ENCODER INTERFACE FOR NSLS-II BEAM LINE MOTION SCANNING APPLICATIONS

Ruslan Kadyrov[#], Kiman Ha, Eli Stavitski, Joseph De Long, Sung-Leung So,
BNL, Upton, New York, USA

Abstract

The NSLS-II synchrotron started operations at BNL, USA. Whereas first beamlines welcome first users, the other beamlines are under design or construction. The variety of motion control applications on existing and future NSLS-II beamlines demand custom control electronics developed to meet specific needs and ease integration into existing systems. Thus an encoder interface was designed for a number of detection techniques that require a live position capture. It implements four identical individually isolated channels with encoder input capturing and output quadrature encoder interface generated. Encoder input handles an incremental quadrature encoder with a digital RS-422A interface and output frequencies up to 10 MHz. The logic, based on Xilinx Virtex-6 FPGA, processes signals from an encoder and associates it with the machine timestamp. Several filtering and compression techniques are also applied. The device then retranslates the interface signals for the motion controller, allowing the device to be installed between encoder and motion controller with no interference to the system. Captured data is streaming to the server using TCP/IP stack, with the server side running an EPICS IOC. The device status and control registers values are also available through EPICS channel access. All operator screens are developed with Control System Studio toolset. The design harmoniously complements the family of the NSLS-II equipment sharing same mechanical and electrical platform. This paper describes the functional design, configuration, operation and current status of the encoder interface design.

INTRODUCTION

Motion scanning applications role in X-Ray experiments consists in cross correlation analysis with data obtained from various optical equipment. Position is usually detected by a quadrature incremental or absolute encoder. Delta Tau GeoBrick motion controller used in NSLS-II motor-based applications doesn't store or timestamp reported position. To keep the design simple, same encoder was decided to be used both for the axis motion control and position capturing. Thus the design should be able to retransmit the encoder signals for the motion controller without interference to the work of the latter.

The logic design concept was inspired by a similar development at Diamond Light Source Ltd [1]. The design instrumentation base shares existing hardware with several devices used for the NSLS-II accelerator FPGA-based applications: power supply controllers [2], beam position monitors [3], cell controller.

[#] rkadyrov@bnl.gov

SYSTEM REQUIREMENTS

The design should provide applicability for various motion scanning applications, easy integration into existing machine control system, and compliance with the NSLS-II beamline instrumentation requirements [4]:

- The design should be able to handle a Renishaw Quadrature incremental encoder with 10 MHz operating frequency aiming to retransmit encoder interface signals for a motion controller allowing bypass connection with minimum or no interference to the encoder connection,
- Relative encoder position is to be timestamped and send to a server allowing to restore an encoder position with 10 kHz sampling rate,
- Standard EPICS driver should be used to minimize device support development needs,
- Ability to start capturing on external trigger or on EPICS PV write,
- Compatibility with existing timing system,
- Flexibility to custom experiment needs, e.g. generating triggers for external devices at certain axis positions or every N steps.

DESIGN COMPONENTS

Instrumentation Base

The design fits in a 2U 19" chassis with hardware based on standard NSLS-II platform. It is a metal enclosure with two power supplies, minimum internal wiring and place for two standard boards mounted on the bottom plate using stand-offs. Front-face mounted FPGA-based main board is in charge of purely digital part of the design, and secondary rear-face mounted interface board for device-specific circuitry. This modular design makes it possible to change functionality or build another device by putting a new interface board and developing a custom firmware for FPGA.

Hardware

The hardware leverages the NSLS-II BPM Digital Front End (DFE) board with Virtex-6 FPGA and periphery [3]. A custom interface board was developed to handle encoder interface specific circuitry. Four identical channels are implemented with each capable to receive and form the quadrature encoder interface signals.

Input and output interfaces are connected to double stacked D-sub 15-pin connectors along the rear panel of

ISBN 978-3-95450-148-9

the device. A standard Renishaw encoder pinout is used for input part and a standard Delta Tau motion controller encoder connector pinout is used for the output interface in compliance with NSLS-II beamline interface instrumentation standard [4].

Interface hardware leverage RS-422A standard with SN75ALS193D receiver IC handling input side and SN75ALS192D driver IC forming output interface. A 120Ohm termination is used to match the receiver to the transmission line impedance. Si8620BC digital isolators along with a 2 W 5 V DC to DC converter for the encoder supply and power domain isolation provide isolation for each channel individually. There is also an option to supply the encoder from a motion controller power directly if the bypass jumper is set.

Conditioned interface signals are passed to the main board to the FPGA buffered I/O through high speed backplane connectors.

Four general purpose TTL inputs and four outputs are also implemented on the board. They are connected to LEMO connectors on the rear panel of the device. The TTL GPIO is isolated and ESD protected with ADUM3402 digital isolator ICs.

Altium Designer software package is used to develop schematics for the board and PADS layout EDA used for PCB design.

Firmware Logic

The design FPGA-based logic for position capturing is implemented in Verilog hardware descriptive language with elements of VHDL using Xilinx ISE 14.7 IDE.

Being passed to the FPGA, the interface signals are processed in hardware modules. Standard modules are used for DDR3 DRAM interface, EMAC module is used for network connection.

The event receiver module (EVR) is used to pass the NSLS-II timing system interface to the logic. It provides a set of triggering events and the timestamp signal which make it possible to store position with 8 ns resolution. For test purposes, there is also an option to generate timestamp internally using on board oscillator.

The FPGA also executes a soft application running on Microblaze, a soft processor core by Xilinx. A set of control and status registers, FIFO, and memory is shared by the modules and the soft processor through the Xilinx PLB system bus and can be used for buffering. The soft processor application runs multiple threads dealing with captured data processing, status and control registers update as well as a serial console for debug and network configuration. Processed data and status registers are being encapsulated in TCP/IP packets using a Xilinx light-weight TCP/IP stack. Microblaze application boots automatically from the on-board flash on device power up. Application was implemented using Xilinx Software Development kit.

Total logic device resources utilization is 6% for registers and 13% for LUTs.

Software

Server side with EPICS soft IOC running communicates with the device through multiple socket connections. Captured position, GPIO lines state; device registers are available as PVs through EPICS channel access leveraging pscdrv, a portable streaming controller EPICS generic FPGA driver toolkit [5]. The driver also unpacks the streaming data from binary packets and put it into waveform PV records.

PV values than be written to the file using EPICS aSub routine for further plot and cross analysis together with data obtained from detectors and other optical equipment.

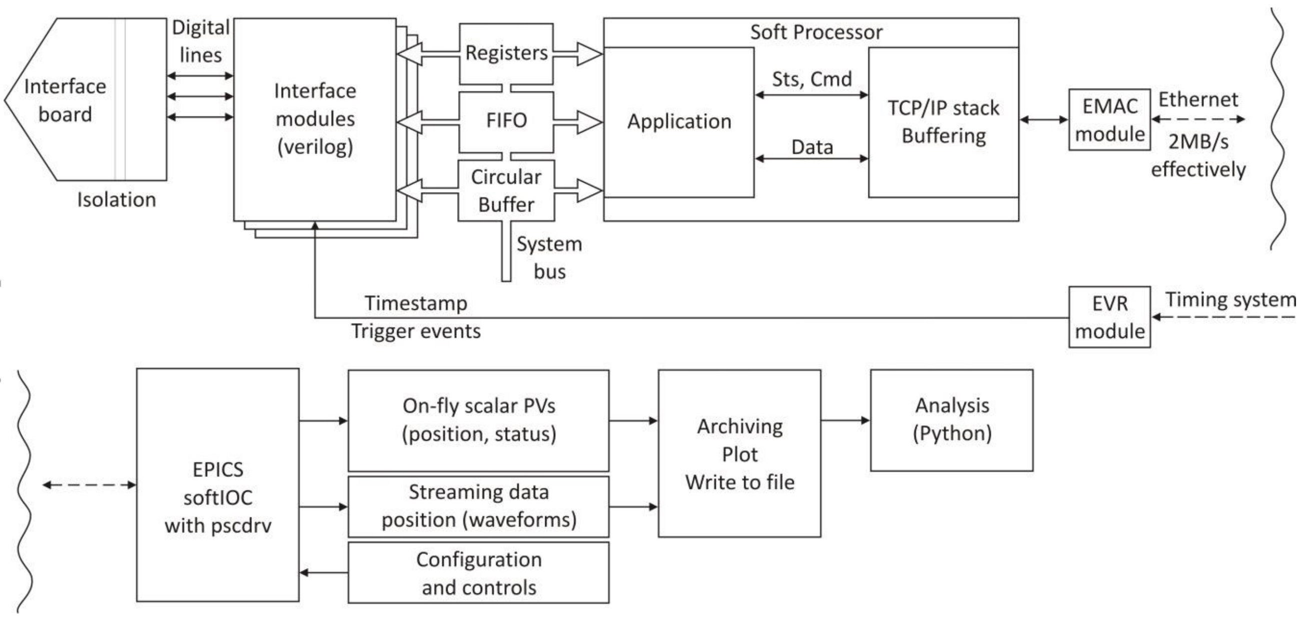

Figure 1: Simplified design dataflow

Operator Panel

To ease work with the device registers and data representation, an operator screen was developed using Control system studio [6]. It allows user to choose operation mode as well as configure device parameters, check its status and operation statistics. Built-in Python scripts are resided for data plot and work directory management.

FUNCTIONAL DESIGN

Position Capturing

Encoder position change generates multiple pulses with 90 degrees phase offset on A and B digital lines of the interface. Quadrature decoder hardware module catches every single transition of quadrature signals using a simple hardware circuit shown in Fig. 2. Every movement detected is captured, and a relative position counter changes accordingly. A reference clock signal of 100 MHz is used to ensure oversampling for connected 10 MHz encoder. Extra D flip-flops are used to avoid metastability.

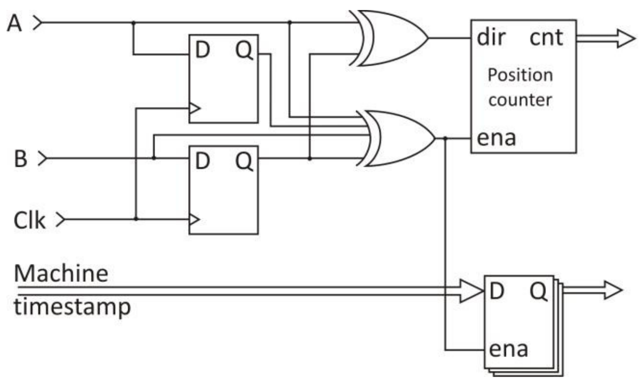

Figure 2: Position capturing simplified hardware circuit.

Encoder signals, relative position counter along with GPIO input states for the current transition are associated with the timestamp received from the timing system at the moment of movement detection. With every single transition detected a 16-byte portion of data (chunk) is captured (see Table 1). Built-in counters and status bits provide diagnostic means for missing position steps, missing pockets and encoder misconnection.

Table 1: Interface chunk data format

Bytes	Field name	Field description
15 to 12	Timestamp, sec	POSIX time of transition
11 to 8	Timestamp, ns	Nanosecond portion of timestamp, 8 ns resolution
7 to 4	Index counter	Chunk sequential number
3 to 1	Position counter	Relative position detected
0	Data byte	Interface signals state

Data byte is combined binary status of signals with A, B and reference mark Z encoder signals, encoder proper connection state followed by states of four GPIO inputs. Index counter with sequential chunk number is also added to ensure data proper processing and storing in the buffer as well as consistency in the data stream and while restoring it for the post analysis.

Circular Buffer

Data chunk with timestamp header is than stored in a circular buffer through an NPI interface. 1 GB of an on-board DDR3 DRAM is used to accommodate the buffer giving maximum read/write bandwidth 800 Mb/s. Thus 256 Mb of buffer space is available per channel. Auxiliary FIFO buffer is used between an encoder interface channel handling hardware module and the NPI for initial data accumulation. Once FIFO reports to be half-full, a relatively rapid NPI transaction reads it out.

Once a certain number of chunks are stored in the buffer, a TCP/IP packet is initiated. Chunk number threshold for a new packet is hardcoded to 1024. To ensure none of the saved chunks left in the buffer at the end of an experiment, circular buffer flush mechanism is implemented. In case of the buffer overflow, a skipping logic erases all or a part of data stored in the buffer. The buffer fill level is also available through the register value as a PV and operator panel widget.

Position Filtering

In an application with high speed movements with an encoder operating at frequencies close to the maximum required, position capturing generates enormous data volume that can overflow the circular buffer causing data skip. Also filtering should be used when the data is streaming from all four channels simultaneously due to TCP/IP stack bandwidth limitation of 2 MB/s of useful data. To use the buffer resources effectively and ensure TCP/IP stack proper operation, a number of filtering techniques are implemented. Among them, the position filtering feature leverages a window-based approach, when a new chunk is formed only if a relative position changes for a number of steps exceeding a predefined threshold level during given period of monitoring, thus the position doesn't fit in a $dY - dT$ window.

Several special use cases are possible: with null dY every single event is detected at the end of dT period. Filter with $dY = 1$ parameter catches the chunk if more than one event was detected during dT. Using dY values exceeding 3 is discouraged as it can filter out a steady drift movement. Setting a null size window disables the filter, and every single movement is registered at the moment of its detection. This can generate large volumes of useless data for fast encoders even in idle state in mechanically noisy environment. Default filter configuration is null dY with dT adjusted to the maximum encoder speed required assuming every step detected during 100 ns

Generation: Bypass Mode

In bypass mode input encoder interface signals are mirrored inside FPGA providing a negligible delay for output interface. Bypass mode leaves reconstructed signals as they were received with no impact to the motion controller operation. There is also an option to disable generation at any mode using disable pin of an output driver IC. Bypass mode is set as default, with output drivers enabled.

Generation: Test Mode

Encoder signals pulse train generation is also implemented for the test purposes. A series of pulses of a configurable width, period and count simulating a real encoder signals transitions can be generated to provide a self-testing fixture if the output interface is looped back to the input simulating a real encoder channel.

Generation: Filter Mode

Another feature implemented implies input interface retransmission with a filtering before retransmission. A filtering window is applied to the raw position captured. If during the *dT* timing interval there was less than *dY* position offset detected, the interval is retransmitted without transitions on the lines as if an encoder didn't report any movement. It is up to the user to set an appropriate *dY* value in order not to filter out a useful position changes, e.g. a continuous drift. Setting *dY* to zero allows generation exact number of steps to catch up the real position at the end of previous *dT* timing interval.

Generation can be performed with a different rate then the real encoder is operated on, faster or slower. If regeneration rate exceeds an input rate, an overflow may occur when a filtered position cannot be reached on a lower rate.

The mode can be used to filter out unnecessary micro movements in noisy mechanical environment as well as roughing an encoder with an excessive resolution.

Control and Status Registers

Status update soft processor thread is continuously running transmitting status registers with 1 Hz update frequency. Circular buffer load and skip counter indicators, relative encoder position, GPIO inputs state, and device timestamp are examples of status fields.

Control packets are sent on demand. Trigger PV is toggled to initiate transaction if a writing request to one of the controls registers is received. Position and generation filters configuration, encoder direction reverse bit, timestamp source, generation enable bit are examples of device controls.

TTL Triggers and GPIO

Auxiliary digital TTL I/O is implemented for external triggering. Every transition of an input is timestamped and stored in the circular buffer as a part of an encoder chunk. Output lines state are mirrored from the register value allowing user to toggle an output state by writing to a PV or configuring soft processor application for user-specific needs such as triggering an external device at certain relative positions of an encoder.

SUMMARY

The product implements position capturing for motion applications with quadrature incremental encoders. Mechanical 2U 19" platform, internal wiring and Virtex-6 FPGA based main board are shared with other devices used on the facility. The encoder signals are passed to FPGA and then re-transmitted for the motion controller, allowing the device to be installed between encoder and motion controller with no interference to the system.

Ability to supply the encoder directly from the board, as well as generating internal timestamp and output encoder interface make it possible to use the design without direct connection to the machine infrastructure.

The design is in production state with first series of eight pieces built and tested waiting to be used in scanning applications at Inner Shell Spectroscopy (ISS) and Soft Matter Interfaces (SMI) beamlines.

Similar design concepts are going to be realized in a new design of general purpose data acquisition and control board. This and future design harmonically complements and extends application variety for the device family at the NSLS-II experimental floor.

REFERENCES

[1] T. Cobb et al., “Zebra: a Flexible Solution for Controlling Scanning Experiments”, ICALEPCS’13, San Francisco, CA, USA, October 2013, TUPPC069, p. 736 (2014); http://www.JACoW.org.

[2] W. Louie et al., “NSLS-II Power Supply Controller”, PAC’11, New York, NY, USA, March 2011, TUP193, p. 1187(2011); http://www.JACoW.org.

[3] Om Singh et al., “NSLS-II Beam Position Monitor Embedded Processor and Control System”, ICALEPCS’11, Grenoble, France, October 2011, TUBL1, p. 316(2013); http://www.JACoW.org.

[4] A. Broadbent, W. Lewis, D. Chabot, “Beamline Systems Instrumentation Interfacing Standard” NSLS-II LT-C-XFD-SPC-CO-IIS-001 Version 5, April 2013.

[5] https://github.com/mdavidsaver/pscdrv.

[6] http://cs-studio.sourceforge.net/.

ISBN 978-3-95450-148-9