Proposal for an Hardware Event Sorter for the IMAGER (ESI)

E.Kendziorra, M.Mohan and R.Staubert

Introduction

The current IMAGER proposal assumes that there is a scientific processing unit (SPU) between the module control electronics (MCE) of the detectors and the DPE. The DPE will be provided by ESA.

The SPU had to perform the following functions:

- process scientific data,
- handle hardware interfaces between module control electronics and the DPE.

Given financial and manpower problems, alternative solutions to replace the SPU must be considered.

Basically, there are two possible solutions to this problem which have already been discussed during the 1/2 August 1995 meeting:

a) The DPE performs all functions of the SPU, including handling of all hardware interfaces.
b) The SPU is replaced by some simple hardware. The DPE performs all S/W tasks.

In this note we discuss the two options and make a proposal for a technically feasible solution.

Solution a)

This option does not seem to be feasible without a major redistribution of tasks within the IMAGER.
The current IMAGER proposal foresees separate data interfaces between the 2 x 8 module electronics and the SPU/DPE, but the DPE provides only one fast data interface.

In our technical note on the "Scientific Data Processing Tasks for the Integral IMAGER", IN-IM-TUB-TN/EL-002, we have shown, that the DPE **can not** handle all the scientific data processing needs. In particular the DPE is not able to detect coincidences between CsI and CdTe events unless **all** CdTe events as well as **all** CsI events are already sorted by time.

Proposal (solution b):

Replace current SPU by a "simple" hardware which is able to:

- collect event data from all MCEs
- sort all CdTe events by time

- sort all CsI events by time
- distribute time clock and time synch to all MCEs.

  and optionally:
- *collect HK data from all MCEs*
- *distribute commands to all MCEs*

This "Hardware Event Sorter" (ESI) could be built on 1 board (>≈100 x 160mm^2), and would consume 2-3 Watts @ 5V (TBC).

This new concept would require only three extra capabilities from the DPE as currently designed:

provide 1 extra slot and connectors
provide 2-3 Watt @5V
provide two instead of one fast serial interface.

Figure 1 shows a block diagram of the ESI. The two event sorters would be identical ASICSs, provided one could agree on a common structure for CsI and CdTe event data. The command and housekeeping handler circuit could be adapted from an existing design for the EPIC pn-CCD control electronics.

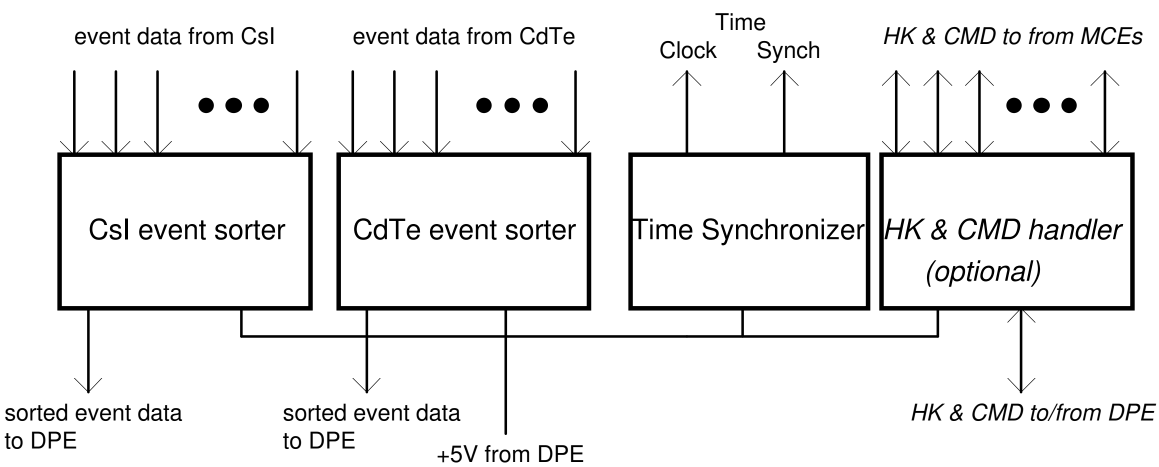

Figure 1: Block diagram of ESI

Figure 2 shows a functional block diagram of the event sorter together with the data interfaces to the MCEs. In the MCEs event data are stored in FIFOs and read out to the ESI through synchronous serial interfaces with three RS422 lines each (data, clock and word synch.). At the input of ESI the data are converted to 16 bit parallel words, which are put on a multiplexer (MUX).
The time sorter finds the first event from the 8 inputs and then writes all data from this event into a RAM, which is organized as a FIFO. Having written all data from one synch period in time sequential order into the RAM, the RAM is dumped to the DPE through a serial interface. The ESI also generates and distributes the time clock and time synch pulses to the MCEs.

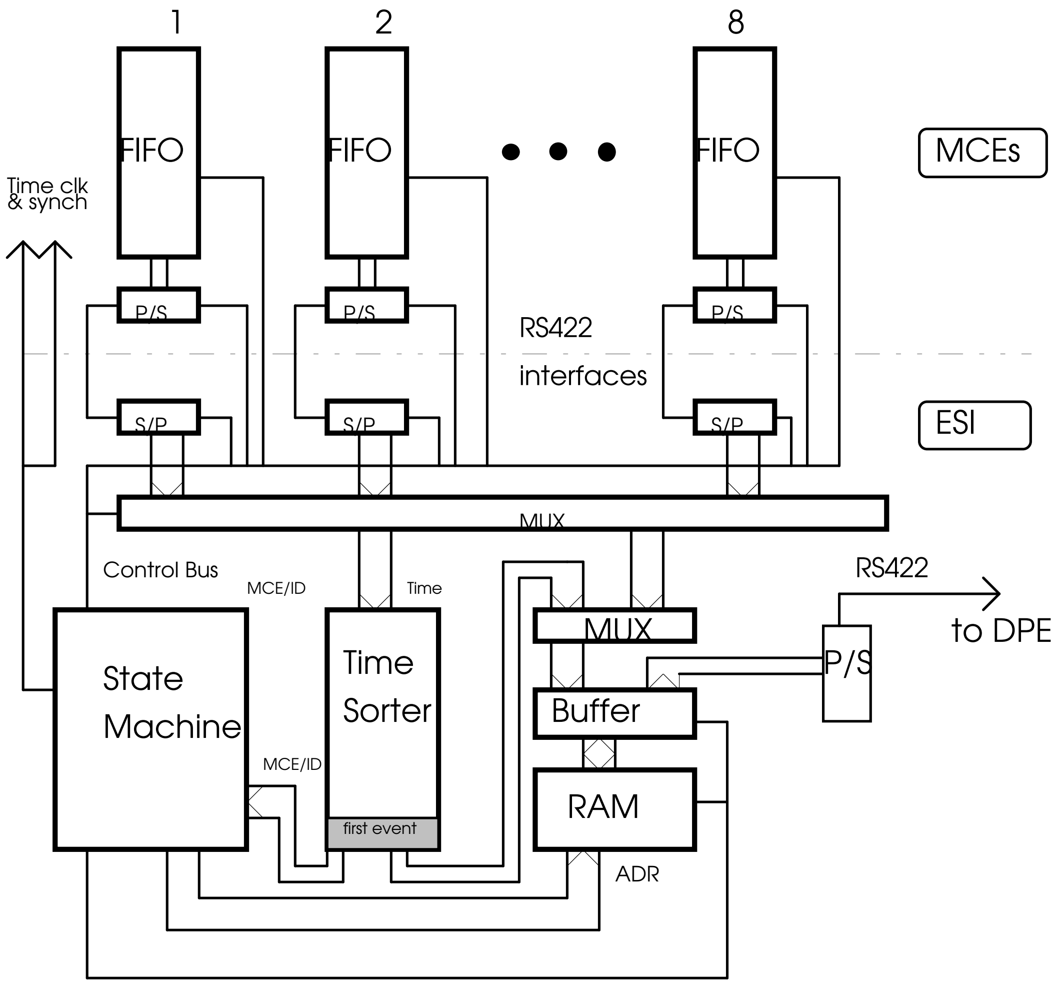

Figure 2: Functional block diagram of the event sorter

The purpose of the event sorter is to read the times from each of the 8 parallel FIFOs and write the event with the lowest time to an output RAM (FIFO).

The concept of data flow in the ESI is shown in Figure 3. The numbers in brackets refer to the sequence in which data are written to the output buffer (RAM).
First the event times from the 8 MCEs are compared and the first time (lowest value) is found by the time sorter unit. This time is written to RAM (1), and then the remaining data from that event are added (2). The module identifier (MId), which is part of the time word, is used as pointer to the MCE. Now the next time word from the MCE, in which the last event was found, is read to the time sorter (MCE No. 2 in our example). Again the earliest event time from this new sample is found and the event data are written to RAM (3) & (4). This loop continues until data from a new synch period have been detected at the outputs of all MCEs. Now the RAM is dumped to DPE.

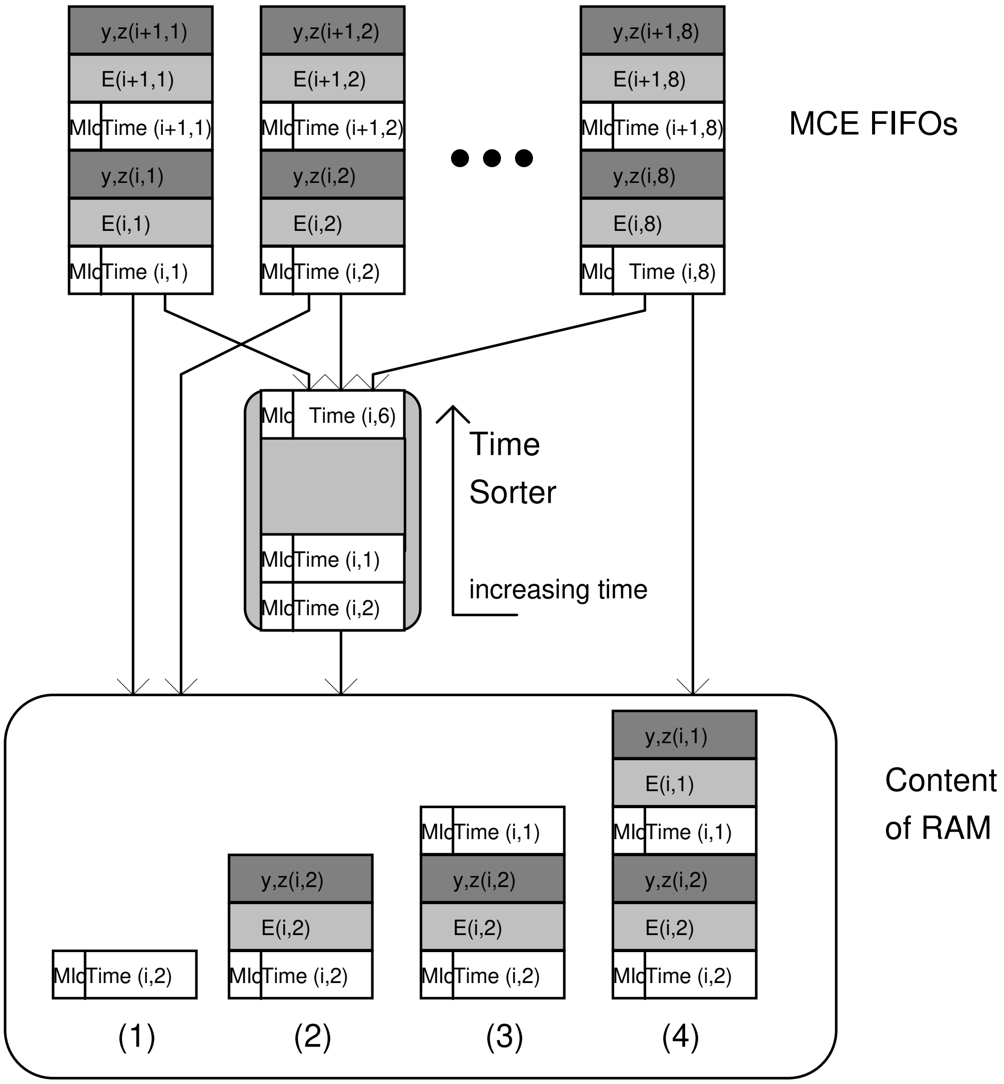

Figure 3: Flow of event data within event sorter