Gigabit Ethernet Media Access Controller

Version 0 Revision 1

Oct. 10, 2018 (July 7, 2018)

Future Design Systems, Inc.
www.future-ds.com / contact@future-ds.com

Abstract

This document describes specifications Gigabit Ethernet MAC.

Table of Contents

1 Overview

'gig_eth_mac_ahb' block provides Gigabit Ethernet Media Access Controller function.

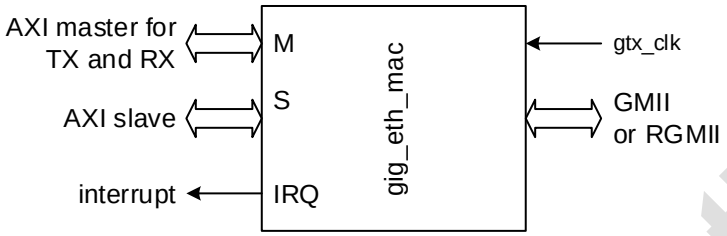


Figure 1: Overview

2 Internal structure

As shown in Figure 2, Ethernet MAC consists of four main parts and has three clock domains.

- CSR (AXI ACLK domain)
- sending DMA (AXI ACLK domain)
- receiving DMA (AXI ACLK domain)
- GIMII TX interface (gtxclk domain)
- GMII RX interface (rxclk domain)

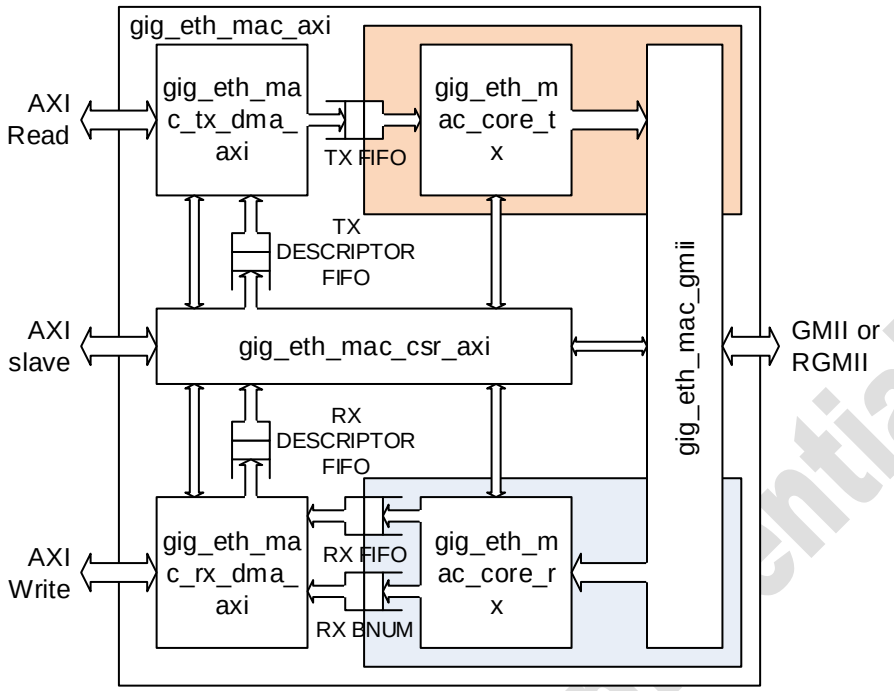


Figure 2: Internal structure[1]

2.1 Parameters

Macros	Note
SIM	pure RTL simulation (do not define 'SIM' and 'SYN' at the same time)
SYN	logic synthesis, which makes use of Xilinx specific things
RIGOR	rigorously check for simulation use this with 'SIM
RGMII	define for RGMII

Parameter	Note
AXI_MST_ID	Drive to M_MID[AXI_WIDTH_CID-1:0]
AXI_WIDTH_CID	Bitwidth of channel ID, which is a part of AxID[…]
AXI_WIDTH_ID	Bitwidth of transaction ID, which is a part of AxID[…]
AXI_WIDTH_AD	Bitwidth of address bus for AxADDR[…]
AXI_WIDHT_DA	Bitwidth of data bus for xDATA[…]
ACLK_FREQ	Frequency of ACLK, which is required to calculate duration of PHY reset.
P_TX_FIFO_DEPTH	depth of the FIFO between DMA and MAC core for TX packet
P_RX_FIFO_DEPTH	depth of the FIFO between DMA and MAC core for RX packet
P_TX_DESCRIPTOR_FAW	depth of the FIFO between DMA and CSR for TX
P_RX_DESCRIPTOR_FAW	depth of the FIFO between DMA and CSR for RX
MAC_ADDR	Default MAC address

[1] At this moment, RGMII is not fully tested yet.

P_TXCLK_INV	Drive “gmii_txc” using180 inverted clock of gtx_clk, when 1. Drive “rgmii_txc” using 90 degree shifted clock of gtx_clk, when 1.
FPGA_FAMILY	specify family of FPGA ● “SPARTAN6” : Spartan-6 ● “SPARTAN7” : Spartan-7 ● “VIRTEX4” : Virtex-4 ● “VIRTEX5” : Virtex-5 ● “VIRTEX6” : Virtex-6 ● “VIRTEX7” : Virtex-7 ● “ARTIX7” : Artix-7 ● “KINTEX7” : Kintex-7 ● “VirtexUS” : Virtex UltraScale ● “KintexUS” : Kintex UltraScale ● “VirtexUSP”: Virtex UltraScale+ ● “KintexUSP”: Kintex UltraScale+ ● “ZYNQ7000” : Zynq-7000

2.2 Control and status register

Name	Address offset		Bit#	description
CONTROL	+00h		RW	CONTROL register (default: 0x0000_0002)
			31	interrupt enable when 1
			30	PHY reset (cause 10msec PHY_RESET_N, when written 1)
			29:2	Reserved
			1:0	MAC speed ● 2’b00: 10Mbps ● 2’b01: 100Mbps ● 2’b10: 1Gbps (*) ● 2’b11: reserved Only 2’b10 is supported
STATUS	+04h		RW	STATUS register (default: 0x0000_0000)
			31	Interrupt pending when 1. Clear when written 1.
			30	PHY_RESET_N value (0 means reset in progress)
			29	RGMII (1 means RMII, 0 means GMII)
			28:0	reserved
VERSION	+08h			RTL version (default: 32’h2018_0625)
			31:0	RTL version in hexadecimal number
	+0Ch			reserved
MAC	+10h		RW	MAC address 0
				Little-endian style MAC address [7:0] = MAC[47:40] [15:8]=MAC[39:32]

				[23:16]=MAC[31:24] [31:24]=MAC[23:16]
MAC	+14h		RW	MAC address 1
				Little-endian style MAC address [7:0]=MAC[15:8] [15:8]=MAC[7:0]
	+18~1Ch			
CONF_TX	+20h			CONFIGURATION for TX (default: 0x0)
			31:4	reserved
			3	conf_tx_no_gen_crc CRC is not attached at the end of packet when 1.
			2	conf_tx_jumbo_en Larger then 2Kbyte payload when 1
			1	conf_tx_en TX part is enabled when 1
			0	conf_tx_reset It starts 1 when system is in reset, and then goes to 0.
	+24			
	+28~2Ch			Reserved
CONF_RX	+30			CONFIGURATION for RX (default: 0x0)
			31:5	reserved
			4	conf_rx_promiscuous All packets are received when 1.
			3	conf_rx_no_chk_crc CRC is not checked when 1.
			2	conf_rx_jumbo_en Larger than 2Kbyte payload when 1
			1	conf_rx_en RX part is enabled when 1
			0	conf_rx_reset It starts 1 when system is in reset, and then goes to 0.
	+34h			
	+38~3Ch			
DES_TX	+40h			TX descriptor control
			31	push an entry when written 1. When it is 1, 'DES_TX' and 'DES_TX_SRC' is pushed at the TX Descriptor FIFO.
			31:16	the num of rooms when read (RO)
			15:0	the num of bytes of the packet (WO) A new packet is ready from the address 'DES_TX_SRC' and the number of bytes is specified by this field.
	+44h			dummy for 64-bit access
DES_TX_SRC	+48h			address of starting of the packet
			31:0	lower 32-bit

DES_TX_SRC	+4Ch			address of starting of the packet
			31:0	higher 32-bit
DES_RX	+50h			
			31	pop an entry when written 1
			31:16	the num of items when read (RO)
			15:0	the num of bytes of the packet (RO) A new packet is ready from the address 'DES_RX_SRC' and the number of bytes is specified by this field.
	+54h			dummy
DES_RX_SRC	+58h			address of starting of the packet
			31:0	lower 32-bit
DES_RX_SRC	+5Ch			address of starting of the packet
			31:0	higher 32-bit
DMA_TX_CHUNK	+60h			CHUNK for TX
			31:16	reserved
			15:0	the num of bytes for a burst
DMA_TX_FLAG	+64h			TX circular fifo status
			31:2	
			1	full flag
			0	empty flag
DMA_TX_START	+68h			TX circular queue starting address
			31:0	inclusive must be word aligned
	+6Ch			
DMA_TX_END	+70h			TX circular queue ending address
			31:0	exclusive must be word aligned
	+74h			
DMA_TX_HEAD	+78h			TX circular queue head address
			31:0	must be word aligned post increment (increment after reading)
	+7Ch			
DMA_TX_TAIL	+80h			TX circular queue tail address
			31:0	must be word aligned post increment (increment after writing)
	+84h			
	+8Ch			
DMA_RX_CHUNK	+90h			CHUNK for RX
			31:16	reserved
			15:0	the num of bytes for a burst
DMA_RX_FLAG	+94h			RX circular fifo status
			31;2	reserved
			1	full flag
			0	empty flag

DMA_RX_START	+98h			RX circular queue starting address
			31:0	inclusive must be word aligned
	+9Ch			
DMA_RX_END	+A0h			RX circular queue ending address
			31:0	exclusive must be word aligned
	+A4h			
DMA_RX_HEAD	+A8h			RX circular queue head address
			31:0	must be word aligned post increment (increment after reading)
	+ACh			
DMA_RX_TAIL	+B0h			RX circular queue tail address
			31:0	must be word aligned post increment (increment after writing)
	+B4			

2.3 Reset

'ARESETn' causes PHY RESET and logic reset (conf_tx_reset and conf_rx_reset) at the same time, while CSR reset only generates logic reset. PHY RESET by CSR write also causes corresponding logic resets.

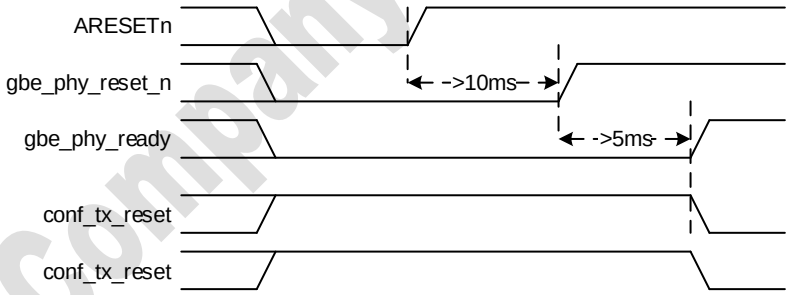


Figure 3: Reset

Logic reset (conf_tx_reset and conf_rx_reset) should remain high (i.e., active reset) for a while after PHY RESET (i.e., gbe_phy_reset_n) goes normal in order to all logic depending on GBE RX clock goes to stable reset state.

2.4 TX data path

Buffer descriptor carries information about a frame.

- ✧ Byte number (16-bit width): the number of bytes have been moved; the maximum number would be 1518[2] or 9018 depending on jumbo flag; the minimum number would be 15, i.e. one byte payload[3].
- ✧ Starting address (30-bit width): starting address of memory where the received packet has been written; lower 2 bits are not used in terms of address, since starting address should be word-aligned.

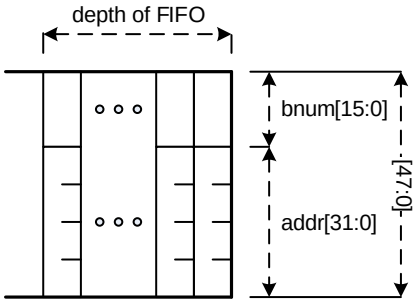


Figure 4: TX descriptor FIFO

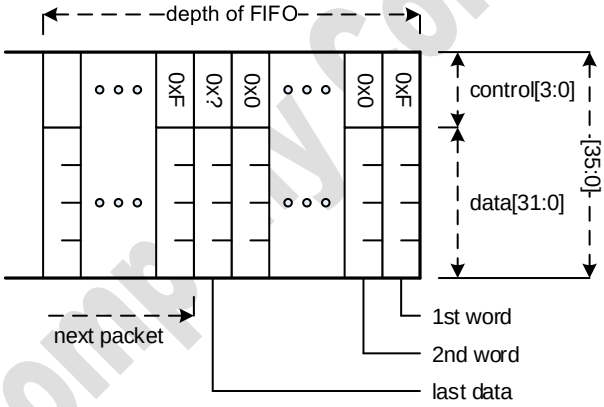


Figure 5: TX packet FIFO

[2] DST-MAC(6), SRC-MAC(6), LENG/TYPE(2), PAYLOAD(1500), CRC(4).
[3] According to Ethernet specification, payload should be 46 at least. As a result, GMII interface takes care of small payload if it is 45 or less.

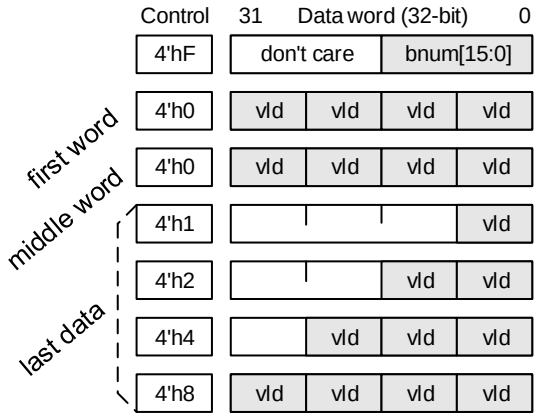


Figure 6: TX packet FIFO data example

Note that the first byte of the packet is stored in the least significant byte (LSB) position (byte 0, i.e. bits 7-0) and this means client will send the LSB-byte first.

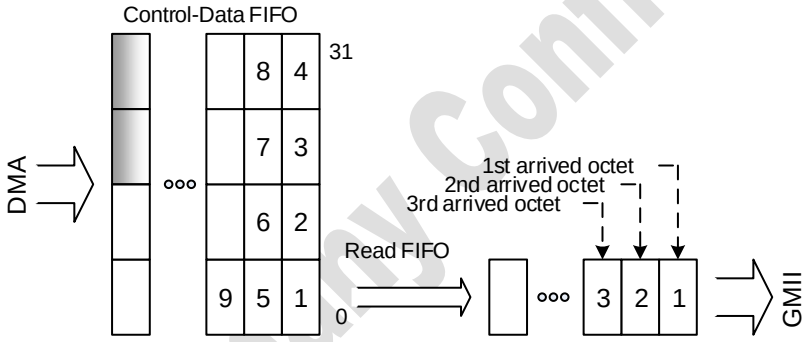


Figure 7: TX packet FIFO data order

2.5 RX data path

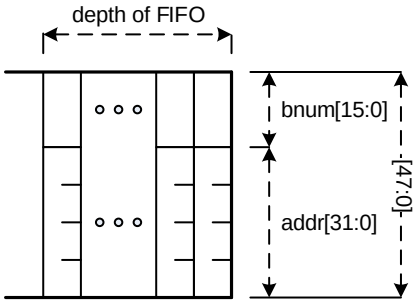


Figure 8: RX descriptor FIFO

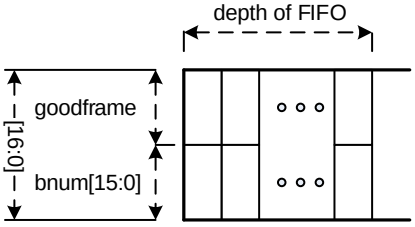


Figure 9: RX number FIFO

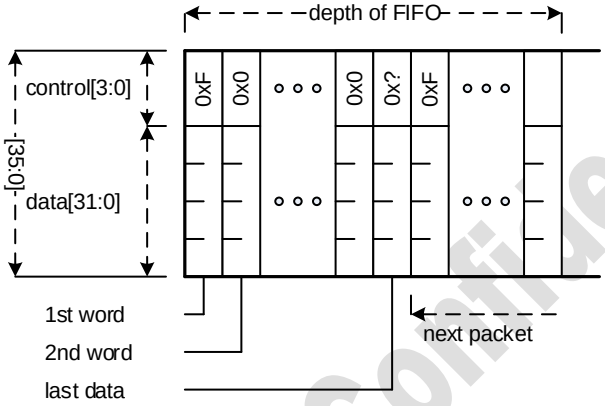


Figure 10: RX packet FIFO

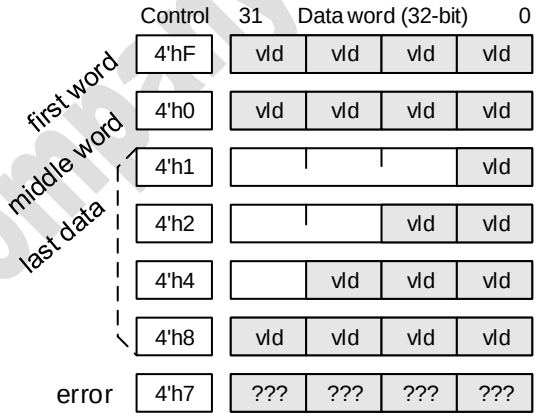


Figure 11: RX packet FIFO data example

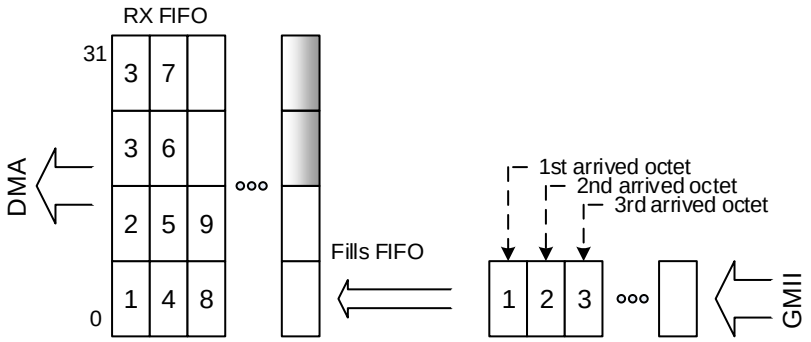


Figure 12: RX packet FIFO data order

2.6 Asynchronous FIFO

FIFO (First-In-First-Out memory queue) is a dual-port memory with built-in read- and write-addressing that reads data in the same order as it is written in, where one port is only for write and the other port is only for read. FIFO in this design uses dual-ready (i.e., ready-valid) handshaking version, which uses two signals in order to control stream style data movement between two blocks, where one is producer and the other is consumer in terms of data. The data moves from producer to consumer whenever both 'vld' and 'rdy' are high at the rising edge of clk, where 'vld' means the data is no valid and 'rdy' means the consumer is ready to accept the data.

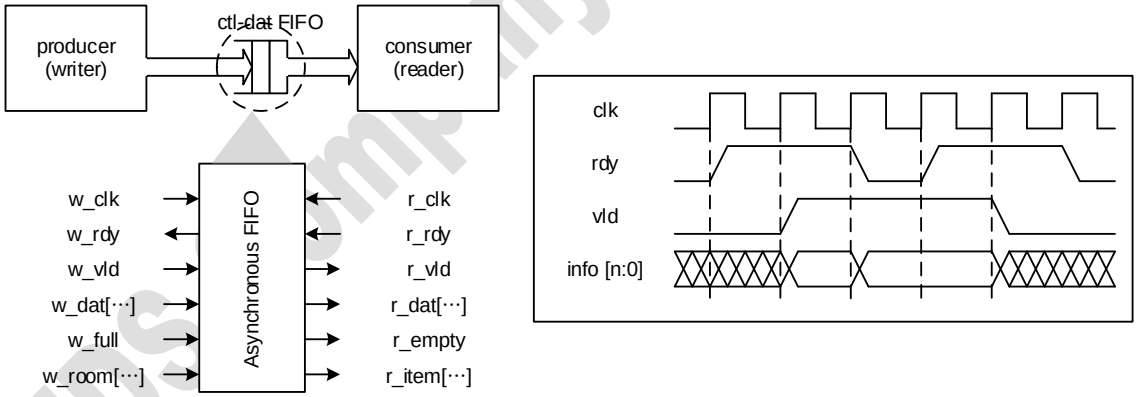
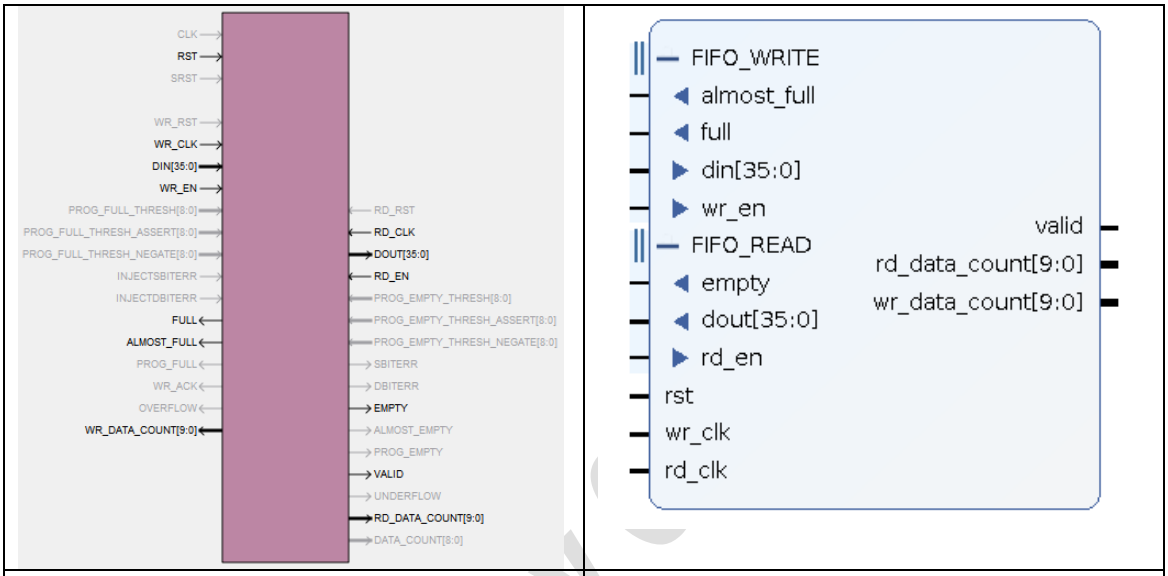


Figure 13: FIFO operation

Asynchronous FIFO should be prepared by using ISE Core Generator or Vivado IP manager and set following attributes.

- Native interface (not AXI4, no AXI Memory Mapped, AXI Stream)
- Independent clocks Block RAM
- First-Word Fall-Through
- No almost empty
- Almost full
- Valid flag for read port (Active High)

- Reset pin
- Enable reset synchronization
- No enable safety circuit[4]
- No programmable full/empty threshold constant
- Use extra logic for more accurate data counts
- Read data count (synched with read clock) = number of items
- Write data count (synched with write clock)



ISE Core Generator	Vivado IP manager

Figure 14: fifo_async_36x512

2.7 Circular queue

There are two circular queues; one for TX and the other is RX. Although the queues reside in the memory, the queue control pointers are maintained in the CSR. TX DMA only read ‘head’ and updates ‘tail’, while the software only read ‘tail’ and updates ‘head’. RX DMA only reads ‘tail’ and updates ‘head, while the software only read ‘head’ and updates ‘tail.

‘start’ points to the lowest address of buffer and ‘end’ points to the next of the highest address of buffer exclusively. ‘tail’ points to where to get the data and ‘head’ points to where to put the data. ‘tail’ and ‘head’ use post-increment discipline. When ‘tail’ and ‘head’ are the same, it is empty. When ‘head’ passes ‘tail’, it is full.

[4] This feature controls ‘wr_rst_busy’ and ‘rd_rst_busy’ pins.

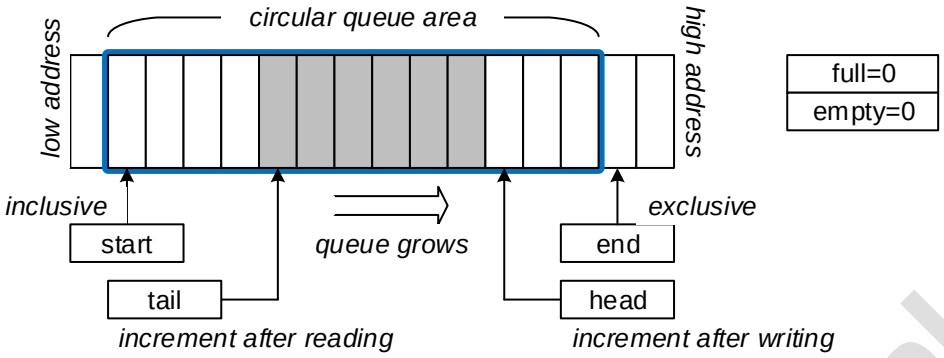


Figure 15: Circular queue

In order to make easy to implement, the buffer size would be larger than two times of the maximum packets[5].

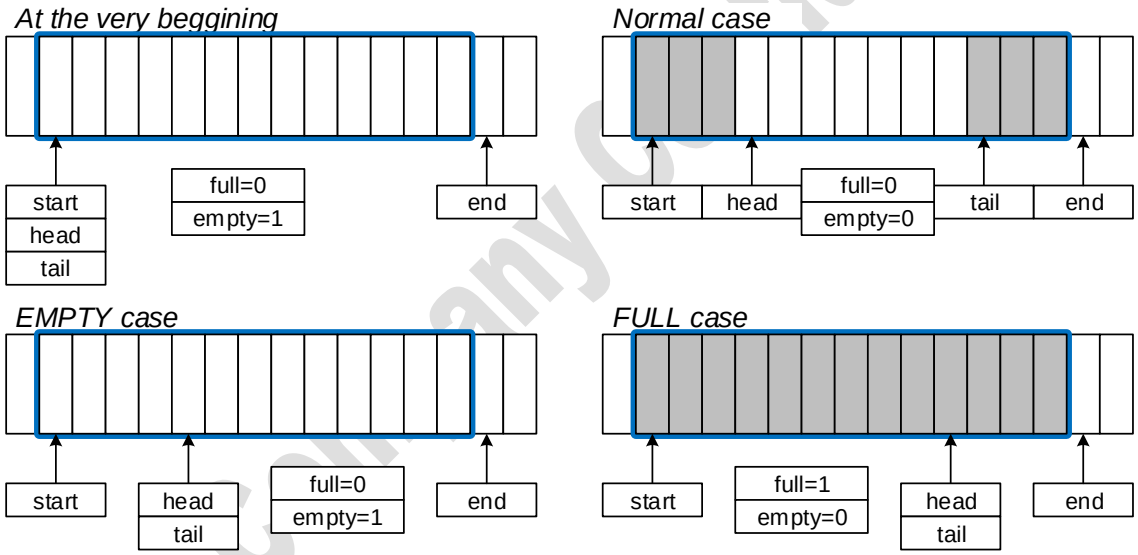


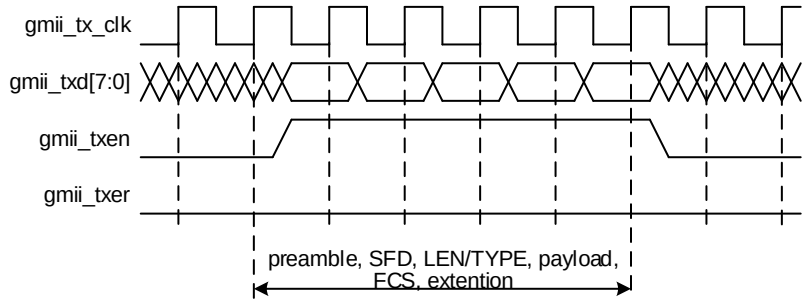
Figure 16: Circular queue operation

3 PHY interface

3.1 GMII

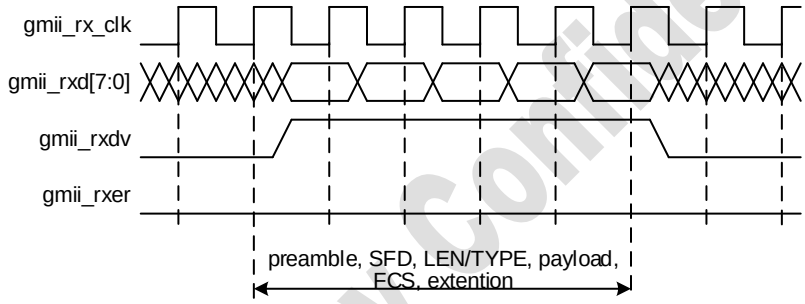
For more details, refer to reference [1]

[5] For non-jumbo, buffer size should be a larger than 1518*2 (=3036) bytes. For jumbo enabled, it should be 18036 bytes. The reason is that when ‘head’ is in the middle of buffer, there should be a space larger than a packet size between ‘head’ and ‘end’.



gmii_txen and gmii_txer are used as follows.

- ✧ {gmii_txen,gmii_txer} = 2'b00 for normal inter-frame
- ✧ {gmii_txen,gmii_txer} = 2'b00 for normal data transmission



gmii_rxdv and gmii_rxer are used as follows.

- ✧ {gmii_rxdv,gmii_rxer} = 2'b00 for normal inter-frame
- ✧ {gmii_rxdv,gmii_rxer} = 2'b01 for carrier sense
- ✧ {gmii_rxdv,gmii_rxer} = 2'b10 for normal data reception
- ✧ {gmii_rxdv,gmii_rxer} = 2'b11 for data reception error

3.2 RGMII

Not tested yet.

4 Simulation

4.1 Directory structure

directory			remarks
rtl	RTL design directory		
	verilog	gig_eth_mac_axi.v	
fifo_async	Asynchronous FIFO project		
	v6	ise14	
	z7	vivado.2017.4	

bench	Test-bench directory		
	verilog	top.v	
sim	RTL simulation directory		
	modelsim.ise		Simulation for ISE devices
	modelsim.vivado		Simulation for Vivado devices
api	c	gig_eth_mac_api.c gig_eth_mac_api.h	Reference C API

4.2 Testing structure

Figure 17 shows hardware structure to simulate ‘gig_eht_mac_axi’ module.

- tester_tx: It generates testing scenario through AMBA AXI accesses.
  - ✧ See ‘bench/verilog/tester_tx.v’
- tester_rx: It receives packet through AMBA AXI accesses.
  - ✧ See ‘bench/Verilog/tester_rx.v’
- gig_eth_mac_axi: Device under test
- PNYsimply forwards packets in a loopback fashion.
- bram_axi_dual: Dual-port memory for TX and RX packets.
- AMBA AXI: AMBA AXI bus

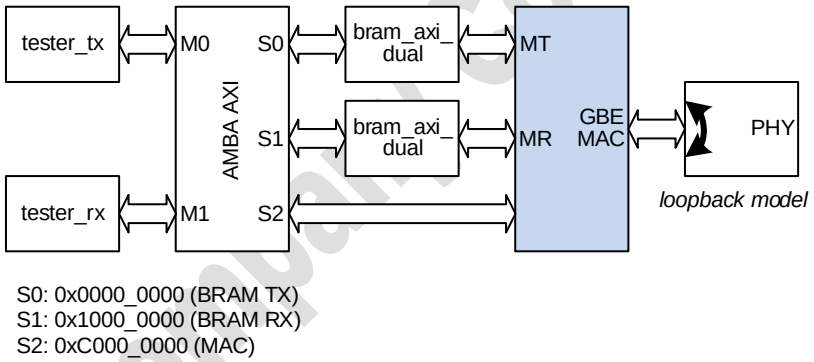


Figure 17: Testing structure

4.3 Simulation

Simulation requires Mentor Graphics ModelSim and Xilinx ISE or Vivado suites.

- ✧ HDL simulator: Mentor Graphics ModelSim
- ✧ FPGA related library: ISE 14 for Virtex-6 or Vivado 2017.4 for Zynq

4.3.1 For ISE supporting FPGA

This includes Spartan 6 or Virtex 6. It requires ‘XILINX’ environment variable to point ISE installation directory[6], such as ‘/opt/Xilinx/14.7/ISE_DS/ISE’ for Linux or ‘C:/Xilinx/14.7/ISE_DS/ISE’ for Windows.

[6] The directory depends of how Xilinx ISE was installed.

1. Go to ‘sim/modelsim.ise’ directory
2. Run ‘make’ for Linux or ‘RunMe.bat’ for Windows
   - ✧ If any errors occurs, have a close look at the message.
   - ✧ Reasons of most of errors may be come from mismatches including path or environment variables.
   - ✧ Simulation version specific options may cause problems.
3. Invoke VCD waveform viewer to check simulation if simulation completes without any errors.
   - ✧ GTKwave would be a good choice

4.3.2 For Vivado supporting FPGA

This includes Zynq 7000. It requires ‘XILINX’ environment variable to point ISE installation directory, such as ‘/opt/Xilinx/Vivado/2017.4’ for Linux or ‘C:/Xilinx/Vivado/2017.4’ for Windows.

1. Go to ‘sim/modelsim.vivado’ directory
2. Run ‘make’ for Linux or ‘RunMe.bat’ for Windows
   - ✧ If any errors occurs, have a close look at the message.
   - ✧ Reasons of most of errors may be come from mismatches including path or environment variables.
   - ✧ Simulation version specific options may cause problems.
3. Invoke VCD waveform viewer to check simulation if simulation completes without any errors.
   - ✧ GTKwave would be a good choice

4.4 Simulation scenario

‘sim_define.v’ in the ‘sim/modelsim.ise’ or ‘sim/modelsim.vivado’ specifies which testing scenario is used. For each scenario, see ‘bench/verilog/tester_tx.v’ for details.

Define corresponding macro as ‘1’ to test, and otherwise define ‘0’.

- TEST_MEM: Check AMBA AXI bus by read-after-write testing on buffer memory
- TEST_MAC_CSR: Check after read all the registers in the ‘gig_eth_mac_axi.
- TEST_MAC_TX_SHORT_PAD
- TEST_MAC_TX_NORMAL
- TEST_MAC_TX_LONG
- TEST_MAC_TX_NORMAL_RANDOM
- TEST_MAC_RX_CRC_ERROR
- TEST_MAC_RX_CRC_ERROR_LONG
- TEST_MAC_TX_LONG_DROP
- TEST_MAC_TX_LONG_DROP_LONG
- TEST_MAC_TX_SHORT_PAD_MANY_DROP_AT_RX

4.4.1 AMBA AXI tasks

'bench/verilog/axi_tasks.v' contains AMBA AXI tasks.

- axi_write: generates AMBA AXI write transaction
- axi_read: generates AMBA AXI read transaction

4.4.2 Ethernet MAC control tasks

'bench/Verilog/gig_mac_tasks.v' contains contril tasks for 'gig_eth_mac_axi'.

4.4.3 Add new scenario

Add a new scenario in 'bench/Verilog/tester_tx.v'.

5 References

[1] IEEE, IEEE Std. 802.3-2008, Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and Physical Layer specifications, Section 2.
[2] ARM, AMBA Specification (Rev 2.0), ARM IHI 0011A, 1999.
[3] Future Design Systems, High-availabilty Seamless Redundancy Controller on Gigabit Ethernet, FDS-TD-2018-10-002.

Wish list

- ☐

Revision history

- ☐ 2018.10.10: Updated by Ando Ki.
- ☐ 2018.7.7: Started by Ando Ki (adki@future-ds.com)

– End of document –