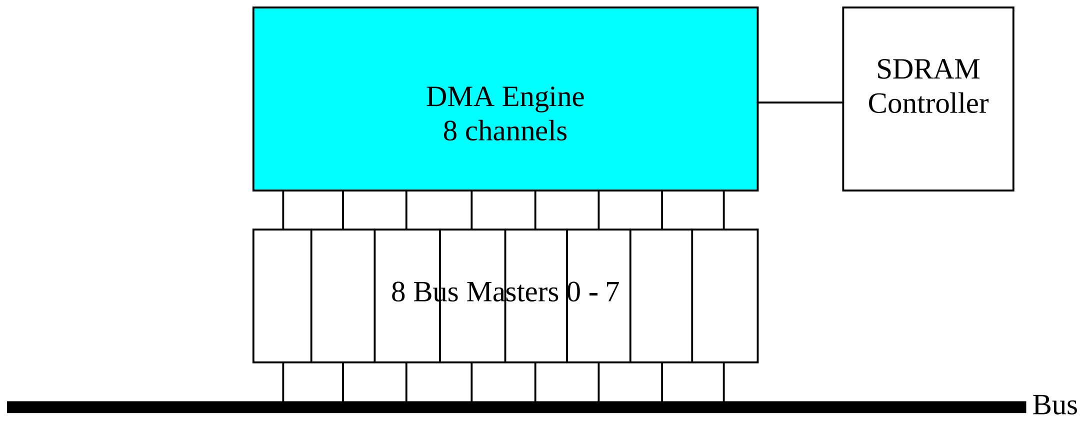
AU-S3000: DMA Engine Core

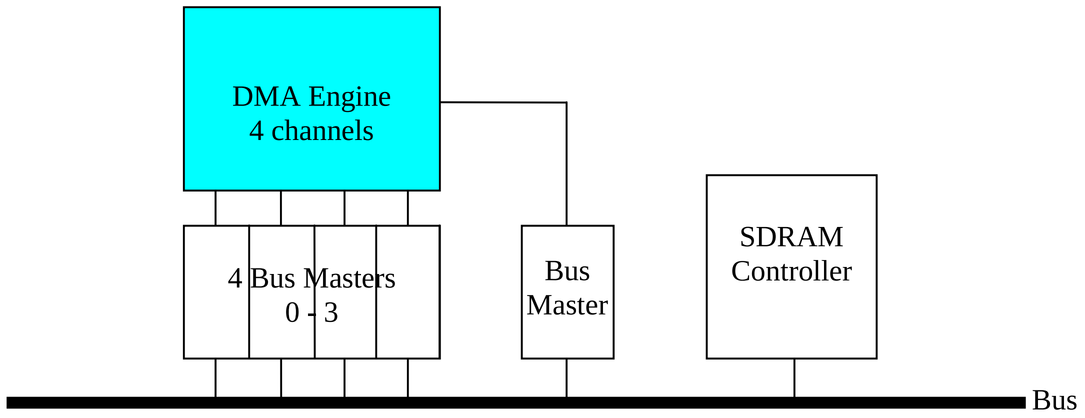
The AU-S3000 DMA Engine Core provides Direct Memory Access (DMA) functionality that moves blocks of data between main memory and peripheral bus devices, and between areas in main memory. Both the main memory interface and the bus interface are generic. The main memory interface will typically connect to a block such as an SDRAM controller. A bus master that supports the chosen system bus will typically connect to the bus interface. Up to eight independent DMA channels are supported. The AU-S3000 DMA Engine Core is available as a synthesizable Verilog model. Contact CustomerService@auroravlsi.com.

The DMA Engine may be used in a variety of system configurations. It is extremely versatile to fit almost any SOC that can benefit from DMA, as shown in the figures below. Some of the possible alternatives are:

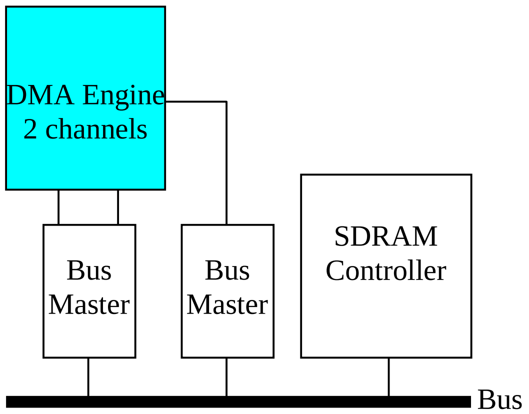
- Bus based RAM controller or bus independent RAM controller
- 1 to 8 DMA channels
- Any system bus
- Multiple RAM requesters or RAM dedicated to the DMA Engine



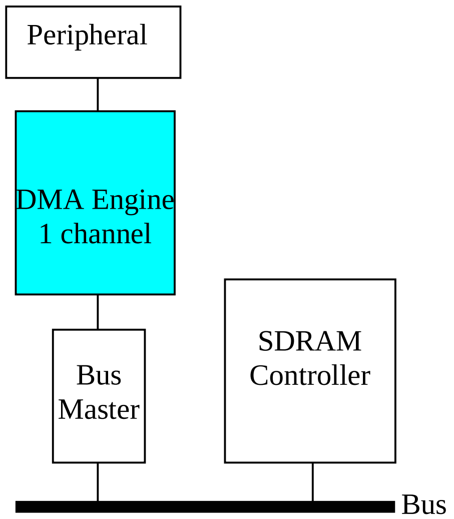
Eight Channels, Dedicated DMA Bus Masters, Isolated RAM Controller



Four Channels, Dedicated DMA Bus Masters, Bus Based RAM Controller



Two Channels, Shared DMA Bus Master,
Bus Based RAM Controller



Bus Based Peripheral DMA Channel

DMA operations include:
- memory to memory block moves
- memory to bus device moves
- bus device to memory moves

Block moves of up to 64K bytes are supported. The exact transfer count of each DMA operation is set by software. DMA operations are done as a series of main memory and bus transactions to move the data block. Main memory accesses are sized to take advantage of higher performance burst accesses to memories such as SDRAMs. 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 32K 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 one, two, 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 memory and bus until the DMA operation is finished. This is under software control.

The source and destination starting addresses are set by software. These addresses are incremented by the DMA Engine to form the request addresses at the memory and bus interfaces of the DMA Engine. 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 DMA Engine 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 Engine contains logic to read the descriptors from memory, and load the appropriate DMA Engine registers with the DMA control information.

A DMA operation begins when software enables a DMA channel, after setting the source and destination starting addresses, transfer count, bus transaction size, and lock feature. The DMA Engine moves the data block, and the DMA operation ends naturally when the number of bytes specified by the transfer count have been moved. A DMA operation may also end early due to an error.

When a DMA operation ends, an interrupt informs the host processor. Each DMA channel has a register that identifies the event that caused the interrupt- a natural DMA operation end or any one of several types of errors.

The DMA Engine memory and bus interfaces have 64 bit data ports for high performance. It is up to the bus master and RAM controller to accept/provide 64 bits at a time, and if necessary break the 64 bit data into smaller data sets to match the bus and RAM data widths.

Each DMA channel has a dedicated bus interface. Therefore, for highest performance, a dedicated bus master is connected to each channel's bus interface. Depending on the characteristics of the bus, a dedicated bus master may be needed for each channel regardless of the desired performance. The DMA Engine has a single interface to memory that is shared by all DMA channels. The DMA Engine uses round robin arbitration to choose the DMA channel that gets each available time slot on the memory interface.

Although the host processor sets the transfer count and initiates the DMA operation, a bus slave device can effectively do this also. A bus slave determines the transfer count by ending a DMA operation early with its error signal. To control the DMA request initiation from the bus slave, once the host has enabled the DMA channel, the bus slave can withhold the data transfer until it wants to start the DMA operation. For this to be possible each channel should have its own bus master so that if one bus master is waiting for a bus slave to respond, other channels may still conduct DMA operations through their dedicated bus masters. Additionally, to attain most efficient bus usage, a bus slave DMA request followed by a DMA channel acknowledge handshake is provided so that each bus burst begins upon the bus slave DMA request that is made when the bus slave can transfer the entire burst without wait states.

DMA Engine features are summarized:

- 1 to 8 channels- user configurable
- DMA between
  - bus devices and RAM memory
  - two memory areas in RAM memory
- Physical DMA
- Programmable source starting address
- Programmable destination starting address
- Programmable transfer count- up to 64 Kbytes
- Programmable bus interface transaction size- 8 to 32 Kbytes
- Programmable bus data transfer size- 1, 2, 4, or 8 bytes
- Memory interface transaction size optimized for RAM burst operations
- Locked DMA operation optional (software programmable)
- Direct software writes or information extracted from descriptors in memory, to program DMA control information
- Scatter/gather DMA using a chained descriptor list in memory as the DMA control information source
- Host processor initiates the DMA operation
- Interrupts signal the end of DMA operations
- Several error types end DMA operations, are recognized and logged
- 64 bit data width at the memory and bus interfaces
- Dedicated bus interface for each DMA channel
- Shared memory interface
- Round robin arbitration for the shared memory interface
- Bus device can optionally determine transfer count and start of the DMA operation
- Request/acknowledge handshake with bus slaves for most efficient bus usage

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