PAUL SCHERRER INSTITUT

AXI to MGT bridges

`axi_bridge_slave_1.0`
`axi_bridge_master_1.0`

Firmware Data Sheet

PSI, 09.01.2017
Rev. 1

Content

Table of Contents

List of Figures

List of Tables

1 Introduction

The GPAC3 contains 4 FPGAs, each providing several MGT connections. Some of these connections are reserved for cross-FPGA communication. Since most of the projects running of the various FPGAs are based on the AMBA AXI bus specification for internal communication, a set of components have been developed to allow the use of the same bus also for cross-FPGA communication. A typical use case is depicted in Figure 1: several FPGAs contain AXI slaves, accessible directly by local masters. Additionally an off-board master can also access all (or part) of the slaves, as if they were connected to the same bus (each slave has a unique address offset in the GPAC3 board). The MGT-to-AXI and AXI-to-MGT components take care that the AXI transactions are properly forwarded through the MGT links when needed.

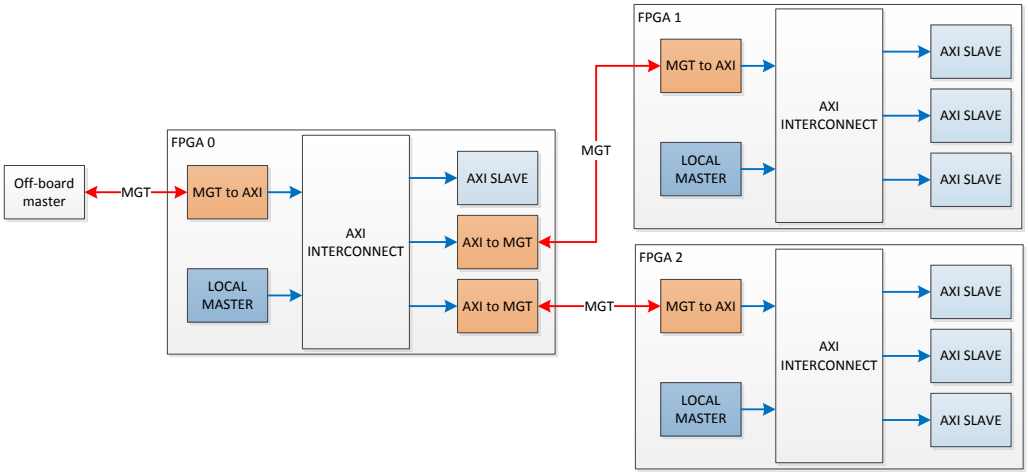

Figure 1 - Cross-FPGA communication in GPAC3.

1.1 Features

The cross-FPGA communication logic is implemented using three different components: the *axis_bridge_slave* component providing an AXI slave port (AXI to MGT in Figure 1) and the *axis_bridge_master* component providing an AXI master port (MGT to AXI in Figure 1). Each component provides also two AXI stream interfaces (TX+RX) that are meant to be connected to the MGT's TX and RX FIFOs. The MGT channel itself is implemented by the components *axis_gtp_bridge/axis_gtx_bridge* (using GTP/GTX transceivers, according to the availability in the target FPGA). The components have the following features:

- Custom address mask (allows to ignore part of the address).
- Support for writes with acknowledge (AXI style) or posted writes (PLB style).
- User-settable timeout.
- Tested up to 125 MHz for the AXI bus and 5 Gbps for the MGT links on Xilinx Kintex-7 and Artix-7.

- Support for interrupts.
- Support for legacy mode (can be connected to a PLB to MGT component). Mode can be detected automatically from the comma-character transmitted on the link, of forced manually.
- CRC check on serial communication.

1.2 Definitions, acronyms, and abbreviations

FPGA	Field Programmable Gate Array
PLB	Processor Local Bus (by IBM)
GPAC	Generic PSI ADC Carrier
AMBA	Advanced Microcontroller Bus Architecture
AXI	Advanced eXtensible Interface
MGT	Multi-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] AMBA AXI and ACE Protocol Specification, ARM, Issue D, 28 October 2011 (ARM IHI 0022D).
[3] AMBA 4 AXI4-Stream Protocol, ARM, Version 1.0, 3 March 2010 (ARM IHI 0051A).
[4] IEEE 802.3 Standard for Ethernet, https://standards.ieee.org/about/get/802/802.3.html

1.4 History

Revision	Date	Author	Description
1.0	08.01.2017	A. Malatesta	First release

2 Firmware Description

The cores *axis_bridge_slave, axis_bridge_master, axis_gtp_bridge, axis_gtx_bridge* are implemented in pure VHDL. All of them contain device specific primitives (Xilinx's GTP7GTX Transceivers and Xilinx FIFOs), therefore its portability is currently limited to Xilinx 7-Series devices providing at least one GTX_QUAD tile. The component is wrapped in a Vivado IP, but its core logic is also usable as an independent VHDL component.

3 MGT cores

Two different MGT cores are provided: one is used on devices that contain GTX transceivers (like the Kintex-7 on the GPAC3), and another one on devices that contain GTP transceivers (like the Artix-7 on the GPAC3). The two cores are functionally equivalent and will be described here as one. The common architecture of the MGT cores is shown in Figure 2.

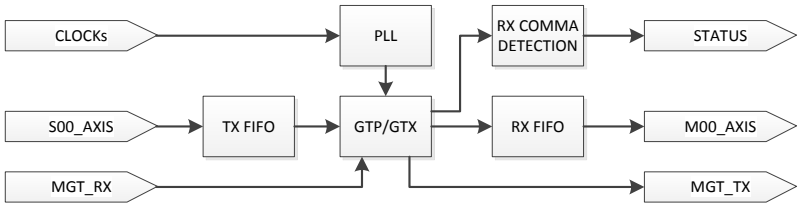

Figure 2 - Top-level block diagram of the MGT component.

The architecture is made up of the following elements:

- GTP/GTX component: instantiates the transceiver.
- TX FIFO: this component instantiates the transmission FIFO and provides additional logic for it. It manages the reset, converts the FIFO's write interface to a 32-bit AXI streaming slave interface and the FIFO's read interface to a 16-bit interface compatible with the MGT's transmit interface (data is sent lower-word first). Data is read from the FIFO and sent through the transceiver as soon as available. **When no data is available, an IDLE character is sent continuously.** The COMMA character is chosen according to the value of the input *comma_lock_axi*: if this value is high, the character K28.1 (reserved for the AXI bridge) is used, otherwise the character K28.5 (reserved for the PLB bridge) is used.
- RX FIFO: this component instantiates the receive FIFO and provides additional logic for it. It manages the reset, converts the FIFO's read interface to a 32-bit AXI streaming master interface and the FIFO's write interface to a 16-bit interface compatible with the MGT's receive interface. Data is read from the transceiver's receive interface, formatted into 32-bit words, and written to FIFO. The receive logic ignores comma characters and waits either for an interrupt K-character or a start-of-packet K-character to start writing.
- PLL: the PLL component uses the input reference clock to generate frequencies needed by the GTP/GTX component.
- RX COMMA DETECTION: the core is able to determine whether an AXI-to-MGT component or a PLB-to-MGT is connected to the far-end of the serial link. This is done by checking which comma character is received from the transceiver's RX interface. If a

continuous sequence made of the same comma character is received, then the relative comma-lock signal is asserted. If a wrong comma is received, the lock is considered to be lost and the locking procedure restarts. With the current settings, the character K28.1 is associated to the AXI-to-MGT protocol, while the character K28.5 is associated to the PLB-to-MGT protocol. A sequence of 32 consecutive comma characters is required in order to achieve lock.

3.1 Ports and Attributes

Table 1 and Table 2 describe the ports and attributes of the *axis_gt*_bridge_v1.0* component.

Table 1 - Port description for axis_gt*_bridge component.

Port name	Dir.	Type	Description
i_clk_gtp	in	std_logic_vector(1:0)	Reference clocks for the GTX/GTP transceiver. Only *i_clk_gtp[0]* is used.
i_gt*_rx_p	in	std_logic	Positive receive line for the GTX/GTP physical layer differential pair.
i_gt*_rx_n	in	std_logic	Negative receive line for the GTX/GTP physical layer differential pair.
o_gt*_tx_p	out	std_logic	Positive transmit line for the GTX/GTP physical layer differential pair.
o_gt*_tx_n	out	std_logic	Negative transmit line for the GTX/GTP physical layer differential pair.
o_lock_comma	out	std_logic	Asserted when the lock has been achieved on any comma character.
o_lock_axi	out	std_logic	Asserted when the lock has been achieved on the comma character K28.1 (AXI to MGT bridge).
o_lock_plb	out	std_logic	Asserted when the lock has been achieved on the comma character K28.5 (PLB to MGT bridge).
aclk	in	std_logic	User clock. Used for all user interfaces.
aresetn	in	std_logic	Active-low reset associated with the clock *aclk*.
s00_axis_tready	out	std_logic	User transmit interface, ready-for-data signal (AXI streaming slave, see [3]).
s00_axis_tvalid	in	std_logic	User transmit interface, data-valid signal (AXI streaming slave, see [3]).
s00_axis_tuser	in	std_logic_vector(3:0)	User transmit interface, user-data signal (AXI streaming slave, see [3]). Each bit refers to one byte in the *s00_axis_tdata* bus. If the bit is asserted, then the corresponding byte is a K-character, otherwise it is a data-byte.
s00_axis_tdata	in	std_logic_vector(31:0)	User transmit interface, data-bus signal (AXI streaming slave, see [3]).
m00_axis_tready	in	std_logic	User receive interface, ready-for-data signal (AXI streaming slave, see [3]).
m00_axis_tvalid	out	std_logic	User receive interface, data-valid signal (AXI streaming slave, see [3]).
m00_axis_tuser	out	std_logic_vector(3:0)	User receive interface, user-data signal (AXI streaming slave, see [3]). Each bit refers to one byte in the *m00_axis_tdata* bus. If the bit is asserted, then the corresponding byte is a K-character, otherwise it is a data-byte.
m00_axis_tdata	out	std_logic_vector(31:0)	User receive interface, data-bus signal (AXI streaming slave, see [3]).
debug_clk	out	std_logic	Reserved
debug	out	std_logic_vector(127:0)	Reserved

Table 2 - Attribute description for axis_gt*_bridge component.

Attribute Name	Type	Description
CPU_CLK_Hz	integer	Frequency of the input clock ACLK (in Hz).
REF_CLK_Hz	integer	Frequency of the input reference clock I_CLK_GTP[0] (in Hz).
BAUD_RATE_Mbps	integer	Transceiver's baud rate (in Megabits per second).
SIM_RESET_SPEEDUP	Boolean	Reserved
C_S00_AXIS_TDATA_WIDTH	integer	Reserved
C_M00_AXIS_TDATA_WIDTH	integer	Reserved

3.2 FPGA Resources

Table 3 –MGT component's resource utilization

Version	Flip Flops	LUTs	LUTRAMs	BRAMs	MGT
axis_gtp_bridge on Artix-7					1 GTP
axis_gtx_bridge on Kintex-7					1 GTX

4 AXI to MGT component (axis_bridge_slave)

The AXI-to-MGT core, provides an AXI4 slave interface on the user side, and a pair of AXI streaming interfaces (master+slave) to be connected to the MGT component (cfr. §3). The architecture of the component is shown in Figure 2.

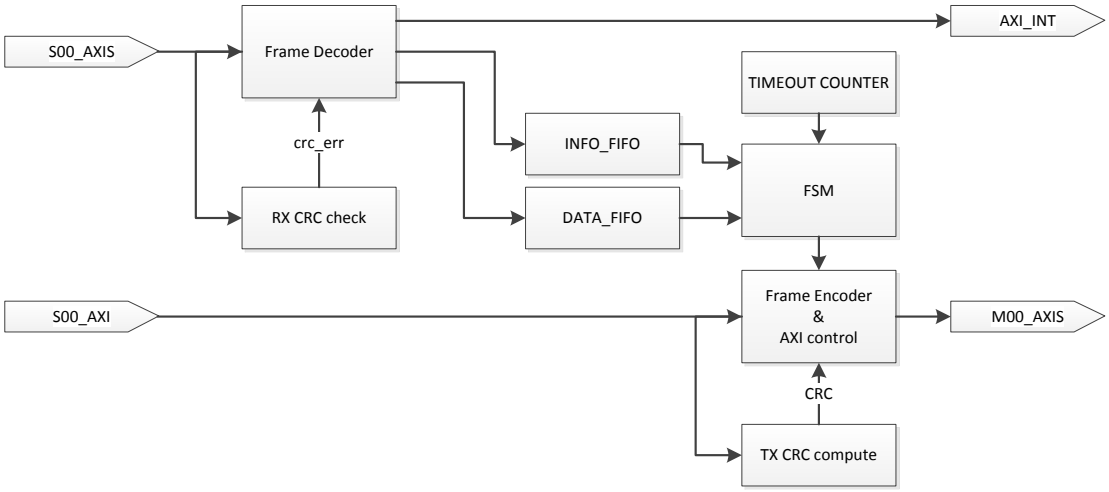

Figure 3 - Top-level block diagram of the AXI-to-MGT component (axis_bridge_slave).

The architecture is made up of the following elements:

- Frame encoder and AXI control: this block decodes the AXI transactions received from the AXI4 slaves, and encodes them into a packet to be transmitted through the MGT link.
- TX CRC compute: this block computes the CRC on the TX packets. The computed value is sent along with the packet itself to allow for error detection.
- Frame decoder: this block detects valid packets in the data stream received from the MGT link through the AXI streaming slave. Packets with a valid structure are written as multiple 32-bit words in the DATA FIFO, each data-word associated to a 4-bit packet ID. The result of the CRC check on the packet is stored in the INFO FIFO together with the same packet ID (single 8-bit word). The block also generate an interrupt pulse whenever an interrupt K-character is received.
- RX CRC compute: this block computes the CRC on the RX packets and determines whether they contain errors. The result of the CRC check (pass/fail) is stored in the INFO FIFO together with an ID assigned to the packet.
- INFO FIFO: stores one 8-bit word for each received packet. The word contains the packet identifier and the result of the CRC check on the packet itself. The FIFO is set as first-word-fall-through.
- DATA FIFO: stores the received packets as a series of 32-bit words. Four additional bits for each words are used to store the ID assigned to each packet. The FIFO is set as first-word-fall-through.

- Timeout counter: this counter is controlled by the FSM. It is started whenever a read-request is sent through the MGT link. If the counter times-out before a response is received, the logic generates an error response on the AXI4 slave.
- FSM: this finite state machine controls the whole component.

4.1 Ports and attributes

Table 4 and Table 5 describe the ports and attributes of the *axis_bridge_slave_v1.0* component.

Table 4 - Port description for the axis_bridge_slave component.

Port name	Dir.	Type	Description
MGT_LINK_UP	in	std_logic	Notifies whether the MGT link is connected. Used to avoid processing junk data received while the MGT link is down.
AXI_ACLK	in	std_logic	Main clock for the AXI interfaces.
AXI_ARESETN	in	std_logic	Active-low reset for the AXI interfaces and the whole component's logic.
AXI_INT	out	std_logic	Interrupt pulse. Asserted for 6 ACLK cycles whenever an interrupt K-character is received from the MGT link.
s00_axis_tready	out	std_logic	User transmit interface, ready-for-data signal (AXI streaming slave, see [3]).
s00_axis_tvalid	in	std_logic	User transmit interface, data-valid signal (AXI streaming slave, see [3]).
s00_axis_tuser	in	std_logic_vector(3:0)	User transmit interface, user-data signal (AXI streaming slave, see [3]). Each bit refers to one byte in the *s00_axis_tdata* bus. If the bit is asserted, then the corresponding byte is a K-character, otherwise it is a data-byte.
s00_axis_tdata	in	std_logic_vector(31:0)	User transmit interface, data-bus signal (AXI streaming slave, see [3]).
m00_axis_tready	in	std_logic	User receive interface, ready-for-data signal (AXI streaming slave, see [3]).
m00_axis_tvalid	out	std_logic	User receive interface, data-valid signal (AXI streaming slave, see [3]).
m00_axis_tuser	out	std_logic_vector(3:0)	User receive interface, user-data signal (AXI streaming slave, see [3]). Each bit refers to one byte in the *m00_axis_tdata* bus. If the bit is asserted, then the corresponding byte is a K-character, otherwise it is a data-byte.
m00_axis_tdata	out	std_logic_vector(31:0)	User receive interface, data-bus signal (AXI streaming slave, see [3]).
AXI4 SLAVE			AXI4 slave (see [2] for details).

Table 5 - Attribute description for axis_bridge_slave component.

Attribute Name	Type	Description
K_SOF	std_logic_vector(7:0)	K-character used as start-of-frame in the packet protocol. Default: 0xFB (K27.7).
K_EOF	std_logic_vector(7:0)	K-character used as end-of-frame in the packet protocol. Default: 0xFD (K29.7).
K_INT	std_logic_vector(7:0)	K-character used as interrupt in the packet protocol. Default: 0xDC (K28.6).
TIMEOUT_CYCLES	integer	Maximum time allowed for a response (in ACLK cycles). If the timeout expires, an error response is generated locally. Default: 512.

POSTED_WRITES	std_logic	When '1' no response is expected from write transactions. The AXI write response is generated locally by the logic.
IGNORE_MGT_BACKPRESSURE	std_logic	When '1' the transmission does not wait when the MGT's TX FIFO gets full (higher speed, but could lead to lost transactions when the AXI throughput is greater than the MGT throughput).
RX_ADDR_MASK	std_logic_vector(31:0)	Address mask. Both write and read addresses are bitwise ANDed with this mask.

4.2 Functional description

The *axis_bridge_slave* component converts AXI4 transactions issued by an AXI4 master, and sends them through the MGT link as data packets formatted according a custom protocol. The transactions are handled as follows:

- Write transaction: the transaction parameters (address, transaction size) and the payload are encoded as a packet and sent through the MGT link. If the POSTED_WRITES mode is enabled, the core generates a positive write acknowledge, otherwise it waits for the write acknowledge to be received from the MGT link. In the latter case, if no acknowledge is received within a certain time, an error acknowledge is generated.
- Read transaction: the transaction parameters (address, transaction size) are encoded as a packet and sent through the MGT link. Then the core waits for the requested data to be received through the MGT link. The received data is then unpacked and sent through the AXI4 interface. If the requested data is not received within a certain time, the core produces a dummy data stream of the requested size, where all the data words are marked as error.
- Interrupts: if an interrupt is received from the MGT link, then a pulse is produced on the AXI_INT output port.

The core behavior is controlled by a finite state machine. The FSM diagram is shown in Figure 4.

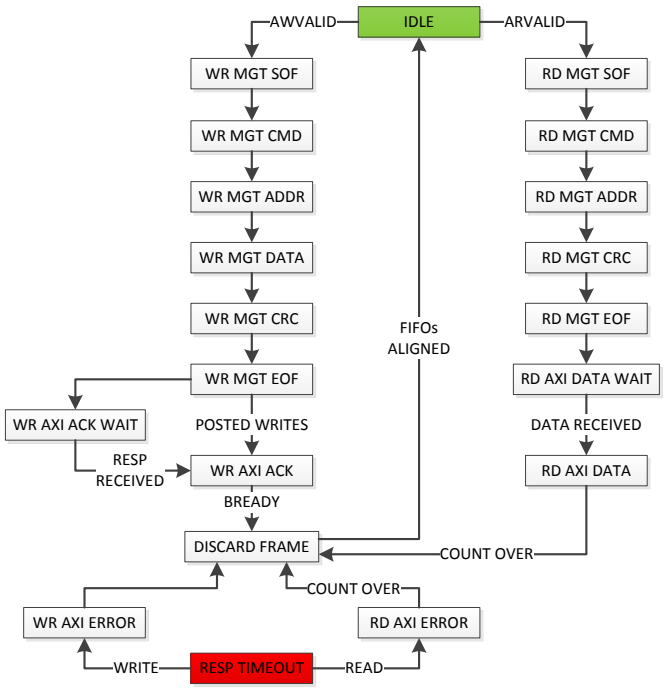

Figure 4 - State diagram for the axis_bridge_slave component's FSM.

4.2.1 FSM functional description

Here is described the FSM's behavior. For details on the packet protocol, see §4.2.2.

After reset the FSM starts in the **IDLE** state. In this state the FSM waits either for a read transaction (ARVALID = '1') or a write transaction (AWVALID = '1'). ***When two transactions are received at the same time, the write transaction is served first***.

If a write transaction is started while in IDLE state, then the transition parameters are stored (AWADDR, AWID, AWBURST, AWLEN), the write transaction is acknowledged on the AXI slave (AWREADY) and the FSM moves to state **WR_MGT_SOF**. In this state the start-of-frame is sent through the AXI Stream Master interface. The FSM moves then to state WR_MGT_CMD.

In state **WR_MGT_CMD**, the command word is sent through the AXI Stream Master interface. The command word is composed according to the protocol described in §4.2.2. The FSM then moves to the WR_MGT_ADDR state.

In state **WR_MGT_ADDR** the AXI address is sent through the AXI Stream master interface. The FSM then moves to the WR_MGT_DATA state.

In state **WR_MGT_DATA** the AXI data is sent through the AXI Stream master interface. The AXI4 Slave signal WVALID is used as a data-valid signal for the AXI Stream master interface. As soon as the last data word has been sent (WLAST = '1') the FSM moves to the WR_MGT_CRC state.

In state **WR_MGT_CRC** the data packet's checksum is sent through the AXI Stream Master interface. A 32-bit CRC is calculated over the whole packet payload (command word,

address, data). See §4.2.2.1 for details on how the CRC is calculated. The FSM moves then to state WR_MGT_EOF.

In state **WR_MGT_EOF**, the end-of-frame is sent through the AXI Stream Master interface. If the POSTED_WRITES option is enabled, then the FSM moves directly to the WR_AXI_ACK state. Otherwise it moves to the WR_AXI_ACK_WAIT state.

In state **WR_AXI_ACK_WAIT**, the FSM waits for a write acknowledge packet to be received from the AXI Stream Slave interface. While in this state, the RX FIFOs are read as soon as data is available (see §4.2.3 for details on the RX interface). Only after the packet is received, the FSM moves to state WR_AXI_ACK. While in this state a timeout counter is enabled. If the timeout expires before the acknowledge packet is received, then the FSM moves to the WR_AXI_ERR state.

In state **WR_AXI_ACK**, the FSM generates a write response (BRESP) on the AXI4 Slave bus. If the POSTED_WRITE option is disabled, then the response value is taken from the write acknowledge packet received from the AXI Stream Slave interface. Otherwise a GOOD response (0x0) is produced. As soon as the write response is acknowledged on the AXI4 Slave bus (BREADY = '1'), the FSM moves to state DISCARD_FRAME.

In state **WR_AXI_ERR**, the FSM generates a write error response (BRESP = 0x2) on the AXI4 Slave bus. As soon as the write response is acknowledged on the AXI4 Slave bus (BREADY = '1'), the FSM moves to state DISCARD_FRAME.

If a read transaction is started while in IDLE state, then the transition parameters are stored (ARADDR, ARID, ARBURST, ARLEN), the read transaction is acknowledged on the AXI slave (ARREADY) and the FSM moves to state **RD_MGT_SOF**. In this state the start-of-frame is sent through the AXI Stream Master interface. The FSM moves then to state RD_MGT_CMD.

In state **RD_MGT_CMD**, the command word is sent through the AXI Stream Master interface. The FSM then moves to the RD_MGT_ADDR state.

In state **RD_MGT_ADDR** the AXI address is sent through the AXI Stream master interface. The FSM then moves to the RD_MGT_CRC state.

In state **RD_MGT_CRC** the data packet's checksum is sent through the AXI Stream Master interface. A 32-bit CRC is calculated over the whole packet payload (command word, address). See §4.2.2.1 for details on how the CRC is calculated. The FSM moves then to state RD_MGT_EOF.

In state **RD_MGT_EOF**, the end-of-frame is sent through the AXI Stream Master interface. The FSM then moves to the RD_AXI_DATA_WAIT state.

In state **RD_AXI_DATA_WAIT**, the FSM waits for a read data packet to be received from the AXI Stream Slave interface. While in this state, the RX INFO FIFO is read as soon as data is available (see §4.2.3 for details on the RX interface). Only after the packet is received, the FSM moves to state RD_AXI_DATA. While in this state a timeout counter is enabled. If the timeout expires before the acknowledge packet is received, then the FSM moves to the RD_AXI_ERR state.

In state **RD_AXI_DATA**, the FSM reads from the DATA FIFO and sends it through the AXI4 Slave interface's read-data channel (RDATA/RVALID). All returned data is marked as good (RRESP = 0x0). The count of returned data words in kept by an internal counter. As soon as the requested amount of data has been returned (ARLEN), the RLAST pulse is generated on the AXI4 Slave interface, and the FSM moves to state DISCARD_FRAME. While in this state a timeout counter is enabled. If the timeout expires before the requested amount of data has been returned (eg. if an incomplete packet has been received), then the FSM moves to the RD_AXI_ERR state.

In state **RD_AXI_ERR**, the FSM generates the remaining part of the requested read transaction. If ARLEN words have been requested, and only N data words have been returned

while in the RD_AXI_DATA state, then (ARLEN-N) dummy data words are generated (0xDEADBEE4) and marked with an error code (RRESP = 0x2) on the AXI4 Slave bus. As soon as the requested amount of data has been returned (ARLEN), the RLAST pulse is generated on the AXI4 Slave interface, and the FSM moves to state DISCARD_FRAME.

In state **DISCARD_FRAME**, the FSM cleans up the RX DATA FIFO. Data is read from the FIFO until the data ID matches the one currently showing on the INFO FIFO's output, or until the data FIFO is empty (see §4.2.3 for details on the RX interface).

4.2.2 Packet protocol

The bridge components exchange data through the MGT channels by converting the AXI transactions into data packets. Each packet is a sequence of 32-bit words, whose length can vary. Each data word in the packet is associated with a 4-bit mask, each bit corresponding to a byte in the data word: if the bit is '0', then the byte is marked as DATA (D), otherwise is marked as COMMAND (K).

In the tables below is described the packet structure for each supported AXI4 transaction. The upper row shows the MASK (in binary) and the lower row the corresponding DATA. The packet fields are described in detail at the end of this section.

Table 6 - Packet protocol: AXI write transaction.

0101	0000	0000	0000	0000	0000	0000	0101
SOF	CMD	ADDR	$DATA_0$	…	$DATA_{n-1}$	CRC	EOF

Table 7 - Packet protocol: AXI write response.

0101	0000	0000	0101
SOF	CMD	CRC	EOF

Table 8 - Packet protocol: AXI read request.

0101	0000	0000	0000	0101
SOF	CMD	ADDR	CRC	EOF

Table 9 - Packet protocol: AXI/PLB read response.

0101	0000	0000	0000	0000	0000	0101
SOF	**CMD**[1]	$DATA_0$	…	$DATA_{n-1}$	CRC	EOF

4.2.2.1 Packet fields description

The **SOF** field is the start-of-frame and it is composed as follows:

0x00	K_SOF	0x00	K_IDL

[1] This field is present only when LEGACY_MODE = 0 (AXI mode). If on the other hand LEGACY_MODE = 1 (PLB mode) then the read response packet contains only the SOF, the returned data, the CRC and the EOF.

K_SOF is the start-of frame K-character set by generic (default 0xFB, see Table 5). The K_IDL is the IDLE K-character, also set by generic (default 0x3C, see Table 5). The SOF field is present in all the packets.

The **EOF** field is the end-of-frame and it is composed as follows:

0x00	K_IDL	0x00	K_EOF

K_EOF is the end-of frame K-character set by generic (default 0xFD, see Table 5). The K_IDL is the IDLE K-character, also set by generic (default 0x3C, see Table 5). The EOF field is present in all the packets.

The **CMD** field contains the transaction details, and it is composed as follows:

31	30:29	28	27	26:23	22:9	8	7:0
RNW	0	BURST	0	WSTRB	0	RESP	LENGTH/BRESP

RNW specifies whether the transaction is a read or a write (‘1’ for read, ‘0’ for write).

BURST specifies whether the transaction is single or burst (‘1’ for burst, ‘0’ for single).

WSTRB is the write strobe (used only for single writes).

RESP identifies the packet as a response.

LENGTH is the number of words in the transaction minus 1 (0 for single transactions). Matches the value of the AWLEN/ARLEN AXI4 signals. In write response packets the lest significant 2-bits contain the value of the AXI BRESP signal.

The CMD field is present in all the packets for AXI transactions. **It is not present in the read-response packet when in legacy mode** (used when the serial link of a MGT_to_AXI component is connected to a PLB-to-MGT component).

The **ADDR** field contains the AXI4 address (AWARRD/ARADDR) AND-masked with the parameter RX_ADDR_MASK (see Table 5). The ADDR field is present only in the Write and Read Request packets.

The **DATA** fields carry the transaction payload. The number of data fields in each packet depends on the value of AWLEN/ARLEN for the corresponding transaction. The DATA fields are present only in the Write and Read Response packets.

The **CRC** field holds the packet’s checksum. As a general rule the checksum is computed on the whole packet, excluding the SOF and EOF data words. The type of checksum implemented in the protocol is a 802.3 CRC32 (see [4]). The CRC polynomial is

$$1+x+x^2+x^4+x^5+x^7+x^8+x^{10}+x^{11}+x^{12}+x^{16}+x^{22}+x^{23}+x^{26}+x^{32}$$

4.2.3 Receive interface

The receive interface is in charge of decoding the received packets, checking their integrity, and storing them. It is made up by a packet decoder, a component that computes CRC, a DATA FIFO and an INFO FIFO.

The packet decoder logic detects the start-of-frame and end-of-frame markers in the data received through the AXI Stream Slave interface. Every time an end-of-frame is detected, a 4-bit ID counter is incremented. All the received data, except for the SOF and EOF words, is

stored directly in the 36-bit DATA FIFO along with the current value of the ID counter. On the stored data the CRC32 is calculated and compared with the one stored in the last word of the packet itself. When the end-of-frame is detected, the result of the CRC computation (pass/fail) is stored in the 8-bit INFO FIFO along with the current value of the ID counter.

With this logic, each received packet is assigned an ID. As soon as a new info-word shows up in the INFO FIFO, it means that a whole packet has been written to the DATA FIFO. The packet itself can be retrieved from the DATA FIFO by reading all the data words with the same ID as the one stored in the info-word. The info-word contains also the result of the CRC check: this information is used to decide whether the packet should be used or discarded.

An example of how a packet is received is shown by the waveform in Figure 5.

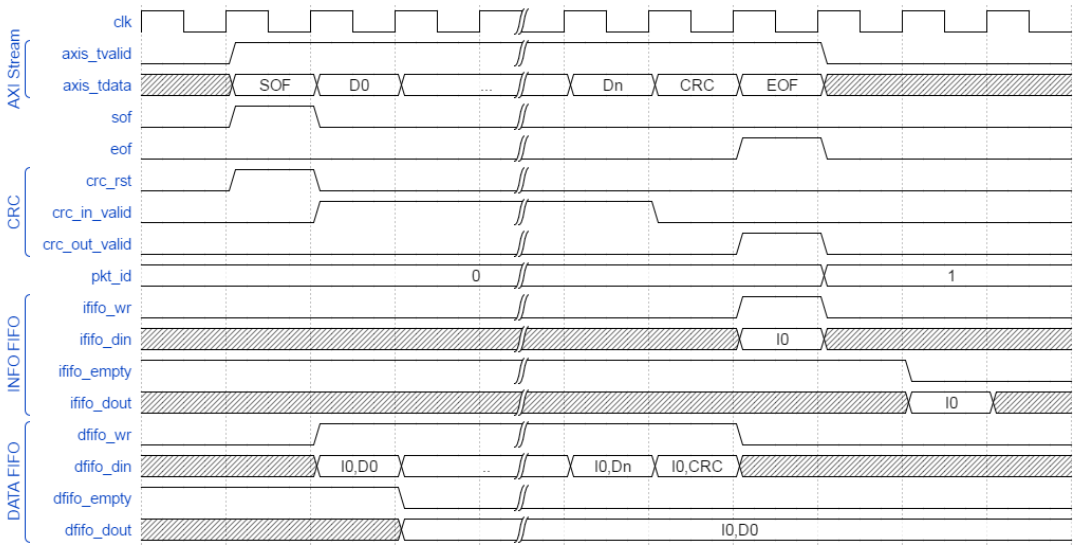

Figure 5 - Waveform describing a packet being received from AXI Stream Slave.

4.2.4 Interrupts

Interrupts are sent and received using a single K-character (specified by the parameter K_INT, see Table 5). When the *axis_mgt_slave* component receives a K_INT character through the AXI Stream Slave interface, it generates a 6-clock cycles long pulse on the output signal AXI_INT. The K_INT symbol shall be aligned on the lower 16 bits of the TDATA input bus. Bits 15:8 shall be set to 0 while bits 31:16 are don't-care. Thus an interrupt is trigger whenever a 32-bit data word with the following format is received through the AXI Stream Slave interface:

31:16	15:8	7:0
D.C.	0x00	K_INT

4.3 FPGA Resources

Table 10 – AXI-to-MGT component's resource utilization

Version	Flip Flops	LUTs	LUTRAMs	BRAMs	GTX
7-Series					

5 MGT to AXI component (axis_bridge_master)

The MGT-to-AXI core, provides an AXI4 master interface on the user side, and a pair of AXI streaming interfaces (master+slave) to be connected to the MGT component (cfr. §3). The architecture of the component is shown in .

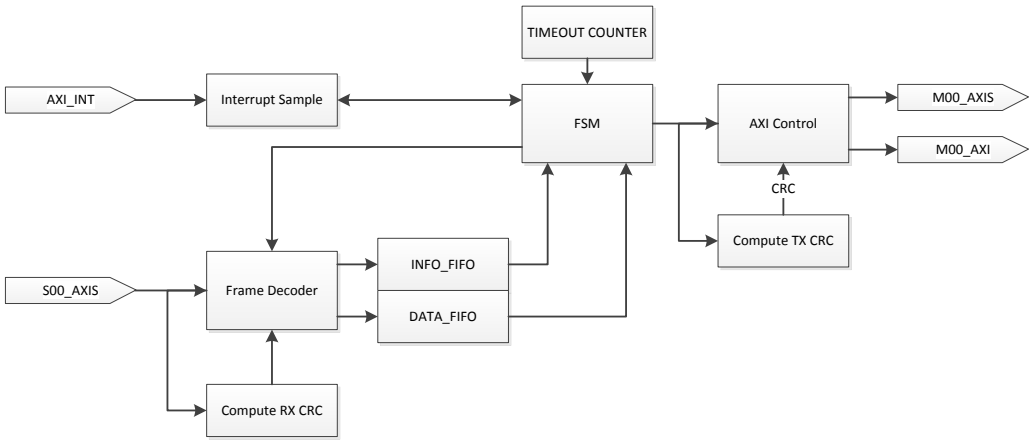

Figure 6 - Top-level block diagram of the MGT-to-AXI component (axis_bridge_master).

The architecture is made up of the following elements:

- Interrupt sampler: this block samples external interrupts and sets a flag for the FSM. When the interrupt is transmitted, the FSM resets the flag.
- Frame decoder: this block detects valid packets in the data stream received from the MGT link through the AXI streaming slave. Packets with a valid structure are written as multiple 32-bit words in the DATA FIFO, each data-word associated to a 4-bit packet ID. The result of the CRC check on the packet is stored in the INFO FIFO together with the same packet ID (single 8-bit word).
- RX CRC compute: this block computes the CRC on the RX packets and determines whether they contain errors. The result of the CRC check (pass/fail) is stored in the INFO FIFO together with an ID assigned to the packet.
- INFO FIFO: stores one 8-bit word for each received packet. The word contains the packet identifier and the result of the CRC check on the packet itself. The FIFO is set as first-word-fall-through.
- DATA FIFO: stores the received packets as a series of 32-bit words. Four additional bits for each words are used to store the ID assigned to each packet. The FIFO is set as first-word-fall-through.
- Timeout counter: this counter is controlled by the FSM. It is started whenever an AXI read request is issued through the AXI4 Master, and the component is waiting for the requested data to be delivered. If the counter times-out before data is received, an alarm flag is set. This signal is currently used only for debug purposes.
- TX CRC compute: this block computes the CRC on the TX packets. The computed value is sent along with the packet itself to allow for error detection.
- AXI control: this block controls the AXI4 Master and the AXI Streaming Master signals according to the FSM status and the data collected from the AXI Streaming Slave.
- FSM: this finite state machine controls the whole component.

5.1 Ports and attributes

Table 11 and Table 12 describe the ports and attributes of the *axis_bridge_master_v1.0* component.

Table 11 - Port description for the axis_bridge_master component.

Port name	Dir.	Type	Description
LEGACY_MODE	in	std_logic	Notifies whether the MGT link is connected to an AXI_to_MGT component ('0') or to a PLB_to_MGT component ('1'). According to this value, the packet protocol changes accordingly (see §4.2.2 for details). This port can be hardwired or connected to the *o_lock_axi* port of the MGT component (see Table 1).
AXI_ACLK	in	std_logic	Main clock for the AXI interfaces.
AXI_ARESETN	in	std_logic	Active-low reset for the AXI interfaces and the whole component's logic.
AXI_INT	in	std_logic	Interrupt request. Only the rising edge is detected. Interrupts are not queued.
s00_axis_tready	out	std_logic	User transmit interface, ready-for-data signal (AXI streaming slave, see [3]).
s00_axis_tvalid	in	std_logic	User transmit interface, data-valid signal (AXI streaming slave, see [3]).
s00_axis_tuser	in	std_logic_vector(3:0)	User transmit interface, user-data signal (AXI streaming slave, see [3]). Each bit refers to one byte in the *s00_axis_tdata* bus. If the bit is asserted, then the corresponding byte is a K-character, otherwise it is a data-byte.
s00_axis_tdata	in	std_logic_vector(31:0)	User transmit interface, data-bus signal (AXI streaming slave, see [3]).
m00_axis_tready	in	std_logic	User receive interface, ready-for-data signal (AXI streaming slave, see [3]).
m00_axis_tvalid	out	std_logic	User receive interface, data-valid signal (AXI streaming slave, see [3]).
m00_axis_tuser	out	std_logic_vector(3:0)	User receive interface, user-data signal (AXI streaming slave, see [3]). Each bit refers to one byte in the *m00_axis_tdata* bus. If the bit is asserted, then the corresponding byte is a K-character, otherwise it is a data-byte.
m00_axis_tdata	out	std_logic_vector(31:0)	User receive interface, data-bus signal (AXI streaming slave, see [3]).
AXI4 MASTER			AXI4 master port (see [2] for details).

Table 12 - Attribute description for axis_bridge_master component

Attribute Name	Type	Description
K_SOF	std_logic_vector(7:0)	K-character used as start-of-frame in the packet protocol. Default: 0xFB (K27.7).
K_EOF	std_logic_vector(7:0)	K-character used as end-of-frame in the packet protocol. Default: 0xFD (K29.7).
K_INT	std_logic_vector(7:0)	K-character used as interrupt in the packet protocol. Default: 0xDC (K28.6).
TIMEOUT_CYCLES	integer	When a transaction takes longer as this time to complete, a flag is set. Used only for debug.
RX_ADDR_MASK	std_logic_vector(31:0)	Address mask. Both write and read addresses are bitwise ANDed with this mask.

5.2 Functional description

The *axis_bridge_master* component receives data packets from the MGT link (sent by a AXI_to_MGT or PLB_to_MGT component) and converts them back to AXI4 master transactions. The core also samples interrupt requests, and sends them through the serial link.

The core behavior is controlled by a finite state machine. The FSM diagram is shown in Figure 4Figure 7.

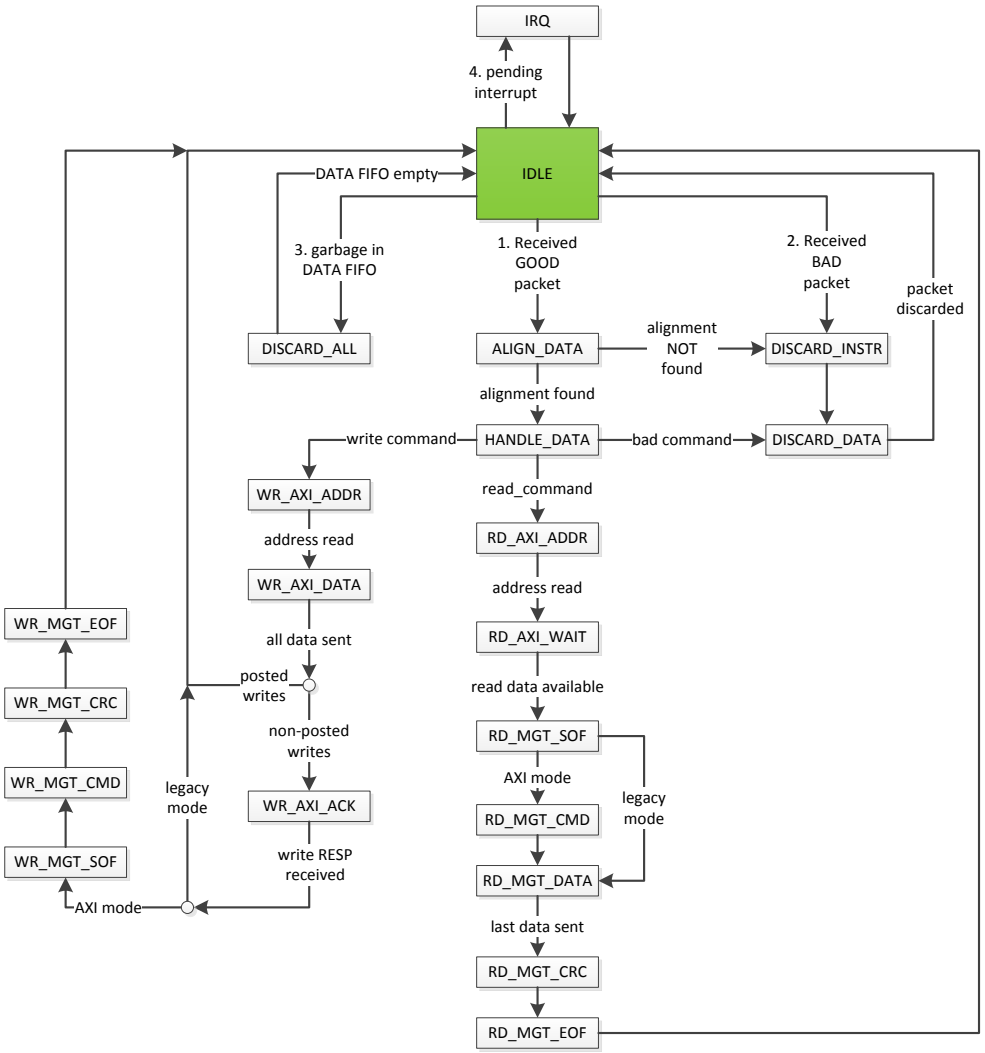

Figure 7 - State diagram for the axis_bridge_master component's FSM.

5.2.1 FSM functional description

Here is described the FSM's behavior. For details on the packet protocol, see §4.2.2.

After reset the FSM starts in the **IDLE** state. In this state the FSM waits for a new packet to be available. Whenever a new packet is received, an info-word is written to the INFO FIFO (see §4.2.3 for details on the RX interface). The info-word is then read by the FSM: if the packet's CRC check was good, then the FSM moves to state ALIGN_DATA, otherwise the FSM moves to the DISCARD_INSTR state, where the bad packet will be discarded. If while in the IDLE state the DATA_FIFO becomes full while the INFO FIFO stays empty, this is interpreted as junk data being received through the AXI Streaming Slave interface. In this case the FSM moves to the DISCARD_ALL state where the DATA FIFO will be cleaned up. Finally, if an interrupt request is pending and none of the previous conditions is met, the FSM moves to the IRQ state. While in the IDLE state the AXI write counter is also reset.

Comment [ML841]: This logic can cause the interrupt request to be delayed significantly when lots of traffic is received through the AXI Streaming slave. Perhaps it would be better to serve the interrupt requests periodically.

In state **DISCARD_ALL** the DATA_FIFO is read until the FIFO itself becomes empty or until the INFO FIFO is not empty anymore. Then the FSM moves back to the IDLE state.

In state **DISCARD_INSTR** the INFO FIFO is advanced (the current info-word is discarded) and the FSM moves directly to the DISCARD_DATA state.

In state **DISCARD_DATA** the data FIFO is read until the ID stored together with the data does not match anymore with the one stored in the info-word currently output by the INFO FIFO, or until the DATA FIFO itself becomes empty. Then the FSM moves back to the IDLE state.

In state **IRQ** the interrupt-pending flag is reset and the interrupt K-character is sent through the MGT link. The FSM moves then back to the IDLE state.

In state **ALIGN_DATA** the FSM checks whether the ID written in the info-word matches the one associated with the data currently showing on the DATA FIFO's output. If so, then the FSM moves to the HANDLE_DATA state. Otherwise it could be that some junk data has been written to the DATA FIFO before a packet has been received. In this case the DATA FIFO is read until a match is found. If a match is found before the DATA FIFO gets empty, then the FSM moves to the HANDLE_DATA state. Otherwise the FSM moves to the DISCARD_INSTR state to discard the info-word for which no matching data was found in the DATA FIFO.

In state **HANDLE_DATA** the FSM parses the received command (1st packet's data word) and decides whether to start a read or write transaction. The relevant information from the packet's CMD field (see §4.2.2.1 for details) is used to build a 4-bit RX_FRAME_OPCODE. The opcode is composed as follows:

LEGACY	RnW	BURST	RESP

Where LEGACY is the value of the LEGACY_MODE input (see §5.1), RNW is the read-not-write flag, BURST specifies if the transaction is SINGLE ('0') or BURST ('1'), and RESP specify whether the packet is a write response. The supported opcodes are the following:

0x0	AXI single write
0x2	AXI burst write
0x4	AXI single read
0x6	AXI burst read
0x8	PLB single write
0xA	PLB burst write
0xC	PLB single read
0xE	PLB burst read

If the opcode corresponds to a supported read transaction, then the value of the AXI signals ARLEN and ARBURST are set accordingly, and the FSM moves to state RD_AXI_ADD. If instead it corresponds to a supported write transaction, then the value of the AXI signals AWLEN, AWBURST and WSTRB are set accordingly, and the FSM moves to state WR_AXI_ADDR. If the opcode is not supported, then the FSM moves to state DISCARD_DATA. Before leaving this state, both FIFOs are read, discarding the already used data (the info-word in the INFO FIFO and the CMD word from the DATA FIFO).

In state **RD_AXI_ADDR** a single data word is read from the DATA FIFO and used as the AXI read address (ARADDR), then the AXI read address valid signal (ARVALID) is asserted. As soon as the address has been sampled (ARREADY = '1') the ARVALID is deasserted and the FSM moves to state RD_AXI_WAIT.

In state **RD_AXI_WAIT** the FSM waits until the requested data is available on the AXI Master Read Channel (RVALID = '1'). Then the CMD field for the read-response packet is composed according to the packet protocol described in §4.2.2, and the FSM moves to the RD_MGT_SOF state.

In state **RD_MGT_SOF** the start-of-packet word is sent through the AXI Streaming Master interface (see §4.2.2 for details). Then the FSM moves to the RD_MGT_CMD state (if the system is in AXI mode, ie. LEGACY_MODE = '0'). If instead the system is in legacy mode (LEGACY_MODE = '1'), then the FSM bypasses the RD_MGT_CMD state and moves directly to the RD_MGT_DATA state. This reflects a fundamental protocol difference between the AXI_to_MGT components and the PLB_to_MGT components: in AXI mode the read response packet contains also a CMD field that identifies the packet as a read response. On the opposite, in PLB/legacy mode, the read response packet contains only the returned data and the CRC (see §4.2.2 for details).

In state **RD_MGT_CMD** the CMD field is sent through the AXI Streaming Master interface (see §4.2.2 for details). Then the FSM moves to the RD_MGT_DATA state.

In state **RD_MGT_DATA** the data received from the AXI read channel (RDATA/RVALID) is sent through the AXI Streaming Master interface. After the data marked as last (RLAST = '1') has been sent, the FSM moves to state RD_MGT_CRC.

In state **RD_MGT_CRC** the CRC computed on the packet currently being transmitted is sent through the AXI Streaming Master interface. Then the FSM moves to the RD_MGT_EOF state.

In state **RD_MGT_EOF** the end-of-frame is sent through the AXI Streaming Master interface (see §4.2.2 for details). Then the FSM moves back to the IDLE state.

In state **WR_AXI_ADDR** a single data word is read from the DATA FIFO and used as the AXI write address (AWADDR), then the AXI write address valid signal (AWVALID) is asserted. As soon as the address has been sampled (AWREADY = '1') the AWVALID is deasserted and the FSM moves to state WR_AXI_DATA.

In state **WR_AXI_DATA** the FSM reads data from the DATA FIFO and sends it through the AXI Master's write channel (WDATA/WVALID). The data is counted while it's sent. As the counter reaches the size of the write transaction specified in the packet's CMD field (AWLEN, see §4.2.2 for details), the write phase is completed. At this point, if the POSTED_WRITES option is enabled, the FSM goes back to the IDLE state. Otherwise the FSM moves to the WR_AXI_ACK state to wait for the AXI write acknowledge.

In state **WR_AXI_ACK** the FSM waits for an AXI Write Acknowledge (BVALID/BRESP) from the AXI Master. When the acknowledge is received, the response content is stored. Then the FSM goes back to the IDLE state if the PLB mode is enabled (LEGACY_MODE = '1'). If

instead the AXI mode is enabled (LEGACY_MODE = '0'), then the FSM moves to state WR_MGT_SOF to send back a write response packet.

In state **WR_MGT_SOF** the start-of-packet word is sent through the AXI Streaming Master interface (see §4.2.2 for details). Then the FSM moves to the WR_MGT_CMD state.

In state **WR_MGT_CMD** the CMD for the write response packet is sent through the AXI Streaming Master interface (see §4.2.2 for details). Then the FSM moves to the WR_MGT_CRC state.

In state **WR_MGT_CRC** the CRC computed on the packet currently being transmitted is sent through the AXI Streaming Master interface. Then the FSM moves to the WR_MGT_EOF state.

In state **WR_MGT_EOF** the end-of-frame is sent through the AXI Streaming Master interface (see §4.2.2 for details). Then the FSM moves back to the IDLE state.

5.2.2 Packet protocol

The packet protocol used by the *axis_bridge_master* component is the same used by the slave component. Please refer to §4.2.2 for a detailed description.

5.2.3 Receive interface

The receive interface implemented in the *axis_bridge_master* component is the same used by the slave component. Please refer to §4.2.3 for a detailed description.

5.2.4 Interrupts

Interrupts are sent and received using a single K-character (specified by the parameter K_INT, see Table 12). The *axis_mgt_master* component interprets a rising edge on the AXI_INT input port as an interrupt request. The interrupt request is then sent through the AXI Stream Slave interface, as a single 32-bit word with the following format:

31:16	15:8	7:0
D.C.	0x00	K_INT

The logic used to decide when to send the interrupt request is described in §5.2.1.

5.3 FPGA Resources

Table 13 – MGT_to_AXI component's resource utilization

Version	Flip Flops	LUTs	LUTRAMs	BRAMs	GTX
7-Series					