ECE 2050 Digital Logic and Systems

Chapter 11: Memory & Logic Arrays

Instructor: Tinghuan CHEN, Ph.D.

香港中文大學（深圳）
The Chinese University of Hong Kong, Shenzhen

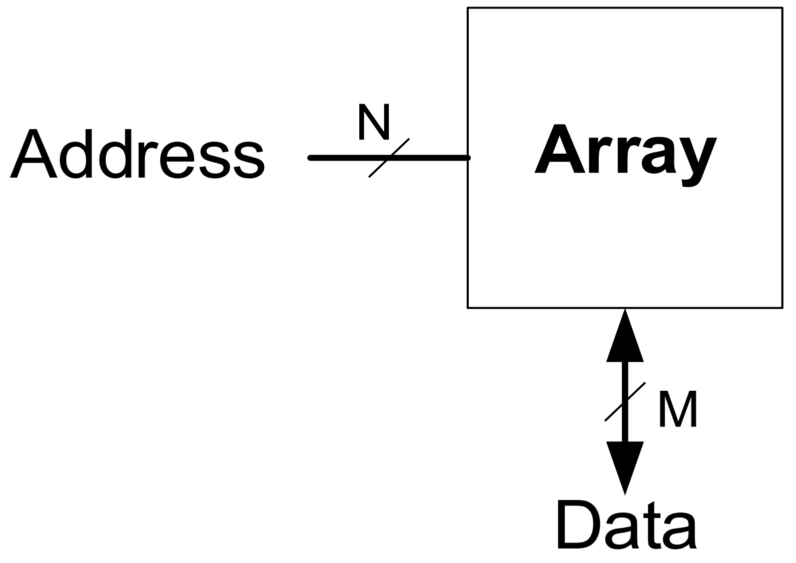
Chapter Review

- Counters
  - Asynchronous
  - Synchronous
- Up/Down Counters
- Design of Synchronous Counters
- Cascaded Counters
- Counter Decoding

Memory

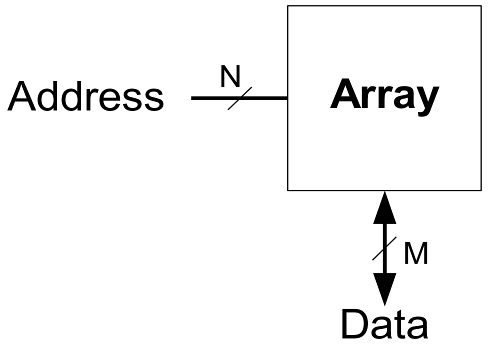
Memory Arrays

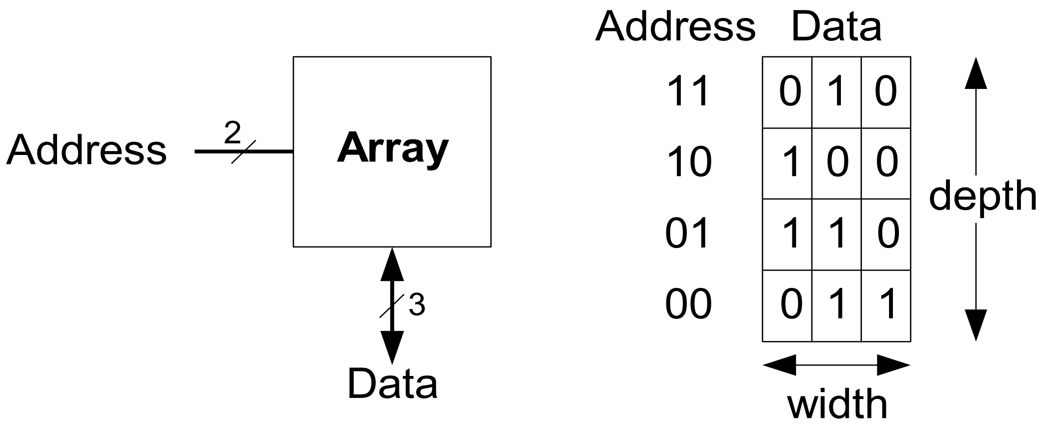
- Efficiently store large amounts of data
- *M*-bit data value read/written at each unique *N*-bit address
- 3 common types:
  - Dynamic random access memory (DRAM)
  - Static random access memory (SRAM)
  - Read only memory (ROM)



Memory Arrays

- 2-dimensional array of bit cells
- Each bit cell stores one bit
- *N* address bits and *M* data bits:
  - 2^N rows and *M* columns
  - **Depth:** number of rows (number of words)
  - **Width:** number of columns (size of word)
  - **Array size:** depth × width = $2^N \times M$

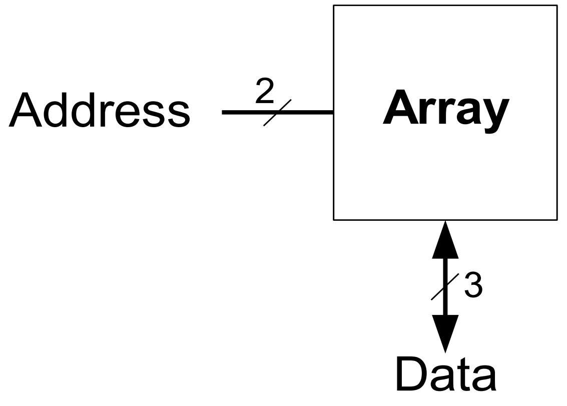


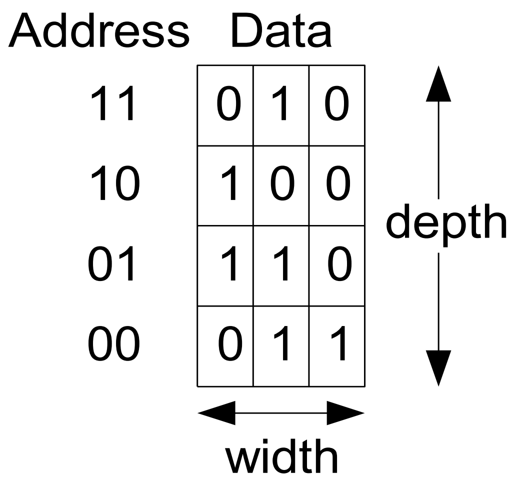


Address	Data
11	0 1 0
10	1 0 0
01	1 1 0
00	0 1 1

Memory Array Example

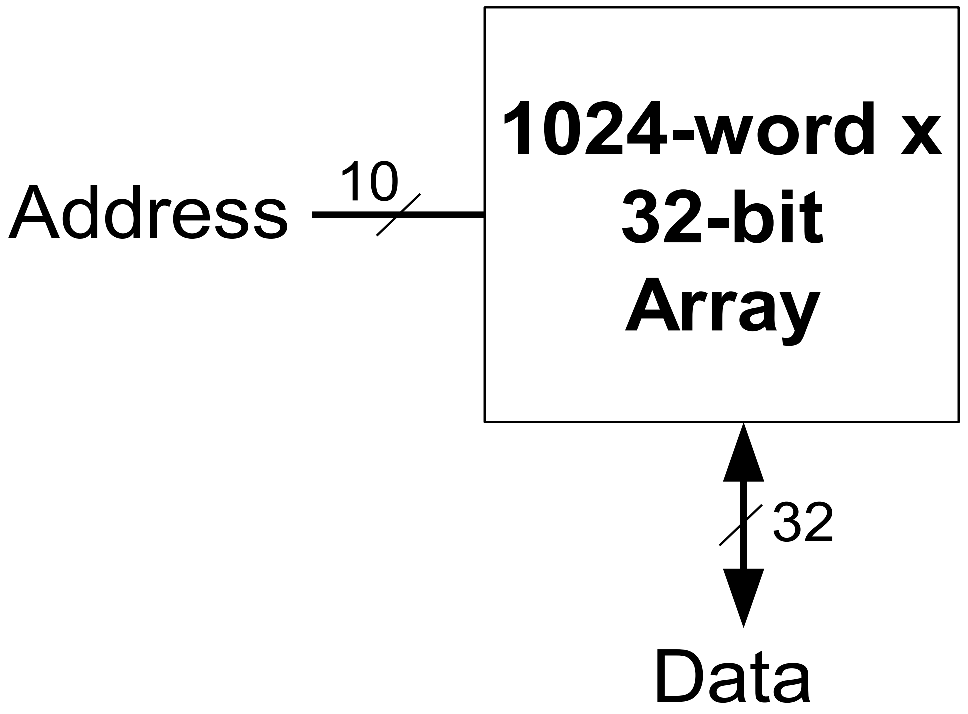
- **$2^2 \times$ 3-bit** array
- **Number of words:** 4
- **Word size:** 3-bits
- For example, the 3-bit word stored at address 10 is 100



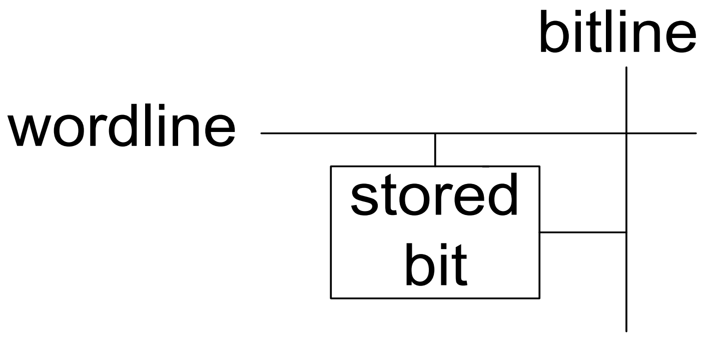


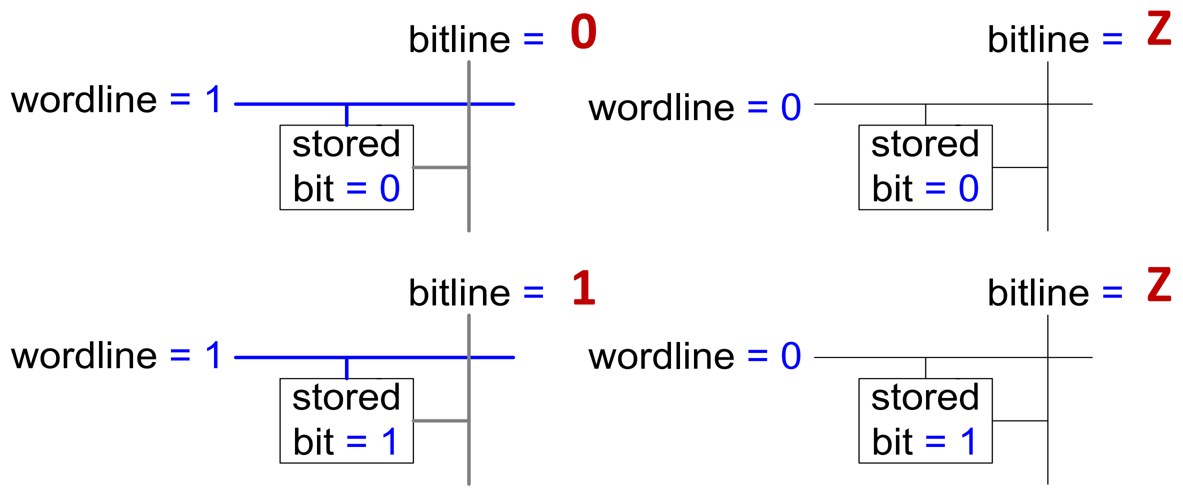
Address	Data		
11	0	1	0
10	1	0	0
01	1	1	0
00	0	1	1

Memory Arrays



Memory Array Bit Cells

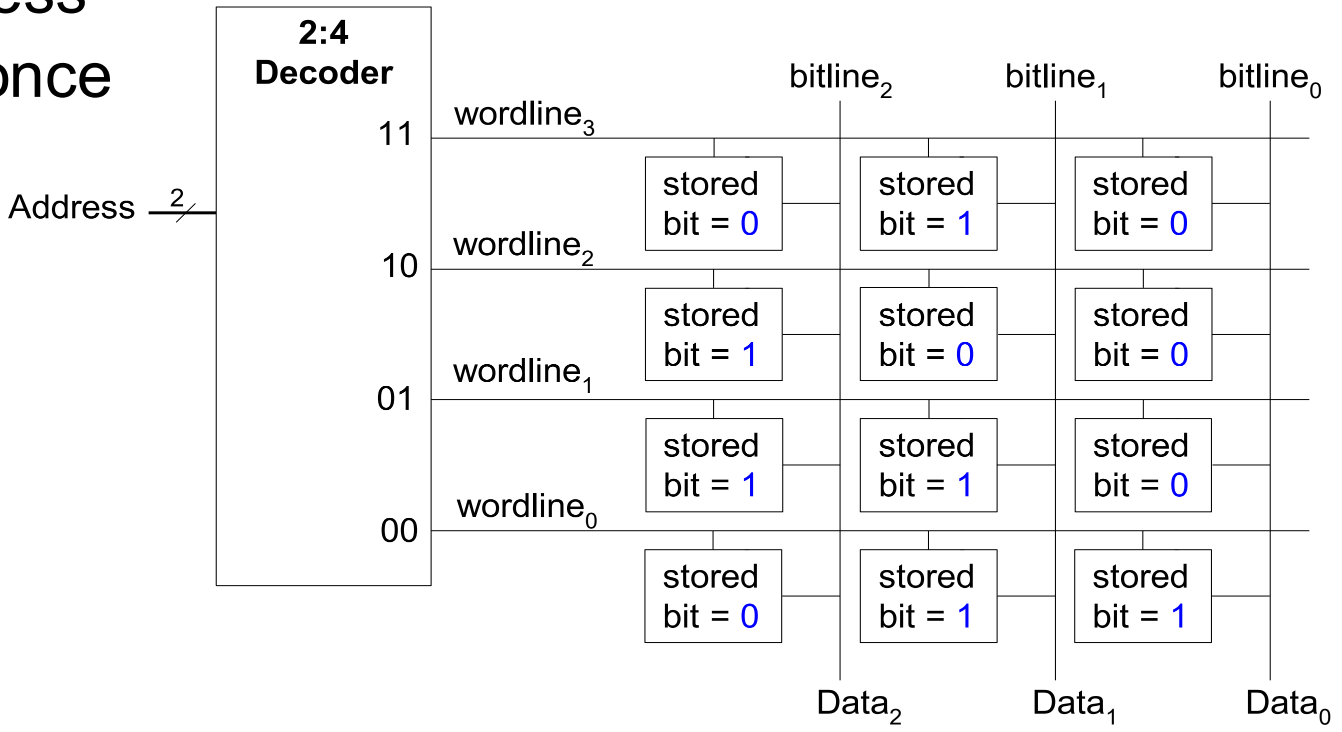




(a) (b)

Memory Array

- **Wordline:**
  - like an enable
  - single row in memory array read/written
  - corresponds to unique address
  - only one wordline **HIGH** at once



Types of Memory

- Random access memory (RAM): **volatile**
- Read only memory (ROM): **nonvolatile**

RAM: Random Access Memory

- **Volatile:** loses its data when power off
- Read and written quickly
- Main memory in your computer is RAM (DRAM)

- Historically called ***random access*** memory because any data word accessed as easily as any other (in contrast to sequential access memories such as a tape recorder)

ROM: Read Only Memory

- **Nonvolatile:** retains data when power off
- Read quickly, but writing is impossible or slow
- Flash memory in cameras, thumb drives, and digital cameras are all ROMs

- Historically called ***read only*** memory because ROMs were written at time of fabrication or by burning fuses. Once a ROM was configured, it could not be written again. This is no longer the case for Flash memory and other types of ROMs.

RAM

Types of RAM

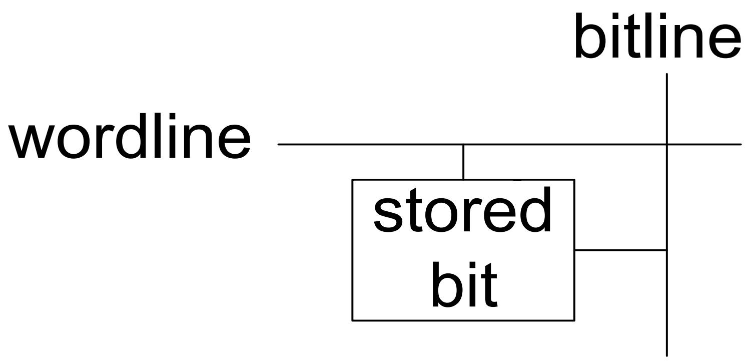
- **DRAM** (Dynamic random access memory)
- **SRAM** (Static random access memory)
- Differ in how they store data:
  - DRAM uses a capacitor
  - SRAM uses cross-coupled inverters

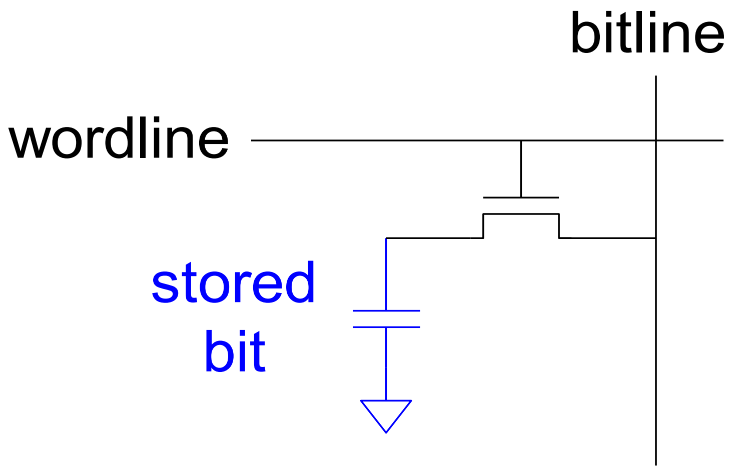
Robert Dennard, 1932 -

- Invented DRAM in 1966 at IBM
- Others were skeptical that the idea would work
- By the mid-1970’s DRAM in virtually all computers

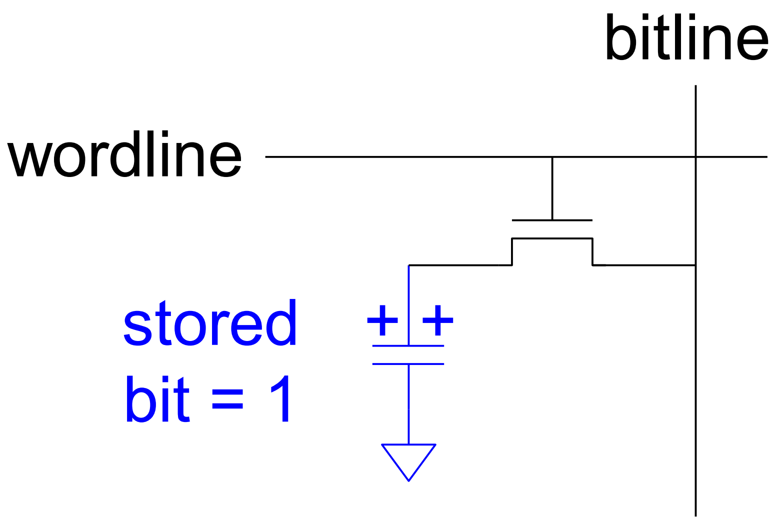
DRAM

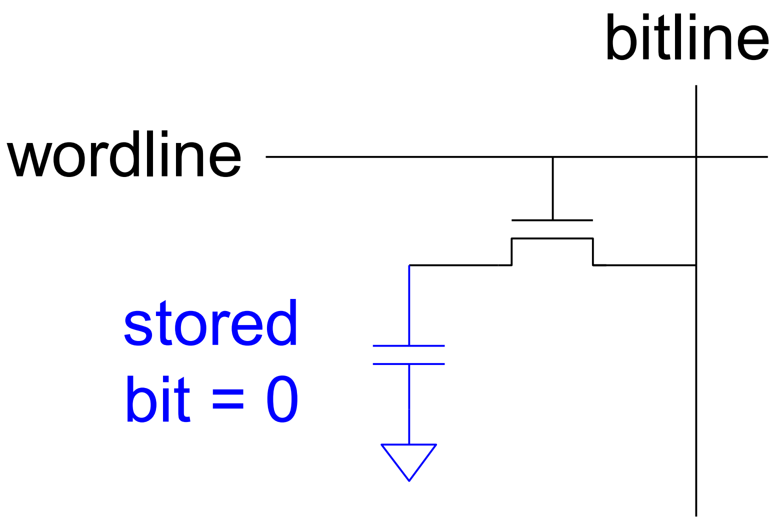
- Data bits stored on **capacitor**
- *Dynamic* because the value needs to be **refreshed** (rewritten) periodically and after read:
  - Charge leakage from the capacitor degrades the value
  - Reading destroys the stored value



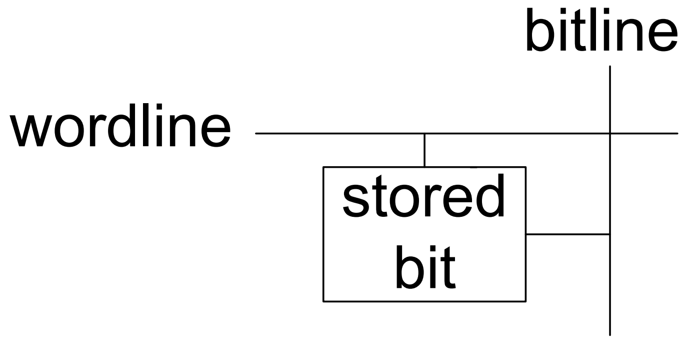


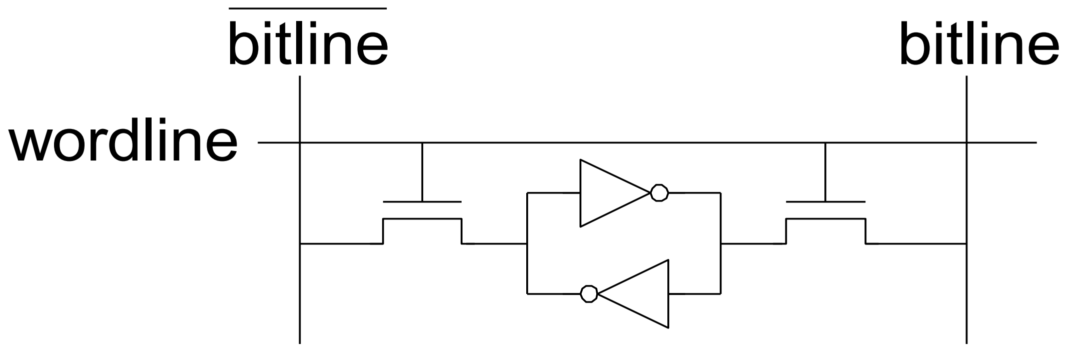
DRAM


bitline
wordline
stored
bit = 1
+ +

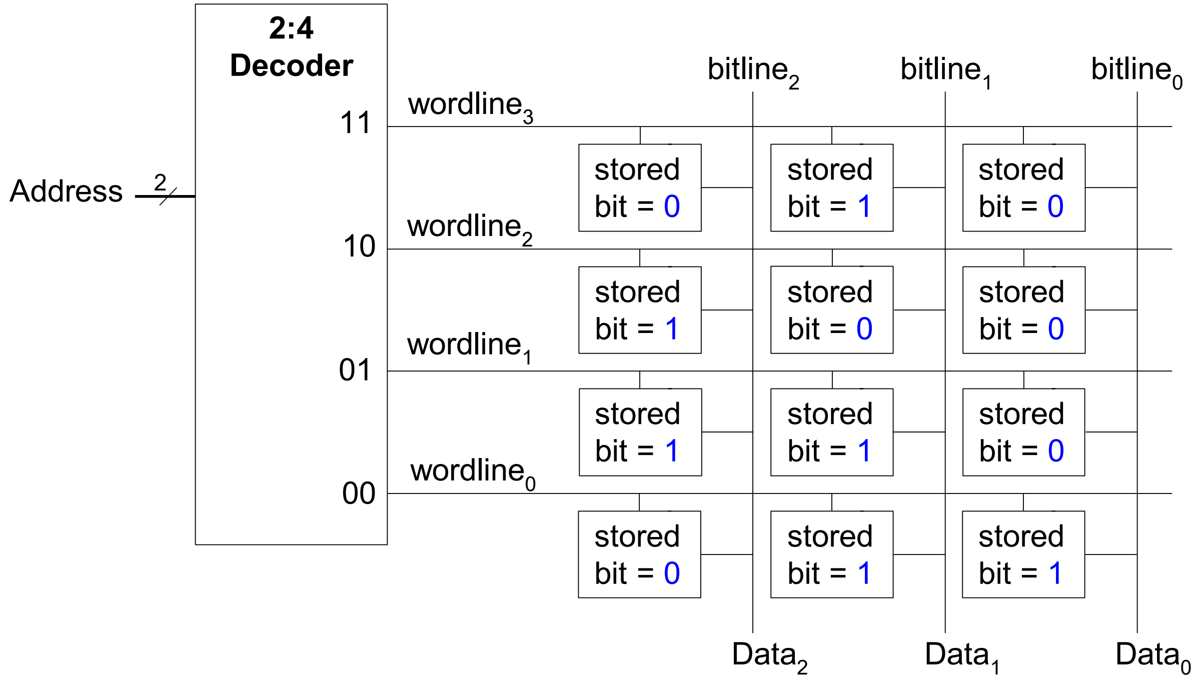

bitline
wordline
stored
bit = 0

SRAM

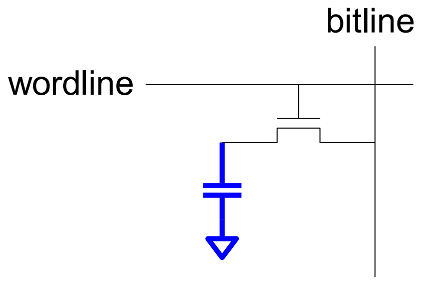

bitline
wordline
stored bit


$\overline{\text{bitline}}$
bitline
wordline

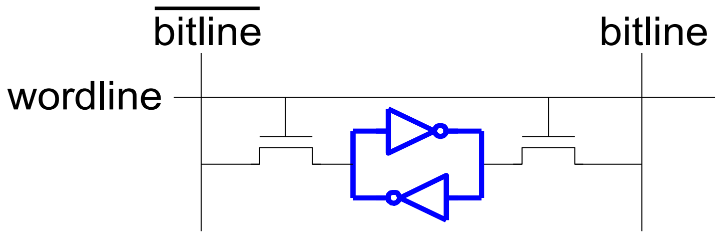
Memory Arrays Review



DRAM bit cell:

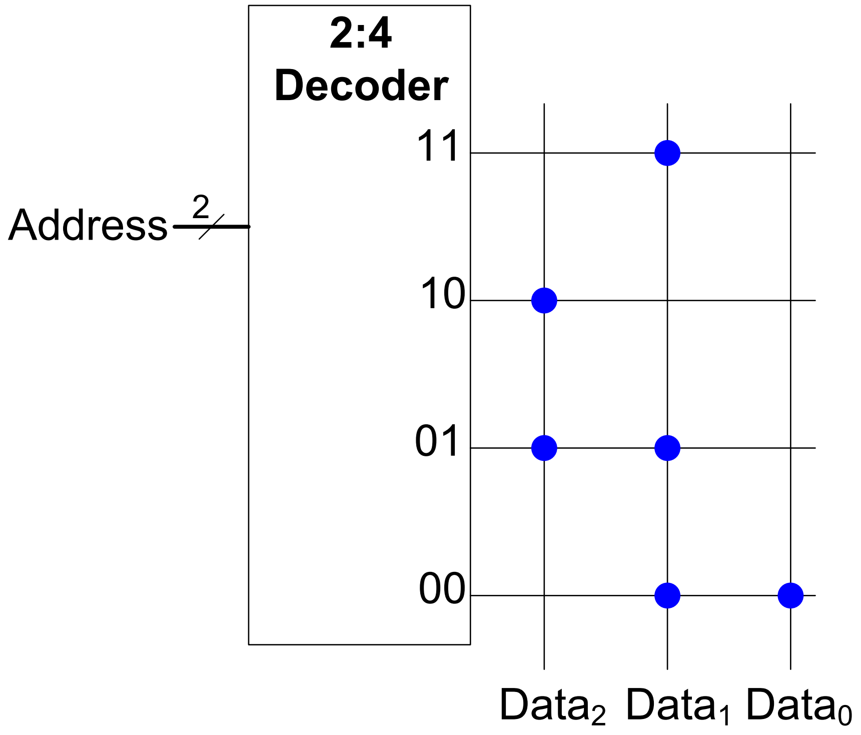


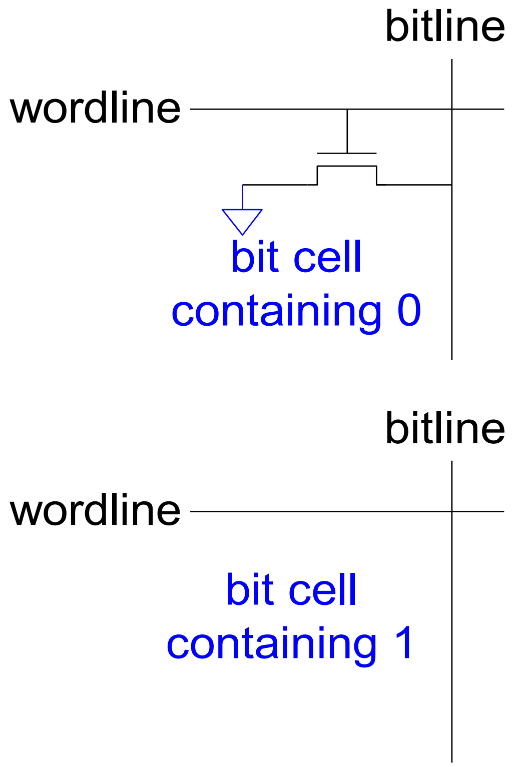
SRAM bit cell:



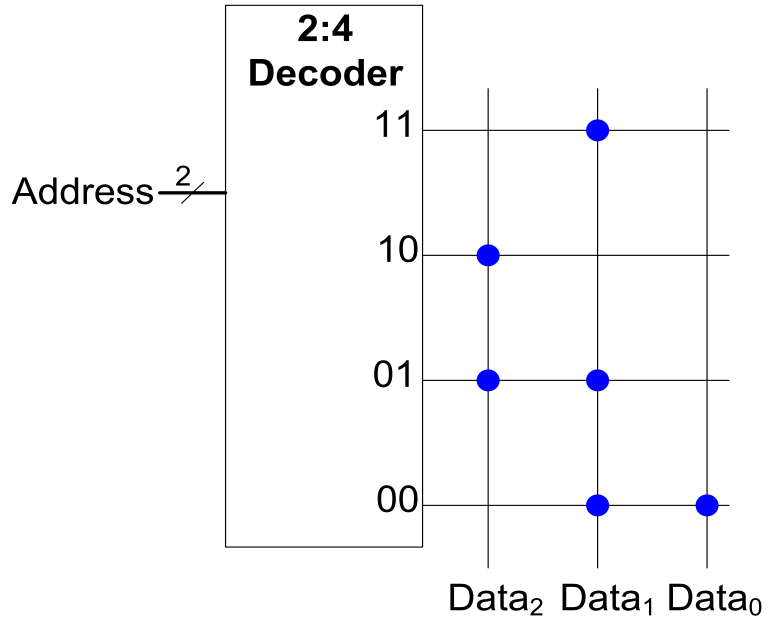
ROM

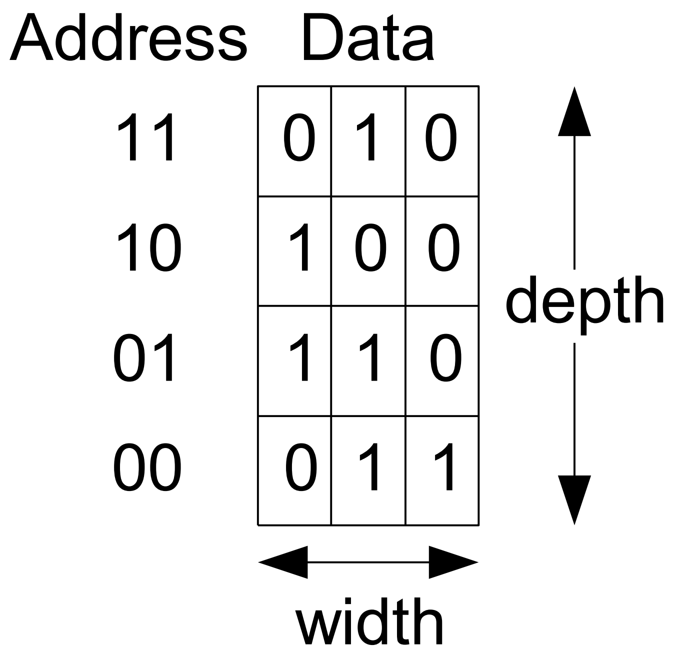
ROM: Dot Notation





ROM Storage



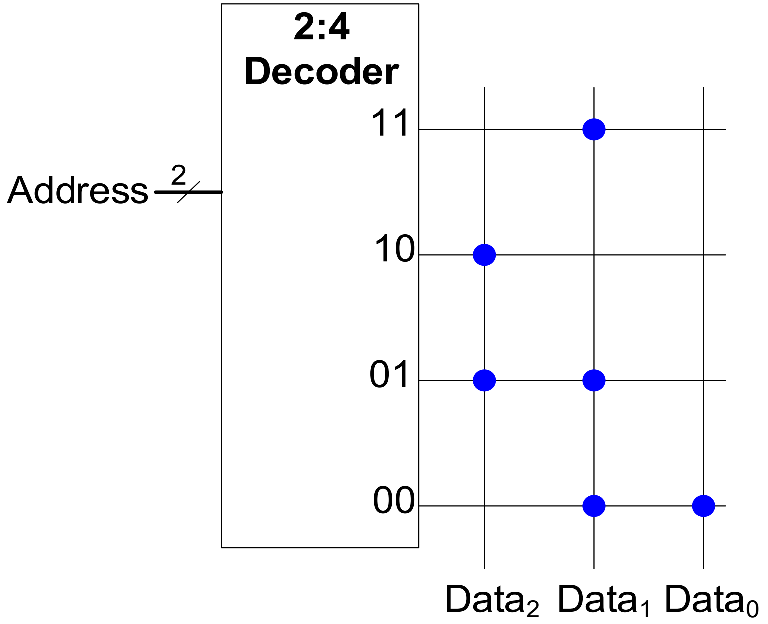

depth
width

Address	Data		
11	0	1	0
10	1	0	0
01	1	1	0
00	0	1	1

Fujio Masuoka, 1944 -

- Developed memories and high speed circuits at Toshiba, 1971-1994
- Invented Flash memory as an unauthorized project pursued during nights and weekends in the late 1970’s
- The process of erasing the memory reminded him of the flash of a camera
- Toshiba slow to commercialize the idea; Intel was first to market in 1988
- Flash has grown into a $25 billion per year market

ROM Logic



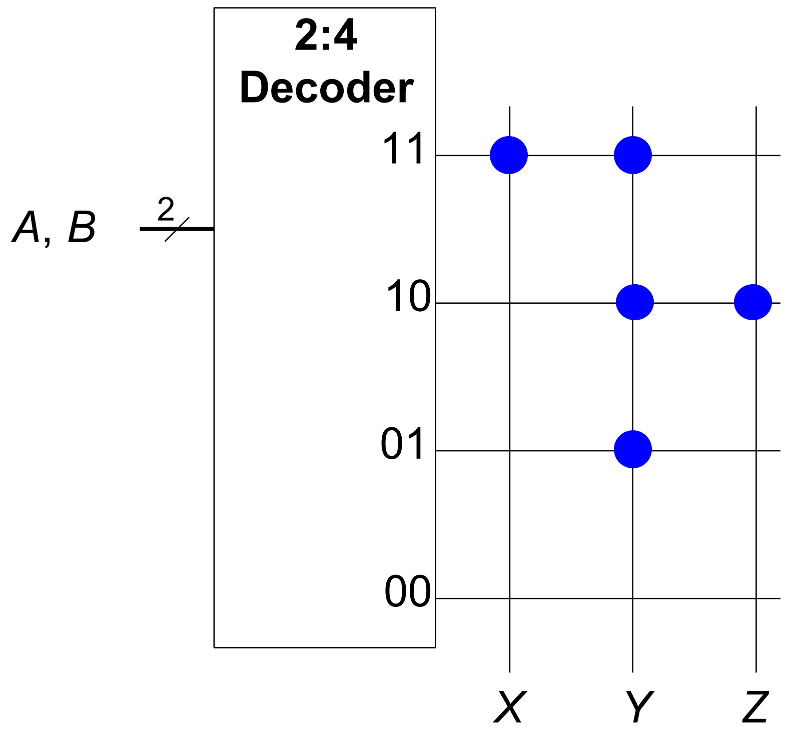
$$Data_2 = A_1 \oplus A_0$$

$$Data_1 = \overline{A}_1 + A_0$$

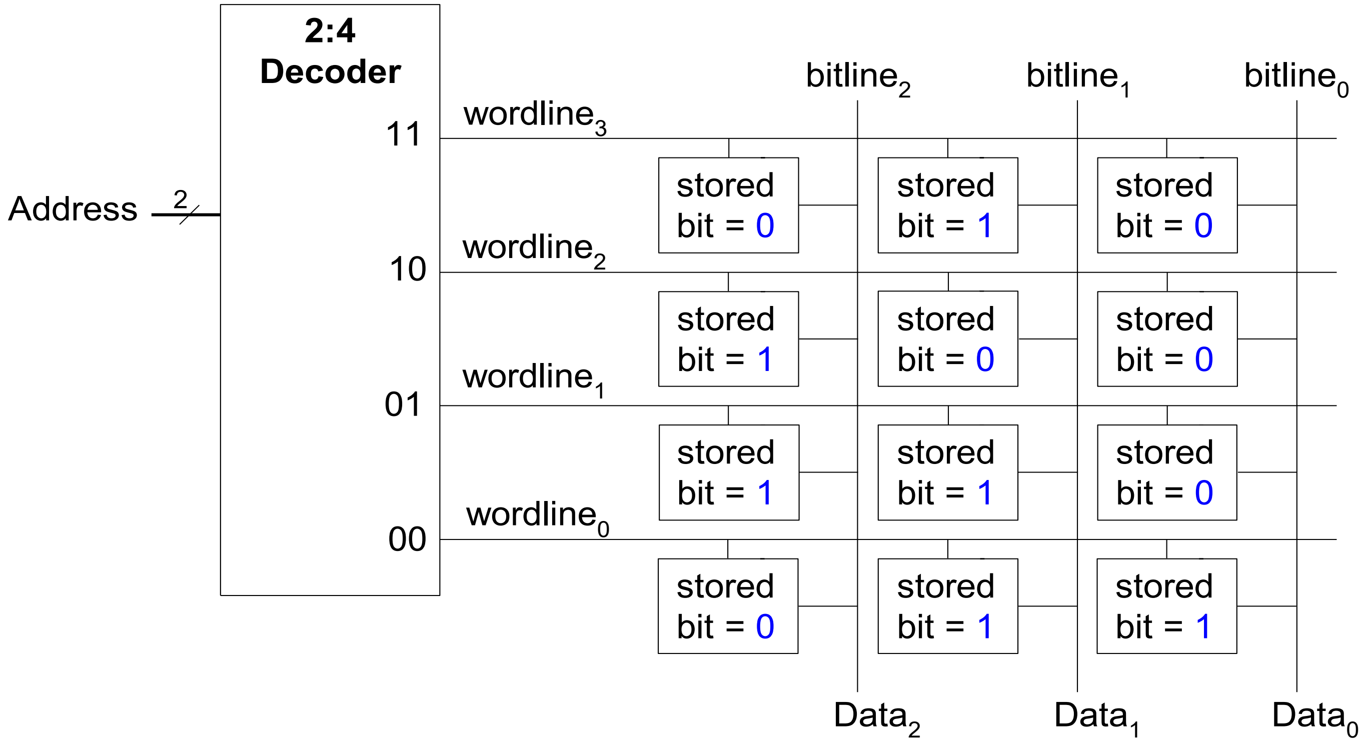
$$Data_0 = \overline{A}_1\overline{A}_0$$

Example: Logic with ROMs

- Implement the following logic functions using a $2^2 \times$ 3-bit ROM:
  - $X = AB$
  - $Y = A + B$
  - $Z = A\overline{B}$



Logic with Any Memory Array



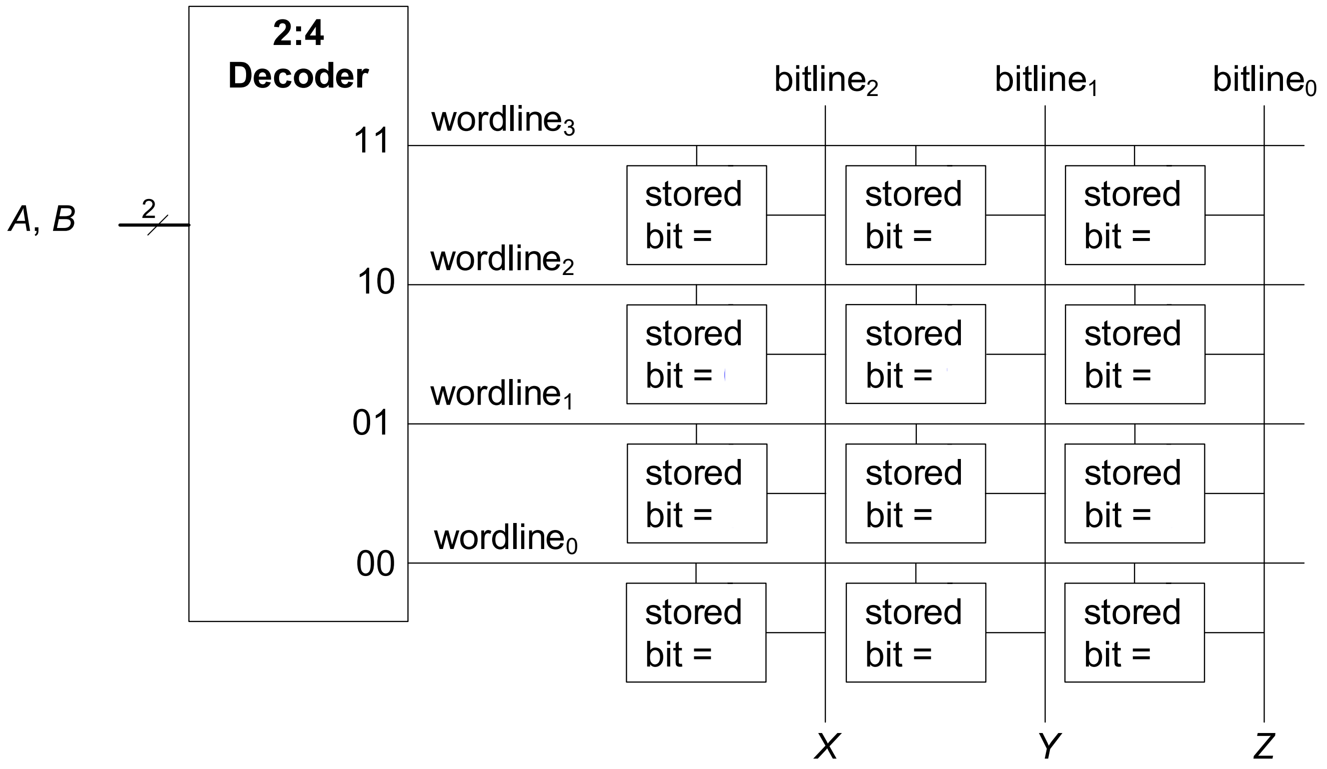
$Data_2 =$

$Data_1 =$

$Data_0 =$

Logic with Memory Arrays

- Implement the following logic functions using a $2^2 \times 3$-bit memory array:
  - $X = AB$
  - $Y = A + B$
  - $Z = A\overline{B}$



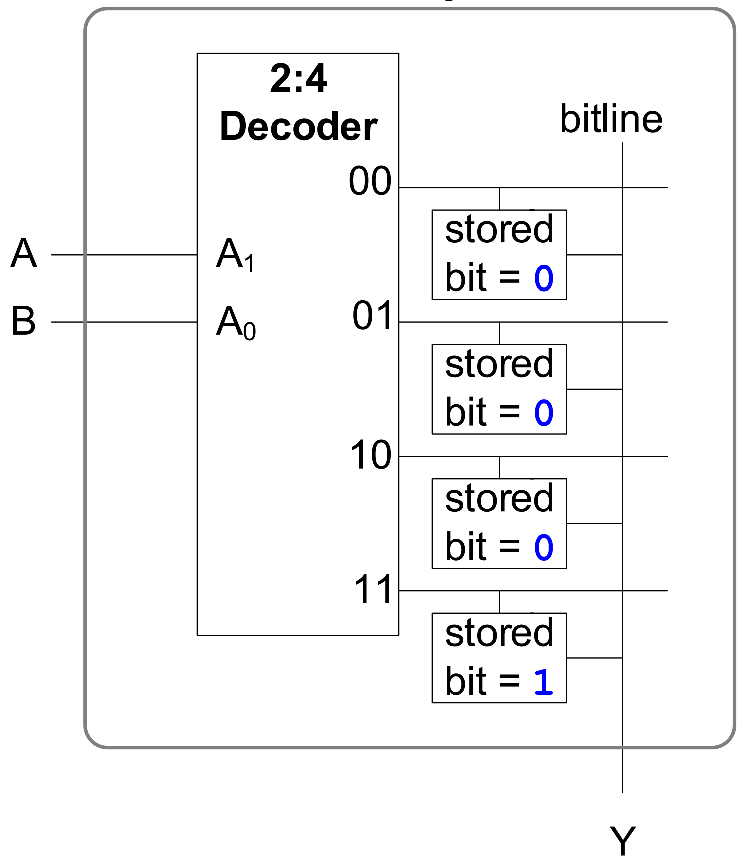
Logic with Memory Arrays

- Called ***lookup tables*** (**LUTs**): look up output at each input combination (address)

Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

4-word x 1-bit Array

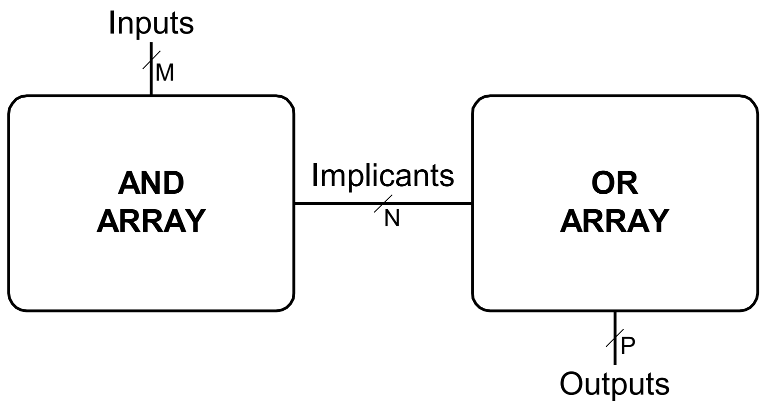


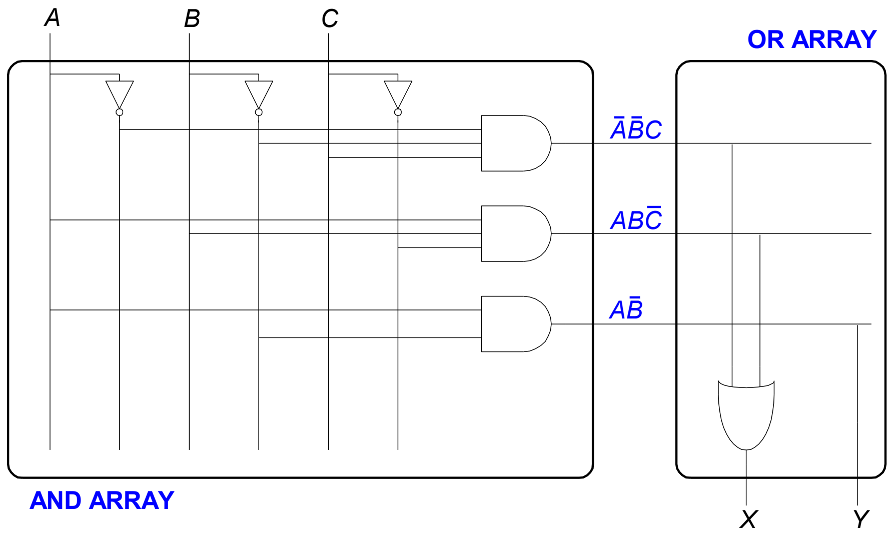
Logic Arrays:
PLAs & FPGAs

- **PLAs** (Programmable logic arrays)
  - AND array followed by OR array
  - Combinational logic only
  - Fixed internal connections
- **FPGAs** (Field programmable gate arrays)
  - Array of Logic Elements (LEs)
  - Combinational and sequential logic
  - Programmable internal connections

PLAs: Programmable Logic Arrays

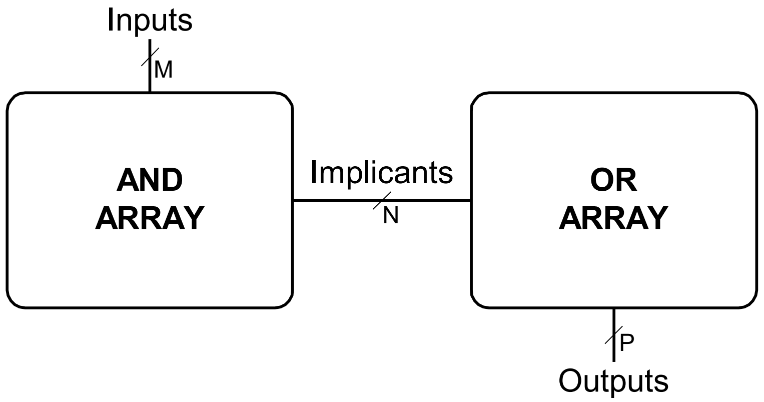
- $X = \bar{A}\bar{B}C + AB\bar{C}$
- $Y = A\bar{B}$

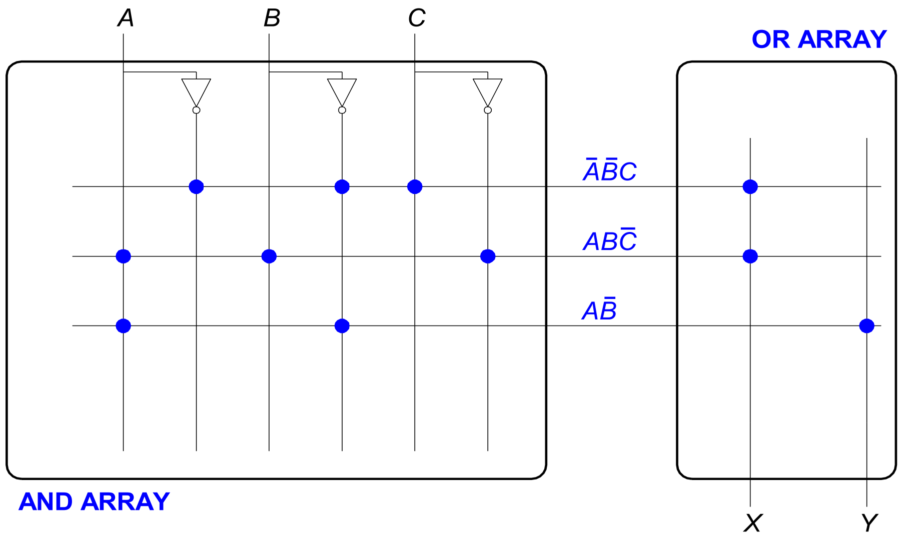




PLAs: Dot Notation

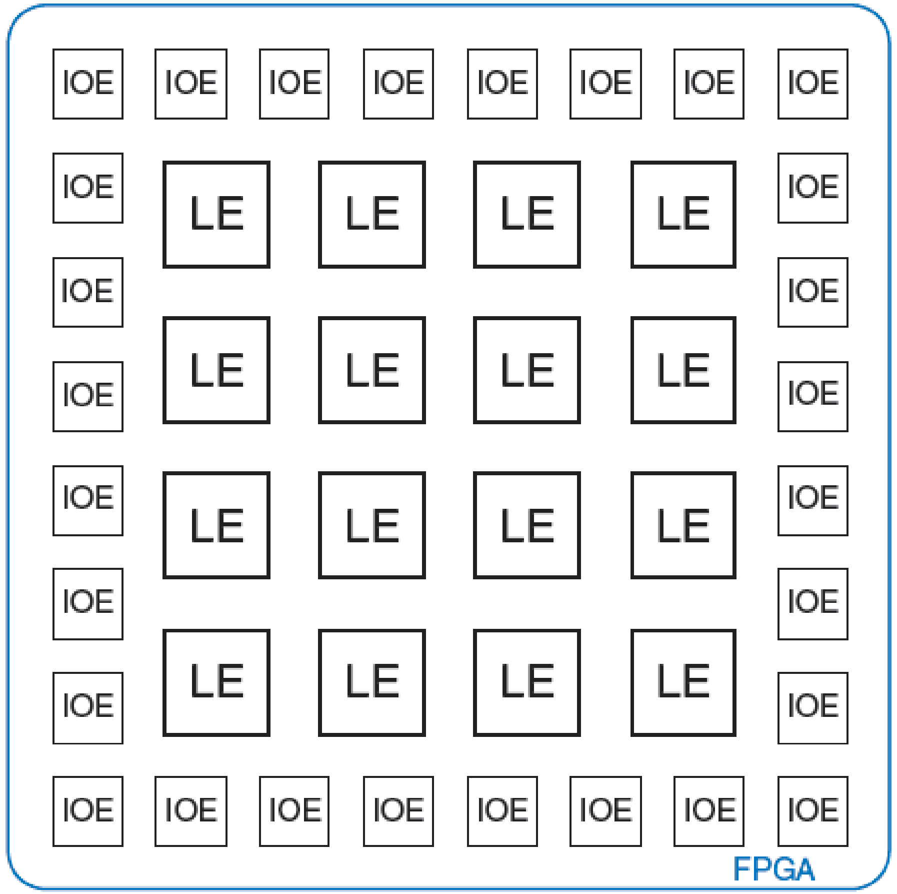
- $X = \bar{A}\bar{B}C + AB\bar{C}$
- $Y = A\bar{B}$





FPGAs: Field Programmable Gate Arrays

- Composed of:
  - **LEs** (Logic elements): perform logic
  - **IOEs** (Input/output elements): interface with outside world
  - **Programmable interconnection:** connect LEs and IOEs
  - Some FPGAs include other building blocks such as multipliers and RAMs



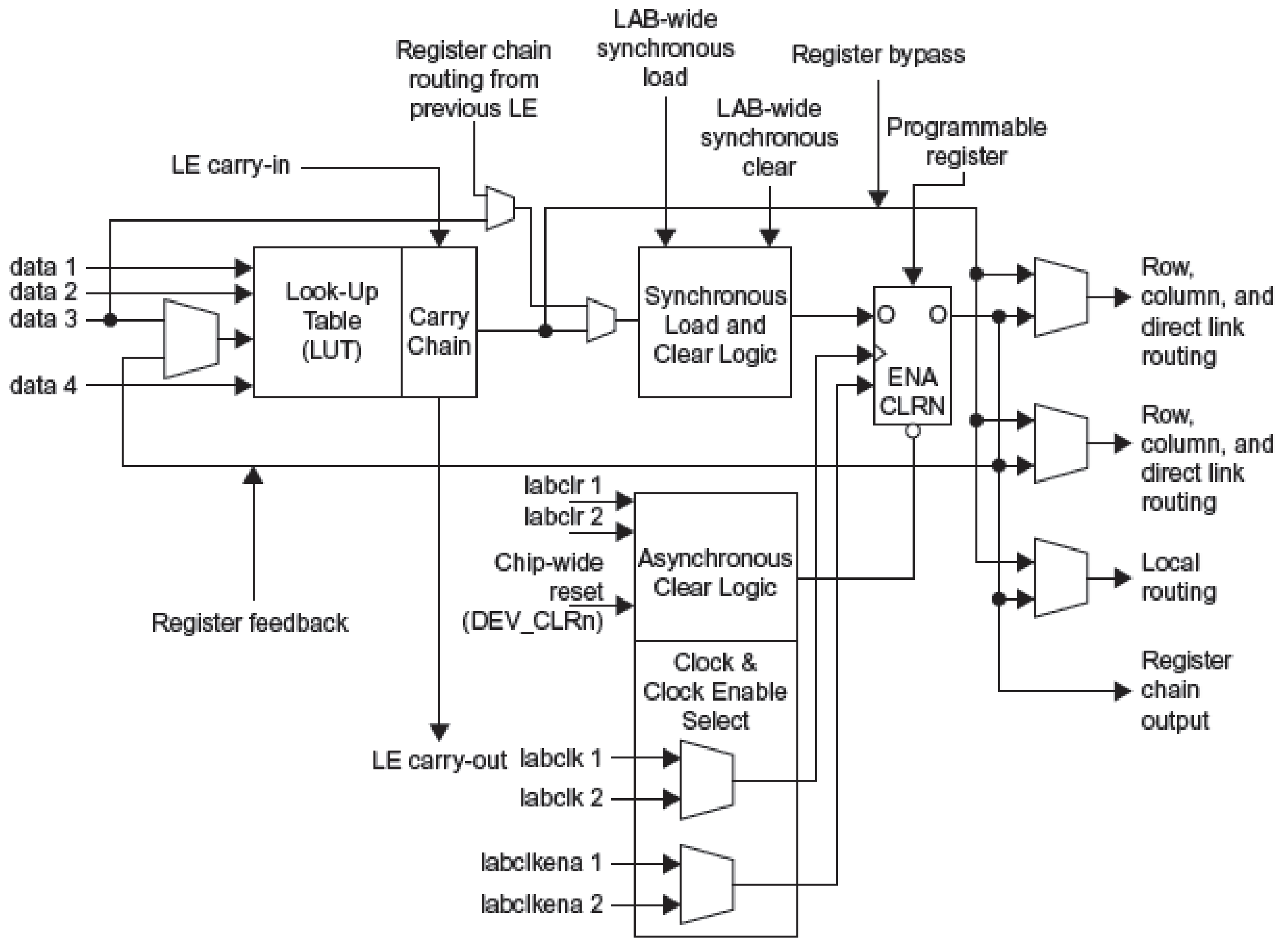
General FPGA Layout

LE: Logic Element

- Composed of:
  - **LUTs** (lookup tables): perform combinational logic
  - **Flip-flops:** perform sequential logic
  - **Multiplexers:** connect LUTs and flip-flops

Altera Cyclone IV LE

- **The Altera Cyclone IV LE has:**
  - 1 four-input **LUT**
  - 1 **registered output**
  - 1 **combinational output**



- *From Cyclone IV datasheet*

FPGA Design Flow

- Using a CAD tool (such as Altera's Quartus II)
    - **Enter the design** using schematic entry or an HDL
    - **Simulate** the design
    - **Synthesize** design and map it onto FPGA
    - **Download the configuration** onto the FPGA
    - **Test** the design