

MAC on the HUB

Y. Ermoline, 20.10.2017
V0.1

This note describe design steps of the MAC on HUB FPGA to work with Ethernet for IPbus.

Contents:

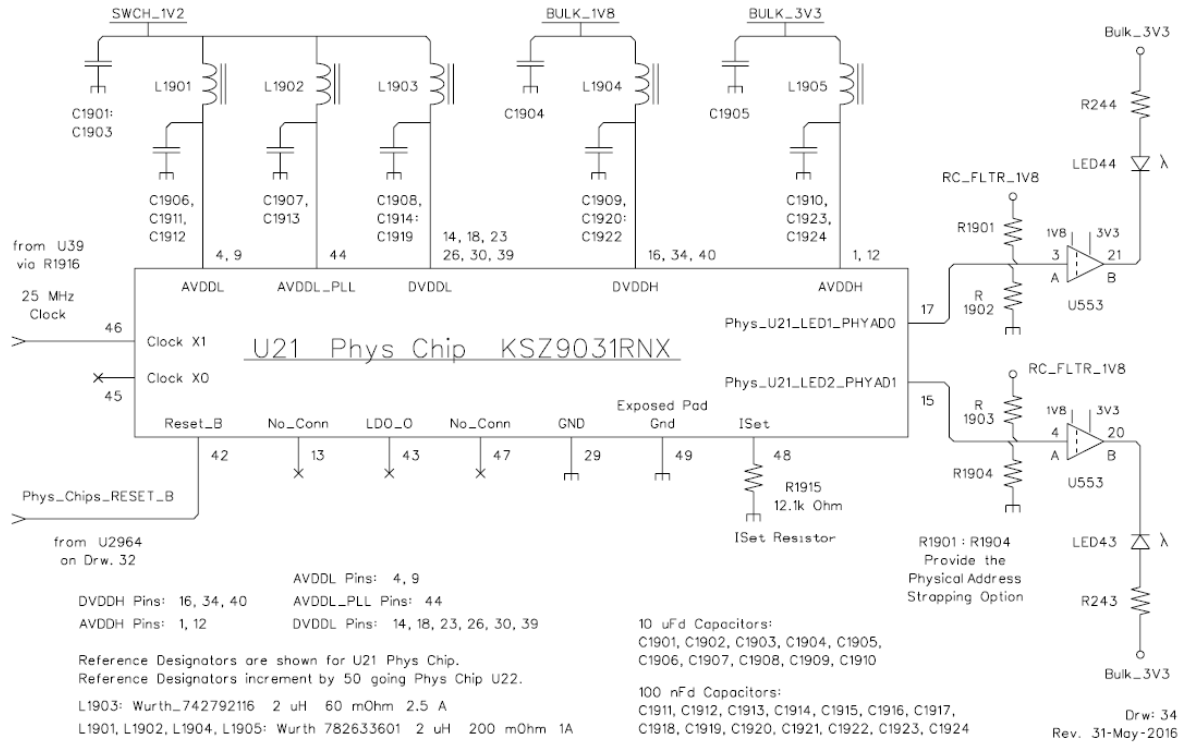
FPGA and PHY chip.....	2
Tri-Mode Ethernet Media Access Controller (TEMAC)	5
MAC design approach	6
MAC IP and example design generation	7
Modifications to the Example Design files.....	8
Testing Tx path.....	9
Testing Rx path.....	13
Remote Loopback in PHY	14
PHY – MAC (RGMII) interface test	16

FPGA and PHY chip

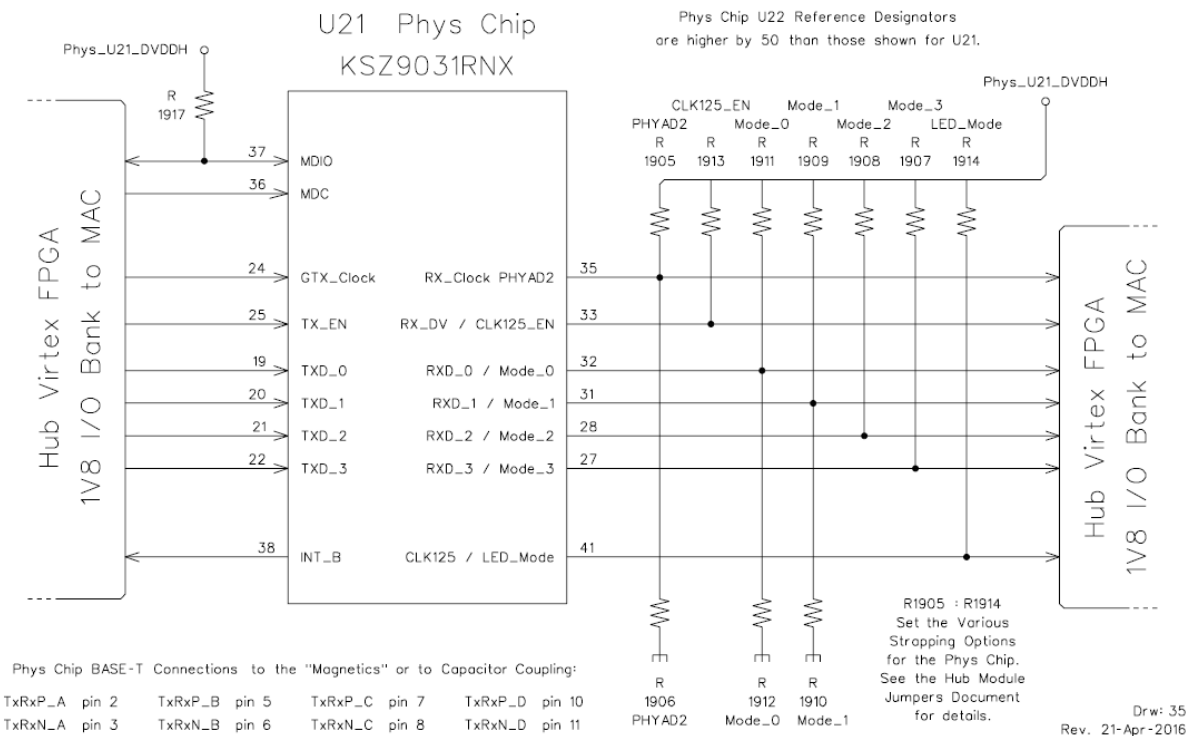
The UltraScale Virtex device on the Hub Module: XCVU125-1FLVC2104I

The PHY: Micrel KSZ9031RX <http://ww1.microchip.com/downloads/en/DeviceDoc/00002117B.pdf>

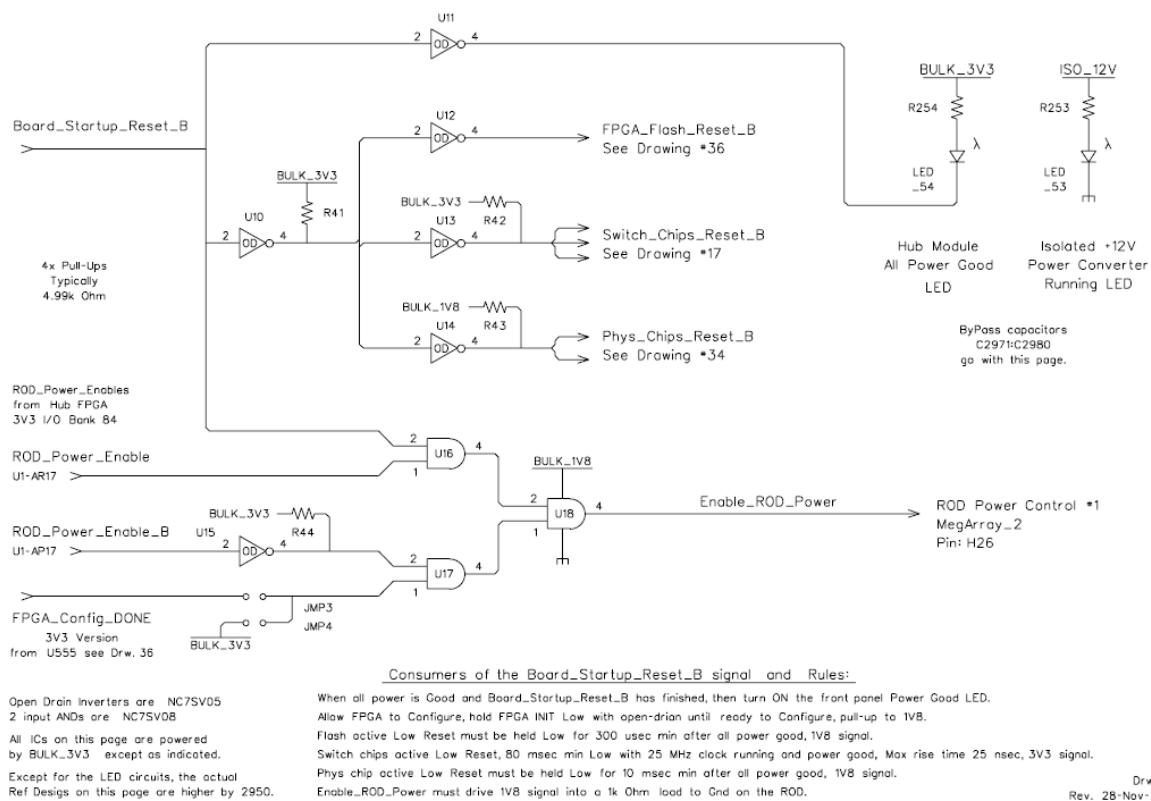
Phys Chip - Power, Clock, Reset, and LED Circuits



Phys Chip - RGMII, MDC/DMIO, and Base-T Circuits



Board Reset Distribution - ROD Power Control



For each of its two Ethernet Phys Chips (PHY) the Hub's FPGA will need to instance a MAC that supports an RGMII connection (along with MDIO/MDC lines) to the PHY. All of these signals to/from the PHY are currently routed through the 1V8 HP Select I/O Bank 68.

After power-up the KSZ9031RNX is configured to RGMII mode if the MODE [3:0] strap-in pins are set to one of the RGMII mode capability options.

There is no reset signal to the KSZ9031RNX from FPGA. An ad hoc manual push button was attached to the PHY chip on the HUB for debugging purposes.

The KSZ9031RNX RGMII port connects to HP I/O pins on the FPGA. The RGMII port consists of 12 signals:

- Transmit Clock to the KSZ9031RNX
- Transmit Control (enable) to the KSZ9031RNX
- Transmit Data 0:3 to the KSZ9031RNX
- Receive Clock from the KSZ9031RNX
- Receive Control (enable) from the KSZ9031RNX
- Receive Data 0:3 from the KSZ9031RNX

The KSZ9031RNX includes a MII Management port. This type of port is also called MDIO Management Data Input/Output. This port allows higher-level devices to monitor and control the KSZ9031RNX. This port allows direct access to the IEEE defined MIIM registers, and the vendor specific registers. This port also allows indirect access to the MMD address space and registers. This port consists of signals: MDC - the clock and MDIO - the data line.

The Hub Module has two KSZ9031RNX PHY chips. There are 14 jumpers associated with each of these PHY chips. These jumpers are resistors that bias a pin in one direction or the other and this value is read when the PHY chip first powers up or is reset.

The KSZ9031RNX has 9 pins (called "Strapping Options") that are read in this way at power up. Because of space limitations and because there is an obvious why that the Hub Module wants some of these Strapping Options set, 4 of them have only one jumper to pull that pin in the direction that is obviously needed for rational operation of the Hub Module.

The PHYADx jumpers set the address of the Management Interface Port on the KSZ9031RNX. The Management Port PHYAD bits 3 and 4 are internally always set to 0,0. Bits 2, 1 and 0 set to Low. Therefore, the PHYADx set to 0.

The Hub Module provides easy control of only the Mode_0 and Mode_1 lines. This provides the following 4 options for the Phys chip (Mode bits listed Mode_3, ..., Mode_0).

- 1100 RGMII 1000 Base-T full duplex only
- 1101 RGMII 1000 Base-T full or half duplex
- 1110 RGMII 10/100/100 all but 1000 half duplex
- 1111 RGMII 10/100/1000 full or half duplex

Mode: SET MODE {3..0} = 1100 - RGMII 1000 Base-T full duplex only

Traces length (in mm) between the FPGA RGMII Rx pins and PHY chip pins:

PHYS_U22_RX_CLK__PHYAD2	65.80
PHYS_U22_RX_DV__CLK125_EN	66.60
PHYS_U22_RXD0__MODE0	65.00
PHYS_U22_RXD1__MODE1	64.77
PHYS_U22_RXD2__MODE2	66.04
PHYS_U22_RXD3__MODE3	68.23

Tri-Mode Ethernet Media Access Controller (TEMAC)

The Xilinx Tri-Mode Ethernet MAC core is a parameterizable core:

<http://www.xilinx.com/products/intellectual-property/temac.html>

Tri-Mode Ethernet MAC v9.0, LogiCORE IP Product Guide, Vivado Design Suite, PG051 April 6, 2016:

https://www.xilinx.com/support/documentation/ip_documentation/tri_mode_ethernet_mac/v9_0/pg051-tri-mode-eth-mac.pdf

In 1000 Mbps mode, the TEMAC core can also connect with industry standard PHY devices.

Optional MDIO interface to managed objects in PHY layers (MII Management)

p.69: Designing with the Core: General Design Guidelines: Design Steps

Generate the core using the Vivado® Design Suite. The core is delivered through the Vivado Design Suite with an HDL example design built around the core, allowing the functionality of the core to be demonstrated using either a simulation package or in hardware, if placed on a suitable board.

p.214: Example Design

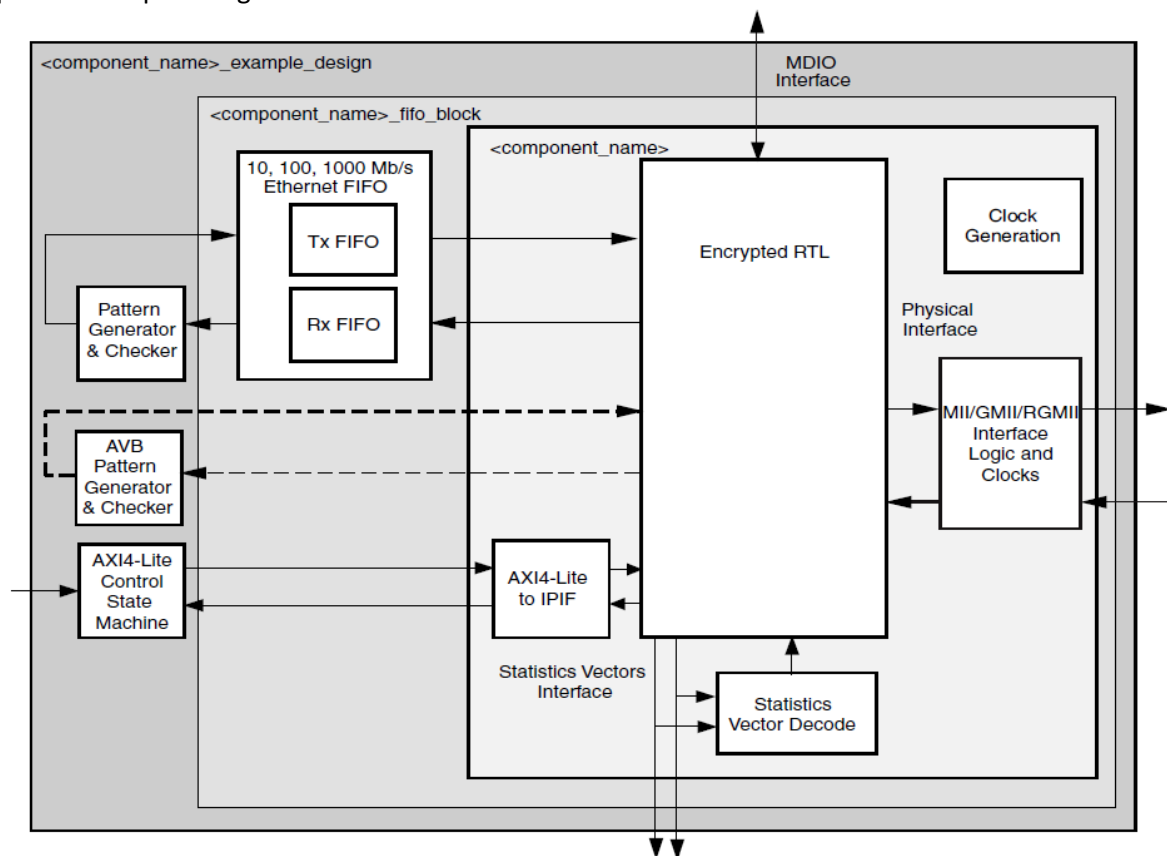


Figure 5-1: HDL Example Design

MAC design approach

The following design approach is based on the suggestion by Ed Flaherty (University of Cambridge):

Step 1: Generate Xilinx MAC Example Design (UltraScale RGMII).

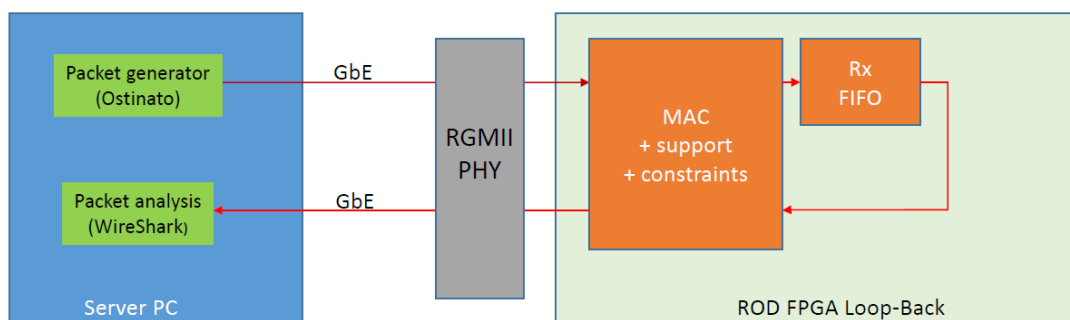
Step 2: Modify the Example Design to HUB board hardware.

Step 3: Generate Tx packets with built-in Simple Frame Generator and capture them in Wireshark.

Step 4: Packet generation logic replaced with simple read fifo (Rx looped back to Tx).

IPBus porting: 3 Steps

Step 2: Stripped down RGMII Example Design



Stripped-down example design

- Packet generation logic replaced with simple read fifo (looped back)
- All of the example design MAC+Support logic and constraints retained
- Verification: Packets sent from server were returned via the loopback proving Rx and Tx paths

Ed Flaherty 7-June-2016

Step 5: Packets sent from server (Ostinato) returned via the loopback proving Rx and Tx paths.

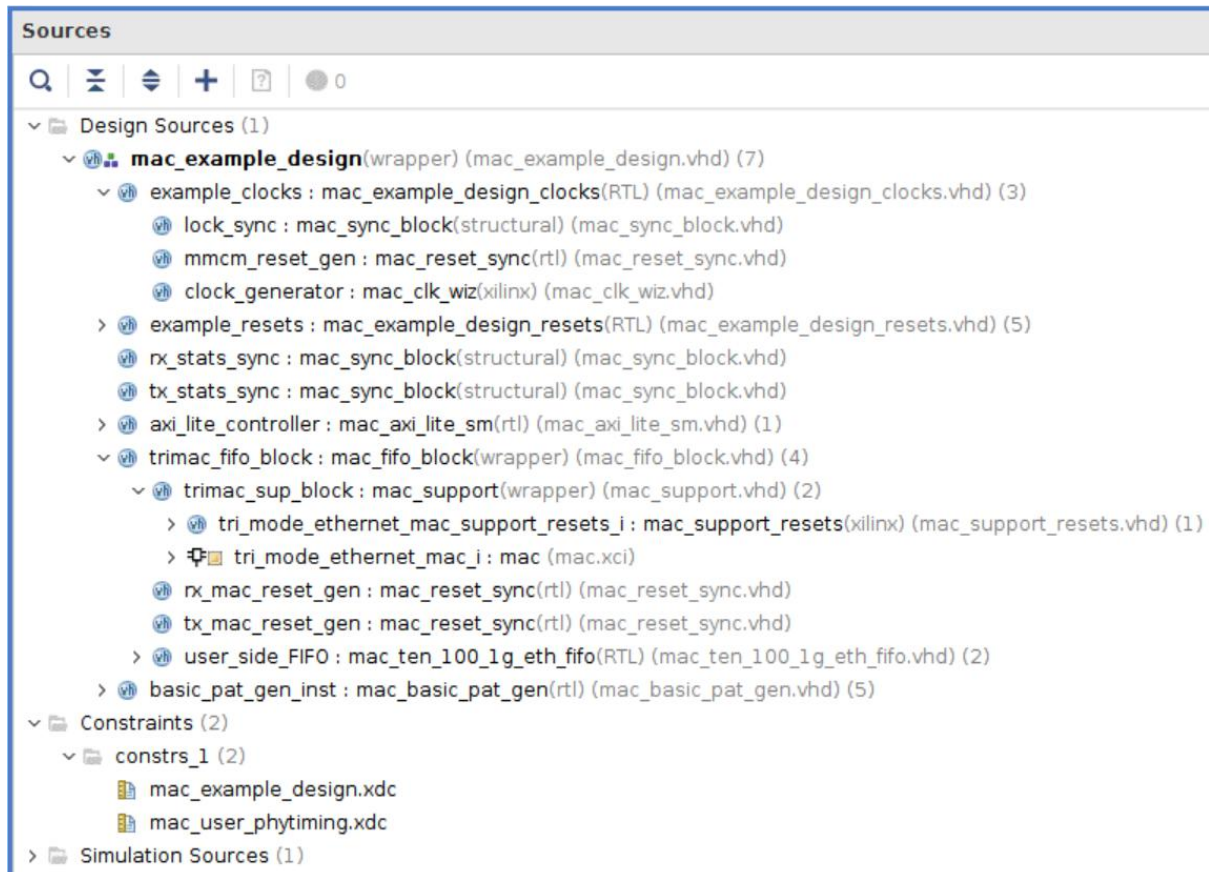
Step 6: Use Wireshark to check returned packets

MAC IP and example design generation

Use Vivado_2017.1, open project, IP catalogue, generate TEMAC IP (AXI4-Lite, 100MHz, MDOI and no Frame Filter) and open IP Example Design.

The Example Design is found on hubdev PC: /home/hubuser/Xilinx/Design/IPB/mac_ex_ref

This is kept as a reference; one may open it and see the TEMAC IP parameters and all unmodified design sources. Here is a structure of the project:



Modifications to the Example Design files

mac_example_design.vhd

- add Safe Configuration ports, set initial value for the IN ports, buffers for clocks
- change clock from 200MHz differential clock to 125MHz single ended clock
- remove unused ports, set controls in the design
- set in component mac_basic_pat_gen MAX_SIZE = MIN_SIZE = packet size = X"040" - 64 bytes
- install VIO to control enable_pat_gen => gen_tx_data

mac_example_design_clocks.vhd

- IBUFG; change clock from 200MHz differential clock to 125MHz single ended clock

mac_clk_wiz.vhd

- change clock from 200MHz differential clock to 125MHz single ended clock
- Tried to modify mac_clk_wiz.vhd to get ref clock 300.0 MHz instead of 333.333 MHz (now it is commented, so clock is still 333.333 MHz)

mac_axi_lite_sm.vhd

- set PHY_ADDR to zero instead of PHYAD 7 (as on the HUB board)
- modify state machine: implement remote and local loopback in PHY

mac_support.vhd

Generate and install 32-bit ILA in trimac_fifo_block/trimac_sup_block(mac_support.vhd)

```
-- Receiver Interface
probe0(7 downto 0) => rx_axis_mac_tdata_int,
probe0(8)          => rx_axis_mac_tvalid_int,
probe0(9)          => rx_axis_mac_tlast_int,
probe0(10)         => rx_axis_mac_tuser_int,
-- Transmitter Interface
probe0(18 downto 11) => tx_axis_mac_tdata,
probe0(19)          => tx_axis_mac_tvalid,
probe0(20)          => tx_axis_mac_tlast,
probe0(21)          => tx_axis_mac_tuser(0),
probe0(22)          => tx_axis_mac_tready_int,
```

mac_support_resets.vhd

Reset circuitry for the IDELAYCTRL reset.

The IDELAYCTRL must experience a pulse, which is at least 50 ns in duration.

This is ten clock cycles of the 200MHz ref clk.

For 333MHz ref clock should be 17 clock cycles ?

- increases the IDELAYCTRL reset - 20 clock cycles of 333MHz ref clock (~60 ns)

mac_example_design.xdc

- add ports for the HUB safe configuration
- change clock from 200MHz differential clock to 125MHz single ended clock

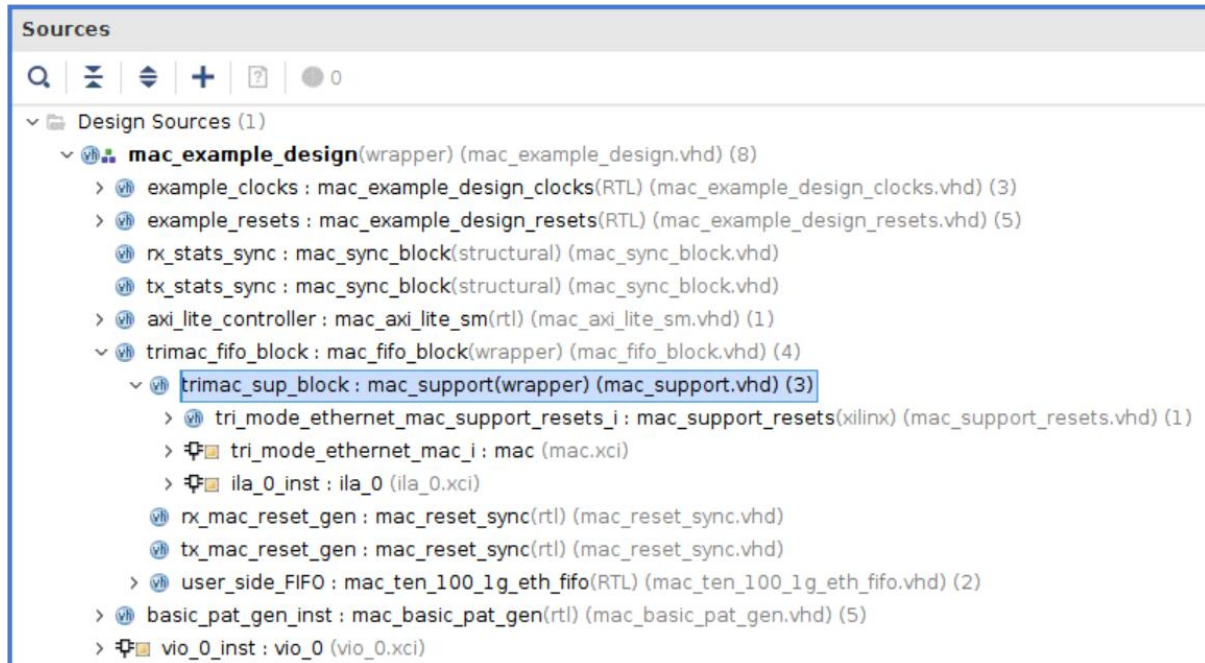
mac_user_phytiming.xdc

- tried to set different delay and adjust for PCB layout

Testing Tx path

Generate Tx packets with built-in Simple Frame Generator (`mac_basic_pat_gen.vhd`) and capture them in Wireshark - FPGA (Frame Generator -> MAC) -> PHY -> Ethernet -> PC (Wireshark).

The Tx Design is found on hubdev PC: `/home/hubuser/Xilinx/Design/IPB/mac_ex_tx`



In `mac_example_design.vhd`:

- Generate and install 4-bit VIO:
`clk => gtx_clk_bufg,`
`probe_out1(0) => gen_tx_data, --> to control Tx on/off`

- set in component `mac_basic_pat_gen` `MAX_SIZE = MIN_SIZE`
component `mac_basic_pat_gen`
generic (
 `DEST_ADDR` : `bit_vector(47 downto 0) := X"da0102030405";`
 `SRC_ADDR` : `bit_vector(47 downto 0) := X"5a0102030405";`
 `--MAX_SIZE` : `unsigned(11 downto 0) := X"1f4";`
 `MAX_SIZE` : `unsigned(11 downto 0) := X"040";`
 `MIN_SIZE` : `unsigned(11 downto 0) := X"040";`

In `mac_support.vhd`

- Generate and install 32-bit ILA in `trimac_fifo_block/trimac_sup_block` (`mac_support.vhd`)
`clk => gtx_clk,`
-- Transmitter Interface
`probe0(18 downto 11) => tx_axis_mac_tdata,`
`probe0(19) => tx_axis_mac_tvalid,`
`probe0(20) => tx_axis_mac_tlast,`
`probe0(21) => tx_axis_mac_tuser(0),`
`probe0(22) => tx_axis_mac_tready_int,`

Generate bit stream, configure FPGA in Hardware Manager and look into ILA in mac_support.vhd:

- set trigger on rising edge of tx_axis_mac_tvalid

Capture Setup - hw_ila_1 Trigger Setup - hw_ila_1 x hw_vio_1

Q | + | - | ↻

Name	Operator	Radix	Value	Port
tx_axis_mac_tvalid	==	...	R	probe0[19]

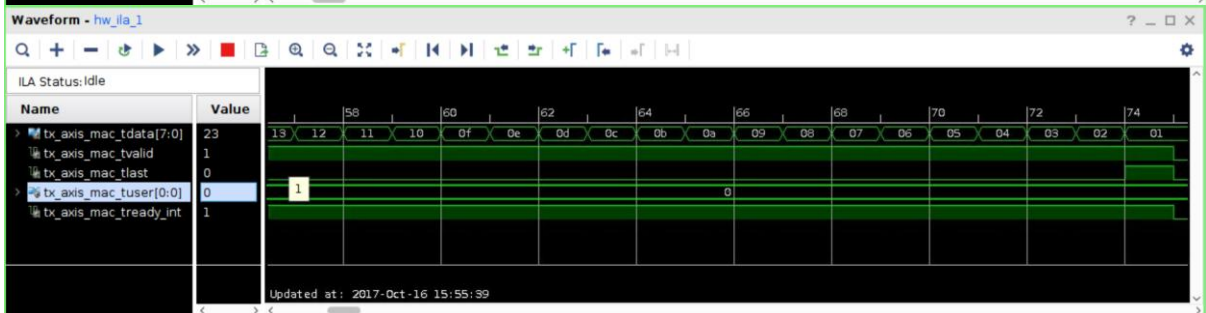
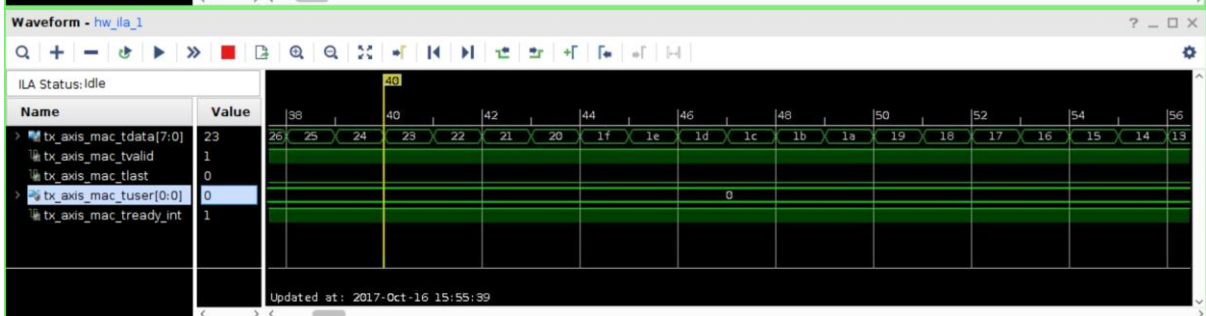
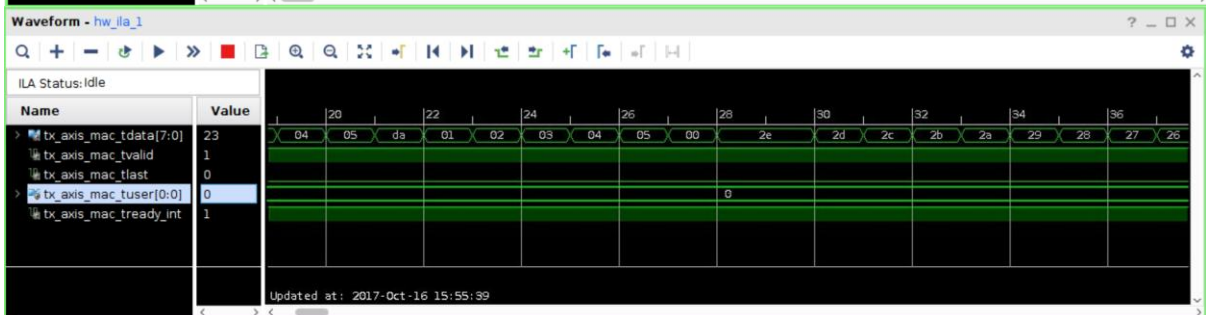
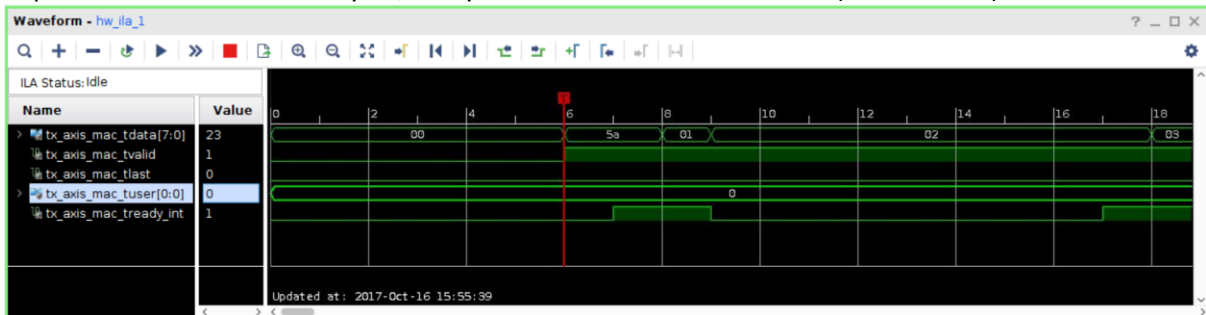
- control enable_pat_gen => gen_tx_data via VIO

Capture Setup - hw_ila_1 Trigger Setup - hw_ila_1 hw_vio_1 x

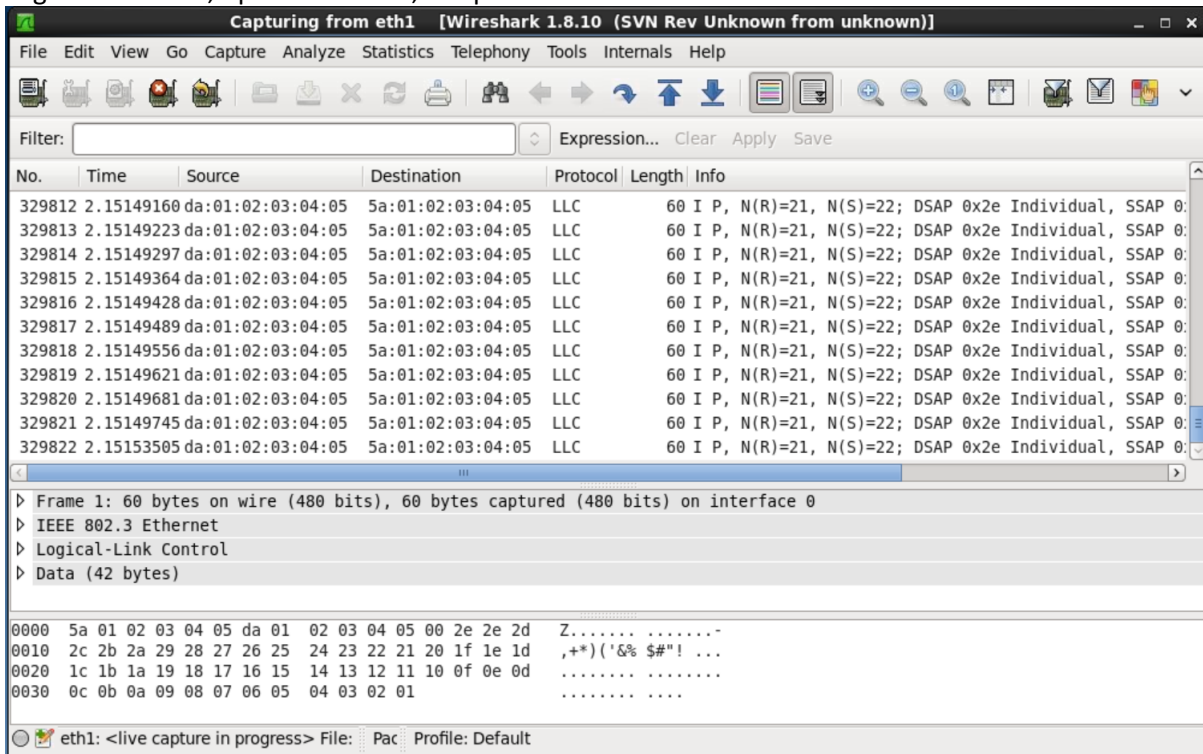
Q | ⚙ | ⚙ | + | -

Name	Value	Activity	Direction	VIO
enable_pat_gen	[B] 0		Output	hw_vio_1

- packet size = X"040" - 64 bytes, 46 bytes of data from 2e to 01 - 64-(6+6+2+4=18)=46

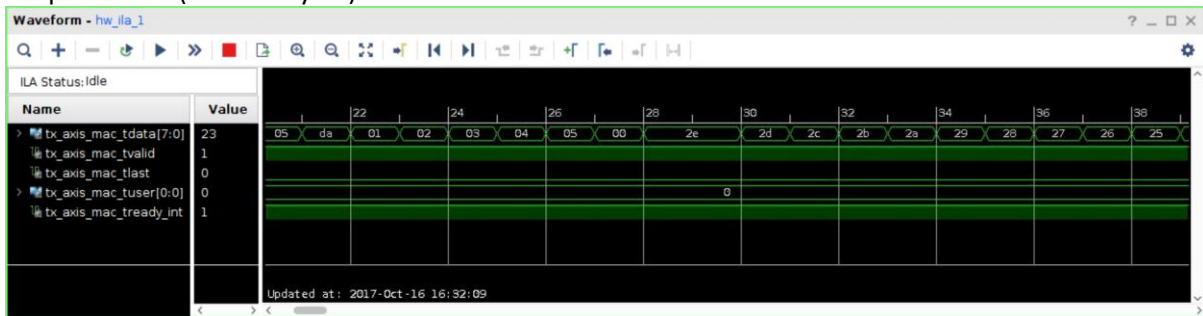


Login into hubttc, open Wireshark, see packets:

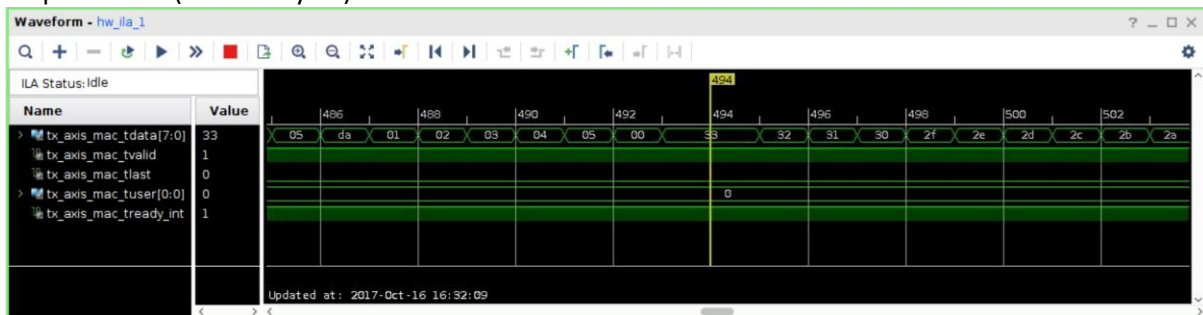


- set in component mac_basic_pat_gen MAX_SIZE = X"045" - to see six different packets:

1st packet: 2e (46 data bytes)



6th packet: 33 (51 data bytes)



- see them in Wireshark:

The image displays two screenshots of the Wireshark 1.8.10 interface, showing network traffic capture on the eth1 interface. The top screenshot shows frame 2652, and the bottom screenshot shows frame 2659.

Top Screenshot (Frame 2652):

No.	Time	Source	Destination	Protocol	Length	Info
2650	0.02276015	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	64	I P, N(R)=23, N(S)=24; DSAP 0x32 Individual, SSAP 0x32
2651	0.02276077	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	65	S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Com
2652	0.02276135	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	60	I P, N(R)=21, N(S)=22; DSAP 0x2e Individual, SSAP 0x2e
2653	0.02276194	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	61	S, func=SREJ, N(R)=22; DSAP 0x2e Group, SSAP 0x2e C
2654	0.02276252	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	62	I P, N(R)=22, N(S)=23; DSAP 0x30 Individual, SSAP 0x30
2655	0.02276308	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	63	U, func=SABM; DSAP 0x30 Group, SSAP 0x30 Command
2656	0.02276372	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	64	I P, N(R)=23, N(S)=24; DSAP 0x32 Individual, SSAP 0x32
2657	0.02276432	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	65	S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Com
2658	0.02280501	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	64	I P, N(R)=23, N(S)=24; DSAP 0x32 Individual, SSAP 0x32
2659	0.02280580	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	65	S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Com
2660	0.02280631	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	60	I P, N(R)=21, N(S)=22; DSAP 0x2e Individual, SSAP 0x2e

Frame 2652: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0

- IEEE 802.3 Ethernet
- Logical-Link Control
- Data (42 bytes)

```

0000  5a 01 02 03 04 05 da 01 02 03 04 05 00 2e 2e 2d  Z.....
0010  2c 2b 2a 29 28 27 26 25 24 23 22 21 20 1f 1e 1d  ,+*)('&% $#"! ...
0020  1c 1b 1a 19 18 17 16 15 14 13 12 11 10 0f 0e 0d  .....
0030  0c 0b 0a 09 08 07 06 05 04 03 02 01  .....
  
```

Bottom Screenshot (Frame 2659):

No.	Time	Source	Destination	Protocol	Length	Info
2650	0.02276015	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	64	I P, N(R)=23, N(S)=24; DSAP 0x32 Individual, SSAP 0x32
2651	0.02276077	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	65	S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Com
2652	0.02276135	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	60	I P, N(R)=21, N(S)=22; DSAP 0x2e Individual, SSAP 0x2e
2653	0.02276194	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	61	S, func=SREJ, N(R)=22; DSAP 0x2e Group, SSAP 0x2e C
2654	0.02276252	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	62	I P, N(R)=22, N(S)=23; DSAP 0x30 Individual, SSAP 0x30
2655	0.02276308	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	63	U, func=SABM; DSAP 0x30 Group, SSAP 0x30 Command
2656	0.02276372	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	64	I P, N(R)=23, N(S)=24; DSAP 0x32 Individual, SSAP 0x32
2657	0.02276432	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	65	S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Com
2658	0.02280501	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	64	I P, N(R)=23, N(S)=24; DSAP 0x32 Individual, SSAP 0x32
2659	0.02280580	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	65	S, func=RR, N(R)=24; DSAP 0x32 Group, SSAP 0x32 Com
2660	0.02280631	da:01:02:03:04:05	5a:01:02:03:04:05	LLC	60	I P, N(R)=21, N(S)=22; DSAP 0x2e Individual, SSAP 0x2e

Frame 2659: 65 bytes on wire (520 bits), 65 bytes captured (520 bits) on interface 0

- IEEE 802.3 Ethernet
- Logical-Link Control
- Data (47 bytes)

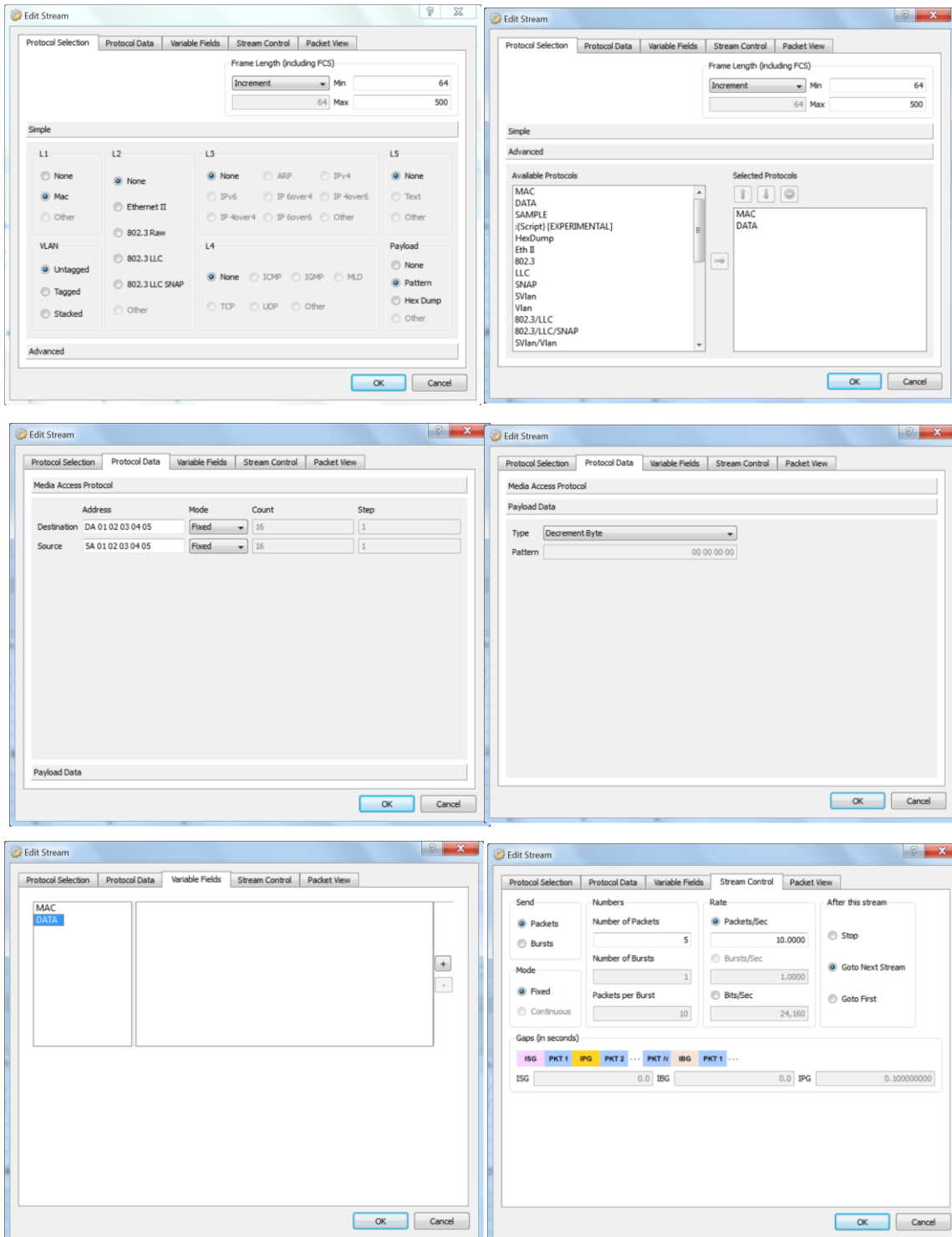
```

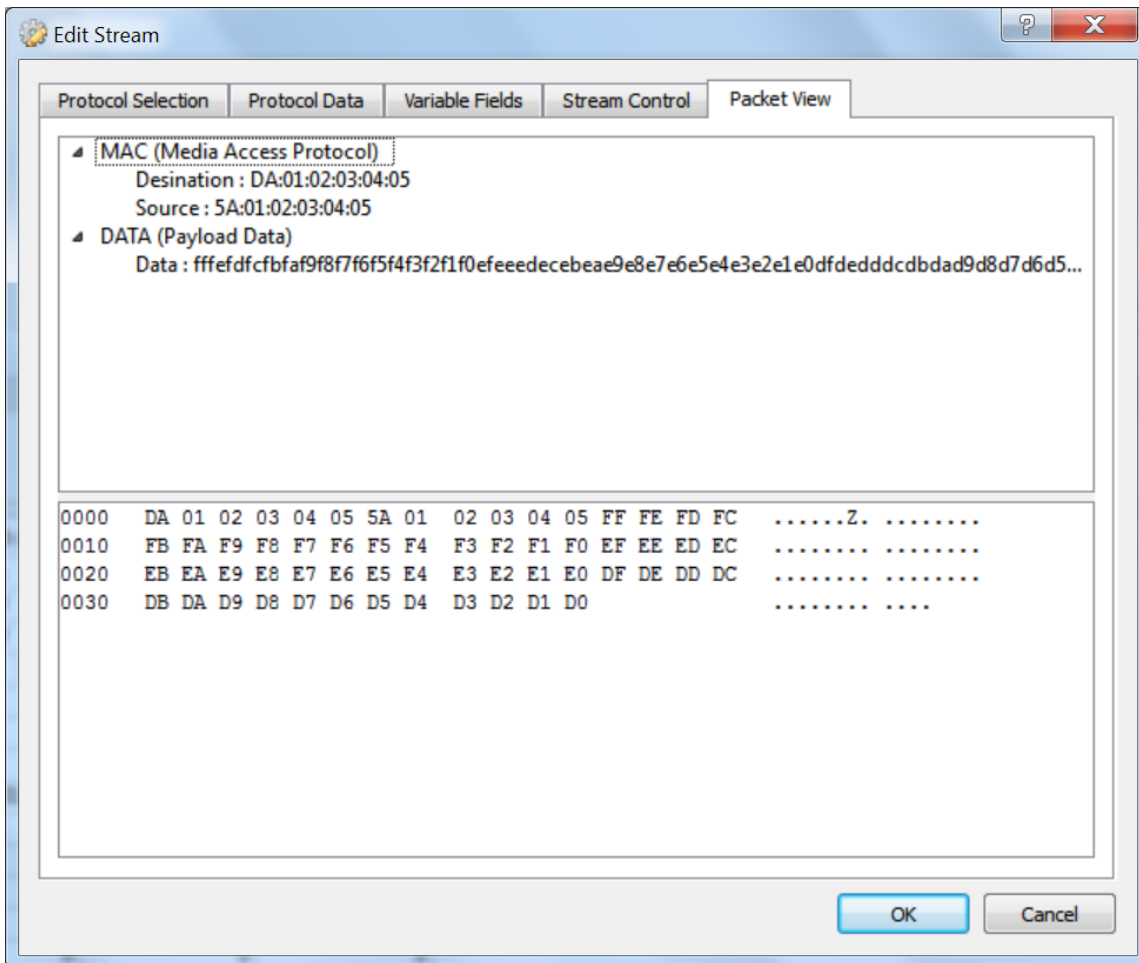
0000  5a 01 02 03 04 05 da 01 02 03 04 05 00 33 33 32  Z.....332
0010  31 30 2f 2e 2d 2c 2b 2a 29 28 27 26 25 24 23 22  10/.-,+)('&% $#
0020  21 20 1f 1e 1d 1c 1b 1a 19 18 17 16 15 14 13 12  ! .....
0030  11 10 0f 0e 0d 0c 0b 0a 09 08 07 06 05 04 03 02  .....
0040  01  .....
  
```

Conclusion: data can be correctly sent out (Tx) via MAC and PHY

Testing Rx path

To test Rx path, the Ostinato program on PC is used to generate packets. The Ostinato stream (vc709.ostm) is on the hubttc PC in the directory: hubuser/vc709





Remote Loopback in PHY

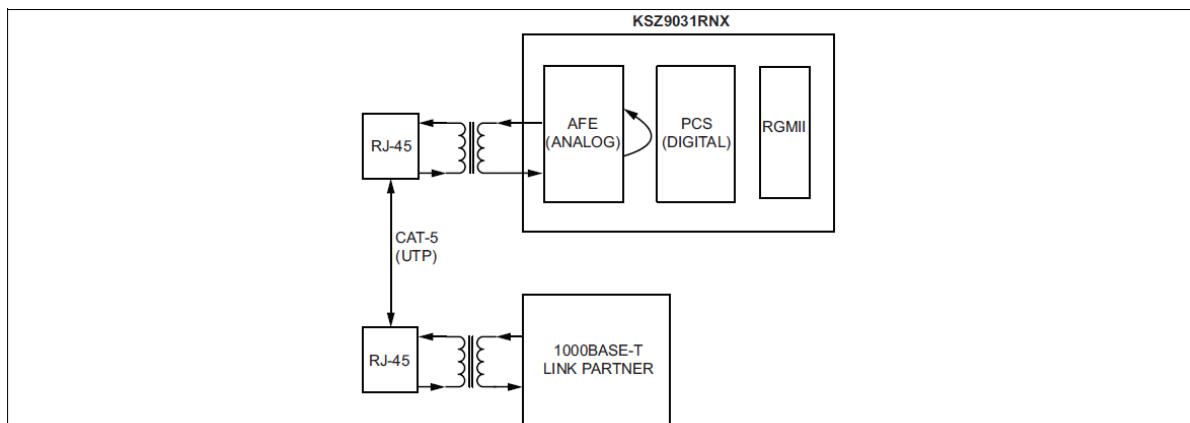
Micrel KSZ9031RNX <http://ww1.microchip.com/downloads/en/DeviceDoc/00002117B.pdf> p. 25:

3.13.2 REMOTE (ANALOG) LOOPBACK

This loopback mode checks the line (differential pairs, transformer, RJ-45 connector, Ethernet cable) transmit and receive data paths between KSZ9031RNX and its link partner, and is supported for 1000BASE-T full-duplex mode only. The loopback data path is shown in [Figure 3-6](#).

1. The Gigabit PHY link partner transmits frames to KSZ9031RNX.
2. Frames are wrapped around inside KSZ9031RNX.
3. KSZ9031RNX transmits frames back to the Gigabit PHY link partner.

FIGURE 3-6: REMOTE (ANALOG) LOOPBACK



The following programming steps and register settings are used for remote loopback mode.

1. Set Register 0h,
 - Bits [6, 13] = 10 // Select 1000 Mbps speed
 - Bit [12] = 0 // Disable auto-negotiation
 - Bit [8] = 1 // Select full-duplex mode

Or just auto-negotiate and link up at 1000BASE-T full-duplex mode with the link partner.

2. Set Register 11h,
 - Bit [8] = 1 // Enable remote loopback mode

The Remote Loopback Design on hubdev PC: /home/hubuser/Xilinx/Design/IPB/mac_ex_rem_loop

In **mac_example_design.vhd**:

```
axi_lite_controller : mac_axi_lite_sm
- phy_loopback => '1', -- in YE enable loopback
basic_pat_gen_inst : mac_basic_pat_gen
- enable_pat_gen => '0', -- YE '0' - no Tx (disabled)
```

In **mac_axi_lite_sm.vhd**

- set PHY_ADDR to zero instead of PHYAD 7 (as on the HUB board)
- modify state machine: implement remote and local loopback in PHY (new states and regs)
 - add new reg constant PHY_RMT_LPB in mac_axi_lite_sm.vhd
 - modify states: MDIO_RESTART and MDIO_LOOPBACK
 - enable remote loopback code part and disable local loopback code part

Test remote loopback - shall see the Ostinato frames in Wireshark: works - sending 5 packet from Ostinato, Wireshark see 10 packets (5 packet5 is sent by Ostinato and the other 5 packets is the ones send back by the PHY in the remote loopback mode).

Capturing from eth1 [Wireshark 1.8.10 (SVN Rev Unknown from unknown)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	5a:01:02:03:04:05	da:01:02:03:04:05	0xffff	60	Ethernet II
2	0.00013624	5a:01:02:03:04:05	da:01:02:03:04:05	0xffff	60	Ethernet II
3	0.09999628	5a:01:02:03:04:05	da:01:02:03:04:05	0xffff	61	Ethernet II
4	0.10012659	5a:01:02:03:04:05	da:01:02:03:04:05	0xffff	61	Ethernet II
5	0.19999945	5a:01:02:03:04:05	da:01:02:03:04:05	0xffff	62	Ethernet II
6	0.20012952	5a:01:02:03:04:05	da:01:02:03:04:05	0xffff	62	Ethernet II
7	0.29999785	5a:01:02:03:04:05	da:01:02:03:04:05	0xffff	63	Ethernet II
8	0.30007845	5a:01:02:03:04:05	da:01:02:03:04:05	0xffff	63	Ethernet II
9	0.39999535	5a:01:02:03:04:05	da:01:02:03:04:05	0xffff	64	Ethernet II
10	0.40007608	5a:01:02:03:04:05	da:01:02:03:04:05	0xffff	64	Ethernet II

Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0

Ethernet II, Src: 5a:01:02:03:04:05 (5a:01:02:03:04:05), Dst: da:01:02:03:04:05 (da:01:02:03:04:05)

Data (46 bytes)

```
0000 da 01 02 03 04 05 5a 01 02 03 04 05 ff fe fd fc .....Z. ....
0010 fb fa f9 f8 f7 f6 f5 f4 f3 f2 f1 f0 ef ee ed ec .....
0020 eb ea e9 e8 e7 e6 e5 e4 e3 e2 e1 e0 df de dd dc .....
0030 db da d9 d8 d7 d6 d5 d4 d3 d2 d1 d0 .....
eth1: <live capture in progress> Fil... P... Profile: Default
```

Conclusion: data correctly received by PHY from Ostinato and send beck by PHY (in remote loopback mode) to Wireshark.

PHY – MAC (RGMII) interface test in local loopback

Micrel KSZ9031RNX <http://ww1.microchip.com/downloads/en/DeviceDoc/00002117B.pdf> p. 25:

3.13.1 LOCAL (DIGITAL) LOOPBACK

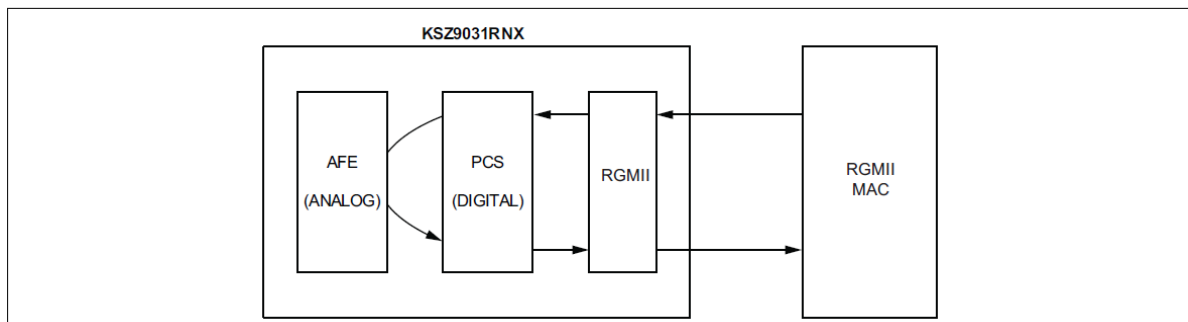
This loopback mode checks the RGMII transmit and receive data paths between KSZ9031RNX and external MAC, and

is supported for all three speeds (10/100/1000 Mbps) at full-duplex.

The loopback data path is shown in [Figure 3-5](#).

1. RGMII MAC transmits frames to KSZ9031RNX.
2. Frames are wrapped around inside KSZ9031RNX.
3. KSZ9031RNX transmits frames back to RGMII MAC.

FIGURE 3-5: LOCAL (DIGITAL) LOOPBACK



The following programming steps and register settings are used for local loopback mode.

For 1000 Mbps loopback,

1. Set Register 0h,
 - Bit [14] = 1 // Enable local loopback mode
 - Bits [6, 13] = 10 // Select 1000 Mbps speed
 - Bit [12] = 0 // Disable auto-negotiation
 - Bit [8] = 1 // Select full-duplex mode
2. Set Register 9h,
 - Bit [12] = 1 // Enable master-slave manual configuration
 - Bit [11] = 0 // Select slave configuration (required for loopback mode)