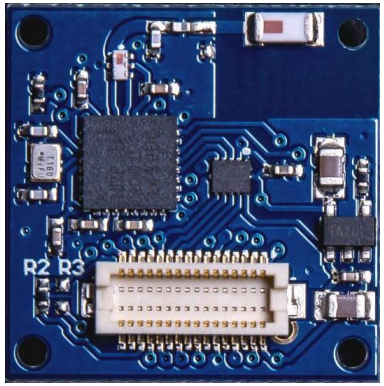
EE107 Spring 2019

Lecture 5

Direct Memory Access

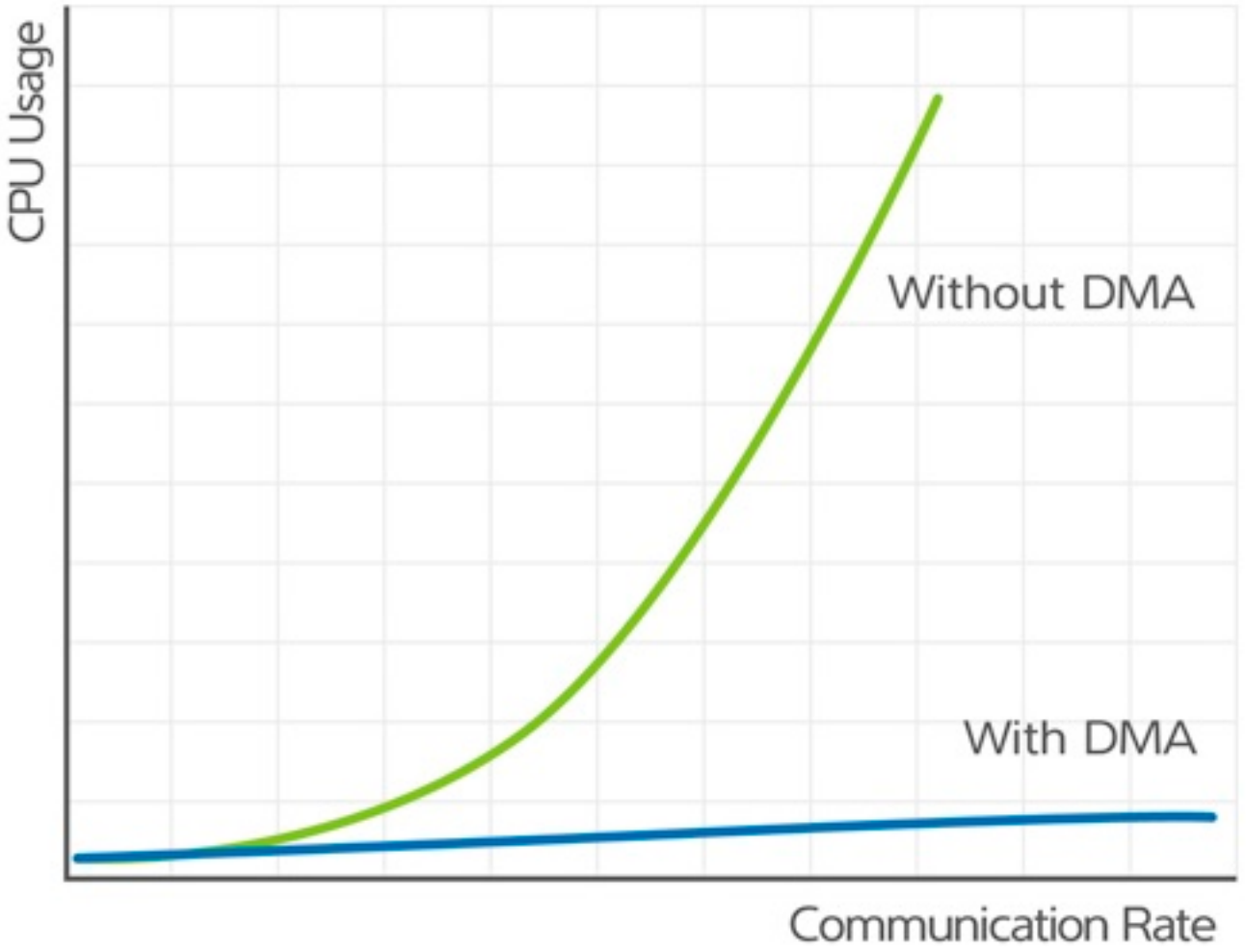


Embedded Networked Systems

Sachin Katti

*slides adapted from Aaron Schulman's CSE190

Why do we need DMA?



Why do we need DMA?

- Polling and Interrupt driven I/O concentrates on data transfer between the processor and I/O devices.

- Processor determines that the I/O device is ready
  - Either by polling a status flag in the device interface or
  - Waits for the device to send an interrupt request.

- Considerable overhead is incurred, because several program instructions must be executed for each data word transferred.

Why do we need DMA?

- Instructions are needed to increment memory address and keeping track of work count.

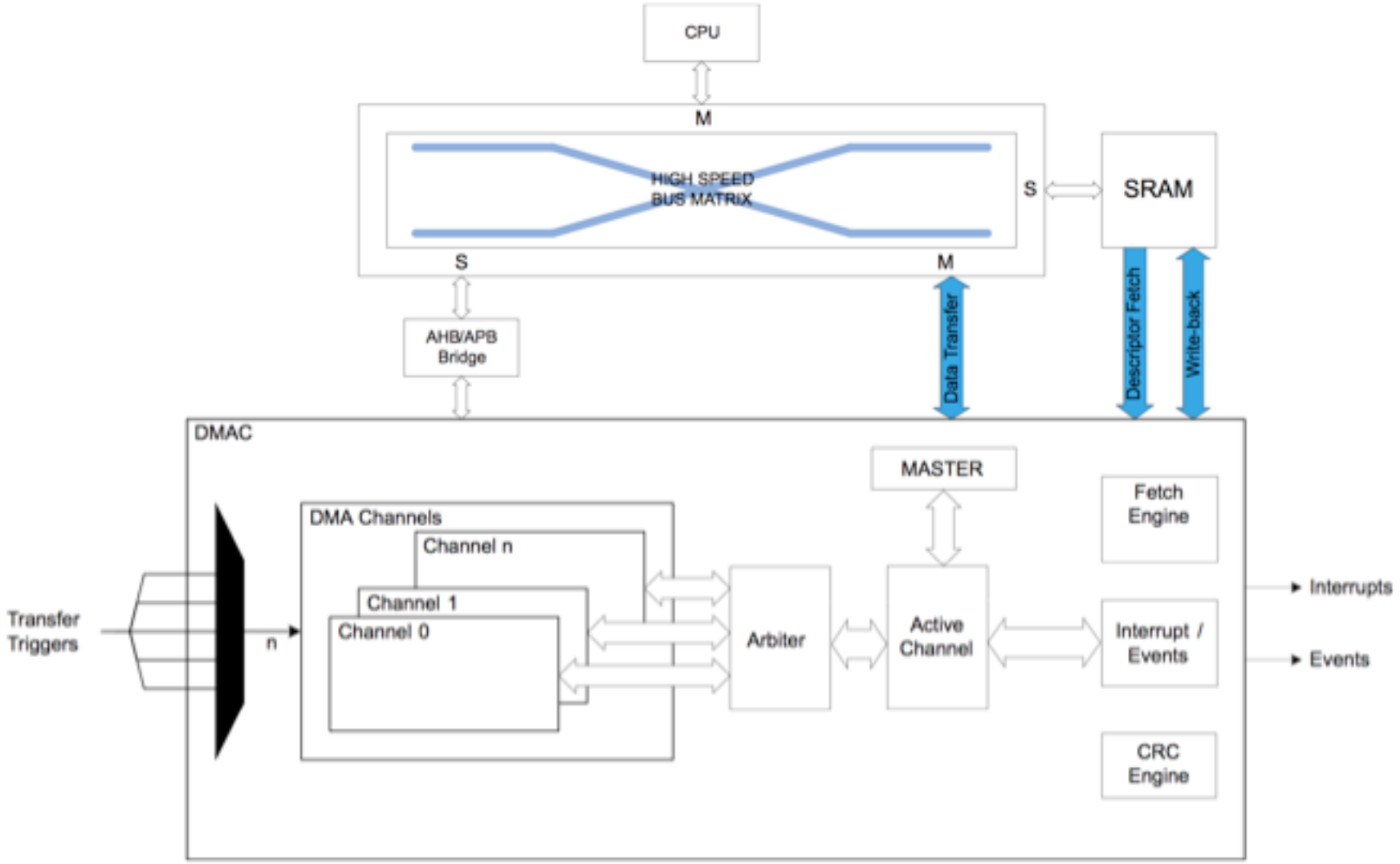
- With interrupts, additional overhead associated with saving and restoring the program counter and other state information.

Direct Memory Access (DMA)

- To transfer large blocks of data at high speed, an alternative approach is used.

- Blocks of data are transferred between an external device and the main memory, without continuous intervention by the processor.

DMA Controllers



Arduino DMA with demo

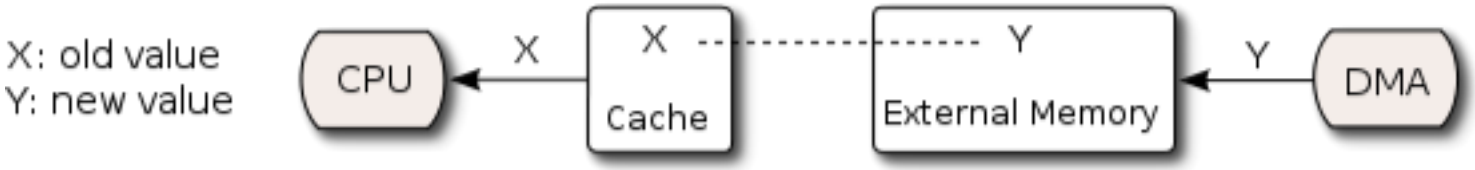
- Data Transfer From:
  - Peripheral-to-peripheral
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
- Transfer Trigger Sources:
  - Software
  - Events from Event System
  - Dedicated requests from peripherals
- SRAM-based Transfer Descriptors:
  - Single transfer using one descriptor
  - Multi-buffer or Circular Buffer modes by linking multiple descriptors

Arduino DMA with demo

- Up to 12 Channels:
  - Enable 12 independent transfers
  - Automatic descriptor fetch for each channel
  - Suspend/resume operation support for each channel
- Flexible Arbitration Scheme:
  - 4 configurable priority levels for each channel
  - Fixed or round-robin priority scheme within each priority level
- From 1 to 256KB Data Transfer in a Single Block Transfer
- Multiple Addressing Modes:
  - Static
  - Configurable increment scheme
- Optional Interrupt Generation:
  - On block transfer complete
  - On error detection
  - On channel suspend

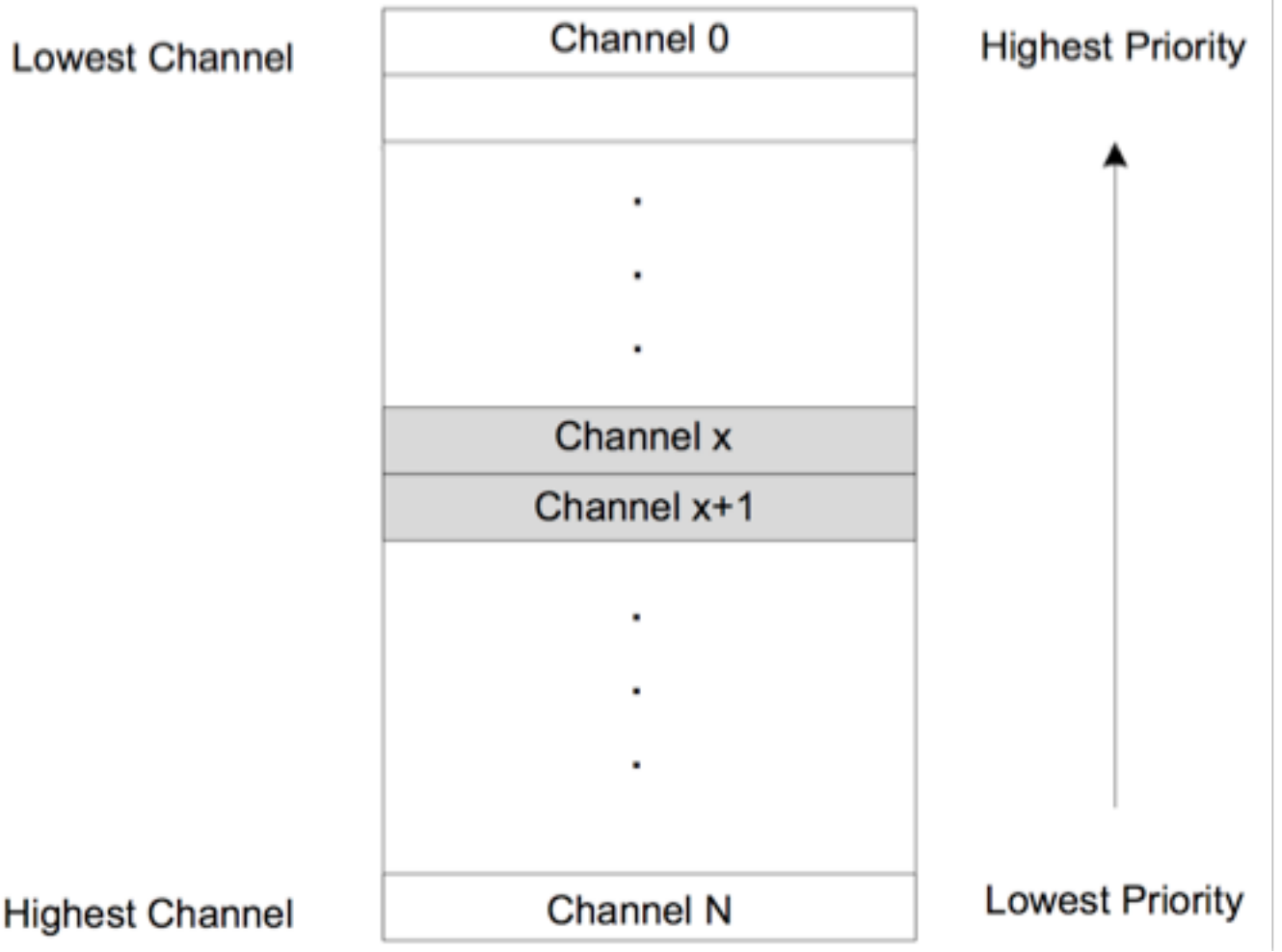
Cache coherency problems

- Imagine a CPU equipped with a cache and an external memory that can be accessed directly by devices using DMA. When the CPU accesses location X in the memory, the current value will be stored in the cache. Subsequent operations on X will update the cached copy of X, but not the external memory version of X, assuming a write-back cache. If the cache is not flushed to the memory before the next time a device tries to access X, the device will receive a stale value of X.



Arbitration

Figure 20-5. Static Priority Scheduling



Arbitration

Figure 20-6. Dynamic (Round-Robin) Priority Scheduling