The EUDET Data Reduction Board (EUDRB)

(firmware version EUDRB_MIMOx)

version 1.0 of Apr. 22nd 2008
version 1.1 of Aug. 8th 2008

A. Cotta Ramusino[1]

Abstract

The EUDET Data Reduction board was developed at INFN-Ferrara in collaboration with University of Insubria-Como and INFN-Roma 3 to read out Monolithic Active Pixel Sensors (MAPS). The motherboard ("EUDRB_MoBo") is a VME64x slave in 6U Eurocard format. The motherboard supports one "analog" ("EUDRB_DCA") and one "digital" ("EUDRB_DCD") daughter cards with PCI Mezzanine Card (PMC) format. The EUDRB_DCD provides detector timing signals and it is also provides the EUDRB with a USB2.0 port for diagnostic and stand-alone data acquisition. The EUDRB-DCA has 4 single-ended/differential analog inputs and it is based on the design developed by IPHC (Institut Pluridisciplinaire Hubert-Curien) and the SUCIMA collaboration. The EUDRB has been so far employed with the IPHC MIMOSA-5, MIMOSTAR 2 and MIMOTel devices.
The "EUDRB_MIOx" version of the firmware extends the functionality of the EUDRB card to include handling for the MIMOSA18 sensor and improves its rate capabilty

[1] INFN,Ferrara, Italy

1 Introduction

The EUDET Data Reduction board (EUDRB) was developed at INFN-Ferrara, in collaboration with University of Insubria in Como and the INFN - Roma3, to read out Monolithic Active Pixel Sensors (MAPS).

The board generates detector timing signals in LVDS logic levels and provides analog to digital conversion of the 4 analog single-ended/differential input with 12bits resolution at a sampling frequency currently selectable between 15MHz and 11.25MHz. The EUDRB also features an output port (with TTL signalling levels) to configure detectors with JTAG programmable features. The motherboard has 4 banks of 256k x 48bit SRAM memories providing storage for up to 3 full frames for a 1Mpixel sensor like the MIMOSA-5 by IPHC. On this collected data the EUDRB performs Correlated Double Sampling (CDS), Pedestal Subtraction and Threshold Comparison on to reduce the data size for the events selected by the experimental trigger. The EUDRB may however provide the full frame information when the "raw" operating mode is selected. The EUDRB features a 256k x 32bit FIFO memory to provide temporary storage for data selected by a trigger and waiting to be readout. Both a VME and a USB2.0 interface are implemented; the EUDRB may be used for reading out MAPS both in stand-alone mode on a bench-top and in a standard VME-based data acquisition system.

The trigger input port on the front panel of the EUDRB was designed to interface it to the EUDET Trigger Logic Unit (TLU) and a trigger bus is also foreseen to distribute/receive trigger information over the free lines of the VME J2 connectors by means of a private bus on falt cable. Two LEMO connectors on the front panel of the EUDRB can be used to synchronize the data collection operations over a pool of boards.

The operation of the EUDRB is controlled by an ALTERA Cyclone II Field Programmable Gate Array (FPGA) device in which a NIOS-II microcontroller is also implemented to do initialization, housekeeping and diagnostic. An 8 MByte serial configuration device EPCS64 is used to store both the FPGA configuration information and the microcontroller code. At power up the FPGA loads its configuration from the EPCS64 and begins operation; the first task executed by the NIOS-II is to copy its ROM-resident code from the EPCS64 to its 1MByte program/data RAM and then jump to executing the code in RAM.

The FPGA handles the operations related to data collection, trigger servicing and I/O port interfacing with sequencers and logic blocks described in VHDL code or schematic diagrams. The FPGA project files are processed by the ALTERA Quartus development system to produce a configuration file (with a **.sof** extension). The development of the NIOS-II firmware is instead carried out using the ALTERA NIOS-IDE, an Eclipse-based framework; the compilation of the C-source code produces a loadable file (with a **.elf** extension). The "FLASH PROGRAMMER" tool of the NIOS-IDE is used to store both the .sof and the .elf files in the EPCS64 configuration device via the JTAG port of the FPGA.

The present description of the EUDRB refers then to a EUDRB loaded with the "EUDRB-MIMOx" combination of the TopLevel_MIMOx.sof + EUDRB_MIMOX.elf file developed by the writer for the commissioning of the demonstrator telescope at DESY and CERN.

The URL for the compressed file containing the whole project is (work on it still in progress):

http://www.fe.infn.it/u/cotta/ILC/EUDET/EUDRB-MIMOx/TopLevel_MIMOx_temp.rar

2 EUDRB hardware: an overview

The figure on the following page represents a block diagram of the EUDRB.

2.1 EUDRB Motherboard (EUDRB_MOBO)

The larger green block represents the motherboard (EUDRB_MoBo) and the resources supported by it:

- the ALTERA EP2C70F896C8 FPGA (dashed blue outline) with the NIOS-II block in evidence. Details of the sequencers and logic blocks implemented in the FPGA are given in a later section
- the four banks of 256k x 48bit SRAM whose function is to hold the pixel voltage samples recorded during the last three scans of the four MAPS submatrices. The SRAM also hold values of pedestal (6bits) and threshold (6bits) which are specific to each pixel
- the 256k x 32bit FIFO used as temporary storage for the data requested with a trigger pending the readout through the VME or USB2.0 port
- utilities:
  - the 256k * 32bit SRAM used as program/data memory by the NIOS-II
  - the 1M * 8bit Flash (non volatile) memory which could be used by the NIOS to store permanent data (i.e. default pedestal/threshold values)
  - the configuration device EPCS64
  - the configuration controller based on an ALTERA EPM240T100 which provides an alternate method for bootstrapping the FPGA
  - the “EUDET TLU” ports: the front panel port to connect the EUDRB directly to the EUDET Trigger Logic Unit. An EUDRB connected to the TLU will act as the “TLU-Interface” for all the EUDRBs in a VME crate. The “TLU-Interface” fans out the trigger informations to other EUDRBs via a private bus on a cable segment installed on the free pins of the VME P2 connectors
  - the RS-232 port which allows a simple and direct connection of the NIOS-II to an host PC for lower level diagnostics and debugging

2.2 EUDRB Digital Daughter Card (EUDRB_DCD)

The EUDRB_DCD is a board with standard PCI Mezzanine Card (PMC) format and PMC compatible connectors toward the EUDRB_MOBO.

On the front side of the EUDRB_DCD four connectors are available, whose pin definition is reported in Fig. 2.3.

The EUDRB_DCD features a Cypress CY7C68013A-56PVXC, which, like in the MAPS readout boards designed by the SUCIMA collaborations, provides a relatively simple way of interfacing the EUDRB’s FPGA to an USB2.0 bus, via the front panel connector J4.

The CY7C68013A-56PVXC holds its bootstrapping data into a 24LC128 128kbit serial EEPROM. The EUDRB_DCD then provides an I/O port with single ended 3.3V TTL signal levels (connector J1, RJ45) to control the configuration of sensors with features programmable via a JTAG interface, like the IPHC MIMO*2.

The J2 connector provides detector timing signals (Scan Clock and Scan Reset) in LVDS format.

For driving the MIMOTel sensor used in the demonstrator telescope an interface board has been developed at INFN Ferrara (“MIMOTel level adapter”) which translates the voltage levels of the JTAG signals from the EUDRB_DCD J1 connector, rearranges connector pin

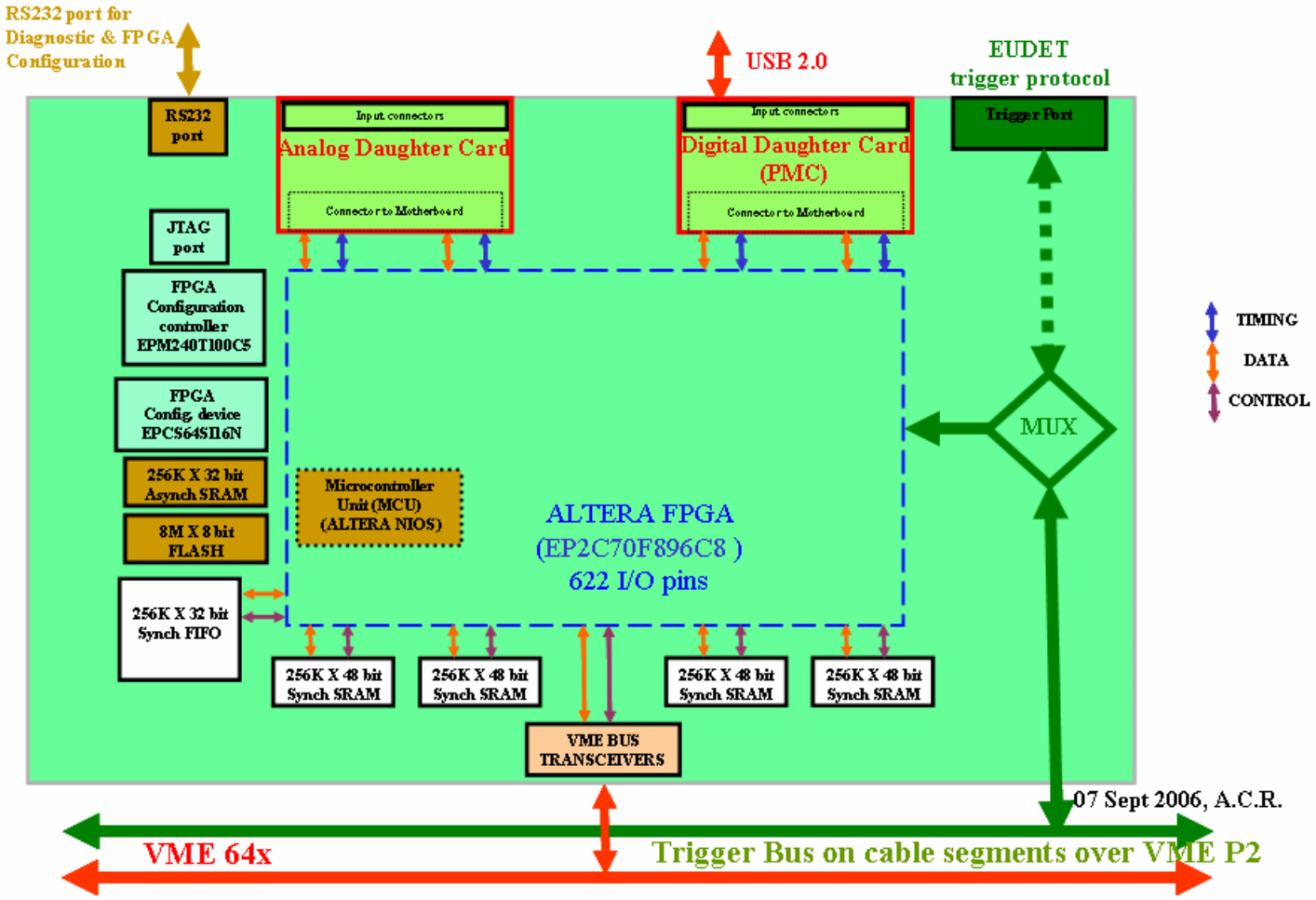


Fig. 2.1 EUDRB ‘s block diagram

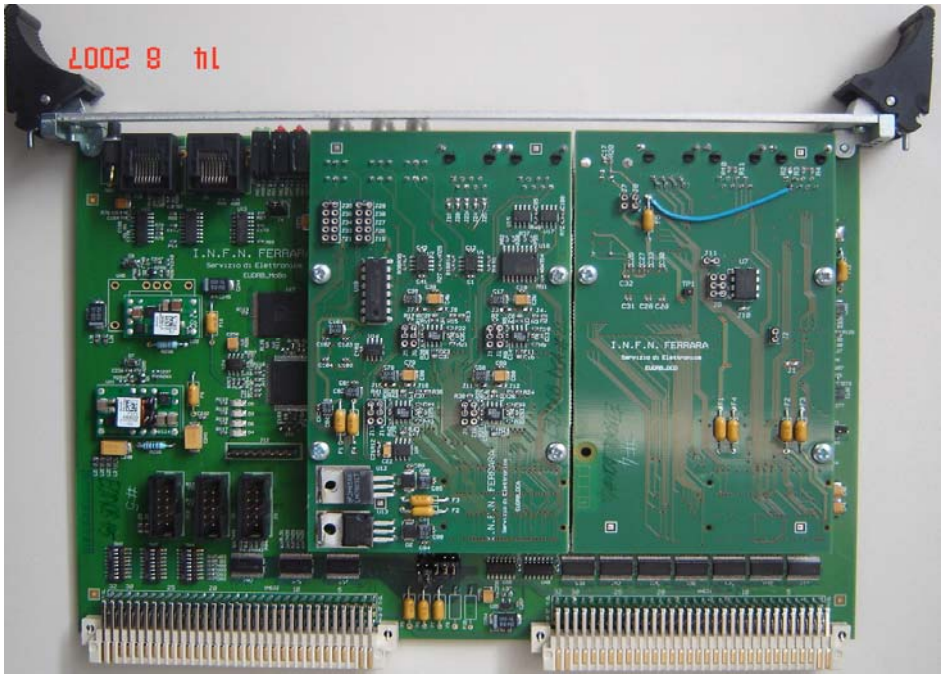


Fig. 2.2 EUDRB’s top view with daughter cards installed

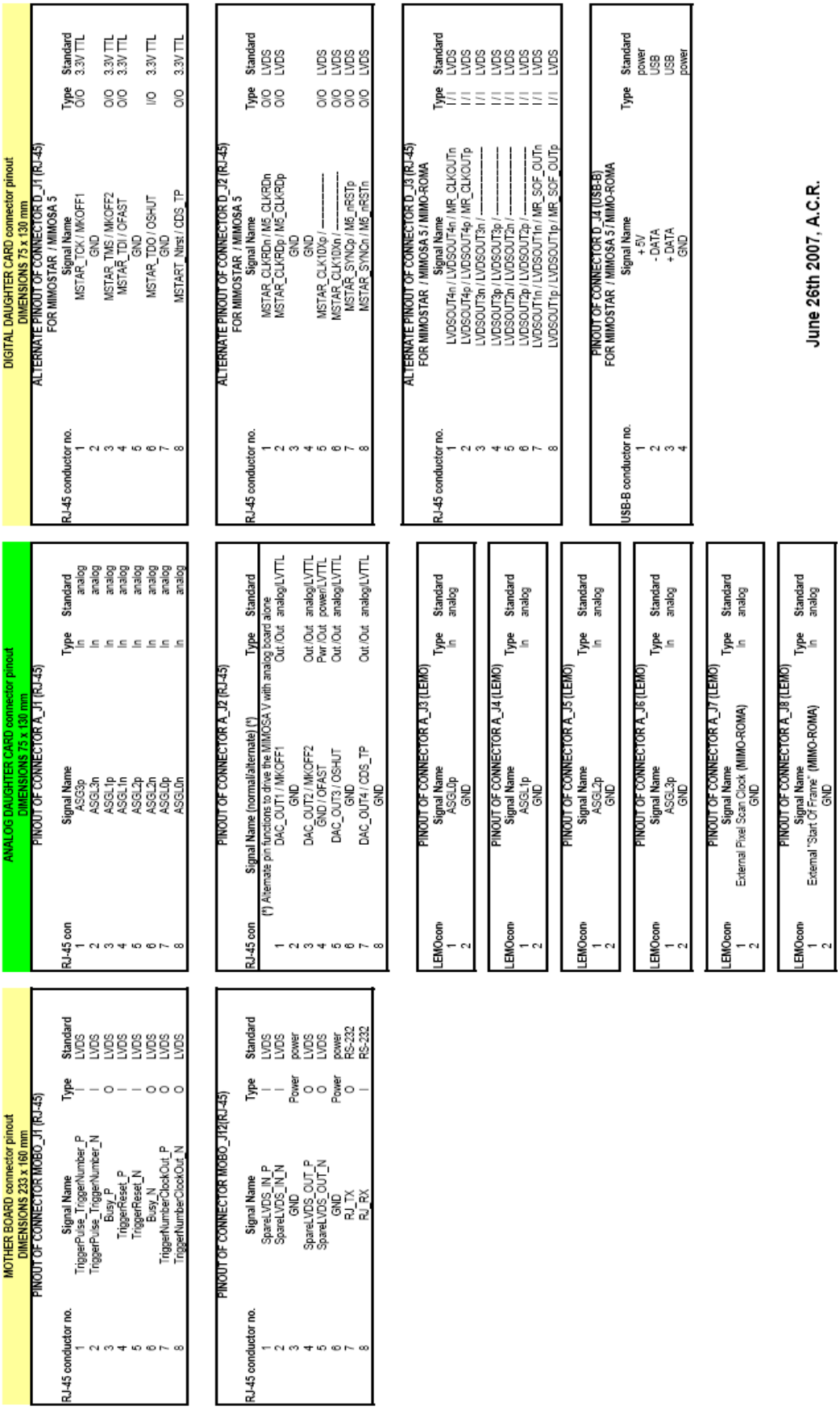


Fig. 2.3 EUDRB's front panel connectors pinout

definitions and drives the output signals through two RJ45 connectors to the MIMOTel. Fig. 2.4 shows the electric diagram of the “MIMOTel level adapter”.

The J3 connector receives detector timing signals (Scan Clock and Scan Reset) in LVDS format from an external source (like a sensor’s proximity board or another EUDRB). This feature has not been used in the demonstration telescope because a different port, on the analog daughter card) for such synchronization signals has been used.

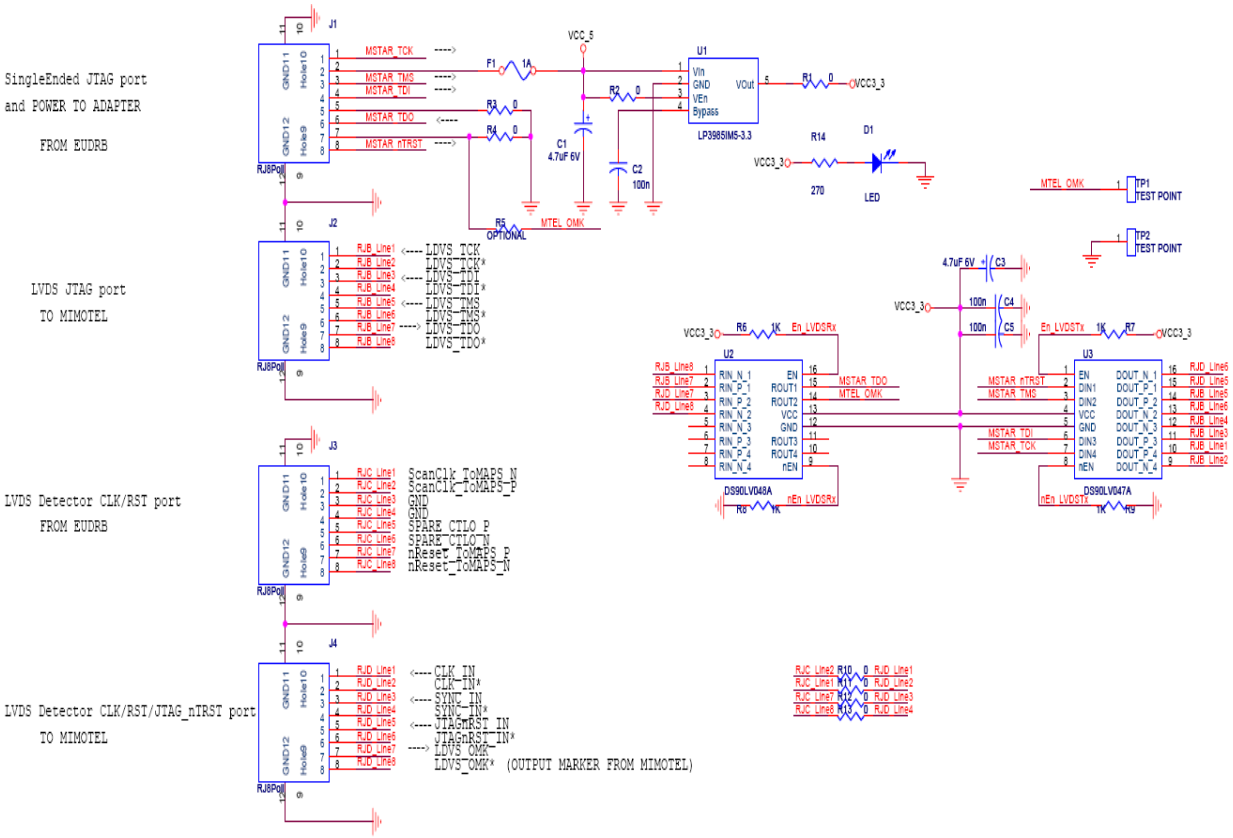


Fig. 2.4 MIMOTel level adapter electric diagram

2.3 EUDRB Analog Daughter Card (EUDRB_DCA)

The EUDRB_DCA is a board with standard PCI Mezzanine Card (PMC) format and a pair of IEEE-1386 PMC connectors toward the EUDRB_MOBO; the signals assignment on this connectors is not compatible with the PMC standard.

On the front side of the EUDRB_DCA two RJ45 connectors are available, whose pin definition is reported in Fig. 2.3; there are also 6 LEMO connectors, as shown in Fig. 2.5.

The EUDRB_DCA J1 is the port for the differential analog input signals, which are usually buffered and driven differentially by the sensors proximity board along a four-pair shielded twisted cable, to reduce noise pick-up.

The EUDRB_DCA J2 is an output port which could provide four channel of polarization voltages or four static signals to the MIMOSA-5 detector to configure its operating mode.

The main task of the EUDRB_DCA is to digitize the four input signals. The design of the ADC stage and its ADC driver is based on the designs developed by the IPHC in Strasbourg and the University of Cracow and exploited by the SUCIMA collaboration.

The four A/D converters used on the board are of the type AD9226 by Analog Devices. The AD9226 resolution is 12 bit and its sampling frequency can reach 65MHz.

The AD9226's differential inputs (with a 2V dynamic range) are driven by a buffer stage based on the differential amplifier AD8138. The buffer stages also take care of translating the single ended inputs from the LEMO connectors into the differential format; this option can be enabled by modifying solder-jumpers on the analog daughter card.

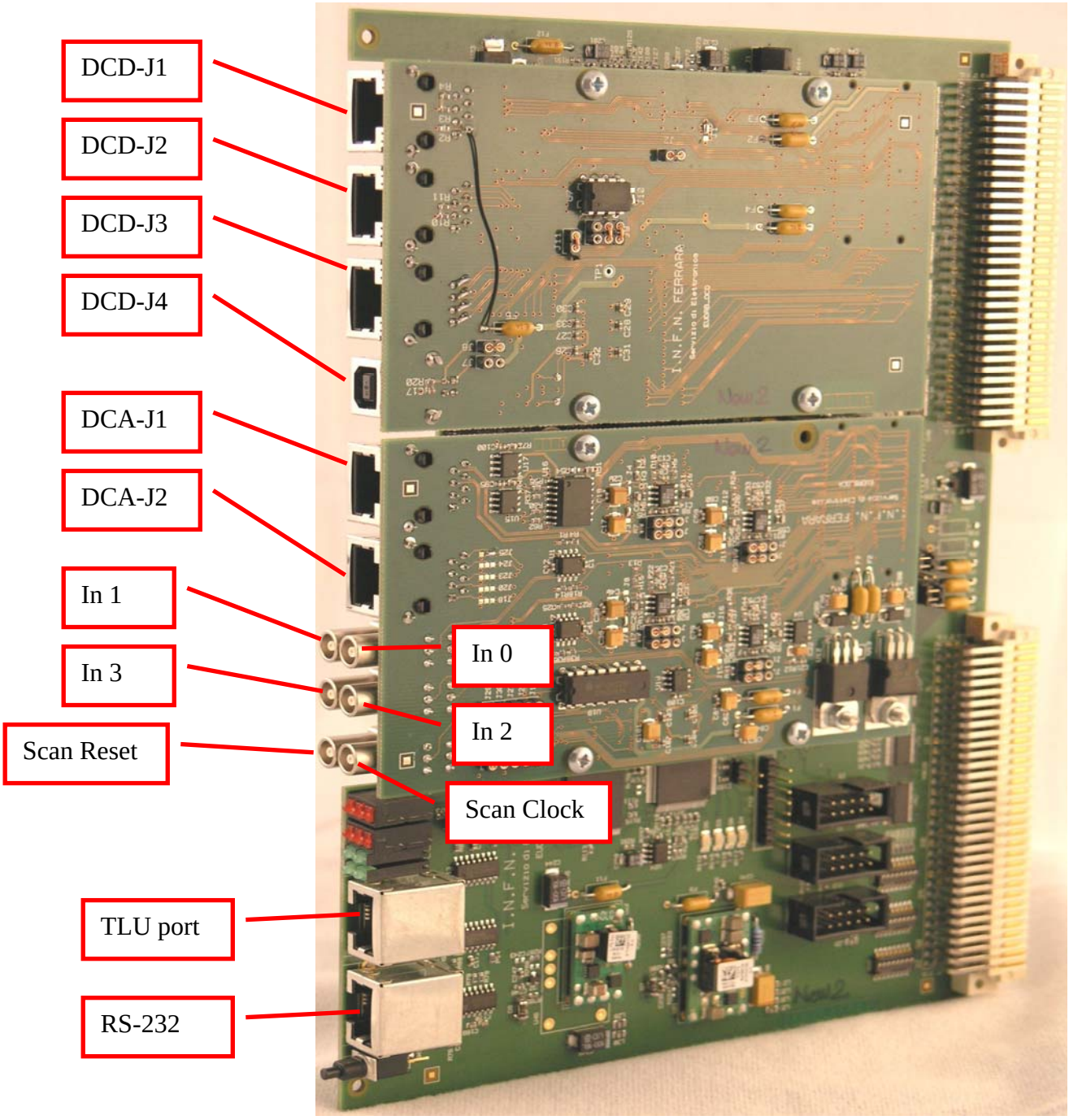


Fig. 2.5 EUDRB connectors

3 EUDRB operation: an overview (EUDRB_MIMOx firmware)

The main tasks of the EUDRB are:

a) to configure the operating mode and parameters of the sensor via its JTAG port; this task is performed by the NIOS-II (not needed for MIMOSA18)
b) to provide the sensor with the scan clock and a scan reset (which could be periodical for sensors like the MIMOSA-5) for initialization of the sensor's pixel addressing resources
c) to sample and A/D convert the pixel voltage levels as they sequentially appear at the analog inputs at the pace set by the scan clock signal
d) to store the voltage samples in the frame buffers, i.e. the four banks of 256K * 48bit SRAM described in the overview
e) to respond to a trigger according to one of the following two operating modes :

- "**raw**" mode:
  when a trigger is received by the board the scan is continued for a number of cycles equal to the frame size. Then the scan stops and the contents of the pixel memories are formatted and transferred to the output FIFO.
  For the EUDRB_MIMOx only two samples are transferred to the output FIFO for each pixel: the sample BEFORE the trigger and the sample AFTER. The samples data is formatted into a packet which includes a header and a trailer. For this reason the total event size is 4 long words larger than the output FIFO size. For this reason I have decided, for the time being, to drop out from the event packet the samples concerning pixel (255,255) of each subframe. The sensor scan resumes as soon as one event has been written to the output FIFO and the board can thus receive a new trigger while the readout of the previous event is pending.
- "**zero suppressed**" mode:
  the scan clock does never stop in this mode. When the trigger is received by the EUDRB the boards records the address of the pixel currently being sampled (the "pivot" pixel) and starts calculating on-the-fly the CDS for every pixels acquired afterwards for a number of clock cycles equal to the submatrix pixel count. If the result of the CDS for a pixel is above its specific threshold after correcting for its specific noise pedestal, then the pixel address and its signals are stored in the output FIFO. The pixel specific values for threshold and pedestal are stored in the pixel memories along with pixel data and need initialization via USB or VME before stating the actual zero-suppressed (ZS) data taking.

  With the EUDRB_MIMOx firmware the zero-suppression process is carried on by four "MIMOx_CDS sequencers" working in parallel, one for each of the inputs, i.e one per sub-matrix. Each "MIMOx_CDS sequencer" is equipped with a FIFO (internal to the FPGA and 1024 words deep) which stores locally the ZS data pending its transfer to the Output FIFO (256K words deep). The 4 de-randomizing FIFOs are continuously scanned by the sequencer described in the module "MIMOx_ZSFIFOBuffer" which then writes the ZS data from any sub-matrix to the output FIFO. If any of the de-randomizing FIFOs were filled in this process by an event with an anomalous multiplicity its FULL condition would appear as information in the trailer of the event data packet.

When the EUDRB receives the first trigger it sets the BUSY signal (preventing new triggers) only for the time needed for the scan for "hit" pixel.
As soon as the scan is over and the results stored in the output FIFO by the "MIMOx_ZSFIFOBuffer" module the BUSY is released and the EUDRB is ready to take another trigger.
If a second trigger arrives while the first has not been yet readout it is still processed immediately by the "MIMOx_CDS Sequencers" and "hit" data stored in the local FIFO.
The BUSY signal however stays ON until the event in the output FIFO is read and the "MIMOx_ZSFIFOBuffer" can transfer data of "hit" pixels from the local FIFOs to the output one.
In this way the operating rate of the DAQ system based on EUDRBs should reach the limit imposed by the sensor scan time.

f) to transfer the data requested by the trigger to the data acquisition system through the active output bus, either the VME or the USB2.0. The EUDRB prepared for the commissioning of the demonstrator telescope features a slave interface to the VME bus capable of transferring readout data in MBLT (Multiplexed Block Transfer) mode at a peak rate of about 40MB/s during the MBLT cycle.

The EUDRB can also be readout via the USB2.0. In the configuration prepared for the demonstrator telescope's first test beam the data is extracted from the output FIFO and moved to the USB2.0 link by the NIOS-II rather than by a dedicated sequencer described in VHDL. While this mode is useful when debugging the board because it allows an easy and deterministic way (the NIOS-II code can be written in C language) to custom-process the data as it is transferred, it does not allow to exploit the full bandwidth of the USB2.0 bus.

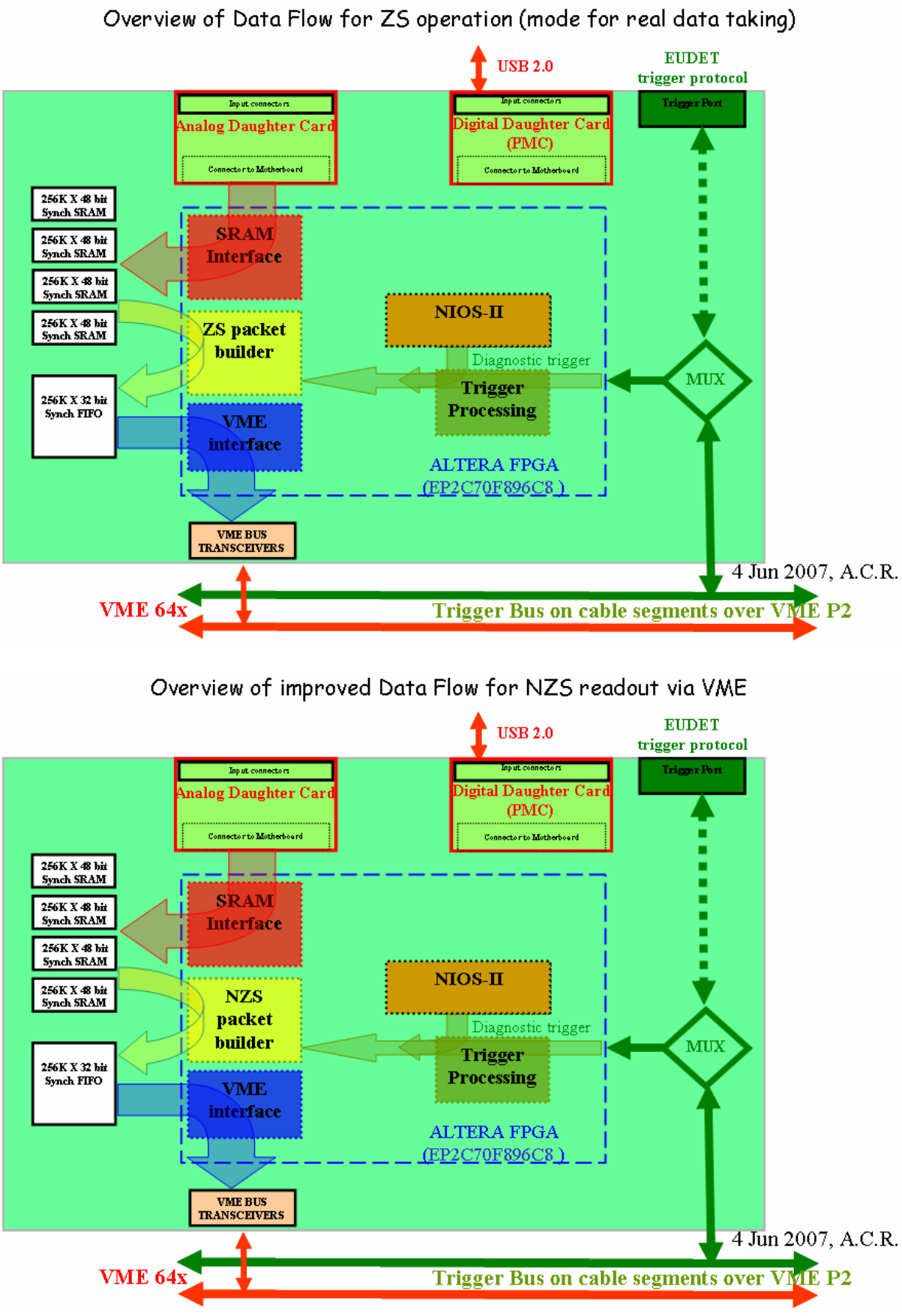


Fig. 3.1 Schematic representation of data flow for ZeroSuppressed and raw modes

3.1 LED indicators

The part reference on the EUDRB_MOBO electric diagram, the signal source and the meaning of each LED indicator is reported in the table below. The location of the LED indicators is shown in the picture in Fig. 3.1.

PartRef	SignalSource	Meaning when lit
J6-3	FPGA.L25	A command to the NIOS-II has been received via the VME port. The LED is turned off by the NIOS-II at the end of the execution.
J6-2	FPGA.M22	A flash on this LED means that the EUDRB has acknowledged a VME transaction
J6-1	FPGA.N21	The NIOS-II can be controlled via the VME or the USB port. If the LED is OFF (when jumper J4 is removed) then the NIOS-II is expecting commands from its UART connected to the EUDRB's RS-232 port
J5-3	FPGA.E13	The JTAG configuration of the detector has been performed, the detector has been provided with the scan clock and reset and the post-reset time has expired. The detector should thus be ready for data taking.
J5-2	FPGA.E23	The NIOS-II has been commanded to generate an internal fake trigger for diagnosing the EUDRB trigger response
J5-1	FPGA.L24	The VME interface holds one event to transfer or which is being transferred
J8-2	FPGA.C29	The EUDRB is operating in Zero Suppressed mode
J8-1	Jumper J4.2	The NIOS-II can be controlled via the VME or the USB port. If the LED is OFF (when jumper J4 is removed) then the NIOS-II is expecting commands from its UART connected to the EUDRB's RS-232 port
J7-2	U18.55	Not used
J7-1	U24.6	Power OK: all supply voltages are on. The LED goes off also while pushing the reset pushbutton

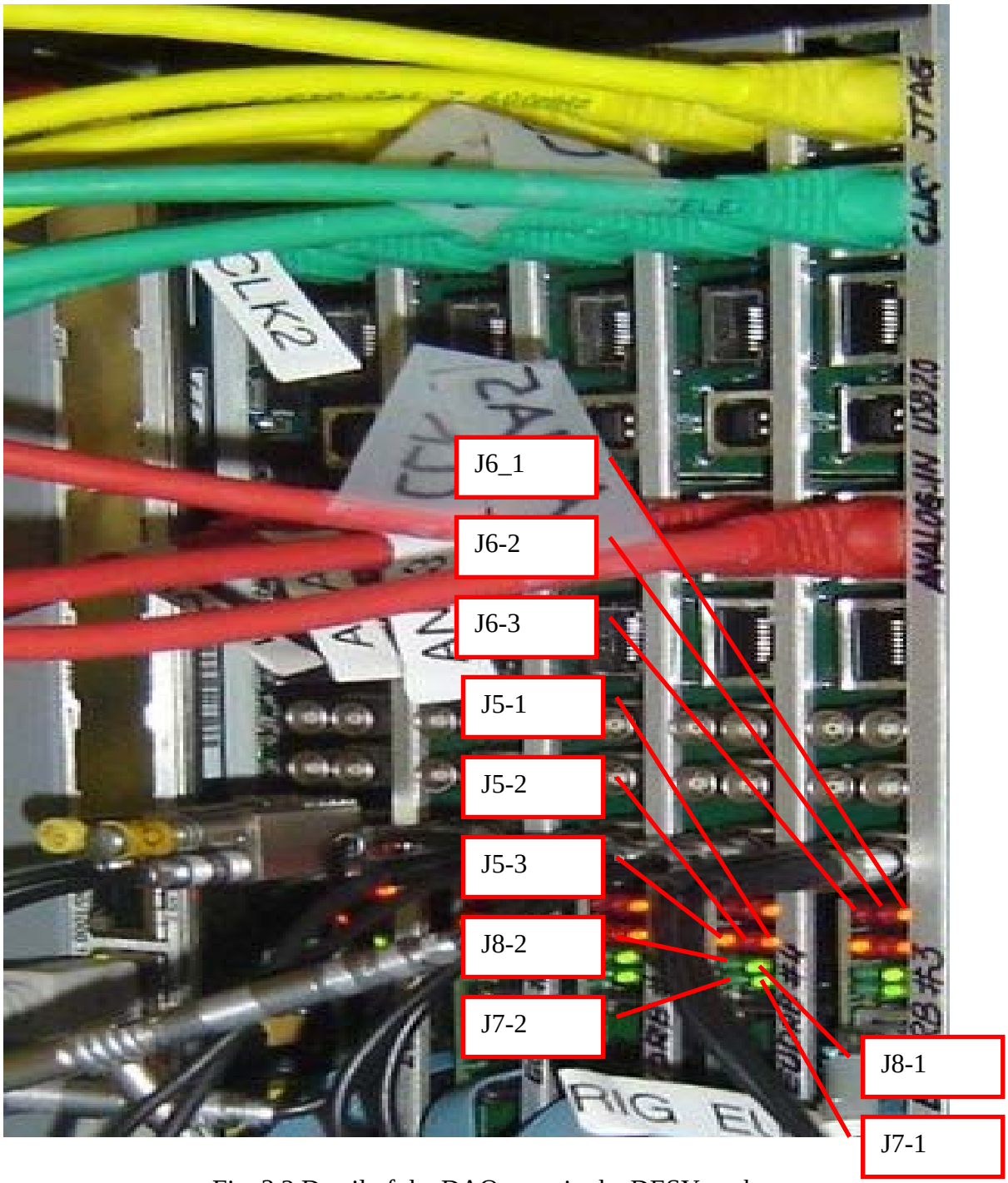


Fig. 3.2 Detail of the DAQ crate in the DESY testbeam

4 EUDRB FPGA: the “hard-coded” functions

The development system for the ALTERA FPGAs accepts entries in form of both:

- Schematics (file with an extension .bdf),

showing the interconnections among different functional units, be they simple primitive or blocks representing complex units fully described at lower levels of the hierarchy. Interconnections in a schematics are made by “wires” and their labels

- Text,

describing the functionality of a module in the hardware description language of choice, VHDL in this case. The VHDL description file (extension .vhd) can then be associated to a symbol which can be entered and interconnected in the schematic view

Both entry methods have been used in the design of the ALTERA FPGA installed on the EUDRB.

The “TopLevel_MIMOx.bdf” schematic of **Fig. 4.1** represents the top of the hierarchy of design files making up the whole project; it shows how the functional blocks associated with different aspects of the EUDRB operation are interconnected.

The whole design hierarchy can be recovered from the archive with the URL indicated in the introduction.

The main modules have been indicated by coloured block in Fig. 4.1, to help identifying them and their area of influence in the data flow diagram presented above.

4.1 The “MAPS_uC” block (1)

This block of the “TopLevel_MIMOx.bdf” schematic represents the set of VHDL files which describe the behaviour of the NIOS-II embedded microcontroller. The NIOS-II source files are automatically generated by a the “SOPC Builder” Wizard integrated into the ALTERA QUARTUS II design framework. Besides creating the VHDL source files, which are compiled together with all the other files in the project hierarchy to synthesize the logic network, the “SOPC Builder”creates a system description file (with extension .ptf) which is then used by the “ALTERA NIOS IDE” (Integrated Development Environment) to create the header files and the library links needed when compiling the C-source code for the NIOS-II.

A later chapter gives a little more detail on the NIOS-II firmware and operation.

4.2 The “MIMOx_CoreModules” block (2)

This block, shown in **Fig. 4.2**, is the “container” of all the VHDL description for the resources controlling the EUDRB operation.

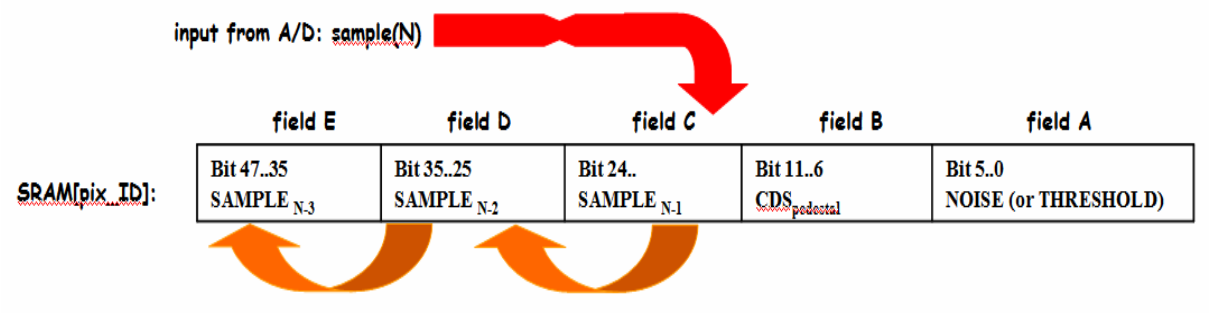
The “MIMOx_CoreModules” can easily be singled-out from the entire project to be developed and simulated independently from the NIOS-II block for faster turn-around times.

The resources involved in the control of specific tasks are described each in a well definite block. A brief description of the modules in which the “MIMOx_CoreModules”is partitioned follows below.

4.2.1 The “MIMOxPixelsToSRAMs” block (2.1)

This block contains the VHDL description of the resources generating the detector timing signals, controlling the A/D conversion of pixel voltage samples and controlling the subsequent storage of A/D output in the pixel frame buffer memories.

It is worth noting here how the “MIMOxPixelsToSRAMs” unit operates on the frame buffers, with the help of the following diagram:



Each memory location, 48 bit wide, is associated with one pixel of the sensor and it holds the values of the last three pixel voltage samples, encoded in a 12bit field. The memory word also holds the values of the pixel pedestal, i.e. the baseline value of the result of the CDS operation, and the threshold that decides whether the pedestal-corrected signal after the CDS is to be considered a "hit" and, as such, copied into the output FIFO (along with the pixel coordinate).

This memory organization follows the standard already established by the designers of the data acquisition boards developed at the IPHC in Strasbourg and by the University of Cracow collaborating at the SUCIMA project. These previous designs only provided storage for two pixel samples, resulting in a memory word of just 32bit.

The addition of the storage for a further pixel sample should provide more information on the history of the pixel voltage in a wider interval around the trigger time, which could be useful when characterizing new sensors.

4.2.2 The "MIMOxTrigProcSequencer" block (2.2)

This block contains the VHDL description of the scheduler of all processes involved in the response of the EUDRB to triggers (real from EUDET-TLU or fake from the on-board NIOS-II). A flow diagram of the scheduler state machine is shown in **Fig. 4.3.**

The "MIMOxTrigProcSequencer" contains also one instance of the "MIMOx_TLUInterFace" VHDL component, which describes the resources used in extracting the trigger information sent, according to a serial communication protocol, by the EUDET TLU. This module also controls the Busy line monitord by the TLU to determine when the EUDRB is ready for the next trigger. The EUDRB connected to the TLU through the front panel connector is the "TLU-Interface".

The status of the BUSY signal that the "TLU-Interface" returns to the TLU reflects the BUSY status of all the TLU in the crates; this information is obtained by means of an open collector signals distributed to all the EUDRBs in the crate via the user-defined lines of the VME bus.

4.2.3 The "MIMOx_ZSEventBuilder" block (2.3)

This block performs, when scheduled by the "MIMOxTrigProcSequencer" module, the fetching of "hit" data in Zero-Suppressed mode without stopping detector scan and frame buffer update. Four "MIMOx_CDS" submodules perform the "hit"extraction in parallel (CDS and comparison with threshold after perdestal subtraction), one for each submatrix, as already described in the EUDRB operation overview on page 8.

This processor controls local FIFOs to store hit information until the output FIFO is available for receiving a new event data packet. The event data packet contains "hit" signal amplitude and pixel address encompassed within a Header and a Trailer. the format is given in a later chapter.

4.2.4 The “MIMOx_NZSEventBuilder” block (2.4)

This block performs, when scheduled by the “MIMOxTrigProcSequencer” module, the transfer of two voltage samples for each pixel, one BEFORE and one AFTER the trigger to the output FIFO as soon as this is available for receiving a new event data packet. The event data packet contains only the sample voltage amplitude. No need for addressing information is needed since the scan order is a priori defined. A Header and a Trailer are foreseen for flagging the begin and the end of the event data packet, as explained in detail in a later chapter. The sensor scan stops when the “MIMOx_NZSEventBuilder” begins operation and restart automatically when the event packet produced by the trigger has been completely readout.

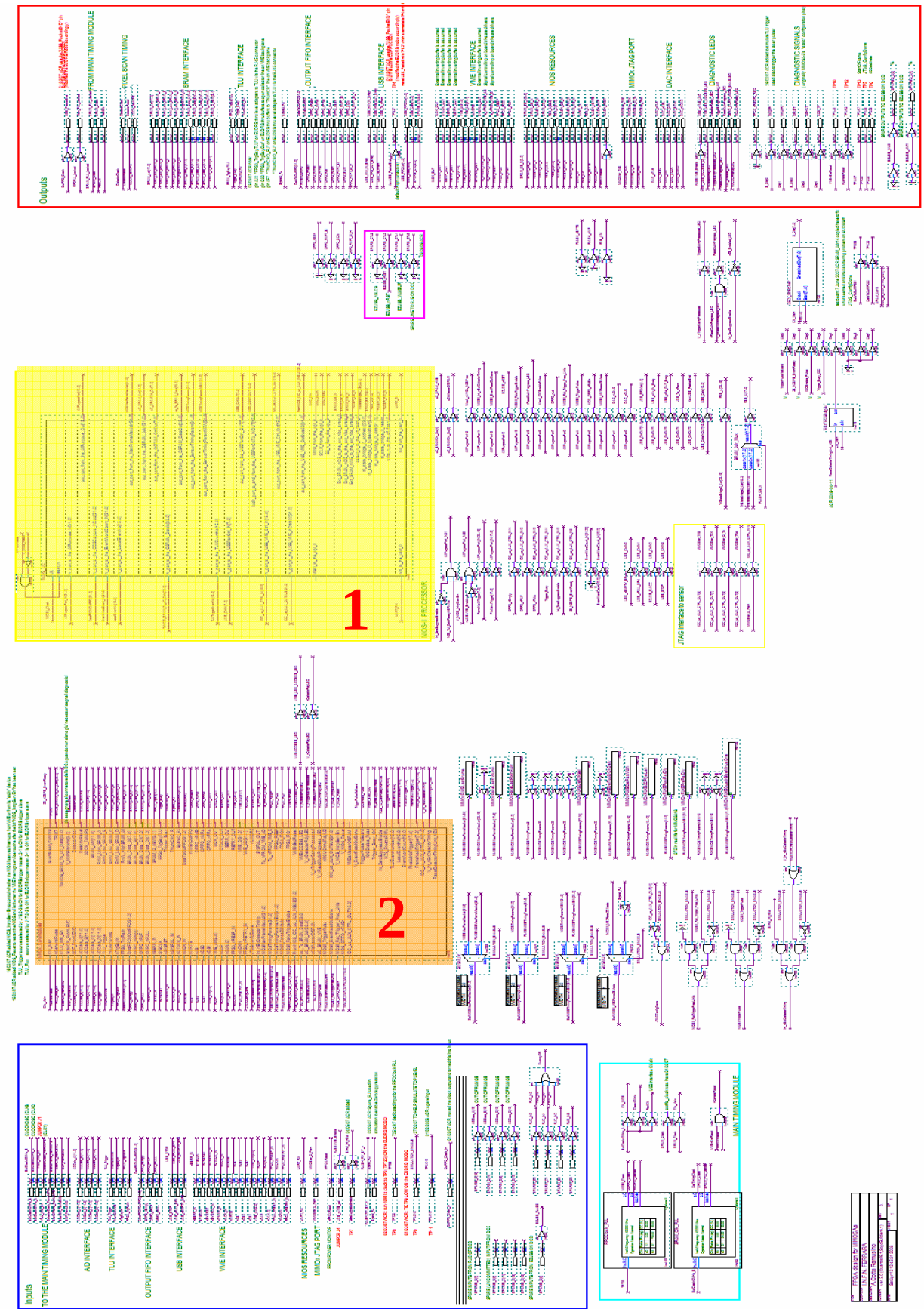
4.2.5 The “MIMOxIOController” block (2.5)

This block performs, when scheduled by the “MIMOxTrigProcSequencer” module, the transfer of the contents of the output FIFO to the VME CMU by responding to “Multiplexed Block Read Transfers” (MBLT for short) commanded by the CPU. The MIMOxIOController module actually demands this task to the “MIMOxMBLTSequencer” sub-unit.

Fig. 4.4 shows the flow chart of the “MIMOxMBLTSequencer” supervising this process. The sequencer is complicated from the fact that the VME MBLT transfer of a large number of words is broken down by the VME CPU in a number of “atomic” block transfers of up to 2048 bytes, of length not fixed a priori.

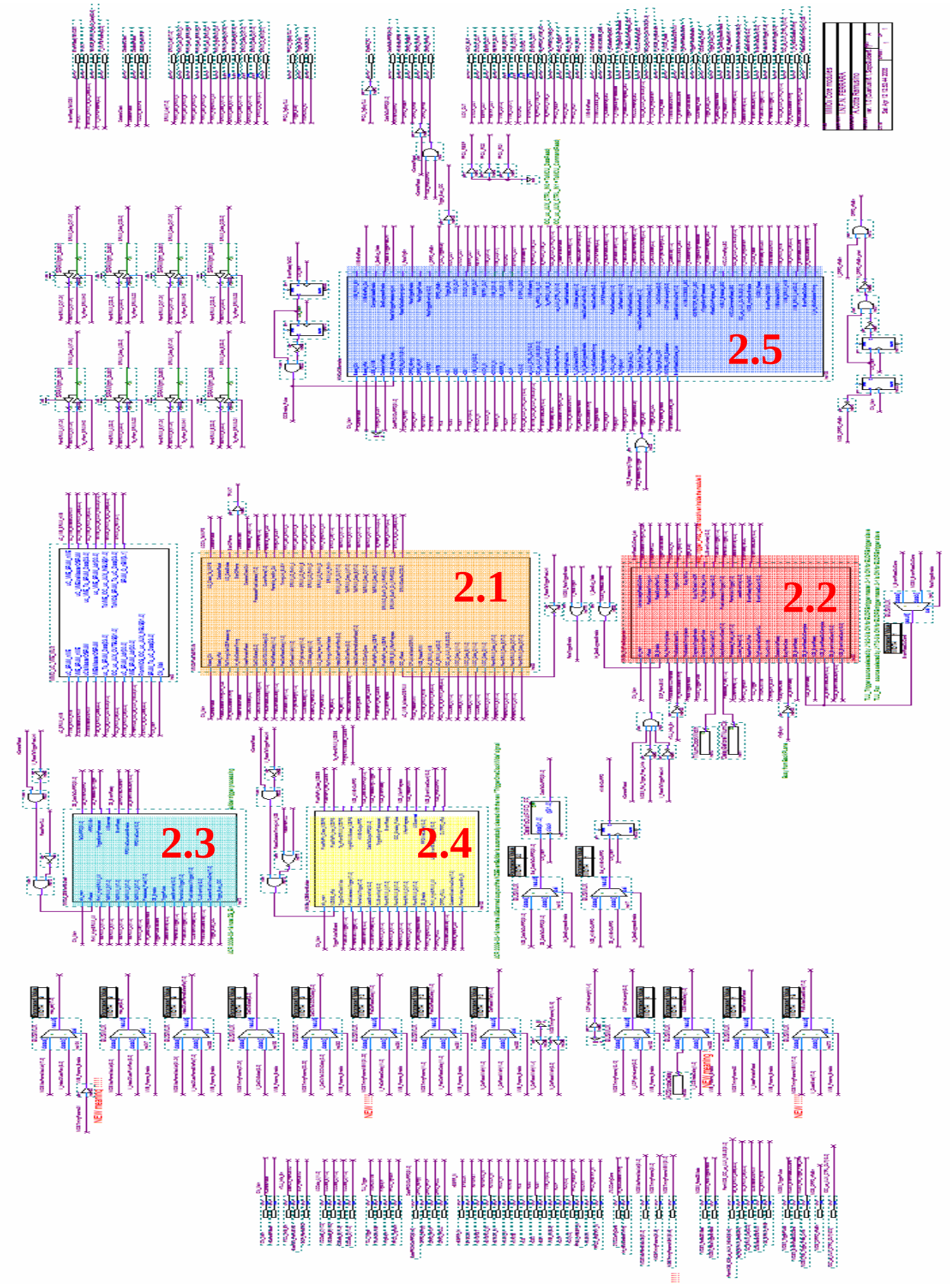
Fig. 4.5 shows two of such “atomic” MBLT tranfers scheduled by the VME CPU at the end of an MBLT transfer of 8208 bytes.The signals shown in the picture are internal FPGA signals captured thru its JTAG port by means of special dedicated resources. **Fig. 4.5** also shows how the transfer of data from the “MIMOx_ZSEventBuilder” local FIFO buffers starts soon after completion of the readout of data from a previous trigger.

The “MIMOxIOController” also features a VME A32/D32 slave interface for all configuration and house-keeping tasks controlled by the VME_CPU. It is worth noting here that the VME interface can map the frame buffers directly into the VME addressable space, in order to allow the user to directly write the pixel pedestal and threshold fields described above. The details of the resources mapped onto the VME bus are given in a later section of this note.



Fig,4.1 EUDRB_MIMO's design "TopLevel_MIMOx.bdf" schematic.

Block 1: MIMOx_CoreModules. Block 2: NIOS-II microcontroller instance



Fig,4.2 EUDRB_MIMOx's design "MIMOx_CoreModules.bdf" schematic.
Block 1: "MIMOxPixelsToSRAMs". Block 2: MIMOxTrigProcSequencer. Block 3: MIMOx_ZSEventBuilder. Block 4: MIMOx_NZSEventBuilder. Block 5: MIMOxIOController

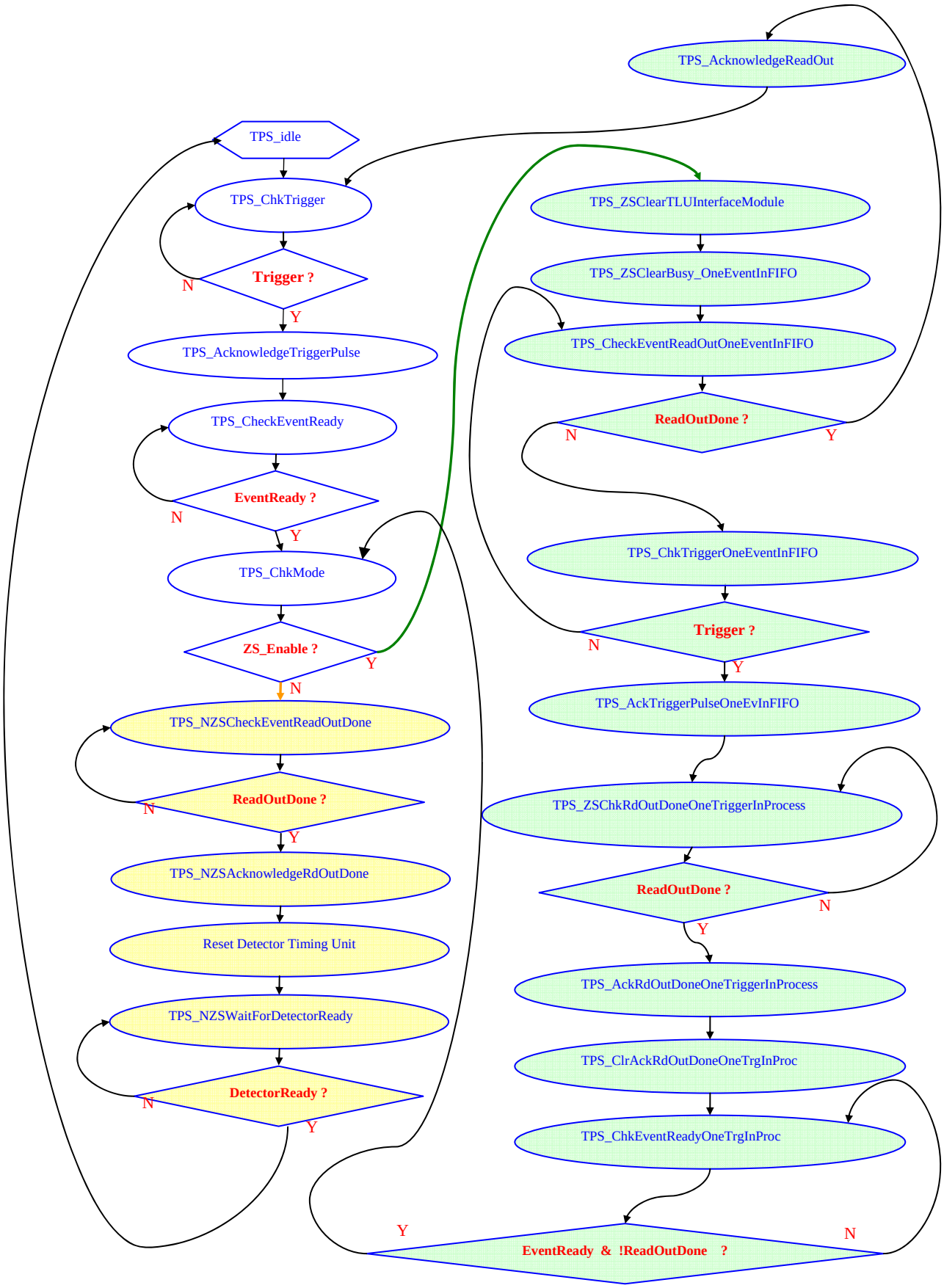


Fig. 4.3 Flow chart of the "MIMOxTrigProcSequencer" finite state machine

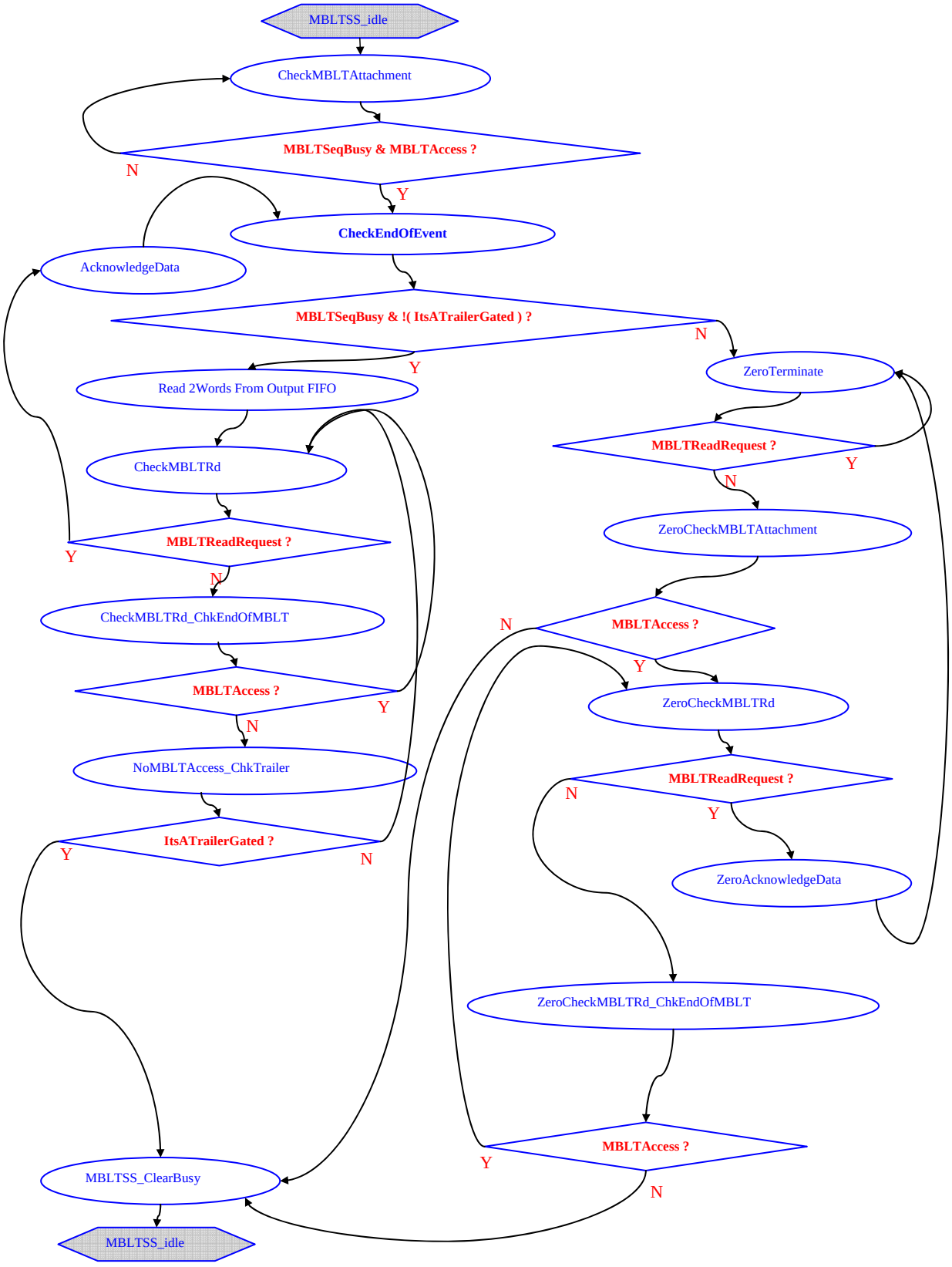


Fig. 4.4 Flow chart of the "MIMOxIOController| MIMOxMBLTSequencer" finite state machine

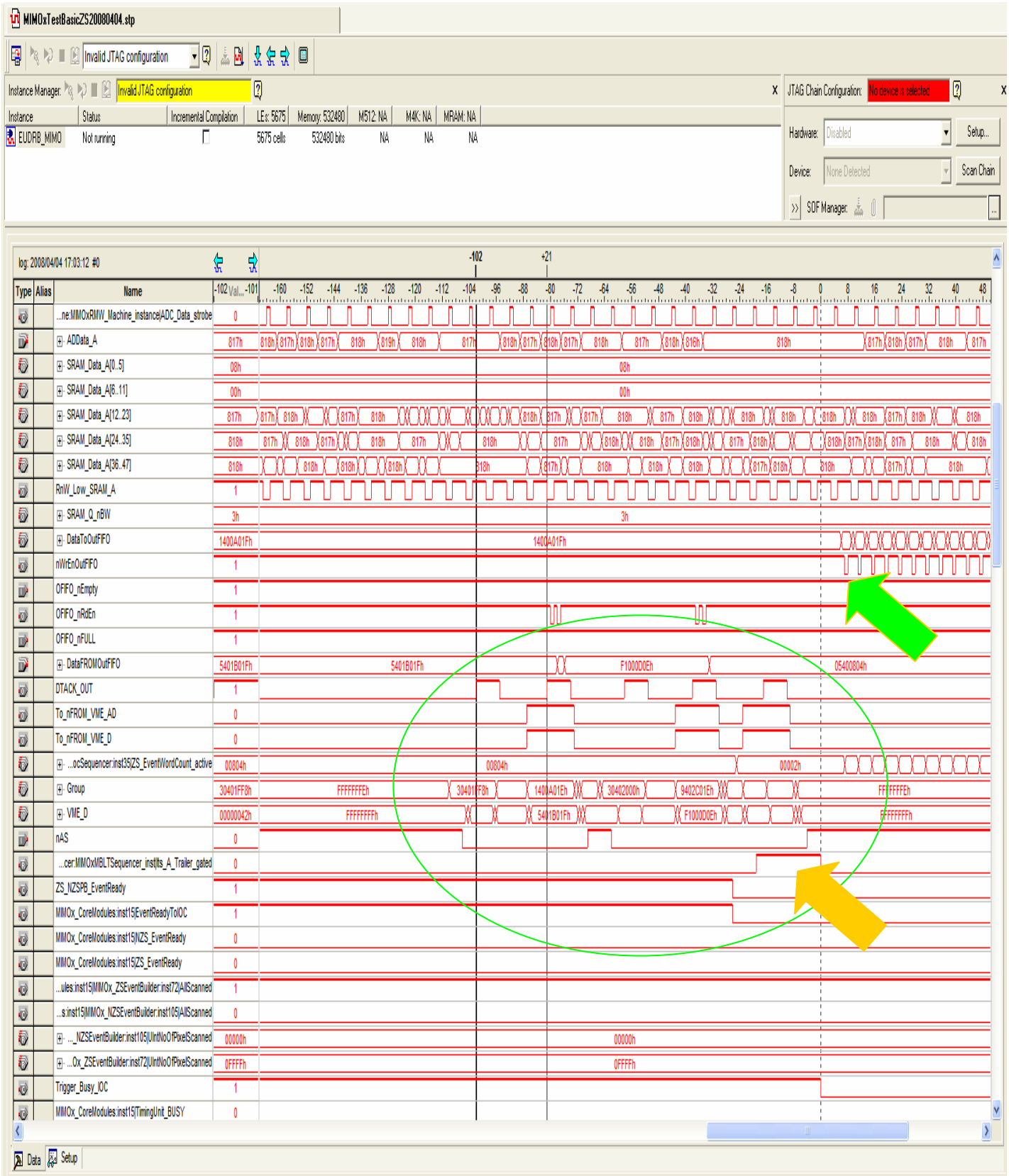


Fig. 4.5 Timing diagram of internal FPGA activity captured through its "Signal Tap Logic Analyzer": transfer of a new event from the MIMOx_CDS Sequencer local FIFOs to the output FIFO starts (green arrow) immediately after one event is read from VME (orange arrow). It can be seen as the VME CPU is not very efficient sometimes at scheduling MBLT transfers. Two MBLT cycles are captured here (green circle): one with just one 64bit word as payload and one with two 64bit words (including the Trailer which marks the end of event).

5 EUDRB FPGA: the NIOS-II firmware

As it was anticipated in the previous section, the “ALTERA NIOS IDE” (Integrated Development Environment) is the environment (a “custom” version of the Eclipse software development framework) of choice to develop the applications running on the NIOS-II.

The NIOS-II application prepared for the EUDET demonstrator first test beam by the writer performs tasks that are needed only for off-line board diagnostics, housekeeping and stand-alone data acquisition as well as tasks that are performed during the initialization phases of the demonstrator telescope data taking.

The most important task which the NIOS-II performs at the start of a data taking run is the configuration of the programmable features of the sensor through its JTAG port. The relevant commands for downloading the default configuration and those for changing the defaulta parameters as needed are described in a later section of this note.

The NIOS-II can perform a number of low level diagnostic operation, like writing and reading back test patterns into the frame buffers and the output FIFO, useful in commissioning the boards. For this purpose it is best to connect a computer running a terminal emulator program to the RS-232 port of the EUDRB. The terminal emulator program must be configured for a 115200 baud rate, 8bit, no parity, 1 stop bit, no flow controls. Jumper J4 on the board must be OFF to use the RS-232 port; a custom assembly is needed to adapt the standard 9-pin D connector of a usual computer serial port to the RJ45 connector of the EUDRB.

The NIOS-II source code developed consist of about 6000 lines of code between the “EUDRBFirmWareMIMOx.c” and the “EUDRBFirmWareMIMOx.h” files.

The compilation results in a 140kByte executable which leaves the rest of the 1MByte program/data memory available for stack/heap and application data.

The code developed for the configuration of the sensors via JTAG is a porting of the C++ code developed by Gilles Claus of IPHC for the data acquisition system based on the IPHC USB2.0-based MAPS readout card.

6 Diagnostics and stand-alone sensor readout via the USB2.0 port

A GUI application was also developed by the writer to perform diagnostics, housekeeping and stand-alone sensor readout via the EUDRB’s USB2.0 port. The data path for the readout via USB2.0 is shown in Fig. 6.1, which refers to the EUDRB configuration used for the demonstrator telescope; in this setup the NIOS-II microcontroller is in charge of fetching the event data from the output FIFO and send it over the USB2.0 bus to the host computer.

The URL for the archive containing the Microsoft Visual Studio 6.0 project is:
http://www.fe.infn.it/u/cotta/ILC/EUDET/EUDRB-MIMOx/USBLinkToEUDRBMIMOx.rar

The URL for the EUDRB’s USB port driver is:
http://www.fe.infn.it/u/cotta/ILC/EUDET/EUDRB-MIMOx/EUDRB_USB_DriverGoodinf.rar

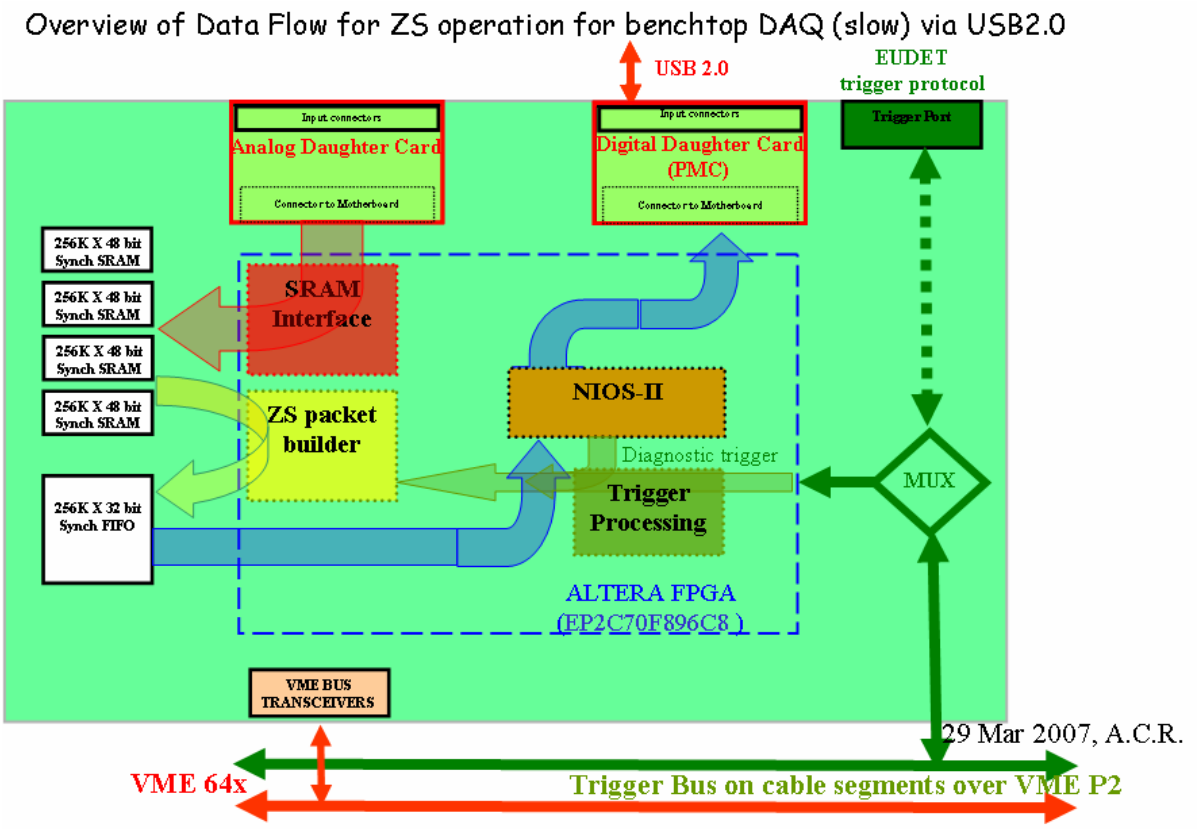


Fig. 6.1 Schematic representation of the data flow for stand-alone data acquisition via the EDURB's USB2.0 port

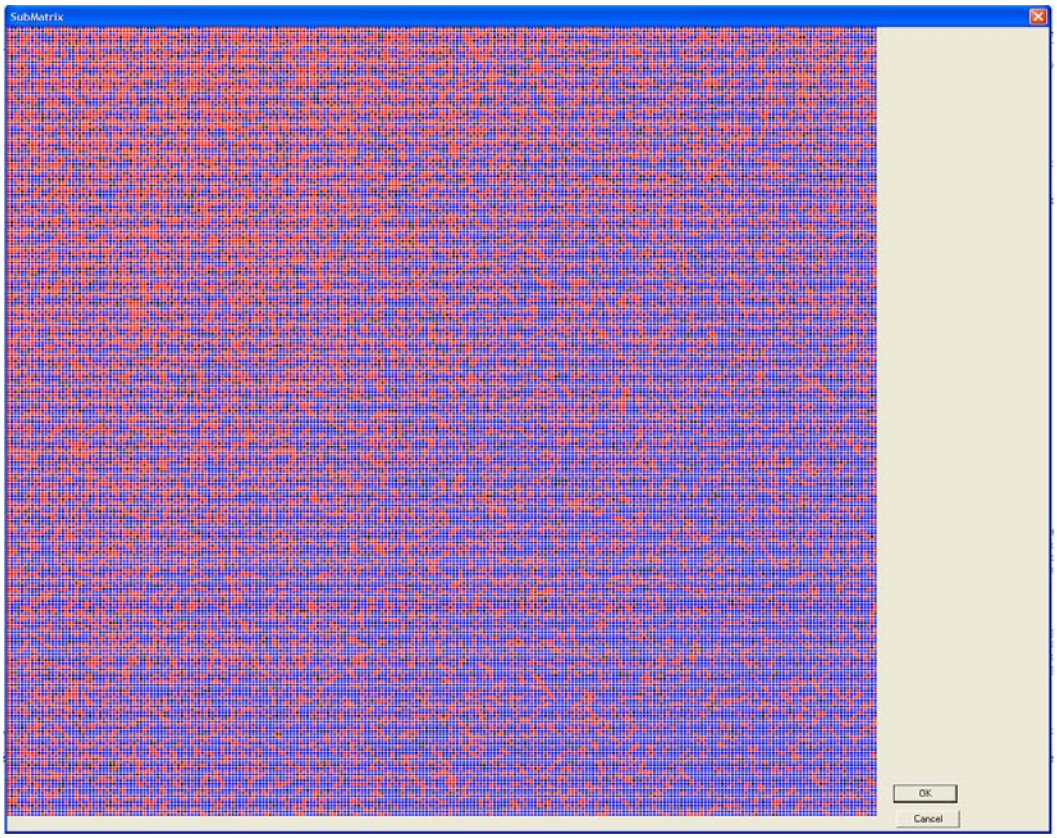


Fig. 6.2 MIMOSA18 frame captured with the GUI application through the USB2.0 port

7 EUDRB's VME interface detailed description

The EUDRB responds to VME accesses to its CR/CSR (Configuration ROM/Control and Status Register) area defined by the VME64 standard. The EUDRB also acknowledges the following VME accesses in User Space:

- Write and Read "Single Cycle Transfers" (SCT) in A32 / D32 mode
- MBLT block transfers, for which the data is transferred 8 bytes at a time. In MBLT transfers the A31..1 lines and the nLONGWORD line of the VME Data Transfer Bus are used to transfer the 4 most significant bytes

7.1 EUDRB CR/CSR Address space and geographical addressing

The EUDRB implements the (optional) minimal set of registers in the CR/CSR space defined by the VME64 standard:

- the Base Address Register (**BAR**): a byte-wide register located at offset **0x7FFFF** in the CR/CSR space. Bits 7..3 of the BAR contain a code (allowed range for 'm' in the figure is 1 ..31 decimal) is used as a comparison term for the VME Address lines A(23..19) to determine whether or not a given EUDRB acknowledges a VME access. The contents of the BAR register thus uniquely identify a board in the crate, since no two boards can have the same BAR value. The method to set the BAR with a unique address for each board in the crate (allowed range is 1 .. 31) is not defined by the standard VME64.

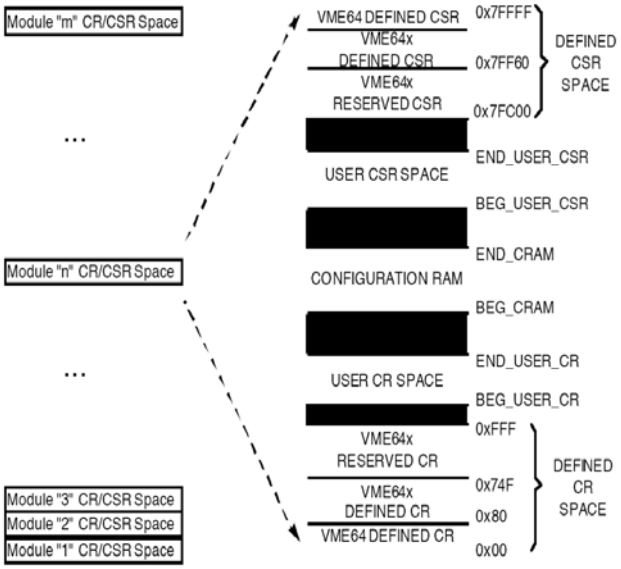


Figure 10-1: Structure of CR/CSR Space

- the Bit Set Register (**BSR**): a byte-wide register located at offset **0x7FFB** in the CR/CSR space which may be used mainly to put the board in the reset state.
- the Bit Clear Register (**BCR**): a byte-wide register located at offset **0x7FF7** in the CR/CSR space which can be used mainly to clear the board reset condition.

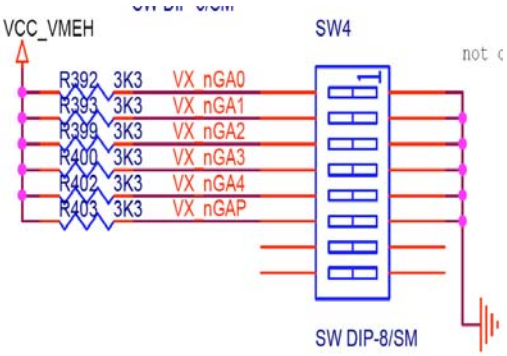
To determine its own base address the EUDRB simply transfers to the BAR the pattern present at the dedicated Geographical Address pins on row D of the 5-row, VME64X connector.

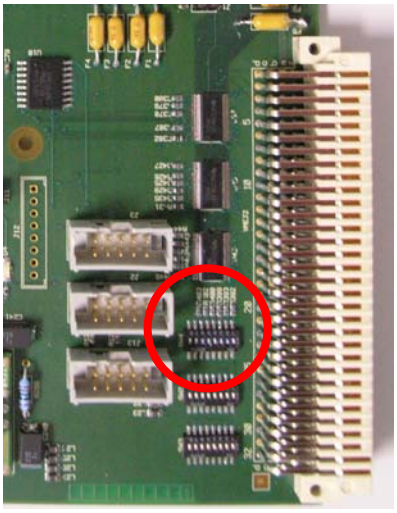
The slot position is encoded into five bit and one bit is added for parity check. The VME signal names for the Geographical Addressing pins are nGA[4..0], while nGAP is the parity line. The prefix n means that negative logic is used for these signals.

As the standard requires, the nGA[4..0] position information is mapped to bits BAR[7..3]; the 5 most significant VME address bits set by a VME transaction will have to match the BAR[7..3] code in order for the board to respond to the transaction.

If the EUDRB is to be placed in a standard VME crate which lacks the (nGA4..0, nGAP) pins, then the Geographical Addressing information must be supplied by the switches in the dip-switch SW4

Note: if the EUDRB is to be plugged into a VME64x backplane then all switches in the SW4 dip-switch must be OFF (toward the resistors).





The table below reports the (nGA4..0, nGAP) patterns for each position in a crate with 21 positions and the corresponding base address. Please note that the geograpical address bits are in negative logic and thus a "GND" connection at the nGAx line turns into a logical "1". The nGAP parity bit is set so that the number of "GND" in a pattern is always ODD.

Position	nGAP nGA4 nGA3 nGA2 nGA1 nGA0	EUDRB Base address
1	Open Open Open Open Open GND	0x08000000
2	Open Open Open Open GND Open	0x10000000
3	GND Open Open Open GND GND	0x18000000
4	Open Open Open GND Open Open	0x20000000
5	GND Open Open GND Open GND	0x28000000
6	GND Open Open GND GND Open	0x30000000
7	Open Open Open GND GND GND	0x38000000
8	Open Open GND Open Open Open	0x40000000
9	GND Open GND Open Open GND	0x48000000
10	GND Open GND Open GND Open	0x50000000
11	Open Open GND Open GND GND	0x58000000
12	GND Open GND GND Open Open	0x60000000
13	Open Open GND GND Open GND	0x68000000
14	Open Open GND GND GND Open	0x70000000
15	GND Open GND GND GND GND	0x78000000
16	Open GND Open Open Open Open	0x80000000
17	GND GND Open Open Open GND	0x88000000
18	GND GND Open Open GND Open	0x90000000
19	Open GND Open Open GND GND	0x98000000
20	GND GND Open GND Open Open	0xA0000000
21	Open GND Open GND Open GND	0xA8000000

7.2 EUDRB resources mapped to VME user address space

The VME's address modifier signals determine the type of each transaction occurring on the VME Data Transfer Bus (DTB). The EUDRB recognizes the following address modifiers, related to addressing modes and transaction types:

- **AM=0x2F**: used for accessing the EUDRB's CR/CSR space in **A24 D32** mode
- **AM=0x09**: used for accessing the EUDRB's non-privileged (User) space in **A32 D32** mode
- **AM=0x08**: used for accessing the EUDRB's non-privileged (User) space in **A32 D64 MBLT** mode

Table 7.1 shows what EUDRB resources are accessible via VME and their allocation in the VME addressing space with respect to the board's Base Address.

Table 7.2 describes the meaning of the bits in the "Functional Control Status Register".

Resource name	OFFSET (hex)	Data flow	Brief description of resource
FunctionalCtrl_Stat_Reg	0	R/W	EUDRB main control / status register
CommandToMCU	10	W	CommandToMCU register and CommandReady Flag (1bit) are set by an access to this location
DataToMCU	14	W	DataToMCU register and DataReady Flag (1bit) are set by an access to this location
DataFromMCU	18	R	Register written by the MCU with output data
SensorInterfaceParam_Reg_01	0x20	R/W	Register holding parameters for the configuration of the sensor interface timing
SensorInterfaceParam_Reg_23	0x24	R/W	Register holding parameters for the configuration of the sensor interface timing
SensorInterfaceParam_Reg_45	0x28	R/W	Register holding parameters for the configuration of the sensor interface timing
SensorInterfaceParam_Reg_6	0x2c	R/W	Register holding parameters for the configuration of the sensor interface timing
DataReadPort	400000	R	FIFO for Zero Suppressed MBLT readout
StatusDataReadPort	400004	R	**Bit 31**: this bit is **set** when: • the event has been written into the output FIFO (WIDTH: 32bit DEPTH: 2^{18}) **Bit 31** is **CLEARED** automatically when the event is read out Bits 20..0: count (max value is 2^{18} - 4) of 32bit words written inside the output FIFO (NOT INCLUDING the Header -2 words- and the Trailer -2 words-).
PixelSRAM Port's base address	800000	R/W	Each channel has two memories 24bit wide (for a total of 48 bit bus width) and (2^{18}-1) locations deep. The VME address lines are so decoded: VMEA[22] = 1 to point to upper 24 bits of the memory buses; 0 to point to lower 24 bits of the memory buses VMEA[21..20] = 00 : points to memories for channel A 01 : points to memories for channel B 10 : points to memories for channel C

			11 : points to memories for channel D VMEA[19..2] = point to a location in the 256K (2^{18}-1) available to each channel (4-byte boundary alignement) NOTE: VMEA[0] does not exist

Table 7.1: address mapping of the EUDRB resources

FunctionalCtrl_Stat_Reg (OFFSET = 0x0)			
EUDRB main control / status register (32 bit)			
bit		Name	Brief description
0	R	TriggerProcessingBusy	Internal signal name: MBLT_ReadOut_Busy Goes to '1' when the an event data packet has been written in the output FIFO following a trigger and it goes to '0' when the event has been completely readout
1	R	OFIFO_nEmpty	Internal signal name: OFIFO_nEmpty '1' means that the output FIFO is not empty
2	R	OutputFIFOAlmostFull	Internal signal name: OFIFO_nPAF '0' when the output FIFO has only 1K locations free (out of 256K)
3	R/W	NIOS_Reset	Internal signal name: V_NIOS_Reset '1' puts the NIOS II in the reset state. '0' clears the reset NIOS II reset condition
4	R/W	Compact_nExtended	Not implemented
5	R/W	ZS_Enabled	Internal signal name: V_ZeroSup_Mode '1' forces the "Zero Suppressed" readout mode (**in OR with the "NIOSII_MS2ForceZSMode" signal**)
6	R/W	TriggerProcessingReset	Internal signal name: V_ResetToTriggerProcUnit A '1' sets the V_ResetToTriggerProcUnit and a '0' clears it. The V_ResetToTriggerProcUnit reaches the following modules: - "**MIMOx_TrigProcSequencer**" high level handler of the trigger processing chain. This module resets in turn also the following modules: - "**MIMOx_TLUInterFace**" - "**MIMOxADStrobeGeneration**" and the "**MIMOxPixAdrGeneration**" (but only if the EUDRB is TLU_Interface_Master) - "**MIMOxMBLTSequencer**", handler of the readout process - "**MIMOxIOController:MIMO_ZSEventBuilder**" handler of trigger processing in ZS mode - "**MIMO_NZSEventBuilder**" handler of trigger processing in NZS mode - "**MIMOxPixelToSRAMs:MIMOxRMW_Machine**", handler of the SRAM Read_Modify_Write process for storing pixel samples in the pixel memories
7	R/W	ResetTLUMasterBusy	Internal signal name: V_ ResetMasterBusyTerm In the latest firmware version (code: 2) the "TLU Interface" board holds the BUSY line active from the moment it receives a trigger to the moment the "BusyTermResettableFromVME" is cleared. To clear the "BusyTermResettableFromVME" signal it is necessary to first write a '1' to the

			"ResetTLUMasterBusy" bit and then write a '0' to the same bit.
8	R/W	FakeTriggerEnable	Internal signal name: V_FakeTrigEn '1' disables the TLU generated signals and enables the diagnostic ones, generated by NIOS-II under control of the VME or the USB port
9	R/W	VME_IsMasterOfSRAM	1: allows the VMEBus to connect directly to the pixel storage memories (pixel SRAMs) for debugging purposes or for setting pedestals and thresholds. While in this mode the VME address lines are so decoded: VMEA[22] = 1 to point to upper 24 bits of the memory buses; 0 to point to lower 24 bits of the memory buses VMEA[21..20] = 00 : points to memories for channel A 01 : points to memories for channel B 10 : points to memories for channel C 11 : points to memories for channel D VMEA[19..2] = point to a location in the 256K range assigned to each channel When accessing the SRAMs via the VMEBus the data lines carry the following information: • when VMEA[22] = 0 VMED[5..0] = threshold data VMED[11..6] = pedestal data VMED[23..12] = sample N+1 data • when VMEA[22] = 1 VMED[11..0] = sample N data VMED[23..12] = sample N-1 data
10	R/W	FirmWareVersion_bit0 (LSB)	Three bit code to identify the FirmWare version
11	R/W	FirmWareVersion_bit1	
12	R/W	FirmWareVersion_bit2 (MSB)	
13	R/W	VME_ForceInternalTiming	Internal signal name: V_ Int_nExtDetectorTiming '1' forces internal timing mode i.e. the board is not synchronized to a Master Timing EUDRB. (**Note: V_ Int_nExtDetectorTiming is in OR with the "NIOSII_Int_nExtDetectorTiming" signal**)
14	R	VME_Params_Enable	The parameter is set by the NIOS-II. It normally defaults to 1 to enable control of sensor timing parameters from VME. It can be controlled by the USB interface when running the EUDRB in stand-alone mode
15	R	uC_VME_IsMasterOfSRAM_status	1: the pixel memories are controlled by the NIOS-II or by the VME and not updated. Default = 0
16	R	ZeroSuppressEnable_status	1: the ZeroSuppressed mode is enabled. Default = 0
17	R	Int_nExtDetectorTiming_status	1: the EUDRB is a Detector Timing master 0: the module "**MIMODetectorTimingUnit**", synchronizes its signals to a SYNCH signal coming

			from a “TIMING MASTER” EUDRB Default = 1
18	R	FakeTriggerEnable_status	1: the diagnostic TRIGGER and TRIGGER_RESET are enabled and the TLU ones disabled Default = 0
19	R	nTLU_Intfc_En_status	0: the EUDRB is interfacing to the TLU and distributing trigger signals over the custom trigger backplane. Determined by jumper J1.
20	R	nTrgBsyIn_status	0: this is the status of the ACTIVE LOW nBUSY_OC (open collector) line of the trigger backplane.
21	R	Trigger_Busy_TU_status	Busy from the “MIMODetectorTimingUnit” module
22	R	Trigger_Busy_TrigProc_status	OR of all internal sources of Busy: trigger data extraction units, MIMODetectorTimingUnit or MIMOIOControllerUnit
23	R	BusyFromTLU_IFace_status	The OR of Trigger_Busy_TrigProc and the Busy set by the “TLU Interface module”
24	R	Trigger_Busy_IOC_status	Busy from the “MIMO_IOController” (the unit sending data to the VME port) alone
25	R	DetectorNotReady	This bit is cleared after the NIOS-II processor has completed configuration via JTAG of the MIMOTEL/MIMO*2 sensor and sent the “Detector Reset” signal
26	R/W	-	
27	R/W	-	
28	R	FromMCU_ErrCode0	Error code bits returned by the NIOS II in reply to a CMD/DATA operation
29	R	FromMCU_ErrCode1	
30	R	MCUDataReady	‘1’ indicates that the NIOS II MCU has updated the “DataFromMCU” register with the result of a read command
31	W	nUserSpace_BoardReset	Internal signal name: nUserSpace_BoardReset A VME write to the FunctionalCtrl_Stat_Reg (OFFSET = 0x0) with this bit set generates a reset pulse to the entire board. Always reads back as 0.

Table 7.2: description of EUDRB’s main control / status register

The following tables describe meaning and usage of the other resources mapped into the EUDRB's User Addressing space.

SensorInterfaceParam_Reg_01 (OFFSET =**0x20**)			
bit		Name of bit /bit field	Brief description
8..0	R/W	IndexOfLastPixelInRow Example values: 65 for MIMO*2, MIMOTEL 255 for MIMOSA18 511 for MIMOSA5	Maximum value of the lowest 9 bit of the pixel addresss, encoding the position in a row of maximum 512 elements
12.9	R/W	-	
14..13	R/W	ADStrobeDelay Default value: 1	Delay between the AD converter clock and the internal signal strobing the AD converter output
15	R/W	NeedPeriodicReset Default value: 0	Enable this bit for sensors needing a periodic reset such as the MIMOSA5
24..16	R/W	IndexOfLastRow Example values: 127 for MIMO*2 255 for MIMOSA18, MIMOTEL 511 for MIMOSA5	Maximum value of the highest 9 bit of the pixel addresss, encoding the position in a column of maximum 512 elements
31..25	R/W	-	
Example setting for MIMOSA18: (BaseAddress+0x20) <- 0x00FF20FF			

Table 7.3: description of EUDRB's SensorInterfaceParam_Reg_01 register

NOTE: with the EUDRB_MIMOx firmware the format of pixel addressing has changed. A pixel location is identified by its Row and Column indexes which are now mapped in two equally sized bit fields of the PixelAddress.
PixelAddress[8..0] = index of pixel location along a row; range 0 to 511
PixelAddress[17..9] = index of row; range 0 to 511

SensorInterfaceParam_Reg_23 (OFFSET = 0x24)			
bit		Name of bit /bit field	Brief description
11..0	R/W	DetResetWidth Useful range: 0 to 15 Default value: 10	Width of the Reset pulse sent to sensor
15..12	R/W	MAPS_Params	Bit 12 = MIMOSA 5 MKOFF1 Bit 13 = MIMOSA 5 MKOFF2 Bit 14 = MIMOSA 5 OFAST Bit 15 = MIMOSA 5 OSHUT
27.16	R/W	PostDetResetDelay Useful range: full 12 bit Default values: 3 for MIMOTEL, MIMO*2 260 for MIMOSA18	Number of Detector clock cycles to wait, after sending a sensor reset pulse, before start of storing of new samples into the pixel memories
31..28	R/W	-	
Example setting for MIMOSA18: (BaseAddress+0x24) <- 0x0104000a			

Table 7.4: description of EUDRB's SensorInterfaceParam_Reg_23 register

SensorInterfaceParam_Reg_45 (OFFSET = 0x28)			
bit		Name of bit /bit field	Brief description
7..0	R/W	IndexOfLastFrameBeforeReset Useful range: 0 to 255 Default value: 31	number of scan frames (minus one) between periodic resets for sensors needing them. Bit "NeedPeriodicReset" of SensorInterfaceParam_Reg_01 must be set for this parameter to affect the sensor scan
10..8	R/W	DetClkTOADClkDelay Useful range: 0 to 7 Default value: 1	Delay of AD converters clock with respect to Detector clock in units of the FPGA main_clk period (1/90MHz)
11	R/W	-	-
15..12	R/W	DetClkSelect Useful range: 1 to 2 Default value: 1	1 => 15MHz sensor scan frequency 2 => 11.25MHz sensor scan frequency
27.16	R/W	PreDetResetDelay Useful range: full 12 bit Default values: 64	Number of Detector clock cycles to wait before sending a sensor reset pulse
31..28	R/W	ADPipelineLenght Useful range: 0 to 15 Default value: 8	Length of AD pipeline in units of ADC Clock
Example setting for MIMOSA18: (BaseAddress+0x28) <- 0x8040111F			

Table 7.5: description of EUDRB's SensorInterfaceParam_Reg_45 register

NOTE about the detector scan frequency selection: it is presently possible to choose between 2 values only. IF THE SENSOR CLOCK DUTY CYCLE DOES NOT HAVE TO BE STRICTLY 50% then a wider selection range could be implemented.

SensorInterfaceParam_Reg_6 (OFFSET = 0x2C)			
bit		Name of bit /bit field	Brief description
17..0	R/W	QuadrantSizeMinusOne Useful range: 0 to 2^{18}-1 Default value: 8447 for MIMO*2 16895 for MIMOTEL 65535 for MIMOSA18	Size of the subframe - 1
31..18	R/W	-	-
Example setting for MIMOSA18: (BaseAddress+0x2c) <- 0x0000FFFF			

Table 7.6: description of EUDRB's SensorInterfaceParam_Reg_45 register

CommandToMCU (OFFSET = 0x10)			
CommandToMCU register (32 bit) : interface to the NIOS II MicroController Unit (MCU)			
bit		name	Brief description
31..0	W	CommandToMCU	A write to this location sets a flag which is polled by the MCU. When the MCU detects that the flag is set it decodes the command written to this register, executes it and sets the "MCUCommandExecuted" bit in the FunctionalCtrl_Stat register. A write to this location also clears the MCUCommandExecuted flag set by a previous execution.

Table 7.7: description of EUDRB's CommandToMCU register

DataToMCU (OFFSET = 0x14)			
DataToMCU register (32 bit) : interface to the NIOS II MicroController Unit (MCU)			
bit		name	Brief description
31..0	W	DataToMCU	This register contains data to be passed to the MCU routine started by a write to the CommandToMCU register. DataToMCU must be set prior to issuing the MCU command if this requires a parameter.

Table 7.8: description of EUDRB's DataToMCU register

DataFromMCU (OFFSET = 0x18)			
DataFromMCU register (32 bit) : interface to the NIOS II MicroController Unit (MCU)			
bit		name	Brief description
31..0	R	DataFromMCU	This register contains data returned by the execution of an MCU routine. The data is valid only if the the "MCUCommandExecuted" bit in the FunctionalCtrl_Stat register is set.

Table 7.9: description of EUDRB's DataFromMCU register

Description of commands executed by the NIOS II MCU				
Command Name	"CommandToMCU" register contents	"DataToMCU" register contents	"DataFromMCU" register contents	Brief description
Exit_InterruptServiceLoop	0xb0000000	Not required	none	This NIOS II command must be issued to exit from the loop in which the NIOS II is only serving VME and trigger interrupts. The NIOS II returns to the execution of the main MENU loop, waiting for user's input from the RS232 port
SET_NZS_ProcessingEn	0xE0000001	Not required	none	23/07/07: the EUDRB is by default set to transfer data from the pixel SRAMs to the output FIFO via a VHDL-coded module when working in NZS mode.The execution of this command causes the transfer to happen instead under control of the NIOS-II processor (in 736ms for 3 MIMOTEL frames).
CLR_NZS_ProcessingEn	0xE0000000	Not required	none	23/07/07: the EUDRB is by default set to transfer data from the pixel SRAMs to the output FIFO via a VHDL-coded module (in 7ms for 3 MIMOTEL frames).when working in NZS mode.The execution of this command re-establishes this condition.
SynchToTimingMaster_SET	0xD0000001	Not required	none	08/05/07: the EUDRB is by default set to generate the detector timing signals internally. By sending this command the EUDRB synchronizes to the Detector Clock signal coming from a EUDRB Timing Master
SynchToTimingMaster _CLR	0xD0000000	Not required	none	08/05/07: the EUDRB is by default set to generate the detector timing signals internally. By sending this command the EUDRB returns to generating the detector timing signals internally
FakeTriggerEnable_SET	0xF0000001	Not required	none	27/02/07: the internal fake trigger signals (FAKE_TLU_trigger and FAKE_TLU_clear) are now by default routed OUT of the EUDRB and back in through the TLU port to emulate the TLU. If this is not desired the

				fake trigger signals can be routed all internally to the FPGA by setting the FakeTriggerEnable flag
FakeTriggerEnable_CLR	0xF0000000	Not required	none	27/02/07: clear the FakeTriggerEnable flag.
FakeTrig_Generate	0xa0000000	Not required	none	This NIOS II command must be issued to generate a diagnostic trigger to the board in this MIMO*2 test configuration. No matter when this command is issued the trigger pulse will be generated when the detector is being scanned for the fourth time (FRAMENumber=3) after a periodic reset and the Fake trigger event number is always 77.
ClearTrigProcUnits	0xc0000000	Not required	none	This NIOS II command must be issued after acquiring the data from a diagnostic trigger generated by a "FakeTrig_Generate" command and before a new diagnostic trigger can be issued
AutoRstTrigProcUnits_SET	0x70000001	Not required	none	Enable the automatic generation of a "Rst_Trigger_Proc_Units" signal at the falling edge of the "IOC_TrigBusy" signal. DEFAULT: SET
AutoRstTrigProcUnits_CLR	0x70000000	Not required	none	Disable the automatic generation of a "Rst_Trigger_Proc_Units" signal at the falling edge of the "IOC_TrigBusy" signal
uCIsMasterOfSRAM_SET	0x90000001	Not required	none	The FPGA internal signal:"uCIsMasterOfSRAM" is set when this command is issued to the NIOS II MCU. The pixel data memories are put under complete control of the MCU, which can then test them for integrity and/or write the "pedestal" and "Threshold" fields used in the ZeroSuppressed readout mode. **The memory contents are not cleared** by this operation so any **SRAM read** command issued to the MCU **will return the values last written** by the pixel data acquisition machine
uCIsMasterOfSRAM_CLR	0x90000000	Not required	none	The FPGA internal signal: "uCIsMasterOfSRAM" is

				cleared. The pixel data acquisition machine is restarted and the pixel data memories are updated with the new sample values.
SRAM_Access	0x1WUaaaaa Examples: Write lower bits to SRAM_B at location 0x1234: 0x18041234 Read upper bits from SRAM_D at location 0x1234: 0x101C1234	Not required	none	The command code for accessing the pixel SRAM via the NIOS II MCU can be evaluated considering that: - W is the hex digit 8 for a write operation and 0 for a read - U is the hex digit 1 for accessing the upper 24 bits of a pixel data location and the hex digit 0 for accessing the lower 24 bits - aaaaa is a 20 bit address where the 2 MSBs select the target bank of memory (0 for bank A, 1 for bank B etc.) and the remaining 18 bits select the location within the target memory In case of a **WRITE** access, the desired SRAM **data** must be written to the location "**DataToMCU**" (OFFSET = 0x14) **BEFORE issuing the "SRAM_Access" command.** In case of a **READ** type of "SRAM_Access", the returning **data can be read** at the VME location "**DataFromMCU**" (OFFSET = 0x18)
MIMOStar/MIMOTEL Config	0x4WTMiidd Examples: Select the MIMOTEL sensor: 0x4C010000 Write 1 to the "TestEnable" bit (bit0) in the RO_MODE set of 1bit registers: 0x48200001 Write 100 to the "Test1" 8bit register (index=11) in the set of BIAS_DAC registers: 0x48100B64	Not required	none	The command code for changing the MIMO*2 / MIMOTEL configuration parameters and downloading them to the chip can be evaluated considering that: - **W** is the hex digit: C for selecting the sensor 8 when setting a parameter (**SET**) or when sending Hard/Soft Resets or when downloading parameters to the sensor 4 for **reading the parameter value from the NIOS II memory** 0 for **reading the parameter value returned from the chip after downloading (RdBck)** - **T** selects the type of MIMO*2 / MIMOTEL target register for this operation:

	Copy the value of register “Test1” readback from the chip via JTAG to the “**DataFromMCU**” (OFFSET = 0x18) location in the VME address space			T=1 for the BIAS_DAC reg.s (14*8bits) T=2 for the RO_MODE reg. (6*1bits) T=3 for the BSR_PIN reg. (10*1bits) T=4 for the DIS_COL reg. (128*1bits) T=A -> JTAG HARD reset T=B -> for loading the images, in NIOS II memory, of the sensors’ JTAG registers with a set of default parameters T=C -> JTAG SOFT reset T=D : it causes the **download** of the parameters to the chip via JTAG - **M** selects whether the MIMOTEL (M=1) or the MIMO*2 (M=2) or the MIMOSA18 (M=3) is to be connected to the EUDRB - **ii** is the index to the target register (see MIMO*2 description for details) - **dd** is the desired data for a **SET** operation The execution of a **RdBck** access copies the data readback via JTAG from the chip into the location “**DataFromMCU**” (OFFSET = 0x18)

Table 7.10 : description of commands executed by the NIOS II MCU

7.3 EUDRB Event data format – Non Zero Suppressed mode

The pixel memory is built for capturing at least three consecutive frames of digitized (12bit) pixel voltages.

While acquiring data in Non Zero Suppressed mode the EUDRB equipped with the EUDRB_MIMOx firmware sends out two samples per pixel: the last sample before trigger arrival and the sample after trigger arrival.

The format of the data block for the NonZeroSuppressed mode follows the one proposed by Davide Spazian (Univ. di Ferrara) and implemented for the MIMOSA configuration of the EUDRB. Data for the two pixel samples are interspersed, due to the need of transferring all the information from the pointed SRAM cells before incrementing the pointer to get the next pixel information. The 64-bit words transferred over the VMEBus are organized as described below:

64bit word #00000 : **HEADER**

64bit word #00001 : **N-1**_C00_R000_**D, N-1**_C00_R000_**C, N-1**_C00_R000_**B, N-1**_C00_R000_**A**

64bit word #00002 : **N**_C00_R000_**D, N**_C00_R000_**C, N**_C00_R000_**B, N**_C00_R000_**A**

64bit word #00003 : **N-1**_C01_R000_**D, N-1**_C01_R000_**C, N-1**_C01_R000_**B, N-1**_C01_R000_**A**

64bit word #00004 : **N**_C01_R000_**D, N**_C01_R000_**C, N**_C01_R000_**B, N**_C01_R000_**A**

….

64bit word #.....: **N-1**_Clast_Rlast_**D, N-1**_Clast_Rlast_**C, N-1**_Clast_Rlast_**B, N-1**_Clast_Rlast_**A**

64bit word #.....: **N**_Clast_Rlast_**D, N**_Clast_Rlast_**C, N**_Clast_Rlast_**B, N**_Clast_Rlast_**A**

64bit word #.....: **TRAILER**

where:

N?_C00_R000_**?** are 16bit words in which:

N?_C00_R000_?<**11..0**> is the 12bit unsigned ADC reading of the pixel voltage.

N?_C00_R000_**?**<**13..12**> is 0 for data from frame N-1(BEFORE the trigger), 1 for frame N (after the trigger).

N-1_C00_R000_**X**<**15..14**> identify the channel: 0..3 identify channels A..D

The data block is enclosed between a header and a trailer:

The HEADER is built as:

bits 63..56: header mark = 0xF0

bits 55..40: LocalEventNumber = a local count of the triggers received since last Trigger Reset generated by the TLU

bits 39..32: FrameNumberAtTriggerTime = the frame number is generated by an 8-bit counter reset at detector reset time. this variable contains the value of the FrameCounter at the time of arrival of the trigger

bits 31..28: 0

bits 27..20: ascii "H" = 0x48

bits 19..18: 0

bits 17..0 : PixelAddressAtTrigger = the address of the "pivot' pixel

The TRAILER is built as:

bits 63..56: trailer mark = 0xF1

bits 55..40: TLUEventNumber = the EventNumber received from the TLU

bits 39..32: FrameNumberAtTriggerTime = the frame number is generated by an 8-bit counter reset at detector reset time. this variable contains the value of the FrameCounter at the time of arrival of the trigger

bits 31..28: 0x0

bits 27..20: ascii "T" = 0x54

bits 19..0 : FIFOWordCount = number of 32-bit words written into the output FIFO (including header and trailer)

7.4 EUDRB Event data format – Zero Suppressed mode

While acquiring data in Non Zero Suppressed mode the EUDRB sends out only the information fort the pixels whose voltage, after CDS and pedestal correction, is above a threshold individual to each pixel. Let's call "hit" a pixel above threshold.

The information for each "hit" must then include the coordinates of the pixel, in addition to its "signal" (pedestal corrected CDS). For the MIMOTEL sensor the active matrix is subdivided in four submatrices, each processed by a different channel of the EUDRB. The hit coordinates must then specify the submatrix it belonged to and the position within the submatrix, given with a Column and Row index.

The hit information is coded into a 32 bit word, with the lower 12 bits being the "signal" and the upper 20bits being the address.

The data block then looks like:

64bit word #00000 : **HEADER**

64bit word #00001 : **HitData_1, HitData_0**

64bit word #00002 : **HitData_3, HitData_2**

….

64bit word #00002 : **HitData_N, HitData_N-1**

64bit word #50688: **TRAILER**

where:

HitData_N (31..30) = channel ID: 0 -> ChannelA, 1 -> ChannelB, 2 -> ChannelC, 3 -> ChannelD

HitData_N (29..12) = PixelAddress(17 downto 0);

HitData_N (11…0) = PedestalCorrectedSignal, 12 bit signed representation.

The data block is enclosed between a header and a trailer:

The HEADER is built as:

bits 63..56: header mark = 0xF0

bits 55..40: LocalEventNumber = a local count of the triggers received since last Trigger Reset generated by the TLU

bits 39..32: FrameNumberAtTriggerTime = the frame number is generated by an 8-bit counter reset at detector reset time. this variable contains the value of the FrameCounter at the time of arrival of the trigger

bits 31..28: 0

bits 27..20: ascii "H" = 0x48

bits 19..18: 0

bits 17..0 : PixelAddressAtTrigger = the address of the "pivot' pixel

The TRAILER is built as:

bits 63..56: trailer mark = 0xF1

bits 55..40: TLUEventNumber = the EventNumber received from the TLU

bits 39..32: FrameNumberAtTriggerTime = the frame number is generated by an 8-bit counter reset at detector reset time. this variable contains the value of the FrameCounter at the time of arrival of the trigger

bits 31..28: CDS_FIFOFull_Latch[3..0]

bits 27..20: ascii "T" = 0x54

bits 19..0 : FIFOWordCount = number of 32-bit words written into the output FIFO (including header and trailer)

NOTE: CDS_FIFOFull_Latch[3..0] are the FULL_FLAGs of the internal FPGA FIFOs used in CDS processing in the EUDRB_MIMOx firmware. A ‘1’ flags the the FIFO for the corresponding channel became full during the last trigger processing -> some hit information has been lost. The FIFOs are 1024 words deep.

8 EUDRB Jumper configuration

The EUDRB can be configured to act as the “TLU Interface” (i.e. the only board in the crate to which the TLU is connected through the front panel RJ45 connector) or as a “TLU Slave”. In the latter case the EUDRB gets the trigger signals through the private backplane on a cable segment connecting the uncommitted pins of the VME J2 connector (rows A and C).

The table below describes the settings for the jumpers on the motherboard according to the requested mode of operation of the EUDRB.

On EUDRB_MOBO	For TLU Interface	For TLU Slave
J1	**ON**	**OFF**
J4	**ON**	**ON**
J14	**ON**	**OFF**
J15	**ON**	**OFF**
J16	**OFF**	**ON**
J17	**across pin 2 and 3**	**across pin 1(*) and 2**
J18	**across pin 2 and 3**	**across pin 1 and 2**

(*) pin 1 of the strip of three is the one closer to the back of the board

The EUDRB can be configured to act as the “Timing Master” when it distributes its “Detector clock” and “Detector Reset” signals to the other EUDRBs (“Timing Slaves”) which is connected to.

Jumpers on the EUDRB_DCA (the analog daughter card) determine whether the EUDRB is a “Timing Master” or a “Timing Slave”.

The table below describes the settings for the jumpers on the EUDRB_DCA according to the requested mode of operation of the EUDRB. The jumpers are side by side in pair, with the first pair toward the front of the board.

On EUDRB_DCA	For Timing Master	For Timing Slave
J35,J29	**ON,ON**	**ON,ON**
J36,J30	**OFF,OFF**	**Termination enable (*)**
J34,J27	**OFF,OFF**	**OFF,OFF**
J33,J26	**OFF,OFF**	**OFF,OFF**
J21,J19	**ON,ON**	**OFF,OFF**

(*) Termination enable: jumpers should be ON,ON for the last slave in the chain

The board in the next picture is, for instance configured as a “TLU Slave”, “Timing Slave”.

It may be convenient but not necessary that a “TLU Master” would be also a “Timing Master”.

Fig. 8.1 An EUDRB configured as “TLU Slave”, “Timing Slave”

9 EUDRB external connections

9.1 EUDRB front panel connections for synchronized operation

For EUDRBs to run in synchronized mode, the “Timing Master” must be connected to the “Timing Slaves” through the LEMO connectors at the front panels, as shown in Fig. 9.1.

The picture also shows that for this setup (August 2007 at DESY) an external termination with 50Ohm plugs was adopted.

The LEMO for the “Detector Clock” signal of an EUDRB is at the bottom right, looking at the board as in the picture.

The LEMO for the “Detector Reset” signal of an EUDRB is at the bottom right, looking at the board as in the picture.

The picture shows how the daisy chain among different boards was arranged for that setup.

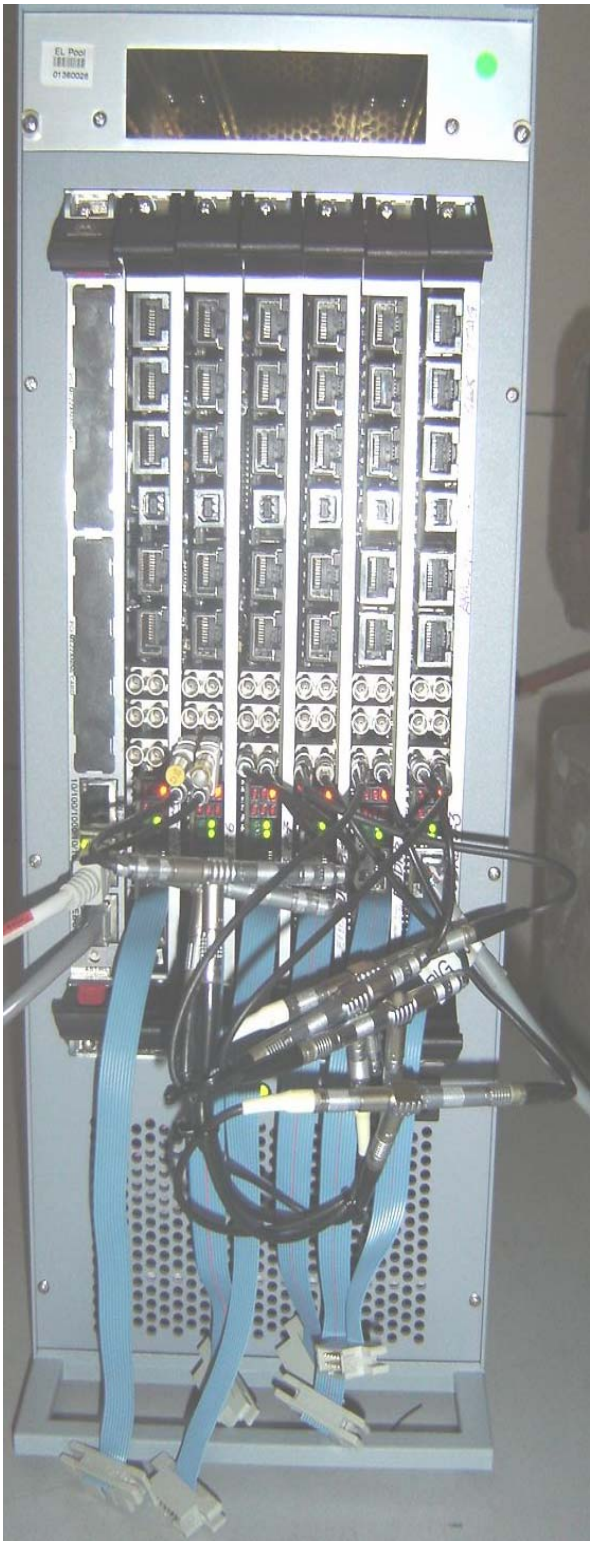


Fig. 9.1 The DAQ crate at DESY with the cables for distributing the synchronization signals across the EUDRBs

9.2 Private bus on cable segment for distributing trigger related information across EUDRBs in the same DAQ crate

To use the EUDRBs in the EUDET-JRA1 telescope a private bus using the free rows of the J2 VME /VME64x bus was designed.

This bus requires only the four lines shown in the schematic below (and the corresponding ground lines): T_TRIG_RST, T_TRIG_TNUM, T_BrdBsy_OCOUT, TNumClk. The picture next page shows the working implementation used for the DESY test beam on a VME64x crate with 7 slots.

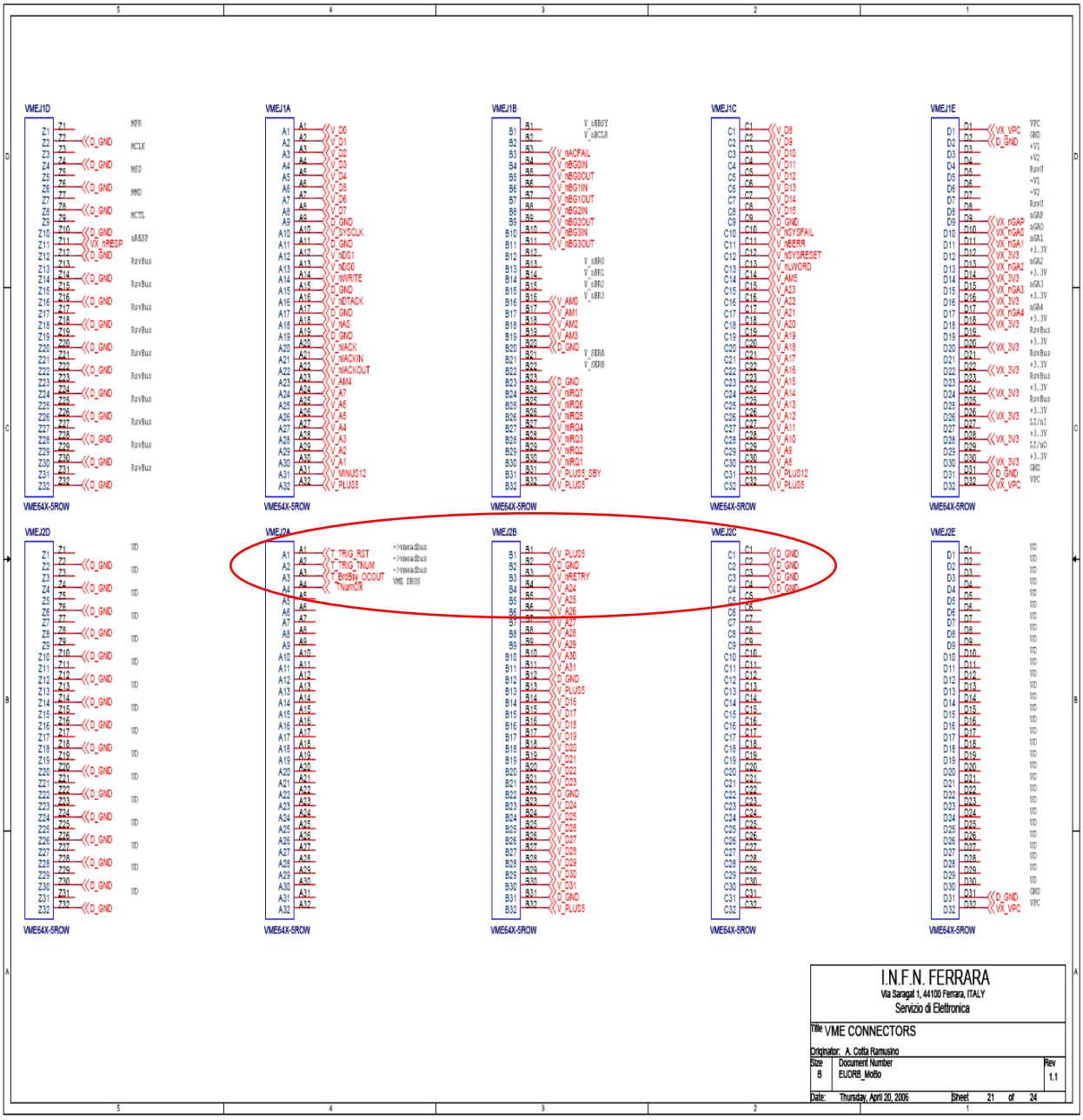


Fig. 9.2 Schematic diagram showing the allocation of the TLU trigger distribution bus on the user-defined pins of the VME J2 connector

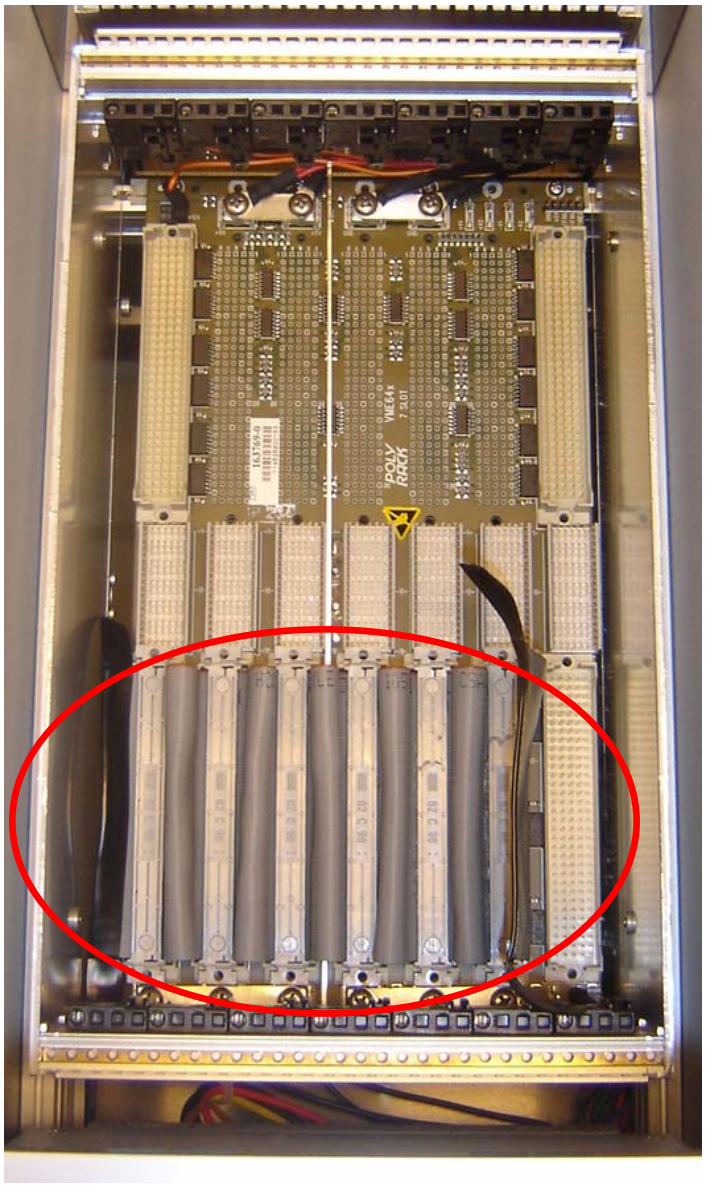


Fig. 9.3 The TLU trigger distribution bus implemented in the DAQ crate for the demonstrator telescope test at DESY, Aug. 2007. The rightmost position is the one occupied by the CPU, not to be connected to the TLU trigger distribution bus.

Conclusion

The demonstator telescope equipped with the EUDRBs has been successfully operated in the test beam at DESY and CERN. The trigger rate at which the telescope DAQ presently runs needs improvement and what can be done at the EUDRB level to this effect is to change the IO module and the trigger processing unit to handle operation in multi-event buffered mode. In this mode the boards would be able to process a new trigger while the readout of the event packet produced by the previous one/ones is/are pending. The readout speed could also be improved by implementing the block read in 2eVME mode, which doubles the peak throughput with respect to MBLT transfers by sending data to the VME CPU on both edges of the “Data Acknowledge” strobe.

Acknowledgement

This work is supported by the Commission of the European Communities under the 6th Framework Programme “Structuring the European Research Area”, contract number RII3-026126.

References

- “ IMAGER 12-bit digitizer-controller card with USB 2.0 connection IMAGER12B_USB2. Technical Documentation Version 0.3”
  IPHC Département Recherches Subatomiques CMOS Group
  23 Rue du Loess, F-67037 Strasbourg Cedex
- “MimoStar 2 Slow Control & CCMOS DAQ”
  CLAUS Gilles, DULINSKI Wojciech, GOFFE Mathieu, Hu Christine, JAASKELAINEN Kimmo, SZELEZNIAK Michal, WINTER Marc
  IPHC (Institut Pluridisciplinaire Hubert-Curien)/DRS (Département de Recherche Subatomiques)
- “ DATA ACQUISITION SYSTEM FOR SILICON ULTRA FAST CAMERAS FOR ELECTRON AND GAMMA SOURCES IN MEDICAL APPLICATIONS (SUCIMA IMAGER) “
  A.CZERMAK†, A.ZALEWSKA B.DULNY, B.SOWICKI, M.JASTRZ_B, L.NOWAK
  H.Niewodnicza_ski Institute of Nuclear Physict, ul.Radzikowskiego 152, Kraków, 31-342, Poland
- “MAPS- DAQ: A proposal for the baseline features of a VME based DAQ card for MIMOSA-V sensors to be used on test beam experiments” JRA1 DAQ workshop, Dec 14 2005 A. Cotta Ramusino, INFN Ferrara
  http://indico.cern.ch/getFile.py/access?contribId=5&sessionId=0&resId=0&materialId=slides&confId=428
- “Sviluppo di una scheda elettronica per l’acquisizione ad alta velocita’ e la riduzione in tempo reale dei dati di un rivelatore di particelle a pixel” Graduation thesis by Davide Spazian. A.A. 2004/2005
- “INFN-EUDRB: A VME based DAQ card for MAPS sensors to be used on test beam experiments. STATUS REPORT” EUDET-JRA1 Review, Apr 04 2006 A. Cotta Ramusino, INFN Ferrara
  http://indico.cern.ch/getFile.py/access?contribId=3&resId=0&materialId=slides&confId=1569
- “EUDRB: A VME-64x based DAQ card for MAPS sensors. STATUS REPORT” JRA-1 Teleconference, Sept 07 2006 A. Cotta Ramusino, INFN Ferrara
  http://indico.cern.ch/getFile.py/access?contribId=2&resId=0&materialId=slides&confId=5693
- “Sistema di trasferimento dati controllato da un Single Board Computer Motorola MVME6100 secondo lo standard VME64x.” Graduation thesis by Lorenzo Chiarelli. A.A. 2005/2006
- “EUDRB: A VME-64x based DAQ card for MAPS sensors. STATUS REPORT” JRA-1 Meeting, Jan 25th 2007 A. Cotta Ramusino, INFN Ferrara
  http://indico.cern.ch/getFile.py/access?contribId=4&sessionId=0&resId=1&materialId=slides&confId=11366
- “EUDRB-MIMO: A VME-64x based DAQ card for MIMOTEL/MIMOSTAR2 sensors: STATUS REPORT” JRA-1 Meeting, June 4th 2007 A. Cotta Ramusino, INFN Ferrara
  http://indico.cern.ch/getFile.py/access?contribId=4&sessionId=0&resId=1&materialId=slides&confId=16769
- “JRA1 – Data acquisition system”, EUDET-Memo-2006-07, D. Haas, E. Corrin, M. Pohl1, Jan. 6 2007
  http://www.eudet.org/e26/e28/e183/e265/JRA1_DAQ_annual2006.pdf
- “Description of the JRA1 Trigger Logic Unit (TLU)”, EUDET-Memo-2007-02, D. Cussans, Jan. 10 2007
  http://www.eudet.org/e26/e28/e182/e291/eudet-memo-2007-02.pdf
- “Testbeam 1 Results” JRA-1 Phone conference, August 21 2007, Antonio Bulgheroni
  http://indico.cern.ch/getFile.py/access?contribId=0&resId=0&materialId=slides&confId=20188
- “First Test Beam Results from the EUDET Pixel Telescope” A. Bulgheroni, EUDET-Report-2007-06, Nov. 19 2007
  http://www.eudet.org/e26/e27/e399/eudet-report-2007-06.pdf