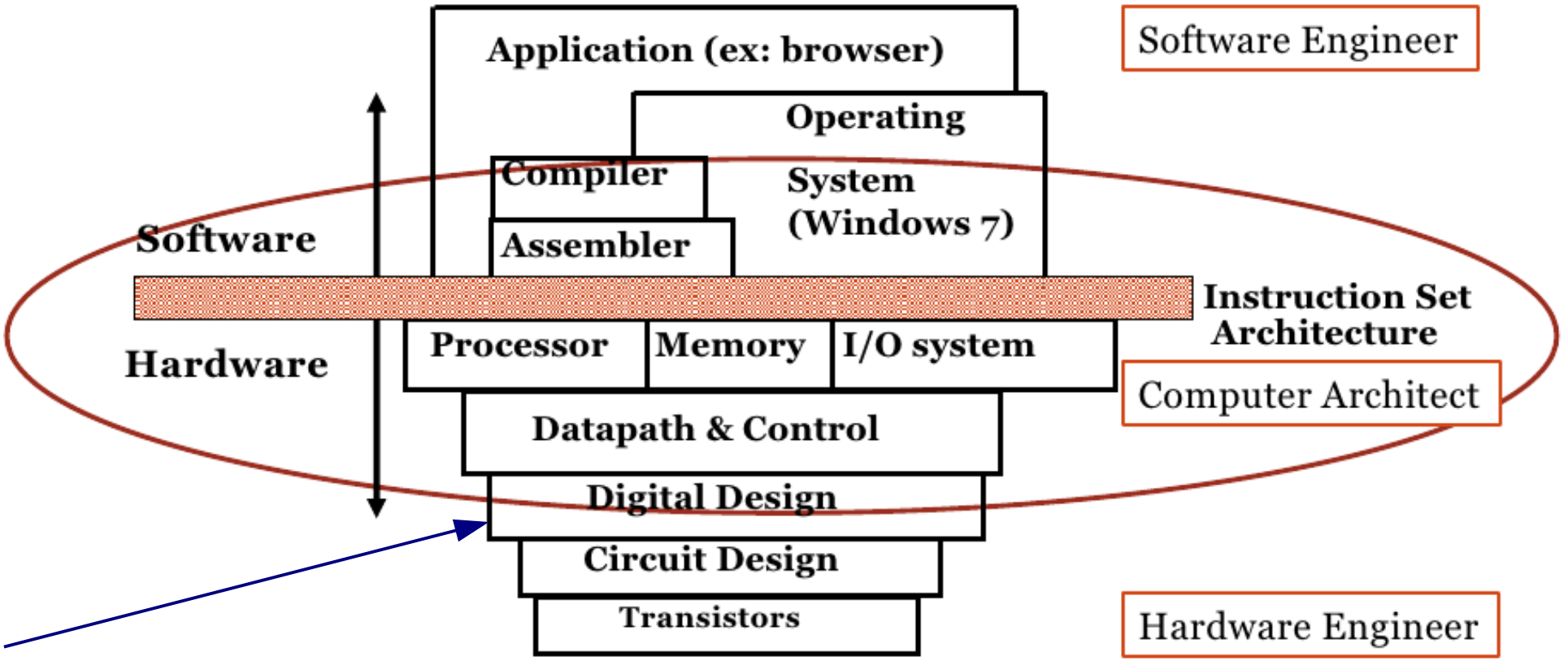
# Computer Architecture

## Ch3 – The Digital Logic Level

Nguyễn Quốc Đính, FIT – IUH

HCMC, Aug 2015

# Computer Structure




Coordination many levels of abstraction



Copied from UC Berkeley CS61C

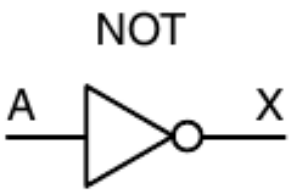
# Outline

- Gates And Boolean Algebra
- Basic Digital Logic Circuits
- Memory
- CPU Chips And Buses
- Example CPU Chips
- Example Buses

# Outline

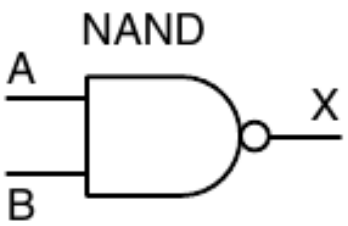
- Gates And Boolean Algebra
- Basic Digital Logic Circuits
- Memory
- CPU Chips And Buses
- Example CPU Chips
- Example Buses

# Gate and Boolean Algebra (1)



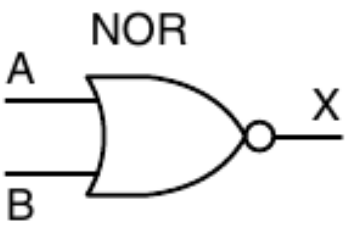

| A | X |
|---|---|
| 0 | 1 |
| 1 | 0 |

(a)



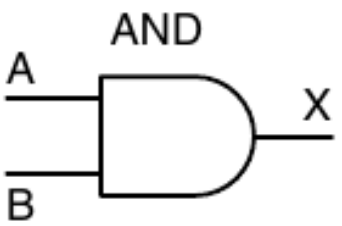

| A | B | X |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(b)



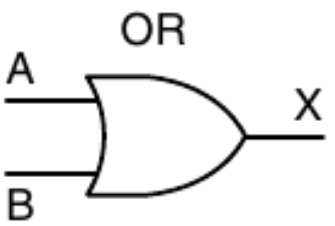

| A | B | X |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

(c)




| A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

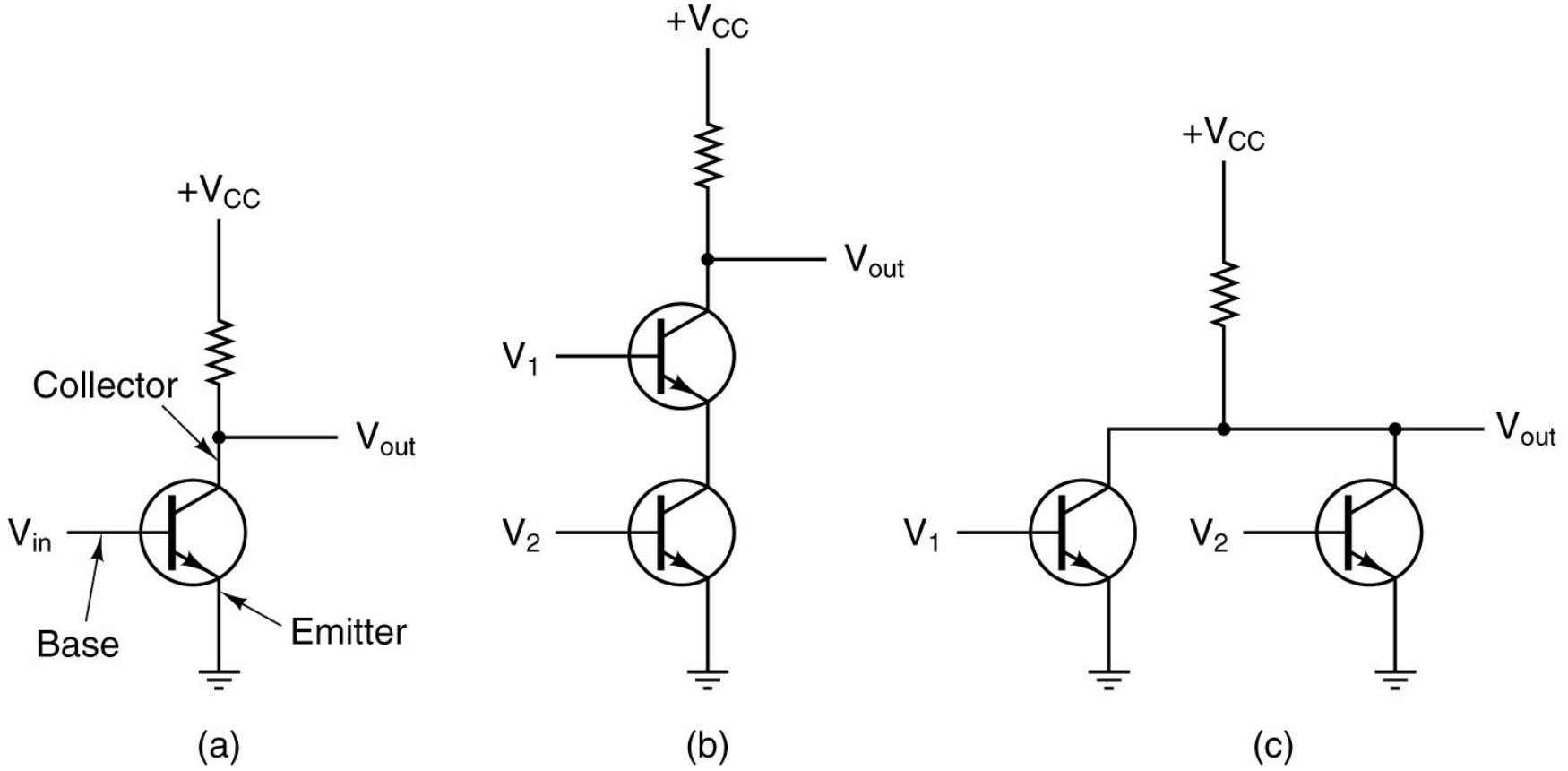
(d)




| A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

(e)

# Gate and Boolean Algebra (2)




(a) A transistor inverter.
(b) A NAND gate.
(c) A NOR gate.

# Boolean Algebra

| Name | AND form | OR form |
|---|---|---|
| Identity law | $1A = A$ | $0 + A = A$ |
| Null law | $0A = 0$ | $1 + A = 1$ |
| Idempotent law | $AA = A$ | $A + A = A$ |
| Inverse law | $A\overline{A} = 0$ | $A + \overline{A} = 1$ |
| Commutative law | $AB = BA$ | $A + B = B + A$ |
| Associative law | $(AB)C = A(BC)$ | $(A + B) + C = A + (B + C)$ |
| Distributive law | $A + BC = (A + B)(A + C)$ | $A(B + C) = AB + AC$ |
| Absorption law | $A(A + B) = A$ | $A + AB = A$ |
| De Morgan's law | $\overline{AB} = \overline{A} + \overline{B}$ | $\overline{A + B} = \overline{A}\,\overline{B}$ |

# Function Implementation

$$M = A \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot C + A \cdot B \cdot C$$

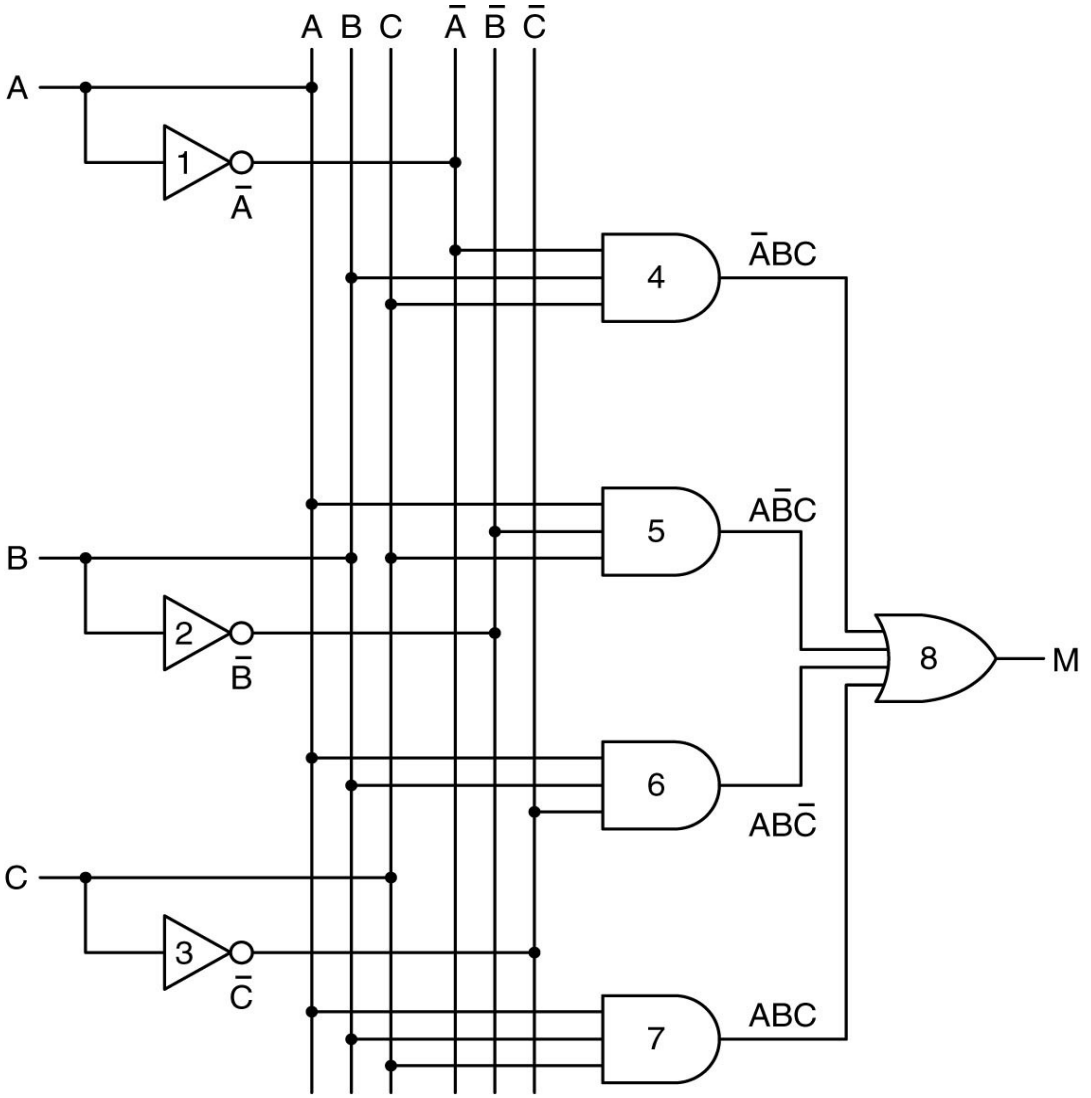
A direct form implementation:

- Generate the required signals and their inverse
- AND the terms that need to be
- OR the sum of products

$$M = A \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot C + A \cdot B \cdot C$$

| A | B | C | M |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

(a)



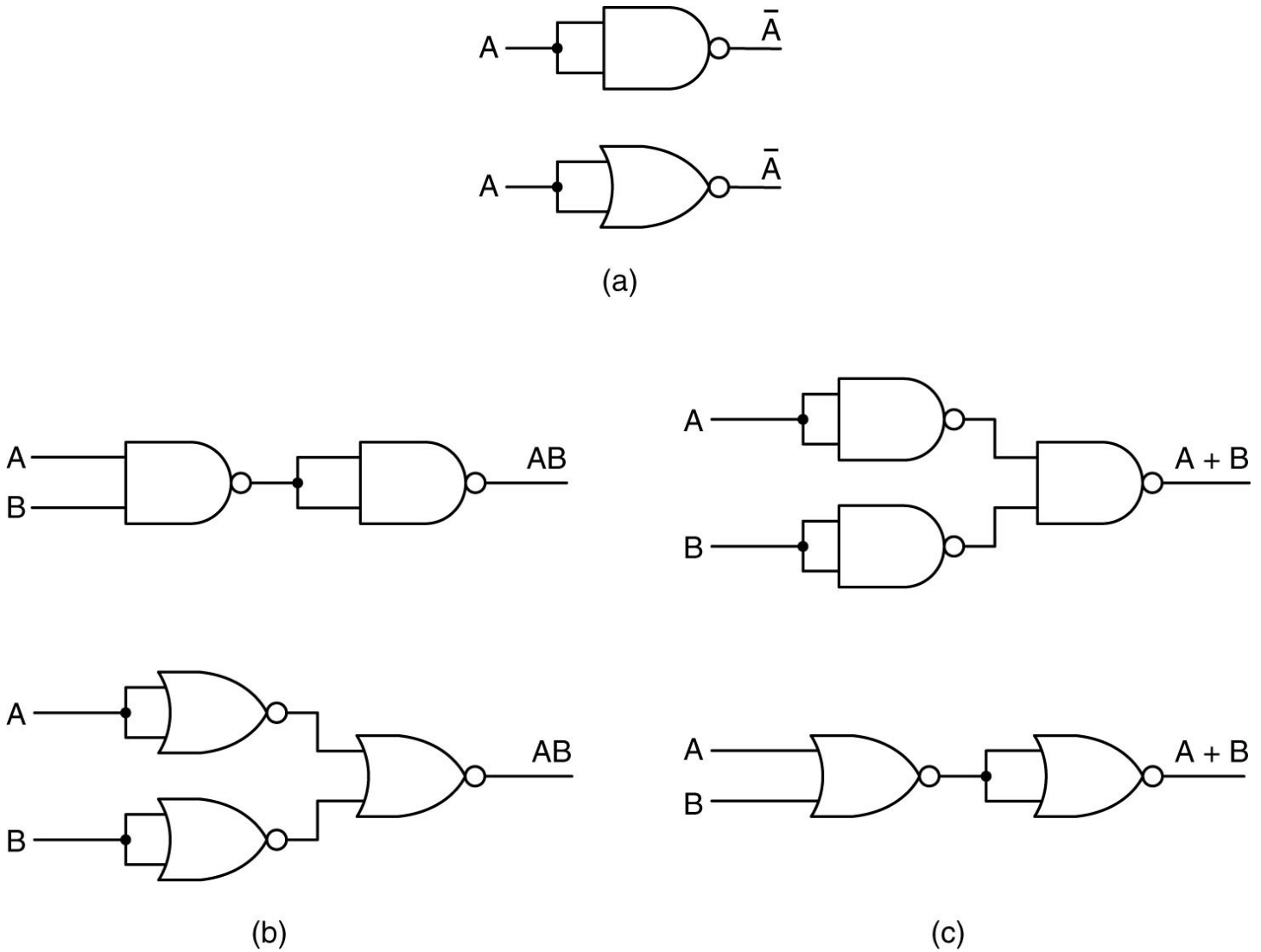

(b)

(a) Truth table for majority function of three variables.
(b) A circuit for (a).

It is straightforward to convert circuits generated by the preceding algorithm to pure NAND or pure NOR form

→ need a way to implement NOT, AND, OR using a single type

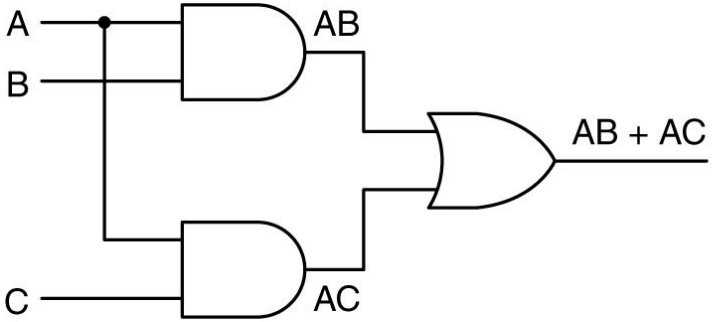
# Equivalent Gate

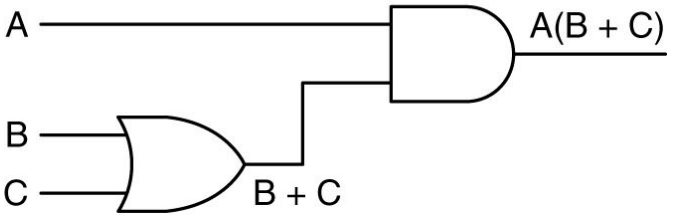



Construction of (a) NOT, (b) AND, and (c) OR gates using only NAND gates or only NOR gates.

Reduce number of gates to
- reduce component cost
- PCB space
- power consumption

# Circuit Equivalence




| A | B | C | AB | AC | AB + AC |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

(a)

| A | B | C | A | B + C | A(B + C) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

(b)

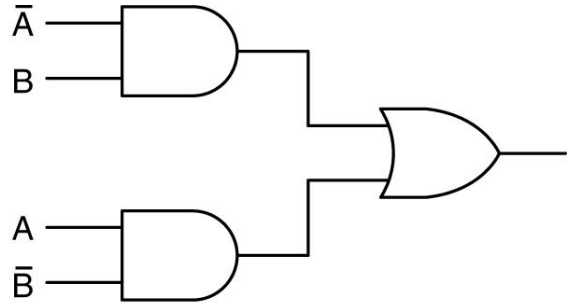
Two equivalent functions (a) AB + AC, (b) A(B + C).

# Circuit Equivalence

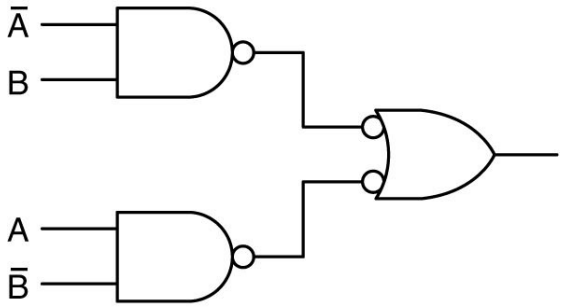
## The Exclusive Or (XOR) Function and Equivalent Gates

| A | B | XOR |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

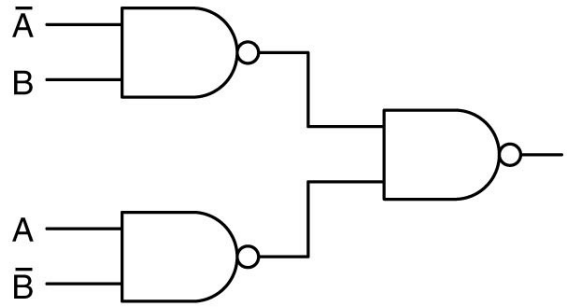
(a)




(b)




(c)




(d)

(a) The truth table for the XOR function.
(b-d) Three circuits for computing it.

# Logic Design Level

| A | B | F |
|---|---|---|
| $0^V$ | $0^V$ | $0^V$ |
| $0^V$ | $5^V$ | $0^V$ |
| $5^V$ | $0^V$ | $0^V$ |
| $5^V$ | $5^V$ | $5^V$ |

(a)

| A | B | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(b)

| A | B | F |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

(c)

(a) Electrical characteristics of a device.
(b) Positive logic.
(c) Negative logic.

# Problem (*):

Write a program to read in two arbitrary Boolean expressions and see if they represent the same function.

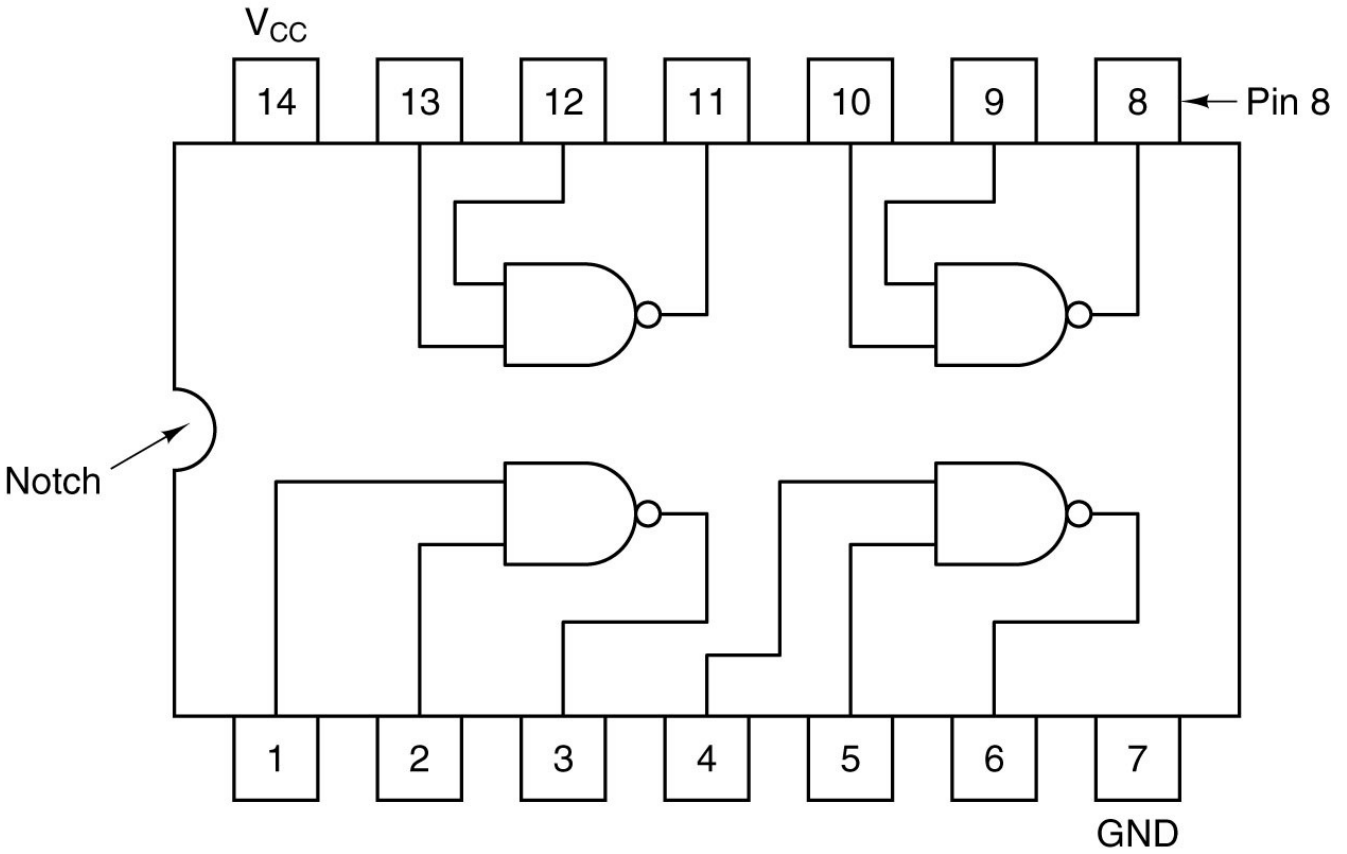
The input language should include single letters, as Boolean variables, the operands AND, OR, and NOT, and parentheses. Each expression should fit on one input line.

The program should compute the truth tables for both functions and compare them.

# Outline

- Gates And Boolean Algebra
- Basic Digital Logic Circuits
- Memory
- CPU Chips And Buses
- Example CPU Chips
- Example Buses

# Integrated Circuit




An SSI chip containing four gates.

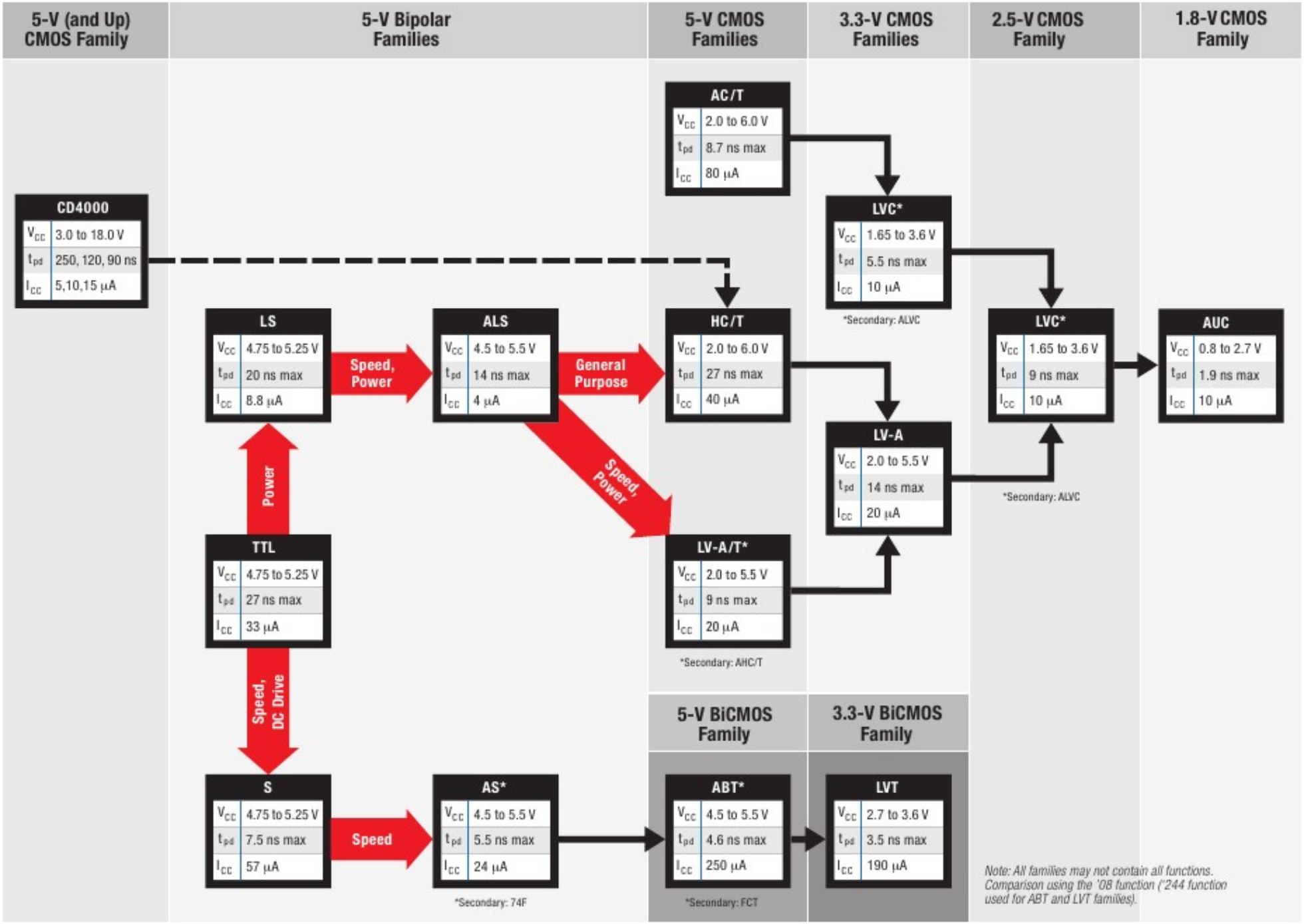
# Integrated Circuit

- Device Classifications
  - SSI: Small-Scale Integrated Circuit
  - MSI: Medium-Scale Integrated Circuit
  - LSI: Large-Scale Integrated Circuit
  - VLSI: Very Large-Scale Integrated Circuit
  - ULSI: Ultra-Large-Scale Integrated Circuit
  - GSI: Giga-Scale Integrated Circuit
  - TSI: Tera-Scale Integrated Circuit

# Integrated Circuit

- Device Packages (http://www.semiconfareast.com/packages.htm )
  - CDIP: Ceramic Dual In-Line Package
  - PDIP: Plastic Dual In-Line Package
  - PQFP: Plastic Quad Flat Pack (QFP)
  - PGA:  Pin Grid Array
  - BGA:  Ball Grid Array
  - PLCC: Plastic Lead Chip Carrier
  - SOJ:  Small Outline J-Lead
  - SOIC: Small Outline Plastic Package
  - PSOP: Plastic Small Outline Package (surface mount)
  - TSOP: Thin-Small Outline package (surface mount)
  - SSOP: Shrink Small Outline Package (surface mount)
  - TSSOP: Thin Shrink Small Outline Package (surface mount)
  - Etc.

From: http://www.ti.com/litv/pdf/sdyu001z, page 5



# Important Circuits for Computer Design

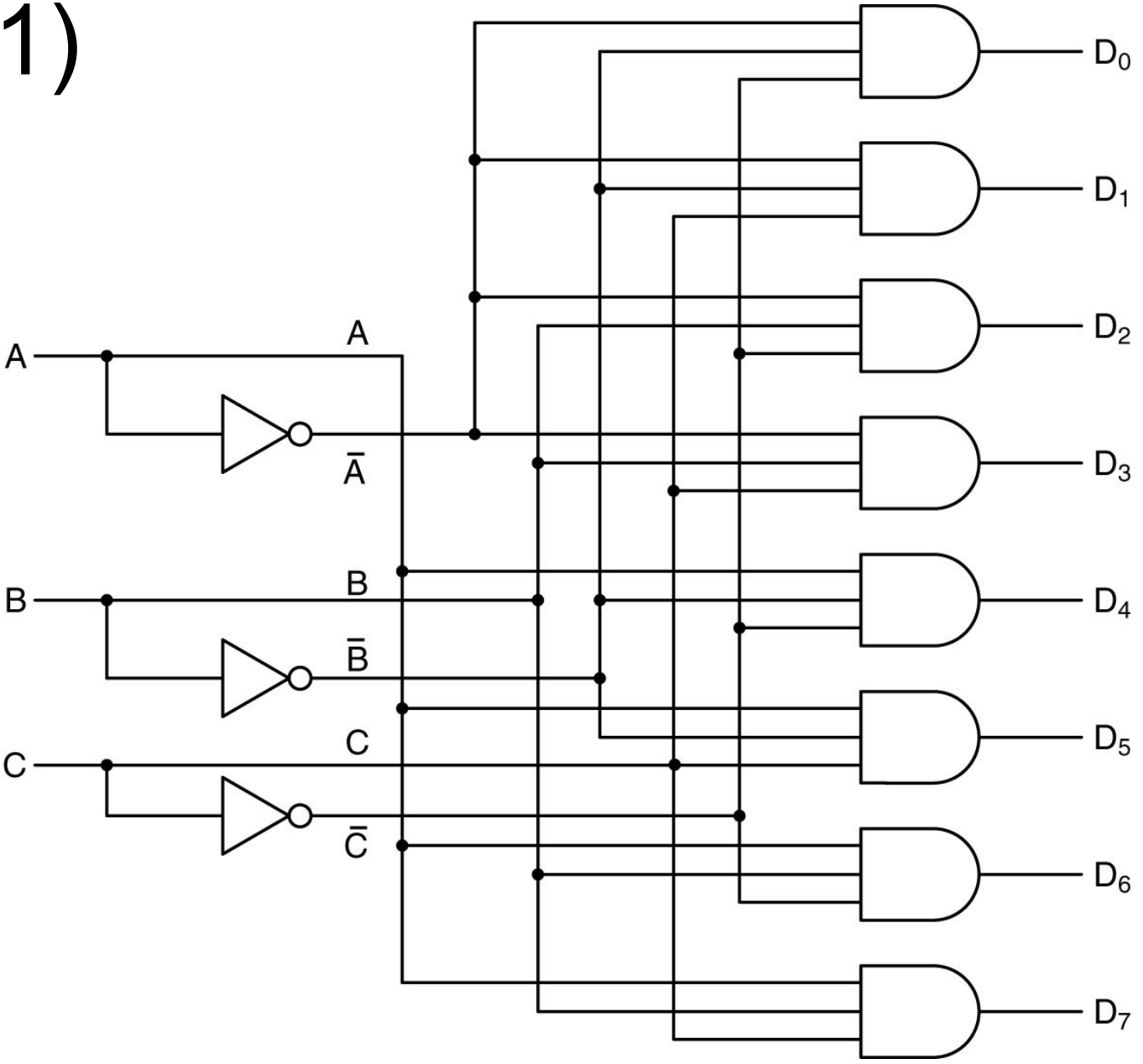
- Combinational Circuits
  - Decoders
  - Multiplexers
  - Comparators
  - Programmable Logic Arrays
- Arithmetic Circuits
  - Adders
  - Shifters
  - Arithmetic Logic Units (ALU)
  - Multipliers
- Latches, Flip-Flops and the Clock Glitch Generator
  - Set-Reset Latch
  - “Gated” SR Latch
  - “Gated” D Latch
  - The clock glitch (pulse) generator (only to be used by experts)
  - D Flip Flop
  - Synchronous Register Writing

# Important Circuits for Computer Design

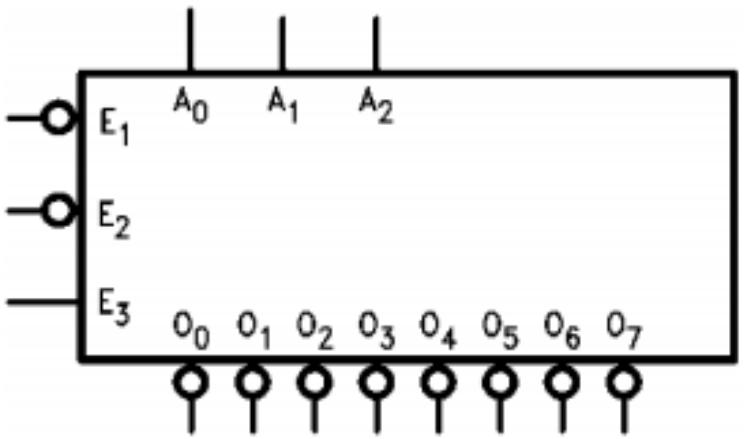
- Combinational Circuits
  - Decoders
  - Multiplexers
  - Comparators
  - Programmable Logic Arrays
- Arithmetic Circuits
  - Adders
  - Shifters
  - Arithmetic Logic Units (ALU)
  - Multipliers
- Latches, Flip-Flops and the Clock Glitch Generator
  - Set-Reset Latch
  - “Gated” SR Latch
  - “Gated” D Latch
  - The clock glitch (pulse) generator (only to be used by experts)
  - D Flip Flop
  - Synchronous Register Writing

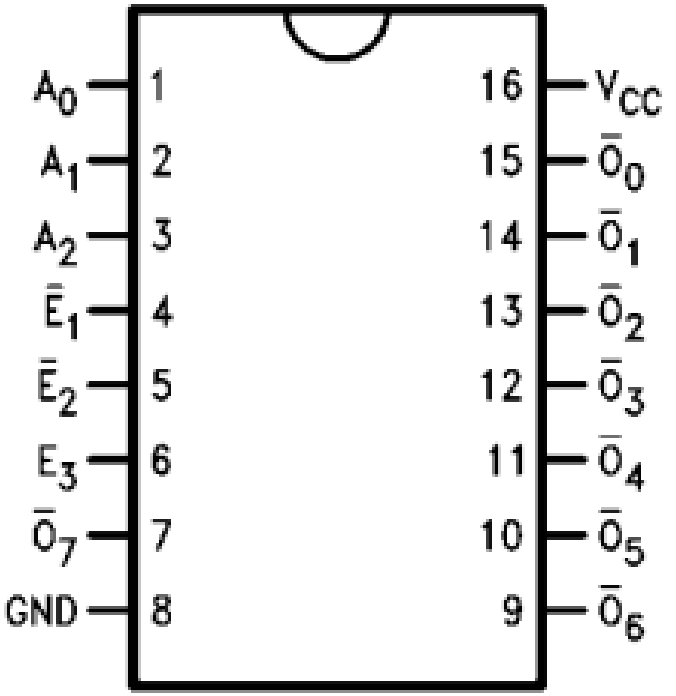
# Decoders (1)

A 3-to-8 decoder circuit.



# Decoders (2)

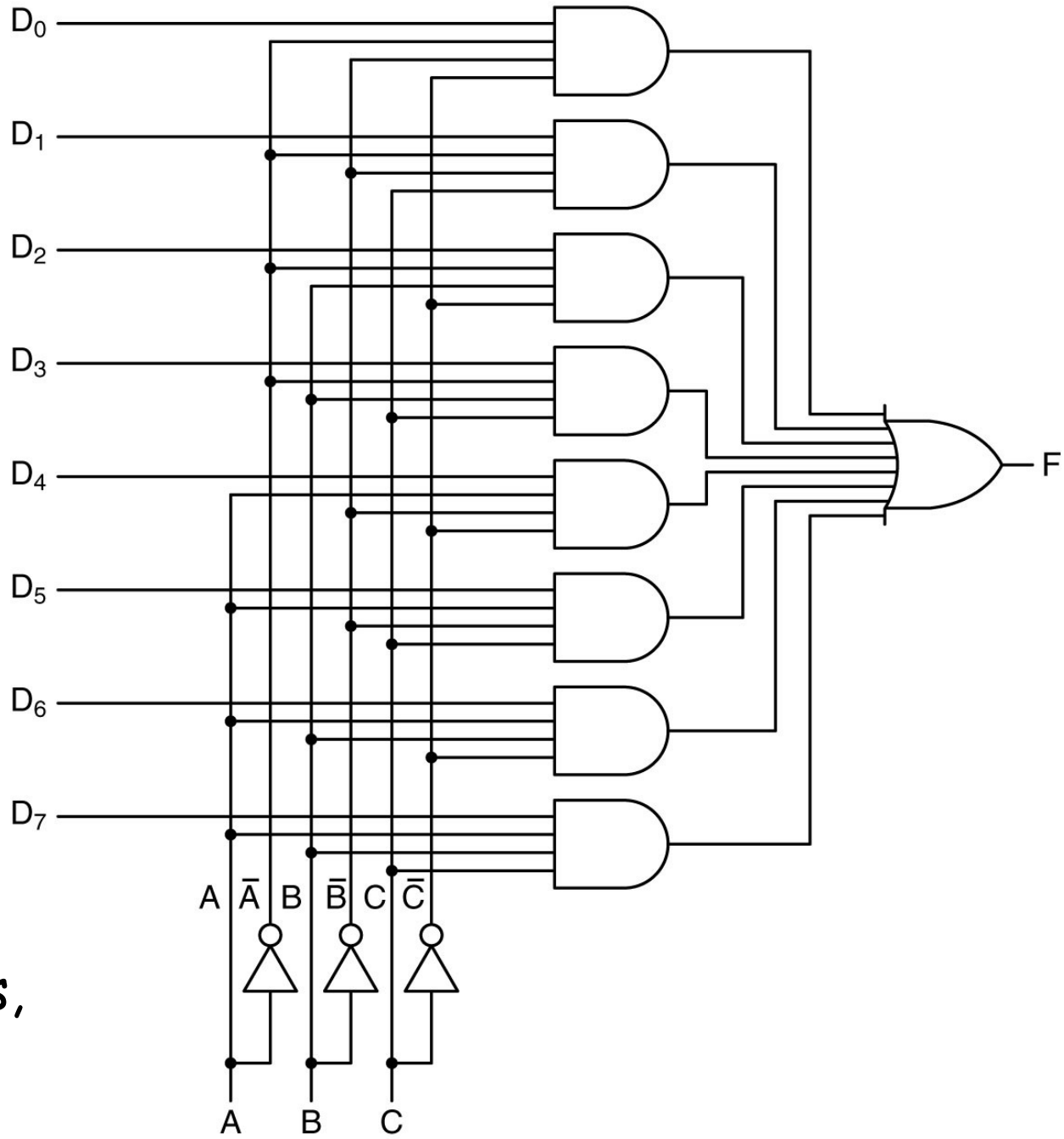



*A* 3-to-8 decoder circuit: 74F138

Used for generating specific “chip-select” signals from address busses or other encoded identification.

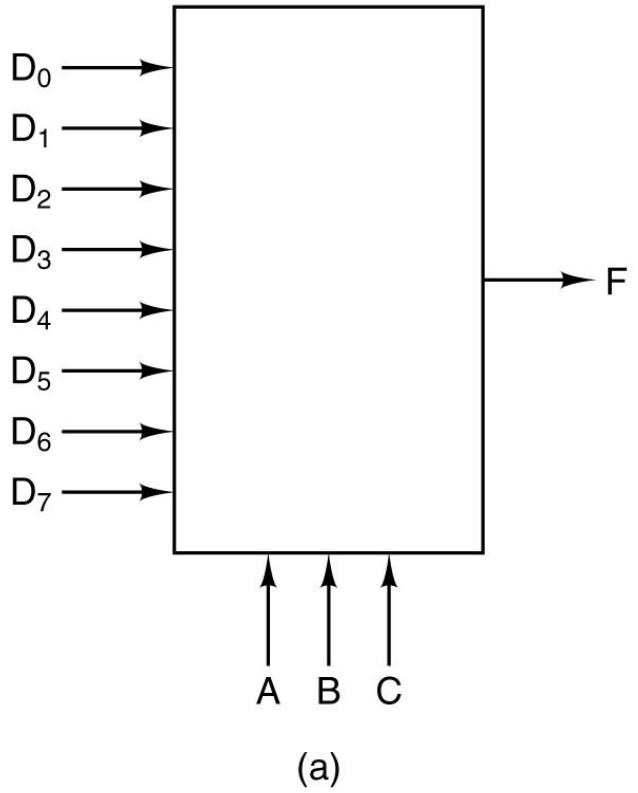
# Multiplexer (1)




8 to 1 multiplexer (8 inputs,
3 coded select lines)

# Multiplexer (2)

74X151/74X152: an MSI multiplexer.

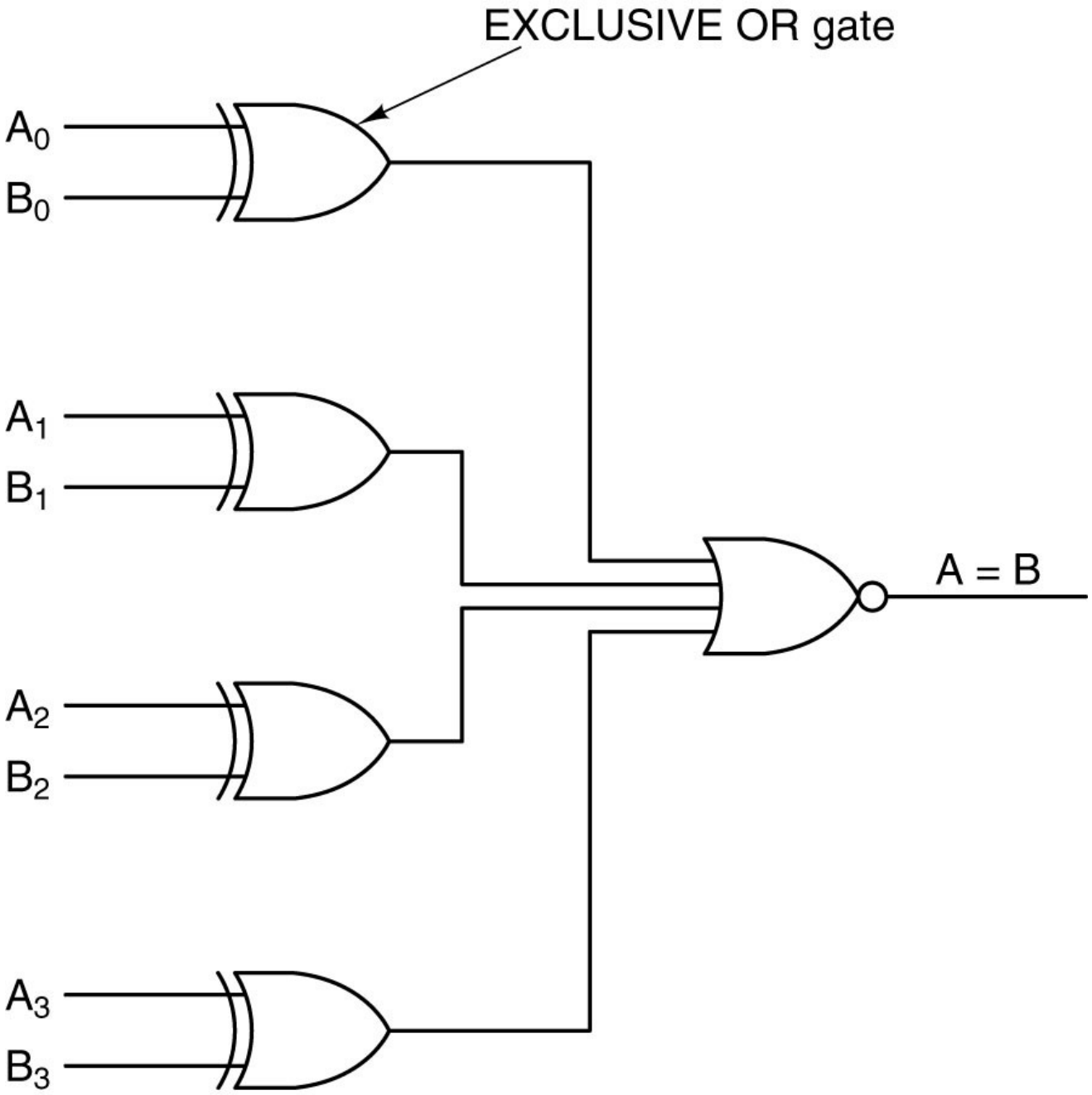


(a)

Used for selecting one of a number of “inputs”, such as serial or parallel signal inputs, monitoring interrupt lines, etc.

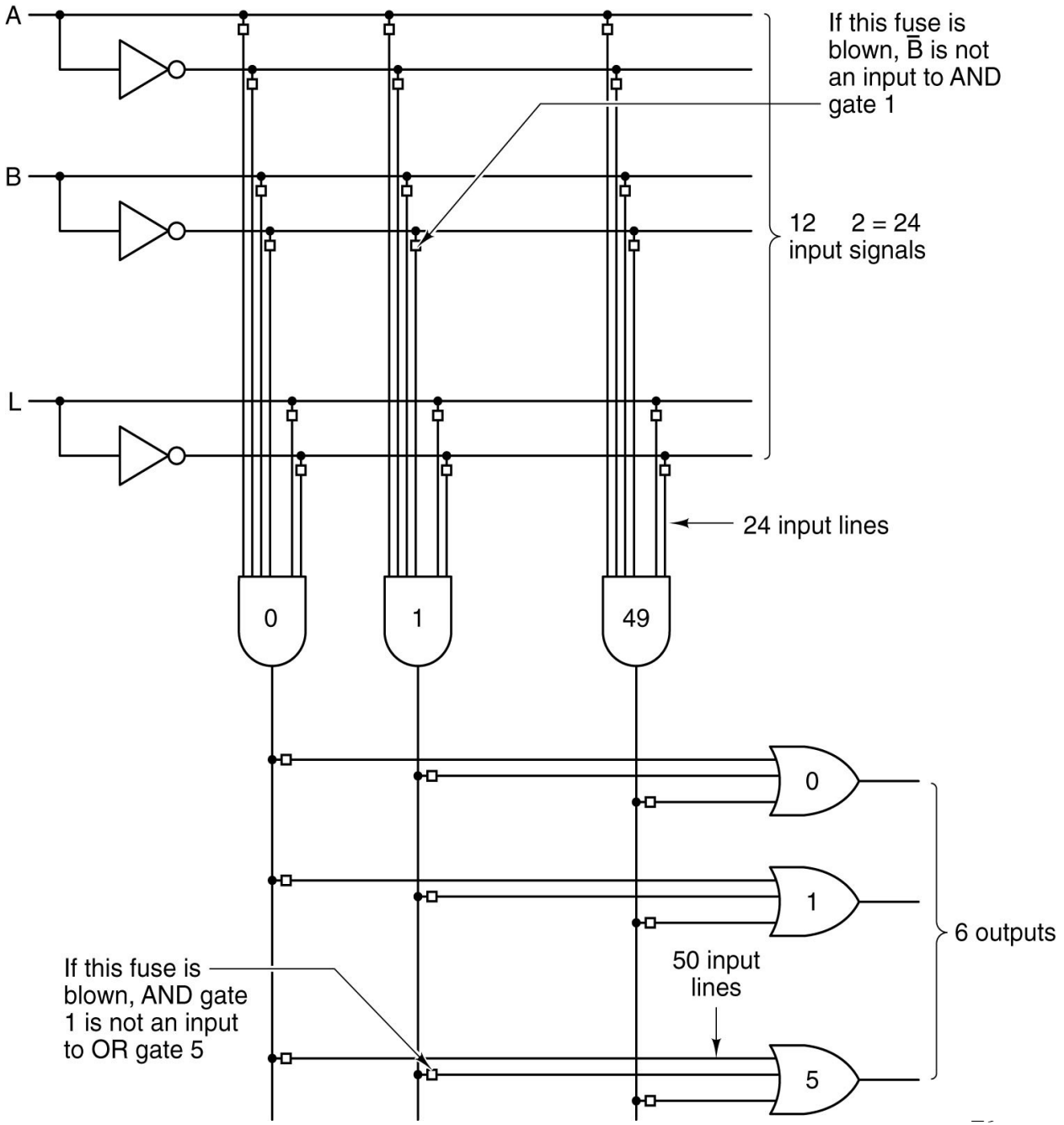
See the TI comparator truth table and logic diagram
74X85 – full magnitude comparator, including equal

# Comparators




A simple 4-bit comparator

# Programmable Logic Arrays



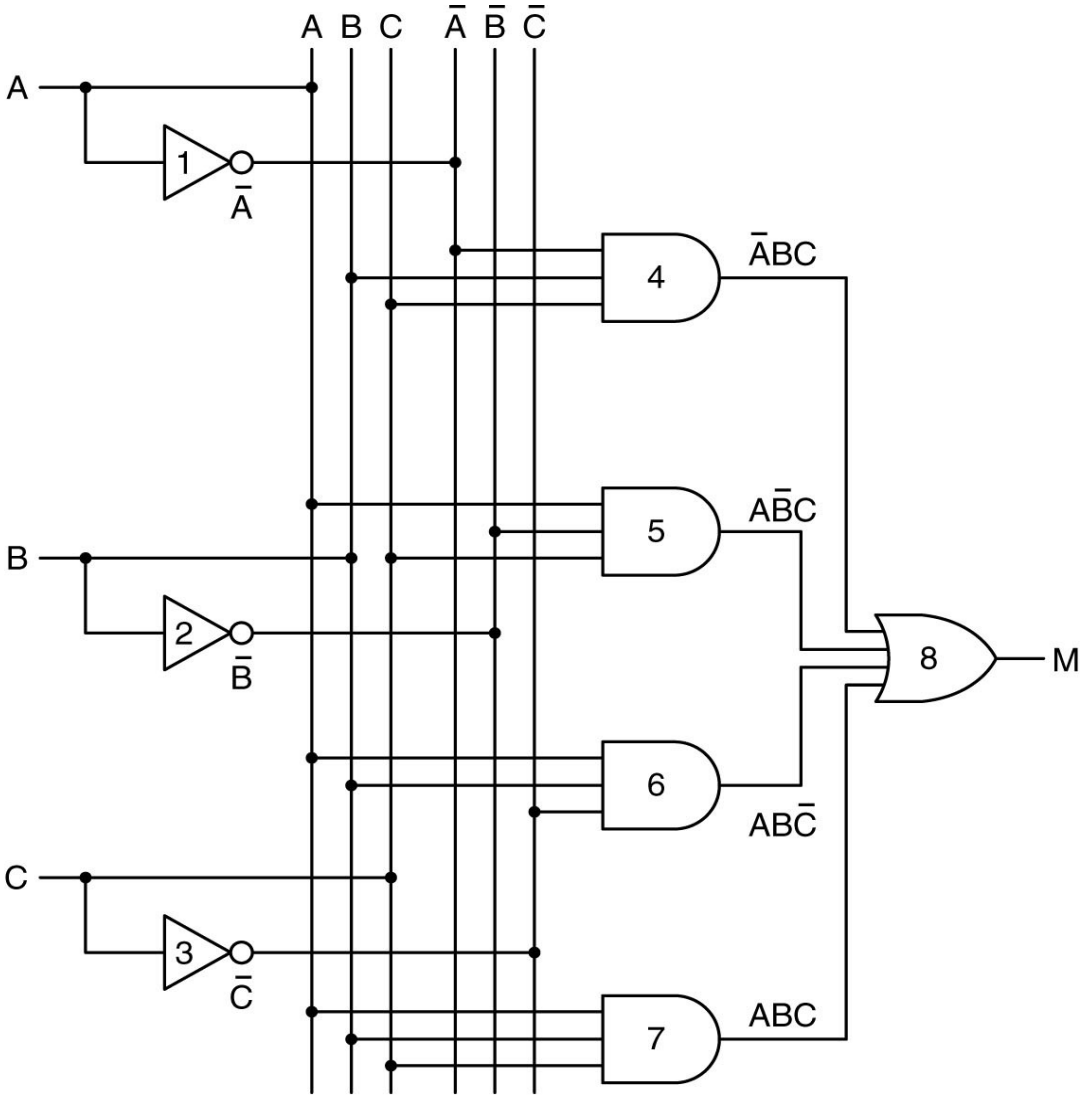

A 12-input, 6-output programmable logic array. The little squares represent fuses that can be burned out.

$$M = A \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot C + A \cdot B \cdot C$$

**RELOOK**

| A | B | C | M |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

(a)




(b)

(a) Truth table for majority function of three variables.
(b) A circuit for (a).

# Programmable Logic Arrays

- INV-AND-OR planes shown for performing basic combinatorial logic functions
- Sufficient (suggested) encoding style for VHDL or Verilog.

TIBPAL22V10 Logic Diagram was (is?) one of the most commonly used programmable devices (see datasheet).

# Problem (*):

Write a program to read in a collection of Boolean expressions and compute the 24 × 50 and 50 × 6 matrices needed to implement them with the PLA of Fig. 3-15. The input language should include single letters, as Boolean variables, the operands AND, OR, and NOT, and parentheses. Print the matrices on the line printer.

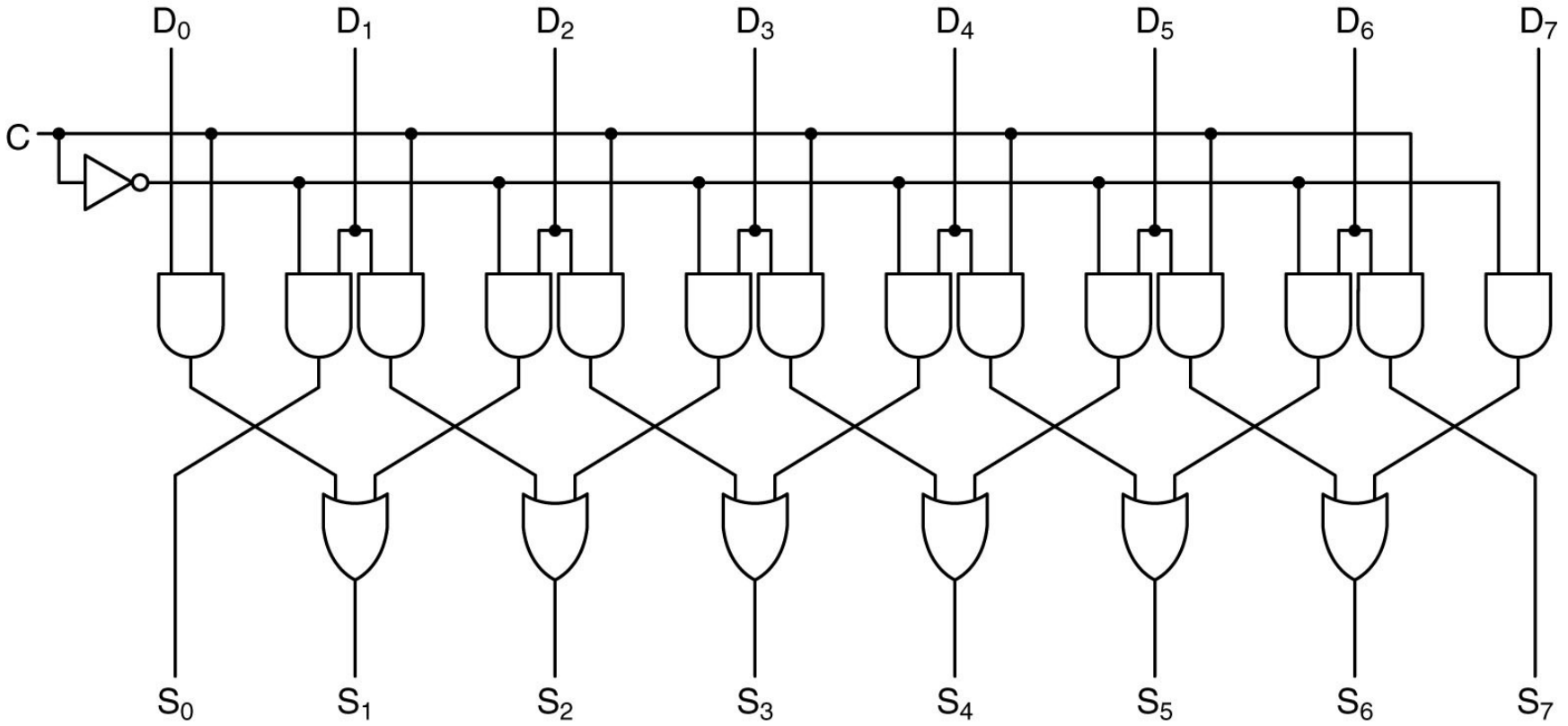
# Important Circuits for Computer Design

- Combinational Circuits
  - Decoders
  - Multiplexers
  - Comparators
  - Programmable Logic Arrays
- Arithmetic Circuits
  - Adders
  - Shifters
  - Arithmetic Logic Units (ALU)
  - Multipliers
- Latches, Flip-Flops and the Clock Glitch Generator
  - Set-Reset Latch
  - “Gated” SR Latch
  - “Gated” D Latch
  - The clock glitch (pulse) generator (only to be used by experts)
  - D Flip Flop
  - Synchronous Register Writing

# Shifter (1)

C=0 for left shift

C=1 for right shift



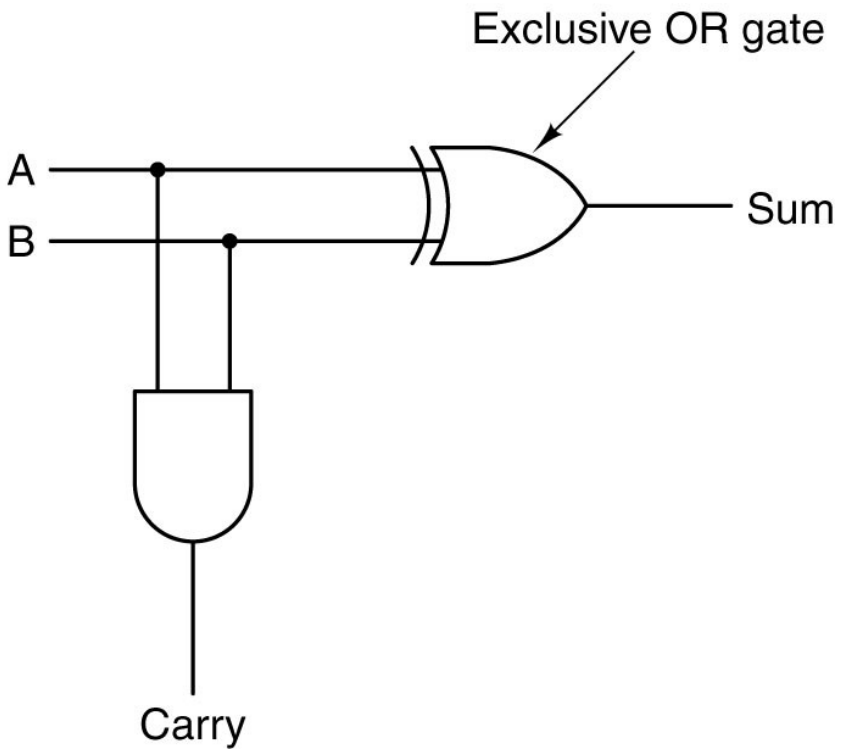

A 1-bit left/right shifter

# Shifter (2)

- Shifter applications:
  - Normalizing and denormalizing floating point number fractional components.
  - Support multipliers using shift and added
  - Multiply by 2^shift (x5 → x4 + x1) (Shifting and adding is sometimes faster than a full multiplier)

# Adder: HA

| A | B | Sum | Carry |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

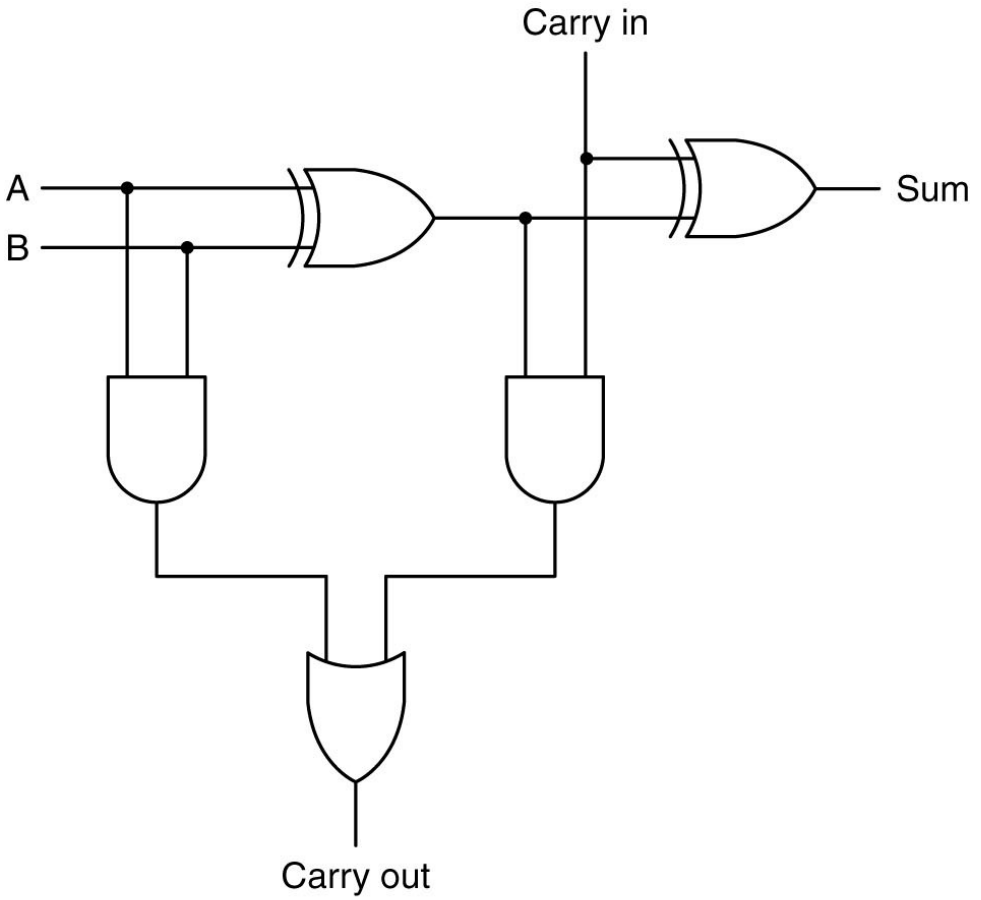



(a) A truth table for 1-bit addition.
(b) A circuit for a half adder **(HA)**.

# Adder: FA

| A | B | Carry in | Sum | Carry out |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

(a)



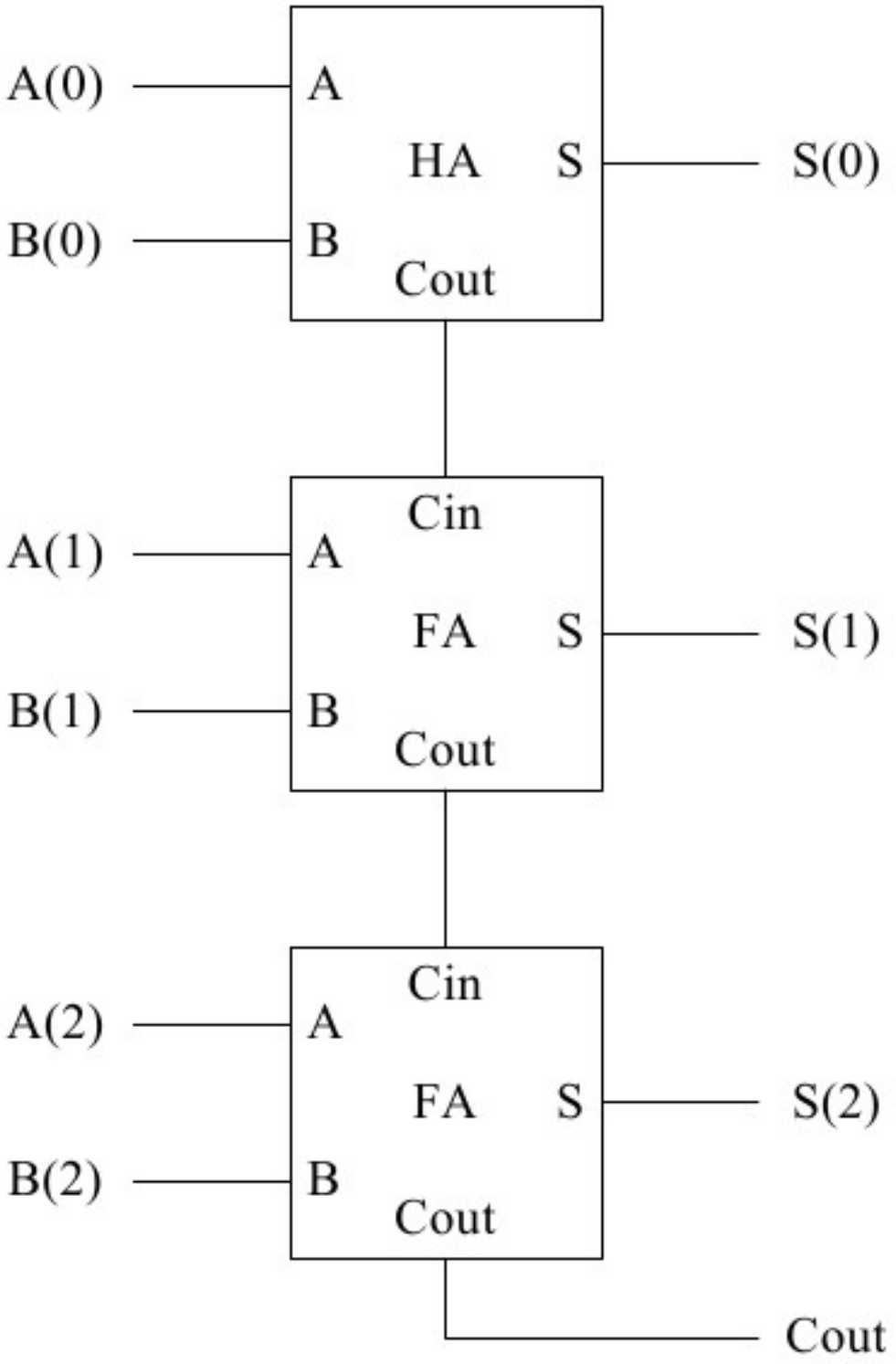

(b)

(a) Truth table for a full adder **(FA)**.
(b) Circuit for a full adder.

Question:

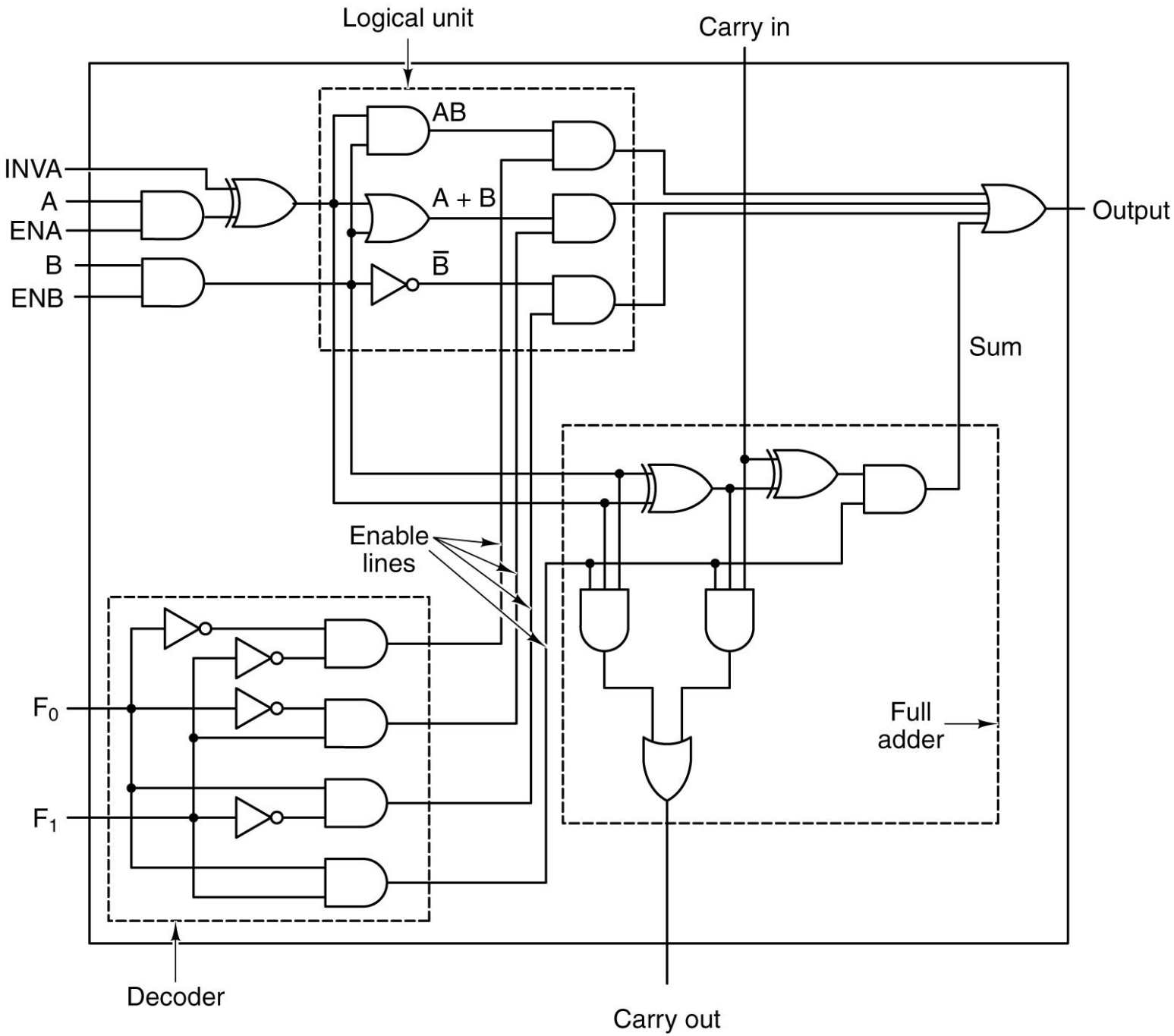
Using HAs and FAs, construct a 2 input - 8 bits adder

# Adder (3)




Ripple Adder Using HA and FA cells (an example of hierarchy)

# Arithmetic Logic Unit




A 1-bit ALU

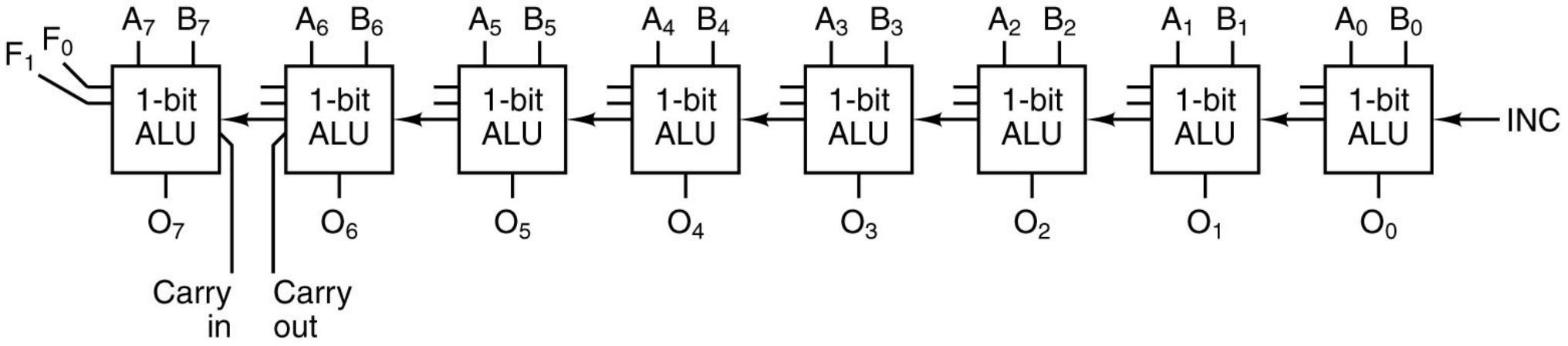
AND, OR, INV, Binary Added with Cin and Cout

Question:

- Why this is called a 1-bit ALU?

- What're the functions of F0, F1?

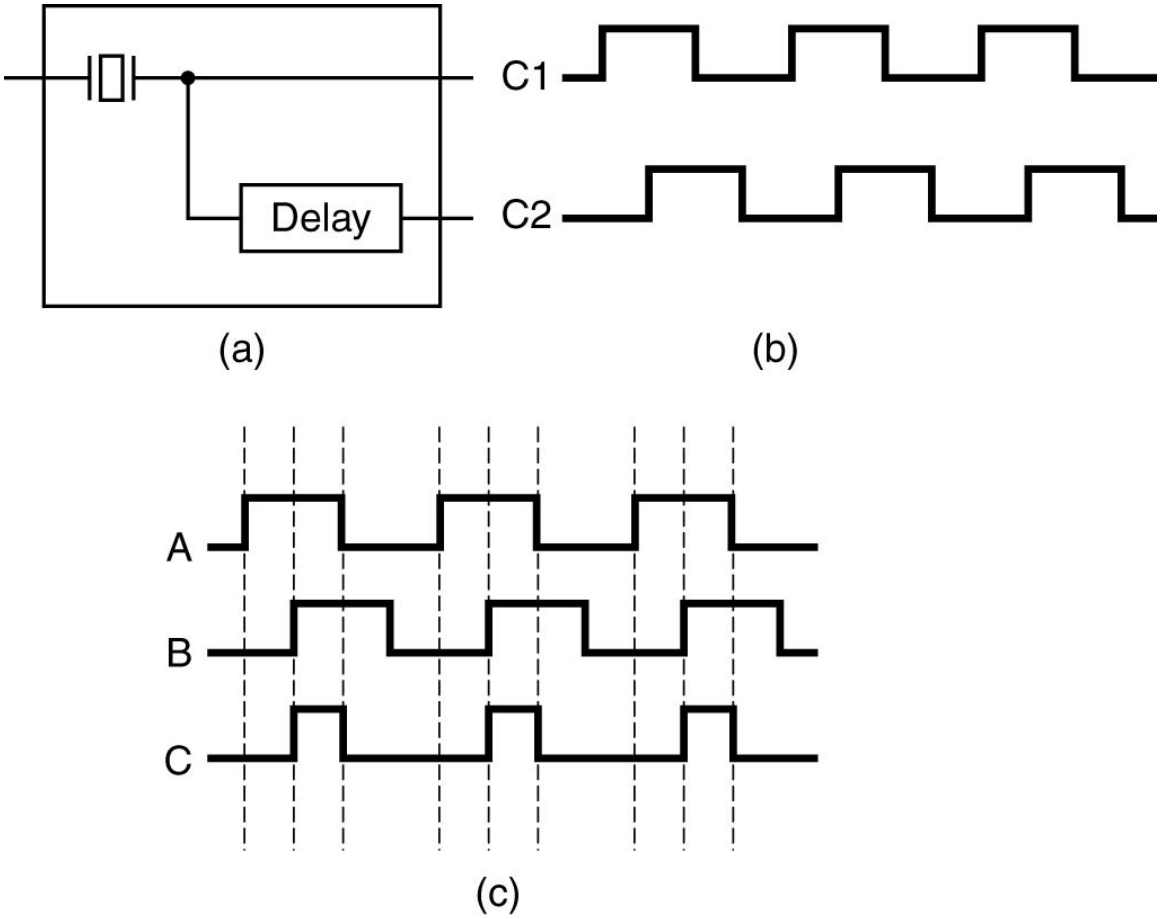
# Arithmetic Logic Unit




Eight 1-bit ALU slices connected to make an 8-bit ALU.
The enables and invert signals are not shown for simplicity.

# Clock

Pulse frequencies are commonly between 1 and 500MHz

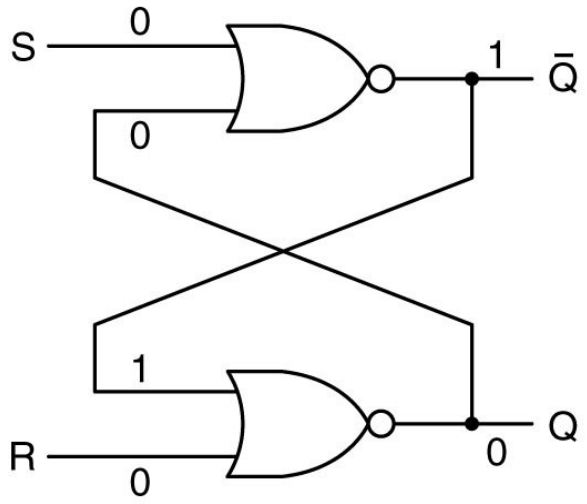



(a) A clock.
(b) The timing diagram for the clock.
(c) Generation of an asymmetric clock.

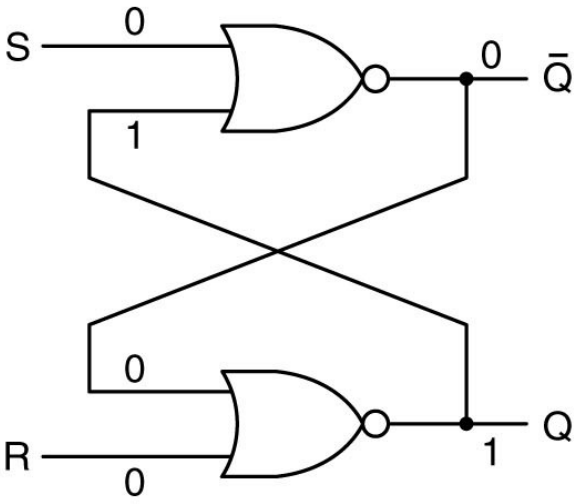
# Important Circuits for Computer Design

- Combinational Circuits
  - Decoders
  - Multiplexers
  - Comparators
  - Programmable Logic Arrays
- Arithmetic Circuits
  - Adders
  - Shifters
  - Arithmetic Logic Units (ALU)
  - Multipliers
- Latches, Flip-Flops and the Clock Glitch Generator
  - Set-Reset Latch
  - “Gated” SR Latch
  - “Gated” D Latch
  - The clock glitch (pulse) generator (only to be used by experts)
  - D Flip Flop
  - Synchronous Register Writing

# Latch (1)




(a)



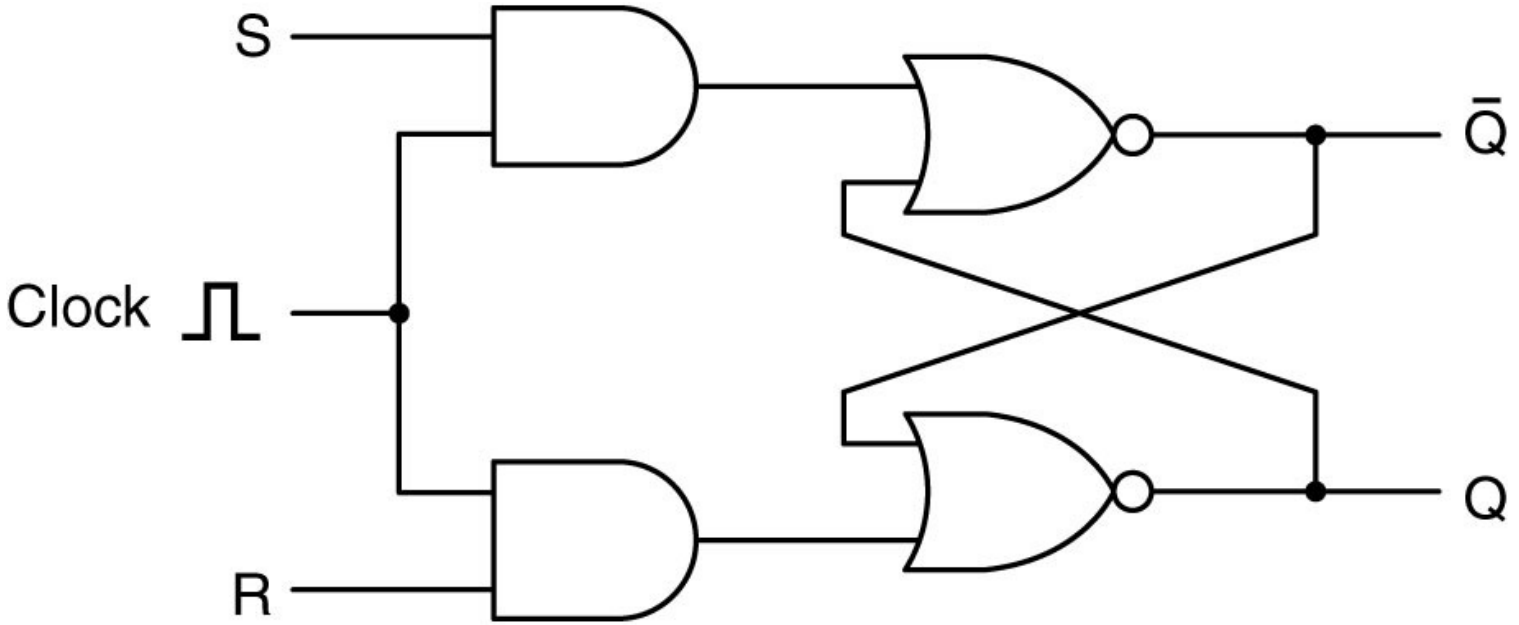

(b)

| A | B | NOR |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

(c)

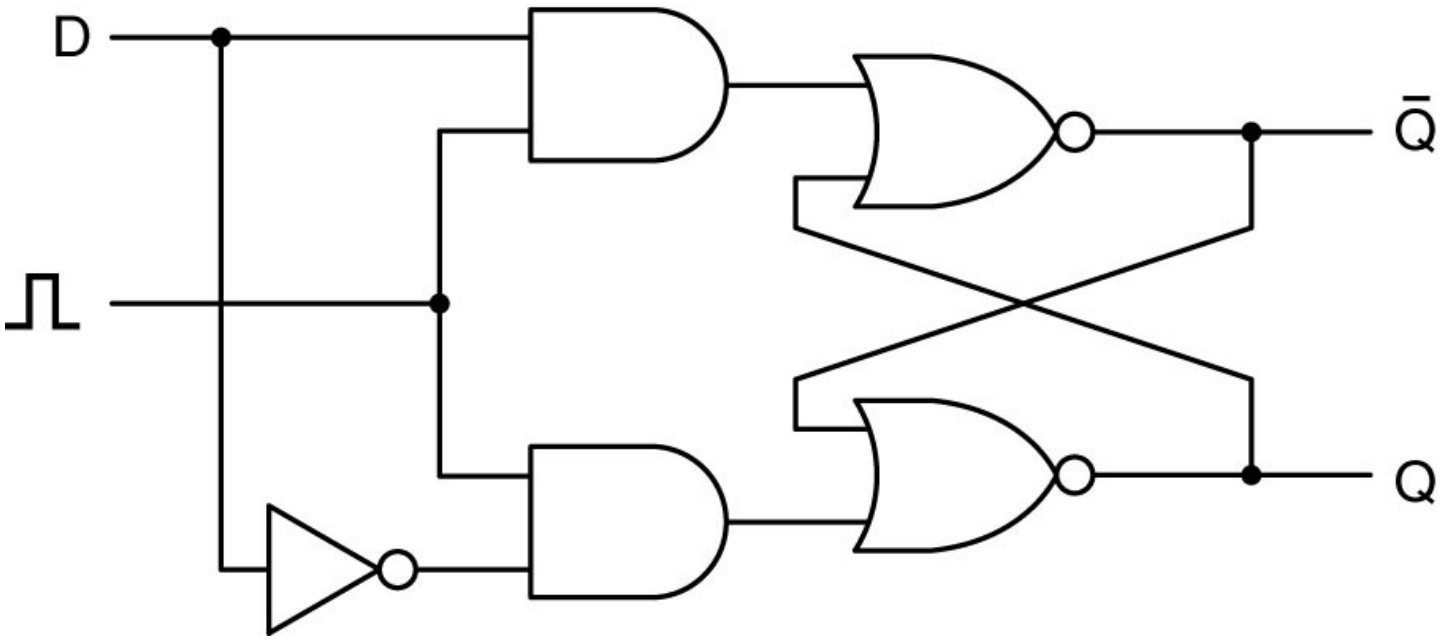
(a) NOR latch in state 0.
(b) NOR latch in state 1.
(c) Truth table for NOR.

# Latch (2)



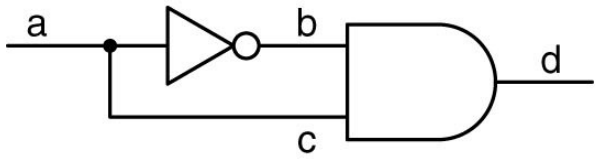

A clocked SR latch.

# Latch (3)

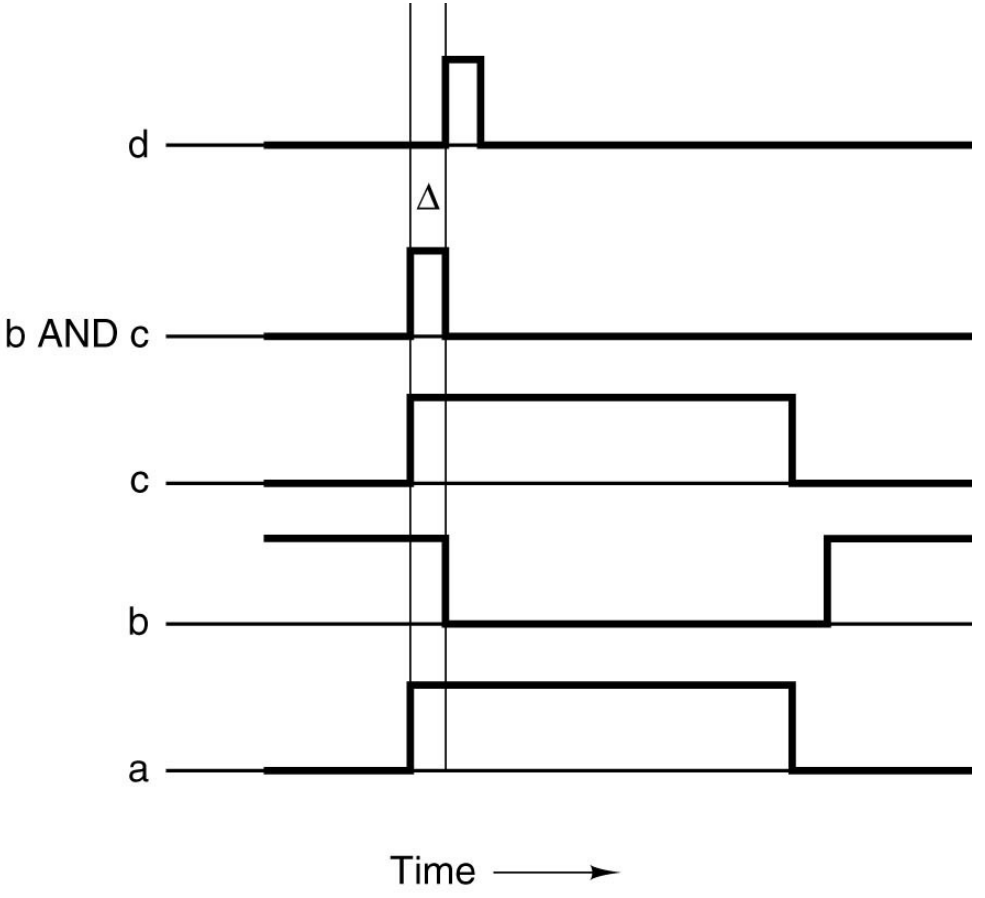



A clocked D latch

# Flip-flop (1)



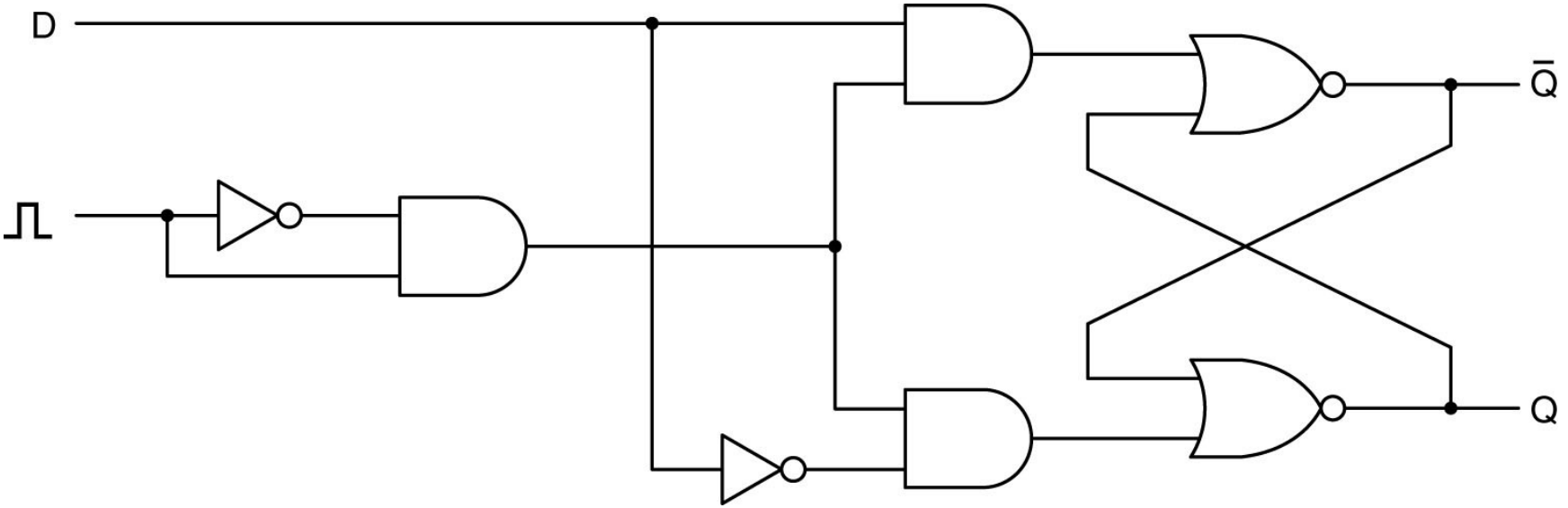

(a)




(b)

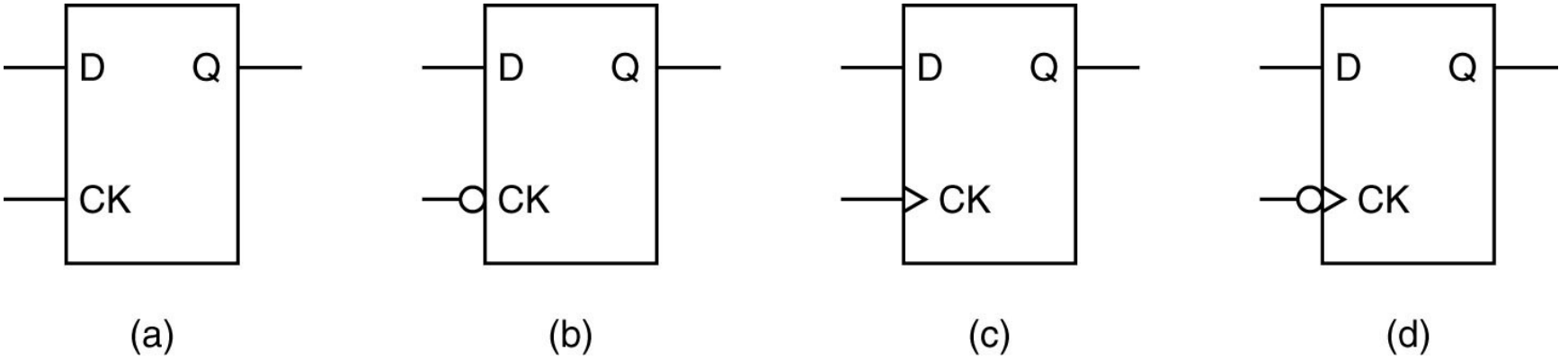
(a) A pulse generator.
(b) Timing at four points in the circuit.

# Flip-flop (2)




A D flip-flop

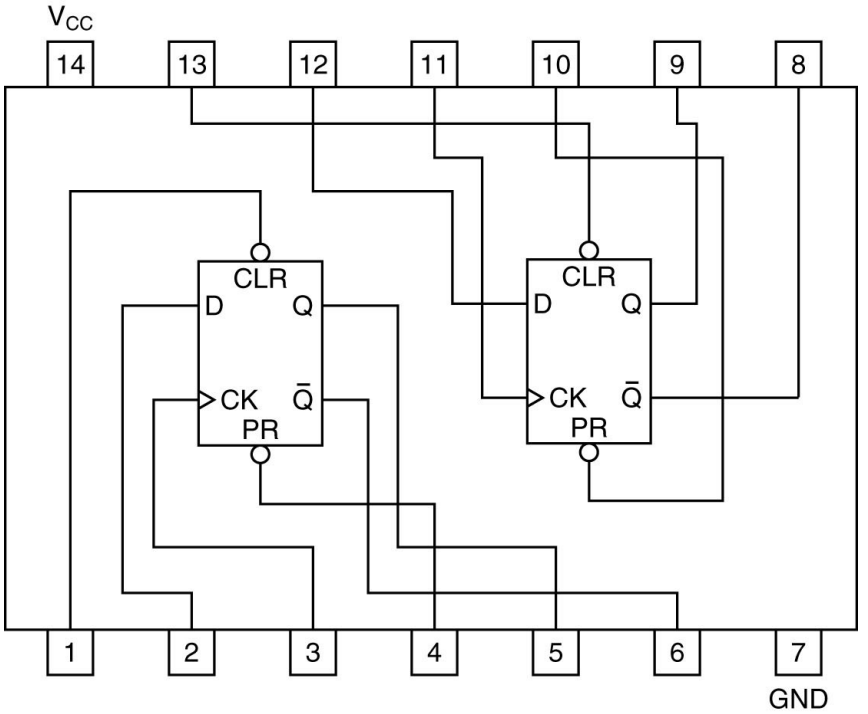
# Flip-flop (3)



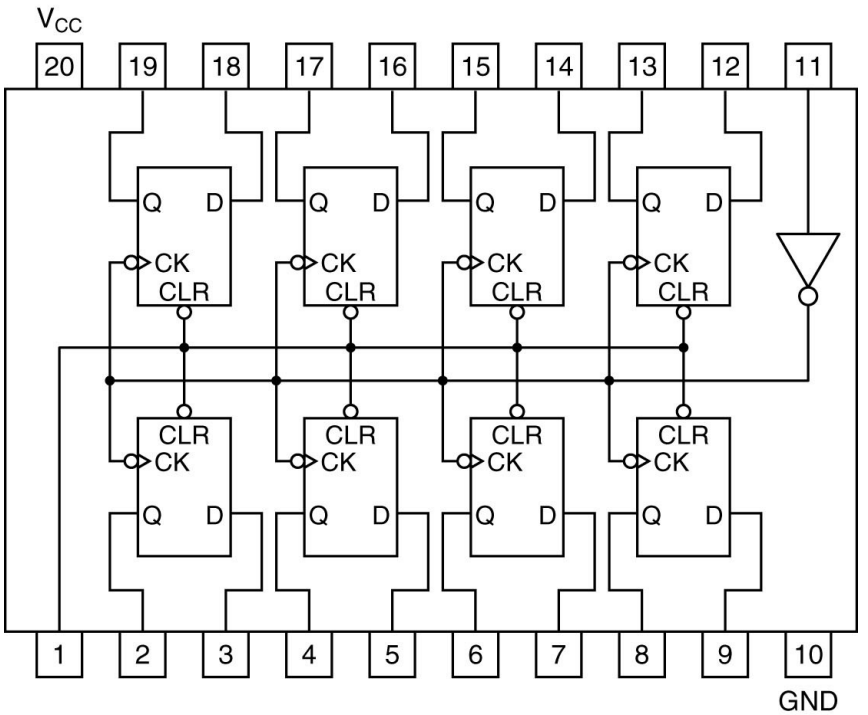

D latches and flip-flops.

# Flip-flop (4)

a) Octal flip-flop.
b) Dual D flip-flop




(a)

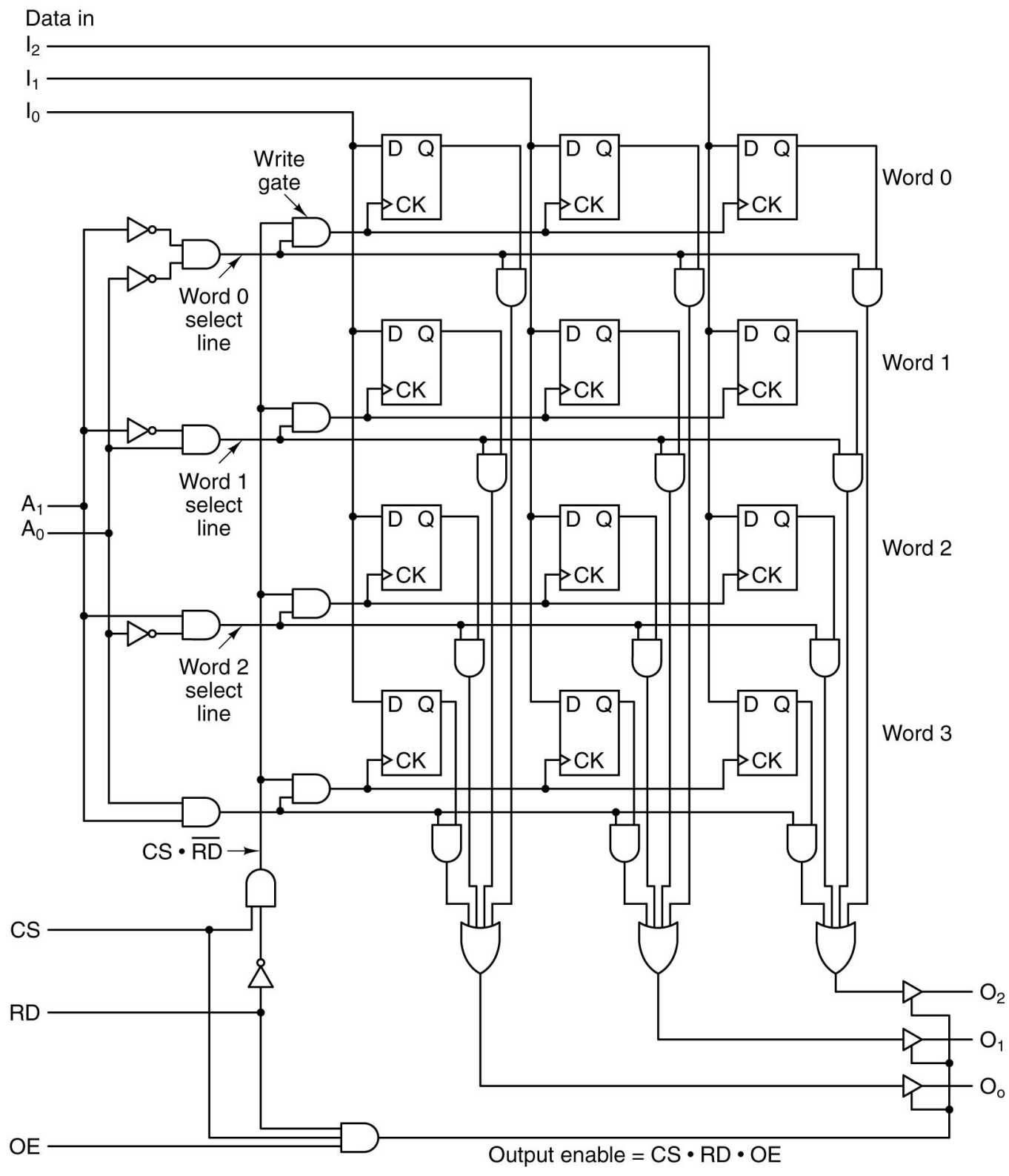



(b)

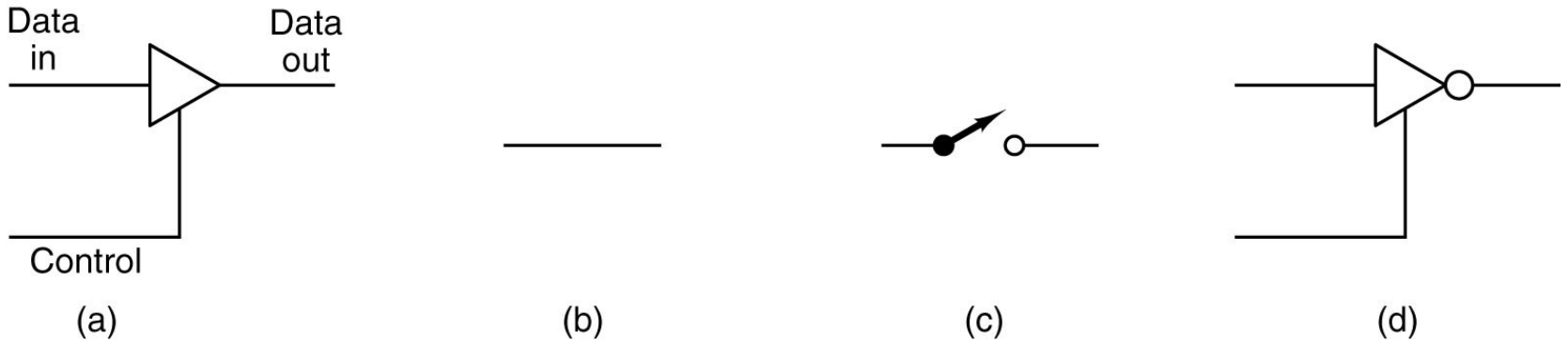
# Memory Organization (1)

Logic diagram for a 4 x 3 memory.

Each row is one of the four 3-bit words.



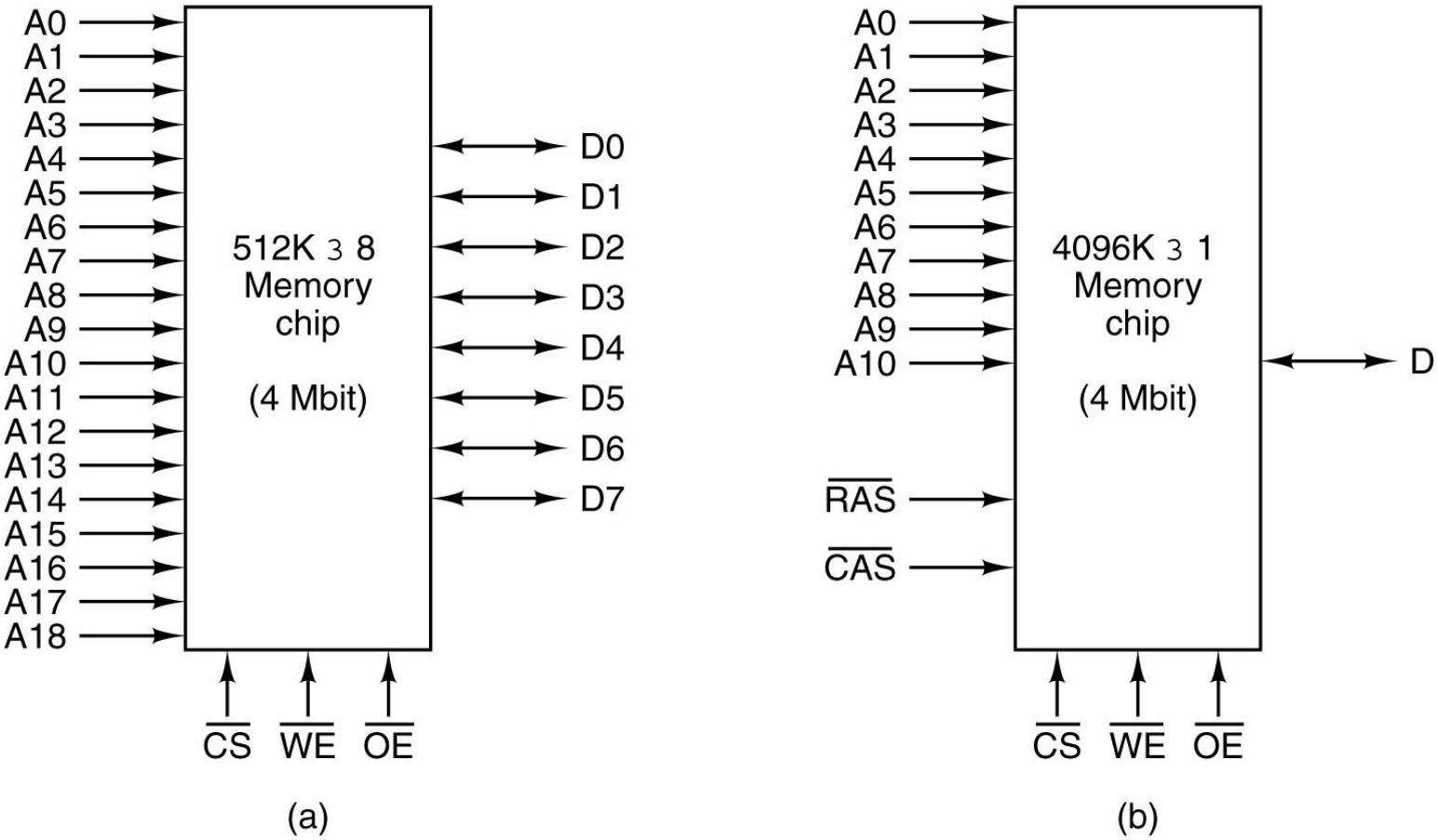
# Memory Organization (2)




(a) A noninverting buffer.
(b) Effect of (a) when control is high.
(c) Effect of (a) when control is low.
(d) An inverting buffer.

# Memory Chip (1)

RAS: row address strobe
CAS: column address strobe
1 bit data → memory chips in parallel

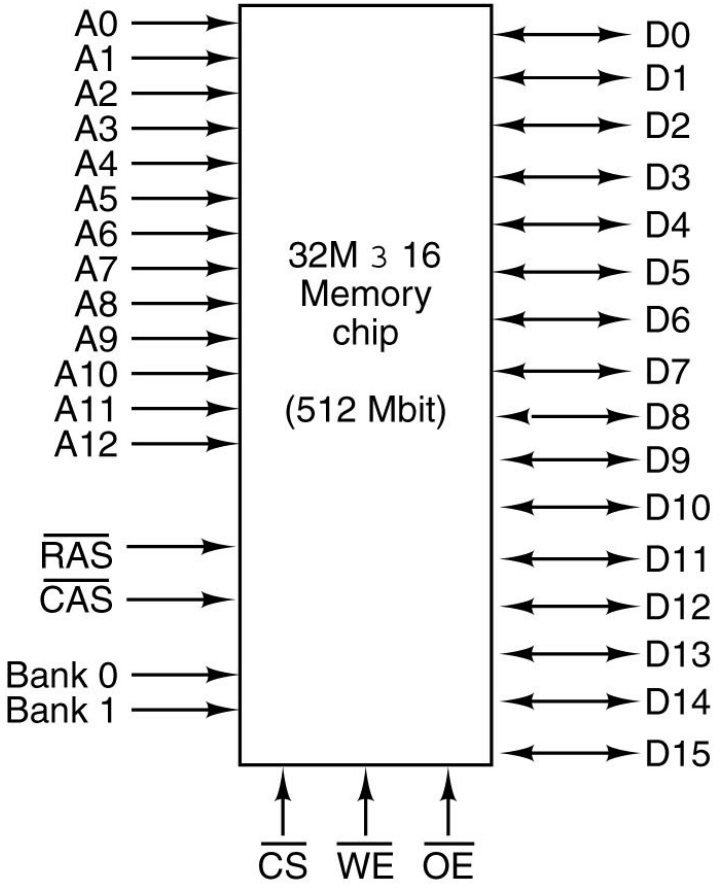


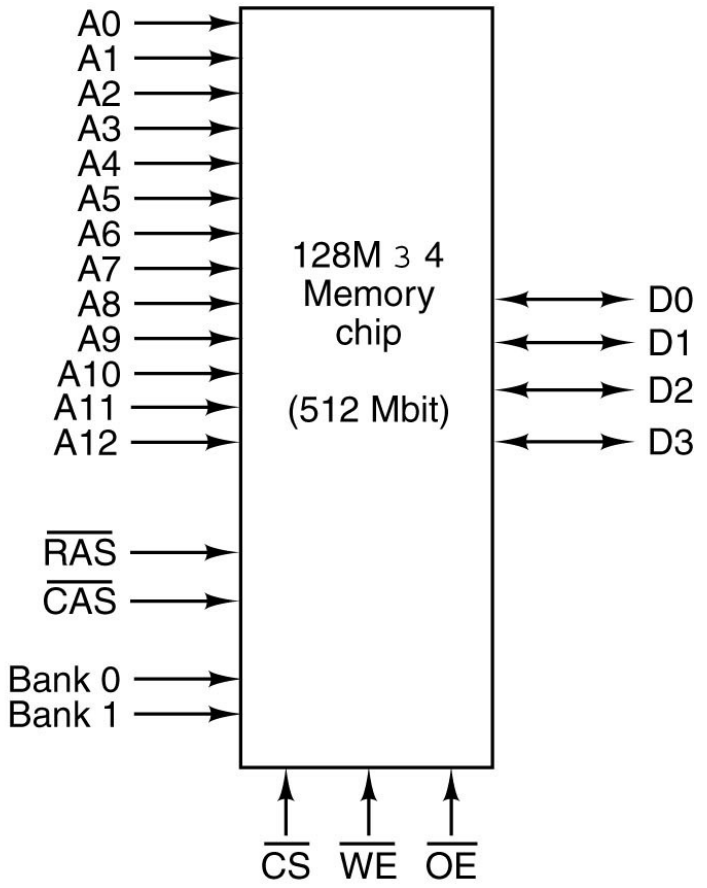

Two ways of organizing a 4-Mbit memory chip

# Memory Chip (2)

13 lines for $\overline{\text{RAS}}$ signal
10 lines for $\overline{\text{CAS}}$ signal
2 lines for bank select

13 lines for $\overline{\text{RAS}}$ signal
12 lines for $\overline{\text{CAS}}$ signal
2 lines for bank select

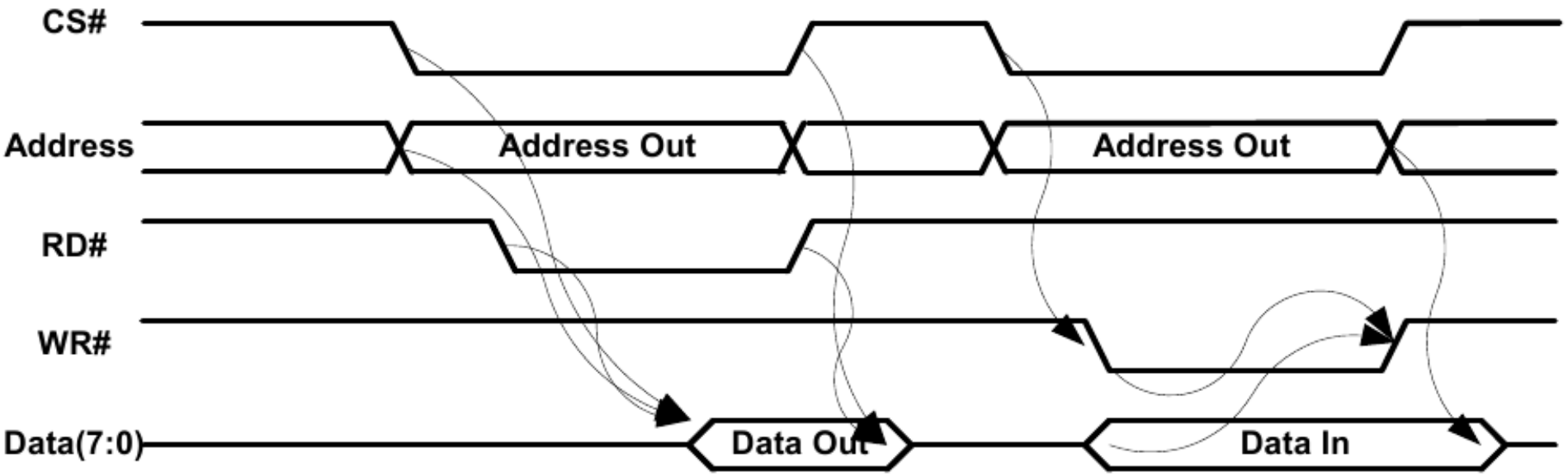





Two ways of organizing a 512 Mbit memory chip.

# Asynchronous SRAM Memory Timing

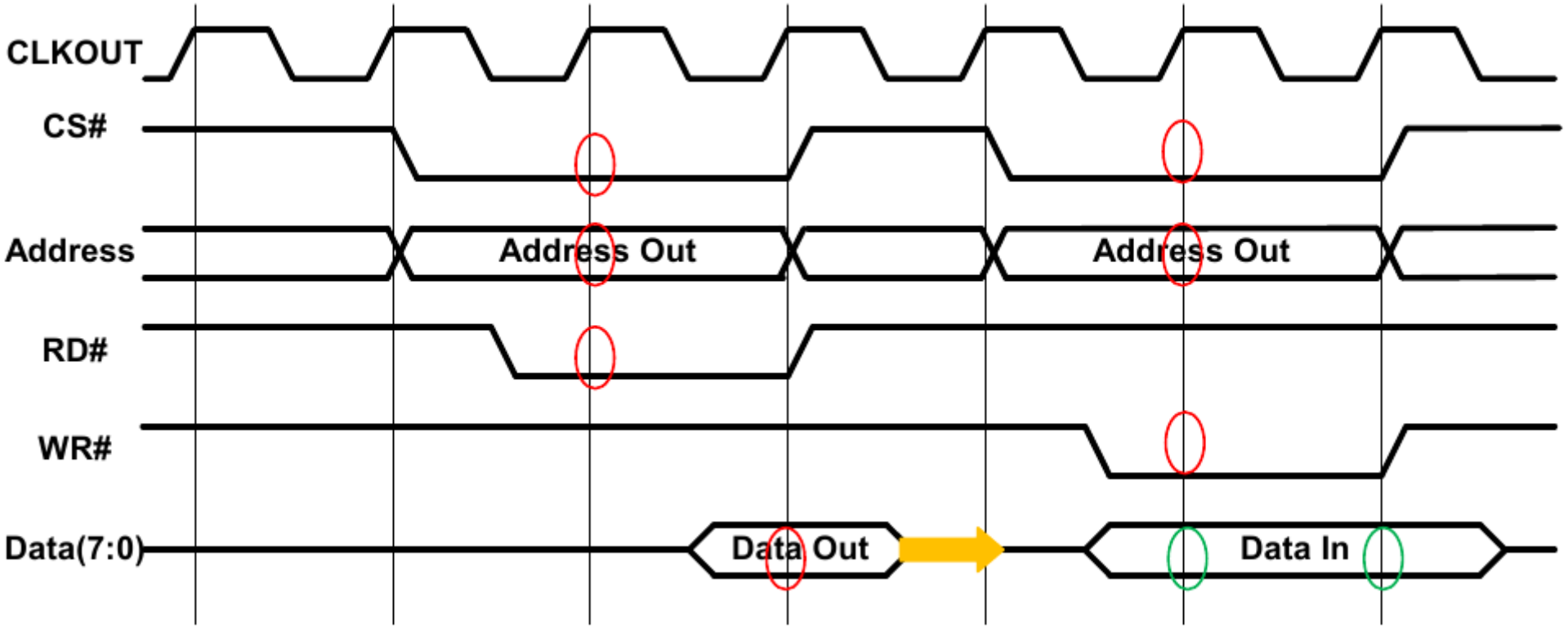



Simple Asynchronous Memory Operation (Memory Perspective)
Note: X# indicates Not(X), the inverse of X

# Asynchronous SRAM Memory Timing

- (1) Chip select required to “turn the memory on”
- (1) Address is required to select word or bit (before write, preferably before a read)
- (2) Read (RD#) and Write (WR#) gates specifically defined or a single (RD/WR#)
- (2 or 3) Data bus is bi-directional ... both input and output (using internal tri-state output enables)

# Synchronous SRAM Memory Timing




Same operations may be asynchronous, but internal control signals are registered for application on clock edges.

# Synchronous SRAM Memory Timing

- Control changes after clock edges (change caused by clocking, positive or negative edge)
- Operations typically occur one clock cycle after “control is clocked”
- When outputting, the output may tri-state asynchronously (when RE# is removed) to avoid multiple devices simultaneously driving the bus.
- Wait states for read/write may be need due to speed of CPU vs. memory?

# Nonvolatile Memory Chips

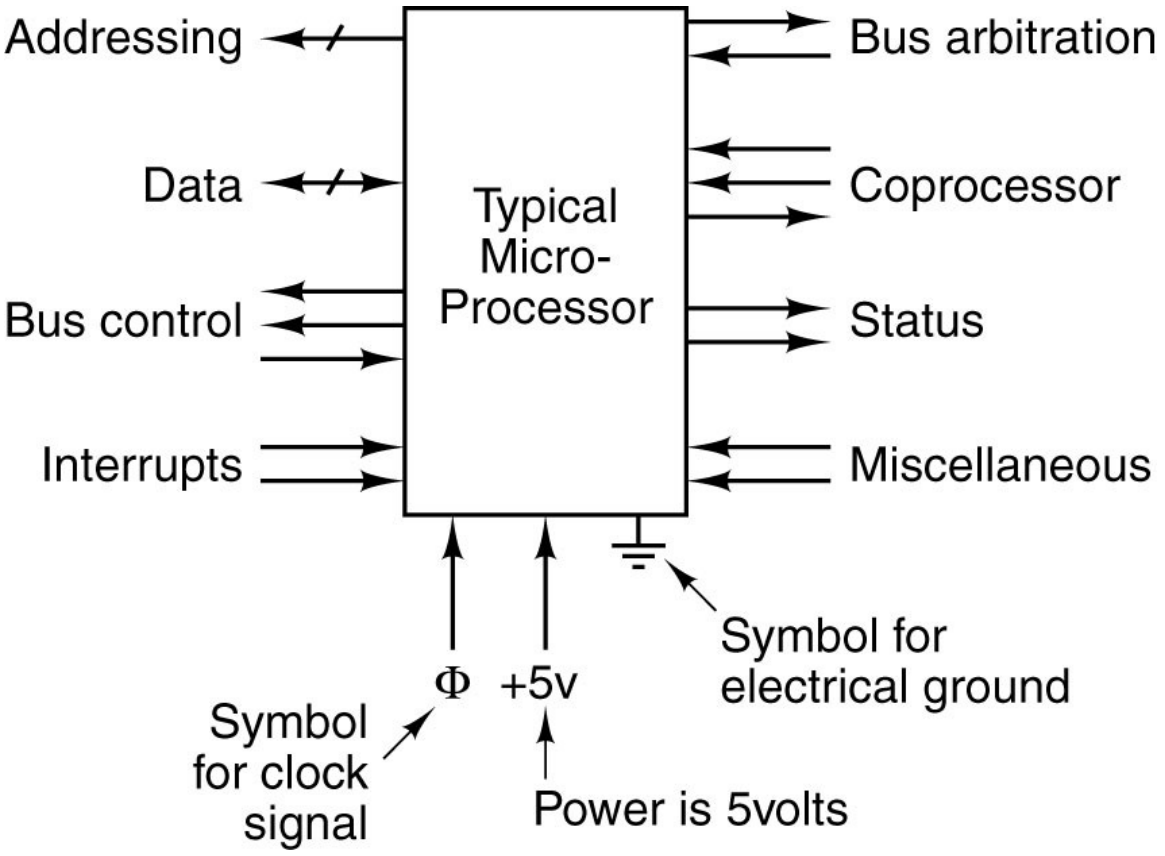
| Type | Category | Erasure | Byte alterable | Volatile | Typical use |
|---|---|---|---|---|---|
| SRAM | Read/write | Electrical | Yes | Yes | Level 2 cache |
| DRAM | Read/write | Electrical | Yes | Yes | Main memory (old) |
| SDRAM | Read/write | Electrical | Yes | Yes | Main memory (new) |
| ROM | Read-only | Not possible | No | No | Large volume appliances |
| PROM | Read-only | Not possible | No | No | Small volume equipment |
| EPROM | Read-mostly | UV light | No | No | Device prototyping |
| EEPROM | Read-mostly | Electrical | Yes | No | Device prototyping |
| Flash | Read/write | Electrical | No | No | Film for digital camera |

A comparison of various memory types.

# Outline

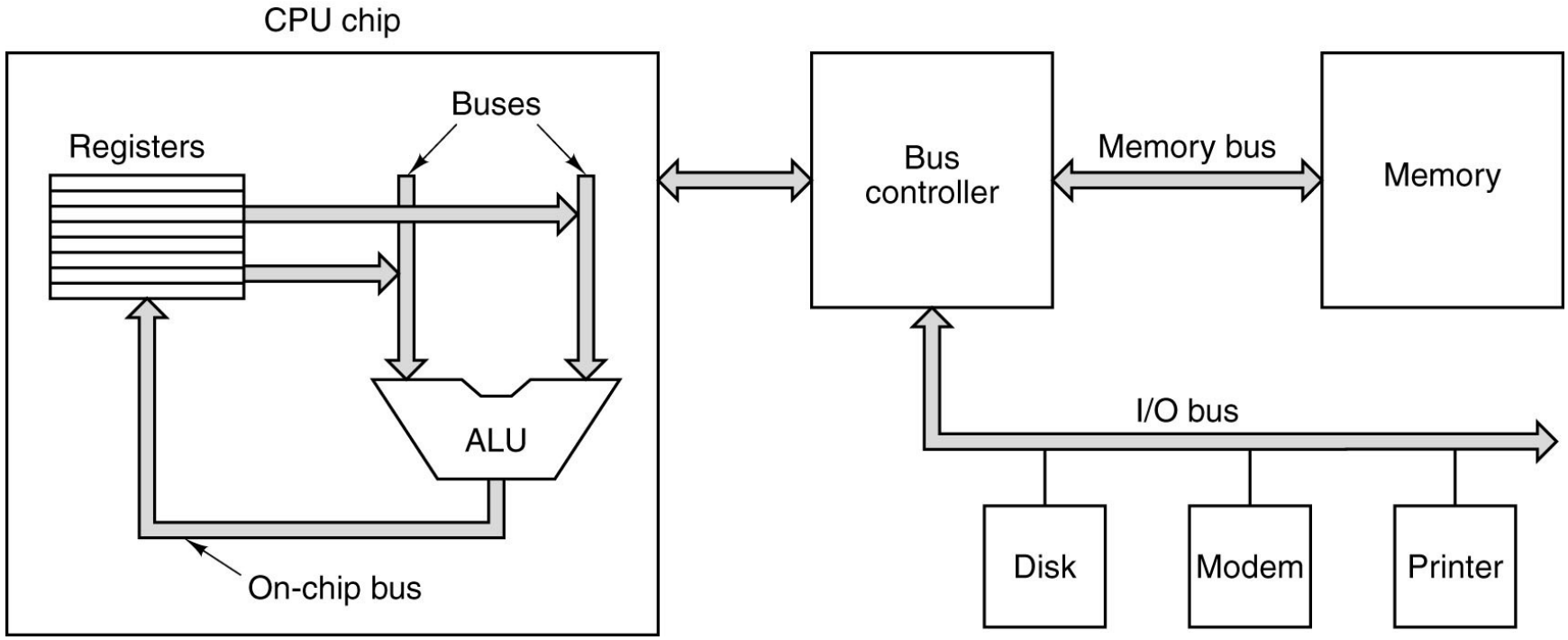
- Gates And Boolean Algebra
- Basic Digital Logic Circuits
- Memory
- CPU Chips And Buses
- Example CPU Chips
- Example Buses

# CPU Chip




Memory and I/O Addressing → address, data, bus control
Bus master/slave control → bus arbitration
Interrupts and Status
Special signals → coprocessor interfaces or miscellaneous (JTAG, IIC, debug, etc.)

# Computer Buses (1)




Computers use numerous buses at multiple levels.
- On-chip buses for registers, execution unit operands, instructions, etc.
- Off-chip buses for memory, peripherals, interrupts, status, etc.

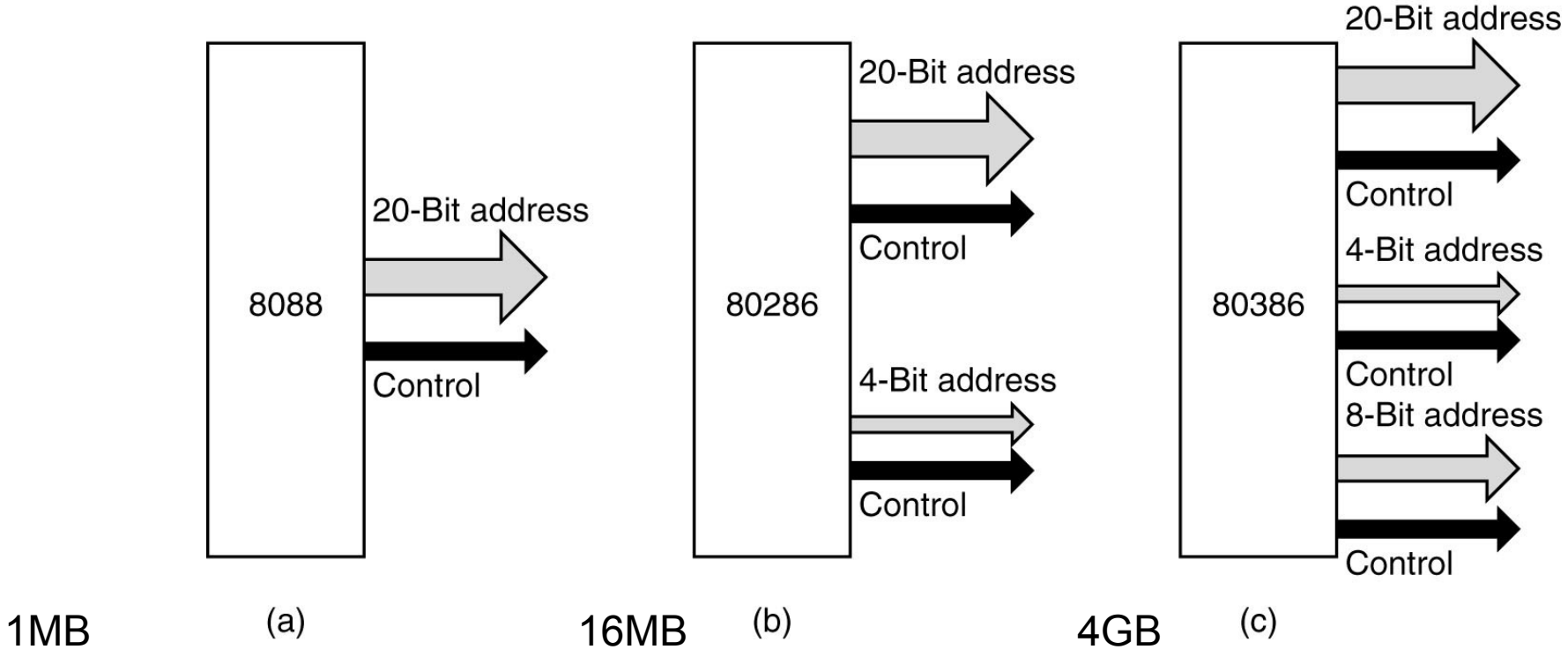
# Computer Buses (2)

| Master | Slave | Example |
|---|---|---|
| CPU | Memory | Fetching instructions and data |
| CPU | I/O device | Initiating data transfer |
| CPU | Coprocessor | CPU handing instruction off to coprocessor |
| I/O | Memory | DMA (Direct Memory Access) |
| Coprocessor | CPU | Coprocessor fetching operands from CPU |

Examples of bus masters and slaves
Master: can initiate bus transfer
Slave: wait for request

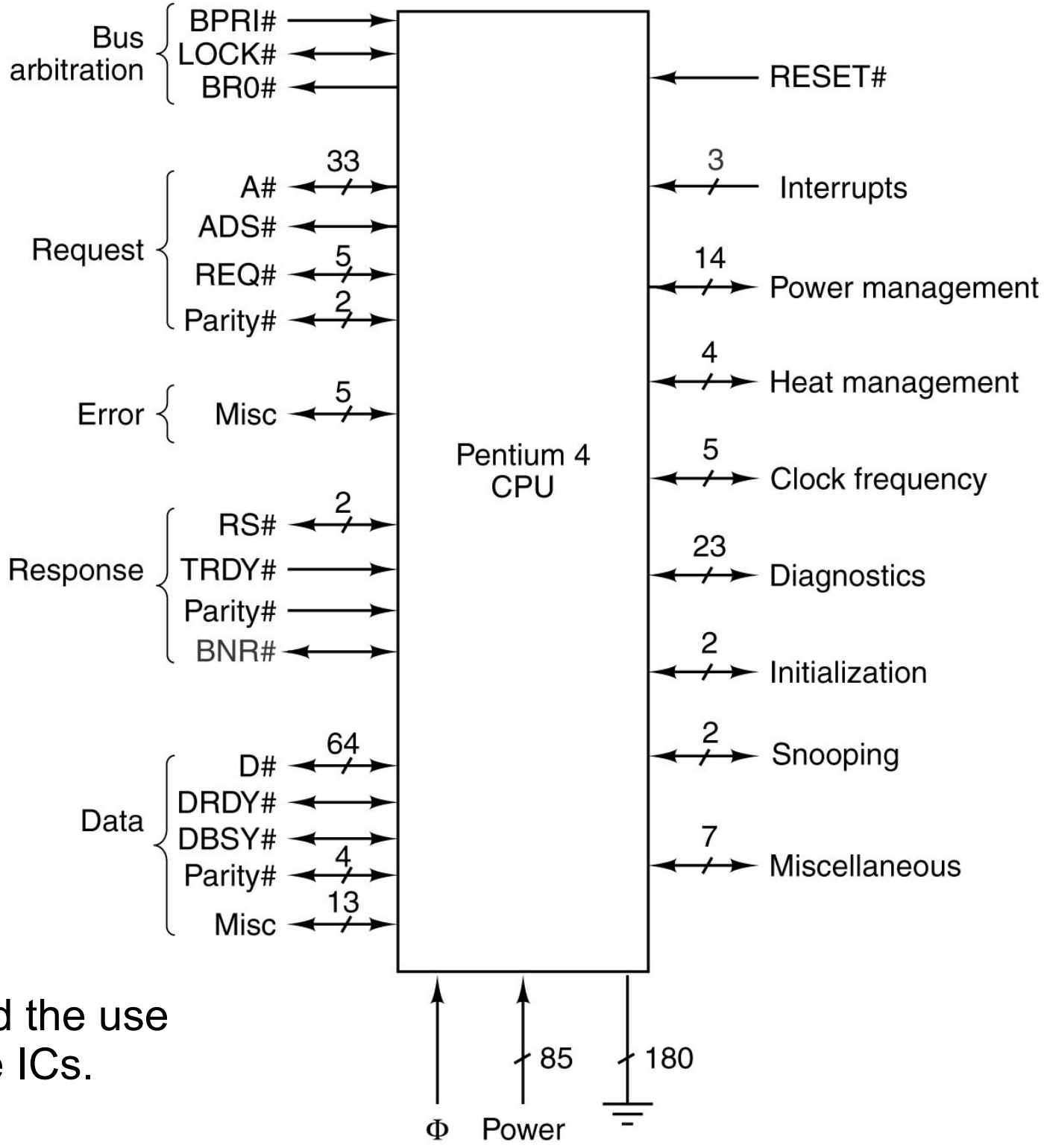
Under no circumstances can memory ever be a master

# Buses Width in Time



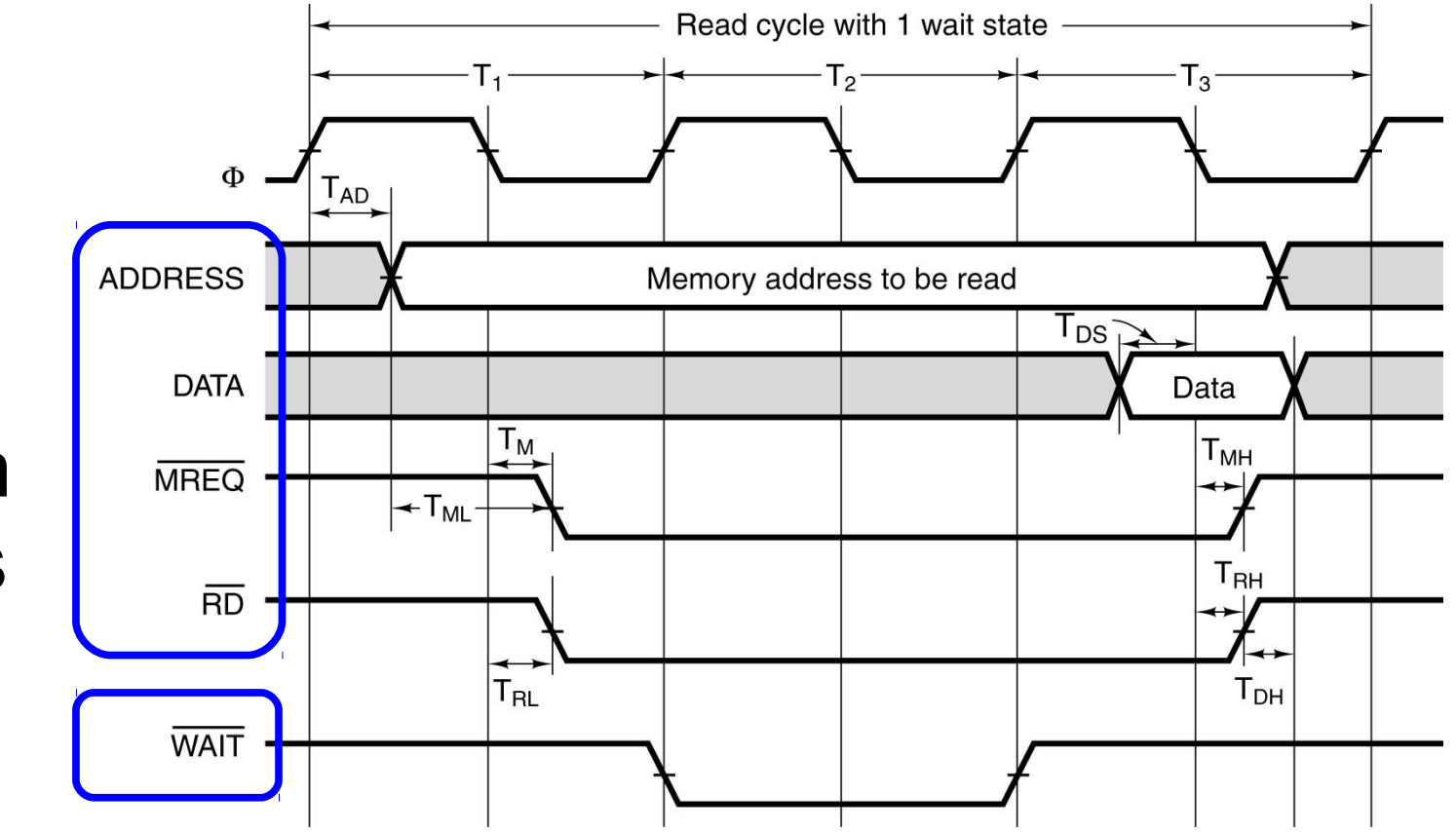

(a) 8088 generation 20A, (b) 80286 generation 24A, (c) 8036 generation 32A

# CPU Buses
# Pentium 4




Note: Pentium 4's started the use of PCI and DRAM bridge ICs.

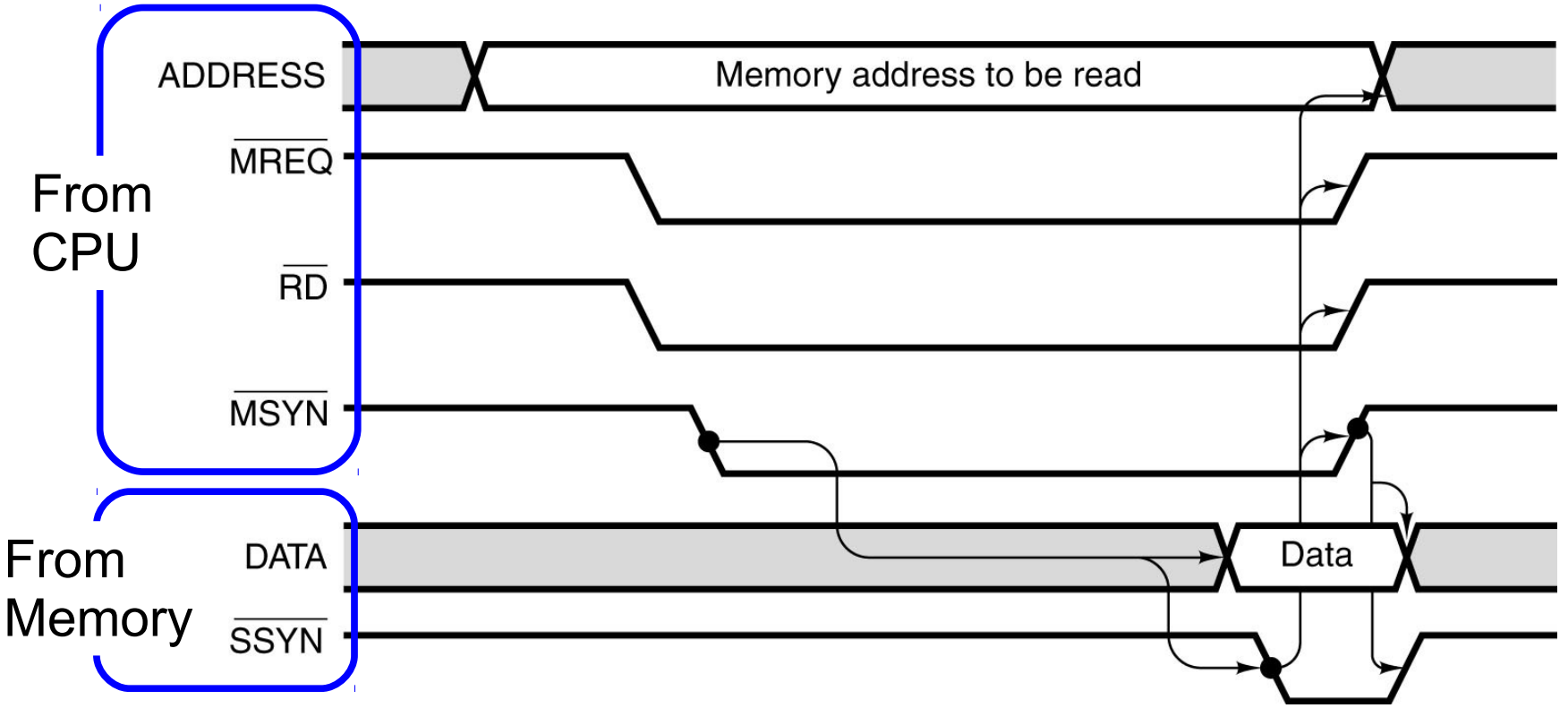
# Synchronous CPU Bus Timing with an Asynchronous Memory




| Symbol | Parameter | Min | Max | Unit |
|---|---|---|---|---|
| $T_{AD}$ | Address output delay | | 4 | nsec |
| $T_{ML}$ | Address stable prior to $\overline{MREQ}$ | 2 | | nsec |
| $T_M$ | $\overline{MREQ}$ delay from falling edge of Φ in $T_1$ | | 3 | nsec |
| $T_{RL}$ | RD delay from falling edge of Φ in $T_1$ | | 3 | nsec |
| $T_{DS}$ | Data setup time prior to falling edge of Φ | 2 | | nsec |
| $T_{MH}$ | $\overline{MREQ}$ delay from falling edge of Φ in $T_3$ | | 3 | nsec |
| $T_{RH}$ | $\overline{RD}$ delay from falling edge of Φ in $T_3$ | | 3 | nsec |
| $T_{DH}$ | Data hold time from negation of $\overline{RD}$ | 0 | | nsec |

(b)

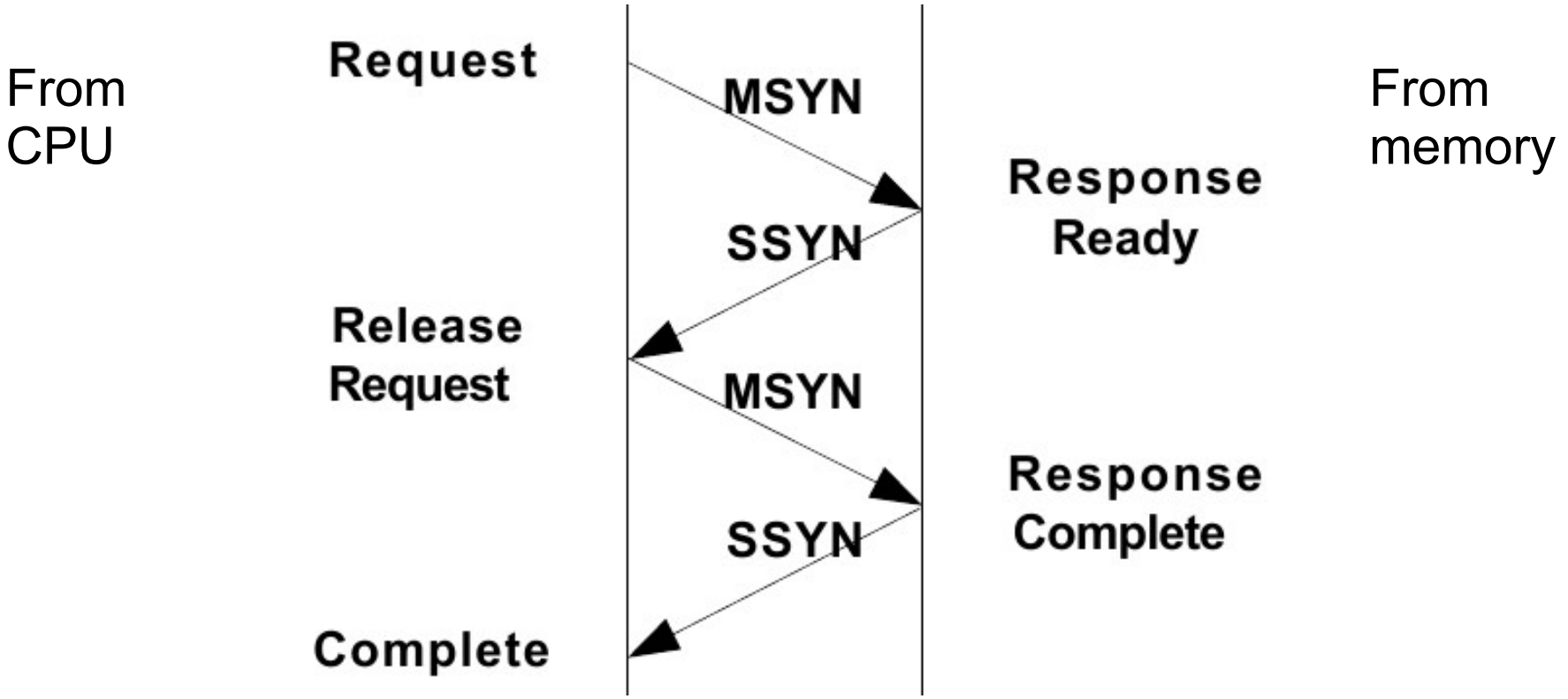
# Asynchronous CPU Bus Timing




Asynchronous Bus Read Cycle with hardware handshaking.
Memory Timing Considerations:

- Address
- Read
- Write
- Handshaking

# Handshaking and Messaging Protocol



# Bus Arbitration

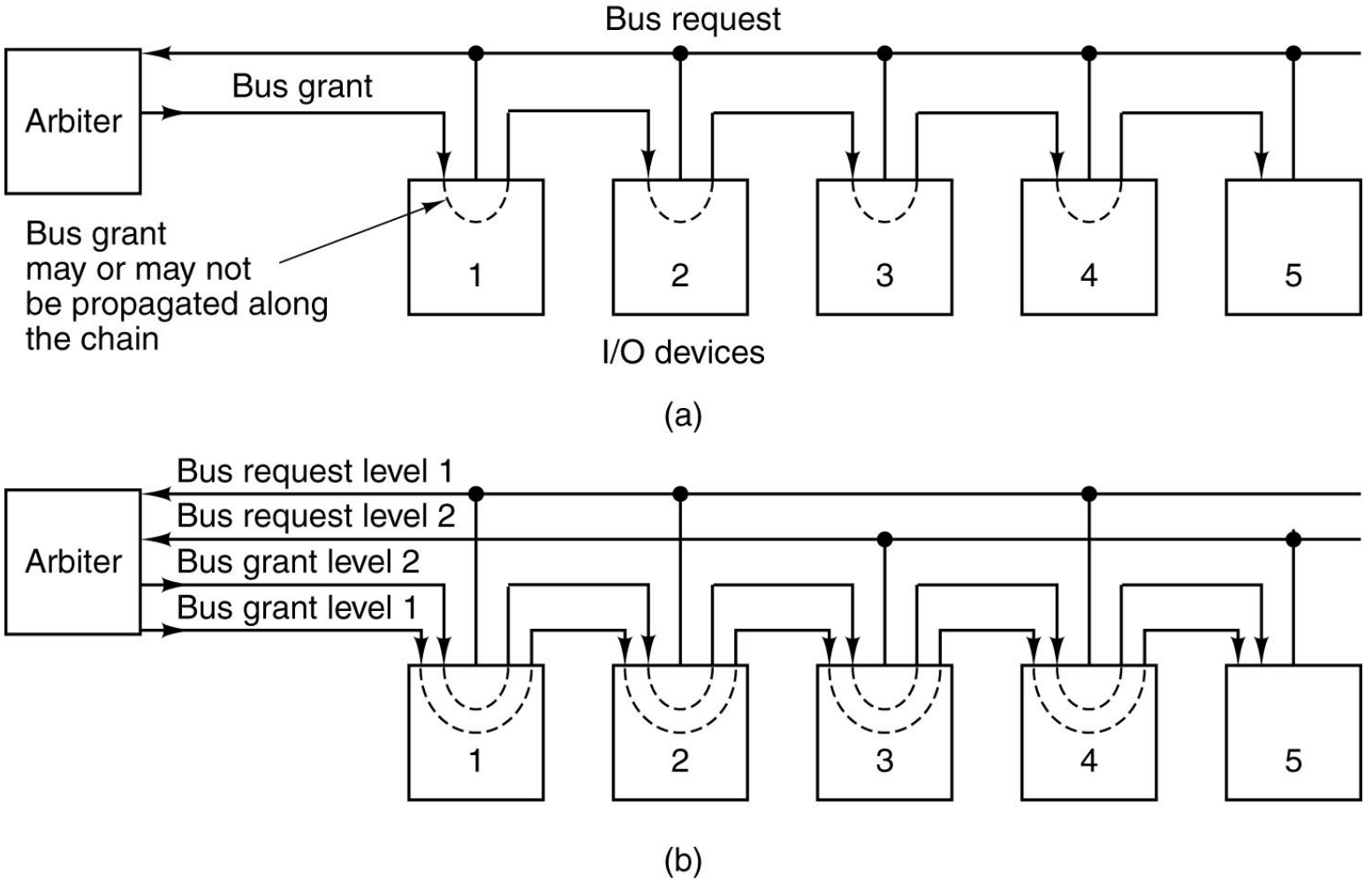
- What happens if two or more devices all want to become bus master at the same time?

→ bus arbitration

  - Centralized
  - Decentralized

# Bus Arbitration – centralized

(with priority)

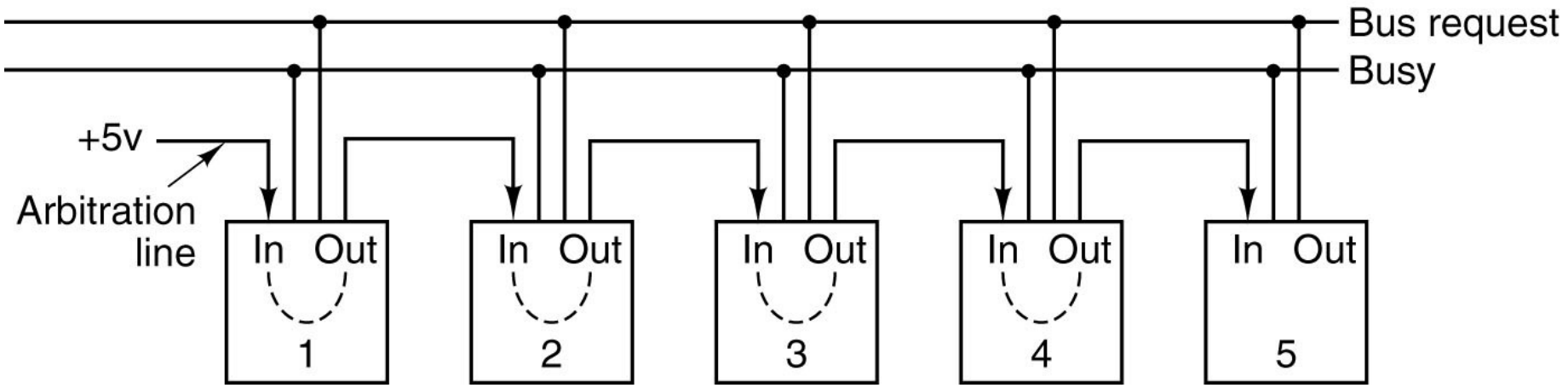



(a) A centralized one-level bus arbiter using **daisy chaining.**
(b) The same arbiter, but with two levels.

# Daisy chain

- Bus Request:
  - Always high, pulled down to ground when I/O devices wants to make a request
- Bus Grant:
  - Starts at arbiter. Each successive device on the “Daisy Chain” can “block” the signal from passing to the next device or pass the signal to the next device.
  - When “blocking”, that I/O device has requested the bus and will use the “Bus Grant” for itself.
  - When “passing”, the I/O device doesn’t care which device controls the bus.
- Arbiters are often “built-in” to the CPU

# Bus Arbitration – decentralized

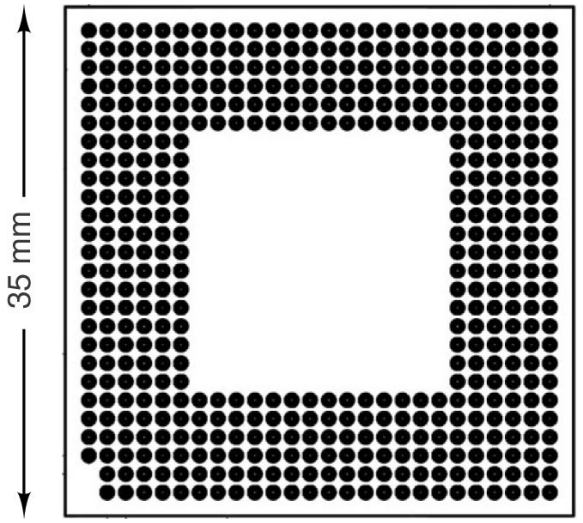


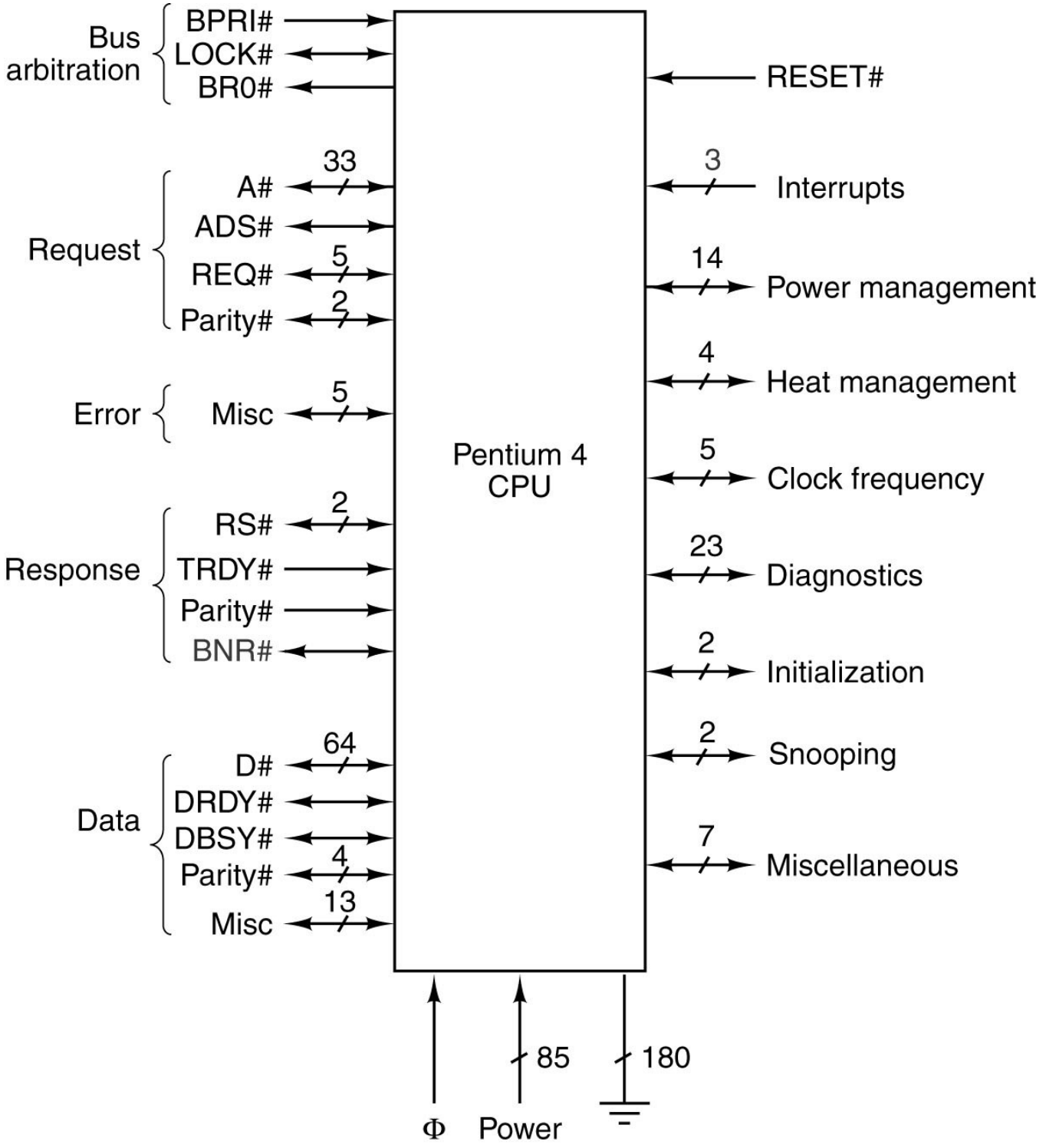

If there is no “arbiter” a “Daisy Chain” can still be used, but an indication that the Bus is Busy is required.

This technique could work for a multiprocessor system where any of the processors can act as the bus master or as a bus slave device.

# EXAMPLE CPU CHIPS

# The Pentium 4



# Pipelining Request on the Pentium 4's memory bus

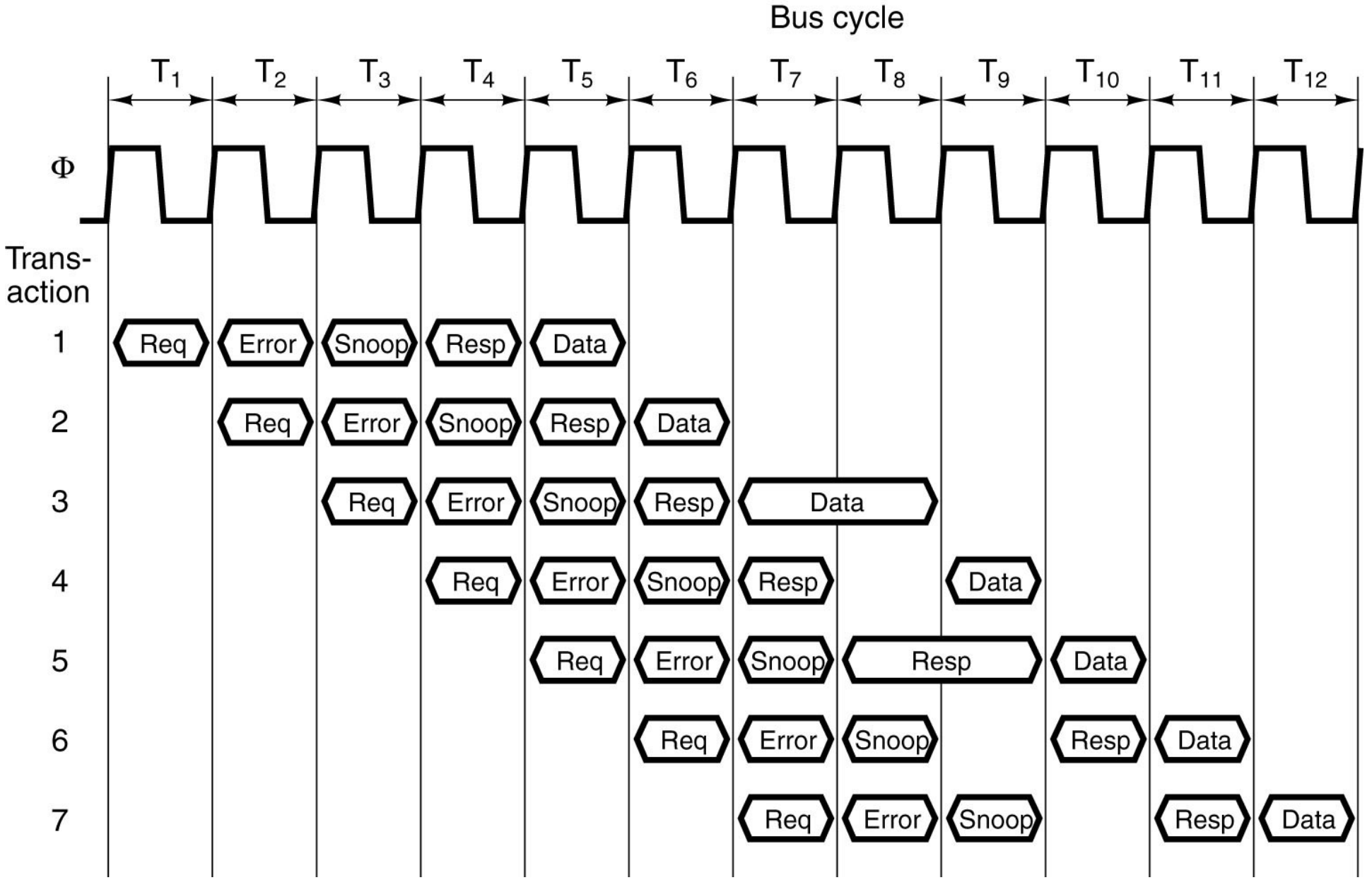
- CPU much faster than DRAM
- It's essential to get the maximum possible throughput from the memory

→ Pentium 4 memory bus is highly pipelined.

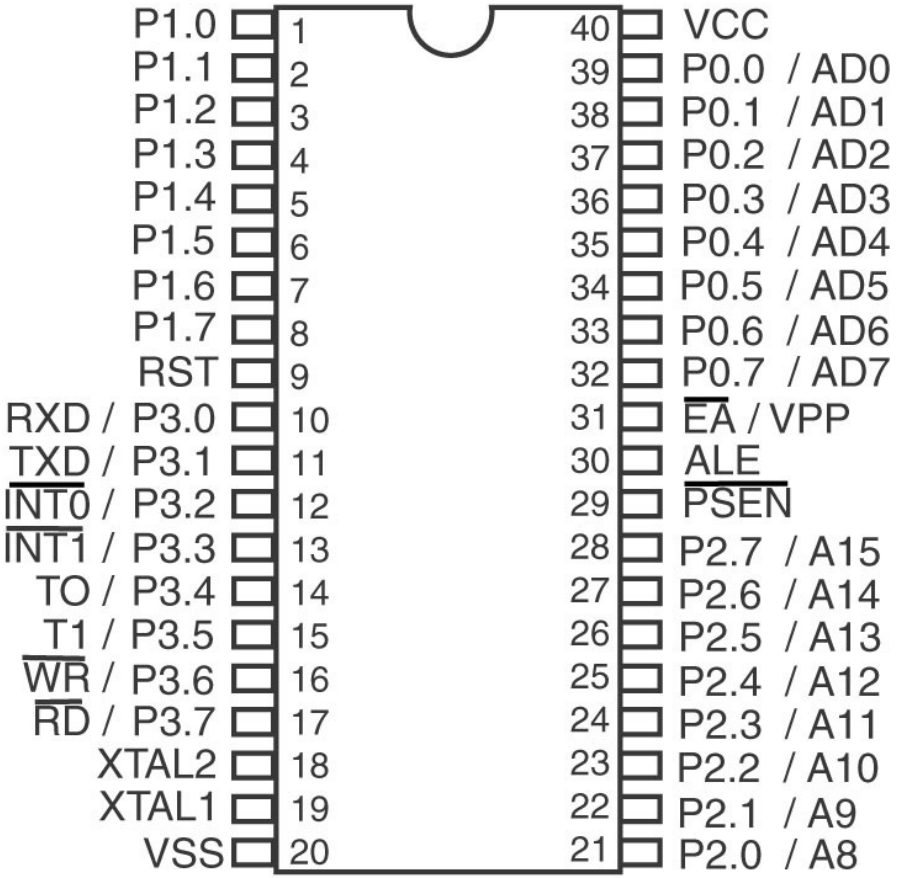
# Pipelining Request on the Pentium 4's memory bus

Bus transaction operations: (similar to PCI coming up)

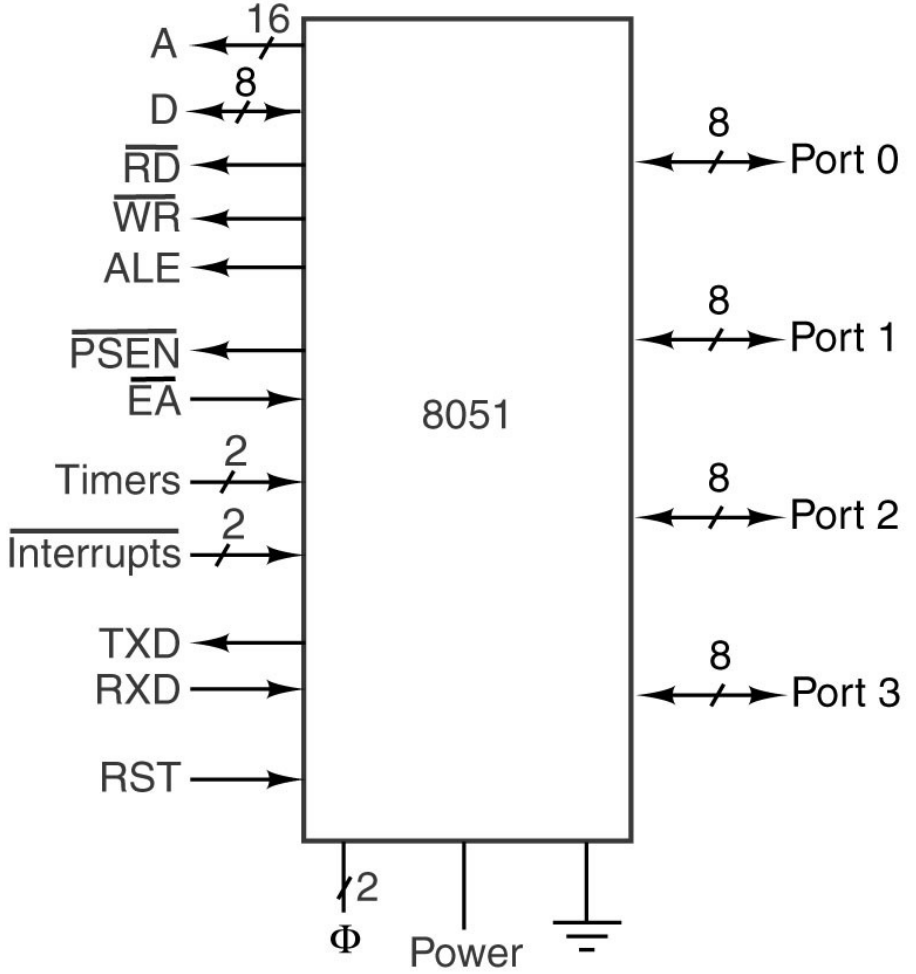
1. The bus arbitration phase (not shown – happens prior to transfer)
2. The request phase – address and command
3. Address/command error reporting phase
4. Snooping phase when multiprocessors are involved
5. The response phase – waiting for the device to respond
6. The data transfer phase


Bus cycle
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
T11
T12
Φ
Trans-
action
1
Req
Error
Snoop
Resp
Data
2
Req
Error
Snoop
Resp
Data
3
Req
Error
Snoop
Resp
Data
4
Req
Error
Snoop
Resp
Data
5
Req
Error
Snoop
Resp
Data
6
Req
Error
Snoop
Resp
Data
7
Req
Error
Snoop
Resp
Data

# The 8051




Physical pinout of the 8051

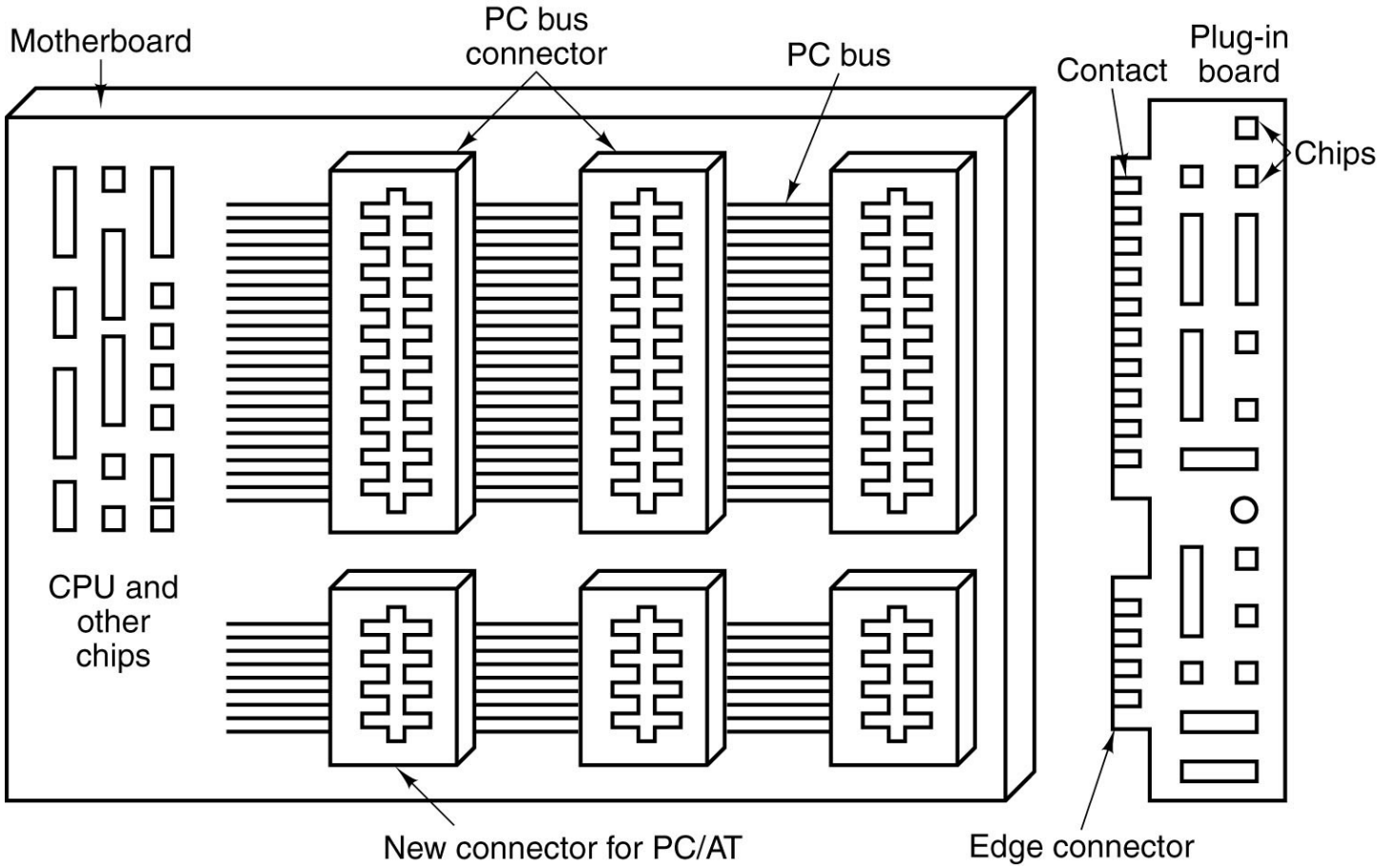



Logical pinout of the 8051.

# EXAMPLE BUSES

ISA, PCI, PCI Express, USB

# The ISA Bus




The PC/AT bus has two components, the original PC part and the new part.
Max rate = 8.33 MB/sec

# Moving away from ISA …

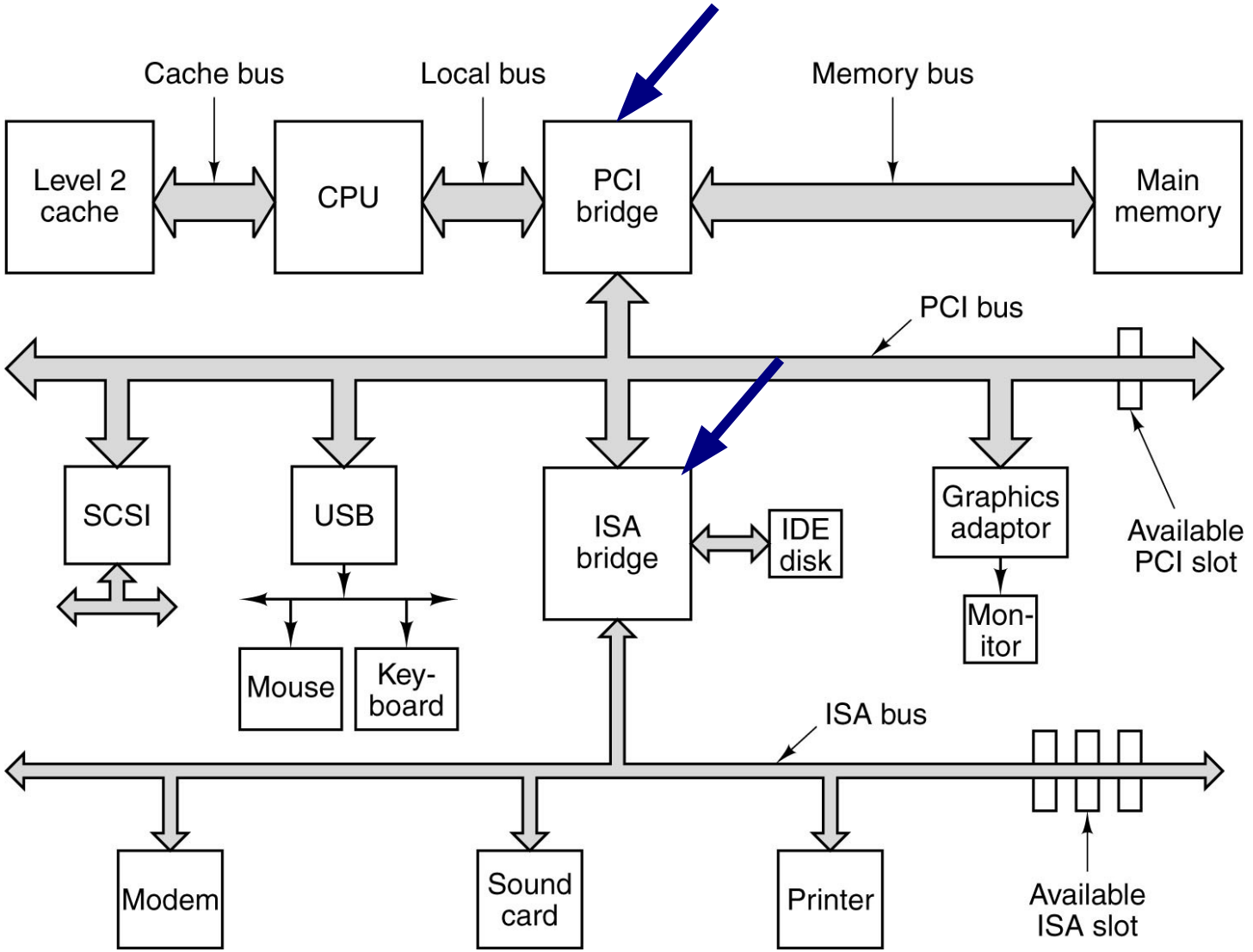
- New application was coming: Windows, GUI
  - 1024x768 screen, true color, 30 screens/sec is needed → data rate of 67.5 MB/sec
  - Need a bus bandwidth of 135 MB/sec or more?

    → ISA bus is out of hand

→ PCI (peripheral component interconnect bus) coming

  - Original PCI bus transferred 32 bits per cycle, ran @33MHz → total bandwidth of 132MB/s
  - PCI 2.1 run @66MHz, handle 64 bit transfer → 528 MB/s

# The PCI Bus (1)



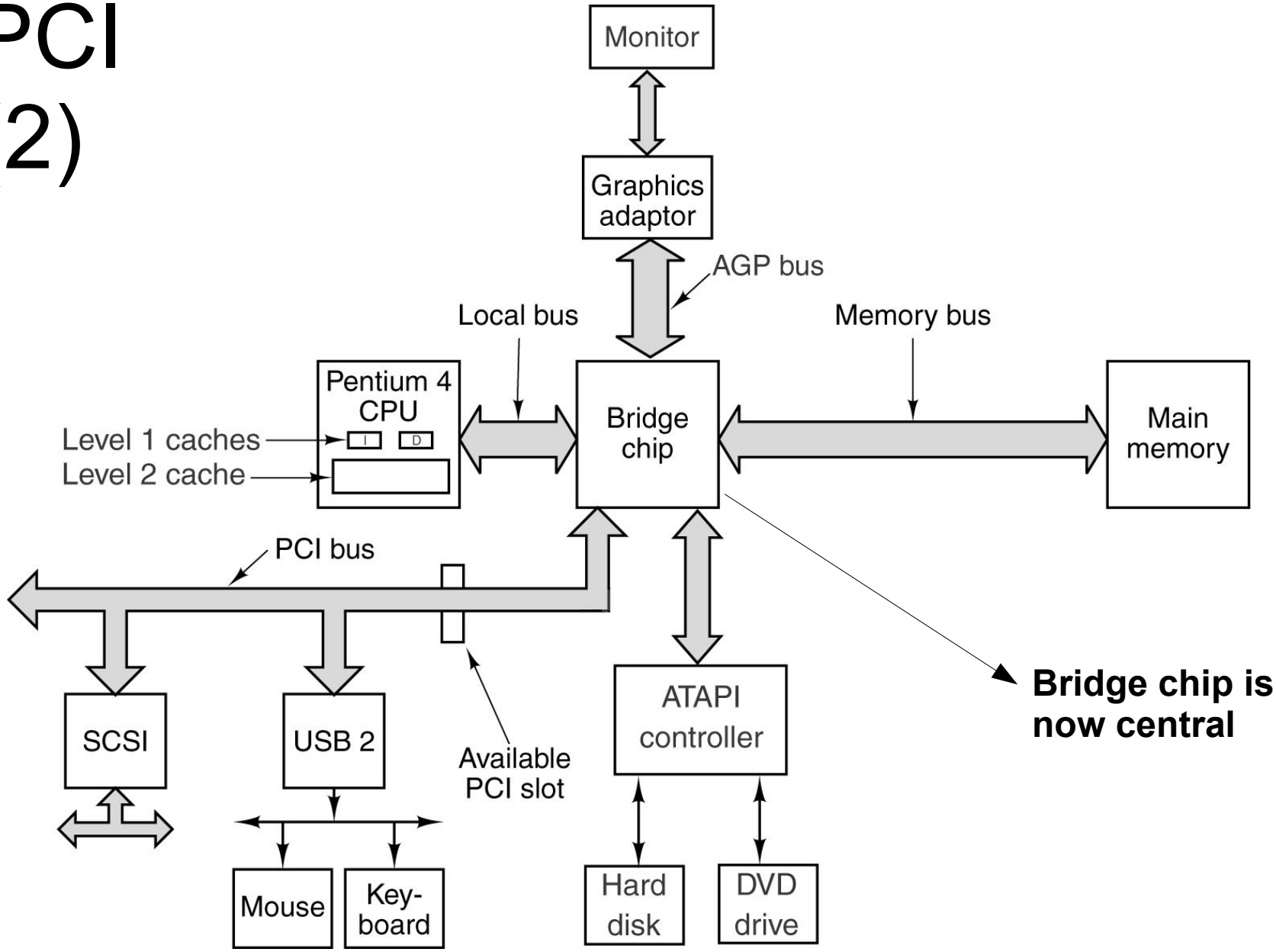

Architecture of an early Pentium system. The thicker buses have more bandwidth than the thinner ones but the figure is not to scale.

# The PCI Bus (2)

- By the late 1990s, ISA bus was dead
- More graphic demand
  - Monitor resolution had increase in cases to 1600x1200
  - Highly interactive game with full-screen full motion video

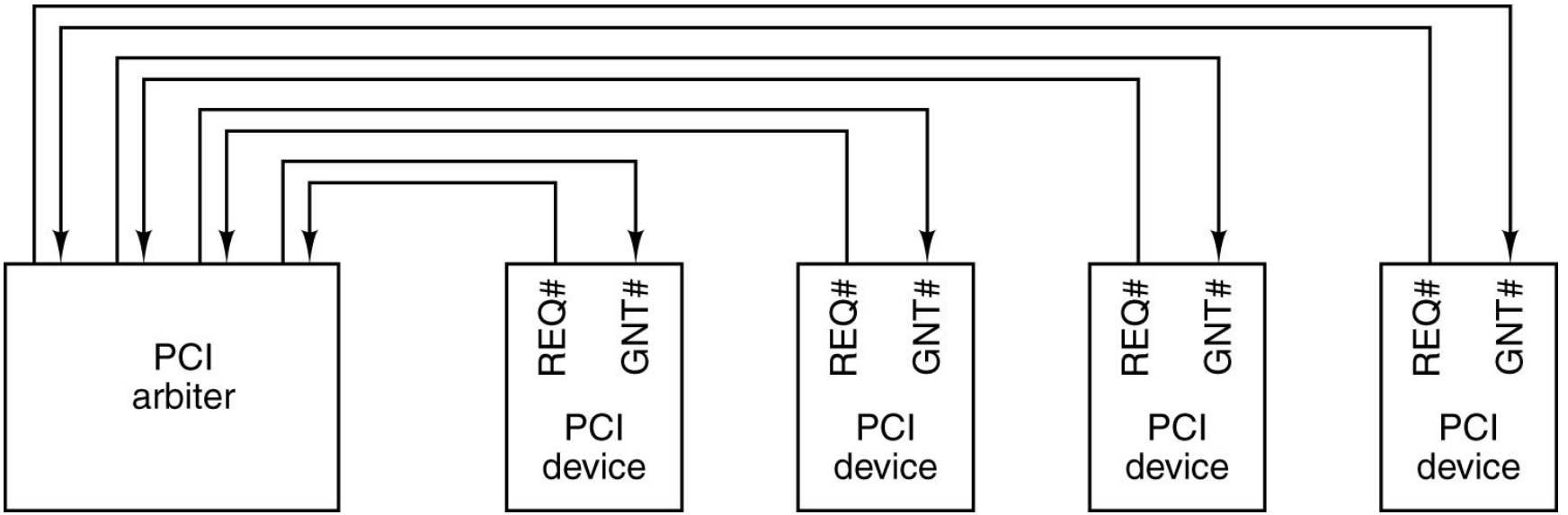
    → Intel released AGP bus (Accelerated Graphic Port bus). AGP 3.0 running at 2.1GB/s

# The PCI Bus (2)




The bus structure of a modern Pentium 4.

# PCI Bus Arbitration




The PCI bus uses a centralized bus arbiter.
Algorithm used by the arbiter is not defined by the PCI specification; in fact round-robin, priority and other scheme are allowed

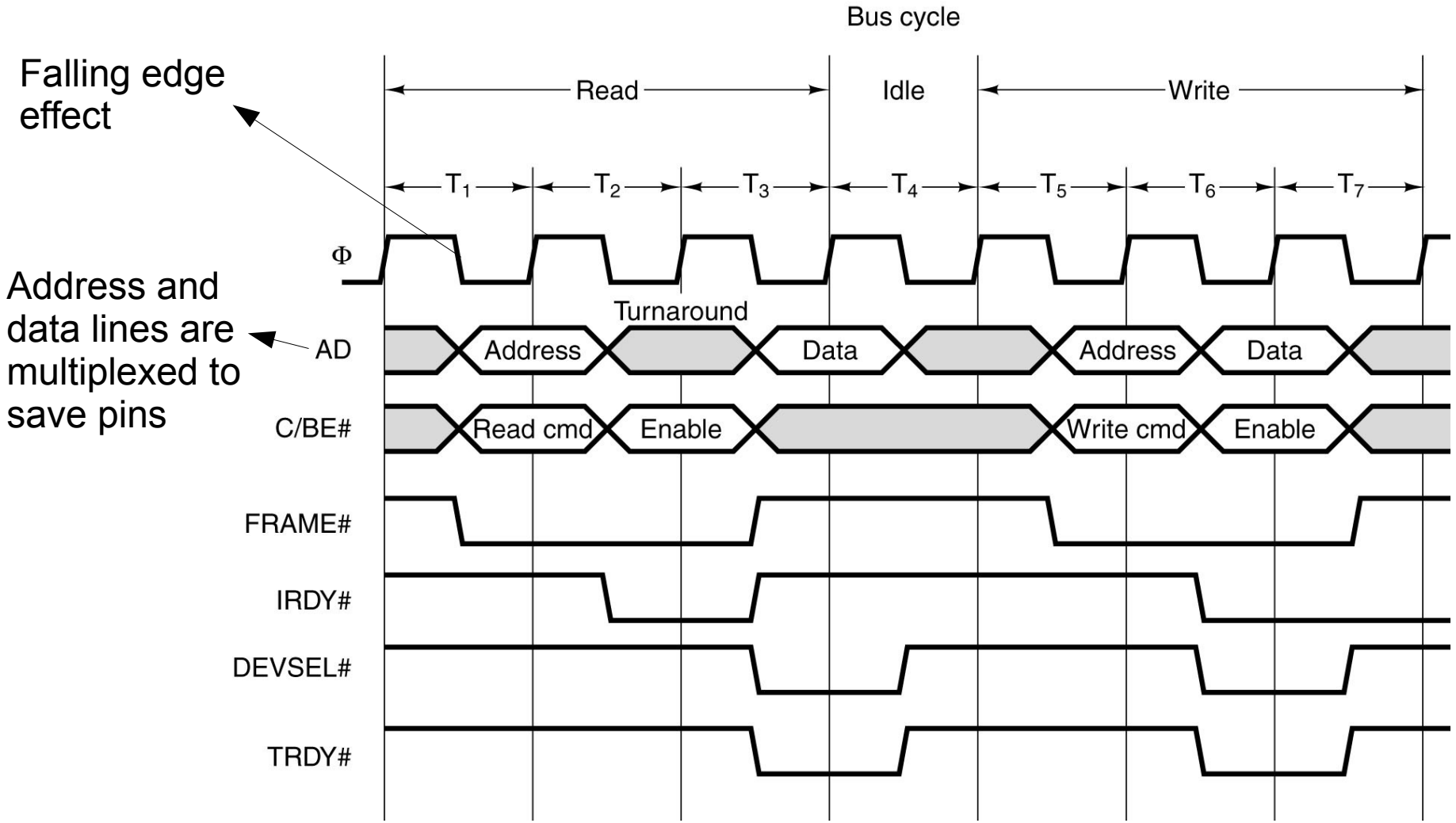
# PCI Bus Signals

| Signal | Lines | Master | Slave | Description |
|---|---|---|---|---|
| CLK | 1 | | | Clock (33 MHz or 66 MHz) |
| AD | 32 | × | × | Multiplexed address and data lines |
| PAR | 1 | × | | Address or data parity bit |
| C/BE | 4 | × | | Bus command/bit map for bytes enabled |
| FRAME# | 1 | × | | Indicates that AD and C/BE are asserted |
| IRDY# | 1 | × | | Read: master will accept; write: data present |
| IDSEL | 1 | × | | Select configuration space instead of memory |
| DEVSEL# | 1 | | × | Slave has decoded its address and is listening |
| TRDY# | 1 | | × | Read: data present; write: slave will accept |
| STOP# | 1 | | × | Slave wants to stop transaction immediately |
| PERR# | 1 | | | Data parity error detected by receiver |
| SERR# | 1 | | | Address parity error or system error detected |
| REQ# | 1 | | | Bus arbitration: request for bus ownership |
| GNT# | 1 | | | Bus arbitration: grant of bus ownership |
| RST# | 1 | | | Reset the system and all devices |



Mandatory PCI bus signals.

# PCI Bus Transactions

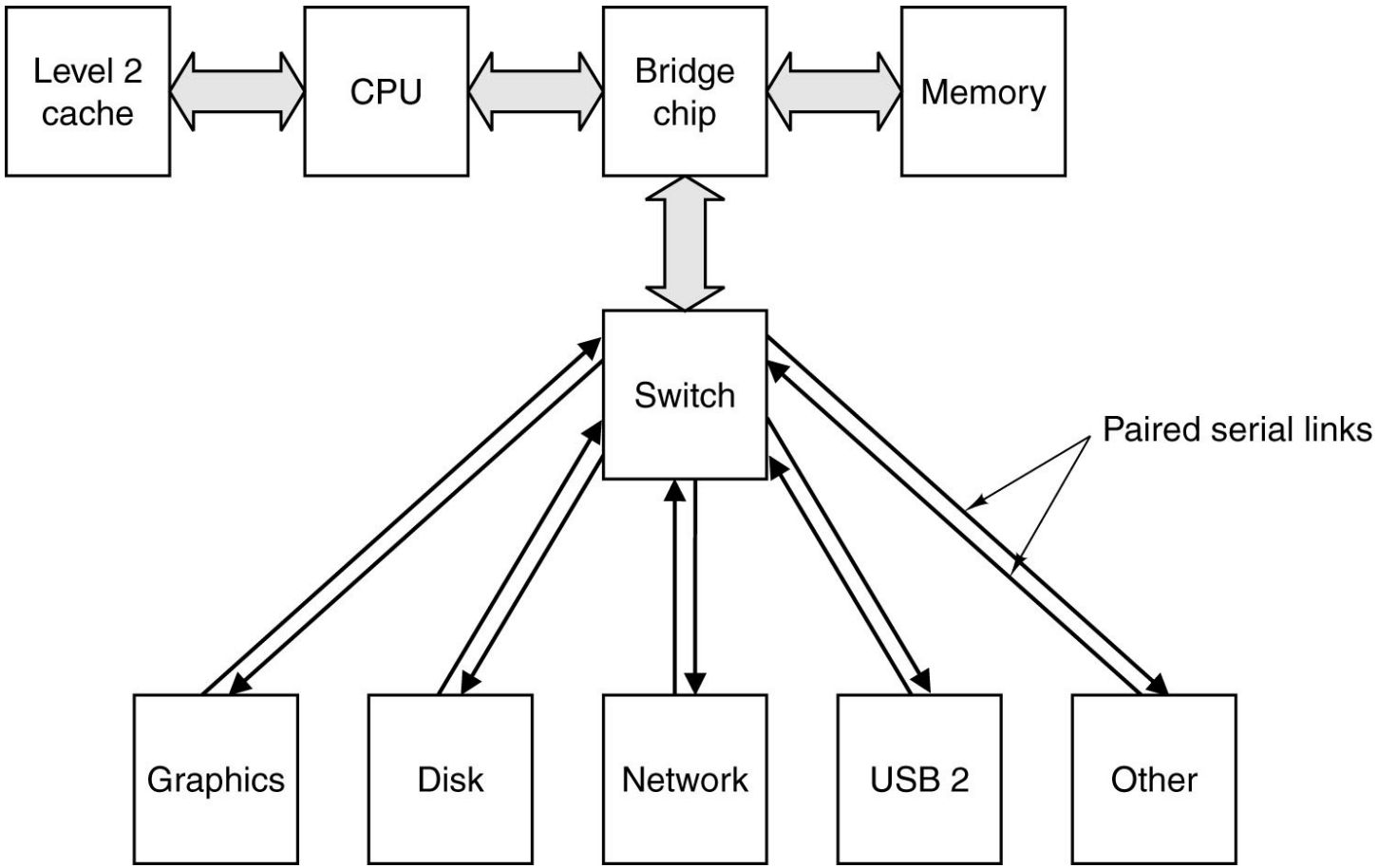



Examples of 32-bit PCI bus transactions. The first three cycles are used for a read operation, then an idle cycle, and then three cycles for a write operation.

# PCI Express

- Parallel buses are challenged as the clock rates are pushed higher + big card. Therefore, PCIe interface is based on high-speed *point-to-point serial communications!*
  - PCIe has PCI name fro marketing purpose only
- In PCIe, multiple “lanes” (x1, x2, x4, x8, x16 or x32) can be connected to the same board.
- Point-to-point connections using a *switch*. (Note: Ethernet switches)

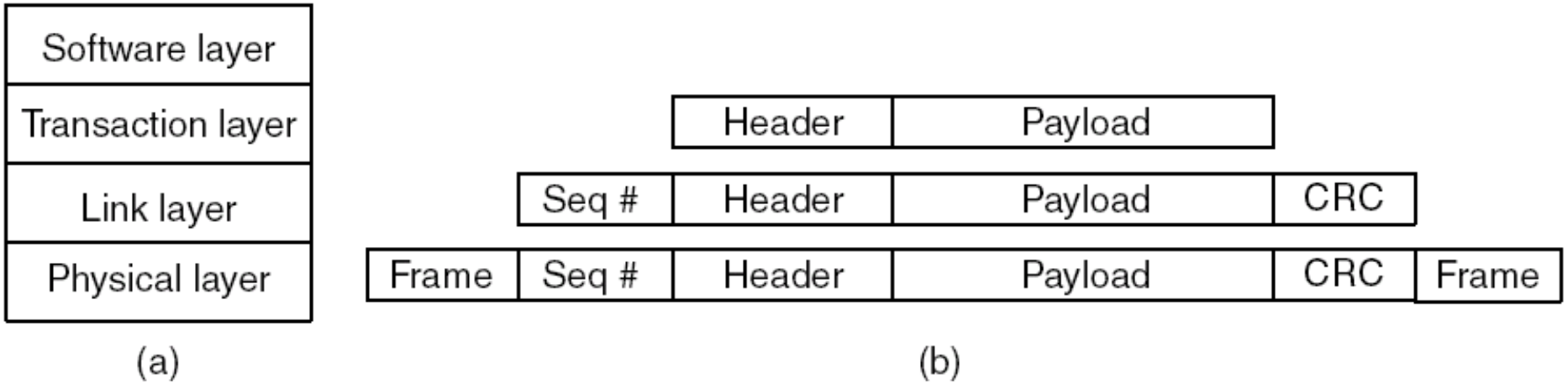
# PCI Express




Each lane uses low-voltage differential signaling (LVDS) and 4-wires for bidirectional communications (one pair TX, one pair RX). A reference clock is provided to synchronize bit transfers.

# PCI Express Protocol Stack

- The packet structure from an OSI software layered protocol consideration is shown below.




(a) The PCI Express protocol stack.
(b) The format of a packet (implement in hardware, compare to software of protocol stack)

# PCI Express

- Key features in PCIe as compared to PCI
  - Centralized switch for routing ...
  - Serial point-to-point connections
  - Packet based communications
  - Error-detection for increased reliability
  - Expandable to multiple level of switching.
  - Hot pluggable: can be inserted or removed when the power is on.

# The Universal Serial Bus (USB)

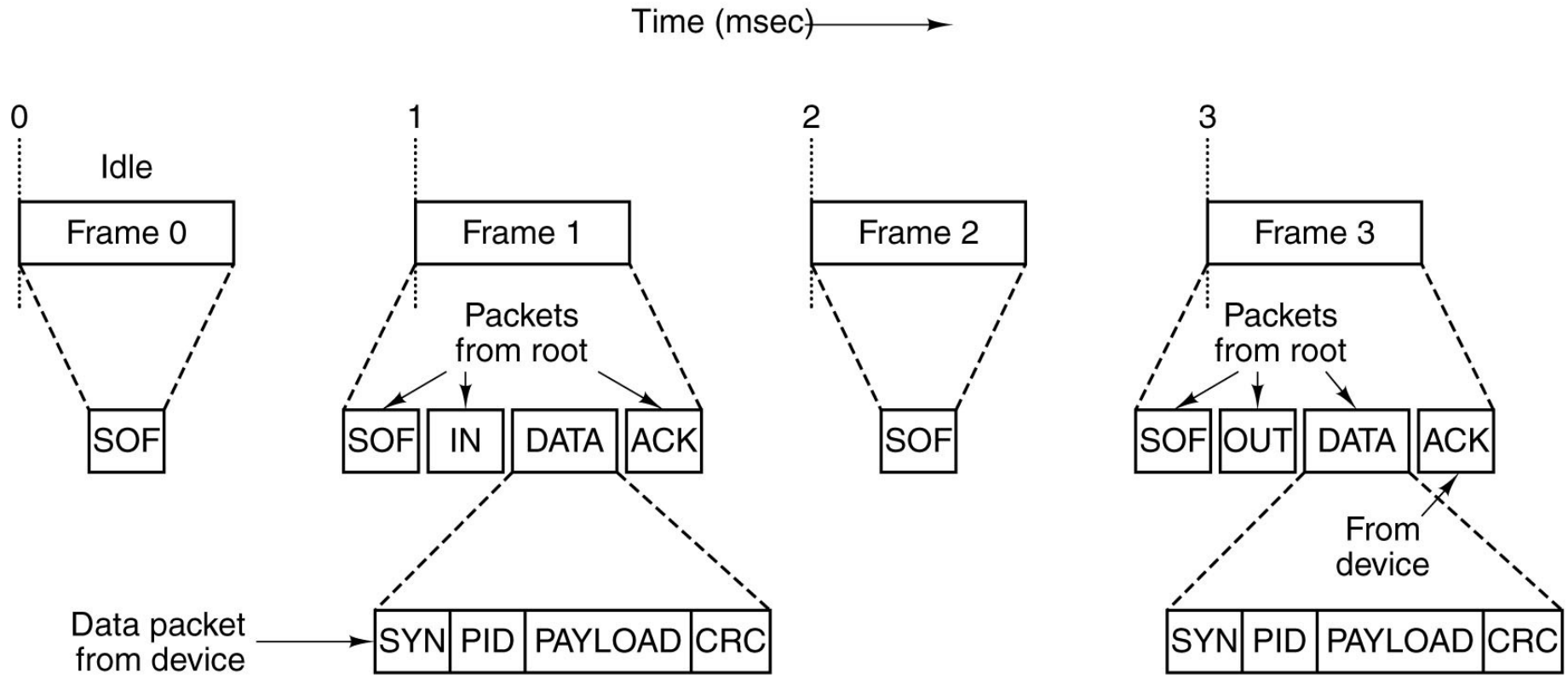
- In 1993. seven companies got together to design a better way to connect low-speed I/O devices (mouse, keyboard, etc.) The companies were Compaq, DEC, IBM, Intel, Microsoft, NEC and Nortel.
- The goals:
  - Users must not have to open cases to install new I/O devices.
  - There should be only one kind of cable, good for interconnecting all devices.
  - I/O devices should get their power from the cable

See next slide →

# The Universal Serial Bus (USB)

- The goals (cont'd)
  - Up to 127 devices should be attachable to a single computer.
  - The system should support real-time devices (e.g. sound, telephone, game inputs)
  - The system should be installable while the computer is running.
  - No reboot should be needed after installing new devices
  - The new bus and its I/O should be inexpensive to manufacture.

# The Universal Serial Bus (USB)




The USB root hub sends out frames every 1.00 ms

# Copyright note

- Slides are adopted form lecture notes of Tanenbaum, Structured Computer Organization, Fifth Edition, (c) 2006 Pearson Education, Inc.