A Real-Time Tracker for CMS

A. Bardi, S. Belforte, M. Dell'Orso,
E. Meschi, P. Giannetti and F. Spinella

Abstract

Motivations and feasibility of a real-time tracker for the CMS experiment are briefly reviewed. We conclude that such a device would fit nicely into the present CMS trigger/DAQ architecture in such a way as to provide tracking information to the Level 2/3 processors.

While it is still an unresolved issue whether this device would be a nice addition, a luxury, a burden, or an absolute need for the CMS experiment, the hardware requirements appear to be within the range of near-future technology.

1 What is it ?

In this document we talk about an hypothetical dedicated electronic device, a"Global Tracker", that could sit in the CMS DAQ architecture before the Level 2/3 buffers (RDPM). This device will receive clustered data from the CMS tracker (pixels, silicons, MSGC's) as they are read out after a positive Level 1 decision, in parallel to the transmission of the full data to the RDPM's. The "Global Tracker" is the first stage of the process described in figure 1 and performs the most brute force part of the pattern recognition work.

The "Global Tracker" implements an algorithm that finds track candidates (*roads*) using the detector as if it had coarser resolution. It keeps up with the Level 1 rate of 100 kHz and delivers *roads*, with the list of full resolution clusters contained inside each *road*, to an RDPM for usage by the higher level CPU's within 1-2 msec. From the Event Builder and the CPU farm point of view, this device appears as an added RDPM from a "pseudodetector", providing ordered and selected tracking information.

This Global Tracker allows to divide the enormous problem of finding tracks inside the global detector into the many simpler problems of track finding inside *roads*.

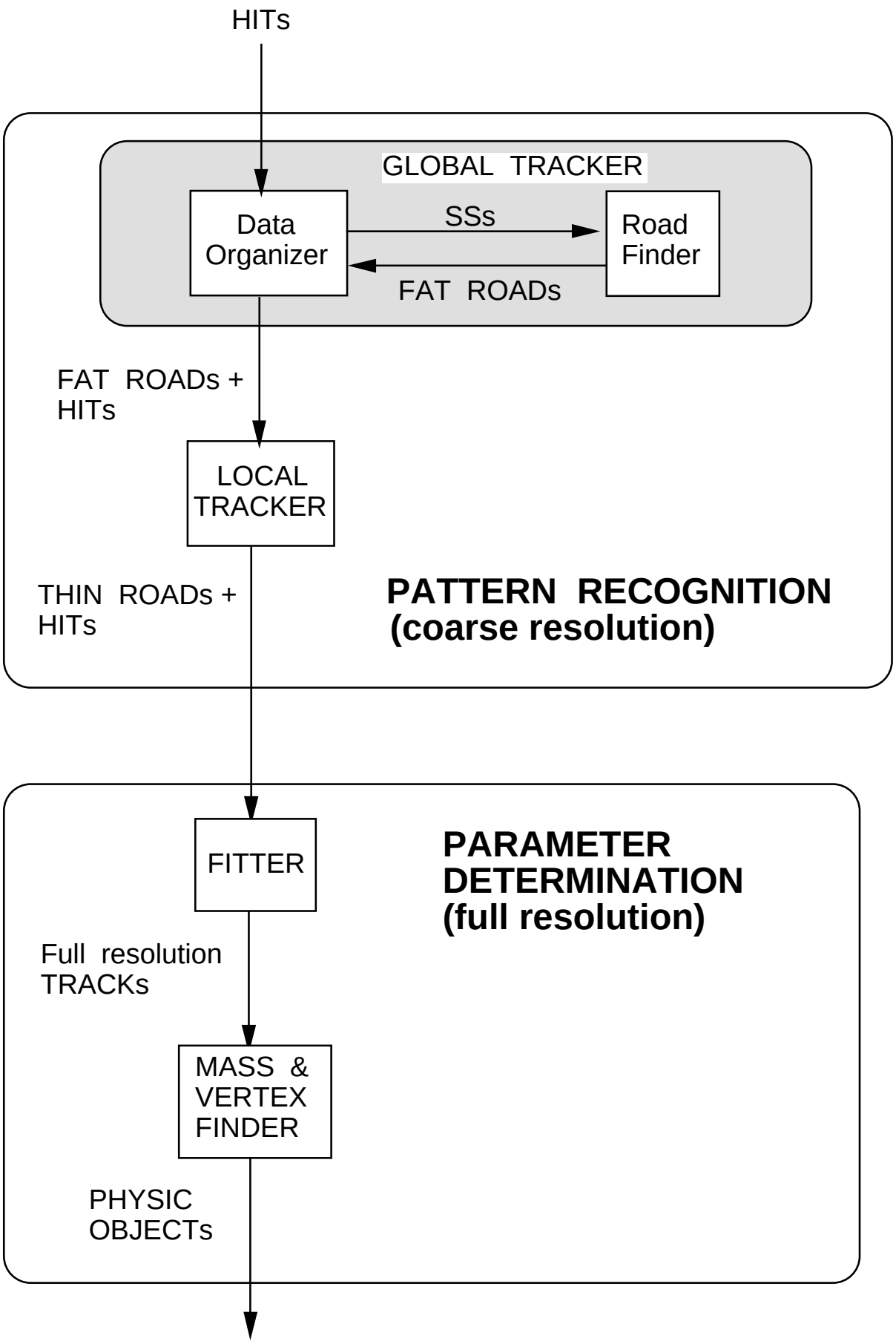

Figure 1: Pipelined algorithms to find tracks and related quantities

One way to look at this, is to realize that, while waiting for virtual level 2 to validate the level 1 trigger, $1-2$ milliseconds go by, and all of this time the tracker data are sitting idle in the level 2 buffers lazily rolling down the RDPM pipelines. The subject of this document is a device that will try to do something useful with those data during this time.

To get a feeling for the meaning of the above numbers, it is instructive to compare with the situation at the CDF experiment where an hardware tracker is currently being built for the Silicon Vertex Detector [1]. The CDF tracker processes data with the same 100 kHz input rate as CMS, but with an overall allowed processing time (latency) of 10μsec. The affordable latency at CMS is at least 100 times greater, a whole world of possibilities opens up for a dedicated tracker! Of course only as long as the device can keep up with the input rate: the amount of data to be handled each 10μsec at CMS is much larger then in the CDF case, but CMS can exploit about 10 years of electronic technology advancement.

1.1 Architecture

Pattern recognition is a CPU time consuming task. Since the CMS trigger has a very long latency time, pipelining can be used extensively to divide the complex work into many pipelined steps with different degrees of approximation.

Figure 1 shows a set of boxes representing different algorithms working in pipeline to perform pattern recognition with the full detector resolution and using tracks to calculate interesting quantities like invariant masses, primary and secondary vertices, etc. These algorithms can be strongly parallelized.

In general terms, all the algorithms shown in figure 1 deal with a detector divided in a certain number of layers, each layer being segmented into a number of *bins*. When charged particles cross the detector they *hit* one bin per layer. Each track generates a set of hits. The coordinate of each hit will actually be the result of some computation performed on raw data, for example the center of gravity of a cluster. These calculations have to be done upstream and the resulting binned coordinates (HITs) are used by the algorithms of figure 1.

The Road Finder receives "SuperStrips"[1)], or SS for short, obtained by logically ORing adjacent detector bins (e.g. by simply ignoring a number of less significant bits in the bin coordinate) and performs a very fast pattern recognition at this reduced level of spatial definition. The "Data Organizer" receives full resolution cluster centroids (HITs) in random order, and buffers them in a data base, to be fetched when the *roads* are received. For each *road* the Data Organizer can send out all full resolution HITs associated to the *road*. The CDF application has shown that the Data Organizer can perform this nontrivial task at full rate, and that the needed time is the same of a simple buffering function.

The Local Tracker should be powerful enough to deal with large *roads* full of tracks.

When the *road* size is reduced enough to contain only few HITs, the pattern recognition problem is almost solved and the few residual hit combinations can be fitted sequentially by the FITTER to find the best track parameters. Each thin *road* with the explicit list of hits, already

[1)] the name substitutes "SuperBins" that is more appropriate. It originates from silicon detectors, and it is taken from the vocabulary of the CDF Silicon Vertex Tracker project

distilled from the plethora of tracker hits, will yield one or more (or none!) offline-quality high-resolution tracks. A fast FITTER can afford a high number of combinations working on large *roads* and reduce the requirements on the Local Tracker.

The full resolution tracks output by the FITTER can be used by higher level algorithms (MASS-VERTEX FINDER) looking for "physical objects" that provides event signatures.

1.2 Is it useful?

We regard this question as vastly academic, the task of the CMS high level trigger CPU farm (Level 2 and Level 3) is so enormous, that tracking information is absolutely needed. It is common knowledge that the needed rejection factor may only come from exploiting the tracking information, and that this will be the most demanding part of the reconstruction code. The sooner reliable tracking information is available to the farm processors, the easier it will be for them to take enlightened decisions about each event destiny. Therefore some pre-digestion of the tracker data that presents to the CPU only the hits in the "interesting regions" is a definite bonus, and the more restricted this region is (i.e. not simply a 10^0 slice of the whole tracker), the better.

We also notice that there are cases where the interesting region is only defined by the track itself [2)]! Here, in lack of a dedicated tracker processor, the trigger farm CPU has to solve the full track reconstruction problem in the whole detector and "find the right track for all hits".

From this we conclude that a list of tracks (or even only track candidates, or even only track-to-be regions) would be extremely useful to the trigger farm of CMS, and we are motivated to keep going in this exercise.

2 Is it needed ?

Now we wonder whether CMS can live without such a device, or will have to give up some potentially significant physics.

There is no demonstration yet that a hardware tracker is absolutely needed. Nor is there any demonstration that a software tracker is fully adequate to the task. Definitely more work in this sector is needed before the question "is it needed ?" is answered. This is not an easy question at all! So we do not make any claim that a hardware tracker is needed, but do not honestly see how one could make the contrary claim either. It is an unsettled issue, and we should work with the assumption that it will remain such for quite some time.

Nevertheless we have a personal prejudice that, even if not absolutely needed, a hardware tracker would be so much helpful that it is definitely worth not to dismiss the issue now, but rather keep investigating its feasibility.

2.1 A future possibility

All the algorithms represented in figure 1 (or equivalent ones) can be implemented using many different technologies. The most convenient technology might be commercial CPUs. This

[2)] e.g. B_s full reconstruction

approach has the advantages of flexibility, standardization and easiness of upgrade: commercial hardware (a processor farm) is surely much easier to install, manage, maintain and upgrade then dedicated home-built hardware. However complexity has some limits. Managing a farm of several thousand elements, may be in the end a not less demanding task (nor necessarily cheaper) than operating a few racks of custom electronics. We should also consider that the need for a tracker upgrading is all in all somehow limited. We can easily tell already that the number of hits in the CMS tracker is hardly going to increase significantly (i.e. by much more than a factor $2 \sim 3$) for many years. In a certain sense a dedicated tracker may be regarded as an extension of the detector and will just stay what it is until the detector itself is replaced.

The proposed CMS trigger farm may be able to find all tracks that are needed to implement the CMS physics program within the trigger time constraints and with affordable cost and complexity. The dedicated hardware may simply not be needed at all since simple selection mechanisms may reduce the data rate to the level where a full fledged offline reconstruction is affordable. However:

- We do not have a definitive list yet of all physics channels that CMS will want to study, of how they will be put on tape, and thus of how much the data filtering will need tracking, and how heavy the requests on the tracking algorithm will be.
- As a consequence no realistic estimate exists yet as to how much CPU time will be required, and how CMS will take advantage of future higher performance commercial processors.
- No matter how well the above points will be resolved in the near future, it is unlikely to be a completely settled issue until data taking has started, and still unforeseen needs may always arise even once the experiment is running, as consequence of deepened detector and/or physics understanding. The TeV mass scale is quite an unexplored domain, all in all.

If the CPUs cannot support the whole work at the necessary rate, it could be conceivable to insert some dedicated hardware. We notice that the pattern recognition, the heaviest part of the work, is exposed to a particular risk: its execution time may not scale linearly with the expected increase in CPU speed of the commercial processors. Fishing out the correct hits that belong to each track calls for a lot of random accesses to a large storage that contains all the tracker data, this amount of data will definitely be outside the high speed cache of the processor and memory access is not going to improve as fast as CPU clock speed. Moreover the "brute-force coarse-resolution work" does not profit very much of CPU flexibility.

Instead, full resolution tasks such as that of the FITTER or MASS-VERTEX FINDER, may need to handle many specific situations like local corrections, alignment effects, exceptions etc. The CPU flexibility is a great advantage for these tasks. In the scheme of figure 1, the commercial CPU will execute the full resolution algorithms and will cover as much as possible of the coarser resolution tasks.

In conclusion, if a dedicated hardware will be useful, it will be advantageous to insert it at low level in the CMS trigger/DAQ system. Although we agree that it may look odd to think now about the upgrade of an experiment still 10 years in the future, it should not be dismissed lightly the point that CMS will be a very large and complex experiment, and whichever flexibility is

not built in it from the beginning, may be very hard to add after the system is in place. Thus it seems convenient leave the door open to the possibility of complementing the present CMS baseline design with a specialized fast tracker, even if we don't know for sure if it will be necessary. Obviously as long as this door can be left open without compromising the baseline performance and cost. This means that we could just implement now few changes there where CMS is already making an investment in specialized hardware, and guarantee the possibility of feeding data to a hardware tracking device (see section 4). We do not need to decide now how much work has to be done in a dedicated system. This amount of work can be "nothing", "something" or "a lot", depending on the CMS physics program and how much the CPU farm can cover. The limit between the commercial CPU and the useful dedicated hardware could be set between the "Global Tracker" and the Local Tracker of figure 1, or between the Local Tracker and the FITTER if the FAT ROADs are too complex for commercial CPUs. We do not need to set now a limit between dedicated processors and CPUs. Physics will decide in the long term.

3 Is it usable ?

This section is dedicated to the issue of whether the proposed device would really fit into CMS. Is it possible to collect all the needed data ? Will this operation keep up with the event rate ? Can data be concentrated in one device with the given time constraint and available technology ? Will the device output be ready in time for CMS to use it and will output data size be handleable ? Which kind of burden would it add to the CMS front end electronic ?

In other words, now we go from words to numbers, and estimate what kind of door one needs, and what is the cost in keeping it open.

3.1 What can a chip handle?

Whichever strategy and architecture will eventually be chosen, any hardware device will eventually have one component that "will look at all the data" in order to perform the pattern recognition.

Here "all the data" is still an ill-defined quantity that will be better specified in the following and will eventually be tuned for a given specific device, but definitely means a good fraction of the CMS tracker information. To be specific we consider as basic device one electronic chip, and ask ourselves which data flow can realistically be pushed into it.

Present top-of-line FPGA's already come in $400 \sim 500$ pin packages. We can realistically imagine to send $200 \sim 300$ bits to a given chip in each clock cycle and still be on the safe side. How short can a clock cycle be? Here we do not expect much improvement in the near future and we assume to be still using present printed circuit board technology. We can imagine 100 MHz to be the upper limit to the operating frequency of a board. A 10 ns clock cycle is not unrealistic given that we are already building electronic boards with 25 ns clock cycles and 20 ns is well within present day technology. 10 ns is an easy round number to work with and so we choose it.

Putting this together, we take the following working hypothesis:

a "VME size" electronic board for a CMS hardware tracker will process $200 \sim 300$ *bits every 10 nanoseconds, i.e.will work at* $20 \sim 30$ *Gbit/sec.*

3.2 A life-size exercise: the CMS tracker barrel.

As basic example we take one typical LHC event in the CMS detector and work out all the relevant data flows.

3.2.1 detector geometry

For the detector we take an only slightly simplified version of the V3 geometry of the tracker barrel: we treat all barrel layers as smooth cylinders (much more like V4) at the (weighted) average radius of the seven V3 wheels, and assume a perfect tiling of each cylinder into the corresponding detectors with no overlap.

The elemental detector is a 5.12×12.5 cm silicon rectangle or a 10.24×25 cm MSGC. Number for pixels are quite tentative at this time, we try to make our best estimate for those as well, but we focus on silicons and microstrip gas chambers for the time being. We end up with the channels/detector count indicated in Table 1.

LAYER				DETECTOR							
	R cm	l (z) cm	Area m2	l (z) cm	pitch micron	Nchan per det	width cm	area cm^2	Ndet	Tot Nchan	bits
pix 1	7	52	0.2	1.60	125	65536	1.60	10.24	223	1.E+07	24
pix 2	11	52	0.4	1.60	125	65536	1.60	10.24	351	2.E+07	25
TOT PIXEL BARREL									574	4.E+07	26
si 1	23.2	175	2.5	12.5	66.7	768	5.12	64	398	3.E+05	19
si 1 stereo	23.2	175	2.5	12.5	200	256	5.12	64	398	1.E+05	17
si 2	30.9	175	3.4	12.5	66.7	768	5.12	64	531	4.E+05	19
si 2 stereo	30.9	175	3.4	12.5	200	256	5.12	64	531	1.E+05	18
si 3	38.7	175	4.3	12.5	100	512	5.12	64	664	3.E+05	19
si 4	46.1	175	5.1	12.5	100	512	5.12	64	792	4.E+05	19
TOT SILICON BARREL									5700	2.E+06	21
msgc 1	64.1	225	9.1	25	200	512	10.24	256	354	2.E+05	18
msgc 1 stereo	64.1	225	9.1	25	200	512	10.24	256	354	2.E+05	18
msgc 2	72	225	10.2	25	200	512	10.24	256	398	2.E+05	18
msgc 3	79.9	225	11.3	25	200	512	10.24	256	441	2.E+05	18
msgc 4	87.9	225	12.4	25	200	512	10.24	256	485	2.E+05	18
msgc 4 stereo	87.9	225	12.4	25	200	512	10.24	256	485	2.E+05	18
msgc 5	95.8	225	13.5	25	200	512	10.24	256	529	3.E+05	19
msgc 6	103.8	225	14.7	25	200	512	10.24	256	573	3.E+05	19
msgc 7	111.7	225	15.8	25	200	512	10.24	256	617	3.E+05	19
msgc 7 stereo	111.7	225	15.8	25	200	512	10.24	256	617	3.E+05	19
TOT MSGC BARREL									4237	2.E+06	22
TOT INNER BARREL									9937	4.E+06	22

Table 1: V3 simplified geometry used in this study

The last column indicates how many bits are needed to uniquely identify one channel in each layer (Nbits=$\log_2$(total number of channels in each layer)).

Now we assume that the hardware tracker will have a first stage, the Global Tracker of figure 1. It does not work at full detector resolution, but instead uses some coarse grain segmentation of the tracker detectors, i.e. the "SuperStrips", or SS for short, already defined in section 2.

How coarse is this division, i.e. how large is a SuperStrips ?

The basic property of the SuperStrip is that its size is chosen for pattern recognition, not for resolution.

A size of few millimeters is chosen for this first exercise. The choice is justified in the following sections. The size and number of SuperStrips are reported in Table 2 together with the number of bits needed to uniquely identify one SS in each layer (the SS address).

Note that for silicon strips and MSGC's a SuperStrip is a rectangle with the same length as the strips (5th column of Table 2) and width as indicated in Table 2, while for the pixels a SuperStrip is a square with the indicated width as side.

LAYER	**S**UPER**S**TRIP			
	SuperSTRIP width mm	area cm^2	N SS	bits
pix 1	1	0.01	228701	18
pix 2	1	0.01	359388	19
TOT PIXEL BARREL				
si 1	2	1.02	24875	15
si 1 stereo	2	1.02	24887	15
si 2	2	1.02	33189	16
si 2 stereo	2	1.02	33206	16
si 3	2	1.02	41506	16
si 4	2	1.02	49514	16
TOT SILICON BARREL			207177	18
msgc 1	5	5.12	17699	15
msgc 1 ste	5	5.12	17699	15
msgc 2	5	5.12	19880	15
msgc 3	5	5.12	22061	15
msgc 4	5	5.12	24270	15
msgc 4 ste	5	5.12	24270	15
msgc 5	5	5.12	26451	15
msgc 6	5	5.12	28660	15
msgc 7	5	5.12	30841	15
msgc 7 ste	5	5.12	30841	15
TOT MSGC BARREL			211831	18
TOT INNER BARREL			419008	

Table 2: SuperStrip numerology

3.2.2 SuperStrips choice: pattern bank size and SS occupancy

A realistic possibility for the Global Tracker is to be a very fast device that works using pre-calculated "templates" or "patterns" stored in a "pattern bank". For the exact definition of "patterns" and "pattern bank" see the reference [2]. With these hypothesis, the SS has to be:

- **fat enough** so that the number of possible valid combinations (patterns) is manageable, i.e. of the order of 10^6. The absolute minimum width is the detector resolution (few tens

of microns), this would mean to completely solve pattern recognition at the SuperStrip level (there would be no more information to add!), likely too ambitious a goal. The number of patterns is rapidly increasing as a function of the number of SuperStrips per layer. For example in the CDF case 250μm SS mark the limit of what can be handled. Since CMS has so much more latency, pattern recognition at SS level does not need to be pushed as far as in the CDF case, we imagine that more work can be left to later stages, and we will assume that CMS can use sa SuperStrips 10 times wider, i.e. $2 \sim 5$ mm (also CMS detectors sit at larger radii then the CDF SVX, wider SuperStrips still cover the same angular range). This is corroborated by an independent study of the complexity of the pattern combinatorics in the CMS tracker [3]. This study indicates that, when limiting oneself to the axial layers, SuperStrips a few mm wide allow to build a pattern bank of the order of a few million patterns that contains more than 90% of the interesting physical tracks. In particular that study suggests SuperStrips 2 mm wide in the silicons and 5 mm wide in the MSGC's as the best candidate.

- **thin enough** so that the number of found *roads* and their complexity (number of hits inside the *road*) are acceptably small for the downstream algorithm, that is the Local Tracker. The Global Tracker is a real advantage only if a reasonably low fraction of the patterns in the pattern bank are fired in the event, i.e. the number of fakes is not too high. In this case the first stage will really apply a positive reduction to the large amount of data. We do not know yet the number of fakes as a function of the *road* size. However we can estimate the SS occupancy as a function of the *road* size. This quantity can be used to evaluate how much the average event is confused due to the lost resolution. Even 100% of the stored patterns can be found into most of the events if too large a SS is chosen! In this case the Global Tracker would not apply any reduction to the tracking information (actually will increase the amount of data). We believe the segmentation should be such to give an SS occupancy definitely lower than 50%. Lower is the SS occupancy, lower will be the number of fake *roads* and the Local Tracker load. We will see in a later section that our choice for the SS size gives occupancies from $11 \sim 12\%$ in the inner silicon layers, to 3% in the outer layers.

By these convergent arguments, we take 2mm in Si and 5mm in MSGC as the baseline SS size.

3.2.3 SuperStrip occupancy evaluation

We take as input for our estimate the recent occupation study [4] in which the typical CMS event is obtained as pile up of many Poisson distributed minimum bias interactions (on average 37.5 single minimum bias events in the MSGC's and half as many in the silicon strip detectors) and one 300 GeV $b\bar{b}$ jet pair using full CMSIM simulation.

There are many uncertainties in this kind of studies, and past experience with hadron colliders has shown that is quite difficult to reproduce the observed multiplicity in a simulation that has not been tuned on the data themselves. In this case it appears reasonable to assign a factor 2 as boundary to the uncertainty [5]. We just take the Monte Carlo occupancies at face value and we leave to the reader to put his/her own guess at the uncertainty.

Specifically, we take Table 14 of [4] as starting point. We assume that the tracker front-end electronics (the FED board) will provide one cluster, "hit", for each reconstructed hit in this

table.

Table 3 shows the results of the Monte Carlo study in the first three columns, where we report from [4] the following quantities for each layer of the tracker barrel:

1. the layer under consideration
2. the percentage of detectors in that layer that are traversed by at least one simulated track (hit detectors)
3. the average number of reconstructed (i.e. after clustering in the FED) hits in the hit detectors

In the next two columns in the center of the Table, we compute the SuperStrip occupancy. The number of fired SuperStrips in that layer is obtained as the products of column 2 and 3 and of the number of detectors in a layer from Table 1. The percentage of fired SuperStrip in the hit detectors gives the "hit detector SS occupancy" in the next column. To evaluate the SS occupancy in the barrel, the "hit detector SS occupancy" has to be moltiplied by the percentage of hit detectors, that is column 1 of the same table.

The right part of Table 3 will be described in the next section.

Important note: the Monte Carlo study we refer to does not give numbers for the pixels, so here the pixel rows have been filled backwards ! I.e. we started from the end, by assuming that we will want a similar data flow for pixels and silicons and computed the corresponding number of hits that can be tolerated. We put in by hand that 100% of the pixel detectors are crossed by one track. One will see in the end that the pixel occupancy needed to prevent them from overwhelming the data flow is quite in agreement with expectation. Eventually all of this will have to be simulated in better detail, of course.

We make the following observations:
i) on the average one half of the detectors are hit
ii) on the average each detector has 3 hits in $5 \sim 10$ cm
iii) outer layer detectors are hit less (20%) than inner layer ones

We already can say something about how large the SuperStrips have to be.

1. The overall picture of the CMS tracker is sort of uniformly lighted, with hits scattered all over the place with a low average density. This is the main feature we see when looking at data flow: the bulk of the data come from the uniformly spread min-bias events. The high P_t objects (jets e.g.) will be the hardest task for pattern recognition, but make very little impact on how much data a hardware tracker, working on reduced resolution, must handle.
2. Since on average there is one hit every few cm, the SS can be several mm wide and still be "thin enough". Probably there is no reason to make them smaller, since occupancy seems to be not too high.
3. We don't know yet which will be the average pile-up of hits in the SuperStrip. The pile-up gives the advantage of a reduction in the amount of data to be sent to the Global Tracker.

LAYER	%hit detect	Nhit /det	hit SS	SS occ %	bits per ev	Mb/sec /layer	ns/SS layer
pix 1	100	5	1117	31%	20106	2011	9
pix 2	100	3	1053	19%	20007	2001	9
TOT PIXEL BARREL			2170		40113	4012	5
si 1	87	3.2	1108	12%	16620	1662	9
si 1 stereo	87	3.2	1109	13%	16635	1664	9
si 2	76	2.4	969	9%	15504	1550	10
si 2 stereo	76	2.4	969	9%	15504	1550	10
si 3	64	1.9	808	7%	12928	1293	12
si 4	51	1.6	646	6%	10336	1034	15
TOT SILICON BARREL			5609		87527	8753	2
msgc 1	66	3.9	911	19%	13665	1367	11
msgc 1 ste	66	3.9	911	19%	13665	1367	11
msgc 2	56	3.6	802	18%	12030	1203	12
msgc 3	45	3.3	655	16%	9825	983	15
msgc 4	35	3.2	544	16%	8160	816	18
msgc 4 ste	35	3.2	544	16%	8160	816	18
msgc 5	29	3.1	476	15%	7140	714	21
msgc 6	24	3	413	15%	6195	620	24
msgc 7	19	2.9	340	14%	5100	510	29
msgc 7 ste	19	2.9	340	14%	5100	510	29
TOT MSGC BARREL			5596		83940	8396	2
TOT INNER BARREL			11205		171467	17149	4

Table 3: Detector occupancy and SuperStrip occupancy and flow

As long as SS are much thinner than a detector, the low average multiplicity inside each detector could mean that there is in general very little pile-up of hits in each SuperStrip: to be conservative we assume zero pile-up, that is each SS will correspond to one hit (and viceversa). This hypothesis is in agreement with preliminary analysis of correlations among the reconstructed hits in this Monte Carlo study [6].

4. The SuperStrip cannot be too large. For example if they are more or less as large as one detector about 50% of the SuperStrips would be hit, which makes the pattern recognition perspectives pretty grim.

The percentage of hit SuperStrip (SS occupancy) in Table 3 is a very important information. If this number were to be too high, too many patterns would be fired by fake combinations. The present value is about 10% for the worse layer, a bit high, but probably reasonable. The only way to reduce the percentage of occupied SuperStrips is to make them thinner. This has very little impact on the data flow described in the next section, (on average we need 15 bits to identify one SS in a given layer, if we make SS four times thinner, they would become 17 bits, a 10% increase, so the overall data flow would change by that much) but may have an enormous effect on the number of possible patterns that need to be stored in the pattern bank.

In the end the Monte Carlo study justifies the choice of 2 and 5 mm for the SuperStrip width as a reasonable first approximation.

3.2.4 SuperStrip data flow

Starting from the total number of hit SuperStrips per layer in each event and multiplying them for the number of bits necessary to encode the SS coordinate (see Table 2), we obtain the overall data flow per event, and per second in any given layer. All these numbers are reported in Table 3. The data flow for each layer is also computed, in the last column on the right, as number of nanoseconds available for each SuperStrip. This number is more instructive than the Mb/sec at its left, and needs a bit of clarification: the data flow in bits/sec is not very useful once we go to the elementary chip, here we have to transmit one SuperStrip at a time for each layer, so the fundamental parameter is how frequently the chip has to accept a new input in order to keep up with the flow, i.e. the minimum clock frequency at which the system has to operate. This is the meaning of the number in the "ns/ss" column.

As we see, our hypothetical tracking chip has to accept a new word (SS) every few ns, if it has to look at all the barrel. This looks too aggressive, especially giving the large uncertainty on this study, on the other hand, by simply dividing the barrel in four quadrants 90^0 each and assuming that tracking will be performed independently in each of them (a very small loss in efficiency, if any), we obtain a very safe situation in which the elemental chip of the Global Tracker needs to process a new word every 35 ns or more.

3.2.5 DAQ data flow

Finally we compute the data flow to the RDPM's for the "standard" CMS DAQ link for our "typical event". This calculation both provides a check that our exercise is in agreement with the expectations and specifications for the CMS DAQ, and provides a measurement of how much collecting the SuperStrip information adds to the overall data flow. We do not expect SuperStrip to be a minor perturbation, since we assumed one hit SS for each detector hit.

The question here is: how much data is logged to the RDPM's for each detector hit ? While data format is not defined yet, we need a work hypothesis to make the calculation. We assume the following: for each hit, the FED finds the corresponding cluster of detector strips and provides in output:
i) the address of the first strip of the cluster (number of bits as from Table 1)
ii) the cluster length (3 bits)
iii) the ADC pulse height for each strip in the cluster (8 bit each)

The cluster length is taken from Table 14 of [4], for silicon stereo layers we arbitrarily assign a reduced cluster width since the pitch is bigger.

We also compute the needed number of FED's assuming one FED for each 16k tracker channels, and one for each 31250 pixels. Numbers are from [7], we also understand that the latest FED specs changed the number of channels per link from 512 to 1024.

From that we obtain the data flow for FED in MBytes/sec, to be compared to the CMS spec of 100 MB/sec (upgradable to 400).

The results are shown in Table 4, where we see that in average the SuperStrip stream adds 30% to the tracker DAQ flow. On the other hand of this rather large number, the overall data flow appear well within the FED specs. In particular we note that the detector occupancy in terms of number of strips whose ADC counts has to be transmitted to the RDPM's is about 1% for

silicons and microstrip, and 0.02% for pixels, in agreement with (actually somehow lower than) current CMS estimates.

LAYER	clust width	Nhit /layer	bits/ event	Mb/sec /layer	SSflow /hitflow	occup ancy	N FED	MByte/s /FED	MByte/s /FED add SS
pix 1	2	1117	49148	4915	41%	0.02%	7	88	124
pix 2	2	1053	47385	4739	42%	0.01%	12	49	70
TOT PIXEL BARREL		2170	96533	9654	42%		19	64	91
si 1	4.8	1108	68031	6803	24%	1.70%	19	45	56
si 1 stereo	3	1109	49905	4991	33%	3.30%	6	104	139
si 2	4.9	969	60272	6027	26%	1.20%	25	30	38
si 2 stereo	3	969	44574	4457	35%	2.10%	8	70	94
si 3	4.3	808	46379	4638	28%	1.00%	21	28	36
si 4	4.4	646	37597	3760	28%	0.70%	25	19	24
TOT SILICON BARREL		5609	306758	30676	29%		104	37	48
msgc 1	1.9	911	33889	3389	40%	1.00%	11	39	55
msgc 1 ste	1.9	911	33889	3389	40%	1.00%	11	39	55
msgc 2	1.9	802	29834	2983	40%	0.70%	12	31	44
msgc 3	1.9	655	24366	2437	40%	0.60%	14	22	31
msgc 4	1.9	544	20237	2024	40%	0.40%	15	17	24
msgc 4 ste	1.9	544	20237	2024	40%	0.40%	15	17	24
msgc 5	1.9	476	18183	1818	39%	0.30%	17	13	18
msgc 6	1.9	413	15777	1578	39%	0.30%	18	11	15
msgc 7	1.9	340	12988	1299	39%	0.20%	19	9	13
msgc 7 ste	1.9	340	12988	1299	39%	0.20%	19	9	13
TOT MSGC BARREL		5596	209400	20941	40%		152	17	24
TOT INNER BARREL		11205	516158	51617	33%		255	25	33

Table 4: Detector occupancy and Data flow

In Table 4 the data flow in-out of the FED is abnormally large for the silicon stereo layers. This is because in our exercise we assigned the FED's on the base of the channel counts (as we understood from our reading of the current CMS documents). In a situation where occupancy is dominated by physics (i.e. after clustering inside the FED) the stereo layers are hit more often since for the same number of channels they cover a larger area due to the wider strip pitch.

3.3 The tracker output

Given that the data from the front-end to our hypothetical real-time tracker is an affordable flow, what about the data from the device to the DAQ ?

It is an easy question, by definition the tracker has to make a large reduction of information and output a limited list of track primitives, of the same order of magnitude as the number of real tracks, i.e. a few thousand words. That much data will never be a problem, even if were to be ten times as large.

If this were not the case, it would mean that the tracker is useless, and it would not exist. So the output data will never be problem, one way or another.

3.4 Exercise summary

One quarter of the tracker barrel is definitely a large enough section of the CMS detector, that it makes sense to have a single device that perform on-line tracking just for that region.

We just saw that the needed data flow is about $2 \sim 3$ Gb/sec for each barrel layer. So it is definitely within near future possibilities to bring to a single device all data from e.g. ten layers while keeping up with the input data rate from the Level 1 trigger (100 kHz).

Thus we found a positive answer to the questions at the beginning of this section.

The next section will deal with what to do with these data.

4 Is it feasible ?

Given that the proposed device can be fed the needed data, can it really be built ? Is it possible to solve the track finding problem with a dedicated hardware device that does not have all the subtleties of a full fledged offline algorithm? Would this device operate in a reasonable time at an affordable cost ? May a realistic affordable dedicated device provide the needed precision? The actual performance of a hardware on-line tracker still has to be addressed. Which efficiency could it reach ? Which fake rate ? Which resolution ? The size and complexity of the device, and the associated cost, may make it non-competitive anyhow. We do not know yet.

No definitive answer is available now, but preliminary studies look promising.

We present here a couple of possible implementations, mainly to put in evidence the few changes that should be implemented in the DAQ system since the beginning. These are only examples, and better solutions can be found. However it is useful to have a concrete starting point, to trigger a concrete discussion and gain confidence that this kind of device can be done.

Figure 2 shows one of the possible implementations. Here the Global Tracker is partly mixed with the existing DAQ, represented by the FEDs, the Concentrators and the RDMP PHs boxes, and partly built as new devices connected to the DAQ (the top and bottom part of the figure outside the Layer Crate boxes) represented by the Associative Memory (AM), the ROADs+HITs RDPM (ROAD_RDPM) and the MERGERs.

The MERGERs implement simple multiplexing functions and are necessary only if the number of links Concentrators-AM on the top of the figure and Concentrators-ROAD_RDPM on the bottom is too high.

The generic algorithms of figure 1 are replaced by the particular ones adopted by CDF, for which prototypes have been built already.

An Associative Memory (the Road Finder of figure 1) [2] receives data from many parallel streams, one per layer. It is not convenient to feed too many layers into a single AM, since the hardware would be uselessly complicated. More than one Associative Memory can work in parallel, to find segments of six/eight layers each, segments that can be linked together to reduce the number of fakes. If the number of fakes is not a problem for the downstream algorithm, a single AM working on six/eight layers can be built. The most clean layers can be used, like the external MSGC's and the Pixels to define *roads* and to collect the right hits from all layers, even from the layers not used in the AM.

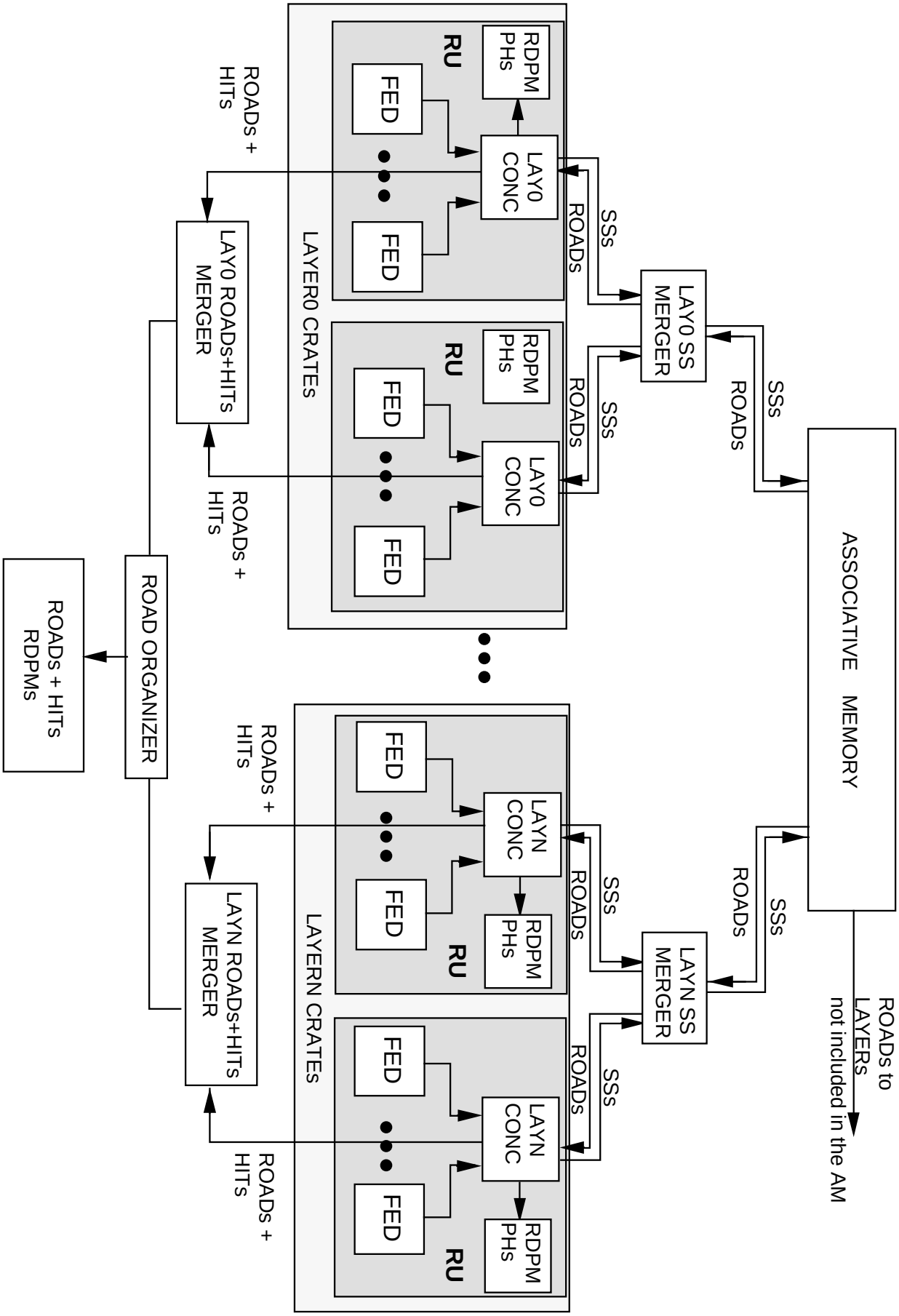

Figure 2: The Global Tracker inserted in the DAQ

To have a feeling of how much hardware the AM will require, let us consider that now in CDF two technologies are being developed. Both give a density of 16 kpatterns per 9U VME board for a detector of 6 layers of 4 Kbins per layer. One technique uses a full custom VLSI chip [8] built with old 0.7 micron two metal layer technology. The other exploits an FPGA [9] manufactured in a 0.6 micron three-layer metal process which is now similarly old.

If we suppose that technology will allow an improvement of a factor at least 50 for a device ready in year 2003-2004 (manufacturer's extrapolations seem to guarantee that factor by 2000), we can think possible to have at least 800 kpatterns/board. Using the preliminary results shown in reference [3] a quart of barrel with 2 millimeter SuperStrips in the silicon detectors and 5 millimeter SuperStrip in the MSGC's would require a pattern bank of 4 Mpattern, this corresponds roughly to 5 boards. With such a bank, *roads* compatible with tracks of Pt as low as 2 GeV, produced in a luminosity region of few millimeters around the detector center, can be found with an efficiency of 90%. Higher efficiencies will require larger and larger banks.

These 5 boards do not need to receive HITs in parallel, but they can work in pipeline, passing data from one to another and thus exploiting the long Level 2 latency to avoid fan-out problems. In conclusion the Global Tracker for the barrel could be a set of 4 AM working in parallel for a total of 20 boards. Similar estimates can be done for pattern recognition in the non-central detectors.

The other important function that has to be implemented is the "data organizer" function [10] . In figure 2 the "data organization" is performed in parallel by the "Concentrator" modules. Each Readout Unit could have such a module. The concentrator would receive the list of clusters from the FEDs. Each cluster will be a packet of words, including the centroid, the number of strips over threshold and the pulse heights of each of them. If each FED treats an integer number of SuperStrips (16000 channels could correspond to a maximum of 1000 SuperStrips) the FED itself should calculate how many clusters are contained into each SuperStrip. Than the clusters of the same SuperStrip should be collected in a packet of words containing:

1. the SuperStrip pointer;
2. the number of clusters inside the SuperStrip;
3. the list of clusters;

If the increase of data to be transferred from the FEDs to the Concentrator (SuperStrip pointer and number of clusters inside each SuperStrip) is too much, all the "SuperStrip work" can be done in the Concentrator: the SuperStrip pointers can be stored there and the number of clusters can be calculated there.

The Concentrator would pass all the pulse heights to the PHs RDPM where the event will sit until a more sophisticated clustering calculation will be necessary or when the event is selected to go on tape. The centroids (hits) will be saved locally in the Concentrator Hit List Memory which is organized into cells, one per SuperStrip. Each hit is written into the cell corresponding to the SuperStrip it belongs. The SuperStrip pointers are sent to the AM (only once, even if more than one cluster belongs to the SuperStrip) and the number of hits per SuperStrips is stored internally in the Concentrator. When the *road* number is back from the AM, the number of hits and the *road* pointer is used to immediately fetch the full resolution hits inside each interested

SuperStrip and send them to the ROADs+HITs RDPM. The Concentrator has to store as many events as can be at the same time in the Global Tracker pipeline. The concentrator will be simpler or more complex, depending on how much work can be done inside the FED. The total number of FEDs per quart of layer from the previous section seems to be not larger than six. This means that probably no more than two concentrator will be necessary per quart of layer.

Figure 3 shows a second possible implementation very similar to the first one. In this case the existing DAQ scheme and the Global Tracker are more independent, since the new devices should receive the necessary amount of data (SS pointers, number of clusters and list of clusters for each SuperStrip) from an independent FED connector. The second scheme reduces the contact between the DAQ and the new device. However the number of links is duplicated.

4.1 Requirements fro CMS DAQ

In conclusion we like tracking data to be organized in the FED crates in such a way that the system represented in figure 2, or 3, or some equivalent system could be implemented.

It is fundamental that each 1/4 or 1/8 of layer (not only the barrel, but also the forward and the backward detector data could be fed into a Global Tracker) corresponds to an integer number of crates located in such a way that the cabling with the Global tracker is reasonable. One slot should be left free to allocate an extra module if the Data Organizer is not integrated in the DAQ Concentrator.

The integration of some necessary functions in the FEDs and the Concentrators can help a lot to reduce the necessary number of links and to avoid duplication of functions.

5 Conclusion

6 References

References

[1] S.Belforte et al., The CDF trigger Silicon Vertex Tracker (SVT), IEEE Trans. on Nucl. Science 42, 860 (1995); SVT Silicon Vertex Tracker TECHNICAL DESIGN REPORT, CDF Note 3108, November 22, 1994, Fermilab.

[2] M.Dell'Orso and L. Ristori, VLSI structures for track finding, NIM A278 (1989), 436-440.

[3] R. Carosi et al. A test for pattern recognition in CMS, DRAFT CMS Note.

[4] T. Kachelhoffer. Occupancy Studies for Minimum Bias and b Bar b Jets Events with the V3 Geometry of the CMS Tracker. CMS NOTE 1997-088. We used a draft kindly provided by the author on October 27, 1997.

[5] A. Caner, private communication. November 1997.

[6] G.Dissertori, private communication. November 1997. We thank Guenther for having produced for us several instructive histograms.

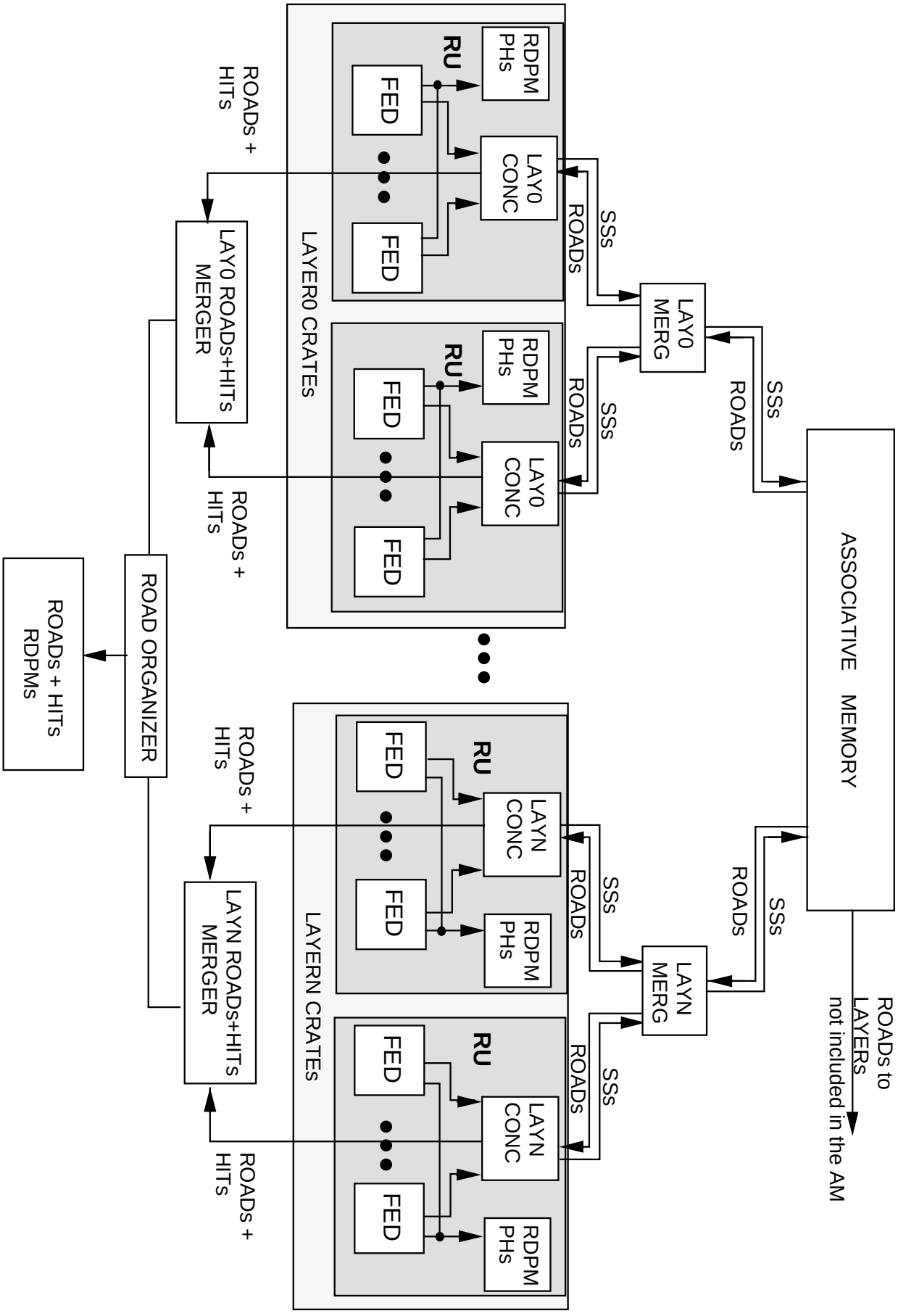

Figure 3: The Global Tracker receiving data from the DAQ

[7] S.Cittolin et al. Front End Driver in CMS DAQ. CMS note TN/95-020. Revision 1.03. February 1995.

[8] S.R. Amendolia et al., The AMchip: a Full-custom CMOS VLSI Associative Memory for Pattern Recognition. IEEE Trans. on Nucl. Sci., Vol. 39, N. 4, August 1992.

[9] P. Giannetti, “The Programmable Associative Memory”, SVT note 88, December 1998; P. Giannetti et al.“The Programmable Associative Memory”, to be submitted to NIM.

[10] S. Belforte et al., The SVT Hit Buffer, IEEE Trans. on Nucl. Sci., Vol. 43, N. 3, June 1996, 1810-1813.