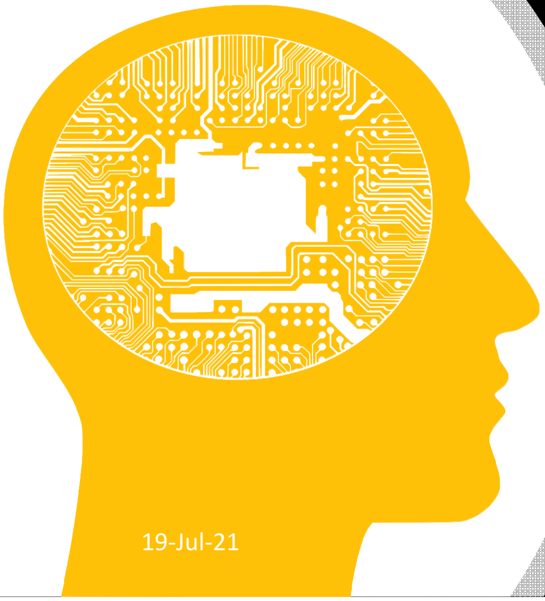


Microprocessor Applications

Direct Memory Access (DMA)

19-Jul-21

© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard

Lecture Outline

The following will be covered in this lecture:

- Motivation for DMA systems
- A general overview and evaluation of DMA systems
- A brief discussion of the DMA and other related systems available within XMEGA AU microcontrollers

© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard

Motivation

- In general, most computing applications require some transfer of data.
  - If a CPU is handling the data transfer, the relevant application may not progress until the data transfer is complete.
- Thus, depending on the amount and frequency of data transferred, data transmission may easily cause a bottleneck on program execution.
- In order to solve this, we often wish to perform data transfer with some system other than a CPU.

© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard

Introducing DMA

- Conventionally, a system that is designed for the purposes of independently transferring data between two sets of memories is referred to as a **D**irect **M**emory **A**ccess **(DMA)** system.
  - In this lecture, Direct Memory Access (DMA) will also refer to the general notion of using a DMA system.

Introducing DMA, cont.

- The main goal for a DMA system is to remove the need for a processing unit to perform data transfer.
  - Other important goals are to minimize the amount of time required for data transfer, maximize the possible amount of data that can be transferred, and provide a generic, configurable interface for performing data transfer.

© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard

Evaluating DMA

- Fortunately, a generic DMA interface can normally be incorporated into a computer system while both greatly reducing the amount of time required to transfer data between memories and increasing the amount of data that can be transferred.
  - Unfortunately, it often turns out that memory components themselves impose a bottleneck on data transfer, and thus on program execution.
  - Separately, incorporating a DMA also generally increases design complexity for both hardware and software, although not necessarily by much.

© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard

Evaluating DMA, cont.

- To effectively utilize a DMA system within some system, the DMA should be synchronized with other relevant components.
  - For example, it would generally be most appropriate to control exactly when some data transfer should begin.
- Thus, a DMA is generally not *entirely* independent from other systems.
  - However, depending on the system, synchronization and control may not always require a processing unit, e.g., some dedicated signal such as an interrupt may suffice for initiating a data transfer.

© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard

Interfacing with a DMA

- For electronic components that provide configuration capabilities, there generally exists a separate hardware **controller** for the purposes of controlling the relevant device.
  - DMA systems generally provide no exception.
- In Atmel XMEGA AU microcontrollers, there exists a robust DMA system that is controlled by another system known as the **D**irect **M**emory **A**ccess **C**ontroller **(DMAC)**.
  - In this context, the DMAC is the system that provides a configurable interface for the DMA.

© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard

The XMEGA AU DMAC

- The DMAC system within XMEGA AU microcontrollers provides a configurable interface for a DMA system with four separate communication channels.
  - Each communication channel within the DMA can act independently of one another and can transfer any amount of data between any two sets of (sequential) locations in the data memory space via the data bus of the microcontroller.
  - Since all peripheral systems within the microcontroller are memory-mapped within the data memory space, each one has the potential to be accessed via DMA.

© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard

The XMEGA AU DMAC, cont.

- Since the DMA communication channels are designed to share the data bus of the microcontroller with the CPU, the CPU may still need to utilize the data bus during a DMA data transfer.
  - In cases where the DMA system has control of the data bus and the CPU needs to transfer data, it is defined that the CPU has priority and will stall the DMA transfer after a specified number of bytes have been transferred.

© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard

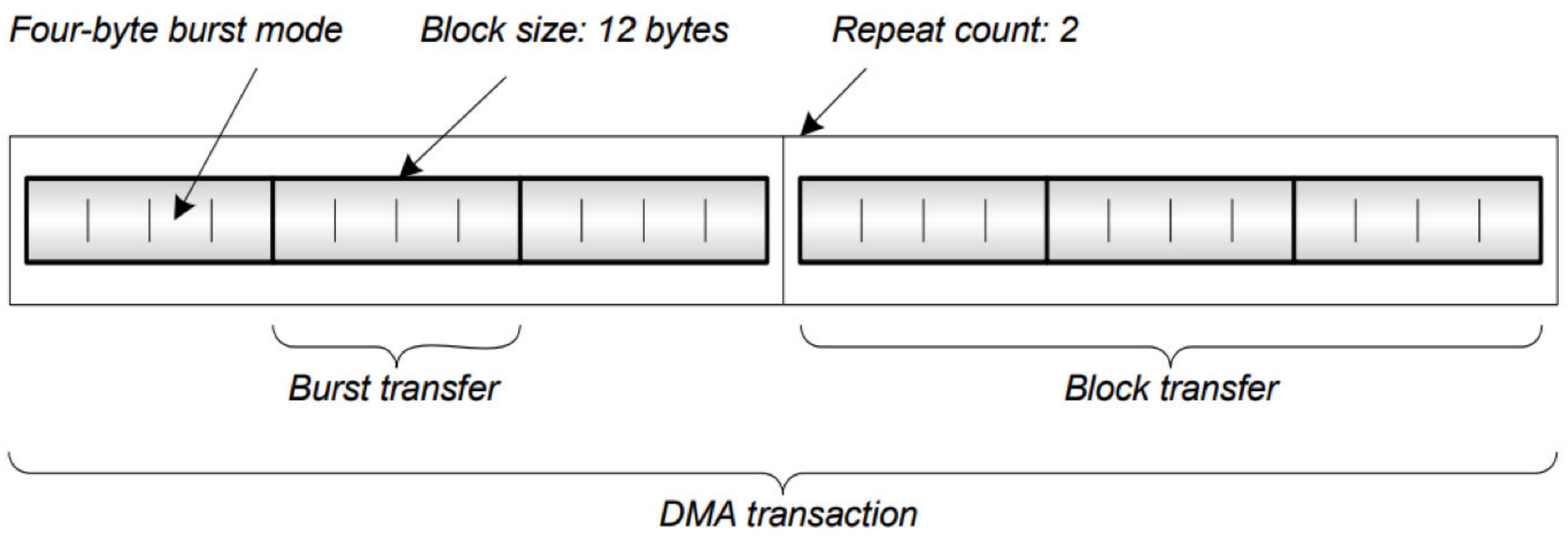
The XMEGA AU DMAC, cont.

- The number of bytes of data that is guaranteed to be transferred via the DMA without CPU interruption is referred to as a **burst**.
  - A burst may be configured to consist of either 1, 2, 4, or 8 bytes.
  - A burst is considered the fundamental unit of a DMA transfer.

The XMEGA AU DMAC, cont.

- The two additional units of a DMA transfer are a **data block** and a **transaction.**
- A data block is an ordered sequence of some number of bytes between 1 and 65535.
- A transaction is an ordered sequence of some number of data blocks between 1 and 255.
  - More generally, a DMA transaction can also be thought of as a complete DMA read and write operation between two sets of memory locations within the data memory space.

The XMEGA AU DMAC, cont.



© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard

The XMEGA AU DMAC, cont.

- The specific address within the data memory space from where data is to be retrieved is referred to as the **source address**.
- The specific address within the data memory space to where data is to be transferred is referred to as the **destination address.**
- Both the source and destination addresses can either increment, decrement, or remain constant during a data transfer. Additionally, they can both be reset to an initial value after either a burst, block, or transaction.

© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard

The XMEGA AU DMAC, cont.

- For each communication channel within the DMA, data transfers can be requested either by the CPU or by a signal available within a set of predefined interrupt and event signals.
  - For example, an overflow condition for a timer/counter module could be used to trigger some data transfer at a periodic rate.

© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard

The XMEGA AU DMAC, cont.

Some other notable features include:

- ❖ Single-shot (i.e., single-burst) data transfer
- ❖ Double buffering between DMA channels
- ❖ Configurable priority schemes between DMA channels

Lecture Review

The following was covered in this lecture:

- A discussion of DMA systems and DMA controllers
- A brief overview of the DMAC and DMA systems available within XMEGA AU microcontrollers

© Dr. Eric M. Schwartz | Christopher Crary | Wesley Piard