HyperTransport™ Technology

HyperTransport to Optic Converters

Rev. 1.01 6/2002

The HyperTransport Technology Consortium disclaims all warranties and liability for the use of this document and the information contained herein and assumes no responsibility for any errors that may appear in this document, nor does the HyperTransport Technology Consortium make a commitment to update the information contained herein.

DISCLAMER

This document is provided "AS IS" with no warranties whatsoever, including any warranty of merchantability, noninfrigment, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification or sample.

The HyperTransport Technology Consortium disclaims all liability for infringement of property rights, relating to use of information in this document. No license, express, implied, by estoppels, or otherwise, to any intellectual property rights is granted herein.

Trademarks

HyperTransport is a trademark of the HyperTransport Technology Consortium.

AMD is a trademark of Advanced Micro Devices, Inc.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

HyperTransport Technology is a result of collaborative development efforts between AMD and other industry partners. This new high-speed, high performance link specification is being offered as an open standard through membership to the HyperTransport Technology Consortium.

About HyperTransport™ Technology

HyperTransport technology is a high-speed, high-performance, point-to-point link for integrated circuits, and is designed to meet the bandwidth needs of tomorrow's computing and communications platforms. HyperTransport technology helps reduce the number of buses while providing a high-performance link for PCs, workstations, and servers, as well as numerous embedded applications and highly scalable multiprocessing systems. It is designed to allow chips inside of PCs, networking and communications devices to communicate with each other up to 48 times faster than with some existing bus technologies.

About the HyperTransport Technology Consortium

The HyperTransport Technology Consortium is a nonprofit corporation managed by its members. The consortium promotes the common business interests of providers to the networking, telecommunications, computer and high-performance embedded application through the conduct of a forum for the future development and adoption of the HyperTransport specification.

AMD, Apple Computers, Broadcom, Cisco Systems, NVIDIA, PMC-Sierra, SGI, Sun Microsystems, and Transmeta are the charter members that comprise the Executive Committee of the HyperTransport Technology Consortium.

Companies interested in the HyperTransport specification are invited to join the consortium. Members of the consortium pay annual dues and receive a royalty-free license to HyperTransport IP, gain access to technical documentation and may attend consortium meetings and events. To become a member, visit the consortium Web site at www.hypertransport.org. Please review the Bylaws of the HyperTransport Technology Consortium, and complete the online membership application.

Revision History: HyperTransport to Optics Converters

Rev	Date	Comment
1.00	05.02	Document created.
1.01	06.02	Document revised and updated.

Preface

This application note presents the outlines of two serialization alternatives to give users added options when connecting HyperTransport 1.x components that may be on different PC boards or in different shelves. These alternatives consume more power and add latency compared to the native HyperTransport-1.x interface. These alternatives are:

- Utilizing the XGMII/XAUI components standardized for use in 10Gb/s ethernet applications.
- Utilizing parallel optic components designed for use with fiber optic ribbon cables.

A challenging aspect of this problem is HyperTransport has an 8.25-bit interface, not an 8-bit interface. Although the CTL line arrives every clock edge, it is only allowed to change every 32 bits. CTL distinguishes control double words from data double words. It is somewhat difficult to reduce this interface to an eight-bit interface since the HyperTransport I/O Link allows control packets to be inserted in the middle of data packets. The solutions presented herein collect the CTL bit for each 32 bits and carry it transparently.

Note: *This application note may contain errors. At the time of its preparation, this note had not been realized into a device and may incorporate errors or contain omissions. Regard this document as an outline of how to approach these two problems rather than as a specification.*

HyperTransport to XGMII/XAUI Conversion

The following is the outline of a conversion function between HyperTransport rev 1.x and XGMII. XGMII may be useful for some users because through XGMII you can get to the 10G ethernet PHYs as well as to XAUI (4x3.125 Gb/s) and other 10GE solutions over one fiber. Through XAUI, a HyperTransport I/O link can be carried over a single fiber in at a moderate cost by using some Coarse WDM transceivers that are available in the marketplace. These transceivers utilize the XENPAK Multi-Source Agreement. This outline shows a means of allowing a flexible number of XGMII/XAUI interfaces to be used, supporting differing bandwidths.

A representative part to try to connect to is the PM8355 QuadPHY-II device from PMC-Sierra. This device has four bidirectional SERDES parts in it. These SERDES can run at 3.125 Gb/s each. The part as a whole can pass a 10-gigabit ethernet.

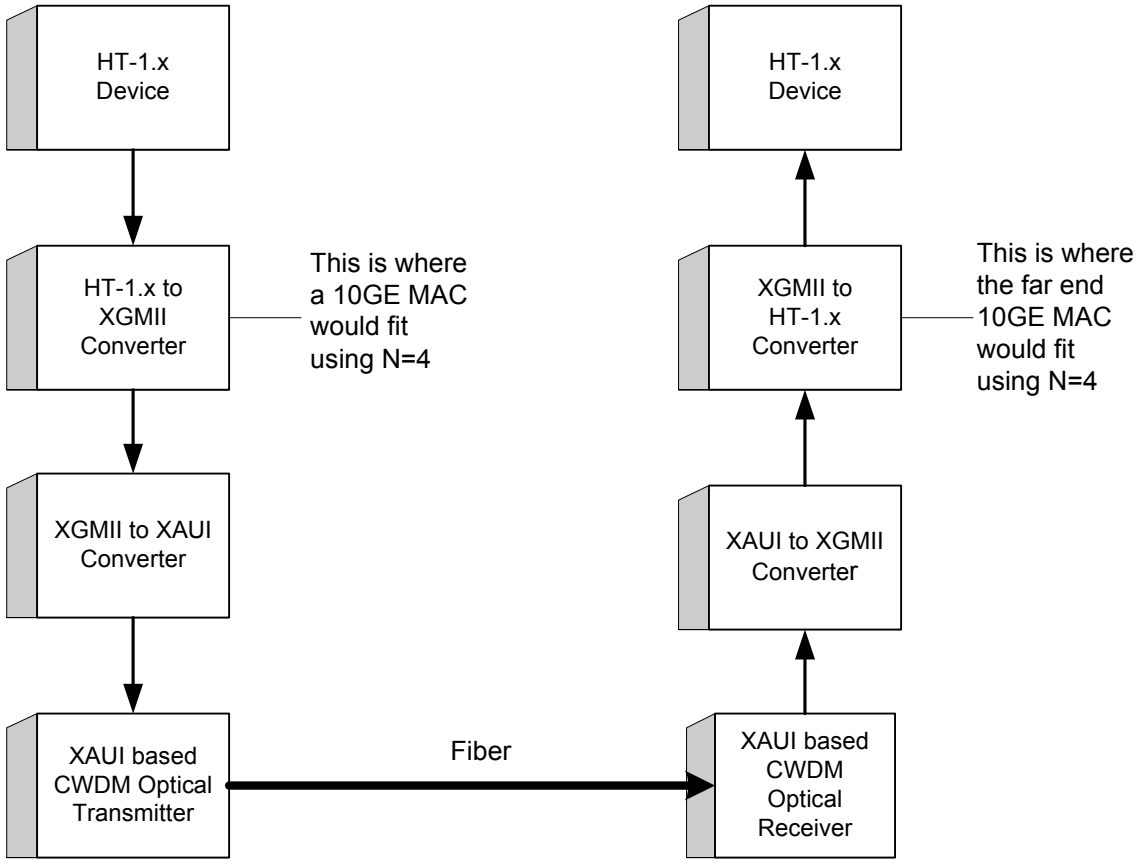
References

For details about XGMII and XAUI, see the IEEE 802.3ae specification at
http://grouper.ieee.org/groups/802/3/ae/index.html

For details about the PM8355, go to www.pmc-sierra.com

For details about XAUI transceivers, go to www.xenpak.org

Block Diagram of HyperTransport to XGMII Conversion



The XGMII/XAUI Converter

1. Use the XGMII/XAUI solution but disable the trunking feature found some multi-channel XAUI parts. Details can be found in the IEEE 802.3ae spec.

2. Select the HyperTransport clock frequency that fits the available XGMII bandwidth with some overhead (the amount as per step 15 below).

3. Label the N XGMII interfaces 0 to (N-1) and their collected data inputs 0 to (N*8-1). M is the label for the Mth XGMII interface.

4. Every J bytes have the HT-1.x to XGMII converter put a comma character into interface 0. (An XGMII interface has nine bits, 1 for control and 8 for data. A comma character is a special control code.)

5. At a K byte offset to the next lower numbered XGMII interface, insert a comma character. Chose the K so that J = K * N. With knowledge of J and K, the receiver can deskew the byte lanes by tracking the position of the comma characters. Spreading the bytes out reduces the latency of the solution.

6. Create a frame 80 bytes long consisting of 16 sets of an Overhead byte plus 4 bytes which contain the information found on the CAD lines. Delimit this frame with a special 8B10B non-comma command code.

7. Encode bits 7 and 5 of the Overhead byte as the CTL bit and bit 6 as the inverse of the CTL bit, all associated with the following 4 CAD bytes. Bit 4 is reserved

8. Encode bits 3 to 0 of the Overhead byte with the following message:

    a. Nibble 0 - bit 3 PwrOK, bit 2 RESET#, bit 1 LDTSTOP#, bit 0 LDTREQ#

    b. Nibble 1 - FrameCount – incrementing MOD 16, 4 bit count of frames

    c. Nibbles 2-3 – the link width as determined per section 12.2 of HT 1.03

    d. Nibbles 4-7 – a CRC-16 on the previous frame including both the user data and the CTL Overhead data

    e. Nibbles 8-11 – count of the # of non-user bytes included between the 8B10B delimiters for the previous frame

    f. Nibbles 12– 15 – 16 bit user-defined data

9. Don't use any of the data in the Overhead bytes until they have been validated by the CRC-16.

10. If there is not enough data to fill the XGMII line, put a special 8B/10B character on the line (by using the XGMII CTL pin) that is different from either the comma character or the frame delimiter control character of step 6. Design it so that one of these characters is inserted every P bytes on average.

11. By interpreting the XGMII control words, the receiver can reconstruct the exact number of clock pulses that were received from the HT-1.x interface by the transmitter. Use the count of the # of non-user bytes in the Overhead Word as a check of the observed counts to make the PLL more tolerant to bit errors.

12. Produce a reconstruction of those clock pulses.

13. Smooth that reconstructed clock with a PLL. Use the smoothed clock to drive the HT-1.x interface.

14. Pass the receive data through a FIFO and read it out with the smoothed clock, sending it over the HT-1.x interface.

15. The HT-1.x clock frequency that can supported is determined from the speed of the XGMII interface and the values of J, K N, and P from above.

16. Use the data from the Overhead byte to control the PwrOK, RESET#, LDTSTOP# and LDTERQ# outputs. See discussion below.

Implementation

PWROK, RESET#, LDTSTOP#, and LDTREQ# Support

Since these signals are Open Drain signals, the conversion function must remember which end asserted them. If a signal is asserted in one direction, it must not be asserted in the reverse direction through the link. If both ends assert the signal simultaneously, the link should remember that the signal was asserted when the assert from the other end arrived and should not assert it locally.

Link Width Indication

Because the conversion device may be narrower than is supported by the two ends, the conversion device must participate in the link width auto negotiation procedure by propagating on the CAD lines the minimum of its own supported width and the far end's supported width as described in section 12.2 of the HyperTransport I/O Link specification rev 1.03.

Link Frequency Support

The PLL in the receiver must be agile enough to support a range of frequencies. HyperTransport I/O Link specification rev 1.03 defaults to 200 MHz on reset, so this value must be supported by the HyperTransport to XGMII conversion device. The upper frequency range is application specific, but the conversion device must support all of the configurable frequencies between 200 MHz and the maximum frequency.

The conversion device may support this by observing the link frequency at the transmit end and reproducing that frequency at the receive end.

Because the maximum supported frequency of conversion device and the XGMII/XAUI link may be less than what is supported by the HyperTransport nodes, the system software may need to be aware of the presence and limitations of the conversion device when configuring the link frequency CSR bits in the HyperTransport nodes. An alternative solution is for the transparent link to become either a HyperTransport tunnel device or a pair of HyperTransport tunnel devices. In this case, the link frequency CSR bits would then be configured as normal.

Latency

The latency of this solution is significant, perhaps on the order of 14 or more byte times. If you want lower delay, you could generate the XAUI signals directly. The difficulty with this solution is that the rates are high.

The added latency of the XGMII solution may increase the buffer sizing required in the nodes to which it connects for those nodes to support full bandwidth. A standard problem when adding latency between nodes is that it may bring out operational problems within those nodes.

Errors

Since HyperTransport I/O Link specification rev 1.03 is intolerant of errors, and this overall solution is likely to have a measurable error rate, this link should only be used with the 1.x release that adds the error handling protocol.

Implementation Discussion

An advantage of this solution is that the converter chip is all at moderate frequency. Another advantage of this solution is that it leverages the 10GE components.

The implementation has some FIFOs, barrel shifters at each end and a PLL at the far end.

Alternative Implementation

If the Transmit clocks at the two ends are synchronized through some other means, it may be possible to loop back the transmit clock at each end for use as the receive clock. This avoids the need for a PLL.

HyperTransport to Parallel Optics Conversion

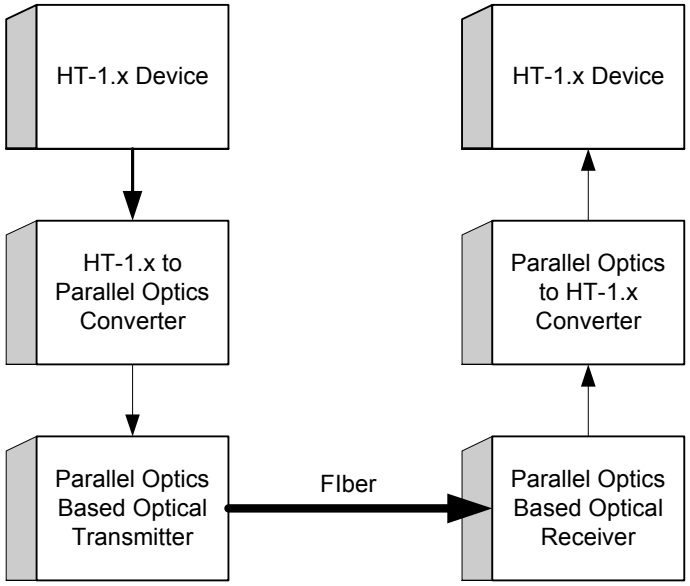
The following is the outline of a conversion function between HT -1.x and Parallel Optics components.

A representative part to try to connect to is the Infineon V23832-T1431-M1 and V23832-R421-M1 Parallel Optical Link devices or "PAROLI" device. This pair of devices has 12 LVDS I/O links. It has no clock encoding or recovery, instead it has simple LVDS in and out. This device interfaces with fiber optic ribbon cables.

Reference

http://www.infineon.com/cmc_upload/documents/037/639/V23832_T1431-R421_M1_020123.pdf

Block Diagram of HyperTransport to Parallel Optics Conversion



The Parallel Optics Converter

1. Select the HyperTransport clock frequency that fits the available parallel optics bandwidth.
2. Label the PAROLI I/O Opt[11:0}
3. In the transmit portion of the converter, Retime the CAD input as per the HyperTransport electrical specification.
4. Since the optics typically have 12 transceiver pairs. The CAD information will always appear on Opt[7:0]
    a. When connected to a 2 bit interface use Opt[1:0]
    b. When connected to a 4 bit interface use Opt[3:0]
    c. When connected to an 8 bit interface, use Opt[7:0]
    d. When connected to a 16 bit interface, put the information retimed from CAD[7:0] first on Opt[7:0] followed by the data retimed from CAD[15:8]
    e. When connected to a 32 bit interface, put the information retimed from CAD[7:0] first on Opt[7:0] followed by the data retimed from CAD[15:8], then CAD[23:16], then CAD[31:24].

5. Put the CTL bit on Opt[8]

    a. For 2, 4, and 8 bit interfaces, put the retimed CTL bit on directly

    b. For 16 bit interfaces, put a duplicated copy of the associated CTL bit for each of the two bytes.

    c. For 32 bit interfaces, put four duplicate copies of the associated CTL bit for each of the four bytes.

6. Create an 8 byte Overhead frame as per the following:

    a. Nibble 0 - bit 3 PwrOK, bit 2 RESET#, bit 1 LDTSTOP#, bit 0 LDTREQ#

    b. Nibble 1 - FrameCount – incrementing MOD 16, 4 bit count of frames

    c. Nibbles 2-3 – the link width

    d. Nibbles 4-7 – a CRC-16 on the previous 16 byte frame including both the user data and the CTL data

    e. Nibbles 8– 15 – 16 bit user-defined data

7. Encode that frame via byte synchronous HDLC.

8. Put that encoded frame on Opt[9] every 16 bytes (which is slower than the worst possible HDLC expansion).

9. Don't use any of the data in the Overhead bytes until they have been validated by the CRC-16.

10. Put a half rate (DDR style) clock on Opt[10] with the rate appropriate to the data to be handled, presumably using a PLL to create that clock.

11. In the receiver, use the clock from Opt[10] to retime the data.

12. Also use the clock on Opt[10] to drive a PLL to generate the clock to go out on the CLK pin dividing as per step 5.

13. Use the transitions on the CTL bits on Opt[8] to indicate how to demultiplex the data to the proper CAD outputs.

14. Put the CTL bits from Opt[8] out on the CTL line at the proper moment.

15. Use the information from the Overhead frame to control the PwrOK, RESET#, LDTSTOP#, and LDTREQ#. See discussion below.

Implementation

PWROK, RESET#, LDTSTOP#, and LDTREQ# Support

Since these signals are Open Drain signals, the conversion function must remember which end asserted them. If a signal is asserted in one direction, it must not be asserted in the reverse direction through the link. If both ends assert the signal simultaneously, the link should remember that the signal was asserted when the assert from the other end arrived and should not assert it locally.

Link Width Indication

Because the conversion device may be narrower than is supported by the two ends, the conversion device must participate in the link width auto negotiation procedure by propagating on the CAD lines the minimum of its supported width and the far end's supported width as described in section 12.2 of the HyperTransport specification rev 1.03.

Link Frequency Support

The PLL in the receiver must be agile enough to support a range of frequencies. HyperTransport I/O Link specification rev 1.03 defaults to 200 MHz on reset, so this value must be supported by the HyperTransport to XGMII conversion device. The upper frequency range is application specific, but the conversion device must support all of the configurable frequencies between 200 MHz and the maximum frequency.

The conversion device may support this by observing the link frequency at the transmit end and reproducing that frequency at the receive end.

Because the maximum supported frequency of conversion device and the XGMII/XAUI link may be less than what is supported by the HyperTransport nodes, the system software may need to be aware of the presence and limitations of the conversion device when configuring the link frequency CSR bits in the HyperTransport nodes. An alternative solution is for the transparent link to become either a HyperTransport tunnel device or a pair of HyperTransport tunnel devices. In this case, the link frequency CSR bits would then be configured as normal.

Latency

The latency of this solution is significant, perhaps on the order of 10 or more byte times. If you want lower delay, you could generate the XAUI signals directly, but the rates are high. The added latency may increase the buffer sizing needed in the nodes to achieve full bandwidth.

The added latency of the XGMII solution may increase the buffer sizing required in the nodes to which it connects for those nodes to support full bandwidth. A standard problem when adding latency between nodes is that it may bring out operational problems within those nodes.

Errors

Since HyperTransport I/O Link specification rev 1.03 is intolerant of errors, and this overall solution is likely to have a measurable error rate, this link should only be used with the 1.x release, which adds the error handling protocol.