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

J. Weber, M. Chin
Lawrence Berkeley National Lab, Berkeley, CA 94720, USA

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

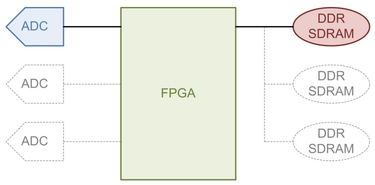

Figure 1. ADC-FPGA-DDR model for bunch-by-bunch data acquisition.

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.

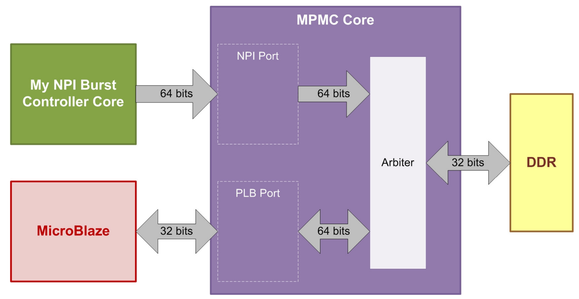

Figure 2. Multi-Port Memory Controller connections in the ALS TFB system.

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.

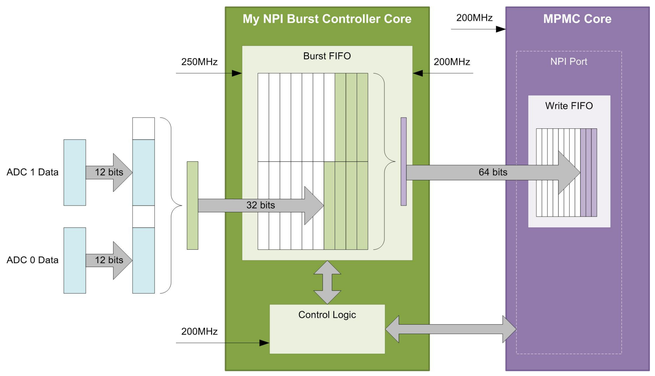

Figure 3. My NPI Burst Controller core architecture.

The burst FIFO provides a mechanism for the ADC data to cross clock domains from the ALS RF clock domain (fRF/2 = 250MHz) to the DDR clock domain (LO 100MHz × 2 = 200MHz) 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.

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 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.

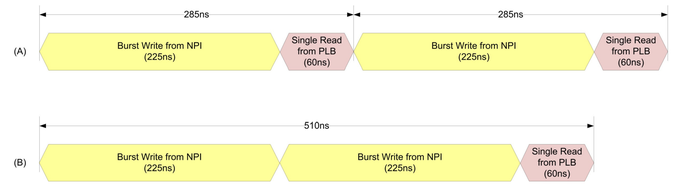

Figure 4. Typical DDR transaction timing diagrams for different MPMC arbitration schemes. (A) Round-robin arbitration. (B) Custom arbitration with the NPI port always the highest priority.

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].

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.

ALS COMMUNICATIONS

* Supported by the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.