Digital System Design

by
Dr. Lesley Shannon
Email: lshannon@ensc.sfu.ca
Course Website: http://www.ensc.sfu.ca/~lshannon/courses/ensc350

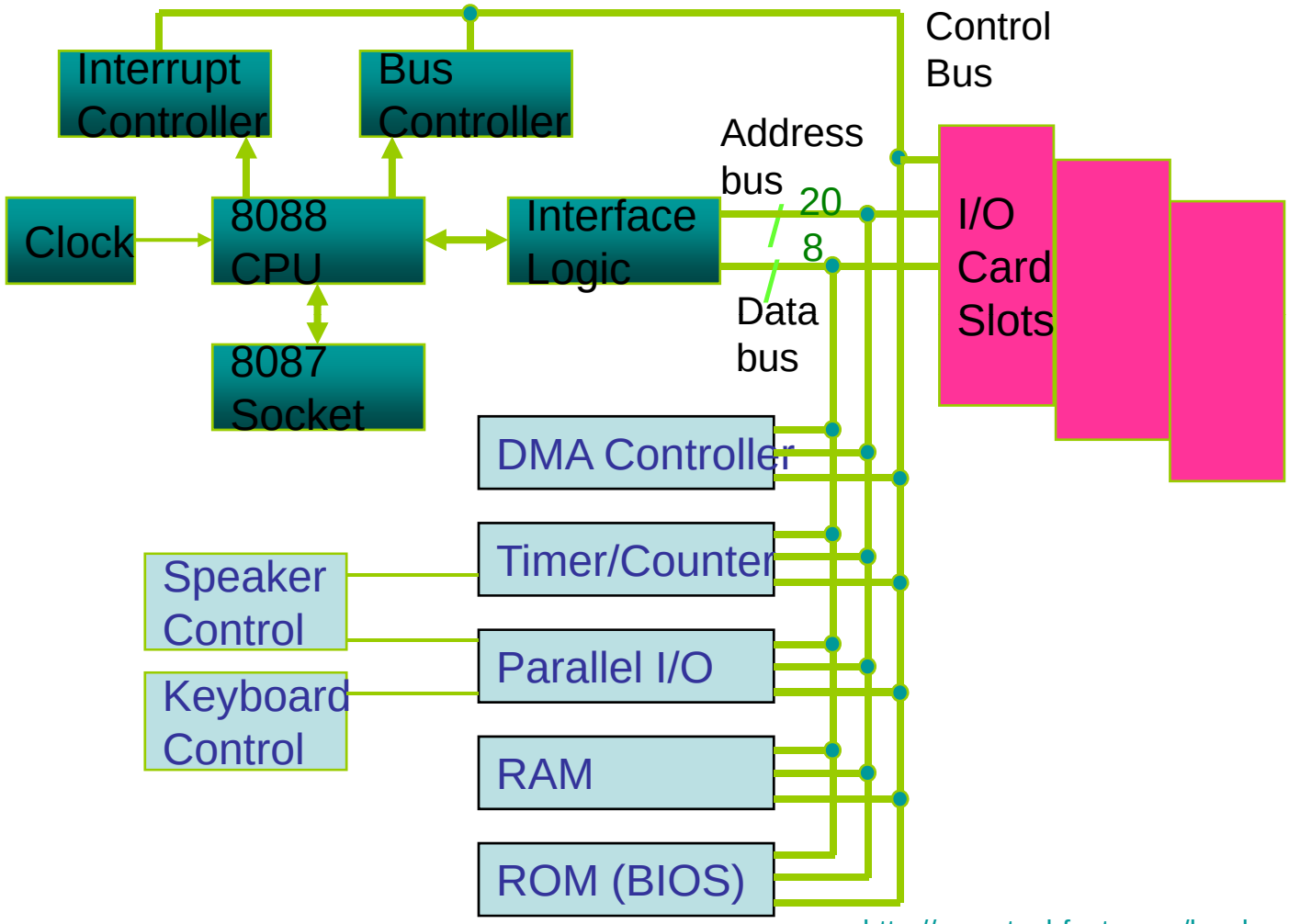
Simon Fraser University

Slide Set: 15
Date: March 30, 2009

Slide Set Overview

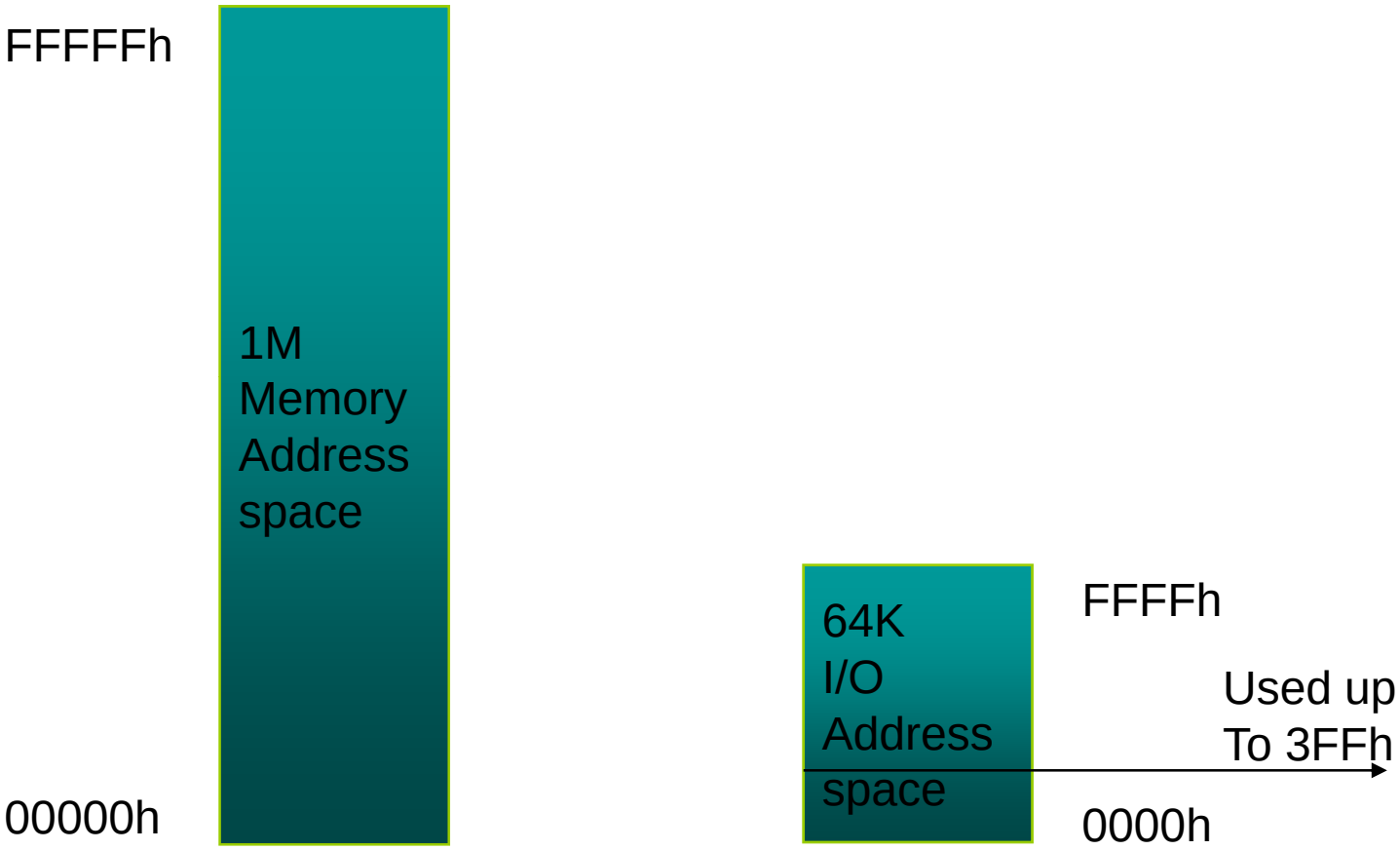
- Buses
  - Most computing systems in service today use bus based architectures
  - Therefore, it's important to understand how they work
  - If you design a core, the two likely interfaces are a bus or a FIFO
    - We've covered FIFOs and you've seen an example Bus protocol for your lab, but we're going to look another simple bus protocol here

IBM PC/XT Block Diagram



http://www.techfest .com/hardware/bus/isa.htm

8088 and 8086 memory and I/O address space




```
mov     AX, $5555
```

```
in      AL, $300
out     AL, $300
```

Example data move instruction:

TABLE 5-1
PC/XT Memory Map

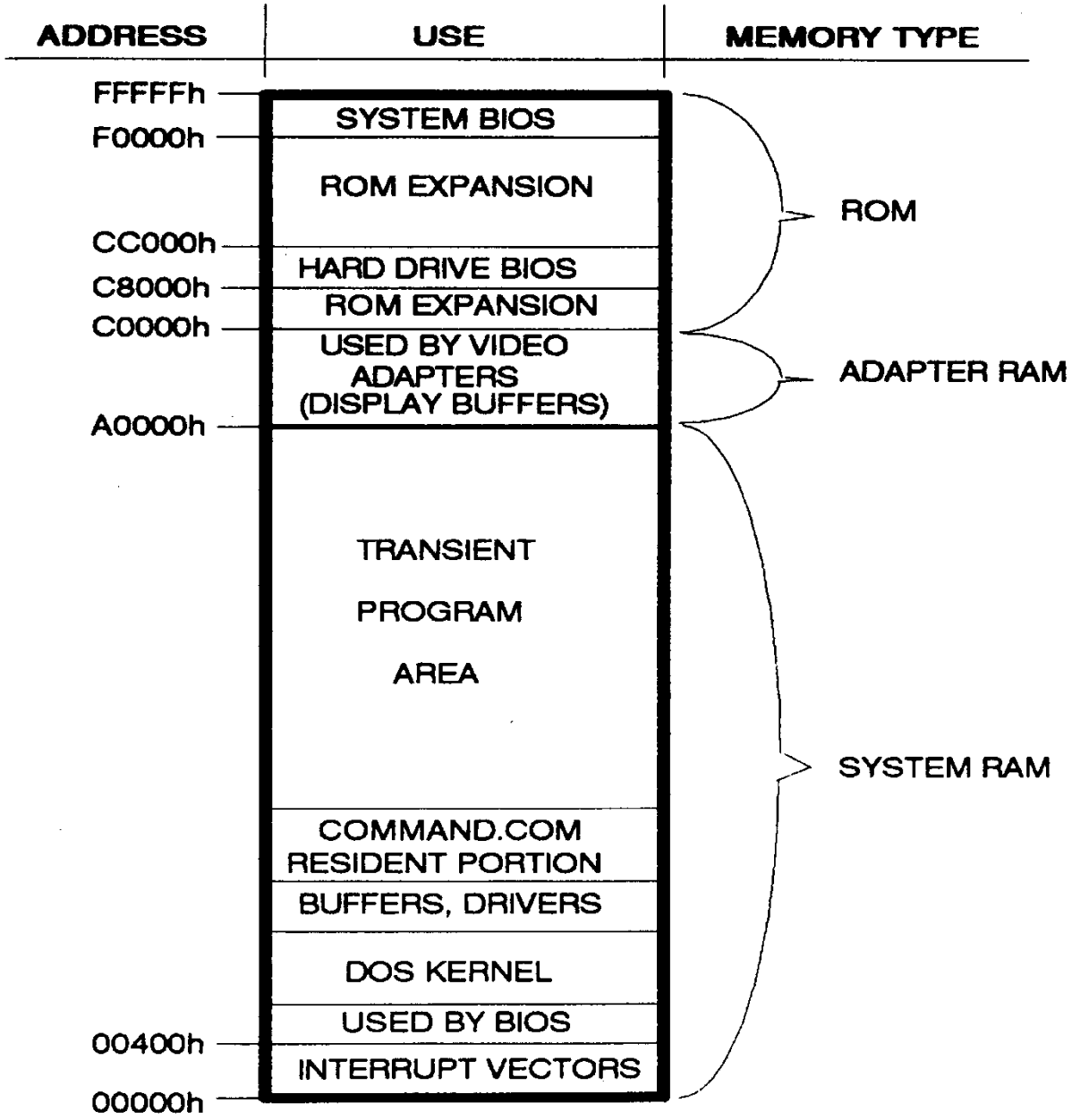


TABLE 5-2

PC/XT I/O Address Map

I/O ADDRESS	USE	
000 - 00Fh	DMA CONTROLLER	ON MOTHERBOARD
020 - 021h	INTERRUPT CONTROLLER	
040 - 043h	TIMER	
060 - 063h	PPI (8255)	
080 - 083h	DMA PAGE REGISTERS	
0A0	NMI MASK REGISTER	
200 - 20Fh	GAME ADAPTER	ON ADAPTER CARDS
210 - 217h	EXPANSION UNIT	
2F8 - 2FFh	ASYNCH ADAPTER (COM2:)	
300 - 31Fh	PROTOTYPE CARD	
320 - 32Fh	HARD DISK DRIVE ADAPTER	
378 - 37Fh	PRINTER ADAPTER	
380 -38Ch	SDLC COMM ADAPTER	
390 -393h	CLUSTER ADAPTER	
3A0 - 3A9h	BISYNC ADAPTER	
3B0 - 3BFh	MONO DISPLAY/PRINTER ADAPTER	
3D0 - 3DFh	CGA ADAPTER	
3F0 - 3F7h	DISKETTE DRIVE ADAPTER	
3F8 - 3FFh	ASYNCH ADAPTER (COM1:)	

	B	A	
B1	GND	-I/O CH CK	A1
	RESET DRV	SD7	
	+5V	SD6	
	IRQ9	SD5	
	-5VDC	SD4	
	DRQ2	SD3	
	-12VDC	SD2	
	OWS	SD1	
	+12VDC	SD0	
B10	GND	I/O CH RDY	A10
	-SMEMW	AEN	
	-SMEMR	SA19	
	-IOW	SA18	
	-IOR	SA17	
	-DACK3	SA16	
	DRQ3	SA15	
	-DACK1	SA14	
	DRQ1	SA13	
	-REFRESH	SA12	
B20	CLK	SA11	A20
	IRQ7	SA10	
	IRQ6	SA9	
	IRQ5	SA8	
	IRQ4	SA7	
	IRQ3	SA6	
	-DACK2	SA5	
	T/C	SA4	
	BALE	SA3	
	+5VDC	SA2	
	OSC	SA1	
B31	GND	SA0	A31

	D	C	
D1	-MEM CS16	SBHE	C1
	-I/O CS16	LA23	
	IRQ10	LA22	
	IRQ11	LA21	
	IRQ12	LA20	
	IRQ15	LA19	
	IRQ14	LA18	
	-DACK0	LA17	
	DRQ0	-MEMR	
D10	-DACK5	-MEMW	C10
	DRQ5	SD08	
	-DACK6	SD09	
	DRQ6	SD10	
	-DACK7	SD11	
	DRQ7	SD12	
	+5VDC	SD13	
	-MASTER	SD14	
D18	GND	SD15	C18

Figure 5-4 PC/AT I/O card slot connectors.

Industry Standard Architecture (ISA)

- The example bus we're going to look at is the ISA
- It's an older standard, and simpler than some of the standards used in modern PC design
- Peripheral card manufacturers must conform to the appropriate standards to maintain plug in compatibility
- ISA is an 8-bit or 16-bit bus standard designed for the IBM PC, XT or AT

8-bit PC and XT ISA Signal Lines

A19 - A0 -- 20 address lines used for memory access

-- bottom 10 lines used for I/O access

ALE -- address latch enable, indicates a valid address present during a memory or I/O access

D7 - D0 -- 8 bidirectional data lines

IRQ7 - IRQ2 -- 6 maskable interrupt request lines

IOR_N, IOW_N -- I/O read and write control signals

SMEMR_N, SMEMW_N -- memory read and write control signals

IO_CH_RDY_N -- used by a memory or peripheral board to generate wait states

8-bit PC and XT ISA Signal Lines

RESET -- system reset signal

CLK -- 4.77 to 8.33 MHz clock synchronized to read / write operations

OSC -- A 14.318 MHz signal used by some video cards. Not synchronized to rest of the bus.

DRQ3 - DRQ1 -- Direct Memory Access (DMA) request lines;
DRQ0 is used for DRAM refresh on the PC and the XT

DACK3 - DACK1 -- DMA acknowledge lines

0WS -- signal from the fast peripherals to instruct CPU remove wait states

8-bit PC and XT ISA Signal Lines

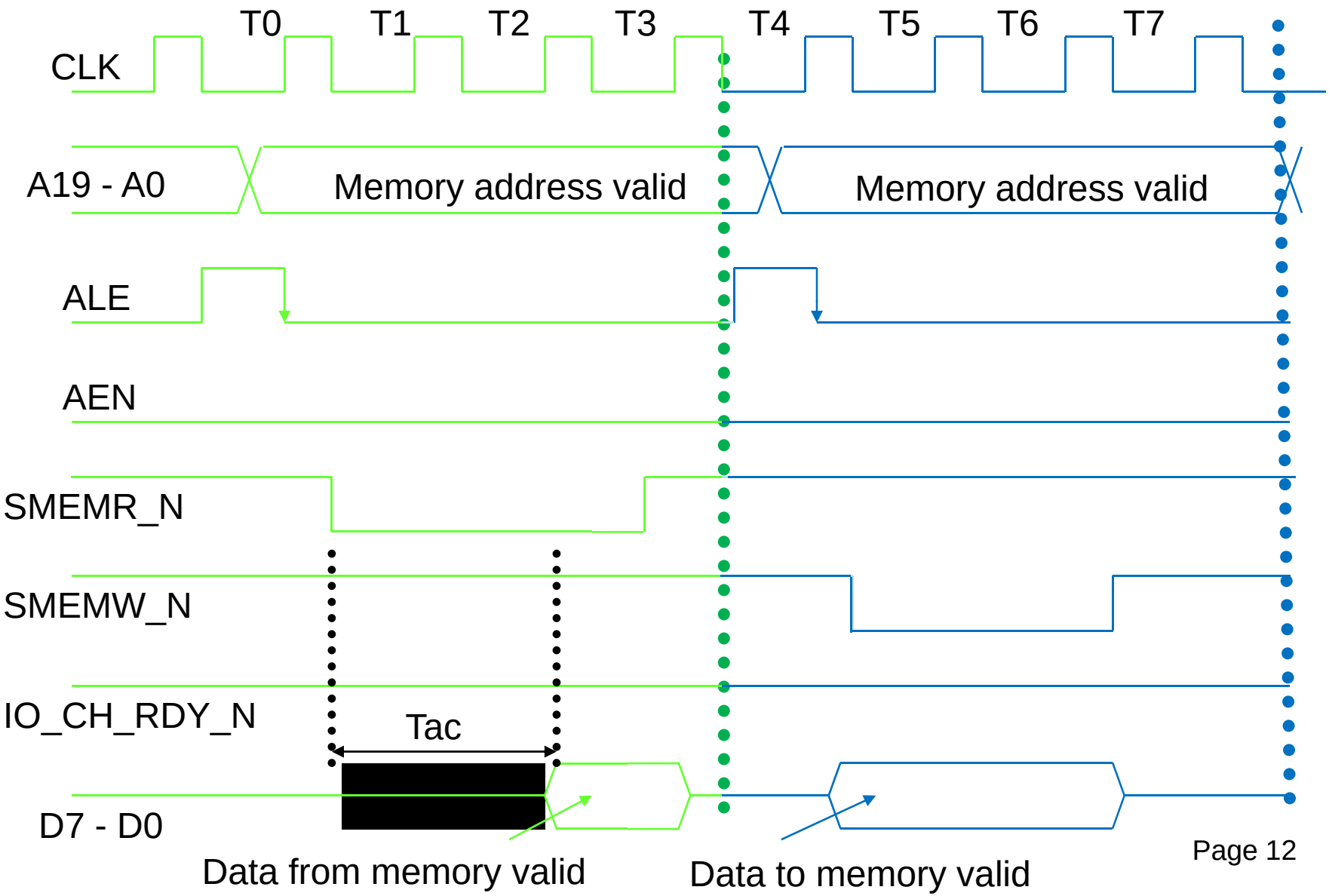
AEN -- address enable indication signal. Goes high during a DMA operation, stays low during a regular memory or I/O access. This is asserted when a DMAC has control of the bus. This prevents an I/O device from responding to the I/O command lines during a DMA transfer.

TC -- terminal count, DMA controller indicates the requested number of memory transactions during a DMA operation until completion. It sends this signal to the requesting device and causes the device to generate an interrupt to the processor.

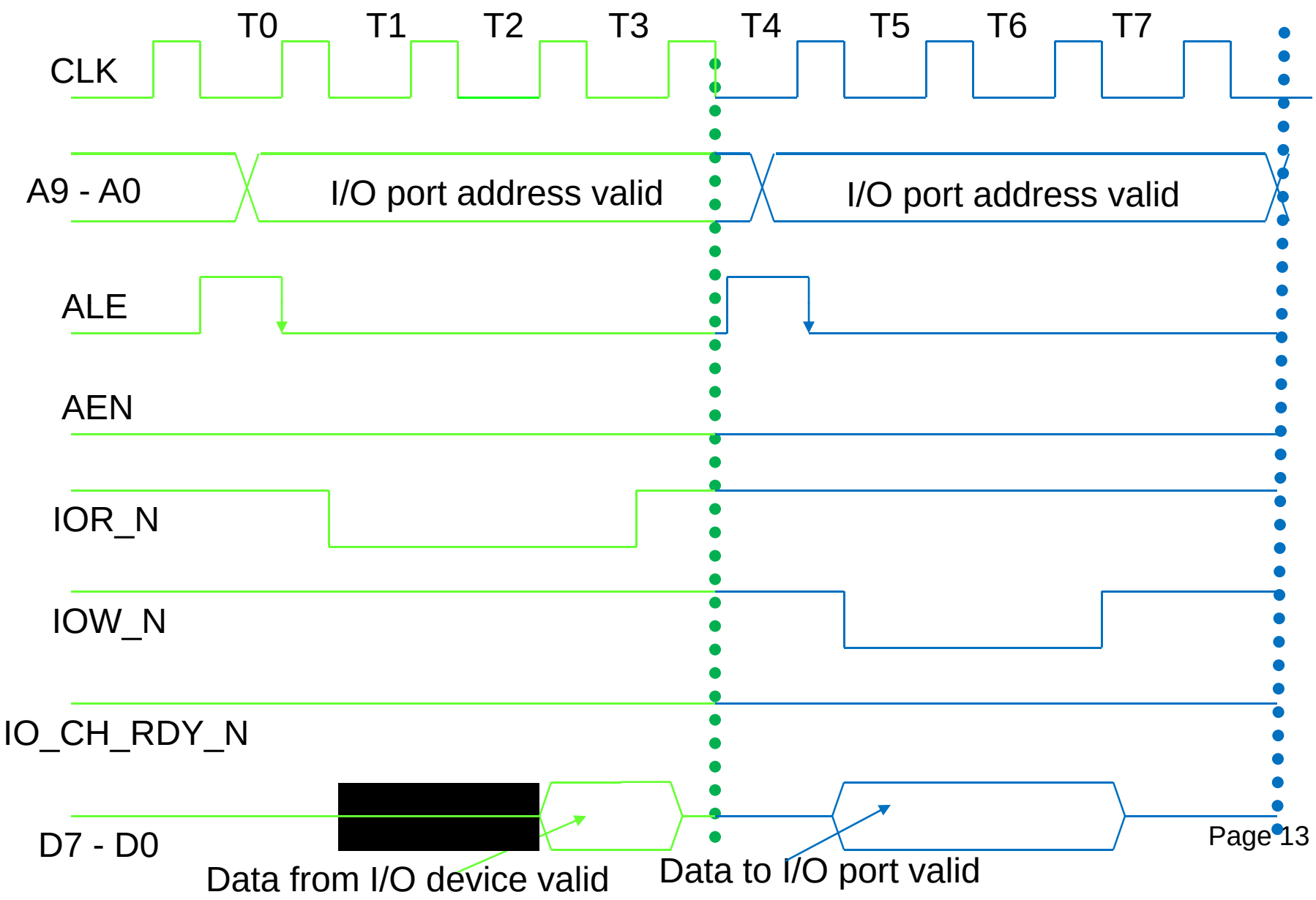
IO_CH_CHK_N -- alerts the processor to parity and other errors via a nonmaskable interrupt

+5V, -5V, +12V, -12V, GND -- supply voltages and ground

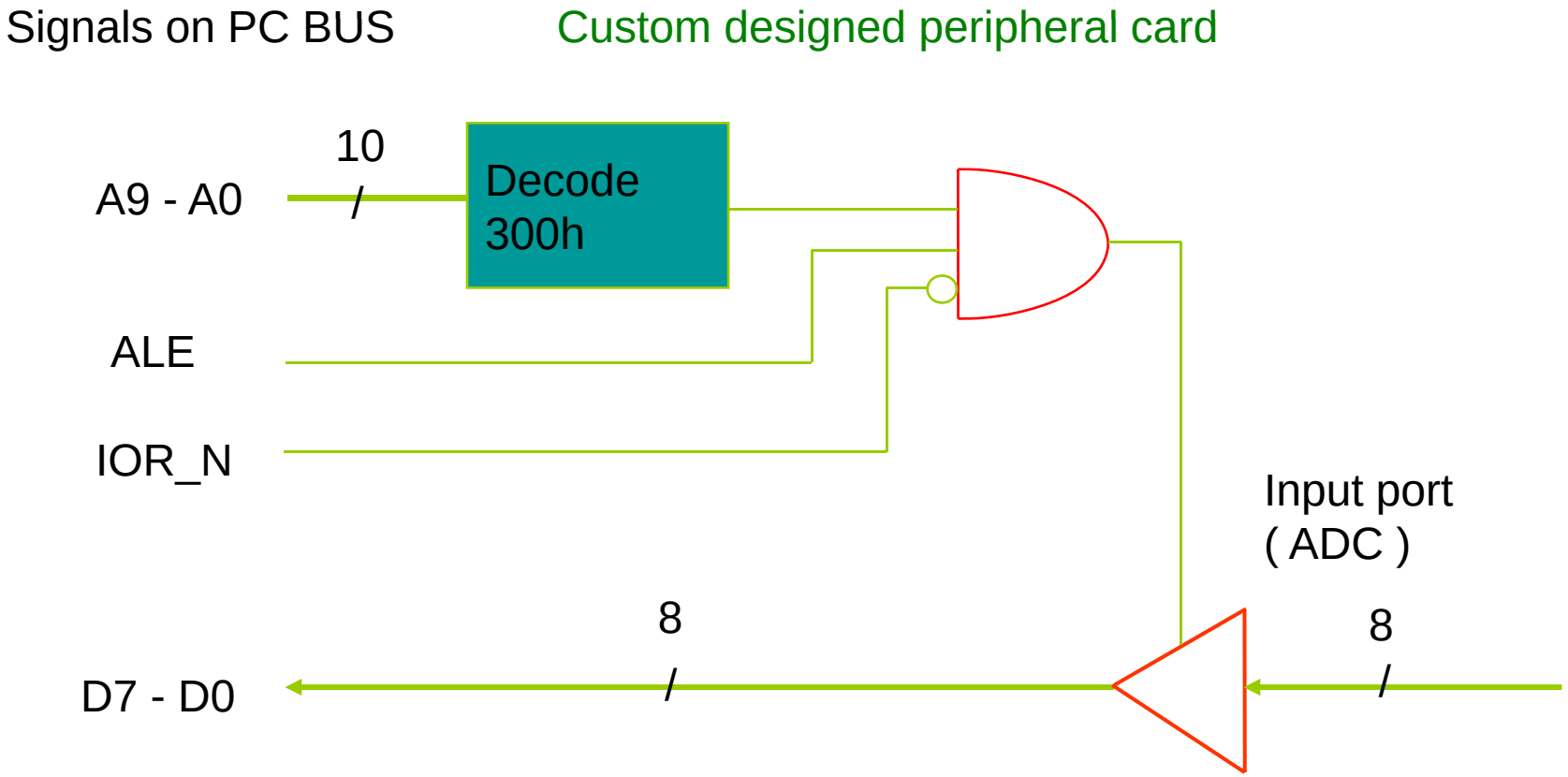
8088 CPU memory bus read and write cycle



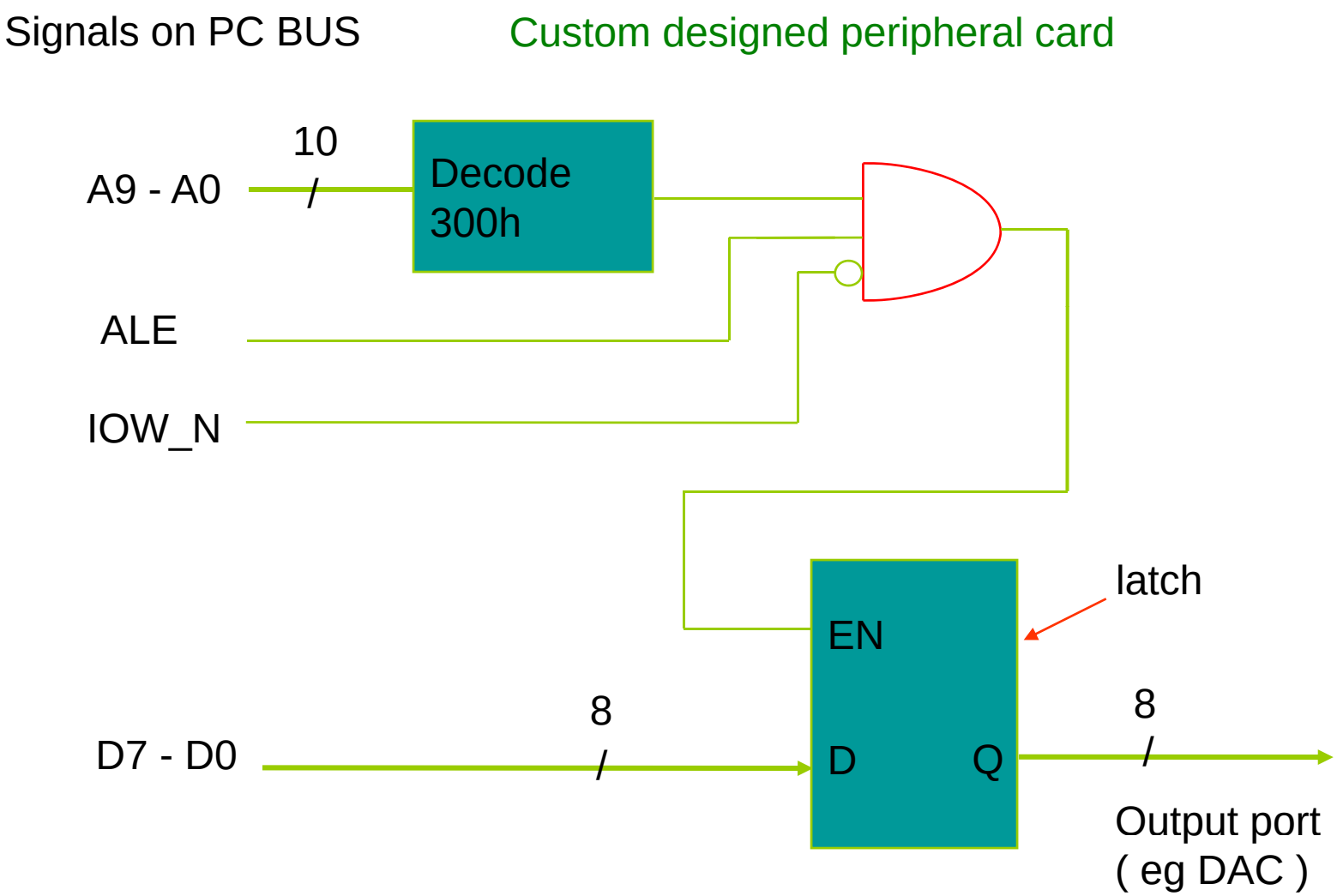
8088 CPU I/O port read and write cycle



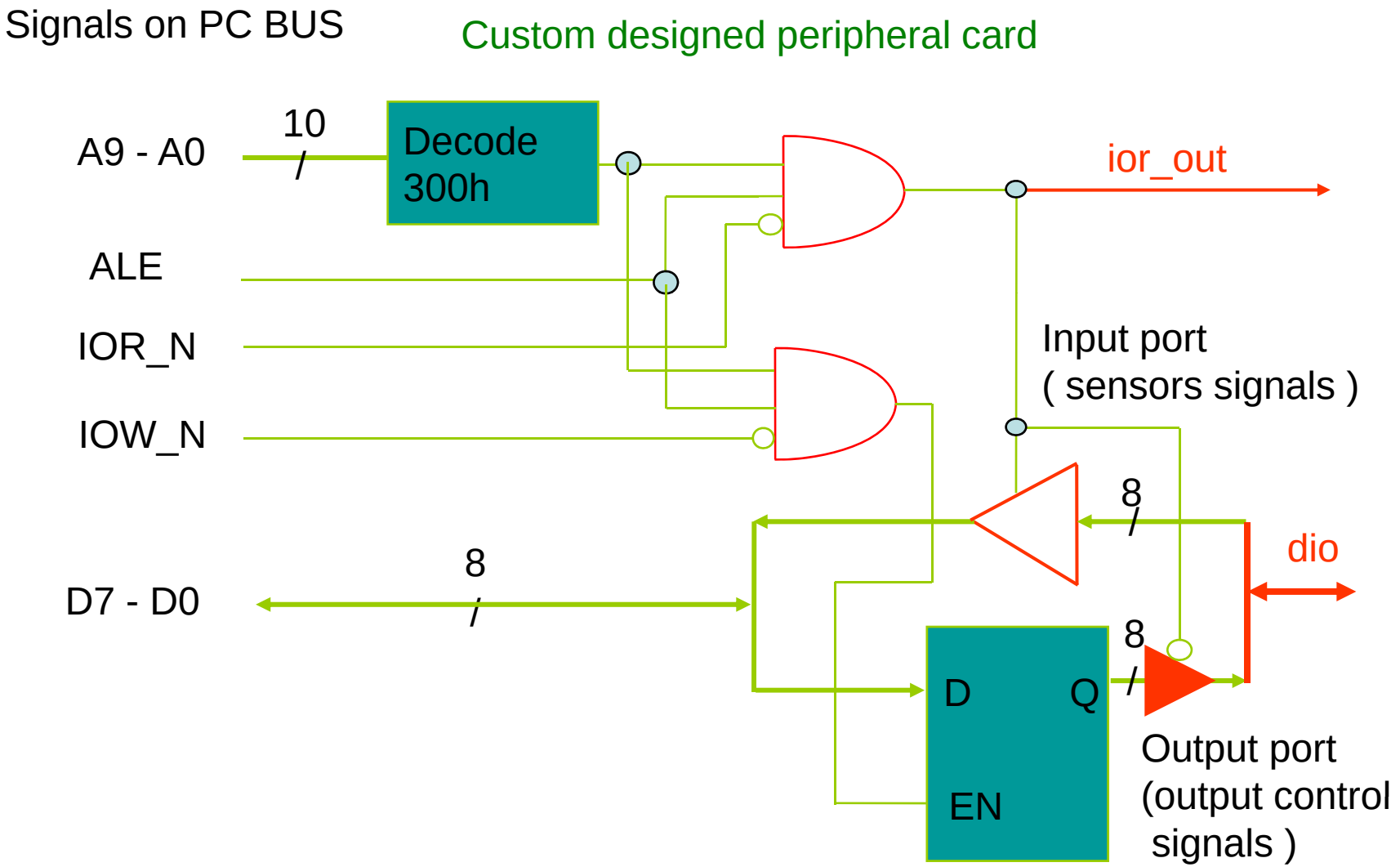
Simple 8-bit PC/XT digital input port design



Simple 8-bit PC/XT digital output port design

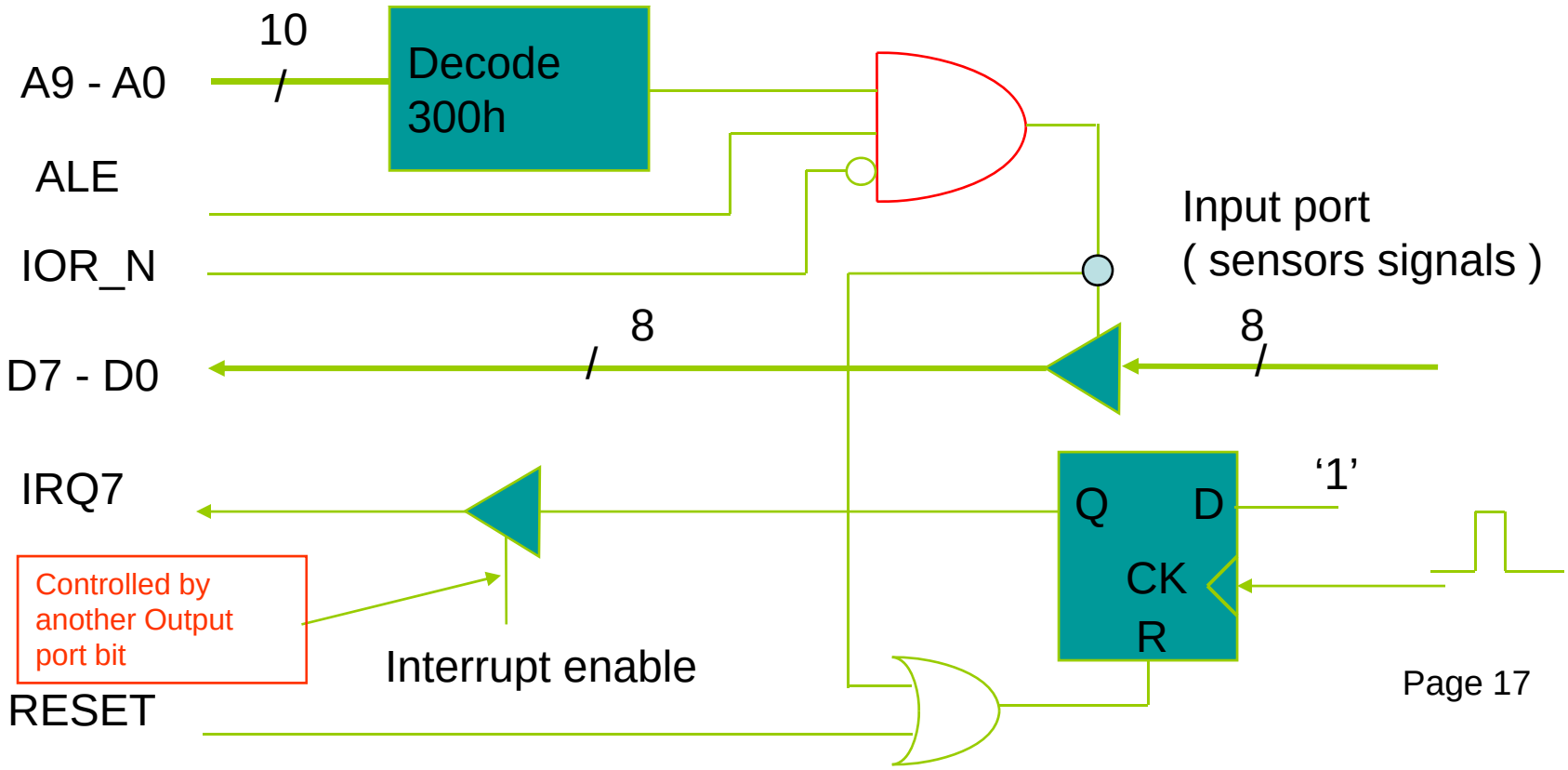


Simple 8-bit PC/XT digital input/output port design

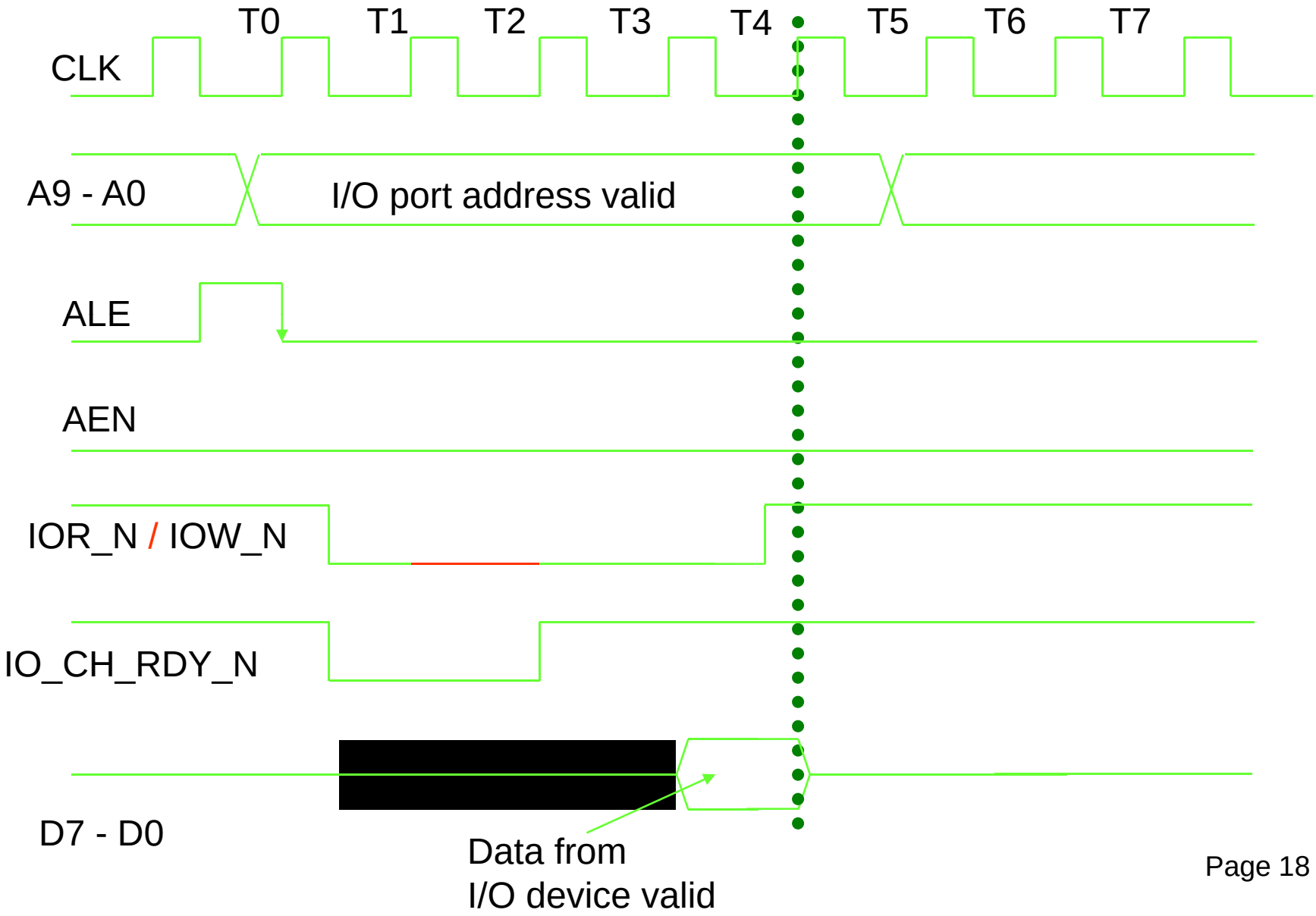


Interrupt driven 8-bit PC/XT digital input port

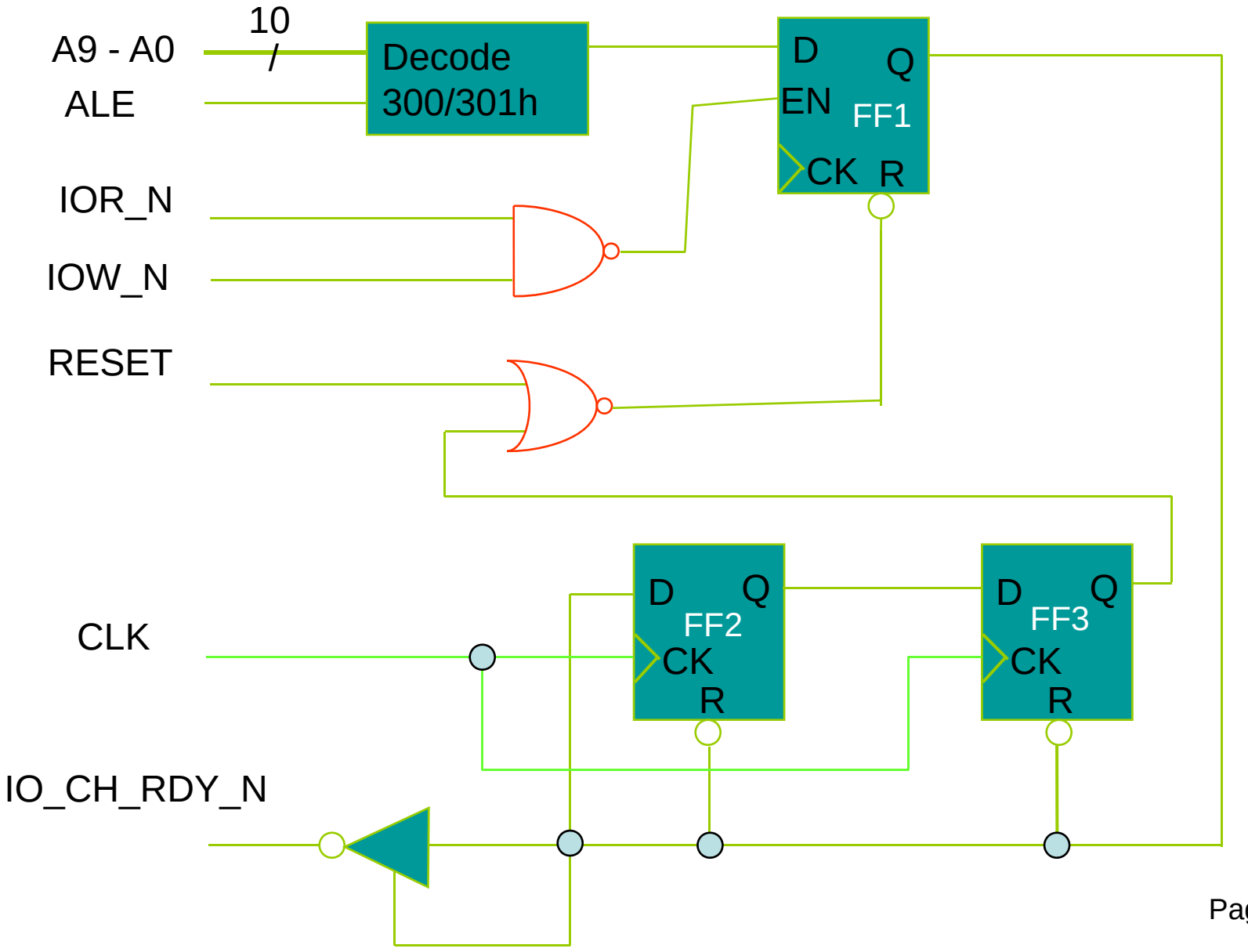
- Peripheral device has data ready to be read by CPU
- It places data on the input port (300h) and notify the CPU with an interrupt signal
- CPU handles the interrupt by reading the input data and reset the interrupt flip flop at the same time



8088 CPU wait state timing



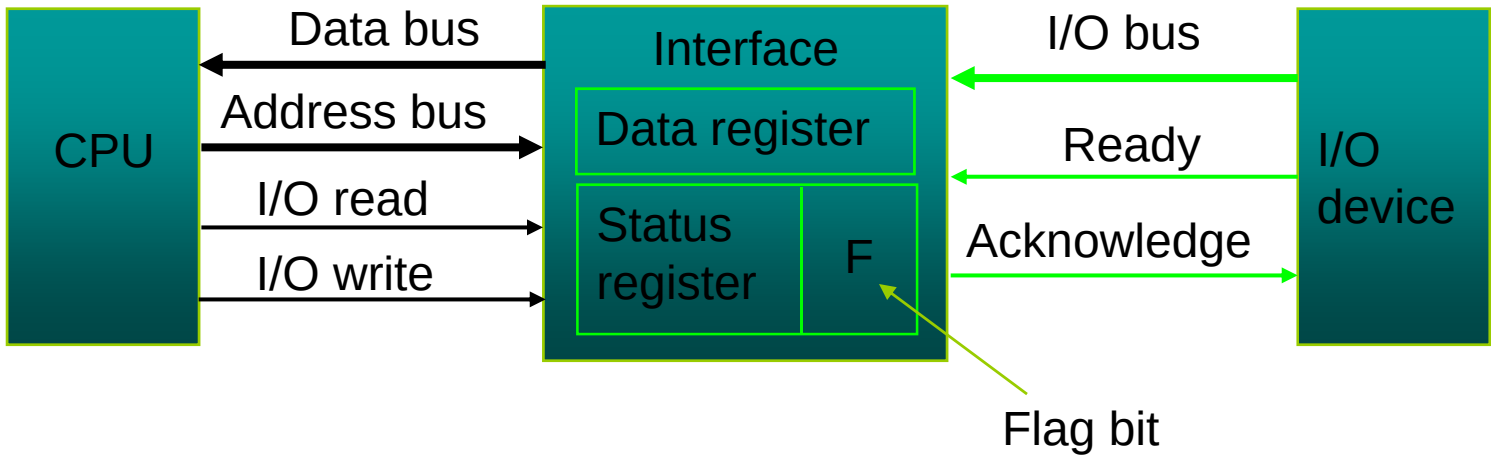
Wait state generation logic



Modes of Data Transfer between a CPU and a peripheral

Data can be transferred between a computer and an I/O device via a different mode of operation:

1. Programmed I/O -- the CPU constantly monitors the status of the interface and reads in (or supplies) the data as required.



Data transfer from I/O device to CPU

Mano pp.594 – 597.

Sequence of Events

a. I/O device outputs data on the I/O bus and raises Ready line

b. Output data stored in Data register and Status bit F is set

c. Interface will assert Acknowledge signal

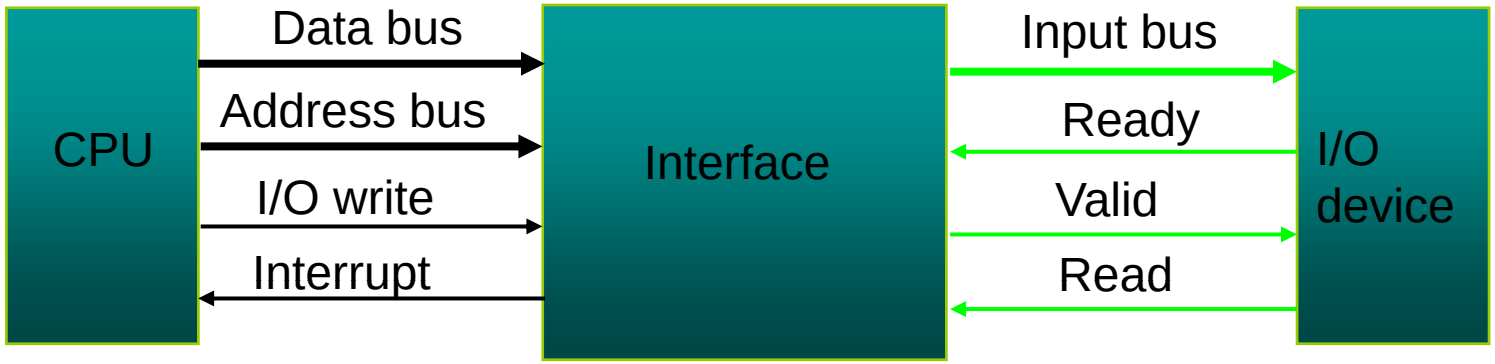
d. CPU polls the flag bit

e. If flag bit is set, the CPU will read contents from Data register, and , at the same time, clear the flag bit. Else, go to d

f. Acknowledge is dropped, I/O device can initiate another data transfer cycle

In the program-controlled transfer, the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer; the CPU is kept (needlessly) busy all the time so it won't lose any data!

2. Interrupt-initiated I/O -- the I/O device initiates a data transfer by drawing the attention of the CPU using a hardware interrupt. The CPU branches off from its normal operation to handle the data transfer and subsequently return to its normal operation.



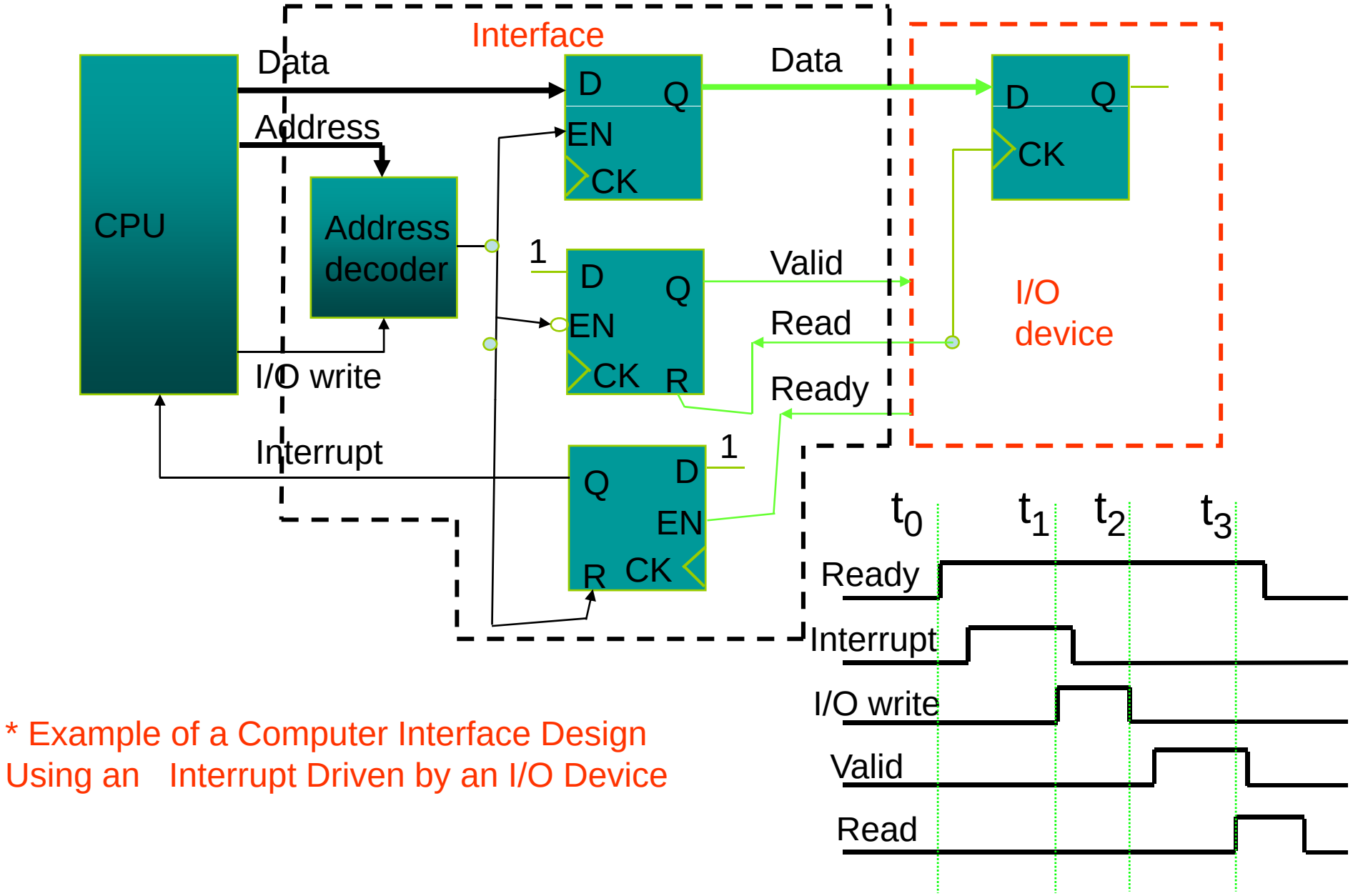
Data transfer from CPU to I/O device using an interrupt

a. I/O device, when ready to accept data, asserts Ready and triggers an interrupt

b. CPU services the interrupt by writing a data byte to the interface, which removes the interrupt signal from the CPU and raises the Valid line.

c. I/O device reads in data and thereby resets Valid

Still a fair amount of CPU overhead for every data transfer!



* Example of a Computer Interface Design Using an Interrupt Driven by an I/O Device

3. Direct Memory Access (DMA)

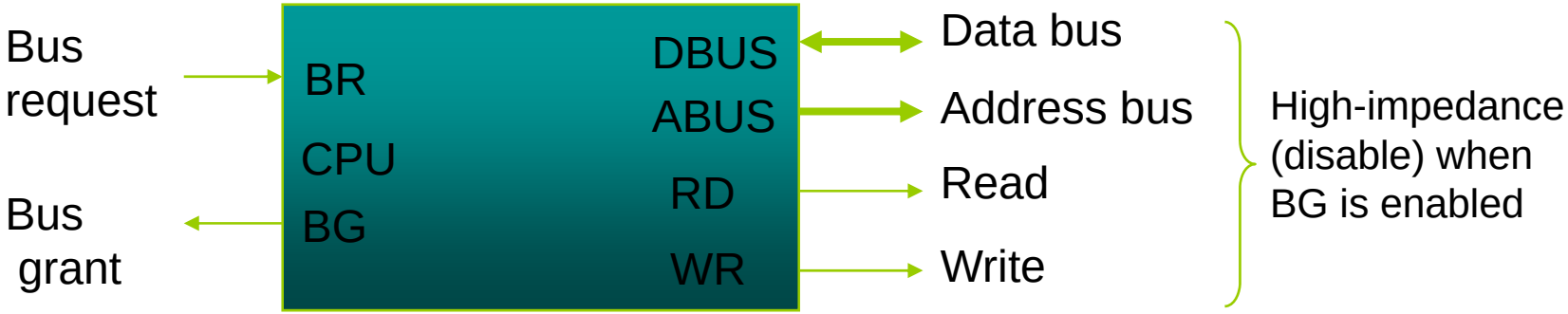
Mano pp.602 - 605

When a large amount of data is required to be transferred at a high rate, the operation usually involves memory access, e.g., hard disk, video display, etc.

If this is the case, is it necessary to get the CPU involved in every data Transfer cycle?

In a direct memory access operation, the CPU (middle man) is removed from the data path (idling) when it puts its address, data, and control bus in a high impedance state. The peripheral device, instead manages these buses directly with the help of a DMA controller.

A CPU which supports the DMA operation will have to have the following features:



The CPU uses 2 control signals to facilitate the DMA operation:

Bus request -- an input from the DMA controller to request the CPU to give up control of the bus

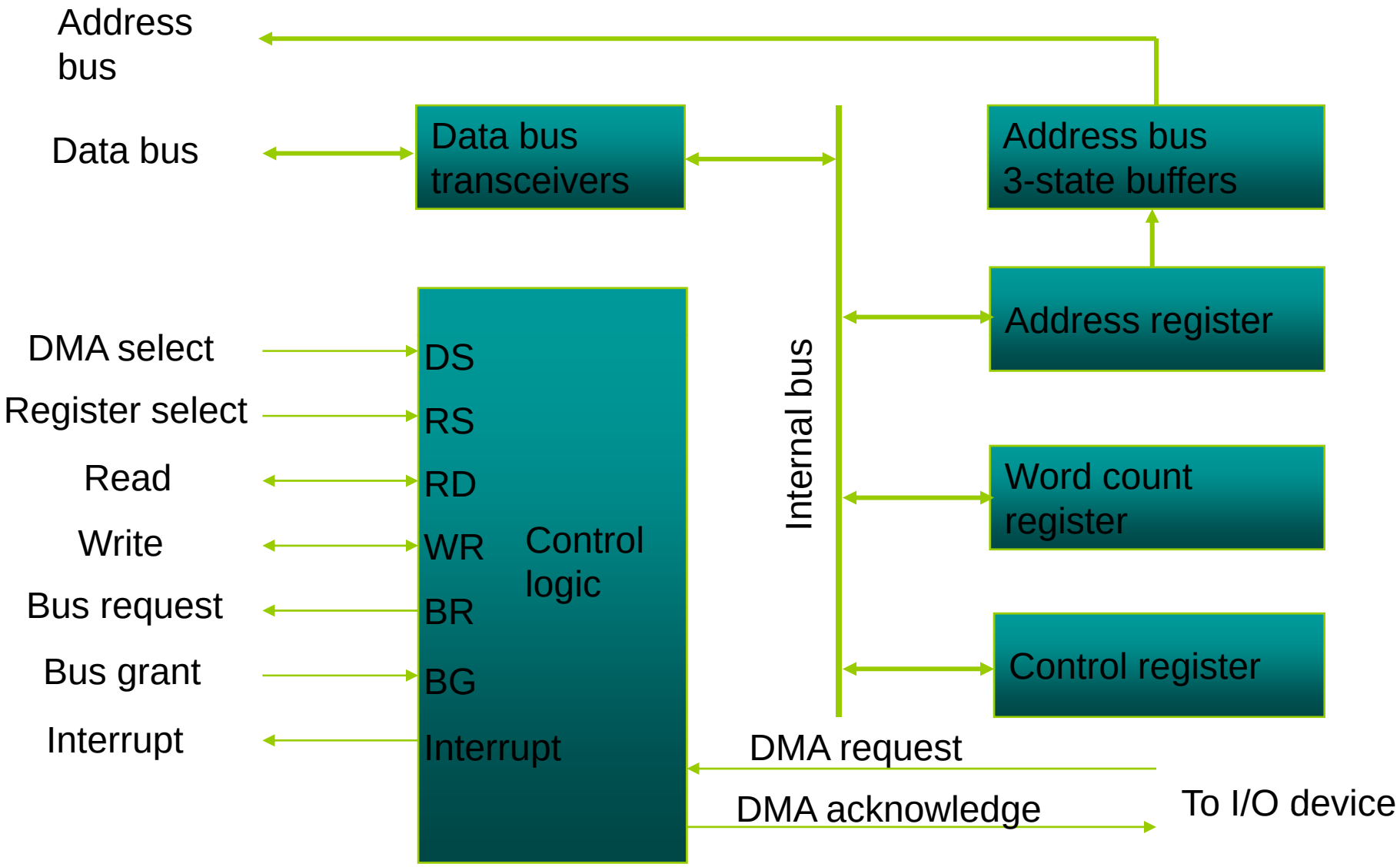
Bus grant – output from the CPU to indicate that the DMA controller can take control over the bus (including address, data, read, write)

There are 2 modes of DMA transfer:

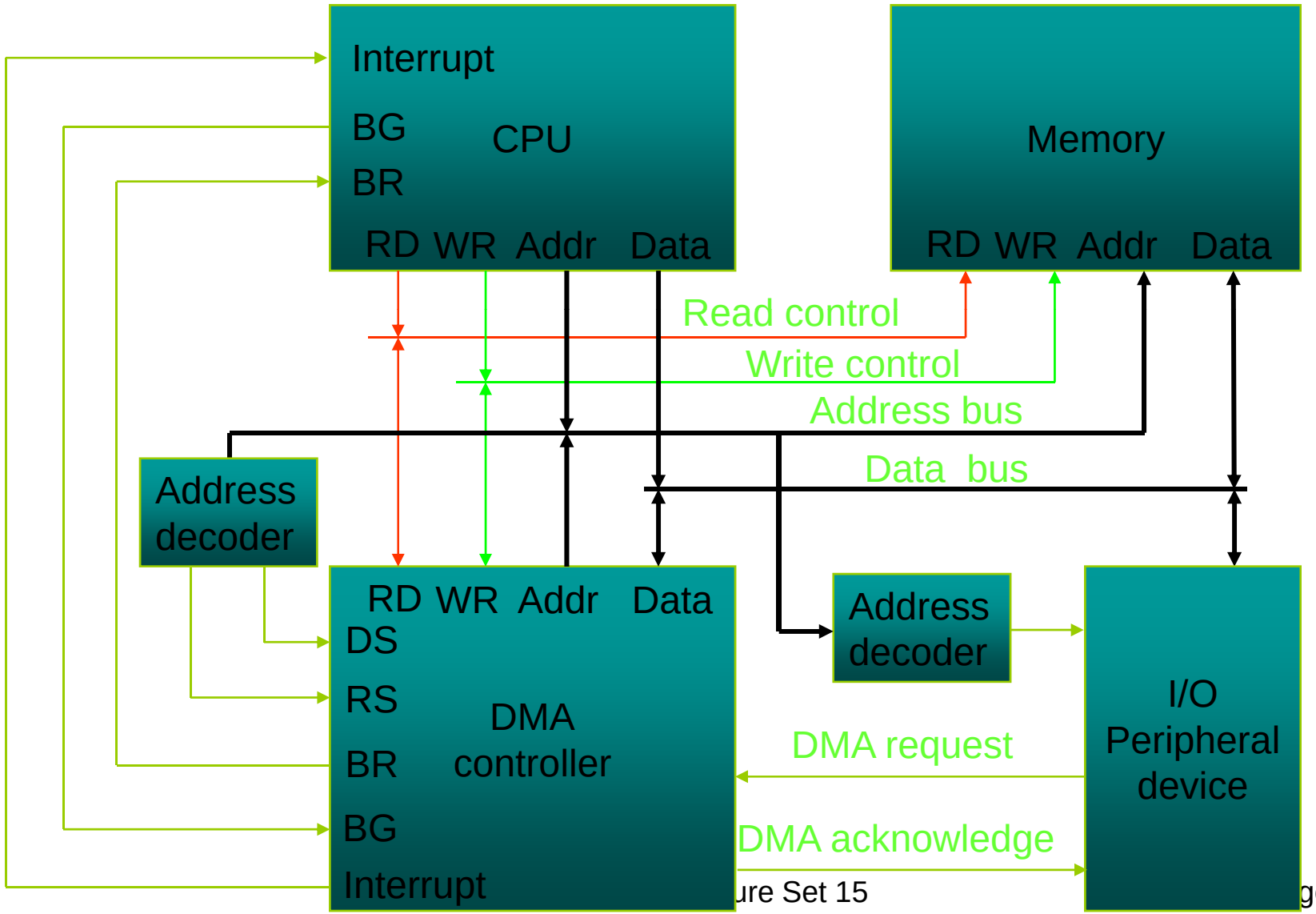
Burst – data words are transferred in a block

Cycle stealing – one word of data is transferred at a time; the CPU regains control after each cycle

Block diagram of DMA controller



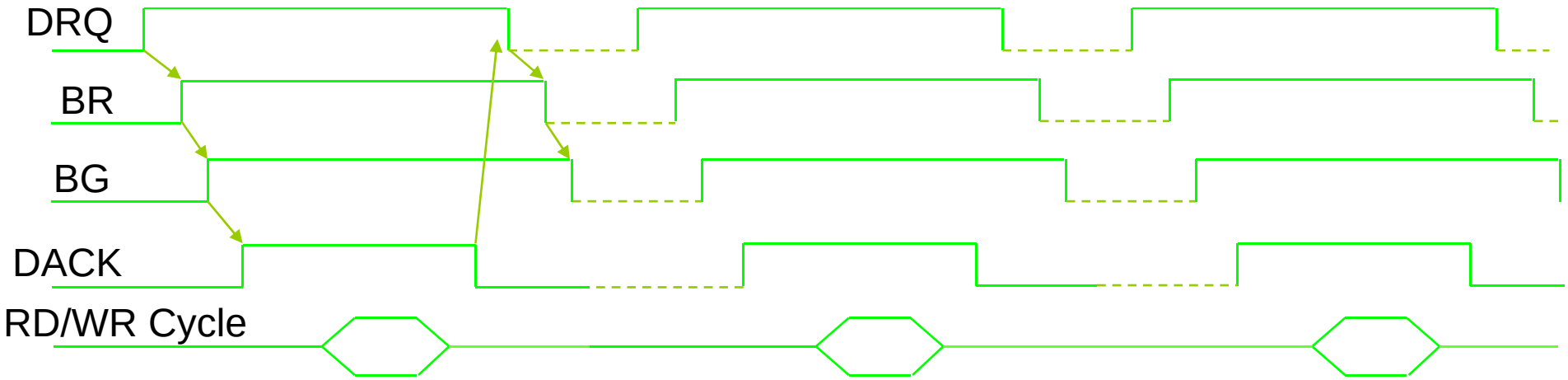
DMA transfer in a computer system



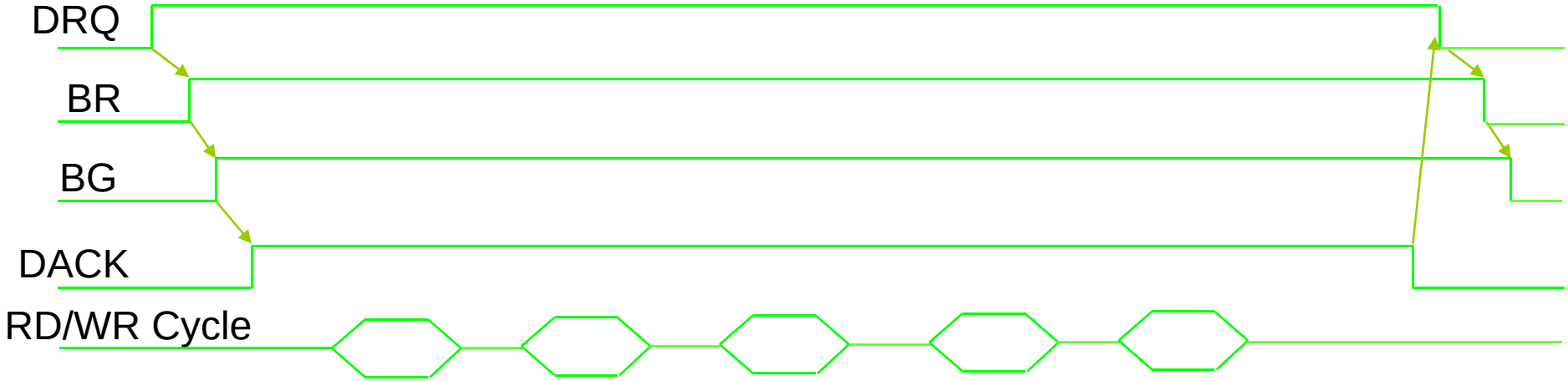
The sequence of operation in a DMA transfer:

- Under software control CPU configures DMA controller
- CPU sets up the I/O device for the desired operation
- I/O device sends DMA request when ready
- DMA controller activates Bus request
- CPU completes current instruction, relinquishes bus control,and issues Bus grant
- DMA controller gain control of the bus, and sends DMA acknowledge to I/O device requesting the transfer
- I/O device puts data on the data bus, DMA controller generates Read or Write signal to operate on memory at the address indicated by the internal address register
- DMA request goes low at the end of each transfer cycle. This signal is sampled by the controller to see if the word count is not zero. For a slower data transfer rate, DMA controller may let CPU regain the bus for a short while until DMA request is active again
- An interrupt is issued to the CPU when the word count register is zero

DMA Timing Diagram

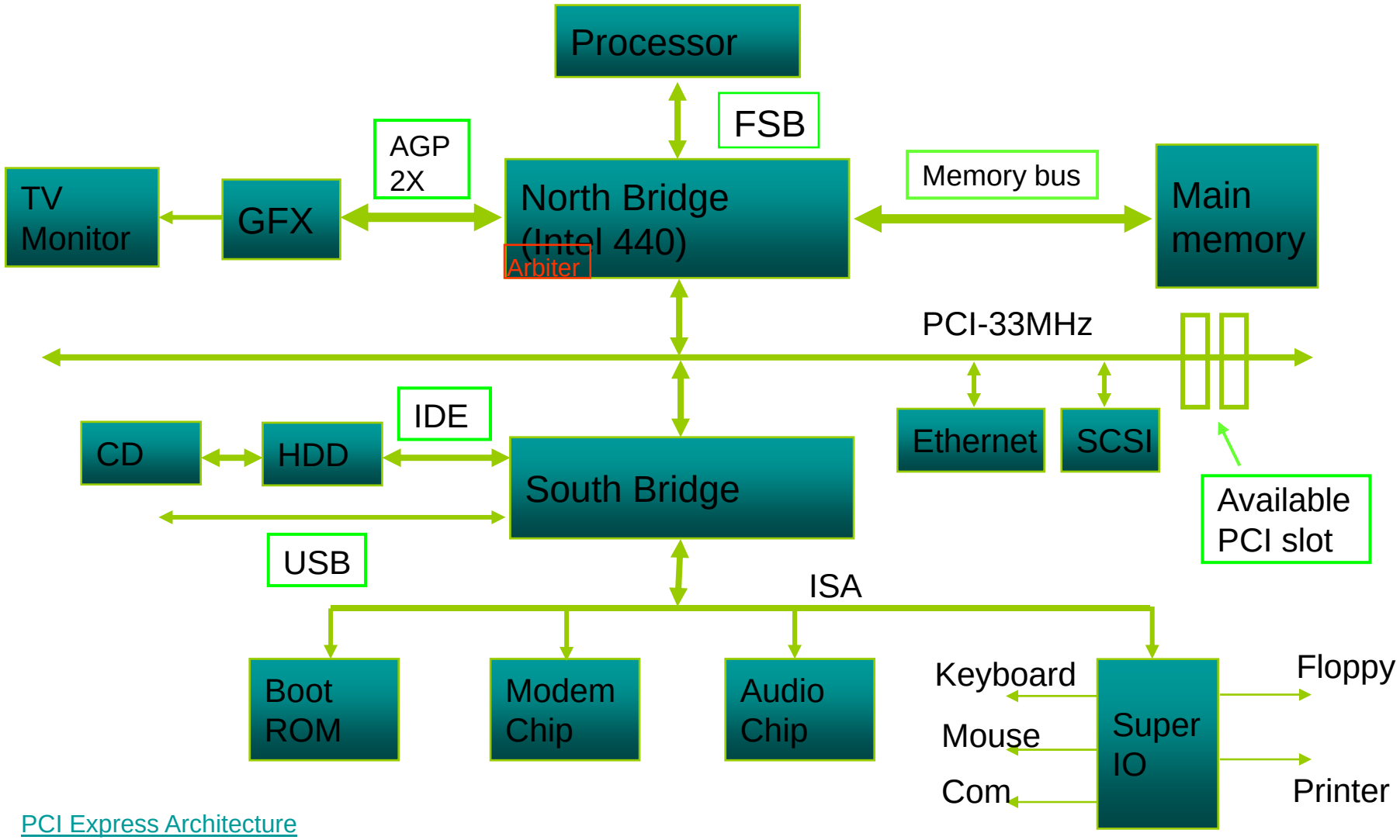


Single cycle DMA data transfer mode



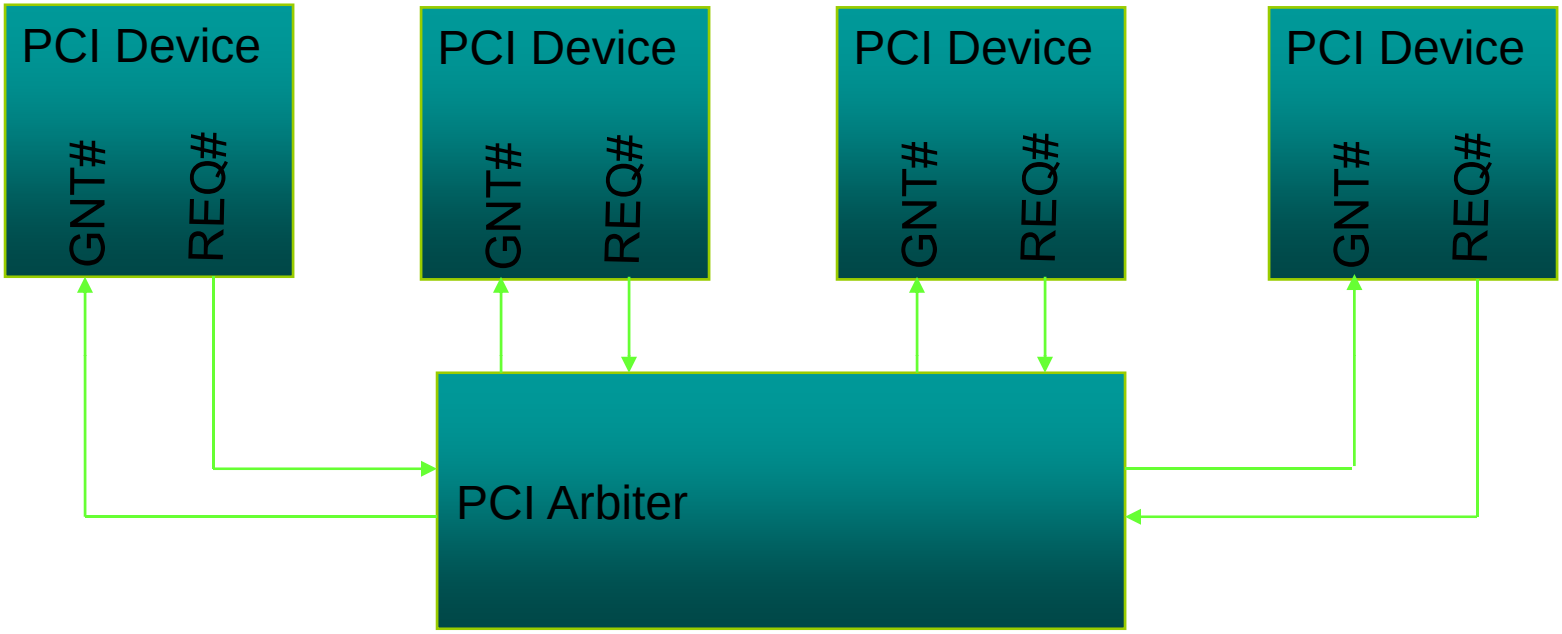
Burst DMA data transfer mode

Architecture of a Modern CPU system with a 33MHz PCI Bus



PCI Express Architecture

PCI uses a centralized synchronous bus arbitration scheme to arbitrate request of bus mastership by the various PCI devices in the system:



No particular arbitration algorithm has been specified by the PCI standard

Possible algorithms include : First Come First Serve
Round robin
.
.

PCI system designer has to ensure reasonable fairness in bus arbitration

Questions

- What is the difference between memory mapping and I/O mapping peripherals?

- What does the acronym ISA stand for in ISA bus?

Questions

- Given a base address and a memory footprint, you should be able design necessary input/output circuitry.

- What are the three different modes of data transfers between I/O peripherals and the CPU?

Questions

- What is the advantage of using a DMA?

- How does a DMA work?

Questions

- What are the steps for a memory read/write by a CPU?

- Modern CPU systems have a Northbridge/Southbridge architecture. Why?