Hardware fundamentals for the software engineer

Reference: Simon chapter 1 and 2

Nuno Alves (nalves@wne.edu), College of Engineering

Embedded system

- An embedded system is any computer system inside a product other than a computer
- Embedded systems have a processor and a memory. Some have a serial port or a network connection. The usually not have keyboards, screens or disk drives.

Difficulties when writing embedded software

Some difficulties

- **Throughput**: Your system may need to handle a lot of data in a short period of time.
- **Response**: You system may need to react to events quickly.
- **Testability**: Setting up equipment to test your embedded system may be hard.
- **Debugability**: Without a screen of a keyboard, finding out what is wrong with your application (besides the fact that its not working) can be hard.

More difficulties

- **Reliability**: Embedded systems must be able to handle any situation without human intervention.
- **Memory space**: Memory is limited on embedded systems, and you must make the software and the data fit in whatever. memory exists
- **Program installation**: You need special tools to get your software in embedded systems.
- **Power consumption**: Portable systems must run on battery power, and the software in these systems must conserve power.

Even more difficulties

- **Processor hogs**: Computing that requires large amounts of CPU time may complicate the response problem.

- **Cost**: Reducing the cost of the hardware is a concern for many embedded systems; often time software functions on a barely adequate system for the job.

Hardware fundamentals: terminology

C/N	REV	DESCRIPTION	DATE	APPROVED
D-2520	A	ORIGINAL ISSUE	2-9-93	

U1 Z180 MPU

U2 27512

U3 TC55257

U4 Z85230

U5 ZILOG 12/3/88 EPLD: EP610 Z180-SCC BUSLOGIC OFF

U6 HC14

U7 HC10

U8 MAX232

Z8S180

J1 J2 J3 J4

P1 P2 P3 P4 P5 P6

X1 18.432

X2

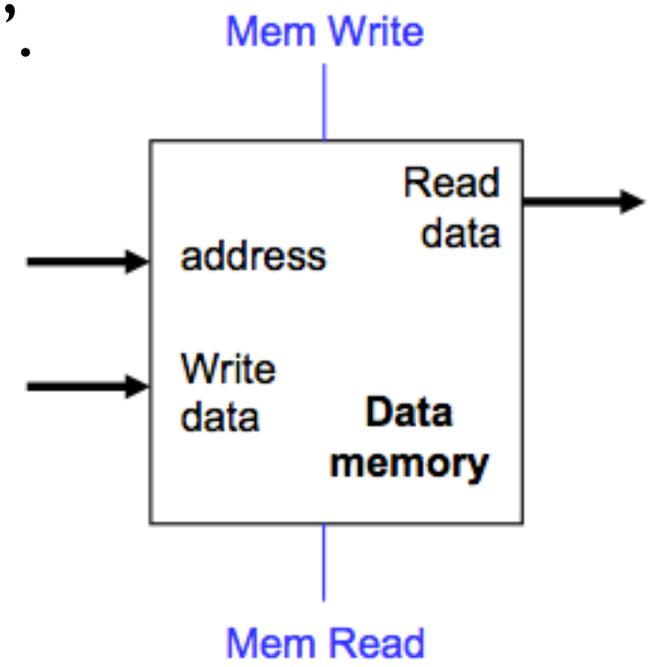
CONFIDENTIAL COMPANY PROPERTY - ZILOG INC.

TITLE: SCHEMATIC DIAGRAM Z8S180 ESCC EVALUATION	ENGINEER: TUCHOLSKI
FILE: Z8S180AP.SCH	APPROVED: D PEREIRA
	DATE: 5-26-1992
DWG NUMBER: 96C0288-001	REV: A
	PAGE: 1 OF 1

Figure 3.20 A Sample Schematic

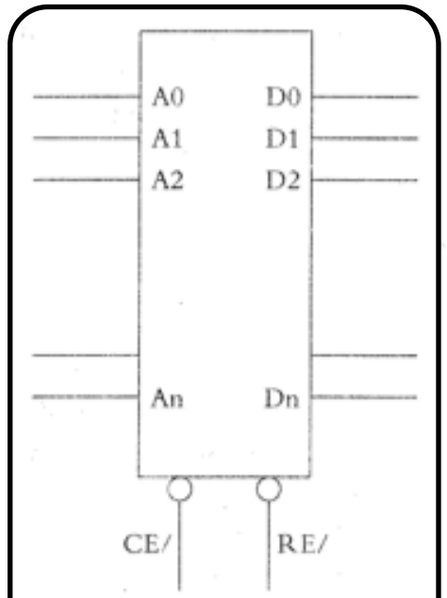
Signal assertion

- Every circuit contains many signals whose purpose is to indicate various conditions. For example "reset the microprocessor", or "get this data from this memory chip".
- These signals are said to be **asserted**, when they are signaling whatever they want to signal.
- For example when the microprocessor needs data from memory, the engineer must assert the "get this data from this memory chip" signal.
- Some signals may be asserted when HIGH others when LOW.

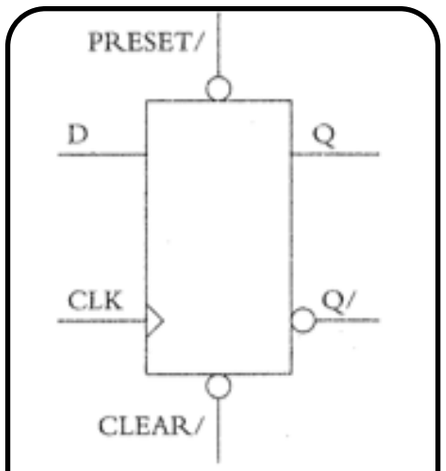


Naming conventions

- Most of the signals in the design have different labels.
- Data signals are usually named D0, D1, D2,...
- Address signals are usually named A0, A1, A2,...
- The signal that indicates "read memory now" is usually MEMREAD.
- If there is a slash (/) or an asterisk (*), usually means that a LOW signal must be asserted for the desired outcome to occur.



ROM



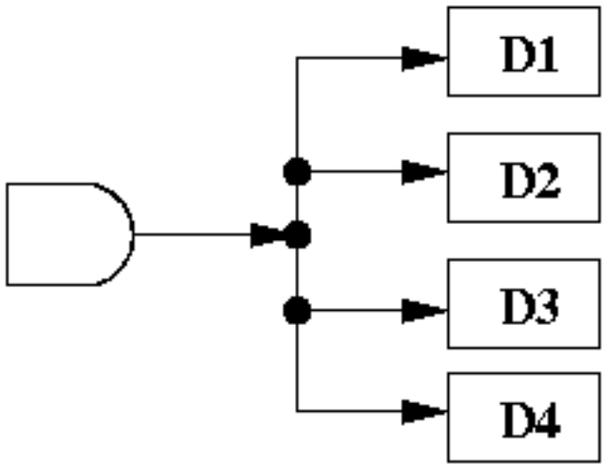
DFlip Flop

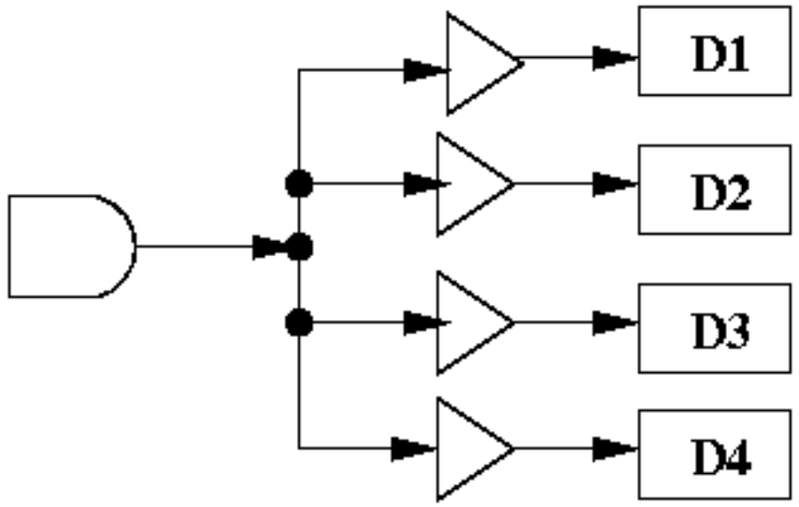
Floating outputs

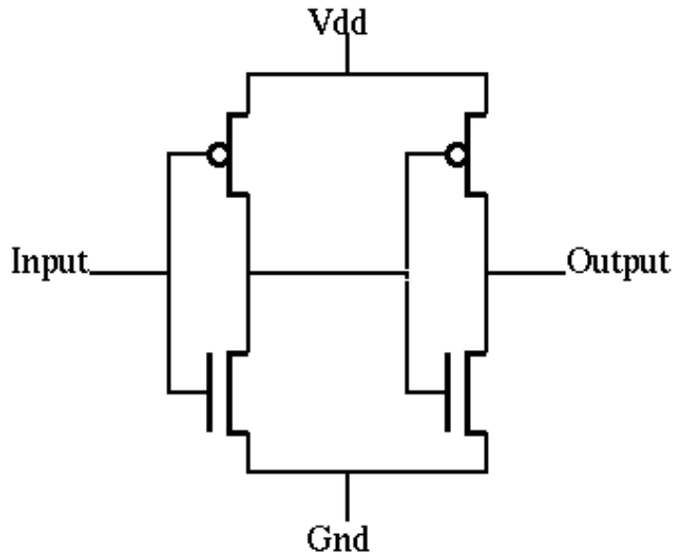
Floating output is has an high impedance state

- If no part of the circuit is driving a signal, then the signal is said to be floating.
- Floating output = high impedance = Z
- Its voltage is indeterminate and may change as time passes.

What is the purpose of adding buffers





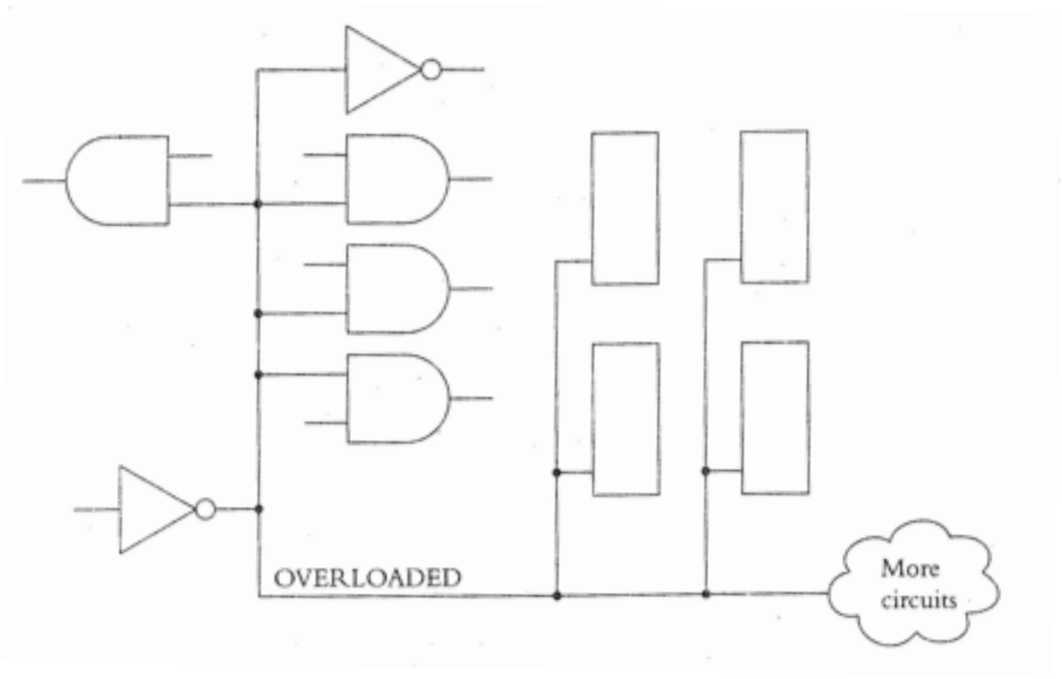


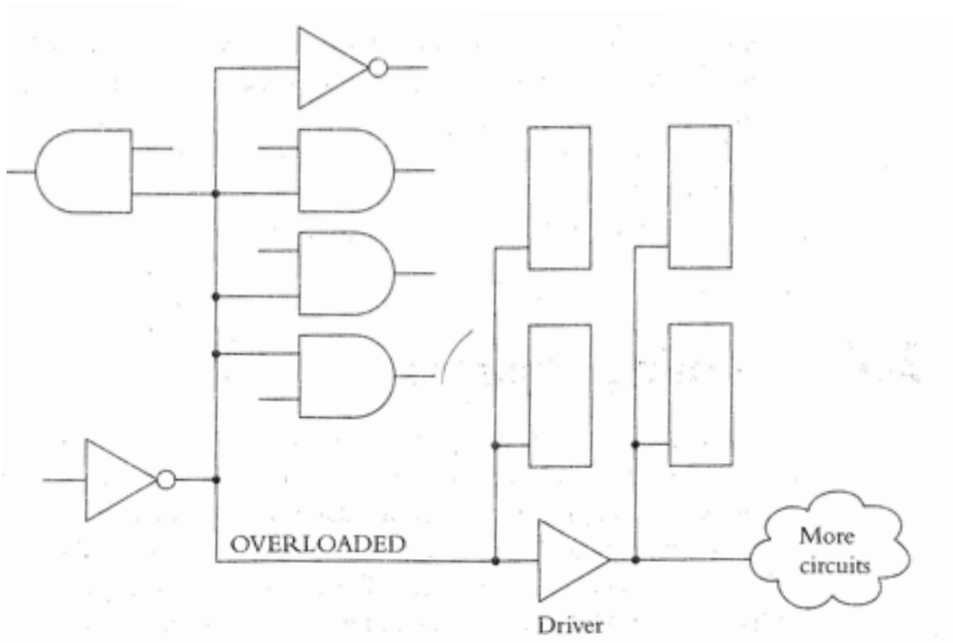
- This AND gate has a fanout of 4
- If each of the four devices gets equal current, then each device gets 1/4 of the initial current
- By adding a buffer, I am boosting current to each device
- This can be done because a buffer has a power connection

Important: Why should I care about ensuring each gate gets a decent amount of current? You can't really accurately measure voltage levels if there is no current!

Example of a loading problem

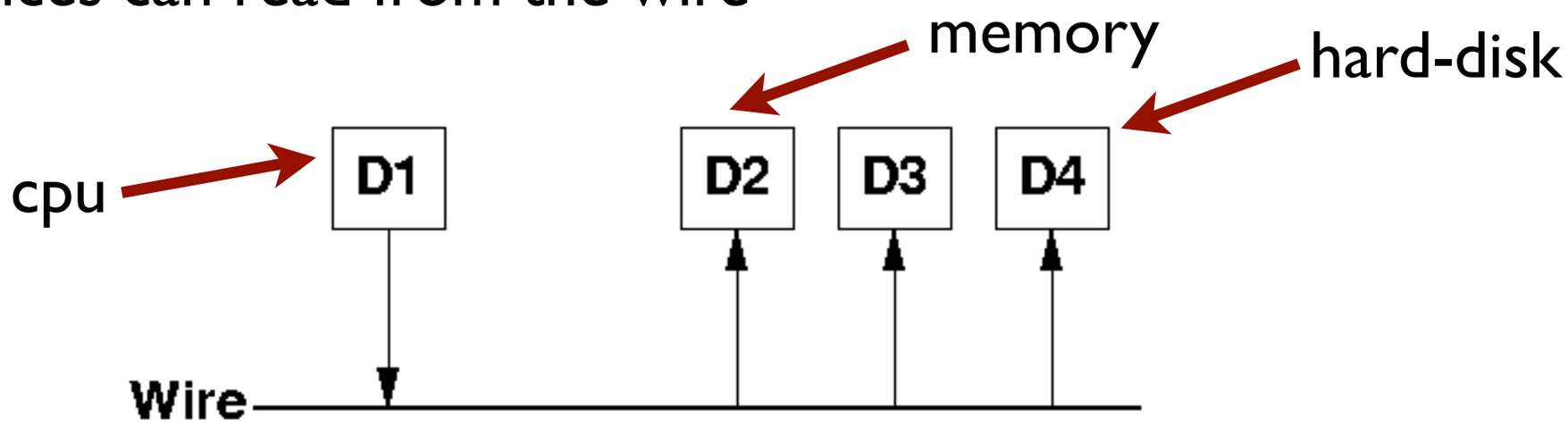
- The OVERLOADED line does not have enough current to load all the chips.
- With a driver, we should have enough current to load all the elements.
- As software engineers we don't have to deal with these things, but they normally show up in designs and we need to know what they are doing.





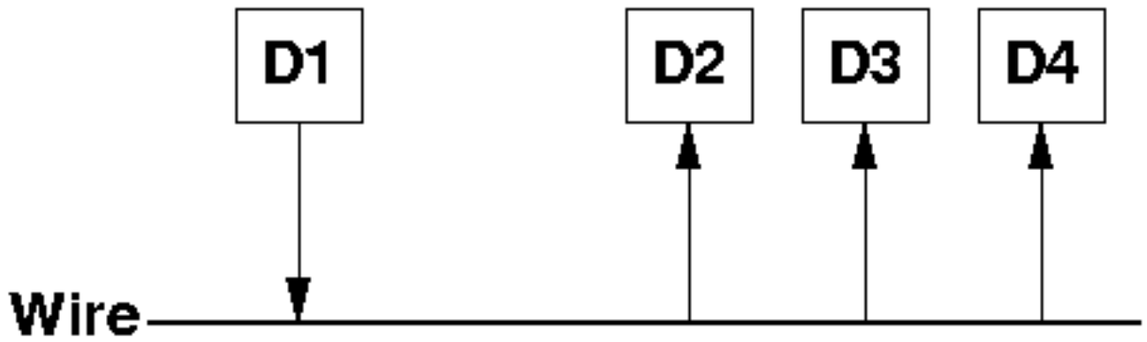
What is High Impedance (Z)?

- We can define a wire as a piece of conductive material that allows electron flow
- A wire allows a 1-bit signal to be sent on it
- At most one device can write to a wire
- A device can write either a '0' or '1' on the wire
- Devices can read from the wire



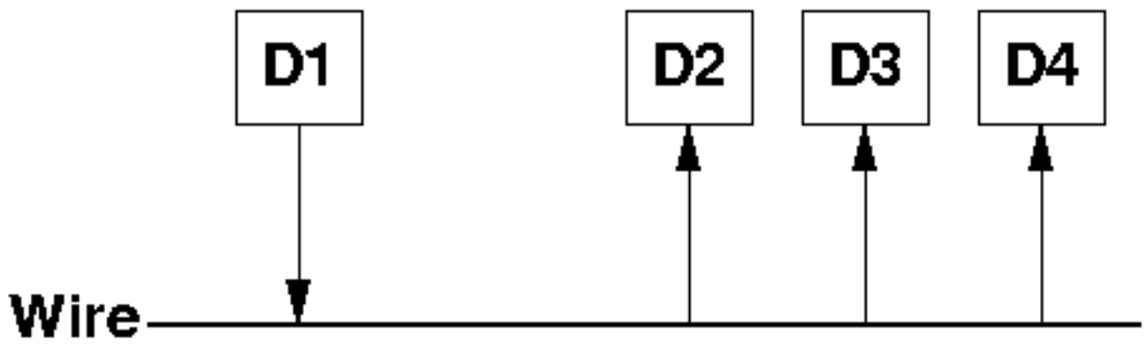
Wire may connect multiple devices

- When a device writes a '1' or '0', in reality, it is asserting a voltage, such as 0 volts for '0' and 5 volts for '1'
- If two devices attempt to write a '0' and '1', then the wire is assumed to have a garbage value
- A device attempting to read from the wire, in such a situation, may read '0's' sometimes and read '1's' at other times
- We want to avoid two devices writing at the same time
- More than one device can read a value from a wire

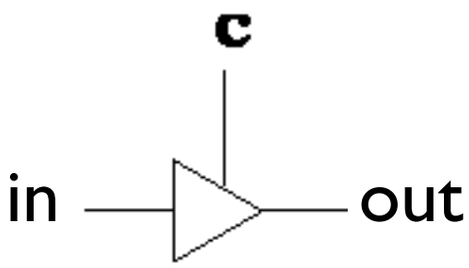


High impedance state

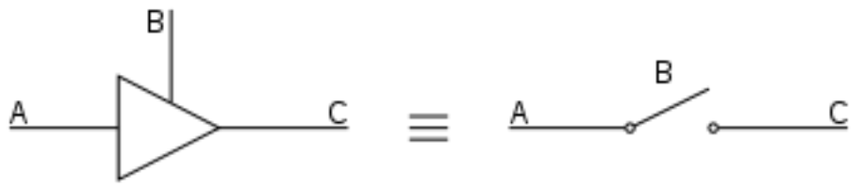
- If no devices write to the wire, then the wire has value Z, which stands for high impedance
- High impedance means that it is neither 0 nor 1
- If no device is writing to a wire, then reading from a wire gets an unknown value (either 0 or 1, but nothing predictable)
- A wire has no memory. That is, if you write a logic-1 to the wire, the wire does not store the value. The device must continuously assert a logic-1



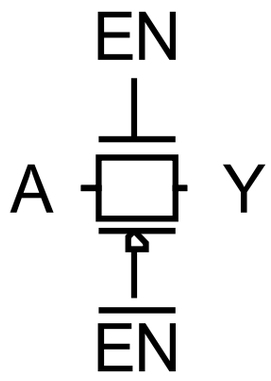
What is a tri-state buffer?



This is how a tri-state buffer looks like....



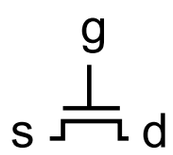
If B is OFF, then output C is floating.

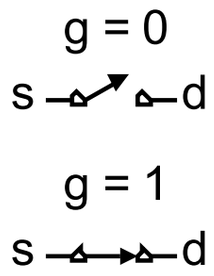


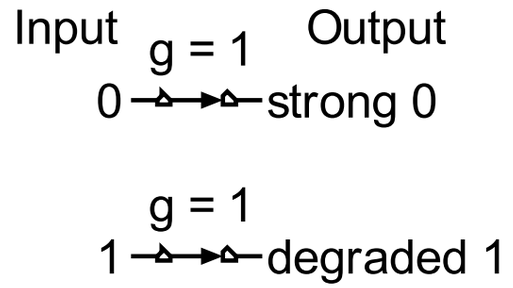
CMOS implementation of a tri-state buffer

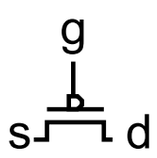
Quick review on transmission gates

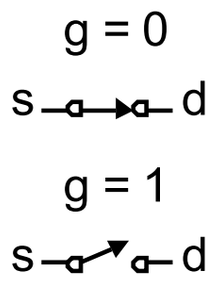
Transistors can be used as switches

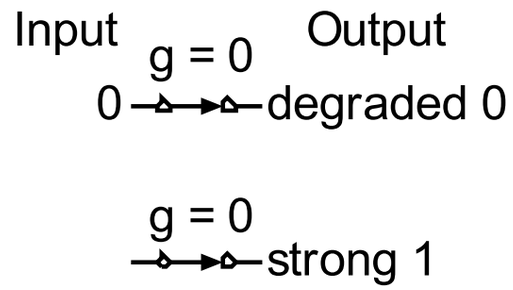




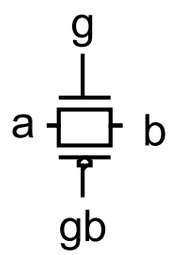


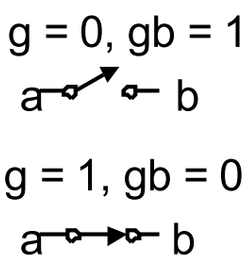


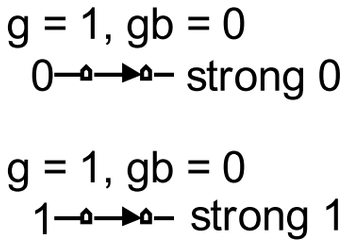




Transmission gates pass strong 1's and 0's

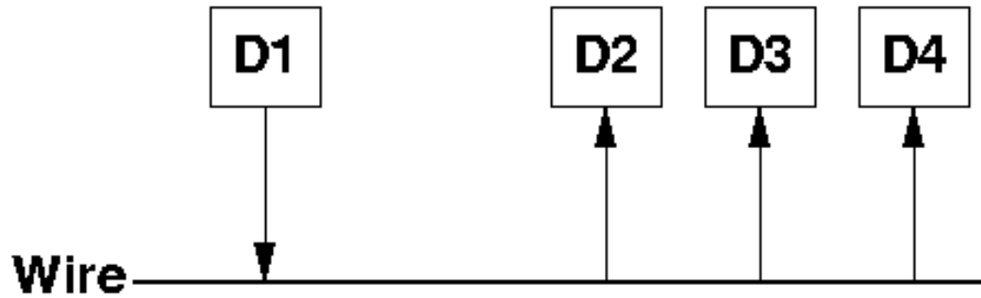






Wire truth-table

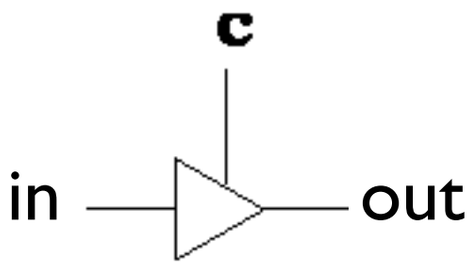
Device #1	Device #2	Output
Write 0	Write 0	0
Write 0	Write 1	Garbage
Write 1	Write 0	Garbage
Write 1	Write 1	1
Write nothing (Z)	Write 0	0
Write nothing (Z)	Write 1	1
Write nothing (Z)	Write nothing (Z)	Nothing (Z)



Garbage as in... sometimes we read a '1' other times we read a '0'.

Tri-state buffer is a control valve

- When the control input is not active, the output is "Z"
- The "valve" is open, and no electrical current flows through
- Thus, even if x is 0 or 1, that value does not flow through

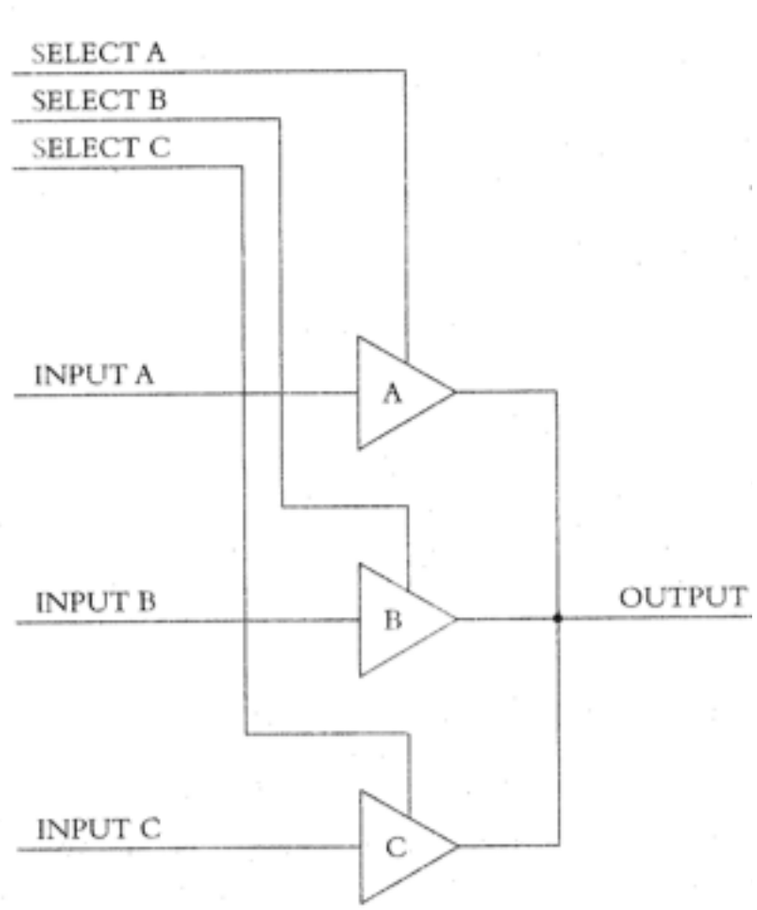


c	in	out
0	0	Z
0	1	Z
1	0	0
1	1	1

Tri-state buffer usage example

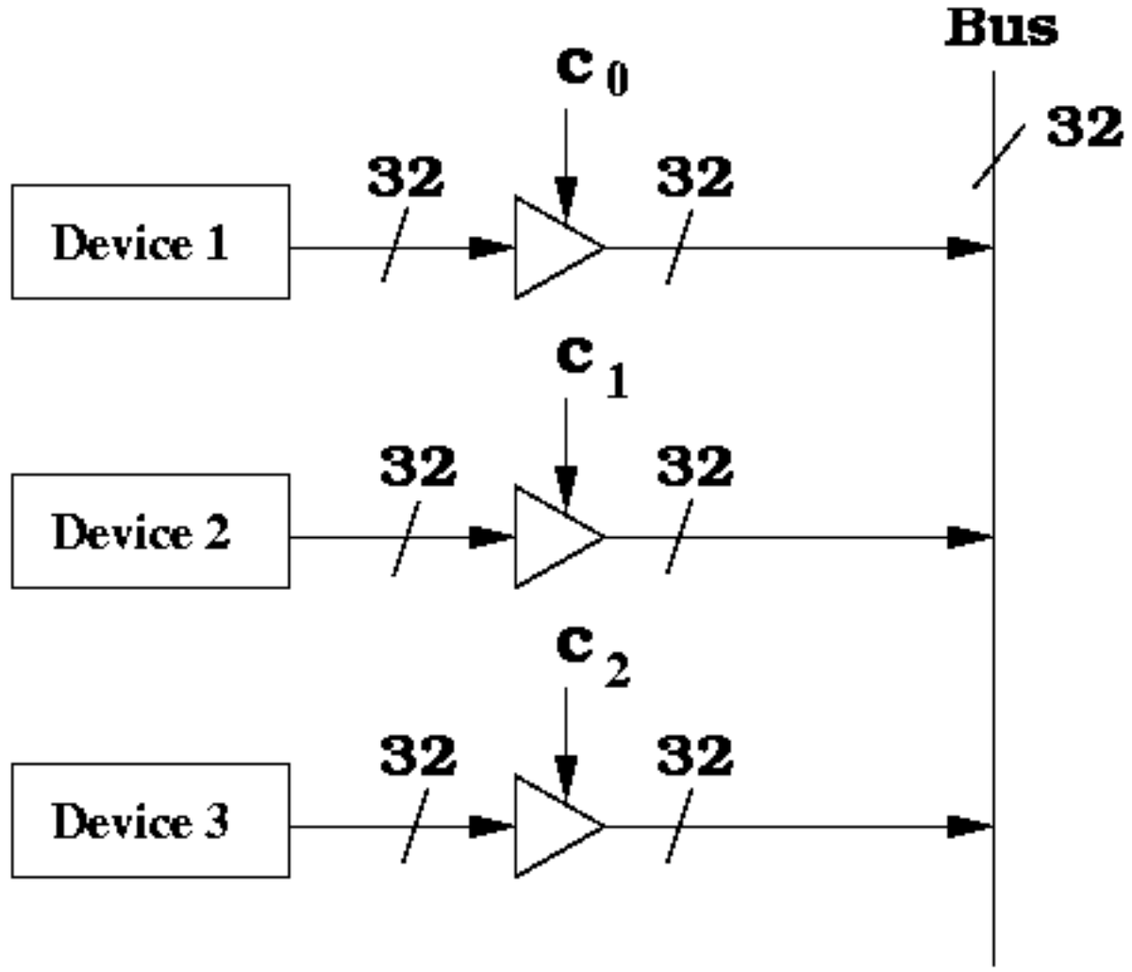
- If SELECT A is asserted, SELECT B and SELECT C not, then INPUT A will drive the OUTPUT signal high or low.

- A microprocessor, a memory chip and some I/O devices must send bytes to one another. Having a tri-state buffer allows us to have a single data bus that just send signals.



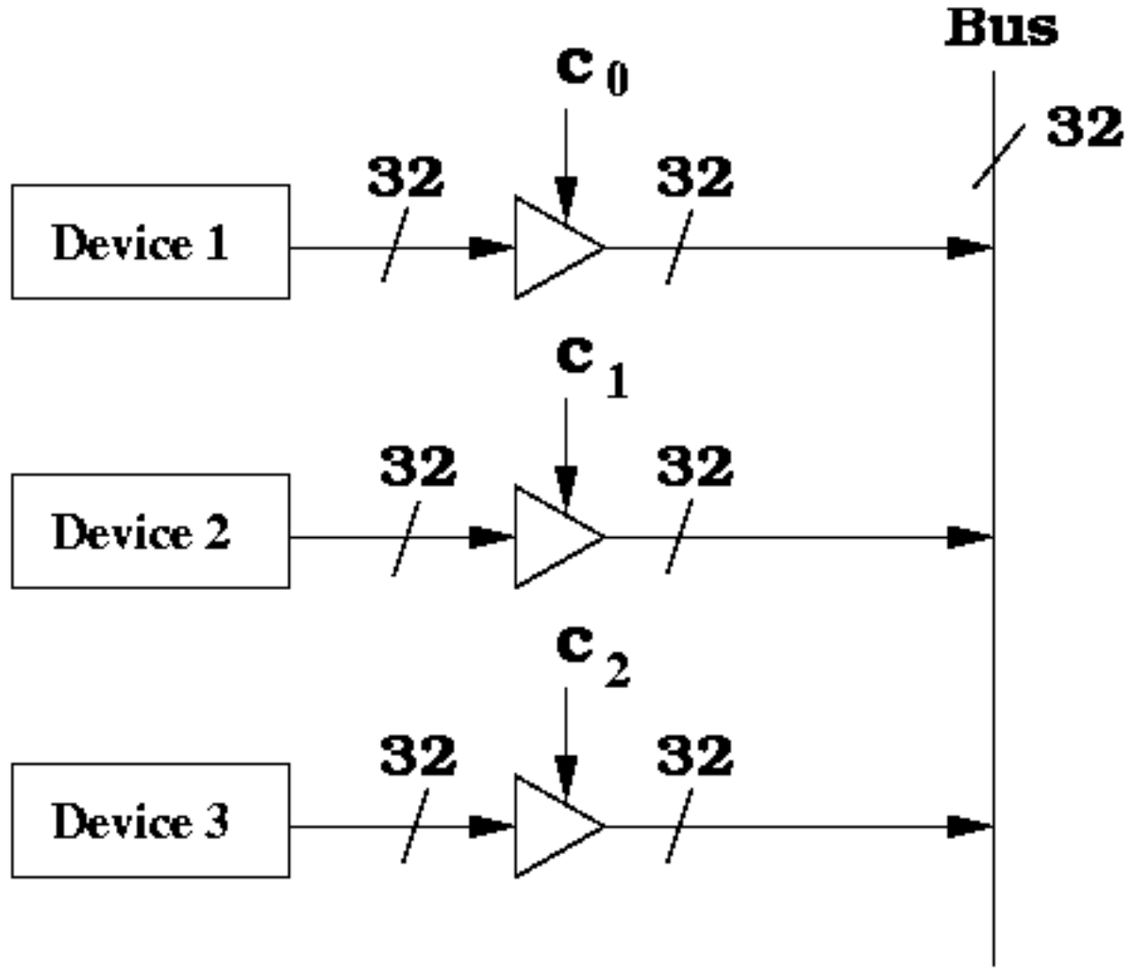
Another example, now with 32 bit tri-state buffers

- A common way for many devices to communicate with one another is on a bus
- That a bus should only have one device writing to it, although it can have many devices reading from it
- Since many devices always produce output (such as registers) and these devices are hooked to a bus, we need a way to control what gets on the bus, and what doesn't.



Tri-state buffers vs MUX

- Who cares? Why don't I replace these three state buffers with a single MUX?
- With a MUX we're guaranteed only one device makes it to the bus.
- However, the MUX ensures that at least one signal must be on the BUS. What if we don't want any devices to make it to the bus?
- Yeah... sure... we can always add an enable input to a MUX.



Bus fight

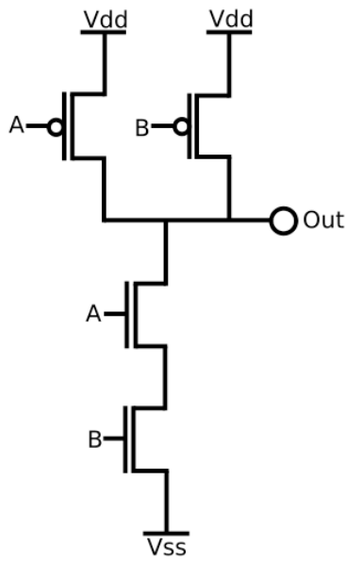
- If two parts drive the same signal with the same value there is no problem.
- If two parts drive a different signal on the same bus, then we have a big problem!
- The parts get very hot and usually they get damaged (this is a called a **bus fight**).
- Bus fights that occur for very short periods of time (nanoseconds) will not be enough to destroy the pieces, but the system will operate unreliably.
- A bus fight is as indication of poor hardware design.

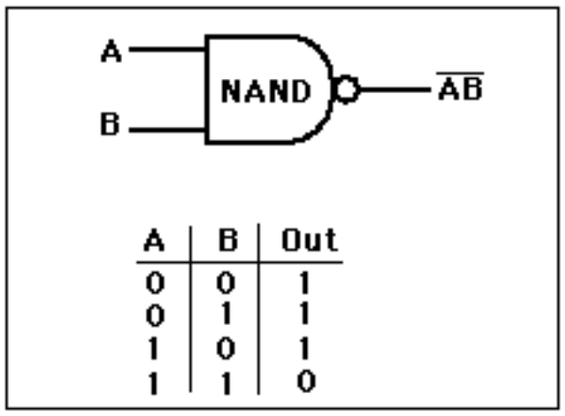
Power and Decoupling

Nuno Alves (nalves@wne.edu), College of Engineering

Signal activity

- Each circuit in a chip has a power pin and also a ground pin.
- However this is not often visually represented on circuit schematics.



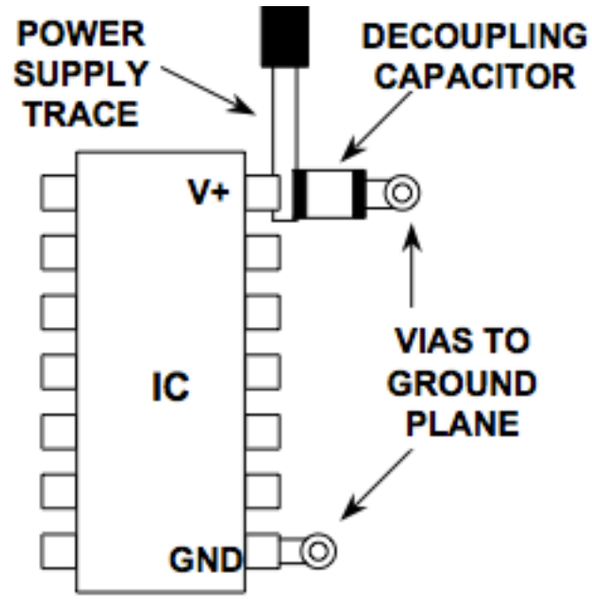


A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

- Problem: Sometimes some of these chips use more power than others. If a chip must change many of its inputs from *LOW* to *HIGH* or *HIGH* to *LOW* at the same time, then this requires a lot of power.

Decoupling capacitors

- To prevent localized power brownout for a few micro-seconds we need to address this issue.
- Most chips stop working temporarily if the voltage drops below 10%.
- To deal with these engineers must add capacitors to the circuit, with one end connected to the signal providing power, and the other end connected to ground.
- A capacitor stores a minuscule amount of energy, like a very small rechargeable battery.



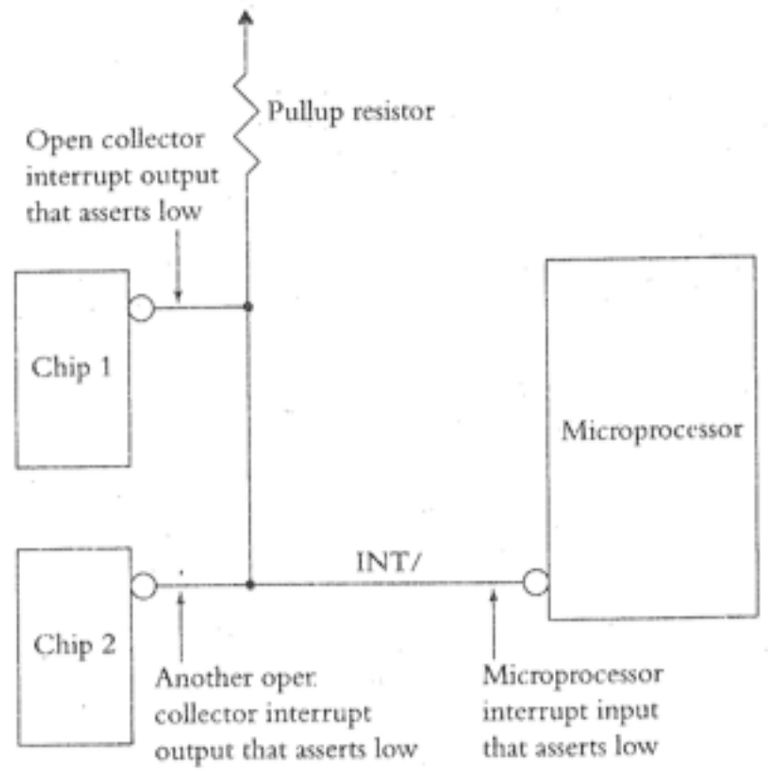
Open collector and tri-state outputs

Open collector

- One special class of outputs allows to attach the outputs of several devices together to drive a single signal.

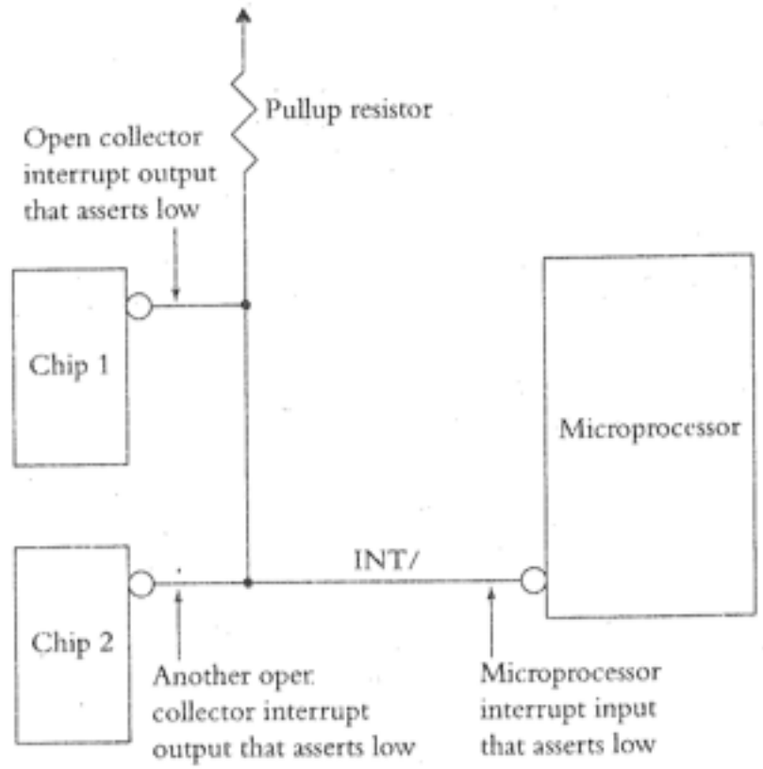
- Unlike the usual signals that drive signals *HIGH* or *LOW*, the open collector can drive the outputs *HIGH*, *LOW* or leave them *floating*.

- With open collectors there is no such thing as a bus fight. If one signal is driven low, all are driven low.



Open collector example

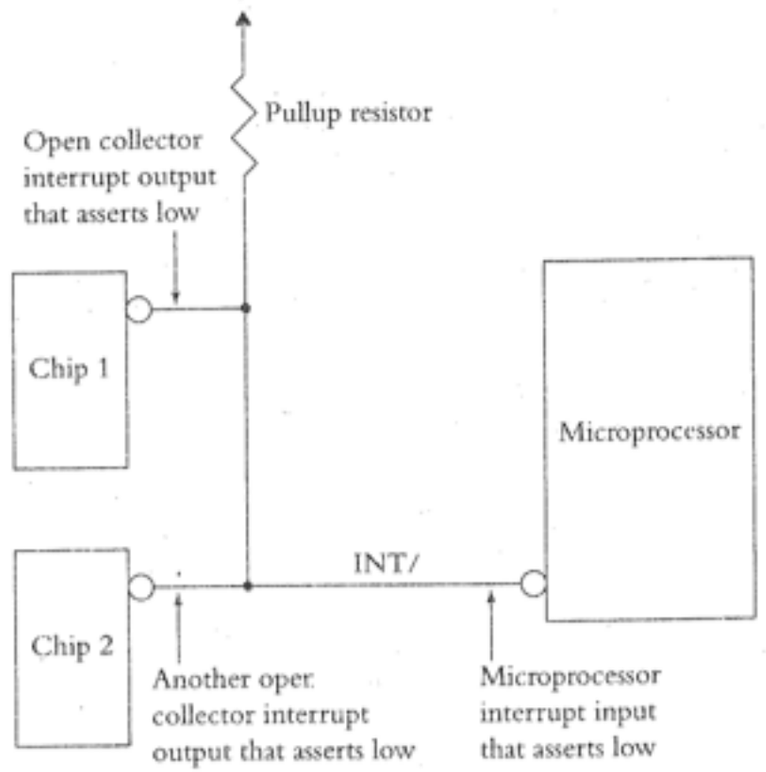
- Example: your microprocessor only has one input for the interrupt, but we have two devices that need to signal interrupts.



- The interrupt (INT/) input is triggered with a LOW signal. Hence the slash (/) after the INT.
- If one of the chips wants to trigger the INT/, then it drives the interrupt line to LOW.
- If no chips want to trigger an interrupt, then the **pullup resistor** will ensure INT/ is HIGH.

Warnings

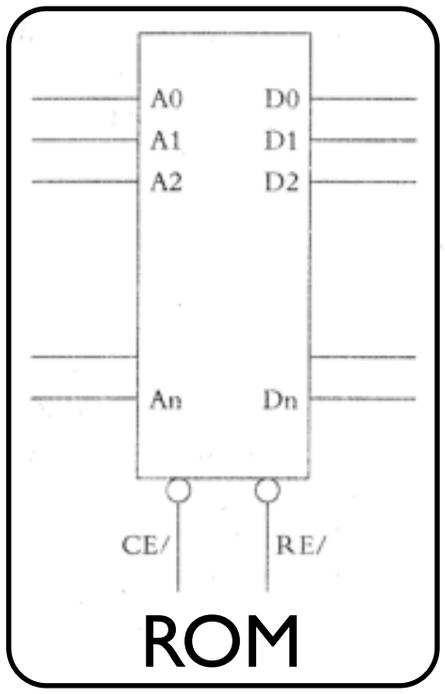
- You cannot omit the pullup resistor and connect INT/ directly to VCC!
- If you did this, you'd have a bus fight, as soon as one of the chips tried to drive INT/ LOW, since VCC would try to keep INT/ HIGH.



Memory

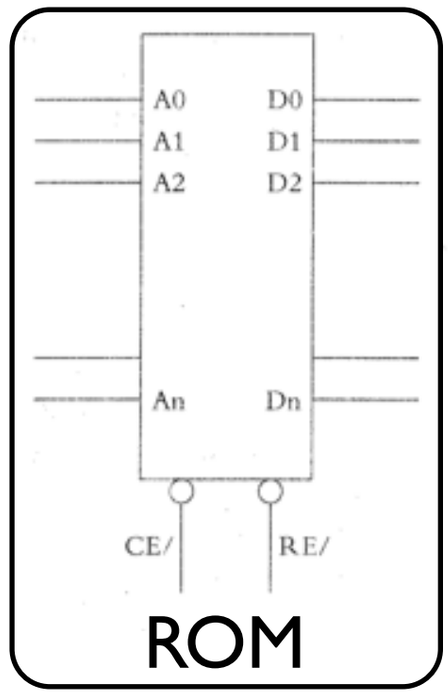
Read only memory

- Non-volatile, means its contents do not disappear once power goes out.
- The microprocessor can read contents from ROM as fast at it can execute them
- The micro processor cannot write new data into the ROM (data is unchangeable)
- When the power is turned ON the micro-processor will start fetching the program from the rom.



Typical ROM signals

- A0, A1, A2,... **address signals**, which indicates the address we want to read from ROM
- D0, D1, ... are the **data signals** driven by the ROM block. They are normally 16.
- CE/ is the **chip select signal**. The ROM will ignore anything, unless CE/ is asserted (a "/" indicates its a 0)
- RE/ is the **read enable signal**, which tells that the ROM should output the data elements when signal is asserted.



Timing example

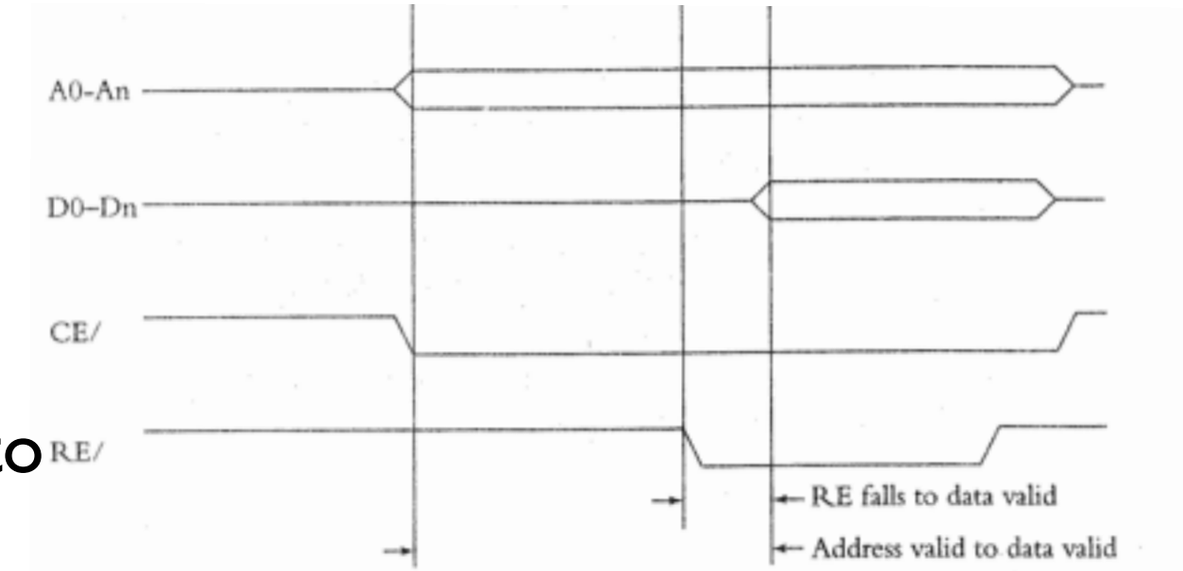
- The boxed part of the signal indicated that multiple values are changing.

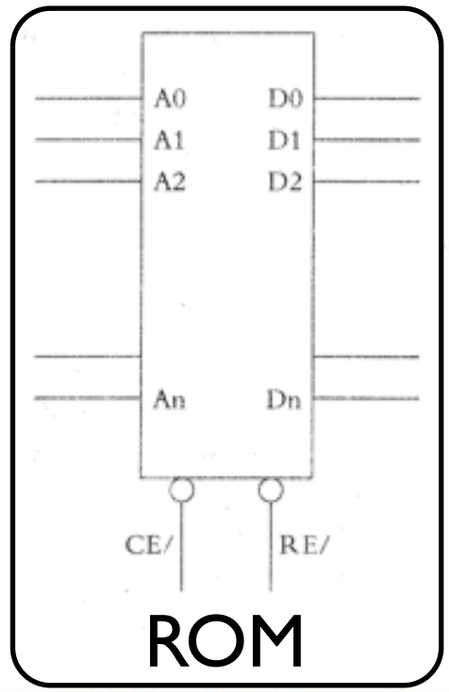
1. Microprocessor drives the ROM address lines with location it wants to fetch from ROM

2. Asserts CE/ signal

3. A little while later it will assert the RE/

4. After some delay, the data lines will contain the appropriate data.





Random Access Memory

- Microprocessor can read data from RAM faster than ROM.
- Microprocessor can write data into RAM, but that data disappears after power is turned off.
- RAM is not a good place to store the program, but raw data.
- Two types of RAM: SRAM and DRAM.

SRAM