DIRECT MEMORY ACCESS

EE3232 DIGITAL SYSTEMS III
CLASS NOTES CHAPTER 11

Department of Electrical Engineering
University of New Brunswick

 January 5, 2000

SUMMARY

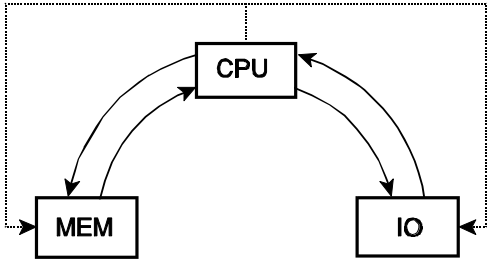
- Objectives.
- The need.
- DMA operation.
- Classifying DMA transfers.
- DMA operation in burst mode.
- DMA interfacing and operation.
- Multi-processors.

OBJECTIVES

- To explain the classes and modes of direct memory access.
- To explain DMA controller handshaking and data transfers.

THE NEED

- The CPU is often a bottleneck for high speed IO. Why ?
  - Information must pass through the CPU before reaching its destination.
  - Loop indexing and control instructions.
  - Interrupt latency ...



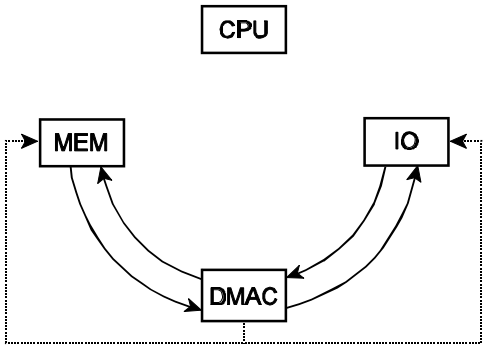
DMA OPERATION

- A direct memory access controller, DMAC, is optimized for
  - IO ← MEMORY transfers,
  - MEMORY ← IO transfers,
  - MEMORY ← MEMORY transfers and,
  - IO ← IO transfers.
- The CPU initializes a DMA Controller for a transfer.
- The DMAC requests control of the system bus from the CPU.
- The CPU disconnects itself from the system bus.
- The DMAC generates the address and control signals for the data transfer.
- The DMAC relinquishes control of the bus.
- The CPU resumes normal operation.
- The CPU must test for ***or*** be notified when the transfer completes.

CLASSIFYING DMA TRANSFERS

- DMA transfers may be classified as:
  1. ***Sequential*** : Data from MEMORY or IO is first read into the DMAC, then the data is written to IO or MEMORY.

CLASSIFYING DMA TRANSFERS (CONT'D)



2. ***Simultaneous*** : Data is read from MEMORY or IO and simultaneously written to IO or MEMORY.

- Within each class of DMA there are various modes of operation:

1. ***Burst*** : A block of characters is transferred. After the block is transferred control is relinquished to the CPU.

2. ***Cycle Steal*** : A single character is transferred whenever the CPU is not executing a R/W bus cycle. Control is then relinquished to the CPU. This is repeated until all characters have been transferred.

3. ***Demand Transfer*** : Allows the IO device to briefly suspend a DMA transfer so it can "catch up".

DMAC OPERATION IN BURST MODE

Assume IO ← MEMORY ***or*** MEMORY ← IO transfers.

1. The CPU initializes the DMAC specifying the mode, starting address, count, etc ... (CPU is in ***master mode***, DMAC is in ***slave mode***).

2. An IO device (***with DMA capability*** and previously initialized) activates a DMA request, DREQi.

3. The DMAC responds by activating the CPU HOLD signal.

4. The CPU finishes executing the current bus cycle and then puts all system bus signals into hi-impedance state.

5. The CPU acknowledges the DMAC by activating HLDA.

6. The DMAC is now in ***master mode*** and the CPU is (electrically) disconnected from the system bus.

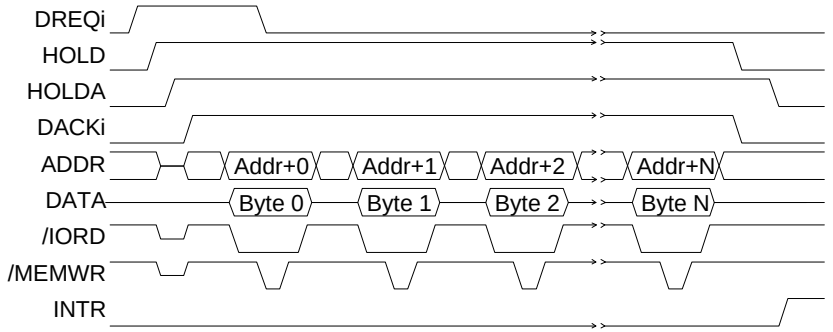
7. The DMAC selects the IO device by activating /DACKi.

8. The DMAC now has control of the system bus and operates with ***simultaneous transfers*** as follows :

   while (COUNT != 0) {
   Put first / next address on address bus ;
   Assert /MEMRD or /DACKi ***and*** /DACKi or /MEMWR ;
   Increment address pointer ;
   Decrement COUNT ;
   }

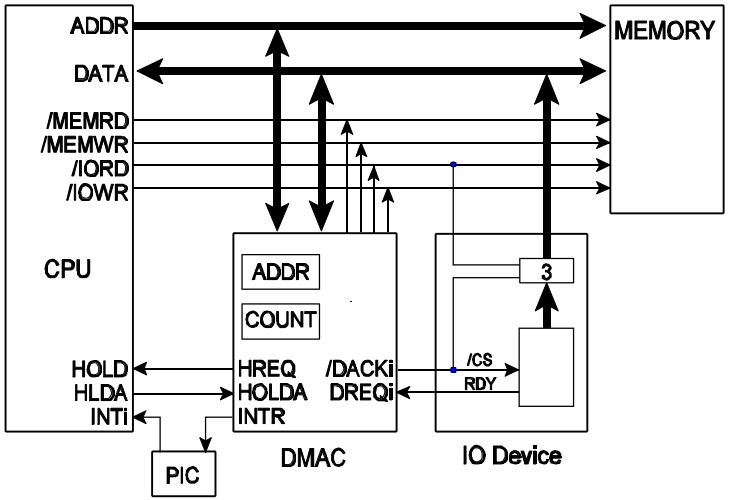
9. The DMAC deactivates /DACKi and HREQ relinquishing control of the bus to the CPU and the CPU resumes normal operation.

10. The DMAC then generates an interrupt to signal the CPU that the DMA transfer has now finished, (or the CPU polls a DMAC status bit).



DMA - Burst Simultaneous Transfer.

DMAC INTERFACING AND OPERATION (AS MASTER)



MULTI-PROCESSORS

- One can think of a DMAC as a ***processor*** dedicated for hi-speed IO.
- Other (general purpose) processors may also be interfaced to a system bus (like a DMAC) to increase computing power.
- These are called ***multi-processor systems***.