SVX II Silicon Strip Detector Upgrade Project

SVX II Emulator Board

TEST STAND SPECIFICATION

--PRELIMINARY--

19 September, 1994
Revision 1.0

John T. Anderson
Electronic Systems Engineering Department
Computing Division
Fermi National Accelerator Laboratory

Document # ESE-SVX-940919

1 GENERAL INFORMATION 1
1.1 System Introduction 1
1.2 Description Of Component & How It Fits Into The System 1
1.3 List Of Component Requirements 2
2 THEORY OF OPERATION AND OPERATING MODES 4
2.1 Basic Features & Operation (Including Block Diagram) 4
2.2 Diagnostic Features 12
3 EMBEDDED & DIAGNOSTIC/DEVELOPMENT SOFTWARE 14
3.1 Embedded Software 14
3.2 Development & Diagnostic Software 14
3.2.1 Description Of Hardware Test Platform 14
3.2.2 Description Of Software Test Platform 14
3.2.3 Software Tools & Methodologies 14
3.2.4 Test Features 14
3.2.5 User Interface Examples 15

1 GENERAL INFORMATION

The SVX II Emulator Card is designed to provide emulation of the digital data transfer functions of the SVX II ASIC[1]. Some emulation of the analog functions is provided in that SVX-like data may be generated upon command for readout through the data acquisition system. The SVX II Emulator is designed to emulate either a single SVX II or a chained set, providing the opportunity to exercise system intialization software.

Various levels of software and hardware are required to verify that the Emulator is in proper operating condition. This document describes the hardware, software and algorithms used to test the Emulator card in the absence of any other SVX system hardware, and may be used as a basis for in-system diagnostic development.

This document pre-supposes a general understanding of both the SVX II ASIC and the functional characteristics of the SVX II Emulator card. A detailed description of the Emulator module is found in the SVX Emulator Design Specification.

1.1 System Introduction

(These paragraphs should briefly describe the system in which this component is used. This subsection should be identical for all components of a particular system. This section will be provided for you at a later date. It has not yet been written.)

1.2 Description Of Component & How It Fits Into The System

A single SVX II Emulator is connected to the SVX Port Card, using a 60 pin ribbon cable which plugs in to the front of the Emulator board. The SVXII Emulator card itself is a 9U X 400 mm VME Slave module. Although power for the SVX chip in the final system is provided over the cable by the Port Card, the SVX Emulator will draw all power from the VME backplane, as the load of the SVX Emulator is envisioned to be significantly greater than the actual SVX chip.

A simple VME Slave interface is provided which allows access to all internal registers and memories of the SVX Emulator. The SVX emulator connection to the Port Card is provided at the front panel of the module. The SVX Emulator provides three basic functions to the software:

1) A **Data Generator** which takes data patterns loaded from VME in to a FIFO and clocks them out over the Port Card interface in response to Readout cycles. By correct programming of the **Data Generator**, the SVX Emulator may emulate an arbitrary number of

Page: 1
[0]As described in **A Beginners Guide to the SVXII**, written by R. Yarema, dated 4/94.[1]

SVX chips, and arbitrary physics data within each one of those chips. This data generator is used to emulate the **Readout** mode of the SVX II chip.

2) A **Serial Parameter Memory** which provides access to the serial parameters loaded to the SVX Emulator by the rest of the SVX system, to determine that they were loaded correctly. This memory emulates the **Initialize** function of the SVX II chip. Some of the parameters loaded in this block are used to form the data in the **Data Generator.** The **Serial Parameter Memory** is also made available to the VME interface, so that data may be checked independent of the rest of the SVX system, and also so that software may vary the data stored in the **Data Generator** based upon loaded parameters.

3) A **Slow A/D Converter Block** which gives the VME bus access to coarse digitizations of the various bias voltages and currents found on the SVX chip interface, to allow diagnosis of bad cables or faults in the analog sections of the Port Card.

In addition to these three basic functions, four **Clock Counters** are provided which count the number of clock cycles provided to the SVX Emulator by the rest of the system. It is envisioned that these counters will be useful in chasing down errors in the controlling hardware related to improper numbers of channels being digitized. The **Clock Counters** are initialized by VME software which is presumed to be synchronized with the rest of the system. Each counter measures the number of clock pulses received in one of the four operating modes. The output value of all four counters is always available to VME.

1.3 List of Test Stand Requirements

The general requirements of the SVX II Emulator test stand are as follows:

1) Ability to test generic VME I/O access to the Emulator, including checking for individual failures in data, address or address modifier bits as recevied or driven on the backplane.
2) Generation and readback of serial initialization data over the front panel connector, in streams up to 2000 bits long under program control. The test program must be able to read the data stored in the serial parameter section of the Emulator card and compare it not only to the data sent but also to the data re-emitted serially from the Emulator during subsequent initialization cycles.
3) Ability to exercise the variable-depth feature of the serial parameter section by programming different data set lengths in to the Emulator and the serial data source and comprehending the resultant failures.
4) Ability to input programmable numbers of clocks to the Emulator in each of the various modes of operation, and to determine if the Clock Counters in the Emulator correctly report the number of clock edges delivered by the clock source.

5) Ability to take initialization parameters as loaded from the serial parameter section of the Emulator and use this data, in conjunction with user-supplied data, to fill the Data Generator FIFO of the Emulator.

6) Ability to compare data emitted from the Emulator's Data Generator section to the data loaded, and categorize errors in data transmission.

7) Ability to create a sufficient set of analog input combinations to exercise the Slow A/D converter block of the Emulator.

8) Provide and document a series of Emulator functions as part of the development set which may be used in modular fashion to easily integrate the Emulator test stand in to a Port Card, FIB or system test stand.

9) Provide a stand-alone Emulator test program which may be used to determine to reasonable degree that an Emulator board loaded in to another test system is functional independent of the state of any of the other interface boards in the system.

10) Implement the above using a common set of host processor, operating system and user interface tools such that Emulator programs are easily re-used not only in other test stands of the SVX project, but also in the final system installation.

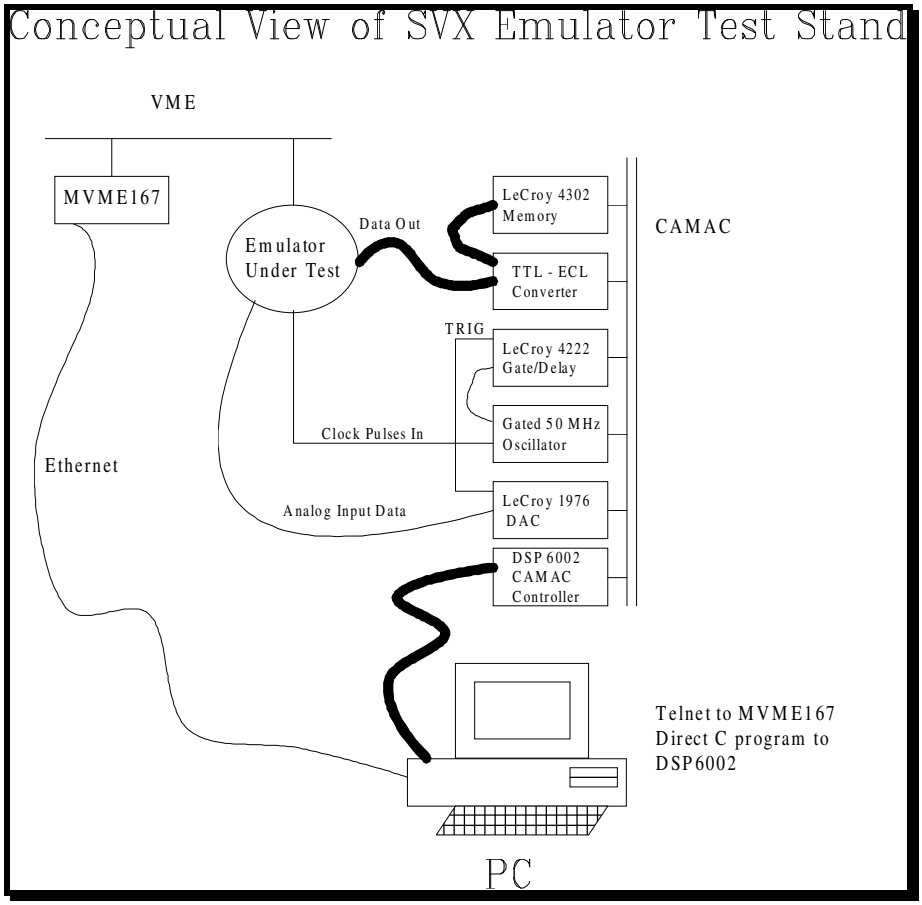
2 TEST STAND ARCHITECTURE

2.1 Basic Features & Operation (Including Block Diagram)

The Emulator Test Stand is designed to exercise all functional blocks of the SVX Emulator. A significant fraction of the logic within the Emulator board may be exercised without the use of external hardware, but certain hardware is required for a complete test. Each functional block of the Emulator will be examined in turn, and for each, the general algorithm and required external equipment will be detailed.

The test stand is based in 9U VME, controlled by a Motorola MVME162 processor board. Any other processor board, or a PC with a bus interface, may be substituted if software compatibility is assured by the user. Some external signals are provided using a CAMAC backplane and standard CAMAC test modules, in those cases where CAMAC modules are 'at hand' and VME equivalents would have to be ordered.

Communication with the MVME162 is coordinated with CAMAC activity by using an IBM PC clone as both CAMAC host processor and remote terminal for the MVME162 processor, as shown here:



VME Interface Block

The Emulator module is designed to respond to 16-bit VME single data transfers. A series of read/write registers are defined, plus a pair of FIFO memories. Each register is essentially read-write, allowing VME to write data values and see if they come back. A few registers have 'pulse' bits or 'set-reset' bits, which may require the use of scope loops in the software.

The Emulator is designed to respond only to 16-bit data transfers. Only the bare minimum of address modifiers, corresponding to single word transfers, are supported. Upon being successfully addressed, an internal signal ITS_ME* should be asserted, resulting in an LED on the front panel being illuminated. The ITS_ME* signal is also provided on the front panel via a logic analyzer connection.

The state of both DS0* and DS1* being low starts a pipelined response which results in DTACK* being asserted within about 5 ticks of the internal clock. This clock may be sourced either from SYSCLK* or from an on-board oscillator, as selected by a jumper. Using a nominal frequency of 20 Mhz, the DS* to DTACK* time should be approximately 250 nsec. Simulation of the DS to DTACK response corroborates this estimate.

To fully test the VME interface block, the following tests need be performed:

1) Addressing test. Various VME addresses need be sent to the Emulator, and it should only respond to those meeting the DIP switch settings. To fully test the Emulator, a full 32-bit 'walking 1's' pattern of addresses should be sent. The Emulator may be programmed to accept either 24-bit or 32-bit address cycles. Physical identification of the addressing mode employed by the Emulator is achieved by checking chip location U5. If U5 is popluated by a 74ALS521 chip, then the Emulator is set to 32-bit addresssing mode. If, instead, a plug with a wire from pin 19 to pin 1 is found, then the Emulator is set to 24-bit addressing mode.

Note: The SVX_DECD FPGA is normally programmed to only accept 24-bit addresses. A variant version is required for 32-bit address recognition.

In both modes, the Emulator compares address bits from bit 12 on up via comparator chips to a DIP switch setting, and expects that address bits 6-11 will be zeroes. Bits 1-5 are used to select the register of interest. Thus, the Emulator may be set to occupy the lower 32 word addresses of any 4K word address block.

2) Address modifier test. The Emulator is designed to respond only to single-transfer address modifiers. If the AM specifies any form of Block transfer, the Emulator will not honor the address, even if it matches the DIP switch settings, resulting in BERR* upon the data half of the transacation. Only the following address modifiers are accepted:

57 (hex 39): Standard non-privileged data access (24-bit address)

58 (hex 3A): Standard non-privileged program access (24-bit address)

61: (hex 3D): Standard supervisory data access (24-bit address)

62 (hex 3E): Standard supervisory program access (24-bit address)

3) DS to DTACK response test. From the software side, only go/no-go testing (received DTACK or BERR) is possible. On the hardware side, the internal response of the Emulator to the DS signals is seen by scope-probing the output pins on U9, the SVX_DECD FPGA. The internal state of the DS pipeline is visible on the following pins:

DSA: pin 67 (test pad provided on board)

DSB: pin 71 (test pad provided on board)

DSC: pin 73 (test pad provided on board)

ACK: pin 80

ANY_READ: pin 76

ANY_WRITE: pin 51

Use of logic analyzer connector #2 will prove beneficial here. This connector provides access to BDS1*, BDS0*, BAS*, ANY_READ, ANY_WRITE, ITS_ME and DTACK*, which should show the entire cycle. If no DTACK* is obtained in response to a VME cycle, the most logical failures are as follows:

1) The Master is using a 32-bit address, and therefore a 32-bit address modifier, which is not supported by the standard Emulator logic. If 32-bit address modifiers are to be used, the SVX_DECD FPGA needs to be reprogrammed to accept them.
2) If neither ANY_READ, ANY_WRITE nor DTACK* is seen, check the outputs of the ‘ALS521 address comparators on pin 19 of each ‘ALS521 with an analyzer and standard DIP clips to determine that the address is being recognized and that the DIP switches setting the address are correctly set. Insure that the address comparators settle before the falling edge of AS*, as that is when they are sampled.
3) If the address is properly set up, add probes to look at DSA*, DSB* and DSC*. If they are also not pulsing, the clock is probably set up wrong. Inspect jumper J31 to determine that a clock is present on pin 2. If no clock is present, this may be due to there being no SYSCLK on the VME backplane, which can be handled by setting J31 to the 1-2 position which uses the on-board oscillator at position U43.

4) Data transfer bus test. Full tests of the 16-bit data bus to a test register is, unfortunately, not in the cards for the Emulator. However, most of the data bits may be tested by performing write/read cycles to the Initialize Mode Counter, which should act as a simple 12-bit latch when accessed directly. The upper data bits are rarely used in the Emulator.

Note: To make this thing more testable, I should see if I can change the INIT mode counter to act as a full 16-bit latch for write/read cycles by adding a 4-bit latch. The latch should be clocked by writing to the init mode counter directly, and cleared by CLR_INIT_CNTR. This would give a semi-independent check of the entire data bus.

5) Register addressing. With a simple loop of addresses, one may probe the decoder logic by looking at the various register decodes for writes at the output pins of U9, or the clocks of the registers themselves. Decoding of reads is performed in parallel spots, so the best test of bus contentions is to work with the board only partially stuffed at first (only one or two FPGA's) and check the bus zeroing logic of the 'zero' PAL, then add FPGA's until the bug shows up.

Slow A/D Conversion Block

The Slow A/D conversion block utilizes two multiplexed 8-bit A/D converters to provide coarse analog measurement of the various power supplies and analog bias signals in the SVX system.

To test each of these signals, a 16-bit DAC housed in a LeCroy 1976 ADC/TDC test module is used to provide a programmable output voltage. The voltage output is then wired to the Emulator, one input at a time, and the digitized value is compared manually to the set value. Only coarse digitization is required, so no calibration of pedestal, integral non-linearity or differential non-linearity is called for.

A test program SETDAC on the PC allows the user to set the 1976 DAC to various voltages, which can then be read back through the A/D converters of the Emulator card.

Similarly, the various power supply voltage digitzations provided in the Emulator may be tested by connecting the 1976 DAC output to the power supply pins, and using the on-board A/D to measure the results.

Testing the Serial Parameter FIFO

The Serial Parameter FIFO is designed to be programmable on input and output so that either port may be connected either to VME or to the Port Card. Internal testing of the FIFO is accomplished by setting the device to it's Parallel In/Parallel Out mode via the Serial Parameter Control Register and performing standard FIFO memory testing. Once correct operation of the FIFO has been verified, correct operation in the Serial mode is tested by using a loopback cable.

The loopback test is performed by placing a known data block in to the FIFO such that less than half of the FIFO depth is used, using the Parallel input mode from VME. After the FIFO is loaded, the FIFO is switched to the Serial In/Serial Out mode and an external cable is used to connect the TNBR (data out) serial output to the BNBR (data in) serial input. A series of clocks is applied to the Emulator in Iniailize mode which should cause the data in the FIFO to be shifted out of the ouput, through the cable and back in to the input.

After the clocks are applied, the FIFO is then read out using Parallel output mode. Presuming that the number of clocks applied to the system is equal to the number of bits of data originally loaded, the FIFO should contain the same data as originally loaded. To check that data has actually been transferred, a number of clocks greater or lesser than the exact count may be used to offset the data pattern.

Initimately associated with the Serial Parameter FIFO is the Initialize Mode Clock Counter and the Number of Chips to Emulate register. When testing the Serial Parameter FIFO, the Number of Chips to Emulate setting, plus any values loaded in to the Chip Boundary Counter register, will affect the number of data bits that can be passed from output to input in the Serial Parameter FIFO self-test sequence.

The serial parameter block stores the serial parameter data as loaded from the Port Card during the Initialize mode of the SVX system and also provides serial readout of those same parameters. This section contains serial-to-parallel conversion, a FIFO to store the parallel data, and a parallel-to-serial converter to allow readout of the stored data. While data is being loaded, a bit counter will count which bit position is being stored in to the FIFO.

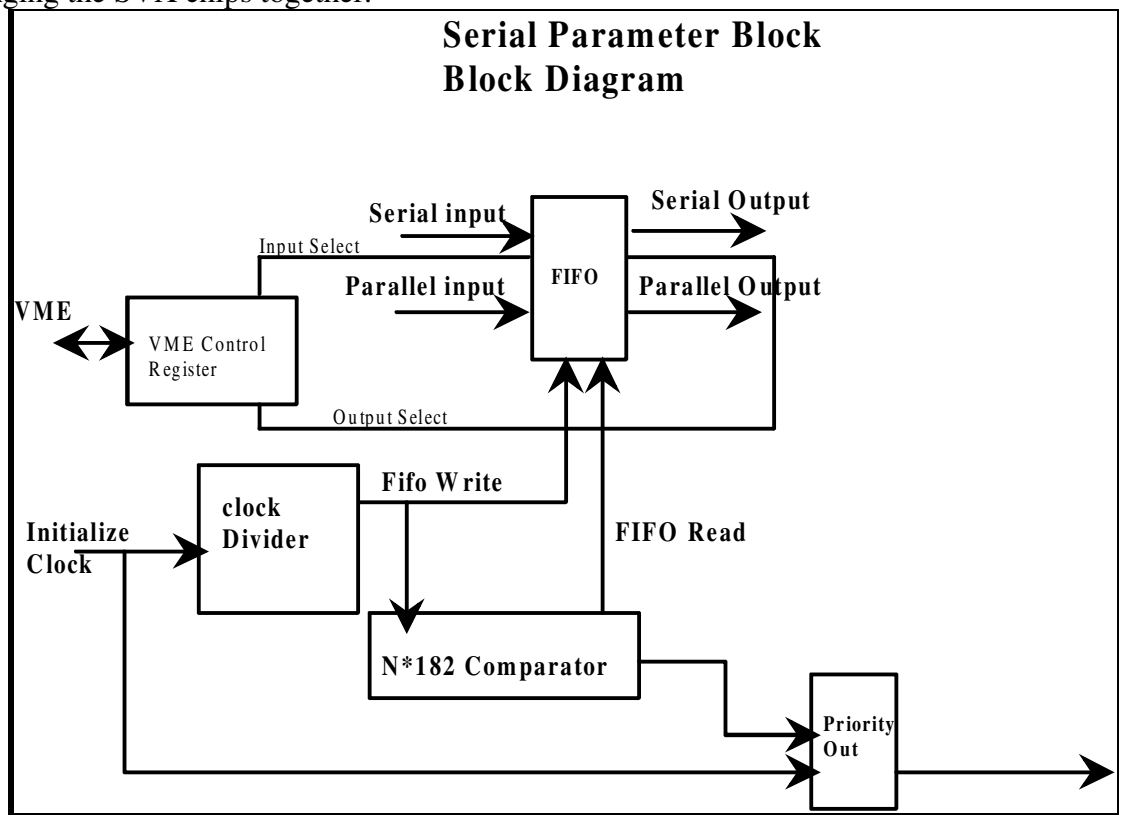
The IDT 72103 Serial-Parallel FIFO component is used to implement this function. Two mode bits control the mode of the input and output sections of the chip, respectively. Each bit may select either serial or parallel operation. With this configuration, the serial parameter functions are easily tested. Each of the four selections has a use:

Input	Output	Function
Serial	Serial	Direct emulation of serial shift register of N chips
Serial	Parallel	Serial input which is then read and checked over VME
Parallel	Serial	Load test data from VME, shift out to Port Card

Parallel Parallel VME self-test of FIFO chip

A secondary counter will be used to count the number of bits loaded in to the system. For diagnostic purposes this counter will be made available to VME whenever the SVX Emulator is in the Acquire, Digitize or Readout modes. During the Initialize mode the counter is counting, and so any read during this time cannot be guaranteed to yield correct data. The lower 4 bits of this counter are also used to count how many clocks have occurred since the FIFO behind the serial-to-parallel converter has been written to. Since the serial-to-parallel converter holds eight bits, a simple decoding scheme may be used to load the FIFO.

To provide a cross-check, and to simplify loading the Data Generator block, an "n * 182" comparator circuit is connected to the bit counter. Every 182 clocks, the comparator will trigger a second chip boundary counter. This counter will, at the end of the Initialization sequence, tell how many SVX chip's worth of data was actually sent. This same value can also be used to determine how much data should be loaded in to the Data Generator. In accordance with the SVX specifications, the chip boundary counter will be compared to a register which is programmed with the number of SVX chips being emulated. Should the boundary counter become equal to the number of chips being emulated, an internal status line will be set that enables the serial data to start shifting out of the FIFO - emulating the fixed-length shift register created by stringing the SVX chips together.



The Data Generator Block

The **Data Generator** block contains a 8K X 9-bit FIFO which is used to hold the data patterns representative of SVX data. The FIFO is loaded from VME prior to using the Emulator, and may be loaded with either cable-test bit patterns (e.g. walking 1's, walking 0's, A/5, 5/A, etc.) or simulated detector data. The lower eight bits of the data pattern are the eight-bit data which is provided by the SVX chip. The 9th bit is used to mark the 'end of data block', allowing the single FIFO to actually contain multiple data sets. No marker is required between emulated chips, as the Port Card is connected only to the Priority Out of the last chip in the chain. Thus, all the Emulator board has to do is provide a block of data, followed by Priority Out. This architecture, as a side benefit, allows the VME software to create error conditions of either 'too few chips in the chain' or 'too many chips in the chain' on an as-needed basis.

The ability to contain multiple data sets is necessary to be able to test the Port Card and readout system at speed. VME bus is probably incapable of reloading the FIFO with a new data pattern before another readout cycle would begin, and a single test pattern is not sufficient to exercise the full data handling and processing capabilities of the SVX readout system. By loading multiple 'events' in to the SVX Emulator FIFO, system performance with variable event sizes and identification of 'missed' chips or headers may be tested.

In practical use, the **Data Generator** is loaded with data after a first Initialize cycle has loaded the **Serial Parameter Block** with chip ID's. This presumes that the overall system has the ability to halt after the Initialize cycle, and allow the software which controls the SVX Emulator the time required to read the chip ID's, calculate the data pattern, and load the **Data Generator.** Once loaded, the SVX Emulator may be cycled through limitless Acquire-Digitize-Readout cycles, so long as the output data pattern (or pattern set) is usable. Should a new data pattern or pattern set be required, the Acquire-Readout-Digitize sequence must be halted until the new data has been loaded in to the **Data Generator.**

The FIFO Empty flag is used to set a separate flip-flop that will signal the board, during the Acquire/digitize sequence following the readout, that the retransmit flag of the FIFO should be pulsed, allowing endless repetition of the group of events stored therein.

Within a single Readout sequence, the **Data Generator** will emit data with each edge (rising or falling) of the externally supplied clock, incrementing the FIFO internal read address, until a word with the 9th bit set is emitted. At this point, further increments of the FIFO address is halted and an external flip-flop is set, asserting the PRIORITY-OUT signal (which should cause the port card to cease issuing clocks). The flip-flop is also set by the empty flag of the FIFO. The PRIORITY-OUT flip-flop is cleared by the CHANGE-MODE signal, so that when the port card leaves Readout mode the FIFO is immediately re-enabled to emit the next data block. Should the FIFO be empty (as is expected when the last data block has been read out), the reset of the PRIORITY flip-flop will also clock the Retransmit pin of the FIFO, resetting the read pointer back to zero and restarting the sequence.

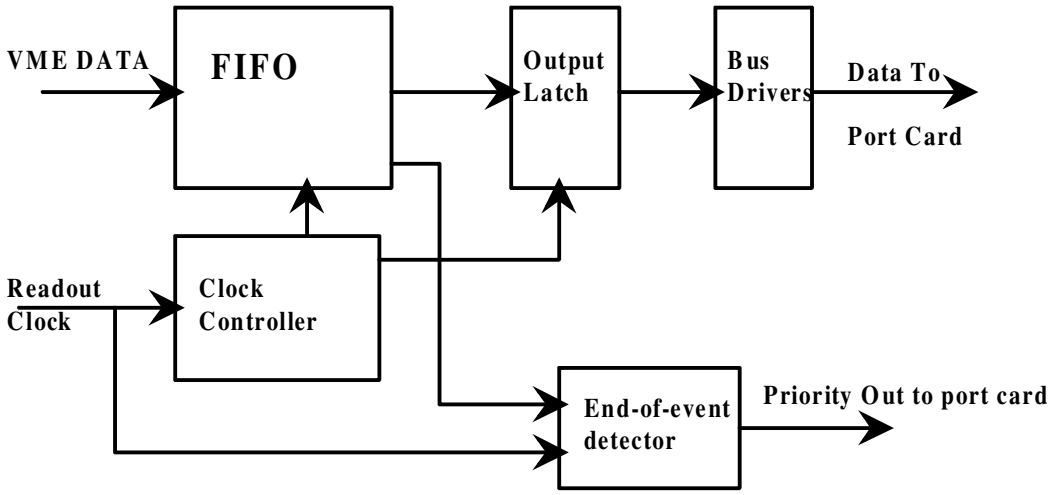
The 8K depth of the **Data Generator** FIFO allows for the storage of multiple events. A single calibration event (all channels have data) would require 128 bytes of data storage. To emulate 10 SVX chips in a chain would need 10 times this amount, or 1280 bytes. The 8K depth chosen is sufficient to hold any reasonable set of calibration event plus "normal" events (assuming 10% occupancy of channels in a "normal" event).

In order to allow the **Data Generator** to function at the full speed of the system, a latch is used to pipeline the data output over the SVX cable. When data transfer is initiated (by the CHANGE-MODE which selects Readout mode), the Read strobe is brought low to access the first data word in the FIFO. As each edge of the clock is received from the Port Card, the output latch is clocked by the invert of Read - thus latching the data for the port card and holding it stable while the next data value is accessed out of the FIFO chip. This will result in data being asserted on the cable in about 10 nsec after the receipt of the clock, allowing for the clock-to-Q propagation delay of the output latch and the propagation delay of the data drivers.

Checks with various suppliers of FIFO parts has shown that single FIFO's fast enough to fulfill this function are not yet available, although they are listed in the data books. In order to achieve the desired data rate, two slower FIFO chips will be used in 'Ping-Pong' fashion, with a multiplexer to select between the outputs.

During the review of the STAR module it was discovered that a prototype data generator of identical function has already been tested as part of the STAR. In order to minimize duplicated effort, the STAR's data generator will be copied in the SVX Emulator if feasible.

Data Generator Block
Block Diagram



2.2 Diagnostic Features

General diagnosis of failures within the SVX Emulator will be accomplished by the following techniques:

• The **Serial Parameter Block** may be exercised from VME by setting the FIFO to parallel-in/parallel-out mode and then checking the FIFO from VME. By selecting either input or output to be serial (with the other side left set to parallel), data received from the Port Card may be examined by VME, or test data sent to the Port Card. The normal serial-in/serial-out mode may then be used to exercise normal operation of the system.

• Suspected errors in the **Data Generator** may be exercised by slowing down the clock rate and sending the data, not to the Port Card, but to standard CAMAC memory input modules such as the LeCroy 4302 or the DYCTI (still in development). What may be required, dependent upon signal levels, is a converter, but such are easily manufactured in-house as wirewrap prototypes. This loop would test that data written to the **Data Generator** by VME actually comes out as expected, but is limited by the limited speed of the CAMAC memories. If higher speed testing is required, then one of two techniques may be employed:

- Setting up a test with a faster memory such as a VDAS system, or
- Using a scope/logic analyzer combination to ferret out setup/hold timing violations and relegating data checking to the slower speeds.

The actual speed at which the CAMAC memories cycle is usually quoted at 20 MHz- that is, they can accept strobes at up to that speed. Since the SVX system runs with data transferred at both edges of the clock, this means that a CAMAC unit could only run at a 10 MHz clock speed, which is only about 40% of planned system speeds. However, since the CAMAC memories are 16 bits wide, and the SVX data is only 8 bits wide, the required level translator could also act as a multiplexer and double the effective readout rate - at the cost of a somewhat more complicated translator board.

• Testing of the **Slow A/D Converter** block is accomplished by simply connecting a known reference supply (such as an HP or Lambda power supply) or a CAMAC test module (like the LeCroy 1976) to the various analog signals and checking that the different channels of the A/D actually convert. Neither integral- or differential-non-linearity is a critical parameter in this block, so all that need be tested is that it provides conversions with a gain slope within a few percent of nominal. Should a channel fail, isolation of the fault between the A/D and the buffering amplifier is easily done with a scope.

• **Mode Selection** and the **System Clock** may be exercised from VME by accessing bits in the general control register of the module which allow the software to either monitor or override the externally generated settings. When the override feature is employed, the mode may be set as desired from VME, and a slow clock (caused by toggling a bit in the control register) may be controlled directly from software. This 'software clock' will also allow localized testing of all counter circuits within the SVX Emulator.

•Many internal signals will be brought out to the logic analyzer connections, which are mounted on the front panel of the module for easy access.

3 EMBEDDED & DIAGNOSTIC/DEVELOPMENT SOFTWARE

3.1 Embedded Software

There is no embedded software in the SVX II Emulator. All functions are accomplished with simple state machines.

3.2 Development & Diagnostic Software

3.2.1 Description Of Hardware Test Platform

3.2.2 Description Of Software Test Platform

Diagnostics to provide decoding of the diagnostic FIFO and exercise of all board functions (within reason, limited by clock rates) will be accomplished by a series of small ‘C’ programs. A host system based upon a 68000-family processor in a VME board has been selected. Software help in the person of Ming-Shen Gao or Don Walsh has been mentioned as a possibility.

Little external hardware other than a scope, pulse generator, bench-type variable power supply and perhaps a CAMAC memory module is envisioned as the bench test setup.

3.2.3 Software Tools & Methodologies

It is assumed that all software will be written in C, using compilers provided for the hardware test platform. The GNU C compiler under VxWorks or UNIX is presumed.

A simplistic methodology consisting of a single test program per main block of the Emulator is planned, with a menu-driven interface to allow selection of any test desired. For using the Emulator to test other portions of the SVX system, a similar menu-driven program with an extremely limited number of choices (e.g., one choice per operational mode of the SVX) appears appropriate.

3.2.4 Test Features

The SVX II Emulator will have the usual 'anything which can be written is readable' feature, plus embedded test points to facilitate easy logic analyzer hookup. External commercial test devices should be more than sufficient to exercise the unit 'in vitro', save for the 53 MHz clock rates that may accompany serial parameter data loading. For that rare case, a special test jig

that clocks out serial data at 53 MHz may have to be wirewrapped if a prototype of the Port Card is not available at the time the SVX Emulator prototype enters testing.

3.2.5 User Interface Examples

Without code in place, all that can be said is that simple arrow-key and function-key interface will be used, with no fancy stuff. Delivery time is of utmost importance, and so eliminating a ‘pretty’ user interface for the minimal type which provides the necessary functionality is planned.