Integration of VICbus, FDL, SCI and Ethernet in the CERN CASCADE data acquisition system

J.Petersen, C. Bizeau, A.Bogaerts, D. Burckhart, R.Divià, L. Ingebrigtsen [1], M.Joos, M.Liebhart, J.Niewold, C.Parkman, Y. Perrin, B.Thiercelin, L.Tremblet, A.Vascotto, P.Werner

ECP Division, CERN, 1211 Geneva 23, Switzerland

[1] University of Oslo, Department of Informatics, Oslo, Norway

Abstract

CASCADE is a multi-processor real-time data-acquisition system for HEP experiments developed at CERN by the ECP-DS group. Configurations supported today include VMEbus processors running OS-9 and UNIX workstations. The CASCADE data acquisition processes, called stages, communicate via links, at present VICbus between VME crates and Ethernet between VMEbus processors and workstations. Work is in progress to introduce new inter-stage links based on the Fast Data Link between VME crates and on SCI for data exchange between SUN stations.

The paper gives a short description of the architecture of CASCADE with emphasis on the link aspects. The implementation and current status of the inter-stage links based on VICbus, Ethernet, FDL and SCI will be described and results on the performances presented.

I. INTRODUCTION

CASCADE [1] [2] [3] is a distributed, multiple-platform real-time data-acquisition system developed by the ECP-DS group for HEP experiments at CERN. It provides services for data collection and buffering, data flow control, event building, event sampling for on-line monitoring, data recording and run control. Originally developed following a request from the NOMAD experiment [4], it has been designed to adapt to a wide range of applications and system configurations. Its modular structure facilitates the integration of new technologies, as for example new links for the communication between CASCADE processes (stages).

After a general description of CASCADE with emphasis on the inter-stage links, the lower-level software for the VICbus [6], Fast Data Link (FDL) [8] and Ethernet is described in some detail and performance figures are given. It should be pointed out that the intention has not been to perform a complete evaluation of the link hardware interfaces, only features relevant to the CASCADE integration have been studied in detail.

A CASCADE test system based on VMEbus and including the FDL, VICbus and Ethernet links is then described and the performances in terms of event overheads and transfer rates are presented. These results - measured in a realistic complex data acquisition environment - may be compared with the often more theoretical figures given by the manufacturers.

For the SCI, work is in progress to demonstrate a parallel event builder based on four SUN workstations, using CASCADE. Since a VMEbus to SCI interface is not yet available, a direct comparison with the FDL and VICbus links is not possible at the present time.

II. THE CASCADE DATA ACQUISITION SYSTEM

CASCADE data-acquisition systems are built from two basic construction units, the *stage* and the *inter-stage link*. The stage is a single process which provides the basic data acquisition functions : event data I/O and buffering, event building (if required), distribution of events to monitoring programs and connection to run control. Two stages are connected via the inter-stage link which is the term used for the ensemble of link software and hardware which allow stages to communicate. This communication is based on a high level protocol and a software interface which hide the specific features of the link to the internals of CASCADE.

Based on these two construction units, configurations of varying complexity can be built, possibly including several types of processors and links. An example is given in Fig.1, which shows the present CASCADE configuration of the NOMAD experiment. Five front-end stages are linked to an event builder stage (via VICbus) which in turn is connected to a data recorder (via shared memory).

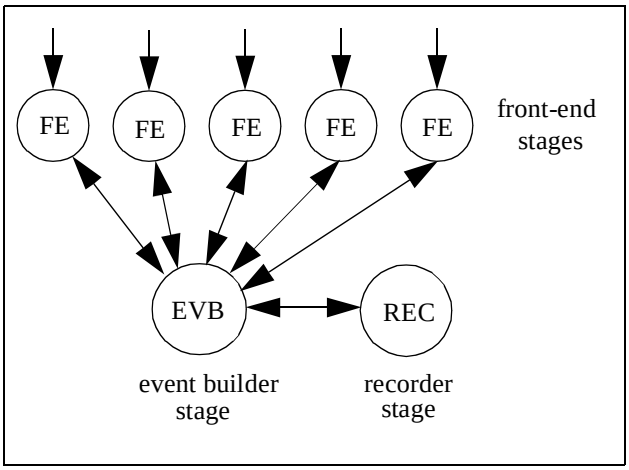

Fig. 1. CASCADE configuration of the NOMAD experiment

The stages and their inter-connections are described in the CASCADE *configuration file* which maps a logical representation of the type shown in Fig.1 onto a physical implementation. Each stage is described by a number of parameters which define the event buffer size, the number of input and output ports and their characteristics. To a large extent the user controls the functioning of the CASCADE system via the parameters in the configuration file. This applies, for example, to the inter-stage links which are

entirely specified via entries in the stage input/output ports section of the configuration file. User code has to be provided for reading event data and other interfaces are defined for including application dependent functions related to event building and run control. In addition, the user is responsible for the development of the monitoring programs which are independent processes.

2.1 The Stage

The stage is the fundamental construction unit of CASCADE and provides the basic functions of a general single-processor, single-process, data-acquisition kernel. It is organised in several threads of execution which allows operations to take place concurrently on different events so that the stage can deal with a number of input and output ports at their individual rates. Each thread corresponds to a given operation to be performed on an event or to a control action originating from the run control process, see Fig.2.

A scheduler has been developed to control the execution of the threads. Each thread is triggered by a given signal which is issued either externally by other processes or internally by one of the other threads [1]. The scheduling in the stage is performed on a priority basis and has no fixed sequence. This is illustrated in Fig.2, which also indicates the threads that are activated by external signals.

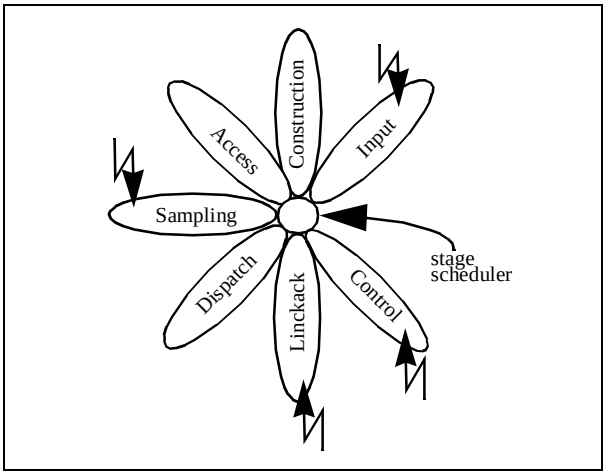

Fig. 2. Stage threads

2.1.1 The event data flow through the stage

Threads directly involved with the main data-flow are called *phases*. The transition of an event through the stage phases follows a fixed sequence and is controlled via signals generated by the experiment trigger, the handshake between connected stages, monitoring requests, or by another thread.

An event in CASCADE has two components, a descriptor and data, which may flow independently through the system. An event is created when the stage receives a trigger and disappears when all phases have terminated their work related to this event. Upon reception of a trigger, events pass sequentially through (see Fig.3):

- *the input phase*, which creates an event descriptor and reserves memory space, reads the event data from the experiment or copies the event data from the previous stage,
- *the construction phase*, which, in the case of an event builder, links together the descriptors of the sub-events until a complete event is created,
- *the access phase*, where the events are marked for monitoring,
- *the dispatch phase* which formats and outputs the events to one or several other stages (e.g. a data recorder),
- *the link-acknowledge phase* which marks the events to be released once the transfer through a particular output port has been acknowledged.

A set of *sampling phases* handles the connection and event requests issued by monitoring programs attached to the stage. After connection to a stage, a monitoring program issues requests for events and receives in return the relevant pointers and sizes so that it can access the event. The monitoring program has to release the event before it can be removed from the stage.

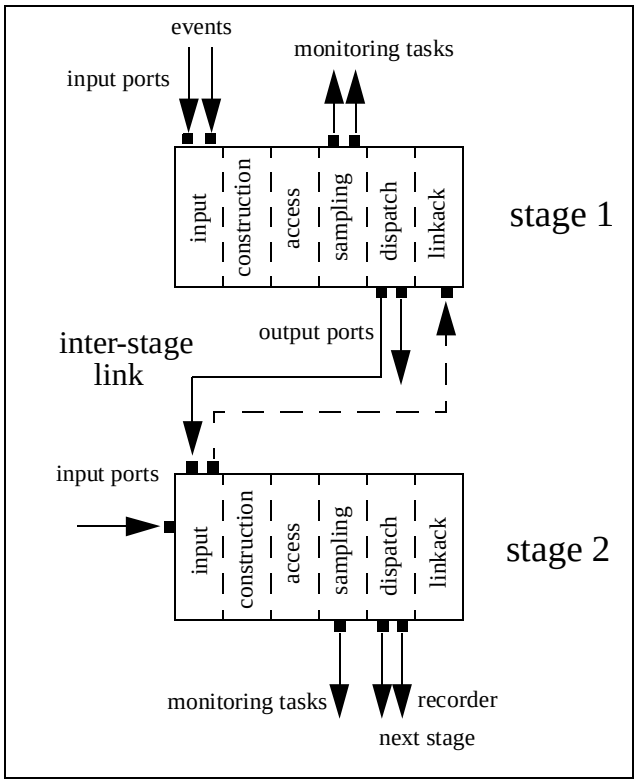

Fig. 3. CASCADE two-stage system

2.2 The Inter-stage Link

Two consecutive stage units in the data-flow topology are linked by the CASCADE connection unit, the inter-stage link. The communication between stages follows a high level protocol involving the dispatch and linkack phases of the upstream stage and the input phase of the downstream stage as illustrated in Fig. 3:

- in the dispatch phase a trigger message is sent to the other stage. It contains a condensed description of the event to be transferred including its type, size and memory address (in case of a memory mapped link)

- the input phase of the downstream stage is scheduled upon arrival of the message. Based on the information in the message, an event descriptor is built, memory space reserved and the event copied across the link into the space just allocated. An acknowledge message is returned.
- the linkack phase of the upstream stage is scheduled when the acknowledge message arrives. If all operations on the event are finished, the event descriptor is removed from the stage and the associated space released. The dispatch phase is scheduled to check if new events should be sent across the link. If there are none, the dispatch phase returns, else the protocol is restarted.

III. TEST SYSTEM SET-UP

The timing measurements described below have been performed using the hardware set-up shown in Fig.4 which allows a direct comparison to be made between the VICbus, FDL and Ethernet links in a CASCADE environment. The two VME crates are equipped with CES FIC8234, MC68040 based VMEbus processor boards [5] running the OS-9 operating system which is widely used for data acquisition at CERN.

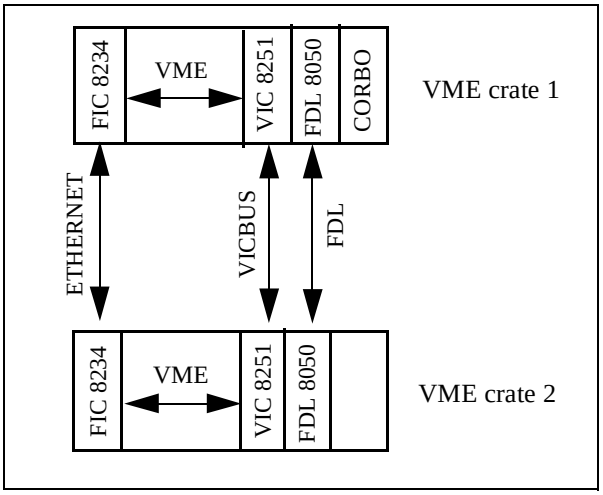

Fig. 4. Test set-up

The VICbus connection is implemented using the VIC8251 VMEbus to VICbus interface from CES [7] which provides a transparent, memory mapped connection between VME crates. The FDL8050 is an interface between VMEbus and the Fast Data Link developed recently by CES [8]. It is an intelligent communications controller and is designed for data transfer rates up to 50 Mbytes/s. The VIC8251 and the FDL8050 will be described in more detail in the following sections.

The 'CORBO' is a VMEbus interrupt module from CES [9]. Finally, the Ethernet link is a direct connection between the FIC8234 CPUs.

IV. VICBUS

The VIC8251 interface enables a VMEbus processor to access memories in other VME crates 'transparently' via its MMU routing logic. The best performance is obtained when data is transferrred from the memory of the VIC8251 ('VIC' memory, 4 or 16 Mbytes) via *direct* VICbus cycles. Remote VMEbus memories can also be accessed via *transparent* VICbus cycles (VME-VME) but with a lower performance due to an additional remote VMEbus cycle per data transfer. The VIC8251 supports all types of VMEbus cycles with the exception of D64.

The VIC8251 is a non-intelligent interface and appears to the programmer as a set of registers and mapping memories and hence, for use in a multitasking operating system, the appropriate driver and library have to be developed. For OS-9, an inter-crate message system was implemented [10] that allows the transmission of messages between tasks running in VMEbus processors in different crates. When a message is transmitted over VICbus, a VMEbus interrupt is generated by the VIC8251 in the destination crate and an OS-9 signal identifying the sending task is produced. This asynchronous mechanism is essential for the implementation of the inter-stage link protocol, as discussed in section II.

Data is transferred from remote crates under the control of the FIC8234 which is equipped with a DMA like device (the Block Mover Accelerator) which is able to move data between external memory and FIC memory at high speed. A data transfer library [11] which hides the details of the transfer mechanisms (e.g alignment) to the user and supports the VMEbus/VSB block transfer modes has been developed.

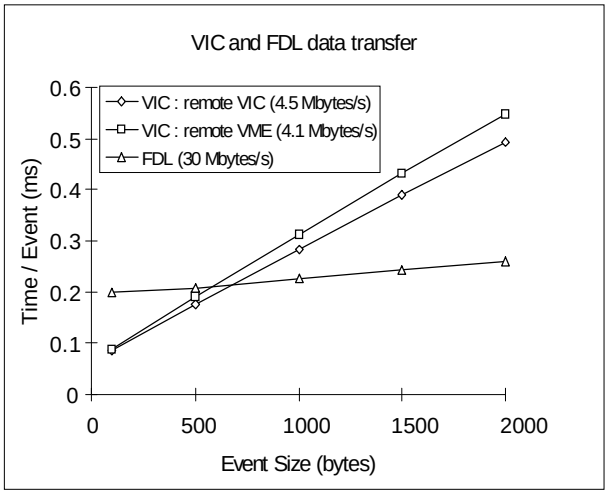

Fig. 5. VICbus and FDL : transfer of small blocks

Fig. 5 shows the results of a simple test program which reads small blocks of data from either the memory of the FIC8234 or VIC memory in crate 1 into the memory of the FIC8234 in crate 2 with optimised transfer parameters : VME/D32 block transfer mode with a block size of 256 bytes under DMA control and read prefetch on VICbus. The intercept is 70 μs and corresponds to the DMA software overhead. Transfer rates are 4.5 Mbyte/s in the case of remote VIC memory and 10% lower, 4.1 Mbyte/s, for the

VMEbus memory. The results for the FDL will be discussed in the next section.

Based on these software tools two test programs which implement the CASCADE inter-stage link protocol have been written. Trigger and acknowledge messages are sent using the VICbus message system and 'event' data transferred from the VIC memory of crate 1 into the memory of the FIC8234 in crate 2 with the same optimised parameters as above. The results are plotted in Fig. 6 which shows the transfer rates in Mbyte/s and, equivalently, the time per event in ms as a function of the event size. The event overhead of about 0.3 ms is mainly due to the exchange of messages. For large events the protocol overhead is insignificant and the data rate approaches 4.5 Mbyte/s.

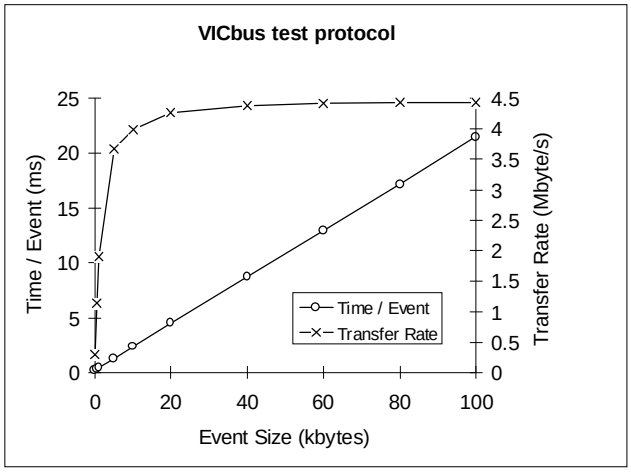

Fig. 6. VICbus : test CASCADE link protocol

V. FAST DATA LINK

The Fast Data Link (FDL) developed by CES is a multimaster/multidrop synchronous link with a bandwidth of 100 Mbyte/s. It is designed to transfer data between multiple sources and destinations at high speed and is deterministic : the arbitration mechanism guarantees maximum latencies for bus mastership attribution and completion of data transfers. When link protocol overheads are subtracted, nominal maximum data transfer rates of about 70 Mbyte/s are quoted which are an order of magnitude higher than VICbus.

The FDL8050 [8] is a VMEbus to FDL interface which is designed to transfer data between VME crates at rates up to 50 Mbyte/s. As opposed to the VIC8251, the board is an intelligent interface equipped with two MIPS R3052 processors, fast data movers, memories etc., in addition to complex communication firmware. A user program interacts with the FDL8050 by sending request blocks specifying the type of data transfer to be performed under control of the interface. This mechanism implies a significant overhead for each data transfer as compared to the VIC8251 which provides transparent access to remote VME crates. An allocation scheme for request blocks allows the control of multiuser access and operating system dependent driver software is therefore not required (except for interrupt handling). The FDL8050 can be used directly from application programs and the manufacturer provides a library and a set of C macros which facilitate the programming.

The FDL8050 has powerful list processing functions which allow collecting and dispatching data to and from different non-contiguous VMEbus areas as well as sparse data scans. However, in the present paper only simple data transfer functions are used to allow for a comparison with the VIC8251.

Based on the software from CES, a small library was developed to provide the link functions required by CASCADE. Routines are available for asynchronous exchange of messages and transfers of data across the FDL from a remote VME crate. To achieve the highest possible speed, the request block parameters for the data transfer were defined as VME/D64 block transfers with a block size of 256 bytes and no flow control on FDL.

A simple test program was written to read data from the VME port of the FIC8234 in crate 1 into the memory of the FIC8234 in crate 2 with optimised parameters as described above. The results are plotted in Fig.5 together with those for the VICbus. The transfer overhead is 200 µs and the transfer speed 30 Mbyte/s which combine to show a 'cross-over' point between VIC and FDL transfers of about 700 bytes.

As in the case of VICbus, two test programs were written to model the CASCADE inter-stage link. In this case data is transferred between the FIC8234 memories under control of the FDL8050s. The results are displayed in Fig. 7 which shows the time per event and the overall transfer rate as a function of the event size. The event overhead is about 0.5 ms which is due to the processing of the requests blocks by the FDL8050 in addition to the OS-9 interrupt overheads. For large events a data transfer rate of about 32 Mbyte/s is achieved. For certain applications it may be required to enable flow control on the FDL in which case the transfer rate decreases to about 20 Mbyte/s.

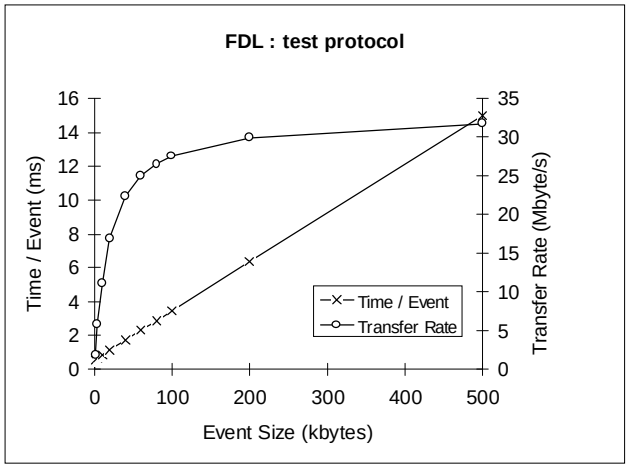

Fig. 7. FDL : Test CASCADE link protocol

VI. ETHERNET

The first inter-stage link in CASCADE was implemented over Ethernet. In this case the underlying software is the

TCP/IP socket library provided with the operating system. A portable higher level library with support for asynchronous data transfers was developed and is available for OS-9, SunOS, Solaris 2, Digital UNIX, HP-UX and LynxOS. For Ethernet, the protocol described in section II is slightly more complicated since a data transfer requires a pair of send/receive calls (peer-to-peer) and an additional acknowledge after the trigger message has been introduced. The results of two test programs simulating the inter-stage link protocol similar to those for VICbus and FDL are shown in Fig. 8. The protocol overhead is 7 ms which is an order of magnitude more than for the VICbus and FDL and the asymptotic transfer rate is about 700 kbytes/s.

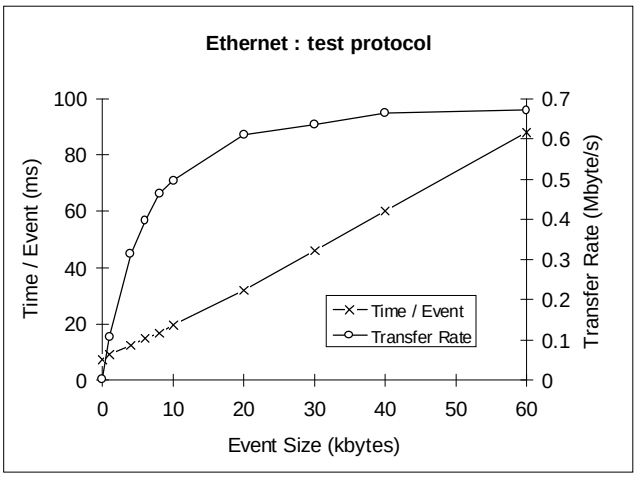

Fig. 8. Ethernet : test CASCADE protocol

VII. INTEGRATION INTO CASCADE

Given the availability of the link software described in the previous sections, the implementation of the CASCADE inter-stage links is relatively straightforward. The libraries for Ethernet, VICbus and FDL are included in the stage which thereby provides 'built-in' support for all three types of inter-stage links. The user selects the type of link in the CASCADE configuration file. To measure the link performances in the CASCADE environment the two-stage system logically defined in Fig.9 and running on the test set-up in Fig.4 was implemented.

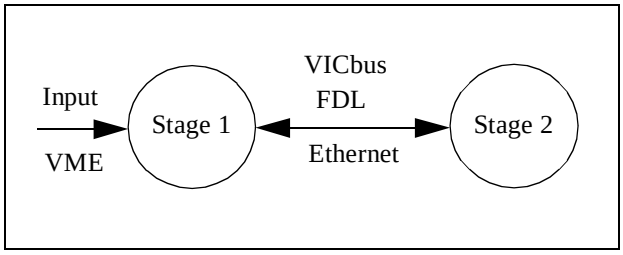

Fig. 9. CASCADE two-stage configuration for test set-up

Stage 1 is triggered at very high speed via VMEbus interrupts and simulated event data is injected into the stage and transmitted to stage 2 via the inter-stage link. The performances in terms of time per event or Mbyte/s are displayed in Fig. 10, below. The event overheads now include the 'internal' CASCADE overheads - e.g due to the buffer management - in addition to those associated with the link protocol. The values are 8 ms in case of Ethernet and about 2 ms for the VICbus and FDL which corresponds to an event throughput of about 500 events/s for the latter two. For large events, the overheads are insignificant and asymptotically the same data transfer rates as obtained with the test programs are observed, namely 700 kbytes/s for Ethernet, 4.5 Mbyte/s for VICbus and 32 Mbyte/s for FDL.

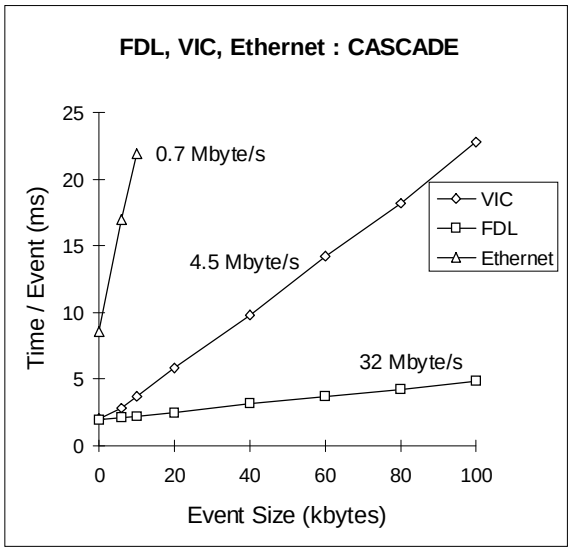

Fig. 10. VICbus, FDL and Ethernet : CASCADE two-stage system

VIII. SCI

Work is currently in progress with the RD24 Collaboration at CERN ("Application of the Scalable Coherent Interface to Data Acquisition at LHC") to introduce SCI as an inter-stage link and, at the same time, to demonstrate event-building. A *parallel Event Builder* is illustrated in Fig.11. Data is produced by n data sources each containing event fragments, numbered sequentially 1,2,..i,.., which are combined into whole events in one of the m data destinations.

A first implementation has been made for stages executing on SUN workstations equipped with SBUS/SCI bridges from Dolphin Interconnect Solutions AS, Oslo, Norway, operating at 125 Mbyte/s. Tests have been made with a 2x2 Event Builder (two Front-End Data Producers, two Event Builders) connected as a single SCI ring, as well as a star configuration using a 4-way SCI switch from Dolphin.

The transmission protocols for this first version have been modelled after the Ethernet networking library, resulting in an SCI Interstage Communication library. Since the present SUNOS SCI driver does not allow asynchronous transfers, an intermediate process is used to wait on read operations and signal arrival of data to the application : a CASCADE stage.

Although the first version is very inefficient because it does not make use of the real SCI capabilities, it is functionally correct, scalable, and can handle millions of events without difficulty.

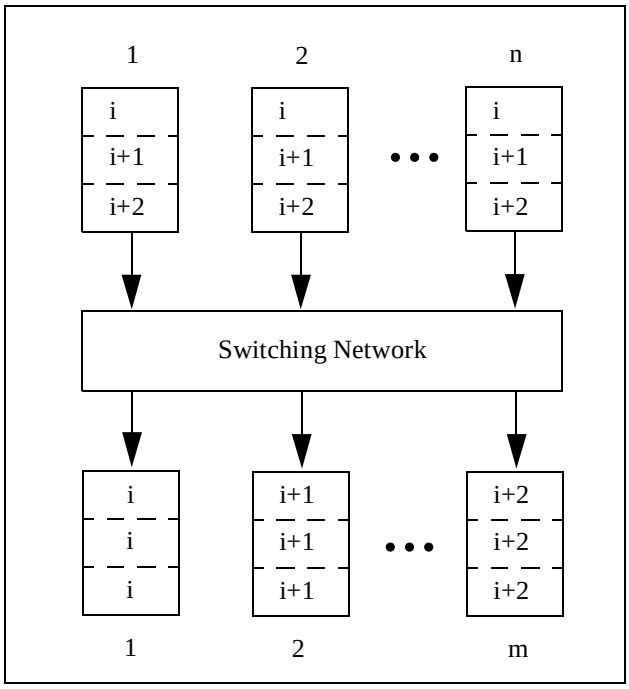

Fig. 11. SCI : parallel event builder

The driver is being improved to allow asynchronous data transfers. In a second phase, the inter-stage communication protocols will be modelled after the VICbus and FDL links to make use of the real SCI shared memory capabilities. Research carried out by RD24 has shown that communication protocol overheads can become very small resulting in an event rate of ~ 50 KHz (“zero length events”) for similar configurations [12].

IX. SUMMARY

The specification of a hardware independent high-level protocol for inter-stage links in CASCADE has allowed the introduction of new types of links without changing the structure of CASCADE. We have described the integration of the FDL, VICbus and Ethernet and measured performances in terms of event overheads and data transfer rates for a two-stage CASCADE system implemented on a VMEbus based test set-up, as shown in Fig. 4.

For the FDL, data transfer rates exceeding 30 Mbyte/s across the inter-stage link have been obtained. If flow control is enabled on the FDL, the rate decreases to 20 Mbyte/s. The event overhead due to the processing of the FDL request blocks is about 200 μs and small compared to the total event overhead in CASCADE (2 ms). The FDL8050 is an intelligent interface with highly complex communication firmware and while this offers certain advantages - complete control of the data transfers and operating system independence - it also makes the module prone to errors. Indeed, during the evaluation work several changes to the firmware were implemented to correct problems (but also to add improvements).

For the VICbus, rates of 4.5 Mbyte/s were measured with the data transfer controlled by the DMA of the FIC8234 with a software overhead of 70 μs. For small data blocks the most efficient method is to let the CPU move the data (memcpy) in which case the VICbus becomes truly transparent. For blocks larger than about one kbyte, the performance of the FDL8050 is superior to that of the VIC8251, asymptotically by a factor of seven. At CERN, VICbus is used in many applications and inter-stage links based on VICbus have worked reliably in the NOMAD experiment for more than a year.

The Ethernet inter-stage link, which is included mainly to provide a reference, has a CASCADE event overhead one order of magnitude higher than that measured for VICbus and FDL with a data transfer rate of about 700 kbytes/s (which is quite adequate in many applications).

Finally, the current status of the work to introduce SCI as a link between CASCADE stages running on SUN workstations in a parallel event building application, is summarised in the previous section.

X. REFERENCES

[1] Y.Perrin et al. , “CASCADE: A Toolkit for the construction of distributed, real-time, data-acquisition systems”, Conference Record of RT’93, Vancouver, Canada, 1993.

[2] CASCADE User’s Guide, CERN ECP/DS, internal notes, http://cascade.cern.ch/Projects/Cascade

[3] D.Burckhart et al., “Review and prospects of the CASCADE Data Acquisition System at CERN”, Conference Record of RT’95, Michigan, U.S., 1995

[4] NOMAD: WA96 Proposal, CERN-SPSC/P261,1991, Geneva, Switzerland.

[5] FIC8234, Dual 68040 Fast Intelligent Controller, User’s Manual, Creative Electronic Systems,

[6] C.Parkman, “VICbus : VME Inter-Crate bus, A Versatile Cable Bus”, Conference Record of RT‘91, Julich, Germany, 1991 (ISO/IEC 11458: 1993)

[7] VIC8251, VIC to VME interface with Mirrored Memory, User’s Manual, Creative Electronic Systems.

[8] FDL8050, Fast Data Link, User’s Manual, Creative Electronic Systems.

[9] RCB 8047, CORBO, VME read-out control board, User’s manual, Creative Electronic Systems.

[10] F.Meijers, J.Petersen, A VME intercrate message system based on VICbus and OS-9, ECP-DS note 93-23

[11] C. van der Vlugt, CERN specification for Standard DMA controller Routines, version 2.53, CERN, 1993

[12] RD24 collaboration : “RD24 Status report” CERN/LHCC 95-42 LCRB , RD24 Status Report/RD24, CERN, Geneva, Switzerland, Aug. 95