This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2688586, IEEE Transactions on Nuclear Science

A Coprocessor for the Fast Tracker Simulation

Christos Gentsos, Guido Volpi, Stamatios Gkaitatzis, Paola Giannetti, Saverio Citraro, Francesco Crescioli, Kostas Kordas, and Spiridon Nikolaidis, *Senior Member, IEEE,*

***Abstract*—The Fast Tracker (FTK) executes real time tracking for online event selection in the ATLAS experiment. Data processing speed is achieved by exploiting pipelining and parallel processing. Track reconstruction is executed in two stages. The first stage, implemented on custom ASICs called Associative Memory (AM) Chips, performs pattern matching to identify track candidates in low resolution. The second stage, implemented on Field Programmable Gate Arrays (FPGAs), builds on the pattern matching results, performing track fitting in full resolution.**

The use of such parallelized architecture for real time event selection opens up a new huge computing problem related to the analysis of the acquired samples. Millions of events have to be simulated to determine the efficiency and the properties of the reconstructed tracks with a small statistical error. The AM chip emulation is a computationally intensive task when implemented in software running on commercial resources.

This paper proposes the use of a hardware co-processor to solve this problem efficiently. We report on the implementation and performance of all the functions requiring massive computing power in a modern, compact embedded system for track reconstruction. That system is the miniaturization of the complex FTK processing unit, which is also well suited for powering applications outside the realm of High Energy Physics.

***Index Terms*—Pattern matching, Image processing, Parallel processing, Real time systems, Trigger circuits, FPGAs, Application specific integrated circuits**

I. Introduction

REAL time track reconstruction in hadron collider experiments is often considered an important feature of the event selection of such experiments. The most interesting subatomic processes are very rare and lie hidden within an extremely large amount of background information. Selecting interesting events from the background in real time is therefore essential to fully exploit the physics potential of these experiments, since only a small fraction of the produced data can be stored to be used for further analysis. A multi-level trigger [1] is an effective solution to select events. The level-1 (L1) has a tight latency constraint (a few microseconds) and is based on custom processors. It reduces the rate of events from the machine event production ($40\,\mathrm{MHz}$ for the Large Hadron Collider, LHC, at CERN) down to tens of kHz. After the L1, the high-level trigger (HLT) performs the event reconstruction (including tracking), using a large dedicated CPU farm. However the processing power necessary to perform high-quality tracking for real-time event selection at very high rates has never been available in the LHC experiments. Real-time tracking was planned since the beginning for limited detector regions or on a small subset of events previously selected using other detectors [1].

The Fast Tracker (FTK) [2], an ATLAS [3] upgrade, and similar R&D developments performed by the CMS experiment [4], provide full event real-time tracking using massively-parallel high-performance dedicated hardware.

The key feature of the architecture is a Processing Unit where track reconstruction is executed in two stages. The latest generation Field Programmable Gate Arrays (FPGAs) are interfaced with full-custom ASICs, named Associative Memories (AM), that provide the majority of the computing power [5]. Data processing speed is achieved with pipelining and parallel processing.

The AM implements the first stage processing, by recognizing track candidates ("patterns") at low resolution, by means of pattern matching. The low resolution is obtained by subdividing each layer of the detector into bins (Super Strips, SS) of equal size. A pattern is a combination of Super Strips, one in each layer. The AM chip is suitable for massive parallelism in data correlation searches. It is able to execute 5×10^{12} comparison instructions per second, featuring an equivalent memory bandwidth of $2 \times 10^{14}\,\mathrm{GB/s}$. It takes full advantage of the intrinsic parallel nature of the combinatorial problem by comparing, at once, the data under analysis to a set of pre-calculated "expectations", or patterns stored in a large "pattern bank". Patterns matching the incoming data are called "roads". This approach is very efficient and can provide solutions at the same rate as the data can be loaded into the system.

The second stage, composed of the Track Fitter (TF) and the Data Organizer (DO), is implemented in FPGAs. The DO is a form of smart database, which interfaces the low resolution pattern matching stage to the high resolution track fitting. The TF receives track candidates as combinations of high resolution hits to refine the results of the pattern matching. "Hit" refers to a particle's energy deposition cluster centroid, as the particle traverses the silicon detector layers. Track fitting is done rapidly by replacing a helical fit with a simplified calculation that is linear in the local hit position in each

Manuscript received June 24, 2016. Revised February 2, 2017.

This work receives support from Istituto Nazionale di Fisica Nucleare; Grant-in-Aid for Scientific Research from the Japan Society for the Promotion of Science and MEXT, Japan; and the European community FP7 People grants ARTLHCFE 254410 FP7-PEOPLE-2009-IOF and FTK 324318 FP7-PEOPLE-2012-IAPP.

C. Gentsos is with the Aristotle University of Thessaloniki, Greece, University of Pisa, Italy, and INFN - sector of Pisa, Italy, e-mail: cgentsos@physics.auth.gr.

G. Volpi and S. Citraro are with the University of Pisa, Italy, e-mails: saverio.citraro@for.unipi.it, guido.volpi@pi.infn.it.

S. Gkaitatzis is with the Aristotle University of Thessaloniki, Greece, and University of Pisa, Italy, e-mail: stamatios.gkaitatzis@cern.ch.

P. Giannetti is with the University of Pisa, Italy and the INFN — sector of Pisa, Italy, e-mail: paola.giannetti@pi.infn.it

F. Crescioli is with the LPNHE CNRS, Paris, France, e-mail: francesco.crescioli@lpnhe.in2p3.fr.

S. Nikolaidis and K. Kordas are with the Aristotle University of Thessaloniki, Greece, e-mails: snikolaid@physics.auth.gr, kostas.kordas@cern.ch.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2688586, IEEE Transactions on Nuclear Science

silicon layer. The calculation is a set of scalar products of the hit coordinates and the pre-calculated constants that take into account the detector geometry and alignment. Tracks satisfying a cut on the fit χ^2 are kept.

The described real time tracker provides the required performance for relatively low cost, low power consumption and small space demands compared to the CPU farm needed to provide equivalent results. However, even if the trigger problem is strongly mitigated by the dedicated hardware, the offline computing task executed by CPU farms, remains extremely complex. In fact, for each kind of trigger selection implemented in the experiment, the FTK, like any other system that performs computations in real-time, has to be simulated in detail, since its effects on the efficiency and background rejection of the acquired samples have to be carefully studied. In addition, real time tracking is very important at the trigger level, and it will be involved in almost all sample selections for more than 20 years, making the problem of Monte Carlo production for studies of trigger performances a "Big Data" problem that requires an extremely efficient solution. The emulation of the functionality of such parallelized hardware on conventional CPUs is slower compared to the performance on the dedicate hardware since the high parallelism of the hardware needs to be emulated in a large number of sequential steps on the CPUs. The large AM bank emulation in particular is quite resource-demanding: it requires a large memory and CPU time for the processing of entire events [6]: almost 6 minutes are needed at the highest pileup level foreseen for the Large Hadron Collider (LHC) before 2020. This is to be compared with $10\,\mu s$ on the real time hardware [2]. Given that hundreds of events are written on tape each second and simulated samples should amount to a volume comparable to that of the acquired samples, emulating the AM in software is impractical.

We propose as a solution the use of a hardware co-processor, made of the actual AM chips and FPGAs, to implement a software / hardware co-simulation environment. Consequently, the computational load is carried out by the hardware. Using the chip itself will guarantee the offline algorithm to be perfectly equal to the online one. Simulating the behavior of the AM chip is rendered useless, since the chip itself is used to provide the results.

We present here a novel, miniaturized implementation of the real time processing unit, the mini-PU, that is an embedded system able to handle efficiently a set of AM chips (or any other pattern matching system), and able to be interfaced with general purpose computing systems like a multi-core CPU or a GPU. The following elements are required for the mini-PU: (1) at least one powerful FPGA interfaced to a low-latency memory chip and directly connected to a pattern matching system, (2) Ethernet & PCI Express I/O for connection with general-purpose computers, (3) A configuration protocol, to configure and control the pattern bank and its behavior, (4) Processing functions (in particular data sorting and track fitting) implemented on an FPGA, to be executed in real-time.

Fig. 1 shows a candidate mini-PU, where the elements (1), (2), (3) and (4) are successfully implemented and currently used for AM chip tests in FTK [7] and CMS [8], based on a latest generation Xilinx Ultrascale evaluation board (Xilinx KCU105). We present in this paper the implementation of the fourth element, the most complex functions complementary to the pattern matching, showing much better performance than obtained in the past [2], [9].

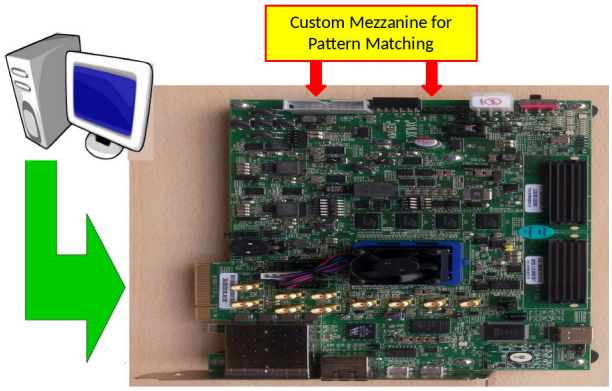

Fig. 1. EK-U1-KCU105-G Kintex Ultrascale Development board used to implement the mini-PU.

II. Beyond the State of the Art for Monte Carlo Production

Large Monte Carlo productions simulating thousands of events in the complex High Energy Physics (HEP) detectors have been always executed by large CPU farms. Tracking devices, and in particular the silicon detectors which are becoming the predominant tracking technology, play an essential role in the observation of interesting events. In fact, they provide very detailed information for charged particles and they can separate most of the different particle trajectories in the overlapping collisions recorded in the same event. However, these detectors contain hundreds of millions of channels, making the problem of complete tracking a formidable challenge even for large computing farms [10].

The amount of produced data has increased dramatically over the last years, outperforming even Moore's law. The gap between the data growth compared to transistor scaling has started becoming obvious over the last 5 years. The "Data Deluge" is increasing computation requirements, demanding new processing and computing methods, new storage devices to tackle the increasing gap between the growth of data and processing power.

In addition, further scaling of transistor leads to steadily increasing power consumption (Dennard scaling law), high power density and high clock frequencies, heading away from the operating points of the most efficient computing system, the biological brain [11].

Compared to CPU / GPU implementations, FPGA-based hardware accelerators can achieve up to 25x higher performance per watt and 50–75x smaller latencies [12]. FPGAs can provide much higher energy efficiency in several applications like sliding-windows applications and in some cases orders of magnitude better than CPUs and GPUs [13]. The fine-grain programmable logic elements of FPGAs can provide high flexibility but they come with a need of a large area for the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2688586, IEEE Transactions on Nuclear Science

switching elements that connect the configurable blocks. When targeting specifically dataflow-based applications, an efficient utilization of the programmable elements can further reduce the power consumption by 4x–8x.

Our mini-PU is in line with the innovative ideas of new brain-like architectures and cognitive computing systems recently proposed to face the Data Deluge [11]. This technology, that was developed for the trigger systems of hadron collider experiments, is based on a very efficient pattern matching algorithm implementation, exploiting the low level parallelized, dedicated hardware for an extremely efficient pre-processing step. This organization is similar to models of the vision processing task performed by the brain. The most convincing models that try to validate brain functioning hypotheses are extremely similar to the real-time architectures developed for HEP. A multilevel model seems appropriate to describe the brain organization for image processing [14]: "the brain works by dramatically reducing input information by selecting for higher-level processing and long-term storage only those input data that match a particular set of memorized patterns. The double constraint of finite computing power and finite output bandwidth determines to a large extent what type of information is found to be meaningful or relevant and becomes part of higher level processing and longer-term memory".

Our mini-PU has been used for image pre-processing [15]. The goal of this work, instead, is to provide the same advantages and organization to the offline computing. It is a co-processor that reduces strongly the amount of information for the high level algorithms, providing a very efficient dedicated hardware solution for a specific task, the full simulation of the online tracker, that would require enormous resources if executed on general purpose computers.

III. The mini-PU Hardware Implementation

The new embedded system (mini-PU) has to perform the two main algorithms executed by the FTK Processing Unit [2]: the Pattern Matching (PM) and the track fitting. They are executed in a pipeline, and interfaced by the Data Organizer (DO) that provides reduced resolution hits to the Pattern Matching and well organized full resolution hits to the TF.

Fig. 1 shows our mini-PU based on a latest generation Xilinx Ultrascale evaluation board. A mezzanine with devices performing Pattern Matching can be connected to the large connectors on the top of the board. The mezzanine organization offers flexibility: it allows using the system in different configurations and with different AM chip versions (parallel or serial I/O, different pattern densities). This system is scalable, since the needed number of mini-PUs can work in parallel to obtain the required pattern bank size or timing performance.

The Data Organizer and the Track Fitter have been implemented on the Kintex Ultrascale. The implementation does not depend on the type of pattern matching device used on the mezzanine. Fig. 2 shows the diagram of the FPGA logic with all the connections to the needed external devices: (a) the IPbus [16] for communication with the CPU; (b) high frequency serial links (GTX) to communicate with the AM pattern bank (AM); (c) a RAM controller to handle a large external look-up table (AMMap) that, when addressed by a found road, is able to provide the addresses needed to recover the full resolution hits associated to the road.

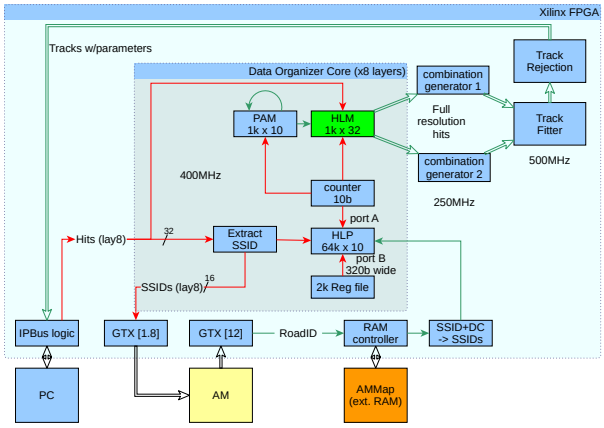

Fig. 2. Simple firmware block diagram.

A. The Data Organizer

The Data Organizer is a core component of the FPGA firmware, interfacing the pattern matching step with the generation of hit combinations and the full-resolution track fitting. It stores the full resolution hits (writing phase, whose data paths are shown in red in Fig. 2) according to their 16-bit Super-Strip identifiers (SSIDs, representing low resolution hits), and later retrieves them (reading phase, whose data paths are shown in green in Fig. 2) according to the pattern matching results.

There are a number of requirements that have to be met by the DO, making the design of a fast and compact implementation challenging. First of all, it has to support the AM Chip's Don't Care capability [4]. Each pattern and each detector layer within a pattern is able to use the global SS width, but we can use the "don't care" feature to increase the width when that is more appropriate. Patterns are able to match a maximum of 8 consecutive SSs per detector layer. The Data Organizer must be able to handle a variable number of consecutive SSs to select hits belonging to a road that can have variable size. This improves the overall efficiency of the pattern bank, effectively allowing variable shape patterns. Furthermore, there should be no hard limit imposed on the number of hits that can be stored for a Super-Strip (SS). Finally, there should be no restriction on the order in which the full resolution hits arrive.

A simplified block diagram of the DO and its role within the system can be seen at Fig. 2. The basis of the DO architecture is the DO core, which is replicated for each detector layer. In this way, all cores are working in parallel and independently to each other. Inside the DO core, there is a set of BRAM-based memories working together to form the equivalent of a "collection of linked lists" in hardware; one linked list for each SSID. The DO core is built around three main memories: the Hit List Memory (HLM), the Hit List Pointer (HLP), and the Previous Address Memory (PAM). The full resolution hits are written in the HLM in the order in which they arrive (in Fig. 2 this is represented by the "counter 10b" which is incremented

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2688586, IEEE Transactions on Nuclear Science

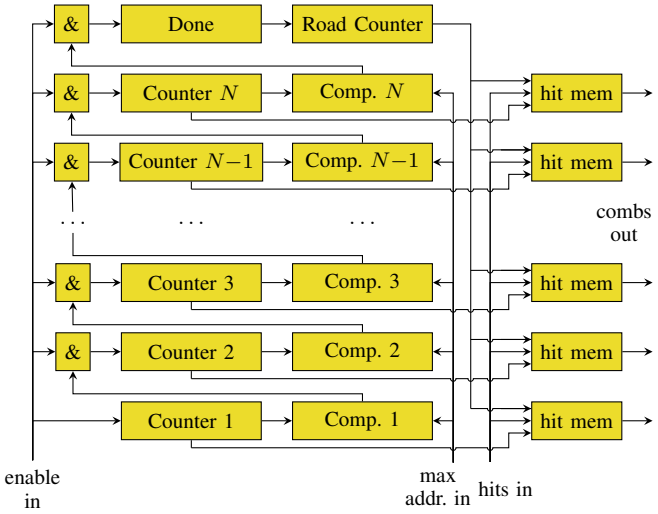

Fig. 3. Simple combiner block diagram.

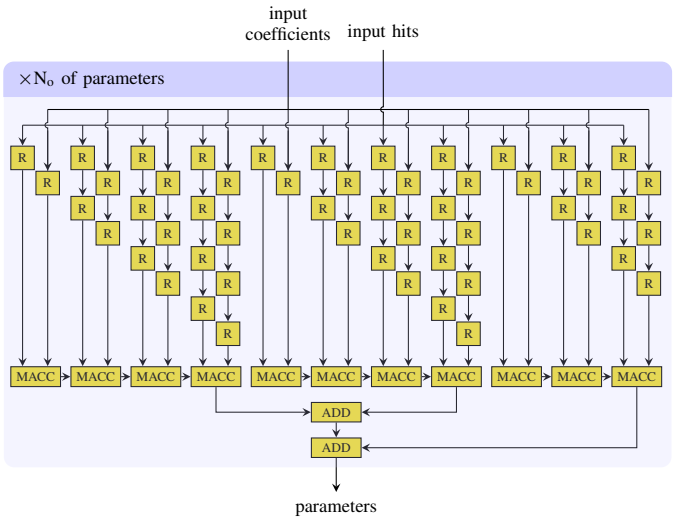

Fig. 4. Simple TF block diagram.

at each hit arrival). The HLP memory locations have a one-to-one correspondence to all possible SSIDs. At the same time the hit arrives to the HLM, the corresponding SSID arrives to address the HLP memory, whose memory content is then updated with the address of the last hit arrived to the HLM for that specific SS. At the same time, the PAM, having the same size as the HLM, is updated with the invalidated address the HLP held for that SS. Thus, the PAM holds the address in the HLM of the previous hit that belongs to the same SSID, if there is one; or zero, if the first hit of the SSID is concerned.

To read the contents of a SS, when a road is received, the HLP is addressed by the SSID generated by the AMMap, producing the address in the HLM of the last hit that appeared for that SS. The PAM initially gets addressed by that and then by its own content, producing one by one the HLM addresses of the rest of the hits belonging to that SS, until it produces an address of zero, when there are no more hits to be read. At that point, the HLM / PAM memory combination is ready to process the next SS. To keep the read bandwidth high, the HLM / PAM memories are replicated, and two ports per memory are used. Thus, when one unit is busy reading the hits of a SS, the next SS can be read out by other memory ports.

To account for the case of a missing layer and support the Don't Care pattern matching capability, the HLP contents have to be reset across events. That is due to the fact that in these cases, the DO may be requested to read the contents of SSIDs for which no data have been written in an event. If no provision is made for that fact, and the HLP memory contents are not reset, leftover HLP data from previous events may be read instead, erroneously. However, BRAM memory contents cannot be reset in a single clock cycle. To overcome this, a Region Valid (RV) register file is used, which is reset at the beginning of each event. To reduce the size of the register file (2k instead of 64k, which would be necessary to accommodate all possible 16-bit SSIDs), the HLP memory width has been selected in such a way that each HLP memory location actually covers 32 SSIDs. Thus, each bit of the RV register file indicates if current event data have been written in any of the 32 SSIDs that make up its corresponding region; only a subset of the 2k regions are activated in any event. The first time in an event that data are written in a region, the whole region gets reset before the writing. On the read phase, if data are requested from a non activated region, the request is ignored. This also has a favorable impact on performance: it makes it possible to read in one clock cycle, all the possible hit locations of the SSIDs requested by patterns with Don't care bits enabled. Hence, on the reading phase, the HLP can fetch all the addresses for each pattern, on each clock cycle. As mentioned before, the dual-port HLM / PAM memories are duplicated, so there are four channels reading at the same time. That provides the necessary memory bandwidth to support the fast, parallelized HLP output.

The maximum operating frequency of the Data Organizer is $f_{max,DO} = 400\,\mathrm{MHz}$. When writing it accepts batches of 8 hits, followed by idle periods of 8 clock cycles. In the first set of 8 cycles, the RV and the HLP contents are read. All possible collisions within the batch of 8 hits are examined, and in the following 8 clock cycles the correct data are written to the memories. Thus, the input rate for the hit writing phase is $200\,\mathrm{MHits/layer/s}$. Thanks to the parallelism and replication of the memory structures, the hit reading phase has a maximum output rate of $1600\,\mathrm{MHits/layer/s}$.

B. The Combiner and the Track Fitter

Between the DO module and each TF unit, a Combiner module is necessary to compute all the possible hit combinations that can form valid tracks.

The Combiner's function is simple, but in order to be able to process one combination per clock cycle, the hardware implementation has a long critical path, which prevents its operating frequency from matching that of the TF. To effectively utilize the computing resources, two combiner units are assigned to each TF, each running at a frequency exactly half that of the TF ($f_{max,Comb} = 250\,\mathrm{MHz}$), with their results

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2688586, IEEE Transactions on Nuclear Science

being multiplexed on its input. Every clock cycle, a Combiner unit produces exactly one combination. It's worth to note that between successive roads, there is no downtime, making it possible to process a different one-track road on each clock cycle.

The block diagram of the Combiner unit core can be seen in Fig. 3. Each hit memory is split into sectors, each one filled with the hits from a specific road. During the read state, the sector data are written by the DO, asynchronously to the Combiner, in a cyclic way. That is similar to the way an asynchronous FIFO memory handles its data, with empty and full flags being generated for their corresponding clock domains to protect the hit data from being overwritten. Due to its simple structure, the architecture is easily adaptable for an arbitrary number of layers. Since increasing the number of layers will bring the operating frequency down from $f_{max,Comb} = 250\,\mathrm{MHz}$ in the case of 8 layers, one can exploit higher level parallelism by changing the ratio of Combiner units per TF.

The TF units extract the five helix parameters from the local hit coordinates by employing linear functions and using a χ^2 cut as a goodness of fit indicator. The resolution of the resulting fit approaches that of a full helical fit, as long as the fit coefficients are generated for a narrow sector of the detector.

Making extensive use of dedicated DSP units in modern FPGA devices, one can realize a very fast implementation. The block diagram of the TF architecture can be seen in Fig. 4. The architecture tries to strike a balance between low latency and resource usage (both DSP and register), while maintaining a high operating frequency and exploiting the parallelism capabilities of FPGAs. Each TF unit operating at $f_{max,TF} = 500\,\mathrm{MHz}$ can output a fit every $2\,\mathrm{ns}$, after an initial latency of $\sim$90 ns. Four units can fit in a modern mid-range device, providing a maximum fitting performance of $2\,\mathrm{GFits/s}$. Such a high performance can help in maintaining the trigger efficiency in a high occupancy environment.

IV. Event Processing Time and Validation

While the AM chip performance is well known since the chip is an ASIC and has a fixed hardware configuration and has been used extensively in the FTK project [17], the performance of the algorithms implemented on the FPGA is much more sensitive to the specific design and working conditions.

The FPGA design performance has been studied using the simulation of a "toy" detector that resembles the silicon detector geometries of the LHC experiments. The detector simulation is strongly simplified compared to the LHC detector simulations (e.g. detector material and noise are not simulated), however it is appropriate for our main goals which are: (a) to measure the mini-PU processing speed as a function of number of roads to be processed in the event and fits per real track to be executed; and (b) to verify that the algorithms implemented in the FPGA do not have any loss of precision compared to the equivalent ones executed in the CPU. The simulated events reproduce the occupancy generated by crowded, energetic jets. Inside those jets the number of fits

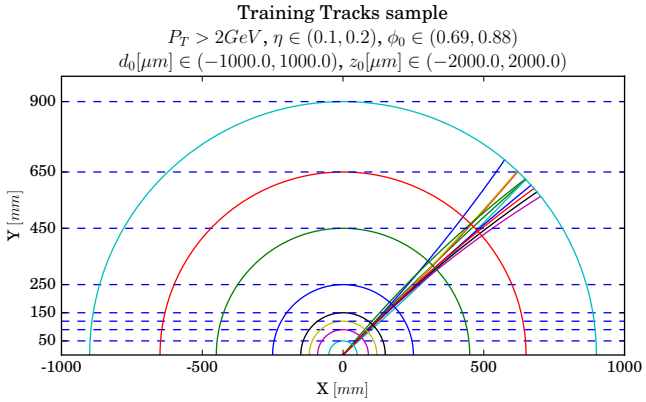

Fig. 5. The detector geometry with concentric cylindrical layers, with a z axis coinciding with the axes of the cylinders, along the beam direction.

to be executed increases exponentially due to the fact that two or more tracks can enter inside the same road [18]. Detector details are not relevant for our study since the overload caused by jets is enough to produce the desired exponential grow of found roads and fits. Fig. 5 shows a cross-section transverse to the beam. The three inner cylindrical layers represent pixel detectors and the five outer layers, strip detectors.

A stand-alone simulation program has been used to generate tracks in a restricted phase space. The generated tracks have a flat P_T distribution (momentum transverse to the beam axis, z) starting from a minimum of $2\,\mathrm{GeV}$, a uniform impact parameter in the transverse plane, $d_0[\mathrm{mm}] \in (-1, 1)$, and a uniform distance from the center of the detector along the beam, $z_0[\mathrm{mm}] \in (-2, 2)$. The direction of the tracks with respect to the z axis, $\eta = -\ln(\tan(\theta/2)) \in (0.1, 0.2)$, and with respect to the x axis on the transverse $x - y$ plane, $\phi_0[\mathrm{rad}] \in (0.69, 0.88)$, represents a narrow region, as shown in Fig. 5. Quasi-realistic resolutions are used in the "toy" experiments to generate the hit positions in the detector layers: $8\,\mu\mathrm{m}$ for the pixel layers in both directions, and $20\,\mu\mathrm{m}$ for the strip layers.

A. The processing time as a function of the detector occupancy

The simulation program performs the two fundamental steps to reconstruct the event, in order to reproduce the behavior of the hardware procedure: (1) coarse track (road) finding: all roads are found by simulating the pattern matching; (2) track fitting: each found road is processed to compute the track parameter values of the best quality tracks and to reject the fake ones.

Because of the low resolution used for the PM, a road may contain physical hits belonging to different particles. Also, depending on the road size and on the event hit density, there is a level of combinatorial background consisting of fake roads, i.e., track candidates that will be rejected at full resolution. Therefore, the processing time of the track fitting strictly depends on the following quantities:

- occupancy: the track density, which is the average number of tracks in the area handled by a mini-PU

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2688586, IEEE Transactions on Nuclear Science

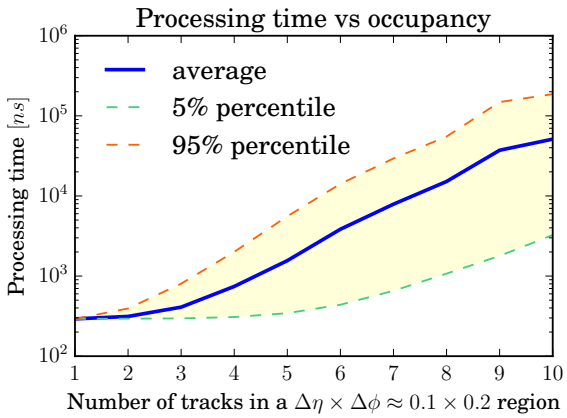

Fig. 6. The processing time as a function of the number of generated tracks inside the phase space $\Delta\eta \times \Delta\phi \approx 0.1 \times 0.2$.

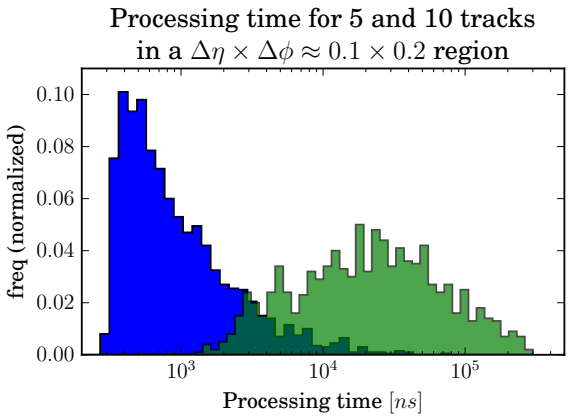

Fig. 7. Histogram of the processing times for 5 and 10 tracks inside the phase space $\Delta\eta \times \Delta\phi \approx 0.1 \times 0.2$.

- roads per track: the average ratio between the number of matching roads and the number of actual tracks per event;
- combinations per road: the average number of hit combinations to be fitted per road.

The average number of fits executed per real track is calculated as the product of the roads per track and the combinations per road, and is strongly dependent on the occupancy of the detector: when many tracks are in a small phase space, like inside high energy jets, the number of fits to be executed increases substantially, and with it, the time needed to process such events.

The effect of the detector occupancy on the processing time is shown in Fig. 6. The processing time is studied as a function of the main parameter of the system, the track density in the region processed by a mini-PU, a cone of size $\Delta\eta \times \Delta\phi \approx 0.1 \times 0.2$ for this study. For each different occupancy value, many equivalent events are generated and processed by the mini-PU. For five and ten tracks in this narrow space, a histogram of the processing time can be seen in Fig. 7. The processing time adheres very well to the formula $t_{proc} = 290 + 2N_{fits}$[ns], which describes two components: a fixed initial latency; and a factor that depends on the number of fits to be performed. The distribution for five tracks is asymmetric, as the initial latency is a significant component of the distribution, skewing it to the right. Both distributions have long tails towards high values, corresponding to the cases of large number of fits to be processed. The average number of roads for five tracks is $\overline{N}_{roads,5} = 34$ and for ten tracks it is $\overline{N}_{roads,5} = 197$; while the average number of fits is $\overline{N}_{fits,5} = 683$ and $\overline{N}_{fits,10} = 25403$, respectively. The Super-Strip size is a constant in the system, and chosen for each detector layer to be: $SS_{pix}[\text{mm}] = \{(1,1),(2,1),(3,1)\}$, $SS_{strip}[\text{mm}] = \{2,3,10,22,44\}$

For each fixed detector occupancy the processing time of the mini-PU is measured, and the average for a large number of events is calculated. Fig. 6 shows the average processing time for each detector occupancy. For 1-3 tracks per event the processing time is almost constant, just $\sim$300 ns. For crowded events it increases and it becomes $\sim$50 µs in the case of events with an occupancy of 10 tracks, which means that $\sim$5 µs are needed per track. This is a very good result, considering that the last point represents a really extreme case, that can be compared to the most energetic and crowded jets [19]. Real events are more complete since they contain additional hits from spurious other collisions, low pT particles, and detector noise, however for a rough estimation of number of fits to be executed, latency and number of fakes, the contribution of stiff, collimated, very near tracks is the most relevant.

B. Hardware Validation and Tracking Performance

The simulation produces input test vectors for the FPGA (the detector hits, the roads found by the AM, and the fitting constants) and generates the expected output, fitted tracks, to check the hardware functionality and to validate the tracking quality of as a function of track density.

The simulation was run over thousands of tracks to validate the hardware, and there was a bit-level agreement between the simulation and the hardware results for all the relevant quantities: the fitted parameters and the χ^2 values, used to select the real tracks and reject the fakes. Fig. 8 shows the theoretical χ^2 distribution (for 6 degrees of freedom), superimposed to the χ^2 distribution of fits executed by a floating point calculation (CPU task), and by a fixed point calculation (mini-PU task) on events with a single track inside the cone, to exclude fakes and only consider good tracks. The distributions overlap perfectly. Requiring a χ^2 of less than 20, retains 99.5 % of the tracks; this is the cut used to reject the fake tracks.

Fig. 9 shows the track efficiency and the percentage of fakes as a function of the occupancy in the region shown in Fig. 5, $\Delta\eta \times \Delta\phi \approx 0.1 \times 0.2$. The efficiency is flat and high (greater than 99 %) for any detector occupancy, while the number of fakes is very low for a small occupancy (<2 % for a track density less than 3) and reaches 8 % for a high detector occupancy. The last point in the plot corresponds to 10 tracks in a region equivalent to a jet cone of radius $\sim$0.22. This is a very high density of tracks and the increase of fakes

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2688586, IEEE Transactions on Nuclear Science

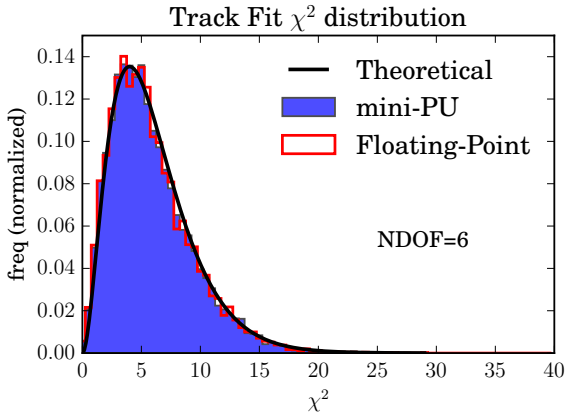

Fig. 8. χ^2 distributions for single track fits executed with floating point calculation in CPUs and fixed integer calculation executed by the mini-PU, superimposed to the theoretical χ^2.

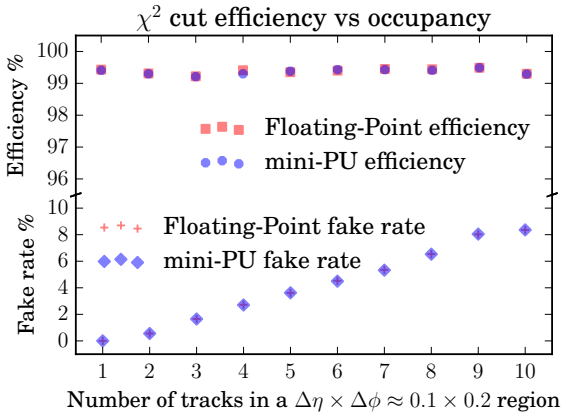

Fig. 9. The track efficiency and the number of fake tracks as a function of the number of generated tracks inside the phase space $\Delta\eta \times \Delta\phi \approx 0.1 \times 0.2$.

is an expected effect. The χ^2 cut can be tuned choosing lower values to reduce the fakes at the price of a lower efficiency.

Fig. 10 shows that the resolution of the fitted curvature ($1/P_T$) and impact parameter (d_0), for tracks that have a good χ^2 (below the cut at 20) in a minimum occupancy environment, is in excellent agreement between the CPU and mini-PU calculations. The resolutions shown in Fig. 10 ($\sigma_{1/P_T} = 6.5 \times 10^{-4}$, $\sigma_{d_0} = 9\,\mu\text{m}$) reflect the quasi-realistic resolution used in the "toy" experiments ($8\,\mu\text{m}$ for the hit positions on the pixel layers, and $20\,\mu\text{m}$ for the strip layers).

V. Track Reconstruction Performance Comparison

The new implementations of the DO and TF algorithms is quite improved even with respect to the FTK [2]. The presented firmware runs at frequencies of 250 MHz, 400 MHz and 500 MHz (see Section III) well above the 200 MHz clock of the same FTK functions.

An interesting technology which recently has attracted the attention of the high energy physics community for real-time applications is GPU processing. Both ATLAS [17] [20] and CMS [21] are studying the performance of real time tracking at LHC executed on modern GPUs. Even if the comparison with the CPU performances is promising, the latency to execute tracking is at least tens of milliseconds for simplified algorithms and lower detector occupancy, with a fast grow above hundreds of milliseconds when the occupancy increases. In conclusion, our hardware approach is today thousands of times faster than any available commercial computing device.

The short latency achievable by the parallelized AM system, in conjunction with the FPGA implementation presented here, is today thousands times faster than any commercially available device. Therefore, both CMS and ATLAS consider its possible application at the first level trigger (L1) [4] for the future accelerator upgrades, when the LHC luminosity will cause hundreds of pile-up collisions and will require much faster and more efficient trigger selections.

VI. Conclusions

The implementation and performance of a hardware co-processor for the simulation of online, very parallelized tracking have been described. All the functions complementary to the pattern matching, have been implemented on an FPGA device, using a development board. A python program implementing a simplified toy detector model was designed to help with the characterization and validation of these functions. The hardware was successfully validated, with the hardware results matching the software results perfectly for thousands of events. The performance of the implemented system in terms of speed is many times faster than any other existing solution. The DO runs at 400 MHz, providing a maximum output rate of 1600 MHits/layer/s to the TF that running at 500 MHz is able to output 1 fit each 2 ns. This performance surpasses even that of the original FTK system. The resolution of the fixed-point implementation of the fitting algorithms is also very close to that of the floating-point CPU implementation. In conclusion the mini-PU is a good candidate to speed up the simulation of thousands of Monte Carlo events needed to study the trigger efficiency in experiments that are planning the use of highly parallelized online tracking devices.

References

[1] W. H. Smith, "Triggering at LHC experiments," *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 478, no. 12, pp. 62 – 67, 2002, proceedings of the ninth Int.Conf. on Instrumentation. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016890020101720X

[2] ATLAS Collaboration, "Fast TracKer (FTK) Technical Design Report," CERN, Geneva, Tech. Rep. CERN-LHCC-2013-007. ATLAS-TDR-021, Jun 2013. [Online]. Available: {http://cdsweb.cern.ch/record/1552953}

[3] ATLAS Collaboration, "The ATLAS Experiment at the CERN Large Hadron Collider," *J. Instrum.*, vol. 3, p. S08003. 437 p, 2008, also published by CERN Geneva in 2010. [Online]. Available: https://cdsweb.cern.ch/record/1129811

[4] A. Annovi, G. Broccolo, A. Ciocci, P. Giannetti, F. Ligabue, D. Magalotti, A. Nappi, M. Dell'Orso, R. Dell'Orso, F. Palla, E. Pedreschi, M. Piendibene, L. Servoli, S. Taroni, and G. Volpi, "Associative Memory for L1 Track Triggering in LHC Environment," *IEEE Trans. Nucl. Sci.*, vol. 60, no. 5, pp. 3627–3632, Oct 2013.

[5] M. Dell'Orso and L. Ristori, "VLSI structures for track finding," *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 278, no. 2, pp. 436 – 440, 1989. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0168900289908620

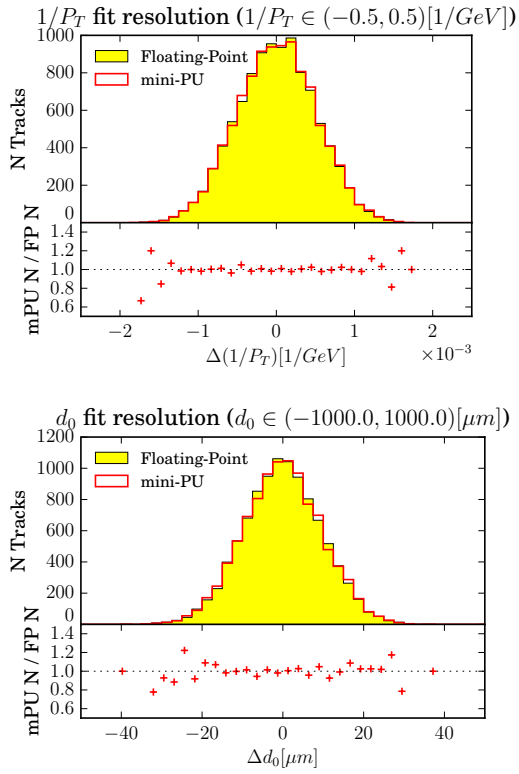

Fig. 10. The curvature (top) and impact parameter (bottom) resolution for good tracks with a $\chi^2 < 20$.

[6] L. S. Ancu, A. Annovi, D. Britzger, P. Giannetti, J. W. Howarth, C. Luongo, C. Pandini, S. Schmitt, and G. Volpi, "Associative memory computing power and its simulation," in *Real Time Conference (RT), 2014 19th IEEE-NPSS*, May 2014, pp. 1–5.

[7] A. Andreani, A. Annovi, R. Beccherle, M. Beretta, N. Biesuz, M. Citterio, F. Crescioli, P. Giannetti, V. Liberali, S. Shojaii, and A. Stabile, "Characterisation of an associative memory chip for high-energy physics experiments," in *2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, May 2014, pp. 1487–1491.

[8] D. Magalotti, L. Alunni, N. Biesuz, G. Bilei, S. Citraro, F. Crescioli, L. Fan, G. Fedi, G. Magazz, L. Servoli, L. Storchi, F. Palla, P. Placidi, E. Rossi, and A. Spiezia, "A pattern recognition mezzanine based on associative memory and FPGA technology for level 1 track triggers for the HL-LHC upgrade," *J. Instrum.*, vol. 11, no. 02, p. C02063, 2016. [Online]. Available: http://stacks.iop.org/1748-0221/11/i=02/a=C02063

[9] S. Amerio, A. Annovi, M. Bettini, M. Bucciantonio, P. Catastini, F. Crescioli, M. Dell'Orso, P. Giannetti, D. Lucchesi, M. Nicoletto, M. Piendibene, and G. Volpi, "The GigaFitter: A next generation track fitter to enhance online tracking performances at CDF," in *Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE*, R. C. Lanza, Ed. IEEE, Oct 2009, pp. 1143–1146. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5384532

[10] The CMS Collaboration, "Description and performance of track and primary-vertex reconstruction with the CMS tracker," *J. Instrum.*, vol. 9, no. 10, p. P10009, 2014. [Online]. Available: http://stacks.iop.org/1748-0221/9/i=10/a=P10009

[11] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, aug 2014. [Online]. Available: https://doi.org/10.1126/science.1254642

[12] *The Xilinx SDAccel Development Environment*, Xilinx, 2014. [Online]. Available: http://www.xilinx.com/support/documentation/backgrounders/sdaccel-backgrounder.pdf

[13] J. Fowers, G. Brown, P. Cooke, and G. Stitt, "A performance and energy comparison of fpgas, gpus, and multicores for sliding-window applications," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '12. New York, NY, USA: ACM, 2012, pp. 47–56. [Online]. Available: http://doi.acm.org/10.1145/2145694.2145704

[14] M. M. Del Viva, G. Punzi, and D. Benedetti, "Information and perception of meaningful patterns," *PLoS ONE*, vol. 8, no. 7, pp. 1–9, 07 2013.

[15] C.-L. Sotiropoulou, P. Luciano, S. Gkaitatzis, S. Citraro, P. Giannetti, and M. Dell'Orso, "High performance embedded system for real-time pattern matching," *Nucl. Instrum. Methods Phys. Res., Sect. A*, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168900216305721

[16] C. G. Larrea, K. Harder, D. Newbold, D. Sankey, A. Rose, A. Thea, and T. Williams, "IPbus: a flexible Ethernet-based control system for xTCA hardware," *J. Instrum.*, vol. 10, no. 02, p. C02019, 2015. [Online]. Available: http://stacks.iop.org/1748-0221/10/i=02/a=C02019

[17] D. Emeliyanov and J. Howard, "GPU-Based Tracking Algorithms for the ATLAS High-Level Trigger," *J. Phys. Conf. Ser.*, vol. 396, no. 1, p. 012018, 2012. [Online]. Available: http://stacks.iop.org/1742-6596/396/i=1/a=012018

[18] A. Annovi *et al.*, "Hadron collider triggers with high-quality tracking at very high event rates," *IEEE Trans. Nucl. Sci.*, vol. 51, pp. 391–400, 2004.

[19] ATLAS Collaboration, "Measurement of the jet fragmentation function and transverse profile in proton–proton collisions at a center-of-mass energy of 7 TeV with the ATLAS detector," *Eur. Phys. J. C*, vol. 71, no. 11, p. 1795, 2011. [Online]. Available: http://dx.doi.org/10.1140/epjc/s10052-011-1795-y

[20] J. Mattmann and C. Schmitt, "Track finding in ATLAS using GPUs," *J. Phys. Conf. Ser.*, vol. 396, no. 2, p. 022035, 2012. [Online]. Available: http://stacks.iop.org/1742-6596/396/i=2/a=022035

[21] V. Halyo, A. Hunt, P. Jindal, P. LeGresley, and P. Lujan, "GPU enhancement of the trigger to extend physics reach at the LHC," *J. Instrum.*, vol. 8, no. 10, p. P10005, 2013. [Online]. Available: http://stacks.iop.org/1748-0221/8/i=10/a=P10005