ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA115

A NEW TRANSVERSE AND LONGITUDINAL BUNCH BY BUNCH FEEDBACK PROCESSOR

M.G. Abbott, G. Rehm, Diamond Light Source, UK
I.S. Uzun, STFC, UK

Abstract

We describe the development of firmware to support Longitudinal Bunch by Bunch Feedback at Diamond Light source. As well as feedback, the system supports complex experiments and the capture of detailed electron beam diagnostics. In this paper we describe the firmware development and some details of the processing chain. We focus on some of the challenges of FPGA development from the perspective of a software engineer.

INTRODUCTION

At Diamond Light Source (DLS) we have been working on multi-bunch feedback for more than a decade [1–7], this work being originally based on developments at the ESRF [8]. Up to now our work has been based on the Libera bunch-by-bunch platform [9], which is now obsolete and has limited capacity for further developments — at the time of writing the Virtex-II Pro FPGA at the heart of the Libera processor is 15 years old. With this platform we have focused on stabilising and measuring only transverse instabilities.

More recently we have been asked to provide support for measurement and stabilisation of longitudinal multi-bunch instabilities as part of an ongoing project to install normal conducting RF cavities [10]. It is anticipated that these may introduce longitudinal resonances and instabilities which will need to be managed.

We have therefore been working on a project to upgrade our TMBF (Transverse Multi-Bunch Feedback) system to run on more modern hardware and to add longitudinal capabilities to the system, so creating an LMBF (Longitudinal Multi-Bunch Feedback) processor. We have already reported [11] on our preliminary work on the new system and on the choice of hardware; we'll discuss this further below.

In this paper we will describe the architecture of the new LMBF (Longitudinal Multi-Bunch Feedback) processor based on our chosen hardware, and discuss some of the lessons learned during this development. From a software engineering perspective, development of a complex FPGA system presents some remarkable challenges, which we'll also discuss.

HARDWARE PLATFORM

The development of our new LMBF processor was driven by two motivations: to ensure that we are ready for any new longitudinal instabilities, and to increase the capabilities of our existing system [10, 11]. We also wanted to improve our knowledge and understanding of high speed processing hardware relevant to synchrotron diagnostics.

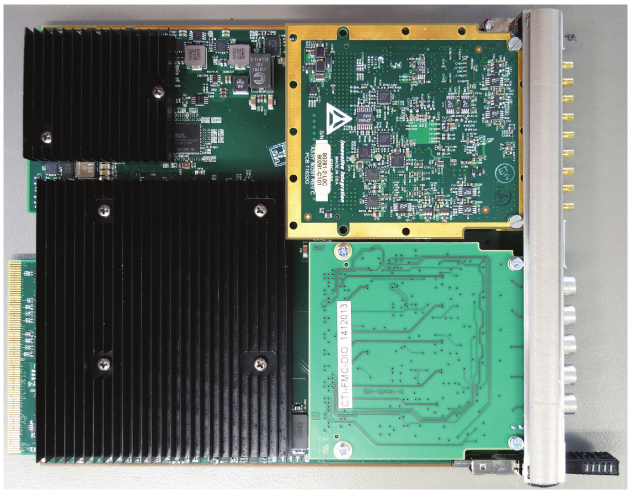
Figure 1: Photo of assembly of FMC-500 and Digital IO FMC on the AMC525 carrier, with FMC-500 at top right.

We started with an investigation to determine the appropriate hardware platform for this kind of development. Early on it was decided that we would need a powerful FPGA with FMC (FPGA Mezzanine Card) support. Initially we looked at self contained FPGA platforms, and even briefly considered creating our own, but in the end we converged on MicroTCA [12]. This platform provides us with a wide choice of crates and AMC (ATCA Mezzanine Card) processing modules with high speed interconnect. The same platform is being used for a joint development with ALBA of digital low level RF [13].

Having selected MicroTCA as our platform we then selected the following hardware. Figure 1 shows the digital processing hardware assembled ready for insertion in the MicroTCA crate.

FMC-500M (HPC) High Pin Count FMC providing dual channel 500 MS/s 14-bit ADC and dual channel 1230 MS/s 16-bit DAC [14]. This will support bunch-by-bunch operation at our machine RF frequency of 500 MHz, and can be driven by our machine clock.

FmcDIO5chTTLa Five port digital IO FMC [15]. This is used for miscellaneous triggering and other signals.

AMC525 Double width AMC card with two HPC FMC slots, 2 GB of fast on board DRAM and 128 MB of slower DRAM connected to a Virtex-7 690 FPGA, supporting an 8 lane gen3 PCIe connection over the MicroTCA backplane [16]. This is where all the FPGA firmware will run, and the fast backplane connection will allow us to do a lot of data processing in the associated CPU.

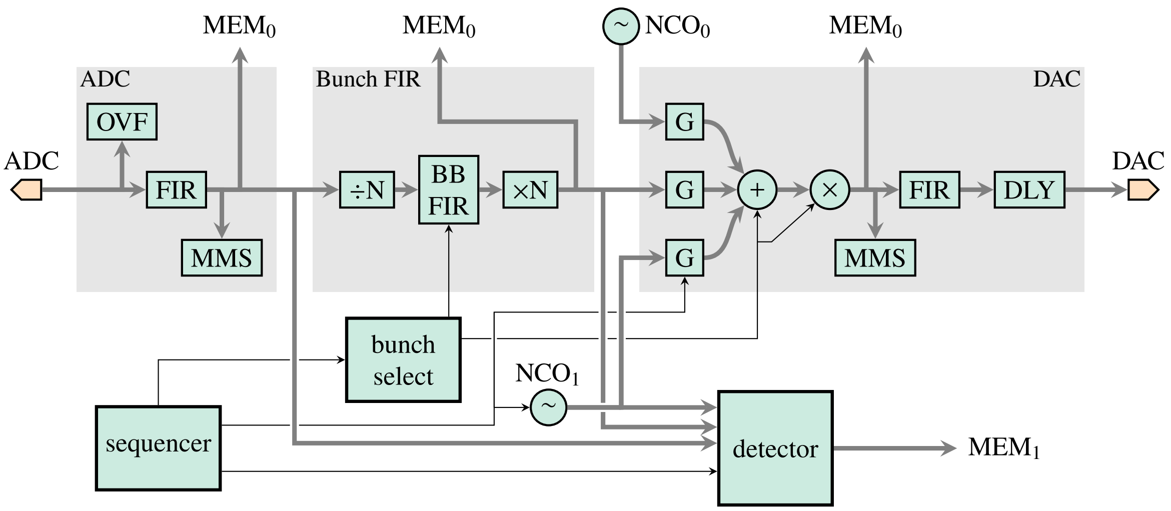

Figure 2: Overview of LMBF signal processing chain. Key: OVF ADC input overflow detection; FIR I/O compensation filter; MMS bunch position and motion measurement; ÷N bunch by bunch decimation; BB FIR bunch by bunch filter; ×N bunch by bunch interpolation; G gain control; DLY output alignment delay; ~ controllable oscillator (NCO).

AMC720 This is an AMC card with an Intel Xeon processor with ample memory and sufficient local storage. We install Red Hat Enterprise Linux 7 on this, and run our control system and any associated signal processing.

VT814 This is a 2U MicroTCA chassis with 6 AMC slots and dual redundant power supplies.

FUNCTIONALITY AND DESIGN

Functionality

The bunch-by-bunch processor has two functions: to stabilise coupled bunch oscillations of the electron beam, and to provide diagnostics functions for measuring beam behaviour and detailed parameters.

Bunch oscillation is stabilised by a feedback filter applied to each bunch to generate feedback at around 180° out of phase.

For diagnostics our processor provides the following features:

- Capture of up to 2 GB of two channels of 500 MS/s data, selectable from different processing stages. This allows the capture of bunch by bunch behaviour and feedback channels.
- Multiple Numerically Controlled Oscillators (NCOs) for driving excitation onto the beam. This is used for tune measurement and grow damp experiments.
- Detectors for capturing IQ data by mixing an NCO with the beam response. Outputs from the detectors are written to slow memory and can be used for tune measurement and other experiments.
- Programmable sequencer for programming a sequence of NCO frequencies and controlling the behaviour of the detectors.
- Individual bunch controls. For each bunch a different excitation and different feedback filter can be selected. This allows us to perform tune measurement on a chosen set of bunches, and to have a different feedback filter for a hybrid bunch.

Diagnostics experiments are performed by programming the bunch controls, the available filters and the sequencer, and then the sequencer is triggered. Measurements are then written to slow memory and then read out and processed by the control system.

System Design

Figure 2 shows the signal processing chain as implemented in the current design. This is similar to the existing TMBF as reported earlier [1]. The main enhancements so far (work is still in progress) are the improved bunch by bunch motion measurement, longer FIR filters throughout, and rate change to support longitudinal processing.

There are two memory capture targets, MEM_0 used for dual channel full data rate capture, configured as a 2 GB circular buffer, and MEM_1 used for detector capture, configured as two separate 64 MB blocks.

As both the ADC and DAC on the FMC-500 are dual channel, one FPGA image supports two channels of operation. In transverse mode of operation the two channels operate independently on X and Y.

Bunch-by-Bunch Min/Max/Sum

In the earlier design the motion of each bunch is captured by capturing the minimum and maximum position of each bunch over a 200 ms interval and displaying the difference as the beam motion. In the current design this data has been extended by accumulating both sum and sum of squares for each bunch over the sampling interval.

After processing in the control system, this now allows us to display the mean bunch position and the standard deviation of beam motion. This provides a more sensitive view of bunch behaviour.

Longitudinal Processing Features

There is not too much difference in the processing required for longitudinal vs transverse bunch by bunch feedback — the biggest differences is in the output drive chain after the DAC, as described in [11]. However, there are three signal processing differences that are important to our system.

Operating on beam phase. The input signal to LMBF comes from the sum signal from a set of electron beam position monitor (EBPM) buttons, processed by a front end to generate two signals with 90° phase separation, processed as channels I and Q of input. The front end phase will be controlled to drive the overall signal level of Q as close to zero as possible.

After this preparation, the signal I should measure the fill pattern, and Q will measure the phase relative to RF of each bunch (scaled by signal level, but that's a feature common to all bunch-by-bunch processors). Longitudinal feedback is performed on the Q signal alone.

Generating single side-band IQ. In longitudinal mode the output signal will be used to drive a mixer. To reduce out of band power demands we will operate with an IQ mixer and will generate a single side-band feedback signal.

This is done by duplicating the Q input signal onto both processing chains immediately before bunch-by-bunch FIR processing, and programming the FIRs in the two channels to generate phase shifts of the synchrotron tune separated by 90°. The resulting two DAC outputs will drive the IQ mixer.

Bunch-by-bunch frequency shift. The synchrotron tune (longitudinal oscillation frequency) is a low fraction of the machine revolution frequency, in our case typically around 0.004. This will require a substantial frequency shift, which is done by decimation and interpolation.

Each bunch is averaged for a programmable number of turns before being fed to the bunch-by-bunch FIR. The output from the filter is then held stationary for the same number of turns to generate the up sampled output.

Longitudinal vs Transverse Processing

It is helpful to be able to use the same firmware and software for both longitudinal and transverse processing. Switching between these two modes is done by configuring the decimation and configuring inter-channel operation.

The bunch-by-bunch FIR decimation and interpolation stage is disabled by setting the decimation count to 1.

There are three inter-channel cross-bars, configured for straight-through operation in transverse mode, and with the following special settings for longitudinal operation:

- The second ADC input channel Q is duplicated onto both processing channels immediately before FIR processing. This allows capture of ADC for I, but supports operation on the beam phase.
- Only one sequencer is active and is configured to control both channels simultaneously.

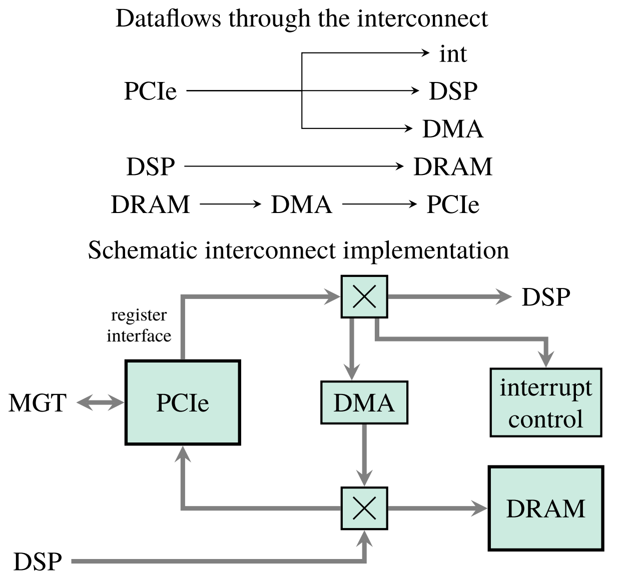

Figure 3: Interconnect between PCI Express core (PCIe), memory (DRAM), and the LMBF data processor (DSP). ⊠ indicates an AXI crossbar, arrows point from master to slave. Data flows are shown at the top of the figure.

- The NCO outputs are configured to drive the same signal with 90° phase shift into the two outputs, and again only once set of NCOs is active.

IMPLEMENTATION

Interconnect

After the "Hello World" step of getting flashing LEDs, the very first development was of what we've called the "Interconnect". This part of the system glues together all of the core resources provided by the FPGA carrier card to provide a basic environment for the development of the rest of the system.

Figure 3 illustrates the basic structure of the interconnect. The register interface is used by the control system to manage the rest of the processing chain (referred to as DSP here), a simple interrupt controller, and a DMA engine for transferring data from DRAM to processor memory.

This part of the FPGA design was constructed using the Vivado Block Design editor, which is a graphical tool for configuring and linking Xilinx components into a complete design. The AXI bus is used to connect all of the components. The rest of the system is written in VHDL.

Kernel Driver

A dedicated kernel driver manages the interface from the processor to this part of the core FPGA system. The DSP registers are available to be mapped into memory, whereas the DMA and interrupt control registers are managed by the driver only. This allows for separation between the constantly evolving system interface, running in user space, and the fixed interrupt and memory management. Also, it is helpful to separate out the memory and interrupt management, as this part of the system is able to crash or otherwise compromise the behaviour of the control processor.

The kernel driver presents three device nodes to userspace, one for mapping registers into memory and receiving interrupt events, and two for memory readout from the two banks of DRAM provided by the AMC525. The design and implementation of this driver has remained unchanged since the start of the project.

Top Level System Design

The FPGA design is structured into the interconnect described above (Fig. 3), interfaces to the two FMC cards, two channels of feedback processing (DSP), and a shared register control interface. Four banks of registers provide control over:

- System level registers for clock management and FMC-500 control and setup.
- Control registers for resources shared between the two DSP channels including triggering and fast capture to memory.
- Two banks of registers, one dedicated to each DSP channel.

A custom software tool is used to manage the list of register assignments. From a central list of registers and fields, VHDL definitions, C `struct` definitions, and documentation are automatically generated. This makes it straightforward to add, remove, or rearrange registers, which is otherwise laborious and error prone.

FMC-500 Configuration

An initial challenge to getting the FMC-500 up and running was that the documentation for the card [14] was lacking in important detail; we understand that the normal application for this component is as part of a vendor supplied system. However, the manufacturer was very helpful in answering our questions and providing the missing pieces.

The FMC-500 is built from three key devices, a PLL clock controller (LMK04828), a dual channel ADC (AD9684), and a dual channel DAC (AD9122), each of which is controlled by a dedicated SPI interface. The ADC and DAC are straightforward to configure (the DAC is somewhat more complex), but the PLL controller is very complex.

A Python script captures the complex register interface to the LMK04828 and is run on the processor during system startup.

Signal Processing Chain

Most of the LMBF signal processing chain was taken from our last TMBF design [1], but nevertheless there has been a lot of rework. The most important changes to the original design include the following:

- Beam motion measurement now also measures mean beam position and standard deviation of motion.
- Bunch by bunch down and up sampling is now part of the bunch-by-bunch FIR. This is needed for LMBF to support the very low synchrotron tune number.
- There are now multiple tune measurement detectors with individual bunch enables on each detector.
- Cross-bars are provided at key points in the signal processing chain to support IQ processing for LMBF feedback and excitation.

Pipelines and Delay Compensation

For any complex FPGA design it is necessary to pay a lot of attention to pipelines in the data flow. Inevitably these introduce delays, and unless compensated for, there can be data skews between different parts of the system. In our firmware we have handled these delays and skews by a process of automated measurement and software compensation.

The skews of interest are those that affect which bunch appears as “bunch zero” in the various readouts and controls, such as the various min/max/sum waveforms, capture to DRAM, bunch-by-bunch configuration control, and detector configuration.

We have created a Python script which works through all the relevant configuration options and automatically measures all the skews. At the time of writing the delays cover a range of over 100 RF clock ticks! The EPICS driver then uses these measurements to compensate all the waveforms it delivers to users.

EPICS Driver

The control system runs on the dedicated processor card and provides an EPICS interface to all aspects of the system. User interface screens are built using EDM.

DEVELOPMENT CHALLENGES

FPGA Culture Shock

Coming to FPGA development as a software engineer is a real culture shock. The huge differences in semantic model are interesting, but really this is the smallest issue. The main challenges are two fold.

Firstly, the state of the art in Hardware Definition Languages (HDLs) is very weak. The mainstream languages (VHDL, Verilog, SystemVerilog) all suffer from very poor abstraction, mismatch between language and target, and excessive verbosity (VHDL especially). There does seem to be some work on alternatives, but unfortunately we come to the second challenge.

All FPGA development is strongly and inextricably tied to vendor tools. The general philosophy is that *everything* is licensed. All components are referred to as “IP” (Intellectual Property), and encryption is widespread. The whole environment is inimical to open source development and sharing.

In the Accelerator and Large Experimental Physics community we rely on intellectual sharing for reuse and shared development. From a management perspective, sharing is important to help manage the risks associated with the mismatch between the relatively brief lifetime of modern technology, and the extended lifetime of large facilities.

Development Cycle

Three challenges stand out in the development cycle: source control, build times, and debugging.

For source control it is important to separate built from edited files, and it is desirable to store source files in a format which is friendly to diff. Although this can be done with the Xilinx Vivado development tool, this process goes against the grain of the tool. Scripted development is supported, but the process is surprisingly fragile.

The build process is extraordinarily slow, particularly in comparison with compiling a software language. The process is certainly more complex, and in particular the placement and routing of resources on the FPGA is difficult.

Debugging FPGA code is more laborious than debugging software, in particular it is much more difficult to inspect what is going on inside the device, and the slow build process makes debugging by instrumentation not very practical. Instead, debugging is mostly done through simulation, and there are some very sophisticated and complex simulation tools available to complement the development process.

Running at 500 MHz

In the previous TMBF design we operated with 4 lanes of digital processing with an FPGA clock of 125 MHz. Implementing this makes for some clumsy complications in the firmware, so when we discovered that the our new FPGA supports 500 MHz operation it was very tempting to simplify the code by reworking the design for single lane operation. This might have been unwise.

According to Xilinx documentation, any 7-series FPGA of speed grade 2 supports 500 MHz operation of the key components, block memory, and DSP units. However, there are two main challenges.

The first obstacle was rather unexpected: when allowing Vivado to infer block memory assignment (the normal way of using this resource), it turns out that the tool automatically configures the memory in a slower mode of operation which is not compatible with full speed operation. This was fixed with an explicit mode overwrite in the FPGA constraints.

The second obstacle is that achieving timing closure is very hard, and is very brittle against any changes in the code. It seems that the component placement state of the development tool is not well optimised for 500 MHz placement, and it needs a lot of manual guidance. So far this has been done using "pblocks" which are used to restrict where certain parts of the design are placed on the device.

CONCLUSIONS

Development of the new TMBF and LMBF system has been very instructive, and will provide DLS and other projects with enhanced bunch-by-bunch feedback and diagnostics capabilities. We are looking to share this development, and are actively exploring collaborations with a number of laboratories.

Work is in progess in progress to sort out licensing issues so that we can share our work freely with other facilities.

We will be extending the diagnostic capabilities of the system described here with further filtering, with more excitation and detection options, and with other improvements to be determined.

REFERENCES

[1] M.G. Abbott, G. Rehm, and I.S. Uzun, "Architecture of transverse multi-bunch feedback processor at Diamond", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 298–301, doi:10.18429/JACoW-ICALEPCS2015-MOPGF097

[2] A.F.D. Morgan, G. Rehm, and I. Uzun, "First tests of the transverse multibunch feedback at Diamond", in *Proc. DIPAC'07*, Venice, Italy, May 2017, pp. 295–297.

[3] A.F.D. Morgan, G. Rehm, and I. Uzun, "Performance and features of the Diamond TMBF system", in *Proc. EPAC'08*, Genoa, Italy. Jun. 2008, pp. 3281–3283.

[4] G. Rehm, M.G. Abbott, A.F.D. Morgan, J. Rowland, and I. Uzun, "Measurement of lattice parameters without visible disturbance to user beam at Diamond Light Source", in *Proc. BIW'10*, Santa Fe, NM, USA, pp. 44–48.

[5] I. Uzun, M.G. Abbott, M.T. Heron, A.F.D. Morgan, and G. Rehm, "Operational status of the transverse multibunch feedback system at Diamond", in *Proc. ICALEPCS'11*, Grenoble, France, Oct. 2011, pp. 219–222.

[6] M.G. Abbott, G. Rehm, and I.S. Uzun, "Capability upgrade of the Diamond transverse multibunch feedback", in *Proc. IBIC'13*, Oxford, UK, Sep. 2013, pp. 682–685.

[7] G. Rehm, M.G. Abbott, and A.F.D Morgan, "New features and measurements using the upgraded transverse multibunch feedback at Diamond", in *IBIC'14*, Monterey, CA, USA, pp. 696–699.

[8] E. Plouviez, P. Arnoux, F. Epaud, J. Jacob, J.M. Koch, N. Michel, G.A. Naylor, J.-L. Revol, V. Serriere, and D. Vial, "Broadband bunch by bunch feedback for the ESRF using a single high resolution and fast sampling FPGA DSP", in *Proc. EPAC'06*, Edinburgh, UK, Jun. 2006, pp. 2976–2978.

[9] Instrumentation Technologies, "Libera Bunch-by-Bunch", http://www.i-tech.si

[10] C. Christou *et al.*, "Progress with the Diamond Light Source RF upgrade", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017, pp. 4358–4361, doi:10.18429/JACoW-IPAC2017-THPIK112

[11] G. Rehm, M.G. Abbott, and A.F.D. Morgan, "Measurements of longitudinal coupled bunch instabilities and status of new feedback system", in *Proc. IBIC'16*, Barcelona, Spain, Sep. 2016, pp. 298–301, doi:10.18429/JACoW-IBIC2016-TUCL03

[12] MicroTCA Overview, PICMG, https://www.picmg.org/openstandards/microtca/.

[13] P. Gu *et al.*, "Digital low level RF systems for Diamond Light Source", in *Proc. IPAC'17*, Copenhagen, Denmark, May 2017, pp. 4089–4091, doi:10.18429/JACoW-IPAC2017-THPAB152

[14] Innovative Integration, "FMC-500, FMC Module with 2x 500 MSPS 14-bit A/D, 2x 1230 MSPS 16-bit DACs with PLL and Timing Controls", https://www.innovative-dsp.com/products.php?product=FMC-500

[15] CERN Open Hardware Repository, “FMC DIO 5ch TTL a”, `https://www.ohwr.org/projects/fmc-dio-5chttla`

[16] Vadatech, “AMC525, dual FMC carrier, 2 FMC, 690T FPGA, FFG-1761 package, FPGA Mezzanine Card”, `http://vadatech.com/product.php?product=393`