L2 Software

- Overview
- Algorithms
- Error Handling code
- Code management
- Connection to trigger DB
- Code testing
- Plan, schedule, manpower

Peter Wittich, for L2 software types:
Stephen Miller, Heather Ray, Masa Tanaka, Tom Wright
Overview

Basic job of processor:

1. Get data from clients
2. Evaluate triggers
3. Report trigger decision

Implementation:

- C++ code running on α processor
  - Have most strengths of C++: type safety, inheritance, inlined code
  - Don’t use some of the ‘costlier’ aspects of C++: RTTI, exceptions, virtual inheritance, STL….
- No OS - Evaluation Board Software Development Kit (EBSDK) as pseudo-OS to provide basic services such as console I/O
- Code is compiled on DEC Workstation running OSF 4.0D and same CPU as on α board.
Overview

Code runs on $\alpha$ processor, with a simplified event loop as follows:

- Initiate data transfer from clients
- Manage DMA FIFOs (receive data from interface boards) to receive next event
- Reformat data into local structures
- Error checking
- Evaluate triggers
- Communicate decision to TS
- On accept, read out RECES/muon board and create TL2D

The strategy is to make the code
1. Correct
2. Robust
3. Fast
in that order.

Code to be moved to FPGA

$L2$ review, Dec 7, 2001
Algorithms

• Necessary algorithms defined by CDF note 4718.
  - ~40 triggers, covered by roughly 20 trigger objects/algorithms with differing options.
  - 4718 ‘lite’ (all the physics, half the code?): an interim selection with somewhat fewer triggers.

• All algorithms $\alpha$-independent
  - Allow porting to other platforms and to offline code.
Currently existing algorithms

We currently have code in CVS for the following sets of triggers.

- AutoAccept
- CentralElectron
- Photon
- GlobalEt
- HadronicB
- Jet
- Dijet
- L2Error
- SvtTest
- SvtTrack
- SvtXftTrack

Legend:
- ✔️ tested successfully
- ✌️ being tested
- 🕒 code written, not tested
- ✗ placeholder

L2 review, Dec 7, 2001
L2 Software and CDF4718

JSM slide:

L2 options written are highlighted in **Blue**
Many still need to be tested

<table>
<thead>
<tr>
<th>Trigger</th>
<th>L2 Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 zerebias</td>
<td>AutoAccept</td>
</tr>
<tr>
<td>2 minbias</td>
<td>AutoAccept</td>
</tr>
<tr>
<td>3 singletower5</td>
<td>AutoAccept</td>
</tr>
<tr>
<td>4 Jet20</td>
<td>Jet</td>
</tr>
<tr>
<td>5 Jet50</td>
<td>Jet</td>
</tr>
<tr>
<td>6 Jet70</td>
<td>Jet</td>
</tr>
<tr>
<td>7 Jet100</td>
<td>Jet</td>
</tr>
<tr>
<td>8 HighEtCentralEl</td>
<td>CentralElectron</td>
</tr>
<tr>
<td>9 Muon</td>
<td>AutoAccept</td>
</tr>
<tr>
<td>10 BtoPiPi</td>
<td>HadronicB</td>
</tr>
<tr>
<td>11 Bs</td>
<td>HadronicB</td>
</tr>
<tr>
<td>12 Met+2Jets</td>
<td>NJets</td>
</tr>
<tr>
<td>13 Met</td>
<td>GlobalMissEt</td>
</tr>
<tr>
<td>14 JPsiMuMu</td>
<td>AutoAccept</td>
</tr>
<tr>
<td>16 JPsiEE</td>
<td>JPsiEE</td>
</tr>
<tr>
<td>17 RadiativeB</td>
<td>ElectronPlusSvt</td>
</tr>
<tr>
<td>18 W/ZHiggs</td>
<td>SvtTrack</td>
</tr>
<tr>
<td>19 tt-&gt;jets</td>
<td>ClusterSum</td>
</tr>
<tr>
<td>20 Diffraction</td>
<td></td>
</tr>
<tr>
<td>20.1 AutoAccept</td>
<td></td>
</tr>
<tr>
<td>20.2 Jet</td>
<td></td>
</tr>
<tr>
<td>20.3 Jet</td>
<td></td>
</tr>
<tr>
<td>20.4 AutoAccept</td>
<td></td>
</tr>
<tr>
<td>20.5 AutoAccept</td>
<td></td>
</tr>
</tbody>
</table>
Most algorithms have code written for them at varying stages of completion and testing.

<table>
<thead>
<tr>
<th>Trigger</th>
<th>L2 Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>22 HighEtPhoton_Iso</td>
<td>Photon</td>
</tr>
<tr>
<td>23 UHEtPhoton</td>
<td>Photon</td>
</tr>
<tr>
<td>24 SHEtCluster</td>
<td>Photon</td>
</tr>
<tr>
<td>25 HEtDiPhoton_Iso</td>
<td>Photon</td>
</tr>
<tr>
<td>26 LoEtDiPhoton_Iso</td>
<td>Photon</td>
</tr>
<tr>
<td>27 Photon+2Jet</td>
<td>Photon</td>
</tr>
<tr>
<td>28 Three EmClust</td>
<td>Photon</td>
</tr>
<tr>
<td>29 MET + 2btags</td>
<td>SvtTrack</td>
</tr>
<tr>
<td>30 MET+PEM</td>
<td>Photon</td>
</tr>
<tr>
<td>31 .1 e+isol track</td>
<td>ElectronPlusTrack</td>
</tr>
<tr>
<td>31 .2 CMUP+isol track</td>
<td>AutoAccept</td>
</tr>
<tr>
<td>31 .3 CMX+isol track</td>
<td>AutoAccept</td>
</tr>
<tr>
<td>32 Dileptons</td>
<td></td>
</tr>
<tr>
<td>32.1 CEM+CEM</td>
<td>CentralElectron</td>
</tr>
<tr>
<td>32.2 CEM+PEM</td>
<td>ElectronPlusPhoton</td>
</tr>
<tr>
<td>32.3 CMUP+CMUP</td>
<td>AutoAccept</td>
</tr>
<tr>
<td>32.4 CMUP+CMX</td>
<td>AutoAccept</td>
</tr>
<tr>
<td>32.5 CMX+CMX</td>
<td>AutoAccept</td>
</tr>
<tr>
<td>32.6 CEM+CMUP</td>
<td>CentralElectron</td>
</tr>
<tr>
<td>32.7 CEM+CMX</td>
<td>CentralElectron</td>
</tr>
<tr>
<td>32.8 PEM+CMUP</td>
<td>Photon</td>
</tr>
<tr>
<td>33 Two XFT Tracks</td>
<td>XFTTrack</td>
</tr>
<tr>
<td>34 Photon+muon (charm)</td>
<td>Photon</td>
</tr>
<tr>
<td>35 Z-&gt;bb</td>
<td>SvtTrack</td>
</tr>
<tr>
<td>36 HighpT b-jet</td>
<td>HighPtBjet</td>
</tr>
<tr>
<td>37 .1 CMUP+disp. track</td>
<td>SvtTrack</td>
</tr>
<tr>
<td>37 .2 CEM+disp. track</td>
<td>ElectronPlusSvt</td>
</tr>
<tr>
<td>38 Di-tau</td>
<td>DITau</td>
</tr>
<tr>
<td>39 MET+tau</td>
<td>CentralElectron</td>
</tr>
<tr>
<td>41 HiEtPhoton_Iso</td>
<td>Photon</td>
</tr>
<tr>
<td>42 W (no track)</td>
<td>Photon</td>
</tr>
<tr>
<td>43 e+track (no e iso)</td>
<td>ElectronPlusTrack</td>
</tr>
</tbody>
</table>
Code Testing

Right Now:
- Currently, we have two methods of testing code
  - Run it in the detector and check results (CDF as a pulser)
  - Run Masa’s test code:
    - Writes L2 raw data into $\alpha$
    - $\alpha$ runs triggers
    - Read out TL2D and compare to expectation.
With this we can do:
  - Correctness checks
  - Simple robustness and timing checks

Longer Term:
- Generate test fixture to run code in AC++ environment
- L2_Pulsar to drive test patterns (similar to Masa’s code)

Short of using CDF, no realistic method for getting timing or performance w/o L2_Pulsar.

L2 review, Dec 7, 2001
Code Performance: Timing

Only performance tests so far on prototype triggers:

- Time to run single triggers measured on single events
- Time to perform ‘bookkeeping’ measured
- Average performance goal is ~10 µsec/event for entire event loop (excludes event loading)
- Unclear (to me, anyhow) where we stand with respect to this goal.

➢ However, clear paths to improving the software exist, if need be.
L2 Error handling in place and tested

- Code generates L2 timeout for three current error conditions:
  - No L1 magic bus words sent
  - CList buffer number mismatch
  - SVT-XTRP BC mismatch

More error checks can be added pretty easily.

- On error, a message appears in run control explaining the error
- First two run by default in all triggers.

To Do:

- Implement Error triggers
- Switch from L2 timeout to pulling CDFError
  - Code exists, needs to be tested
- Implement error statistics

L2 review, Dec 7, 2001
Code Management

- Code exists in three packages
- All packages under version control system (CVS) in online DAQ repository
- Code specific to hardware implementation separated from trigger algorithms
- Trigger-table specific code limited to two files that are created by TriggerDB and #include’d into the source code
  - This is the only connection between the trigger DB and the code

Unfinished business:

- Need CVS package release version numbers in Trigger DB
- Need to split L2 code from L3 code - update L2 code w/o changing L3, others (affects trigger tables?)
- Adapt build procedure to use well-defined CVS version.

L2 review, Dec 7, 2001
Trigger DB to code

- Trigger table-specific code is included via two files:

```c
void mainloop(void)             // The main prog.
{
    // [code elided]
    #include "L2cuts.hh"

    MagicBusDMA mbusdma;       //see mbusdma.h
    TSI tsi;                   //see tsi.h
    // event loop
    while (1) {
        // [code elided]
        #include "L2triggers.hh"
    }
}
```

Code is auto-generated by Trigger DB GUI.

```
// TRIGGER TABLE Physics_0_01_v-48
numL2Trigs = 38;
AutoAccept l2trig_787;
l2trig_787._l2Bit = 14;
l2trig_787._l1Bit = 21;
```

define triggers

evaluate triggers
Manpower/Schedule

- Identified L2 software writers:
  - Stephen Miller
  - Heather Ray
  - Masa Tanaka
  - Peter Wittich
  - Tom Wright

  However, it is unclear what fraction of these peoples’ time will be available to write L2 software.

Goals:
- Code for 4718 lite by mid-January 2002
- Code for all 4718 triggers by end of January 2002