GEM TN-93-471

GEM Data Acquisition System Preliminary Design Document

M. Bowden, J. Carrel, J. Dorenbosch, V. S. Kapoor, A. Romero
Superconducting Super Collider Laboratory

September 30, 1993

Abstract:

This document includes a proposed baseline architecture and suggested implementations for Data Acquisition System components of the GEM electronics. The protocols defined for data and control interfaces are subject to review and agreement by the various detector electronics subsystems.

GEM TN-93-471

GEM Data Acquisition System Preliminary Design Document

September 30, 1993

Abstract:

This document includes a proposed baseline architecture and suggested implementations for Data Acquisition System components of the GEM electronics. The protocols defined for data and control interfaces are subject to review and agreement by the various detector electronics subsystems.

M. Bowden, J. Carrel, J. Dorenbosch, V. S. Kapoor, A. Romero

Superconducting Super Collider Laboratory

System Architecture 3
Control and Data Flow 5
Selective Readout Mode 6
Full Readout Mode 8
Control Network 8
Trigger Supervisor 10
Gating Logic 10
Front-end Interface 10
Clock and Crossing Sync 11
Control Link 12
Control Link Options 16
Data Links 17
Event Data Collectors 19
Level 2 Trigger Interface 21
EDC Data Format 22
Event Builder 24
Event Data Distributors 29
Software 31
System Monitor 31
Processor Farm Interface 32
Event Data Distributor 32
Event Data Collector 33
Trigger Supervisor 34
Tape Server Interface 35
Control and Monitoring 35
System Errors 35
General Error Detection/Correction 36
EDC Error Detection 36
Bandwidth requirements 37
Data bandwidth requirements 37
Control bandwidth requirements 38

System Architecture

The baseline GEM Data Acquisition system is designed to support a bandwidth of approximately 10 GBytes/sec (figure 1). This implies a maximum event size of 1 MByte at the expected Level 1 rate of 10 kHz (or 100 kBytes at the highest allowable Level 1 rate of 100 kHz). If necessary, this bandwidth can be increased by a factor of four through the simple addition of output channels.

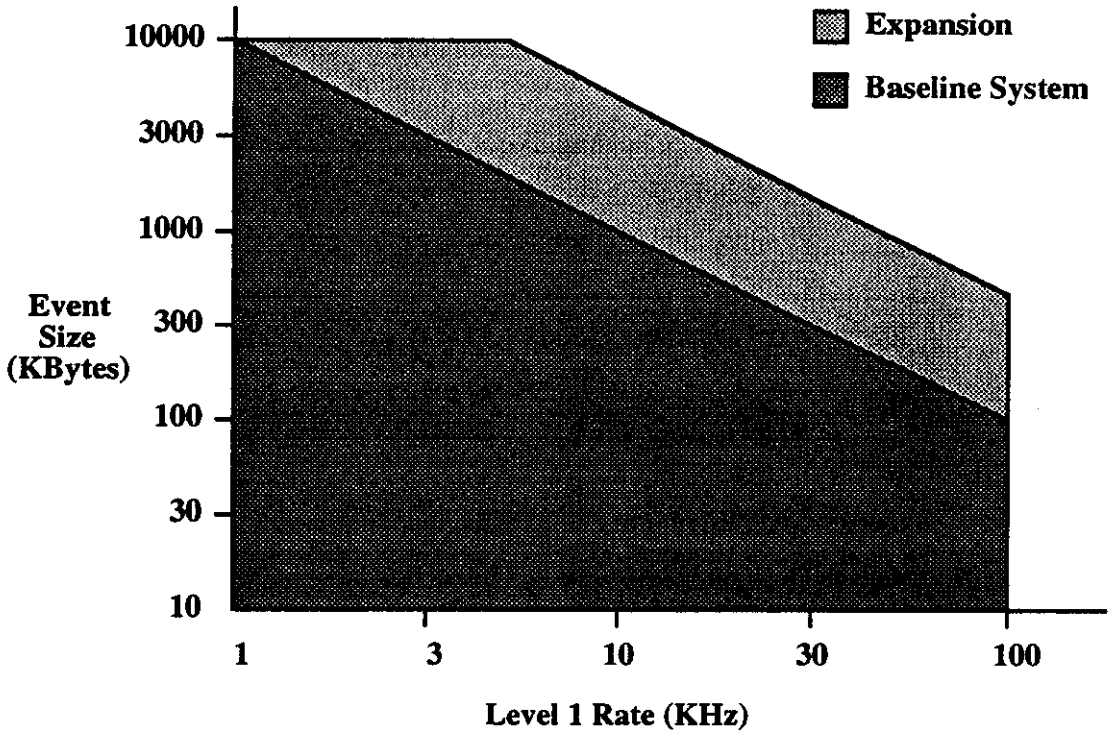

Figure 1. Data Acquisition System Operating Range.

The basic system architecture is shown in figure 2. Most of the Data Acquisition system is located in the Electronics Room and is connected to the front-end electronics through independent fiber data links. Approximately 10000 front-end modules are specified in the current GEM design (up to 16000 front-end modules are supported).

Following a Level 1 Trigger Accept, all digitized data from the front-end system are transmitted to the Event Data Collectors (EDCs) where they are buffered during the Level 2 Trigger decision. This approach allows a relatively long latency and greater flexibility in the design of the Level 2 Trigger. Level 2 will initially be implemented in software using the general-purpose processors of the Level 3 system. As the algorithms are refined, some portion may be transferred to hardware. The architecture allows any combination of hardware or software implementations.

A request for event data is broadcast to all EDCs in parallel. The data is returned sequentially, through a high speed Event Builder, to an Event Data Distributor (EDD). The system is heavily pipelined, with requests for data from many events (typically 10-20,000) being processed simultaneously by the EDCs.

The system allows up to 512 EDCs and 512 EDDs. The initial system is expected to use approximately 450 EDCs and 128 EDDs. The required number of EDCs is deter-

mined by the number of front-end modules to be supported. The required number of EDDs is determined by the data volume.

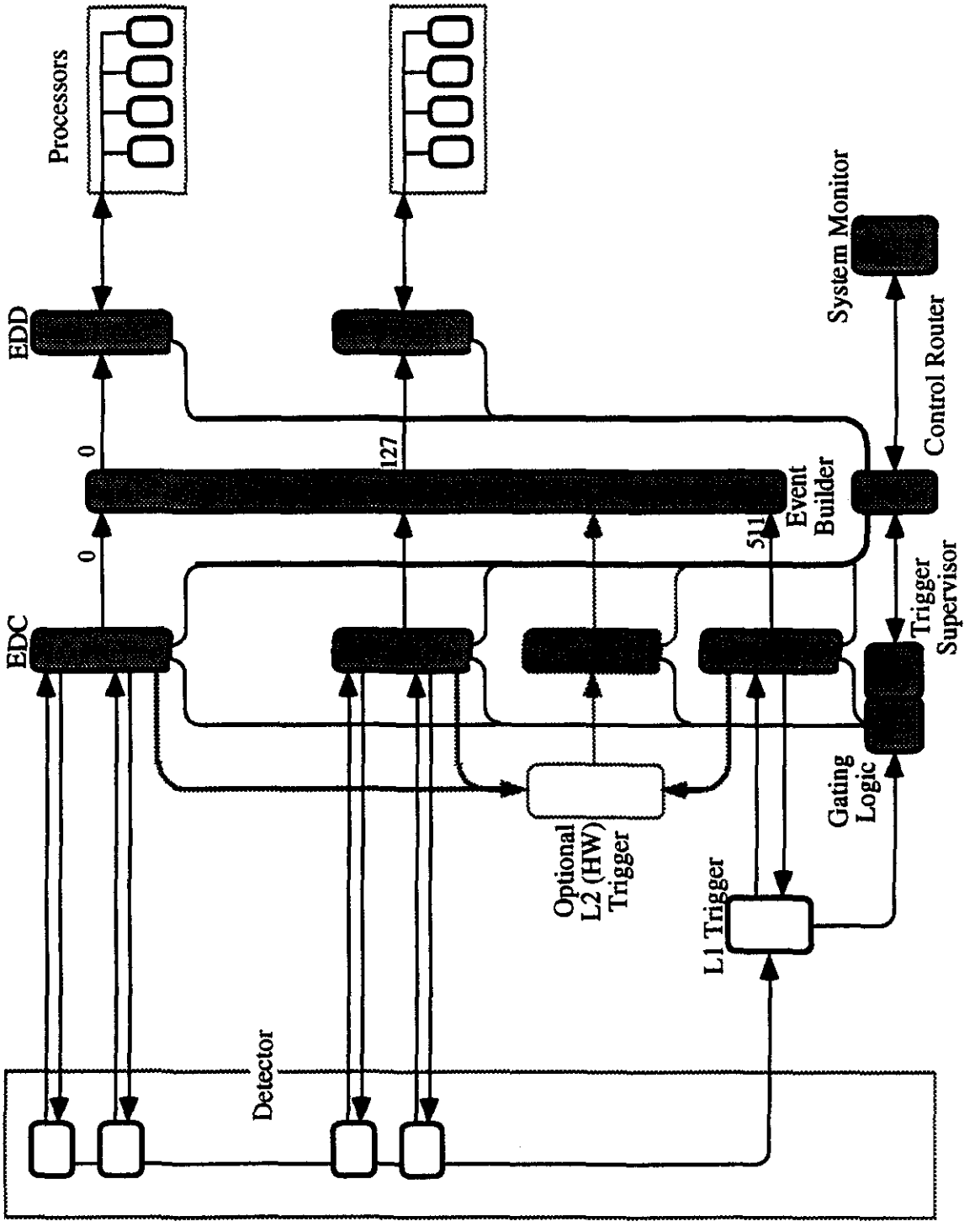

Figure 2. GEM Data Acquisition System.

Control and Data Flow

Processors with free buffer space may request a Level 2 event by sending an *EVENT_REQUEST* message via the EDD to the Trigger Supervisor (figure 3). This message can be a simple request (any event), or it can be a request for an event from a specific partition as labeled by the Level 1 Trigger and Trigger Supervisor. The Trigger Supervisor responds by sending an *EVENT_ASSIGN* message to the Processor. The Processor then uses the EVENT_NUMBER of the assigned event to request data from some or all of the EDCs in arbitrary order.

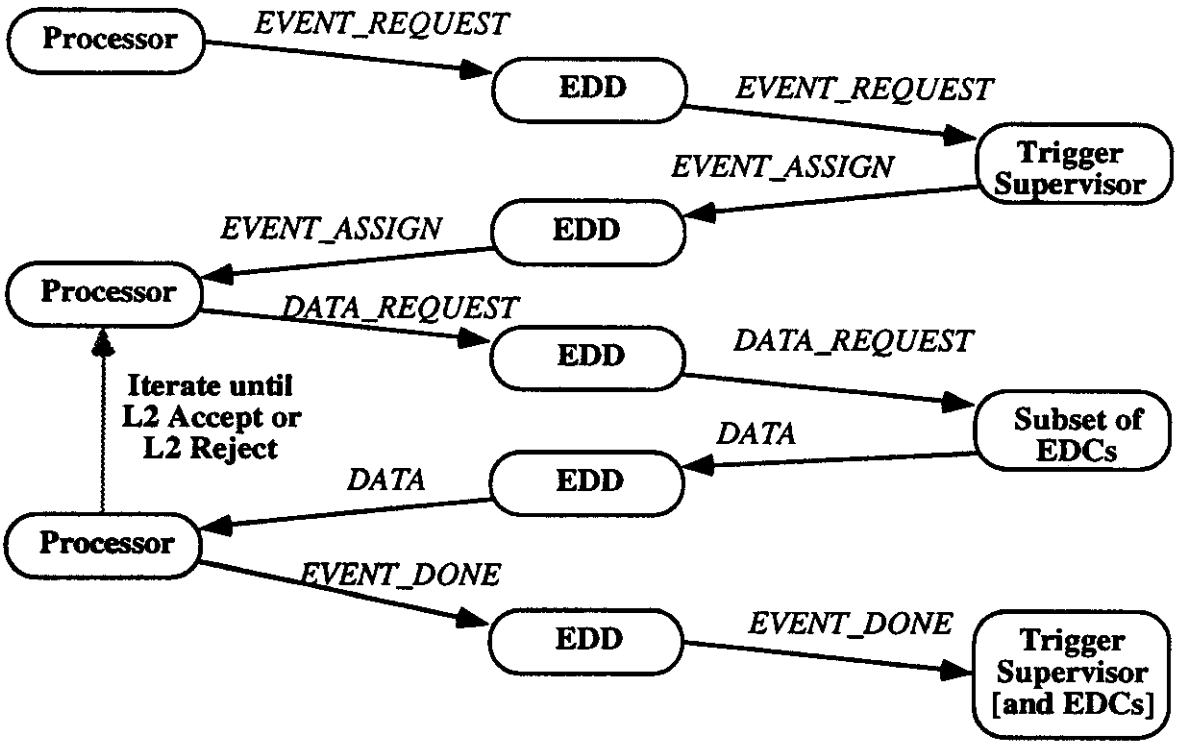

Figure 3. Event Readout Sequence.

Figure 4 shows a general schematic of the system control and data flow. Individual components and control messages are described later. Signal definitions in the drawing include:

Clink	Control Link for control messages
Dlink	Data Link for event data
Tlink	Data Link for trigger information
Sclk	60 MHz clock signal with phase adjustment
Aclk	60 MHz clock signal without phase adjustment
CS	Crossing Sync signal
L1A	Level 1 Trigger Accept and Trigger type information

The normal operating sequence is as follows:

1) The Level 1 Trigger receives trigger data from the front-end modules and generates a Level 1 Accept (along with trigger type information) to the Trigger Supervisor/Gating Logic. The Level 1 Accept is distributed by the Gating Logic to the EDCs and to all associated front-end modules.

2) The front-end modules transmit data to the EDC.

3) The Processor issues an *EVENT_REQUEST* to the EDD. The EDD passes this request through the Control Network to the Trigger Supervisor.

4) The Trigger Supervisor matches this request with a previously received Level 1 Accept and returns an *EVENT_ASSIGN* message through the Control Network to the EDD and Processor.

5) The Processor then issues *DATA_REQUEST* messages for some or all of the data. These requests again flow through the EDD and Control Network to the EDCs. The EDD retains a copy of the *DATA_REQUEST* message for use in determining when all the requested data has been delivered.

6) The EDC transmits the requested data through the Event Builder to the EDD. When the requested data from all EDCs arrives, it is assembled and forwarded to the Processor.

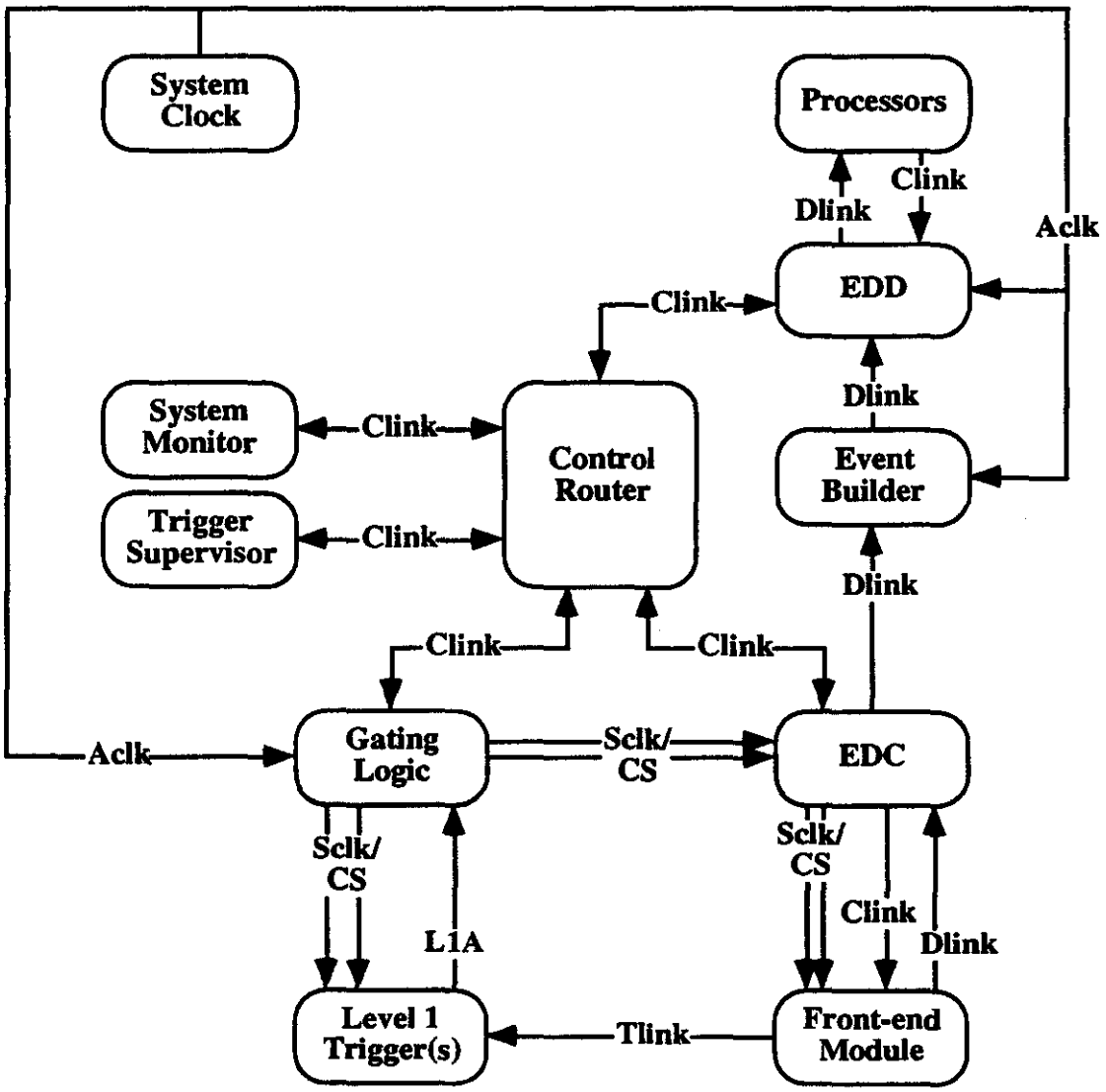

Figure 4. Complete Control & Data Flow Schematic.

Other types of messages may be sent directly from the EDCs, either automatically or at the request of the System Monitor or Gating Logic. The Trigger Supervisor/Gating Logic inhibits the Level 1 Accept if necessary and generates the appropriate synchronization signals to the front-end modules and EDCs. The EDC monitors the front-end mod-

ules by checking the header integrity of each event. If an error is reported, the EDC can issue a restart message to the front-end module. The EDC then sends a message to the System Monitor summarizing the errors and the range of event numbers affected. The System Monitor correlates these messages with other errors to determine if a full system reset is necessary.

Selective Readout Mode

To reduce the required data bandwidth (and cost) of the Event Builder, a selective readout option is used in the baseline design. In this mode, Processors may request data from individual EDCs or groups of EDCs. Only the data needed by the Level 2 algorithm is transferred through the Event Builder. The remainder of the data is stored in EDC buffers until the Level 2 decision is complete.

This mode does not appear difficult to implement, but it places additional requirements on the EDCs, EDDs and Control Network. The EDC buffers are larger to accommodate the increased latency, EDC memory management must allow full random access to event data, EDDs must assemble partial events of varying size (up to a full event) and the Control Network must provide additional bandwidth for the Event Data Request Messages.

The higher latency involved in transferring only requested data through the Event Builder implies that the Processors must handle many events simultaneously (requiring fast context switching and expanded memory). In addition, the random data traffic through the Event Builder switching network is less efficient.

Because of the added complication, the selective readout mode has been proposed mainly as a cost saving measure. It may not be the generally preferred approach.

Full Readout Mode

In the more expensive Full Event readout mode, ALL data for each event accepted by Level 1 is transmitted to the processors. This requires higher bandwidth in the Event Builder, or a correspondingly lower trigger rate. The advantage of this approach is that the Control Network requirements are minimal. Communication between EDDs and EDCs is limited a to single broadcast *DATA_REQUEST*, which reduces the Control Network traffic by approximately 75%.

Additional benefits include reduced memory management requirements in the EDC (events are processed in FIFO order) and reduced buffer requirements in both the EDCs and processors (as a result of decreased latency).

The negative aspect of the Full Event readout mode is the increased bandwidth requirement with the corresponding increase in the number of Event Builder and EDD modules. The estimated cost differential for the Full Event readout mode is $2-3M.

Control Network

The Control Network provides a connection to every EDD and EDC module in the Data Acquisition system. It also connects the Trigger Supervisor and System Monitor. To reduce bandwidth requirements, it must support broadcast messages. Messages may be directed to individual modules or to any predefined group of modules. For example, a readout message from an EDD might be directed to:

- an individual Calorimeter EDC,
- an individual Calorimeter EDC and its nearest neighbors,
- all EDCs in a selected η–ϕ region or sub-region of the Calorimeter (partial Calorimeter readout),
- all Calorimeter EDCs (full Calorimeter readout),
- all EDCs in a selected η–ϕ region or sub-region (partial event readout),
- all EDCs (full event readout), or
- any other combination which seems appropriate.

This allows control or readout of selected logical groupings with a single message. In particular, the Level 2 Accept or Reject message, which may be sent to all EDCs, requires only a single broadcast. Groupings are completely arbitrary and can be customized for individual events and detector partitioning.

The Control Network must guarantee order of arrival, i.e., successive messages from an EDD to an EDC must arrive in the original order at each EDC. This requirement simplifies the assembly of event fragments in the EDDs. It can be satisfied by routing all EDD-EDC messages across the same physical data path.

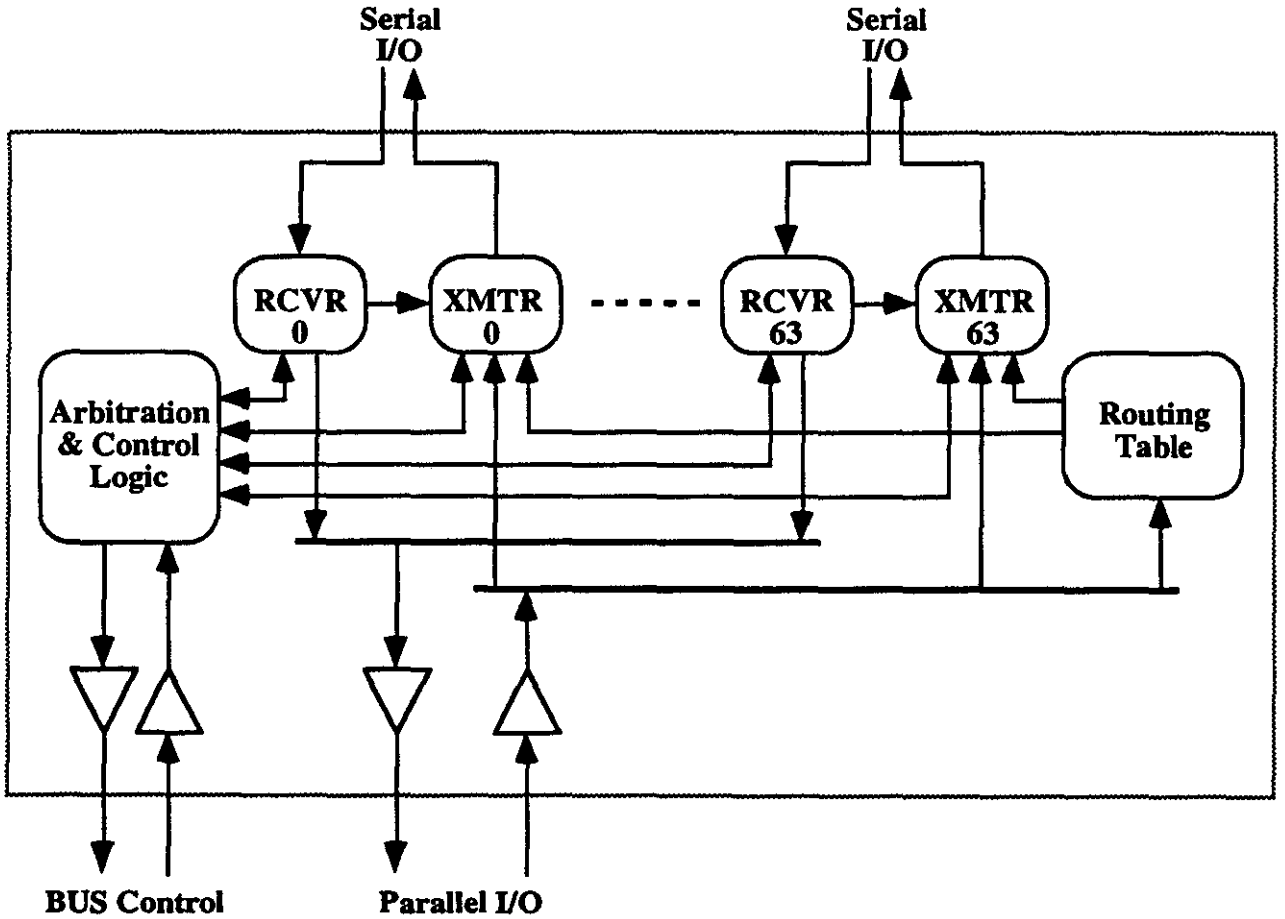

Figure 5. Control Network Interface Module.

The point-to-point links which connect individual EDD and EDC modules to the Control Network can be implemented using a custom protocol, or (more likely) a commercial protocol such as the Transputer OS or DS link.

If possible, the Control Network interface modules (figure 5) will reside in a single crate called the Control Router (figure 6). Messages are broadcast across the backplane as 64 bit single word writes. The destination field of the message is decoded by a routing table on each interface module. This enables the transmitter buffer for the selected output links. The routing table provides space for 64 K group codes. Any output link may be a member of any group. The routing tables are initialized by the System Monitor.

Messages are buffered at the input and output of the Control Network Interface module. If any buffer is full, the BUSY signal is asserted. Modules at the destination end of the link may transmit flow control (XON / XOFF) tokens to control the BUSY signal assertion. There are two priorities for messages in the Control Router. Messages that are not time critical are processed in a simple rotation. Messages from the Gating Logic are given a higher priority, since the Level 1 Accept may be inhibited. The Gating Logic interface module also contains the central bus controller.

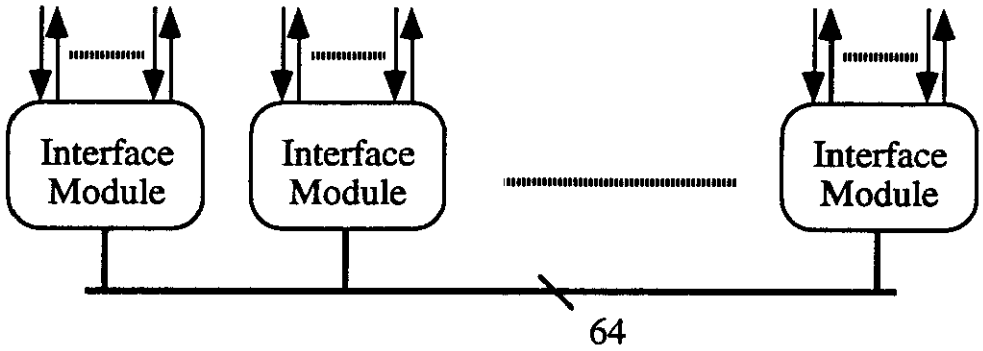

Figure 6. Control Router.

Trigger Supervisor

The Trigger Supervisor is a central control point in the system. Its main function is to assign EVENT_NUMBERs to Processors. It does not directly control data flow. All data flow and buffer management is handled by individual Processors and by the Event Data Collectors and Distributors. However, the Trigger Supervisor can help to balance data flow into the processors by ensuring a uniform distribution of event assignments.

The Trigger Supervisor receives the Level 1 Trigger signals and event type information from the Trigger system, identifies the partition and assigns the corresponding events to Processors. If the number of pending event requests from the Processors falls far behind the number of Level 1 Triggers, this indicates that the trigger rate is too high and the Trigger Supervisor may inhibit Level 1 accepts. The Trigger Supervisor may also inhibit triggers momentarily to allow transfer of synchronous control messages to the front-end system.

Level 1 event numbers are sequential within a partition. An EDC may be assigned to only one partition. This means that the event numbers are also sequential within an EDC and the EDC does not need to be aware of the system partitioning. The

Processor may only request data from EDCs in the partition corresponding to the assigned event. This is simplified by inclusion of partitioning information in the Control Network routing tables.

Gating Logic

The Gating Logic is responsible for distribution of the timed control signals. These signals include the 60 MHz system clock and crossing synchronous control. Initialization messages to the Gating Logic are used to map control outputs to specific partitions. The Level 1 Trigger system generates the Level 1 Accept and Trigger Type information used to select the partition. Functionally, independent Gating Logic, Trigger Supervisor and Level 1 Triggers exist for each partition.

Front-end Interface

The front-end modules in the GEM system are predominantly standalone (not crate based) devices. For this reason, we attempt to minimize the number of connections required to interface a front-end module to the Data Acquisition and Clock/Trigger Distribution systems. A minimal front-end interface consists of precisely timed **Clock** and **Crossing Sync** signals, plus serial **Control** and **Data Links** (figure 7).

Each front-end circuit should have an internal Event Counter to check event synchronization. The Event Counter is post-incremented by the Level 1 Accept signal. The value of this counter is included in the Data packet header. It is used in the readout to verify that data blocks belong to the same event. Separate checks of the counter can be made by reading the EVENT_NUMBER register. At startup, the first write to the EVENT_NUMBER registers in the front-end modules will synchronize the event counters.

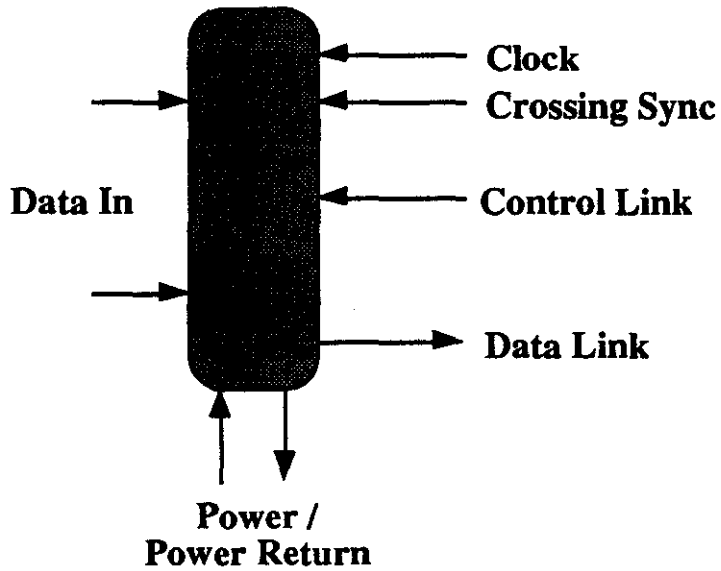

Figure 7. Minimal front-end Interface.

Clock and Crossing Sync

Clock is the global 60 MHz accelerator clock. **Crossing Sync** is a **Clock** synchronous signal indicating a specific clock cycle at which an action is to take place. In

the baseline proposal, **Clock** and **Crossing Sync** signals are multiplexed on a single optical fiber to a Control Fan-out module and then separated with a local PLL (figure 8a). This reduces the number of fiber links required and provides a locally regenerated **Clock** signal which is less sensitive to system noise. Connections from the Control Fan-out module to each front-end module are electrical. The use of a fan-out module reduces the overall system cost, but also reduces the system reliability and limits effective load balancing.

The front-end module receives the **Clock** and **Crossing Sync** signals from the Control Fan-out module. The Level 1 Accept from the Trigger system is "ORed" with all synchronous requests to form the **Crossing Sync** signal. **Crossing Sync** is then delayed by a programmable number of **Clock** cycles for coarse alignment. The **Clock** and **Crossing Sync** signals pass through a one clock programmable vernier delay for precise phasing. The **Clock** vernier must be set with a precision of about a ns, as **Clock** may be used for ADC gating and TDC time reference. The vernier adjustment requirements on the **Crossing Sync** are less demanding, as **Crossing Sync** only has to identify specific clock cycles.

The **Clock** and **Crossing Sync** signals are transmitted via the EDCs which distribute these signals to individual Control Fan-out modules. The driver and cable delays from a selected Control Fan-out module to the individual front-end modules are assumed to be matched to approximately 1 ns. If this goal cannot be met or if a **Clock** timing resolution of less than 1 ns is required, additional vernier delay on the front-end module will be necessary.

The source of the synchronous control signals used to generate **Crossing Sync** for a specific EDC or group of front-end modules is determined by partition mapping information in the Gating Logic. This information is preloaded by the Trigger Supervisor.

Control Link

The baseline design assumes that **Control Link** messages are transmitted asynchronously at one third the **Clock** rate (20 Mbps, ≤ 10 MHz NRZ) to allow oversampling. If the routing of the **Control Link** fiber is the same as the **Clock**, it should be possible to operate the **Control Link** synchronously at 60 Mbps (≤ 30 MHz NRZ). Each message includes a START bit and 16 message bits. The **Control Link** message is defined as follows, with the START bit (1) transmitted first:

	st.	*pg.*		
Control Link Message	1	P	Offset (7 bits)	Value (8 bits)

All messages are mapped into a generic memory space. The memory space is divided into 65,536 pages of 128 Bytes each. "P" is the Page_Select bit. If zero, then page 0 is selected. If one, then a page is selected according to the contents of the PAGE_SELECT_HIGH and PAGE_SELECT_LOW registers. Page 0 contains the most commonly used registers and is always accessible, regardless of the settings of the Page Select registers. The "Offset" field identifies one of the 128 registers on the selected Page. The "Value" field contains the 8-bit value to be written to that register. Read operations are implemented as split transactions. A message is sent with the "Offset" pointing to the READ register (register 20 in page 0). The "Value" field contains the Page bit and offset of the register which is to be read. The front-end module responds by transmitting the Value of that register on its output **Data Link** in standard "FE Message" format:

	st.	pg.		READ (20)						pg.	7 bits
"READ" Message	1	0	0	1	0	0	0	0	0	P	Offset (7)

	st.	Msg. Code								8 bits
"READ" Response	1	1	1	0	1	X	X	X	X	Value

This format allows eight million (256 X 256 X 128) registers per front-end module. If the front-end module requires fewer than 128 registers, then the PAGE_SELECT_LOW and PAGE_SELECT_HIGH registers need not be implemented.

The following registers in Page 0 are preassigned:

Offset		Function
(general mode/status registers)		
0F	R/W	TEST_LENGTH
(synchronous command registers)		
10	R/W	TEST
11-1F		reserved
(asynchronous command registers)		
20	W	READ
21	W	BLOCK_READ
22	W	BLOCK_WRITE
23	W	ECHO
24	R/W	EVENT_NUMBER
25	W	RESTART
26-2F		reserved
(Fan-out module registers)		
30	R/W	CHANNEL_MASK_0
31	R/W	CHANNEL_MASK_1
32	R/W	CHANNEL_MASK_2
33	R/W	CHANNEL_MASK_3
34	R/W	BROADCAST_ENABLE
35-37		reserved
(Clock phase adjustment registers)		
38	R/W	PHASE_ADJUST_COARSE
39	R/W	PHASE_ADJUST_FINE
3A-3F		reserved

(Compatibility registers)

40	R/W	Level 2 Accept
41	R/W	Level 2 Reject
42-47		reserved

(Maintenance)

48-4F	reserved for Boundary Scan

(Subsystem defined)

50-7F

"Registers" 30-37 in Page 0 are required if a Fan-out module implementation is used. These registers exist only in the Fan-out module (not in the front-end module). Registers in all other pages are subsystem defined.

A few commands to the front-end module must be crossing synchronous. Instead of dedicating a separate crossing synchronous control line to each of these signals, they are multiplexed into a single **Crossing Sync** signal. The interpretation of **Crossing Sync** depends on the last **Control Link** write to a synchronous command register. This approach is based on the following assumptions:

- There is no additional control information which must be supplied with a Level 1 Accept. This assumes that the number of crossings to be sampled and the internal processing of each event do not change on an event-by-event basis (as a function of trigger type or ambiguous crossing resolution in the trigger).

- All other crossing synchronous signals occur at a frequency which is low in comparison to Level 1 Triggers. In addition, it is assumed that the front-end modules do not directly use crossing synchronous accelerator information such as placement of the bunch 0 crossing or abort gaps for synchronization or calibration.

The following "synchronous command" registers are defined:

TEST	Following a write to this register, the front-end module should generate a test event at the next **Crossing Sync**. The number of successive **Crossing Syncs** to be interpreted as test events before returning to the default Level 1 Accept mode is determined by the content of the TEST_LENGTH register. The test type is determined by the content of the Value field. Test type 0, if implemented, should return a pre-defined event data packet.

Any **Crossing Sync** signal which has NOT been preceded by a write to a "synchronous command" register is interpreted as a Level 1 Accept. Level 1 Accepts will be inhibited between the transmission of the synchronous command and the **Crossing Sync** which executes that command.

A synchronous CROSSING_CHECK register was originally defined. It has since been removed because the Level 1 pipeline is self-restoring and very little diagnostic information was provided by the check.

"Asynchronous command" registers support commands which invoke an action in the front-end module but do not require a corresponding **Crossing Sync**. They include:

RESTART	Writing to this register causes the front-end module to clear its Event Data output buffer and ignore Level 1 Accepts until the next write to the EVENT_NUMBER register. The Data Link is still active for transmission of non-data messages. The Value field is either ignored or used to distinguish various restart options.
EVENT_NUMBER	On a write operation to this address, the front-end module should set it's internal Event Counter to the 8 bit Value field. If the module is currently in RESTART mode, it should resume processing Level 1 Accepts. On a read of this register, the front-end module should return the current value of its Event Counter.
READ	The front-end module should return the value of the register addressed by the current contents of the PAGE_SELECT_HIGH & PAGE_SELECT_LOW registers and the PAGE/Offset in the Value field.
BLOCK_READ	The front-end module should return the value of the register addressed by the current contents of the PAGE_SELECT_HIGH, PAGE_SELECT_LOW and BLOCK_PAGE_OFFSET registers. This address is then incremented by 1. The value field is ignored.
BLOCK_WRITE	The front-end module should write the value field of the command into the register addressed by the current contents of the PAGE_SELECT_HIGH, PAGE_SELECT_LOW and BLOCK_PAGE_OFFSET registers. This address is then incremented by 1.
ECHO	The front-end module should return (echo) the contents of the Value field in a Data Link FE message.

"General Mode/Status" registers hold static information used in executing other commands. They include:

BLOCK_PAGE_OFFSET	This is the Page bit and Offset used as the starting value in a BLOCK_READ or BLOCK_WRITE command.
PAGE_SELECT_LOW	Low 8 bits of the Page select register.
PAGE_SELECT_HIGH	High 8 bits of the Page select register.
STATUS	Front-end module status (bits TBD).

FE_LOGICAL_ID_0	Unique module ID (low byte), assigned during the initialization phase.
FE_LOGICAL_ID_1	Unique module ID (high byte), assigned during the initialization phase.
MODULE_TYPE	Fixed (non-volatile) module type identifier.
SERIAL_NUMBER_0	Fixed (non-volatile) module serial number.
SERIAL_NUMBER_1	Fixed (non-volatile) module serial number. Note: if unique module SERIAL_NUMBERs cannot be implemented, the value of these registers default to the FE_LOGICAL_ID.
TEST_LENGTH	This register contains a count of the number of **Crossing Sync** pulses (up to 15) to be interpreted as TEST events before reverting to Level 1 Accepts. If the test length is greater than 1, all TEST **Crossing Syncs** must occur within the Level 1 pipeline delay so that the **Crossing Sync** signal can revert to Level 1 Accept in time to sample the test event data.

The following registers are used by the Fan-out module (if implemented):

CHANNEL_MASK_0	Individual bit enables for front-end modules 0-7. If a bit is set, then commands are forwarded to the front-end module.
CHANNEL_MASK_1	Individual bit enables for front-end modules 8-15.
CHANNEL_MASK_2	Individual bit enables for front-end modules 16-23.
CHANNEL_MASK_3	Individual bit enables for front-end modules 24-31.
BROADCAST_ENABLE	When zero, disables broadcast of commands to all front-end modules (overrides current channel mask settings). When one, enables broadcast of commands to all front-end modules (overrides current channel mask settings). Any other value uses current CHANNEL_MASK settings.
PHASE_ADJUST_COARSE	Integral clock delay at Fan-out module. Affects all front-end modules.
PHASE_ADJUST_FINE	Vernier adjust of clock delay at Fan-out module. Affects all front-end modules.

The following registers are defined for compatibility with a hardware-based Level 2 Trigger System:

LEVEL_2_ACCEPT	A write operation to this register enables transmission of the next event in the Level 2 buffer.

LEVEL_2_REJECT	A write operation to this register disables transmission of the next event in the Level 2 buffer.
Boundary Scan	These registers are reserved for IEEE-1149 Boundary Scan operation of the front-end modules (TBD).

Control Link Options

Several additional options are still under consideration in the implementation of the **Clock** and **Control Links** (figure 8):

1) if the operating frequency is raised to 120 Mbps, the **Clock, Crossing Sync** and **Control Link** can all be encoded on the same optical fiber. This saves the cost of the second optical link to each Fan-out module.

2) the Fan-out modules may be removed, and the control signals supplied directly from the EDCs to each front-end module (figure 8b). This increases the number and cost of fiber links but eliminates the cost of the Fan-out modules. It also allows simple load balancing through the reassignment of links at the EDC (not possible with a common Fan-out module).

3) a standard front-end interface IC may be developed. This IC would include the phase-locked loop and any required serial encoding/decoding logic. It can optionally include serial/parallel conversion logic and decoding of **Control Link** messages. The interface IC would also provide a Boundary Scan test controller and miscellaneous module services such as module ID registers. This option probably includes the previous two options.

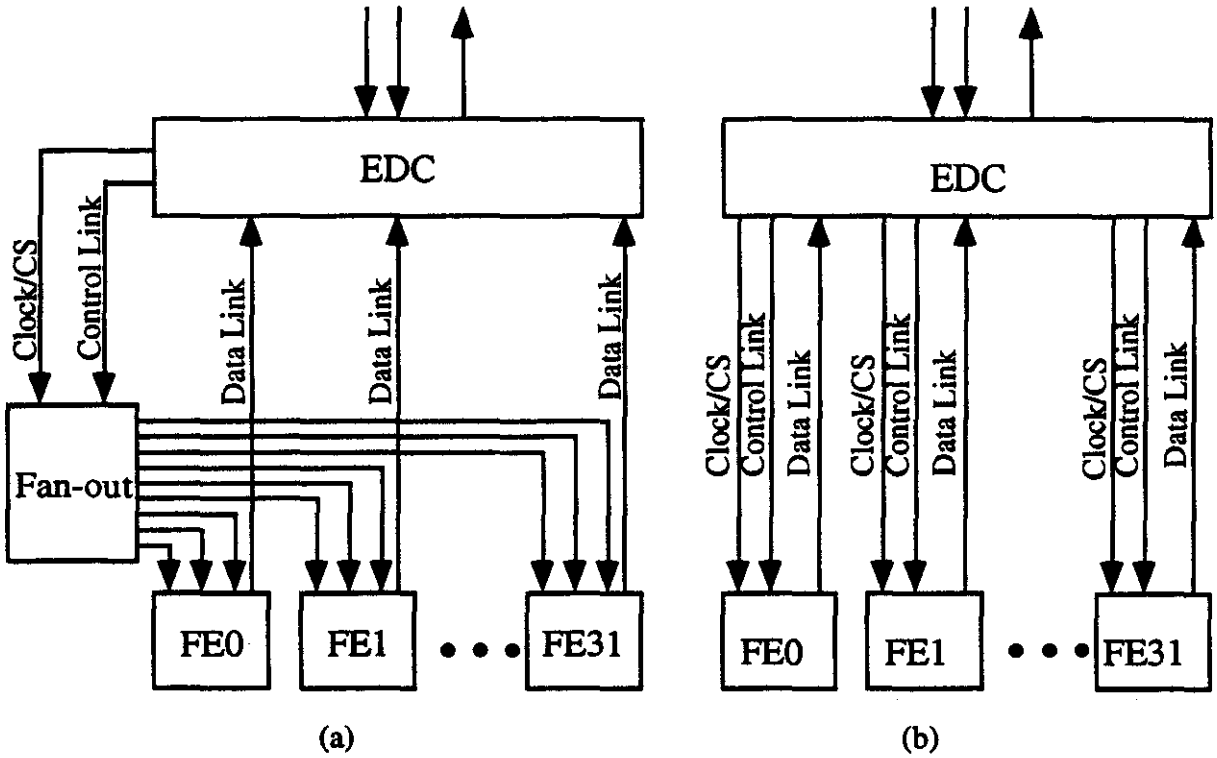

Figure 8. Front-end Signal Distribution.

Data Links

Data Links are 60 Mbps (≤ 30 MHz NRZ) synchronous. The same global 60 MHz reference used by the front-end transmitters is available at the EDCs, so there is no need for clock extraction at the receiver. The sampling clock for each EDC input channel is selected from **Clock** or **Clock-Bar** (whichever best matches the phase of the received data stream).

If may be feasible to use a direct coupled receiver for the 60 Mbps links. This would reduce the overall power requirements in the front-end module by eliminating the continuous transmission of synchronization frames required in DC balanced protocols. To obtain a BER of 10^{-14} or better, the received signal should be at least 3 dB above the listed receiver sensitivity at standard 10^{-9} operation. If high speed links are used in place of the standard 60 Mbps links, they will necessarily use an encoded (DC balanced) protocol. The Event Data Collector link receivers could be implemented on daughter boards to allow the use of alternate links.

The Data Link format is bit and frame synchronous, with 1 start bit and 16 data bits per word. A zero in the start bit position of a multiple word event data packet identifies End-of-packet There may be any number of idle (zero) bits between packets. The Data Links support both data and message traffic as determined by a code in the most significant bits of the first word in the packet.

Words are transmitted Most Significant Byte and Most Significant bit first. The preferred format is shown in figure 9. The first word of each packet contains a fixed pattern, status information and the FE_Event_Number. The patterns distinguish data from messages. The FE_Status field is defined by the subsystem. The FE_Event_Number is used by the EDC as a check that the correct event is in the input queue and also as an additional check of the previous event's end-of-packet marker.

	st.	*Data Code*					
Normal FE Data Packet	1	0	1	1	0	FE_Status	FE_Event_Number
	1	Data 1					
	1	Data 2					
	1	. . . . .					
	0	Data N (may be checksum)					

Empty FE Data Packet	1	0	1	1	0	FE_Status				FE_Event_Number							
	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

	st.	*Msg. Code*								
FE Message	1	1	1	0	1	X	X	X	X	FE_Message

Figure 9. Recommended Packet Formats.

If one of the FE_Status bits is used to denote empty data packets, the second word of the "Empty FE Data Packet" format can be deleted. An event data packet is expected from each front-end module for every event. Absence of an event packet is interpreted as a failure. If a failure is detected, the EDC will remove the module from its readout list and notify the System Monitor.

When running at 60 MHz, a link can send 3.5 million 17-bit words per second. Each event has an overhead of one word. Hence, at a trigger rate of 100 kHz the total overhead is 100,000 words, plus one START bit for every 16 data bits. Total overhead at maximum load is about 9%, mostly caused by the start bits. Note that the average data size per event per front-end cannot be larger than 68 Bytes at the maximum trigger rate. At lower rates the packets can be larger. An absolute maximum event size is set by the receiver FIFOs in the EDCs. Under special running conditions, event packets slightly smaller than the full FIFO size can be read out.

Suggested uses of the status bits in the Data packet header are shown below:

User Defined	Throttle_Req	FE_Data_Lost	FE_Error

- The FE_Data_Lost status bit indicates that data from this front-end has been suppressed for this event. This may be related to buffer overflow due to high occupancy or trigger rates.
- FE_Error indicates that the data may be affected by errors on the front-end board.
- A front-end can set the Throttle_Request status bit when internal buffers get dangerously full. The EDC will generate a THROTTLE_REQUEST message to the gating logic in response.

Message packets are transmitted on the same link as event data packets. A message packet may only be transmitted between event data packets. It is always only one (17 bit) word long. At present, front-end modules send message packets only in response to commands. They do not initiate message transfers.

If a standard front-end interface IC is developed, the **Data Link** bandwidth may be increased to 120 Mbps. This would alleviate the possibility of **Data Link** overflow in some subsystems at very high trigger rates.

Event Data Collectors

The Event Data Collector (figure 10) receives data from up to 32 sources on individual 60 Mbps fiber links. As an alternative, a smaller number of higher speed data links may be used in place of the 60 Mbps links. The sum of the average data throughput of these links should not exceed 100 MBytes/sec. Each **Data Link** is connected to a FIFO buffer with a common output bus driving a large common memory. The common memory must support simultaneous read and write at 100 MBytes/sec (200 MBytes/sec total). Sixteen bit wide interleaved, synchronous DRAM or Video DRAM is assumed.

The following is a description of a very simple, though inefficient, EDC memory management scheme (refer to figure 11). It assumes a large circular buffer for event data, which is overwritten after some period of time greater than the maximum allowed readout + Level 2 Trigger latency. At a 100 kHz trigger rate, the allowed latency would be approximately one second for each 50 MBytes of EDC memory. At lower trigger rates, the latencies can be proportionately longer. Although the average latency is much less than a second, the peak latency can be several seconds. If the Level 2 processing time of an event is approaching the maximum allowed latency, the processor must interrupt processing of the trigger algorithm and issue a request for all remaining data.

Data is written into EDC memory in event order, with each new event starting on a page boundary. At the peak event rate of 100 kHz, the event size is limited to an average of 1000 bytes per EDC. A page size of 256 bytes provides a reasonable compromise

between excessive memory utilization and control overhead. This page size is also used in the Event Builder. A pointer to the first page of each event (EVENT_PTR) and the number of pages used to store the event (PAGE_COUNT) are written to the Event Directory in the EDC controller local memory. This is a fixed size directory indexed by EVENT_NUMBER.

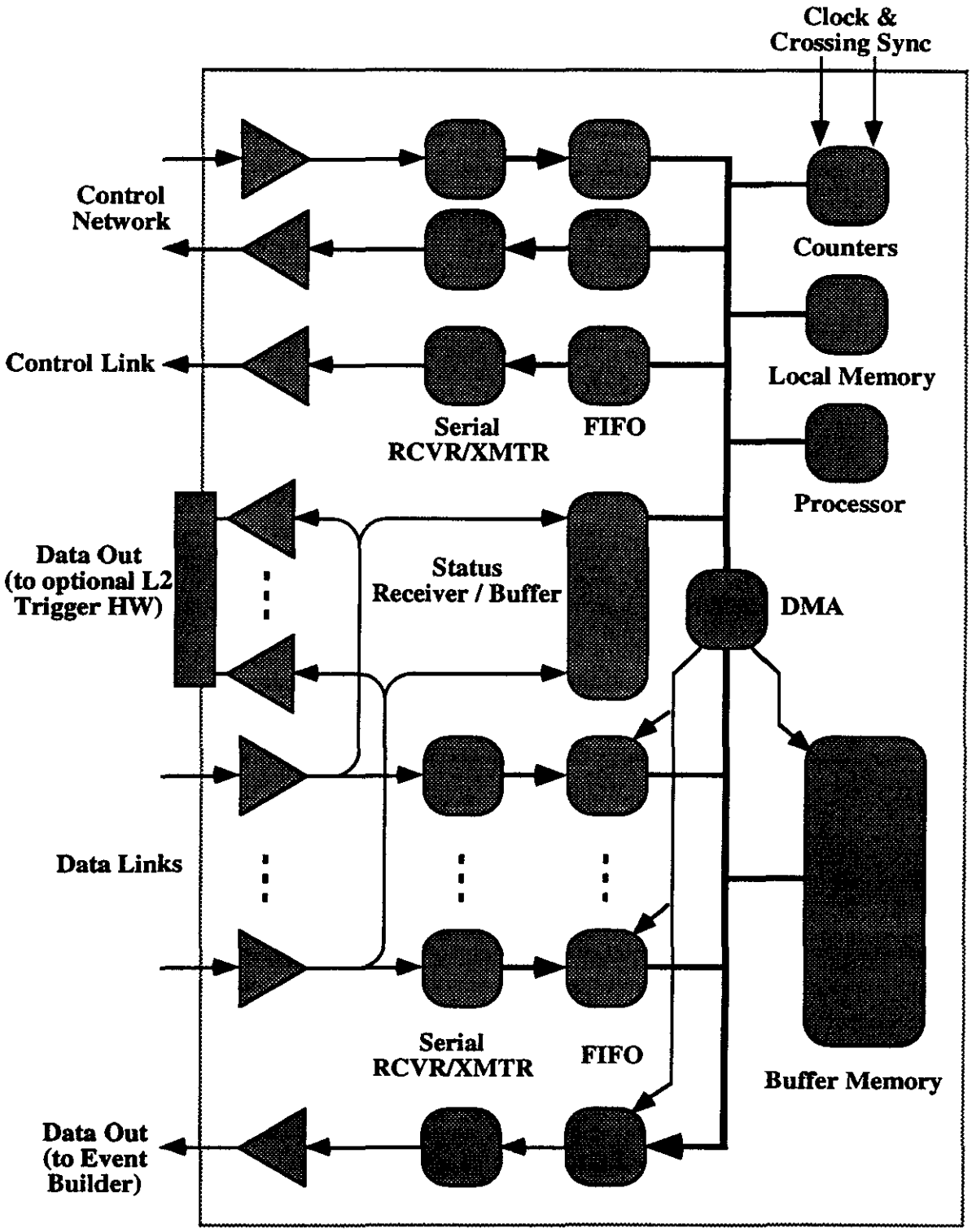

Figure 10. Event Data Collector.

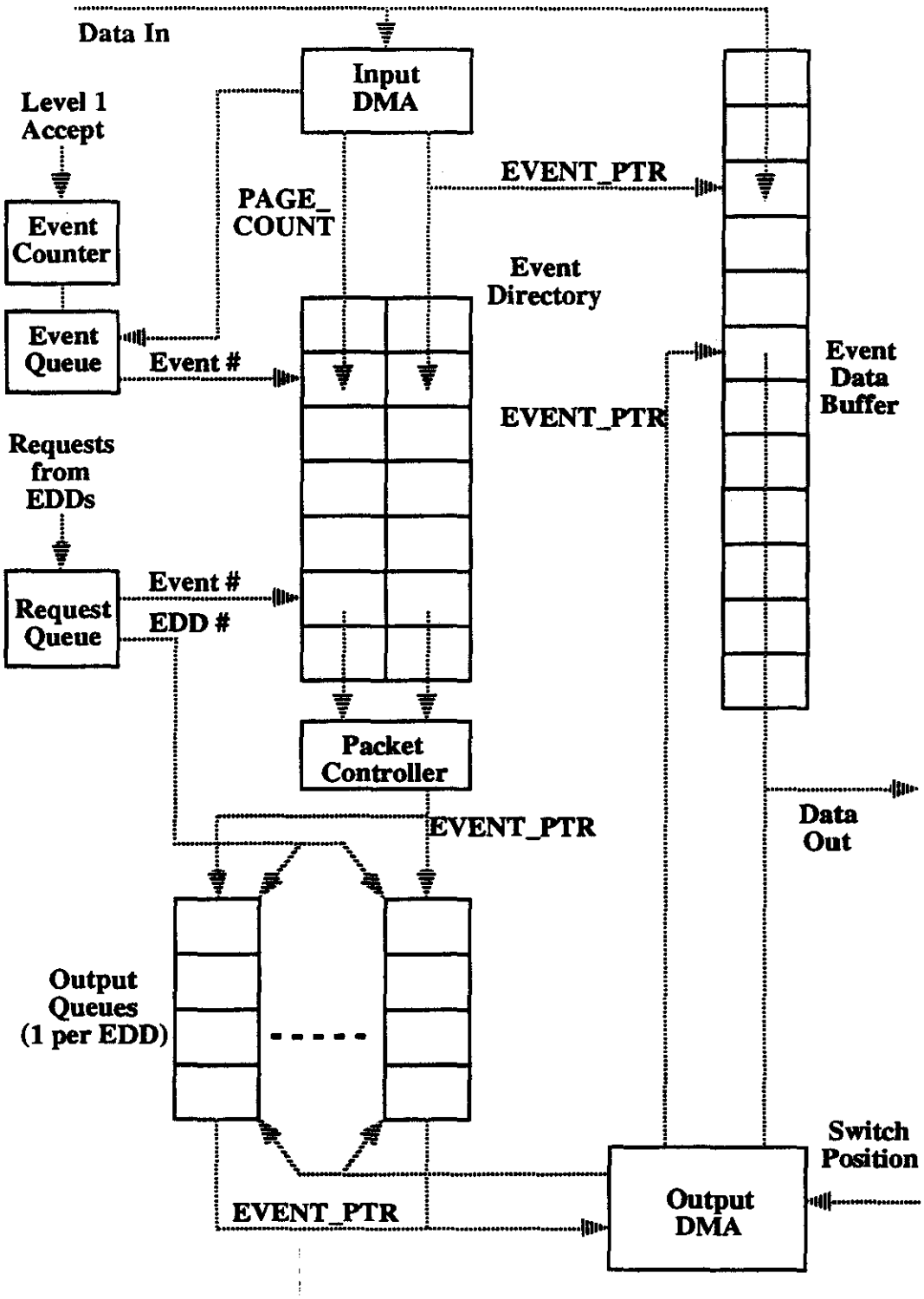

Figure 11. Basic EDC Memory Management.

The EDC controller also maintains a logical output queue for each EDD in the system (up to 512). When the controller receives a message requesting a specific event, it gets the EVENT_PTR and PAGE_COUNT for the event from the Event Directory and writes the page pointer(s) for that event to the appropriate output queue. The Output DMA controller cycles through the output queues in the order of the Event Builder switch

rotation. At each switch setting it sends one page of data from the selected queue. If there is no data to be sent, a null packet is transmitted.

A more efficient memory management scheme would used a linked list to allocate memory pages non-sequentially (as in a standard disk file structure). A free-page stack provides the next available empty page. EDC event memory requirements are greatly reduced at the expense of higher controller overhead and increased controller local memory.

Level 2 Trigger Interface

Data from the individual front-end modules to the EDCs are transmitted on fiber-optic links. The EDC receivers convert these optical signals to electrical signals before decoding the serial data stream. Differential Pseudo-ECL buffers are provided on the EDC to supply these serial data streams to external devices such as a hardware Level 2 Trigger system (figure 12).

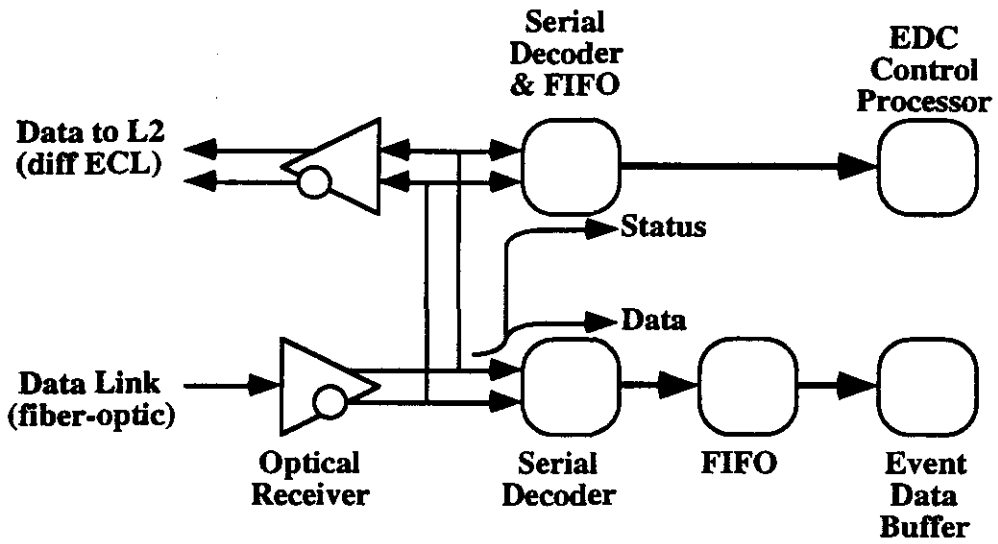

Figure 12. Level 2 Interface.

Level 2 hardware must provide the serial decoding and event building functions for the data streams which it monitors. The result of the Level 2 calculation can then be stored in another EDC for access by the Level 2/3 processors. The original data are retained in the same EDCs which translate the data for input to the hardware Level 2 trigger.

EDC Data Format

The EDC builds the FE data packets into larger event fragments. The original FE data packet is transformed into a standard format which is acceptable to higher level processors (figure 13). Start bits are stripped and the original front-end packet is prefixed by a word count, a front-end ID and the data descriptor. The count, in units of 16-bit words, includes the word count itself. The FE_ID is a unique module identifier assigned during the initialization phase. It serves as a geographical address. FE_DD describes the format of the data and the format version currently used. For empty packets only word count, FE_ID, FE_DD and the original packet header are included. Optionally the EDC can

suppress empty data packets for which the status is zero. Overhead from headers is of the order of 12% at the maximum data rate.

FE Data Packet

FE Packet Word Count					
FE_ID					
FE_DD					
0	1	1	0	FE_Status	FE_Event_Number
Data 1					
Data 2					
. . . . .					
Data N					

Empty Data Packet

FE Packet Word Count (= 5)															
FE_ID															
FE_DD															
0	1	1	0	FE_Status				FE_Event_Number							
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 13. Front-end data format after event building in EDC.

The EDC prefixes the front-end data packets with a header that contains the overall word count, a unique identifier, the EDC data format descriptor, the EDC event number and the EDC status (figure 14). The front-end event number has too few bits to be useful. It is substituted by an event number calculated by the EDC which has sufficient bits to serve as a unique event identifier while the event is processed by the level 2 trigger. The level 2 and level 3 trigger may add another event number.

EDC Data Packet

EDC Packet Word Count					
EDC_ID					
EDC_DD					
EDC_Event_Number (m.s. Bytes)					
EDC_Event_Number (l.s. Bytes)					
				EDC_Status	Reserved
FE Packet 1					
FE Packet 2					
. . . . .					
FE Packet 32					

Figure 14. Event data format used by EDC.

Suggested uses of the EDC_Status bits are shown below:

Reserved	Reserved	EDC_Data_Lost	EDC_Error

The EDC_Data_Lost status bit indicates that data from this EDC has been suppressed for this event. This may be related to buffer overflow due to high occupancies or trigger rates.

EDC_Error indicates that the data may be affected by errors in the EDC.

Event Builder

Several options have been considered for the switching network component of the Event Builder. These include a basic Time Division Multiplexed (TDM) switch, a general-purpose commercial switching network and a network based on the Scalable Coherent Interface (SCI) standard.

The advantage of the basic TDM approach is that there is no external control required. The switch is completely non-blocking with no arbitration or handshaking. No packet length, source or destination headers are required in the data packets because the packet size and order of interconnection are fixed. The disadvantage is that the bandwidth may not be used very efficiently for random traffic patterns. Traffic in classic event building applications is fairly uniform, so the efficiency can be relatively high, but this is not necessarily the case for the selective readout option discussed earlier.

For random traffic, a general-purpose commercial switching network (e.g., ATM or Fibre Channel) may have higher efficiency. Existing Fibre Channel switches are optimized for large data transfers, so many events must be concatenated to reduce overhead. Concatenation of events increases the buffer requirements and eliminates the selective readout option. Source and destination headers in the data packets are required for routing. Production cost of a commercial switch is higher due to the general-purpose control and bi-directional data path, but development cost should be lower.

A general-purpose switch is self-synchronizing. On the first set of events, all sources may attempt to send a packet to the same destination. All but one is blocked. On the second set of events, the source which was not blocked on the first set is free to send data to the second destination while the remaining sources all arbitrate again for the first destination. Eventually, all sources are skewed by approximately one event set and blocking is minimal.

An Event Builder based on SCI has also been considered. An SCI network would provide higher efficiency for the selective readout mode, but may not be cost-effective for the full readout mode. For total data rates above 8 GBytes/sec, an SCI Event Builder will need a very high speed crossbar switch as the center stage of the network.

The synchronous TDM switch is used in the baseline design because it is thought to be the least expensive (although not necessarily the most efficient) approach. It is implemented as a modular three stage network with a maximum size of 512 X 512 channels. The initial configuration provides 512 inputs and 128 outputs. This is accomplished as shown in figure 15, with 64 modules in the first stage and 16 modules in each of the second and third stages. The second and third stages together are grouped into 64 X 64 blocks (figure 16), so that the system can be expanded in increments of 64 channels. The same 8 X 8 module is used in each stage.

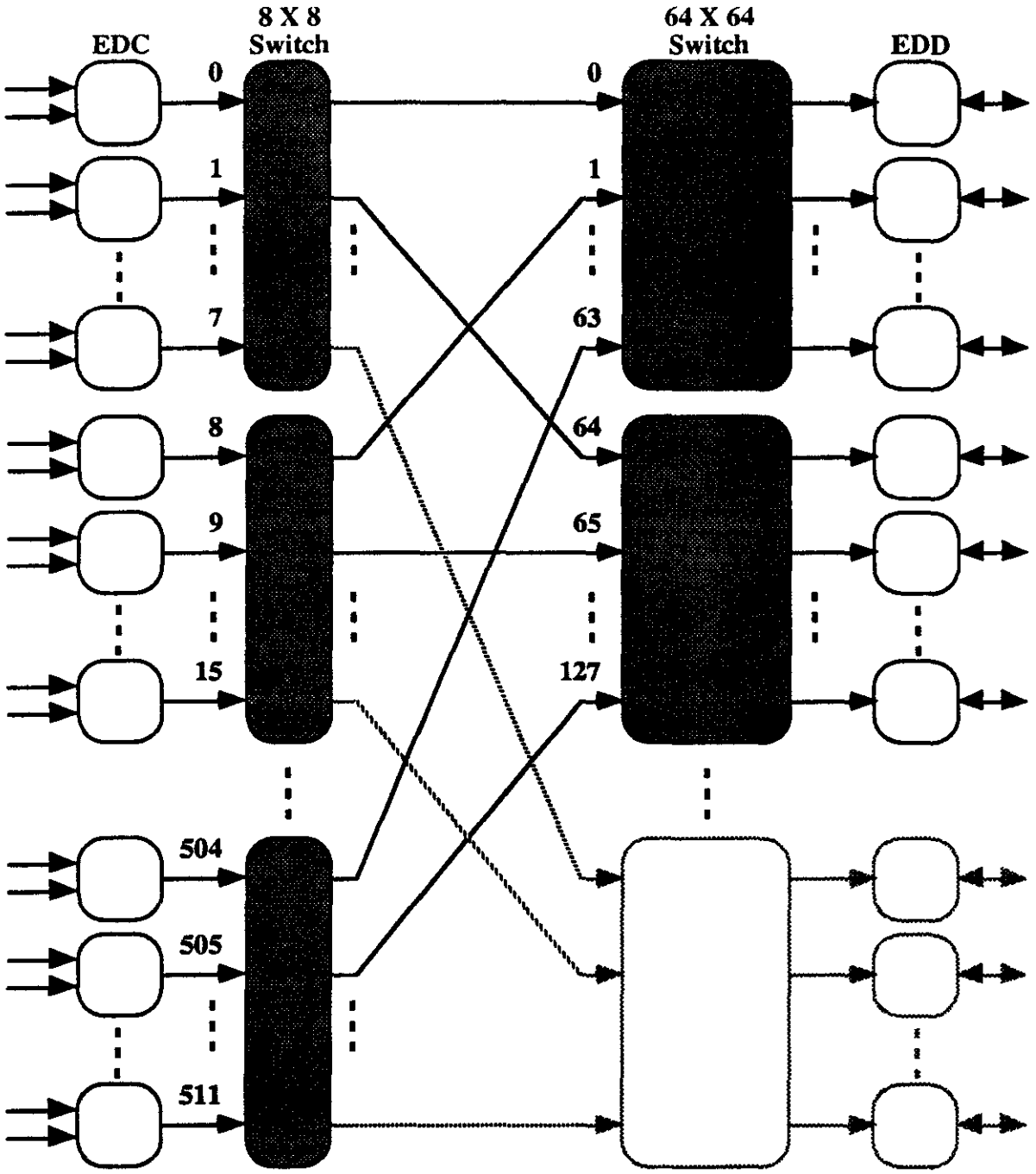

Figure 15. 512 X 128 Modular Switch (expandable to 512 X 512).

Figure 17 shows a single 8 X 8 module. It contains an 8 X 8 VLSI crossbar switch and configuration memory. The configuration memory holds the routing information for each time slot. The interconnection pattern is typically programmed as a synchronous barrel shift rotation, forming a conventional TDM switch with equal bandwidth allocated to each combination of input and output. Inputs and outputs are 1 Gbps (100 MBytes/sec) serial links. Switch modules are decoupled by a small FIFO on each receiver, which eliminates the need for synchronization at the bit or word level. Internally, data transfers are 16 bits wide (at 60 MHz) to avoid the design problems associated with switching of GHz serial signals.

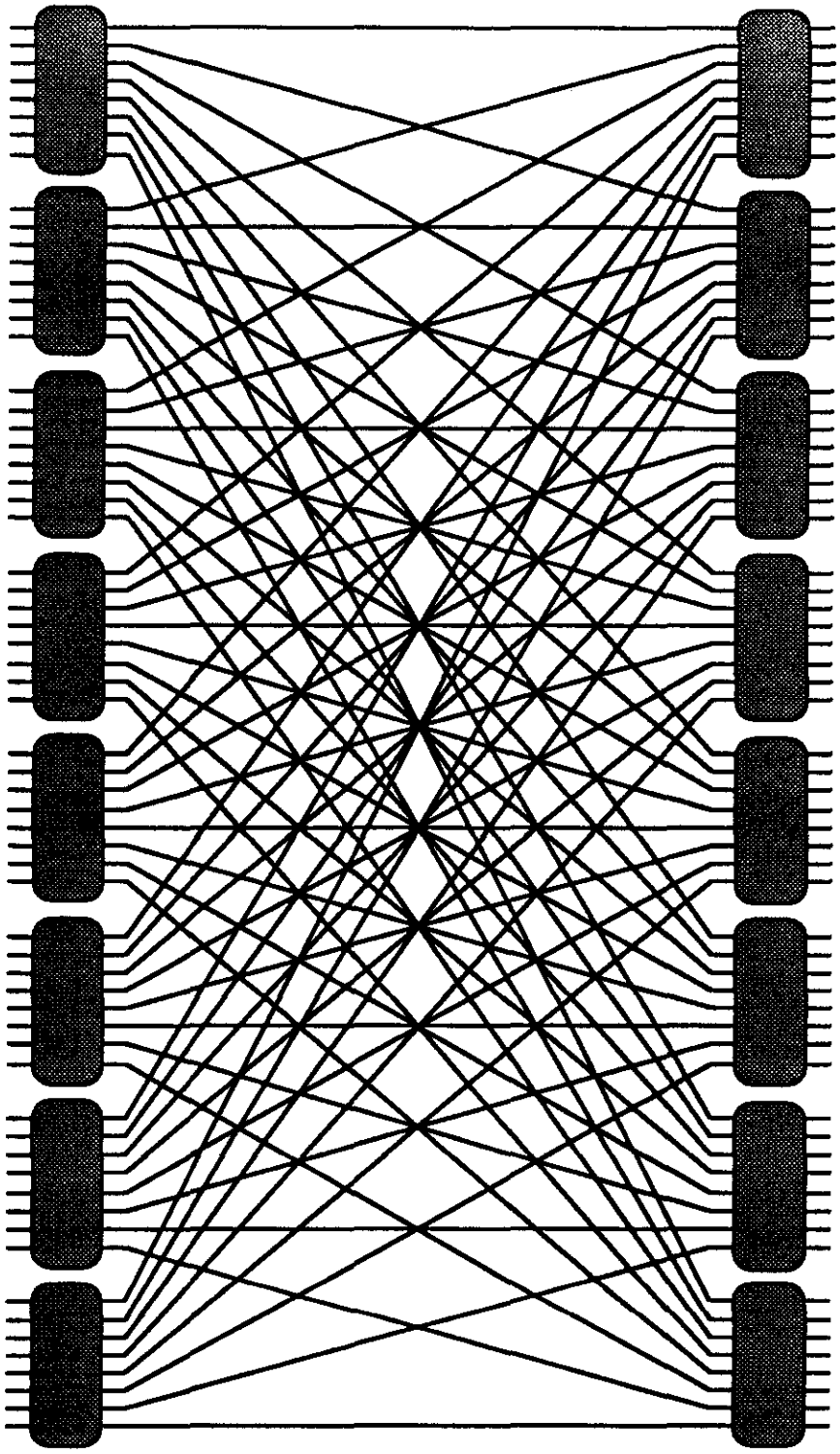

Figure 16. 64 X 64 Switch (single crate, 6.4 GByte/sec bandwidth).

For a 512 X 512 channel switch, the TDM operation provides the equivalent of 262,144 virtual channels, each operating at approximately 200 kBytes/sec. In the initial 512 X 128 configuration, the peak bandwidth is 12.8 GBytes/sec. This is expandable to 51.2 GBytes/sec in increments of 6.4 GBytes/sec.

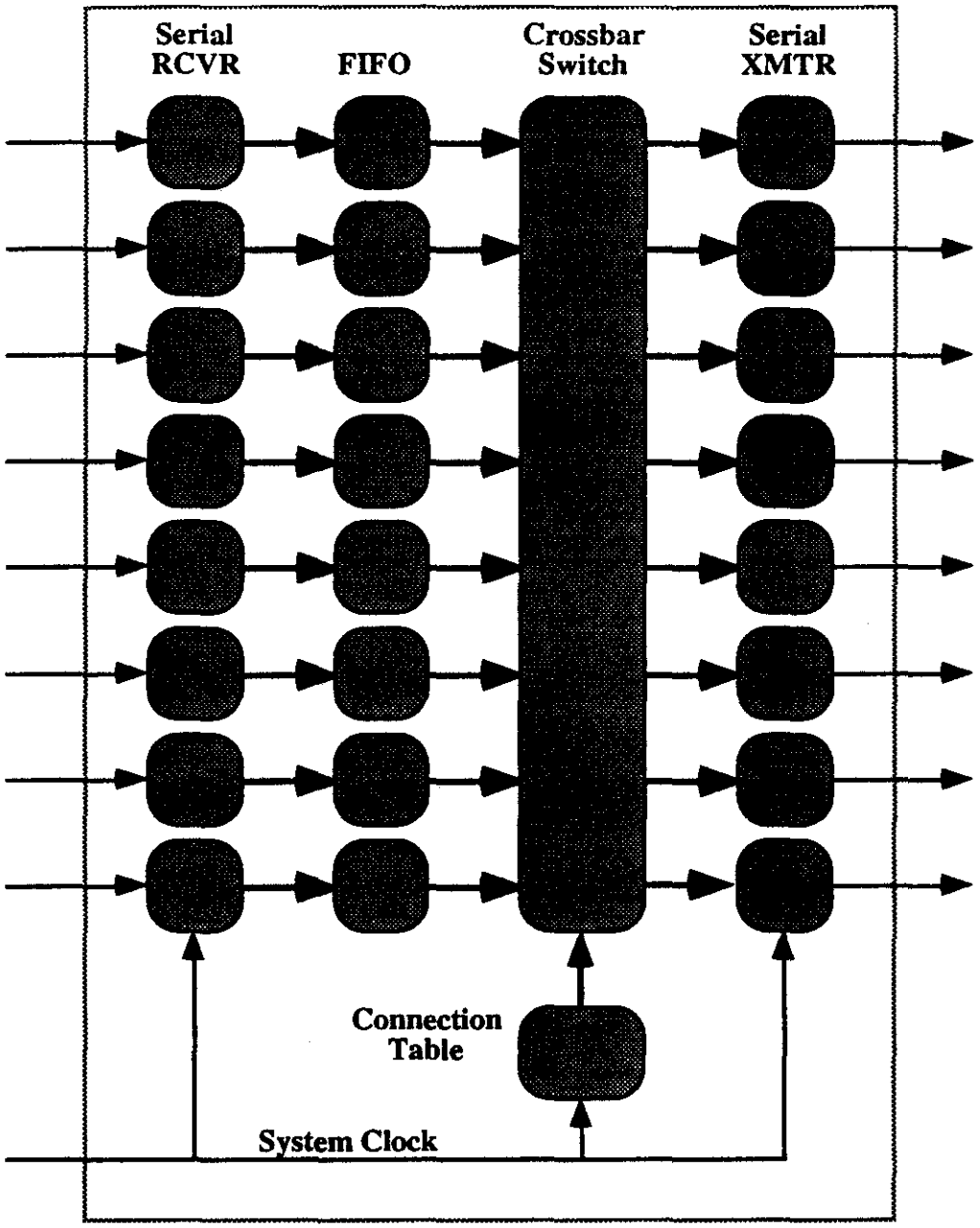

Figure 17. 8 X 8 Switch Module.

Switch efficiency is increased by the use of full bypass input queuing in the EDCs (figure 18). To eliminate "head-of-line" blocking, there is a separate logical FIFO buffer for each virtual switch channel. Based on the current switch position, a packet is transferred from the EDC buffer corresponding to the connected EDD.

Event Data Distributors

An EDD (figure 19) receives data from some or all of the EDCs and assembles the data to form a partial or full event for transfer to a processor. The EDD-Processor connection is a 1 Gbps bi-directional link. Protocol for this link is defined by the processor and is assumed to be a commercial standard (e.g., Fibre Channel). To accommodate the selective readout mode, the EDD buffers the data request messages from the proces-

sors. Data from a given EDC arrives at the EDD in the same order as it is requested. By inspecting the buffered processor data requests, the EDD can determine which EDCs are expected to supply data for a given request.

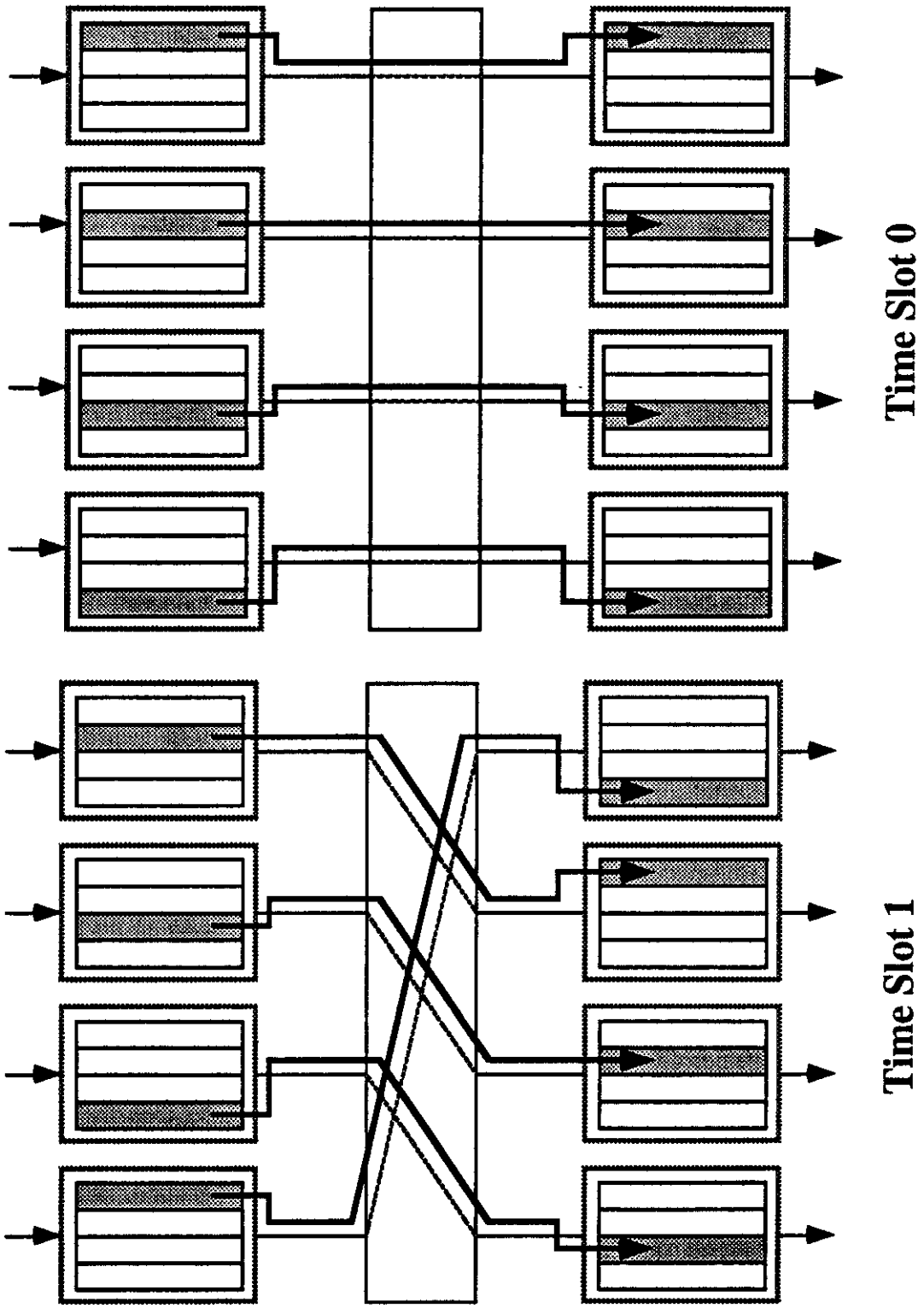

Figure 18. Input Queuing (in a 4 X 4 Event Builder).

The EDD maintains a pointer queue (figure 20) for each EDC in the system (up to 512). As data packets arrive from the EDCs they are stored in the Event Data buffer and a pointer to each packet is stored in the pointer queue corresponding to the connected EDC. The EDD Output Controller loops over the output of these 512 pointer queues and, based on the leading WORD_COUNT in the header of the first packet, copies one or more packets to the output link. If all data packets forming an EDC event fragment have not yet arrived, the EDD waits at that queue (subject to system time-out). If the routing table indicates no data are expected from an EDC, the EDD skips that queue.

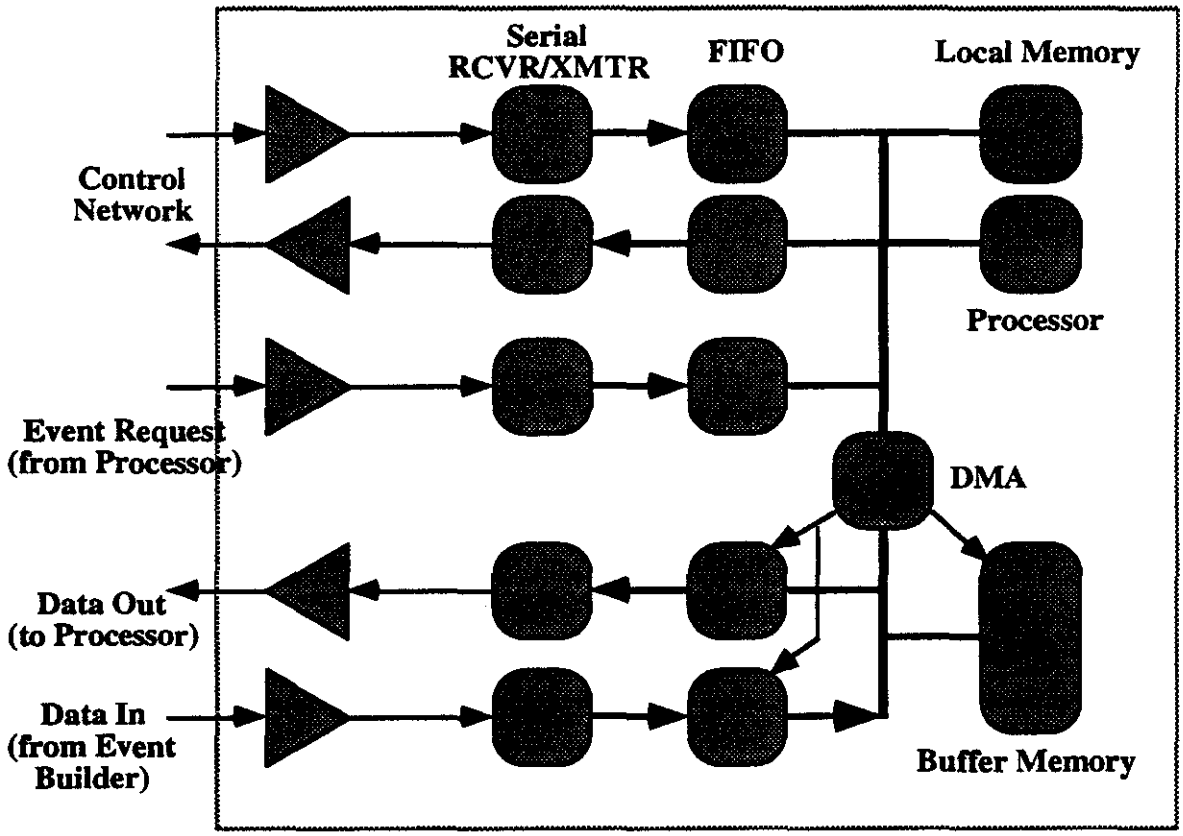

Figure 19. Event Data Distributor.

When issuing a data request to the EDD, the processor provides a group address (for EDC mapping), event number, process identifier and address pointer (for the returned data). The EDD does not add any header information to the collected data. When all the data has been written into processor memory, the EDD writes an interrupt message with the original process identifier.

Software

System Monitor

The System Monitor will be a set of processes running on a Workstation under a UNIX environment that supports networking and X-windows. The processes on the System Monitor will include alarm handling, diagnostics, system configuration and run control. At run time, the system monitor will receive a history of errors for each range of events; the history of errors is logged to the tape server. Using client-server technology or RPC the system monitor will share data or be a slave to the Global Control facilities

(i.e. EPICS). An object oriented methodology and productivity tools will be used to reduce software development time and maintenance cost. Figure 21 illustrates the run time software components of the System Monitor.

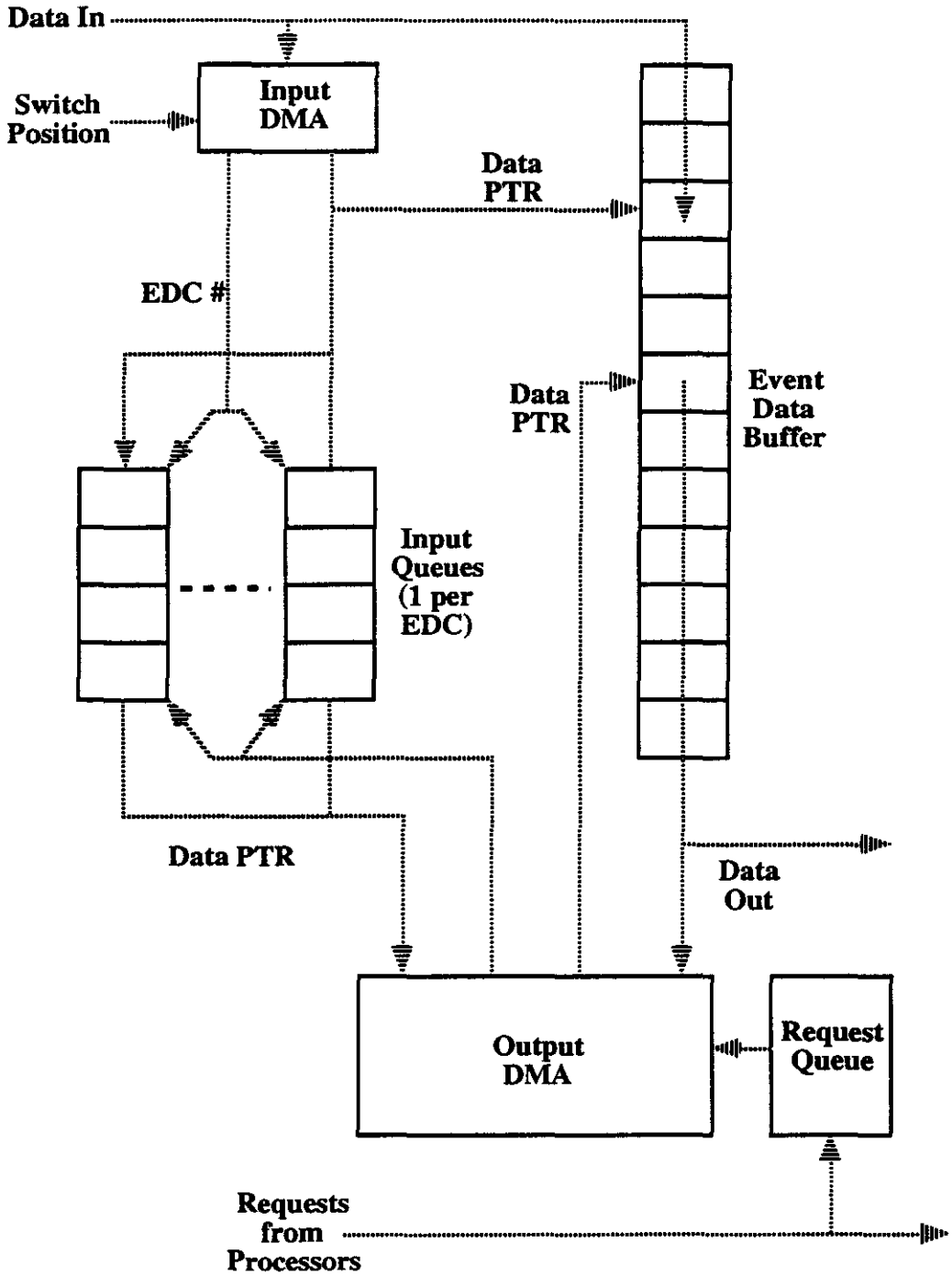

Figure 20. Basic EDD Memory Management.

The software development efforts will include: (a) Control Network Application Interface (CNAPI) to allow tasks in standard UNIX environments to communicate via message packets to the control network. The message packets will be 64 bits wide, with source (16 bits), destination (16 bits), command (8 bits) and data (24 bits). (b) Commu-

nication Device Driver or Demon to service the CNAPI and deliver the packets to the carrier. (c) Tape Server Application Interface (TSAPI) to allow the system monitors and farm processors to deliver data and the history of error detection to the tape server. (d) System Configuration tools to generate configuration files describing the sub-detectors that will take place in the run. (e) Loader to load all necessary software modules as described in the configuration file. When running in diagnostic mode, it will allow loading a real time kernel to the EDD and EDC. (f) Diagnostic tools to provide low level access to the front-end modules through the CNAPI. The FE developers will be able to access any address range (Control, Status and Data registers) that are visible at the EDC level. (g) Run Control GUI to enable start and stop data taking according to the configuration file. (h) Transaction Logger to keep track of the operator requested transaction, errors and alarms with appropriate time stamps. (i) Interfaces to allow calibration of subsystems (j) Software facilities to enhance security and fault tolerance (k) A buffer manager and application interface (BMAPI) to distribute messages to/form the control network.

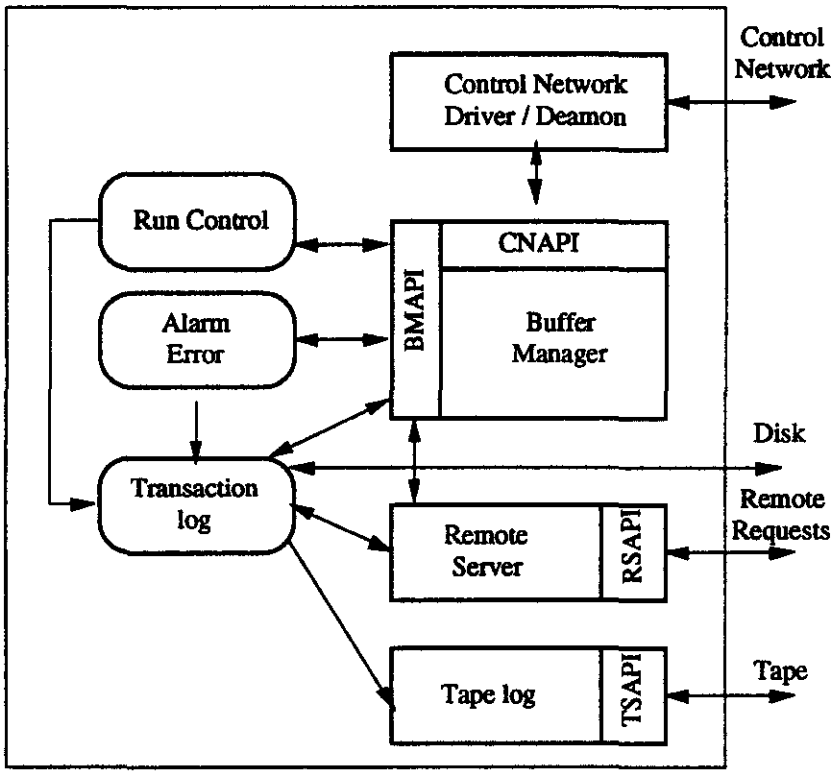

Figure 21. System Monitor Software Components.

Processor Farm Interface

Each processor group has a 100 MByte/sec input data link from the EDD, and a 1-2 MByte/sec output data link to a tape server (or network) for archival. In the full readout mode, the EDD transmits a full event record (200-400 kBytes) to the processor at a rate of approximately 250 events per second. The average processing time per event is approximately 4 ms. In the selective readout mode, the event data fragments received from the EDD are a few kBytes in size at up to 10,000 per second. Therefore the processor farm will have, on average, 100 μsec of time to receive and process the data. A CPU or thread processing the event data fragment will be blocked for 20-30 ms waiting for each packet: the EVENT_ASSIGN and the EVENT_DATA packet. If a multipurpose CPU and kernel is used, additional time will be spent in task management and in the device driver layers. Figure 22 illustrates a thread in the processor farm:

```
int ev, size;
int src = ctlNetOpen();
int dst = logOpen();
while( ctlNetRunning() )
        {
        ev = ctlNetEventRequest( src, .. );          /* blocks here 30 ms */
        ctlNetGetEvent(ev, buffer, &size, .. );      /* blocks here 30 ms */
        if (accept_event(buffer, size))              /* L3 accept */
                logWrite(dst, buffer, size);         /* if true, blocks here */
        }
ctlNetClose(src);
logClose(dst);
```

Figure 22. Processing Farm Thread for L3 accept.

To minimize interference with the farm's processing ability, the input and output data links should be driven by a dedicated CPU or DMA controller. The data should be delivered into the threads address space to avoid multiple copies of the same data (i.e. from kernel to user space). For each EVENT_DATA request, the processing farm will receive from the EDD a data stream containing the event data followed by an interrupt message containing a subset of the original data request message (CPU or thread ID and event number). The event data is ordered by EDC number. Each block of EDC data is contained in a set of one or more pages. The page size is constant and tentatively 256 bytes. The first page contains a word count and event number, the following pages contain only event data, and the last page may not be completely full of data.

Event Data Distributor

The EDD has four sources of data input, and same number of data outputs. The EDD acts as a bridge. (a) At peak rates 1,000 EVENT_REQUESTs are received from the processor farm and passed to the control network. (b) At peak rates 10,000 DATA_REQUESTs are received and passed to the control network. With each data request, the farm will send a list of EDC numbers needed to satisfy the request. (c) Event data fragments are received from the EDC through the Event Builder and passed to the corresponding processor. (d) Other 64 bits messages interchanged between the control network and the EDD used for status, loading and diagnostic purposes.

Once all the EDC pages for an event have arrived, the EDD will write the data to the farm at the buffer location specified in the DATA_REQUEST message. The EDD is an intelligent board with two modes of operation, at run time only embedded code or a single thread of execution will be available. For diagnostic, calibration and configuration a real time kernel will be loaded. The EDD software development efforts will include: (a) Application interface for the link to the processor farm. (b) Application interface to the control network. (c) Diagnostics (d) Error Reporting (e) Run time Embedded code (e) Profiling, latency studies and optimization.

Event Data Collector

The Event Data Collector will have two sources of data input: (a) It will receive and buffer event fragments from the FE at a peak rate of 100,000 per second, (b) It will receive request messages from the Control Network. The messages received are 64 bits wide (source and destination 16 bit each, 8 command bits and 24 data bits). The messages will include requests for data, control and diagnostics messages . The EDC will have three output destinations: (a) Messages will be sent to the FE after stripping the source and destination. FE messages will be 16 bits wide, with 8 data bits. (b) Event

data fragments to the Event builder in response to L2 and Farm Processing data requests. (c) Error messages to the System Monitor when FE, Event Counter or Data Errors are encountered. The EDC will be an intelligent board with two or more control processors. One processor (IO_CPU) will drive the input and output DMA controllers. The buffer memory is logically organized in pages of size equal to the amount of data that can be pushed into the event builder input port (tentatively 256 bytes). Figure 23 illustrates the EDC servicing the EVENT_DATA requests.

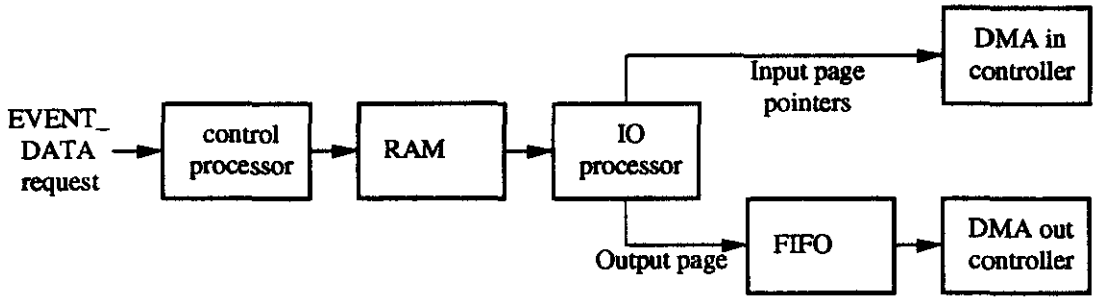

Figure 23. EDC servicing EVENT_DATA requests.

For each event the IO_CPU supplies one or more buffer memory page pointers. The input DMA controller uses the page pointers to transfer data from all the FE data input FIFOs to buffer memory. On the first page of an event fragment, the input DMA will reserve the first three words (word count, event number and event type). The EDC has a second processor (CN_CPU) servicing the control network. The CN_CPU receives EVENT_DATA requests from the processor farm at a peak rate of 100 kHz. For each EDD (up to 512) the CN_CPU will keep a list of EVENT_DATA requests. A dual ported memory will be used to communicate requests from the CN_CPU to the IO_CPU. The IO_CPU will use a FIFO to communicate with the output DMA controller. The FIFO will be continuously filled with output page pointers, it is mandatory that on every cycle of the switch (Event Builder) one page pointer be supplied per output port (EDD). The FIFO should be able to interrupt the IO_CPU when it is empty, half-full and full. The embedded code for the EDC will run with no kernel support; the amount of work to be done per data request and control message is limited to 10 μs in full readout mode, and 100 μsec in selective readout mode. For diagnostic and calibration purposes the CPU board should allow loading a real time kernel that provides multitasking, networking, and inter process communication facilities. Figure 24 shows the two modes of operation: run time and diagnostic mode.

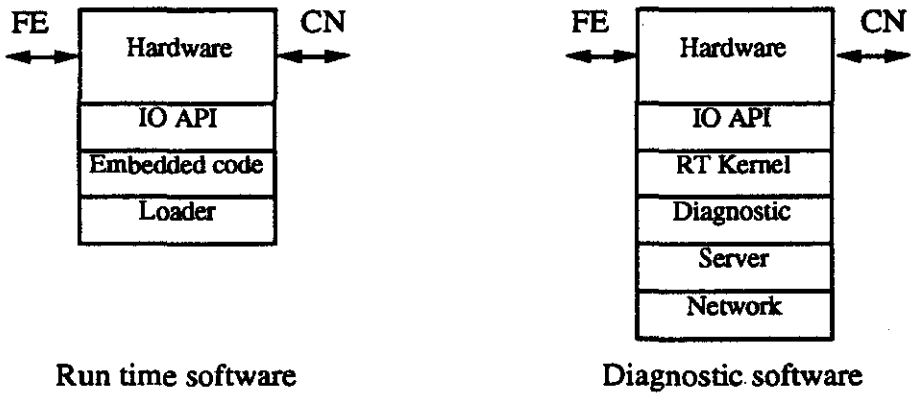

Figure 24. EDC Dual Mode of operation.

The EDC software development efforts will include: (a) Loader and configuration, (b) I/O libraries, (c) Diagnostics and Error Reporting, (d) Calibration Support, (e) Run time embedded code, (f) Profiling, latency studies and optimization.

Trigger Supervisor

The Trigger Supervisor will receive and respond to EVENT_REQUESTs from the Control Network, at a peak rate of 100 kHz. The amount of software development will be determined by the functionality excluded from the hardware/firmware design. Figure 25 displays the primary task of the Trigger Supervisor.

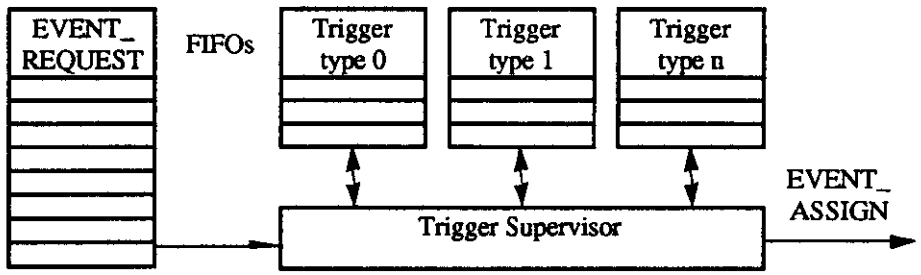

Figure 25. Trigger Supervisor Event Assignment.

All L1 Accepts are received by the Trigger Supervisor. Each trigger will have a trigger type associated with it. A FIFO for each different event trigger type is filled with the event number. The EVENT_REQUESTs from the Control Network are also buffered in a FIFO. For each EVENT_REQUEST serviced, the Trigger Supervisor consumes a word from one of the event trigger type FIFOs. The EDC software development efforts will include: (a) Loader and configuration, (b) I/O libraries, (c) Diagnostics and Error Reporting, (d) Simulation, (e) Run time embedded code, (f) Profiling, latency studies and optimization.

Tape Server Interface

The Tape Server will receive up to 1 MByte/sec from each of the processor groups. An error history from the System Monitor will also be shipped to the Tape Server. The error history will specify the range of events for which errors were received. Areas of software development will include: (a) Interface to System Monitor and Farm Processors, (b) Interface to media, (c) Distribution of data, (d) Data Formatting, (e) Data integrity and error detection, (f) Software tools for retrieval and data visualization.

Control and Monitoring

Overall coordination of the system is done by the System Monitor. All fast aspects of control are delegated to the Gating Logic, as it may involve the scheduling of dead time and the generation of **Crossing Sync** signals.

Control messages are used to monitor the status of the system. Checks must be performed at a rate that allows sufficiently fast detection to avoid serious data loss. Recovery from errors must be fast to avoid large dead times.

The protocol used to check the front-ends is optimized to avoid the number of messages going back to the System monitor. The EDCs will accumulate and forward status information to the System Monitor only at low frequency or when errors have oc-

curred. These messages will normally not add significantly to the traffic through the Control Router. Separate EDC_STATUS_REQUESTs may be issued to verify the correct functioning of the EDCs themselves.

System Errors

The following failure rates are only estimates (based on published values for typical standard commercial components, divided by the approximate number of components in the overall system). These rates are given as a preliminary basis for determining the error detection / error correction requirements of the system. The actual failure rates may be higher when the system environment, cabling and application-specific components are considered.

FE module (10,000 modules)	1 failure per day
FE Data Link (10,000 links)	1 failure per 4 days
FE Data Link soft error (10,000 links @ 60 Mbps)	1 error per 200 seconds
EDC/EDD module (640 modules)	1 failure per 6 days
EDC/EDD buffer soft error (~40 GBytes)	1 error per 4 hours
Event Builder module (96 modules)	1 failure per 150 days
Control Router (~20 modules)	1 failure per 50 days
Control Network soft error (~650 links @ 20 Mbps)	1 error per day

At the error rates predicted here, one approach might be to allow a limited reset mechanism for individual front-end module and data link errors, while requiring a full system reset in response to most other detected errors.

General Error Detection/Correction

The front-end data will be transmitted on serial links. The link BER typically improves by 10^{-2} for each dB increase in optical power, so a total BER of 10^{-14} should be attainable over short distances with sufficient transmitter power. At this rate, we expect a system bit error every 200 seconds. Less than 10% of these errors will occur in header words or start bits. Errors in data are not considered critical (to DAQ system operation) and do not affect the data readout. A data checksum may be included to detect and log these errors, but detection of a data error does not interrupt data flow.

For most types of serial encoding the probability of a single bit error is very low compared to double bit and burst errors, making error correction techniques such as SEC Hamming codes ineffective for serial data links. Block codes (Reed-Solomon, etc.) would be more effective but are relatively expensive.

At the expected error rate of < 1 error per 10 million events, error correction of any kind is probably not cost-effective. The event should either be discarded or the error treated as noise. For data error detection, an optional data checksum may be included. For header error detection, the only critical information is the location of the End-of-packet marker. Loss of synchronization is detected by checking the pattern word and the EVENT_NUMBER field in the header of the next event.

EDC Error Detection

The EVENT_NUMBER in the header of each event serves as an incrementing pattern word check by the EDC. The wrong EVENT_NUMBER may be caused by an Event Counter error, or it may indicate an error in the location of the End-of-packet marker of the previously received event. In either case the front-end module is automatically removed from the EDC's "readout list". Other channels connected to that EDC continue to be read out. An "empty data packet" with a "FE_error" status flag is inserted in place of the missing event fragment in EDC memory.

To resynchronize the front-end module, the EDC may write to the following front-end registers:

1) RESTART On receipt of this write message, the front-end module should ignore all Level 1 Accepts and clear its Event Data output buffer. After allowing sufficient time for the front-end output buffer to clear, the EDC should then clear its own corresponding input buffer. Clearing these two buffers will flush all event data from the selected front-end module data stream.
2) EVENT_NUMBER On receipt of this write message, front-end modules set the value of their Event Counters to the value contained in the message. Any module which is in RESTART mode should resume processing Level 1 Accepts. The EDC will then resume accepting data from the corresponding input channel, starting with the designated event number.

The EVENT_NUMBER register is written at startup and whenever the System Monitor wishes to resynchronize the front-end Event Counters (e.g., in response to an error message from an EDC). Reads and writes to this register are not synchronous, but L1 Accepts are inhibited during transmission of the command to avoid changing the value of the EVENT_NUMBER.

The EDC checks that EVENT_NUMBERs in packet headers are sequential and correctly aligned (as determined by the End-of-packet marker of the previous event). If a front-end module receives the wrong number of trigger pulses, the EVENT_NUMBERs will still be sequential, but there may be an extra (or missing) event in the front-end module output stream. This error is not visible in the event data stream and can only be detected by periodically reading the EVENT_NUMBER register in all front-end modules. Each front-end module returns the current value of its Event Counter in response to a read of the EVENT_NUMBER register. The EDC compares these values and may send a message to the System Monitor indicating that data from a specific front-end module is questionable over some range of Level 1 event numbers. This information can be added to the overall run record.

Bandwidth requirements

Data bandwidth requirements

Data is transported in units of complete event fragments. When front-ends or EDCs are read out, they always supply all data they have for the relevant event. Table 1 gives the rates at which event fragments move on the different connections in the system. We have assumed a level 1 trigger rate of 100 kHz, an average event size of 400 kBytes, Selective Readout Mode and a Level 2 Trigger that get a rejection of 90% using 10% of the data in the EDCs. The Level 2 accept rate is 10 kHz. Summation of the rates for each type of link gives the average net data flows shown in table 2:

Table 1. Event fragment rates for the different readout functions, the associated bandwidth and the number of links in the system.

Function	Link	Rate (Hz)	Pkt Size	# Links
Front-end readout	FE -> EDC	100,000	40	10,000
Level 2 data	EDC -> EVB	10,000	781	512
	EVB -> EDD	40,000	781	128
	EDD -> CPUs	40,000	781	128
Level 3 data	EDC -> EVB	10,000	781	512
	EVB -> EDD	40,000	781	128
	EDD -> CPUs	40,000	781	128

Table 2. Net data rates on the data links and total bandwidth for each type of link.

Link Type	Event/s/Link (Hz)	MB/s	# Links	Total MB/s
FE -> EDC	100,000	4	10,000	40,000
EDC -> EVB	20,000	16	512	8,000
EVB -> EDD	80,000	63	128	8,000
EDD -> CPUs	80,000	63	128	8,000

The link between a front-end and its EDC is used after each Level 1 Accept. With an average event size of 400 kB, 10k front-ends and 100 kHz rate, the average data bandwidth is 4 MBytes/s. Header words add 2 bytes per event per link (5%). Start bits and control bits add 19% overhead for a total of 40 Mbits/s. Since the maximum throughput of these links is only 60 Mbits/s, a reduced event size and good load balancing are very important.

In Full Readout Mode the required bandwidth through the event builder switch is 100 kHz * 400 kB = 40 GB/s. As each switch output channel runs at 1 Gbits/s, this option requires the maximal number of 512 EDDs to be present, giving a total maximum throughput of about 50 GB/s. For Selective Readout Mode the throughput depends on the second level algorithms that are used. If level 2 can get a 90% rejection using 10% of the data, it will need 100 kHz * 0.1 * 400 kB = 4 GB/s. Complete events would be read at 10 kHz, requiring another 10 kHz * 400 kB = 4 GB/s. The total data bandwidth requirement adds up to 8 GB/s, while 128 EDDs can absorb a maximum of 12 GB/s. Event fragment headers will add less than 2%. Overhead in a barrel switch is expected to be minimal.

Control bandwidth requirements

Control traffic is dominated by messages related to data readout on behalf of level 2 and level 3 and by synchronization messages to the front-ends. All messages between EDDs, EDCs, Trigger Supervisor and Gating Logic are passed through the Control Router (CR), which may present a bottleneck.

Estimates are presented in table 3 for the Selective Readout Mode where control is most intensive. We assume that the level 2 algorithms on average need to read 10 groups of EDCs, covering 10% of the data. For 10% of the events, Level 3 will issue one additional message to all EDCs to read the data.

Table 3. Traffic associated with different kinds of messages

Message	Link	Rate (Hz)	# Links	Total (kHz)
EVENT_REQUEST	EDD -> CR	781	128	100
(1 per event)	CR -> TrigSvr	100,000	1	100
EVENT_ASSIGN	TrigSvr -> CR	100,000	1	100
(1 per event)	CR -> EDD	781	128	100
Level 2 DATA_REQUEST	EDD -> CR	7,813	128	1,000
(10 per event)	CR -> EDC	40,000	512	20,480
Level 3 DATA_REQUEST	EDD -> CR	78	128	10
(0.1 per event)	CR -> EDC	10,000	512	5,120
EVENT_DONE	EDD -> CR	781	128	100
(1 per event)	CR -> TrigSvr	100,000	1	100
(1 per event)	CR -> EDC	100,000	512	51,200
READ_EVENT_NUMBER	Gating -> CR	10	1	0
	CR -> EDC	10	512	5
	EDC -> FE	10	10,000	100
FE_Event_Number_Return	FE -> EDC	10	10,000	100
FE_STATUS_REQUEST	SysMon -> CR	10	1	0
	CR -> EDC	10	512	5
	EDC -> FE	10	10,000	100
FE_Status_Return	FE -> EDC	10	10,000	100
EDC_STATUS_REQUEST	SysMon -> CR	10	1	0
	CR -> EDC	10	512	5
EDC_Status_Return	EDC -> CR	10	512	5
	CR -> SysMon	5,120	1	5
EDD_STATUS_REQUEST	SysMon -> CR	10	1	0
	CR -> EDD	10	128	1
EDD_Status_Return	EDD -> CR	10	128	1
	CR -> SysMon	1,280	1	1

Table 4. Message rate on the different links.

Connection	Time/Message	Messages	# Links	Total (kHz)
EDC -> CR	100 ms	10	512	5
EDD -> CR	106 us	9,463	128	1,211
TrigSvr -> CR	10 us	100,000	1	100
Gating -> CR	100 ms	10	1	0
SysMon-> CR	33 ms	30	1	0
Total into CR (kHz)	760 ns			1,316
CR -> EDC	7 us	150,030	512	76,815
CR -> EDD	1 ms	791	128	101
CR -> TrigSvr	5 us	200,000	1	200
CR -> Gating		0	1	0
CR -> SysMon	156 us	6,400	1	6
Total from CR (kHz)	13 ns			77,123
EDC -> FE	50 ms	20	10,000	200
FE -> EDC (on data link)	50 ms	20	10,000	200

Summation of the message rate by link gives the estimated total rates on the various types of links (Table 3). Traffic related to error reporting and correction and messages related to throttling is not included.

In total, messages are coming into the Control Router at about 1300 kHz. Most are related to Level 2 Data Requests and Event Done messages. Per message, about 760 ns is available on the backplane of the Control Router. Messages from the Gating logic related to synchronization will have to be routed with high priority. They occur at relatively low frequency. The link into the Trigger Supervisor has a very high rate. This may require the use of a special purpose processor for message processing.

In Full Readout mode The EDDs only issue one readout request per event. It is broadcast to all EDCs. In this mode, the traffic into the Control Router crate is reduced by about 70%. Data rates through the switch, however, go up by a factor of 5. Additional EDDs are needed to absorb the data traffic.