Input / Output Organization

Overview

- Computer has ability to exchange data with other devices.
  - Human-computer communication
  - Computer-computer communication
  - Computer-device communication

Accessing I/O Devices

Single Bus

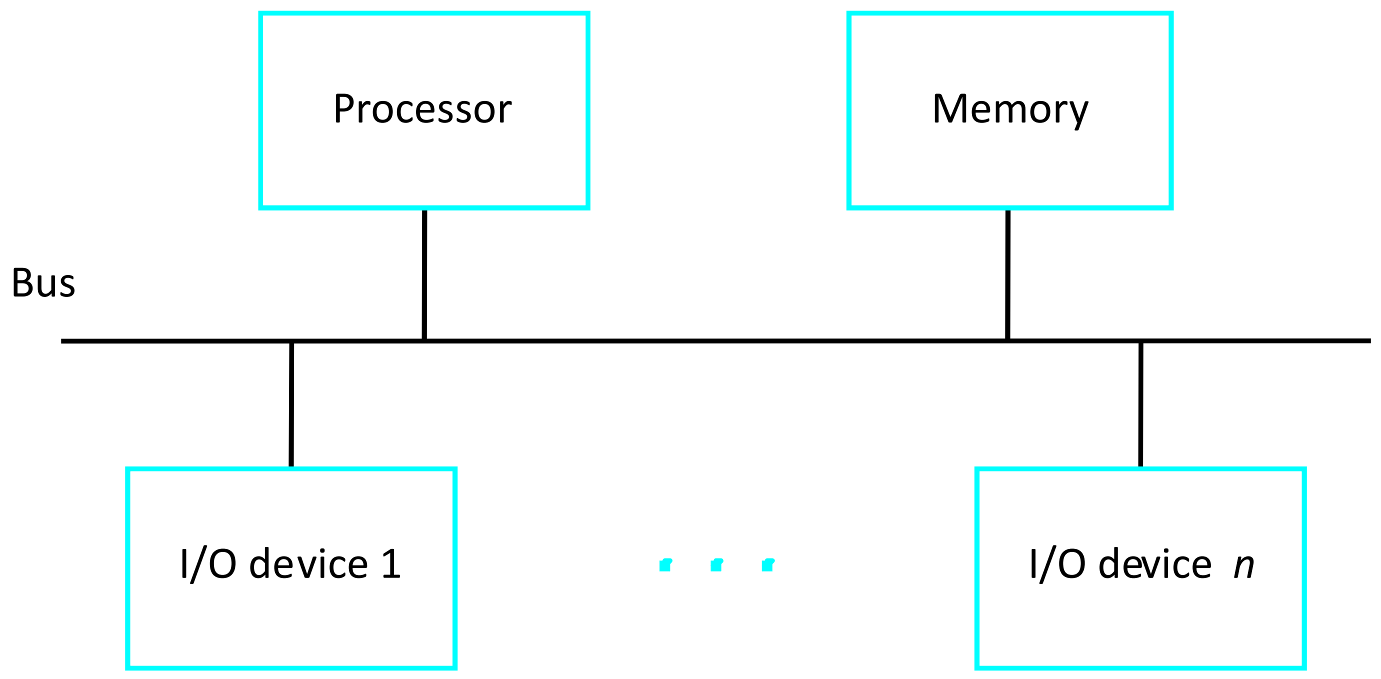


Figure 4.1. A single-bus structure.

Memory-Mapped I/O

- When I/O devices and the memory share the same address space, the arrangement is called memory-mapped I/O.
- Any machine instruction that can access memory can be used to transfer data to or from an I/O device.

```
Move  DATAIN, R0
Move  R0, DATAOUT
```

- Some processors have special In and Out instructions to perform I/O transfer.

ISOLATED vs MEMORY MAPPED I/O

<u>Isolated I/O</u>

- Separate I/O read/write control lines in addition to memory read/write control lines
- Separate (isolated) memory and I/O address spaces
- Distinct input and output instructions

<u>Memory-mapped I/O</u>

- A single set of read/write control lines
  (no distinction between memory and I/O transfer)
- Memory and I/O addresses share the common address space
  -> reduces memory address range available
- No specific input or output instruction
  -> The same memory reference instructions can be used for I/O transfers
- Considerable flexibility in handling I/O operations

Interface

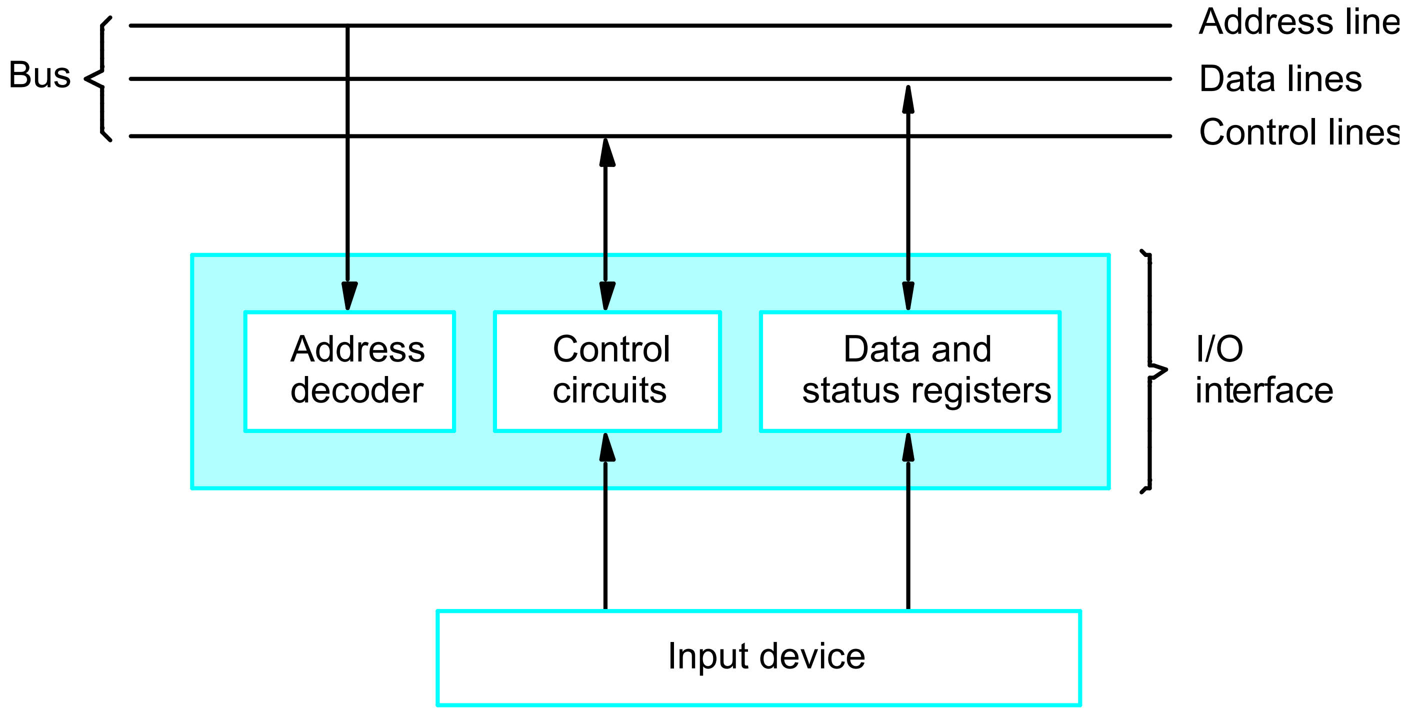
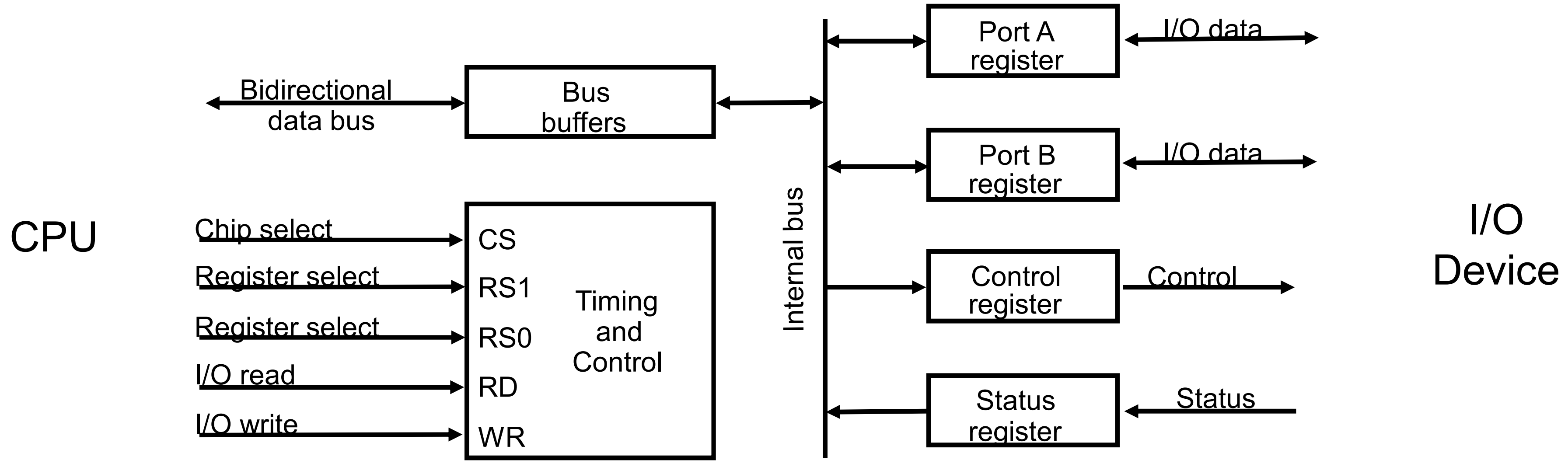


Figure 4.2. I/O interface for an input device.

I/O INTERFACE



CS	RS1	RS0	Register selected
0	x	x	None - data bus in high-impedence
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

Programmable Interface

- Information in each port can be assigned a meaning depending on the mode of operation of the I/O device
  -> Port A = Data; Port B = Command; Port C = Control; Port D = Status

- CPU initializes(loads) each port by transferring a byte to the Control Register
  -> Allows CPU can define the mode of operation of each port
  -> *Programmable Port*: By changing the bits in the control register, it is possible to change the interface characteristics

ASYNCHRONOUS DATA TRANSFER

Synchronous and Asynchronous Operations

Synchronous - All devices derive the timing information from common clock line
Asynchronous - No common clock

Asynchronous Data Transfer

Asynchronous data transfer between two independent units requires that *control signals* be transmitted between the communicating units *to indicate the time at which data is being transmitted*

Two Asynchronous Data Transfer Methods

Strobe pulse

- A strobe pulse is supplied by one unit to indicate the other unit when the transfer has to occur

Handshaking

- A control signal is accompanied with each data being transmitted to indicate the presence of data
- The receiving unit responds with another control signal to acknowledge receipt of the data

HANDSHAKING

Source-Initiated

The source unit that initiates the transfer has no way of knowing whether the destination unit has actually received data

Destination-Initiated

The destination unit that initiates the transfer no way of knowing whether the source has actually placed the data on the bus

To solve this problem, the *HANDSHAKE* method introduces a second control signal to provide a *Reply* to the unit that initiates the transfer

Three Major Mechanisms

- Program-controlled I/O – processor polls the device.
- Interrupt
- Direct Memory Access (DMA)

Program-Controlled I/O

- I/O devices operate at speeds that are very much different from that of the processor.
- Keyboard, for example, is slower in speed than processor.
- In this method, the processor constantly checks the status flags, and when it finds that the flag is set it performs the appropriate operation.

Interrupts

Overview

- In program-controlled I/O, the program enters a wait loop in which it repeatedly tests the device status. During the period, the processor is not performing any useful computation.

- However, in many situations other tasks can be performed while waiting for an I/O device to become ready.

- Let the device alert the processor.

Enabling and Disabling Interrupts

- Since the interrupt request can come at any time, it may alter the sequence of events from that predicted by the programmer.
- Interrupts must be controlled.

Enabling and Disabling Interrupts

- The interrupt request signal will be active until it learns that the processor has responded to its request. This must be handled to avoid successive interruptions.

➢Let the interrupt be disabled/enabled in the interrupt-service routine.

➢Let the processor automatically disable interrupts before starting the execution of the interrupt-service routine.

Vectored Interrupts

- A device requesting an interrupt can identify itself by sending a special code to the processor over the bus.
- Interrupt vector
- Avoid bus collision

Interrupt Nesting

- Simple solution: only accept one interrupt at a time, then disable all others.
- Problem: some interrupts cannot be held too long.
- Priority structure

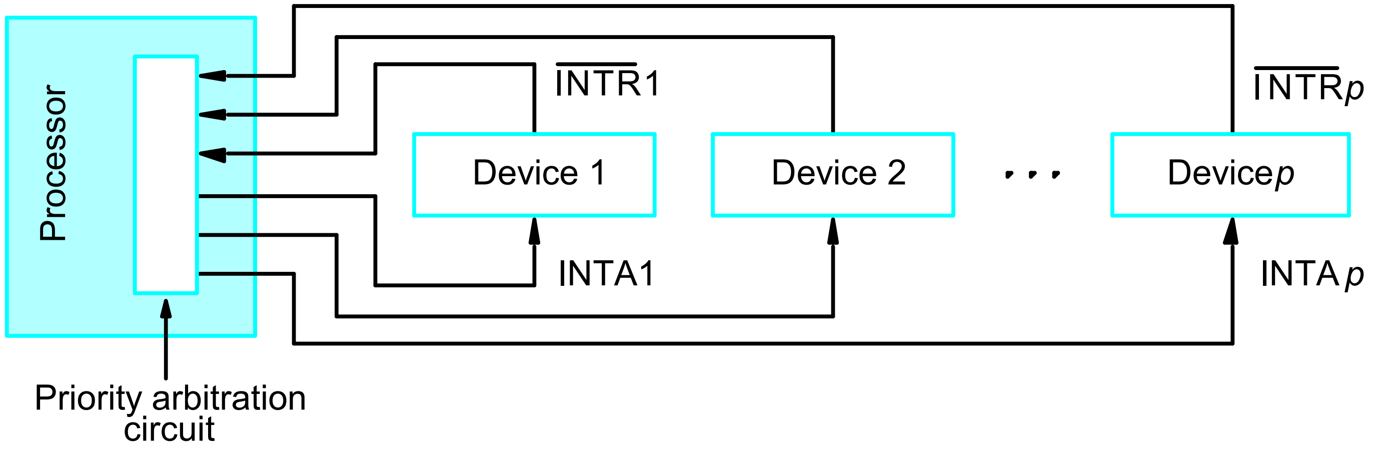
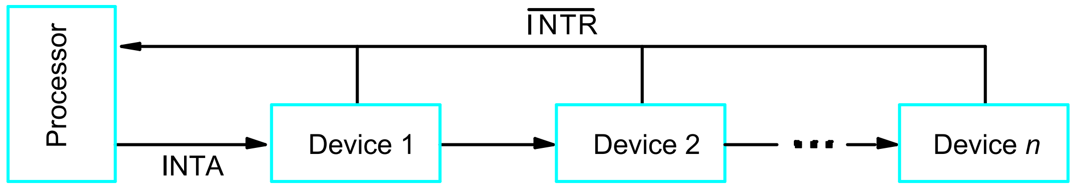
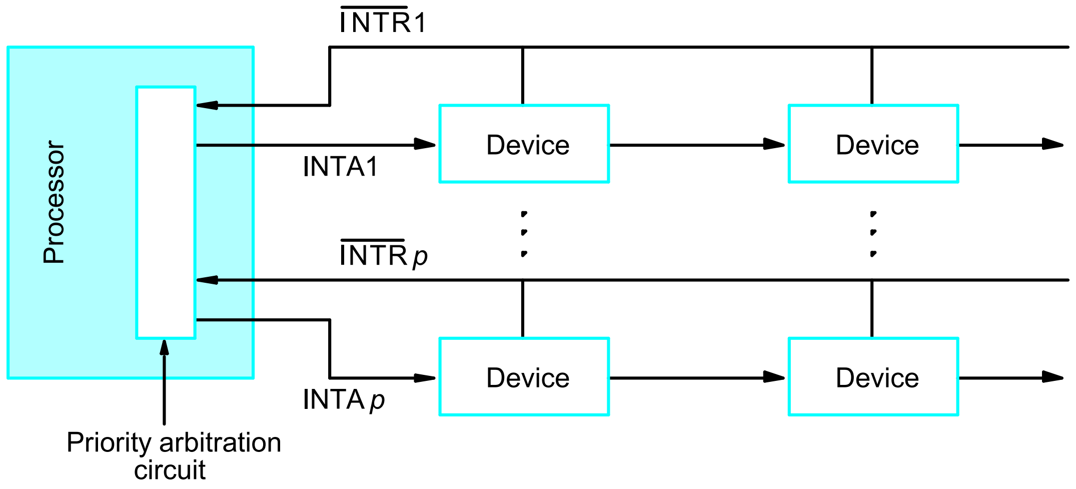

Processor
$\overline{\text{INTR}}1$
$\overline{\text{INTR}}p$
Device 1
Device 2
Device p
INTA1
INTA p
Priority arbitration circuit

Figure 4.7. Implementation of interrupt priority using individual interrupt-request and acknowledge lines.

Simultaneous Requests



(a) Daisy chain



$\overline{\text{INTR}}$1
Processor
INTA1
Device
Device
$\overline{\text{INTR}}\,p$
INTA *p*
Device
Device
Priority arbitration circuit

(b) Arrangement of priority groups

Figure 4.8. Interrupt priority schemes.

Controlling Device Requests

- Some I/O devices may not be allowed to issue interrupt requests to the processor.
- At device end, an interrupt-enable bit in a control register determines whether the device is allowed to generate an interrupt request.
- At processor end, either an interrupt enable bit in the PS register or a priority structure determines whether a given interrupt request will be accepted.

Exceptions

- Recovery from errors
- Debugging
  - Trace
  - Breakpoint
- Privilege exception

Use of Interrupts in Operating Systems

- The OS and the application program pass control back and forth using software interrupts.
- Supervisor mode / user mode
- Multitasking (time-slicing)
- Process – running, runnable, blocked
- Program state

Processor Examples

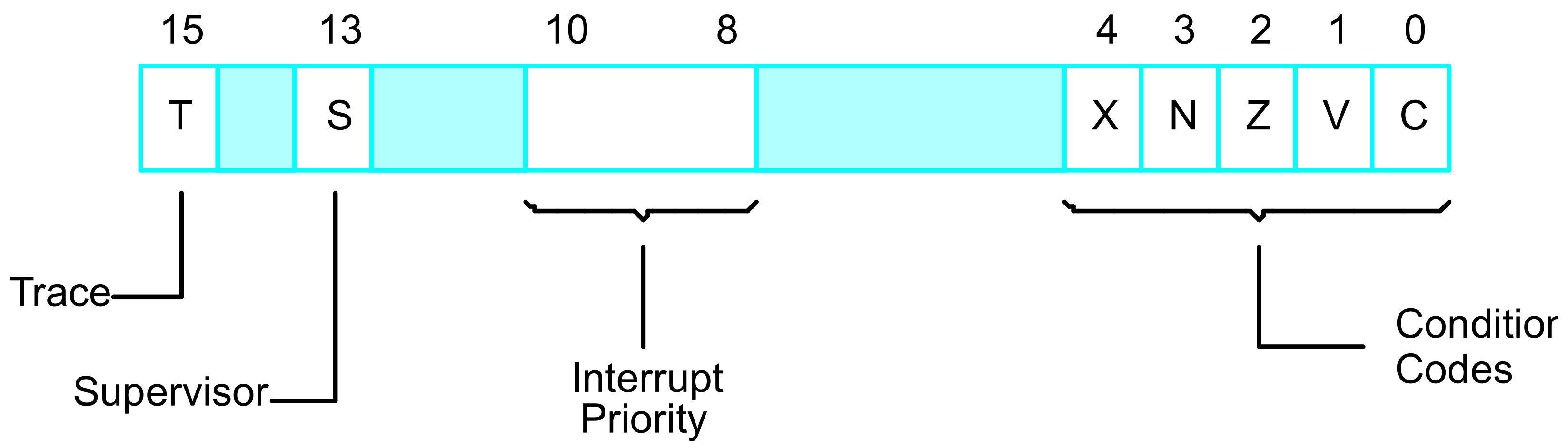


Figure 4.14. Processor statusgister in the 68000 processor

Direct Memory Access

DMA

- A special control unit may be provided to allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor – direct memory access (DMA).
- The DMA controller provides the memory address and all the bus signals needed for data transfer, increment the memory address for successive words, and keep track of the number of transfers.

DMA Procedure

- Processor sends the starting address, the number of data, and the direction of transfer to DMA controller.
- Processor suspends the application program requesting DMA, starts DMA transfer, and starts another program.
- After the DMA transfer is done, DMA controller sends an interrupt signal to the processor.
- The processor puts the suspended program in the ‘Run’ state.

DMA Register

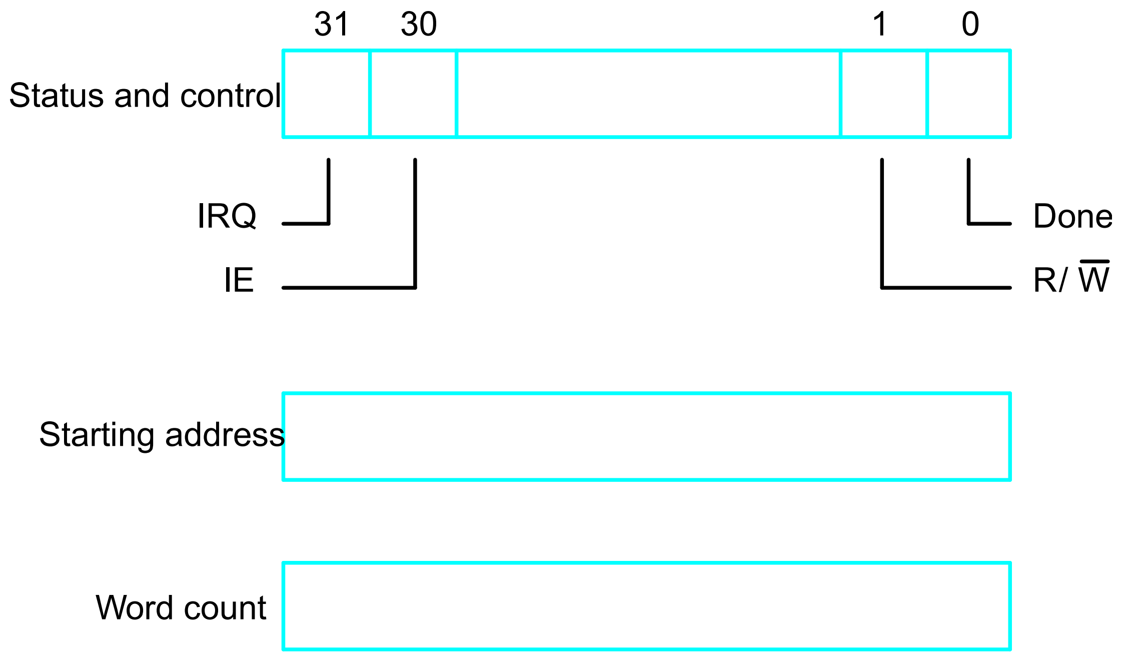


Figure 4.18. Registers in a DMA interface.

System

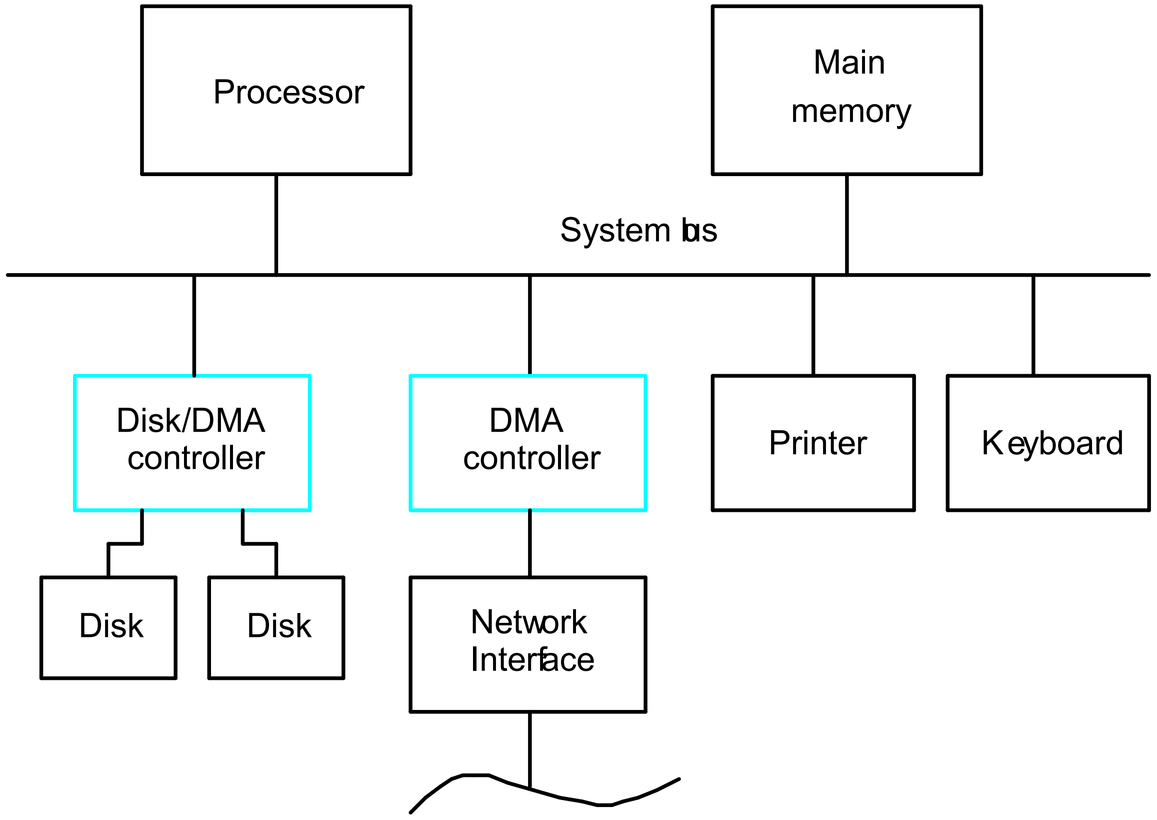


Figure 4.19. Use of DMA controllers in a computer system.

Memory Access

- Memory access by the processor and the DMA controller are interlinked.
- DMA device has higher priority.
- Among all DMA requests, top priority is given to high-speed peripherals.

Bus Arbitration

- The device that is allowed to initiate data transfers on the bus at any given time is called the bus master.
- Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it.
- Need to establish a priority system.
- Two approaches: centralized and distributed

Centralized Arbitration

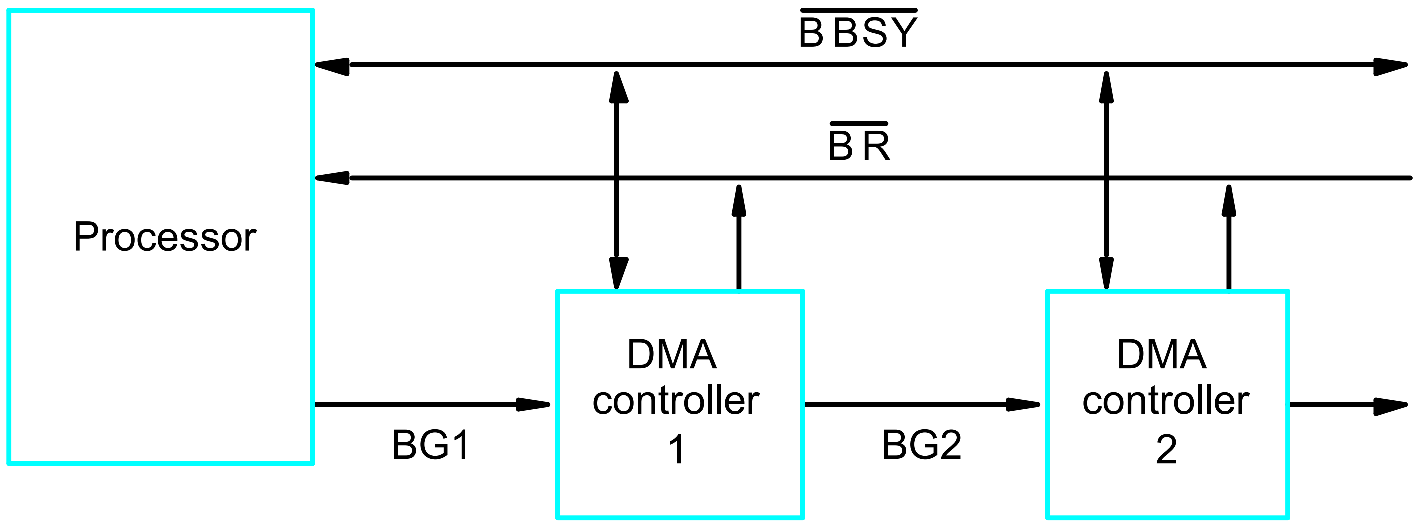


Figure 4.20. A simple arrangement for bus arbitration using a daisy chain

Distributed Arbitration

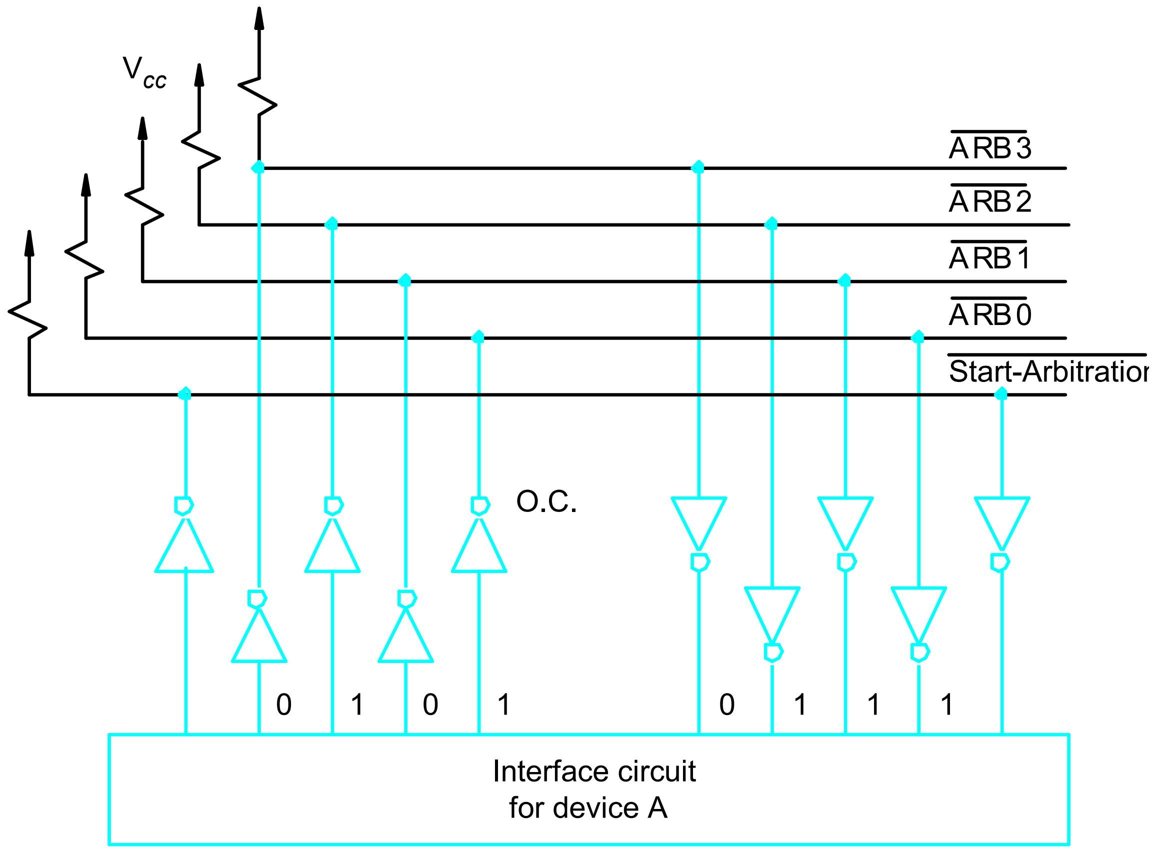


Figure 4.22. A distributed arbitration scheme.

Bus

- The primary function of a bus is to provide a communications path for the transfer of data.
- A bus protocol is the set of rules that govern the behavior of various devices connected to the bus as to when to place information on the bus, assert control signals, etc.
- Three types of bus lines: data, address, control
- The bus control signals also carry timing information.
- Bus master (initiator) / slave (target)