

CONTINUOUS BUNCH-BY-BUNCH 16-BIT DATA ACQUISITION USING DDR2 SDRAM CONNECTED TO AN FPGA*

J. Weber, M. Chin, LBNL, Berkeley, CA, U.S.A.

Abstract

A hardware system that acquires and stores a large buffer of bunch-by-bunch 16-bit data has been realized. A high resolution (up to 16-bit) analog-to-digital converter (ADC), or bank of ADCs, samples the analog signal at the bunch frequency. The digitized data is fed into a Field Programmable Gate Array (FPGA), which contains an interface to a bank of Double Data Rate SDRAM (DDR) type memory. With appropriate data bus widths, the FPGA bursts the ADC data into the DDR fast enough to keep up with the bunch-by-bunch ADC data continuously. The realized system demonstrates continuous data transfer at a rate of 1GByte/sec, or 16-bit data at 500MHz, into a 64MByte DDR. This paper discusses the implementation of this system and the future of this architecture for bunch-by-bunch diagnostics.

INTRODUCTION

For many years, beam motion in particle accelerators has been understood well enough to design bunch-by-bunch feedback systems capable of stabilizing the beam. Until recently, these systems lacked the diagnostic capability to monitor the RMS motion of each bunch and store it continuously for an extended period. This information can be used to compute the various coupled bunch modes of oscillation. Monitoring the strength of each mode over time can provide valuable insight that could ultimately be used to better stabilize the beam. High bit resolution is desirable to increase the accuracy of the modal analysis.

One method of capturing long records of high resolution bunch-by-bunch data is digitizing the beam signal with one or several high resolution ADCs, sending the data to a high performance FPGA to pack into a buffer in a large DDR or bank of DDRs (see Figure 1). Only recently has the combination of ADC/FPGA/DDR technologies achieved the performance required to make such a measurement on an accelerator beam with a bunch rate of 500MHz or greater, such as at the Advanced Light Source (ALS).

A new transverse feedback system (TFB) is under development at the ALS that includes hardware capable of performing large bunch-by-bunch captures [1]. We have developed firmware for the FPGA that demonstrates the ability to capture up to 16-bit data at 500MHz and store it in a 64MB DDR, giving us the ability to capture over 15ms of continuous bunch-by-bunch data. This paper describes the hardware requirements and firmware options to achieve the measurement, as well as an example targeted for the ALS TFB and the NSLS-II RF

BPM, and the future of this technology for bunch-by-bunch measurements.

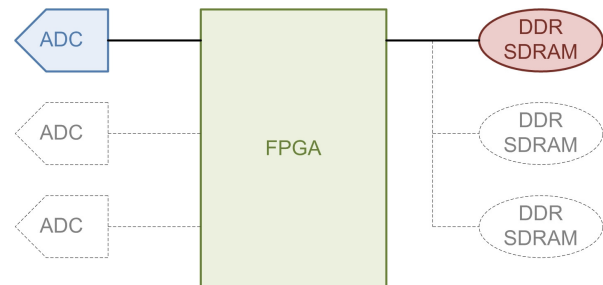


Figure 1: ADC-FPGA-DDR model for bunch-by-bunch data acquisition.

HARDWARE REQUIREMENTS

To capture bunch-by-bunch data, the effective sampling rate of the ADC circuit must match the maximum bunch rate of the accelerator. This can be accomplished by using a single ADC if the sampling rate and bit resolution are sufficient, or by time-interleaving samples from multiple ADCs. Ideally, a single ADC is used to eliminate sources of error caused by multi-ADC sampling, such as phase and amplitude mismatch and thermal drift. The ALS TFB system requires 12-bit resolution and 500Mps sampling, but at the time no single ADC was available that met those requirements. Instead, two 12-bit ADCs are used, each sampling at 250Mps and clocked 180 degrees out of phase to achieve an effective sampling rate of 500Mps.

The FPGA must be able to clock the related logic fast enough to process all the ADC data continuously. In a single ADC system, the FPGA must be clocked at the full bunch rate, unless the ADC has multiple parallel output data buses. In a multi-output ADC or multi-ADC system, the FPGA clock rate can be reduced since the data can be processed in parallel, although this uses additional logic resources. In practice, it is recommended to clock the FPGA well below its specified maximum clock rate since the amount of logic used in the FPGA is large enough that routing at such high frequencies becomes problematic or impossible. The ALS TFB uses a Xilinx Virtex-5 LX50 speed grade -1 FPGA clocked at 250MHz to match the ADC sampling rate, well below the FPGA maximum output clock frequency of 450MHz [2].

The FPGA provides the clock and control interface to the DDR chips, which all must be clocked synchronously and fast enough to transfer the ADC data continuously. At the basic level, the DDR theoretical bandwidth can be calculated by multiplying the maximum allowable DDR clock rate by 2, then by the data bus width. For the ALS TFB, which uses a DDR2 with maximum clock rate of 200MHz [3], the theoretical bandwidth is $200\text{MHz} \times 2 \times$

* This work was supported by the Director, Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

4 Bytes = 1.6 GByte/s. The ADC data is 12-bits wide, but since it is simpler to transfer data to the DDR without crossing byte boundaries, the ADC values are packaged as 2 Bytes, so the required DDR transfer rate is $250\text{Mps} \times 2 \times 2 \text{ Bytes} = 1.0 \text{ GByte/s}$. Although the theoretical bandwidth appears to be sufficient to meet the requirement, the firmware implementation impacts the maximum achievable continuous transfer rate. This will be explored in greater detail in the following sections.

FIRMWARE

There are many ways to implement firmware that transfers data from the ADC to DDR. On one extreme, an entirely custom logic solution can be developed, which provides the most flexibility to optimize the design for the specific application. On the other extreme, it is also possible to use design examples and configurable cores from the FPGA manufacturer with minimal customizations to field a less flexible solution more quickly.

For the ALS TFB system, the Xilinx Multi-Port Memory Controller core (MPMC) was selected as the DDR interface. For the Avnet Xilinx Virtex-5 LX50 evaluation board used in the ALS TFB system, the default design includes the MPMC. The MPMC can be configured to use up to 8 separate ports, which is important if the DDR is connected to multiple interfaces. All ports are arbitrated internally in the MPMC using one of the standard arbitration schemes, or a user-defined scheme.

The MPMC supports several types of interface ports. The most basic and high performance of these is the Native Port Interface (NPI), which was chosen to stream the ADC data to the DDR in the ALS TFB design. In addition, the MPMC has a PLB port that the Microblaze embedded processor uses to access OS and program code stored in the DDR. Figure 2 shows the MPMC connections in the ALS TFB system.

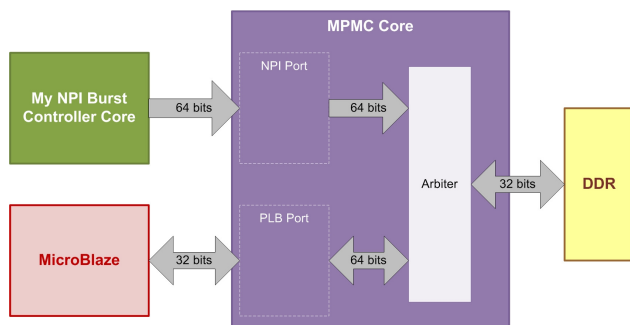


Figure 2: Multi-Port Memory Controller connections in the ALS TFB system.

Native Port Interface (NPI)

My NPI Burst Controller Core (My NPI core) is a custom logic core that transfers the ADC data to the MPMC (Figure 3). One 12-bit value from each ADC is packed into a 32-bit word and clocked into the burst FIFO

in My NPI core continuously at 250MHz. Once the burst FIFO contains enough data for a 64-word burst, the control logic in My NPI core enables a 64-word transfer to the write FIFO in the MPMC NPI port at the DDR clock rate of 200 MHz. When the transfer is complete, My NPI core control logic requests a burst write from the NPI port to DDR.

The burst FIFO provides a mechanism for the ADC data to cross clock domains from the ALS RF clock domain ($f_{RF}/2 = 250\text{MHz}$) to the DDR clock domain ($LO\ 100\text{MHz} \times 2 = 200\text{MHz}$) as well as some amount of buffering to allow continuous ADC data acquisition in case the NPI burst request is held off by the MPMC arbiter. The burst FIFO can be as deep as the resources available in the FPGA allow. The NPI port write FIFO is required for NPI burst transfers to DDR.

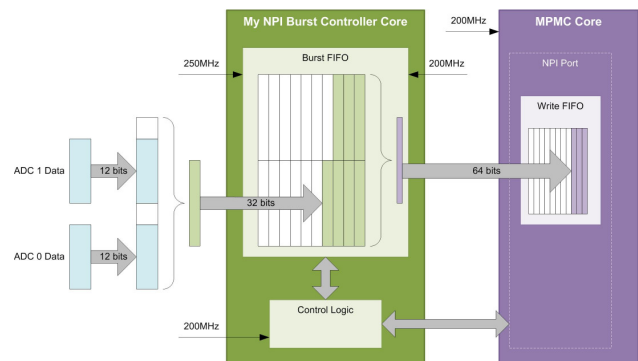


Figure 3: My NPI Burst Controller core architecture.

The maximum continuous transfer rate via an NPI port in burst mode depends on the DDR clock rate, the DDR data width, and the NPI port data width. For the ALS TFB using the maximum DDR clock rate (200MHz), DDR data width fixed in hardware (32 bits), maximum NPI data width (64 bits), and maximum NPI burst size (64 words), the maximum continuous transfer rate is 1.138 GByte/s [4] or 225ns per burst.

MPMC Arbitration

For a single port MPMC system, the maximum NPI burst transfer rate would be sufficient to meet the 1 GByte/s rate required to transfer the ADC data. However, in a multi-port system, the MPMC arbitration scheme must balance the performance needs of all ports to avoid starving any one port and potentially losing critical data. There are several standard arbitration schemes and a configurable custom scheme available in the MPMC. For the ALS TFB system, we base-lined the performance of two schemes: round-robin and custom with the NPI port configured to always have top priority. For each case, the Microblaze ran a simple standalone program out of FPGA Block RAM. The code requests single reads of DDR data in a tight loop to approximate a program executing out of DDR. The code for this program is shown below:

```
#include "xio.h"

int main (void)
```

```

{
  Xuint32 data;
  Xuint32 addr;

  addr = 0x8F000000;
  while (1) {
    data = XIo_In32(addr);
  }
}
    
```

The default MPMC arbitration scheme is round-robin. In this scheme, each port is sequentially given top priority (i.e. port 1, 2, 3, 1, 2, 3, etc.). For the ALS TFB with a 2 port MPMC, this results in the top priority alternating between the 2 ports (port 1, 2, 1, 2, etc.). Simulations using this scheme indicate that the NPI burst interface does not get enough service to sustain continuous transfer of the ADC data because alternating transactions limited the continuous NPI burst throughput to 256 Bytes per 285 ns (Figure 4A), or 0.90 GByte/s.

Next, the MPMC arbitration scheme was modified so that the NPI port always has priority. This means that the NPI port is serviced as soon as possible after a transaction request, and the PLB port is only serviced if there is no request pending from the NPI port. This scenario was also simulated, and a typical pattern of transactions for this case is shown in Figure 4B. PLB port transactions only occur about every 2 NPI bursts so that the NPI interface can transfer $256 * 2 = 512$ Bytes every 510 ns, or just over 1 GByte/s.

For a complex system with nondeterministic Ethernet software, the software must, at a minimum, handle the expected payload required for system operation. In most systems, this payload is well known and worst-case scenarios can be simulated and tested on an isolated network. To make the system more robust against traffic on a shared network, the software can be optimized to ignore irrelevant traffic and minimize the burden on the DDR. This challenge is not unique to Ethernet interfaces; any nondeterministic external interface that requires software access to the DDR will cause the same behavior.

FUTURE CONSIDERATIONS

The MPMC core is a flexible and capable solution for FPGA embedded processor systems demanding multi-tasking capability from their DDR SDRAM banks. The ALS TFB design is an example of how the MPMC can be configured to efficiently support two interfaces with very different requirements. The MPMC arbitration scheme used is one of simplest; in the custom scheme, time slot prioritization can be used to precisely tune the arbitration to the specific needs of the application and maximize the DDR bandwidth efficiency. Also, the depth of the burst FIFO can be adjusted to compensate for longer periods of port starvation due to the configuration of the arbitration scheme. For more complex designs requiring many MPMC ports, the arbitration scheme is critical to meet the

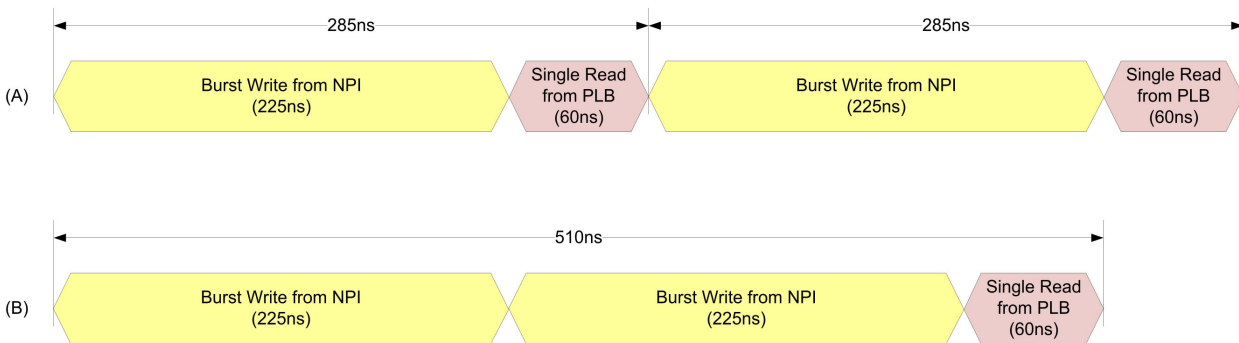


Figure 4: Typical DDR transaction timing diagrams for different MPMC arbitration schemes. (A) Round-robin arbitration. (B) Custom arbitration with the NPI port always the highest priority.

The design was built and tested on the ALS TFB hardware and ran continuously for several hours without corrupting the ADC data capture buffer. This was verified both by monitoring a latched version of the burst FIFO full signal and by reading out the data in the buffer after a stop capture trigger. A slightly modified version of My NPI core (64-bit input port) was built and successfully tested on the Xilinx ML507 evaluation board for potential use in the NSLS-II RF BPM system [5].

This project is now ready to be integrated with Ethernet software that provides remote access to the system. On a shared network, the Ethernet software must handle variable rates of traffic coming from the network, the software load is usually nondeterministic, and the behavior can be difficult to simulate or even test in hardware.

demands of all your port interfaces.

Many of the Xilinx FPGA evaluation boards now use a DIMM interface to DDR SDRAM to support much larger depths of memory, allowing easy memory upgrade by replacing only the DIMM module, not the FPGA board. FPGA and DDR clock rates continue to climb and DDR transfer bandwidths increase proportionally, limited only by the maximum burst size and the data bus bandwidth. For example, the Xilinx ML605 Combines a Virtex-6 FPGA with a 512MB DDR3 SO-DIMM 64 bits wide that can be clocked at 533MHz, which could support a theoretical transfer rate of 8.5 GByte/s, more than 5 times greater than the ALS TFB system.

CONCLUSION

Continuous 16-bit bunch-by-bunch data acquisition at 500 MHz has been realized on the ALS TFB system. The MPMC core provides a fairly flexible interface to the DDR that includes multi-port arbitration. The NPI port provides the highest performance burst transaction of the port types supported by the MPMC. The performance requirements for each port must be carefully considered and addressed by the arbitration scheme. The future of the technologies in use will support ever increasing demands on resolution, bandwidth, and buffer length for bunch-by-bunch data capture.

ACKNOWLEDGEMENTS

The authors would like to thank David Robin, Greg Portmann, and the rest of the ALS Physics Group for supporting and motivating this project. Also, thanks to Kurt Vetter and Kiman Ha for informal discussions and emails contributing to this effort and for testing the firmware on the ML507 evaluation platform.

REFERENCES

- [1] J. Weber, M. Chin, “ALS FPGA-Based Transverse Feedback Electronics”, BIW08, Tahoe City, CA, May, 2008.
<<http://accelconf.web.cern.ch/AccelConf/BIW2008/papers/tuptpf017.pdf>>.
- [2] Xilinx. “Virtex-5 FPGA Data Sheet: DC and Switching Characteristics”, DS202 (v5.2), June 25, 2009,
<http://www.xilinx.com/support/documentation/data_sheets/ds202.pdf>.
- [3] Micron. “256Mb: x4, x8, x16 DDR2 SDRAM Features”, 256MbDDR2.pdf – Rev. M 7/09 EN, July, 2009.
<<http://download.micron.com/pdf/datasheets/dram/ddr2/256MbDDR2.pdf>>.
- [4] Xilinx. “Multi-Port Memory Controller (MPMC) (v5.04.a)”, DS643, December 2, 2009,
<http://www.xilinx.com/support/documentation/ip_documentation/mpmc.pdf>.
- [5] K. Vetter, et al, “NSLS-II RF Beam Position Monitor”, these proceedings.