

**CpE358/CS381**

**Switching Theory and  
Logical Design**

**Class 15**

# Today

- Fundamental concepts of digital systems (Mano Chapter 1)
- Binary codes, number systems, and arithmetic (Ch 1)
- Boolean algebra (Ch 2)
- Simplification of switching equations (Ch 3)
- Digital device characteristics (e.g., TTL, CMOS)/design considerations (Ch 10)
- Combinatoric logical design including LSI implementation (Chapter 4)
- Flip-flops and state memory elements (Ch 5)
- Sequential logic analysis and design (Ch 5)
- Counters, shift register circuits (Ch 6)
- Hazards, Races, and time related issues in digital design (Ch 9)
- **Synchronous vs. asynchronous design (Ch 9)**
- **Memory and Programmable logic (Ch 7)**
- Minimization of sequential systems
- Introduction to Finite Automata

# Asynchronous Design

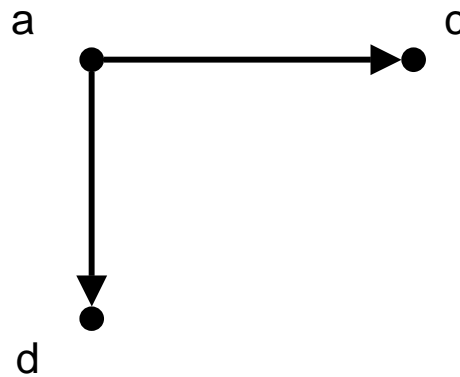
- Problem 9-21: Consider the reduced flow table below.
  - Obtain transition diagram and show that three state variables are needed for a race-free binary state assignment.
  - Obtain the expanded flow table using the ~~multiple~~<sup>shared</sup> row method assignment

	00	01	11	10
a	a	c	a	d
b	a	b	c	b
c	c	c	c	d
d	d	b	a	d

# Asynchronous Design

	00	01	11	10
a	a	c	a	d
b	a	b	c	b
c	c	c	c	d
d	d	b	a	d

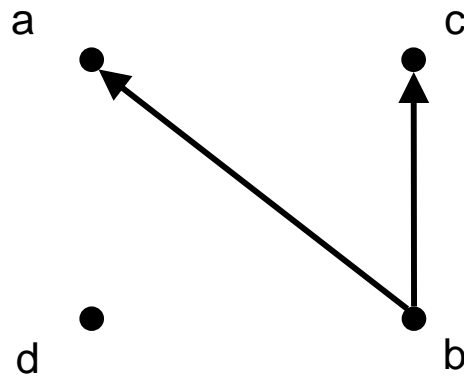
- Create transition diagram:



# Asynchronous Design

	00	01	11	10
a	a	c	a	d
b	a	b	c	b
c	c	c	c	d
d	d	b	a	d

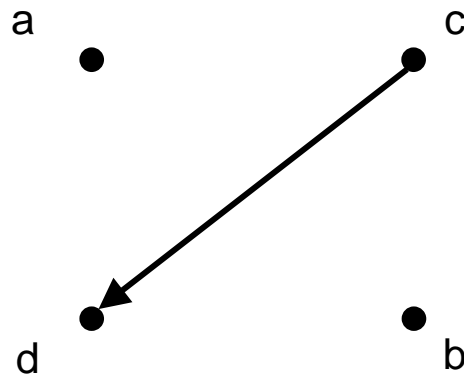
- Create transition diagram:



# Asynchronous Design

	00	01	11	10
a	a	c	a	d
b	a	b	c	b
c	c	c	c	d
d	d	b	a	d

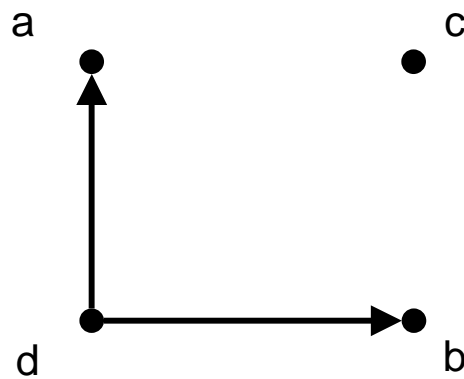
- Create transition diagram:



# Asynchronous Design

	00	01	11	10
a	a	c	a	d
b	a	b	c	b
c	c	c	c	d
d	d	b	a	d

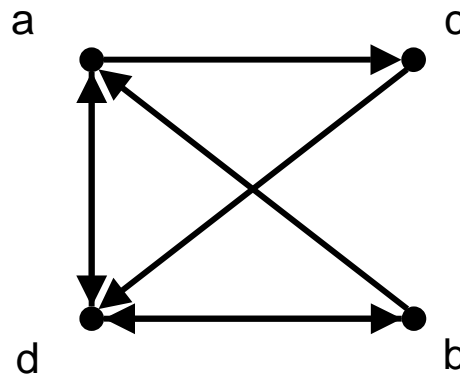
- Create transition diagram:



# Asynchronous Design

	00	01	11	10
a	a	c	a	d
b	a	b	c	b
c	c	c	c	d
d	d	b	a	d

- Create transition diagram:

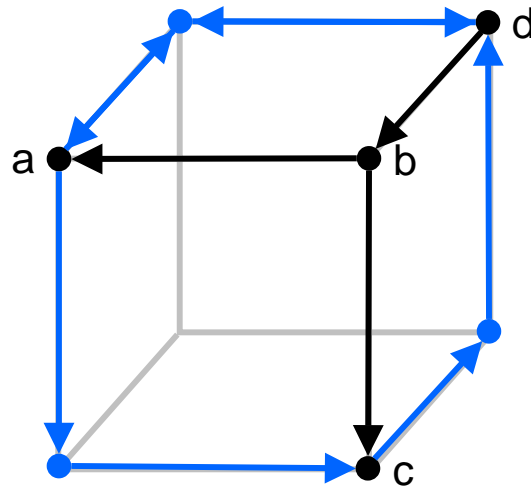


If there were only vertical or horizontal transitions, only 1 state variable would change. Since there are horizontal, vertical, and diagonal transitions, races are unavoidable with 2 state variables.

# Asynchronous Design

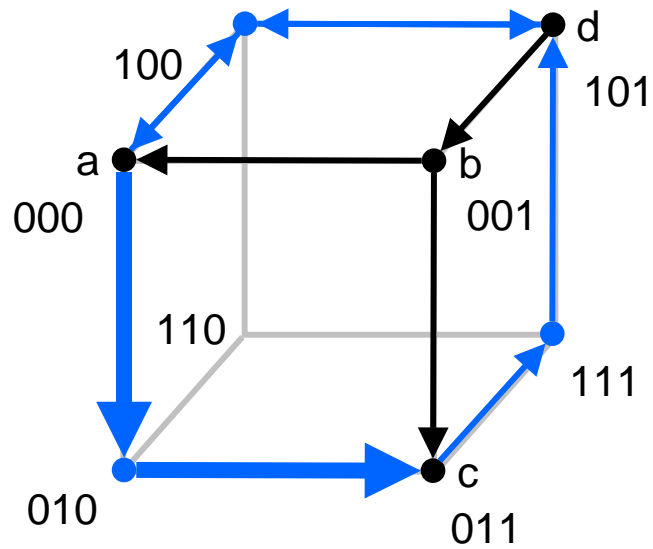
	00	01	11	10
a	a	c	a	d
b	a	b	c	b
c	c	c	c	d
d	d	b	a	d

- Consider this transition diagram:



The third state variable creates alternative paths that only require one state variable change at a time.

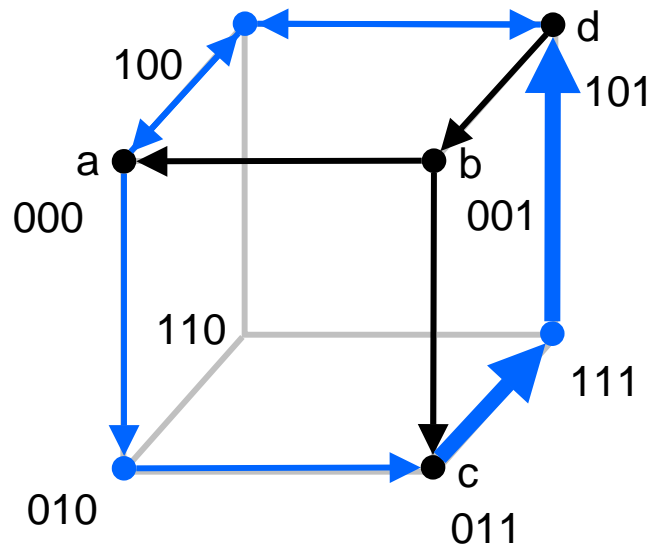
# Asynchronous Design



	00	01	11	10
a	a	W	a	d
b	a	b	c	b
c	c	c	c	d
d	d	b	a	d
W		c		

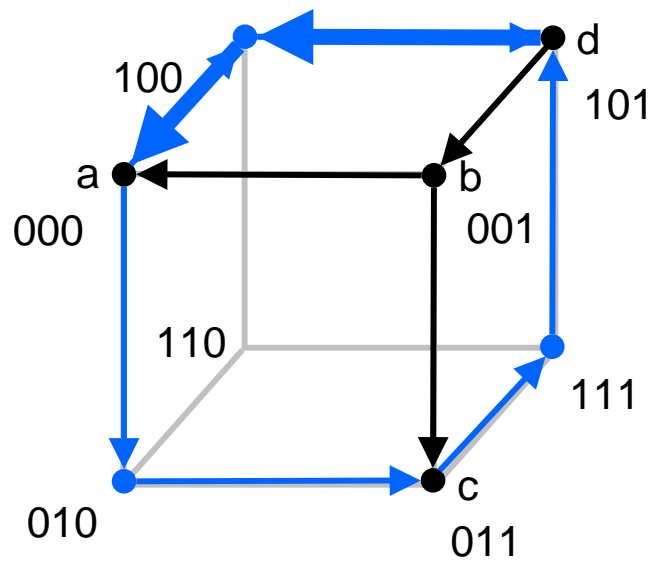
Replace transitions that would require change in more than 1 variable with two transitions, each requiring single change.

# Asynchronous Design



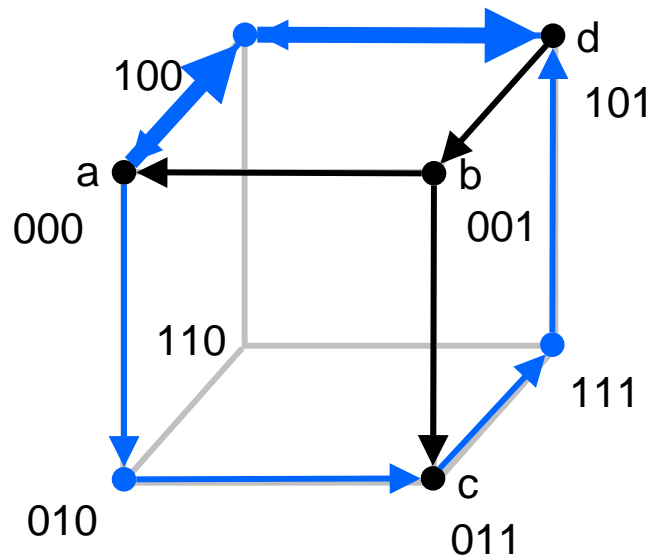
	00	01	11	10
a	a	W	a	d
b	a	b	c	b
c	c	c	c	X
d	d	b	a	d
W		c		
X				d

# Asynchronous Design



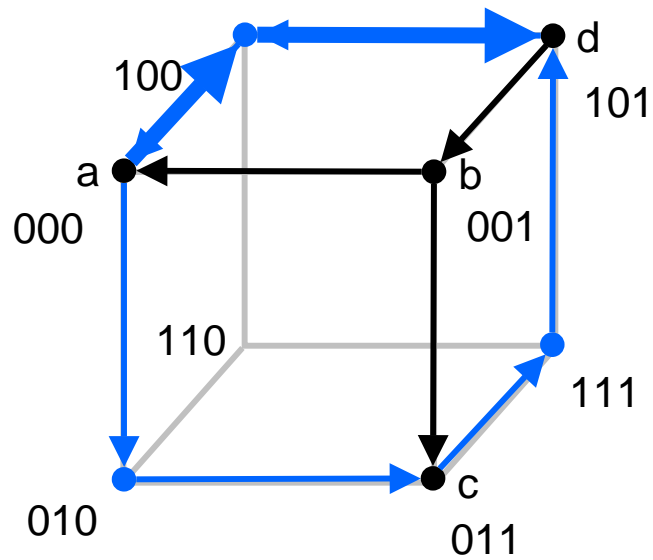
	00	01	11	10
a	a	W	a	d
b	a	b	c	b
c	c	c	c	X
d	d	b	Y	d
W		c		
X				d
Y			a	

# Asynchronous Design



	00	01	11	10
a	a	W	a	Y
b	a	b	c	b
c	c	c	c	X
d	d	b	Y	d
W		c		
X				d
Y			a	d

# Asynchronous Design

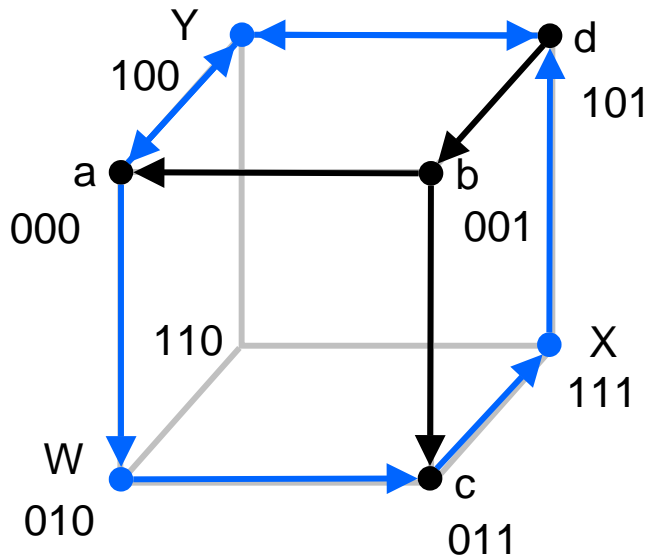


	00	01	11	10
a	a	W	a	Y
b	a	b	c	b
c	c	c	c	X
d	d	b	Y	d
W	-	c	-	-
X	-	-	-	d
Y	-	-	a	d
Z	-	-	-	-

Mark other conditions as don't cares

Note: extra states are not stable states.

# Asynchronous Design



	00	01	11	10
000	000	010	000	100
001	000	001	011	001
011	011	011	011	111
101	101	001	100	101
010	-	011	-	-
111	-	-	-	101
100	-	-	000	101
110	-	-	-	-

Make binary state assignments

# Asynchronous Design

	00	01	11	10
000	000	010	000	100
001	000	001	011	001
010	-	011	-	-
011	011	011	011	111
100	-	-	000	101
101	101	001	100	101
110	-	-	-	-
111	-	-	-	101

Sort transition table

# Asynchronous Design

	$x_1x_2$			
$y_1y_2y_3$	00	01	11	10
000	000	010	000	100
001	000	001	011	001
010	-	011	-	-
011	011	011	011	111
100	-	-	000	101
101	101	001	100	101
110	-	-	-	-
111	-	-	-	101

Create excitation tables

$Y_1$

$y_1=0$		$x_1x_2$			
		00	01	11	10
$y_2y_3$	00	0	0	0	1
	01	0	0	0	0
	11	X	0	X	X
	10	0	0	0	1

$y_1=1$		$x_1x_2$			
		00	01	11	10
$y_2y_3$	00	X	X	0	1
	01	1	0	1	1
	11	X	X	X	X
	10	X	X	X	1

# Asynchronous Design

	$x_1x_2$			
$y_1y_2y_3$	00	01	11	10
000	000	010	000	100
001	000	001	011	001
010	-	011	-	-
011	011	011	011	111
100	-	-	000	101
101	101	001	100	101
110	-	-	-	-
111	-	-	-	101

Create excitation tables

$$Y_1 = x_1x_2'y_3' + x_2'y_1 + x_1y_1y_3$$

$y_1=0$		$x_1x_2$			
		00	01	11	10
$y_2y_3$	00	0	0	0	1
	01	0	0	0	0
	11	X	0	X	X
	10	0	0	0	1

$y_1=1$		$x_1x_2$			
		00	01	11	10
$y_2y_3$	00	X	X	0	1
	01	1	0	1	1
	11	X	X	X	X
	10	X	X	X	1

# Asynchronous Design

Create excitation tables

	$x_1x_2$			
$y_1y_2y_3$	00	01	11	10
000	000	010	000	100
001	000	001	011	001
010	-	011	-	-
011	011	011	011	111
100	-	-	000	101
101	101	001	100	101
110	-	-	-	-
111	-	-	-	101

$Y_2$

$y_1=0$		$x_1x_2$			
		00	01	11	10
$y_2y_3$	00	0	1	0	0
	01	0	0	1	0
	11	X	1	X	X
	10	1	1	1	1

$y_1=1$		$x_1x_2$			
		00	01	11	10
$y_2y_3$	00	X	X	0	0
	01	0	0	0	0
	11	X	X	X	X
	10	X	X	X	0

# Asynchronous Design

	$x_1x_2$			
$y_1y_2y_3$	00	01	11	10
000	000	010	000	100
001	000	001	011	001
010	-	011	-	-
011	011	011	011	111
100	-	-	000	101
101	101	001	100	101
110	-	-	-	-
111	-	-	-	101

Create excitation tables

$$Y_2 = y_1'y_2 + x_1'x_2y_3' + x_1x_2y_1'y_3$$

$y_1=0$		$x_1x_2$			
		00	01	11	10
$y_2y_3$	00	0	1	0	0
	01	0	0	1	0
	11	X	1	X	X
	10	1	1	1	1

$y_1=1$		$x_1x_2$			
		00	01	11	10
$y_2y_3$	00	X	X	0	0
	01	0	0	0	0
	11	X	X	X	X
	10	X	X	X	0

# Asynchronous Design

Create excitation tables

	$x_1x_2$			
$y_1y_2y_3$	00	01	11	10
000	000	010	000	100
001	000	001	011	001
010	-	011	-	-
011	011	011	011	111
100	-	-	000	101
101	101	001	100	101
110	-	-	-	-
111	-	-	-	101

$Y_3$

$y_1=0$		$x_1x_2$			
		00	01	11	10
$y_2y_3$	00	0	0	0	0
	01	0	1	1	1
	11	X	1	X	X
	10	1	1	1	1

$y_1=1$		$x_1x_2$			
		00	01	11	10
$y_2y_3$	00	X	X	0	1
	01	1	1	0	1
	11	X	X	X	X
	10	X	X	X	1

# Asynchronous Design

	$x_1x_2$			
$y_1y_2y_3$	00	01	11	10
000	000	010	000	100
001	000	001	011	001
010	-	011	-	-
011	011	011	011	111
100	-	-	000	101
101	101	001	100	101
110	-	-	-	-
111	-	-	-	101

Create excitation tables

$$Y_3 = y_2 + x_1'y_1 + x_2'y_1 + x_2y_1'y_3 + x_1y_1'y_3$$

$y_1=0$		$x_1x_2$			
		00	01	11	10
$y_2y_3$	00	0	0	0	0
	01	0	1	1	1
	11	X	1	X	X
	10	1	1	1	1

$y_1=1$		$x_1x_2$			
		00	01	11	10
$y_2y_3$	00	X	X	0	1
	01	1	1	0	1
	11	X	X	X	X
	10	X	X	X	1

# Designing to Avoid Static Hazards

- Problem 9-22:

Find a circuit that has no static hazards and implements the Boolean function

$$F(A,B,C,D) = \Sigma(0,2,6,7,8,10,12)$$

# Designing to Avoid Static Hazards

- Problem 9-22:

Find a circuit that has no static hazards and implements the Boolean function

$$F(A,B,C,D) = \Sigma(0,2,6,7,8,10,12)$$

- Plot minterms on Karnaugh map

		CD			
		00	01	11	10
AB	00	1	0	0	1
	01	0	0	1	1
	11	1	0	0	0
	10	1	0	0	1

# Designing to Avoid Static Hazards

- Problem 9-22:

Find a circuit that has no static hazards and implements the Boolean function

$$F(A,B,C,D) = \Sigma(0,2,6,7,8,10,12)$$

- Find minimum coverage of 1's

		CD			
		00	01	11	10
AB	00	1	0	0	1
	01	0	0	1	1
	11	1	0	0	0
	10	1	0	0	1

$$F(A,B,C,D) = B'D' + A'BC + AC'D'$$

# Designing to Avoid Static Hazards

- Problem 9-22:

Find a circuit that has no static hazards and implements the Boolean function

$$F(A,B,C,D) = \Sigma(0,2,6,7,8,10,12)$$

- Find minimum coverage of 1's

		CD			
		00	01	11	10
AB	00	1	0	0	1
	01	0	0	1	1
	11	1	0	0	0
	10	1	0	0	1

Hazard

$$F(A,B,C,D) = B'D' + A'BC + AC'D'$$

# Designing to Avoid Static Hazards

- Problem 9-22:

Find a circuit that has no static hazards and implements the Boolean function

$$F(A,B,C,D) = \Sigma(0,2,6,7,8,10,12)$$

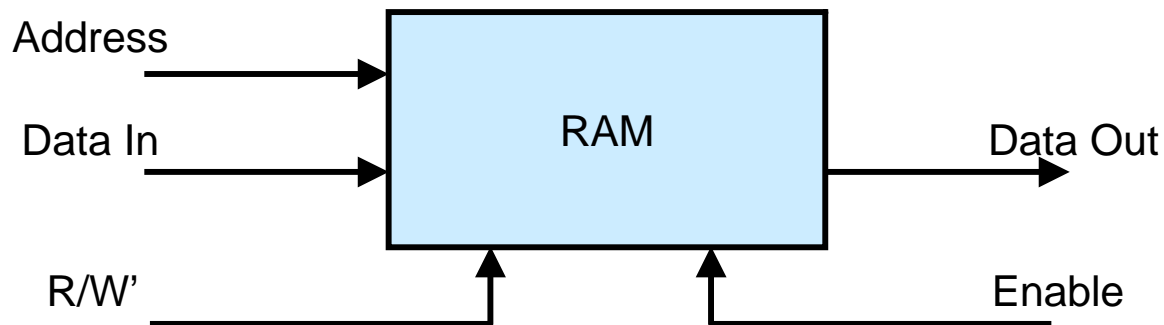
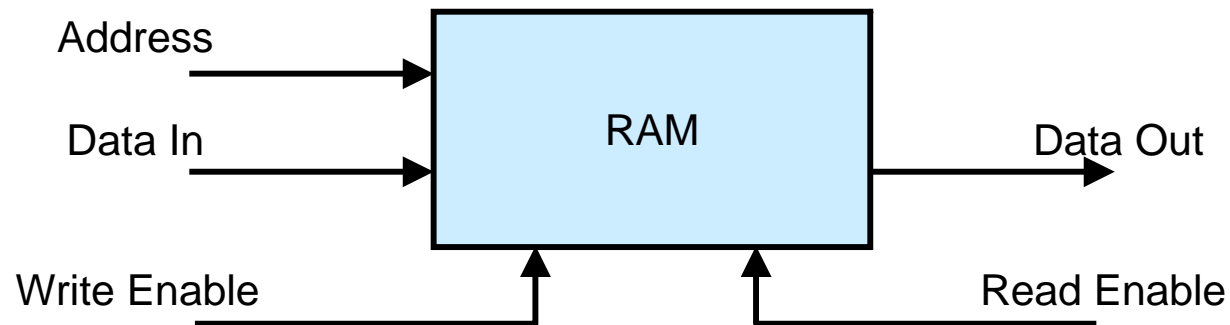
- Find minimum coverage of 1's

		CD			
		00	01	11	10
AB	00	1	0	0	1
	01	0	0	1	1
	11	1	0	0	0
	10	1	0	0	1

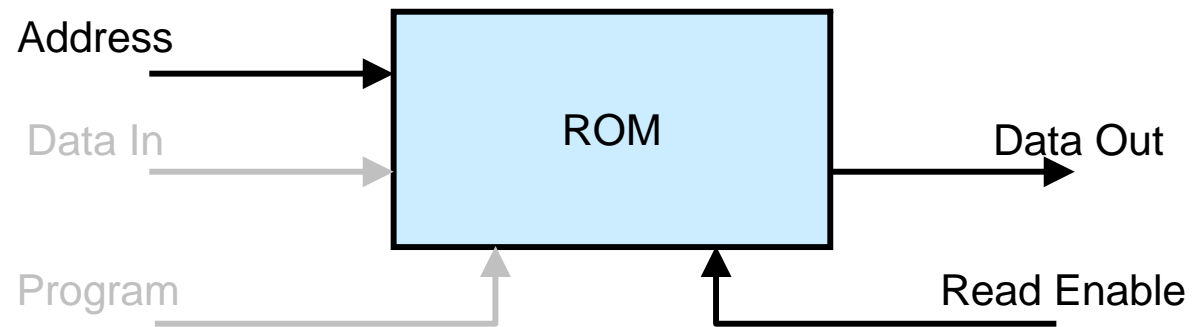
Hazard Covered

$$F(A,B,C,D) = B'D' + A'BC + AC'D' + A'CD'$$

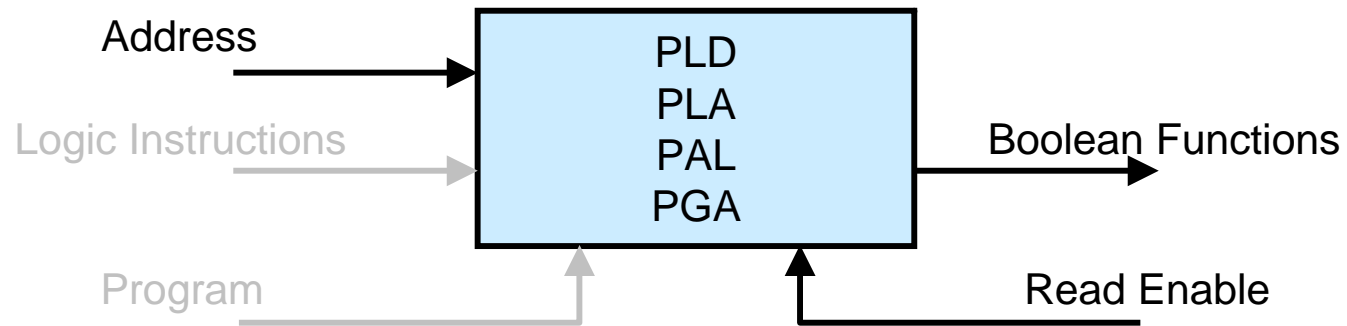
# Memory and Programmable Logic



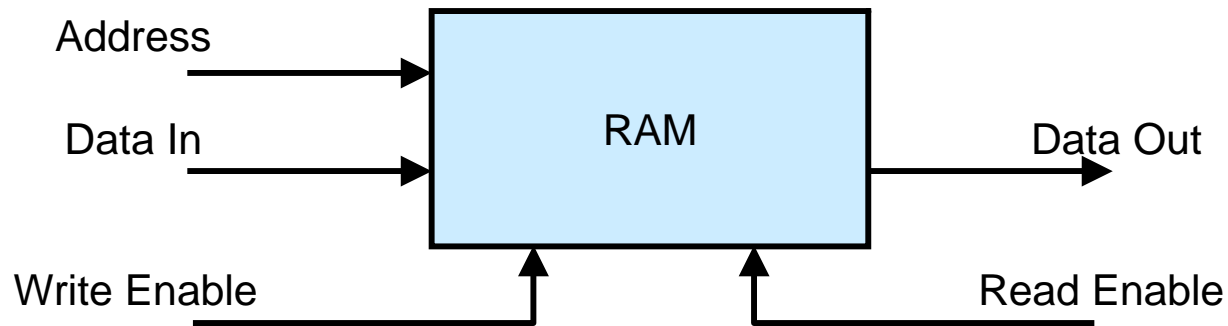
# Memory and Programmable Logic



# Memory and Programmable Logic

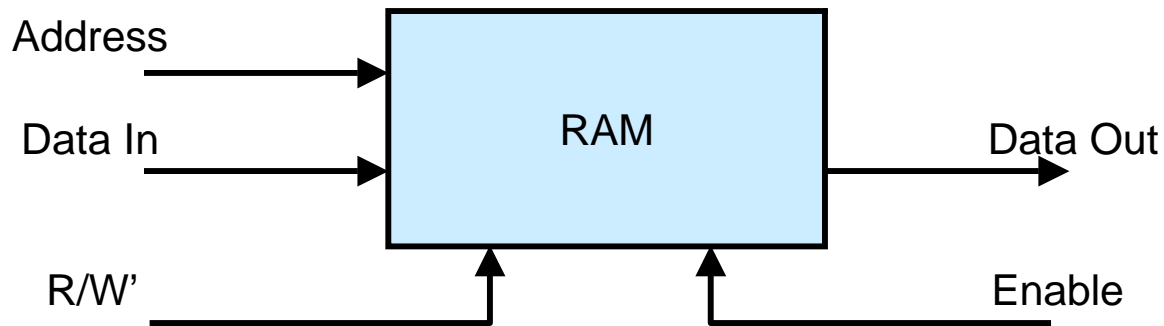


# RAM



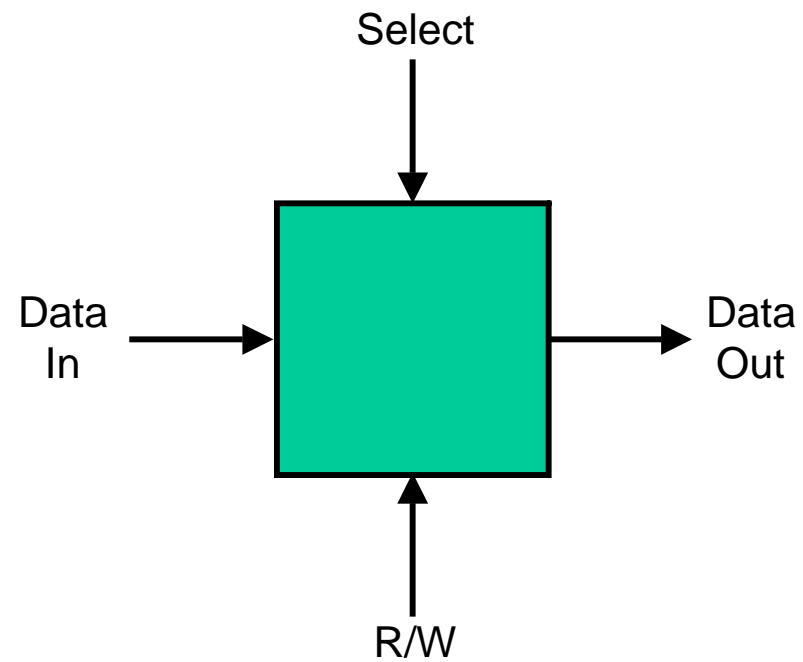
Address	Data In	WE	RE	Data Out	Operation
XXXX	XXXX	0	0	?	None
ABCD	XXXX	0	1	Contents(ABCD)	Read Contents of ABCD
ABCD	MNOP	1	0	?	Write value MNOP to ABCD

# RAM

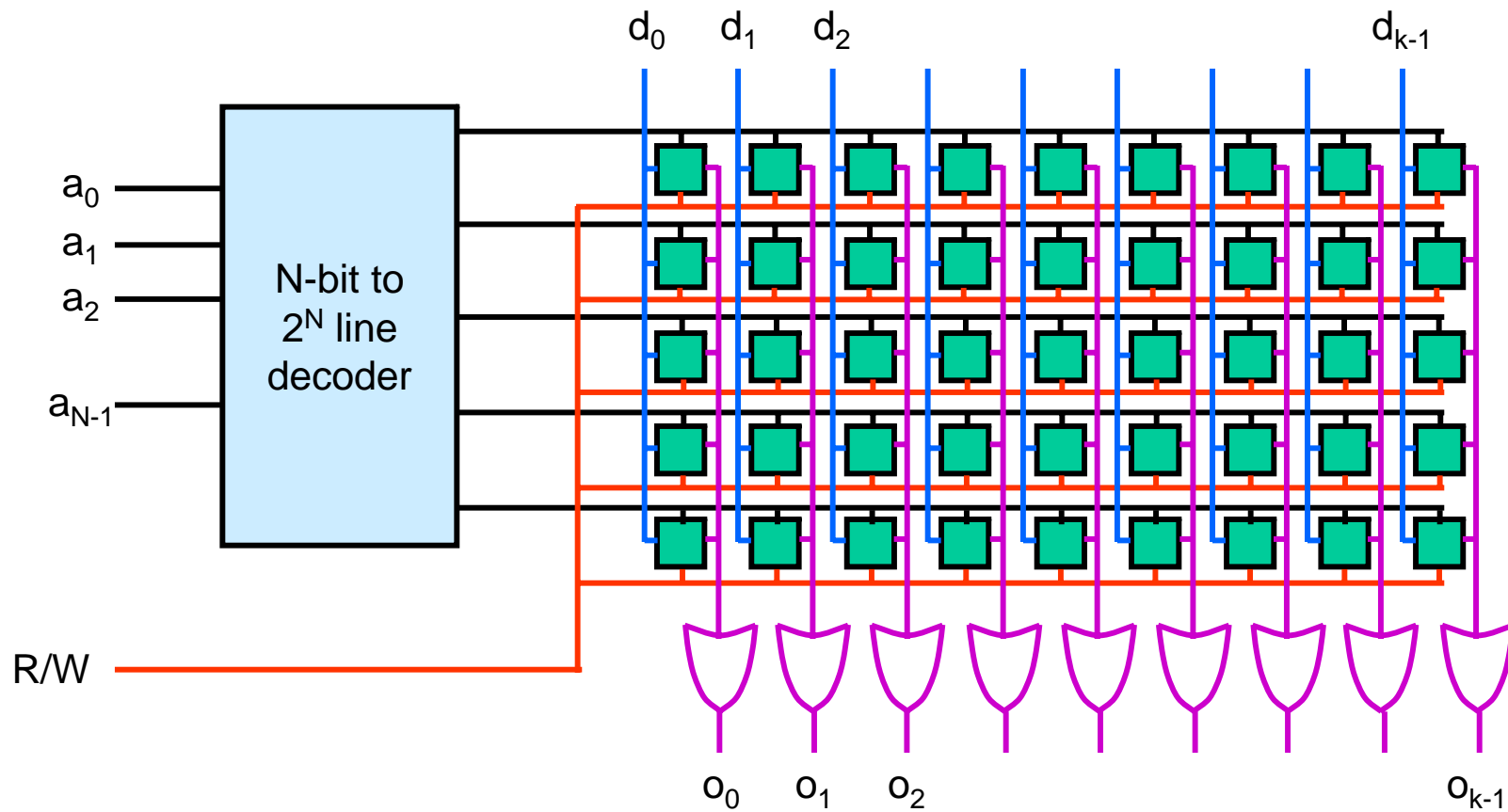


Address	Data In	Enable	R/W'	Data Out	Operation
XXXX	XXXX	0	0	?	None
ABCD	MNOP	1	0	?	Write value MNOP to ABCD
ABCD	XXXX	1	1	Contents(ABCD)	Read Contents of ABCD

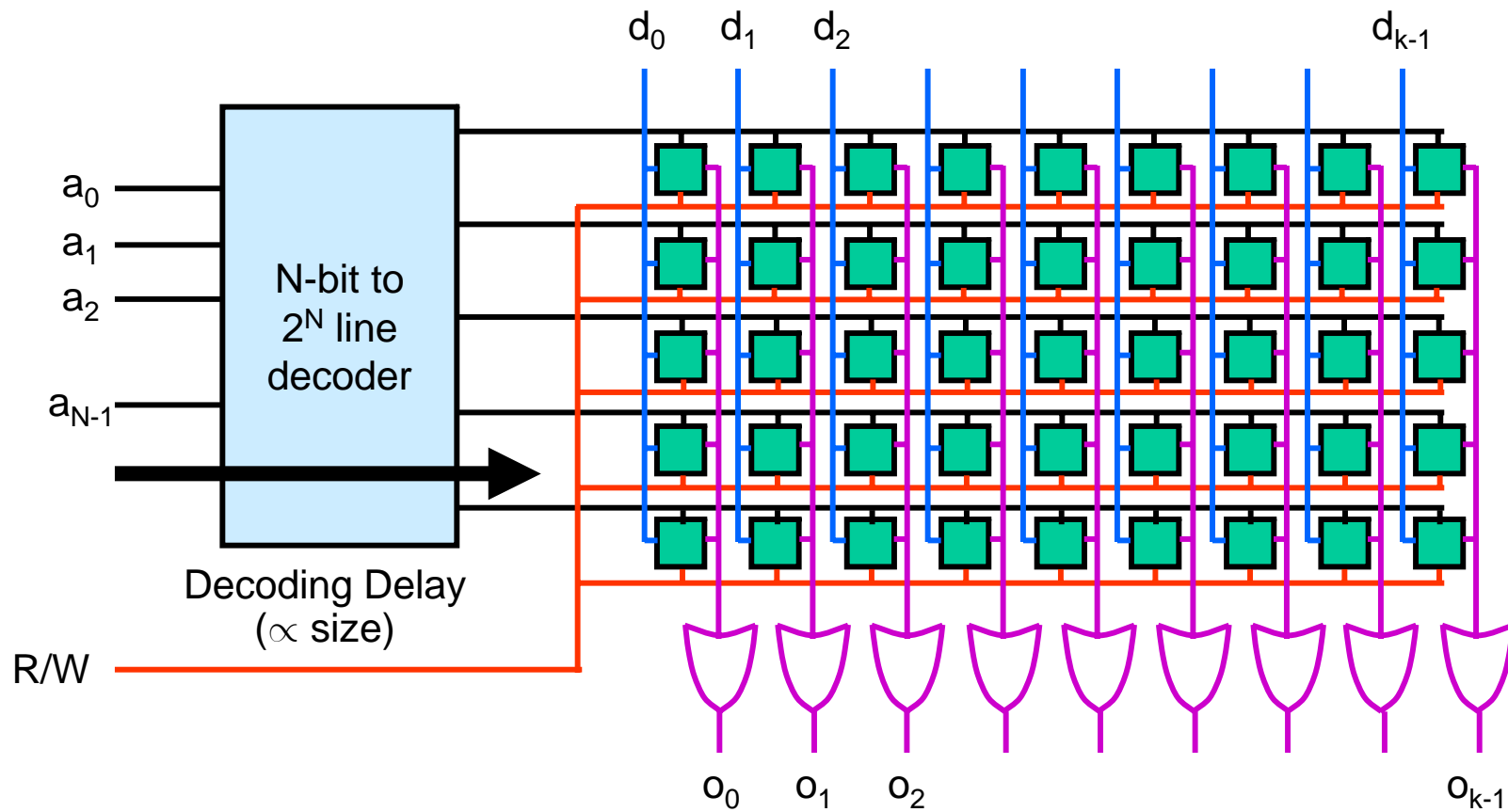
# RAM Memory Cells



# Addressing Memory



# Addressing Memory

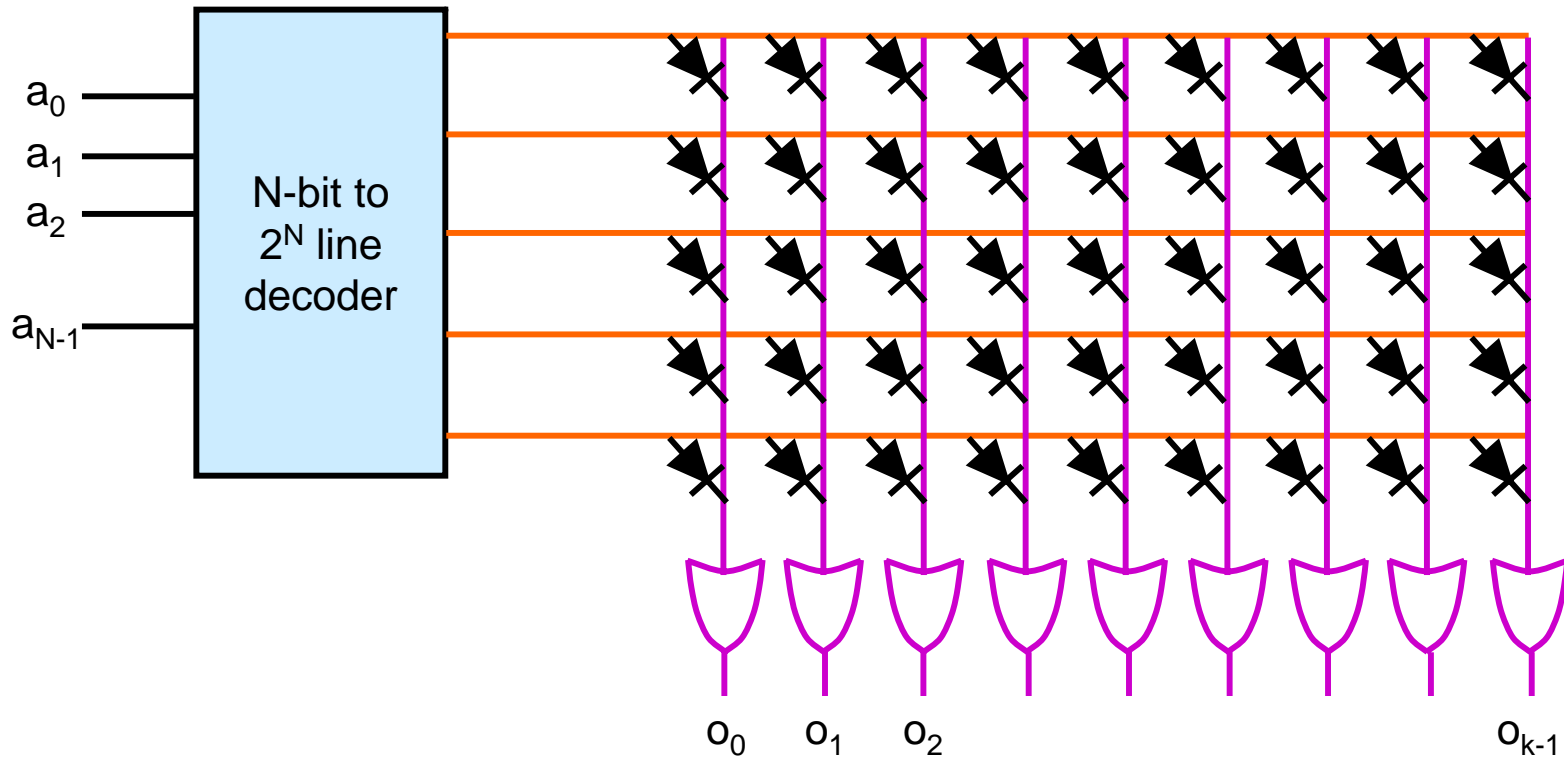


# RAM Variants

- Static RAM
  - Memory contents are maintained as long as power is applied
  - Memory cells are more complex, so physical size is larger
  - CMOS technology allows battery backup for long periods
- Dynamic RAM
  - Memory contents fade with time, requires “refreshing”
  - Simpler memory cells, so physical size is smaller
  - Requires memory controller to perform maintenance functions

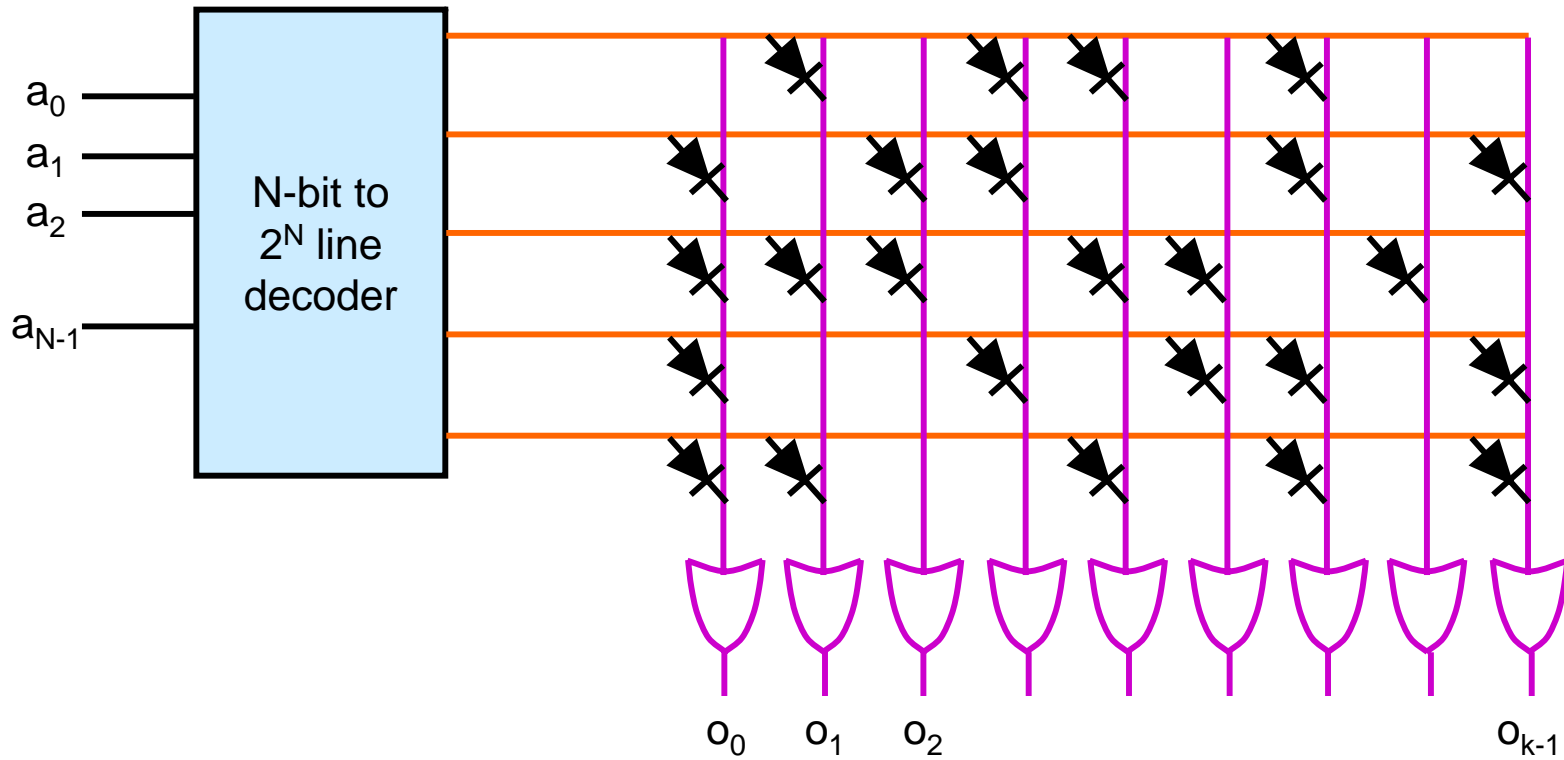
# ROM

- Unprogrammed ROM:



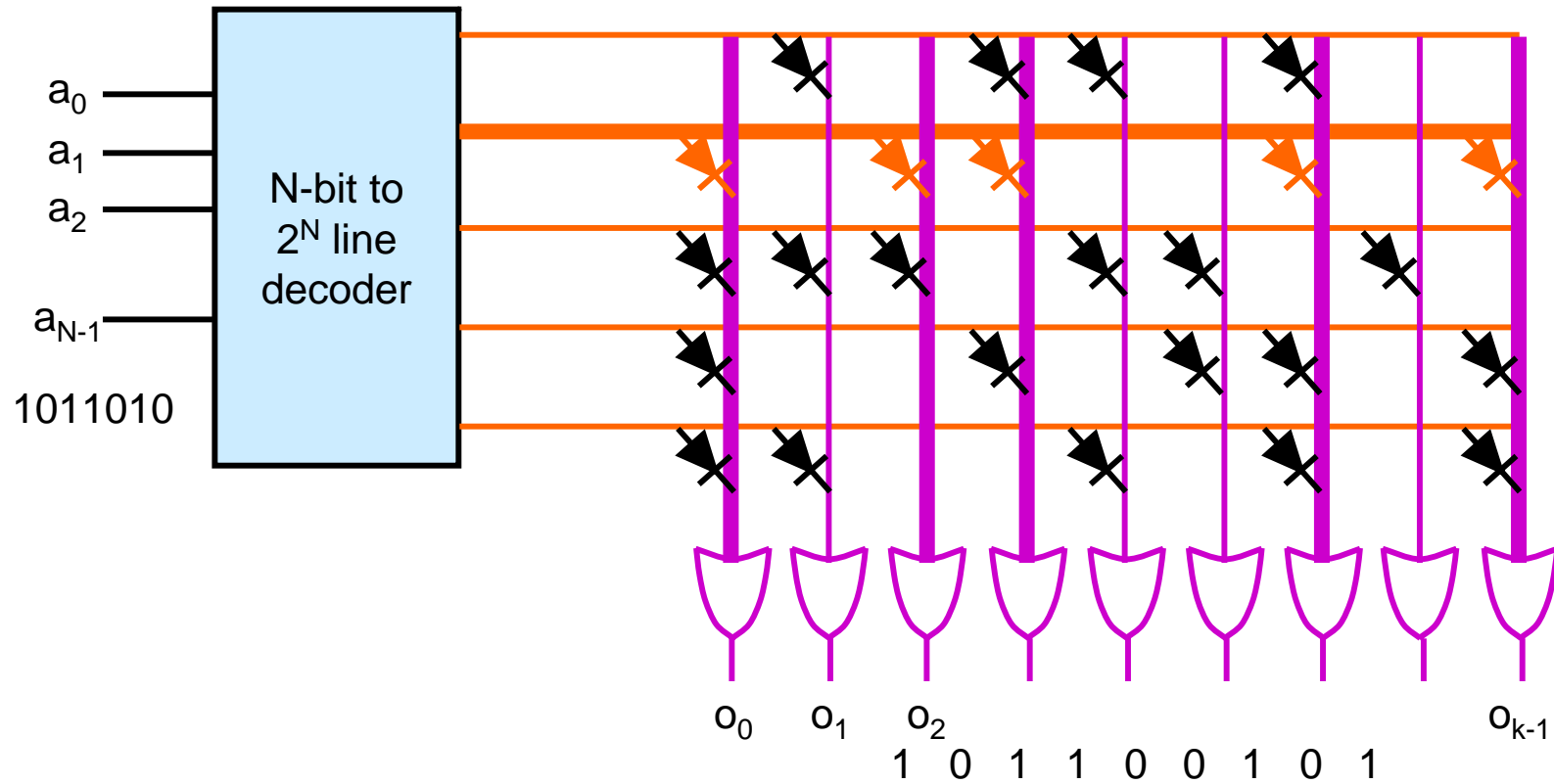
# ROM

- Programmed ROM:



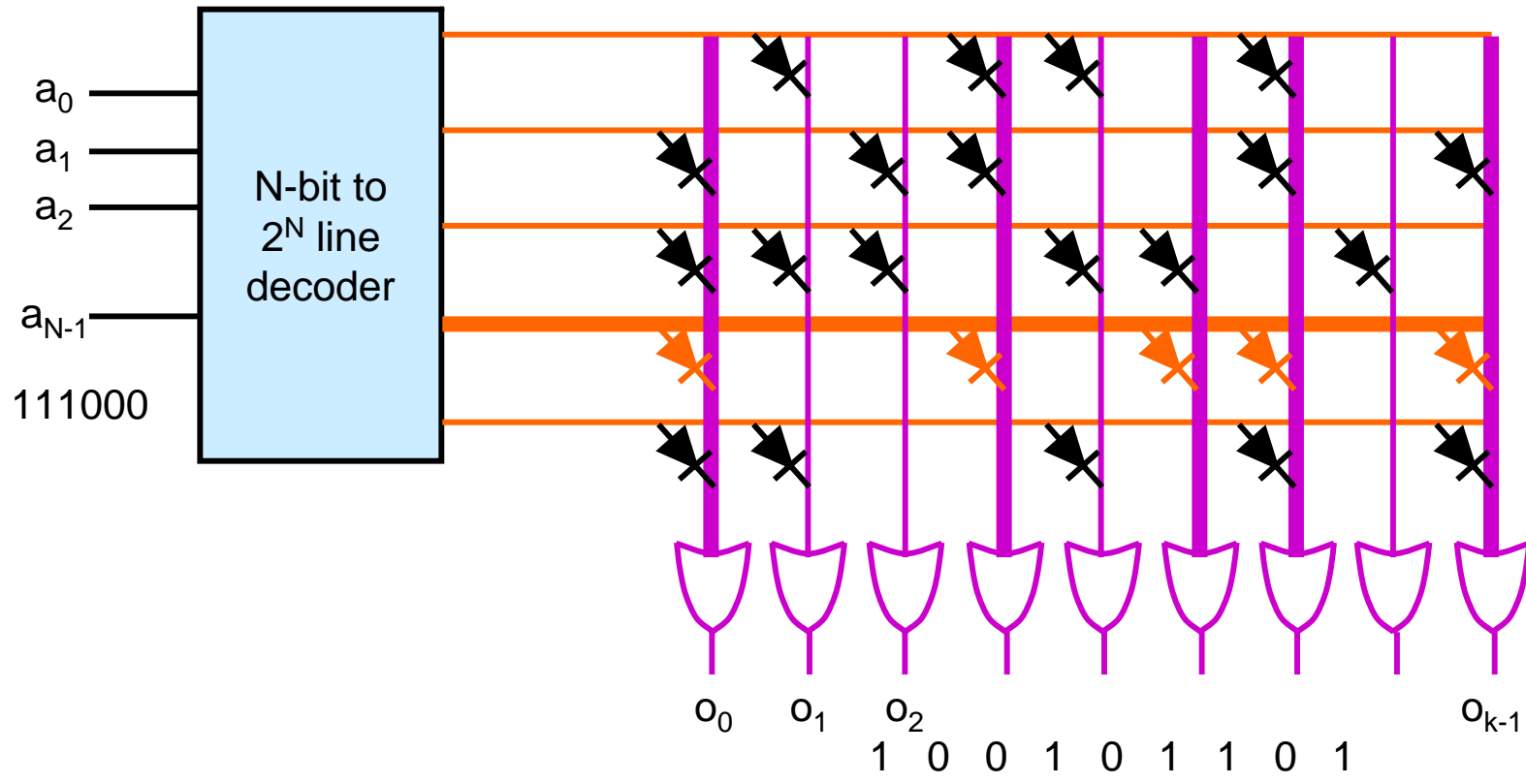
# ROM

- Programmed ROM:



# ROM

- Programmed ROM:



# ROM Variants

- Mask programmed ROM
  - ROM contents are designed before device is manufactured
- PROM
  - Device is manufactured with constant value (usually 1) stored in each cell
  - One-time programming cycle blows fusible link, changing cell (usually to 0)
- EPROM
  - Device can be programmed 100s – 1000s of times with UV erase cycle in between
- EEPROM
  - Device can be erased in circuit and reprogrammed.
  - Practically unlimited read cycles, but finite number of write cycles

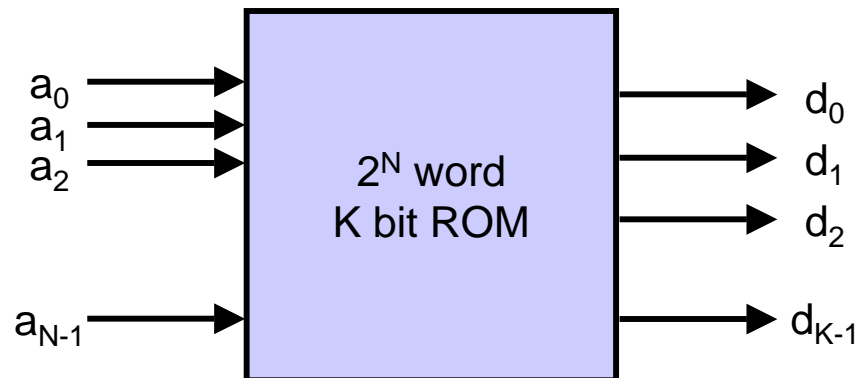
# Using ROM to Implement Combinatorial Circuits



Viewed as a ROM

Address	Data
a <sub>N-1</sub> a <sub>N-2</sub> ...a <sub>0</sub>	d <sub>K-1</sub> d <sub>K-2</sub> ...d <sub>0</sub>
00...000	D <sub>0</sub>
00...001	D <sub>1</sub>
00...010	D <sub>2</sub>
11...111	D <sub>N-1</sub>

# Using ROM to Implement Combinatorial Circuits



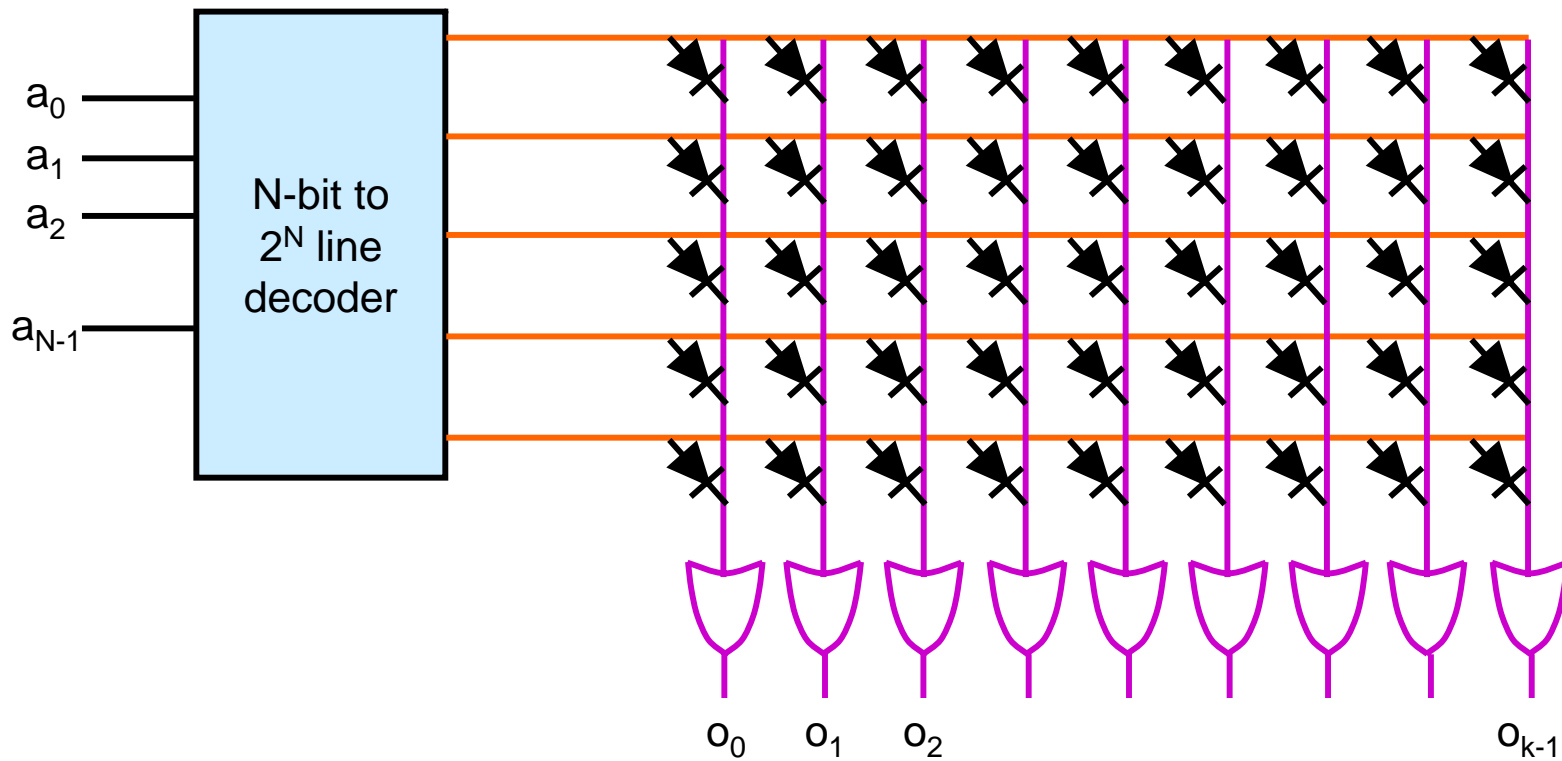
Viewed as a ROM

Address	Data
$a_{N-1}a_{N-2}\dots a_0$	$d_{K-1}d_{K-2}\dots d_0$
00...000	$D_0$
00...001	$D_1$
00...010	$D_2$
11...111	$D_{N-1}$

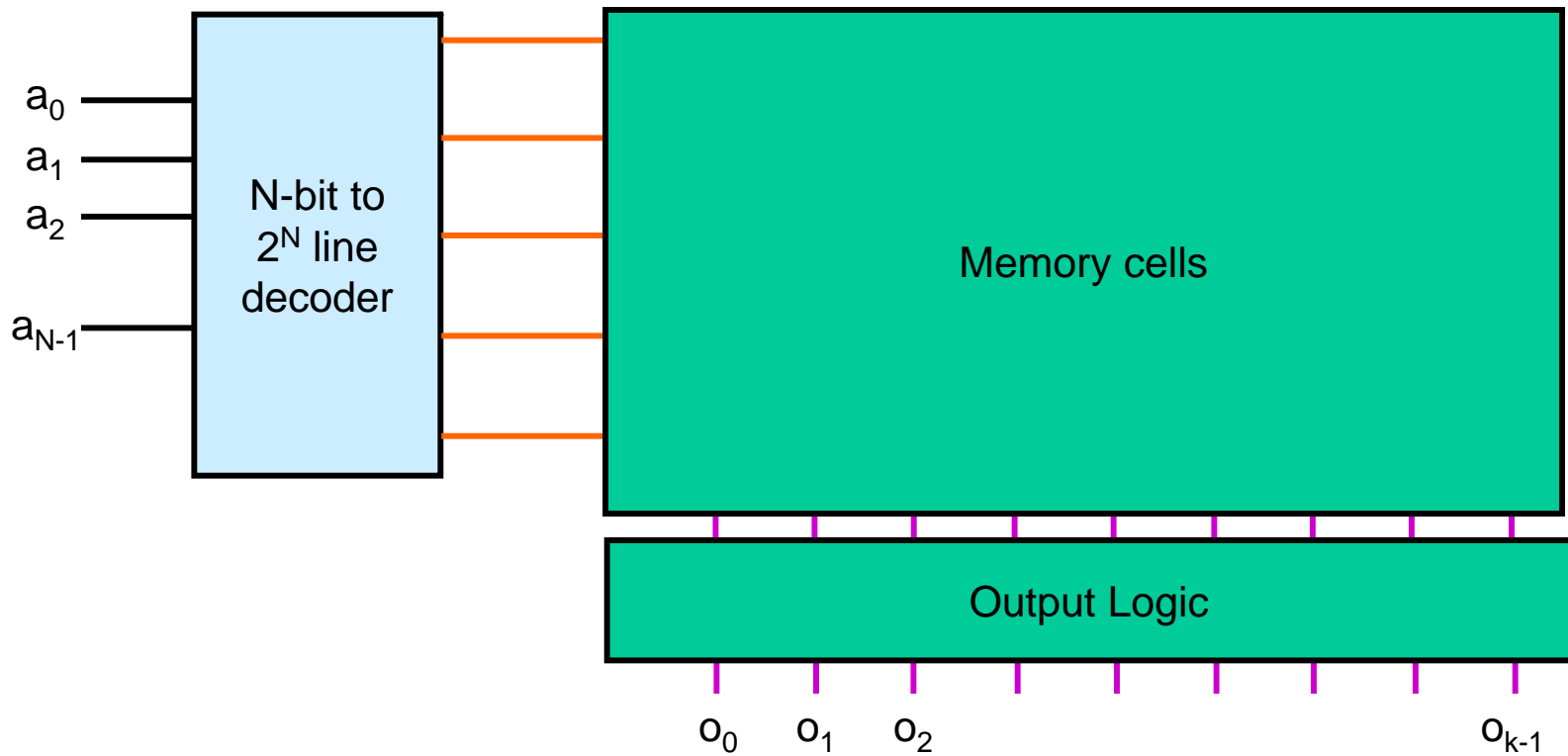
Viewed as a Truth Table

Input						Output				
$a_{N-1}$	$a_{N-2}$	.	.	.	$a_0$	$d_{K-1}$	$d_{K-2}$	.	.	$d_0$
0	0	.	.	.	0					
0	0	.	.	.	1					
1	1	.	.	.	1					

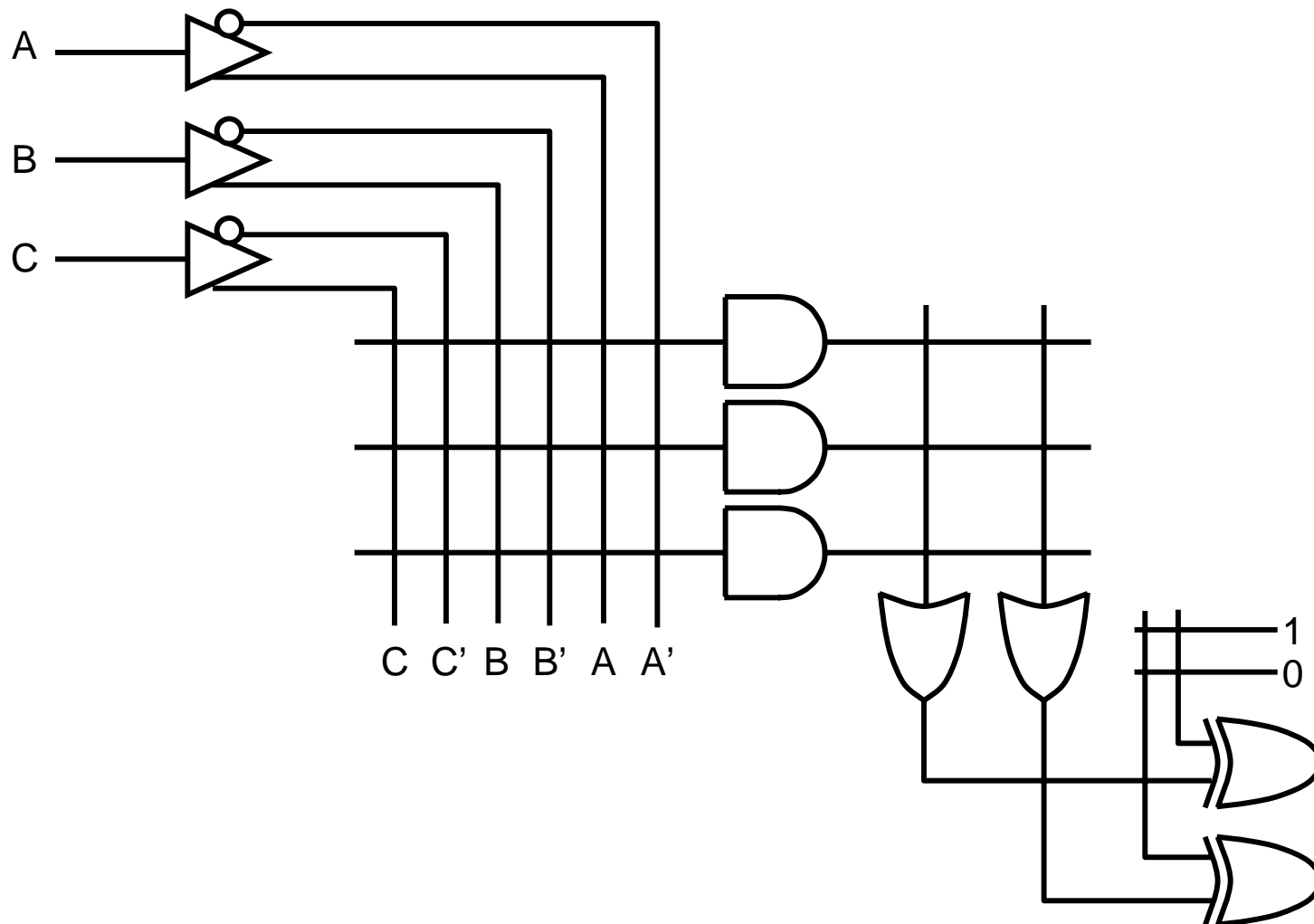
# Using ROM to Implement Combinatorial Circuits



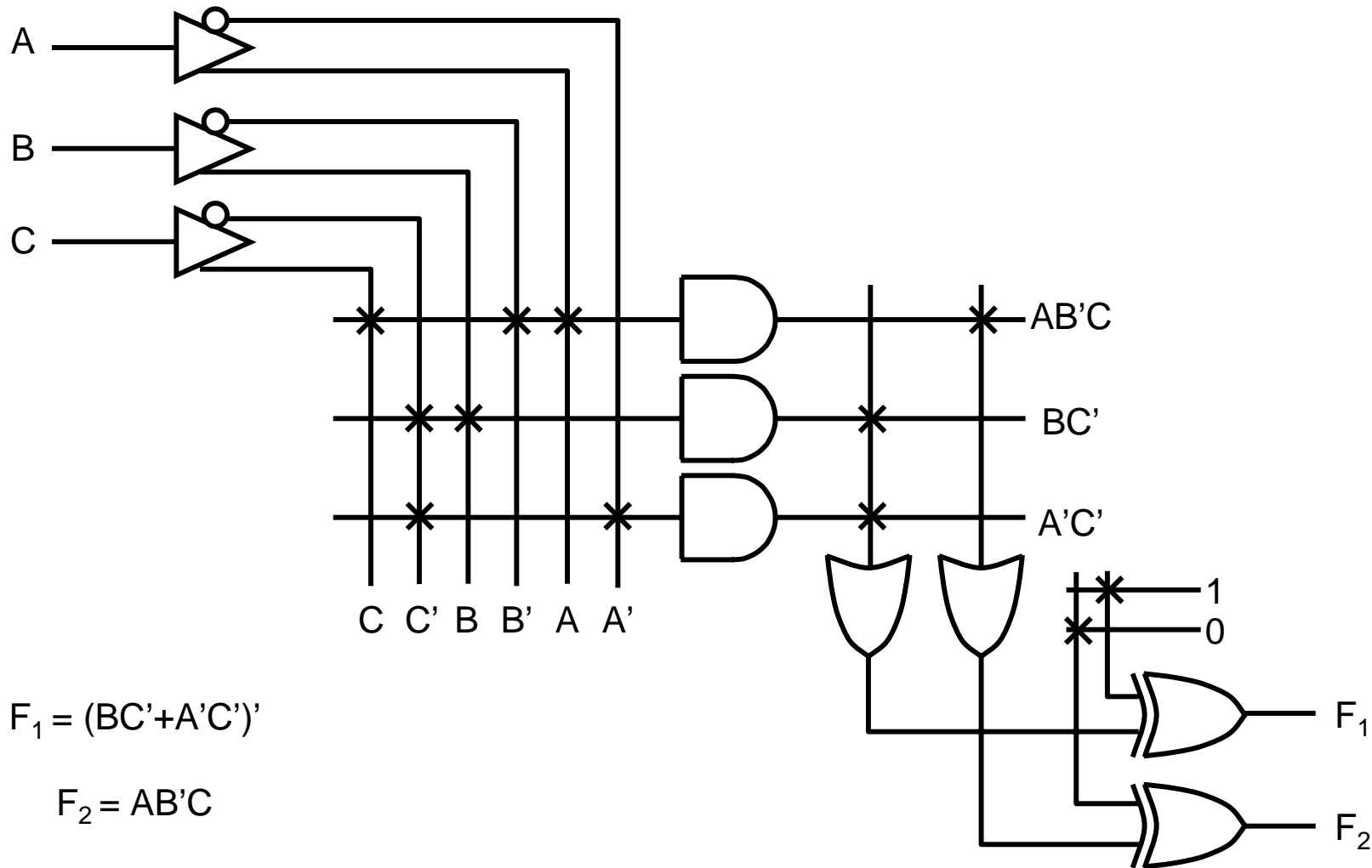
# Using ROM to Implement Combinatorial Circuits



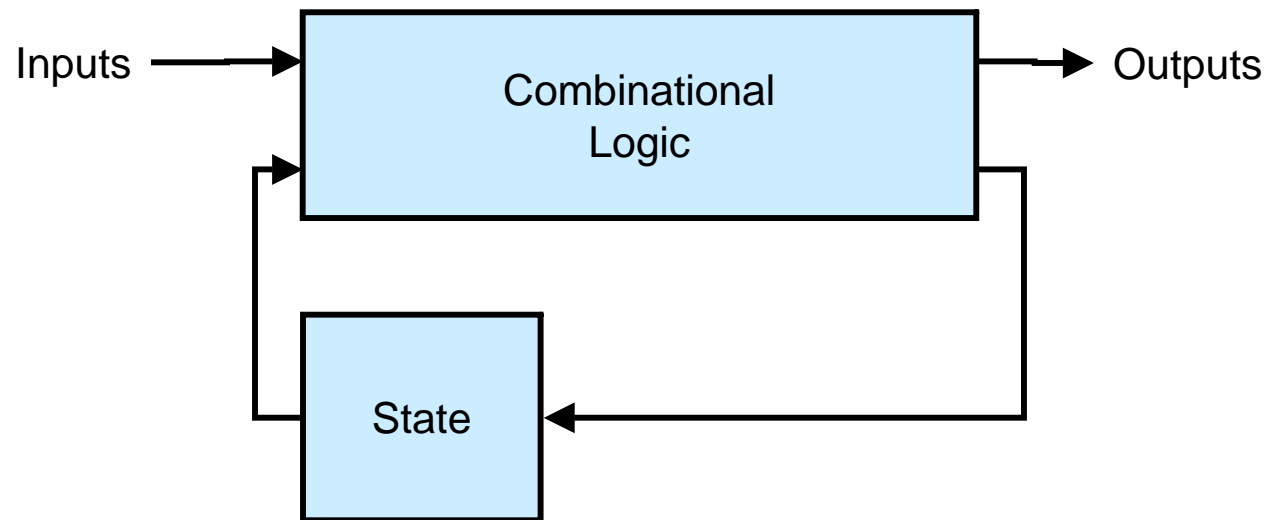
# Programmable Logic Array



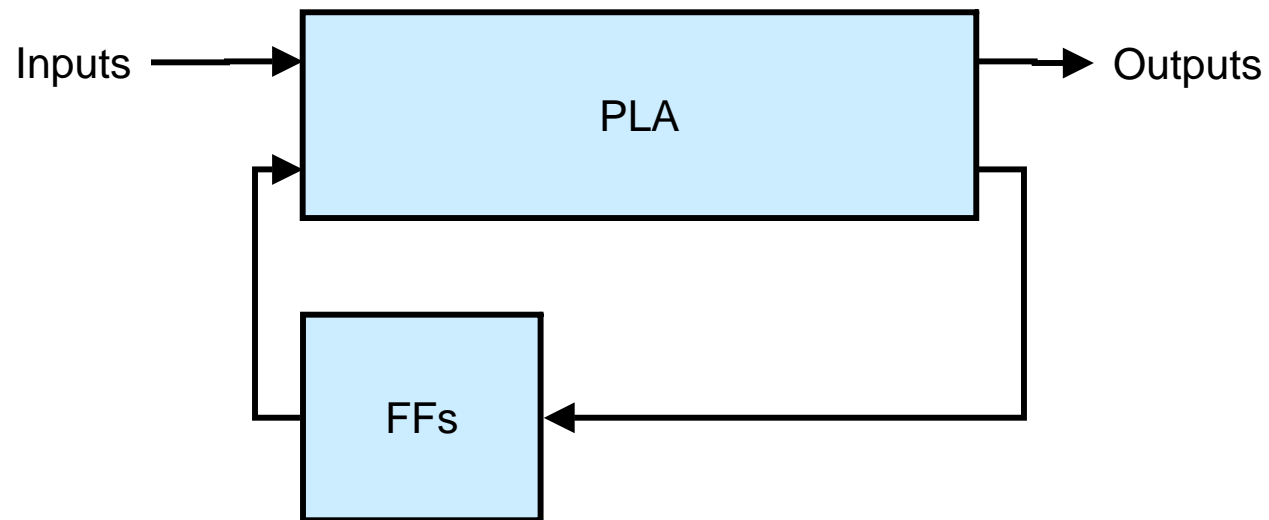
# Programmable Logic Array



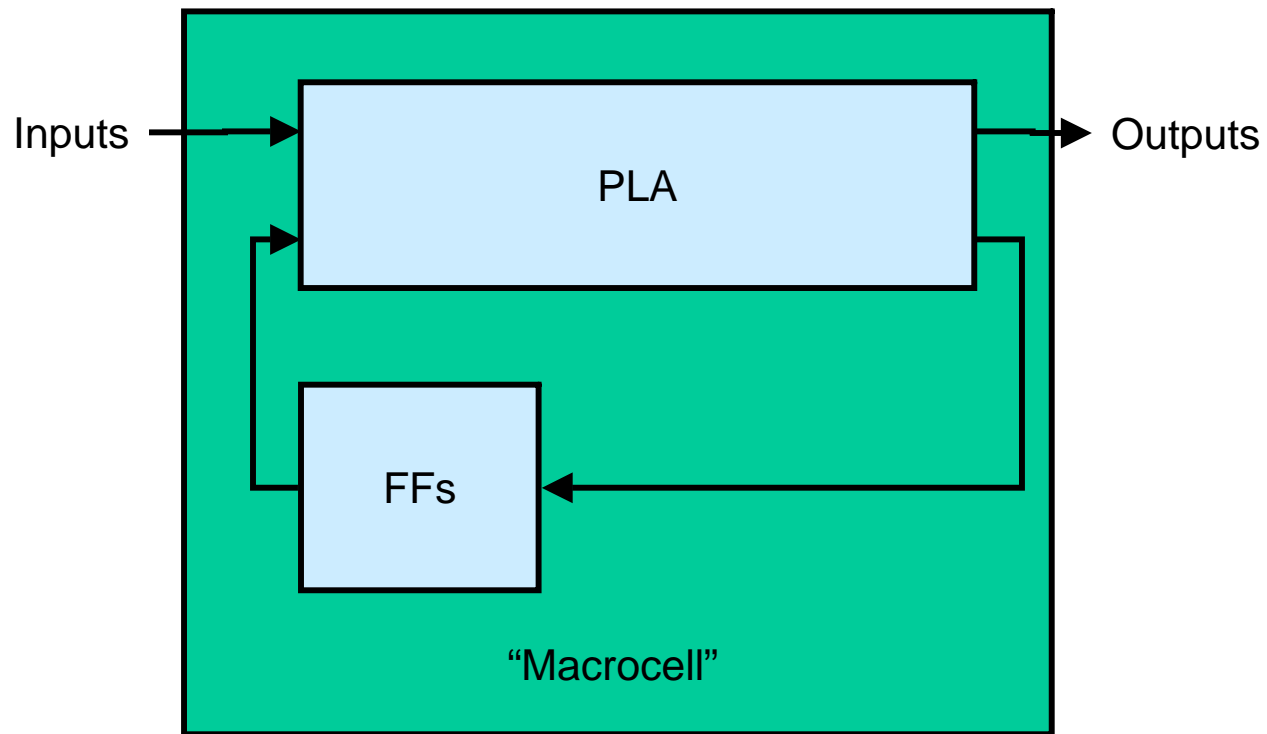
# Sequential Systems



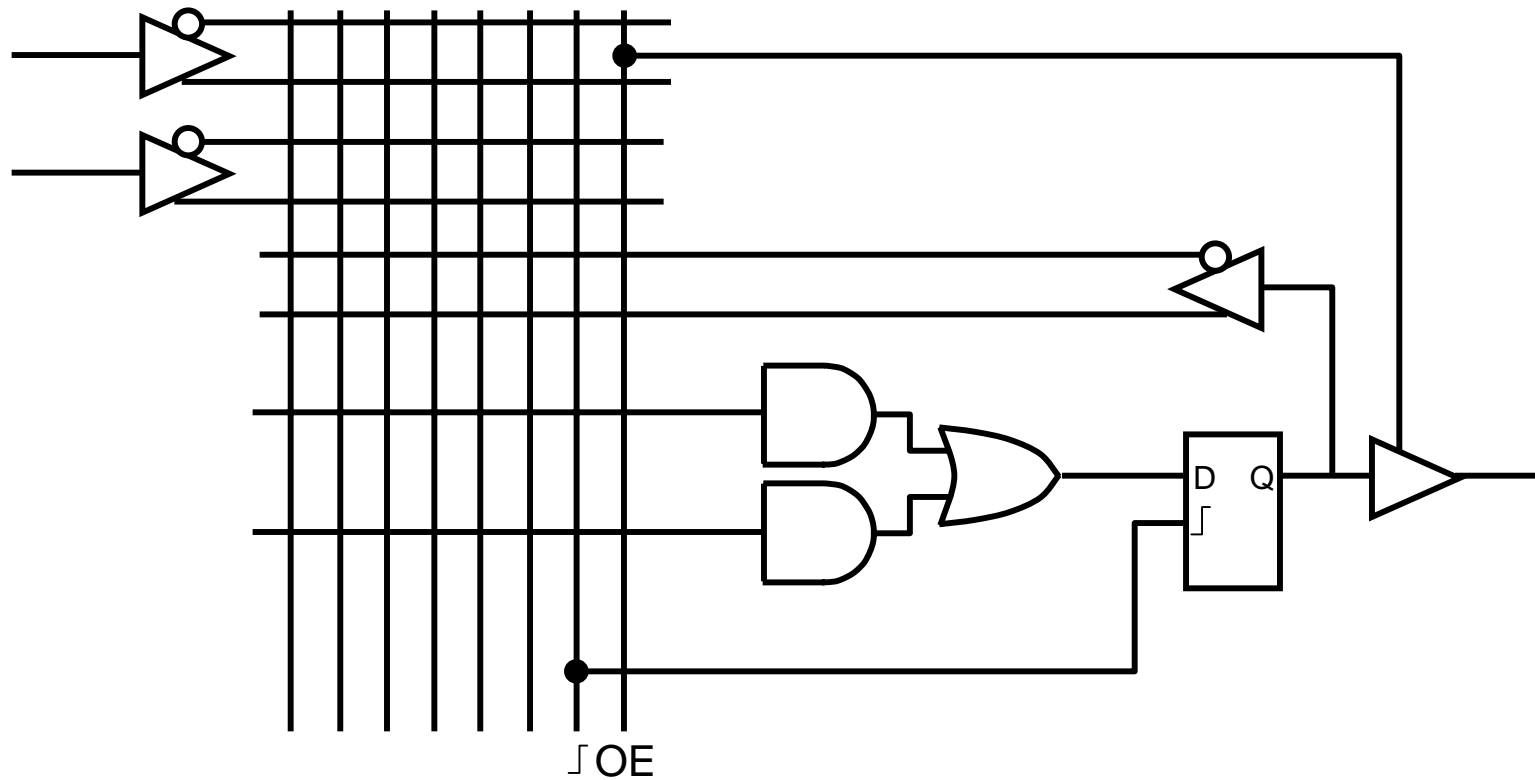
# Sequential Programmable Logic



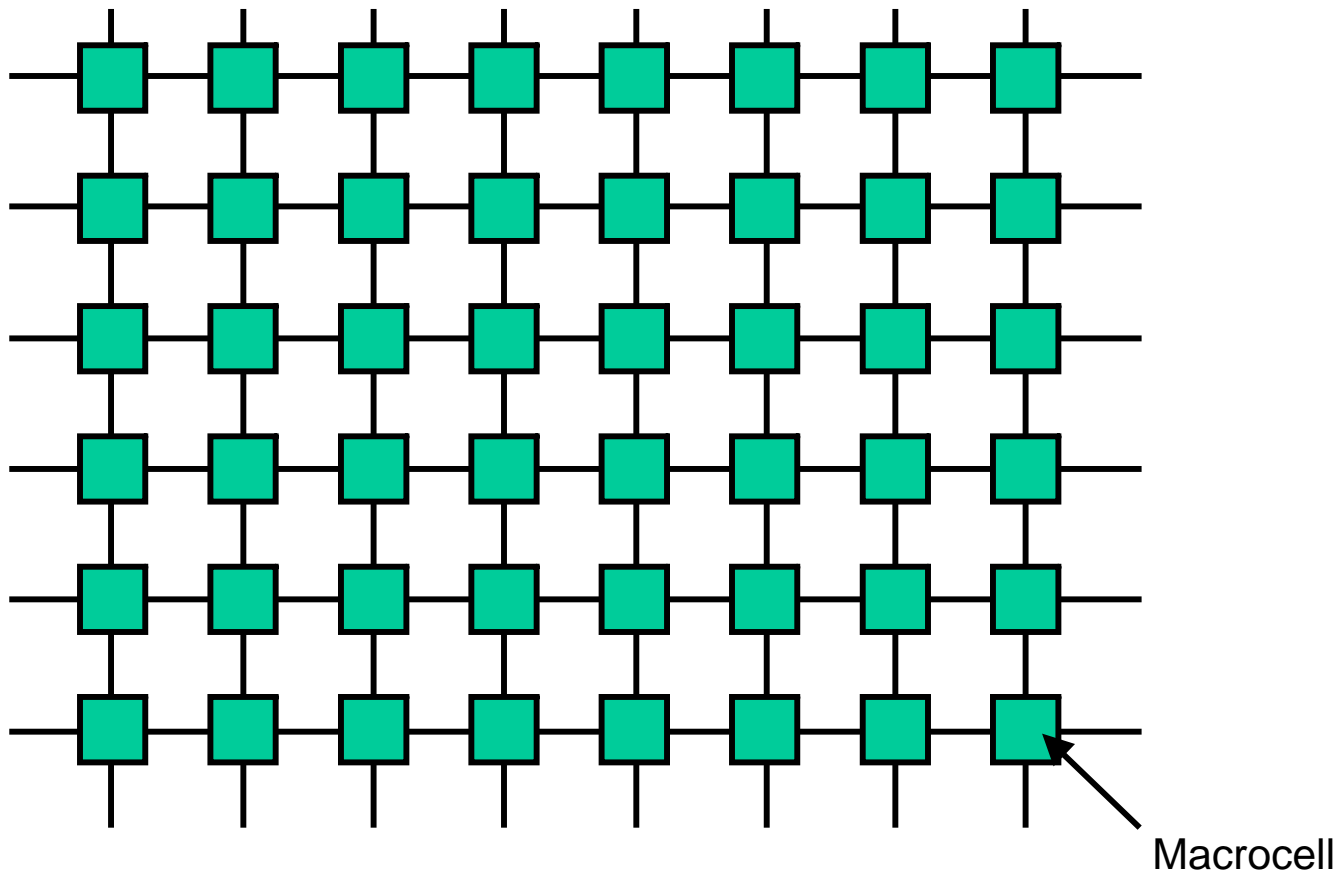
# Sequential Programmable Logic



# Generic Macrocell



# Complex Programmable Logic



# Summary

- Fundamental concepts of digital systems (Mano Chapter 1)
- Binary codes, number systems, and arithmetic (Ch 1)
- Boolean algebra (Ch 2)
- Simplification of switching equations (Ch 3)
- Digital device characteristics (e.g., TTL, CMOS)/design considerations (Ch 10)
- Combinatoric logical design including LSI implementation (Chapter 4)
- Flip-flops and state memory elements (Ch 5)
- Sequential logic analysis and design (Ch 5)
- Counters, shift register circuits (Ch 6)
- Hazards, Races, and time related issues in digital design (Ch 9)
- **Synchronous vs. asynchronous design (Ch 9)**
- **Memory and Programmable logic (Ch 7)**
- Minimization of sequential systems
- Introduction to Finite Automata

# Homework 15 – due in Class 17

- Show all work
- Problems 7-15, 7-17