A test system for second level trigger and data acquisition architectures

Green, B., Strong, J.
Royal Holloway, University of London, Egham, U.K

Anguelov, T., McLaren, R.
ECP Division, CERN, Geneva, Switzerland

Denes, E.
KFKI, Budapest, Hungary

The system provides data to second level trigger or data acquisition architectures at a rate and of a complexity that allows a realistic test of the performance of the architecture. To meet the data rates needed and the flexibility to cope with different detectors, trigger architectures and links between them, the design of the hardware is kept simple and the data generation and formatting is performed in software. Currently, the system is being used to test an event distribution architecture and will shortly be used to test an event building system and second level trigger architectures.

Introduction

Future hadron collider experiments present severe problems in moving and storing data at all levels of the trigger and data acquisition chain and it is essential that these aspects of the experiment are tested at an early stage, well before the start of collider operation or the availability of detector modules. Hardware description languages can be used to provide models of proposed systems and to simulate system performance, but these models, if they are sufficiently detailed, run at many orders of magnitude slower than reality and it must be shown that they accurately simulate the performance of the hardware. Information gained using a hardware test system can be used to refine the model; it can test the architecture at realistic rates and can be used for debugging the architecture during the lifetime of the experiment. Discussions within the RD-11 [1] collaboration led to the design presented here of the SLATE (Second Level Architecture Test Environment) system.

Basic Design

The task of the SLATE system is to provide data to second level trigger and data acquisition architectures under test at a rate and of a complexity that allow a realistic appraisal of the performance of the architecture. Assuming commercially supported link designs (e.g. HiPPI [2], SCI) would be used we have designed a system capable of sustaining a rate of about 1Gbit/s on a single link and which can be scaled to any number of outputs. Because of an existing interest, the prototype board has been designed with a HiPPI interface [3].

To meet the data rates needed and the flexibility to cope with different detectors, different trigger architectures and different links between them, we have kept the design of the hardware.
simple and moved all test specific tasks to the software. Basically, the SLATE hardware is a data module containing a memory into which pre-formatted data can be loaded and a link from that memory to the architecture under test. Data can be loaded into the memory slowly and then read out at the rate needed for the test. The prototype described here assumes a data-push protocol and allowances for destination response must be incorporated into the software. Later versions will include (maskable) flow-control. Although the software [4] is of prime importance for the proper running of the system, only the hardware will described in this paper.

Description

The system can be used to provide data for a complete range of tests from basic link protocol tests to full experimental data. Fig. 1 shows a system for providing an architecture with test data from a full simulation of an experiment. Physics generators and detector simulations are used to produce standard data files. From these, further files are constructed containing data from regions of interest to the second level architecture being exercised and relevant data from the first level trigger. The data for a set of events are transferred to a workstation and formatted into a form suitable for the SLATE data module. Information from the user on data rates, link protocols and other parameters are added at this stage. A Unix based workstation running X-windows is used with an Ethernet connection between the workstation and the VME crate which houses the SLATE data modules. Other platforms could be used without significant software modifications. VME operations are supervised by a resident CPU running OS9 with programs written in C. As data modules use only the P1/J1 backplane connector and function only as VME slaves, they can work with any VME master.

Fig. 1 SLATE SYSTEM

Fig. 2 SLATE DATA MODULE

The basic data module is constructed in two parts; a mother-board which contains the memory section and a daughter-board which contains the link electronics. All mother-boards are identical; the daughter-boards are those required for the links to the architecture under test and may be different for different areas of the architecture. For the example shown in Fig. 1
one module would store trigger information and others the corresponding event data; the links to the architecture could be different for the two data types.

The principle elements of the mother-board are the control and event memories. Fig. 2 shows only the interconnections for data readout; the VME interface and the bus for loading and controlling the operation of the module are not shown. The event memory is 48-bits wide and is loaded with event data and link protocol information; it is constructed from 64k by 4-bit 25ns memories. Data for each event are loaded into sequential locations in the event memory starting from a known location and with the final data word flagged by an end-of-event flag in bit 47; arbitrary gaps can be left between events stored in the event memory. The control memory stores a list of start addresses and the delay time to the start of the next event for each start address entry. The control memory uses four 64k by 4-bit memories to store event start information and four similar memories to hold the delay times between events. The address lines for the start address and delay times are connected in common. One 25Mhz clock is used to drive all parts of the SLATE board including the daughter-board. Status and Control registers on the mother-board with connections to the daughter-board are used to set the mode of operation of the module (e.g. Load, Stop, Go, cycle etc.) and monitor its status (e.g. Reset, Stopped, Waiting for Go etc.).

On receipt of a start signal, the start address currently output from the control memory is loaded into the event memory counter and the delay value into the delay time counter. Data are read from sequential locations in the event memory by incrementing the address counter with the on-board clock until an end-of-event flag is detected when the address counter is disabled. The event memory information is passed to the daughter-board and translated to the format and timing sequences required by the link. Using the on-board clock, the delay counter counts to full scale and then increments the control memory address and this initiates a new cycle using the next start address and delay value. If the delay follow-on flag is set (bit 15 of the delay value) a new delay value is loaded and the delay cycle initiated but no data are read from the event memory. In this way, delays longer than 1.3ms can be programmed into the system.

For basic link or protocol tests, the event memory can be loaded with test patterns (containing, where desired, protocol errors). For architecture tests, fully simulated event data can be used. The control memory can be set for a single pass through its start address set, starting from a given point to the end of the control memory, or to cycle continuously through the set from the chosen point to the end. The set of start addresses can contain any number (up to 64k) of the stored events accessed in any order with any delay between them. This allows a wide variety of data cycles including quasi-random event ordering. The time to read straight through the event memory is 2.6ms. For events of 1kByte selected randomly and with no delay between them, one cycle through the control memory uses each event an average of 256 times and increases the read time to 670ms. This expands to over 80s if all delay times are set to maximum.

Current status and uses

After its completion in late 1991, the prototype board was transferred to CERN for testing. These tests showed that the system can run at the full HiPPI transfer rate of 100MBytes/s. Since then the unit has remained at CERN and has been incorporated into tests on a number of projects. Use of a HiPPI switch purchased by the L3 collaboration has allowed tests on the different projects to proceed with changes only to the software configuration; no changes are necessary to the hardware connections (Fig. 3).
Under RD-11, a HiPPI to MaxBus (HiMax) unit has been built which will allow transfer of data via the SLATE/HiPPI board to a MaxVideo [5],[6] system for testing this architecture as a candidate second level trigger system. When a prototype ENABLE [7] machine becomes available, a SLATE/HiPPI board will be used to provide data to test its performance. For RD-13 [8], the test of a switch as an event builder is under way using the HiPPI switch. The SLATE/HiPPI board is used as one of the data sources.

For NA48, the switch is being tested as means of distributing events to a workstation farm. The SLATE/HiPPI board acts as the event source with the I-field of the data block used to specify the address of the destination (Fig. 4). In this test three 16kByte blocks of data are stored in the SLATE board and then distributed to the destinations stored in the block headers. Tests show that transfer rates of at least 90MBytes/s can be achieved [9].

**Future developments**

For the next stage of the mother-board development we intend to add interrupt registers and the ability to switch to an external clock and stop/start control. Flow control will be added so that the system can remain synchronous and operate without loss of data should the buffers in the architecture under test become saturated. External triggering of a single event transfer will be added. To allow a multi-board system to operate synchronously, we will add a front panel connector which will be used to bus signals between boards on small systems and to connect to a central control unit for larger systems.

**References**


704