BPM packet router for IBFB

ibfb_bpm_router
Revision 1.0

Firmware Data Sheet

PSI, 23.06.2016

Content

Table of Contents

1 Introduction

The IBFB system uses data from a set of BPMs in order to compute the corrections to be applied through the kickers. Since the number of available BPMs is high, it's not feasible to connect each of them to the IBFB controller. Instead a double daisy-chain topology is used: each BPM transmits data to the previous BPM in the chain (upstream link) and to the following (downstream link). In addition each BPM forwards to the upstream link the data coming from the downstream link. Likewise the data from the downstream link is forwarded to the upstream link. A simplified diagram of the BPM links is shown in Figure 1.

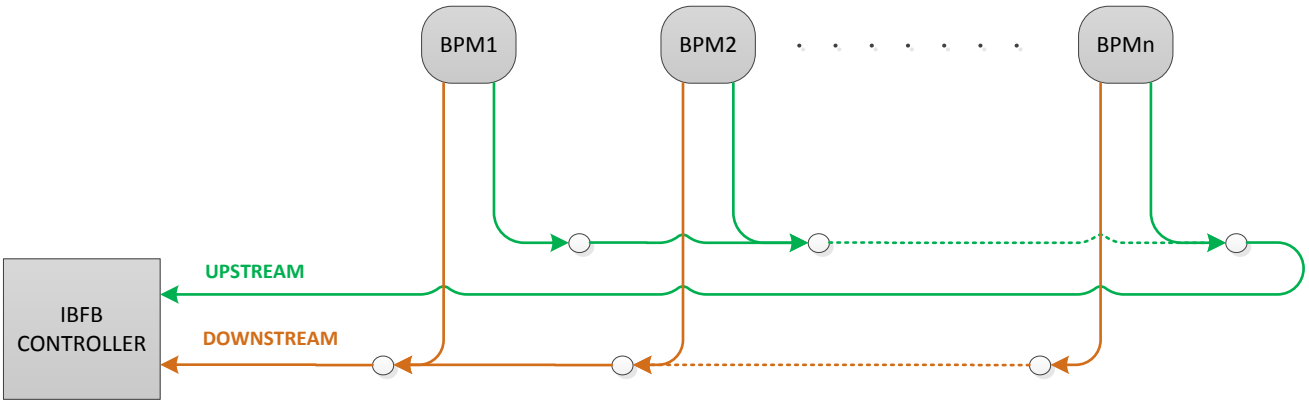

Figure 1 - Data link between BPMs (simplified)

In practice each GPAC board contains two FPGAs, each implementing the digital signal processing firmware for a different BPM. Each FPGA has one link available for the interconnection, so the daisy chain is implemented using also the internal links between the two FPGAs, as shown in Figure 2.

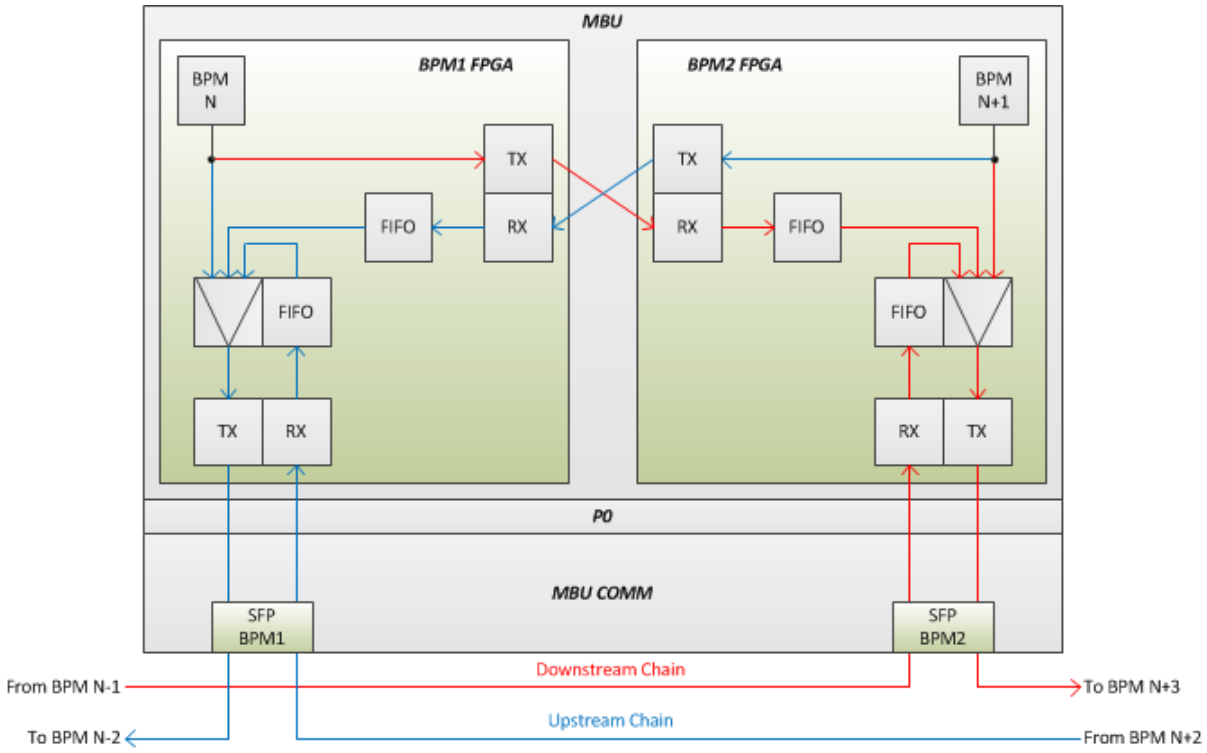

Figure 2 - Implementation of a single node of the daisy chain

The two FPGAs contain the same firmware. Each FPGA collects the data from the BPM, packages into a custom protocol (see §2.4.2 for details) and sends it both to the backplane link (through the P0 connector and COMM rear-transition-module) and to the other FPGA (through the cross-FPGA link). Moreover, data received from the cross-FPGA link is forwarded to the backplane link, while data from the backplane link is just forwarded to the backplane link itself.

1.1 Features

The *ibfb_bpm_router* component has the following features:

- PLB register interface for command and control (tested up to 125 MHz PLB clock frequency).
- Provides two fast full-duplex serial links. One on the backplane through the P0 connector, and one for cross-FPGA communication.
- Tested up to 3.125 Gbps link speed (corresponding to 78.125 MHz core clock frequency)
- Real-time collection of data from BPM (BPM_CAV_EXFEL component [2]). Data is formatted into packets and sent through the upstream and downstream links.
- Data received from the backplane link is forwarded through the TX channel of the same link.
- Data from the cross-FPGA link is forwarded to the backplane link.

1.2 Definitions, acronyms, and abbreviations

FPGA	Field Programmable Gate Array
PLB	Processor Local Bus (by IBM)
GPAC	Generic PSI ADC Carrier
BPM	Beam Position Monitor
IBFB	Intra-bunch Feedback

1.3 References

[1] ”Generic PSI ADC Carrier, VME64x FPGA Carrier Board for High Speed Mezzanines”, *Datasheet*, Board Rev. 2.1, 12.07.2013.
[2] “European XFEL cavity BPM bpm_cav_exfel”, Datasheet, Rev. 1.5, 27.10.2015
[3] Virtex-5 FPGA RocketIO GTX Transceiver, User Guide, UG198, V3.0, 30.10.2009

1.4 History

Revision	Date	Author	Description
1.0	23.06.2016	A. Malatesta	First release

2 Firmware Description

The core is implemented in pure VHDL. It does contain device specific primitives (Xilinx's GTX Transceivers), therefore its portability is currently limited to Xilinx Virtex5 devices providing at least two GTX_DUAL tiles. The component is wrapped in a Xilinx EDK pcore with slave PLB interface, but its core logic is also available as an independent VHDL component.

2.1 Architecture

The core's architecture is shown in Figure 3. BPM data is received from the processing core **bpm_cav_exfel** through a custom interface (cfr. [2]). The block **CDC FIFO & INTERFACE** takes care of the clock-domain crossing between the interface's clock and the core's clock, and generates additional data like the bucket number (cfr. §2.4.1). The block **IBFB_PACKET_TX** reads data from the input FIFO and formats it into packets according to the protocol described in §2.4.2. The input packets are then written to the BPM packet FIFO. The block **PACKET_ROUTER** collects the incoming packets from the BPM packet FIFO and the GTX links, and routes them to the appropriate destination: content from the BPM packet FIFO has priority over other channels, and is forwarded both to the backplane link (P0 GTX) and to the cross-FPGA link (BPM GTX). Packets coming from both the backplane and the cross-FPGA receive channels are forwarded to the backplane TX channel (cfr. §2.4.3 for details).

In addition, a PLB slave controller allows the access to a register bank for command and control.

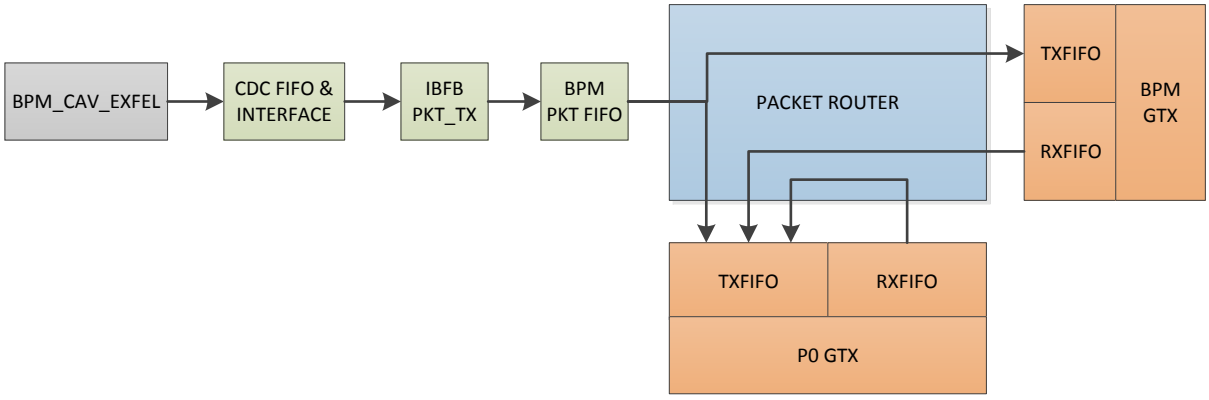

Figure 3 – Top-level block diagram of the core.

2.2 Ports and Attributes

Figure 4 contains component overview with ports and attributes. Table 1 and Table 2 describe the meaning of port and attributes.

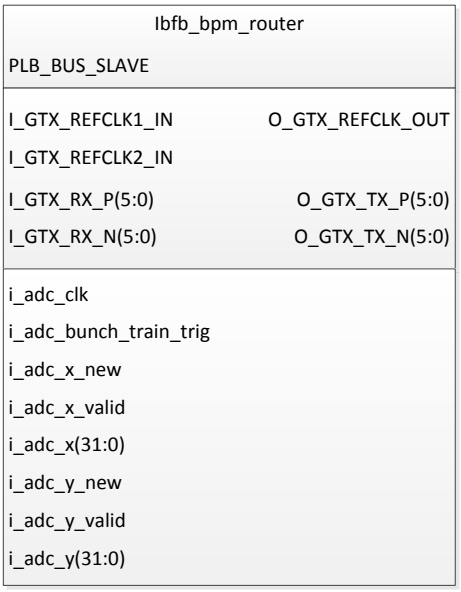

Figure 4 - Top entity of the ibfb_bpm_router.

Table 1 - Port description for ibfb_bpm_router

Port name	Dir.	Type	Description
I_GTX_REFCLK1_IN	in	std_logic	First reference clock for the GTX transceivers. Any of the three GTX tiles contained in the component can be assigned to use this clock by properly setting the generic C_GTX_REFCLK_SEL (cfr. Table 2).
I_GTX_REFCLK2_IN	in	std_logic	Second reference clock for the GTX transceivers. Any of the three GTX tiles contained in the component can be assigned to use this clock by properly setting the generic C_GTX_REFCLK_SEL (cfr. Table 2).
O_GTX_REFCLK_OUT	out	std_logic	NOT CONNECTED
I_GTX_RX_P	in	std_logic_vector(5:0)	Positive receive wires for the GTX physical layer differential pairs (see Table 3 for GTX channel assignment).
I_GTX_RX_N	in	std_logic_vector(5:0)	Negative receive wires for the GTX physical layer differential pairs (see Table 3 for GTX channel assignment).
O_GTX_TX_P	out	std_logic_vector(5:0)	Positive transmit wires for the GTX physical layer (see Table 3 for GTX channel assignment).
O_GTX_TX_P	out	std_logic_vector(5:0)	Negative transmit wires for the GTX physical layer (see Table 3 for GTX channel assignment).
i_adc_clk	in	std_logic	CAV_BPM_EXFEL interface: clock (166.67 MHz). See [2] for details.
i_adc_bunch_train_trig	in	std_logic	CAV_BPM_EXFEL interface: trigger signaling the start of a new bunch. See [2] for details.
i_adc_x_new	in	std_logic	CAV_BPM_EXFEL interface: when this signal is high, then the ports *i_adc_x_valid* and *i_adc_x* can be sampled. See [2] for details.

Port name	Dir.	Type	Description
i_adc_x_valid	in	std_logic	CAV_BPM_EXFEL interface: this signal is valid whenever *i_adc_x_new* is high. If it's asserted, then the measure on the bus *i_adc_x* is valid. Otherwise the measure is not valid. See [2] for details.
i_adc_x	in	std_logic_vector(31:0)	CAV_BPM_EXFEL interface: this bus is sampled when *i_adc_x_new* is high. Its value represent the beam's X-position. The measure is deemed valid only when *i_adc_x_valid* is asserted, otherwise the value should be discarded. See [2] for details.
i_adc_y_new	in	std_logic	CAV_BPM_EXFEL interface: when this signal is high, then the ports *i_adc_y_valid* and *i_adc_y* can be sampled. See [2] for details.
i_adc_y_valid	in	std_logic	CAV_BPM_EXFEL interface: this signal is valid whenever *i_adc_y_new* is high. If it's asserted, then the measure on the bus *i_adc_y* is valid. Otherwise the measure is not valid. See [2] for details.
i_adc_y	in	std_logic_vector(31:0)	CAV_BPM_EXFEL interface: this bus is sampled when *i_adc_y_new* is high. Its value represent the beam's X-position. The measure is deemed valid only when *i_adc_y_valid* is asserted, otherwise the value should be discarded. See [2] for details.
PLB_SLAVE	in	BUS	PLB slave bus (non bursting) used to access local register bank for command and control purposes. PLB bus clock is used to drive only the register bank logic. The core itself runs with the transceiver's parallel clock.

Table 2 - Attribute description for ibfb_bpm_router

Attribute Name	Type	Description
C_K_SOP	std_logic_vector(7:0)	K-character used as Start-of-packet symbol in IBFB protocol. Shall be K-character accepted by Xilinx GTX transceivers 8b/10b logic (see [3]).
C_K_EOP	std_logic_vector(7:0)	K-character used as End-of-packet symbol in IBFB protocol. Shall be K-character accepted by Xilinx GTX transceivers 8b/10b logic (see [3]).
C_GTX_REFCLK_SEL	std_logic_vector(2:0)	Define which reference clock is used by every GTX slice. Each bit represent a slice: bit 2 selects clock for cross-FPGA channels 2 and 3, bit 1 for cross-FPGA channels 0 and 1, bit 0 for backplane channels 0 and 1. Value 0 selects REFCLK0, while value 1 selects REFCLK1. According to which physical transceivers are used, restrictions apply (see [3] for details).
C_P0_REFCLK_FREQ	natural	Reference clock frequency for backplane channels (in Megahertz)
C_BPM_REFCLK_FREQ	natural	Reference clock frequency for cross-FPGA channels (in Megahertz)
C_P0_BAUD_RATE	natural	Baud rate for backplane channels (in Kilobits per second)
C_BPM_BAUD_RATE	natural	Baud rate for cross-FPGA channels (in Kilobits per second)

Table 3 - GTX channels assignment.

GTX channel index	Signals	Description
GTX P0 Channel 0	I_GTX_RX_P(0) I_GTX_RX_N(0) O_GTX_TX_P(0) O_GTX_TX_N(0)	Connected to GPAC's P0_2MGT_*(1) signal. Not used.
GTX P0 channel 1	I_GTX_RX_P(1) I_GTX_RX_N(1) O_GTX_TX_P(1) O_GTX_TX_N(1)	Connected to GPAC's P0_2MGT_*(0) signal. Used for the BACKPLANE link. GTX_P0.1 => P0_2MGT(0) => BPMx_2MGT(0) => SFP
GTX BPM channel 0	I_GTX_RX_P(2) I_GTX_RX_N(2) O_GTX_TX_P(2) O_GTX_TX_N(2)	Connected to GPAC's BPM_4MGT_*(0) signal. Used for the CROSS-FPGA link.
GTX BPM channel 1	I_GTX_RX_P(3) I_GTX_RX_N(3) O_GTX_TX_P(3) O_GTX_TX_N(3)	Connected to GPAC's BPM_4MGT_*(1) signal. Not used.
GTX BPM channel 2	I_GTX_RX_P(4) I_GTX_RX_N(4) O_GTX_TX_P(4) O_GTX_TX_N(4)	Connected to GPAC's BPM_4MGT_*(2) signal. Not used.
GTX BPM channel 3	I_GTX_RX_P(5) I_GTX_RX_N(5) O_GTX_TX_P(5) O_GTX_TX_N(5)	Connected to GPAC's BPM_4MGT_*(3) signal. Not used.

2.3 PLB Register map

The PLB register map is shown in Table 4 below. Each register is 32-bit wide. Addressing is bytewise. Parameters are read-write unless specified as read-only.

Table 4 – PLB register map

PLB Address offset	Bit range	Field name	Description
0x0	0	RST_P0.0	Reset backplane GTX link, channel 0 (P0.0).
0x0	1	RST_P0.1	Reset backplane GTX link, channel 0 (P0.1).
0x0	2	RST_BPM0	Reset cross-FPGA GTX link, channel 0 (BPM.0).
0x0	3	RST_BPM1	Reset cross-FPGA GTX link, channel 1 (BPM.1).
0x0	4	RST_BPM2	Reset cross-FPGA GTX link, channel 2 (BPM.2).
0x0	5	RST_BPM3	Reset cross-FPGA GTX link, channel 3 (BPM.3).
0x0	6	RST_PKT_FIFO_WERR	Reset packet-FIFO write error counter.
0x0	7	RST_PKT_FIFO_RERR	Reset packet-FIFO read error counter.
0x0	8	RST_ROUTER_ERR0	Reset packet router's output 0 error counter.
0x0	9	RST_ROUTER_ERR1	Reset packet router's output 1 error counter.
0x4	2:0	LOOPBACK_P0.0	Loopback setting for GTX backplane channel 0 (P0.0). See [3] chapter 9 for details.
0x4	6:4	LOOPBACK_P0.1	Loopback setting for GTX backplane channel 1 (P0.1). See [3] chapter 9 for details.
0x4	10:8	LOOPBACK_BPM0	Loopback setting for GTX cross-FPGA channel 0 (BPM0). See [3] chapter 9 for details.
0x4	14:12	LOOPBACK_BPM1	Loopback setting for GTX cross-FPGA channel 1 (BPM1). See [3] chapter 9 for details.

PLB Address offset	Bit range	Field name	Description
0x4	18:16	LOOPBACK_BPM2	Loopback setting for GTX cross-FPGA channel 2 (BPM2). See [3] chapter 9 for details.
0x4	22:20	LOOPBACK_BPM3	Loopback setting for GTX cross-FPGA channel 3 (BPM3). See [3] chapter 9 for details.
0x4	31:24	BPM_ID	BPM identifier. Shall be unique in the system. This value is written into data packets sent from current BPM.
0x8	0	LOCK_P0	PLL lock for backplane GTX tile (both channels). See [3] for details. Read-only.
0x8	9	RST_DONE_P0.0	Reset done for backplane GTX link, channels 0 (P0.0). See [3] for details. Read-only.
0x8	8	RST_DONE_P0.1	Reset done for backplane GTX link, channels 1 (P0.0). See [3] for details. Read-only.
0x8	5:4	LOS_P0.0	Status of Loss-of-sync FSM for backplane GTX link, channel 0. See [3], chapter 7 for details. Read-only.
0x8	7:6	LOS_P0.1	Status of Loss-of-sync FSM for backplane GTX link, channel 1. See [3], chapter 7 for details. Read-only.
0x8	8	LOCK_BPM01	PLL lock for cross-FPGA GTX tile (channels 0 and 1). See [3] for details. Read-only.
0x8	9	RST_DONE_BPM0	Reset done for cross-FPGA GTX link, channel 0 (BPM0). See [3] for details. Read-only.
0x8	10	RST_DONE_BPM1	Reset done for cross-FPGA GTX link, channel 1 (BPM1). See [3] for details. Read-only.
0x8	13:12	LOS_BPM0	Status of Loss-of-sync FSM for cross-FPGA GTX link, channel 0 (BPM0). See [3], chapter 7 for details. Read-only.
0x8	15:14	LOS_BPM1	Status of Loss-of-sync FSM for cross-FPGA GTX link, channel 1 (BPM1). See [3], chapter 7 for details. Read-only.
0x8	16	LOCK_BPM23	PLL lock for cross-FPGA GTX tile (channels 2 and 3). See [3] for details. Read-only.
0x8	17	RST_DONE_BPM2	Reset done for cross-FPGA GTX link, channel 2 (BPM2). See [3] for details. Read-only.
0x8	18	RST_DONE_BPM3	Reset done for cross-FPGA GTX link, channel 3 (BPM3). See [3] for details. Read-only.
0x8	21:20	LOS_BPM2	Status of Loss-of-sync FSM for cross-FPGA GTX link, channel 2 (BPM2). See [3], chapter 7 for details. Read-only.
0x8	23:22	LOS_BPM3	Status of Loss-of-sync FSM for cross-FPGA GTX link, channel 3 (BPM3). See [3], chapter 7 for details. Read-only.
0x8	24	RX_SYNC_P0.0	Receiver synchronization status for backplane GTX link, channel 0 (P0.0). When ‘1’ the RX channel is correctly synchronized. Read-only.
0x8	25	RX_SYNC_P0.1	Receiver synchronization status for backplane GTX link, channel 1 (P0.1). When ‘1’ the RX channel is correctly synchronized. Read-only.
0x8	26	RX_SYNC_BPM0	Receiver synchronization status for cross-FPGA GTX link, channel 0 (BPM0). When ‘1’ the RX channel is correctly synchronized. Read-only.
0x8	27	RX_SYNC_BPM1	Receiver synchronization status for cross-FPGA GTX link, channel 1 (BPM1). When ‘1’ the RX channel is correctly synchronized. Read-only.
0x8	28	RX_SYNC_BPM2	Receiver synchronization status for cross-FPGA GTX link, channel 2 (BPM2). When ‘1’ the RX channel is correctly synchronized. Read-only.

PLB Address offset	Bit range	Field name	Description
0x8	29	RX_SYNC_BPM3	Receiver synchronization status for cross-FPGA GTX link, channel 3 (BPM3). When '1' the RX channel is correctly synchronized. Read-only.
0xC	15:0	LOS_COUNTER_P0.0	Loss-of-sync counter for backplane GTX channel 0 (P0.0). The counter can be reset by resetting the GTX channel. Read-only.
0xC	31:16	LOS_COUNTER_P0.1	Loss-of-sync counter for backplane GTX channel 1 (P0.0). The counter can be reset by resetting the GTX channel. Read-only.
0x10	15:0	LOS_COUNTER_BPM0	Loss-of-sync counter for cross-FPGA GTX channel 0 (BPM0). The counter can be reset by resetting the GTX channel. Read-only.
0x10	31:16	LOS_COUNTER_BPM1	Loss-of-sync counter for cross-FPGA GTX channel 1 (BPM1). The counter can be reset by resetting the GTX channel. Read-only.
0x14	15:0	LOS_COUNTER_BPM2	Loss-of-sync counter for cross-FPGA GTX channel 2 (BPM2). The counter can be reset by resetting the GTX channel. Read-only.
0x14	31:16	LOS_COUNTER_BPM3	Loss-of-sync counter for cross-FPGA GTX channel 3 (BPM3). The counter can be reset by resetting the GTX channel. Read-only.
0x18	7:0	K_EOP	End-of-packet K-character used in IBFB protocol. Read-only.
0x18	15:8	K_SOP	Start-of-packet K-character used in IBFB protocol. Read-only.
0x1C	15:0	PKT_FIFO_WERR_CNT	Packet FIFO write error counter. Incremented whenever an attempt is made to write the packet FIFO while full. Read-only.
0x1C	31:16	PKT_FIFO_RERR_CNT	Packet FIFO read error counter. Incremented whenever an attempt is made to read the packet FIFO while empty. Read-only.
0x20	15:0	ROUT_ERR_CNT0	Router output error counter 0. Incremented whenever an attempt is made to write on the backplane GTX link 0 (P0.0) while its TX FIFO is full. Read-only.
0x20	31:16	ROUT_ERR_CNT1	Router output error counter 1. Incremented whenever an attempt is made to write on the cross-FPGA GTX link 0 (BPM0) while its TX FIFO is full. Read-only.
0x7C	31_0	FW_VERSION	Component's version number (unsigned integer, incremental).

2.4 Functional description

The core has two main purposes: the first is to generate packets containing data from the locally connected BPM processing logic, and inject the data into the daisy chain upstream and downstream rings. The second is to implement the daisy chain node by properly forwarding the packets traveling through the two chain rings.
The main components in the core are the following (as seen in Figure 3):

- **Input interface and CDC FIFO:** this component collects data from the BPM signal processing block and writes it to the input FIFO along with additional data needed to compose the IBFB packet. The bucket number is also generated here.

- **IBFB Packet Transmitter:** this component formats the BPM data into packets according to the IBFB protocol, and writes the packets to FIFO. Packet CRC is computed here.
- **Packet router:** this component injects the new BPM packets into the daisy chain rings (upstream and downstream). Other packets coming from other BPMs are routed to the appropriate destination.

2.4.1 Input interface and CDC FIFO

The **ibfb_bpm_router** component is directly connected to the **bpm_cav_exfel** component. The latter provides data using a custom interface, as shown in Figure 5. The whole interface is synchronous to the ADC clock (**o_adc_clk** signal, not shown in figure).

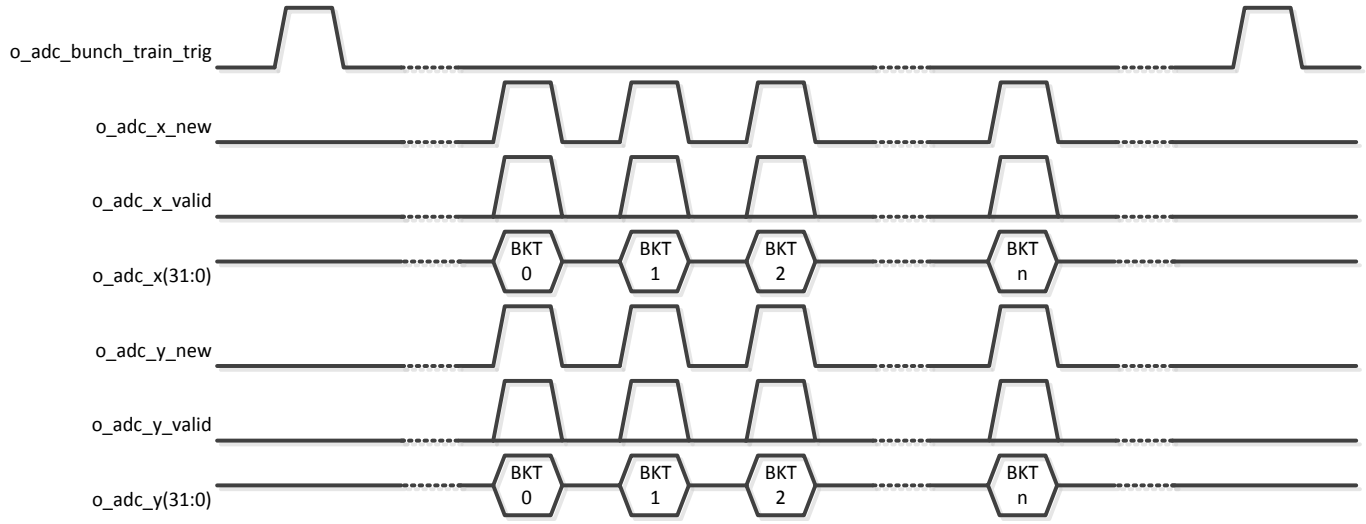

Figure 5 - Input interface signals

Each bunch starts with a pulse of the **o_bunch_train_trig** signal. This causes the bucket counter to reset. After that, data for the X and Y position is sent over two separate buses, one value for every bucket. The **o_adc_x/y_new** signal signals that data can be sampled. The signal **o_adx_x/y** carries the data itself. The signal **o_adc_x/y_valid** notifies whether the data is usable or not.
Data from each of the two buses is written to a separate 36 bit FIFO (32 bits of data plus 1 bit for the valid flag). Whenever a bunch start trigger is received, a special word is written to both FIFOs. The write side of the FIFO is synchronous to the **o_adc_clk** signal.
The data written to the FIFOs is formatted as shown in Table 5.

Table 5 - Format of data written to INPUT FIFO

35 34	33	32	31 0
SPARE(2)	**TRIG**	**VALID**	**DATA(32)**

A separate component (**INFIFO_READ**) reads data from the two input FIFOs and feeds it to the IBFB packet transmitter. The data is considered valid only when the **TRIG** flag is not set. The INFIFO_READ component generates also the bucket number by incrementing a counter (bucket counter) every time a **o_adc_x/y_new** pulse is received. The counter is reset whenever the bit **TRIG** is set in the data word read from FIFO. An additional input of the INFIFO_READ

component is the BPM number. This number is constant for every BPM and set using a PLB register. The data collected from the INFIFO_READ component is then sent to the IBFB_PKT_TX component. The input interface component's architecture is shown in Figure 6

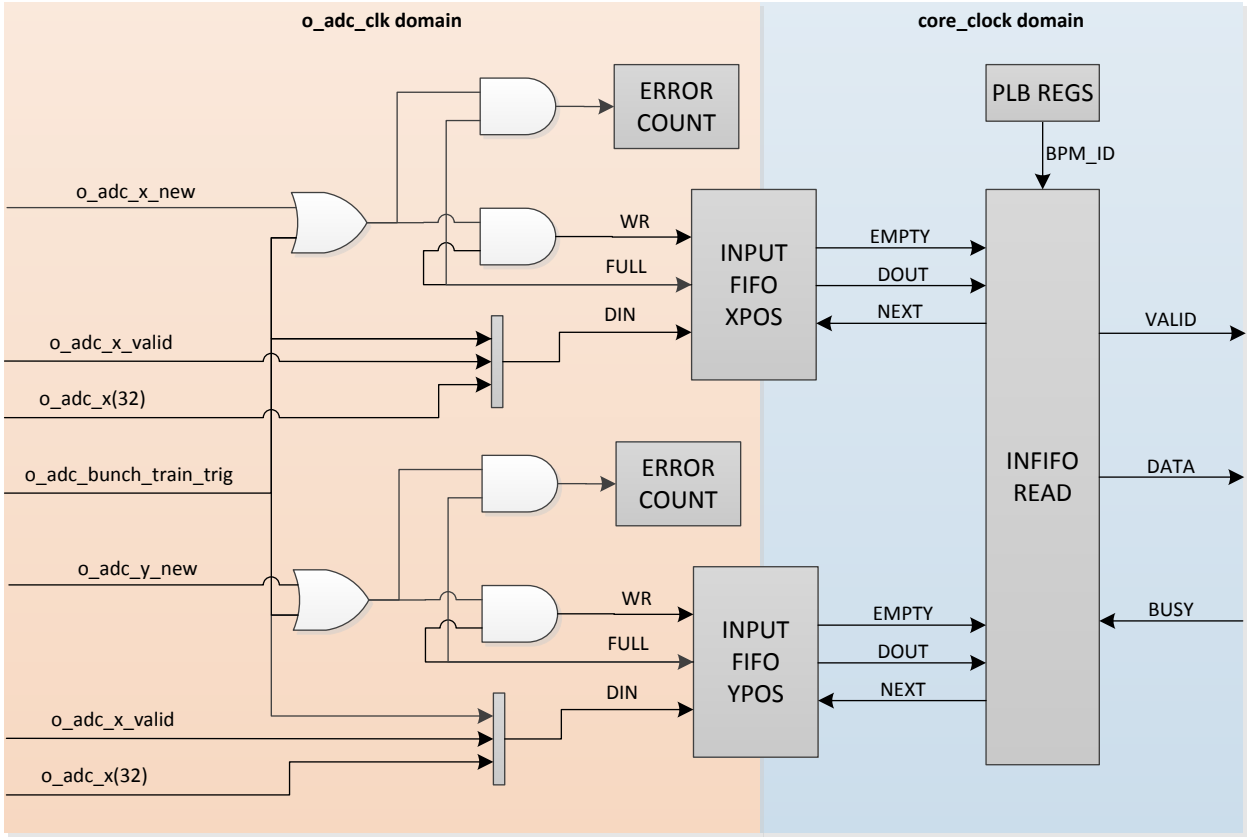

Figure 6 - Architecture of the Input Interface component

2.4.2 IBFB Packet Transmitter

The **IBFB_PKT_TX** component generates the BPM packets according to the IBFB protocol. Each packet contains data related to a specific bucket from a specific BPM. The packet format is shown in Table 6. Each packet is 16 Bytes long, split into four 32-bit words to match the GTX channel's FIFO width. Packet fields are described in Table 7.

Table 6 - IBFB Packet Format

Dword 0				Dword 1				Dword 2				Dword 3			
0x00	SOP	CTRL	BPM_ID	BKT0	BKT1	X0	X1	X2	X3	Y0	Y1	Y2	Y3	CRC	EOP

Table 7 - IBFB packet fields' description

Field	Format	Description
SOP	8 bit K-character	Start of packet byte. Must be a valid 8b10b K-character. Currently set to 0xFB (K27.7)
CTRL	8 bit mask	Bit 0 = Y position valid Bit 1 = X position valid Bits 7:2 = currently not used Value 0xFF is reserved for the IBFB Player component (it is never generated by the IBFB_PKT_TX component).
BPM_ID	8 bit unsigned integer	BPM identification number. Shall be unique for every BPM in the system. It is set through a PLB register.
BKT	16 bit unsigned integer	Bucket number. Identifies a single position reading in a bunch. Buckets are numbered sequentially starting from 0 for the first reading after the bunch trigger. The value is sent Least Significant Byte first.
X	32 bit single precision floating point	X position reading from the BPM. The value is sent Least Significant Byte first.
Y	32 bit single precision floating point	Y position reading from the BPM. The value is sent Least Significant Byte first.
CRC	8 bit unsigned integer	CCITT CRC8 ($1+x+x^2+x^8$) calculated on the four 32-bit words that compose the packet (cfr. Table 6). For CRC calculation the CRC field in the packet is set to 0x00.
EOP	8 bit K-character	End of packet byte. Must be a valid 8b10b K-character. Currently set to 0xFD (K29.7)

The IBFB_PKT_TX component samples input data whenever the input valid signal is high. Then it needs the following 5 clock cycles to send a whole packet. During this period the component won't accept further input data. This means that the component is able to handle a maximum throughput of 1 packet every 6 clock cycles. The output data is written to the BPM Packet FIFO. The component's timing diagram is shown in Figure 7.

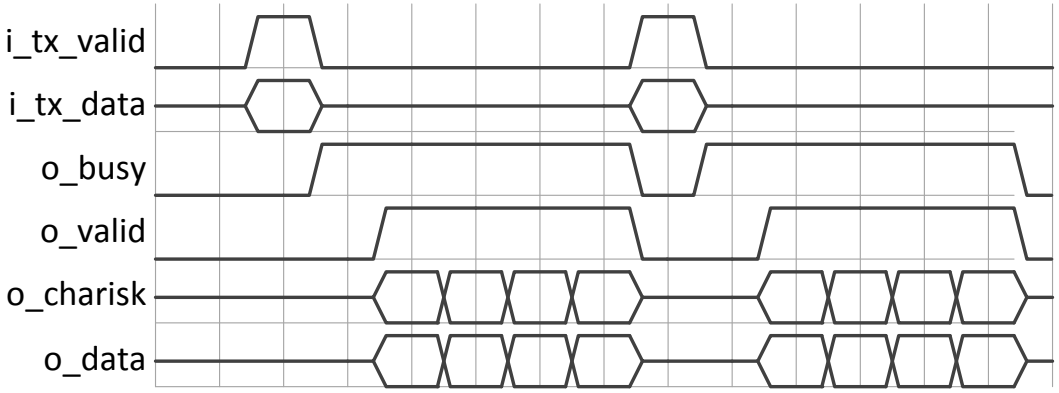

Figure 7 - IBFB_PKT_TX timing diagram

2.4.3 Packet Router

The Packet Router is in charge of routing all the input IBFB BPM packets to the proper destination. It reads data from three FIFOs (Local BPM, backplane link, cross-FPGA link) and writes to two FIFOs (backplane link, cross-FPGA link). The packet routing is performed according to the algorithm described in Figure 8.

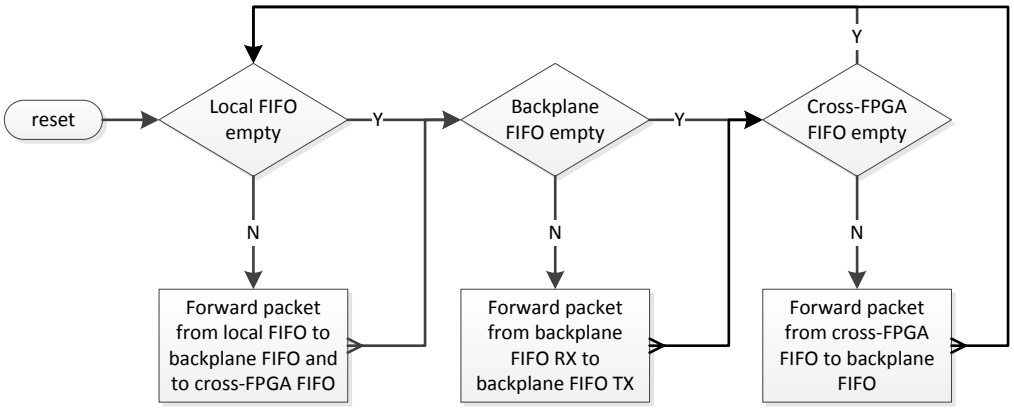

Figure 8 - Packet Router's polling algorithm

For this purpose the component **ibfb_packet_router** is used. This component allows to read round robin from a set of input FIFOs and route them to a set of output FIFOs. The number of input and output channels can be set via parameter. The routing rules are set by means of a routing table. The routing table allows to select for each input channel, to which output channels the packets should be forwarded. In this case the component is configured with 3 input channels and 2 output channels. The routing table is set up as described in Table 8: rows represent input channels while columns represent output channels. A '1' at the intersection means that every packet coming from the input channel is forwarded to the output channel.

Table 8 - Packet router's routing table

		Output Channels	
		xFPGA link TX (OUT_0)	**Backplane TX link (OUT_1)**
Input Channels	**Local BPM(IN_0)**	1	1
	xFPGA RX link (IN_1)	0	1
	Backplane RX link (IN_2)	0	1

The architecture of the **ibfb_packet_router** is shown in Figure 9. Each input channel is connected to an input buffer component (ibfb_packet_buffer). This component attempts to buffer a whole packet whenever data is available in the input FIFO. If a whole packet can be buffered, the component waits until the packet is read from the following component in the pipeline. The component checks that the packet's structure is compliant with the format described in §2.4.2 (CRC is not checked). Data that is not compliant with the expected packet structure is discarded.
The *Input port selector* polls the input buffers round-robin. Whenever a packet is found, it is forwarded to the proper channel according to the routing table content.

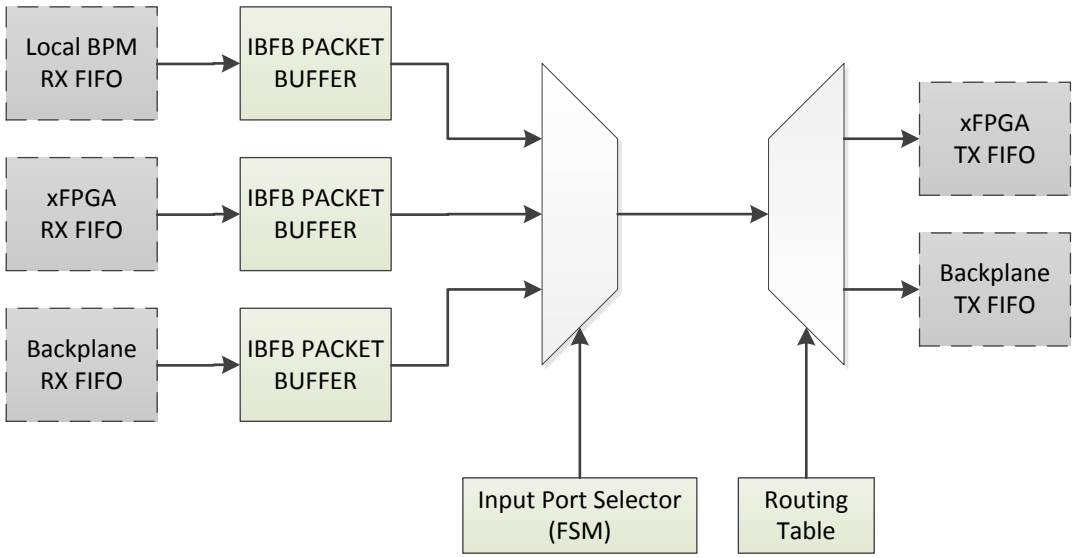

Figure 9 - Packet router's architecture

2.5 FPGA Resources

Table 9 – Resource utilization

Version	Flip Flops	LUTs	LUTRAMs	BRAMs	GTX
Virtex-5	1268	1155	32	11	3

2.6 Design constraints

The core has no specific constraints.