A Real-Time Tracker for Hadronic Collider Experiments

A. Bardi[2], S. Belforte[2], M. Dell' Orso[1,2], S. Galeotti[2], P. Giannetti[2], E. Meschi[3], F.Morsani[2], F. Spinella[2]

[1]Dipartimento di Fisica, Università di Pisa, Piazza Torricelli 2, 56100 Pisa, Italy

[2]INFN Pisa, Via Livornese 1291, 56010 S. Piero A Grado (PI), Italy

[3]CERN/Div. PPE, CH-1211 Geneva, Switzerland

Abstract

In this paper we propose highly parallel dedicated processors, able to provide precise on-line track reconstruction for future hadronic collider experiments. The processors, organized in a 2-level pipelined architecture, execute very fast algorithms based on the use of a large bank of pre-stored patterns of trajectory points.

An associative memory implements the first stage by recognizing track candidates at low resolution to match the demanding task of tracking at the detector readout rate. Alternative technological implementations for the associative memory are compared.

The second stage receives track candidates and high resolution hits to refine pattern recognition at the associative memory output rate. A parallel and pipelined hardware implements a binary search strategy inside a hierarchically structured pattern bank, stored in high density commercial RAMs.

I. INTRODUCTION

Hadronic collider experiments require large online computing power to reach the enormous rejection factor necessary to select events to be written on tape. One of the most demanding tasks is usually the track reconstruction from measured points (detector *hits*) of particle trajectories. In this paper we propose the use of highly parallel dedicated processors to efficiently execute two fast track finding algorithms [1, 2]. The algorithms are based on the idea of a large bank of pre-calculated hit patterns to be compared to the event [3].

The proposed system is an evolution of the Silicon Vertex Tracker (SVT) [4] currently being built for the CDF experiment at Fermilab. The CDF tracker processes events with a 100 kHz input rate, and an overall allowed processing time (latency) of 10 μsec. Hits from five vertex detector layers can be linked to segments observed in the tracking chamber to reconstruct real time tracks precisely enough to measure *b* quark decay secondary vertices.

Next generation hadronic collider experiments can exploit technology advancements to realize a more powerful system that can work with the same performances on more complex tracking detectors. The availability of very large, low cost memories allows buffering of many large events. Therefore, new experiments' triggers can have a long latency time and pipelining can be used extensively to subdivide the complex pattern recognition into simpler sequential steps with increasing degrees of approximation.

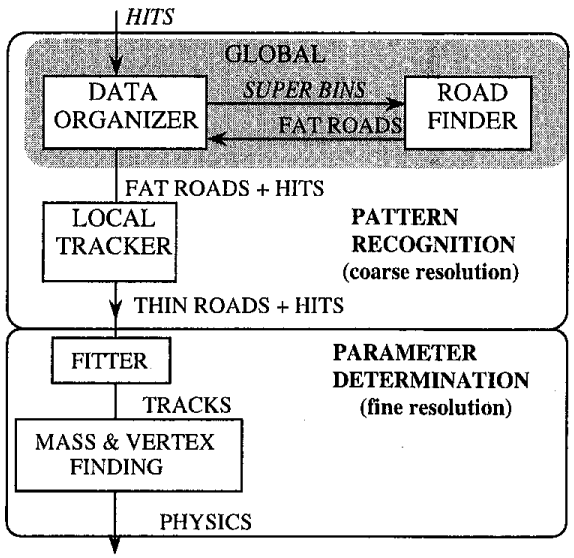

Figure 1: The process of finding track-based physical quantities starting from detector hits.

Figure 1 outlines a computational architecture that takes off from detector hits and goes up to compute track-based physical quantities such as invariant masses and decay vertices. There are significant advantages in splitting the overall computation in two sections (*coarse resolution* and *fine resolution*) and applying different mixes of hardware and software technologies for their implementations. High performance commercial CPUs are the best choice for the *fine resolution* section, where flexibility is a great advantage to handle many variables and specific situations such as local corrections, alignment effects, exceptions etc. Resolutions of few tens of microns are obtained with the use of large data bases updated on a run by run basis. On-line use of standard CPUs can exploit offline software. On the other hand, the *coarse resolution* section has a lot to gain from the extremely large performance increase obtained from dedicated hardware since a huge combinatorial problem must be solved to filter out few relevant track hits from the plethora of detector data. A large fraction of CPU time is usually wasted in data sorting or similar tasks consisting of multiple repetitions of simple actions, such as random accesses to large memories, that could fit well in dedicated highly parallel architectures.

In the rest of this paper we describe a number of hardwired blocks that actually implement the coarse resolution segment of a track finding processor. Part of these blocks have been designed and developed taking into account the requirement of

0018-9499/99$10.00

present generation experiments and the constraints of present technology. The technology improvements in the next 9-10 years is expected to cover almost automatically a large part of the future experiment complexity increase.

II. A Parallel Architecture

The Global Tracker of figure 1 performs the most CPU consuming part of the pattern recognition. It splits the problem of finding tracks inside the whole detector into many simpler problems of finding tracks inside detector "slices" called fat roads. Depending on applications, the fat road width spans from 100 μm to 100 mm. The Global Tracker consists of a Road Finder and a Data Organizer.

The Road Finder looks for low resolution track candidates in the whole detector. The resolution must be low enough to make the problem solvable at high rate, (for this reason the track candidates are called fat roads). At this level, multiple hits inside a road are treated as a single hit, allowing multiple unresolved tracks.

The Data Organizer is a high-speed data traffic node between the detector, the Road Finder, and the Local Tracker. At full rate (e.g. 30 MHz in CDF [5]) the Data Organizer performs the following operations: (a) receives full resolution detector hits in any order, (b) buffers them in an internal database, (c) sends low-resolution hits called super bins to the Road Finder (the super bins are obtained by logically ORing a number of adjacent detector bins or channels), (d) receives fat roads from the Road Finder and fetches from the internal data base all the detector hits contained in the fat roads, (e) sends each fat road with its set of full resolution hits to the Local Tracker.

Fat roads usually contain several unresolved tracks. The Local Tracker is powerful enough to deal with large roads full of tracks at the same rate of the Global Tracker's output, but with higher resolution. The thin roads, output by the local tracker, are narrow enough to hold only few hit ambiguities, if any. These ambiguities can be solved by sequentially fitting the residual hit combinations and choosing the best fit. The width of the fat roads must be optimized for the characteristics of the specific experiment. Too small or too great widths would require intolerably high performances respectively to the Global Tracker or to the Local Tracker.

We propose the use of an Associative Memory (AM) [1] as Road Finder and of a Tree Search Processor (TSP) [6] as Local Tracker.

III. The Associative Memory

AM is a dedicated device where parallelism is pushed to the maximum level since each stored hit pattern is provided with its private hardware necessary to compare itself with the event. Matching patterns are identified by outputting their addresses. The device is so powerful that tracks can be found during the detector readout. This feature makes the AM a perfect road finder.

The AM pattern bank is limited by the size of the hardware, mainly consisting of low-density custom memories. However, the coarse resolution of the Road Finder prevents divergence of the pattern bank size even for the complex detectors of the next generation colliders.

A. *The Associative Memory as a pipeline*

AM is physically composed of many boards organized into a pipeline (see figure 2). Different boards can work on different events.

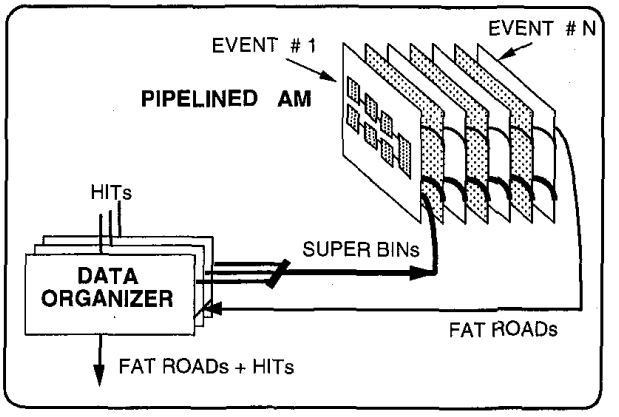

Fig. 2: A pipeline of AM boards fed by a set of Data Organizers.

The pipeline receives data from a number of independent streams. Each stream is driven by a Data Organizer that handles hits related to one or more detector layers, depending on the available time to feed AM with an event. For example the data from the most crowded layers can be fed into the AM using a stream per each layer, while the most empty layers can be merged on a single stream. The processing event rate is set by the stream that transfer the larger average number of hits per event.

Figure 3 shows a schematic view of an AM board. Each AM board receives data from the preceding board and send them to the next in the pipeline. Hits and Roads belonging to different events are separated by a *start* and an *end event words*. The Data Organizers process asynchronously the events that have to be synchronized at the input of each AM board. Each board input is provided of deep FIFOs for this goal. If a FIFO becomes Almost Full a HOLD signal is sent to the board feeding the FIFO and the data flow is suspended until the FIFO is less than Almost Full.

When an AM board starts to process an event, the hits are popped in parallel from the input stream FIFOs. They are sent at the same time to the Local AM Bank (LAMB) and to the output registers that drive the connections to the next board FIFOs. Data from the different streams are checked for consistency: if they belong to different events a severe error is issued and the whole system needs to be synchronized again.

As soon as hits are downloaded into the LAMB, the locally found roads are accumulated into the LAMB FIFO. When the end event word from a particular hit input stream is received, no more words are popped from the relative FIFO until the end event word is received from all the other hit streams. When all the hit end event words are received, the event is completely fed into the LAMB that in few clock cycles provides the last found roads. As soon as all the LAMB roads are loaded into

the LAMB FIFO, an end event word is added to close the list, the LAMB is reset and a new event is downloaded.

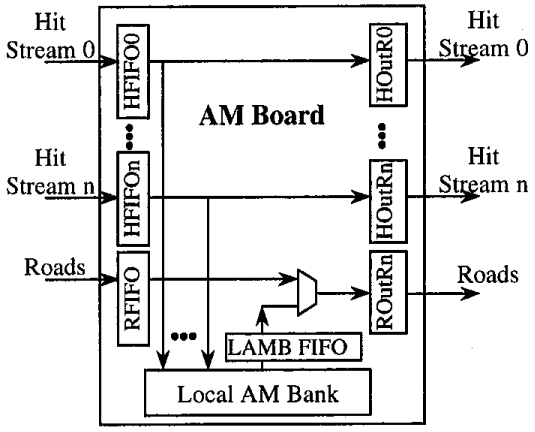

Figure 3: Schematic view of an AM Board.

Roads have to cross the whole pipeline to go back to the Data Organizers. Inside each AM board the roads coming from two different streams are accumulated into two different FIFOs and multiplexed on a single output. The Road input stream fills the road input FIFO (RFIFO) and the Local AM Bank fills the LAMB FIFO. Roads coming from the previous board or found internally by the LAMB can be mixed if they belong to the same event. Thus the AM board pops them from the FIFOs to the output register as soon as they are available, in any order. However, if an end event word is received from one of the two streams, no more roads are popped from the corresponding FIFO until the end event word is received also from the second stream. When both the end event words have been received an end event word is sent to the road output register.

B. *LAMB implementations*

CDF plans to use a full custom VLSI [7], built with 0.7-μm two-layer metal technology, to house a bank of 4×10^5 patterns in 24 9U-VME boards (AM-boards) for a detector of 6 layers, with 48000 250μm wide super bins per layer. An AM-board with 128 custom chips, 128 patterns per chip, was built and tested at 30 MHz [8].

Field Programmable Gate Arrays (FPGA), an easy-to-upgrade technology, has been recently used to design a Programmable Associative Memory (PAM) [9]. PAM offers a high degree of flexibility, a simple architecture, a prompt exploitation of technology advancement, and an easy prototype testing and debugging.

Patterns are stored in the FPGA at power on, when the configuration bit-stream is downloaded in the chip.

A 9U-VME board assembled with PAMs can house the same number of patterns as the CDF AM-board [9].

To simplify input/output operations, the PAM chips on a PAM board are grouped into PAM units composed of 64 chips. A prototype 9U-VME board has been implemented to test a single PAM unit. The chips of a PAM unit are located on both board sides, 32 chips per side. The physical size of a PAM unit is only 4.4" × 11" and three of them fit on one board to reach the same pattern density of the CDF AM-board. Details of the prototype are reported into [9].

Patterns were downloaded by programming the FPGAs through the VME controlled JTAG port. Chains of eight PAMs each are downloaded in parallel to limit programming time. The VME 32-bit wide data transfer allows to program 32 chains in parallel for a total of four PAM units per board. Downloading time was measured to be few seconds, dominated by the ethernet data transfer from the host computer to the VME controller. The board, provided with a CDF compatible interface, has been successfully tested at 30 MHz in a CDF level 2 trigger test stand [8].

A high-level language representation of the associative memory allows at any time a fast synthesis of the project in a common gate array or in an ASIC standard cell for further improvements in pattern density. FPGAs feature low development efforts and costs, but their densities are by far lower than those of ASICs. A good strategy is to use FPGAs for the system development, and to switch to a pin compatible ASIC only for system production when the experiment is getting close to data taking.

IV. THE DATA ORGANIZER

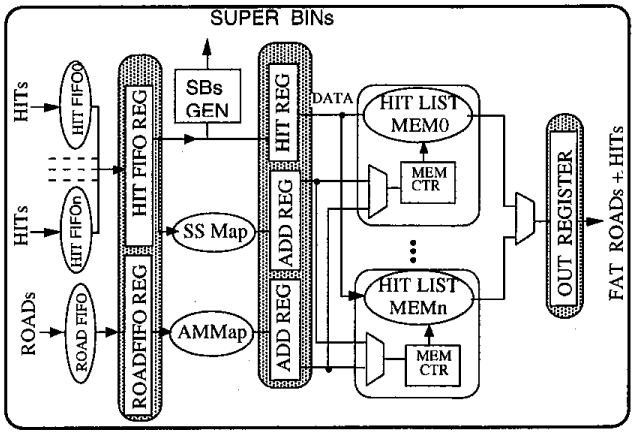

Figure 4: A schematic view of the Data Organizer

Figure 4 shows a possible internal implementation of the Data Organizer. It consists of a pipeline (registers are included into gray boxes) that allows the organization of each event in a special data base, the Hit List Memory (HLM). The Data Organizer has as many Hit List Memories as many AM boards are in the AM pipeline to handle all the events that could be processed at the same time by AM.

It collects hits from more than one input stream and merge them on a single hit input bus. When a hit arrives from some detector layer, it is stored into a memory section dedicated to the super bin the hit belongs to. The memory section address is calculated (SSMAP is a look-up table) directly from the hit itself.

The Data Organizer receives also the roads back from the AM pipeline. When a road arrives the memory address of the super bin belonging to the road is calculated (AMMAP is a

look-up table) from the road itself. The hits belonging to the road super bin can be easily fetched from the Hit List Memory and sent to the output register. If the Data Organizer handles more than one layer, the related super bins are fetched one after the other.

V. The Tree Search Processor

If the fat roads produced by the Associative Memory are too wide to adequately scale down the combinatorial problem, an intermediate step of pattern recognition at higher resolution must be performed by a local tracker before the final track fit. The Tree Search Processor (TSP) is designed to this purpose as it takes fat roads in input and it outputs thin roads.

It is a dedicated parallel and pipelined hardware that implements a binary search strategy on the pattern bank. The bank is organized into a hierarchical structure whose levels correspond to the pipeline stages. Many state machines compare independent bank sections to the event for track identification. A uniform average flux of data along the pipeline is obtained by an optimized distribution of the computing power among structure levels. High density commercial RAMs stores the bank; the machines are easily packed into FPGA devices. A complete description of the processor is available in reference [11].

VI. Data Flow to the Tracker

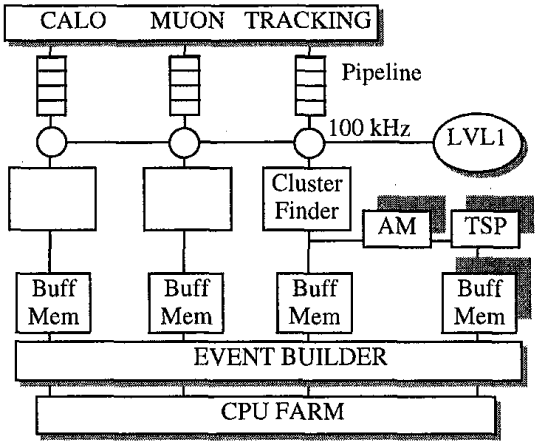

Fig. 5: The AM and TSP in a modern DAQ system.

Figure 5 shows how the Associative Memory and Tree Search Processor could be integrated in modern data acquisition systems. Tracking data are calibrated and searched for clusters at the Level 1 (LVL1) trigger rate, then they are stored into large memory buffers. These buffers are interfaced by an event builder to a large CPU farm for higher level triggers. The AM, possibly followed by the TSP, spies the cluster centroids on their way to the memory buffers and performs strong data reductions. Interesting candidate tracks are stored into an extra memory buffer from where the CPUs can get ordered information at high rate.

The advantage of this kind of implementation is that the new system is almost totally independent. Its interference with the data acquisition is minimal. It can be added even after the baseline as been built, as an upgrade, if the possibility of adding a bypass to spy the events is included in the DAQ since the beginning.

Is it possible to make such a large amount of data available to the AM in due time? CDF data are serialized on a single 15-bits wide data bus, where the hits flow at 30 MHz for a total rate of 0.45 Gbit/sec. Future experiments will increase the data flow rate with a parallel readout of the detector layers. Data could be fed in a new AM chip with a rate of 100 bits every 20 nsec for a total of 5 Gbit/sec, without severe technical problems.

Table 1
One fourth of the CMS silicon barrel

Layer R (cm)	Strips per layer	Occup. (%)	Number of FEDs	nsec per cluster
Pixel 7	2.65×10^6	0.034	15	22 ns
Pixel 11	4.05×10^6	0.017	12	29 ns
Si-strip ϕ 22	53152	2.67	4	14 ns
Si-strip z 22	26624	3.6	2	21 ns
Si-strip ϕ 30	55296	1.6	4	22 ns
Si-strip z 30	36864	2.4	3	22 ns
Si-strip ϕ 40	118272	1.2	8	14 ns
Si-strip ϕ 50	96768	0.9	6	23 ns
Si-strip ϕ 58	111104	0.8	7	22 ns
Si-strip z 58	55552	1.	4	28 ns

As an example of data flow to the AM we report the expected CMS silicon barrel rates [12] on table 1. Numbers are calculated in the hypothesis that 4 AM systems work in parallel, each one receiving data from a quarter of the barrel. Occupancy is defined as the total number of detector channels in reconstructed clusters divided by the total number of channels. The last column is an estimate of the maximum clock period that can be used to transfer cluster centroids to the AM for a 100 kHz event rate. It is assumed that two strips, on average, belong to a cluster and no cluster overlap occurs inside super bins. Even in the very conservative assumption of a 100 kHz level 1 rate, the time for transferring each cluster is reasonable. Should the rate be too high, it is possible to transfer to the AM only a subset of the level 1 triggers. For example, only multi-jet triggers could be analyzed to search for b-jets, interesting for low-mass Higgs physics.

In CMS the whole tracking data are calibrated and searched for clusters by 1000 devices, called FEDs, working in parallel. The column *Number of FEDs* in table 1 reports the expected number of devices necessary to handle each forth of layer in the silicon CMS barrel. The number for the Pixel layers is taken from reference [12] while the other numbers are obtained taking into account that each FED is expected to handle roughly 16 thousands channels.

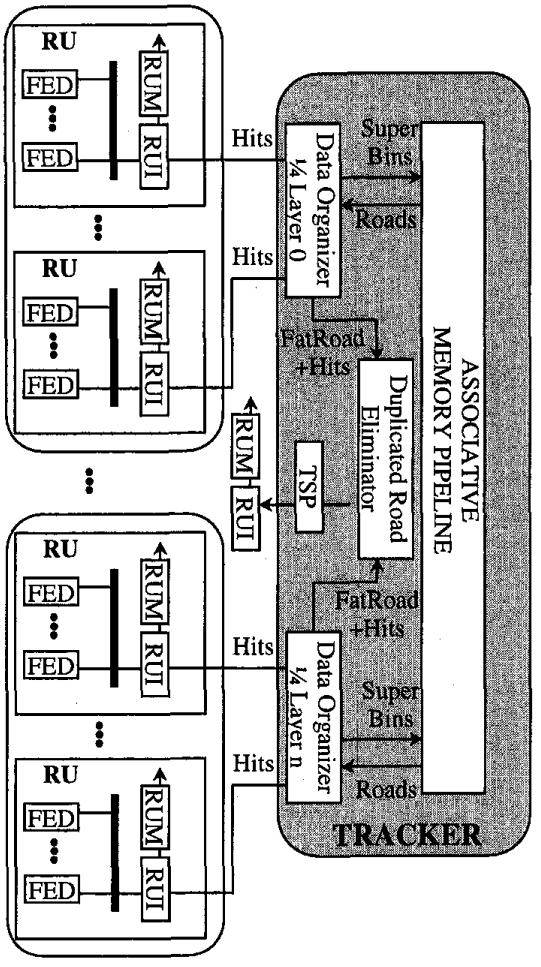

Figure 6: a detailed example of a possible insertion of the online road filter in the CMS DAQ.

Figure 6 shows a detailed scheme of the idea presented in figure 5, applied to the chosen example of CMS. Data from a number of FEDs (less than 6) are collected in a single dual port memory (RUM) trough a Readout Unit Input (RUI). The RUI could be a good place where data can be duplicated and sent to the tracker that in the figure 6 appears as a number of boards grouped in the gray box.

The tracker is provided of a Data Organizer for each forth of layer. Given the number of FEDs shown in table 1, each Data Organizer should receive 2-4 links from as many Readout Units (RU). Layers that are particularly crowded should not be fed into the associative memory since their occupancy is too high when used with reduced resolution. In this case the relative Data Organizer does not send the super bins to the associative memory. However it receives back the road information to provide the super bin hits to the more precise tracking algorithms that work in pipeline with the associative memory.

The fat roads with their full resolution hits are collected from each Data Organizer into the Duplicated Road Eliminator board. There the different layers are finally put together to be later fed into the TSP and finally into a private dual port memory. Roads are checked before the TSP to eliminate not useful duplications. As a matter of fact, the majority logic implemented by the associative memory produces a large number of roads that would differ only for the super bin that is actually empty.

In conclusion the necessary hardware for a forth of barrel can fit into a 9U-VME crate. The tracker takes data from 2-4 RUIs per layer and sends back data to an extra RUI that will buffer in its RUM the filtered roads with their full resolution hits.

To limit the number of links to the associative memory, two independent AMs can search internal and external track segments in the forth of barrel. The higher level CPUs can looks for links between the found segments.

VII. Conclusions

The described processors can find tracks at an event rate of 100 kHz. They are eligible for tracking data reduction in trigger applications. Hits of track candidates, with P_t above a threshold of few GeV and with impact parameters compatible with *b* quark decay, can be filtered among a huge number of other hits. The ambitious goal of trigger selection of *b* decays at the future hadron colliders can benefit from our architecture.

VIII. References

[1] M. Dell'Orso and L. Ristori, "VLSI structures for track finding", *Nucl. Instr. and Meth., vol. A278, 1989* pp. 436-440.

[2] M. Dell'Orso and L. Ristori, "A highly parallel algorithm for track finding", *Nucl. Intsr. and Meth., vol. A287, 1990* pp. 436-440.

[3] H. Grote, "Pattern recognition in high-energy physics", *Rep. Prog. Phys., vol. 50, 1987* pp. 473-500.

[4] S. Belforte et al., "SVT: An Online Silicon Vertex Tracker for the CDF Upgrade", *Nucl. Intsr. and Meth., vol. A409, 1998* pp. 658-661.

[5] S. Belforte et al., "The SVT Hit Buffer", *IEEE Trans. on Nucl. Sci., Vol. 43, 1996* pp. 1810-1813.

[6] P. Battaiotto et al., "The Tree-Search Processor for Real Time Track Pattern Recognition", *Nucl. Intsr. and Meth., vol. A287, 1990* pp. 431-435.

[7] R. Amendolia et al., "The AMchip: a Full-custom MOS VLSI Associative memory for Pattern Recognition", *IEEE Trans. on Nucl. Sci., Vol. 39, 1992* pp. 795-797.

[8] A. Bardi et al. "A large Associative Memory system for the CDF Level 2 Trigger", N16-2 IEEE 1998, Toronto, Canada, 8-14 November 1998.

[9] A. Bardi et al. "A Programmable Associative Memory for Track Finding", *Nucl. Intsr. and Meth., vol. A413/2-3, 1998* pp. 367-373; A. Bardi et al. "A Prototype Programmable Associative Memory for Track Finding" N21-69 IEEE 1998, Toronto, Canada, 8-14 November 1998.

[10] http://www.xilinx.com.

[11] A. Bardi et al. "The Tree Search Processor for Real-Time Track Finding" N21-71 IEEE 1998, Toronto, Canada, 8-14 November 1998.

[12] CMS coll. "The Tracker Project", Technical Design Report, CERN/LHCC 98-6, CMS TDR 5.