OpenCores.Org

10_100_1000 Mbps Tri-mode Ethernet MAC Specification

Author: Jon Gao

gaojon@yahoo.com

Rev. 0.1

January 25, 2006

This page has been intentionally left blank.

Revision History

Rev.	Date	Author	Description
0.1	11/28/05	Jon Gao	First Draft

Contents

1 Introduction

10_100_1000 Mbps tri-mode ethernet MAC implements a MAC controller conforming to IEEE 802.3 specification. It is designed to use less than 2000 LCs/LEs to implement full function. It will use inferred RAMs and PADs to reduce technology dependance.To increase the flexibility,three optional modules can be added to or removed from the project.

A GUI configuration interface,created by tcl/tk script language,is convenient for configuring optional modules,FiFo depth and verifcation parameters. Furthermore,a verifcation system was designed with tcl/tk user interface,by which the stimulus can be generated automatically and the output packets can be verified with CRC-32 checksum.

functional description:

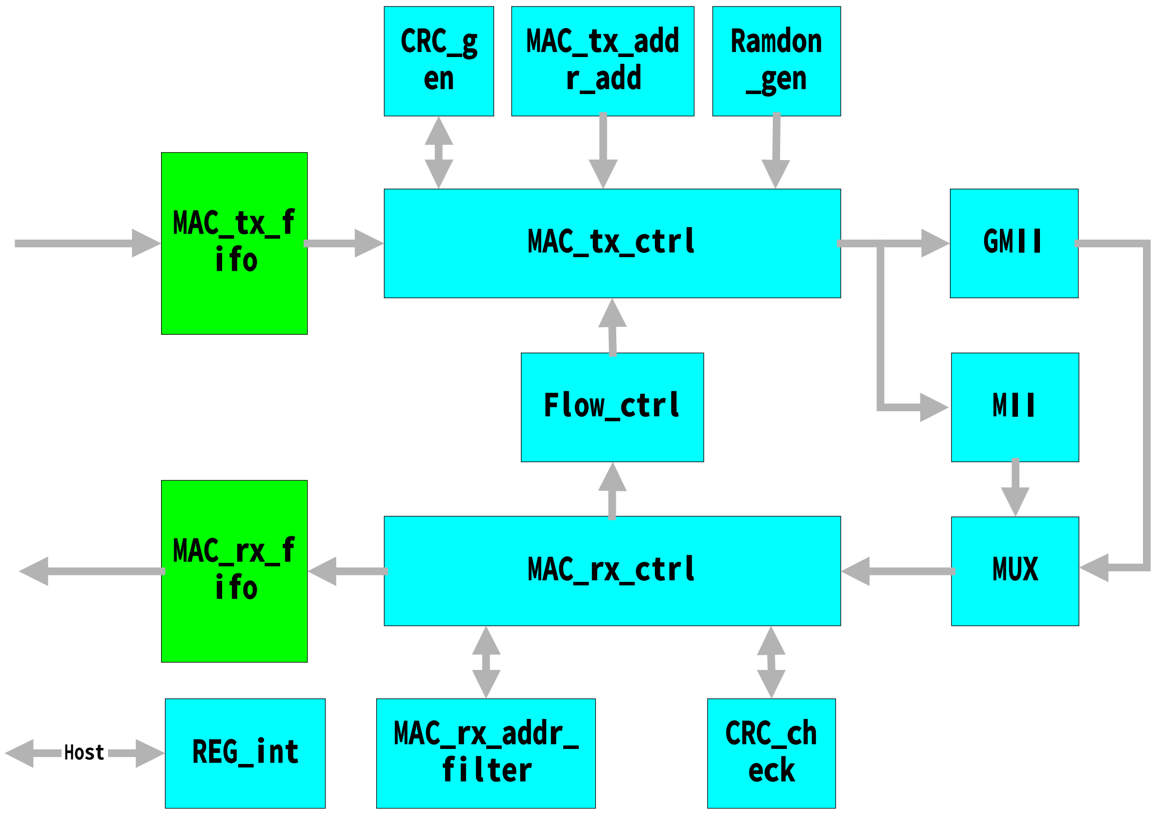
- Implements the full 802.3 specification.
- half-duplex support for 10 100 Mbps mode
- FIFO interface to user application
- support pause frame generation and termination
- transmitting frames source MAC address insertion(optional)
- receiving frames destination MAC address filter(optional)
- receiving broadcast frames throughout constraint(optional)
- support Jumbo frame 9.6K
- RMON MIB statistic counter

Before you start to use this IP core, you need to make your working enviroment ready. At fist, you need to install WIN2K operation system(recommend) or other stable operation system.In addtion, Cygwin is needed to run some bash scripts.Tcl/tk is also needed for many GUI scripts. Finally, the cadence LDV nc-sim is needed for simualtion. If you are using worksation, you need Tcl/tk of your platform for GUI scripts and LDV nc-sim. I recommend WIN2K + Cygwin0528 + LDV5.0 +Tcl/tk.

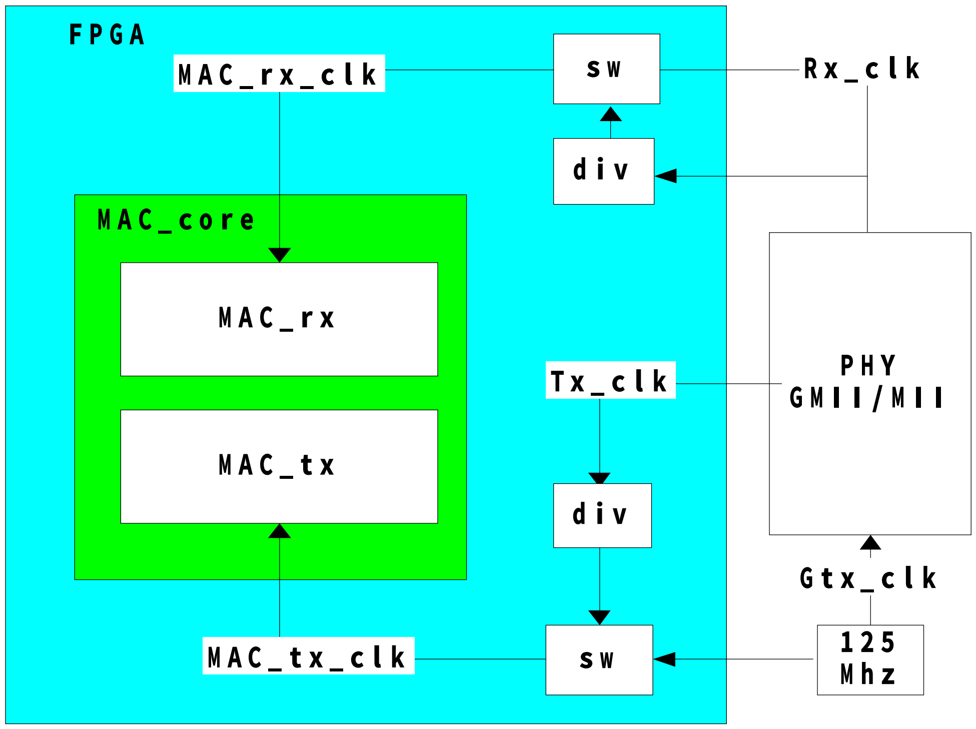
2

Architecture

block diagram



clock distribution



3

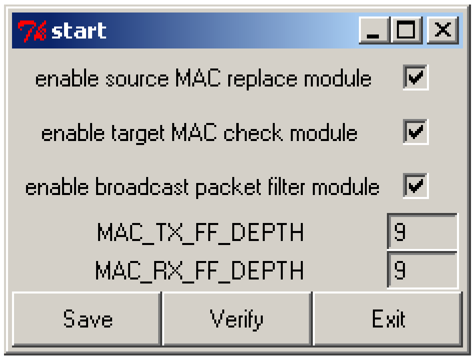
Operation

Config optaional modules

There are three optional modules can be removed frome IP core to reduce area. You can start a GUI configuration tools to customize this IP core. Type the following command in root directory to run a tcl/tk script.

#vish start.tcl

if the tcl/tk was installed correctly, a GUI windows will appear on your screen. It's looks like that:


start

enable source MAC replace module ☑

enable target MAC check module ☑

enable broadcast packet filter module ☑

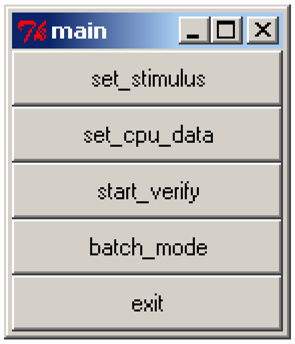
MAC_TX_FF_DEPTH 9

MAC_RX_FF_DEPTH 9

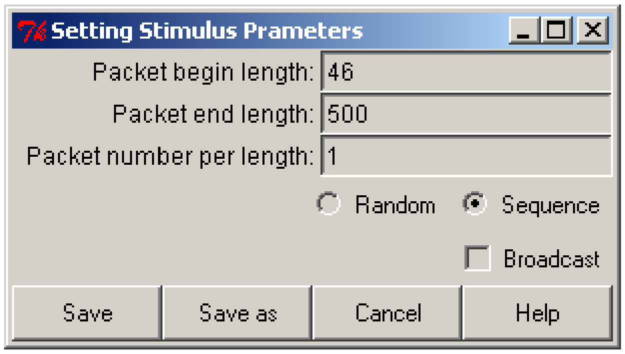
To click the checkbutton will enable the corresponding module of IP core.By the way, the Fifo depth can be set in this window.The default setting of Fifo depth is 9,which means that the fifo can contain 512 words. Because of the fifo width of user side is 32bit, the actual capacity of fifo is 512*4=2K bytes. After you changed the setting, don't forget to save the new configuration.

Verification

To click the "verify"button of above window, a new window will appear for verification.



the first button “set_stimulus” is used to config the parameters used for automatically generate stimulus for simulation.


Setting Stimulus Prameters

Packet begin length: 46

Packet end length: 500

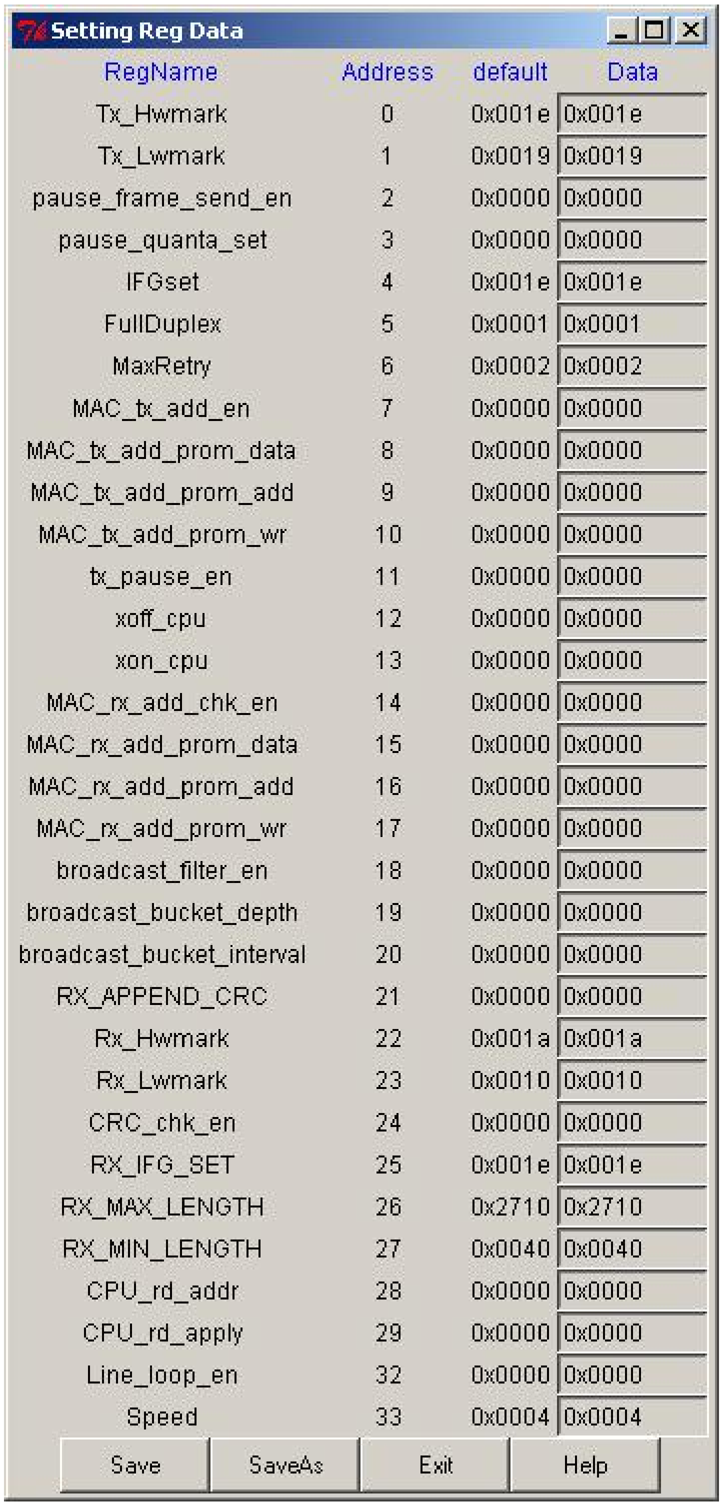
Packet number per length: 1

Random [✓ Sequence]

Broadcast

If you select “Random” mode , the generated packet length will be random in the range of “Packet begin length “ and “Packet begin length”. The generated packet number will be equal to “Total Gen Packet number”.If you want to generate broadcast packets , just click the “broadcast” slectbutton. The “save” button will save current configuration to “config.ini” file.Furthermore, you can use “save as” button to save the configuration as another file which can be used for “batch_mode”

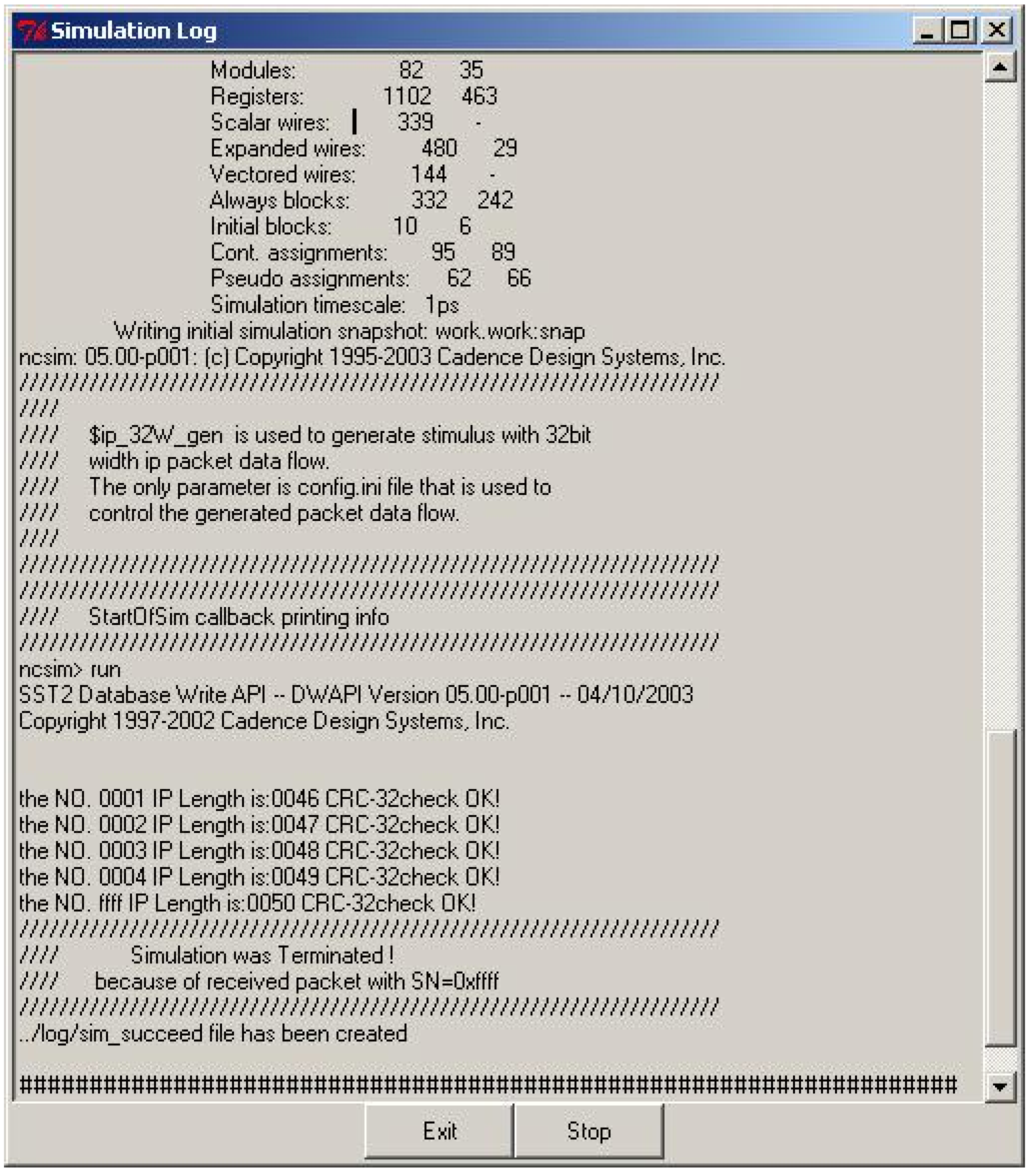
“set_cpu_data” button of main window is used to config internal registers.All the registers will be list in following forms:



RegName	Address	default	Data
Tx_Hwmark	0	0x001e	0x001e
Tx_Lwmark	1	0x0019	0x0019
pause_frame_send_en	2	0x0000	0x0000
pause_quanta_set	3	0x0000	0x0000
IFGset	4	0x001e	0x001e
FullDuplex	5	0x0001	0x0001
MaxRetry	6	0x0002	0x0002
MAC_tx_add_en	7	0x0000	0x0000
MAC_tx_add_prom_data	8	0x0000	0x0000
MAC_tx_add_prom_add	9	0x0000	0x0000
MAC_tx_add_prom_wr	10	0x0000	0x0000
tx_pause_en	11	0x0000	0x0000
xoff_cpu	12	0x0000	0x0000
xon_cpu	13	0x0000	0x0000
MAC_rx_add_chk_en	14	0x0000	0x0000
MAC_rx_add_prom_data	15	0x0000	0x0000
MAC_rx_add_prom_add	16	0x0000	0x0000
MAC_rx_add_prom_wr	17	0x0000	0x0000
broadcast_filter_en	18	0x0000	0x0000
broadcast_bucket_depth	19	0x0000	0x0000
broadcast_bucket_interval	20	0x0000	0x0000
RX_APPEND_CRC	21	0x0000	0x0000
Rx_Hwmark	22	0x001a	0x001a
Rx_Lwmark	23	0x0010	0x0010
CRC_chk_en	24	0x0000	0x0000
RX_IFG_SET	25	0x001e	0x001e
RX_MAX_LENGTH	26	0x2710	0x2710
RX_MIN_LENGTH	27	0x0040	0x0040
CPU_rd_addr	28	0x0000	0x0000
CPU_rd_apply	29	0x0000	0x0000
Line_loop_en	32	0x0000	0x0000
Speed	33	0x0004	0x0004

the "save" button will save current configuration to file "CPU.dat".Also, you can use "save as" button to save register setting to any others files you like. Some complex operation of register such as reading statistic counters need to edit "CPU.dat" file manually.

The “start_verify” of main window will start simulation. The compiling and simulation output will be printed in the following windows:



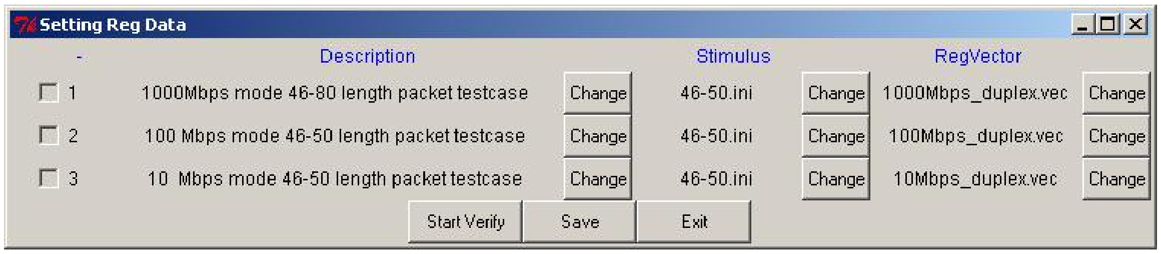
```
                Modules:            82      35
                Registers:        1102     463
                Scalar wires:  |    339      -
                Expanded wires:       480     29
                Vectored wires:      144      -
                Always blocks:       332     242
                Initial blocks:     10       6
                Cont. assignments:     95     89
                Pseudo assignments:     62     66
                Simulation timescale:   1ps
        Writing initial simulation snapshot: work.work:snap
ncsim: 05.00-p001: (c) Copyright 1995-2003 Cadence Design Systems, Inc.
///////////////////////////////////////////////////////////////////////////
////
////    $ip_32W_gen  is used to generate stimulus with 32bit
////    width ip packet data flow.
////    The only parameter is config.ini file that is used to
////    control the generated packet data flow.
////
///////////////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////////////
////    StartOfSim callback printing info
///////////////////////////////////////////////////////////////////////////
ncsim> run
SST2 Database Write API -- DWAPI Version 05.00-p001 -- 04/10/2003
Copyright 1997-2002 Cadence Design Systems, Inc.


the NO. 0001 IP Length is:0046 CRC-32check OK!
the NO. 0002 IP Length is:0047 CRC-32check OK!
the NO. 0003 IP Length is:0048 CRC-32check OK!
the NO. 0004 IP Length is:0049 CRC-32check OK!
the NO. ffff IP Length is:0050 CRC-32check OK!
///////////////////////////////////////////////////////////////////////////
////          Simulation was Terminated !
////     because of received packet with SN=0xffff
///////////////////////////////////////////////////////////////////////////
../log/sim_succeed file has been created

###########################################################################
```

At first , a bash script will be invoked to compile the source file .If no any error occurred , the nc-simulator will be invoked to start simulation. When any error packet received , the simulator will stop and print the data of received error packet.

The “batch_mode” button of main windwos will invoke following windows:



-	Description		Stimulus		RegVector	
☐ 1	1000Mbps mode 46-80 length packet testcase	Change	46-50.ini	Change	1000Mbps_duplex.vec	Change
☐ 2	100 Mbps mode 46-50 length packet testcase	Change	46-50.ini	Change	100Mbps_duplex.vec	Change
☐ 3	10 Mbps mode 46-50 length packet testcase	Change	46-50.ini	Change	10Mbps_duplex.vec	Change

this window will be used to perform verify the IP core with several testcase.In this window, you can change the description , stimulus and regvector of a testcase.

Reg Description

Register list

Reg Name	Operaion	Address	Default Value
Tx_Hwmark	R/w	7'd000	16'h001e
Tx_Lwmark	R/w	7'd001	16'h0019
pause_frame_send_en	R/w	7'd002	16'h0000
pause_quanta_set	R/w	7'd003	16'h0000
IFGset	R/w	7'd004	16'h0012
FullDuplex	R/w	7'd005	16'h0001
MaxRetry	R/w	7'd006	16'h0002
MAC_tx_add_en	R/w	7'd007	16'h0000
MAC_tx_add_prom_data	R/w	7'd008	16'h0000
MAC_tx_add_prom_add	R/w	7'd009	16'h0000
MAC_tx_add_prom_wr	R/w	7'd010	16'h0000
tx_pause_en	R/w	7'd011	16'h0000
xoff_cpu	R/w	7'd012	16'h0000
xon_cpu	R/w	7'd013	16'h0000
MAC_rx_add_chk_en	R/w	7'd014	16'h0000
MAC_rx_add_prom_data	R/w	7'd015	16'h0000
MAC_rx_add_prom_add	R/w	7'd016	16'h0000
MAC_rx_add_prom_wr	R/w	7'd017	16'h0000
broadcast_filter_en	R/w	7'd018	16'h0000
broadcast_bucket_depth	R/w	7'd019	16'h0000
broadcast_bucket_interval	R/w	7'd020	16'h0000
RX_APPEND_CRC	R/w	7'd021	16'h0000
Rx_Hwmark	R/w	7'd022	16'h001a

Rx_Lwmark	R/w	7'd023	16'h0010
CRC_chk_en	R/w	7'd024	16'h0000
RX_IFG_SET	R/w	7'd025	16'h0012
RX_MAX_LENGTH	R/w	7'd026	16'h2710
RX_MIN_LENGTH	R/w	7'd027	16'h0040
CPU_rd_addr	R/w	7'd028	16'h0000
CPU_rd_apply	R/w	7'd029	16'h0000
CPU_rd_grant	R	7'd030	16'h0000
CPU_rd_dout_l	R	7'd031	16'h0000
CPU_rd_dout_h	R	7'd032	16'h0000
Line_loop_en	R/w	7'd033	16'h0000
Speed	R/w	7'd034	16'h0004

Water mark

Tx_Hwmark	R/w	7'd000	16'h001e
Tx_Lwmark	R/w	7'd001	16'h0019

This two registers are used to set transmit Fifo high water mark and low water hark.When the packet data stored in transmit Fifo meet low water,transmit control logic will begin to read packet data from Fifo and send it to Phy through GMII/MII interface. In addition,hight water mark and low water mark setting is correlated with **Tx_mac_wr** signal. When the transmit Fifo meet high water mark , **Tx_mac_wr** will be asserted 0 to tell user application to hold packet transmitting. When the transmit Fifo under low water mark, the **Tx_mac_wr** signal will be disasserted and the user application can proceed to transmit packet data.

Rx_Hwmark	R/w	7'd022	16'h001a
Rx_Lwmark	R/w	7'd023	16'h0010

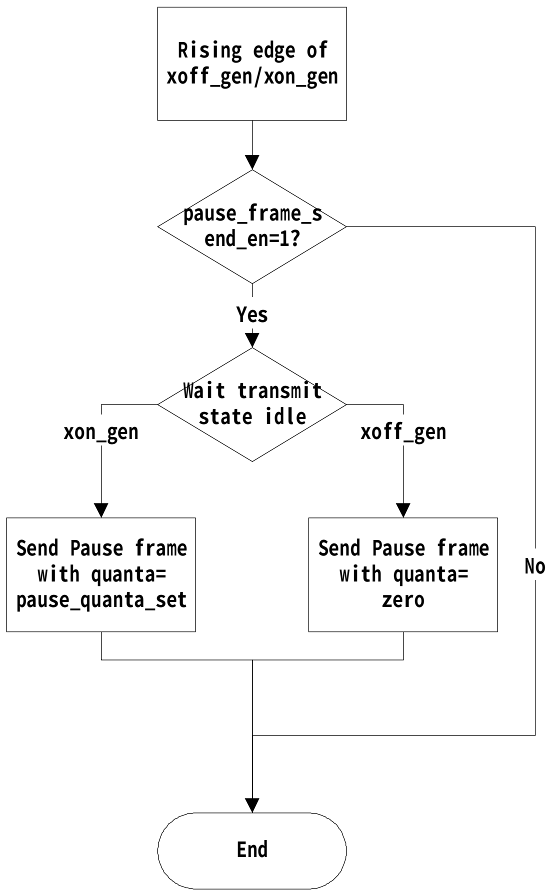
This two registers are used to set receive Fifo water mark. When Fifo received one full packet or stored packet data meet hight water mark,the **Rx_mac_ra** will be asserted.The **Rx_mac_ra** will be disasserted while Fifo below low water mark. If there is one full packet received from PHY, The **Rx_mac_ra** will disasserted when the whole packet is read out from Fifo.

Flow control

pause_frame_send_en	R/w	7'd002	16'h0000

pause_quanta_set	R/w	7'd003	16'h0000
xoff_cpu	R/w	7'd012	16'h0000
xon_cpu	R/w	7'd013	16'h0000

pause_frame_send_en register is used to enable transmit logic to send PAUSE frame. **pause_quanta_set** register is used to setting quanta value in send PAUSE frame. The rising pulse of **xon_cpu** signal is used to start transmit one PAUSE frame with quanta value of **pause_quanta_set** when the transmit in idle state. The rising pulse of **xoff_cpu** signal is used to start transmit one PAUSE frame when the transmit in idle state with quanta zero, asking remote ethernet controller jump out from pause state.


Rising edge of xoff_gen/xon_gen
pause_frame_send_en=1?
Yes
No
Wait transmit state idle
xon_gen
xoff_gen
Send Pause frame with quanta= pause_quanta_set
Send Pause frame with quanta= zero
End

tx_pause_en	R/w	7'd011	16'h0000

When this register is “1”, this core will respond to received pause frame.The transmit state machine will enter PAUSE state according to quanta value in received packet . One quanta time is equal to the time of transmit 512bit data.

IFG

IFGset	R/w	7'd004	16'h0012
RX_IFG_SET	R/w	7'd025	16'h0012

According to IEEE802.3 ,The minimum IfG value is 96bit. If the system clock is 125Mhz with 8bit data width, the interval time between packet is at least 18 clock cycles

RX_IFG_SET is used to set received frame gap. If the gap between two received packets is less than RX_IFG_SET,the second packet will be drop as an invalid frame.

Full duplex

FullDuplex	R/w	7'd005	16'h0001

This config register is only used in 100Mbps and 10Mbps. When FullDuplex register is equal to "1" , the transmit state machine will be FullDuplex mode.Otherwise,the transmit state machine will detect collision ,perform random slot time back off , retransmit collision packet and some other half-duplex operations.

MaxRetry	R/w	7'd006	16'h0002

When collision occurred in half duplex mode, the transmit state machine will try to transmit this collision packet again. If one packet collide **MaxRetry** times, this packet will be drop and never try again.

Target MAC address

MAC_tx_add_en	R/w	7'd007	16'h0000
MAC_tx_add_prom_data	R/w	7'd008	16'h0000
MAC_tx_add_prom_add	R/w	7'd009	16'h0000
MAC_tx_add_prom_wr	R/w	7'd010	16'h0000

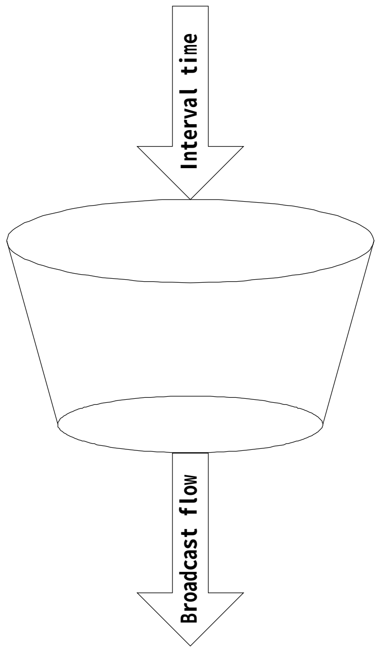
Those registers are used to set mac address which will replace the target mac address of transmit packet. This function will be enable one when register **MAC_tx_add_en** set to "1".

At the rising edge of signal **MAC_tx_add_prom_wr**, the value of **MAC_tx_add_prom_data** will be write to prom address **MAC_tx_add_prom_add .** You need repeat six times to write six bytes length target mac to prom.

Broadcast_filter

broadcast_filter_en	R/w	7'd018	16'h0000
broadcast_bucket_depth	R/w	7'd019	16'h0000
broadcast_bucket_interval	R/w	7'd020	16'h0000

The broadcast packet filter will enable only when broadcast_filter_en is set “1”.



The bucket wil be periodically refilled after **broadcast_bucket_interval** time. **broadcast_bucket_depth** register is used to setting the bucket depth. When a byte of broadcast packet is received, the bucket will decrease 1. If the bucket is empty, the received packet will be drop until the bucket is refilled next time. The radio of **broadcast_bucket_depth** to **broadcast_bucket_interval** will determine maximal broadcast packets throughput in on second. For example, if the radio is 0.1, the broadcast flow will be restricted to 10Mbps when MAC in 100Mbps mode. The **broadcast_bucket_depth** will determine the burst number of broadcast packets.

RX_APPEND_CRC

RX_APPEND_CRC	R/w	7'd021	16'h0000

In some condition, the user application need MAC to retain FCS of ethernet frame. When RX_APPEND_CRC signal is equal “1” , the FCS of ethernet frame will transmit to user application.

CRC_chk_en

CRC_chk_en	R/w	7'd024	16'h0000

By default, the receive logic will drop any packet with FCS checksum error. By setting **CRC_chk_en** register to zero, you can disable received packet FCS checking.

Packet length restrict

RX_MAX_LENGTH	R/w	7'd026	16'h2710
RX_MIN_LENGTH	R/w	7'd027	16'h0040

The length of Received packet beyond minimal length and maximal length will be droped in receive Fifo. if the packet is already trasmitting in user interface , a error flag will enclosed at end of packet .

Statistic counters

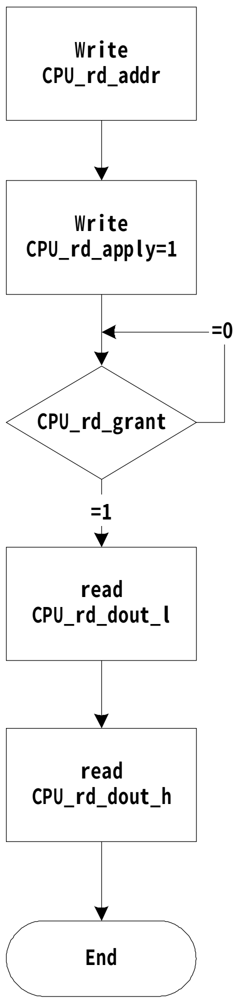
CPU_rd_addr	R/w	7'd028	16'h0000
CPU_rd_apply	R/w	7'd029	16'h0000
CPU_rd_grant	R	7'd030	16'h0000
CPU_rd_dout_l	R	7'd031	16'h0000
CPU_rd_dout_h	R	7'd032	16'h0000

All statistic counters are stored in a blockram. When you read a counter,you need to write the corresponding address to **CPU_rd_addr** register and assert **CPU_rd_apply** signal. When the counter data register **CPU_rd_dout** is available , the signal **CPU_rd_grant** will be “1”.

CPU_rd_addr	**Statistic counter**
6’h00	RxPacketLengthBytesCounter
6’h01	RxPacketCounter
6’h02	RxBroadcastCounter
6’h03	RxMulticastCounter
6’h04	RxUnicastCounter
6’h05	RxCRCErrCounter
6’h06	RxFifoFullErrCounter
6’h07	RxTooShortTooLongCounter
6’h08	RxPacketLength<64
6’h09	RxPacketLength_64
6’h0a	RxPacketLength_64_127

6’h0b	RxPacketLength_128_255
6’h0c	RxPacketLength_256_511
6’h0d	RxPacketLength_512_1023
6’h0e	RxPacketLength_1024_1517
6’h0f	RxPacketLength>1518
6’h10	RxPuaseFrameCounter

CPU_rd_addr	Statistic counter
6’h20	TxPacketLengthBytesCounter
6’h21	TxPacketCounter
6’h22	TxBroadcastCounter
6’h23	TxMulticastCounter
6’h24	TxUnicastCounter
6’h25	TxJamDropCounter
6’h26	TxFifoUnderflowCounter
6’h27	TxFifoOverflowCounter
6’h28	TxPacketLength<64
6’h29	TxPacketLength_64
6’h2a	TxPacketLength_64_127
6’h2b	TxPacketLength_128_255
6’h2c	TxPacketLength_256_511
6’h2d	TxPacketLength_512_1023
6’h2e	TxPacketLength_1024_1517
6’h2f	TxPacketLength>1518
5’h30	TxPuaseFrameCounter



line side loopback

Line_loop_en	R/w	7'd033	16'h0000

If **Line_loop_en** =1 , the packet transmited to Phy will loopback to receive side. This function is used for test purpose.

Speed level

Speed	R/w	7'd034	16'h0004

This register is used to set speed level of ethernet mac core.

3'b100: 1000Mbps mode

3'b010: 100Mbps mode

3'b001: 10Mbps mode

4 Clocks

This section specifies all the clocks. All clocks, clock domain passes and the clock relations should be described.

Name	Source	Rates (MHz)			Remarks	Description
		Max	Min	Resolution		
Clk_125M	Input Pad	125	-	-		For GMII interface
Clk_user	Input Pad	66	-	-		User clock
Clk_reg	Input port	80	50	-		Host interface cock

Table 1: List of clocks

5

IO Ports

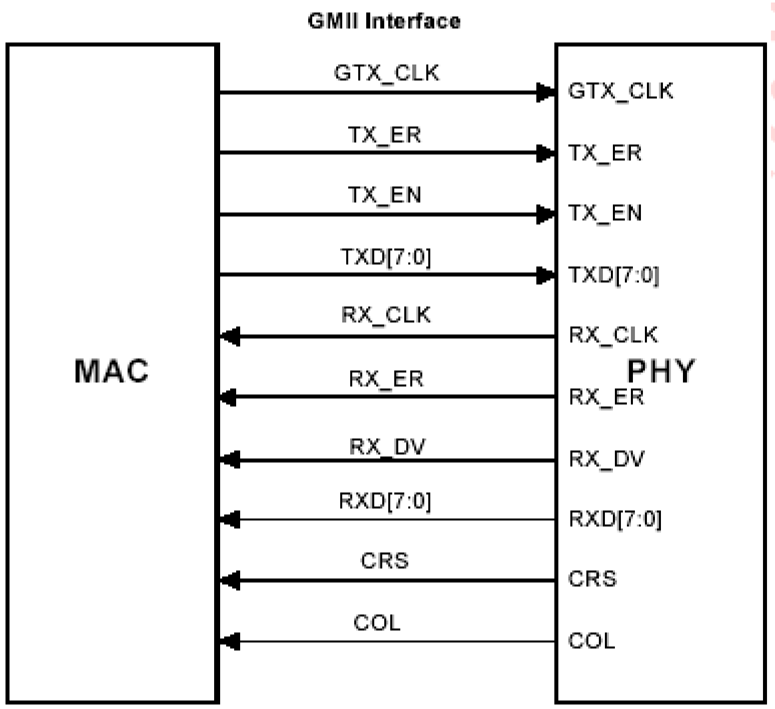
PHY interface

Gigabit Media Independent Interface (GMII/MII)

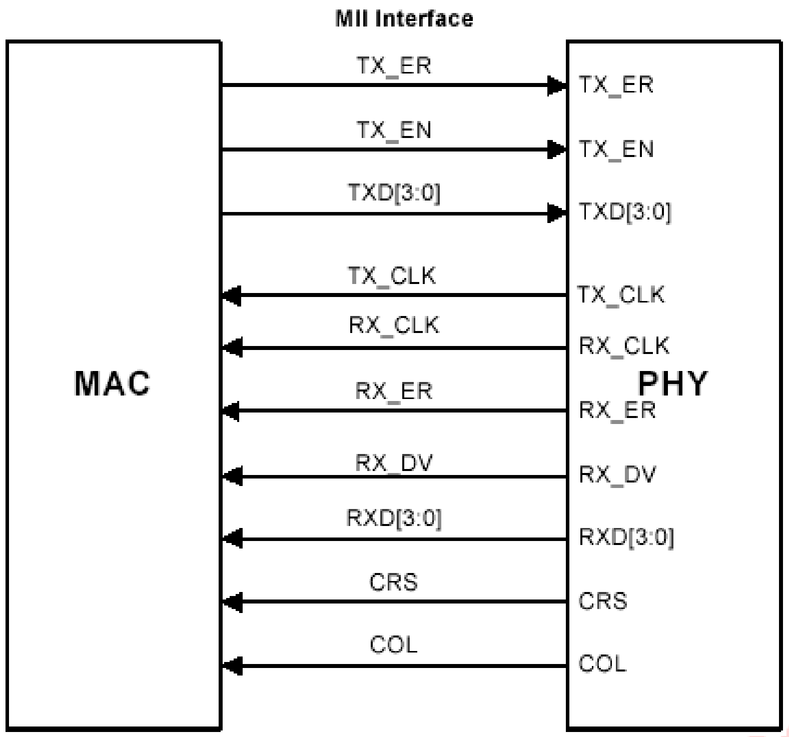
signal mapping

88E1111 Device Pins	GMII	MII
GTX_CLK	GTX_CLK	-
TX_CLK	-	TX_CLK
TX_ER	TX_ER	TX_ER
TX_EN	TX_EN	TX_EN
TXD[7:0]	TXD[7:0]	TXD[3:0]
RX_CLK	RX_CLK	RX_CLK
RX_ER	RX_ER	RX_ER
RX_DV	RX_DV	RX_DV
RXD[7:0]	RXD[7:0]	RXD[3:0]
CRS	CRS	CRS
COL	COL	COL

gmii signal diagram

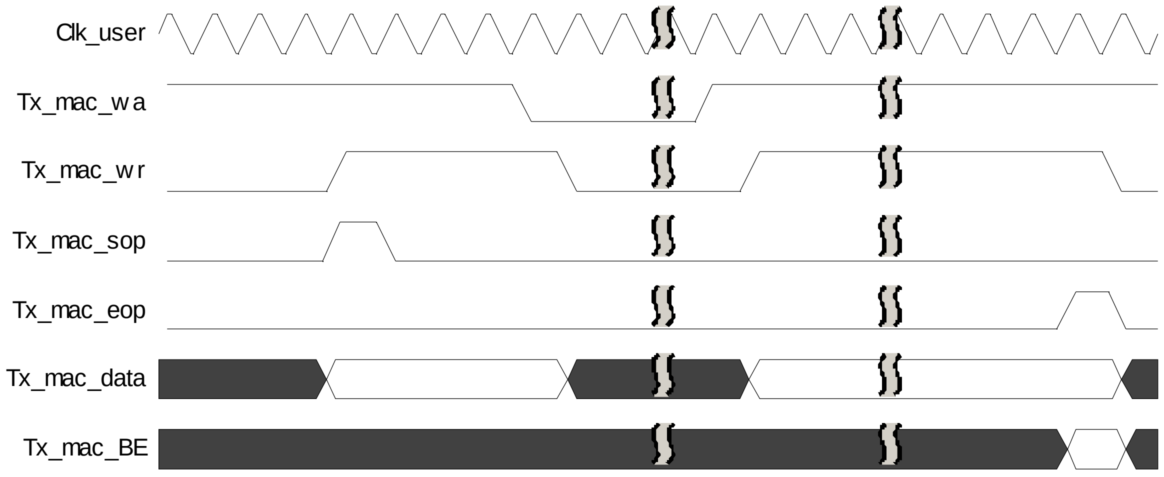

GMII Interface
MAC
PHY
GTX_CLK
TX_ER
TX_EN
TXD[7:0]
RX_CLK
RX_ER
RX_DV
RXD[7:0]
CRS
COL
GTX_CLK
TX_ER
TX_EN
TXD[7:0]
RX_CLK
RX_ER
RX_DV
RXD[7:0]
CRS
COL

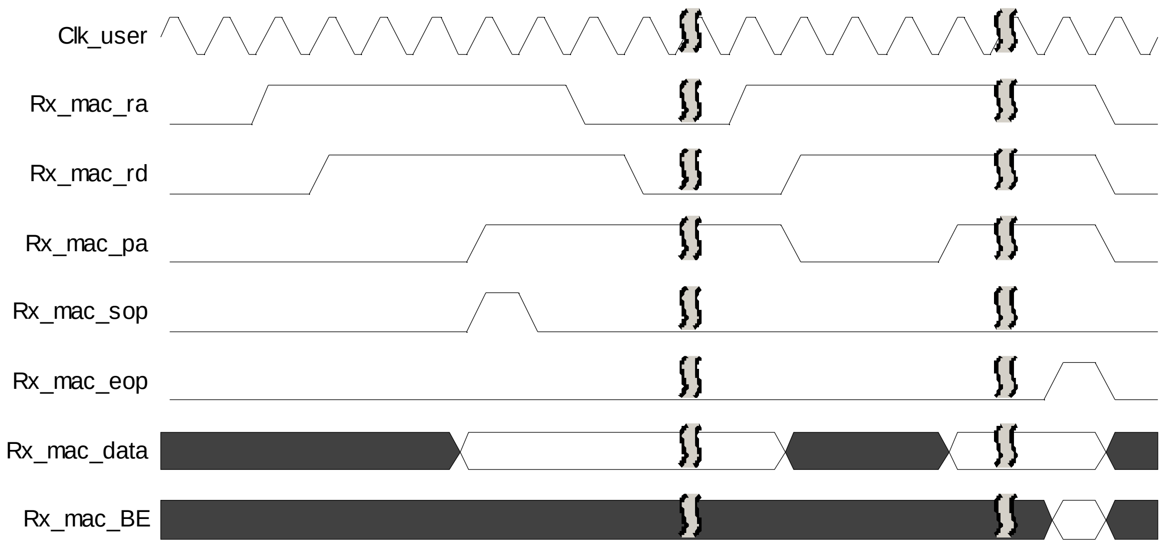
mii diagram


MII Interface
MAC
PHY
TX_ER
TX_EN
TXD[3:0]
TX_CLK
RX_CLK
RX_ER
RX_DV
RXD[3:0]
CRS
COL
TX_ER
TX_EN
TXD[3:0]
TX_CLK
RX_CLK
RX_ER
RX_DV
RXD[3:0]
CRS
COL

user interface

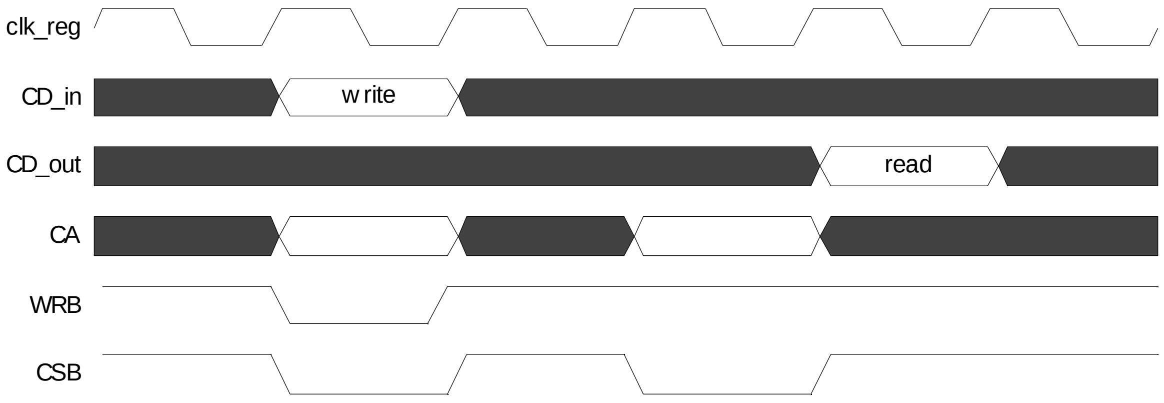
output		Rx_mac_ra	//
input		Rx_mac_rd	
output	[31:0]	Rx_mac_data	
output	[1:0]	Rx_mac_BE	
output		Rx_mac_pa	
output		Rx_mac_sop	
output		Rx_mac_eop	
			//
output		Tx_mac_wa	
input		Tx_mac_wr	
input	[31:0]	Tx_mac_data	
input	[1:0]	Tx_mac_BE	//
input		Tx_mac_sop	
input		Tx_mac_eop	





Host interface

input		CSB
input		WRB
input	[15:0]	CD_in
output	[15:0]	CD_out
input	[7:0]	CA



Appendix A

Name

This section may be added to outline different specifications.

Index

This section contains an alphabetical list of helpful document entries with their corresponding page numbers.