Table of Contents

My notes on the CMS fedbuilder

Misc.

- Switches seem to be twelve CLOS256+256 (see e.g. http://www.myri.com/myrinet/product_list.html). The whole network seems to be implemented in two identical copies (for bandwidth doubling). One instance consists of three switches (crates) in USC and three in SXC.

- The line cards are probably M3-SW32-16F. Each of these has one xbar32 (crossbar) chip which can route 16 ports (front panel) on 16 ports (backplane) where each port seems to consist of an input and an output.

- A clos256+256 consists of (up to) 16 M3-SW32-16F connected to the FRLs (USC) or Readout Units (SXC) and four M3-4SW32-16Q which make 256 connections to another myrinet switch (i.e. the connect the switches in USC to those in SXC).

- Latest snapshot of the UCSD myrinet routing software: click here (Java webstart and Java 1.5 or newer required. If you can run EVO, this link should probably work).

Thoughts on routing

- in order to avoid collisions, the 16 inputs of an xbar must be distributed to the 16 outputs (and vice versa) of the xbar

  Thus one category (corresponding to one SCX output xbar) must be distributed across the 16 SCX input xbars.

  Thus each category must only cross an SCX input xbar once.

- for the moment, one category also corresponds to an USC input xbar, thus the same argument (for category crossing) applies for the USC output xbars as well.

Links

- Myricom presentation
- The CMS experiment at the CERN LHC, section 9.4 'The Event Builder' (DOI).

2010-03-16

- Andre's presentation in the DAQ meeting of the CMS week: slides

2010-05-31

- routing seems to work as follows:
  - FedBuilderMultiCrate.route(..) seems to loop over all USC output xbars by calling AssignInterXbar(..)
  - AssignInterXbar(..) calls FedBuilder.doAssignL1(..) with the first USC input xbar (and the given USC output xbar) to assign routes. This method then calls itself with the next USC input xbar.

    - FedBuilder.doAssignL1(..) calls itself and FedBuilder.doAssignL2(..) which assigns a single line.
    - FedBuilder.doAssignL2(..) essentially calls SetUSCOutput(..), performs some consistency checks and undoes the assignment if there are conflicts.

- user specified routing constraints (apart from physical configuration such as cabling and broken fibers):
    - ForceOutputCrateData: which USC input xbars should be routed to which SCX crate

- for assigning/unassigning: the only thing which should be unassigned is the actual route, other auxiliary quantities (such as the number of used/unused outputs etc.) should be determined on the fly from the routing...

2010-06-01

- another thing which is not clear to me: why for example input lines can be marked as 'done' while they don't have a category assigned ?

2010-06-07

- the nameserver seems to know 640 ru's.

Talking to Sham:

2010-07-02 myrinet monitoring

* some notes by Sham are here:
https://twiki.cern.ch/twiki/bin/view/CMS/CDAQOnCallHowTOs#Monitoring_Myrinet_Optical_Links

(from a mail sent on the cms-daq-daqoncall list on this date)

2010-07-15

- possible failures:

-
    - single myrinet link goes down, single myrinet transceiver (NIC or FRL) card goes down: can be masked because there is a parallel link (gives reduced bandwidth though)
    - RU (entire PC, e.g. power supply) goes down: what happens then ? Must mask the slice until the PC is repaired ?
    - FRL goes down: must fix this immediately...

- counted the number of RU's per slice: 69 of them. This gives 69*8 = 552 links. On the other hand, we have 12 linecards per switch on the Myrinet router, this gives 12 linecards * 3 crates * 16 lines = 576 links. This means we have 24 more links than currently needed by the RUs.

2010-08-10

- number of FRLs and FEDs from the CMS JINST Paper:

Table 9.1: Sub-detector read-out parameters.

sub-detector	number of channels	number of FE chips	number of detector data links	number of data sources (FEDs)	number of DAQ links (FRLs)
Tracker pixel	≈ 66 M	15840	≈ 1500	40	40
Tracker strips	≈ 9.3 M	≈ 72 k	≈ 36 k	440	250 (merged)
Preshower	144384	4512	1128	56	56
ECAL	75848	≈ 21 k	≈ 9 k	54	54
HCAL	9072	9072	3072	32	32
Muons CSC	≈ 500 k	≈ 76 k	540	8	8
Muons RPC	192 k	≈ 8.6 k	732	3	3
Muons DT	195 k	48820	60	10	10
Global Trigger	n/a	n/a	n/a	3	3
CSC, DT Track Finder	n/a	n/a	n/a	2	2
Total	≈ 55 M			626	458

2010-08-11

- Jim pointed out that the Java program (with the classic algorithm) routes 4 USC line cards to each SCX crate which is not optimal (because there are twice the links to the same number crate as there are for non-same number crates).

- Marco has sent the table used at point 5 and the program which he used to produce this table (the version I was using before did not have the actual constraints used at point 5).

2010-08-12

- discussion on (amongst others) the requirements for the routing program
    - ♦ Frans (or somebody else) will have to implement into the Myrinet driver / firmware flexibility for configuring the entire route (not compile it into the driver/firmware)

- for the moment, still route linecards to linecards. In the future, one could be more flexible.
- One Fedbuilder seems to be 1-16 FRLs (on the FRL side) and N RUs on the RU side (where N is the number of slices which is currently 8). Note that a Fedbuilder on the FRL side can have more than eight FRLs.

- doing

```
grep -i frlpc hostnames | grep -v CNAME | cut -d.  -f1
```

in `~/hostnames` (on the online cluster), I see 3 `dvfrlpc` and 50 FRL PCs

2010-08-13

New routing algorithm fails with the following four links broken:

```
failed to find a route with the following list of broken links:
  edu.ucsd.hep.fedbuilderroute.routingalgorithms.RoutingAlgorithmException: too many categories t
  (USC to SCX link: USC crate=2,USC xbar=8,USC line=2
```

(and other examples). Note that the link breaking recipe was:

- break four links between non-same-number crates
- make sure no xbar has more than one broken link

So in principle, the routing algorithm should still find a solution...

2010-08-26

- the problem of finding the 4x4 routing categories (with the presence of broken links) can be formulated as a **Latin Square** problem:
  - columns are USC xbar
  - rows are SCX xbar
  - small squares are the links between an USC and an SCX xbar
  - colors are memberships to a 4-route
  - broken links are assigned their own color. If there is more than one broken link per xbar, more than one color (i.e. more than one category for broken links) is needed

    Not clear how to generalize to the case of 32 links between 4 and 4 xbars (between same numbered crates)

- From the Latin Square wikipedia page:

```
The problem of determining if a partially filled square can be completed to form a Latin square i
```

which means that it is in NP which in turn means that there is no polynomial time algorithm known (see http://en.wikipedia.org/wiki/NP_%28complexity%29).

- Sudoku can be solved by a back-tracking algorithm, which seems to be essentially a tree (of all combinations) with some pruning (i.e. when it becomes obvious that the current solution can not be completed). This seems to be what Elliott did (with the recursion, but on all 16 xbars, not the 4x4 ones).

* Note that the Latin Square of size 4 has only 576 different squares (that corresponds to the case of non-equal crate numbers). And there are only 2 isotopy classes (where isotopic is defined as equal after permutation of rows/columns and the label names).

2010-08-27

- The linux driver seems to be here: https://svnweb.cern.ch/trac/cmsos/browser/trunk/daq/myrics/myrfb/driver/myrfb_drv.c

- in fact the FRL PCs seem to have several 'myrinet network cards' according to lspci. `/sbin/lspci | grep -i myrinet` on frlpc-S1D06-01 gives:

```
05:0a.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
05:0b.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
05:0c.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
05:0d.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
05:0e.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
05:0f.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
06:08.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
06:0a.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
06:0b.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
06:0c.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
06:0d.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
06:0e.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
06:0f.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
```

```
07:0d.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
07:0e.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
07:0f.2 Network controller: MYRICOM Inc. Myrinet 2000 Scalable Cluster Interconnect (rev 05)
```

- the driver struct seems to be of type `myrfb_dev_t`.
- the struct `flow_t` is defined in https://svnweb.cern.ch/trac/cmsos/browser/trunk/daq/myrics/myrfb/include/myrfb_mcp.h . This contains `netadr` (the destination address), but what about the actual routing (the addresses of the intermediate xbars ?)

2010-09-03

- The chromatic polynomial gives the number of possible colorings of a graph as function of the number of colors to be used. When setting 'color' = category route and the graph has vertices which corresponds to myrinet links and edges which mean 'go to the same USC or SCX xbar', the number of colorings is the number of different ways of grouping the links into categories.
- There are also efficient algorithms to construct this polynomial for a given graph.

2010-09-06

- Calculating Chromatic Polynomials in Mathematica is described here. This function seems to count all permutations of colors, so in order to get the number of different possible distributions of categories, one would have to divide by the number of permutations for the given number of categories.

  The example shown on the Mathematica page shows 114 possible colorings for three colors which is what one also gets from the corresponding Chromatic polynomial:

```
def poly(z):
  from math import pow
  return pow(z,8) - 12 * pow(z,7) + 66 * pow(z,6) - 214 * pow(z,5) + 441 * pow(z,4) - 572 * pow(z
```

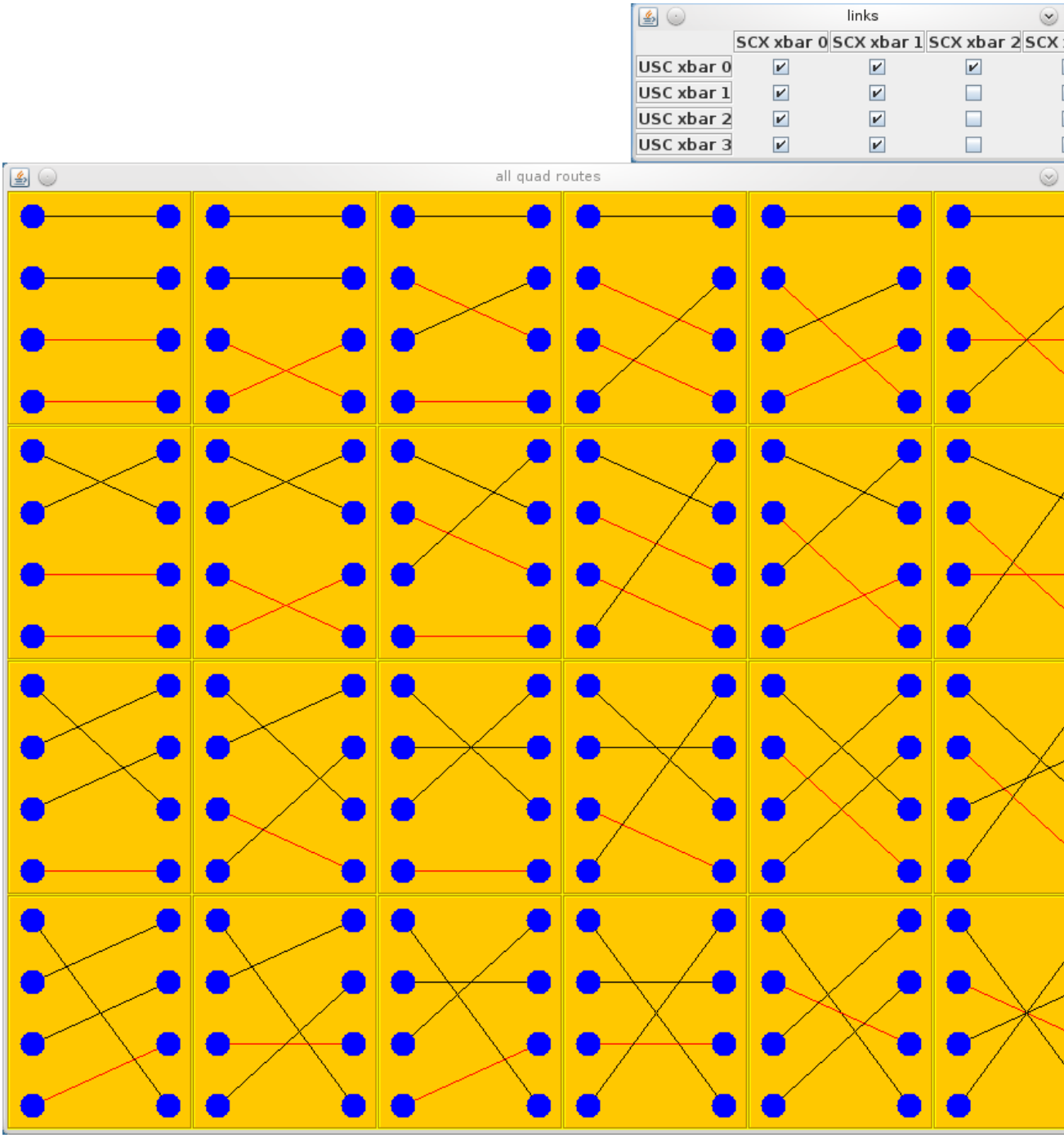
- starting Mathematica on lxplus:

```
/afs/cern.ch/project/parc/math70/bin/Mathematica
```

NOTE: the Chromatic Polynomial does not say ANYTHING about how many vertices ('links') are assigned to a given color. In our case, we need to have exactly (at least) four vertices assigned to the same color, but the chromatic polynomial does not care about this.

2010-09-07

- one could formulate e.g. the 32 link routing problem between 4+4 xbars as a linear programming problem (make sure that each tuple of (USC xbar,SCX xbar) does not occur more than twice). But then it's an INTEGER linear programming problem which is also NP-hard according to Wikipedia.
- 2010-09-07-quad-xbar-routing-example.png:



2010-10-25

- found the following line of code:

```
int myriAddr = (int) (1024 + 16 * outputLineCardId + 2 * getSliceNumberFromName(sliceName));
```

in
http://isscvs.cern.ch/cgi-bin/cvsweb.cgi/TriDAS/RunControl/framework/utilities/hwcfg/fillers/src/rcms/utilities/hwcfg/

2010-10-31

For the 'between same crates' routing problem: each of the links can be used at most twice (or less if some of them are broken). We can formulate this as an integer programming problem (see also

http://en.wikipedia.org/wiki/Linear_program#Integer_unknowns):

maximize

```
c^T * x
```

with constraint

```
A * x <= b
```

where x is the vector of 'how many times a pattern in the above list of 24 patterns is taken' (i.e. has length 24), b is the number of times a link is available (i.e. has length 16) and A_ij is 1 if link i is used in pattern j (or zero if it isn't). Note that this matrix is NOT square. The vector c consists of all ones such that c^T * x is the number of four-patterns included in the solution.

See e.g. also http://www.cs.brown.edu/courses/csci1490/slides/CS149-TUM.pdf

2011-03-21

New CMS wide twiki page created: https://twiki.cern.ch/twiki/bin/view/CMS/MyrinetRoutingProgram

2011-03-25

Getting the association of FRLpcs, Myrinet addresses and FED source ids:
http://srv-c2d04-30.cms:9941/urn:xdaq-application:lid=400/retrieveCollection?fmt=html&flash=urn:xdaq-flashlist:frlV

or in a more machine readable format:

http://srv-c2d04-30.cms:9941/urn:xdaq-application:lid=400/retrieveCollection?fmt=plain&flash=urn:xdaq-flashlist:frlV

2011-04-05 Talking to Sham

- his connector numbering starts at 1 (at the bottom), which is also the fiber number which is also the slot number in the FRLpc (which is also the digit `x` in the name of the proc files `/proc/driver/myrfb/myrfb_gx`).

2012-02-20

the program for visualization of the switch front panel numbering is `CrateFrontPanelSummaryPanel`

-- AndreHolzner - 09-Jan-2010

This topic: Main > AndreHolznerFEDBuilder
Topic revision: r31 - 2012-02-20 - AndreHolzner