GRETH ERRATUM: Overrun May Cause GRETH Receiver to Hang

Technical note

2015-10-29

Doc. No GRLIB-TN-0008

Issue 1.0

CHANGE RECORD

Issue	Date	Section / Page	Description
1.0	2015-10-29		First issue.

TABLE OF CONTENTS

1 OVERVIEW 3
1.1 Scope of the Document 3
1.2 Affected versions 3
1.3 Affected components 3
1.4 How to check if LEON3/LEON3FT design is affected 3
1.5 Distribution 3
1.6 Contact 3
2 DESCRIPTION 4
3 IMPACT 4
4 WORKAROUND 4
5 FAQ 5
5.1 Can I see in the GRETH RTL code if I am affected? 5

1 OVERVIEW

1.1 Scope of the Document

This document describes a problem in specific versions of the GRETH Ethernet Ethernet Media Access Controller (MAC).

1.2 Affected versions

The errata is present in all GRETH versions prior to GRLIB 1.1.0-b4104. The problem does NOT affect the GRETH_GBIT IP core.

1.3 Affected components

Cobham and Aeroflex components that are affected are:

- GR712RC
- LEON3-RTAX – versions with GRETH with a build ID lower than 4104.
- UT699

1.4 How to check if LEON3/LEON3FT design is affected

If you are licensing GRLIB for use in your own FPGA or ASIC design, you can check the following conditions in the design's VHDL source to see if the erratum applies to your system:

Check the GRLIB revision. This can be seen in the file name of the release package, in the directory name after unpacking the release, and in the file *lib/grlib/stdlib/version.vhd* in the release file tree (constant grlib_build). If this is equal to or higher than 4104 then you are NOT affected by this erratum.

1.5 Distribution

Users that have devices with the affected errata are free to use the material in this document in their own errata sheets. Contact Cobham Gaisler for inquires on other distribution.

1.6 Contact

For questions on this errata, please contact Cobham Gaisler support at support@gaisler.com. When requesting support, include the part name if the question is a specific device or the full GRLIB IP library package name if the question relates to a GRLIB IP library license.

2 DESCRIPTION

The state in which the GRETH receiver control finite state machine discards packets did not take overrun into account in all cases. The discard state is entered when a received packet is determined to be dropped. The size of the packet is checked and that amount of data is read from the FIFO. When an overrun has occurred, the amount written to the FIFO is not the same as the length of the packet and in those cases the FIFO should be emptied completely. This was not done if:

1. The packet in question had a MAC address which the core should not receive, or
2. if the receiver was enabled when the packet was received but the descriptor was not enabled and the descriptor read was so slow that an overrun occurred during that time.

3 IMPACT

The issue described in the previous section causes the Ethernet receiver to hang. No further traffic can be received and the GRETH must receive a hardware reset to recover. This hardware reset can be attained by either reseting the whole device or, in devices that have individual clock gating of the Ethernet controller, forcing a reset of the GRETH via the system's clock gating unit.

4 WORKAROUND

As described in section 2, the problem occurs when a packet is received and

1. the packet in question had a MAC address which the core should not receive, or
2. if the receiver was enabled when the packet was received but the descriptor was not enabled and the descriptor read was so slow that an overrun occurred during that time.

The first condition can be avoided by enabling promiscuous mode for the Ethernet controller (note that this mode is called Open Packet Mode in the UT699 documentation). This mode is enabled via the GRETH control register. Promiscuous mode may lead to additional software load of the systems since more packets may need to be processed. In applications where the GRETH is used in a limited network with few members the addition of promiscuous mode should have less negative effect.

Use of switches instead of hubs in the network since the GRETH controller will in this case receive fewer packets with MAC addresses that will be discarded.

The second trigger condition is system and application specific. The condition is not expected to happen in systems like the affected components listed earlier in this document. Other systems with memory controllers that have high latency and systems that introduce several bridges between the GRETH controller and main memory may encounter condition 2.

There is no status bit in the controller that will indicate that the receiver has hanged. The condition can be detected by the ethernet controller no longer receiving any packets. This could require implementation at a higher level of a heart beat function so that system software can detect that packets are missing and correct the situation. The only way to correct the condition is to perform a hardware reset of the GRETH controller. The soft reset functionality that can be triggered via the GRETH control register is not enough to resolve the receiver hang.

5 FAQ

5.1 Can I see in the GRETH RTL code if I am affected?

The patch that corrects the issue is listed below:

```
diff --git a/lib/eth/core/grethc.vhd b/lib/eth/core/grethc.vhd
index 3824fd0..bf2db02 100644
--- a/lib/eth/core/grethc.vhd
+++ b/lib/eth/core/grethc.vhd
@@ -1010,14 +1010,25 @@ begin
        v.status.rxahberr := '1'; v.ctrl.rxen := '0';
      end if;
    when discard =>
-     if (r.rxdoneold = '0') or ((r.rxdoneold = '1') and
-        (conv_integer(r.rxcnt) < conv_integer(r.rxbytecount))) then
+     if (r.rxdoneold = '0') then
        if conv_integer(r.rfcnt) /= 0 then
          v.rfrpnt := r.rfrpnt + 1; v.rfcnt := r.rfcnt - 1;
          v.rxcnt := r.rxcnt + 4;
        end if;
-     elsif (r.rxdoneold = '1') then
-       v.rxdstate := idle; v.ctrlpkt := '0';
+     else
+       if r.rxstatus(3) = '1' then
+         v.rfcnt := (others => '0'); v.rfwpnt := (others => '0');
+         v.rfrpnt := (others => '0'); v.writeok := '1';
+         v.rxbytecount := (others => '0'); v.rxlength := (others => '0');
+         v.rxdstate := idle;
+       elsif (conv_integer(r.rxcnt) < conv_integer(r.rxbytecount)) then
+         if conv_integer(r.rfcnt) /= 0 then
+           v.rfrpnt := r.rfrpnt + 1; v.rfcnt := r.rfcnt - 1;
+           v.rxcnt := r.rxcnt + 4;
+         end if;
+       else
+         v.rxdstate := idle; v.ctrlpkt := '0';
+       end if;
      end if;
    when others =>
      null;
```