- Problem #1: we have latch (or flip-flop) based registers, but they aren't **addressable**.
  - An **address** (or **index**) allows dynamic rather than static reference to some stored datum.
  - By rough analogy to a C program, we have the left-hand side

Listing (C)

```
1 int A0, A1, A2, A3;
2
3 A0 = 0;
4 A1 = 0;
5 A2 = 0;
6 A3 = 0;
```

Listing (C)

```
1 int A[ 4 ];
2
3 A[ 0 ] = 0;
4 A[ 1 ] = 0;
5 A[ 2 ] = 0;
6 A[ 3 ] = 0;
```

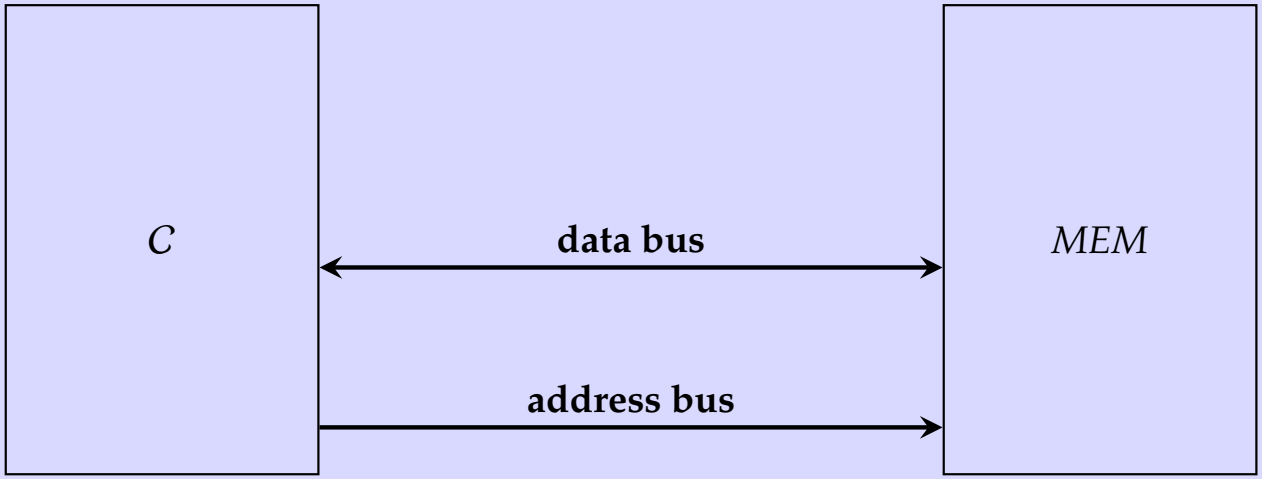
  but want the right-hand side.

- Problem #2: latches and flip-flops need a (relatively) large number of transistors per-bit; to support large capacities, it can make sense to use different components.

- There are various ways to classify a given memory component, e.g.,
  1. **Volatility**
     - **volatile**, meaning the content is lost when the component is powered-off, or
     - **non-volatile**, meaning the content is retained even after the component is powered-off.
  2. **Access type**
     - random versus constrained (e.g., sequential access to content),
     - **Random Access Memory (RAM)** which we can read from *and* write to, and
     - **Read Only Memory (ROM)** which, as suggested by the name, supports reads only.
  3. **Interface type**
     - **synchronous**, where a clock or pre-determined timing information synchronises steps, or
     - **asynchronous**, where a protocol synchronises steps.

- We're (mainly) interested in a how a (volatile, synchronous) RAM component works:

Example



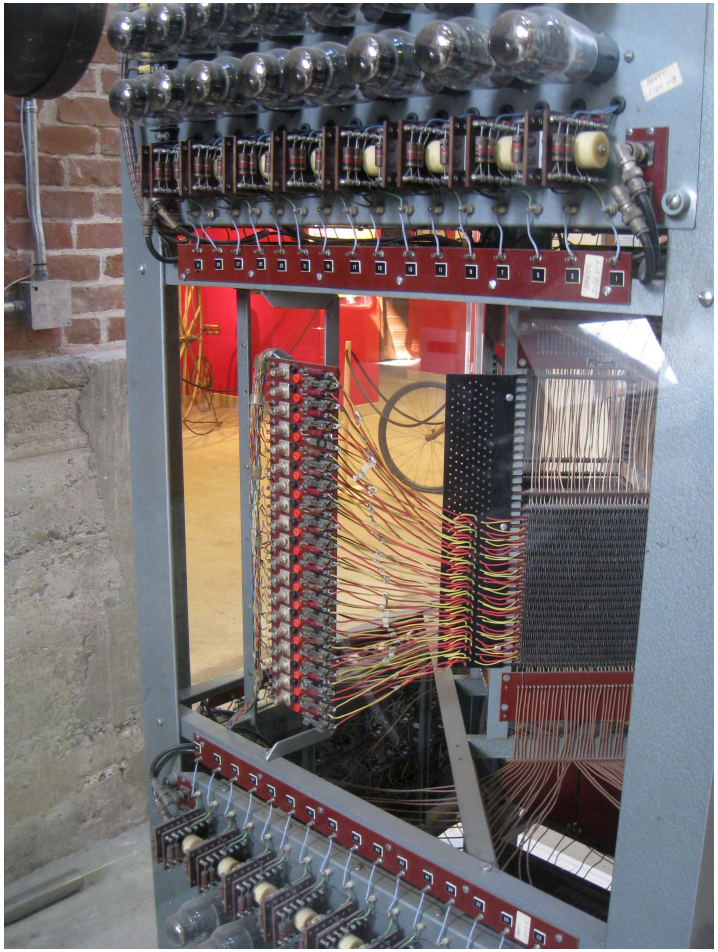
- Goal: a concrete implementation of *MEM* with capacity for $n = 2^{n'}$ addressable words each of w bits (where $n \gg w$).

An Aside: History

- The EDSAC used **delay line** memory, where the rough idea is:
  - Each "line" is a tube of mercury (or something else in which sound waves propagate fairly slowly).
  - Put a speaker at one end to store sound waves into the line, and a microphone at the other to read them out.
  - Values are stored in the sense the corresponding waves take time to propagate; when they get to one end they are either replaced or fed back into the other.
- This is **sequential access** (cf. **random access**): you need to *wait* for the data you want to appear!

http://www.cl.cam.ac.uk/Relics/jpegs/delay_lines.jpg

An Aside: History



- The Whirlwind used **magnetic-core** memory, where the rough idea is:
  - The memory is a matrix of small magnetic rings, or "cores", which can be magnetically polarised to store values.
  - Wires are threaded through the cores to control them, i.e., to store or read values.
  - The magnetic polarisation is retained, so core memory is non-volatile!
- You might still hear main memory termed **core memory** (cf. **core dump**) which is a throw-back to this technology.

---

http://en.wikipedia.org/wiki/File:Project_Whirlwind_-_core_memory,_circa_1951_-_detail_1.JPG

Low-level Implementation (1) – SRAM and DRAM Cells

Definition (**SRAM**)

Static RAM (SRAM) is

- manufacturable in lower densities (i.e., smaller capacity),
- more expensive to manufacture,
- fast(er) access time (resp. lower access latency),
- easy(er) to interface with,
- ideal for use in **register files** and **cache memory**.

Definition (**DRAM**)

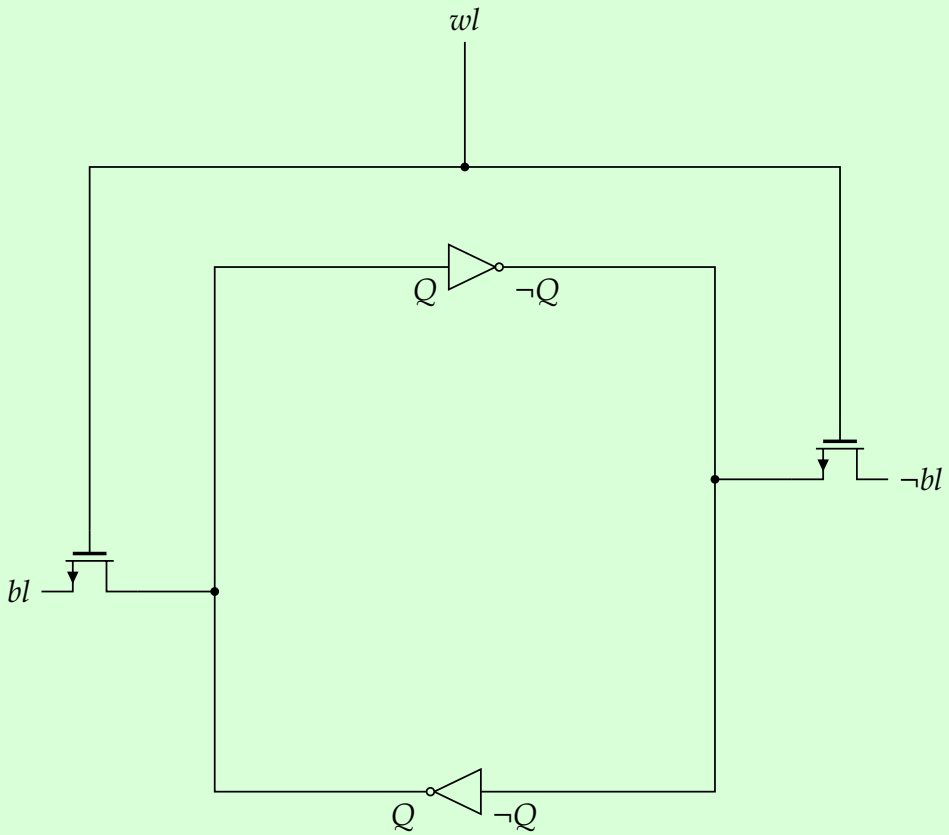
Dynamic RAM (DRAM) is

- manufacturable in higher densities (i.e., larger capacity),
- less expensive to manufacture,
- slow(er) access time (resp. higher access latency),
- hard(er) to interface with,
- ideal for use as **main memory**.

Low-level Implementation (2) – SRAM and DRAM Cells

- Abstractly, an **SRAM cell** resembles two NOT gates ...

Circuit (SRAM cell)

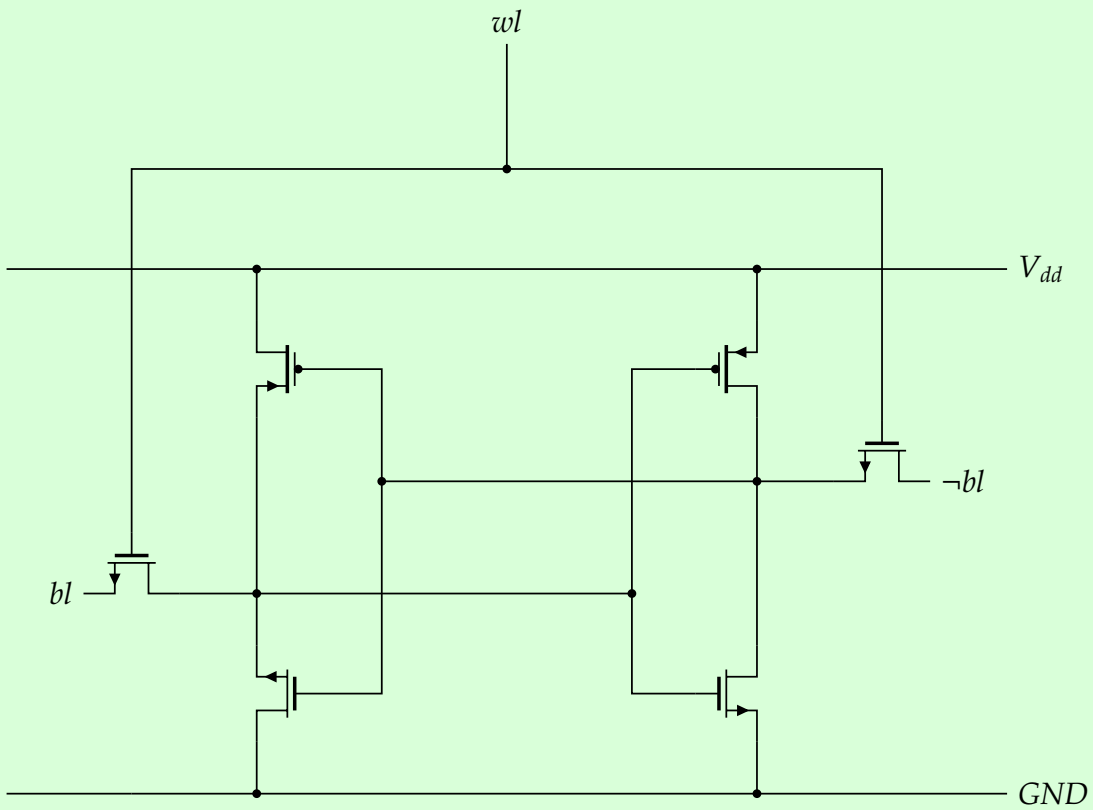


- ... concretely, we can fill in the NOT gates with the transistor-level equivalents; each "6T SRAM cell" requires 6 transistors (cf. ∼ 30 or 40 for a flip-flop).

Low-level Implementation (2) – SRAM and DRAM Cells

- Abstractly, an **SRAM cell** resembles two NOT gates ...

Circuit (SRAM cell)

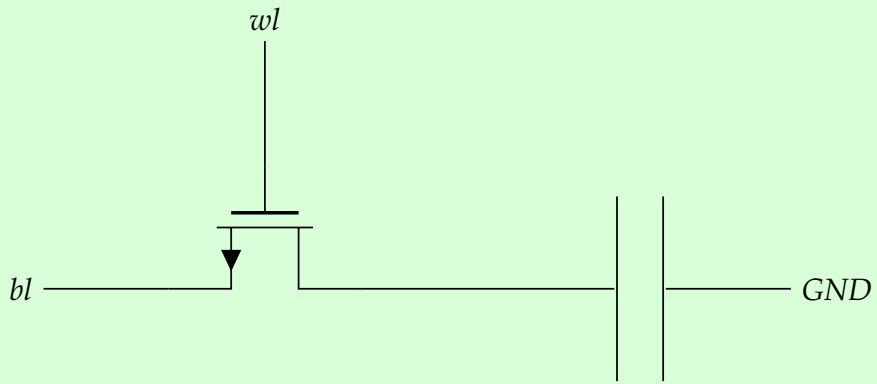


- ... concretely, we can fill in the NOT gates with the transistor-level equivalents; each "6T SRAM cell" requires 6 transistors (cf. ∼ 30 or 40 for a flip-flop).

Low-level Implementation (4) – SRAM and DRAM Cells

- A **DRAM cell** is constructed using 1 transistor and a capacitor:

Circuit (DRAM cell)



High(er)-level Implementation (1) – Cells ⇝ Device

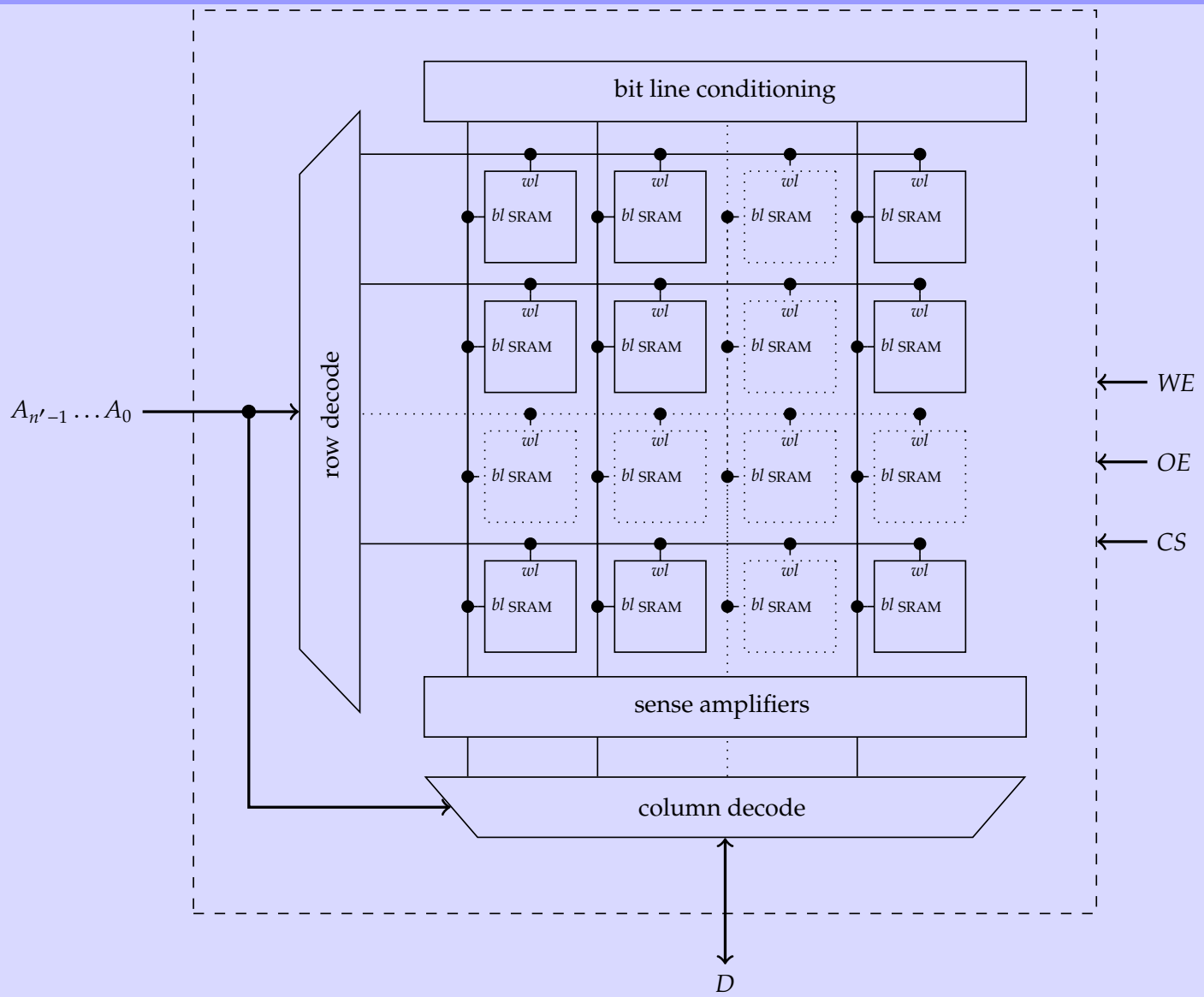
- A **memory device** is constructed from (roughly) three components
  1. a **memory array** (or matrix) of replicated cells with
     - r rows, and
     - c columns

     meaning a $(r \cdot c)$-cell capacity.
  2. a **row decoder** which given an address (de)activates associated cells in that row, and
  3. a **column decoder** which given an address (de)selects associated cells in that column

  plus additional logic to allow use (depending on cell type), e.g.,
  1. **bit line conditioning** to ensure the bit lines are strong enough to be effective,
  2. **sense amplifiers** to ensure output from the array is usable.

High(er)-level Implementation (3) – Cells ⇝ Device, SRAM

Example (an n-cell SRAM memory device)



High(er)-level Implementation (5) – Cells ⇝ Device, SRAM

Austin Semiconductor, Inc.

SRAM
MT5C1001
Limited Availability

1M x 1 SRAM

SRAM MEMORY ARRAY

AVAILABLE AS MILITARY SPECIFICATIONS

- SMD 5962-92316
- MIL-STD-883

FEATURES

- High Speed: 20, 25, 35, and 45
- Battery Backup: 2V data retention
- Low power standby
- Single +5V (±10%) Power Supply
- Easy memory expansion with CE\ and OE\ options.
- All inputs and outputs are TTL compatible
- Three-state output

OPTIONS **MARKING**

- **Timing**

20ns access	-20
25ns access	-25
35ns access	-35
45ns access	-45
55ns access	-55*
70ns access	-70*

- **Package(s)**

Ceramic DIP (400 mil)	C	No. 109
Ceramic LCC	EC	No. 207
Ceramic Flatpack	F	No. 303
Ceramic SOJ	DCJ	No. 501

- **Operating Temperature Ranges**

Industrial (-40°C to +85°C)	IT
Military (-55°C to +125°C)	XT

- 2V data retention/low power — L

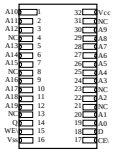
*Electrical characteristics identical to those provided for the 45ns access devices.

For more products and information please visit our web site at
www.austinsemiconductor.com

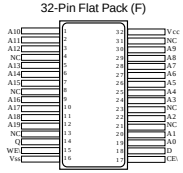
PIN ASSIGNMENT
(Top View)

28-Pin DIP (C)
(400 MIL)

32-Pin LCC (EC)
32-Pin SOJ (DCJ)



32-Pin Flat Pack (F)



GENERAL DESCRIPTION

The MT5C1001 employs low power, high-performance silicon-gate CMOS technology. Static design eliminates the need for external clocks or timing strobes while CMOS circuitry reduces power consumption and provides for greater reliability.

For flexibility in high-speed memory applications, ASI offers chip enable (CE\) and output enable (OE\) capability. These enhancements can place the outputs in High-Z for additional flexibility in system design. Writing to these devices is accomplished when write enable (WE\) and CE\ inputs are both LOW. Reading is accomplished when WE\ remains HIGH while CE\ and OE\ go LOW. The devices offer a reduced power standby mode when disabled. This allows system designs to achieve low standby power requirements.

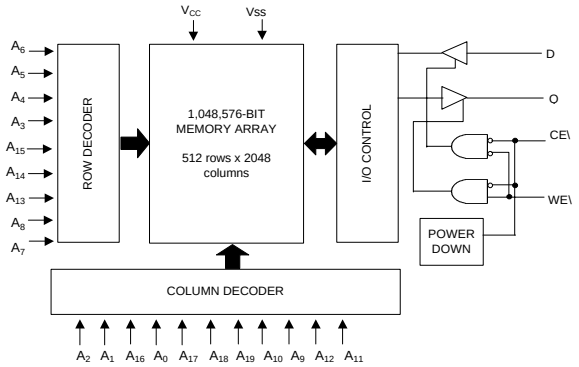
The "L" version provides an approximate 50 percent reduction in CMOS standby current (I_{SBC2}) over the standard version.

All devices operation from a single +5V power supply and all inputs and outputs are fully TTL compatible.

Austin Semiconductor, Inc.

SRAM
MT5C1001
Limited Availability

FUNCTIONAL BLOCK DIAGRAM



TRUTH TABLE

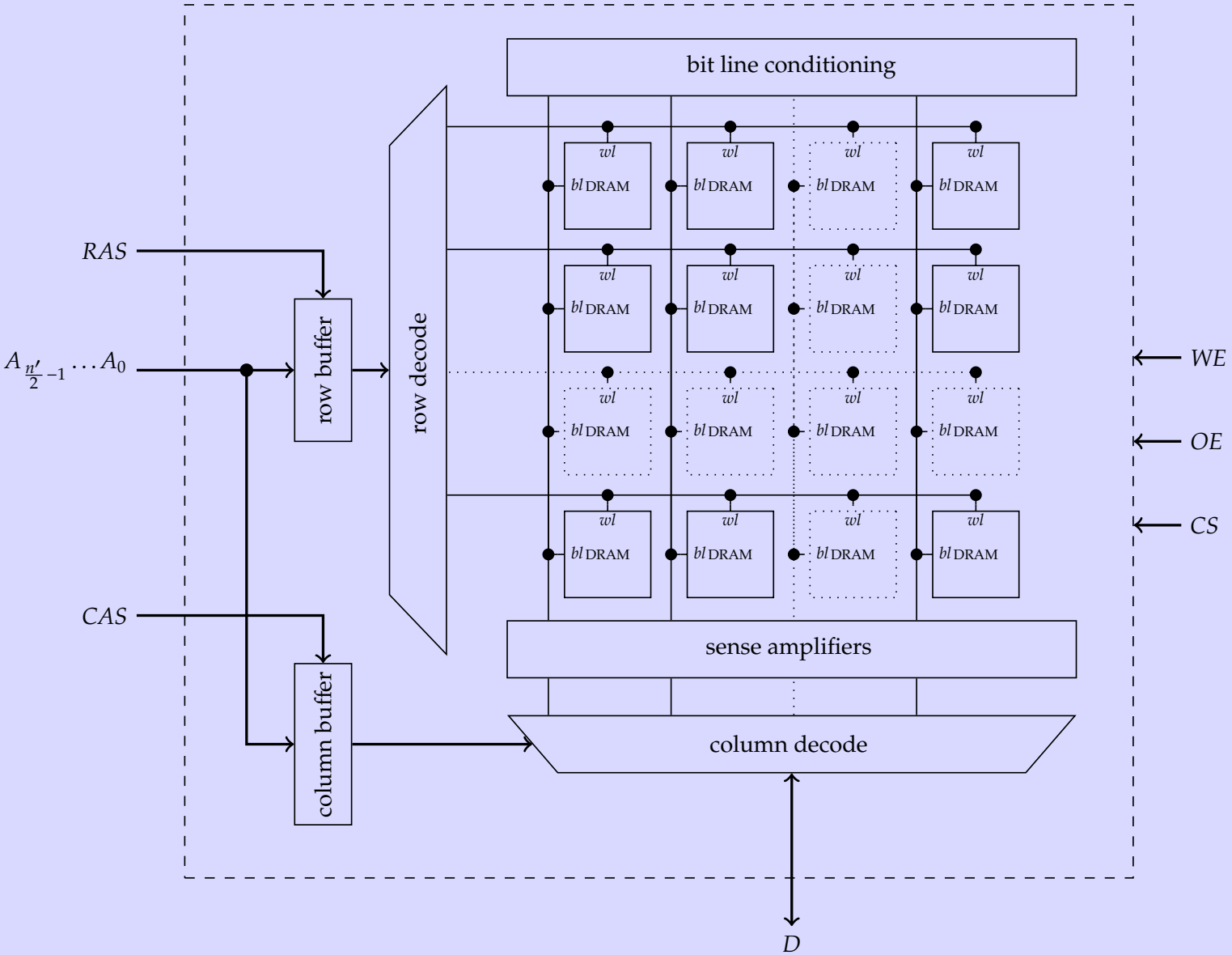
MODE	CE\	WE\	OUTPUT	POWER
STANDBY	H	X	HIGH-Z	STANDBY
READ	L	H	Q	ACTIVE
WRITE	L	L	HIGH-Z	ACTIVE

PIN ASSIGNMENTS

PIN	ASSIGNMENT
A_0-A_{19}	Address Inputs
WE\	Write Enable
CE\	Chip Enable
D	Data Input
Q	Data Output
NC	No Connection
V_{CC}	+5V Power Supply
V_{SS}	Ground

High(er)-level Implementation (7) – Cells ⇝ Device, DRAM

Example (a n-cell DRAM memory device)



High(er)-level Implementation (9) – Cells ⇝ Device, DRAM

DRAM

4 MEG x 1 DRAM

FAST PAGE MODE

AVAILABLE AS MILITARY SPECIFICATONS

- SMD 5962-90622
- MIL-STD-883

FEATURES

- Industry standard x1 pinout, timing, functions and packages
- High-performance, CMOS silicon-gate process
- Single +5V ±10% power supply
- Low-power, 2.5mW standby; 300mW active, typical
- All inputs, outputs and clocks are fully TTL and CMOS compatible
- 1,024-cycle refresh distributed across 16ms
- Refresh modes: $\overline{RAS}$-ONLY, $\overline{CAS}$-BEFORE-$\overline{RAS}$ (CBR), and HIDDEN
- FAST PAGE MODE access cycle
- CBR with $\overline{WE}$ a HIGH (JEDEC test mode capable via WCBR)

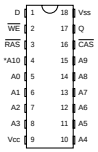
OPTIONS — MARKING

- Timing
  - 70ns access - 7
  - 80ns access - 8
  - 100ns access -10
  - 120ns access -12
- Packages

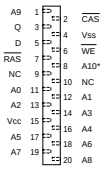
Ceramic DIP (300 mil)	CN	No. 101
Ceramic DIP (400 mil)	C	No. 102
Ceramic LCC	ECN	No. 202
Ceramic SOJ	ECJ	No. 504
Ceramic ZIP	CZ	No. 400
Ceramic Gull Wing	ECG	No. 600

PIN ASSIGNMENT (Top View)

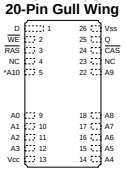
18-Pin DIP



20-Pin ZIP



20-Pin SOJ
20-Pin LCC
20-Pin Gull Wing



*Address not used for $\overline{RAS}$-ONLY REFRESH

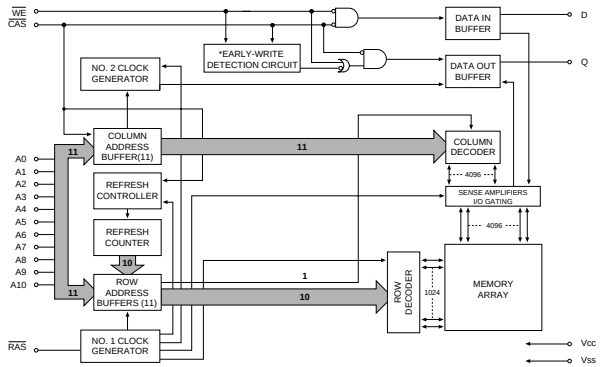
GENERAL DESCRIPTION

The MT4C1004J is a randomly accessed solid-state memory containing 4,194,304 bits organized in a x1 configuration. During READ or WRITE cycles, each bit is uniquely addressed through the 22 address bits which are entered 11 bits (A0-A10) at a time. $\overline{RAS}$ is used to latch the first 11 bits and $\overline{CAS}$ the latter 11 bits. A READ or WRITE cycle is selected with the $\overline{WE}$ input. A logic HIGH on $\overline{WE}$ dictates READ mode while a logic LOW on $\overline{WE}$ dictates WRITE mode. During a WRITE cycle, data-in (D) is latched by the falling edge of $\overline{WE}$ or $\overline{CAS}$, whichever occurs last. If $\overline{WE}$ goes LOW prior to $\overline{CAS}$ going LOW, the output pin remains open (High-Z) until the next $\overline{CAS}$ cycle. If $\overline{WE}$ goes LOW after data reaches the output pin, Q is activated and retains the selected cell data as long as $\overline{CAS}$ remains LOW (regardless of $\overline{WE}$ or $\overline{RAS}$). This LATE-$\overline{WE}$ pulse results in a READ-WRITE cycle. FAST PAGE MODE operations allow faster data operations (READ, WRITE or READ-MODIFY-WRITE) within a row-address (A0-A10) defined page

boundary. The FAST PAGE MODE cycle is always initiated with a row address strobed-in by $\overline{RAS}$ followed by a column address strobed-in by $\overline{CAS}$. $\overline{CAS}$ may be toggled-in by holding $\overline{RAS}$ LOW and strobing-in different column addresses, thus executing faster memory cycles. Returning $\overline{RAS}$ HIGH terminates the FAST PAGE MODE operation. Returning $\overline{RAS}$ and $\overline{CAS}$ HIGH terminates a memory cycle and decreases chip current to a reduced standby level. Also, the chip is preconditioned for the next cycle during the $\overline{RAS}$ HIGH time. Memory cell data is retained in its correct state by maintaining power and executing any $\overline{RAS}$ cycle (READ, WRITE, $\overline{RAS}$-ONLY, $\overline{CAS}$-BEFORE-$\overline{RAS}$, or HIDDEN REFRESH) so that all 1,024 combinations of $\overline{RAS}$ addresses (A0-A9) are executed at least every 16ms, regardless of sequence. The $\overline{CAS}$- BEFORE-$\overline{RAS}$ cycle will invoke the refresh counter for automatic $\overline{RAS}$ addressing.

FUNCTIONAL BLOCK DIAGRAM

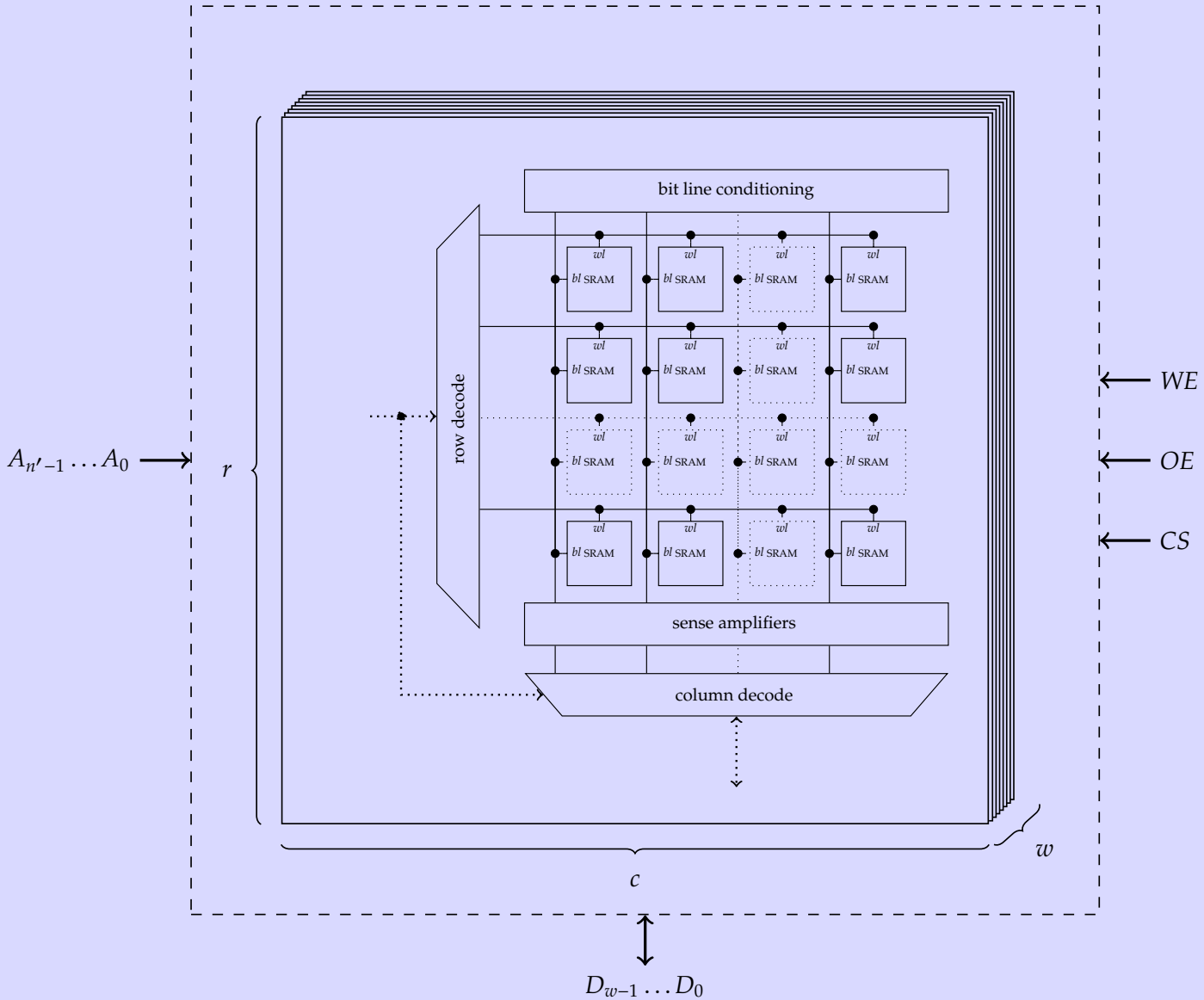
FAST PAGE MODE



*NOTE: $\overline{WE}$ LOW prior to $\overline{CAS}$ LOW, EW detection circuit output is a HIGH (EARLY-WRITE)
$\overline{CAS}$ LOW prior to $\overline{WE}$ LOW, EW detection circuit output is a LOW (LATE-WRITE)

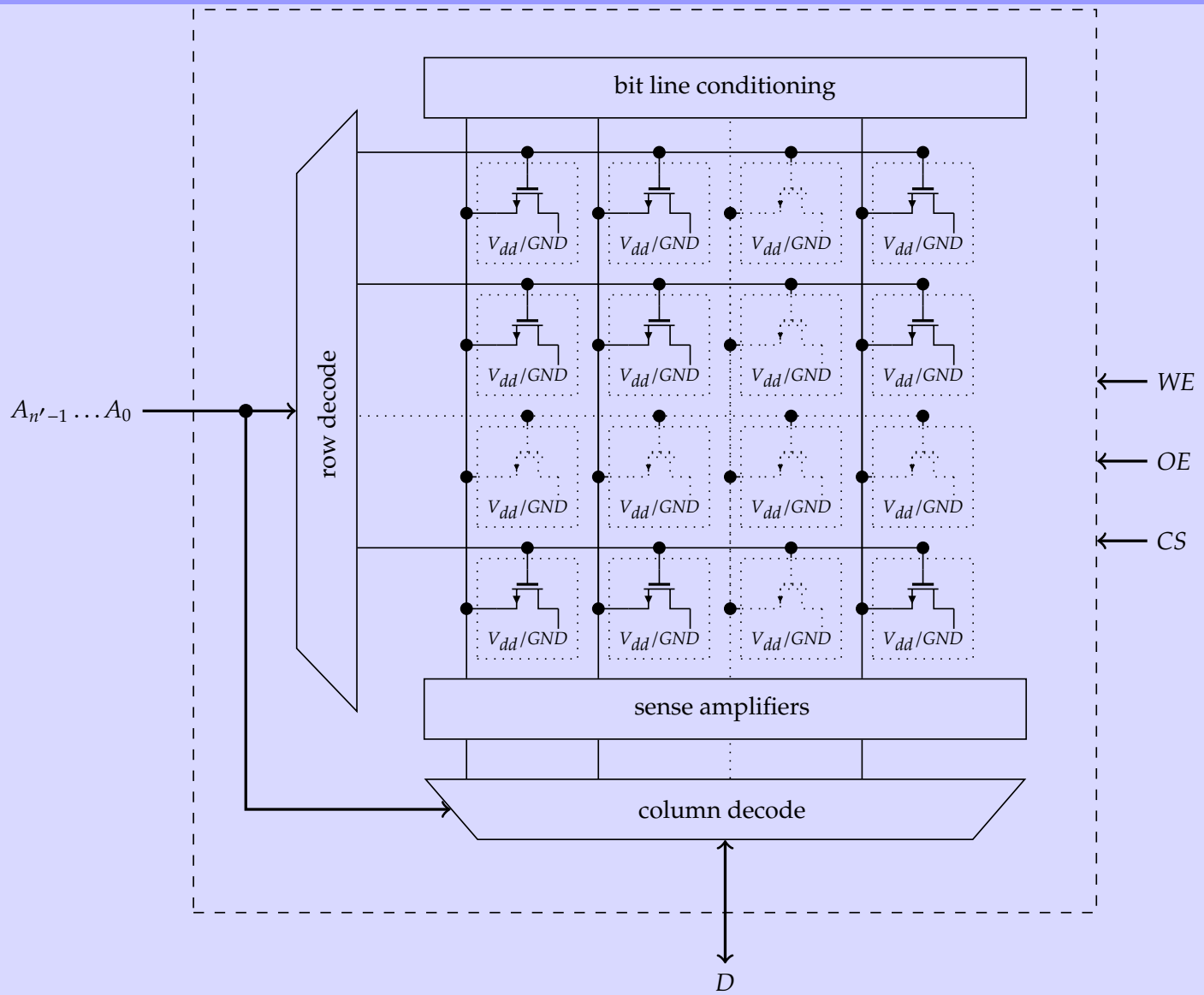
High(er)-level Implementation (11) – Cells ⇝ Device

Example (from a 1-bit to w-bit SRAM device via replication)



An Aside: "what about ROM devices?"

Example (an n-cell ROM memory device)



Conclusions

- Take away points:
  1. The initial goal was an n-element memory of w-bit words; the final solution is motivated by divide-and-conquer, i.e.,
     - 1.1 one or more channels, each backed by
     - 1.2 one or more physical banks, each composed from
     - 1.3 one or more devices, each composed from
     - 1.4 one or more logical banks, of
     - 1.5 one or more arrays, of
     - 1.6 many cells
  2. The major complication is a large range of increasingly detailed options:
     - lots of parameters mean lots of potential trade-offs (e.g., between size, speed and power consumption),
     - need to take care of detail: there are so many cells, any minor change can have major consequences!
  3. Even so, there is just one key concept: we have some cells, and however they are organised we just need to identify and use the right cells given some address.

References and Further Reading

[1] Wikipedia: Computer memory.
http://en.wikipedia.org/wiki/Category:Computer_memory.

[2] Wikipedia: Dynamic random access memory.
http://en.wikipedia.org/wiki/Dynamic_random-access_memory.

[3] Wikipedia: Memory geometry.
http://en.wikipedia.org/wiki/Memory_geometry.

[4] Wikipedia: Static random access memory.
http://en.wikipedia.org/wiki/Static_random_access_memory.

[5] U. Drepper.
What every programmer should know about memory.
http://www.akkadia.org/drepper/cpumemory.pdf.

[6] D. Page.
Chapter 8: Memory and storage.
In *A Practical Introduction to Computer Architecture*. Springer-Verlag, 1st edition, 2009.

[7] W. Stallings.
Chapter 5: Internal memory.
In *Computer Organisation and Architecture*. Prentice-Hall, 9th edition, 2013.

[8] A.S. Tanenbaum.
Section 3.3.4: Memory organisation.
In *Structured Computer Organisation* [11].

References and Further Reading

[9] A.S. Tanenbaum.
Section 3.3.5: Memory chips.
In *Structured Computer Organisation* [11].

[10] A.S. Tanenbaum.
Section 3.3.6: RAMs and ROMs.
In *Structured Computer Organisation* [11].

[11] A.S. Tanenbaum.
Structured Computer Organisation.
Prentice-Hall, 6th edition, 2012.