2-1-1992

The RIT IEEE-488 buffer design

John Connor

The RIT IEEE-488 BUFFER DESIGN

by

John Connor

A Thesis Submitted

in

Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

in

Electrical Engineering

Approved by:

Prof. Lynn Fuller
(Thesis Advisor)

Prof. George A. Brown

Prof. James E. Palmer

Prof. R. Unnikrishnaw
(Department Head)

Department of Electrical Engineering

College of Engineering

Rochester Institute of Technology

Rochester, New York

FEBRUARY 1992

R·I·T

Rochester Institute of Technology

Wallace Library
Post Office Box 9887
Rochester, New York 14623-0887
716-475-2562 Fax 716-475-6490

SAMPLE statements to reproduce an RIT thesis:

PERMISSION GRANTED
Title of thesis The RIT IEEE-488 Buffer Design

I _____ hereby **grant** permission to the Wallace Memorial Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date: 2/27/92 **Signature of Author:** John Connor

PERMISSION FROM AUTHOR REQUIRED
Title of thesis _____

I _____ **prefer to be contacted** each time a request for reproduction is made. I can be reached at the following address:

_____
_____
_____

PHONE: _____

Date: _____ **Signature of Author:** _____

PERMISSION DENIED
Title of thesis _____

I _____ hereby **deny** permission to the Wallace Memorial Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part.

Date: _____ **Signature of Author:** _____

To Marilyn,
Thank you for your patience, support, & love during the school years.

1. Abstract

This document describes the design of an NMOS ASIC used to control an RIT IEEE-488 Buffer previously designed by the author. Past designs used discrete components to implement an asynchronous controller and a synchronous, one-hot controller. The present design utilizes a multiple controller architecture incorporated within the ASIC. The ASIC is used to control bus protocol, bus transceivers, and memory.

At power-up, the buffer configures itself as an active listener on the bus and waits for a talker to initiate communication. The buffer accepts a data file (a plot file for example) from the talker, then takes control of the bus, addresses a listener, transfers the stored data to the listener, unaddresses the listener, releases the bus, and finally, reassumes the active listener configuration.

The RIT IEEE-488 buffer can realize time savings for a user in a controllerless system. The buffer accepts data from a talker in a matter of seconds and then takes on the chore of driving a slow listener. Thus, the talker is returned quickly to the operator for further use. At present, the buffer isn't queueable - it **cannot** accept another data file until it completes the transfer of the present file.

The author has also added five nmos cells (schematic/layout) into the '/user/pub' directory on the Apollo workstations in the Computer Engineering Department's VLSI LAB at RIT. Cell names are VSCLK, SYNC, CLOCK_GEN_STACK, PAD_TRISTATE, and PAD_TRISTATE_BUFFERED. All five cells have been simulated and successfully run through DRC, ERC, and LVS checks.

2. WARNINGS: Don't FRY your equipment! Check them out!

1. The RIT IEEE-488 Buffer **cannot** be used in systems with a controller. It can only be used in a "controllerless system" provided the two warnings listed below aren't violated. Refer to the section 4 for a discussion on systems with and without controllers.

2. Some talkers do not release the bus when done with file transfer in a controllerless system. These talkers continue to drive one or more bus signals LOW. The RIT IEEE-488 buffer **cannot** be used with such talkers. Perform the following check of your controllerless system without the RIT IEEE-488 Buffer installed. Measure the voltage level on the five bus management lines, the three bus handshake lines and the eight bus data lines after transfer of a data file from talker to listener. [See section 4 for details on the various bus signals and connector pin out.] If all voltages are HIGH (2.0 volts or greater) except for NDAC which should be LOW (less than 0.8 volts), then the RIT IEEE-488 Buffer can be incorporated into the system provided the warning listed below isn't violated. NDAC is driven LOW by the listener since it is in listen only mode. If the RIT IEEE-488 Buffer is installed, the listener must be taken out of listen only mode and given an address, between 0 and 30.

3. A talker's bus interface *may* still be in talk mode after file transfer is complete. However, if it isn't driving any bus signals LOW, it acts as if it is OFF the bus. The RIT IEEE-488 Buffer can be incorporated into a controllerless system using this type of talker provided its bus drivers **can** be pulled LOW by the bus drivers of the buffer. If the talker's drivers meet the requirements for bus drivers established in the IEEE Standard 488-1978, the issue mentioned above should not be a concern. See section 4 for specifications on IEEE-488 drivers/receivers.

4. The author has only looked at a few Hewlett Packard and Tektronix products. Please be careful when using this buffer with your equipment.

TABLE OF CONTENTS

TABLE OF CONTENTS [continued]

Section 8 The RIT IEEE-488 Buffer Controller (ASIC) [continued]

TABLE OF CONTENTS [continued]

LIST OF FIGURES

LIST OF TABLES

3. Introduction

The author came up with the idea of designing an IEEE-488 buffer after having to wait for up to several minutes while a parameter analyzer dumped waveform information to an attached plotter. This time was often spent "twiddling one's thumbs" since the parameter analyzer's front panel controls were locked out while driving the plotter. The RIT IEEE-488 Buffer is capable of accepting a file from a talker in a matter of several seconds (thus quickly freeing up the talker for further use) and then take on the chore of driving the a slow listener. The buffer accepts data from some source instrument and then addresses a listener, supplies that listener with the stored data, unaddresses the listener after data transfer is complete, and finally, returns itself to an active listener configuration. The buffer accepts a data from the talker during the *data input phase* of its operation. Once the buffer has accepted all the data from the talker, it enters the *data output phase* of its operation the addressing of the intended receiver, the transfer of the stored file, and the unaddressing of the listener. At present, the buffer can only accept and complete one file transfer at a time. Future design plans intend to make the buffer queueable but some time savings should be realized with the present implementation.

Two discrete prototype buffers (a asynchronous design and a synchronous, one-hot design) have been built and successfully tested by the author. The design discussed in this document uses a multiple controller architecture incorporated within an NMOS ASIC. The ASIC was designed, simulated, and laid out by the author using $\lambda = 2\ \mu m$ MOSIS rules. The ASIC is to be fabricated by the Microelectronic Engineering Department at RIT in its Class 1000 Clean Room. Although not part of the thesis, the ASIC will be tested at the chip level, and if found to work correctly, packaged, and installed into a buffer and tested.

The author has also added five nmos cells (schematic/layout) to the '/user/pub/pads' directory on the Apollo workstations in the Computer Engineering Department's VLSI LAB at RIT. The names of these cells are VSCLK, SYNC, CLOCK_GEN_STACK, PAD_TRISTATE, and PAD_TRISTATE_BUFFERED. All cells have been simulated and successfully run through DRC, ERC, and LVS checks. All of these cells are used in the design of the ASIC.

4. The IEEE 488.1 BUS [1,2,3,4]

The IEEE-488 interface, often referred to as the GPIB (general purpose interface bus), is an adaptation of the HP-IB (Hewlett Packard - interface bus). In June of 1987, the IEEE Standard 488-1978 was revised to the IEEE-488.1. At the same time, the IEEE-488.2 (Codes, Formats, Protocols, and Commands) was published. The IEEE-488.1 is the same as the old IEEE-488. The codes, formats, protocols, and commands described in the IEEE-488.2 are used on the IEEE-488.1.

Many electronic instruments have available from the manufacturer the option of being equipped with an IEEE-488 interface. Such instruments, connected together using cables meeting the specifications described in the IEEE Standard 488-1978, are capable of sending and/or receiving bus commands and device dependent data over the bus.

In some configurations, a system controller (typically a PC), is responsible for issuing bus commands and dictating which instrument will send data, the active talker, and which instrument(s) will receive the data, the active listener(s). There can only be one talker on the bus at a time but several listeners can exist on the bus simultaneously. Generally not every instrument on the bus participates in data transfer at the same time. The utility of such a system is that a program can be written to automate a task. For example, a PC acting as the active controller could instruct an oscilloscope to perform a measurement. It could then instruct the oscilloscope to become the active talker follow by an instruction to itself to become the active listener. This would allow transfer of the measurement data from the oscilloscope to the PC. Once the measurement data was received, the PC could resume the role of active controller, instruct a printer to become an active listener, and then instruct itself to become the active talker with the intent of getting a printout of the measurement data. Note that if the oscilloscope is capable of formatting the measurement data so that it is recognizable by the printer, then the transfer of data can be made directly between the oscilloscope and the printer; provided the active controller had addressed the scope to be the active talker and the printer to be the active listener.

In some applications no controller is used. An example of a controllerless system is an oscilloscope connected to a plotter via a GPIB capable. In such controllerless systems, a measurement can be taken and the collected data dumped over the bus to the attached plotter. The plotter must be addressed to its "listen-only" mode using an on board dip switch or front panel entry. A device so configured will listen to *all* activity on the bus and can not be taken out of listen mode by any command passed over the bus. In a system without a controller, the talker may be operated manually. After the operator has captured a waveform on the scope's crt, a front panel button on the oscilloscope is usually pressed to start data transfer to the plotter if a hardcopy of a waveform is desired. When sending data to the active listener, the talker's front panel controls are locked out (unusable) and the instrument is unavailable to an operator until the data transfer is complete, unless the transfer is aborted. A plotter with no onboard memory may accept a byte of data, interpret it, and execute the instruction before signaling that it is ready for another byte. An instruction such as pick up a pen or draw a line may take seconds to execute and adds directly to the total time the operator must wait before getting use of the measurement device again. Complex plots may take several minutes.

The RIT IEEE-488 Buffer is designed for use in such a controllerless system. The buffer is capable of accepting all measurement data in a matter of several seconds. This relieves the talker from the chore of driving a slow listener, and allows the user access to the talkers front panel controls in a manner of seconds instead of minutes.

The IEEE Standard 488-1978 defines ten Interface Functions (Table 1) which can be implemented to various degrees within an IEEE-488 compatible instrument. The RIT IEEE-488 Buffer is designed to emulate the SH, AH, T, L, and C Interface Functions but does **not** implement these functions according to the state diagrams provided in the IEEE Standard 488-1978. These state diagrams are overkill for the intended application. The authors physical circuit emulation of these functions is described in various parts of section 8.

TABLE 1 List of IEEE-488 Interface Functions

SH (Source Handshake)
AH (Acceptor Handshake)
T (Talker)
L (Listener)
SR (Service Request)
RL (Remote Local)
PP (Parallel Poll)
DC (Device Clear)
DT (Device Trigger)
C (Controller)

The IEEE-488.1 bus consists of eight data lines (DIO8-DIO1), five bus management lines (IFC, REN, ATN, EOI, SRQ), three handshaking lines (DAV, NRFD, NDAC), a signal ground line, a shield line, and six signal return lines. Table 2 presents a description of the important signals comprising the IEEE-488.1 bus. Figure 1 displays the pin out of an IEEE-488 connector. All signals on the IEEE-488.1 bus are active LOW. Data transfer is accomplished in a "byte-serial, bit-parallel fashion". That is, one eight bit word is sent in parallel across the bus's eight data lines at a time. The rate of data transfer is dependent on the talker and listener(s) with the slowest device setting the speed of transfer. The IEEE Standard 488-1978 restricts the maximum data transfer rate at 1M bytes per second, with typical applications using rates of 0.5M - 0.25M bytes per second. Handshaking (Fig. 2) between devices is done in an asynchronous manner. The listener(s) will indicate that it is ready for data, the talker will then make a byte of data available, the listener will accept the data byte, and the cycle will repeat.

Devices on the bus can be in one of three configurations: controller, talker, or listener. The RIT IEEE-488 buffer takes on each of these configurations at various times during the data file transfer. An instrument becomes the active controller when it sets the ATN (Attention) line TRUE (LOW). Only an active controller can assert the ATN line and only the system controller in a system with more than one controller can assert the REN (Remote ENable) or IFC (InterFace Clear) bus lines. ATN, REN, and IFC are examples of uniline bus commands. A uniline bus command has an individual bus wire dedicated to it.

TABLE 2 Description of the IEEE-488 Bus Signals

Signal	Name and Function
DIO8-DIO1	**Data Input/Output 8-1:** These eight signal lines are used by the bus to transfer multiline messages.
SRQ	**Service Request:** Signal used by a device to indicate that it needs servicing due to an error or some other condition. An active controller usually responds to SRQ going TRUE (LOW) by conducting a serial poll reading the status register of the effected device. The SRQ signal is driven by an open collector driver. This results in a wire-ORed configuration of all devices' SRQ line drivers.
ATN	**Attention:** ATN can only be asserted by an active controller. When ATN is TRUE (LOW), only bus commands are passed on the bus's eight data lines. All instruments on the bus must monitor the ATN line at all times and must respond to it going TRUE within 200ns. Instruments respond to ATN going TRUE by setting their NDAC line driver LOW. Every instrument will examine each bus command and if appropriate, will carry out the instruction. When ATN is FALSE, device-dependent data is passed on the data lines and only the addressed talker and addressed listener(s) are involved in any communication.
EOI	**End or Identify:** The active talker asserts EOI (LOW) to indicate that the **last** data byte of a device dependent data file is being sent.
DAV	**Data Valid:** This signal is driven LOW (TRUE) by an active controller or an active talker to indicate the present of a valid, stable command or device dependent data byte on the DIO lines.
NRFD	**Not Ready For Data:** Used by a device to indicate whether it is ready to receive a multiline message on the bus's eight DIO lines. When TRUE (LOW), the device is not ready to accept another byte; when FALSE (HIGH), the device is ready to accept a byte. Devices use an open collector driver to drive the NRFD bus line. This results in an wire-ORed configuration among devices' NRFD line drivers. Therefore, the NRFD line won't actually go HIGH until the slowest device on the bus is ready to receive an octal message.
NDAC	**Not Data Accepted:** Used by a device to indicate whether the multiline message (bus command or device dependent data) on the bus's DIO lines has been accepted. When TRUE (LOW), this signal indicates that the device has not yet accepted the byte. When FALSE (HIGH), NDAC indicates that the all device involved in the transaction have accepted the octal message. Since the NDAC driver of each device is wire-ORed together, the NDAC line doesn't

TABLE 2. Description of the IEEE-488 Bus Signals [continued]

actually go HIGH until the slowest device on the bus has accepted the byte on the bus's DIO lines.

IFC — **Interface Clear:** This signal can only be asserted (LOW) by the system controller. It is used to immediately clear the bus the talker and all listeners are unaddressed. The IEEE-488 standard requires that all instruments be capable of responding to an interface clear in less than 100µs. Instruments respond to an interface clear by setting both there NDAC and NRFD line drivers FALSE (HIGH).

REN — **Remote Enable:** REN can only be asserted (LOW) by the system controller. This signal is used to place a device into remote programming mode if the device is subsequently addressed while REN is TRUE. In remote programming mode, a device receives instructions via its IEEE-488 interface and not via front panel controls (local mode) which are locked out.

Multiline commands and messages are passed on the bus's eight DIO lines. When ATN is TRUE (command mode), only bus commands are passed on the bus's eight data lines. All instruments on the bus, including the talker, must monitor the ATN line at all times, and must respond to it going TRUE within 200ns. Instruments respond to ATN going TRUE by setting their NDAC line driver LOW. Each instrument will examine every bus command and if appropriate, will carry out the instruction. Examples of a multiline bus commands include all devices receiving the GTL (GoTo Local leave remote programming mode) command, an instrument being addressed to listen, or another device being addressed to talk. When ATN is FALSE (talker mode), device-dependent data is passed on the data lines and only the designated talker and designated listener(s) are involved in any communication. Examples of device dependent data are an oscilloscope sending waveform data to a plotter, a logic analyzer sending state table data to a printer, or a computer (acting as a talker and not a controller since ATN=F) sending set-up and measurement instructions (device dependent data) to a parameter analyzer.

The assertion of REN (LOW) along with subsequent device listen address commands from the system controller places the addressed devices into remote programming mode. When a device is in remote programming mode, it receives instructions through its bus interface and its front panel controls are locked out. Devices which aren't addressed remain in local programming mode (instructions are entered by user through front panel controls) and do not participate in communication over the bus. REN can only be asserted by the system controller. REN is asserted by the buffer when in command mode. [Note: This is the reason why the buffer shouldn't be used in a system with another controller.] The IEEE Standard 488-1978 requires that all instruments respond to REN going FALSE within 100µs. Instruments respond to REN going FALSE by entering local mode.

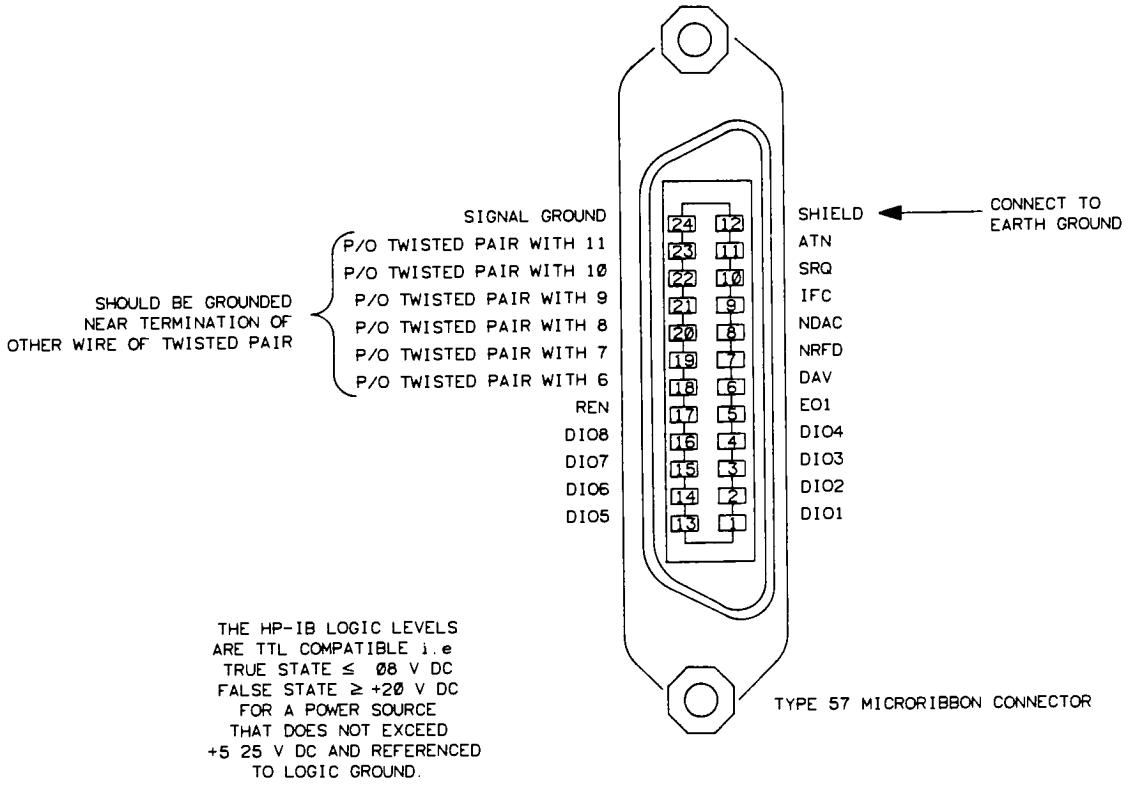
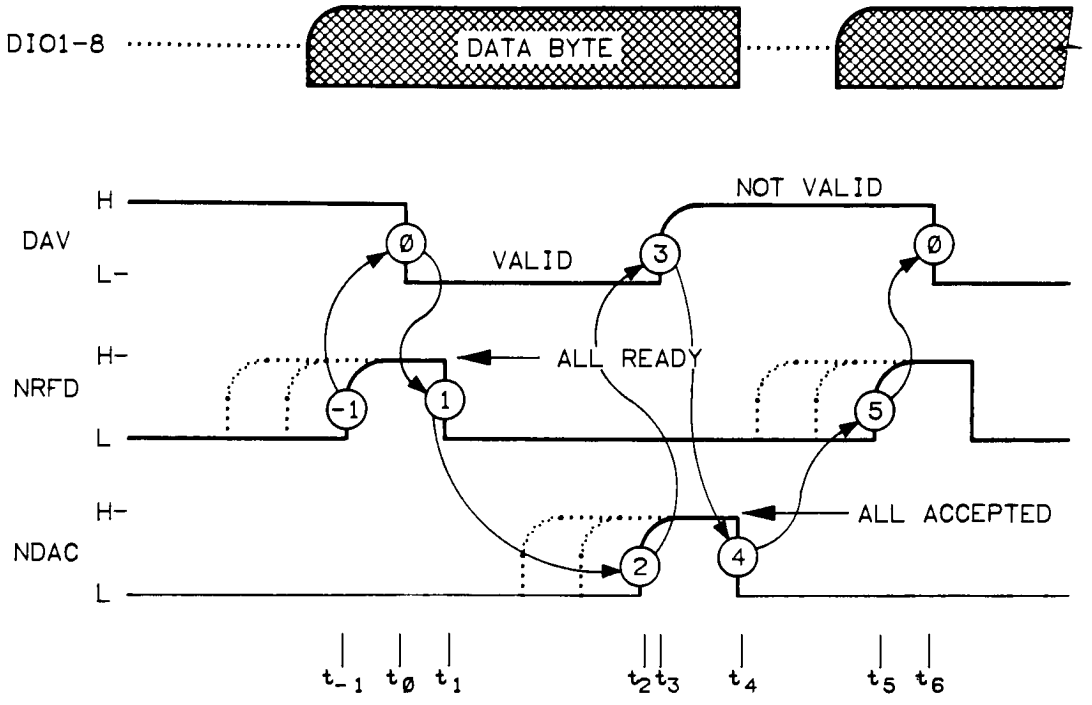


Figure 1 IEEE-488 Connector and PIN OUT. Copyright Hewlett-Packard Company 1980. Reproduced with Permission of Hewlett-Packard Company.

Figure 3 displays the bus driver/receiver requirements as specified in the IEEE Standard 488-1978. On the left hand side of the figure is the DC load line and a typical transceiver implementation is shown on the right hand side. The important point to notice are resistors R_{L1} and R_{L2}. If VCC, R_{L1}, and R_{L2} have the values shown in the figure, then voltage division results in 3.3 volts appearing at the output of the transceiver when not driven by either the bus or the transceiver. This fact will be referred to later in the reading.



Preliminary: Source checks for listeners and places data byte on data lines.

t_{-1}: All acceptors become ready for byte. NRFD goes high with slowest one.

t_0: Source validates data (DAV low)

t_1: First acceptor sets NRFD low to indicate it is no longer ready for a new byte.

t_2: NDAC goes high with slowest acceptor to indicate all have accepted the data.

t_3: Source sets DAV high to indicate this data byte is no longer valid.

t_4: First acceptor sets NDAC low in preparation for next cycle.

t_5: Back to t-1 again.

Figure 2 IEEE-488 Handshake Protocol. Copyright Hewlett-Packard Company 1980. Reproduced with Permission of Hewlett-Packard Company.

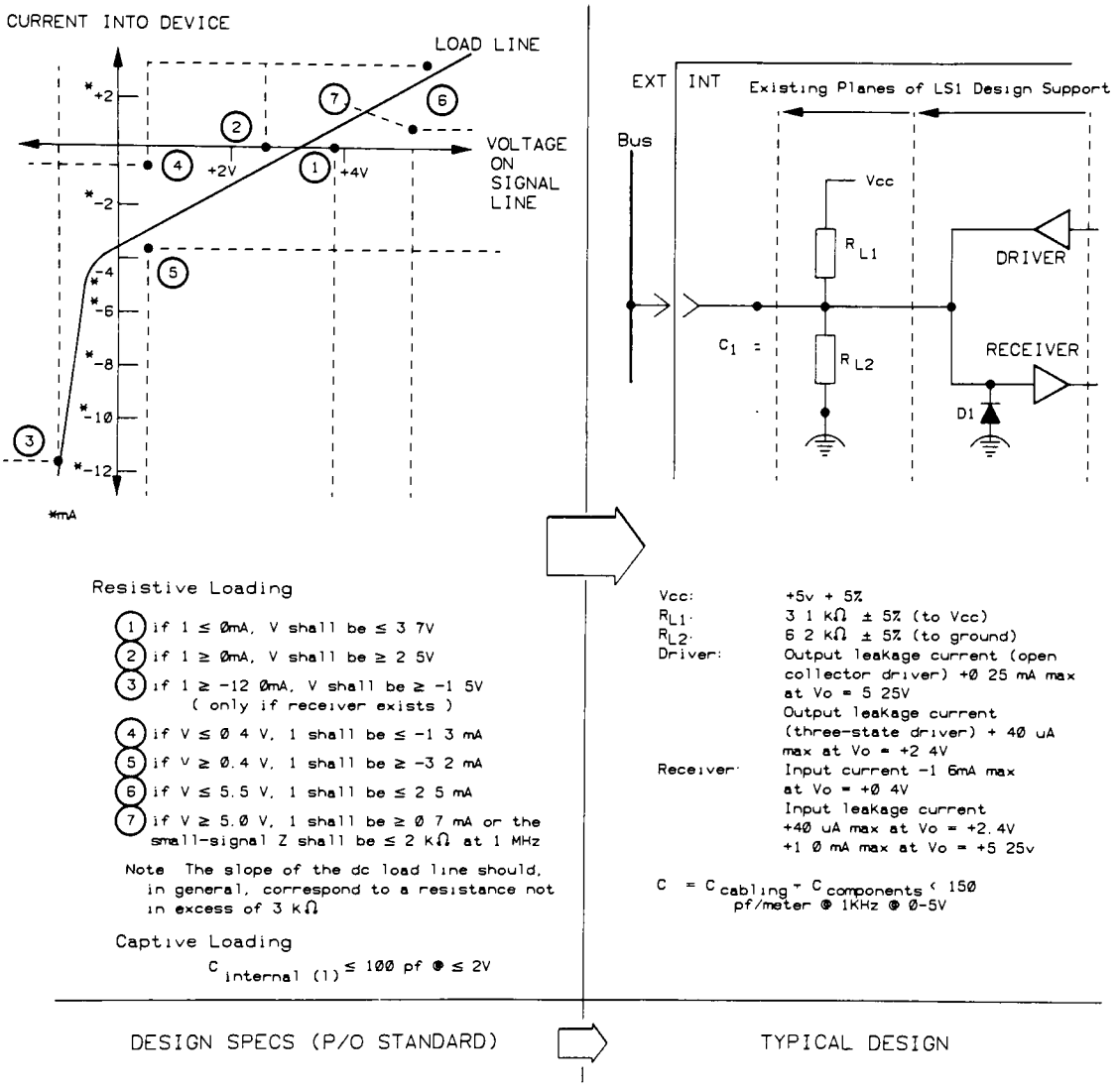


Figure 3 IEEE-488 Driver/Receiver Requirements. Copyright Hewlett-Packard Company 1980. Reproduced with Permission of Hewlett-Packard Company.

5. Buffer Application

The RIT IEEE-488 Buffer is intended for use with a IEEE-488 interfaceable talker capable of transmitting a data file to an attached IEEE-488 interfaceable listener in a controllerless system. Any talkers should be configured to send an EOI (End Or Identify) with the **last** data byte sent. The talker must also release the bus after file transfer is complete deactivate all bus drivers. Any listeners should be taken out of "listen-only" or "listen-always" mode and given an address between 0-30. This address is then entered **into the buffer** using the on board dip switches. If more than one device is to receive the transmitted data, then all receiving devices should be given the same address. If file transfer to each listeners is done individually, then the intended receiver's address should be entered into the buffer via the dip switches before file transfer. Note: length of GPIB cables used must be in accordance with the IEEE Standard 488-1978.

The buffer isn't equipped with on board power and must be connected to a regulated 5V/0.7A power supply. At power-up, a power-on reset circuit initializes the buffer and configures it as an active listener. The buffer waits in this initialized state until the talker commences communication. When data transmission is started, the buffer accepts the entire file from the talker, then addresses the intended receiver, transfers the stored file to the addressed listener, unaddresses the listener, and finally, resumes the active listener configuration. Note: the amount of data transferred can not exceed the amount of memory available for storage on the buffer. The present design contains 32K of memory. Memory can be increased by cascading additional counters, comparators, and latches.

The buffer's reset switch can be used at any time to terminate a file transfer and/or re-initialize the buffer by placing it in the CLEAR position. The switch must be placed back into the OPERATE position for the buffer to function. If an addressed listener is accepting data when the buffer is reset, the addressed listener may **not** exit listener mode. The addressed listener must be taken out of listener mode by some means usually by turning its power OFF and back ON, before the next file is transferred. Also, the talker should **not** start a second file transfer before the first transfer has completed.

A typical example of a system (Fig. 4) which could utilize the RIT IEEE-488 Buffer involves an HP4145B Parameter Analyzer connected to an HP COLOR-PRO graphics plotter addressed to "listen-only" mode. Implementation would require the buffer being connected to the measurement device and data recording device with GPIB cables. Many measurement devices have a user selectable option of sending an EOI (end or identify) signal with the last data byte transferred. This option **must** be selected when using the buffer. On the HP4145B, this option is selected by setting a dip switch on the rear panel. The plotter should be taken out of "listen-only" mode and given some address between 0 and 30. This address is then entered into the buffer using the on board dip switches. Dip switches 1-5 are used to input the plotter's address while switches 6-8 are not used. A measurement can then be taken and if a hardcopy is desired, the data is dumped into the buffer by hitting the Plot key followed by the Execute key on the HP4145B. Then, while the buffer drives the data recording device, the measurement device may be set up for the next measurement, the measurement taken, and the result examined and manipulated - comments inserted, marker positioned, slopes added, etc. Once the buffer is done driving the data-recording device with the previous measurement's

data, a clean sheet of paper can be loaded into the plotter, and the latest measurement data can be dumped to the buffer and the cycle repeated.

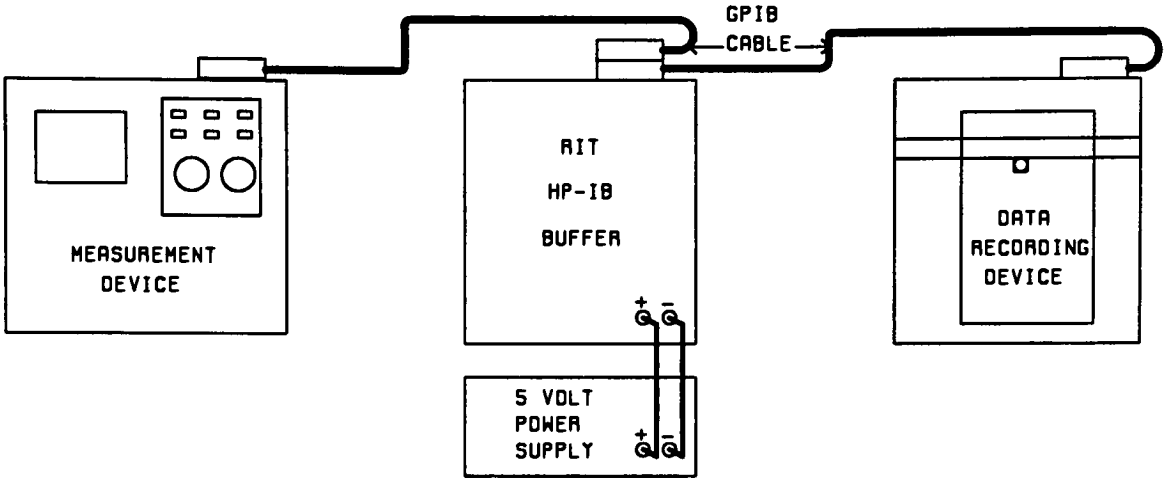


Figure 4 Typical Controllerless System using RIT IEEE-488 Buffer

6. Signal Naming Convention

As one reads this thesis, he or she will notice several groups of similarly named signals which have been listed in Table 3. Signals within a group (EOI, EOI.L, EOII.L, and EOIO.L) represent the same signal at various places within the system. Signals on the bus side of the line drivers (FIG. 5) use the IEEE-488 naming convention - for example, EOI, DIO1. Signals on the buffer side of the SN75160 GPIB transceiver make up the buffer bus called DATA (8:1). This bus connects the buffer side of the SN75160 GPIB transceiver with the eight ASIC DO pins and the eight data I/O pins on the SRAM. Bus management and handshake signals on the buffer side of the SN75161 GPIB transceivers have a .L (active LOW) extension added for instance, EOI.L. The .L extension signals are used as ASIC pin/pad names and in several cases, as inputs to synchronizer cells internal to the ASIC. Inside the ASIC (Fig. 6), signal names with an 'I' suffix (NDACI.L) indicate that the signal is used during the *data input phase* of buffer operation. Signal names with the 'O' suffix (EOIO.L, DO1) indicate that the signal is used during the *data output phase* of operation.

TABLE 3 Similarly Named Signals Used Throughout This Document

NDAC, NDAC.L, NDACI.L, NDACO.L
NRFD, NRFD.L, NRFDI.L, NRFDO.L
DAV, DAV.L, DAVI.L, DAVO.L
EOI, EOI.L, EOII.L, EOIO.L
ATN, ATN.L, ATNO.L
REN, REN.L
SRQ, SRQ.L
IFC, IFC.L
DIO8-DIO1, DATA8-DATA1, DO8-DO1
PHASE_1, PHI1, PHI_1A, PHI_1B, ϕ1
PHASE_2, PHI2, PHI_2A, PHI_2B, ϕ2
TE.H, TE.H1, TE.H2, TE
DC.H, DC.HB, DC
C.H, LE.H
PQ.L, PQ.H

When the IEEE-488 bus signals (NDAC, NRFD, DAV, EOI, REN, ATN, IFC, SRQ, and DIO8-DIO1) appear in the text, they refer to the signals actually passing between devices on the GPIB cable. For instance, the bus signal NDAC represents the wire in the cable which carries the signal Not Data ACcepted, pin number eight on the GPIB connector, and the trace/wire connecting pin number eight to pin number four on the SN75161 GPIB transceiver. Hence, an IEEE-488 bus signal represent everything on the bus side of the GPIB transceivers.

Continuing the example from above, NDAC.L represents the same signal as NDAC except on the buffer side of the GPIB transceiver at pin 17. The bus management and handshaking signals are routed directly to pins on the IEEE-488 Buffer Controller (ASIC). NDAC.L is routed to the NDAC.L pin (pin number 9) on the ASIC. [NDAC.L is also connected to the

CLK input (pin number 2) of four cascaded SN74ALS561 counters.] The signal at the NDAC.L pads continues through the pad's INTO_CHIP port to the input of a synchronizer cell within the ASIC. The output of the synchronizer cell is labeled NDACO.L. Signals at the output of a synchronizer cell trail transitions at the cell's input by a $\phi 1$-$\phi 2$ synchronization delay. The OFF_CHIP port on the NDAC.L pad is driven by Output State Machine (OSM) signal NDACI.L.

At times, signals names within a group are used interchangeable. For example, the statement "The buffer drives NDAC HIGH in state DAC." indicates that NDACI.L, NDAC.L, and NDAC are all driven HIGH. NDACO.L is forced to ground during the *data input phase* but is isolated from NDACI.L, NDAC.L, and NDAC by an internal synchronizer cell. During the *data output phase*, signals NDACO.L, NDAC.L, and NDAC are used by the buffer while NDACI.L is prevented from corrupting the signal at the NDAC.L pad since its pad driver is disabled.

The ASIC contains an internal two phase, nonoverlapping clock generator with outputs PHASE_1 and PHASE_2. As these clock signals are distributed and superbuffered throughout the ASIC, they take on the various names listed in the Table 3. The terms $\phi 1$ and $\phi 2$ are used to refer to any of the signal names within there respective group.

TE.H is an output of the ASIC's State Machine Controller (SMC). This signal drives two superbuffers which generate the signals TE.H1 and TE.H2. It was necessary to duplicate this signal due to the large load that TE.H must drive. One of the loads that TE.H2 drives is the ASIC output pin TE. The signal TE is the same as TE.H minus the delays through the superbuffer and the ASIC pad driver. DC.H is also an output of the SMC. DC.HB represents the superbuffered version of DC.H while DC is the name of an ASIC output pin driven by DC.HB.

C.H is an output of the ASIC's Input State Machine (ISM). This signals drives ASIC output pin LE.H. C.H and LE.H are essentially equivalent except for a small delay through the LE.H pad driver.

PQ.L is the NANDed output of the two external comparators and drives ASIC input pin PQ.L. PQ.L is synchronized and inverted within the ASIC to produce the signal PQ.H where the .H extension indicates an active high signal.

The buffer design uses mixed logic both positive and negative logic. When a signal is said to be asserted, it means that the signal is TRUE. Therefore, when NDAC.L is asserted, it is TRUE ($\approx$ 0 volts) or LOW. When PQ.H is asserted, it is TRUE ($\approx$ 5 volts) or HIGH. When a signal is said to be deactivated, it means that the signal is FALSE. When NDAC.L is deactivated, it is FALSE ($\approx$ 5 volts) or HIGH. When PQ.H is deactivated, it is FALSE ($\approx$ 0 volts) or LOW. Also, TRUE is equal to a logic ONE and FALSE is equal to a logic ZERO. When a signal transitions, the phases "goes" or "going" or "go to" are represented by the symbol =>. For instance, NDAC.L=>T means "NDAC.L going TRUE" or "NDAC goes TRUE" or "NDAC to go TRUE" depending on the context of the sentence. NDAC.L=>F means "NDAC.L going FALSE" or "NDAC.L goes FALSE" or "NDAC to go FALSE" depending on the context of the sentence.

7. System Architecture

Figure 5 displays the RIT IEEE-488 Buffer schematic. The buffer consist of two IEEE-488 transceivers, a 32Kx8 SRAM, an ASIC, four counters, two octal latches, two octal comparators, a dip switch, power-up reset circuitry, a GPIB connector, and various other resistors, capacitors, and gates - see parts list (Appendix A) for further details.

The ASIC has Schmitt triggers connected to its PON and CLK_PULSE pins. RC networks at the two Schmitt trigger inputs determine the length of time that ASIC pins PON and CLK_PULSE are driven HIGH when the buffer is first powered on.[5] Once the inputs to the Schmitt triggers have charged to V_{T+}($\approx$ 1.5 $volts_{min}$), their output swings LOW. The RC network at the input to the Schmitt trigger driving CLK_PULSE is set to reach V_{T+} at $\approx$ 30ms. CLK_PULSE going LOW starts the ASIC's internal Clock Generator. The RC network at the input to the Schmitt trigger driving PON is set to reach V_{T+} at $\approx$ 100ms. CLK_PULSE is allowed to go LOW first so that the ASIC's clocked logic can initialize itself in a known state the reset condition.

The buffer is also placed into its reset condition anytime the debounced manual reset switch is placed in the CLEAR position. The reset switch circuitry drives ASIC pin SWITCH. The reset switch must be placed back in the OPERATE position for the buffer to function.

The address of the intended receiver is entered into the buffer via the dip switch. Once the buffer has accepted a file from the talker, it uses the address entered at the dip switch to address the device which is to receive the stored information.

The two GPIB transceivers, the SN75160 and SN75161, are used by the buffer to drive signals onto the bus and to receive signals off the bus through a GPIB connector. The SN75161 GPIB transceiver handles uniline bus commands and handshaking signals. ASIC outputs DC (direction control) and TE (talk enable) control the direction in which bus management and handshake signals pass through the transceiver. During the *data input phase* (DC=T and TE=F), the buffer is listening to the bus and only drives the handshake signals NDAC and NRFD. During the *data output phase* (DC=F and TE=T), the buffer acts as a controller and a talker and drives the bus management signals IFC, REN, ATN, and EOI along with the handshake signal DAV. ASIC output TE is also tied to the SN75160 GPIB transceiver and determines in which direction signals on the bus's DIO8-DIO1 lines pass through the transceiver.

ASIC pin TE is also connected to the $R/\overline{W}$ pin on the memory chip and is used to enable writing to memory while the buffer is in listener mode and reading from memory while the buffer is in talker mode. The SRAM (HM62256LP-15) is a 32Kx8, 150ns device which has its $\overline{OE}$ (Output Enable) pin tied to ground and its $\overline{CE}$ (Chip Enable) pin tied to the ASIC output CS.L. The SRAM's 15 address inputs are tied to the output of the four cascaded counters via the buffer's ADDRESS (15:0) bus and its eight data I/O pins are connected to the buffer side of the SN75160 GPIB transceiver via the buffer's DATA (8:1) bus.

The buffer contains four 4-bit SN74ALS561 counters cascaded together which count each byte of data received or transmitted. This is accomplished by tying the NDAC.L line directly to the clock pin on the four counters. Every time the buffer accepts a data byte during the *data input phase*, it drives NDAC.L FALSE thus incrementing the memory address count. Every time the addressed listener accepts a data byte during the *data output phase*, it drives NDAC HIGH which causes the memory address count to increment. The output of the four counters, the memory address count, is used to address the memory location of where each data byte is stored in SRAM.

While the power-on reset circuit is active, ASIC output ACLR.L (asynchronous clear) is TRUE. ACLR.L forces the outputs of the four binary counters to zero (0000_h) and prevents any counting due to start-up transients on the counters' clock pins. Once the power-on reset circuit has timed out (RESET=F), ACLR.L=>F and counting is enabled. After accepting a data file from the talker and addressing the intended receiver to listener, the buffer again asserts ACLR.L. This is done to reposition the memory address count to the location of the first data byte (0000_h) in preparation for file transfer to the addressed listener. The buffer also asserts ACLR.L immediately before making the transition from bus controller back to active listener in preparation for receiving the next file transfer from the talker.

At power-up, the buffer is placed in listener mode it drives the NDAC (Not Data ACcepted) bus line TRUE (LOW) and the NRFD (Not Ready For Data) bus line FALSE (HIGH). This condition indicates the presence of an active listener on the bus. After the power-on reset circuit has timed out, the buffer remains in its initial state and waits for the talker to send the first multiline message over the eight DIO lines. The talker initiates communication by driving the DAV (Data AVailable) bus line TRUE (LOW). The buffer accepts the data file from the talker a byte at a time and stores each byte in the memory cell addressed by the memory address counters. The talker indicates transfer of the *last* data byte by asserting EOI (LOW). When EOII.L goes TRUE, ASIC output pin LE.H is driven HIGH. LE.H drives the latch enable pin on the two SN74ALS573 octal latches. The two octal latches are used to store the address of the final data byte the *final count value.* When LE.H is true, inputs to the latches become enabled and signals at the latches' inputs are passed through to the latches' outputs. The outputs of the memory address counters are run to the P inputs on the two SN74ALS518 octal identity comparators. The outputs of the two octal latches are run to the Q input on the comparators. The P=Q outputs of the two comparators are connected to a two input NAND gate which drives ASIC pin PQ.L. When the value contained in the octal latches equals the output of the memory address counters, the P=Q output on the two comparators go HIGH thus forcing PQ.L=>T. In response to PQ.L=>T, the ASIC deactivates LE.H thus latching the *final count value.*

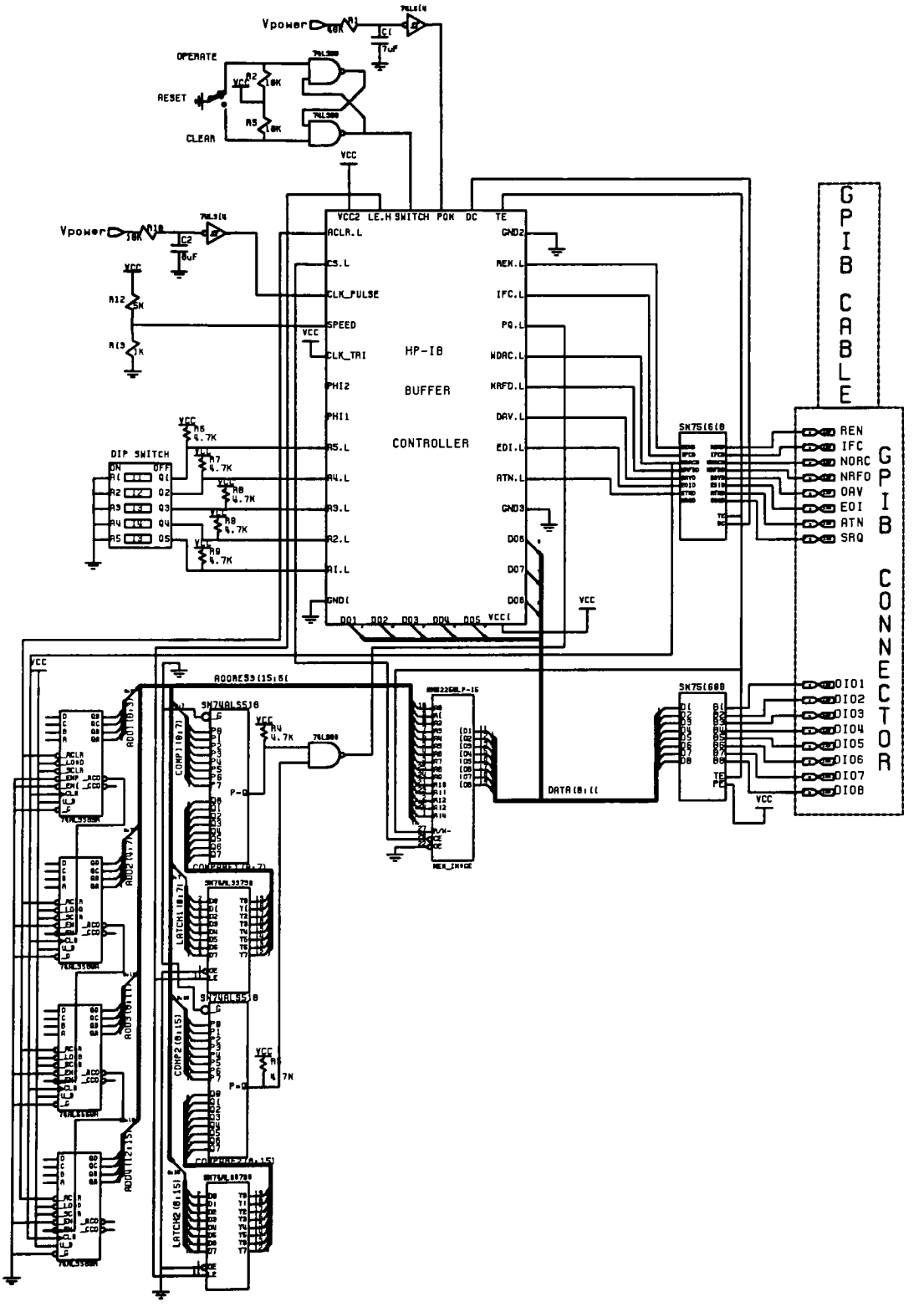


Figure 5 The RIT IEEE-488 Buffer

After the buffer receives the last data byte from the talker, it makes the transition from active listener to active controller. As the bus controller, the buffer addresses the intended receiver to listen and resets the counter to zero. Next, the buffer assumes the role of active talker and sets the DAV line LOW indicating that valid data from memory address (0000_h) is available. Once the addressed listener indicates acceptance of the byte (NDAC going HIGH), the memory address count is incremented and the buffer checks the new memory address count with the value stored in the octal latches using two SN74ALS518 octal identity comparators. If the two counts aren't equal, the next byte of data is placed on the bus and transferred. This sequence continues until the NANDed P=Q outputs of the two identity comparators go LOW indicating that the present memory address count equals the *final count value* contained in the two octal latches. At that point, the buffer sets the EOI.L line TRUE and sends out the last data byte. EOI going LOW informs the addressed listener that it is receiving the last data byte of the current file transfer. The buffer then resumes the role of active controller, unaddresses the intended receiver, and finally, makes the transition from bus controller back to active listener. The buffer is then again ready to accept another file transfer from the talker.

The bus signal SRQ is used by instruments to interrupt the system controller when an error or other servicing is needed. The RIT IEEE-488 Buffer does not support the Service Request Interface Function. The SRQ pin on the buffer side of the SN75161 GPIB transceiver is left unconnected. During the *data input phase*, this signal is an output of the buffer. Since the SRQ line is driven by an open collector driver, it appears to be driven HIGH to other instruments on the bus unless pulled LOW by an instrument other than the buffer. During the *data output phase*, the SRQ signal is an input to the buffer. Since the SRQ pin is left unconnected on the buffer side of the transceiver, any request for service is ignored by the buffer. This would normally be the case in a controllerless system anyway since only a controller can respond to a service request.

8. The RIT IEEE-488 Buffer Controller (ASIC)

The ASIC (Fig. 6) consists of six major modules the Clock Generator, the State Machine Controller, the Input State Machine, the Output State Machine, the Delay Counter, and the Command Multiplexer (CMD MUX). The ASIC also contains circuitry for generating an internal reset signal (RESET) and five synchronizer cells for synchronizing asynchronous signals. The ASIC uses four type of pads - an input pad, an output pad, a bi-directional pad, and a bi-directional pad with superbuffered input. Each of the modules, the synchronizer, and the bi-directional pads are discussed in separate sections below. Table 4 provides a quick reference description of the ASIC's I/O pins. Under the column labeled TYPE, an input pin is designated with an I, an output pin with an O, a bi-directional pin with a I/O, and a bi-directional pin with superbuffered input with a IS/O.

ASIC pins PON and SWITCH are NORed within the ASIC to produce the asynchronous signal RESET. This signal is tied directly to pull-down transistors in the SMC, ISM, CMD MUX, and to the NOR gate sourcing ASIC output pin ACLR.L. RESET=>T causes the SMC's outputs DC.H=>T and TE.H=>F, the ISM to initialize to state PON, the CMD MUX output MUX.L=>T, and ACLR.L=>T. Further initialization is carried out by TE.H and DC.H.

The following two paragraphs summarize the condition of various modules during the *data input phase* and the *data output phase*. The reader may find it useful to refer back to these paragraphs while reading the remainder of this section

During the *data input phase,* the ISM is engaged while the OSM and Delay Counter are held inactive. The drivers on ASIC pads IFC.L, REN.L, ATN.L, DAV.L, and EOI.L are tristated (pads act as inputs) while the NDAC.L and NRFD.L pad drivers are enabled (pads act as outputs). Bus signals REN, IFC, ATN, EOI, and DAV are passed from the bus through the SN75161 GPIB transceiver to RIT IEEE-488 Buffer Controller. The buffer drives bus signals NDAC and NRFD through the SN75161 transceiver out onto the bus. Data bytes on the bus's DIO lines are passed from the bus through the SN75160 GPIB transceiver to the inputs of the SRAM which is in write mode. During the *data input phase*, the ASIC's DO8-DO1 pad drivers are tristated and the inverted (bus uses negative logic) intended receiver's listen address command is present at the output of the Command Multiplexer.

The Delay Counter is started and the ISM is disabled at the **start** of the *data output phase.* Approximately 100μs latter, the OSM is activated. The direction in which signals pass through the two GPIB transceivers is opposite of the direction they traveled during the *data input phase*. The buffer now sources bus signals IFC, REN, ATN, DAV, and EOI and receives as input bus signals NDAC and NRFD. The drivers of ASIC pads IFC.L, REN.L, ATN.L, DAV.L, and EOI.L are enabled (pads act as outputs) while the NDAC.L and NRFD.L pad drivers are tristated (pads act as inputs). The SRAM is in read mode and when enabled (CS.L=T) drives stored data through the SN75160 transceiver out onto the DIO lines.

During the *data input phase* DAV, EOI, and DIO8-DIO1 are driven by the talker. IFC, REN, and ATN are most likely not driven since no controller is present on the bus during the *data input phase*. The bus side of the IEEE-488 bus transceivers are designed to float at approximately 3.3 volts when these signals aren't driven (Fig. 3). Thus, IFC, REN, and ATN

appear to be driven HIGH. However, the ASIC doesn't use these signals during the *data input phase* so the INTO_CHIP port on pads IFC.L, REN.L, and ATN.L are not connected internally within the ASIC.

TABLE 4 Quick Reference Summary of ASIC Inputs and Output Signals

Signal	Type	Name and Function
ATN.L	I/O	**AtTentioN (pin 5):** ATN is used by an active controller to get the attention of all devices when issuing interface commands. Since the buffer is only designed for use in a dedicated talker/listener (non-controller) configuration, the INTO_CHIP port of this pad is left unconnected. The buffer asserts this signal in command mode (OSM states BUS_CMD, CMD_OUT, and CMD_ACC). The OFF_CHIP port on this pad is driven by OSM output ATNO.L.
EOI.L	I/O	**End Or Identify (pin 6):** EOI.L is an input sourced by the talker while the buffer is in listener mode. The INTO_CHIP port on this pad drives the synchronizer which sources EOII.L. The talker assets EOI.L to indicate that the *last* data byte from the present file transfer is being sent. When the buffer is in command and talker modes, EOI.L is an ASIC output. The OFF_CHIP port on this pad is driven by OSM output EOIO.L. The OSM drives EOI.L TRUE while sending the *last* data byte to the addressed listener.
DAV.L	I/O	**DAta Valid (pin 7):** Used as an input to the ASIC when in listener mode and an output of the ASIC when in talker or command mode. DAV.L TRUE indicates the presence a valid, stable command or data byte on the DIO8-DIO1 lines. The pad's INTO_CHIP port drives the synchronizer which sources DAVI.L and its OFF_CHIP port is driven by OSM output DAVO.L.
NRFD.L	I/O	**Not Ready For Data (pin 8):** This signal is driven by the ASIC while in listener mode. It is used to inform the talker when the buffer is ready to receive another data byte. The OFF_CHIP port on this pad is driven by ISM output NRFDI.L. In command and talker modes, NRFD.L is an input to the ASIC and drives the synchronizer sourcing NRFDO.L via the pad's INTO_CHIP port.
NDAC.L	I/O	**Not Data ACcepted (pin 9):** This signal is driven by the ASIC while in listener mode. It is used to inform the talker of the buffer acceptance of the data byte present on the DIO lines. The OFF_CHIP port on this pad is driven by ISM output NDACI.L. In command and talker modes, NDAC.L is an input to the ASIC and drives the synchronizer sourcing NDACO.L via the pad's INTO_CHIP port.

TABLE 4. Quick Reference Summary of ASIC Input and Output Signals [continued]

IFC.L	I/O	**InterFace Clear (pin 11):** This signal is an input while the buffer is in listener mode. IFC is used by the system controller to cease all bus activity and unaddress all devices. Since the buffer is only designed for use in a dedicated talker/listener (non-controller) configuration, the INTO_CHIP port of this pad is left unconnected. The OFF_CHIP port on this pad is tied to VCC. Hence, IFC.L is never asserted during the *data output phase.*
REN.L	I/O	**Remote ENable (pin 12):** This signal is an input while the buffer is in listener mode. REN is used by the system controller to prepare devices to enter remote programming mode. Since the buffer is only designed for use in dedicated talker/listener (non-controller) configuration, the INTO_CHIP port of this pad is left unconnected. The OFF_CHIP port on this pad is tied to GND. Hence, REN.L *is asserted* while the buffer is in command mode and talker mode.
DC	O	**Direction Control (pin 15):** This pin is connected to the DC pin on the SN75161 GPIB transceiver. It is used in conjunction with ASIC pin TE to control the direction in which signals pass through the SN75161 GPIB transceiver. TE is the complement of DC. When DC is HIGH, bus signals REN, IFC, ATN, DAV, and EOI are passed from the bus to the buffer while bus signals NDAC and NRFD are passed from buffer to bus. These signals reverse direction when DC is LOW.
TE	O	**Talk Enable (pin 14):** This pin is tied to the TE pin on the two GPIB transceivers and to the R/W pin on the SRAM. When TE is LOW, signals on the bus's DIO lines pass through the SN75160 GPIB transceiver from bus to buffer and vice-versa when TE is HIGH. The SRAM is in write mode while TE is LOW and in read mode while TE is HIGH. See DC signal description for additional information.
DO8-DO1	O	**Data Output 8-1 (pins 3-1, 37-33 respectively):** These pins are used to drive the inverted intended receiver's listen address command or UNL command onto the buffer's DATA(8:1) bus. The pad drivers on these pins are only enabled in OSM states BUS_CMD and CMD_OUT.
CS.L	O	**Chip Select (pin 21):** ASIC output connected to CE.L (chip enable) input on the SRAM. CS.L is used to enable the memory device. Memory is enabled during listener mode and talker mode. It is disabled in command mode when the ASIC is driving the DO8-DO1 lines. This prevent bus contention while the ASIC is issuing bus commands.

TABLE 4. Quick Reference Summary of ASIC Input and Output Signals [continued]

PQ.L	I	**P=Q (pin 10):** Asynchronous ASIC input used to indicate whether the present memory address count is equal to the value contained in the two octal latches. This signal is passed through a synchronizer cell and an inverter within the ASIC to produce the signal PQ.H. PQ.H=>T during the *data input phase* informs the buffer that the address of the last byte is contained in the two octal latches and that the buffer should deactivate ASIC output LE.H to latch this address. PQ.H=>T during the *data output phase* informs the buffer that it is about to transmit the final byte of the present file transfer and that it should assert OSM output EOIO.L.
LE.H	O	**Latch Enable (pin 18):** ASIC output connected to the latch enable pin on the two SN74ALS573 Octal D-TYPE Transparent Latches. This signal is asserted during the *data input phase* when EOI.L is TRUE to allow entry of the last data byte's address from the current file transfer. It is then deactivated when the ASIC input PQ.L=>T thus latching the *final count value*.
ACLR.L	O	**Asynchronous CLeaR (pin 20):** ASIC output connected to the ACLR pin on the four 74ALS561 4-bit binary counters. This signal is asserted when the buffer is first powered-on or anytime it is reset. A LOW voltage on ACLR.L resets the 4-bit output of each counter to 0_h. ACLR.L is also asserted before the buffer begins transmitting the stored file to the addressed listener and before the buffer makes the transition from bus controller back to active listener.
A5.L-A1.L	I	**Address bits 5-1 (pins 27-31 respectively):** These pins are used for entering the five bit address of the device which will ultimately receive the stored data file. The address is entered via the dip switch on board the buffer which has one side connected to the respective A5.L-A1.L pins.
PON	I	**Power ON (pin 16):** Asynchronous input to ASIC's reset circuitry driven by the external power-on circuit.
SWITCH	I	**SWITCH (pin 17):** Asynchronous input to ASIC's reset circuitry driven by the debounced reset switch circuit.

TABLE 4. Quick Reference Summary of ASIC Input and Output Signals [continued]

PHI1	IS/O	**PHI1 (pin 25):** This is the ψ1 clock signal. When ASIC pin CLK_TRI is TRUE (HIGH), the PHI1 pad driver is enabled and the voltage at the PHI1 pad follows the Clock Generator output PHASE_1. When CLK_TRI is FALSE, the PHI1 pad driver is disabled thus allowing the PHI1 signal to be supplied externally.
PHI2	IS/O	**PHI_2 (pin 26):** This is the ψ2 clock signal. When ASIC pin CLK_TRI is TRUE (HIGH), the PHI2 pad driver is enabled and the voltage at the PHI2 pad follows the Clock Generator output PHASE_2. When CLK_TRI is FALSE, the PHI2 pad driver is disabled thus allowing the PHI2 signal to be supplied externally.
SPEED	I	**SPEED (pin 23):** This signal is a analog input which is routed to the Clock Generator's SPEED port. The voltage at SPEED is used to control the frequency of the Clock Generator's two phase, nonoverlapping clock. The voltage at the SPEED pin is set using two external resistors. The author recommends a maximum internal clock frequency of 2MHz. Running the ASIC faster than this reduces the Delay Counter's delay period. Note: The SPEED pad is just a metal pad. A standard input pad would produce a digital signal at the SPEED port.
CLK_PULSE	I	**CLocK_PULSE (pin 22):** At power-up, an external one-time pulse generator drives this pin HIGH for approximately 30ms. This pad is routed to the CLK_PULSE.H port on the Clock Generator. The Clock Generator needs a one-time start pulse to kick it off when power is applied. Once CLK_PULSE.H=>F, the Clock Generator begins to run.
CLK_TRI	I	**CLocK_TRIstate (pin 24):** When CLK_TRI is TRUE (HIGH), the PHI1 & PHI2 pad drivers are enabled and the voltage at these pads follows there respective Clock Generator outputs. When CLK_TRI is FALSE, the PHI1 & PHI2 pad drivers are disabled thus allowing external clocking of the ASIC.
VCC	I	**5 Volt (pins 19 & 38):** The ASIC contains two power pins.
GND	I	**Ground (pins 4, 13, & 32):** The ASIC contains three ground pins.

8.1 The Bi-Directional Pads

After the buffer receives all the data from the talker, it must make the transformation from active listener to active controller. During the transition, some bus lines which were used as inputs into the buffer during the *data in phase* become outputs from the buffer. For instance, the bus signals, DAV and EOI, are used as inputs by the buffer during the *data input phase* but become outputs sourced by the buffer after the transition. Also, the bus signals NDAC and NRFD, are sourced by the buffer when in listener mode but become sourced by the addressed listener after the transition. Therefore, a method is needed within the ASIC to separate the input and output signal paths. This is accomplished by the bi-directional pads.

When the author first started this project, the only pads available (on the Apollo workstations in the Computer Engineering VLSI Lab at RIT) were a standard input pad and a standard output pad. The author essentially copied the schematic and layout of two bi-directional pads PAD_TRISTATE* (Fig. 7) and PAD_TRISTATE_BUFFERED** (Fig. 8). The only change involved replacing butted contacts with buried contacts. These two pads are now available on the Apollo workstations at RIT in working directories /user/pub/pads/pad_tristate and /user/pub/pads/pad_tristate_buff. The only difference between the two pads is that the INTO_CHIP signal on PAD_TRISTATE_BUFFERED is run through a superbuffer. The PAD and INTO_CHIP ports on PAD_TRISTATE are logical connected. In the original schematic, the author added a small resistor between the INTO_CHIP and PAD ports on PAD_TRISTATE. However, this configuration will not allow LVS to run. Therefore, the resistor and PAD port were removed.

When TRI_CTL (tristate control) is LOW, both pad drivers (M15 and M16) are OFF and the pad acts as an input pad. When TRI_CTL is HIGH, the drivers are enabled and the pad acts as an output pad - the PAD port follows the signal at the OFF_CHIP port. The voltage at INTO_CHIP follows the voltage at OFF_CHIP when TRI_CTL is HIGH. When TRI_CTL is LOW, the signal at the OFF_CHIP port is isolated from the PAD and INTO_CHIP ports. Look at the DAV signal for example (Figs. 5 & 6). During the *data input phase* (TE.H=F), this bus signal is driven through the SN75161 GPIB transceiver to ASIC pin DAV.L. The TRI_CTL port on the DAV pad is driven by TE.H2. Since TE.H2 is LOW during the *data input phase,* the pad acts as an input and the signal DAV.L is passed through the pad's INTO_CHIP port to the input of a synchronizer cell. The synchronizer drives the signal DAVI.L used by the ISM. During the *data input phase*, the OSM idles in state TRANS where it drives DAVO.L FALSE. DAVO.L is connected to the OFF_CHIP port on the DAV.L pad. Since TE.H2 is FALSE, the drivers on pad DAV.L are disabled thus providing isolation between the DAVO.L and DAV.L. After the transition, the buffer takes on the role of active controller and sources bus signals DAV. TE.H2 is now HIGH so the DAV.L pad acts as an output and DAV.L follows OSM output DAVO.L. DAVI.L will also follow DAVO.L after a ψ1-2ψ synchronization delay but this has no effect since the ISM is held in state PON during the *data output phase*.

---

*Robert W. Hon and Carlo H. Sequin, A Guide to LSI Implementation, 2nd ed.; Xerox, Palo Alto Research Center, 1980, pp 138-139.

**Robert W. Hon and Carlo H. Sequin, A Guide to LSI Implementation, 2nd ed.; Xerox, Palo Alto Research Center, 1980, pp 138-139.

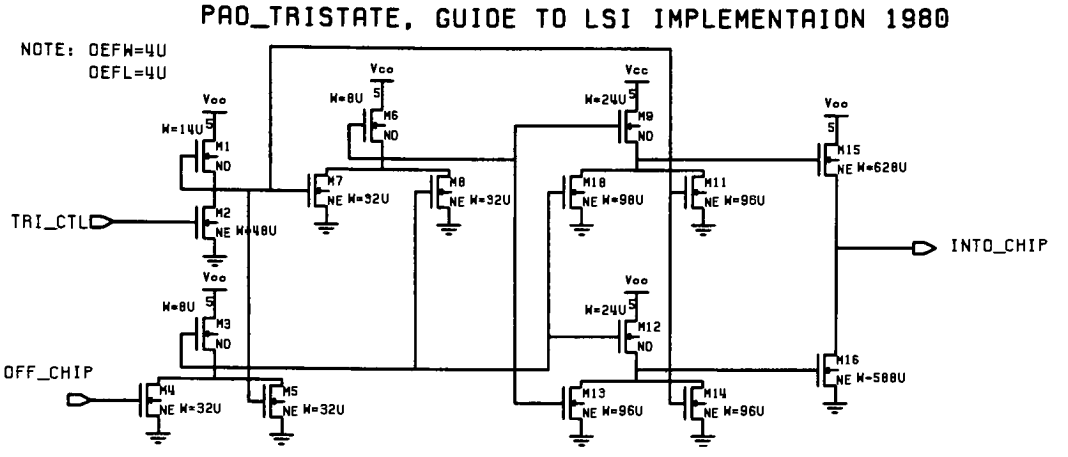


Figure 7 PAD_TRISTATE Schematic

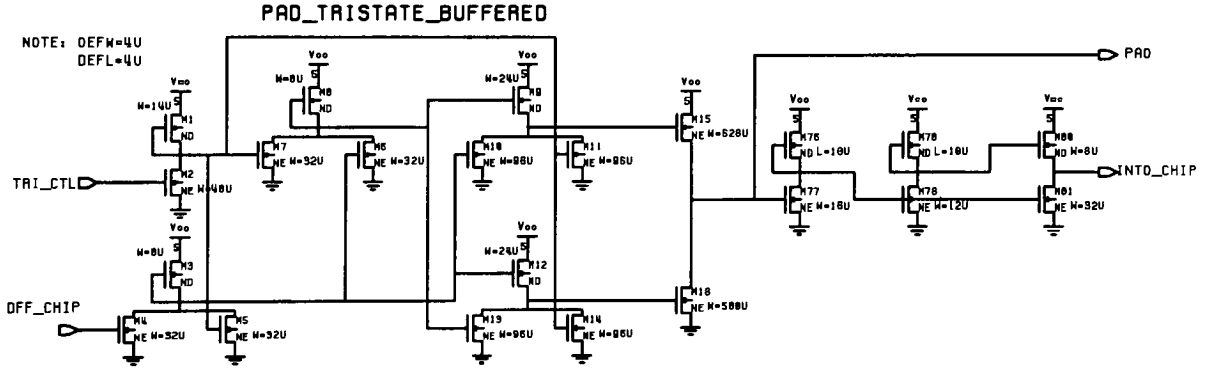


Figure 8 PAD_TRISTATE_BUFFERED Schematic

8.2 The Clock Generator

The function of the Clock Generator is to internally generate and supply the clocking signals needed by the ASIC's clocked logic. A basic building block of the Clock Generator is the variable speed, stoppable, clock cell - VSCLK* (Fig 9). Table 5 provides a summary description of the cell's three inputs and lone output. Four VSCLK cells are daisy chained together (OUT connected to IN of the next) to make a two phase, nonoverlapping clock generator.

TABLE 5 Quick Reference Summary of VSCLK I/O Ports

Signal	Type	Name and Function
IN	I	A pulsed input into VSCLK. When IN goes LOW, OUT goes HIGH.
OUT	O	Pulsed output of VSCLK. OUT goes HIGH after IN goes LOW. The width of OUT is determined by SPEED. The width of OUT is extended if STRETCH is HIGH.
SPEED	I	The voltage at SPEED sets the width of OUT. The width of OUT decreases as the voltage at SPEED increases.
STRETCH	I	STRETCH HIGH causes OUT to go HIGH. This signal is used to extend the width of OUT.

Figure 10 displays the simulation of a single VSCLK cell. Note that waveforms M7/s and M11/d are in units of current while the remaining waveforms have units of volts. At t=0ns, VSCLK inputs STRETCH and IN are both 0.0 volts while SPEED is set to 1.0 volt. Working through the resultant nodal voltages at t=0ns, one observes that M2/d (drain) is a logical ONE and that M5/d (output of second inverter) is a logical ZERO. Consequently, pass transistor M3 is ON and pass transistor M7 and pull-down transistor M11 are both OFF. M6 is barely ON due to the small analog input (1.0 volt) at SPEED. Therefore, with M7 OFF, M9/g (gate) is pulled LOW through M6 and M3. This causes M19/g (input to the fourth inverter) to be at a logical ONE, M13/g at a logical ZERO, M16 and M18 turned ON, and OUT at $\approx$ 0 volts.

As IN goes HIGH at t=50ns, M3 turns OFF and M7 & M11 turn ON. This causes M9/g to charge toward (Vcc $[V_{th} + \alpha V_b]$) $\approx$ 4.0 volts in the simulation through M7 (notice increased current, more negative, on M7/s waveform) and M19/g to discharge toward ground. Since pull-down transistor M11 is ON, the output of the fourth inverter (M13/g) is prevented from going HIGH. Note increased current on M11/d waveform. Thus, OUT remains $\approx$ 0 volts.

*John Newkirk and Robert Mathews, **The VLSI Designer's Library** (Reading, Massachusetts : Addison-Wesley Publishing Company, Inc. 1983), pp. 48-50.

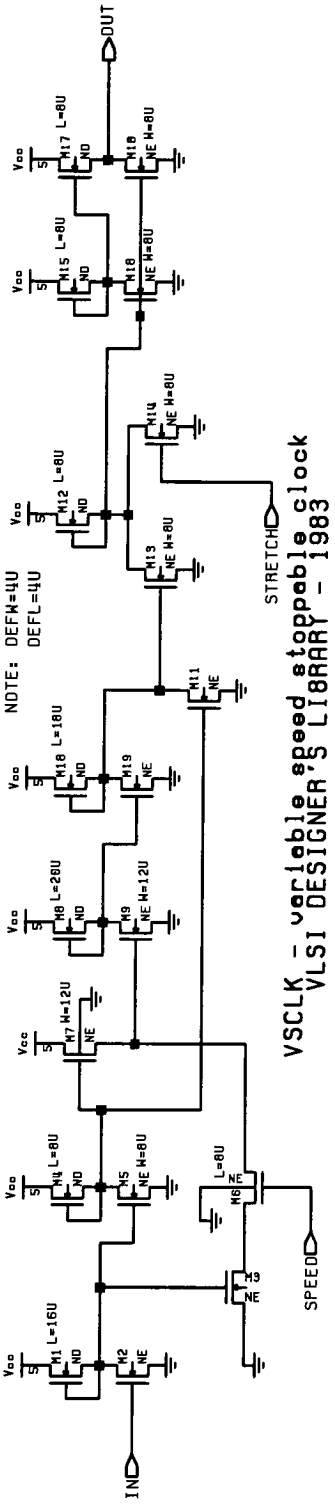


Figure 9 Variable Speed Stoppable Clock (VSCLK) Schematic

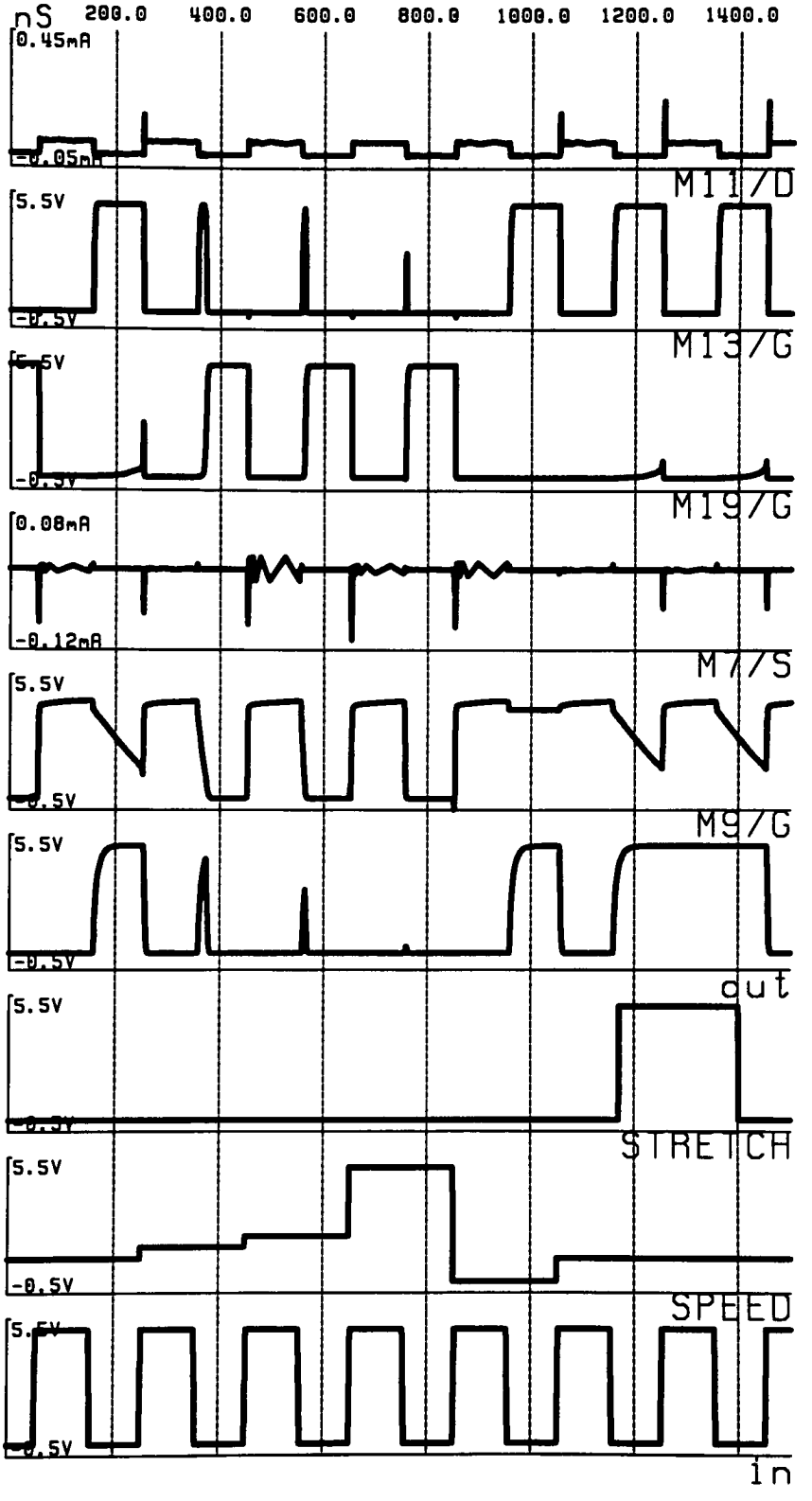


Figure 10 VSCLK Simulation Waveforms.

At t=150ns, IN goes LOW resulting in M3 turning ON and M7 & M11 turning OFF. With M11 OFF, the output of the fourth inverter is no longer pulled down and M13/g rises to a logical ONE. M13/g going HIGH leads to M16 and M18 turning OFF thus allowing OUT to go HIGH. While this is happening, the charge at M9/g is slowly sunk to ground through the weakly ON M6 and the fully ON M3. The simulation shows M9/g discharging from ≈ 4.3 volts to ≈ 1 volt between t=155ns and t=252ns. As M9/g sinks close to its threshold voltage, M9 begins to turn OFF and the input to the fourth inverter (M19/g) begins to charge spike at t=250ns. If allowed to continue, M13/g would sink to a logical ZERO causing OUT to go back LOW. However, the dominant effect at t=250ns is IN going HIGH thus turning OFF the discharge path through M3 and turning ON the charging path through M7 and the pull-down path through M11. If SPEED were left at 1 volt, OUT would continue to go HIGH approximately every 400ns (2.5MHz) with a pulse width of ≈ 200ns.

The analog input SPEED is increased to 1.5 volts at t=250ns causing M6 to turn ON more fully and thus able to conduct higher current. When IN is brought back LOW at t=350ns, the discharging of M9/g through M6 occurs more rapidly. Consequently, M19/g charges faster which leads to a shorten M13 ON time. The result is OUT having a narrower pulse width and lower peak potential. As SPEED is increased, the pulse width and peak potential continue to degrade until OUT almost disappears (t=750ns) at SPEED = 5 volts. At t=850ns, SPEED is set to 0 volts which turns OFF M6 thus eliminating the M9/g discharge path. When IN goes LOW at t=950ns, the M9/g discharge path through M6 is no longer present and the nodal voltage is retained. This results in a completely restored OUT pulse since M13/g remains HIGH until IN goes back HIGH. At t=1170ns, STRETCH is brought HIGH for approximately 230ns. This results in the pulse width of OUT to be extended independent of IN. Once STRETCH goes back LOW, OUT again becomes dependent on IN.

The Clock Generator is constructed of four VSCLK cells as displayed in Figure 11. This construction results in the formation of a variable speed, stoppable, two phase, nonoverlapping clock generator. The speed of the clock is controlled by the voltage at the analog input SPEED. The stoppable feature (STRETCH held HIGH) is useful when synchronizers (discussed below and in next section) are used.

Figure 12 displays the Clock Generator module and its associated circuitry as incorporated into the ASIC. The design allows for both external sourcing or internal generation of the $\psi 1$ and $\psi 2$ clock signals. It is the author's intention to generate the clocking pulses internally via the Clock Generator cell. However, as a precaution, the clocking circuitry is designed so that the Clock Generator can be turned OFF thus allowing external clocking via the PHI1 and PHI2 pins.

The Clock Generator is activated by a start pulse applied to ASIC input pin CLK_PULSE at power up. The CLK_PULSE pad is routed internally to the CLK_PULSE.H port on the Clock Generator cell. Clock Generator outputs PHASE_1 and PHASE_2 are routed to the OFF_CHIP port on ASIC pads PHI1 and PHI2, respectively. The PHI1 and PHI2 pins use the bi-directional pad with superbuffered input (PAD_TRISTATE_BUFFERED). ASIC input pad CLK_TRI (clock tristate) is routed to the TRI_CTL ports on the PHI1 and PHI2 pads while ASIC pin SPEED is routed to the SPEED port on the Clock Generator. One can adjust the internal clock frequency by varying the voltage at SPEED and monitoring the resultant

frequency at the PHI1 and PHI2 pins. The author recommends a maximum internal clock frequency of 2MHz. Running the ASIC faster than this reduces the Delay Counter's delay period.

When the PHI1 and PHI2 pad drivers are enabled (CLK_TRI tied to 5 volts), the Clock Generator's outputs, PHASE_1 and PHASE_2, are used to supply the ASIC's clock signals. The voltage at the PHI1 and PHI2 pads follows the PHASE_1 and PHASE_2 signals. The signal at the PHI1 and PHI2 pads are driven through there respective on-pad superbuffer to there respective INTO_CHIP port. The INTO_CHIP port on each pad is used to drive two additional superbuffers. The PHI1 pad drives the superbuffers sourcing PHI_1A and PHI_2A while the PHI2 pad drives the superbuffers sourcing PHI_2A and PHI_2B. These superbuffers drive the various clocked logic on board the RIT IEEE-488 Buffer Controller.

The ASIC uses five synchronizer cells to synchronize asynchronous signals with the internally generated clocks. If a signal at a synchronizer's input transitions as $\phi 1$ goes LOW, there is a chance that the synchronizer's internal flip flop may go metastable. In this situation, the synchronizer deactivates (drives LOW) its STRETCH_BAR output causing the Clock Generator to extend the PHASE_2 pulse width until the metastable storage cell takes on a digital value. The STRETCH_BAR output (or STH# output where # is either 1, 2, 3, 4, or 5) of each of the five synchronizer's is connected to a gate on the five input NAND which sources the signal STRETCH. This signal drives the STRETCH port on the Clock Generator. Any time one of the synchronizer's STRETCH_BAR output goes LOW, STRETCH goes HIGH thus extending the PHASE_2 pulse width.

If the Clock Generator doesn't function properly, it can be disabled by tying the SPEED pin to 5 volts. This completely kills the OUT1, PHASE_1, OUT3, and PHASE_2 signals as seen in the VSCLK simulation. To use external clocking, the CLK_TRI pin should be connected to ground. This tristates the drivers on ASIC pads PHI1 and PHI2. External clocks can then be used to drive the PHI1 and PHI2 pads.

The VSCLK and Clock Generator cell can be found in working directories /user/pub/pads/vsclk and /user/pub/pads/clock_gen_stack on Apollo workstations in the Computer Engineering VLSI LAB at RIT.

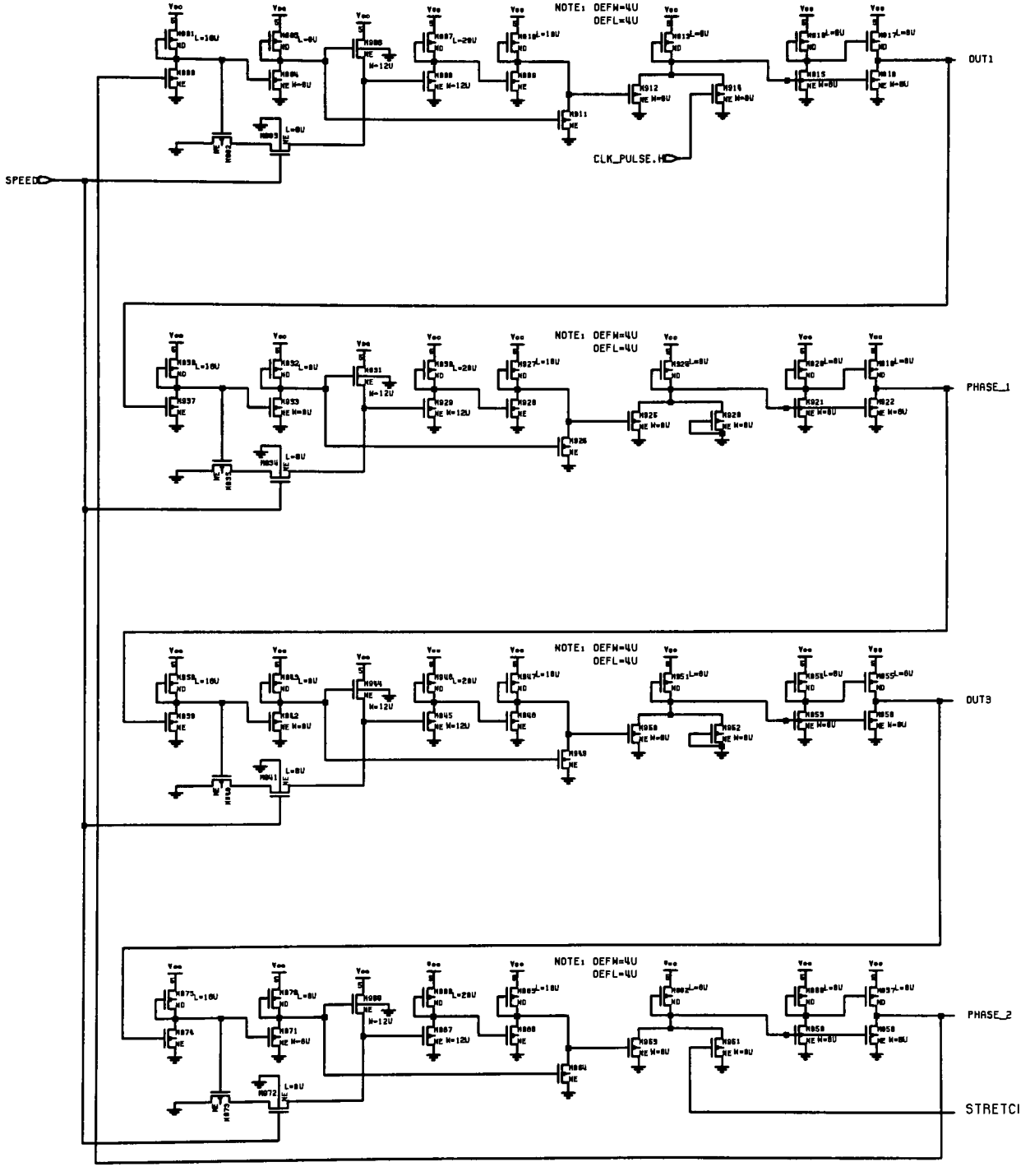


Figure 11 Nonoverlapping, Two Phase, Variable Speed, Stoppable Clock Generator Schematic

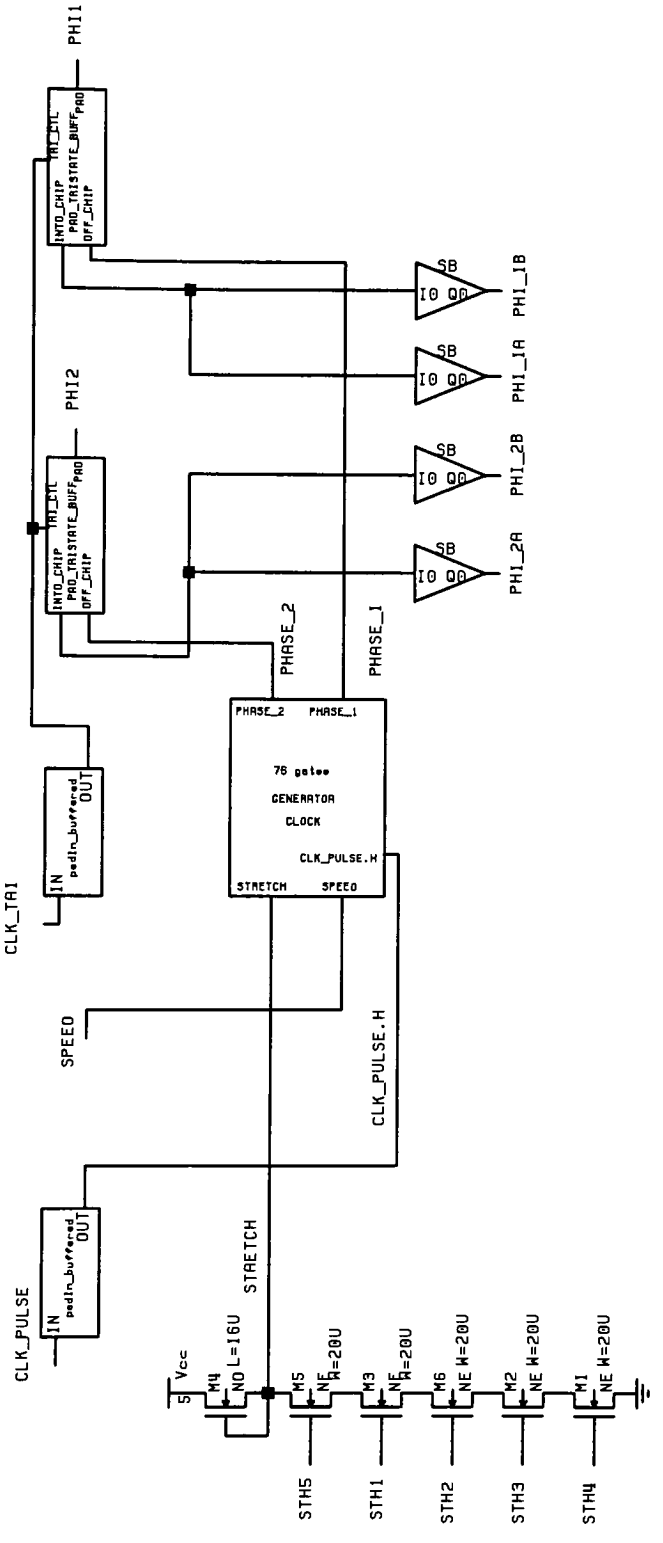


Figure 12 RIT IEEE-488 Buffer Controller Clock Generation Circuitry

8.3 The Synchronizer (SYNC)

The synchronizer* cell (Fig. 13) is used to synchronize an asynchronous input with the ASIC's internally produced two phase, nonoverlapping clock. An asynchronous signal can transition at the synchronizer input (IN) at any time. A strong logic ZERO or logic ONE present at IN, which does **not** transition as $\psi 1$ goes LOW, results in the respective signal level appearing at OUT after a $\psi 1$-$\psi 2$ delay. A signal at IN which transitions or is indeterminate as $\psi 1$ goes LOW, can lead to a non digital voltage at M102/g. This in turn can lead to a non digital output at the drain of M102 (output of first inverter) and once PHASE_2 goes HIGH, an unknown signal level at OUT, and a logic ZERO at STRETCH_BAR. STRETCH_BAR going LOW (STRETCH going HIGH) extends the period of the PHASE_2 clock until the voltage at M102/d settles to a logic ZERO or logic ONE. The probability of the signal at M102/d remaining non digital for a long period of time is extremely small. When the Clock Generator is running at 2MHz, there is a 125ns gap between the time that PHASE_1 goes LOW and when M117/g becomes driven at the rising edge of PHASE_2. This 125ns delay will most likely be enough time for M102/d to settle to a logic ZERO or logic ONE. Table 6 lists the name and function of the synchronizer's inputs and outputs. The SYNC cell is also available in working directory /user/pub/pads/sync.

TABLE 6 Quick Reference Summary of Synchronizer Cell I/O Ports

Signal	Type	Name and Function
IN	I	Asynchronous input to synchronizer cell.
OUT	O	Synchronized, stable-$\psi 1$ equivalent of IN after a $\psi 1$-$\psi 2$ delay.
STRETCH_BAR (output)		Signal used to notify the Clock Generator that a metastable condition exist at the output of the synchronizer's storage cell. The Clock Generator responds by extending the PHASE_2 period. STRETCH_BAR goes LOW once OUT settles to a logic ONE or logic ZERO.
PHI_1	I	The $\psi 1$ clock signal to the synchronizer.
PHI_2	I	The $\psi 2$ clock signal to the synchronizer.

The ASIC has five asynchronous inputs - NDAC.L, NRFD.L, DAV.L, EOI.L, and PQ.L. The first four signals are passed through the SN75161 GPIB transceiver. The transceiver provides a minimum of 400mv hysteresis. Hence, there shouldn't be a problem with slow

*John Newkirk and Robert Mathews, **The VLSI Designer's Library** (Reading, Massachusetts : Addison-Wesley Publishing Company, Inc. 1983), pp. 43-46.

rising or falling signals at the respective ASIC pins. Once the voltage on the bus side of the transceiver reaches the threshold voltage (V_{T-} or V_{T+}), the output on the buffer side will shoot to the ground or power rail. The PQ.L input is sourced by a 74LS00 NAND gate. The rise and fall times on this part can be as much as 15ns depending on loading. Both the fast and slow transitioning signals have the potential to create a metastable condition in the synchronizer's internal storage cell.

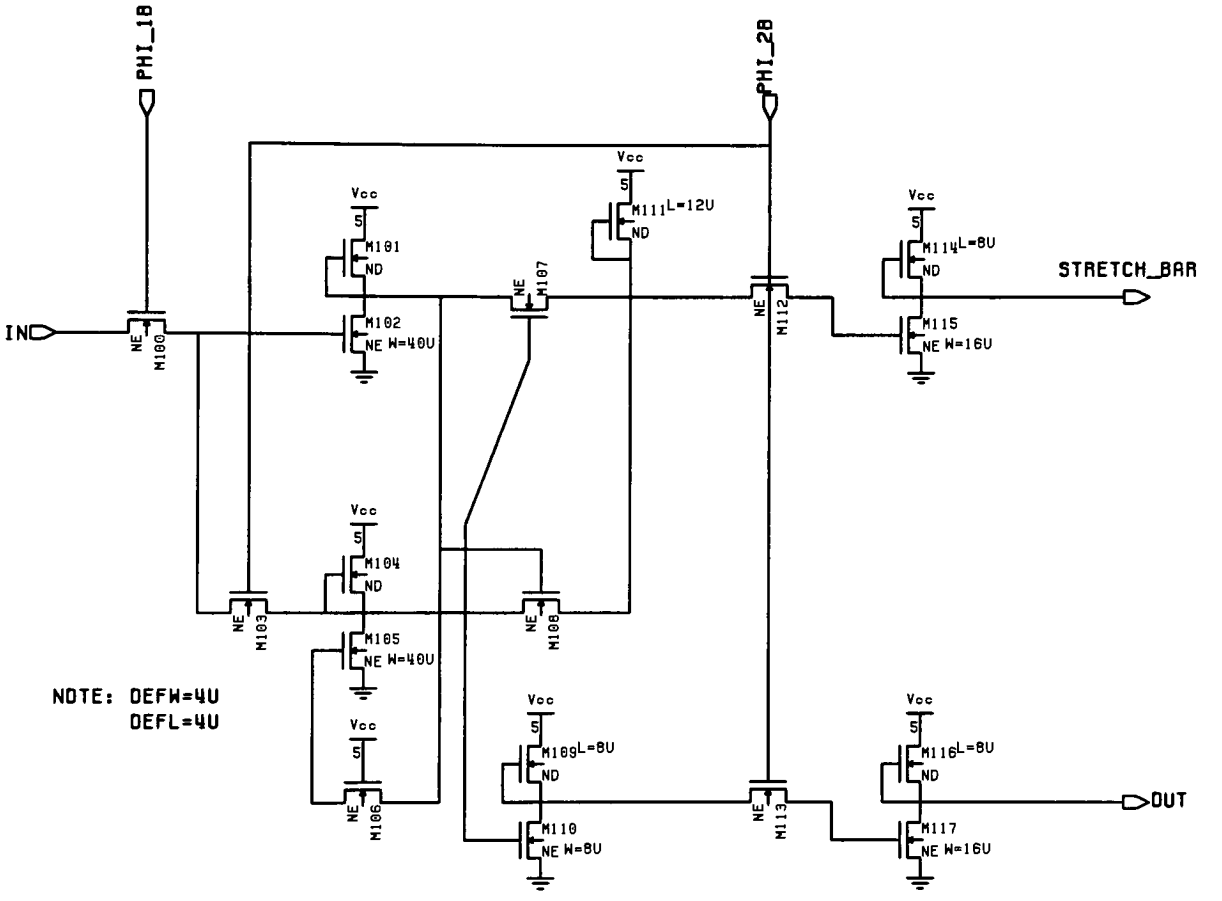


Figure 13 Synchronizer Schematic

In an effort to demonstrate how the synchronizer works in conjunction with the stoppable two-phase clock, the author constructed the schematic show in Figure 14 and ran the simulation shown in Figure 15. In this example, the author attempts to show a metastable condition corrected by the synchronizer. The example is not representative of the ASIC operation. The simulation shows IN going HIGH for only one clock cycle. Asynchronous inputs to the ASIC will remain TRUE or FALSE for at least a few clock cycles unless effected by noise. This example also shows the PHASE_1 and PHASE_2 clocks overloaded 5pF loads. In the actual chip layout, the PHASE_1 and PHASE_2 clocks are not as loaded as shown in the simulation. This loading was used to help bring about a metastable condition.

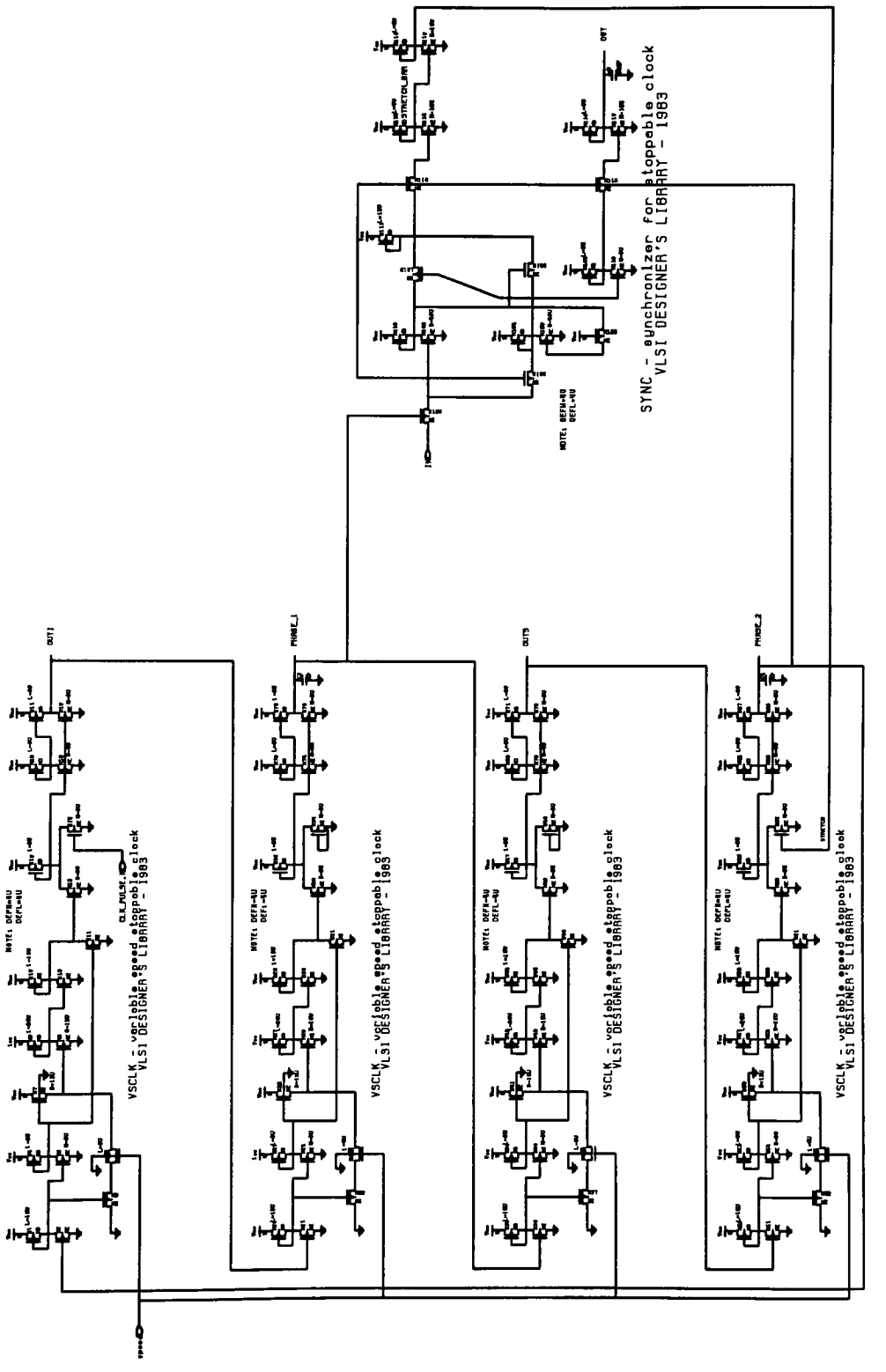


Figure 14 Nonoverlapping, Two Phase, Variable Speed, Stoppable Clock with Synchronizer

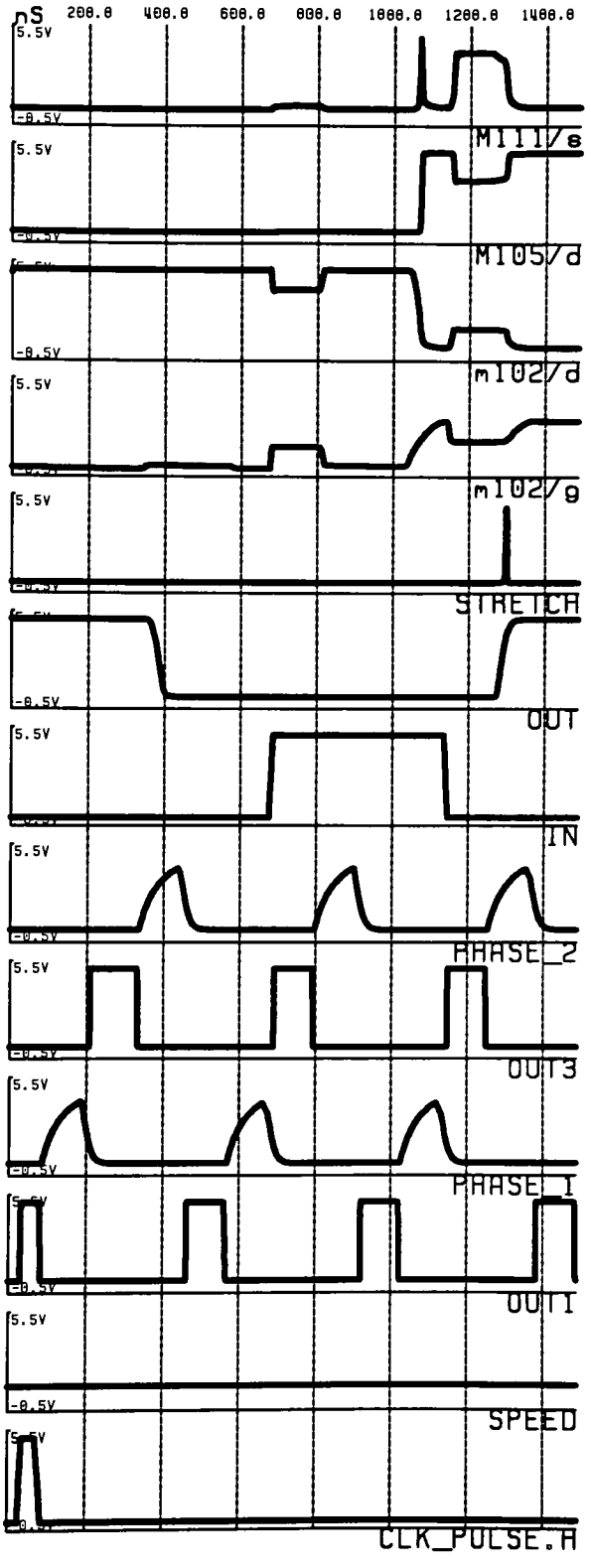


Figure 15 Clock Generator with Synchronizer Simulation Waveforms

The simulation also shows how the Clock Generator is activated by a one time start pulse (CLK_PULSE.H) and the relationship between its four output signals - OUT1, PHASE_1, OUT3, and PHASE_2. The Clock Generator is at a stable operating point when all its outputs (OUT1, PHASE_1, OUT3, and PHASE_2) are set to zero volts. If one assumes these conditions, he or she can work through the corresponding nodal voltages and confirm that this is indeed a stable operating point from which the Clock Generator will not leave. Since this condition could result at power-up, a method is needed to activate the clock. This is accomplished by incorporating a one time pulse generator into the power-up circuitry.* The pulse generator consists of an RC network and a Schmitt trigger. A 8.0μF capacitor has one end connected to the power rail and the other end connected to the Schmitt trigger input. A 10KΩ resistor is placed between ground and the Schmitt trigger input. The output of the Schmitt trigger is connected to ASIC input pin CLK_PULSE which is internally routed to the CLK_PULSE.H port on the Clock Generator cell. When power is first applied to the buffer, it may take several milliseconds for the power rail to reach 5 volts. At the same time, the input to the Schmitt trigger begins to charge according to the RC time constant which is set to be **much** slower (30ms) than the time required for the power rail to reach 5 volts. Therefore, the Schmitt trigger input will be below V_{T+} ($1.5\ volts_{min.}$) for some period of time once the power rail has reached 5 volts. This results in the Schmitt trigger output (CLK_PULSE) going HIGH. After the Schmitt trigger input has charged to V_{T+}, its output goes LOW and remains LOW for the entire time that the buffer is powered on. When the Clock Generator input signal CLK_PULSE.H is brought HIGH, output OUT1 is forced HIGH. The Clock Generator begins operation once CLK_PULSE.H goes LOW.

The analog simulator first calculates a stable condition from which to begin the transient analysis at t=0ns. This stable condition is referred to as the DC operating point (DCOP). If none of the circuit inputs changed, the circuit would remain in this stable condition. To calculate the DCOP, the simulator assumes that all nodal voltages are at zero potential except for those which are being forced or those which have been given an Initial Condition. Since PHASE_1 and PHASE_2 are close to zero volts at t=0ns, all pass transistors clocked by PHASE_1 and PHASE_2 are OFF. Therefore, the nodal voltage at M117/g is assumed initially to be zero volts by the simulator which results in the signal OUT, at M117/d, having a calculated DCOP value of ≈ 5 volts at t=0ns see simulation waveforms. That is, if the input to an inverter is assumed at logic ZERO, the simulator will calculate its output to be at logic ONE. The same situation is TRUE for M102/g which results in M102/d being HIGH at time t=0ns. Since M117/g wasn't assigned an Initial Condition of 5 volts, the synchronizer's output, OUT, doesn't equal its input, IN, at t=0ns. After the first ψ1-ψ2 clock cycle, the synchronizer's output equals its input, OUT = IN.

In the simulation, IN goes HIGH right as the second PHASE_1 pulse goes LOW. This results in M102/g charging to ≈ 1.4 volts and M102/d discharging to ≈ 3.2 volts. The 3.2 volts is fed through the always ON M106 (M106/g tied to VCC) to the gate of M105. The 3.2 volts at M105/g is enough to keep M105/d at ≈ 0.5 volts. When PHASE_2 goes TRUE, the voltage at M105/d is fed back through M103 to M102/g. Thus, M102/g is pulled LOW enough to cause

*Prosser, Franklin P. and Winkel, David E., The Art of Digital Design, 2nd ed. ; New Jersey: Prentice-Hall Inc., 1987, pp. 486-488.

M102/d to charge back HIGH at ≈ 800ns. The first transition of IN demonstrates the input to the synchronizer transitioning as the PHASE_1 goes LOW. The amount of charge which accumulates at M102/g before M100 turns OFF is insufficient to cause the synchronizer's output to follow its input or its internal flip flop to go metastable. Hence, OUT stays LOW. If IN were to remain HIGH during the next PHASE_1 clock pulse, then OUT would equal the input at the following PHASE_2 clock pulse.

Continuing on, since IN is HIGH, when PHASE_1 again goes TRUE, M102/g begins to charge causing M102/d to discharge. As M102/d discharges, M105/d charges. M102/d going LOW turns OFF M108 while M105/d charging turns ON M107. There will be a period of time however where both M107 and M108 are OFF. This results in M111/s (source) charging (spike at ≈ 1080ns on the M111/s waveform) until M107 turns ON. IN transitions LOW as PHASE_1 goes LOW resulting in M102/g charging while M102/d discharges. M102/g discharges to ≈ 1.7 volts before pass transistor M100 switches OFF. With a voltage of ≈ 1.7 volts at its input gate, the output of the first inverter (M102/d) is ≈ 1.4 volts a metastable condition. In this situation, the voltage at M111/s rises because pass transistors M107 and M108 are only weakly turned ON. With only 1.45 volts applied to M108/g and only 3.3 volts applied to M107/g, M111/s is only pulled down to ≈ 3.7 volts. At PHASE_2, the 3.3 volts at M105/d is fed back to M102/g which charges to ≈ 3 volts causing M102/d to discharge to 0.3 volts. However, since M111/s was ≈ 3.7 volts before PHASE_2 went TRUE, the output STRETCH begins to charge (spike at ≈ 1300ns on STRETCH waveform). If M102/d had remained metastable, STRETCH would have remained HIGH causing the extension of the PHASE_2 pulse until M102/d assumed some digital value. Instead, M102/d goes LOW, M105/d charges and M107 turns ON fully thus sinking M111/s. The second transition of IN demonstrates how the synchronizer handles a metastable condition at the output of its internal flip flop. In this situation discussed above, the synchronizer output again failed to follow the transition at its input. If IN remains LOW, OUT will go LOW at the following PHASE_2 clock pulse.

8.4 The State Machine Controller (SMC)

The SMC is the top level controller in the ASIC's multiple controller architecture. It determines which *one* of the two ASIC's state machines is currently active. The SMC has a number of other responsibilities also. It controls tristating of the ASIC's bus management and handshaking pads, the direction in which signals pass through the external GPIB transceivers, and enabling/resetting of the Delay Counter. The SMC also controls whether the external SRAM is in read or write mode and partially controls whether SRAM is in reduced power standby mode or active mode.

The SMC has two states as shown in Figure 16. In state DATA_IN (DC.H=T, TE.H=F) the buffer assumes the role of an active listener. This state is entered upon power-up or anytime the buffer is reset. In state DATA_IN the buffer receives a data file from the talker a byte at a time and stores it in on board SRAM. When the talker has finished sending the data file to the buffer, the SMC moves from state DATA_IN to state DATA_OUT (DC.H=F, TE.H=T). In this state the buffer addresses a device to listen, sends the data file stored in memory to the addressed listener, and then unaddresses the addressed listener. At that point, the SMC re-enters state DATA_IN where the buffer idles while waiting for a talker to send another data file. Throughout this document, state DATA_IN is referred to as the *data input phase* while state DATA_OUT is referred to as the *data output phase*.

The SMC (Fig. 17) consist of two combinational inputs and a "safe semistatic register stage".* The SMC outputs, DC.H and TE.H, determine which *one* of the ASIC's two state machines is active. When DC.H=T and TE.H=F (buffer in listener mode), the ISM is active and the OSM is held inactive; when DC.H=F and TE.H=T (buffer in command or talker mode), the ISM is held inactive while the OSM is activated. Equations for the SMC's outputs are given below. Table 7 provides a description of the SMC's inputs and outputs.

$$TE.H = \overline{DAVI.L} * \overline{EOII.L} * LATCH.H \qquad (1)$$

$$DC.H = CMD\_ACC.H * NDACO.L * \overline{NRFDO.L} * \overline{MUX.L} + RESET \qquad (2)$$

All inputs to the SMC are stable-ψ1 signals except for RESET which is an asynchronous input. The signal RESET is forced HIGH whenever the buffer is powered-on or reset. While RESET is HIGH, pull-down transistor M125 is ON. This forces TE.H LOW and DC.H HIGH. The safe semistatic register stage obeys strict two phase clocking discipline.** When both equations 1 and 2 are FALSE, the register stage refreshes itself through pass transistor M124 the TE.H output is fed back to the input of the register stage. When equation 1 is TRUE, it indicates that the buffer has received the last data byte from the talker and that the talker has

*Amar Murkerjee, **Introduction to nMOS & CMOS VLSI System Design** (Englewood Cliffs, N. J. : Prentice-Hall, 1986), pp. 108-112.

Amar Murkerjee, **Introduction to nMOS & CMOS VLSI System Design (Englewood Cliffs, N. J. : Prentice-Hall, 1986), p. 90.

release the bus. This condition causes a logic ONE to be loaded into the register stage through pass transistor M119. The SMC responds by switching its outputs (DC.H=>F, TE.H=>T) at the following $\psi 2$ clock pulse. When equation 2 is TRUE, it indicates that the buffer has finished sending data to the listener and has successfully unaddressed the listener. This causes a logic ZERO to be loaded into the register stage through pass transistor M118. The SMC again responds by switching its outputs (DC.H=>T, TE.H=>F) at the following $\psi 2$ clock pulse.

Both SMC outputs (TE.H and DC.H) are run through superbuffers since they must each drive large capacitive loads. In fact, the load on TE.H is so large that it is run to the input of two superbuffers to generate the signals TE.H1 and TE.H2. A third superbuffer sources the signal DC.HB.

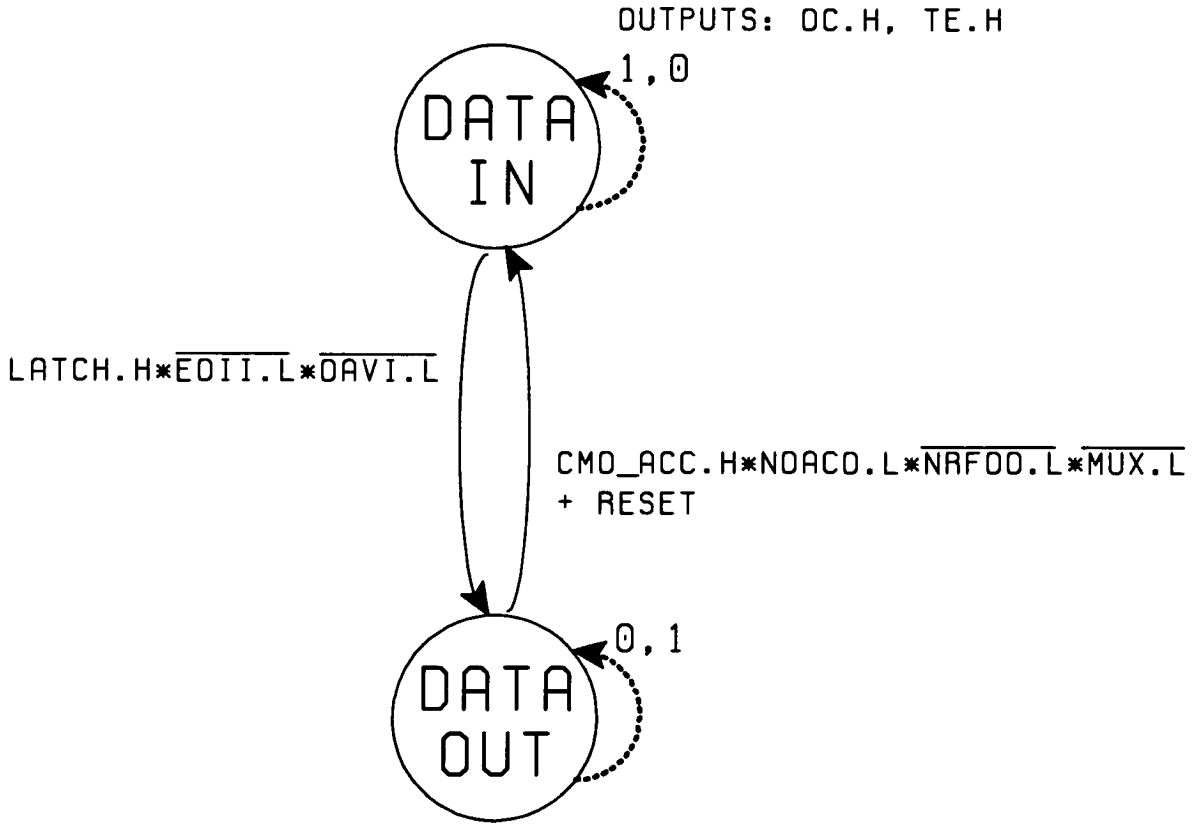


Figure 16 State Machine Controller State Diagram

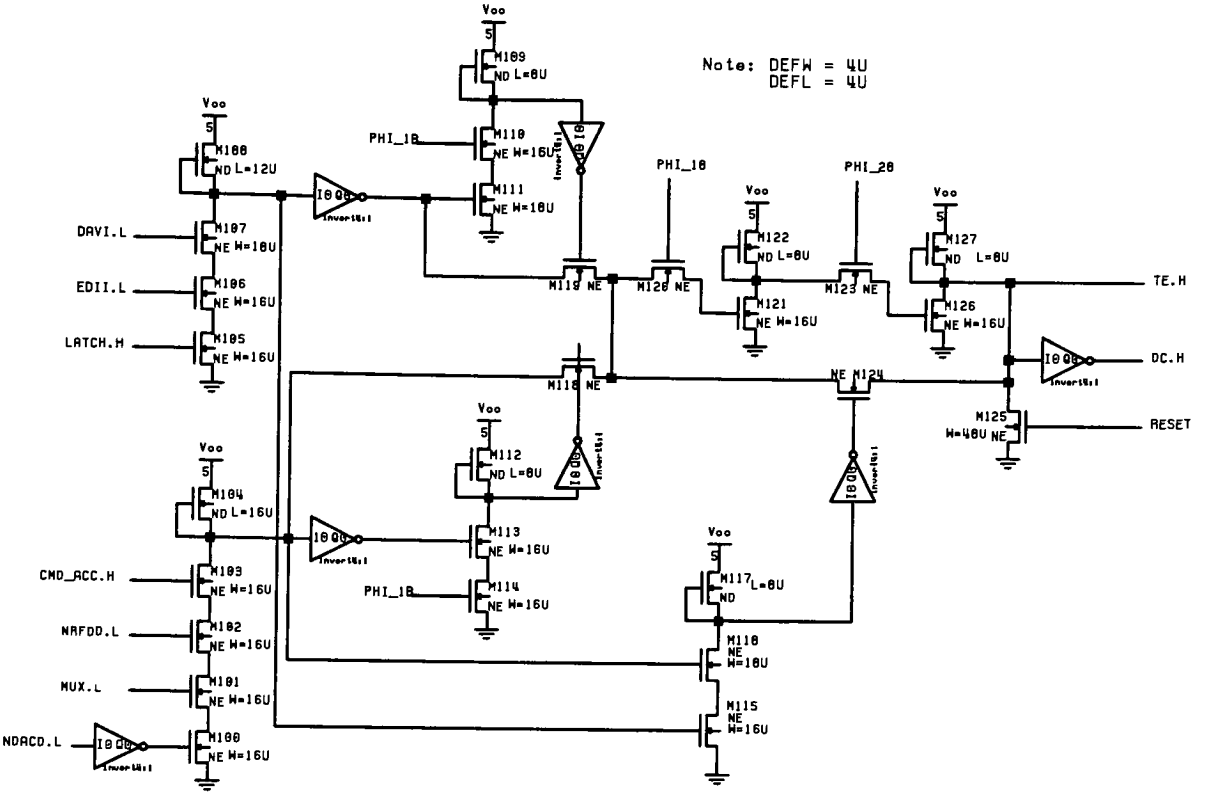


Figure 17 State Machine Controller Schematic

TE.H1 is connected to the ISM and to the Delay Counter. When TE.H is FALSE, the ISM is active provided RESET is FALSE. When TE.H=>T the ISM is deactivated. The Delay Counter output HOLD.H is forced TRUE while TE.H is FALSE. HOLD.H is an input to the OSM. While HOLD.H is TRUE, the OSM is disabled. The Delay Counter output HOLD.H goes FALSE approximately 100μs after the assertion of TE.H1.

TE.H2 is tied to the TRI_CTL (tristate control) input on ASIC pads REN.L, IFC.L, ATN.L, DAV.L, and EOI.L. These pads act as inputs when TE.H is FALSE. When TE.H=>T, ASIC pads REN.L, IFC.L, ATN.L, DAV.L, and EOI.L become driven by the ASIC.

TE.H2 also drives the ASIC output pad TE. ASIC pin TE is tied to the R/$\overline{W}$ pin on the SRAM and to the TE pin (Talk Enable) pin on the two GPIB transceivers. TE determines the direction in which signals pass through the two GPIB transceivers and whether the SRAM is in read or write mode. During the *data input phase* (TE.H2=F), SRAM is in write mode. Once TE.H2=>T, SRAM enters read mode. The SN75160 GPIB transceiver passes the DIO8-DIO1 signals. When TE.H=F, data is passed from bus to buffer. When TE.H=T, data passes from buffer to bus. TE together with ASIC output DC determine the direction in which bus management and handshaking signals pass through the SN75161 GPIB transceiver. During the *data input phase* (DC.H=T, TE.H=F) the buffer drives bus signals NDAC and NRFD and

TABLE 7 Quick Reference Summary of I/O Ports on the State Machine Controller

Signal	Type	Name and Function
RESET	I	**RESET:** Asynchronous input. When TRUE (HIGH) SMC output DC.H is immediately forced HIGH while SMC output TE.H is immediately forced LOW. When RESET is FALSE, the outputs of the SMC are free to response to inputs.
DAVI.L	I	**Data Valid:** When TRUE indicates that the talker has a valid, stable data byte on the eight DIO lines. See LATCH.H description for additional information.
EOII.L	I	**End or Identify:** When TRUE indicates that the talker will be sending the last data byte to the buffer. See LATCH.H description for additional information.
LATCH.H	I	**LATCH:** When TRUE indicates that the ISM is in state LATCH. Inputs LATCH.H, DAVI.L, and EOII.L are used to load the SMC's register stage with a logic ONE when the expression DAVI.L * EOII.L * LATCH.H is TRUE. This causes the SMC's outputs to switch (DC.H=>F, TE.H=>T) at the following $\psi 2$ clock pulse.
NRFDO.L	I	**Not Ready For Data:** When TRUE indicates that the addressed listener isn't ready to receive another multiline message from the buffer. See CMD_ACC.H description for additional information.
NDACO.L	I	**Not Data Accepted:** When FALSE indicates that the addressed listener has accepted the multiline message on the DIO lines. See CMD_ACC.H description for additional information.
MUX.L	I	**Multiplexer:** When TRUE indicates that the inverted (bus uses negative logic) intended receiver's listen address command is present at the output of the Command Multiplexer. When FALSE indicates that the UNL command is present at the output of the Command Multiplexer. See CMD_ACC.H description for additional information.
CMD_ACC.H	I	**Command Accept:** When TRUE indicates that the OSM is in state CMD_ACC. Inputs CMD_ACC.H, MUX.L, NDACO.L, and NRFDO.L are used to load the SMC's register stage with a logic ZERO when the expression CMD_ACC.H * NDACO.L * NRFDO.L * MUX.L is TRUE. This causes the SMC's outputs to switch (DC.H=>T, TE.H=>F) at the following $\psi 2$ clock pulse.

TABLE 7. Quick Reference Summary of I/O Ports on the State Machine Controller [continued]

DC.H	O	**Direction Control:** DC.H is the complement of TE.H. DC.H=T indicates that the buffer is in listener mode. DC.HB, the superbuffered equivalent of DC.H, drives ASIC output pin DC, enables/disables drivers on ASIC pins NDAC.L and NRFD.L and is an input to the CMD MUX and to the logic driving ASIC output CS.L.
TE.H	O	**Talk Enable:** When TRUE indicates that the buffer is in command or talker mode. TE.H=T inactivates the ISM, activates the Delay Counter which in turn activates the OSM, enables drivers on ASIC pads REN.L, IFC.L, ATN.L, DAV.L, and EOI.L, causes data to be passed from buffer to bus through the SN75160 transceiver, and places external SRAM in read mode. TE.H=>F causes the buffer to enter listener mode, resets the Delay Counter which in turns deactivates the OSM, activates the ISM, tristates ASIC pads REN.L, IFC.L, ATN.L, DAV.L, and EOI.L, causes data to be passed from bus to buffer through the SN75160 transceiver, and places SRAM in write mode.

receives as input bus signals REN, IFC, ATN, DAV, and EOI. In the *data output phase* (DC.H=F, TE.H=T), the direction in which these signals travel through the SN75161 GPIB transceiver reverses.

DC.HB is tied to the TRI_CTL port on ASIC pads NDAC.L and NRFD.L. While DC.HB is TRUE, the drivers on these bi-directional pads are enabled (pads act as outputs). Once DC.HB=>F, the drivers on these pads become tristated (pads act as inputs). DC.HB also drives the ASIC output pad DC. ASIC pin DC is tied to the DC pin (Direction Control) on the SN75161 GPIB transceiver. DC together with ASIC output TE determine the direction in which signals pass through the SN75161 GPIB transceiver.

DC.HB is used as an input to the three input NOR gate driving ASIC output pad CS.L The CS.L pin is connected to the chip enable pin on the external SRAM. CS.L is always TRUE while the buffer is in listener mode since DC.HB being TRUE pulls the output of the NOR gate LOW. When the buffer enters the *data output phase*, CS.L follows DC.HB going FALSE thus disabling SRAM. Finally, DC.HB along with ISM outputs PI0.H and PI1.H are used to load the Command Multiplexer with a logic ZERO. See the section 8.9 titled The Command Multiplexer (Cmd MUX) for more information on its operation.

8.5 The Input State Machine (ISM)

The Algorithmic State Machine* (ASM) chart for the ISM is shown in Figure 18. This chart shows the four states of the ISM, the value of each ISM output during a particular state, and the conditions necessary for a change in state. The name of the present state is shown encircled at the top left of each state box. The two numbers at the top right of the box represent the logical value of state variables PI1.H and PI0.H. For example, in state RDY the numbers 01 mean that PI1.H=F and PI0.H=T. Written inside each state box are the values of the present state's unconditional outputs. For instance, the output NDACI.L is always TRUE in state PON. Enclosed in brackets (i.e. <EOII.L>) below each state box are the inputs to the ISM. If a particular input is TRUE, the conditional branch marked "T" will be followed; if the input is FALSE, the "F" branch is taken. Continuing the state PON example, if inputs EOII.L and DAVI.L are FALSE during the $\psi 1$ clock pulse, then at the beginning of the next $\psi 2$ clock pulse, the ISM will remain in state PON. However, if EOII.L=F and DAVI.L=T during the $\psi 1$ clock pulse, then state RDY will be entered when $\psi 2$ goes TRUE. An output enclosed in an oval below the state box and along one of the conditional paths is a conditional output. The output C.H is a conditional output. It is not asserted unless the ISM is in state PON and the input EOII.L is TRUE. From an ASM chart, one can construct a Truth Table (TABLE 8) and generate output equations. Equations 3 through 7 describe the ISM's outputs.

$$C.H = \overline{PI1.H}*\overline{PI0.H}*EOII.L*\overline{PQ.H} \quad (3)$$

$$PI0.H_{next} = \overline{PI1.H}*\overline{PI0.H}*DAVI.L*\overline{PQ.H} + \overline{PI1.H}*\overline{PI0.H}*DAVI.L*\overline{EOII.L} + PI0.H*EOII.L + PI1.H*PI0.H \quad (4)$$

$$PI1.H_{next} = PI1.H*DAVI.L + PI0.H \quad (5)$$

$$NRFDI.L = \overline{PI1.H}*\overline{PI0.H}*DAVI.L*\overline{PQ.H} + \overline{PI1.H}*\overline{PI0.H}*DAVI.L*\overline{EOII.L} + PI1.H*DAVI.L + PI0.H \quad (6)$$

$$NDACI.L = \overline{PI1.H} \quad (7)$$

*Prosser, Franklin P. and Winkel, David E., The Art of Digital Design, 2nd ed. ; New Jersey: Prentice-Hall Inc., 1987, pp. 170-175.

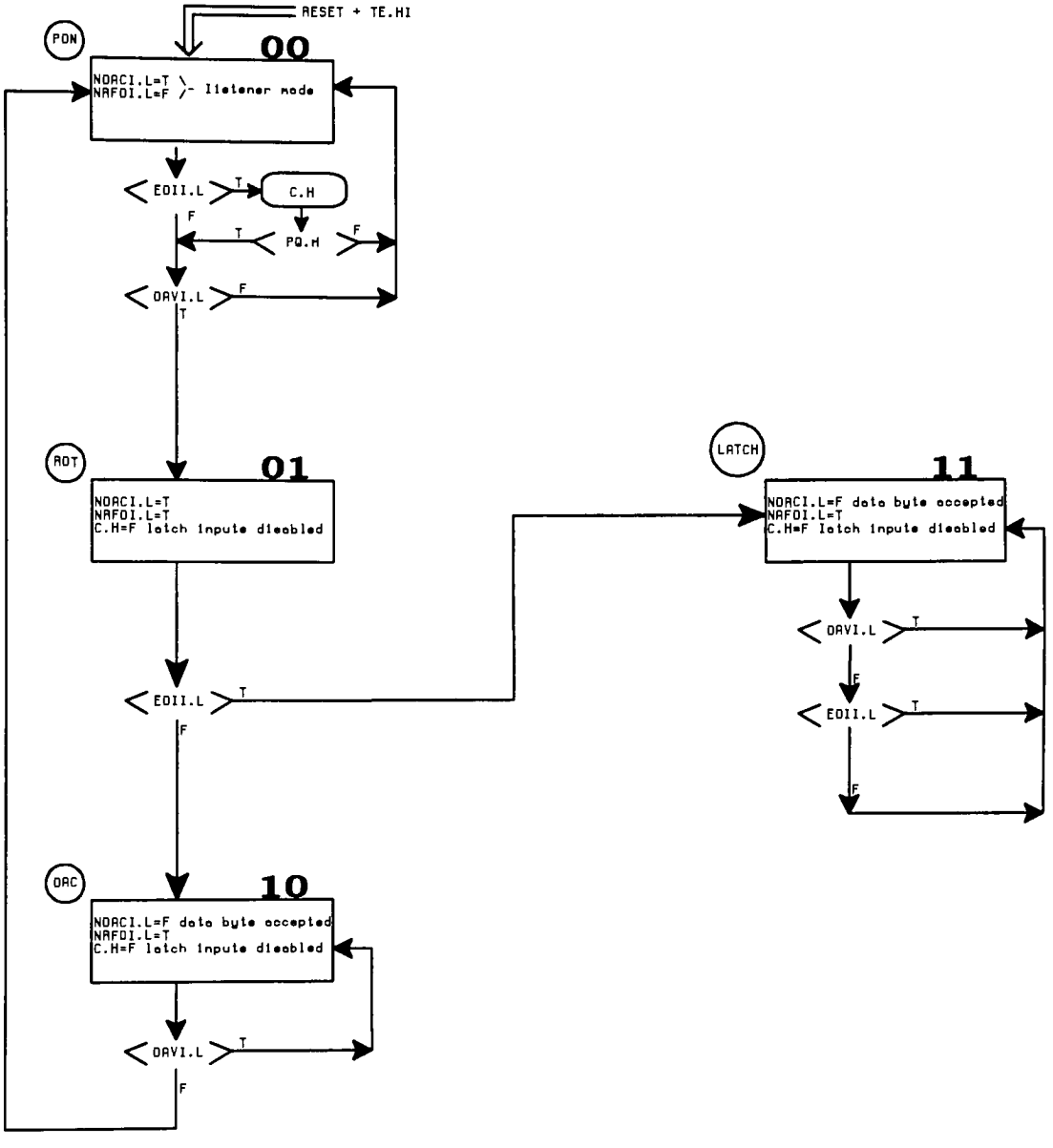


Figure 18 ASM Chart for the Input State Machine

TABLE 8 Input State Machine Truth Table

Present State			Inputs			Outputs			Next State		
NAME	P1	P0	OAVI.L	EOII.L	PQ.H	NOACI.L	NAFOI.L	C.H	N1	N0	NAME
PON	0	0	F	F	X	T	F	F	0	0	PON
			F	T	F	T	F	T	0	0	PON
			F	T	T	T	F	F	0	0	PON
			T	T	F	T	F	T	0	0	PON
			T	T	T	T	T	F	0	1	ROY
			T	F	X	T	T	F	0	1	ROY
ROY	0	1	X	F	X	F	T	F	1	0	OAC
			X	T	X	F	T	F	1	1	LATCH
OAC	1	0	T	X	X	F	T	F	1	0	OAC
			F	X	X	T	F	F	0	0	PON
LATCH	1	1	T	X	X	F	T	F	1	1	LATCH
			X	T	X	F	T	F	1	1	LATCH
			F	F	X	F	T	F	1	1	LATCH=>PON

The ISM (Fig. 19) is a seven product term PLA consisting of five inputs and four outputs. One output has a dual function. The present state variable PI1.H is used as in input to the ISM, to the logic sourcing the signal LATCH.H, and to the CMD MUX. PI1.H happens to be TRUE in all states that NDACI.L is FALSE and is FALSE in all states when NDACI.L is TRUE. Therefore, a separate output for NDACI.L isn't needed since NDACI.L = $\overline{\text{PI1.H}}$ (equation 7). All inputs and outputs of the ISM are stable-ψ1 signals with the exception of RESET which is a asynchronous input. The seven product term expressions are used as net names and appear above there respective net in the AND plane of the PLA. Table 9 provides a quick reference summary of the Input State Machine inputs and outputs.

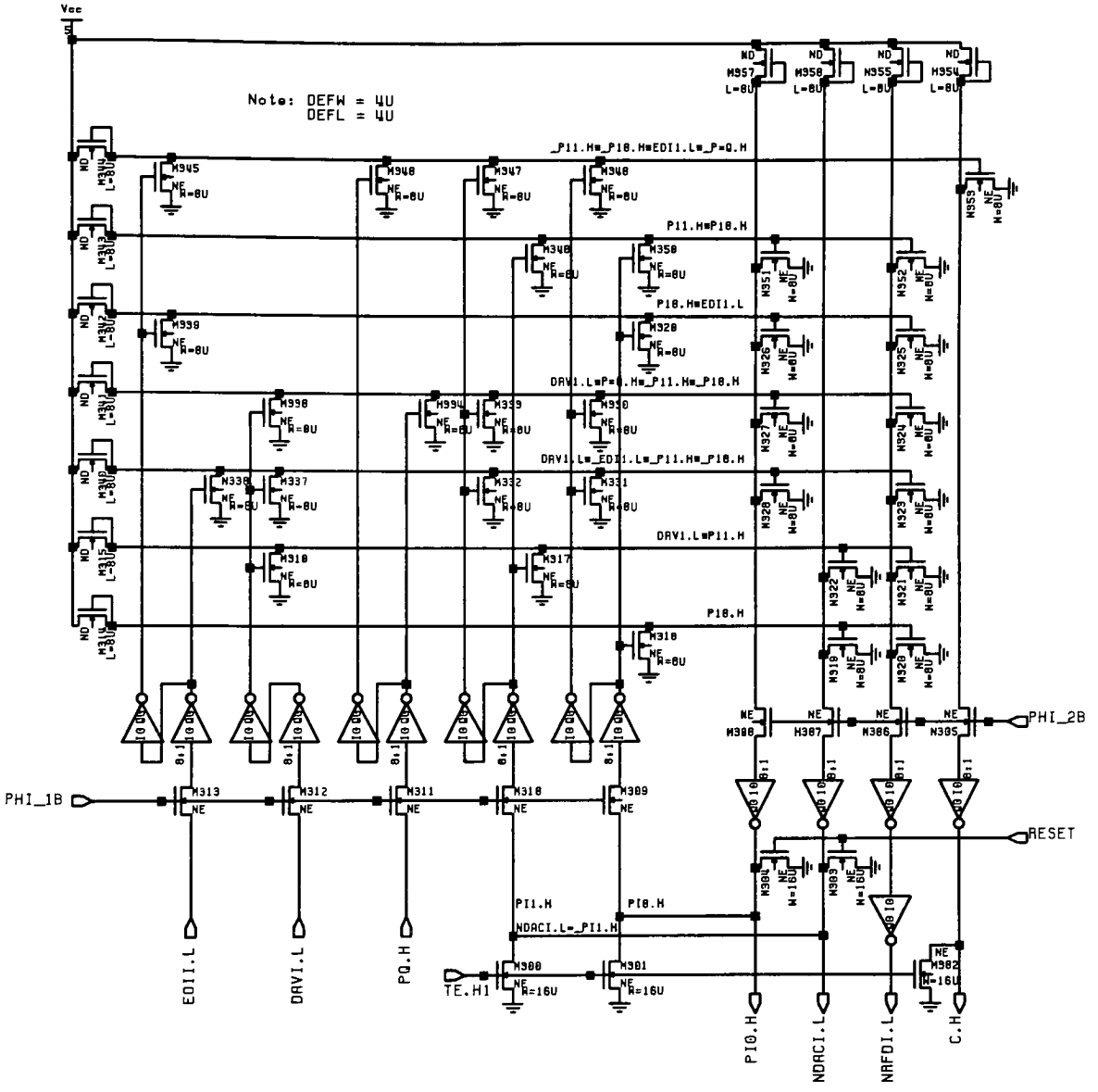


Figure 19 Input State Machine PLA

TABLE 9 Quick Reference Summary of the ISM Inputs and Outputs

Signal	Type	Name and Function
RESET	I	**RESET:** Asynchronous input. When TRUE (HIGH) immediately forces the ISM into state PON where it will remain as long as RESET is TRUE.
TE.H1	I	**TALK ENABLE:** When asserted immediately forces the ISM into state PON where it will remain as long as TE.H=T. TE.H1 is the superbuffered equivalent of SMC output TE.H. TE.H is TRUE during the *data output phase* (while the buffer is in command mode or talker mode). When RESET is FALSE and TE.H=F, the Input State Machine is free to respond to its inputs. TE.H is FALSE during the *data input phase* (while the buffer is in listener mode).
PHI_1	I	**PHI_1:** ψ1 clock signal.
PHI_2	I	**PHI_2:** ψ2 clock signal.
EOII.L	I	**End or Identify:** When TRUE indicates that the talker will be sending the last data byte to the buffer. EOII.L is the synchronized equivalent of the asynchronous bus signal EOI which is driven by talker during *data input phase.*
DAVI.L	I	**Data Valid:** When TRUE indicates that the talker has a valid, stable data byte on the eight DIO lines. DAVI.L is the synchronized equivalent of the asynchronous bus signal DAV which is driven by talker during the *data input phase.*
PQ.H	I	**P=Q:** When asserted (HIGH) indicates that the output of the four cascaded SN74ALS561 counters is equal to the output of the two SN74ALS573 latches. The two SN74ALS518 comparators compare the output of the four cascaded counters with the output of the two latches. If the outputs are equal, PQ.L=T; if the outputs are not equal, PQ.L=F. PQ.H is the synchronized and inverted equivalent of the asynchronous signal PQ.L delayed by one ψ1-ψ2 clock cycle. PQ.H=>T informs the Input State Machine that the *final count value* is contained in the two SN74ALS573 latches and that the ISM should deactivate C.H to latch the *final count value.*

TABLE 9. Quick Reference Summary of the ISM Inputs and Output [continued]

PI0.H	O	**Present State Variable PI0:** Represents the LSB of the two present state variable bits. This signal is fed back to the input of the ISM. It is also used as an input to the CMD MUX and to the logic that generates the signal LATCH.H.
C.H	O	**Latch Enable:** This signal drives ASIC output pin LE.H. When C.H=T, LE.H=T and vice versa.
NDACI.L/ PI1.H	O	**Not Data Accepted/Present State Variable PI1:** This output has a dual function. As NDACI.L, it is used by the buffer to indicate acceptance of a data byte on the DIO8-DIO1 lines. NDACI.L is connected to the OFF_CHIP port on ASIC pad NDAC.L. When the bus signal NDAC goes FALSE (HIGH), it indicates that the buffer has accepted the data byte and that the talker may now deactivate DAV. NDAC.L is used to increment the four cascaded counters also. As PI1.H, this signal represents the MSB of the two present state variable bits which is fed back to the input of the ISM. PI1.H is also used as an input to the CMD MUX and to the logic that generates the signal LATCH.H.
NRFDI.L	O	**Not Ready For Data:** This signal is connected to the OFF_CHIP port on ASIC pad NRFD.L. This signal is used by the buffer when in listener mode to indicate whether or not it is ready to receive another data byte. NRFDI.L drives NRFD.L which in turns drives the bus signal NRFD during the *data input phase*.

8.5.1 State Power ON (PON)

During a reset condition (RESET=T), the ISM is held in state PON (PI1.H=F, PI0.H=F) by pull-down transistors M303 & M304. The ISM is again forced into state PON by pull-down transistors M300 & M301 during the *data output phase* (TE.H1=T). Activation of the ISM commences after RESET=>F or whenever the buffer makes the transition from bus controller back to active listener (DC.H=>T, TE.H=>F). Activation means that the ISM is no longer **held** in state PON; it can respond to its inputs and change states. The ISM will not leave state PON however until the talker initiates communication. While the ISM idles in state PON, it assumes an active listener configuration - NDACI.L=T, NRFDI.L=F. Hence, the buffer drives bus signals NDAC LOW and NRFD HIGH. The talker begins communication by placing a data byte on the DIO lines and driving the bus signal DAV TRUE (LOW). DAV going TRUE causes DAVI.L=>T after a $\psi 1$-$\psi 2$ synchronization delay. The ISM remains in state PON until DAVI.L=>T at which point state RDY is entered during the following $\psi 2$ clock pulse provided EOII.L=F. EOI is only asserted by the talker when it is sending the last byte of current file transfer.

If EOII.L is TRUE in state PON, the ISM asserts the conditional output C.H. C.H drives the ASIC output pad LE.H. The LE.H pin is tied to the active-high latch enable pin on the two SN74ALS573 octal latches. When TRUE, LE.H enables the outputs of the latches to respond to there inputs which in this case is the memory address of the last data byte. The output of the two latches are routed to the Q inputs of the two octal identity comparators where they are compared to the output of the memory address counters. Since these two values are the same, the comparators' NANDed P=Q outputs force PQ.L TRUE. After asserting C.H, the ISM waits in state PON for the test input PQ.H to go TRUE. PQ.H is the synchronized and inverted equivalent of PQ.L. PQ.H=>T indicates that the *final count value* has been loaded into the SN74ALS573 latches. The ISM responds to PQ.H=>T by deactivating C.H. LE.H=>F disables the inputs to the two octal latches thus capturing the *final count value*. The comparison is done in state PON to make sure the last data byte memory location is captured before the counters are again clocked. C.H=>F ensures that the *final count value* can't be corrupted due to changes in the memory address count.

The SN74ALS573 latches will always contain some value. At power-up, the value is unknown. If the latches contain all zeros (0000_h) at power-up, then the ISM idles in state PON with PQ.H=T while waiting for the talker to initiate communication. When the buffer has completed a file transfer, the value held in the latches is the *final count value* from the previous file transfer. Therefore, the comparison between the present memory address value and the value contained in the latches could cause PQ.L to go TRUE after the acceptance of any random data byte during the current file transfer. This could happen if the number of bytes (*final count value*) in the previous file transfer is less than that in the present file transfer. PQ.H=>T before the assertion of EOII.L doesn't present a problem however. If PQ.H=T while EOII.L=F, C.H is not asserted. The ISM simply ignores PQ.H=T and continues receiving data. When the **next** byte is accepted (NDACI.L=>F), the memory address count is incremented and PQ.H=>F. EOII.L will not be TRUE until the last data byte of the current file transfer is sent. Once EOII.L=>T, the ISM asserts C.H and replaces the previous *final count value* with that of the present file transfer.

C.H is also not asserted when the present file transfer has ***exactly the same number of bytes*** as the previously transferred file. This could happen if a copy of the previous file transfer is desired. This results in PQ.H already being asserted when EOII.L **goes** TRUE. In this case, there is no need to latch the *final count value* since it is already captured in the SN74ALS573 latches. If this were not the case, P would not equal Q. Therefore, the ISM does not assert C.H. It simply waits in state PON for DAVI.L=>T and then moves to state RDY.

Present state bits PI1.H and PI0.H along with superbuffered SMC output DC.HB load the semistatic register stage of the Command Multiplexer with a logic ZERO every time state PON is entered during the *data input phase*. This forces the Command Multiplexer output MUX.L to go/remain TRUE. When MUX.L=T, the inverted intended receiver's listen address command appears at the CMD8-CMD1 outputs of the multiplexer. See the section 8.9 titled The Command Multiplexer (Cmd MUX) for more information on its operation.

8.5.2 State Ready (RDY)

In state RDY, the ISM drives NRFDI.L TRUE thus forcing NRFD.L and NRFD LOW. NRFD going TRUE indicates that the active listener is not ready to receive *another* data byte from the talker.* The ISM remains in state RDY for only one $\psi 1$-$\psi 2$ clock period 500ns at 2MHz. If test input EOII.L is FALSE, the ISM enters state DAC; if EOII.L is TRUE (last data byte), state LATCH is entered.

8.5.3 State Data Accepted (DAC)

In state DAC, the ISM forces NDACI.L FALSE to indicate acceptance of the data byte on the DIO lines. The talker responds to NDAC going HIGH by deactivating DAV. DAVI.L follows DAV going FALSE after a $\psi 1$-$\psi 2$ synchronization delay. DAVI.L=>F causes the ISM to re-enter state PON during the following $\psi 2$ clock pulse. As the ISM re-enters state PON, NDACI.L is driven TRUE while NRFDI.L is driven FALSE. NDAC=>T and NRFD=>F informs the talker that the active listener is ready to receive another data byte.

Figure 20 displays the buffer's two buses and the components that they connect. The currently accessed memory cell is constantly being written to during the *data input phase* since CS.L is always TRUE and the SRAM's R/$\overline{W}$ pin is held LOW by ASIC output pin TE. The memory cell may have garbage written into it initially as the talker places the next data byte on the DIO lines in state PON but once the talker asserts DAV, the data byte will remain valid and stable until DAV is deactivated in state DAC. NDACI.L must remain TRUE for at least 150ns (the required memory access time for a 150ns device), after DAVI.L=>T in state PON, before going FALSE in state DAC. This requirement is easily met by passing through state RDY since the clock period is 500ns at 2MHz and one clock cycle is the least amount of time that any state can exist. Therefore, the 500ns provided in state RDY is adequate to ensure that the correct data byte has time to be written to the correct memory cell.

The LOW-to-HIGH voltage transition of NDACI.L in state DAC is used to clock the four cascaded counters and indicate acceptance of the data byte present on the DIO lines. NDAC.L=>F increments the memory address counters causing the next memory location to be addressed. The counters' outputs must stabilize before data can be written to the desired memory location. The memory device used requires a stable address at its A14-A0 inputs for at least 150ns. The maximum delay through the SN74ALS561 counters from the clock input going HIGH (NDAC.L=>F) to any output changing logic levels is 18ns. The next memory cell addressed **must** be accessed *before* the talker deactivates the bus signal DAV. DAV (the bus signal not DAV.L or DAVI.L which occurs after some delay) going FALSE indicates that valid, stable data is no longer present on the DIO lines. If DAV goes FALSE *before* the next memory cell is accessed, there is a danger that the previous cell contents might be overwritten

---

*The active listener (buffer) **is ready** to receive the multiline message currently present on the DIO lines and does so by deactivating NDAC in state DAC. The active listener **isn't ready** to receive *another* data byte besides the one currently on the DIO lines. After accepting the data byte, it holds NRFD LOW until it is ready to receive the next data byte. NRFD going HIGH in state PON indicates that the active listener is ready to receive *another* data byte.

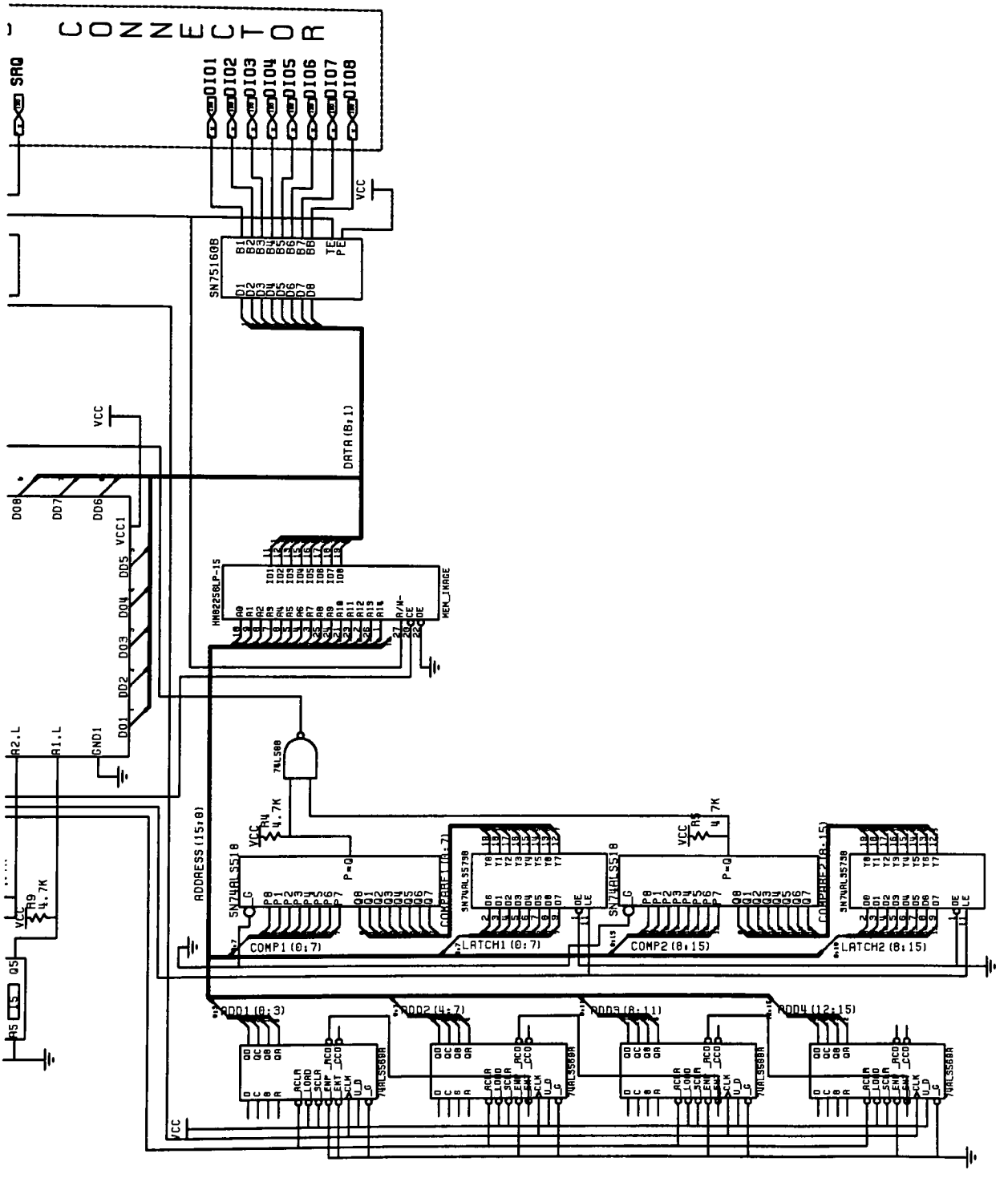


Figure 20 ADDRESS (15:0) Bus and DATA (8:1) Bus

with garbage. This has not been a problem in systems using the discrete designs and should not be a problem with the integrated design if the time for the talker to respond to NDAC=>F is greater than the time necessary to access the next memory location. If this does present a problem, the use of faster SRAM may provide relief. The talkers that the author has had experience with, take hundreds of microseconds to deactivate DAV in response to NDAC going HIGH. Therefore, the next memory location is accessed long before the previously addressed valid data byte is removed from the DIO lines. Hence, the previous data byte is initially written to the new memory location, then it is over written by garbage as the talker places the next data byte on the DIO lines, and finally, the correct information is written once the new data byte stabilizes on the DIO lines.

8.5.4 State LATCH

The ISM moves from state RDY to state LATCH when test input EOII.L is TRUE. In state LATCH, the ISM drives NDACI.L FALSE to indicate acceptance of the final data byte. NDAC.L=>F causes the memory address counters to increment which results in PQ.H=>F after a $\psi 1$-$\psi 2$ synchronization delay. The internal ASIC signal LATCH.H (equation 8) is also asserted in state LATCH. LATCH.H=>T indicates that the ISM is in state LATCH.

$$\text{LATCH.H} = \overline{\text{PI0.H}} * \overline{\text{PI1.H}} \qquad (8)$$

$$\text{TE.H} = \overline{\text{DAVI.L}} * \overline{\text{EOII.L}} * \text{LATCH.H} \qquad (9)$$

Note from the ASM chart that there is no exit path from ISM state LATCH. The ISM waits in state LATCH for the talker to release the bus, DAVI.L and EOII.L both going FALSE, and then stays in state LATCH. This satisfies the conditions necessary to force SMC output TE.H (repeated in equation 9) to go TRUE after a $\psi 1$-$\psi 2$ clock cycle. Since DC.H is the complement of TE.H, DC.H is forced FALSE. The SMC's outputs switching causes the buffer to make the transition from active listener to bus controller. TE.H1=>T turns ON ISM pull-down transistors M300, M301, and M302. Transistors M300 and M301 pull present state variables nets PI1.H and PI0.H FALSE thereby forcing the ISM into state PON where it remains during the entire *data output phase*. Transistor M302 pulls ISM output C.H FALSE. This ensures that the inputs to the two octal latches are disabled during the *data output phase* thus preventing corruption of the *final count value*.

During the *data output phase*, the EOII.L and DAVI.L inputs follow there respective signal at the EOI.L and DAV.L pads after a $\psi 1$-$\psi 2$ synchronization delay. The PQ.H input will also follow the inverted voltage at the PQ.L pad after a $\psi 1$-$\psi 2$ synchronization delay. The PI0.H, PI1.H, and C.H outputs will be prevented from responding to these inputs since they are pulled LOW through transistors M300, M301, and M302, respectively while TE.H1=T. The NRFDI.L output is not forced LOW while TE.H1=T and will respond to the inputs EOII.L, DAVI.L, and PQ.H. However, NRFDI.L is prevented from interfering with the signal NRFD.L since the NRFD.L pad driver is disabled while DC.HB=F.

8.5.5 ISM Operation Summary

The ISM is responsible for conducting listener handshaking during the *data input phase*, for incrementing the data bytes received count, and for latching the *final count value*. The ISM idles in state PON until the talker commences communication (DAVI.L=>T). At that point, the ISM enters state RDY and asserts NRFDI.L. NRFDI.L=>T causes NRFD.L and NRFD to follow it LOW. The ISM remains in state RDY for only one clock cycle and then enters state DAC where it deactivates NDACI.L. NDACI.L=>F forces NDAC.L and NDAC HIGH. NDAC.L=>F increments the memory address counters causing the next memory location to be addressed. NDAC going HIGH informs the talker that the active listener has accepted the data byte on the bus's eight DIO lines. The talker responds to NDAC going FALSE by deactivating DAV. The ISM remains in state DAC until DAVI.L=>F and then re-enters state PON. The ISM continues to cycle through states PON, RDY, and DAC as the talker transfers the data file a byte at a time. Each time the ISM enters state PON it checks the test input EOII.L. The talker asserts EOI to inform the active listener that the last data byte is being sent. If EOII.L is TRUE, the ISM asserts the conditional output C.H. C.H drives ASIC pad LE.H (Latch Enable). LE.H is connected to the enable pins on the two SN74ALS573 latches. LE.H=>T causes the latches to accept the present memory address as input. Since the output of the memory address counters now equals the output of the two latches, the two SN74ALS518 comparators' P=Q outputs go HIGH. This results in the PQ.L pin on the ASIC going TRUE. While EOII.L is TRUE, the ISM remains in state PON until both PQ.H and DAVI.L go TRUE. At that point, the ISM deactivates C.H and moves to state RDY. LE.H going FALSE captures the *final count value* in the two latches. The ISM spends one clock cycle in state RDY and then enters state LATCH where it deactivates NDACI.L. The ISM then waits in state LATCH for the talker to deactivate DAV and EOI. When DAVI.L and EOII.L go FALSE, the SMC outputs switch (DC.H=>F, TE.H=>T) and the buffer makes the transition from active listener to bus controller. TE.H1 going TRUE forces the ISM into state PON where it remains during the *data output phase*.

8.6 A Detailed Look at the Transition from Active Listener to Bus Controller

The RIT IEEE-488 Buffer Controller accepts the last data byte from the talker by entering ISM state LATCH and deactivating NDACI.L. The internal ASIC signal LATCH.H is also asserted as state LATCH is entered. Having completed the file transfer, the talker responds to NDAC going FALSE by releasing the bus - deactivating DAV and EOI.* DAVI.L and EOII.L follow there respective bus signals going FALSE after a $\psi 1$-$\psi 2$ synchronization delay. Note from its ASM chart (Fig. 18) that the ISM remains in state LATCH even after DAVI.L and EOII.L go FALSE. This condition results in equation 1 being TRUE which leads to the SMC's outputs switching, TE.H=>T and DC.H=>F, after a $\psi 1$-$\psi 2$ clock cycle.

The SMC's outputs switching causes the buffer to make the transition from active listener to active controller. The transition also defines the end of the *data input phase* (DC.H=T, TE.H=F) and the beginning of the *data output phase* (DC.H=F, TE.H=T). The buffer assumes control of the bus in order to address the intended receiver of the stored data file. Once the intended receiver has been made the active listener via a bus command issued by the active controller, the buffer assumes the role of active talker and begins transfer of the data file.

The switching of the SMC's outputs (DC.H=>F, TE.H=>T) causes the occurrence of several simultaneous events. ASIC pad drivers IFC.L, REN.L, ATN.L, EOI.L, and DAV.L become enabled while the NDAC.L and NRFD.L pad drivers are tristated, the direction of signals passed through the GPIB transceivers are reversed, the external SRAM I/O pins are tristated (CE.L=>F due to CS.L=>F), the Delay Counter is started (TE.H1=>T), and the ISM is forced into state PON (TE.H1=>T) where it idles during the *data output phase*.

The TRI_CTL port on ASIC pads IFC.L, REN.L, ATN.L, EOI.L and DAV.L are connected to TE.H2. The drivers of these bi-directional pads are tristated (pads act as inputs) during the *data input phase* (TE.H2=F) but become enabled (pads act as outputs) when TE.H=>T. The direction in which these signals travel through the SN75161 GPIB transceiver also reverses upon TE.H=>T and DC.H=>F. The buffer therefore, drives bus signals IFC, REN, ATN, EOI, and DAV during the *data output phase*. The OFF_CHIP port on pad IFC.L is tied HIGH. Thus, IFC is never asserted by the buffer. Bus signals ATN, EOI, and DAV are all driven by the OSM via its ATNO.L, EOIO.L, and DAVO.L outputs. These signals are all driven HIGH while the OSM idles in its initial state (TRANS). The OSM idles in state TRANS during the *data input phase* and for approximately 100µs after the SMC outputs switch. Therefore, bus signals ATN, EOI, and DAV are all driven HIGH (FALSE) by the buffer once TE.H2 goes TRUE not 100µs later. A more detailed description of the OSM operation is given in the section 8.8.

TE.H=>T also reverses the direction of the multiline messages passed through the SN75160 GPIB transceiver. While TE.H is FALSE, data passes from the bus through the transceiver to buffer. In the *data output phase*, instructions and data pass from buffer to bus.

---

*A talker's bus interface *may* still be in talk mode but since it isn't currently asserting (driving LOW) any bus signals, it acts as if it is OFF the bus. Provided the talker's bus drivers meet the specifications set forth in the IEEE Standard 488-1978, there shouldn't be a problem when the talker's bus drivers are pulled LOW by the bus drivers of the buffer.

The buffer side of the SN75160 isn't driven immediately after the SMC's outputs switch since both the SRAM outputs and the buffer's DO8-DO1 pads are tristated. However, the bus side of the transceiver is designed to float HIGH (≈ 3.3 volts) when its respective inputs on the device side aren't driven. Therefore, it appears to instruments on the bus that no command byte or device dependent data byte is present on the DIO lines.

The TRI_CTL port on pads NDAC.L and NRFD.L are connected to DC.H. In the *data input phase,* DC.H is TRUE and these bi-directional pads behave as outputs. Once DC.H=>F, the pad drivers on these pads are tristated and the pads behave as inputs. The direction which these three signals pass through the SN75161 GPIB transceiver also reverses with DC.H=>F and TE.H=>T. NDAC and NRFD go from being sourced by the buffer to being inputs to the buffer sourced by other instruments on the bus. Immediately after the transition, there won't be any active listener on the bus and these signals will not be driven. The NDAC and NRFD signal lines are implemented using a wired-OR configuration. Therefore, these signals should all pull HIGH after the transition.

When DC.H=>F, ASIC output CS.L is forced FALSE also. CS.L is connected to the CE.L pin on the buffer's SRAM. Therefore, CE.L will follow CS.L going HIGH. When CE.L goes FALSE, the SRAM enters reduced power standby mode and its eight I/O pins are placed in high impedance. The SRAM outputs are tristated because the buffer will be driving the DIO lines when it addresses the intended receiver to listen.

While the buffer's GPIB transceivers are switching the direction of signals passed through them, the signals lines on either sides of the transceivers may experience electrical noise - spikes, ringing, etc. For instance, the bus signals DAV and EOI drive the internal ASIC signals DAVI.L and EOII.L during the *data input phase* but become driven by ASIC signals DAVO.L and EOIO.L after the transition. Similarly, bus signals NDAC and NRFD are sourced by the ASIC signals NDACI.L and NRFDI.L during the *data input phase* but after the transition, these bus signal are sourced externally and drive the ASIC signals NDACO.L and NRFDO.L. In this potentially noisy environment, it is important to ensure that outputs don't respond to glitches at their inputs. The buffer must therefore wait for the transition to complete and for voltage levels to stabilize before accepting any signals as valid input. The Delay Counter, discussed in section 8.7, prevents the OSM from responding to voltages at its inputs for approximately 100µs after the start of the transition.

The OFF_CHIP port on pad REN.L is tied to ground. When TE.H=>T, the ASIC output REN.L goes TRUE causing the bus signal REN to go LOW.* REN is used to place instruments into remote programming mode once they are subsequently addressed. The buffer waits approximately 100µs before sending any multiline message over the bus after the assertion of REN. Because the time involved is so much greater than the system's clock period (100µs versus 500ns), the ASIC requires a method for determining when the 100µs delay has expired. This function is accomplished by the Delay Counter.

***Warning:** REN can only be asserted by the system controller. Hence, the buffer acts as the system controller when it asserts REN. This is the reason why the buffer shouldn't be used in a system with another controller.

8.7 The Delay Counter

The author was lead to believe by a Hewlett-Packard tutorial on the bus, that a 100µs delay was necessary after the **assertion** of REN before sending any multiline message over the bus. The Delay Counter's function is to hold the Output State Machine in its initial state for approximately 100µs after the SMC's outputs switch (TE.H=>T, DC.H=>F). This prevents the OSM from addressing a listener and beginning transfer of the stored data file before the 100µs delay has expired. The author learned after the ASIC design, simulation, and layout was complete that the IEEE Standard 488-1978 actually requires instruments to respond (enter local mode) to REN going **FALSE** within 100µs. Therefore, the 100µs delay after the assertion of REN isn't really needed. However, the 100µs delay does provides more than enough time for any line noise to subside or line capacitance charging/discharging to occur in response to the two GPIB transceiver switching the direction of signals pass through them. Once the Delay Counter has timed out, it releases its hold on the OSM which is then free to respond to its inputs and change states.

The Delay Counter (Fig. 21) is a synchronous 5-bit binary down counter with a superbuffered storage cell on its output. It consist of five toggle flip-flops, a safe semistatic register stage, an inverting superbuffer, and some glue logic. The Delay Counter has three inputs, TE.H1, PHI_1A, and PHI_2A, and one output, HOLD.H. The combine _Q outputs of the toggle flip-flops, C4-C0, represent the count value with C4 as the MSB and C0 as the LSB. When enabled (TE.H1=T), the module down counts from 31 decimal (11111_b) to zero (00000_b) and then rolls over to 31 decimal again on the next decrement. If TE.H1=F, the Delay Counter is held in its reset condition, C4-C0 = 11111_b and HOLD.H=T. While HOLD.H is TRUE, the OSM is held in its initial state.

The counter takes six ψ1-ψ2 clock cycles to decrement the count one digit. For instance, it takes six ψ1-ψ2 clock pulses for the count to go from 11101_b to 11100_b Each ψ1-ψ2 clock cycle is 500ns long @ 2MHz; thus, six ψ1-ψ2 clock cycles equals 3µs. Therefore, 3µs/count times 32 counts equals 96µs. Another ψ1-ψ2 clock cycle is added going through the semistatic register at the output of the Delay Counter module. Thus, the unit provides 96.5µs of delay between successive counts.

When the count reaches 00000_b, the output of the five input NOR gate, OR5, goes HIGH and the storage cell is loaded with a logic ONE during the next ψ1 clock pulse. At the following ψ2 clock pulse, the logic ONE appears at the output of the storage cell and is run through the inverting superbuffer causing the Delay Counter output HOLD.H to go FALSE. Once HOLD.H goes FALSE, the OSM is free to respond to its inputs and change states. Since HOLD.H transitions during the ψ2 clock pulse, it is a stable-ψ1 signal.

While the buffer is operating in the *data input phase* (TE.H1=F), the Delay Counter is held in reset - C4-C0 = 11111_b and HOLD.H=T. Since the count value of 11111_b is held for more than six clock cycles, the counter doesn't wait the full 3µs before decrementing the count once TE.H1=>T. Instead it moves from 11111_b to 11110_b in just 500ns. Therefore, 2.5µs must be subtracted from the designed delay value of 96.5µs to get the total delay provided by the counter. Hence, the Delay Counter provides 94µs of delay between the time TE.H1=>T and

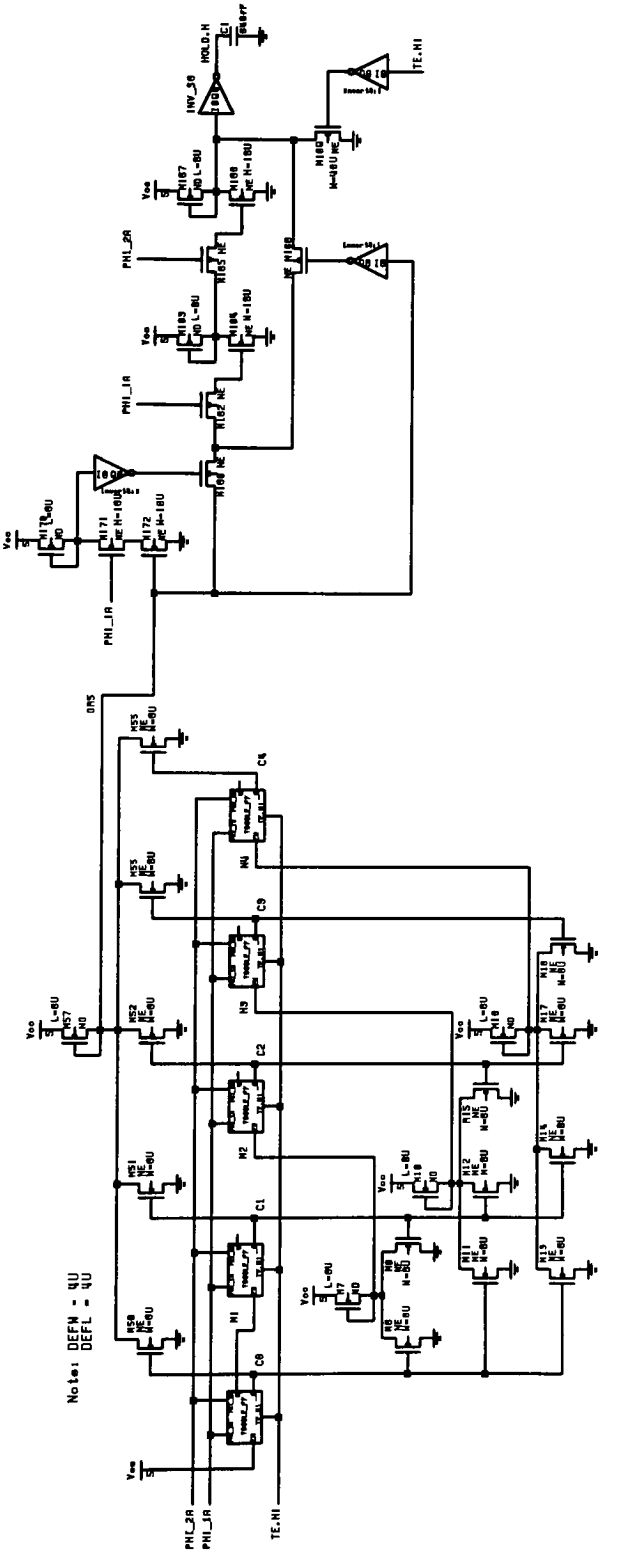


Figure 21 Delay Counter Module Schematic

the release of the OSM (HOLD.H=>F).

The schematic for the toggle flip flop is shown in figure 22. The toggle flip flop has a reset input (TE.H1), an enable input (EN), two clock inputs (PHI_1A and PHI_2A), and two complementary outputs (Q and _Q). When TE.H1 is FALSE, the toggle flip flop is held in a reset condition, Q forced to a logic ZERO and _Q forced to a logic ONE. When TE.H1 is HIGH, the EN input controls the mode in which the flip flop operates. When EN is LOW, Q and _Q retain there previous logic values and when EN is HIGH, the Q and _Q outputs toggle. The toggle flip flop contains a five stage shift register which together with clocked pass transistors M4 and M9 delay the Q and _Q outputs from switching for six clock cycles. This results in the 3μs delay between count decrements when the Delay Counter is running at 2MHz.

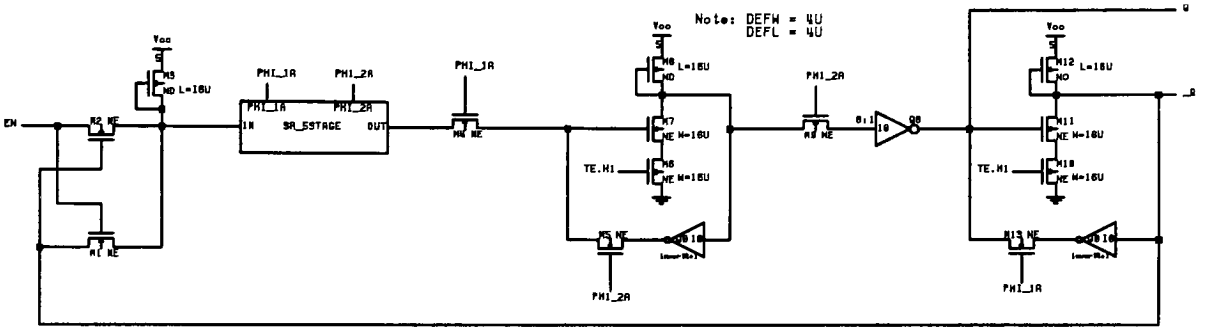


Figure 22 Toggle Flip-Flop With 5 Stage Shift Register

While the toggle flip flop is held in reset (TE.H1=F), the output of the two NAND gates, M8/s and _Q, are HIGH. If the flip flop has been in reset long enough, five clock cycles, the logic ONE at _Q will have been fed back through the five stage shift register and will appear at M7/g when pass transistor M4 is clocked by $\phi 1$. The two input NAND gate composed of transistors M6, M7, & M8 has a feedback loop to M7/g made up of an inverter and pass transistor M5. M5 turns ON and pulls the logic ONE at M7/g LOW at $\phi 2$. Since the output of the inverter in the feedback path and the voltage passed through M4 at $\phi 1$ are of opposite polarity while the flip flop is reset, the following indeterminate condition may occur when TE.H1=>T. TE.H1 is a stable-$\phi 1$ signal. This means that when TE.H transitions, it does so during the $\phi 2$ clock pulse. The desired course of events is for the logic ONE at M7/g to be pulled LOW through the $\phi 2$ clocked pass transistor M5 before TE.H1 going HIGH turns ON M6. This results in the output of the NAND gate, M8/s, remaining HIGH. If however, TE.H1 going HIGH turns ON M6 before the logic ONE at M7/g is pulled LOW, the output of the NAND gate may start to fall. This would tend to cause the voltage at the output of the inverter in the feedback path to rise. This in turn could lead to M8/s going indeterminate. Since M9 is also clocked by $\phi 2$, the Q and _Q outputs may also go indeterminate. This would lead to unpredictable logic levels on the C4-C0 lines immediately after TE.H1 goes HIGH. In the worst case, TE.H1 going HIGH significantly before M5 turning ON would result in the output of the NAND going to a logic ZERO and the count going from 11111_b to 00000_b in 500ns (instead of 94μs) providing virtually no delay.

Simulation of the entire ASIC shows no indeterminate condition occurring within the toggle flip flops. However, the simulation doesn't take into account any potential clock skew on the PHI_2A line. In the chip layout, the PHI_2B clock is used to gate the logic generating TE.H. A visual inspection of the clock lines indicates that the PHI_2B line is less loaded than the PHI_2A line used to clock the Delay Counter. However, the TE.H1 is very heavily loaded. In terms of chip performance, the author is not sure which signal, PHI_2A or TE.H1, would win the race discussed above. Its unfortunate that the author couldn't get the LPE netlist to run in AccuSim since this would have given a better indication of chip performance. See section 8.12.3 for further discussion on the TE.H1/PHI_2A signal race. If in an actual system, the buffer appears to accept data from the talker but does not successfully transfer data to a listener, the voltage level of the HOLD.H line should be monitored. This will have to be done at the chip level since this signal isn't brought out to a pin.

The potential indeterminate condition can only occur as TE.H1 goes HIGH. Once the Delay Counter is running, the voltage at M7/g will be the at the same logic level as the NAND output (M8/s). When TE.H1 is TRUE, the NAND gate essentially acts as an inverter and the fed back, inverted output will agree with the input at M7.

Figure 23 displays the simulation waveforms for the Delay Counter. The counter is initially in its reset condition (C4-C0 = 11111_b and HOLD.H=T) while TE.H=F. TE.H is forced FALSE at power-up and while the buffer is in listener mode. Once TE.H=>T, the counter starts decrementing the count. Note that the count immediately goes from 31 decimal (11111_b) to 30 decimal (11110_b) after TE.H1 goes HIGH. The EN input on the toggle flip flop sourcing C0 is tied to VCC thereby allowing its _Q output to toggle every six clock cycles - see the C0 waveform. The EN input to the flip flop sourcing C1 is tied to the Q output of the C0 flip flop. This results in the C1 flip flop being disabled for six out of every twelve clock cycles see the N1 waveform. Thus, the C1 flip flop toggles every 12 clock cycles. The NOR gates connected to the EN input of the three remaining flip flops control the enabling of these flip flops. The C2, C3, and C4 flip flops toggle every 24, 48, and 96 clock cycles, respectively. When the count reaches 00000_b, the output of the five input NOR gate (OR5) goes HIGH followed by HOLD.H going FALSE one clock cycle later.

The spikes shown on signal lines N4-N2 and OR5 are due to the C4-C0 inputs to the NOR gates simultaneously switching one input to the NOR going HIGH while another input is going LOW. These spikes present no problem to the logical operation of the Delay Counter since they subside long before the next $\psi 1$ clock pulse. The energy radiated by these spikes however might be of a concern. For this reason, the Delay Counter is positioned away from the rest of the cells in the chip layout in an effort to reduce the influence of any cross talk. A better design would have used gray coding of the count to avoid these glitches, since only one signal changes at a time. A 5 bit gray code approach resulted in a 22 term PLA which the author thought might be prohibitive in terms of chip real estate. With the chip layout now completed, it seems that ample room for such a PLA is present. If a revision of the chip is undertaken, a gray coding solution should be considered.

After the OSM is released (HOLD.H=>F), the Delay Counter continues to down count since TE.H1 is TRUE during the *data output phase*. The register stage at the Delay Counter's

output accepts input only when the count value is equal to zero (00000_b). Therefore, the register stage will be loaded with a logic ONE every 96μs. The register stage is already loaded with a logic ONE from the first time the count reached 00000_b. Therefore, loading the register stage every 96μs doesn't change the Delay Counters output. The semistatic register is designed to refresh the data entered at its input every clock cycle when not being loaded. The only way for HOLD.H to go TRUE again is to reset the Delay Counter. This occurs when the buffer is finished with file transfer and again assumes the role of active listener (DC.H=>T, TE.H=>F, HOLD.H=>T).

8.8 The Output State Machine (OSM)

During the *data input phase,* the ISM performs active listener handshaking and capture of the *final count value* while OSM idles in state TRANS. During the *data output phase*, the ISM idles in state PON while the OSM takes on the responsibility of addressing a listener and transferring the stored data file to the intended receiver. The OSM handles active controller and active talker bus protocol by driving bus signals ATN, DAV, and EOI via its ATNO.L, DAVO.L, and EOIO.L outputs. It also controls tristating of the ASIC DO8-DO1 pad drivers with its G.H output. The OSM present state bit PO2.H is used as an input to the combinational logic sourcing the signal CS.L and OSM present state bits PO1.H and PO0.H are used as inputs to the combinational logic which generates the signal CMD_ACC.H. EOIO.L is also used as an input to the Command Multiplexer and causes the UNL command to appear at the multiplexer's outputs when TRUE.

The ASM Chart and Truth Table for the OSM are displayed in Figure 24 and Table 10. The OSM uses only six of the eight possible states provided by the three OSM Present State Bits (PO2.H, PO1.H, & PO0.H). States 110 and 111 are not used. To prevent a state machine from hanging in an unused state, an exit path to a desired state is usually provided via the addition of extra logic. Inclusion of the two unused states in an Espresso* input file resulted in a 16 product term PLA. A second Espresso input file **not** containing the two unused states generated a 15 product term PLA. The author thought the addition of the extra term was an unnecessary precaution since the OSM is forced to start up in a known initial condition and since all its inputs and outputs are stable-ψ1 signals. The author was also concerned about conserving chip real estate. Therefore, the OSM was implemented as a 15 term structure. Now that the chip layout has been completed, there appears to be ample room for the 16 term PLA. Should chip level debug indicate that the 16^{th} term would be useful, then it should be implemented in any revision of the ASIC layout.

The OSM (Fig. 25) is constructed from a 15 product term PLA with its inputs clocked by ψ1 and its outputs clocked by ψ2. All inputs and outputs of the PLA are stable-ψ1 signals. The 15 product term expressions are used as net names and are displayed on their respective nets between the AND and OR planes of the PLA. Table 11 provides a quick reference summary of the OSM I/O signals.

*Espresso is a logic minimization tool.

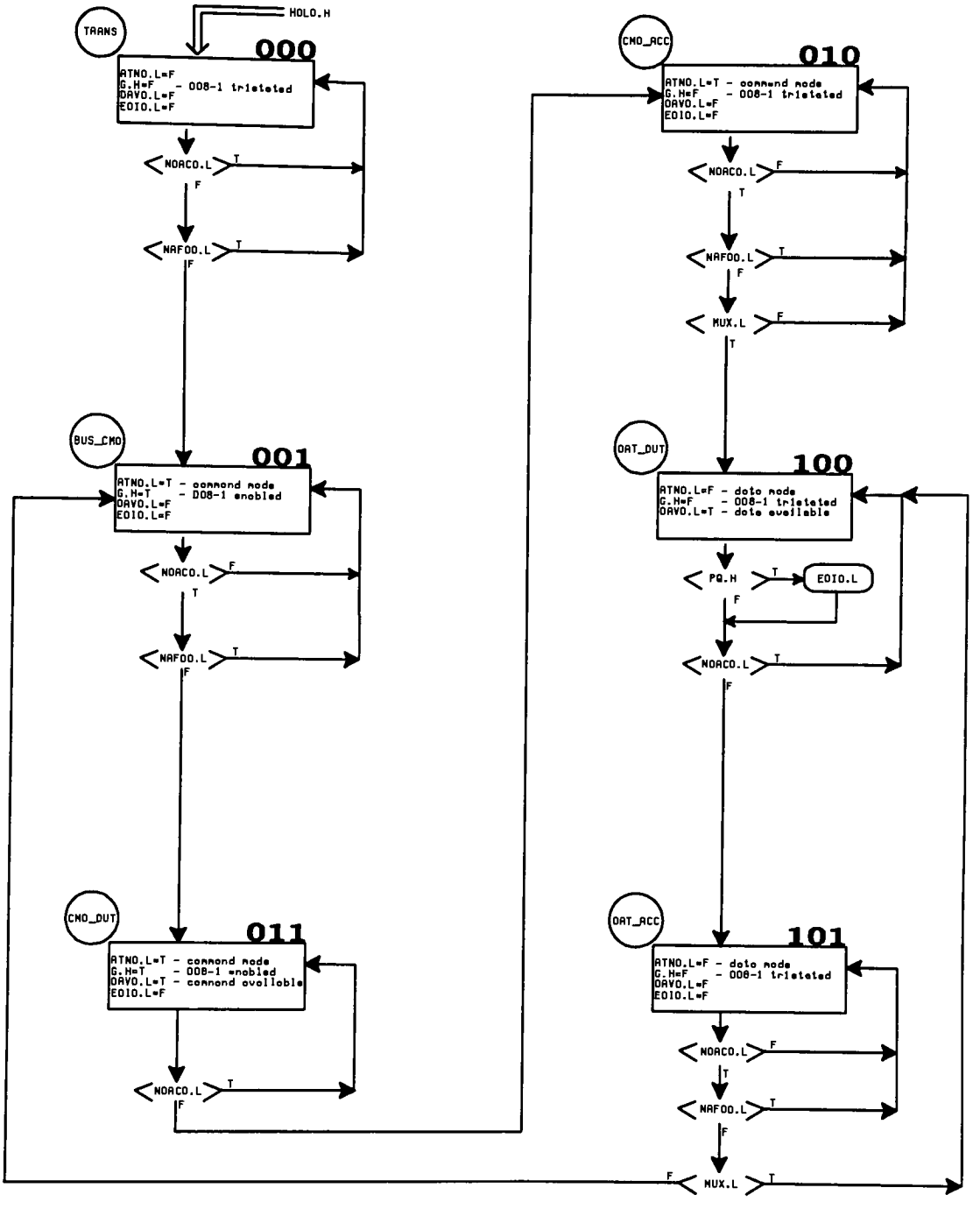


Figure 24 ASM Chart for the Output State Machine

TABLE 10 Output State Machine Truth Table

Present State				Inputs				Outputs				Next State			
NAME	P2	P1	P0	NOAC	NRFO	MUX	P=Q	ATN	G	OAV	EOI	N2	N1	N0	NAME
TRANS	0	0	0	T	X	X	X	F	F	F	F	0	0	0	TRANS
				X	T	X	X	F	F	F	F	0	0	0	TRANS
				F	F	X	X	T	T	F	F	0	0	1	BUS_CMO
BUS_CMO	0	0	1	F	X	X	X	T	T	F	F	0	0	1	BUS_CMO
				X	T	X	X	T	T	F	F	0	0	1	BUS_CMO
				T	F	X	X	T	T	T	F	0	1	1	CMO_OUT
CMD_OUT	0	1	1	T	X	X	X	T	T	T	F	0	1	1	CMO_OUT
				F	X	X	X	T	F	F	F	0	1	0	CMO_ACC
CMO_ACC	0	1	0	F	X	X	X	T	F	F	F	0	1	0	CMO_ACC
				X	T	X	X	T	F	F	F	0	1	0	CMO_ACC
				T	F	F	X	T	F	F	F	0	1	0	CMO_ACC
				T	F	T	X	F	F	T	F	1	0	0	OAT_OUT
OAT_OUT	1	0	0	T	X	X	T	F	F	T	T	1	0	0	OAT_OUT
				T	X	X	F	F	F	T	F	1	0	0	OAT_OUT
				F	X	X	X	F	F	F	F	1	0	1	OAT_ACC
OAT_ACC	1	0	1	F	X	X	X	F	F	F	F	1	0	1	OAT_ACC
				X	T	X	X	F	F	F	F	1	0	1	DAT_ACC
				T	F	T	F	F	F	T	F	1	0	0	OAT_OUT
				T	F	T	T	F	F	T	T	1	0	0	OAT_OUT
				T	F	F	X	T	T	F	F	0	0	1	BUS_CMO

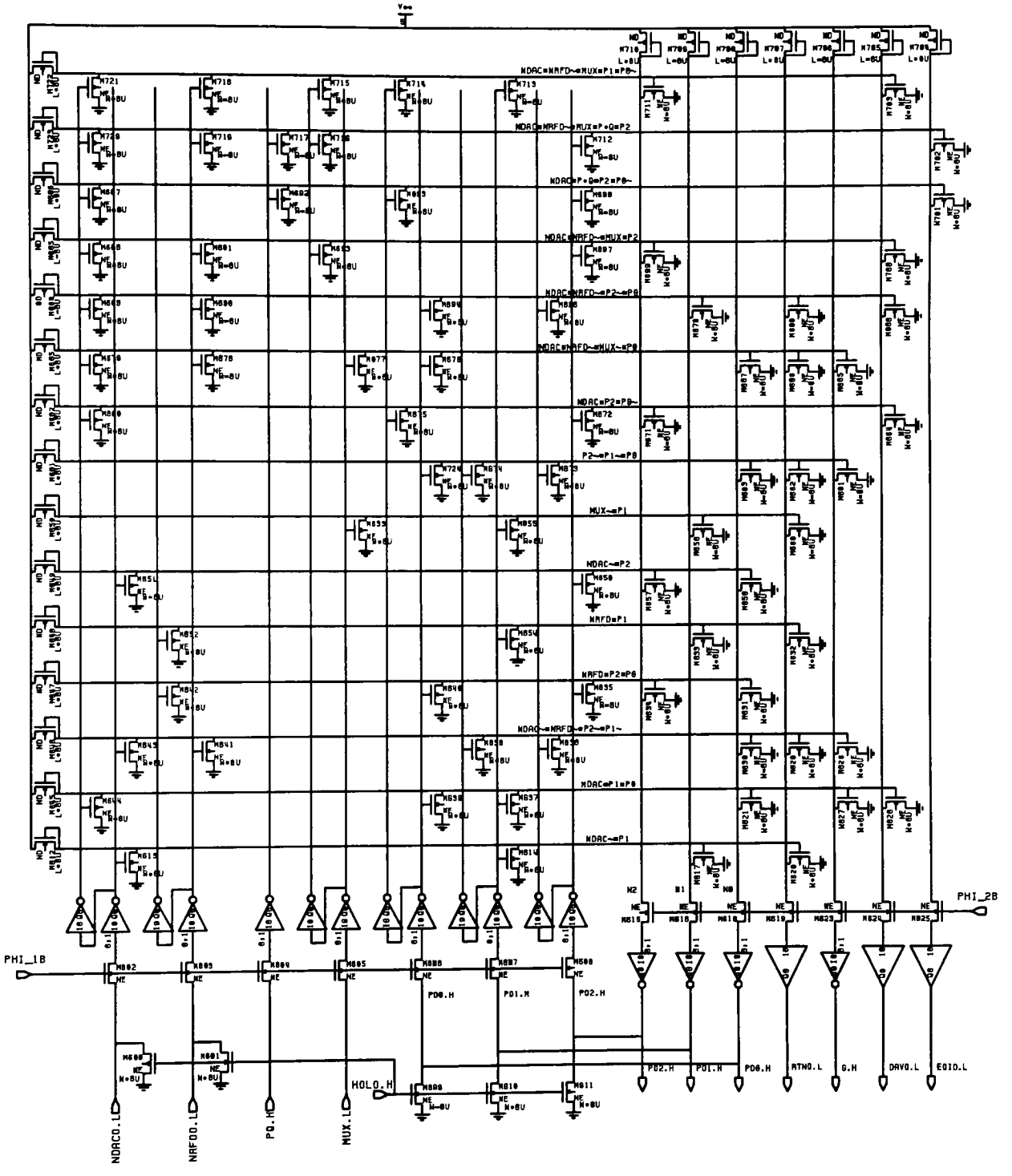


Figure 25 Output State Machine PLA Schematic

TABLE 11 Quick Reference Summary of I/O ports on the Output State Machine

Signal	Type	Name and Function
HOLD.H	I	**Hold:** While this signal is TRUE, the Output State Machine idles in state TRANS. When FALSE, the OSM is free to respond to its inputs and change states. HOLD.H is TRUE while the buffer is in listener mode and remains TRUE for approximately 100µs *after* the SMC outputs switch (DC.H=>F, TE.H=>T). HOLD.H=>F when the Delay Counter times out and remains FALSE until the buffer again resumes the role of active listener after file transfer has completed.
NDACO.L	I	**Not Data Accepted:** NDACO.L is the synchronized equivalent of the bus signal NDAC. When the buffer is in command mode (ATNO.L=T), NDACO.L=>F indicates that all devices on the bus have accepted the command byte on the bus's eight DIO lines. When the buffer is in talker mode (ATNO.L=F), NDACO.L=>F indicates that the addressed listener has accepted the data byte on the eight DIO lines. The OSM responds to NDACO.L=>F by deactivating its DAVO.L output.
NRFDO.L	I	**Not Ready For Data:** NRFDO.L is the synchronized equivalent of the bus signal NRFD. The assertion of NRFDO.L, indicates that one or more device on the bus is not ready to receive another byte across the bus's data lines. The OSM waits for NRFDO.L=>F before asserting DAVO.L.
PQ.H	I	**P=Q:** PQ.H is the synchronized and inverted equivalent of the PQ.L. When PQ.L is asserted, it indicates that the 16 bit value (present memory address) at the output of the four cascaded SN74ALS561 counters is equal to the value (*final count value*) at the output of the two SN74ALS573 latches. As the buffer sends data to the addressed listener during the *data output phase*, it compares the present memory address with the *final count value* stored in the two latches. If the two values are equal, the NANDed output (PQ.L) of the two SN74ALS518 comparators goes TRUE. PQ.H follows PQ.L=>T after a $\psi 1$-$\psi 2$ synchronization delay. The OSM responds to PQ.H=>T by asserting the conditional output EOIO.L in state DAT_OUT.
MUX.L	I	**Multiplexer:** This signal indicates which octal command is present at the CMD8-CMD1 outputs of the Command Multiplexer. MUX.L=T indicates that the inverted intended receiver's listen address command is present at the multiplexer's outputs. MUX.L=F indicates that the UNL (unlisten) command is present at the multiplexer's outputs. If MUX.L=T in state DAT_ACC, the buffer remains an active talker and

TABLE 11. Quick Reference Summary OSM I/O Ports [continued]

		re-enters state DAT_OUT. If MUX.L=F in state DAT_ACC, the OSM enters state BUS_CMD causing the buffer to resume the active controller role.
PO0.H	O	**OSM Present State Bit PO0:** Represents the LSB of the three present state bits. PO0.H and PO1.H are used as inputs to the combinational logic producing the signal CMD_ACC.H. CMD_ACC.H = PO1.H * $\overline{PO0.H}$
PO1.H	O	**OSM Present State Bit PO1:** Represents the middle bit of the three present state bits. PO0.H and PO1.H are used as inputs to the combinational logic sourcing the signal CMD_ACC.H. CMD_ACC.H = PO1.H * $\overline{PO0.H}$
PO2.H	O	**OSM Present State Bit PO2:** Represents the MSB of the three present state bits. PO2.H is used as an input to the three input NOR driving ASIC output CS.L. CS.L is TRUE while PO2.H is TRUE.
EOIO.L	O	**End or Identify:** The OSM asserts this signal in state DAT_OUT if its PQ.H input is TRUE. EOIO.L=T forces the bus signal EOI LOW thus notifying the addressed listener that the last data byte is being transmitted. EOIO.L=T also loads the Command Multiplexer with a logic ONE which results in the UNL message appearing at the CMD8-CMD1 outputs of the multiplexer.
DAVO.L	O	**Data Valid:** DAVO.L TRUE drives the bus signal DAV LOW thus notifying one or more devices that a stable bus command or device-dependent message is present on the bus's DIO8-DIO1 lines.
G.H	O	**ENABLE:** This signal is used to control the tristating of the ASIC's eight Data Output (DO8-DO1) pad drivers. When G.H=T, the DO8-DO1 pads drivers are enabled; when G.H=F, the DO8-DO1 pad drivers are tristated. This signal is only asserted in OSM states BUS_CMD and CMD_OUT when the buffer is either placing the intended receiver's listen address command on the bus or issuing the UNL (unlisten) command. At all other times, the Data Output pad drivers should be tristated to avoid bus contention with drivers of either the SRAM or SN75160 GPIB transceiver.
ATNO.L	O	**Attention:** This signal is only asserted (LOW) while the buffer is acting as the active controller (OSM states BUS_CMD, CMD_OUT, and CMD_ACC). ATNO.L is used to drive the bus signal ATN during the *data output phase*. The buffer will either be addressing a device to listen or issuing the UNL (unlisten) command while ATNO.L=T.

8.8.1 State Transition (TRANS)

At power-up, during a reset condition, or while the buffer is operating as an active listener, the OSM is forced to idle in state TRANS. This results from the fact that the SMC output TE.H is FALSE during the *data input phase*. The Delay Counter output HOLD.H is forced TRUE while TE.H=F. With HOLD.H=T, the OSM is held in state TRANS (PO2.H, PO1.H, PO0.H = 000) by pull-down transistors M609, M610, and M611 (Fig. 25). The desired condition in state TRANS is for all 15 product term nets to be at ≈ 0.0 volts which ensures that the OSM outputs are at there proper logic levels all FALSE. If one works through the corresponding nodal voltages, they will notice that the logical value of the three Present State Bits (000) forces 14 of the 15 product term nets to ≈ 0.0 volts. The only exception is the net labeled NDAC~*NRFD~*P2~*P1~ (where ~ indicates NOT TRUE). This net is consequently a function, in state TRANS, of OSM inputs NDACO.L and NRFDO.L. Net NDAC~*NRFD~*P2~*P1~ will be ≈ 0.0 volts if either NDACO.L or NRFDO.L is TRUE. Both NDACO.L and NRFDO.L will be TRUE since pull-down transistors M600 & M601 are ON while HOLD.H=T. Analysis of nodal voltages now leads to OSM outputs ATNO.L, DAVO.L, EOIO.L, and G.H as all FALSE in state TRANS.

Pull-down transistors M600 & M601 aren't really needed during the *data input phase* since NDACO.L and NRFDO.L are never both simultaneously FALSE while the buffer is in listener mode. NDAC and NRFD both FALSE indicates the absence of any listeners on the bus which is not the case while the buffer is an active listener. The buffer is driving ISM outputs NDACI.L and NRFDI.L while an active listener. Hence, it is driving ASIC pads NDAC.L and NRFD.L. If not for pull-down transistors M600 & M601, OSM inputs NDACO.L and NRFDO.L would follow the voltage at their respective ASIC pads after a $\psi 1$-$\psi 2$ synchronization delay. Since NDAC and NRFD are never both FALSE during the *data input phase,* NDACO.L and NRFDO.L would never be both FALSE. Therefore, net NDAC~*NRFD~*P2~*P1~ would always be forced LOW during the *data input phase.*

The utility of pull-down transistors M600 & M601 comes into play during the buffer transition from active listener to bus controller. M600 & M601 prevent NDACO.L and NRFDO.L from being effected by noise at ASIC pads NDAC.L and NRFD.L due to the tristating of there pad drivers and reversal of direction in which these signals pass through the GPIB transceivers. These pull-down transistors remain ON because the Delay Counter doesn't allow HOLD.H=>F for approximately 100μs after TE.H=>T. This provides enough time for stabilization of NDAC and NRFD at logic ZERO's (FALSE). NDAC and NRFD both HIGH indicates no listener present on the bus which is indeed the case until the buffer addresses a device to listen. If M600 & M601 weren't present, net NDAC~*NRFD~*P2~*P1~ would be at a logic ONE since both NDACO.L and NRFDO.L would be simultaneously FALSE. With net NDAC~*NRFD~*P2~*P1~ at a logic ONE, OSM transistors M628, M629, & M630 are turned ON which makes OSM outputs G.H, ATNO.L and PO0.H want to go TRUE. ATNO.L going TRUE would cause the bus signal ATN to follow it LOW since the ATN.L pad driver is enabled when TE.H=>T. In response to ATN going LOW, all devices on the bus would drive there NDAC line driver LOW. The OSM can not leave state TRANS, once HOLD.H=>F, unless both NDACO.L and NRFDO.L are FALSE. ATN going LOW in state TRANS would cause the OSM to hang in state TRANS since every device would be driving NDAC LOW.

The inclusion of pull-down transistors M600 & M601 provides a remedy to this problem. Bus signals NDAC and NRFD stabilize FALSE shortly after the NDAC.L and NRFD.L pad drivers tristate and signal direction through the GPIB transceivers reverses. With NDAC.L and NRFD.L HIGH, the synchronizers will try to drive NDACO.L and NRFDO.L HIGH. Pull-down transistors M600 & M601 will prevent NDACO.L and NRFDO.L from going HIGH while HOLD.H=T. Since OSM inputs NDACO.L and NRFDO.L are pulled LOW, net NDAC~*NRFD~*P2~*P1~ will be at logic ZERO and OSM outputs PO0.H, ATNO.L, G.H, and EOIO.L will be FALSE.

The Output State Machine remains in state TRANS until the Delay Counter times outs (HOLD.H=>F). Since HOLD.H is a stable-ψ1 signal, it will begin to change logic levels shortly after ψ2 goes HIGH. HOLD.H=>F will turn OFF the five pull-down transistors M600, M601, M609, M610, & M611. However, OSM outputs PO0.H, ATNO.L, and G.H will not go TRUE once HOLD.H=>F since HOLD.H was TRUE during the ψ1 clock pulse. Therefore, NDACO.L and NRFDO.L will have still been pulled LOW while the PLA inputs were clocked in during the ψ1 pulse. This results in PO0.H, ATNO.L, and G.H remaining FALSE when the PLA outputs are clocked out at the next ψ2 pulse. HOLD.H=>F during the **same** ψ2 pulse turns OFF M600 & M601 thereby allowing the synchronizer output to drive NDACO.L and NRFDO.L HIGH. Since HOLD.H is now FALSE, the OSM is free to respond to its inputs and change states. At the next ψ1 pulse, NDACO.L and NRFDO.L are both FALSE when the PLA inputs are clocked in which result in the OSM entering state BUS_CMD at the following ψ2 pulse since the expression NDACO.L * NRFDO.L is now TRUE.

8.8.2 State BUS COMMAND (BUS_CMD)

In state BUS_CMD, the OSM asserts ATNO.L and G.H. ATNO.L going LOW forces the bus signal ATN TRUE. The assertion of ATN places the buffer in command mode. If an instrument's bus interface isn't ready to receive a command byte, it responds within 200ns to ATN going TRUE by driving both its NDAC and NRFD lines drivers LOW. Then, when ready to receive the command byte, by driving its NRFD line driver back HIGH. The time needed for an instrument to prepare its interface to receive a command byte is a function of that device. Depending on when NRFD went LOW and for how long NRFD is kept LOW, the HIGH to LOW to HIGH transition of the NRFD bus line may or may not be reflected at the NRFDO.L output of the synchronizer. However, the OSM input NDACO.L follows the bus signal NDAC going LOW after a ψ1-ψ2 synchronization delay.

For instance, the author observed the following response using an IBM XT PC to address an HP Color-Pro graphics plotter while the IEEE-488 bus was monitored with a Tektronix's 1230 logic analyzer. The analyzer waveforms showed that when ATN went TRUE, both NRFD and NDAC went TRUE simultaneously. Then after approximately 800ns, the NRFD line went FALSE. NRFD held LOW for 800ns will surely be reflected at the synchronizer NRFDO.L output. NRFD LOW indicates that one or more instruments aren't yet ready to receive a message over the DIO lines. When the slowest device is ready for communication, it stops driving its NRFD line driver LOW and the wire-ORed NRFD signal line pulls HIGH. For this reason, the OSM remains in state BUS_CMD until the expression

NDACO.L*$\overline{\text{NRFDO.L}}$ is TRUE at which point the OSM enters state CMD_OUT.

The G.H line is tied to the TRI_CTL port on the eight ASIC DO pads. When G.H goes HIGH in BUS_CMD, the Command Multiplexer's outputs are driven onto the buffer's DATA(8:1) bus. Since CS.L is HIGH, the data I/O pins on the memory chip are in HIGH impedance so no fight exist between the drivers of the ASIC's DO pads and those of memory. The direction of data passed through the SN75160 GPIB transceiver is from buffer to bus in the *data output phase* (TE.H=T). In its first pass through state BUS_CMD, the buffer is addressing a device to listen so MUX.L is TRUE. Therefore, the inverted intended receiver's listen address command is driven from the ASIC DO8-DO1 pads through the SN75160 GPIB transceiver and out onto the bus's DIO8-DIO1 lines. After file transfer is complete, the OSM again passes through state BUS_CMD. This time MUX.L is FALSE and the $\overline{\text{UNL}}$ message is driven out onto the bus.

8.8.3 State COMMAND OUT (CMD_OUT)

In state CMD_OUT the OSM asserts DAVO.L. DAV going TRUE informs all devices on the bus that a valid command is present on the eight DIO lines. Devices respond to DAV going LOW by first forcing there NRFD bus driver LOW. When ready, each device then accepts the command byte by setting there respective NDAC bus driver FALSE. NDAC won't go HIGH until all of the devices have accepted the command byte since the NDAC line is driven by open collector drivers. NDACO.L follows the bus signal NDAC going HIGH after a $\psi 1$-$\psi 2$ synchronization delay. NDACO.L=>F causes the OSM to move from state CMD_OUT to state CMD_ACC.

The outputs of the Command Multiplexer (CMD8-CMD1) are driven onto the bus's DIO lines in state BUS_CMD. The command byte continues to be driven onto the bus in state CMD_OUT since G.H remains TRUE. Signals on the DIO lines need some time to stabilize before being accepted by the addressed listener (NDAC going HIGH). The IEEE-488 specifications recommends at least 700ns settle time for multiline messages when tristate drivers are used on the DIO, DAV, ATN, and EOI lines.*

As mention previously, the minimum duration for any state is 500ns when running at 2MHz, approximately 70% of the 700ns settle time needed. The buffer remains in state BUS_CMD for at least **two** clock cycles (1000ns) the *first* time this state is entered since there are no active listeners on the bus. If the bus interface for all devices are prepared to receive a command byte (NRFD=F, NDAC=T) in response to ATN going LOW before the next $\psi 1$ clock pulse, it takes one clock cycle for the signals to get through the synchronizers and a second clock cycle before the any change occurs at the OSM's outputs. Therefore, the 1000ns delay more than satisfies the 700ns settle time needed before DAVO.L is asserted in state CMD_OUT.

The OSM remains in state BUS_CMD for only **one** clock cycle (500ns) the *second* time this state is entered since the addressed listener meets the condition needed for the ISM to enter state CMD_OUT (NRFDO.L=F, NDACO.L=T). Therefore, the buffer provides only 70%

*This settle time has a few qualifications, see IEEE Standard 488-1978 pp. 55

of settle time before indicating that a valid, stable command is present on the bus by asserting DAVO.L in state CMD_OUT. If NDAC goes HIGH in less than 200ns after the assertion of DAV, the 700ns settle time will not be met and there is no guarantee that a valid message is accepted by instruments on the bus. The buffer design relies on devices not accepting the multiline command for at least 200ns after the assertion of DAV. This is not an unreasonable expectation since instruments must first notice that DAV is TRUE, then drive NRFD LOW, and finally drive NDAC HIGH. Also, the maximum data rate on the IEEE-488 bus that a signal line can operate at is 1Mb/s and this requires specials considerations.* Most devices typically operate at 250K-500Kb/s. Instruments which the author has worked with using the discrete buffer have taken several milliseconds to accept multiline message.

8.8.4 State COMMAND ACCEPTED (CMD_ACC)

Both G.H and DAVO.L are driven FALSE as the OSM enters state CMD_ACC. G.H going FALSE tristates the ASIC's DO8-DO1 pad drivers. Devices on the bus respond to DAV going FALSE by setting there NDAC driver TRUE and after some delay (when ready for the next command byte) by setting there NRFD driver FALSE. NRFD won't go HIGH however until all of the devices are again ready to receive a command byte since the NRFD line is driven by open collector drivers.

In state CMD_ACC, the signal CMD_ACC.H=>T since PO1.H=T and PO0.H=F (equation 10). CMD_ACC.H going TRUE causes ASIC outputs ACLR.L and CS.L to go TRUE. ACLR.L going TRUE resets the counters which causes the buffer to address memory location 0000_h the location of the first data byte in the stored data file. CS.L going TRUE causes the memory chip to leave reduced power standby mode. Once the memory chip becomes active, its I/O pins begin to drive the buffer's DATA(8:1) bus since its $R/\overline{W}$ pin is driven HIGH by ASIC output TE. The data byte at memory location 0000_h is then driven through the SN75160 GPIB transceiver out onto bus.

$$CMD\_ACC.H = PO1.H * \overline{PO0H}. \qquad (10)$$

If one examines the values of the OSM Present State Bits (Fig. 24), they will notice that gray coding (only one bit changing logic value at a time) was used in states TRANS (000), BUS_CMD (001), CMD_OUT (011), and CMD_ACC (010). This was done to avoid glitches on the internal ASIC signal CMD_ACC.H which may have led to glitches on ASIC outputs ACLR.L and CS.L. CMD_ACC.H is generated from a two input AND gate with inputs PO1.H and $\overline{PO0.H}$ (Fig. 6). Assume, however, that state CMD_OUT had been assigned the value 010 and that state CMD_ACC had been assigned the value 011 in accordance with normal binary counting. Then CMD_ACC.H would have then been equal to PO0.H*PO1.H. In that case, the logic for CMD_ACC.H would be exactly the same as presently implemented except for the inverter on PO0.H input. As the OSM moved from state BUS_CMD (001) to state CMD_OUT

*IEEE Standard Digital Interface for Programmable Instrumentation, IEEE Std 488-1978, p. 59, section 5.2.

(010), PO0.H would have switched from logic ONE to logic ZERO while PO1.H switched from logic ZERO to logic ONE. This simultaneous switching would have caused the gate voltage of the two pull-down transistors in the NAND to be above V_{th} for some small period of time. With the two pull-down transistors weekly ON, the voltage at the NAND gate output may have been below Vth causing CMD_ACC.H=>T momentarily while the OSM moved between states BUS_CMD and CMD_OUT. This may have led to glitches on ACLR.L and CS.L depending on the duration of the CMD_ACC.H glitch. With the gray code implementation, only one OSM Present State Bit changes as the OSM moves through the first four states thereby avoiding this potential problem.

The avoidance of bus contention on the buffer's DATA(8:1) bus is now discussed. The buffer's DATA(8:1) bus (FIG. 20) connects the ASIC's DO8-DO1 pins and the SRAM eight I/O pins with the buffer side of the SN75160 GPIB transceiver. During the *data input phase*, the talker sends device dependent data over the DIO lines to the buffer's SN75160 GPIB transceiver which in turn drives the data onto the buffer's DATA(8:1) bus where each transmitted byte is received by the SRAM and stored. Since the OSM idles in state TRANS during the *data input phase*, the ASIC's DO8-DO1 pads drivers are in HIGH impedance (G.H=F) and bus contention is avoided.

When the buffer makes the transition from active listener to bus controller, ASIC output CS.L goes FALSE due to DC.H going FALSE. CS.L going FALSE results in the SRAM I/O pins tristating. At the same time, the direction in which data passes through the SN75160 GPIB transceiver reverses due to TE.H=>T. The ASIC's DO8-DO1 pads drivers aren't enabled for approximately 94µs after TE.H goes HIGH. Therefore, the buffer's DATA(8:1) bus is not driven in state TRANS during the *data output phase* since output drivers on both the SRAM and ASIC are tristated. The DIO lines appear to be driven HIGH however since the IEEE-488 bus side of the SN75160 transceiver is designed to float at ≈ 3.3 volts when not driven.

The ASIC's DO8-DO1 pad drivers are enabled (G.H=>T) once the OSM enters state BUS_CMD. The memory chip output drivers must be tristated while the buffer is driving the inverted intended receiver's listen address message or UNL message on the bus to avoid bus contention with the ASIC DO8-DO1 pad drivers. The memory chip I/O pins are tristated in states TRANS, BUS_CMD, and CMD_OUT since CS.L is FALSE in these states.

On acceptance of the command byte, the buffer enters state CMD_ACC where G.H is forced FALSE. At the same time, OSM outputs PO0.H=>F and PO1.H=>T causing CS.L=>T. This results in a possible race between the ASIC's DO8-DO1 pad drivers turning OFF and the SRAM drivers turning ON. G.H is run through a superbuffer to the TRI_CTL port on the eight ASIC DO pads. The logic which generates CS.L experiences four gate delays before arriving at the OFF_CHIP port on the CS.L pad. This pad must then pull the SRAM's CE pin LOW. CE going LOW causes the memory chip to exit reduced power standby mode to enter active mode. The memory address counters are also zeroed (ACLR.L=>T) in state CMD_ACC. Therefore, the address inputs to the SRAM may also be changing as the device enters active mode. The author isn't sure if the ASIC's DO pads will have tristated by the time the SRAM's I/O pins begin to drive the DATA(8:1) bus. To determine this would require knowledge of interconnect parasitics, external delay between the ASIC's CS.L pin and the SRAM's CE.L pin,

and the speed of the SRAM used. However, it doesn't really doesn't matter if the drivers for both devices are ON for a small period of time since the drivers of the memory chip and those of the ASIC can survive a short tug of war.

The potential for bus contention also occurs when the OSM re-enters state BUS_CMD after file transfer is complete. This time state TRANS is skipped so CS.L goes FALSE at the same time that G.H goes TRUE. Again, since the drivers of the SRAM and ASIC's DO pads can survive a short tug of war, the only concern is the amount of impact on the UNL command settle time as discussed in the section 8.8.3.

Once devices on the bus are again ready to receive another command byte (NDACO.L=T and NRFDO.L=F), the status of test input MUX.L is checked. The logic level of MUX.L represents which message (an address to listen or UNL) is passed through the Command Multiplexer. The **first** time through state CMD_ACC, MUX.L will be TRUE since a listener was just addressed and the OSM will move to state DAT_OUT. The OSM passes through state CMD_ACC a **second** time after file transfer is complete. This time however, the test input MUX.L is FALSE. Note from its ASM Chart that the OSM remains in state CMD_ACC when MUX.L is FALSE. The expression CMD_ACC.H * NDACO.L * $\overline{\text{NRFDO.L}}$ * $\overline{\text{MUX.L}}$ (equation 2) is now TRUE which causes the SMC outputs to switch, DC.H=>T, TE.H=>F. This results in the buffer resuming the role of active listener. TE.H going FALSE also causes the Delay Counter to immediately reset its output - HOLD.H=>F. HOLD.H going FALSE forces the OSM into state TRANS where it idles during the *data input phase.*

8.8.5 State DATA OUT (DAT_OUT)

As the buffer leaves state CMD_ACC and enters state DAT_OUT, ASIC outputs ACLR.L and ATN.L are both deactivated. ACLR.L going FALSE (due to CMD_ACC.H=>F) enables counting by the memory address counters. ATN going FALSE causes all instruments which weren't instructed to listen to no longer hold there NDAC bus drivers TRUE. The NDAC line will remain TRUE however, due to the addressed listener driving it LOW.

The OSM output DAVO.L is asserted in state DAT_OUT. With the deactivation of ATN and the assertion of DAV, the buffer assumes the role of active talker as it leaves state CMD_ACC and enters state DAT_OUT. In this capacity, the buffer transfers the stored data file a byte at a time to the addressed listener by cycling through OSM states DAT_OUT and DAT_ACC. Unaddressed devices ignore device dependent data passed from the active talker to the addressed listener.

DAV going TRUE informs the addressed listener that a stable, device dependent data byte is present on the bus's DIO lines. The addressed listener responses to DAV going LOW by first driving it NRFD driver LOW and then when ready, by setting its NDAC driver HIGH. NDAC.L going FALSE increments the present memory address count. For example, when the buffer first enters state DAT_OUT the memory address count equals 0000_h. By driving NDAC HIGH the addressed listener accepts the first data byte at memory address 0000_h. NDAC.L going FALSE increments the memory address count to 0001_h thus addressing the second byte

in the stored data file. Every time the count is incremented, the two comparators compare the present memory address with the *final count value* stored in the two SN74ALS573 latches. When the two addresses are equal, the P=Q outputs of the two comparators go HIGH causing the NAND gate they drive to force ASIC pin PQ.L LOW. Since this signal is asynchronous to the ASIC internal clock, it is run through a synchronizer cell. An inverter at the output of the synchronizer (use to create the correct voltage level) sources the internal signal PQ.H. This signal is routed to the PQ.H inputs of the OSM and ISM. [PQ.H is not acted upon by the ISM during the *data output phase* since it is held in state PON.]

Each time state DAT_OUT is entered, the test input PQ.H is examined to check if the memory location addressed is that which contains the last data byte. If not, the buffer waits for the addressed listener to accept the data byte (NDACO.L=F) and then moves to state DAT_ACC. When the memory address count ***does*** equals the *final count value*, the OSM asserts EOIO.L with the transfer of the last data byte in state DAT_OUT. EOIO.L is a conditional output of state DAT_OUT. It is only asserted when PQ.H=T in state DAT_OUT. EOIO.L is used to drive the bus signal EOI during the *data output phase*. EOI LOW informs the addressed listener that it is receiving the last byte of the current data transmission. EOIO.L is also tied directly to an input on the Command Multiplexer. EOIO.L=T loads the safe semistatic register in Command Multiplexer with a logic ONE. This results in MUX.L=>F after a ψ1-ψ2 clock cycle and the UNL command being passed through the multiplexer to its CMD8-CMD1 outputs. The OSM once again waits for its NDACO.L input to go FALSE and then enters state DAT_ACC at the next rising edge of the ψ2 clock.

A potential problem may exist as the buffer moves from state CMD_ACC to DAT_OUT. ATNO.L goes FALSE at the same time (rising ψ2 clock edge) that DAVO.L goes TRUE. Both these signals are routed to the OFF_CHIP port of there respective pad - ATN.L or DAV.L. The ATNO.L signal will experience slightly more delay than the DAVO.L signal due to the longer path it takes in the chip layout to the ATN.L pad. These signals continue onto there respective pins on the SN75161 GPIB transceiver and then out onto the bus. If the DAV bus line happens to go LOW before the ATN line goes HIGH, there is a possibility that devices on the bus may interrupt the multiline message as another command byte rather than the first device dependent data byte met for the addressed listener. Judging from the AH (Acceptor Handshake) Interface Function State Diagram,* the author believes that the unaddressed devices may initially be involved in accepting the device dependent data byte as a command. However, once an unaddressed device notices ATN is FALSE, it will drop out of the handshaking process. In order to accept device dependent data, a device must be in the listener active state (LACS)**. An instrument will enter this state within 200ns once ATN goes FALSE after being addressed to listen. Devices not addressed to listen, will enter acceptor idle state (AIDS) once ATN goes FALSE. See section 9 titled Recommendations for Future Designs for further details on forcing ATNO.L FALSE in state CMD_ACC.

The buffer's ADDRESS (15:0) bus ties the output of the memory address counters to the memory chip address pins, to the P inputs on the two octal comparators, and to the inputs of

*See IEEE Std. 488-1978, page 23.

**See IEEE Std. 488-1978, page 31.

the two octal latches (Fig. 20). [During the *data output phase* inputs to the two octal latches are disabled.] When the addressed listener accepts the data byte on the DIO lines, it deactivates NDAC. NDAC.L=>F will cause the memory address count to increment. Maximum delay through the SN75161 transceiver and the SN74ALS561 counters is 20ns and 18ns, respectively. The maximum time between a stable address appearing at the address pin of the 150ns SRAM used by the author and valid output at its I/O pins is 150ns. Therefore, the **next** data byte is driven onto the DIO lines a maximum of 188ns after NDAC goes FALSE. Depending on when the asynchronous input NDAC makes its transition, its synchronized equivalent, NDACO.L, follows anywhere from $^1/_4$ to $1^1/_4$ clock cycles later - (125ns to 625ns at 2MHz). The buffer doesn't make the transition to state DAT_ACC until an additional clock cycle after NDACO.L=>F 500ns at 2MHz. Therefore, the buffer remains in state DAT_OUT between $1^1/_4$ and $2^1/_4$ clock cycles after NDAC goes HIGH (625ns to 1125ns at 2MHz). There isn't a problem with the next data byte being driven onto the bus while the buffer is still in state DAT_OUT since the addressed listener no longer samples the DIO lines after deactivating NDAC. Hence, the new data byte has plenty of time to stabilize before it is sampled by the addressed listener when state DAT_OUT is again entered.

When the addressed listener accepts the *second to last data byte*, NDAC going HIGH causes the following chain of events. NDAC.L=>F clocks the memory address counters causing the count to increment. This results in a memory address equal to the *final count value* to appear at the output of the SN74ALS561 counters, at the P inputs of the comparators, and at the address inputs of the SRAM. Since the comparators' P inputs now equal there Q inputs, there P=Q outputs will go TRUE (HIGH). The two P=Q comparator outputs are connected to the two input NAND gate which sources PQ.L. Two highs at the input to the NAND forces its PQ.L output TRUE. The maximum delay through the comparators and NAND gate is 33ns and 8ns, respectively. Therefore, the worst case delay between the bus signal NDAC.L=>F and PQ.L=>T is 79ns (20ns + 18ns + 33ns + 8ns). NDACO.L follows NDAC.L going FALSE after a $\psi 1$-$\psi 2$ synchronization delay. As mentioned above, this could be anywhere from $^1/_4$ to $1^1/_4$ clock cycles following NDAC going FALSE. Therefore, PQ.H may also go TRUE at the same time that NDACO.L goes FALSE or it may go TRUE one clock cycle later depending on when the asynchronous input NDAC made its transition and on the propagation delay through the circuitry generating PQ.L. Regardless of the value of PQ.H, the OSM moves to state DAT_ACC at the next $\psi 2$ clock pulse following NDACO.L going FALSE. Therefore, the conditional output EOIO.L is prevented from going TRUE until the OSM re-enters state DAT_OUT for the transfer of the last data byte.

8.8.6 State DATA ACCEPTED (DAT_ACC)

In state DAT_ACC, the OSM drives DAVO.L FALSE. The addressed listener responses to DAV going FALSE (HIGH) by setting its NDAC bus driver TRUE (LOW). It then sets its NRFD bus driver FALSE (HIGH) when ready to receive another data byte.

The OSM idles in state DAT_ACC until the addressed listener indicates that it is ready to receive the next data byte (NDACO.L=T and NRFDO.L=F). At that point, the OSM either re-enters state DAC_OUT if MUX.L=T to continue transfer of the stored data file or it enters

state BUS_CMD if MUX.L=F. MUX.L is TRUE until the last data byte is sent. MUX.L goes FALSE one $\psi 1$-$\psi 2$ clock cycle after the assertion of EOIO.L in state DAT_OUT.

8.8.7 OSM Operation Summary

When the SMC switches it outputs, TE.H=>T and DC.H=>F, the buffer tristates the SRAM's I/O pins and begins to drive bus signals REN, IFC, ATN, EOI, and DAV. With the assertion of REN (LOW) the buffer becomes the system controller. Even though the OSM is driving ATN, EOI, and DAV, it remains in state TRANS until the Delay Counter times out. At that point, the OSM enters command mode (ATNO.L=T), enables the ASIC's DO8-DO1 pad drivers, and addresses the listener. It then tristates the ASIC's DO8-DO1 pad drivers, zeros the address count (ACLR.L=T), and enables SRAM. With the deactivation of ATN, the OSM takes on the role of active talker and sends the stored data file to the listener. When the memory address count becomes equal to the *final count value*, the OSM asserts EOIO.L with the transfer of the final byte. The assertion of EOIO.L causes the command multiplexer to pass the UNL command to its CMD8-CMD1 outputs. With the data file sent, the OSM resumes the role of bus controller (ATNO.L=>T) and again disables the external SRAM (CS.L=>F) and enables the ASIC's DO8-DO1 pad drivers (G.H=>T). It then issues the UNL (unlisten) command which unaddress any listeners. The OSM then re-tristates the ASIC's DO8-DO1 pad drivers, clears the counters, and enables SRAM. At that point, the SMC kicks in (DC.H=>T, TE.H=>F) causing the buffer to make the transition from active controller back to active listener. Since the buffer is again operating in the *data input phase*, the OSM again idles in state TRANS.

8.9 The Command Multiplexer (CMD MUX)

There are 22 multiline commands specified by the IEEE-488 Standard. The RIT IEEE-488 Buffer is capable of issuing just two of the specified commands.* The Command Multiplexer is used to select between the intended receiver's listen address command and the UNL command. The MLA (my listen address) command is used to address a device and put its bus interface into listener mode. The MLA command is referred to in this document as the intended receiver's listen address command. The UNL (unlisten) command is used to unaddress all listener on the bus.

Any device-dependent message or multiline interface command sent over the bus uses the DIO8-DIO1 lines. DIO8 is the most significant bit (MSB) and is usually a don't care (X). The UNL command in binary is X0111111 while any MLA command in binary is $X01A_5A_4A_3A_2A_1$ where A_5-A_1 represents the setable five bit address required on all IEEE-488 devices. The user must inform the buffer of the intended receiver's address by entering it via dip switches on board the buffer. The OFF side of the dip switches are tied to ASIC input pins A5.L-A1.L (Fig 5). The OFF side of each dip switch is also connected to VCC through a pull-up resistor. A dip switch in the OFF position supplies ≈ 5 volts to its corresponding ASIC pin. The ON side of the five dip switches are grounded. In the ON position, ≈ 0.0 volts is supplied to the corresponding ASIC pin. Therefore, an address dip switch is set FALSE by placing it in the ON position (pull-up is pulled LOW through dip switch) and set TRUE by placing it in the OFF position.

The Command Multiplexer (Fig. 26) consist of a two input, resetable, safe semistatic register stage and an octal 2-to-1 multiplexer. The octal multiplexer is really a five bit inverting multiplexer with the remaining three bits hard wired. ASIC input pads A5.L-A1.L are connected to the Command Multiplexer's ADD5-ADD1 inputs. The GPIB bus uses negative logic (LOW voltage equals True). Hence, the UNL and intended receiver's listener address commands must be complemented to yield 11000000 (UNL) and $110\overline{A_5}\,\overline{A_4}\,\overline{A_3}\,\overline{A_2}\,\overline{A_1}$ where $\overline{A_\#}$ represents the complemented address bit voltage. Since the upper three bits are the same, there is no need to select between them so these bits are hard wired to the complementary voltage. The two MSB's are tied HIGH while the third MSB is grounded. When the register stage is loaded with a logic ZERO, the multiplexer's ADD5-ADD1 inputs are inverted and passed to its CMD5-CMD1 outputs. Since the UNL command doesn't change, its lower five bits are also hard wired internally within the ASIC. When the register stage is loaded with a logic ONE, the five lower bits of the UNL command are inverted and passed to the CMD5-CMD1 outputs. The CMD8-CMD1 outputs are routed to the OFF_CHIP port on ASIC pads DO8-DO1. Note: Nets CMD8-CMD6 are shown in the ASIC schematic but are

---

* The RIT IEEE-488 Buffer is **not** capable of recognizing or responding to any bus commands. The buffer is designed for use in a controllerless system - one dedicated talker and one or more dedicated listeners. In a controllerless system the talker sends device dependent data to the dedicated listener(s). Only a controller is capable of issuing bus commands. Since there is no controller present in such a system, there is no need for the buffer to possess the capability of responding to bus commands. Therefore, the INTO_CHIP ports on ASIC pads IFC.L, REN.L, and ATN.L are left unconnected internally.

actually implemented in the chip layout by tying the OFF_CHIP port HIGH on pads DO7 and DO8 and by connecting the OFF_CHIP port on pad DO6 to GND.

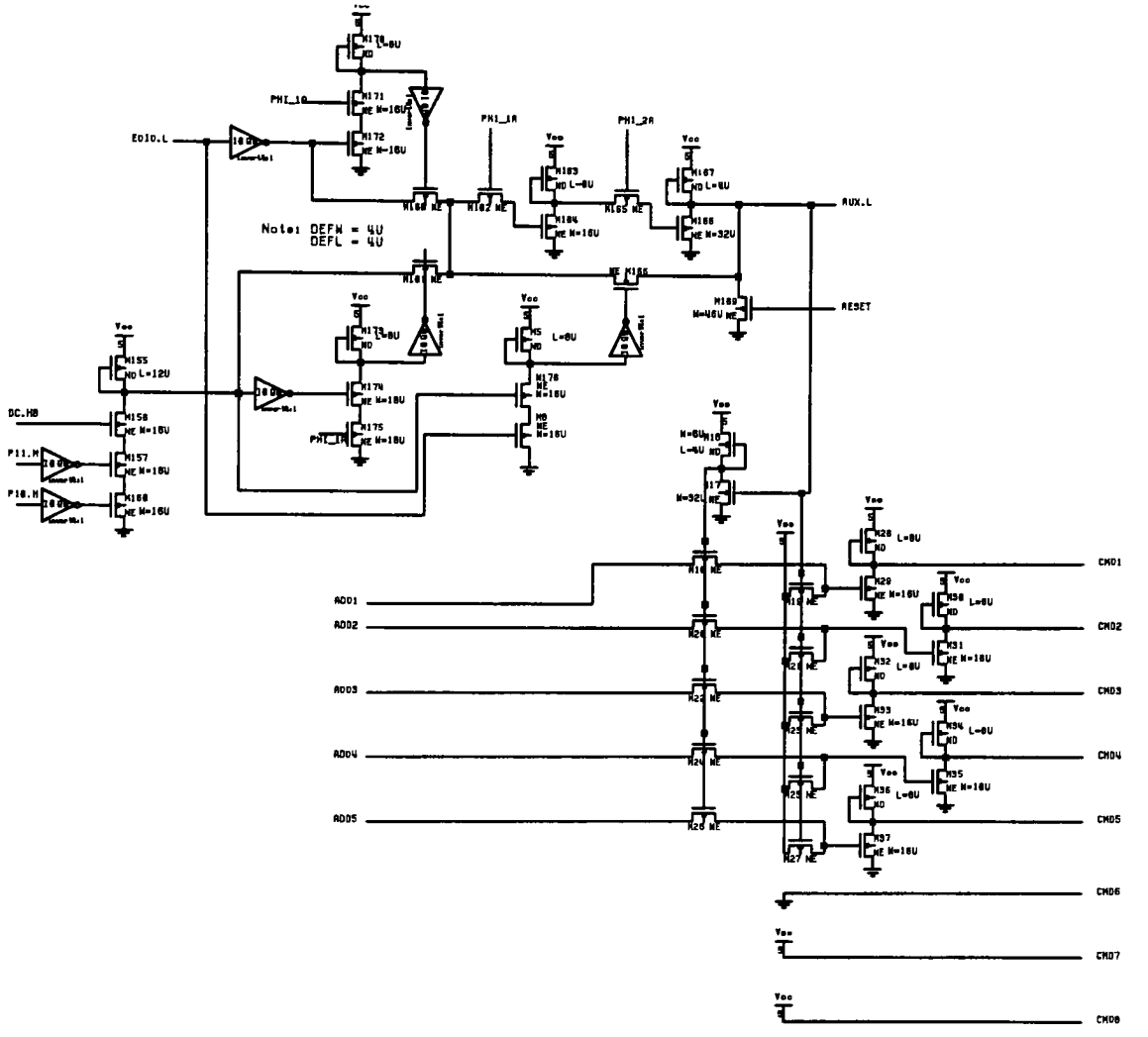


Figure 26 Command Multiplexer Schematic

The Command Multiplexer output MUX.L (equation 11) is used as an input to the pass transistors gating the lower five bits of the UNL command and to the inverter driving the pass transistors which gate the ADD5-ADD1 inputs. MUX.L also serves as an input to the SMC and OSM. MUX.L indicates which command is present at the multiplexer's CMD8-CMD1 outputs. When the buffer is powered-on or reset (RESET=T), pull-down transistor M169 is turned ON. This forces MUX.L TRUE and causes a logic ZERO to be loaded into the register stage. When MUX.L is TRUE (LOW), the inverted intended receiver's listen address message is passed to the CMD8-CMD1 outputs. After RESET goes FALSE, the ISM is active (DC.H=T) and the buffer waits in state PON (PI0.H=T, PI1.H=T) for the talker to begin transmitting data. Equation 11 is satisfied whenever the ISM is in state PON while DC.H is

TRUE. Hence, a logic ZERO is again loaded into the semistatic register stage but this time it enters through pass transistor M161.* This has no affect on the output MUX.L however, since it is already TRUE. As the buffer accepts data from the talker, it cycles through ISM states PON, RDY, and DAV. Each time the buffer passes through state PON the register stage is again loaded with a logic ZERO. Again, the output MUX.L is unaffected.

$$MUX.L = DC.H * \overline{PI1.H} * \overline{PI0.H} + RESET \qquad (11)$$

The SMC outputs switch (DC.H=>F, TE.H=>T) after the buffer accepts the last data byte from the talker. This results in equation 11 going FALSE since DC.H is no longer TRUE. When equation 11, EOIO.L, and RESET are all FALSE, the semistatic register stage refreshes itself by feeding back its output through pass transistor M166 to the register stage input.

The OSM output G.H is tied to the TRI_CTL port on ASIC pads DO8-DO1. The drivers of these pads are always tristated except in states BUS_CMD and CMD_OUT (G.H=T) when the buffer is driving a command out onto the bus. This is necessary to prevent bus contention on the buffer's DATA(8:1) bus. The buffer enters command mode (ATNO.L=T) twice during each file transfer. The first time occurs as OSM state BUS_CMD is entered after the buffer makes the transition from active listener to bus controller. This results in the inverted intended receiver's listen address command being driven out onto the DIO lines during states BUS_CMD and CMD_OUT. When all devices have accepted the command, OSM state CMD_ACC is entered and the DO8-DO1 pad drivers become tristated (G.H=>F) once again.

The OSM cycles through states DAT_OUT and DAT_ACC while transferring the stored data file to the addressed listener. Each time state DAT_ACC is entered, the OSM checks the status of MUX.L. If MUX.L is TRUE, the OSM re-enters state DAT_OUT. The OSM asserts EOIO.L in state DAT_OUT with the transfer of the last data byte. EOIO.L going TRUE causes a logic ONE to be loaded into the Command Multiplexer's register stage through pass transistor M160 during the following ψ1 clock pulse. The logic ONE forces MUX.L FALSE at the following ψ2 clock pulse. MUX.L going FALSE causes the UNL command to be passed to the Command Multiplexer's CMD8-CMD1 outputs. After the addressed listener has accepted the last data byte, the status of MUX.L is checked in state DAT_ACC. This time MUX.L is FALSE and the OSM enters state BUS_CMD where it assumes the role of active controller for the second time. With MUX.L=F, the UNL command is driven out onto the DIO lines during states BUS_CMD and CMD_OUT. When all instruments have accepted the UNL command, the OSM enters state CMD_ACC. The OSM will remain in state CMD_ACC even after NDACO.L=>T and NRFDO.L=>F since MUX.L is now FALSE. This condition causes the SMC outputs to switch (DC.H=>T, TE.H=>F) at the next ψ2 clock pulse. As the buffer makes the transition from bus controller back to active listener, equation 11 becomes TRUE since the ISM was idling in state PON while the OSM was active. This results in the logic ONE stored in the multiplexer's register stage to be replaced by a logic ZERO. The logic ZERO causes MUX.L=>T at the following ψ2 clock pulse thus allowing the intended receiver's listen address command to pass through to the multiplexer's CMD8-CMD1 outputs.

* The register stage is loaded with a logic ZERO during *each* ψ1 clock pulse while the ISM sits in state PON and DC.H is TRUE.

8.10 A Detailed Look at the Transition from Bus Controller to Active Listener

If MUX.L is FALSE in OSM state DAT_ACC, state BUS_CMD is entered. This results in the buffer leaving talker mode and resuming the active controller role since ATNO.L is set TRUE in state BUS_CMD. SRAM is also tristated (CS.L=F) in state BUS_CMD and the ASIC's DO pad drivers are enabled (G.H=T) causing the $\overline{\text{UNL}}$ command to be driven onto the IEEE-488 bus. All devices on the bus respond to ATN going TRUE (LOW) by setting there respective NDAC bus driver TRUE (LOW). The OSM remains in state BUS_CMD for only one clock period since the addressed listener has driven NDACO.L TRUE and NRFDO.L FALSE during the previous clock cycle. Therefore, the conditions for moving to state CMD_OUT are satisfied. In state CMD_OUT, the OSM asserts DAVO.L to inform devices that a valid, stable command byte is present on the DIO lines. Devices respond to DAV going TRUE by asserting there NRFD driver followed by deactivating there NDAC driver to indicate acceptance of the command byte. After NDACO.L=>F, the OSM enters state CMD_ACC during the next $\psi 2$ clock pulse. In state CMD_ACC the buffer deactivates G.H and DAVO.L. G.H going FALSE tristates the DO8-DO1 pad drivers. The SRAM also becomes enabled (CS.L=>T) in state CMD_ACC since the internal ASIC signal CMD_ACC.H goes TRUE. CMD_ACC.H going TRUE also forces ACLR.L TRUE. ACLR.L=>T resets the memory address counters in preparation for receiving the next file transfer once the buffer re-enters listener mode. Devices respond to DAV going FALSE by asserting there NDAC driver followed by deactivating there NRFD driver when ready to receive another command byte. The OSM remains in state CMD_ACC even after NDACO.L=>T and NRFDO.L=>F since MUX.L is now FALSE. This condition satisfies equation 2 which results in the SMC outputs switching (DC.H=>T, TE.H=>F) during the next $\psi 2$ clock pulse.

The SMC's outputs switching causes the buffer to make the transition from active controller to active listener. The transition also defines the end of the *data output phase* (DC.H=F, TE.H=T) and the beginning of the *data input phase* (DC.H=T, TE.H=F). DC.H=>T loads the Command Multiplexer with a logic ZERO which results in MUX.L=>T one clock cycle later. DC.H=>T also enables the drivers on ASIC pads NDAC.L and NRFD.L. TE.H=>F immediately resets the Delay Counter (HOLD.H=>T) which in turn immediately forces the OSM into state TRANS. The OSM idles in state TRANS during the *data input phase*. Since the internal ASIC signal CMD_ACC.H is not TRUE in state TRANS, ACLR.L goes FALSE thus enabling the memory address counters. HOLD.H=>T also forces OSM inputs NRFDO.L and NDACO.L to ground by turning ON pull-down transistors M600 and M601. TE.H=>F disables the drivers on ASIC pads REN.L, IFC.L, ATN.L, DAV.L, and EOI.L along with reversing the direction in which signals travel through the GPIB transceivers. The buffer has now completed the transition back to an active listener where it idles in ISM state PON until the talker again commences communication.

8.11 ASIC Simulation

In this section the analog simulation results for the ASIC are displayed and discussed. A capacitor was placed on the output of several ASIC pins to emulate the load that these pins are required to drive. The author assumed a load of 10pF per pin that is driven. This value should be large enough to account for the pin capacitance plus any trace/wire capacitance. The NDAC.L pin drives the largest load (50pF) during the *data input phase*. This pin is connected to the NDAC pin on the SN75161 GPIB transceiver and to the clock pin on the four SN74ALS561 counters. The ACLR.L pin drives the largest load (40pF) during the *data output phase*. This pin is tied to the ACLR pin on the four SN74ALS561 counters. From looking at the simulation waveform, the NDAC.L and ACLR.L pins don't appear to have any problem driving there loads.

The first simulation (Fig. 27) displays the RIT IEEE-488 Buffer starting out in a power-on or reset condition, then accepting a three byte transmission from the talker, and finally, making the transition from active listener to bus controller. While RESET is HIGH, the ISM is held in state PON (PI0.H=F, NDACI.L=T), SRAM is enabled (CS.L=T), and ASIC output ACLR.L is forced TRUE. ACLR.L=>T clears the memory address counters which causes SRAM memory location 0000_h to be accessed. During the *data input phase*, SRAM is in write mode since its $R/\overline{W}$ pin is tied to ASIC output TE. At power-up, the contents of the SN74ALS573 latches are unknown. If the octal latches happen to contain all zeros (0000_h), then PQ.L will go TRUE since the current memory address count also equals 0000_h. This situation is represented in the simulation by PQ.L=>T at ≈ 500ns followed by PQ.H=>T at the next $\psi 2$ clock pulse. As discussed in section 8.5.1, PQ.H=>T while EOII.L is FALSE has no affect on the buffer operation.

After RESET goes LOW, the memory address counters become enabled (ACLR.L=>F) and the ISM waits in state PON for the talker to commence communication. The talker begins communication by placing a device dependent data byte on the DIO lines and asserting (driving LOW) DAV. The data byte is driven through the SN75160 GPIB transceiver and onto the buffer's DATA(8:1) bus where it is received by the SRAM and stored in memory location 0000_h. The buffer responds to DAVI.L going TRUE by entering state RDY and asserting NRFDI.L. The ISM remains in state RDY for only one clock cycle before entering either state DAC or LATCH. If EOII.L is FALSE, the ISM enters state DAC and deactivates NDACI.L. NDAC.L=>F increments the memory address counters causing memory location 0001_h to be accessed. Since the present memory address count (0001_h) no longer equals the value contained in the two octal latches (0000_h), the PQ.L signal goes FALSE followed by PQ.H=>F at the next $\psi 2$ clock pulse. The talker responds to NDAC going HIGH in state DAC by deactivating DAV. DAVI.L=>F causes the ISM to re-enter state PON (at ≈ 3900ns) where NRFDI.L is set FALSE and NDACI.L is set TRUE. The ISM again waits in state PON for the talker to make available the second data byte. Upon DAVI.L=>T, the ISM again enters state RDY and asserts NRFDI.L. This time the data byte on the DIO lines is written to SRAM memory location 0001_h. After checking the status of EOII.L, the ISM enters state DAC and deactivates NDACI.L. NDAC.L=>F increments the memory address counters causing memory location 0002_h to be accessed. Once DAVI.L=>F, the ISM re-enters state PON and asserts NDACI.L and deactivates NRFDI.L.

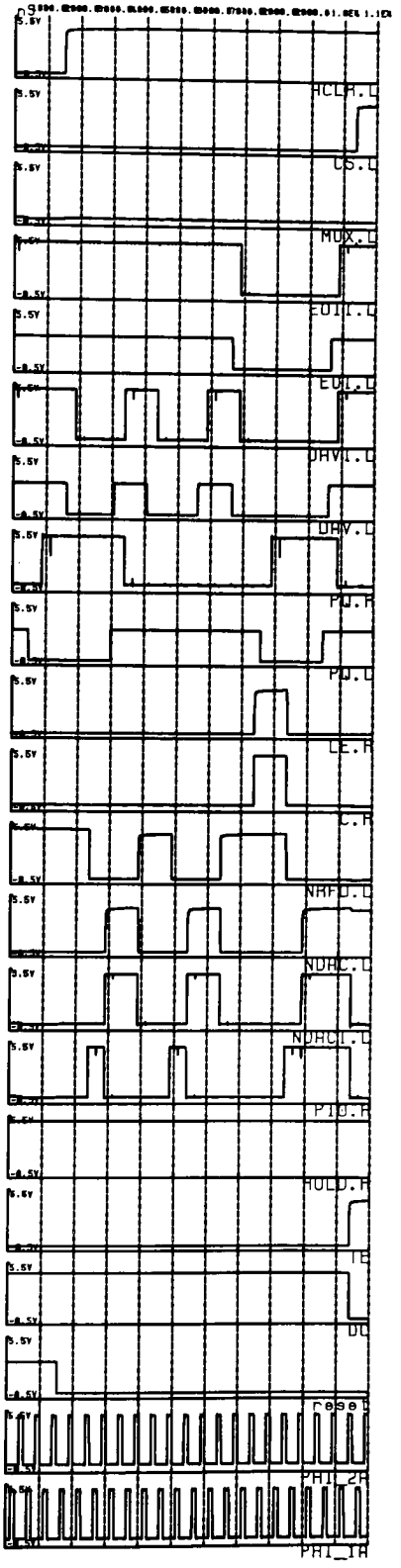


Figure 27 ASIC Simulation During the Data Input Phase

Since the third data byte is the last byte of the current transmission, the talker asserts both DAV and EOI (at ≈ 6900ns) to inform the active listener that it will be receiving the final data byte. After EOII.L=>T, the ISM asserts C.H at the next ψ2 clock pulse. ASIC output LE.H follows C.H going TRUE. LE.H going TRUE enables the inputs to the octal latches thus allowing the present memory address count (0002_h) to be passed through to the latches' outputs. This causes PQ.L=>T since the contents of the SN72ALS573 latches now equals the output of the memory address counters. The ISM will not leave state PON while EOII.L is TRUE until PQ.H=>T. At that point, C.H is set FALSE thus disabling the octal latches' inputs and capturing the *final count value*. The ISM also enters state RDY provided DAVI.L=T. The final data byte is written to memory location 0002_h. Since EOII.L is now TRUE, the ISM moves from state RDY to state LATCH during the next ψ2 clock pulse. In state LATCH, the ISM deactivates NDACI.L thus accepting the last data byte. NDAC.L=>F increments the memory address count to 0003_h. Since the inputs to the SN74ALS573 latches are disabled (LE.H=F), the *final count value* (0002_h) is not corrupted. The talker responds to NDAC going FALSE by deactivating DAV and EOI. The ISM remains in state LATCH even after DAVI.L and EOII.L go FALSE. This condition results in the SMC outputs switching (DC.H=>F, TE.H=>T) during the following ψ2 clock pulse. DC.H=>F causes the SRAM to enter reduced power standby mode (CS.L=>F) and disables the drivers on ASIC pads NDAC.L and NRFD.L. TE.H=>T immediately forces the ISM into state PON (PI0.H=F, NDACI.L=T). The NDAC.L waveform remains HIGH however due to the 50pF load capacitor attached to it. This capacitor represents the 5 pin load that the NDAC.L pad must drive during the *data input phase*. TE.H=>T also causes the Delay Counter to start down counting, enables the drivers on ASIC pads REN.L, IFC.L, ATN.L, DAV.L, and EOI.L, and changes the direction of signals passing through the GPIB transceivers. At this point, the buffer has completed the transition from active listener to bus controller.

The second simulation (Fig. 28) shows the RIT IEEE-488 Buffer entering command mode, issuing the intended receiver's listen address command, and finally, transferring the first device dependent byte to the addressed listener. The second simulation displays some overlap with the previous simulation. It begins with the buffer still in the *data input phase* (DC.H=T, TE.H=F). The ISM is in state LATCH and it is assumed (waveforms not shown) that DAV.L and EOI.L have just gone FALSE. This causes DAVI.L (waveform not shown) and EOII.L (waveform shown) to go FALSE at the following ψ2 clock pulse. EOII.L and DAVI.L going FALSE in state LATCH satisfy the conditions necessary to make the buffer transition from an active listener to bus controller as demonstrated by DC.H=>F and TE.H=>T at the following ψ2 clock pulse. There may be noise on signals which pass through the GPIB transceivers immediately after the SMC's outputs switch; due to the change in direction in which these signals travel through the transceivers, and due to the tristating/enabling of there respective ASIC pads. Therefore, the logic levels of these signals may be unknown until voltages on these lines stabilize. To represent these unknown logic levels in the simulation, ASIC inputs NRFD.L and NDAC.L are driven at 1.3 volts right after the SMC's outputs switch (at ≈ 1000ns). After a while, the voltage on these two lines will stabilize HIGH since no device will be driving these signal lines LOW - no active listener is present on the bus immediately after the transition from active listener to bus controller.

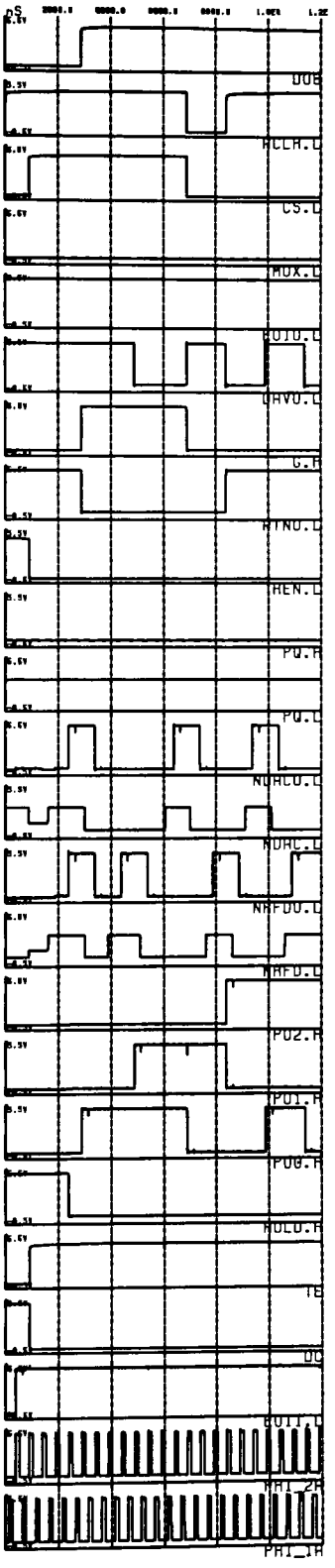


Figure 28 First Half of ASIC Simulation During the Data Output Phase

While HOLD.H is TRUE, the OSM is held in state TRANS and its NDACO.L and NRFDO.L inputs are forced to ground through pull-down transistors M600 and M601. Therefore, OSM inputs NDACO.L and NRFDO.L don't respond to the noise at ASIC pins NDAC.L and NRFD.L and remain LOW even after the NDAC.L and NRFD.L settle HIGH. The actual Delay Counter in the IEEE-488 Buffer Controller waits ≈ 94μs before deactivating HOLD.H after TE.H=>T. Not wanting to wait that long, a module called Delay Test was used to shorten the delay time to just three clock cycles in the simulation. The OSM remains in state TRANS for an additional clock cycle after HOLD.H=>F (at a ψ2 clock edge,) since inputs to the OSM are clocked in during the ψ1 clock pulse. Once HOLD.H=>F, pull-down transistors M600 and M601 are turned OFF and the output of the two synchronizers sourcing NDACO.L and NRFDO.L are no longer pulled LOW thus allowing these signals to be driven HIGH. At the next ψ1 clock edge, the OSM inputs are clocked in and the OSM moves from state TRANS to state BUS_CMD at the following ψ2 clock edge. In state BUS_CMD, the OSM asserts G.H and ATNO.L. G.H=>T enables the drivers on ASIC pads DO8-DO1 thus allowing the inverted (bus uses negative logic) intended receiver's listen address command to be driven out onto the bus. One can notice the DO8 waveform making the LOW to HIGH transition in response to G.H=>T. Instruments are required to respond to ATN going TRUE (LOW) within 200ns. Instruments not ready to receive a interface command respond by asserting both there NDAC and NRFD drivers. This scenario is represented in the simulation by NDAC.L and NRFD.L both going TRUE at ≈ 3000ns. When prepared to receive the bus command, the instrument deactivates (≈ 3900ns) its NRFD driver. The OSM remains in state BUS_CMD until all instruments are ready to receive the multiline interface command (NDACO.L=T and NRFDO.L=F) and then enters state CMD_OUT. In state CMD_OUT, the OSM asserts DAVO.L to inform devices on the bus that a valid, stable command is present on the DIO lines. Devices respond to DAV going TRUE by asserting there NRFD driver and after sampling the command byte, by deactivating there NDAC driver. Once NDAC.L=>F, the memory address counters are incremented (to 0004_h in this example unless the count was affected by the noise on the NDAC.L line). This has no affect on PQ.L, it remains LOW since the *final count value* (0002_h in this example) is not the same as the present memory address count. After NDACO.L=>F, the OSM enters state CMD_ACC at the next ψ2 clock edge and deactivates both DAVO.L and G.H. G.H=>F tristates the drivers on ASIC pads DO8-DO1. The DO8 waveform remains HIGH in the simulation however due to the 10pF load capacitor attached to the DO8 net. The capacitor is used to represent the pin on the SN75160 GPIB transceiver that ASIC pin DO8 must drive. Devices react to DAV going FALSE by asserting there NDAC driver and when ready to receive another command byte, by deactivating there NRFD driver. In state CMD_ACC, the internal ASIC signal CMD_ACC.H is TRUE. This signal is used as an input to the logic which generates ACLR.L and CS.L. While CMD_ACC.H is TRUE both ACLR.L and CS.L. are asserted. CS.L=>T enables the SRAM's eight data I/O drivers. The SRAM is now in read mode since its R/W pin is tied to ASIC output pin TE. ACLR.L=>T, causes the memory address counters to clear. In this example, the address counters outputs go from 0004_h to 0000_h This causes SRAM memory location 0000_h to be accessed - the location of the first data byte in the stored data file. Since the SRAM is now active, the byte at memory location 0000h is driven out onto the bus. Once NDACO.L=>T and NRFDO.L=>F, the OSM moves from state CMD_OUT to state DAT_OUT since MUX.L is TRUE.

The OSM deactivates ATNO.L and asserts DAVO.L as state DAT_OUT is entered. The internal ASIC signal CMD_ACC also goes FALSE as state DAT_OUT is entered. This causes ASIC output ACLR.L=>F thus enabling the memory address counters. DAV going TRUE informs the addressed listener that a valid, stable device dependent data byte is present on the DIO lines. Devices not addressed ignore bus activity while ATN is FALSE (HIGH). The addressed listener reacts to DAV going TRUE by asserting its NRFD driver and after sampling the data byte, by deactivating its NDAC driver. NDAC.L=>F increments the memory address counters to 0001_h. Once NDACO.L=>F, the OSM moves to state DAC_ACC where it deactivates DAVO.L. The addressed listener reacts to DAV going FALSE by asserting its NDAC driver and when ready to receive the next data byte, by deactivating its NRFD driver. The second simulation ends with the OSM entering state DAT_OUT in response to NDACO.L=>T and NRFDO.L=>F. DAVO.L going TRUE in state DAT_OUT informs the addressed listener that the data byte at SRAM memory location 0001_h is available on the DIO lines.

The third simulation (Fig. 29) displays the buffer completing file transfer to the addressed listener, then re-entering command mode, and finally, making the transition back to active listener. The simulation starts in OSM state DAT_OUT (DAVO.L=T) with the second to last data byte (from memory location 0001_h in this example) present on the DIO lines. The addressed listener responds to DAV being TRUE by asserting NRFD followed by deactivating NDAC. When NDAC.L=>F, the memory address count is incremented causing memory location 0002_h to be accessed and PQ.L=>T (at ≈ 2000ns) since the present memory address now equals the *final count value*. NDACO.L and PQ.H both transition at the following $\psi 2$ clock pulse. The OSM responds to NDACO.L=>F by entering state DAT_ACC during the following $\psi 2$ clock pulse and deactivating DAVO.L. The addressed listener responds to DAV going FALSE by asserting NDAC and deactivating NRFD when ready to receive the next data byte. When NRFDO.L=>F (at ≈ 3900ns), the OSM re-enters state DAT_OUT and this time asserts both DAVO.L and EOIO.L since PQ.H is now TRUE. EOIO.L=>T causes a logic ONE to be loaded into the Command Multiplexer's register stage. This results in MUX.L=>F at the next $\psi 2$ clock pulse and the UNL command being passed through to the multiplexer's outputs. The addressed listener again responds to DAV going TRUE by asserting NRFD followed by deactivating NDAC. NDAC.L=>F again causes the memory address count to increment (memory address is 0003_h accessed) which results in PQ.L=>F. Once the last data byte is accepted (NDACO.L=>F), the OSM again enters state DAT_ACC where DAVO.L and EOIO.L are both deactivated and waits for the addressed listener to indicate it is ready to receive another data byte NDACO.L=T, NRFDO.L=F. This time however, MUX.L is FALSE. MUX.L=F is a flag which informs the OSM to leave talker mode and resume the role of active controller. Therefore, the OSM enters state BUS_CMD (at ≈ 7300ns) instead of state DAT_OUT. In state BUS_CMD, the OSM asserts both ATNO.L and G.H and deactivates CS.L (since PO2.H=>F). CS.L=>F puts the SRAM in reduced power standby mode. ATNO.L=>T place the buffer in command mode. G.H=>T enables the pad drivers on ASIC pins DO8-DO1 thus allowing the UNL command (11000000_b) to be driven out onto the DIO lines. The waveform for DO8 is shown at the top of the figure. One can see this signal transitioning from LOW to HIGH in response to G.H=>T.

The IEEE-488 Standard requires that all instruments respond to ATN going TRUE within 200ns. Instruments respond to ATN going TRUE by asserting (driving LOW) there NDAC bus driver while driving there NRFD bus driver HIGH if ready to receive a command byte. If not ready to receive a command byte, instruments respond by driving both there NDAC and NRFD drivers TRUE (LOW). While the buffer was in talker mode, the only device participating in bus transactions with it was the addressed listener. Once the buffer asserts ATNO.L, it enters command mode. Every device participates in bus transactions while ATN is TRUE (LOW). The OSM moves from state DAT_ACC to state BUS_CMD because the addressed listener is ready to receive another multiline message (NDACO.L=T, NRFDO.L=F) and because MUX.L=F. When ATNO.L=>T at the $\psi 2$ clock pulse ($\approx$ 7400ns), the synchronizer's NDACO.L and NRFDO.L outputs remain TRUE and FALSE, respectively. If some instrument isn't ready to receive an interface command it will assert both its NDAC and NRFD drivers within 200ns of ATN going TRUE as stated previously.* However, the OSM will not see the NRFD line being pulled LOW since the signal is delayed by a $\psi 1$-$\psi 2$ clock cycle passing through the synchronizer. That is, during the next $\psi 1$ clock pulse, the synchronizer's input may be pulled LOW by some device not yet ready to receive a bus command (NRFD.L=T) but the synchronizer's output will be driving NRFDO.L FALSE while it is clocked into the OSM. Therefore, the OSM will not wait for devices which aren't ready. Rather, it moves immediately from state BUS_CMD to state CMD_OUT at the next $\psi 2$ clock pulse. This doesn't present a major problem however because the OSM will wait in state CMD_OUT until every instruments accepts the UNL command NDACO.L=>F. Since the NDAC line is driven by open collector drivers, this line cannot float HIGH until the slowest device stops driving its NDAC driver LOW.

As discussed above, the simulation shows the OSM remaining in state BUS_CMD for only one clock period and then entering state CMD_OUT since the addressed listener had driven NDACO.L TRUE and NRFDO.L FALSE during the previous clock cycle. In state CMD_OUT the OSM asserts DAVO.L to inform devices that a valid, stable command byte is present on the DIO lines. Devices respond to DAV going TRUE by asserting there NRFD driver followed by deactivating there NDAC driver to indicate acceptance of the command byte. After NDACO.L=>F, the OSM enter state CMD_ACC during the next $\psi 2$ clock pulse. In state CMD_ACC the buffer deactivates G.H and DAVO.L. G.H going FALSE tristates the DO8-DO1 pad drivers. The DO8 waveform remains HIGH due to the 10pF capacitor attached to the DO8 net. This capacitor represents the SN75160 GPIB transceiver pin that the DO8 line must drive. The SRAM also becomes enabled (CS.L=>T at $\approx$ 8800ns) in state CMD_ACC since the internal ASIC signal CMD_ACC.H goes TRUE. CMD_ACC.H going TRUE also forces ACLR.L TRUE. ACLR.L=>T resets the memory address counters in preparation for receiving the next file transfer once the buffer re-enters the *data input phase.* Devices respond to DAV going FALSE by asserting there NDAC driver followed by deactivating there NRFD driver when ready to receive another command byte. The OSM remains in state CMD_ACC even after NDACO.L=>T and NRFDO.L=>F since MUX.L is now FALSE. This

*This could happen if an instrument were busy. For instance, the original talker might be making another measurement. This is presumably the reason for installing the RIT IEEE-488 Buffer into the system. So the operator can get back the talker quickly for some further use while the buffer takes on the chore of driving a slow listener.

condition results in the SMC outputs switching (DC.H=>T, TE.H=>F) during the next $\psi 2$ clock pulse. DC.H=>T loads the Command Multiplexer with a logic ZERO which results in MUX.L=>T one clock cycle later. DC.H=>T also enables the drivers on ASIC pads NDAC.L and NRFD.L. TE.H=>F immediately resets the Delay Counter (HOLD.H=>T) which in turn immediately forces the OSM into state TRANS (PO0.H, PO1.H, PO2.H = 000). Since the internal ASIC signal CMD_ACC.H is not TRUE in state TRANS, ACLR.L goes FALSE thus enabling the memory address counters. HOLD.H=>T also forces OSM inputs NRFDO.L and NDACO.L to ground by turning ON pull-down transistors M600 and M601. TE.H=>F disables the drivers on ASIC pads REN.L, IFC.L, ATN.L, DAV.L, and EOI.L along with reversing the direction in which signals travel through the GPIB transceivers. HOLD.H doesn't go TRUE until the $\psi 2$ clock pulse; the OSM inputs have already been clocked in during the $\psi 1$ clock pulse. Therefore, ATNO.L remains TRUE for one more clock period. This doesn't cause any problems since the ATN.L pad driver is tristated by TE.H=>F. The buffer has now completed the transition back to an active listener where it idles in ISM state PON until the talker again commences communication.

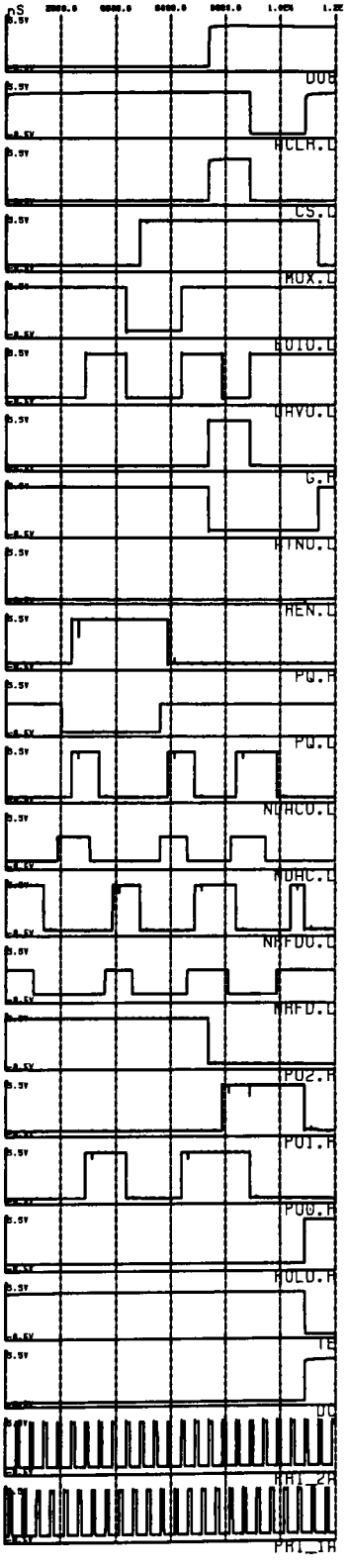


Figure 29 Second Half of ASIC Simulation During the Data Output Phase

8.12 ASIC LAYOUT

Figure 30 displays the full custom, hiearchical layout of the RIT IEEE-488 Buffer Controller. The design was done using $\lambda = 2\mu m$ MOSIS NMOS design rules. The chip is to be fabricated in the Class 1000 Clean Room at RIT by the Microelectronic Engineering Department. The author used the 38 pad ring designed by former RIT graduate student Thecla Chomicz. The Microelectronic Engineering Department had a probe card built for wafer level testing of designs using this ring.

Both the ASIC's power and ground rails are split in two. It was not possible to distribute power without splitting the rails using the chip floor plan decided on by the author. Each rail is connected to one of the ASIC's two PWR or two GND pins.

The author performed successful ERC (electrical rule checks), DRC (design rule checks), and LVS (layout versus schematic) checks on all cells and the entire layout. The ASIC was also simulated, as discussed in section 8.11, without the inclusion of parasitic resistances and capacitances. The author successfully ran LPE (layout parameter extraction) on the completed layout. LPE generates a SPICE netlist with parasitic capacitances extracted from the layout. This netlist is useful in determining delays (performance) due to routing and physical layout within the chip. The author attempted to run analog simulation with the LPE netlist but was unsuccessful. On page 3-88 in the Mentor Graphic's **AccuSim User's Manual**, they discuss how netlists created from parameter extraction can be brought in and simulated in AccuSim. They also state that regular AccuSim force commands **cannot** used with such netlists - standard SPICE stimulus commands must be added to the LPE netlist. In the Mentor Graphic's **Analog Simulators Reference Manual**, pages 5-43 through 5-47, they discuss procedures necessary for running Berkeley SPICE2 netlists in AccuSim and MSPICE. These procedures include installing subroutines provided by Mentor Graphics and the writing of three additional subroutines - MSEND, MREAD, and MCOND. The author believes the necessary subroutines discussed above are not available on the workstations in the Computer Engineering VLSI Lab at RIT. The author added standard SPICE commands to the LPE netlist which caused "Unrecognized or invalid lines errors" to occur as the netlist was read into the analog simulator. The author recommends that installation of the subroutines necessary to run SPICE netlists be investigated by the Computer Engineering Department.

As a work around to this problem, the author considered making a symbol which would incorporate the LPE netlist as a subcircuit. This way standard AccuSim forcing statements could be used to provide stimulus. This approach has two problems. Subcircuits in AccuSim require that node names be labeled **N#** where # is 1, 2, 3, . . .* Also, voltages and currents within the subcircuit are not observable - can't be traced.** Since subcircuit nets can't be observed, the author felt that this wasn't a viable approach.

---

*__AccuSim User's Manual__, Mentor Graphics Corporation, 1989, p.1-27.

**__AccuSim User's Manual__, Mentor Graphics Corporation, 1989, pp. 1-27 through 1-28.

It is doubtful that the analog simulator would have run with the LPE netlist even if the problems above didn't exist. In a non-LPE full ASIC simulation on an HP 400S workstation, the author had to replace the Delay Counter module with a module called Test Delay to reduce the node count. Test Delay contains only one toggle flip flop and a semistatic register stage. With the inclusion of the Delay Counter module, the ASIC contains approximately 1300 transistors. This resulted in matrix errors during simulation dealing with SPICE options PIVREL and PIVTOL that the Mentor hotline folks couldn't help the author over come. Simulation of the LPE netlist would only have aggravated this problem.

Therefore, the author decided to hand extract the load on certain critical paths and then perform circuit simulation to gain a feel for chip performance. One method of estimating a load is to calculate all the resistances and capacitances on a signal line and lump them into one RC network. If it is assumed that all resistances and capacitances are equal, then equation 12 results - the summation of n resistor times n capacitors where τ_{eff} is the time necessary for the signal at the load to reach 63% of the value (voltage) of the signal driving the load. However, simulating a load as a single resistor/capacitor network greatly over estimates the delay.* Long interconnect lines can be more accurately modeled as distributed RC networks**. Equation 13 provides a more accurate estimation of the signal delay along a interconnect line model as a series connection of m RC networks. As m becomes very large, equation 13 can be reduced to equation 14 where l is the length of the interconnect trace, r is the resistance per unit length, and c is the capacitance per unit length.

$$\tau_{eff} = n^2 \, RC \qquad (12)$$

$$\tau_{eff} = \frac{RC\, m\, (m+1)}{2} \qquad (13)$$

$$\tau_{eff} = \frac{rcl^2}{2} \qquad (14)$$

The equations above are difficult to use on signal lines made up of several sections of metal and poly driving distributed transistor gates. The author took a compromise approach of lumping resistance and capacitance at convenient points along a signal line into RC pairs and incorporating these loads into a schematic see figure 32 for an example. Even though this

*Amar Murkerjee, Introduction to nMOS & CMOS VLSI System Design, Englewood Cliffs, N. J. : Prentice-Hall, 1986, pp. 195-197.

**Neil Weste and Kamran Eshraghian, Principles of CMOS VLSI Design, A System Perspective, Reading, Massachusetts : Addison-Wesley Publishing Company, Inc. 1985, pp. 134-137.

approach over estimates the signal delay, it does provide some of the benefit of using distributed RC networks (a more realistic approximation of the delay) without having to generate numerous RC pairs. Inclusion of the RC pairs into a schematic also allows simulation of a load being driven by the actual driving gate.

Hand extraction of signal line's parasitics is a tedious process. An example of the hand extracted loading of the CMD MUX on the PHI_1A clock line (Fig. 31) discussed in section 8.12.1 is given below. Parasitic resistance and capacitance values used represent the worst case. The R_{mux} and C_{mux} values obtained are used to represent the loading of the CMD MUX in the schematic of the PHI_1A clock line (Fig. 32).

R_{mux} =

R_{cc}:	(100Ω/cc) * (1cc)	100Ω
R_{mtl}:	(70μm/8μm) * (0.08Ω/[])	0.7Ω
R_{ply}:	((114 + 22 + 42)μm / 4μm)[] + 0.66[]) * (100Ω/[])	4516Ω
		4616Ω

C_{mux} =

C_{pf}:	(4μm * (114+4+22+42)μm + (8μm * 8μm)) * (0.8 x 10^{-4}pF/μm^2)	63fF
C_{mf}	(70μm x 8μm) * (0.4 x 10^{-4}pF/μm^2)	22fF
C_{mp}	(4μm x (8+6+20+16+6)μm) * (0.9 x 10^{-4}pF/μm^2)	20fF
Cg:	($9C_{g[]}$) * (16fF/$C_{g[]}$)	144fF
		250fF

The author extracted and simulated the loading on the PHI_1A, PHI_2B, and TE.H lines. The simulation of each of these signal lines is discussed in separate subsections below. The author chose the clock signals to get an idea of clock skew throughout the ASIC. The PHI_1A is the more heavily loaded between the PHI_1A/PHI_2A pair so this clock line was examined with the understanding that the PHI_2A clock signal would experience less delay. Of the PHI_1B/PHI_2B pair, PHI_2B is the more heavily loaded. The author chose TE.H since this signal line is distributed throughout the chip and has the most loading of any signal line.

As discussed in section 8.2, the Clock Generator drives ASIC pads PHI1 and PHI2 (Fig. 12). The PHI1 and PHI2 signals in turn drive the superbuffers which source PHI_1A, PHI_1B, PHI_2A, and PHI_2B. When the author speaks of the ASIC's clocked logic, he means the gates actually clocked by the PHI_1A, PHI_1B, PHI_2A, and PHI_2B clock signals. Clock skew is kept to a minimum between PHASE_1 and PHI1 and PHASE_2 and PHI2 by keeping interconnects routes and load relatively identical. Near identical interconnecting routes and loads also provides minimum skew between PHI1 and PHI_1A & PHI_1B and between PHI2 and PHI_2A & PHI_2B. For example, the PHI1 signal arrives at the input to the two superbuffers sourcing PHI_1A and PHI_1B at virtually the same instance. Hence, the only skew between PHI_1A and PHI_1B should be due to the size of the load they drive the larger a load the more delay.

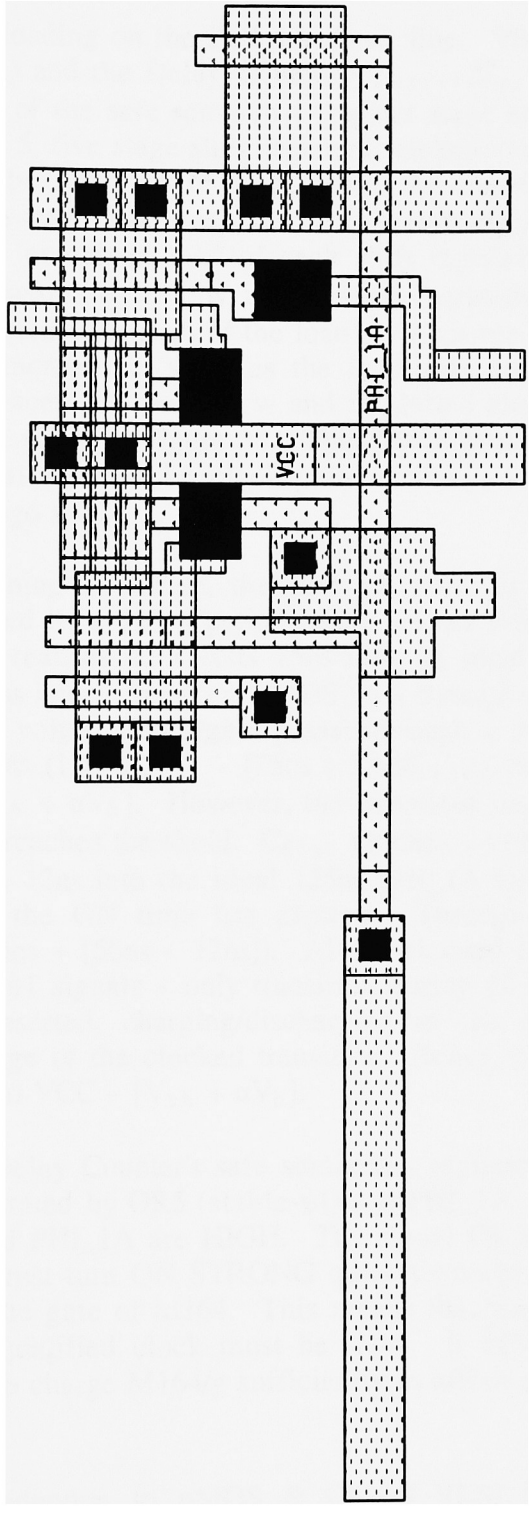


Figure 31 CMD MUX Loading on the PHI_1A Clock Line

8.12.1 PHI_1A Clock Line Loading

Figure 32 displays the loading on the PHI_1A clock line. The PHI_1A clock line drives the CMD MUX (R_{mux}/C_{mux}) and the Delay Counter. R_{wire7}/C_{wire7} and transistors M162 & M171 represent the loading of the safe semistatic register stage at the Delay Counter's output. The Delay Counter contains 5, five stage shift registers stacked vertically on top of each other. $R_{dly\#}/C_{dly\#}$ (where # can be 1, 2, 3, 4, or 5) represents the combined clocked gate loading of five individual stages - one stage from each of the five shift registers. For instance, the load R_{dly1}/C_{dly1} represents the combined load of each shift register's *first* stage. The various $R_{wire\#}/C_{wire\#}$ pairs represent the various interconnect parasitics. Figure 33 displays the resultant clock skew associated with each of the loads in response to the PHI_1A clock signal. One can notice a gradual increase in skew as the load gets further from the output of the superbuffer. C_{dly1} experiences the most skew and is plotted along with PHI1, PHI_1A, and the idealized PHI_2A clock pulse in Figure 34. This figure provides a more visually graphic view of the skew. Note that the PHI_1A clock pulse at C_{dly1} settles LOW again before the idealized PHI_2A begins to go HIGH.

When the ASIC is running at 2MHz, the PHI_1A pulse is ideally asserted HIGH for approximately 125ns followed by a 125ns period where neither PHI_1A or PHI_2A is asserted. In the simulation, PHI_1A reaches 4.95 volts 75ns into its ideal 125ns ON time and C_{dly1} rises to 4.95 volts about 23ns later. Since an NMOS pass transistor can't pass a HIGH without degrading it*, a HIGH ($\approx$ 5 volts) input signal passed through a transistor clocked by PHI_1A at C_{dly1} will only have 27ns ($125ns_{ideal}$ [75ns + 23ns]$_{skew}$), in the worst case, to charge a clocked node to VCC - [V_{th} + αV_b]. However, the transistor begins to conduct significantly as soon as its gate voltage reaches threshold. C_{dly1} reaches 1 volt, the approximate threshold voltage of the clocked gate, 12ns into the ideal 125ns PHI_1A ON time and doesn't fall back to 1 volt until 50ns *after* the ON time has expired. Therefore, the clocked transistor is actually ON for 163ns (125ns + [50ns 12ns]). All ψ1 clocked inputs to the Delay Counter and CMD MUX are stable-ψ1 signals - only transition during ψ2. Since the input voltage is valid before PHI_1A is asserted, charging/discharging of the clocked input node is only dependent on the gate voltage of the clocked transistor. Hence, clocked input nodes shouldn't have any problem charging to VCC [V_{th} + αV_b].

The input gate to the Delay Counter's safe semistatic register stage (M160) is controlled by a qualified clock** generated by OR5 (stable-ψ1) and PHI_1A. The qualified clock is only asserted when both OR5 and PHI_1A are HIGH. The signal OR5 is also used as the input to the register stage. M160 must turn ON STRONG quickly enough for OR5 to pass through it and M162 to fully charge the gate of M164. This means that the delay through NAND gate and inverter sourcing the qualified clock must be short. If M160 doesn't turn ON quickly enough, OR5 won't be able to charge M164/g sufficiently to effect a change at the Delay

---

*Amar Murkerjee, Introduction to nMOS & CMOS VLSI System Design, Englewood Cliffs, N. J. : Prentice-Hall, 1986, pp. 16-18.

**Amar Murkerjee, Introduction to nMOS & CMOS VLSI System Design, Englewood Cliffs, N. J. : Prentice-Hall, 1986, pp. 92-93.

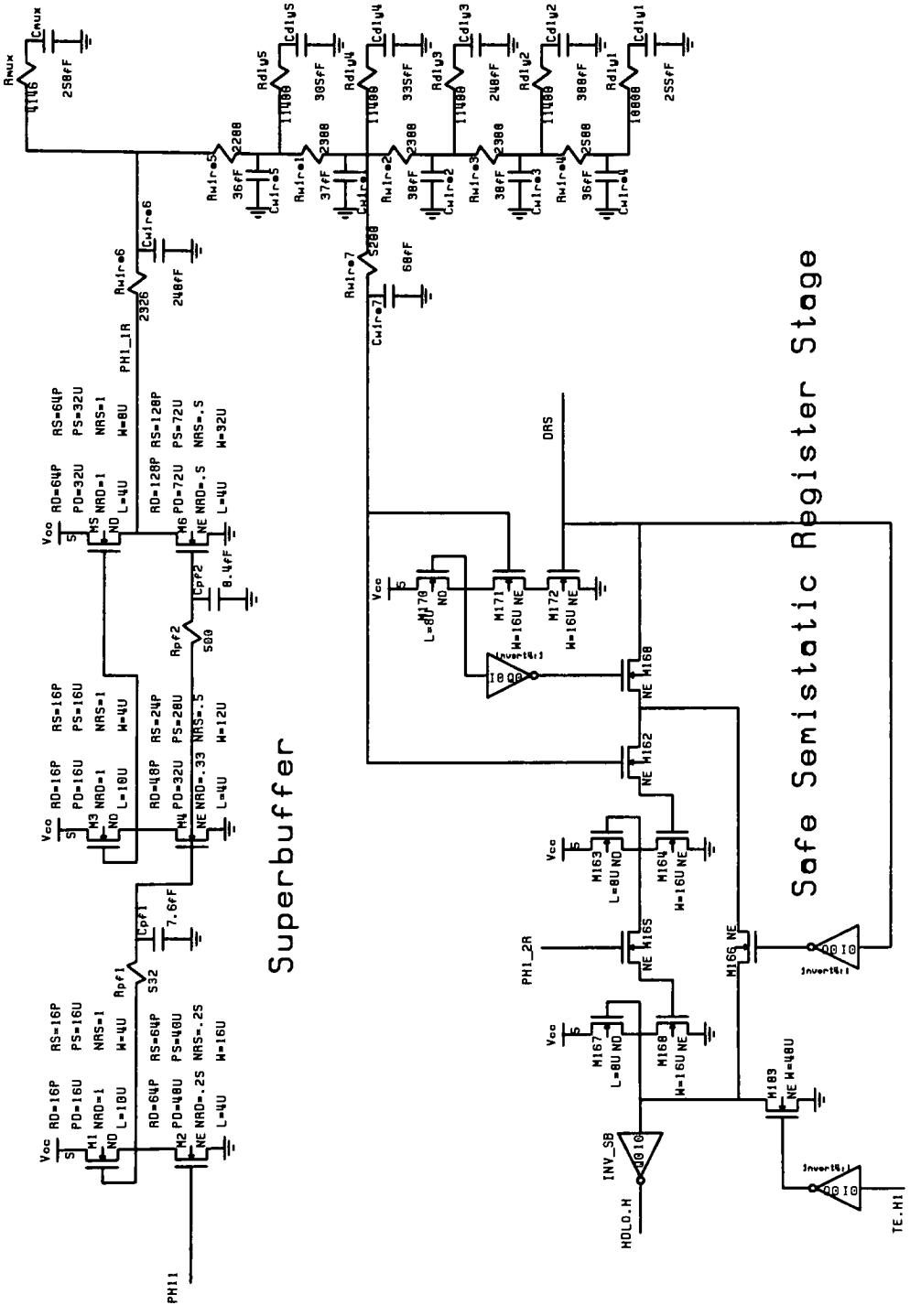


Figure 32 PHI_1A Clock Line Loading

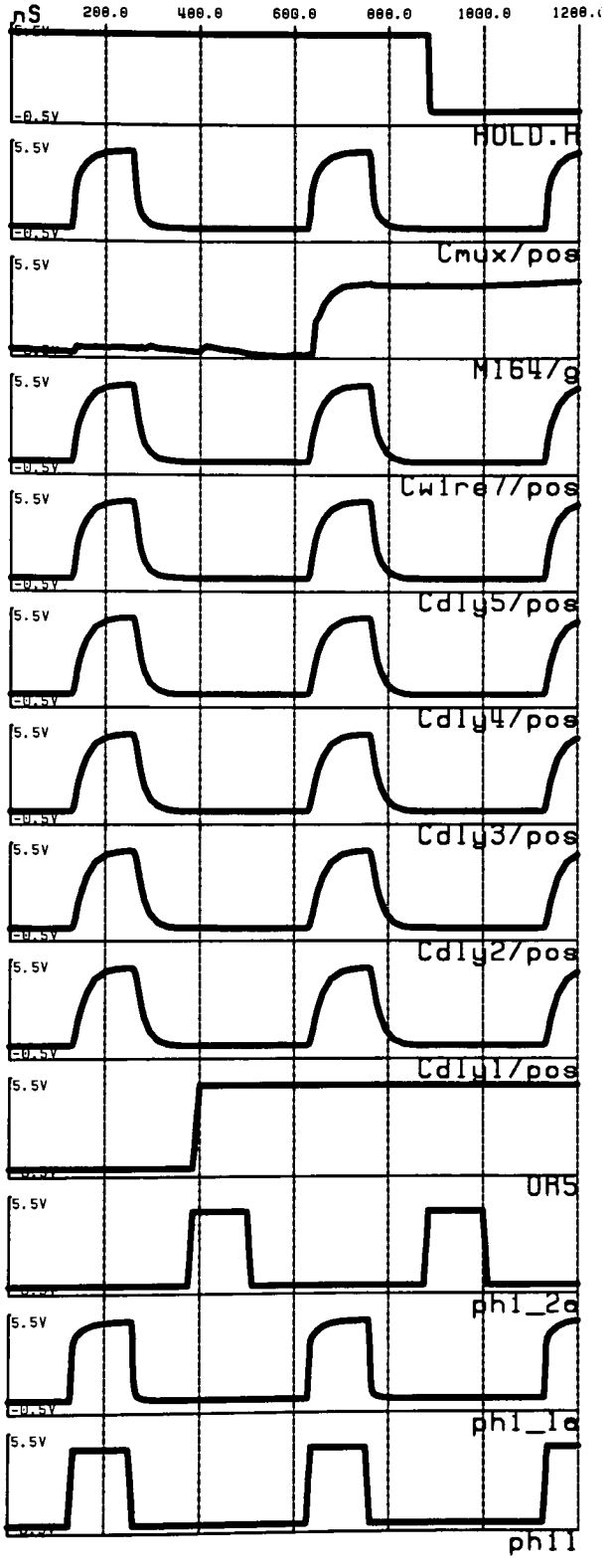


Figure 33 Simulation of PHI_1A Line

Counter's output HOLD.H before PHI_1A goes LOW. Simulation results (Figs. 33 & 35) shows no apparent difficulty in charging M164/g and effecting a change of logic level at HOLD.H.

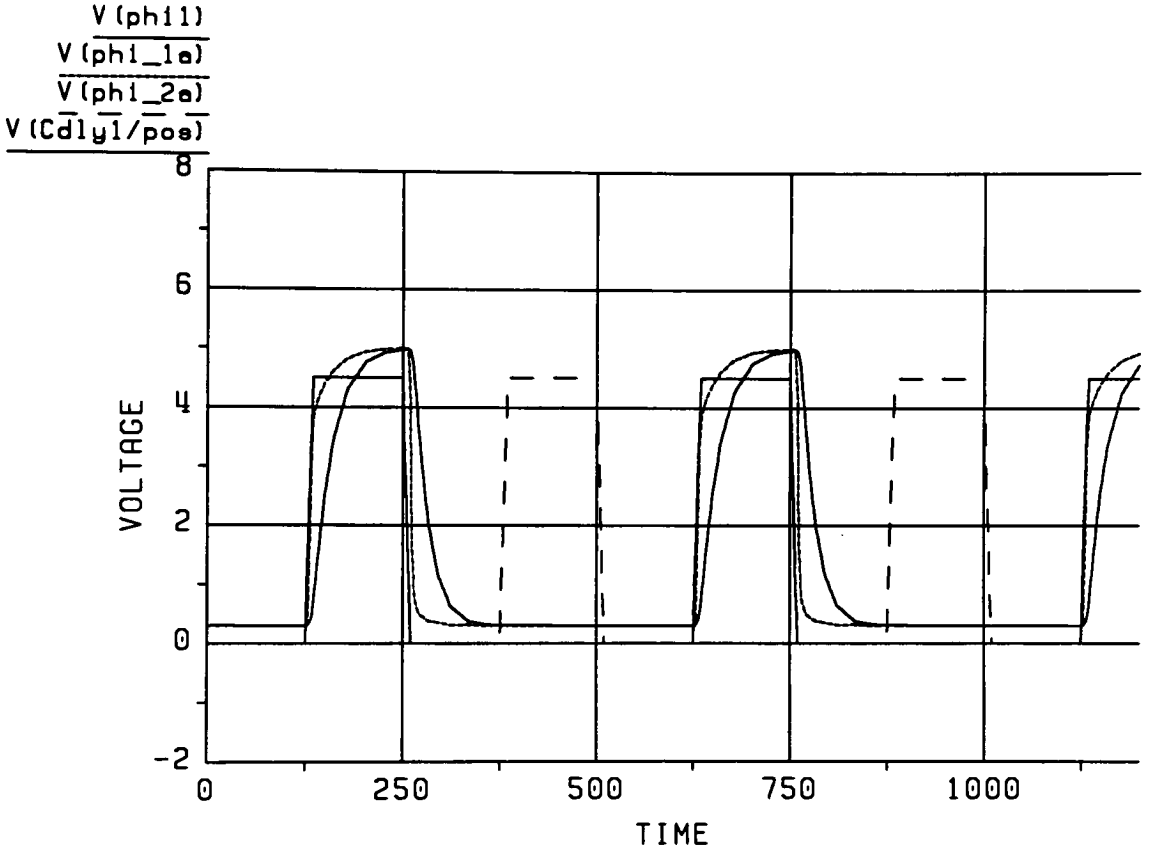


Figure 34 Clock Skew on PHI_1A Clock Line

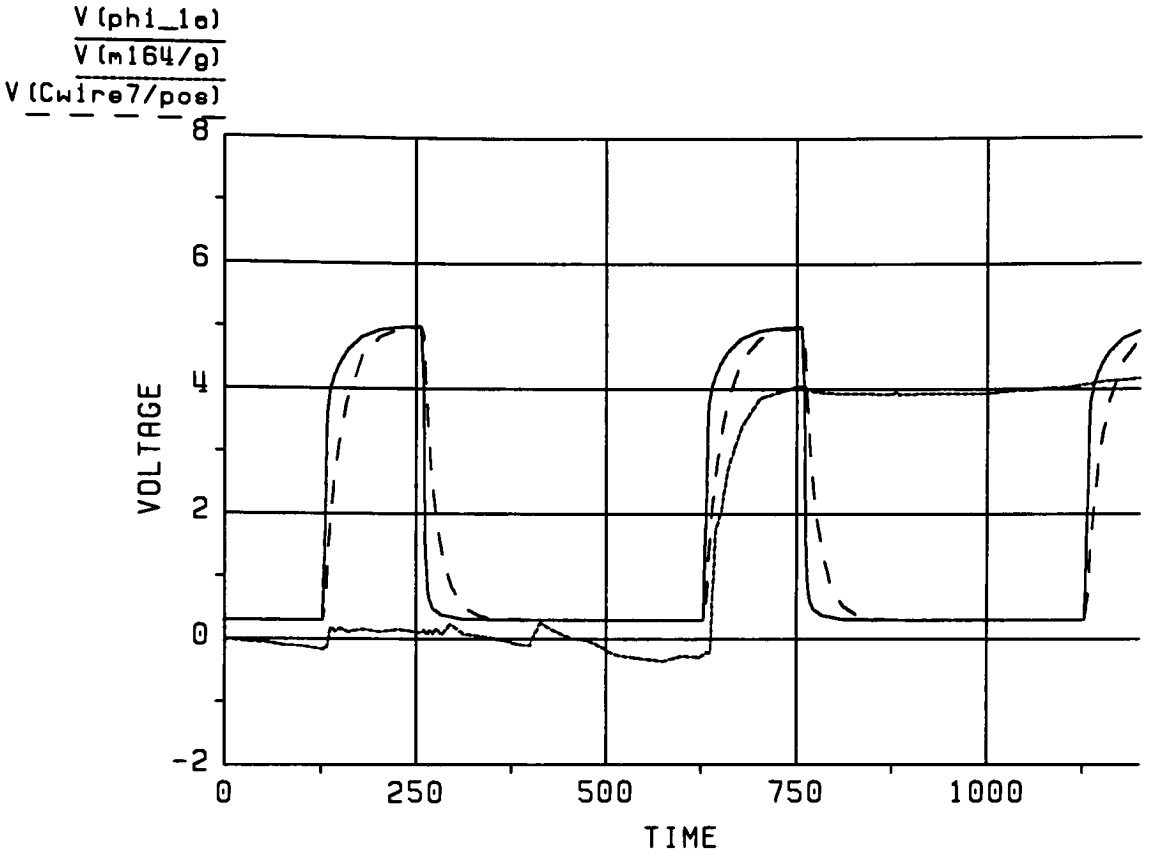


Figure 35 Loading of Delay Counter's Semistaitc Register Stage

8.12.2 PHI_2B Clock Line Loading

Figure 36 displays the distributive RC loading on the PHI_2B clock line. The PHI_2B clock line drives the five synchronizer cells ($R_{sync\#}/C_{sync\#}$ where # is equal to 1, 2, 3, 4, or 5), the SMC (R_{smc}/C_{smc}), ISM (R_{ism}/C_{ism}), and OSM (R_{osm}/C_{osm}). Figure 37 displays the resultant clock skew associated with each of the loads in response to the PHI_2B clock. One can notice the gradual increase in skew as a load gets further from the superbuffer output. The R_{smc}/C_{smc} and R_{ism}/C_{ism} loads experience the longest and virtually the same delay. Figure 38 provides a more visually obvious view of the skew between PHI_2B and the indicated loads.

The output of the superbuffer sourcing PHI_2B doesn't reach 4.95 volts until 70ns into its idealized 125ns ON time. C_{smc} just reaches 4.95 volts 59ns latter as PHI_2B begins to

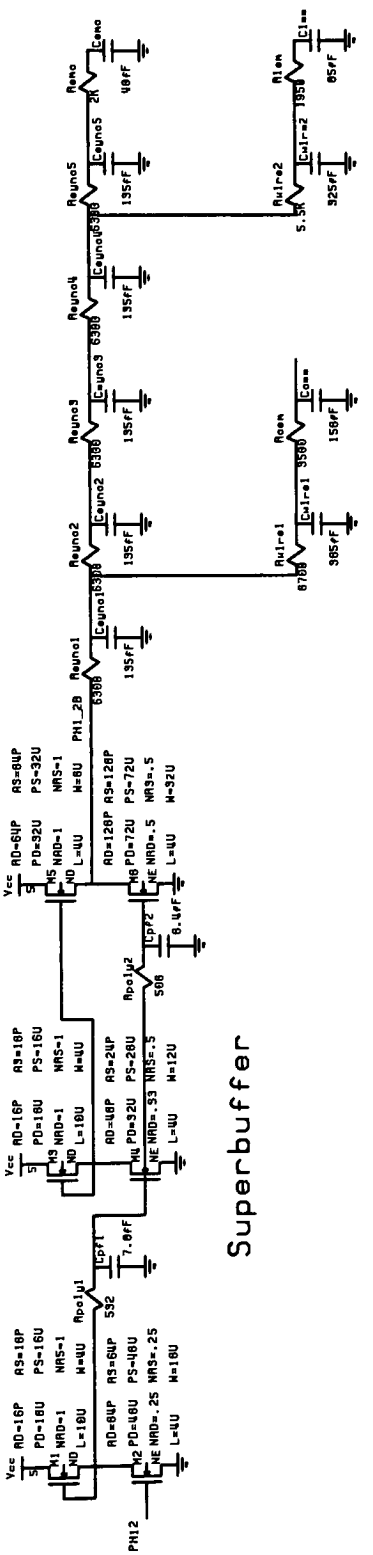


Figure 36 PHI_2B Clock Line Loading

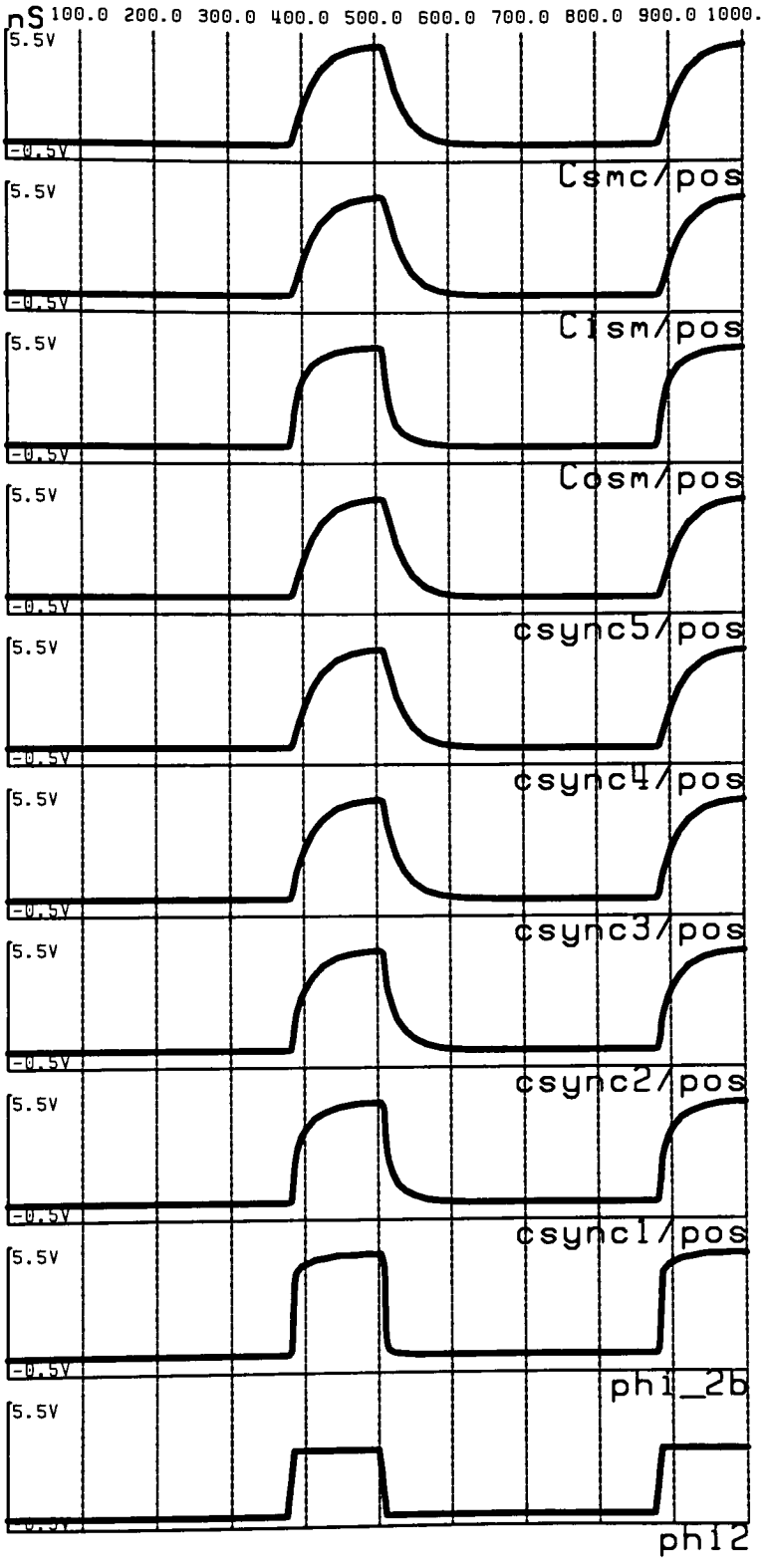


Figure 37 Simulation Waveforms of Loading on PHI_2B Clock Line

transition back LOW. As a result, PHI_2B clocked nodes in the SMC don't quite reach VCC - $[V_{th} + \alpha V_b]$. However, the pass transistors clocked by PHI_2B begin to conduct significant current once there gate voltage reaches threshold voltage. C_{smc} reaches 1 volt ≈ 17ns into the idealized PHI_2B ON time and doesn't fall back to 1 volt until 60ns after the idealized ON time has expired. This results in the SMC's clocked outputs remaining ON for 168ns ($125ns_{ideal}$ [60ns - 17ns]). Since signals passing through the PHI_2B clocked gates within the SMC are stable-ψ2, the SMC's outputs should be able to sufficiently charge HIGH or pull LOW any nodes they are driving in 168ns. The PHI_2B clock goes LOW ≈ 108ns before the idealized PHI_1B clock goes HIGH.

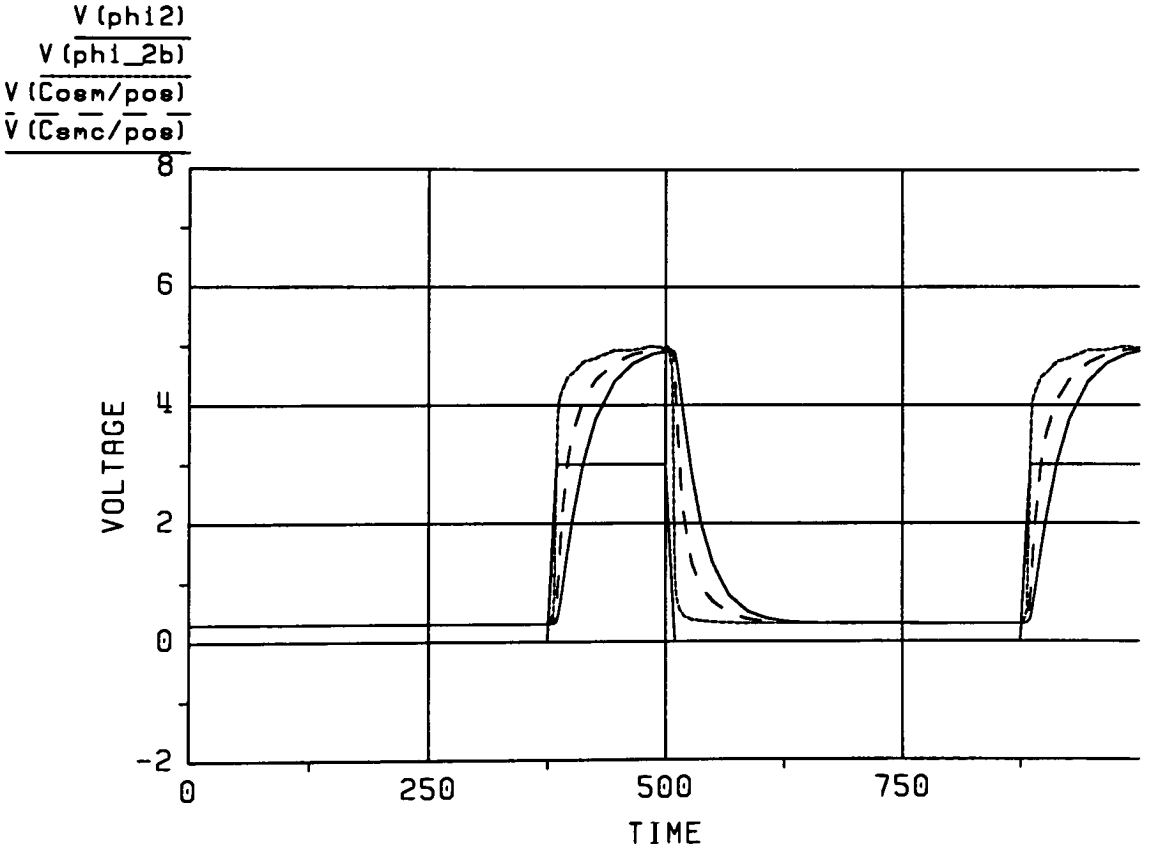


Figure 38 Clock Skew on the PHI_2B Clock Line

8.12.3 TE.H Signal Line Loading

Figure 39 displays the hand extracted loading on the TE.H signal line. Pass transistor M7 (clocked by PHI_2B) and the inverter composed of M8 & M9 represent the second half of the safe semistatic register stage in the SMC which sources TE.H. The figure shows TE.H being routed to the input of two superbuffers which generate the signals TE.H1 and TE.H2. TE.H1 is routed to the ISM (R_{ism}/C_{ism}) and to the Delay Counter ($R_{dly\#}/C_{dly\#}$ where # is 1, 2, 3, 4, or 5). TE.H2 is routed to the TRI_CTL port on ASIC pads REN.L, IFC.L, DAV.L, EOI.L, and ATN.L. TE.H2 also drives ASIC pad TE (R_{te}/C_{te}). The TE pin drives the R/$\overline{W}$ pin on the SRAM and the Talk Enable pin on the two GPIB transceivers which are represented by the load R_{ext}/C_{ext}. The author assumed a load capacitance of 10pF per external pin driven and a resistive load of 1KΩ per external pin driven. Hence, C_{ext} is 30pF and R_{ext} is 3KΩ.

Simulation of the TE.H signal line is displayed in Figure 40. TE.H=>T at the first PHI_2B clock pulse causes the buffer to make the transition from active listener to bus controller. The superbuffered outputs (TE.H1 and TE.H2) follow TE.H=>T without any significant delay. Except for the external load at C_{ext}, the remaining load waveforms appear to follow the TE.H1 and TE.H2 signals without significant delay.

TE.H1=>T turns ON pull-down transistors in the ISM which force it to enter state PON. The simulation shows C_{ism} reaching 1 volt (approximate threshold voltage of the pull-down devices) ≈ 11ns after PHI_2B goes HIGH at 375ns and reaching 4 volts ≈ 65ns after the assertion of PHI_2B. PHI_2B is ideally asserted for 125ns at a clock frequency of 2MHz. Therefore, these pull-down devices will be fully turned ON before PHI_2B is deactivated.

The Delay Counter load C_{dly1} represents the furthest load from both the SMC output TE.H and the superbuffer sourcing PHI_2A. The Delay Counter is clocked by PHI_1A and PHI_2A while the SMC is clocked by PHI_1B and PHI_2B. TE.H=>T starts the Delay Counter down counting. As discussed in section 8.7, there is a potential indeterminate signal problem in the toggle flip flops of the Delay Counter as TE.H1=>T. An indeterminate signal condition may occur if TE.H1=>T before or right after PHI_2A goes HIGH. The PHI_1A clock pulse at C_{dly1} was shown in section 8.12.1 to reach 1 volt ≈ 12ns after the start of the ideal PHI_1A ON time. PHI_1A is the more heavily loaded clock line between PHI_1A and PHI_2A. Since the author didn't simulate the PHI_2A clock signal, simply adding 250ns to the PHI_1A clock signal will adequately represent the PHI_2A signal. Therefore, one can assume that the PHI_2A clock at C_{dly1} reaches 1 volt 12ns into the idealized PHI_2A ON time. In the PHI_2B simulation (section 8.12.2), the PHI_2B signal at the SMC (C_{smc}) was shown to reach 1 volt 17ns into the idealized PHI_2B ON time. Therefore, the PHI_2A signal has at least a 5ns (17ns-12ns) head start on the TE.H1 signal. Also, the TE.H simulation shows the TE.H1 signal at C_{dly1} reaching 1 volt 22ns (at 397ns) into the idealize PHI_2B ON time and 21ns after the PHI_2B pulse used in the simulation reaches 1 volt. Therefore, the PHI_2A signal reaches 1 volt 26ns (21ns + 5ns) ahead of TE.H1. Although this treatment is by no means exact, it does provide some insight into weather the indeterminate signal problem is an issue. The treatment shows PHI_2A turning ON the clocked pass transistors at C_{dly1} before the pass transistor gating TE.H1 is turned ON by PHI_2B. The treatment also shows that the TE.H1 signal experiences significant delay before reaching 1 volt at the C_{dly1} load. Based on the

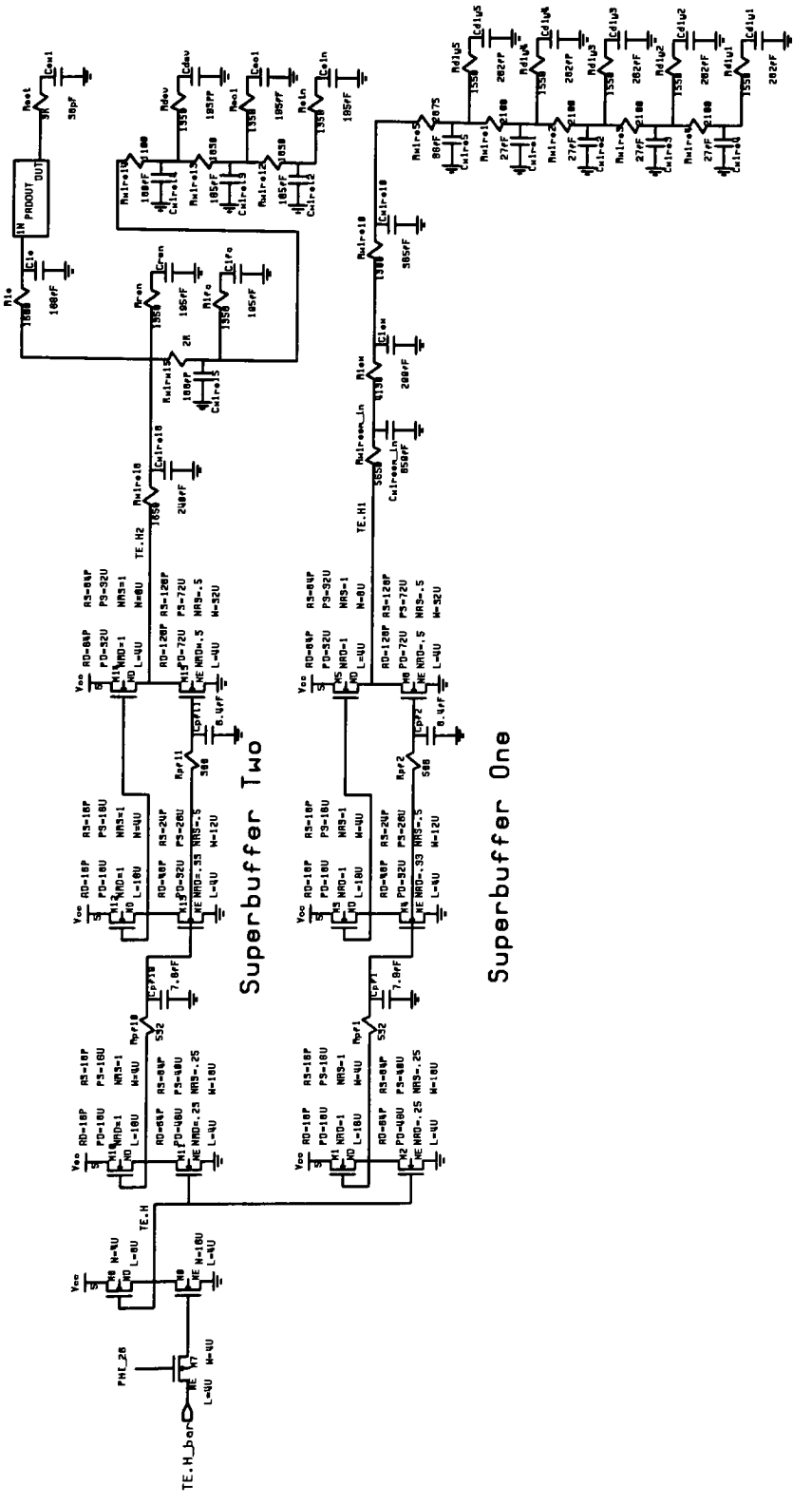


Figure 39 Schematic of Hand Extracted Loading on the TE.H Line

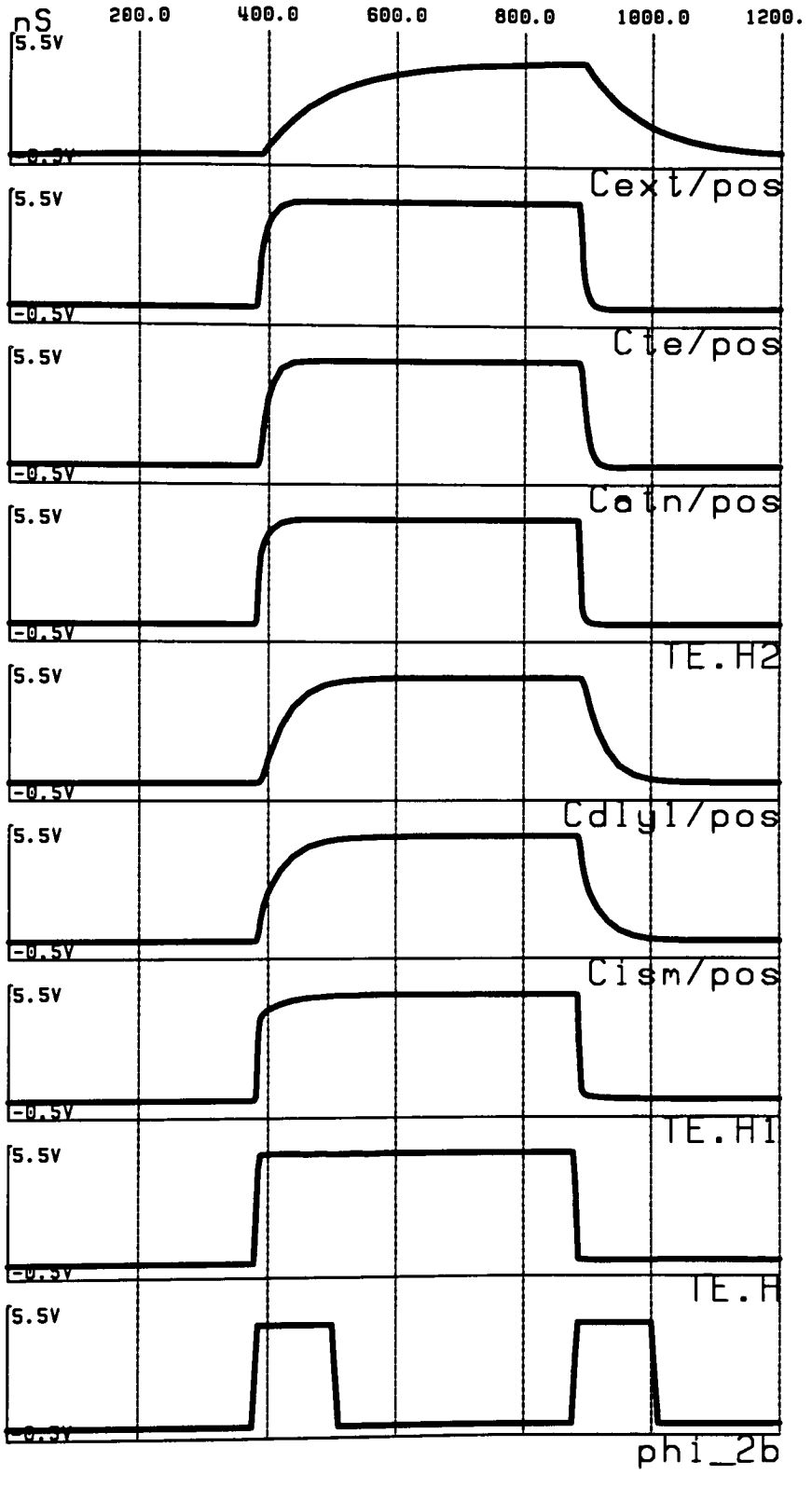


Figure 40 Simulation of TE.H Line with Parasitic Parameters Added

results presented here and full ASIC simulation results, the author is inclined to believe that potential indeterminate condition in the Delay Counter is not a concern.

TE.H2=>T also enables the drivers on ASIC pads REN.L, IFC.L, DAV.L, EOI.L, and ATN.L. The ATN.L pad is the furthest pad from the SMC's outputs. C_{atn} (load representing the TRI_CTL port on ASIC pad ATN.L) reaches 1 volt at = 13ns after PHI_2B goes HIGH and 4 volts = 19ns latter. Thus, the buffer doesn't appear to have any problem enabling these pad drivers before the next ψ1 clock pulse.

ASIC pin TE going HIGH causes the direction in which signals travel through the two GPIB transceivers to reverse. TE going HIGH also causes the SRAM to leave WRITE mode and enter READ mode. [However, DC.H=>F causes the SRAM to enter reduced power standby mode.] The GPIB transceivers and SRAM use TTL logic levels (V_{ILmax} = 0.8 volts and V_{IHmin} = 2.0 volts). TE rises to 2 volts = 83ns into the $125ns_{ideal}$ PHI_2B ON time. From the time that the GPIB transceivers' control signal (TE) switches to pins on the buffer side of the transceivers going into high impedance or to a valid logic level takes a maximum of 55ns. Therefore, 143ns (55 + 83) after PHI_2B goes HIGH or 112ns before PHI_1B ideally goes HIGH, bus signals will have theoretically reversed direction through the transceivers. As discussed earlier, there may be noise on the signals passing through the transceivers once TE.H2=>T. Therefore, the Delay Counter holds the buffer in OSM state TRANS for = 100μs after the SMC's outputs switch.

At the second PHI_2B clock pulse, TE.H=>F. TE.H=>F causes the buffer to leave command mode and re-enter listener mode. Again, the only load showing significant delay is C_{ext}. When TE.H=>F, C_{ext} falls to 0.8 volts = 174ns seconds **after** PHI_2B transitions LOW or 76ns before PHI_1B ideally goes HIGH. In this simulation, C_{ext} had only reached 4.2 volts before PHI_2B goes HIGH a second time. Under normal buffer operation, ASIC pin TE would be at 5 volts when the SMC outputs switched. Therefore, the time needed for C_{ext} to fall to 0.8 volts will actually be more than shown in the simulation. Pins on the buffer side of the transceivers need 55ns maximum to reach high impedance or a valid logic level. Therefore, the possibility exist that bus signals at the ASIC's inputs will be indeterminate or at the wrong logic level when the next ψ1 clock pulse occurs. However, this only present a problem if DAV.L is LOW when ψ1 goes LOW 375ns after the assertion of PHI_2B ideal. The buffer design realize on ASIC pin DAV.L not experiencing any significant noise as TE.H=>F and DC.H=>T. When the buffer transitions back to the active listener configuration, it waits in ISM state PON for the talker to initiate communication by asserting DAV (LOW). If there is ring or spikes while ψ1 is asserted, the ISM could enter its next state (RDY) in response to DAV.L being LOW momentarily due to electrical noise. The author believes this scenario is unlikely for the following reason. In OSM state CMD_ACC, the buffer drives DAV.L HIGH. Therefore, the buffer trace DAV.L and bus signal line DAV will be charged HIGH. When the transition occurs, the DAV bus line won't be pulled LOW until the user starts the next file transfer. Since the DAV signal line isn't assuming a new logic level, the DAV.L signal shouldn't experience much noise as the direction in which this signal travels through the transceiver is reversed. With the discrete buffer designs, the author hasn't seen any evidence of the buffer entering a wrong state while resuming the active listener configuration. The simulation suggestion of such an occurrence may be due to the assumed loads of 10pF and 1KΩ

per external pin driven being too high.

TE.H2=>F disables the drivers on the ASIC pads mentioned above. C_{atn} falls to 1 volt ≈ 17ns after the second PHI_2B clock pulse goes HIGH. TE.H1=>F immediately forces the Delay Counter output (HOLD.H) and its internal toggle flip flops into a reset condition. The simulation shows C_{dly1} falling to 1 volt ≈ 77ns after PHI_2B goes HIGH. TE.H1=>F also causes the ISM to become active since the pull-down transistors holding it in state PON are turned OFF. C_{ism} falls to 1 volt ≈ 65ns into the $125ns_{ideal}$ PHI_2B ON time. Since all these events occur well before PHI_2A & PHI_2B goes LOW, there should be no problem with normal buffer operation due to signal delay on the TE.H1 or TE.H2 lines.

8.13 ASIC Power Requirements

Computation of the ASIC 's maximum static power consumption is a two step procedure.* First, one must determine the operating condition which causes the largest number of pull-up devices to be in saturation at the same time. Secondly, the current drawn by these devices is computed. One starts calculating the current draw of devices at the leaves of the power tree making sure that the supporting branches are wide enough to supply sufficient current (both static and dynamic) as one works back toward the power trunk.

> "The maximum current rating for a metal line is about 1 mA/μm. For λ = 2 μm, metal will carry about 2.0 mA/λ. The minimum width of metal is 3λ; it is usually made 4λ wide to fit contact cuts, which gives about 8 mA at the lowest level."**

The saturation current (equation 15) conducted by a MOS device is dependent on process parameters carrier surface mobility (μ), gate oxide thickness (T_{ox}), threshold voltage (V_{th}), as well as the physical length (L) and width (W) of the device. Using Mukherjee's process gain factor ($\mu\epsilon_{Si02}\epsilon_o/T_{ox}$) of 30 μA/V^2 and V_{thdep} = -4 volts, (equation 15) results in I_{dsSAT} = 0.24 (W/L) mA.

$$I_{dsSAT} = \frac{\mu\epsilon_{Si02}\epsilon_o}{2\,T_{ox}} \frac{W}{L} (V_{gs} \quad V_{th})^2 \qquad (15)$$

The author determined that the ASIC draws the most current at power up*** while PHI1, PHI2, CLK_PULSE, SWITCH, and PQ.L are LOW, CLK_TRI and PON are HIGH, and four out of five A5.L-A1.L address bit pins are LOW. During power up ASIC pads DO8-DO1, REN.L, IFC.L, DAV.L, EOI.L, and ATN.L are all tristated. The author took the length and width of all depletion mode devices operating in saturation under these conditions and applied I_{dsSAT} = 0.24 (W/L) mA. This resulted in ≈ 100mA of total static current drawn by the ASIC. The ASIC's pads account for 77mA of the 100mA value. Table 12 displays the calculated static current draw of ASIC pads when forced HIGH, LOW, or in high impedance.

---

*Amar Mukherjee, Introduction to nMOS & CMOS VLSI System Design, Englewood Cliffs, N. J. : Prentice-Hall, 1986, pp. 206-208.

**Amar Mukherjee, Introduction to nMOS & CMOS VLSI System Design, Englewood Cliffs, N. J. : Prentice-Hall, 1986, p. 207.

***Power rail has reached 5 volts, ASIC's clock is running, buffer waiting for power on circuitry to time out and drive ASIC pin PON LOW.

TABLE 12 Calculated Static Current Draw of ASIC Pads in mAmps

PAD	HIGH	LOW	HIGH Z
1. PADIN_BUFFERED	.096	.576	
2. PAD_OUT	1.92	1.92	
3. PAD_TRISTATE	2.76	2.76	3.84
4. PAD_TRISTATE_BUFFERED	3.24	3.33	3.936/4.416*

Figure 41 shows the schematic of a tristateable, bi-directional pad driving a one pin external load. Simulation results are displayed in Figure 42. While the pad's drivers are enabled (TRI_CTL equals HIGH), depletion mode devices M1, M3, & M12 are conducting I_{dsSAT} static current while the OFF_CHIP port is held HIGH. In this condition, the simulation shows the pad drawing about 2.9mA of static current: M1/s = -0.9mA, M3/s = -0.5mA, and M12/s = -1.5mA. When the OFF_CHIP port goes from LOW to HIGH at 50ns, the simulation shows enhancement transistor M15/s conducting a dynamic peak current of -12mA as it charges the external load and then decreasing back to ≈ 0mA (leakage current) 100ns latter. When the pad's OFF_CHIP port is forced LOW at approximately 300ns into the simulation, depletion mode devices M1, M6, & M9 conduct I_{dsSAT} static current. Enhancement mode transistors M16 also sinks a peak dynamic current of -6mA before settling back to ≈ 0mA 50ns latter. When the pad is tristated (TRI_CTL equals LOW) at 700ns in the simulation, depletion mode devices M3, M6, M9, & M12 are all in saturation. The simulation shows the pad drawing ≈ 4.0mA while tristated. In the simulation, the author used different values of μ, T_{ox}, and V_{th} than those used by Mukherjee. This accounts for the difference in current draw values resulting from simulation and those listed in Table 12 for PAD_TRISTATE.

After calculating the ASIC's current draw, the author realized that the ASIC's VCC and GND rails to the pads needed to be widen. The VCC rail was widen to 60μm around the perimeter of the chip. This allows a maximum current carrying capability of 60mA before the onset of metal migration. The ASIC pads were calculated to draw a maximum of 80mA static current. The two VCC pins are diagonally opposite of each other on the chip. If the pads' current requirements are equally distributed, each VCC pin will supply 20mA to its left and 20mA to its right. This would leave 40mA (60mA - 20mA) of dynamic current capability on the VCC rail. Starting with the DO6 pad and traveling clockwise to the LE.H pad, the total static current consumed by pads along this section of the VCC rail is ≈ 42.7mA. If each VCC pin supplies half of this static current demand, the remaining dynamic current capability in this section of the VCC rail is 38.65mA. Pads on the lower section of the VCC rail consume

*Pad acts as input when tristated. When pad is driven LOW externally, it draws 4.416mA; when driven HIGH, it draws 3.936mA.

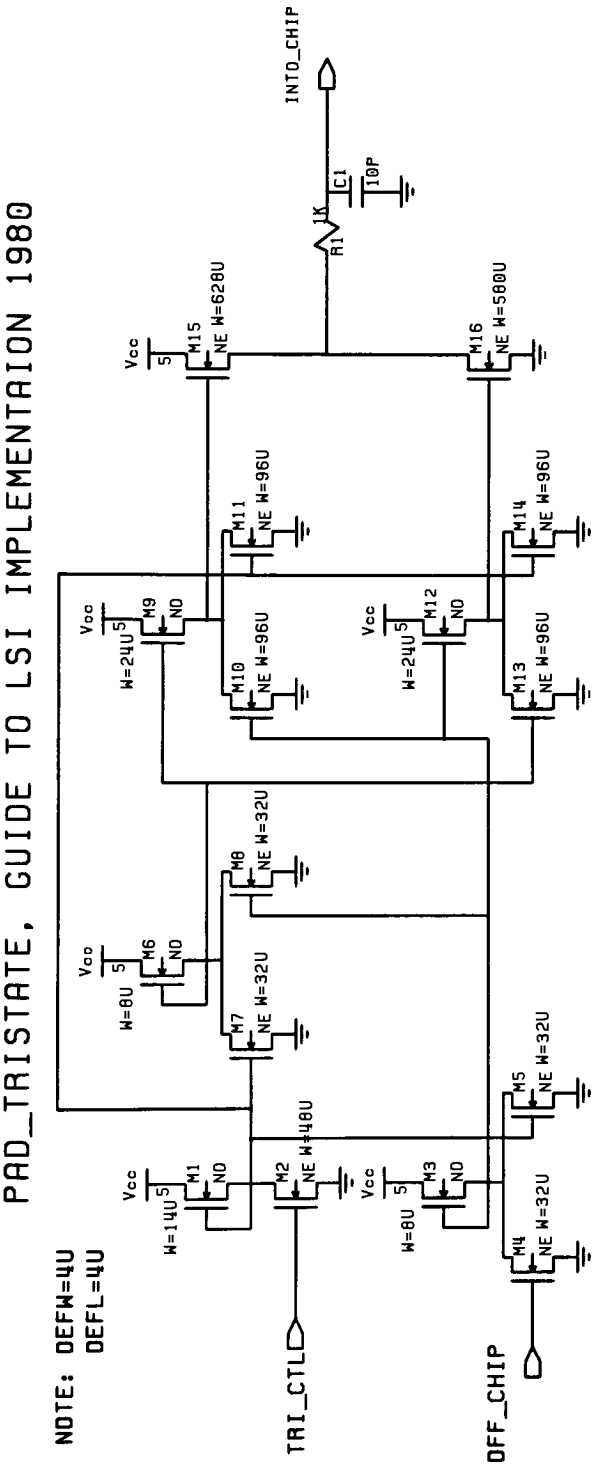


Figure 41 Schematic of Pad_Tristate Used to Determine Pad Current Draw

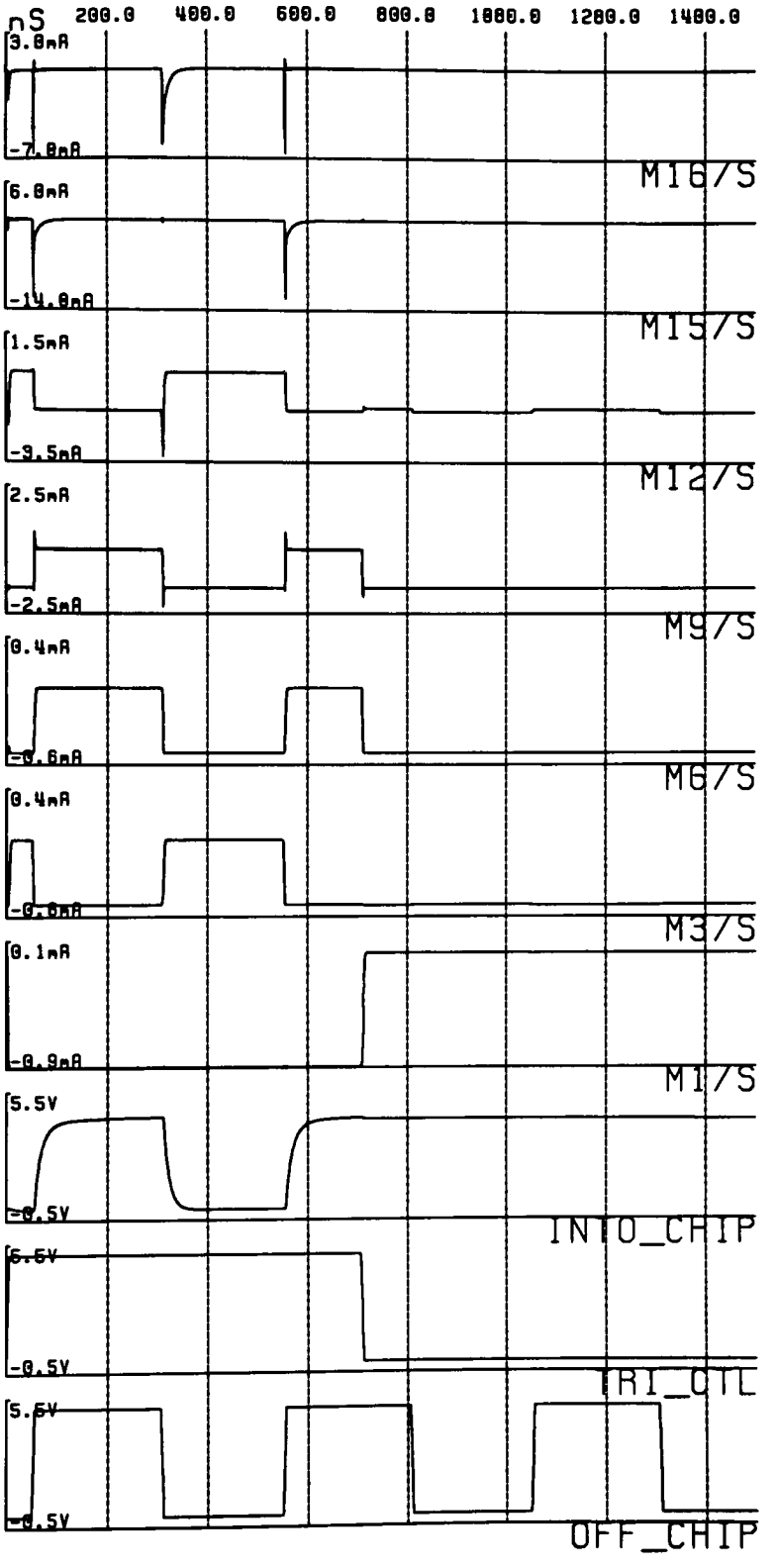


Figure 42 Simulation of Current Drawn by Pad_Tristate

≈ 33.25mA of static current. If each VCC pin supplies half of the static requirement, the remaining dynamic current capacity in this section of the VCC rail is 43.38mA.

The GND rail routed through the ASIC's pads can only be widen a minimal amount before encroaching on the metal bonding pads on one side and pad metal interconnects on the other side. Widening the GND rail significantly requires lengthening the layout of the pads themselves. Since the author calculated current draw after chip layout was thought to be complete, lengthening the pads would have met a major revision of the chip layout. Therefore, the GND rail was made as wide as possible within the pads. Starting with ASIC pad A1.L and going right to pad ACLR.L, the GND trace through these pads is 40μm wide. These pads sink ≈ 14mA of static current. This allows a dynamic current capability of 26mA in this section of the GND rail. Traveling from ASIC pad D01 to pad DO5, the GND trace is 52μm wide. These pads sink ≈ 19.2mA of static current which leaves a GND rail dynamic current capability of 32.8mA. Along the right side of the chip the GND rail is 40μm wide in pads TE through LE.H. These pads sink ≈ 10.3mA of static current which results in a 29.7mA dynamic current capability in this section of the GND rail.

The GND rail is 52μm wide in ASIC pads D06 through ATN.L. It then narrows down to 38μm through ASIC pads EOI.L, DAV.L, NRFD.L, and NDAC.L. In pad PQ.L, the GND rail widens to 44μm; it widens again to 52μm through pads IFC.L and REN.L. The GND pad between ASIC pads DO8 and ATN.L was originally the SRQ.L pad. After computing the ASIC's maximum static current consumption, the author replaced the SRQ.L pad with the present GND pad. The GND rail can only be widen to 38μm through ASIC pads EOI.L, DAV.L, NRFD.L, and NDAC.L. When the SRQ.L pad was present in the layout, pads DO6 through NDAC.L were calculated to sink ≈ 31.3mA of static current. This left only a 6.7mA (38mA 31.3mA) dynamic current capability in this section of the GND rail. In the simulation above, one bi-directional pad sank a peak dynamic current of 6mA when pulling a 1KΩ/10pF load LOW. This section of the GND rail contains six bi-directional pads which pull external loads LOW at various times. The author was uncomfortable with such a low dynamic current capability in this section of the GND rail. Instead of resorting to a major revision of the chip layout, the author replaced the SRQ.L pad with the additional GND pad. This lowers the static current in the GND rail along the top of the ASIC and increase its dynamic current capacity. With the third GND pad included in the layout, the worst case static current condition is 10.4mA in the 38μm wide section of the GND rail which leaves a 27.6mA dynamic current capability. Replacing the SRQ.L pad with a GND pad doesn't present any problems. The ASIC never asserts SRQ signal while a listener, and doesn't response to requests for service while a controller. If the SRQ pin is left unconnected on the buffer side of the SN75161 GPIB transceiver, the bus side simply floats HIGH during the *data output phase*, since the SRQ line uses open collector drivers.

Table 13 provides an estimate of the static current requirement for the regulated 5 volt power supply needed by the RIT IEEE-488 Buffer. Of course, the power supply must be capable of sourcing perhaps two-to-three times the 700mA value dynamically.

TABLE 13 Estimate of Buffer Static Supply Current

1. ASIC		100 mA
2. GPIB Transceivers	2 * 110 mA	220 mA
3. SRAM		105 mA
4. Counters	4 * 36 mA	144 mA
5. Comparators	2 * 17 mA	34 mA
6. Latches	2 * 27 mA	54 mA
7. NAND		4.4 mA
8. Schmitt Triggers		21 mA
9. Resistors		
a.	7 * (5.25 / 4.7KΩ)	7.8 mA
b.	6 * (5.25 volts / 10KΩ)	3.2 mA
c.	(5.25 volts / (1KΩ + 5KΩ))	0.9 mA
		700 mA

9. Recommendations for Future Designs

The author realized after completion of the chip layout that ASIC pad IFC.L is not needed. This signal is used passively by the buffer; it is never asserted. Hence, its OFF_CHIP port is tied HIGH the non-asserted state. This signal is connected to the IFC pin on the buffer side of the SN75161 GPIB transceiver. A signal pin on the bus side of the transceiver floats HIGH when its corresponding pin on the buffer side is not driven (Fig. 3). In other words, the buffer side must be actively driven LOW to pull its corresponding pin on the bus side LOW. Therefore, this pin could have been left unconnected on the buffer side of the transceiver. It then would appear to other devices on the bus as if the IFC line had not been asserted unless a device other than the buffer asserted it. This would free up one pin on the ASIC for other uses. If a revision of the chip layout is undertaken, the above discussion should be kept in mind.

The author also realized after completion of the ASIC layout that equation 11 could have been reduced to just MUX.L = DC.H. This would eliminate the three input NOR gate in the Command Multiplexer and would shorten two long runs in the layout, since PI0.H and PI1.H would not need to have been run to the Command Multiplexer. This should be kept in mind if a revision of the ASIC layout is undertaken.

In recent days, the author learned some facts from the IEEE Std. 488-1978 that he wasn't aware of during the design of the two discrete buffers and the present integrated buffer. The author was mislead by a Hewlett Packard tutorial on the IEEE-488 bus in reference to the assertion/deactivation of the signal REN. The author was under the impression that the controller had to allow 100μs for instruments to prepare their bus interface for communication after the *assertion* of REN. The Standard actually requires that instruments return to local programming mode within 100μs after the *deactivation* of REN. Therefore, the Delay Counter module is not really needed. The Delay Counter is useful for holding the buffer in OSM state TRANS while any noise present on bus signal lines subside. However, the 100μs delay period can most likely be reduced significantly.

Devices may have a problem distinguishing the first device dependent data byte from a bus command since ATN and DAV may be momentarily TRUE as the buffer enters OSM state DAT_OUT. This potential confusion can be eliminated by deactivating ATNO.L in OSM state CMD_ACC instead of state DAT_OUT. The author left ATNO.L LOW in state CMD_ACC in order force instruments to drive their NDAC line driver LOW and their NRFD line driver HIGH. In this condition, all devices are waiting for the active controller to make available the next command byte. Since the intended receiver has already been addressed to listen, there is no need to keep other devices waiting to receive a multiline command. The author determined recently from the IEEE-488 specification that the addressed listener enters the listener active state (LACS) within 200ns of ATN going FALSE.* Then, once the addressed listener is ready to receive the first device dependent data byte, it enters acceptor

*IEEE Standard Digital Interface for Programmable Instrumentation, New York: The Institute of Electrical and Electronics Engineers, Inc. 1978, section 2.6, pp. 29-34.

ready state (ACRS) where it drives its NDAC line driver LOW and its NRFD line driver HIGH.* These are the same conditions necessary along with MUX.L=T for the OSM to leave state CMD_ACC and enter state DAT_OUT. Devices not addressed to listen enter the acceptor idle state (AIDS) in response to ATN going FALSE. After the buffer had completed data transfer and unaddressed the addressed listener, all instruments would enter state AIDS causing NDAC and NRFD to be driven HIGH (FALSE) if ATNO.L were set FALSE in state CMD_ACC. Therefore, SMC equation 2 would have to be changed to equation 12 in order for the buffer to make the transition back to active listener. Bringing ATNO.L FALSE in state CMD_ACC also requires some modifications to the OSM. The author has already run the Espresso file which resulted in the modification of just a few OSM product terms.

$$DC.H = CMD\_ACC.H * \overline{NDACO.L} * \overline{NRFDO.L} * \overline{MUX.L} + RESET \quad (12)$$

Ideas for major design changes include installing an on board 5 volt power supply, doing a board layout, converting the ASIC design to CMOS, adding design for testability circuitry, making the buffer queueable, and the addition of more features. One feature might be not requiring EOI LOW to be sent with the last data byte. For example, the Tektronix's 2430A Digital Oscilloscope doesn't send EOI (end or identify) along with the last data byte when sending HP-GL strings. This leaves the buffer with no way to determine that data transmission has ended. This feature would enable the buffer to start file transfer to an intended listener after a time out period had expired in which the buffer received no further data. Such a feature would make the buffer compatible with many more IEEE-488 devices. Another feature might include having the buffer issue the UNL command as part of its reset sequence. When a user presently aborts file transfer (reset switch placed in the CLEAR position), the addressed listener might remain in listener mode. Thus, when the user goes to send another file, both the device in listener mode and the buffer accept data from the talker. This defeats the purpose of using the buffer. As presently designed, the user must power OFF and ON the device to get it out of listener mode after aborting file transfer. Having the UNL command incorporated into the buffer reset sequence would unaddress any addressed listeners thus relieving the user from power cycling the intended receiver.

The author had intended to make the integrated buffer design queueable user able to send a second file to the buffer before it had finished transfer of the first file to the addressed listener. The author spent three weeks trying to come up with state diagrams for such a buffer but had trouble keeping all the details straight. The author would recommend the use of some top down design tools (VHDL) for such a project.

A queueable buffer (Fig. 43) would require the unit to posses two GPIB ports (connectors). Such a buffer would provide isolation between fast talkers and slow listeners on the bus. A queueable buffer could also be designed for used in systems with controllers. Using the queueable buffer in a system with a controller would require that the buffer accept the

*IEEE Standard Digital Interface for Programmable Instrumentation, New York: The Institute of Electrical and Electronics Engineers, Inc. 1978, section 2.4, pp. 22-25.

intended receiver's address (MLA command) over the bus from the active controller rather than entry of the intended receiver's address via dip switches. A file would be accepted from a talker on the buffer's listen port, and stored in memory. If the buffer contained a revolving memory management setup, the buffer's talk port could then start sending data to the addressed listener while still receiving the rest of the file from the talker. The buffer would have keep track of the start and end address of each stored file and the address of each intended receiver. It would also have to hold off further data transfer between talker and buffer if memory became full.

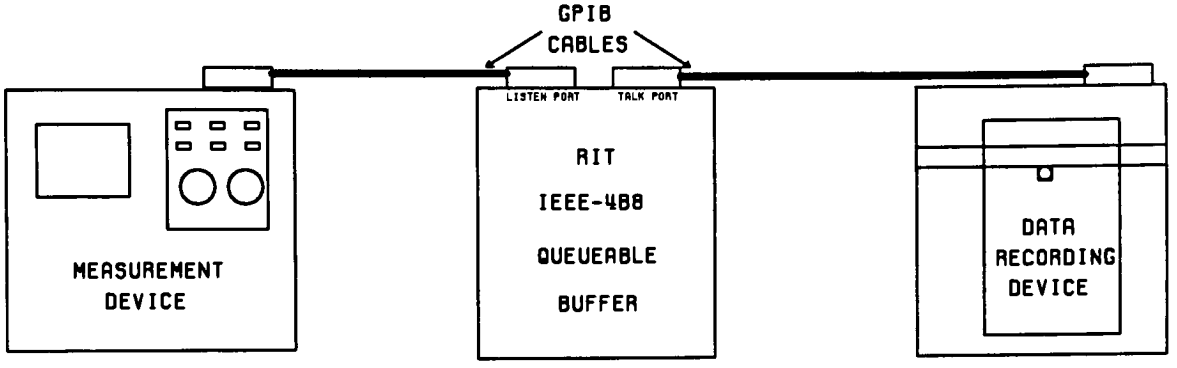


Figure 43 System Using Proposed Queueable Buffer

10. Conclusion: Let's fabricate this thing and see if it works!

The thesis discussed the design, simulation, and full custom layout of an NMOS ASIC used to control the RIT IEEE-488 Buffer. The ASIC incorporates a multiple controller architecture, internal clock generation, and synchronization of asynchronous inputs. The buffer design utilizing the ASIC reduces the major component count from 50 in the discrete synchronous, one-hot design to 16 not counting incidental resistors and capacitors.

The author raised several issues in regards to the ASIC design. When designing in NMOS, one needs to consider early in the chip layout the amount of current drawn by pads and make sure that the power and ground rails are wide enough to accommodate both static and dynamic requirements. Knowledge of external loading is needed to accurately compute dynamic current needs. Examination of clock skew and critical path delay are also important factors in ASIC design.

As far as ***successful*** chip design and fabrication at RIT is concerned, the author see three areas which need to be addressed. First, the Microelectronic Engineering Department must make available to designers ***reliable*** process parameters. Reference lists of parameters for NMOS, CMOS, and Bipolar processes can be found in reference 16, Appendix 2. Secondly, the Computer Engineering Department needs to investigate acquiring and installing the software needed to run the netlist generated from LPE in AccuSim. Also, it would be ideal if PRE (parasitic resistance extraction) available from Cadence was obtained. This would allow more realistic simulation of the student ***entire*** integrated circuit designs. The author had to resort to hand extraction of parasitics. This is a tedious and error prone endeavor. As a result, the author only modeled and simulated portions of the ASIC.

A. Hardware Parts List

- one RIT IEEE-488 Buffer Controller (ASIC)
- four 4-bit counters — SN74ALS561AN
- two octal latches — SN74ALS573AN
- one 32K-word x 8-bit static ram — M5M5256P-15L
- two octal identity comparators — SN74ALS518N
- two octal GPIB transceivers — SN75160BN, SN75161BN
- one NAND gate — SN74LS00
- one Schmitt Trigger — SN74LS14
- other assorted capacitors and resistors

8	4.7KV
1	40KΩ
3	10KΩ
1	1KΩ
12	0.1μf
1	8.0μf
1	7.0μf

- one external 5 volt power supply capable of delivering at least 0.7 Amps.
- one 8-bit dip switch used to input the data recording device's address
- one 24 pin GPIB connector
- one single pull double throw switch
- one GPIB cable

B. Selected References

1. IEEE Standard Digital Interface for Programmable Instrumentation, New York: The Institute of Electrical and Electronics Engineers, Inc. 1978.

2. Tutorial Description of the Hewlett-Packard Interface Bus, Hewlett-Packard Company, Rev. November, 1987.

3. Pippenger, D.E. and Tobaben, E.J., Linear and Interface Circuit Applications, Vol. 2 : Line Circuits, Display Drivers, Texas Instruments Incorporated, 1985.

4. Programming Manual, Model 220/230, Keithley Instruments, Inc. 1982.

5. Prosser, Franklin P. and Winkel, David E., The Art of Digital Design, 2nd ed. ; New Jersey: Prentice-Hall Inc., 1987.

6. Linear Circuits Data Book, Texas Instruments Incorporated, 1984.

7. ALS/AS Logic Data Book, Texas Instruments Incorporated, 1986.

8. Interface Circuits Data Book, Texas Instruments Incorporated, 1987.

9. The TTL Logic Data Book, Texas Instruments Incorporated, 1988.

10. Palmer, J. E., Class notes, EEEE 650 Design of Digital Systems, Rochester Institute of Technology, 1988.

11. Robert W. Hon and Carlo H. Sequin, A Guide to LSI Implementation, 2nd ed.; Xerox, Palo Alto Research Center, 1980.

12. Amar Murkerjee, Introduction to nMOS & CMOS VLSI System Design, Englewood Cliffs, N. J. : Prentice-Hall, 1986.

13. John Newkirk and Robert Mathews, The VLSI Designer's Library, Reading, Massachusetts : Addison-Wesley Publishing Company, Inc. 1983.

14. IEEE Standard Dictionary of Electrical and Electronics Terms, Fourth Edition, New York, N. Y., The Institute of Electrical and Electronics Engineering, Inc. 1988.

15. Neil Weste and Kamran Eshraghian, Principles of CMOS VLSI Design, A System Perspective, Reading, Massachusetts : Addison-Wesley Publishing Company, Inc. 1985.

16. Randall Geiger, Phillip Allen, and Noel Strader, VLSI Design Techniques for Analog and Digital Circuits, New York, N.Y: McGraw-Hill Publishing Company, 1990.