



# Digital Logic and System Design

## 8. Registers, Counters, and Memory

COL215, I Semester 2024-2025

Venue: LHC 408

'E' Slot: Tue, Wed, Fri 10:00-11:00

Instructor: Preeti Ranjan Panda

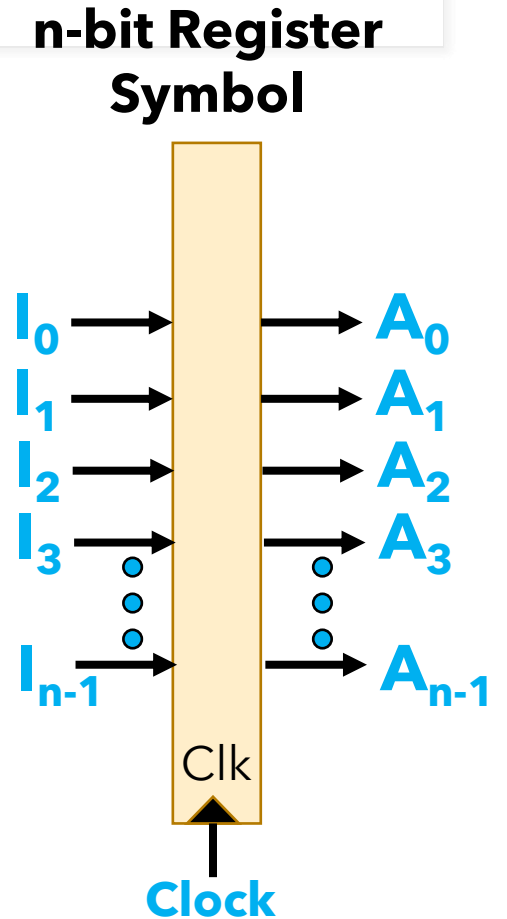
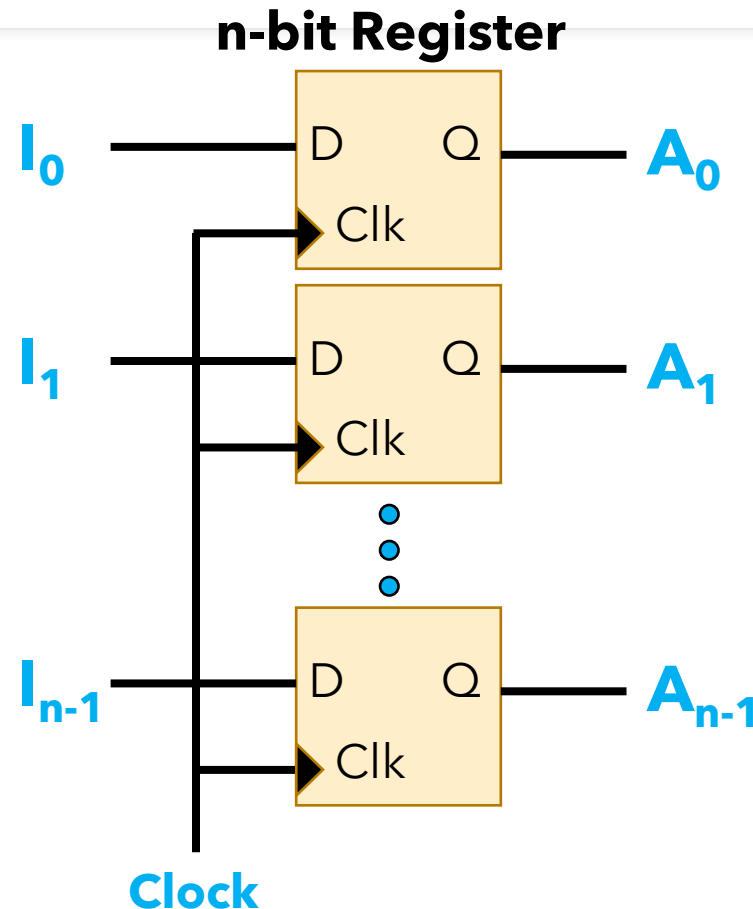
[panda@cse.iitd.ac.in](mailto:panda@cse.iitd.ac.in)

[www.cse.iitd.ac.in/~panda/](http://www.cse.iitd.ac.in/~panda/)

Dept. of Computer Science & Engg., IIT Delhi

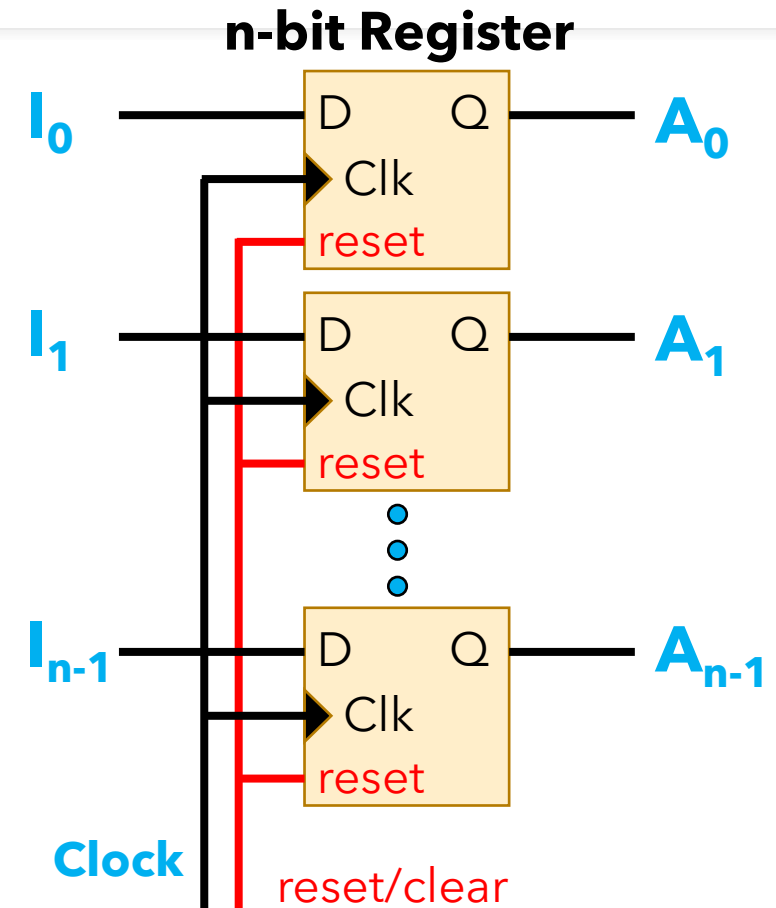
# Registers

- Group of Flip-flops
  - common **clock**
  - stores **1 bit** per flip-flop
- Recall: **State Register** of FSM
- n-bit value transferred from Ds to Qs on clock edge
  - storing **n-bit data**



# Registers with Reset

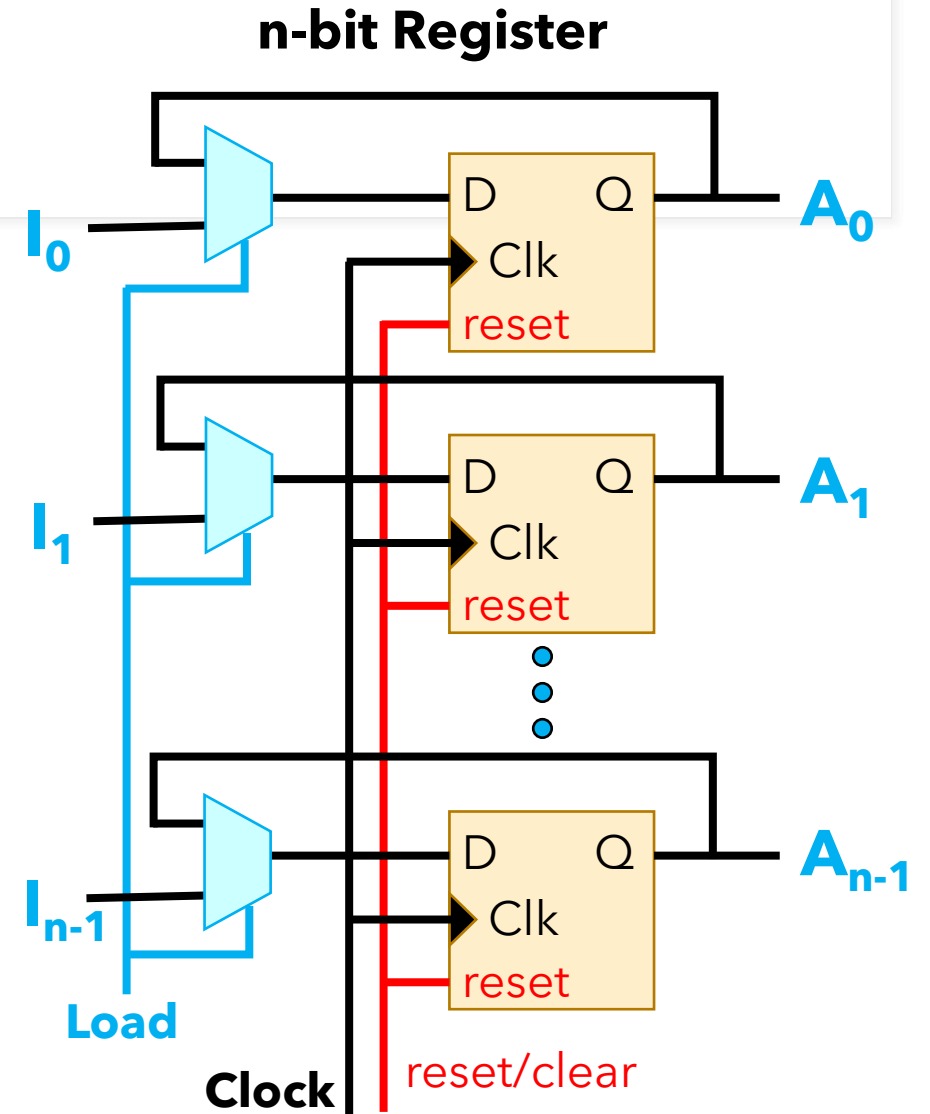
- **Reset/Clear** signal **asynchronously** clears **A** to **0**
  - independent of Clk
  - using DFF with asynchronous Reset





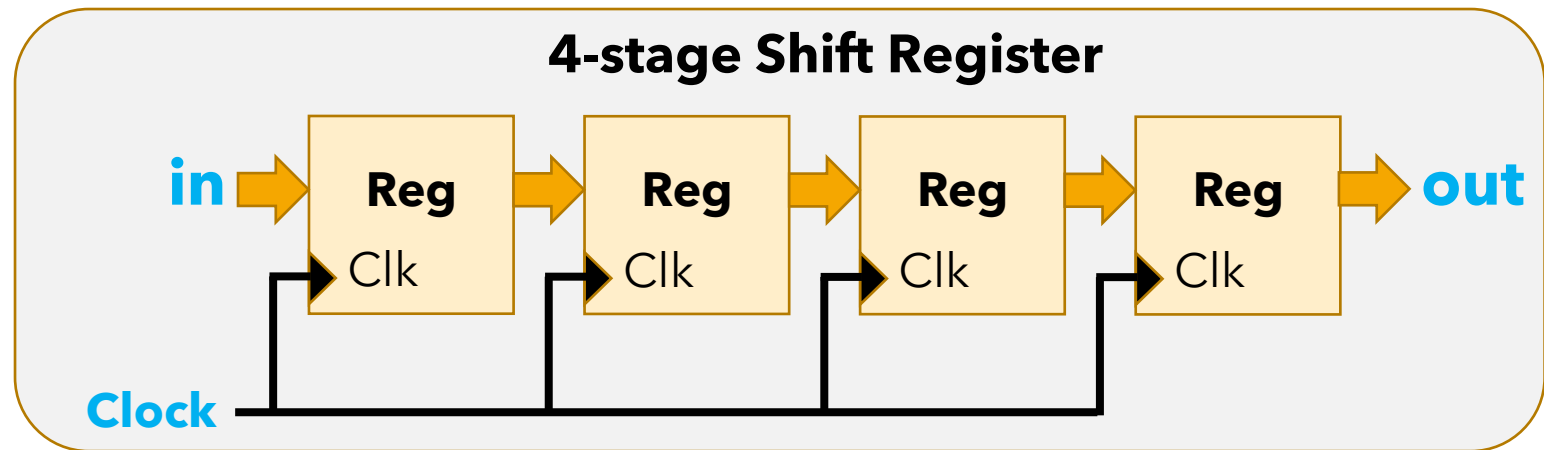
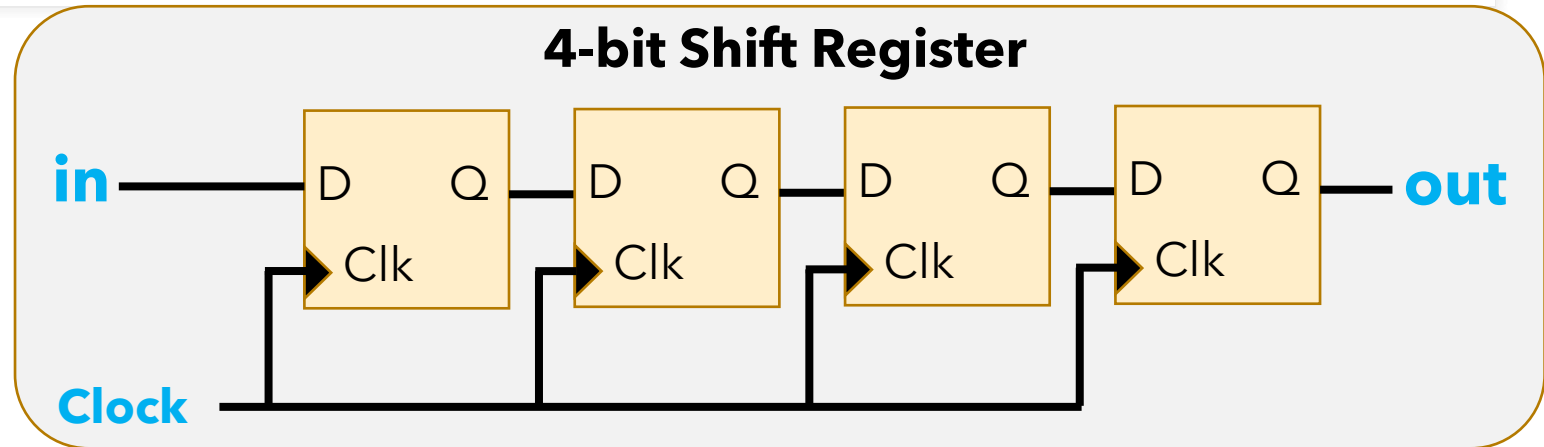
# Registers with **Load** signal

- Register function:
  - Transfers I to A on every clock
  - Called **Loading** new value to register (or **updating** the register)
- Desired function:
  - Load new value **only when required**
  - New control signal **Load**
- **Modify D input**
  - **Not clock** (don't disturb clock, causes uneven propagation delays)

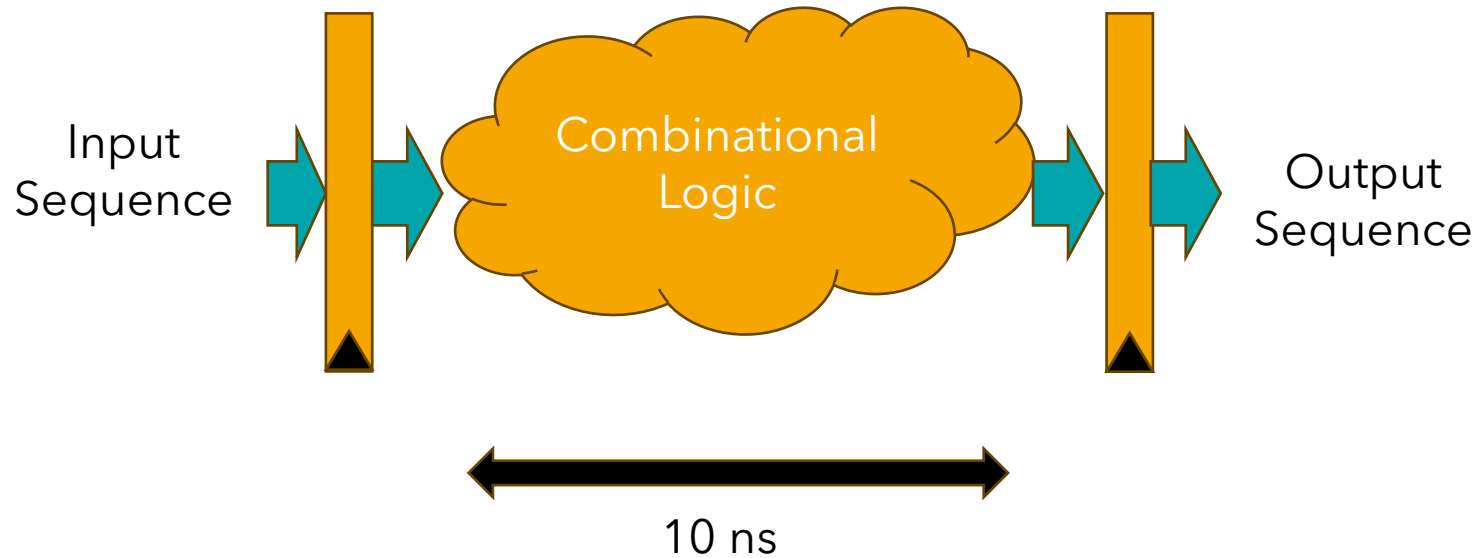


# Shift Registers

- **Cascade/Chain:** Q of one stage connected to D of next stage
- Common **Clock**
- **Shifts** bit to next DFF on clock edge
- General: Chained **data registers** (n-bits wide)

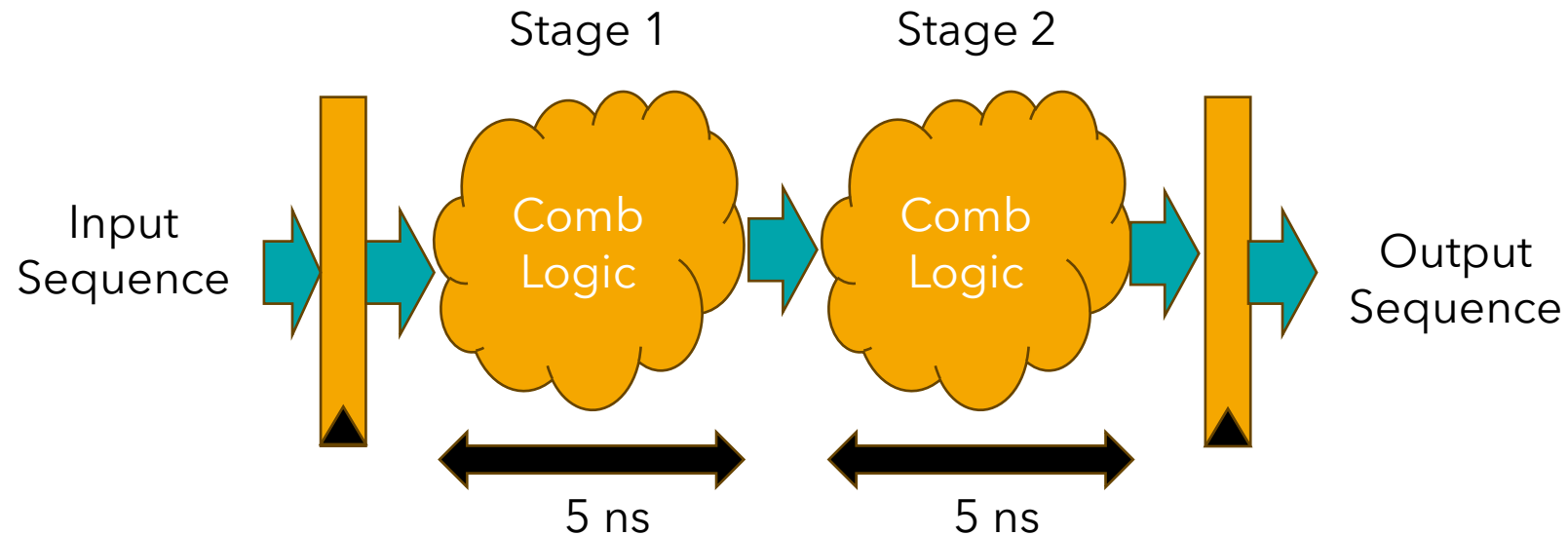


# Repeated Computation on Data Sequence



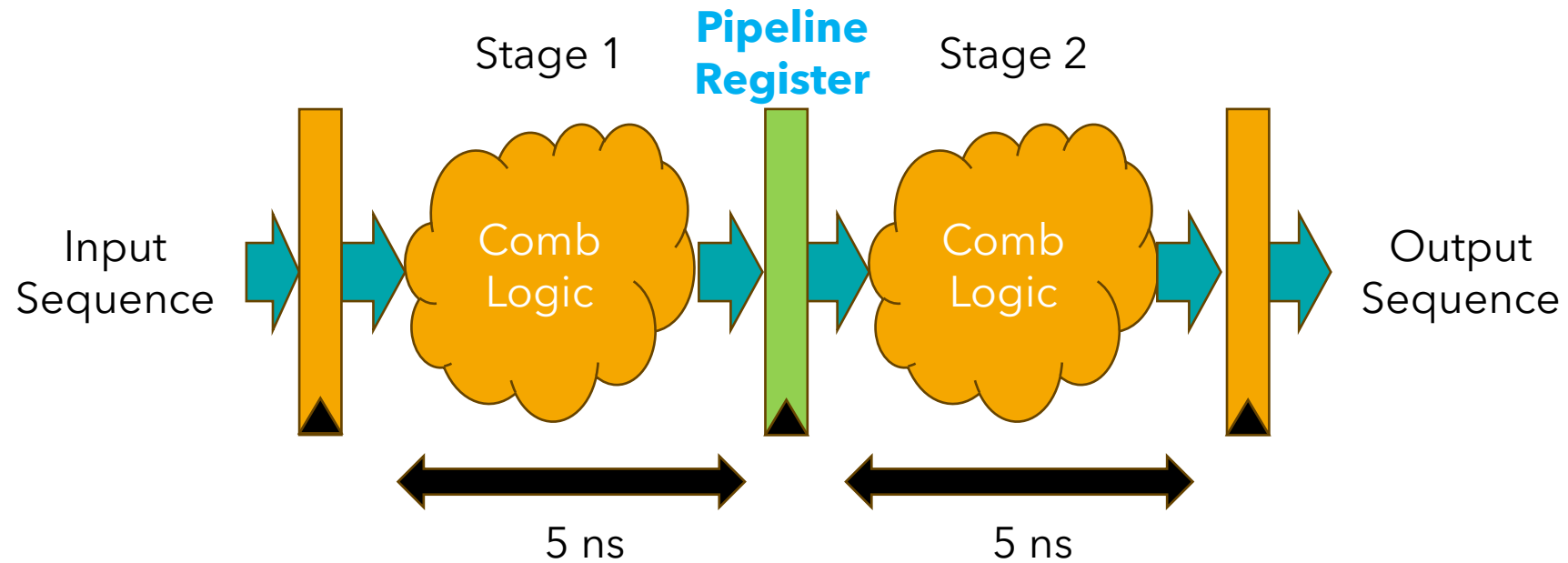
**Processing time for 1 input set: 10 ns**  
**Processing time for n input sets: ?**

# Repeated Computation: Split into Stages



**Does this help?**

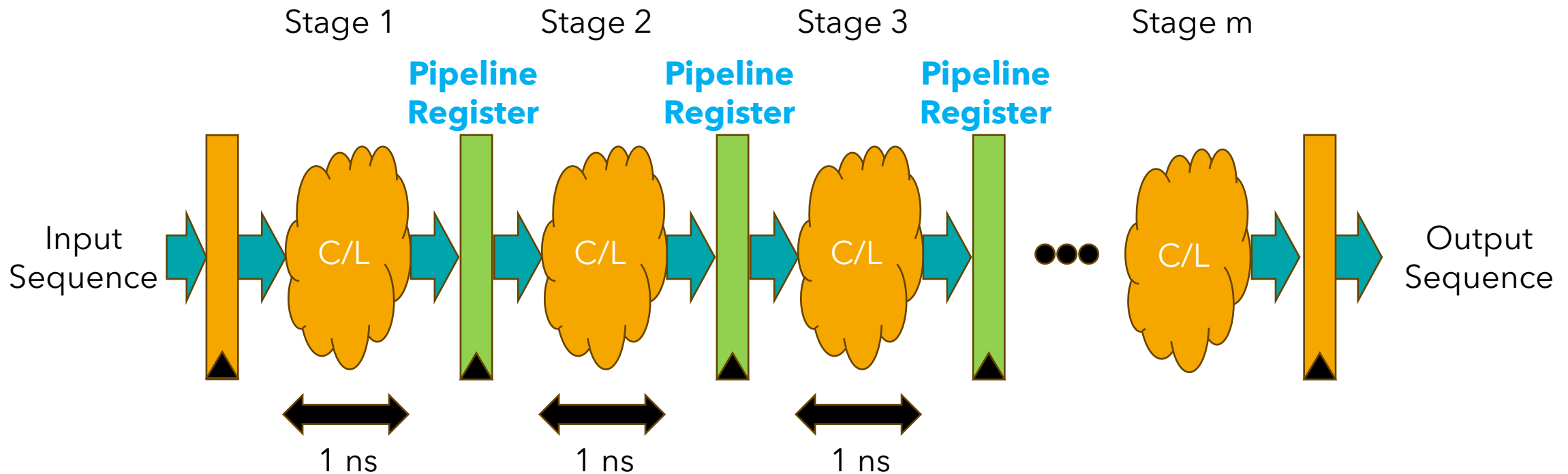
# Pipelining: Register Between Stages



**Processing time for n input sets: ?**



# Pipelining: Generalise to Multiple Stages



**Processing time for n input sets: ?**

# Counters

- Register going through a given **sequence of states on input pulse**
- Already studied: **mod 3 counter**
  - Sequence: 0,1,2,0,1,2,0,1,2
- Counting on common **clock pulse**:  
**synchronous counter** (e.g., mod 3 counter)
- Pulse could be internal signal: **ripple counter**
- Counter value on **Q of DFFs**

B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>	Sequence ↓
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

# Counters

- How do we generate sequence for DFF **B<sub>0</sub>**?
  - Alternating Sequence
- How do we generate sequence for DFF **B<sub>1</sub>**?
  - Alternating Sequence
  - Triggered by?
- How do we generate sequence for DFF **B<sub>2</sub>**?
  - Alternating Sequence
  - Triggered by?

**DFF: B<sub>2</sub> B<sub>1</sub> B<sub>0</sub>**

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Sequence

# Counters

- How do we generate sequence for DFF **B<sub>0</sub>**?
  - Alternating Sequence
  - Triggered by **External Count Signal**
- How do we generate sequence for DFF **B<sub>1</sub>**?
  - Alternating Sequence
  - Triggered by **Negative edge of B<sub>0</sub>**
- How do we generate sequence for DFF **B<sub>2</sub>**?
  - Alternating Sequence
  - Triggered by **Negative edge of B<sub>1</sub>**

**DFF: B<sub>2</sub> B<sub>1</sub> B<sub>0</sub>**

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

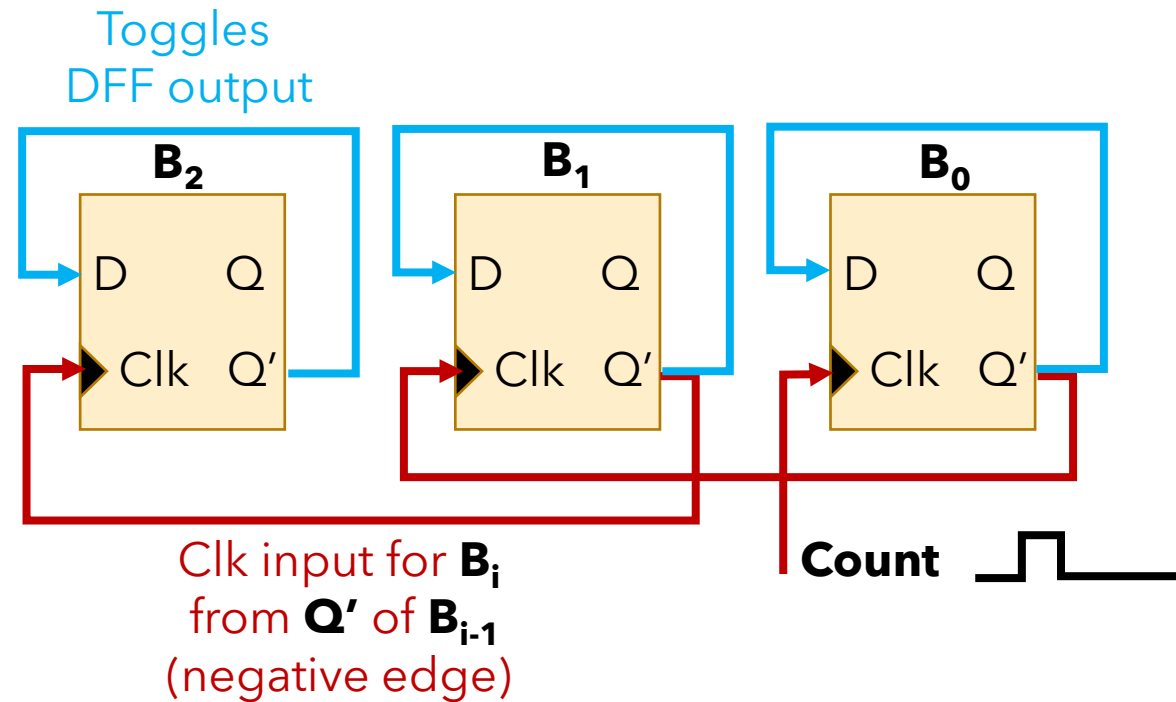
Sequence

# Ripple Counters

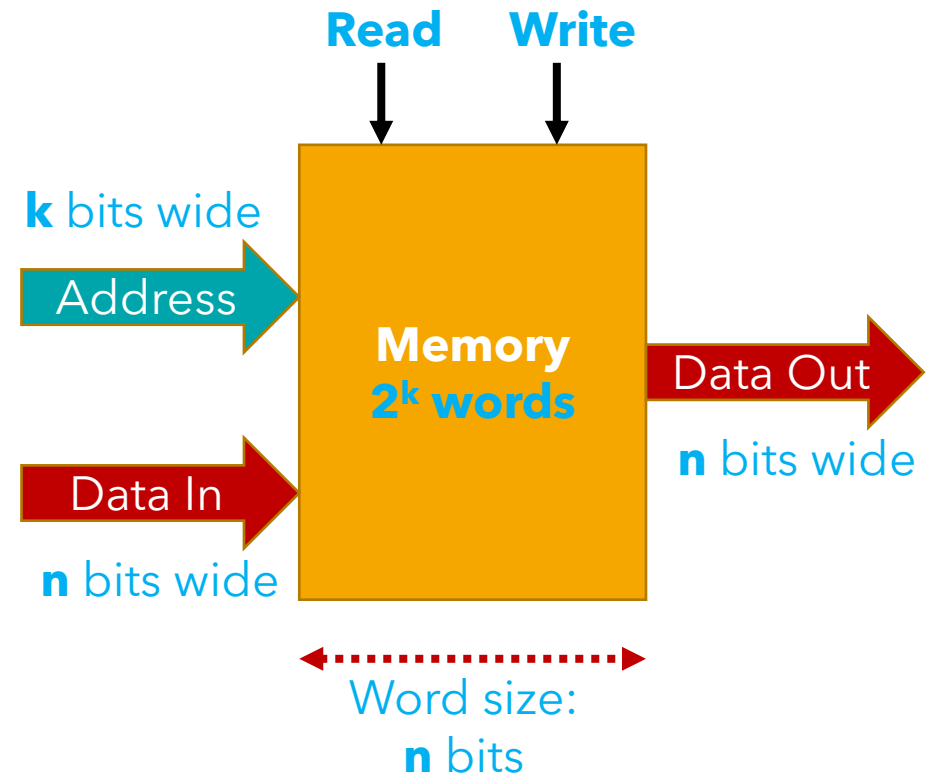
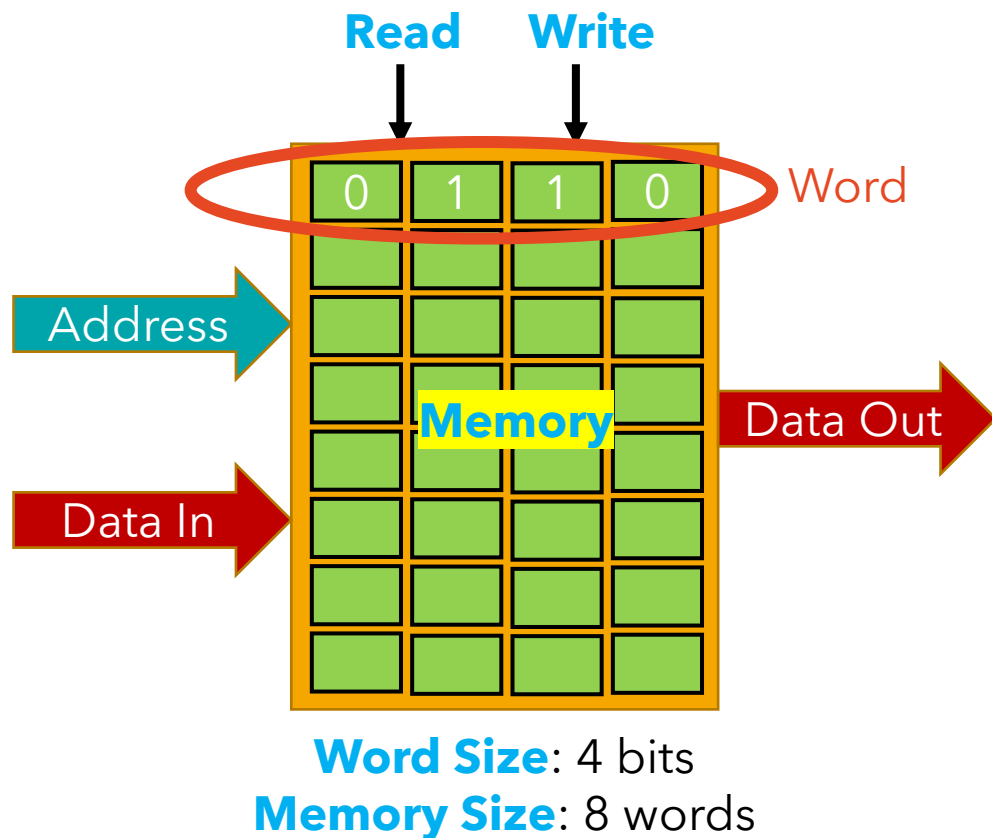
**DFF:**

$B_2$	$B_1$	$B_0$
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Sequence

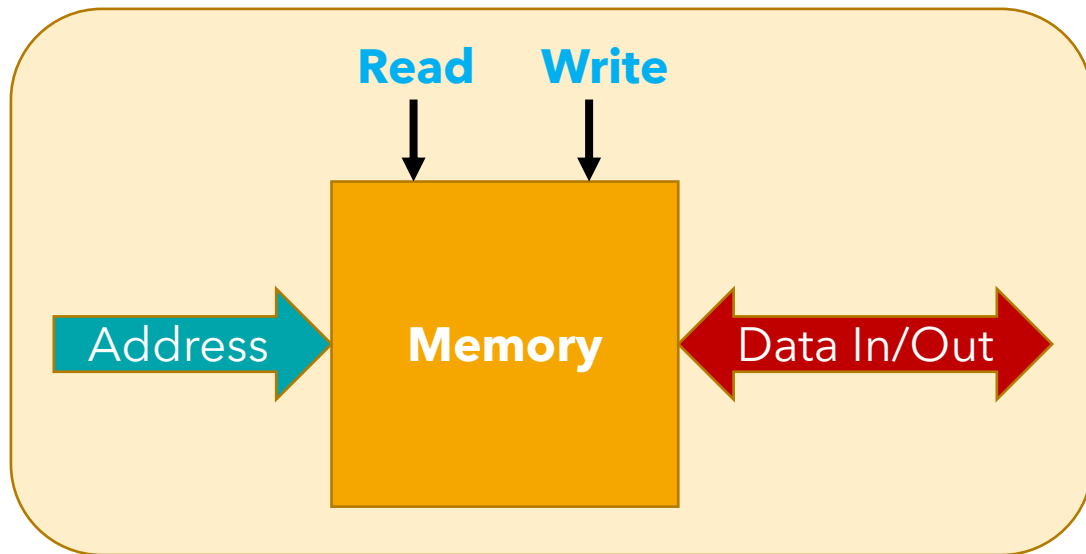


# Recall: Memory Interface and Function

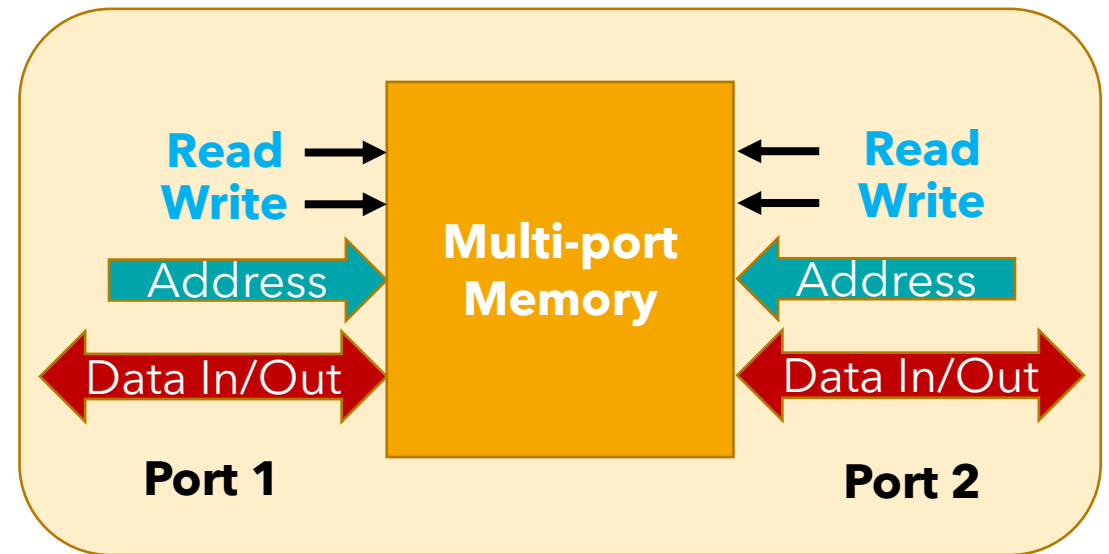




# Memory: General Architectures



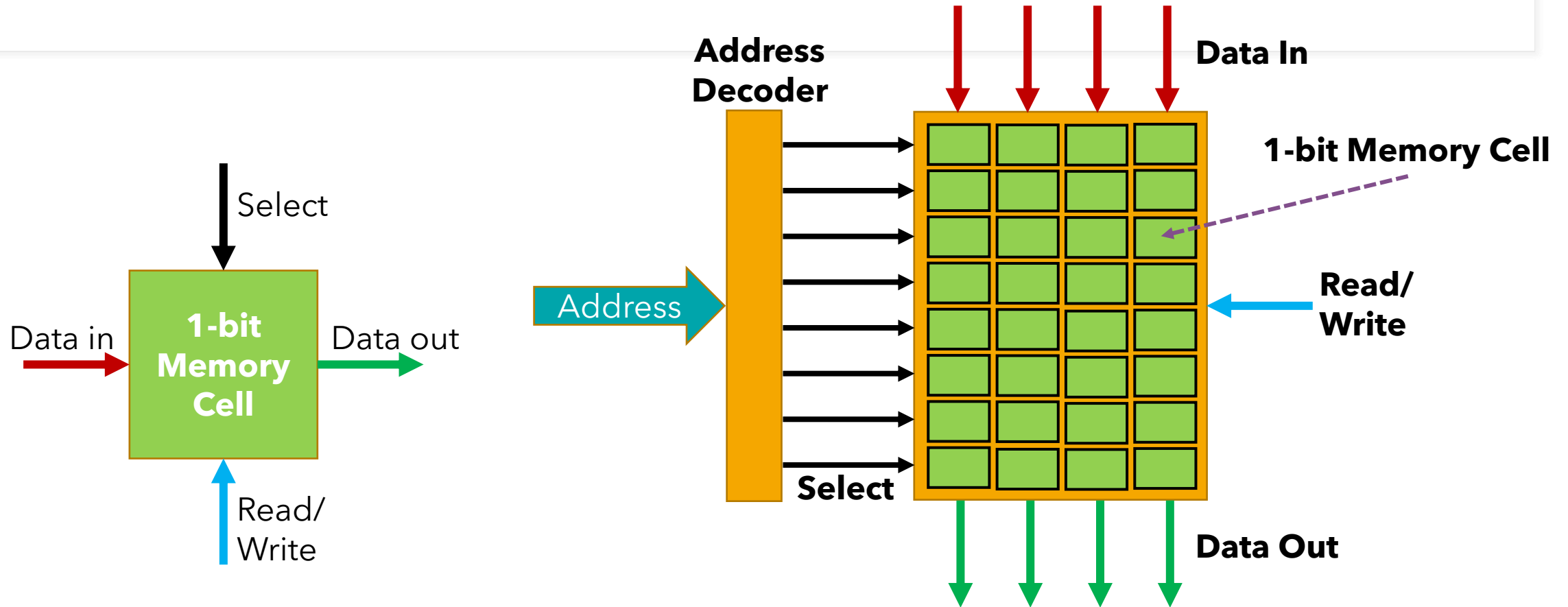
**Bi-directional Data Bus**



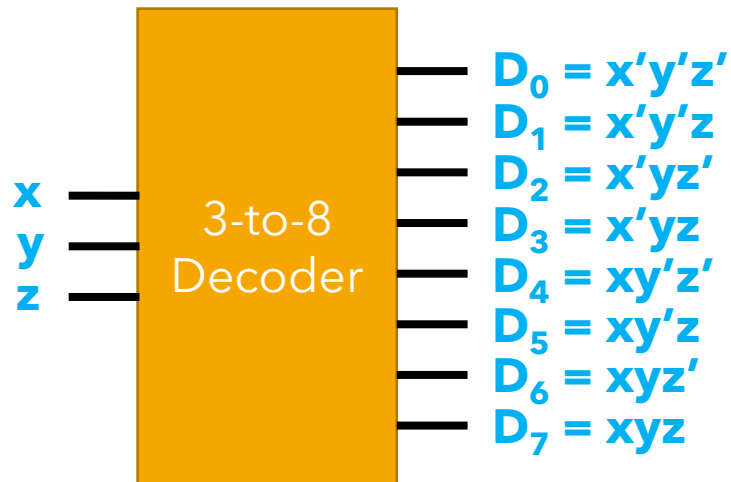
**Multiple Simultaneous Accesses**

Read & Write on Same Address: ??

# Memory Cell



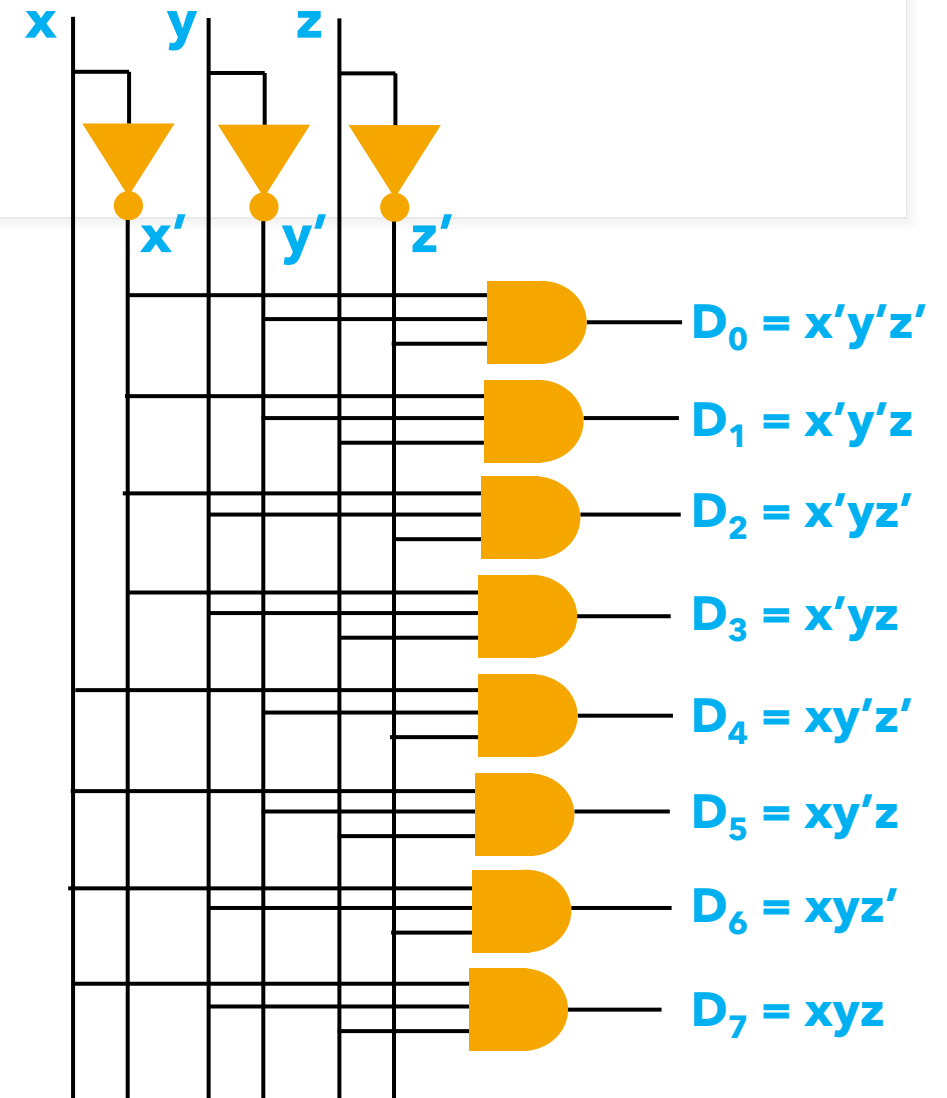
# Recall: Decoder Implementation



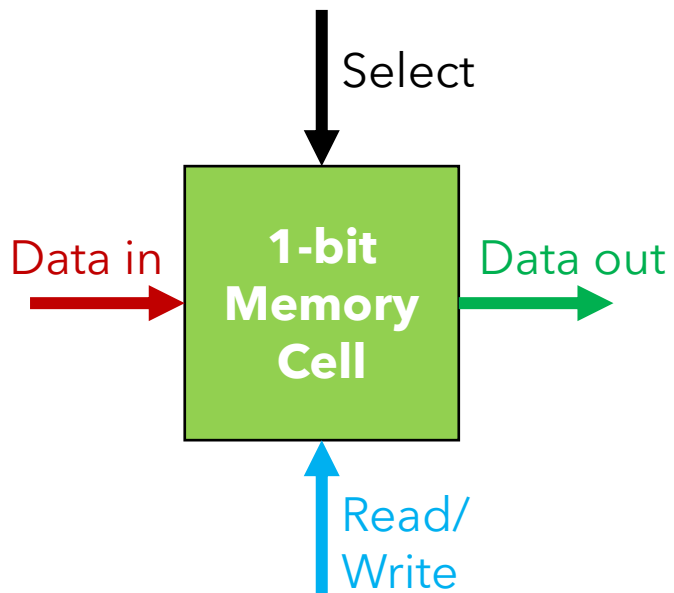
Each output is a **minterm**

Truth Table?

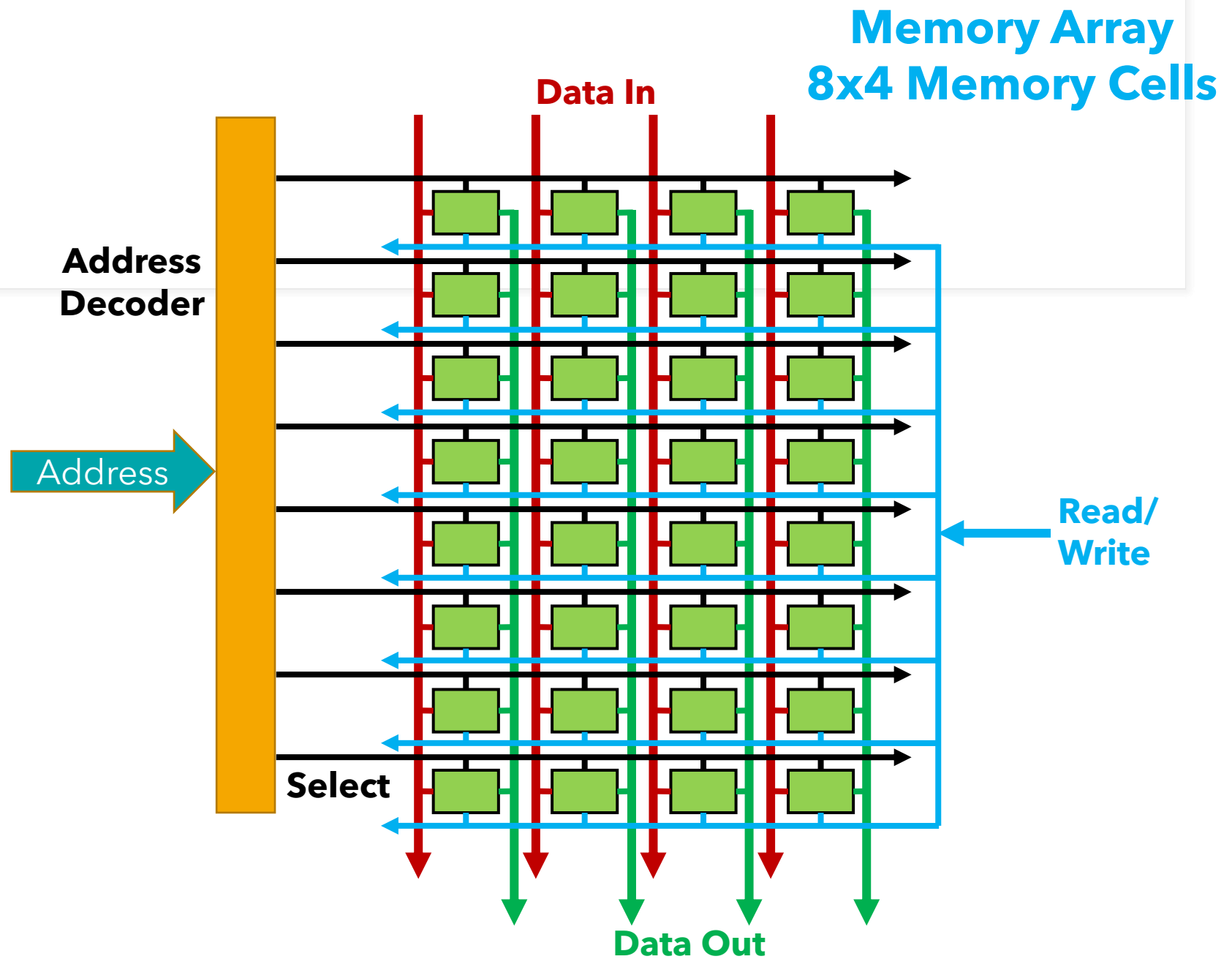
a b c	$D_0$	$D_1$	$D_2$	$D_3$	$D_4$	$D_5$	$D_6$	$D_7$
0 0 0	1	0	0	0	0	0	0	0
0 0 1	0	1	0	0	0	0	0	0
0 1 0	0	0	1	0	0	0	0	0
0 1 1	0	0	0	1	0	0	0	0
1 0 0	0	0	0	0	1	0	0	0
1 0 1	0	0	0	0	0	1	0	0
1 1 0	0	0	0	0	0	0	1	0
1 1 1	0	0	0	0	0	0	0	1



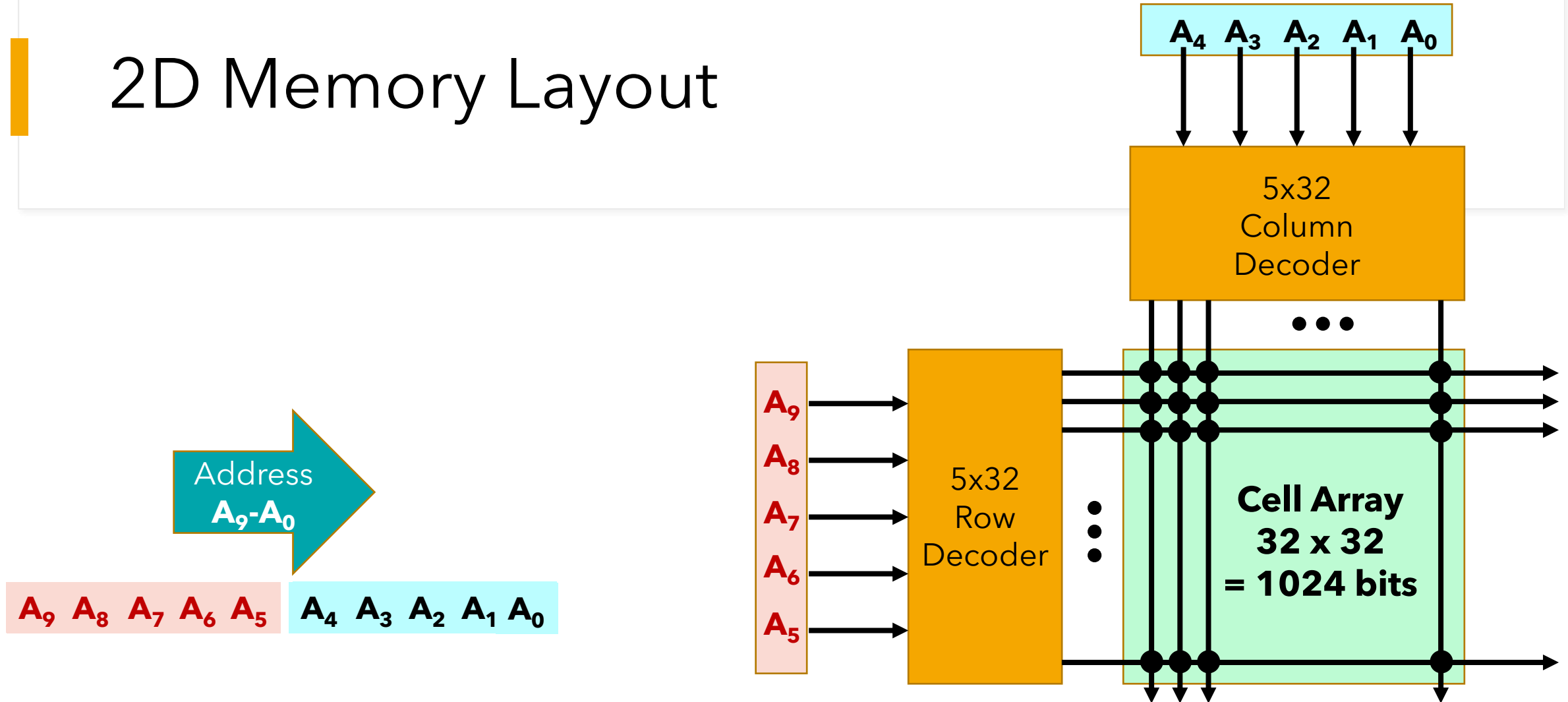
# Memory Cell and Array



Memory Cell

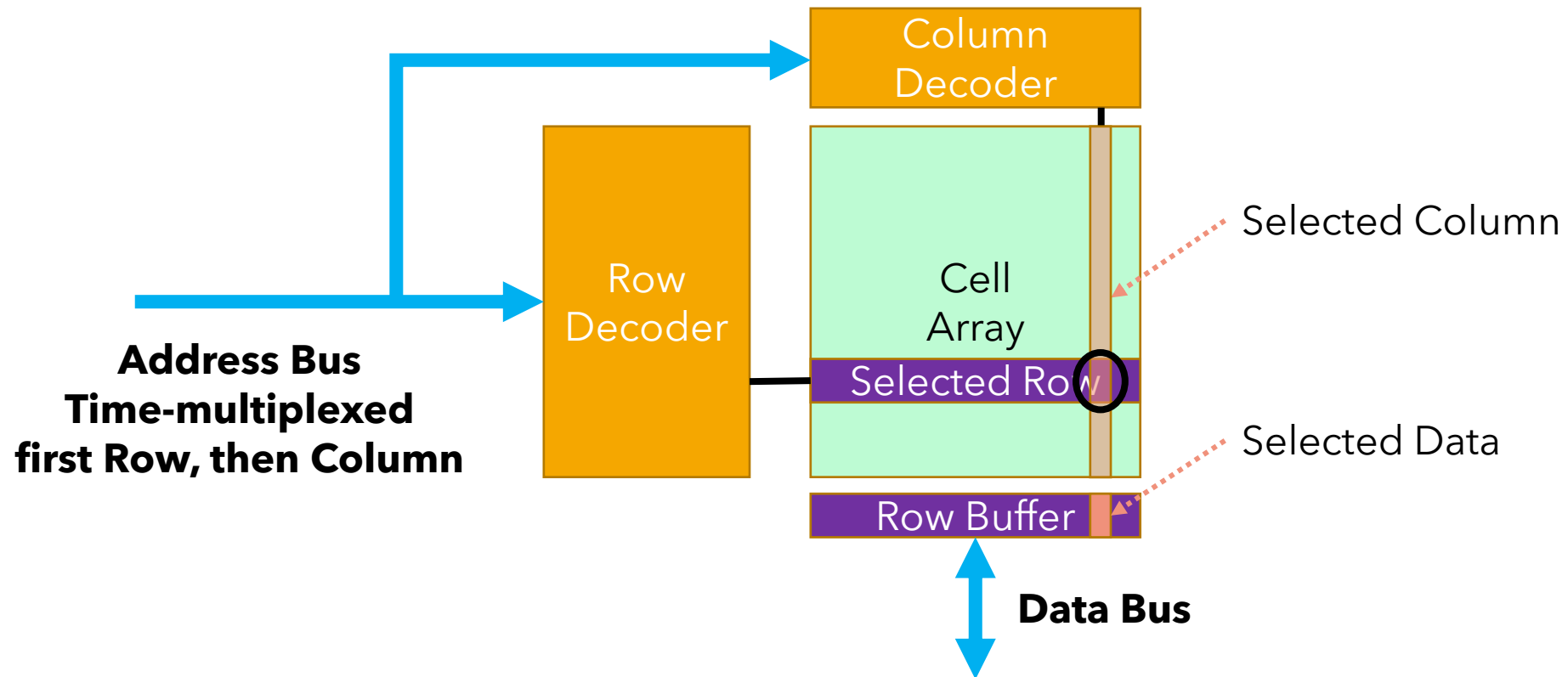


# 2D Memory Layout



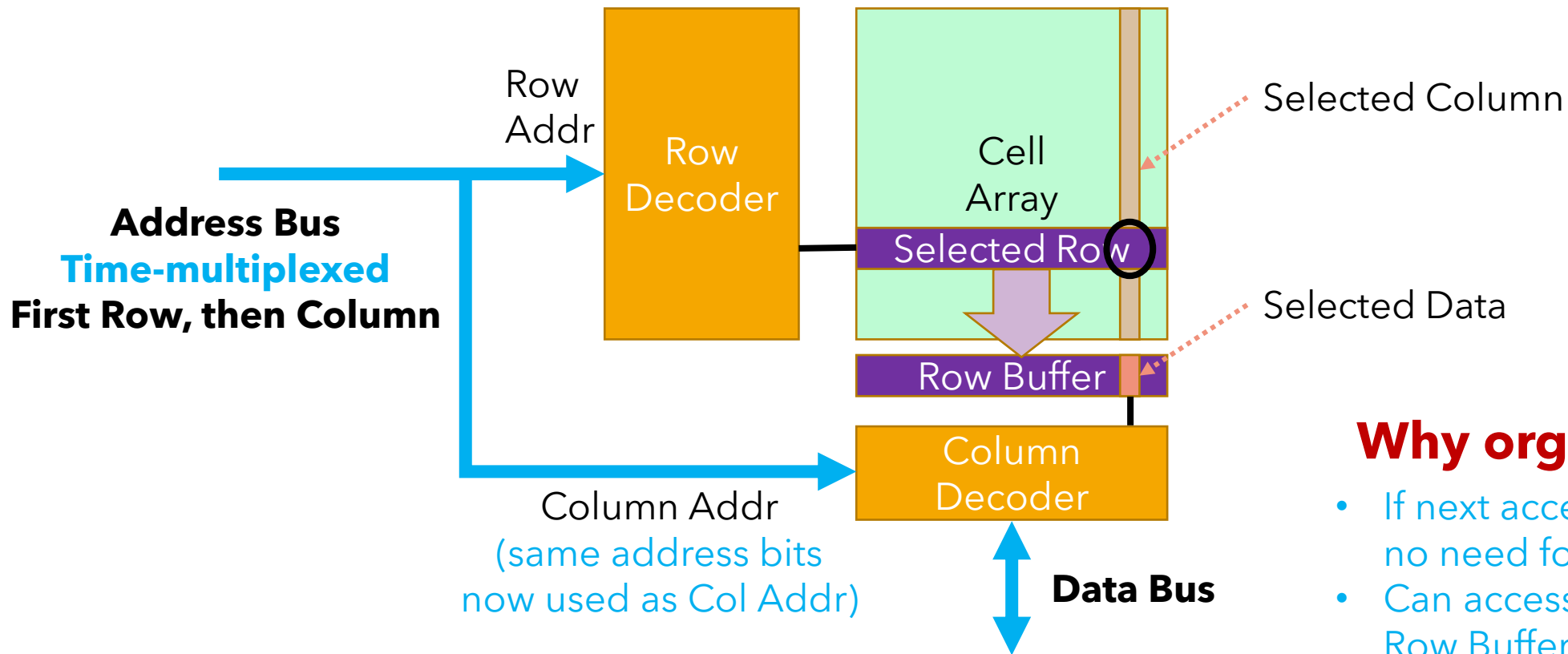
**Data Arranged in Square/Rectangle**

# Dynamic Random Access Memory (DRAM) Addressing





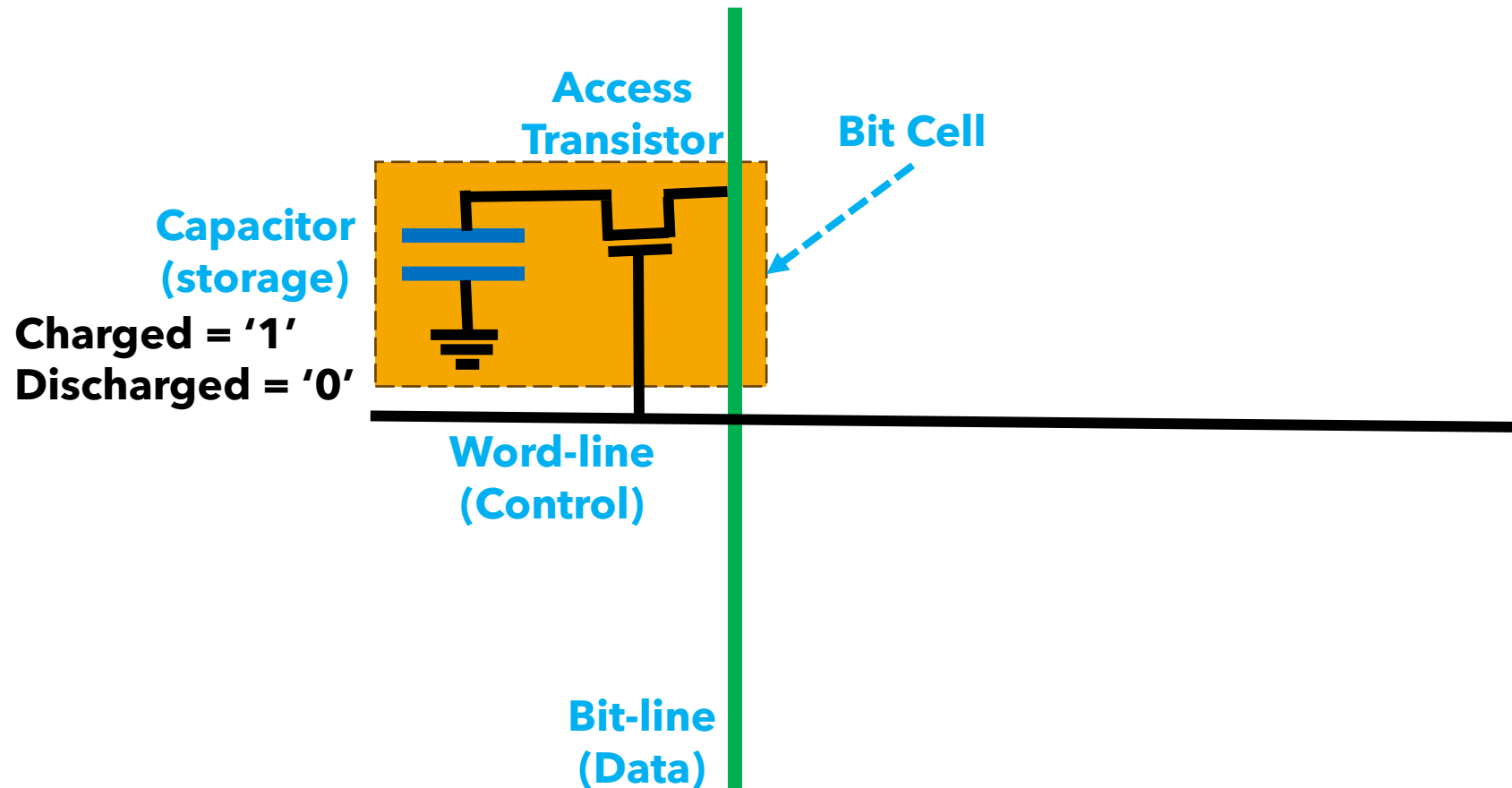
# DRAM Addressing



## Why organize this way?

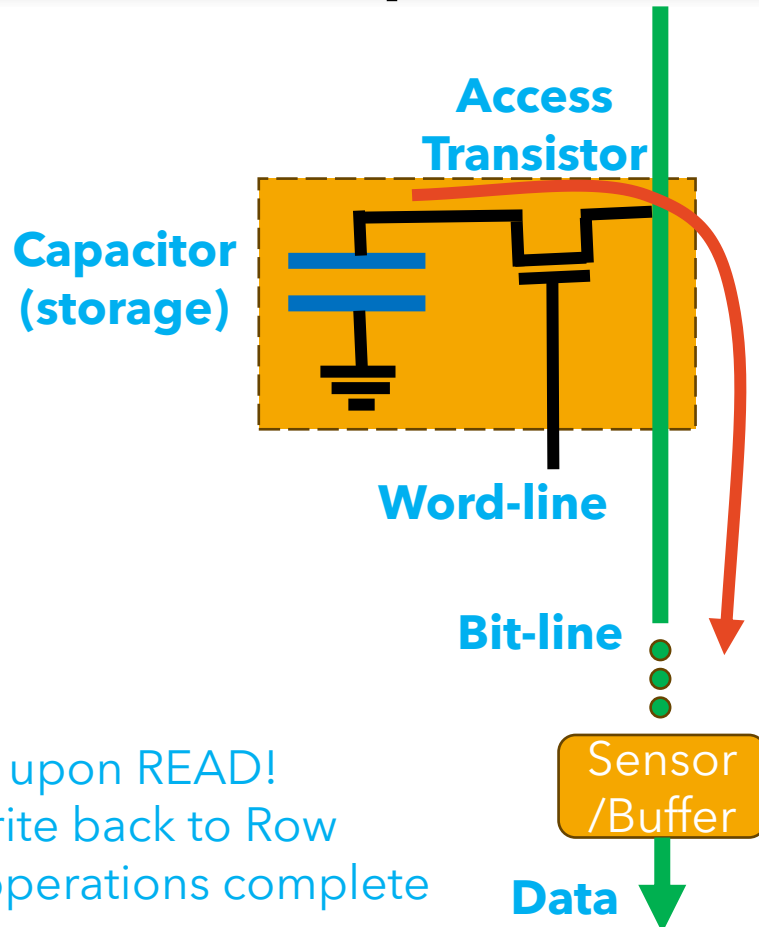
- If next access is to same row, no need for Row Decoding
- Can access data directly from Row Buffer. Only Col Decoding

# DRAM Storage



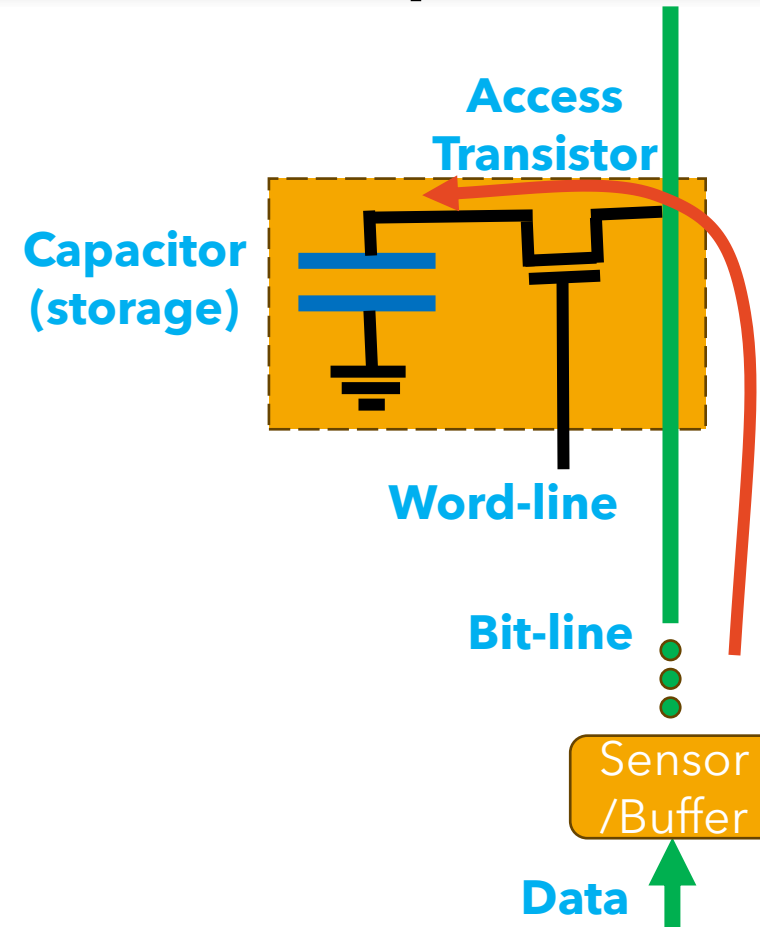
# DRAM Operations

## READ Operation



- Data **LOST** upon READ!
- Need to write back to Row once row operations complete

## WRITE Operation



# DRAM **Refresh** Operation

- Data leaks away from capacitor storage
- DRAM Data Needs to be **Refreshed** frequently
  - **Refresh** = **READ**, then **WRITE**
- Cell/Row unavailable for READ/WRITE during Refresh operation
- *Compare*: Flip-flop data is NOT lost  
STATIC memory: No Refresh required.