Using FPGA Co-processors for Improving the execution Speed of Pattern Recognition Algorithms in ATLAS LVL2 Trigger

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von
Dipl. Physiker Andrei Khomich
aus Soligorsk

Mannheim, 2006

Dekan:	Prof. Dr. Matthias Krause, Universität Mannheim
Referent:	Prof. Dr. Reinhard Männer, Universität Mannheim
Korreferent:	Prof. Dr. Peter Fischer, Universität Mannheim

Tag der mündlichen Prüfung: 4. Oktober 2006

ZUSAMMENFASSUNG

ATLAS (A Toroidal LHC ApparatuS) ist einer der vier Detektoren am neuen Teilchenbeschleuniger Large Hadron Collider (LHC) der Europäischen Organisation für Kernforschung (CERN), der im Jahr 2007 in Betrieb gehen wird. Bei ATLAS handelt es sich um einen Mehrzweck-Detektor, der die Produkte von Proton-Proton-Kollisionen mit der Schwerpunktenergie 14 TeV registrieren wird.

Die *pp*-Kollisionsrate von 40 MHz bedingt ein schnelles und effizientes Triggersystem, das die Ereignisrate auf rund 100 Hz reduzieren muss. Der ATLAS-Trigger hat eine dreistufige Architektur. Die rein auf Hardware basierende erste Stufe (LVL1) trifft Entscheidungen aufgrund der schnellen Analyse der Daten von den Kalorimeter- und Muon-Sub-Detektoren. Die nächsten zwei Stufen – 2. Stufe (LVL2) und Ereignis-Filter (EF) – bilden den "High-Level-Trigger" (HLT). Beide beruhen auf Softwarealgorithmen, die auf PC-Farmen laufen.

Die Spurrekonstruktion spielt bei der Ereignisauswahl im ATLAS HLT eine entscheidende Rolle. Die früheste Phase, in der Spurinformationen verwendet werden können, ist der LVL2. Die Ereignis-Verarbeitung muss innerhalb von 10 ms durchgeführt werden. Diese Einschränkung, zusammen mit der hohen Anzahl an Spuren aufgrund der Vielfach-Kollisionen, stellt sehr große Herausforderungen an die Spurrekonstruktionsalgorithmen.

Im Rahmen dieser Arbeit wurde ein möglicher Ansatz zur Beschleunigung der Spurrekonstruktionsalgorithmen unter Einsatz von hybriden, FPGA/CPU-basierten Systemen untersucht. Der TRT LUT-Hough-Algorithmus – einer der Spurrekonstruktionsalgorithmen für den ATLAS LVL2 – wurde für diesen Zweck ausgewählt. Bei diesem Algorithmus handelt es sich um eine "Look-Up-Tabellen"-basierte (LUT) Hough-Transformation für die Teilchenidentifikation im Übergangs-Strahlungs-Detektor (TRT). Dieser Algorithmus wurde speziell für die B-Physik-Aufgaben entwickelt und beinhaltet eine schnelle Suche für niederenergetische Teilchenspuren im ganzen TRT-Volumen. Ein derartiger "Full Scan" benötigt eine extrem hohe Rechenleistung.

Der TRT LUT-Hough-Algorithmus besteht aus einer Spur-Kandidaten-Suche gefolgt von einem Fit-Verfahren, das durchgeführt wird, um die Spur-

Parameter zu bestimmen. Da die zu findenden Teilchentrajektorien im Voraus berechnet werden können, ist die Hough-Transformation gut für die anfängliche Spurensuche im TRT geeignet. Der Algorithmus beruht auf der Idee, dass jeder Punkt ("Hit") im dreidimensionalen Detektor-Bild mehreren möglichen (vorherbestimmten) und durch verschiedene Parameter charakterisierten Spuren angehören kann. Alle diese Spuren ("Roads") werden in der LUT gespeichert. Jeder weitere Hit vergrössert die Wahrscheinlichkeit für die Existenz der jeweils zugehörigen Spuren (Histogrammier-Schritt). Das Histogramm für eine Spur besteht aus einer "bow-tie"-geformten Region von Histogrammzellen mit Einträgen mit einem Scheitelpunkt im Zentrum der Region. Die Zelle am Scheitelpunkt des Histogramms enthält im idealen Fall alle Hits der Spur. Die umliegenden Zellen, die jeweils eine Road repräsentieren, enthalten jeweils nur ein Subset der Hits der Spur. Das Histogramm für ein komplizierteres Ereignis besteht aus einer Überlagerung der Einträge von den mehreren Spuren. Die Zellen, die den gesamten Satz von Hits von jeder Spur enthalten, können als lokale Maxima im Histogramm identifiziert werden.

Der TRT LUT-Hough-Algorithmus wurde in C++ implementiert und ins ATLAS-Softwareframework integriert. Die Zeitmessung der C++-Implementierung des Algorithmus' zeigt, dass der grösste Teil der Rechenzeit in den Teilschritten "Histogramming" und "Thresholding / Local Maximum Finding" verbraucht wird (∼62% der Gesamtverarbeitungszeit). Diese zwei Stufen sind gute Kandidaten für eine FPGA-Implementierung. Für jeden "Hit" werden in jeweils einem Schleifendurchgang ∼120 – den vorherbestimmten Spuren zugeordneten – Histogrammzähler erhöht. Dies wird in der CPU seriell abgearbeitet. Im FPGA kann dies parallel erfolgen.

Zum Nachweis der Machbarkeit ("proof of concept") der Algorithmus-Beschleunigung mit der Hilfe eines FPGA-Koprozessors wurden die ersten Stufen des TRT LUT-Hough-Algorithmus' in VHDL implementiert. Der an der Universität Mannheim entwickelte FPGA-Koprozessor MPRACE ("Multi Purpose Reconfigurable Accelerator / Computing Engine") wurde als Hardwareplattform für den Test verwendet. MPRACE ist ein FPGA-Koprozessor in Form einer 64Bit/66MHz-PCI-Karte. Herzstück der Karte ist ein VirtexII-FPGA des Herstellers Xilinx. Es handelt sich um einen universell nutzbaren Koprozessor, der bereits für verschiedene Anwendungen eingesetzt wird.

Die Implementierung nutzt sowohl den SRAM auf der Koprozessor-Karte als auch die internen RAM-Blöcke des FPGAs aus. Die LUT wird im SRAM gespeichert und das interne Block-RAM wird für die Histogrammierzähler verwendet. Das FPGA-Design wird durch ein 64 MHz-Taktsignal synchron betrieben.

Die hybride Implementierung des Algorithmus' (der zeitaufwändige Teil

des Algorithmus wird durch den FPGA-Koprozessor beschleunigt, alle anderen Teile laufen auf der Host-CPU), wurde für den Vergleich in dasselbe Softwareframework wie die C++-Implementierung integriert. Die rein CPU basierte Version und die hybride Implementierung liefern identische physikalische Ergebnisse. Die Ergebnisse der Zeitmessung zeigen, dass der in VHDL implementierte zeitkritische Teil auf dem FPGA Koprozessor ca. 4-mal schneller läuft als auf der modernen Vergleichs-CPU (Intel Xeon 2.4 GHz). Der gesamte Algorithmus läuft unter Nutzung des FPGA-Koprozessors ca. 2-mal schneller.

Für eine noch höhere Beschleunigung muß ein weiterer Teil des Algorithmus' ("Track Splitting") für die Ausführung im FPGA implementiert werden. Wir erwarten, dass die FPGA-Realisierung des "Track Splitting" mindestens dieselbe Beschleunigung wie die Stufe "Histogramming / Maximum-Finding" gibt. In diesem Fall wird die hybride Implementierung sogar um einen Faktor ∼3.2 schneller sein als die reine CPU-Implementierung.

Die Speicherbandbreite ist ein kritischer Punkt für die FPGA basierte Algorithmusausführung. Das Design des MPRACE-Koprozessors erlaubt es, den On-Board-Speicher mittels aufsteckbarer Subboards zu erweitern. Dies führt zu einer Verdopplung der Speicherbandbreite und gibt der Algorithmus-Implementierung zusätzlich einen Beschleunigungsfaktor in der Größenordnung von zwei.

Eine zusätzliche Beschleunigung kann erreicht werden, wenn ein FPGA der neuen Baureihe Virtex4 des Herstellers Xilinx zum Einsatz kommt. Das VHDL-Design der ersten Stufe des Algorithmus' wurde geringfügig modifiziert und für die Virtex4-FPGA-Architektur synthetisiert. Die Durchführung des "Place-and-Route"-Prozesses ergibt eine Taktfrequenz für den Design-Core von ∼180 MHz. Das führt zu einem zusätzlichen Beschleunigungsfaktor von ∼3 und zu einem Gesamtbeschleunigungsfaktor von rund 10.

ABSTRACT

ATLAS (A Toroidal LHC ApparatuS) is one of the four detectors at the Large Hadron Collider (LHC) facility at the European Organization for Nuclear Research (CERN) which will start operation in 2007. It is a general purpose detector which will detect the products of proton-proton collisions with center of mass energy of 14 TeV.

The LHC bunch crossing rate of 40 MHz implies fast and efficient trigger system which must bring the event rate to the order of 100 Hz. ATLAS Trigger has a three-level architecture. The hardware-based first level (LVL1) makes decision from quick analysis of data from calorimeters and muon subdetectors. High Level Trigger (HLT) consists of the Level-2 (LVL2) and the Event Filter (EF). Both are based on software algorithms running on PC farms.

Tracking has a central role in the event selection at the High Level Triggers of ATLAS. The earliest stage where tracking information can be used is the Second Level Trigger, where about 10 ms will be available for event processing. This constraint, together with the high multiplicity environment of ATLAS due to the multiple pp collisions, poses great challenges to the track reconstruction algorithms.

In the scope of this thesis one of the possible approaches to acceleration the tracking algorithms using the hybrid FPGA / CPU systems has been investigated. The TRT LUT-Hough algorithm – one of the tracking algorithms for ATLAS Level2 trigger – is selected for this purpose. It is a Look-Up Table (LUT) based Hough transform algorithm for Transition Radiation Tracker (TRT). The algorithm was created keeping in mind the B-physic's tasks: fast search for low-p_{T} tracks in entire TRT volume. Such a full subdetector scan requires a lot of computational power.

The TRT LUT-Hough algorithm consists of a track candidate search followed by track-fit performed to determine the track parameters. Since all the particle trajectories to search for can be calculated in advance a histogramming method based on the Hough Transform is well suited for the initial track search in the TRT. The algorithm is based on the idea that every hit in the three-dimensional detector image can belong to a number of possi-

ble (predefined) tracks characterized by different parameters. All such tracks (or roads) are stored in the LUT. Thus every hit increases the "probability" for the existence of these tracks (histogramming). The histogram for a single track consists of a "bow-tie" shaped region of bins with entries with a peak at the centre of the region. The bin at the peak of the histogram will, in the ideal case, contain all the hits from the track. The roads corresponding to the other filled bins share straws with the peak bin, and so contain sub-sets of the hits from the track. The histogram for a more complex event consists of a superposition of the entries from the individual tracks. The bins containing the complete set of points from each track can be identified as local maxima in the histogram.

This algorithm is implemented in C++ and integrated into software framework for ATLAS Trigger investigation. Profiling of a C++ implementation of the TRT full scan algorithm shows that the most of the computing time is spent in access of LUT, incrementing of 8-bit numbers, and a local maximum finding. A CPU-only implementation of Histogramming (or Initial Track Finding) and Thresholding / Local Maximum Finding require ∼62% of the total processing time. These two steps are good candidates for an FPGA implementation. The loop over all predefined roads for one straw, which is executed once per hit and increments ∼120 histogram counters, is executed sequentially in the general purpose CPU. This can be done in parallel in the FPGA.

First steps of TRT LUT-Hough algorithm was implemented in VHDL as a proof of concept of the tracking algorithm acceleration with the help of a FPGA based co-processor. The FPGA coprocessor – MPRACE (Multi Purpose Reconfigurable Accelerator / Computing Engine) developed at the University of Mannheim was used as a hardware platform for testing. MPRACE is an FPGA-Coprocessor based on Xilinx VirtexII FPGA and made as a 64 Bit / 66 MHz PCI card. It is a universal co-processor which is used for different applications.

The implementation takes advantage of both the external SRAM on co-processor board and the internal RAM blocks of the FPGA. The LUT is stored in the SRAM and internal block-RAM is used for histogram counters. The FPGA design is synchronised by a 64 MHz clock signal.

Hybrid implementation of the algorithm (when the most time consuming part of algorithm is accelerated by FPGA co-processor and all other parts are running on a general purpose CPU) is integrated in the same software framework as a C++ implementation for comparison. Identical physical results are obtained for both the CPU and the Hybrid implementations. Timing measurements results show that a critical part, is implemented in VHDL runs on the FPGA co-processor ∼4 times faster than on the more or less modern

CPU (Intel Xeon 2.4 GHz) and the whole algorithm runs ∼2 times faster.

For even higher speed-up, another part of the algorithm (the "Track Splitting") should be implemented in the FPGA as well. We expect that the FPGA realisation of "Track Splitting" gives at least the same speed-up as "Histogramming / Maximum Finding". In this case the hybrid implementation will be by a factor of ∼3.2 faster than the CPU-only implementation.

The memory bandwidth is a critical point for algorithm implementation. Design of MPRACE co-processor allows to increase amount of on-board memory by adding memory expansion mezzanine boards. That increases the memory bandwidth by factor two and gives to the algorithm implementation additional speed-up factor close to two.

Additional acceleration can be achieved by using a newer FPGA like Xilinx Virtex4. The VHDL design for initial track finding was slightly modified and synthesised for the Virtex4 family. After the place and route process the estimated clock frequency for the design core is ∼180 MHz. This leads to addition speed-up by factor ∼3 and to total speed-up factor close to 10.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ASIC	Application-Specific Integrated Circuit
ATLAS	A Toroidal LHC Apparatus
CLB	Configurable Logic Block
COTS	commercial "off-the-shelf"
CSC	Cathode Strip Chambers
DAQ	Data Acquisition system
DC	Data Collection subsystem
DSP	Digital Signal Processor
EDM	Event Data Model
EF	Event Filter
FEX	Feature extraction
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
HLT	High Level trigger
HLTSSW	High Level Trigger Selection Software
ID	Inner Detector
LEP	Large Electron Positron Collider
LHC	Large Hadron Collider
LUT	Look-Up Table
LVL1	Level-1 trigger
LVL2	Level-2 trigger

MDT	Monitored Drift Tubes
MPRACE	Multi Purpose Reconfigurable Accelerator/Computing Engine
ROB	Read-Out Buffer
RoI	Region-of-Interest
RoIC	RoI collector
RPC	Resistive Plate Chambers
SCT	Semiconductor Tracker
T/DAQ	Trigger and Data Acquisition system
t2Ref	ATLAS Level 2 Reference software
TES	Transient Event Store
TGC	Thin Gap Chambers
TRT	Transition Radiation Tracker
TRT LUT-Hough	A look-up table based Hough transform algorithm
ZBT SRAM	Zero-Bus-Turnaround SRAM

INTRODUCTION

Last hundred years physicists make their investigations in the world of elementary particles. Particle and High Energy physics is the frontier field which investigates the basic structure of matter, elementary particles and their interactions. It is one of the most interesting and dynamic branches of the modern science. Its ultimate aim is to find a complete description of elementary constituents of matter and of forces acting between them. The description should be as simple as possible.

Through a combination of theory and experiment, a mathematical model that describes or explains all particle physics observed so far by physicists has been worked out. This model is called the Standard Model[1]. It is based on the relativistic quantum gauge field theory and consists of elementary particles grouped into two classes: bosons (particles that transmit forces) and fermions (particles that make up matter). The bosons have an integer spin (0, 1, 2, ...). The fermions have an odd half-integer spin (like 1/2, 3/2, ...). There are two types of fermions: leptons and quarks. Latest, according to what is currently believed (but not yet rigorously proved), is permanently bound inside hadrons ([1], [2]).

However the Standard Model leaves many unsolved questions. Among them, is the reason why elementary particles have a mass and why their masses are different. It is remarkable that such a familiar concept is so poorly understood. The theoretical answer for this question was done by Peter W. Higgs in [3], [4], [5]. According to this, the whole of space is filled with the "Higgs field", and by interacting with this field, particles acquire their masses. Particles, which interact strongly with the Higgs field are heavy, whilst those which interact weakly are light. The Higgs field has at least one new particle associated with it, the Higgs boson. This very important for the Standard Model particle is not found experimentally until now. Without experimental prove of the Higgs boson existence whole Standard Model can not be considered as proved. There are theoretical limits for mass of the neutral Higgs boson. A low limit was made by experiments on the LEP (Large Electron Positron Collider) in the CERN and equal $M_H > 114.4$ GeV at 95% confidence level [6]. An upper limit can be obtained from analysis of experimental

measured parameters of the Standard Model (so called "electroweak fit") and it is unlikely to be heavier than 260 GeV [7], or up to 1 TeV from the theory. Looking for particles in this energy range is a task for recent and future experiments..

The Standard Model is a great theory, but physicists believe that it can not be the final "Theory of Everything". It can be (in the best case) only a low energy approximation of the more general theory. Nowadays there are several such theories. A very popular one is called supersymmetry or SUSY for short. SUSY predicts that for each known particle there is a "supersymmetric" partner. Looking for these partners and some others things of the more general theory is another tasks for future experiments in the high energy physics.

To perform the experimental research physicists rely on particle colliders, which employ technologies from the forefront of engineering developments. Particle accelerators today achieve energies up to 1 TeV. Three different types of collider can be distinguished: hadron-hadron machines, lepton-lepton machines and hadron-lepton machines. One of the most important collider parameter is a luminosity. It is a measure of sensitivity and gives the number of events per second for a cross section of $1cm^2$. Several accelerators and their parameters are listed in Table 0.1 (information from Particle Data Group [8]).

One of future colliders the Large Hadron Collider (LHC) will start operation at the European Organisation for High Energy Physics (CERN) in 2007 [9]. The LHC is a particle accelerator which will probe deeper into matter than ever before. When turned on in 2007, the Large Hadron Collider will be the worlds biggest and the most powerful particle accelerator. The machine will send two beams of protons in opposite directions around a 27-km tunnel at close to the speed of light. Proton beams with energies around 7-on-7 TeV and beam crossing points of unsurpassed brightness will be collided, providing the experiments with high interaction rates. It can also collide beams of heavy ions such as lead ions with total collision energy in excess of 1,250 TeV. When the beams are smashed together, showers of new particles and a possible glimpse at what the universe looked like in its first few moments will be created for physicists to study.

Five experiments, with huge detectors, will study what happens when LHC's beams collide. This "mighty fives" are: ALICE [10], ATLAS [11], CMS [12], LHCb [13] and TOTEM [14]. Three of them are specialized experiments:

- TOTEM – Total Cross Section, Elastic Scattering and Diffraction Dissociation at the LHC.
- LHCb – experiment for precise measurements of the CP violation and rare decays.

Tab. 0.1: Modern and future (LHC) colliders and their parameters.

Collider	**Institute**	**Particles**	**Beam energy** (GeV)	**Luminosity** $(10^{30} cm^{-2} s^{-1})$	**Time between collisions** (ns)
KEK B	KEK	$e^- e^+$	8.0 + 3.5	11305	8
PEP II	SLAC	$e^- e^+$	9.0 + 3.1	6777	4.2
LEP	CERN	$e^- e^+$	101	24	2200
HERA	DESY	ep	30 + 920	75	96
RHIC	BNL	pp	100	6	213
Tevatron	Fermilab	$p\bar{p}$	980	50	396
LHC	CERN	pp	7000	10^4	25

- ALICE – dedicated heavy-ion detector to exploit the unique physics potential of nucleus-nucleus interactions at LHC energies.

Two others, ATLAS and CMS, are general purpose experiments with very wide physics program which included (but not limited):

- searching the Higgs bosons ([15], [16]),
- B-Physics studies ([17], [18]),
- measurements of top quark properties ([19], [20]),
- some studies "Beyond Standard Model" like supersymmetry ([21], [22]),
- a search for extra dimensions ([23] [24] [25]),

and other interesting investigations (for example [26], [27], [28], [29]).

Detectors are built like the Russian dolls, with one specialized layer fitting snugly inside the next. Collisions happen right in the middle, and each layer measures different properties of the emerging particles. The experimental environment at the LHC is characterized by a high event rate and a large background. Detectors will produce huge amount of data, which can not be fully recorded for a future offline analysis. The task of the trigger system is to select rare events and to suppress background events as efficiently as possible. This requires at least a partial analysis of the event, which has to be done before the data is recorded, without disturbing further event detection.

Triggering is one of the greatest challenges at hadron collider experiments. At the LHC beams will be colliding every 25 ns. The pp interaction leading to the interesting physics process (referred to as the physics event) will be accompanied by several minimum bias interactions (~ 20 at the LHC design luminosity) producing extremely high multiplicities. Rare events must be selected from complex raw data, which may contain a million times more events.

One of the most demanding tasks is usually a track reconstruction from measured points (detector hits). Various track finding methods are used in high energy physics tracking applications. Tracking algorithms can be divided into two major classes: local and global.

Local methods select one track candidate at a time by starting with a few points (the track initialization). An algorithm makes predictions as to the further points belonging to this track candidate and tries to find other points that are appropriate to spatially extend the initial tracklet by following the rules that define a valid particle track (the track model). If more of them exist continuously, a track candidate is created; if not, the tracklet is discarded. Typical members of this group are:

The Track Following method, which tries to extend a track "seed" (usually two or three hits in a low occupancy region) by including hits that show the best correlation with the track model and the orientation of the seed. That method always looks only at the next few points, using most recently found ones to extrapolate the track.

The Kalman filter can be viewed as a statistically optimal refinement of track following. The state of a charged track at any given surface in the detector can be described by five parameters: two for the position, two for the direction, and one for the curvature (or momentum). A collection of this parameters is called "state vector". The Kalman filter consists of two steps. In the prediction step the current state vector is extrapolated to the next detector surface, taking into account multiple scattering and energy loss. In the filter step the extrapolated state vector is updated by taking a weighted mean with measurement.

The Track Road method, which starts from both ends of a possible particle track. Additional hits are included, if they meet a road of a certain width around the track candidate, which is compatible with the track model and the initial hits.

Track Segment methods are an extension of the track following method. Elements that are added to the initial seed are also clusters of few hits and likewise have an orientation.

A disadvantage of all local methods consists in that most of the detector hits must be touched multiple times, many of them for each attempt to complete a particle track. Moreover, with increasing hit multiplicity, the computing time increases more rapidly than linearly.

The global methods do not compute single points or groups of points individually. Instead, all points undergo the same treatment, whose first step often is a parameter transformation. This transformation produces a list of track candidates, or at least a structure, in which tracks can be found more easily than in the original detector image. Important global methods are:

The combinatorial method. All combinations of hits are grouped into all possible particle tracks and tested against the track model. The track is accepted, if it is rated as good according to criteria defined in the model. This method can only be applied to detector images with only a few hits, because the computing effort grows extremely fast with the detector occupancy.

Template matching. This method requires a dictionary of all possible classes (tracks). It compares a detector image to patterns from the dictionary. Each match indicates existence of the respective track. This approach is only viable, if the number of possible particle tracks is sufficiently low to handle the pattern set. The method does not work verry well in case of noisy data, and especially data with missing detector hits.

The histogramming method. It is an improvement over the pattern matching approach. In this case, one defines a set of n different functions of the point coordinates and enters the function values in a histogram of n dimensions (one dimension for each function). An n-dimensional histogram has to be visualized as an n-dimensional array of n-dimensional cells. Each cell contains a counter which is increased by a given weight when the coordinates of the measurement define a point inside that cell. Local maxima in the histogram denote the particle tracks, which are contained in the detector image with the highest probability. The method is robust against a certain level of noise and missing data.

In histograms of several dimensions, recognizing the track clusters (local maxima) turns out to be more difficult than finding the tracks directly via a track model. This limits the number of dimensions (n) to one or two.

The histogramming method is a special case of a general algorithm which is called the Hough Transform [30, 31]. The histogram represents a discrete parameterization of all possible particle tracks and can be filled by transformation parameters of the detector image.

Algorithms utilizing an Artificial Neural Network. A typical neural network of the "feed-forward" type consists of an appropriate number of nodes arranged in layers: an input layer, mostly one or two "hidden" layers and an output layer, which for classification problems can be represented by a single node. The nodes of each layer feed their outputs to one or more nodes of a subsequent layer. A node's output is determined only by the sum of its weighed input values minus a threshold value. The network is trained with real or realistic data, i.e. the weights and thresholds are adjusted to produce the correct classification for the set of input patterns. After the training the network is capable to classify similar input patterns.

Purely global methods are independent of the order in which points enter the algorithm. Local methods are not, since the treatment of each point depends

on the "track finding history". A comprehensive introduction to tracking algorithms, trigger and data acquisition systems is given in [32].

Tracking in trigger systems should be done efficiently and fast. To be fast one can use large commercial "off-the-shelf" (COTS) processor farms to run in parallel. Another approach is to use hardware co-processors for most time consuming parts of algorithms which are inherently parallel. Suitable hardware may be based on DSPs (DSP - Digital Signal Processor), that feature dedicated functional units for certain operations like FFT or matrix operations (see, for example, [33]). Reconfigurable Field Programmable Gate Arrays (FPGA) can be programmed to represent any digital circuit that is allowed by the amount of existing logic units.

The present thesis will describes the studies of using the FPGA based co-processor for improving the execution speed of the pattern recognition algorithms from ATLAS High Level Trigger. Improving of physics performance of this algorithm will be discussed as well.

The first chapter introduces the ATLAS detector and its Trigger and Data Acquisition system.

Chapter 2 discusses various approaches to the trigger systems in currently running and planed high energy physics experiments

Chapter 3 focuses on the presentation of the ATLAS software infrastructures for trigger studies.

Chapter 4 gives a short description of the FPGA co-processor board developed at the University of Mannheim and used in the scope of this thesis.

Chapter 5 shows execution time measurements results for some of the inner detector track reconstruction algorithms which are common to all B-physics channels and the standard RoI processing and investigates the feasibility of using a FPGA co-processor for improving their speed.

A TRT LUT-Hough tracking algorithm designed with keeping in mind the B-physic's tasks (search for low-p_{T} tracks in the entire Transition Radiation Tracker volume) and the possibility of further use of FPGA co-processor is described in the chapter 6. Its VHDL implementation for hybrid CPU/FPGA system (PC with MPRACE co-processor), performance comparison with software, C++ implementation, and some improvements for better physics performance are presented as well.

A brief description of the future work for the ATLAS High Level trigger is done in chapter 7.

Throughout this thesis the following conventions are used: the coordinate system has its origin at the interaction point. The z-axis is parallel to the beam, and Cartesian coordinates x, y, or polar coordinates (the radius r and the azimuthal angle ϕ) are used to denote positions in the transverse plane. Instead of the polar angle θ, the pseudorapidity $\eta = -\ln\tan\theta/2$ is

used. Masses, momenta and energies are expressed in natural units where $\hbar = c = 1$.

Chapter 1

ATLAS EXPERIMENT

The possibility of the acceleration of tracking algorithms used in the experimental high energy physics with a FPGA-based co-processor help is considered in the scope of this thesis. One of the trigger algorithms for the ATLAS experiment is used for this task. Therefore, description of the ATLAS experiment (detector and its trigger and data acquisition system) is done in this chapter.

1.1 ATLAS Detector

Accelerators as the LHC can be seen as the "probe" for observing the world of elementary particles. The corresponding "eye" is then the detector, where the incident particles lead to observables effects ([34]).

ATLAS (A Toroidal LHC Apparatus) is a multi-purpose 4π detector (coverage up to $|\eta| = 5$) designed to operate at the LHC design luminosity of $10^{34}\,cm^{-2}s^{-1}$ (first years – $10^{33}\,cm^{-2}s^{-1}$). ATLAS has been designed as a multipurpose experiment, to be capable both detecting and measuring new physical phenomena predicted by currently available theories (see [35]) and performing precision Standard Model measurements. At the same time it must also be open to unexpected signals from unpredicted physics scenarios and thus has to be sensitive to any kind of event topology.

To meet the physics goals, the ATLAS experiment is designed to measure all products of the collision that have a sufficiently long lifetime to be detected. The detector is composed of a number of specialized subdetectors as shown in the Figure 1.1 [1].

The global layout is largely determined by the configuration of the magnet system used for the momentum measurements. The Magnet System [36] consists of two independent parts. A superconducting solenoid surrounds the inner tracker. On the outside of this is the calorimeter which is itself sur-

[1] All figures in this chapter are taken from various ATLAS TDRs

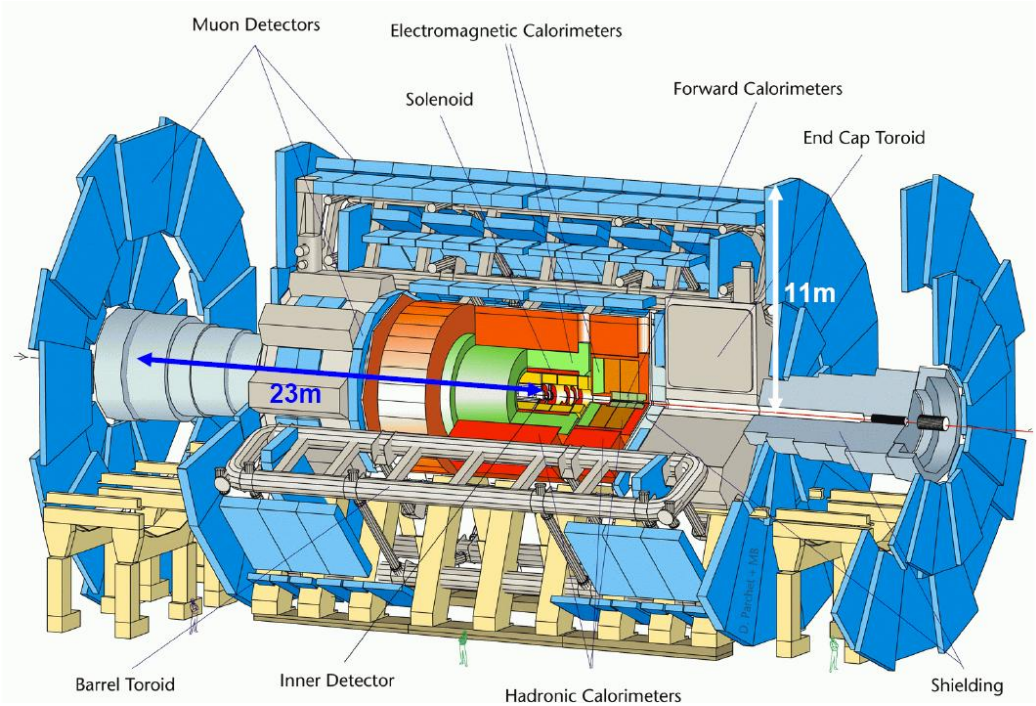

Fig. 1.1: The ATLAS Detector

rounded by superconducting air-core toroids consisting of independent coils with eight-fold ϕ-symmetry.

This magnet configuration allows to build a high-resolution, robust standalone Muon Spectrometer [37] with minimal constraints on the Calorimeter and the Inner Detector. The air-core toroid system, with a long barrel and two inserted end-cap magnets, generates a large field volume and strong bending power with a light and open structure. Multiple scattering effects are therefore minimal, and an excellent muon momentum resolution is achieved with three stations of high-precision tracking chambers. Four different technologies, namely Monitored Drift Tubes (MDT), Cathode Strip Chambers (CSC), Resistive Plate Chambers (RPC) and Thin Gap Chambers (TGC), are used in the muon system, depending on the expected rates and the required space and time resolutions for each chamber.

The Calorimeter System [38, 39, 40] is located inside the muon spectrometer and consists of an inner electro-magnetic calorimeter and an outer hadron calorimeter. The first will absorb electrons and photons, while the latter will absorb hadrons, decay products of taus and most other particles except neutrinos and muons. The calorimeter specific aim is to reconstruct energies and directions of electrons, photons (electro-magnetic calorimeter), and jets

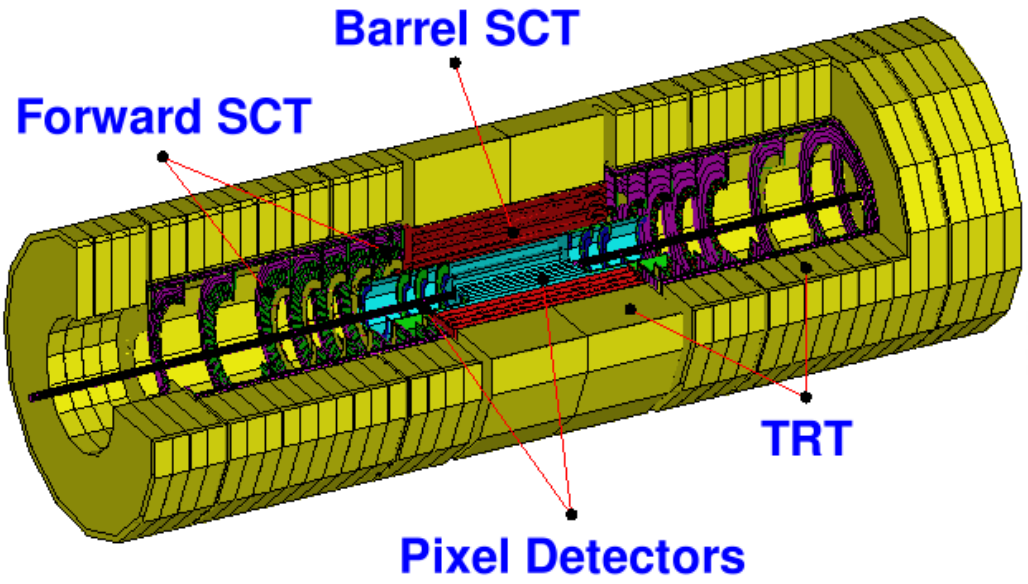

Fig. 1.2: The ATLAS Inner Detector

(electro-magnetic and hadronic calorimeters) as well as to measure missing transverse energy. The electro-magnetic calorimeter and the end-cap regions of the hadronic calorimeter use a liquid ionization technique. Their passive parts consist of lead (or copper) absorber plates with liquid argon filled gaps in between. The design of the barrel (and extended barrel) of the hadronic calorimeter is based on a scintillator technique. This part of the calorimeter consists of scintillating tiles and steel absorber material.

1.1.1 Inner Detector

Inner Detector tracking algorithms will be discussed in later chapters of this thesis. Therefore the Inner Detector is described more detailed.

The Inner Detector [41] is shown in the Figure 1.2. It is a cylinder of $1.15\,m$ radius and $6.8\,m$ in length centered around the interaction point. The purpose of the Inner Detector is to make high-precision measurements of the kinematical parameters of charged particles moving in a solenoidal magnetic field with maximal capability for pattern recognition, particle identification and triggering. High-precision measurements in the environment with very large track density can be made with semiconductor tracking detectors, using silicon microstrip (SCT) and pixel technologies. The highest granularity is achieved around the vertex region using semiconductor pixel detectors. The

total number of precision layers must be limited because of the material they introduce, and because of their high cost. Typically, three pixel layers and eight strip layers (four space points) are crossed by each track. A large number of tracking points is produced by the straw tube tracker (TRT), which provides the continuous track-following with much less material per point and a lower cost. The combination of the two techniques gives very robust pattern recognition and high precision in both ϕ and z coordinates.

Pixel Detectors

The pixel detector [42] is designed to provide a very high-granularity, high-precision set of measurements as close to the interaction point as possible. The system provides three precision measurements over the full acceptance, and mostly determines the impact parameter resolution and the ability of the Inner Detector to find short-lived particles such as B hadrons and τ leptons. The two-dimensional segmentation of the sensors gives space points without any of the ambiguities associated with crossed strip geometries, but requires the use of advanced electronic techniques and interconnections for the read-out.

The system consists of three barrels at average radii of $\sim 4\,cm$, $10\,cm$, and $13\,cm$, and five disks on each side, between radii of 11 and $20\,cm$, which complete the angular coverage. The system is designed to be highly modular, containing approximately 1500 barrel modules and 700 disk modules. The pixel modules are designed to be identical in the barrel and the disks. The individual sensitive element in the barrel (end-cap) is a pixel $50\,mm$ in $R-\phi$ and $300\,mm$ in $z(R)$. The modules are overlapped on the support structure in order to give hermetic coverage.

Semiconductor Tracker

The Semiconductor Tracker (SCT) system [41] is designed to provide eight precision measurements per track in the intermediate radial range, contributing to the measurement of momentum, impact parameter and vertex position, as well as providing good pattern recognition by the use of high granularity. The system is an order of magnitude larger in surface area than previous generations of silicon microstrip detectors, and in addition must face radiation levels which will alter the fundamental characteristics of the silicon wafers themselves.

The barrel SCT uses eight layers of silicon microstrip detectors to provide precision points in the R,ϕ and z coordinates, using small angle stereo to obtain a z measurement. Each detector is $6.36 \times 6.40\,cm^2$ silicon wafer with

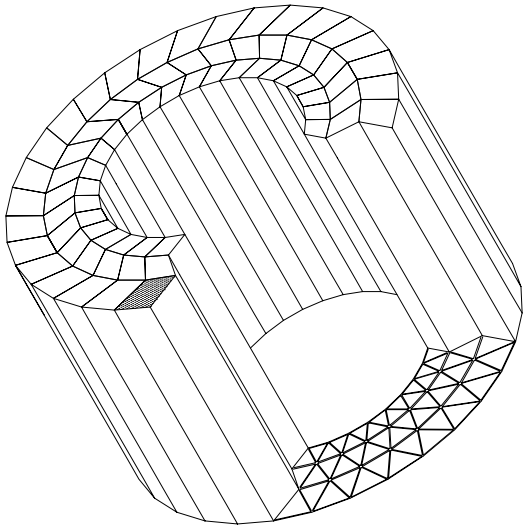

Fig. 1.3: Layout of the barrel TRT

768 readout strips of $80\,\mu m$ pitch. Each module consists of four singlesided p-on-n silicon detectors. On each side of the module, two detectors are wire-bonded together to form $12.8\,cm$ long strips. Two such detector pairs are then glued together back-to-back at a $40\,mrad$ angle to give a stereo measurement, separated by a heat transport plate, and the electronics is mounted above the detectors on a hybrid. The end-cap modules are very similar in construction but use tapered strips, with one set aligned radially.

The detector contains $61\,m^2$ of silicon detectors, with 6.2 million readout channels. The spatial resolution is $16\,\mu m$ in $R-\phi$ and $580\,\mu m$ in z, per module containing one $R-\phi$ and one stereo measurement. Tracks can be distinguished if separated by more than $\sim 200\,\mu m$.

Both the pixel and the SCT systems require a very high mechanical stability, cold operation of the detectors, and the removal of the heat generated by the electronics and the detector leakage current. These two systems together are called "precision tracker".

Transition Radiation Tracker

The Transition Radiation Tracker (TRT) [41] is a set of straw detectors, which can operate at the very high rates because of their small diameter and the isolation of the sense wires within individual gas volumes. This technique is intrinsically radiation hard, and allows a large number of measurements (typically 36) to be made on every track. Electron identification capability is added by employing Xenon gas to detect transition-radiation photons created

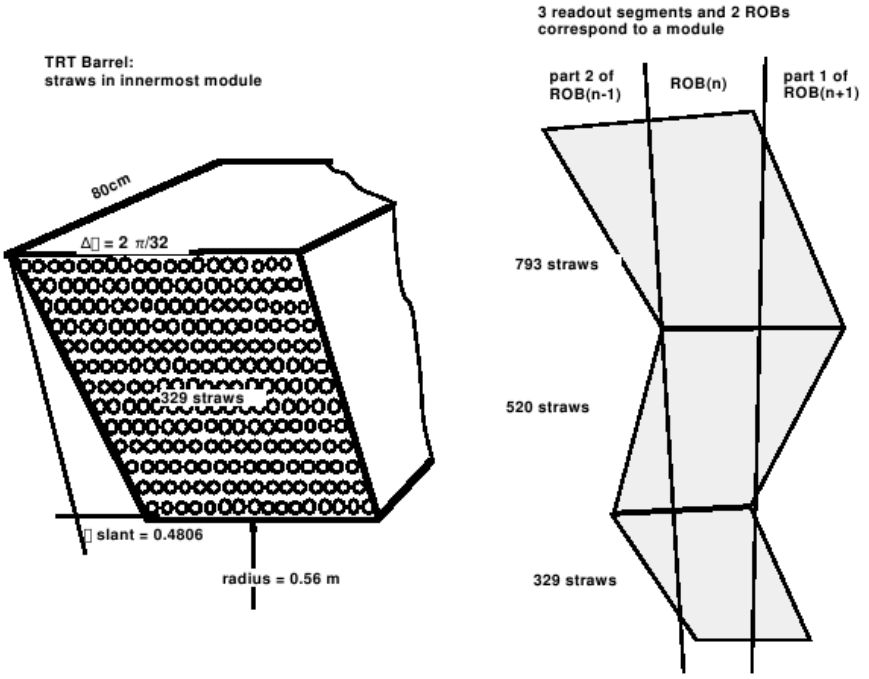

Fig. 1.4: TRT barrel modules (mechanical layout)

in a radiator between the straws. There are two thresholds on the TRT front-end electronics. Hits from transition-radiation photons are identified as signals over the higher threshold.

The TRT consists of a central barrel part and two end-cap sections. The barrel TRT is built from individual modules with between 329 and 793 axial straws each. These modules arranged in three radial sections, each with 32 modules as shown in the Figure 1.3, covering the radial range from 56 to 107 cm. Each section contains straws arranged in planar layers of roughly constant r, with a total of 73 layers, at a radial distance of roughly 6.8 mm [43]. Layers are grouped into three types of modules (see Figure 1.4). The first six radial layers are inactive over the central 80 cm of their length, in order to reduce their occupancy, while providing extra coverage of the crack between the barrel and end-cap sections.

On first stages of the ATLAS simulation (up to so called DC1 stage) a different geometry for the TRT barrel was used. Straws were arranged in cocentric cylindrical layers. Layers were separated by 6.8 mm in R and within a layer straws were separated in $R - \phi$ by 6.8 mm as shown in the Figure 1.5 [41].

Those two end-caps each consist of 18 wheels, each containing 8 or 16

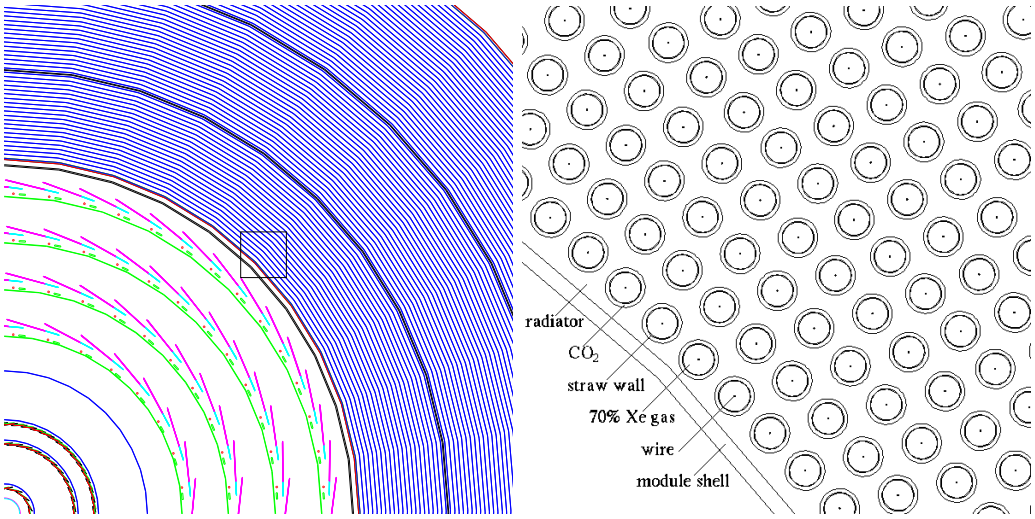

Fig. 1.5: TRT barrel geometry for simulation

planes equidistant along z. Wheels are mounted non-equidistant along z. The fourteen wheels nearest to the interaction point cover the radial range from 64 to 103 cm, while the last four wheels extend to an inner radius of 48 cm in order to maintain a constant number of crossed straws over the full acceptance. To avoid unnecessary increase in the number of crossed straws and material at medium rapidity, wheels from seventh to fourteenth have a half as many straws per cm in z as the other wheels. Detailed geometry description can be found in [41, 42, 43, 44]

Magnetic field

The magnetic field in the inner detector cavity is produced by a superconducting solenoid. The solenoid yields a $2T$ field parallel to the beam axis. The length of the solenoid is less than the length of the inner detector tracking volume and deviations from a homogeneous field are large. In the Figure 1.6 field maps for the longitudinal (B_z) and radial (B_R) components of the field are shown. Such magnetic field configuration leads to deviation of track trajectory from a perfect helix [45] and degradation in the p_{T} resolution (as consequence of reduced bending power). Therefore, simulation and track reconstruction algorithms should use a realistic field map to reduce inaccuracy in the reconstruction.

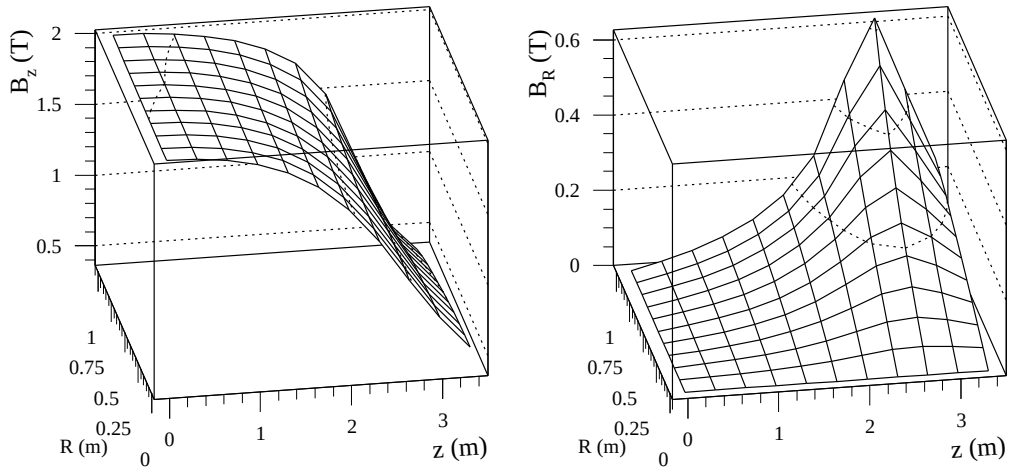

Fig. 1.6: Longitudinal and radial components of the magnetic field as a function of R and z

1.2 Trigger and Data Acquisition System

The pp interaction that leads to the interesting physics process (referred to as the physics event) will be accompanied by several (∼20) minimum bias interactions occurring simultaneously. ATLAS is a very complex system with $\sim 150\times10^6$ channels which will produce approximately 1 MByte data for each event. With a LHC event rate of 40 MHz ATLAS will generate ∼40 TByte of data per second which has to be handled by the data acquisition system (DAQ). Current data storage technology limits the amount of data that can be permanently stored to the order of 100 MBytes/s. The trigger system is designed to bridge this gap. The task of the trigger system is to select rare interesting events, to suppress background events as efficiently as possible. It is designed to bring the event rate to the order of 100 Hz. The task of the DAQ is to move the data produced by the ATLAS detector for selected physics events to the permanent storage for the later analysis.

The ATLAS trigger strategy foresees a reduction of the event rate at several levels: Level-1 (LVL1) and High Level trigger (HLT) which in turn is subdivided into Level-2 (LVL2) and Event Filter (EF) [46]. A schematic view of the ATLAS trigger is shown in the Figure 1.7.

The hardware-based first level (LVL1) makes decision from quick analysis of data from calorimeters and muon subdetectors. The selection is based on reduced granularity data from a subset of detectors. The muon trigger uses only trigger chambers to identify high-p_{T} muons. The calorimeter trigger uses

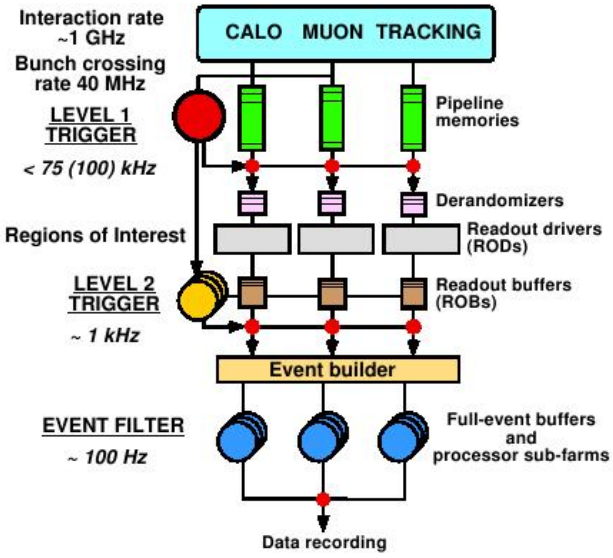

Fig. 1.7: Block diagram of the ATLAS Trigger/DAQ system.

all of the ATLAS calorimeters, but with reduced granularity. The calorimeter trigger searches for electromagnetic clusters, jets and large missing in the total transverse energy. Electromagnetic clusters are characteristic of high-p_{T} electrons and photons, and taus decaying to hadrons. The data from these subdetectors are passed to the processors of the muon and calorimeter triggers. These are custom hardware processors, ASICs (Application-Specific Integrated Circuit) or FPGAs based, with a fixed set of algorithms with programmable parameters.

When the LVL1 trigger accepts an event, the data are moved from the pipeline memories on the detector and stored in read-out buffers (ROBs) during the LVL2 processing and event filter collection time.

To reduce the amount of data requested to a few percent of the full event size only a part of event data from so called Regions-of-Interest (RoI – regions of the detector where the LVL1 Trigger found some activity which lead to accepting an event) is analyzed at Level-2.

The second level trigger uses full granularity, full precision data from the calorimeter and muon detectors but also from the tracking detectors. There are about 1.6 RoIs per event in average [46] (event with three RoIs shown in the Figure 1.8). For each RoI the features from all subdetectors are combined to provide better particle identification and more precise measurements than were possible at LVL1.

The important feature of the HLT event selection strategy is a sequential signature validation. Processing is performed in steps of feature extrac-

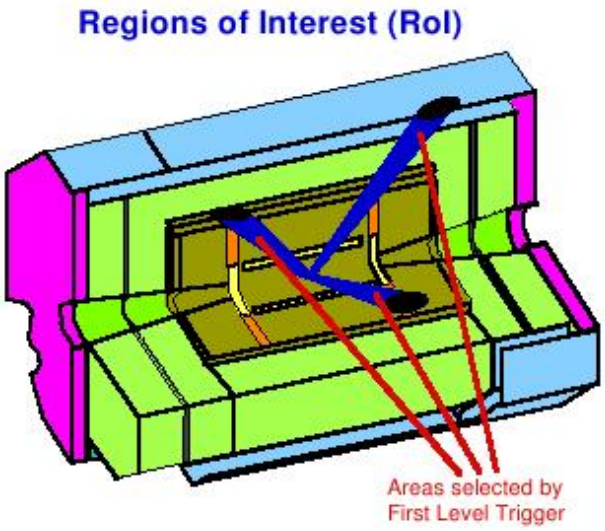

Fig. 1.8: Region-of-Interest (RoI).

tion (FEX) and hypothesis testing algorithms. Events should be rejected at the earliest possible step. The HLT event decision is derived in a series of steps.

After each step when all necessary algorithms have been executed, an event is checked whether it fulfils a signature from a list of signatures (so called a "menu table") defined for this step. In case that at least one signature of this list is fulfilled the event is accepted in this step and it is passed to the next step. Otherwise the event processing is stopped and the processor is free to accept a next event. The definition of signatures for the each step is one part of the configuration. For this reason the configuration provides a hierarchy of menu tables. There is a menu table for each step of event processing. The menu table of a given step contains a list of trigger signatures of which one has to be fulfilled in order the event to be accepted in that step. Connections between signatures of different steps are called "trigger chains".

Algorithm tables are the second part of the configuration. Event characteristics (so called "trigger elements") are refined in the seeded and stepwise event processing by demanding information from more and more subdetectors and combining it. This refinement usually starts from a LVL1 RoI, which identify a certain active region in the detector, and defined by so called "sequences", which specify the trigger elements being required as input, refinement algorithms to be run and a trigger element to be created in case if the algorithm completes it's work successfully. Thus, trigger elements at the step N are transformed to trigger elements at the step N+1 by means of

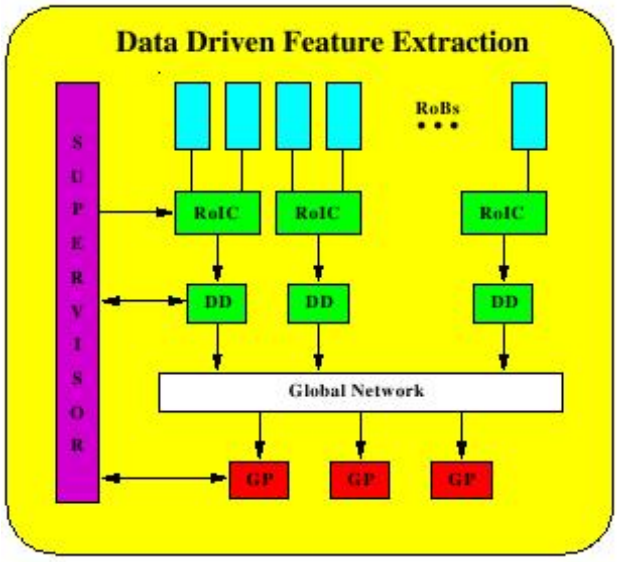

Fig. 1.9: Level2 Trigger Architecture A.

sequences.

The actual processing of raw detector data is restricted to be done by the algorithms. A specific component for the control and configuration is called "Steering". Algorithms can communicate with the Steering only via the creation or non-creation of trigger elements. The Steering in turn decides which algorithm runs on which trigger element according to the existence of trigger elements and fulfilled signatures at the step before. All sequences of a certain step are collected in the "algorithm table" of that step. The configuration has to provide a hierarchy of algorithm tables: for each step there is one algorithm table.

For the low luminosity phase, the LVL2 trigger will have to process B-physics event candidates as well as the high-p_{T} ones. Processing B-hadron events is different from the standard RoI processing [47]. Hadrons containing b quarks are pairwise produced and decay independently. One of the hadrons might decay semi-leptonically which could be triggered by an inclusive muon trigger. The muon from the decay of a B-particle does not indicate the flight direction of the other B-hadron. Therefore, B-hadron events are triggered by a low-p_{T} single muon at LVL1. This muon is confirmed at LVL2, firstly in the muon detector and then in an association with a track from the Inner Detector (ID). The track search can be initiated from outside using TRT information or from inside using Pixel data. A full-scan is performed to search tracks in the entire volume of the TRT or Pixel Detector respectively above

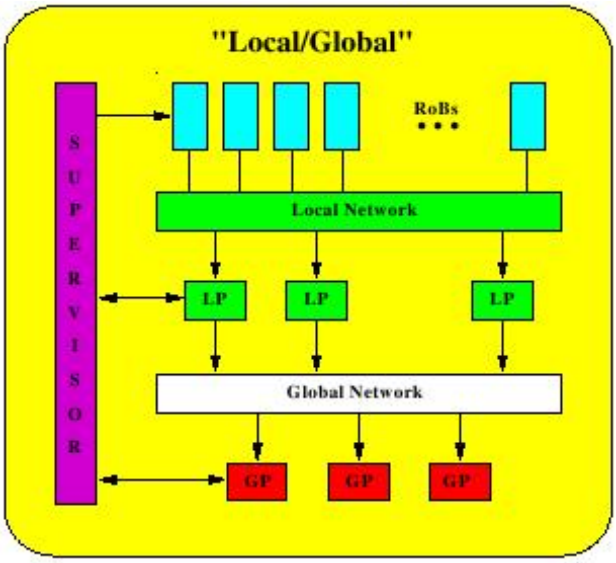

Fig. 1.10: Level2 Trigger Architecture B.

very low p_{T} thresholds ($\sim 1\,GeV$). The result of the full-scan is a set of tracks which are used as track seeds for an algorithm in the semiconductor tracker (SCT).

The LVL2 trigger has several interfaces to external units, such as the LVL1 trigger, the read-out buffers (ROBs), the DAQ system and several functional units: the RoI builder and supervisor, the preprocessing units, RoI collectors (RoIC), feature extraction (FEX) and global decision systems. Every of those components can be implemented by several ways. Three various architectural implementations of the Level2 trigger system have been investigated. These implementations referred as Architecture A, B and C (see Figures 1.9, 1.10, 1.11) respectively.

In the architecture A processing is partitioned into local and global sections. The local section regroups all processing procedures up to and including FEX (ROBs, RoI collectors, and data driven FEXs or DD in the figure). The global section collects the FEX outputs and performs the global processing. The local processing is carried out in fast custom data-driven processors (FPGAs) and the global processing in general purpose processors (marked as LP and GP in the figure 1.9 respectively).

The architecture B is similar to A but the local processors (LP) is general purpose processors arranged in several farms so that feature extraction can be performed in parallel in each subdetector for each RoI.

The architecture C is a single farm of general purpose processors performing the full LVL2 processing for an event. Detailed description of LVL2

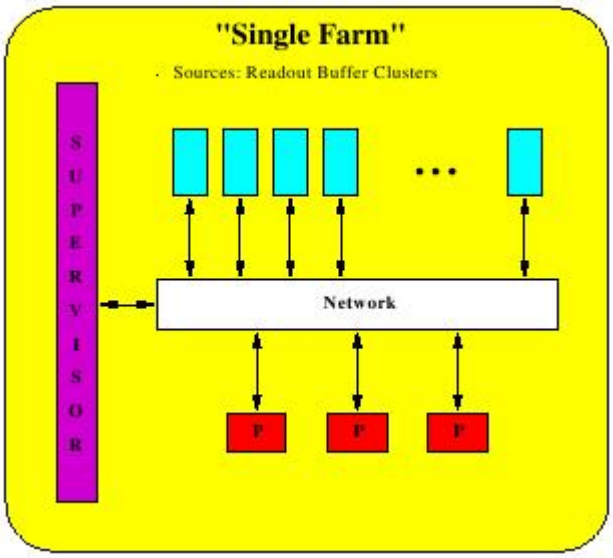

Fig. 1.11: Level2 Trigger Architecture C.

architectures is done in [48, 49, 50]. According to [46], the Architecture C is assumed to represent the final choice for the second level trigger.

The target average decision time for LVL2 should be 10 ms. After the LVL2 trigger accepts an event, the data for that event are transferred to the Event Filter (EF). EF will have access not only to the full event data but also to the geometry and calibration constants. The selection algorithms will require full data decoding and event reconstruction. As an option the reconstruction can be seeded by LVL2 RoIs and the complete event processing might not always be necessary. Offline reconstruction code should be used at the EF stage as far as possible. Because the EF is the first stage of the trigger which has an access to the full data, it will be used for monitoring and calibration studies, such as trigger performance and physics quality. The EF processing will be implemented by a farm of commercial general purpose processors.

Chapter 2

TRIGGER SYSTEMS IN OTHER EXPERIMENTS

Dedicated hardware is widely used in the experimental High Energy physics. Trigger systems from different experiments are described below. At the end of this chapter I will try to analyse the usage of commercial PC farms and custom hardware in these triggers and compare them with the ATLAS approach.

The staged trigger architecture is widespread at various currently operational and planed high energy physics experiments. In most cases the trigger strategy is similar. Only the number of trigger levels and their implementation differs from experiment to experiment.

2.1 Running experiments

The most demanding experiments from currently running are CDF [51] and D0 [52] at the Tevatron (Run IIb), BABAR at the PEPII [53] and Belle at the KEK [54]. BABAR and Belle have similar physic program – the studies of rare B-meson decay modes – and similar trigger systems.

2.1.1 BABAR Trigger System

BABAR, the detector for the SLAC PEP-II asymmetric e^+e^- B Factory operating at the $\Upsilon(4S)$ resonance [53], was designed to allow comprehensive studies of CP-violation in B-meson decays. Tracks of charged particles are measured in a multi-layer silicon vertex tracker surrounded by a cylindrical wire drift chamber. Electromagnetic showers from electrons and photons are detected in an array of CsI crystals located just inside the solenoidal coil of a superconducting magnet. Muons and neutral hadrons are identified by arrays of resistive plate chambers inserted into gaps in the steel flux return of the magnet. Charged hadrons are identified by dE/dx measurements in the

tracking detectors and by a ring-imaging Cherenkov detector surrounding the drift chamber. The trigger, data acquisition and data-monitoring systems, VME- and network-based, are controlled by custom-designed online software.

The trigger is implemented as a two-level hierarchy, the Level 1 in hardware followed by the Level 3 in software.

The Level 1 trigger decision is based on detected charged tracks in the drift chambers above a preset transverse momentum, showers in the electromagnetic calorimeter, and tracks detected in the instrumented flux return (system designed to identify muons and to detect neutral hadrons). Trigger data are processed by three specialized hardware processors (FPGA based). Each of the three Level 1 trigger processors generates trigger primitives, summary data about the position and energy of particles, which are sent to the global trigger.

The current drift chamber trigger uses a 2D track reconstruction which has no z information and thus it cannot distinguish some background events. The BABAR trigger upgrade will provide 3D tracking capabilities at the first trigger level. New $z_0 p_{\mathrm{T}}$ Discriminator (ZPD) boards will reconstruct 3D track information and make trigger decisions based upon the z_0 origin of the tracks [55]. The ZPD Track Finder uses a Hough transform based algorithm which runs on boars with 8 Xilinx Virtex-II FPGAs.

The L3 trigger software comprises event reconstruction and classification, a set of event selection filters, and monitoring. This software runs on the online computer farm - a farm of commercial UNIX processors. The filters have an access to the complete event data for making their decision, including the output of the L1 trigger processors and use more sophisticated algorithms. For example, the drift chambers based algorithm performs fast pattern recognition (track finding) and track fitting, which determines the five helix track parameters for tracks with p_{T} above 250 MeV/c. To speed up the process of pattern recognition, algorithm starts with the track segments from the Level 1 system and improves the resolution by using the actual drift chambers information.

2.1.2 CDF Trigger System

The Collider Detector at Fermilab (CDF) [51] is a general purpose particle detector which has been taking data again (after accelerator and detector upgrade) since 2001. It pursues a broad physics program at Fermilab's Tevatron proton-antiproton collider, comprising topics as diverse as top quark production and charmed meson decay.

CDF uses a three-level trigger system. Level-1 and Level-2 use custom

hardware on a limited subset of the data and Level-3 uses a processor farm running on the full event readout.

On each beam crossing (396 or 132 ns), the data from the entire front-end is digitized (silicon data is sampled and holded). A pipeline of programmable logic forms axial drift chamber tracks and can match these with the calorimeter and muon chamber data. On the Level 1 accept, front-end boards store the event into one of four buffers (silicon data is digitized and transmitted to the silicon trigger and event builder). One of the main different from the previous Level-1 tracking systems is an addition of track finding, which is normally available only at Level-2. This allows a matching the track information with an electromagnetic calorimeter energy cluster for improved electron identification or with a stub in the muon system for the better muon identification and momentum resolution. Also, tracks may be used alone for triggers such as $B^0 \to \pi^+\pi^-$. The Level-1 accept can also be generated based on calorimeter energy, missing $E_{\rm T}$, or the kinematical properties of the observed track pairs.

The Level-1 track finding is performed by the drift chamber hardware track processor, XFT [56]. The track identification is accomplished in two processes by the Finder and the Linker. The Finder searches for high-$p_{\rm T}$ track segments in each of the four axial superlayers of the Central Tracker. The Linker searches for a four-out-of-four match among segments in the four layers, consistent with a prompt high-$p_{\rm T}$ track. If no track is found, the Linker searches for a three-out-of-three match among segments in the innermost three layers.

The Level 2 processing additionally includes calorimeter clustering, electromagnetic calorimeter data and fast silicon tracking. For each event passed through Level 1, the Silicon Vertex Trigger (SVT) [57] swims each XFT track into the silicon detector, associates silicon hit data from four detector planes, and produces a transverse impact parameter measurement. Three key features allow SVT to carry out in a silicon track reconstruction:

- a highly parallel/pipelined architecture;
- custom VLSI pattern recognition (ASIC-based);
- a linear track fit in fast FPGAs;

The final Level 2 decision is done in software on a single-board computer, so a wider range of thresholds and derived quantities are possible (e.g. transverse mass of muon track pairs), even for information that is in principle available at Level 1. On Level 2 accept, front-end VME crates transmit data to the event builder.

At Level 3, a farm of 250 commodity PCs runs full event reconstruction. This is the first stage at which three-dimensional tracks (e.g. for invariant mass calculation) are available. Events passed through Level 3 are stored on the permanent storage media for offline analysis.

2.2 Experiments at LHC

2.2.1 CMS Trigger System

The CMS experiment [12] employs a general-purpose detector with nearly complete solid-angle coverage in the LHC machine.

In order to get an adequate event rejection fulfilling the available time and data bandwidth constraints, the CMS trigger selection is subdivided into several steps (levels), each one takes a decision using only a part of the available data. The CMS trigger consists of a First Level Trigger (L1) implemented using a dedicated programmable hardware and a software High Level Trigger (HLT) system running on a farm of commercial processors [58].

While the First Level Trigger performs a rapid (latency of the order of 3 μs) decision on the basis of information from the calorimeters and muon chambers with limited granularity only, at the HLT step the fine granularity data from all CMS subdetectors can be accessed and analysed. From a logical point of view, the particle identification performed by the HLT can be divided in the following phases:

1. Level 2: only the calorimeter information is used;

2. Level 2.5: partial tracker information, coming from the pixels, is included;

3. Level 3: the complete tracker information from the CPU consuming full charged particle track reconstruction is exploited.

A custom hardware for the real time track finding based on associative memory was considered as well [59] but it has been decided that the CMS High-Level trigger will run on a farm of mass-market processors using a code that is as close as possible to the offline reconstruction code.

In principle, algorithms of arbitrary complexity can be implemented in the CMS HLT environment. The most severe constraint is posed by the available CPU time. Several techniques are employed in CMS HLT to reconstruct high-quality tracks at a minimum computational load [60].

A considerable speed-up is obtained by a regional application of the algorithm. Typically, regions-of-interest are defined on the basis of the result of

the preceding trigger level (a calorimetric cluster, the extrapolation of a track in the muon chambers, etc.) The time performance of the regional approach allows the implementation of the default offline algorithm - the combinatorial Kalman filter based track finder[61] to be used in the later stages of the high level trigger. Thus, the obtained track reconstruction is of comparable quality to the offline reconstruction.

For the earlier stages of the high level trigger, speed becomes even more of a concern. A fast reconstruction based on hit triplets in the pixel detector has been developed. The algorithm is described in detail in reference [62]. Pixel-only reconstruction is sufficiently fast that global reconstruction of all tracks with transverse momentum greater than 1 GeV can be performed in the high level trigger. The simplified pattern recognition has to rely on three hits out of three pixel layers, thus posing a severe requirement on the single layer efficiency. Therefore, the robustness of the algorithm against defective components is much reduced.

2.2.2 LHCb Trigger System

LHCb [13] is designed as a single-arm dipole spectrometer for high precision studies of CP-violation in the B-hadron system. The polar angular coverage, between approximately 10 and 350 mrad, gives a high acceptance for B-hadrons. In the setup still recently foreseen for the LHCb detector, so-called LHCb-Classic, the tracking stations[1] providing space point measurements along particle trajectories were roughly equally spread out from the end of the Vertex Locator (VELO) to the entry window of RICH-2. There were nine such tracking stations: two stations upstream from the magnet (T1, T2), three stations inside the magnet (T3, T4, T5) and finally four stations just downstream of this (T6 to T9).

The trigger system [63] is divided into three distinct levels (Level-0, Level-1, and HLT) which process an increasing amount of information from the subdetectors.

The zeroth level (Level-0) trigger identifies events with particles which have high transverse momenta p_{T} based on signals in the calorimeters and muon system; this being a typical signature of b-decays. This level will have a fixed latency and the processing will be done in hardware residing at the detector. There are four Level-0 subsystems: the calorimeter trigger, the muon trigger, the pile-up trigger, and the decision unit. The latter collects reconstructed information of the other three and makes the Level-0 trigger decision.

The Level-1 trigger is the first of two software triggers running on a

[1] Sets of tracking detectors

commodity CPU farm. Eighteen hundreds CPU's are shared with the High-Level Trigger. The trigger algorithms run in the same software environment as the offline algorithms. In contrast to the fixed latency of the Level-0 trigger, Level-1 has variable latency.

The Level-1 scheme is to search for B decay tracks with a high impact parameter in combination with a large transverse momentum (p_{T}). The p_{T} requirement serves to reject low-momentum particles which have obtained a high impact parameter due to multiple scattering. The Level-1 trigger tries to identify events with a secondary vertex[2]. Doing standalone pattern recognition on the VELO hits, 3D track candidates are constructed. The impact parameter of a track with respect to the reconstructed primary vertex is used to assign a probability that the track originates from a secondary vertex [64, 65].

The higher level triggers use information from all subdetectors and perform partial or full event reconstruction. The HLT algorithms run concurrently on the same CPU nodes as the Level-1 algorithm, but with a lower priority. The data from all subdetectors is available at this trigger level. The track finding and reconstruction method has been based on the Kalman filter techniques, which performs simultaneously pattern recognition and track fit [66]. Information on the particle identity is added to these tracks when they are matched to muon or electron candidates.

2.2.3 ALICE Trigger System

The ALICE [10] detector is designed to study high energy heavy ion collisions with the aim of observing the predicted phase transition from normal hadronic matter to plasma of deconfined quarks and gluons (QGP). It is optimized for heavy-ion reactions and, thus it has a very different design than the other three LHC experiments (ATLAS, CMS, LHCb). The detector has to be able to track and identify particles from very low ($\sim$100 MeV/c) up to fairly high ($\sim$100 GeV/c) transverse momentum, to reconstruct short-lived particles such as hyperons, D and B mesons, and to perform these tasks in an environment of extreme particle density (up to 8000 charged particles per unit of rapidity). The high tracks multiplicity has governed the choice of detectors, and in particular the use of a time-projection chamber as the principal tracking detector.

Tracking relies on a set of high-granularity detectors: an Inner Tracking System (ITS) consisting of six layers of silicon detectors, a large-volume

[2] The secondary vertex is a point of decay of a particle which originated at the some point of a significant finite distance away from the center of impact (primary vertex)

Time-Projection Chamber (TPC), and a high-granularity Transition Radiation Detector (TRD). TPCs have a long drift time, which means that some care should be taken to ensure that the tracking efficiency is not prejudiced by overlapping tracks from events at times close to the triggered event.

Particle identification in the central region is performed by measuring the energy loss in the tracking detectors, the transition radiation in TRD, and the Time-Of-Flight (TOF) with a high-resolution TOF detector. Two smaller single-arm detectors complete particle identification at the mid-rapidity detecting the Cherenkov radiation with a High-Momentum Particle Identification Detector (HMPID), and photons with an electromagnetic calorimeter based on scintillating crystals (the PHOton Spectrometer – PHOS). The detection and identification of muons is performed with a dedicated forward spectrometer.

Additional detectors located at large rapidities complete the detection system to characterize the event and to provide the interaction trigger.

The very high multiplicity leads to very complicated events, and effectively rules out the extraction of detailed trigger signals inside the allowed time budget. The design concepts of the ALICE trigger system [67] are similar to the design concepts of the trigger system in the NA57 experiment [68]. It is foreseen to operate in three different levels – Level 0 (L0), Level 1 (L1) and Level 2 (L2) – implemented in hardware close to detectors front-end electronic. These different levels correspond to criteria imposed from different detectors, where the selection criteria get stronger as the trigger number increases.

At the trigger level no attempt is made to correlate matching solid angles in different detectors. Instead, trigger signals are treated as characterizing events as a whole, and trigger conditions are restricted to Boolean combinations of these. More detailed analysis is left to the High Level Trigger (HLT), which has a much longer time budget.

HLT [67, 69] is a trigger layer logically being located between L2 and the event building. This trigger layer can perform very complex functions and is a logically hierarchical commodity cluster. Although it focuses on the largest data source – TPC, HLT incorporates all relevant detectors. The design of HLT is guided by two principles:

- pattern recognition and data compression are done as local as possible in order to benefit from the natural locality and parallelization in the data,
- event reconstruction combines relevant information from all detectors.

HLT is the only online instance performing event reconstruction. An HLT

online tracking prototype was constructed and tested at the RHIC STAR experiment (where it was called the L3 trigger) [70, 71, 72]. Both the selective trigger and data reduction modes have being working well, however at much lower occupancies.

A farm of clustered SMP-nodes based on off-the-shelf PCs and connected with a high bandwidth low latency network provides the necessary computing power for HLT. The hierarchy of the farm has to be adapted to both the parallelism in the data flow and to the complexity of the pattern recognition. Data from each sub-sector are transferred via an optical fiber from the detector into receiver nodes of the PC farm. Each receiver node is interfaced to the front-end electronics via their internal PCI-bus, using a custom PCI receiver card. These boards provide a FPGA co-processor for data intensive tasks of the pattern recognition.

There are two different approaches to track parameters estimation [73, 74]: the sequential feature extraction and the iterative feature extraction. The sequential method first finds a number of cluster centroids with a "Cluster Finder" and then uses a "Track Follower" on these space points to extract the track parameters. The iterative method first finds a number of track candidates using a suitable defined Circle Hough Transform on the raw data and then assigns clusters to the track candidates. A helix fit on the assigned clusters finally determines the track parameters. Some parts of algorithms can be highly parallelized in hardware and can run directly on the FPGA co-processors while reading out the data.

2.3 Outlook

As it can be seen from the previous and this chapter, ATLAS has more or less traditional two stage (or three if count Level2 and Event Filter as a separate stages) trigger system. The physics tasks of ATLAS close to CMS but differents in detector construction and subdetector sets (for example, there are no gaseous tracking detectors in CMS) determine the differents in the trigger systems. The regions-of-interest approach is used in the CMS as well as in the ATLAS. But in CMS foreseen to use full tracking information only at Level3 and, therefore, more sophisticated (and slow) algorithms (similar to algorithm from the offline analysis code) can be used. Same approach we have in ATLAS at Event Filter step. However, as soon as we want to make some tracking analysis at LVL2 we need simple and fast algorithms as well.

Various hardware processors (ASIC, FPGA, DSP etc.) are widely used for track finding in trigger applications in running high energy physics experiments. However, it is planed to drop custom hardware at Level 1 and use

farms of commercial "off-the-shelf" (COTS) PC at the earliest possible trigger level in future experiments. This approach has a number of advantages compared with custom electronics components:

- Parts of the system can be exchanged against more powerful components without a redesign of electronics. This requires the compatibility of new components which is also a widespread issue in industrial applications.
- The components can be purchased in a short time. Thus recent hardware can be used at the experiment's start-up.
- The components are more cost effective than the custom hardware.
- Offline reconstruction algorithms can be used in the last trigger level with very small modificytions.

ALICE is the only experiment at LHC which has plans to use the custom hardware (FPGA co-processors) in the High Level Trigger but even there it is not standalone custom processors, but co-processors which are used together with "standard", commercial PCs. As it was mentioned, FPGAs for data preprocessing will be installed on data receiver cards. Therefore, information from detectors will be available for trigger farm after processing in FPGAs.

Using systems with hardware co-processors allows to reduce the size of PC farms which are needed for a tracking algorithm execution without degradation of the execution time and with the same quality of results.

In scope of this thesis I have tried to investigate the feasibility of using FPGA co-processor for speeding up of some tracking algorithm from the ATLAS Level-2 trigger (a part of the High Level trigger). Similar to ALICE, so called "hybrid FPGA/CPU" approach is used, when a co-processor works with a standard PC and is used for most time consuming parts of the tracking algorithm. It allows more or less easy integration of the custom hardware into the existing trigger scheme. An object-oriented software design simplifies this task as well. However, in opposite to ALICE, PCs in the ATLAS Level2 processing farm (LVL2 Processing nodes) will get the data from the ethernet network and retransfer it to the FPGA co-processor over PCI bus (if required). Another possibility is to have FPGA co-processor with its own ethernet interface. However, in this case a complicated mechanism for receiving LVL1 accept signals and taking decisions which data should be processed at LVL2 must be implemented in FPGA.

Chapter 3

SOFTWARE FOR TRIGGER STUDIES

Several algorithms and trigger strategies were considered during the design and preparation period of ATLAS. To select the best suited ones trigger testing framework is required.

Long time the ATRIG [75] (ATLAS Trigger Simulation) package was a standard tool for the trigger simulation studies which runs within the framework of the ATLAS offline software. But during last years several other packages has being created to study different aspects of the trigger system. In the scope of this thesis we are interested in the software for second level trigger studies that allows to compare several algorithms or several implementations of the same algorithm. Frameworks which were used during the work on this thesis are described below.

3.1 ATLAS Level-2 Reference Software

The ATLAS Level 2 Reference software (t2Ref) [76] has been developed as a part of the ATLAS Second-Level Trigger Pilot Project. Its goal was to have a single common source code base for the various activities in the Pilot Project. The most important requirements were a modular design, which would allow to modify and optimize the software for different purposes. This includes the network technology studies in the various hardware testbeds as well as the physics algorithms development and benchmarks. Emulators were available for each component in the final system which might be implemented in a special hardware. An emulator can be replaced by the real hardware when that is available.

Several scenarios were foreseen for the reference software:

- Single feature extraction algorithms. A user should be able to write an isolated FEX algorithm and run it in a simple framework over a sequence of events, producing output in any format he or she likes.

The algorithms can be used without change in a full system, if they are written according to some simple rules.

- Single node trigger processing. This mode is a full menu based execution of multiple FEX algorithms in a common framework. The data is read from a file. It is a full functional test of the trigger algorithms and the steering code.

- Multinode system with skeleton applications. This scenario implements the full data flow of the trigger system, but without executing any algorithms.

- Multinode system with algorithms. It is a complete test of available systems.

Only the first scenario is interesting in the scope of this thesis. The Trigger Algorithms in the Reference Software are fulfilling several contrary requirements. They must be fast, efficient, and lead to a high overall rejection factor. Furthermore, they have to fit into the LVL2 Testbed, which has implications on the implementation of the algorithms, e.g. they must be multiple thread save.

3.2 CTrig

For a more convenient LVL2 trigger algorithms development and for measuring their performance another "self-contained", "flexible" and relatively "lightweight" environment was created. The environment was called CTrig [77] and was written by John Baines. CTrig was conceived from the need for a relatively small package for the LVL2 simulation which would be faster to compile and run than the full ATLAS offline software. The aim was to provide a environment where the cycle time for "edit-compile-test" was minimised.

CTrig includes code written in C and C++ and runs independently of the ATLAS frameworks. As an input it uses ASCII data files which can include information from the LVL1 and LVL2 trigger simulation. The ASCII file format was developed in collaboration with the pilot project reference software and benchmarking groups. The algorithms are intended to be self-contained with a minimal and well defined interface. Ideally the same algorithms would be able to run within the ATRIG or CTrig frameworks. It should be relatively simple to reconfigure the main program for different applications. Two different frameworks were provided for:

- electron ID studies within LVL1 calorimeter RoI.

- B physics studies: following a full scan of the TRT, the SCT+Pixel FEX is run in regions guided by the TRT tracks.

The ultimate test of the algorithms performance can, in general, only be made by combining information from more than one system. For example the electron trigger object is formed by combining information from the LVL2 calorimeter, TRT and SCT+Pixel algorithms. CTrig provides the ability to combine the results from these different systems in common ntuples[1]. The results stored in the ntuple for a given system may be either those produced by the FEX in cTrig or those of the ATRIG FEX read from the file or both. This removes the necessity to, for example, run the CTrig SCT+Pixel algorithm when developing and testing the TRT algorithm. It also allows results of ATRIG FEX to be used as a reference to compare with results from an algorithm in cTrig.

3.3 High Level Trigger Selection Software

After years of prototyping and testing the ATLAS High Level Trigger community should provide a design of event selection software which will run in a final system. The goal is to replace and unify the functionality of existing online and offline trigger implementations ATRIG, CTrig and the LVL2 reference software. This involves both the infrastructure or framework and the algorithms. The latter are to be provided either by the PESA group or, in case of the algorithms for the EF, by the offline reconstruction group. Major parts of the online reconstruction will have to be based on offline reconstruction algorithms. This is an important constraint for the design of the new High Level Trigger Selection Software (HLTSSW) [78].

It is foreseen to do the HLT selection in two steps: LVL2 and the EF (see section 1.2). The logical boundary between LVL2 and EF is not precise. Indeed, flexibility in setting the boundary should be retained in order to profit from the complementary features of both trigger steps. The key roles of the HLTSSW are "event selection" and "event classification". Abstract objects representing candidates of e.g. electrons, jets, and muons are reconstructed from the event data by a particular set of HLT algorithms and parameters. An event is selected if the reconstructed objects satisfy at least one physics signature in the *Trigger Menu*. At both stages, LVL2 and EF, events are rejected if they do not pass any of the selection criteria designed to meet the signal efficiency and rate reduction targets of the trigger. The result at

[1] Data structure which is used in experimental high energy physics (see, for example, http://wwwasdoc.web.cern.ch/wwwasdoc/hbook_html3/node23.html).

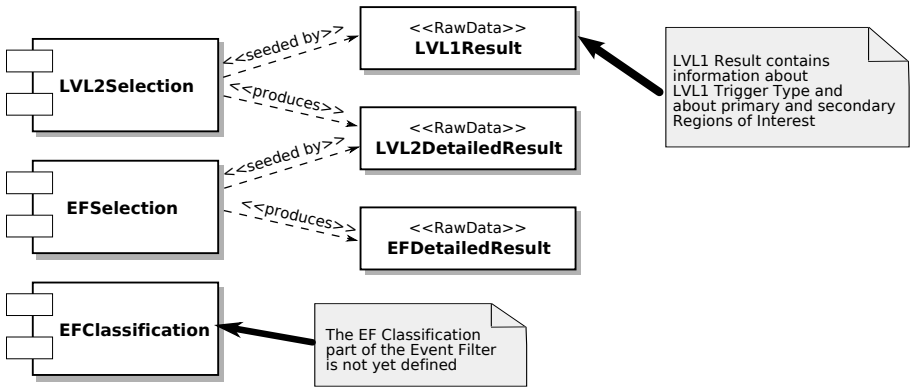

Fig. 3.1: A component diagram of the High Level Trigger selection chain

one trigger level provides seeds for the next level. The EF detailed result can be used to assign tags to the events or even assign them to particular output streams. Event tags may serve to select efficiently interesting events for offline reconstruction and analysis.

The basic structure of the HLT selection chain is shown in the Figure 3.1 in a simplified form. The starting point for the HLT is the *LVL1 Result*. It contains the *LVL1 Trigger Type* and the information about the primary RoIs that caused the LVL1 accept, plus the secondary RoIs that are not considered for the LVL1 accept. Both types of RoIs are used to seed the LVL2 selection. The concept of seeded reconstruction is fundamental to the LVL2, apart from the special case of B-physics.

During data taking the HLTSSW will run in an online software environment provided by the HLT subsystem of ATLAS T/DAQ, with event data coming via the Data Collection (DC) subsystem. Therefore the HLTSSW needs to comply with the online requirements, like thread safety, online system requirements and services, as well as online performance goals.

It is essential though that the HLTSSW is also able to run directly in the offline environment ATHENA [79] to facilitate development of algorithms, to study the boundary between LVL2 and EF and to allow performance studies for physics analyses. Therefore the HLTSSW needs to comply with the control framework and services that are provided by the offline software architecture team.

The package diagram of the HLTSSW is shown in the Figure 3.2. The sub-packages are:

- The *Steering* [80] controls the selection software. It "arranges" the algorithm processing for the event analysis in the correct order, so that the required data is produced and the trigger decision is obtained. The

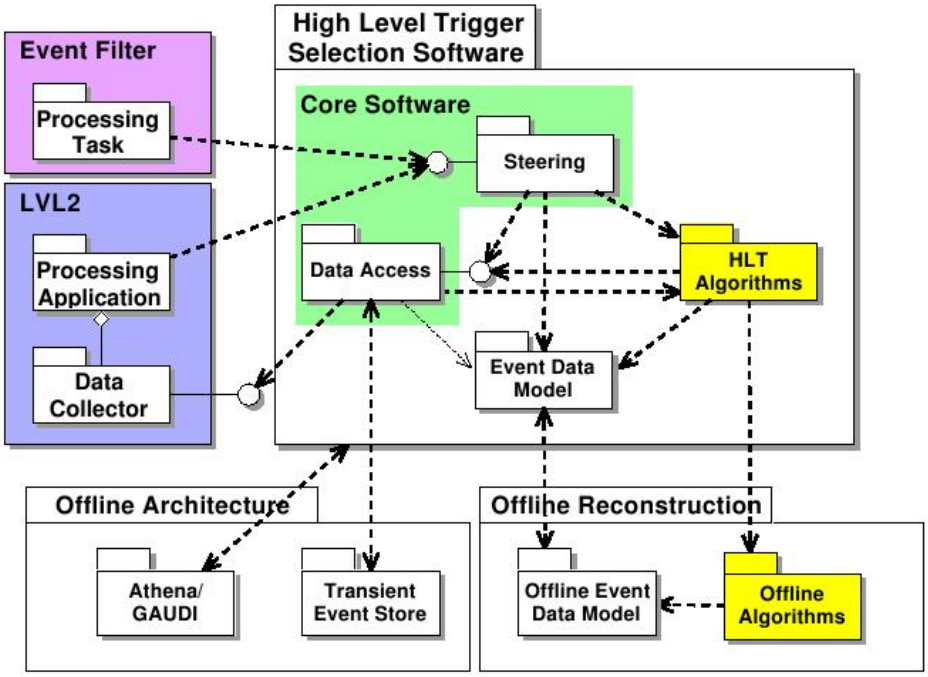

Fig. 3.2: A package diagram of HLTSSW

Steering Controller[81] is the component of the *Steering* that implements the interface to the LVL2 *Processing Unit Application* (when running in the LVL2 *Processing Unit*) and to the *Processing Task* (when running in the *Event Handler*).

- The event data is structured following the *Event Data Model* (EDM). The EDM covers all data entities in the event and their relationships with each other. The data entities span from the *Raw Data* in byte stream format (originating from the detector Read-Out Drivers), the *LVL1 Result* and all other reconstruction entities up to the *LVL2 and EF Detailed Result*.

- The *HLT Algorithms*[82] are used by the *Steering* to process the event and to obtain the data on the basis of which the trigger decision is taken. The dependency on *Offline Algorithms* is especially important for the implementation of the EF. Trigger versions of *Offline Algorithms* are to be adapted to online requirements. Hence, a common definition of the EDM and the *Offline Event Data Model* is desirable in order to facilitate the reuse of *Offline Algorithms*.

- The *Data Access Manager*[83] handles the event data during the trigger

processing. It provides the necessary infrastructure for the EDM. For the LVL2 case the communication with the Read-Out Buffer (ROB) Data Collector (or respectively its emulation) to access the ROB Data fragments is part of the *Data Manager*. Also shown is the dependence of the ROB Data Collector on the Read-Out System Data Preparation.

At Level-2, algorithms actively request portions of event data from the Data Collection System. The relevant data are defined by RoIs based on information from the decision from Level-1 or a previous result in Level-2 processing. For each RoI, the total data volume with respect to the whole detector is roughly a few percent. Hence, this restricted data access strategy represents a significant reduction in the required HLT processing and networking resources.

For a given RoI, typically defined by an extent in η and ϕ within the physical detector volume, a *Region Selector* [84] translates the physical volume into a set of offline identifiers. These identifiers are translated at a subsequent stage into online identifiers which may then be used to request the data themselves.

It may seem counterintuitive to use such a scheme (i.e., conversion into a geometrical region which requires translation into Offline identifiers which then require translation into Online identifiers). There are, however, a variety of motivations for the *Region Selector*. The prime motivation is to gain access in a uniform and rapid way to event data from sub-detectors which do not participate in the Level-1 trigger decision (e.g., event data from Inner Detector tracking information given a Level-1 trigger based on an energy deposit in the Calorimeters). An additional motivation includes allowing for possible secondary RoIs as needed by an algorithm which may lie outside the primary RoI defined by Level-1.

Raw data from the ATLAS detector will be delivered in terms of a ByteStream of data consisting of hierarchically arranged fragments formatted in a sub-detector-dependent way. This ByteStream data must be converted into objects which can then be used by the algorithms. Modelling this flow and conversion of ByteStream data in a realistic way is vital to an accurate modelling of the HLT performance and subsequent estimation of required network and computing resources.

The HLTSSW adopts a scheme whereby the interaction of HLT algorithms with the Data Collection System is hidden behind a call to the *Transient Event Store* (TES). Figure 3.3 illustrates this scheme. An algorithm requests data within a certain region by first feeding the parameters of the region to the *RegionSelector*. The *RegionSelector* returns a set of Offline identifiers which the algorithms then use to request collections of relevant

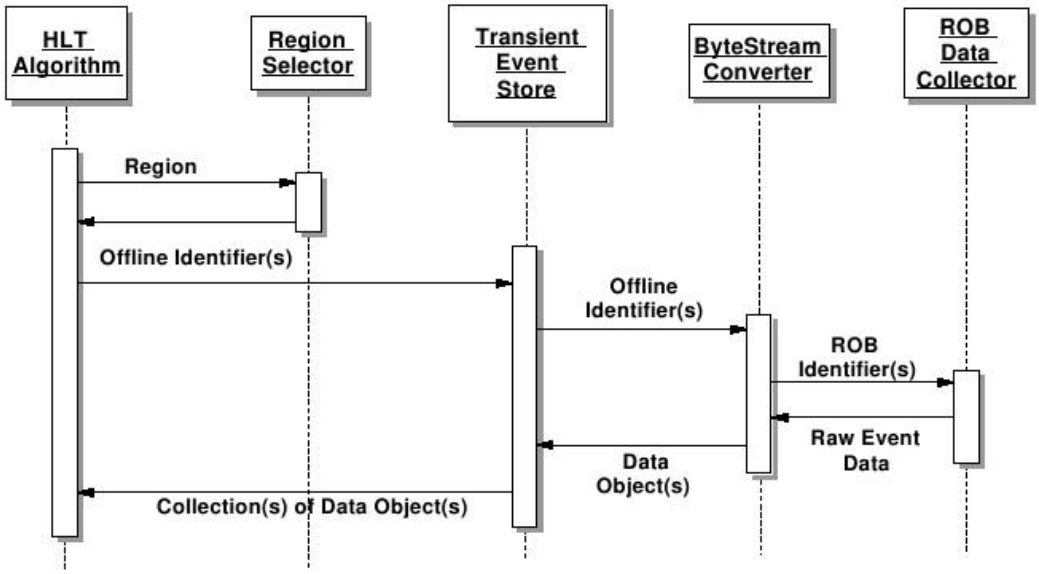

Fig. 3.3: A simplified schematic diagram of the sequence by which HLT algorithms request and receive event data.

data objects from TES. If TES does not contain the data objects, it requests these data from a *RawDataConverter*. The Offline identifiers are translated into Online or Read-out Buffer (ROB) identifiers which are used to request the data from the Data Collection System. The raw data returned from the Data Collection System are in the ByteStream format and are converted into data objects and are stored in TES in collections tagged with Offline identifiers. Then TES returns the collections of data objects the algorithm originally requested.

There are three types of algorithms in HLTSSW. *Data Conversion* algorithms use a sub-detector specific information and should prepare so called "low level" *Features*, which are input to the second category of HLT algorithms. *Feature Extraction* algorithms operate on abstract *Features* and *Trigger Related Data* to refine the event information. They built upon the *Data Conversion* output and extract the necessary input data to derive the *Trigger Decision*. It is beneficial to structure the algorithm processing in such a way that algorithms from a library of *Algorithm Tools* carry out common tasks such as track fitting or vertex finding.

Two possible subtypes of *Feature Extraction* algorithms are shown in the Figure 3.4. The *Reconstruction Algorithms* process *Features* and produce new types of *Features*, just like the offline reconstruction algorithms. The trigger specific is the use of the information in RoI objects to restrict the processing

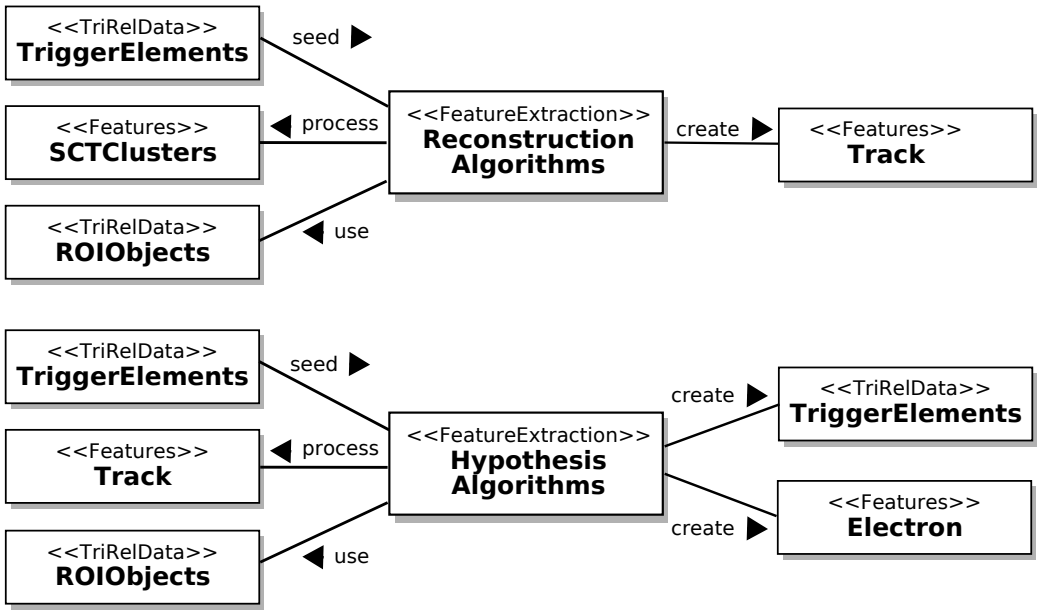

Fig. 3.4: Two types of Feature Extraction algorithms

to geometrical regions of the detector, which were identified by the LVL1 system. The *Trigger Elements* used by the *Steering* to "seed" the *Reconstruction Algorithms* represent these trigger relevant aspects of the event. The second subtype of algorithms is *Hypothesis Algorithms*. Their task is similar to particle identification. A *Hypothesis Algorithm* produces a new *Trigger Element* out of reconstructed *Features* whenever they validate the hypothesis of an assumed physics object. An example is the validation of an "electron" using a reconstructed *Features* as a *Calorimeter Cluster* and a *Track*, for which a new *Trigger Element* would be created.

In scope of this thesis one of the *Reconstruction Algorithms* for Transition Radiation Tracker (TRT LUT-Hough) is implemented in HLTSSW and possibility of improving of execution speed of different algorithms with reconfigurable co-processor help is considered.

Chapter 4

RECONFIGURABLE COMPUTING

4.1 Short introduction

A research-oriented experimental equipment must often to offer high-performance computing capabilities and a high degree of flexibility simultaneously. There are two primary methods in conventional computing for the execution of algorithms. The first is to use hardwired technology like Application Specific Integrated Circuit (ASIC) or a group of individual components on a board, to perform the operations in hardware. This hardware is designed specifically to perform a given computation and it is very fast and efficient for this task. However, this solution is inflexible in case of changes (even small) in the algorithms.

The second method is to use software-programmed microprocessors. It is a far more flexible solution. These processors execute a set of instructions to perform a computation. By changing these instructions, the functionality of the system is altered without changing the hardware. However, for the flexibility one should pay by performance. The processor must read each instruction from memory, decode its meaning, and only then execute it. This results in a high execution overhead. The set of instructions that may be used by a program is determined by processor design. Any other operations must be built composing existing instructions.

Reconfigurable computing is intended to fill the gap between hardware and software, achieving potentially much higher performance than software with a higher level of flexibility than hardware. Reconfigurable devices like Field-Programmable Gate Arrays (FPGA) contain an array of logical elements combined into configurable logic blocks (CLB) whose functionality is determined through programmable configuration bits. The CLBs are the main logic resource for implementing sequential as well as combinatorial circuits. Each CLB element is connected to a switch matrix to access to the general routing matrix. One CLB from Xilinx Virtex-4 FPGA [85] is shown in the Figure 4.1. A CLB element contains four interconnected slices. These

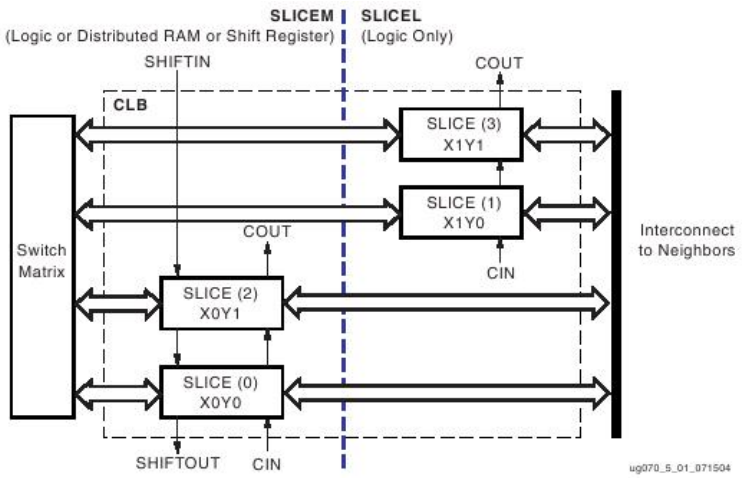

Fig. 4.1: Arrangement of Slices within the Xilinx Virtex-4 CLB.

slices are grouped in pairs and each pair is organized as a column. A simplified Virtex-4 slice is shown in the Figure 4.2.

Any custom digital circuits can be mapped to the reconfigurable hardware. Limiting factors are only the amount of available logic, routing resources, and I/O pins. Because of their flexibility, FPGAs can carry out different operations simultaneously and independently, so they are an excellent platform for parallel computing. The drawbacks are that FPGAs run at lower clock frequencies and draw more power than ASICs. For large volume products, FPGAs are more expensive then ASIC based hardware.

During the early stages of field programmable gate arrays, FPGAs were used primarily as glue logic for chip-to-chip communication or for other bridging type functionality. With the fast growing amount of the available logic it becames possible to implement complete functional blocks inside FPGA. This can be done by the user with a specifically created FPGA design or it can be hardwired inside the chip by the vendor. Components which are good candidates for a hardwired implementation are “on-chip memory” and various interface controllers. Current FPGA solutions encompass complete embedded processors and processing subsystems. But getting internal blocks of your FPGA to run fast is only half of the battle. Maximum system performance requires efficient interaction between FPGA and other components in your system. Modern FPGAs provide powerful high-speed IO interfaces. Therefore, over the past two decades FPGAs have evolved from a collection of gates for programmable logic to platform FPGAs integrating system-level capabilities:

- Clock management.

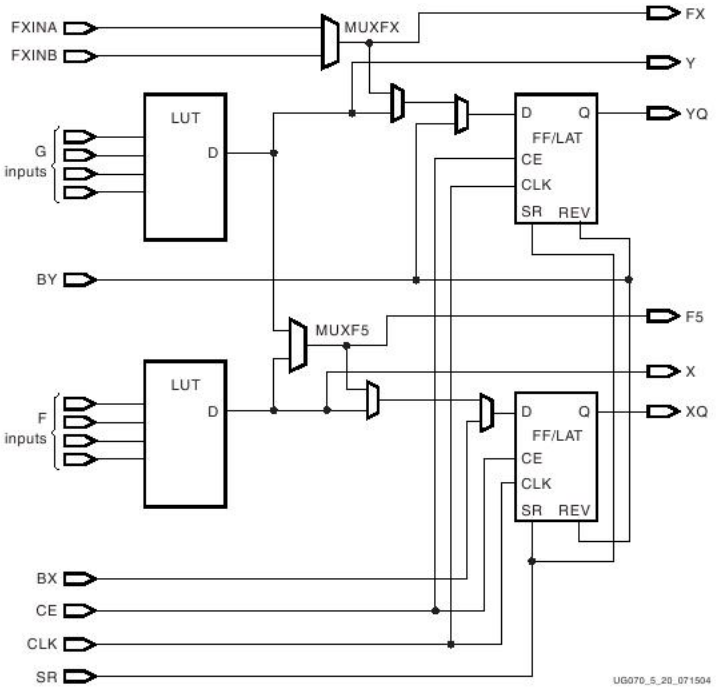

Fig. 4.2: Simplified Virtex-4 General Slice.

- Memories.
- Parallel and serial I/Os.
- Ethernet MACs.
- Microcontroller(s) and microprocessor(s).

The block diagram of a modern FPGA is shown in Figure 4.3.

Of these components on-chip memory is one of the most general-purpose, because memory in various forms is needed in almost any, more or less complicated design. Modern FPGAs provides internal block RAM, which is more compact and fast than memory created from logical elements. This embedded block RAM can be configured as a true Dual-Port RAM which allows an independent read and write operations and, therefore allows a fast read-modify-write cycle, what is particularly useful for the presented tracking application.

An FPGA is traditionally programmed using a dedicated programming language, such as Verilog [86] or VHDL [87]. The digital design must first be described at a low abstraction level using one of these languages and then compiled through several development tools. Results of this process

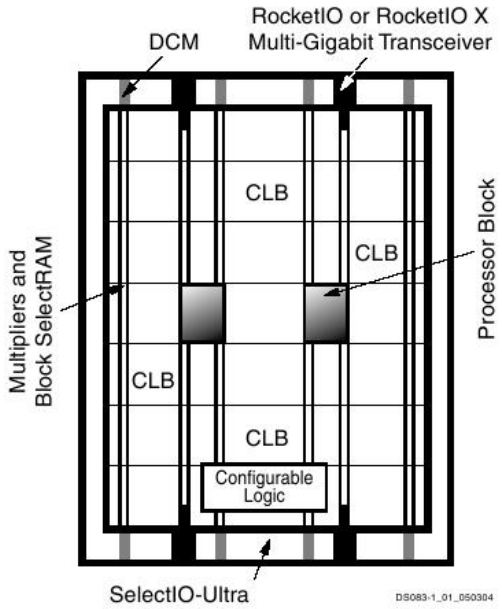

Fig. 4.3: A Generic Architecture Overview of modern FPGA (Xilinx Virtex2-Pro in this case).

is a *bitstream configuration file*. It contains the digital design in form that configure the FPGA to perform the task. The initial development is carried out by a hardware engineer with typical issues to be resolved such as clock cycles, registers, memory buses, arbitration schemes etc.

Today there are FPGAs with more than 100 000 logic elements. The typical frequency at which they can run is 150-250 MHz. The large number of logic resources and the fairly high operating frequency make FPGAs operate in the Tera-Instructions-Per-Second range if the system is decomposed into parallel subsystems. The inherent parallelism in FPGAs provides the unique capability to achieve significant acceleration through hardware. This is particularly important when managing compute-intensive applications. To achieve performance benefits and support a wide range of applications, reconfigurable systems are often made as a combination of reconfigurable logic and general-purpose processors. There compute-intensive calculations are mapped to the reconfigurable hardware and the processors are responsible for controlling the reconfigurable logic and executing program code that cannot be efficiently accelerated.

There are several ways to couple the reconfigurable logic with general-purpose processors (for example, see [88]). In the most tight couple case reconfigurable units can be integrated inside processor (or embedded processor

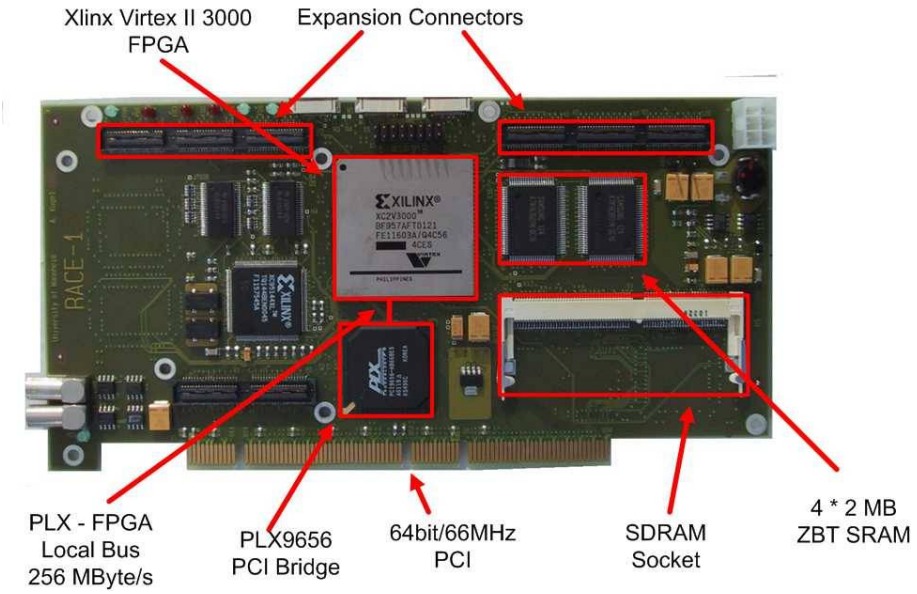

Fig. 4.4: Multi Purpose Reconfigurable Accelerator / Computing Engine

integrated inside reconfigurable hardware). However, the one of the most popular approaches is to make a reconfigurable unit as a co-processor which is connected to a bus for peripheral devices (PCI, USB, etc.). In such case a dedicated hardware is used as a co-processing engine to achieve algorithmic acceleration. The latter is the most flexible solution which allows a system-independent coupling.

4.2 The FPGA co-processor MPRACE

In a scientific laboratory, the FPGA-based technologies allow scientists to build systems based on the standard PC-technology, with hardware acceleration for fast computing, using an extension board rather than acquiring specialized processing units. The FPGA processor group at the department of Computer Science V from the University of Mannheim has developed several FPGA based reconfigurable computers (standalone processors [89, 90] as well as co-processors with Compact-PCI [91] and PCI interfaces [92]).

The FPGA co-processor – MPRACE (Multi Purpose Reconfigurable Accelerator / Computing Engine) [92] developed at the University of Mannheim was used as a hardware platform in the scope of this thesis. MPRACE is an FPGA co-processor based on a Xilinx Virtex-II FPGA and is made as a 64 Bit/66 MHz PCI card shown in Figure 4.4. The main features of this board are:

Dedicated PCI-to-Local bus bridge PLX9656 [93] with support for 32/64-bit, 33/66 MHz PCI operation and 32-bit, 64 MHz local bus operation. Read/Write bandwidth up to 230MB/s.

High capacity FPGA Xilinx Virtex-II [94] XC2V3000-4BF957C with 14 336 slices (28 672 LUTs/FFs), 12 DCMs, 96 Multipliers, 1728 Mbit of internal block RAM (96 RAM blocks×18 kBit), 684 I/Os is used in the first version of the board. There are also a boards with XC2V6000 FPGA for tasks with higher resource requirements. The FPGA can be configured via a PCI bus as well as via a flash-eeprom.

High bandwidth memory subsystem: 4 banks of ZBT SRAM (Zero-Bus-Turnaround[1]), each 512 k×36 bits (8 Mbytes in total) specfied for a clock frequency of up to 167 MHz. Additionally there is a slot for SDRAM SO-DIMM module with direct connection to the FPGA with a 64 bit data bus.

Flexible clock system supporting 2 modes: single clock and split clock modes. In the later case MPRACE provides two clock domains. One is used by the PLX local-bus and can be set to a frequency of 8, 16, 32 or 64 MHz. The second clock domain is distributed to the FPGA and all other MPRACE components (SRAM, SDRAM, ...). It can be switched to 8, 16, 32, 64 or 125 MHz. Alternative frequencies have to be generated inside the FPGA using one of the digital clock managers (DCM).

Extension capability For extension purposes two high speed connectors are mounted on MPRACE. Each provides 94 signal lines directly attached to the FPGA with up to 4 clock signals per connector.

Private board-to-board interconnect via a high-speed serial protocol (5 bits per direction at 200 MBit for each bit)

MPRACE is an universal co-processor which is used for various applications like High Energy Physics DAQ and trigger systems (see [95, 96]), image processing [97] and astrophysical simulations [98]. To make using of the co-processor for different tasks simpler, a control software was developed. It consists of a low-level library for the hardware access and a device drivers for the Linux and Microsoft Windows operating systems..

[1] That is similar to standard synchronous SRAM but has the same timing for read and write cycles. There is no needs to add wait cycles when changing from read to write direction.

The low-level library is an object-oriented code written in C++ which provides a complete set of access functions to control the co-processor. This includes:

- Detecting and initialising the MPRACE hardware.
- Uploading the FPGA configuration bitstream via PCI.
- Programming the on-board clock system.
- Data reading and writing to / from the FPGA co-processor via programmed I/O (PIO).
- DMA memory allocation and control of DMA read and write communication with the board.

All this can be done through the high level user-mode API (an end-user does not need to use kernel-mode driver calls) and can be used under the Linux and Windows NT/2000/XP operating systems (more details can be found in [99]).

An object-oriented design allows more or less easy integration of the FPGA co-processor into the existing (object-oriented designed) software for testing and performance comparison.

Chapter 5

RECONSTRUCTION ALGORITHMS FOR INNER DETECTOR

The first requirement in exploiting reconfigurable devices for high-performance computing applications is determining if your application is well suited to acceleration. Some of the ATLAS Inner Detector track reconstruction algorithms which are common to all B-physics channels and standard RoI processing have been tested for execution time and assessed for the suitability for speed-up by using a FPGA co-processor.

The main task of this timing measurements is to check how existing algorithms are fitted to the ATLAS LVL2 timing requirements ($\sim 10ms$). However, the algorithms analysis which is done together with the execution time tests allows us to select the best suited algorithm for the FPGA acceleration.

Using FPGA for algorithm acceleration foreseen in the architecture A of the ATLAS LVL2 Trigger (see section 1.2). FPGA co-processors can fit into the architecture B and C to reduce the size of the Local Processing farm(or just processing farm in case of architecture C). Usage of the FPGA co-processor can give some reasonable speed-up as contrasted to the general purpose processor only for those algorithms (or parts of algorithms), for which there is a possibility to fulfil calculations with a major degree of parallelism. The studies presented here were performed in the C/C++ framework, CTrig (CTrig-01-15-12), which was the richest set of algorithms available at the time of study, running on the computer equipped with dual Intel Xeon 2.4 GHz, 64 bit/66 MHz PCI bus, 1 GB DDR RAM main memories with the CERN Red Hat Linux 7.1. However, only one CPU has been used by algorithms. It is supposed that the algorithms will be included in ATLAS Trigger software which will have single threaded algorithms by design. All measurements were made using the dataset Y00347 (file 1: Y00347_1.atrdmp.data) containing approximately 156 events ($B \rightarrow \mu X$) with a pile-up at luminosity $10^{33}\, cm^{-2} s^{-1}$.

Pile-up occurs when the readout of a particle detector includes informa-

tion from more than one primary beam particle interaction – these multiple interactions are said to be "piling-up". At the LHC design luminosity of $10^{34}\,cm^{-2}s^{-1}$ pile-up is a major issue for ATLAS detector because the LHC beams will produce an average of 23 interactions each time they cross and the ATLAS detector is sensitive to tracks from more than one bunch crossing (the beams cross every 25 ns). This means that in addition to the hits of the physics event that triggers the detector readout, hits caused by many other interactions are recorded in the readout. The hits from these other interactions are not related to the physics event and represent a serious background.

In the scope of this thesis we are interested in algorithms for track finding in the Inner Detector. Typical Inner Detector events for low and high luminosity are shown in Figure 5.1 (figure from [41]).

The algorithms are briefly described and execution time measurements results are presented in this thesis. The trigger algorithms and their performance in terms of efficiency are described more fully elsewhere (for example in [82]). The detector geometry was described in section 1.1.1. The main task of this measurements was to check how existing algorithms fits to the ATLAS LVL2 timing requirements (∼10 ms for algorithm execution) and to have a look how can we (at least theoretically) improve timing with FPGA co-processor help.

5.1 Pixel-Scan

A "Pixel-scan" algorithm [100] uses only information from the Pixel Detector. The input data are the pixel cluster positions produced by a separate data preparation step [101]. In the first step of the full-scan algorithm, clusters are combined in pairs. All combinations of a one cluster per layer in the inner two planes of the barrel and in two of the end-cap planes are tested. A pair is retained if the trajectory extrapolation inwards passes within a given distance of the interaction point. A track is formed if a hit is found close to the extrapolated track into the third layer barrel layer or in one of the outer pixel end-cap layers. In both cases the extrapolation is linear.

Therefore the track finding proceeds independently for every triplet of layers. Lines connecting every couple of points in layer 1 and layer 2 (so called "links") are created for every triplet. This part of the algorithm may be done in parallel. For example:

- Step 1: Store all points from layer 1 in the co-processor memory
- Step 2: Transfer points from layer 2 one by one into the co-processor and calculate "links" parameters for this point and every point from

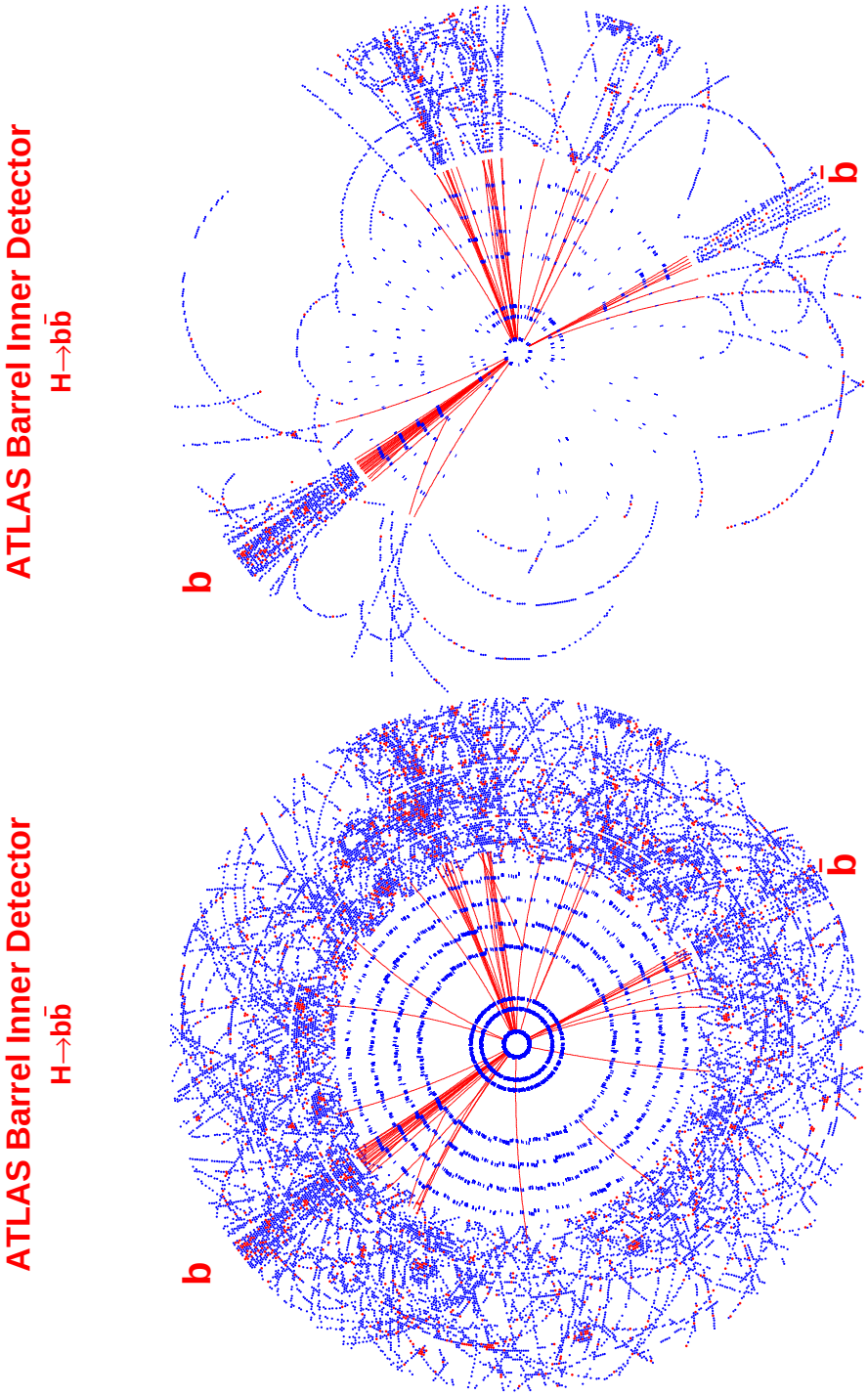

Fig. 5.1: Display of simulated $H \rightarrow b\bar{b}$ event in the ATLAS barrel Inner Detector. The upper figure is at low luminosity; the lower figure is at design luminosity of $10^{34}\, cm^{-2} s^{-1}$

Tab. 5.1: Execution times on a dual Xeon 2.4 GHz PC for the Pixel Scan. *The total time is not the sum of the average times shown in the table, but rather is the average of the total times per event.*

Algorithm part	**Execution time** ($p_{\mathrm{T}} > 0.5$)
Prepare cluster table	52.669 ms
Build tracks	4.775 ms
Solve ambiguities	3.451 ms
Create structure	0.076 ms
Delete list of clusters	2.243 ms
Total	**60.975 ms**

layer 1 in parallel.

But for calculating a single "link" we need six floating-point numbers (3 coordinates for every point in "link") and three of them should be read from the memory. If 32 bit floating-point numbers are used we should read 96 bits of data from the memory for only one "link" calculation even without any parallelism. According to [100] we have $\sim 50 - 75$ clusters per layer and for calculating all links for one point $\sim 150 - 225$ floating-point numbers should be read from the memory.

Therefore, even easily parallelized algorithms (or their parts) are not good candidates for an FPGA implementation if floating-point calculations are in use. Floating-point libraries for FPGA exist ([102] is one example of using floating-point arithmetic on FPGA), but the memory bandwidth is a limiting factor in such cases.

However, one could work with a reduced precision instead of 32 bit floats or with fixed-point calculations. In this case, if we use, for example, the MPRACE board with the XILINX Virtex-2 FPGA, we will have 96 RAM blocks with 18 kBit each were at least 96 numbers can be stored. Therefore we can calculate 96 "links" in parallel.

The execution time measurements results for this algorithm shown in the Table 5.1.

5.2 Precision tracker data preparation

The precision tracker data preparation [101] algorithm is divided into four different steps: data pre-selection, clustering, space-point formation and post-selection (optional). This set of algorithms process the raw data from the SCT and pixel detectors into a format suitable for the precision tracker LVL2 Feature EXtraction (FEX) stage. The raw data consists of hits in the format

produced by the Read-Out Drivers (RODs). Data preparation algorithms may be implemented in the ROD or in the LVL2 processor, EF processor or in processors dedicated to the data preparation (e.g. FPGA). All this algorithms are fast enough. The usage of the FPGA co-processor can give some reasonable speed-up only if the co-processor will be on the raw data path between the ROD and the LVL2 processor i.e. if the raw data from the ROD first came to the co-processor and then, after the preparation, come to the LVL2 processor. Otherwise time for the data transfer from the LVL2 processors memory to the co-processor can eliminates the possible speed-up.

5.3 IDSCAN

IDSCAN algorithm for high-p_{T} track finding in SCT [103] consists of four steps: z-finder, hit filter, group cleaner and track fitting.

During the first step the z position of the event is found. For this the RoI is divided into many ϕ bins. In each ϕ bin, all possible pairs of hits from different layers are constructed. For each pair the z position of the event is found by the linear extrapolation and an one-dimensional histogram is filled. After finding the z of the physics event, a two-dimensional histogram in (ϕ_0,$1/p_{\mathrm{T}}$) space is created. Each bin in that histogram corresponds to a small solid angle. A track (above certain p_{T}) from the physics event will be fully contained in one such bin, while a pile-up track from a different z will cross many bins. Therefore, each bin counts how many different layers have been hit. If the number of hits in a bin is higher than a predefined threshold all hits in this bin are accepted, else rejected. This step puts hits from the neighbouring bins into groups and returns these groups as result. The next step rejects garbage groups and garbage hits in a group and splits groups that contain many tracks to individual track candidates. For this another two dimensional histogram in (ϕ_0,$1/p_{\mathrm{T}}$) space is filled with hit triplets. It is a bit like Hough transform but for a single triplet only a single bin is incremented. Track candidates are accepted if groups contain enough hits. Finally, the track candidates are passed to the Kalman filter based track fitter [104].

All three steps have a histogramming phase but in the every case only the one histogram bin is incremented for the one input value. Therefore something like a pipelining technique should be used for these steps. As far as a data transfer from the host memory to the co-processor board can be fast enough (a DMA transfer, for example), one can transfer several input values to the board and run bin incrementing for this input values in parallel. However, on the fast host PC (fast processor with big amount of cache mem-

Tab. 5.2: Execution times on a dual Xeon 2.4 GHz PC for the IDScan. *The total time is not the sum of the average times shown in the table, but rather is the average of the total times per event.*

Algorithm part	**Execution time**
ZFinder	2.072 ms
Hit Filter	9.693 ms
Kalman Filter based fitter	0.725 ms
Last Loop	0.742 ms
Total	**13.234 ms**

ory and fast main memory) transferring can take more time than doing the histogramming on the host processor.

The execution time measurements results for this algorithm are shown in the Table 5.2. It is very fast algorithm which is nearly fit to the LVL2 timing requirements (10 ms) even on 2.4 GHz CPU.

5.4 SCTKalman

SCTKalman [105, 106] is an algorithm based on the Kalman filter. A track search may be initiated by either an external or an internal seed. For the purposes of the B-physics trigger, the Si-Kalman FEX uses either pixel or TRT tracks as external seeds in the pixel-seeded or TRT-seeded mode respectively. The track segment is extended by adding nearby hits layer by layer. The algorithm takes into account multiple scattering.

The algorithm starts from a small track segment or from a fitted track of a neighbouring detector, and then extends the candidate tracks by adding measured points one by one. The fitted parameters and weight matrix of the candidate track are updated when adding a point, and give an increasing precision on prediction of the next point. Thus, pattern recognition and track fitting can be accomplished simultaneously.

It is a consecutive algorithm and it can not benefit from a FPGA co-processor usage.

The execution time measurements results for this algorithm shown in the Tables 5.3.

The reason why there is so big differens in execution speed between the pixel-seeded and the TRT-seeded mode is following. The hit density in TRT is very high and the reconstruction algorithms can find a lot of tracks (not only tracks from interesting particles, but so called "fake" tracks, tracks from background or composed from hits from others tracks, as well), which is used

Tab. 5.3: Execution times on a dual Xeon 2.4 GHz PC for the SCTKalman with different seeds.
The total time is not the sum of the average times shown in the table, but rather is the average of the total times per event.

Seeded by:	*Pixel*	*TRT-LUT*	*TRTKalman*
Algorithm part	**Execution time** ($p_T > 0.5$)		
Data preparation	0.495 ms	3.510 ms	1.948 ms
SctKalman FEX	11.187 ms	79.447 ms	53.348 ms
Output preparation	3.603 ms	14.439 ms	9.128 ms
Total	**15.286 ms**	**97.396 ms**	**64.423 ms**

for seeding the SCTKalman.

5.5 SiTree

This algorithm associates the locally related hits in the SCT using a "link-and-tree" method [107, 108]. The basic element used in this algorithm is not a single space point, but a "link". The link is a segment of a potential track with enough hits to determine the track parameters of the segment. If the tracks being sought are straight lines or circles coming from the interaction point, only two hits are required to define a link. If the tracks are circles originating at arbitrary points, at least three hits are necessary to define a link since three points define a circle. In the current implementations of the pixel-guided and TRT-guided FEX with the tree algorithm only the two hit link is used assuming the origin of the track is very close to the interaction point.

This looks very similar to the Pixel-scan algorithm from the FPGA co-processor point of view (see section 5.1) and same comments can be applied. We can build "links" in parallel, but for these we need a floating-point arithmetic. However, we can work with a reduced precision or with fixed-point numbers.

The execution time measurements results for this algorithm shown in the Table 5.4

5.6 TRTKalman

The TRTKalman algorithm [109] for high-p_T track reconstruction consists of four stages:

- An initial track search using a histogramming method

Tab. 5.4: Execution times on a dual Xeon 2.4 GHz PC for the Pixel-seeded SiTree.
The total time is not the sum of the average times shown in the table, but rather is the average of the total times per event.

Algorithm part	**Execution time** ($p_{\mathrm{T}} > 0.5$)
Data preparation	0.409 ms
SctKalman FEX	9.241 ms
Fill output track structures	0.049 ms
Total	**9.699 ms**

- Fine tuning
- Track fitting
- Track candidate selection

The initial search looks for hits on a straight line in the appropriate projection. The trajectory of a charged particle is linear in the $z-\phi$ plane and approximates to a straight line in the $R-\phi$ plane for particles with high p_{T} produced close to the origin. The trajectory can be described as:

$\phi \approx \phi_0 + C_t \cdot R$ (for Barrel)

$\phi = \phi'_0 + C_z \cdot R$ (for End-Cap)

where $\phi'_0 = \phi_0 - z_0 \cdot C_z$; $C_t \approx 0.003 \cdot B \cdot q/p_{\mathrm{T}}$; $C_z = C_t \cdot \tan\theta$ and θ is a polar angle of the track. Thus all points on the track lie on a line characterized by a slope C and an intercept ϕ_0. For each hit in a detector, within the RoI, the value of ϕ_0 is calculated for each value of the slope between $-C_{max}$ and $+C_{max}$. The value of C_{max} is determined by the lowest p_{T} track that is to be sought. Each hit will populate many cells of the histogram, but for each track in the RoI there will be a bin where all hits on the track have an entry (provided an appropriate bin size has been chosen). Thus the track candidates can be identified from peaks in the histogram. If the RoI contains more than one part of the TRT ($-z$ end-cap, $-z$ barrel, $+z$ barrel and $+z$ end-cap), the initial search is performed independently in the two parts. The bin size in ϕ_0 is not fixed but is determined dynamically for each RoI by the straws with hits. The low-ϕ and high-ϕ boundaries of all straws with hits in the RoI, sorted in the order of increasing ϕ, determine the division of the histogram. For speed, some quantities are calculated in an initialisation phase and stored in look-up tables. For each bin with more than eight (this is a parameter) hits (out of a maximum of typically 40 straws), the procedure continues to the fine tuning stage.

Tab. 5.5: Execution times on a dual Xeon 2.4 GHz PC for the TRTKalman. *The total time is not the sum of the average times shown in the table, but rather is the average of the total times per event.*

Algorithm part	**Execution time** ($p_T > 0.5$)
Detector Geometry Initialization	49.964 ms
Space point production	3.359 ms
Histogramming	92.273 ms
Fitting	24.134 ms
Total	**169.73 ms**

The execution time measurements results for this algorithm shown in the Table 5.5

An initial track search which utilizes a Hough transform with look-up table (most time consuming part of this algorithm) can be done in parallel inside the FPGA like it is implemented for the TRT LUT-Hough algorithm.

The TRT LUT-Hough algorithm - a look-up table based Hough transform algorithm for TRT - was selected as a best suited algorithm for hybrid FPGA/CPU (VHDL/C++) realisation. Description of this algorithm and timing measurements results are presented in the next chapter.

Chapter 6

TRT LUT-HOUGH ALGORITHM

A look-up table (LUT) based Hough transform algorithm for TRT (TRT LUT-Hough algorithm) was created keeping in mind the B-physic's tasks (searching for low-p_{T} tracks in the entire TRT volume) and it is the best candidate for the acceleration with the FPGA co-processor. Here a short description of TRT LUT-Hough algorithm is done. Detailed information can be found in [110, 111]. Results of hybrid realisation of this best case are discussed in this chapter as well as some works for improving the physics performance of the algorithm.

6.1 TRT LUT-Hough Algorithm description

The Transition Radiation Tracker (TRT) (see section 1.1.1) provides tracking information and contributes to the electron identification. TRT straws provide a two-dimensional position measurement, r-ϕ in the barrel and z-ϕ in the end-caps. Typically, the TRT provides 36 measurements on each track in the detector acceptance. This results in $\sim$20 000 hits per event.

The TRT LUT-Hough algorithm consists of a track candidate search followed by track-fit performed to determine the track parameters. Since all the particle trajectories to search for can be calculated in advance a histogramming method based on the Hough Transform is well suited for the initial track search in the TRT. The Hough transform is a standard method in the image analysis that allows recognition of global patterns in an image space by recognition of local patterns in a transformed space [30, 31]. The idea of the Hough transform is that for a set of points (x_i, y_i), lying on the straight line $y = a_0 x + b_0$, the relations $y_i = ax_i + b$ have to be satisfied. These relations give the set of straight lines in (a, b) coordinate system, crossing at (a_0, b_0). In the presence of measurement errors, i.e. if $ax_i + b - y_i = h_i$, where h_i are random numbers with a known distribution $f_i(x)$, one can consider the function $H(\vec{p}) = \sum_i f_i(h_i(\vec{p}))$, with a vector of parameters $\vec{p}$, defining the pattern – set of points, lying close to straight line (in this case $\vec{p} = (a_0, b_0)$).

The algorithm makes use of the fact that the trajectory of a charged particle produced close to the origin is linear in the $\phi - z$ plane and approximates to an exact circle in the $\phi - R$ plane. The assumption of a straight line in the $\phi - R$ is not sufficiently accurate for particles with low-p_{T}. The candidate search looks for points which lie on a predefined line in the appropriate projection. The trajectory of a particle produced at $(R, \phi, z) = (R, \phi_0, z_0)$ can be described in terms of an intercept, ϕ_0 (ϕ_0'), and a slope, C_{t} (C_z), in the $\phi - R$ ($\phi - z$) projection respectively:

$$\sin(\phi - \phi_0) \approx C_{\mathrm{t}} \cdot R \ (\phi - R \,\text{projection})$$

$$\phi = \phi_0' - C_z \cdot z \ (\phi - z \,\text{projection})$$

where $\phi_0' = \phi_0 - z_0 \cdot C_z$. The slope is inversely proportional to the p_{T} of the track (in units of GeV):

$$C_{\mathrm{t}} \approx 0.003 \cdot B_z \cdot q/2p_{\mathrm{T}} \ (\phi - R \,\text{projection})$$

$$C_z = 0.003 \cdot B_z \cdot q \cdot \tan(\theta)/2p_{\mathrm{T}} \ (\phi - z \,\text{projection})$$

where θ is the polar angle of the track, B_z is z component of the magnetic field strength (in units of T) and q is the particle charge (in units of e).

The TRT LUT-Hough algorithm is based on the idea that every hit in the three-dimensional detector image can belong to a number of possible (predefined) tracks characterized by different parameters as described above. All such tracks (or roads) are stored in a Look-Up Table (LUT). Thus every hit increases the "probability" for the existence of these tracks by one (histogramming). The histogram for a single track consists of a "bow-tie" shaped region of bins with a peak at the centre of the region as shown in Figure 6.1. The bin at the peak of the histogram will, in the ideal case, contain all hits from the track. The roads corresponding to the other filled bins share straws with the peak bin, and so contain sub-sets of the hits from the track. The histogram for a more complex (Figure 6.2) event consists of a superposition of the entries from the individual tracks. The bins containing the complete set of points from each track can be identified as local maxima in the histogram. After a clean-up step followed by a fit the final tracks are selected. The LUT based Hough Transform algorithm for TRT is implemented in C++ and integrated into several software frameworks for the ATLAS Trigger investigation. An initial version of the algorithm is implemented in VHDL for the MPRACE board as well. This implementation is integrated into the same framework (t2Ref and CTrig) for a performance study and comparison with the C++ implementation. The VHDL implementation is not tested in the HLTSSW/ATHENA environment but the C++ version of TRT LUT-Hough

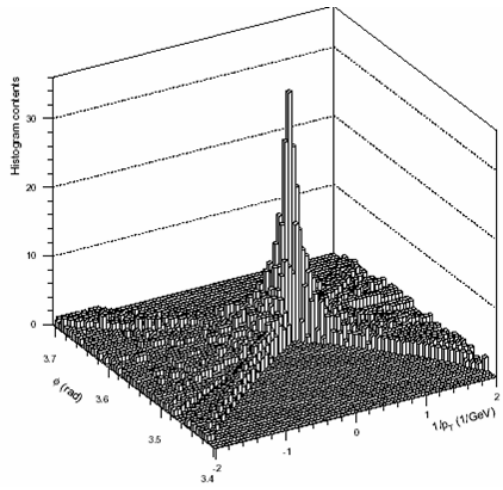

Fig. 6.1: A histogram due to an isolated muon in the barrel TRT.

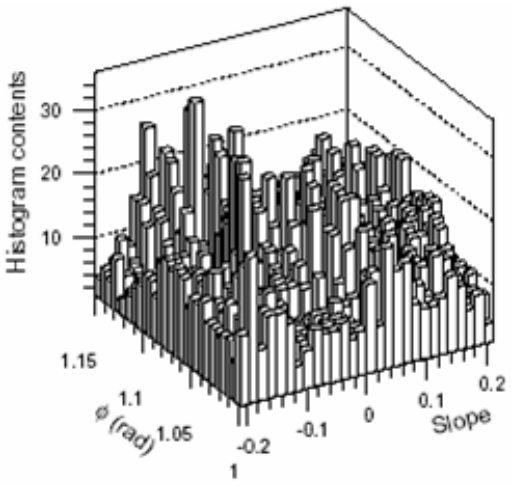

Fig. 6.2: A Histogram created for a B-physics event.

is ported into HLTSSW and all works for algorithm's physics performance improving is done in that framework.

The entire algorithm, as included in the CTrig software, consists of the following steps [110, 111] (Figure 6.3):

Initial Track Finding: This uses a LUT-based Hough Transform to find potential track candidates. A specially created look-up table (so called "straw-ordered" LUT) is used here. In the barrel the Hough Transform is performed from (R, ϕ) space to $(\phi, 1/p_{\mathrm{T}})$ space. This space is divided in 1024 parts (or bins) in ϕ and 80 parts in $1/p_{\mathrm{T}}$. Thus, the LUT consist of 81 920 predefined roads. All predefined roads point to the origin. The assumption of straight lines in the $R-\phi$ projection is not sufficiently accurate for low-p_{T} tracks in magnetic field. Therefore predefined overlapping roads are computed as exact circles in the $x-y$ projection. The road width increases linearly from 4.5 mm at layer 1 (numbering from the innermost layer outwards) to 6.8 mm at layer 42 and is then constant and equal to 6.8 mm from layer 42 to layer 73. With this definition, ∼65 straws are assigned to each road. The predefined roads overlap by 30% - 50% in $1/p_{\mathrm{T}}$ and ϕ. This roads overlapping prevents the loss of hits from a track with a trajectory which could otherwise pass between two predefined roads. On the other hand, it can lead to multiply reconstructed tracks, which have to be eliminated in subsequent steps. Each straw is assigned to ∼120 roads (max.130). Therefore we have two-dimensional table with straw number as an index.

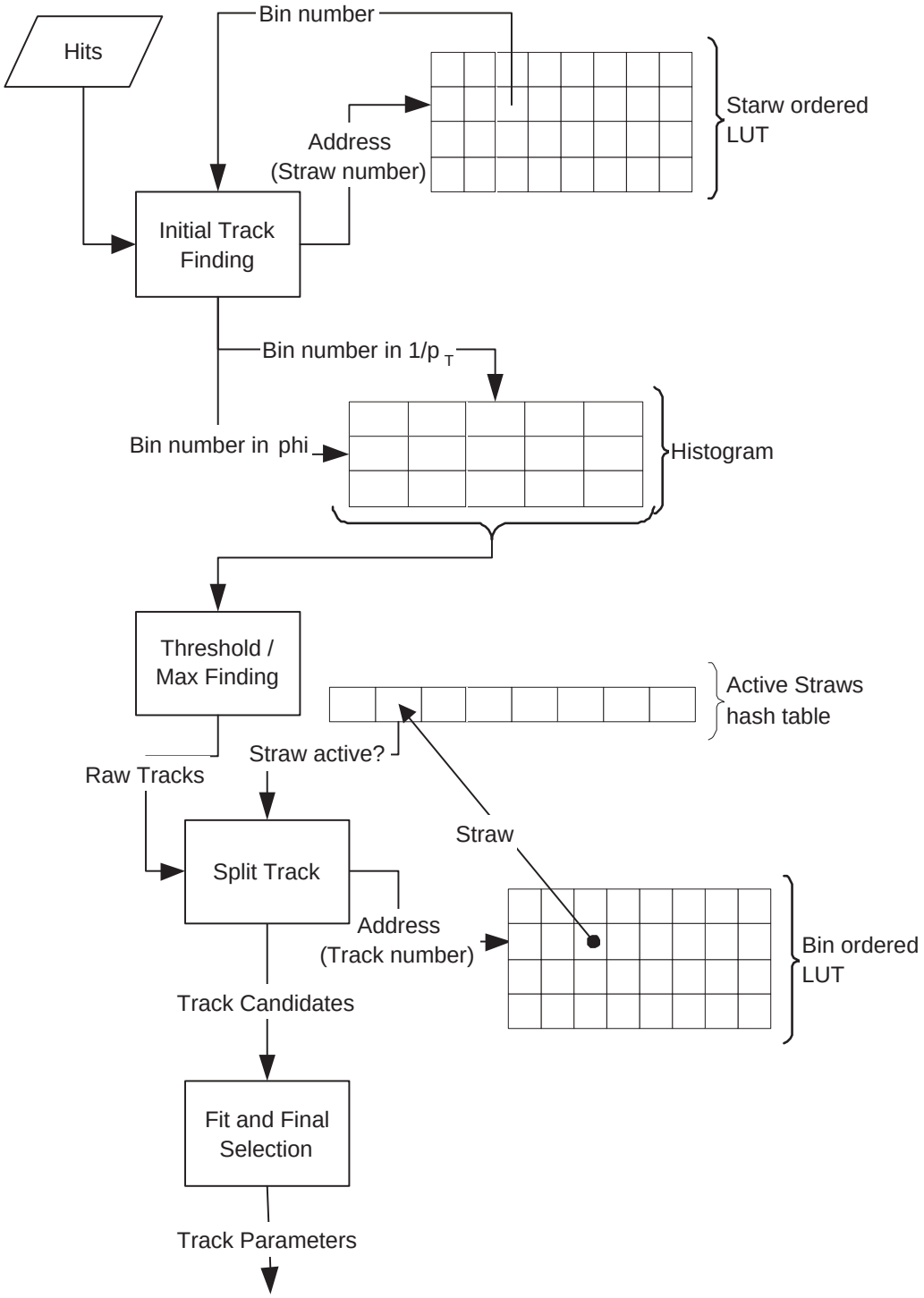

Fig. 6.3: TRT LUT-Hough algorithm

Thresholding and Local Maximum Finding: This selects potential track candidates and eliminates multiple reconstructed tracks. A cut on the number of hits is first applied to reduce the number of bins to be considered by a maximum finding and to eliminate small peaks due to bins with entries from sub-sets of the hits from more than one track. The maximum finding selects as track candidates bins which have more entries than neighbouring bins (regions for maximum finding slightly different in shape in barrel and end-caps). If two neighbouring bins have the same number of hits, only one bin is chosen as a track candidate.

Track Splitting: This removes hits incorrectly assigned to a track, and splits tracks that have been erroneously merged. In this step the pattern of hits associated to a track candidate is analysed. If a potential track candidate contains $N_{\rm is}$ consecutive layers without a hit, the track is split into two separate candidates either side of the gap. If one of the candidates contains more than $N_{\rm thr}$ hits, it is retained. If both candidates pass this threshold, the track segment which starts at the lowest radius is retained. The result of the track splitting step are candidates that consists of a sub-set of the straws in a road. For this step a "bin-ordered" LUT is constructed (each bin correspond to a single road). The list of straws lying within the road is stored in the LUT. To speed-up the retrieval of the information on which straws in the road have a hit, a table is filled once per event with the pattern of 1's and 0's corresponding to straws with and without hits. This table also stores threshold information for all active straws.

Track Fitting and Final Selection: This performs a least square fit in the r-ϕ (barrel) or z-ϕ (end-caps) plane. The algorithm uses only the straw position (i.e. the drift time information is not used). After the fit the threshold $p_{\rm T}^{rec} > p_{\rm T}^{thr}$ is reapplied. The final track candidates are selected during this step.

Profiling a C++ implementation of the TRT full scan algorithm shows that the most of the computing time is spent in accesses of LUTs, incrementing of 8-bit numbers, and the local maximum finding operations. A CPU-only implementation of Histogramming (or Initial Track Finding) and Thresholding / Local Maximum Finding requires ~62% of the total processing time. These two steps are good candidates for the FPGA implementation because of both steps can exploit an 80-fold parallelism (80 $1/p_{\rm T}$ blocks) and make use of the fast internal dual port block RAM. The main difference between the FPGA implementation and the CPU implementation of the Hough transformation is that the loop over all predefined roads for one straw, which is

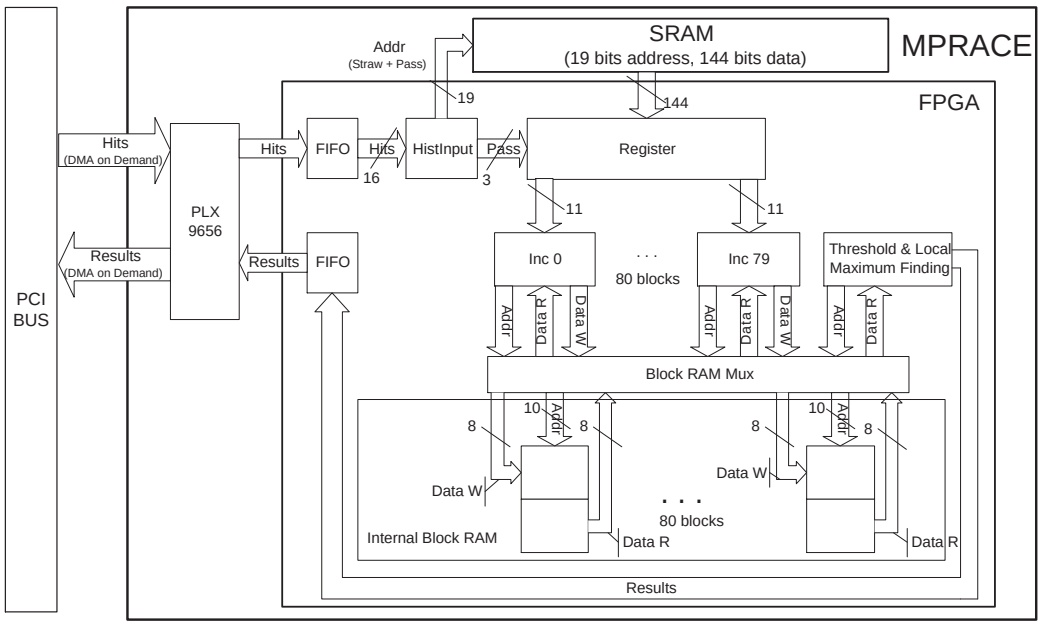

Fig. 6.4: FPGA initial track finding

performed once per hit for incrementing of ∼120 histogram counters, is executed sequentially in the CPU and in parallel in the FPGA.

6.2 VHDL implementation of the TRT LUT-Hough Algorithm

A part of TRT LUT-Hough algorithm for the TRT barrel was implemented in VHDL and investigated in t2Ref and CTrig as a proof of concept to demonstrate the feasibility of ATLAS LVL2 tracking algorithms acceleration with a FPGA co-processor help. A general schematic view of the VHDL implementation for the initial track finding step of the algorithm is shown in Figure 6.4. This implementation takes advantages of both the external SRAM and the internal RAM blocks.

The initial track finding algorithm works as follows. The array of active straws (hits) prepared in the host memory and transferred over PCI bus to MPRACE by DMA. For each hit straw, counters are incremented for all roads containing that straw. The histogram counters for all TRT straws which have to be incremented are stored in the "straw-ordered" LUT. The construction of the LUT with overlapping roads in ϕ and $1/p_{\mathrm{T}}$ guarantees that each active straw contributes to exactly one or two pre-defined ϕ values of a certain predefined $1/p_{\mathrm{T}}$. For the barrel, there are 80 "$1/p_{\mathrm{T}}$-blocks" with 1024 possible

ϕ each, thus 81 920 patterns are predefined. Therefore one or two values out of 1024 possible values, which have to be incremented, are stored in 11 bits, because the (possibly) two values are always directly neighboured (10 bits identify one (always smaller) of the ϕ values + 1 bit to select should we increment neighboured (the next one, not the previous) value: "1", or not: "0").

The "straw-ordered" LUT is stored in the external SRAM of the MPRACE board. An address space of 16 bits for the straws in the "straw-ordered" LUT is required (if information about symmetry of the detector is not in use). The SRAM itself has 19 bit addresses, of which 16 are used to identify the current straw. The LUT stores for each straw address 880 bits of the histogram counters addresses (80×11 bits). The word length of the SRAM is only 144 bits; therefore seven steps (passes) with the same straw address (16 bits) and changing pass addresses (3 bits) are needed for transfer all information corresponding to one straw from SRAM (LUT) to FPGA. Using the information about detector symmetry can significantly reduce the requirements for size and word length of the memory for LUT. However in scope of CTrig we have simulated data for detector geometry without any symmetry (see section 1.1.1). The histogram is stored in FPGA's internal RAM blocks. The implementation profits from a true Dual Port RAM (internal RAM Blocks) to allow a fast read-modify-write cycle during one external SRAM cycle. Therefore histogramming for each straw is executed in seven passes with up to three clock cycles each, whereas the CPU implementation requires looping sequentially over 130 histogram bins. Time for waiting data from SRAM can be reduced by using two separate clock signals inside the FPGA design: one for the SRAM and SRAM controller and another for the rest of design. The SRAM controller can be synchronized by a clock signal with higher frequency. Therefore, seven passes for reading LUT data from SRAM can be done in 3 or 4 "main" clock circles.

The histogramming step is followed by the maximum finding step. Here a threshold filter is applied first, then local maximum search is carried out. The maximum finding has to be applied in two dimensions (ϕ and $1/p_{\mathrm{T}}$). This is done in parallel for 80 $1/p_{\mathrm{T}}$-blocks and sequentially for 1024 ϕ-blocks. Only close neighbours a taken in to account in this procedure. Results are the histogram counters with values above the threshold and which are local maximum in both ϕ and $1/p_{\mathrm{T}}$. The output information contains the predefined ϕ and $1/p_{\mathrm{T}}$ values of the track (predefined from the LUT) and the number of active straws corresponding to the track. These results stored in the output FIFO and transferred over PCI bus to the host memory by DMA for further processing.

Let's consider the VHDL implementation of the initial track finding and

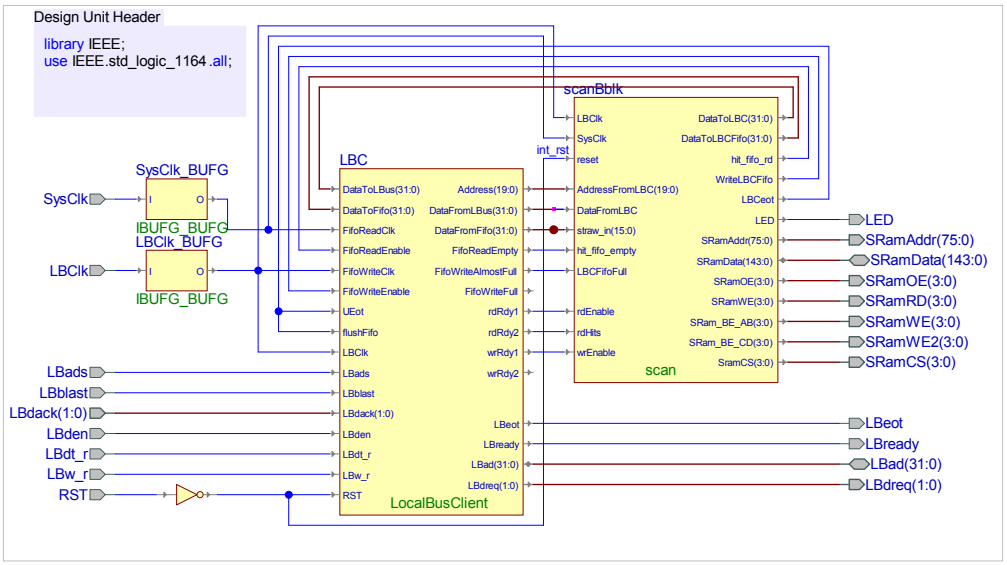

Fig. 6.5: TRT LUT-Hough initial track finding block diagram

maximum finding steps in more details. The design consists of two big parts as shown in Figure 6.5:

Local Bus client is responsible for communication with PLX PCI bridge. It allows data exchange between the FPGA design and the control software over the PLX PCI bridge and the host computer PCI bus. The Local Bus client provides several address ranges for communication with the rest of the design. Two of them are used: one for hits data and results and second for control and status data respectively.

Scan block is a main working part of the TRT LUT-Hough algorithm initial track finding and maximum finding implementation. The internal structure of this part is shown in Figure 6.6. A module for communication with ZBT memory located in this block as well.

The design synchronization is provided by two different clock signals. One ("LBClk" in the figures) is used for local bus operations and limited by the PLX bridge local bus speed (64 MHz). Another ("SysClk") is used for scan block synchronization. Asynchronous FIFOs are used as interfaces between two clock domains.

During algorithm initialization step the co-processor board is initialized and configured and Look-Up tables are created. To write these tables into the MPRACE ZBT memory a special FPGA configuration bitstream is used. After LUT writing, the configuration bitstream for initial track finding is

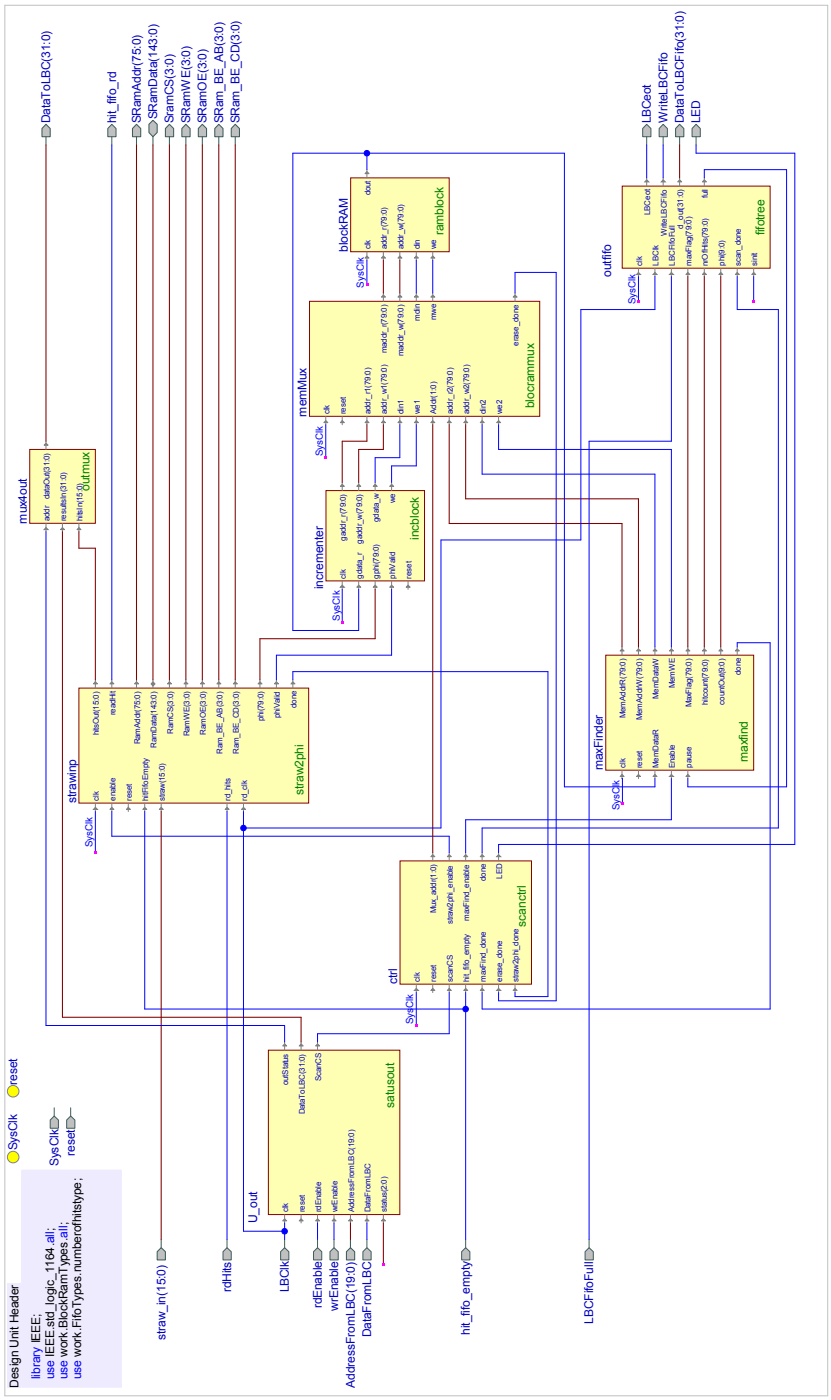

Fig. 6.6: Block diagram for the internal structure of the main working part of the TRT LUT-Hough algorithm initial track finding and maximum finding implementation.

loaded into FPGA. This is a quite slow process but it only need to be done once before the start of data taking and, therefore, has no strict timing requirements. After the configuration and the initial reset, the co-processor is waiting for data.

The TRT LUT-Hough algorithm is supposed to work in the Level-2 trigger, when all hits information for current event is already available. Therefore, for optimization of data transfer over PCI bus, all hits at first are packed (one unsigned integer number contains information about a straw layer number and a number of the straw with a hit in the layer) and stored in host memory. After the last hit information the "end-of-event" marker is stored and transferred. To send this information to the co-processor the DMA transfer utilized PLX9656 PCI bridge DMA controller is used. The data is received by the Local Bus Client and stored in FIFO.

The PLX9656 bridge provides a number of DMA modes being bus-master for PCI. In all of them the PLX bridge reads data from the PCs memory and writes this data to a local-bus address or vice versa. The PC memory area may be a continuous block or a scattered number of buffer areas (defined by a list of start addresses together with length information). A DMA flow control is also available on the local-bus which can delay the DMA operation if no more data is present (DMA-on-demand). It is asserted by a signal line between the FPGA and the PLX 9656, called DREQ (DMA Request). For all DMA operations a number of PIO accesses are required to program a set of registers inside the PLX 9656 chip.

Presence of the data in the FIFO indicated by the special signal (called "*hit_fifo_empty*" in the Figure 6.6). With this signal the "Scan" block is switched to the "histogramming" stage. Hit information is read from the FIFO and is used as part of the address for ZBT memory. Seven passes with the same straw address (16 bits) and changing pass addresses (3 bits) are needed to transfer from SRAM (LUT) all information corresponding to one straw to FPGA. Therefore, the value from the "pass" counter is used as the second part of the address. Data from SRAM ("phi" signal in the picture) is a "address" of histogram counters which should be incremented. It is registered and a histogram counters incrementing is done in one clock circle after the "phiValid" signal is asserted.

The two-dimensional histogram in (ϕ, p_{T}) space is filled during this step of algorithm. As it was already mentioned, we have 1024 steps in ϕ and 80 steps in p_{T}. For storing this histogram 80 (from 96 available in the FPGA) internal RAM blocks are used. Every block corresponds to one of 80 predefined p_{T} values. Eleven bits are used to address one or two (directly neighboured) from 1024 ϕ values with selected p_{T}. The true Dual Port RAM organization allows a fast read-modify-write cycle for the histogram counters.

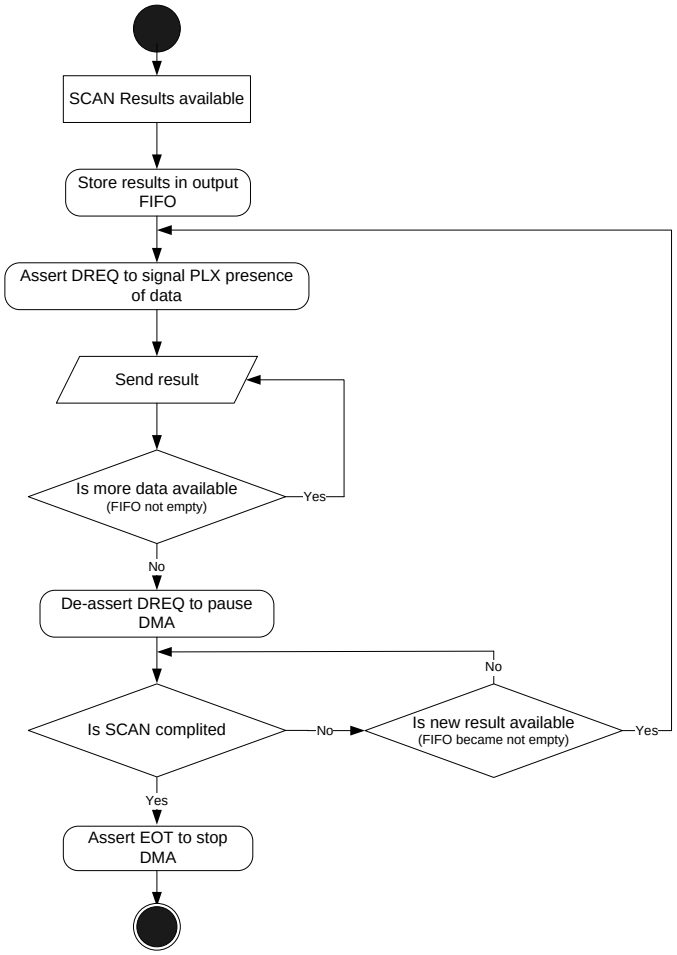

Fig. 6.7: Flow diagram of the DMA-on-demand data transfer

After receiving the "end-of-event" marker the design is switched to the "maximum finding" stage. During this step we are looking for local maxima in the histogram data after applying the threshold. Only direct neighbours are taken in account. The histogram data is read in parallel from 80 RAM blocks. The results contain a number of hits, p_{T} and ϕ block number packed in one world. These results are stored in the output FIFO and transferred to the host memory using the DMA-on-demand feature of the PLX bridge. Figure 6.7 shows the flow diagram of this DMA transfer. The PLX9656 is pre-initialized for DMA-on-demand by the control software (for example by CTrig with a running algorithm). To start the transmission the control signal DREQ is used. When data is available the DREQ signal is asserted and the PLX9656 starts to fetch (read) the initial track finding results. They are

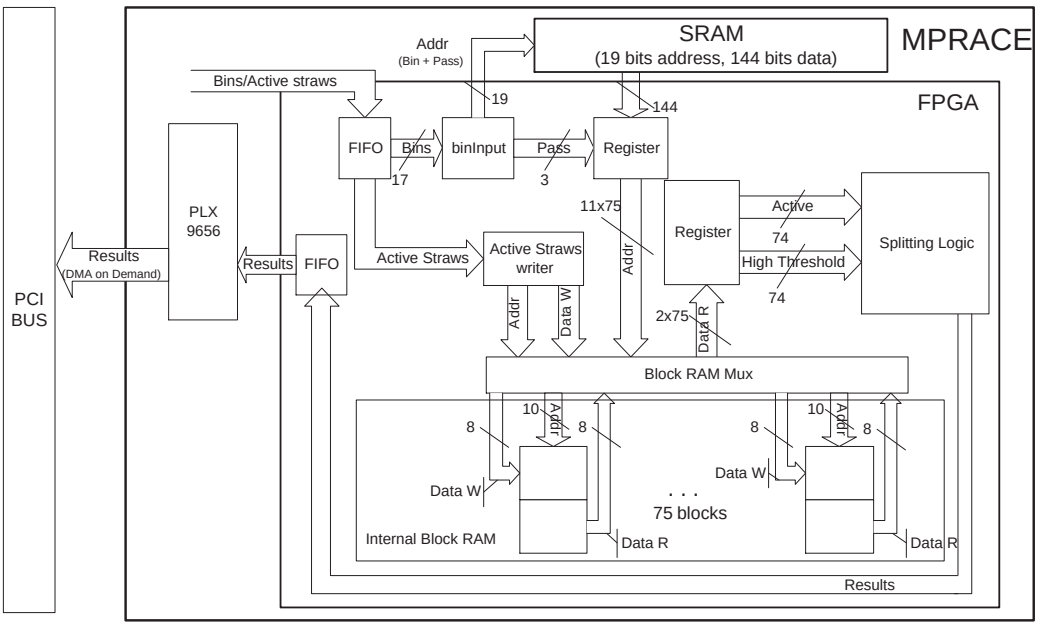

Fig. 6.8: FPGA track splitting

written sequentially into the DMA buffer memory of the host-PC. It is a pull-scenario in which the FPGA is the passive and the PLX9656 the active component. When all available data is transferred but the "maximum finding" process is still running, the DREQ signal is de-asserted to pause the DMA transfer. The transfer will be resumed when new results became available and DREQ will be asserted again. When all data is transferred and "maximum finding" process is ended, the EOT ("End-Of-Transfer") signal is asserted. It is lead to flashing the DMA FIFO in PLX9656 and stopping the data transmission.

If the track splitting step is also done inside the FPGA, the results of the local maximum finding step are read from the FIFO by the "splitting" part of the design.

A schematic view of the FPGA implementation of the track splitting step is shown in Figure 6.8. It has similar to initial track finding and local maximum finding input block and ZBT SRAM interface and uses absolutely the same interface to the internal block RAM. The track splitting step uses 75 internal RAM blocks (each block for one layer/plane) for a hash-table. The table stores information for each straw in this layer/plane if it is active or not ("active straws" hash-table), and the threshold information (and, optionally, the drift-time). The SRAM is used to address the bin-ordered LUT with histogram bin numbers above the threshold which were a local maximum. The

LUT outputs all straws which belong to the pattern definition layer by layer. The bin-ordered LUT stores 825 bits (75 layers×11 bits for straw number in layer). The word length of the SRAM is 144 bits; therefore, six steps (passes) with the same bin address and changing pass addresses are needed to transfer all information corresponding to one pattern from SRAM (LUT) to FPGA. The same technique like in initial track finding is used. SRAM address consists of two parts: pattern number created from p_{T} and ϕ block number from the previous step and the value from the "pass" counter. Data from SRAM is locked in the register. Additional speed-up can be achieved by using separate clock signals for SRAM controller and main design.

Straw numbers from the LUT are used as an address for the "active straws" hash-table stored in the internal block RAM. This table should be filled during the first step ("histogramming"), when information about hits are transferred to the FPGA. We have two 74 bit words from this hash-table: one of them gives us information about active straws, the second – about high threshold hits. This information is used to perform the track splitting and the result is a 28 bit word which contains a number of hits, a number of high threshold hits, and the first and the last hit layer for every track candidate. This result is passed to the host CPU over PCI bus by "DMA-on-demand" for the track fit.

It is possible to put both LUTs (straw-ordered and bin-ordered) into SRAM memory of the MPRACE board if information about detector symmetry will be in use. For the barrel the 32-fold symmetry allows to store only $32\ \phi \times 80\ 1/p_{\mathrm{T}} = 2\,560$ bins. Therefore we need a 20 bit address at the SRAM: 16 bits for "straw-ordered" LUT (52 544 straws), 3 bits for passes and 1 bit for selecting one of LUTs ("bin-ordered" or "straw-ordered"). Without the information about symmetry we need 21 bit address: 17 bits for "bin-ordered" LUT, 3 bits for pass and 1 bit for LUT selecting.

6.3 Execution Time Measurement Results

Measurements have been performed on the same platform as described in Chapter 5 (computer equipped with dual Xeon 2.4 GHz CPU, 64-bit, 66 MHz PCI bus, 1024 MB DDR RAM main memory with CERN Red Hat Linux 7.1 running CTrig-01-15-12) and the MPRACE board. Only one CPU has been used by the algorithm (see Chapter 5).

Only initial track finding and maximum finding steps of the TRT LUT-Hough algorithm was implemented in VHDL. The software was compiled with GNU gcc-2.95. The Mentor Graphics LeonardoSpectrum and the Xilinx ISE4 software were used for synthesis and place and route. The FPGA design

Tab. 6.1: FPGA resources utilization summary.

Resource	**Used**	**Percentage**
Number of External IOBs	283 out of 684	41%
Number of RAMB16s	95 out of 96	98%
Number of SLICEs	10131 out of 14336	70%
Number of BUFGMUXs	2 out of 16	12%

is synchronised by the 64 MHz clock signal. FPGA (XC2V3000-4BF957C) resources utilization is summarized in the Table 6.1.

A data file (Y00347_1.atrdmp.data) containing approx. 156 single particle events with a pile-up at luminosity $10^{33}\,cm^{-2}s^{-1}$ was used.

Identical results are obtained for the initial track finding step for both the CPU and FPGA implementations.

The TRT consists of two subdetector types, the barrel and the end-cap. Both of them measure in two dimensions, the barrel in $r-\phi$ and the end-cap in $z-\phi$. The algorithms for barrel and end-cap are differ, but the principles are identical. For the performance measurements presented here only the barrel algorithm is used. The TRT barrel is composed of two identical (in the scope of this document) parts, the left and right barrel half. The following numbers refer to one half barrel.

The total number of straws is 52 500. The resolution required for the Initial Track Finding is defined by 1024 bins in ϕ space and 80 bins in $1/p_{\mathrm{T}}$ space leading to a total search space of 81 920 patterns. The information about barrel symmetry is not used (it was not included in simulation). Measurements results are shown in Table 6.2.

From these results one can see that a critical part is implemented in VHDL and runs on the FPGA co-processor ∼4 times faster than on the more or less modern CPU and the whole algorithm runs ∼2 times faster.

For even higher speed-up, the "Track Splitting" part of the algorithm should be implemented in the FPGA as well. We expect that the FPGA realisation of "Track Splitting" gives at least the same speed-up as "Histogramming / Maximum Finding". In this case we will have "Track Splitting" done in ∼285 μs instead of 1 246 μs and a full extract time will be ∼1 350 μs. Therefore, the hybrid implementation will be by a factor of ∼3.2 faster than the CPU-only implementation.

One can increase the SRAM word length to 288 bits by adding memory expansion mezzanine boards. In this case only 4 passes instead of 7 will be needed for transfer all necessary information for initial track finding and local maximum finding from SRAM to FPGA (for track splitting – 3 passes instead of 6). That gives additional speed-up factor close to two.

Tab. 6.2: Execution times on a Xeon 2.4 GHz PC for the TRT LUT-Hough CPU-only and hybrid implementation.
The total time is not the sum of the average times shown in the table, but rather is the average of the total times per event.
Number of events: 156
Average number of hits in event: 2 167.413
CPU: Pentium-IV XEON 2.4 GHz
FPGA board: MPRACE (VIRTEX-II), 64 MHz, 64-bit, 66 MHz PCI
Pattern-ordered LUT with 80 $1/p_{\mathrm{T}}$ *and 1024* ϕ *values predefined*
Threshold: $N_{\mathrm{thr}} = 12$*; Isolation:* $N_{\mathrm{is}} = 9$
Average number of track candidates per event: 57

Task	Platform	Time(μs)
CPU-only Implementation		
Fill event	CPU	107.77
Histogramming	CPU	2314.03
Threshold/Max Find	CPU	376.76
Histo/Thresh/Max	**CPU**	**2694.72**
Track Splitting	CPU	1246.18
Track Fitting	CPU	151.91
Final Selection	CPU	25.05
Copy tracks	CPU	145.62
Extract time	**CPU**	**4371.24**
Hybrid Implementation		
Fill event	CPU	115.23
Data converting	CPU	24.62
prepare write buffer	CPU	81.55
DMA write/Histogramming	PCI/FPGA	433.04
prepare read buffer	CPU	23.44
Max Finding/DMA read	FPGA/PCI	22.42
output formatting	CPU	17.71
Histo/Thresh/Max	**FPGA/CPU**	**612.77**
Track Splitting	CPU	1274.87
Track Fitting	CPU	155.54
Final Selection	CPU	24.80
Copy tracks	CPU	149.71
Extract time	**FPGA/CPU**	**2332.92**

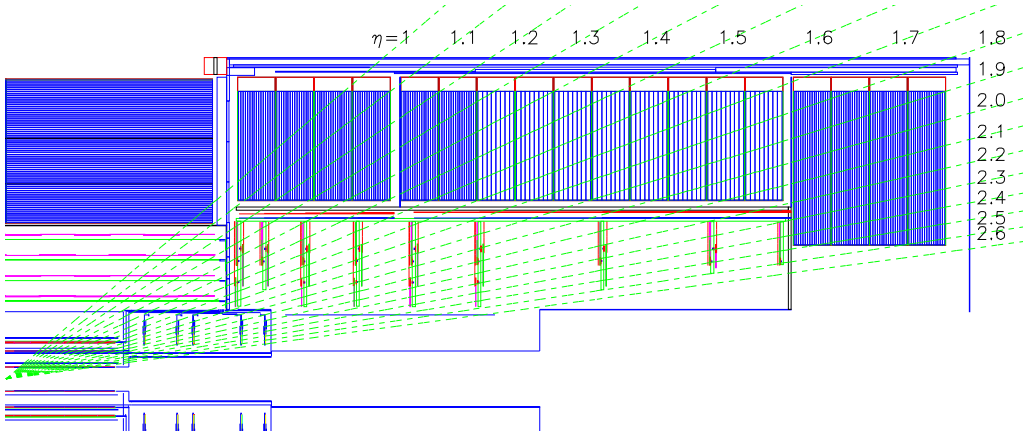

Fig. 6.9: View of one quarter of the ATLAS Inner Detector in the (R, z) plane.

Additional acceleration can be achieved by using a newer FPGA like Xilinx Virtex4. The VHDL design for initial track finding was slightly modified and synthesised for the Virtex4 family using Synplicity Synplify 8.0 and Xilinx ISE 7.1. After the place and route process the estimated clock frequency for the design core is ∼180 MHz. This leads to addition speed-up by factor ∼3 and to total speed-up factor close to 10. A co-processor board with Xilinx Virtex4 is currently under development by the FPGA processor group at the department of Computer Science V from the University of Mannheim.

6.4 TRT LUT-Hough in HLTSSW

The TRT LUT-Hough algorithm was designed for the ATLAS Level-2 Reference Software and was ported in to the CTrig (without big changes) and to the High Level Trigger Selection Software which suppose to be the final design of software for the ATLAS High Level trigger (see section 3.3). The algorithm was adopted for real (so called DC1 and DC2) geometry. It becomes possible to configure the algorithm via ATHENA job option files, to control it by the common trigger steering component, to get input data (hits information) from ATHENA services and to store results of work in the common store (TES). A part of the algorithm for reading the TRT geometry was changed to get geometry information from the ATHENA services instead of the ASCII text file. The algorithm becomes a part of ATHENA and it gives us a possibility to check physics performance of the TRT LUT-Hough algorithm in conditions very close to real (according to current view on ATLAS High Level trigger architecture).

The TRT LUT-Hough algorithm was tested with the simulated data for

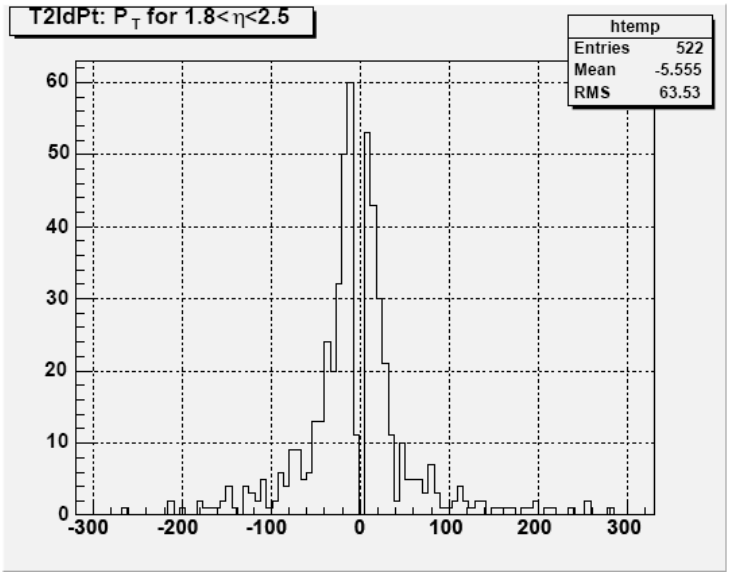

Fig. 6.10: Single 20 GeV electron events without pile-up. p_{T} distribution for reconstructed tracks with $|\eta| > 1.8$

high-p_{T} single particle events (p_{T} equal to 20 and 30 GeV) with and without pile-up in the ATHENA environment. These tests have showed that the track reconstruction quality varies in different parts of the TRT and is far from the ideal in case of events with pile-up (even single particle). Works for improving the recognition quality of the TRT LUT-Hough algorithm was done by the author and has been described in the rest of this thesis.

6.4.1 Magnetic Filed

One of the possible sources of the errors in reconstruction is an inhomogeneous magnetic field. The worst quality of reconstruction is observed in the outer part of TRT end-cap. p_{T} distribution for reconstructed tracks in this part for single 20 GeV electron events without pile-up is shown in the Figure 6.10. In this detector region the magnitude of the magnetic field is only 30% of the magnitude in the central part of the detector (see section 1.1.1). But the TRT LUT-Hough algorithm uses a constant field approximation. In a constant field, the trajectory of a charged particle of a given momentum has a constant curvature and can be represented by a straight line after some conformal transformation. The effect of the solenoidal magnetic field for the TRT tracking algorithm has been studied in [45]. In case of the solenoidal field the track curvature changes and in some regions the curvature may even changes the sign. As an example, one can see the trajectories of muons

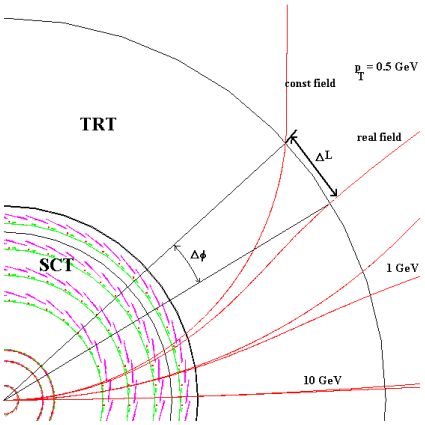

Fig. 6.11: Trajectories of the particles with various transverse momenta through the Inner Detector. $R-\phi$ view.

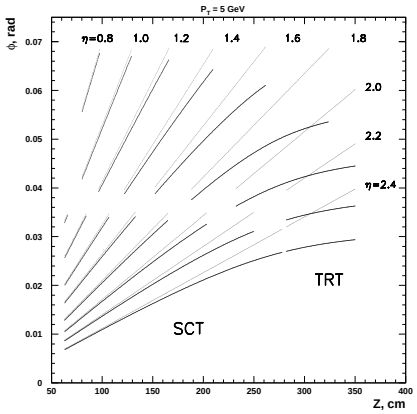

Fig. 6.12: Trajectories of muons with transverse momenta of 5 GeV for cases with a constant and a solenoidal magnetic fields. A view in a $\phi - z$ plane at various η.

in the (R, ϕ) plane in the Figure 6.11. The Figure 6.12 shows $\phi - Z$ projections of trajectories of muons with transverse momentum $p_{\mathrm{T}} = 5$ GeV in the end-cap region of the Inner Detector. The most significant deviation of trajectories from a straight line (i.e. constant field case) is at $\eta \sim 1.8$ (the layout of one quarter of the ATLAS Inner Detector in the (R, z) plane with η-lines is shown in the Figure 6.9; TRT is shown in blue). For $\eta > 2.2$ the part of the trajectory crossing the low-radius part of the outer four TRT wheels is an almost straight line, but its slope is different from that in the case of the nominal constant field.

The degradation of performance is larger for higher momenta. The reasons are obvious: the measurements of the track position in the region of inhomogeneous field are less precise, since the real field bending power is lower than for the constant field case.

The consequence of using the constant field approximation in the histogramming step of the algorithm is that in the real field some points will be lost as they lie outside the search road. However, it is certainly not favourable to widen the search road in the high-occupancy conditions expected at high luminosity. For high-p_{T} tracks (p_{T} above 5 GeV), the average number of hits, N_{hits}, is only slightly lower at $\eta = 1.7 - 2.0$ than in the case of the constant

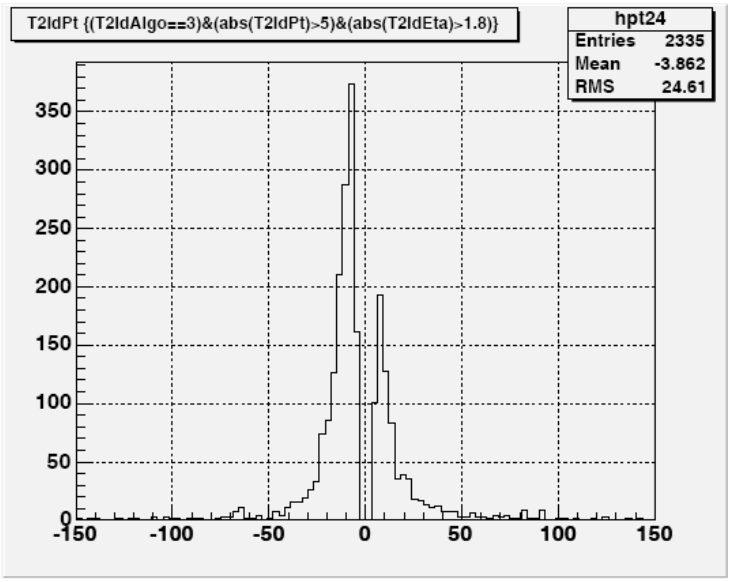

Fig. 6.13: Single 20 GeV electron events without pile-up. Distribution of p_{T} for reconstructed tracks with $|\eta| > 1.8$. LUT calculation done in case with an inhomogeneous magnetic field

field, while for the low-p_{T} tracks ($p_{\mathrm{T}} \leq 1$ GeV) there is a substantial loss of hits at mid-rapidity.

To recover from these losses, the search roads (and the look-up table) have to be calculated using the expected actual trajectories of the charged particles. The fact that the correction factors depend (approximately) only on the z-position of the straws (according to [45]) makes this correction simpler. As it was mentioned in section 6.1, at the histogramming stage we are searching for sets of hit straws satisfying the following straight line conditions:

$$\phi = \phi'_0 + C_z z;\ \phi'_0 = \phi_0 - z_0 C_z;\ C_z = 0.003 \cdot B \cdot q \cdot \tan(\theta)/2p_{\mathrm{T}},$$

where z_0, ϕ_0, θ are the initial z-coordinate, azimuthal and polar angles of the particle trajectory with an electric charge q in a field of strength B (B in Tesla, p_{T} in GeV, angles in radians). The correction function should be applied at the histograming stage to obtain a search road with a modified "slope":

$$C_z^{\mathrm{solen}} = \frac{0.003 \cdot q}{2p_{\mathrm{T}}} \int_0^L \mathbf{B}_{\mathrm{solen}}(r,z) \times d\mathbf{l}_{\mathrm{solen}} \approx C_z \cdot f_{\mathrm{corr}}(z),$$

where $f_{\mathrm{corr}}(z)$ can be found from the ratio between the change in azimuthal angle in the solenoidal magnetic field,$\Delta\phi_{\mathrm{solen}}$, and in the constant magnetic

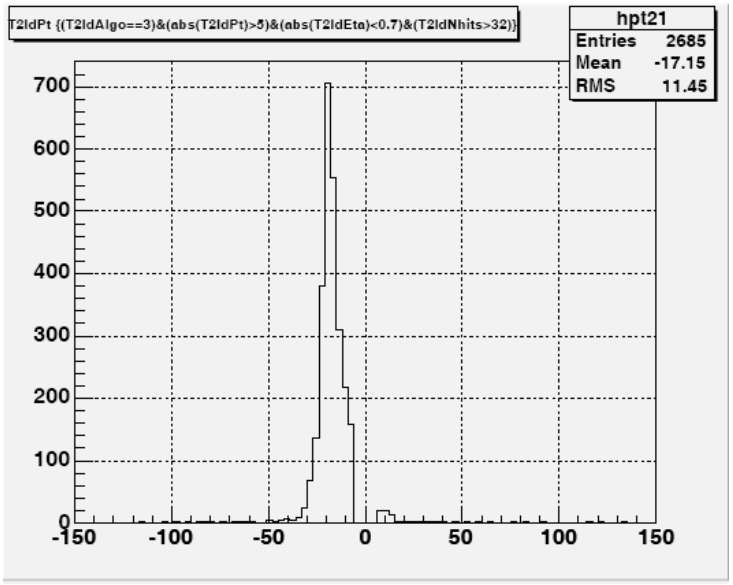

Fig. 6.14: Single 20 GeV electron events without pile-up. *Distribution of* p_{T} *for reconstructed tracks with* $|\eta| < 0.7$. *A number of generated events: 2758 (Efficiency: 97%)*

field, $\Delta\phi_{\mathrm{const}}$:

$$\frac{\Delta\phi_{\mathrm{solen}}}{\Delta\phi_{\mathrm{const}}} = \frac{\int B_{z,\mathrm{solen}}(z) dl_{\mathrm{solen}}}{\int B_{z,\mathrm{const}} dl_{\mathrm{const}}} = f_{\mathrm{corr}}(z)$$

The algorithm part for the look-up table generation was changed to be able to calculate the LUT for the case with an inhomogeneous magnetic field (information about magnetic field taken from ATHENA). It slightly improves the algorithm performance for the outer part of the TRT end-cap (see the Figure 6.13).

For the barrel part of TRT, where the magnetic field is close to be constant, the algorithm shows good physics performance (efficiency (calculated as ratio between number of reconstructed and number of generated tracks) >90%) for single particle events without pile-up. But in the case of events with pile-up it gives a lot of fake tracks. For example, a p_{T} distribution for reconstructed tracks in the barrel for single 20 GeV electron events without pile-up is shown in the Figure 6.14. The same distribution for single 20 GeV electron events with a pile-up at luminosity 2×10^{33} is shown in the Figure 6.15.

Two sources of uncorrelated tracks are important in the search for isolated leptons [112]: the real tracks from the underlying pile-up and the fake tracks from the random association of hits and track segments.

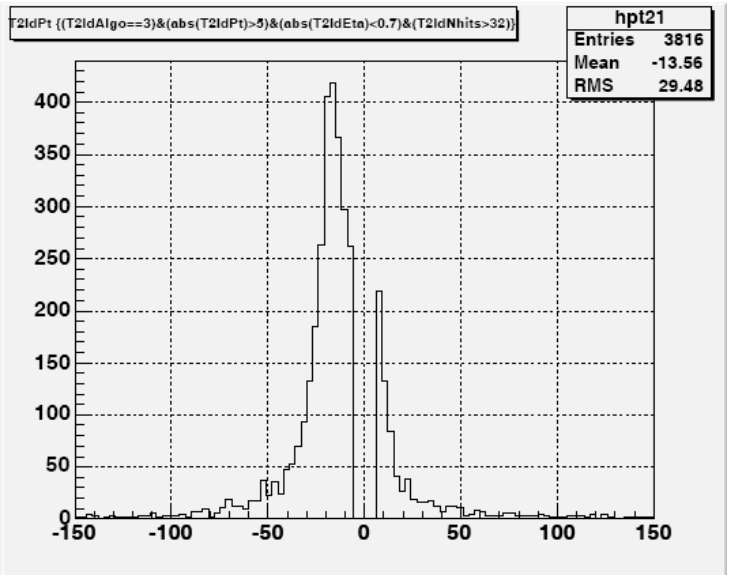

Fig. 6.15: Single 20 GeV electron events with pile-up at luminosity 2×10^{33}. *Distribution of* p_{T} *for reconstructed tracks with* $|\eta| < 0.7$. *A number of generated events: 1402*

6.4.2 Track merging

The TRT LUT-Hough algorithm gives in results a high number of track candidates per event. For example, the Figure 6.16 shows the distribution of the number of reconstructed tracks per event. There we can see that for 69 single particle events in the barrel and end-cap the algorithm has found 379 tracks only in the barrel (an average of ∼20 tracks per event). One of the possible reasons is following.

Due to physics processes in the Inner Detector (Bremsstrahlung), tracks in the TRT may not point to the origin. These tracks are reconstructed as several track segments none of which contain all active straws belonging to that track. Several reconstructed tracks from one generated track are in most cases neighbours in the resulted tracks list and they share most of the hits. Therefore a very simple technique for such "ghosts" removing can be used: the tracks with more than a predefined number of common hits should be merged. To improve execution speed in the barrel one can just select the track with the biggest number of hits from the set of tracks with common hits instead of merging. However, this cannot be done in the end-cap, where the end-points of the track influence the η and p_{T} measurements. Missing hits can seriously degrade the reconstructed track parameters. Results of track segments merging in the TRT barrel are shown in the Figure 6.17. Here we

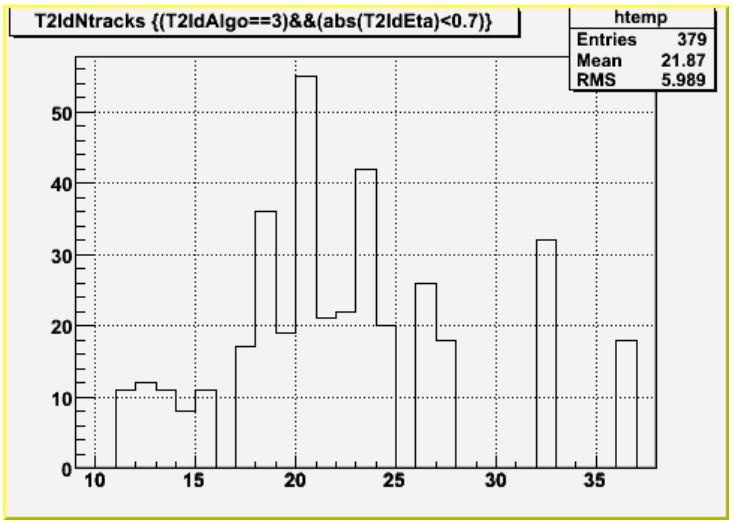

Fig. 6.16: A number of reconstructed tracks in TRT barrel. *Tested on 69 events.*

can see the reasonable amount of 26 reconstructed tracks in the barrel (one or two tracks per event mostly).

6.4.3 Likelihood approach

The default parameters used in the algorithm are loose enough to be able to reconstruct different kinds of particles with a wide range of the p_{T} in the whole accessible η-region. At the design luminosity the occupancy of TRT is between 10% and 40%, the higher values are for the inner layer of the barrel and the long straws in the end-caps. The large number of straws with hits means that there is a significant probability for a "fake" track candidate to be formed by a combination of hits from tracks in the minimum bias events. As a result, the algorithm produces several candidates per real track. The majority of track candidates in events with pile-up are due to real low-p_{T} tracks. However, the p_{T} threshold cannot be set very high even if we are looking for high-p_{T} tracks (electrons) because of the bremsstrahlung effect. One of the methods to choose the best candidate (so called "likelihood approach") was described in [113]. The short description of this method is presented here.

Consider the reconstructed track which crossed N_{s} straws and with number of hits N_{hit}. The likelihood approach is based on a calculation of the ratio $L_{\mathrm{t}}/L_{\mathrm{b}}$ of likelihoods for the hypothesis that hit is produced by the real track:

$$L_{\mathrm{t}} = \binom{N_{\mathrm{s}}}{N_{\mathrm{hit}}} \cdot P^{N_{\mathrm{hit}}} \cdot (1-P)^{(N_{\mathrm{s}}-N_{\mathrm{hit}})}$$

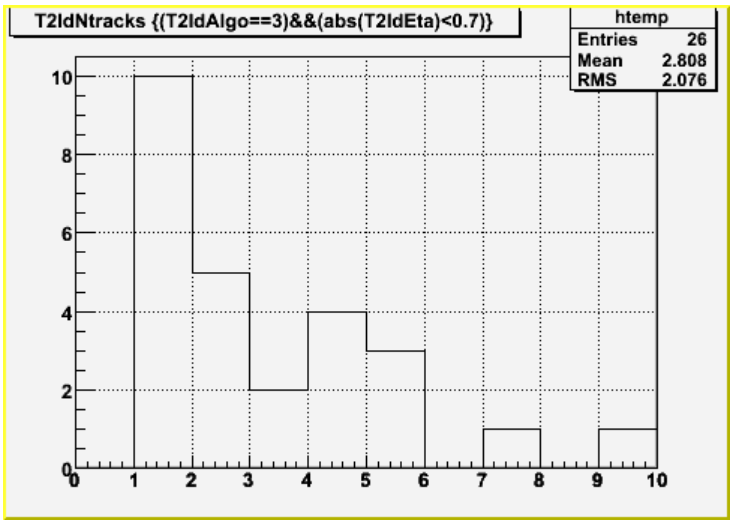

Fig. 6.17: Number of reconstructed tracks in TRT barrel after merging of the track segments with more then 75% of common hits. *Tested on 69 events*

or by a minimum bias or a noise:

$$L_{\mathrm{b}} = \binom{N_{\mathrm{s}}}{N_{\mathrm{hit}}} \cdot P_{\mathrm{b}}^{N_{\mathrm{hit}}} \cdot (1 - P_{\mathrm{b}})^{(N_{\mathrm{s}} - N_{\mathrm{hit}})}$$

where $P = P_{\mathrm{b}} + P_{\mathrm{t}} \cdot (1 - P_{\mathrm{b}})$ is the probability of a hit on a track, P_{t} is the probability of a hit caused by the real particle ($\sim$0.95) and P_{b} is the probability of a hit from a minimum bias or noise ($\sim$0.15-0.5 depending on straw length and position). Then the best estimator for the track is that which maximises the L defined as :

$$L(N_{\mathrm{hit}}, N_{\mathrm{hole}}) = \log(L_{\mathrm{t}}/L_{\mathrm{b}}) = w_1 \cdot N_{\mathrm{hit}} + w_2 \cdot (N_{\mathrm{s}} - N_{\mathrm{hit}})$$

where weights $w_1 = \log(P/P_{\mathrm{b}})$ and $w_2 = \log((1 - P)/(P_{\mathrm{b}}))$ can be estimated from the experimental data for the real detector or from a Monte-Carlo simulation. This estimation was done for TRT and results can be found in [109]. Similar values can be calculated for the likelihood $L(N_{\mathrm{TR}}, N_{\mathrm{s}})$ (in the case of electron track), where N_{TR} is a number of transition-radiation hits (hits over the higher threshold on read-out electronics) and for the likelihood $L(N_{\mathrm{time}}, N_{\mathrm{s}})$ (in case when drift time information is used). In the TRT LUT-Hough algorithm the drift time information is not used and the best discriminator for electron tracks is:

$$L(N_{\mathrm{s}}, N_{\mathrm{hit}}, N_{\mathrm{TR}}) = w_{\mathrm{h}} \cdot N_{\mathrm{hit}} + w_{\mathrm{s}} \cdot N_{\mathrm{s}} + w_{\mathrm{TR}} \cdot N_{\mathrm{TR}}$$

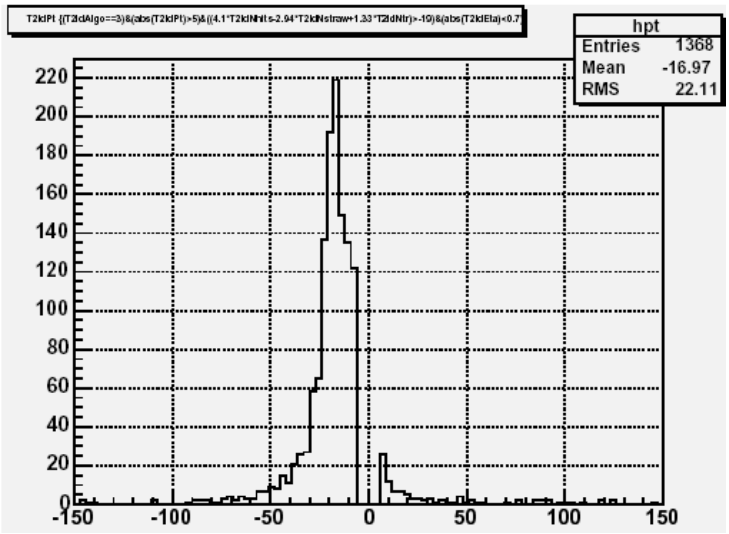

Fig. 6.18: Single 20 GeV electron events with pile-up at luminosity $2 \times 10^3 3$. p_{T} distribution for reconstructed tracks with $|\eta| < 0.7$. A number of generated events with $|\eta| < 0.7$: 1402 (Efficiency: 97.5%)

This method was tested with the TRT LUT-Hough algorithm. Weights w_{h}, w_{s}, and w_{TR} are calculated for four separate $|\eta|$ regions to allow the variation in occupancy and probabilities P_{t} and P_{b}. Results for the TRT LUT-Hough algorithm with the new track selection procedure (p_{T} distribution for reconstructed tracks in barrel for single 20 GeV electron events with pile-up at luminosity $2 \times 10^3 3$) are shown on the Figure 6.18.

For more improvements in the physics performance and quality of the track reconstruction in the TRT the track following approach was developed by D. Emelyanov at RAL. Tracks from Pixel Detector and SCT are used as seeds for the track extrapolation into the TRT [114]. To minimize amount of the data to be analysed in the extrapolation step one can use a pre-calculated Look-Up table in the same way like it is done in the TRT LUT-Hough algorithm. For testing of this idea the new AlgTool was developed by the author. This tool takes tracks parameters and returns the set of identifiers for straws with hits which belong to the number of predefined roads from the LUT. This approach shows a very good results for the standard RoI based reconstruction. However, this approach can not be used for B-physic reconstruction with the initial TRT full scan and seeding from the TRT (as soon as it require an external seed for track reconstruction in TRT).

6.4.4 Results of the review of the LVL2 Inner Detector algorithms

A review of the LVL2 Inner Detector algorithms took place in July/August 2005. The aim was to review the current status of the LVL2 trigger algorithms for the Inner Detector, including the Event Data Model and use of the geometry information, and to draw up a workplan to consolidate the software into a set of tools to be used in the final system.

The following comments from reviewers related to the algorithmic work done in this thesis were received:

"Complementary approaches of SiTrack vs. IDscan and TRTxK/LUT vs. IDscan+extrapolation to the TRT are worth keeping as an option to cope with unexpected initial detector problems. Proposed to continue to develop alternative approaches, but in the context of tools with common interfaces. Retain TRT stand-alone reconstruction, where effort allows, as a tool to be used in commissioning or when there are detector problems in early running. TRTxK/LUT require quite some work to meet LVL-2 timing requirements as well as the data preparation steps for the TRT" [115].

The full text of the LVL2 Inner Detector Reviewers Comments can be found on the web page with the following address:
https://uimon.cern.ch/twiki/bin/view/Atlas/TrigInDetRevComment

According to decisions made during the review of the LVL2 Inner Detector algorithms, all algorithms should be re-packaged as a set of track finding tools based on the global approach for the pattern recognition: histogramming, LUT; track fitting tools; tools for the simultaneous pattern recognition and the track fitting – the local pattern recognition. New algorithms will use these tools to perform the track reconstruction. The tools can also be used in special monitoring/debugging algorithm wrappers. The first step is to define abstract interfaces for the tools currently ongoing. This approach allows to use the strongest parts of the algorithms in various combinations, for example, the pattern recognition from one algorithm can be combined with the fitter from another or even with the offline fitter.

Chapter 7

FUTURE WORKS FOR THE ATLAS TRIGGER

According to decisions made during the review of the LVL2 Inner Detector algorithms, TRT stand-alone reconstruction should be retained.

However, there are still a lot of thing to do, including:

Repackage LVL2 ID reconstruction as sets of tools. Existing reconstruction algorithms should be splitted on track finding tools based on the global approach for the pattern recognition: histogramming, LUT; track fitting tools; tools for simultaneous pattern recognition and track fitting – the local pattern recognition, also known as "track following". New algorithms will use these tools to perform a track reconstruction. These tools can also be used in the special monitoring and debugging algorithm wrappers (see the end of previous Chapter).

Timing optimization (including data preparation step): One of the general timing problem: the ByteStream-to-cluster conversion is done for the whole ROB (see section 3.3). ROBs most of the time contain data from many more *Detector Elements* than algorithms are interested in (the ones inside an RoI). Significant speed-up can be achieved by not unpacking everything. Works for improvement of the data preparation step has been started [116]. The RoI size and shape can be (and has to be) optimized as well.

Test functionalities: Trigger algorithms (and whole Trigger software) should fit into "ATLAS Software Testing scheme". The data set that produces a reference result (histograms, rates, etc.) and acceptance mechanism should be developed for every algorithm. The first common ATLAS frameworks for testing already exist (for example Run Time Tests: http://www.hep.ucl.ac.uk/atlas/NightlyTests/RTTpage1.html). Test suites, reference datasets and reference results for trigger algorithms

should be created. Tools for Data Quality Monitoring and for identification of cabling and conversion problems should be developed. One should think about tests with various simulated errors as well.

Integration with on-line software: Algorithms should be able to run in multi-threaded environment (AthenaMT), support re-initialization and fully conform to other on-line software requirements. Therefore some work for integration of trigger software with on-line framework should be done.

Connections with off-line software: One should compare LVL2 requirements for tools interfaces and Event Data Model classes with off-line requirements and investigates the possibility of using common base classes. The geometry information used by LVL2 algorithms should be reviewed in order to provide this information in the most efficient way, either directly from the Geomodel description (off-line) or from a LVL2-specific cache.

Alignment: Trigger algorithms should be tested against an alignment, calibration, beam spot position and beam axis tilt. First studies of the misalignment effects have been done some times ago [117] but more detailed investigations (trigger performance as a function of misalignment, trigger performance as a function of dead/inefficient detectors/read-out channels) in the new software environments should be done. Sets of algorithms parameters for various initial conditions should be developed.

DataBase configuration: Currently algorithms parameters are stored in so called "JobOptions files". Information stored in form of python scripts which are used by Athena and steering code for algorithms configuration. But it is not easy to maintain this configuration information. Trigger algorithms should be configurable from the configuration database. This work is related to changes in the ATLAS High Level Trigger Selection software (and probably to changes in the off-line software framework).

Cosmic run: Algorithms should be prepared (and, probably, slightly modified) to be used in the cosmic run. Some preparatory work on the cosmic running can be started once Monte-Carlo data will be available. Special algorithms should be developed and tested as well (see, for example, [118])

Implementation of Trigger menus: Existed algorithms should be grouped and configured for creating a complete trigger menu [80, 119]. Hypothesis

algorithms (in addition to feature extraction algorithms) should be created as well as different menu configurations should be available. A set of algorithms for different trigger tasks (like e/gamma, tau, jets etc.) should be selected and tested. One should have a look at configuration of algorithms for a full menu to see whether separate configuration parameters are needed or whether common parameters can be used in some cases. Histograms for monitoring track reconstruction performance and reference histograms for each of the slices should be defined.

Some tools for matching tracks from different algorithms should be created. These tools should use combination of information from different algorithms to improve identification of track parameters.

Tools for Secondary Vertex reconstruction at LVL2 should be developed.

Strategies for measuring the trigger efficiencies with the real data (without Monte-Carlo information) should be created.

All trigger menus should be tested on simulated data and efficiencies and rates should be estimated for the full menu, not just for one algorithm.

CONCLUSIONS

A possible reduction of the algorithms execution time has a large impact on the size of LVL2 trigger processing farm. In the scope of this thesis one of the possible approaches to acceleration the tracking algorithms using the hybrid FPGA/CPU systems has been investigated and tested on the simulated single particle events ($B \rightarrow \mu X$) with a pile-up at luminosity $10^{33}\, cm^{-2}s^{-1}$. The most time consuming part of the algorithm has been accelerated by the FPGA platform. The usage of the FPGA co-processor can give some reasonable speedup as contrasted to the general purpose processor only for those algorithms (or parts of algorithms), for which one there is a possibility to fulfil calculations with a major degree of parallelism. Some of the track reconstruction algorithms which are common to all B-physics channels and the standard RoI processing have been tested for execution time and assessed for suitability for speed-up by using FPGA co-processor in scope of this thesis. One from these algorithms – TRT LUT-Hough – utilizes very popular method in track reconstruction: look-up table based Hough transform (histogramming). Most time consuming parts of it were implemented in VHDL for running on the FPGA co-processor board MPRACE. Our work shows that the use of the FPGA co-processor can give us speed-up by factor ∼2-3 for the hybrid FPGA/CPU realisation in comparison with the CPU only implementation.

Algorithm realisation described in this thesis has exacting requirements on a FPGA co-processor board, especially its memory subsystem. Higher speed-up (close to factor 10) can be achieved by using a co-processor with a higher memory bandwidth (in the case of the MPRACE board one can use both extension slots for two additional SRAM banks) and a newer FPGA (or even several FPGAs on one board with own memory).

To spend no time for the data transfer between algorithm host and co-processor, it is better to place FPGA board on the raw datapath from detectors to the trigger PC or trigger farm (PC or farm which will be used for trigger algorithms), like it is planed to be done in ALICE (see section 2.2.3). However this approach requires more complicated and specialized co-processor board and FPGA design. According to [120] ALICE col-

laboration has a Hough Transform based HLT algorithm for track finding in TPC and has plans to use FPGAs for algorithm acceleration. TPC provides a three-dimensional information, but its data can be splitted in bins of pseudo-rapidity and two-dimensional Hough Transform can be used. Following Hough space binning is used ([120]): 100 η bins divided on 80×120 bins each. Algorithm is similar to TRT LUT-Hough and some parts of the VHDL design which has been developed in the scope of this thesis can be reused. ALICE realization requires lower memory bandwidth as soon as 2D Hough space has 80×120 bins (compare to 80×1024 in TRT LUT-Hough), therefore, one can expect a higher speed-up.

Developers in high-performance computing area long time have been intrigued by the potential of reconfigurable computing to accelerate some computationally-intensive applications. But the barriers to achieving the performance gains that reconfigurable computing can theoretically provide are well known: the complexity of programming for reconfigurable computing devices and the relatively long hardware development circle. To be used a FPGA co-processor and a configuration bitstream (algorithm implementation) for it should be designed and tested at least one year before the beginning of the experiment.

From other side, commercial "off-the-shelf" (COTS) components have a number of advantages compared to custom electronics components:

- Parts of the system can be exchanged by more powerful components without a redesign of electronics. This requires the compatibility of new components which is also a widespread issue in industrial applications.

- The components can be purchased in a short time. Thus recent hardware can be used at the experiment's start-up.

- The components are more cost effective then custom hardware.

Recently, the ATLAS TDAQ community has decided not to use custom hardware in the trigger (except LVL1) and software developers have focused on fine tuning applications to run faster on standard microprocessors.

The TRT LUT-Hough algorithm was ported into software framework (High Level Trigger Selection Software) which will run in a final ATLAS trigger system. Improvements in the track candidates selection procedure have been integrated into the final version of the TRT LUT-Hough for higher quality of reconstruction. For additional improvements more sofisticated fitter can be used. ALICE team ([120]) has interesting idea to count not a hits but a gaps along the track trajectory during histogramming stage and to take as a final track candidate a peak, but with number of gaps less then

predefined threshold for decreasing the number of fake tracks. It would be interesting to have a look how this approach can be used for ATLAS TRT.

BIBLIOGRAPHY

[1] Gordon Kane. *Modern Elementary Particle Physics – The fundamental Particles and Forces.* Addison-Wesley, 1993.

[2] Fritz W. Bopp. *Kerne, Hadronen und Elementarteilchen. Eine Einführung.* B.G. Teubner, 1989.

[3] Peter Ware Higgs. Broken symmetries, massless particless and gauge fields. *Physics Letters*, 12(2):132–133, 1964.

[4] Peter Ware Higgs. Broken symmetries and masses of gauge bosons. *Physical Review Letters*, 13(16):508–509, 1964.

[5] Peter Ware Higgs. Spontaneous symmetry breakdown without massless bosons. *Physical Review*, 145(4):1156–1163, 1966.

[6] ALEPH Collaboration, DELPHI Collaboration, L3 Collaboration, OPAL Collaboration, and The LEP Working Group for Higgs Boson Searches. Search for the Standard Model Higgs boson at LEP. *Physics Letters*, B565:61–75, 2003.

[7] ALEPH Collaboration, DELPHI Collaboration, L3 Collaboration, OPAL Collaboration, The LEP Electroweak Working, and The SLD Heavy Flavour Working Group. A combination of preliminary electroweak measurements and constraints on the Standard Model. Technical report, CERN/EP 2002-091, 2002.

[8] S. Eidelman, K.G. Hayes, K.A. Olive, M. Aguilar-Benitez, C. Amsler, D. Asner, K.S. Babu, R.M. Barnett, J. Beringer, P.R. Burchat, C.D. Carone, C. Caso, G. Conforto, O. Dahl, G. D'Ambrosio, M. Doser, J.L. Feng, T. Gherghetta, L. Gibbons, M. Goodman, C. Grab, D.E. Groom, A. Gurtu, K. Hagiwara, J.J. Hernández-Rey, K. Hikasa, K. Honscheid, H. Jawahery, C. Kolda, Kwon Y., M.L. Mangano, A.V. Manohar, J. March-Russell, A. Masoni, R. Miquel, K. Mönig, H. Murayama, K. Nakamura, S. Navas, L. Pape, C. Patrignani, A. Piepke, G. Raffelt, M. Roos, M. Tanabashi, J. Terning, N.A. Törnqvist, T.G. Trippe,

P. Vogel, C.G. Wohl, R.L. Workman, W.-M. Yao, P.A. Zyla, B. Armstrong, P.S. Gee, G. Harper, K.S. Lugovsky, S.B. Lugovsky, V.S. Lugovsky, A. Rom, M. Artuso, E. Barberio, M. Battaglia, H. Bichsel, O. Biebel, P. Bloch, R.N. Cahn, D. Casper, A. Cattai, R.S. Chivukula, G. Cowan, T. Damour, K. Desler, M.A. Dobbs, M. Drees, A. Edwards, D.A. Edwards, V.D. Elvira, J. Erler, V.V. Ezhela, W. Fetscher, B.D. Fields, B. Foster, D. Froidevaux, M. Fukugita, T.K. Gaisser, L. Garren, H.-J. Gerber, G. Gerbier, F.J. Gilman, H.E. Haber, C. Hagmann, J. Hewett, I. Hinchliffe, C.J. Hogan, G. Höhler, P. Igo-Kemenes, J.D. Jackson, K.F. Johnson, D. Karlen, B. Kayser, D. Kirkby, S.R. Klein, K. Kleinknecht, I.G. Knowles, P. Kreitz, Yu.V. Kuyanov, O. Lahav, P. Langacker, A. Liddle, L. Littenberg, D.M. Manley, A.D. Martin, M. Narain, P. Nason, Y. Nir, J.A. Peacock, H.R. Quinn, S. Raby, B.N. Ratcliff, E.A. Razuvaev, B. Renk, G. Rolandi, M.T. Ronan, L.J. Rosenberg, C.T. Sachrajda, Y. Sakai, A.I. Sanda, S. Sarkar, M. Schmitt, O. Schneider, D. Scott, W.G. Seligman, M.H. Shaevitz, T. Sjöstrand, G.F. Smoot, S. Spanier, H. Spieler, N.J.C. Spooner, M. Srednicki, A. Stahl, T. Stanev, M. Suzuki, N.P. Tkachenko, G.H. Trilling, G. Valencia, K. van Bibber, M.G. Vincter, D. Ward, B.R. Webber, M. Whalley, L. Wolfenstein, J. Womersley, C.L. Woody, O.V. Zenin, and R.-Y. Zhu. Review of Particle Physics. *Physics Letters B*, 592:1+, 2004.

[9] LHC Study group. The Large Hadron Collider. Conceptual design report, CERN/AC 95-05, 1995.

[10] ALICE Collaboration. A Large Ion Collider Experiment - technical proposal. Technical report, CERN/LHCC 95-71, 1995.

[11] ATLAS Collaboration. ATLAS technical proposal. Technical report, CERN/LHCC 94-43, 1994.

[12] CMS Collaboration. The Compact Muon Solenoid - technical proposal. Technical report, CERN/LHCC 94-38, 1994.

[13] LHCb Collaboration. Letter of intent for a dedicated LHC collider beauty experiment for precision measurements of CP-Violation. Technical report, CERN/LHCC 95-5, 1995.

[14] TOTEM Collaboration. Total cross-section, elastic scattering and diffraction dissociation at the LHC. Letter of intent, CERN-LHCC-97-49, 1997.

[15] Lidija Zivkovic. Measurements of the Standard Model Higgs parameters at ATLAS. ATLAS Note ATL-PHYS-2004-023, CERN, 2004.

[16] Simonetta Gentile. Search for Higgs bosons with the ATLAS detector. ATLAS Note ATL-PHYS-2004-009, CERN, 2004.

[17] F. Ohlsson-Malek. Prospects of ATLAS and CMS for B Physics and CP Violation. ATLAS Note ATLAS-CONF-2003-003, CERN, 2003.

[18] Sebastien Viret. Rare B decays at LHC. ATLAS Note ATL-PHYS-CONF-2005-005, CERN, 2005.

[19] Frederic Kalen Martens. Top physics capabilities at the LHC. ATLAS Note ATL-PHYS-2004-024, CERN, 2004.

[20] George Stavropoulos. Top physics at ATLAS. ATLAS Note ATL-PHYS-2004-032, CERN, 2004.

[21] D.R. Tovey. Searches for new physics at the LHC. ATLAS Note ATL-CONF-2002-005, CERN, 2002.

[22] Gianluca Comune. SUSY with ATLAS: Leptonic signatures, coannihilation region. ATLAS Note ATL-PHYS-CONF-2005-003, CERN, 2005.

[23] Kamal Benslama. Search for extra dimensions with ATLAS at LHC. ATLAS Note ATL-PHYS-2004-013, CERN, 2004.

[24] C. M. Harrisy, M. J. Palmery, M. A. Parkery, P. Richardsonz, A. Sabetfakhriy, and B. R. Webbery. Exploring higher dimensional black holes at the Large Hadron Collider. ATLAS Note ATL-PHYS-2004-033, CERN, 2004.

[25] F.M. Brochu. Search for extra-dimensions in the ATLAS experiment. ATLAS Note ATL-PHYS-CONF-2005-004, CERN, 2005.

[26] B. C. Allanach, K. Odagiri, M. A. Parker, and B. R. Webber. Searching for narrow graviton resonances with the ATLAS detector at the Large Hadron Collider. ATLAS Note ATL-PHYS-2000-029, CERN, 2000.

[27] Daniel R. Tovey. Measurement of the neutralino mass. ATLAS Note ATL-CONF-2003-005, CERN, 2003.

[28] Giacomo Polesello. Prospects for the detection of heavy charginos and neutralinos with the ATLAS detector at the LHC. ATLAS Note SN-ATLAS-2004-041, CERN, 2004.

[29] Calin Alexa. Heavy lepton physics in ATLAS. ATLAS Note ATL-PHYS-CONF-2005-001, CERN, 2005.

[30] Paul V. C. Hough. U.S. Patent 3 069 654, December 1962.

[31] J. Illingworth and J. Kittler. A survey of the hough transform. *Comput. Vision Graphics, Image Processing*, (44):87–116, 1988.

[32] Rudolf Frühwirth, Meinhard Regler, Rudolf K. Bock, Hans Grote, and Dieter Notz. *Data Analysis Techniques for High-Energy Physics*. Cambridge Monographs on Particle Physics, Nuclear Physics and Cosmology. Cambridge University Press, 2000.

[33] P.E.L.Clarke, R.Cranfield, G.J.Crone, B.J.Green, J.A.Strong, R.E.Hughes-Jones, S.Kolya, R.Marshall, D.Mercer, K.Korcyl, R.Hatley, R.P.Middleton, F.J.Wickens, and A.Guglielmi. SCI with DSPs and RISC processors for LHC 2nd level triggering. ATLAS Internal Note DAQ-No-19, CERN, 1994.

[34] Konrad Kleinknecht. *Detektoren für Teilchenstrahlung*. B.G. Teubner, 1987.

[35] ATLAS Collaboration. ATLAS detector and physics performance technical design report. Technical report, CERN/LHCC 99-15, 1999.

[36] ATLAS Collaboration. ATLAS Magnet System Technical Design Report. Technical report, CERN/LHCC 97-18, 1997.

[37] The ATLAS Muon Collaboration. Muon spectrometer technical design report. Technical report, CERN/LHCC 97-22, 1997.

[38] ATLAS Collaboration. Calorimeter Performance, Technical Design Report. Technical report, CERN/LHCC 96-40, 1996.

[39] ATLAS LARG Unit. Liquid Argon Calorimeter, Technical Design Report. Technical report, CERN/LHCC 96-41, 1996.

[40] ATLAS Tile Calorimeter Collaboration. Tile calorimeter, technical design report. Technical report, CERN/LHCC 96-42, 1996.

[41] ATLAS Inner Detector Community. ATLAS Inner Detector Technical Design Report. Technical report, CERN/LHCC 97-16, 1997.

[42] ATLAS Collaboration. Pixel detector technical design report. Technical report, CERN/LHCC 98-13, 1998.

[43] R. Bock and P. Le Dû. Detector and readout specifications, and buffer-RoI relations, for the Level-2 trigger demonstrator program. ATLAS Note ATLAS DAQ note 62, CERN, 1997.

[44] P. Clarke, S. Falciano, P. Le Dû, J.B. Lane, M. Abolins, C. Schwick, and F.J. Wickens. Detector and read-out specification, and Buffer-RoI relations, for Level-2 studies. ATLAS Note ATL-DAQ-99-014, CERN, 1999.

[45] Sergey Sivoklokov. TRT trigger performance in the solenoidal magnetic field. ATLAS Note ATL-DAQ-99-004, CERN, 1999.

[46] ATLAS HLT/DAQ/DCS Group. ATLAS High-Level Triggers, Data Acquisition and Controls Technical Design Report. Technical report, CERN/LHCC 2003-022, 2003.

[47] J. Baines, A. Baratella, B. Epp, S. George, V. M. Ghete, L. Guy, S. Gonzalez, D. Hutchcroft, W. Li, P. Morettini, A. Nairz, F. Parodi, S. Qian, F. Rizatdinova, D. Scannicchio, M. Sessler, P. Sherwood, S. Sivoklokov, and M. Smizanska. B-Physics event selection for the ATLAS High Level Trigger. ATLAS Note ATL-DAQ-2000-031, CERN, 2003.

[48] J. Bystrický, D. Calvet, J. Ernwein, O. Gachelin, Traudl. Hansl-Kozanecka, J.R. Hubbard, M. Huet, P. Le Dû, I.D. Mandjavidze, M. Mur, M. Smizanska, and B. Thooris. A sequential processing strategy for the ATLAS event selection. ATLAS Note ATL-DAQ-96-059, CERN, 1996.

[49] H. Bertelsen, G. Boorman, R. Cranfield, G.J. Crone, M. Dam, J. Dawson, E. Dénes, R.W. Dobinson, D. Francis, B. Green, J.R.H Hansen, P. Maley, B. Rensch, J.L Schlereth, and J. Strong. A local-global implementation of a vertical slice of the ATLAS second level trigger. ATLAS Note ATL-DAQ-98-081, CERN, 1998.

[50] A. Kugel, K. Kornmesser, R. Lay, R. Männer, K.-H. Noffz, S. Rühl, M. Sessler, H. Simmler, H. Singpiel, V. Dörsing, W. Erhard, P. Kammel, A. Reinsch, L. Levinson, R. Bock, W. Iwanski, K. Korcyl, J. Olszowska, D. Calvet, J. R. Hubbard, P. Le Dû, I. Mandzavidze, and M. Smizanska. ATLAS Level-2 Trigger Demonstrator-A activity report. part 1: Overview and summary. ATLAS Internal Note DAQ-NO-085, CERN, 1998.

[51] CDF-IIb Collaboration. The CDF-IIb detector: Technical design report. Technical report, FERMILAB-TM-2198, 2002.

[52] D0 Collaboration. D0 Run IIB upgrade technical design report. Technical report, FERMILAB-PUB-02-327-E, 2002.

[53] The BABAR Collaboration. The BABAR Detector. *Nuclear Instruments and Methods in Physics Research Section A*, 479:1–116, February 2002.

[54] The Belle Collaboration. The Belle Detector. *Nuclear Instruments and Methods in Physics Research Section A*, 479:117–232, February 2002.

[55] S. Baileya, R. Barlow, J. Boyd, G. Brandenburg, X. Chai, N. de Groot, N. Felt, G. Grenier, V. Halyo, S. Harder, O. Igonkina, W. Innes, M. Kelly, S. Kolya, S. Lee, U. Mallik, D. Mercer, M. Morii, J. Oliver, J. Olsen, N. Sinev, D. Su, E. Torrence, and E. Won. Rapid 3D track reconstruction with the babar trigger system.

[56] E. J. Thomson, C. Ciobanu, J. Y. Chung, J. Gerstenslager, J. Hoftiezer, R. E. Hughes, M. Johnson, P. Koehn, C. Neu, C. Sanchez, B. L. Winer, J. Dittmann, J. Freeman, S. Holm, J. D. Lewis, C. J. Lin, T. Shaw, T. Wesson, K. Bloom, D. Gerdes, N. Goldschmidt, J. Dawson, and W. Haberichter. Online track processor for the cdf upgrade. *IEEE Transactions on Nuclear Science*, 49:1063–1070, June 2002.

[57] B. Ashmanskas, A. Barchiesi, A. Bardi, M. Bari, M. Baumgart, S. Belforte, J. Berryhill, M. Bogdan, R. Carosi, A. Cerri, G. Chlachidze, R. Culbertson, M. Dell'Orso, S. Donati, I. Fiori, H. Frisch, S. Galeotti, P. Giannetti, V. Glagolev, A. Leger, Y. Liu, T. Maruyama, E. Meschi, L. Moneta, F. Morsani, T. Nakaya, G. Punzi, M. Rescigno, L. Ristori, H. Sanders, S. Sarkar, A. Semenov, M. Shochet, T. Speer, F. Spinella, H. Vataga, X. Wu, U.K. Yang, L. Zanello, and A.M. Zanetti. The cdf silicon vertex trigger. *Nuclear Instruments and Methods in Physics Research Section A*, 518:532–536, 2004.

[58] CMS Collaboration. Data Acquisition and High-Level Trigger Technical Design Report. Technical report, CERN/LHCC 2002-26, 2002.

[59] R. Carosi, G. Iannaccone, and G. Varotto. Real time track finding in CMS. CMS Internal Note CMS IN 2000/023, CERN, 2000.

[60] Marcel Vos. Tracking and b and τ tagging in the CMS high level trigger. CMS Conference Report CMS CR 2006/004, CERN, 2006.

[61] W. Adam, Th. Speer, B. Mangano, and T. Todorov. Track reconstruction in the CMS tracker. CMS Note CMS NOTE 2006/041, CERN, 2006.

[62] S. Cucciarelli, M. Konecki, D. Kotliński, and T. Todorov. Track parameter evaluation and primary vertex finding with the pixel detector. CMS Note CMS NOTE 2003/026, CERN, 2003.

[63] LHCb Collaboration. LHCb Trigger Technical Design Report. Technical report, CERN/LHCC 2003-031, 2003.

[64] Olivier Callot. VELO tracking for the High Level Trigger. LHCb Note LHCb 2003-027, CERN, 2003.

[65] Olivier Callot. Online pattern recognition. LHCb Note LHCb 2004-094, CERN, 2004.

[66] Jeroen van Tilburg. *Track simulation and reconstruction in LHCb.* PhD thesis, Vrije Universiteit Amsterdam, 2005.

[67] ALICE Collaboration. ALICE Technical Design Report of the Trigger, Data Acquisition, High-Level Trigger, and Control System. Technical report, CERN/LHCC 2003-062, 2004.

[68] NA57 Collaboration. Experiment NA57 at the CERN SPS. *Journal of Physics G*, 25(2):473–480, February 1999.

[69] R. Bramm, H. Helstrup, J. Lien, V. Lindenstruth, C. Loizides, D. Rohrich, B. Skaali, T. Steinbeck, R. Stock, K. Ullaland, A. Vestbø, and A. Wiebalck. High-Level trigger system for the LHC ALICE experiment. *Nuclear Instruments and Methods in Physics Research Section A*, 502:441–442, 2003.

[70] STAR Collaboration. STAR detector overview. *Nuclear Instruments and Methods in Physics Research Section A*, 499:624–632, March 2003.

[71] F. S. Bieser, H. J. Crawford, J. Engelage, G. Eppley, L. C. Greiner, E. G. Judd, S. R. Klein, F. Meissner, R. Minor, Z. Milosevich, G. Mutchler, J. M. Nelson, J. Schambach, A. S. VanderMolen, H. Ward, and P. Yepes. The STAR trigger. *Nuclear Instruments and Methods in Physics Research Section A*, 499:766–777, March 2003.

[72] C. Adler, J. Berger, M. Demello, T. Dietel, D. Flierl, J. Landgraf, J. S. Lange, M. J. LeVine, Jr. A. Ljubicic, J. Nelson, D. Roehrich, R. Stock, C. Struck, and P. Yepes. The STAR Level-3 trigger system. *Nuclear Instruments and Methods in Physics Research Section A*, 499:778–791, March 2003.

[73] R. Bramm, H. Helstrup, J. Lien, V. Lindenstruth, C. Loizides, D. Rohrich, B. Skaali, T. Steinbeck, R. Stock, K. Ullaland, A. Vestbø, and A. Wiebalck. Online pattern recognition for the ALICE High Level Trigger. *Nuclear Instruments and Methods in Physics Research Section A*, 502:443–445, 2003.

[74] Anders Strand Vestbø. *Pattern Recognition and Data Compression for the ALICE High Level Trigger*. PhD thesis, Institutt for fysikk og teknologi, Universitetet i Bergen, May 2004.

[75] J.T. Baines, J. Carter, P.A.M. Eerola, F. Gianotti, T. Hansl, R. Hawkings, D.B. Hubbard, O. Palamara, S. Petrera, S.Yu. Sivoklokov, M. Smizanska, and A. Watson. ATRIG 1.00: ATLAS Trigger Simulation User Guide Revision 0.00. ATLAS Note ATL-SOFT-94-017, CERN, 1994.

[76] Reiner Hauser. The ATLAS Level 2 Reference Software. ATLAS Note ATL-DAQ-2000-019, CERN, 2000.

[77] John Baines. cTrig - C environment for Trigger Studies. http://hepunx.rl.ac.uk/atlasuk/simulation/level2/doc/ctrig/.

[78] M. Elsing et al. Analysis and conceptual design of the HLT Selection Software. ATLAS Note ATL-DAQ-2002-013, CERN, 2002.

[79] ATHENA - Developer Guide. Atlas Offline Computing web site. http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/General/index.html. document in preparation.

[80] G. Comune, A. Corso-Radu, M. Elsing, M. Grothe, T. Schoerner-Sadenius, D. Wicke, S. George, A. Lowe, T. Shears, J. T. Baines, and S. Gonzalez. The algorithm steering and trigger decision mechanism of the ATLAS High Level Trigger. ATLAS Note ATL-DAQ-2003-031, CERN, 2003.

[81] S. Gonzalez, W. Wiedenmann, and A. Radu. Use of Gaudi in the LVL2 Trigger: The steering controller. ATLAS Note ATL-DAQ-2002-012, CERN, 2002.

[82] S. Armstrong (editor), J. T. Baines, C. P. Bee, M. Biglietti, A. Bogaerts, V. Boisvert, M. Bosman, S. Brandt, B. Caron, P. Casado, G. Cataldi, D. Cavalli, M. Cervetto, G. Comune, A. Corso-Radu, A. Di Mattia, M. D. Gomez, A. dos Anjos, J. Drohan, N. Ellis, M. Elsing, B. Epp, F. Etienne, S. Falciano, A. Farilla, S. George, V. Ghete,

S. Gonzalez, M. Grothe, A. Kaczmarska, K. Karr, A. Khomich, N. Konstantinidis, W. Krasny, W. Li, A. Lowe, L. Luminari, C. Meessen, A. G. Mello, G. Merino, P. Morettini, E. Moyse, A. Nairz, A. Negri, N. Nikitin, A. Nisati, C. Padilla, F. Parodi, V. Perez-Reale, J. L. Pinfold, P. Pinto, G. Polesello, Z. Qian, S. Resconi, S. Rosati, D. A. Scannicchio, C. Schiavi, T. Schoerner-Sadenius, E. Segura, T. Shears, S. Sivoklokov, M. Smizanska, R. Soluk, C. Stanescu, S. Tapprogge, F. Touchard, V. Vercesi, A. Watson, T. Wengler, P. Werner, S. Wheeler, F. J. Wickens, W. Wiedenmann, M. Wielers, and H. Zobernig. Algorithms for the ATLAS High Level Trigger. ATLAS Note ATLAS-DAQ-2003-002, CERN, 2003.

[83] J. T. Baines and W. Li. A Data Manager for the ATLAS High Level Trigger. ATLAS Note ATL-COM-DAQ-2003-021, CERN, 2003.

[84] A.G. Melloy, S. Armstrong, and S. Brandt. An implementation of Region-of-Interest selection for ATLAS High Level Trigger and offline software environments. ATLAS Note ATLAS-SOFT-2003-005, CERN, 2003.

[85] Xilinx. *Virtex-4 User Guide.* http://www.xilinx.com, 2006.

[86] IEEE Comp. Soc. *IEEE Standard Description Language Based on the Verilog^{TM}Hardware Description Language.* IEEE Std 1364-2001, 2001.

[87] IEEE Comp. Soc. *IEEE Standard VHDL Language Reference Manual.* IEEE Std 1076-2002, 2002.

[88] Katherine Compton and Scott Hauck. Reconfigurable computig: A survey of systems and software. *ACM Computig Surveys*, 34(2):171–210, June 2002.

[89] Klaus-Henning Noffz. *Ein FPGA-Prozessor als 2nd-Level-Trigger für ATLAS.* PhD thesis, Universität Mannheim, 1996.

[90] Jozsef Ludvig. *Enable++: Ein universeller FPGA-Triggerprozessor für das ATLAS-Experiment.* PhD thesis, Universität Mannheim, 1998.

[91] Holger Singpiel. *Der ATLAS LVL2-Trigger mit FPGA-Prozessoren.* PhD thesis, Ruprecht-Karls-Universität Heidelberg, November 2000.

[92] Andreas Kugel. MPRACE, preliminary documentation. http://akugel.home.cern.ch/akugel/mpRace/.

[93] PLX Technology. *PCI 9656 Data Book. PLX Technology.* http://www.plxtech.com.

[94] Xilinx. *Virtex™-II Platform FPGAs: Complete Data Sheet.* http://www.xilinx.com, 2003.

[95] Oliver Brosch. *A Kaon Trigger for FOPI.* PhD thesis, Ruprecht-Karls-Universität Heidelberg, May 2004.

[96] Matthias Müller. *Evaluation of an FPGA and PCI Bus based Readout Buffer for the ATLAS Experiment.* PhD thesis, Universität Mannheim, 2004.

[97] Stefan Hezel. *FPGA-basiertes Template-Matching mit Distanztransformierten Bildern.* PhD thesis, Universität Mannheim, February 2004.

[98] Gerhard Lienhart. *Beschleunigung Hydrodynamischer Astrophysikalischer Simulationen mit FPGA-Basierten Rekonfigurierbaren Koprozessoren.* PhD thesis, Ruprecht-Karls-Universität Heidelberg, July 2004.

[99] Christian Hinkelbein. *Control Software for Reconfigurable Coprocessors.* PhD thesis, Universität Mannheim, October 2005.

[100] A. Baratella, P. Morettini, M. Dameri, and F. Parodi. PixTrig: a Level 2 track finding algorithm based on pixel detector. ATLAS Note ATL-DAQ-2000-025, CERN, 2000.

[101] R. Dankers and J. Baines. A data preparation algorithm for the Precision Tracker LVL2 FEX. ATLAS Note ATL-DAQ-99-001, CERN, 1999.

[102] G. Lienhart, A. Kugel, and R. Männer. Using floating point arithmetic on FPGAs for accelerating scientific N-Body simulations. In Jeffrey Arnold and Kenneth L. Pocek, editors, *FCCM02, Symposium on Field-Programmable Custom Computing Machines*, pages 182–191, Napa Valley, California, USA, April 2002. IEEE Computer Society Press.

[103] N. Konstantinidis, M. Sutton, J. Baines, D. Emeliyanov, F. Parodi, C. Schiavi, and H. Drevermann. Fast tracking for the ATLAS LVL2 trigger. ATLAS Note DAQ-CONF-2005-001, CERN, 2005.

[104] Dmitry Emeliyanov. A Kalman filter for track fitting in trigidscan. ATLAS Note ATL-COM-DAQ-2004-012, CERN, 2004.

[105] P. Billoir and S. Qian. Further test for the simultaneous pattern recognition and track fitting by the Kalman filtering method. *Nuclear Instruments and Methods in Physics Research Section A*, 294:219–228, 1990.

[106] P. Billoir and S. Qian. Further test for the simultaneous pattern recognition and track fitting by the Kalman filtering method. *Nuclear Instruments and Methods in Physics Research Section A*, 295:492–500, 1990.

[107] D.G. Cassel and H. Kowalski. Pattern recognition in layered track chambers using a tree algorithm. *Nuclear Instruments and Methods in Physics Research Section A*, 185:235, 1981.

[108] Weidong Li. Track finding in SCT using a tree algorithm. http://www.hep.ph.rhbnc.ac.uk/∼li/ctrig/treeAlgorithm.htm.

[109] Sergey Sivoklokov. High-p_T Level 2 Trigger algorithm for the TRT detector in ATRIG. ATLAS Note ATL-DAQ-2000-043, CERN, May 2000.

[110] C. Hinkelbein, A. Kugel, R. Männer, M. Müller, M. Sessler, H. Simmler, H. Singpiel, J. Baines, R. Bock, and M. Smizanska. Pattern recognition in the TRT for the ATLAS B-Physics trigger. ATLAS Note ATL-DAQ-99-012, CERN, 1999.

[111] Matthias Sessler. *Algorithms on CPUs and FPGAs for the ATLAS LVL2 Trigger*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, February 2000.

[112] U. Egede, T. Akesson, D. Froidevaux, and I. Gavrilenko. Fake tracks in the ATLAS straw detector. ATLAS Internal Note INDET-NO-083, CERN, 1994.

[113] Igor Gavrilenko. Pattern recognition in TRD/Tracker (TRD/T). ATLAS Internal Note INDET-NO-016, CERN, 1992.

[114] Dmitry Emeliyanov. The Probabilistic Data Association filter for the fast tracking in ATLAS TRT. ATLAS Note ATL-COM-DAQ-2005-022, CERN, 2005.

[115] A. DellAcqua (reviewer), S. George (reviewer), W. Wiedenmann (reviewer), J. Baines (PESA ID Algorithms coordinator), and M. Elsing (ID Software coordinator). LVL2 Inner Detector Reviewers Comments. https://uimon.cern.ch/twiki/bin/view/Atlas/TrigInDetRevComment.

[116] Dmitry Emeliyanov. The timing measurements of the Level-2 Trigger data preparation and track reconstruction algorithms. In preparation.

[117] J. Baines, B. Epp, V. Ghete, A. Nairz, S. Sivoklokov, S. George, G. Hollyman, and D. Hutchcroft. Effects of inner detector misalignment and inefficiency on the ATLAS B-physics trigger. ATLAS Note ATL-DAQ-2001-006, CERN, 2001.

[118] Thijs Cornelissen. CTBTracking: track reconstruction for the testbeam and cosmics. ATLAS Note COM-INDET-2006-003, CERN, March 2006.

[119] The PESA Steering Working Group. Steering, configuration and implementation of HLT trigger menus. analysis, conceptual design, requirements. In preparation.

[120] Cvetan Cheshkov. Fast Hough Transform tracking for the ALICE TPC. In *Workshop on Tracking In high Multiplicity Environments*, October 2005.

ACKNOWLEDGMENTS

I would like to thank all the people who have supported me during the work on this thesis.

I thank Prof. Dr. Reinhard Männer for the possibility to work in his research group, great support, advices and very nice working atmosphere. I thank all people from FPGA group (current and former members): Maoyuan Yu, Guillermo Marcus, Stefan Hezel, Oliver Brosch, Holger Singpiel, Harald Simmler. Special thanks to Andreas Kugel, Matthias Müller, and Erich Krause for the very detailed answers for my, some times naive, questions. I would like to express personal gratitude to Christian Hinkelbein for long and very useful discussions about the track recognition and the TRT LUT-Hough algorithm. It is a real pleasure to work in one group with such people.

I am very grateful to Oleg Krivonos and Gerhard Lienhart for the careful reading of this thesis and very useful comments.

Thanks must be given to colleagues from ICM: Dzmitry Stsepankou and Dzmitry Maximov for the interesting discussions and ideas from other then High Energy physic image processing applications.

I would like to acknowledge the support and help of the ATLAS HLT and PESA groups, especially Sergey Sivoklokov, John Baines, Dimitry Emeliyanov, Nikos Konstantinidis, and Steve Armstrong.