

# Using SCI to Implement the Local-Global Architecture for the ATLAS Level 2 Trigger

R.E Hughes-Jones, S.D. Kolya, D. Mercer  
The University of Manchester, Manchester, M13 9PL, UK

D. Botterill, R.P. Middleton, F.J. Wickens  
Rutherford Appleton Laboratory, Chilton, Oxon, OX11 0QX, UK

A. Bogaerts, E. Denes<sup>1</sup>, F. Giacomini, R. Hauser, P. Werner  
CERN, CH-1211 Geneva 23, Switzerland

M. Liebhart  
TU-Graz, Institute for Technical Informatics, 8010 Graz, Austria

J. Dawson, J. Schlereth  
Argonne National Lab, Illinois, USA,

A. Guglielmi  
DEC Joint Project Office, CERN, 1211 Geneva 23, Switzerland

## 1 Introduction

This work forms part of an extensive and on-going programme within ATLAS to explore potential architectures for the Level 2 trigger and the suitability of different technologies.

The system studied consisted of a farm of standard commercial processors interconnected by a high performance (200 MBytes/s) Scalable Coherent Interface, SCI, [1] network. Data, according to the predicted ATLAS traffic patterns, was distributed to the processors from SCI based data sources or buffers and results of trigger decisions were collected using the same network. The Scalable Coherent Interface offers networking with very low latency and high bandwidth, and is thus a good candidate for use in the Level 2 trigger.

The aims of the studies reported in this paper are:

- To implement and demonstrate the operation of a vertical section of the Level 2 trigger Local-Global architecture [2] based on use of the SCI interconnect. This architecture and the division of the required computing are discussed in section 2.
- To show the general suitability of SCI for the ATLAS Level 2 trigger and DAQ.
- To demonstrate that the message protocol [3] is sufficient and stable.
- To measure the performance of the components and the system implementation. This will be used as input to the modelling to allow extrapolation to a full system.
- To include error detection and rudimentary recovery.
- To investigate the interface between the high performance trigger functions and the essential DAQ functions such as initialisation, control, and monitoring.
- To implement the system simply and efficiently, but with flexibility and portability.

---

<sup>1</sup> Now at KFKI Research Inst. For Particle and Nuclear Physics, H-1525 Budapest, Hungary

Further work is in progress and proposed for the “ATLAS Pilot project” to study the more complicated configurations believed necessary for final ATLAS. Research into the performance of the new 500 MByte/s interfaces and large SCI switches as well as how best to optimise the use of SCI and commodity processors within the Level 2 architectures being studied is also under way.

Section 2 discusses the Local-Global Architecture and protocols. Section 3 describes the hardware and software implementation of the vertical slice test system and gives a brief introduction to the way SCI has been used. We then present the topologies studied and describe the measurements made on the system. Section 6 gives results obtained from the tests together with an analysis of the data. This is followed by a discussion and the conclusions on applying this architecture and technology to the ALTAS trigger.

## 2 The Local-Global Architecture

### 2.1 Requirements for Level 2

Given the high bunch crossing rate (40 MHz) and the large number of channels in the ATLAS detector [4], the raw data rate from the detector will be up to  $10^{15}$  Bytes/s. This data rate must be reduced to a more manageable level for long term storage by a three-level trigger system, shown in Figure 2.1.

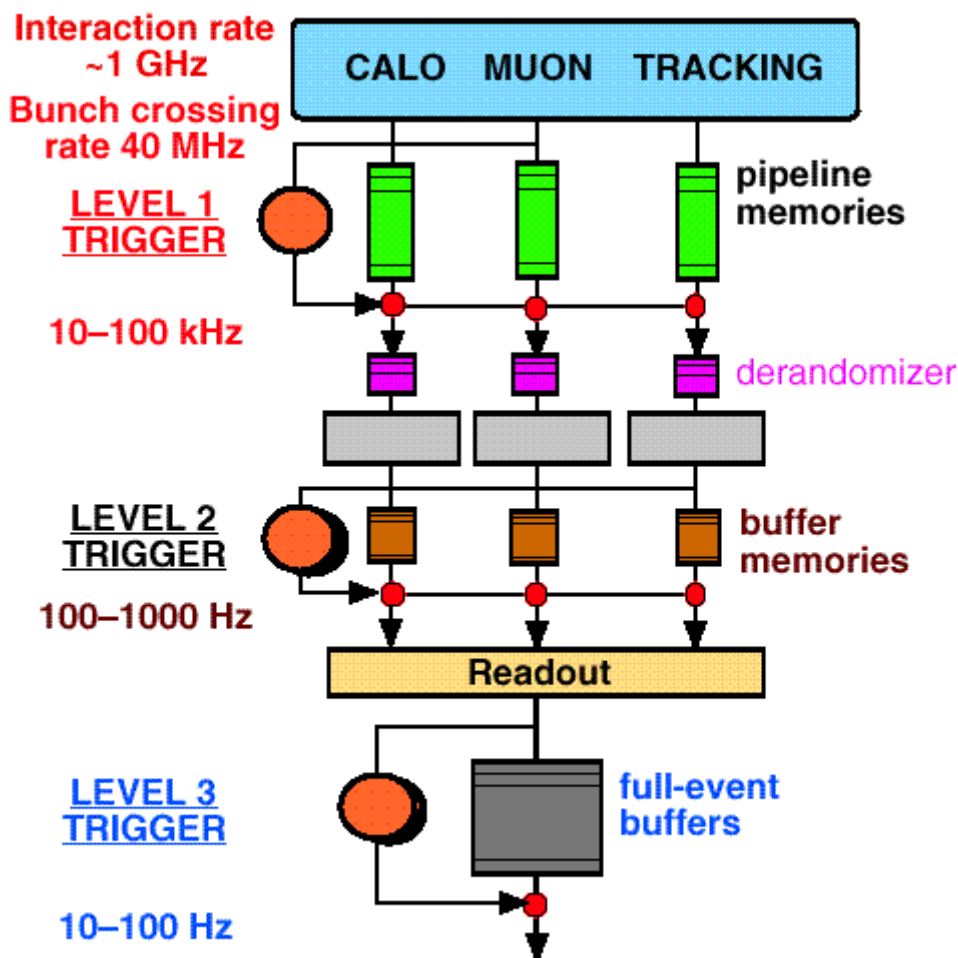


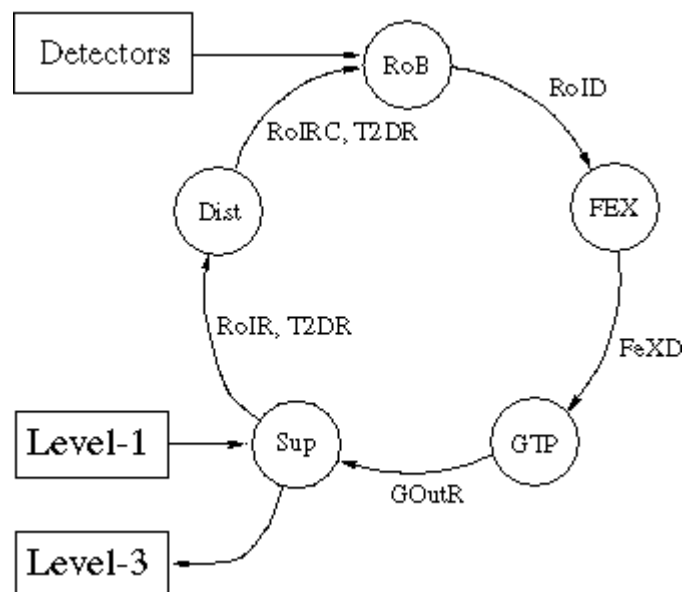
Figure 2.1 Overview of the ATLAS Trigger

The Level 1 trigger will give a rate reduction of approximately 1000 and events which pass this initial selection will be transmitted from the front-end electronics for each part of the detector to corresponding Read-out-Buffers (RoB). At this point the data for each event (typically a few MBytes)

will be distributed across many of the 1500 or so Read-out-Buffers. The Level 1 trigger will also pass to the Level 2 system Region-of-Interest (RoI) pointers indicating where the most significant activity has occurred within the detector. These pointers are used to limit the data which needs to be accessed by Level 2 to only a few percent of the total data for the event. Since the Level 2 system design has to be able to handle events at up to 100 kHz, the data path into the Level 2 system must have a capacity of a few GBytes/s. The Level 2 trigger is required to reduce the total event data rate by a further factor of 100, with an average latency of 1-10 ms per event. The Level 2 system looks at each layer of the detector for each region of interest and determines features (e.g. a particle energy, or track parameters) and then combines the features from each RoI for selection on overall event criteria - e.g. combinations of particles with specific parameters. The decision to keep or discard the event is then passed back to the Read-out-Buffers which either send the event to the next level or simply discard the event. Current estimates are that the Level 2 system will require between 500 and 1000 processors [5].

## 2.2 The Local-Global Architecture

Figure 2.2 shows the functional diagram for the local-global architecture for Level 2 that satisfies the requirements discussed above. This is taken from the corresponding paper describing the work done on implementing the local-global architecture with DS Links [6].



**Figure 2.2 The Functional Model of the Local-Global Architecture**

The basic event sequence and flow of messages are as follows:

- The Level 1 emulator sends details of the Regions of Interest (i.e. the position within the detector and hence which RoBs contain the data needed to make the Level 2 decision) to the Supervisor.
- These details are received by the Supervisor processor in the supervisor complex, and reformatted into RoI requests, RoIR, to be sent to the RoBs. The supervisor tags the requests with a Global Processor identifier indication which global processor will be used. If no processor is available, the supervisor will wait until one becomes free.
- The Supervisor sends the RoIR request to the RoI Distributors, which transmit it as a RoIRC message to the required RoB(s). The RoI Distributors are required to help reduce the fanout load on the Supervisor.
- The RoB(s) send pre-loaded 'event' data, RoID, of a specified length to the FeX(s).
- The FeX(s) builds the trigger event from the RoB(s) and send a short 'feature' packet, FeXD, (64 bytes long) to the assigned Global processor.
- The Global processor combines all of the 'features' of the event and generates an 'event decision', GoutR, which it sends to the Supervisor.
- The Supervisor notes that the Global processor is free and uses the identifier for a new event. It also groups event decisions and when a sufficient number have been assembled, the Supervisor sends the grouped decisions, T2DR, to the RoI Distributor for onward transmission to the RoBs.

- The RoBs then release the event buffer (in the final system they would transmit accepted events to Level 3, the Event Filter).

The key points of the architecture are:

- Requested push, the buffers send data on a request from the supervisor.
- The data messages that are passed round the system contain their own destination address or routing information.
- All the data from one RoI for one sub-detector is sent to 1 FeX.
- Many FeXs are working in parallel on one event, and send their results to one Global processor.
- Many events will be in the system at one time.
- There is a minimal number of control or data -request messages.

### **3 Implementation of the Vertical Slice**

The vertical slice was implemented in two configurations. Figure 3.1 shows all the RoB, FeX, Global, and supervisor nodes connected with a single SCI ringlet. Packets from a message between any 2 nodes pass through all the SCI chipsets on the ringlet. A second configuration, shown in Figure 3.2, used a SCI switch to link 4 separate ringlets. This provided partitioning of the RoB - FeX traffic and gave separation from the Global - Supervisor traffic. It also allowed study of the effect of the transit time of the SCI switch.

The hexadecimal numbers on the figures give the SCI node ID of each device on the ringlets. These were chosen to allow the simple introduction of the SCI switch. The 'S' next to a node denotes a scrubber node, responsible for providing the SCI clock and maintaining the low level packet protocols.

The system was operated with 10 processors and 4 switch ports giving a 14 node SCI network, which is the largest SCI network yet operated at CERN.

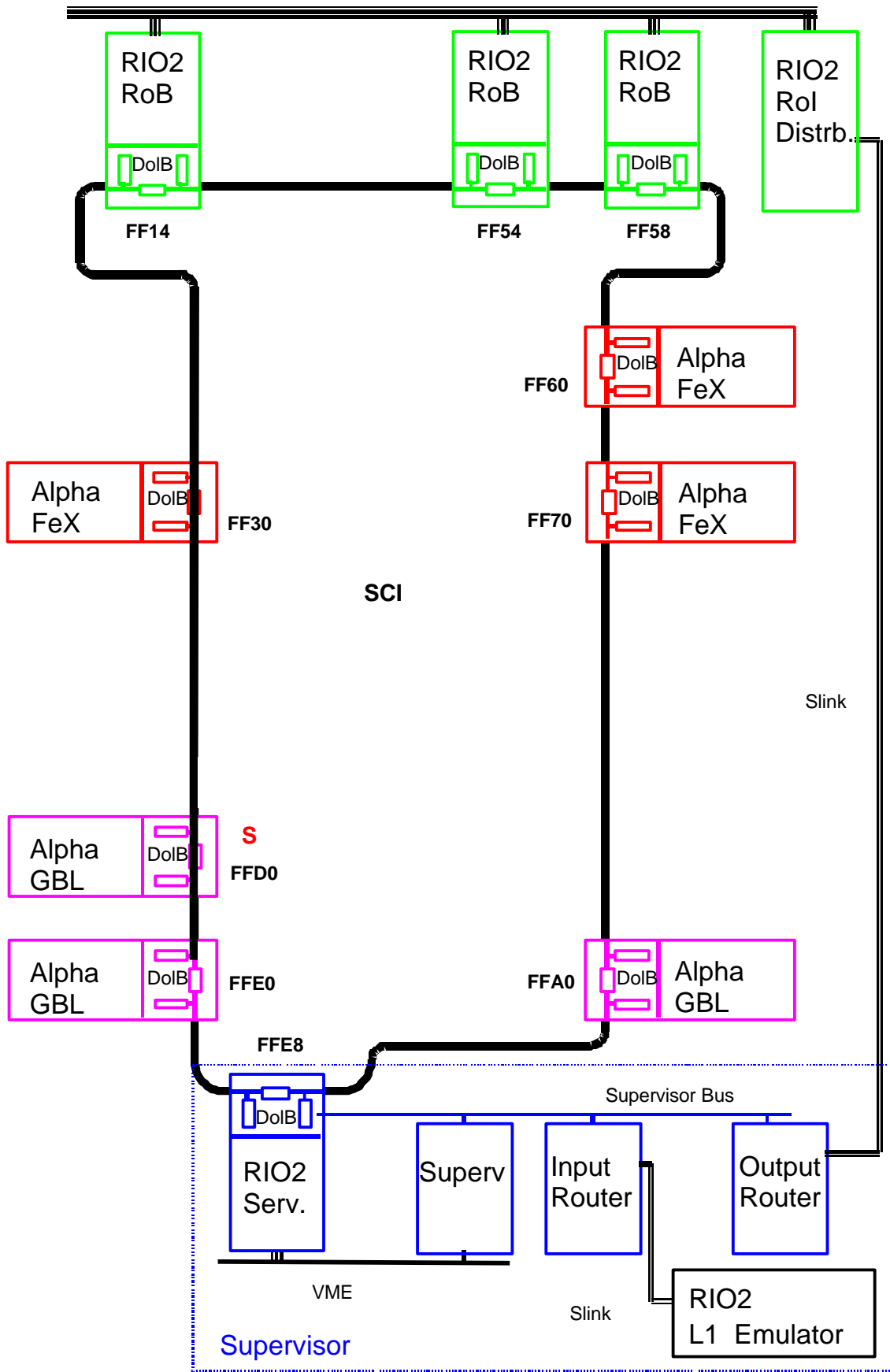


Figure 3.1. Diagram of the SCI Vertical Slice using 1 SCI ringlet.

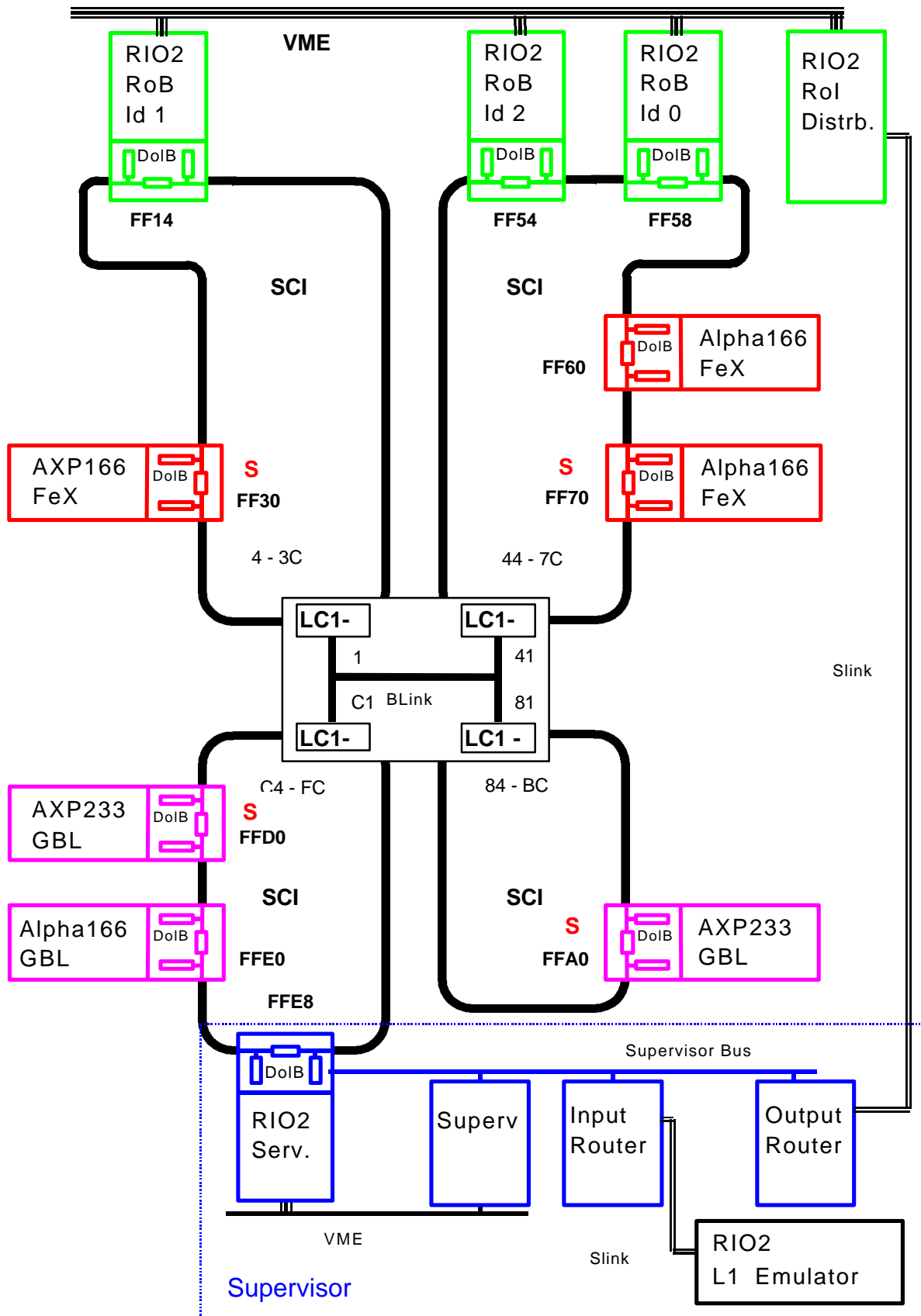
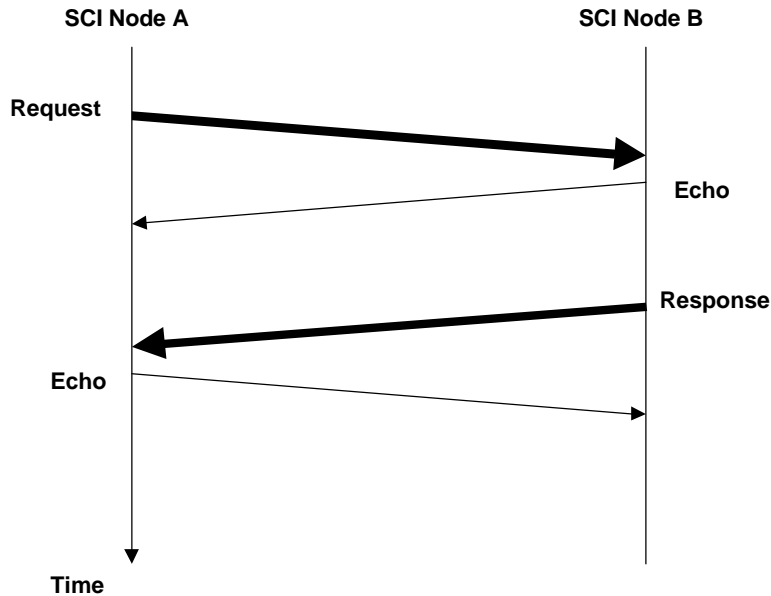


Figure 3.2. Diagram of the SCI Vertical Slice with 4 SCI ringlets linked with a SCI switch.

### 3.1 Overview of SCI and its use in these Studies

The Scalable Coherent Interface is an IEEE standard for interconnecting multi processor systems. An SCI network can be seen as a logical bus but it is actually constructed from unidirectional point-to-point links between devices (typically processors) forming a ring structure. SCI uses a split request - response transaction protocol as shown in Figure 3.3.



**Figure 3.3. SCI split transaction showing the initial request and the response together with the associated echo packets. The echo packets also indicate if the receiving node is busy and this initiates a re-try of the packet.**

The communication is based on the exchange of small packets. An SCI packet has a 16-byte header, a 16 - or 64-byte payload and a 2-byte CRC (the 256-byte payload allowed by the standard has not yet been implemented). The simplest configuration of an SCI network is a ring, but more complex topologies are possible by interconnecting rings using switches.

Custom interfaces and SCI probes have been developed for the demonstrator project. An example of a custom interface is the SCI-PCI card developed in Manchester. This card includes flexible PCI logic, support for 32 or 64-bit PCI (although none of our current processor boards supports the latter option) and provision for extra processing to be added on a daughter card connected via an internal data bus. Although this card uses the older and slower SCI NodeChip [7], its flexibility has enabled detailed understanding of PCI as well as SCI interfaces and components. It will continue to be useful to study options for 'hardware assist' modules, e.g. to aid fragment building.

The vertical slice tests presented in this paper have been obtained using commercial PCI-SCI interfaces [8] from Dolphin Interconnect Solutions based on their link controller LC-1 [9] running at a link speed of 200 MBytes/s. This interface gives high performance by supporting several outstanding requests, so that a new transaction can start even though up to seven previous requests are still waiting for response packets. Even though at the lowest level the communication is based on the exchange of packets, the Dolphin hardware offers the following features for transmitting data:

- Transparent mode: the CPU writes directly into (virtual) memory. The memory management hardware maps this address onto the PCI card, which in turn sends SCI packets destined to the required memory location on the remote SCI node. This is remote shared memory, and the only software intervention required is a "barrier" like operation to flush outstanding buffers and to check for transmission errors.
- DMA mode: The CPU loads the DMA engine on the interface with the location and length of the data in memory and the required SCI destination address. The interface then fetches the data from

memory, creates and transmits multiple SCI packets as required. Status registers provide information on the progress of the transmission and any errors.

- Packet mode: Raw SCI packets are constructed by the CPU. The Dolphin interface just fetches the packet and transmits it on SCI.

The interface provides the following facilities for dealing with incoming packets:

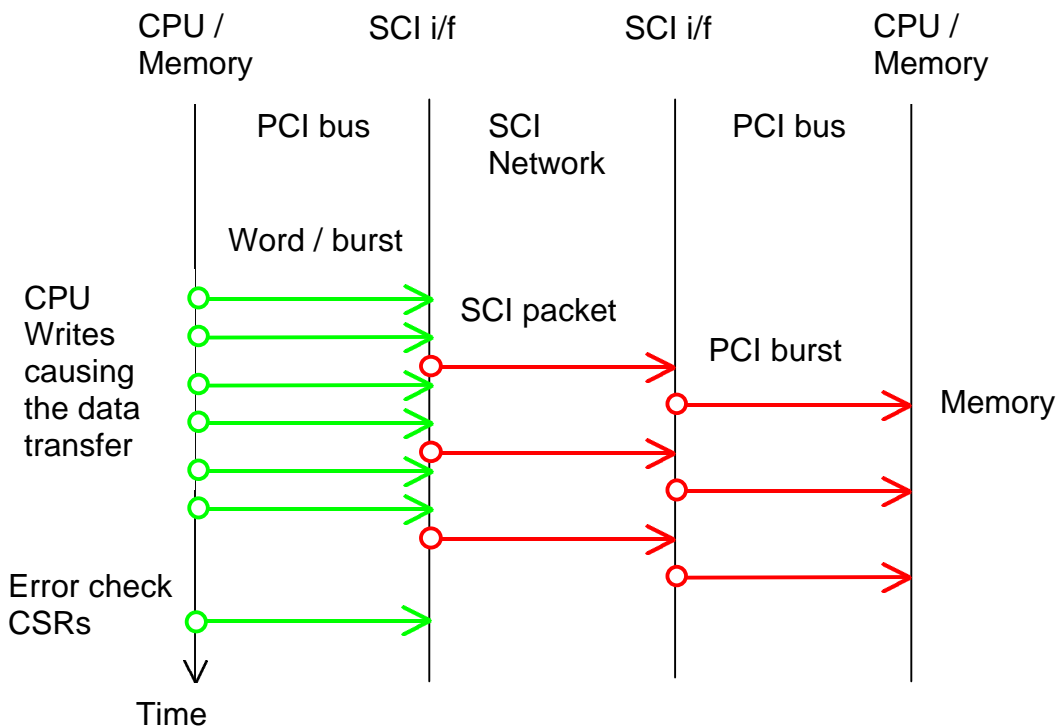
- Transparent mode: The interface places the incoming user data directly in the memory address specified in the incoming SCI packet.
- Ring buffer mode: The interface places the raw SCI packet into a user specified ring buffer in the host memory. The application software has to extract the data from the ring buffer, this process is similar to using a time ordered mailbox.

Two ways of moving data over SCI have been evaluated in these tests:

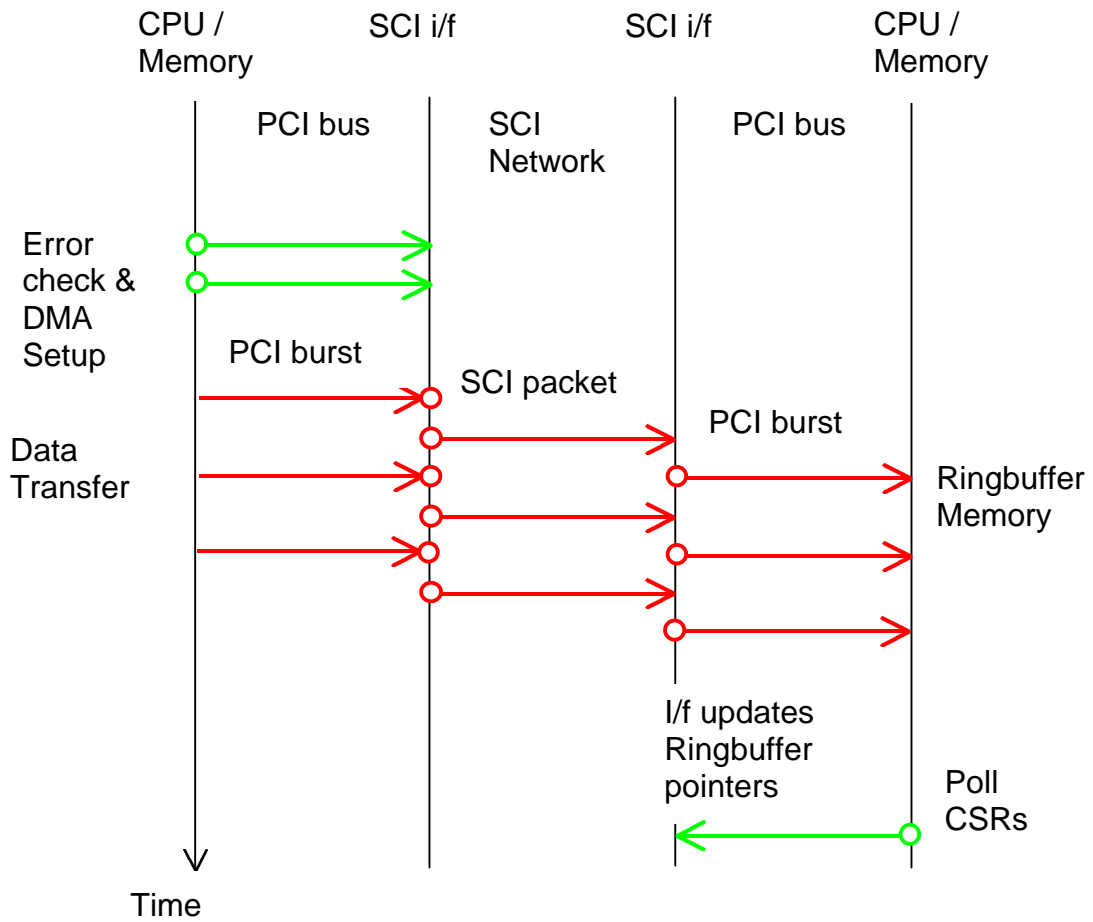
- Transparent mode - Transparent mode. This was used with a message passing library implemented over the remote shared memory described above. The sending CPU wrote every word of the data message into the required remote memory, followed by a control message to an agreed location. The receiving CPU only had to poll these control areas to determine that a message had arrived, separate control areas were used for each remote CPU. On receipt of a message, the CPU sent back an acknowledgement containing the address where the sender could write the next message.
- DMA mode - Ring buffer mode. Here the sending CPU had only to setup the DMA engine to send the data. The receiving CPU polled a CSR register to determine if any message had arrived, but then had to deal with each packet of the message.

The interaction between the CPU, PCI and SCI for these two methods is shown in Figure 3.4; both ways used the DMA engine on the interface to generate bursts of data on the PCI bus. In Transparent mode, performance is critical on the ability of the CPU to send data to the PCI interface in PCI bursts not a single word per PCI transaction. For DMA mode, the time to check for errors and setup the DMA was  $\sim 8\mu\text{s}$ , but the transfer rate of the SCI packets was the same for both modes. Thus the DMA mode took longer in real time to move a block of data, but the CPU time required for long transfers was much less ( 1 Kbyte of data using about half the CPU time).

It is possible to mix transmission and reception modes, in fact, other tests have used DMA mode to transmit the data with Transparent mode for reception.



**Figure 3.4a. The interaction between the CPU, PCI and SCI for Transparent mode – Transparent mode transfers.**



**Figure 3.4b. DMA mode – Ring buffer mode. The PCI bursts to move data from the sender's memory are generated by the DMA engine on the PCI interface.**

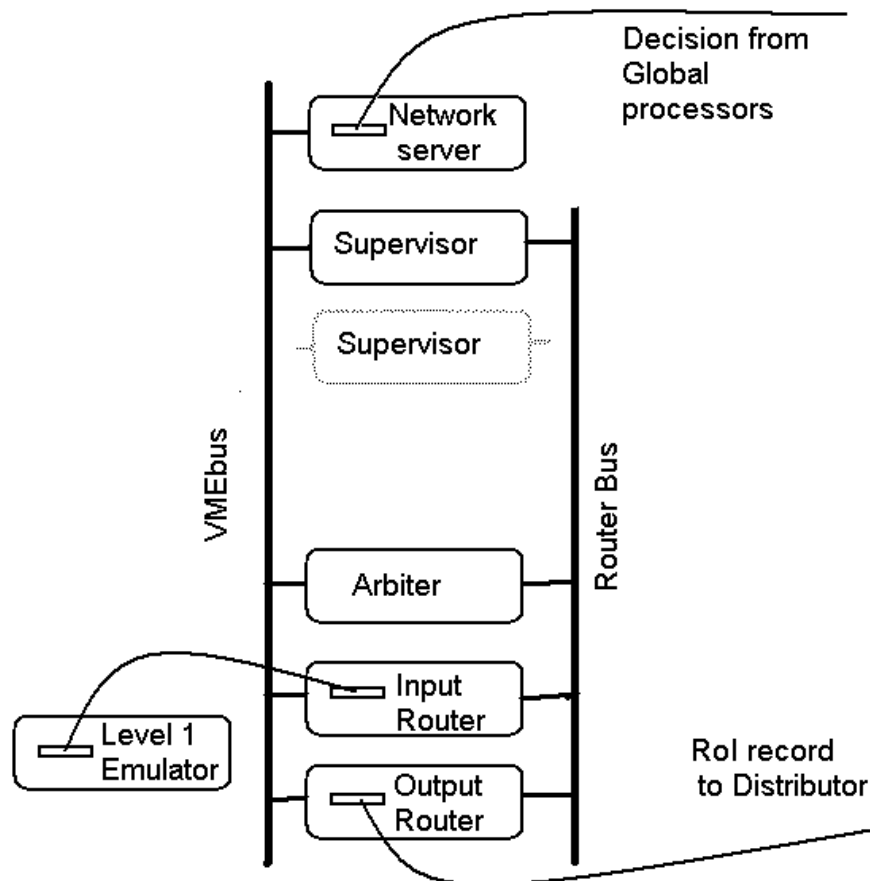
## 3.2 Supervisor

The Supervisor program provided an interface from an emulated source of Level 1 data to the vertical slice of the second level trigger processor farm.

### 3.2.1 Hardware

The Supervisor used custom input and output routers designed at ANL/MSU [10],[11] and implemented in VME. These routers were linked with a special 2-channel high speed front-panel bus and used custom PMC modules to connect to the CES RIO2 Power PC based processors [12]. The Supervisor is shown schematically in Figure 3.5.

**Figure 3.5. Diagram of the Supervisor**



The Input Router accepted emulated Level 1 data, which consisted of Region of Interest (RoI) fragments, via the S-link [13] input interface and distributed the data to the supervisor processor using the Router bus. The processor then constructed the RoI record and sent it to the custom S-Link Output Router via the Router bus. S-link was used to send the RoI records to a single RoI distributor that directed each record to selected read out buffer (RoB) emulators.

The Network Server provided the interface to the SCI network using the Dolphin PCI SCI interface coupled to the PMC slot on the CES processor with an Technobox adapter card[14].

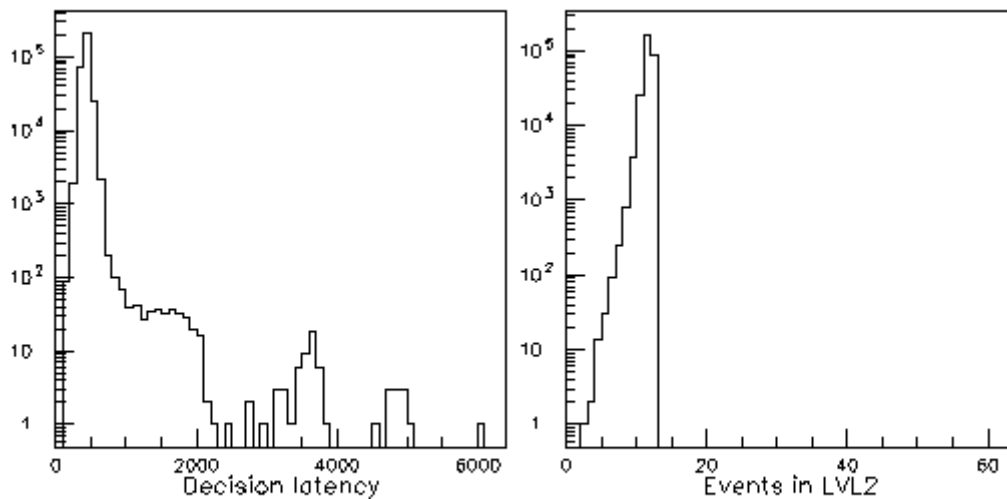
### 3.2.2 Software

The tasks performed by the supervisor processor were the following:

1. Read the RoI fragments either from the input router via the PMC interface or from a file. In either case, an RoI fragment for each trigger type was read until a complete event was built. This was done in order to make the demonstration as realistic as possible. When reading data sent by the

- Level 1 emulator, fixed length blocks were read with a length approximately equal to one complete event.
2. Build an RoIR record for the RoI distributor. Part of this process was to choose a free global processor tag from a prepared list. Two parameters determined how many events could be allowed to queue up in the Level 2 system: the number of global processors and the number of events allowed to be outstanding for each global processor. Together these numbers were used to form a unique global processor tag.
  3. Send the RoIR out the PMC interface to the output router.
  4. Poll for the presence of a global decision and process the decision if present. Processing the global decision consisted of verifying the event ID and global processor tag, reading an interval timer and calculating the latency, updating the run statistics and updating the Level 2 decision record block.
  5. Check for enough decisions to send a group of decision records back out to the RoBs. If the threshold had been crossed, the block of decision records were sent the same way an RoIR was sent.

The Network Server received decisions from the global processors and placed the event ID and decision into shared memory. The supervisor processor, acting as VME master, polled a semaphore and copied the decision information as appropriate.



**Figure 3.6. The decision latency and the number of events in the Level 2 system.**

The measurements made by the supervisor were the Level 2 decision latency, for which a running average as well as a histogram was kept, the throughput and the Level 2 pipeline occupancy, measured with a histogram. The measurements made are discussed in Section 5. Typical distributions of the latency and Level 2 pipeline occupancy, that is the number of events the supervisor had outstanding in the Level 2 system, are shown in Figure 3.6.

### 3.3 RoI Distributor

#### 3.3.1 Hardware

The RoI Distributor was implemented on a CES RIO2 8061 PowerPC board running LynxOS. The processor used a S-link PMC card to receive RoIR requests and T2DR decisions from the Supervisor output router, it then interpreted the message and transferred the requests to only those RoBs which were involved in the current RoI. This RoIRC transfer was made using the onboard PCI/VME bridge. Which RoBs to use was derived from the  $\phi$ -index using the lookup table shown in Table 4.1. In a full system, every VME crate would have its own RoI Distributor.

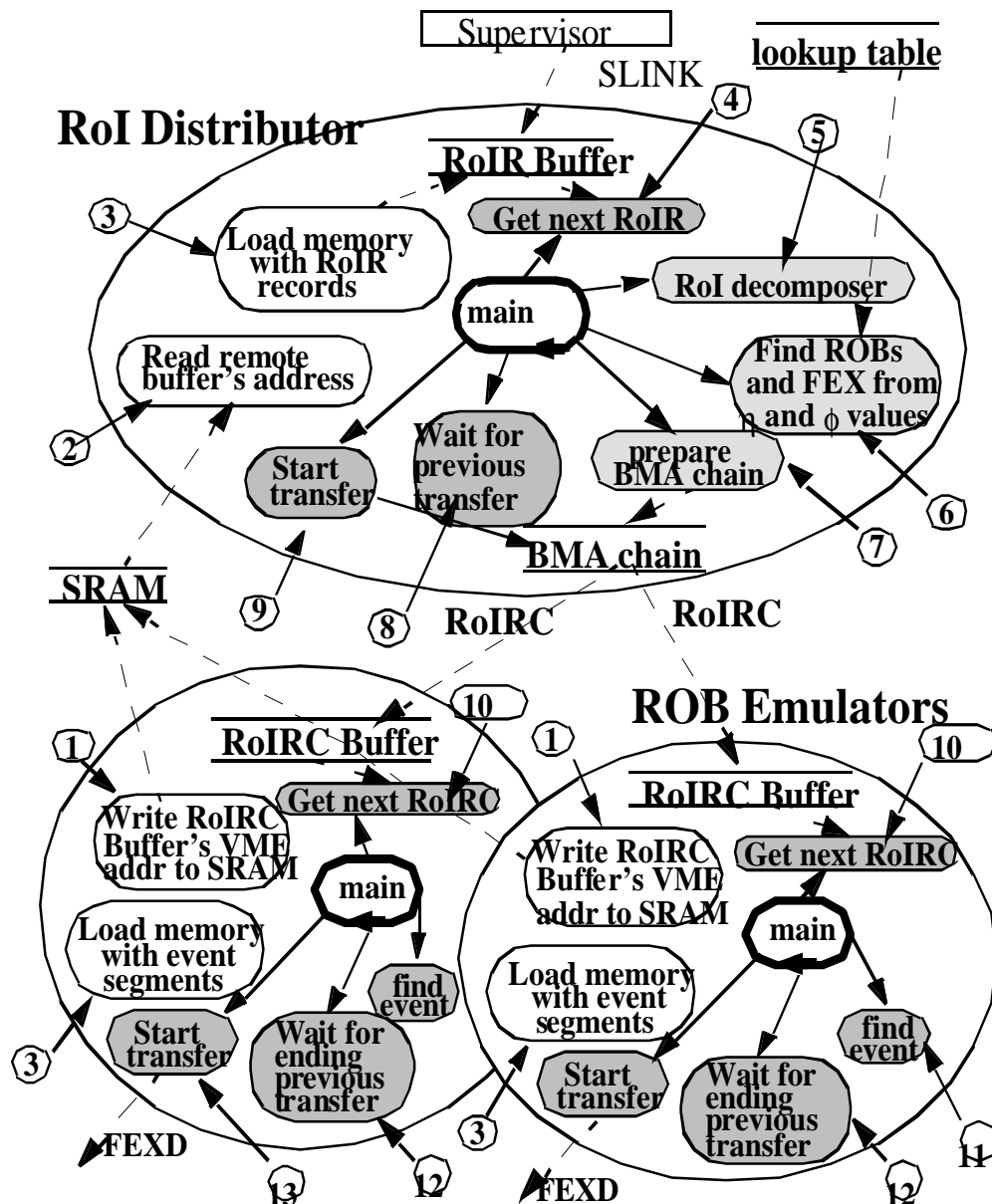


Figure 3.7. The working principle of the RoI Distributor and the RoB Emulator software.

### 3.3.2 Software

Figure 3.7 shows the functions performed by the RoI Distributor and RoB software [15]. A detailed description is given below with the numbered points referring to the corresponding tags on the diagram:

1. The RoB Emulators allocate memory area for message channels. This area will be used to receive RoIRC Request records sent by the RoI Distributor. The address of this channel is written to a memory location which can be read by the RoI Distributor.
2. The Local RoI Distributor finds the above channel addresses and initialises itself to send ROIRC records to these addresses.
3. The RoB Emulators have to be loaded with event fragments. The RoI Distributor reads the lookup tables which define the RoI - RoB correspondence for every RoI type. Then the system is ready to work.
4. The RoI Distributor waits for Supervisor messages. It polls for messages as this is much quicker than using interrupts. When a message arrives, the program checks whether it is a RoIR or a T2DR. In the latter case it counts the number of messages and discards them.
5. If an RoIR record arrived, the RoI Distributor decomposes it to several RoIs.
6. For each RoI the program
  - Checks the subdetector mask. If the given crate is not involved it discards this RoI.
  - If the crate is involved in the RoI, it uses the lookup tables to find which RoBs are involved. If at least one is concerned, it prepares an ROIRC (RoI Request in Crate) record. Adds its own data to the record:  
The FeX address (The  $\eta$  value of the RoI is using to identify the FeX by indexing the lookup table),  
The total number of RoBs and the RoB pattern.
7. Prepares a VME block moving list, specifying a chain of RoB addresses.
8. If VME chained transfers are used the program checks whether the previous transfer is finished.
9. The program starts the chained transfer and returns to the polling of input messages. If an event is to be sent to more than one RoB, all the RoBs must be ready before the chained DMA is started.

The performance of the RoI Distributor has been measured:

Time to receive a message on the S-link	17 $\mu$ s
Processing time and setup of DMA	5 $\mu$ s
Time to check the RoB	3 $\mu$ s
Time to start the DMA	5 $\mu$ s
Time for VME transfer for each RoB	9.5 $\mu$ s

In principal, the receipt of incoming messages from the supervisor and the output of messages to the RoBs over VME can be overlapped. But both use the PCI bus so effectively these operations are sequential. However, processing of the Supervisor messages can be concurrent with the output over SCI operating in DMA mode.

## 3.4 Readout Buffers

### 3.4.1 Hardware

Each RoB was implemented on a CES RIO2 VME board running LynxOS. Both 8061 PowerPCs running at 120 MHz and 8062 PowerPCs running at 200 MHz were used. RoIRC information was written into the RoB shared memory by the RoI Distributor via VME backplane and the RoB VME/PCI bridge. Each RoB was connected to the SCI network using the Dolphin PCI SCI interface coupled to the PMC slot on the CES processor with an Technobox adapter card[14].

### 3.4.2 Software

10. The RoB Emulators poll shared memory for ROIRC messages.
11. When a message arrives, the program tries to find the event in the event buffer and then prepares the RoID record:
  - Copies ROIRC header into the header part of event segment.
  - Adds its own data to the record:
    - record type, record length, RoB identifier, index of the RoB in RoI, number of data words
12. If DMA transfer mode is used for SCI, the program checks for errors and whether the last DMA has finished.
13. The Rob starts sending the RoID record to the FeX address given in the ROIRC record - see step 6 above. In the case of DMA transfers it sets up the SCI DMA and returns to polling of input. For shared memory transfers over SCI, the CPU writes all the data to the SCI interface and then returns to polling.

## 3.5 FeX and Global Processors

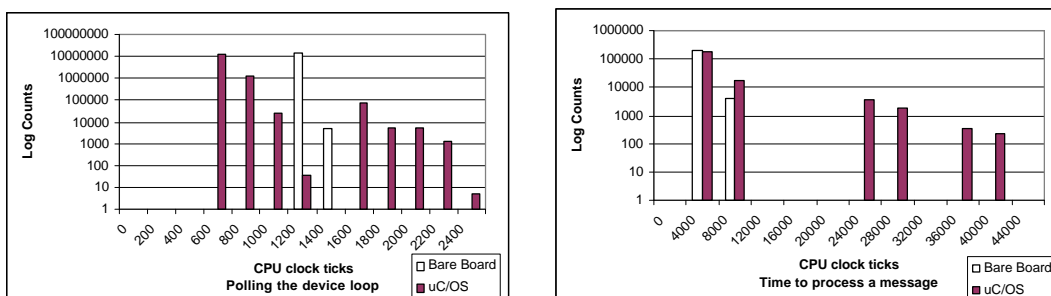
### 3.5.1 Hardware

AXPpci33 AT style single card computers [16] and Multia base units [17] were used for the FeX and Global Processors. Both motherboards had fast second level cache and used the DECchip 21066 alpha processor [18] which has an integrated PCIbus controller that can function as PCI master or slave. By using an on chip write buffer, the CPU is capable of generating 32 byte long data burst transfers on PCI. The Multia operated at 166 MHz and the AXPpci33s at 233 MHz or 166 MHz. The Dolphin PCI-SCI interface with the LC-1 SCI chip was used to connect the system to the network.

### 3.5.2 Software

The software in the FeX and the Global Processors performed almost identical tasks. The FeX received data from SCI and built trigger events from a number of fragments, RoIDs sent by the RoBs. It presented the event to the trigger algorithm function and then sent the result, FeXD to a Global Processor. The Global Processor built its event from the FeXD fragments, sent by the FeXs. It presented the event to the global trigger algorithm function and then sent the result, GOutR to the network server in the supervisor. Due to the different byte ordering in the PowerPC and alpha processors, the FeX and Global software had to manipulate the byte ordering. In these tests, no algorithm was run in the trigger functions.

The processors were operated in two ways: running with no operating system or “bare board”, and with a simple real time kernel  $\mu$ C/OS [19] which provided light weight thread switching. Figure 3.8 shows histograms of the times taken to poll the hardware and process a SCI message when running under the two environments. The data were taken using a 233 MHz AXPpci33, each clock tick corresponding to 4.29 ns.



**Figure 3.8. Comparison of timing for Polling and Message processing with bare board and  $\mu$ C/OS environments.**

The time taken to poll the hardware decreases as there is no need to poll the terminal with  $\mu\text{C}/\text{OS}$ , however there is a tail from  $\sim 7\mu\text{s}$ . The time to process a SCI message is similar in both cases, but  $\mu\text{C}/\text{OS}$  shows small peaks at  $\sim 86\mu\text{s}$  and  $137\mu\text{s}$ . This time and low rate is consistent with background handling the timer interrupt.

## 4 Topologies Studied

The topologies studied with the Local-Global architecture were operated in a closed loop under the control of the Supervisor. The Supervisor ran in “self triggering” mode and no Level 1 emulator was used. Instead, the Supervisor initiated a new event as fast as it could provided that there was a free Global processor and that the maximum number of events, MAXEVENT, allowed in the loop had not been exceeded. For example, if MAXEVENT was set to 3 the Supervisor would issue 3 events in to the system and then wait for the Level 2 processors to return a decision before issuing the fourth event.

By specifying the  $\eta$ ,  $\phi$ , and Global processor id for each event, the Supervisor could define the topology of the event. The value of the  $\phi$ -index determined which RoBs in a sub-detector were part of the RoI. The value of the  $\eta$ -index specified which FeX was to be used. The possible values of  $\eta$  and  $\phi$  are show in Table 4.1.

The following sections describe the different configurations of the Supervisor - RoI Distributor - RoB - FeX - Global - Supervisor chain that have been considered:

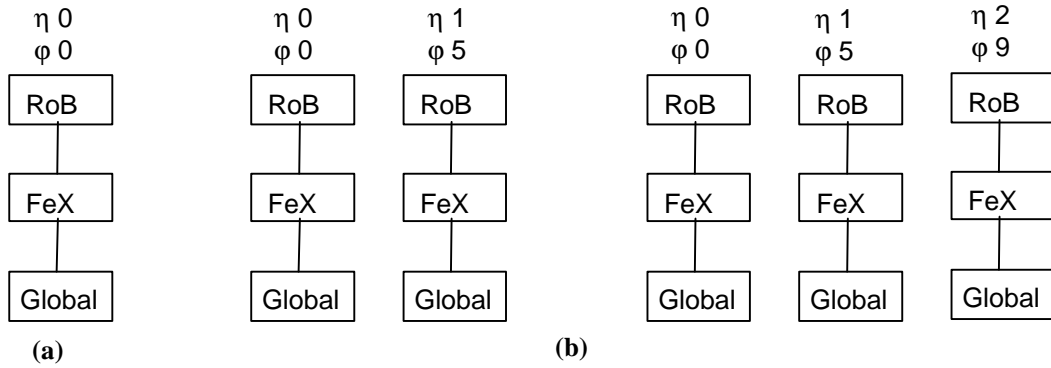
$\phi$	RoB id				
	0	1	2	3	4
0	•				
1	•	•			
2	•	•	•		
3	•	•	•	•	
4	•	•	•	•	•
5		•			
6		•	•		
7		•	•	•	
8		•	•	•	•
9			•		
10			•	•	
11			•	•	•
12				•	
13				•	•
14					•

$\eta$	FeX Tag
0	0
1	1
2	2
3	3

**Table 4.1. RoBs Selected with the  $\phi$ -index and the FeX allocation with  $\eta$**

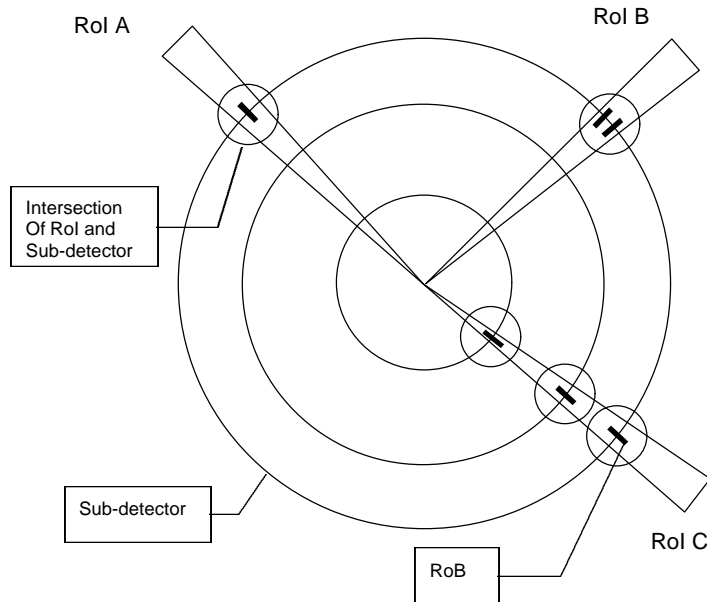
## 4.1 Single and Parallel Pipelines

A single stripe consisting of 1 RoB, 1 FeX and 1 Global processor is depicted in Figure 4.1a. In this case, each event has only one RoI and this RoI is contained in a single RoB. In Figure 4.2 RoI A gives a pictorial representation of this. Since the stripe has an intrinsic sequential structure of processing steps for each event, several events can be pipelined in the stripe up to the maximum number of events allowed in the system by the supervisor.



**Figure 4.1 (a) A single stripe 1 RoB 1 FeX 1 Global.**  
**(b) Parallel stripes with 2 or 3 RoB-FeX-Global**

By increasing the number of Global processor ids available to the Supervisor, and selecting the correct  $\eta$ ,  $\phi$  indices, 2 or 3 RoB-FeX-Global stripes were operated in parallel as shown in Figure 4.1b. Here again, each event has only one RoI and this RoI is contained in a single RoB; by increasing the maximum number of events allowed in the system by the supervisor, many events may be pipelined in each stripe.



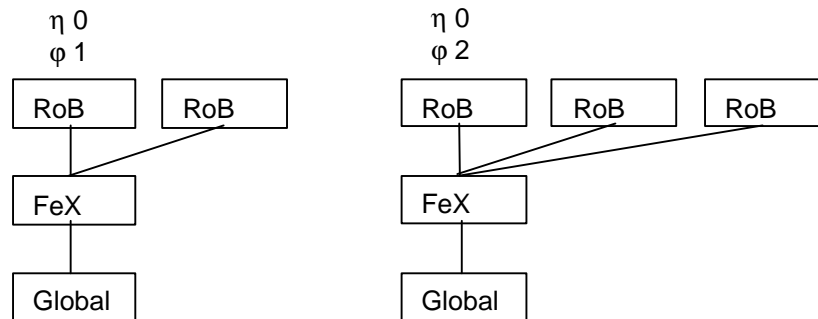
**Figure 4.2. The relation between RoIs, sub-detectors and the RoBs involved.**

## 4.2 Building RoIs from Fragments

The system could be set up with 2 or 3 RoBs feeding 1 FeX and 1 Global, as show in figure 4.3. This covers the case where there is only one RoI per event, but the intersection of the RoI with a sub-detector covers several RoBs, for example RoI B in figure 4.2. The FeX receives data form several RoBs in parallel and has to collect the fragments in order to build the RoI. Again multiple events could

be pipelined in the system. This configuration tests the efficiency of the RoI Distributor and the fragment builder inside the FeX.

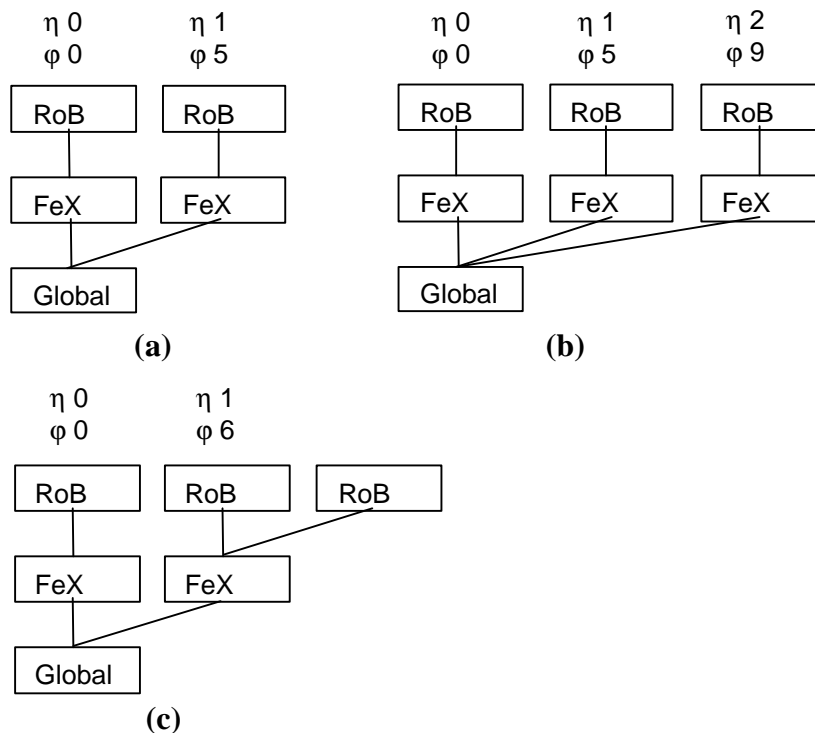
If the multiple data transfers from different RoBs to the FeX completely overlap without interference, the event rate should be similar to that of a single stripe, where there is only one RoB per RoI. If the RoB-FeX transfers are sequential, then the event rate could be half that of a single stripe.



**Figure 4.3. Fragment building in the FeX, multiple RoB per RoI.**

### 4.3 RoI Parallelism

The parallel processing of RoI information was investigated by allowing several FeXs (each possibly receiving data from multiple RoBs) to analyse events with multiple RoIs from several sub-detectors at the same time, as shown in Figure 4.4 (a). RoI C in Figure 4.2 shows the case where 3 RoB feed 3 FeX into 1 Global.



**Figure 4.4. RoI Parallelism, multiple RoB – FeX channels feeding 1 Global processor.**

Figure 4.4 (a) and (b) show events containing 2 or 3 RoIs each covering a single RoB. If the multiple RoB - FeX threads were able to overlap, the performance should be similar to that of a single stripe with the addition of fragment building in the Global Processor.

## 5 The Measurements Made on the System

For each configuration two measures of the system performance were made with the Supervisor running in “self triggering” mode. These were the average time between events completing the loop (i.e. 1/frequency) and the loop latency, defined as the time a RoI being generated and the final Level 2 decision arriving at the supervisor. The average time between events was measured by dividing the total number of events sent round the system by the total elapsed time. The supervisor generated a histogram of the latency of each event. As expected, these histograms showed a clean narrow peak with no structure.

The parameters were measured as a function of the message size from the RoB to the FeX and as a function of the maximum number of events allowed in the system at any one time by the supervisor.

One million events were passed though the system for each configuration and data size, so a total of about 1 billion events or 30 billion SCI packets were used in these tests.

A logic state analyser was used with the SCI breakout cards [22] to examine the behaviour of the FeX and Global PCI busses and various points on the SCI network. The analyser was triggered by issuing a read to the PCI configuration space of the SCI interface when a trigger fragment had completed and arranging the timebase of the analyser to display the bus and network activity.

## 6 Results and Analysis

### 6.1 A Single Pipeline

Figures 6.1 and 6.2 show the event latency and the corresponding average time between events, for a single RoB-FeX-Global stripe measured as a function of the number of events allowed in the loop by the supervisor for various RoB to FeX data sizes. The average time between events is the inverse event rate. With only one event present at a given time, the average time per event is equal to the loop latency. At this point, the latency is the sum of all the individual processing or transfer times less any times that overlap. ( For example, during the time spent transferring a large message out of the RoB in DMA mode, the FeX is processing the packets in its ring-buffer. ) When more events are allowed into the system, the average time/event decreases until with about 4 events all the processing steps are busy and no further improvement is observed. Under these conditions, events are queuing in the system and the average time per event is now equal to that of the slowest element in the loop, shown by the flat sections in Figure 6.2. The latency, shown in Figure 6.1, is initially almost flat and then increases linearly with the number of events when the event queue. When counting the number of events, it is the number in the loop that is important, not the number allowed by the supervisor. In fact:

The number of event actually in the loop = the latency / average time per event

Following the simple analysis given in [6] for the two domains of the latency plots, we can write

$$\text{Fundamental component latency} = \sum_i T_i - \sum_j T_j$$

and

$$\text{Latency} = \max ( \sum_i T_i , N_q T_{\max} )$$

Where

- $T_i$  component latencies
- $T_j$  component latencies that overlap
- $N_q$  Number of event in the system
- $T_{\max}$  the largest or largest set of component latencies

Figure 6.1 shows that the latency also increases with the amount of data send over the network between the RoB and FeX. For a non saturated network, the time to transmit data follows a linear relation:

The time to transfer data = overhead + number of bytes / bandwidth

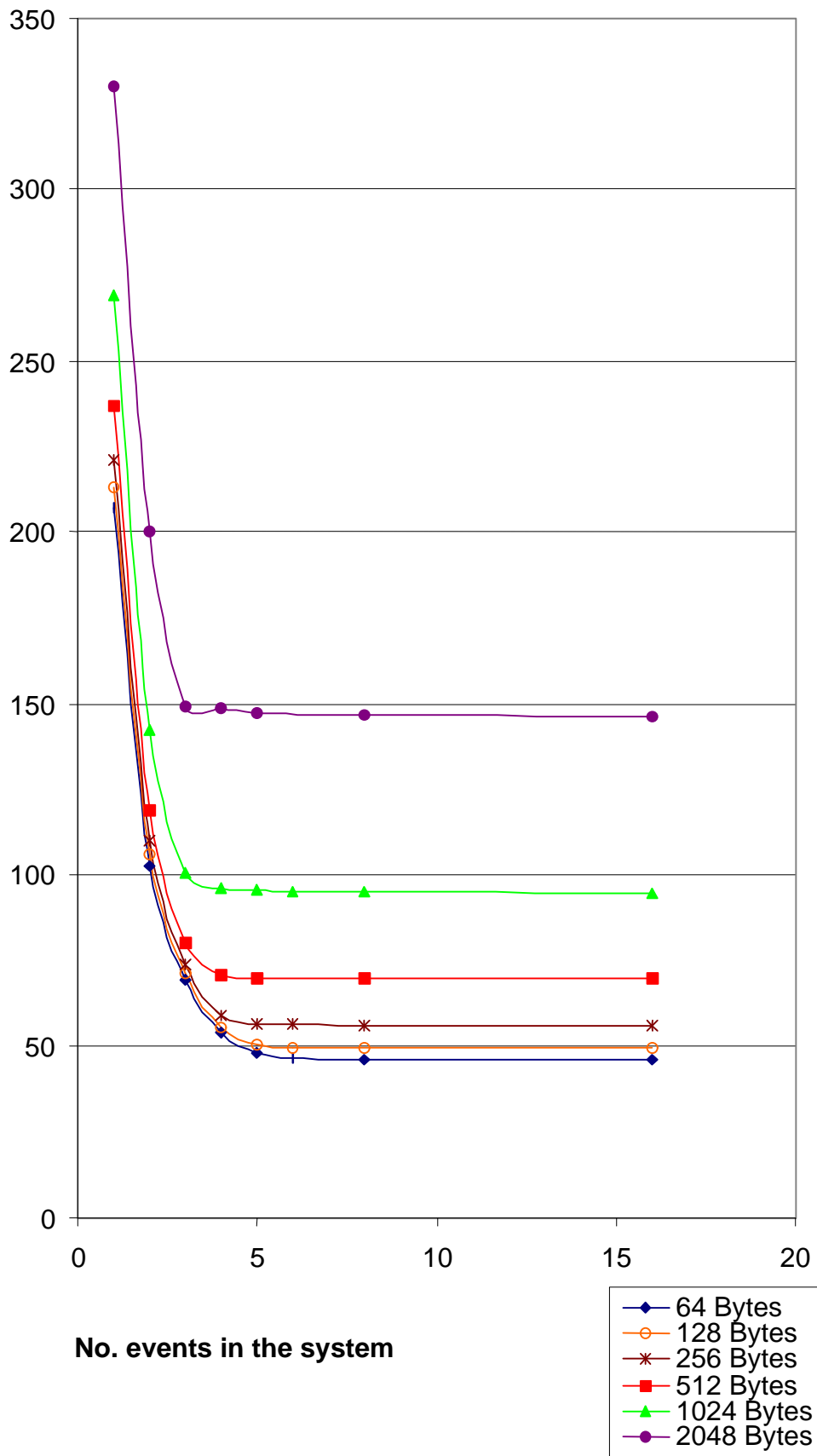


Figure 6.1. Latency vs no. events in system 1RoI/event 1RoB 1FeX 1Global

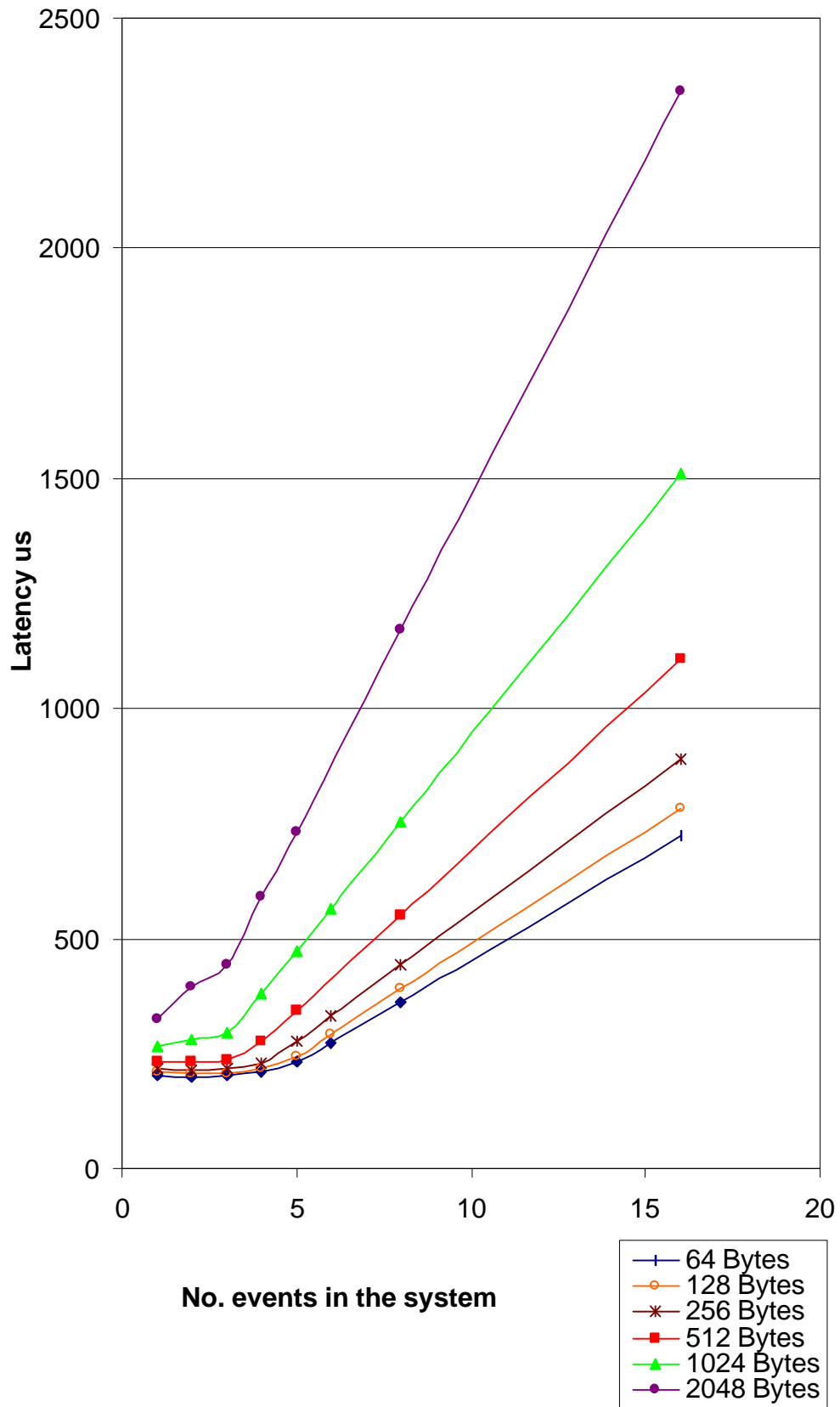


Figure 6.2. Ave. time / event vs no. events 1RoI/event 1RoB 1FeX 1Global

So it would be reasonable that  $T_{max}$  had the same linear variation with the amount of data send

$$T_{max} = T_0 + AB$$

- $T_0$  Transfer time overhead
- A inverse effective bandwidth
- B number of bytes being transferred

This linear dependence is demonstrated in Figure 6.3 which plots  $T_{max}$  against the data payload for one stripe with SCI operating in DMA mode. From the graph, the slope gives an effective bandwidth of 19.8 MBytes/s.

From the data we have

$$\sum_i T_i \quad 202 \mu s$$

$$T_{max} \quad 46 \mu s \text{ for } 64 \text{ bytes and } 94.5 \mu s \text{ for } 1 \text{ Kbyte}$$

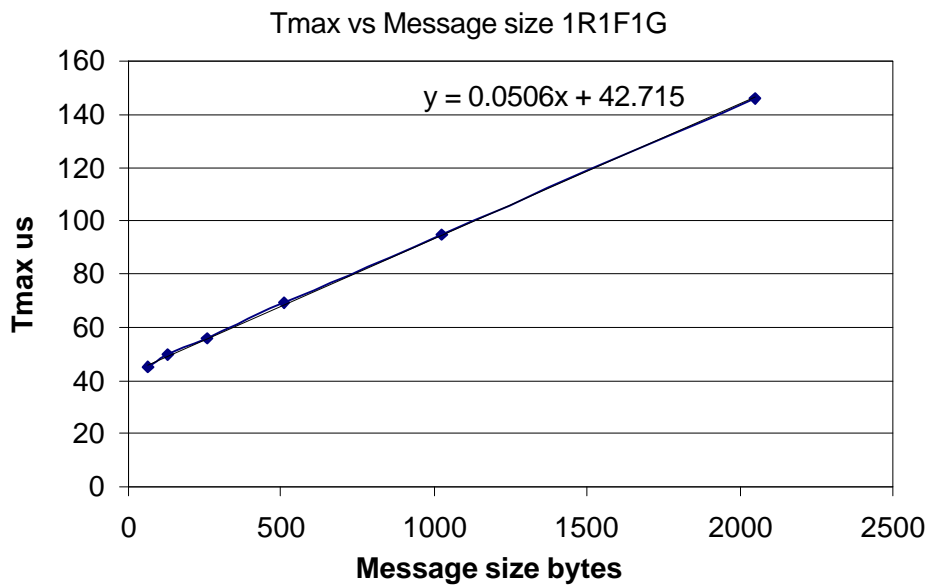


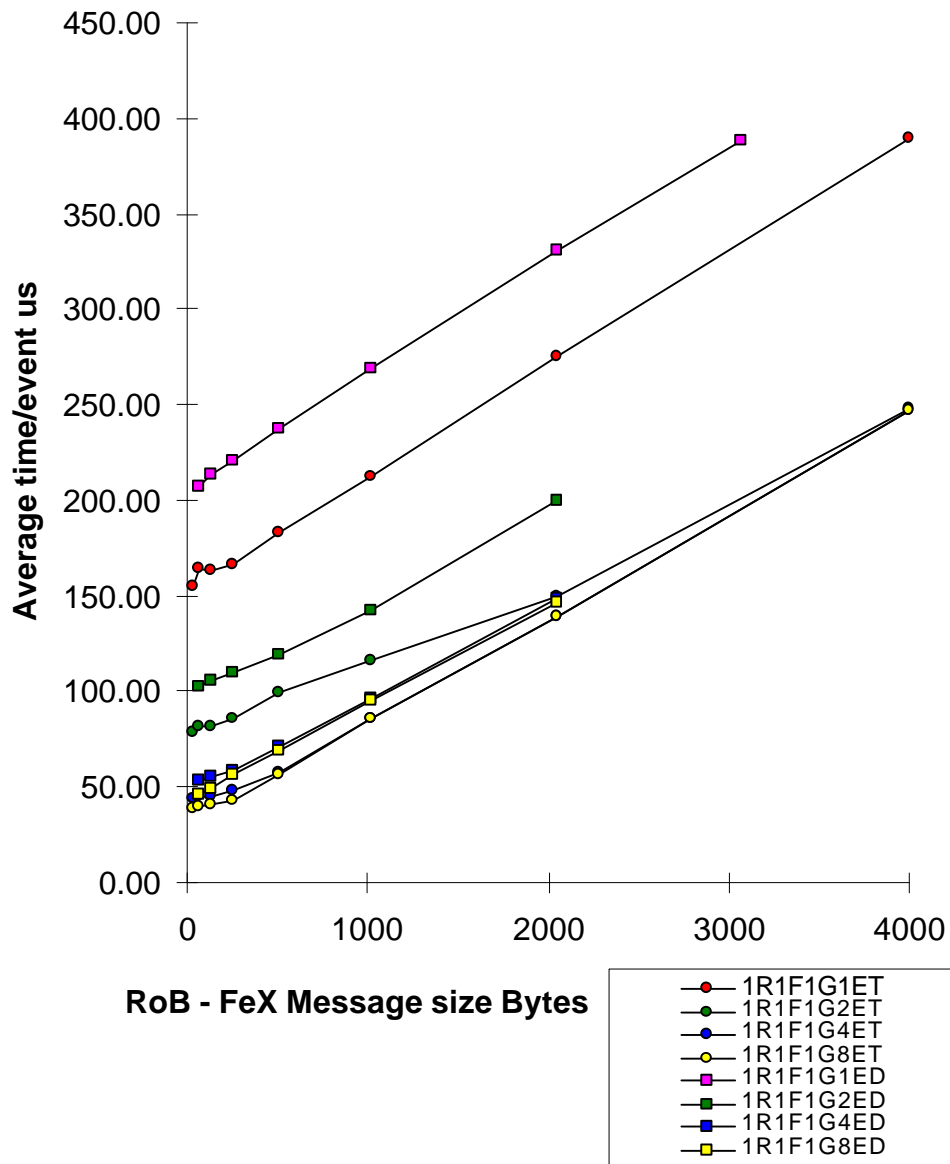
Figure 6.3. The variation of  $T_{max}$  with the amount of data sent from R0B to FeX.

Component	Time $\mu s$
Supervisor	34
Network server	10
RoI Distributor read Slink	17
RoI Distributor	22.5
RoB	6.5
DMA onto SCI	16
FeX	<40
Global	<40
Total	186

Table 6.1. Component times round the loop

Table 6.1 gives the component time for each component in the loop the total of 186  $\mu\text{s}$  is in reasonable agreement with the 202  $\mu\text{s}$  measured. One should note that many of the software component times have large uncertainties due to the coarse time intervals provided by the measurement tools.

In the following graphs the curves are labelled with rR fF gG with r,f,g giving the number of RoBs, FeXs and Global processors; eE indicates the number of events allowed in the system, D or T denotes DMA-ringbuffer mode or Transparent mode and S is appended if the SCI switch was used.



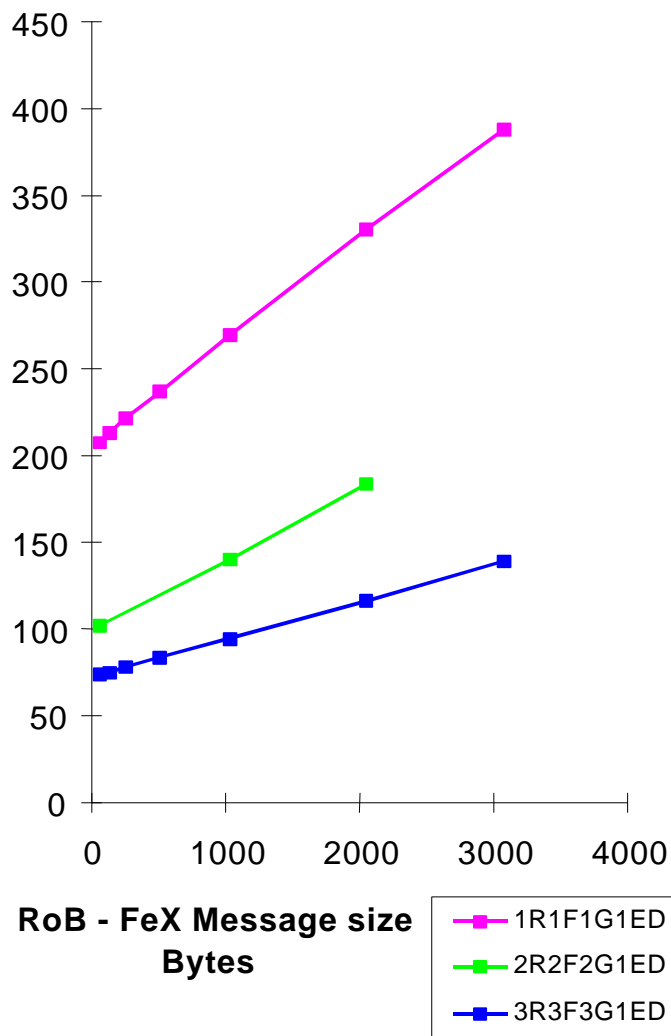
**Figure 6.4. The Average time per event as a function of the RoB – FeX messages size for a single RoB-FeX-Global stripe. Various numbers of events were allowed into the system for both DMA and Transparent modes.**

Figure 6.4 shows the average time per event for all the conditions measured without the switch. From the data we observe the following:

- For large message sizes, say over 256 bytes, the curves are linear indicating that events are queuing in the system. Once the events are queuing, then the time is determined by the slowest element in the chain.
- The DMA transfers have a larger overhead, as expected from the DMA setup and packet processing.

- The slopes for both DMA and Transparent modes are very similar, typically 20 MBytes/s and 18.2 MBytes/s respectively. This suggests that the RoB to FeX transfer times are the important element, and that the extra packet handing in the FeX when using DMA mode is overlapped with the time taken to move the data out of the RoB onto SCI. In transparent mode, there is no extra handing in the receiving CPU but the sending CPU is occupied for longer in sending the data.
- For short, 64 to 128 byte records, and multiple event in the system, the minimum time is 40μs which suggests that the RoI Distributor is the slowest item, see Table 6.1. As the amount of data increased, the RoB –FeX transfer time takes over.
- The 1R1F1G2ET curve is interesting. For small event sizes, it is limited by the RoI Distributor, then the SCI transfer time becomes larger but no queue forms as the two events follow each other round the loop without interference. This accounts for the slope being a factor of 2 less that the other curves. When the data length is over 2 kbytes, the events have to queue and the slope becomes the same as the others.
- Over the interval where the RoI Distributor is limiting, there is a small but significant reduction in the speed of the Distributor. The RoI Distributor polls the RoBs over VME backplane and the RoB VME/PCI bridge, as the SCI traffic increases, these polls encounter delays due to arbitration for the PCI bus in the RoB.

## 6.2 Parallel Pipelines



**Figure 6.5. Average time / event for events in parallel RoB-FeX-Global Stripes for 1 event/stripe using SCI in DMA-Ringbuffer mode.**

Figure 6.5 shows the average time per event as a function of the message size for one, two and three parallel stripes, as described in Figure 4.1, with the SCI operating in DMA-Ringbuffer mode. The

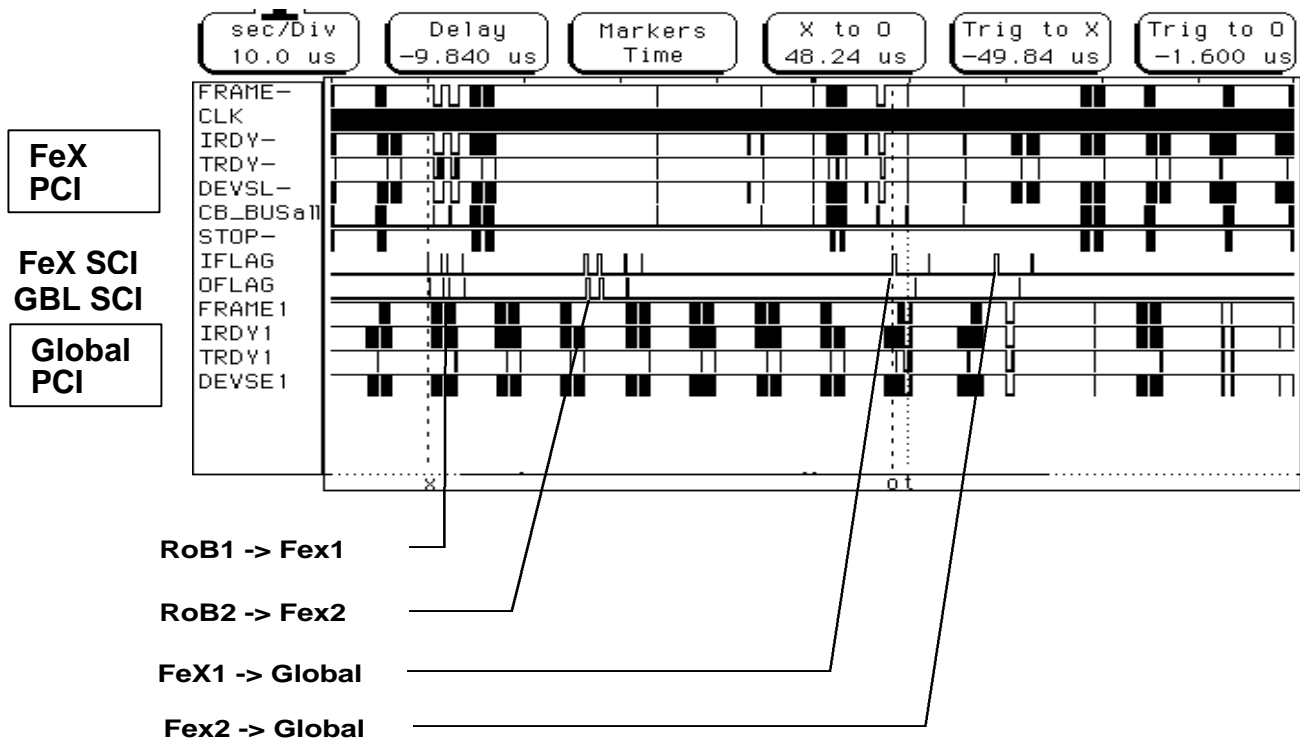
curves are linear and there is very good scaling, the average times decreasing with the number of stripes in use. Table 6.2 demonstrates that the intercept and slope of the linear fits to the points decrease almost in proportion to the number of pipelines in use. This is for the case of one event allowed into each stripe.

	Slope	Intercept
1R1F1G1ED	0.06 (1.0)	205.8 (1.0)
2R2F2G1ED	0.041 (0.68)	99 (0.48)
3R3F3G1ED	0.022 (0.36)	72.7 (0.35)

**Table 6.2. The Slope and Intercept of linear fits to the average times/event demonstrating the scaling with different numbers of stripes in use. The numbers in () show the ratios.**

In going from 1 pipeline to 2 one would expect a factor of two increase in the throughput or decrease in the average time per event, but the RoI Distributor is a shared resource, with the second RoB serviced ~16  $\mu$ s after the first. Also the speed of the first FeX was 233 MHz, but the second and third were only 166 MHz systems. These factors would account for the ratios observed.

The behaviour of the RoI Distributor and the RoBs was confirmed by using a HP logic analyser to probe the PCI buses of the FeX and Global processors at the same time as observing the behaviour of the SCI ringlet. The trace obtained for two stripes is shown in Fig 6.6. Here the time between the outputs from RoB1 and RoB2 is 16.4  $\mu$ s and is due to the sequence time in the RoI Distributor.



**Figure 6.6. Logic Analyser trace of the PCI buses in the FeX and Global processors and the activity on the SCI Ringlet. The time between the signals from RoB1 -> Fex1 and RoB2 -> Fex2 is 16.4  $\mu$ s corresponding to the RoI Distributor sequence time.**

Figures 6.7 and 6.8 show the average times per event for 2 and 3 RoB-FeX-Global stripes multiple events were allowed in each stripe. Due to the time to distribute events in the RoB Complex, the average time between events being presented to each stripe increases with the number of stripes in use, for example this is ~48  $\mu$ s for 3 stripes. This implies that for 3 stripes, the time to send data between the RoB and the FeX does not become the dominant time until it exceeds ~48 $\mu$ s which corresponds to a message length of ~850 bytes. The curves for 4,5, or 8 events shown in Figure 6.8 are essentially flat up to this message length with an average time ~40  $\mu$ s, typical of the time taken by the RoI Distributor.

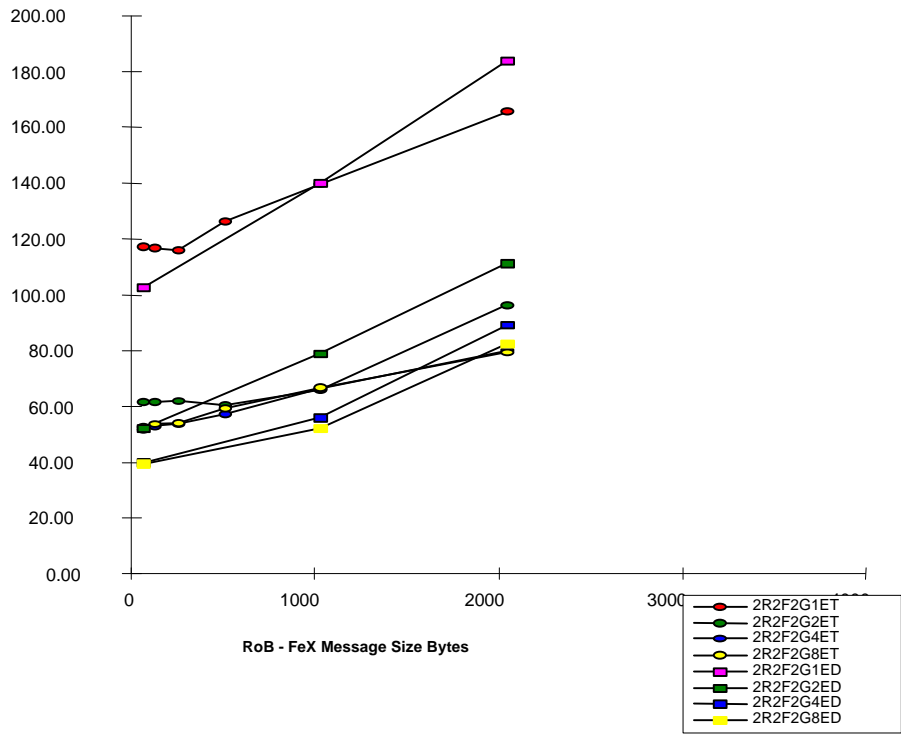


Figure 6.7. The Average time per event as a function of the RoB – FeX messages size for two parallel RoB-FeX-Global stripes. Various numbers of events were allowed into the system for both DMA and Transparent modes.

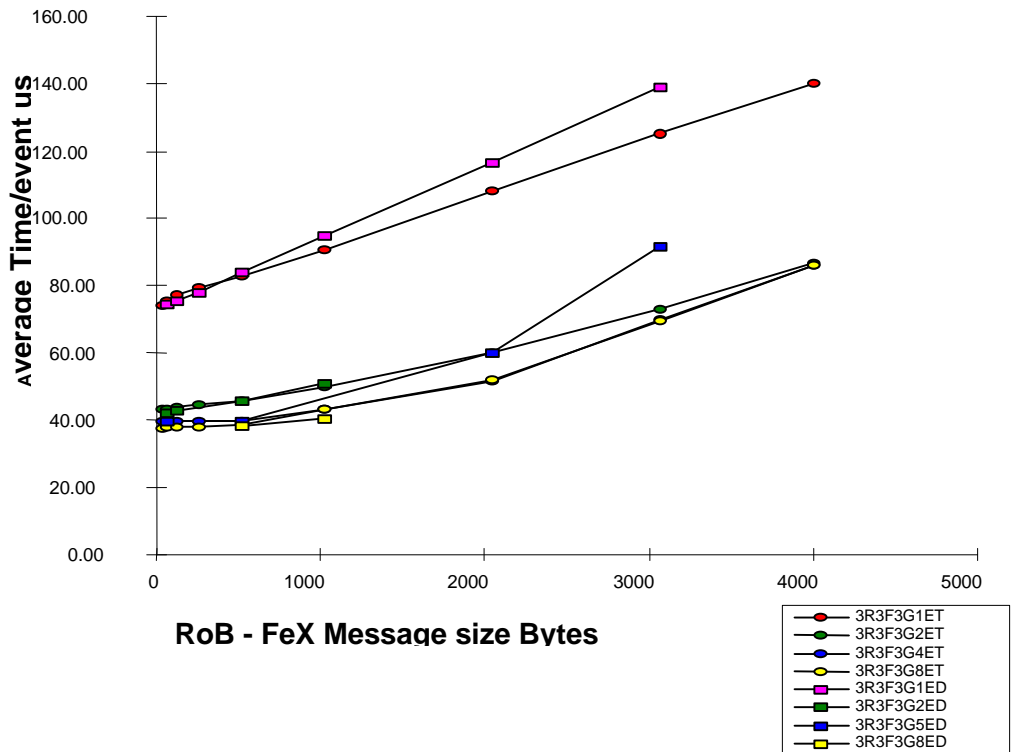


Figure 6.8. The Average time per event as a function of the RoB – FeX messages size for three parallel RoB-FeX-Global stripes. Various numbers of events were allowed into the system for both DMA and Transparent modes.

### 6.3 RoI Building

The effect of event building the trigger data from several RoBs into one FeX, as shown in Figure 4.3, was investigated. Figure 6.9 shows a comparison of the average times for RoI building with data from one, two and three RoBs using the SCI in DMA-Ringbuffer mode, and Figure 6.10 shows the comparison for Transparent mode.

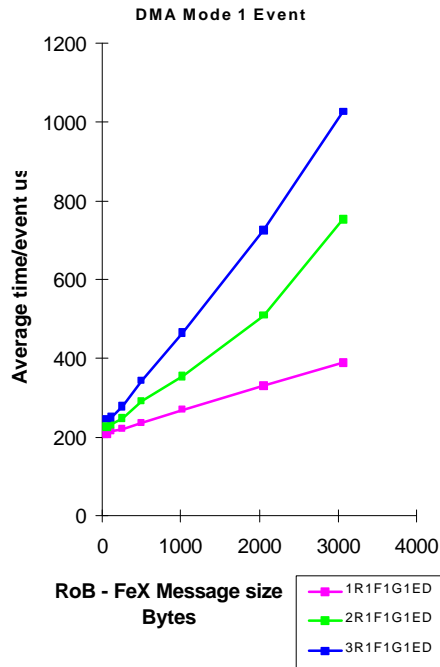


Figure 6.9. RoI Building with multiple RoBs 1 FeX 1 Global, SCI in DMA-Ringbuffer mode.

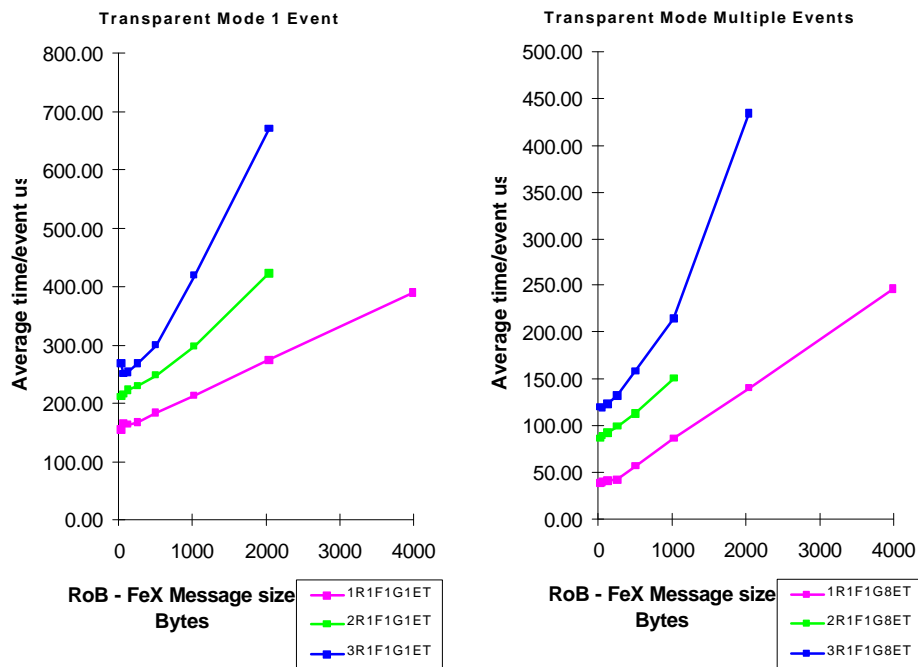
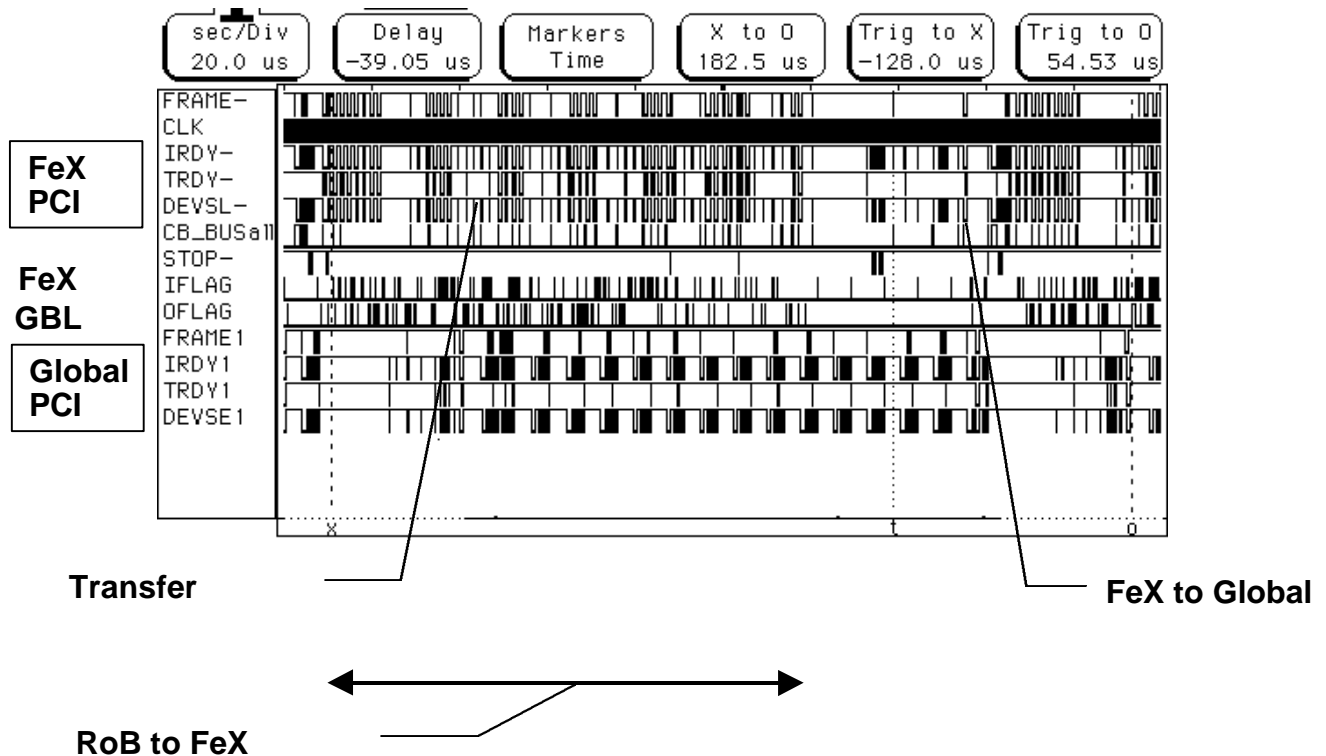


Figure 6.10. RoI Building with multiple RoBs 1 FeX 1 Global, SCI in Transparent mode.

For small data transfers below 128 bytes from each RoB to the FeX, the average time /event increased by  $\sim 20 \mu\text{s}$  per active RoB when using SCI in the DMA-Ringbuffer mode. This is consistent with the  $16\mu\text{s}$  required to distribute RoI information to each extra RoBs as discussed in section 6.2. Each message from RoB to FeX will pass over SCI sequentially, as Figure 6.6 shows, there will be no overlap of the data from each RoB. For larger data transfer sizes, the time to transmit the data from the RoBs to the FeX becomes important, and the data from the second RoB will overlap on SCI with the data from the first RoB. The FeX had to receive data from all the RoBs involved before the event building was complete for a given event.

The difference in the slopes of the curves for one, two and three RoBs seems dramatic, but allowance must be made for the extra data sent into the FeX when using multiple RoBs. The aggregate data rate was 22.4, 11.2 and 11.4 MBytes/s corresponding to using one, two and three RoBs. Similar behaviour was observed when multiple events were allowed in the system.

When SCI was used in Transparent mode, the average time /event increased by  $50 \mu\text{s}$  going from one to two RoBs and by  $36 \mu\text{s}$  going from two to three. It is believed that the message passing protocol and the fragment building account for these times. The aggregate data rate under these conditions was 20, 15 and 16 MBytes/s corresponding to using one, two and three RoBs.



**Figure 6.11. Probing the PCI and SCI during transfers from 2 RoBs to 1 FeX. The “pauses” are  $\sim 10\mu\text{s}$  long.**

Monitoring the SCI and PCI busses during the event building of trigger data from two and three RoBs showed that there were “pauses” in the transfer over SCI when multiple events were allowed into the system. These “pauses” reduce the effective SCI transfer by nearly a factor of 2. Figure 6.11 shows a logic analyser trace when building the RoI from data from 2 RoBs with 4 events in the system. The  $\sim 10\mu\text{s}$  “pauses” were due to contention for the PCI bus in the RoBs between the DMA engine on the SCI card sending the data to the FeX over SCI, and the RoI Distributor polling the RoB memory via the VME-PCI bridge. This effect would account for the different aggregate data rates observed in the DMA-Ringbuffer mode where the PCI interfaces have equal priority on the PCI bus. For transparent mode, the data is written by the CPU which normally has higher priority; this would account for the smaller reduction in the observed aggregate data rates.

Figure 6.12 and Figure 6.13 show the average times/event for all the conditions measured for two and three RoBs.

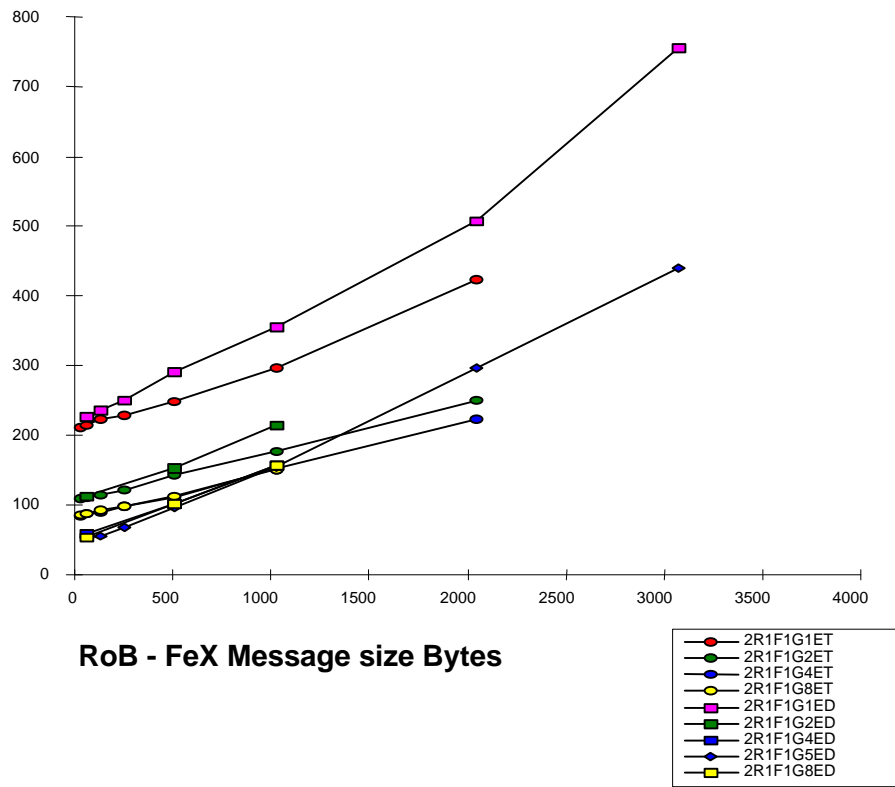


Figure 6.12. Trigger fragment building in the FeX, with 2 RoBs 1 FeX 1Global

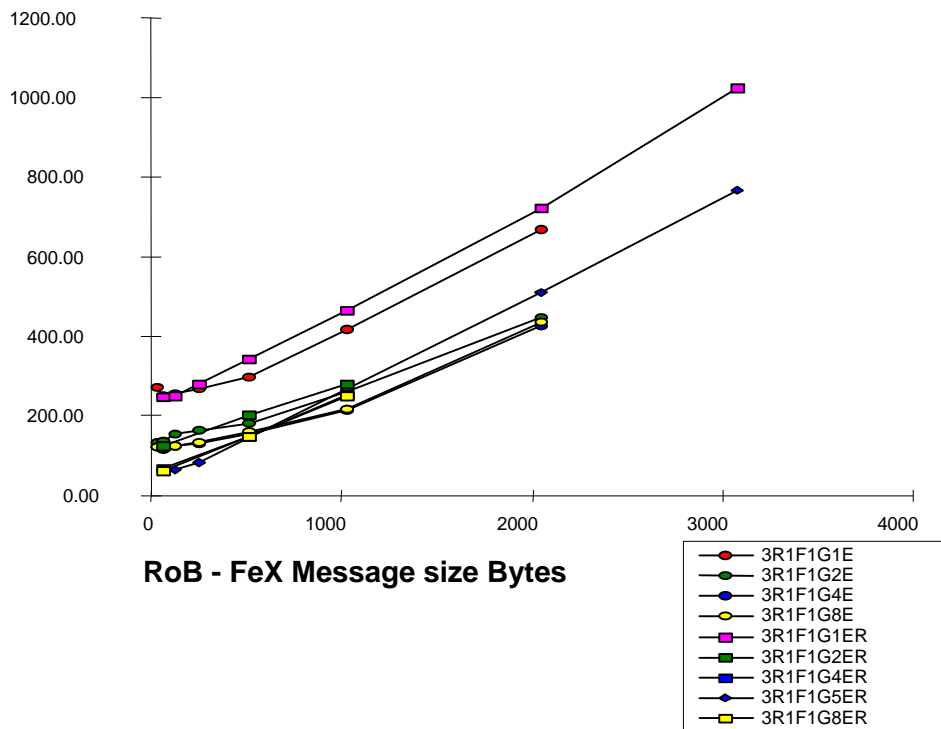


Figure 6.13. Trigger fragment building in the FeX, with 3 RoBs 1 FeX 1Global

## 6.4 RoI Parallelism

Measurements were made with one, two and three RoB – FeX combinations feeding into one Global processor. Depending on the configuration, each event contained one, two or three RoIs, each involving just one RoB. Since the Global had to wait for 64 bytes of data from each FeX before making the decision message to the supervisor, the performance would be similar to that of a single stripe if there were full parallel RoI processing in the FeXs. However, the time to distribute the RoIs to the FeX, discussed above and the time to event build the messages from each FeX will affect the performance. The data are shown in Figure 6.15.

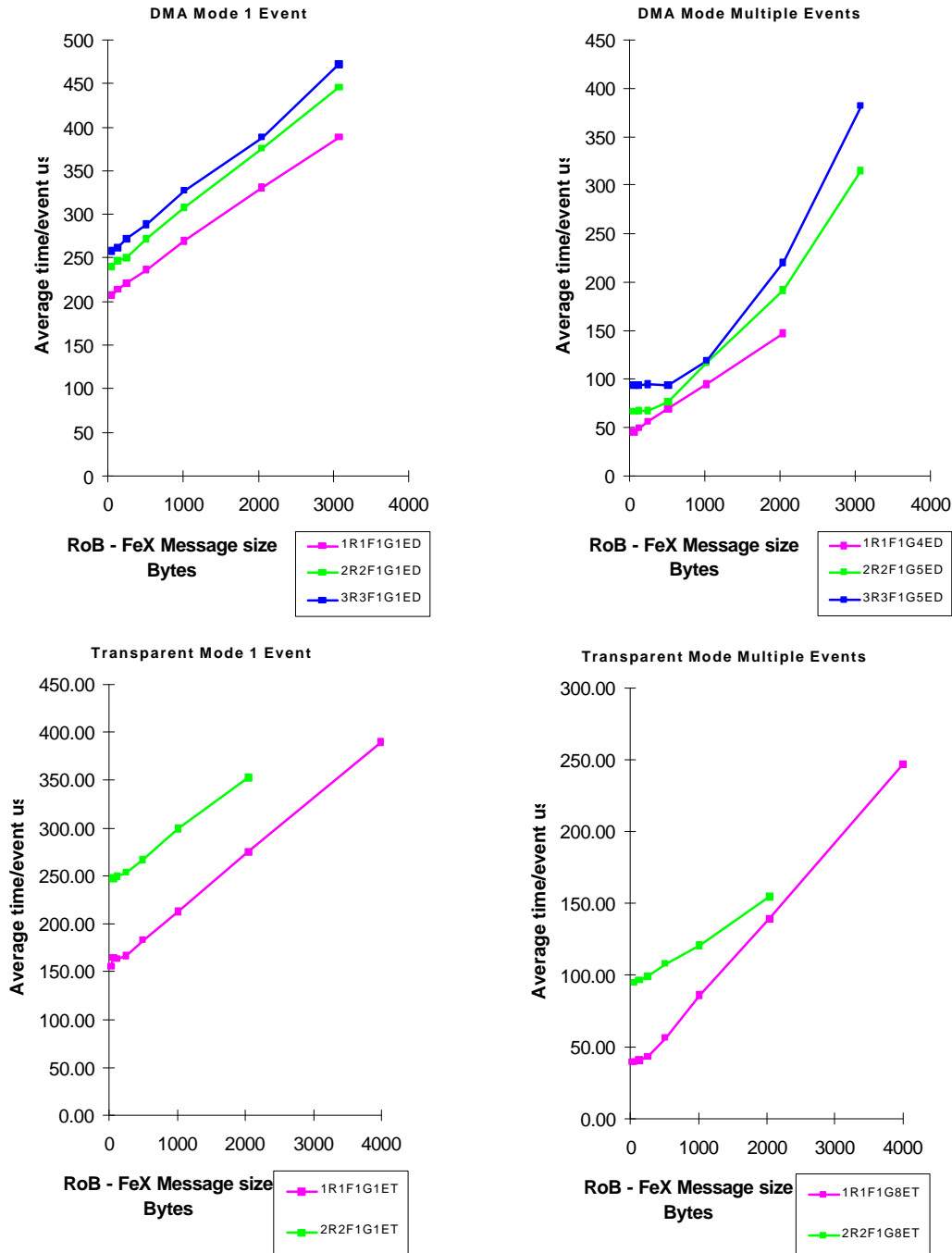


Figure 6.15. RoI Parallelism, one two or three RoIs are processed in parallel and collected by one Global processor.

## 6.5 Use of the SCI Switch

The configuration shown in Figure 3.2 was used to investigate the performance of the vertical slice when a SCI switch was included. No software or changes of the SCI node addresses were required, the only change was to ensure that each ringlet had a node configured as a SCI scrubber. Earlier tests of the SCI switch [21] indicated that it would introduce an extra delay of  $\sim 1.5 \mu\text{s}$  in the packet latency.

Figure 6.16. shows a comparison of the performance with and without a switch. The improvement could be attributed to the following:

- There were less nodes on each ringlet connected to the switch and this reduced the time to send a packet around the ring from RoB to FeX.
- There was less traffic on each ringlet, reducing the chance of any delay in accessing the network.

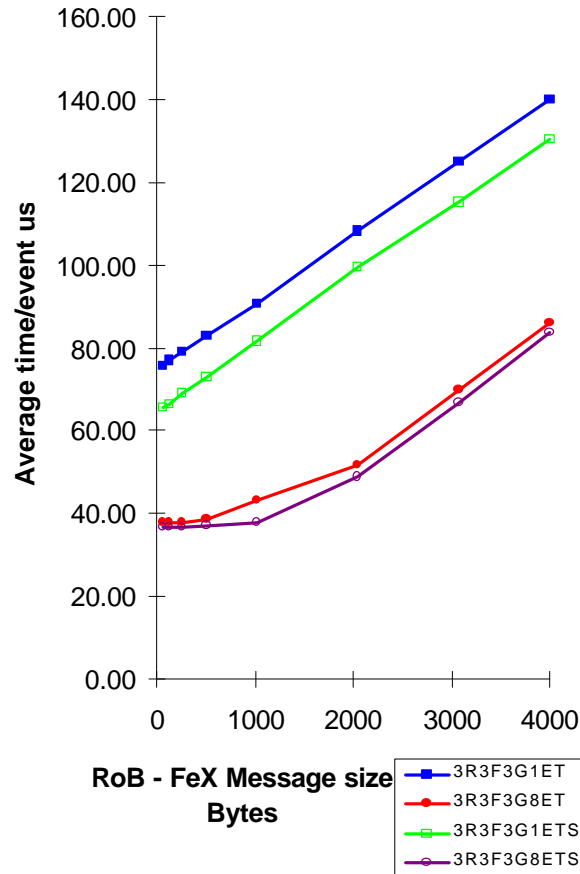


Figure 6.16. Average time per event with and without using the SCI switch. Three RoB-FeX-Global stripes were used, in Transparent mode when 1 and 8 events were allowed in the system.

## 7 Conclusion and Application to ATLAS

The Level 2 vertical slice tests using SCI have demonstrated high rate, 26 kHz, operation of all the components used in the closed loop system. They have shown the general stability and suitability of SCI for the ATLAS Level 2 trigger and DAQ and that the Local-Global Architecture could be used for ATLAS. Scalability in terms of stripe pipelining, event parallelism, fragment building and RoI parallelism has been demonstrated for typical ATLAS data traffic, although further study is required.

### 7.1 The Suitability of SCI

- SCI was confirmed to be a stable and flexible network topology, which could support point-to-point, ringlets and switches.
- The applications do not have to worry about dropped packets, the low level protocols at the chip level take care of packet re-try in the case that the destination is busy.
- The SCI resource allocation protocols ensure forward progress of data.
- The SCI bandwidth allocation protocols ensure that all nodes have equal access to the network
- SCI can write directly to memory, avoiding multiple software copies in the driver.
- The SCI DMA interface acts at the message level providing automatic packetisation, and this also happens in transparent mode.
- Extremely low error rates.
- The inclusion of a SCI switch improved performance despite the small extra delay introduced.

### 7.2 Application to the ATLAS trigger.

The work on the paper model [20] has produced estimates for the average RoB request rates as shown in table 7.1 for low Luminosity and table 7.2 for high Luminosity. Table 7.3 shows the FeX input rates. Even allowing for the safety factor of 2, the SCI implementation of the Local Global architecture can provide the required bandwidth and latency.

	Trigger RoI only	All RoIs Trigger + secondary	All RoIs Sequential selection	Add EtMiss to selection	Add bjet to selection
ECAL	0.99	3.67	1.87	4.17	4.17
HCAL	2.53	8.10	4.30	6.60	6.60
MUON	0.22	0.23	0.23	0.23	0.23
SCAN	4.10	4.13	4.13	4.13	4.13
TRT	0.43	0.51	0.23	0.23	0.23
SCT	1.69	1.79	1.47	1.47	5.18

**Table 7.1. Average RoB request rates in kHz for Low Luminosity**

	Trigger RoI only	All RoIs Trigger + secondary	All RoIs Sequential selection	Add EtMiss to selection	Add bjet to selection
ECAL	1.84	2.5	1.23	2.23	2.23
HCAL	4.49	5.93	3.18	4.18	4.18
MUON	0.24	0.24	0.24	0.24	0.24
TRT	0.65	0.71	0.21	0.21	0.21
SCT	0.64	0.72	0.17	0.17	1.47

**Table 7.2. Average RoB request rates in kHz for High Luminosity**

	High Luminosity	Low Luminosity

CAL	13.9	14.8
MUON	10.5	9.7
TRT	8.1	14.7
SCT	4.5	4.8

**Table 7.3. FeX input rates in kHz.**

### 7.3 Summary

- The maximum closed loop event rate achieved was 27 kHz
- The Supervisor limit in loopback was 33 kHz
- Highest data transfer rates achieved , they would have been higher if not limited by the VME transfers in the system.
- Found the data transfers more serial than expected due to shared components and use of VME.
- Need for care when implementing with shared components/buses.
- The SCI network was stable and robust and did not require data compression.
- The SCI hardware protocols ensure equal access for all nodes and forward progress of the data.
- The interfaces and SCI network can operate at the message level as the software does not have to worry about packet loss.
- Both DMA and Transparent modes have advantages - use of an optimum mix.
- Error detection and rudimentary recovery were included and tested during setting up of the system !
- Run control and configuration were essential but need more work on the software. This would be an ideal workpackage for the Pilot Project.
- The Local-Global model was stable and worked well.
- The architecture and technology met the data rates calculated in the paper model.
- There is confidence that the architecture and technology would meet the requirements of the ATLAS trigger.

Many people from many institutes have successfully cooperated and worked together to achieve DemoB

## 8 References

- [1] [SCI] IEEE Computer Society, IEEE Standard for Scalable Coherent Interface (SCI), IEEE Std 1596-1992 (recognised as an American National Standard (ANSI)), August, 1993.
- [2] J.R.Hansen, "Local-Global Demonstrator Programme for the ATLAS Second Level Trigger", Presented at the X IEEE Real-Time conference Beaune September 1997
- [3] F. Wickens, "Proposed Data Formats for T2 Demonstrator Program", January 1997, [http://hepwww.rl.ac.uk/atlas/l2/demonstrator/docs/demons\\_data\\_defs.html](http://hepwww.rl.ac.uk/atlas/l2/demonstrator/docs/demons_data_defs.html)
- [4] ATLAS Technical Proposal. CERN/LHCC/94-43, 1994
- [5] J.C.Vermeulen et al., "Performance requirements of proposed ATLAS second level trigger architectures from simple models", presented at CHEP97 by S. George, Berlin, Germany, April 1997.
- [6] P. Maley et al, "A Local-Global Implementation of a Vertical Slice of the ATLAS Second Level Trigger", DAQ-NO-81, 1997
- [7] L64601 SCI NodeChip Technical Manual, LSI Logic Corporation.
- [8] "PCI-SCI Cluster Adapter Specification", Dolphin Interconnect Solutions, 1996
- [9] "Link Controller LC-1 Specification", Dolphin Interconnect Solutions, 1995
- [10] R.Blair et al., "The ATLAS Level-2 Supervisor", LHC Electronics Workshop, Balatonfüred, Hungary, Sept 1996.
- [11] R. Blair, "Supervisor for the Demonstrator Project", ATLAS DAQ-note in preparation, 1998
- [12] "RIO II User Guide", CES, Geneve
- [13] O. Boyle et al, "The S-Link specification", Dec 1995  
<http://www.cern.ch/HIS/s-link/spec>
- [14] "PMC to PCI Adaptor Product Manual", Technobox Inc. P/N 1447
- [15] E. Denes, "Local RoI Distributor and RoB Emulator Programs for ATLAS LVL2 Demonstrators. User's Guide", 1997
- [16] "Digital AXppci33 Alpha AXP PC Motherboard OEM Design Guide", Digital Equipment Corporation EK-AXPCI-DG.C01, 1995
- [17] "Multia MultiClient Destop Service Information", Digital Equipment Corporation EK-MULTS\_IN.C01, 1995
- [18] "Alpha Architecture Reference Manual", ISBN 1-55558-098-X, Digital Equipment Corporation
- [19] J.J.Labrosse, "MicroC/OS - the Real-Time Kernel", R&D Publications Inc, Distributed by Prentice-Hall, ISBN 0-13-031352-1, (1992)
- [20] S.George, Paper Model Calculations, ATLAS DAQ-note 1998
- [21] R. Hughes-Jones, "Experience using the LC1 Chip, Blink, and the Dolphin 4\*4 Switch", ATLAS DAQ-NO-075
- [22] See <http://www.hep.man.ac.uk/~rich/sci/breakout.htm>