PERFORMANCE TESTS OF DIGITAL SIGNAL PROCESSING FOR GSI SYNCHROTRON BPMS*

K. Lang[1], P.Forck[1], T.Hoffmann[1], G.Jansa[2], P.Kowina[1], U.Rauch[1]
[1] Gesellschaft für Schwerionenforschung, Beam Diagnostics , Darmstadt, Germany
[2] Cosylab D.D., Ljubljana, Slovenia

Abstract

The Beam Position Monitoring System at GSI heavy ion synchrotron SIS18 consists of twelve shoe-box PUs. Each of the four BPM plates is pre amplified and connected to a Libera Hadron unit from I-Tech Company for digitization and position calculation. The raw data of one BPM sampled by 125 MS/s with 14 Bit ADCs are reduced to about 20 MB/s by the onboard FPGA, using digital filter algorithms, resulting in a bunch-by-bunch position readout. In addition, different timing signals with various requirements are used to verify the functionality of the FPGA algorithms. For a closed orbit measurement, the data of all twelve Liberas have to be read in parallel. A built in Xilinx Rocket IO is used for data transport, which allows up to 1GBit/s data output. Over a dedicated network, the data are merged for further usage on a high performance PC. We describe the general architecture, parts of the FPGA design implementation and present first performance tests.

INTRODUCTION

At present a BPM upgrade program was started at GSI with the goal to implement different measurement modes for the detection of the beam position. This system shall also be used at the FAIR SIS100 and therefore upwards scaleable because of the higher amount of BPM stations.

The measurement modes include closed orbit, turn-by-turn, bunch-by-bunch and also tune measurement in horizontal and vertical plane. The desired resolution is 0.1mm. In SIS18 the accelerating RF varies with large dynamics from 850 kHz to 5 MHz. SIS 18 is operated with four filled buckets. Position calculation is done inside Liberas Xilinx Virtex II Pro FPGA[1] with a filter algorithm to generate the integration windows for the bunch signals.

To allow the different measurement modes without interference each other, all bunch positions from all BPMs will be concentrated at two server PCs in one accelerator cycle. The operator has then the possibility to choose, which measurement mode he wants to display. For controlling, Front-End Software Architecture (FESA) [2] from CERN is used and adapted to the requirements of the GSI BPM System [3].

PERFORMANCE TEST

Hardware Setup

All analog signals of the pick-ups are transmitted via long cables to a single electronics room. Thus all Liberas can be placed in one 19” rack (Fig. 1). To build a dedicated high performance network in which the Liberas can send the position data to the server PCs, a Hewlett Packard ProCurve 2900-24G switch is used, which connects data from its Gigabit Ethernet (GbE) ports to 10 Gigabit Ethernet (10GbE) Ports. These 10GbE ports are then connected to the 10GbE adapters of the server PCs.

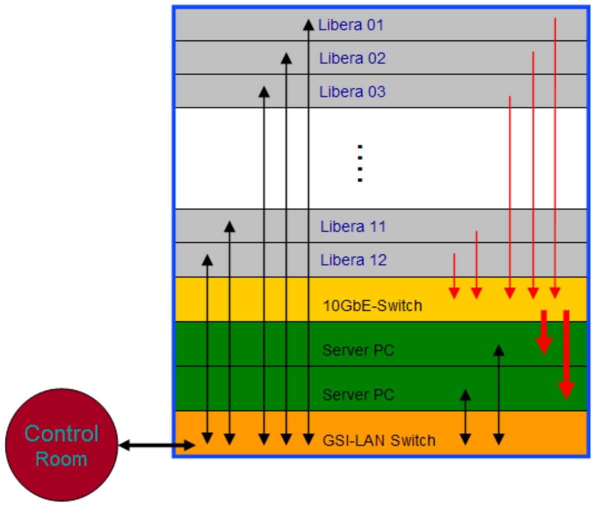

Figure 1: Schematic of BPM System with network connections. Red arrows: dedicated high performance network.

One server PC is a system with two 2.0 GHz Quad-Core Intel Xeon E5405 CPUs, 1333 MHz Front Side Bus and 32 GB DDR2 RAM with a Scientific Linux OS. The 10GbE adapter installed in the server PCs is an Intel 10GbE-MAC-Controller 82598EX for PCI-X.

The Liberas and the server PCs also have extra ports, to connect them to the GSI network to get access to them for controlling.

Position Measurement

The goal of the new digital BPM system is the determination of the beam position in a bunch-to-bunch mode. For the bunch synchronous measurement three separate steps are necessary: 1) baseline restoration, 2) window generation and 3) integration of the bunch signal. The algorithm for baseline restoration is presented in [4], we focus on the FPGA implementation for window generation and signal integration. The position of a bunch is determined by calculating the integrals of the difference and sum signal of two opposing pick-up plates (horizontal or vertical). E.g. the horizontal beam position is proportional to difference of the left and right plate signal

*Work supported by EU, project DIRACsecondary-Beams, contract number 515873

divided by their sum. In order to clearly separate successive bunches from each other and to integrate the signals only over the bunch area, an exact integration window must be generated (Fig 2).

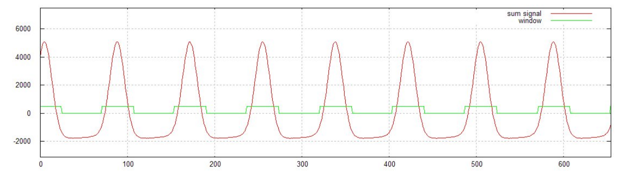

Figure 2: Sum signal of bunches in one plane (red) and generated integration windows (green).

The algorithm presented here generates the integration windows by inspecting the sum signals to determine flat regions between successive bunches [4]. To detect these flat regions even in noisy signals, the incoming sum values *x(z)* are continuously summed up, which leads to a saw-tooth like signal. Afterwards this signal is filtered with a median of the last five values that came in. After this filter process, the window is set active, if additionally eight successive values are strictly increasing.

In more detail, the algorithm works as follows: The summation can be realized inside the FPGA with a simple adder. For the median determination, a buffer of five values in sorted order is used (Fig. 3). Each incoming value *x(z)* is written to a shift register and also compared in parallel with all five values in this sorted buffer, to determine its new storing position. The value $x(z)\cdot z^{-5}$, that came five clock signals before the actual value *x(z)*, is delayed with the shift register. Additionally, this delayed value is compared in parallel with all values in the sorted buffer, to determine the value that stayed in the buffer since five clock cycles. This value has then to be removed from the sorted buffer.

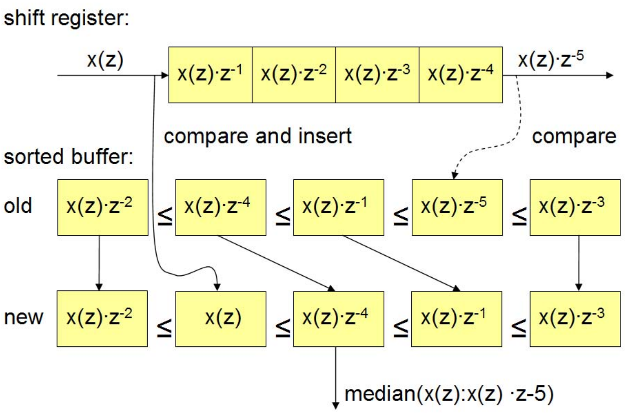

Figure 3: Example of sorted buffer after one clock cycle with $x(z)\cdot z^{-2} < x(z) < x(z)\cdot z^{-4}$.

In fact each cell of the sorted buffer is realized by a multiplexer with three inputs and a registered output (D flip-flop, dff, see Fig. 4). The inputs are connected to the outputs of the left and the right cell and to *x(z)* Exceptions are the cells at the border of the sorted buffer. They have just two inputs: One for *x(z)* and one for the output of their only neighbor cell.

With the result of the two simultaneous comparisons the multiplexers are switched, so that the values that lie between *x(z)* and $x(z)\cdot z^{-5}$ are moved by one cell into the direction of $x(z)\cdot z^{-5}$ with the next clock cycle. Simultaneously, *x(z)* is inserted into the correct place. The median of the five values is now just the content of the sorted buffers third cell.

This module only needs a latency time for the number of values from which the median has to be determined plus one clock cycle – in this case six. The width of the median filter is free scalable bounded by the free space of the FPGA.

These filtered values are put through a shift register for eight values. These values are then compared with each other, to look if they are strictly increasing. As long as this precondition is fulfilled, the integration window is active.

The integration of bunch data is then performed by summing up the incoming signals while the integration window is active. The position calculation itself is done inside a pipelined divider.

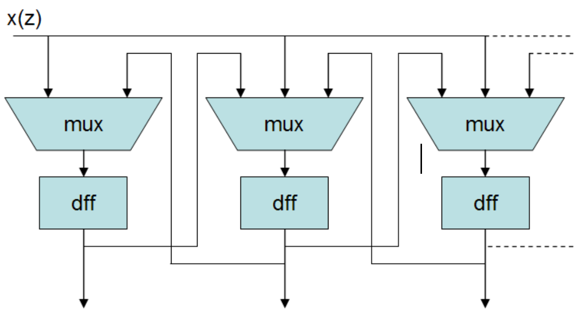

Figure 4: Realization of the left three cells of the sorted buffer.

Error Detection

For a consistent and synchronized data treatment, it is absolutely necessary that the Server PC stores the position data for bunches in correct order. Because of the high dynamics of the bunch frequency and very different bunch shapes, three error cases may occur from window generation: A bunch generates no window, one bunch generates multiple windows or one window is generated over multiple bunches. These errors must not necessarily occur for every BPM for the same bunch or bunches. If this happens, it will lead to desynchronization.

To make it possible for the Server PC to detect such errors, three timestamps are used for resynchronization. One timestamp is the absolute time of the RF period in which the integration window was found. The other two timestamps represent the beginning and the end of the integration window relative to the RF timestamp. With this information it is possible for a server PC to check whether one of the described errors occurred.

Furthermore two data fields are used (one for horizontal and one for vertical plane) to display detected errors in the analog signal, like e.g. clipping of the input signal or poor signal intensity. Altogether, this leads to a data record of 96 Bit for the position data of the two planes for one bunch.

Because of the usage of the UDP there is no control mechanism to get information if the server has correctly received a frame. If a package were lost, the position data on the server for an acceleration cycle would become inconsistent. A counter field is added to the UDP data frame, which is incremented by each sent frame, so that the server has the possibility to check for inconsistencies.

Data Transfer

At the end of an acceleration cycle with 5MHz bunch frequency, a data rate of 480 MBit/s of position data is generated per BPM. For transmission to the server PCs the User Data Protocol inside an Ethernet jumbo frame with a maximum transfer unit of 9000 Byte is used. From the MTU, 24 Byte are used for the IP Header and 8 Byte for the UDP Header. This means 747 beam position data records can be sent per frame. In addition to the MTU, 14 Bytes for the MAC Header and 4 Byte for the CRC checksum are needed. Overall this leads to a real data rate of about 483 MBit/s.

In Ethernet applications, a frame has to be sent in one sequence without breaks. That means that the data content of the frame has first to be buffered inside the FPGA before it can be transmitted. The Xilinx FPGA of the Libera provides several Dual Port RAM blocks, which can be used for this purpose. After the buffer is filled with the position data, its content has to be dumped to the FPGA's RocketIO for transmission. During this process, the content of this buffer must not be changed, but due to the running process of position measurement, data is still generated. For this reason a second buffer has been implemented, which can be filled while the other one is dumped (Fig.5).

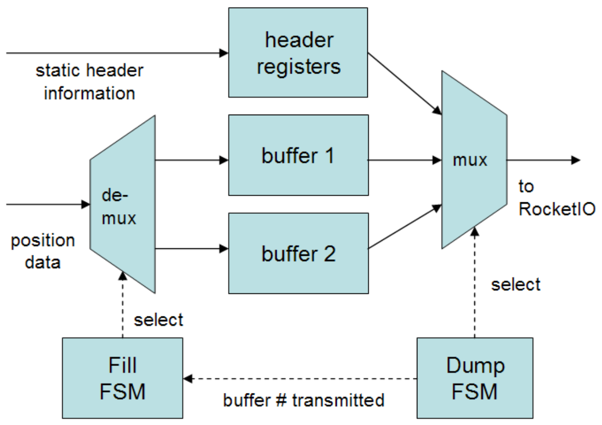

Figure 5: Schematic of output buffers.

A finite state machine (FSM) is used to observe the filling process and choose which buffer has to be filled.

The values of the header fields of the different OSI Layers (MAC, IP and UDP) have to be constant for each sent frame of one Libera, but it is necessary to configure each Libera with individual MAC and IP addresses and UDP Ports. To make the FPGA Design flexible enough, a register block is used, which contains the complete data of all OSI headers in correct order. The access to the necessary header-fields is then allowed over Liberas Single Board Computer (SBC).

The Dump FSM is triggered, when a buffer is full. It then starts an address counter and decides which data has to be transferred to the RocketIO. If the whole data of the buffer was transmitted, a signal is sent to the Fill FSM and the data switches to idle state. In this state, no data is transmitted over the RocketIO.

TEST RESULTS

Tests with the use of one server PC showed, that it is not capable of handling the whole amount of data, when all twelve Liberas send with highest data rate. At first Ethernet Type II Frames with an MTU of 1500 Bytes were used with an expected data rate of about 4800 MBit/s by sending with six Liberas, which resulted in a high amount of package loss on the 10GbE interface of the server PC (Table 1). The use of jumbo Ethernet frames with an MTU of 9000 Bytes didn't bring advancement. The monitor program of the ProCurve Switch showed that this loss does not occur on the way from the Liberas to the server PC, so that it must occur on the way from the 10GbE adapter to the processor.

Table 1: Results of Data Transfer Measurement from Liberas to Server PC with Different MTUs

MTU(Byte)	Packets/s	MBit/s
1500	~157600	~1890
9000	~26250	~1890

SUMMARY AND OUTLOOK

To solve the problems with the package loss, further tests are planned with a different 10GbE adapter for the server PC. If the necessary data rate for our whole system can't be reached, it is also possible to add a further server PC, to reduce the incoming data for each one.

More detailed analysis of the window generation algorithm showed, that it probably won't work properly for all beam parameters. For this case it is planned to implement and test a so-called Double Threshold Algorithm [5], which would cover the requirements for both, SIS18 and SIS100.

REFERENCES

[1] Company Instrumentation Technology, www.i-tech.si
[2] M. Arruat et al., "Front End Software Architecture", in Proc. of ICALEPCS07
[3] T. Hoffmann "FESA - The front-end software architecture at FAIR", these proceedings.
[4] A. Galatis et al., "Digital Techniques in BPM Measurements at GSI-SIS", in Proc. of EPAC '06
[5] U. Rauch et. al.,"Investigation on Base Band Tune Measurments using Direct Digitized BPM Signals", in Proc. of 5th CARE-HHH-ABI Workshop 2007