High-availability Seamless Redundancy Controller on Gigabit Ethernet

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 of IEC 62439-3 HSR (High-availability Seamless Redundancy) controller on Gigabit Ethernet.

Table of Contents

1 Overview

High-availability Seamless Redundancy (HSR) is a network protocol for Ethernet that provides seamless failover against failure of any network component. This 'gig_eth_hsr' is an implementation of IEC 62439-3 (HSR) with QR (Quick Remove) feature, which reduces up to 50% network traffic. HSR module includes Double Attached Node (DANH) and Redundancy Box (RedBox), the former is an HSR-capable (HSR forwarding and HSR tagging capabilities) end node having two HSR ports and the latter is an HSR-capable switch having two HSR ports and one normal port, and enables non-HSR node to be attached at the HSR network.

'gig_eth_hsr' block provides Gigabit Ethernet Media Access Controller functions for HSR, which can be configured as DANH and RedBox. As shown in Figure 1, it requires Gigabit Ethernet MAC for DANH.

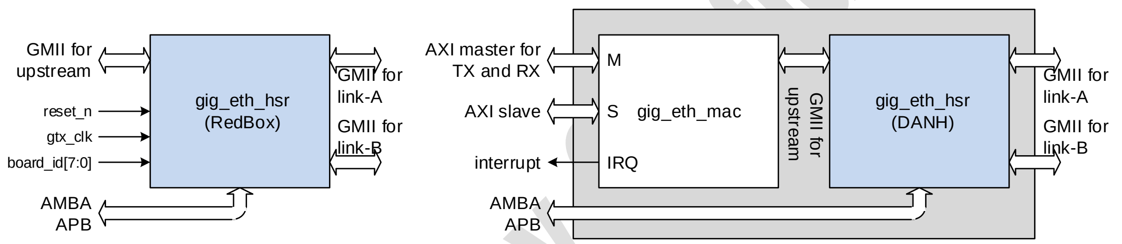


Figure 1: Overview

2 Structure

As shown in Figure 2, HSR consists of three GMII ports and other blocks.

- GMII port for upstream
- GMII port for HSR link-A
- GMII port for HSR link-B
- Configuration and status block with AMBA APB interface
- Proxy table for RedBox
- Quick remove table for HSR links

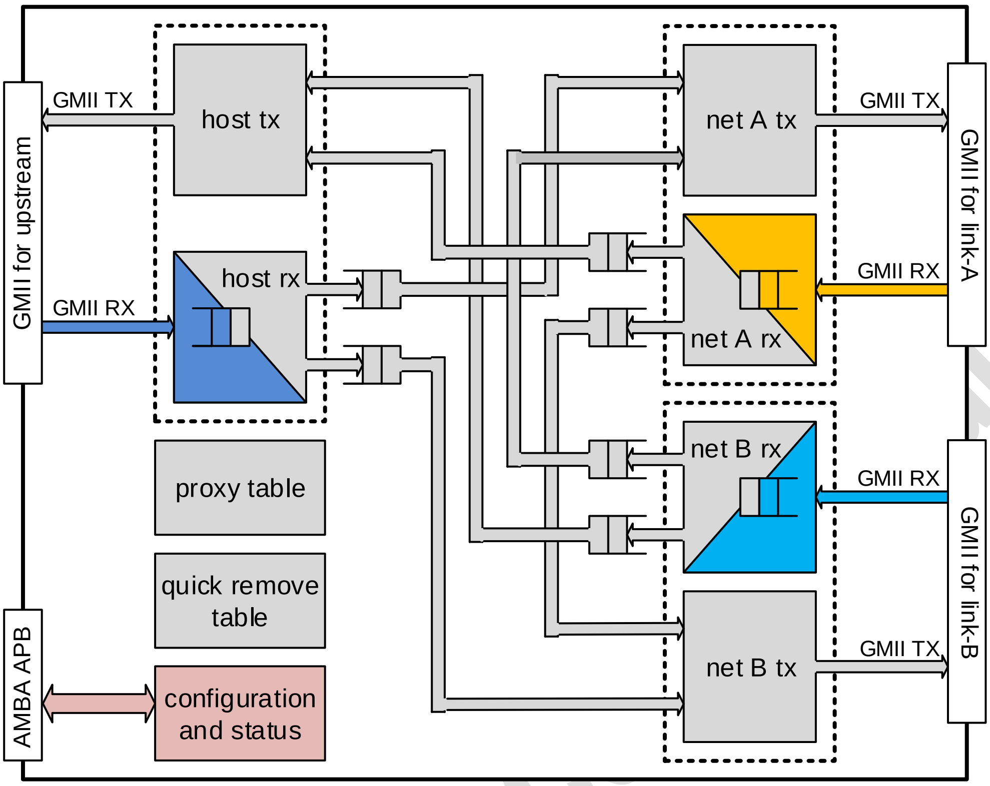


Figure 2: Internal structure

2.1 Block pin signals

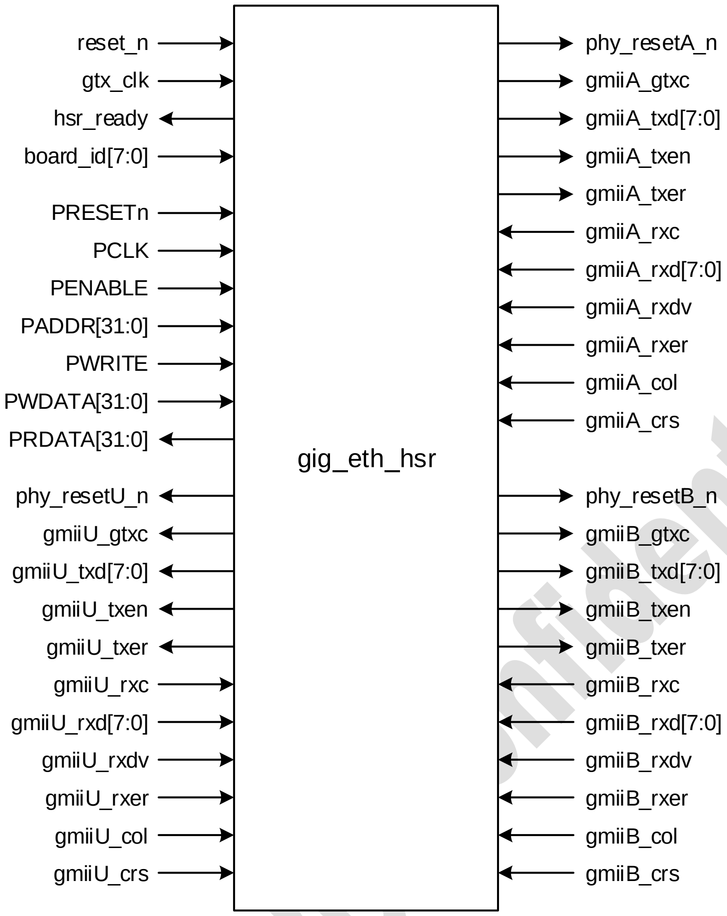


Figure 3: Block pin signals

'reset_n' causes PHY reset and 'hsr_ready' will go high after >15msec.
'gtx_clk' should be 125Mhz.
'hsr_ready' will be high when all PHYs are passed reset processes and it will not be high without PHY.
'board_id' will be used as a lower 8-bit of default MAC address.
'PCLK' should be the same as 'gtx_clk' at this implementation.

2.2 Macros and parameters

Table 1: Macros

Macros	Note
SIM or SYN	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
ISE or VIVADO	ISE: for Xilinx ISE devices such as Spartan-6, Virtex-6

	VIVADO: for Xilinx Vivado device such as Zynq-7000, UltraScale, UltraScale+
XILINX_Z7	For ZYNQ700
GIG_ETH_HSR_FIFO_SYNC_36x512_V	It makes include ‘gig_eth_hsr_fifo_sync_36x512.v’ for HW FIFO.
HSR_PERFORMANCE	It makes gather performance statistics of HSR-related activities. (not fully verified yet) Note that ‘PCLK’ should be the same as ‘gtx_clk’ to use this feature.

Table 2: Parameters

Parameter	Note
CONF_HSR_QR	Quick remove feature enabled when 1
CONF_SNOOP	Pass all data to the host upstream when 1
CONF_DROP_NON_HSR	Drop non-HSR packet when 1
CONF_PROMISCUOUS	Accept all incoming packet when 1 Proxy table will not work since destination match occurs always
CONF_HSR_PATH_ID0	Specify which link will be path id 0, e.g., “A” or “B”
CONF_HSR_NET_ID	Set HSR network ID 3-bit This can be overridden by AMBA APB configuration and status register.
CONF_MAC_ADDR	Default MAC address Valid for DANH only. Lower 8-bit will be determined by ‘board_id[7:0]’ pins. This can be overridden through AMBA APB configuration and status path.
NUM_ENTRIES_QR	Number of entries of quick remove table It should be a power of 2.
NUM_ENTREIS_PROXY	Number of entries of proxy table It is valid only for RedBox. It should be a power of 2.
NET_RX_FIFO_DEPTH_DAT	The number of entries to hold the data by this FIFO. Depth of data FIFO to deal with incoming data from HSR link. It should reflect the maximum packet size; 512 for 2K-byte packet.
HOST_RX_FIFO_DEPTH_DAT	The number of entries to hold the data by this FIFO. depth of data FIFO to deal with incoming data from upstream port it should reflect the number of bytes for the maximum payload of the packet
HOST_RX_FIFO_DEPTH_NUM	depth of length FIFO to deal with incoming data from upstream port. It should reflect the maximum packet size; 512 for 2K-byte packet.
HSR_FIFO_DEPTH	The number of entries to hold the data by this FIFO. depth of data FIFO to deal with incoming data from HSR link It should reflect the number of bytes for the maximum payload of the packet
DANH_OR_REDBOX	Determines DANH or RedBox. “DANH” or “REDBOX”

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

2.3 Configuration and status register

Name	Address offset		Bit#	description
VERSION	+00h		RO	RTL version (default: 0x2018_1001)
			31:0	RTL version
Reserved	+04h			
Reserved	+08h			
Reserved	+0Ch			
MAC address 0	+10h		RW	MAC address 0
				MAC address [7:0] = MAC[47:40] [15:8]=MAC[39:32] [23:16]=MAC[31:24] [31:24]=MAC[23:16] default value will be ‘CONF_MAC_ADDR[47:16]’
MAC address 1	+14h		RW	MAC address 1
				MAC address [7:0]=MAC[15:8] [15:8]=MAC[7:0] default value will be ‘CONF_MAC_ADDR[15:8], board_id[7:0]’
HSR net id	+18h		RW	HSR network ID (default: 0x0)
			31:3	reserved
			2:0	3-bit HSR network ID
CONTROL	+1C		RW	Control and status (default: 0x0000_0006)
			31	HSR type (RO) 1 for “DANH” or 0 for “REDBOX”
			30:4	reserved
			3	snoop

				Pass all data to the host (upstream) without removing HSR header, when 1
			2	hsr_qr Enable quick remove feature when 1
			1	drop_non_hsr Drop all non-HSR packet incoming from HSR link when 1
			0	promiscuous MAC destination address match always when 1
PHY	+20h		RW	PHY related
			31:7	reserved
			6	HSR ready when 1 (RO)
			5	reflect PHY for upstream readiness (RO) 1 means ready
			4	reflect PHY for link-B readiness (RO) 1 means ready
			3	reflect PHY for link-A readiness (RO) 1 means ready
			2	reflect 'PHY_RESET_U_IN' port when read start PHY RESET sequence for link-B when 1 is written
			1	reflect 'PHY_RESET_A_IN' port when read start PHY RESET sequence for link-A when 1 is written
			0	reflect 'PHY_RESET_U_IN' port when read start PHY RESET sequence for upstream when 1 is written
PROXY	+24h		RO	num of entries of Proxy table
			31:0	num of entries of Proxy table
QR	+28h		RO	num of entries of QR table
			31:0	num of entries of QR table

2.4 FIFO depth

There are 10 FIFOs in the design and depth of each FIFO is determined by its corresponding parameter.

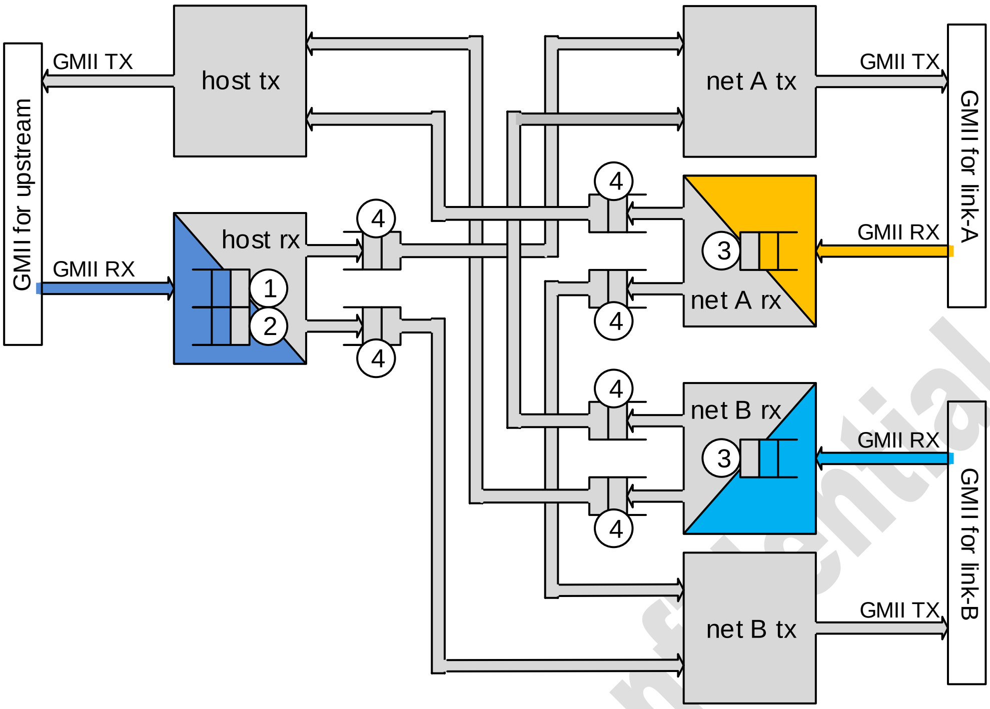


Figure 4: FIFO depth

① HOST_RX_FIFO_DEPTH_DAT: It holds packet from upstream port.
② HOST_RX_FIFO_DEPTH_NUM: It holds DMA descriptors.
③ NET_RX_FIFO_DEPTH_DAT: It holds packet from link-A/B.
④ HSR_FIFO_DEPTH: It holds packets for routing.

1 and 3 depend on the maximum size of packet to handle and it will be 2K-byte by default, where 2K-byte corresponding 512 entries since each entry contains 4 bytes of data.

2.5 Clock domains

All blocks are running with ‘gtk_clk’ excepting GMII receiving sub-ports that operate in its port input clock.

There are 5 clock domains.

- 125Mhz gtx_clk
- 125Mhz host rx clock
- 125Mhz HSR link-A clock
- 125Mhz HSR link-B clock
- 125Mhz AMAB APB PCLK

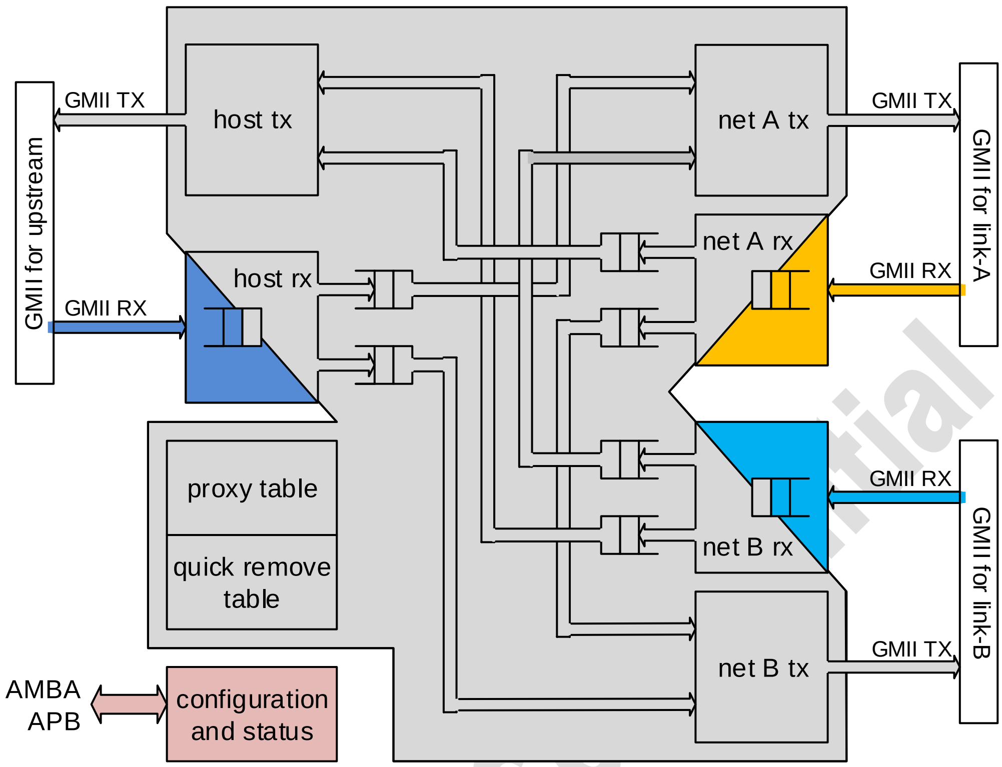


Figure 5: Clock domains

2.6 Upstream

Figure 6 shows upstream block, in which proxy table is required for RedBox configuration.

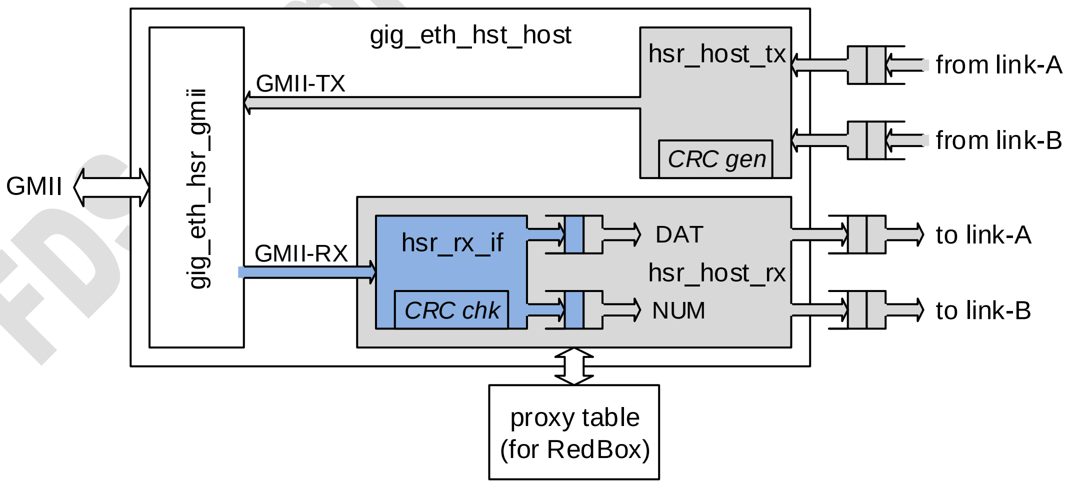


Figure 6: Upstream blocks

'gig_eth_hsr_host_tx' block gets HSR packets from two FIFO's and then drives through GMII-TX after treating the packet.

- add preamble and SFD,
- removes HSR-fields[1] and
- appends CRC

Link-A has higher priority. It does not care of small packet, which means it does not add or remove any padding for less than 46-byte long payload.

'hsr_rx_if' block gets packet and then pushes the packet data to the asynchronous data FIFO.

- removes preamble and SDF, and
- checks CRC and removes it form the packet

At the end of each packet, it also pushes the number of bytes of the packet to the asynchronous number FIFO.

'hsr_host_rx' adds HSR-fields and pushes the two FIFO's for HSR link-A and link-B after duplicating the HSR packet. It starts to push the packet data when num FIFO is ready since the HSR field needs payload size. It drops the whole packet when any of FIFO for link-A and link-B is full.

When this block is configured as DANH, 'hsr_rx_if' maintains proxy table by storing source mac address of the packet.

2.7 Downstream

Figure 7 shows downstream block, which takes care of one of two HSR link-A and link-B.

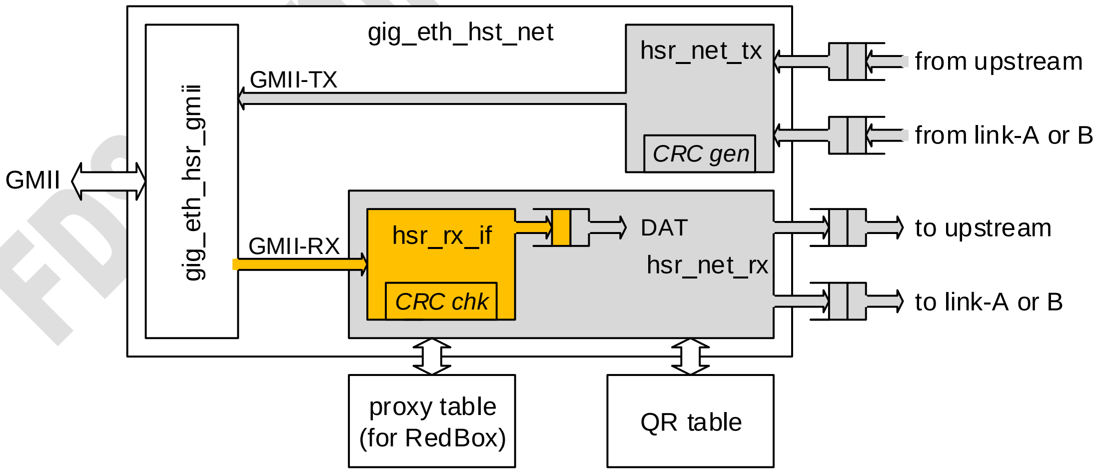


Figure 7: Downstream block

[1] HSR type ('0x892F') and HSR size, and HSR sequence fields

‘hsr_net_tx’ block gets HSR packets from two FIFO’s and drives the packet through GMII-TX.

- add preamble and SDF,
- appends CRC

Link has higher priority, i.e., frames in the ring have a higher priority than inserted frames from the host. It does not care of small packet, which means it does not add or remove any padding for less than 46-byte long payload.

‘hsr_rx_if’ block gets packet and then pushes the packet data to the asynchronous data FIFO.

- removes preamble and SDF, and
- checks CRC and removes it form the packet

The number FIFO is not used for this block.

‘hsr_host_rx’ gets packet from ‘hsr_rx_if’.

- forwards the packet to the link FIFO, and
- pushes the packet to the upstream FIFO after removing HSR fields.

This block also interacts with the proxy table to figure out the packet for the upstream.

This block maintains QR table to remove duplicated packets.

2.8 Reset

Figure 8 (upper) depicts how external reset works, while lower shows how internal CSR reset works.

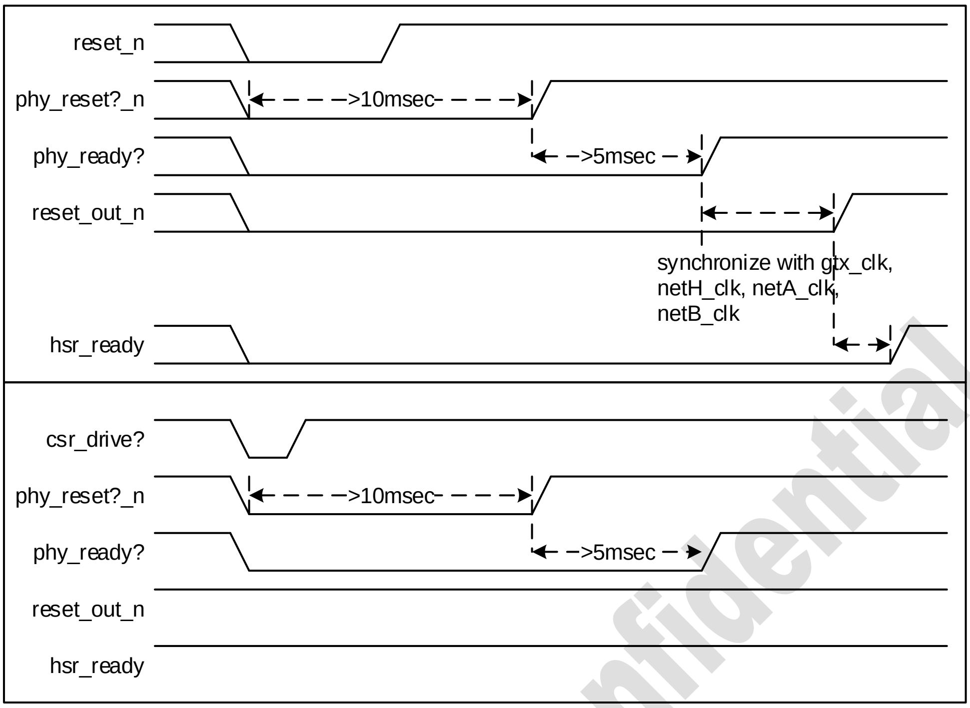


Figure 8: Reset

There is one logic reset (reset_out_n) that makes all logic to be reset state.

External reset input (reset_n) forces all logic and Ethernet PHY to be reset state, while internal PHY reset (csr_drive) driven by CSR write only drives external PHY reset signal.

It should be noted that PHY reset makes stop PHY RX clock so that some logic depending on this clock does not go to its reset state if 'reset_out_n' goes to high before receiving PHY RX clock.

2.9 FIFO structure

There are many FIFOs between blocks and these FIFOs use following conventions. All synchronous and asynchronous FIFO shares the same conventions.

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.

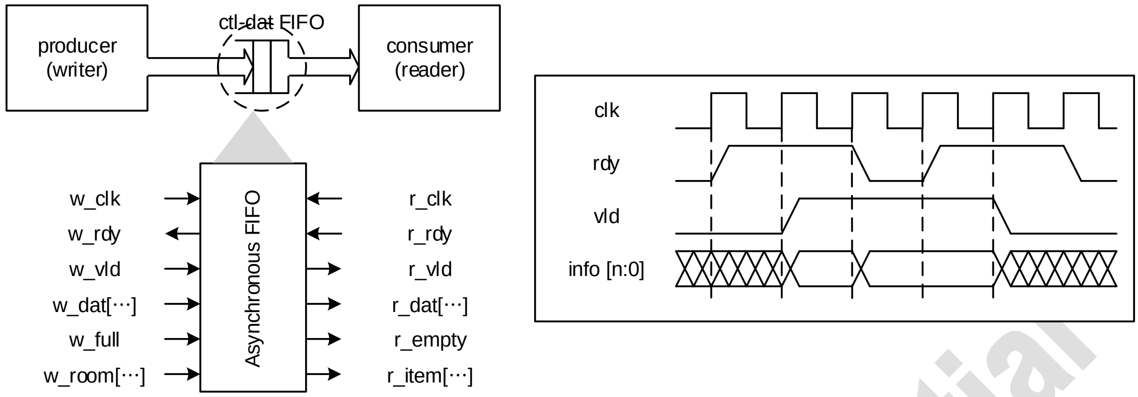


Figure 9: FIFO operation

There are two types of FIFO.

- Data FIFO: It carries packet data in 4-byte format along with control information.
- Number FIFO: It carries the number of bytes of the packet and status information.

Data FIFO consists two fields.

- Four-byte data
- Four-bit control

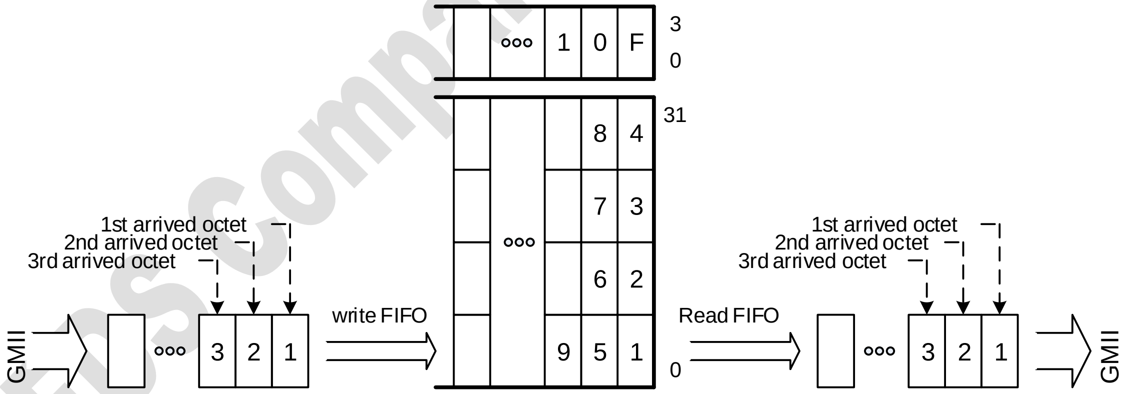


Figure 10: Data FIFO scheme

The control field is used to distinguish first and last word from each other, where last words determine the end of the packet. The control field is 0xF for the first word and control filed is reset to 0 during normal data, and then at the last word of the packet, the control filed will indicate which byte is the last byte in the last word. This is done by setting a 1 in the position of the last byte.

- ✧ control field 0xF: first data
- ✧ control field 0x0: data with 4-byte valid

- ✧ control field 0x1: last data with 1-byte valid at [7:0]
- ✧ control filed 0x2: last data with 2-byte valid at [15:0]
- ✧ control filed 0x4: last data with 3-byte valid at [23:0]
- ✧ control field 0x8: last data with 4-byte valid at [31:0]
- ✧ control filed 0x7: last data and something wrong.

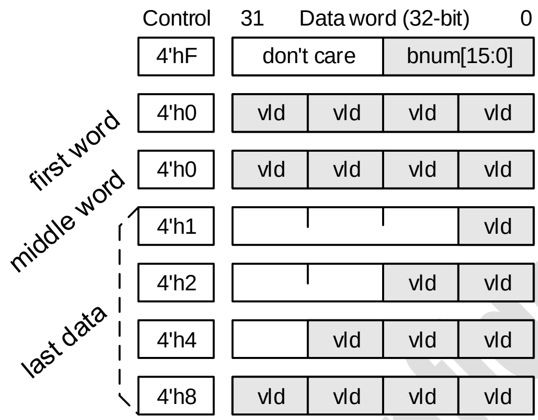
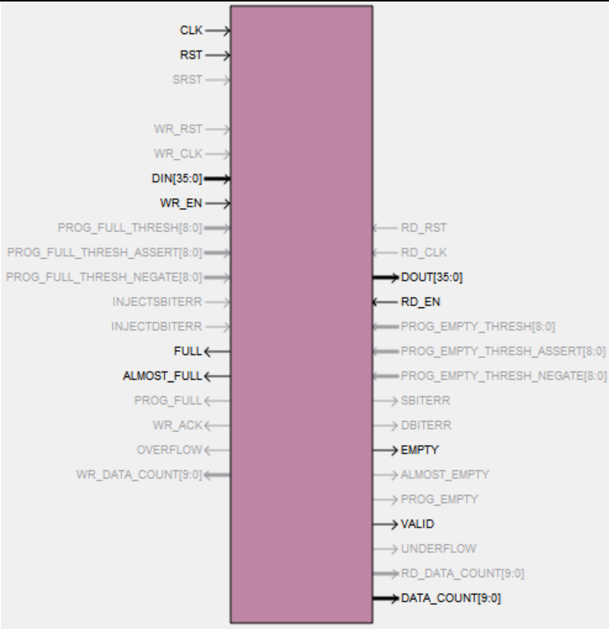


Figure 11: Data FIFO example

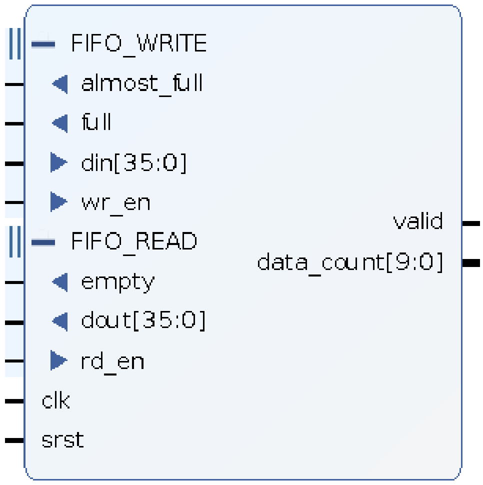
2.9.1 Synchronous FIFO for Xilinx devices

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

- Native interface (not AXI4)
- Common clock Block RAM
- First-Word Fall-Through
- No almost empty
- Almost full
- Valid flag for read port (Active High)
- No programmable full/empty threshold constant
- Use extra logic for more accurate data counts
- Data count (synched with clock) = number of items



ISE Core Generator



Vivado IP manager

Figure 12: fifo_sync_36x512

2.9.2 Asynchronous FIFO for Xilinx devices

Synchronous 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[2]
- 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)

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

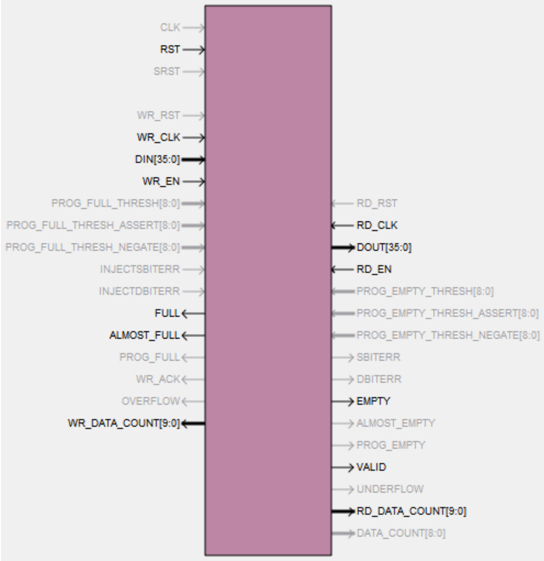
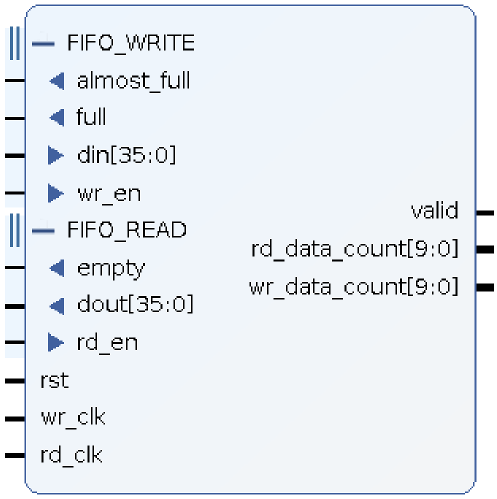
	
ISE Core Generator	Vivado IP manager

Figure 13: fifo_async_36x512

2.10 Proxy table

RedBox configuration requires proxy table, which keeps the information about all nodes that are attached to the host port of the RedBox.

- The table stores the source MAC addresses of all frames that have been received via host receiving port.
- The MAC addresses in the table are compared with the source MAC of the packet from HSR Link-A/B. If match occurs, the packet was issued by this module and it should be removed.
- The MAC address in the table are compared with the destination MAC of the packet from HSR Link-A/B. If match occurs, the packet should be forwarded to the host port.

This table is managed in a fashion of circular queue.

2.11 Quick removal table

Quick removing removes duplicated and circulated packet from the HSR ring.

Quick removal table keeps the information about HSR packets arrived at the node and it is used to remove the following packets that are actually redundancy packets. This implementation keeps source MAC address and HSR sequence number.

3 Simulation

3.1 Directory structure

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

3.2 Testing structure

Figure 14 shows hardware structure to simulate 'gig_eht_hsr' module.

- tester: It generates testing scenario through AMBA APB accesses and GMII packets.
  - ✧ See 'bench/Verilog/tester_gmii.v'
  - ✧ apb_task.v: AMBA APB tasks
  - ✧ task_eth_ip_tcp_udp.v: GMII tasks for Ethernet frames
- gig_eth_hsr: Device under test
- simple_phy: It models Gigabit Ethernet PHY and simply forwards packets.

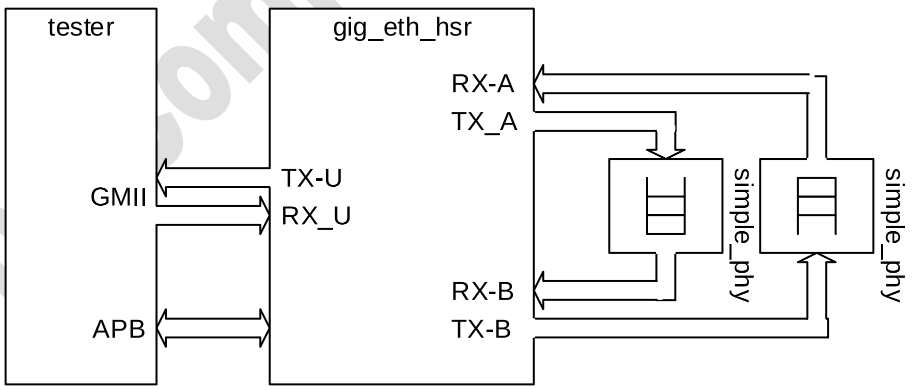


Figure 14: Testing structure

3.3 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

3.3.1 For ISE supporting FPGA

This includes Spartan 6 or Virtex 6. It requires ‘XILINX’ environment variable to point ISE installation directory[3], such as ‘/opt/Xilinx/14.7/ISE_DS/ISE’ for Linux or ‘C:/Xilinx/14.7/ISE_DS/ISE’ for Windows.

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

3.3.2 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

3.4 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_gmii.v’ for details.

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

- TEST_CSR: Check after read all the registers in the ‘gig_eth_hsr’.
- TEST_SHORT_SINGLE_PACKET: prepare a packet with payload less than 46 bytes and send it, where ‘bnum_payload’ determines the number of bytes of payload.

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

- TEST_SHORT_PACKETS: prepare a packet with payload less than 46 and send it several times, where ‘bnum_payload’ determines the number of bytes of payload and ‘tmp_bnum’ determines the number of packets.
- TEST_LONG_SINGLE_PACKET: prepare a packet with payload more than 46 bytes and send it, where ‘bnum_payload’ determines the number of bytes of payload.
- TEST_LONG_PACKETS: prepare packets with payload more than 46 bytes and send them, where ‘bnum_payload’ determines the number of bytes of payload.

Figure 15 shows an example of ‘TEST_SHORT_SINGLE_PACKET’ case, where a short normal packet is received through upstream port, and two HSR packets are forwarded to the Link A and B, and eventually these HSR packets are received through Link A and B. These received HSR packets are not forwarded any ports since these are matched with source MAC address.

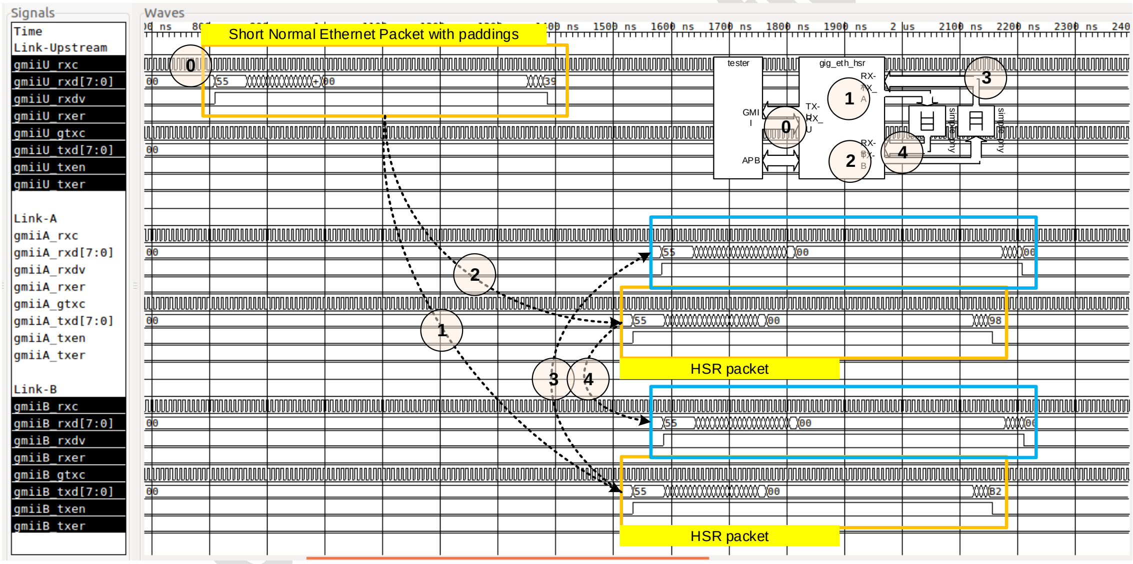


Figure 15: TEST_SHORT_SINGLE_PACKET case

3.4.1 AMBA APB tasks

‘bench/verilog/apb_tasks.v’ contains AMBA APB tasks.

- apb_write: generates AMBA APB write transaction
- apb_read: generates AMBA APB read transaction

3.4.2 Ethernet packet tasks

‘bench/Verilog/task_eth_ip_tcp_udp.v’ contains GMII Ethernet tasks.

- build_ethernet_packet: prepare an Etherent packet frame
- send_ethernet_packet: send Ethernet packet through GMII port

- crc_gen: generates 32-bit CRC
- crc_chk: check 32-bit CRC

3.4.3 Add new scenario

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

4 RedBox: Redundancy Box

As shown in Figure 16 'gig_eth_hsr' provides three GMII ports.

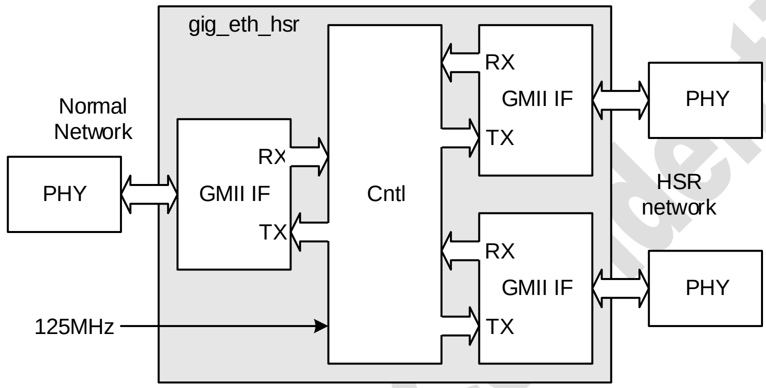


Figure 16: RedBox usage

5 DANH: Double Attached Node with HSR

As shown in Figure 17, DANH consists of 'gig_eth_hsr' and 'gig_eth_mac'.

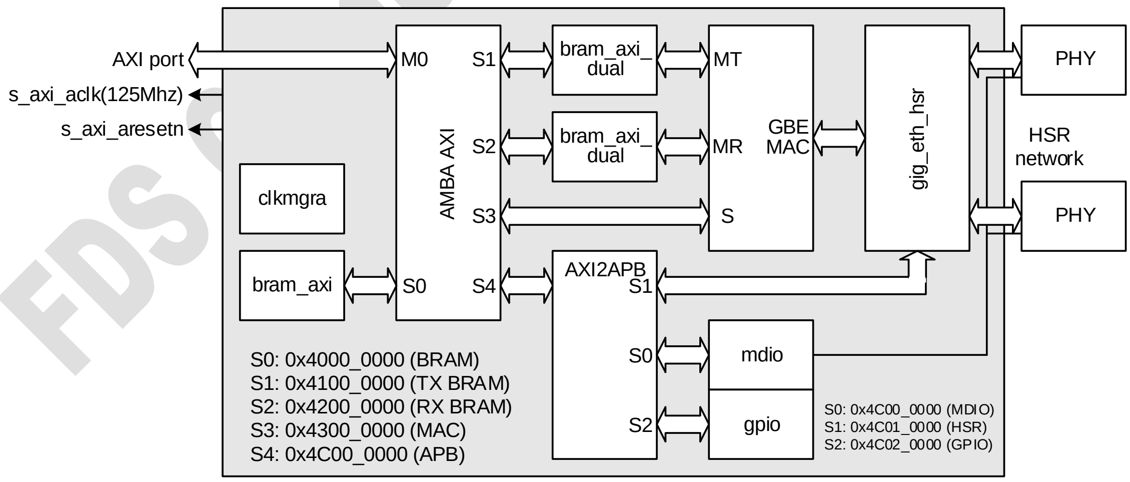
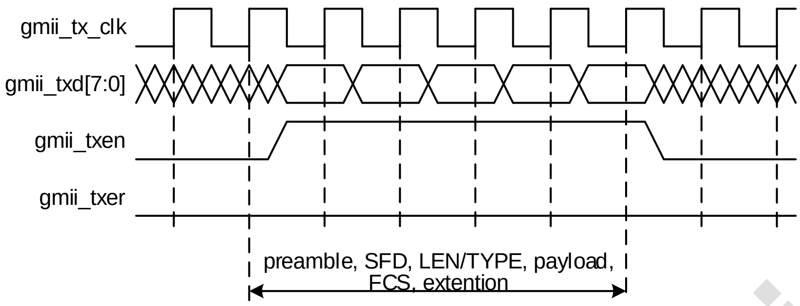


Figure 17: DANH usage

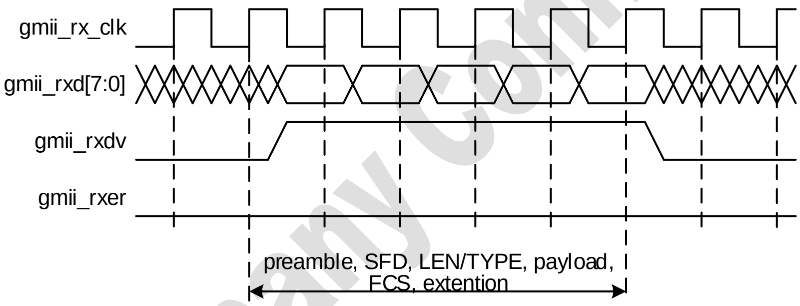
6 GMII

For more details, refer to reference [2].



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

7 Ethernet packet formats

As shown in Figure 18, 6-octet of HSR header is inserted at the normal or VLAN packet.

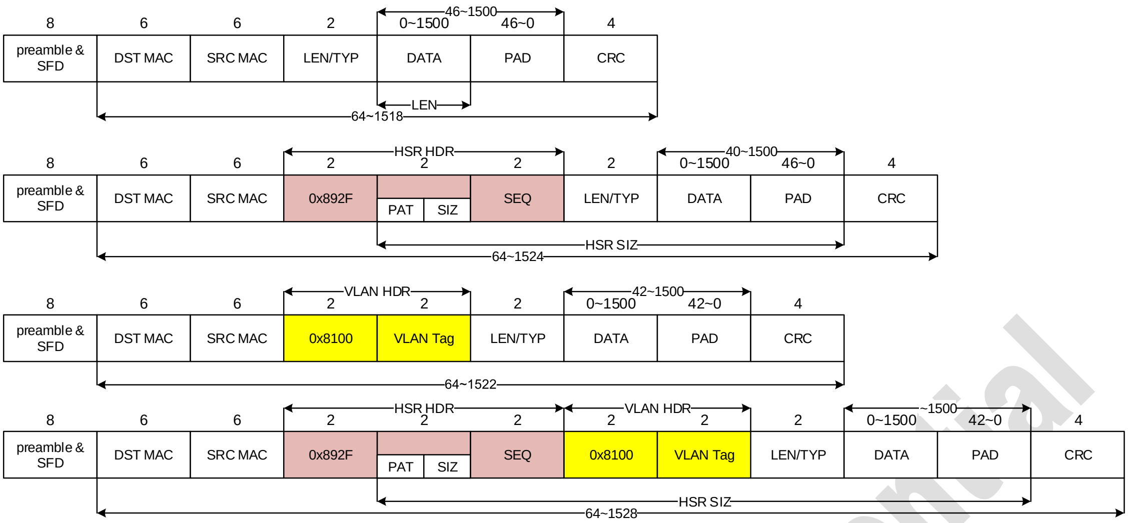


Figure 18: Ethernet packet formats

8 HSR network

Figure 19 shows an example setup to build a complete HSR system.

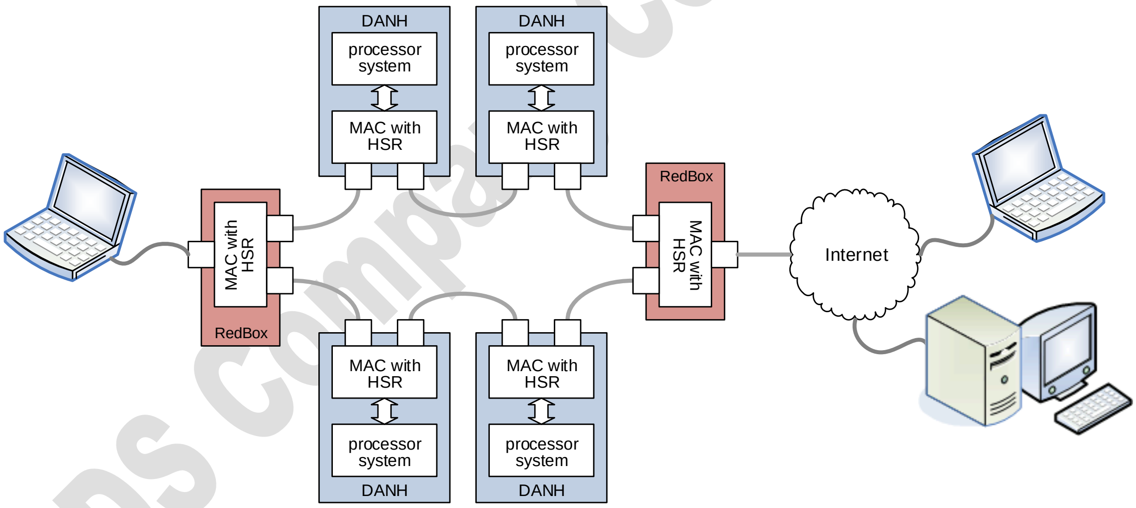
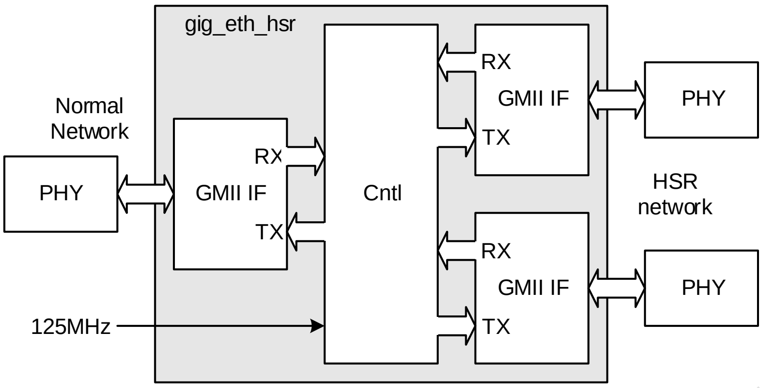


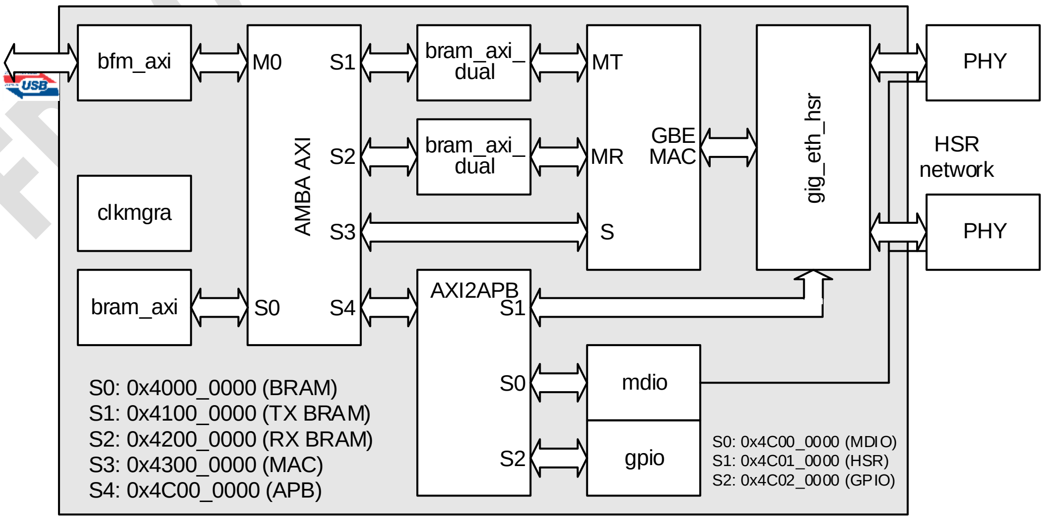
Figure 19: HSR network example

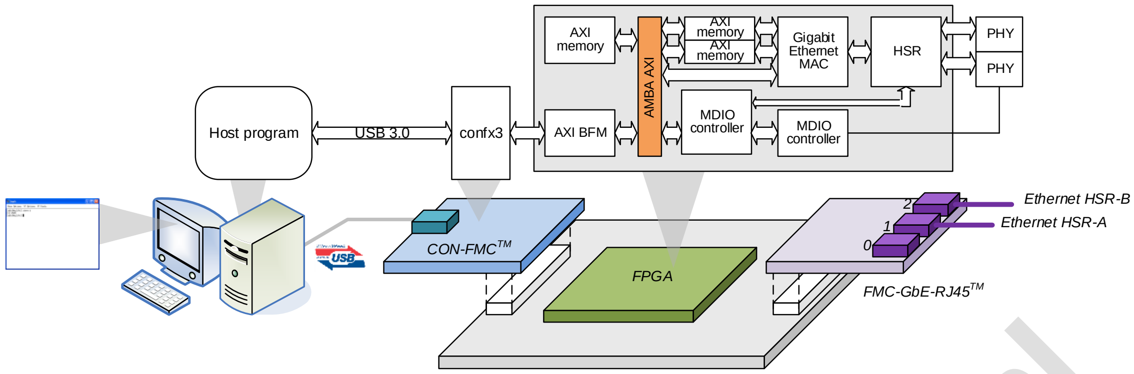
9 Testing

9.1 RedBox

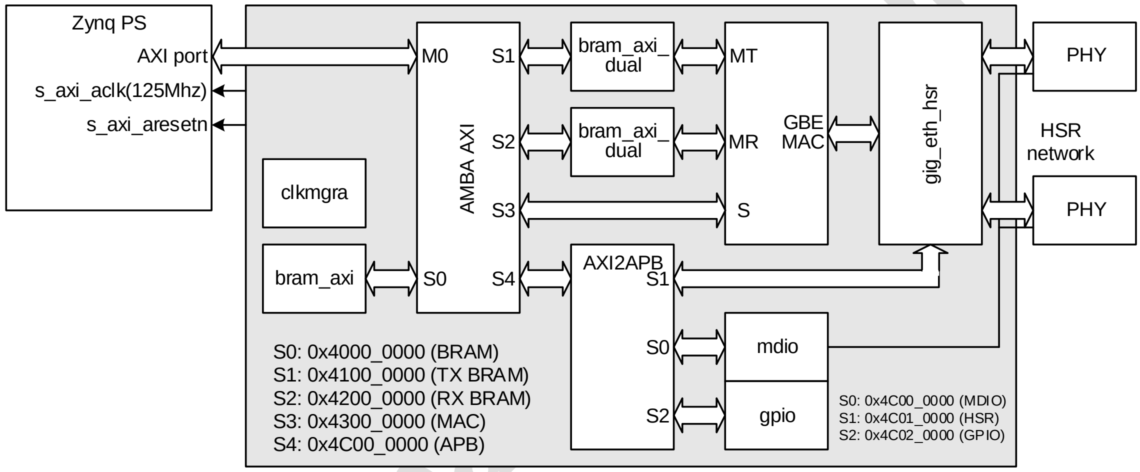


9.2 Host-driven DANH





9.3 ARM-driven DANH



10 References

[1] IEC 62439-3, Industrial communication networks – High availability automation networks – Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR), Edition 2.0, 2012-07.

[2] 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.
[3] ARM, AMBA Specification (Rev 2.0), ARM IHI 0011A, 1999.
[4] Future Design Systems, Gigabit Ethernet Media Access Controller, FDS-TD-2018-10-001.

Wish list

- ☐ Supporting jumbo packet up to 9Kbyte payload
- ☐ Supporting 2K entries of proxy table.
- ☐ Supporting LRU (least recently used) policy for Proxy table and quick remove table.

Revision history

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

– End of document –