Advanced Flexibilis Ethernet Controller (AFEC)

User Manual

This document could contain technical inaccuracies or typographical errors. Flexibilis Oy may make changes in the product described in this document at any time.

Please, email comments about this document to support@flexibilis.com.

Trademarks

Altera®, Arria™GX, Cyclone™ II, Cyclone™ III, Stratix™ II, Stratix™ II GX, Stratix™ III, Avalon™, Nios® II and Quartus® II are trademarks of Altera Corporation.

Contents

Figures

Tables

Revision History

Rev	Date	Description
0.1	28.1.2008	First version of this document
1.0	22.2.2008	First approved version.
1.1	24.6.2009	Updates to chapters: 5.7 Reset 5.3.3 Errors during Reception
1.2	3.7.2009	Upadates to chapter 5.3.3 Errors during Reception Added 'EIF Drop Error' bit to RXDESC registers in Table 7

1 About this Document

This document describes Advanced Flexibilis Ethernet Controller (AFEC). AFEC is Ethernet Media Access Control (MAC) Intellectual Property (IP) core targeted for programmable hardware platforms. AFEC has been written in VHDL language.

AFEC is based on FEC (Flexibilis Ethernet Controller). Basically AFEC is an improved version of FEC; it has several advanced features when compared to FEC. These features include transmit and receive scatter gather, wider bus interface, delayed interrupts, time stamping and an application specific interface that makes it possible to support for example encryption and IEEE 1588 v2 transparent mode. Because of these advanced features, AFEC consumes more FPGA resources than FEC.

This document is targeted at users and potential users of AFEC. This includes those who are designing software employing the functions of AFEC and those who are evaluating the usability of AFEC in their system or project. This documentation is not targeted at those who want to make changes to the implementation of AFEC, and therefore this document does not cover internal implementation of AFEC.

Chapter 2 gives a general overview of AFEC; what it is, what can be done with it, and what the main features of it are. The external signal descriptions are in Chapter 3 and memory maps and register descriptions in Chapter 4. Finally, Chapter 5 demonstrates how the functions of AFEC are controlled using the registers and how AFEC behaves in different situations.

In this document the signal names are written in `SignalName` style. The names of the blocks are written with Capital first letter. Pseudo code is written in `PseudoCode` style and command line commands are written in CommandLine style. References are in [brackets].

The bit and byte order in this document and in AFEC is presented in Figure 1. This byte order applies to all the signals and registers.

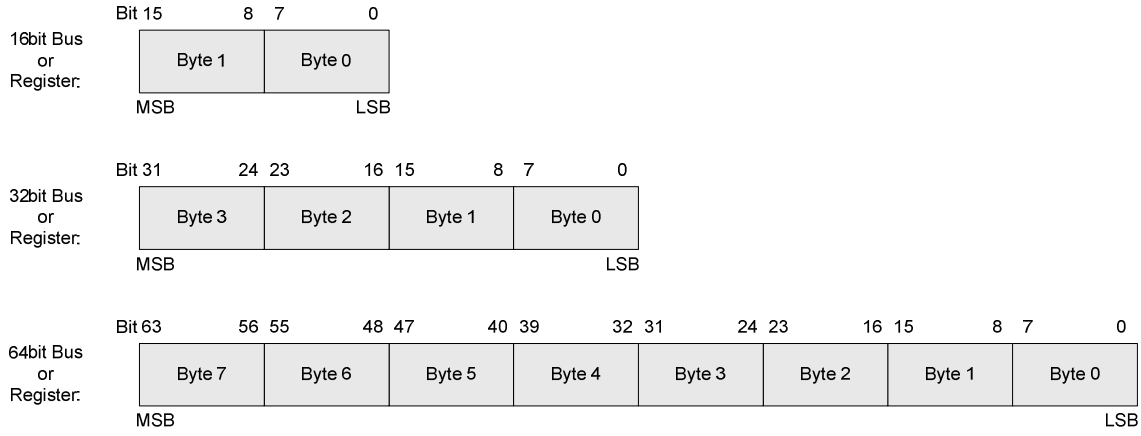

Figure 1 Bit and Byte Order

2 General Overview

AFEC is intended for use as an Ethernet Media Access Control (MAC) in programmable System on Chip (SoC) environments like Altera's Nios II. Altera Avalon [2] SoC bus interface is provided, which makes it possible to use AFEC also directly with Altera's PCI and PCI Express cores without an embedded soft processor. AFEC together with the Ethernet Physical Layer (PHY) forms an Ethernet Network Interface Controller (NIC). See Figure 2.

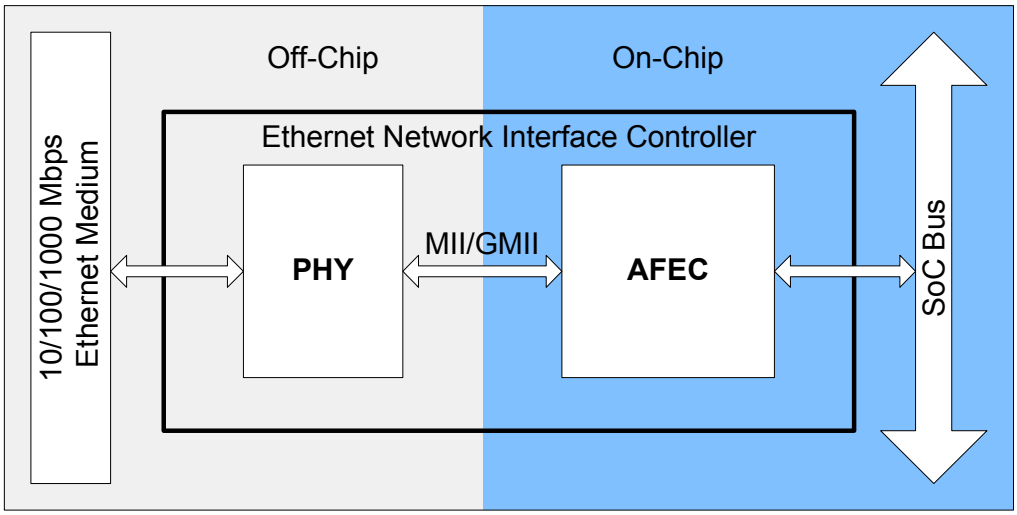

Figure 2 AFEC Overview

2.1 Media Access Control (MAC)

The Media Access Control (MAC) protocol is used to provide the data link layer of Ethernet protocol. The MAC protocol encapsulates data by adding a 14 byte header before the payload and a 32 bit Cyclic Redundancy Check (CRC) checksum after the payload. In addition to this, there is a 7 byte preamble and a 1 byte Start Frame Delimiter (SFD) before the header. See Figure 2.

In case of Virtual LAN (VLAN) tagging being used, the header will be 4 bytes longer because of an additional type field and a VLAN tag. This also increases the Ethernet frame's maximum length from 1518 bytes to 1522 bytes (without preamble and SFD).

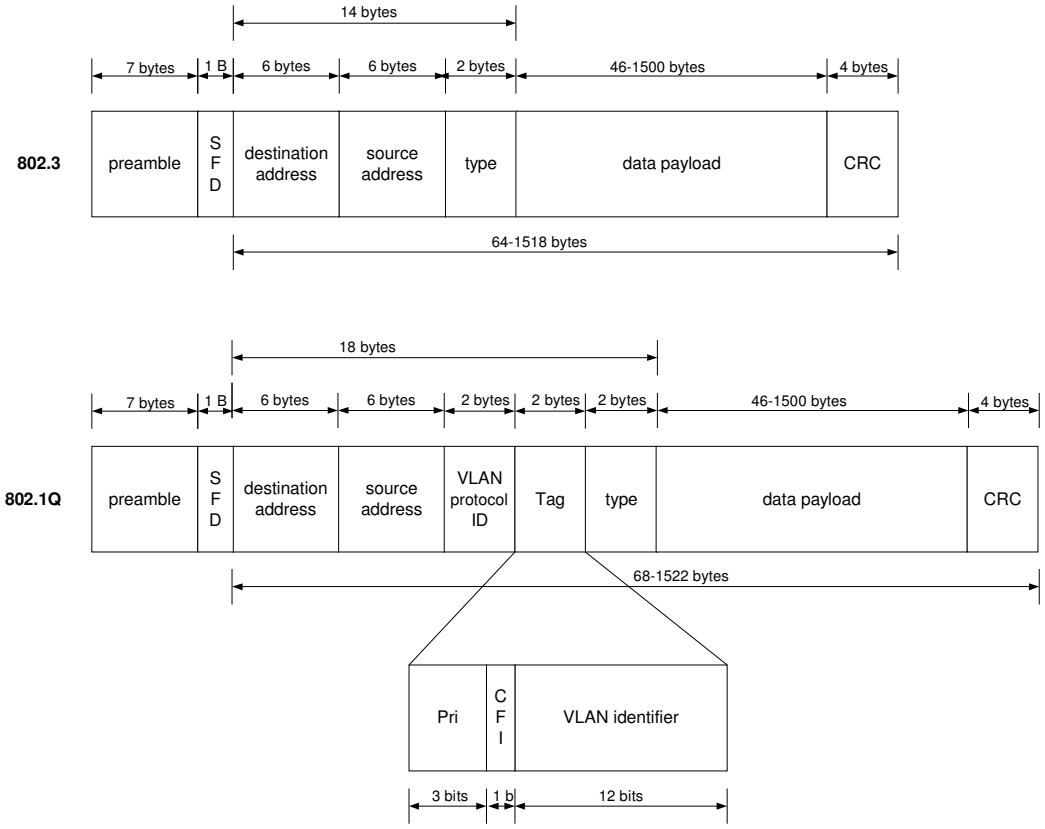

Figure 3 Ethernet Frame

2.1.1 Preamble and Start Frame Delimiter (SFD)

Before the preamble there is a small idle time after the previous frame. After that, a node starts its transmission by sending the preamble sequence which consists of 56 alternating 1's and 0's. The purpose of the preamble is to synchronize the receivers on the network before actual data arrives. After the preamble comes the Start Frame Delimiter (“10101011”) which indicates that a valid frame is to begin.

2.1.2 Header

The MAC header consists of three parts:

- Destination Address (6 bytes)
  The Destination Address specifies also whether the recipient is a single node (unicast), a group of nodes (multicast), or all the nodes at the medium (broadcast).
- Source address (6 bytes)
- Type field (2bytes)
  The type field indicates the protocol being carried. In the case of IEEE 802.3 [1] LLC, this field may also be used to indicate the length of the data.

Ethernet nodes are usually capable of choosing whether they want to receive only frames sent to their own Ethernet address and the broadcast address, or whether they want to receive also frames sent to some or all of the multicast addresses. The nodes may also

choose to receive all the frames, including those destined for other nodes. This is called the promiscuous mode.

2.1.3 CRC

The 32 bits long CRC checksum at the end of the frame provides an error detection mechanism. The CRC checksum is calculated and added to the frame when the frame is sent, and it is checked when the frame is received at the opposite end. Frames with invalid CRC checksums contain an error and should be discarded.

2.1.4 Shared Media

An Ethernet network can be used to provide shared access to a group of network nodes. In this case, the nodes are said to form a Collision Domain. Here, all frames sent on the medium can be seen by all the receivers; however, the destination address in the MAC header ensures that only the specified destination receives the frame. The shared medium allows any node to transmit whenever it wishes, but if two nodes start to transmit at the same time, the frames will collide.

2.1.5 Carrier Sense Multiple Access (CSMA)

To control which nodes are allowed to transmit at a given time in a shared medium, a protocol is needed. Ethernet uses a protocol known as Carrier Sense Multiple Access (CSMA).

With CSMA, when a node has data to transmit, it first listens to the medium to see if some other node is currently transmitting. If the node sees that no other node is transmitting, it may start its transmission. If the physical medium is not idle, it has to wait for the currently transmitting node to stop its transmission.

However, this alone is unable to prevent two nodes from transmitting at the same time. If two nodes try to transmit at the same moment they will both detect that the physical medium is idle. So, they both will then start their transmission at the same time and the frames collide. The collision corrupts the frames being sent. Therefore, Ethernet has Collision Detection (CD).

2.1.6 Collision Detection (CD)

Collision Detection is used to detect collisions caused by two nodes transmitting at the same time. While transmitting, each node also monitors its own transmission concurrently. If the node observes a collision, it stops the transmission and transmits a 32 bits long jam sequence whose purpose is to make it completely sure that the other transmitting node has also detected that a collision has happened. After the transmitting nodes have transmitted their jam sequences, the medium becomes idle, and they may try to transmit again.

To minimize the possibility of the retransmissions to collide, the nodes wait for a random time (back-off time) before trying to retransmit. If also the retransmission leads to a collision, the back-off time is increased, and retransmission is attempted again. Further collisions lead to increase in the back-off time until the maximum back-off time is reached.

Ethernet limits the maximum number of retransmissions of a single frame to 16. After 16 attempts, the transmitter gives up and discards the frame. In practice, this should only happen in highly congested networks.

2.1.7 Late Collision

A normal collision happens when two or more nodes start to transmit at the same time. A Late Collision happens when a node starts to transmit while another node is already transmitting. Late Collisions may be the result of a faulty hardware or an error in configuration (node in Full-Duplex mode when it should be Half-Duplex). Also Ethernet segments whose length is over the specified maximum (100m) cause Late Collisions in Half-Duplex domains.

Note that a Late Collision is an error, whereas a normal collision is not.

2.1.8 Duplex Mode

Carrier Sense Multiple Access (CSMA) and Collision Detection (CD) are used only in Half-Duplex mode. In Half-Duplex mode a node cannot transmit and receive at the same time because the transmitted frame would collide with the received frame. However, in Full-Duplex mode, a node can send and receive at the same time.

There are some requirements which have to be met before Full-Duplex mode can be used. Firstly, the medium has to have independent receive and transmit data paths that can operate simultaneously. Secondly, there has to be a point-to-point link between two stations, and both of the stations have to be capable of and configured to Full-Duplex operation. If these requirements are met, the frames can not collide, and Full-Duplex mode can be used.

2.1.9 Media Independent Interface (MII/GMII)

The Media Independent Interface (MII) is the interface between MAC and the PHY (See Figure 1). The MII interface can exist in different forms: For example, it can act as a connector between two devices. Or, it can be just signals between two chips on a circuit board. The third possibility is that the MII interface exists only inside a single chip. The idea of the MII interface is that it is independent of the physical medium. In practise this means that the same Ethernet MAC can be used with various kinds of Ethernet media. The MII supports both 10 Mbps and 100 Mbps transfer rates. For gigabit media there is the GMII (Gigabit Media Independent Interface).

The difference between MII and GMII is that GMII has eight bits wide data interface while MII has only four bits wide. The clock and data speeds are also higher in GMII (GMII 125MHz, MII 25MHz/2.5MHz) and in GMII the transmit clock is supplied to the PHY whereas in MII the transmit clock supplied by the PHY.

2.2 Altera Avalon

Avalon is Altera's solution for a System-On-Chip interface. Although it is often referred as a bus, Avalon is actually a switch fabric, not a bus. The difference between bus and switch fabric architectures is that in a bus architecture only one master can be using the bus at one time, whereas in switch fabric architecture several masters can be accessing several slaves at the same time, as long as every master is accessing different slave. This makes the performance of the switch fabric architecture significantly better in situations when there are several masters and several slaves.

Avalon is an open interface standard. No licenses are required to produce and distribute custom peripherals employing Avalon interfaces.

2.3 Clock Synchronization

Precision Time Protocol (PTP) for network node synchronization is defined in IEEE standard 1588 [3]. The standard defines a protocol enabling precise synchronization of clocks in packet based networks, for example Ethernet. The devices are automatically synchronized to the

most accurate clock in the network. The protocol supports system wide synchronization accuracy in the sub microsecond range with minimal network and local clock computing resources. The protocol is used for example by test and measurement, power-line management and industrial automation applications.

Typically IP network architectures are hierarchal. According to IEEE 1588 specification, time error accumulates at each level of the network hierarchy and when the number of levels increases, timing accuracy decreases. A number of different application areas such as industrial automation, telecommunication, semiconductor manufacturing, military system, and utility power generation have emerged requirement for the standard to be revised.

2.4 AFEC Features

AFEC main features:

- Compatible with IEEE standard 802.3 [1]
- Half-Duplex and Full-Duplex operating modes at speeds of 10 Mbps and 100 Mbps
- Full-Duplex operating mode at the speed of 1000 Mbps
- Media Independent Interface (MII) and Gigabit Media Independent Interface (GMII) for attaching to an external Physical Layer device (PHY)
- PHY Management Data Input/Output (MDIO) interface controller
- Register interface for accessing control and status registers
- Bus master capable DMA over the SoC bus, enabling high transfer speed and low processor load
- Interrupt generation for signaling the completion of reception and transmission
- Support for delayed interrupts for lightening the interrupt burden of the CPU
- Packet time stamping for supporting IEEE 1588 Precision Time Protocol version 2
- Altera Avalon SoC bus interfaces
    - 64-bit Avalon master for receiver
    - 64-bit Avalon master for transmitter
    - 64-bit Avalon slave for user accessible registers
- Supported FPGA families:
    - Altera Arria GX
    - Altera Cyclone II
    - Altera Cyclone III
    - Altera Stratix II
    - Altera Stratix II GX
    - Altera Stratix III

3 External Signal Descriptions

All AFEC external signals are presented in Figure 3. The external signals consist of MII/GMII signals, MDIO signals, SoC Bus signals, Time Interface signals, General Signals, Frame Handling signals and Symbols alignment. The SoC bus signals in AFEC are those of Altera Avalon.

In some configurations AFEC may include Extension Interface (EIF), which may be used to connect application specific blocks into the RX and TX paths. If EIF is used, additional specification is provided for EIF usage.

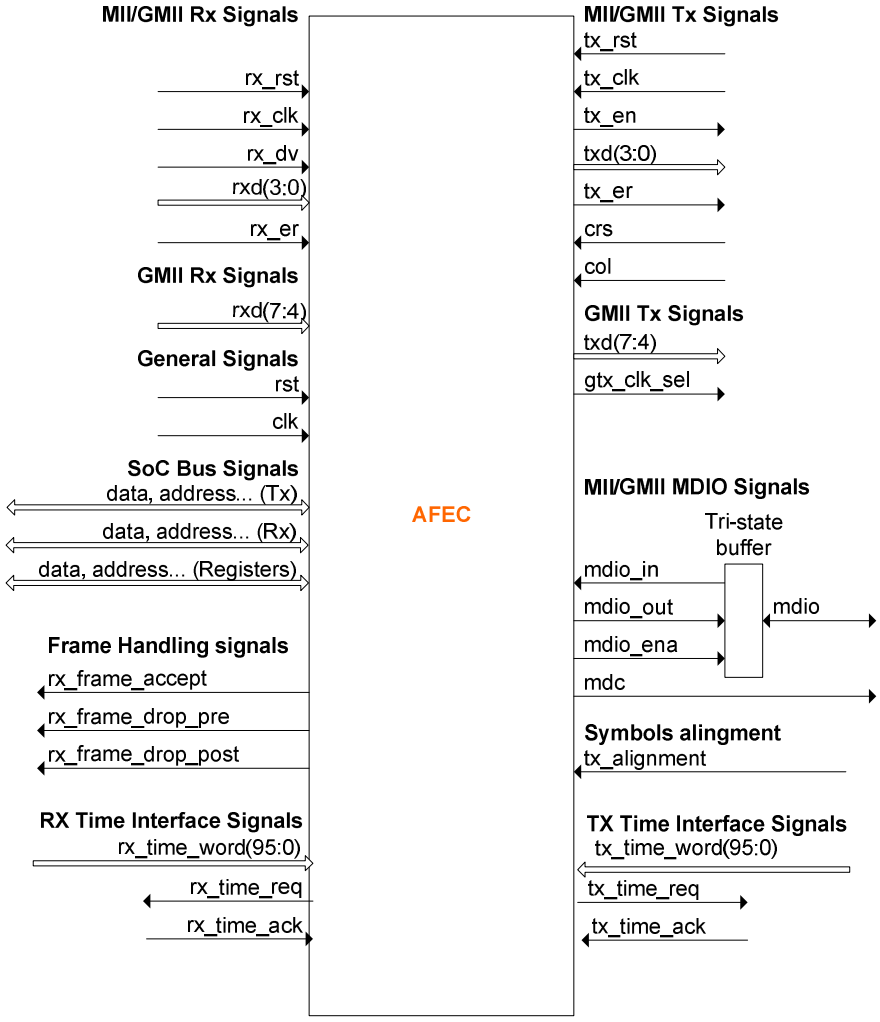

Figure 4 AFEC External Signals

The tri-state buffer for MDIO is external to AFEC (see Figure 3), because tri-state signals are normally not used inside an FPGA chip. The tri-state buffer is used only when the MDIO signal is wired outside of the chip and it is not used if the signal is wired to another block inside the FPGA.

3.1 General Signals

The General Signals consist of global reset and clock signals presented in Table 1.

Name	Direction	Description
rst	Input	Reset Global asynchronous signal that resets all the flip-flops in AFEC to their initial values. This signal is active high.
clk	Input	Main Clock This signal is the AFEC main clock and its frequency must be greater than or equal to 62.5 MHz. Only the rising edge of the signal is used. This signal should be connected to a low skew clock buffer. If there is a shortage of low skew clock buffers at the FPGA, this signal should have priority over the rx_clk and tx_clk signals.

Table 1 General Signals

3.2 MII/GMII Signals

The Media Independent Interface (MII) and Gigabit Media Independent Interface (GMII) signals of AFEC are presented in Table 2. The MII/GMII signals consist of the MII/GMII Rx, MII/GMII Tx and MDIO signals. The MDIO signals are synchronous to the clk signal; MII/GMII Rx signals are synchronous to rx_clk; and MII/GMII Tx signals, including the crs and col signals, are synchronous to tx_clk. In Table 2, unless otherwise stated, all the signals that can activate or enable something are active when their state is high and inactive when their state is low. For further information about the MII interface see IEEE Standard 802.3 specification [1].

Reconciliation Sublayer (RS):		
Name	**Direction**	**Description**
rx_clk	Input	MII/GMII Receive Clock The frequency is 2.5 MHz in the 10 Mbps mode, 25 MHz in the 100 Mbps mode and 125MHz in 1000 Mbps mode. It is allowed that the clock is not running continuously. Discontinuous receive clock does not cause malfunction in AFEC. This signal should be connected to a low skew clock buffer.
rx_dv	Input	MII/GMII Receive Data Valid Indicates when the data lines are carrying valid data from the PHY.
rxd(3:0)	Input	MII/GMII Receive Data Data lines from the PHY.
rxd(7:4)	Input	GMII Receive Data Data lines from the PHY.
rx_er	Input	MII/GMII Receive Error Indicates that an error has been detected while receiving a frame.
tx_clk	Input	MII/GMII Transmit Clock The frequency is 2.5 MHz in the 10 Mbps mode, 25 MHz in the 100 Mbps mode and 125MHz in 1000 Mbps mode. It is allowed that the clock is not running continuously. Discontinuous transmit clock does not cause malfunction in AFEC. This signal should be connected to a low skew clock buffer. Note that although GMII is source synchronous, tx_clk is always input clock to AFEC as presented in Figure 6.
tx_en	Output	MII/GMII Transmit Enable Indicates when the data lines are carrying valid data to the PHY.
txd(3:0)	Output	MII/GMII Transmit Data Data lines to the PHY.
txd(7:4)	Output	GMII Transmit Data Data lines to the PHY.
tx_er	Output	MII/GMII Transmit Error Indicates that an error has occurred while transmitting the frame.
crs	Input	MII Carrier Sense Indicates that the medium is busy. This signal is used in Half-Duplex mode only.
col	Input	MII Collision Detection Indicates that a collision is detected on the medium. This signal is used in Half-Duplex mode only.
gtx_clk_sel	Output	GMII Transmit Clock Select Can be used to controlling an external clock multiplexer to select the transmit clock. See Figure 6.
Station Management (STA):		
Name	**Direction**	**Description**
mdio_in	Input	Management Data Input Serial data input.
mdio_out	Output	Management Data Output Serial data output.
mdio_ena	Output	Management Data Output Enable This signal is a tri-state enable signal for the Management Data Output signal.
mdc	Output	Management Data Clock The frequency is clk / 64. The IEEE Std 802.3 specification requires that the frequency must be lower than or equal to 2.5MHz. This requirement is met when clk is lower than or equal to 160MHz. The clock runs only when data is being transferred.

Table 2 MII/GMII Signals

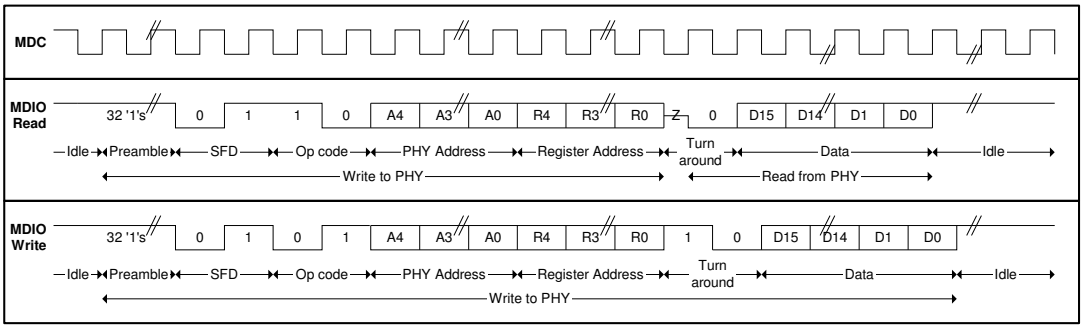

Figure 5 MDIO Management Interface Timing Diagram

3.2.1 Transmit Clock Multiplexer

In gigabit (GMII) mode the receive and transmit clock speeds are faster, and the direction of the transmit clock signal to PHY is different than in MII mode. For this reason, and also because it is reasonable to generate the high-speed GMII transmit clock (gtx_clk) outside of AFEC, an external clock multiplexer is employed. The connection of the clock multiplexer (clk_mux) with AFEC is presented in Figure 6.

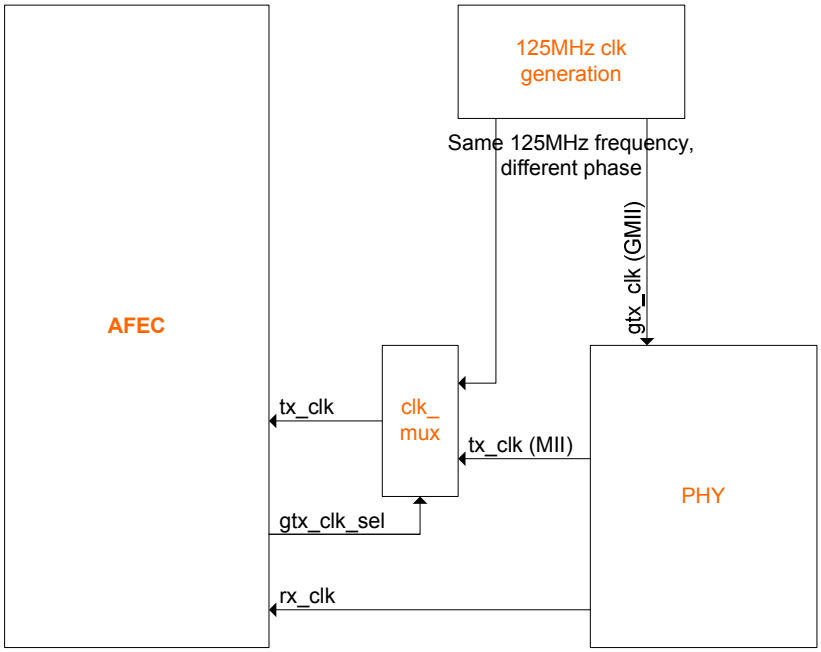

Figure 6 External Transmit Clock Multiplexer

3.3 SoC Bus Signals

AFEC supports Altera Avalon as the SoC bus. Unless otherwise stated, the SoC bus signals are synchronous to the clk signal.

3.3.1 Altera Avalon

AFEC has separate Avalon interfaces for Transmitter, Receiver and user accessible registers. Transmitter and Receiver interfaces are Avalon masters, and the user registers are accessed through an Avalon slave interface.

The Altera Avalon bus master port signals for Transmitter are presented in Table 3 and for Receiver in Table 4. The Avalon slave interface signals for user accessible registers are presented in Table 5.

Name	Direction	Description
tx_m_address(31:0)	Output	Address Address lines to the bus.(Byte address) The MSB is carried by bit 31 and the LSB is carried by bit 0.
tx_m_byteenable(7:0)	Output	Byte Enable Byte enable for read data. Always enabled.
tx_m_read	Output	Read Request Read request to a slave device.
tx_m_readdata(63:0)	Input	Read Data DMA read data from the bus to AFEC. The MSB is carried by bit 63 and the LSB is carried by bit 0.
tx_m_readdata_vld	Input	Read Data Valid **This signal is optional.** Used in pipelined read transfers with variable latency. Marks the rising clock edge when the slave asserts valid read data.
tx_m_waitrequest	Input	Wait Request Wait request from a slave device.

Table 3 Altera Avalon Bus Master Port Signals (Transmitter)

Name	Direction	Description
rx_m_address(31:0)	Output	Address Address lines to the bus. (Byte address) The MSB is carried by bit 31 and the LSB is carried by bit 0.
rx_m_byteenable(7:0)	Output	Byte Enable Byte enable for write data.
rx_m_write	Output	Write Request Write request to a slave device.
rx_m_writedata(63:0)	Output	Write Data DMA write data from AFEC to the bus. The MSB is carried by bit 63 and the LSB is carried by bit 0.
rx_m_waitrequest	Input	Wait Request Wait request from a slave device.

Table 4 Altera Avalon Bus Master Port Signals (Receiver)

Name	Direction	Description
s_cs	Input	`Chip Select` Chip select signal to AFEC.
s_address(13:0)	Input	`Address` Address lines from the bus. Note that this is word address, not byte address. The word is 64 bits wide. The MSB is carried by bit 13 and the LSB is carried by bit 0.
s_byteenable(7:0)	Input	`Byte Enable` Byte enable for read and write data.
s_read	Input	`Read Request` Read request to AFEC.
s_readdata(63:0)	Output	`Read Data` Data lines to the bus. The MSB is carried by bit 63 and the LSB is carried by bit 0.
s_write	Input	`Write Request` Write request to AFEC.
s_writedata(63:0)	Input	`Write Data` Data lines from the bus. The MSB is carried by bit 63 and the LSB is carried by bit 0.
s_waitrequest	Output	`Wait Request` AFEC asserts this signal to request a wait cycle on the bus.
irq	Output	`Interrupt Request` Interrupt request from AFEC. Level sensitive, active high.

Table 5 Altera Avalon Bus Slave Port Signals (User Registers)

For further information about the Altera Avalon bus signals see the Altera Avalon Bus Specification [2].

3.4 Time Interface Signals

Time Interface consists of signals used for time information exchange between AFEC and other FPGA blocks. The handling of the time information is left outside of AFEC because its implementation may vary significantly depending on the hardware. Also several AFECs can be used in a design and there does not have to be more than one clock entity.

The clock time information from the Time Interface `clock_time` –signals is used in time stamping the transmitted and received packets. The exact receive and transmit time information is essential for IEEE 1588 Precision Time Protocol to achieve its best accuracy.

Name	Direction	Description
tx_alignment	Input	Transmit Alignment Control This signal can be used to adjust transmit operation start timing by on clock cycle to achieve optimal symbol alignment when using fiber connection. **If used**, this signal shall toggle on each clock cycle. **If not used**, this signal shall be tied to '0'. This signal is synchronous to the MII/GMII interface transmit clock.
rx_time_req	Output	Receive Time Request When a frame is received, at the moment when the Frame Timestamp Point (see Figure 17) is in MII/GMII interface, this signal toggles. This signal is synchronous to the MII/GMII interface receive clock. If time interface is not used, this signal shall be tied to rx_time_ack.
rx_time_ack	Input	Receive Time Acknowledge This signal shall toggle to follow the rx_time_req signal when data is updated on the rx_time_word(95:0) signal. This signal is synchronous to the MII/GMII interface receive clock. If time interface is not used, this signal shall be tied to rx_time_req.
rx_time_word(95:0)	Input	Receive Time Word Clock time input to AFEC. This signal shall be updated when the rx_time_req signal has opposite state to the rx_time_ack signal. When a frame is received the clock time carried by this signal is used to update the C, D and E descriptors of the first fragment of the frame (in descriptor chain 0). This signal is synchronous to the MII/GMII interface receive clock. If time interface is not used, this signal shall be tied to zero.
tx_time_req	Output	Transmit Time Request This signal toggles to request up to date data used in transmit time stamping operation. This signal is synchronous to the MII/GMII interface transmit clock. If time interface is not used, this signal shall be tied to tx_time_ack.
tx_time_ack	Input	Transmit Time Acknowledge This signal shall toggle to follow the tx_time_req signal when data is updated on the tx_time_word(95:0) signal. This signal is synchronous to the MII/GMII interface transmit clock. If time interface is not used, this signal shall be tied to tx_time_req.
tx_time_word(95:0)	Input	Transmit Time Word Clock time input to AFEC. This signal shall be updated when the tx_time_req signal has opposite state to the tx_time_ack signal. When a frame is transmitted the clock time carried by this signal is used to update the C, D and E descriptors of the first fragment of the frame (in descriptor chain 0). This signal is synchronous to the MII/GMII interface transmit clock. If time interface is not used, this signal shall be tied to zero.

Table 6 Time Interface Signals

4 Memory Maps

4.1 Transmission and Reception of a Frame

AFEC transmits and receives the data frames (packets) directly from/to the system memory by using Direct Memory Access (DMA). Where to in the system memory the received frames are written and where from the transmitted frames are fetched, is determined by using receive and Transmit Descriptors. See register map in Figure 7.

The Receiver receives frames to the memory areas pointed by Receive Descriptors. Each Receive Descriptor points to one single continuous memory area. When receiving a frame, the receiver consumes one or more memory areas pointed by descriptors. The frame splits to as many fragments in the memory, as the number of descriptors consumed. The head of the frame (the first fragment) is always received to memory area pointed by a descriptor in descriptor chain 0, and all the following fragments (if any) are received to memory areas pointed by descriptors in chain 1. How many descriptors are consumed during the reception depends on the length of the received frame, and the lengths of the memory areas pointed by the descriptors.

Figure 8, Figure 9 and Figure 10 present situations where three consecutive frames are received after system reset (counter in the hardware start from the first descriptor). The first frame presented in Figure 8 fits into the memory area pointed by the first Receive Descriptor in chain 0. The second frame in presented Figure 9 consumes two descriptors and the third frame presented in Figure 10 consumes three descriptors and memory areas pointed by the descriptors.

How the Receive Descriptors are used is presented in more details in Chapter 5.3

The Transmitter transmits frames from the memory areas pointed by Transmit Descriptors. Each Transmit Descriptor points to one single continuous memory area. A frame to be transmitted may consist of one or more fragments, and as many Transmit Descriptors. The first Transmit Descriptor, pointing to the head of the frame, is always in Transmit Descriptor chain 0 and the rest of the descriptors are in chain 1. All the fragments can be of different size.

Figure 11, Figure 12 and Figure 13 present situations where three consecutive frames are transmitted after system reset (counter in the hardware start from the first descriptor). The first frame in Figure 11 consists of one single fragment, pointed by Transmit Descriptor in chain 0. The second frame in Figure 12 consists of two fragments and the thirds frame in Figure 13 consists of three fragments.

How the Transmit Descriptors are used is presented in more details in Chapter 5.2

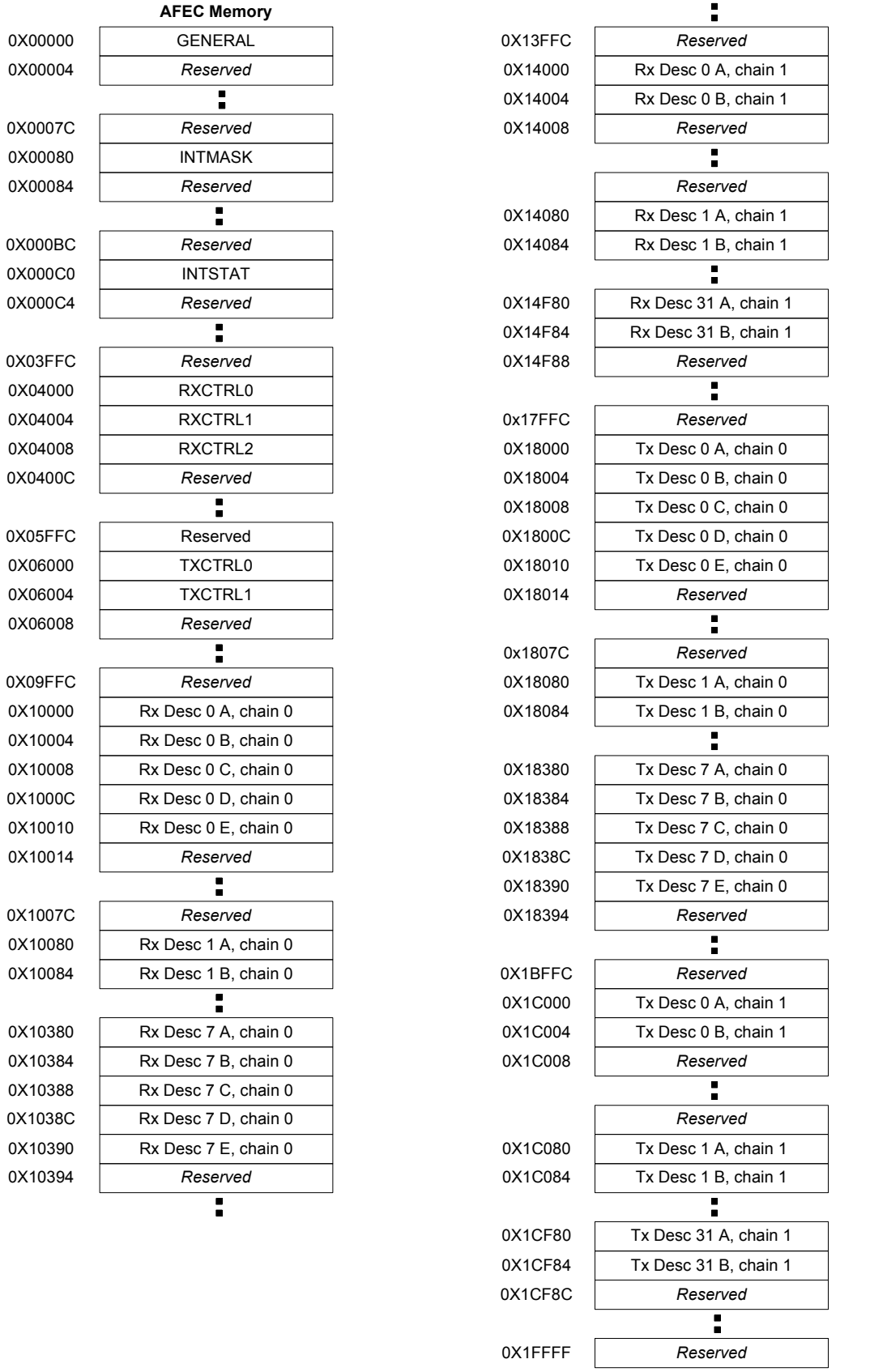

Figure 7 AFEC Register Map

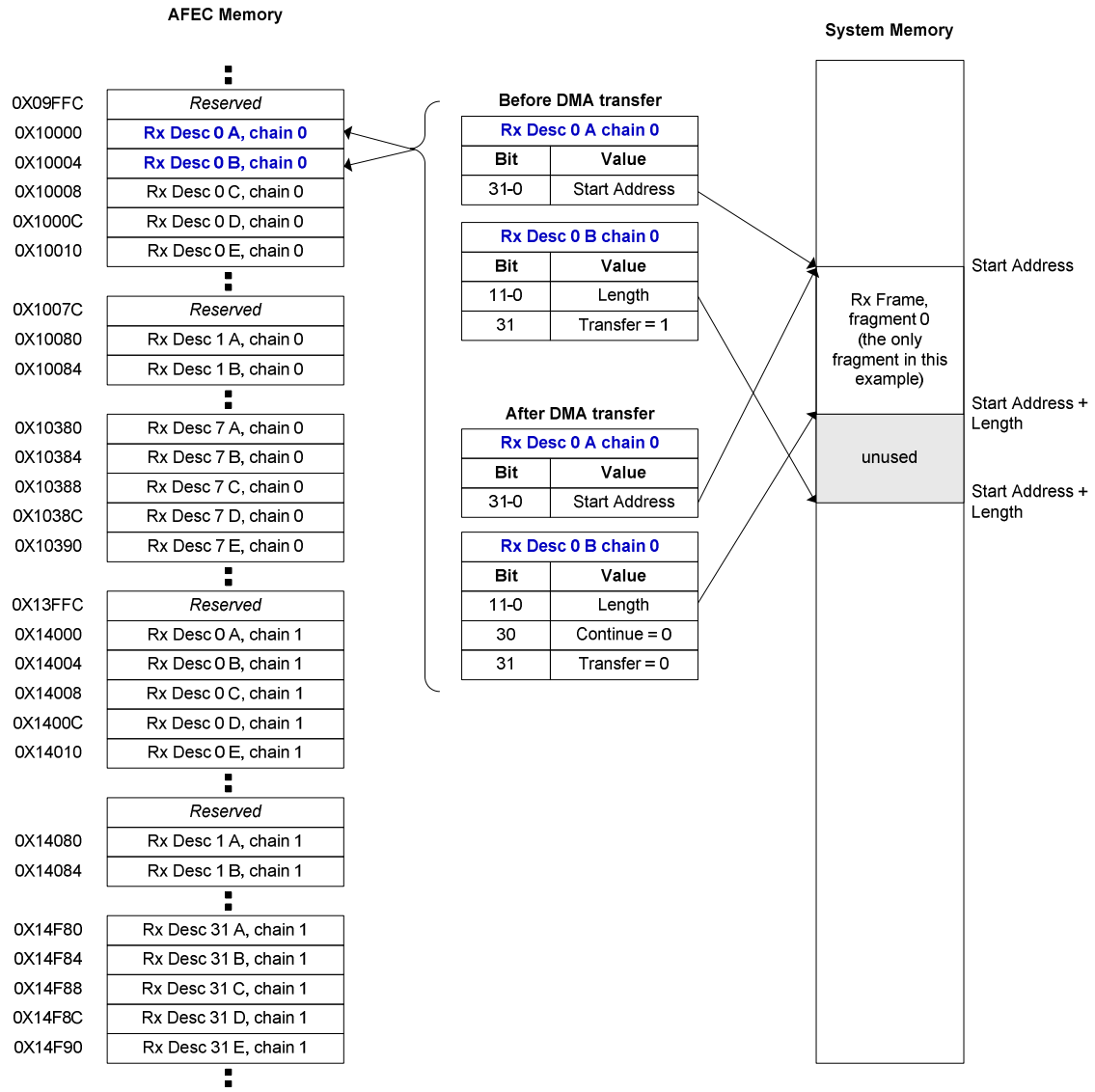

Figure 8 Rx Frame Consisting of One Fragment

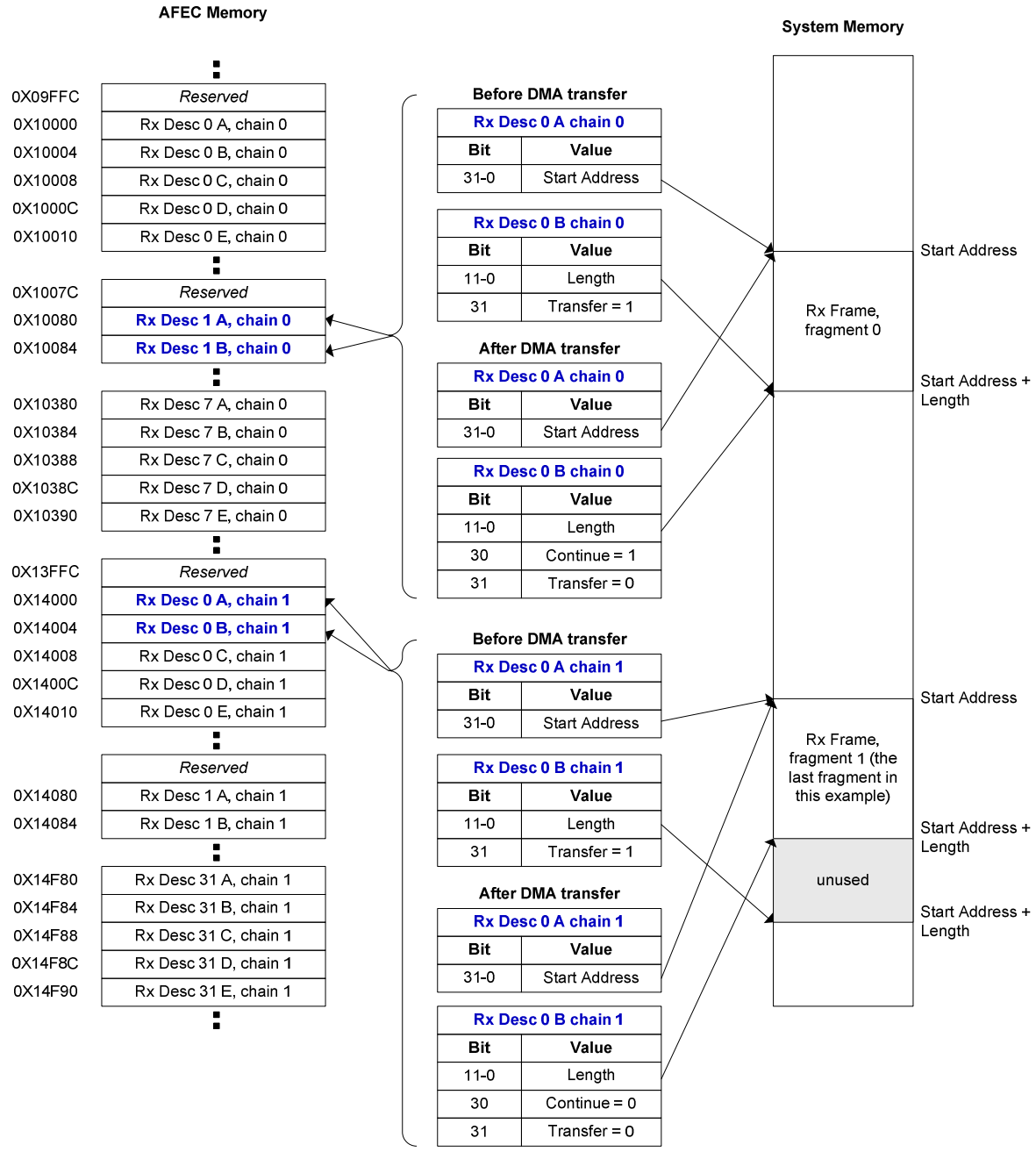

Figure 9 Rx Frame Consisting of Two Fragments

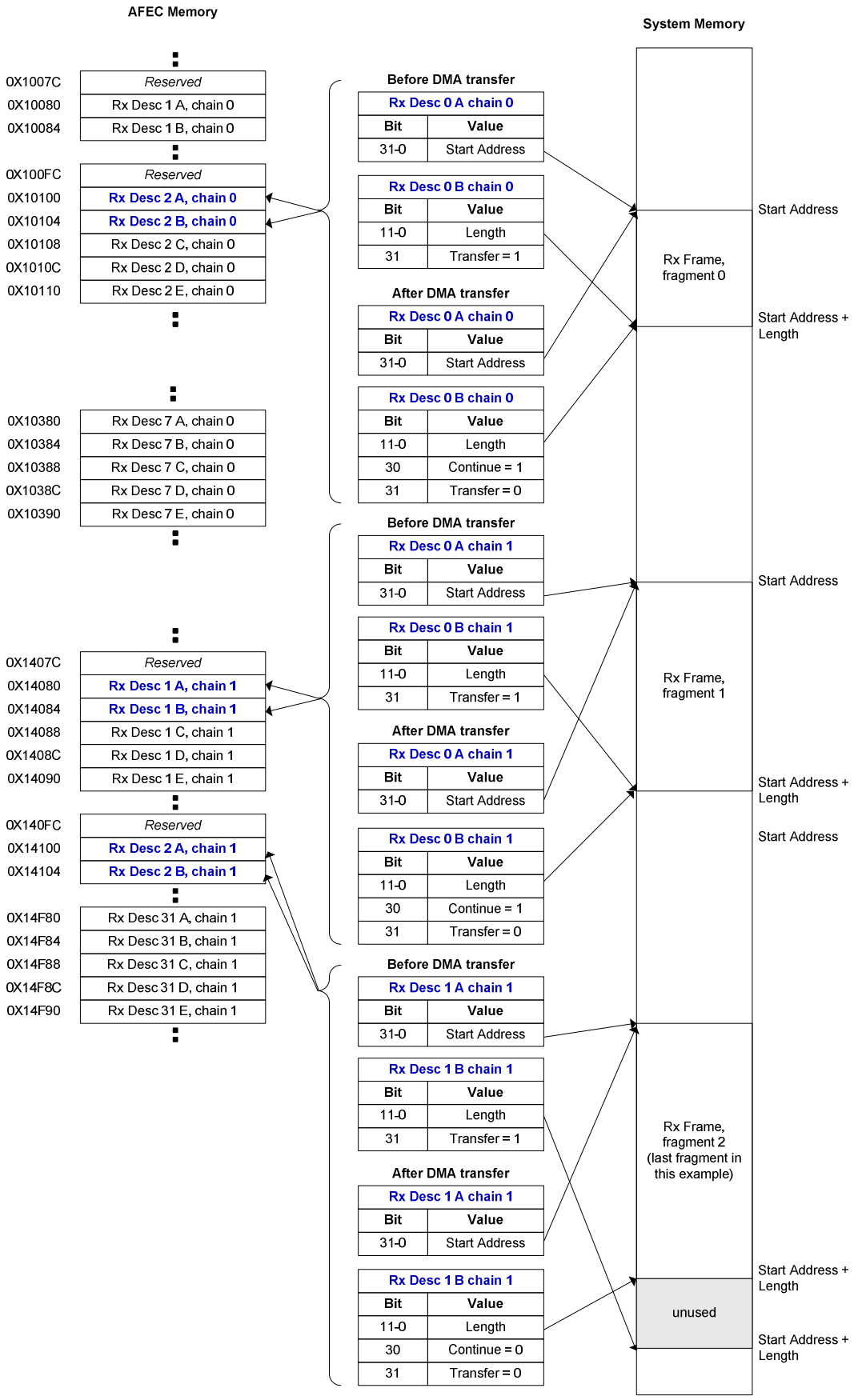

Figure 10 Rx Frame Consisting of Three Fragments

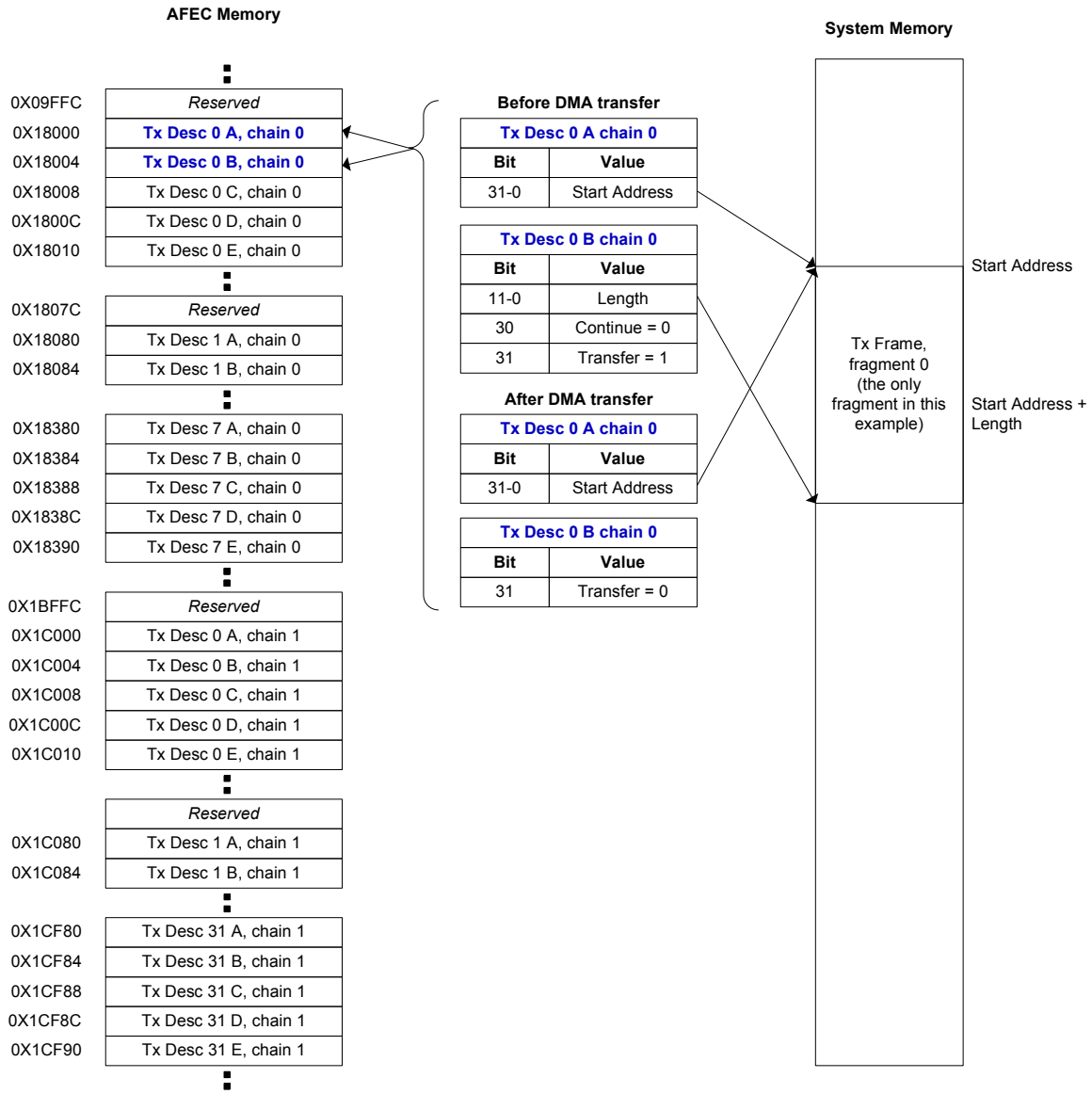

Figure 11 Tx frame consisting of one fragment

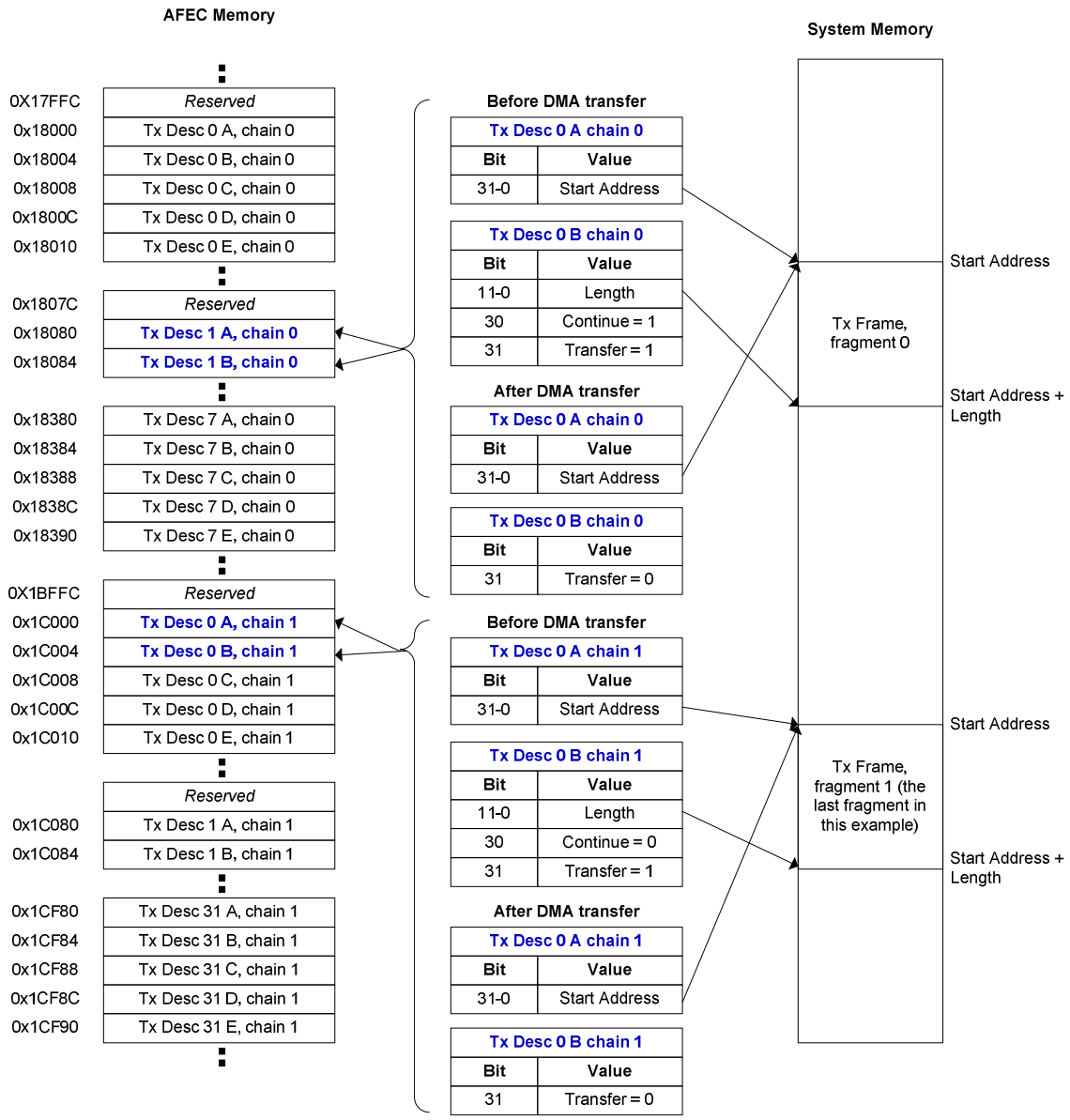

Figure 12 Tx Frame Consisting of Two Fragments

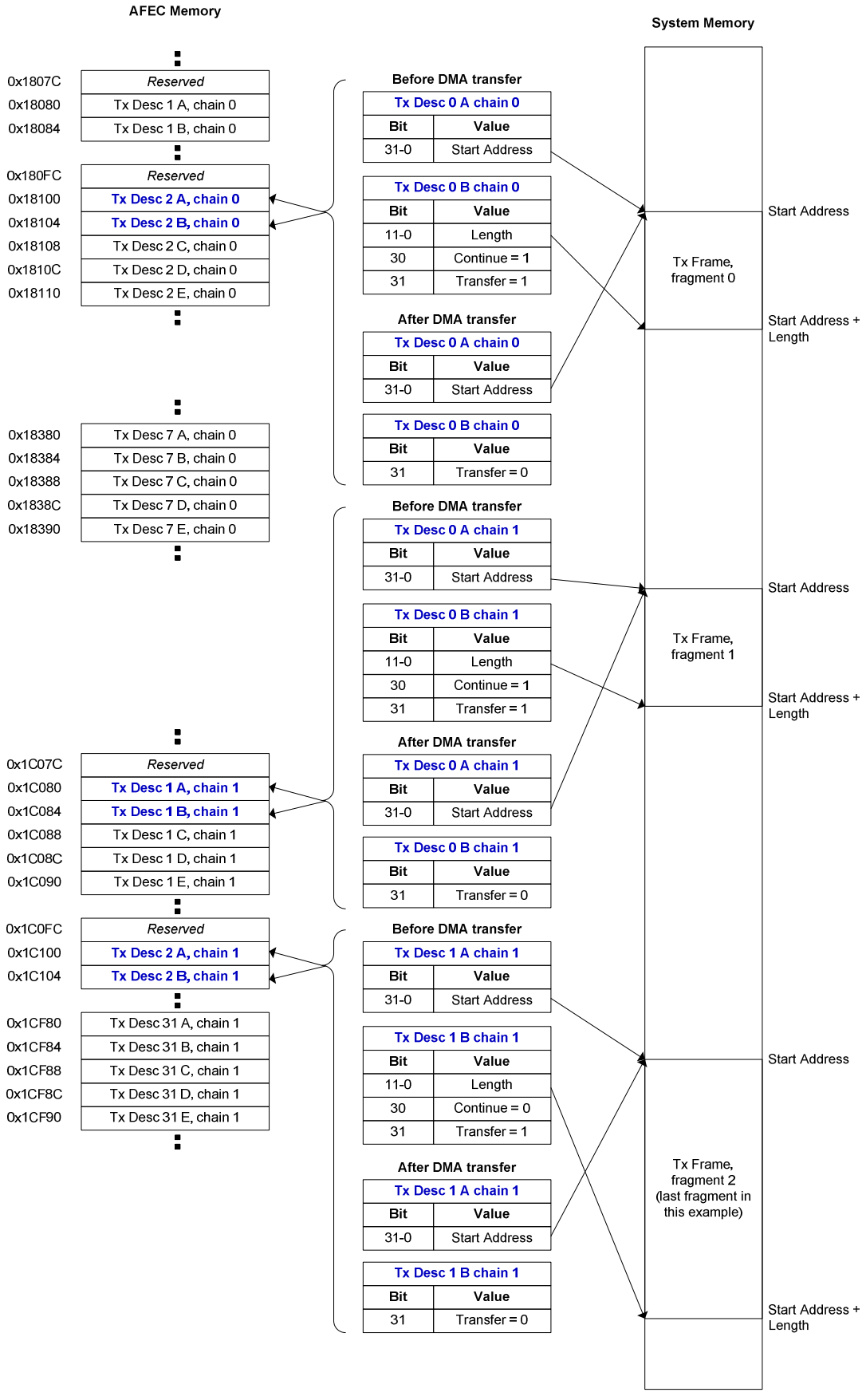

Figure 13 Tx Frame Consisting of Three Fragments

4.2 Registers

The internal registers of AFEC are presented in Table 7. Unless otherwise stated, all the bits that can activate or enable something are active when their value is 1 and inactive when their value is 0. The explanation of the bit types is the following:

- RO = Read capable Only: The bits marked with RO can be read. Writing to these bits is allowed if not otherwise stated. If writing is allowed, it does not affect the value of the bit.

- R/W = Read and Write capable. The bits can be read and written to. Writing 1 onto the bit makes its value 1; writing 0 to the bit makes its value 0.

- R/S = Read and Set capable. The bits can be read and set. Writing 1 to the bit makes its value 1; writing 0 does nothing.

- R/C = Read and Clear capable. The bits can be read and cleared. Writing 0 to the bit makes its value 0; writing 1 does nothing.

- R/SC = Self Clear. The bits can be read and set. Writing 0 to the bit does nothing; writing 1 to the bit makes its value 1 for a moment, after which the value automatically returns back to 0.

The bits marked as *Reserved* should not be written to anything other than 0, even if they are marked as *Read capable Only (RO)*, because their function may change in future versions.

The register locations in Table 7 are relative to the AFEC register base address. Note that not every descriptor is shown in Table 7, because the number of descriptors is quite big and because the descriptor registers in the same chain are alike.

In the register map the descriptor:

- Rx Desc X A in chain 0 is located in address 0x10000 + X * 0x80

- Rx Desc X B in chain 0 is located in address 0x10004 + X * 0x80

- Rx Desc X C in chain 0 is located in address 0x10008 + X * 0x80

- Rx Desc X D in chain 0 is located in address 0x1000C + X * 0x80

- Rx Desc X E in chain 0 is located in address 0x10010 + X * 0x80

- Tx Desc X A in chain 0 is located in address 0x18000 + X * 0x80

- Tx Desc X B in chain 0 is located in address 0x18004 + X * 0x80

- Tx Desc X C in chain 0 is located in address 0x18008 + X * 0x80

- Tx Desc X D in chain 0 is located in address 0x1800C + X * 0x80

- Tx Desc X E in chain 0 is located in address 0x18010 + X * 0x80

, where X is from 0 to 7.

- Rx Desc X A in chain 1 is located in address 0x14000 + X * 0x80

- Rx Desc X B in chain 1 is located in address 0x14004 + X * 0x80

- Tx Desc X A in chain 1 is located in address 0x1C000 + X * 0x80
- Tx Desc X B in chain 1 is located in address 0x1C004 + X * 0x80

, where X is from 0 to 31.

Address	Register	Description
Register base + 0x00000	GENERAL	<u>General</u> Contains AFEC version information and general control bits. Bits 0-7 : RO Revision ID *ID number that is incremented when HW or SW interface of the IP core has been changed in the release.* Bits 8-23 : RO Device ID *0x4: FEC100 with Altera Avalon bus interface* *0x5: FEC100 with IBM CoreConnect OPB interface* *0x6: FEC1000 with Altera Avalon bus interface* *0x7: FEC1000 with IBM CoreConnect OPB interface* ***0x10: AFEC with Altera Avalon bus interface*** Bits 24-30 : R/W Interrupt Delay *Assertion of the interrupt line is delayed as many clock cycles as the value in these bits * 1024.* ***(fixed to zero in the current version)*** Bit 31 : R/SC Reset *1 = start the software reset. AFEC **clears** this bit when done*
Register base + 0x00004	RESERVED	<u>Reserved</u> On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x00008	CFG_ID	<u>Configuration ID</u> On reset: 0 x 00 00 00 00 Contains component ID numbers. Bits 0-15 : RO Configuration ID number *1 = AFEC default config (cfg001)* *2 = AFEC with AES GCM (cfg002)* Bits 16-31 : RO Version control system ID number of the configuration *1-65535 = SVN rev number of source codes that have been used by a release script to generate the configuration.*
Register base + 0x00010	SUB_ID	<u>Subversion ID</u> On reset: 0 x 00 00 00 00 Contains component ID numbers. Bits 0-15 : RO Version control system ID number the component body *1-65535 = SVN rev number of source codes that have been used by a release script to generate the **component body**.* *Bits 16-31 : RO* Reserved
Register base + 0x00014-0x0001C	RESERVED	<u>Reserved</u> On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x00020	DEBUG	<u>Debug</u> On reset: 0 x 00 00 00 00 Contains component ID numbers. Bits 0-31 : RO EIF init request source status
Register base + 0x00024-0x0007C	RESERVED	<u>Reserved</u> On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x00080	INTMASK	<u>Interrupt Mask</u> On reset: 0 x 00 00 00 00 Contains the interrupt control and status bits. An external interrupt is activated when at least one of the following Interrupt Mask - bits is set and the corresponding Interrupt Status bit is 1. If delayed interrupts are used (Interrupt Delay in GENERAL register is non-zero) the activation of the external interrupt line is delayed, but the status bits are updated immediately. Bit 0 : R/W Frame Received *Indicates that a frame has been received and transferred to the system memory.* Bit 1 : R/W Frame Transmitted *Indicates that a frame has been transmitted.* Bit 2 : R/W Receive FIFO Error *Indicates that a receive FIFO overrun has occurred.* Bit 3 : R/W Transmit FIFO Error *Indicates that a transmit FIFO underrun has occurred.* Bit 4 : R/W Receive Frame Dropped *An incoming frame had to be dropped because the Receive Descriptor was not initialized.* Bit 5 : R/W Transmit Error

		Half duplex mode has been selected and a frame has been dropped because of too many collisions. Bit 6 : R/W MDIO Done *Indicates that MDIO data transfer has been done.* Bit 7 : R/W MDIO Error *The PHY was accessed by AFEC but the PHY did not drive its MDIO signal to LOW during turn-around cycle.* Bits 8-30 : RO *Reserved* Bit 31 : R/W EIF init req *Indicates that EIF init request has been asserted.*
Register base + 0x00084-0x000BC	RESERVED	Reserved On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x000C0	INTSTAT	Interrupt Status On reset: 0 x 00 00 00 00 Contains the interrupt control and status bits. An external interrupt is activated when at least one of the following Interrupt Status - bits is set and the corresponding Interrupt Mask bit is 1. Bit 0 : R/C Frame Received *Indicates that a frame has been received and transferred to the system memory.* Bit 1 : R/C Frame Transmitted *Indicates that a frame has been transmitted.* Bit 2 : R/C Receive FIFO Error *Indicates that a receive FIFO overrun has occurred.* Bit 3 : R/C Transmit FIFO Error *Indicates that a transmit FIFO underrun has occurred* Bit 4 : R/C Receive Frame Dropped *An incoming frame had to be dropped because the Receive Descriptor was not initialized.* Bit 5 : R/C Transmit Error *Half duplex mode has been selected and a frame has been dropped because of too many collisions.* Bit 6 : R/C MDIO Done *Indicates that MDIO data transfer has been done.* Bit 7 : R/C MDIO Error *The PHY was accessed by AFEC but the PHY did not drive its MDIO signal to LOW during turn-around cycle.* Bits 8-30 : RO *Reserved* Bit 31 : R/C EIF init req *Indicates that EIF init request has been asserted.*
Register base + 0x000C4-0x03FFC	RESERVED	Reserved On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x04000	RXCTRL0	Receive Control 0 On reset: 0 x 00 00 00 02 Contains the bits required to control the receiver. Bits 0-2 : R/W Receive FIFO Threshold *The fill level at which AFEC tries to keep the Receive FIFO.* *0 and 1 => 12.5% ; 2 => 25% ; 3 => 37.5% ;* *4 => 50% ; 5 => 62.5% ; 6 => 75% ; 7 => 87.5%* *The recommended value is 2.* Bits 3-21 : RO *Reserved* Bit 22 : R/W External EIF Disable **(optional / cfg dependent)** *0 = enable ext EIF, 1 = disable ext EIF* Bit 23 : R/W MII Mode *0 = 10 Mbps, 1 = 100 Mbps* Bit 24 : R/W GMII Enable *0 = MII mode, 1 = GMII mode* Bits 25-27 : RO *Reserved* Bit 28 : R/W All Unicast Enable *0 = the only unicast frames received are those destined to the MAC address in registers RXCTRL1 and RXCTRL2* *1 = enables receiving all unicast addressed frames* Bit 29 : R/W All Multicast Enable *1 = enables receiving all the other frames except the unicast or broadcast addressed* Bit 30 : R/W Broadcast Enable *1 = enables receiving broadcast addressed frames* Bit 31 : R/W Receive Enable

		1 = enable the Ethernet data capturing and polling of the Receive Descriptors. All the Receive Descriptors must be initialized before enabling the Receiver. If bits 28-31 all have value 1, the Receiver will receive all the frames from the network. This is generally known as the promiscuous mode.
Register base + 0x04004	RXCTRL1	Receive Control 1 On reset: 0 x 00 00 00 00 Contains the lower part of the MAC address. Bits 0-7 : R/W 6th octet of the MAC address Bits 8-15 : R/W 5th octet of the MAC address Bits 16-23 : R/W 4th octet of the MAC address Bits 24-31 : R/W 3rd octet of the MAC address
Register base + 0x04008	RXCTRL2	Receive Control 2 On reset: 0 x 00 00 00 00 Contains the higher part of the MAC address. Bits 0-7 : R/W 2nd octet of the MAC address Bits 8-15 : R/W 1st octet of the MAC address *bit 8 of the 1st octet is unicast/multicast select and it should be 0 (unicast address)* Bits 16-31 : RO *Reserved*
Register base + 0x0400C-0x05FFC	RESERVED	Reserved On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x06000	TXCTRL0	Transmit Control 0 On reset: 0 x 00 00 04 06 Contains the bits required to control the transmitter. Bits 0-2 : R/W Transmit FIFO Threshold *The fill level at which AFEC tries to keep the Transmit FIFO.* *0 and 1 => 12.5% ; 2 => 25% ; 3 => 37.5%* *4 => 50% ; 5 => 62.5% ; 6 => 75% ; 7 => 87.5%* *The recommended value is 6.* Bits 3-7 : RO *Reserved* Bits 8-10 : R/W Transmit Start Level *Sets the Transmit FIFO fill level which has to be reached before transmission is started.* *0 and 1 => 12.5% ; 2 => 25% ; 3 => 37.5%* *4 => 50% ; 5 => 62.5% ; 6 => 75% ; 7 => 87.5%* *The value must be the same or smaller than the Transmit FIFO Threshold. The recommended value is 4.* Bits 11-21 : RO *Reserved* Bit 22 : R/W External EIF Disable **(optional / cfg dependent)** *0 = enable ext EIF, 1 = disable ext EIF* Bit 23 : R/W MII Mode *0 = 10 Mbps, 1 = 100 Mbps* Bit 24 : R/W GMII Enable *0 = MII mode, 1 = GMII mode* *Connected directly to gtx_clk_sel signal* Bit 25 : R/W Data Overwrite Enable for dow2 operation Bit 26 : R/W Data Overwrite Enable for dow10 operation Bits 27-29 : RO *Reserved* Bit 30 : R/W Duplex Mode *0 = Half-Duplex (Not supported in GMII mode)* *1 = Full-Duplex* Bit 31 : R/W Transmit Enable *1 = enable polling of the descriptor memory.* *All The Transmit Descriptors must be initialized before enabling the Transmitter.*
Register base + 0x06004	TXCTRL1	Transmit Control 1 On reset: 0 x 00 00 00 00 Contains the bits required to control MDIO data transfers. For further information about the MII Management Interface see the IEEE Std 802.3 specification [1]. Bits 0-15 : R/W Data *If Direction = 0, the data to the PHY* *If Direction = 1, the data from the PHY* Bits 16-20 : R/W REG Address Bits 21-25 : R/W PHY Address Bits 26-29 : RO *Reserved* Bit 30 : R/W Direction *0 = write to the PHY* *1 = read from the PHY*

		Bit 31 : R/SC Start *1 = start read/write operation* *AFEC **clears** this bit when done*
Register base + 0x06008-0x09FFC	RESERVED	Reserved On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x10000	RXDESC0A0	Receive Descriptor 0 A, chain 0 On reset: Undefined Contains the receive frame start address. This is the physical bus address, which is not necessarily the same as for example the virtual address the operating system may be using. Bit 0-31 : R/W Frame Start Address
Register base + 0x10004	RXDESC0B0	Receive Descriptor 0 B, chain 0 On reset: Undefined Contains the bits required to control receive operations. The Receive Descriptor area should be initialized by the software before the receiver is enabled. This is because after reset the descriptors contain undefined values. If the descriptors are not initialized the receiver may trash the system memory by receiving frames to random memory locations when it is enabled. **Before a DMA data transfer** Bits 0-11 : R/W Length *The maximum length of the first fragment in bytes. The value written into this field must be at least 32.* Bits 12-30 : R/W *Reserved* Bit 31 : R/W Transfer *1 = enable transfer of the frame* *After the receiver has been enabled the user must always **set** this bit when writing to it. Clearing this bit after the receiving has been enabled causes an undefined operation.* ***AFEC clears this bit when the fragment has been transferred.*** **After a DMA data transfer** Bits 0-11 : R/W Length *The length of received fragment including the CRC checksum. (Preamble and SFD not included or transferred)* Bits 12-15 : R/W *Reserved* Bit 16 : R/W FIFO Error *FIFO overrun occurred during the reception because AFEC did not get access to the system bus in time. The frame was corrupted. This bit is valid only if the value of Continue bit is 0.* Bit 17 : R/W Size Error *1 = The incoming frame was too long (over 1600 bytes) and it was truncated to 1600 bytes or Descriptor Error in the next descriptor.* *This bit is valid only if the value of Continue bit is 0.* Bit 18 : R/W CRC Error *1 = The CRC checksum in the received frame was not correct.* *This bit is valid only if the value of Continue bit is 0.* Bit 19 : R/W Octet Error *1 = There was an uneven number of half bytes (nibbles) in the received frame.* *This bit is valid only if the value of Continue bit is 0.* Bit 20 : R/W Line Error *1 = During the receiving an Rx error was got from PHY.* *This bit is valid only if the value of Continue bit is 0.* Bit 21 : R/W EIF Drop error *1 = During the receiving external EIF component request to drop this frame. Frame is not valid and should be discarded.* *This bit is valid only if the value of Continue bit is 0.* Bits 22-29 : R/W *Reserved* Bit 30 : R/W Continue *0 = The frame is in one fragment, Length field indicates the total length of the frame.* *1 = The frame continues in Rx descriptor chain 1, Length field is left unmodified.* Bit 31 : R/W Transfer *After receiving the fragment this bit has value 0.*
Register	RXDESC0C0	Receive Descriptor 0 C, chain 0

base + 0x10008		On reset: Undefined Contains the lowest bits of the receive time of the frame. Bits 0-31 : RO Frame Receive time, bits 0-31
Register base + 0x1000C	RXDESC0D0	Receive Descriptor 0 D, chain 0 On reset: Undefined Contains the middle bits of the receive time of the frame. Bits 0-31 : RO Frame Receive time, bits 32-63
Register base + 0x10010	RXDESC0E0	Receive Descriptor 0 E, chain 0 On reset: Undefined Contains the highest bits of the receive time of the frame. Bits 0-31 : RO Frame Receive time, bits 64-95
Register base + 0x10014-0x1007C	RESERVED	Reserved On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x10080	RXDESC1A0	Receive Descriptor 1 A, chain 0 …
Register base + 0x10084	RXDESC1B0	Receive Descriptor 1 B, chain 0 …
…	…	…
Register base + 0x10380	RXDESC7A0	Receive Descriptor 7 A, chain 0 …
Register base + 0x10384	RXDESC7B0	Receive Descriptor 7 B, chain 0 …
Register base + 0x10388	RXDESC7C0	Receive Descriptor 7 C, chain 0 …
Register base + 0x1038C	RXDESC7D0	Receive Descriptor 7 D, chain 0 …
Register base + 0x10390	RXDESC7E0	Receive Descriptor 7 E, chain 0 …
Register base + 0x10394-0x13FFC	RESERVED	Reserved On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x14000	RXDESC0A1	Receive Descriptor 0 A, chain 1 On reset: Undefined Contains the start address of the fragment of the frame. This is the physical bus address, which is not necessarily the same as for example the virtual address the operating system may be using. Bit 0-31 : R/W Fragment Start Address
Register base + 0x14004	RXDESC0B1	Receive Descriptor 0 B, chain 1 On reset: Undefined Contains the bits required to control receive operations. The Receive Descriptor area should be initialized by the software before the receiver is enabled. This is because after reset the descriptors contain undefined values. If the descriptors are not initialized the receiver may trash the system memory by receiving frames to random memory locations when it is enabled. **Before a DMA data transfer** Bits 0-11 : R/W Length *The maximum length of the fragment in bytes. The value written into this field must be at least 32.* Bits 12-30 : R/W *Reserved* Bit 31 : R/W Transfer *1 = enable transfer of this fragment After the receiver has been enabled the user must always **set** this bit when writing to it. Clearing this bit after the receiving has been enabled causes an undefined operation. **AFEC clears this bit when the fragment has been transferred.*** **After a DMA data transfer** Bits 0-11 : R/W Length *The length of received fragment including the CRC*

		checksum. (Preamble and SFD not included or transferred) Bits 12-15 : R/W *Reserved* Bit 16 : R/W FIFO Error *FIFO overrun occurred during the reception because AFEC did not get access to the system bus in time. The frame was corrupted. This bit is valid only if the value of Continue bit is 0.* Bit 17 : R/W Size Error *1 = The incoming frame was too long (over 1600 bytes) and it was truncated to 1600 bytes or Descriptor Error in the next descriptor.* *This bit is valid only if the value of Continue bit is 0.* Bit 18 : R/W CRC Error *1 = The CRC checksum in the received frame was not correct.* *This bit is valid only if the value of Continue bit is 0.* Bit 19 : R/W Octet Error *1 = There was an uneven number of half bytes (nibbles) in the received frame.* *This bit is valid only if the value of Continue bit is 0.* Bit 20 : R/W Line Error *1 = During the receiving an Rx error was got from PHY.* *This bit is valid only if the value of Continue bit is 0.* Bit 21 : R/W EIF Drop error *1 = During the receiving external EIF component request to drop this frame. Frame is not valid and should be discarded.* *This bit is valid only if the value of Continue bit is 0.* Bits 22-29 : R/W *Reserved* Bit 30 : R/W Continue *0 = This was the last fragment of the frame.* *1 = The frame continues in the next descriptor in Rx descriptor chain 1. Length field is left unmodified.* Bit 31 : R/W Transfer *After receiving the fragment this bit has value 0.*
Register base + 0x14008-0x1407C	RESERVED	Reserved On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x14080	RXDESC1A1	Receive Descriptor 1 A, chain 1 …
Register base + 0x14084	RXDESC1B1	Receive Descriptor 1 B, chain 1 …
…	…	…
Register base + 0x14F80	RXDESC31A1	Receive Descriptor 31 A, chain 1 …
Register base + 0x14F84	RXDESC31B1	Receive Descriptor 31 B, chain 1 …
Register base + 0x14F88-0x17FFC	RESERVED	Reserved On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x18000	TXDESC0A0	Transmit Descriptor 0 A, chain 0 On reset: Undefined Contains the start address of the frame to be transmitted. This is the physical bus address, which is not necessarily the same as for example the virtual address the operating system may be using. Bit 0-31 : Frame start address
Register base + 0x18004	TXDESC0B0	Transmit Descriptor 0 B, chain 0 On reset: Undefined Contains the bits required to control transmit operations. Transmit Descriptor area should be cleared by the software before the transmitter is enabled. This is because after reset the descriptors contain undefined values. If the descriptors are not cleared the transmitter may start transmitting non-existent frames from random memory locations when it is enabled. **Before a frame transmission**

		Bits 0-11 : R/W Length *Length of the first fragment including CRC checksum. (CRC is overwritten by the Transmitter, so it does not have to be valid. Preamble and SFD are not included into the length or DMA transfer). The value written into this field must be at least 32. The total length of the frame must be at least 64 bytes.* Bits 12-29 : R/W *Reserved* Bit 30 : R/W Continue *0 = This is the only fragment of the frame.* *1 = The frame continues in the next descriptor in Tx descriptor chain 1.* Bit 31 : R/W Transfer *Set this bit to value 1 to transmit the frame pointed by Transmit Descriptor A. After the transmitter has been enabled the user must always **set** this bit when writing to it. Clearing this bit after the transmitter has been enabled causes an undefined operation. **AFEC clears this bit when the frame has been transferred.*** **After a frame transmission** Bits 0-15 : R/W *Reserved* Bit 16 : R/W FIFO Error *1 = FIFO underrun occurred during the transmission because AFEC did not get access to the SoC bus in time. The frame was corrupted.* *This bit is valid only if the value of Continue bit is 0.* Bit 17 : R/W Late Collision *1 = Late Collision occurred in the medium. The frame was corrupted.* *This bit is valid only if the value of Continue bit is 0.* Bit 18 : R/W Frame Dropped *1 = Frame was dropped (not sent) because of 16 collisions in the medium during its transmission.* *This bit is valid only if the value of Continue bit is 0.* Bits 19-22 : R/W Collision Count *Number of collisions in the medium during the transmission of this frame. A large amount of collisions is an indication of an overloaded medium. Note that a collision is not an error.* *This bit is valid only if the value of Continue bit is 0.* Bits 23-29 : R/W *Reserved* Bit 30 : R/W Continue *0 = This is the only fragment of the frame.* *1 = The frame continues in the next descriptor in Tx descriptor chain 1.* Bit 31 : R/W Transfer *After transmission of the frame this bit has value 0.* Collisions, Late Collisions, and Frame Dropped errors are possible only when AFEC is in Half-Duplex mode (See Chapter 2 for more details).
Register base + 0x18008	TXDESC0C0	Transmit Descriptor 0 C, chain 0 On reset: Undefined Contains the lowest bits of the transmit time of the frame. Bits 0-31 : RO Frame transmit time, bits 0-31
Register base + 0x1800C	TXDESC0D0	Transmit Descriptor 0 D, chain 0 On reset: Undefined Contains the middle bits of the transmit time of the frame. Bits 0-31 : RO Frame transmit time, bits 32-63
Register base + 0x18010	TXDESC0E0	Transmit Descriptor 0 E, chain 0 On reset: Undefined Contains the highest bits of the transmit time of the frame. Bits 0-31 : RO Frame transmit time, bits 64-95
Register base + 0x18014-0x1807C	RESERVED	Reserved On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x18080	TXDESC1A0	Transmit Descriptor 1 A, chain 0 …
Register base + 0x18084	TXDESC1B0	Transmit Descriptor 1 B, chain 0 …
…	…	…
Register	TXDESC7A0	Transmit Descriptor 7 A, chain 0

base + 0x18380		…
Register base + 0x18384	TXDESC7B0	Transmit Descriptor 7 B, chain 0 …
Register base + 0x18388	TXDESC7C0	Transmit Descriptor 7 C, chain 0 …
Register base + 0x1838C	TXDESC7D0	Transmit Descriptor 7 D, chain 0 …
Register base + 0x18390	TXDESC7E0	Transmit Descriptor 7 E, chain 0 …
Register base + 0x18394-0x1BFFC	RESERVED	Reserved On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x1C000	TXDESC0A1	Transmit Descriptor 0 A, chain 1 On reset: Undefined Contains the start address of the fragment to be transmitted. This is the physical bus address, which is not necessarily the same as for example the virtual address the operating system may be using. Bit 0-31 : Frame start address
Register base + 0x1C004	TXDESC0B1	Transmit Descriptor 0 B, chain 1 On reset: Undefined Contains the bits required to control transmit operations. Transmit Descriptor area should be cleared by the software before the transmitter is enabled. This is because after reset the descriptors contain undefined values. If the descriptors are not cleared the transmitter may start transmitting non-existent frames from random memory locations when it is enabled. **Before a frame transmission** Bits 0-11 : R/W Length *Length of the fragment including CRC checksum. (CRC is overwritten by the Transmitter, so it does not have to be valid. Preamble and SFD are not included into the length or DMA transfer.) If this is not the last fragment of the frame (Continue = 1) the length must be at least 32.* Bits 12-29 : R/W *Reserved* Bit 30 : R/W Continue *0 = This is the last fragment of the frame.* *1 = The frame continues in the next descriptor in Tx descriptor chain 1.* Bit 31 : R/W Transfer *Set this bit to value 1 to enable transmitting of the fragment. The transmission of the first fragment in chain 0 has to be enabled last. After the transmitter has been enabled the user must always **set** this bit when writing to it. Clearing this bit after the transmitter has been enabled causes an undefined operation. **AFEC clears this bit when the frame has been transferred.*** **After a frame transmission** Bits 0-15 : R/W *Reserved* Bit 16 : R/W FIFO Error *1 = FIFO underrun occurred during the transmission because AFEC did not get access to the SoC bus in time. The frame was corrupted.* *This bit is valid only if the value of Continue bit is 0.* Bit 17 : R/W Late Collision *1 = Late Collision occurred in the medium. The frame was corrupted.* *This bit is valid only if the value of Continue bit is 0.* Bit 18 : R/W Frame Dropped *1 = Frame was dropped (not sent) because of 16 collisions in the medium during its transmission.* *This bit is valid only if the value of Continue bit is 0.* Bits 19-22 : R/W Collision Count *Number of collisions in the medium during the transmission of this frame. A large amount of collisions is an indication of an overloaded medium. Note that a collision is not an error.* *This bit is valid only if the value of Continue bit is 0.*

		Bits 23-29 : R/W *Reserved* Bit 30 : R/W Continue *0 = This is the last fragment of the frame.* *1 = The frame continues in the next descriptor in Tx descriptor chain 1.* Bit 31 : R/W Transfer *After transmission of the fragment this bit has value 0.* Collisions, Late Collisions, and Frame Dropped errors are possible only when AFEC is in Half-Duplex mode (See Chapter 2 for more details).
Register base + 0x1C008-0x1C07C	RESERVED	Reserved On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*
Register base + 0x1C080	TXDESC1A1	Transmit Descriptor 1 A, chain 1 …
Register base + 0x1C084	TXDESC1B1	Transmit Descriptor 1 B, chain 1 …
…	…	…
Register base + 0x1CF80	TXDESC31A1	Transmit Descriptor 31 A, chain 1 …
Register base + 0x1CF84	TXDESC31B1	Transmit Descriptor 31 B, chain 1 …
Register base + 0x1CF8C-0x1FFFC	RESERVED	Reserved On reset: Undefined This memory area is reserved. Read access is allowed and will not affect AFEC. Write access is strictly **prohibited** and may cause an undefined operation. Bits 0-31 : R/W *Reserved*

Table 7 Registers

5 Functional Description

5.1 General

The internal blocks of AFEC are presented in Figure 14. The functionality of AFEC is controlled by writing into the memory mapped user registers (See also AFEC registers in Table 7). The Rx and Tx DMA controllers transfer receive (Rx) and transmit (Tx) data directly to/from the system memory without processor intervention; the Rx and Tx data paths are presented in Figure 15 and Figure 16.

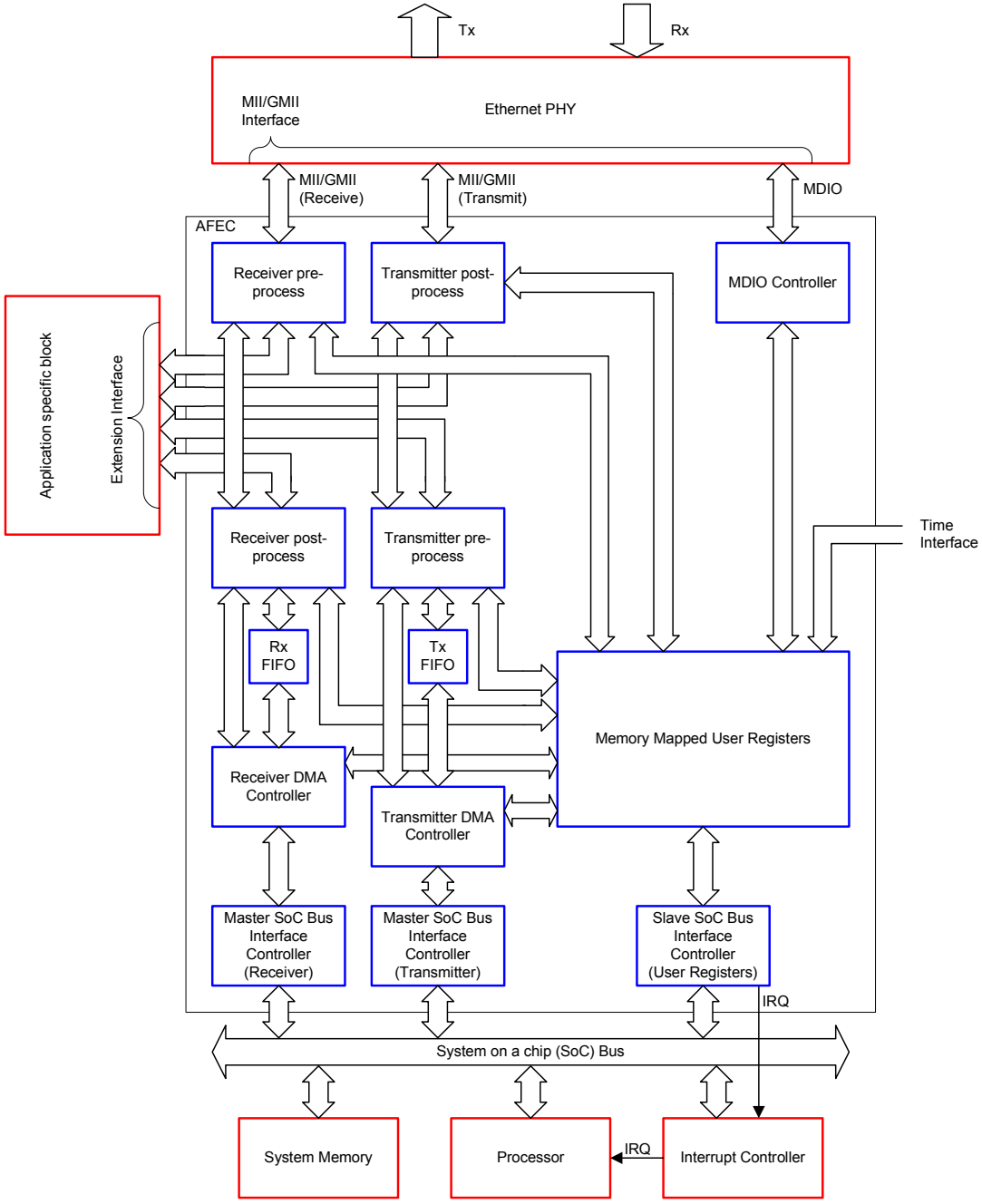

Figure 14 AFEC Block Diagram

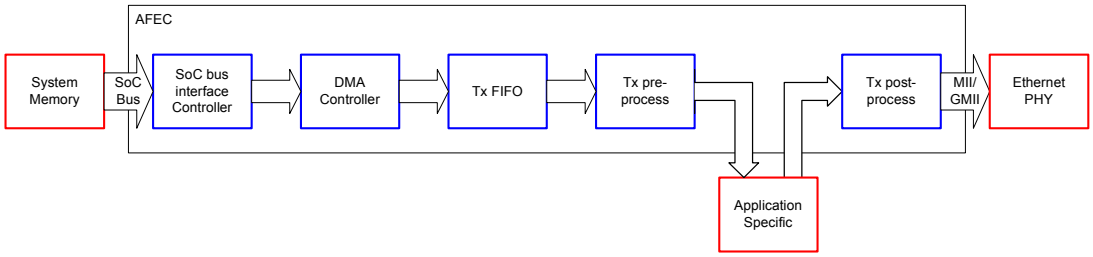

Figure 15 Tx Data Path

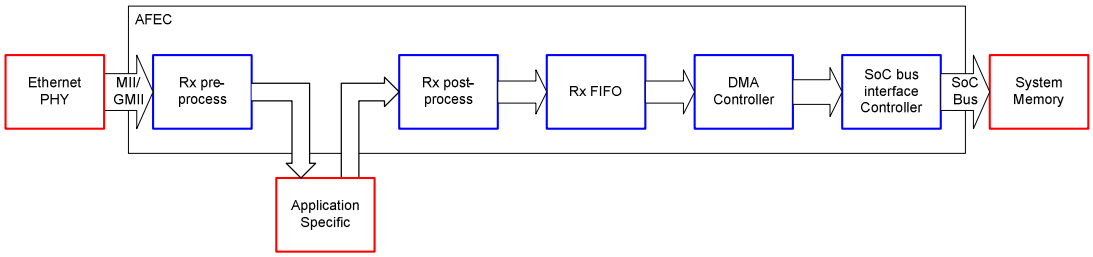

Figure 16 Rx Data Path

5.2 Transmitter

The Transmitter transmits frames to the Ethernet medium. In AFEC the functionalities of Transmitter and Receiver are completely independent of each others; the operation mode of the Receiver has no effect on the functionality of the Transmitter.

The Transmitter uses Direct Memory Access (DMA) to fetch the transmitted frames from the system memory. The locations of the frames in the system memory are given to the Transmitter in Transmit Descriptors. In other words, the Transmit Descriptors act as pointers to frames in the system memory. One frame may consist of one or more fragments, located separately in the system memory. Each one of the descriptors points to one fragment. In addition to acting as pointers, the Transmit Descriptors include also other information on the fragments and frames, for example their length. Commands to transmit frames are given by setting the transfer bits in the corresponding descriptors. After a frame has been transmitted, the Transmitter returns information on how the transmission succeeded in the descriptor of the last fragment and activates the Frame Transmitted Interrupt. The status bits in the other descriptors than the descriptor of the last fragment are not valid. Except for the Transfer bits that are always zeroed immediately after the transmitter has processed the fragment.

The Transmit Descriptors are in the user accessible registers of AFEC (See Register map in Table 7). The Transmit Descriptors are divided to two descriptor chains, chain 0 and chain 1. The first fragment of the frame is always given in a descriptor in descriptor chain 0. If the frame is divided into more than one fragment, the rest of the fragments are in successive descriptors in descriptor chain 1. Each one of the Transmit Descriptors in chain 0 is divided into five registers. The registers are Transmit Descriptor A, Transmit Descriptor B, Transmit Descriptor C, Transmit Descriptor D and Transmit Descriptor E. The descriptors in Transmit Descriptor chain 1 consist of only two registers, Transmit Descriptor A and Transmit Descriptor B. Basically this is because the information in registers Transmit Descriptor C, Transmit Descriptor D and Transmit Descriptor E applies to the whole frame and not to the individual fragments. The usage of two descriptor chains with different kinds of descriptors minimizes the number of registers needed saving the valuable FPGA resources.

There are eight Transmit Descriptors in Transmit Descriptor chain 0, from 0 to 7. A descriptor in chain 0 always carries the first fragment of a frame. After reset, when the first frame is to be sent, its first fragment is sent by using Transmit Descriptor 0 in chain 0. The first fragment of the second frame is sent by using Transmit Descriptor 1, and so on, until finally the first fragment of the eighth frame is sent by using Transmit Descriptor 7. After that, the first fragment of the ninth frame is sent by using Transmit Descriptor 0, the first fragment of the tenth frame by using the Descriptor 1, and so on. The benefit of having more than one Transmit Descriptor in chain 0 is that the Transmitter can be given frames to be sent before it has transmitted the previous frames; with eight descriptors there can be up to seven frames in a queue waiting to be transmitted while one frame is being transmitted. This way, if there are frames waiting in the descriptors, the Transmitter can start sending the next frame right after the transmitting of the previous one has finished.

In chain 1 there are 32 Transmit Descriptors, from 0 to 31. If a frame consists of more than one fragment, all the other fragments except the first one are presented by the descriptors in chain 1. The maximum number of fragments a frame can be divided to is limited only by the number of descriptors in chain 1; a frame can be divided into maximum of 33 fragments in the system memory (the first fragment in chain 0, the rest 32 fragments in chain 1). Just like the descriptors in chain 0, also the descriptors in chain 1 are used by AFEC in ascending order, starting from the first one after the reset.

Note that the Transmit Descriptors in both chains have to be used in the right order by the software (the device driver). After the Transmitter sends a frame using Descriptor *n*, the next frame is sent by using descriptor *n+1* (except in case of the last descriptor in the chain, in which case the next descriptor to be used is Descriptor 0); if the software initializes a wrong descriptor the frame will not be sent before the Transmitter reaches that descriptor. There is no register in the AFEC register interface that would indicate which Descriptor is the next Descriptor to be used. Instead, the software (the device driver) has to remember the Descriptors it used the last time it sent a frame. Both the software reset command and hardware reset signal reset the internal counters and after the reset the Transmitter starts from Descriptor 0 in both chains.

The Transmitter automatically calculates and inserts the correct CRC checksums to transmitted frames. Also the preamble and the Start Frame Delimiter (SFD) are automatically inserted.

The Transmitter duplex mode and FIFO thresholds can be adjusted by using the Transmit Control register. The Transmit Control register has also a bit for enabling and disabling the Transmitter. When the Transmitter is disabled, it does not transmit frames given to it in Transmit Descriptors.

The Transmitter duplex mode (Full-Duplex / Half-Duplex) should be the same as the current duplex mode of the PHY. If the duplex mode of the PHY is forced, setting the Transmitter into the same mode is trivial. However, in the case that the duplex mode of the PHY is a result of an auto-negotiation process, the software (the device driver) has to, by using MDIO, find out in which duplex mode the PHY is, and set the duplex mode of AFEC to the same one. Because the duplex mode may change during the operation, the software has to periodically poll the PHY for mode changes, and when the duplex mode changes, the software has to change the duplex mode of AFEC accordingly.

5.2.1 Initialization

After reset the Transmit Descriptor registers contain undefined values. Therefore value 0 has to be written to all of the Transmit Descriptor registers before the Transmitter is switched on for the first time. If this is not done, and the Transmitter is switched on, the Transmitter may start sending frames from the Descriptors whose Transfer bits have value 1.

5.2.2 Pseudo Code

The functionality of the Transmitter is presented in pseudo code below. Note that after a hardware or software reset `Tx_Index_chain0` and `Tx_Index_chain1` both have value 0.

```
loop
        If TxDescB[Tx_Index_chain0].Transfer == 1 then
                Transmit fragment
                If TxDescB[Tx_Index_chain0].Continue == 1 then
                        Do
                                Transmit fragment
                                Continue = TxDescB[Tx_Index_chain1].Continue
                                If Tx_Index_chain1 < number_of_transmit_descriptors - 1
                                        Tx_Index_chain1++
                                Else
                                        Tx_Index_chain1=0
                        While Continue == 1
                Activate Frame Transmitted interrupt
                If Tx_Index_chain0 < number_of_transmit_descriptors - 1
                        Tx_Index_chain0++
                Else
                        Tx_Index_chain0=0
        Else
                Do nothing
end loop
```

As can be seen, when the Transmitter is not transmitting it just polls the Transfer bit of the next descriptor in chain 0.

5.2.3 Errors during Transmission

There are four kinds of errors that may occur while AFEC is transmitting a frame. These are Transmit FIFO underrun, Late Collision, frame being dropped because of too many collisions, and Descriptor Error.

Transmit FIFO underrun occurs when AFEC is transmitting a frame and it cannot get the data to be transmitted from the system memory fast enough. The FIFO underrun results the transmitted frame to become corrupted. After the Transmit FIFO underrun has occurred, the Transmit FIFO Error Interrupt is activated and the FIFO Error bit in the last Transmit Descriptor is given value 1.

A Late Collision occurs when some other node starts to transmit while AFEC is already transmitting. If the Late Collision occurs, the Late Collision bit in the last Transmit Descriptor is given value 1. The Late Collision is not an error in AFEC; it is most probably an error in the medium or in some of the other nodes. When AFEC notices late collision it will send jam to the line like in normal collision and frame is dropped. AFEC activates Transmit Error interrupt to indicate error in transmission.

Normal collisions are not errors. However, if a single frame collides 16 times it is dropped. After the frame being dropped, the Frame Dropped bit in the last Transmit Descriptor is given value 1 and the Transmit Error Interrupt is activated.

A Transmit Descriptor Error occurs when a descriptor indicates that the frame continues in the next fragment pointed by the next descriptor and the Transfer bit in the next descriptor has value 0. A Transmit Descriptor Error can happen only as a result of faulty software (device driver). It is not required from the Transmitter to recover from this error without a reset. What the Transmitter does after a Transmit Descriptor Error is undefined.

Note that Late Collisions and normal collisions may happen only when AFEC is in Half-Duplex mode. Therefore, the Transmit FIFO error is the only error that may occur during transmission in Full-Duplex mode.

The summary of the recoverable errors during transmission can be found in Table 8.

	Indication:			
Error:	Active bits in Transmit Descriptor:	Transmit FIFO Error Interrupt	Transmit Error Interrupt	Frame Transmitted Interrupt
FIFO underrun	FIFO Error	X	-	-
Late Collision	Late Collision	-	X	-
16 collisions	Frame Dropped	-	X	-

Table 8 Errors during Transmission

Note that in case of more than one fragment, the error indication presented in Table 8 is written only to the descriptor of the last fragment.

5.3 Receiver

The Receiver receives frames from the Ethernet medium. In AFEC the functionalities of Receiver and Transmitter are completely independent of each others; the operating mode of the Transmitter has no effect on the functionality of the Receiver.

The receiver uses Direct Memory access (DMA) to write the received frames to the system memory. The locations where to the received frames are written in the system memory are given to the Receiver in Receive Descriptors. In other words, the Receive Descriptors act as pointers to free spaces in the system memory where to frames can be received. Each one of the descriptors points to a single continuous memory area. A frame can be received by using one or more descriptors depending on the length of the frame and the sizes of the memory areas pointed by the descriptors. After a frame has been received, the Receiver returns information on how the receiving succeeded in the last descriptor of the frame and activates the Frame Received Interrupt. The data transferred to the system memory does not include the CRC nor SFD.

The Receive Descriptors are in the user accessible registers of AFEC (See Register map in Table 7). The Receive Descriptors are divided into two descriptor chains, chain 0 and chain 1. The first fragment of each frame is received into a memory area pointed by Receive Descriptor in descriptor chain 0. If the received frame does not fit into the memory area pointed by the descriptor in descriptor chain 0, the next fragments are received using successive descriptors in descriptor chain 1. Each one of the Receive Descriptors in descriptor chain 0 is divided into five registers. The registers are Receive Descriptor A, Receive Descriptor B, Receive Descriptor C, Receive Descriptor D and Receive Descriptor E. As in the Transmitter, also in the Receiver the descriptors in descriptor chain 1 consist of only two registers, Receive Descriptor A and Receive Descriptor B.

In the Receive Descriptor chain 0 there are eight Receive Descriptors, from 0 to 7. After reset, when the first frame is to be received, its first fragment is received by using Receive Descriptor 0 in chain 0. The first fragment of the second frame is received by using Receive Descriptor 1, and so on, until finally the first fragment of the eighth frame is received by using Receive Descriptor 7 in chain 0. After that, the first fragment of the ninth frame is received by using again Receive Descriptor 0 of chain 0, the first fragment of the tenth frame by using the Descriptor 1 of chain 0, and so on. Each one of the Receive Descriptors has a Transfer bit that indicates to the Receiver that it is allowed to receive a frame by using that descriptor. If a frame arrives from the network, and the Transfer bit is not set in the Receive Descriptor, the Receiver cannot receive the fragment and the data is lost. Also as a consequence, Receive Frame Dropped Interrupt is activated.

In Receive Descriptor chain 1 there are 32 Receive Descriptors, from 0 to 31. If the received frame does not fit into the memory area pointed by the Receive Descriptor in chain 0, the receiving is continued to the memory area pointed by a Receive Descriptor in chain 1. As the descriptors in chain 0 also the descriptors in chain 1 are consumed by AFEC in ascending order, starting from descriptor 0 after reset. If the received frame does not fit to the memory area pointed by the descriptor, the Receiver moves to the next descriptor in chain 1 until the end of the received frame is reached. When receiving a frame, the Receiver uses all the memory pointed by the descriptors it consumes, except the last one, which normally gets only partially filled. After receiving a frame, the receiver writes to the last consumed descriptor the amount of memory the frame used from it, and the receive status. The status information is not written to other descriptors than the last descriptor of the frame. The transfer bits of the consumed descriptors are zeroed by the Receiver right after consuming the descriptor. The software (the device driver) can write to the consumed descriptors (whose transfer bit is zeroed) even if the end of the frame is still to be received.

In order to be able to receive frames, the software (the device driver) has to make sure that there are enough free descriptors available for the Receiver. In practice, it is wise to have as many descriptors free, waiting for frames to arrive, as possible. This way the possibility of packet loss is minimized even in cases where the processor is very busy and frames arrive with high rate.

The Receiver can be enabled and disabled, and the Receiver FIFO thresholds can be adjusted by using Receive Control register (see register map in Table 7). It can also be configured whether the Receiver receives broadcast, multicasts, all frames, or just the frames destined for it.

5.3.1 Initialization

After reset Receive Descriptors contain undefined values. Therefore proper values have to be written to all the Receive Descriptor registers before the Receiver is switched on for the first time. If this is not done, and the Receiver is switched on, the Receiver may start receiving frames from the network. The received frames go to random addresses and trash the system memory.

In practice, before the Receiver is switched on, all the Receive Descriptors should be initialized so that their Transfer bits have value 1, and that they point to memory areas into which frames can be received.

5.3.2 Pseudo Code

The functionality of the Receiver is presented in pseudo code below. Note that after a hardware or software reset `Rx_Index_chain0` and `Rx_Index_chain1` both have value 0.

```
Loop
        Wait for frame to start
        Receive fragment
        If frame_length > RxDescB[Rx_Index_chain0].Length then
                RxDescB[Rx_Index_chain0].Continue = 1
                Frame_length = Frame_length - [Rx_Index_chain0].Length
                Do
                        Receive Fragment
                        If frame_length > RxDescB[Rx_Index_chain1].Length then
                                RxDescB[Rx_Index_chain1].Continue = 1
                                Frame_length = Frame_length -
                                        RxDescB[Rx_Index_chain1].Length
                                Continue = 1
                        Else
                                RxDescB[Rx_Index_chain1].Continue = 0
                                RxDescB[Rx_Index_chain1].Length = Frame_length
                                Continue = 0
                        If Rx_Index_chain1 < number_of_receive_descriptors - 1
                                Rx_Index_chain1++
```

```
                        Else
                                Rx_Index_chain1=0
                Until Continue == 0
        Else
                RxDescB[Rx_Index_chain0].Length = Frame_length
        Activate Frame Received Interrupt
        If Rx_Index_chain0 < number_of_receive_descriptors - 1
                Rx_Index_chain0++
        Else
                Rx_Index_chain0=0
end loop
```

5.3.3 Errors during Reception

There are six kinds of errors that can occur while AFEC is receiving a frame. These are: Receive FIFO overrun, Size Error, CRC Error, Octet Error, Line Error and Receive Descriptor Error.

Receive FIFO overrun occurs when AFEC is receiving a frame and it cannot move the received data fast enough to the system memory. The FIFO overrun results the received frame to become corrupted. After Receive FIFO overrun has happened, Receive FIFO Error and Frame Received interrupts are activated and FIFO Error bit in the Receive Descriptor of the last fragment is given value 1.

Size Error indicates that the received frame is over 1600 bytes long (without preamble and SFD, with CRC). In that case the Frame is truncated to 1600 bytes, and Size Error bit in the Receive Descriptor of the last frame is given value 1. The Frame Received Interrupt is activated.

CRC Error indicates that the CRC checksum in the received frame was not correct. This is a result of an error in the received frame and an indication to the software that it should discard the frame. FIFO overrun, Size Error, Octet Error and Line Error usually cause also a CRC error to the received frame. The CRC Error bit in the Receive Descriptor of the last fragment is given value 1 and Frame Received interrupt is activated.

Octet Error occurs when the received frame contains an uneven number of half bytes (nibbles). This kind of a frame is not valid. As a result, The Octet Error bit in the Receive Descriptor of the last fragment is given value 1 and the Frame Received Interrupt is activated.

Line Error indicates that during receiving of the frame the PHY reported AFEC of an error. In this case the Line Error bit in the Receive Descriptor of the last fragment is given value 1 and the Frame Received Interrupt is activated.

Receive Descriptor Error occurs when a frame is received, but there is no free Receive Descriptors in chain 0 where frame can be wrote. In this case Receive Frame Dropped Interrupt is activated. All descriptors are left untouched and the descriptor pointer is not increased. So the next receive attempt is made using the same descriptor.

Other Receive Descriptor Error is happened, if Receive Descriptor length is too short for incoming frame or if there are no chain 1 descriptors available. In this case the descriptors used by the previous fragments are consumed, and their Transfer bits are zeroed. Receive status is written to the descriptor of the last fragment that was received; Size Error bit in the Receive Descriptor is given value 1.

Frames which size is less than minimum Ethernet frame size shall be dropped by reception without any errors. All frames reserves at least *min_frame_size* +

interframe_gap time. If new frame start before this time (after small frame) it shall be dropped also without errors.

The summary of the possible Errors during reception can be found in Table 9.

Error:	Indication: Active bits in Receive Descriptor:	Receive FIFO Error Interrupt	Frame Dropped Interrupt	Frame Received Interrupt
FIFO	FIFO Error	X	-	X
Size	Size Error	-	-	X
CRC	CRC Error	-	-	X
Octet	Octet Error	-	-	X
Line	Line Error	-	-	X
Descriptor (in chain 0)	-	-	X	-
Descriptor (in chain 1)	Size Error	-	-	X

Table 9 Errors during Reception

Note that in case of more than one fragment, the error indication presented in Table 9 is written only to the descriptor of the last fragment.

5.4 Timestamping

All the received and transmitted frames are timestamped in order to support IEEE 1588 version 2 Precision Time Protocol. The Timestamp Point in an Ethernet frame is presented in Figure 17. When the Timestamp Point of an Ethernet frame is in the MII/GMII interface, the clock time from the `clock_time` signals of the Time Interface is copied to Descriptor C, Descriptor D and Descriptor E registers of the corresponding transmit or Receive Descriptor. It is not specified in this document how the clock time in `clock_time` signals should be presented and AFEC does not limit the presentation in any way; the signals are copied to the descriptors as they are.

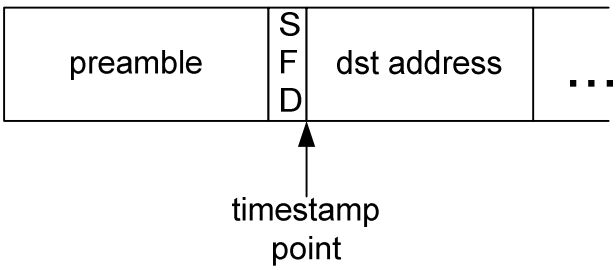

Figure 17 Ethernet Frame Timestamp Point

5.5 MDIO Controller

Registers in the PHY are accessed through the Transmit Control 1 register (see register map in Table 7). Table 10 lists the registers in the PHY. The registers are used, for example, for setting auto-negotiation capabilities or link modes, and getting link statuses and modes. For further information about the register set see the IEEE Standard 802.3 [1] and the datasheet of the Ethernet PHY chip used.

Address	Register in PHY	Basic/Extended	
		MII	GMII
0x00	Control	B	B
0x01	Status	B	B
0x02-0x03	PHY Identifier	E	E
0x04	Auto-Negotiation Advertisement	E	E
0x05	Auto-Negotiation Link Partner Base Page Ability	E	E
0x06	Auto-Negotiation Expansion	E	E
0x07	Auto-Negotiation Next Page Transmit	E	E
0x08	Auto-Negotiation Link Partner Received Next Page	E	E
0x09	MASTER-SLAVE Control Register	E	E
0x0A	MASTER-SLAVE Status Register	E	E
0x0B-0x0E	Reserved	E	E
0x0F	Extended Status	Reserved	B
0x10-0x1F	Vendor Specific	E	E

Table 10 MII/GMII Management Register Set

5.5.1 PHY Initialization

Normally Ethernet PHY chips are fully functional after power-up, which means that no initialization of the PHY is necessarily needed. However, in some situations, depending on the PHY chip model used and the way it has been connected, some initialization could be required before the PHY is fully functional. Consult the datasheets of the PHY silicon device about the matter.

5.6 Interrupt Control

There are two registers for controlling the interrupt generation of AFEC: the Interrupt Mask (INTMASK) register and the Interrupt Status (INTSTAT) register.

When an interrupt is activated, the corresponding bit in the Interrupt Status register is given value 1. AFEC activates the external interrupt signal when any one of the interrupt status bits in the Interrupt Status register is set and the corresponding Interrupt Mask bit has value 1. The external interrupt signal is active high. In case delayed interrupt is used, the assertion of the external interrupt signal is delayed until the time configured in General register in Interrupt Delay bits from the previous interrupt signal assertion has passed. If delayed interrupts are enabled and the interrupt status in Interrupt Status (INTSTAT) register is cleared before the interrupt signal is asserted, the interrupt signal will not be asserted at all.

The interrupt status can be checked at any time by reading the Interrupt Status register, and interrupts can be cleared any time by clearing the corresponding bits in the Interrupt Status register. Note that setting the Interrupt Mask bit of an interrupt to 0 does not have any effect on the behavior of the corresponding Interrupt Status bit; it just means that the Interrupt Status bit will not affect the external interrupt signal.

In practice, when clearing the interrupts, it is recommended to leave the Interrupt Status bits with value 0 into their current state (to not clear them) by writing 1 to them. This prevents the user from accidentally clearing the interrupts that are activated between the reading of the interrupt status and the clearing of the interrupts.

5.7 Reset

5.7.1 Hardware Reset

The hardware reset is made by setting the global asynchronous reset input signal to its active state. All of the flip-flops in AFEC are immediately returned to their reset states. This corresponds to the power-up condition in all the other parts, except for the internal memory blocks of the FPGA that are not initialized.

5.7.2 Software Reset

The software reset is made by writing value 1 to the Reset Bit (bit 31) in the General Register. AFEC cancels all of its current operations and waits until all its state machines have returned to their reset states. After the resetting of AFEC and the application specific blocks connected to the EIF interface has been done, AFEC clears the Reset bit in the General register. After software reset AFEC is in the same state as after hardware reset.

5.7.3 EIF Init request

EIF init request bit (bit31) in INTSTAT register indicates that some block in EIF chain is in state that it cannot work. If this bit is active user should give software reset to block.

5.7.4 MII reset

MII interface TX and RX reset shall reset only the MAC block. If MII reset is caused when frame is received AFEC may concatenate next frame to part of frame received before MII reset. If this happened frame is received with CRC error. If concatenated frame size is more than maximum Ethernet frame size, rx size error is activated. Otherwise next frame is received properly after MII reset.

If MII reset is caused when frame is transmitted, AFEC doesn't notice that frame transmission has been failed. After reset is released transmission is working properly.

6 Glossary

AFEC	Advanced FEC
CD	Collision Detection
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
Desc	Ethernet Frame Descriptor
DMA	Direct Memory Access
EIF	Extension Interface
FEC	Flexibilis Ethernet Controller
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GMII	Gigabit Media Independent Interface
IEEE	Institute of Electrical and Electronics Engineers
IP	Intellectual Property, Internet Protocol
LAN	Local Area Network
LLC	Logical Link Control
MAC	Media Access Control
MDC	Management Data Clock
MDIO	Management Data Input/Output
MII	Media Independent Interface
NIC	Network Interface Controller
OPB	On-Chip Peripheral Bus
PCI	Peripheral Component Interconnect
PTP	Precision Time Protocol
PHY	Physical Layer Device
Rx	Receive
SFD	Start Frame Delimiter
SoC	System on a Chip
Tx	Transmit

VHDL	Very high speed integrated circuits Hardware Description Language
VLAN	Virtual LAN

7 References

[1] IEEE Inc, *IEEE Standard 802.3-2005*, 9 Dec 2005

[2] Altera Corporation, "*Avalon Interface specification"*, April 2005

[3] IEEE Inc, *IEEE Standard 1588-2002,* 8 Nov 2002