Journal of VLSI Signal Processing, [Volumn Number], 1–14 ([Volumn Year])
© [Volumn Year] Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Real-time High-Energy Physics Applications on DECPeRLe-1 Programmable Active Memory

LAURENT MOLL*
JEAN VUILLEMIN*
Pôle Universitaire Léonard de Vinci, 92916 Paris La Défense Cedex, France

PHILIPPE BOUCARD*
Matra-Harris Semiconductors, Saint-Quentin-en-Yvelines, France

LARS LUNDHEIM
CERN, Geneva, Switzerland

Received December 29, 1994. Revised April 8, 1995.

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 DEC-PRL PAM[3] team. We present implementations of the four foremost LHC algorithms on DECPeRLe-1 [1]. Our machine is the only one which presently meets the requirements from CERN (100 kHz event rate), except for another dedicated FPGA-based machine built for just one of the algorithms [2]. 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.

Keywords: High-energy physics, real-time, programmable active memories, field-programmable gate arrays.

1. Introduction

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 the collisions of bunches (small packets) of protons turning in two rings in opposite directions at 14 TeV (7 TeV in each direction), an energy not attainable today. Experimentation in that ring is expected to start at the beginning of the century, around year 2002. Two different detectors are being designed: CMS and ATLAS. They are huge structures implementing a number of very different specific sub-detectors such as Silicon Trackers, Muon Chambers, Transition Radiation Trackers and Calorimeters. A picture of ATLAS is shown on figure 1.

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 distance at almost light-speed.
- There will be over 10 million channels.

The digital data flow provided by the detectors will be over 100 TB/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

* This work was done while Laurent Moll, Jean Vuillemin and Philippe Boucard were employees of Digital Equipment Corporation, Paris Research Laboratory, Rueil-Malmaison, France

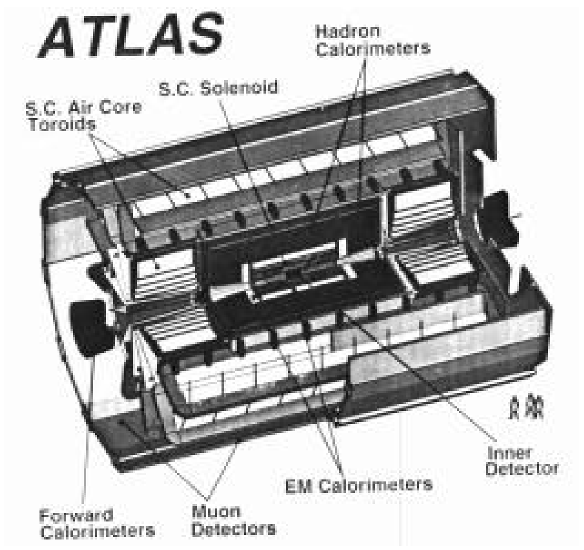

Fig. 1. The ATLAS experiment.

40 MHz) 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 [3]).

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

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 flexibility is very important. In the next section we present the concept of PAM, and the DECPeRLe-1 board, hardware base of the implementations. In the last section we present the implementations themselves: the algorithms for the three best-defined detectors in the ATLAS experiment and the global decision for the second-level trigger.

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

Fig. 2. Data rates inside the LHC.

2. Programmable Active Memories

The purpose of a Programmable Active Memory (PAM) is to implement a *virtual machine* which can be dynamically configured as a large number of specific hardware devices.

The structure of a generic PAM is found in figure 3. It is connected—through the *in* and *out* links—to a host processor. A function of the host is to *download* configuration bitstreams into the PAM. After configuration, the PAM behaves, electrically and logically, like the ASIC defined by the specific bitstream. It may operate in stand-alone mode, hooked to some external system—through the *in′* and *out′* links. It may operate as a co-processor under host control, specialized to speed-up some crucial computation. It may operate as both, and connect the host to some external system, like an audio or video device, or some other PAM.

The High-Energy Physics applications presented here are implemented on a specific PAM: it is named DECPeRLe-1 and it was built at Digital's Paris Research Laboratory in 1992. A dozen copies operate at various scientific centers in the world.

Besides our PAMs, which were built first at INRIA in 1987 [4], then at DEC-PRL, other successful implementations of reconfigurable systems have been reported, in particular at the universi-

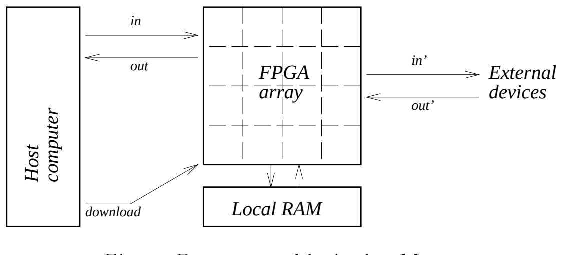

Fig. 3. Programmable Active Memory

ties of Edinburgh [5] and Zurich [6], and at the Supercomputer Research Center in Maryland [7].

The ENABLE machine is a system, built from FPGAs and SRAM, specifically constructed at the university of Mannheim [8] for solving the TRT problem of section 3.2. A second-generation board called ENABLE++ is already being built [25]. Many similar application-specific reconfigurable machines have been built in the recent years: the reconfigurable nature is only exploited while developping and debugging the application. Once done, final configuration is done, once and for all—until the next "hardware release".

Commercial products already exist: QuickTurn [9] and Meta-Systems[4] sell large configurable systems, dedicated to hardware emulation. Compugen [10] sells a modular PAM-like hardware, together with several configurations focusing on genetic matching algorithms. More systems exist than just the ones mentioned here.

A thorough presentation of the issues involved in PAM design, with alternative implementation choices, is given by Bertin [11].

2.1. The DECPeRLe-1 board

The overall structure of DECPeRLe-1 is shown in figure 4. Each of the 23 squares denotes one Xilinx XC3090 FPGA [12]. Each of the 4 rectangles represents 1 MB of static RAM (letter R). Each line represents 32 wires, physically laid out on the printed circuit board (PCB) of DECPeRLe-1. A photo of the system is shown in figure 4.

Depending upon the application, individual units are implemented within one to many FPGAs; they may also be implemented as *look-up tables* (LUT) through the local RAM; some slow processes are implemented by software running on the host. Connections between processing units are mapped, as part of the design configuration, either on PCB wires or on internal FPGA wires.

2.2. FPGA matrix

The computational core of DECPeRLe-1 is a 4×4 matrix of XC3090—letter M in figure 4. Each FPGA has 16 *direct connections* to each of its four Manhattan neighbors. The four FPGAs in each row and each column share two common 16-bit buses. There are thus four 64-bit buses traversing the array, one per geographical direction N, S, E, W.

The purpose of this organization is to best extrapolate, at the PCB level, the internal structure of the FPGA. What we have is close to a large FPGA with 64×80 Programmable Active Bits—except for a connection bottleneck every quarter of the array, as there are fewer wires on the PCB than inside the FPGA. By Noyce's thesis, DECPeRLe-1 implements, with 1992 technology,

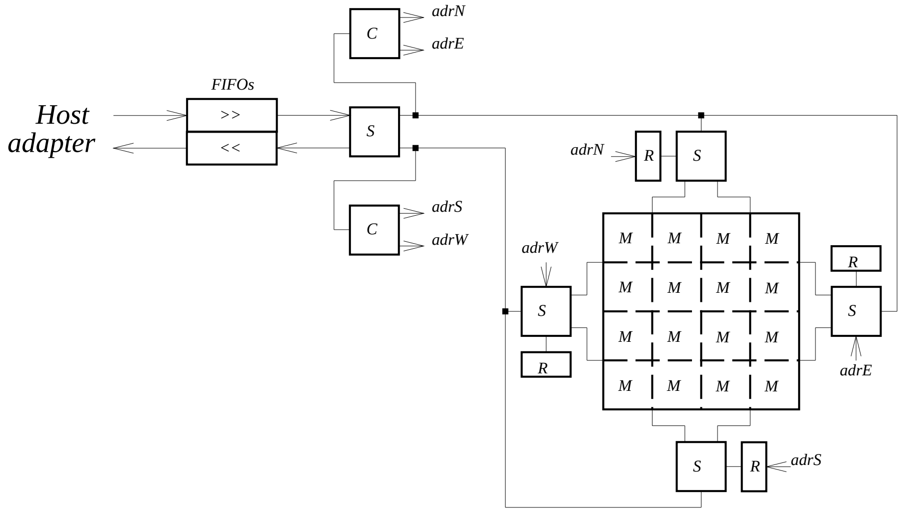

Fig. 4. DECPeRLe-1 architecture

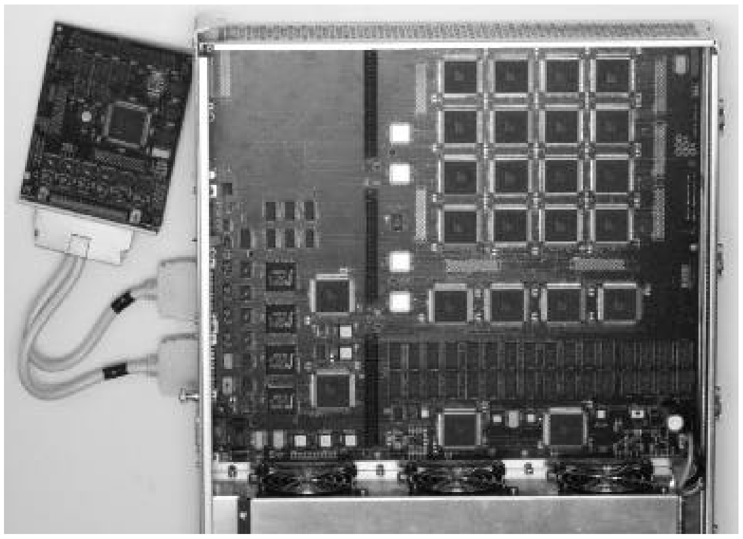

Fig. 4. DECPeRLe-1 and its TURBOchannel interface board

a leading edge FPGA which should become available on a single chip by 1998.

2.3. Local RAM

Some applications are entirely implemented with FPGA logic; most others require some amount of RAM: to buffer and re-order local data, or to implement specialized LUTs.

The size of this cache RAM is 4 MB for DECPeRLe-1, made of four independent 32-bit-wide banks. The 18-bit addresses and read/write signals for each RAM are generated within one of two *controller* FPGAs—letter C in figure 4. Data to and from each RAM goes to the corresponding *switch* FPGA—letter S.

2.4. External links

DECPeRLe-1 has four 32-bit-wide external connectors.

Three of these (not represented on figure 4) link edges of the FPGA matrix to external connectors. They are used for establishing *real-time* links, at up to 33 MHz, between DECPeRLe-1 and external devices: audio, video, physical detectors... Their aggregated peak bandwidth exceeds 400 MB/s.

The fourth external connection links to the host interface of DECPeRLe-1: a 100 MB/s *TURBOchannel* adapter [13]. In order to avoid having to synchronize the host and PAM clocks, host data goes through two FIFOs, for input and output respectively. To the PAM side of the FIFOs is another switch FPGA, which shares two 32-bit buses with the other switches and controllers—see figure 4.

The host connection itself consists of a host-independent part implemented on the mother board and a host-dependent part implemented on a small option board specific to the host bus. A short cable links the two parts—see figure 4.

In addition to the above, DECPeRLe-1 features daughter-board connectors which can provide more than 1.2 GB/s of bandwidth to specialized hardware extensions.

2.5. Firmware

One extra FPGA on DECPeRLe-1 is not configurable by the user; call it POM, by analogy with ROM. Its function is to provide control over the state of the PAM, through software from the host.

The logical protocol of the host bus itself is *programmed* in POM configuration. Adapting from TURBOchannel to some other logical bus format, such as VME, HIPPI or PCI is just a matter of re-programming the POM and re-designing the small host-dependent interface board.

A function of the POM is to assist the host in downloading a PAM configuration—1.5 Mb for DECPeRLe-1. Thanks to this hardware assist, we are able to reconfigure DECPeRLe-1 up to fifty times per second, a crucial feature in some applications. One can regard DECPeRLe-1 as a *software silicon foundry*, with a 20 ms turn-around time.

We take advantage of an extra feature of the XC3090 component: it is possible to dynamically *read back* the contents of the internal state register of each PAB. Together with a *clock stepping* facility—stop the main clock and trigger clock cycles one at a time from the host—this provides a powerful debugging tool, where one takes a snapshot of the complete internal state of the system after each clock cycle. This feature drastically reduces the need for software simulation of our designs.

PAM designs are synchronous circuits: all registers are updated on each cycle of the same global clock. The maximum speed of a design is directly determined by its *critical combinational path*. This varies from one PAM design to another. It has thus been necessary to design a clock distribution system whose speed can be *programmed* as part of the design configuration. On DECPeRLe-1, the clock can be finely tuned, with increments on the order of 0.01%, for frequencies up to 100 MHz.

A typical DECPeRLe-1 design receives a logically uninterrupted flow of data, through the input FIFO. It performs some processing, and delivers its results, in the same manner, through the output FIFO. The host is responsible for filling-in and emptying-out the other side of both FIFOs. Our firmware supports a mode in which the application clock automatically *stops* when DECPeRLe-1 attempts to read an empty FIFO or write a full one, effectively providing fully automatic and transparent *flow-control*.

The full firmware functionality may be controlled through host software. Most of it is also available to the hardware design: all relevant wires are brought to the two controller FPGAs of DECPeRLe-1. This allows a design to synchronize itself, in the same manner, with some of the external links. Another unique possibility is the *dynamic tuning of the clock*. This feature is used in designs where a slow and infrequent operation—say changing the value of some global controls every 256 cycles—coexists with fast and frequent operations. The strategy is then to slow the clock down before the infrequent operation—every 256 cycles—and speed it up afterwards—for 255 cycles. Tricky, but doable.

2.6. PAM programming

A PAM program consists of three parts:

1. The *driving software*, which runs on the host and controls the PAM hardware.
2. The logic equations describing the synchronous hardware implemented on the PAM board.
3. The placement and routing directives that guide the implementation of the logic equations onto the PAM board.

The driving software is written in C or C++ and is linked to a runtime library encapsulating a device driver. The logic equations and the placement and routing directives are generated algorithmically by a C++ program. As a deliberate choice of methodology, all PAM design circuits are *digital* and *synchronous*. Asynchronous features—such as RAM write pulses, FIFO flags decoding or clock tuning—are pushed into the firmware (POM) where they get implemented once and for all.

A full DECPeRLe-1 design is a large piece of hardware: excluding the RAM, twenty-three XC3090 containing 15k PABs are roughly the equivalent of 200k gates. This amount of logic would barely fit in the largest gate arrays available in 1994.

The goal of a DECPeRLe-1 designer is to encode, through a 1.5 Mb bitstream, the logic equations, the placement and the routing of fifteen thousand PABs in order to meet the performance requirements of a compute-intensive task. To achieve this goal with a reasonable degree of efficiency, a designer needs full control over the final logic implementation and layout. In 1992, no existing *computer-aided design* (CAD) tool was adapted to such needs.

We decided to use C++ for reasons of economy and simplicity. VHDL is a complex, expensive language. C++ programming environments are considerably cheaper, and we are tapping a much wider market in terms of training, documentation and programming tools. Though we had to develop a generic software library to handle netlist generation and simulation, the amount of work remains limited. Moreover we keep full control over the generated netlist, and we can include circuit geometry information as desired. The Perle1DC library is fully described in [14].

2.7. The runtime library

At the system level, the programming environment provides two main functions: a device driver interface, and full simulation support of that interface. This simulation capability allows the designer to operate together the hardware and software parts from a PAM program. The device driver interface provides the mandatory controls to the application program: the usual UNIX I/O interface with open, close, synchronous and asynchronous read and write; download of the configuration bitstreams for the PAM FPGAs; readback of their state (i.e. the values of all PAB registers); read and write of the PAM static RAMs; software control of the PAM board clock.

3. The algorithms

All the algorithms presented here have the same global real-time constraint: process the events at 100 kHz. For the detectors, a Region Of Interest has to be treated, corresponding to a small part of the whole detector. The representation of the detector data in this ROI is often a bitmap obtained directly or by projection on two of the axes in cylindrical coordinates.

To process this bitmap, 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 four algorithms implemented show three very different points of interest of PAMs:

- The high input bandwidth of the board.
- The massive parallelization of the tasks, with no temporal cost for the control logic.
- The ease of processing of numbers with only a few 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; the frequency of the designs is around 25 MHz for the four of them. This limit is given by the speed of the chips used (100 MHz toggle rate), and the maximum operating frequency of the RAMs (25 MHz).

3.1. The Calorimeter (CALO)

The Calorimeter detector is composed of many small towers. They are composed of both an electromagnetic layer and a hadronic layer. The algorithm implemented here is based on ROIs of 20 ×

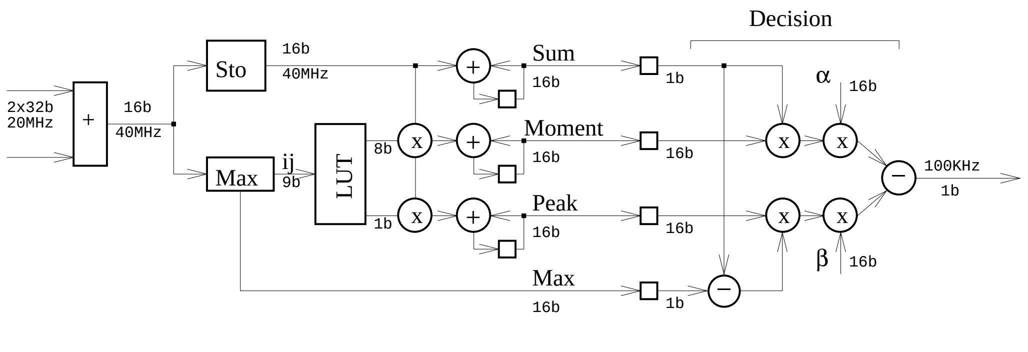

Fig. 5. The Calorimeter datapath.

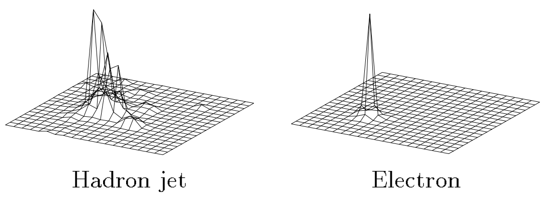

Fig. 5. Calorimeter typical input images

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:

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 5. 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 input data, and because all 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 [15].

3.2. The Transition Radiation Tracker (TRT)

The TRT is a detector based on 76800 radial straw tubes organized in wheels (discs) 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 two $\times 32 \times 16$ bitmaps corresponding to the 2D projection of the cylinder of straws. On figure 6, we can see the arrangement of the straws. The horizontal axis is the z-axis and the vertical axis is the θ-axis. On these bitmaps, the particles have almost straight tracks corresponding to the helicoidal 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 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 [16]. 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 8, 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).

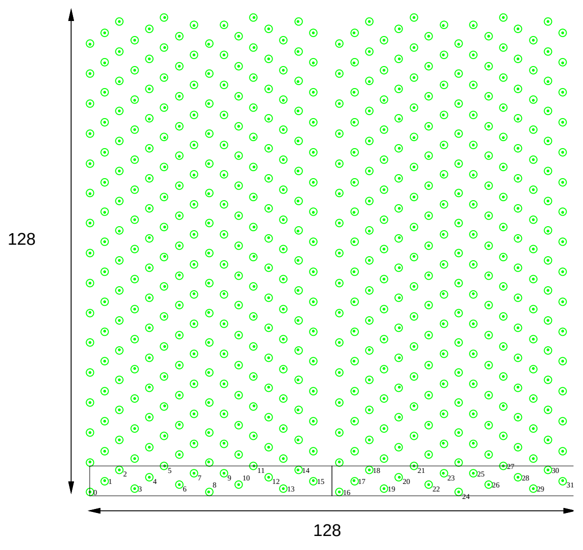

Fig. 6. The TRT geometry.

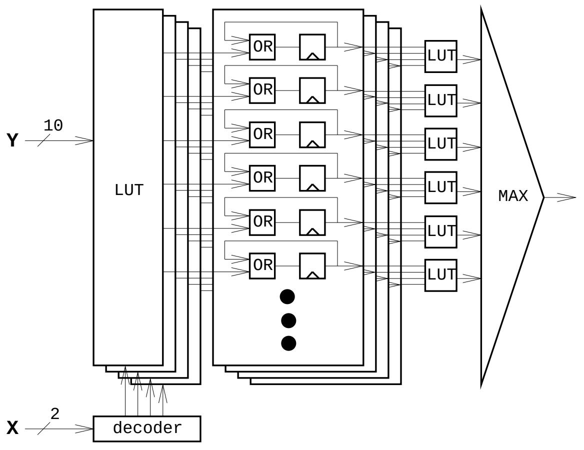

Fig. 7. The Silicon Tracker datapath.

The processing of the Hough Transform by extensive arithmetic computation as opposed to table look-ups, 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 characteristics.

The specification of this implementation of TRT can be found in [17] and the final tests and beam test results in [18].

3.3. 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 ; a particle can thus 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 algorithm 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 7 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 three 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 concurrency, and the huge LUT bandwidth handled by the histogramming part make this implementation far more powerful than any high-level processor.

A complete description of the algorithm and the implementation is given in [19].

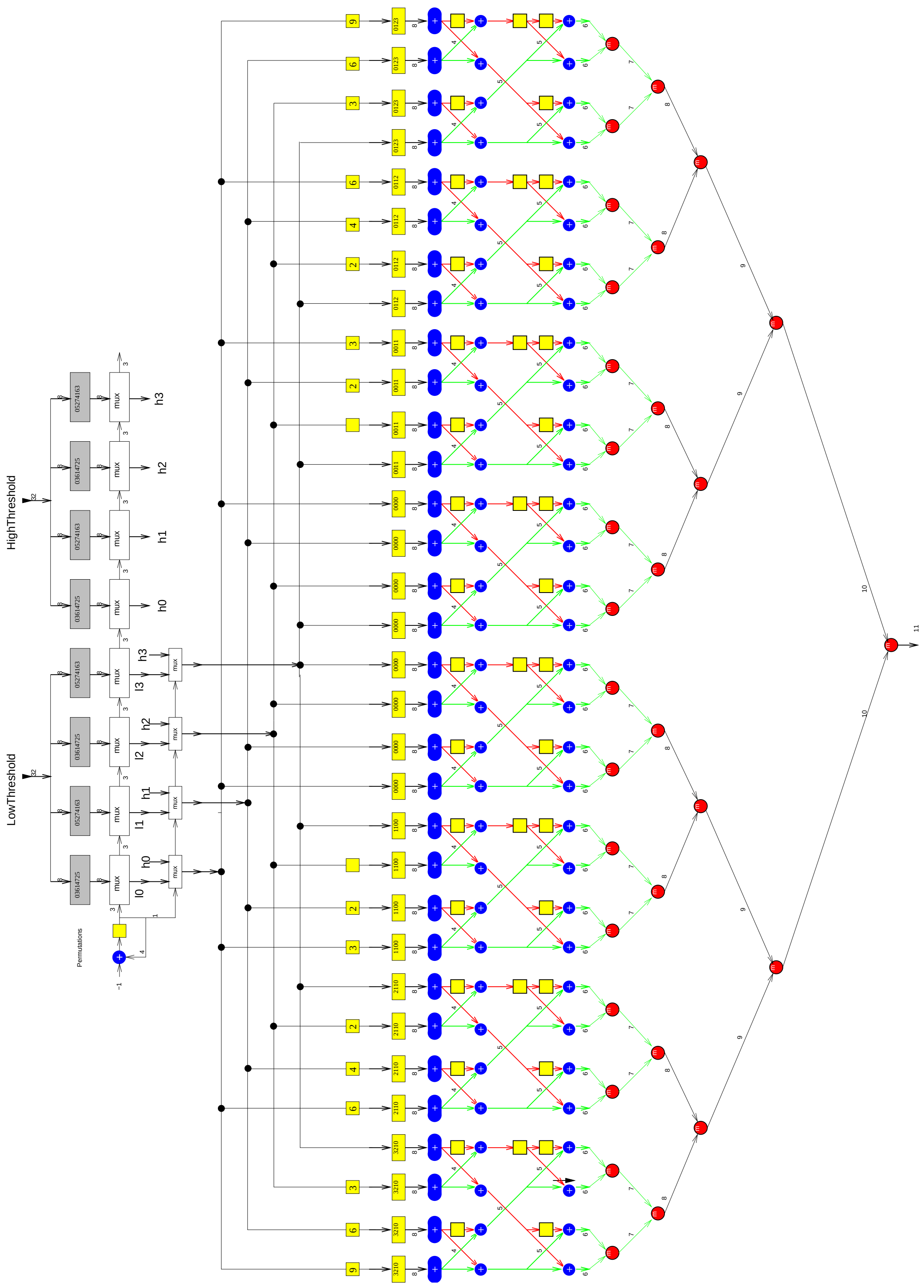

Fig. 8. The TRT datapath.

3.4. Global decision

The algorithms described above perform all so-called *feature extraction tasks.* This is only the first step in second level triggering. When features are extracted from all sub-detectors, one must estimate what kind of particles are involved and whether the physics reaction behind the observed data is an interesting one. This is called the *global decision task* of second level triggering.

Much of the global decision task consists of making simple comparisons between features and thresholds. However, two compute-intensive tasks are involved. One is to do a particle identification for each of the ROIs (the ROI task), and the other is to compute the so-called invariant mass for all possible pairs of ROIs.

By using a neural net, a "fuzzy logic" particle classification has been suggested for the ROI task [20]. The neural net is of the feed-forward type with 12 inputs, a-6 node hidden layer, and 4 output nodes.

Only two kinds of modules are necessary for implementing such an algorithm: multiply accumulate units (MAUs) and sigmoid functions. Both are very straightforward to put on DECPeRLe-1. By slightly modifying the arithmetic circuits for statistical moment used in the CALO implementation, an MAU was obtained. The sigmoid functions are easily implemented in look-up tables in the RAM. By putting the weights in RAM also, adjustment of the net can be done without recompiling the design. A highly parallellized, pipelined design is shown schematically in Figure 9.

Each of the MAUs occupy a little more than one half of an LCA.

For further details about the algorithm and the implementation see [21] and [22]

4. 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 25 MHz 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 quickly provide 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 [23], 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 are implemented in the

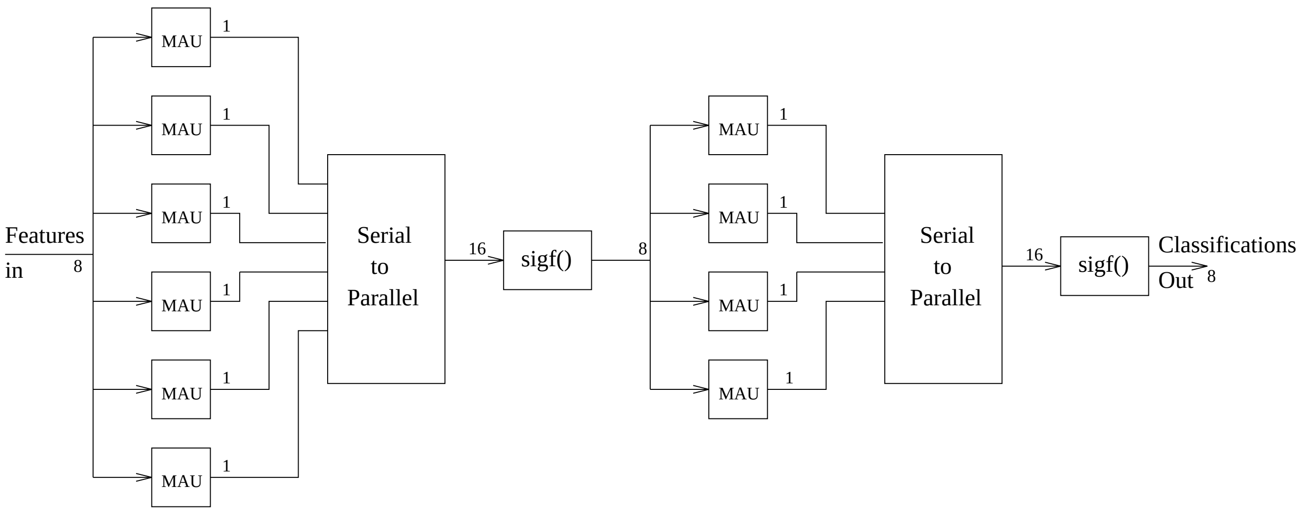

Fig. 9. Neural net datapath.

FPGAs, to control the HTI boards from the host through DECPeRLe-1, and to get data through 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 suspend 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.

5. The performance

The CERN/EAST group is testing a number of other architectures for second-level triggering, but results are really available only for software implementations [24], as a big research direction is the use of high-level processor farms. Moreover, it is not really easy to compare the performance as the specification for the detectors and the input data are quickly evolving.

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

- **ROI task of global decision:** the same algorithm was run on the same data. The software implementation uses optimized code like unrolled loops.

SPARC10	DECPERLE
9	0.64

6. The efficiency of FPGAs for High-Energy Physics applications

The difference of performance between the DECPeRLe-1 and the SPARC10 implementations shows a ratio of 15 to 500 in favor of the former. This difference is especially important for the critical detector algorithms. The price of the board is equivalent to the price of a well-equipped 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.

7. 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 performance, 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 [2], 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. Altogether it has now been demonstrated that PAM-like technology is ideal for the feature extraction phase of the second level triggering in ATLAS, and it is highly probable that it will finally be the chosen solution [26] [27].

Acknowledgements

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

This paper was previously published as "High-Energy Physics on DECPeRLe-1 Programmable Active Memory", and appeared in the proceedings of the ACM/SIGDA Symposium on Field Programmable Gate Arrays, Monterey (CA), February 1995, (c) ACM, Inc. It is reprinted by permission.

Notes

1. European Laboratory for Particle Physics, Geneva, Switzerland.
2. Embedded Architectures for Second-level Triggering [28].
3. Programmable Active Memories.
4. Meta-Systems, Jouy-en-Josas, France

References

1. J. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. Touati and P. Boucard, "Programmable Active Memories: Reconfigurable systems Come of Age," *IEEE Transactions on VLSI* (in press).
2. 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.
3. 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.
4. P. Bertin, D. Roncin and J. Vuillemin, "Introduction to Programmable Active Memories," J. McCanny, J. McWhirter and E. Swartzlander Jr. (eds.) *Systolic Array Processors*, Prentice-Hall, pp. 301–309, 1989.
5. T. Kean, and I. Buchanan, "The use of FPGAs in a novel computing subsystem," *1st International ACM Workshop on Field-Programmable Gate Arrays*, pp. 60–66, Berkeley, CA, USA, 1992.
6. B. Heeb and C. Pfister, "Chameleon, a workstation of a different color," H. Gruenbacher and R. W. Hartenstein (eds.), *Field Programmable Gate Arrays: Architecture and Tools for Rapid Prototyping*, Lecture Notes in Computer Science Nr. 705, Springer-Verlag, 1993.
7. J. Arnold, D. Buell and E. Davis, "Splash II," *4th ACM Symposium on Parallel Algorithms and Architectures*, San Diego, CA, USA, pp. 316–322, 1992.
8. 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, pp. 62–64, 1993.
9. Quickturn Systems, Inc., "RPM Emulation System Data Sheet," Quickturn Systems, Inc., 325 East Middlefield Road, Mountain View, CA 94043, USA, 1991.
10. Compugen, "The Bioccelerator," product brief, Compugen Ltd., 10 Hayetsira St., Rosh-Ha'ayin, 40800 Israël, 1993.
11. P. Bertin, "Mémoires actives programmables: conception, réalisation et programmation," *Thèse de Doctorat*, Université Paris 7, 75005 Paris, France, 1993.
12. Xilinx, Inc., "The Programmable Gate Array Data Book," Xilinx, 2100 Logic Drive, San Jose, CA 95124, USA, 1993.
13. Digital Equipment Corp., "TURBOchannel Hardware Specification," *DEC document EK-369AA-OD-007B*, 1991.

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

15. J.E. Vuillemin, J. Gutknecht (ed.), "On computing power," *In Programming Languages and System Architectures*, pp. 69-86, LNCS 782, Springer-Verlag 1994.

16. J.E. Vuillemin "Fast Linear Hough Transform," *Proceedings of the 1994 International Conference on Application-Specific Array Processors.*

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

18. 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.

19. 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.

20. R. K. Bock *et al.*, "What can artificial neural networks do for the global second level trigger," *CERN EAST note 94-08*, 1994.

21. L. Lundheim *et al.*, "A Neural Network for Global Second Level Trigger - A Real-time Implementation on DecPeRLe-1," *CERN/EAST note 95-01*, 1995.

22. L. Lundheim *et al.*, "A programmable active memory implementation of a neural network for second level triggering in ATLAS," *Proc. AIHEN95*, Pisa, 1995.

23. T. Anguelov, "HIPPI to TURBOchannel Interface," *CERN/EAST note 93-07*, March 1993.

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

25. H. Hogl, A. Kugel, J. Ludvig, R. Manner, K. Noffz and R. Zoz, "Enable++: A Second Generation FPGA Processor," *Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines*, Napa (CA), April 1995.

26. R.K.Bock *et al.*, "A commercial image processing system considered for triggering in future LHC experiments," *Nuclear Instruments and Methods in Physics Research*, Vol. A356, pp. 304-308, 1995.

27. D.Belosloudtsev *et al.*, "Programmable active memories in real-time tasks: implementing data-driven triggers for LHC experiments," *Nuclear Instruments and Methods in Physics Research*, Vol. A356, pp. 457-467, 1995.

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

Laurent Moll received Engineer degrees from Ecole Polytechnique in 1993 and from Ecole Nationale Supérieure des Télécommunications in 1995. Since 1993, he has been working in the PAM project. His fields of interest are programmable harware applications and tools, telecommunications, high-energy physics and image processing. He is currently with the new Léonard-de-Vinci University in La Défense near Paris, France. His E-Mail address is `<Laurent.Moll@inria.fr>`.

Jean E. Vuillemin is a graduate from Ecole Polytechnique. He received a Ph. D. from Stanford University in 1972, and one from Paris University in 1974. He taught Computer Science at the University of California, Berkeley in 1975, and Université d'Orsay from 1976 to 1980. He was at INRIA from 1980 to 1987, and at DEC-PRL from 1988 to 1994. He is now professor at Faculté Léonard de Vinci. He has authored over 100 papers on program semantics, algorithm design and analysis, combinatorics and hardware structures. His current research interests concern programmable hardware, theory, implementations and applications. His E-Mail address is `<Jean.Vuillemin@inria.fr>`.

Philippe Boucard received the Engineer degree from Ecole Nationale Supérieure des Télécommunications (Paris, France) in 1981. From 1991 to 1994, he worked on the PAM project at Digital Equipment Corporation's Paris Research Laboratory. He is currently with Matra MHS (France), in the microcontroller design department. His E-Mail address is `<Philippe.Boucard@matramhs.fr>`.

Lars Lundheim received M.S and Ph.D degrees from the Department of Telecommunications at the Norwegian Institute of Technology (NTH), Trondheim, Norway, in 1985 and 1992, respectively. From February 1985 to May 1992 he was a research scientist at the Electronics Research Laboratory (SINTEF-DELAB), Norwegian Institute of Technology, where he worked with digital signal processing, communications, and data compression techniques for speech and images. Since May 1992 he has been with Sør-Trøndelag College, Department of Electrical Engineering, where he is currently a Senior Lecturer. From January 1994 to July 1995 he was a Scientific Associate at CERN, European Organisation for Nuclear Research, Geneva, Switzerland. His E-Mail address is `<lars@dxrd11.cern.ch>`.