Modeling event building architecture for the triggerless data acquisition system for PANDA experiment at the HESR facility at FAIR/GSI

K Korcyl\textsuperscript{1, 2}, I Konorov\textsuperscript{3}, W Kühn\textsuperscript{4} and L Schmitt\textsuperscript{5}
\textsuperscript{1} Institute of Nuclear Physics Polish Academy of Sciences, Kraków, Poland
\textsuperscript{2} Cracow University of Technology, Kraków, Poland
\textsuperscript{3} Technische Universität München, Physics Department, Garching, Germany
\textsuperscript{4} Justus-Liebig-Universität, Giessen, Germany
\textsuperscript{5} GSI, Darmstadt, Germany
E-mail: Krzysztof.Korcyl@ifj.edu.pl

Abstract. A novel architecture is being proposed for the data acquisition and trigger system of the PANDA experiment at the HESR facility at FAIR/GSI. The experiment will run without hardware trigger signal using timestamps to correlate detector data from a given time window. The broad physics program in combination with the high rate of \(2 \times 10^7\) interactions per second requires very selective filtering algorithms accessing information from many detectors. Therefore the effective filtering will happen later than in today’s systems i.e. after the event building. To assess that, the complete architecture will be built of two stages: the data concentrator stage providing event building and the rate reduction stage. For the former stage, which requires a throughput of 100 GB/s to perform event building, we propose two layers of ATCA crates filled with Compute Nodes - modules designed at IHEP and University of Giessen for trigger and data acquisition systems. Currently each board is equipped with 5 Virtex4 FX60 FPGAs and high bandwidth connectivity is provided by 8 front panel RocketIO ports and 12 backplane ports for the inter-module communication.

We designed simplified models of the components of the architecture and using the SystemC library as support for the discrete event simulations, demonstrate the expected throughput of the full-size system. We also show impact of some architectural choices and key parameters on the architecture’s performance.

1. Introduction
PANDA  \cite{1} is a next generation hadron physics detector planned to be operated at the future Facility for Antiproton and Ion Research (FAIR)  \cite{2} at Darmstadt, Germany. It will use cooled antiproton beams with a momentum between 1.5 GeV/c and 15 GeV/c interacting with various internal targets.

The antiprotons of FAIR will be injected into High-Energy Storage Ring (HESR), a slow ramping synchrotron and storage ring with excellent beam energy definition by means of stochastic and electron cooling. This allows to measure masses and widths of hadronic resonances with an accuracy of 50–100 keV, which is 10 to 100 times better than any \(e^+e^-\) collider experiment. In addition states of all quantum numbers can be directly produced in antiproton-proton annihilations. In the PANDA experiment, the antiprotons will interact with an internal...
target, either a hydrogen cluster jet or a high frequency frozen hydrogen pellet target to reach a luminosity of up to $2 \times 10^{32} \text{cm}^{-2}\text{s}^{-1}$ corresponding to up to an average event rate of up to 20 MHz. For reactions with heavy nuclear targets thin wires or foils are put into the beam halo.

The experiment is focusing on hadron spectroscopy, in particular the search for exotic states in the charmonium region, on the interaction of charm hadrons with the nuclear medium, on double-hypernuclei to investigate the nuclear potential and hyperon-hyperon interactions and on electromagnetic processes to study various aspects of nucleon structure.

2. Triggerless operation

Due to the rich physics program requiring high flexibility in event selection based on varying physics criteria it was decided to build a DAQ system operating without a hardware trigger signal. Instead, the front end electronics will operate in continuous sampling mode in which a detector signal reaching a threshold will generate a message. The messages will be assembled at the Data Concentrators (DCs) during a given period of time. At the end of the period a synchronized message (see next section) will be distributed to all DCs. The DCs will then combine collected data into time stamped messages which will be pushed into the DAQ system. It should be able to perform a burst building operation aimed to route all the messages with the same time tag to a single output. At the output, the fragments from all the DCs will be merged and the full burst info, without any compromise due to an earlier trigger decision, will be available for physics selection processing. It is expected that the total throughput required to transfer data from the current set of detectors will reach 100 GB/s.

3. Time distribution and synchronization system

The SODA (Synchronization Of Data Acquisition) project [3] aims to develop a versatile optical network system which is, first of all, able to provide a common reference time with a precision better than 20 ps R.M.S. In addition, the system synchronizes data taking with the burst structure and performs monitoring of data acquisition modules. Furthermore, it takes responsibility for data flow control. The core of the system is a point-to-multipoint bidirectional optical link which is able to broadcast information from a master module to several hundred destinations and to acquire information from the destination modules via a passive optical fiber network. The precise time is provided by mean of low jitter clock of 155.52 MHz and allows front-end electronics to determine timing of detector signals with high precision.

The burst structure, distributed by the SODA system, is used as an event for PANDA burst building network because data which may correlate between each other are always within one burst. The data concentrators as well as compute nodes are attached to the SODA system. They receive information about active run, burst structure and, what is also important, data pass and destination for every burst data blocks. This feature gives a possibility to balance load of compute nodes and bypass overloaded node or failed one. The SODA system links passing through the Data Concentrators will be used to forward configuration data to the FEE electronics.

4. ATCA crate and Compute Node

The FPGA based Compute Nodes [4] with multi-Gbit/s bandwidth capability using the ATCA [5] architecture are designed to handle tasks such as event building, feature extraction and high level trigger processing. Each board is equipped with 5 Virtex4 FX60 FPGAs. High bandwidth connectivity at the front panel is provided by 8 optical links connected to the RocketIO ports of the FPGAs - two bidirectional ports from each of the four central FPGAs. There are also 4 Gbit Ethernet links at the front panel which are not used for data transfers in our architecture. The module can also connect to 12 other Compute Nodes located in the same ATCA crate using
Figure 1. Photo of current version of the Compute Node. Clearly visible in the front panel the 8 fiber input ports for communication with either FEE or CPU and another layer of ATCA crates.

Figure 2. To provide connectivity between any processing node and any input from the FEE with high throughput the ATCA crate has to be equipped with the full-mesh backplane - example of 14-slots backplane.

full mesh of the ATCA backplane. The connection is provided via the 12 RocketIO ports of the fifth FPGA.

5. Push-only architecture

For the burst building task we propose a push-only architecture based on Compute Nodes (CNs) located in ATCA crates. The architecture is presented in Fig. 3. The data flow is unidirectional: from the front-end electronics towards the CPU processing nodes. We do not plan to send any signals in the opposite direction using the main CN links.

In absence of a hardware trigger signal, each FEE board monitors continuously input from detector channels. In case it crosses the programmed threshold, the change is signaled to the Data Concentrator (DC). The DC collects all the changes during a burst. The burst is related to the operation mode of the accelerator where a continuous flow of antiprotons hitting the hydrogen target produces interactions at a rate of up to 20 MHz. The duration of the burst is 2 $\mu$s and is followed by a 400 ns gap in interactions. At the end of each burst period a special signal will be distributed to all DCs. This signal will be used to group together all the changes recorded during the burst and to form a message which will be pushed into the DAQ architecture for burst building.

Messages in all DCs will be tagged with the same number which will be used in the switching network to guide packets to the common output port. The switching network is based on two stages of the ATCA crates filled up with the CN modules. Each module has 8 front panel bidirectional links. The nodes from the FEE stage will have 8 input links connected to the output from the Data Concentrators and 8 output links leading to the lower CPU stage. The modules at the CPU stage will receive data via 8 input links from the upper stage and forward messages to the CPU nodes using 8 output links. In Fig. 4 we present the wiring scheme between the FEE stage and the CPU stage. For simplicity only 3 crates (out of 4) from the FEE and CPU stages are shown. The main requirement which the switching network has to fulfill is to provide connectivity between any FEE input to any of the CPU output. We propose that four pairs of output links from a module in a given slot at the FEE level will be connected to pairs of input links in all four modules in the same slot number at the CPU level. Such a wiring allows to split the traffic into two streams: using the backplanes at the FEE level to reach desired slot position and then change the levels and outbound to the desired PORT or outbound to proper
CPU crate at the origin slot position and then use the backplanes at the CPU level to reach the desired slot position.

**Figure 3.** PANDA DAQ architecture. Data from front-end is merged in Data Concentrators and pushed forward into two stage burst merging switching network based on Compute Nodes located in ATCA crates. Concatenated fragments with the same time-stamp tag from all Data Concentrators are available for selection based on physics.

**Figure 4.** Wiring between Compute Nodes in two stages of the switching network. Four pairs of output links from a module in a given slot number at the FEE level reach four modules at the same slot number at the CPU level. Odd numbered events will use backplanes at the FEE level to move to the slot number of the CPU and then outbound via the output port. The even numbered events will first go to the CPU level by the output links and then use the backplane to reach desired slot number.

### 6. Model of the architecture

We designed and built a model of the full architecture to demonstrate that it will be able to switch an expected data flow of 100 $\text{GB/s}$ produced by the DCs. The model implements discrete event simulation and uses SystemC [6] libraries to handle the time steps. We implemented simplified models of the architecture components with more focus on internal communication between Virtex chips in the Compute Node.

The architecture is composed of 416 Data Concentrators sending data via 416 input links to 52 Compute Nodes spread evenly in 4 ATCA crates (13 Compute Nodes in each crate) at the FEE level. The Compute Nodes from the FEE level connect to the same number of Compute Nodes at the CPU level located in another 4 ATCA crates. Each Compute node from the FEE level connects to 4 CNs at the CPU level using the wiring scheme presented in Fig. 4. The output links from the CNs at the CPU level connect to 416 Data Sinks.

**6.1. Model of communication port**

As the main aim in modeling the PANDA DAQ is to demonstrate scalability of the proposed architecture we developed a generic model of input and output communication ports, which are then used by models of other components exchanging data in the system. The models use FIFO structures and have rich monitoring extensions which allow to inspect queue occupancy and link utilization. The simplified organization of the two ports is presented in Fig. 5. The models implement a store-and-forward way of operation waiting with forwarding a message until it is fully received. All link connections are point-to-point and the board’s links become Virtex links.
Figure 5. Simplified architecture of communication ports. In the output port the SendFifo can grow if multiple parallel writes enabled or link speed slower than transfer from board or later packets smaller than the former ones. In the input port the ReceiveFifo can grow if head packet can’t be read out due to busy destination or the former packets from the link is larger than the later ones.

In the output port a message can be written into the SendFifo if the number of enabled parallel writes does not exceed the limit (in case of a single write the limit is set to 1). When the write completes a check is made whether the size of the newly added message exceeds the capacity of the FIFO and the message is dropped in case of overflow. When the output link is not busy with sending out a former message the newly added messages start their transfer immediately.

When a message arrives from the link, the input port checks if the current FIFO occupation exceeds the capacity of the FIFO and in case of overflow the newly received message is dropped. If the messages is the only one in the FIFO the interrupt signal is generated. The input port manager consults the routing tables and establishes an output port to which the message should be transferred to. If writes to the output port are allowed, the transfer starts immediately. In other case the input port manager registers itself in the output port manager to get notification when the transfer of the messages becomes possible.

6.2. Data Concentrators and Data Sinks
The model of the Data Concentrator uses two parameters to produce a data message: the number of bytes produced at a single interaction and a number of bursts for merging data. The Data Concentrators use an exponential distribution to establish a number of interactions, with the average rate of 20 MHz, that have been generated during the $2 \mu s$ burst. The number is common for all Data Concentrators, thus each FEE link to the CN receives the same amount of data. In the model we assume the usage of the SODA time and synchronization system, and all the Data Concentrator output simulated data at the same time.

The Data Sink model takes only one parameter defining a number of fragments expected to complete the event building. The model allows to monitor event building latency but also event collection time which spans from the moment the first packet of the new burst arrives to the moment when the all expected packets reached the destination.
Figure 7. Communication links between FPGA Virtex (Vx) chips installed in the Compute Node. Virtex0 (V0) relays all communications to/from the backplane. Each of the remaining Virtexes (V1-V4) houses two input and two output ports from the front panel. All Virtex-es are interconnected between themselves with bidirectional links. The links between Virtex-es are labeled with numbers used as indexes in the throughput plots. The backplane links include the loop-back link which is not used in the data transfers.

6.3. Compute Node

We developed a simplified model of the Compute Node including internal communication links between Virtex chips (V0 - V4) depicted in Fig. 7. The node connects to the outside via 8 input, 8 output and 12 backplane links. The bidirectional backplane port implements the input and the output ports. Each of the Virtex 1-4 connects to 2 input and 2 output ports whereas the Virtex0 feeds all communication to the backplane ports.

The architecture should provide connectivity between any input at the FEE level with any output at the CPU level. The design task included the wiring scheme (described in section 5) which then was followed by a definition of the routing algorithm used to transfer data packets between ports of the FPGAs. The routing decisions depend on the location of the CN board - either the FEE or the CPU level and on the ID of the Virtex. We assumed that the CPU number (in range from 0 to 415) will be used in the burst message as identifier for the destination port. We aimed to explore huge backplane bandwidth at the both levels and split the traffic depending on the parity of the destination CPU number:

- FEE level: The odd numbered bursts should first go via the backplane to the slot at the same position as the destination CPU is attached to and then use the proper output port to get to the module with the destination CPU. The even numbered bursts will go to the output port leading to the CPU crate where the destination CPU is attached to.
- CPU level: The odd numbered bursts should go to the output port where the destination
CPU is attached to. The even numbered bursts should first go via the backplane to the slot where the destination CPU is attached to and then use the proper output to reach the destination CPU.

The routing inside the Virtex-es depends on the ID of the Virtex. In case of V1-V4 the messages heading for the ATCA backplane will be sent to the ports with links to the V0; the messages to one of the board’s output ports will be sent to the port which connects to the Virtex managing the desired output port.

7. Simulation results

We ran a series of simulations of the full size system with 416 Data Concentrators and 416 Data Sinks to verify whether the proposed architecture meets the requirements for the PANDA DAQ system. In the simulations all links were configured to offer 6.5 Gb/s throughput (like the ROCKET I/O from Virtex).

Figure 8. Evolution in time of event building latency with two fragment sizes corresponding to total throughput of 100 GB/s and 177 GB/s for two burst sizes: 1 and 10. The latency stays constant during the simulation time signifying ability of the architecture to handle both throughputs.

Figure 9. Distribution of the event building latency for each CPU for the 10-bursts events and 100 GB/s total throughput. The zoomed part shows a pattern between the distributions which originates from the way of assignment of the CPU for the next event \( N = N_{\text{previous}} + 79 \).

The model was using an exponential distribution for each burst to establish the number of interactions. The number was then propagated to each Data Concentrator and used to calculate initial size of the data packet. The size was then enlarged by 25% to take into account the 8b/10b coding and a small control overhead. All DCs were using the same scale factor, hence all fragments for a given burst had the same size, but the size was different between bursts. At the inter burst gap the signal was generated to all DC to push the prepared data packet into the architecture. The Data Sinks were configured to collect 416 fragments with the same timestamp tag to measure the burst building latency.

In our simulations we evaluated the possibility to group a number of bursts into a super-burst to reduce the burst rate at the destination CPU. Building an event from a single burst (2.4 \( \mu \)s) generates 1 kHz rate for each CPU with the average size of 250kB. Grouping 10 bursts at the Data Concentrators into a single fragment reduces the event rate to 100 Hz per CPU but increases the event size to 2.5 MB. The larger fragments pushed into the network require larger buffers to hold data at congestion points where links or ports are occupied longer.

As switching of the 100 GBps fills-up the 416 6.5 Gb/s input links only in 30% we run simulations with increased amount of data to reach 70% of link capacity, what corresponds
to switching of 177 GBps. In Fig. 8 we present the latency evolution in time for two global throughputs: 100 and 177 GBps and for two bursts grouping schemes: with a single burst and with super-burst of 10 bursts. During the simulation time the average latency stays constant which signifies the ability of the architecture to switch the required throughput. All the following results from simulations refer to the 10 bursts operation mode.

**Figure 10.** Evolution in time of links utilization between modules at FEE and CPU crates. The utilization was averaged over the links in all FEE modules with the same output port index. Equal values of link occupation for all ports confirm properly balanced routing at the FEE level.

**Figure 11.** Evolution in time of maximal receive queue size at the CPU level. The sizes were averaged over all CPU modules with the same input port index. The two times smaller values at index 2 and 3 originate from the algorithm used to calculate the CPU number for the next burst.

Fig. 9 shows the distribution of the event building latency at each of the 416 CPUs. The small differences between nodes in distribution and visible pattern originate from the formula which was used to calculate number of the destination CPU for the next burst \( N = N_{\text{previous}} + 79 \).

Running the simulations gives an unique opportunity to analyse network traffic patterns in places usually inaccessible for measurements. Every 100 ms the model dumps full statistics from all ports and links. In Fig. 10 we plot average utilization of the links between the modules at the FEE and the CPU levels. During the first 10s of simulation time the link's utilization was stable at around 38% of the 6.5 Gb/s link's throughput. In Fig. 10 values from all output links with the same index from the CN modules at the FEE level were averaged. The same value for all ports indicates that the routing scheme at the FEE level does not prioritize traffic in any direction.

The plot in Fig. 11 shows the evolution in time of maximal queue size at the CPU level for the first 10s of the simulation time. The maximal size increases from roughly 500kB to 600 kB during the first two seconds but then it flattens. The two ports with indexes 2 and 3 manifest two times smaller maximal sizes which is related to the algorithm selecting the CPU for the next burst.

As the CN model includes communication links between Virtexes it became possible to inspect the traffic patterns between the chips. In Fig. 12 we present the evolution of maximal length of the input queues during the first 10s of simulation time. The monitored queues develop in ports connecting the chips in the CN module at the CPU level. The pattern of maximal queue values is the same in all CN modules. The zoomed picture shows that the queues for the first 8 ports have half of the size of the remaining 12 ports. The differences in the queue occupation represent dynamics of the packet flow and originate from the algorithm selecting the CPU for the next burst. To identify individual ports please consult the Fig. 7 where port indexes are
shown in small boxes attached to links between the Virtex chips.

In Fig. 13 we show utilization of the links between the Virtex chips in the CN node at the CPU level. The links were monitored every 100 ms during the first 10 s of simulation time. The first 8 links between Virtex1-4 and Virtex0 are used to transfer packets between the board and the backplane. As half of the traffic goes to the backplane and packets to any of the Virtex1-4 arrive via two input links, then the backplane link occupation corresponds to the occupation of a single link between the levels. The remaining 12 links between Virtex1-4 are used to transfer the packets to the Virtex chips which handle the output ports leading to the target CPU. As each Virtex chip handles 2 output ports, a quarter of the intra-module traffic (half of the traffic entering the board as the other half goes to the backplane) goes via each link connecting the Virtex1-4 chips (8% of the full link throughput).

**Figure 12.** Evolution in time of the averages of maximal queue sizes in input ports of Virtex chips at the CPU level. The structure visible in the zoomed part shows queue dynamics related to the algorithm used to select the next CPU for the next event.

**Figure 13.** Evolution in time of the Virtex links utilization at the CPU level. The structure visible in the zoomed part shows three times larger load on links transferring traffic to/from the backplane (see text for more explanation).

### 8. Conclusions
The DAQ system for the PANDA experiment running without a hardware trigger signal requires a novel approach to perform event building at rate of a 100 GB/s. We designed the push-only architecture based on ATCA standard and Compute Nodes - modules with programmable logic (Virtex) designed for Panda and currently used for BESIII and Hades experiments. The architecture consists of two layers of ATCA crates and provides connectivity between any input from the data Concentrators and Front End Electronics at the FEE layer to any processor attached at the CPU layer.

To evaluate the validity of the proposed architecture we designed and built simplified models of the components in discrete events domain, coded in C++ and used SystemC libraries to handle simulation time. The simulation of 1 second of the full size system takes 5 hours of 2.4 GHz Intel CPU.

Simulation results of the full scale system indicate that the proposed architecture meets the requirements and can perform event building with aggregated throughput of 100 GB/s. With the input links loaded up to 70% of their nominal throughput of 6.5 Gbps the throughput of the architecture reaches 173 GB/s.
References
[1] Technical Progress Report - www-panda.gsi.de/db/papersDB/PC19-050217_panda_tpr.pdf
[2] FAIR project at GSI - www.gsi.de/fair/index_e.html
[3] I. Konorov et al. "SODA: Time distribution system for the PANDA experiment", IEEE Nuclear Symposium Conference Record (NSS/MIC), 2009 IEEE, p. 1863-1865
[4] W. Kühn et al. "FPGA based Compute Nodes for High Level Triggering in PANDA", CHEP'07, JoP Conference Series 119 (2008) 022027
[5] Advanced Telecommunications Computing Architecture - www.picmg.org
[6] SystemC - www.systemc.org.