

# DRAM and DDR Memory

## Dynamic Random Access Memory Architecture

# Loading and Prefetching

## Loading:

- ▶ Data copied from SSD to DRAM during program startup
- ▶ CPU can only process data after it's in DRAM
- ▶ Takes time (loading bar appears)

## Prefetching:

- ▶ Moving data to DRAM before it's needed
- ▶ Anticipates future data requirements
- ▶ Minimizes wait time during execution

# DRAM Module Overview (DIMM)

## Structure:

- ▶ DIMM = Dual Inline Memory Module
- ▶ 8 DRAM chips per module (typical)
- ▶ Connected to CPU via memory channels

## Connection:

- ▶ Two independent memory channels
- ▶ Memory controller in CPU manages communication



# Memory Channels in DDR5

## Channel Structure:

- ▶ Each memory channel divided into Channel A and Channel B
- ▶ Independent operation, 32 bits transferred per channel

## Signal Lines per Channel:

- ▶ 32 data wires for actual data transfer
- ▶ 21 address wires for memory location
- ▶ 7 control wires for commands

## Parallel Processing:

- ▶ All 4 chips on channel receive same address/commands
- ▶ Data lines divided: each chip handles 8 bits
- ▶ Power managed by dedicated chips on module

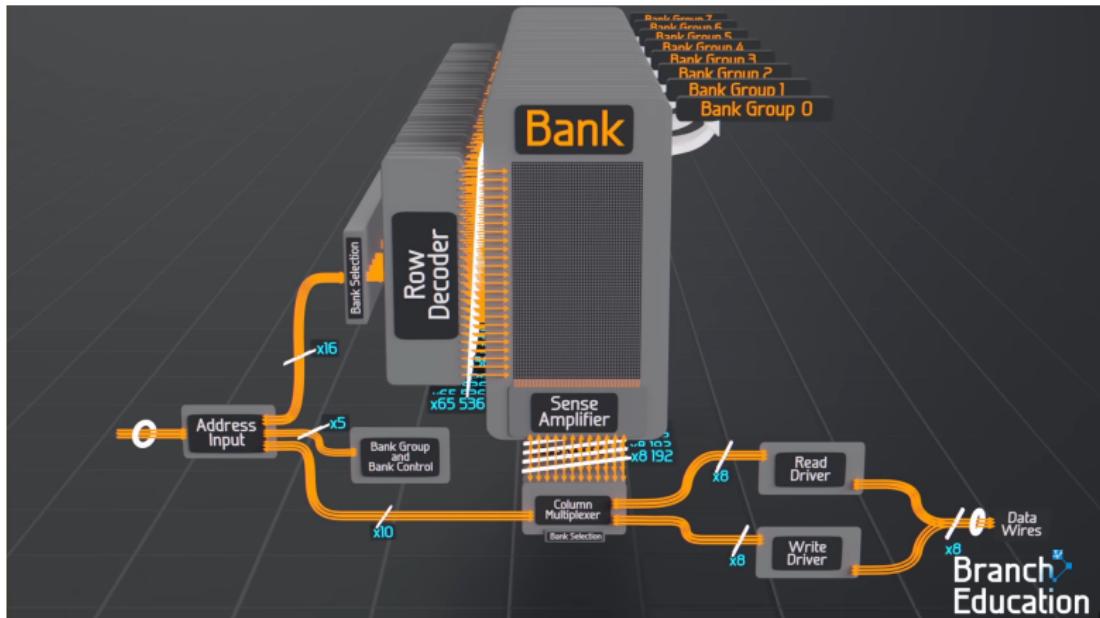
# Inside a DRAM Chip

## Physical Components:

- ▶ Packaging contains interconnection matrix
- ▶ Ball grid array connects to die (main chip)

## Die Organization (2 GB chip):

- ▶  $8 \text{ bank groups} \times 4 \text{ banks} = \mathbf{32 \text{ banks total}}$
- ▶ Each bank:  $65,536 \text{ rows} \times 8,192 \text{ columns}$
- ▶ Total:  $\sim 17 \text{ billion memory cells}$
- ▶ Complex network of wires and supporting circuits



# 31-Bit Memory Addressing

**Accessing 17 billion cells requires 31-bit address:**

- ▶ **3 bits:** Bank group selection (8 groups)
- ▶ **2 bits:** Bank selection (4 per group)
- ▶ **16 bits:** Row selection (65,536 rows)
- ▶ **10 bits:** Column group ( $8,192 \div 8$ )

**Transmission Optimization:**

- ▶ Address sent in two parts using 21 wires
- ▶ Part 1: Bank group + bank + row
- ▶ Part 2: Column address
- ▶ Each access reads/writes 8 bits simultaneously

# Memory Cell Structure: 1T1C

**Components (each stores 1 bit):**

## 1. Capacitor:

- ▶ Deep trench in silicon (few dozen nanometers)
- ▶ Two conductive surfaces with dielectric insulator
- ▶  $1V = \text{binary } 1, 0V = \text{binary } 0$

## 2. Access Transistor:

- ▶ Wordline activates transistor gate
- ▶ Bitline connects to transistor channel
- ▶ Controls capacitor access

# How Memory Cells Store Data

## Write Operation:

- ▶ Wordline ON → connects capacitor to bitline
- ▶ Charge flows to write 1 or discharge for 0

## Read Operation:

- ▶ Wordline ON → measure capacitor charge
- ▶ Charge amount indicates stored value

## Data Retention Challenge:

- ▶ Wordline OFF isolates capacitor
- ▶ Electron leakage through tiny transistor
- ▶ Requires periodic refresh

# Memory Array Organization

## Array Structure:

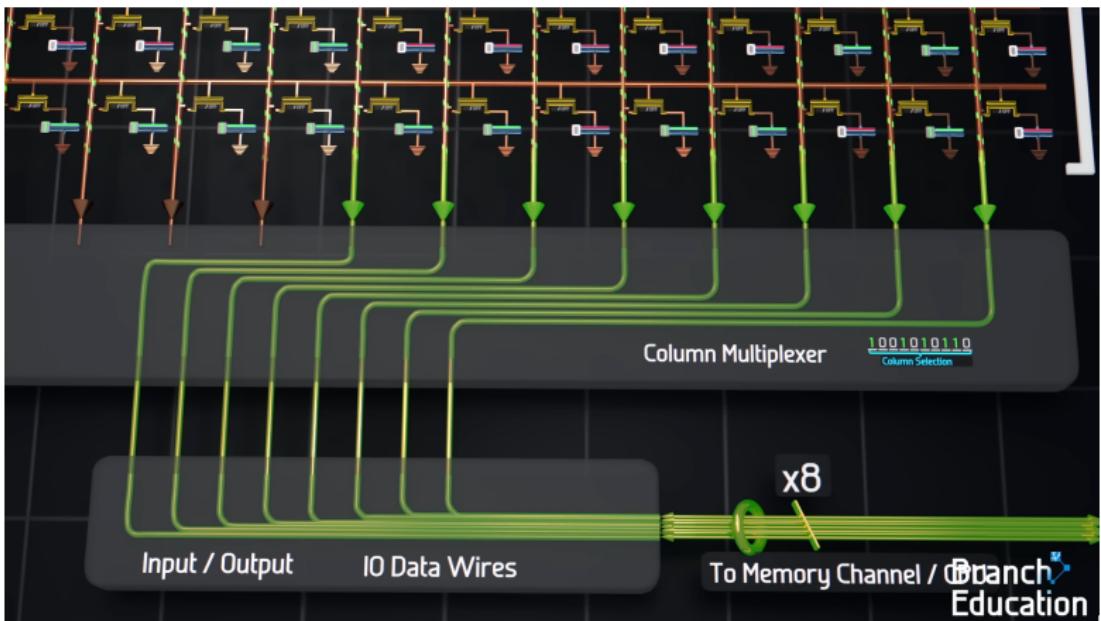
- ▶ **Wordlines:** Rows connecting to transistor gates
- ▶ **Bitlines:** Columns connecting to transistor channels
- ▶ Different vertical layers (no physical contact)

## Operation Rules:

- ▶ Active wordline connects entire row to bitlines
- ▶ **Only ONE wordline active at a time**
- ▶ Multiple active wordlines cause data interference

## Cell Access:

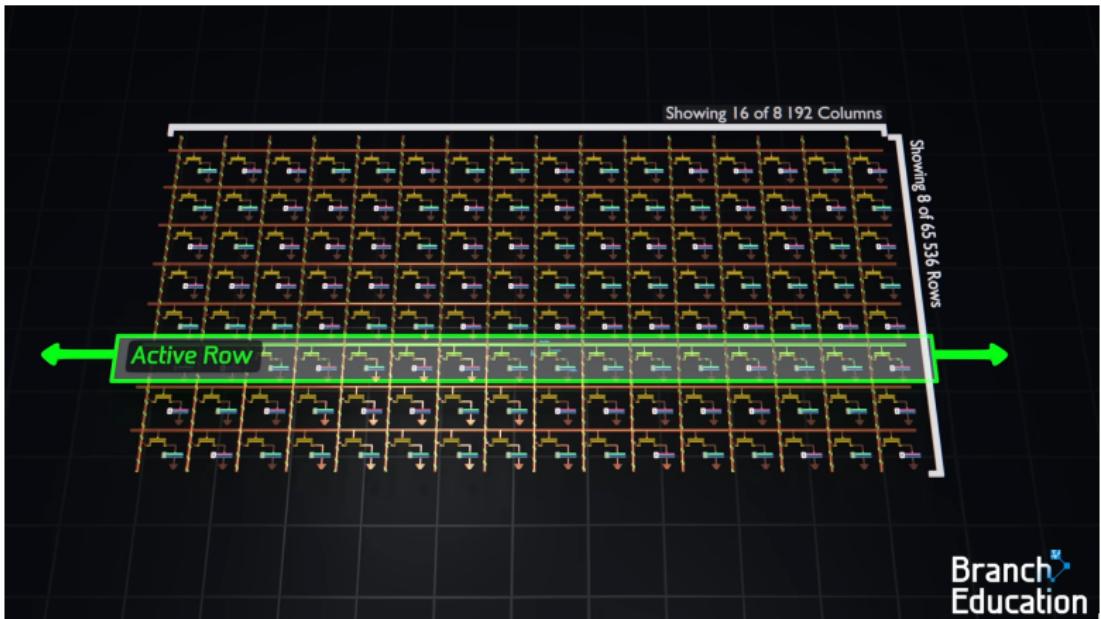
- ▶ Row decoder: 16 bits activate single wordline
- ▶ Column multiplexer: 10 bits select 8 bitlines



# Reading from Memory Cells

## Process:

- ▶ CPU sends read command + address
- ▶ Select bank, precharge bitlines to 0.5V
- ▶ Activate wordline (one row)
- ▶ Capacitors slightly change bitline voltage
- ▶ Sense amplifiers amplify to full 1V or 0V
- ▶ Column multiplexer selects 8 specific bits
- ▶ Data transmitted to CPU



# Writing to Memory Cells

## Process:

- ▶ CPU sends write command + address + data
- ▶ Select bank, precharge bitlines
- ▶ Activate row, sense amplifiers open row
- ▶ Column multiplexer selects 8 bitlines
- ▶ Write drivers override voltages (1V or 0V)
- ▶ Capacitors updated with new data

**Note:** All 4 chips operate in parallel



# Refresh Operation

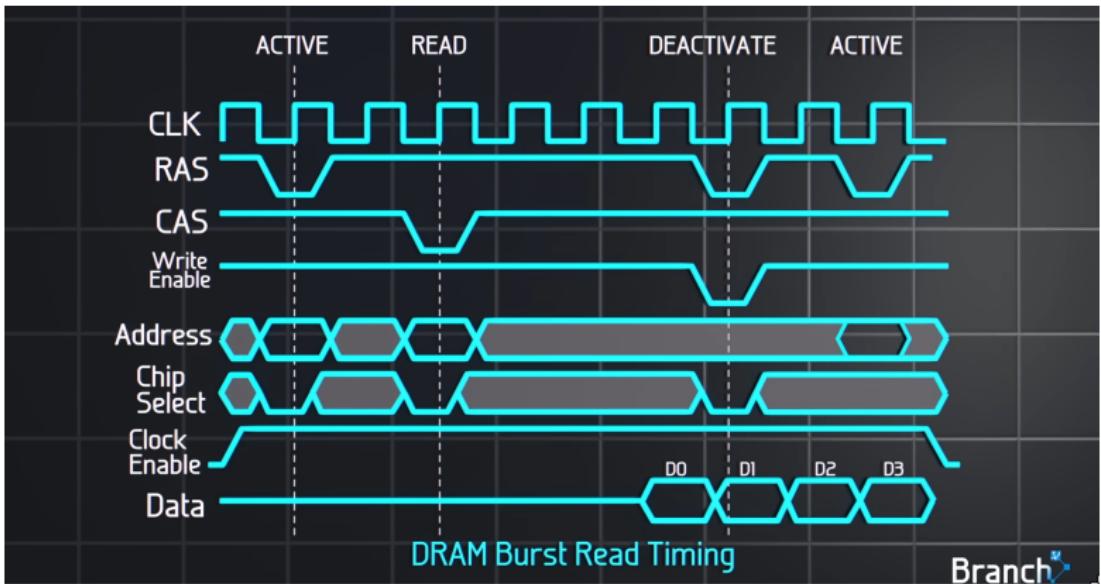
## Why Refresh:

- ▶ Electrons leak through nanoscale transistors
- ▶ Data loss prevention

## Process:

- ▶ Precharge bitlines to 0.5V
- ▶ Open one row at a time
- ▶ Sense amplifiers restore full voltage

**Timing:** 50ns per row, occurs every 64ms per bank



# Row Hits (Page Hits)

## What is a Row Hit:

- ▶ Next address is in already-open row
- ▶ Use only column address (10 bits)
- ▶ Skip row opening steps

## Benefits:

- ▶ Much faster access
- ▶ Can occur repeatedly on same row
- ▶ Systems optimized to maximize row hits

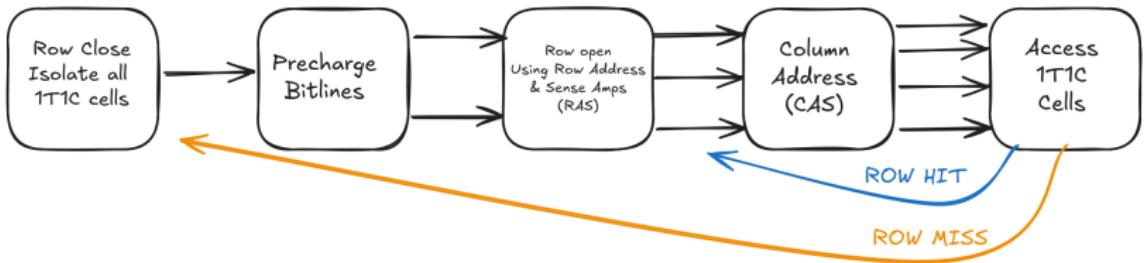
# Row Misses and Timing

## What is a Row Miss:

- ▶ Next address in different row
- ▶ Must close current row, open new row
- ▶ Significantly slower than row hit

## Timing Parameters (clock cycles):

- ▶ CAS latency: row hit to data
- ▶ RAS to CAS delay: row opening time
- ▶ Row precharge time
- ▶ Row active time



# Why Multiple Banks?

## Independent Operation:

- ▶ Each bank operates independently
- ▶ Multiple rows open across different banks
- ▶ Increases row hit probability

## Performance Impact:

- ▶ Reduced average access time
- ▶ Better parallel processing

# Bank Groups Organization

## Structure:

- ▶ 8 bank groups  $\times$  4 banks = 32 total banks
- ▶ Groups enable better refresh management

## Refresh Optimization:

- ▶ Refresh one bank per group
- ▶ Use other three banks during refresh
- ▶ Minimizes performance impact

# Burst Buffer Optimization

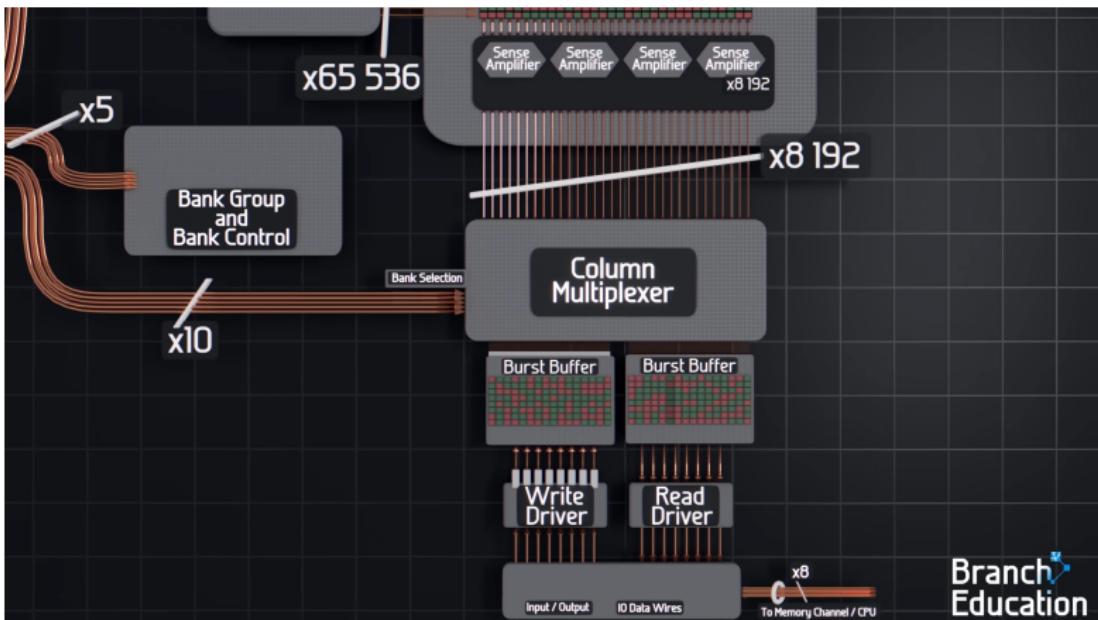
## Design:

- ▶ 128-bit temporary storage buffer
- ▶ Column address split: 6 bits (mux) + 4 bits (buffer)

## Operation:

- ▶ Load 128 cells into buffer using 6 bits
- ▶ Select 8 data locations using 4 bits, cycle through 16 combinations
- ▶ Result: 1,024 bits accessed quickly per chip (burst length = 16)

**Benefit:** Fast sequential access with random access capability



# Subarrays and Hierarchical Design

## Challenge:

- ▶ Large  $65,536 \times 8,192$  arrays
- ▶ Extremely long wordlines and bitlines

## Solution:

- ▶ Subdivide into  $1,024 \times 1,024$  subarrays
- ▶ Intermediate sense amplifiers per subarray
- ▶ Hierarchical row decoding

## Benefits:

- ▶ Shorter bitlines  $\rightarrow$  smaller capacitors
- ▶ Shorter wordlines  $\rightarrow$  reduced capacitive load
- ▶ Faster transistor activation
- ▶ Improved performance

# Folded Layout



Sense  
Amplifier

Branch  
Education

# Differential Pair Architecture

## Design:

- ▶ Two bitlines per sense amplifier
- ▶ Alternating rows connect to different bitlines
- ▶ Half bitlines active, half passive at any time

## Cross-Coupled Inverter:

- ▶ Active bitline = 1 → passive = 0
- ▶ Active bitline = 0 → passive = 1
- ▶ Creates differential pair (opposite values)
- ▶ Passive bitline not connected to cells

## Key Benefits:

1. Easy precharging: charge equalizes to 0.5V
2. Noise immunity from opposite charges
3. Reduced parasitic capacitance

