Karlsruher Institut für Technologie

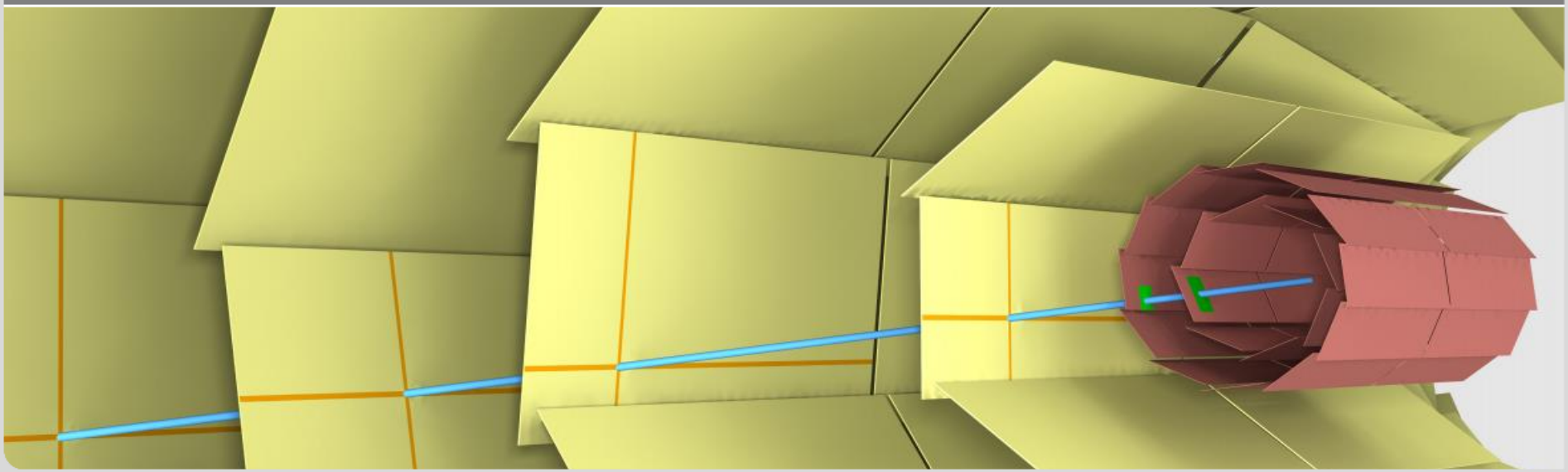
Institut für Technik der Informationsverarbeitung

Prof. Dr.-Ing. Dr. h.c. J. Becker
Prof. Dr.-Ing. Eric Sax
Prof. Dr. rer. nat. W. Stork

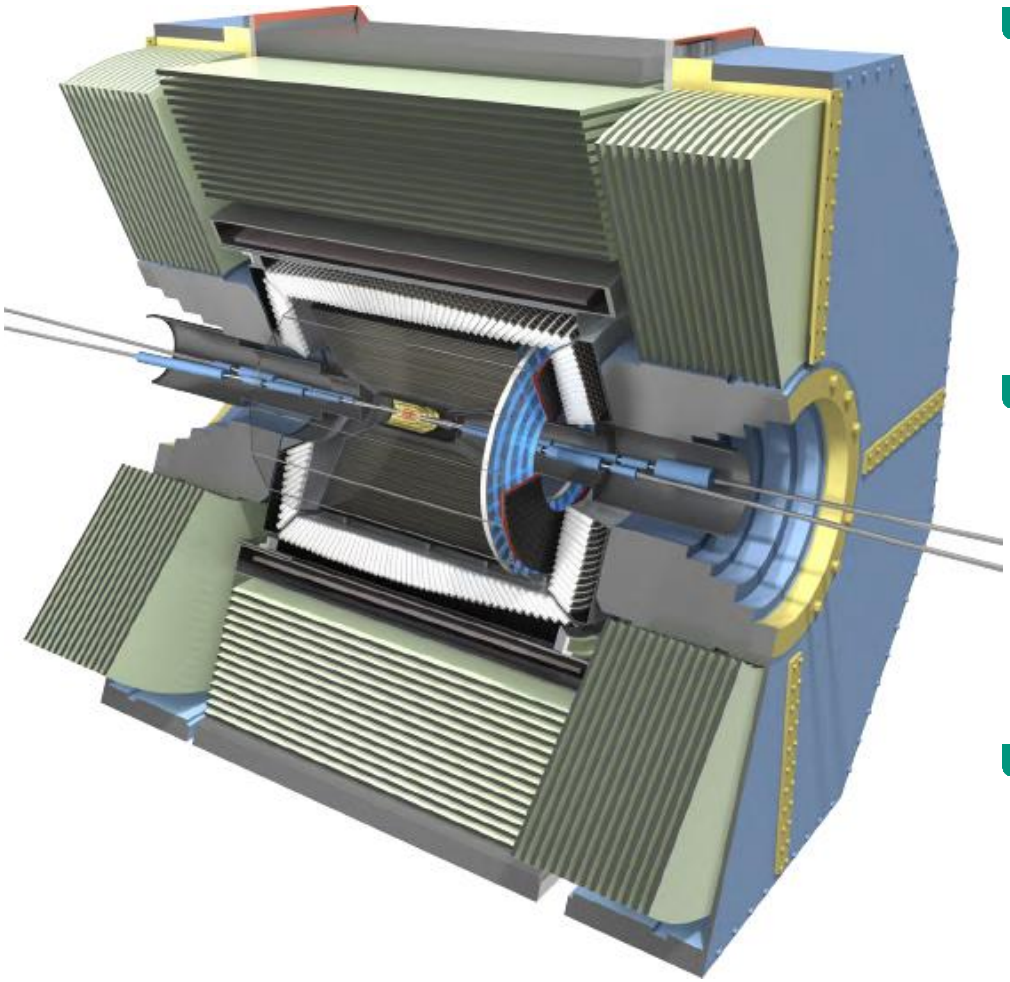
A framework for porting the NeuroBayes machine learning algorithm to FPGAs

Steffen Bähr

Institute for Information Processing Technologies (ITIV), Institute for Experimental Nuclear Physics (IEKP)

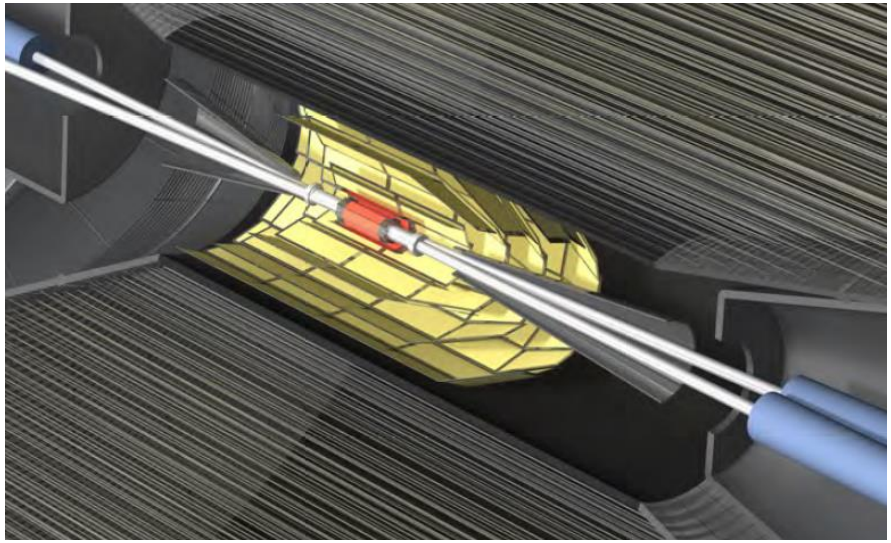


Data Flood in Belle-II



- Higher luminosities
- More data generated
- Pixel Detector generates around 1 Mbyte/event

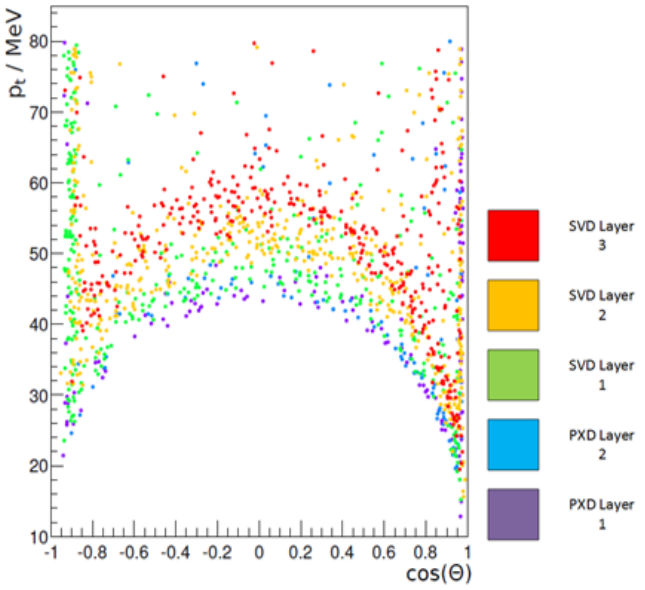
Pixel Detector Data Reduction in Belle-II



- Limited bandwidth to storage available
  - 100 Kbyte/event available, reduction of 90 % needed
- Separation predefined background from signal
  - Only signal is stored
  - Has to be performed online on FPGAs

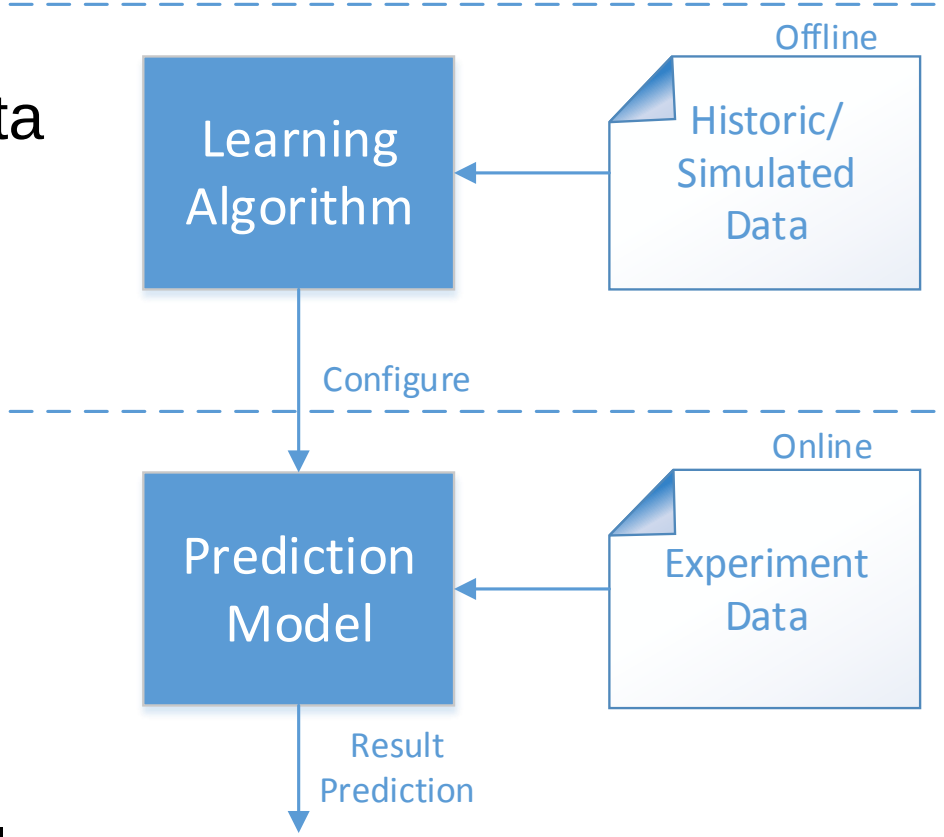
Online-Cluster-Analysis on FPGAs

- Identification of slow Pions (Pt < 65 MeV) at the Pixel Detector
  - Use properties of hit clusters
  - Processing on FPGAs close to the pixel detector
- Suppression of particles with low transversal momentum by track extrapolation
  - Slow Pions do not reach outer layers of the detector



Usage of Machine Learning for Identification

- Generation of a prediction model using simulated data
- Make predictions about correct classification
- NeuroBayes algorithm
  - Developed for particle identification
  - First Fortran now Python
  - Many configurations possible



Requirements for Online-Cluster-Analysis

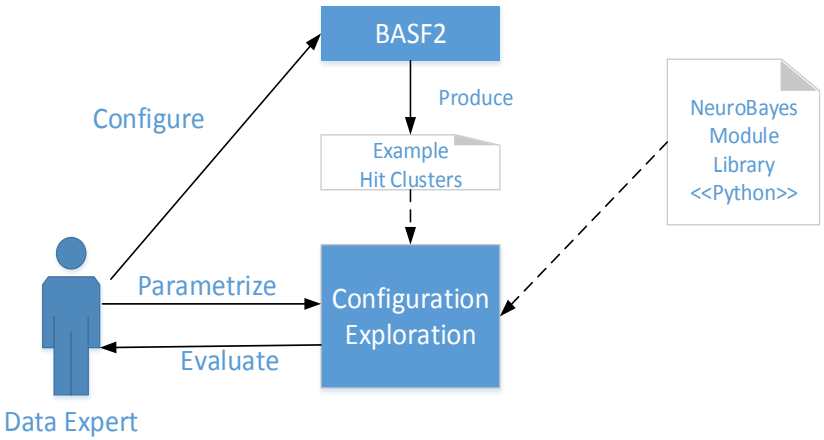
- Hard constraints on throughput, rejection rate and resources
  - 200 Mio clusters per second to be analyzed
  - 90% Reduction rate has to be achieved
  - 30 % of logic and 50 % of DSPs available on used FPGA

- Implementation has to be adapted over time
  - Adapt to new conditions of sensors
  - Find even more interesting particles

Need for an Automated Framework

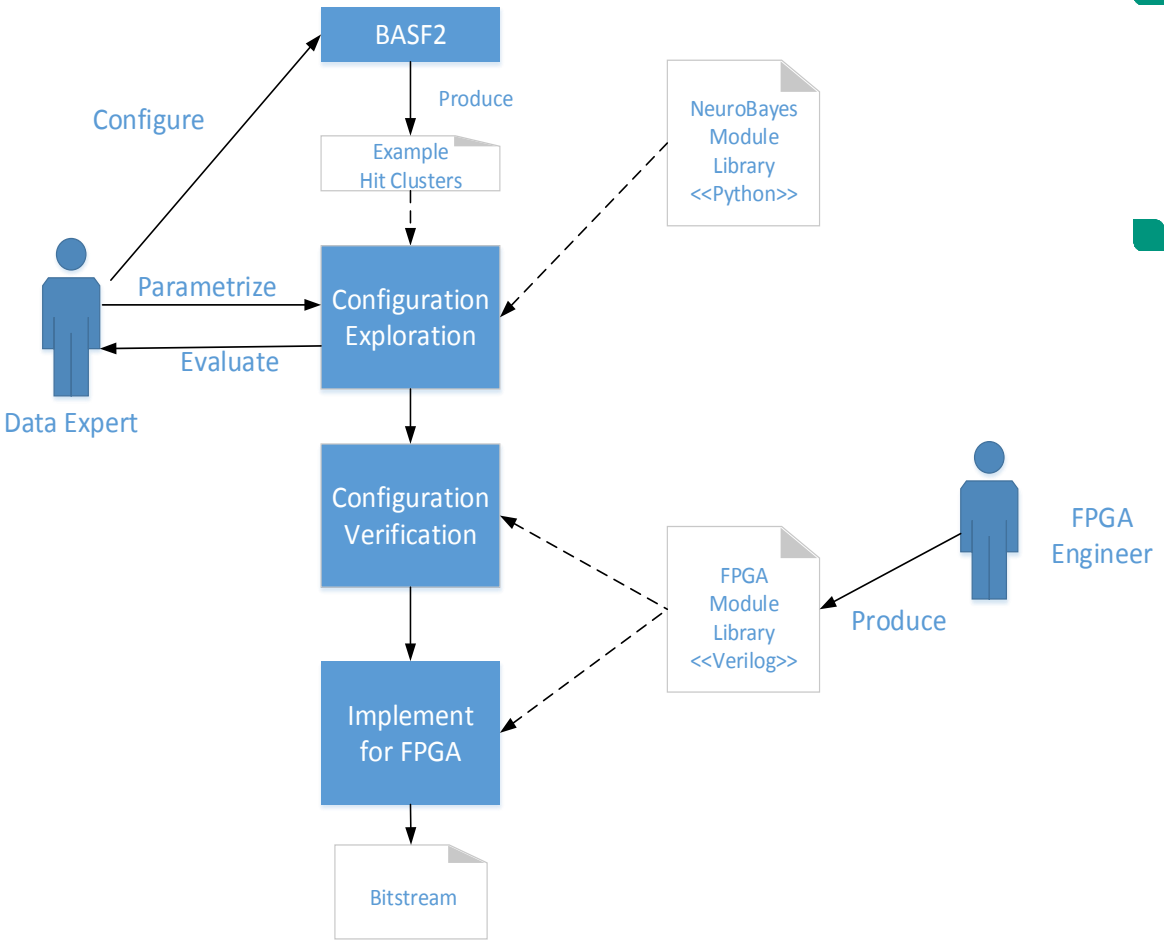
- Several possible configurations of the algorithm
  - Trade-Off between Performance-Resource-Quality
  - Exploration of the design space
- Bridge the gap between algorithm written in Python and FPGAs
  - Fix-Point representation necessary
- Optimal configuration depends on input data

Framework for Porting the NeuroBayes



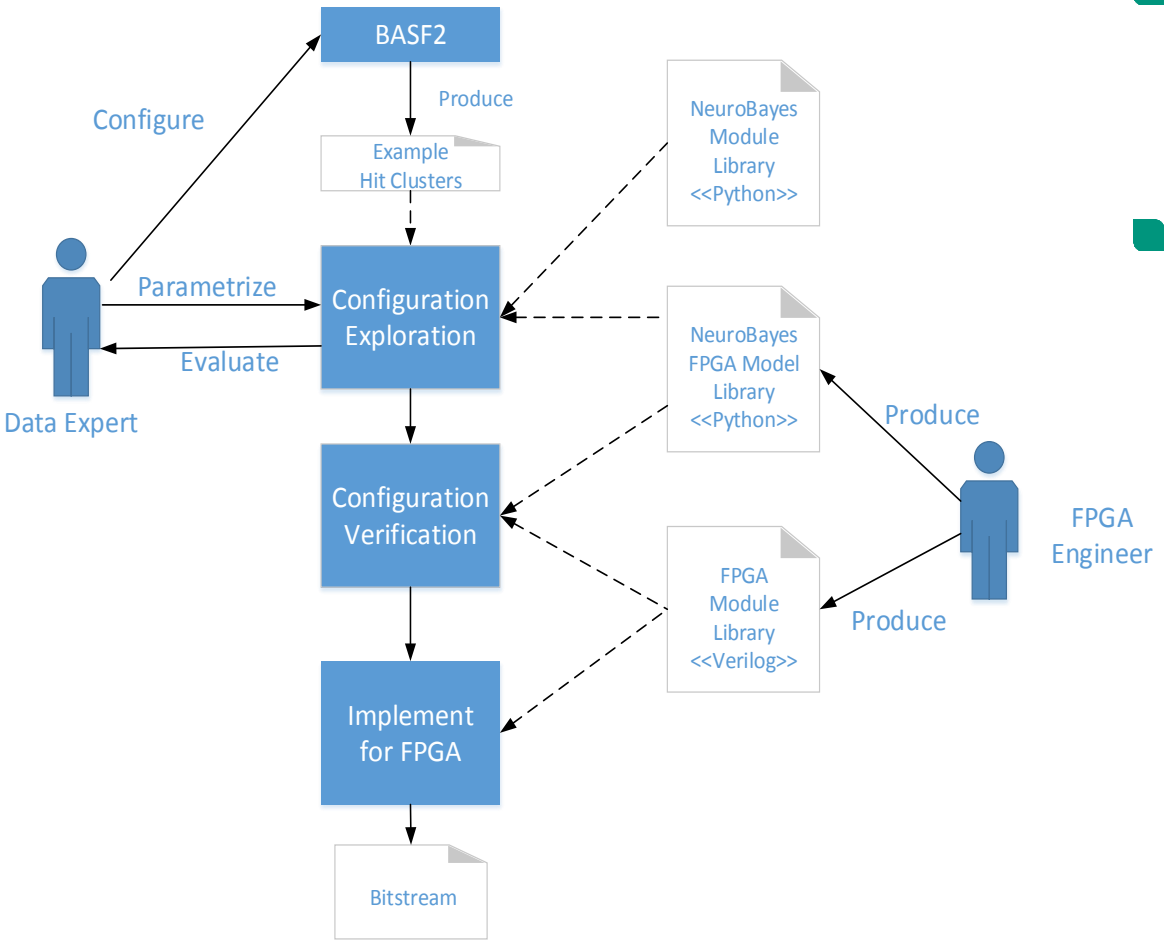
- Generation of simulated data in BASF2
- Configuration of algorithm selected by „Data Expert“
- No feedback of HW details

Framework for Porting the NeuroBayes



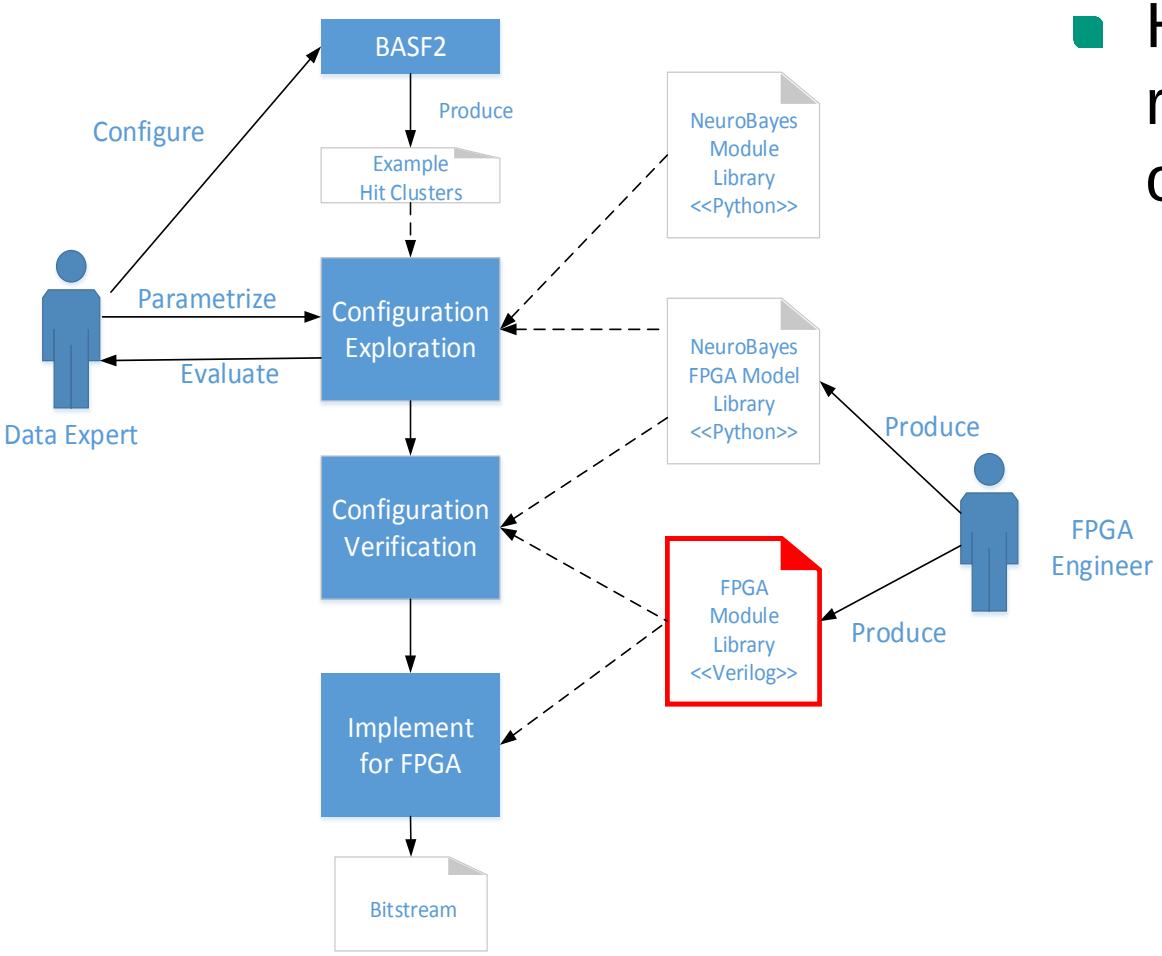
- FPGA Enigneer generates modules for implementation
- No communication between both, leads several iterations

Framework for Porting the NeuroBayes



- Models for FPGA processing Modules provided in Python
- Fast feedback to Data Expert possible

NeuroBayes FPGA Libraries

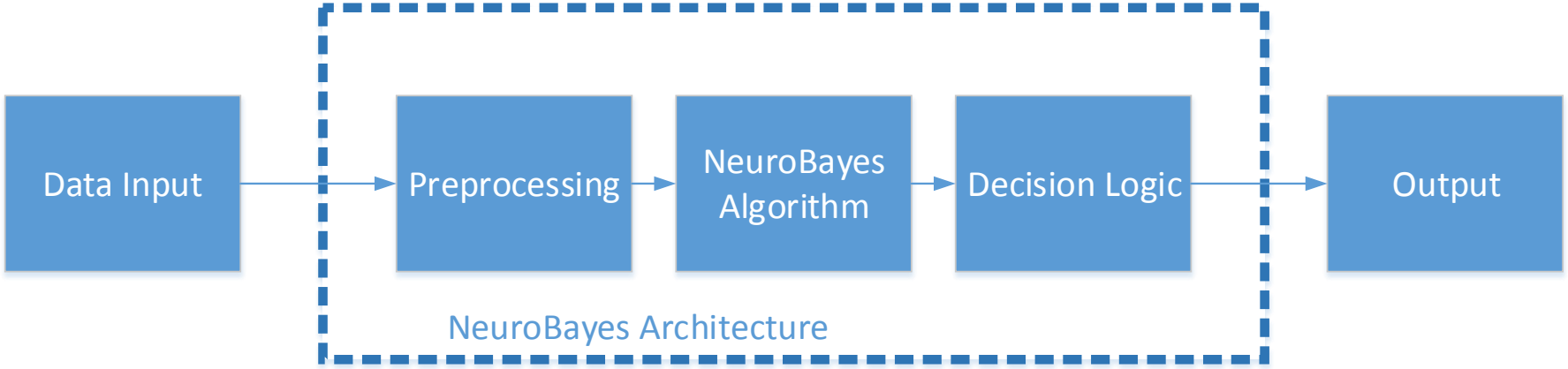


- Hardware modules realized in HDL and organized in Libraries

NeuroBayes Based Processing Architecture

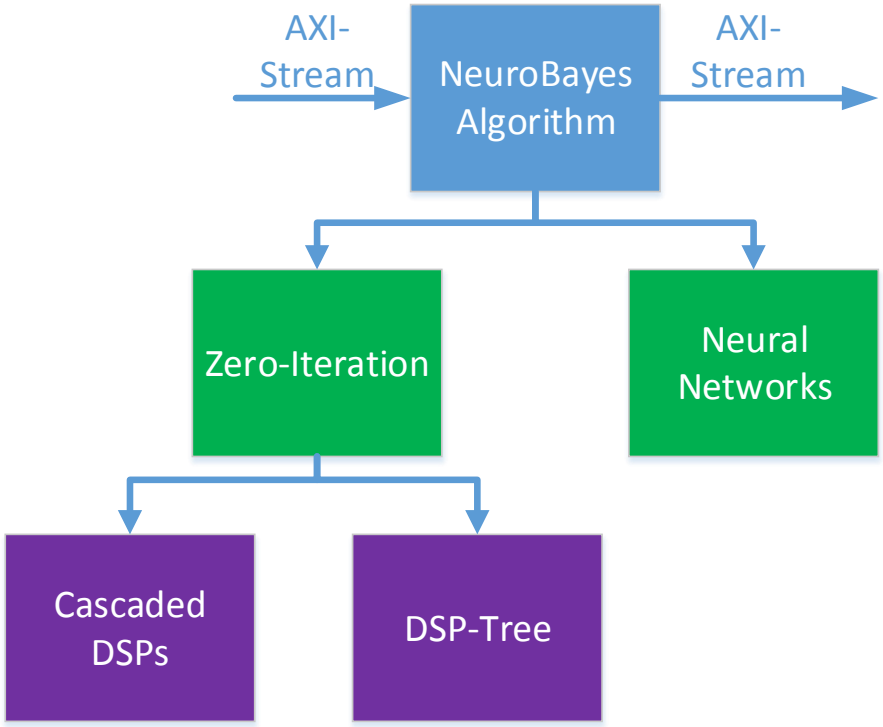
- Pipelined architecture three major architectural components present

- Every component has several possible configurations
  - Selected configuration of one influences the others



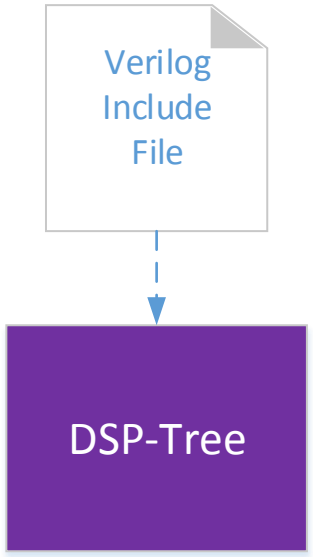
Module library

- Library of modules used together in NeuroBayes architecture
  - Implemented in HDL to achieve most efficient solution
  - Different implementations for same module possible
- Same architectural components with same interface
  - Different implementations and algorithms without changing connected modules

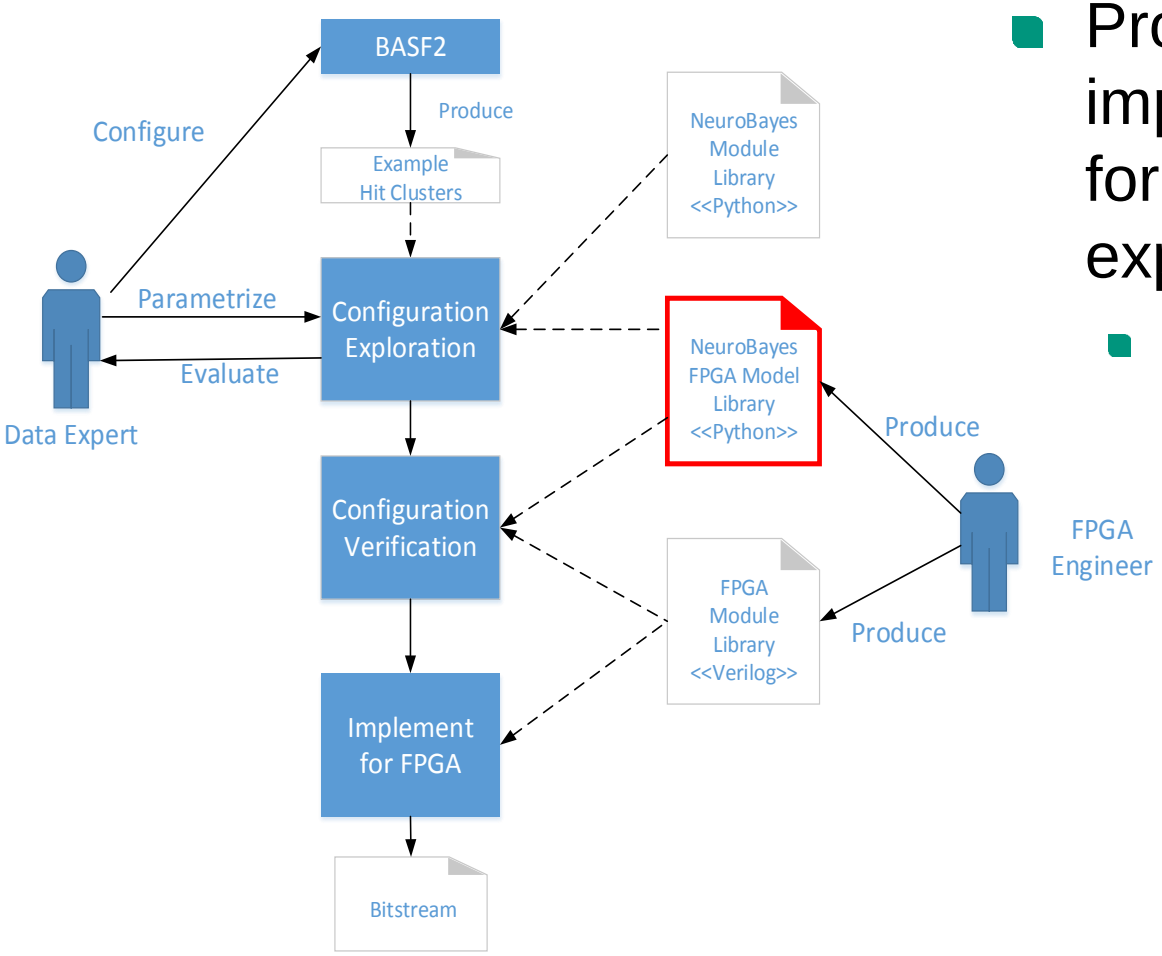


Configurable Modules

- Usage of packages for custom configuration of module
  - Only the packages have to be adjusted
  - Generated automatically after selection of configuration
- Configuration includes
  - Bitwidth of DSPs
  - Length of processing pipeline
  - Number of DSPs
  - NeuroBayes specific constants
  - ..



Configuration Exploration



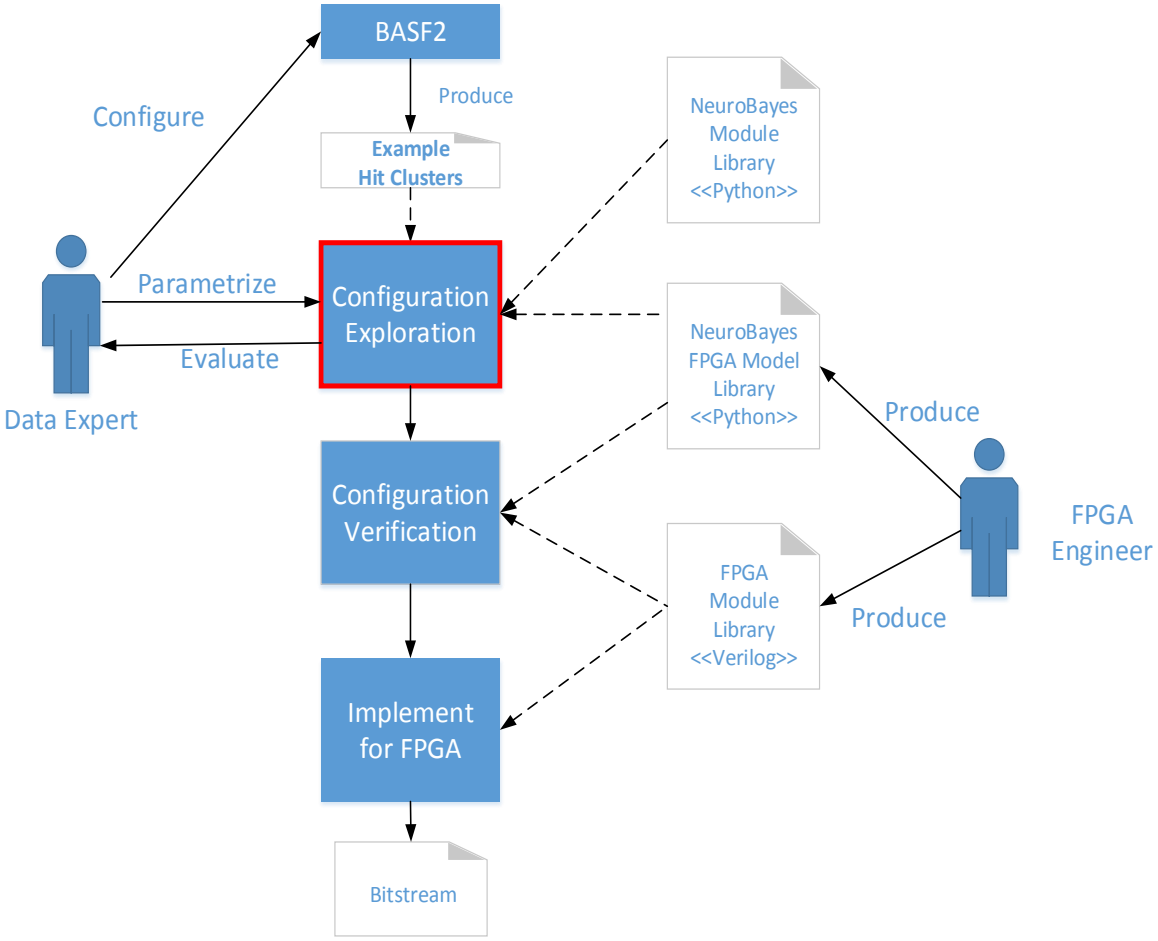
- Provide models of FPGA implementation in Python for usage in configuration exploration
  - Exchange model with SW implementation

MyHDL

- Python library that introduces types and operations typically used in HDLs
  - Bit vectors
  - Fix point representation and operations
- Allows Coupling of HDL with Python code
  - Python Verilog Co-Simulation with Modelsim possible
- Abstraction Level of synthesizable code around RTL

http://www.myhdl.org/

Exploration of possible configurations



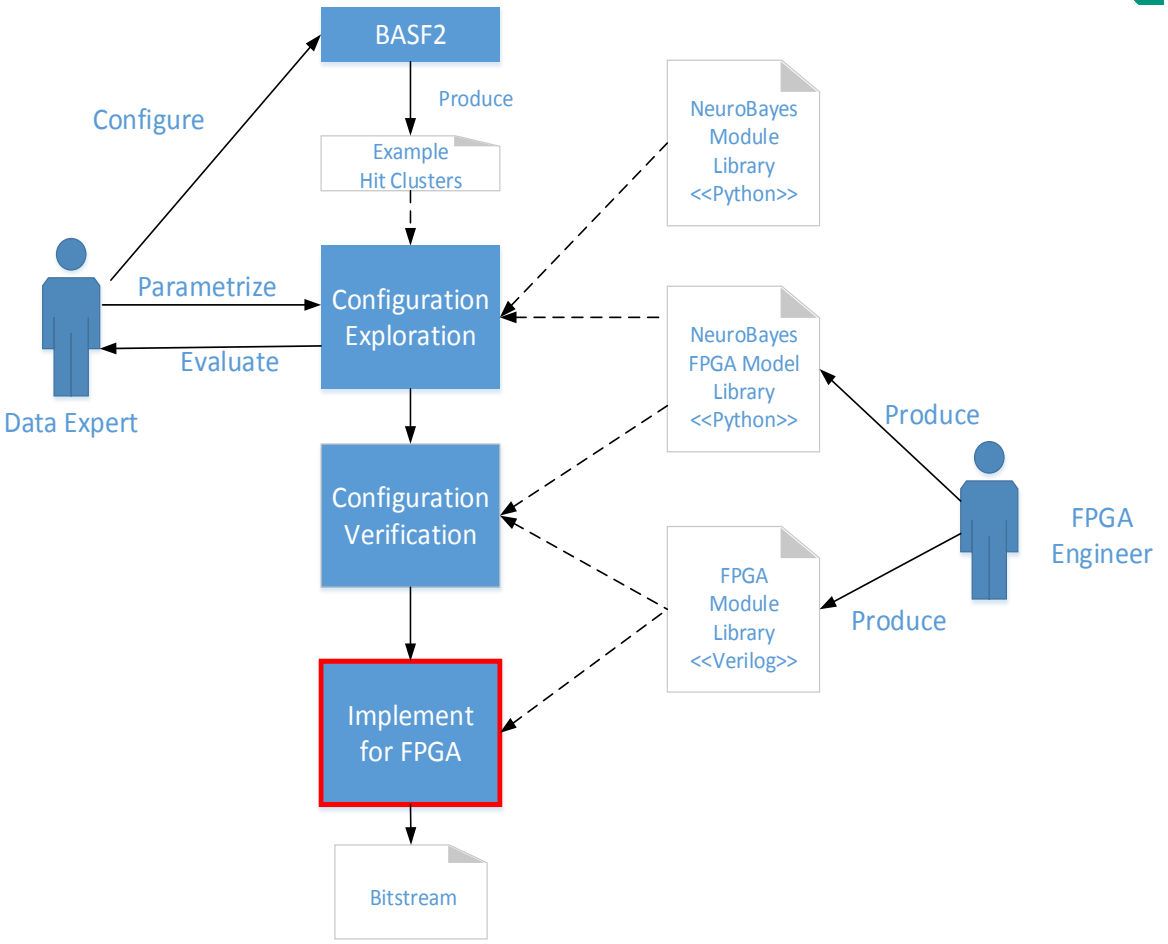
Exploration of suitable configurations

- Execute NeuroBayes on sample data with models of HDL components
  - No HDL Co-Simulation necessary
  - Higher abstraction level for faster exploration
- Unit Tests for validating the model against the reference
  - Already existing tests in Python
  - Bit Vector conversions necessary
- Quality of Service for each implementation provided
  - Throughput
  - Latency
  - Resource demand

Example Evaluation for Cluster-Analysis

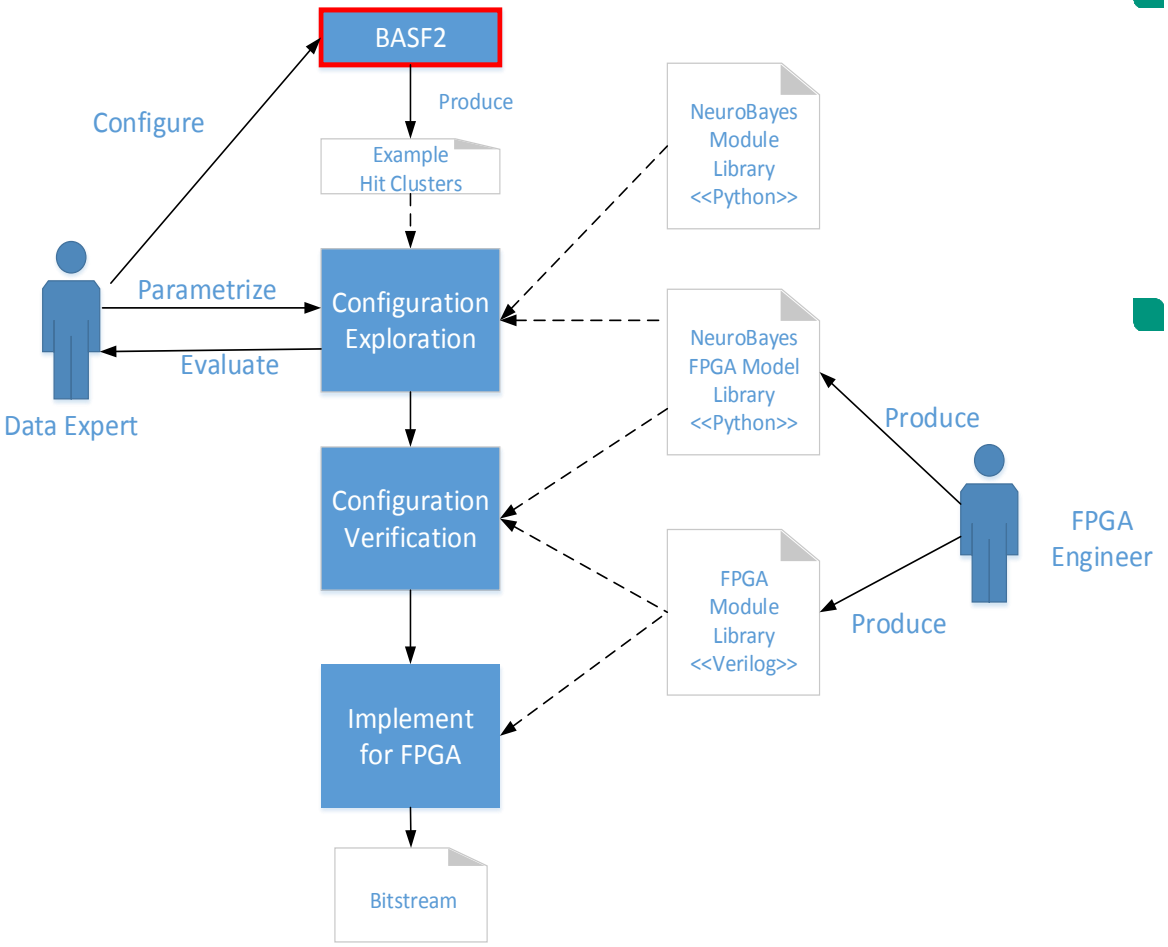
- Preprocessing configured for Binning with Interpolation
  - Implemented as Look Up Table
  - Latency : 1 cycle
  - Throughput : 440 Mio Cluster/s
  - Resources for Virtex6 VLX75T : ~ 3 % of Logic
- NeuroBayes configured as zero iteration
  - Implemented with cascaded DSPs
  - Latency : 11 cycle
  - Throughput : 350 Mio Cluster/s
  - Resources for Virtex6 VLX75T : ~ 2 % of DSPs
- Overall accuracy compared to reference in Software
  - Difference in prediction < $1.5 * 10^{-5}$
  - 92 % Rejection Rate

Creating a Complete HDL Design



- Generation of
  - Verilog header files for module configuration
  - Testbench for simulation
  - Top design file with Instantiations

Coupling with BASF2



- Update to match new signal and background definitions
- FPGA Engineer „only“ involved if something goes wrong

Outlook

- Framework was used to create current implementation of cluster-analysis for slow pion rescue on FPGAs
- Coupling with BASF2 for future updates of signal/background definition
- Usage of sophisticated exploration strategies
- Investigation of possible usage for estimation of the z-vertex in Belle-II

Thank you for your interest