# Highlights of Design F

23-september-81

Design F is the fifth design I've made in the last couple weeks. The designs up to and including design F have sported increasing communication bandwidth. I have now achieved a bandwidth comparable to that of DMA channels, with the added flexibility of 4-cycle communication.

Design F also incorporates a feature for global communication. A controlling processor (ISBC 86/12) will provide a number of utility functions to the cube processors: a clock, a reset signal, a periodic interrupt to refresh rams, and general communication signals.

There are two general communicaton paths: an path from the controlling processor to the cube processors, and a shared path from each of the cube processors to the controlling processor. There are 3 data signals generated by the controlling processor that can be read by each of the cube processors. There is a 5 bit open collector bus connecting all of the cube processors to the controlling processor.

In addition, a switch is provided on each processor that can be read by the respective processors[1]. A LED is provided on each processor that can be illuminated under processor control[2].

These capabilities, although somewhat irregular, allow some important functions. The periodic interrupt can be used to synchronize messages transmitted over the three input lines. The open collector output lines can be used for such purposes as deadlock detection[3] The LED will permit visual identification of a board if its position in the array is known (such as identification of defective boards). The switch will permit identification of a board's position in the array given that its physical position is known (i.e. the button is pressed).

I may be slightly, or greatly, off in my design. I do not rule out the possibility that the optimal design is with UART chips or FIO chips, but my inspirations in other directions have not been as good. There is also room for fine tuning of this design - it would be a valuable contribution to demonstrate how to save even one chip. Constructive advice and help will be greatly appreciated.

[1] It is the fourth input bit

[2] It is the sixth output bit

[3] If a processor is idle it releases the line - if all processors are idle the line floats high and deadlock is assumed.

DeBenedictis

## Highlights of Design F

23-september-81

Design F is the fifth design I've made in the last couple weeks. The designs up to and including design F have sported increasing communication bandwidth. I have now achieved a bandwidth comparable to that of DMA channels, with the added flexibility of 4-cycle communication.

Design F also incorporates a feature for global communication. A controlling processor (ISBC 86/12) will provide a number of utility functions to the cube processors: a clock, a reset signal, a periodic interrupt to refresh rams, and general communication signals.

There are two general communicaton paths: an path from the controlling processor to the cube processors, and a shared path from each of the cube processors to the controlling processor. There are 3 data signals generated by the controlling processor that can be read by each of the cube processors. There is a 5 bit open collector bus connecting all of the cube processors to the controlling processor.

In addition, a switch is provided on each processor that can be read by the respective processors[1]. A LED is provided on each processor that can be illuminated under processor control[2].

These capabilities, although somewhat irregular, allow some important functions. The periodic interrupt can be used to synchronize messages transmitted over the three input lines. The open collector output lines can be used for such purposes as deadlock detection[3] The LED will permit visual identification of a board if its position in the array is known (such as identification of defective boards). The switch will permit identification of a board's position in the array given that its physical position is known (i.e. the button is pressed).

I may be slightly, or greatly, off in my design. I do not rule out the possibility that the optimal design is with UART chips or FIO chips, but my inspirations in other directions have not been as good. There is also room for fine tuning of this design - it would be a valuable contribution to demonstrate how to save even one chip. Constructive advice and help will be greatly appreciated.

[1] it is the fourth input bit

[2] it is the sixth output bit

[3] if a processor is idle it releases the line - if all processors are idle the line floats high and deadlock is assumed.

# Interprocessor Communications Principles
# (design f)

This document describes an interprocessor communication proposal for the cube processor. This proposal is implemented in design F, 23-september-81.

## Design Goals

Speed is the primary design goal. Problems can be devised where the performance of the entire machine will be proportional to the speed of the communications. There are, however many important problems where the communications speed is of no practical importance.

Cost is a factor that is intertwined with the other goals and must be considered. The goal of the machine is to perform calculations at minimal cost. The speed or cost of an individual processor is not overwhelmingly important: a speed increase of 50% is unacceptable if the cost increase is 60%. The proposed design understands the importance of low cost, and disregards more powerful designs if the cost is much higher.

A less troublesome design goal, but an equally important one, is that the design be amenable to a high level software communications system. This goal reqires that attention be given to such questions of queueing and deadlock at the design proposal phase. Were these questions to be ignored at this point then the extra software overhead required might anull any gain in hardware performance.

## Full Handshake Communications

The last design goal will be satisfied by a full handshake communications system. Full handshake means that a message cannot be transmitted by a sender until the receiver is ready to accept it. Many common communications schemes are not like this: UARTS will overrun if the receiver is full, HPIB interfaces are similar.

Communications proposed for this design will involve FIFOs. A fifo is an asynchronous device that can be loaded and unloaded with data words. The device is capable of storing a fixed number of data words, and when it is full it indicates so and refuses to load further data.

Our implementation will have two fifos per communication link. A sending processor will examine one fifo to determine if it can accept another message, and if so will load another message. The message will be stored in fifos until a receiving processor checks its fifo to determine if at least one message is available. If a message is available it is read and removed from the receiver fifo. The transmitter and receiver fifos are connected by a communications link (a cable).

If the receiver stops reading from its fifo then the fifos may fill completely. When this happens the transmitter will stop transmitting.

## Data Path Size

There are two very important considerations concerning the number of bits in the data words. Wide data paths are undesirable because they require thick bundles of wire and expensive connectors. On the contrary, small data paths are undesirable because a processor should not be burdened with assembling small data words.

This proposal addresses these conflicting requirements by using a small data word on the communicatons link and a large word at the processor interface. There is special hardware to convert between small and large words.

The communication path that traverses the cables is four bits wide. The width of the cable is much larger than this, however: the link is bidirectional, and there are two handshake lines in each direction. Including four ground conductors, a 16 conductor cable is required.

The fifos are implemented as 16x4 fifos.

The communication to the processor[1] is in 16 bit words. The processor will read or write an entire word in one cycle. Special hardware translates the 16 bits into 4 nibbles[2] and cycles these individually to the fifos.

## Interrupts

When a message is available in a fifo the processor in interrupted. The processor can then read the entire message from the fifo. Since the fifos are 4 bits wide they could conceviably contain 0-16 nibbles. If the processor did not know how many nibbles were in the fifos it would be required to check the fifo empty bit between each nibble. Such checking is unacceptably time consuming and has been eliminated.

The strategy for generating an interrupt is that a fifo be entirely full. If the fifo is full then there are 16 nibbles in the fifo and the processor can reliably read all 16 nibbles without looking.

The penalty for this speedup is that messages less than a full fifo long cannot be reliably sent. Such a message would partially full the fifo in the receiver and would remain there until another message came along to push it through.

A full sized message is now 64 bits.

## Details of the Inter-fifo Communication

Communication over the cables occurs with 4-cycle asynchronous handshake timing. The fifo chips (almost) support the 4-cycle without additional logic. The only chips needed in addition to the fifos (besides one gate to solve the 4-cycle problem) are buffer chips to provide drive and hysterisis for the long cable.

[1] an 8086

[2] nibble is a term for a 4-bit word

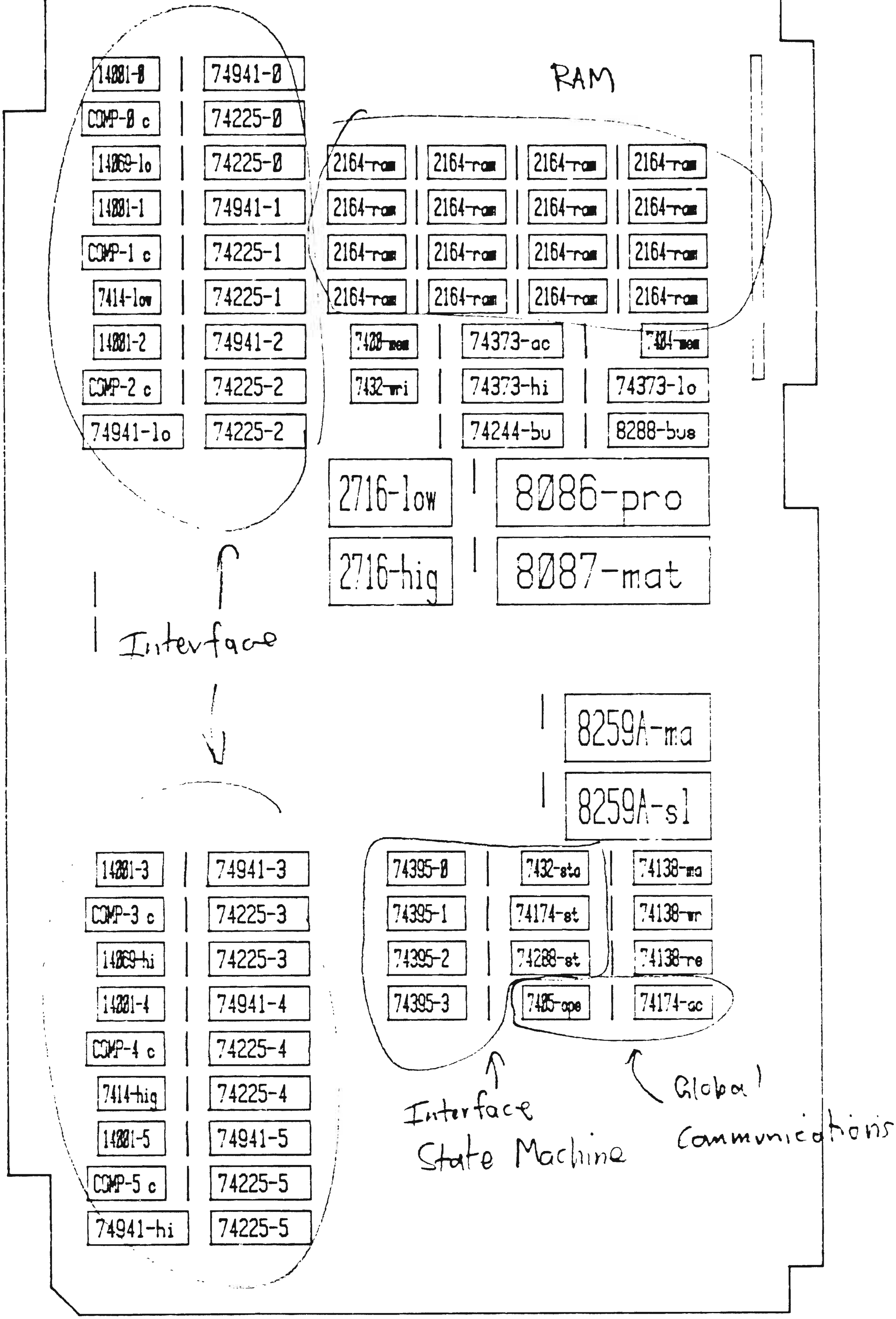
## Details of the 4 to 16 Bit Conversion

Conversion from nibbles to words is accomplished with a 4 bit wide, 4 bit long shift register. Conversion from nibbles to words is accomplished by clocking the register four times with a new nibble at the input each time. After the four clocks 16 bits are available at the parallel outputs of the shift register. Conversion from words to nibbles is accomplished by loading the 16 bits of the shift register in parallel and the shifting the register four times. Each shift load one nibble into a fifo.

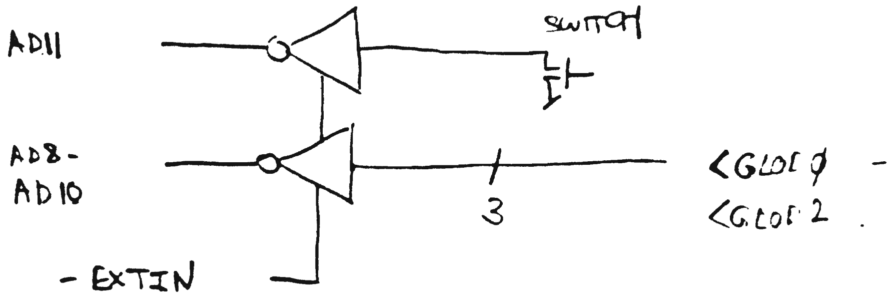
Clocking for these conversions is generated by a state machine. The state machine is activated by either a io-read or io-write to any of the fifo addresses. The state machine, clocked by the system clock, immediately removes the ready line to the processor to extend the cycle. The state machine will then operate for 5 clock cycles, shifting the register four times and reading or writing in parallel. Following the five cycles the state machine releases the CPU and waits for another fifo operation.

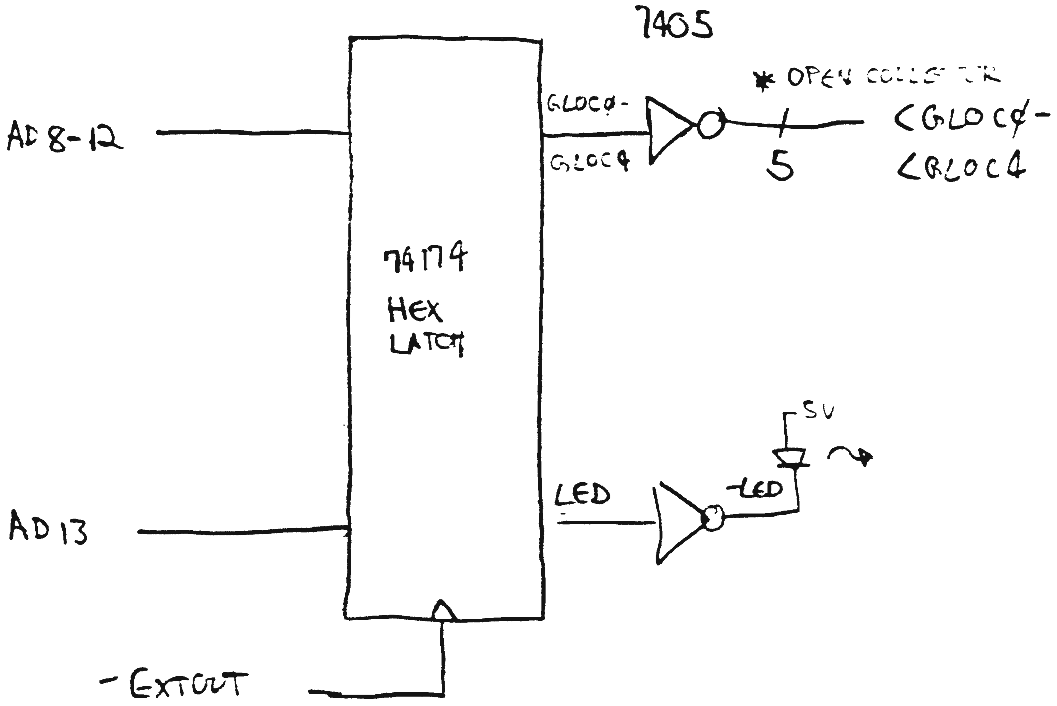
The amount of delay of the processor is minimal. Normally, a cycle is 3 clock periods; the state machine extends the cycle to 6[3].

[3] I think, maybe a few more

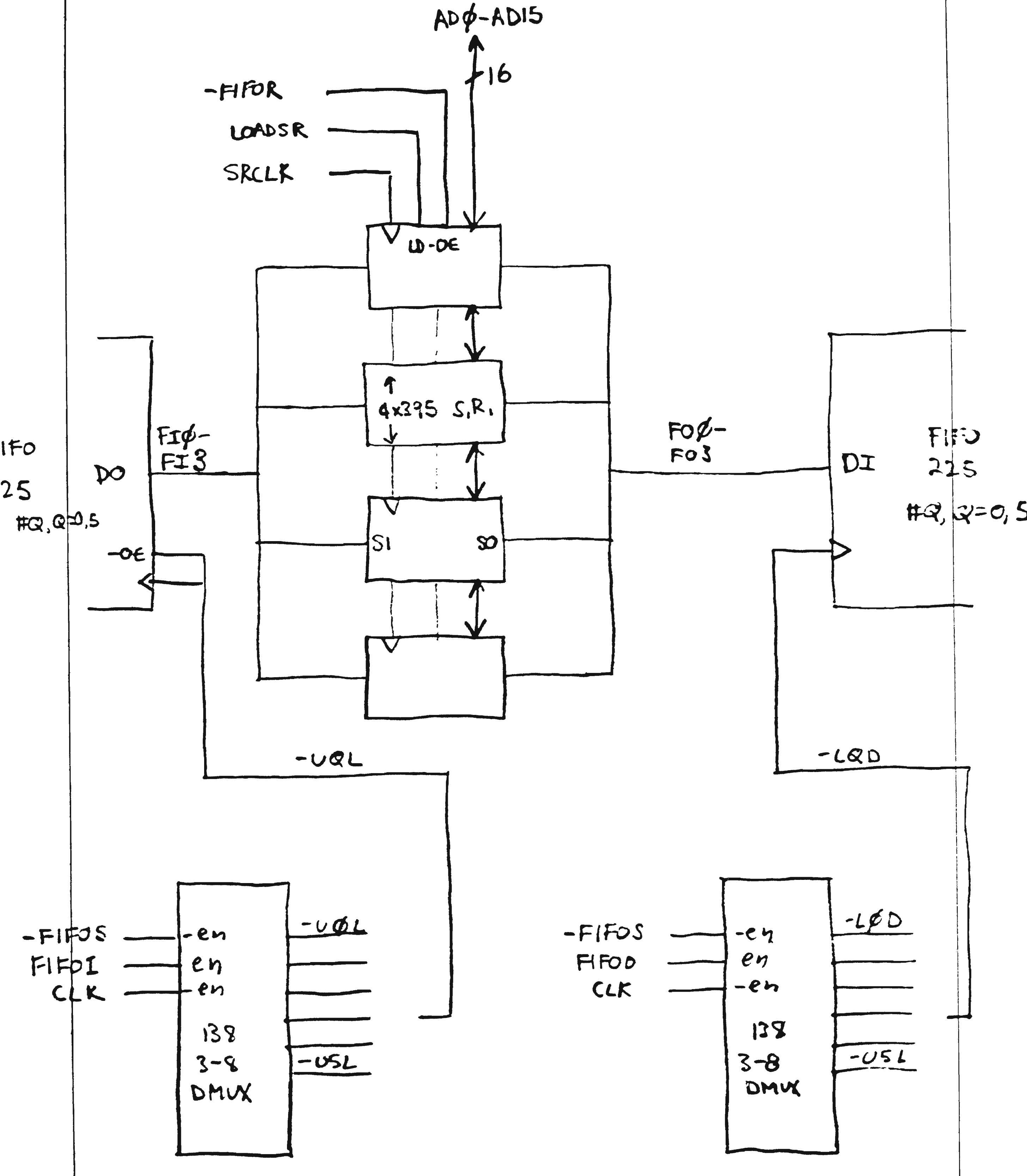

RAM
14001-Ø
74941-Ø
COMP-Ø c
74225-Ø
14069-lo
74225-Ø
14001-1
74941-1
COMP-1 c
74225-1
7414-low
74225-1
14001-2
74941-2
COMP-2 c
74225-2
74941-lo
74225-2
2164-ram
2164-ram
2164-ram
2164-ram
2164-ram
2164-ram
2164-ram
2164-ram
2164-ram
2164-ram
2164-ram
2164-ram
2164-ram
2164-ram
2164-ram
2164-ram
7408-mem
74373-ac
7404-mem
7432-wri
74373-hi
74373-lo
74244-bu
8288-bus
2716-low
8Ø86-pro
2716-hig
8Ø87-mat
Interface
8259A-ma
8259A-sl
14001-3
74941-3
COMP-3 c
74225-3
14069-hi
74225-3
14001-4
74941-4
COMP-4 c
74225-4
7414-hig
74225-4
14001-5
74941-5
COMP-5 c
74225-5
74941-hi
74225-5
74395-Ø
7432-sto
74138-ma
74395-1
74174-st
74138-wr
74395-2
74288-st
74138-re
74395-3
7485-ope
74174-ac
Interface
State Machine
Global
Communications

# ACCESSORIES

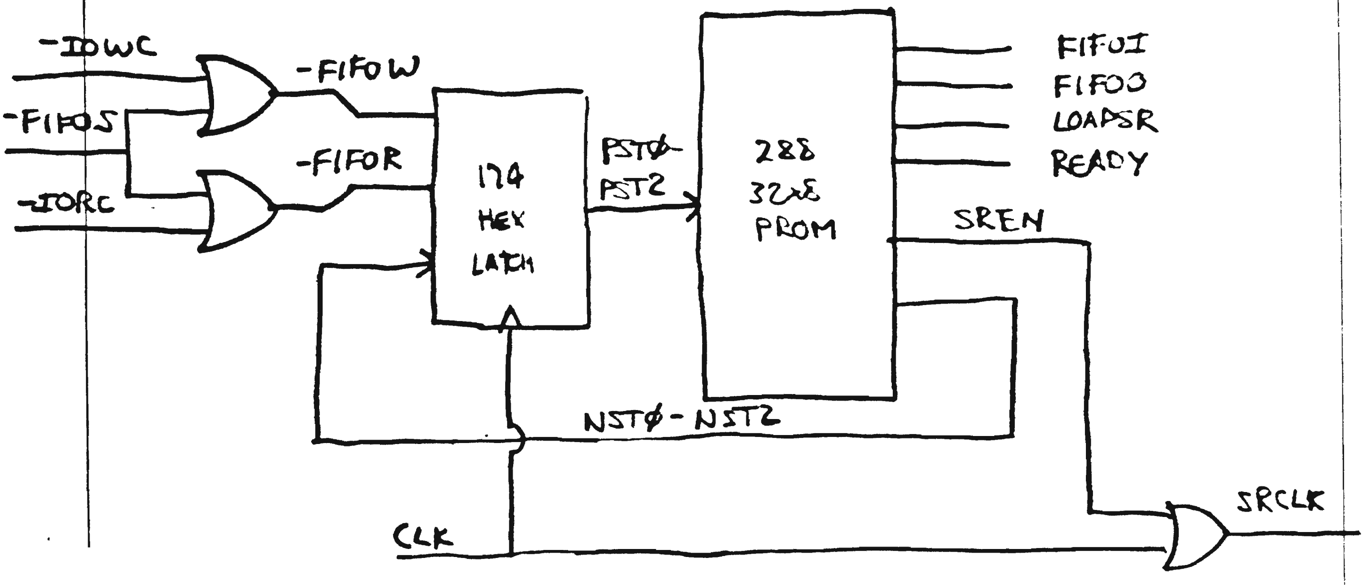


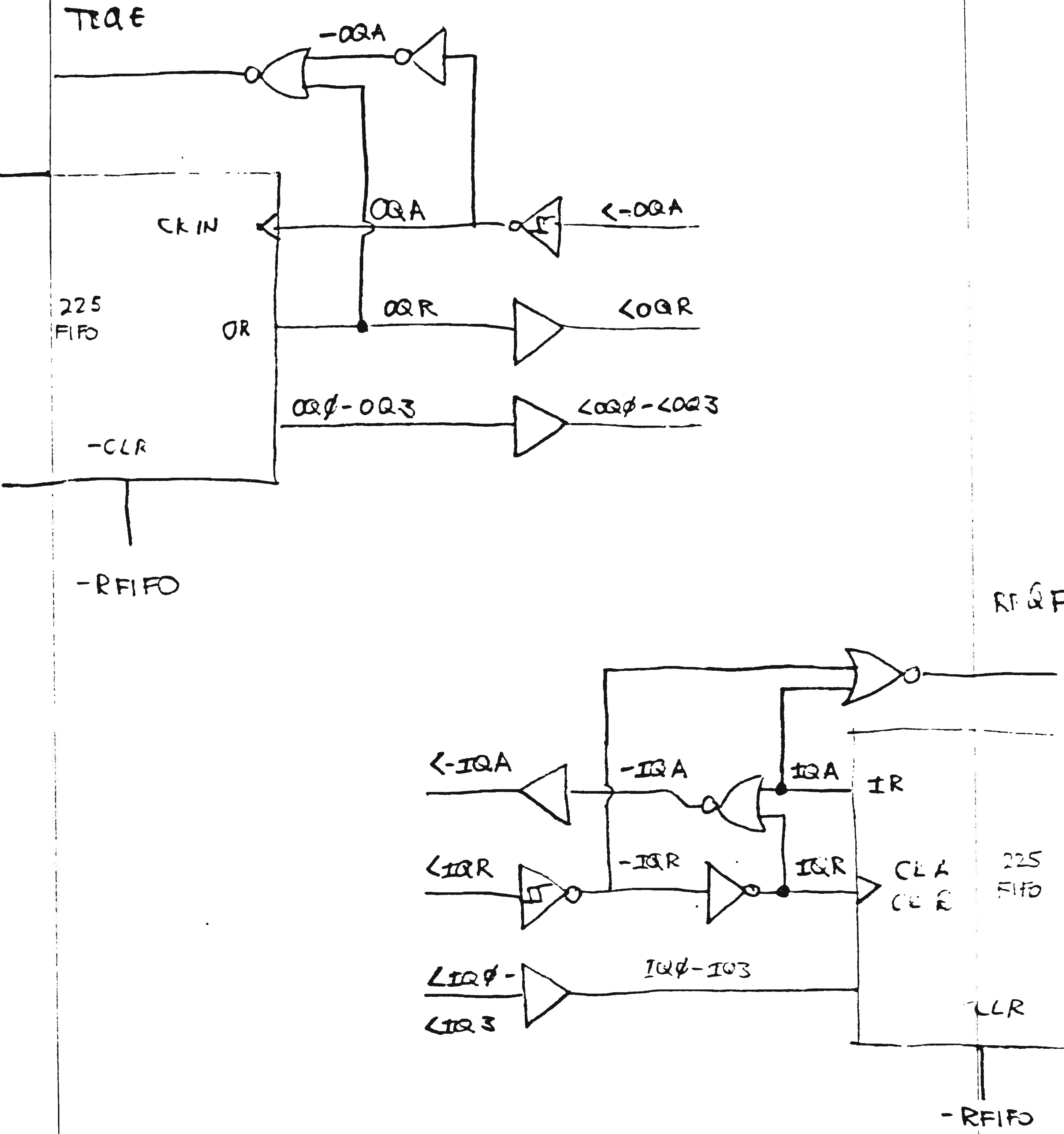


# FIFO 4-16-4 Converter



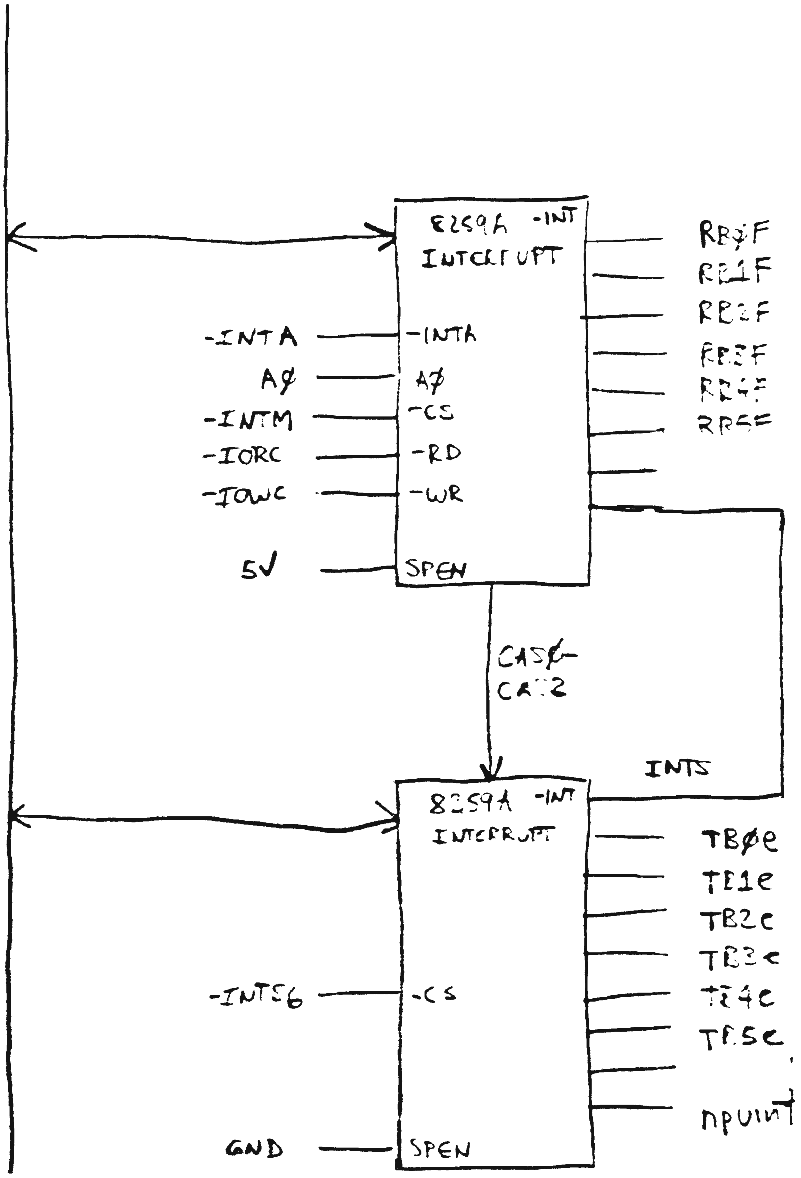
# FIFO State Machine



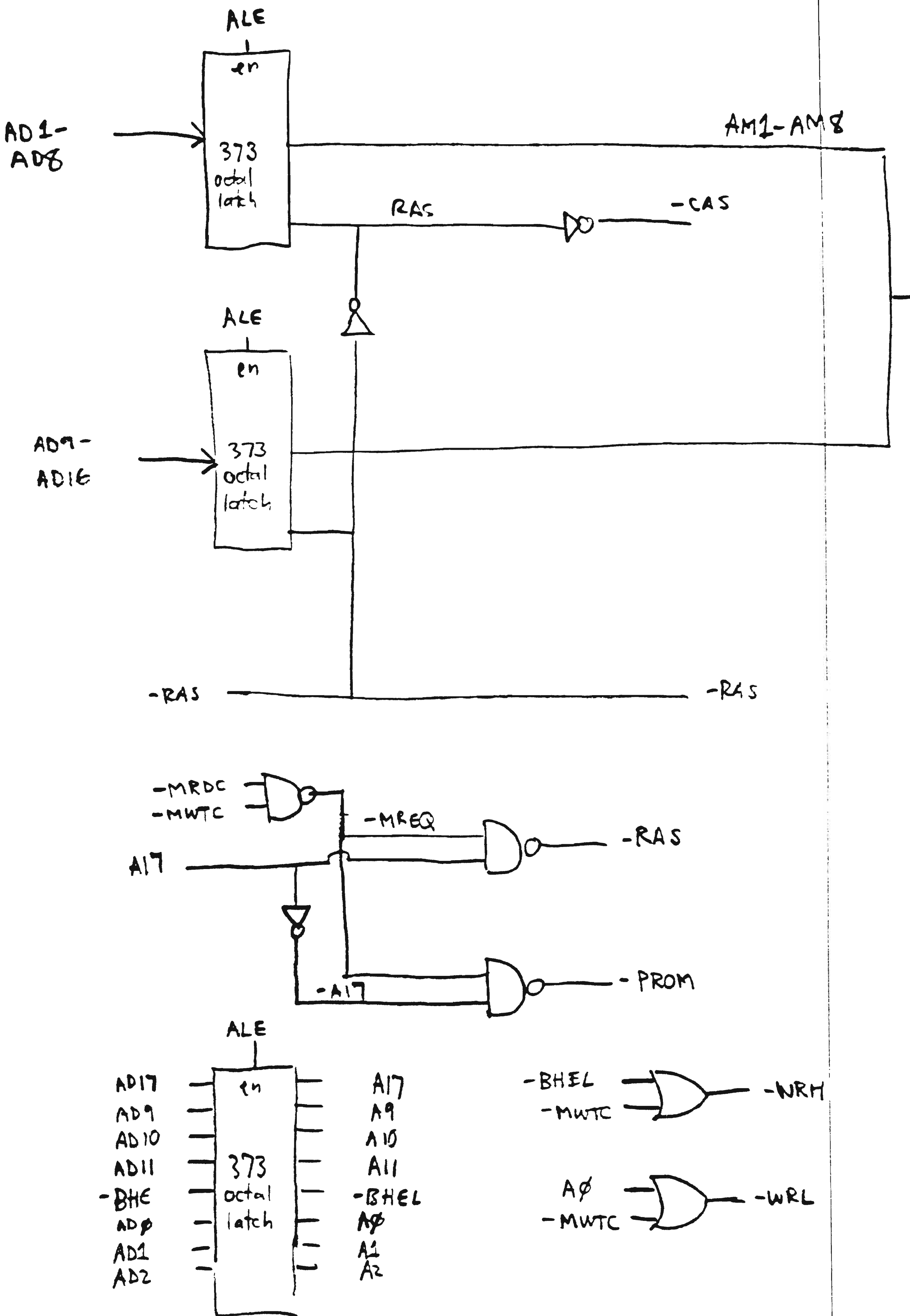


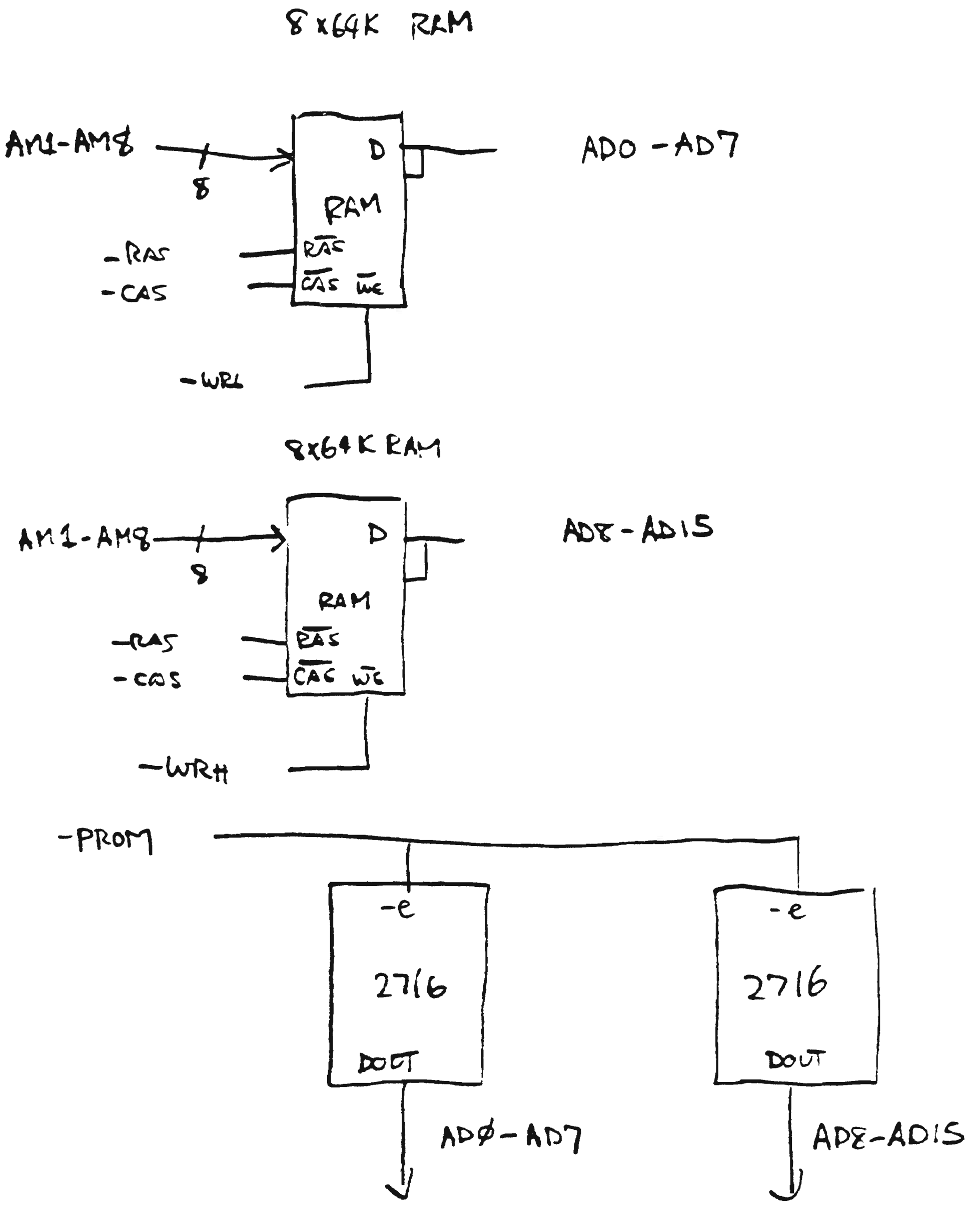
TRQE
-OQA
CK IN
OQA
<-OQA
225
FIFO
OR
OQR
<OQR
OQØ-OQ3
<OQØ-<OQ3
-CLR
-RFIFO
RFQF
<-IQA
-IQA
IQA
IR
<IQR
-IQR
IQR
CLK
CLR
225
FIFO
<IQØ-
<IQ3
IQØ-IQ3
-CLR
-RFIFO

8259c

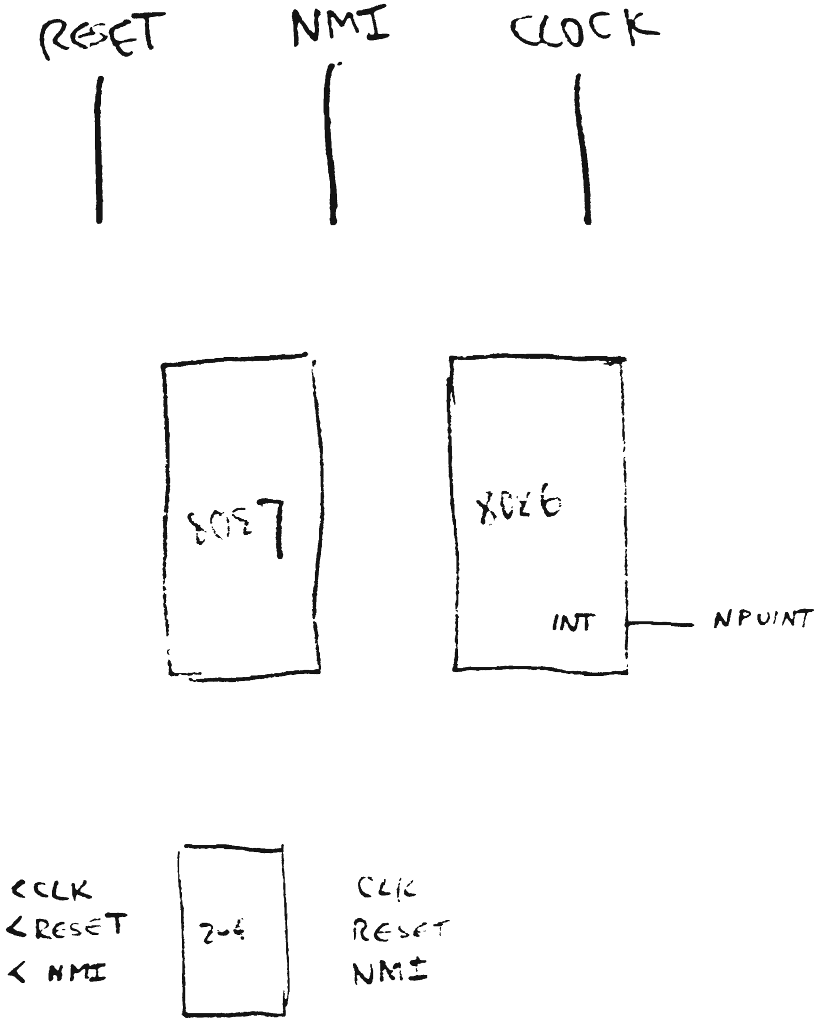

8259A -INT
INTERRUPT
-INTA
AØ
-INTM
-IORC
-IOWC
5V
-INTA
AØ
-CS
-RD
-WR
SPEN
RBØF
RB1F
RB2F
RB3F
RB4F
RB5F
CASØ-
CAS2
INTS
8259A -INT
INTERRUPT
-INTS6
-CS
GND
SPEN
TBØe
TB1e
TB2c
TB3c
TB4c
TB5c
npuint

# RAM/PROM TIMING




8 x64K RAM
AM1-AM8
8
D
RAM
RAS
CAS
WE
ADO -AD7
-RAS
-CAS
-WRL
8x64K RAM
AM1-AM8
8
D
RAM
RAS
CAS
WE
AD8-AD15
-RAS
-CAS
-WRH
-PROM
-e
2716
DOUT
ADØ-AD7
-e
2716
DOUT
AD8-AD15

# CPUs


RESET
NMI
CLOCK
8087
8086
INT
NPUINT
<CLK
<RESET
< NMI
2-4
CLK
RESET
NMI

IQA

-IQA

-LDQ

| | |
|---|---|
| 1 | GND |
| 2 | 0∅ |
| 3 | 01 |
| 4 | 02 |
| 5 | 03 |
| 6 | GND |
| 7 | OR |
| 8 | IA |
| 9 | OA |
| 10 | IR |
| 11 | GND |
| 12 | I3 |
| 13 | I2 |
| 14 | I1 |
| 15 | I∅ |
| 16 | GND |