RESET_MASTER**
SLAVE_SAW_PAR_ER



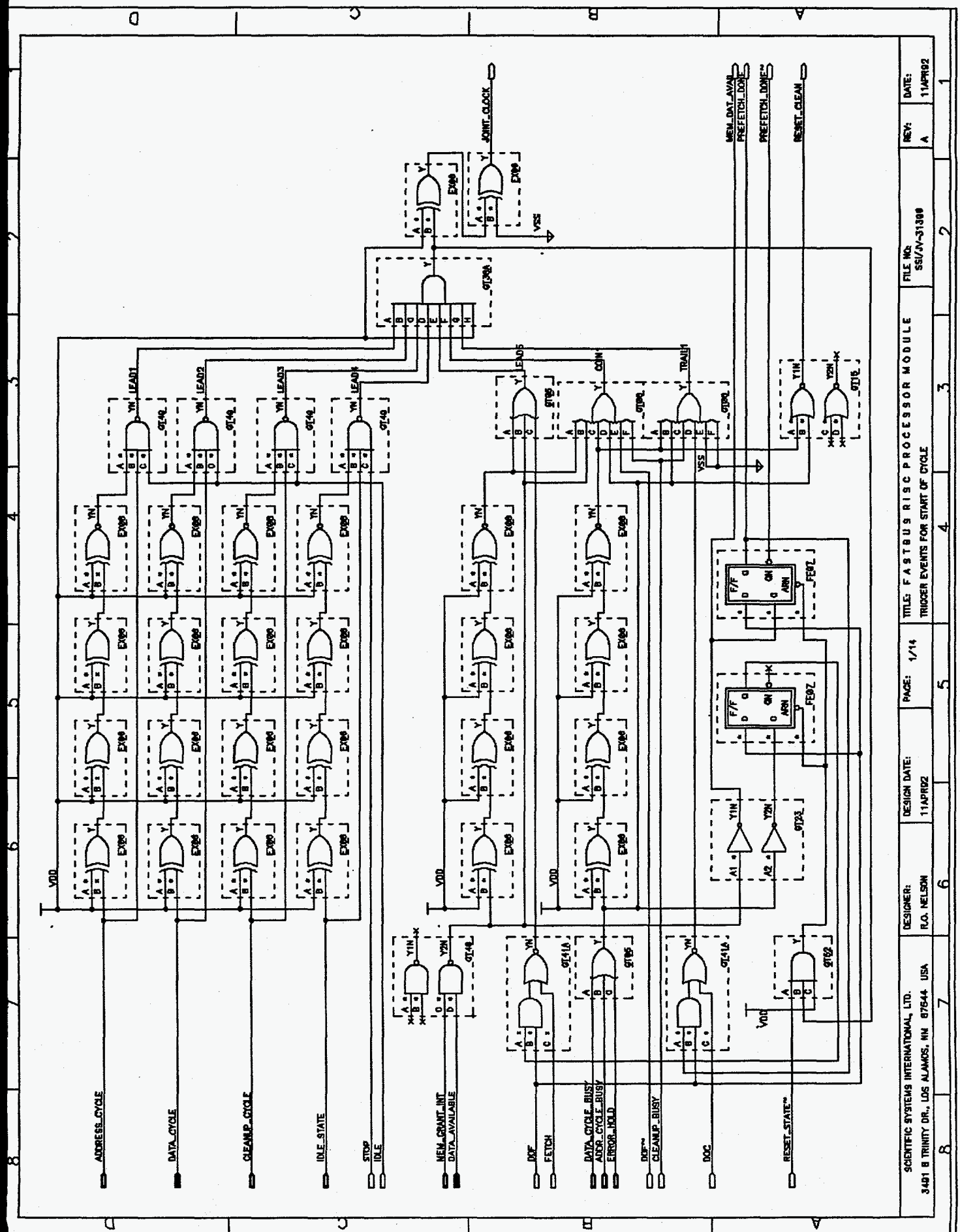
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 14MAY92	PAGE: 3/8	TITLE: FASTBUS RISC PROCESSOR MODULE MONITOR PARITY ERROR ON READ DATA CYCLES	FILE NO: SSI/JV-31282	REV: A	DATE: 14MAY92

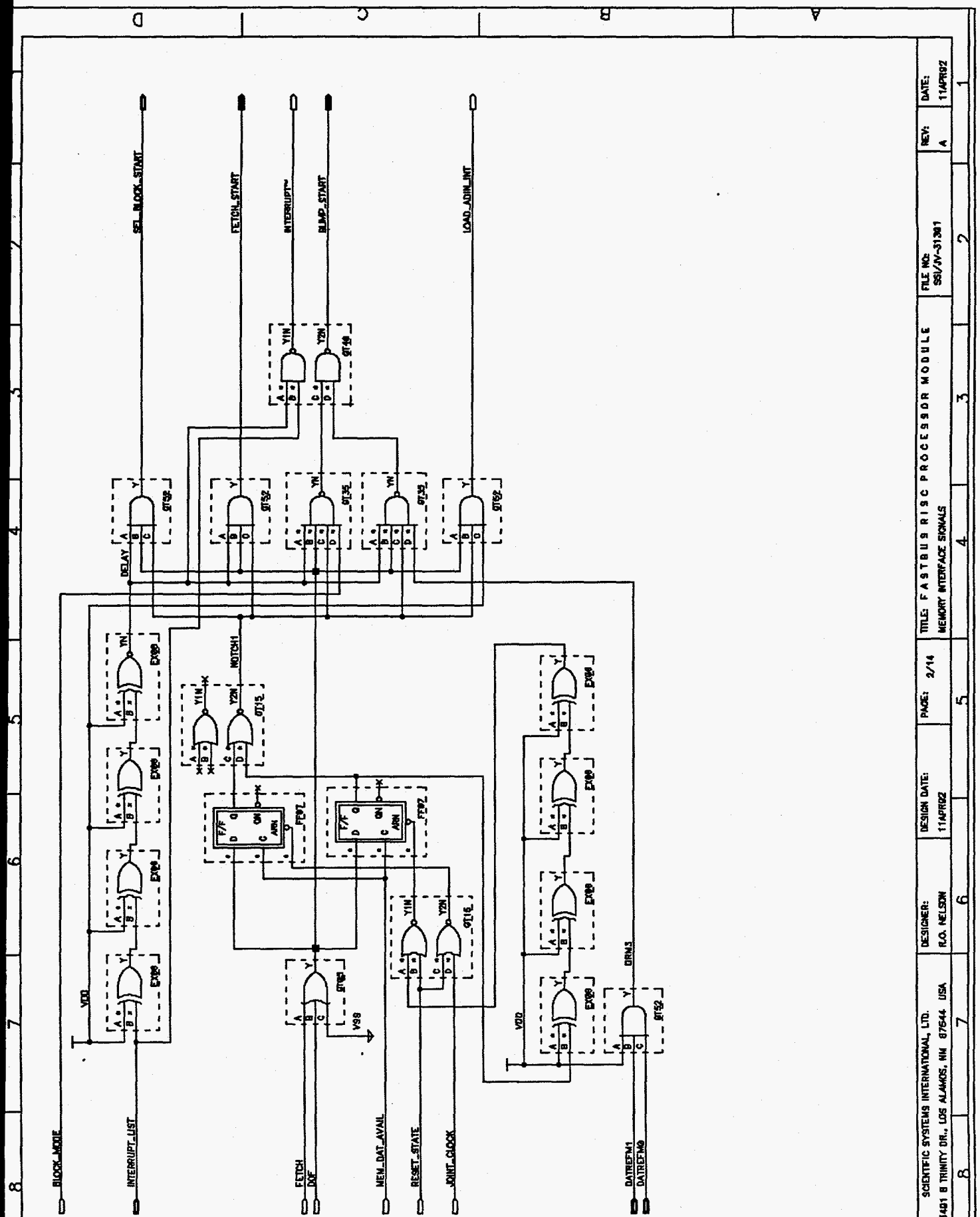

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: RON
DESIGN DATE: 5/14/92
PAGE: 4/8
TITLE: FAST BUS RISC PROCESSOR MODULE
DATA PATHS AND INCREMENTERS OR CSR10
FILE NO: SSI/JV-31203
REV: B
DATE: 5/14/92
ADIN(31:0)
WT_TIMEOUT~
ERROR_WT_TIMER
HRST_WT_TIMER
WT_TIMEOUT
VSS
VDD
WTOCT(7:0)
WTOCT(15:8)
INCR(7:0)
INCR(15:8)
MUX(7:0)
MUX(15:8)
REG32
LATCH32
AD_LATCH(31:0)
ADINBSY
AD(31:0)
LATCH_AD~
WAIT_TIMER_ENABL
TIMER_IN
WT
WT_TMOUT_FILTER
CSR1D(7:0)
RESET_MASTER
CSR1D(15:8)

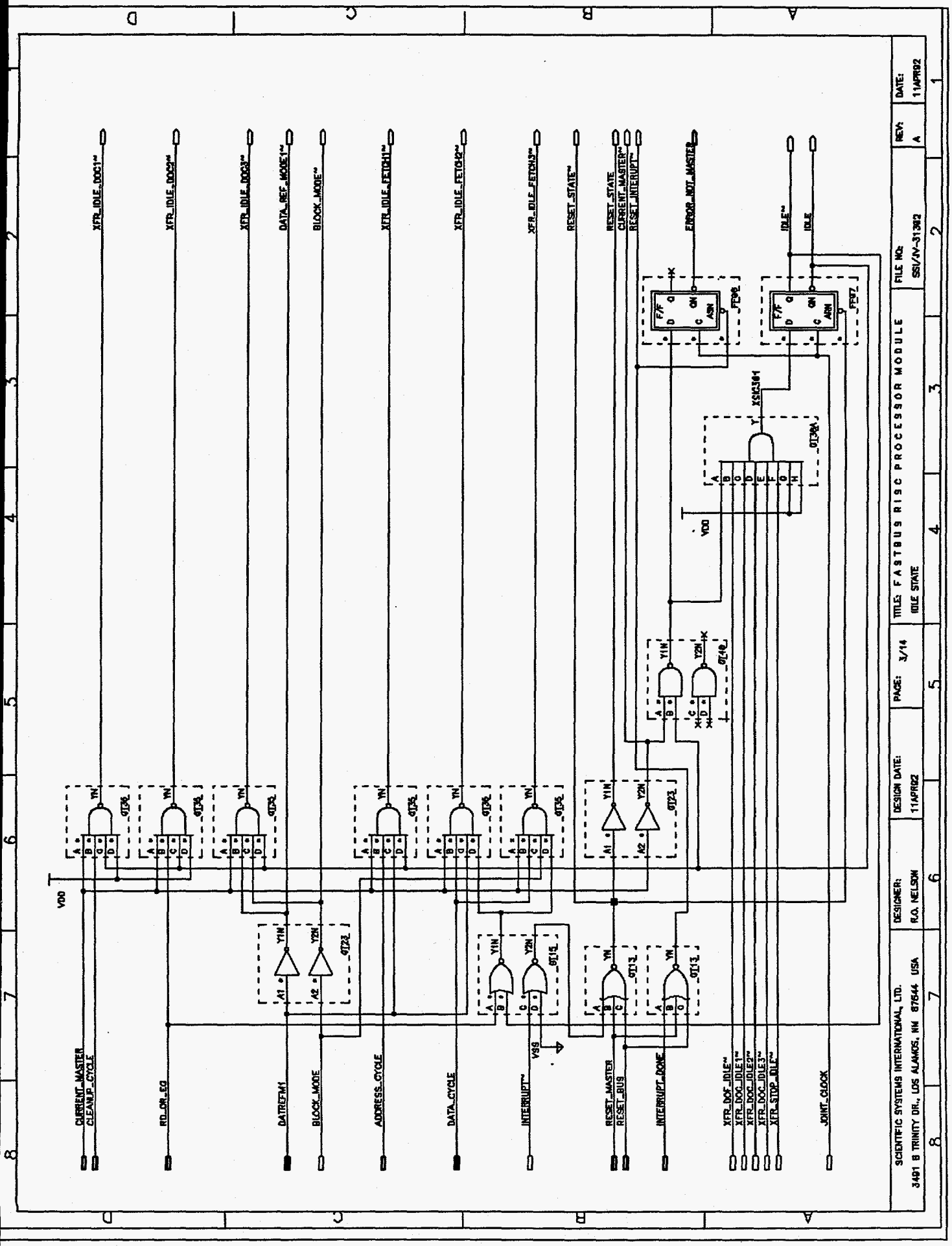

TIMEOUT_ENB*
PIPE_IN
IGNOR_TIMEOUT
ERROR_TIMER
TIMEOUT_ABORT
MST_TIMER
TIMEOUT_ENB
VDD
VSS
RESET_MASTER*
TIMEOUT
MS_CODE1
MS_CODE0
OPERATION_CODE1
OPERATION_CODE0
MS_CODE2
AR_TMOUT_FILTER
AK_TMOUT_FILTER
DK_TMOUT_FILTER
TIMER_ENABLE
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 14MAR92
PAGE: 6/6
TITLE: FASTBUS RISC PROCESSOR MODULE
ADDRESS AND DATA CYCLE COMPLETION TIMING
FILE NO: SSI/JV-31284
REV: A
DATE: 14MAR92

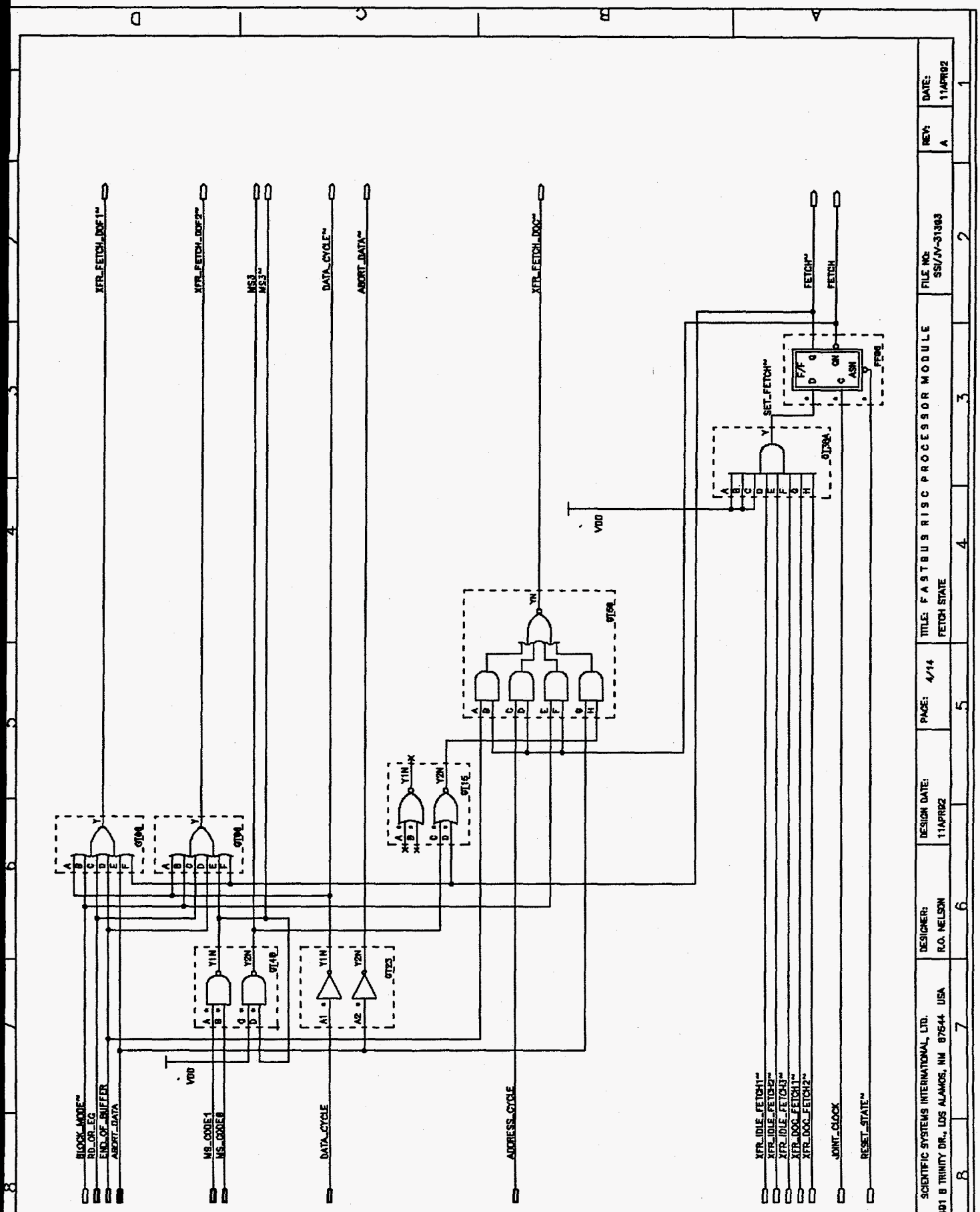


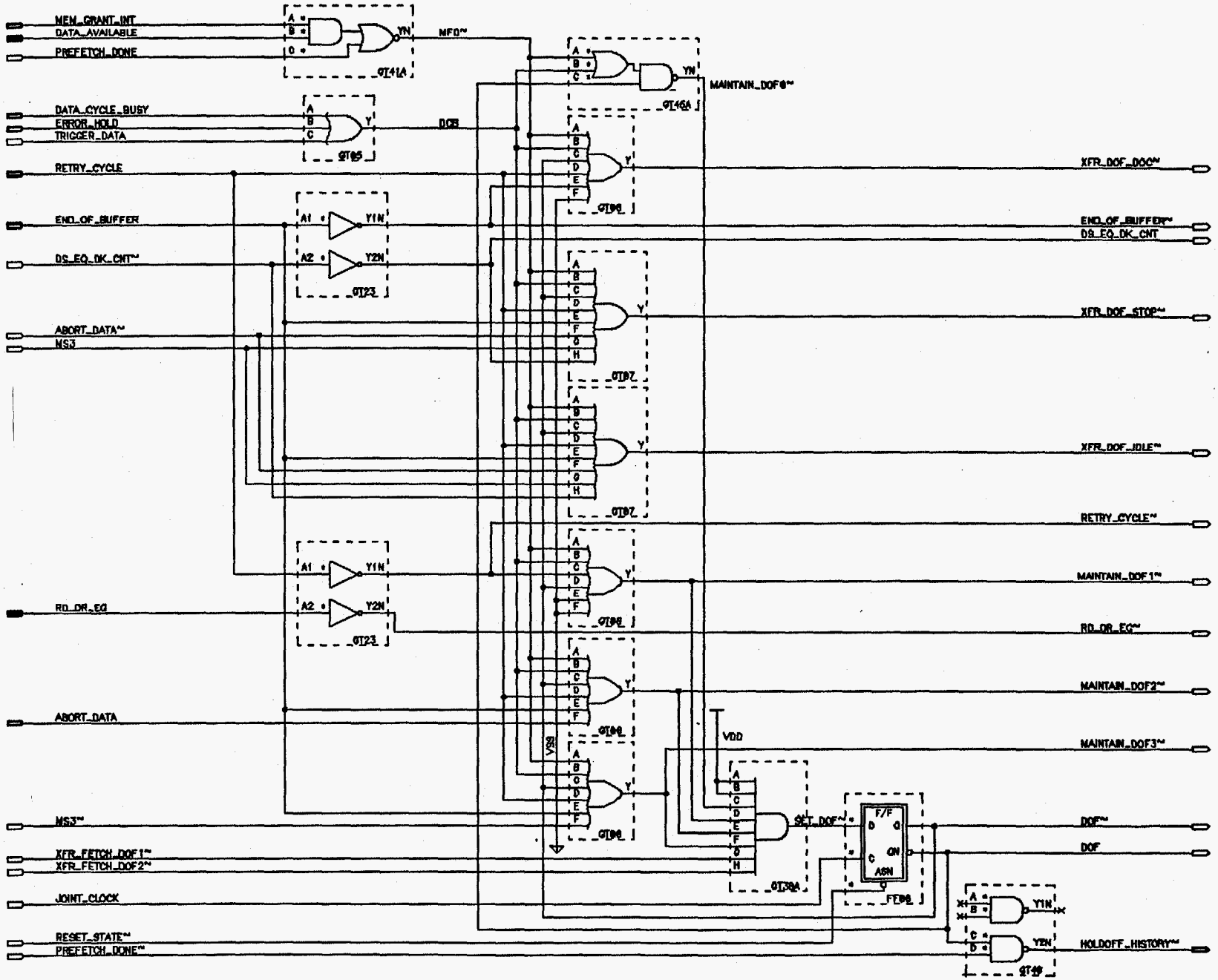
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: RON	DESIGN DATE: 6/14/92	PAGE: 6/6	TITLE: FASTBUS RISC PROCESSOR MODULE MISC STATUS	FILE NO: SSI/JV-31286	REV: A	DATE: 6/14/92


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 11APR92
PAGE: 1/14
TITLE: FASTBUS RISC PROCESSOR MODULE
TRIGGER EVENTS FOR START OF CYCLE
FILE NO: SSI/JV-31300
REV: A
DATE: 11APR92
JOINT_CLOCK
MEM_DAT_AVAL
PREFETCH_DONE
PREFETCH_DONE~
RESET_CLEAN
ADDRESS_CYCLE
DATA_CYCLE
CLEANUP_CYCLE
IDLE_STATE
STOP
IDLE
MEM_GRANT_INT
DATA_AVAILABLE
DOF
FETCH
DATA_CYCLE_BUSY
ADDR_CYCLE_BUSY
ERROR_HOLD
DOF~
CLEANUP_BUSY
DOC
RESET_STATE~
VDD
VSS

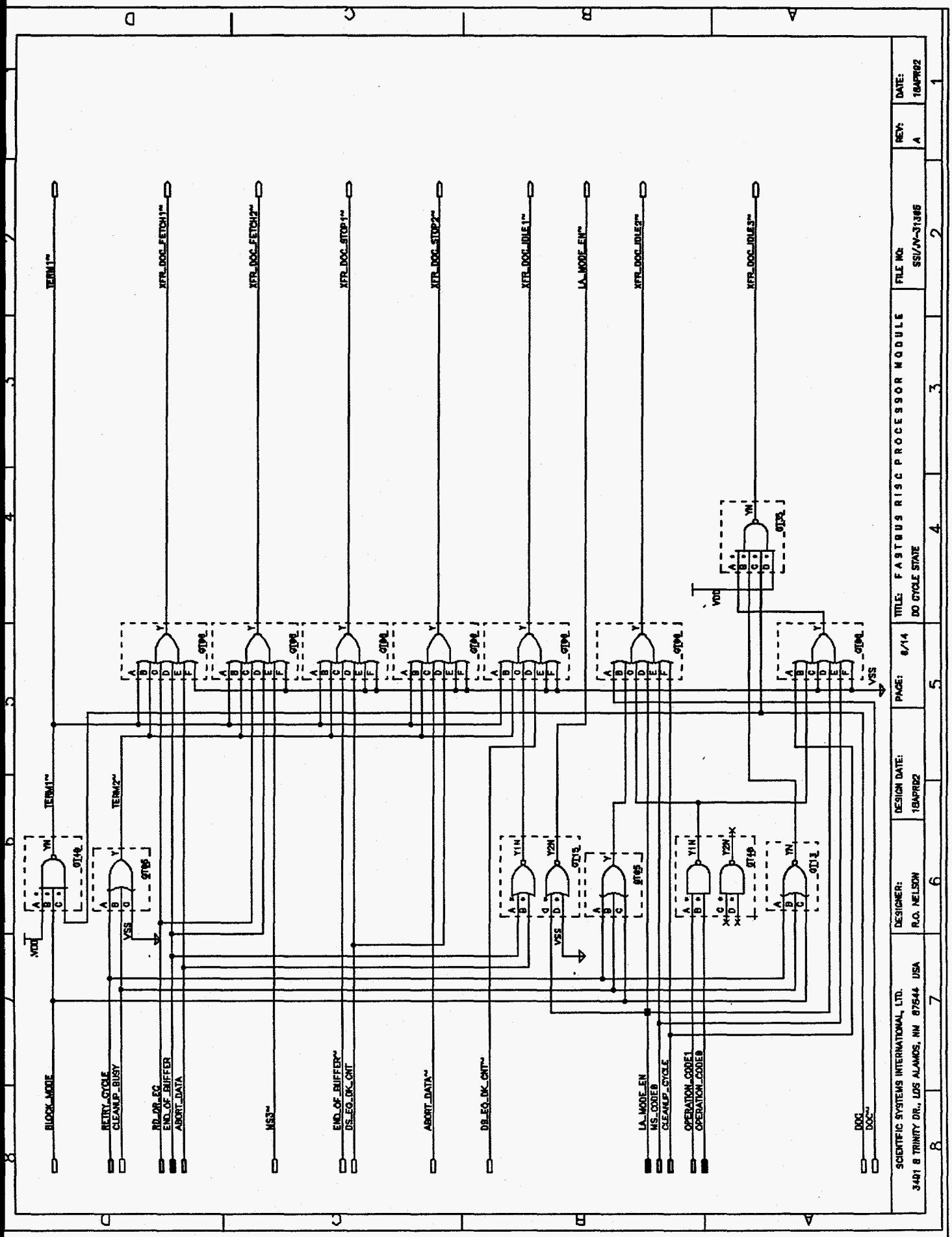

SEL_BLOCK_START
FETCH_START
INTERRUPT~
BUMP_START
LOAD_ADDR_INT
DELAY
NOTCH1
VDD
BLOCK_MODE
INTERRUPT_LIST
FETCH
DAF
MEM_DAT_AVAIL
RESET_STATE
JOINT_CLOCK
DATREFM1
DATREFM0
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 11APR92
PAGE: 2/14
TITLE: FASTBUS RISC PROCESSOR MODULE
MEMORY INTERFACE SIGNALS
FILE NO: SSI/JV-31301
REV: A
DATE: 11APR92

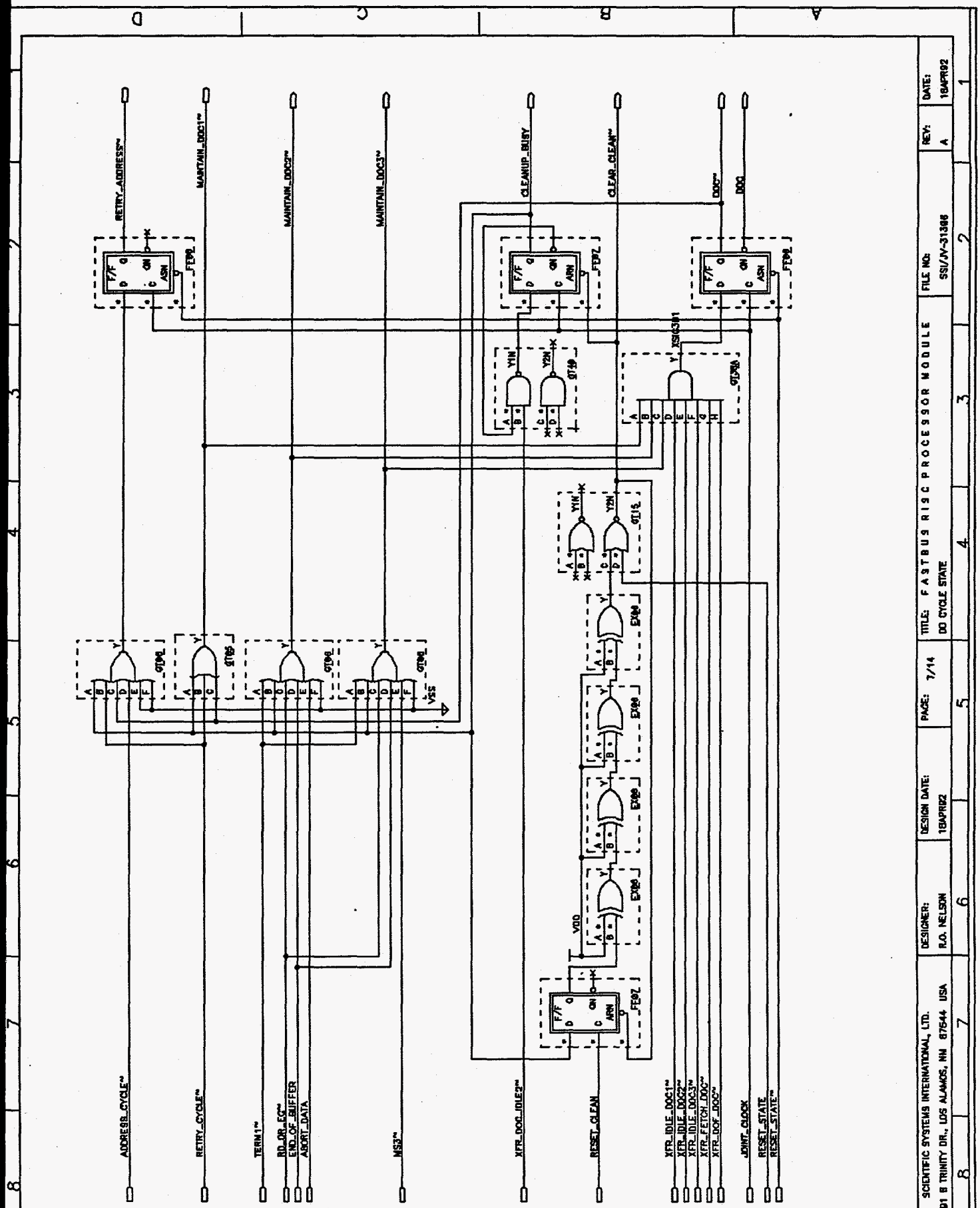

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 11APR92
PAGE: 3/14
TITLE: FASTBUS RISC PROCESSOR MODULE
IDLE STATE
FILE NO: SSI/JV-31382
REV: A
DATE: 11APR92
XFR_IDLE_DOC1~
XFR_IDLE_DOC2~
XFR_IDLE_DOC3~
DATA_REF_MODE1~
BLOCK_MODE~
XFR_IDLE_FETCH1~
XFR_IDLE_FETCH2~
XFR_IDLE_FETCH3~
RESET_STATE~
RESET_STATE
CURRENT_MASTER~
RESET_INTERUPT~
ERROR_NOT_MASTER
IDLE~
IDLE
VDD
CURRENT_MASTER
CLEANUP_CYCLE
RD_OR_EQ
DATREFM1
BLOCK_MODE
ADDRESS_CYCLE
DATA_CYCLE
INTERRUPT~
VSS
RESET_MASTER
RESET_BUS
INTERRUPT_DONE
XFR_DOC_IDLE~
XFR_DOC_IDLE1~
XFR_DOC_IDLE2~
XFR_DOC_IDLE3~
XFR_STOP_IDLE~
JOINT_CLOCK

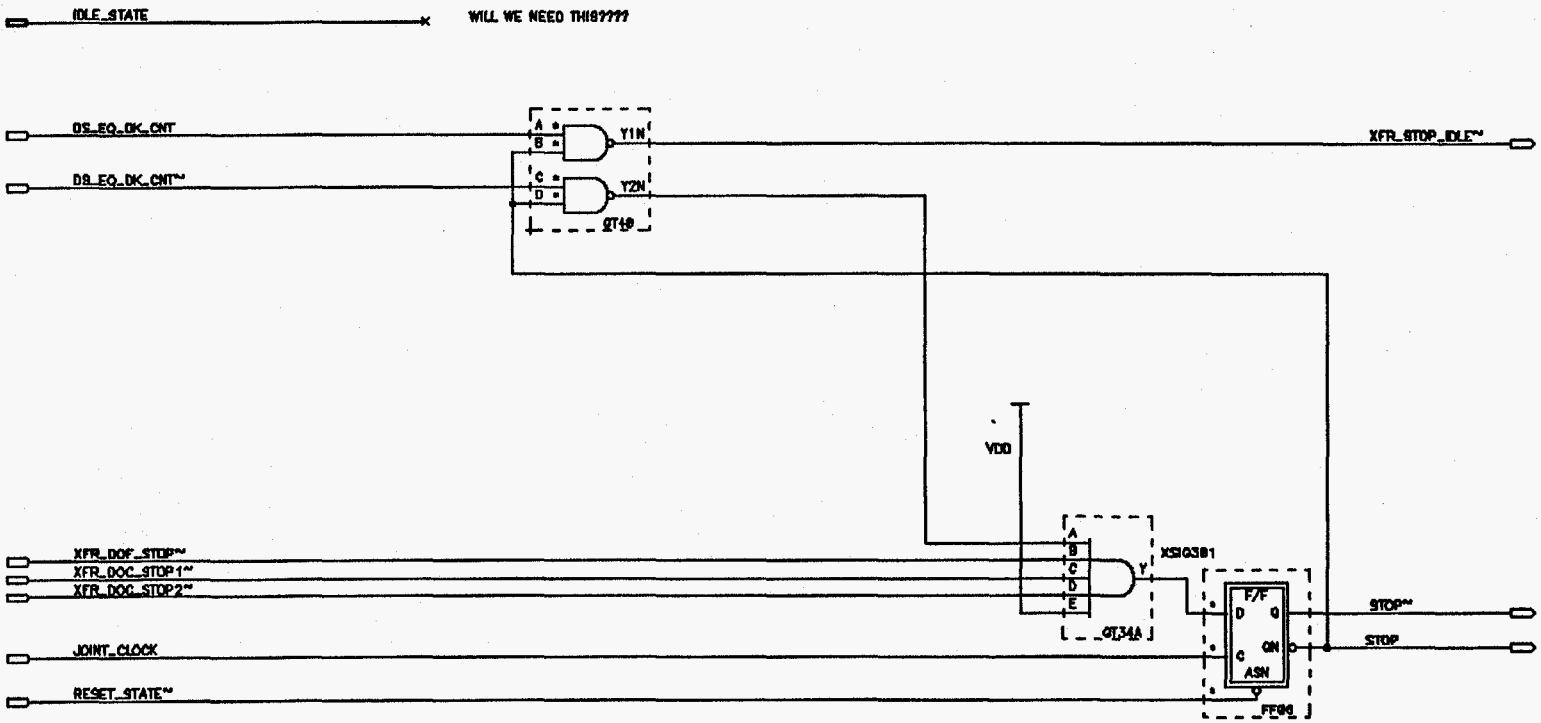

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 11APR92
PAGE: 4/14
TITLE: FASTBUS RISC PROCESSOR MODULE
FETCH STATE
FILE NO: SSI/JV-31383
REV: A
DATE: 11APR92
XFR_FETCH_DOF1*
XFR_FETCH_DOF2*
MS3
MS3*
DATA_CYCLE*
ABORT_DATA*
XFR_FETCH_DOC*
FETCH*
FETCH
SET_FETCH*
VDD
BLOCK_MODE*
RD_OR_EC
END_OF_BUFFER
ABORT_DATA
MS_CODE1
MS_CODE0
DATA_CYCLE
ADDRESS_CYCLE
XFR_IDLE_FETCH1*
XFR_IDLE_FETCH2*
XFR_IDLE_FETCH3*
XFR_DOC_FETCH1*
XFR_DOC_FETCH2*
JOINT_CLOCK
RESET_STATE*



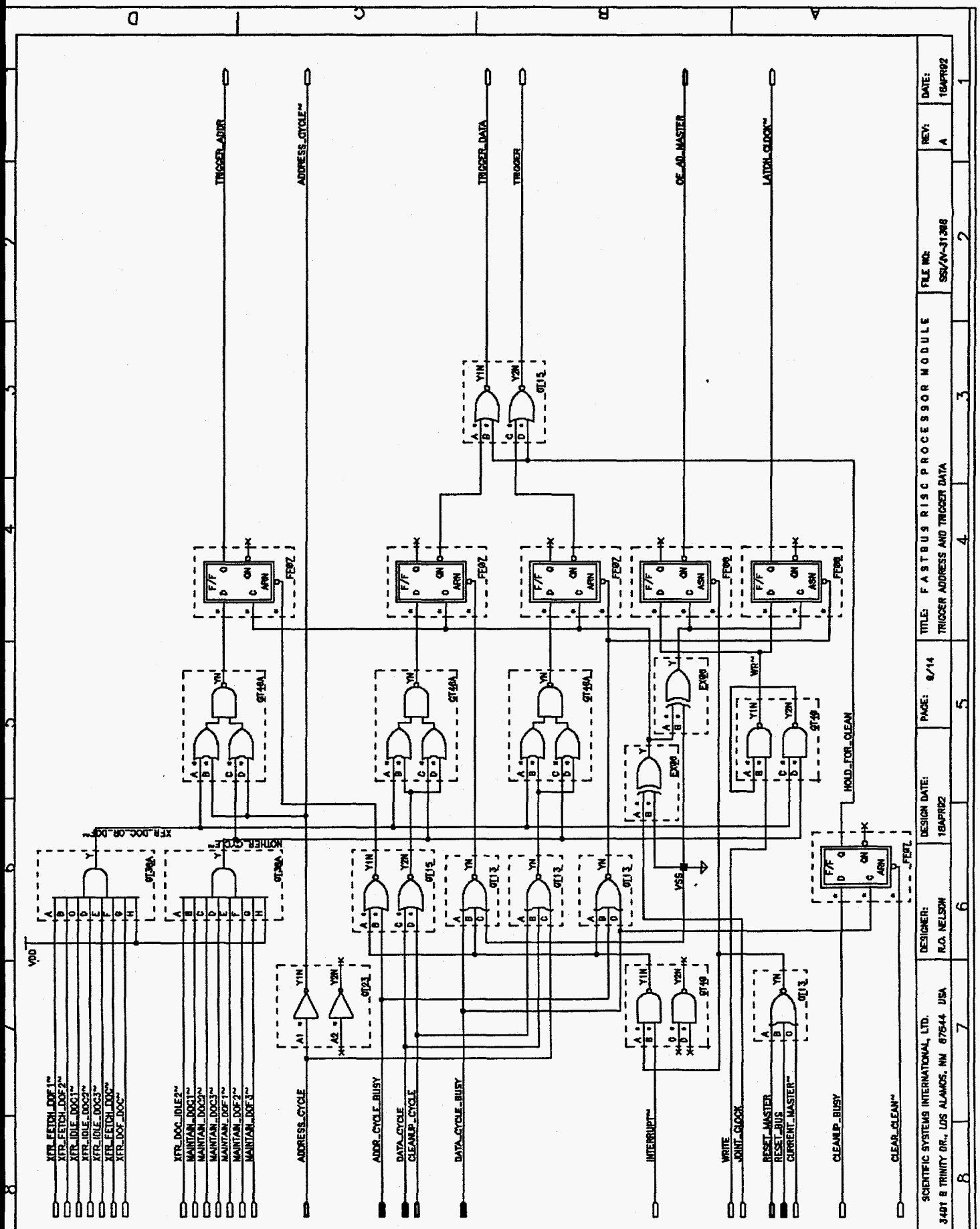
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 18APR92	PAGE: 6/14	TITLE: FASTBUS RISC PROCESSOR MODULE DO AND FETCH STATE	FILE NO: SSI/JV-31384	REV: A	DATE: 18APR92

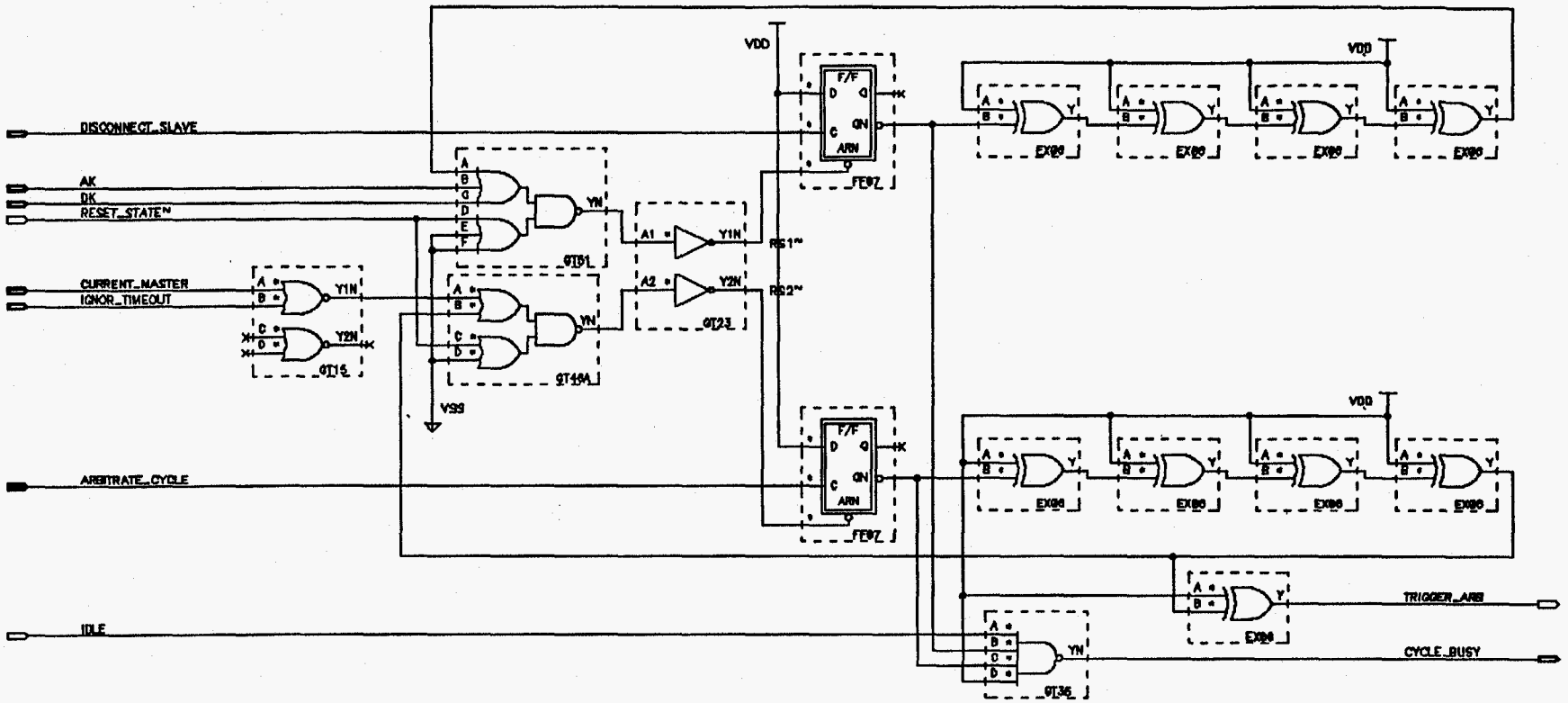

TERM1~
XFR_DOC_FETCH1~
XFR_DOC_FETCH2~
XFR_DOC_STOP1~
XFR_DOC_STOP2~
XFR_DOC_IDLE1~
LA_MODE_EN~
XFR_DOC_IDLE2~
XFR_DOC_IDLE3~
VDD
VSS
TERM1~
TERM2~
BLOCK_MODE
RETRY_CYCLE
CLEANUP_BUSY
RD_OR_EC
END_OF_BUFFER
ABORT_DATA
MSG~
END_OF_BUFFER~
DS_EQ_DK_CNT
ABORT_DATA~
DB_EQ_DK_CNT~
LA_MODE_EN
MS_CODE0
CLEANUP_CYCLE
OPERATION_CODE1
OPERATION_CODE0
DOC
DOC~
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 18APR02
PAGE: 6/14
TITLE: FASTBUS RISC PROCESSOR MODULE
DO CYCLE STATE
FILE NO: SSI/JV-31385
REV: A
DATE: 18APR02


RETRY_ADDRESS™
MAINTAIN_DOC1™
MAINTAIN_DOC2™
MAINTAIN_DOC3™
CLEANUP_BUSY
CLEAR_CLEAN™
DOC™
DOC
ADDRESS_CYCLE™
RETRY_CYCLE™
TERM1™
RD_OR_EC™
END_OF_BUFFER
ABORT_DATA
MS3™
XFR_DOC_IDLE3™
RESET_CLEAN
XFR_IDLE_DOC1™
XFR_IDLE_DOC2™
XFR_IDLE_DOC3™
XFR_FETCH_DOC™
XFR_DOF_DOC™
JOINT_CLOCK
RESET_STATE
RESET_STATE™
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 18APR92
PAGE: 7/14
TITLE: FASTBUS RISC PROCESSOR MODULE
DO CYCLE STATE
FILE NO: SSI/JV-31388
REV: A
DATE: 18APR92

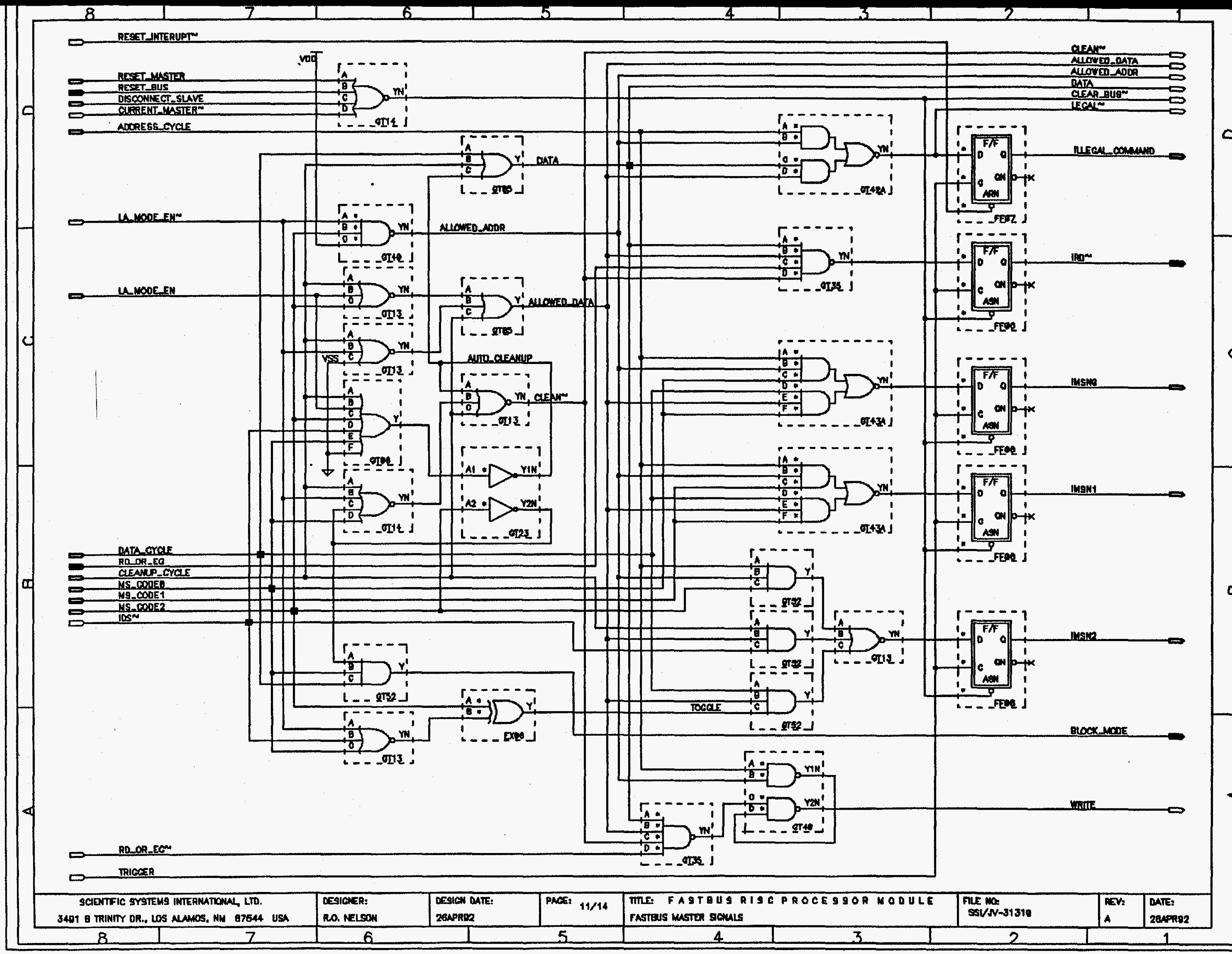


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 18APR92	PAGE: 8/14	TITLE: FASTBUS RISC PROCESSOR MODULE STOP STATE		FILE NO: SSU/JV-31387	REV: A	DATE: 18APR92

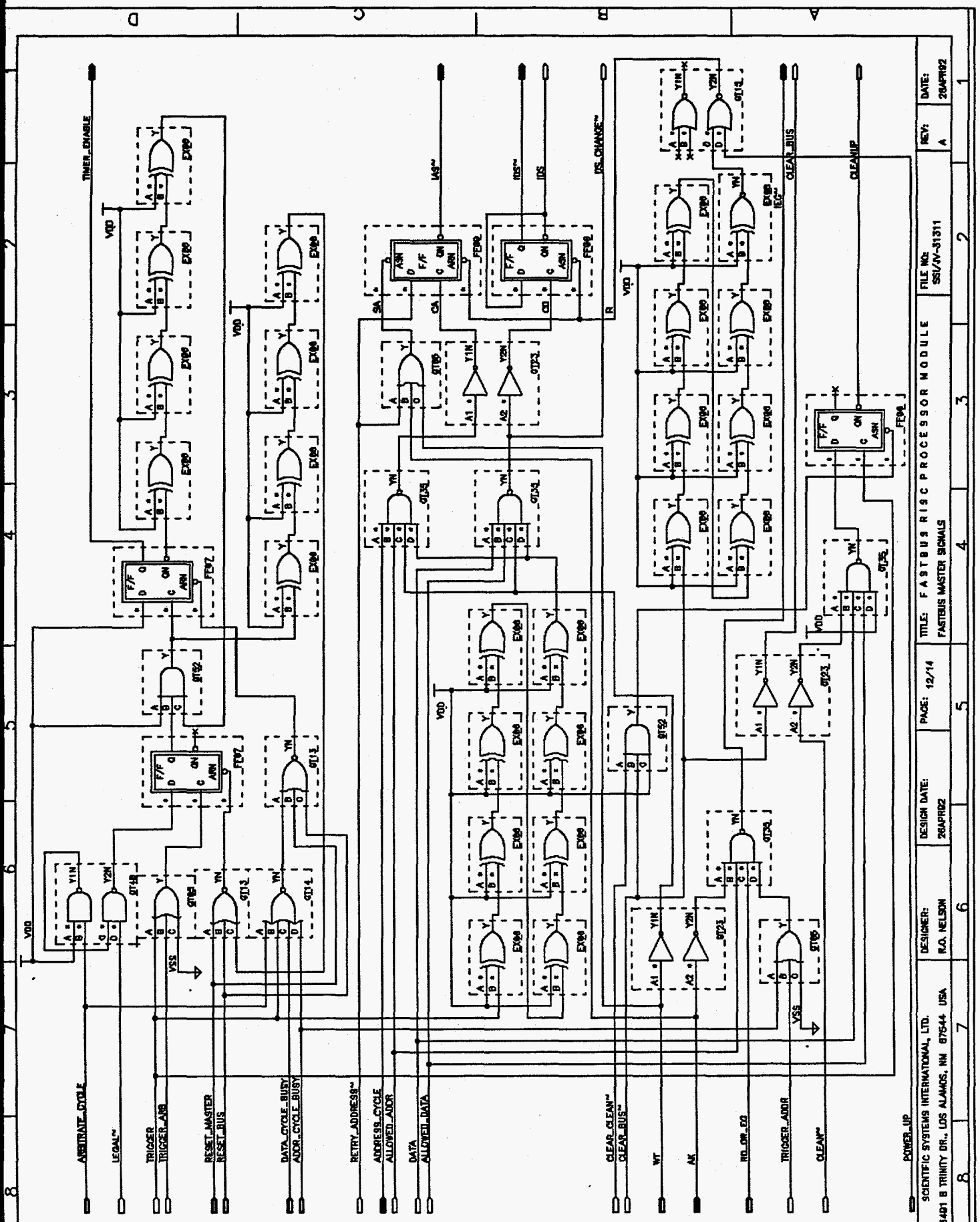

TRIGGER_ADDR
ADDRESS_CYCLE~
TRIGGER_DATA
TRIGGER
OE_AD_MASTER
LATCH_CLOCK~
HOLD_FOR_CLEAN
XFR_DOC_OR_DOF~
MOTHER_CYCLE~
VDD
VSS
XFR_FETCH_DOF1~
XFR_FETCH_DOF2~
XFR_IDLE_DOC1~
XFR_IDLE_DOC2~
XFR_IDLE_DOC3~
XFR_FETCH_DOC~
XFR_DOF_DOC~
XFR_DOC_IDLE2~
MAINTAIN_DOC1~
MAINTAIN_DOC2~
MAINTAIN_DOC3~
MAINTAIN_DOF1~
MAINTAIN_DOF2~
MAINTAIN_DOF3~
ADDRESS_CYCLE
ADDR_CYCLE_BUSY
DATA_CYCLE
CLEANUP_CYCLE
DATA_CYCLE_BUSY
INTERRUPT~
WRITE
JOINT_CLOCK
RESET_MASTER
RESET_BUS
CURRENT_MASTER~
CLEANUP_BUSY
CLEAR_CLEAN~
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 18APR92
PAGE: 6/14
TITLE: FASTBUS RISC PROCESSOR MODULE
TRIGGER ADDRESS AND TRIGGER DATA
FILE NO: SSI/AV-31398
REV: A
DATE: 18APR92

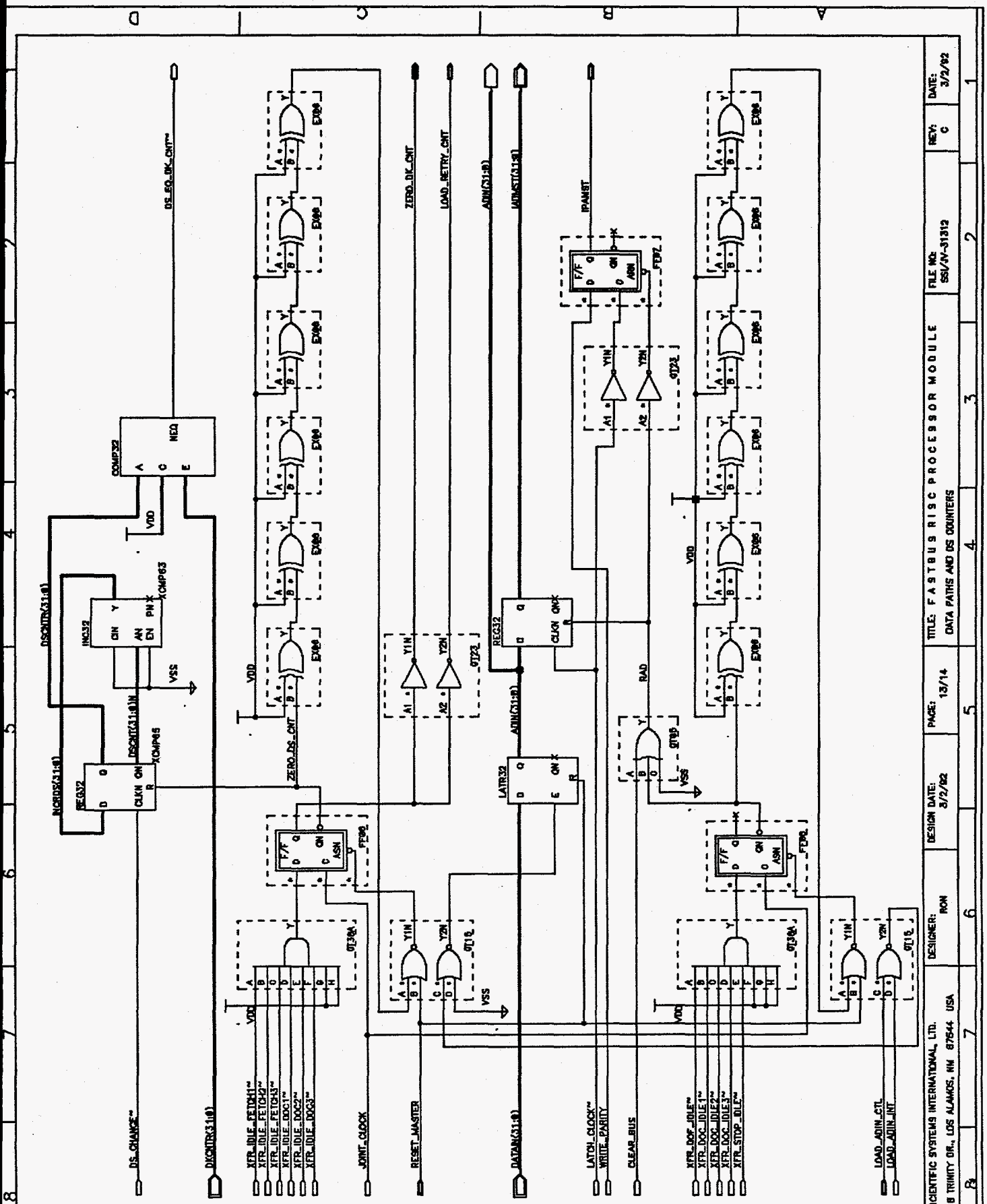


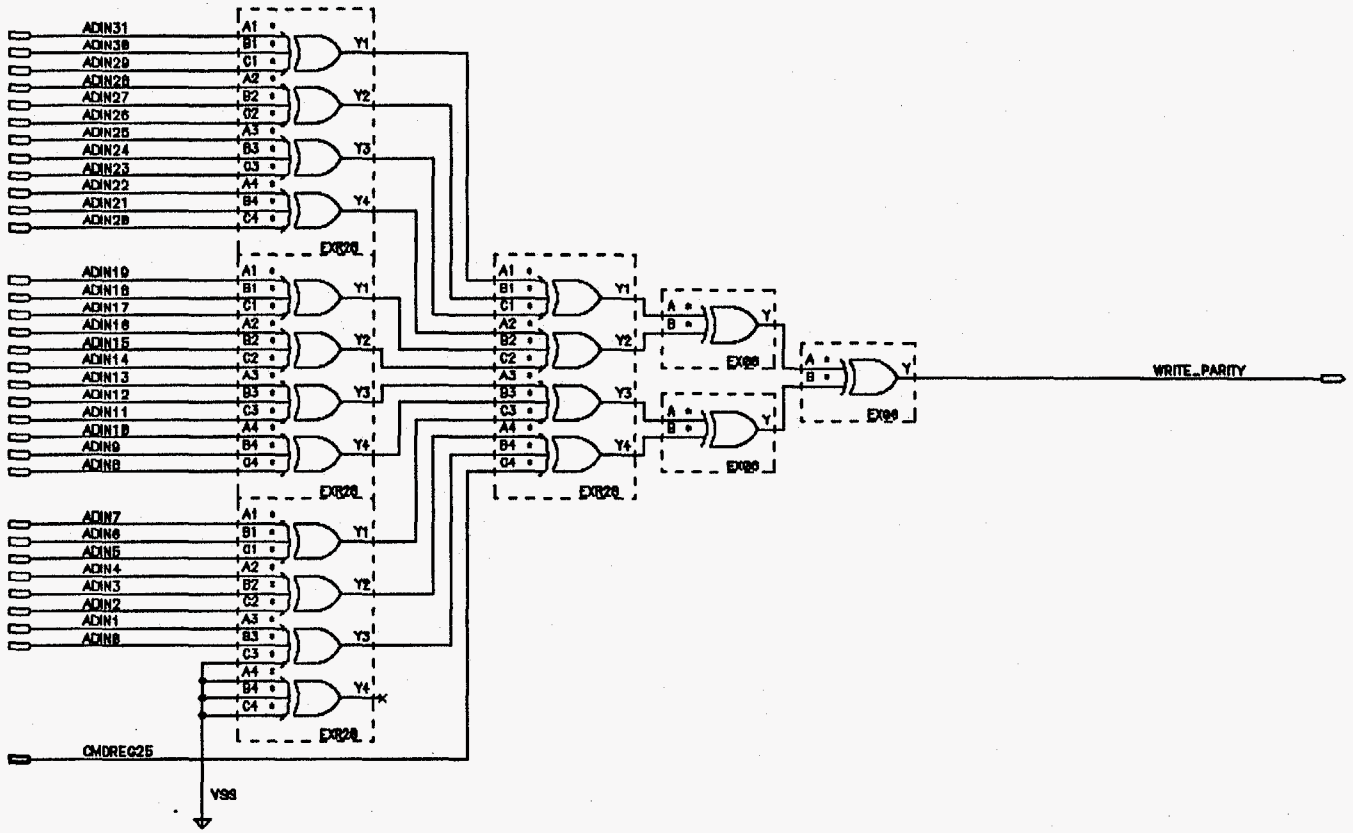
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3421 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 16APR92	PAGE: 10/14	TITLE: FASTBUS RISC PROCESSOR MODULE TRIGGER ADDRESS AND TRIGGER DATA	FILE NO: SSI/JV-31309	REV: A	DATE: 16APR92



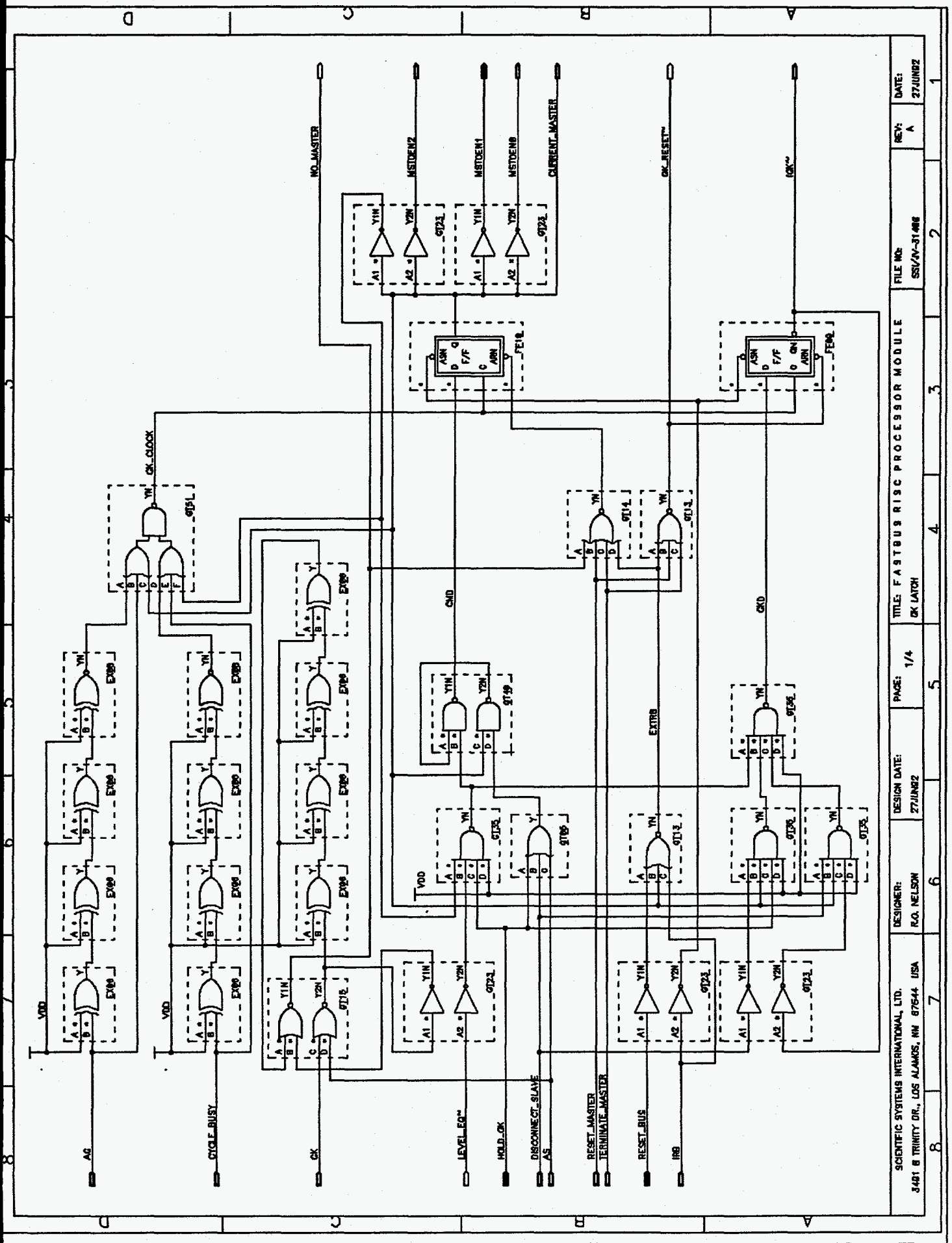
RESET_INTERUPT~
RESET_MASTER
RESET_BUS
DISCONNECT_SLAVE
CURRENT_MASTER~
ADDRESS_CYCLE
LA_MODE_EN~
LA_MODE_EN
DATA_CYCLE
RD_OR_EQ
CLEANUP_CYCLE
MS_CODE0
MS_CODE1
MS_CODE2
IDS~
RD_OR_EC~
TRIGGER
VDD
VSS
DATA
ALLOWED_ADDR
ALLOWED_DATA
AUTO_CLEANUP
CLEAN~
TOGGLE
CLEAN~
ALLOWED_DATA
ALLOWED_ADDR
DATA
CLEAR_BUS~
LEGAL~
ILLEGAL_COMMAND
IRQ~
IMSN0
IMSN1
IMSN2
BLOCK_MODE
WRITE
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 26APR92
PAGE: 11/14
TITLE: FASTBUS RISC PROCESSOR MODULE
FASTBUS MASTER SIGNALS
FILE NO: SSI/JV-31310
REV: A
DATE: 28APR92

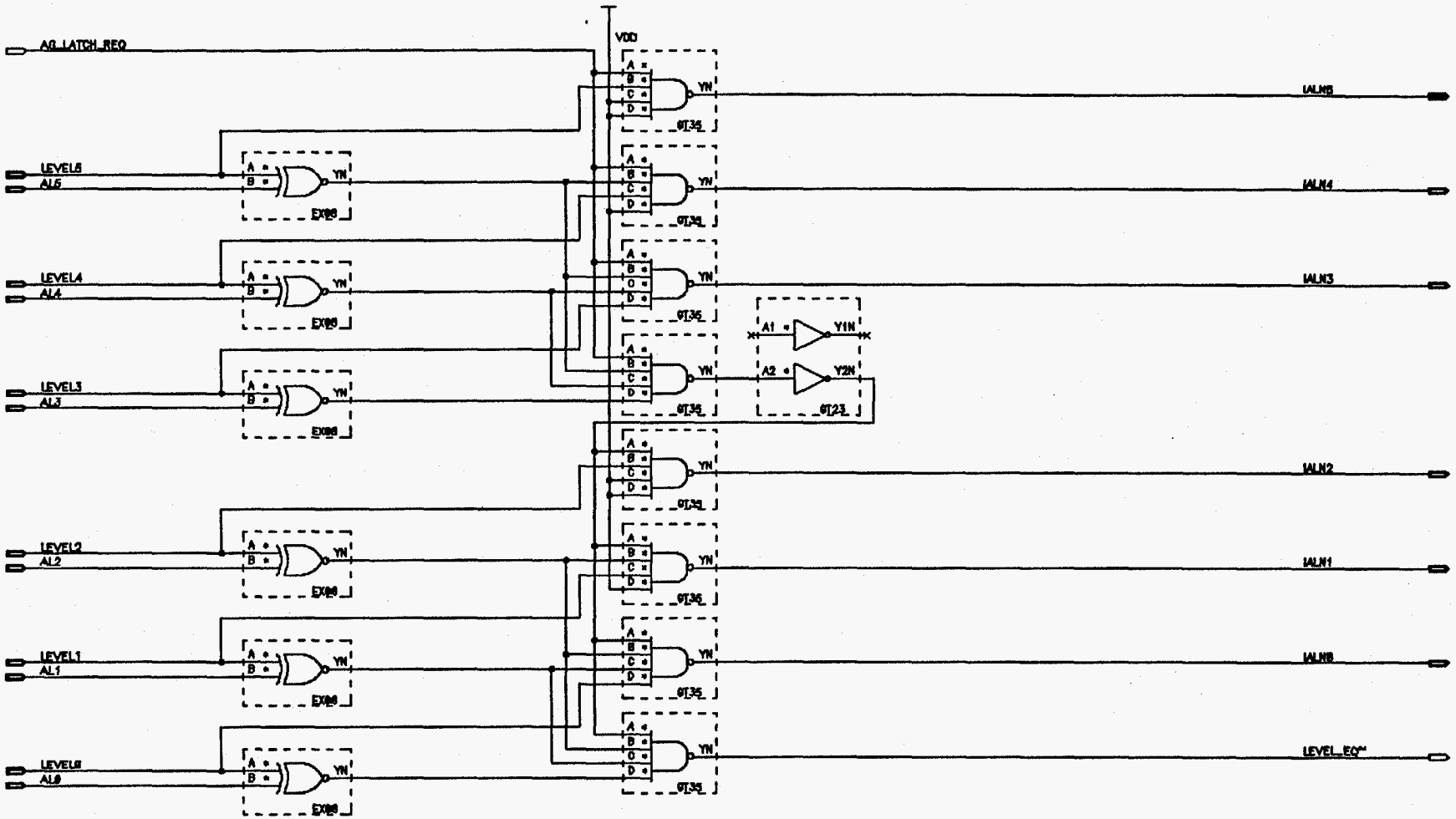

TIMER_ENABLE
VDD
VSS
ARBITRATE_CYCLE
LEGAL~
TRIGGER
TRIGGER_ARB
RESET_MASTER
RESET_BUS
DATA_CYCLE_BUSY
ADDR_CYCLE_BUSY
RETRY_ADDRESS~
ADDRESS_CYCLE
ALLOWED_ADDR
DATA
ALLOWED_DATA
CLEAR_CLEAN~
CLEAR_BUS~
WT
AK
RD_OR_EQ
TRIGGER_ADDR
CLEAN~
POWER_UP
CLEAR_BUS
CLEANUP
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 26APR92
PAGE: 12/14
TITLE: FASTBUS RISC PROCESSOR MODULE
FASTBUS MASTER SIGNALS
FILE NO: SSI/JV-31311
REV: A
DATE: 26APR92


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: RON
DESIGN DATE: 8/2/92
PAGE: 13/14
TITLE: FASTBUS RISC PROCESSOR MODULE
DATA PATHS AND DS COUNTERS
FILE NO: SSI/AY-31312
REV: C
DATE: 3/2/92
COMP32
INC32
REG32
LATR32
F/F
DS_EQ_DK_CNT~
DSCNTR(31:0)
DSCNTR(31:0)N
INCDS(31:0)
ZERO_DS_CNT
ZERO_DK_CNT
LOAD_RETRY_CNT
ADIN(31:0)
MDMAST(31:0)
IPAMST
RAD
DS_CHANGE~
DKCNTR(31:0)
XFR_IDLE_FETCH1~
XFR_IDLE_FETCH2~
XFR_IDLE_FETCH3~
XFR_IDLE_DOC1~
XFR_IDLE_DOC2~
XFR_IDLE_DOC3~
JOINT_CLOCK
RESET_MASTER
DATAIN(31:0)
LATCH_CLOCK~
WRITE_PARITY
CLEAR_BUS
XFR_DOF_IDLE~
XFR_DOC_IDLE1~
XFR_DOC_IDLE2~
XFR_DOC_IDLE3~
XFR_STOP_IDLE~
LOAD_ADIN_CTL
LOAD_ADIN_INT

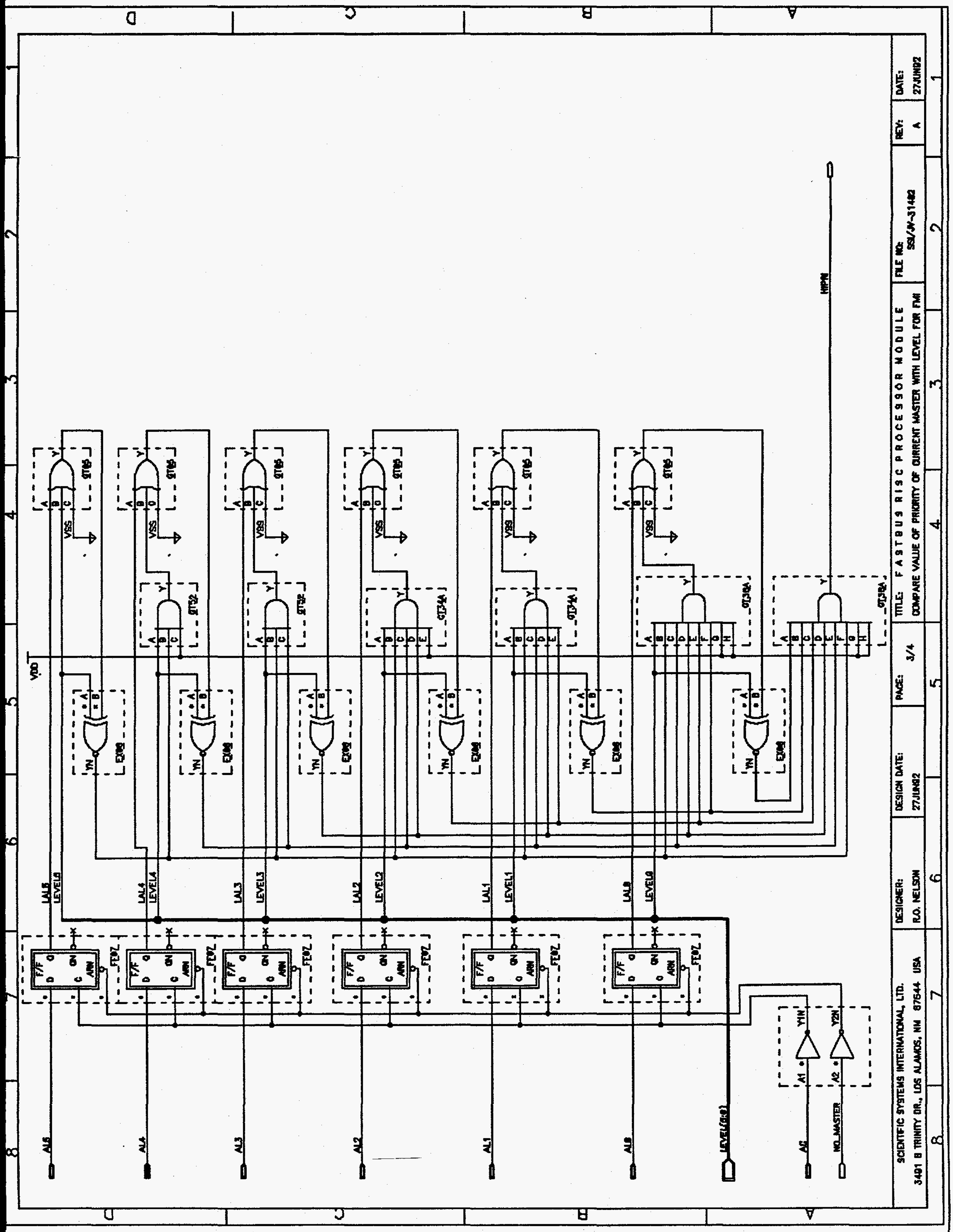


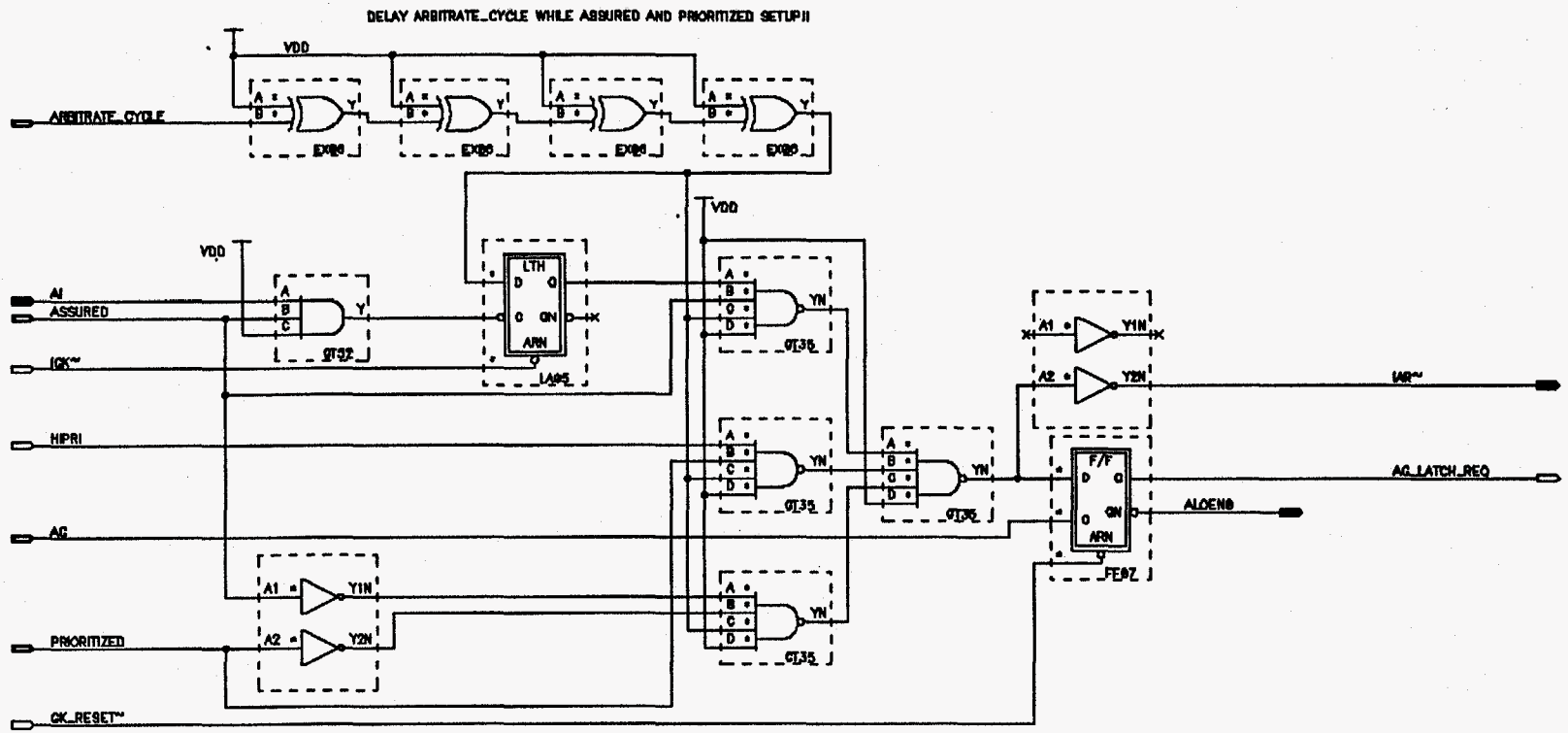
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 17OCT92	PAGE: 14/14	TITLE: FASTBUS RISC PROCESSOR MODULE PARITY FOR MASTER WRITE CYCLE	FILE NO: SSI/JV-31313	REV: A	DATE: 17OCT92


TITLE: FASTBUS RISC PROCESSOR MODULE
CK LATCH
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 27JUN92
PAGE: 1/4
FILE NO: SSI/AV-31488
REV: A
DATE: 27JUN92
NO_MASTER
MSTOEN2
MSTOEN1
MSTOEN0
CURRENT_MASTER
CK_RESET~
!CK~
CK_CLOCK
CMD
EXTRB
VDD
AS
CYCLE_BUSY
CK
LEVEL_EA~
HOLD_CK
DISCONNECT_SLAVE
RESET_MASTER
TERMINATE_MASTER
RESET_BUS
INB

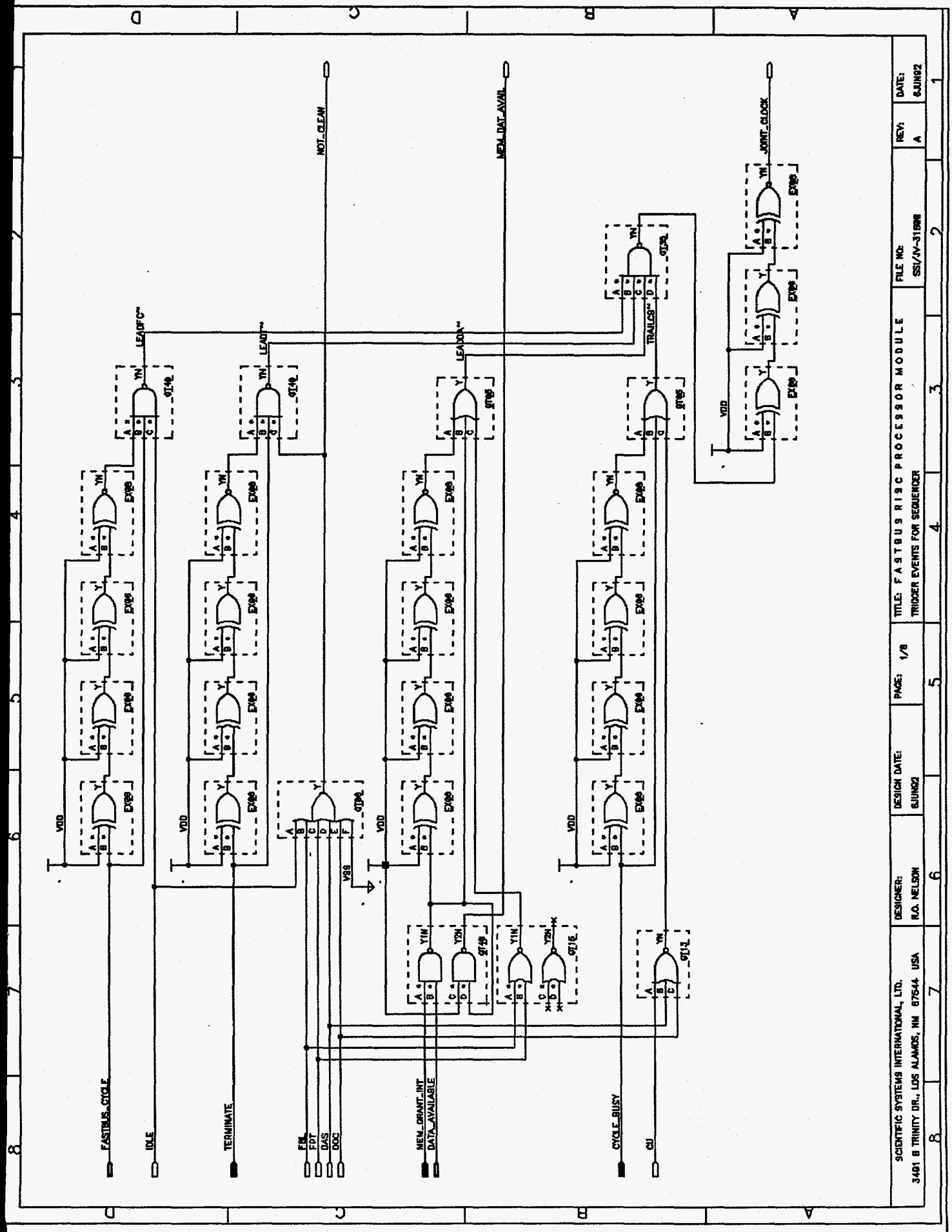


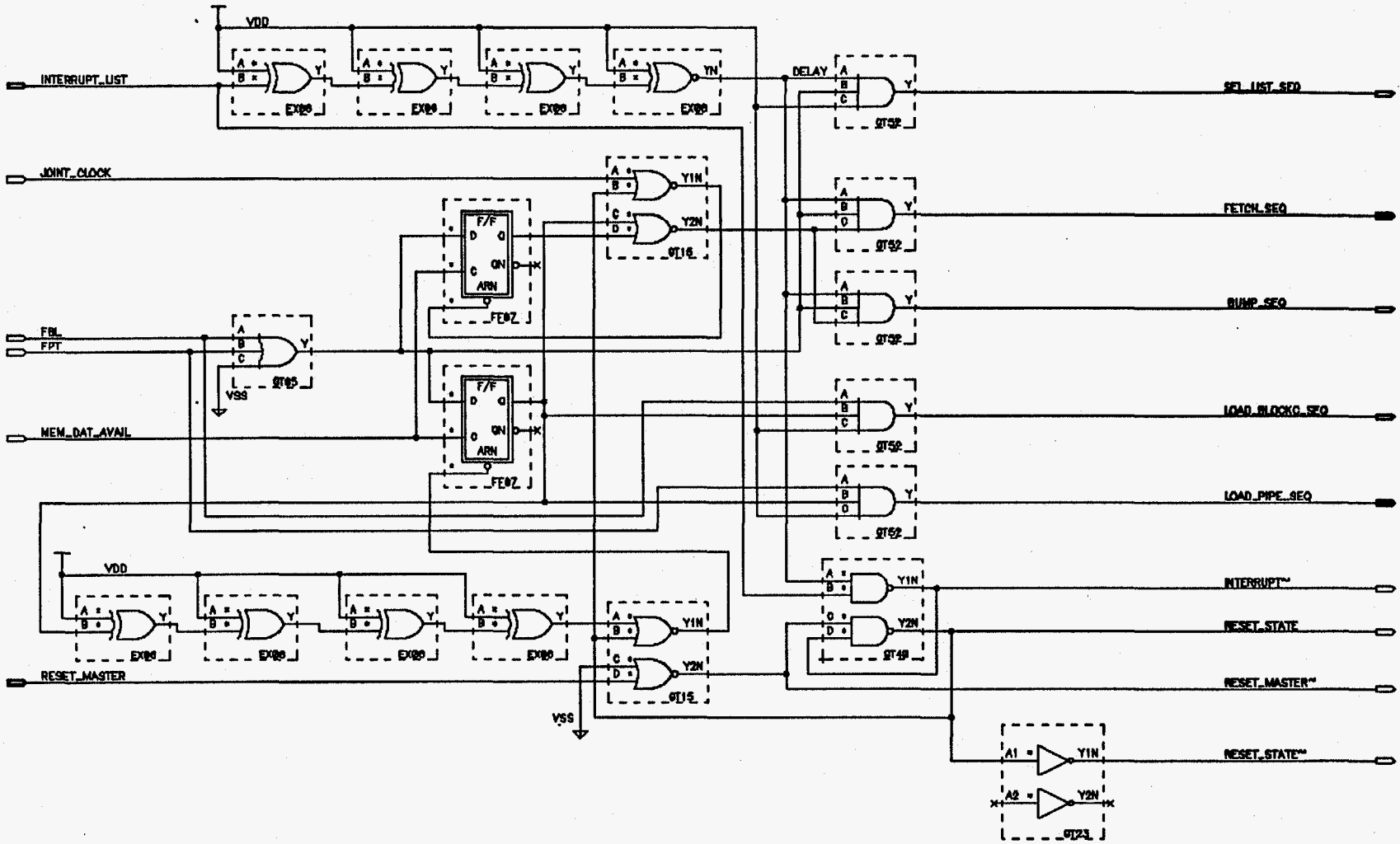
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3481 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 27JUN92	PAGE: 2/4	TITLE: FASTBUS RISC PROCESSOR MODULE DRIVE AL LINES DURING ARBITRATION CYCLE	FILE NO: SSI/JV-31481	REV:	DATE: 27JUN92


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER:
R.O. NELSON
DESIGN DATE:
27JUN92
PAGE:
3/4
TITLE: F A S T B U S R I S C P R O C E S S O R M O D U L E
COMPARE VALUE OF PRIORITY OF CURRENT MASTER WITH LEVEL FOR FMI
FILE NO:
SSI/AY-31482
REV:
A
DATE:
27JUN92
AL5
AL4
AL3
AL2
AL1
AL0
LEVEL(5:0)
AE
NO_MASTER
LAL5
LEVEL5
LAL4
LEVEL4
LAL3
LEVEL3
LAL2
LEVEL2
LAL1
LEVEL1
LAL0
LEVEL0
VDD
VSS
HIPRI

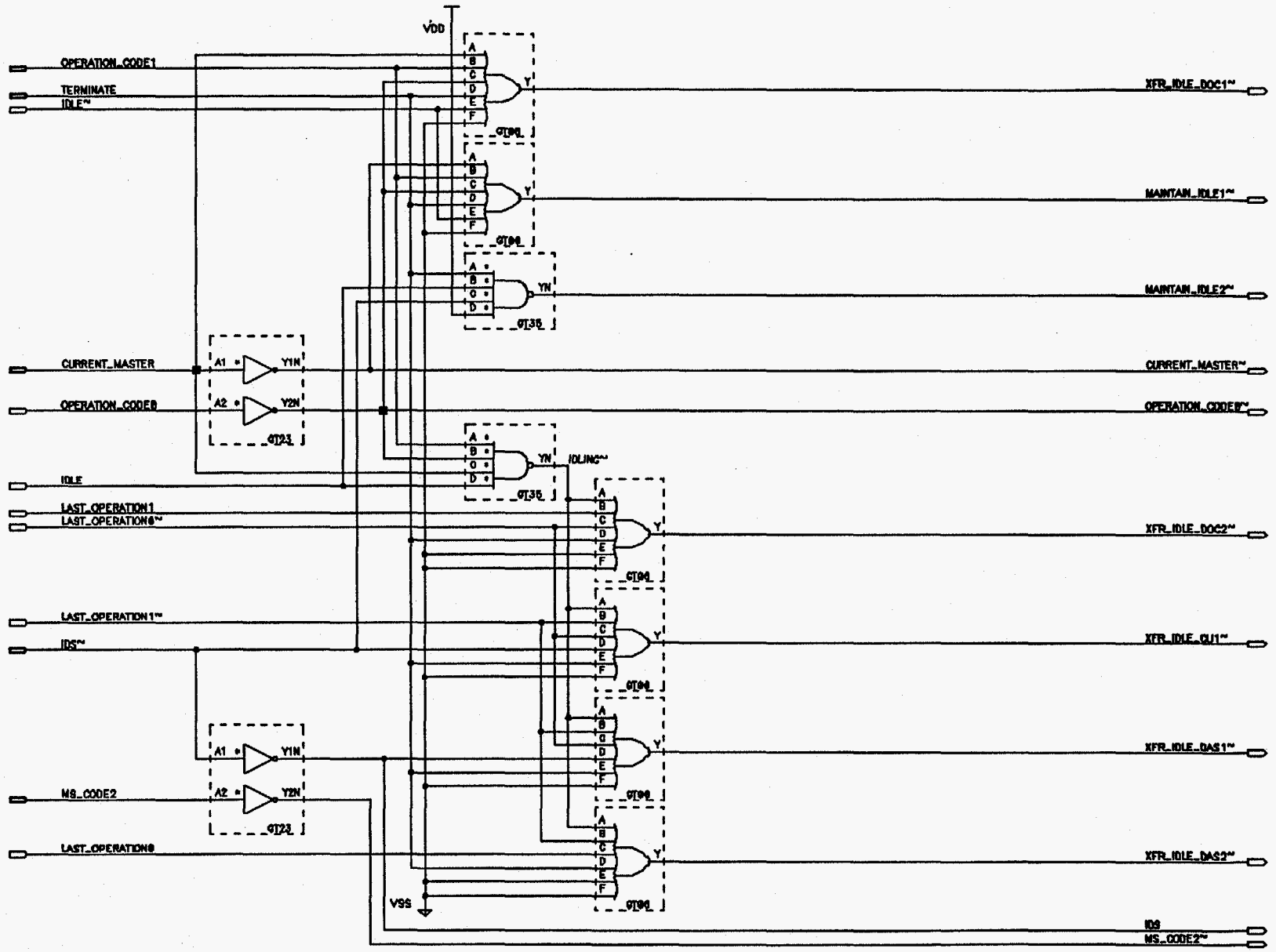


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA		DESIGNER: DRM R.O. NELSON	DESIGN DATE: 27JUN92	PAGE: 4/4	TITLE: FASTBUS MASTER INTERFACE REQUEST LATCH		FILE NO: SSI/JV- 31483	REV: B	DATE: 10/27/92
8	7	6	5	4	3	2	1		

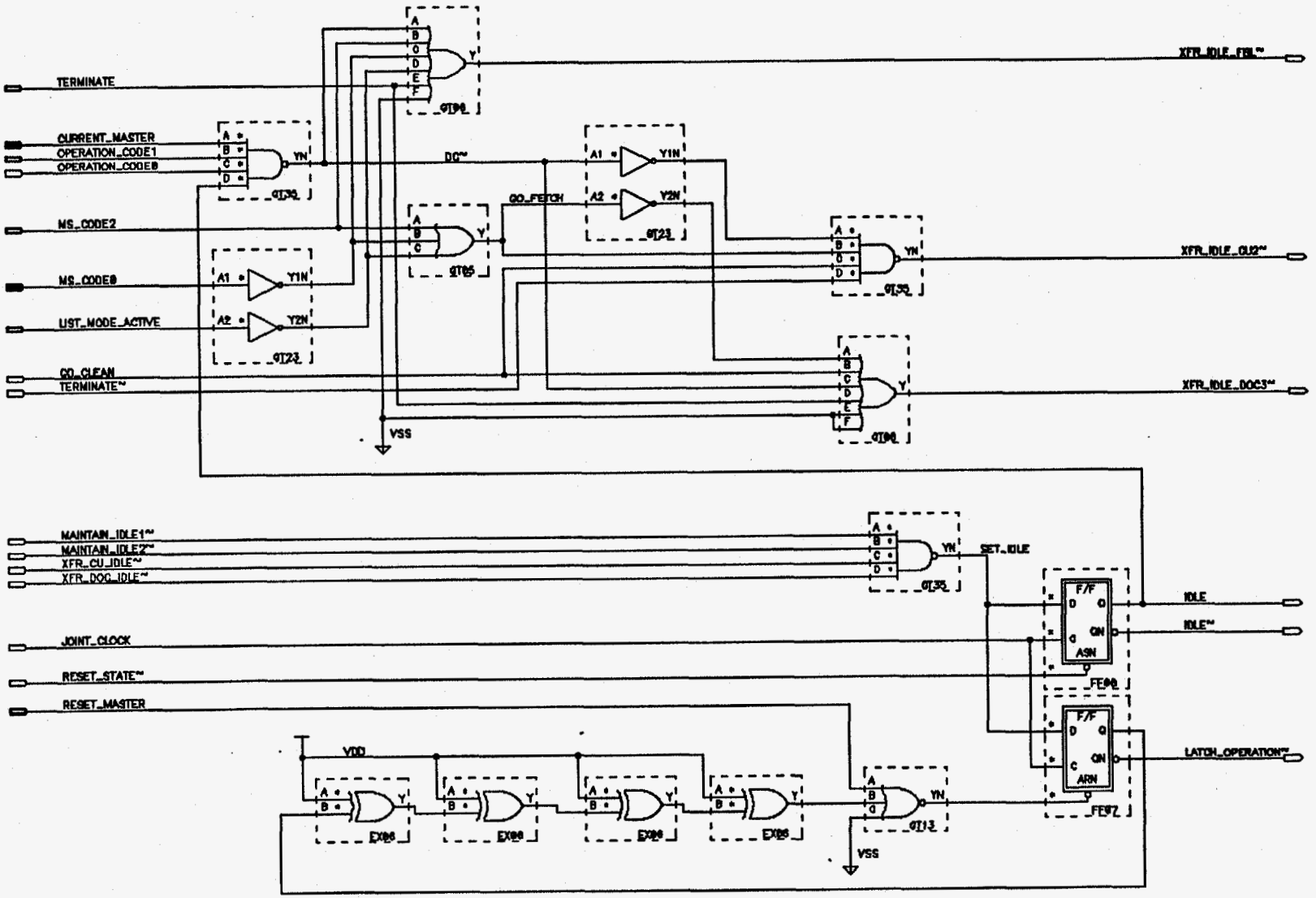

SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 6JUN92
PAGE: 1/8
TITLE: FASTBUS RISC PROCESSOR MODULE
TRIGGER EVENTS FOR SEQUENCER
FILE NO: SSI/AV-31888
REV: A
DATE: 6JUN92
NOT_CLEAN
MEM_DAT_AVAIL
JOINT_CLOCK
LEADFC~
LEADT~
LEADDA~
TRAILDS~
VDD
V38
FASTBUS_CYCLE
IDLE
TERMINATE
FBL
FDT
DAS
DOC
MEM_GRANT_INT
DATA_AVAILABLE
CYCLE_BUSY
CU



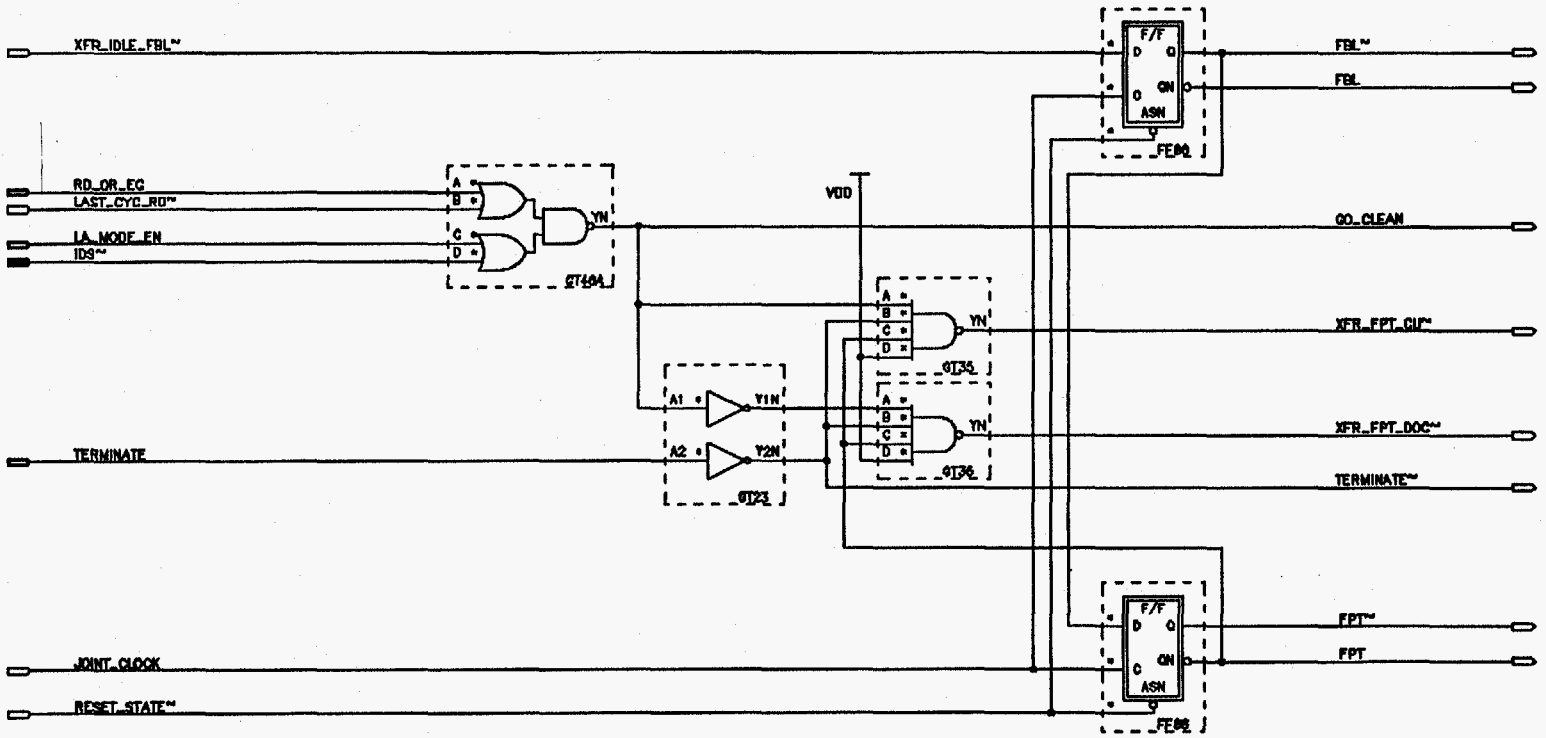
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 6JUN92	PAGE: 2/8	TITLE: FASTBUS RISC PROCESSOR MODULE MEMORY INTERFACE SIGNALS		FILE NO: SSI/JV-31981	REV: A	DATE: 6JUN92
8	7	6	5	4	3	2	1	



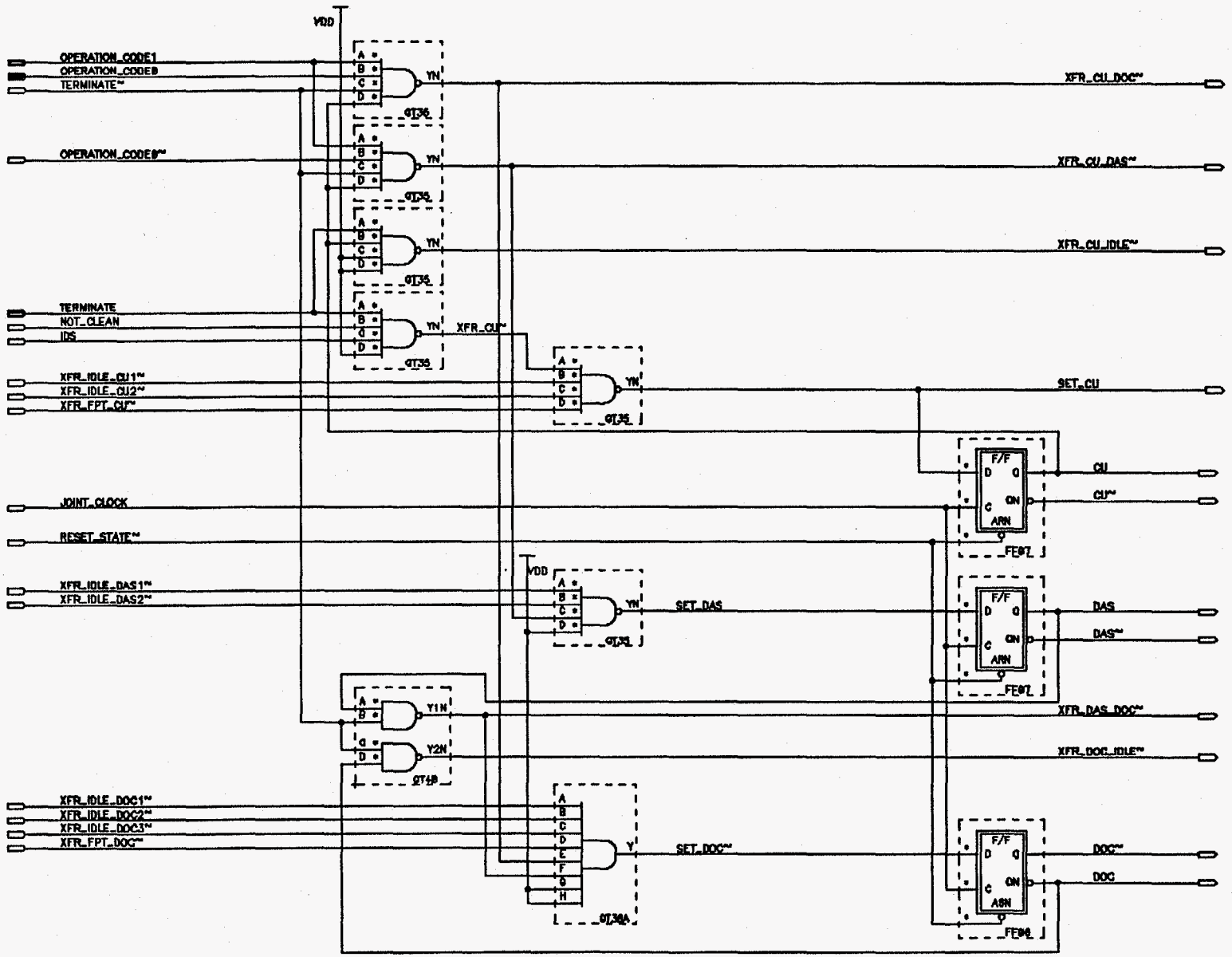
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3491 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 11APR92	PAGE: 3/8	TITLE: FASTBUS RISC PROCESSOR MODULE IDLE STATE	FILE NO: SSI/JV-31582	REV: A	DATE: 11APR92



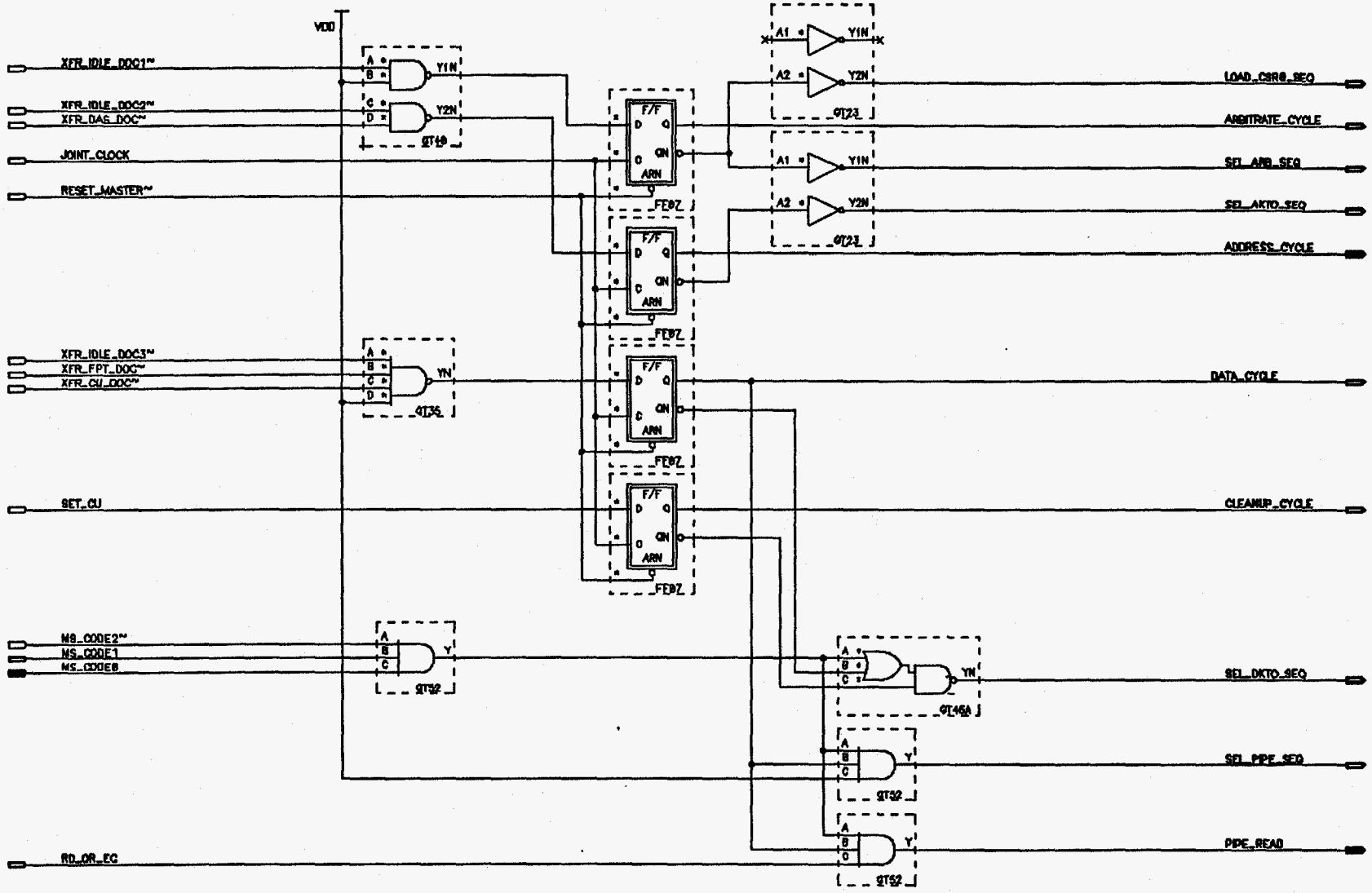
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.G. NELSON	DESIGN DATE: 11APR92	PAGE: 4/8	TITLE: FASTBUS RISC PROCESSOR MODULE IDLE STATE	FILE NO: SSI/JV-31803	REV: A	DATE: 11APR92



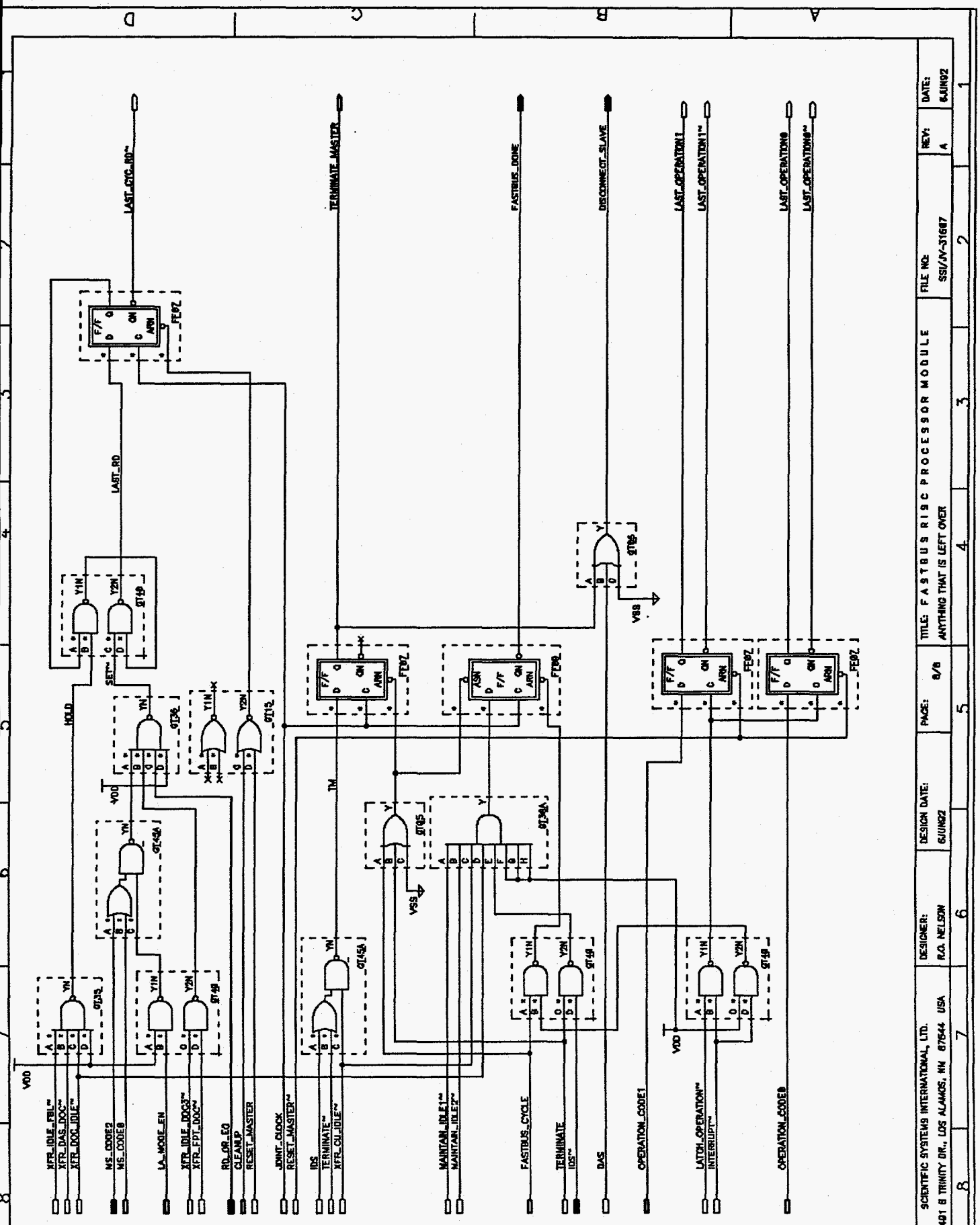
SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 6JUN92	PAGE: 5/8	TITLE: FASTBUS RISC PROCESSOR MODULE FETCH BLOCK LENGTH AND FETCH PIPE TIME STATES	FILE NO: SSI/JV-31864	REV: A	DATE: 6JUN92



SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3421 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 6JUN92	PAGE: 4/8	TITLE: FASTBUS RISC PROCESSOR MODULE CLEANUP STATE, DROP AS STATE, AND DO CYCLE STATE	FILE NO: SSI/JV-31686	REV: A	DATE: 6JUN92



SCIENTIFIC SYSTEMS INTERNATIONAL, LTD. 3481 B TRINITY DR., LOS ALAMOS, NM 87544 USA	DESIGNER: R.O. NELSON	DESIGN DATE: 6JUN92	PAGE: 7/8	TITLE: FASTBUS RISC PROCESSOR MODULE DECODE OF CYCLE TYPES	FILE NO: SSI/JV-31586	REV: A	DATE: 6JUN92


SCIENTIFIC SYSTEMS INTERNATIONAL, LTD.
3401 B TRINITY DR., LOS ALAMOS, NM 87544 USA
DESIGNER: R.O. NELSON
DESIGN DATE: 6/JUN92
PAGE: 8/8
TITLE: FASTBUS RISC PROCESSOR MODULE
ANYTHING THAT IS LEFT OVER
FILE NO: SSI/JV-31687
REV: A
DATE: 6JUN92
LAST_CYC_RQ*
TERMINATE_MASTER
FASTBUS_DONE
DISCONNECT_SLAVE
LAST_OPERATION1
LAST_OPERATION1*
LAST_OPERATION0
LAST_OPERATION0*
LAST_RD
HOLD
SET*
TM
VDD
VSS
XFR_IDLE_FBL*
XFR_DAS_DOC*
XFR_DOC_IDLE*
MS_CODE2
MS_CODE0
LA_MODE_EN
XFR_IDLE_DOC3*
XFR_FDT_DOC*
RD_OR_EQ
CLEAN_P
RESET_MASTER
JOINT_CLOCK
RESET_MASTER*
IDS
TERMINATE*
XFR_CU_IDLE*
MAINTAIN_IDLE1*
MAINTAIN_IDLE2*
FASTBUS_CYCLE
TERMINATE
IDS*
DAS
OPERATION_CODE1
LATCH_OPERATION*
INTERRUPT*
OPERATION_CODE0

APPENDIX F

FRPM ADVERTISING FLYER

THE FASTBUS RISC PROCESSOR MODULE

(F R P M)

- ***SUN SparcEngine Processor***
- ***Simple FASTBUS Interface***
- ***VxWorks RTOS***

Scientific Systems International Ltd.

3491B Trinity Dr.
Los Alamos, NM 87544 USA
505 662-6712

International Engineering & Consulting Services

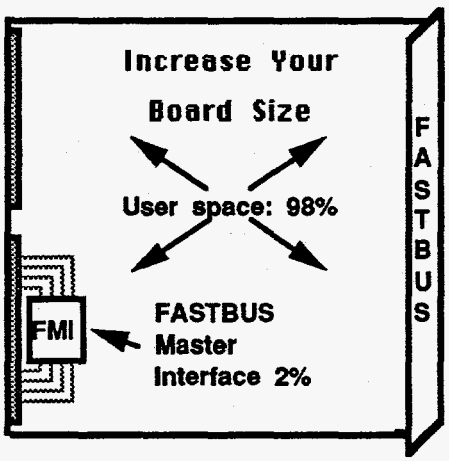
FASTBUS MASTER INTERFACE

- Single-chip pin-grid array
- Meets IEEE 960 specifications for FASTBUS slaves
- ECL I/O
- Direct attachment to FASTBUS
- Memory-like user interface
- FMI chip implements:

TBD

- User controls for:
  Timing

TBD



Implemented in a Raytheon gate array using the Bit 1 ECL technology

--- -I/O pins, --- pin PGA package

Uses 21 cm^2 of board space

Q4-1990

Scientific Systems International Ltd.

3491B Trinity Dr.
Los Alamos, NM 87544 USA
(505)662-6712

International Engineering & Consulting Services

January 1990