A Novel Simulation and Verification Approach in an ASIC Design Process

D. Husmann, M. Keller, K. Mahboubi, U. Pfeiffer, C. Schumacher
Kirchhoff-Institut für Physik, Universität Heidelberg, Germany

Abstract— **We have built a fast signal-processing and readout ASIC (PPrAsic) for the Pre-Processor system of the ATLAS Level-1 Calorimeter Trigger. Our novel ASIC design environment incorporates algorithm development with digital hardware synthesis and verification. The purely digital ASIC was designed in Verilog HDL (hardware description language) and embedded in a system wide analog and digital simulation of implemented algorithms. We present here our design environment and experience that we gained from the design process.**

Keywords— **ASIC design flow, combined simulations, complex design environment, test-vector generation.**

I. Introduction

The ATLAS experiment at CERN requires a highly selective trigger system with optimal efficiency. The event selection in ATLAS will be achieved by a three level trigger system. The first trigger level (Level-1 Trigger) is a fast pipelined system for the selection of rare physics processes. It is designed to achieve an event rate reduction from 40 MHz LHC bunch-crossing rate down to the Level-1 accept rate of 75 kHz (100 kHz upgrade)[1]. This is achieved by algorithms which work on a three-dimensional energy map generated by the ATLAS calorimetry (Calorimeter Trigger) and a selection based on coincidences in muon trigger chambers (Muon Trigger). The algorithms of the Level-1 Calorimeter Trigger have to identify 'local' and 'global' energy depositions within the calorimetry. The digital data for these algorithms are provided by the Pre-Processor system which is located at the front-end of the Calorimeter Trigger. It receives about 7200 analog input signals (trigger towers) from the electromagnetic and the hadronic calorimeters. The maximum latency for the Level-1 Calorimeter Trigger is 2.0 μs. Of this, only a small fraction (300 ns, or 12 bunch-crossings) is allowed for the Pre-Processor ASIC, the main part of the Pre-Processor System.

The maximum latency and the large number of analog signals put tight constraints on the Pre-Processor system. It has to be built as a hard-wired front-end to perform fast signal processing on all analog input signals in parallel. The system provides the input data for the Level-1 Calorimeter Trigger algorithms and it performs the readout of data on which the Level-1 Trigger has based its decision. In this way it has the flexibility to allow tests on the trigger performance.

An overview of the Pre-Processor ASIC and its tasks is given in Section II. This is followed by requirements and problems of an ASIC design process in Section III. It also illustrates the design environment which was developed to meet this requirements. The modules of these environment, various simulation tools, are described and their integration into a complex design environment is explained. Section IV presents the documentation which attends such a design process. Results and design experiences are mentioned in Section V.

II. The Pre-Processor ASIC (PPrAsic)

The Pre-Processor ASIC performs signal processing on digitized data from four trigger-tower signals. The design is completely described in the Hardware Description Language Verilog. The ASIC will have a power supply of 3.3 V and will be manufactured in a 0.6 μm CMOS process. For internal data storage of raw and processed data the ASIC is equiped with 24 RAM blocks with a total size of 66 kbit with an area of 14.12 mm^2. This and the digital core logic lead to a total chip area of about 60 mm^2. The tasks that the Pre-Processor system has to perform, based on its 7200 analog input signals, can be summarised as follows[2]:

- Preprocessing: The trigger algorithms have to be provided with digital data containing the deposited transverse energy, identified to a unique interaction in time (bunch-crossing identification, BCID). Two algorithms exist to perform the BCID. One is applied to non-saturated signals and uses a finite-impulse-response filter and a peak-finder. The other is applied to saturated signals. The algorithms are fed with synchronized digital data for particles with different time-of-flight from the interaction point to the calorimeter and for different cable lengths from the calorimeter to the trigger cavern.
- Readout of event data: On each Level-1 decision the corresponding raw trigger data together with some samples preceding and following the identified bunch-crossing can be read out. This is needed for diagnostic and monitoring purposes of the performance of the trigger system. Therefore the PPrAsic contains pipeline memories and buffers to store the incoming data. To guarantee high flexibility, the ASIC has many programmable registers which can be accessed through two serial interfaces. Each one of these is responsible for 2 channels. In addition, optional daisy-chaining is also foreseen.

III. The Design Environment

The requirements of the ATLAS trigger system, like optimal efficiency and fast processing, make it necessary to develop application specific algorithms implemented in integrated circuits. Furthermore the system complexity requires compact electronics with a high level of integration.

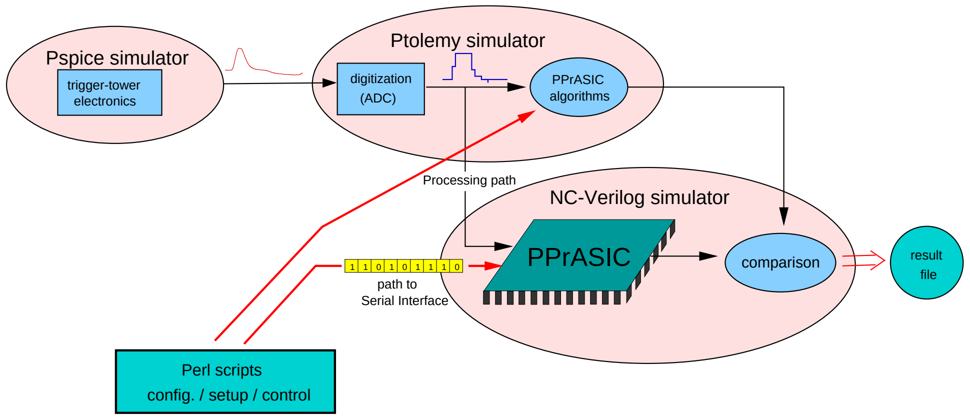


Fig. 1. Design environment used for algorithm development, hardware description and verification of the Pre-Processor ASIC

In order to meet all these constraints a flexible design environment was developed for the Pre-Processor ASIC. The following list shows the requirements and difficulties that appear in an ASIC design process mapped on the development of the PPrAsic:

- Tools: The hardware description language and the tools for synthesis, layout and timing analysis have to be chosen. The tools for simulation and testing have to be chosen and combined in an effective manner to a compact and flexible system. This system should also support the development of algorithms that will be implemented in the ASIC.
- Design cycle: The number of design cycles aimed for has to be determined. In the case of the PPrAsic the high cost of the design and the short time for the design and manufacture forced the decision to attempt to finish the project in just one design cycle. This decision requires a very methodical and careful design process.
- Time to experiment: The tight constraints for a compact system demand a high level of integration on Multi-Chip Modules (MCMs). To start the design and production of these components it is necessary to keep the ASIC development and production short in order not to retard the production of the final system.
- Code coverage: A simulation environment has to be developed that allows a complete test of the hardware description. It should show that the written code meets the specification and system requirements.
- Fault coverage: It has to be assured that the design is error free and that no timing violations have occured. Therefore a test is needed that checks the correct behaviour of the design and the manufactured hardware in all possible configurations.
- Test of the hardware: To save design-time it is necessary to develop a complex simulation environment that can be utilized in all steps of the design process. Of course it should be helpful in developing the hardware description, but besides that it should also be applicable for the later testing of the hardware.
- Documentation: A project that involves several designers requires not only a complete documentation of the final design, it should guarantee a detailed documentation all along the design process.
- Software: Software that guarantees direct access to all the hardware produced is needed. It should also allow system integration of the hardware without losing flexibility.

A. Simulation Tools

Our aim has been to provide a platform for a system-wide simulation of implemented algorithms and their surrounding electronics. With this simulation it is possible to develop the algorithms that will be implemented on the ASIC and to demonstrate their efficiency. It can be used to create vectors for the simulation of the hardware description during the design process. Once the ASIC is back from its foundry, the simulation can create the test vectors and the expected results for real operation.

Figure 1 shows the design environment. It is built up out of three types of simulators: an analog circuit simulator (Pspice[3]) for analog input pulse generation, a heterogeneous simulator for analog and digital algorithm simulation (Ptolemy[4]), and a simulator for the hardware description (NC-Verilog[5]). In the following the tasks of these simulators will be explained in more detail:

1. Pspice simulation: This simulator is used to create realistic analog input signals. Therefore it includes the complete analog *trigger-tower* electronics and models of the cables that connect them with the Pre-Processor system. The simulation allows production of both saturated and non-satureted signals. In case of the saturated signals saturation can be simulated for all the parts in the trigger tower chain where it can occur.
2. Ptolemy simulation: The Ptolemy tool from the University of California at Berkeley allows discrete-event as well as synchronous data-flow modeling. This allows to building up complex simulations using modules of both types. The modules are written with the object oriented C++ language. To be able to check the complexity of the algorithms and their interplay, Ptolemy was used to create a model of the complete data path as it is implemented in the ASIC. This allows simulation of the efficiency of the algorithms depending on different input signals (Pspice outputs) and under different conditions including noise and time-jitter effects on the signals.

The second task of the Ptolemy simulation is to produce,

in addition to the digitized input data for the PPrAsic, the expected output data for comparison
3. NC-Verilog simulation: This simulator is needed for the necessary checks of the correctness of the hardware description written in Verilog.

The necessary data exchange between the simulations is done by Perl scripts. They glue the different parts to a flexible and complex system that can be used in all the steps of the design process.

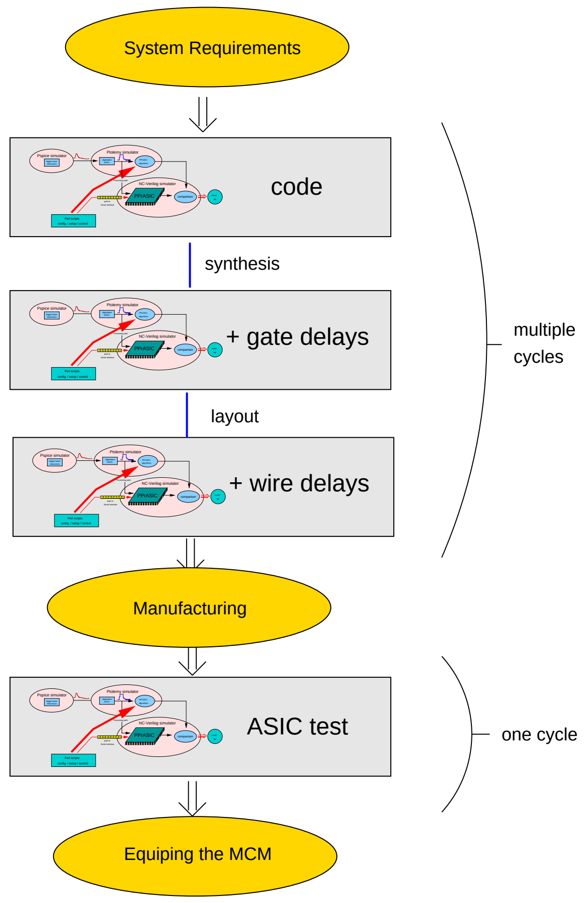


Fig. 2. Design flow of the PPrAsic. The verification process is illustrated on each level of the design process, where different timing information is present.

B. Design Flow

The combination between Pspice, Ptolemy and NC-Verilog simulation allows an automatic test for multiple configurations of the PPrAsic. The digitized data that are produced by the Ptolemy simulation are fed into the NC-Verilog simulation. Then the results from both simulations are compared automatically by a Verilog module. This builds up a flexible system for automatic tests that can be utilized in all four steps of the design flow as shown in figure 2.

In the first step the simulation environment is used to develop the algorithms for the BCID and their interplay. Then it can be used to check the matching of the hardware description with the simulated algorithms. The check of the netlist, which includes the gate delays, and the layout, which includes also wire delays, can also be done with the simulation environment. After the first three steps, which are passed through several times, the manufacture will take place. The simulation environment has the flexibility to be useful also in the real hardware test. It produces the input vectors and the expected output data for specific test runs.

C. Simulation Setup

To run the Ptolemy and the NC-Verilog simulation in parallel for an automatic comparison it has to be guaranteed that they work on the same PPrAsic configuration. Therefore a Perl script is used. It uses as input ASCII files where the register definition, the configuration of the registers and the default values of the registers are stored. Then it creates the right data stream which configures the PPrAsic through its serial interface. The function of the script is illustrated into figure 3.

The system integration is done by software that was developed in Heidelberg. It is called Hardware Diagnostic, Monitoring and Control Software (HDMC)[6] and its task is to access all the hardware that is produced for the system. It generates the register definition for the hardware description of the ASIC and provides a direct access to them and the memories of the produced hardware.

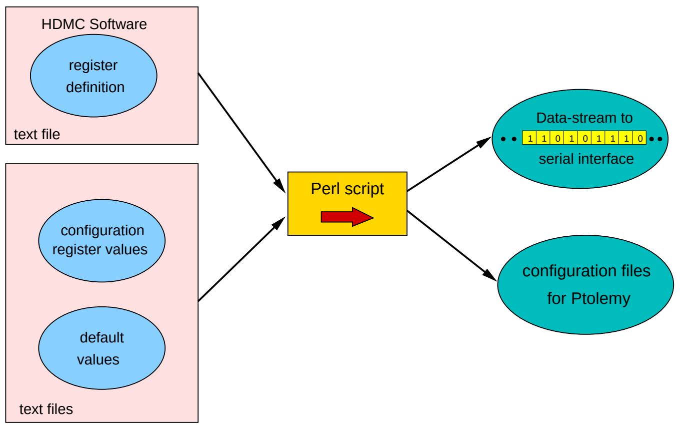


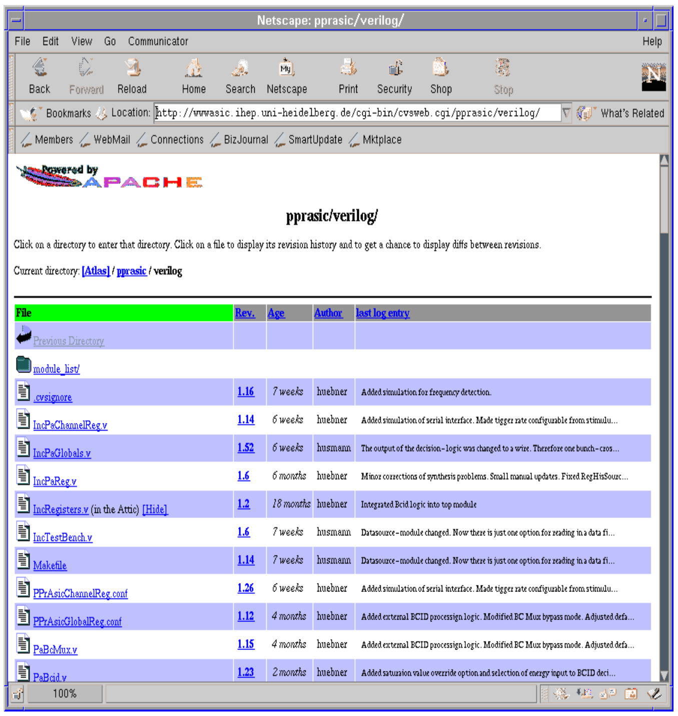
Fig. 3. The creation of stimulus vectors for the serial interface of the PPrAsic and the synchronous production of the configuration files for the Ptolemy simulation

IV. Documentation

The development of an ASIC by different designers at the same time makes it necessary to agree about some conventions which allow building up a design from pieces done by different people. This includes also the aspect of design re-use, where design blocks orginating from other projects are imported. The use of appropriate conventions minimizes the effort to fit these blocks into a design. It also facilitates the introduction of new designers, which can profit from the experiences on which these conventions are based.

Besides these necessities all the tools and procedures used for the PPrAsic design and all information about design entry, synthesis and layout have to be described. All this is done by a document called *Pre-Processor Asic Design Guide*[7]. It also includes all the information about the simulation tools and their interactions.

A design as flexible and complex as the PPrAsic needs to be very well described in order to allow, years after the design process, faultless operation by different people than the designers. All the information about the design itself with registers, input and output pins and its internal protocols have to be explained in detail. It is also necessary to describe the correct handling of the serial interface and the Jtag interface as well as the functioning of the readout and the control logic. All this is done in the documentaion *Pre-Processor Asic User and Reference Manual*[8]. In addition to that infomation the documentation describes also the working of the algorithms used for the bunch-crossing identification.



http://wwwasic.ihep.uni-heidelberg.de/cgi-bin/cvsweb.cgi/pprasic/verilog/

pprasic/verilog/

Click on a directory to enter that directory. Click on a file to display its revision history and to get a chance to display diffs between revisions.

Current directory: [Atlas] / pprasic / verilog

File	Rev.	Age	Author	last log entry
Previous Directory				
module_list/				
.cvsignore	1.16	7 weeks	huebner	Added simulation for frequency detection.
IncPaChannelReg.v	1.14	6 weeks	huebner	Added simulation of serial interface. Made tigger rate configurable from stimulu...
IncPaGlobals.v	1.52	6 weeks	husmann	The output of the decision-logic was changed to a wire. Therefore one bunch-cros...
IncPaReg.v	1.6	6 months	huebner	Minor corrections of synthesis problems. Small manual updates. Fixed RegHisSourc...
IncRegisters.v (in the Attic) [Hide]	1.2	18 months	huebner	Integrated Bcid logic into top module
IncTestBench.v	1.6	7 weeks	husmann	Datasource-module changed. Now there is just one option for reading in a data fi...
Makefile	1.14	7 weeks	husmann	Datasource-module changed. Now there is just one option for reading in a data fi...
PPrAsicChannelReg.conf	1.26	6 weeks	huebner	Added simulation of serial interface. Made tigger rate configurable from stimulu...
PPrAsicGlobalReg.conf	1.12	4 months	huebner	Added external BCID processign logic. Modified BC Mux bypass mode. Adjusted defa...
PaBcMux.v	1.15	4 months	huebner	Added external BCID processign logic. Modified BC Mux bypass mode. Adjusted defa...
PaBcid.v	1.23	2 months	huebner	Added saturaion value override option and selection of energy input to BCID deci...

Fig. 4. A screenshot of the web interface of the Concurrent Version System

A. *Code Managment*

Many designers working on the same code simultaneously leads to the necessity of documentation which follows the design process along the way. It has to handle different versions of the same files edited by different designers. This is done by the *Concurrent Version System (CVS)*[9]. This system holds the code in a central database called *repository*. If someone wants to edit the code he checks out a local copy to his disk. After editing he commits changes back to a repository adding a line commenting on the reason of the changes. CVS takes care of merging files and resolving conflicts arising from different people editing the same piece of code. It also keeps tracks of older versions. If changes are commited, no data are deleted. This way the designer can switch back to an older version if he likes and he can access the complete history of all the changes that lead to the actual version.

The fact that the ATLAS collaboration has members in different institutes requires an easy way of accessing the status of the design from outside. Therefore CVS allows colleagues who are directly involved in the process to access CVS from outside with rights of changing the code and uploading it. For other people who want to keep informed about the design process a web interface[10] exists that documents the complete history of the code. Figure 4 shows a screenshot of this web interface.

B. *The Logging of Simulations*

To keep the different designers informed about the simulations that have been done and their results, a Perl script was developed which creates a log-file for each simulation. After each simulation the designer is asked to enter a short line which summarizes the aim of the simulation and a short text describing the expected output vectors of the simulation. This is stored together with all the necessary stimulus files to set up the simulation and the complete output of it into a log file. The logfiles can be accessed via a graphical interface which gives the designers a complete overview about the simulations done and their code coverage.

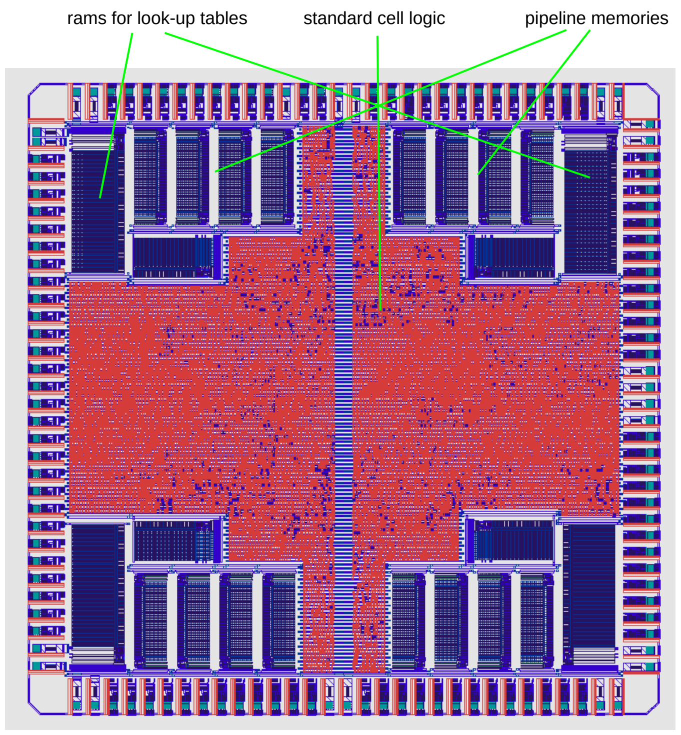


Fig. 5. The final layout of the PPrAsic. The dark areas are the memories

V. Results and Design Experience

The design environment described was used for the design of the PPrAsic. It has helped to meet the requirements and to manage the ASIC design process. It was successfully used to develop the integrated BCID algorithms. Therefore, the design environment combined the different neccessary simulators into one flexible system. The system showed its usefulness in all the steps of the design flow, from the code generation and verification over the netlist to the final layout. It was also used to prepare the chip test. With the design environment developed it was possible to save designer resources. The final PPrAsic could be completed in one fast design cycle. The requirements for a project with many designers were reached by complex code management based on the Concurrent Version System.

The final design consists of 34,000 standard cells, 66 kbit RAM and has a size of about 60 mm^2. A picture of the final layout is shown in figure 5.

References

[1] ATLAS Level-1 Trigger Group, *ATLAS First-Level Trigger Technical Design Report*, ATLAS TDR-12, CERN/LHCC/98-14, CERN, Geneva 24 June 1998. http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/TDR/tdr.html

[2] U. Pfeiffer et al., *ATLAS Level-1 Calorimeter Trigger System Architecture*, Fourth Workshop on Electronics for LHC Experiments, CERN/LHC/98-36, Rome, Italy 21-25 September 1998. http://wwwasic.kip.uni-heidelberg.de/atlas/publications.html

[3] Pspice V9 "Mixed A/D circuit simulator," *Orcad*, http://www.orcad.de

[4] Ptolemy 0.7.1 "Heterogeneous Modeling And Design," *University of California at Berkeley*, http://ptolemy.eecs.berkeley.edu

[5] NC-Verilog *Cadence Design Systems Inc.*, http://www.cadence.com

[6] C. Schumacher, "Hardware Diagnostic, Monitoring and Control Software", *Universität Heidelberg*, http://wwwasic.kip.uni-heidelberg.de/atlas/projects/hdmc.html

[7] D. Husmann, M. Keller, K. Mahboubi, C. Schumacher, *Pre-Processor Asic Design Guide*, Universität Heidelberg, 2000. http://wwwasic.kip.uni-heidelberg.de/atlas/docs/index.html

[8] D. Husmann, M. Keller, K. Mahboubi, C. Schumacher, *Pre-Processor Asic User and Reference Manual*, Universität Heidelberg, 2000. http://wwwasic.kip.uni-heidelberg.de/atlas/docs/index.html

[9] "Concurrent Version System", http://www.cvshome.org

[10] The web interface is reachable under: http://wwwasic.kip.uni-heidelberg.de/cgi-bin/cvsweb.cgi