

UK Rob-in performance measurements

Last update: 27-Oct-1999

Participants

Royal Holloway University of London: B.Green, J.Strong

University College London: R.Cranfield, G.Crone

Introduction

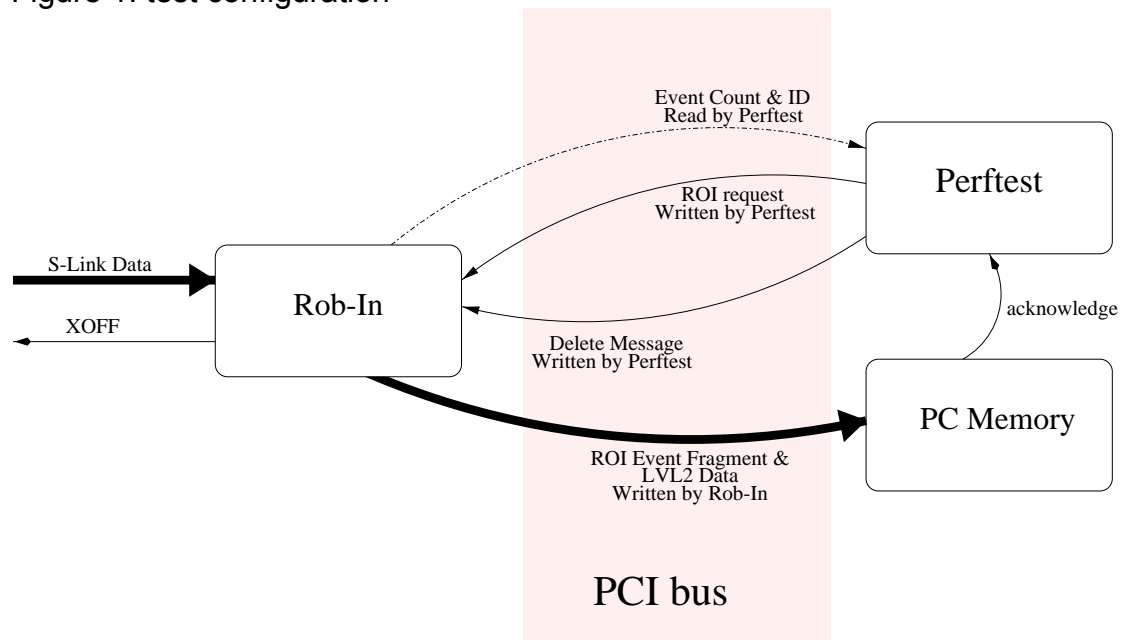
The performance of the UK ROB-in [1] module has been measured in a test configuration designed to emulate the inputs from and outputs to the real system environment. The general approach was to determine the maximum input event rate that could be sustained for particular combinations of key parameter values. This is one of the alternatives specified for comparative benchmarking of ROB prototypes in the ATLAS Level-2 Pilot Project [2], and the parameter values used were a superset of those specified for this benchmarking. Although the event fragments contained dummy data and the output fragments were sent to a sink, the ROB-in itself performed all its normal operations.

Pilot Project benchmark values

- Grouping factor for decisions: 100
- LVL1 event rate
- Event fragment sizes: 256, 512, 1024, 2048 and 4096 bytes
- Percentage of the events for which a LVL2 accept is received: 1 %
- ROI request percentages: 5, 10, 14.3 or 20 %

Test setup description

Figure 1: test configuration



Overview

The ROB-in was plugged into a PCI slot on a PC which was programmed to emulate the Level-2/DAQ environment, whilst front-end data was generated in a SLIDAS [3] data source plugged on to the ROB-in.

The PC was used to generate both RoI requests and Event Builder requests (Level-2 accepts). Since from the ROB-in point of view these two types of request look basically the same (with the only potential difference being their destination addresses) the PC was programmed to simply generate event-fragment requests at a frequency equal to the sum of the required Level-2 accept and RoI request frequencies. The PC was also programmed to generate event-fragment release requests in synchronism with the incoming event fragments to keep the buffer from overflowing.

The input event rate itself was automatically controlled by an XOFF mechanism between the ROB-in buffer and the SLIDAS data source so that the system would naturally run at the maximum event rate possible, up to the limit of the SLIDAS, for the given parameter values.

For a realistic measurement of the ROB-in performance requested event fragments were sent out to a real destination on the PCI bus, which was actually the PC main memory. Fragments were indexed and accessed properly according to their event Ids, though the Ids themselves were generated internally by the ROB-in.

Data source

The SLIDAS is a PMC-sized S-Link [4] data source capable of generating ATLAS-type event fragments. It was mounted as a daughter card on the ROB-in's S-Link interface and set to a clock frequency of 40 MHz. This clock frequency allows the SLIDAS to produce data at up to 160 Mbytes/s, with the actual rate being regulated by flow control as mentioned above. The module used was a SLIDAS 3.0 which has been designed to produce Rob format data at switch-selectable event sizes corresponding to the Pilot Project test benchmark specification. The actual event sizes used were a superset of the Pilot Project benchmark values.

ROB-in

The UK ROB-in exists in two versions: as a PMC daughter card and as a card that plugs into a normal PC-style PCI slot. The latter version was used in these tests, but the two versions are electrically equivalent. The ROB-in features paged memory managed by both hardware (a MACH5 CPLD [5]) and software. Controlling software, the *Buffer Manager*, normally runs on an on-board i960-Rx processor [6], but can also be run off-board, accessing the ROB-in memory and registers across PCI. Front-end (event fragment) input is via S-Link, through an S-Link connector on the board. When the on-board *Buffer Manager* is running it receives fragment requests across PCI, via the bridge built into the i960. Requested event fragments are output by DMA to PCI addresses across the same bridge. The availability of the buffer memory is reflected in the number of free-page addresses in the “free-page fifo” which automatically generates XOFF on the S-Link input link whenever the number of free-pages drops below a programmable threshold. XOFF will also be generated if the *input* fifo fills beyond another programmable threshold.

PC host

The host PC was a AMD K6-2 350 MHz processor with VIA PCI chipset and 64 MB PC100 memory.

Software

The PC test software runs under Linux. It consists of a simple Linux driver and a user-level program (*perftest*). The driver maps ROB-in PCI memory space into user memory space, thereafter allowing control and communication from the user-level program which communicates with the *Buffer Manager* on the ROB-in to control the latter's operation.

The *perftest* program sends two kinds of message to the ROB-in: fragment requests and fragment delete messages. For these tests the percentages of ROI requests were considered to be 0% 5% 10% 14% and 20%. Adding a fixed 1% of LVL2 accept (Event Builder) requests to these values gives fragment request percentages of 1%, 5%, 11%, 15%, and 21%, which were the actual values used. The delete messages are lists of Ids of event fragments to be released as a group; for this test a group size of 100 Ids was used.

In the real system fragment requests would be automatically synchronised with fragment data via the Level-1 trigger, the Level-2 RoI builder, and the DAQ Event Builder. In our test environment this synchronisation is emulated by the *perfest* program monitoring the number of events that have been indexed by the *Buffer Manager*. The buffer manager maintains a counter of the number of events indexed which is visible across the PCI bus. Since event IDs are assigned automatically in sequence by the i960, this counter also gives the latest event ID. *Perfest* monitors the counter and, when it sees that 100 events have been indexed, it requests the required percentage of them. The event request protocol is *blocking* i.e. a new request is not sent until the previously requested transfer has completed. Once all the requested transfers have completed *perfest* sends a delete message for those 100 events.

Test procedure

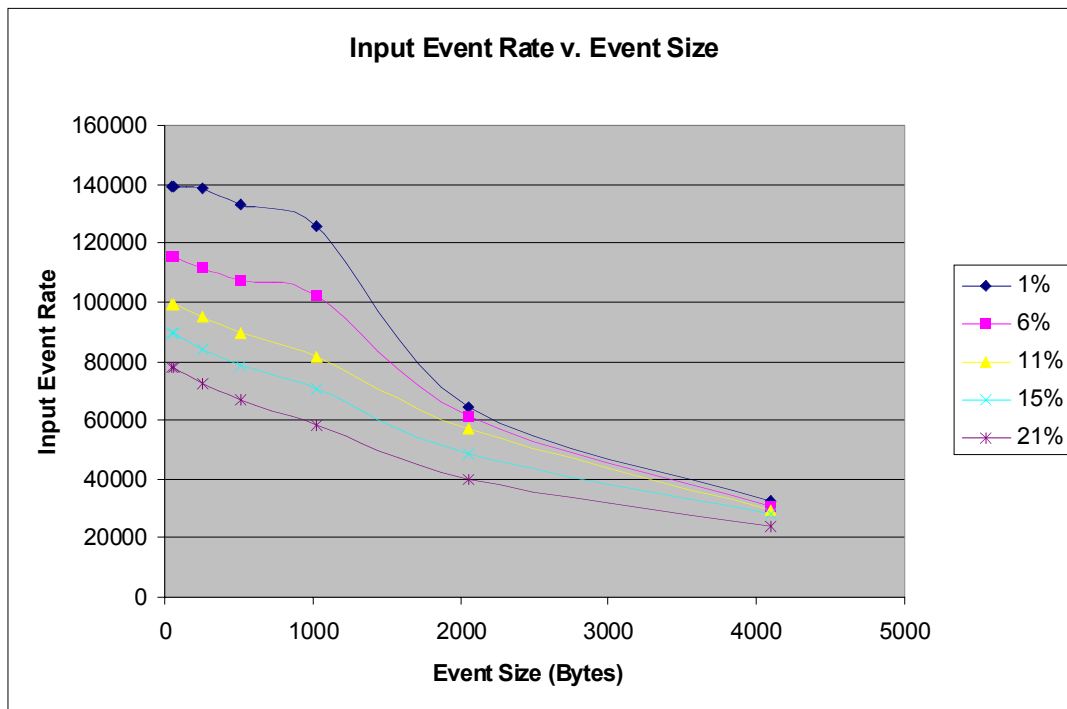
The *perfest* program is run with the percentage of requests as a parameter. When the program is terminated it prints out the number of events indexed.

Each measurement was made for about 5 seconds with the Linux clock being used to measure the exact duration of the test. One of the measurements for each event size was repeated for 100 seconds as a check on the 5-second measurements. For the 1%-request measurement the test had to be run for 30 seconds to get stable results. The results were recorded and plotted in an Excel spreadsheet.

Results

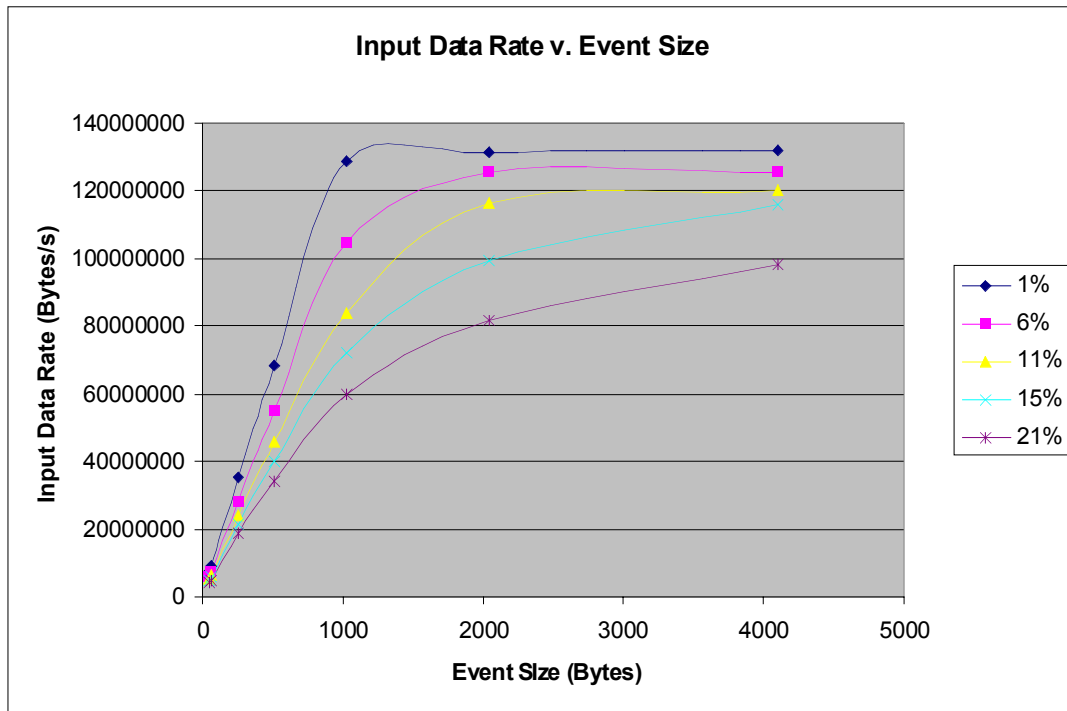
The basic measurement results are shown in Figure 2, as the maximum event rate (in fragments per second) against event (fragment) size in bytes for different percentages of fragments requested for output. It can be seen that for the canonical event size of 1 Kbyte the original design specification for the ROB-in is met (100 KHz event rate for 5% RoI requests). At smaller and larger request percentages the event rate is correspondingly higher or lower. The individual plots below are consistent with discontinuities existing at event sizes that are multiples of 1 Kbyte. This is to be expected since the ROB-in page size is 1 Kbyte and fragments larger than this will therefore occupy more than one page, incurring the extra overhead associated with the indexing of multi-page fragments.

Figure 2: Input Event Rate v. Event Size



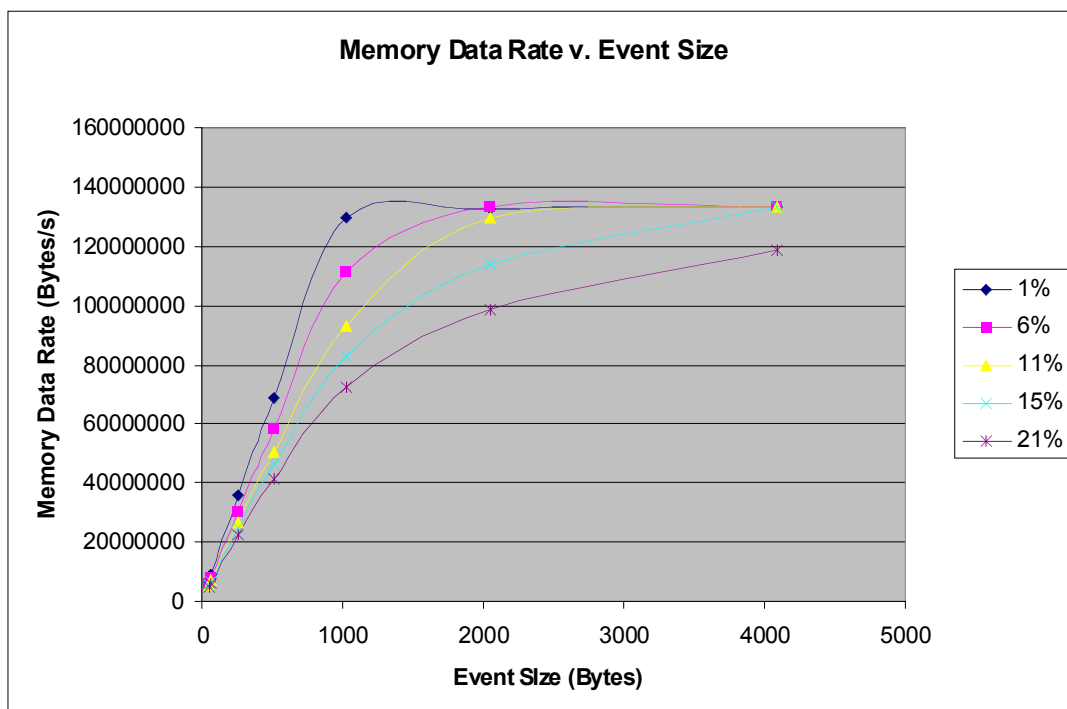
The data from Figure 2 are re-plotted in Figure 3, expressing the results in terms of input data rate rather than input event rate.

Figure 3: Input Data Rate v. Event Size



The plot in Figure 4, which is again derivable from the Figure 2 data, shows the total data rate into and out of the buffer memory. This is the sum of the ROB-in's input and output data rates. The plot illustrates how, with large enough fragment sizes, the system is limited by the memory bandwidth (132 MB/s), whilst at smaller fragment sizes the bottleneck is elsewhere (*Buffer Manager* software).

Figure 4: Memory Data Rate v. Event Size



Plots for *output* rates are not shown since they are determined entirely by the request percentages and are simply those same fractions of the input rates. Since the theoretical PCI bandwidth for output is the same as the buffer memory bandwidth it is clear that the PCI will not be saturated at the realistic request rates (percentages much less than 100) used in the test and that this will therefore not be the limiting factor.

References

- [1] UK ROB-in: <http://www.hep.ucl.ac.uk/atlas/rob-in>
- [2] Pilot Project: <http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/L2PILOT/l2pilot.html>
- [3] SLIDAS: <http://www.cern.ch/HSI/s-link/devices/slidas/>
- [4] S-Link: <http://www.cern.ch/HSI/s-link>
- [5] MACH5: <http://www.latticesemi.com/products/devices/mach5.html>
- [6] i960-Rx: <http://developer.intel.com/design/iio/datashts/273001.htm>