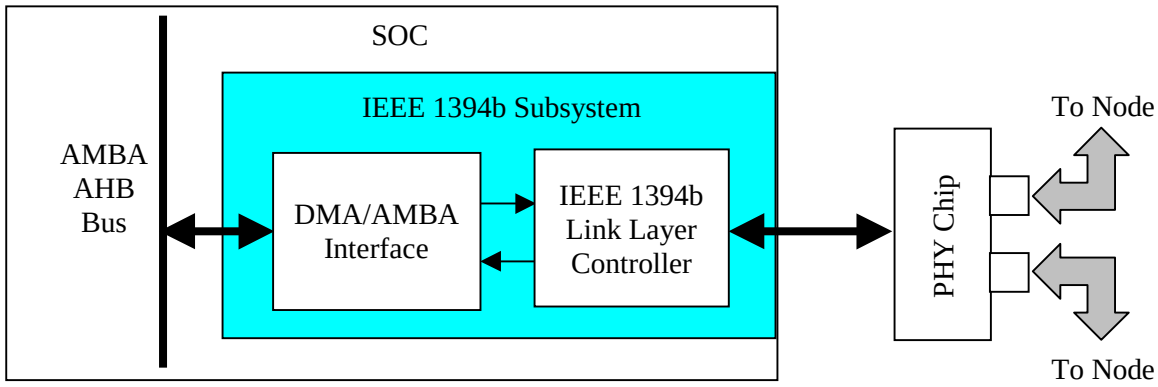
AU-FB8080: IEEE 1394b AMBA Subsystem Core

AMBA AHB Bus IEEE 1394b Link Layer Controller with DMA

The AU-FB8080 IEEE 1394b AMBA Subsystem provides an IEEE 1394b peripheral subsystem for AMBA based SOCs. It contains an IEEE 1394b Link Layer Controller that connects seamlessly to the AMBA AHB Bus. A DMA Engine is included to move packet data. The figure below shows its use within an SOC. The IEEE 1394b AMBA Subsystem Core is available as a synthesizable Verilog model from Aurora VLSI, Inc. Contact CustomerService@auroravlsi.com.



IEEE 1394b is a peer to peer standard. The IEEE 1394b Subsystem is an AMBA Bus peripheral that connects an application to the IEEE 1394b serial bus as a peer node. IEEE 1394b serial bus speeds of 100mb/s, 200mb/s, 400mb/s, 800mb/s, and 1600mb/s are supported. The IEEE 1394b Link Layer Controller block of the IEEE 1394b Subsystem interfaces to an external PHY using the IEEE 1394b-2002 parallel PHY standard.

Internal to an SOC, the IEEE 1394b Subsystem is a peripheral on the AMBA AHB Bus. Direct Memory Access- DMA, between the DMA/AMBA Interface block of the IEEE 1394b Subsystem, and AMBA Bus targets, is used to transfer the transmit and receive data. When doing DMA, the IEEE 1394b Subsystem is an AMBA Bus master. The IEEE 1394b Subsystem is an AMBA Bus slave for configuration, control, and status register accesses. Configuration, control, and status register accesses originate in an AMBA Bus master outside of the IEEE 1394b Subsystem, such as an embedded or host processor.

AMBA is a trademark of ARM Limited

IEEE 1394b AMBA Subsystem features are summarized:

IEEE 1394b Link Layer Controller

- Compliant with IEEE 1394b
- IEEE 1394b link layer functionality
- IEEE 1394b-2002 parallel PHY interface
- Supports 100, 200, 400, 800, and 1600 mbits/s
- Cycle master capability
- Generates CRC for transmit and checks CRC for receive packets
- Asynchronous and isochronous transfers are supported
- Separate transmit data FIFOs for asynchronous and isochronous packets
- Receive data FIFO for incoming packets
- Packet status captured in 2 status FIFOs
  - Transmit Status FIFO- 4 entries
  - Receive Status FIFO- 4 entries
- PHY status and cycle sync status are provided

DMA/AMBA Interface

- AMBA AHB Bus interface
- 3 channel DMA Engine
  - asynchronous transmit data DMA from data source to Async Transmit FIFO
  - isochronous transmit data DMA from data source to Iso Transmit FIFO
  - receive data DMA from Receive FIFO to data destination
- Physical DMA addresses
- Programmable DMA starting address
- Programmable DMA transfer count- up to 64 Kbytes
- Programmable DMA AMBA Bus interface transaction size- 8 to 1024 bytes
- Programmable DMA AMBA Bus data transfer size- 4 or 8 bytes
- Locked DMA operation optional (software programmable)
- Direct software writes or information extracted from descriptors in memory, to program DMA control information
- Dedicated AMBA Bus master interface for each DMA channel
- AMBA Bus slave interface for register reads and writes- configuration, control, and status
- Interrupts:
  - transmit packet DMA completed
  - receive packet DMA completed
  - packet transmit completed
  - packet receive completed

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

AMBA is a trademark of ARM Limited

IEEE 1394b Link Layer Controller

The IEEE 1394b Subsystem includes the AU-F8080 IEEE 1394b Link Layer Controller (LLC) Core. Additional logic at the application interface of the IEEE 1394b LLC provides a DMA Engine, AMBA Bus interface, and host processor interrupts for the IEEE 1394b Subsystem.

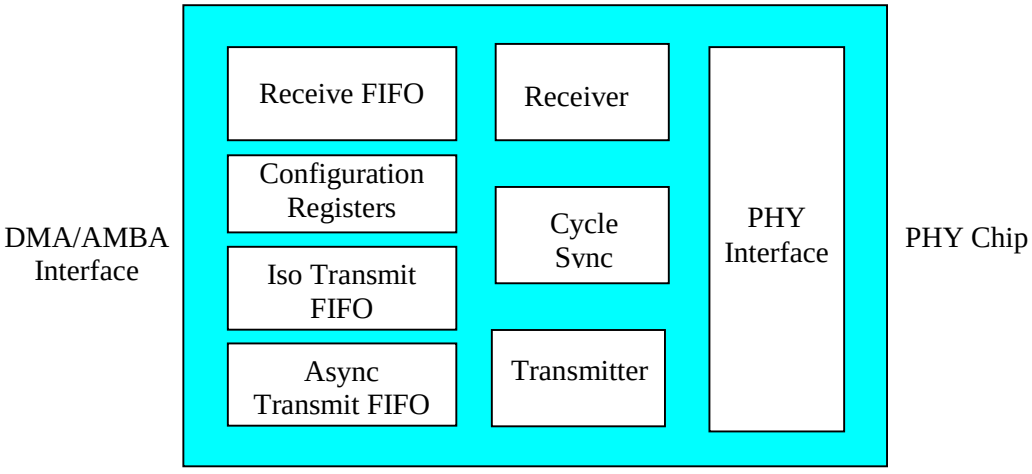
A block diagram of the IEEE 1394b LLC is shown below.

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 the Receive FIFO. CRC checks are performed for both the header and data packets. The CRC is not written into the Receive FIFO. 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 DMA/AMBA Interface 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 IEEE 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.



Transmit and receive status are captured for each transmitted and received packet, respectively. Transmit packet status is captured in a four entry Transmit Status FIFO. The four entry Receive Status FIFO captures receive packet status. The host processor may read these status FIFOs or optionally, the IEEE 1394b Subsystem automatically reports the status by sending it out over the AMBA Bus.

AMBA is a trademark of ARM Limited

DMA/AMBA Interface

The DMA/AMBA Interface includes a three channel DMA Engine. One channel is used to transfer asynchronous transmit data from the data's source, over the AMBA Bus, to the Async Transmit FIFO. The second channel transfers isochronous transmit data to the Iso Transmit FIFO. Receive data is sent from the Receive FIFO, over the AMBA Bus, to the received data's destination, by the third DMA channel.

Block moves of up to 64K bytes are supported by the DMA Engine. The exact transfer count of each DMA operation is the size of the packet that is being transferred, and is set by software. DMA operations are done as a series of Transmit FIFO or Receive FIFO accesses, and bus transactions to move the data block. The length of each bus transaction is software programmable so that it can be optimized according to system characteristics. Each individual bus transaction is from 8 bytes to 1024 bytes according to a value programmed into a DMA channel's control register. Additionally, the data size of each data transfer on the bus is software programmable to be four or eight bytes.

The series of accesses that make up a complete DMA operation may be locked together so that no other device gets the AMBA Bus until the DMA operation is finished. This is under software control.

The DMA starting address is set by software. This is the transmit data source starting address in memory for transmit data, and the receive data destination starting address in memory for receive data. These starting addresses are incremented by the DMA Engine to form the AMBA Bus transaction addresses as the DMA operation progresses. All addresses are physical addresses.

The DMA control information that is set by software- starting address, transfer count, bus transaction size, data transfer size, and lock flag, can be set by direct software writes to IEEE 1394b Subsystem registers that hold this DMA control information. Alternatively, this DMA control information can be set from descriptors in memory that hold the DMA control information. Scatter/gather DMA is done using a chained descriptor list as the DMA control information source. When using descriptors to set the DMA control information, the descriptors are initialized by software. To support DMA configuration from descriptors, the DMA/AMBA Interface contains logic to read the descriptors from memory, and load the appropriate DMA/AMBA Interface registers with the DMA control information.

A DMA operation begins when the DMA channel is enabled, after the starting address, transfer count, bus transaction size, data transfer size, and lock flag are configured. The DMA channel moves the data block, and the DMA operation ends when the entire packet has been moved.

The DMA/AMBA Interface generates interrupts to notify the host processor of events that are important to driver software. These interrupts include:

- Interrupt upon a completed DMA operation between the transmit data source and Transmit FIFO
- Interrupt upon a completed DMA operation between the Receive FIFO and receive data destination
- Interrupt upon completing one or multiple packet transmissions- typically used to check transmit status
- Interrupt upon one or multiple complete packets received and sent out over the AMBA Bus - typically used to start processing the received packet

AMBA is a trademark of ARM Limited