WebPHY DATABUS

User Manual

TABLE OF CONTENTS

INTRODUCTION....3
REFERENCED DOCUMENTS....3
CORE IO....3
USING THE CORE....5
Clock and Reset....5
Web Page ROM Interface....5
Network Parameters....5
User Databus Example Design....6
Status Port....7
10Base-T Interface....9
Generating the Web Page ROM....10
Web Page Development....10
Web Page Debug....12
Delayed ACK....12

Introduction

This manual is intended to assist in setting up and using the core for user applications.

Referenced Documents

Number	Title
RFC 2616	HTTP 1.1 Protocol
RFC-793	TCP Protocol
RFC826	ARP Protocol
RFC791	IP Protocol
IEEE 802.3i	10Base-T Ethernet Protocol

Core IO

Port Name	Description
clk/reset	
clk80	80MHz clock
rst80	Reset, synchronous to 80MHz clock
Web Page ROM Interface	
webrom_addr	Connect to generated web page ROM
webrom_rdata	Connect to generated web page ROM
webrom_rd	Connect to generated web page ROM
webrom_size	Connect to generated web page ROM
Network Parameters	
ip_address_src_1	Set IP address for core
ip_address_src_2	Set IP address for core
ip_address_src_3	Set IP address for core
ip_address_src_4	Set IP address for core
mac_address_src_1	Set Ethernet MAC address for core

mac_address_src_2	Set Ethernet MAC address for core
mac_address_src_3	Set Ethernet MAC address for core
mac_address_src_4	Set Ethernet MAC address for core
mac_address_src_5	Set Ethernet MAC address for core
mac_address_src_6	Set Ethernet MAC address for core
User Databus Interface	
addr	Drive user address lines
rd_en	Drive user rd enable
rd_ack	User drives with rd acknowledgment
rd_dat	User drives with rd data
wr_en	Drive user wr enable
wr_dat	Drive wr data to user
10Base-T Interface	
tx_p	Connect to FPGA pin (LVCMOS25)*
tx_n	Connect to FPGA pin (LVCMOS25)*
rx_p	Connect to FPGA pin (LVDS+)**
rx_n	Connect to FPGA pin (LVDS-)**
led_link_l	Connect to FPGA pin
led_act_l	Connect to FPGA pin
Status	
status	Status bits from core

* Note: to comply with Ethernet specifications, this voltage must be 2.5V.
** Note: correct polarity must be observed for FPGA LVDS IO

Using the Core

Clock and Reset

The WebPHY-*DATABUS* core requires an 80MHz clock for correct operation. The reset input is asserted active HI and must be synchronous to the 80MHz clock. All other ports on the core are synchronous to the clk input port.

Web Page ROM Interface

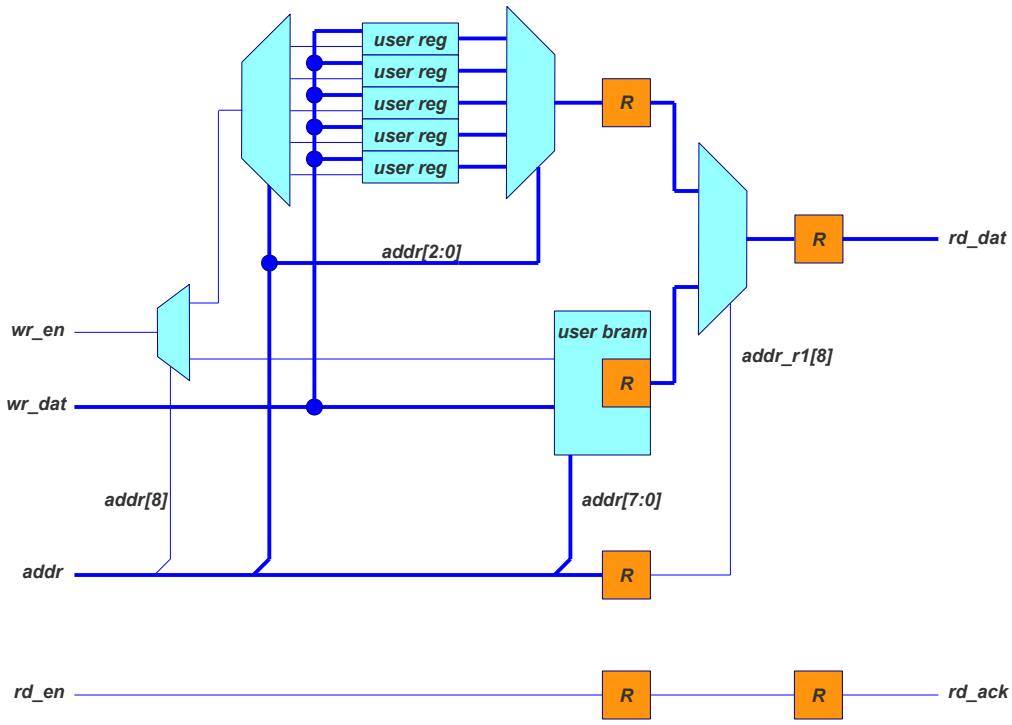
These ports must be connected to the corresponding ports on the generated VHDL ROM module. The core file and the generated ROM are then instantiated in the user's design.

Network Parameters

These ports set the core's 48-bit Ethernet MAC address and 32-bit IPv4 address. The ports can be set as constants at build time or connected to registers for run time configuration. Note that the values of these ports must be held constant during HTTP transactions to prevent errors. It is recommended to hold the core in reset if changing these parameters during run-time.

User Databus Example Design

WebPHY-*DATABUS* acts as a bus master which reads and writes the user's address space. A read acknowledgment is used to allow for pipeline latency in the user's read back logic, and serves and a “data valid” qualifier for read back data from the user to the core. In the included example design, a 256-byte BRAM and several read-back registers are connected to the WebPHY-*DATABUS* core. Below is the diagram:



In this example design, the user's 256-byte BRAM is instantiated at addresses 0 – 255, and five read-back registers are instantiated at addresses 256 – 260. When a wr command is received from the web-client, the core drives the data bus by asserting *addr*, *wr_en* and *wr_dat* on the same clock cycle, for each byte written. When a rd command is received, the core asserts *addr* and *rd_en* for each byte read. The orange *R* blocks are pipeline registers used to re-time the read data path. The resulting latency requires two pipeline registers to be added between *rd_en* and *rd_ack,* so that *rd_ack* is asserted synchronously with valid data on *rd_dat*.

Status Port

The status port contains several outputs which can be useful for debugging and general health checking of the WebPHY-DATABUS core.

Bit Name	Functional Name
status(0)	tcp_connected
status(1)	tcp_pkt_rx
status(2)	tcp_pkt_tx
status(3)	tcp_pkt_retransmit
status(4)	tcp_bad_rx_checksum
status(5)	arp_req_rcvd
status(6)	eth_bad_rx_crc
status(7)	eth_rx_tst
status(8)	eth_tx_tst

tcp_connected
This output remains high during TCP connection. Since only non-persistant HTTP connections are supported by the core, this should only be high for the length of the wr or rd command transaction.

tcp_pkt_rx
This output pulses whenever a TCP packet with a correct sequence number, TCP checksum and Ethernet CRC is received.

tcp_pkt_tx
This output pulses whenever the core sends a TCP packet.

tcp_pkt_retransmit
This output pulses whenever the core retransmits a TCP packet due to ACK timeout from the peer.

tcp_bad_rx_checksum
This output pulses whenever a received TCP packet fails TCP checksum and is discarded. This may occur when communicating with the core over the Internet and a TCP segment is corrupted.

arp_req_rcvd
This output pulses whenever the core responds to an ARP request.

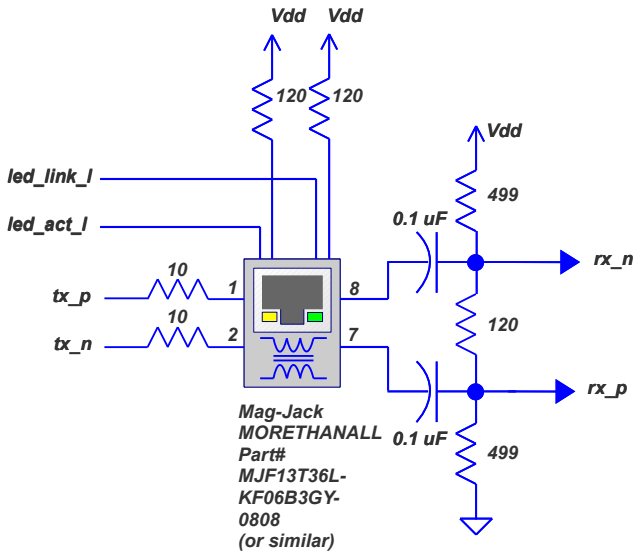
eth_bad_rx_crc
This output pulses whenever a received Ethernet frame fails Ethernet CRC and is discarded. This may occur when communicating with the core over a LAN and the frame is corrupted.

eth_rx_tst
This output is the single-ended output of the differential buffer used to received 10Base-T data. It can be connected to a scope for physical layer debug.

eth_tx_tst
This output replicates the tx_p output from the core used to transmit 10Base-T data. It can be connected to a scope for physical layer debug.

10Base-T Interface

The WebPHY-*DATABUS* core features a built-in 10Base-T interface, requiring only a biasing network and Ethernet mag-jack to connect the FPGA to a network.



The Mag-Jack provides the galvanic isolation required for Ethernet while the capacitors and biasing provide the interface from the magnetics to the FPGA's LVDS I/O. For debugging, the test-points *eth_rx_tst* and *eth_tx_tst* are provided. *eth_rx_tst* is the single-ended output of the differential LVDS receiver contained in the core, and can be used to help verify wire-layer receipt of Ethernet packets and the NLP (normal link pulse) by the core. When connected to an Ethernet link partner, an incoming NLP should be seen at this test-point as a high-going, 100 nS-wide pulse, occurring every 16 mS, and should cause the Ethernet Link LED to illuminate. Incoming and outgoing Ethernet packets will cause the Activity LED to illuminate. *eth_tx_tst* is a replication of the *tx_p* output from the core and can be connected to an oscilloscope to see packets being transmitted from the core.

During PCB layout, ensure that 100-ohm differential impedance is maintained for differential runs for optimal signal integrity.

Note that the core does not support the CSMA/CD portion of the 10Base-T specification. As a result, the core cannot be connected to a collision domain with other stations, and therefore network connection via hub is not recommended. Please use a switch instead.

Note that when connecting the core to a PC or laptop, an Ethernet crossover cable may be required for proper operation.

Generating the Web Page ROM

The user creates web content in a single HTML file and runs the provided Perl script to generate an initialized VHDL-inferred BRAM “ROM”. This ROM is then attached to the core and instantiated in the user design. Here are the steps to perform the ROM generation:

1. Ensure Perl is installed on the development system.
2. Edit the .html file containing the user web content, for example mywebpage.html.
3. At the command line run the Perl script, specifying the paths of mywebpage.html and the VHDL ROM that will be generated, for example mywebpagerom.vhd:

   perl romgen.pl mywebpage.html ../src/mywebpagerom.vhd
4. Instantiate the generated mywebpagerom.vhd in your design and connect it to the WebPHY- *DATABUS* core.

Web Page Development

WebPHY- *DATABUS* allows up to 64k bytes of user web-content, which can be completely customized by the user. This allows design-specific GUI controls to be added, such as buttons, check boxes, text boxes, radio buttons and slider controls.

The Javascript language offers powerful client-side development constructs with wide browser support. This allows string formatting, file IO, floating-point math operations, and other advanced capabilities which can be brought to the FPGA interface without burdening the FPGA itself. The combined WebPHY-*DATABUS* core and client-side Javascript creates a powerful and seamless user-interface between the FPGA and the client.

To communicate with the core, the Javascript code running on the web-client sends and receives data using the two commands shown in the following examples:

```
wr 0x0 0x1234 --Write 0x1234 to 0x0
rd 0x0 0x02 --Read 2 bytes from 0x0
```

Note that these strings are case sensitive when sent to the core, and must be sent as shown. The rd and wr must be lower, and the x in 0x must be lower case. Javascript, for example, can be used to automatically convert the case of a command entered by a user in a text box to the required case, as is done in the included example web page.

The core is capable of parsing and executing multiple wr commands in the same HTTP POST transaction. This means that a file containing multiple wr commands can be POSTed to the core in a single shot. The included web page features a file browse dialog to demonstrate this capability. Below is an example file containing multiple wr commands:

```
wr 0x0 0x12345678
wr 0x4 0xABCD
wr 0x6 0x00112233
```

Web Page Debugging

Mozilla provides a HTML/Javascript debugger add-on called FireBug for its Firefox browser. This allows real-time debug and even some on-the-fly editing of custom web-page code as it runs on the host communicating with the FPGA. Breakpoints, watch windows, and other useful debug utilities are available using this tool which can greatly aid in debug of custom web-pages for WebPHY-*DATABUS*. In addition, the web page file can be debugged offline by simply opening it in a web-browser.

Web Page Live Upload

The WebPHY-*DATABUS* core allows live uploading of web pages from a web client. This allows for rapid development of web pages for the core. In addition to generating the initialized block RAM VHDL, the included Perl script also generates an upload file containing the bytecode for the user's web page. POSTing this command to the core results in updating the core with the new web page.

Troubleshooting and known issues

Google Chrome

Google Chrome TCP pre-connections immediately following web page loads (see Chromium issue 116982). This ties up the DATABUS core and causes unstable behavior immediately following web-page loads. This can be remedied by disabling the “predict network actions to improve page load performance” option in the Chrome Advanced Settings menu.

TCP Delayed Acknowledgment

Some operating systems such as Windows implement TCP delayed acknowledgment. This causes the OS's TCP stack to delay sending acknowledgment segments by up to 200 mS in anticipation of receiving more segments from the peer. Since the DATABUS core supports one outstanding transmitted segment, a 200 mS delay is added to every acknowledgment, resulting in slow upload speeds from the core. To remedy this situation, TCP Delayed Acknowledgment can be disabled in the OS. Note that this does not affect upload speeds to the core.