INITIAL DESIGN OF THE FAST ORBIT FEEDBACK SYSTEM FOR DIAMOND LIGHT SOURCE

I. S. Uzun, R. Bartolini, G. Rehm, J.A. Dobbing, M. T. Heron, J. Rowland
Diamond Light Source, Rutherford Appleton Laboratory, Chilton, Didcot
Oxon, OX11 0QX, United Kingdom

ABSTRACT

This paper presents the initial design and implementation of the global Fast Orbit Feedback (FOFB) system for Diamond Light Source to enhance electron orbit stability. The FOFB system consists of 168 electronic Beam Position Monitors (BPMs), 24 VME-based feedback processors and a FOFB communication network to distribute the beam position values to the feedback processors. The main topic of the paper is the design of the communication controller (CC), which is central to the data distribution and its FPGA-based implementation. The CC design is unaware of the network topology so enables a number of options for the physical structure and further supports diverse routes so enabling operation to continue with single and multi-link faults.

INTRODUCTION

Diamond Light Source is the 3rd generation 3 GeV electron synchrotron currently under construction in the UK. The facility will comprise a 3 GeV electron storage ring, injected from a 100 MeV linac through a full energy booster synchrotron, and has approval for 22 beamlines.

The global Fast Orbit Feedback (FOFB) system for Diamond Light Source is currently being implemented. The FOFB system will stabilise the electron beam position towards an ideal electron beam orbit.

The FOFB system consists of:

- 168 Electronic Beam Position Monitors (BPMs) located around the storage ring,
- 24 VME-based processors running the feedback algorithm,
- Power supply controller units for 96 fast corrector and 168 slow corrector magnets, and
- FOFB communication network to distribute the beam position values.

This paper reports on the design and realisation of FOFB communication network and in particular the implementation of the Communication Controller.

FOFB SYSTEM OVERVIEW

Diamond Storage Ring consists of 24 identical lattice cells. Each cell has control and instrumentation for that area in an air conditioned, temperature stabilised room, called a CIA. Each CIA houses the racks and instrumentation including a diagnostics rack and a power supply rack as shown in Figure 1.

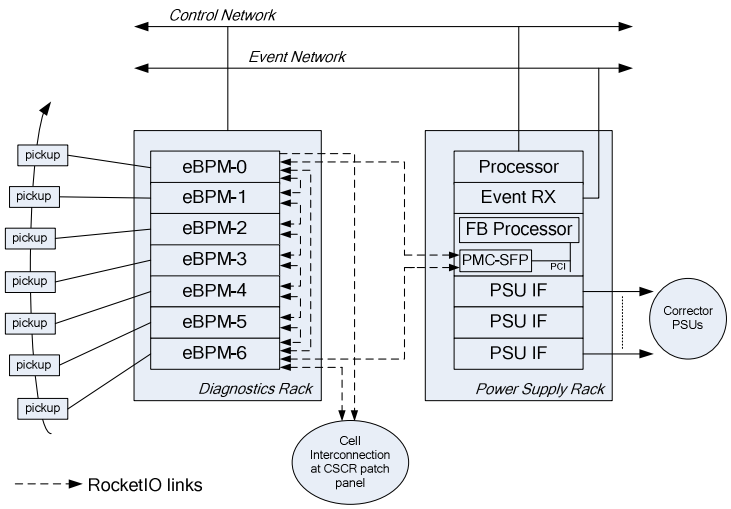

Figure 1: Storage Ring Cell Block Diagram.

The diagnostics rack contains 7 Libera BPMs. Each BPM is connected to the set of four button-type pickup electrodes mounted to the vessel. The analog signals picked up by the electrodes are a function of the beam position and intensity. The beam position values are calculated by BPMs from a suitable scaled difference-over-sum calculation.

The power supply rack holds an EPICS IOC and a second MVME5500 processor dedicated to feedback algorithm calculation. A PMC-SFP interface module is connected on the PCI bus of the feedback processor in order to receive (and forward) beam position values from the FOFB communication network.

FOFB SYSTEM PHYSICAL REALISATION

Hardware Components

Following hardware components have been used in the realisation of the FOFB system:

Libera BPM: Instrumentation Technologies is delivering the beam position monitor hardware, called Libera [1], which has a Xilinx XCV2VP30-5 Virtex-II Pro Field Programming Gate Array (FPGA). Libera will acquire electron beam position data and provide down-sampled and filtered beam position values to the fast feedback Communication Controller (CC) subsystem at fast feedback rate around 10kHz. It also provides eight Small Form factor Pluggable (SFP) slots physically connected to FPGA RocketIO links which are used for the implementation of the FOFB communication network.

Feedback Processors: A Motorola MVME5500 VME processor with VxWork real-time operating system will be used for running the feedback algorithm.

PMC-SFP Interface Module: The PMC-SFP Interface Module is responsible for distributing the beam position values to the feedback processors. It will be placed on the PMC locations of the feedback processors. The module has a Xilinx XCV2VP30-6 Virtex II Pro FPGA with four RocketIO links used for connecting the module to the FOFB communication network. The module is mainly based on the PMC Event Receiver module from Micro Research [2].

FOFB Communication Network

The FOFB communication network is implemented by fast, serial RocketIO links. The BPM digital board provides 106MHz clock source to FPGA to drive the RocketIOs. Therefore, a serial data communication rate of 2.12Gbps (20 x 106M) is achieved.

The links between BPMs in the diagnostic rack in each cell have been implemented using electrical high-speed copper SFP-SFP/.5m patch cables from CS Electronics[3]. Network topology implemented between BPMs in each cell is shown in Figure 1. “BPM-0” and “BPM-6” are assigned as *Primary BPMs* to be used for connections to PMC-SFP module and Control Systems Computer Room (CSCR) patch panel.

The links between the primary BPMs and PMC-SFP module in the same cell have a length of 5m. Therefore, optical links are required. Agilent Technologies multimode fiber-optic LC transceivers and Duplex LC-LC 5m OM3 fiber patch cables are used to implement these links.

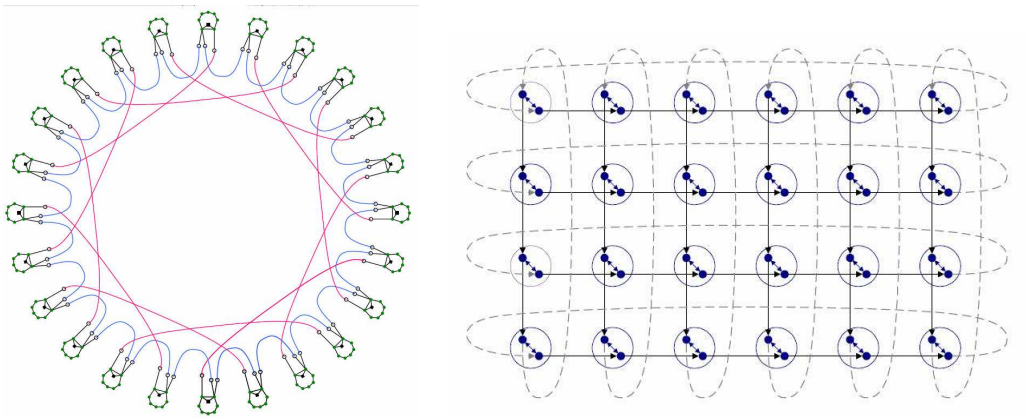

Figure 2: FOFB communication network topologies. Optimised Ring topology and Torus Mesh topology.

Primary BPMs in each cell are connected to the patch panel at CSCR where the connections between the cells are implemented. These links have a total lengths upto 700m, therefore they are implemented using single mode optical transceivers and Duplex LC-LC OS1 fibers. Although the FOFB system enables us to implement any network topology, we have considered implementing two network topologies proposed by SuperComputing Systems [4] as shown in Figure 2.

FOFB COMMUNICATION

Distribution of BPM Position Values

The feedback processor in each CIA uses the position values from all 168 BPMs around the storage ring for running feedback algorithm. Therefore, beam position values from all of the BPMs around the ring have to be distributed to all 24 feedback processors.

In the current communication scheme, time is divided into "time frames" of equal length. In the beginning of each time frame, each BPM injects its own beam position values to the FOFB network, and then starts a forwarding or discarding process on the BPM values that it receives. Each BPM position then propagates to all 168 BPMs and 24 PMC-SFP modules on the network before the end of the time frame.

The BPM values are updated on the feedback processors with the rate of 10 kHz. This results in the time frame duration of 100 µsec.

Packet Format and Communication Protocol

Each BPM injects a packet containing its own position values into the communication network when the "time frame start" pulse is generated by Libera timing core. The packet is broadcast to all connected nodes. Each node forwards received packets only once to all connected nodes.

BPM x^- and y^- beam position values are enclosed in a 20-byte payload as shown in Table 1.

Table 1:BPM Packet Payload Structure.

Field	Size	Description
Header	32 Bit	BPM ID, Source type, TimeFrameStart
x-Position	32 Bit	x-Position of the beam [nm]
y-Position	32 Bit	y-Position of the beam [nm]
Reserved	32 Bit	Reserved
Time Stamp	32 Bit	Time stamp of position data acquisition [ns]

The packets are protected by 32-bit invariant CRC. The RocketIO transmitter computes the 4-byte CRC of the payload data. If a CRC error occurs, the packet is discarded and no retransmission is issued. The redundancy in the network topology will deliver the same BPM packet through another path.

COMMUNICATION CONTROLLER DESIGN AND IMPLEMENTATION

The architectural block diagram of the BPM Communication Controller (CC) is shown in Figure 3. It has three main modules.

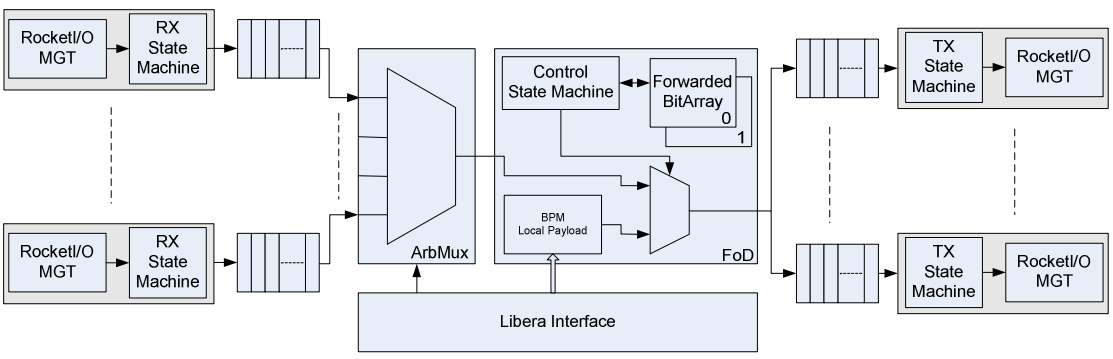

Figure 3: CC Design for BPM

Receiver Block

The receiver block consists of the following modules:

- RocketI/O MGT Receiver Channel
- Receiver Logic
- RX State Machine
- Five 32 x 32-bit RX FIFO

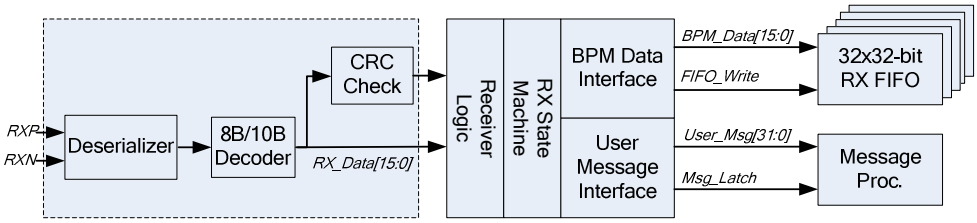

Figure 4: CC Receiver Block.

When a RocketI/O transceiver receives a packet through its RX channel, it deserialises this stream into 10-bit data and control symbols, known as the link layer payload. After deserialisation, the link layer payload is decoded into a stream of octets.

The receiver logic handles link layer payload stripping. The link layer stripping procedure includes removal of framing characters (/SOP/ and /EOP/), idle characters and clock compensation sequences. The RX state machine controls the receiver logic, RX FIFO and FOFB message processor interface.

Data Processing (Store, Analyse and Forward) Block

The CC transmits a packet over all its output links at the same time. This multiplies an incoming packet by the number of output links. To prevent flooding of the FOFB communication network, the CC transmits a specific BPM value only once in a time frame. If the received BPM packet has already been forwarded in the present time frame, CC discards the packet.

Outputs of the RX FIFOs are fed in to Arbiter/Multiplexer (ArbMux) module. The data path from each channel at the output of the RX FIFO is (5*32)-bit wide. The ArbMux module consists of a simple state machine implementing the round-robin algorithm in order to retrieve queued packets from all incoming channels. Each channel is assigned four (4) time slots as long as there is data queued in the channel FIFO. The *ActiveChannel* variable keeps the channel number where data is being read. If the FIFO for the *ActiveChannel* is empty, *ActiveChannel* variable is incremented by 1 to retrieve data from the next available channel.

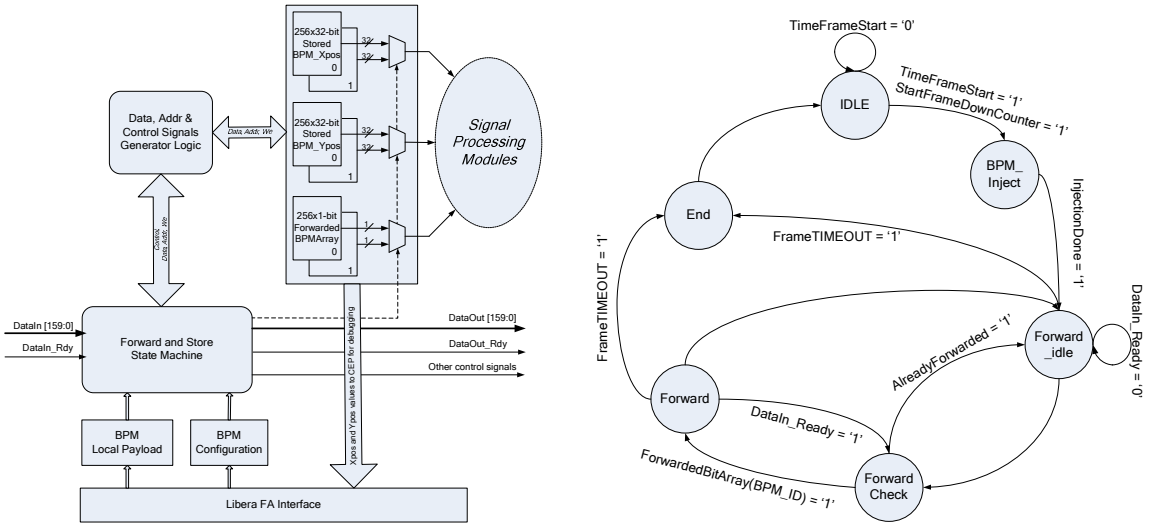

Figure 5: Forward and Discard Module (a) architecture (b) state machine.

The Forward or Discard (FoD) module is responsible for the injection of the BPM's own payload at the beginning of each time frame, and then processing of the received BPM payloads. The FoD receives one BPM payload at a time from the ArbMux output. The data path in the FoD module contains a parallelised five by 32-bit payload. The control state machine, shown in Figure 6, checks the ForwardedBitArray look-up table to determine if the arbitrated BPM value has already been forwarded in this time frame. If not, it forwards the value by writing the payload on all output TX FIFOs and sets the corresponding bit in the ForwardedBitArray look-up table. Else, it discards the BPM payload. In addition, x^- and y^- position values from each BPM are stored in a double buffer to make the BPM values available for further signal processing in case they are required. When a new time frame starts, all three buffers (shown in Figure 6) are switched.

Transmitter Block

Each output channel has five 32x32-bit TX FIFOs. The transmitter block sends out packets when the TX FIFO is not empty. The 5x32-bit BPM payload or 32-bit user message is encapsulated with control symbol sequences, /SOP/ and /EOP/. The resulting data structure is referred to as the *link layer payload*. The link layer payload is 8B/10B encoded and then transmitted through the channel.

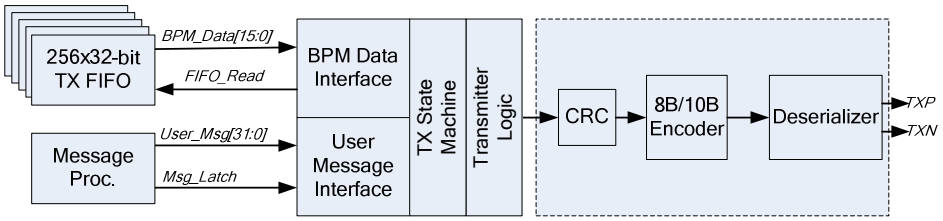

Figure 5: CC Transmitter Block.

CC Design on PMC

The CC design for PMC is very similar to the BPM design. The main differences are listed below.

- An interrupt is sent to the feedback processor when all values have been received, or at end of a TimeOut period.
- Since the PMC module does not have its own BPM beam position values, it doesn't perform an injection.
- The PMC module has no direct connection to the Libera timing core, therefore it detects the start of a time frame by using the "Time Frame Start" bit embedded in the packet header.

FPGA Implementation Results

The communication controller design has been implemented in the VHDL language. The design has been synthesised for a Xilinx XC2VP30-5 FPGA using the Xilinx ISE development platform. FPGA resource usage in terms of FPGA area (slices) for the main blocks of the CC design is given in Table 2.

The CC design with four RocketIO channels on the BPM uses 14% of the available FPGA area (2050 out of 13696 Slices).

Table 2: BPM CC FPGA resource usage.

Module	**Slices**	**Flip Flops**	**4 input LUTs**	**BRAMs**
ArbMux	239	70	462	
FoD	255	393	396	6
1 RIO Chanell	222	337	236	8
RX and TX State Machines	59	84	106	
Whole BPM CC design with 4 RIO channels	2050	3039	3252	38

FPGA firmware integration

The Fast Application Interface (FAI) specification has been developed with Instrumentation Technologies for easy integration of Diamond's communication controller design into the Libera FPGA core. The top-level VHDL entity declaration that will be used for firmware integration for the BPM CC design is given in Figure 7.

```
entity fofb_cc_bpm is
   generic (
      RioCount       : integer := 4;                                  -- Number of MGTs on BPM-CC
      PacketSize     : integer := def_PacketSize;                     -- Packet size in 32-bit words
      BpmNo          : std_logic_vector(9 downto 0) := "0000000011"; -- BPM ID for simulation
      BpmCount_g     : std_logic_vector(9 downto 0) := "0000000111"  -- BPM Count for simulation );
   port (
      -- Clocks
      adcclk              : in std_logic;                             -- ADC rate clock
      clk106              : in std_logic;                             -- 106MHz clock for RocketIO design
      clk125              : in std_logic;                             -- 125MHz system clock
      -- Fast Acquisition Data Interface
      fai_fa_block_start  : in std_logic;                             -- Transfer of block data in progress
      fai_fa_data_valid   : in std_logic;                             -- Clock signal for incoming data
      fai_fa_d            : in std_logic_vector(15 downto 0);         -- Fast acquistion data
      -- FOFB Communication Controller Configuration Interface
      fai_cfg_a           : out std_logic_vector(10 downto 0);        -- Address bus for the cfg space
      fai_cfg_do          : out std_logic_vector(31 downto 0);        -- Data to be written to CFG BRAM
      fai_cfg_di          : in  std_logic_vector(31 downto 0);        -- Data being read from CFG BRAM
      fai_cfg_we          : out std_logic;                            -- Write Enable signal to BRAM to write Config Data
      fai_cfg_clk         : out std_logic;                            -- Clock signal from the CC side to CFG BRAM
      fai_cfg_act_part    : in  std_logic;                            -- Active part indicator for BRAM space (0 or 1)
      -- Serial I/Os for eight RocketIOs on the Libera
      fai_rio_rdp         : in  std_logic_vector(7 downto 0);         -- Receive data
      fai_rio_rdn         : in  std_logic_vector(7 downto 0);         -- Inverse receive data
      fai_rio_tdp         : out std_logic_vector(7 downto 0);         -- Transmit data
      fai_rio_tdn         : out std_logic_vector(7 downto 0);         -- Inverse transmit data
      -- RocketIO transceiver's monitor and control interface
      fai_rio_los         : in  std_logic_vector(7 downto 0);         -- Loss of optical input signal
      fai_rio_tdis        : out std_logic_vector(7 downto 0);         -- Disable trasmitters (0:Enable/1:Disable)
      fai_rio_txf         : in  std_logic_vector(7 downto 0);         -- Transmitter fault indication
      fai_rio_def2        : inout std_logic_vector(7 downto 0);       -- Data line for the I2C to trasmitter EEPROM
      fai_rio_def1        : out std_logic_vector(7 downto 0);         -- Clock line for the I2C to trasmitter EEPROM
      fai_rio_def0        : in  std_logic_vector(7 downto 0);         -- Tied to GND within transmitter
      fai_rio_rate        : out  std_logic_vector(7 downto 0);        -- Rate select pin
      -- Slow Acquisition Data Interface
      fai_sa_wr_en        : in std_logic;                             -- Write enable signal for FIFO
      fai_sa_data_valid   : in std_logic;                             -- Clock signal for data
      fai_sa_a            : in std_logic_vector(2 downto 0);          -- Offset to FIFO pointer
      fai_sa_d            : in std_logic_vector(31 downto 0);         -- Slow acquisition data
      -- RS485 interface for steerers
      fai_485x_txd        : out std_logic;                            -- TXD for X dimension steering
      fai_485x_rxd        : in std_logic;                             -- RXD for X dimension steering
      fai_485y_txe        : out std_logic;                            -- Transmit enable for Y dimension steering
      fai_485y_txd        : out std_logic;                            -- TXD for Y dimension steering
      fai_485y_rxd        : in std_logic                              -- RXD for Y dimension steering
);
end fofb_cc_bpm;
```

Figure 7 BPM CC design VHDL entity declaration.

FOFB MANAGEMENT

A set of control and status monitor registers have been defined and implemented in the current CC design. A BPM management interface will be developed to access (read and write) CC configuration registers (ie. BPM_ID, BPM_Count), and monitor the CC operation (ie. correct link wiring, link statistics) through EPICS.

ACKNOWLEDGMENT

The work presented in this paper is based on the study "DLS FFS communication specification" by Leo Breuss and David Müller from Supercomputing Systems AG, Switzerland [4].

REFERENCES

[1] The Libera product. http://www.i-tech.si/products-libera.html
[2] Micro-Research Finland Oy, "Final design Review, Diamond Light Source, Timing System Modules", 2004.
[3] http://www.cselex.com/fibre-channel-cables.htm
[4] DLS FFS Communication Specification, Supercomputing Systems, 2004.