Associative Memory computing power and its simulation.

L. S. Ancu
University of Geneva
E-mail: Lucian.ancu@unige.ch

A. Annovi
Laboratori Nazionali di Frascati
E-mail: alberto.annovi@lnf.infn.it

D. Britzger
Deutsches Elektronen-Synchrotron (DESY)
E-mail: daniel.britzger@desy.de

P. Giannetti
INFN Pisa
paola.giannetti@pi.infn.it

J. W. Howarth
Deutsches Elektronen-Synchrotron (DESY)
E-mail: james.william.howarth@cern.ch

C. Luongo
INFN Pisa
E-mail: carmela.luongo@pi.infn.it

C. Pandini
Università di Milano
E-mail: carloenrico.pandini@studenti.unimi.it

S. Schmitt
Deutsches Elektronen-Synchrotron (DESY)
E-mail: sschmitt@mail.desy.de

G. VOLPI[*†]
University and INFN Pisa
E-mail: guido.volpi@cern.ch

The associative memory (AM) chip is an ASIC device specifically designed to perform “pattern matching” at very high speed and with parallel access to memory locations. The most extensive use for such a device will be the ATLAS Fast Tracker (FTK) processor, where more than 8000 chips will be installed in 128 VME boards, specifically designed for high throughput in order to exploit the chip’s features. Each AM chip will store a database of about 130000 pre-calculated patterns. Any data inquiry is broadcast to all memory elements simultaneously within the same clock cycle (10 ns), thus making data retrieval independent of the database size.
Speed and size of the system are crucial for real-time High Energy Physics applications, such as the ATLAS FTK processor. Using 80 million channels of the ATLAS tracker, FTK finds tracks within 100 μs. The simulation of such a parallelized system is an extremely complex task when executed in commercial computers based on normal CPUs. The algorithm performance is limited on CPU, due to the lack of parallelism, with the additional issue of the very large memory requirement. In fact the AM chip uses a content addressable memory (CAM) architecture. We report on the organization of the simulation into multiple jobs to satisfy the memory constraints and on the optimization performed to reduce the processing time.
Finally, we introduce the idea of a new computing unit based on a small number of AM chips that could be plugged inside commercial PCs as coprocessors. This unit would both satisfy the need for very large memory and significantly reduce the simulation time due to the use of the highly parallelized AM chips.

Technology and Instrumentation in Particle Physics 2014
2-6 June, 2014
Amsterdam, the Netherlands

*Speaker.
†Supported by FTK-IAPP project

1. Introduction

High energy physics experiments at the Large Hadron Collider (LHC) are facing an important real-time computational problem, with the need to extract information from a very dense environment where thousands of hits are produced by hundreds of particles generated in the multi-TeV collisions. This is particular important in the data acquisition system, where most of the collisions need to be rejected because of limitations in the data storage. The experiments indeed need to identify the interesting events in a trigger system, to allow the physicists to focus only on those specific events.

The ATLAS trigger is a common example: designed as a multi-level system with a Level-1 made of custom hardware and electronic boards, with a input rate up to 40 MHz and an accept rate up to 100 KHz, a maximum latency of 20 μs. The next stage is named High Level Trigger (HLT) and is based on a farm of commercial PC, connected through a high performance network, that will further reduce the output rate to about 400 Hz, with an average latency of about 500 ms.

In the coming Run II, starting from spring 2015, the experimental condition will be quite challenging: the energy of the collisions will reach 13 or 14 TeV, with up to 80 contemporary proton-proton collisions at each bunch crossing (pileup). The most challenging aspect would be the pileup level, with the result of increasing the confusion in the event, because objects originating from different collisions can be impossible to distinguish, resulting in a far less effective event selection.

To mitigate the effect of the pileup the use of the inner tracker detectors is crucial. This detector is important because of the high precision in measuring the parameters of single particles, allowing to identify the contribution of each collision to physics objects, for example as jets. Other well known contribution of the ID is in identifying final states, such as b-jets or τ-jets.

With the goal to support tracking in the trigger Run II ATLAS will install an electronic processor, the Fast Tracker (FTK)[1], that would allow to reconstruct, in real time, the tracks in the region covered by the silicon inner detector. The system can cope with the computational challenge using fast electronic components as the Associative Memory (AM) chips [3] and high level commercial FPGAs. This document will focus on the AM chip, a custom memory chip specifically designed to help fast track identification at hadronic colliders and used with great success at the CDF experiment, operating on the Tevatron accelerator at Fermilab until 2010 [4].

2. The ATLAS Fast Tracker Processor

The FTK processor will be installed within the ATLAS TDAQ infrastructures and will work as support of the existing HLT computing farm. The system is designed as a set of parallel pipelines of electronic boards, based on VME and ATCA computing standards. The design is modular and will allow the system to evolve according to the experimental conditions.

The tracks are reconstructed receiving data from both pixel and strip silicon sensors, installed in the most internal part of the ATLAS inner detector (ID). The ID is composed of 12 concentric layers: 4 pixel sensor layers in the innermost part, followed by 4 pairs of strip layers up to a radius of about 56 cm. The ID data will be collected by the FTK at each Level-1 trigger accept, at the

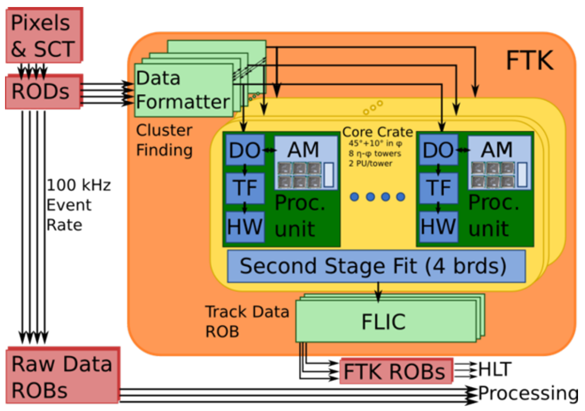

Figure 1: The sketch shows how the FTK is connected to the ATLAS TDAQ. The internal components of the system are also highlighted.

maximum rate of 100 KHz, from all the read-out drivers. The system will then perform the track reconstruction within a total average latency smaller than 100 μs.

As shown in Fig. 1, the FTK electronic system input stage is performed by the Data Formatter (DF), implemented in a set of ATCA boards. This part receives the data from the ID and performs the cluster finding on the incoming data. The clusters are then routed, according to the module position to one, or more, of the independent pipelines that perform the real track finding. The number of pipelines for the final FTK system, is 64 and follows the cylindrical symmetry of ATLAS ID detectors: the ID is segmented in 4 barrels along Z and 16 slices on the transverse plane. Each pipeline is also referred as a tower and covers about 1.2 units in pseudo-rapidity, $\eta = -\log(\tan\theta/2)$, and about 22.5 degree in the transverse plane.

In the final system each tower has then 2 VME boards that contain the core of the processor. Each board contains 64 AM chips and exchanges data through a rear module, the auxiliary (AUX) card. The AUX card tasks are not limited to I/O and communication but have an active computational role and each AUX card is equipped with 4 Arria V Altera FPGA. At this stage the processor uses only 8 layers out of the 12, allowing 1 missing layer. The 8-layer track candidates from the AUX card are sent to the 2nd stage fit board (SSB), where the track candidates from the AUX are refined adding up to 4 additional hits, resulting in an improvement in resolution and a better rejection of fake tracks. The final board is the FTK to Level 2 Interface (FLIC), where the data after the SSB are collected and prepared for the ATLAS HLT processing farm.

The system is designed to allow track reconstruction with a minimum transverse momentum of 1 GeV, with $|\eta| < 2.5$, representing the limit of the ID reconstruction. The absolute tracking reconstruction efficiency is about 93% for muon tracks, the average fraction of fake tracks at the highest expected pileup scenario of 80 overlapping collisions would be about 5%. Those parameters depend on optimization studies and can change during the final optimization.

3. The FTK emulation framework

To support the hardware development a full simulation of the system has been developed. The software has two main goals: describe the logic of the main parts inside the processor, to verify

Element type	**Unit Size**	**Number of units**	**Memory**
Pattern bank	36 B	10^9	36 GB
Fit Constant (1st stage)	616 B	$\approx 10^6$	500 MB
Fit Constant (2nd stage)	1 KB	$1.5 \cdot 10^6$	1.5 GB

Table 1: The table reports the largest configuration elements that the simulation has to load in order to reproduce the HW behavior. The numbers refer to a complete FTK configuration, for a maximum luminosity of $3 \times 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$, with 128 core units boards.

the computational load on the different components, and evaluate the quality of the output tracks and the trigger selections built on those. The current code is composed of more than 20 thousands lines of C++, able to describe the logic of the main components of the FTK pipeline. The code is highly modular and allows to change all the parameters that control the hardware, allowing to test new features. It can be used as standalone software or as a part of the ATLAS main simulation and software framework, ATHENA[5].

The code has a very limited dependency from external libraries, limited to ROOT[7] and BOOST[8], available within the ATLAS computing framework and generally common in installation for HEP computing. The data persistency uses ROOT, ensuring the flexibility to store the complex objects used to describe roads and tracks, while allowing evolution of the data format and performance scalability.

All the code is grouped in 2 main blocks: road finding (RF) and track fitting (TF). The RF part controls the input: the emulation of FTK clustering mezzanine as well as the distribution of the hits in the towers, done by the DF. Then the pattern matching of the AM chips is carefully emulated. The RF output is composed of a list of output roads and the associated hits. Additional statistical information on the RF workload is also saved with the roads to study the load of the related HW parts.

The TF parts emulate the 2$^{\text{nd}}$ FTK fitting stage and prepare the tracks for the HLT integration. This emulation part loops over the list of roads, retrieving the hits associated for each road and composing all the track candidates with one hit in each layer and performs a track fit using up to 8 hits.

The simulation is characterized by the use of large set of constants representing the patterns used for the RF and values used by the fit, see Table 1. The FTK final system will have about 1 billion patterns, a sequence of 8 numbers of 18 bits, that in the simulation are rounded to 9 integers of 32 bits, to store a reference to the fit constants, totaling a minimal request of about 36 GB of memory. The fits require about 20 thousands set of constants per tower, with still a large memory requirement. The global memory requirement is extremely high and has indeed created the need to develop precise strategies to perform simulation of the whole FTK system. In particular we were forced to divide the simulation task in independent jobs, each controlling only a fraction of the system, that can be smaller than a single tower. This segmentation of the simulation task allows to fit it into a standard machine, with about 2 GB/core of memory and up to 4 GB of memory per process.

The current segmentation is 256 jobs, 4 for each hardware tower. The jobs of each tower are executed in sequence, with a partial merge at the end of the last job. This proliferation complicates

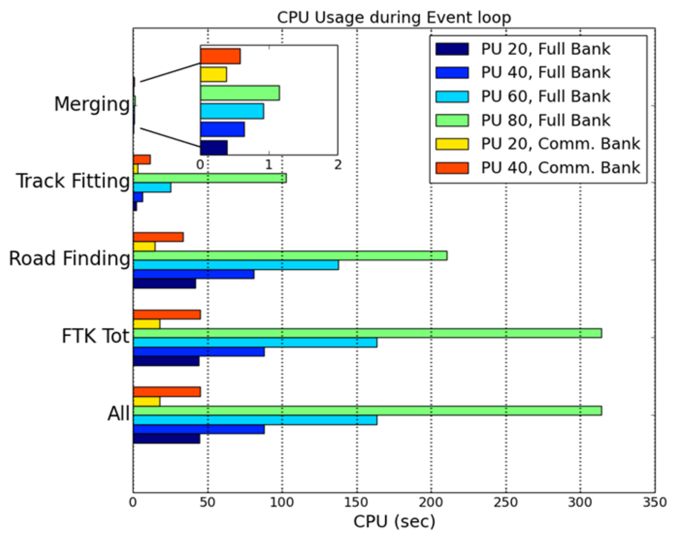

Figure 2: The bars show the execution time of the main FTK emulation parts, the different colors represent different FTK configurations and simulation based on events with different pileup (PU) levels.

the emulation workflow but allows to run with a minimal amount of memory, of about 1.5 GB for the large banks. This limitation can be removed using special high-memory machines, but they are a rare resource in the ATLAS computing model, mostly used for tests and small MC events production, with the consequence of a reduction of the effective usable CPU resources.

The emulation performance has been evaluated as function of the pattern bank size and pileup condition of the physics event. Fig. 2 shows how the emulation of the road finder stage dominates the overall execution time. Only at the highest pileup levels the track fitter starts to be important.

4. The AM features and computing power

The AM chip in FTK performs an extremely intensive task: it compares all the hits with a huge set of patterns. Within the AM there is a total of 64 chip for AM board. The version of the AM chip for the FTK will be the AMchip06, able to store 128K patterns per chip, working at a clock 100 MHz clock, with a consumption of about 2.5 W per chip. Each pattern is composed of 8 words of 18 bits, send through independent buses. The chip will be able to make a comparison for every location at every clock cycle. The comparison of a single word can use “don’t care” (DC) feature [6], allowing to tune the precision of the match in each location. These numbers and characteristics will allow to have up to 1 billion of patterns available for the final FTK system.

All the communication within the AM boards are controlled by high speed serial link, with an extremely large bandwidth. To bring the hits to the AM chips and to collect the results each AM board has about 750 serial links, with a total bandwidth of about 200 GB/s. The whole 128 boards within FTK reach a total bandwidth of about 25 TB/s, used for the pattern matching functionality only. To simplify the comparison of the AM system performance with standard CPU in terms of computing power we can round the depth of buses as 4 32 bit words for each pattern; this means that at each clock cycle 128 thousands comparisons of 4 words are made, so roughly 500

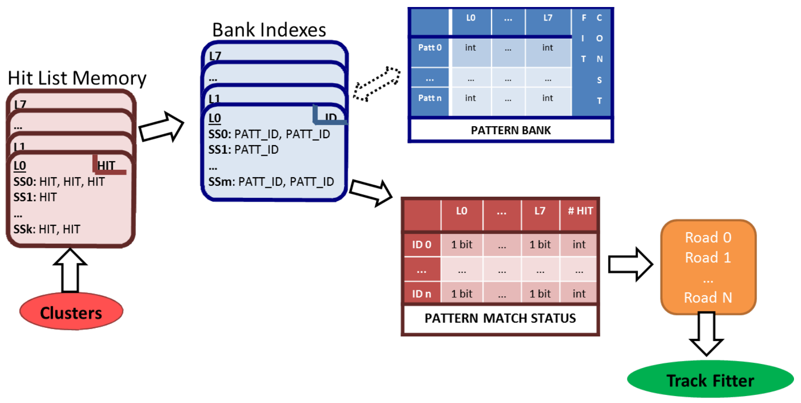

Figure 3: The scheme shows the working principle of the AM emulation. The incoming clusters are first grouped according the super-strip (SS) value, the found SSs in an events are then compared with an indexing structure that allows to speedup the update of the matching status for each of the involved patterns. When the match is completed the the roads are sent to the output.

thousands comparison per clock cycle or 50×10^6 million of comparison instructions per second (MIPS). Scaling this to the whole AM system, composed by 128 boars, with 64 chips each, this is equal to 40×10^{10} MIPS. Such a huge number instructions per second cannot be accomplished in any current CPU, not even super computers.

5. AM emulation software characteristics and performance

As described in section 3 the FTK group has developed a software able to emulate the logic of the system. The AM emulation module workflow is shown in Fig. 3 and it is the most important and time consuming part of the whole simulation. The hardware pipeline shown in Fig 1, for technical reasons, integrates or controls functionalities in the emulated AM module, that in real hardware belongs to the AUX and the DF board.

In order to emulate the AM working principle the code needs some internal structure to store the data in the chip, the status of the match during an event, and the data to be sent to the next emulation step. The first data structure is the pattern's table: this stores the description of the patterns. This table has a number of columns equal to the number of layers used by the chip plus one integer as reference for the fit constants, for the current FTK this means 9 columns. The number of rows is equal to the number of patterns loaded in the module, that is limited to several millions. This table is loaded at the initialization of the emulation. A second data structure records the status of the match of the patterns during the event. This structure has the same size but for each pattern a single bit per layer is enough and the 8 bits required by a single pattern are grouped together.

In order to match an incoming hit with the patterns, the simplest approach is a linear scan. This has been proven to be extremely time-consuming because of the size of the pattern bank. Instead of linear scan, an additional data structure is used to index the pattern by Super Strip. In this way an incoming hit is compared to the pattern index for its layer and the list of patterns is quickly retrieved, allowing an update of the match status avoiding the linear scan. As further improvement

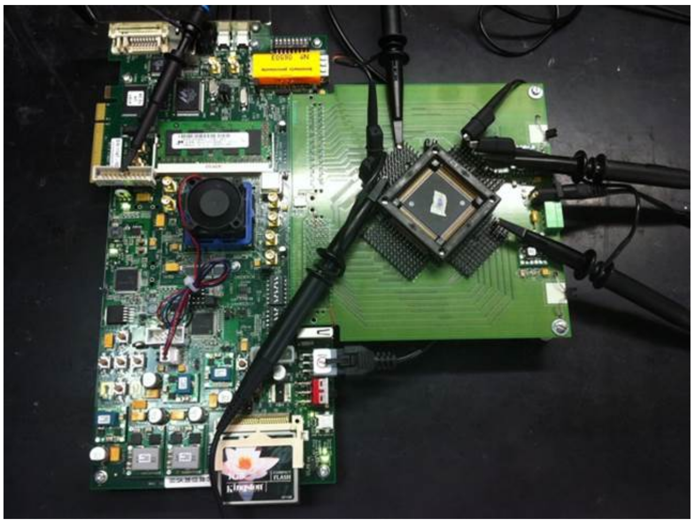

Figure 4: The picture shows the board used for the AM chip tests [9]. This board is based on a Xilinx Kintex 7 development board, with AM chips installed on an external mezzanine, compatible with the format used on the AM boards.

this is done only when a SS is found for the first time in an event, the following hits are only accumulated because they will be used for the track fitting in the proper emulation module.

A final structure related to the AM work is the list of "fired patterns", updated at runtime while the hits are received. This avoids a linear scan of the patterns at the end of the event. Despite the optimization effort, as shown in Fig. 2, the AM emulation remains the top consumer of the FTK emulation. This part of the code consumes about 2/3 of the execution time at the highest pileup level, or more at lower pileup level conditions. This justifies why the ongoing optimization efforts focus mostly on this component.

6. AM accelerator concept

The previous section has shown how the computational power of the AM chip is impossible to match with commercial CPU-based system. Optimization of the emulation code is ongoing, with possible large improvements, but still the expected performance is many order of magnitudes slower than the real AM system.

To overcome the intrinsic limitation of the CPU-based system in performing comparison operations, natural for the AM chips, an attempt to develop an external coprocessor equipped with AM chips is ongoing. The idea is to free the PC from the need to perform the comparison between the hits and the patterns, that represents the most difficult part for the PC, connecting the PC with a board that has AM chip.

The first prototype for a similar system is the AM chip test board, Fig. 4. This board has been used to connect a single AM chip to a host PC. The communication, such as setting up test patterns and sending a set of hits able to test the small pattern bank, happens through the ethernet port. The current planning is to extend these functionalities by connecting the board to a standard program while plugging in additional mezzanine boards, able to store more patterns.

7. Conclusions

We have presented the ATLAS FTK processor, expected to start operating after summer 2015 to support the ATLAS HLT in the track finding. The document highlighted how the performance of the system depends on the extreme computing power of the core components, in particular of the AM chip: able to compare the incoming data with 1 billion of patterns every 10 ns. The power of the AM is clear when compared to standard CPU units, that have clear limitations in the memory size available by normal programs, the speed in accessing the location and performing the pattern matching in a way compatible to the AM chip. One only has to compare the emulation time of few minutes on a CPU with the few 10s of μs time of the FTK processor.

The emulation of the working principle of a system based on AM chips with commercial CPU is challenging and has required to develop special solutions. To accomplish the task a network of jobs performs the system emulation and overcome memory and computing power limitation. The results are good but further improvements are required. To reduce the unavoidable bottleneck in the memory access the integration of AM-based extension cards will be pursued. This approach can guarantee a better performance for the pattern matching, that represents the slowest part of the emulation. But it still requires further development steps.

References

[1] A. Annovi et al., Fast TracKer (FTK) Technical Design Report, CERN-LHCC-2013-007, ATLAS-TDR-021

[2] R. Bartoldus et al, Technical Design Report for the Phase-I Upgrade of the ATLAS TDAQ System, CERN-LHCC-2013-018, ATLAS-TDR-023

[3] F. Morsani et al., The AMchip: A VLSI associative memory for track ïňĄnding, Nucl. Instrum. Meth. A315, 446 (1992)

[4] S. Belforte et al., The CDF trigger silicon vertex tracker (SVT), IEEE Trans.Nucl.Sci. 42, 860 (1995)

[5] ATLAS Collaboration, ATLAS computing: Technical design report, CERN-LHCC-2005-022 (2005), ATLAS-TDR-017.

[6] A. Annovi et al., A new variable-resolution associative memory for high energy physics, (Int. Conf. on Advancements in Nuclear Instrumentation Measurement Methods and their Applications, ANIMA), Proc. IEEE 116, 117 (2011).

[7] Rene Brun and Fons Rademakers, ROOT - An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also http://root.cern.ch/

[8] Boris Schaeling, The Boost C++ Libraries. en.highscore.de/cpp/boost/

[9] Alberti F. et al, Performance of the AMBFTK board for the FastTracker Processor, ATL-DAQ-PROC-2012-062