Array Architecture

- $2^n$ words of $2^m$ bits each
- If $n \gg m$, fold by $2^k$ into fewer rows of more columns

- Good regularity easy to design
- Very high density if good cells are used
12-Transistor SRAM Cell

- Basic building block: SRAM Cell
  - Holds one bit of information, like a latch
  - Must be read and written
- 12-transistor (12T) SRAM cell
  - Use a simple latch connected to bitline
  - $46 \times 75$ λ unit cell
6-Transistor SRAM Cell

- Cell size accounts for most of array size
  - Reduce cell size at expense of complexity

6T SRAM Cell
- Used in most commercial chips
- Data stored in cross-coupled inverters

- Read:
  - Precharge bit, bit_b
  - Raise wordline

- Write
  - Drive data onto bit, bit_b
  - Raise wordline
SRAM Read

- Precharge both bitlines high
- Then turn on wordline
- One of two bitlines will be pulled down by the cell
- Example: \( A = 0, A_b = 1 \)
  - Bit discharges, bit\(_b\) stays high
  - But \( A \) bumps up slightly
- **Read stability**
  - \( A \) must not flip
  - \( N1 >> N2 \)

![SRAM Read Diagram](image-url)
SRAM Write

- Drive one bitline high, the other low
- Then turn on wordline
- Bitlines overpower cell with new value
- Example: $A = 0$, $A_b = 1$, bit = 1, bit$_b$ = 0
  - Force $A_b$ low, then $A$ rises high
- **Writability**
  - Must overpower feedback inverter
  - $N2 \gg P1$

![Diagram of SRAM write operation]

![Graph showing the write operation over time]
SRAM Sizing

- High bitlines must not overpower inverters during reads
- But low bitlines must write new value into cell
SRAM Column Example

Read

Write

Bitline Conditioning

More Cells

SRAM Cell

word_q1

bit_v1f

out_v1r

φ₁

φ₂

word_q1

bit_v1f

out_v1r

data_s1

Write
Cell size is critical: $26 \times 45 \lambda$ (even smaller in industry)

Tile cells sharing $V_{DD}$, GND, bitline contacts
Decoders

- $n : 2^n$ decoder consists of $2^n$ n-input AND gates
  - One needed for each row of memory
  - Build AND from NAND or NOR gates

Static CMOS

Pseudo-nMOS
Decoders must be pitch-matched to SRAM cell
- Requires very skinny gates
Large Decoders

- For \( n > 4 \), NAND gates become slow
- Break large gates into multiple smaller gates
Many of the gates are redundant
- Factor out common gates into predecoder
- Saves area
- Same path effort
Column Circuitry

- Some circuitry is required for each column
  - Bitline conditioning
  - Sense amplifiers
  - Column multiplexing

Bitline Conditioning

- Precharge bitlines high before reads
- Equalize bitlines to minimize voltage difference when using sense amplifiers
Sense Amplifiers

- Bitlines have many cells attached
  - Example, 32-kbit SRAM has 256 rows x 128 cols
  - 128 cells on each bitline
- \( t_{pd} \propto (C/I) \Delta V \)
  - Even with shared diffusion contacts, 64C of diffusion capacitance (big C)
  - Discharged slowly through small transistors (small I)
- Sense amplifiers are triggered on a small voltage swing (reduce \( \Delta V \))

Example: Differential Pair Amplifier

- Differential pair requires no clock
- But always dissipates static power

![Differential Pair Amplifier Diagram]
Clocked Sense Amplifier

- Clocked sense amp saves power
- Requires sense_clk after enough bitline swing
- Isolation transistors cut off large bitline capacitance
Twisted Bitlines

- Sense amplifiers also amplify noise
  - Coupling noise is severe in modern processes
  - Try to couple equally onto bit and bit_b
  - Done by twisting bitlines

![Diagram showing twisted bitlines](image-url)
Recall that array may be folded for good aspect ratio

Example: 2K word × 16 array folded into 256 rows × 128 columns

- Must select 16 output bits from the 128 columns
- Requires 16 8:1 column multiplexers
- Column MUX can use pass transistors
  - Use nMOS only, precharge outputs
- One design is to use $k$ series transistors for $2^k : 1$ mux
  - No external decoder logic needed

---

**Diagram:**

```
  A0  B0  B1  B2  B3  B4  B5  B6  B7
  A0  
  A1  
  A1  
  A2  Y
  A2

  B0  B1  B2  B3  B4  B5  B6  B7
  B0  
  B1  
  B2  
  B3  
  B4  
  B5  
  B6  

  Y
  Y
to sense amps and write circuits
```
Single Pass-Gate Multiplexer

- Eliminate series transistors with separate decoder
Example: 2-Way Muxed SRAM
We have considered single-ported SRAM
- One read or one write on each cycle

Multiported SRAMs are needed for register files

Examples:
- Multicycle MIPS must read two sources or write a result on some cycles
- Pipelined MIPS must read two sources and write a third result each cycle
- Superscalar MIPS must read and write many sources and results each cycle
Simple dual-ported SRAM
- Two independent single-ended reads
- Or one differential write

Do two reads and one write by time multiplexing
- Read during ph1, write during ph2
Multi-Ported SRAM

- Adding more access transistors hurts read stability
- Multiported SRAM isolates reads from state node
- Single-ended design minimizes number of bitlines
Serial Access Memories

- Serial access memories do not use an address
  - Shift Registers
  - Tapped Delay Lines
  - Serial In Parallel Out (SIPO)
  - Parallel In Serial Out (PISO)
  - Queues (FIFO, LIFO (Stacks))
- Some of these used in circuitry for communications

Shift Registers Store and Delay Data

- Simple design: cascade of registers
- **Watch your hold times!**

![Shift Register Diagram]

- clk
- Din
  - 8
- Dout
Denser Shift Registers

- Flip-flops are not very area-efficient
- For large shift registers, keep data in SRAM instead
- Move R/W pointers to RAM rather than data
  - Initialize read address to first entry, write to last
  - Increment address on each cycle
- Shifter register with a programmable number of stages
- Set number of stages with delay controls to mux
  - Example, 0 – 63 stages of delay

![Tapped Delay Line Diagram]

 clk

 Din

 SR32
 delay5

 SR16
 delay4

 SR8
 delay3

 SR4
 delay2

 SR2
 delay1

 SR1
 delay0

 Dout
Serial/Parallel Conversion

**Serial In Parallel Out**

- 1-bit shift register reads in serial data
  - After N steps, presents N-bit parallel output

![Diagram of Serial In Parallel Out](image)

**Parallel In Serial Out**

- Load all N bits in parallel when shift = 0
  - Then shift one bit out per cycle
Queues

- Queues allow data to be read and written at different rates.
- Read, Write each use their own clock, data.
- Queue indicates whether it is full or empty.
- Build with SRAM and read/write counters (pointer.

![Diagram of Queue structure with inputs and outputs]
**First In First Out (FIFO)**
- Initialize read and write pointers to first element
- Queue is EMPTY
- On write, increment write pointer
- If write almost catches read, Queue is FULL
- On read, increment read pointer

**Last In First Out (FIFO)**
- Also called a stack
- Use a single stack pointer for read and write
Remove the two p-channel transistors from the static RAM cell to get a four-transistor dynamic RAM cell

Data stored as charge on gate capacitors (complementary nodes)

Data must be refreshed regularly

Dynamic cells must be designed very carefully
3-Transistor Dynamic RAM Cell

- Data stored on the gate of a transistor
- Need two additional transistors, one for write and the other for read control
1-Transistor Dynamic RAM Cell

- Cannot get any smaller than this: data stored on a (trench) capacitor C, need a transistor to control data
- Bit line normally precharged to $\frac{1}{2}V_{DD}$ (need a well-designed sense amplifier)
- Value of capacitances must be chosen very carefully; voltages on stored bit and bit-line affected by charge sharing
Single-Event Upsets

- High-energy particle produces electron-hole pairs in substrate; when collected at source and drain, will cause current pulse
- Cosmic Radiation

A “bit-flip” can occur in the memory cell due to the charge generated by the particle – called a “single-event upset”

- Seen in spacecraft electronics in the past, now in computers on the ground

Source: Aerospace Corporation

Interaction of a Cosmic Ray and Silicon

Energetic Charged particle

A simple memory element

MOS Transistor
Example, from ARM

- Single Port SRAM
- Dual Port SRAM
- Single Port Register file
- Two Port Register file
- Via and Diffusion Programmable ROM

Memory Compilers

- Automatically generate memory structures
- High density, high speed and low power SRAMs
- Over 15 different foundries and 65 process variants from 28nm to 250nm
ARM Embedded Memory IP