Memory Design

● Outline
  – Memory Architecture
  – Memory cell types
    * Static RAM (SRAM)
      · Single and multi-port SRAM
      · Content Addressable Memory (CAM)
      · Dynamics of an SRAM cell
    * Dynamic RAM (DRAM)
  – Sense Amplifiers
  – Memory address decoders
Memory Architectures

- A semiconductor memory stores $n$ words each of width $a$ bits. Common values of $a$ are 1, 4, 8. Such a memory is usually referred to as a $n \times a$ memory.

- Each bit is implemented as a memory cell.

- Usually $n$ is a power of 2.

- The user may write to or read from any word.

- If we want a memory with $n$ words of width $ab$ bits, we can use $b$ memory chips as above, and address them similarly. These memory units are referred to as banks.

- Relevant parameters of a memory are:
  - **Read access time** - the time between when a word is requested and when it is available.
  - **Write access time** - the time between when a word is requested to be written and when it is actually written.
  - **Read/write access time** - the minimum time between successive read or write operations.
• Sometimes memories have multiple **ports**. In this case the same data can be accessed by several “customers” at once.

  – The read cycle may be staggered in time so that a read can occur just after a write. Such a multi-port memory is significantly challenging to design

  – Several ports may read the memory at once. Read access times drop with each additional port attempting to read a memory word

  – A single write is allowed to any given word at a time.

  – Several writes to **different words** are allowed. This poses significant problems with bit and word-line cross-talk

  – Multi-port memories are usually static

  – It is hard to design multi-port memories with more than 2 ports.

  – Routing becomes harder as the number of ports increases

  – Multi-port memories are used in register files for RISC microprocessors, since very high bandwidth access to registers is required in such designs
- Memory is often organized in a **hierarchy**. So if a item of data is not found on level $i$, then it we search for it in level $i+1$ memory. Usually higher levels of memory are slower and larger. The last level of memory storage is usually a hard disk in such an arrangement.

- What does the inside of the $n$ word memory look like (where a word is $a$ bits)?
  
  - Simple-minded solution - Arrange the words **linearly**

  ![Memory Cell Diagram]

  - But this requires $n$ select lines, and $n$ is large.
  - This motivates the use of a **row decoder**
So then the number of address bits is $p = \log_2(n)$. Hence the number of input pins to the memory drops exponentially. The decoder ensures that only one of the select lines $s_i$ is active at a time.

However, with this approach, the memory is tall and thin. It is $n$ words by $a$ bits, with $n >> a$.

For a memory with $n = 2^{20}$ and $a = 8$, the aspect ratio is roughly $2^{20}/2^3 = 2^{17} \approx 128,000$.

This is a problem for two reasons:

* As a general rule, for the same circuit area, roughly square ICs result in best utilization of wafer area, are easy to manipulate and are faster since the longest wire length is smaller.
* The thin and tall memory cell has very long bit-lines, resulting in highly capacitive wires and large output delays.

– So we want to make the memory more or less square.

– To do this we use **row and column decoders** as shown below

Here, \( p = \log_2(n) \)
The idea is to array $2^k$ words horizontally.

We will read/write from one of these words as indicated by the column decoder.

We will address one horizontal array of words as indicated by the row decoder.

We have $2^{p-k}$ rows, and each row has $2^k$ words. Hence we have a total of $2^{p-k} \cdot 2^k = 2^p = n$ words.

For a given $n = 2^p$, as we increase $k$, we need less rows. So we can control the aspect ratio of the resulting circuit.

We choose $k$ such that the resulting memory is roughly square.

Since word and bit lines are long, they need drivers.

Also, for some types of memory cells, we need sense amplifiers as well to determine the value of read data.

This is great, but for very large memories, a monolithic architecture as the one described above may be too slow due to long wires.

A solution often utilized is to break the memory up into blocks as shown below:
This block-style design ensures that the longest wire in the design is smaller than in a monolithic memory. Hence it is faster.
Static RAM (SRAM)

- Data, once written, need not be refreshed
- Recall the definition of bit and word lines
- SRAM cells are among the larger memory cells
- A SRAM is utilized in applications where
  - Memory size is smaller but fast and clean signals are desired.
  - Typical application is a register file
- Generically, a SRAM cell looks like:
- Effectively it is a pair of cross-coupled inverters that store state.

- The SRAM cell operates as follows:
  
  - **Writing data:** Place the desired data \( A \) on the \( BL \) line, and \( \overline{A} \) on \( \overline{BL} \). Then assert \( WL \). This causes the node \( a \) to attain the \( A \) value, and the node \( b \) attains the \( \overline{A} \) value. \( WL \) is de-asserted now. The cell is in steady state.

  - **Reading data:** First we precharge both \( BL \) and \( \overline{BL} \). Now we assert \( WL \). Depending on the state of \( a \) and \( b \), either \( BL \) or \( \overline{BL} \) is pulled low, the other stays high.
• The load $R$ can be implemented in several ways:
  
  — If it is implemented as a PMOS device as shown below, we get a 6-T SRAM cell (6-T stands for 6-transistor).

  — If it is implemented as a POLY2 resistor then we have a 4-T cell. POLY2 is a highly resistive form of POLY. Its high resistivity comes from the fact that it is undoped (unlike regular POLY which gets doped when transistor diffusions are doped).

  — If the cell is single-ended as opposed to differential, we get a 5-T (or 3-T) cell. Such a cell
does not write a “1” value (placed on the \( BL \)) fast enough, since it the NMOS passgate does not propagate “1” values without degradation. To solve this, \( WL \) is driven above \( VDD \) during the write cycle. This adds design complexity and hence 5-T / 3-T cells are relatively rare.

\[ \text{If we add more bit and word lines, we get a multi-ported SRAM} \]
Content Addressable Memory (CAM)

- Variant of SRAM, used in caches
- It can be written to and read from just as a SRAM
- In addition, the content of the cell can be “queried”. If there is a match, the cell indicates this.
• If we want to do a match operation, we place the data \((A)\) on \(\overline{BL}\), and \(\overline{A}\) on \(BL\). This is unlike what we do for the write operation!

• \textit{MATCH} is precharged. If there is a match, then the \textit{MATCH} line stays high since the match transistor is off.

• If there is no match, then the match transistor turns on and the \textit{MATCH} line is pulled low
Dynamics of a SRAM cell

- Architecture determines wire lengths, and hence read/write times

- Large pass transistors also result in capacitive word-lines and bitlines (pass transistor diffusion contributes to the $BL$ capacitance, and its gate contributes to the $WL$ capacitance). Also it contributes to larger cell area.

- Small pass transistors mean slower writes to the cell

- Usually the pass transistors are minimum-sized

- Large transistors in the cell inverter result in faster reads, since they can switch $BL$ faster. However they result in larger cells.

- Fall time for the $BL$ is $2\tau_f = \frac{8C_m}{\beta_NV_{DD}}$, where $\beta_N$ is the gain of a minimum-sized NMOS device. The factor of 2 is because the pass-gate and pull-down transistors are in series when the $BL$ is pulled low.
DRAM memory cell

- In the SRAM cell, a heavy area penalty resulted from the use of resistive load devices.

- If we do away with these devices, we would slowly lose the charge stored in the cell, but save significant cell area. This motivates a DRAM cell.

- Since charge will be lost by decay, we need to refresh the charge periodically (typically every few milliseconds)

- The increased circuit complexity is more than compensated by the decreased cell area which allows us to implement significantly larger memories.

- Typically we encounter 3-T and 1-T DRAM cells. Let's first discuss the 3-T DRAM cell.
- Removing the load devices from the 6-T SRAM cell, we get a “4-T DRAM” cell. Recognizing that this cell has redundancy (it stores the value of the bit and its complement) we can save another device to get the 3-T DRAM cell.
• **Write**: put date on $BL1$, assert $WWL$ (write word-line). Data is retained on the storage capacitor $C_T$ after $WWL$ goes low.

• **Read**: Precharge $BL2$ (alternately, $BL2$ is resistively held high). Assert $RWL$ (read word-line). If $C_T$ was $VDD$, then $BL2$ is pulled low, else it stays high. Note that the complement of the stored value is written to $BL2$.

• A refresh operation comprises a read followed by a write. A refresh is done by reading the stored value, inverting it, and writing the inverted value back into the cell.

• Advantages are:
  
  – Full-rail read operation.
  
  – Device size ratios are not important unlike in 1-T cells. This is because the operation of this cell does not rely on charge sharing.
  
  – Reads are non-destructive, unlike in 1-T cells.
  
  – Does not require specialized fabrication processes unlike the 1-T cell.
  
  – For the above reasons, the 3-T cell is robust and reliable. This is why it is a preferred choice for implementing embedded memory in some ASICs.
— Also it is smaller than the 6-T SRAM cell.

— The 3-T DRAM is single-ended unlike the differential SRAM cells. This can result in some area savings since less signals are to be routed to each cell.

• Disadvantages:

— WWL may need to be bootstrapped (i.e. driven higher than VDD) in order not to incur a threshold drop due to the passgate device whose gate is connected to WWL.

— The cell area is still reasonably large. We can do better.

• Some modifications to the base 3-T cell are:

— Combine BL1 and BL2 into a single bitline. The write or read cycles are unaltered, but the refresh cycle becomes longer and more complex.

— Alternately, combined WWL and RWL into a single wordline. Reading the cell results in a refresh, in this case. We need to prevent the writing of the cell before reading it.
• The 3-T DRAM cell can be made smaller yet…

• Consider a single transistor (1-T) DRAM cell
  – Results in significantly smaller DRAM cell than the 3-T cell
  – Workhorse for commercial DRAM ICs

  ![Diagram of DRAM cell with capacitances](image)

  - **Write**: put data on $BL$, assert $WL$

  - **Read**: precharge $BL$ to a intermediate voltage $V_{pch}$. Now assert $WL$. This causes charge redistribution between $C_T$ and $C_{BL}$.

  - How do we compute the resultant voltages? Answer - refer class notes section on “Charge Sharing”.
• As computed in the Charge Sharing notes, the read voltage when the cell stores a high value is
\[
V_{final}^1 = \frac{V_{pch} C_{BL} + (VDD - V_T) C_T}{(C_{BL} + C_T)}.
\]

• In other words,
\[
V_{final}^1 - V_{pch} = ((VDD - V_T) - V_{pch}) \frac{C_T}{(C_{BL} + C_T)}.
\]

• \( \frac{C_T}{C_T + C_{BL}} \) is referred to as the charge transfer ratio, and it is between 1% and 10%.

• Accordingly, the voltage excursion of BL when a logic 1 is read, is 10s or 100s of millivolts.

• It is common to bootstrap the WL to \( VDD \), so that more charge is stored on the capacitor \( C_T \) during a write. In that case, the term \( (VDD - V_T) \) in the above expressions for \( V_{final}^1 \) should be replaced with \( V_{bootstrap} - V_T \).

• Since BL does not exhibit full-rail transitions we need to use a sense amplifier to re-generate clean logic values.

• Reading is destructive, so we re-assert the sensed value of the cell on BL after reading it, and assert WL to restore the charge in the DRAM cell.

• This is how refresh is done as well.
• The major challenge in DRAM design is creating a large $C_T$.
  
  — Hard to make a large enough $C_T$ in a small area.
  
  — Some solutions include

  * **Trench capacitors**: Etch a trench (few $\mu$m deep) in the substrate, deposit electrodes and insulators to create a capacitor. Recall that the junction depth is usually less than 1 $\mu$m.

  * **Stacked capacitors**: Deposit the electrodes **above** the transistor device, saving area. Usually several layers of electrodes are deposited, in a mesh or comb-like fashion.

  — In both cases, capacitances of 10s of $fF$ are realized.

  — The down-side is need for highly complex fabrication processes in order to create these large capacitors.

  — Yield drops as well, due to the complex processing steps.

• Advantages:
  
  — Extremely small cell, hence we can make large memories

  — Used extensively in DRAM ICs today
• Disadvantages:
  – Bootstrapping of \( WL \) increases complexity
  – Complex processing required, hence these cells are used for DRAM chips and not for memory which is used along with logic on the same die (i.e. embedded memory).
  – Needs sense amplifiers unlike some other memory cells
Sense Amplifiers

- To rapidly restore logical values, we use a sense amplifier

- In essence, a sense amplifier is a differential amplifier

- Sense amplifiers are used in DRAMs, and in larger SRAMs.
  - In DRAMs, their use is imperative since $BL$ is not a full-rail signal
  - In larger SRAMs (even though bitlines have full-rail transitions), their use is required since the bitlines are slow to transition. A sense amplifier speeds up this transition.

- A sense amplifier along with its bitlines is shown below:
- When PCHG is low, bitlines precharge, and get equalized by the EQ device.

- When PCHG is high, WL is asserted, and one of $BL$ or $\overline{BL}$ pulls low.
• To speed up the sensing of this transition, a sense amplifier is used. It generates differential outputs $out$ and $\overline{out}$.

• A simple sense amplifier is shown below:

![Sense Amplifier Diagram](image)

• Recall its similarity with the DCVSL structure. In fact, it is effectively a DCVSL buffer/inverter.

• Once the read operation has begun, $ENABLE$ is asserted.

• Another sense amplifier is shown below:
• Once the read operation has begun, ENABLE is asserted.

• The cross-coupled inverters provide positive feedback and pull the bitlines to their full rail values rapidly.

• Notice that the preceding sense amplifiers had differential inputs and differential outputs.

• But DRAM cells generate a single-ended BL.

• How do we provide the “other” input to the sense amplifier while sensing the BL?
• Answer - we appropriately bias the “other” input of the sense amplifier to $V_{pch}$.

• A creative solution is to place sense amplifiers in the center of a $BL$, with an equal number of DRAM cells on either side of it.
  
  — The left half of $BL$ is connected to $in$, and the right half to $\overline{in}$.
  
  — When bitlines are precharged, the two halves are equilibrated across the sense amplifier
  
  — This ensures that the sense amplifier inputs have equal voltages, before a read operation is performed
  
  — An additional advantage is that $C_{BL}$ is halved for each side of the bitline, allowing us to double the size of the resulting memory

• The second type of sense amplifier is commonly used for DRAMS, since it pulls $BL$ to its full-rail value while sensing its value. This eases the refresh operation.
Memory Address Decoders

- From the memory block diagram, recall that we need row and column decoders to address the appropriate memory cells.

- Decoders significantly impact memory performance

- Row decoders are simply $1 - of - 2^{p-k}$ decoders, or demultiplexors

- Column decoders are $2^k - to - 1$ multiplexors

- Let's consider row decoders first.

- Consider a 2-input row decoder. It has 2 inputs, $A_0$ and $A_1$. It should have 4 outputs, $WL_0$, $WL_1$, $WL_2$ and $WL_3$.

- If $A_0 = 1$ and $A_1 = 1$, then $WL_3$ should be 1, and all other $WL_i$ should be 0.

- If $A_0 = 1$ and $A_1 = 0$, then $WL_1$ should be 1, and all other $WL_i$ should be 0.

- Similarly for other values of $A_0$ and $A_1$. 
• Static or dynamic decoders can be envisioned - we describe static decoders below

• Row decoders can be implemented as NOR or NAND arrays, much like the NOR-NOR and NAND-NAND PLAs which we studied, but **without** the output plane

• $m$-input NOR decoders have effectively $2^m$ $m$-input NOR gates as shown below for a 2-input decoder.

• For $m$-input NAND or NOR decoders we require $O(m \cdot 2^m)$ transistors. This is expensive for reasonable $m$. 
- $m$-input NAND decoders have effectively $2^m$ $m$-input NAND gates as shown below for a 2-input decoder.

- Note that these decoders generate complemented $WL_i$ signals.
- NAND decoders are slower since body effect problems are exhibited.

- NOR decoders are larger since GND wires are required between address lines.

- NOR decoders consume more power since all $WL_i$’s except one pull low at every clock cycle. In NAND decoders, only one $\overline{WL_i}$ pulls low each cycle.

- To reduce the area (and the number of transistors) in a decoder, **pre-decoder** structure is often employed.

- Such a structure is illustrated below
• The use of a multi-level circuit reduces the transistor count. The transistor count for the above decoder is \((1024 \times 6) + (5 \times 4 \times 4) = 6224\). This is about half that of a NAND or NOR decoder.

• The delay of the circuit reduces significantly as well. This is because the number of inputs to the NAND gate is halved and the vertical address lines’ loads are reduced since \(1024/4 = 256\) connections are required per vertical address line.

• Pre-decoders also help reduce power consumption by enabling the designer to selectively disable a decoder.
• Pre-decoders are extensively used in larger memories.

• Now let's consider column decoders.

• $k$ input column decoders are $2^k - 1$ muxes.

• One implementation uses a (row-style) decoder at the core:

  ![Diagram](image)

  - This decoder has only one transistor in series for every bit-line, resulting in a faster column decoder.
• This decoder requires $O(k \cdot 2^k)$ transistors and is therefore expensive.

• A more common style of column decoder is the *tree decoder*.

• Note that no (row-style) decoder is required.

• The number of devices needed is equal to $2^k + 2^{k-1} + 2^{k-1} + \cdots + 4 + 2 = 2^{k+1} - 2$. This is significantly less than the previous column decoder.
- The downside is that each path has $k$ series-connected devices, slowing down the output. Note the similarity to PTL.

- Buffering intermediate stages is often used to speed up the output response time.