System-on-an-FPGA Design for Real-time Particle Track Recognition and Reconstruction in Physics Experiments

Ming Liu[†‡], Wolfgang Kuehn[‡], Zhonghai Lu[†], Axel Jantsch[†]

† Dept. of Electronic, Computer and Software Systems, Royal Institute of Technology, Sweden

‡ II. Physics Institute, Justus-Liebig-University Giessen, Germany

{mingliu, zhonghai, axel}@kth.se, wolfgang.kuehn@exp2.physik.uni-giessen.de

Abstract

In particle physics experiments, the momenta of charged particles are studied by observing their deflection in a magnetic field. Dedicated detectors measure the particle tracks and complex algorithms are required for track recognition and reconstruction. This CPU-intensive task is usually implemented as off-line software running on PC clusters. In this paper, we present a system-on-chip design for the track recognition and reconstruction based on modern FPGA technologies. The basic principle of the algorithm is ported from software into the FPGA fabric. The fundamental architecture of the tracking processor is described in detail. Working as processing engines in compute nodes, the tracking processor contributes to recognize potential track candidates in real-time and promotes the selection efficiency of the data acquisition and trigger system. Our design study shows that the tracking module can be integrated in a single Xilinx Virtex-4 FX60 FPGA. The processing capability of the design is about 16.7K sub-events per second per module with our experimental setup, which achieves 20 times speedup compared to the software implementation.

1 Introduction

Modern nuclear and particle physics experiments, for example HADES [1] and PANDA [2] at GSI, BESIII [3] at IHEP, LHC [4] at CERN, are expected to run at a very high reaction rate (e.g. PANDA, 10-20 MHz) and able to deliver a data rate of up to hundred GBytes/s (PANDA, up to 200 GBytes/s). Among the huge amounts of data, in many cases only a rare proportion is of interest due to its particular physics contents and should be selected for in-depth physics analysis. Besides, these amounts of data cannot be entirely stored because of the storage limitation. Therefore it is essential to realize an efficient on-line data acquisition and trigger system which processes the sub-events coming from detectors and reduces the data rate by several orders of magnitude via rejecting background data. In the contemporary facilities, the trigger system is normally divided into multiple levels, and some feature extraction algorithms were implemented as sophisticated criteria in latter levels, specifically Cherenkov ring pattern recognition, Time-Of-Flight (TOF) analysis and Shower pattern recognition [5] [6] [7]. Only the sub-events which possess expected patterns and could be successfully correlated with each other receive a positive decision and are forwarded to the event-builder and further the mass storage for later off-line analysis. Others will be discarded in the real-time processing.

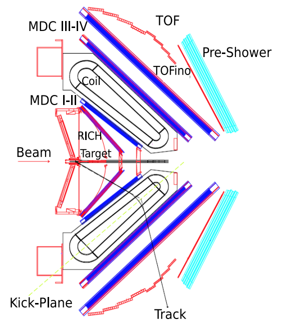
Particle track reconstruction in detectors is a process of computing and identifying the flying tracks of charged particles traversing a magnetic field. It is an important task with which the particle's momentum can be studied [8] [9] [10]. This work is most CPU-intensive and traditionally the calculation was implemented as the off-line software executed on PC clusters after experiments. However, due to the increasing data rate in modern experiments, it becomes highly necessary to port fully or partially of the algorithm into the in-field data acquisition and trigger hardware for selective event filtering. Together with other feature extraction methods, the track recognition and reconstruction algorithm works to select interesting physics events and thereby significantly reduce the amount of raw data written to the mass storage.

As the great development of FPGA technologies and the HW/SW co-design methodology, FPGA based designs are well suited for high performance and reconfigurable applications especially the field of scientific computing. Here we base our design on the Xilinx platform FPGA, which has the capability to build embedded systems on-chip, as well provides large freedom to design application-specific processors with the programmable logic resource. In section 2 we will first explain the basic physics principle of the track recognition and reconstruction in an example detector, the HADES experiments at GSI Darmstadt. Some previous work will also be addressed. In section 3, the hierarchical architecture of the computation platform will be demonstrated. In section 4 we focus on the tracking application and describe the Tracking Processing Unit (TPU) design in detail. In section 5 the experimental results are presented and in section 6 we conclude the paper and propose our future work.

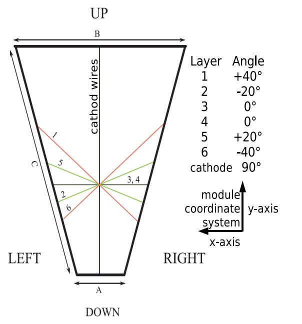
2 Application Description

2.1 Principle of Track Recognition and Reconstruction

The HADES tracking system, as an instance of modern experimental facilities, consists of four Mini Drift Chamber (MDC) detectors which have six identical trapezoidal sectors. Two MDC layers are located before and two behind the toroidal magnetic field which is produced by 6 superconducting coils, as shown in figure 1(a). A total number of 24000 sense wires are arranged in six orientations and 24 layers: +40°, -20°, 0°, 0°, +20°, -40°, as shown in figure 1(b). Sense wires are elementary units used to reconstruct particle tracks by generating pulse signals when charged particles traverse the detector close to them. The important term *drift cell* is defined as the long thin strip area centered around a sense wire and delimited by the field wires and the cathode wires [11] [12]. When a particle passes through this area and causes the sense wire to produce a pulse signal, we call that this drift cell or the sense wire is “fired”.



(a) Lateral cut-away view of the HADES detector system



(b) One sector of the MDC with six orientation wires

Figure 1. MDCs in HADES detector system

Particle tracks will be bent in the magnetic field between inner (I - II) and outer (III - IV) MDCs. The magnet is constructed in such a way that the magnetic field practically does not penetrate into MDC modules. Thus segments of tracks before and behind the coil could be approximately described by straight lines. They can be reconstructed separately with the inner or outer MDC information. The basic principle is quite similar and thus in this paper we focus only on the inner part.

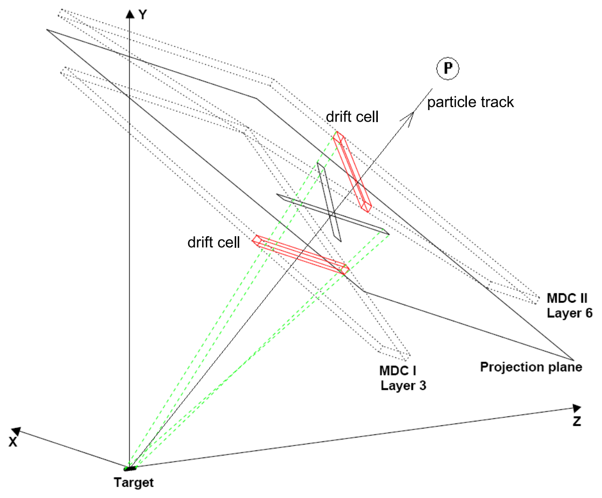
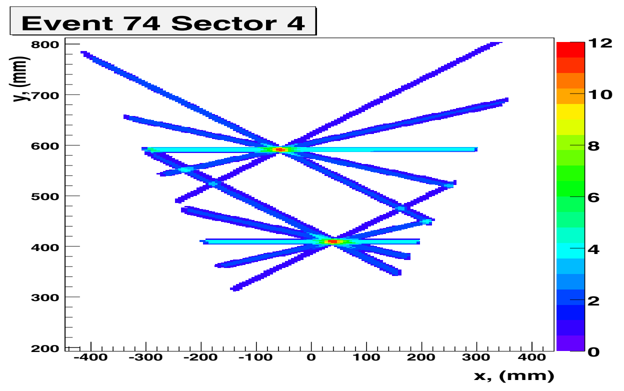


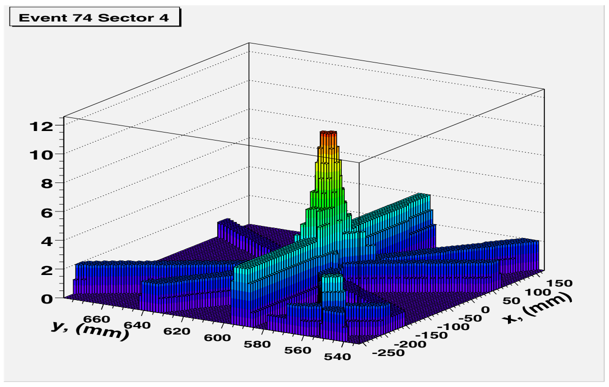
Figure 2. Track recognition and reconstruction in inner MDCs

Figure 2 shows a coordinate system in which a charged particle passes through two inner MDCs. When the beam hits the target, charged particles are emitted from the target position and go forward through drift cells in different layers in a straight path. Therefore pulse signals are produced on the fired sense wires with high probability (>95%). Visualizing every fired drift cell as a line, apparently the particle passed through the MDC at the point where all the wires from different layers cross. In our practical computation, the track candidate search is performed for two chambers per sector simultaneously. As shown in figure 2, the sensitive volume of each drift cell is projected from the boundary of the target onto a plane located between the chambers. When a particle goes through all the 12 layers (6 orientations and layers per MDC module) from the target, projections of the fired drift cells provide overlapped regions on the projection plane. To search for such regions the projection plane is treated as a two dimensional histogram with the projection area as bins (grids). For each fired drift cell, its projection bins are all increased by one. By finding the locally maximum bins whose values are also above a given threshold, track candidates can be recognized and the tracks are reconstructed as straight lines from the point-like target to those bins.

Obtained from the off-line analysis, figure 3(a) is a 2-dimensional plot for one sector and with 2 passed particles. The scale on the right side shows the correspondence between the bin values and the colors in the plot. Figure 3(b) shows the 3-dimensional display of figure 3(a) for a single track. The peak in the center is recognized as the candidate bin which was most likely caused by a particle passing through this point.



(a) Projection plane with two passed tracks



(b) 3D display of the accumulated bins for a single track

Figure 3. Particle tracks in the projection plane of one sector

2.2 Previous Work

The tracking computation has the features of high computing power requirement and complex algorithm. Many contemporary experiments use the software-on-PC-cluster off-line solution to search for particle tracks. For example the existing HADES on-line trigger processing does not exploit the information from the tracking sub-system due to the large amount of raw data (a few GBytes/s). This trigger system was developed more than 10 years ago based on the Xilinx XC4000 series FPGAs, and at that time the tracking algorithm was hardly implemented in hardware [6]. The lack of the on-line tracking processing led to a significant loss in event selectivity and thus the corresponding part is expected to be added in the data acquisition and trigger system.

There does also exist some hardware implementation, for instance the ATLAS level 2 trigger [7]. However their solution appears as PCI cards in commodity PCs, which only releases the simple but computing-intensive steps to FPGAs while remains others on CPUs. Hence it is in fact hardware/software co-processing and the work relies much on the PC. In addition the limited bandwidth between CPUs and FPGAs via the PCI bus becomes the bottleneck and has to be considered when partitioning the algorithm steps to the CPU and the FPGA.

3 Computation Platform

3.1 Computation Network Architecture

To manage the data rate of up to hundred GBytes/s, all feature extraction algorithms will be partitioned and run in parallel in a network architecture. Shown in figure 4, Compute Nodes (CN) provide both internal and external channels. Via the external interconnections including optical links and Gigabit Ethernet, data streams are received from detectors and the selected results after processing are forwarded to the PC farm for storage or high level analysis. The internal high-speed connections are employed to partition the algorithms and to correlate results among them. We utilize the ATCA full-mesh backplane [13] to provide flexible communications among all compute nodes.

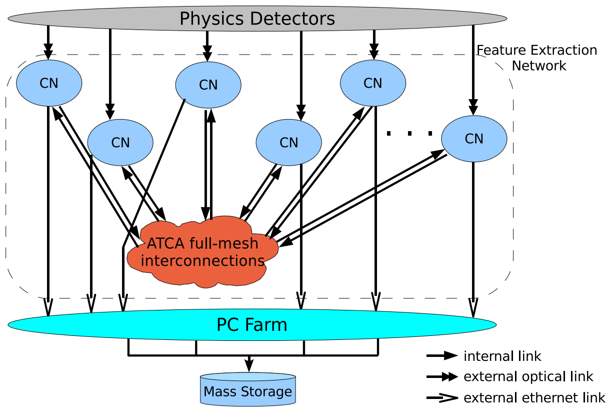


Figure 4. Computation network for on-line feature extraction

3.2 Compute Node

Figure 5 shows the schematic of a CN board for prototype design. On each board there are five Xilinx Virtex-4 FX60 FPGAs, four of which (No. 1 to 4) work as algorithm processors and the fifth (No. 0) as a switch interfacing to other CNs via the full-mesh backplane. Each processor FPGA has multiple external links and all five ones are equipped with local DDR2 memories for data buffering and large look-up table purposes. Point-to-point on-board interconnections make it convenient to partition algorithm implementations and realize parallel processing in different FPGA nodes.

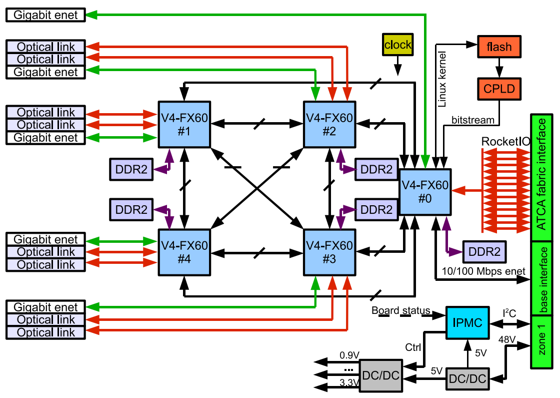


Figure 5. Compute node design

3.3 System on the FPGA Node

The Xilinx platform FPGA possesses many kinds of hardcore components, such as the PowerPC 405 embedded CPU, RocketIO multi-gigabit transceivers, and Gigabit Ethernet MAC [14]. Accompanied with softcore libraries, a bus-based system can be built and integrated in a single FPGA (see figure 6). Feature extraction processors, in our case the Tracking Processing Unit (TPU), are connected to the system bus. They utilize the hardware parallel and pipelined architecture to speedup the data processing over software. Running on the embedded PowerPC, the open-source Linux operating system takes charge of network protocol processing, as well as interaction tasks with the operators including reconfiguring the system, adjusting experimental parameters, or displaying results, etc.. Hence the CPU/FPGA solution can be implemented on a single chip, with which the density of computing power as well as the flexibility of reconfiguring components and interconnections are increased significantly.

Compared to the previous implementation, our design is able to process detector data without dependence on the CPU. The pure hardware processing eliminates the bandwidth limitation between the CPU and the algorithm processor, and makes it possible to integrate multiple cores in one bus system for parallel processing.

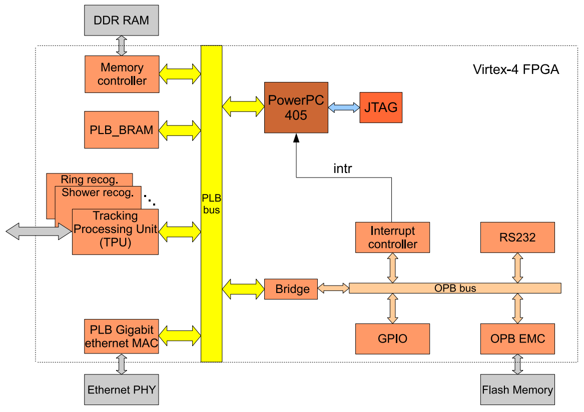


Figure 6. FPGA node with TPU connected to the PLB bus

4 TPU Design in FPGA

As shown in figure 7, the TPU design is decomposed into four modules, specifically two Look-Up Tables (LUT), the accumulate unit and the peak finder. For each sense wire, the address LUT stores its address information and the projection LUT shows which bins on the projection plot will be touched by the wire's projection. The address LUT is arranged in the Block RAM (BRAM) inside the FPGA to save access time and decrease bus utilization. The projection LUT may be large therefore it is implemented in the external DDR memory. With the guidance of the address information, the accumulate unit accumulates the touched times by fired wires for all the bins. Finally the peak finder calculates and figures out the exact points where particles probably passed through.

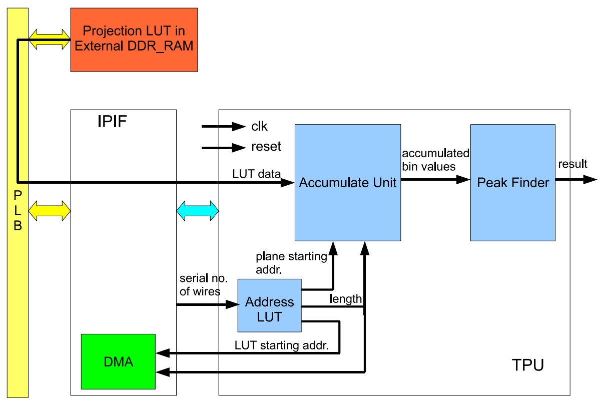


Figure 7. Block diagram of the TPU structure

4.1 Projection LUT and Address LUT

The inputs of the TPU are serial numbers of fired sense wires from the MDC circuits. To decide which bins on the projection plane might be touched by the shadow of fired wires, real-time calculation should be avoided due to the geometrical complexity. Instead, an off-line built projection LUT is used to store this information with the serial numbers of wires as entries. To save storage space, only the bins in the projection plot from the *plane starting address* and within the *length* should be considered when building the LUT mapping. As an instance of projecting a +20° drift cell and shown in figure 8, only the bottom bins are considered as the projection information since the bins on the top are obviously not touched by the shadow. Hence from the serial number of a sense wire, the *address LUT* derives the *plane starting address* and the *length*, as well as the *LUT starting address* which indicates the starting location in the physical memory and initializes DMA transfers.

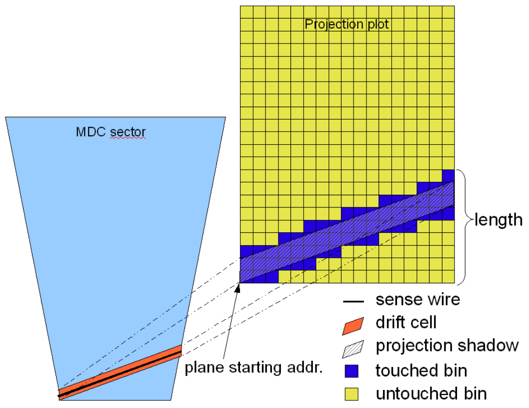


Figure 8. Projection and touched bins of a fired drift cell on the projection plane

In practice, we use 128 x 256 bins as the resolution of the projection plot. Compared to the method which concerns all the bins in the projection plot for the wires, our mechanism not only shrinks the *projection LUT* size to even less than one fifth (1.5 MB vs. 8.6 MB), but also significantly saves clock cycles when feeding data from the DDR LUT to the processing unit and avoids meaningless computation.

4.2 Accumulate Unit

The accumulate unit works to accumulate the touched times of bins, when the LUT data for different orientation wires are supplied by DMA transfers.

This module is basically structured by a dual-port BRAM block, some adders and registers (shown in figure 9). The BRAM block is 3-dimensional and has 4 bits per bin which allow to represent the maximum 12 layer wires of two MDCs. Initially all the bins are reset to zero. As the projection LUT provides data which show if a specific bin will be touched (a bit of '1') by a fired wire or not (a bit of '0'), the values of bins are correspondingly increased by one or not. For each wire, the accumulation only happens on those bins from the *plane starting address* and within the *length*, which are both from the address LUT outputs. The computation process is pipelined in two stages. One is to write the accumulated results into the BRAM block. The other is to read out the values of the next address and get them ready for the next cycle accumulation. After the processing of all fired wires within a time slot, the entire BRAM block will be exported to the next level peak finder, searching for the peak bins or track candidates. Then the BRAM block will be reset again and get ready for the next round computation.

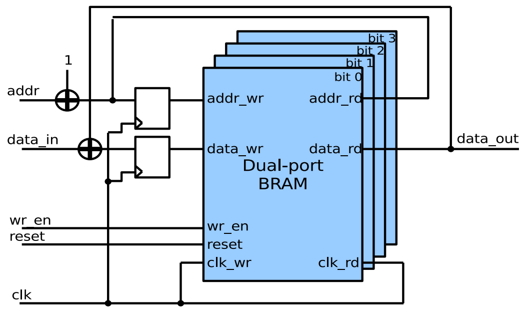


Figure 9. Pipelined structure of the accumulate unit

4.3 Peak Finder

The peak finder is the most computing-intensive part in the entire TPU design. It indicates not only which bins have values no less than the threshold, but also the exact peak bins in their neighbourhoods where tracks probably passed. Figure 10 shows the motivation to build such a module. There are seven bins meeting the threshold requirement assuming the threshold is 10. However in fact they belong to a single track. So more delicate computation should be done to find out the peak bin in this area, which is most probably the point a particle passed through in the projection plane. In case of multiple peaks with the same value and neighboured with each other, we can export any one of them or directly all according to the system requirement.

To indicate the peak in an area, each bin is arranged to be compared with all its eight neighbours, except those in the boundaries which have fewer neighbours. If none of the neighbours is greater than that bin, it is determined

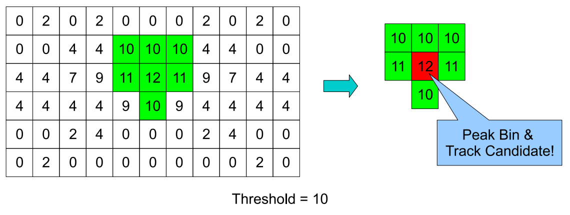


Figure 10. Selection of the peak bin in the neighbourhood

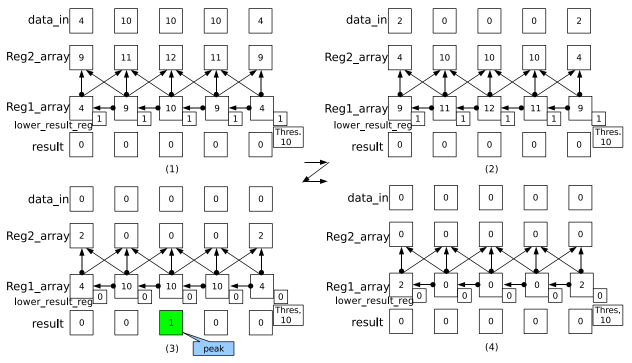


Figure 11. Pipelined peak finding process

as the peak. Otherwise the greater neighbours overcome it and keep on searching. This calculation was implemented as pipelined comparisons in our design. Shown in figure 11, the input data go downwards cycle by cycle from "data_in", passing two levels of registers "Reg1_array" and "Reg2_array" until the results come out. Using the same data from figure 10, the pipelined peak finding process is demonstrated by four sub-figures for four clock cycles. Each square in "Reg1_array" which represents a single bin with its value inside, is being compared with four of its neighbours: the upper-left, the upper, the upper-right, and the left, with an arrow indicating a comparison. It is also being compared with its right neighbour as the comparison is initiated by its right neighbour bin. Moreover the bins in "Reg2_array" are being compared with their respective lower-left, lower and lower-right neighbours by using the same comparators. Hence in summary, each bin is compared with its three lower neighbours when it stays in "Reg2_array", and with its five upper and horizontal neighbours after it steps to "Reg1_array" in the next clock cycle. The comparison results with the lower neighbours are stored in the register "lower_result_reg" temporarily. A bit of "1" means none of its lower neighbours is larger than that bin while "0" means it cannot be the peak and will be ignored. In case of two neighboured bins having the same value, the one in the higher address direction (right and upper) wins the comparison and may go on being compared with those at even higher addresses. The bit from "lower_result_reg" will be ANDed with the results from the five upper and horizontal comparisons and the threshold comparison. If the final result is '1', that bin is the peak in its area and is recognized as a track candidate. Figure 12 illustrates the Register Transfer Level (RTL) schematic of the pipelined peak finder.

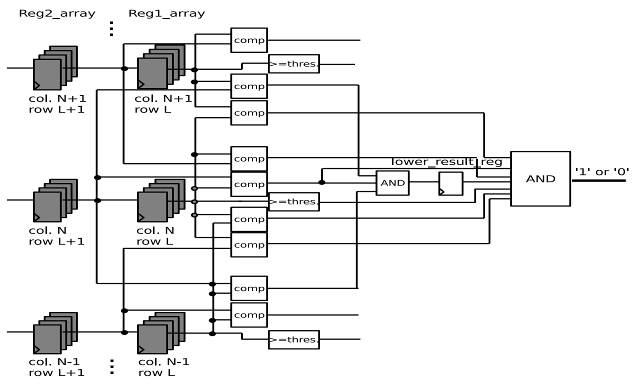


Figure 12. The RTL structure of the peak finder

5 Experimental Results

5.1 Implementation Results

The TPU design was described in VHDL and synthesized using Xilinx ISE 8.2. When the projection plane is configured as 128 x 256 bins, the resource consumption statistics are shown in table 1 as well as the utilized percentages of the Xilinx Virtex-4 FX60 FPGA. The TPU is connected to the PLB bus via an IP interface (IPIF) [15]. Hence the resource utilizations of both the compute node platform and the PLB-IPIF are listed in the table as well. From the statistics, we can see that 17.7% Block RAM resource is used by the TPU, in which the dual-port BRAM block in the accumulate unit and the address LUT are two main consumers. In addition 10.2% LUT resource and 3.4% Flip-Flops contribute to mainly construct the comparators and registers in the device. As shown in the last column of the table, the whole system with a TPU connected utilizes 32.8% LUTs, 18% Flip-Flops, 25.4% BRAMs and 6.3% DSP slices of the Virtex-4 FX60. So we conclude that it is feasible to integrate the entire system in a single Virtex-4 FX60 to perform the on-line inner track recognition and reconstruction.

The synthesis timing summary shows that the TPU design could run at above 125 MHz without any optimization

Resources	TPU	compute node platform	PLB-IPIF	system with TPU (sum)
4-input LUTs	5175 out of 50560 (10.2%)	8531 out of 50560 (16.9%)	2900 out of 50560 (5.7%)	16606 out of 50560 (32.8%)
Slice Flip-Flops	1715 out of 50560 (3.4%)	5724 out of 50560 (11.3%)	1640 out of 50560 (3.2%)	9079 out of 50560 (18.0%)
Block RAMs	41 out of 232 (17.7%)	18 out of 232 (7.8%)	0	59 out of 232 (25.4%)
DSP Slices	0	8 out of 128 (6.3%)	0	8 out of 128 (6.3%)

Table 1. Resource consumption

effort. To match the speed of the PLB bus, we fix the clock frequency of the TPU as 100 MHz.

5.2 Performance Measurements

The processing capability of the TPU depends heavily on the counts and the position of fired wires within a time slot. Since the projection LUT mappings for the wires in different positions have various sizes, transferring data to the TPU takes different clock cycles. In our cycle-accurate simulation, we assumed 30 fired sense wires (around 3 particle tracks) in each sub-event from MDCs and the projection LUT size of 5.7 Kbits per wire on average (1.5 MB for 2110 wires). Both the PLB bus and the TPU run at 100 MHz. With this experimental setup, it takes less than 6000 cycles, which is equivalent to 60 μs, to process one MDC sub-event. This implies a processing capability of 16.7K sub-events per second. In contrast the software in C program which ran on an Intel Xeon 2.4 GHz and 1 GB DDR2 memory server with the Gentoo Linux operating system, achieves only 0.82K sub-events per second in the same experimental condition. Hence the speedup of 20 for the hardware processing is foreseen and even more might be achieved after the design optimization.

6 Conclusion and Future Work

We have presented the work of porting the track recognition and reconstruction algorithm into the FPGA fabric and constructing the embedded system on the FPGA. The tracking processing unit appears as a PLB device in the bus system and performs the pattern recognition computation. The design was described in VHDL with the partition of four modules: two LUTs, the accumulate unit and the peak finder. The implementation results justify the feasibility of integrating the system in a single Xilinx Virtex-4 FX60 FPGA.

The future work includes to partition and parallelize the algorithm and to distribute in multiple nodes of the computation network. Design optimizations such as improving the clock frequency or increasing the bus data width are also expected for gaining higher performance.

Acknowledgement

This work was supported in part by BMBF under contract Nos. 06GI179 and 06GI180. In addition the authors would like to thank Dr. Vladimir Pechenov for his work and knowledge support on the physics background.

References

[1] High Acceptance Di-Electron Spectrometer (HADES) @ GSI, Darmstadt, Germany, www-hades.gsi.de.

[2] antiProton ANnihilations at DArmstadt (PANDA) @ GSI, Darmstadt, Germany, www.gsi.de/panda.

[3] BEijing Spectrometer (BES) @ Institute of High Energy Physics, Beijing, China, http://bes.ihep.ac.cn/bes3/index.html.

[4] The Large Hadron Collider (LHC) @ CERN, the European Organization for Nuclear Research, http://lhc.web.cern.ch/lhc/.

[5] I. Froehlich, A. Gabriel, et al., Pattern recognition in the HADES spectrometer: an application of FPGA technology in nuclear and particle physics, *In Proc. of the 2002 IEEE International Conference on Field-Programmable Technology*, pages 443-444, Dec. 2004.

[6] Michael Traxler, Real-Time Dilepton Selection for the HADES Spectrometer, November 2001, Ph.D thesis, II. Physics Institute of Justus-Liebig-University Giessen.

[7] C. Hinkelbein, A. Kugel, R. Manner, M. Muller, M. Sessler, H. Simmler and H. Singpiel, Pattern Recognition Algorithms on FPGAs and CPUs for the ATLAS LVL2 Trigger, *IEEE Transactions on Nuclear Science*, Volume 48, Issue 3, Part 1, pp. 296-301, Jun. 2001.

[8] K. Rinnert, Track Reconstruction at the LHCb Experiment, *In Proc. of the 10th ICATPP Conference on Astroparticle, Particle, Space Physics, Detectors and Medical Physics Applications*, Oct. 2007.

[9] M. T. Schiller, Standalone Track Reconstruction for the Outer Tracker of the LHCb Experiment Using a Cellular Automation, Jul. 2007, Diploma thesis, Physics Institute of University of Heidelberg.

[10] S. J. Bailey, G. W. Brandenburg, N. Felt, T. Fries, S. Harder, M. Morii, J. N. Oliver, N. B. Sinev and E. Won, Rapid 3-D Track Reconstruction with the BABAR Trigger Upgrade, *IEEE Transactions on Nuclear Science*, Volume 51, Issue 5, Part 1, pp. 2352-2355, Oct. 2004.

[11] Daniel Kirschner, Level 3 Trigger Algorithm and Hardware Platform for the HADES Experiment, Oct. 2007, Ph.D thesis, II. Physics Institute of Justus-Liebig-University Giessen.

[12] G. N. Agakishiev and V. N. Pechenov, Dubna Tracks Reconstruction User Manual, Dec. 2001, HADES internal manual.

[13] PCI Industrial Computers Manufactures Group (PICMG), PICMG 3.0 Advanced Telecommunications Computing Architecture (ATCA) specification, Dec. 2002.

[14] Xilinx, Inc., “Virtex-4 User Guide”, UG070(v2.2) April 10, 2007.

[15] Xilinx, Inc., Xilinx LogiCORE PLB IPIF (v2.01a), Aug. 2004.