AU-F8080: IEEE 1394b Link Layer Controller Core

The AU-F8080 is the Link Layer Controller for IEEE 1394b. It is a fully synthesizable core, which can be integrated seamlessly with any application. The AU-F8080 interfaces to the PHY Chip using the IEEE 1394b-2002 parallel PHY standard. On the application side, a simple interface provides direct access to receive and transmit FIFOs. The AU-F8080 supports 100, 200, 400, 800, and 1600 Mbits/s transfers. The 1394b LLC Core is available as a synthesizable Verilog model from Aurora VLSI, Inc. Contact CustomerService@auroravlsi.com.

The AU-F8080 can transmit and receive asynchronous and isochronous packets. Transmit and receive FIFOs are provided for data buffering in high-speed applications. There are separate transmit FIFOs for asynchronous and isochronous data. On the receive side, there is a common receive FIFO. The FIFO management unit provides status of all the FIFOs. At the end of a receive or transmit transfer, the application is notified of the transfer status using separate receive and transmit status signals along with a status qualifier.

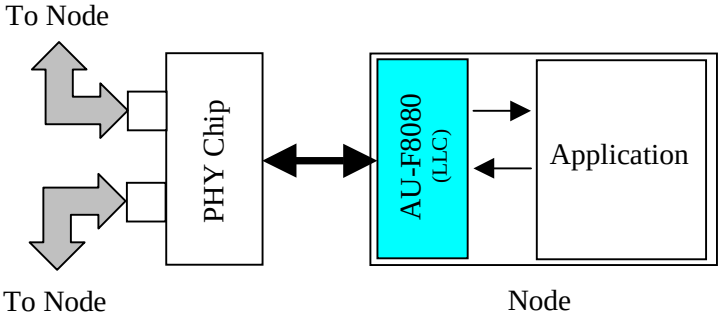
The AU-F8080 can be programmed as the cycle master. It generates cycle start packets when programmed as the master. When not in master mode, the controller monitors the cycle sync events, generates cycle status such as cycle too long, cycle missing, and provides it to the application on a cycle status bus.

The AU-F8080 provides a simple mechanism to access the PHY chip registers. The status of the 1394 bus provided by the PHY chip is maintained by the link core and given to the application on a PHY status bus along with a status qualifier.

Features are summarized:

- Compliant with IEEE 1394b
- AU-F8080 has link layer functionality
- Uses IEEE 1394b-2002 parallel PHY standard to interface to the PHY chip
- Asynchronous and isochronous transfers are supported
- Supports 100, 200, 400, 800, and 1600 mbits/s
- Simple application interface
- Direct FIFO access for both asynchronous and isochronous interface
- Cycle master capability
- Generates CRC for transmit and checks CRC for receive packets
- Separate FIFOs for transmitting asynchronous and isochronous packets

IEEE 1394b is a peer to peer standard. The AU-F8080 provides a simple application interface to which any application logic can interface to the 1394b serial bus as a peer node



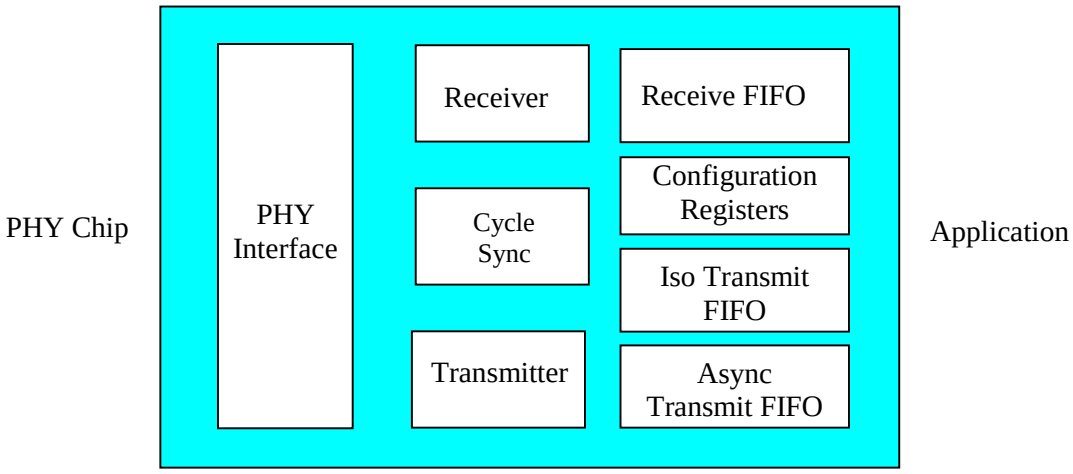
A block diagram of the AU-F8080 is shown below.

The PHY Interface provides PHY level services to the Receiver, Transmitter, and the Cycle Sync blocks. These services include gaining access to the bus, sending data packets at the required speed over the bus, receiving data packets, and sending and receiving acknowledge packets. This interface complies with the parallel PHY interface specification in the IEEE 1394b-2002 standard.

The Receiver takes the incoming data from the PHY Interface and determines if the packet is addressed to this node. If it is, it starts assembling the packet header and packet data. Both the header and the packet data are transferred to Receive FIFO. CRC checks are performed for both the header and data packets. The CRC is not written into the Receive FIFO. The application monitors the Receive FIFO status signal. When data is available in the FIFO, it reads the data. During the configuration phase, the Receiver receives and processes Self-ID packets, and stores them in the Receive FIFO.

The Transmitter interfaces with the PHY Interface on one side and with the application through the Transmit FIFOs, on the other side. The data paths for asynchronous and isochronous transmit data are handled independently using separate FIFOs for each.

The Cycle Sync block has the cycle timer registers and counters. The 1394b LLC Core can be programmed to be the cycle master or a cycle slave. In cycle master mode, it generates cycle start packets. In cycle slave mode, it monitors the received cycle start packets.



The core is delivered as a synthesizeable RTL Verilog model. Deliverables include:

- RTL Verilog source code model of the core
- Verilog testbench and test cases
- Synthesis scripts examples
- Complete detailed documentation and training class notes