High-Energy Physics on DECPeRLe-1 Programmable Active Memory

Laurent Moll*
Jean Vuillemin†
Philippe Boucard‡

Abstract

The future Large Hadron Collider (LHC) to be built at CERN[1], by the turn of the millenium, provides an ample source of challenging real-time computational problems. We report here some results from a collaboration between CERN EAST[2] (RD-11) group and DECPRL PAM[3] team. We present the implementations of the three foremost LHC algorithms on DECPeRLe-1 [2]. Our machine is the only one which presently meets the requirements from CERN (100 kHz event rate), except for another dedicated FPGA-based board built for just one of the algorithm [3]. All other implementations based on single and multiprocessor general purpose computing systems fall short either of computing power, or of I/O resources or both.

1 Introduction

1.1 High-Energy Physics

The community of High-Energy Physics is about to decide to go forward with the next generation collider to be built at CERN, the LHC. With this new instrument, it will be possible to observe proton-proton collisions of 8000 GeV, an energy not attainable today. Experimentation in that ring is expected to start at the beginning of the century, around year 2002. Two different detector sets are being designed: CMS and ATLAS. They are huge structures implementing a number of very different specific detectors such as Silicon Tracker, Muon Chambers, Transition Radiation Tracker and Calorimeters. A picture of ATLAS is shown on figure 1.

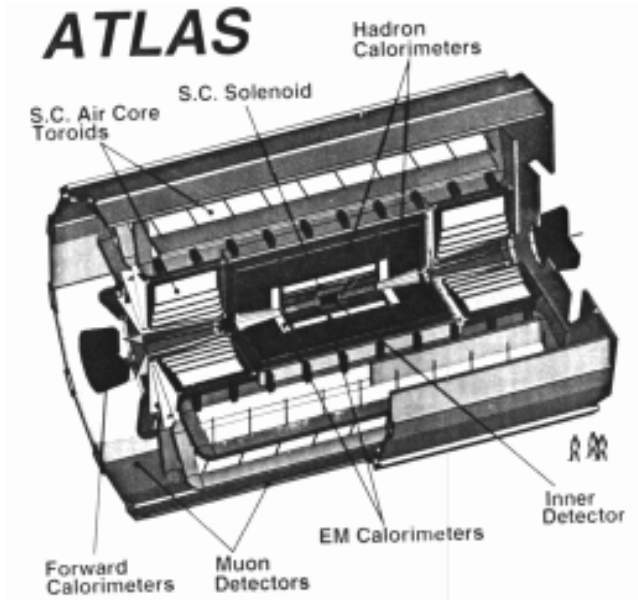

Figure 1: The ATLAS experiment.

The data rate effectively generated by all different detectors will reach very high levels, for two reasons:

- The bunch crossing frequency is about 40 MHz, or 8 meters in terms of of distance.
- There will be over 10 million channels.

The digital data flow provided by the detectors will be over 100 GB/s. As it is obviously not possible to store or use directly this huge quantity of information, a multi-layer scheme has been proposed to reduce the data flow to amounts treatable by high-level processors and storable continuously. It is organized in three trigger levels:

- The *first-level trigger* includes an analog and a digital part. Its purpose is to select quickly (at 40 MHz)

*Ecole Nationale Supérieure des Télécommunications, Paris, France.
†Pôle Universitaire Léonard-de-Vinci, La Défense, France
‡Matra-Harris Semiconductors, Saint-Quentin-en-Yvelines, France.
This work was done while the authors were employees of Digital Equipment Corporation, Paris Research Laboratory, Rueil-Malmaison, France.

[1] European Organization for Nuclear Research, Geneva, Switzerland.
[2] Embedded Architectures for Second-level Triggering [1].
[3] Programmable Active Memories.

which Regions Of Interest (ROIs) in the whole picture seen by the detectors are to be kept for further analysis. A huge data switch sends only the selected regions to the next level.

- The *second-level trigger*, where DECPeRLe-1 fits, processes each of the selected ROIs, over the full data from the detectors. It extracts in real-time useful physics features, such as tracks and energy level distribution in order to achieve discrimination based on physics criteria.

- The *third-level trigger* is composed of high-level processors performing experiment-specific processing of the filtered data to provide further filtering before storage (actually, this three-level architecture is still evolving, but it has been chosen as a starting point for the ATLAS experiment, see [4]).

The foreseen data rates at the different levels are shown on figure 2.

level	event frequency	data rate
1	40MHz	100TB/s
2	100kHz	100GB/s
3	1000Hz	1GB/s

Figure 2: Data rates inside the LHC.

The most interesting problems for FPGA-based machines are situated in the second-level triggering part, as the data flow is high, the computation needs are well suited for parallel hardware implementation, and the flexibility is very important. In the next section are presented the implementations of the second-level triggers for the three best-defined detectors in the ATLAS experiment.

1.2 The DECPeRLe-1 board

These implementations have been done on DECPeRLe-1, which is a general-purpose FPGA-based board. The computational core is a 4×4 matrix of XC3090-100 chips. Each FPGA has 16 direct connections to each of its four nearest neighbors and 16 other connections to each of the four horizontal and vertical buses. Each side of the matrix can be used for external links with 4 32-bit connections. Four more FPGAs are used as switches between the matrix and the four 1MB SRAMs, and two other ones to control these RAMs. The last chip is used to manage the link with the host, through a FIFO. More information on this system can be found in [2].

The board is linked to the host by a small TURBOchannel board based on another Xilinx FPGA. This link allows configuration and readback of the chips, clock control (speed, starting, stopping, stepping...) and data transfer (by I/O, DMA or asynchronous lines).

Designs are made with special CAD tools, using a C++ library to describe the synchronous circuit to be implemented. This approach allows at the same time a great flexibility due to the high-level language used, and a complete control over the specific Xilinx statements (placement and routing hints). The Perle1DC library is fully described in [5].

2 The algorithms

All the algorithms presented here have the same global real-time constraint: process the events at 100 kHz. Two types of algorithms are used:

- The dense map processing computes the full image digitized by the detector from a scan line input.

- The sparse map or list processing takes for input a list of the non-zero pixels in each image. Pixels are processed one at a time, generally through a look-up table (LUT).

The optimal choice between these two types of processing depends on the density of the images provided by the detectors, the computational complexity of the two algorithms, and the bandwidth of the RAMs.

The three algorithms implemented show three very different interests of the PAMs:

- The high input bandwidth of the board.

- The massive parallelisation of the tasks, with no temporal cost for the control logic.

- The ease of processing of numbers with only afew bits, and the small surface required.

For each algorithm, a comment is made on the differences with the software implementation, and on the advantage of using FPGAs. The surface used on the DECPeRLe-1 board is indicated, and the frequency of the designs is around 25 MHz for the three of them. This limit is given by the speed of the chips used (100 MHz of toggle rate), and the maximum operating frequency of the RAMs (25 MHz).

2.1 The Calorimeter (CALO)

The Calorimeter detector is composed of many small towers. There are both an electromagnetic layer and a hadronic layer. The algorithm implemented here is based on ROIs of 20×20 towers (pixels). Each pixel value provides the energy level for the two layers, presented by two 16-bit integers. The resulting data input rate is high: 160 MB/s.

The analysis of an event is done by computing:

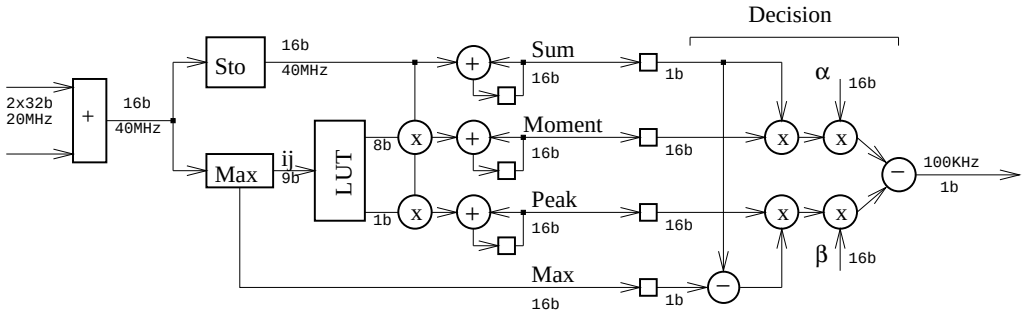

Figure 3: The Calorimeter datapath.

1. The pixel-wise sum.
2. The total energy.
3. The maximum energy.
4. The first statistical moment.
5. The peak energy.
6. The final discriminant uses the previously calculated characteristics of an event to distinguish electrons from pions or from hadronic jets, or even pions from jets.

The complete datapath is shown on figure 3. The features extracted for further processing are computed in real-time.

The CALO implementation on DECPeRLe-1 uses a datapath completely equivalent to the software implementation. It is efficient because it can handle the 160 MB/s of input data, and because all of the operators are carefully designed (e.g. using Booth coding for the multipliers) and sized (in terms of number of bits in input and output) so that the overall surface needed is about 4 LCAs, or 1/4 of the computational matrix of the PAM. This design is mainly composed of arithmetic operators with some control.

The CALO algorithm and its cost in computing resources is analyzed in [6].

2.2 The Transition Radiation Tracker (TRT)

The TRT is a detector based on 76800 radial straw tubes organized in wheels perpendicular to the z-axis. A ROI is composed of 32 wheels $\times$ 16 straws. There are 2 thresholds per straw at 2 different energy levels. So, for each event, we have to treat $2 \times 32 \times 16$ bitmaps. On these bitmaps, the particles have almost straight trajectories.

We have to extract the track on which there is the largest number of points. To achieve this goal, we compute a Hough Transform on the complete bitmaps: it consists on calculating the histogram of the number of points on each line defined by its intercept and its slope. It is done line by line, so that we can get the resulting histogram also line by line. We can then compute the maximum by comparing each line to the maximum, which allows us not to store the complete histogram.

To cope with the 100 kHz constraint and to increase the number of different tracks recognizable, J.Vuillemin has introduced a variation of the Hough Transform called Fast Hough Transform (FHT), which is based on a *divide and conquer* algorithm and reduces the computational complexity from $O(n^3)$ to $O(n^2.\log(n))$. The FHT is fully described in [7]. Thanks to it, we can provide a choice of 128 different intercept $\times$ 31 different slopes = 3968 tracks.

The datapath is heterogeneous, as shown on figure 4, and is very different from all the other implementations, software or hardware. It has thus been the longest to specify and to code (about 2 man$\times$months).

The processing of the Hough Transform by extensive arithmetic computation as opposed to table lookups, and the use of a special *divide and conquer* algorithm make this implementation far more efficient than the other ones. The computational matrix of DECPeRLe-1 is almost full, but no RAM is used for the algorithm. The design is mainly composed of 1-bit adders, comparators and delays put together in such a way that it perfectly fits a FPGA implementation (full use of elementary blocks, logic and flip-flops, and of routing resources).

The high-level C++ library used as CAD tool was especially useful for this application, because the logic, the control, the placement, the routing of the design and the driving software all depend on the geometry of the detector, which changed many times during the development. As we were using the same C file to describe the detector, a simple modification in this file and a recompilation changed in a matter of ten minutes the design and the software to make them compatible with the new caracteristics.

The specification of this implementation of TRT can be found in [8] and the final tests and beam test results

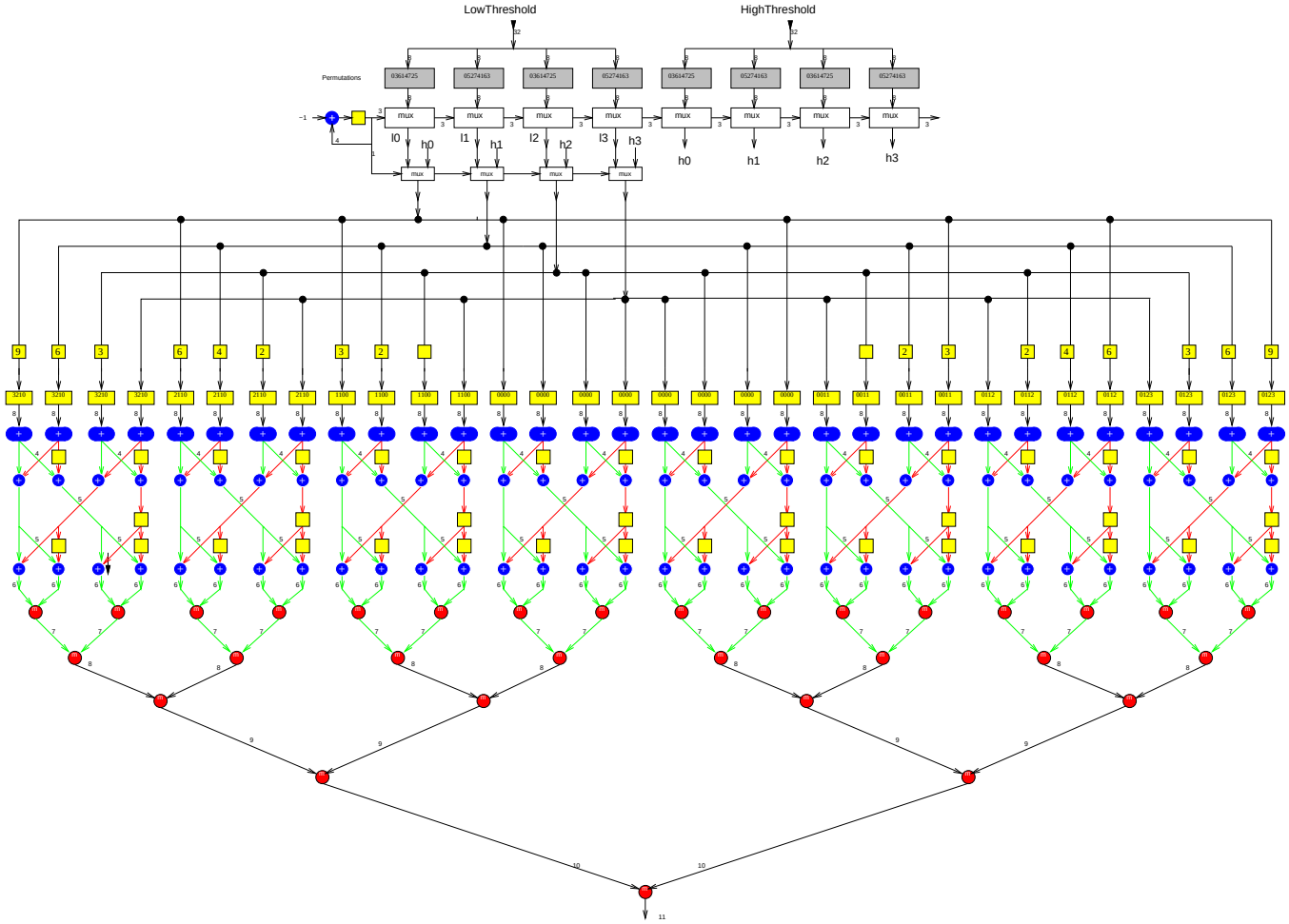

Figure 4: The TRT datapath.

in [9].

2.3 The Silicon Tracker (SCT)

The SCT detector is composed of six cylindrical layers covered with 100 μm sensors. Four of these layers are useful in this algorithm, and so a particle can be followed through 4 impacts. A ROI is composed of 4 layers $\times$ 1040 sensors.

As the hit precision is very high, the density of hits is very low, even with the additional noise. Simulation gives an average occupancy of the sensors at 2.5 %. As a consequence, a sparse map algorithn using LUTs has been chosen: for each point on the image, a LUT points to the tracks to which the point belongs. We histogram the complete Hough-Transform of the ROI. For each possible track in the histogram, a 4-bit count is maintained, one bit per layer. At the end of the list of points, the histogram is emptied, and for each track, we compute the number of points on the track. The best track is extracted by the largest number of points (generally, there is one 4-point track).

A total of $64 \times 32 = 2048$ tracks can be extracted at 100 kHz with an average occupancy of 3 %, in the DECPeRLe-1 implementation.

The datapath shown on figure 5 is very homogeneous, and the DECPeRLe-1 design mainly composed of a regular matrix of histogram and max-extracting cells, plus some control. This implementation of the SCT algorithm is fully optimized for both the available material and the LUT algorithm. The computational matrix is completely used: all the CLBs of the Xilinx parts are fully used (5 inputs, 2 logical functions, 2 flip-flops), all the longlines are used, and the routing is still very hard. All the external connections of the FPGAs are used, and the switches and controllers are almost full.

The bandwidth needed from the RAMs is so high (2048 tracks at 25 MHz = 51.2 Gb/s) that we must compress the contents of the LUT to fit into only 3 32-bit RAMs. This application fits perfectly in a FPGA-based machine, and the obtained performance is very good. This implementation uses the same algorithm as the software implementation, but the high level of con-

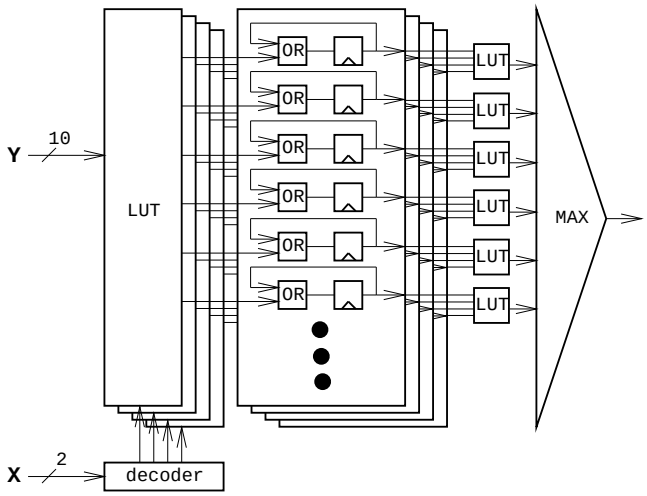

Figure 5: The Silicon Tracker datapath.

currency, and the huge LUT banwidth handled by the histogramming part make this implementation far more powerful than any high-level processor.

The complete description of the algorithm and the implementation is given in [10].

3 The real-time environment

The TRT detector and the router logic was sufficiently advanced in June 1994 to allow a real beam test with a prototype with a few straws, a router, some different second-level feature extractors and the data acquisition.

To interface with the detector, we receive data from two HIPPI lines (monodirectional 32-bit at 25MHz on differential pairs), and send final results through ethernet to a VME crate running an OS/9 system handling all the real-time environment and providing the HIPPI connections.

The aim was to provide quickly an efficient TRT feature extraction machine based on DECPeRLe-1. We use the *HIPPI to TURBOchannel Interface board* (HTI) developed at CERN and commercialized by HYTEC [11], which is the easiest way to interface with HIPPI without creating a specialized board.

Two of these boards are plugged directly onto DECPeRLe-1 internal connectors, and two full TURBOchannel interfaces is implemented in the FPGAs, to control the HTI boards from the host through DECPeRLe-1, and to get data by DMA.

Working in the real-time environment forced us to add a 256 kB 25 MHz FIFO implemented with two of DECPeRLe-1's RAMs. As the results are sent to the Unix workstation, which may swap the reading process for 20 ms, we must be able to store data for the same amount of time.

The development of the TURBOchannel interface in DECPeRLe-1 took 3 weeks to an expert in both TURBOchannel and FPGAs. It uses 2 RAMs to double-bufferize the data from the HTI boards and some part of the switches and controllers. The settlement of the board with the correct algorithm (the TRT for this beam test) took some more weeks because of the unclear specifications of the environment provided. We can therefore evaluate the amount of work to go from the specs to a working feature extraction device to 3 man$\times$months, which is really short compared to all the other known devices having provided the same functionalities.

4 The performance

The CERN/EAST group is testing a number of other architecture for second-level triggering, but results are really available only for software implementations [12], as a big research direction is the use of high-level processor farms. Moreover, it is not really easy to compare the performances as the specification for the detectors and the input data are quickly evolving.

The following tables show the performances obtained by DECPeRLe-1 compared to SPARC10 implementations. All timings are in μs. To meet the real-time requirements, they all should be inferior to 10 (= 100 kHz), which is the case for the three DECPeRLe-1 implementations.

- **Calorimeter**: the software implementation measured here provides about 4 times as much precision as the PAM, but the design uses approximately a quarter of the whole FPGAs resources. This difference is due to incompatible specifications.

SPARC10	DECPeRLe-1
1290	10

- **Transition Radiation Tracker**: the software implementation uses a completely different algorithm with thresholded data, and is four times less precise than DECPeRLe-1 implementation.

SPARC10	DECPeRLe-1
865	10

- **Silicon Tracker**: both implementations use exactly the same algorithm with the same data. This comparison is thus the best we can do.

SPARC10	DECPeRLe-1
4990	8

5 The efficiency of FPGAs for High-Energy Physics applications

The difference of performance between the DECPeRLe-1 and the SPARC10 implementation show a ratio of 100 to 500 in favor of the former. The price of the board is equivalent to the price of a well-equiped workstation, and it is not much longer to design a whole circuit than to do the software, including the real-time and I/O handling.

The FPGA-based solutions prove to be the perfect solution for getting high performance at low cost without being obliged to develop an ASIC. The needs of computation power are such that even with that level of performance, a lot of devices will be needed (typically some hundreds). That is why the maximum of computation power should be put in each device. It allows the use of general-purpose FPGA-based machines, like DECPeRLe-1, used at full power.

6 Conclusion

Thanks to the ease of use of the development tools for DECPeRLe-1, all these designs have been done in a very short time: from one week for the SCT to two months for the TRT including the TURBOchannel interface and the real-time tests. The flexibility of the designs are such that we were able to follow the changes of specification of the detector (which happen quite often at this stage of development) in only one day. If we add the performances, which are already good enough to use the PAM technology in the 2002 detector, we find that FPGA-based machines are probably the best trade-off in terms of cost / performance / development time.

A team from Mannheim university built a dedicated FPGA-based board for the same tests [3], and a cooperation between this team and DEC-PRL PAM team is developing a set of hardware and tools to match the exact CERN needs. There is no doubt now that PAM-like technology will be used for the second-level triggering for LHC.

7 Acknowledgements

Without the following people, none of this would have been done. We wish to thank Rudy K. Bock, Lars Lundheim and Werner Krischer, members of the EAST project at CERN.

References

[1] J. Badier, R. Bock et al. *Evaluating Parallel Architectures for two Real-Time Applications with 100kHz Repetition Rate.* In IEEE Transactions on Nuclear Science, 40:1:45-55, 1993.

[2] J. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati, P. Boucard. *Programmable Active Memories: Reconfigurable systems Come of Age.* To be published in the IEEE Transactions on VLSI.

[3] F. Klefenz, K.H. Noffz, R. Zoz and R. Maenner. *ENABLE—A Systolic 2nd Level Trigger Processor for Track Finding and e/pi Discrimination for ATLAS/LHC.* Proc. IEEE Nucl. Sci. Symp., San Francisco, CA, USA, 62–64, 1993.

[4] C.P. Bee, G. Mornacchi and G. Polesello. *The ATLAS Test Beam Data Acquisition. A proposal for a Flexible and Adaptable System.* CERN/PPE division note, March 1994.

[5] H. Touati. *Perle1DC: a C++ Library for the Simulation and Generation of DECPeRLe-1 Designs.* DEC-PRL Technical Note #4, February 1994.

[6] J.E. Vuillemin. *On computing power.* In *Programming Languages and System Architectures*, J. Gutknecht ed., p 69-86, LNCS 782, Springer-Verlag 1994.

[7] J.E. Vuillemin. *Fast Linear Hough Transform.* Proceedings of the 1994 International Conference on Application-Specific Array Processors.

[8] J.E. Vuillemin. *Specifications for the Transition Radiation Tracker on DECPeRLe-1.* CERN/EAST note 94-12, April 1994.

[9] L. Lundheim, L. Moll, P. Boucard and J. Vuillemin. *TRT on DECPeRLe-1: Algorithm, Implementation, Test and Future.* CERN/EAST note 94-20, August 1994.

[10] W. Krischer and L. Moll. *Implementation of a pattern recognition algorithm for the Silicon Tracker on DECPeRLe-1.* CERN/EAST note 94-21, September 1994.

[11] T. Anguelov. *HIPPI to TURBOchannel Interface.* CERN/EAST note 93-07, March 1993.

[12] R. Hauser and I.C. Legrand. *Second Level Trigger Feature Extraction Algorithms.* CERN/EAST note 93-13, May 1993.