Sparse Distributed Memory Prototype: Address Module Hardware Guide

M. J. Flynn, R. Zeidman, E. Lochner

Technical Report CSL-TR-88-373

December 1988

This research is supported by NASA under contract NAGW 419, in cooperation with RIACS (Ames Research Center).

Sparse Distributed Memory Prototype: Address Module Hardware Guide

by
M. J. Flynn, R. Zeidman, and E. Lochner

Technical Report CSL-TR-88-373
December 1988

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305-4055

Abstract

This document is a detailed specification of the hardware design of the Address Module for the prototype Sparse Distributed Memory. It contains all of the information needed to build, test, debug, modify and operate the Address Module.

Contents

List of Figures

List of Tables

Chapter 1

Applicable documents

The following documents, along with this one, comprise a set which is useful for understanding the operation of the sparse distributed memory prototype. For clarification of any subjects discussed in this document, please consult these other documents in the set.

1. "Sparse Distributed Memory Prototype: Principles of Operation (Revised)," M. J. Flynn, P. Kanerva, B. Ahanin, N. Bhadkamkar, P. Flaherty, P. Hickey, E. Lochner, K. Webber, R. Zeidman. Stanford University, Stanford, CA. Computer Systems Laboratory Technical Report CSL-TR-87-338.

2. "Sparse Distributed Memory Prototype: User's and Programmer's Guide," P. Hickey. NASA Research Institute for Advanced Computer Science, Mountain View, CA. Technical Report RIACS TR 88.48.

Chapter 2

Design and Operation

This chapter provides a complete description of the design and operation of the Address Module.

2.1 General Overview

The Address Module is a custom board which holds the 8,192 256-bit addresses of the Sparse Distributed Memory in its Hard Address Memory. When a Reference Address is written to it, the Address Module will sequentially compare each Hard Address to the Reference Address. When the Hamming distance between the two addresses is less than or equal to a predefined limit, stored in the Limit Register on the Address Module, the Address Module signals the Control Module that there has been a hit. When this occurs, a 13-bit tag associated with the relevant hard address is stored in a tag cache on the Address Module. By reading this Tag Cache, the Control Module can determine which Hard Address caused the hit and the Hamming distance between the Hard Address and the Reference Address. This operation is described in more detail in the subsequent sections.

The Address Module communicates with the Control Module over a standard VME bus using long word (32-bit) transfers.

2.2 Block Diagram

The Block diagram of the Address Module is shown in Figure 2.1.

The VME bus has 32 address lines (bus A) and 32 data lines (bus D) on it. The A bus is decoded and one of the 28 REG lines is asserted, depending on which register is to be read/written. The D bus, after going through line drivers in the VME bus data interface, is referred to as the XD bus. When the Hard Address Memory has been loaded and the appropriate registers have been written via the Hard Address Register (HAR),

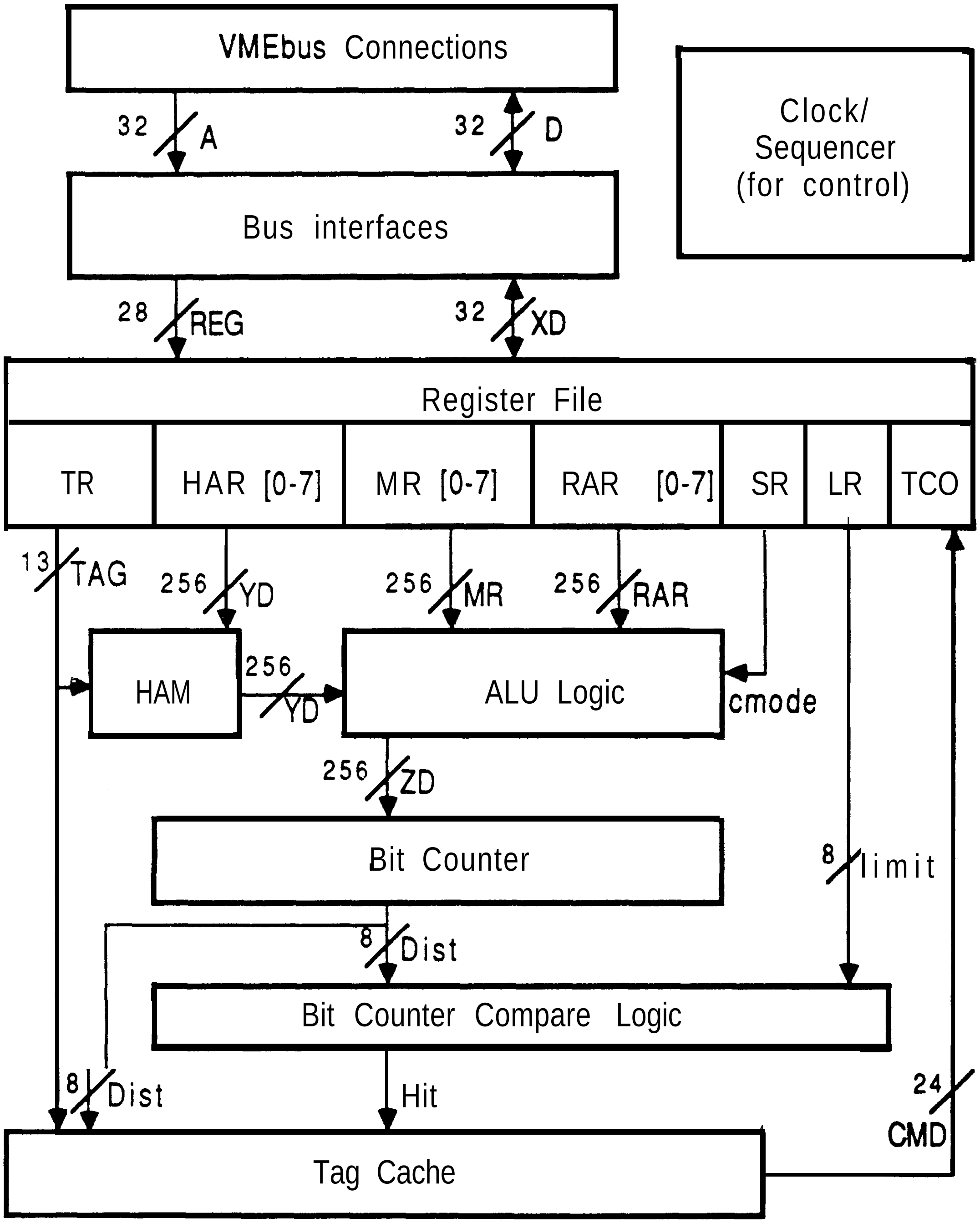


Figure 2.1: Block Diagram of Address Module.

Table 2.1: Address Module Register Set.

Register	Longname	[width]	R/W	#32-bit Regs	R#
TR	Tag Register	[16]	W	1	0
A 13 bit "Tag" which locates a Hard Address in the Hard Address Memory.					
HAR	Hard Address Register	[256]	W	8	1-8
Enters a Hard Address into the Hard Address Memory.					
RAR	Reference Address Reg.	[256]	W	8	9-16
Sets the Reference Address-the Address in Question.					
MR	Mask Register	[256]	W	8	17-24
Sets the Hamming AND mask.					
LR	Limit Register	[8]	W	1	25
Defines the radius of the Hamming Sphere. Any tag with Hamming distance less than or equal to this will be written to the TCO.					
SR	Command/Status Register	[8]	R/W	8	26
The bits in this register determine the current status of the machine.					
TCO	Tag Cache Output	[24]	R	1	2i
256-word cache that holds the accepted tags.					

the state machine is put in RUN mode by writing to the Status Register (SR). The ALU then receives signals from the Hard Address Register (HAR), Mask Register (MR), and Reference Address Register (RAR). The output of the ALU is the ZD bus, which has a one in each bit location in which the YD and RAR bits differ (or, if the CMODE signal is asserted, the bit locations in which they are the same.) The bit counter counts the number of ones on the ZD bus and outputs the result as DIST. Details of the bit counter can be found in Figures 2.3 and 2.4, or in the appendices. If the bit counter compare finds that the distance is less than or equal to the limit, it signals the Tag Cache to store the tag and distance. When the Tag Cache Output (TCO) is read by the Control Module, the data is taken off the CMD bus and put on the XD bus and finally out to the VME bus. Note that there are more control signals than are shown in the figures; these will be dealt with in more detail in the following sections.

2.3 The Register Set

The Address Module appears to the Control Module as a set of sequential memory locations on the VME bus. The addresses are defined as shown in Table 2.3.

2.3.1 **Reading the Command/Status Register**

Reading the Command/Status Register returns the following bits:

Bits	7	6	5	4	3	2	1	0
	X	FULL	DONE	CMODE	EMPTY	RST	ST1	ST0

These bits are defined as follows:

X: DON'T CARE

STATE:

ST1	ST0	
0	0	Reset
0	1	Running
1	0	Hit
1	1	Wait/Done

Reset: At power-on, reset is asserted for approximately 200 msec.

RST	
0	Ready
1	Reset

Tag Cache Empty Indicator:

EMPTY	
0	TCO not empty
1	TCO empty

Complemented Mode:

CMODE	
0	Uncomplemented Addresses
1	Complemented Addresses

Done Indicator:

DONE	
0	AM not done processing
1	AM done processing

Full Indicator:

FULL	
0	TC is full
1	TC is not full

2.3.2 **Writing the Command/Status Register**

Writing the Command/Status Register involves the following bits:

Bits	7	6	5	4	3	2	1	0
	X	X	X	CMODE	FORCE	FRST	FST1	FST0

These bits are defined as follows:

X: DON'T CARE

FORCED STATE:

FORCE	FST1	FST0	
0	X	X	Normal operation
1	0	0	Reset
1	0	1	Running
1	1	0	Hit
1	1	1	Wait/Done

Forced Reset:

FRST	
0	Ready
1	Reset

Complemented Mode:

CMODE	
0	Use uncomplemented addresses
1	Use complemented addresses

Note that when the FORCE bit is set, the AM is forced into a particular state until the FORCE bit is reset. This is used for debugging purposes only.

2.3.3 Reading the Tag Cache Output Register

Bits in the tag cache output are defined as follows:

Bit23 - Bit16	Hamming Distance
Bit15 - Bit0	Tag ID

2.4 Theory of Operation

2.4.1 Major Submodules

Figure 2.1 is the block diagram for the AM. It shows the seven submodules of the AM which are:

1. Clock and Sequencer: This submodule contains the master state machine and the master clock.
2. Bus Interface. The bus interface allows the Address Module to communicate with the Control Module over the VME bus. Interface logic provides for address

mapping and maintenance of control information. The VME bus also supplies power and ground to the board.

3. Register File. The register file contains all registers used by the Address Module.

4. Hard Address Memory. All 8,192 256-bit addresses are contained in this memory, which consists of 32 8KB static RAMs.

5. Arithmetic Logic Unit. This submodule performs the 256-bit exclusive-OR and mask operations. The Reference Address Register and Mask Register are combined here with a Hard Address by a logic unit, which consists of 64 programmable logic arrays (PLAs). The ALU result is a 256-bit quantity with a 1 in each bit position in which the Reference Address differs from the Hard Address and is not masked by a 0 in the corresponding bit position in the Mask Register. The *number* of 1s in this result is the Hamming distance between the Reference Address and the Hard Address. The Mask is a 256-bit user-specified pattern to restrict the Hamming distance calculation to a subset of the 256 bits, if desired. The CMODE bit in the Command/Status Register can be set which causes the ALU to use the complemented Hard Address.

6. The Bit Counter and Bit Counter Compare Logic. The bit counter calculates the Hamming distance by adding the 256 bits of the ALU output and then compares the result with the Limit Register. If the result is less than or equal to the Limit Register, there is a "hit," and both the Tag associated with the particular Hard Address and its Hamming distance from the Reference Address are written into the Tag Cache.

7. Tag Cache. During an operation, when the Address Module finds a Hard Address whose Hamming distance from the Reference Address is within the limit prescribed by the Limit Register, the Tag associated with that Hard Address, as well as the Hamming distance, is written into the Tag Cache. The Tag Cache acts as a First In–First Out buffer between the Address Module and the Control Module, thus allowing the ALU operation to occur without the need for synchronization between the two modules. It can hold up to 256 Tags and their associated Hamming distances. When the Tag Cache has recorded 256 Hits, the Address Module enters the WAIT state, suspending operation until the Control Module has read all 256 locations. Once this occurs, the Address Module enters the RUN state, resuming operation until another 256 Hits have been recorded. In this way, the Tag Cache is capable of recording an unlimited number of hits.

2.4.2 State Transitions

Figure 2.2 details the state diagram of the Address Module. The four processor states are as follows:

0. **RESET.** This is the power-up state of the AM. However, the Address Module will enter the RUN state and begin processing garbage data once the initial power-on reset mechanism has finished. Before writing the Hard Address Memory or in any other way initializing the AM, the Control Module must put the AM in the RESET state by setting the FRST bit in the Command/Status Register. The AM will then remain in the reset state until the FRST bit is cleared, causing the AM to enter the RUN state and begin processing the Reference Address.

1. **RUN.** The AM enters this state from a **reset** when the FRST bit in the Command/Status Register is zeroed. It continues to loop within this state until a hit occurs or until the Hard Address Memory is exhausted.

2. **HIT.** When a hit occurs, the AM enters the HIT state and writes the Tag associated with the Hard Address, and its Hamming distance, into the Tag Cache. This state is included for timing reasons, enabling the Address Module to process data at a higher frequency (1 MHz clock rate).

3. **WAIT/DONE.** This state is entered when the Hard Address Memory has been completely examined or when the Tag Cache is full (data has been stored in its last location which has not yet been read by the Control Module). The DONE status bit allows the Control Module to differentiate between these two conditions. The AM will leave the WAIT state and resume the RUN state when the CM reads the last location in the Tag Cache. The AM will remain in the DONE state until the CM sets the FRST status bit, forcing the AM into the RESET state during which it can be set up for more Reference Address processing.

We will now examine each submodule of the AM in detail. The schematics for each submodule are given in Appendix D.

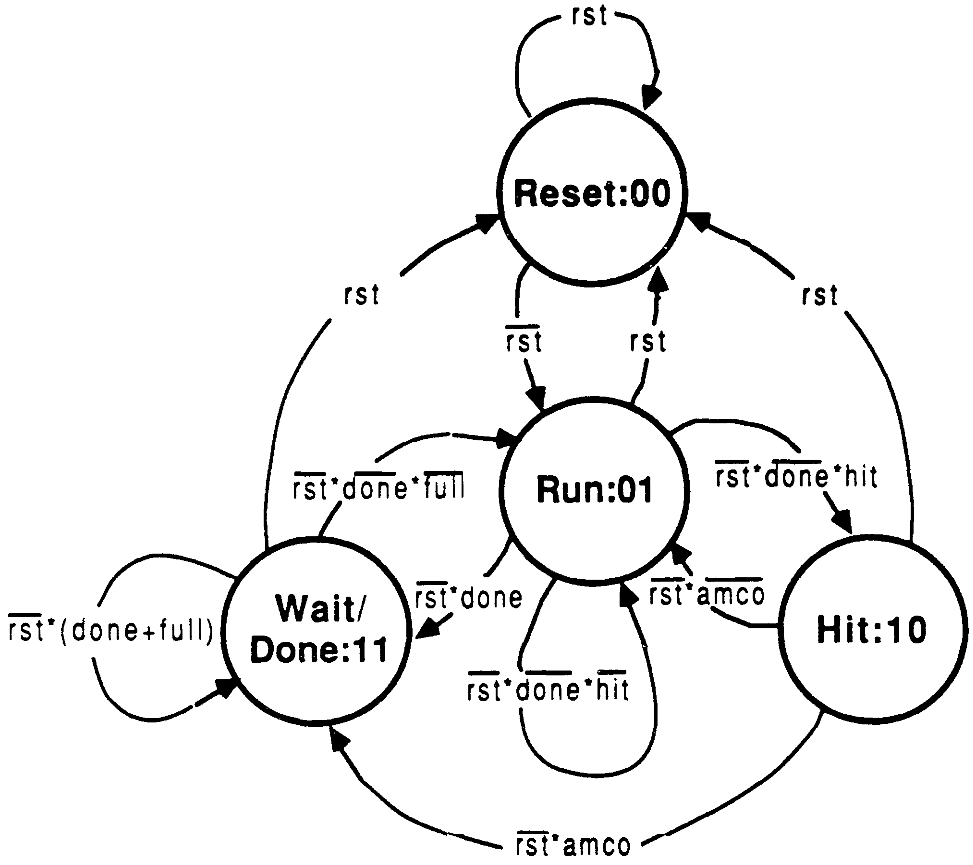
2.4.3 The Clock/Sequencer

The Clock/Sequencer implements the state machine described above. It consists of a system clock, power-on reset circuitry, a registered PAL to implement the state machine, and a 13-bit tag counter.

The clock runs at 4.0 MHz and is generated by a crystal, which is then divided down by two flip flops in a single 74LS74 package. The resulting system clock is 1.0 MHz for use by the Address Module.

The power-on reset circuitry consists simply of a buffered resistor and capacitor which insures that the board is reset when power is applied, for a minimum time of 200 msec. This feature may be eliminated in future versions since the Control Module must reset the AM before initializing it anyway.

The 16R4 registered PAL implements the state machine described earlier. It is responsible for putting the AM into one of its four states: RESET, RUN, HIT, or


rst
Reset:00
rst
rst
$\overline{rst}$
rst
Run:01
$\overline{rst}$*$\overline{done}$*$\overline{full}$
$\overline{rst}$*$\overline{done}$*hit
$\overline{rst}$*(done+full)
Wait/
Done:11
$\overline{rst}$*done
$\overline{rst}$*$\overline{amco}$
Hit:10
$\overline{rst}$*$\overline{done}$*$\overline{hit}$
$\overline{rst}$*amco

Description of Signals:

rst The value of the state register reset line.

hit Is high when the Hamming distance is less than or equal to the limit.

amco Tag Cache will be full at the next positive clock edge.

full Tag Cache is full.

done The entire address space has been searched.

Figure 2.2: State transition diagram of Address module.

WAIT/DONE. It also outputs the signals ACNTEN , TCNTEN , CMTAG , and AMTAG , which are described in the section on internal interfaces.

The tag counter consists of two 74ALS867 counter chips. This counter is used to generate tags in order to address each location in the Hard Address Memory during RUN cycles.

2.4.4 The Bus Interface

The Bus Interface submodule handles all of the VME-bus addressing for the AM, and internal to external bus translations. It consists of a set of bus transceivers, a 16L8 PAL for board address decoding, two 74LS154 demultiplexers to generate the register select lines, and connectors to the VME bus for communication and to supply the necessary power to the board. Only Long Words (32-bit quantities) should be read or written to the AM. Any other data type may result in incorrect data transfers.

NOTE: It is important to note that the board address from the VME bus is qualified with the data strobes DS0 and DS1. While this should work in most cases, there is a possibility that the address can become invalid before the data strobes are deasserted due to skew on the backplane. This would result in incorrect registers being accessed for a short time. To avoid this problem in any future designs, the register selects should be latched on the falling edge of the data strobes.

2.4.5 The Register File

The Register File contains all registers used by the Address Module. These are:

1. The Hard Address Registers (HARs) are actually 32 8K x 8 static RAMs which make up the Hard Address Memory. The Control Module accesses them via the VME bus as if they were eight 32-bit registers.

2. The Limit Register (LR) consists of a single 74LS373 8-bit latch which specifies a maximum Hamming distance. If the Reference Address is within this distance from a particular Hard Address, it is stored in the Tag Cache as a hit.

3. The Mask Register (MR) consists of 32 74LS373 8-bit latches which hold the 256-bit mask. The mask specifies which bits are to be ignored when the Reference Address is compared to the Hard Address Memory.

4. The Reference Address Register (RAR) consists of 32 74LS373 8-bit latches, which hold the 256-bit address which the Address Module compares to the Hard Address Memory.

5. The Command/Status Register (SR) consists of a buffer for reading and a latch for writing. When read, the 8-bit 74LS244 buffer outputs the status of the Address

Module onto the VME bus. The individual status bits, FULL, DONE, CMODE, EMPTY, RESET, ST1, ST0, were defined earlier. When written the 74LS373 8-bit latch stores the command bits CMODE, FORCE, FRST, FST1, FST0 which are used by the Control Module to control the AM and were defined earlier.

6. The Tag Cache Output (TCO) is the 256 x 24 bit output of the 7130 and 7140 dual part memories which make up the Tag Cache. The Control Module reads the TCO as a 24-bit register via the VME bus.

7. The Tag Register (TR) consists of two 74LS373 8-bit latches used to hold a 13-bit quantity. This quantity, the Tag, is used to reference one of the 8,192 different Hard Addresses in the Hard Address Memory.

2.4.6 The Hard Address Memory

The Hard Address Memory stores all 8,192 256-bit Hard Addresses. It consists of 32 8k-by-8-bit static CMOS RAMs, along with 74LS244 buffers which demultiplex the inputs and outputs of the RAMs.

2.4.7 The Arithmetic Logic Unit

The ALU uses the Reference Address Register, the Mask Register, and a logic unit to perform a 256-bit XOR-AND operation. The logic unit is implemented in 64 16L8 programmable array logic chips (PALs) which performs the masked exclusive-or operation. The PAL programs are given in Appendix A.

2.4.8 The Bit Counter and Compare Logic

The resulting 256-bit word from the ALU is then passed to the Bit Counter. The number of bits in this word that are '1' is the Hamming distance between the Reference Address and the Hard Address.

The Bit Counter implements a 256-bit wide one-bit adder in four stages as represented functionally in Figures 2.3 and 2.4. Each stage is constructed from several 32KB EPROMs. The first stage consists of 17 identically programmed EPROMs. Each of the 15 EPROM address lines are connected to one of the 256 bits of the ALU output. The four bit output is a binary number representing the sum of the 15 bits. Since 15 x 17 = 255, there is one bit from the ALU that is not added, but carried into the next stage.

Now, the binary outputs of the first stage must be added to each other along with the 256th bit that was left over. In the second stage, 15 bits in the 1's position are added together to give a four bit partial sum hex number. Similarly, 15 bits in the 2's position, 4's position, and 8's position are added to give 4-bit binary number sums as shown in Figure 2.3. There still remain two 4-bit binary numbers remaining from stage

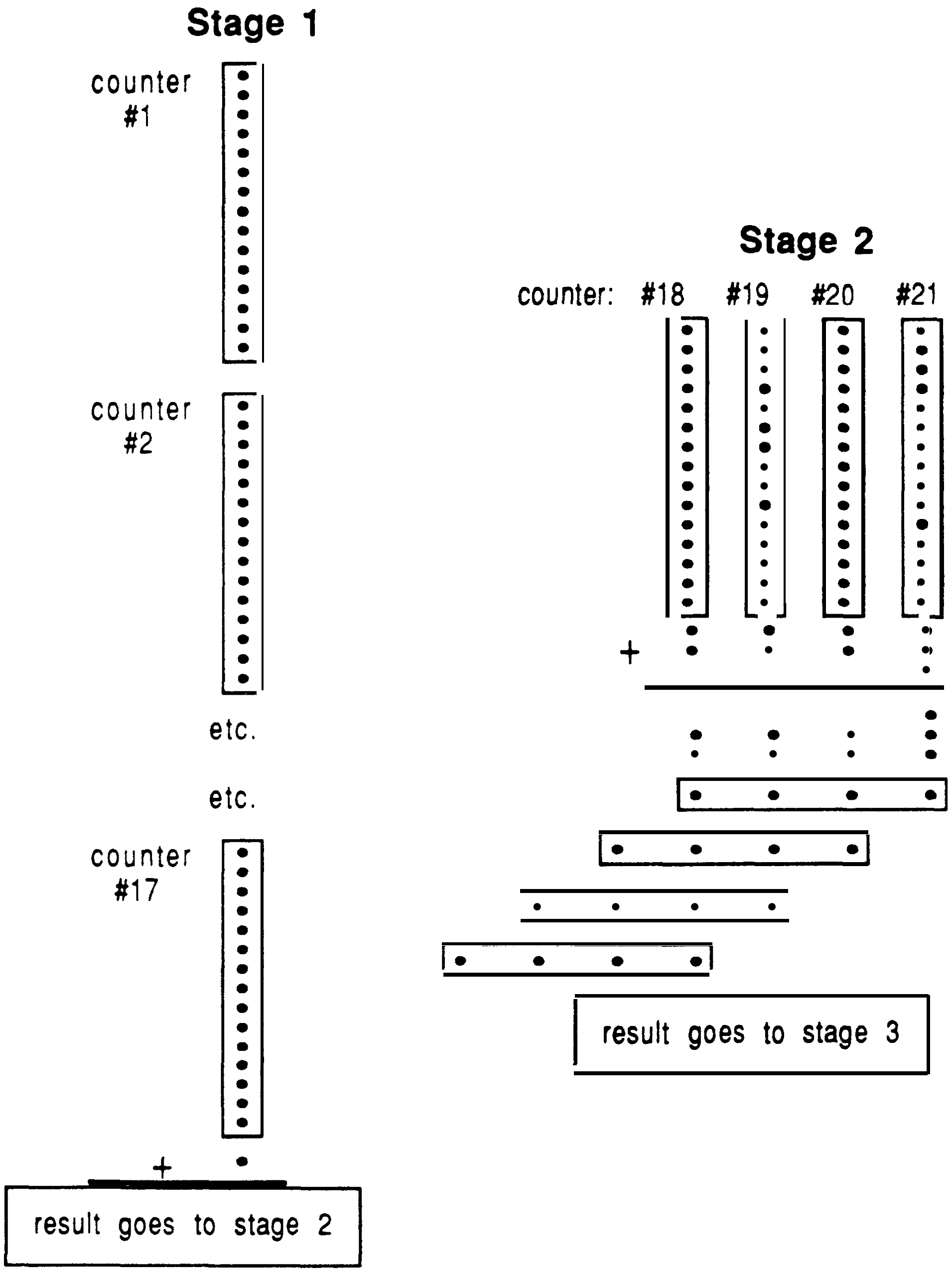


Figure 2.3: Stages 1 and 2 of the Bit Counter.

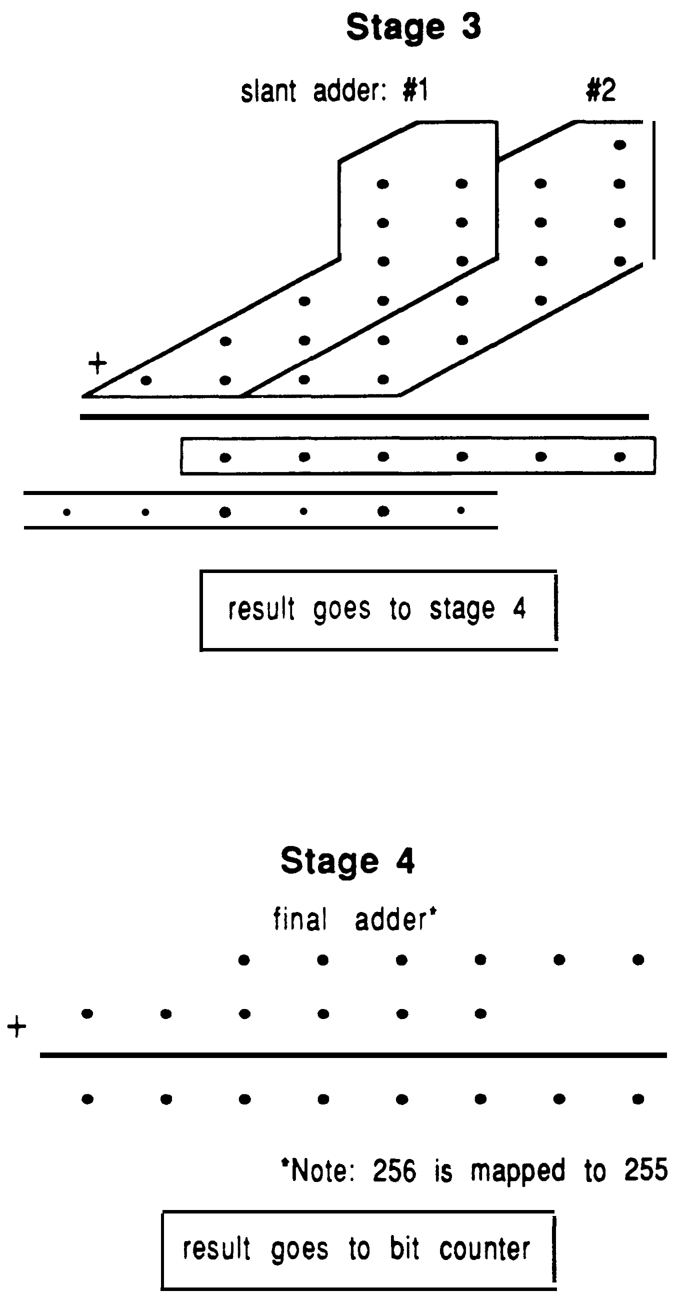


Figure 2.4: Stages 3 and 4 of the Bit Counter.

one and the 256th bit output of the ALU to be summed. Note that the operation, and thus the programming. of the EPROMS in stage two is identical to that of stage one.

The quantities to be added are the 256th bit from the ALU, the two 4-bit quantities from stage 2, and the four 4-bit quantities from stage 3. The quantities from stage 3 need to be shifted according to which place value they represent. The diagram in Figure 2.4 shows that this addition can be broken up into two identical additions of 12 and 13 bits, resulting in two 6-bit binary quantities as shown. The EPROMs in stage 3 are programmed to perform this addition.

Finally, in stage 4, the two 6-bit quantities are added to form a single 8-bit quantity in stage 4. Note that if a sum of 256 is produced, which would require 9 bits to represent, it is mapped to 255 which is represented by eight ones. This necessary anomaly was determined to have no practical effect on the usefulness of the Sparse Distributed Memory System.

This eight-bit result is then compared with the Limit Register. The comparison is done with two 74LS85 comparators. If the result is less than or equal to the limit, the Hit status bit is set causing the AM will enter the HIT state, storing the Tag and the Hamming distance into the Tag Cache.

2.4.9 The Tag Cache

The central components of the Tag Cache are a trio of dual-ported 256 byte static RAMs. The IDT7130 chip is the master dual port RAM while the two IDT7140 chips are the slave dual port RAMs. One RAM stores the Hamming distance, the other two store the Tag associated with the Hard Address. In operation, the internal write counter, implemented with a 74ALS867, points to the next available address in the DPRAM where the Tag and Hamming distance can be stored by the Address Module. Similarly, the internal read counter, also implemented with a 74ALS867, points to the the next location in the Tag Cache to be read by the Control Module. Thus the dual port RAMs and the read and write counters create a FIFO. When the AM enters the HIT state, the Tag and distance are written to the DPRAM after which the internal write counter is incremented. The EMPTY status bit is then cleared, signalling the Control Module that the Tag Cache has data to be read. When the CM reads the Tag Cache, the internal read counter is incremented. When the read and write counters are equal, the Tag Cache is empty and the EMPTY status bit is set. It is important that the CM read the Tag Cache *only* when the EMPTY bit is zero. Otherwise the operation of the AM will be unpredictable until it is once again reset by the CM or through powering off.

The Tag Cache outputs are buffered by 74LS244 buffers since the IDT7130 and IDT7140 do not have enough drive capability.

The Tag Cache PAL is used to synchronize the output of the read counter overflow signal to the Address Module clock. and then generate the FULL status bit, the EMPTY status bit, and a write strobe to the Tag Cache. For detailed information on this PAL, see appendix A.

BUSYR is asserted by the Dual Port SRAM whenever the two addresses are equal and the internal arbitration logic has given access to the right (write counter generated) address. Similarly, BUSYL is asserted when the left (read counter generated) address is given access. The important fact, though, is that when one of these signals is asserted, the two addresses are identical and thus the Tag Cache is empty.

When the write counter overflows, the Tag Cache is full until the read counter similarly overflows.

When the Address Module is in the HIT state, the Tag Cache Write Enable is asserted during the first half of the clock period.

2.4.10 Operation of the Address Module

The AM operates in two distinct modes. Before an application is run on the SDM system, specific Hard Addresses must be loaded into the AM and the various registers must be initialized. This in accomplished in the RESET state as shown in the State Transition Diagram, Figure 2.2.

Once set-up is complete, applications can be run on the system. Application processing begins when the Control Module takes the Address Module out of the RESET state and into the RUN state by clearing the FRST status bit in the Command/Status Register.

A single read or write for an application causes the AM to:

1. Execute the RUN state 8192 times. During each execution, the Reference Address is compared to one of the Hard Addresses, and the Hamming distance between them is calculated and compared to the value in the Limit Register.

2. Visit the HIT state once for every hit encountered during the RUN state.

3. Visit the WAIT state whenever the Tag cache gets filled.

4. Visit the DONE state when the entire contents of the Hard Address Memory have been examined.

Note that the WAIT state and DONE state are identical except for the value of the DONE status bit.

The procedure for running an application is described below.

Set up

1. The CM sets the FRST bit in the Command/Status Register, placing the AM in the RESET state.

2. The CM loads the Hard Address Memory. This is accomplished by first loading the Tag Register with the appropriate tag, and then loading the Hard Address Register associated with that tag. The Hard Address Register consists of eight 32 bit quantities which can be loaded in any order, forming a single 256 bit Hard Address.

3. The CM loads the other registers: the Mask Register, the Limit Register and the Reference Address Register. The order of access is not important. Also, values loaded into the registers and the Hard Address Memory will remain intact even after the AM has been placed in the RESET state. The registers and the Hard Address Memory will contain random data, however, after a power up.

Operating

1. The CM clears the RST bit in the Command/Status Register, and the state machine enters the RUN state.

2. During each RUN cycle, the next Hard Address in the Hard Address Memory (beginning at tag 0 and incrementing to tag 8191) is transferred to the ALU, to be XORed with the Reference Address. The result is then ANDed with the Mask Register and passed to the Bit Counter. The result propagates through all four stages of the Bit Counter and is then compared to the Limit Register. If the result is less than or equal to the Limit, the HIT status bit is set and the AM enters the HIT state. If there is no hit, the AM continues in the RUN state, examining the next Hard Address. When the last Hard Address has been examined, the AM enters the DONE state.

3. During a HIT cycle, the Hamming distance is sent to the Tag Cache, along with the Hard Address Tag. If the Tag Cache is then full, the AM enters the WAIT state. If the last Hard Address has been examined, the AM enters the DONE state. Otherwise the AM resumes execution in the RUN state.

4. During a WAIT cycle, the AM suspends operation and waits until the CM has read the last entry in the Tag Cache. At that point, if all Hard Addresses have been examined, the AM enters the DONE state. Otherwise the AM resumes execution in the RUN state.

5. During a DONE cycle, the AM asserts the DONE status bit and performs no operation until the CM sends it into the RESET state by setting the RST status bit, or by a power-on reset.

A complete block diagram of the Address Module may be found in Figure 2.1.

Table 2.2: Internal Buses.

Bus	Bits	Description
XD	32	Used to move data back and forth between the VME bus and the 28 Address Module registers.
REG	28	During a read or a write, only one of these lines will be high, signifying which register is being addressed.
TAG	13	Holds the index to the 8K of Hard Address Memory.
YD	256	Used to move a word from the Hard Address Memory to the ALU.
MR	256	Used to move the Mask Register contents to the ALU.
RAR	256	Used to move the Reference Address Register contents to the ALU.
ZD	256	This bus is the output of the ALU and the input to the Bit Counter. It has a one in each position in which the YD bus and the RAR bus differ (or, if the CMODE signal is asserted, the positions in which they are the same.)
Dist	8	Carries the result of the Bit Counter to the distance comparitor.
Limit	8	Carries the Limit Register (LR) output to the distance comparitor.
AMA	8	Output of the write counter. Used as the address for the Address Module's port of the Tag Cache.
CMA	8	Output of the read counter. Used as the address for the Control Module's port of the Tag Cache.

2.5 Internal Interfaces

There are a number of buses and control signals that are internal to the Address Module. A list of the internal buses is given in Table 2.2 and a list of control signals is given in Table 2.3.[1]

2.6 External Interface: The VME bus

The Address Module communicates with external components via a VME bus interface. The AM is set up to operate as a slave device that responds to data transfers in the address range of 0x00000000 to 0x0000007C, inclusive [2]. This corresponds to the space

[1] On the schematics, if a signal is suffixed with a dash, it is an active low signal. Otherwise, the signal is active high.

[2] The Address Module has been placed at the bottom of VME bus address space. However, it will not necessarily reside at the same location in the address space of the VME bus master device. For example, if the master device has allocated locations 0x04000000 to 0x08000000 for the VME bus, then it will find the AM at locations 0x04000000 to 0x0400007C. The addresses actually placed on the bus and sent to the AM, though, will be from 0x00000000 to 0x0000007C.

Table 2.3: Internal Control Signals.

Status/Control Register:	
CMODE	Makes the ALU use the complement of the RAR.
FORCE	Used to force the State Machine into a known state.
FST1	If force is asserted, this will become the new value of st1 (see below).
FST0	If force is asserted, this will become the new value of st0 (see below).
RST	Reset the Address Module.
State Machine:	
CLK	A 1 Mhz clock.
ST1	High state bit.
ST0	Low state bit.
TCNTEN	Enables the Hard Address tag counter which sequentially addresses the HAM.
ACNTEN	Enables the write counter (holds the address for the Address Module's port of the Tag Cache.)
CMTAG	Tells the Hard Address memory logic to index the HAM with the value stored in the Tag Register sent from the Control Module (reset mode only).
AMTAG	Tells the Hard Address memory logic to index the HAM with the value of the Hard Address tag counter (used when not in reset mode).
Read/Write Control:	
WE	Write enable. Trailing edge used to latch data from XD bus into the HAM.
RE	Read enable. Is asserted when it is okay to put data on the XD bus for the Control Module to read.
TCWE	Tag Cache write enable. Trailing edge used to write data into Tag Cache.
OK	Is asserted when the Address Module is being addressed.
R/W	Read/Write control signal from the VME bus.
Cache Status:	
FULL	Tag Cache is full. State Machine must enter wait state.
EMPTY	Tag Cache is empty. Control Module must wait until the Address Module writes the cache again or DONE is asserted.
DONE	Signals the Control Module that the entire Hard Address Memory has been searched.
HIT	The Hamming distance is less than or equal to the limit. The state machine goes into the Hit state and stores the distance and tag in the Tag Cache.

needed by 32 32-bit registers, though only 28 of them are actually used. Data is transferred in quad-byte (32-bit) quantities. Thus, all 32 bits are used on both the address and data buses (labeled bus A and bus D in the schematics, respectively.)

The only other VME bus signals used are the address strobe (AS), data strobes (DS0 and DS1, data acknowledge (DTACK), and the read/write signal (WRITE). The address strobe is used to latch in the address, while the data strobes are used to signal that valid data is on the VME bus. The AM asserts DTACK 1 micro-second after the data is valid. This was determined to be enough time for the data to be latched into the appropriate register.

Chapter 3

Hardware Specifications and Power Consumption

The Address Module consists of a 400mm 9U sized VME card with two standard 96 pin VME connectors. Table 3.1 shows a list of the components along with the current drawn by each.

The total maximum power was found with the following formula:

$$TOTAL = \Sigma\text{TYP} + \sqrt{\Sigma(\text{MAX-TYP})^2}$$

Since each part actually belongs to a distribution, the typical power is the mean for the distribution while the maximum is usually three sigma above the mean. The sum of the typical numbers will give the typical power of the board. The formula above gives the power at three sigma above the mean for the entire board. Essentially, since the board is composed of many parts, the chance that all are operating at maximum power is very small. Also notice that the more parts that comprise the board, the more likely it is that the typical power will be the actual operating power. Thus for more parts, the maximum number approaches the typical number.

Table 3.1: Parts listing and power consumption.

		mA PER PART		mA TOTAL	
PART	QTY	TYP	MAX	TYP	MAX
16L8D	65	120	180	7800	8284
16R4D	2	120	180	240	325
27256 (EPROM's)	24	—	30	—	720
74AS867	4	100	160	400	520
74F02	1	i	13	i	13
74F244	3	60	90	180	232
74F245	4	100	125	400	450
74HC174	1	1	4	1	4
74LS00	1	2	4	2	4
74LS02	6	3	5	18	22
74LS04	2	4	7	8	12
74LS27	1	4	7	4	7
74LS32	3	5	10	15	24
74LS74	1	4	8	4	8
74LS85	2	10	20	20	34
74LS125	1	10	20	11	20
74LS154	2	9	14	18	22
74LS244	36	32	54	1152	1284
74LS373	68	24	40	1632	1764
IDT7130	1	65	180	65	180
IDT7140	2	65	180	130	293
MS6264L-70	32	—	55	—	1760
TOTAL (mA)				14,587	15,167
TOTAL (Watts)				72.9	75.8

Chapter 4

Design Notes

This chapter comments on the current design, pointing out aspects that should be improved on subsequent designs.

1. The board address from the VME bus is qualified with the data strobes DS0 and DS1. While this should work in most cases, the VME bus specifications do leave open the possibility that the address can become invalid before the data strobes are deasserted due to skew on the backplane. This would result in incorrect registers being accessed for a short time. To avoid this problem in the future, the register selects should be latched on the falling edge of the data strobes.

2. According to the VME bus specifications, slave boards must not respond to reserved address modifier codes. The current board does not observe the address modifier codes. For the current design this causes no problem but it does limit the expandability of the system and puts certain restrictions on the address space allowed for other boards in the system.

3. Also according to the VME bus specifications, slave boards must not respond to data transfer cycles when IACK is asserted. The current design does not look at IACK. Again this causes no problem in the current system but may put limitations on future systems using standard VME bus masters.

4. In the current design, the ALU, bit counter, and limit register comparison all occur sequentially. A pipelined design would allow the board to process data much faster although increasing the number of components and complexity of the design. This option should be explored.

Appendix A

PAL Programs

Programmable Array Logic (PAL) chips are used in four places in the design of the Address Module:

- The Finite State Machine
- The Arithmetic and Logical Unit (ALU)
- Control of Tag Cache Signals
- The Address Half of the Bus Interface

All of the PAL's were programmed with a SD 20/24 PAL Development System. The programs used to burn the PALs are listed in the following sections.

A.1 Finite State Machine PAL

```
PAL16R4
P5
FSMPAL
STANFORD UNIVERSITY

CLK FSO FS1 RST F DONE /HIT /AMCO /FULL CND
CND /AMT /CMT SO S1 RST1 RST2 /CTEN /QHIT VCC

AMT = S1 + so
CUT = /S1*/SO
CTEN = /S1*SO*/DONE*/HIT + S1*/SO*/DONE + RST2
QHIT = S1*/SO
/SO := F*/FSO + RST + RST2 + /F*/S1*SO*/RST*/DONE*HIT
```

```
/S1 := F*/FS1 + RST + RST2 + /F*/SO*/S1*/RST2 +
                             /F*/S1*SO*/RST*/HIT*/DONE +
                             /F*S1*/SO*/RST*AMCO +
                             /F*S1*SO*/RST*/FULL*/DONE

/RST1 := /RST
/RST2 := /RST1
```

DESCRIPTION:

A.2 ALU PAL

```
PAL16L8
P1
ALUPAL
STANFORD UNIVERSITY

R2 R3 YO Y1 Y2 Y3 MO MI M2 GND M3 NC RO NT R1 ZO Z1 Z2 Z3 VCC

/ZO = /MO + /RO*/YO*/NT + RO*YO*/NT + RO*/YO*NT + /RO*YO*NT
/Z1 = /MI + /R1*/Y1*/NT + R1*Y1*/NT + R1*/Y1*NT + /R1*Y1*NT
/Z2 = /M2 + /R2*/Y2*/NT + R2*Y2*/NT + R2*/Y2*NT + /R2*Y2*NT
/Z3 = /M3 + /R3*/Y3*/NT + R3*Y3*/NT + R3*/Y3*NT + /R3*Y3*NT
```

DESCRIPTION:

A.3 Tag Cache PAL

```
PAL16R4
P5
TAGPAL
STANFORD UNIVERSITY

CLK CLK2 so sl /AMCO /CMCO /RESET /BUSYL /BUSYR /GND
CND /TCWE /SIGNAL NC /FULL X /CMCO2 /EMPTYO /EMPTY VCC

CMC02 :=CMCO
FULL := /FULL*AMCO + FULL*/X*/CMCO2*/RESET + FULL*/X*CMCO2*/RESET
        + FULL*X*CMCO2*/RESET
/X := /FULL + /CMCO2
EMPTY0 = BUSYL*/FULL + BUSYR*/FULL + FULL*/X*CMCO2 + FULL*X*/CMCO
/EMPTY = /EMPTYO
```

```
SIGNAL = S1*/S0*/CLK2 + SIGNAL*S1*/S0 + SIGNAL*CLK2
TCWE = S1*/S0*/SIGNAL

DESCRIPTION:
```

A.4 Bus Interface (Address) PAL

```
PAL16L8
P4
ADDRPAL
STANFORD UNIVERSITY

A31 A30 A29 A28 A27 A26 A25 A24 A23 GND
A22 AM A21 A20 A19 Al8 /INT1 /INT2 NAM VCC

INT1 = /A31*/A30*/A29*/A28*/A27*/A26*/A25
INT2 = /A24*/A23*/A22*/A21*/A20*/A19*/A18
/AM = /INT1 + /INT2
/NAM = INT1*INT2

DESCRIPTION:
```

Appendix B

EPROM Programs

The Bit Counter is implemented in four stages of Erasable Programmable Read Only Memory (EPROM) chips. The first two stages consists of 17 and 4 15-bit counters, respectively. The program to generate the data file for these counters is in 'count.c'. The third stage consists of 2 adders, which are generated by the file 'slant.c'. The final stage is a single adder, generated by 'adder.c'. All of the EPROMs were programmed with the EPRO System 4000 Programmer I attached to an IBM PC-XT.

B.1 Stages One and Two

```
/* count.c */

/*******************************************************************/
/*                                                                 */
/* This file will generate the data for the first two stages       */
/* of the eprom bit counter.  It is a (15,4) counter.              */
/*                                                                 */
/*      To generate the file for the EPRO programmer, compile     */
/*      the program and type:                                      */
/*                                                                 */
/*            count > count .dat                                   */
/*                                                                 */
/*      The output file count.dat can then be down-loaded to       */
/* the programmer.                                                 */
/*                                                                 */
/*******************************************************************/
/* Regarding the Intel MCS-86 hex format:                          */
/*                                                                 */
/* There are four record types:                                    */
/*      00      data record                                  */
/*      01      end record (end of file)                     */
/*      02      paragraph address (16 bytes, defaults to 0)  */
/*      03      start record (ignored)                       */
/*                                                           */
/*                                                           */
/* The format is as follows:                                 */
/*                                                           */
/*      :BCAAAARTDDDDDDDDDDDDDDDDDDDDDDCC<CR><LF>            */
/*                                                           */
/*      BC      number of data bytes in the record           */
/*      AAAA    offset address for this record               */
/*      RT      record type (see above)                      */
/*      DD      ascii coded hex data bytes                   */
/*      cc      twos complement of the sum of all the bytes  */
/*                  <this is the checksum; adding all the    */
/*                   bytes with CC produces a zero result>   */
/*                                                           */
/*************************************************************/

#define N 15                /* # of input bits */
#define M 8                 /* # of output bits */

short input [N];
/*          a0  al a2 a3 a4 a5 a6 a7 a8 a9 a10 all al2 al3 al4 */
int w[N] = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  1,  1,  1,  1}; /* weights
*/
char line[81];
int aa = 0;                 /* offset address (used by 'format') */

int increment ()
{
    int i, c_in = 1, c-out;

    for(i = 0; i < N; i++) {
        c_out = (c_in + input[i]) / 2;
        input [i] = (input[i] + c_in) % 2 ;
        c_in = c-out;           /* c_in for next iteration is c_out */
    }
    return(c_out);
}
```

```
itox(val, high, low)
int val;
char *high, *low;
{
    int i;

    i = val / 16;
    if(i < 10)
        *high = (i + '0');
    else
        *high = (i - 10 + 'A');

    i = val % 16;
    if(i < 10)
        *low = (i + '0');
    else
        *low = (i - 10 + 'A');
}


format(bc, bs)
int bc, bs;
{
    char high, low;
    int cc;                 /* twos complement of the sum of all the bytes
*/

    itox(bc, &high, &low);
    line[1] = high;
    line[2] = low;          /* byte count stored in hex */

    itox(aa/256, &high, &low);
    line[3] = high;
    line[4] = low;
    itox(aa%256, &high, &low);
    line[5] = high;
    line[6] = low;          /* store the effective address */

    bs = (bs + bc + aa/256 + aa%256) % 256;      /* sum of all bytes */
    cc = (256 - bs) % 256;  /* checksum of all the bytes */
    itox(cc, &high, &low);
    line[9 + 2*bc] = high;
    line[10 + 2*bc] = low;  /* store the checksum */
    line[11 + 2*bc] = '\0'; /* terminate the line with a null character */

    aa += bc;               /* increment the effective address for next time*/
}



main0
{
    int bc = 0,             /* byte count */
        bs = 0;             /* sum of all the bytes */

    int i, sum;
    char high, low;

    line[0] = ':';              /* for formatting purposes */
    line[7] = '0';
    line[8] = '0';              /* record type = 0 (data record) */

    printf(":020000020000FC\n");    /* paragraph address */
    do {
        sum = 0;
        for(i = 0; i < N; i++)
            sum += input[i] * w[i];
        if (sum == 256)
            sum = 255;
        itox(sum, &high, &low);
        line[9  + 2*bc] = high;
        line[10 + 2*bc] = low;

        ++bc;
        bs = (bs + sum) % 256;

        if (bc == 16) {
            format(bc, bs);
            printf("%s\n",line);
            bc = 0;
            bs = 0;
        }

    } while (increment0 == 0);
    if(bc) {                    /* some bytes have not been stored */
        format(bc, bs);
        printf ("%s\n",line);
    3
    printf (":00000001FF\n");    /* end of record */
}
```

B.2 Stage Three

The only difference between this program and the one for the first two stages is the file header and the initialization of the weights by which the input is to be multiplied. Thus, only these changes need to be shown here:

```
/* slant.c */

/*********************************************************************/
/*                                                                   */
/* This file will generate the data for the third stage eprom       */
/* of the bit counter.  It is a (1,2,2,4,4,6) counter.               */
/*                                                                   */
/*     To generate the file for the EPRO programmer, compile        */
/*     the program and type:                                         */
/*                                                                   */
/*          slant > slant.dat                                        */
/*                                                                   */
/*     The output file slant.dat can then be down-loaded to          */
/* the programmer.                                                   */
/*                                                                   */
/*********************************************************************/

/*          a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 a10 a11 a12 a13 a14 */
int w[N] = {1, 1, 2, 1, 2, 1, 2, 2, 4, 4,  8,  8, 16,  0,  0};   /* weights
*/
```

B.3 Stage Four

The only difference between this program and the one for the first two stages is the file header and the initialization of the weights by which the input is to be multiplied. Thus, only these changes need to be shown here:

```
/* adder.c */

/*******************************************************************/
/*                                                                 */
/* This file will generate the data for the fourth stage eprom    */
/* of the bit counter.  It is a (1,1,2,2,2,2,1,1,8) counter        */
/* with greater than 256 mapped to 255.                            */
/*                                                                 */
/*      To generate the file for the EPRO programmer, compile     */
/*      the program and type:                                      */
/*                                                                 */
/*              adder > adder.dat                                  */
/*                                                                 */
/*      The output file adder.dat can then be down-loaded to       */
/* the programmer.                                                 */
/*                                                                 */
/*******************************************************************/

/*            a0 al a2 a3 a4 a5 a6 a7 a8 a9  a10 all al2 al3 al4 */
int w[N] = {1, 2, 4, 8, 16, 32, 4, 8, 16, 32, 64,  128,  0,  0,  0};/*weights
*/
```

Appendix C

Critical Design Considerations

This appendix provides an analysis of critical considerations that affected the design of the Address Module.

C.1 Critical Timing Path

The maximum frequency of the Address Module is determined by the path when it is in the RUN mode and evaluating whether a particular Hard Address is a hit. In this case, after the clock, the Tag Counter of the Clock/Sequencer section advances and addresses the Hard Address Memory. The Static RAMs in the Hard Address Memory must output a Hard Address which passes through the ALU to the Bit Counter. The Bit Counter EPROMs produce a sum of bits which is compared to the Limit Register to produce a hit (or the absence of a hit). The hit signal then goes to the state machine PAL to determine the next state. The flow through each chip and the worst case timing are given below. Locations of chips are given with respect to the page of the schematics (see Appendix D).

Part type	Location	Parameter	Function	Time(ns)
74AS867	p. 34	CLK to Q	Tag Counter output	15.0
74F244	p. 9	A to Y	HAM address buffer	6.2
8Kx8 SRAM	pp. 1–8	Addr access	Hard Address Memory	70.0
PAL16L8D	pp. 20–27	in to out	ALU	10.0
27256 EPROM	pp. 28–29	Addr access	Bit Counter stage 1	200.0
27256 EPROM	p. 30	Addr access	Bit Counter stage 2	200.0
27256 EPROM	p. 30	Addr access	Bit Counter stage 3	200.0
27256 EPROM	p. 30	Addr access	Bit Counter stage 4	200.0
74F85	p. 31	data nput to A=B output	Limit Register compare	12.0
74F85	p. 31	A=B input to A=B output	Limit Register compare	6.5
PAL16R4D	p. 34	register setup	State Machine	10.0
TOTAL				929.7

Thus, the maximum clock frequency of the current design of the Address Module is 1 MHz, which implies a 1000 nanosecond cycle time.

C.2 Tag Cache Write Enable

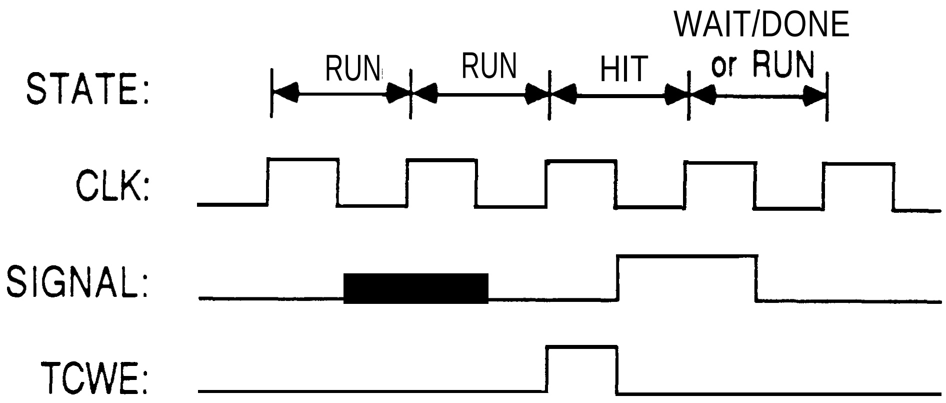
The Tag Cache is written in the middle of the HIT state. This is done by creating a signal in the Tag Cache PAL (see Appendix A) called SIGNAL. The timing and equations for SIGNAL and Tag Cache Write Enable (TCWE) are shown in Figure C.1 (note that these signals are discussed as active high signals when in fact the PAL produces active low signals).

SIGNAL is asserted during the second half of the HIT state through the first half of the next state. The third term in the equation for SIGNAL assures that there is not a glitch during the state transition. TCWE is asserted during the first half of the HIT state and deasserted during the second half. Data is written into the Tag Cache when TCWE is deasserted. See page 32 of the schematics for reference.

C.3 FULL status bit State Machine

The state diagram for the FULL status bit is given in Figure C.2 The equations for FULL and X are given in the Tag Cache PAL in Appendix A. AMCO is the Address Module Counter (Write Counter) Overflow signal. CMC02 is the synchronized Control Module Counter (Read Counter) Overflow signal. See page 32 of the schematics for reference.

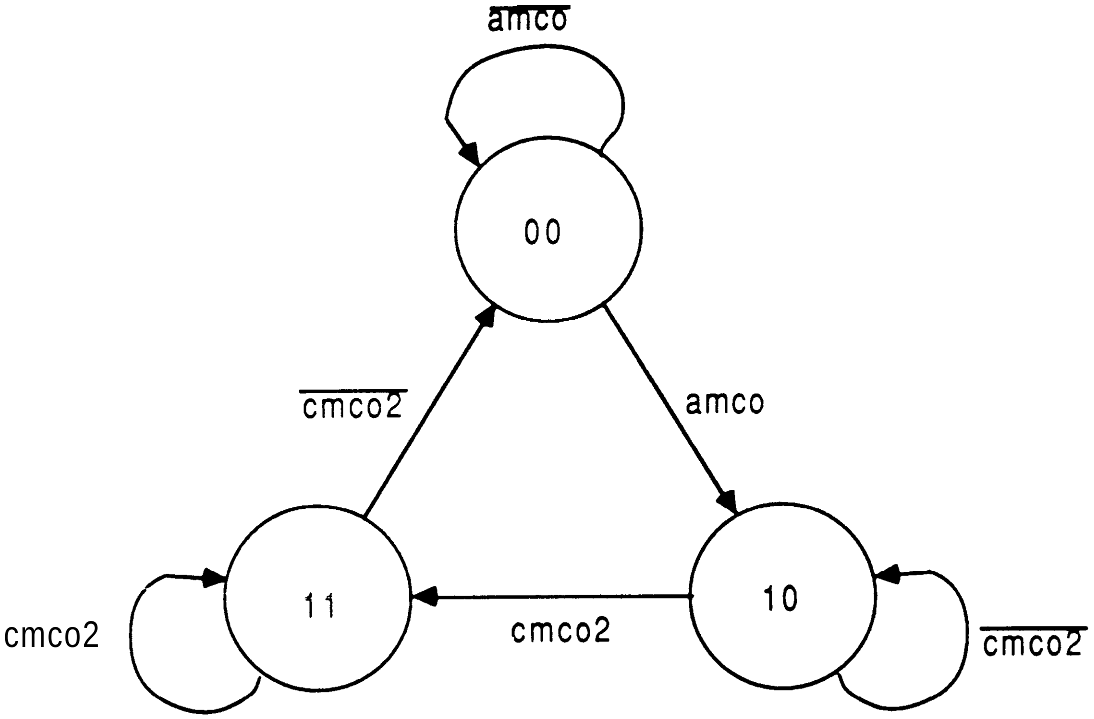
Initially the state machine is in state 0. When the Address Module Counter overflows, implying that the Tag Cache has been written with 256 entries, the FULL status



$$\begin{aligned} \text{signal} &= s1*\overline{s0}*\overline{clk} \\ &\quad +signal*clk \\ &\quad +signal*s1*\overline{s0} \end{aligned}$$

$$\text{tcwe} = s1*\overline{s0}*\overline{signal}$$

Figure C.1: SIGNAL and TCWE timing.



The state bits are FULL and X, respectively, in TAG CACHE PAL.

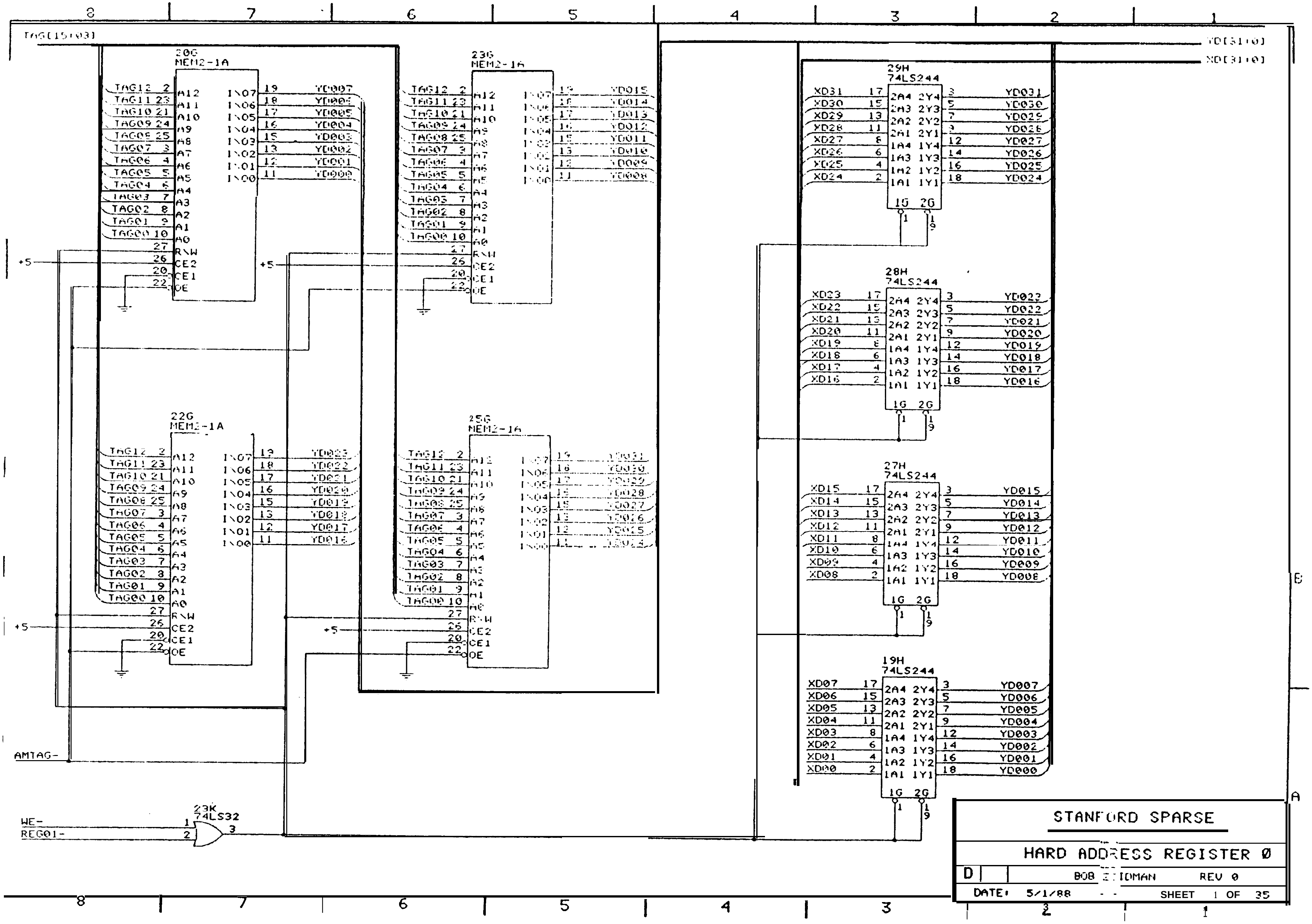
Figure C.2: State diagram for the FULL status bit.

bit is set. The state machine enters state 2 and waits for the Control Module Counter to overflow. implying that the last entry in the Tag Cache is being read. It then enters state 3 and waits for CMCO2 to be deasserted, implying that the last entry has been read and the counter has been reset to zero. The state machine then enters state 0, clearing the FULL status bit.

Appendix D

Sparse Address Module Schematics

Schematic diagrams and wire lists were created with the SCHEMA II Schematic Processing Program from Omation. All 35 schematic pages are contained in this appendix. Note that if a signal is suffixed with a dash, it is an active low signal. Otherwise, the signal is active high.



STANFORD SPARSE

HARD ADDRESS REGISTER Ø

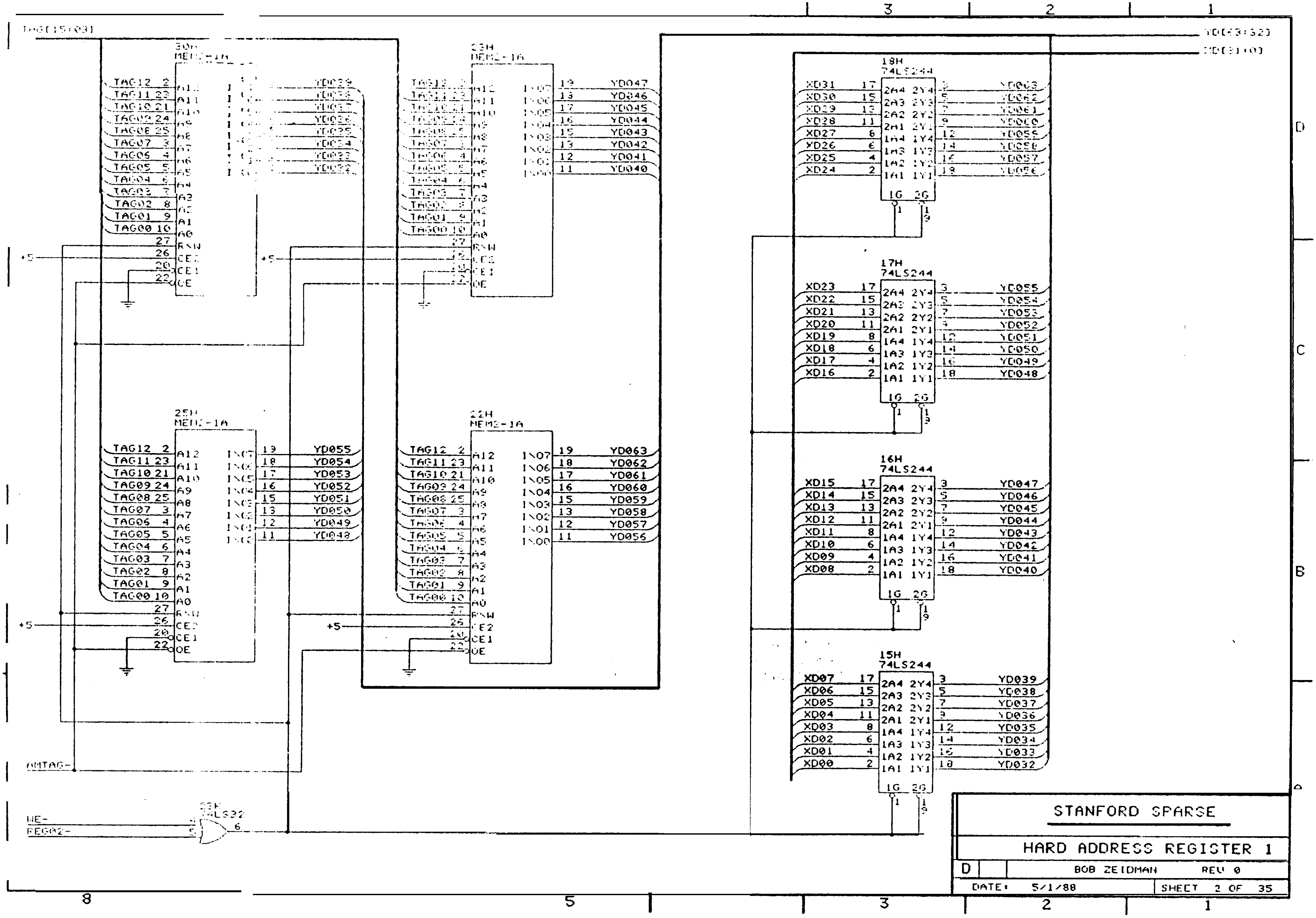
D

BOB ZEIDMAN

REV 0

DATE: 5/1/88

SHEET 1 OF 35

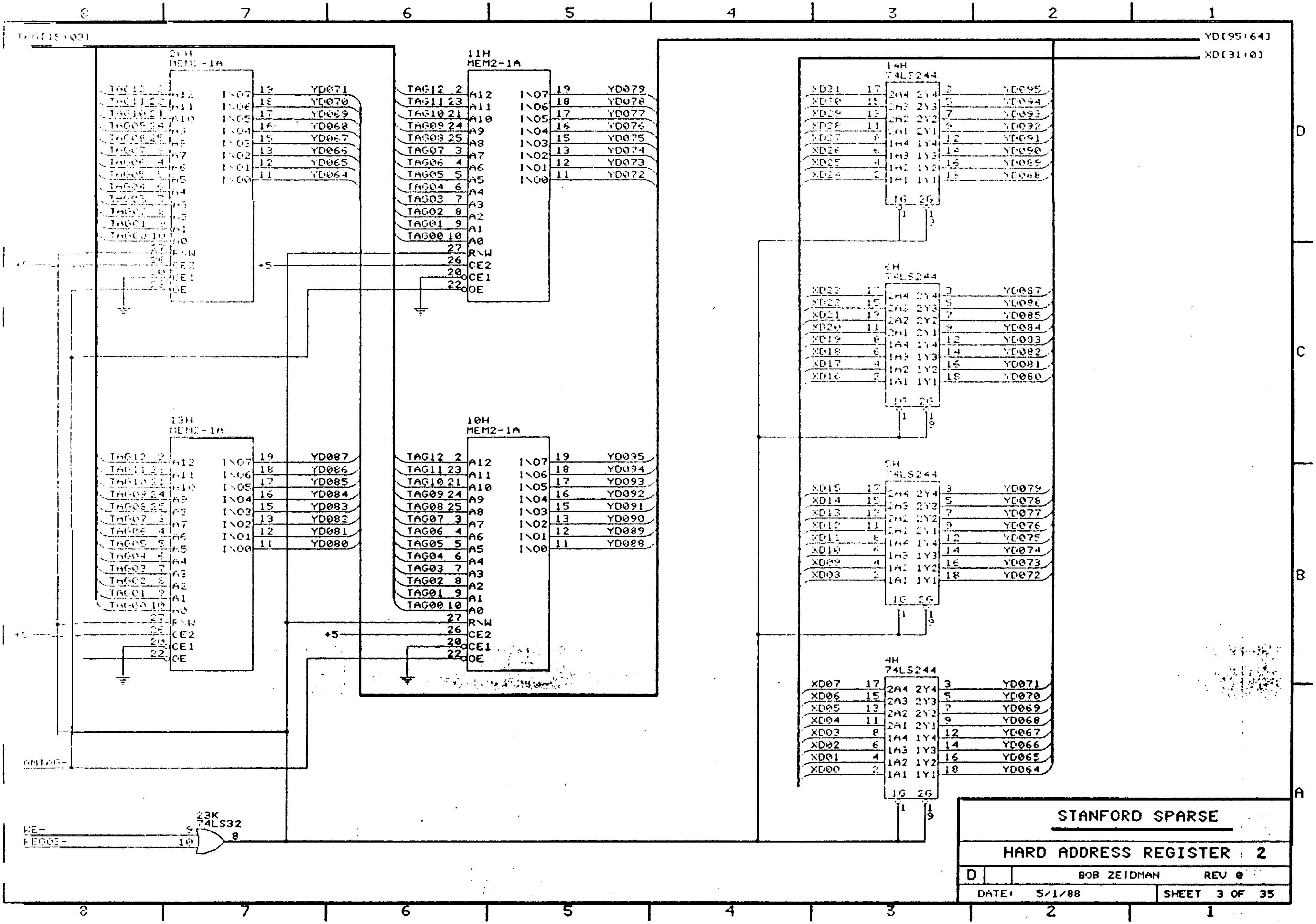


STANFORD SPARSE

HARD ADDRESS REGISTER 1

D BOB ZEIDMAN REV 0

DATE: 5/1/88 SHEET 2 OF 35

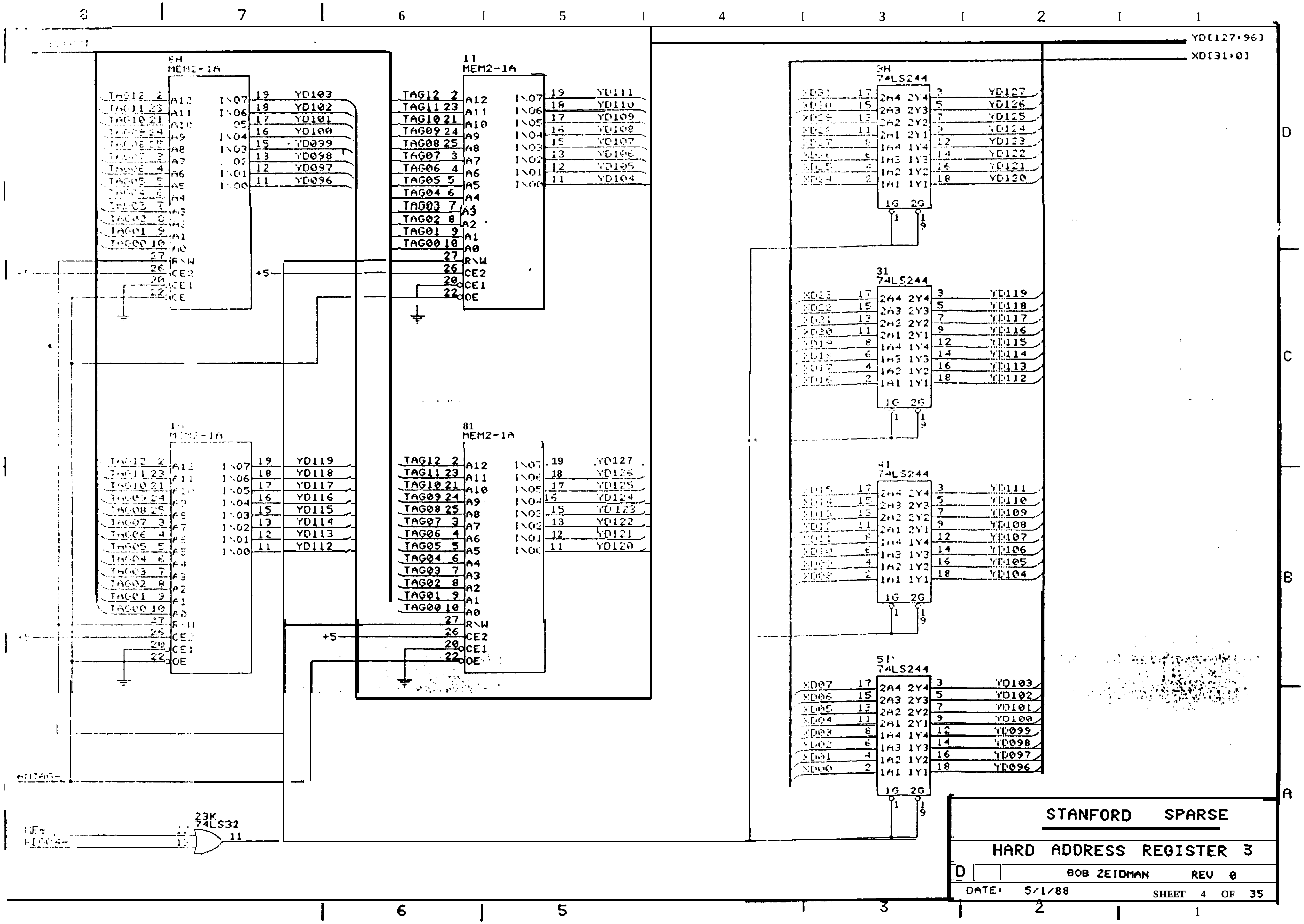

TAG[15:00]
YD[95:64]
XD[31:0]
20H MEM2-1A
11H MEM2-1A
13H MEM2-1A
10H MEM2-1A
14H 74LS244
6H 74LS244
5H 74LS244
4H 74LS244
23K 74LS32
A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
R\W CE2 CE1 OE
I\O7 I\O6 I\O5 I\O4 I\O3 I\O2 I\O1 I\O0
TAG12 TAG11 TAG10 TAG09 TAG08 TAG07 TAG06 TAG05 TAG04 TAG03 TAG02 TAG01 TAG00
YD071 YD070 YD069 YD068 YD067 YD066 YD065 YD064
YD079 YD078 YD077 YD076 YD075 YD074 YD073 YD072
YD087 YD086 YD085 YD084 YD083 YD082 YD081 YD080
YD095 YD094 YD093 YD092 YD091 YD090 YD089 YD088
XD31 XD30 XD29 XD28 XD27 XD26 XD25 XD24
XD23 XD22 XD21 XD20 XD19 XD18 XD17 XD16
XD15 XD14 XD13 XD12 XD11 XD10 XD09 XD08
XD07 XD06 XD05 XD04 XD03 XD02 XD01 XD00
2A4 2A3 2A2 2A1 1A4 1A3 1A2 1A1
2Y4 2Y3 2Y2 2Y1 1Y4 1Y3 1Y2 1Y1
1G 2G
+5
AMTAG-
WE-
REG03-
8 7 6 5 4 3 2 1
D C B A

STANFORD SPARSE

HARD ADDRESS REGISTER 2

D BOB ZEIDMAN REV 0

DATE: 5/1/88 SHEET 3 OF 35



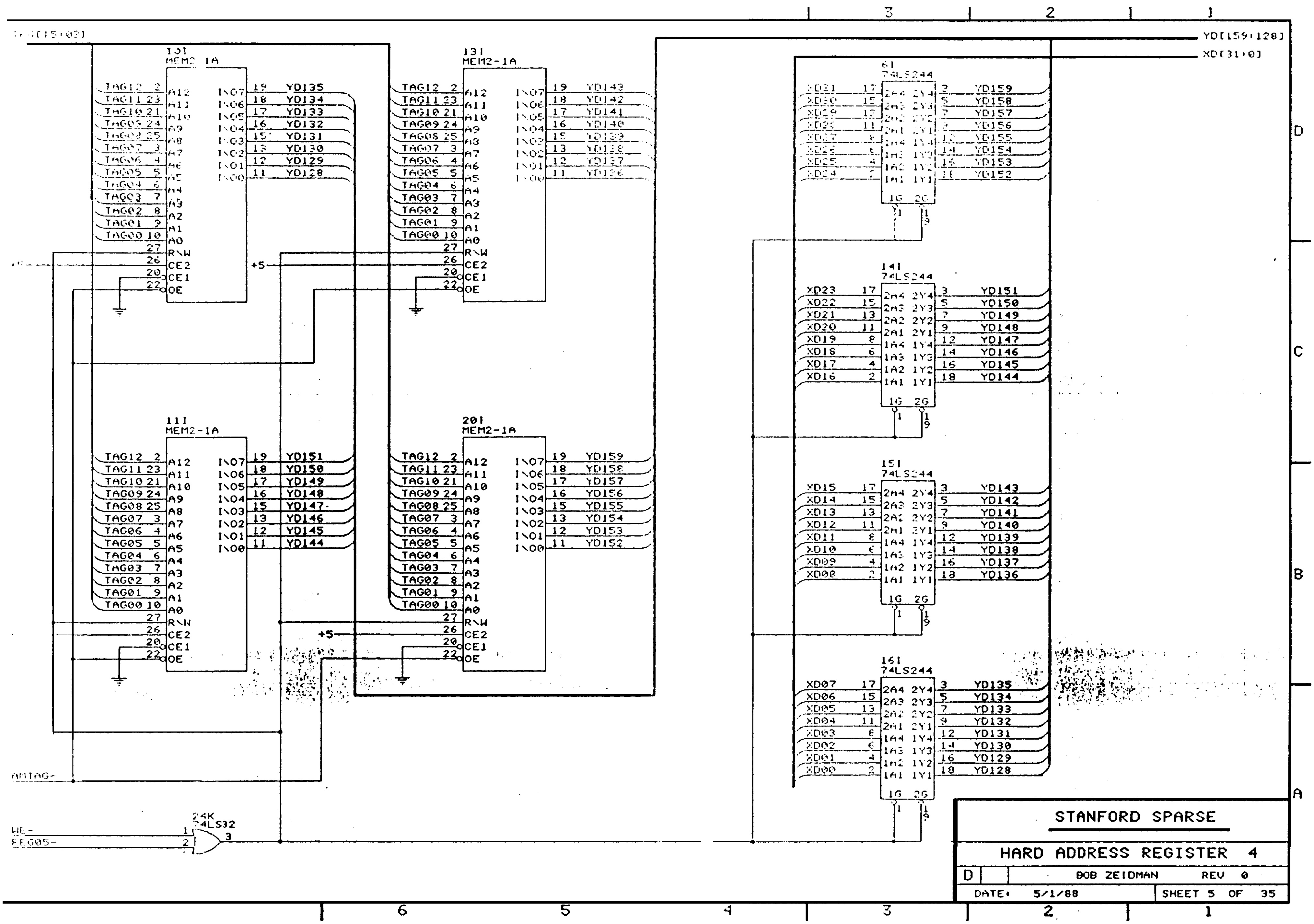
STANFORD SPARSE

HARD ADDRESS REGISTER 3

D

BOB ZEIDMAN REV 0

DATE: 5/1/88 SHEET 4 OF 35

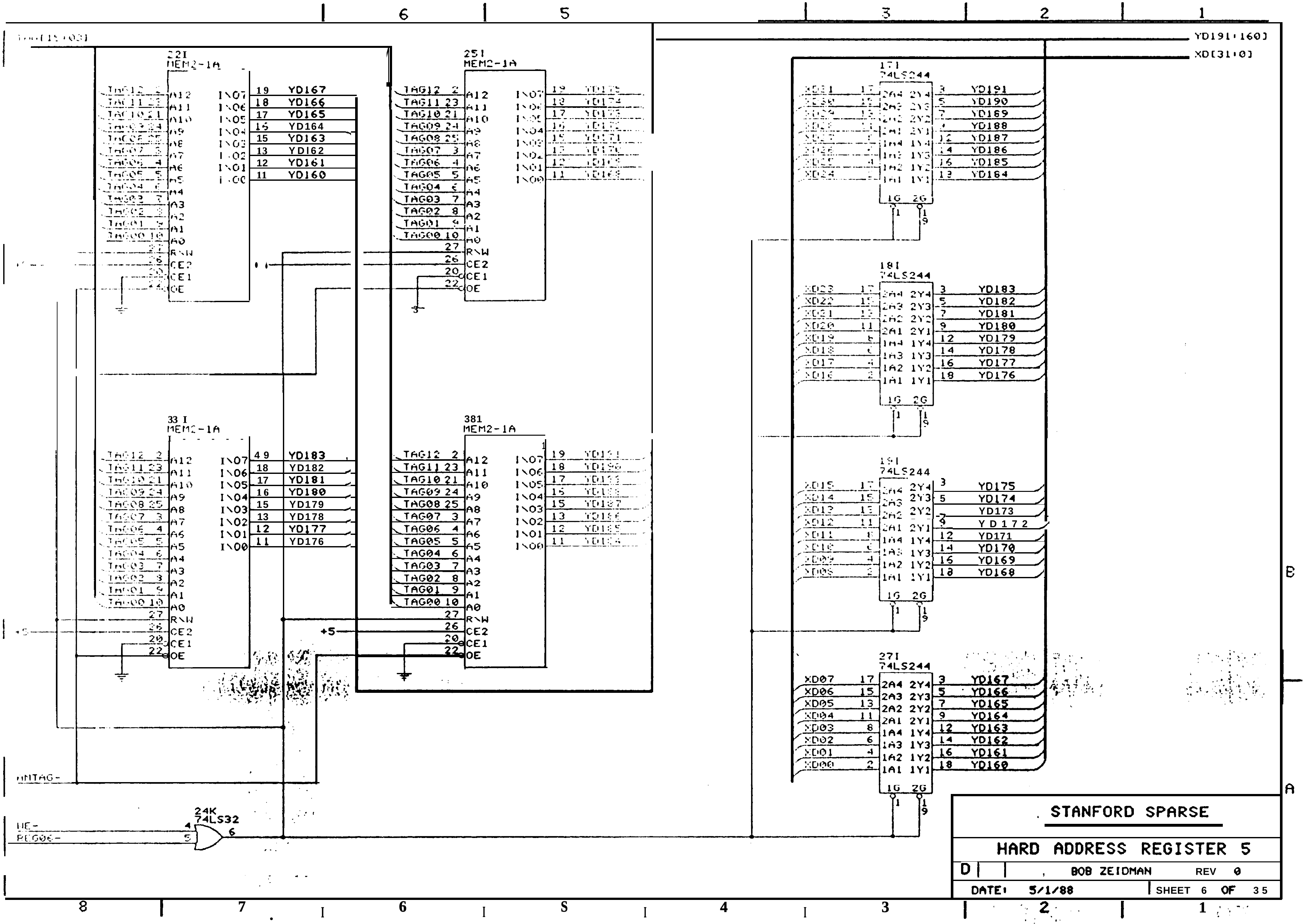


STANFORD SPARSE

HARD ADDRESS REGISTER 4

D | | BOB ZEIDMAN | REV 0

DATE: 5/1/88 | SHEET 5 OF 35

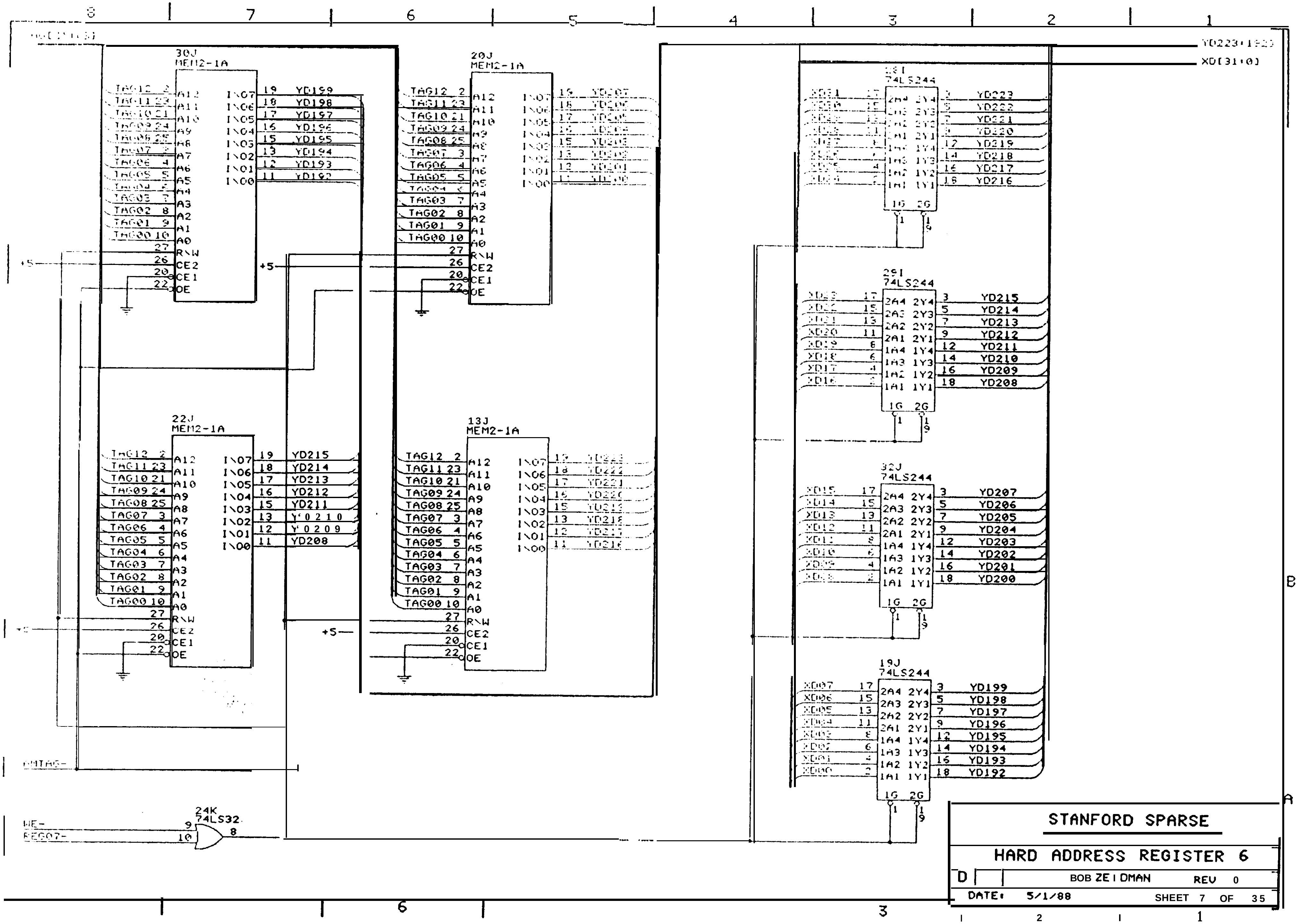


STANFORD SPARSE

HARD ADDRESS REGISTER 5

D | | , BOB ZEIDMAN REV 0

DATE: 5/1/88 | SHEET 6 OF 35



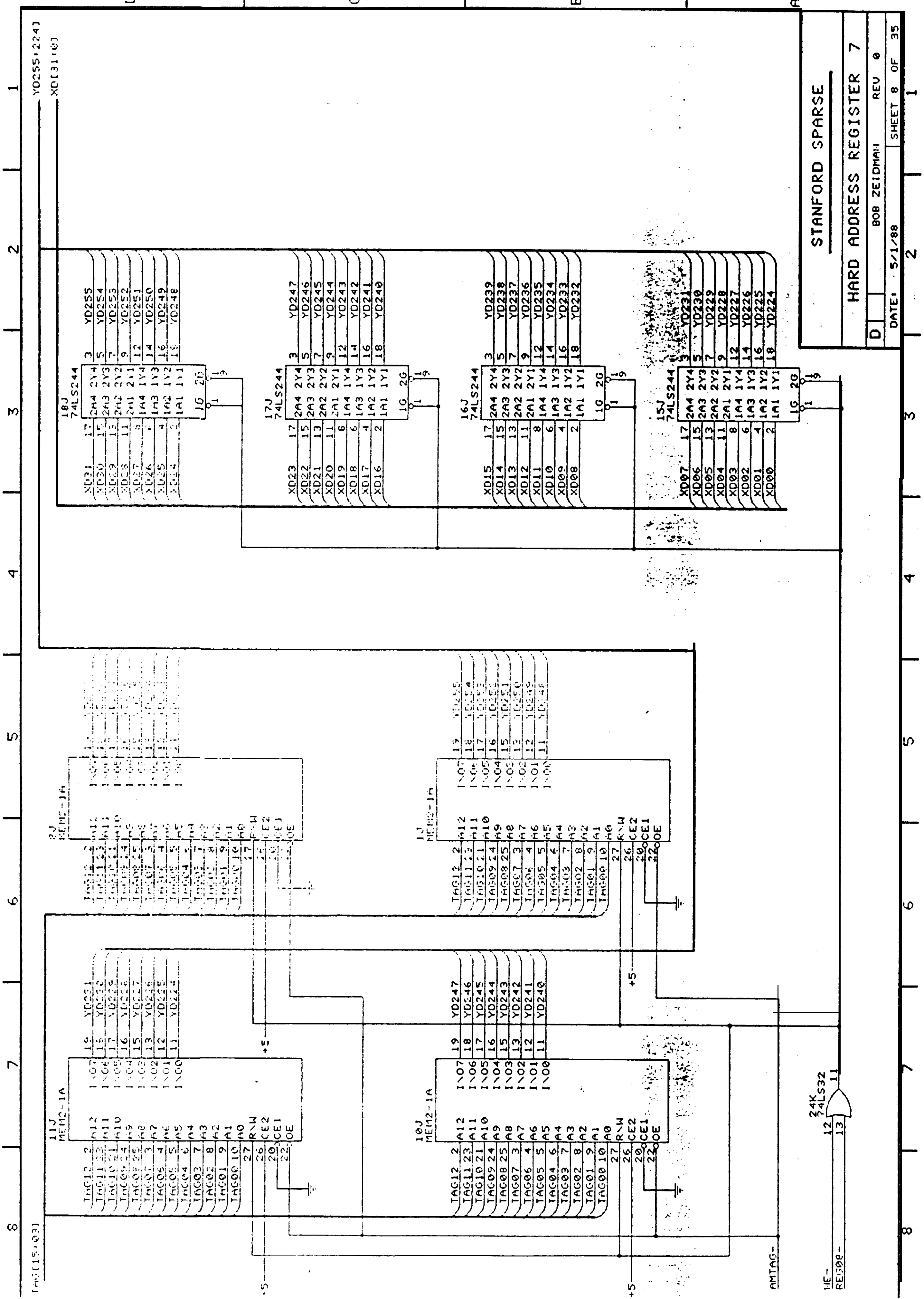
30J MEM2-1A
20J MEM2-1A
22J MEM2-1A
13J MEM2-1A
18J 74LS244
29J 74LS244
32J 74LS244
19J 74LS244
24K 74LS32
TAG12 TAG11 TAG10 TAG09 TAG08 TAG07 TAG06 TAG05 TAG04 TAG03 TAG02 TAG01 TAG00
A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
R\W CE2 CE1 OE
I\O7 I\O6 I\O5 I\O4 I\O3 I\O2 I\O1 I\O0
YD199 YD198 YD197 YD196 YD195 YD194 YD193 YD192
YD207 YD206 YD205 YD204 YD203 YD202 YD201 YD200
YD215 YD214 YD213 YD212 YD211 YD210 YD209 YD208
YD223 YD222 YD221 YD220 YD219 YD218 YD217 YD216
XD31 XD30 XD29 XD28 XD27 XD26 XD25 XD24
XD23 XD22 XD21 XD20 XD19 XD18 XD17 XD16
XD15 XD14 XD13 XD12 XD11 XD10 XD09 XD08
XD07 XD06 XD05 XD04 XD03 XD02 XD01 XD00
2A4 2A3 2A2 2A1 1A4 1A3 1A2 1A1
2Y4 2Y3 2Y2 2Y1 1Y4 1Y3 1Y2 1Y1
1G 2G
YD223:192
XD[31:0]
+5
OMTAG-
WE-
REG07-
8 7 6 5 4 3 2 1
A B

STANFORD SPARSE

HARD ADDRESS REGISTER 6

D | BOB ZEIDMAN | REV 0

DATE: 5/1/88 | SHEET 7 OF 35



STANFORD SPARSE

HARD ADDRESS REGISTER 7

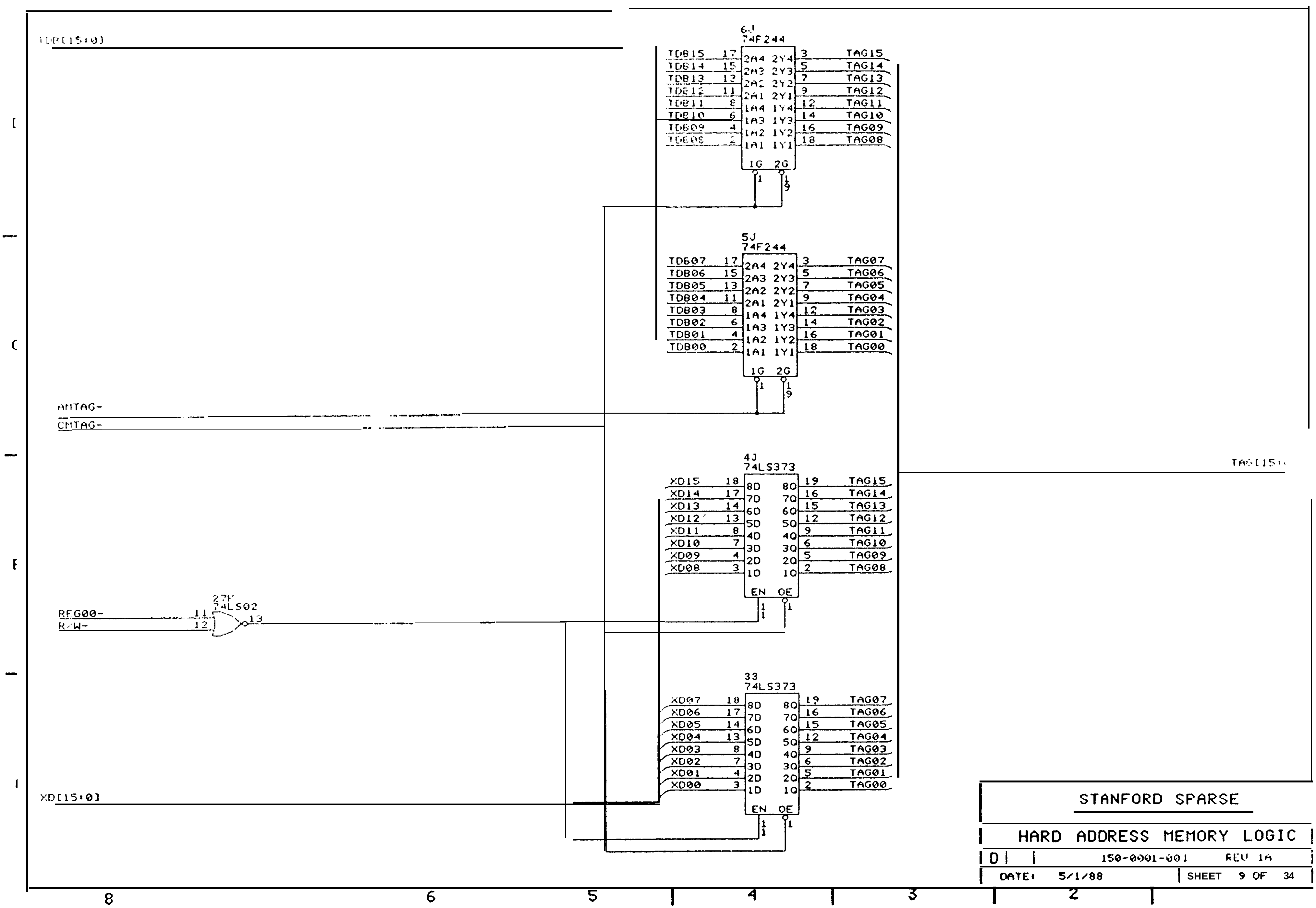
BOB ZEIDMAN

REV 0

D

DATE: 5/1/88

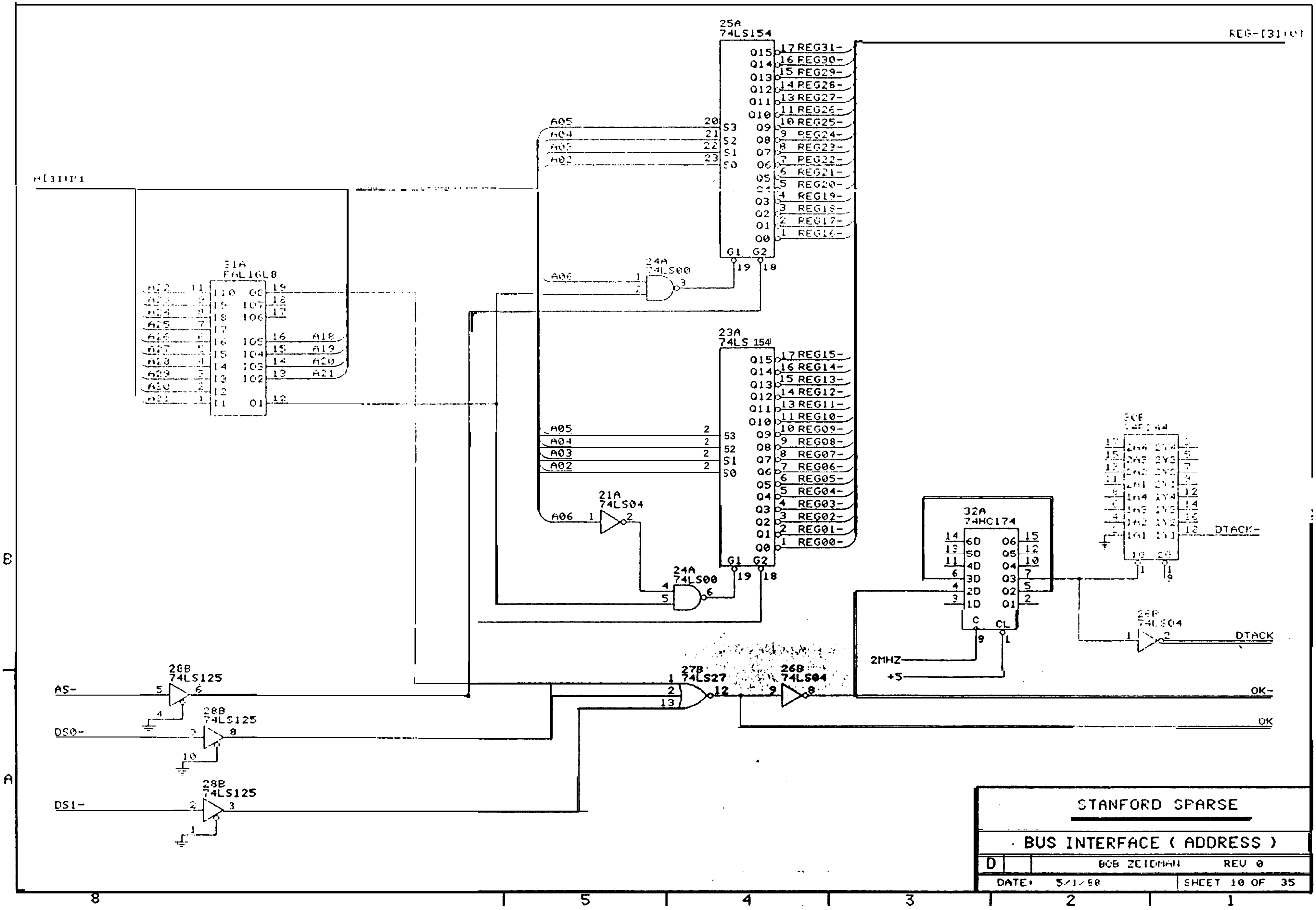
SHEET 8 OF 35



STANFORD SPARSE

HARD ADDRESS MEMORY LOGIC

D		150-0001-001	REV 1A
DATE:	5/1/88	SHEET	9 OF 34

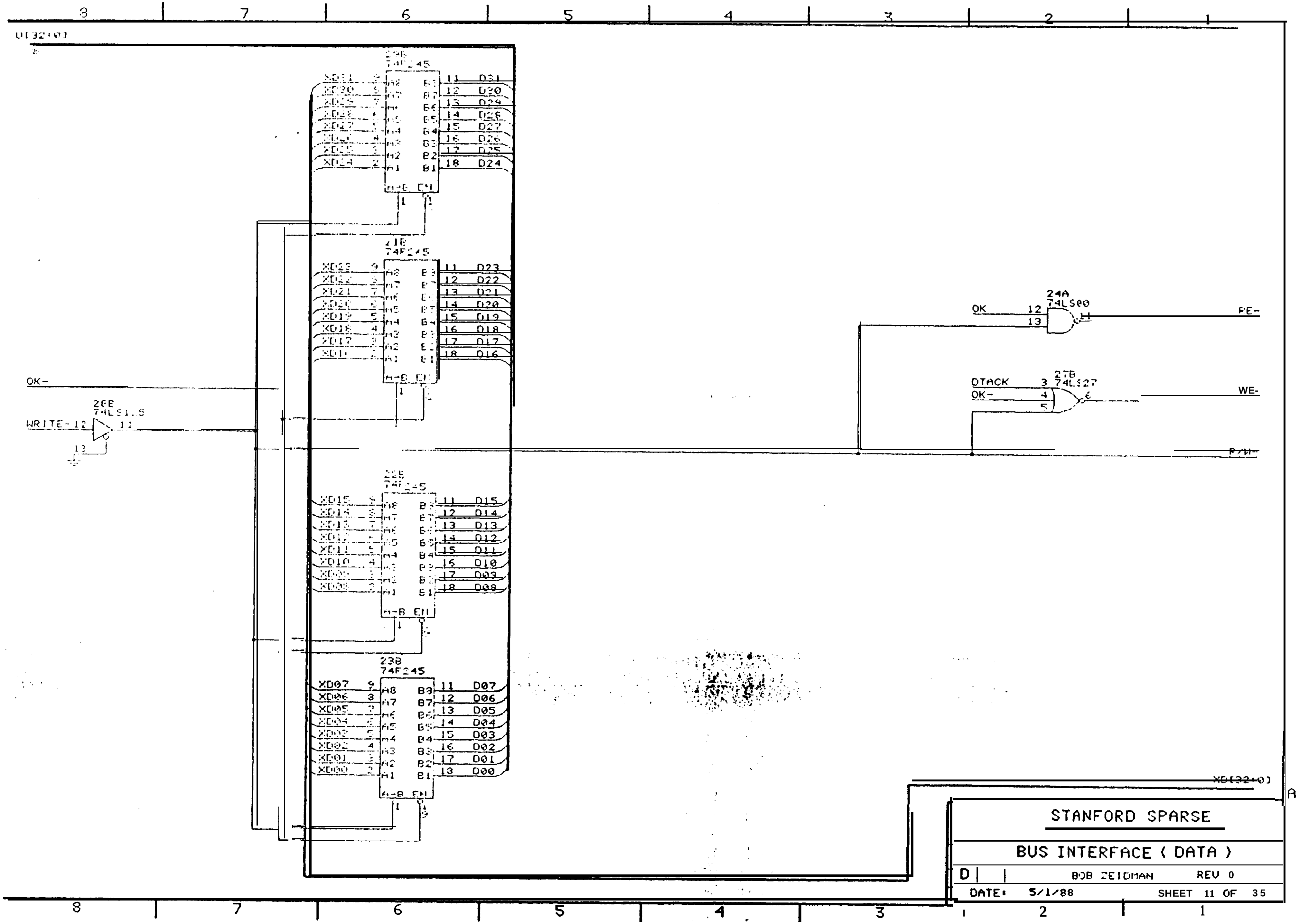


STANFORD SPARSE

BUS INTERFACE (ADDRESS)

D | BOB ZEIDMAN | REV 0

DATE: 5/1/88 | SHEET 10 OF 35

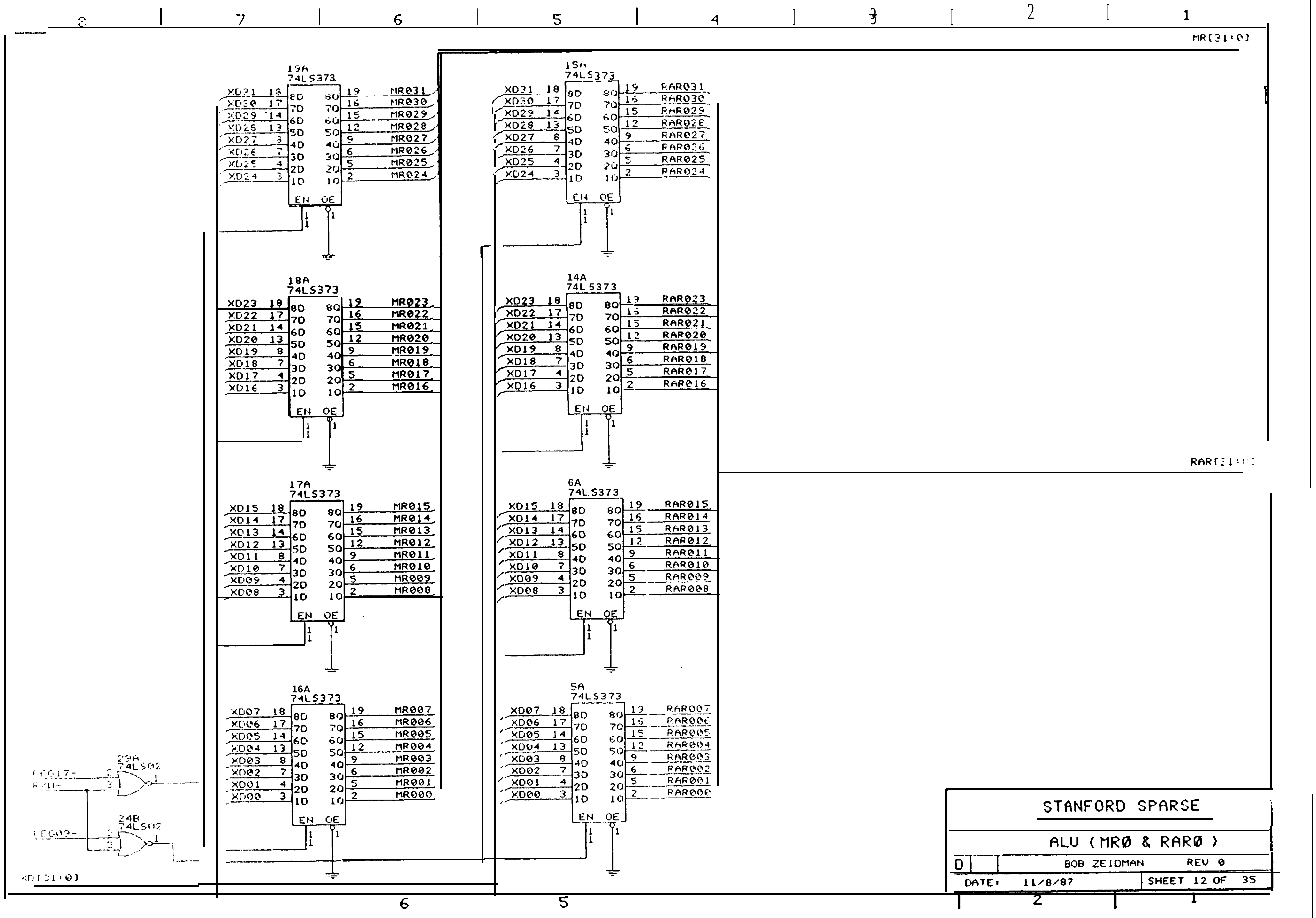


STANFORD SPARSE

BUS INTERFACE (DATA)

D | | BOB ZEIDMAN REV 0

DATE: 5/1/88 SHEET 11 OF 35

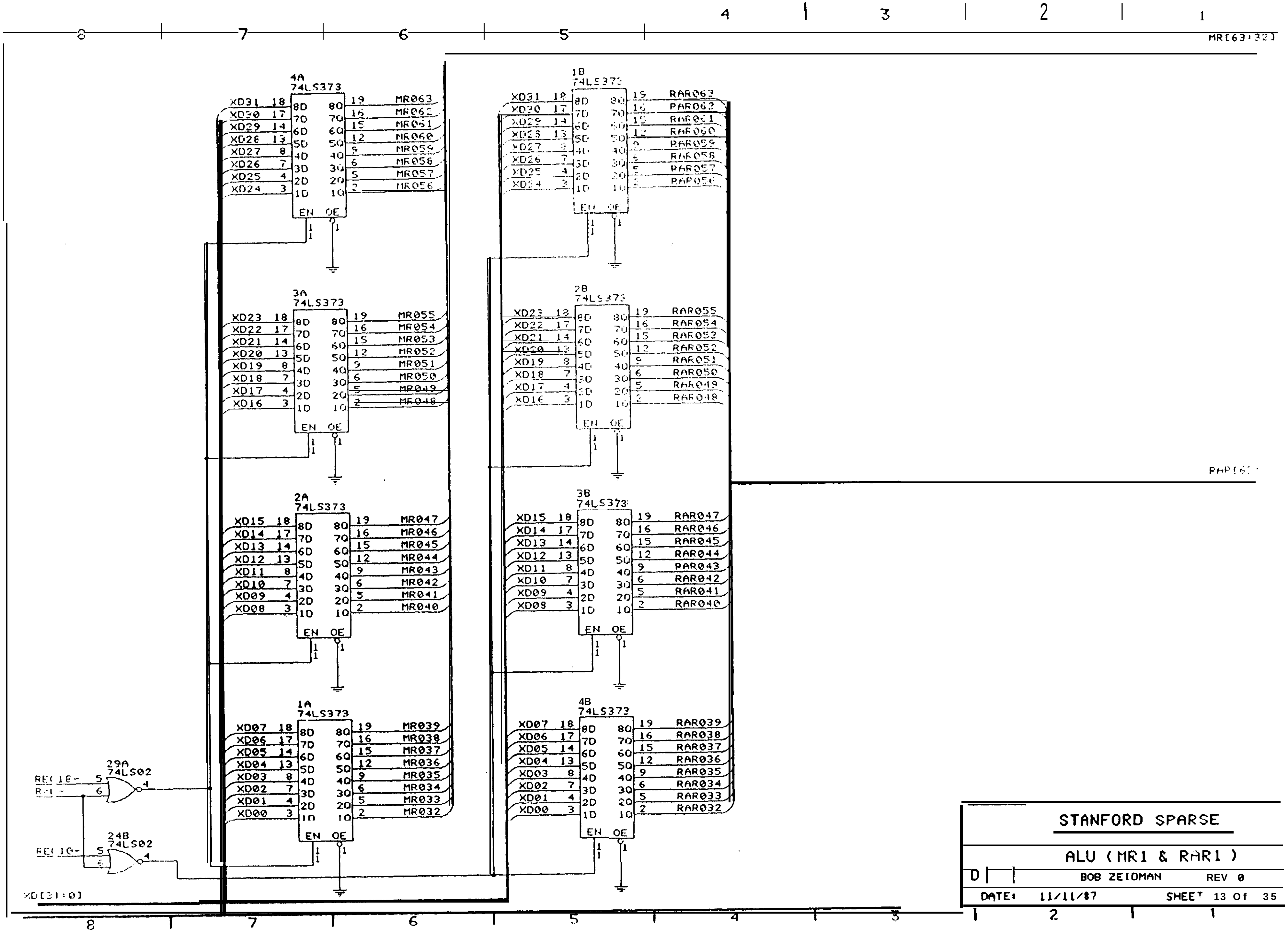


STANFORD SPARSE

ALU (MRØ & RARØ)

D BOB ZEIDMAN REV 0

DATE: 11/8/87 SHEET 12 OF 35


MR[63:32]
RAR[63:32]
XD[31:0]
4A 74LS373
3A 74LS373
2A 74LS373
1A 74LS373
1B 74LS373
2B 74LS373
3B 74LS373
4B 74LS373
29A 74LS02
24B 74LS02
RE(1E-
RE(1O-
XD31 XD30 XD29 XD28 XD27 XD26 XD25 XD24
XD23 XD22 XD21 XD20 XD19 XD18 XD17 XD16
XD15 XD14 XD13 XD12 XD11 XD10 XD09 XD08
XD07 XD06 XD05 XD04 XD03 XD02 XD01 XD00
MR063 MR062 MR061 MR060 MR059 MR058 MR057 MR056
MR055 MR054 MR053 MR052 MR051 MR050 MR049 MR048
MR047 MR046 MR045 MR044 MR043 MR042 MR041 MR040
MR039 MR038 MR037 MR036 MR035 MR034 MR033 MR032
RAR063 RAR062 RAR061 RAR060 RAR059 RAR058 RAR057 RAR056
RAR055 RAR054 RAR053 RAR052 RAR051 RAR050 RAR049 RAR048
RAR047 RAR046 RAR045 RAR044 RAR043 RAR042 RAR041 RAR040
RAR039 RAR038 RAR037 RAR036 RAR035 RAR034 RAR033 RAR032
8D 7D 6D 5D 4D 3D 2D 1D
8Q 7Q 6Q 5Q 4Q 3Q 2Q 1Q
EN OE
8 7 6 5 4 3 2 1

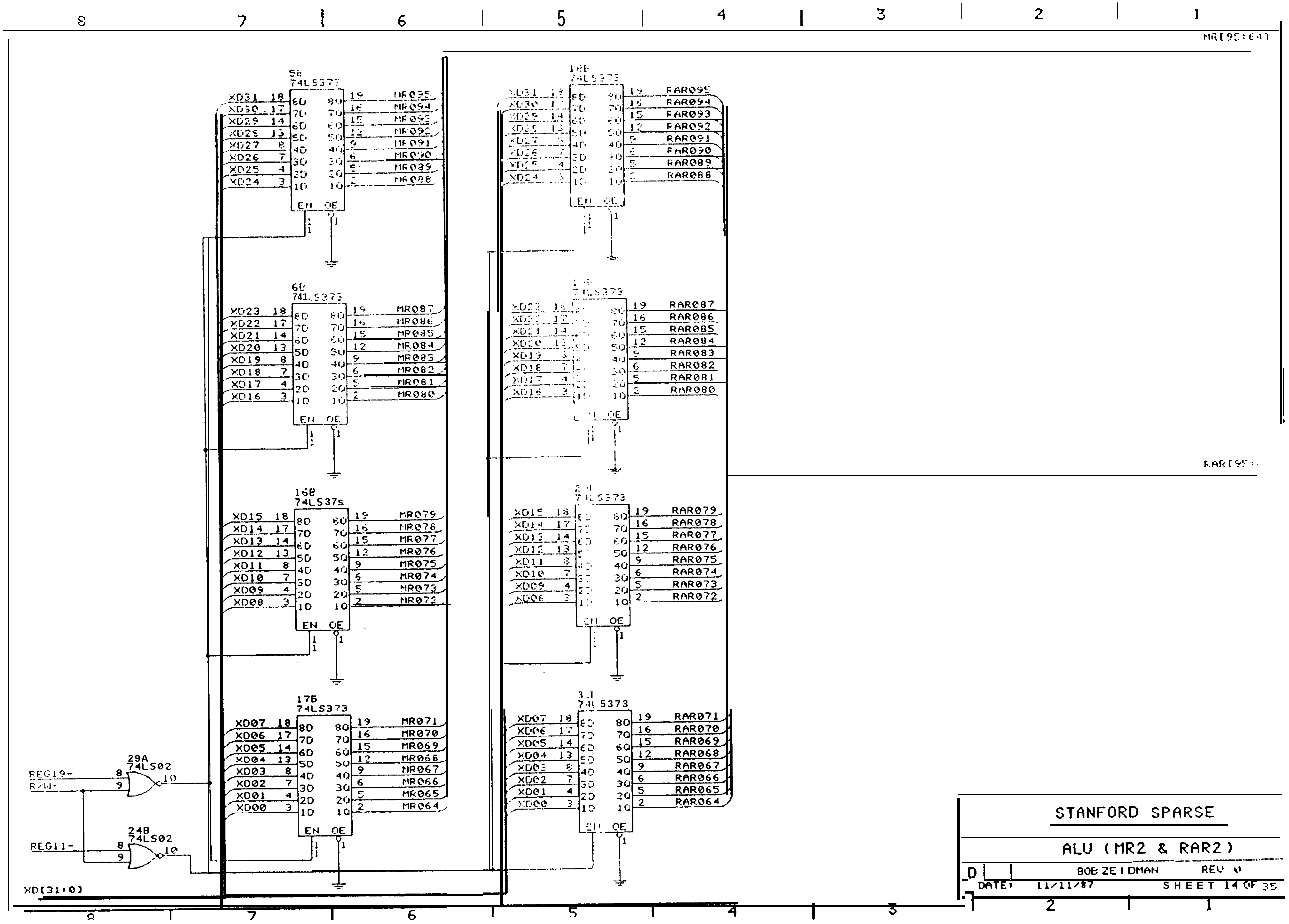
STANFORD SPARSE

ALU (MR1 & RAR1)

D

BOB ZEIDMAN REV 0

DATE: 11/11/87 SHEET 13 Of 35

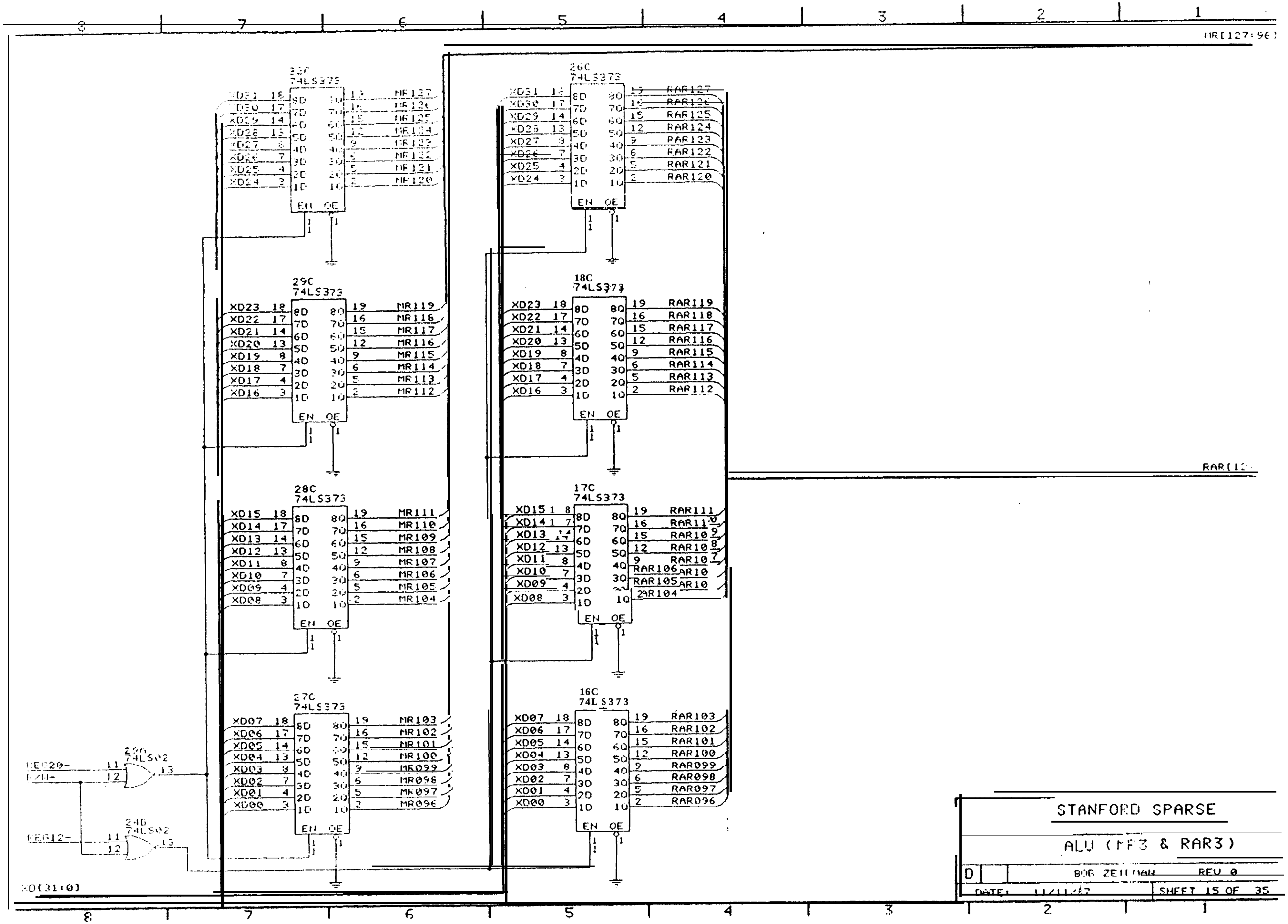


STANFORD SPARSE

ALU (MR2 & RAR2)

D | BOB ZEIDMAN | REV 0

DATE: 11/11/87 | SHEET 14 OF 35



MR[127:96]
RAR[12
XD[31:0]
33C 74LS373
29C 74LS373
28C 74LS373
27C 74LS373
26C 74LS373
18C 74LS373
17C 74LS373
16C 74LS373
23D 74LS02
24D 74LS02
REG20-
REG12-
XD31 XD30 XD29 XD28 XD27 XD26 XD25 XD24
XD23 XD22 XD21 XD20 XD19 XD18 XD17 XD16
XD15 XD14 XD13 XD12 XD11 XD10 XD09 XD08
XD07 XD06 XD05 XD04 XD03 XD02 XD01 XD00
MR127 MR126 MR125 MR124 MR123 MR122 MR121 MR120
MR119 MR118 MR117 MR116 MR115 MR114 MR113 MR112
MR111 MR110 MR109 MR108 MR107 MR106 MR105 MR104
MR103 MR102 MR101 MR100 MR099 MR098 MR097 MR096
RAR127 RAR126 RAR125 RAR124 RAR123 RAR122 RAR121 RAR120
RAR119 RAR118 RAR117 RAR116 RAR115 RAR114 RAR113 RAR112
RAR111 RAR110 RAR109 RAR108 RAR107 RAR106 RAR105 RAR104
RAR103 RAR102 RAR101 RAR100 RAR099 RAR098 RAR097 RAR096
8D 7D 6D 5D 4D 3D 2D 1D
8Q 7Q 6Q 5Q 4Q 3Q 2Q 1Q
EN OE
8 7 6 5 4 3 2 1

STANFORD SPARSE

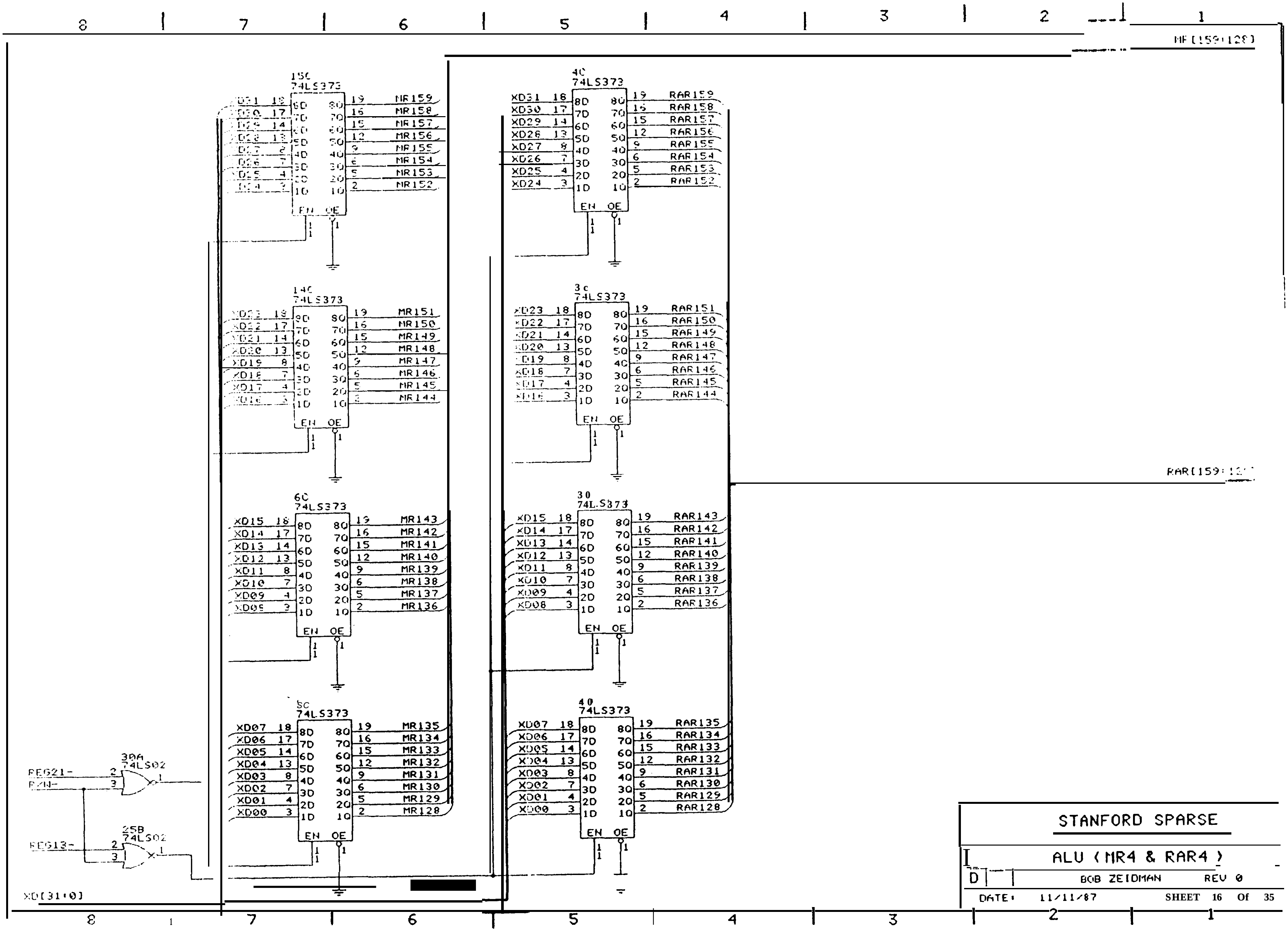
ALU (MF3 & RAR3)

D

BOB ZEITMAN REV 0

DATE: 11/11/87

SHEET 15 OF 35

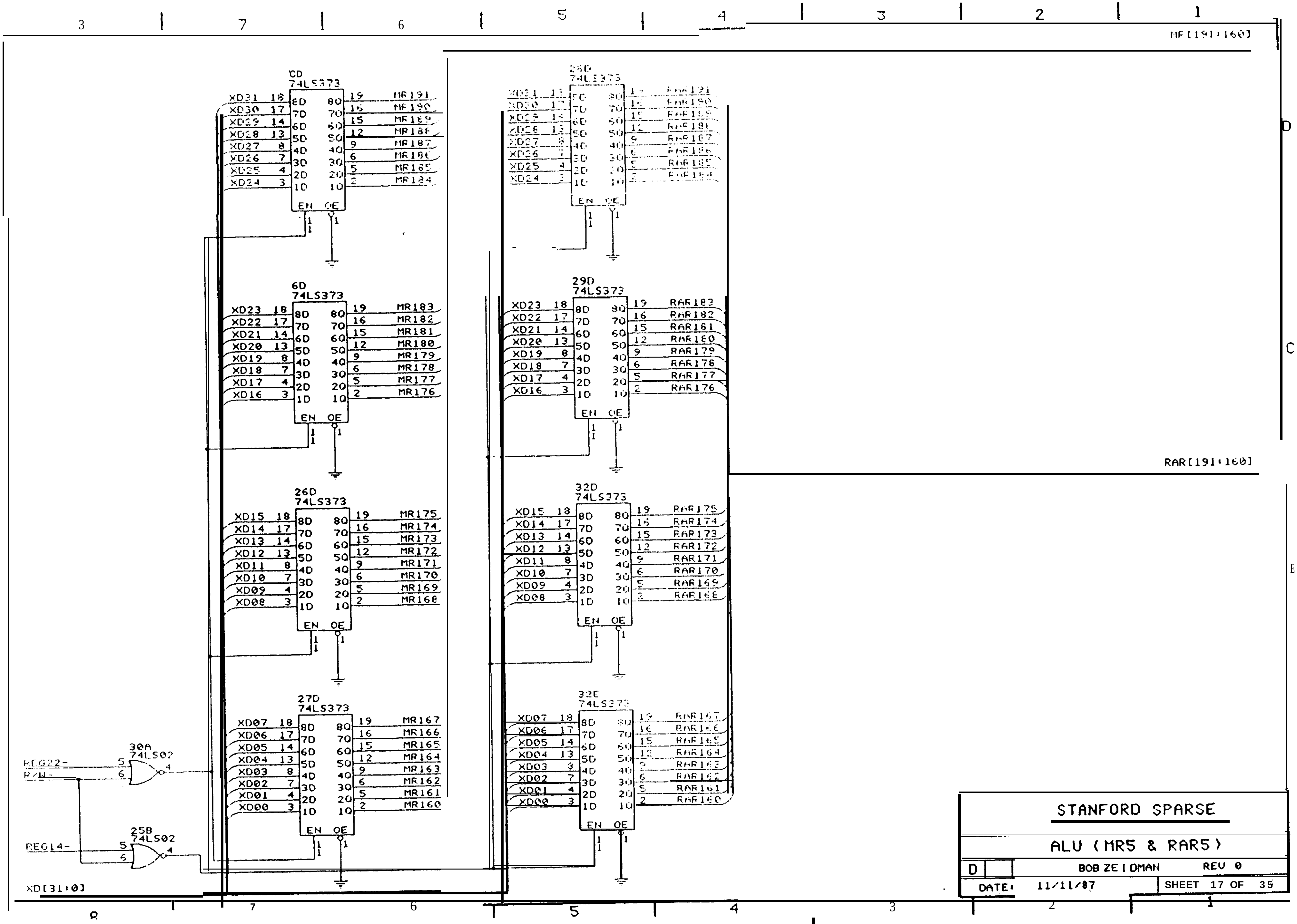

8
7
6
5
4
3
2
1
MF[159:128]
RAR[159:128]
XD[31:0]
15C 74LS373
14C 74LS373
6C 74LS373
5C 74LS373
4C 74LS373
3C 74LS373
3D 74LS373
4D 74LS373
30A 74LS02
25B 74LS02
REG21-
RZN-
REG13-
8D 7D 6D 5D 4D 3D 2D 1D
8Q 7Q 6Q 5Q 4Q 3Q 2Q 1Q
EN OE
XD31 XD30 XD29 XD28 XD27 XD26 XD25 XD24
XD23 XD22 XD21 XD20 XD19 XD18 XD17 XD16
XD15 XD14 XD13 XD12 XD11 XD10 XD09 XD08
XD07 XD06 XD05 XD04 XD03 XD02 XD01 XD00
MR159 MR158 MR157 MR156 MR155 MR154 MR153 MR152
MR151 MR150 MR149 MR148 MR147 MR146 MR145 MR144
MR143 MR142 MR141 MR140 MR139 MR138 MR137 MR136
MR135 MR134 MR133 MR132 MR131 MR130 MR129 MR128
RAR159 RAR158 RAR157 RAR156 RAR155 RAR154 RAR153 RAR152
RAR151 RAR150 RAR149 RAR148 RAR147 RAR146 RAR145 RAR144
RAR143 RAR142 RAR141 RAR140 RAR139 RAR138 RAR137 RAR136
RAR135 RAR134 RAR133 RAR132 RAR131 RAR130 RAR129 RAR128

STANFORD SPARSE

ALU (MR4 & RAR4)

D BOB ZEIDMAN REV 0

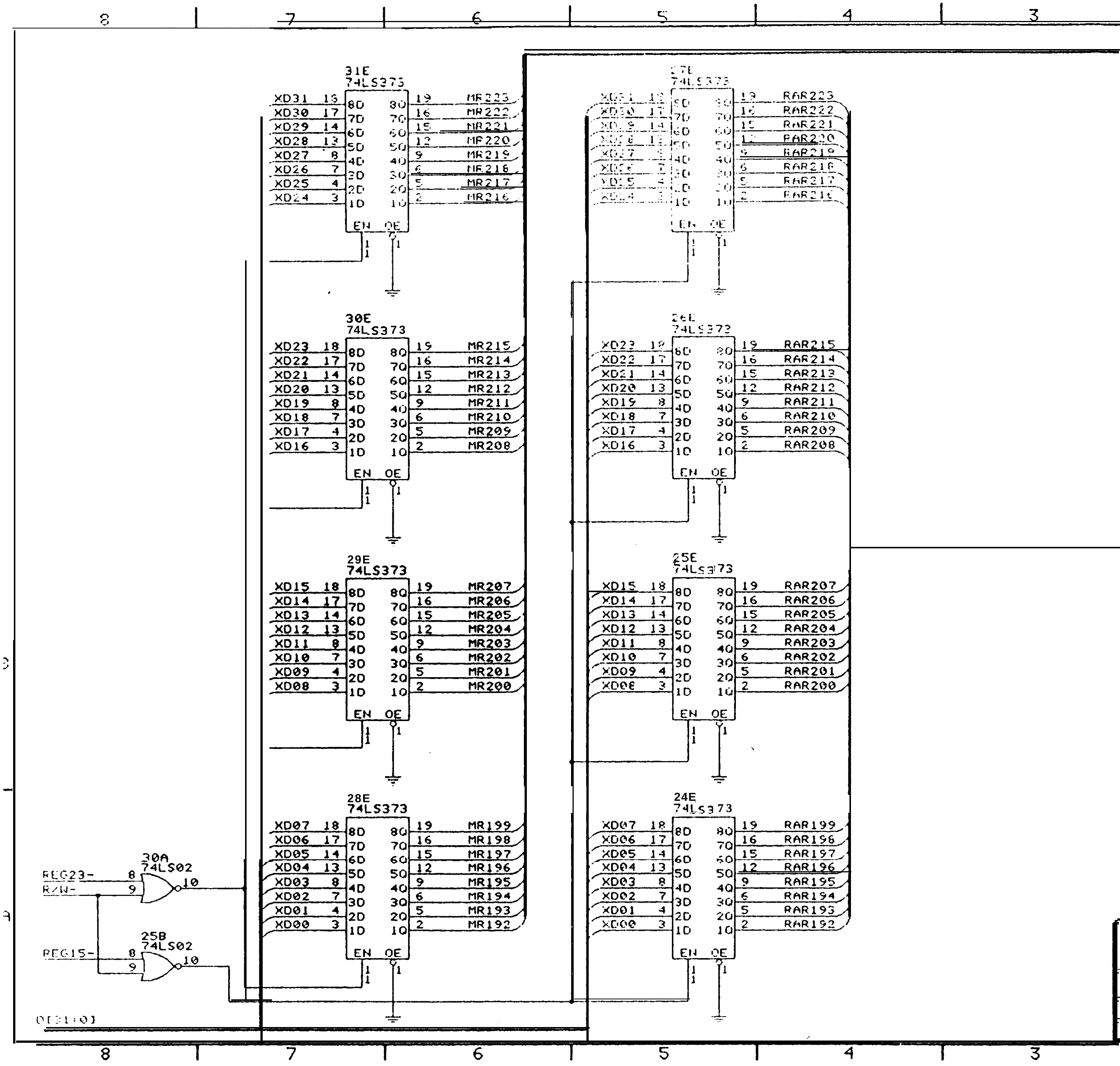
DATE: 11/11/87 SHEET 16 Of 35


MR[191:160]
RAR[191:160]
XD[31:0]
CD
74LS373
6D
74LS373
26D
74LS373
27D
74LS373
28D
74LS373
29D
74LS373
32D
74LS373
32E
74LS373
8D 7D 6D 5D 4D 3D 2D 1D
8Q 7Q 6Q 5Q 4Q 3Q 2Q 1Q
EN OE
XD31 XD30 XD29 XD28 XD27 XD26 XD25 XD24
XD23 XD22 XD21 XD20 XD19 XD18 XD17 XD16
XD15 XD14 XD13 XD12 XD11 XD10 XD09 XD08
XD07 XD06 XD05 XD04 XD03 XD02 XD01 XD00
MR191 MR190 MR189 MR188 MR187 MR186 MR185 MR184
MR183 MR182 MR181 MR180 MR179 MR178 MR177 MR176
MR175 MR174 MR173 MR172 MR171 MR170 MR169 MR168
MR167 MR166 MR165 MR164 MR163 MR162 MR161 MR160
RAR191 RAR190 RAR189 RAR188 RAR187 RAR186 RAR185 RAR184
RAR183 RAR182 RAR181 RAR180 RAR179 RAR178 RAR177 RAR176
RAR175 RAR174 RAR173 RAR172 RAR171 RAR170 RAR169 RAR168
RAR167 RAR166 RAR165 RAR164 RAR163 RAR162 RAR161 RAR160
30A
74LS02
25B
74LS02
REG22-
R/W-
REG14-
1 2 3 4 5 6 7 8
C D E

STANFORD SPARSE

ALU (MR5 & RAR5)

D		BOB ZEIDMAN	REV 0
DATE:	11/11/87	SHEET 17 OF 35	

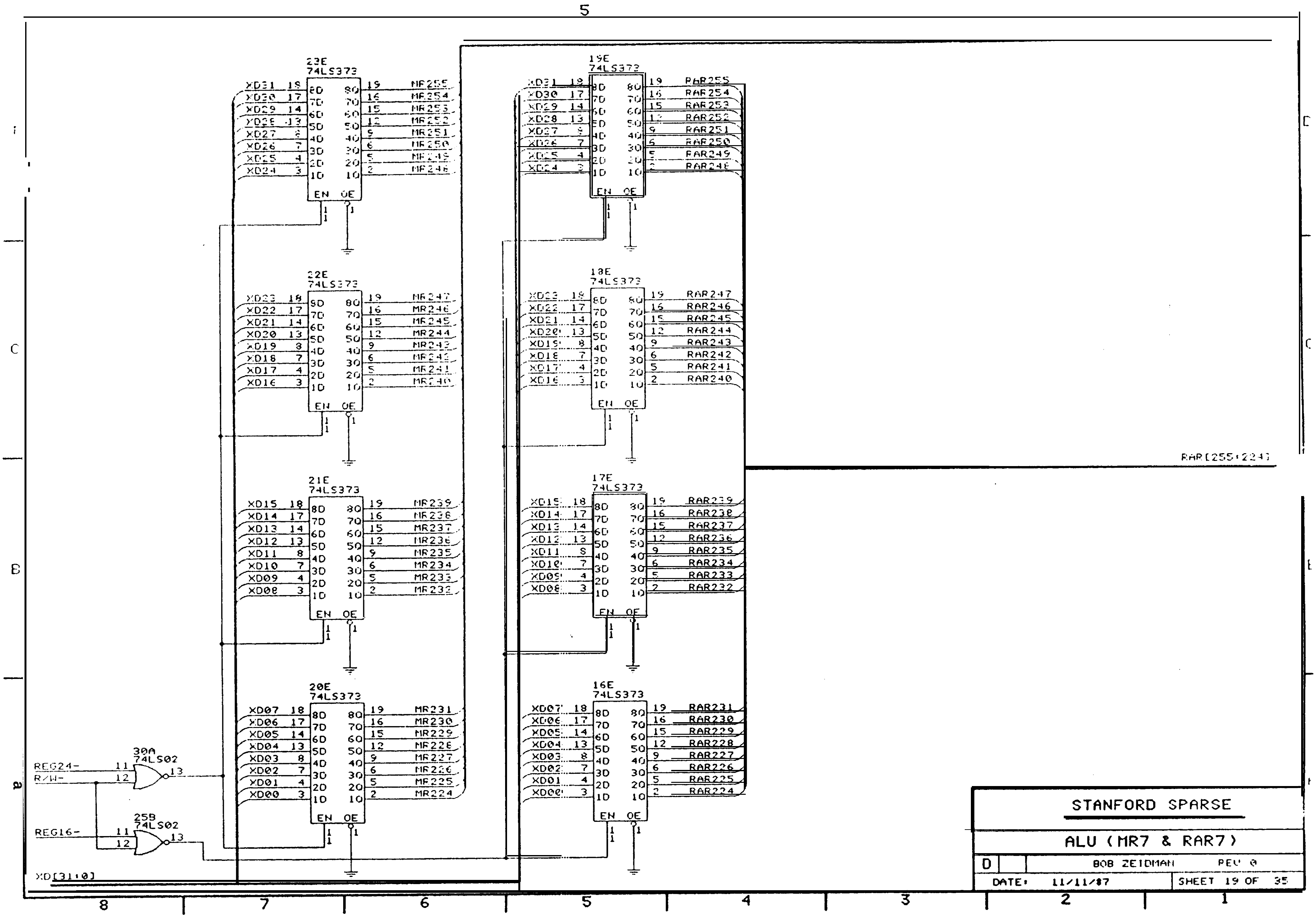

MR[223:192]
RAR[223:192]
31E 74LS373
30E 74LS373
29E 74LS373
28E 74LS373
27E 74LS373
26E 74LS373
25E 74LS373
24E 74LS373
XD31 XD30 XD29 XD28 XD27 XD26 XD25 XD24
XD23 XD22 XD21 XD20 XD19 XD18 XD17 XD16
XD15 XD14 XD13 XD12 XD11 XD10 XD09 XD08
XD07 XD06 XD05 XD04 XD03 XD02 XD01 XD00
MR223 MR222 MR221 MR220 MR219 MR218 MR217 MR216
MR215 MR214 MR213 MR212 MR211 MR210 MR209 MR208
MR207 MR206 MR205 MR204 MR203 MR202 MR201 MR200
MR199 MR198 MR197 MR196 MR195 MR194 MR193 MR192
RAR223 RAR222 RAR221 RAR220 RAR219 RAR218 RAR217 RAR216
RAR215 RAR214 RAR213 RAR212 RAR211 RAR210 RAR209 RAR208
RAR207 RAR206 RAR205 RAR204 RAR203 RAR202 RAR201 RAR200
RAR199 RAR198 RAR197 RAR196 RAR195 RAR194 RAR193 RAR192
8D 7D 6D 5D 4D 3D 2D 1D
8Q 7Q 6Q 5Q 4Q 3Q 2Q 1Q
EN OE
30A 74LS02
25B 74LS02
REG23-
R/W-
REG15-
D[31:0]
8 7 6 5 4 3 2 1

STANFORD SPARSE

ALU (MR6 & RAR6)

D | BOB ZEIDMAN | REV 0

DATE: 11/11/87 | SHEET 18 OF 35

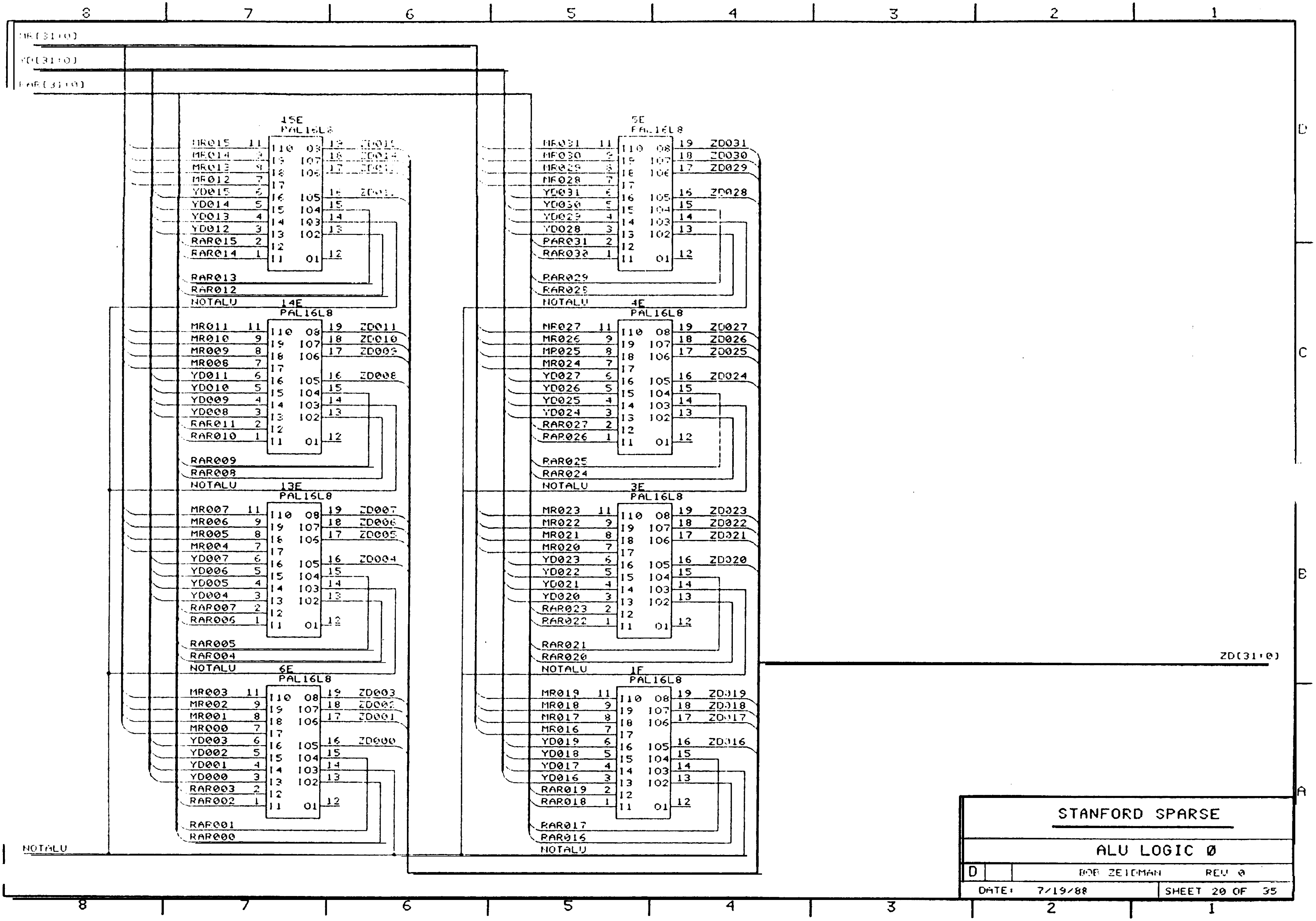

23E
74LS373
22E
74LS373
21E
74LS373
20E
74LS373
19E
74LS373
18E
74LS373
17E
74LS373
16E
74LS373
30A
74LS02
25B
74LS02
REG24-
R/W-
REG16-
XD[31:0]
RAR[255:224]
XD31 XD30 XD29 XD28 XD27 XD26 XD25 XD24
XD23 XD22 XD21 XD20 XD19 XD18 XD17 XD16
XD15 XD14 XD13 XD12 XD11 XD10 XD09 XD08
XD07 XD06 XD05 XD04 XD03 XD02 XD01 XD00
MR255 MR254 MR253 MR252 MR251 MR250 MR249 MR248
MR247 MR246 MR245 MR244 MR243 MR242 MR241 MR240
MR239 MR238 MR237 MR236 MR235 MR234 MR233 MR232
MR231 MR230 MR229 MR228 MR227 MR226 MR225 MR224
RAR255 RAR254 RAR253 RAR252 RAR251 RAR250 RAR249 RAR248
RAR247 RAR246 RAR245 RAR244 RAR243 RAR242 RAR241 RAR240
RAR239 RAR238 RAR237 RAR236 RAR235 RAR234 RAR233 RAR232
RAR231 RAR230 RAR229 RAR228 RAR227 RAR226 RAR225 RAR224
8D 7D 6D 5D 4D 3D 2D 1D
8Q 7Q 6Q 5Q 4Q 3Q 2Q 1Q
EN OE
8 7 6 5 4 3 2 1
D C B a

STANFORD SPARSE

ALU (MR7 & RAR7)

D BOB ZEIDMAN REV 0

DATE: 11/11/87 SHEET 19 OF 35

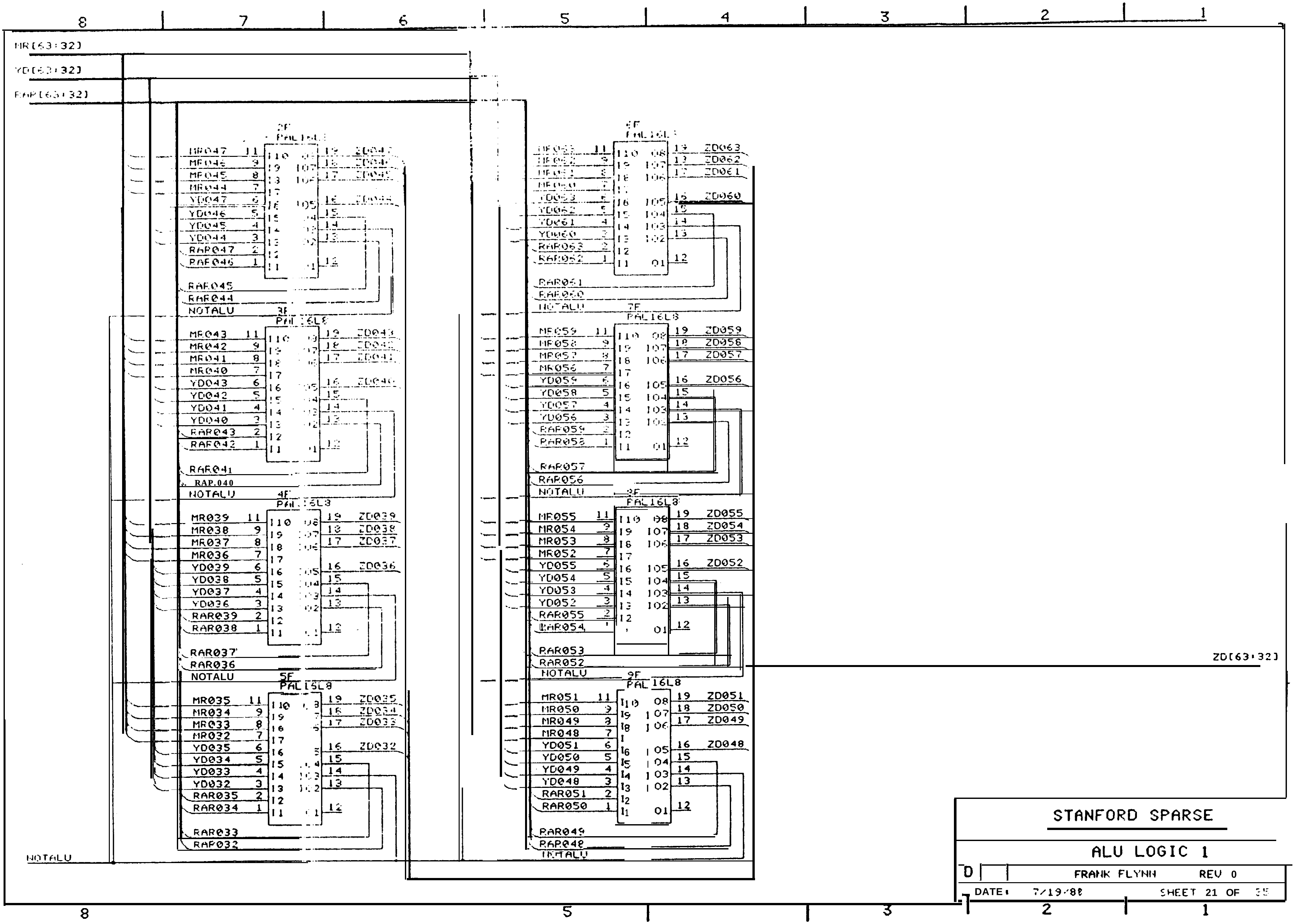

MR[31:0]
YD[31:0]
RAR[31:0]
ZD[31:0]
NOTALU
15E PAL16L8
MR015 MR014 MR013 MR012 YD015 YD014 YD013 YD012 RAR015 RAR014
ZD015 ZD014 ZD013 ZD012
RAR013 RAR012 NOTALU
14E PAL16L8
MR011 MR010 MR009 MR008 YD011 YD010 YD009 YD008 RAR011 RAR010
ZD011 ZD010 ZD009 ZD008
RAR009 RAR008 NOTALU
13E PAL16L8
MR007 MR006 MR005 MR004 YD007 YD006 YD005 YD004 RAR007 RAR006
ZD007 ZD006 ZD005 ZD004
RAR005 RAR004 NOTALU
6E PAL16L8
MR003 MR002 MR001 MR000 YD003 YD002 YD001 YD000 RAR003 RAR002
ZD003 ZD002 ZD001 ZD000
RAR001 RAR000 NOTALU
5E PAL16L8
MR031 MR030 MR029 MR028 YD031 YD030 YD029 YD028 RAR031 RAR030
ZD031 ZD030 ZD029 ZD028
RAR029 RAR028 NOTALU
4E PAL16L8
MR027 MR026 MR025 MR024 YD027 YD026 YD025 YD024 RAR027 RAR026
ZD027 ZD026 ZD025 ZD024
RAR025 RAR024 NOTALU
3E PAL16L8
MR023 MR022 MR021 MR020 YD023 YD022 YD021 YD020 RAR023 RAR022
ZD023 ZD022 ZD021 ZD020
RAR021 RAR020 NOTALU
1F PAL16L8
MR019 MR018 MR017 MR016 YD019 YD018 YD017 YD016 RAR019 RAR018
ZD019 ZD018 ZD017 ZD016
RAR017 RAR016 NOTALU
I10 I9 I8 I7 I6 I5 I4 I3 I2 I1
O8 IO7 IO6 IO5 IO4 IO3 IO2 O1
11 9 8 7 6 5 4 3 2 1
19 18 17 16 15 14 13 12
8 7 6 5 4 3 2 1
D C B A

STANFORD SPARSE

ALU LOGIC Ø

D | BOB ZEIDMAN | REV 0

DATE: 7/19/88 | SHEET 20 OF 35



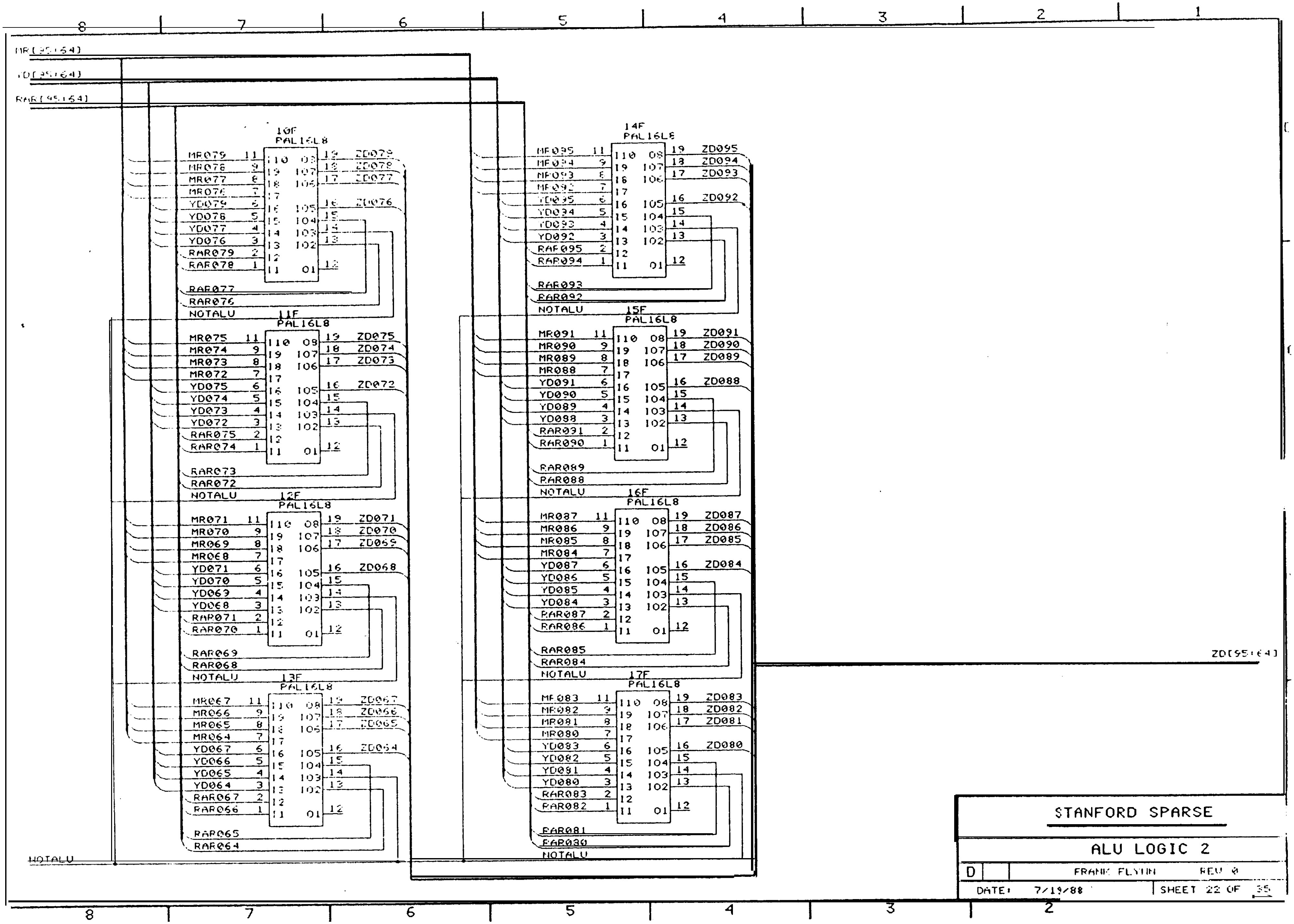
8
7
6
5
4
3
2
1
A
MR[63:32]
YD[63:32]
RAR[63:32]
NOTALU
ZD[63:32]
2F PAL16L8
MR047 MR046 MR045 MR044 YD047 YD046 YD045 YD044 RAR047 RAR046
RAR045 RAR044 NOTALU
ZD047 ZD046 ZD045 ZD044
3F PAL16L8
MR043 MR042 MR041 MR040 YD043 YD042 YD041 YD040 RAR043 RAR042
RAR041 RAR040 NOTALU
ZD043 ZD042 ZD041 ZD040
4F PAL16L8
MR039 MR038 MR037 MR036 YD039 YD038 YD037 YD036 RAR039 RAR038
RAR037 RAR036 NOTALU
ZD039 ZD038 ZD037 ZD036
5F PAL16L8
MR035 MR034 MR033 MR032 YD035 YD034 YD033 YD032 RAR035 RAR034
RAR033 RAR032
ZD035 ZD034 ZD033 ZD032
6F PAL16L8
MR063 MR062 MR061 MR060 YD063 YD062 YD061 YD060 RAR063 RAR062
RAR061 RAR060 NOTALU
ZD063 ZD062 ZD061 ZD060
7F PAL16L8
MR059 MR058 MR057 MR056 YD059 YD058 YD057 YD056 RAR059 RAR058
RAR057 RAR056 NOTALU
ZD059 ZD056 ZD057 ZD056
8F PAL16L8
MR055 MR054 MR053 MR052 YD055 YD054 YD053 YD052 RAR055 RAR054
RAR053 RAR052 NOTALU
ZD055 ZD054 ZD053 ZD052
9F PAL16L8
MR051 MR050 MR049 MR048 YD051 YD050 YD049 YD048 RAR051 RAR050
RAR049 RAR048 NOTALU
ZD051 ZD050 ZD049 ZD048
I10 I9 I8 I7 I6 I5 I4 I3 I2 I1
O8 IO7 IO6 IO5 IO4 IO3 IO2 O1
11 9 8 7 6 5 4 3 2 1
19 18 17 16 15 14 13 12

STANFORD SPARSE

ALU LOGIC 1

D | FRANK FLYNN | REV 0

DATE: 7/19/88 | SHEET 21 OF 35



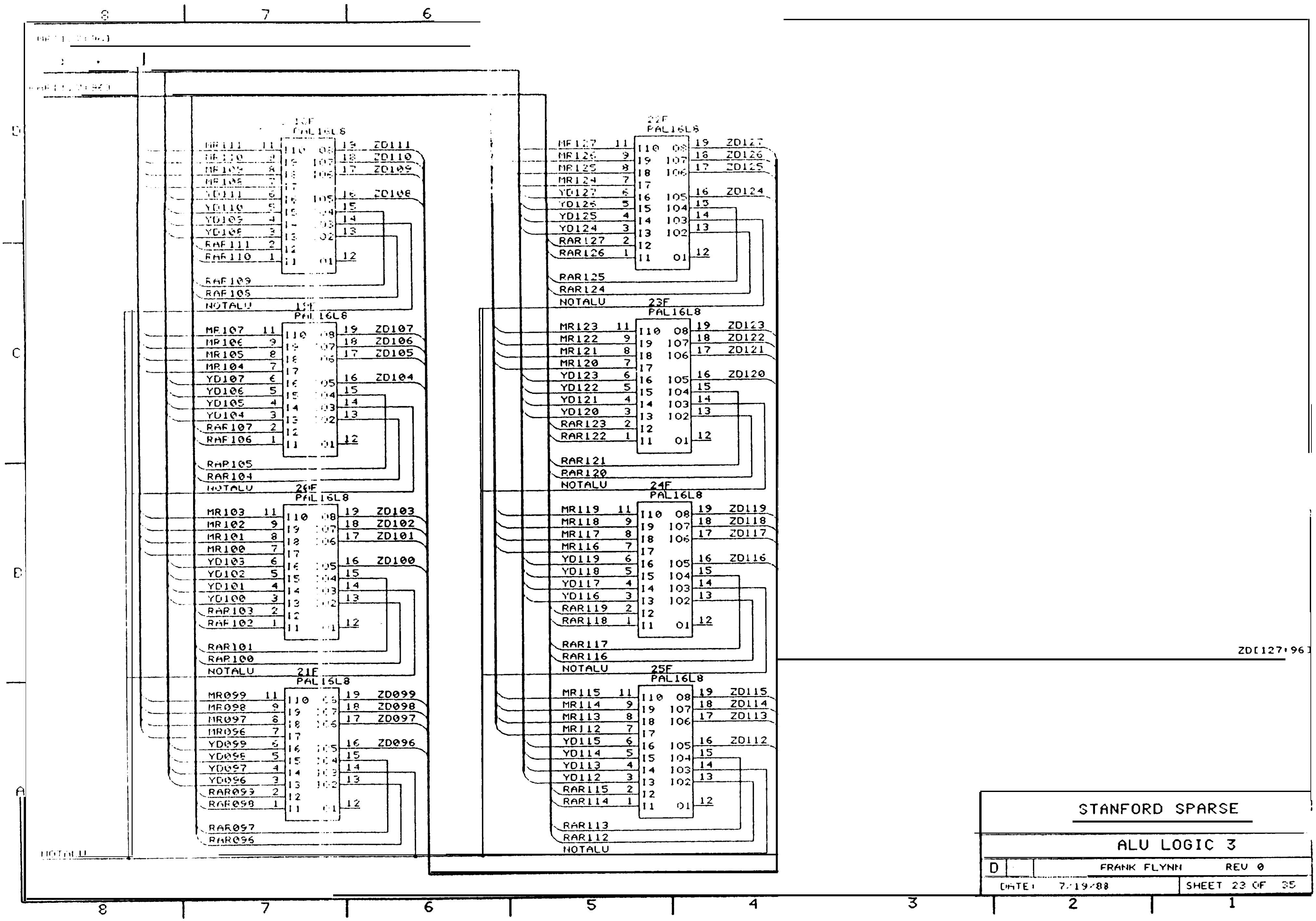
STANFORD SPARSE

ALU LOGIC 2

D

FRANK FLYNN REV 0

DATE: 7/19/88 SHEET 22 OF 35



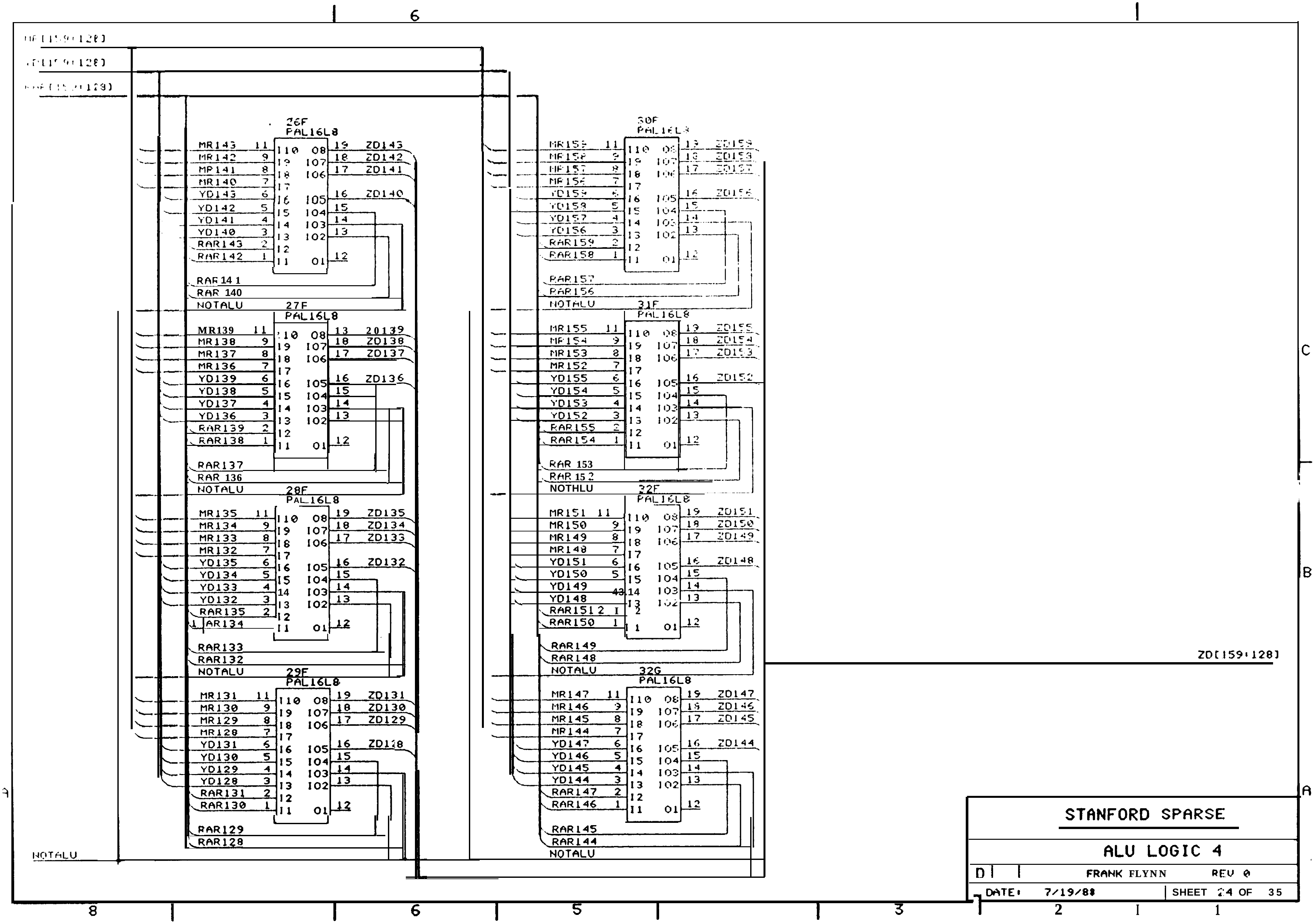
ZD[127:96]
NOTALU
20F PAL16L8
MR111 MR110 MR109 MR108 YD111 YD110 YD109 YD108 RAR111 RAR110
ZD111 ZD110 ZD109 ZD108
RAR109 RAR108 NOTALU
19F PAL16L8
MR107 MR106 MR105 MR104 YD107 YD106 YD105 YD104 RAR107 RAR106
ZD107 ZD106 ZD105 ZD104
RAR105 RAR104 NOTALU
26F PAL16L8
MR103 MR102 MR101 MR100 YD103 YD102 YD101 YD100 RAR103 RAR102
ZD103 ZD102 ZD101 ZD100
RAR101 RAR100 NOTALU
21F PAL16L8
MR099 MR098 MR097 MR096 YD099 YD098 YD097 YD096 RAR099 RAR098
ZD099 ZD098 ZD097 ZD096
RAR097 RAR096
22F PAL16L8
MR127 MR126 MR125 MR124 YD127 YD126 YD125 YD124 RAR127 RAR126
ZD127 ZD126 ZD125 ZD124
RAR125 RAR124 NOTALU
23F PAL16L8
MR123 MR122 MR121 MR120 YD123 YD122 YD121 YD120 RAR123 RAR122
ZD123 ZD122 ZD121 ZD120
RAR121 RAR120 NOTALU
24F PAL16L8
MR119 MR118 MR117 MR116 YD119 YD118 YD117 YD116 RAR119 RAR118
ZD119 ZD118 ZD117 ZD116
RAR117 RAR116 NOTALU
25F PAL16L8
MR115 MR114 MR113 MR112 YD115 YD114 YD113 YD112 RAR115 RAR114
ZD115 ZD114 ZD113 ZD112
RAR113 RAR112 NOTALU
I10 I9 I8 I7 I6 I5 I4 I3 I2 I1
O8 IO7 IO6 IO5 IO4 IO3 IO2 O1
8 7 6 5 4 3 2 1
D C B A

STANFORD SPARSE

ALU LOGIC 3

D | FRANK FLYNN | REV 0

DATE: 7/19/88 | SHEET 23 OF 35

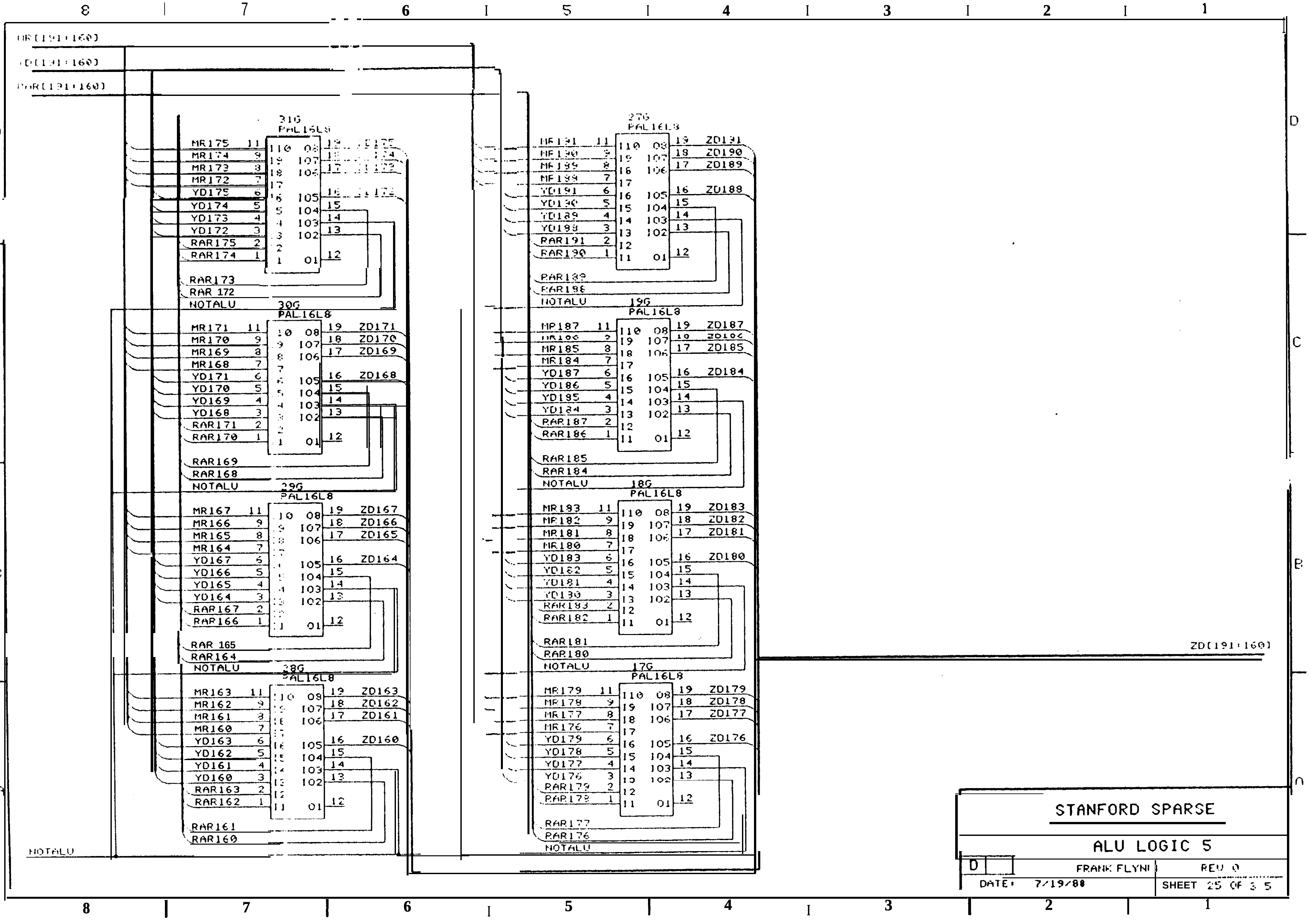

MR[159:128]
YD[159:128]
RAR[159:128]
ZD[159:128]
NOTALU
26F
PAL16L8
MR143 MR142 MR141 MR140 YD143 YD142 YD141 YD140 RAR143 RAR142
ZD143 ZD142 ZD141 ZD140
RAR 141 RAR 140 NOTALU
27F
PAL16L8
MR139 MR138 MR137 MR136 YD139 YD138 YD137 YD136 RAR139 RAR138
ZD139 ZD138 ZD137 ZD136
RAR137 RAR 136 NOTALU
28F
PAL16L8
MR135 MR134 MR133 MR132 YD135 YD134 YD133 YD132 RAR135 RAR134
ZD135 ZD134 ZD133 ZD132
RAR133 RAR132 NOTALU
29F
PAL16L8
MR131 MR130 MR129 MR128 YD131 YD130 YD129 YD128 RAR131 RAR130
ZD131 ZD130 ZD129 ZD128
RAR129 RAR128
30F
PAL16L8
MR159 MR158 MR157 MR156 YD159 YD158 YD157 YD156 RAR159 RAR158
ZD159 ZD158 ZD157 ZD156
RAR157 RAR156 NOTALU
31F
PAL16L8
MR155 MR154 MR153 MR152 YD155 YD154 YD153 YD152 RAR155 RAR154
ZD155 ZD154 ZD153 ZD152
RAR 153 RAR 152 NOTHLU
32F
PAL16L8
MR151 MR150 MR149 MR148 YD151 YD150 YD149 YD148 RAR151 RAR150
ZD151 ZD150 ZD149 ZD148
RAR149 RAR148 NOTALU
32G
PAL16L8
MR147 MR146 MR145 MR144 YD147 YD146 YD145 YD144 RAR147 RAR146
ZD147 ZD146 ZD145 ZD144
RAR145 RAR144 NOTALU
I10 I9 I8 I7 I6 I5 I4 I3 I2 I1
O8 IO7 IO6 IO5 IO4 IO3 IO2 O1
11 9 8 7 6 5 4 3 2 1
19 18 17 16 15 14 13 12
8 6 5 3 2 1
A B C

STANFORD SPARSE

ALU LOGIC 4

D | FRANK FLYNN REV 0

DATE: 7/19/88 SHEET 24 OF 35



STANFORD SPARSE

ALU LOGIC 5

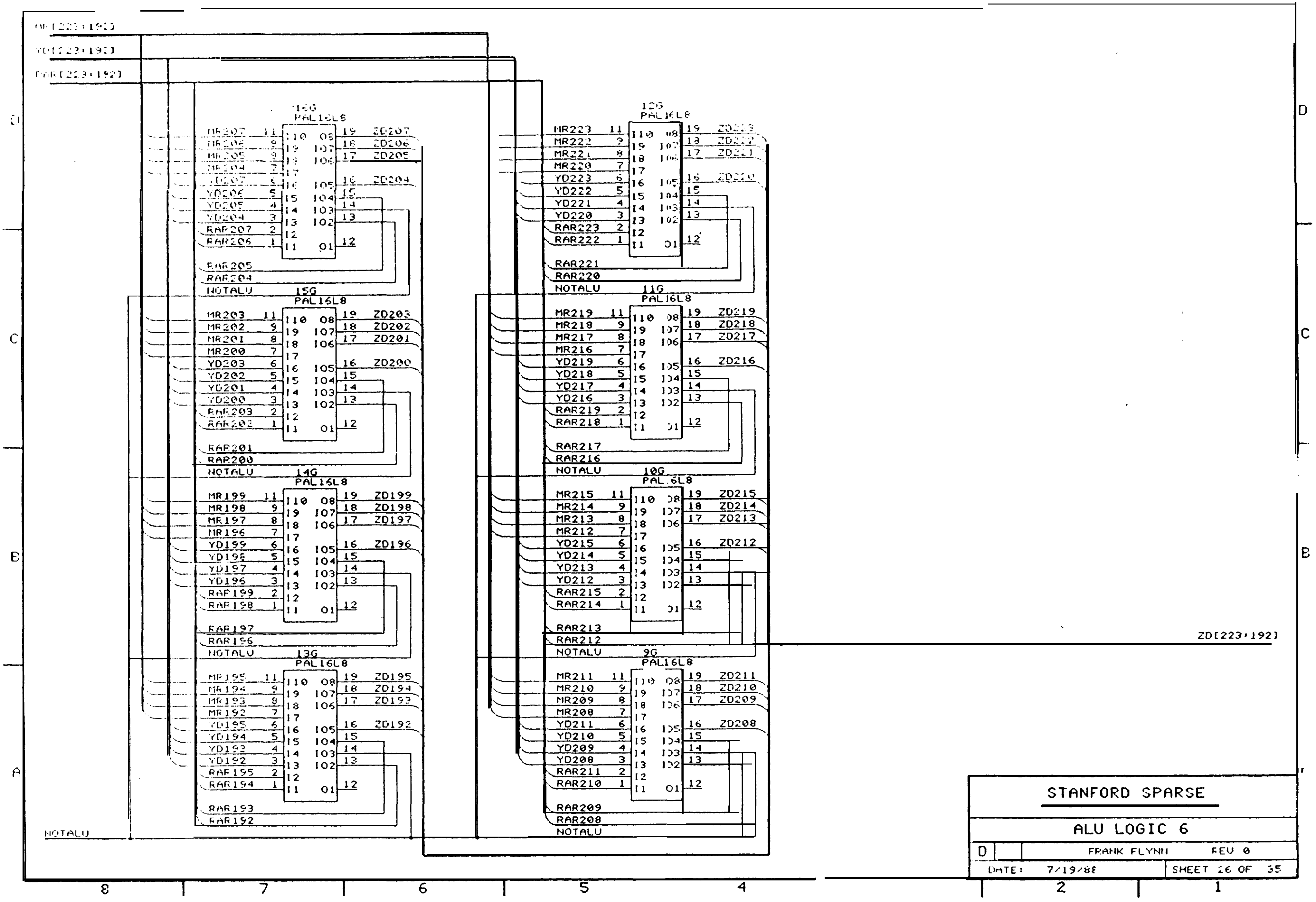
D

FRANK FLYNN

REV 0

DATE: 7/19/88

SHEET 25 OF 35

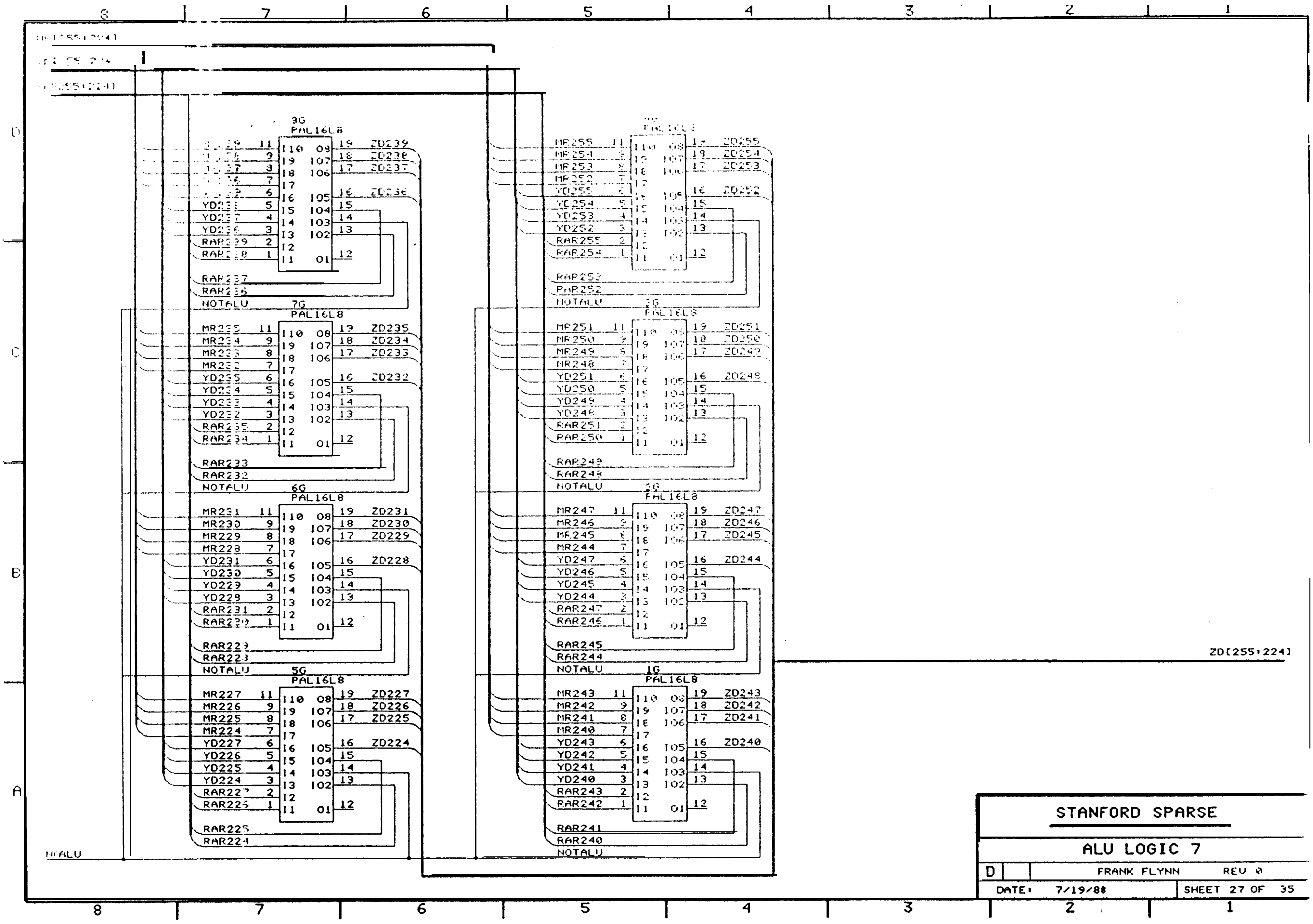

MR[223:192]
YD[223:192]
RAR[223:192]
ZD[223:192]
NOTALU
16G PAL16L8
MR207 MR206 MR205 MR204 YD207 YD206 YD205 YD204 RAR207 RAR206 RAR205 RAR204 NOTALU
ZD207 ZD206 ZD205 ZD204
15G PAL16L8
MR203 MR202 MR201 MR200 YD203 YD202 YD201 YD200 RAR203 RAR202 RAR201 RAR200 NOTALU
ZD203 ZD202 ZD201 ZD200
14G PAL16L8
MR199 MR198 MR197 MR196 YD199 YD198 YD197 YD196 RAR199 RAR198 RAR197 RAR196 NOTALU
ZD199 ZD198 ZD197 ZD196
13G PAL16L8
MR195 MR194 MR193 MR192 YD195 YD194 YD193 YD192 RAR195 RAR194 RAR193 RAR192
ZD195 ZD194 ZD193 ZD192
12G PAL16L8
MR223 MR222 MR221 MR220 YD223 YD222 YD221 YD220 RAR223 RAR222 RAR221 RAR220 NOTALU
ZD223 ZD222 ZD221 ZD220
11G PAL16L8
MR219 MR218 MR217 MR216 YD219 YD218 YD217 YD216 RAR219 RAR218 RAR217 RAR216 NOTALU
ZD219 ZD218 ZD217 ZD216
10G PAL16L8
MR215 MR214 MR213 MR212 YD215 YD214 YD213 YD212 RAR215 RAR214 RAR213 RAR212 NOTALU
ZD215 ZD214 ZD213 ZD212
9G PAL16L8
MR211 MR210 MR209 MR208 YD211 YD210 YD209 YD208 RAR211 RAR210 RAR209 RAR208 NOTALU
ZD211 ZD210 ZD209 ZD208
I10 I9 I8 I7 I6 I5 I4 I3 I2 I1
O8 IO7 IO6 IO5 IO4 IO3 IO2 O1
11 9 8 7 6 5 4 3 2 1
19 18 17 16 15 14 13 12
D C B A
8 7 6 5 4 2 1

STANFORD SPARSE

ALU LOGIC 6

D FRANK FLYNN REV 0

DATE: 7/19/88 SHEET 16 OF 35

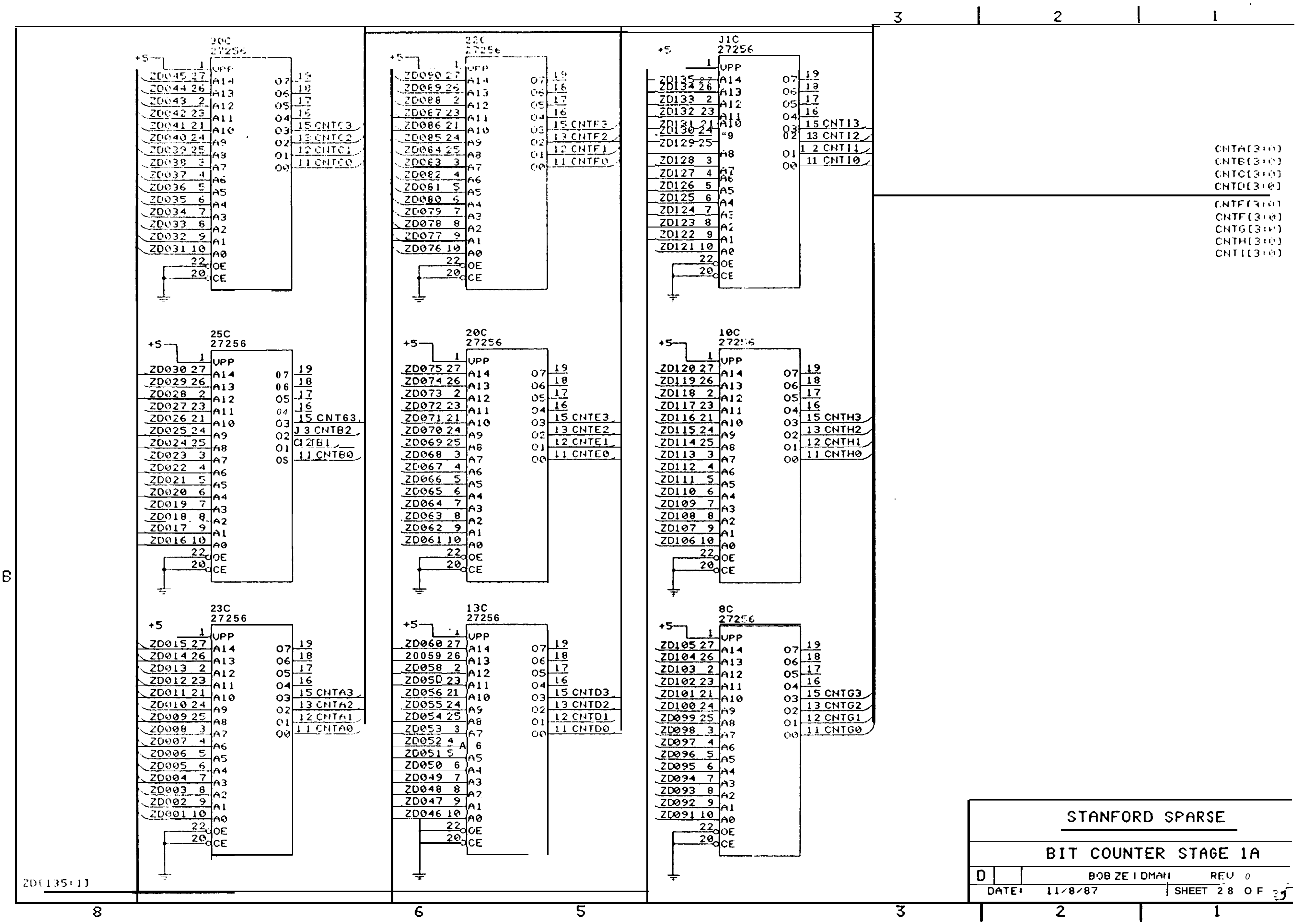


STANFORD SPARSE

ALU LOGIC 7

D FRANK FLYNN REV 0

DATE: 7/19/88 SHEET 27 OF 35

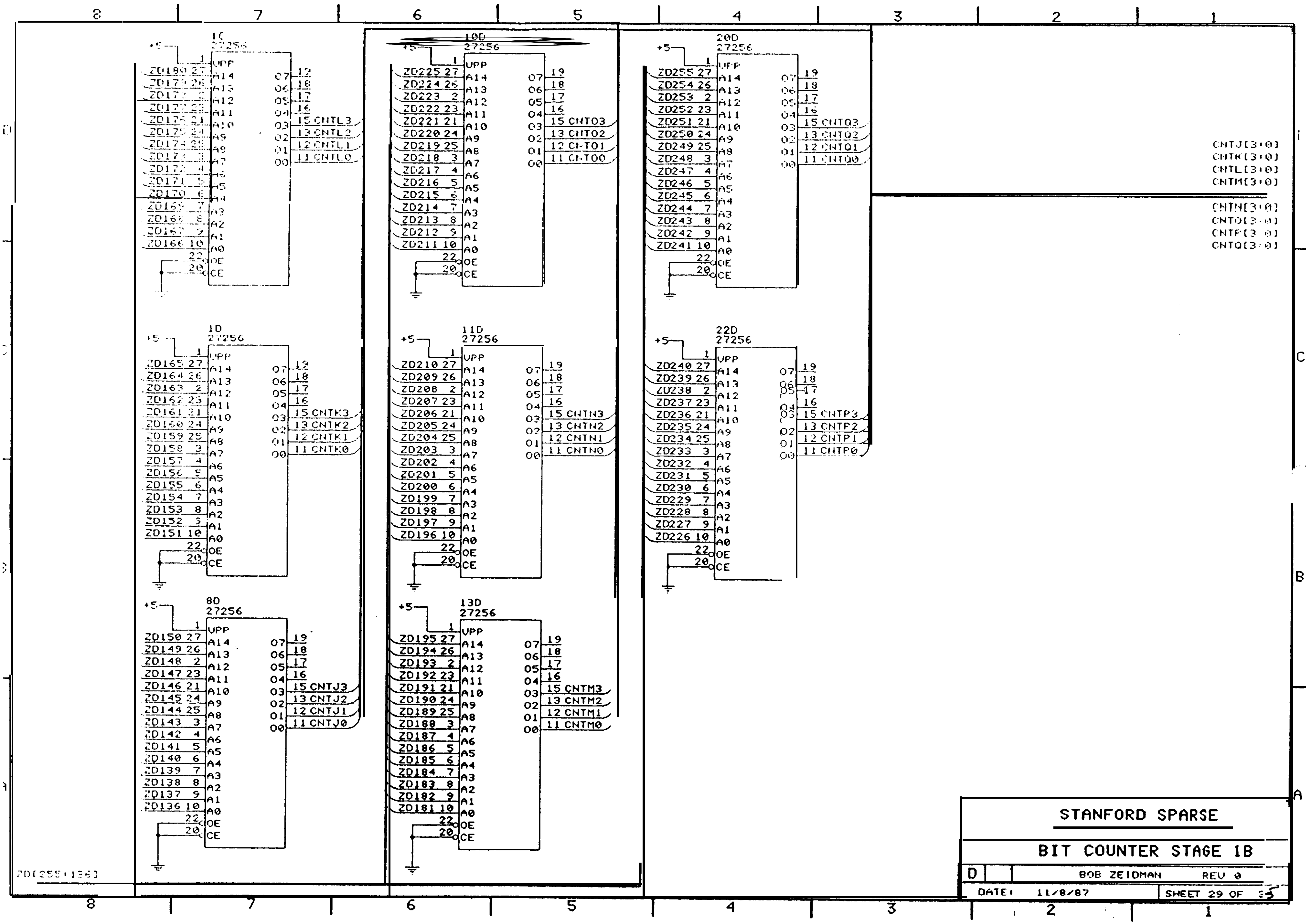


STANFORD SPARSE

BIT COUNTER STAGE 1A

D | BOB ZEIDMAN | REV 0

DATE: 11/8/87 | SHEET 28 OF 35

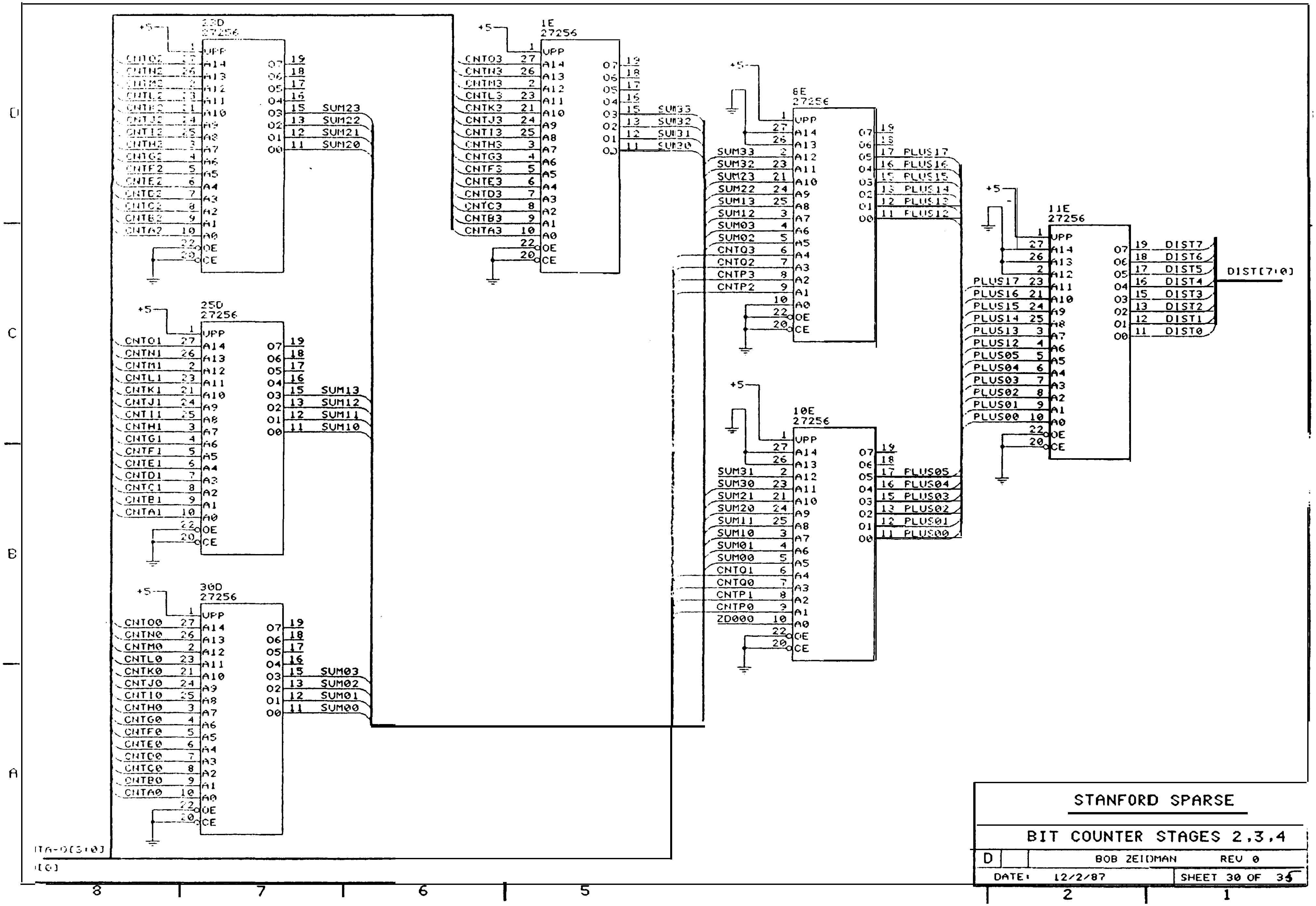


STANFORD SPARSE

BIT COUNTER STAGE 1B

D | BOB ZEIDMAN | REV 0

DATE: 11/8/87 | SHEET 29 OF 35

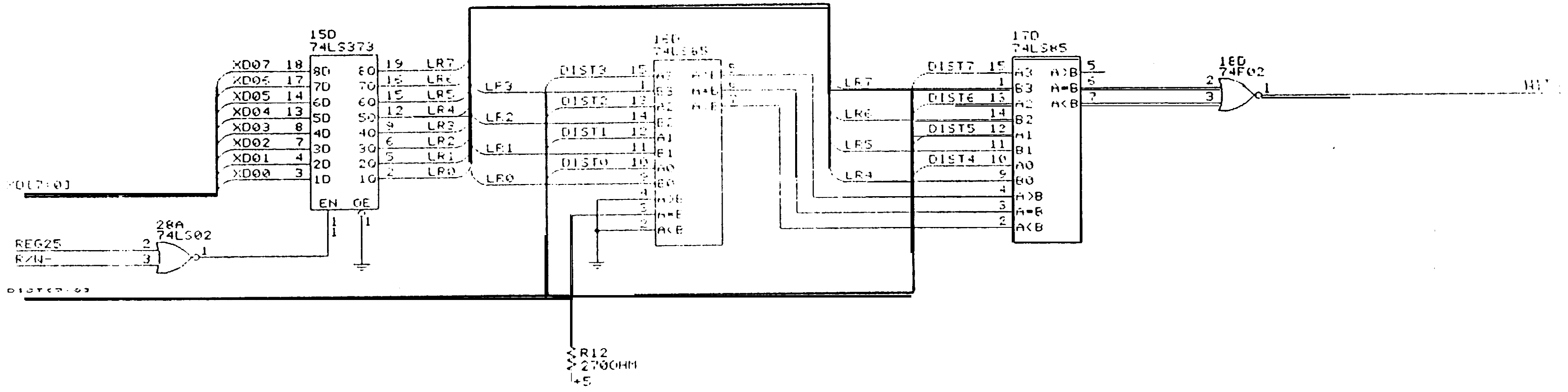


STANFORD SPARSE

BIT COUNTER STAGES 2,3,4

D | BOB ZEIDMAN | REV 0

DATE: 12/2/87 | SHEET 30 OF 35



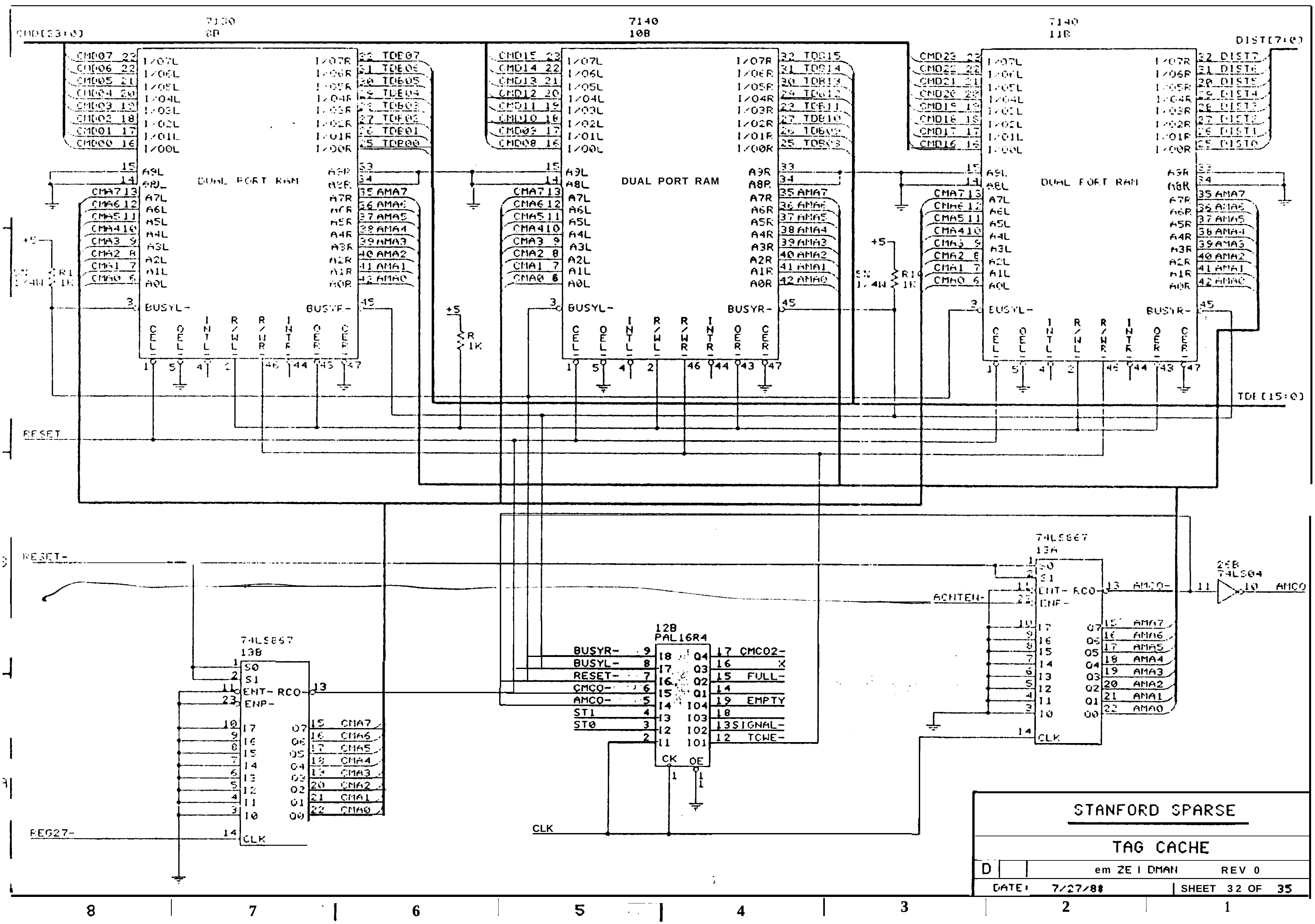
STANFORD SPARSE

BIT COUNTER COMPARE

D | | BOB ZEIDMAN | REV 0

DATE' 11/8/87 | SHEET 31 OF

8 7 6 5 4 3 2 1

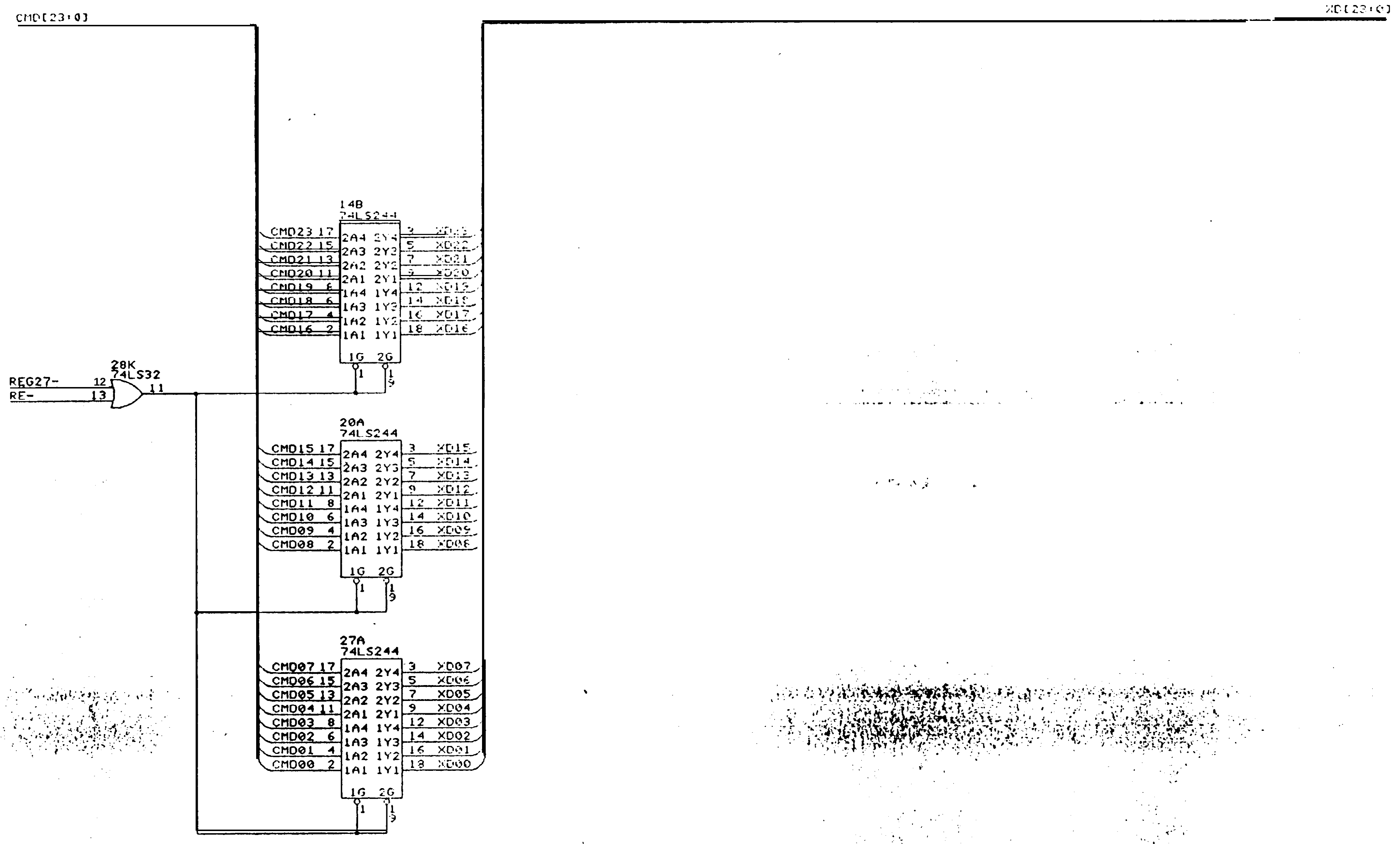


STANFORD SPARSE

TAG CACHE

D | em ZE I DMAN | REV 0

DATE: 7/27/88 | SHEET 32 OF 35

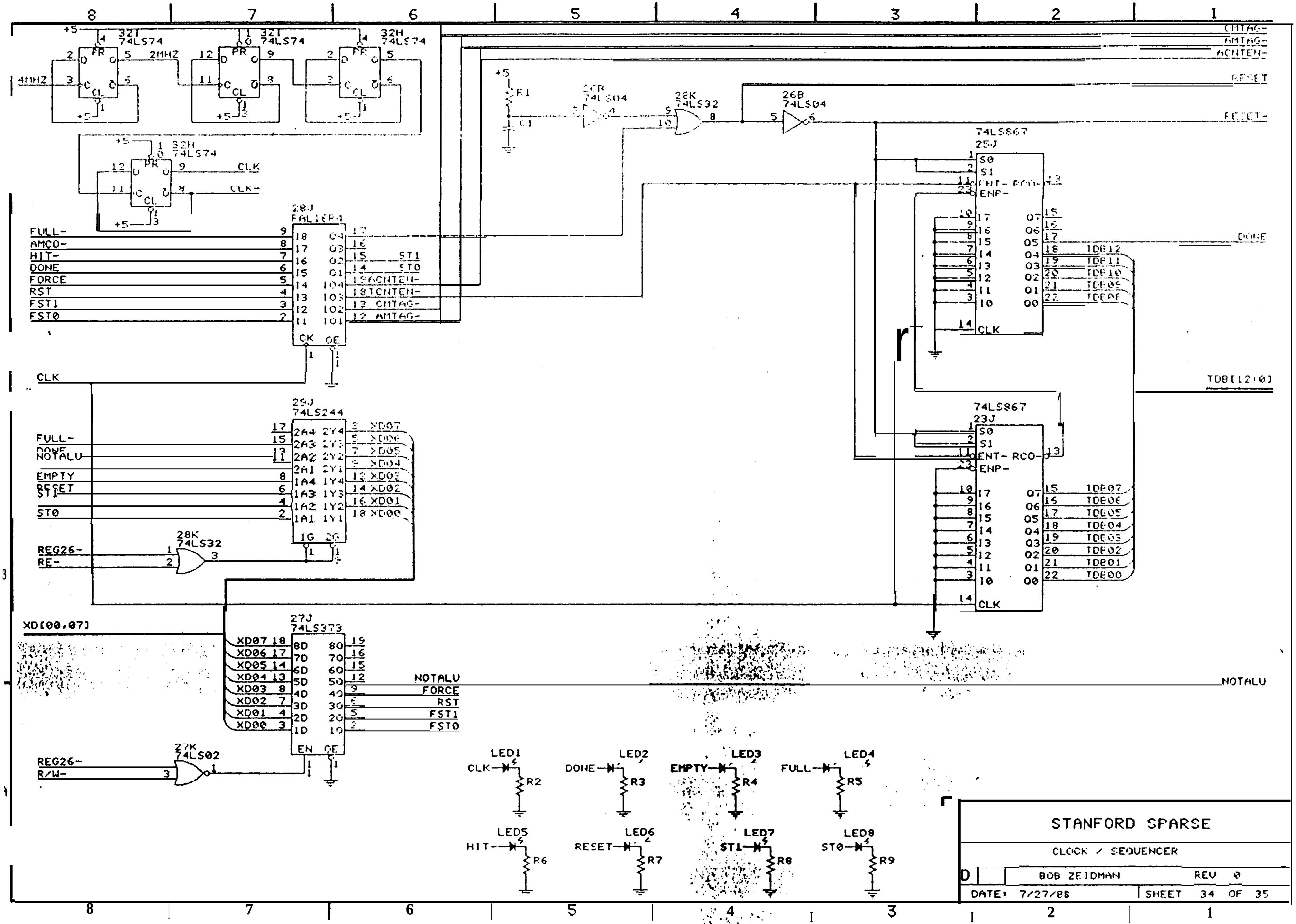


STANFORD SPARSE

TAG CACHE OUTPUT DATA

D | ERIC LOCHNER | REV 0

DATE: 7/21/88 | SHEET 33 OF 35



STANFORD SPARSE

CLOCK / SEQUENCER

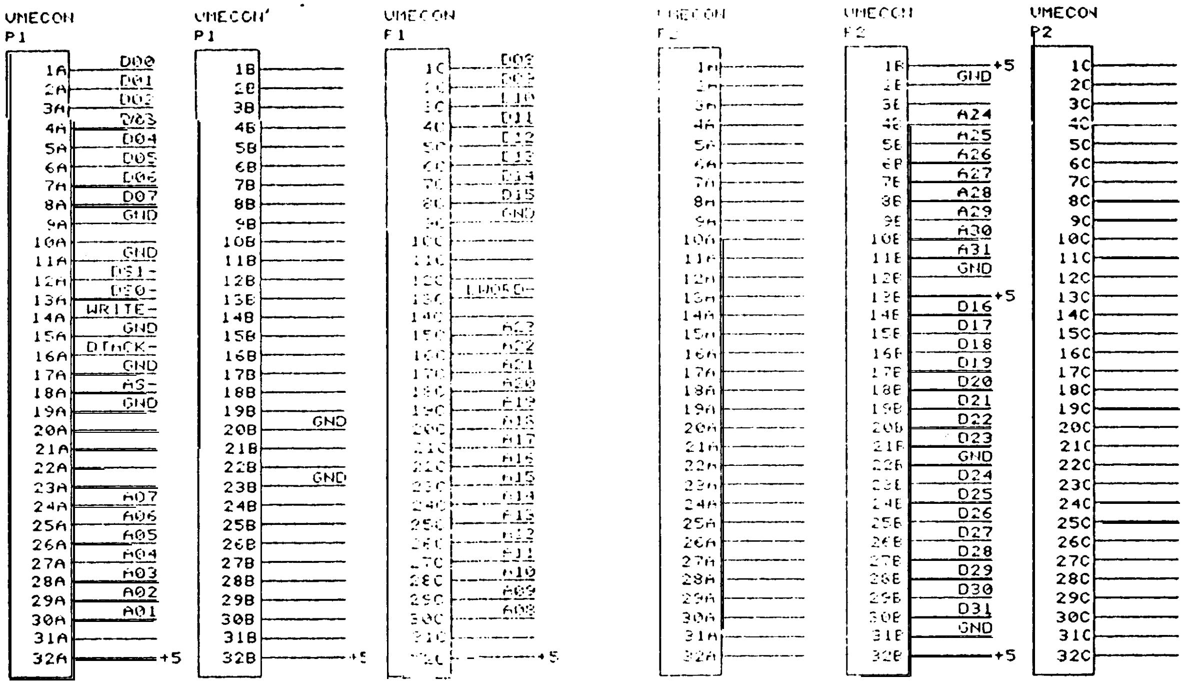
D

BOB ZEIDMAN

REV 0

DATE: 7/27/88

SHEET 34 OF 35


VMECON P1
1A 2A 3A 4A 5A 6A 7A 8A 9A 10A 11A 12A 13A 14A 15A 16A 17A 18A 19A 20A 21A 22A 23A 24A 25A 26A 27A 28A 29A 30A 31A 32A
D00 D01 D02 D03 D04 D05 D06 D07 GND GND DS1- DS0- WRITE- GND DTACK- GND AS- GND A07 A06 A05 A04 A03 A02 A01 +5
VMECON P1
1B 2B 3B 4B 5B 6B 7B 8B 9B 10B 11B 12B 13B 14B 15B 16B 17B 18B 19B 20B 21B 22B 23B 24B 25B 26B 27B 28B 29B 30B 31B 32B
GND GND +5
VMECON F1
1C 2C 3C 4C 5C 6C 7C 8C 9C 10C 11C 12C 13C 14C 15C 16C 17C 18C 19C 20C 21C 22C 23C 24C 25C 26C 27C 28C 29C 30C 31C 32C
D08 D09 D10 D11 D12 D13 D14 D15 GND LWORD- A23 A22 A21 A20 A19 A18 A17 A16 A15 A14 A13 A12 A11 A10 A09 A08 +5
VMECON F2
1A 2A 3A 4A 5A 6A 7A 8A 9A 10A 11A 12A 13A 14A 15A 16A 17A 18A 19A 20A 21A 22A 23A 24A 25A 26A 27A 28A 29A 30A 31A 32A
VMECON F2
1B 2B 3B 4B 5B 6B 7B 8B 9B 10B 11B 12B 13B 14B 15B 16B 17B 18B 19B 20B 21B 22B 23B 24B 25B 26B 27B 28B 29B 30B 31B 32B
+5 GND A24 A25 A26 A27 A28 A29 A30 A31 GND +5 D16 D17 D18 D19 D20 D21 D22 D23 GND D24 D25 D26 D27 D28 D29 D30 D31 GND +5
VMECON P2
1C 2C 3C 4C 5C 6C 7C 8C 9C 10C 11C 12C 13C 14C 15C 16C 17C 18C 19C 20C 21C 22C 23C 24C 25C 26C 27C 28C 29C 30C 31C 32C

STANFORD SPARSE

VME BUS CONNECTORS

D BOB ZEIDMAN REV 0

DATE: 8/11/88 SHEET 35 OF 35