IBFB switch

`ibfb_switch`
`Revision 2.1`

Firmware Data Sheet

PSI, 21.08.2016

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.

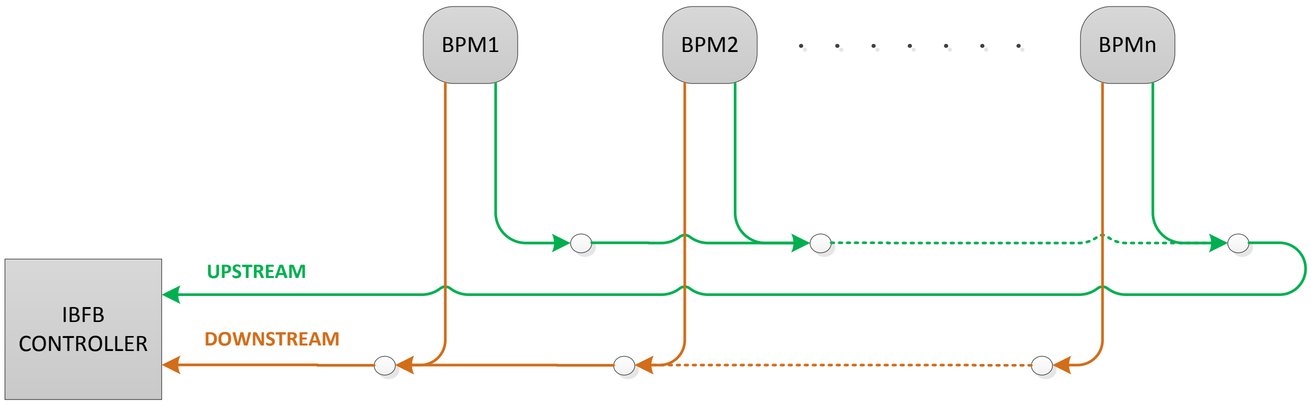

Figure 1 - Data link between BPMs (simplified)

Data from each BPM is formatted in packets according to a custom protocol (see §TBD for details). Given the topology, the same data packet arrives two times to the IBFB controller: one time from the upstream chain and one from the downstream chain. Which packet arrives first depends on the BPM's physical position in the chain. The *ibfb_switch* component receives data from up to 2 BPM chains (1 upstream + 1 downstream), filters out the duplicated packets, and forwards the output data to the backplane connection and optionally to another switch component (through cross-FPGA links). The *ibfb_switch* also receives and routes packets coming from another switch component, in order to allow cascaded configurations.

1.1 Features

The *ibfb_switch* component has the following features:

- PLB register interface for command and control (tested up to 125 MHz PLB clock frequency on Virtex-5).
- Provides up to 4 Input GTX channels. Input channels are logically grouped into pairs. Each pair can be configured either as an upstream+downstream pair (filtering duplicated packets) or used as a single independent input. Each Input channel uses a GTX serial transceiver link to receive data.
- Provides two full duplex GTX side channels (one for X-position and one for Y-position packets) to allow cascading of multiple IBFB switches. Data received from the side channel is forwarded together with the data coming from the Chain-Input channels.
- Output data can be sent either to the backplane channels or to the side channel or to both by enabling/disabling the relative outputs via control register.
- Tested up to 3.125 Gbps GTX link speed on Virtex-5.

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
GTX	Xilinx X-class Gigabit Transceiver

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] PDC QSFP, Quad SFP Module, *Schematic*, Revision 0.11, 2.7.2013
[3] Virtex-5 FPGA RocketIO GTX Transceiver, User Guide, UG198, V3.0, 30.10.2009

1.4 History

Revision	Date	Author	Description
1.0	18.07.2016	A. Malatesta	First release
2.0	29.08.2016	A. Malatesta	Updated with details on new dual-router architecture
2.1	21.08.2016	A. Malatesta	Added section on PING feature

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 three GTX_DUAL tiles. The component is wrapped in a Xilinx EDK pcore with slave PLB interface, but its core logic (packet router and filters) is also available as independent VHDL components.

2.1 Architecture

The core is explicitly developed to be instantiated in the GPAC's BPM FPGA (cfr. [1]). Data can be received from 6 different GTX channels, 4 of which connected to a QSPF piggyback (cfr [2]) and the 2 to one of the cross-FPGA GTX tiles.

Received data is split into two separate data streams: one containing only packets carrying X-position information, and one for Y-position. Each data stream is collected by a Packet Router component (X-router and Y-router), and forwarded to one or more output channels.

QSFP inputs are configured as pairs connected to a filter that discards packet duplicates within the same bunch train. By connecting only one of the two inputs and leaving the other unused, an independent data stream can be received without any filtering. Each filter provides two outputs: one for the X-position packets and one for the Y-position packets.

Data output can be configured by enabling/disabling the outputs of the two packet routers. Each router collects data from both filters and from one of the cross-FPGA receive channels. Routers have two outputs: one connected to a backplane TX link (P0) and one to one cross-FPGA TX link. Each output can be enabled independently via control register.

A PLB register interface allows to access configuration parameters and status information.

A timing component generates an internal trigger for the packet filters. This trigger is generated from an external one. An arbitrary trigger delay can be set via PLB interface.

Figure 2 shows how the switch is used in the IBFB system: data is received from three upstream+downstream chains plus a single channel from a collimator BPM. This architecture in implemented in a single GPAC board using both BPM FPGAs, each equipped with a QSFP mezzanine card.

The first FPGA (on the left in Figure 2) contains an *ibfb_switch* component with the QSFP inputs used as two pairs, and the routers with only the backplane outputs enabled. An additional data stream comes from the outputs of the *ibfb_switch* instantiated in the second FPGA, and it is also forwarded to the backplane links.

The second FPGA (on the right in Figure 2) contains an *ibfb_switch* component with two QSFP links configured as a pair, and only one the two remaining channels used as an independent input. The routers have only the cross-FPGA output enabled, so that the data is forwarded to the first FPGA trough the cross-FPGA links.

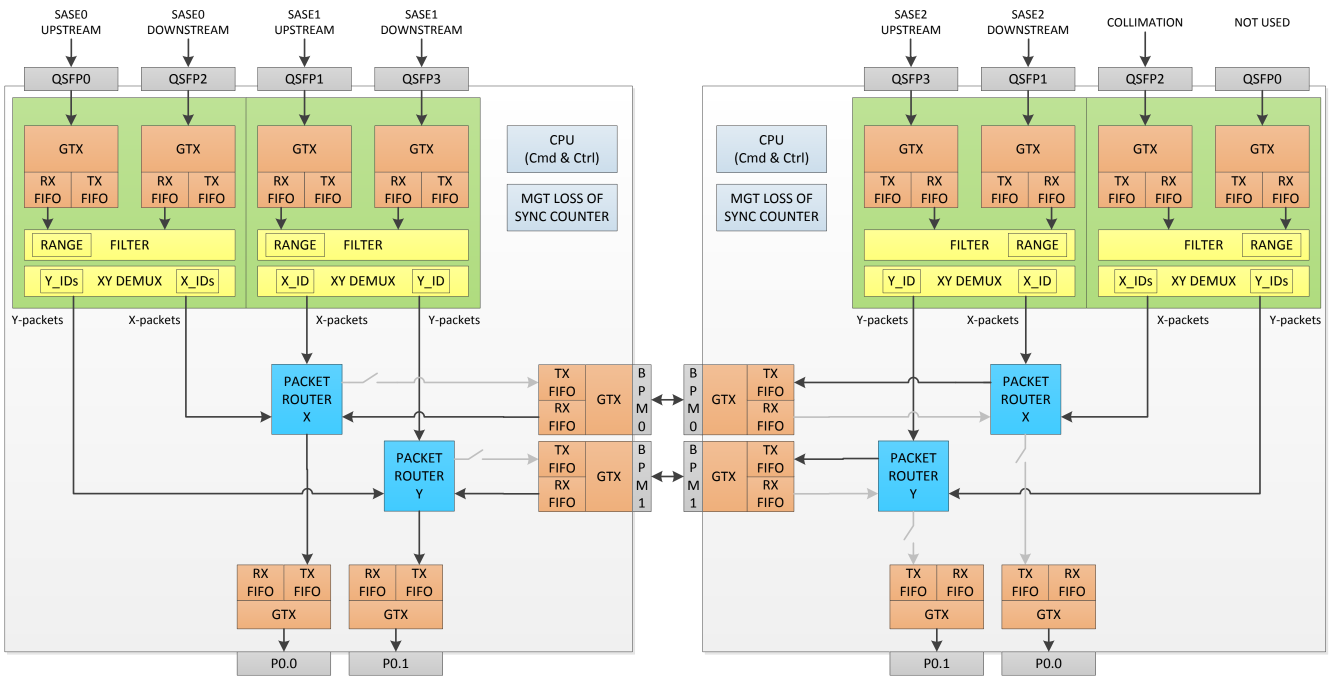

Figure 2 – System-level architectural view containing two ibfb_switch cores on different FPGAs.

2.2 Ports and Attributes

Figure 3 contains component overview with input and output ports. Table 1 and Table 2 describe the meaning of port and attributes.

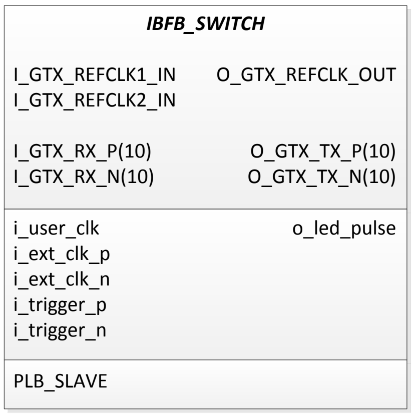

Figure 3 - Top entity of the ibfb_switch.

Table 1 - Port description for ibfb_switch

Port name	Dir.	Type	Description
I_GTX_REFCLK1_IN	in	std_logic	First reference clock for the GTX transceivers. Any of the five 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 five 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(9:0)	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(9:0)	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(9:0)	Positive transmit wires for the GTX physical layer (see Table 3 for GTX channel assignment).
O_GTX_TX_P	out	std_logic_vector(9:0)	Negative transmit wires for the GTX physical layer (see Table 3 for GTX channel assignment).
i_user_clk	in	std_logic	Main core clock.
i_ext_clk_(p/n)	in	std_logic (differential pair)	External clock. Used to drive the *i_adc_clk* input of the timing component.
i_trigger_(p/n)	in	std_logic (differential pair)	External trigger. Fed to the timing component, and used to generate the start trigger for all the players.
o_led_pulse	out	std_logic	From the timing component. Meant to be connected to a LED in order to monitor trigger's activity.
PLB_SLAVE	in	BUS	PLB slave bus (bursting) used to access local register bank for command and control purposes, and the players' RAMs. PLB bus clock is used to drive only the register bank logic and the RAM interface. The core itself runs with the clock specified by the C_USE_EXTERNAL_CLOCK parameter.

Table 2 - Attribute description for ibfb_switch

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 a 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 a K-character accepted by Xilinx GTX transceivers 8b/10b logic (see [3]).
C_GTX_REFCLK_SEL	std_logic_vector(4:0)	Define which reference clock is used by every GTX slice. Each bit represent a slice: • Bit 4: cross-FPGA channels 2 and 3 • Bit 3: cross-FPGA channels 0 and 1 • Bit 2: P0 channels 0 and 1. • Bit 1: QSFP channels 2 and 3 • Bit 0: QSFP channels 0 and 1 Value 0 selects I_GTX_REFCLK1_IN, while value 1 selects I_GTX_REFCLK2_IN. According to which physical transceivers are used, restrictions apply (see [3] for details). For configuration described in Table 3, the value MUST be “01100”.
C_SFP02_REFCLK_FREQ	natural	Reference clock frequency for QSFP channels 0 and 2 (in Megahertz)
C_SFP13_REFCLK_FREQ	natural	Reference clock frequency for QSFP channels 1 and 3 (in Megahertz)
C_P0_REFCLK_FREQ	natural	Reference clock frequency for P0 channels 0 and 1 (in Megahertz)
C_BPM_REFCLK_FREQ	natural	Reference clock frequency for cross-FPGA channels (in Megahertz)
C_SFP02_BAUD_RATE	natural	Baud rate for SFP channels 0 and 2 (in Kilobits per second)
C_SFP13_BAUD_RATE	natural	Baud rate for SFP channels 1 and 3 (in Kilobits per second)
C_P0_BAUD_RATE	natural	Baud rate for P0 channels 0 and 1 (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 SFP 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 PB_8MGT_*(4) signal.
		Used only to receive data.

GTX SFP 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 PB_8MGT_*(6) signal.
		Used only to receive data.

GTX SFP Channel 2	I_GTX_RX_P(2)
	I_GTX_RX_N(2)
	O_GTX_TX_P(2)
	O_GTX_TX_N(2)
		Connected to GPAC’s PB_8MGT_*(5) signal.
		Used only to receive data.

GTX SFP Channel 3	I_GTX_RX_P(3)
	I_GTX_RX_N(3)
	O_GTX_TX_P(3)
	O_GTX_TX_N(3)
		Connected to GPAC’s PB_8MGT_*(7) signal.
		Used only to receive data.

GTX P0 Channel 0	I_GTX_RX_P(4)
	I_GTX_RX_N(4)
		Connected to GPAC’s P0_2MGT_*(1) signal.
		Transmit only (X-position packets).

	O_GTX_TX_P(4)
	O_GTX_TX_N(4)	

GTX P0 channel 1	I_GTX_RX_P(5)
	I_GTX_RX_N(5)
	O_GTX_TX_P(5)
	O_GTX_TX_N(5)
		Connected to GPAC's P0_2MGT_*(0) signal.
		Transmit only (Y-position packets).

GTX BPM channel 0	I_GTX_RX_P(6)
	I_GTX_RX_N(6)
	O_GTX_TX_P(6)
	O_GTX_TX_N(6)
		Connected to GPAC's BPM_4MGT_*(0) signal.
		Used to transmit and receive X-position packets (data to/from *ibfb_switch* component on another FPGA).

GTX BPM channel 1	I_GTX_RX_P(7)
	I_GTX_RX_N(7)
	O_GTX_TX_P(7)
	O_GTX_TX_N(7)
		Connected to GPAC's BPM_4MGT_*(1) signal.
		Used to transmit and receive Y-position packets (data to/from *ibfb_switch* component on another FPGA).

GTX BPM channel 2	I_GTX_RX_P(8)
	I_GTX_RX_N(8)
	O_GTX_TX_P(8)
	O_GTX_TX_N(8)
		Connected to GPAC's BPM_4MGT_*(2) signal.
		Not used.

GTX BPM channel 3	I_GTX_RX_P(9)
	I_GTX_RX_N(9)
	O_GTX_TX_P(9)
	O_GTX_TX_N(9)
		Connected to GPAC's BPM_4MGT_*(3) signal.
		Not used.

2.3 PLB Interface

2.3.1 Registers

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_QSFP0	Reset QSFP channel 0 GTX FIFOs.
0x0	1	RST_QSFP1	Reset QSFP channel 1 GTX FIFOs.
0x0	2	RST_QSFP2	Reset QSFP channel 2 GTX FIFOs.
0x0	3	RST_QSFP3	Reset QSFP channel 3 GTX FIFOs.
0x0	4	RST_P0.0	Reset P0 channel 0 GTX FIFOs.
0x0	5	RST_P0.1	Reset P0 channel 1 GTX FIFOs.
0x0	6	RST_BPM0	Reset BPM channel 0 GTX FIFOs.
0x0	7	RST_BPM1	Reset BPM channel 1 GTX FIFOs.
0x0	8	RST_BPM2	Reset BPM channel 2 GTX FIFOs.
0x0	9	RST_BPM3	Reset BPM channel 3 GTX FIFOs.
0x0	10	RST_FILT13	Reset packet filter for QSFP channel 1 and 3
0x0	11	RST_FILT02	Reset packet filter for QSFP channel 0 and 2
0x0	12	RST_ROUTER	Reset packet routers (both X and Y)
0x0	13	RST_ROUTER_ERR	Reset packet routers' error counters (both X and Y)
0x0	16	TRIG_FILT13	Send trigger to packet filter for QSFP channels 1 and 3 (needs a 0->1 transition)
0x0	17	TRIG_FILT02	Send trigger to packet filter for QSFP channels 0 and 2 (needs a 0->1 transition)
0x0	20	RST_LOS_CNT_QSFP0	Reset loss of sync counter for QSFP0 GTX link.
0x0	21	RST_LOS_CNT_QSFP1	Reset loss of sync counter for QSFP1 GTX link.

PLB Address offset	Bit range	Field name	Description
0x0	22	RST_LOS_CNT_QSFP2	Reset loss of sync counter for QSFP2 GTX link.
0x0	23	RST_LOS_CNT_QSFP3	Reset loss of sync counter for QSFP3 GTX link.
0x0	24	RST_LOS_CNT_P0.0	Reset loss of sync counter for P0.0 GTX link.
0x0	25	RST_LOS_CNT_P0.1	Reset loss of sync counter for P0.1 GTX link.
0x0	26	RST_LOS_CNT_BPM0	Reset loss of sync counter for BPM0 GTX link.
0x0	27	RST_LOS_CNT_BPM1	Reset loss of sync counter for BPM1 GTX link.
0x0	28	RST_LOS_CNT_BPM2	Reset loss of sync counter for BPM2 GTX link.
0x0	29	RST_LOS_CNT_BPM3	Reset loss of sync counter for BPM3 GTX link.
0x0	31	CRC_ERR_CNT_RST	Reset CRC error counter for output packets.
0x4	2:0	LOOPBACK_QSFP0	Loopback setting for QSFP0 GTX channel. See [3] chapter 9 for details.
0x4	6:4	LOOPBACK_QSFP1	Loopback setting for QSFP1 GTX channel. See [3] chapter 9 for details.
0x4	10:8	LOOPBACK_QSFP2	Loopback setting for QSFP2 GTX channel. See [3] chapter 9 for details.
0x4	14:12	LOOPBACK_QSFP3	Loopback setting for QSFP3 GTX channel. See [3] chapter 9 for details.
0x4	18:16	LOOPBACK_BPM0	Loopback setting for BPM0 GTX channel. See [3] chapter 9 for details.
0x4	22:20	LOOPBACK_BPM1	Loopback setting for BPM1 GTX channel. See [3] chapter 9 for details.
0x4	26:24	LOOPBACK_BPM2	Loopback setting for BPM2 GTX channel. See [3] chapter 9 for details.
0x4	30:28	LOOPBACK_BPM3	Loopback setting for BPM3 GTX channel. See [3] chapter 9 for details.
0x8	2:0	LOOPBACK_P0.0	Loopback setting for P0.0 GTX channel. See [3] chapter 9 for details.
0x8	6:4	LOOPBACK_P0.1	Loopback setting for P0.1 GTX channel. See [3] chapter 9 for details.
0x8	28	BPM1_OUT_EN (Y)	Enable Y-router output to cross-FPGA link (other ibfb_switch component on a second FPGA)
0x8	29	P0.1_OUT_EN (Y)	Enable Y-router output to backplane link (P0 connector, channel 0).
0x8	30	BPM0_OUT_EN (X)	Enable X-router output to cross-FPGA link (other ibfb_switch component on a second FPGA)
0x8	31	P0.0_OUT_EN (X)	Enable X-router output to backplane link (P0 connector, channel 1).
0xC	0	LOCK_QSFP13	PLL lock for QSFP GTX tile (channels 1 and 3). See [3] for details. Read-only.
0xC	1	RST_DONE_QSFP3	Reset done for QSFP3 GTX link. See [3] for details. Read-only.
0xC	2	RST_DONE_QSFP1	Reset done for QSFP1 GTX link. See [3] for details. Read-only.
0xC	5:4	LOS_QSFP3	Status of Loss-of-sync FSM for QSFP3 GTX link. See [3], chapter 7 for details. Read-only.
0xC	7:6	LOS_QSFP1	Status of Loss-of-sync FSM for QSFP1 GTX link. See [3], chapter 7 for details. Read-only.
0xC	8	LOCK_QSFP02	PLL lock for QSFP GTX tile (channels 0 and 2). See [3] for details. Read-only.
0xC	9	RST_DONE_QSFP2	Reset done for QSFP2 GTX link. See [3] for details. Read-only.
0xC	10	RST_DONE_QSFP0	Reset done for QSFP0 GTX link. See [3] for details. Read-only.
0xC	13:12	LOS_QSFP2	Status of Loss-of-sync FSM for QSFP2 GTX link. See [3], chapter 7 for details. Read-only.

PLB Address offset	Bit range	Field name	Description
0xC	15:14	LOS_QSFP0	Status of Loss-of-sync FSM for QSFP0 GTX link. See [3], chapter 7 for details. Read-only.
0xC	16	LOCK_BPM01	PLL lock for BPM GTX tile (channels 0 and 1). See [3] for details. Read-only.
0xC	17	RST_DONE_BPM0	Reset done for BPM0 GTX link. See [3] for details. Read-only.
0xC	18	RST_DONE_BPM1	Reset done for BPM1 GTX link. See [3] for details. Read-only.
0xC	21:20	LOS_BPM0	Status of Loss-of-sync FSM for BPM0 GTX link. See [3], chapter 7 for details. Read-only.
0xC	23:22	LOS_BPM1	Status of Loss-of-sync FSM for BPM1 GTX link. See [3], chapter 7 for details. Read-only.
0xC	24	LOCK_BPM23	PLL lock for BPM GTX tile (channels 2 and 3). See [3] for details. Read-only.
0xC	25	RST_DONE_BPM2	Reset done for BPM2 GTX link. See [3] for details. Read-only.
0xC	26	RST_DONE_BPM3	Reset done for BPM3 GTX link. See [3] for details. Read-only.
0xC	29:28	LOS_BPM2	Status of Loss-of-sync FSM for BPM2 GTX link. See [3], chapter 7 for details. Read-only.
0xC	31:30	LOS_BPM3	Status of Loss-of-sync FSM for BPM3 GTX link. See [3], chapter 7 for details. Read-only.
0x10	0	LOCK_P0	PLL lock for P0 GTX tile (both channels). See [3] for details. Read-only.
0x10	9	RST_DONE_P0.0	Reset done for P0.0 GTX link. See [3] for details. Read-only.
0x10	8	RST_DONE_P0.1	Reset done for P0.1 GTX link. See [3] for details. Read-only.
0x10	5:4	LOS_P0.0	Status of Loss-of-sync FSM for P0.0 GTX link. See [3], chapter 7 for details. Read-only.
0x10	7:6	LOS_P0.1	Status of Loss-of-sync FSM for P0.1 GTX link. See [3], chapter 7 for details. Read-only.
0x10	8	ERR_OUT1_YROUTER	Error on Y-router’s output to cross-FPGA link (BPM1 TX FIFO overflow). Read-only.
0x10	9	ERR_OUT0_YROUTER	Error on Y-router’s output to backplane (P0.0 TX FIFO overflow). Read-only.
0x10	10	ERR_OUT1_XROUTER	Error on X-router’s output to cross-FPGA link (BPM0 TX FIFO overflow). Read-only.
0x10	11	ERR_OUT0_XROUTER	Error on X-router’s output to backplane (P0.1 TX FIFO overflow). Read-only.
0x10	23:16	K_EOP	End-of-packet K-character used in IBFB protocol. Read-only.
0x10	31:24	K_SOP	Start-of-packet K-character used in IBFB protocol. Read-only.
0x14	8	SFP0_RX_SYNC	Receiver synchronization status for QSFP0 GTX link. When ‘1’ the RX channel is correctly synchronized. Read-only.
0x14	9	SFP1_RX_SYNC	Receiver synchronization status for QSFP1 GTX link. When ‘1’ the RX channel is correctly synchronized. Read-only.
0x14	10	SFP2_RX_SYNC	Receiver synchronization status for QSFP2 GTX link. When ‘1’ the RX channel is correctly synchronized. Read-only.
0x14	11	SFP3_RX_SYNC	Receiver synchronization status for QSFP3 GTX link. When ‘1’ the RX channel is correctly synchronized. Read-only.

PLB Address offset	Bit range	Field name	Description
0x14	12	P00_RX_SYNC	Receiver synchronization status for P0.0 GTX link. When ‘1’ the RX channel is correctly synchronized. Read-only.
0x14	13	P01_RX_SYNC	Receiver synchronization status for P0.1 GTX link. When ‘1’ the RX channel is correctly synchronized. Read-only.
0x14	14	BPM0_RX_SYNC	Receiver synchronization status for BPM0 GTX link. When ‘1’ the RX channel is correctly synchronized. Read-only.
0x14	15	BPM1_RX_SYNC	Receiver synchronization status for BPM1 GTX link. When ‘1’ the RX channel is correctly synchronized. Read-only.
0x14	16	BPM2_RX_SYNC	Receiver synchronization status for BPM2 GTX link. When ‘1’ the RX channel is correctly synchronized. Read-only.
0x14	17	BPM3_RX_SYNC	Receiver synchronization status for BPM3 GTX link. When ‘1’ the RX channel is correctly synchronized. Read-only.
0x1C	7:0	FILT13_BPM_ID_0 (X)	Allowed BPM ID for X-position packets coming from SFP 1 and 3.
0x1C	15:8	FILT13_BPM_ID_1 (X)	Allowed BPM ID for X-position packets coming from SFP 1 and 3.
0x1C	23:16	FILT13_BPM_ID_2 (Y)	Allowed BPM ID for Y-position packets coming from SFP 1 and 3.
0x1C	31:24	FILT13_BPM_ID_3 (Y)	Allowed BPM ID for Y-position packets coming from SFP 1 and 3.
0x20	7:0	FILT02_BPM_ID_0 (X)	Allowed BPM ID for X-position packets coming from SFP 0 and 2.
0x20	15:8	FILT02_BPM_ID_1 (X)	Allowed BPM ID for X-position packets coming from SFP 0 and 2.
0x20	23:16	FILT02_BPM_ID_2 (Y)	Allowed BPM ID for Y-position packets coming from SFP 0 and 2.
0x20	31:24	FILT02_BPM_ID_3 (Y)	Allowed BPM ID for Y-position packets coming from SFP 0 and 2.
0x24	15:0	LOS_COUNTER_QSFP0	Loss-of-sync counter for QSFP0 GTX link. Can be reset with the RST_LOS_CNT_QSFP0 command. Read-only.
0x24	31:16	LOS_COUNTER_QSFP1	Loss-of-sync counter for QSFP1 GTX link. Can be reset with the RST_LOS_CNT_QSFP1 command. Read-only.
0x28	15:0	LOS_COUNTER_QSFP2	Loss-of-sync counter for QSFP2 GTX link. Can be reset with the RST_LOS_CNT_QSFP2 command. Read-only.
0x28	31:16	LOS_COUNTER_QSFP3	Loss-of-sync counter for QSFP3 GTX link. Can be reset with the RST_LOS_CNT_QSFP3 command. Read-only.
0x2C	15:0	LOS_COUNTER_BPM0	Loss-of-sync counter for BPM0 GTX link. Can be reset with the RST_LOS_CNT_BPM0 command. Read-only.
0x2C	31:16	LOS_COUNTER_BPM1	Loss-of-sync counter for BPM1 GTX link. Can be reset with the RST_LOS_CNT_BPM1 command. Read-only.
0x30	15:0	LOS_COUNTER_BPM2	Loss-of-sync counter for BPM2 GTX link. Can be reset with the RST_LOS_CNT_BPM2 command. Read-only.

PLB Address offset	Bit range	Field name	Description
0x30	31:16	LOS_COUNTER_BPM3	Loss-of-sync counter for BPM3 GTX link. Can be reset with the RST_LOS_CNT_BPM3 command. Read-only.
0x34	15:0	LOS_COUNTER_P0.0	Loss-of-sync counter for P0.0 GTX link. Can be reset with the RST_LOS_CNT_P0.0 command. Read-only.
0x34	31:16	LOS_COUNTER_P0.1	Loss-of-sync counter for P0.1 GTX link. Can be reset with the RST_LOS_CNT_P0.1 command. Read-only.
0x3C	0	GLOBAL_TRG_ENA	Timing component: *global_trg_ena* input (cfr. [???] for details).
0x3C	8	TRG_MODE	Timing component: *trg_mode* input (cfr. [???] for details).
0x3C	18:16	TRG_SOURCE	Timing component: *trg_source* input (cfr. [???] for details).
0x40	27:0	B_DELAY	Timing component: *b_delay* input (cfr. [???] for details).
0x44	15:0	B_NUMBER	Timing component: *b_number* input (cfr. [???] for details).
0x44	31:16	B_SPACE	Timing component: *b_space* input (cfr. [???] for details).
0x48	2:0	TRG_RATE	Timing component: *trg_rate* input (cfr. [???] for details).
0x4C	ANY	TRG_ONCE	Timing component: any write to this register generates a pulse on the timing component's *trg_once* input (cfr. [???] for details).
0x50	0	EXT_TRG_MISSING	Timing component: *ext_trg_missing* output (cfr. [???] for details).
0x50	8	READ_READY	Timing component: *read_ready* output (cfr. [???] for details).
0x54	15:0	SFP02_FILT_PKTS_IN1	Number of packets received from SFP2 input since last trigger.
0x54	31:16	SFP02_FILT_PKTS_IN0	Number of packets received from SFP0 input since last trigger.
0x58	15:0	SFP13_FILT_PKTS_IN1	Number of packets received from SFP3 input since last trigger.
0x58	31:16	SFP13_FILT_PKTS_IN0	Number of packets received from SFP1 input since last trigger.
0x5C	15:0	SFP02_FILT_PKT_DISC_Y	Number of Y-position packets coming from SFP channels 0 and 2, that have been discarded by the filter since last trigger.
0x5C	31:16	SFP02_FILT_PKT_DISC_X	Number of X-position packets coming from SFP channels 0 and 2, that have been discarded by the filter since last trigger.
0x60	15:0	SFP13_FILT_PKT_DISC_Y	Number of Y-position packets coming from SFP channels 1 and 3, that have been discarded by the filter since last trigger.
0x60	31:16	SFP13_FILT_PKT_DISC_X	Number of X-position packets coming from SFP channels 1 and 3, that have been discarded by the filter since last trigger.
0x64	15:8	SFP13_FILT_BAD_BPM_ID	BPM_ID of first discarded packet received from SFP channels 1 and 3 during current bunch.
0x64	31:24	SFP02_FILT_BAD_BPM_ID	BPM_ID of first discarded packet received from SFP channels 0 and 2 during current bunch.
0x68	15:0	SFP02_FILT_PKT_PASS_Y	Number of Y-position packets coming from SFP channels 0 and 2, that have been forwarded by the filter since last trigger.

PLB Address offset	Bit range	Field name	Description
0x68	31:16	SFP02_FILT_PKT_PASS_X	Number of X-position packets coming from SFP channels 0 and 2, that have been forwarded by the filter since last trigger.
0x6C	15:0	SFP13_FILT_PKT_PASS_Y	Number of Y-position packets coming from SFP channels 1 and 3, that have been forwarded by the filter since last trigger.
0x6C	31:16	SFP13_FILT_PKT_PASS_X	Number of X-position packets coming from SFP channels 1 and 3, that have been forwarded by the filter since last trigger.
0x70	15:0	FILT02_BUCKET_MAX	Highest bucket ID allowed so that a packet coming from channels 0 and 2 can be forwarded by the filter.
0x70	31:16	FILT02_BUCKET_MIN	Lowest bucket ID allowed so that a packet coming from channels 0 and 2 can be forwarded by the filter.
0x74	15:0	FILT13_BUCKET_MAX	Highest bucket ID allowed so that a packet coming from channels 0 and 2 can be forwarded by the filter.
0x74	31:16	FILT13_BUCKET_MIN	Lowest bucket ID allowed so that a packet coming from channels 0 and 2 can be forwarded by the filter.
0x7C	31:0	FW_VERSION	Component’s version number (unsigned integer, incremental). Current value 0x7.
0x80	24	SFP0_PING_EN	Enable sending PING packets through the SFP0 TX channel.
0x80	16	SFP1_PING_EN	Enable sending PING packets through the SFP1 TX channel.
0x80	8	SFP2_PING_EN	Enable sending PING packets through the SFP2 TX channel.
0x80	0	SFP3_PING_EN	Enable sending PING packets through the SFP3 TX channel.
0x84	24	SFP0_PING_RX	This bit is set whenever a valid PING packet is received from the SFP0 RX channel. It is reset at every rising edge of the bunch train trigger.
0x84	16	SFP1_PING_RX	This bit is set whenever a valid PING packet is received from the SFP1 RX channel. It is reset at every rising edge of the bunch train trigger.
0x84	8	SFP2_PING_RX	This bit is set whenever a valid PING packet is received from the SFP2 RX channel. It is reset at every rising edge of the bunch train trigger.
0x84	0	SFP3_PING_RX	This bit is set whenever a valid PING packet is received from the SFP3 RX channel. It is reset at every rising edge of the bunch train trigger.
0x88	31:0	SFP0_PING_LATENCY	PING latency for SFP0 link. Difference between the local free-running counter at the time the PING packet is received, and the SEND_TIME value contained in the packet itself. The value is updated every time a valid PING packet is received. The value shall be considered valid only when SFP0_PING_RX = 1.
0x8C	31:0	SFP1_PING_LATENCY	PING latency for SFP1 link. Difference between the local free-running counter at the time the PING packet is received, and the SEND_TIME value contained in the packet itself. The value is updated every time a valid PING packet is received. The value shall be considered valid only when SFP1_PING_RX = 1.

PLB Address offset	Bit range	Field name	Description
0x90	31:0	SFP2_PING_LATENCY	PING latency for SFP2 link. Difference between the local free-running counter at the time the PING packet is received, and the SEND_TIME value contained in the packet itself. The value is updated every time a valid PING packet is received. The value shall be considered valid only when SFP2_PING_RX = 1.
0x94	31:0	SFP3_PING_LATENCY	PING latency for SFP3 link. Difference between the local free-running counter at the time the PING packet is received, and the SEND_TIME value contained in the packet itself. The value is updated every time a valid PING packet is received. The value shall be considered valid only when SFP3_PING_RX = 1.

2.4 Functional description

The core's main function is implemented in the *ibfb_packet_filter* and in the *ibfb_packet_router* components. Both the *filter* and the *router* are instantiated two times as shown also in **Figure 2**. The core's top level just connects the various instances to the proper GTX channels and provides the external PLB interface.

2.4.1 IBFB_packet_filter

The **ibfb_packet_filter** component has the basic task of avoiding that the same information coming from the upstream and the downstream chain is forwarded twice to the IBFB controller. This means that after a position packet is received, any other position packet with the same BPM_ID (source) and the same BUCKET number (time offset wrt trigger) shall be discarded. The filter's memory is reset after each trigger.
The component allows also to set allowed ranges of BPM_IDs and BUCKET numbers.
Forwarded packets are then output as two separate streams, one for the X-position and one for the Y-position.
The component's architecture is shown in Figure 4.

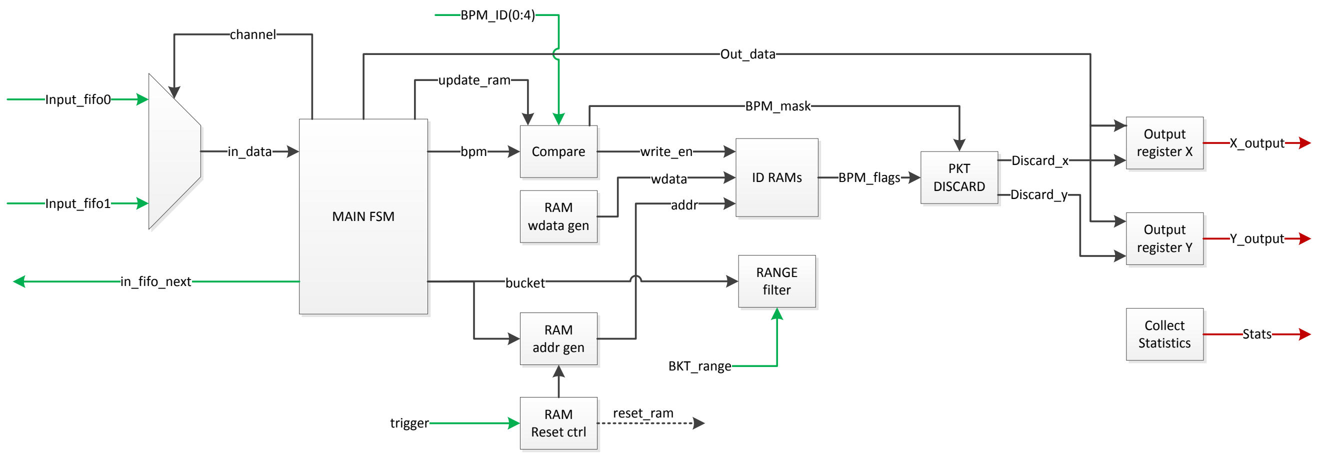

Figure 4 – IBFB packet filter's architecture

Input data is read from 2 FIFOs: one containing packets from the upstream chain, and one from the downstream chain. FIFOs are read round-robin, and the reading process is handled by the *MAIN_FSM*.
The *MAIN_FSM* is reset on external trigger (and kept under reset until the RAMs have been reset). Then it repeats the following algorithm:

- Read a packet from the currently selected input FIFO and register the data
- Wait until the output is ready to receive a new packet
- Enable the logic that keeps memory of the received packets (*update_ram*)
- Switch to the other input channel and repeat from start

As soon as the *MAIN_FSM* asserts the *update_ram* signal, the *Compare* block compares the BPM_ID from the received packet to the 4 allowed BPM_IDs and generates a 4 bit *BPM_mask*. A bit of the *BPM_mask* is set if the BPM field from the received packet matches the corresponding BPM_ID input parameter. Input BPM_ID parameters 0 and 1 identity X-position packets while BPM_ID parameters 2 and 3 identify Y-position packets.
A bank of four 4096x1bit RAMs (one for each allowed BPM_ID) is used to record whether a packet has already been received with the specified BPM_IDs. If a bit of the *BPM_mask* vector

is set, then a ‘1’ is written to the corresponding RAM. The address to which the value is stored is equal to the received packet’s *BUCKET* field.
The content of the four ID_RAMs is sequentially reset after each trigger. During the RAM reset process the whole component is kept under reset and it’s not able to receive any data. Since the RAM’s depth is 4096, this “blind” period lasts about 4096 clock cycles after each trigger.
The RAMs’ outputs are then used to decide whether the current packet can be forwarded or should be discarded. The packet discard/forward process is handled using two discard signals: one for the x-position packets (*discard_x*) and one for the y-position packets (*discard_y*). If *discard_x = 0*, then the packet is forwarded through the x_output. Likewise if *discard_y = 0* then the packet is forwarded through the y_output. The logic is as follows:

- If the *BPM_mask* has no bit set, then the current packet’s BPM does not match any of the allowed ones. The packet is marked as discarded both for X and Y outputs (*discard_x = discard_y = 1*)
- If *BPM_mask(i)* is set then the *RAM(i)* output is checked. If *RAM(i) = 1*, then the packet has already been received and it’s marked as discarded for both X and Y outputs. If *RAM(i) = 0*, then the packet is marked as good according to the value of the index *(i)*. If *(i)* is 0 or 1, then the packet is forwarded through X-output (*discard_x = 0*), while if *(i)* is 2 or 3, then then the packet is forwarded through Y-output (*discard_y = 0*).
- An additional check is made on the range of the *BUCKET* field. If the *BUCKET* is outside the range specified by the parameters *BUCKET_MIN* and *BUCKET_MAX*, the packet is discarded no matter the results of the other checks.

In short:

$$discard\_x = (packet\_valid) \cdot \left[(bpm\_mask \cdot ram\_out) + (bpm\_mask(1:0) = "00") + (bucket\_off\_range)\right]$$
$$discard\_y = (packet\_valid) \cdot \left[(bpm\_mask \cdot ram\_out) + (bpm\_mask(3:2) = "00") + (bucket\_off\_range)\right]$$

Data output is handled by two registers (*output_register_X* and *output_register_Y*). Each registers forwards the data coming from the *MAIN_FSM*. If the corresponding *discard_x/y* is ‘0’, then the packet is forwarded without changes. If otherwise the discard bit is set, then the *end_of_packet* field is replaced with a discard character (specified by the parameter *K_BAD*). The component receiving the data shall then discard the packets according to that data field.
The packet structure is summarized in Table 5.

Table 5 - IBFB position packet

DWORD	Content			
	31:24	23:16	15:8	7:0
0	0x00	K_SOP	CTRL_POS	BPM
1	BUCKET		XPOS(15:0)	
2	XPOS(31:16)		YPOS(15:0)	
3	YPOS(31:16)		CRC	K_EOP

Figure 5 shows a simplified timing diagram of the *ibfb_packet_filter* operation.

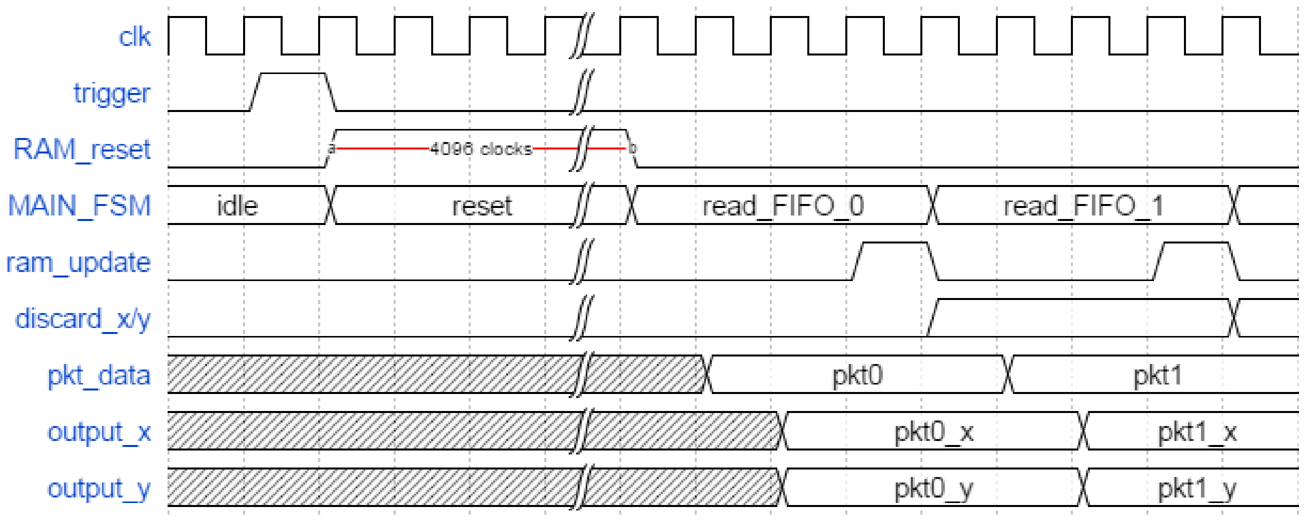

Figure 5- Timing diagram of ibfb_packet_filter

2.4.1.1 PING packets and link latency calculation

In order to evaluate the total latency of the upstream/downstream daisy chains, the TX channel of each SFP link can be enabled to send PING packets.
If an SFP has the PING_EN bit set to 1, then a PING packet is sent through the TX channel just after every trigger. The packet is expected to be received from the RX channel of the same connector before the next PING packet is sent.
The format of the ping packet is shown in Table 6 unten.

Table 6 - IBFB ping packet

DWORD	Content			
	31:24	23:16	15:8	7:0
0	0x00	K_SOP	CTRL_PING	SERIAL
1	SPARE		SEND_TIME(15:0)	
2	SEND_TIME(31:16)		SPARE	
3	SPARE		CRC	K_EOP

The structure used is the same as the IBFB position packet.
The field CTRL_PING has the constant value “0001xxxx” (‘x‘ stands for don’t care) and identifies the packet as a PING packet (as opposed to position packets that have a CTRL field equal to “0000xxxx“).
The SERIAL field identifies the specific PING packet: a received ping packet is considered valid only if it has the same serial number as the last PING packet sent. The SERIAL field is generated by means of a counter incremented on each rising edge of the trigger.
The SEND_TIME field is the value of a free-running counter, registered the moment the PING packet is sent. When a PING packet is received, the difference between the SEND_TIME value and the current counter value can be used to represent the latency of the communication chain.
The following actions are performed whenever a trigger edge is detected:

- The SERIAL counter is incremented
- A ping packet is sent containing the current value of the TIMER counter (SEND_TIME)

- The PING_RECEIVED flag is reset
- The system starts to wait for received ping packets
- If a ping packet is received, the SERIAL field is first compared to the current SERIAL counter. If the values match, then the PING_RECEIVED flag is set and the difference between the current TIMER value and the SEND_TIME field is computed and stored in the LATENCY register.

A simplified waveform describing the PING packet sending and receiving process is shown in Figure 6.

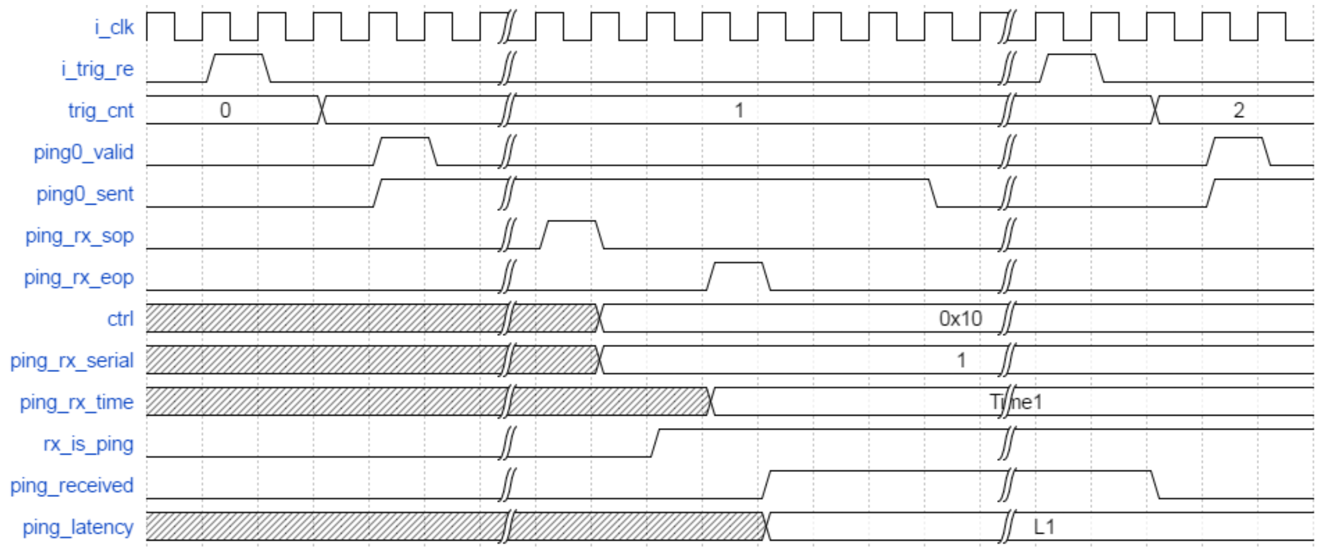

Figure 6 - Simplified waveform for the PING packet send/receive process.

2.4.1.2 Ports and parameters

Table 7 - IBFB_packet_filter component parameters.

Parameter name	Type	Description
K_SOP	std_logic_vector(7:0)	K-character used as Start-of-packet symbol in IBFB protocol. Shall be a K-character accepted by Xilinx GTX transceivers 8b/10b logic (see [3]).
K_EOP	std_logic_vector(7:0)	K-character used as End-of-packet symbol in IBFB protocol. Shall be a K-character accepted by Xilinx GTX transceivers 8b/10b logic (see [3]).
K_BAD	std_logic_vector(7:0)	Value used to replace End-of-packet symbol in IBFB packets to mark them as discarded. Shall be a K-character accepted by Xilinx GTX transceivers 8b/10b logic only if the packet is going to be transmitted through a GTX channel (see [3]). If otherwise the packet is discarded by local logic, the value can be anything different from K_EOP and K_SOP.
PLB_SLAVE	BUS	PLB slave bus (bursting) used to access local register bank for command and control purposes, and the players' RAMs. PLB bus clock is used to drive only the register bank logic and the RAM interface. The core itself runs with the clock specified by the C_USE_EXTERNAL_CLOCK parameter.

Table 8 - IBFB_packet_filter component ports.

Port name	Dir.	Type	Description
i_bpm_id	in	array(0:3) of std_logic_vector(7:0)	Allowed BPM ID. Packets with BPM field equal to BPM_ID(0) or BPM_ID(1) will be forwarded through the X_output port. Packets with BPM field equal to BPM_ID(2) or BPM_ID(3) will be forwarded through the Y_output port.
i_bkt_min	in	std_logic_vector(15:0)	Minimum allowed bucket. Any incoming packet with a BUCKET field lower than this value will be discarded.
i_bkt_max	in	std_logic_vector(15:0)	Maximum allowed bucket. Any incoming packet with a BUCKET field higher than this value will be discarded.
i_ping_enable0	in	std_logic	Enable sendig of PING packets from TX channel 0
i_ping_enable1	in	std_logic	Enable sendig of PING packets from TX channel 1
o_ping_rx0	out	std_logic	Notify whenever a valid PING packet is received from RX channel 0. The flag is reset on i_trig rising edge.
o_ping_rx1	out	std_logic	Notify whenever a valid PING packet is received from RX channel 1. The flag is reset on i_trig rising edge.
o_ping_latency0	out	std_logic_vector(31:0)	Difference between the time a PING packet is sent through TX channel 0, and the time the same packet is received through RX channel 0. Measured in i_clk cycles.
o_ping_latency1	out	std_logic_vector(31:0)	Difference between the time a PING packet is sent through TX channel 1, and the time the same packet is received through RX channel 1. Measured in i_clk cycles.
i_ram_clk	in	std_logic	Clock used to read the internal RAMs. Used only for debug purposes.
i_ram_raddr	in	std_logic_vector(11:0)	Read address for the internal RAMs. Used only for debug purposes.
o_ram_rdata	out	std_logic_vector(3:0)	Output data from internal RAMs' debug port.
i_clk	in	std_logic	Main core clock.
i_rst	in	std_logic	Active high synchronous reset.
i_trig	in	std_logic	Bunch train trigger. Resets the whole filter logic. After a rising edge is detected on this signal, the filter cannot accept input data for about 4096 i_clk cycles.
o_resetting	out	std_logic	Signal asserted during the filter's reset period (see i_trig).
o_pkt_valid	out	std_logic	Asserted whenever a valid packet is read from an input FIFO. **WARNING**: shall not be used to sample the filter's output.
o_pkt_discard_x	out	std_logic	Asserted whenever a packet is marked as discarded on the X-output.
o_pkt_discard_y	out	std_logic	Asserted whenever a packet is marked as discarded on the Y-output.
o_rxfifo_next0	out	std_logic	Input FIFO 0: next signal (input FIFO shall be FWFT).
i_rfifo_empty0	in	std_logic	Input FIFO 0: not_valid signal (input FIFO shall be FWFT).
i_rxfifo_charisk0	in	std_logic_vector(3:0)	Input FIFO 0: bit mask that identifies k-characters on the i_rxfifo_data0 bus (1 bit per byte).
i_rxfifo_data0	in	std_logic_vector(31:0)	Input FIFO 0: input data.
i_txfifo_full0	in	std_logic	Output FIFO 0: FIFO full signal.
o_txfifo_write0	out	std_logic	Output FIFO 0: FIFO write signal.
o_txfifo_charisk0	out	std_logic_vector(3:0)	Output FIFO 0: bit mask that identifies k-characters on the o_txfifo_data0 bus (1 bit per byte).

Port name	Dir.	Type	Description
o_txfifo_data0	out	std_logic_vector(31:0)	Output FIFO 0: output data
o_rxfifo_next1	out	std_logic	Input FIFO 1: next signal (input FIFO shall be FWFT).
i_rfifo_empty1	in	std_logic	Input FIFO 1: not_valid signal (input FIFO shall be FWFT).
i_rxfifo_charisk1	in	std_logic_vector(3:0)	Input FIFO 1: bit mask that identifies k-characters on the i_rxfifo_data1 bus (1 bit per byte).
i_rxfifo_data1	in	std_logic_vector(31:0)	Input FIFO 1: input data.
i_txfifo_full1	in	std_logic	Output FIFO 1: FIFO full signal.
o_txfifo_write1	out	std_logic	Output FIFO 1: FIFO write signal.
o_txfifo_charisk1	out	std_logic_vector(3:0)	Output FIFO 1: bit mask that identifies k-characters on the o_txfifo_data0 bus (1 bit per byte).
o_txfifo_data1	out	std_logic_vector(31:0)	Output FIFO 1: output data
i_output_next_x	in	std_logic	X-output: signal used by the component reading the filter's output to notify that data has been sampled (works like the READ signal in a FWFT FIFO).
o_output_valid_x	out	std_logic	X-output: data valid for the output buses o_output_charisk_x and o_output_data_x. Signal is kept asserted until i_output_next_x is sampled high.
o_output_charisk_x	out	std_logic_vector(3:0)	X-output: bit mask that identifies K-characters on the o_output_data_x bus (1 bit per byte).
o_output_data_x	out	std_logic_vector(31:0)	X-output: output data.
i_output_next_y	in	std_logic	Y-output: signal used by the component reading the filter's output to notify that data has been sampled (works like the READ signal in a FWFT FIFO).
o_output_valid_y	out	std_logic	Y-output: data valid for the output buses o_output_charisk_y and o_output_data_y. Signal is kept asserted until i_output_next_y is sampled high.
o_output_charisk_y	out	std_logic_vector(3:0)	Y-output: bit mask that identifies K-characters on the o_output_data_y bus (1 bit per byte).
o_output_data_y	out	std_logic_vector(31:0)	Y-output: output data.
o_statistics	out	record	Record containing statistics for the current bunch train. Values are reset on each rising edge of the input trigger. Data fields are the following: • Packets_chan0_in: number of valid packets received from FIFO0 • Packets_chan_in1: number of valid packets received from FIFO1 • Packets_discarded_x: number of discarded packets on X-output • Packets discarded_y: number of discarded packets on Y-output • Packets_passed_x: number of forwarded packets on X-output • Packets passed_y: number of forwarded packets on Y-output • Wrong_bpm_id: BPM_ID of first packet discarded during current bunch train

2.4.2 IBFB packet router

The **ibfb_packet_router** component collects data from several sources and forwards it to several destinations according to a routing table. Input packets are checked for consistency: malformed packets are not forwarded. The number of inputs and outputs can be set via parameter. Outputs can be enabled/disabled in real time via input port.

The component’s architecture is shown in Figure 7.

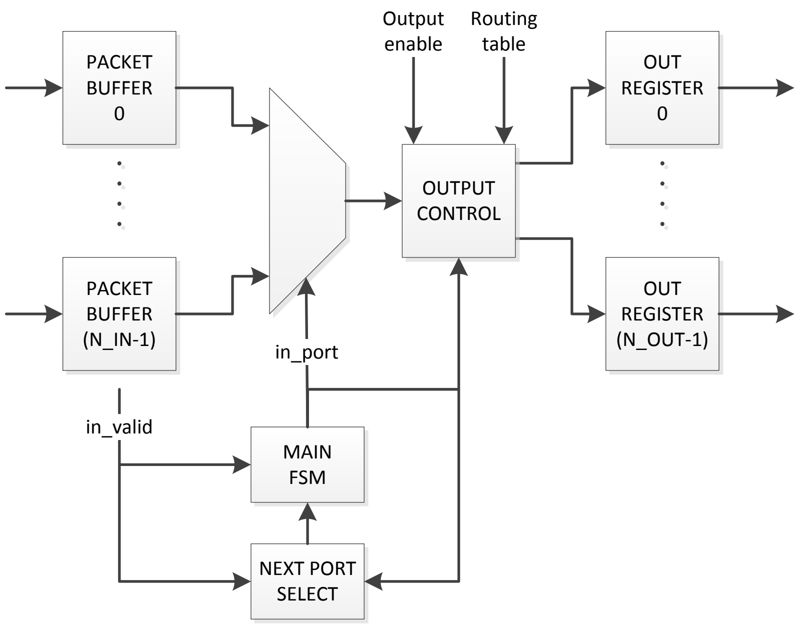

Figure 7 - IBFB_packet_router architecture.

Input data is handled by an array of *IBFB_packet_buffer* components (one for each of the router’s inputs). Each packet buffer component temporarily stores the incoming packets and checks whether their length and structure is correct. If not the packet is discarded, otherwise it’s forwarded to the rest of the logic.
The *MAIN_FSM* selects the currently active input and reads data from it. After a packet has been read from a certain input, the FSM selects the next input according to the information provided by the *NEXT_PORT_SELECT* block.
The *NEXT_PORT_SELECT* logic is fully combinatorial and finds in a single clock cycle the next buffer containing data. This allows to save the clock cycles that in a sequential scan are needed to check buffers that do not contain data. The next port selection works as follows:

- Set buffer B+1 as the 1st buffer to be checked (where B is the last buffer from which data has been read).
- Scan the buffers sequentially starting from B+1 and stop as soon as one buffer contains data (wrap to buffer 0 if last buffer is reached)
- Send the found buffer index to the *MAIN_FSM*
- Wait until data has been read from the buffer, then repeat

The data read from the buffers is then sent to the output control. This block sends the data to several output registers according to the values contained in the *Routing_Table*. The table is a 2-dimensional array of bits of size N_INPUTS x N_OUTPUTS. If the bit table(i)(j) is set, then data from input (i) is forwarded to output (j), otherwise it is not. Outputs can be selectively disabled by setting the corresponding enable bit to 0. In this case the routing table is ignored for that particular output.
The output registers provide an interface that can be directly connected to a FIFO. Anyway no backpressure is allowed, so if any output FIFO becomes full, then the data will be lost. An output error signal notifies whenever an attempt is made to write to a full output FIFO.

2.4.2.1 Ports and parameters

Table 9 - IBFB_packet_router component parameters.

Parameter name	Type	Description
K_SOP	std_logic_vector(7:0)	K-character used as Start-of-packet symbol in IBFB protocol. Shall be a K-character accepted by Xilinx GTX transceivers 8b/10b logic (see [3]).
K_EOP	std_logic_vector(7:0)	K-character used as End-of-packet symbol in IBFB protocol. Shall be a K-character accepted by Xilinx GTX transceivers 8b/10b logic (see [3]).
N_INPUT_PORTS	natural	Number of input ports (allowed range 1 to 32).
N_OUTPUT_PORTS	natural	Number of output ports (allowed range 1 to 32).
ROUTING_TABLE	array(0:31) of std_logic_vector(31:0)	Static routing table. The number of valid indexes depends on the parameters N_INPUT_PORTS and N_OUTPUT_PORTS. The 1st index specifies the input port while the 2nd index specifies the output port. If the bit ROUTING_TABLE(i)(j) is set, then all the packets received from port (i) will be forwarded to port (j). If not, then the packets won't be forwarded to that particular output.

Table 10 - IBFB_packet_router component ports.

Port name	Dir.	Type	Description
i_clk	in	std_logic	Main core clock.
i_rst	in	std_logic	Active high synchronous reset.
o_next	out	std_logic_vector (0:N_INPUT_PORTS)	Input ports: next signal. Notifies when input data has been sampled. Can be connected to the READ port of a FWFT FIFO.
i_valid	in	std_logic_vector (0:N_INPUT_PORTS)	Input ports: valid signal. Can be connected to the inverted EMPTY signal from a FWFT FIFO..
i_charisk	in	array(0:N_INPUT_PORTS) of std_logic_vector(3:0)	Input ports: bit mask that identifies k-characters on the i_data buses (1 mask per port, 1 bit per byte).
i_data	in	array(0:N_INPUT_PORTS) of std_logic_vector(31:0)	Input ports: input data.
i_out_en	in	std_logic_vector (0:N_OUTPUT_PORTS)	Output port enable. Any output port can be completely disabled by setting the corresponding bit to 0. In that case no data will be output, no matter the input or the routing table settings.
i_next	in	std_logic_vector (0:N_OUTPUT_PORTS)	Output ports: signal used by the component reading the outputs to notify that data has been sampled (works like the READ signal in a FWFT FIFO).
o_valid	out	std_logic_vector (0:N_OUTPUT_PORTS)	Output ports: data valids for the output buses o_charisk and o_data. Signal o_valid(n) is kept asserted until i_next(n) is sampled high.
o_charisk	out	array(0:N_OUTPUT_PORTS) of std_logic_vector(3:0)	Output ports: bit mask that identifies K-characters on the o_data buses (1 mask per bus, 1 bit per byte).
o_data	out	array(0:N_OUTPUT_PORTS) of std_logic_vector(31:0)	Output ports: output data.
o_err	in	std_logic_vector (0:N_OUTPUT_PORTS)	Output ports: error signal. O_err(n) is asserted whenever o_valid(n) = 1 while i_next(n) = 0 (write to full FIFO).

2.5 FPGA Resources

Table 11 – Resource utilization

Version	Flip Flops	LUTs	LUTRAMs	BRAMs	GTX
Virtex-5	3371	2823	376	21	5

2.6 Design constraints

The core has no specific constraints.