EMU-DAQ and EMU Specific Hardware

S. Durkin
EMU DAQ Workshop U. Florida
Sept. 19-20, 2002
EMU DAQ System

Fiber Inputs from 432 CSC Chambers connect to 36 DDU Cards in the Counting House

6 DDU’s data is concentrated in a DCC and data sent thru S-LINK to Main DAQ

To average rate CSC’s attached to a given DDU are staggered

Local Computer Farm

For Spy Data and Calibration data each DDU has a Local Gigabit Fiber output to a Local Computer Farm

- Farm has 18 Computers (counting house or surface?)
- 2 DDUs per farm computer

- data analysis is spread across 18 separate computers
- calibrations will not hand-shake with slow control
**Gigabit Fiber -> Local Computer Software**

Each local computer has two commercial gigabit ethernet cards.

Can communicate using Linux Raw Sockets:

```c
sock=socket(PF_PACKET,SOCK_RAW,PROTO);
size=recfrom(sock,9000,0,&from,&fromlen);
```

This is slow (~100 Mbit/s) and one can lose packets.

A better way is to bypass the ethernet protocol completely. All packets are passed from ethernet drivers to the higher level through the routine netif_rx. Replacing this call in the ethernet driver with netif_rx_hook one can write directly from the ethernet card to memory at (~650 Mbits/s) with no packet loss. We retrieve the data from memory with a simple character driver.

```c
open(schar,RDWR_O);
size=read(schar,buf,9000);
```

For calibration we will write all calibration data to memory and then analyse it later.

Note: Without Raid Disks it is impossible to write 650 Mbits/s to hard disk. Even with 2 GigaHz processors one can only do trivial data analysis on events at this rate.
Electronic Hardware Slow Control

A single computer (counting house) will talk to all peripheral crates

Peripheral Crate

Dynatem 360
Controller software written and tested (3/01)

Clock Control Board
Prototype produced & tested
TTC software working.

Trigger Motherboard
Prototype produced and under test

DAQ Motherboard
Pre-production board produced and tested (3/02)

Custom Backplane
Prototype produced and tested with GTLP at 80 MHz
(5/01)

VME can access individual card or broadcast

Imbedded VME Crate Controller

VME Crate Controller
- Provide Slow Control Services
- Embedded processor in each crate.
- Communicates with Linux PC via 10 Base FL Ethernet
- VME to JTAG interface by FPGA on each board in crate

Slow Control functions
- Debug electronics
- Download FPGA firmware, thresholds, calibration constants
- Monitor LV voltages, currents, temperatures.
- Immune to magnetic field: Uses 10 Base-FL in place of 10 Base-T (has transformer).
- Radiation tolerant: Tested with 63 MeV protons.
- Requires only ~50 resets in 10 LHC-years.

Linux PC communicates to Dynatems via Sockets
- Standard Communications TCP
- Broadcast UDP
How Not to Program – 10 Base-T

It seems natural to pass VME commands in a simple format thru sockets to the Dynatem.

<table>
<thead>
<tr>
<th>rd/wr</th>
<th>vme add</th>
<th>vme data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>4 bytes</td>
<td>2 bytes</td>
</tr>
</tbody>
</table>

Given the ~ 2 Mbit/s throughput of 10 Base-T this is a very bad thing to do.

JTAG commands to TMB VME

<table>
<thead>
<tr>
<th>vme add</th>
<th>vme data</th>
</tr>
</thead>
<tbody>
<tr>
<td>x80f00001</td>
<td>00000000000001001</td>
</tr>
<tr>
<td>x80f00001</td>
<td>00000000000000001</td>
</tr>
<tr>
<td>x80f00001</td>
<td>00000000000010000</td>
</tr>
<tr>
<td>x80f00001</td>
<td>0000000000000000000</td>
</tr>
</tbody>
</table>

Thus one sends 112 bits for each useful bit. If one reads back the data things are even worse. 2048 bits for each useful bit.

64x8x2x2 = 2048

At 2 Mbits/s loading a 2Mbit FPGA takes 35 minutes.

Lesson Learned
- Pack Data on PC and Unpacking Data on Dynatem
- Buffer Multiple JTAG Commands into long ethernet packet
With packing and buffering we can download 2 Mbit FPGA in 1 second.

**OSU is Designing a New VME Controller**

A simple PC board (parts available ~ 2months)

- Xilinx Virtex II Pro – built in gigabit ethernet
- Gigabit Ethernet Transceiver

We already have a Gigabit ethernet Xilinx FPGA core. We are designing a limited VME master core.

<table>
<thead>
<tr>
<th>IEEE 802.3</th>
<th>Preamble</th>
<th>Start Frame Delimiter</th>
<th>Destination Address</th>
<th>Source Address</th>
<th>Type/Length</th>
<th>DSAP</th>
<th>SSAP</th>
<th>Control</th>
<th>Data</th>
<th>Pad bytes</th>
<th>Frame Check Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>size (bits)</td>
<td>56</td>
<td>8</td>
<td>48</td>
<td>48</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Not Needed for Raw Packets

Packet length 64 bytes minimum, 9000 bytes maximum

Data Format: 

<table>
<thead>
<tr>
<th>crate</th>
<th>cmd</th>
<th>vme address</th>
<th>vme data</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
<td>32</td>
<td>16</td>
</tr>
</tbody>
</table>

Even without packing or buffering VME rates are reasonable!

2048 bit => 2 microseconds download 2 Mbit FPGA in 2 seconds

- much simpler VME controller
- programming all on Linux PC
- same drivers as alternate DAQ path
- commercial gigabit ethernet card in PC
Local triggers (e.g. calibration) will be handled by the CMS L1 Trigger Control System

Local Trigger Control (LTc)
- Program thru VME backplane
- Trigger using 6 FRONT panel BNC’s
- Available in January 2003 ????

EMU has 2 L1 Trigger Partitions
Buckeye Calibration Example

1. Set L1 Trigger Control to Local (TCS)

2. Program Local Trigger Controller (LTC)

3. Shift Buckeye Calibration Bits (Slow Control)

4. Set Calibration DAC, Timing (Slow Control)

5. Send pulse to Local Trigger Control (Slow Control)

6. Process data in Local DAQ Farm (Local DAQ Farm)