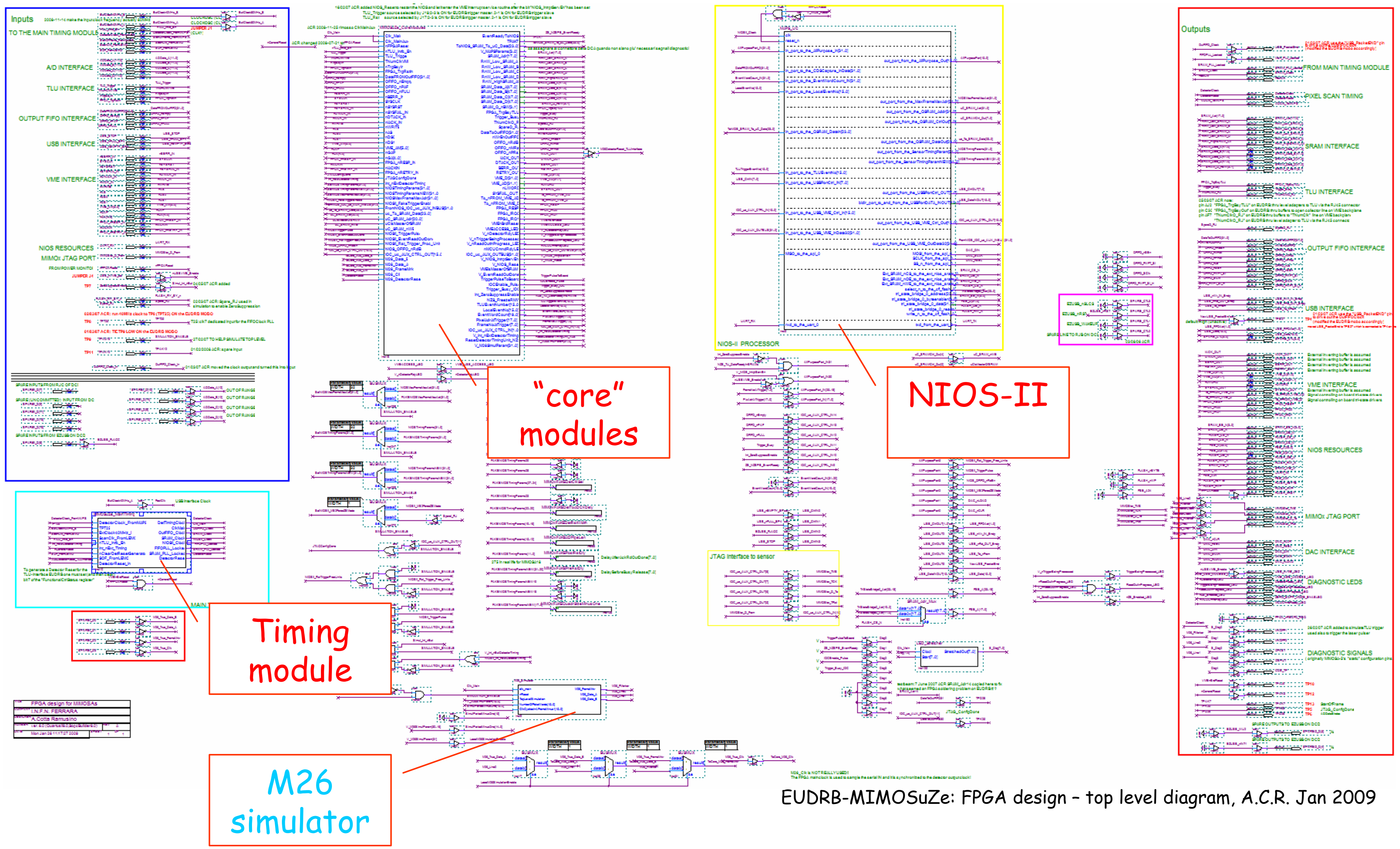


EUDRB-MIMOSuZe: FPGA design – top level diagram, A.C.R. Jan 2009

Progress of the readout board: EUDRB-MIMOSuZe

Progress of the readout board: EUDRB-MIMOSuZe

Summary:

- "EUDRB-MIMOSuZe" features

- overview of the "EUDRB-MIMOSuZe" FPGA design
    - the "M26_Simulator" module used to test the "EUDRB-MIMOSuZe" design
    - the "MimoSuze_MainTiming" module
    - the "MimoSuzeIntBufferWr" module: M26 acquisition and storage controller
    - the "MIMOSuZeIOController" module: VME interface

- "EUDRB-MIMOSuZe" performance figures

Progress of the readout board: EUDRB-MIMOSuZe

"EUDRB-MIMOSuZe" features:

✓ it connects to an M26 sensor through a "MIMOTEL" level adapter

✓ the FPGA features an embedded M26 simulator operating in mode 0 (two channels @ 90 MHz) to exercise the "EUDRB-MIMOSuZe" firmware

✓ the M26 interface operates at up to 90MHz (mode 0 operation tested by looping the simulated M26 ouput signals through a modified MIMOTEL level adapter to include the EUDRB hardware in the test)

✓ overlapping INPUT (acquisition of M26 frames) and OUTPUT (VME readout) operations

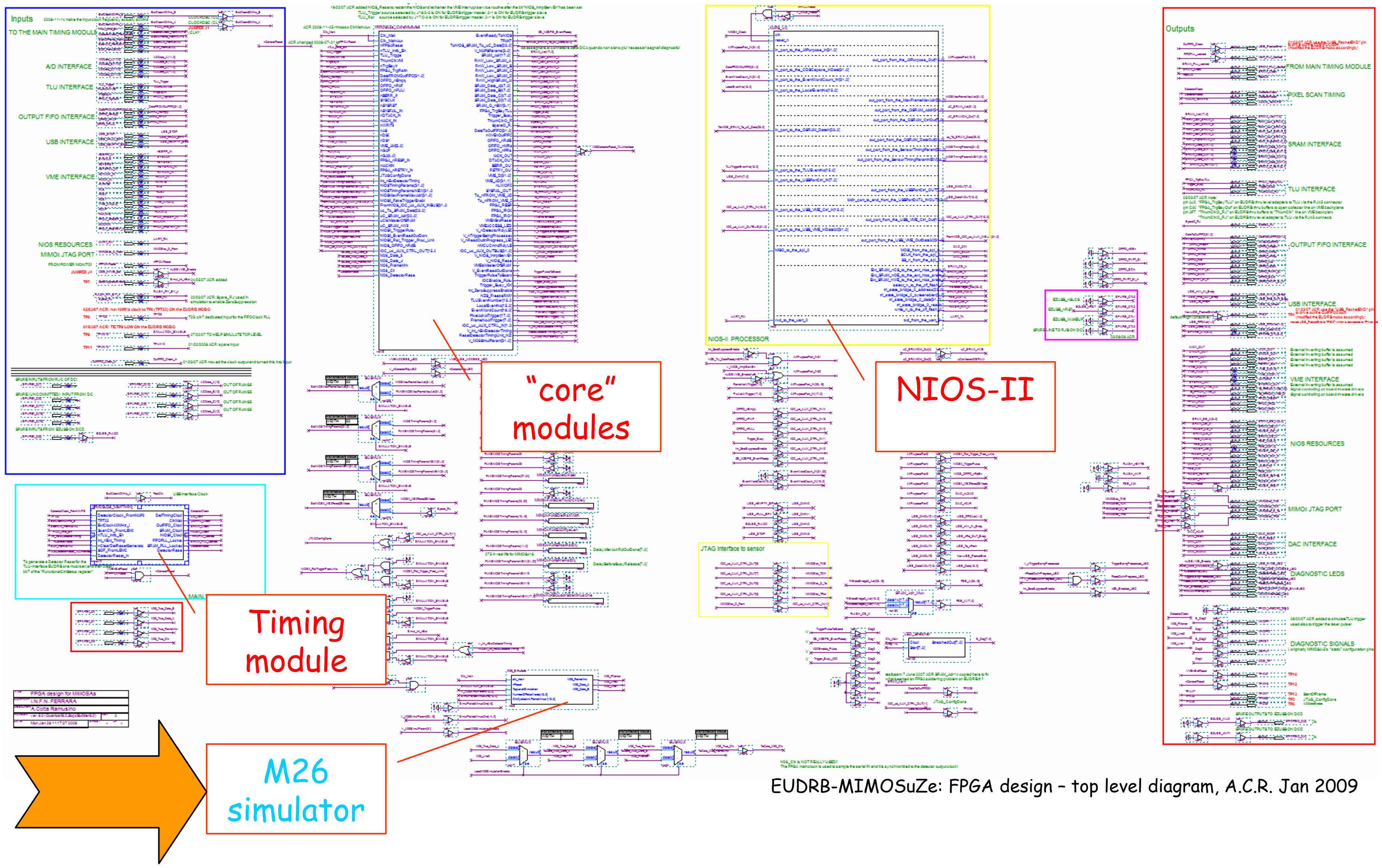
✓ generation of an interrupt to the VME CPU when the event data is ready to be read out

✓ 2e-SST block transfer (> 100MB/s burst rate)

✓ leading word count in the output event data block

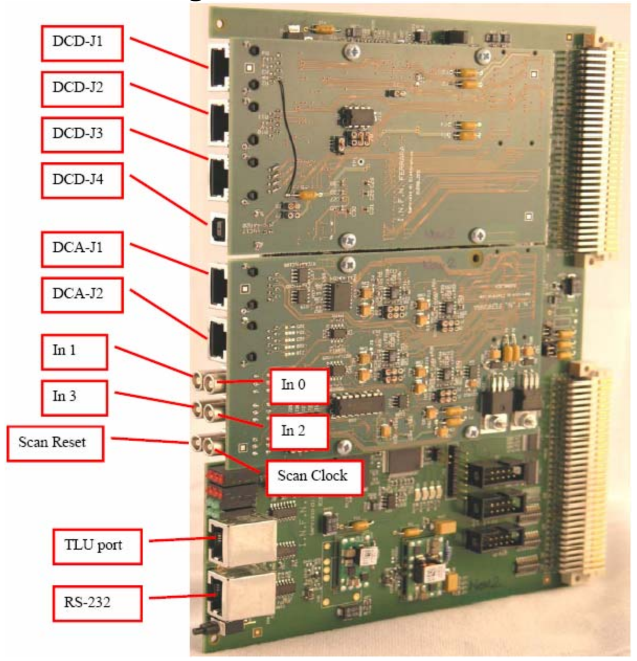
✓ "TLU_Interface" module reviewed to prevent triggering on constant levels on the "trigger/trigger_number" line from the TLU (which occurs after 2^{16}-1 triggers if only 16 bits of the 32-bit TLU trigger counter are readout).

Progress of the readout board: EUDRB-MIMOSuZe: overview of the "EUDRB-MIMOSuZe" FPGA design



EUDRB-MIMOSuZe: FPGA design – top level diagram, A.C.R. Jan 2009

Progress of the readout board: EUDRB-MIMOSuZe: overview of the "EUDRB-MIMOSuZe" FPGA design



"M26_Simulator" output (single ended from "DCD-J1") converted to LVDS and fed back to the LVDS receivers through the "DCD_J3" connector. Stable operation observed with clock frequency of 90MHz.

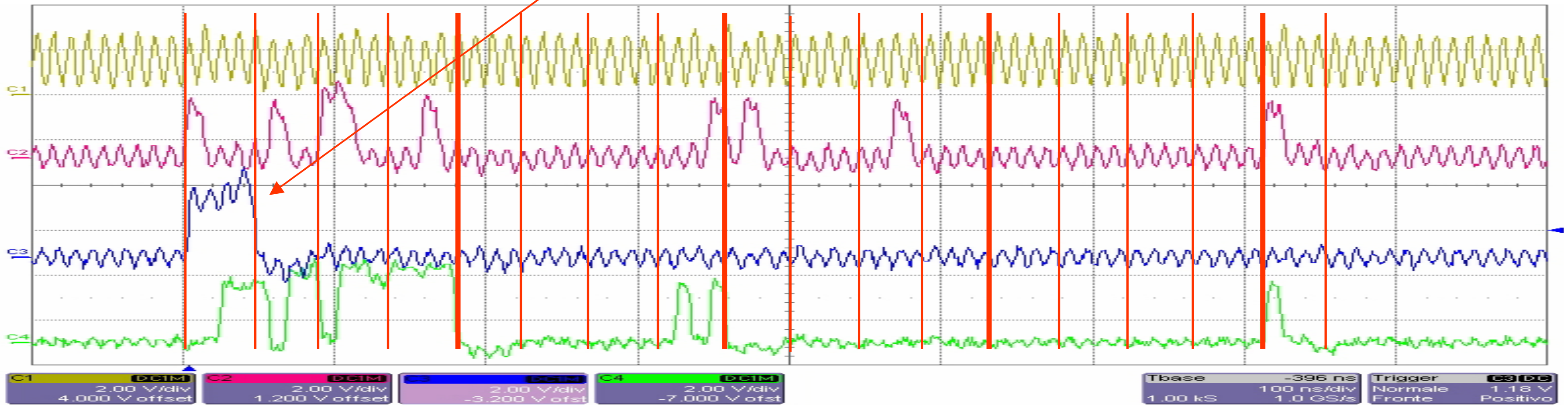
the waveforms below show the start of one M26 data packet.

I am sending:

Data_0: 0x4321, 0x8000, 0x0402(*), 0x0000, 0x0001, 0x0002, …
0x01fe, 0x01ff, 0x7654

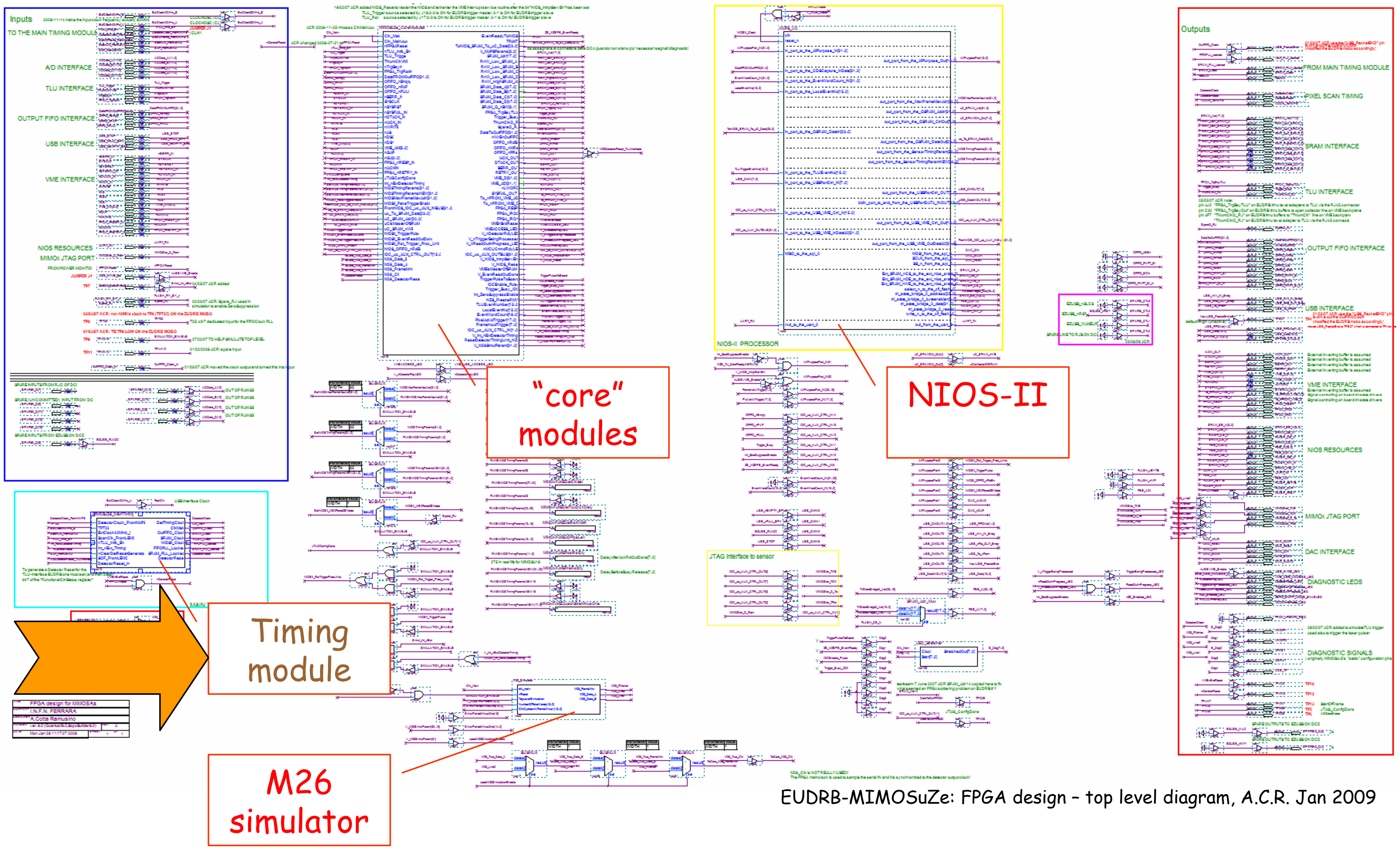
Data_1: 0xFEDC, 0xA000, 0x0000(*), 0x0000, 0x0001, 0x0002, …
0x01fe, 0x01ff, 0xDCBA

(*) number of 16-bit words in the rest of the packet = 2 * 200(Hex) + 2 words for the trailer



the waveforms refer to single-ended signal picked up (with a 500MHz bandwidth scope) at the modified "MIMOTEL level adapter" used to convert the single-ended M26 simulator out into LVDS and loop them back to the EUDRB.

Progress of the readout board: EUDRB-MIMOSuZe: overview of the "EUDRB-MIMOSuZe" FPGA design



EUDRB-MIMOSuZe: FPGA design – top level diagram, A.C.R. Jan 2009

Progress of the readout board: EUDRB-MIMOSuZe: overview of the "EUDRB-MIMOSuZe" FPGA design

"MimoSuze_MainTiming" module:

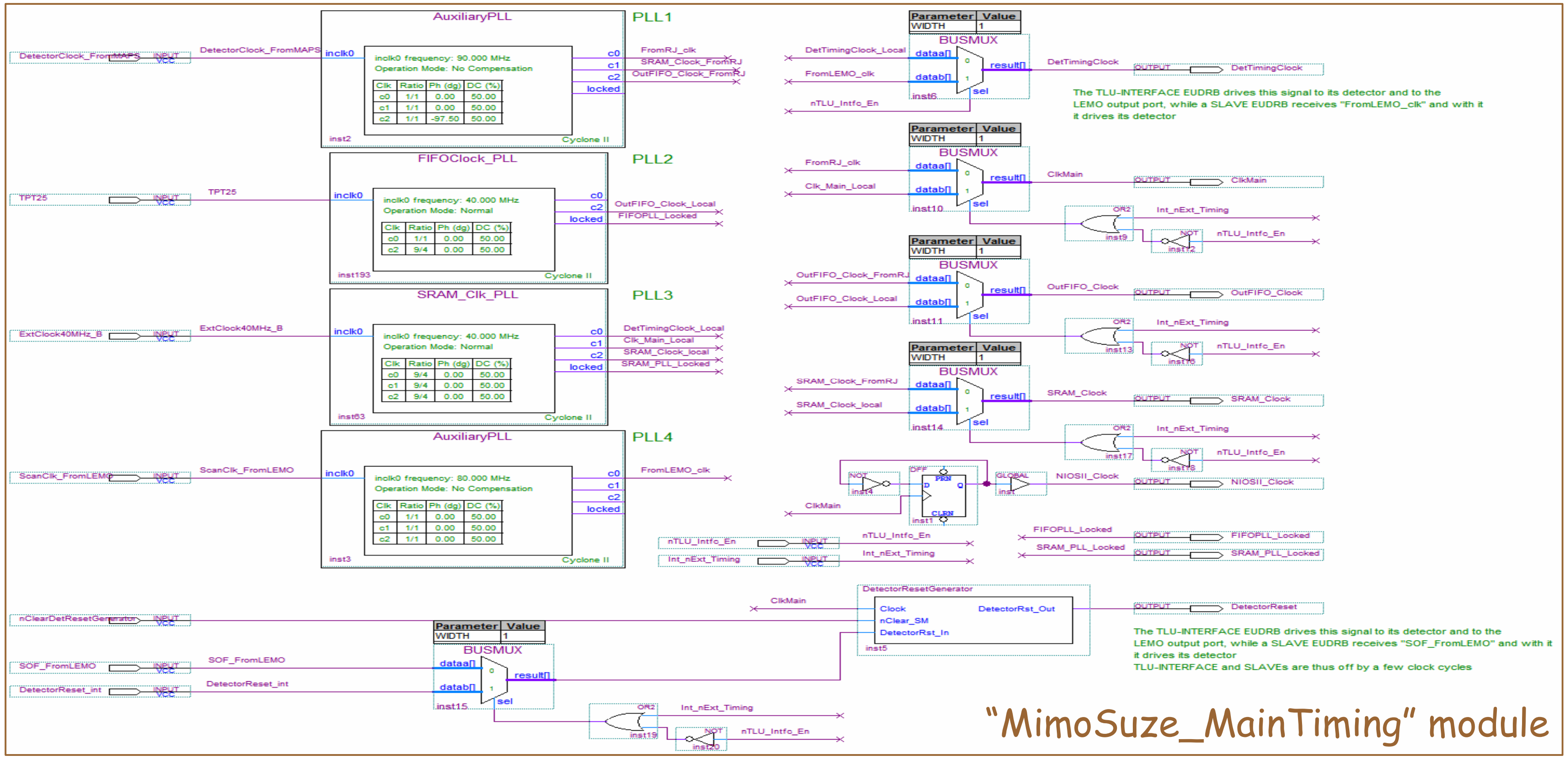
• the "Detector" clock sent to the M26 sensor is derived from:

- the local oscillator for the "Timing Master" EUDRB
- the LEMO input for the "Timing Slave" EUDRB

  (the LEMO "in" connector on the "timing slave" should be connected to the LEMO "output" connector of the "timing master" EUDRB)

• the main EUDRB clock, used for all signal processing, is derived from the "ReadOut" clock coming from the M26 sensor

➔ The EUDRBs in the telescope can be "synchronized" all to the "timing master" clock

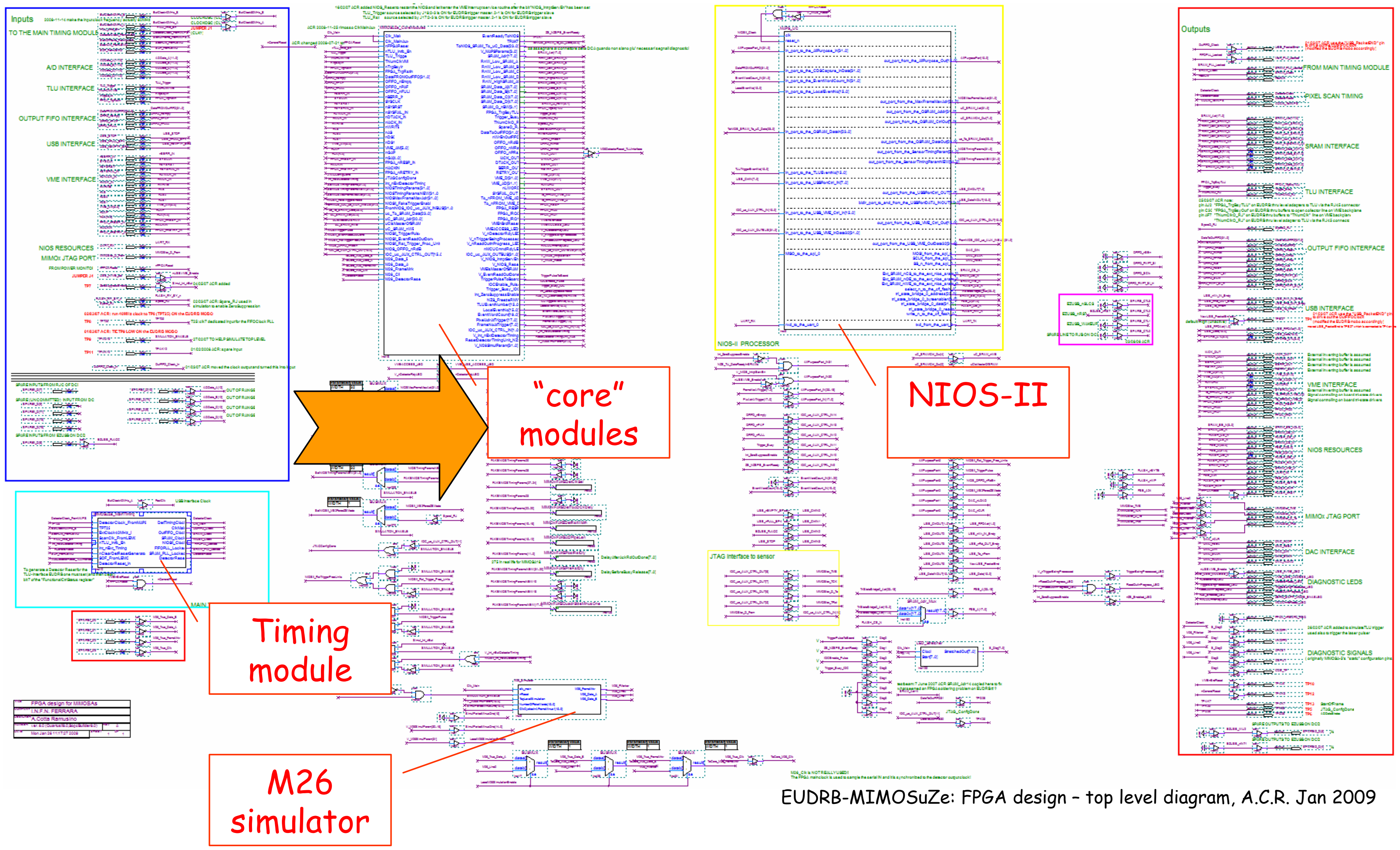


The TLU-INTERFACE EUDRB drives this signal to its detector and to the LEMO output port, while a SLAVE EUDRB receives "FromLEMO_clk" and with it it drives its detector

The TLU-INTERFACE EUDRB drives this signal to its detector and to the LEMO output port, while a SLAVE EUDRB receives "SOF_FromLEMO" and with it it drives its detector
TLU-INTERFACE and SLAVEs are thus off by a few clock cycles

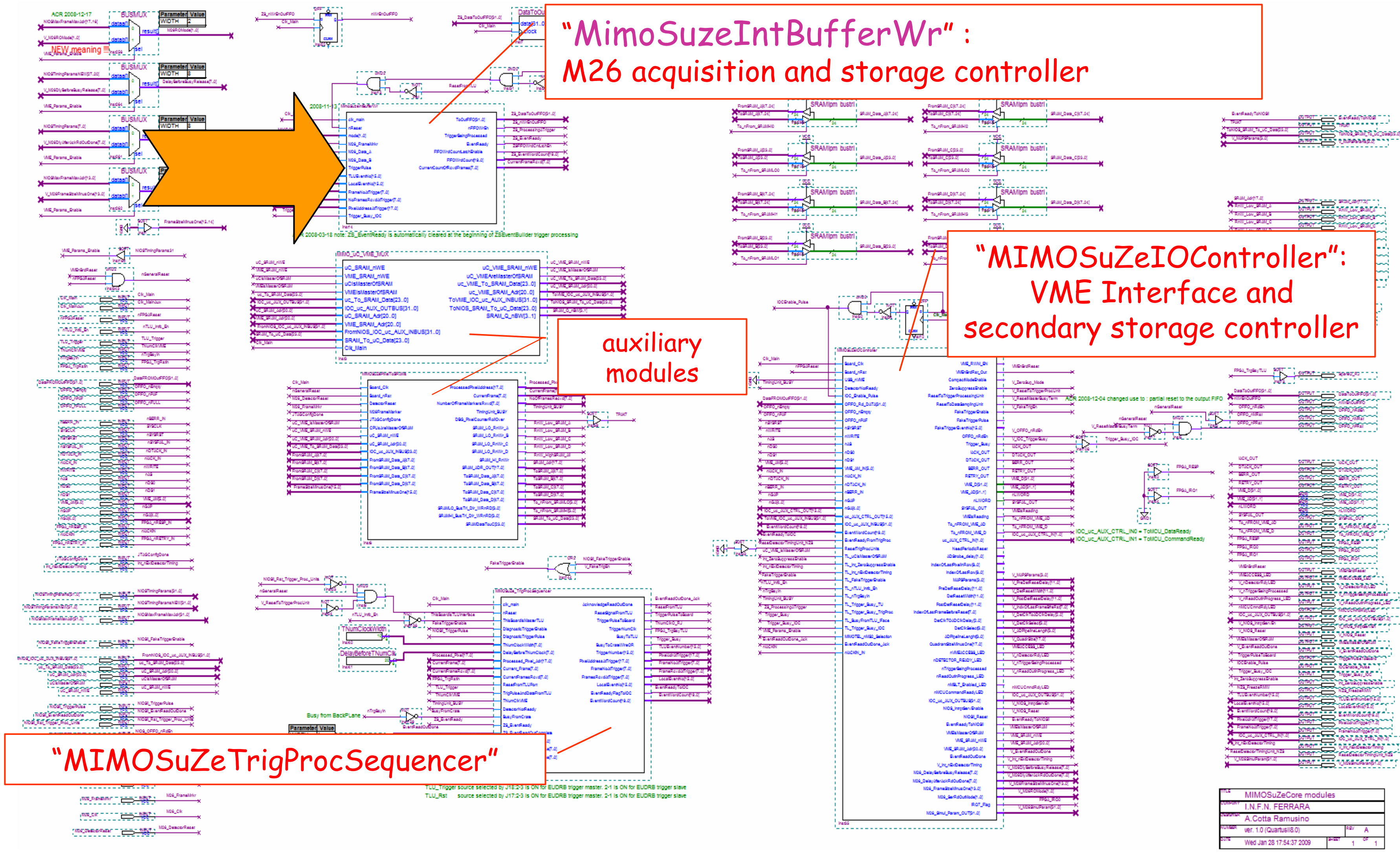
"MimoSuze_MainTiming" module

Progress of the readout board: EUDRB-MIMOSuZe: overview of the "EUDRB-MIMOSuZe" FPGA design



EUDRB-MIMOSuZe: FPGA design – top level diagram, A.C.R. Jan 2009

Progress of the readout board: EUDRB-MIMOSuZe: overview of the "EUDRB-MIMOSuZe" FPGA design



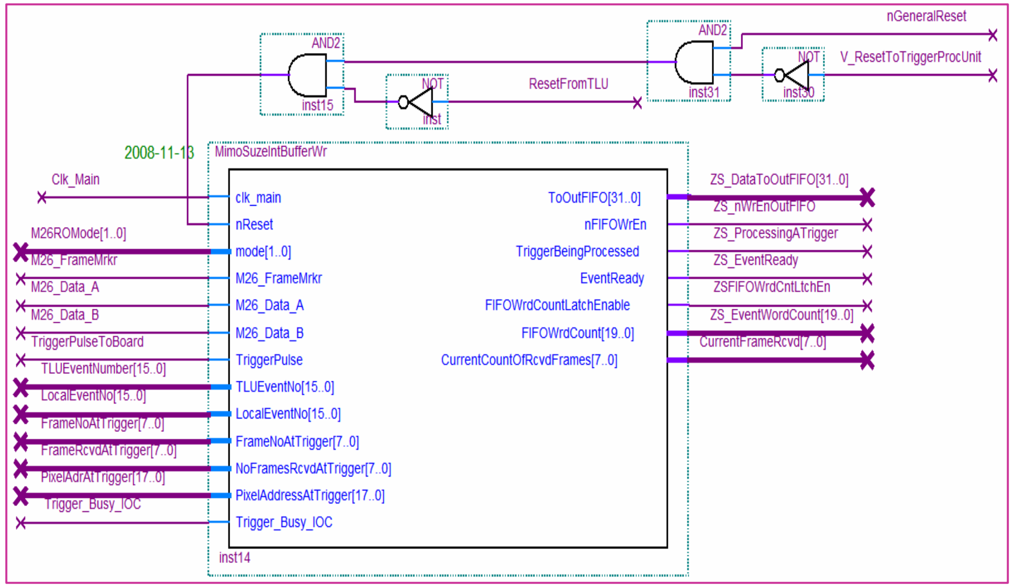
"MIMOSuZe_CoreModules": top level diagram

EUDRB-MIMOSuZe: FPGA design - "MIMOSuZe-CoreModules" block diagram, A.C.R. Jan 2009

"MimoSuzeIntBufferWr" module:

it is the block (coded in VHDL) which receives the serial stream from the M26 sensor and stores in the output FIFO, external to the FPGA, a formatted event data packet in response to a trigger request.

<u>It uses 2 internal FIFOs, 2048 words deep, as a primary storage, which allow INPUT/OUTPUT overlapping</u>



When a trigger is received the "MimoSuzeIntBufferWr" does:

✓ prepare the event packet header:

```
ZSHeaderWord_H(31 downto 24)        <=              X"F0";
ZSHeaderWord_H(23 downto  8)        <=              LocalEventNo;
ZSHeaderWord_H( 7 downto  0)        <=              FrameNoAtTrigger;
ZSHeaderWord_L(31 downto 28)        <=              X"0";
ZSHeaderWord_L(27 DOWNTO 20)<=  X"48"
ZSHeaderWord_L(19 downto 18)        <=              B"00";
ZSHeaderWord_L(17 downto  0)        <=              PixelAddressAtTrigger;
```

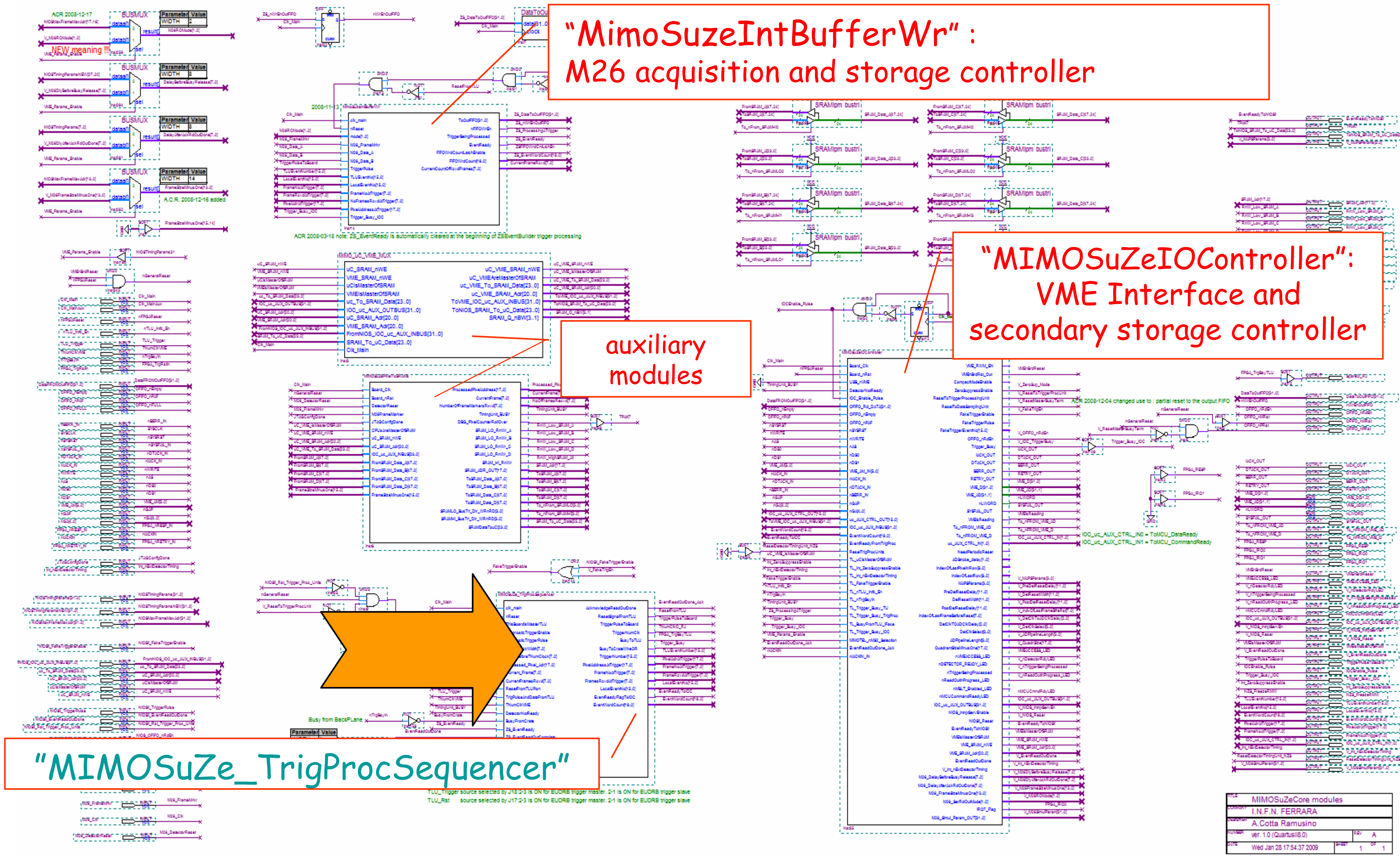
✓ record the data for:

- the frame current at the time of arrival of the trigger
- the following frame

✓ prepare the event packet trailer:

```
ZSTrailerWord_H(31 downto 24)       <=              X"F1";
ZSTrailerWord_H(23 downto  8)       <=              TLUEventNo;
ZSTrailerWord_H( 7 downto  0)       <=              NoFramesRcvdAtTrigger;
ZSTrailerWord_L(31 downto 28)       <=              X"0";
ZSTrailerWord_L(27 downto 20)       <=              X"54";
ZSTrailerWord_L(19 downto  0)       <=
                    STD_LOGIC_VECTOR(UInt_FIFOWrdCountPlusOne); -- to adjust the
wordcount stored in the trailer
```

Progress of the readout board: EUDRB-MIMOSuZe: overview of the "EUDRB-MIMOSuZe" FPGA design

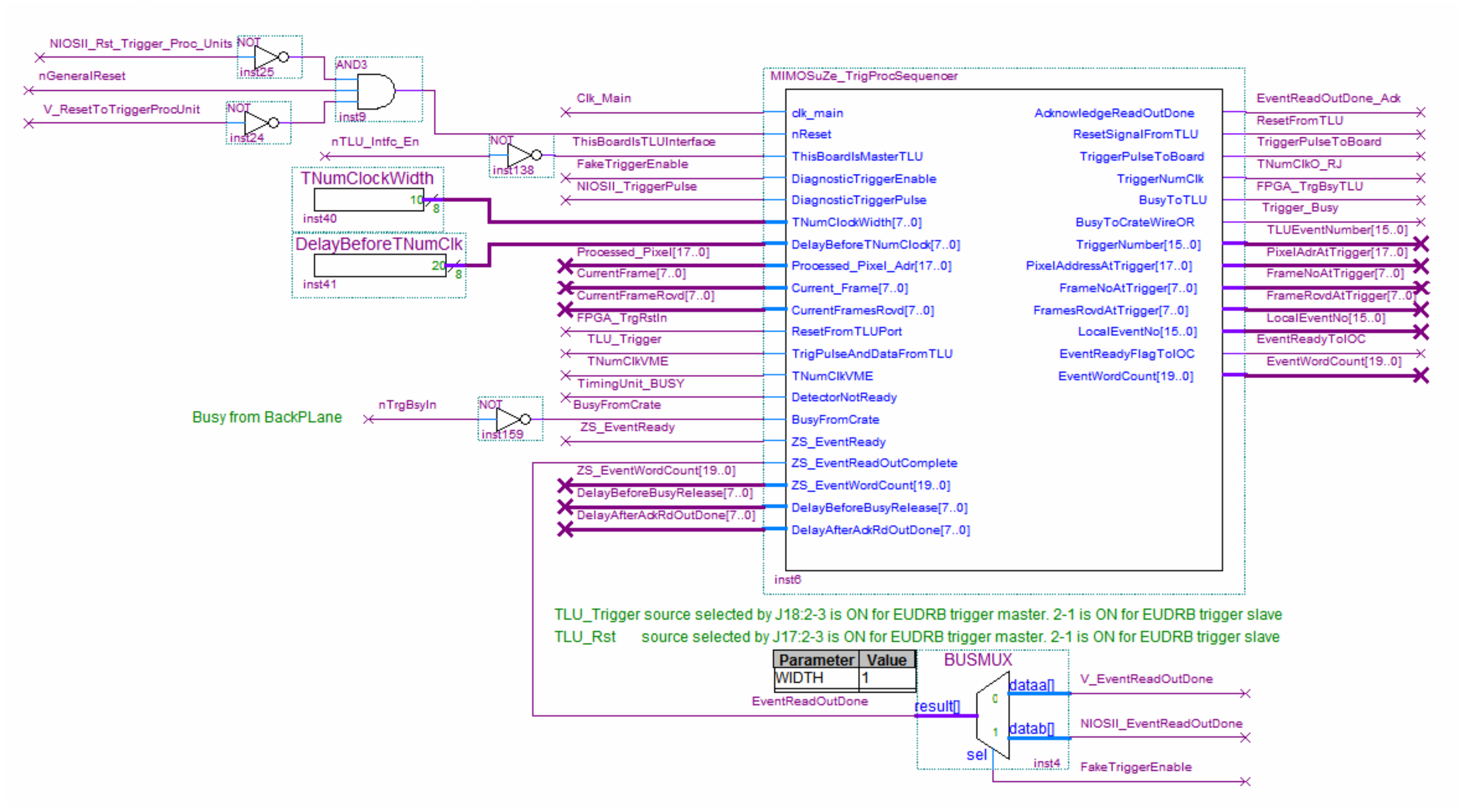

"MimoSuzeIntBufferWr" : M26 acquisition and storage controller
"MIMOSuZeIOController": VME Interface and secondary storage controller
auxiliary modules
"MIMOSuZe_TrigProcSequencer"
MIMOSuZeCore modules
I.N.F.N. FERRARA
A.Cotta Ramusino

"MIMOSuZe_CoreModules": top level diagram

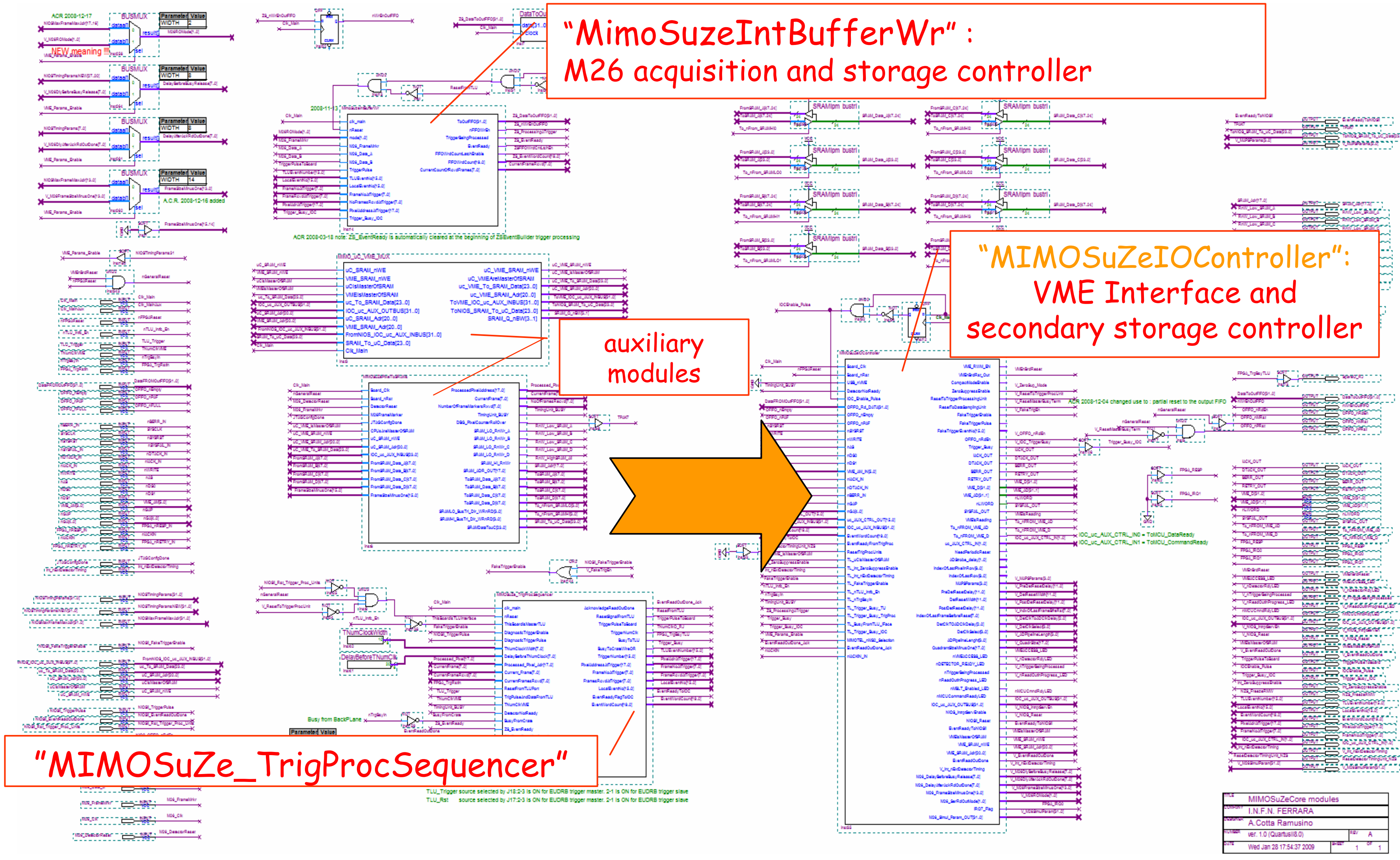
EUDRB-MIMOSuZe: FPGA design - "MIMOSuZe-CoreModules" block diagram, A.C.R. Jan 2009

"MIMOSuZe_TrigProcSequencer" module:

it is the block (coded in VHDL) in charge of scheduling the processing of a trigger from TLU according to the status of the output buffer attached to the VME bus; an intermediate buffer in the "MimoSuzeIntBufferWr" module allows overlapping of trigger processing and output readout).



Progress of the readout board: EUDRB-MIMOSuZe: overview of the "EUDRB-MIMOSuZe" FPGA design



EUDRB-MIMOSuZe: FPGA design - "MIMOSuZe-CoreModules" block diagram, A.C.R. Jan 2009

"MIMOSuZe_CoreModules": top level diagram

Progress of the readout board: EUDRB-MIMOSuZe: overview of the "EUDRB-MIMOSuZe" FPGA design

"MIMOSuZeIOController" module:

it is the block (coded in VHDL) interfacing the EUDRB to the VMEbus.

It also controls the output FIFO (256K deep 32bit wide) on which the event data extracted by a trigger resides.

As soon as the "MimoSuzeIntBufferWr" has finished writing an event packet the "MIMOSuZeIOController" module does:

- generate an interrupt to the VME CPU (only the "TLU Interface" EUDRB should be enabled to do so)
- respond to the subsequent block read request in 2e-SST mode from the VME CPU:

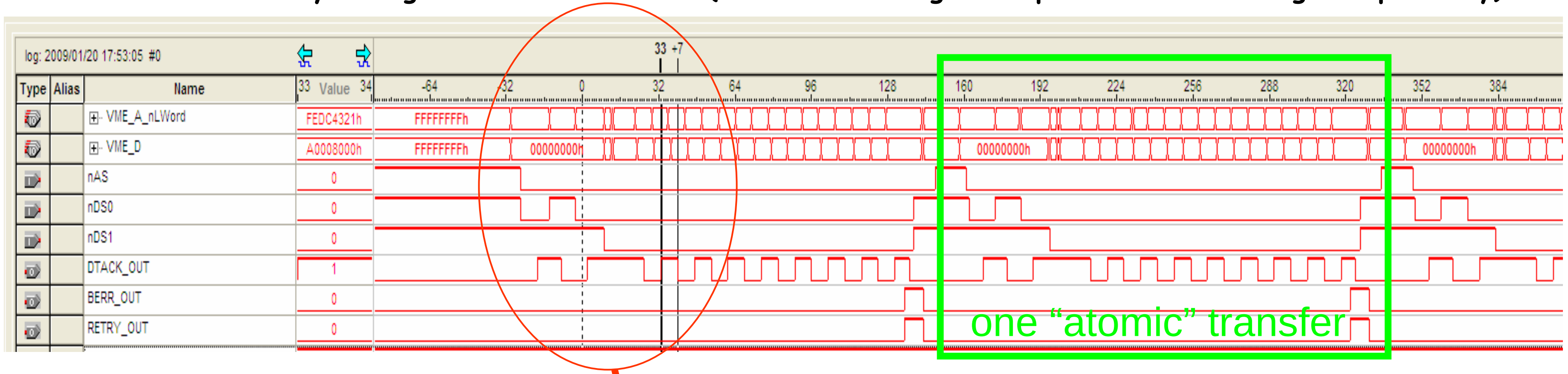
the "MIMOSuZeIOController" prepends the event wordcount (and some auxiliary information such as the firmware version) to the event data packet

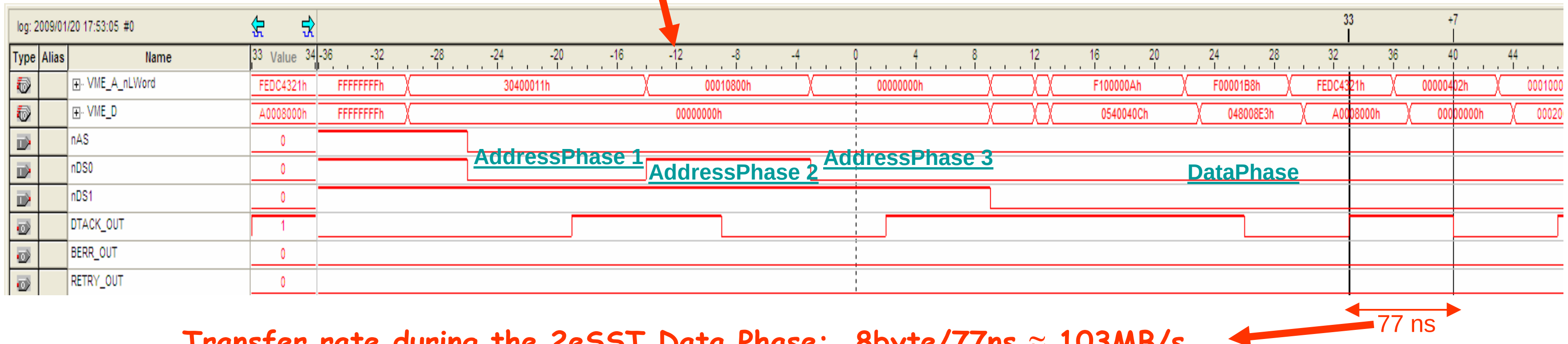
```
BaseAddress: 40400000
1 : 4000040c
2 : fe002601
3 : 4000040c
4 : fe002601
5 : f0000186
6 : 4800631
7 : fedc4321
8 : a0008000
9 : 402
a : 0
b : 10001
c : 20002
d : 30003
```

Partial dump of an event acquired via 2e-SST readout

EUDRB operation frequency: 90MHz; 2eSST block transfer testing : performance and problems encountered

Record of FPGA activity during the 2eSST transfer (internal FPGA signals captured with the "SignalTap" utility):





Transfer rate during the 2eSST Data Phase: 8byte/77ns ≈ 103MB/s

The CPU breaks the 2eSST block transfer into many "atomic" transfers. The VME standard says that the atomic transfers can be made of up to **128** cycles, while the MVME6100 never schedules more than **8** cycles **thus making the transfer inefficient due to the overhead of the 3 address phases at the start of each "atomic" cycle.**

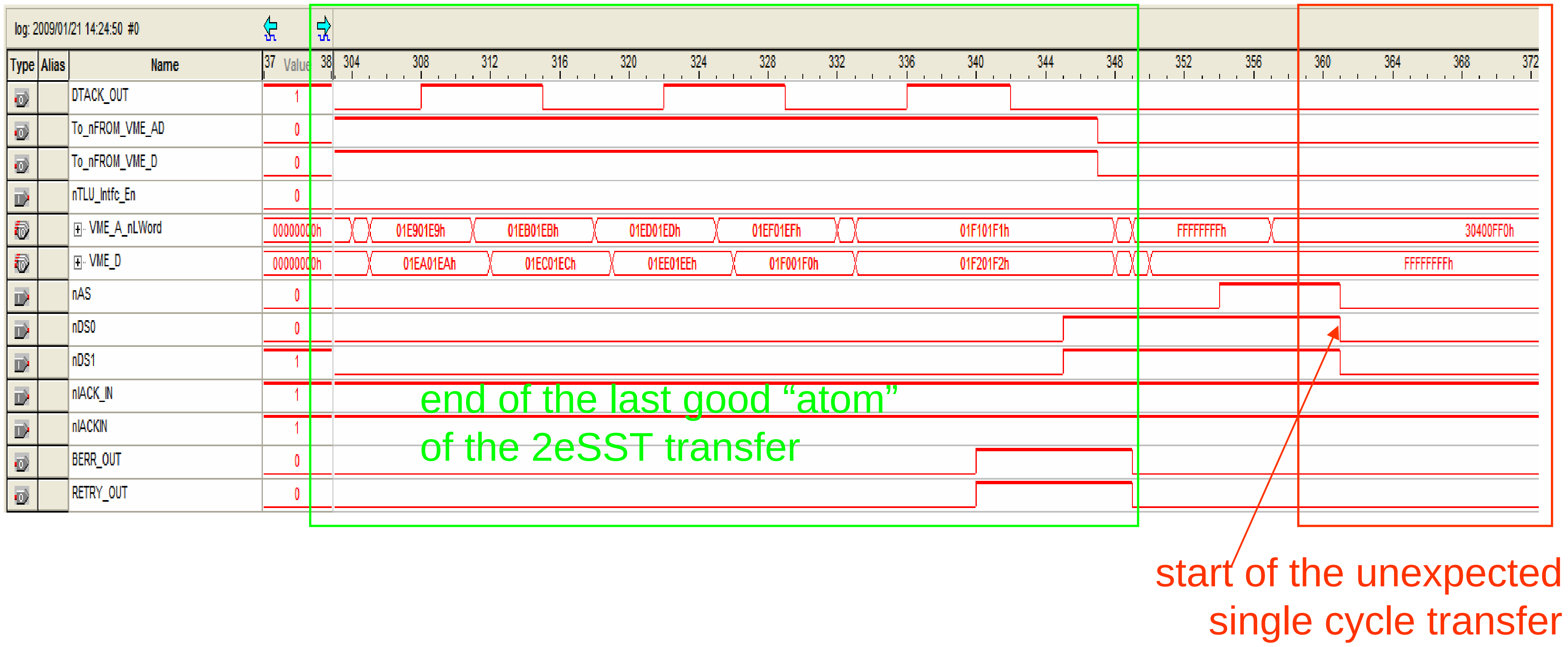
→the Tsi148 driver must be re-written to increase efficiency ?!?

continues…

EUDRB operation frequency: 90MHz; 2eSST block transfer testing : performance and problems encountered

The screenshot below was recorded using the “SignalTap” Logic Analyzer attached to the operating FPGA.

It shows that after the 4096th byte (the boundary is coded into the Tsi148.c driver) transferred, the CPU schedules an unexpected single cycle access to VME address 0x30400FF0 which is not recognized by the EUDRB ➔ the 2eSST transfer never gets the part of the simulated event following 0x01f201f2 and the VMEBus goes into an error state until the Bus Error Timeout (of many ms) expires.



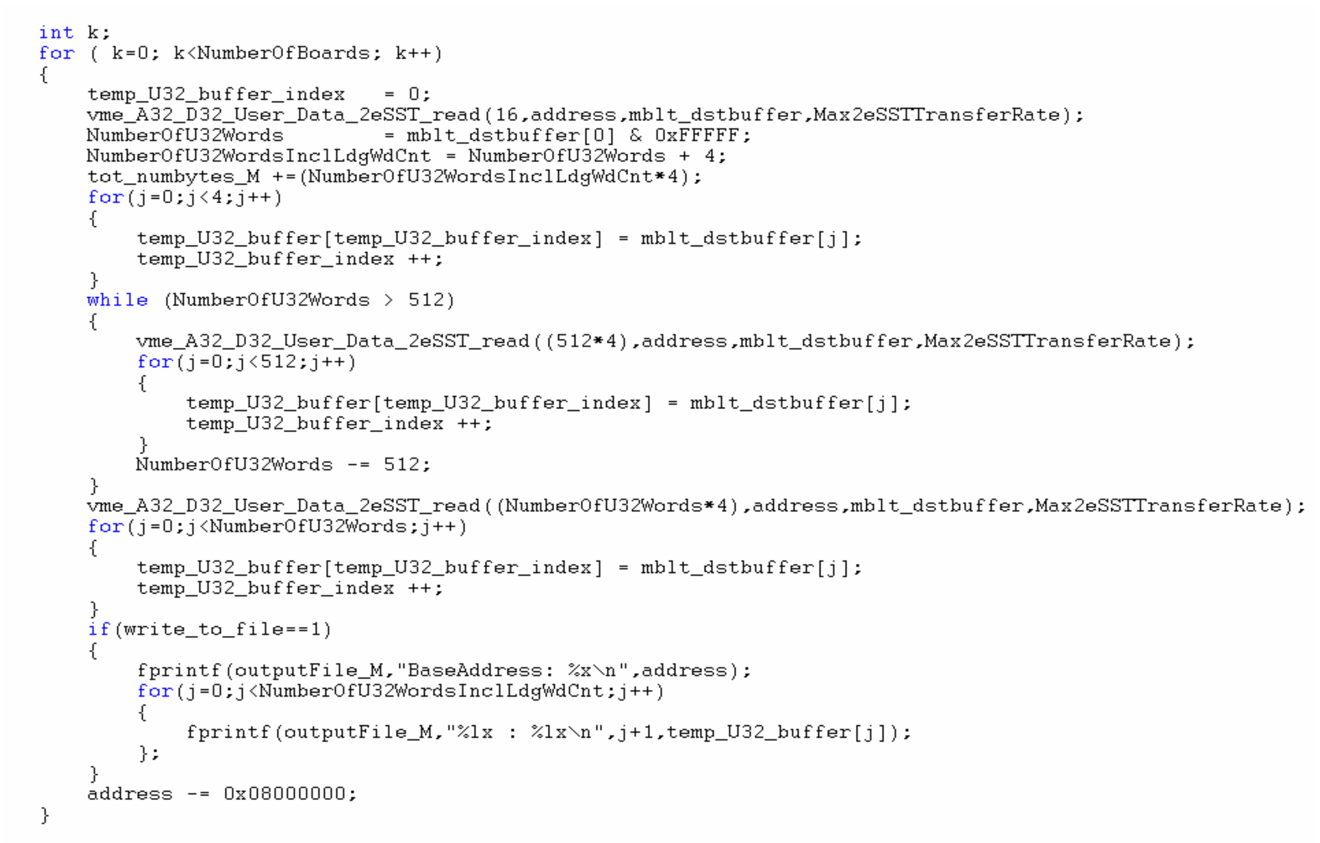
EUDRB operation frequency: 90MHz; 2eSST block transfer testing : performance and problems encountered

Work around to the “4096 boundary” problem:

• write the code so that it reads the event word count with a “1-cycle” 2eSST access

• subdivide the block transfer for the whole event in smaller block transfers for a fixed number N_chunck of bytes <4096 (I used N_chunck = 512, for instance)

• perform a final 2e-SST block transfer for the remaining number of byte < N_chunck



```
int k;
for ( k=0; k<NumberOfBoards; k++)
{
    temp_U32_buffer_index    = 0;
    vme_A32_D32_User_Data_2eSST_read(16,address,mblt_dstbuffer,Max2eSSTTransferRate);
    NumberOfU32Words         = mblt_dstbuffer[0] & 0xFFFFF;
    NumberOfU32WordsInclLdgWdCnt = NumberOfU32Words + 4;
    tot_numbytes_M +=(NumberOfU32WordsInclLdgWdCnt*4);
    for(j=0;j<4;j++)
    {
        temp_U32_buffer[temp_U32_buffer_index] = mblt_dstbuffer[j];
        temp_U32_buffer_index ++;
    }
    while (NumberOfU32Words > 512)
    {
        vme_A32_D32_User_Data_2eSST_read((512*4),address,mblt_dstbuffer,Max2eSSTTransferRate);
        for(j=0;j<512;j++)
        {
            temp_U32_buffer[temp_U32_buffer_index] = mblt_dstbuffer[j];
            temp_U32_buffer_index ++;
        }
        NumberOfU32Words -= 512;
    }
    vme_A32_D32_User_Data_2eSST_read((NumberOfU32Words*4),address,mblt_dstbuffer,Max2eSSTTransferRate);
    for(j=0;j<NumberOfU32Words;j++)
    {
        temp_U32_buffer[temp_U32_buffer_index] = mblt_dstbuffer[j];
        temp_U32_buffer_index ++;
    }
    if(write_to_file==1)
    {
        fprintf(outputFile_M,"BaseAddress: %x\n",address);
        for(j=0;j<NumberOfU32WordsInclLdgWdCnt;j++)
        {
            fprintf(outputFile_M,"%lx : %lx\n",j+1,temp_U32_buffer[j]);
        };
    }
    address -= 0x08000000;
}
```

EUDRB operation frequency: 90MHz; 2eSST block transfer testing : performance and problems encountered

• Testing with ONE EUDRB in the crate:

Interrupt driven readout trigger rate measured with 2eSST transfers: ≈ 4.88kHz.

Very close to the max theoretical limit of 4979Hz (reciprocal of 2 M26 frame scan times @90MHz)

Test conditions:

• one EUDRB

• detector scan clock 90MHz

• input pulse rate to the TLU: 40KHz

• fixed size of the simulated M26 events: 2112 bytes

(decreased simulated event size not to cross the 4096 bytes boundary)

→ Fixed size of 2eSST block read, i.e. no polling done on the EUDRB to know the data size

• average time for acquiring 2000 events (without writing to file) with "Interrupt driven mimoloop": **0.41s**

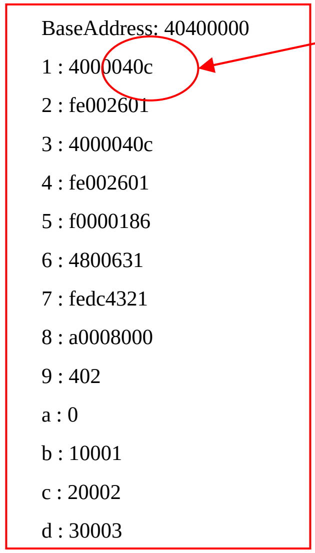
```
-sh-3.00# ./mimoloop -n2000
vmeVirq.waitTime    : 0
vmeVirq.timeOutFlag : 0
vmeVirq.level       : 7
vmeVirq.vector      : 0
Total time 409819 uSec.
-sh-3.00# ./mimoloop -n2000
vmeVirq.waitTime    : 0
vmeVirq.timeOutFlag : 0
vmeVirq.level       : 7
vmeVirq.vector      : 0
Total time 409811 uSec.
-sh-3.00# ./mimoloop -n2000
vmeVirq.waitTime    : 0
vmeVirq.timeOutFlag : 0
vmeVirq.level       : 7
vmeVirq.vector      : 0
Total time 409852 uSec.
```

EUDRB operation frequency: 90MHz; 2eSST block transfer testing : performance and problems encountered

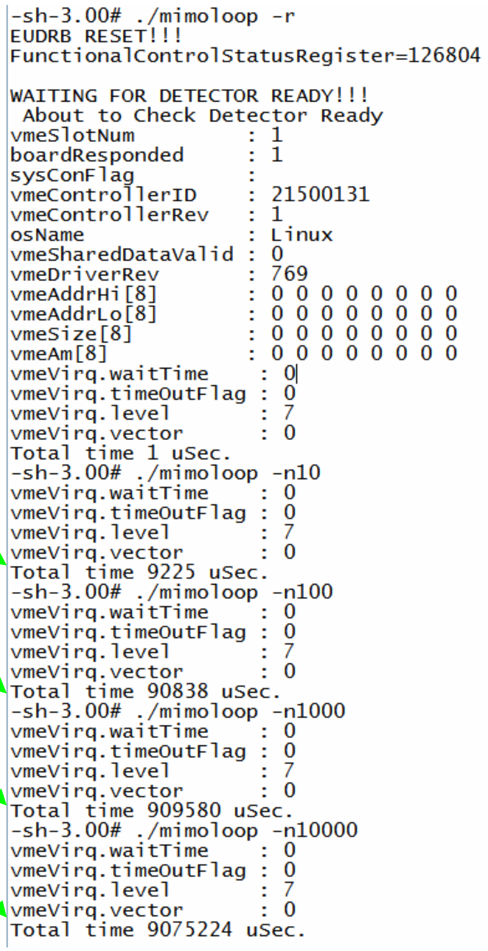
• Testing with THREE EUDRBs in the crate:

Interrupt driven readout trigger rate measured with 2eSST transfers: ≈ 1.1kHz

Test conditions:
- three EUDRBs, 1 Master + 2 Slaves synchronized
- detector scan clock 90MHz
- input pulse rate to the TLU: 40KHz
- fixed size of the simulated M26 events: 4144 bytes



```
BaseAddress: 40400000
1 : 4000040c
2 : fe002601
3 : 4000040c
4 : fe002601
5 : f0000186
6 : 4800631
7 : fedc4321
8 : a0008000
9 : 402
a : 0
b : 10001
c : 20002
d : 30003
```

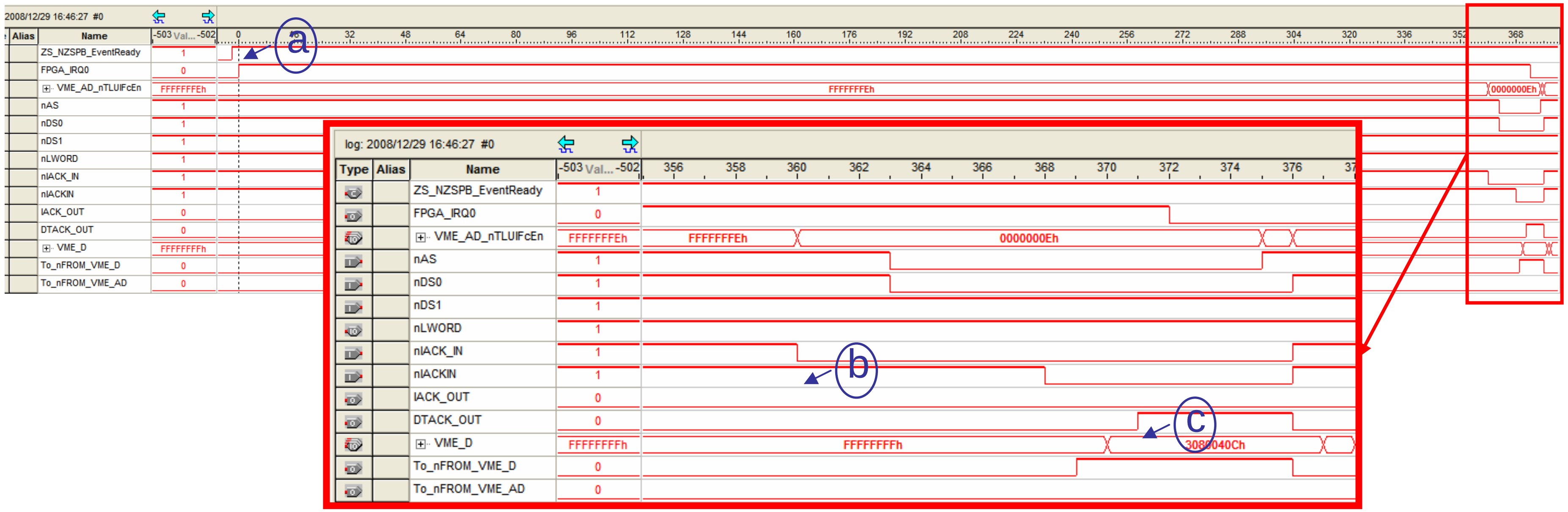


```
-sh-3.00# ./mimoloop -r
EUDRB RESET!!!
FunctionalControlStatusRegister=126804

WAITING FOR DETECTOR READY!!!
 About to Check Detector Ready
vmeSlotNum          : 1
boardResponded      : 1
sysConFlag          :
vmeControllerID     : 21500131
vmeControllerRev    : 1
osName              : Linux
vmeSharedDataValid  : 0
vmeDriverRev        : 769
vmeAddrHi[8]        : 0 0 0 0 0 0 0 0
vmeAddrLo[8]        : 0 0 0 0 0 0 0 0
vmeSize[8]          : 0 0 0 0 0 0 0 0
vmeAm[8]            : 0 0 0 0 0 0 0 0
vmeVirq.waitTime      : 0
vmeVirq.timeOutFlag : 0
vmeVirq.level         : 7
vmeVirq.vector        : 0
Total time 1 uSec.
-sh-3.00# ./mimoloop -n10
vmeVirq.waitTime      : 0
vmeVirq.timeOutFlag : 0
vmeVirq.level         : 7
vmeVirq.vector        : 0
Total time 9225 uSec.
-sh-3.00# ./mimoloop -n100
vmeVirq.waitTime      : 0
vmeVirq.timeOutFlag : 0
vmeVirq.level         : 7
vmeVirq.vector        : 0
Total time 90838 uSec.
-sh-3.00# ./mimoloop -n1000
vmeVirq.waitTime      : 0
vmeVirq.timeOutFlag : 0
vmeVirq.level         : 7
vmeVirq.vector        : 0
Total time 909580 uSec.
-sh-3.00# ./mimoloop -n10000
vmeVirq.waitTime      : 0
vmeVirq.timeOutFlag : 0
vmeVirq.level         : 7
vmeVirq.vector        : 0
Total time 9075224 uSec.
```

Extra slides

•Dec 29th 2008 : implementation of interrupt generation on “EventReady”.



The “MIMOSuZeIOController” module of the “TopLevel_MIMOSuZe” design has been updated to drive the VME INTERRUPT LINE IRQ7 (highest priority) as soon as a new event is ready to be read out.

The above snapshot of the signals captured inside the FPGA by the “SignalTap” logic analyzer shows that:

- the “EventReady” condition triggers (a) the generation the interrupt (FPGA_IRQ0 is mapped onto the VME IRQ7 interrupt request line)
- the MVME6100 has recognized the interrupt (b) and has started an Interrupt Acknowledge Cycle with the proper code (7) on the address lines; the MVME6100 takes 360*12.5ns=4.5us to generate the “Interrupt Acknowledge/Status Read” cycle
- the EUDRB has recognized the Interrupt Acknowledge Cycle, has driven (c) the VME data bus with a 32 bit status word “3080040c” and has driven the VME DTACK line (the MVME6100 is actually programmed now to request an 8 bit Status ID from the interrupter)

EUDRB-MIMOSuZe: A VME-64x based DAQ card for M26 sensors. STATUS UPDATE

•Jan 8th : interrupt driven readout of one (Master) EUDRB.

The MVME6100 CPU uses the Tundra's Tsi148 chip to bridge the VME to the CPU's PCI/x bus. The Tsi148 software driver "vmelinux.c" can be used to check for interrupts from the VME bus, with a call to the system function ioctl :

ioctl(fdCtl, VME_IOCTL_GET_IRQ_STATUS, &vmeVirq);

I have thus modified Chiarelli's "mimoloop.c" to **include the ioctl call and start an MBLT read cycle when a VME interrupt is detected** (I have modified the FPGA firmware to generate an interrupt vector equal to the upper byte of the base address of the board ➔ irq vector = 0x30 for the board under test)

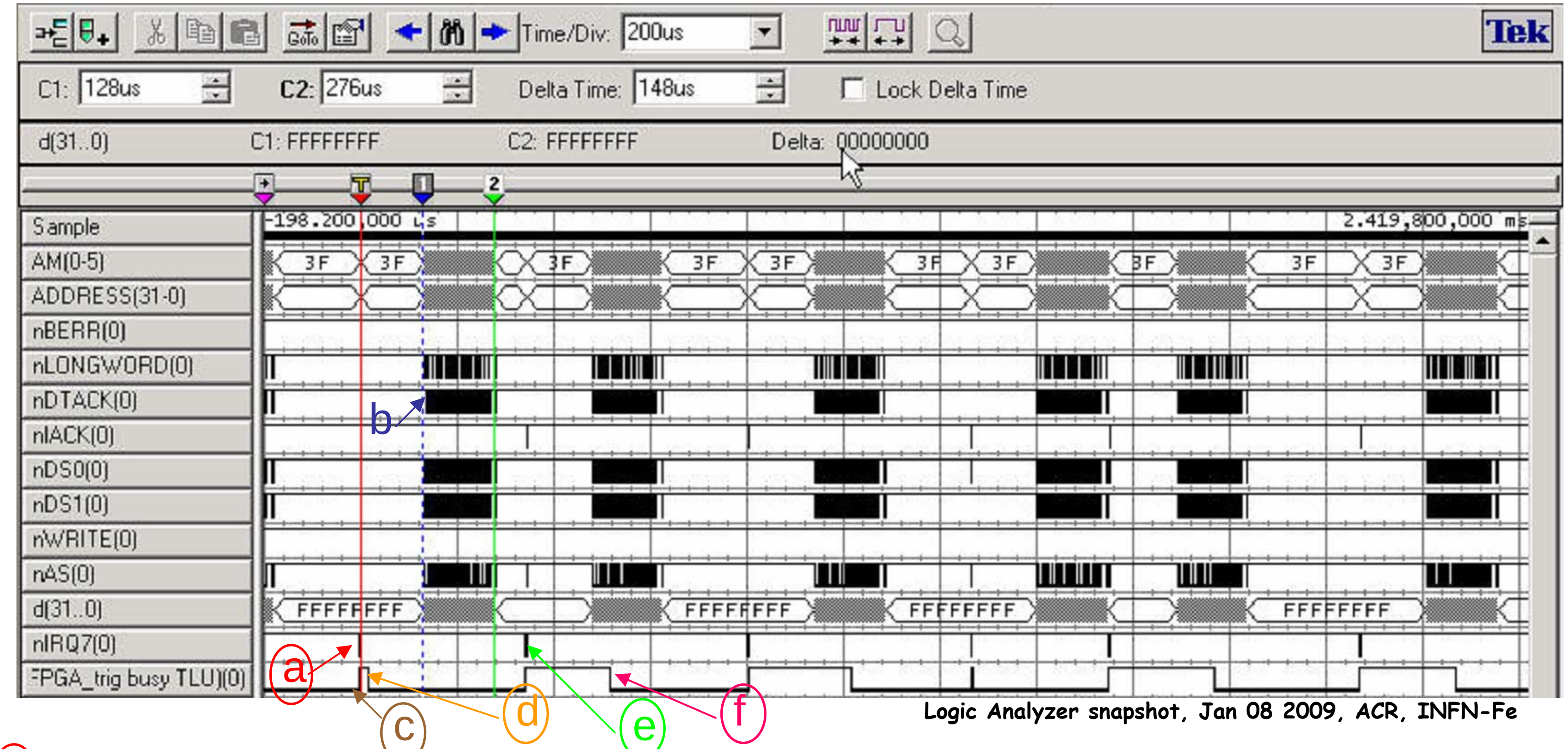
In this test the **number of bytes to read from the EUDRB is fixed ➔ the Single-Cycle-Transfer (SCT) needed to read the wordcount from the EUDRB can be skipped in this test.**

The event size is always 4144 bytes.

```
……….
for (i=0;i<number_of_loops;i++)
{
    WaitOnIRQ:
        memset(&vmeVirq, 0, sizeof(vmeVirq));
        vmeVirq.waitTime = 0;
        vmeVirq.level  = 7;
        vmeVirq.vector      = 0x30;
        CtlStatus = ioctl(fdCtl, VME_IOCTL_GET_IRQ_STATUS, &vmeVirq);
        if (CtlStatus < 0) {
            printf("VME_IOCTL_GET_IRQ_STATUS failed.  Errno = %d\n", errno);
            _exit(1); }
        if (vmeVirq.timeOutFlag==0) {
            vmeVirq.level  = 7;
            vmeVirq.vector      = 0x30;
            CtlStatus = ioctl(fdCtl, VME_IOCTL_CLR_IRQ_STATUS, &vmeVirq);
            if (CtlStatus < 0) {
                printf("VME_IOCTL_CLR_IRQ_STATUS failed.  Errno = %d\n", errno);
                _exit(1);
            }
             vme_A32_D32_User_Data_MBLT_read((number_of_bytes_M*4),address,mblt_dstbuffer);
            if(write_to_file==1)
            {
                for(j=0;j<number_of_bytes_M;j++)
                {
                    fprintf(outputFile_M,"%lx : %lx\n",j+1,mblt_dstbuffer[j]);
                };
            }
            tot_numbytes_M+=(number_of_bytes_M*4);
        }
        else {
            goto WaitOnIRQ;
        }
}
……….
```

EUDRB-MIMOSuZe: A VME-64x based DAQ card for M26 sensors. STATUS UPDATE

•Jan 8th : interrupt driven readout of one (Master) EUDRB: performance



Logic Analyzer snapshot, Jan 08 2009, ACR, INFN-Fe

In a the EUDRB generates an interrupt (IRQ7, active LOW) when a new event has been stored in the output FIFO.

In response to the interrupt, the CPU schedules an MBLT read, which starts about **128us later** b. In the meantime, the "FPGA_trig_busy" (active low) is released (in c) because the M26 event has been saved to the output buffer and a new one can be captured onto the input buffer.

In d the "FPGA_trig_busy" goes LOW (= ACTIVE) again because a new trigger has arrived.

In e the cycle repeats. In f "FPGA_trig_busy" goes LOW (ACTIVE) again: this time the new trigger does not immediately follow the release of the busy: the frequency of the pulser driving the input of the TLU was set at **5KHz** for this test → up to 200us delay between the release of the busy and the arrival of a new trigger pulse can be expected.

EUDRB-MIMOSuZe: A VME-64x based DAQ card for M26 sensors. STATUS UPDATE

•Jan 8th : interrupt driven readout of one (Master) EUDRB: performance

Summary of today's test results:

Test conditions:
- one EUDRB
- detector scan clock 80MHz
- input pulse rate to the TLU: 5KHz
- fixed size of the simulated M26 events: 4144 bytes
- average time for acquiring 20000 events (without writing to file) with "Interrupt driven mimoloop": **9.1s**

Interrupt driven readout rate measured in this setup: ≈ 2.2kHz

```
-sh-3.00# ./mimoloop -n20000
vmeVirq.waitTime    : 0
vmeVirq.timeOutFlag : 0
vmeVirq.level       : 7
vmeVirq.vector      : 0
Total time 9075308 uSec.
-sh-3.00# ./mimoloop -n20000
vmeVirq.waitTime    : 0
vmeVirq.timeOutFlag : 0
vmeVirq.level       : 7
vmeVirq.vector      : 0
Total time 9077115 uSec.
-sh-3.00# ./mimoloop -n20000
vmeVirq.waitTime    : 0
vmeVirq.timeOutFlag : 0
vmeVirq.level       : 7
vmeVirq.vector      : 0
Total time 9076500 uSec.
-sh-3.00#
```

Screen dump, Jan 08 2009, ACR, INFN-Fe

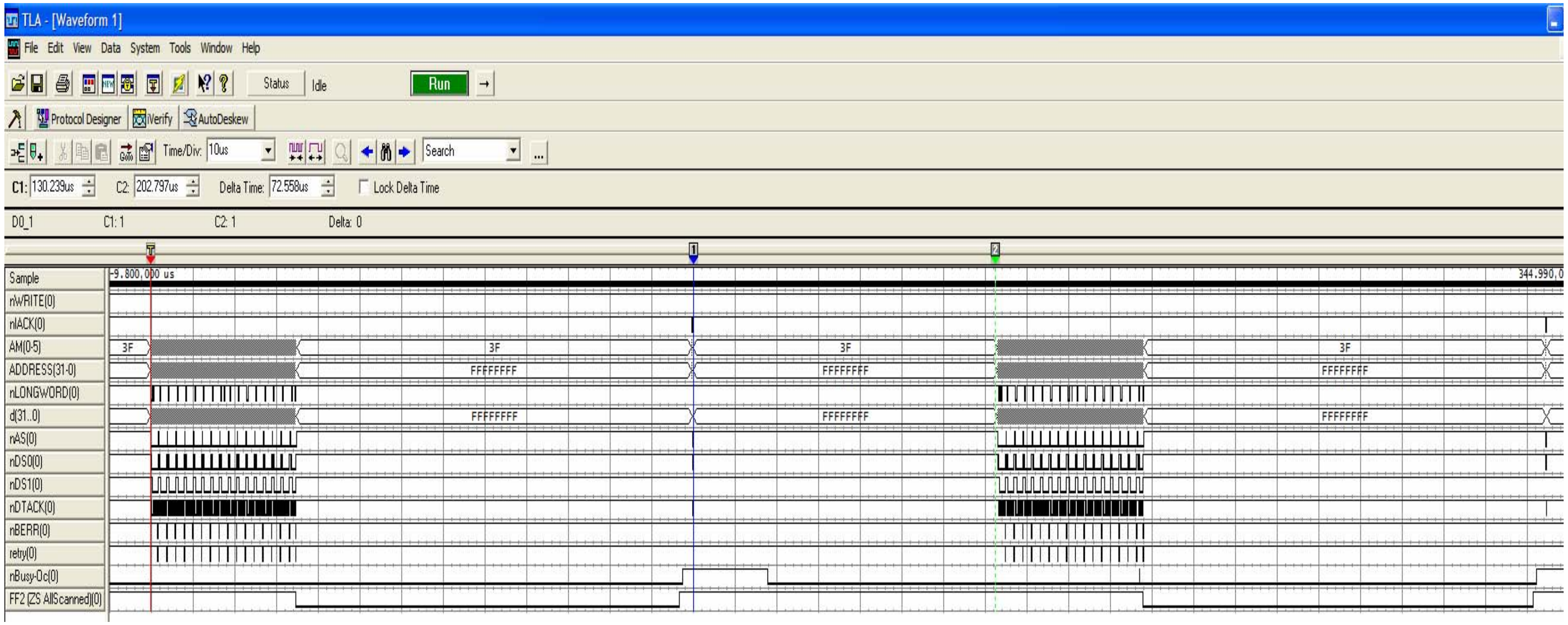
•next:

- MODIFY THE "TopLevel-MIMOSuZe" FIRMWARE TO TRANSFER DATA IN 2e-SST MODE, WITH SLAVE TERMINATION ⇔ THE CPU WOULD ALWAYS SCHEDULE MBLT READ TRANSACTION FOR THE MAXIMUM NUMBER OF BYTES POSSIBLE: THE EUDRB BEING ADDRESSED WILL TERMINATE THE TRANSACTION BY ISSUING THE BERR SIGNAL ➔ THERE WILL BE NO NEED TO SWITCH FROM BLOCK TRANSFER MODE TO SINGLE-CYCLE-TRANSFER MODE ➔ HIGHER PERFORMANCE
- Test DAQ operation with at least 2 boards in the crate (one configured as "TLU-Interface/TimingMaster" and one as "Slave/ExternalTiming" for synchronous operation)
- evaluate a Bit Error Rate with the setup described above

EUDRB-MIMOSuZe: **A VME-64x based DAQ card for M26 sensors. STATUS UPDATE**

•Jan 22th : operation frequency upgraded to 90MHz; 2eSST block transfer capability implemented : performance and problems encountered.

• Logic analyzer screenshot of a 2eSST data transfer session:


Status Idle

Time/Div: 10us

C1: 130.239us C2: 202.797us Delta Time: 72.558us Lock Delta Time

D0_1 C1: 1 C2: 1 Delta: 0

Measurements (for a one Board DAQ):

• time to transfer 2112 bytes of EUDRB event data over VMEbus (**<u>including addressing phase overhead</u>**): ~ 35us

➔**60MB/s of "net" data bandwidth** (could be improved if the Tsi148 driver were optimized transfer were made of larger "atomic" units)

• time between the INTERRUPT GENERATION (cursor 1) and the start of the 2eSST transfer (cursor 2): ~ 72us

(might starting directly the read from within the Interrupt Service Routine)

•time to acquire 2 * M26 frames (roughly from nBusy↘ to nBusy↗): ~ 185us