# GPU-based Computation of Energy and Time for the Upgrade of the Tile Calorimeter of the ATLAS Detector

**Marc Sacks, Bruce Mellado**

School of Physics, University of the Witwatersrand, Johannesburg 2050, South Africa

E-mail: marc.sacks@cern.ch

**Abstract.** After the expected 2022 upgrade of the Large Hadron Collider, increased running luminosity will necessitate the redesign of the front-end and back-end detector electronics. The massive increase in raw data generated by the ATLAS Tile Calorimeter (TileCal) as well as the increased out of time pile-up will require more sophisticated hardware and signal processing algorithms for accurate energy reconstruction. ARM processors are common in mobile devices due to their low cost, low energy consumption and good performance. Coupled with high performance general purpose GPU processing, a cost-effective, high data throughput Processing Unit (PU) is being developed at the University of the Witwatersrand. This PU, working in tandem with the planned Tile Pre-Processor (TilePPr), will enable computationally complex signal processing on the TileCal raw data while maintaining minimal software design difficulty for the end-user. The GPU will allow for parallel processing, which can be used effectively in the context of the detector which has nearly 10000 independently read-out channels. An overview of the planned PU design is given. Performance and throughput test results for several implementations of algorithms on the NVIDIA TK1 GPU are given.

## 1. Introduction

A Toiroidal LHC ApparatuS (ATLAS) is a detector in the Large Hadron Collider (LHC) at CERN [1]. It is a general purpose detector used in the search for Standard Model and beyond the Standard Model physics, comprising several sub-detectors; namely the inner detector (ID), the electromagnetic and hadronic calorimeters, and the muon spectrometer. These detectors measure the energy and trajectories of the particles formed within it due to high-energy proton-proton (p-p) and heavy ion collisions. The calorimeter has two subsystems, namely the electromagnetic (EM) and hadronic calorimeters. The EM detector is a liquid argon based detector, while the hadronic calorimeter is composed of sheets or tiles of plastic scintillators sandwiched between inactive plates of steel. The tiles of plastic scintillator have led to the hadronic calorimeter being referred to as the Tile Calorimeter or TileCal [2]. Proton bunches collide in ATLAS approximately every 50 ns. This bunch crossing rate produces hundreds of millions of collisions per second, which the TileCal read-out system reads out at a total bandwidth of about 205 Gbps [3]. After the upgrade of TileCal, in 2022, this will increase to 41 Tbps [3]. Storing this amount of data is neither possible nor desirable, as such a trigger system is used to separate rare physical processes from background events. The approach is to detect events of interest in real-time and to store those events for analysis; all other data

is discarded. The trigger system must also deal with increased pile-up, which will occur as a function of increased luminosity. To accommodate for the increased bandwidth and pile-up, new trigger systems are being developed [4]. The current and future trigger systems are composed of 3 and 4 separate trigger levels respectively, with a read-out driver (ROD) forming the interface between the Level-1 and Level-2 triggers [4]. The ROD is being upgraded to the so-called Tile Pre-Processor (TilePPr). This paper will describe a proposed co-processing unit (PU) for the TilePPrs based on ARM and graphic processing unit (GPU) technology. First a more complete description of the trigger system and associated hardware is given in section 2, followed by a description of GPUs and parallel processing systems in section 3, and finally a discussion of the TilePPr co-processing unit and results to some preliminary tests of trigger algorithms implemented on GPUs are given in section 4.

## 2. ATLAS Trigger System

On the boundary between the Level-1 and Level-2 triggers are the read-out drivers (ROD) [2]. The RODs are responsible for reading in data from the Level-1 trigger upon a Level-1 accept and transmitting this data to the next level of the trigger [3]. The RODs are also responsible for the reconstruction of the time-energy profiles of collision events, calculating the quality-factor of these reconstructions, synchronising data and trigger, compressing digital samples, calculating transverse energy, and other maintenance activities [3]. In Phase-II, the Level-1 trigger will be implemented at the level of the TilePPr [4].

### *2.1. Trigger Hardware*

Current hardware designs are largely based on Field Programmable Gate Arrays (FPGA) [3]. FPGAs are unique pieces of hardware in that they behave somewhat like a blank slate of an integrated chip (IC). The user of an FPGA is in control of an array of programmable logic blocks and can configure them in many ways. For instance, an FPGA can be programmed to behave like a collection of operational amplifiers, or even to behave like something as complex as a CPU. In many ways an FPGA can be seen as a formattable ASIC and indeed can be used to emulate an ASIC before it is physically created. The current Level-1 trigger algorithm has been implemented as a massively parallel system on FPGAs and the entire TileCal processing chain relies on its rapid latency [3]. Parallel systems are described in the following section. The RODs and TilePPrs also make use of FPGAs, and additionally DSPs [3]. The HLT uses computer farms consisting of x86 CPU architecture, however research into the use of GPUs to speed up processing time is being investigated [5]. The versatility, customizability and performance of FPGAs are diminished by their high cost and difficulty of use. FPGA programming is low-level and unfamiliar to many programmers, requiring a knowledge of electrical circuits, and therefore they do not lend themselves to rapid prototyping and alteration of code.

## 3. Parallel Systems and Parallel Processing

Parallel processing refers to the concurrent processing of data by two or more processors [6]. Concurrent processing usually implies a speed-up as compared to the serial processing of the same system. That multiple processors can function simultaneously usually implies that they can function independently of one another. This is not always the case; processors may need to communicate with one another, and the extent of their independence can be used as an indicator of the degree of parallelism of the system being analysed. The degree of parallelism can in turn be used to estimate the amount of speed-up that can be expected from the parallel processing of a system. A simple example of a parallel system is the process of matrix multiplication, where entries of the output matrix can be calculated separately and simultaneously. TileCal can be viewed as a parallel system of 10000 channels before the trigger selection algorithm, as

the trigger might need to access one or many channels at once As such, the hardware used in the trigger system takes advantage of this parallelism.

*3.1. Parallel Processing with Graphic Processing Units*

Graphic Processing Units (GPU) are responsible for generating images for display on screen. This can be understood as a parallel process: pixels on a screen can be thought of as entries in a matrix and a GPU can process these elements independently. The parallel processing capabilities of GPUs has been extended into fields other than graphics processing and this is referred to as general purpose graphic processing units (GPGPU) [7]. GPUs are typically less expensive than FPGAs and they can be programmed in a variety of languages, for instance in a C/C++ platform developed by NVIDIA called Compute Unified Device Architecture (CUDA) [7]. The programming languages C, C++, and their derivatives are widely known and as such GPGPU makes it possible for non-specialists to program parallel code rapidly, as well as to understand, use, and alter existing code with ease.

## 4. TilePPr Co-Processing Unit

An ARM-based processing unit (PU) which will operate in conjunction with the TilePPr is being developed at the University of the Witwatersrand [8]. ARM processors are low-cost, energy efficient alternatives to traditional x86 architecture processors. As such they are found in many mobile devices (cellular phone, tablets) since the 1990s. ARM processors are performant in their own right, but to take advantage of the massively parallel environment of TileCal, the introduction of GPGPU into the PU is being investigated. Details such as the number and configuration of the GPUs in the PU are at this stage unclear. In the early testing phases of the PU, the GPU(s) will most likely interface with ARM processors with PCIe x3.

The PU is intended to offer the necessary computational power to deal with the increased pile-up of the HL-LHC. Algorithmic methods aimed at providing an improved calculation of quality-factor as well as improved resolution of events suffering from pile-up are under investigation [9]. These signal-processing techniques could be implemented partially or wholly on the PU. To determine what techniques could be implemented on the PU, further investigation into the performance of GPGPU in conjunction with ARM processors must be conducted.

Two algorithms were implemented here; the Optimal Filtering (OF) algorithm which is currently implemented on the RODs and used for online energy reconstruction, and an algorithm which reconstructs pulse shapes based on a deconvolution technique. OF calculates the amplitude, baseline, and phase shift of a given pulse using three sets of weights combined with a set of samples in a dot product operation. The deconvolution technique approximates the transfer function of the detector system and, knowing the output, can calculate the input. The deconvolution algorithm has shown promising results with regard to energy estimation in high pile-up environments especially when used in conjunction with OF. Details of the OF and deconvolution algorithms can be found in [9] and [10], for the purposes of this text it is sufficient to understand them as simple matrix multiplication operations.

The results in table 1 are based on the implementation of these two algorithms on the NVIDIA TK1 GPU. The table gives results of the performance of the OF algorithm alone and OF running in parallel with the deconvolution algorithm.

The algorithms were run on randomly generated static data; future implementations will use streamed simulated detector data. The following observations can be made regarding the results shown in table 1. Computation time increases linearly with number of events when running both algorithms. However, it can be seen that when only the OF algorithm is performed on the events, the performance is drastically reduced. Performing only OF on the data results in a computation time up to 7.5× slower than with the combined algorithms. Because the GPU is performing both algorithms in parallel, it is more efficient to run multiple algorithms on the same data set

**Table 1.** GPU Performance for OF Only and OF and Deconvolution Combined

| Number of Events | Computation Time ($\mu s$) | GFlop/s[†] |
|---|---|---|
| 96 (OF Only) | 565 | 0.09 |
| 96 | 81 | 0.61 |
| 128 | 167 | 0.39 |
| 256 (OF only) | 2136 | 0.06 |
| 256 | 295 | 0.44 |
| 512 | 555 | 0.47 |
| 512 (OF only) | 2826 | 0.06 |
| 1024 | 1085 | 0.48 |
| 4096 | 4221 | 0.50 |

[†]b of floating point operations per second

simultaneously. This is desirable in the context of the PU where running as many algorithms as possible on a given data set will enable the passing of richer results to the Level-2 trigger. The poor performance of the GPU when running OF only is also due to the way hardware manages memory. The layout of GPU memory is more difficult to manage than a typical CPU and performance is heavily reliant on this. To illustrate this consider the results shown in table 2.

**Table 2.** GPU Performance for Alternate Implementation of OF and Convolution

| Number of Events | Computation Time ($\mu s$) | GFlop/s |
|---|---|---|
| 128 | 86 | 0.74 |
| 512 | 163 | 1.61 |
| 4096 | 1066 | 0.61 |

This alternate implementation of the algorithms is achieved by varying the dimensions of the matrices undergoing multiplication, and results in a much reduced computation time. The implementation of each individual algorithm as well as the interactions of the algorithms with each other will be the subject of future investigation.

## 5. Conclusion

The HL-LHC will require an upgraded trigger system to deal with increased data production and increased pile-up. A co-processing unit for the TilePPr based on ARM and GPU technology is under investigation. The parallel nature of the energy reconstruction in the TileCal detector makes the use of parallel processing technology an attractive option. It is possible to implement a parallel system with FPGAs, however they are expensive and difficult to program. GPGPU provides a less expensive alternative and one which is also easier to program and update. Preliminary test results indicate that GPGPU could be used in the high throughput environment of TileCal as part of a PU operating in assistance to the TilePPr.

**References**

[1] The ATLAS Collaboration *et al* 2008 *JINST* **3** S08003
[2] ATLAS Trigger Group *et al* 2008 *JINST* **3** P03001
[3] Carrió F, Mellado B, Reed R *et al* 2014 *JINST* **9** C02019
[4] ATLAS Collaboration 2012 CERN-LHCC-2012-022. LHCC-I-023
[5] Halyo V, Hunt A, Jindal P, LeGresley P, Lujan P 2013 *JINST* **3** P10005
[6] Pacheco P S 2011 *An Introduction to Parallel Programming* (USA: Elsevier Inc)
[7] Sanders J, Kandrot E 2010 *CUDA by Example* (Addison-Wesley Professional) 2-7
[8] Cox M, Reed R, Mellado B 2015 *JINST* **10** C01007
[9] Balabram L E, Seixas J M, Filho L M 2014 Quality Factor for the Hadronic Calorimeter in High Luminosity Conditions *J. Phys.: Conf. Ser.* 608 012044 http://iopscience.iop.org/1742-6596/608/1/012044)
[10] Stärz S 2012 *JINST* **7** C12017