



D0 Silicon Strip Detector Upgrade Project

Port Card Module

D0 Engineering Note Number 3823.112-EN-393

May 2, 1994

M. Utes

D0 Silicon Vertex Detector Port Card

- 1 SYSTEM INTRODUCTION**
- 2 GENERAL DESCRIPTION OF PORT CARD**
- 3 DATA STRUCTURE OF THE CONTROL LINK**
- 4 EPLD**
 - 4.1 Synchronization of the Control Link**
 - 4.2 Decoding the Control Codes**
 - 4.3 Master State Machine**
 - 4.3.1 Parity Error**
 - 4.3.2 Loss of Synch**
 - 4.3.3 Loss of G-LINK Lock**
 - 4.3.4 Diagnostic Mode**
 - 4.4 Main State Machine**
 - 4.4.1 Initialization**
 - 4.4.2 Acquisition**
 - 4.4.3 Digitization**
 - 4.4.4 Readout**
 - 4.4.5 Power Up**
 - 4.4.6 Digitize Test Pulse**
- 5 LOW SPEED FIBER OPTIC LINK**
- 6 HIGH SPEED FIBER OPTIC LINK**
- 7 INITIALIZING THE CHIPS**
 - 7.1 VME Interface**
 - 7.2 1553 Interface**

1 SYSTEM INTRODUCTION

The Port Card will be one link in the data acquisition system for the D0 Silicon Vertex Detector. This system consists of the following parts, starting at the detector: Silicon strip detectors are mounted in a spaceframe and wire-bonded to custom bare-die integrated circuits (SVX-II chips) that digitize the charge collected by the strips. The 128-channel chips are mounted on a High-Density Interconnect (HDI) that consists of a small flex circuit that routes control signals and eight data bits for each of three to ten chips onto a common data bus. A cable then routes this bus approximately thirty feet out from the detector to the Port Card. The Port Card houses a commercial chipset that serializes the data in real time and converts the signal into laser light impulses that are then transmitted through a multi-mode optical fiber about 150 feet to a Silicon Acquisition & Readout board (SAR)¹. Here, the data is transformed back to parallel electrical signals that are stored in one of several banks of FIFO memories. The FIFOs place their data onto the VME backplane to a VME Buffer Driver (VBD) which stores the event data in buffers for eventual readout over a thirty-two signal ribbon cable to the Level Two Computers and subsequent tape storage.

Control and sequencing of the whole operation starts with the Silicon Acquisition/Readout Controller (SARC)² working in tandem with the D0 Clock System. The SARC resides in the same VME crate as the SARs, and transforms signals from the Trigger System into control codes distributed to the various Port Cards via optical fibers operating at 53Mb/s. It is through these control codes that data taking operations such as data-acquisition, digitization, readout, and various resets can be carried out. The Port Card receives the control codes and manipulates the SVX-II chips in the proper way to effect proper data taking.

There will be a total of *about* 700,000 channels, which translates into about 5580 SVX-II chips, 66 to 100 Port Cards, 66 to 100 SARs, and four to eight SARCs.

2 GENERAL DESCRIPTION OF PORT CARD

Each Port Card houses two or possibly three "Port Card Equivalents", and resides in a 9U by 280 mm crate underneath the detector in an area known as the "Platform". There are four such crates, each crate housing about twenty Port Cards. A Port Card Equivalent consists of the following major parts (see Fig. 1.):

- Optical to electrical converters and associated circuitry for reception of the control codes
- High current drivers and Futurebus transceivers to interface to each of four flex cables; each cable connects to one HDI.
- A bus interface to be used for providing a serial data stream necessary to download essential programmable information into the SVX-II chips.

¹Daniel Mendoza, Silicon Acquisition and Readout Module, D0 Engineering Note 3823.112-EN-394, Fermilab, May 2, 1994.

²Mark Baert, Silicon Acquisition and Readout Controller Module, D0 Engineering Note 3823.112-EN-395, Fermilab, May 2, 1994.

- A stable analog voltage source to be used for a calibration pulse for the SVX-II chips.
- Two Hewlett-Packard G-Link transmitters for real-time serialization of the SVX-II parallel data. Each transmitter has a sixteen bit wide input bus.
- Two Finisar Electrical to Optical converters for converting the output of the G-Link into 1300nm laser light for transmission through the multimode fiber back to the SAR.
- TTL to ECL translators for interfacing to the G-Link chips.
- An Altera EPM7128 EPLD for the logic necessary to interpret the control codes and create the signals necessary to control the SVX-II chips, G-Links, Transceivers, counter and tri-state buffers.

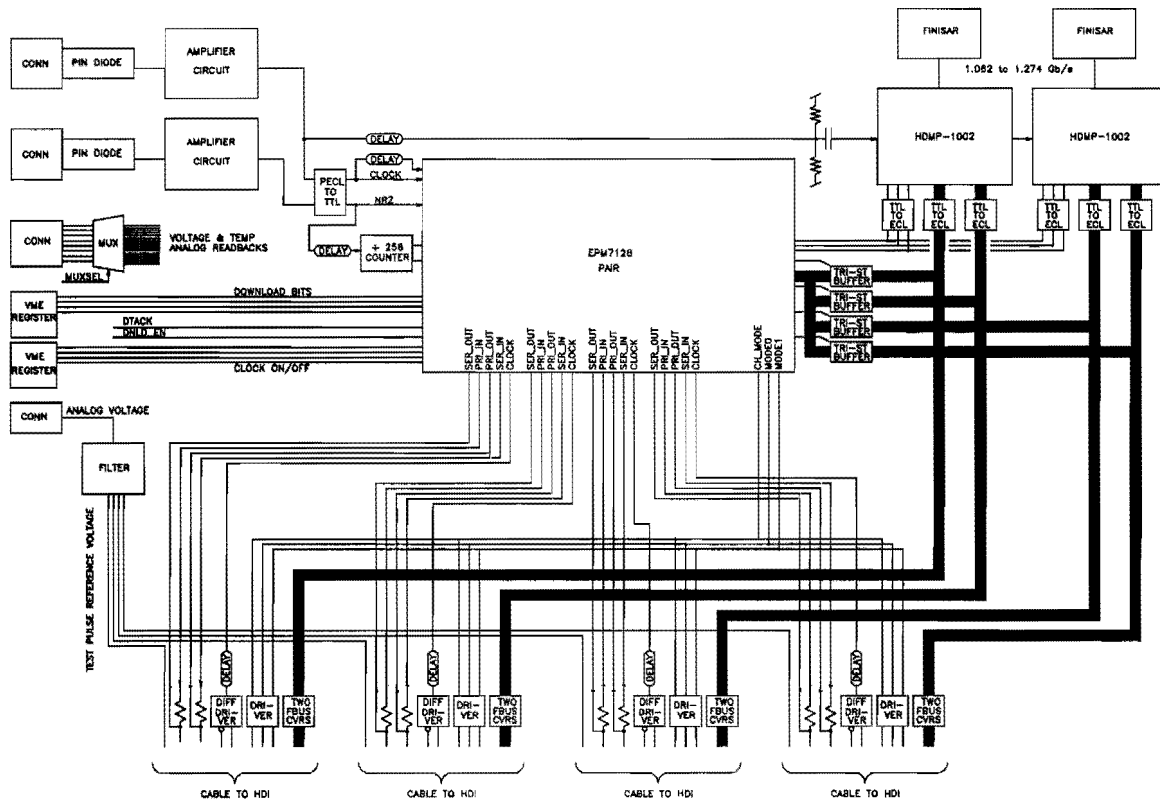


Figure 1. Block diagram of the Port Card Equivalent

- A divide-by-256 counter to gauge the number of clock pulses sent to the SVX-II chips during the digitization mode.
- Four tri-state buffers to isolate the EPLD from the data bus when the SVX-II chips are transmitting data to the G-Links during readout mode.
- Several lumped-constant delay lines for carefully controlling the phasing of the clocks going to the G-Links, counter, EPLD, and SVX-II chips.
- One programmable silicon delay line for controlling the pulse width of the preamp reset signal sent to the SVX-II chips during data acquisition.

3 DATA STRUCTURE OF THE CONTROL LINK

Most of the operations of the Port Card is controlled by the Control Link. Originating in the SARC and routed through the SARs to the Port Cards, the control link is a serial bit stream implemented in two optical fibers. One optical fiber carries a 53MHz square wave. The other fiber carries a 53 Mbit/sec NRZ pulse train. The Port Card uses the square wave as a clock to latch the data on the NRZ line. The SAR can adjust the relative phasing of the two signals to suit the Port Card's set up and hold times.

The serial bit stream is segmented into constant, back-to-back, seven-bit packets. This corresponds to the minimum bunch spacing of 132 ns in the Tevatron, so the control link is synchronous with beam crossings. The first bit of each packet is a Framing Bit which is always one, and is used by the Port Card to maintain synchronization of the link. The second bit is called the Crossing Bit; this bit is high in every packet that may contain a beam crossing. For bunch spacing of 132ns, the crossing bit will be one in every packet. For bunch spacing of 395 ns, the crossing bit will be one in every third packet. The Port Card, upon receiving the Crossing bit, advances the pipeline by one and discharges the next SVX-II pipeline capacitor just prior to the crossing. The next four bits are encoded to produce one of sixteen possible control codes; the first bit to arrive is the MSB. These are the codes originating ultimately in the SARC for SVX-II control. The final bit of the packet is the Parity bit.

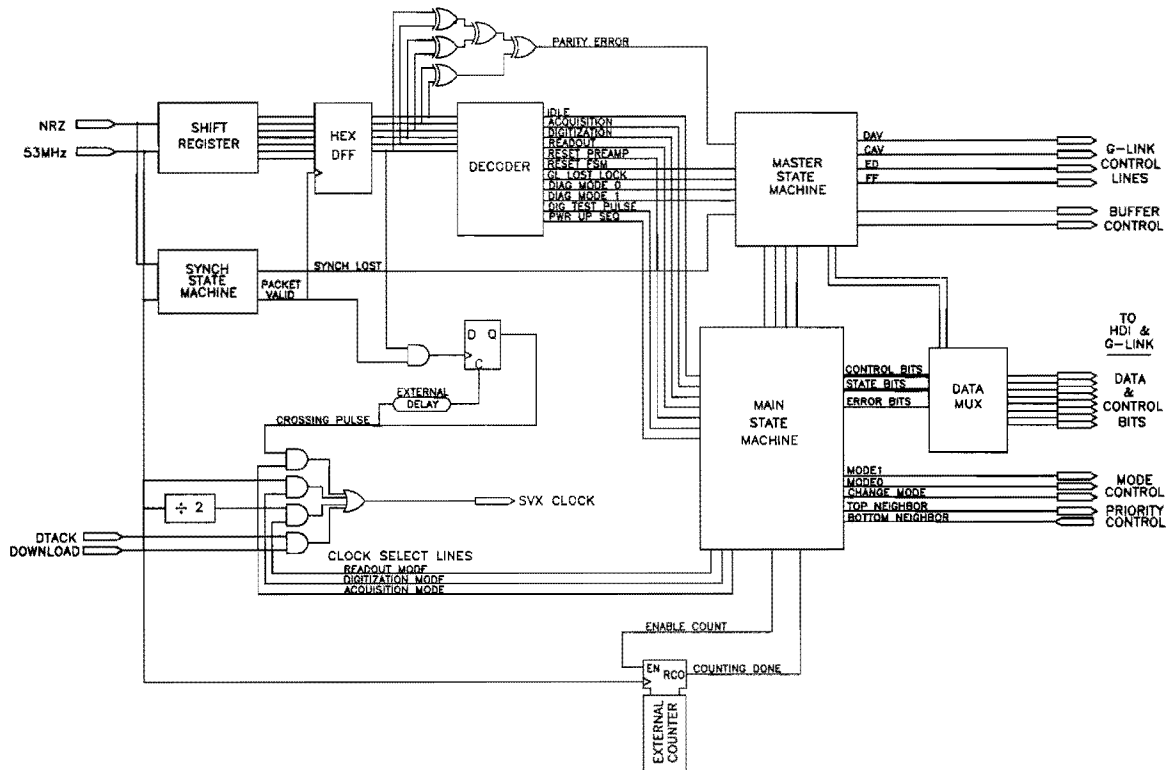


Figure 2. Block diagram of the contents of the EPLD.

4 EPLD

The EPLD consists of the following circuits (see Fig. 2):

- Finite state machine that checks for the Framing Bit every seventh bit and if missing, waits for a string of sixteen zeroes in a row before interpreting the next one as a new Framing Bit.
- Seven bit shift register to put the seven bits of the packet in parallel.
- Hex D flip-flop which registers the last six bits of the packet based on a clock derived from the Framing bit.
- A Parity checking circuit
- A decoder circuit that decodes the four encoded control bits into a maximum of sixteen separate control lines.
- A pulse forming circuit for precise width control of the SVX-II preamp reset pulse. This circuit makes use of an external programmable silicon delay line.
- A small "Master" state machine which handles the resynchronization protocol of the G-Link, puts the EPLD in a diagnostic mode when summoned to do so, and controls some of the resetting functions on the chip.
- A larger "Main" state machine which manipulates the control lines of the SVX-II chips, controls the various clocking of the SVX-IIs, the digitization clock counter, and the tri-state buffers and transceivers, all based on the control codes coming from the decoder.
- A data multiplexer that selects various data to be output to either the cables to the HDI or to the G-Link. This data can be one of the following:
 - ⇒ Eight bits to each HDI for control of the SVX-II chips during all modes but readout
 - ⇒ Eight bits to each half of a G-Link just prior to data readout for placing a Port Card identification code into the data stream.
 - ⇒ Seven bits to each half of a G-Link during a special diagnostic mode whereby the state variables of the main state machine in the Port Card is transmitted to the SAR.
 - ⇒ Two bits to be sent through the G-Link when either a Parity error occurs or the Control Link has lost synchronization.
- A clock multiplexer for selecting one of several clocks to be sent to each HDI.

4.1 Synchronization of the Control Link

As mentioned, the control link consists of a continuous stream of data bits divided into seven bit "frames" with the first bit having the sole function of delineating the frames by always being high. One state machine in the Port Card EPLD performs the function of checking for the presence of this Framing bit as well as providing timing pulses used to latch the control bits. The state machine is synchronous with the 53MHz clock and consists of two loops. The first loop has seven states and is active when synchronization exists. The machine runs through the six states prior to the Framing bit and enters the seventh state just prior to the beginning of the expected Framing bit. If the next clock registers a one on the NRZ line, the machine loops back to the first state and remains in the loop. If, however, the NRZ line registers zero, the Framing bit is missing, and the machine branches to a second loop where resynchronization can take place.

When the second loop is entered a control code is sent through the G-LINKS back to the SAR requesting a resynchronization cycle. The SAR immediately sends a string of sixteen zeroes on NRZ and then begins sending meaningful frames again. In the meantime, the Port Card has passed through thirty sequential states and then enters a state which will put the machine back into the first loop only when a one is registered on the

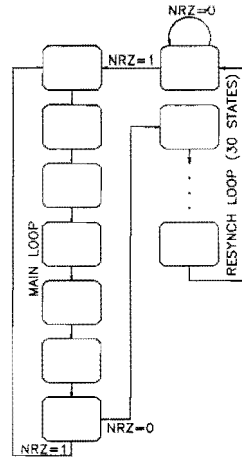


Figure 3. Synchronization state machine

NRZ line. The thirty states are necessary to account for the time of flight of the "loss of synch" message to reach the SAR and for the subsequent string of zeroes to arrive at the Port Card (all intervening ones are not to be mistaken as a Framing bit). This synchronization sequence is also used immediately after power-up.

This scheme is simple but not foolproof since the link may be out of synch and still produce a one every seventh bit when the state machine expects one. With changing control codes, though, loss of synch is likely to be detected quite quickly.

4.2 Decoding of the Control Codes

A shift register parallelizes the serial data stream and sends six lines to a hex-D flip-flop. When the packet is shifted into the right bins, the Synch state machine sends a clock and registers the data in the packet every 132 ns. The four control codes are decoded as follows:

<u>BIT CODE</u>	<u>FUNCTION</u>
0000	IDLE
0001	ACQUISITION
0011	DIGITIZATION
0010	READOUT
0101	RESET PREAMP
0110	RESET PORT CARD
1011	DIAGNOSTIC MODE 1
1001	DIAGNOSTIC MODE 0
0111	DIGITIZE TEST PULSE
1100	POWER UP SEQUENCE
1111	G-LINK LOSS OF LOCK

Not all codes are presently assigned or needed.

If the crossing bit is set during acquisition then the SVX clock line will be pulsed accordingly. Also, the bits of the frame are sent into a simple parity checker so that the master state machine may notify the SAR should a parity error occur.

4.3 MASTER STATE MACHINE

This consists of a small state machine responsible for reporting errors to the SAR and for putting the Port Card in diagnostic mode.

4.3.1 Parity Error

Should a parity error occur and the SVXs are not sending data to the SARs, a control word will be sent up the G-LINK indicating that a parity error has occurred.

4.3.2 Loss of Synch

If the Synch state machine does not detect a one when it expects a Framing bit, the Master state machine will send a control word through the G-LINK indicating loss of synch. The SAR will then respond accordingly to re-establish synch of the low speed link.

If the SVXs are sending data in readout mode, the Port Card will not be sent the control word until readout is complete.

4.3.3 Loss of G-LINK Lock

If the SAR G-LINK receiver loses data frame lock, the SAR will send constant "loss of lock" codes down the low speed link until lock is re-established. Upon receiving the loss of lock code, the Master state machine manipulates the appropriate G-LINK transmitter inputs to effect a relocking of the G-LINK transmitter/receiver pair. The re-lock operation takes approximately one millisecond.

4.3.4 Diagnostic Mode

When the SAR sends a diagnostic mode code, the operation of two HDIs is sacrificed to provide one G-LINK with constant, real time access to the state variables of the Main state machine. The main state machine operates at 53MHz as does the output of the G-LINK receiver on the SAR. The result is access to the operation of the Port Card as viewed by a virtual remote logic analyzer. A Reset code clears this mode.

4.4 MAIN STATE MACHINE

The Main state machine directly manipulates the SVX-II control lines and selects the proper clock to be sent to the chips. It controls the MODE0, MODE1, and CH_MODE lines which determine whether the chips are in initialize, acquisition, digitization, or readout mode.³ It also controls the eight bit data bus which acts as a set of control lines during all modes but readout. The timing requirements set forth in the SVX-II Guide are met by adding states, each state contributing 18.8 ns to the delay. The following are brief descriptions of the branches of the state machine; the control lines are otherwise manipulated as specified in the SVX-II Guide.

³Ray Yarema, A Beginners Guide to the SVX-II, D0 Engineering Note 3823.112-EN-399, Fermilab, May 2, 1994

4.4.1 Initialization

Most of the action occurring during initialization is a result of the download (initialization) bus described below. If a download is to occur, a signal from the download bus puts the main state machine in initialize mode. Downloading may then take place.

4.4.2 Acquisition

When an acquisition code is received, the output of the crossing circuit is routed through the clock multiplexer to advance the pipeline and place charge on a different capacitor for each crossing. The state machine also looks for either a digitize command or a reset_preamp command. The reset command sends the machine into a loop assuring adequate time for proper resetting of the preamps and is expected to occur only during large beam gaps. The digitize command halts the pipeline clock and sends the machine through a series of states accomplishing a pipeline readout.

4.4.3 Digitization

The machine passes directly from the pipeline readout sequence into the digitization sequence where the external counter is enabled and the 53MHz clock is then routed to both the counter and the SVX-IIs. When the counter reaches the proper value (which is settable), the state machine proceeds to the readout sequence.

4.4.4 Readout

During readout the clock being sent to the SVX-IIs is the 53MHz clock divided by two. The states are adjusted so that two clocks prior to the first SVX data appearing at the inputs of the G-LINK, the Port Card places two bytes of Port Card Identification into each half G-LINK. The SVX data then follows and when the state machine detects the Priority Out signals from all four HDIs the machine goes into Idle.

4.4.5 Power Up

The SVX-II requires a special power up sequence in order to put the chip in a known state. To accomplish this the Port Card's Main state machine will provide this sequence when a power up code is received, followed by an idle code.

4.4.6 Digitize Test Pulse

When the digitize test pulse code is received, the machine pulses the control line that injects the test voltage into the SVX. The machine is then sent into the normal digitize mode and subsequent readout mode.

5 LOW SPEED FIBER OPTIC LINK⁴

A multimode graded index 62.5/125 um fiber carries each of the two signals comprising the Control link from a SAR to a Port Card. On the Port Card, a PIN diode/transimpedance amplifier HFBR-2416 converts the light into PECL levels and a PECL to TTL translator prepares the signal for the inputs of the EPLD.

⁴Jorge Amaral, Low Speed Fiber Optic Link, D0 Engineering Note 3823.112-EN-397, Fermilab, May 2, 1994.

6 HIGH SPEED FIBER OPTIC LINK⁵

An SVX-II chip sends eight bits of data or channel identification over its HDI every 18.83 ns. With four HDIs per Port Card Equivalent, the number of conductors to make a link to the SAR would be prohibitive. Two Hewlett Packard HDMP-1000 G-Link chipsets are therefore employed which, together with two Finisar laser modules, reduce this cabling requirement to two multimode, graded index fibers with cladding/core diameters of 62.5/125 microns.

The G-Link transmitter takes the parallel data from two HDIs present on its sixteen input lines, adds four bits for error checking protocol, and serializes the result onto a differential pair of output pins. The input strobe frequency is therefore multiplied by twenty to get the output bit rate of the differential pair. Since the SVX-II data changes every 18.83 ns (53.104 MHz), the resulting bit rate out of the differential pair is 1.062 Gb/s. Input logic levels are ECL compatible.

7 INITIALIZING THE CHIPS

Each chip requires 182 bits to download such things as test pulse mask, polarity, pipeline length, preamp bandwidth, etc. Since the chips are connected in series, and there can be as many as ten chips on one HDI, there can be as many as 1820 bits to serially place onto each HDI. There are two options to implement downloading of the SVX-II chip strings.

7.1 VME Interface

The VME interface is used exclusively to produce a serial bit stream to download the SVX-II chips. At a VME bus rate of 200 ns per transfer it would take about 400 μ s to download one HDI. Since, however, the VMEbus is parallel, more than one HDI can be downloaded at one time. If there are two Port Card Equivalents per board the resulting eight HDIs could be downloaded concurrently using one VME address. Software will take care of serializing and collating the data to be downloaded in this fashion. It then simply becomes a matter of using DTACK as a clock in download mode and the task is accomplished with very little hardware.

7.2 1553 Interface

An alternate method of accomplishing initialization uses the MIL-STD-1553 system which already exists on the Platform to some extent. MIL-STD-1553 (Aircraft Internal Time Division Command/Response Multiplex Data Bus) consists of a Bus Controller external to the Platform, a bus which is a shielded twisted pair, and a Remote Terminal which might be allocated one per Port Card.

According to 1553 protocol, the data is transmitted serially through the twisted pair at 1Mb/s with twenty consecutive bits making up one word. An initial Control Word is sent which contains addressing, read/write information, and number of Data Words (up to 32) to follow. The addressing scheme uses five bits for "Remote Terminal" address and five bits for "subaddresses" within each Remote Terminal. There can be a maximum of 31 Remote Terminals and 30 subaddresses. Data words follow the Control Word and the

⁵Jorge Amaral, High Speed Optical Link, D0 Engineering Note 3823.112-EN-398, Fermilab, May 2, 1994.

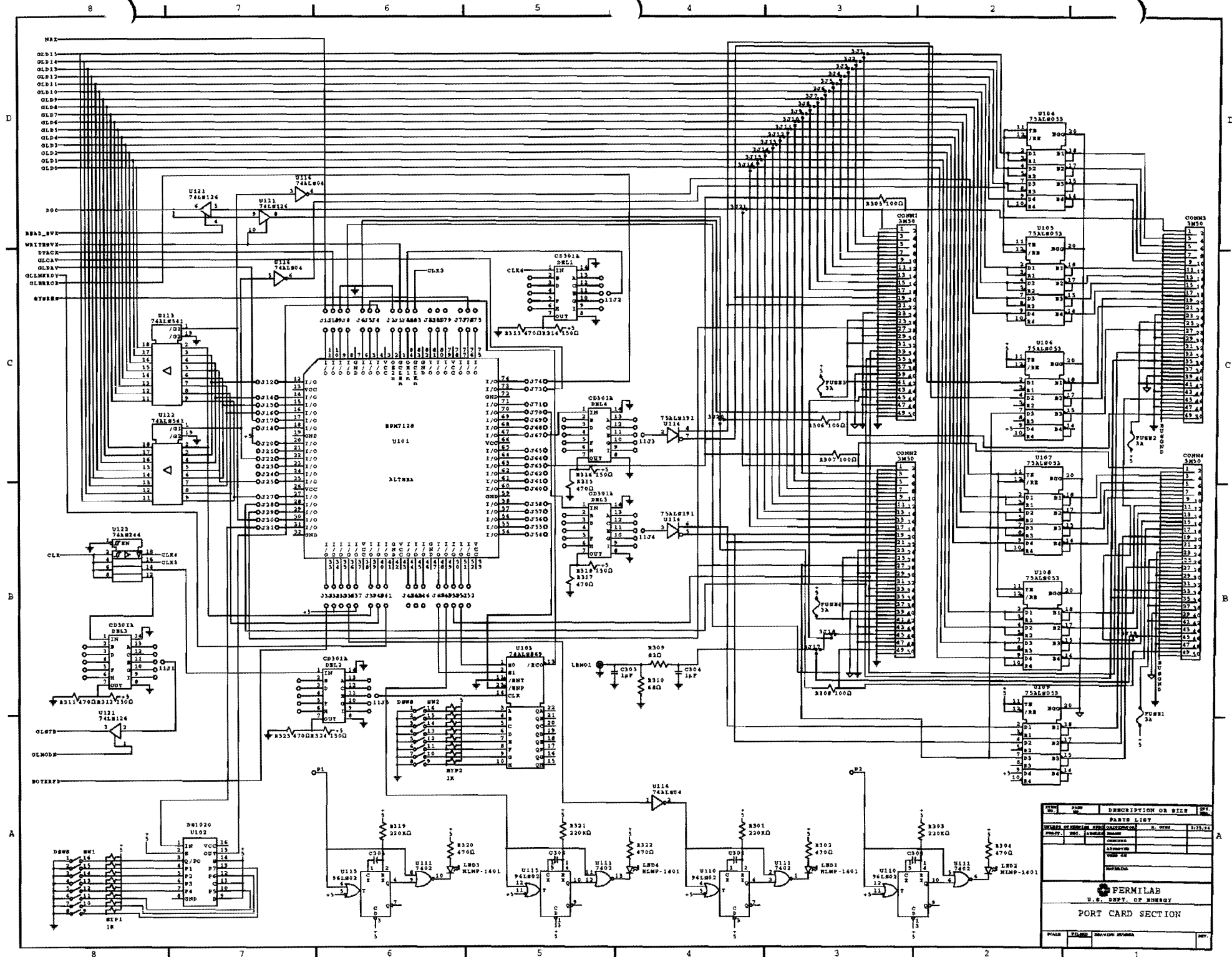
Remote Terminal decoding the address receives sixteen data bits in each twenty bit Data Word sent. After 32 words another such "transfer" may occur.

The planned scheme will put one Remote Terminal chip onto each Port Card. Each Port Card Equivalent will be a subaddress, and data will be taken off one of the sixteen data bits one transfer at a time. The download will propagate serially from one HDI to the next until all chips on one Port Card Equivalent are filled. A maximum of forty chips per Port Card Equivalent at 182 channels per chip gives 7280 total bits. At 32 bits per transfer, the maximum number of transfers per Port Card Equivalent is 228. This takes approximately 150ms. With 31 Remote Terminals possible on a Bus Controller, the maximum download time then becomes about 4.7 sec.

In either option, readback of downloaded data requires performing a destructive read. After the read, therefore, another write operation must take place.

APPENDIX A

PORT CARD PROTOTYPE SCHEMATIC



REV.	DATE	DESCRIPTION OR SIZE	BY
1	1/15/74	PORT CARD SECTION	...

PARTS LIST		QTY	REMARKS
U101	74ALS124	1	
U102	74ALS124	1	
U103	74ALS124	1	
U104	74ALS124	1	
U105	74ALS124	1	
U106	74ALS124	1	
U107	74ALS124	1	
U108	74ALS124	1	
U109	74ALS124	1	
U110	74ALS124	1	
U111	74ALS124	1	
U112	74ALS124	1	
U113	74ALS124	1	
U114	74ALS124	1	
U115	74ALS124	1	
U116	74ALS124	1	
U117	74ALS124	1	
U118	74ALS124	1	
U119	74ALS124	1	
U120	74ALS124	1	
U121	74ALS124	1	
U122	74ALS124	1	
U123	74ALS124	1	
U124	74ALS124	1	
U125	74ALS124	1	
U126	74ALS124	1	
U127	74ALS124	1	
U128	74ALS124	1	
U129	74ALS124	1	
U130	74ALS124	1	
U131	74ALS124	1	
U132	74ALS124	1	
U133	74ALS124	1	
U134	74ALS124	1	
U135	74ALS124	1	
U136	74ALS124	1	
U137	74ALS124	1	
U138	74ALS124	1	
U139	74ALS124	1	
U140	74ALS124	1	
U141	74ALS124	1	
U142	74ALS124	1	
U143	74ALS124	1	
U144	74ALS124	1	
U145	74ALS124	1	
U146	74ALS124	1	
U147	74ALS124	1	
U148	74ALS124	1	
U149	74ALS124	1	
U150	74ALS124	1	
R1	220K	1	
R2	220K	1	
R3	220K	1	
R4	220K	1	
R5	220K	1	
R6	220K	1	
R7	220K	1	
R8	220K	1	
R9	220K	1	
R10	220K	1	
R11	220K	1	
R12	220K	1	
R13	220K	1	
R14	220K	1	
R15	220K	1	
R16	220K	1	
R17	220K	1	
R18	220K	1	
R19	220K	1	
R20	220K	1	
R21	220K	1	
R22	220K	1	
R23	220K	1	
R24	220K	1	
R25	220K	1	
R26	220K	1	
R27	220K	1	
R28	220K	1	
R29	220K	1	
R30	220K	1	
R31	220K	1	
R32	220K	1	
R33	220K	1	
R34	220K	1	
R35	220K	1	
R36	220K	1	
R37	220K	1	
R38	220K	1	
R39	220K	1	
R40	220K	1	
R41	220K	1	
R42	220K	1	
R43	220K	1	
R44	220K	1	
R45	220K	1	
R46	220K	1	
R47	220K	1	
R48	220K	1	
R49	220K	1	
R50	220K	1	
R51	220K	1	
R52	220K	1	
R53	220K	1	
R54	220K	1	
R55	220K	1	
R56	220K	1	
R57	220K	1	
R58	220K	1	
R59	220K	1	
R60	220K	1	
R61	220K	1	
R62	220K	1	
R63	220K	1	
R64	220K	1	
R65	220K	1	
R66	220K	1	
R67	220K	1	
R68	220K	1	
R69	220K	1	
R70	220K	1	
R71	220K	1	
R72	220K	1	
R73	220K	1	
R74	220K	1	
R75	220K	1	
R76	220K	1	
R77	220K	1	
R78	220K	1	
R79	220K	1	
R80	220K	1	
R81	220K	1	
R82	220K	1	
R83	220K	1	
R84	220K	1	
R85	220K	1	
R86	220K	1	
R87	220K	1	
R88	220K	1	
R89	220K	1	
R90	220K	1	
R91	220K	1	
R92	220K	1	
R93	220K	1	
R94	220K	1	
R95	220K	1	
R96	220K	1	
R97	220K	1	
R98	220K	1	
R99	220K	1	
R100	220K	1	
C1	100P	1	
C2	100P	1	
C3	100P	1	
C4	100P	1	
C5	100P	1	
C6	100P	1	
C7	100P	1	
C8	100P	1	
C9	100P	1	
C10	100P	1	
C11	100P	1	
C12	100P	1	
C13	100P	1	
C14	100P	1	
C15	100P	1	
C16	100P	1	
C17	100P	1	
C18	100P	1	
C19	100P	1	
C20	100P	1	
C21	100P	1	
C22	100P	1	
C23	100P	1	
C24	100P	1	
C25	100P	1	
C26	100P	1	
C27	100P	1	
C28	100P	1	
C29	100P	1	
C30	100P	1	
C31	100P	1	
C32	100P	1	
C33	100P	1	
C34	100P	1	
C35	100P	1	
C36	100P	1	
C37	100P	1	
C38	100P	1	
C39	100P	1	
C40	100P	1	
C41	100P	1	
C42	100P	1	
C43	100P	1	
C44	100P	1	
C45	100P	1	
C46	100P	1	
C47	100P	1	
C48	100P	1	
C49	100P	1	
C50	100P	1	
C51	100P	1	
C52	100P	1	
C53	100P	1	
C54	100P	1	
C55	100P	1	
C56	100P	1	
C57	100P	1	
C58	100P	1	
C59	100P	1	
C60	100P	1	
C61	100P	1	
C62	100P	1	
C63	100P	1	
C64	100P	1	
C65	100P	1	
C66	100P	1	
C67	100P	1	
C68	100P	1	
C69	100P	1	
C70	100P	1	
C71	100P	1	
C72	100P	1	
C73	100P	1	
C74	100P	1	
C75	100P	1	
C76	100P	1	
C77	100P	1	
C78	100P	1	
C79	100P	1	
C80	100P	1	
C81	100P	1	
C82	100P	1	
C83	100P	1	
C84	100P	1	
C85	100P	1	
C86	100P	1	
C87	100P	1	
C88	100P	1	
C89	100P	1	
C90	100P	1	
C91	100P	1	
C92	100P	1	
C93	100P	1	
C94	100P	1	
C95	100P	1	
C96	100P	1	
C97	100P	1	
C98	100P	1	
C99	100P	1	
C100	100P	1	

FERMILAB
 U.S. DEPT. OF ENERGY
PORT CARD SECTION

APPENDIX B
PORT CARD EPLD DESIGN FILES

SUBDESIGN 'FSM_SYNC'

```
(
  NRZ, CLK, RESET      : INPUT;
  SYNC, LAT            : OUTPUT;
  LOOP[4..0]          : OUTPUT;
)
```

VARIABLE

```
synch : MACHINE of bits (loop[4..0])
WITH STATES (s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, sA, sB, sC, sD, sE, sF,
```

```
          s10, s11, s12, s13, s14, s15, s16, s17, s18, s19, s1A, s1B, s1C,
s1D, s1E, s1F);
```

BEGIN

```
synch.clk = CLK;
synch.reset = RESET;
```

CASE SYNCH IS

```
WHEN s0 => IF NRZ THEN SYNCH = s1; ELSE SYNCH = s0; END IF; SYNC = GND; LAT = GND;
```

```
WHEN s1 => SYNCH = s3;          SYNC = VCC; LAT = GND;          %Working Loop%
WHEN s3 => SYNCH = s2;          SYNC = VCC; LAT = GND;
WHEN s2 => SYNCH = s6;          SYNC = VCC; LAT = GND;
WHEN s6 => SYNCH = s7;          SYNC = VCC; LAT = GND;
WHEN s7 => SYNCH = s5;          SYNC = VCC; LAT = GND;
WHEN s5 => SYNCH = sD;         SYNC = VCC; LAT = GND;
WHEN sD => IF NRZ THEN SYNCH = s1; ELSE SYNCH = sC; END IF; SYNC = VCC; LAT = VCC;
```

```
WHEN sC => SYNCH = sE;          SYNC = GND; LAT = GND;          %Resynch Loop%
WHEN sE => SYNCH = sF;          SYNC = GND; LAT = GND;
WHEN sF => SYNCH = sB;          SYNC = GND; LAT = GND;
WHEN sB => SYNCH = s1B;         SYNC = GND; LAT = GND;
WHEN s1B => SYNCH = s1F;        SYNC = GND; LAT = GND;
WHEN s1F => SYNCH = s1E;        SYNC = GND; LAT = GND;
WHEN s1E => SYNCH = s1C;        SYNC = GND; LAT = GND;
WHEN s1C => SYNCH = s1D;        SYNC = GND; LAT = GND;
WHEN s1D => SYNCH = s15;        SYNC = GND; LAT = GND;
WHEN s15 => SYNCH = s17;        SYNC = GND; LAT = GND;
WHEN s17 => SYNCH = s13;        SYNC = GND; LAT = GND;
WHEN s13 => SYNCH = s12;        SYNC = GND; LAT = GND;
WHEN s12 => SYNCH = s11;        SYNC = GND; LAT = GND;
WHEN s11 => SYNCH = s19;        SYNC = GND; LAT = GND;
WHEN s19 => SYNCH = s18;        SYNC = GND; LAT = GND;
WHEN s18 => SYNCH = s10;        SYNC = GND; LAT = GND;
WHEN s10 => SYNCH = s14;        SYNC = GND; LAT = GND;
WHEN s14 => SYNCH = s16;        SYNC = GND; LAT = GND;
WHEN s16 => SYNCH = s1A;        SYNC = GND; LAT = GND;
WHEN s1A => SYNCH = sA;         SYNC = GND; LAT = GND;
WHEN sA => SYNCH = s8;          SYNC = GND; LAT = GND;
WHEN s8 => SYNCH = s0;          SYNC = GND; LAT = GND;
WHEN s9 => SYNCH = sC;          SYNC = GND; LAT = GND;
WHEN s4 => SYNCH = sC;          SYNC = GND; LAT = GND;
```

END CASE;

END;

TITLE "7-bit Packet message decoder";

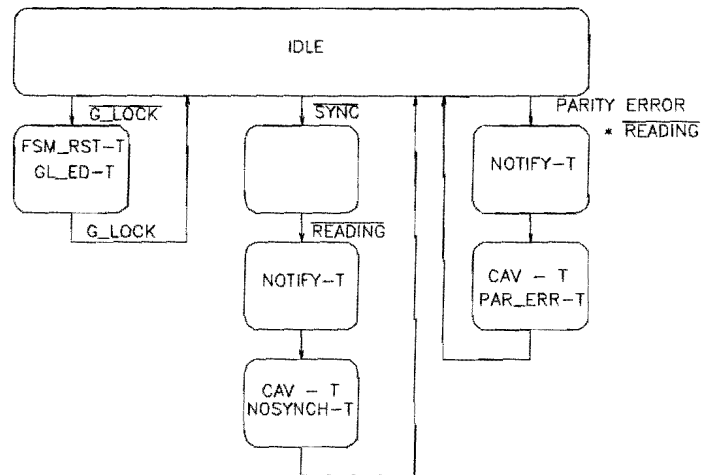
SUBDESIGN decoder

```
BIT6, BIT5, BIT4, BIT3 : INPUT;
IDLE, ACQUISITION, DIGITIZATION, READOUT, RESET PREAMP, RESET,
G LINK LOCK TOGGLE, LOGIC ANALYZER 1, LOGIC ANALYZER 0,
DIGITIZE_TEST_PULSE, READ_STATUS, PWR_UP :OUTPUT;
```


)
BEGIN

```
Idle           = !bit3 & !bit4 & !bit5 & !bit6;      % 0 0 0 0 %
Acquisition    = !bit3 & !bit4 & !bit5 & bit6;       % 0 0 0 1 %
Digitization   = !bit3 & !bit4 & bit5 & bit6;        % 0 0 1 1 %
Readout        = !bit3 & !bit4 & bit5 & !bit6;       % 0 0 1 0 %
Reset Preamp   = !bit3 & bit4 & !bit5 & bit6;        % 0 1 0 1 %
Reset          = !bit3 & bit4 & bit5 & !bit6;        % 0 1 1 0 %
Logic Analyzer 1 = bit3 & !bit4 & bit5 & bit6;       % 1 0 1 1 %
Logic Analyzer 0 = bit3 & !bit4 & !bit5 & bit6;       % 1 0 0 1 %
Digitize Test Pulse = !bit3 & bit4 & bit5 & bit6;    % 0 1 1 1 %
Read Status    = !bit3 & bit4 & !bit5 & !bit6;       % 0 1 0 0 %
Pwr Up         = bit3 & bit4 & !bit5 & !bit6;        % 1 1 0 0 %
G_Link_Lock_Toggle = bit3 & bit4 & bit5 & bit6;      % 1 1 1 1 %
```

END;



REV.	DESCRIPTION	DRAWN	DATE
		APPD.	DATE

ITEM	PART NO.	DESCRIPTION OR SIZE	QTY.
PARTS LIST			
UNLESS OTHERWISE SPEC.		ORIGINATOR	M. J. LITES
DATE		6/2/84	
FRACIONS	DECIMALS	ANGLES	DRAWN
			CHECKED
			APPROVED
USED ON			
MATERIAL-			
MSTRSEM			
 FERMI NATIONAL ACCELERATOR LABORATORY UNITED STATES DEPARTMENT OF ENERGY			
PORT CARD MASTER STATE MACHINE			
SCALE	FILED	DRAWING NUMBER	REV.

SUBDESIGN 'MASTER' %CONTROLS OUTPUTS BASED ON INPUT CODES%

```
(
PARERR, RESET, G_LOCK, LA1, LA0, 53MHz, SYNC, BUS_EN, READING, VMERST :INPUT;
BUS_EN1, BUS_EN0, CAV, ED, FF, NOTIFY, _LA, NOSYNC, PAR_ERR, RST :OUTPUT;
)
```

VARIABLE

```
MASTER: MACHINE OF BITS (MBIT[3..0])
      with states (s0, s1, s2, s3, s4, s5, s6, s7, s8);
```

```
LOG1: MACHINE OF BITS (L0BIT) with states(s9, sA);
LOG0: MACHINE OF BITS (L1BIT) with states(sB, sC);
```

```
LA0, LA1 :NODE;
```

BEGIN

DEFAULTS

```
CAV=GND; ED=VCC; FF=GND;
NOTIFY=GND; NOSYNC=GND; PAR_ERR=GND; RST=GND; _LA0=GND; _LA1=GND;
```

END DEFAULTS;

```
MASTER.clk = 53MHz;
MASTER.reset = VMERST;
```

```
LOG1.clk = 53MHz;
LOG1.reset = VMERST;
```

```
LOG0.clk = 53MHz;
LOG0.reset = VMERST;
```

```
BUS_EN1 = LA1 # BUS_EN; % Also connected /RE of Turbotransceivers %
BUS_EN0 = _LA0 # BUS_EN; % Also connected /RE of Turbotransceivers %
```

```
_LA = _LA0 # _LA1;
```

CASE LOG0 IS %Latches Logic Analyzer 0 mode%

```
WHEN sB =>
  IF LA0 THEN LOG0 = sC; END IF;
WHEN sC =>
  IF RESET THEN LOG0 = sB; END IF; _LA0=VCC;
END CASE;
```

CASE LOG1 IS %Latches Logic Analyzer 1 mode%

```
WHEN s9 =>
  IF LA1 THEN LOG1 = sA; END IF;
WHEN sA =>
  IF RESET THEN LOG1 = s9; END IF; LA1=VCC;
END CASE;
```

```
CASE MASTER IS
  WHEN s0 =>
    IF G LOCK THEN MASTER = s1;           %If G Link lost lock%
    ELSIF !SYNC & !READING THEN MASTER = s2; %If Low Speed link loses synch%
    ELSIF PARERR & !READING THEN MASTER = s5; %If a Parity error occurs%
    END IF;

  WHEN s5 => MASTER = s6;           NOTIFY=VCC;
  WHEN s6 => MASTER = s7;           NOTIFY=VCC; PAR_ERR=VCC;
  WHEN s7 => MASTER = s0;           NOTIFY=VCC; CAV=VCC; PAR_ERR=VCC;

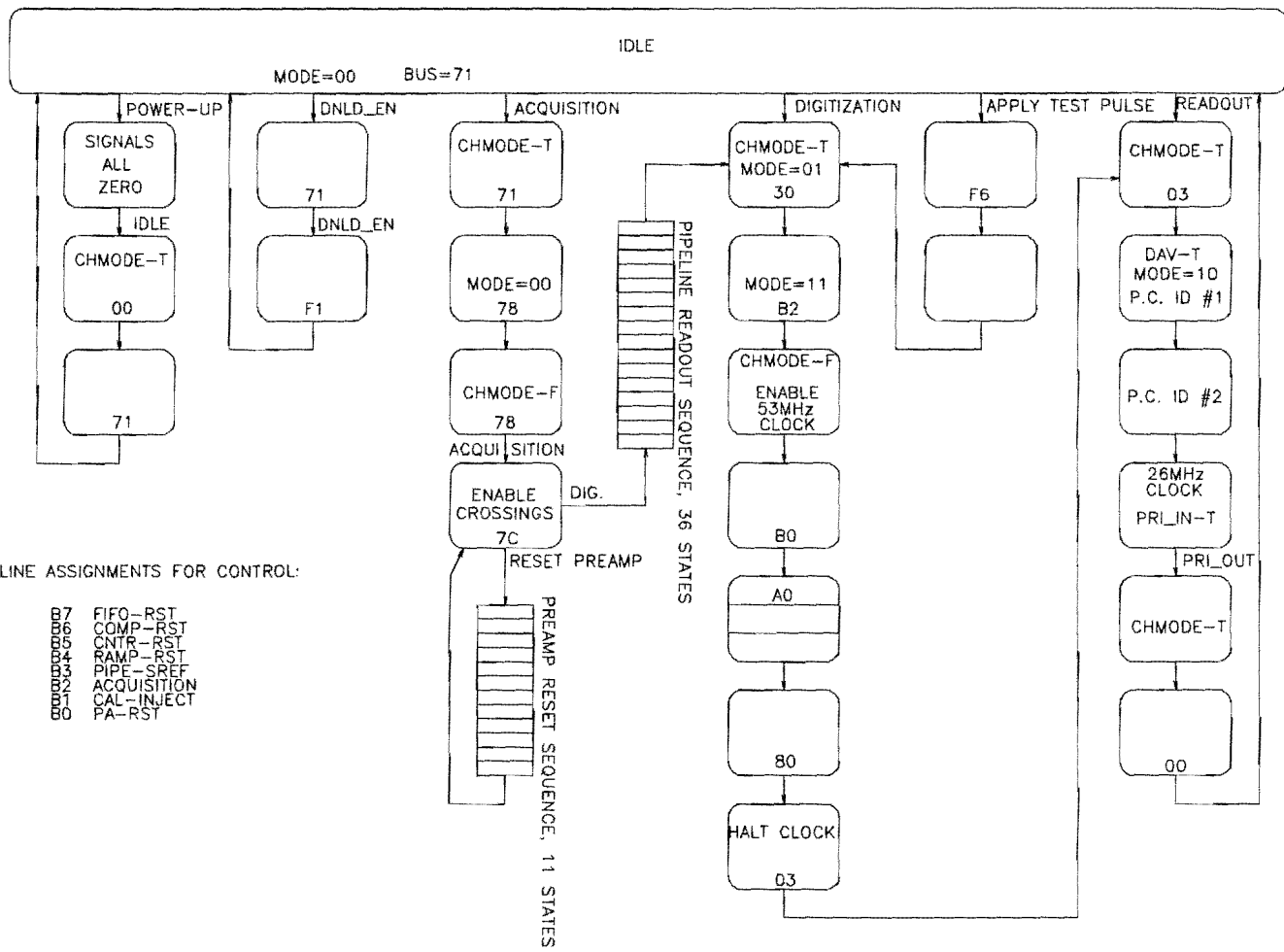
  WHEN s2 => MASTER = s3;           NOTIFY=VCC;
  WHEN s3 => MASTER = s4;           NOTIFY=VCC; NOSYNC=VCC;
  WHEN s4 =>                       NOTIFY=VCC; CAV=VCC; NOSYNC=VCC;
    IF SYNC THEN MASTER = s0;
    ELSIF !SYNC THEN MASTER = s4; END IF;

  WHEN s1 =>                       RST=VCC; ED=GND;
    IF !G LOCK THEN MASTER = s0;
    ELSIF G LOCK THEN MASTER = s1; END IF;

END CASE;

END;
```

REV.	DESCRIPTION	DATE



BUS LINE ASSIGNMENTS FOR CONTROL:

- B7 FIFO-RST
- B6 COMP-RST
- B5 CNTR-RST
- B4 RAMP-RST
- B3 PIPE-SREF
- B2 ACQUISITION
- B1 CAL-INJECT
- B0 PA-RST

ITEM	PARTY NO.	DESCRIPTION OR SIZE	QTY.
PARTS LIST			
UNLESS OTHERWISE SPEC.		ORIGINATOR	M. J. LITES 5/2/94
FRACTIONS	DECIMALS	ANGLES	DIMEN
USED ON			
MATERIAL-			
HDFSM			
FERMI NATIONAL ACCELERATOR LABORATORY UNITED STATES DEPARTMENT OF ENERGY			
PORT CARD			
MAIN STATE MACHINE			
SCALE	FILMED	DRAWING NUMBER	REV.

SUBDESIGN 'HDIFSM' % SVXII control %

```
(
  RST, IDLE, ACQ, DIG, READOUT, R_PRE, DIGTP, PWR_UP, 53MHZ, /RCO, PRI_OUT, DNLD_EN      : INPUT;

  BUS EN, DAV, D[7..0]                                                                : OUTPUT;
  HBIT6, HBIT5, HBIT4, HBIT3, HBIT2, HBIT1, HBIT0  : OUTPUT;
  MODE0, MODE1, CH MODE                                                                : OUTPUT;
  ENACRO, ENA53, ENA26, PRI_IN, SMCLK                                                : OUTPUT;
)
```

VARIABLE

```
HDI : MACHINE OF BITS (HBIT[6..0])
      with states ( s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, sA, sB, sC, sD, sE, sF,
                    s10, s11, s12, s13, s14, s15, s16, s17, s18, s19, s1A, s1B, s1C, s1D, s1E, s1F,
                    s20, s21, s22, s23, s24, s25, s26, s27, s28, s29, s2A, s2B, s2C, s2D, s2E, s2F,
                    s30, s31, s32, s33, s34, s35, s36, s37, s38, s39, s3A, s3B, s3C, s3D, s3E, s3F,
                    s40, s41, s42, s43, s44, s45, s46, s47, s48, s49, s4A, s4B, s4C, s4D, s4E, s4F,
                    s50, s51, s52, s53, s54, s55, s56, s57, s58, s59, s5A, s5B, s5C, s5D, s5E, s5F,
                    s60, s61, s62, s63, s64, s65, s66, s67, s68, s69, s6A, s6B, s6C, s6D, s6E, s6F,
                    s70, s71, s72, s73, s74, s75, s76, s77, s78, s79, s7A, s7B, s7C, s7D, s7E, s7F);
```

```
ENACRO : DFF;
ENCRO  : NODE;
ENA53  : DFF;
EN53   : NODE;
ENA26  : DFF;
EN26   : NODE;
```

BEGIN

DEFAULTS

```
BUS EN=VCC; D[7..0]=H"00"; MODE0=GND; MODE1=GND; CH MODE=GND;
PRI IN=GND; ENCRO=GND; ENA53=GND; ENA26=GND; DAV=GND; SMCLK=GND;
```

END DEFAULTS;

```
HDI.clk = 53MHz;
HDI.reset = RST;
```

CASE HDI IS

WHEN s0 =>

```
D[]=H"71"; %Idle State%
%Idle State%
```

```
IF DNLD EN THEN HDI = s43;
ELSIF PWR UP THEN HDI = s45;
ELSIF ACQ THEN HDI = s1;
ELSIF DIG THEN HDI = s4A;
ELSIF READOUT THEN HDI = s4B;
ELSIF DIGTP THEN HDI = s46; END IF;
```

```

WHEN s1 => HDI = s2;          CH MODE=VCC; D[]=H"71"; %Acq Sequence%
WHEN s2 => HDI = s3;          MODE0=VCC;   CH MODE=VCC; D[]=H"78";
WHEN s3 => HDI = s4;          MODE0=VCC;   D[]=H"78";
WHEN s4 =>                     MODE0=VCC;   ENACRO=VCC; D[]=H"7C";

```

IF DIG THEN HDI = s10; ELSIF R_PRE THEN HDI = s5; ELSIF !DIG & !R_PRE THEN HDI = s4; END IF;

```

WHEN s5 => HDI = s6;          MODE0=VCC;   ENACRO=VCC; D[]=H"7D"; %Preamp Reset%
WHEN s6 => HDI = s7;          MODE0=VCC;   ENACRO=VCC; D[]=H"7D";
WHEN s7 => HDI = s8;          MODE0=VCC;   ENACRO=VCC; D[]=H"7D";
WHEN s8 => HDI = s9;          MODE0=VCC;   ENACRO=VCC; D[]=H"7D";
WHEN s9 => HDI = sA;          MODE0=VCC;   ENACRO=VCC; D[]=H"7D";
WHEN sA => HDI = sB;          MODE0=VCC;   ENACRO=VCC; D[]=H"7D";
WHEN sB => HDI = sC;          MODE0=VCC;   ENACRO=VCC; D[]=H"7D";
WHEN sC => HDI = sD;          MODE0=VCC;   ENACRO=VCC; D[]=H"7D";
WHEN sD => HDI = sE;          MODE0=VCC;   ENACRO=VCC; D[]=H"7D";
WHEN sE => HDI = sF;          MODE0=VCC;   ENACRO=VCC; D[]=H"7D";
WHEN sF => HDI = s4;          MODE0=VCC;   ENACRO=VCC; D[]=H"7D";

```

```

WHEN s10 => HDI = s11;        MODE0=VCC;   D[]=H"78"; %Pipeline Readout%
WHEN s11 => HDI = s12;        MODE0=VCC;   SMCLK=VCC; D[]=H"78"; %Before Digitize%
WHEN s12 => HDI = s13;        MODE0=VCC;   SMCLK=VCC; D[]=H"78";
WHEN s13 => HDI = s14;        MODE0=VCC;   SMCLK=VCC; D[]=H"78";
WHEN s14 => HDI = s15;        MODE0=VCC;   D[]=H"78";
WHEN s15 => HDI = s16;        MODE0=VCC;   D[]=H"78";
WHEN s16 => HDI = s17;        MODE0=VCC;   D[]=H"78";
WHEN s17 => HDI = s18;        MODE0=VCC;   D[]=H"70";
WHEN s18 => HDI = s19;        MODE0=VCC;   D[]=H"70";
WHEN s19 => HDI = s1A;        MODE0=VCC;   D[]=H"70";
WHEN s1A => HDI = s1B;        MODE0=VCC;   D[]=H"70";
WHEN s1B => HDI = s1C;        MODE0=VCC;   D[]=H"70";
WHEN s1C => HDI = s1D;        MODE0=VCC;   D[]=H"70";
WHEN s1D => HDI = s1E;        MODE0=VCC;   D[]=H"70";
WHEN s1E => HDI = s1F;        MODE0=VCC;   D[]=H"70";
WHEN s1F => HDI = s20;        MODE0=VCC;   D[]=H"30";
WHEN s20 => HDI = s21;        MODE0=VCC;   SMCLK=VCC; D[]=H"38";
WHEN s21 => HDI = s22;        MODE0=VCC;   SMCLK=VCC; D[]=H"38";
WHEN s22 => HDI = s23;        MODE0=VCC;   D[]=H"38";
WHEN s23 => HDI = s24;        MODE0=VCC;   D[]=H"38";
WHEN s24 => HDI = s25;        MODE0=VCC;   D[]=H"38";
WHEN s25 => HDI = s26;        MODE0=VCC;   D[]=H"38";
WHEN s26 => HDI = s27;        MODE0=VCC;   D[]=H"38";
WHEN s27 => HDI = s28;        MODE0=VCC;   D[]=H"38";
WHEN s28 => HDI = s29;        MODE0=VCC;   SMCLK=VCC; D[]=H"38";
WHEN s29 => HDI = s2A;        MODE0=VCC;   SMCLK=VCC; D[]=H"38";
WHEN s2A => HDI = s2B;        MODE0=VCC;   SMCLK=VCC; D[]=H"38";
WHEN s2B => HDI = s2C;        MODE0=VCC;   D[]=H"38";
WHEN s2C => HDI = s2D;        MODE0=VCC;   D[]=H"38";
WHEN s2D => HDI = s2E;        MODE0=VCC;   D[]=H"38";

```



```

WHEN s2E => HDI = s2F;          MODE0=VCC;          D[]=H"30";
WHEN s2F => HDI = s30;          MODE0=VCC;          D[]=H"30";
WHEN s30 => HDI = s31;          MODE0=VCC;          D[]=H"30";
WHEN s31 => HDI = s32;          MODE0=VCC;          D[]=H"30";
WHEN s32 => HDI = s33;          MODE0=VCC;          D[]=H"30";
WHEN s33 => HDI = s34;          MODE0=VCC;          D[]=H"30";

```

```

WHEN s34 => HDI = s35;          MODE0=VCC;          CH MODE=VCC; D[]=H"30";      %Dig Sequence%
WHEN s35 => HDI = s36;          MODE1=VCC;          MODE0=VCC;          CH MODE=VCC; D[]=H"B2";
WHEN s36 => HDI = s37;          MODE1=VCC;          MODE0=VCC;          EN53=VCC;   D[]=H"B2";
WHEN s37 => HDI = s38;          MODE1=VCC;          MODE0=VCC;          EN53=VCC;   D[]=H"B0";
WHEN s38 => HDI = s39;          MODE1=VCC;          MODE0=VCC;          EN53=VCC;   D[]=H"A0";
WHEN s39 => HDI = s3A;          MODE1=VCC;          MODE0=VCC;          EN53=VCC;   D[]=H"A0";
WHEN s3A => HDI = s3B;          MODE1=VCC;          MODE0=VCC;          EN53=VCC;   D[]=H"A0";
WHEN s3B =>                      MODE1=VCC;          MODE0=VCC;          EN53=VCC;   D[]=H"80";
      IF !/RCO THEN HDI = s3C;      ELSIF /RCO THEN HDI = s3B; END IF;
WHEN s3C =>                      MODE1=VCC;          MODE0=VCC;          D[]=H"03";      %Dig Done%
      IF READOUT THEN HDI = s3D;    ELSIF !READOUT THEN HDI = s3C; END IF;      %Readout Command%

```

%Readout Sequence%

```

WHEN s3D => HDI = s3E;          CH MODE=VCC; MODE1=VCC; MODE0=VCC; D[]=H"71";
WHEN s3E => HDI = s3F;          DAV=VCC; CH MODE=VCC; MODE1=VCC; D[]=H"AA";
WHEN s3F => HDI = s40;          DAV=VCC;          MODE1=VCC; D[]=H"BB";
WHEN s40 =>                      DAV=VCC; BUS EN=GND; EN26=VCC; PRI IN=VCC; MODE1=VCC; D[]=H"7F";
      IF PRI OUT THEN HDI = s41;    ELSIF !PRI OUT THEN HDI = s40; END IF;
WHEN s41 => HDI = s42;          CH MODE=VCC; MODE1=VCC; D[]=H"7F";
WHEN s42 => HDI = s0;           CH_MODE=VCC; D[]=H"00";

WHEN s48 => HDI = s49;          D[]=H"F6";      %Cal Inj%
WHEN s4A => HDI = s34;          D[]=H"F6";      %Cal Inj%

WHEN s43 =>                      D[]=H"71";      %Download%
      IF !DNLD EN THEN HDI = s44;  ELSIF DNLD EN THEN HDI = s43; END IF;
WHEN s44 => HDI = s0;           D[]=H"F1";

WHEN s45 =>                      D[]=H"00";      %Power Up Sequence%
      IF IDLE THEN HDI = s46;      ELSIF !IDLE THEN HDI = s45; END IF;
WHEN s46 => HDI = s47;          CH MODE=VCC; D[]=H"00";
WHEN s47 => HDI = s0;           CH_MODE=VCC; D[]=H"71";

WHEN s4A => HDI = s34;          CH_MODE=VCC; D[]=H"30";      %IDLE to DIG%

WHEN s4B => HDI = s3D;          CH_MODE=VCC; D[]=H"03";      %IDLE to READ%

```

WHEN OTHERS => HDI = s0;

END CASE;

ENACRO.CLK = 53MHz;
ENACRO.D = ENCRO;
ENA53.CLK = 53MHz;
ENA53.D = EN53;
ENA26.CLK = 53MHz;
ENA26.D = EN26;

END;

SUBDESIGN 'SELECTOR' %Selects outputs as HDI Control, HDI Data, or Status%

```
(
    NOTIFY, LA, NOSYNC, PAR ERR          :INPUT;
    D[7..0]                               :INPUT;
    HBIT6, HBIT5, HBIT4, HBIT3, HBIT2, HBIT1, HBIT0 :INPUT;

    A[7..0]                               :OUTPUT
)
```

BEGIN

```
% CONTROL & READOUT # LOGIC ANALYZER # COMM. ERRORS %
% CAV ONLY %
A7 = !NOTIFY & ! LA & D7 # LA & !NOTIFY & GND # NOTIFY & GND;
A6 = !NOTIFY & ! LA & D6 # LA & !NOTIFY & HBIT6 # NOTIFY & GND;
A5 = !NOTIFY & ! LA & D5 # LA & !NOTIFY & HBIT5 # NOTIFY & GND;
A4 = !NOTIFY & ! LA & D4 # LA & !NOTIFY & HBIT4 # NOTIFY & GND;
A3 = !NOTIFY & ! LA & D3 # LA & !NOTIFY & HBIT3 # NOTIFY & GND;
A2 = !NOTIFY & ! LA & D2 # LA & !NOTIFY & HBIT2 # NOTIFY & GND;
A1 = !NOTIFY & ! LA & D1 # LA & !NOTIFY & HBIT1 # NOTIFY & PAR ERR;
A0 = !NOTIFY & !_LA & D0 # _LA & !NOTIFY & HBIT0 # NOTIFY & NOSYNC;
```

END;