High-Level Data Flow Description of FPGA Firmware Components for Online Data Preprocessing*

H. Engel[1], *F. Grüll*[1], *and U. Kebschull*[1]
[1]IRI, Institut für Informatik, Johann Wolfgang Goethe-Universität Frankfurt, Frankfurt, Germany

FPGA firmware for detector read-out is commonly described with VHDL or Verilog. Data processing on the algorithmic level is a complex task in these languages and creates code that is hard to maintain. There are high level description frameworks available that simplify the implementation of processing algorithms. A sample implementation of an existing algorithm and the comparison with its VHDL equivalent show promising results for future online preprocessing systems.

Field Programmable Gate Arrays (FPGAs) are widely used in high energy physics detector read-out chains due to their flexibility. The protocols and interfaces are usually implemented with hardware description languages like VHDL or Verilog. With FPGAs getting bigger and faster they become more and more suitable for performing complex data processing tasks. This can reduce the data volume and significantly ease demands on later software based processing steps. The drawback of the commonly used hardware description languages is that they are mostly working on the Register Transfer Level. This is perfect for high performance protocol and low level interface implementations. However, using these languages to implement data processing on an algorithmic level requires experienced developers and usually involves customized IP cores and latency matching of components. This creates a rather complex and static design. There are several high level hardware description frameworks available that provide their own languages to describe data processing steps on an algorithmic or data flow level. Some of them also come with an own framework including building blocks for PCIe or DRAM interfaces. This significantly speeds up the development compared to a description in plain VHDL or Verilog.

The underlying framework of this work is made by Maxeler Technologies. The platform generates a pipelined version of the algorithm after its data flow graph has been described in a Java-like programming language [1]. The compiler manages the scheduling of the design, inserts latencies in the generated pipelines wherever needed to keep the data in sync, and instantiates interfaces to PCIe or DRAM if required. A software environment with a device driver and C API provides easy to use stream interfaces to the hardware. The compiler translates the data flow description into VHDL code which is then run through the vendor tools.

The algorithm described in this way is the FastClusterFinder that was used as a VHDL core in the readout of the ALICE Time Projection Chamber during LHC run period 1 [2]. A simplified overview of the algorithm is shown in Fig. 1. The incoming raw data is decoded into a data stream with time and location information. The center of gravity and the deviation of peaks are calculated in time direction. In a second step neighboring cluster candidates are merged to get the center of gravity and the deviation of the full cluster in pad direction. The last step is a floating point division. The VHDL implementation is a rather complex design due to its data flow control structures and the number of fixed point and floating point arithmetics.

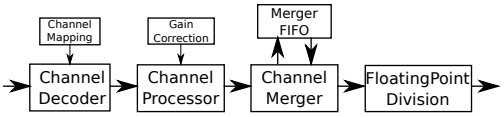

Figure 1: Schematic picture of the ALICE TPC FastClusterFinder algorithm.

A functionally identical version of the FastClusterFinder algorithm has been described with the Maxeler data flow description language. The behavior of this implementation has been verified in simulation using recorded detector data from ALICE TPC. The output of the data flow simulation is directly compared to the output of a Modelsim simulation of the original VHDL code. In comparison to the VHDL implementation the number of lines of code is significantly reduced for the data flow description. Especially the computing intensive parts of the design are very easy to understand and to maintain. The resource usage of the generated design in its current state is slightly different in details but overall in the same order of magnitude as the VHDL implementation.

This implementation shows that there are tools available to describe processing algorithms on an algorithmic or data flow level that are able to generate hardware with comparable resource usage but significantly reduced code volume. This greatly improves maintainability of the code. A next step will be to implement and test the code in actual hardware. Furthermore, the generation of VHDL code out of the data flow description allows the processing elements to be extracted from the vendor framework and integrated as IP core into an own firmware environment.

References

[1] Maxeler Technologies, *Programming MPC Systems*, White Paper, June 2013

[2] T. Alt and V. Lindenstruth, *Status of the HLT-RORC and the Fast Cluster Finder*, GSI Scientific Report 2009

* Work supported by HGS-HIRe, HIC4FAIR

doi:10.15120/GR-2014-1-IT-14