Logic Design Concepts

Zainalabedin Navabi

Logic Design Concepts

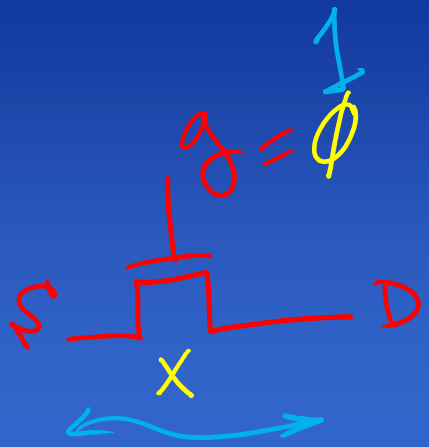
- Number Systems
- Binary Arithmetic
- Basic Logic Gates and Structures
- Designing Combinational Circuits
- Storage Elements
- Sequential Circuit Design
- Memories
- Bidirectional IO
- A Comprehensive Example: Serial Adder
- Summary

Logic Design Concepts

- Number Systems
- Binary Arithmetic
- Basic Logic Gates and Structures
- Designing Combinational Circuits
- Storage Elements
- Sequential Circuit Design
- Memories
- Bidirectional IO
- A Comprehensive Example: Serial Adder
- Summary

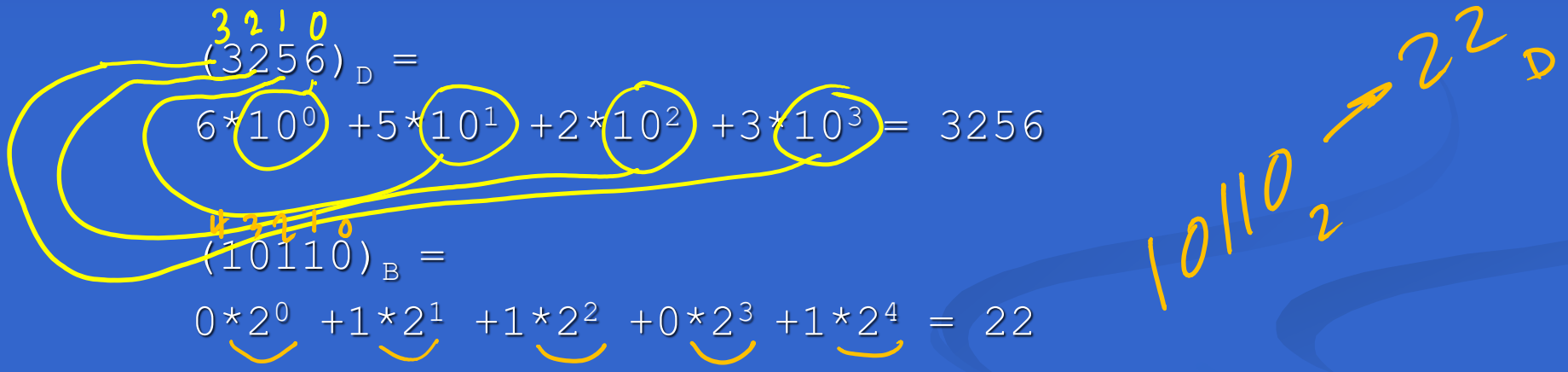
Number Systems

- Binary Numbers
- Hexadecimal Numbers



Number Systems

- Binary Numbers



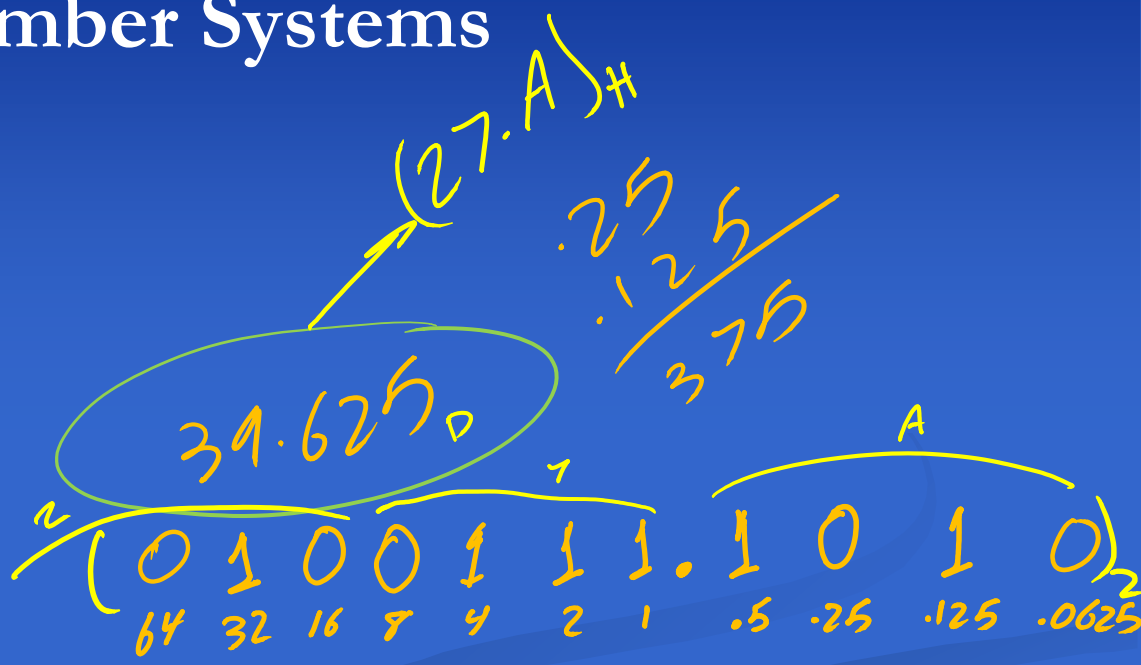
- A number in specified base is evaluated by considering the weight of each digit.

Number Systems

- Binary Numbers

$(1101.011)_B$ is $(13.375)_D$

$(19.7)_D$ is $(10011.101)_B$

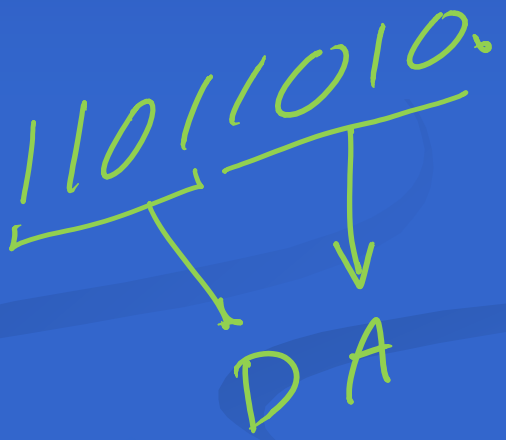


- **The weights of binary digits on the right hand side of binary point of a fractional binary number are 2-1, 2-2, 2-3, … .**

Number Systems

- Hexadecimal Numbers

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111



- **Hexadecimal Digits and Their Equivalent Decimal and Binary Representations**

Number Systems

- Hexadecimal Numbers

$(10011.101)_B = (13.A)_H$

$(11101100)_B = (EC)_H$

  - **Conversion from (to) a binary numbers to its (from) hexadecimal number are straightforward processes.**

$(A12.3C)_H$

$(1010\,0001\,0010.0011\,1100)_B$

Logic Design Concepts

- Number Systems
- Binary Arithmetic
- Basic Logic Gates and Structures
- Designing Combinational Circuits
- Storage Elements
- Sequential Circuit Design
- Memories
- Bidirectional IO
- A Comprehensive Example: Serial Adder
- Summary

Binary Arithmetic

- Signed Numbers
- Binary Addition
- Binary Subtraction
- Two's Complement System
- Overflow
- Fixed Point Numbers
- Floating Point Numbers

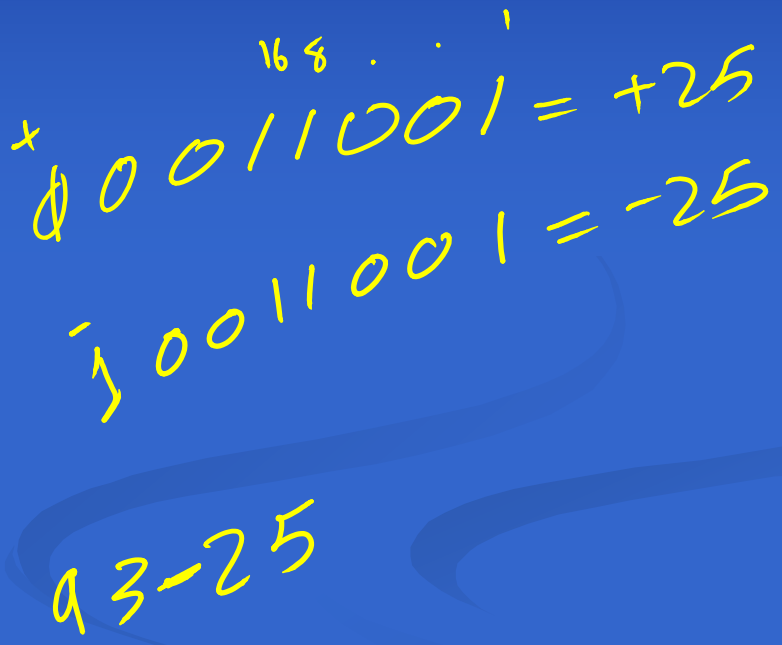
Binary Arithmetic

- Signed Numbers

(S&M)

+25 is **00011001**

-25 is **10011001**



- **Left most bit of the number represents its sign.**

Binary Arithmetic

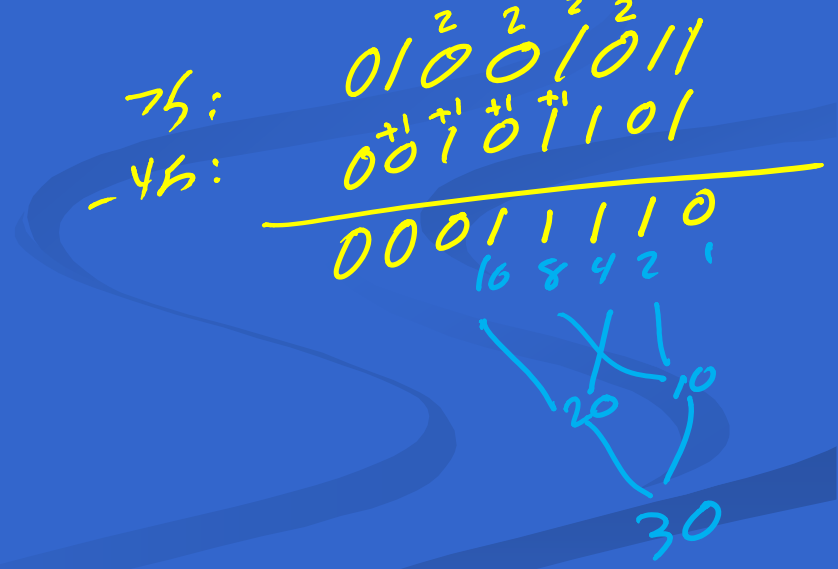
- Binary Addition

Carry Bits:	0	0	0	1	1	1	1	
	128	64	32	16	8	4	2	1
A:	0	1	0	0	1	0	1	1
B:	0	0	1	0	1	1	0	1
Result:	0	1	1	1	1	0	0	0

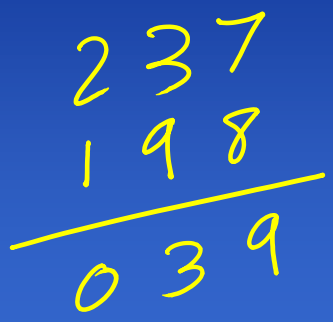
- **The carry generated in each order is carried over to the next higher order slice.**

Binary Arithmetic

- Binary Subtraction
  - Use Borrows
  - A Different Process from Binary Addition
  - A different hardware is needed

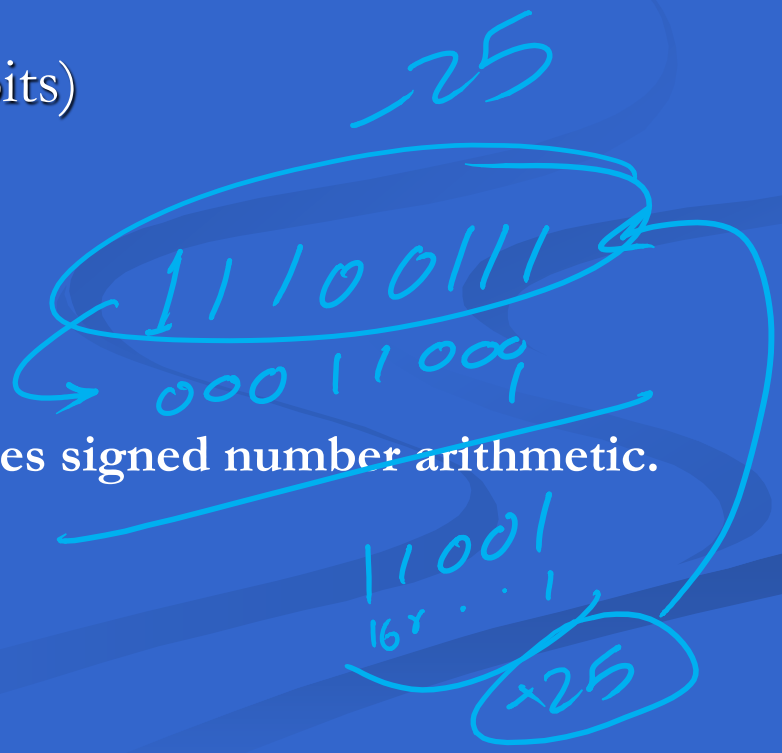


Binary Arithmetic



- Two's Complement System

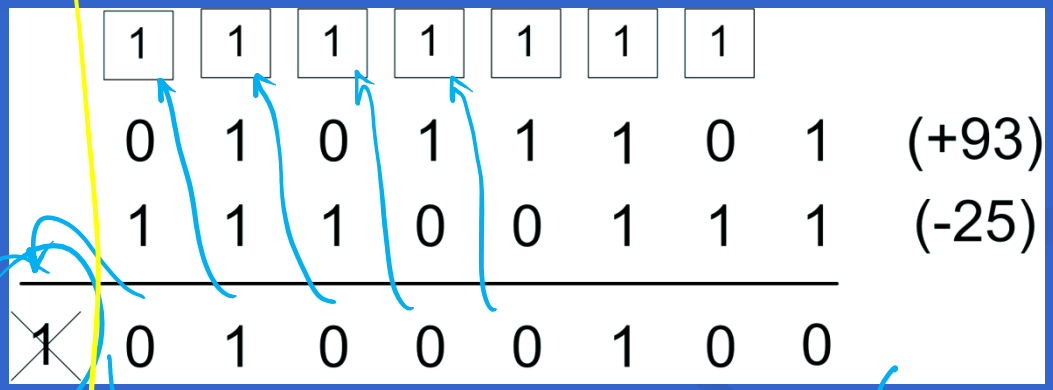
00011001	(=25)
11100110	(complementing all bits)
00000001	(adding a **1**)
11100111	(-25)



- **Two's complement system simplifies signed number arithmetic.**

Binary Arithmetic

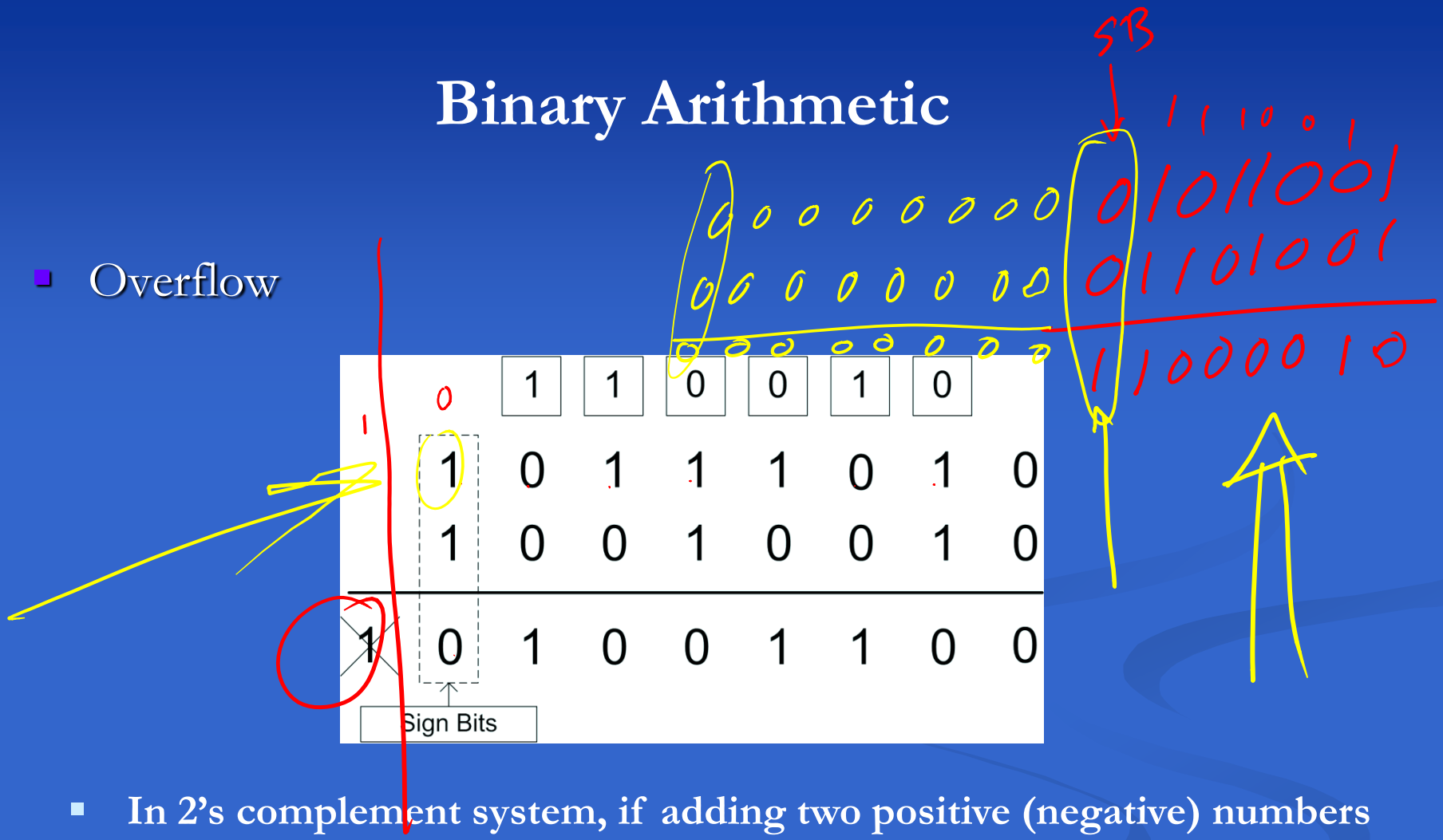
- Two's Complement System



- **In subtraction, A-B is done by A+(-B), in which –B is two's complement of B.**

Binary Arithmetic

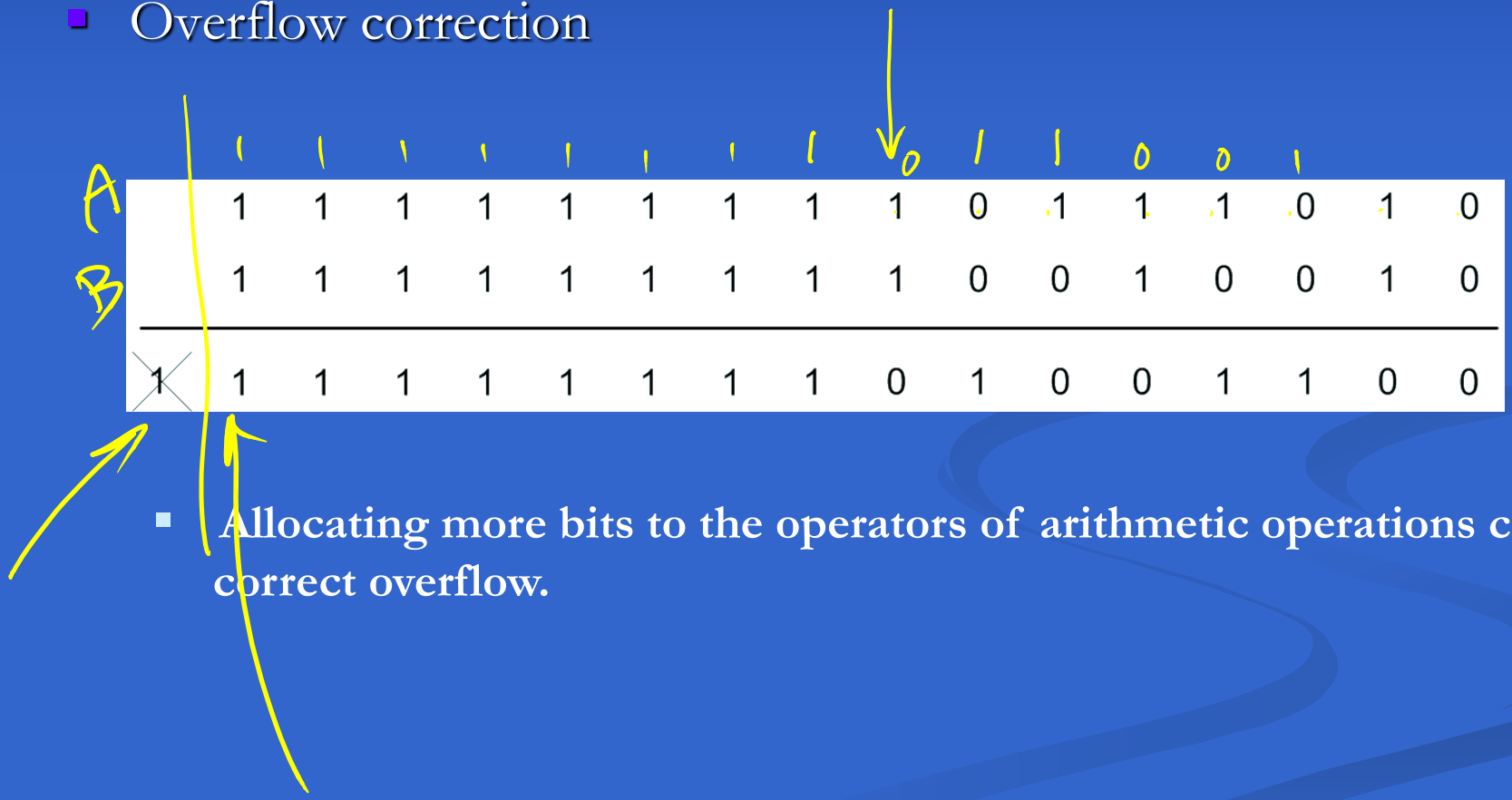
- Overflow



- In 2's complement system, if adding two positive (negative) numbers results in negative (positive) number, an overflow occurs.

Binary Arithmetic

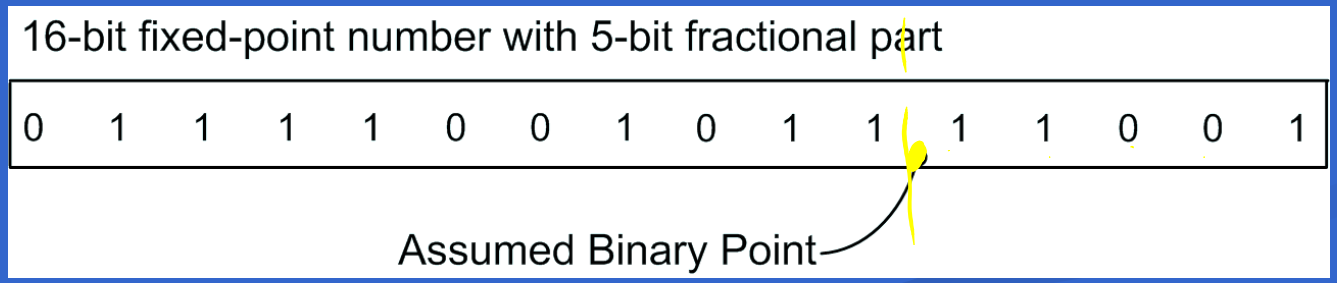
- Overflow correction



  - **Allocating more bits to the operators of arithmetic operations can correct overflow.**

Binary Arithmetic

- Fixed Point Numbers

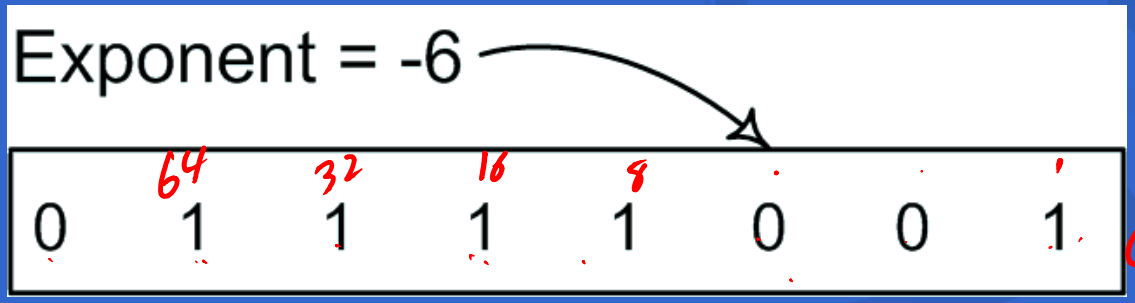


- **A fixed-point number has integer and fractional parts.**

Binary Arithmetic

- Floating Point Numbers

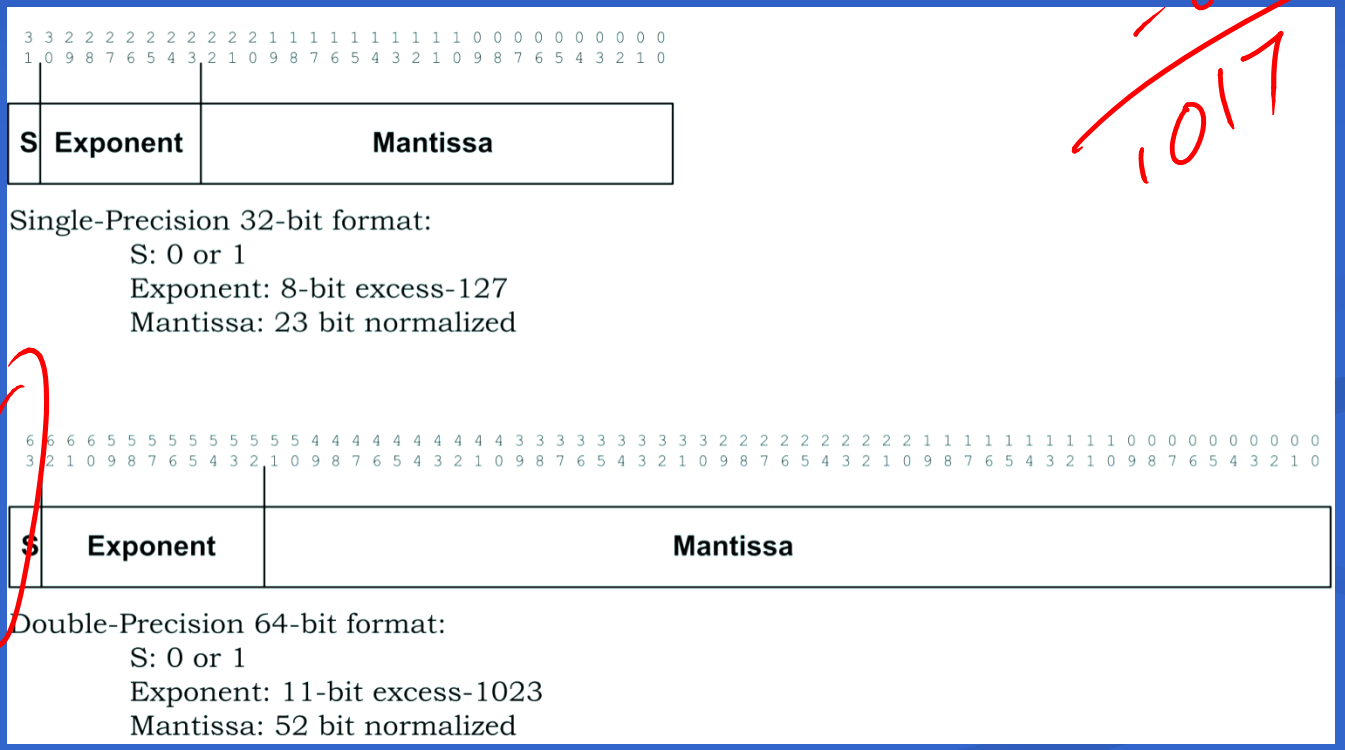
Sign Mantissa × *Radix*^{Exponent}



- Exponent is signed number plus $2^{n-1}-1$
- Here it is excess 127, that becomes 127-6, which is 121

Binary Arithmetic

- Floating Point Numbers



- **A Single-Precision 32-Bit Format and a Double-Precision 64-Bit Format**

Logic Design Concepts

- Number Systems
- Binary Arithmetic
- Basic Logic Gates and Structures
- Designing Combinational Circuits
- Storage Elements
- Sequential Circuit Design
- Memories
- Bidirectional IO
- A Comprehensive Example: Serial Adder
- Summary

Basic Logic Gates and Structures

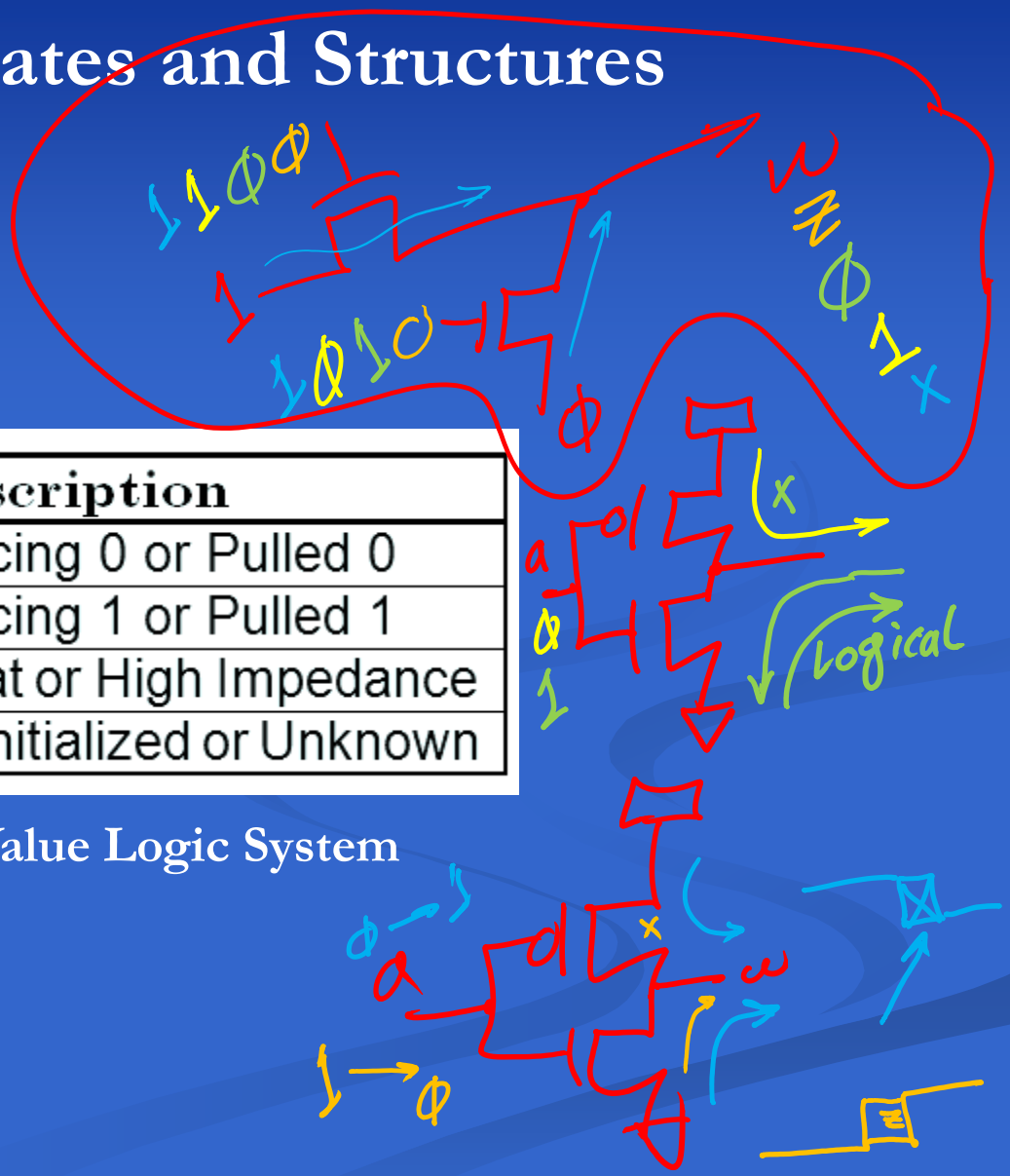
- Logic Value System
- Logic Function Representation
- Inverter
- NAND and NOR gates
- AND and OR gates
- XOR gate
- MUX gate
- Three-State Gates
- Look-up Tables (LUT)
- Majority Function

Basic Logic Gates and Structures

- Logic Value System

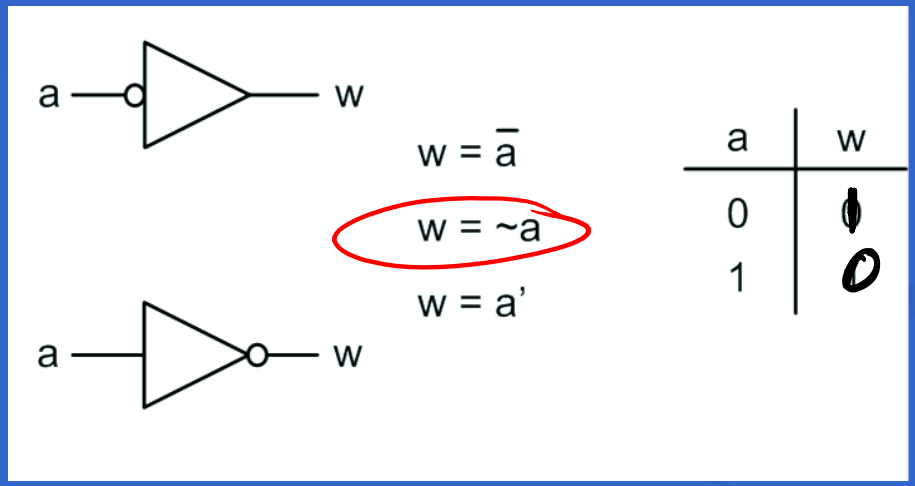
Value	Description
0	Forcing 0 or Pulled 0
1	Forcing 1 or Pulled 1
Z	Float or High Impedance
X	Uninitialized or Unknown

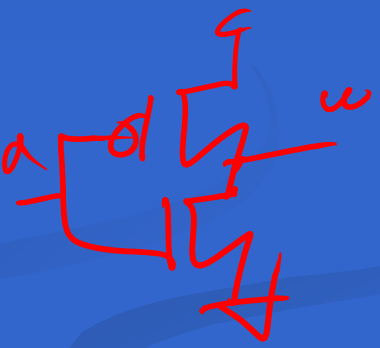
- **Four-Value Logic System**



Basic Logic Gates and Structures

- Inverter Gate

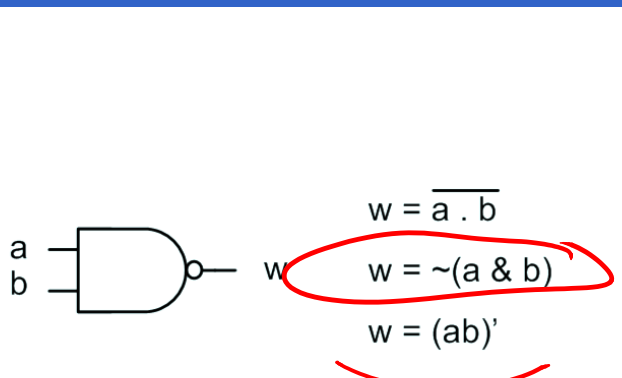




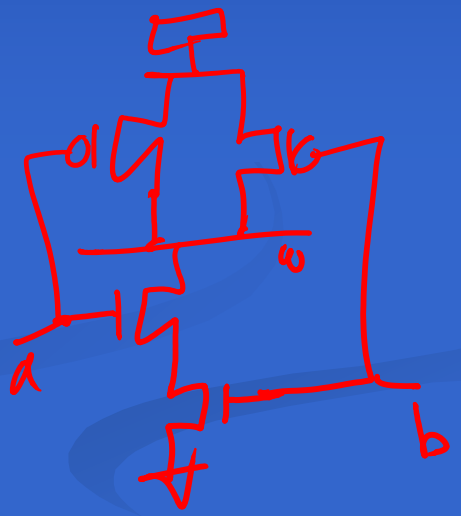
- **Inverter Gate, Its Logic Symbol, Its Algebraic Notations, and Its Truth Table**

Basic Logic Gates and Structures

- NAND Gate



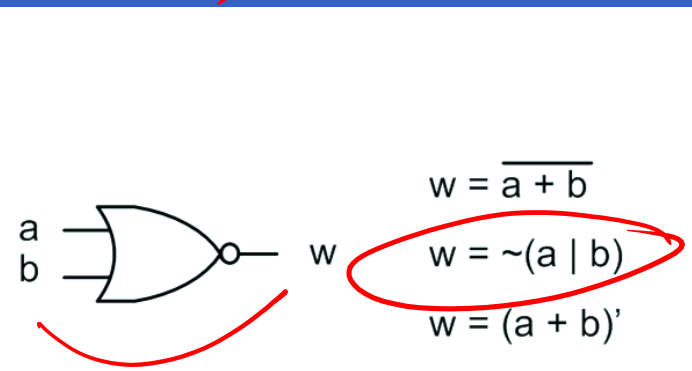
a	b	w
0	0	1
0	1	1
1	0	1
1	1	0



- **NAND Gate, Its Logic Symbol, Its Algebraic Notations, and Its Truth Table**

Basic Logic Gates and Structures

- NOR Gate

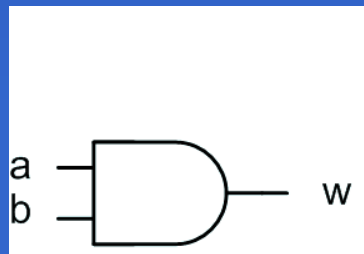


a	b	w
0	0	1
0	1	0
1	0	0
1	1	0

- **NOR Gate, Its Logic Symbol, Its Algebraic Notations, and Its Truth Table**

Basic Logic Gates and Structures

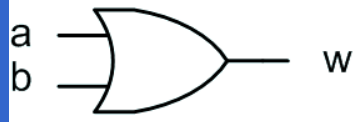
- AND and OR Gates



$w = a \cdot b$

$w = a \,\&\, b$

a	b	w
0	0	0
0	1	0
1	0	0
1	1	1



$w = a + b$

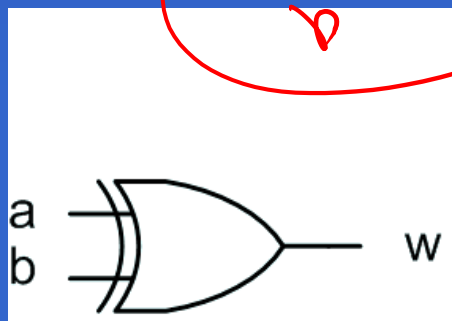
$w = a \mid b$

a	b	w
0	0	0
0	1	1
1	0	1
1	1	1

- **AND and OR Gate, Their Logic Symbol, Their Algebraic Notations, and Their Truth Table**

Basic Logic Gates and Structures

- XOR Gate



$w = a \oplus b$

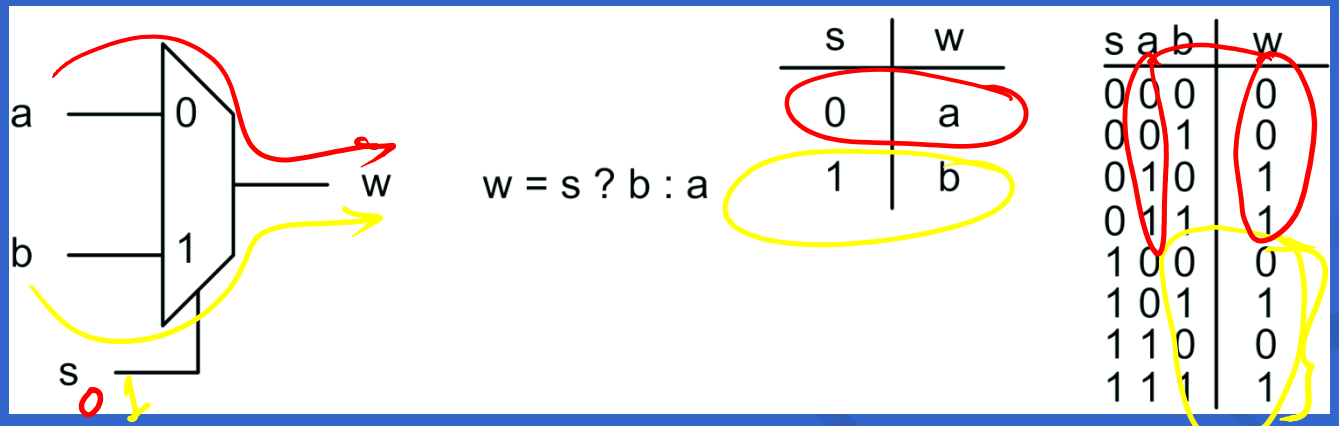
$w = a \wedge b$

a	b	w
0	0	0
0	1	1
1	0	1
1	1	0

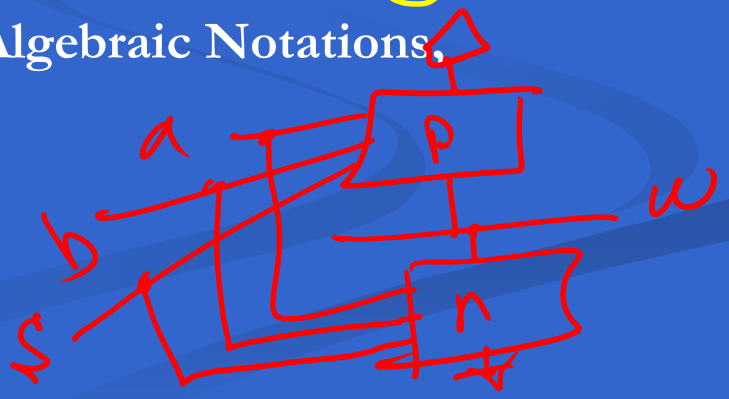
- **XOR Gate, Its Logic Symbol, Its Algebraic Notations, and Its Truth Table**

Basic Logic Gates and Structures

- MUX Gate

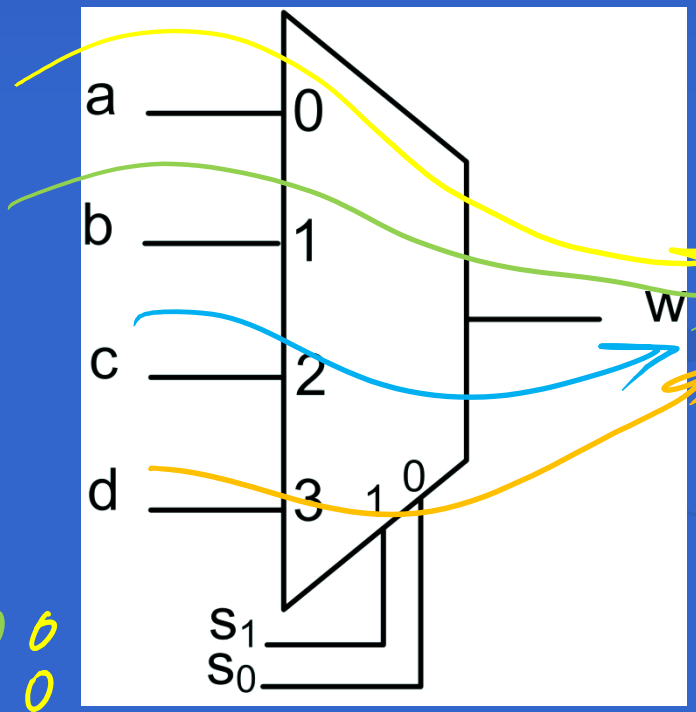


- **MUX Gate, Its Logic Symbol, Its Algebraic Notations, and Its Truth Table**



Basic Logic Gates and Structures

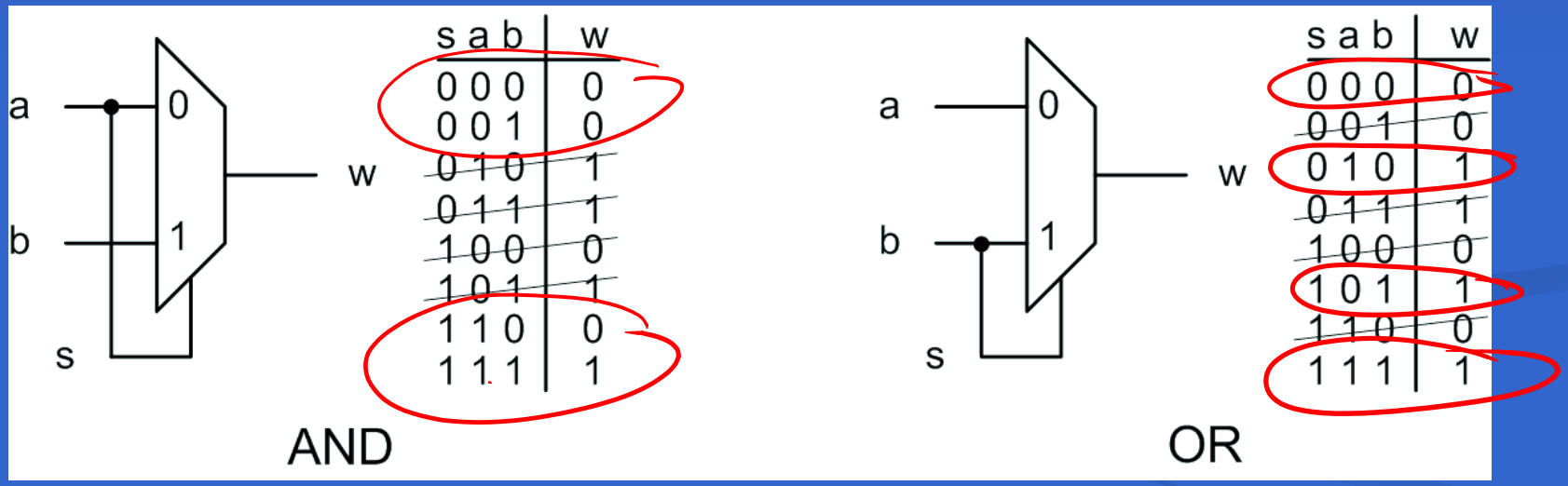
- MUX Gate – four to one



- A multiplexer can be used as a switch that selects one of its inputs depending on the binary value of its select input.

Basic Logic Gates and Structures

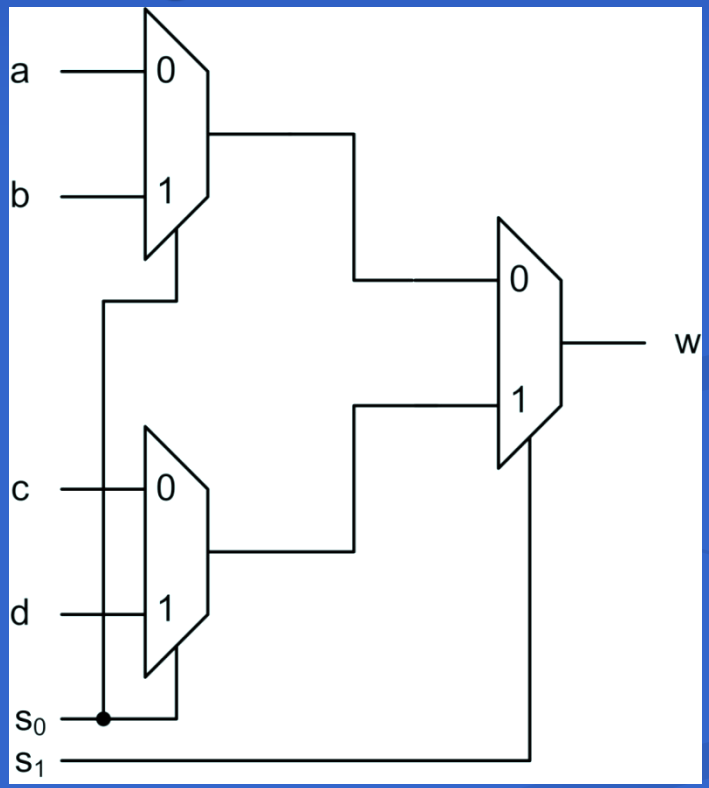
- MUX Gate – other logic gates



- **Multiplexer Configurations**

Basic Logic Gates and Structures

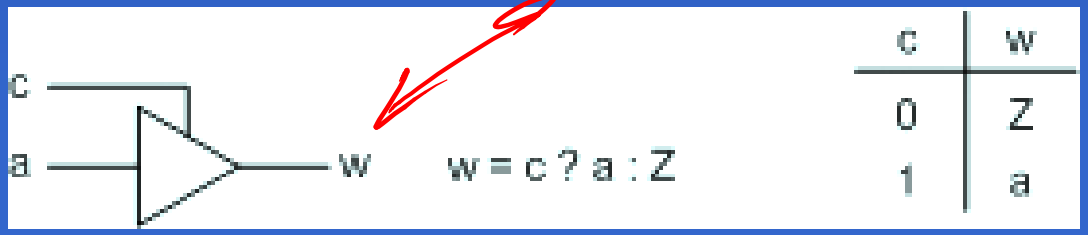
- MUX Gate - cascading



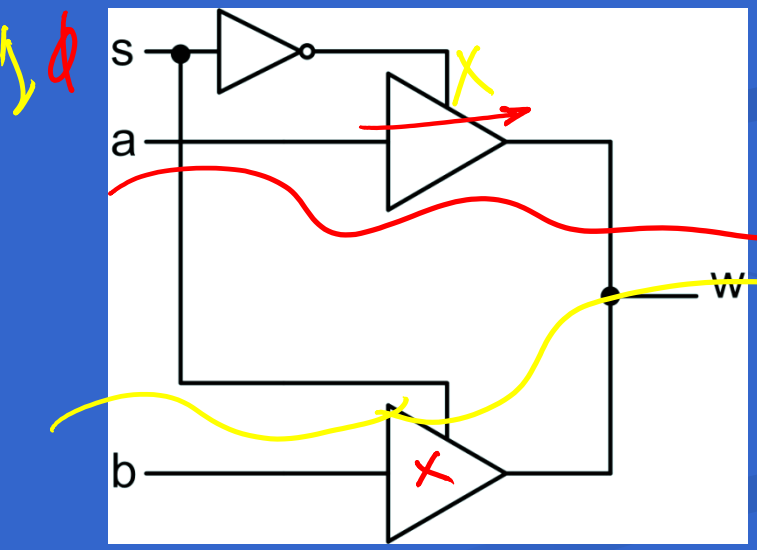
- **A 4-to-1 Multiplexer Built out of Three 2-to-1 Multiplexers**

Basic Logic Gates and Structures

- Three-State Gates



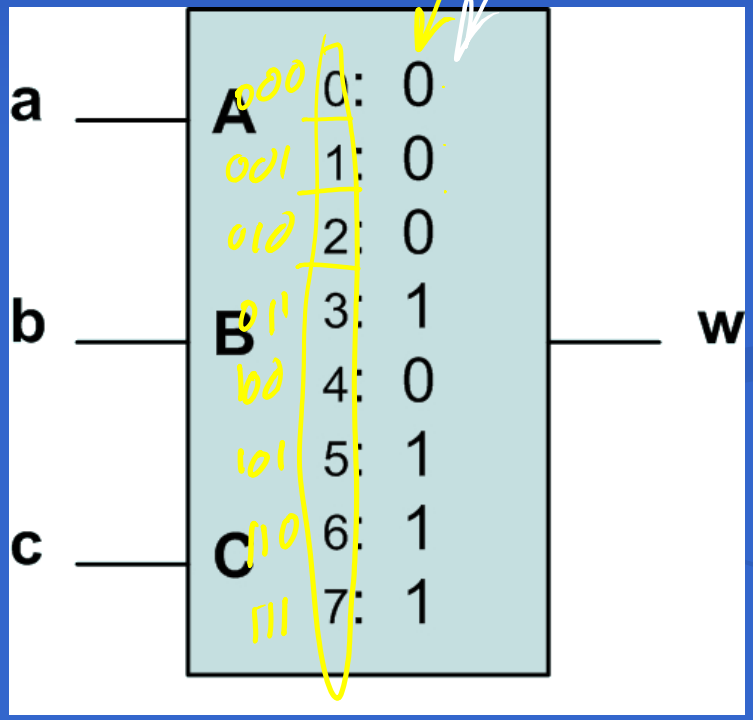
c	w
0	Z
1	a

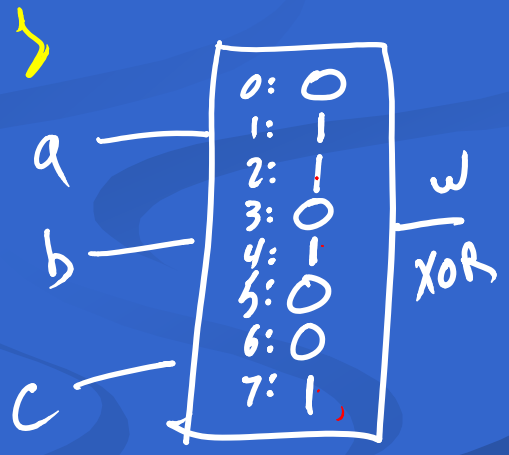


- **Building a Multiplexer with Tri-state Gates**

Basic Logic Gates and Structures

- Look-up Tables (LUT)





- Implementation of Majority Circuit using LUT

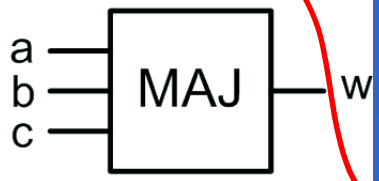
Basic Logic Gates and Structures

- Logic Function Representation

	a	b	c	w
0:	0	0	0	0
1:	0	0	1	0
2:	0	1	0	0
3:	0	1	1	1
4:	1	0	0	0
5:	1	0	1	1
6:	1	1	0	1
7:	1	1	1	1

$w = MAJ(a, b, c)$

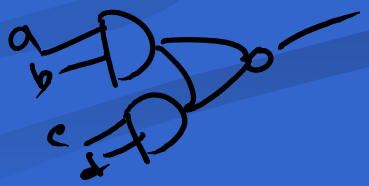
$w = \Sigma_m(3, 5, 6, 7)$



- **Example of Logic Function with Three Inputs and One Output**

AOI

$w = \overline{(a \cdot b) + (c \cdot d)}$



Logic Design Concepts

- Number Systems
- Binary Arithmetic
- Basic Logic Gates and Structures
- Designing Combinational Circuits
- Storage Elements
- Sequential Circuit Design
- Memories
- Bidirectional IO
- A Comprehensive Example: Serial Adder
- Summary

Designing Combinational Circuits

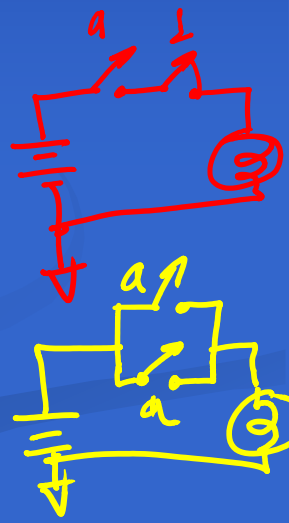
- Boolean Algebra
- Karnaugh Maps
- Don't Care Values
- Minimal Coverage
- Iterative Hardware
- Multiplexers and Decoders
- Activity Levels
- Enable / Disable Inputs
- A High-Level Design

Designing Combinational Circuits

Two Values Variables: 0 1
Operators: ' · +

- Boolean Algebra

Involution Law	• $(a')' = a$
Identity Laws	• $a + 0 = a$ • $a \cdot 1 = a$
Null Elements	• $a + 1 = 1$ • $a \cdot 0 = 0$
Idempotent Laws	• $a + a = a$ • $a \cdot a = a$



Designing Combinational Circuits

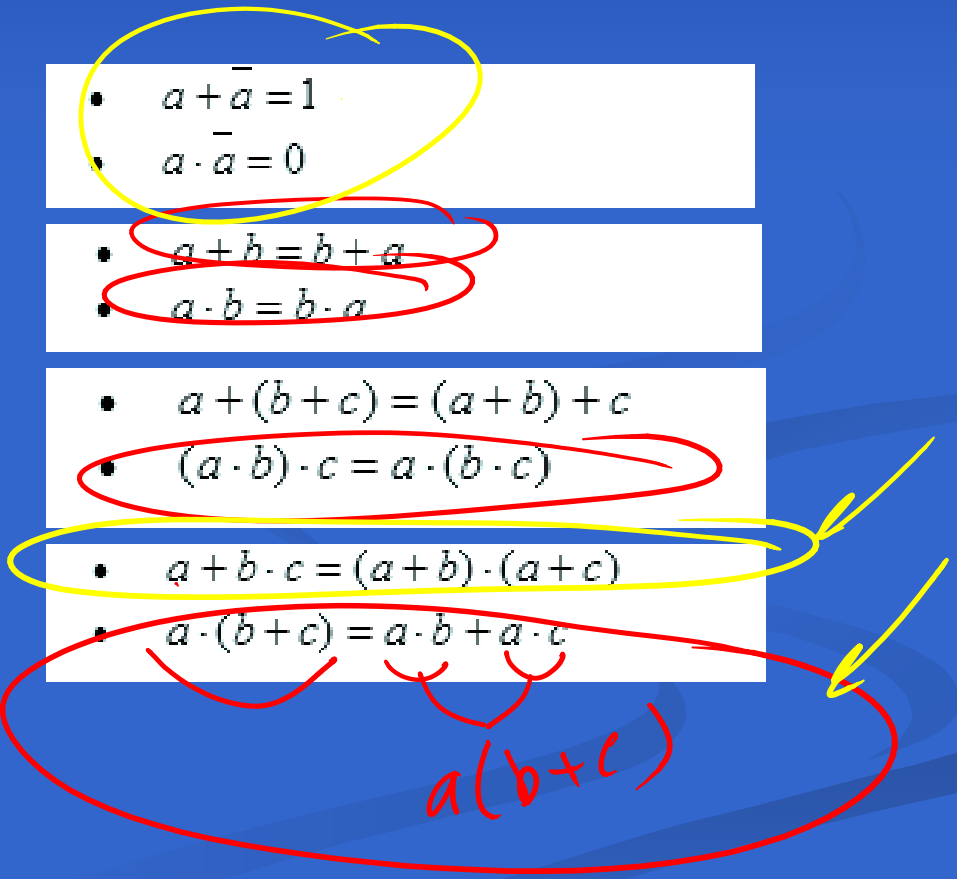
- Boolean Algebra

Complementary Laws

Commutative Laws

Associative Laws

Distributive Laws



Designing Combinational Circuits

- Boolean Algebra

$a + a \cdot b = a$

$a \cdot (a + b) = a$

Absorption Laws

$a + a \cdot b = a$

$a \cdot (a + b) = a$

Extended Absorption Laws

$a + \bar{a} \cdot b = a + b$

$a \cdot (\bar{a} + b) = a \cdot b$

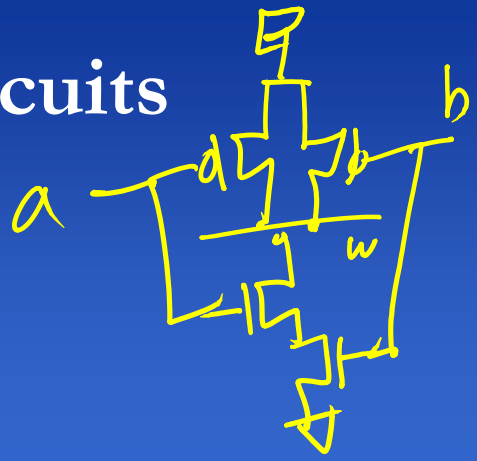
Designing Combinational Circuits

- Boolean Algebra
  - Duality
    - If E is true, changing AND (.) to OR (+), OR (+) to AND(.), 1 to 0, and 0 to 1 results in E_D that is also true.

DeMorgan's Law

- $\overline{a \cdot b} = \bar{a} + \bar{b}$
- $\overline{a + b} = \bar{a} \cdot \bar{b}$

Designing Combinational Circuits

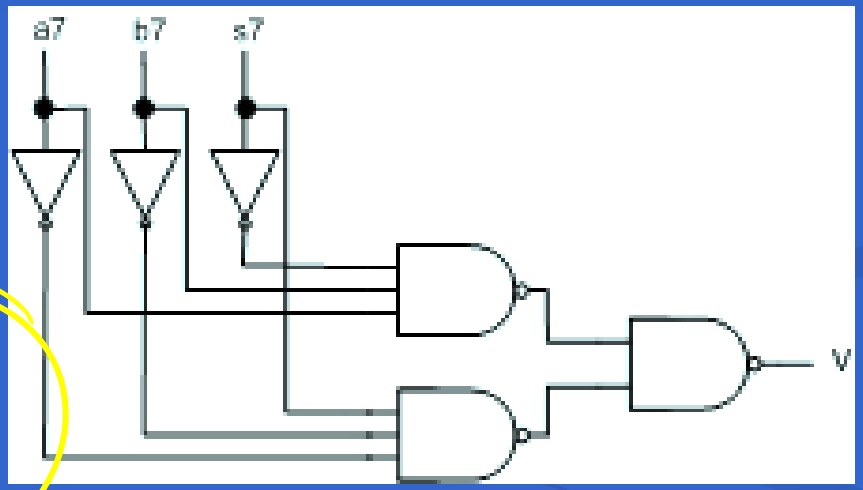


- Boolean Algebra – DeMorgan's Example

$$v = a7 \cdot b7 \cdot \overline{s7} + \overline{a7} \cdot \overline{b7} \cdot s7$$

$$v = \overline{\overline{a7 \cdot b7 \cdot \overline{s7}} \cdot \overline{\overline{a7} \cdot \overline{b7} \cdot s7}}$$

NAND NAND



- An Overflow Detector

Designing Combinational Circuits

- Boolean Algebra

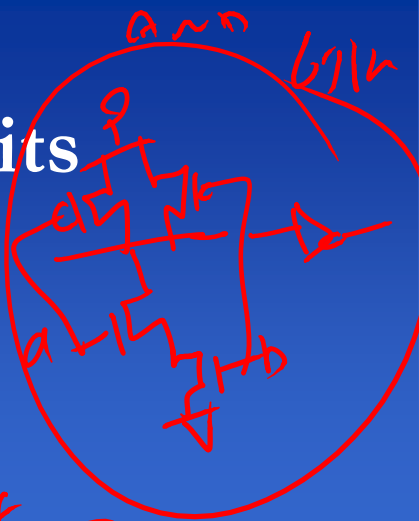
$$f = a \cdot b + \bar{a} \cdot c + b \cdot c$$

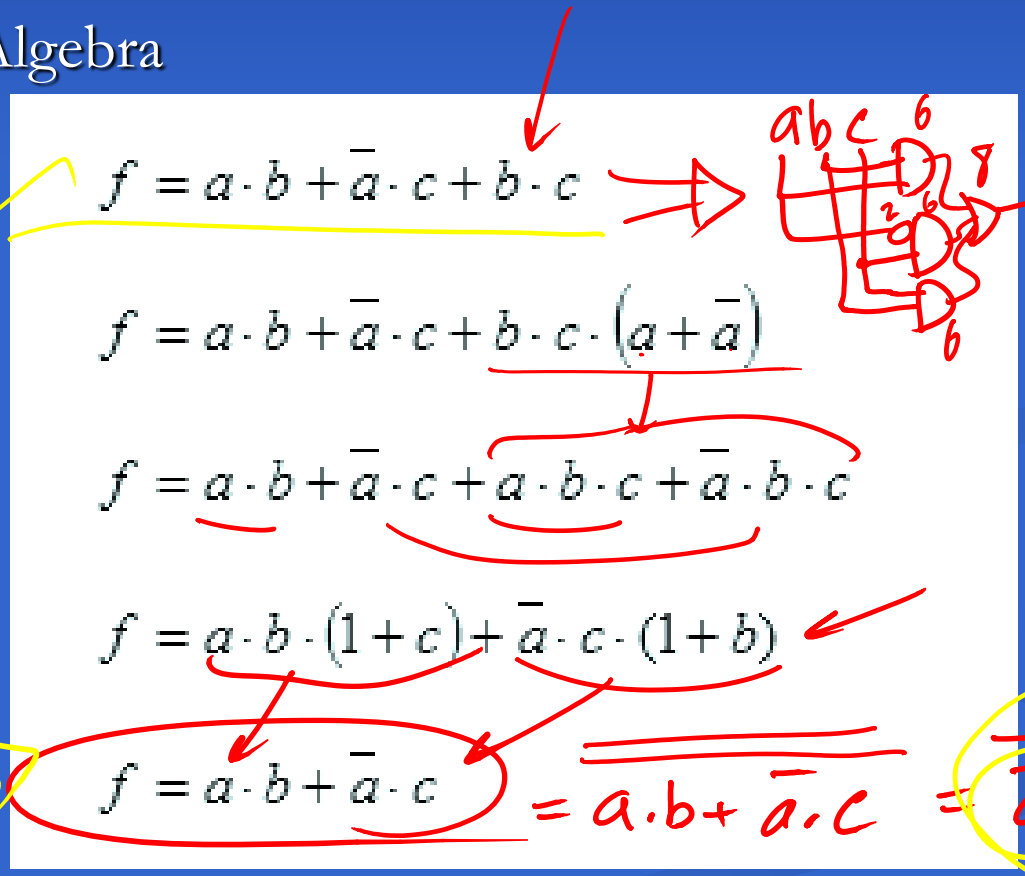
$$f = a \cdot b + \bar{a} \cdot c + b \cdot c \cdot (a + \bar{a})$$

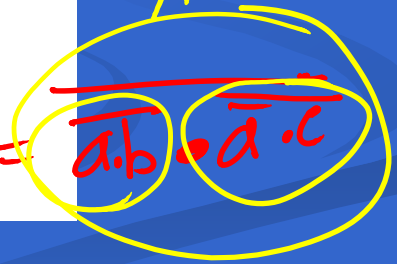
$$f = a \cdot b + \bar{a} \cdot c + a \cdot b \cdot c + \bar{a} \cdot b \cdot c$$

$$f = a \cdot b \cdot (1 + c) + \bar{a} \cdot c \cdot (1 + b)$$

$$f = a \cdot b + \bar{a} \cdot c$$







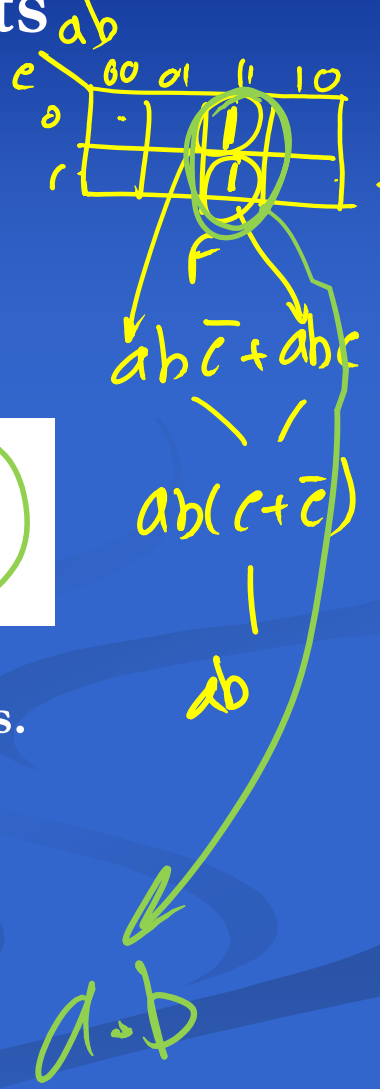
- **Minimization of Boolean Expression**

Designing Combinational Circuits

- Karnaugh Maps

$$f = \bar{a} \cdot b \cdot c + a \cdot \bar{b} \cdot c + a \cdot b \cdot c + a \cdot b \cdot \bar{c}$$

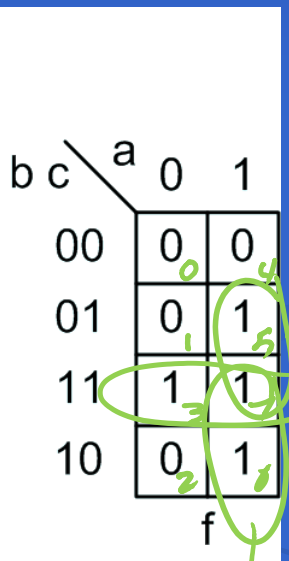
- Boolean expressions can be shown with Karnaugh maps.



Designing Combinational Circuits

- Karnaugh Maps

Row	a	b	c	f
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

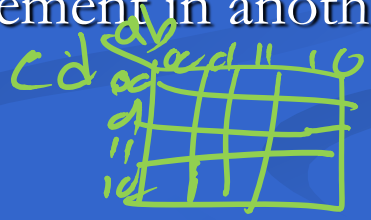


- A product term of a Boolean expression corresponds to every box of the Karnaugh map that contains a 1.

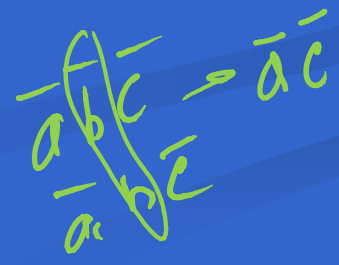
Designing Combinational Circuits

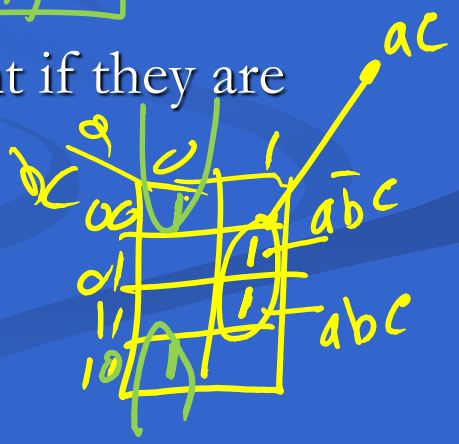
- Karnaugh Maps

- Boolean Adjacency: Two product terms are adjacent if they consist of the same Boolean variables and only one variable appears in its true form in one and complement in another (in one, -bar in another).



- Physical Adjacency: Two k-map boxes are adjacent if they are horizontally or vertically next to each other.

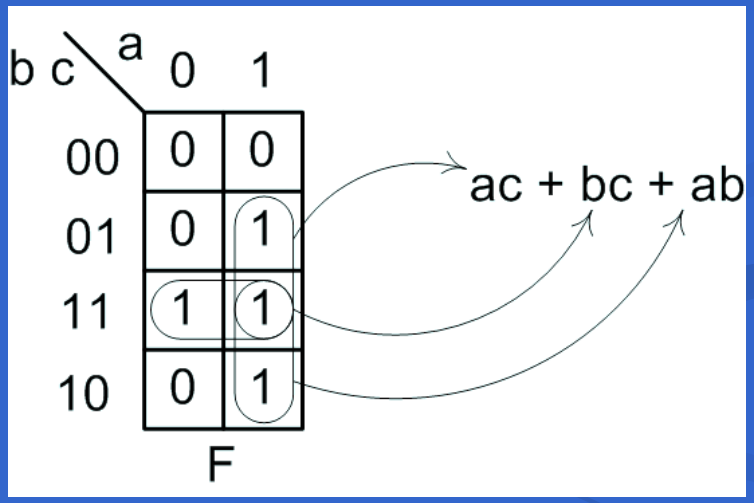




Designing Combinational Circuits

- Karnaugh Maps

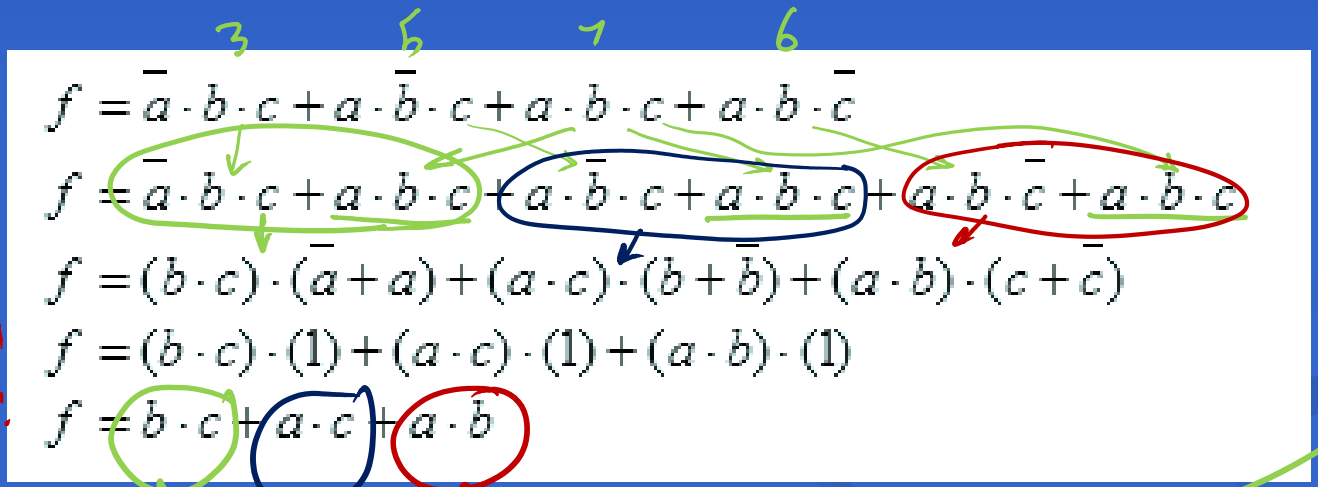
$$\bar{a} \cdot b \cdot c + a \cdot b \cdot c = b \cdot c$$



- **Minimization of Functions Using K-map Grouping of Terms**

Designing Combinational Circuits

- Karnaugh Maps



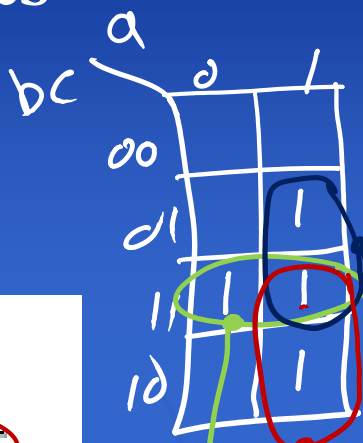
$$f = \bar{a}\cdot\bar{b}\cdot c + a\cdot\bar{b}\cdot c + a\cdot b\cdot c + a\cdot b\cdot\bar{c}$$

$$f = \bar{a}\cdot\bar{b}\cdot c + a\cdot\bar{b}\cdot c + a\cdot\bar{b}\cdot c + a\cdot b\cdot c + a\cdot b\cdot\bar{c} + a\cdot b\cdot c$$

$$f = (\bar{b}\cdot c)\cdot(\bar{a}+a) + (a\cdot c)\cdot(\bar{b}+\bar{b}) + (a\cdot b)\cdot(c+\bar{c})$$

$$f = (\bar{b}\cdot c)\cdot(1) + (a\cdot c)\cdot(1) + (a\cdot b)\cdot(1)$$

$$f = \bar{b}\cdot c + a\cdot c + a\cdot\bar{b}$$



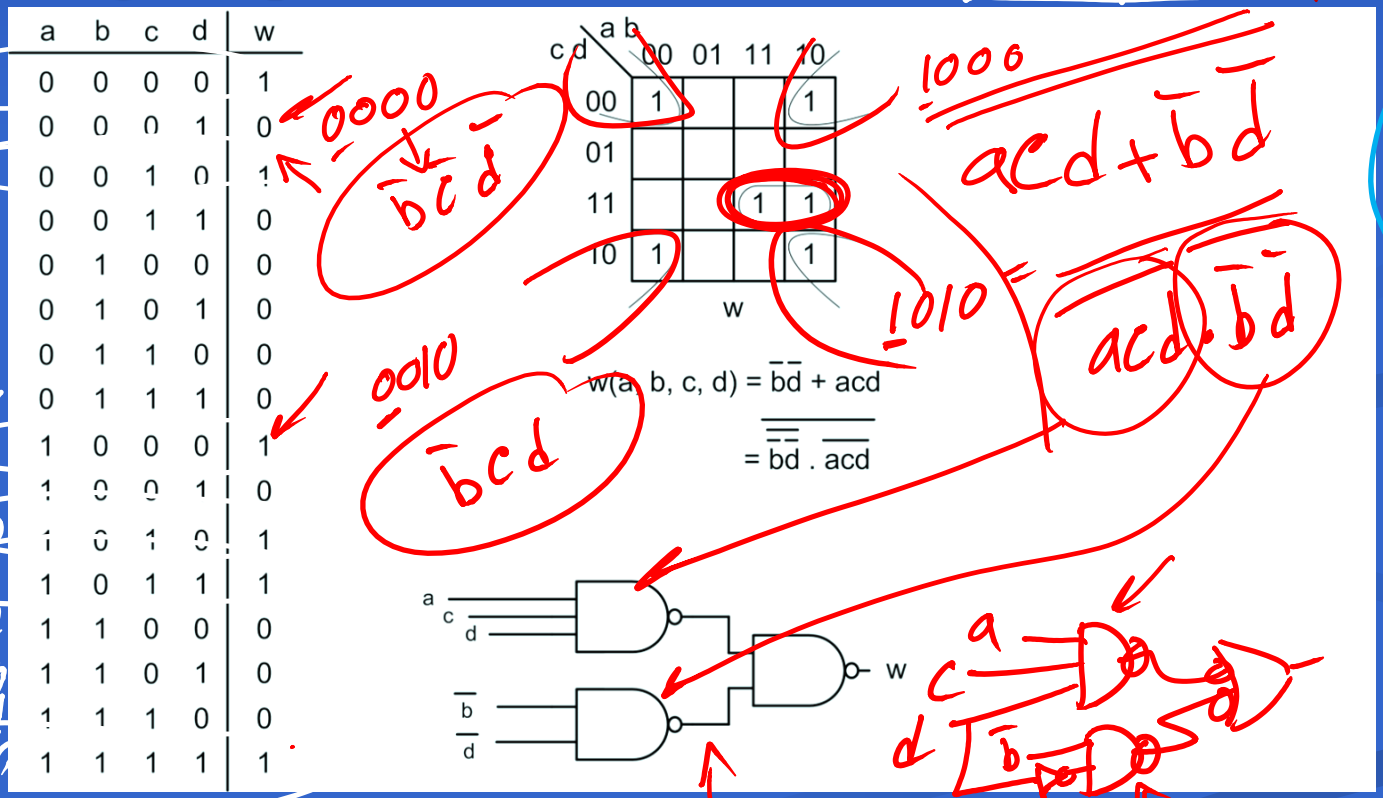
- Boolean manipulations correspond to the mapping of function f.

$a+a=a$

$a+a+a=a$

Designing Combinational Circuits

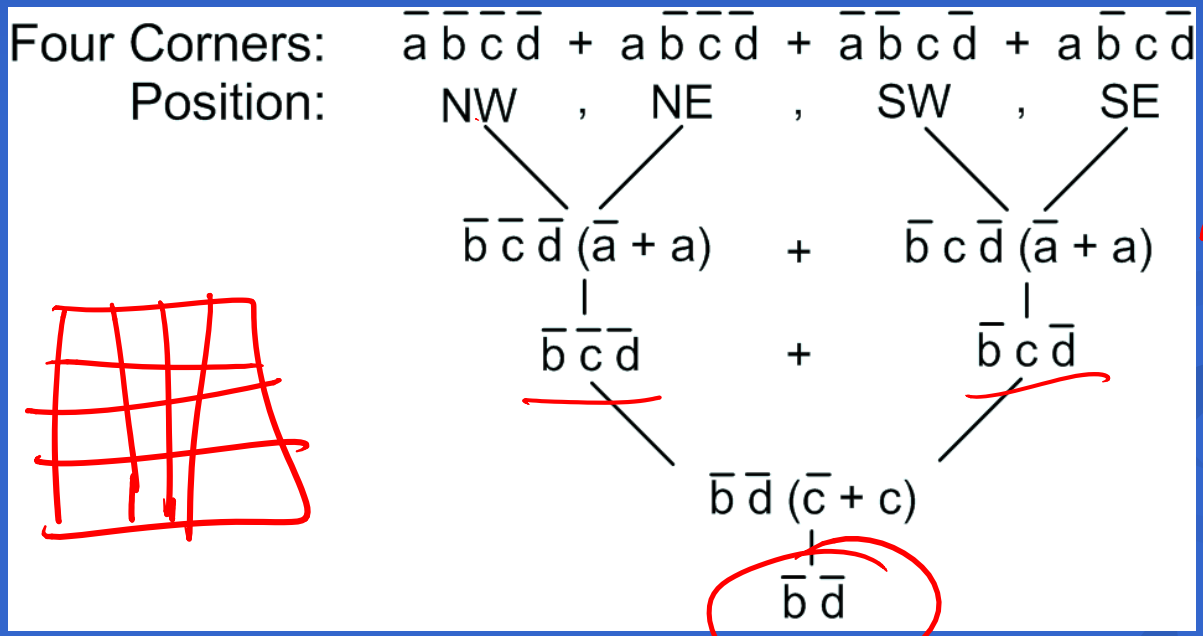
- Karnaugh Maps

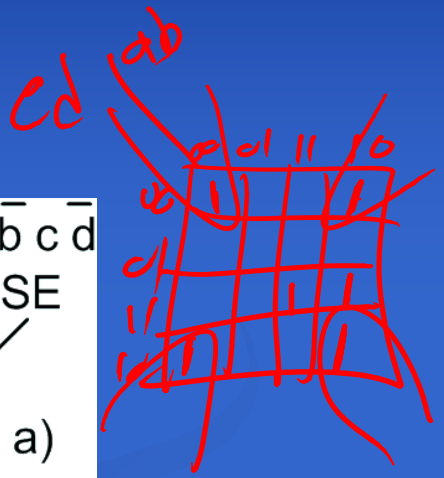


- Minimizing a 4-variable Function

Designing Combinational Circuits

- Karnaugh Maps



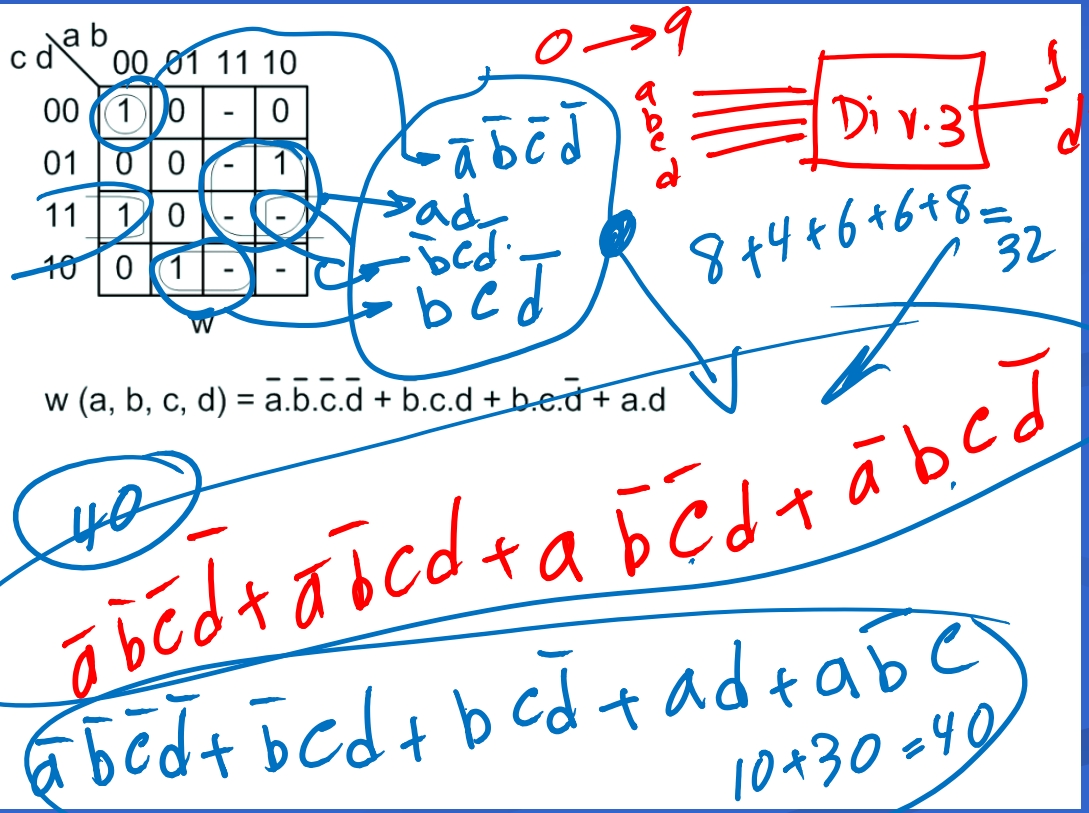


- **Combining Four Corners of a 4-variable Map**

Designing Combinational Circuits

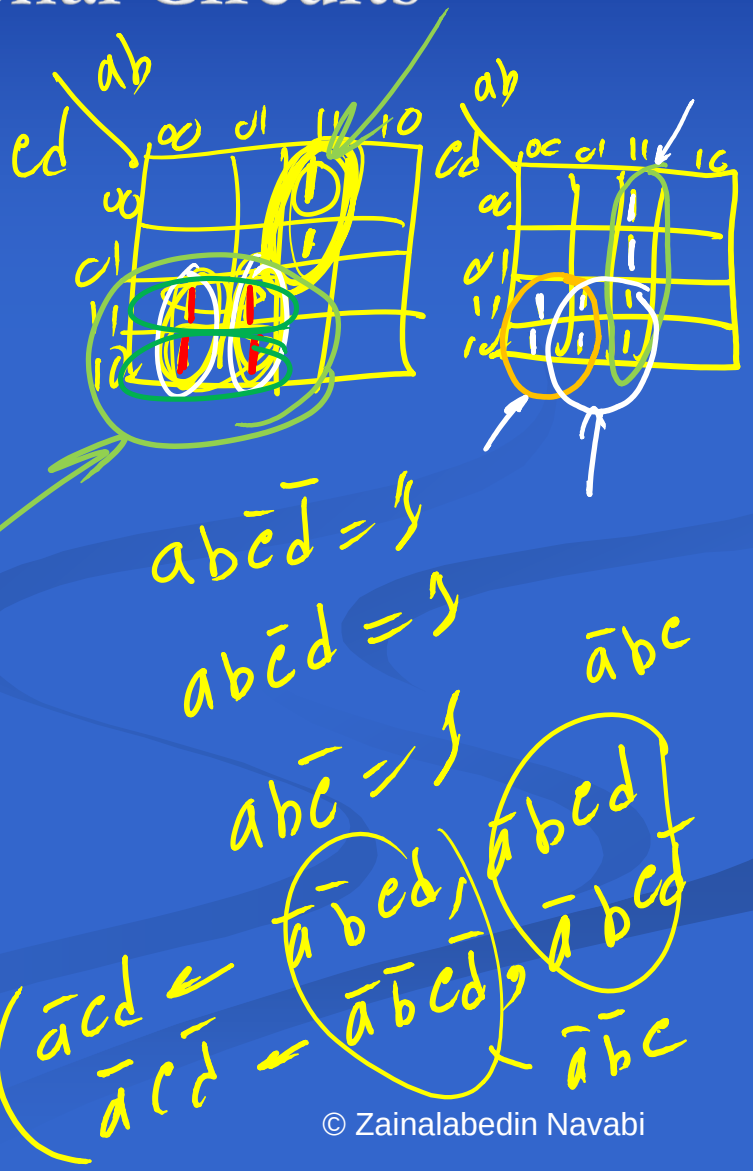
- Don't Care Values

a	b	c	d	w
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	-
1	0	1	1	-
1	1	0	0	-
1	1	0	1	-
1	1	1	0	-
1	1	1	1	-



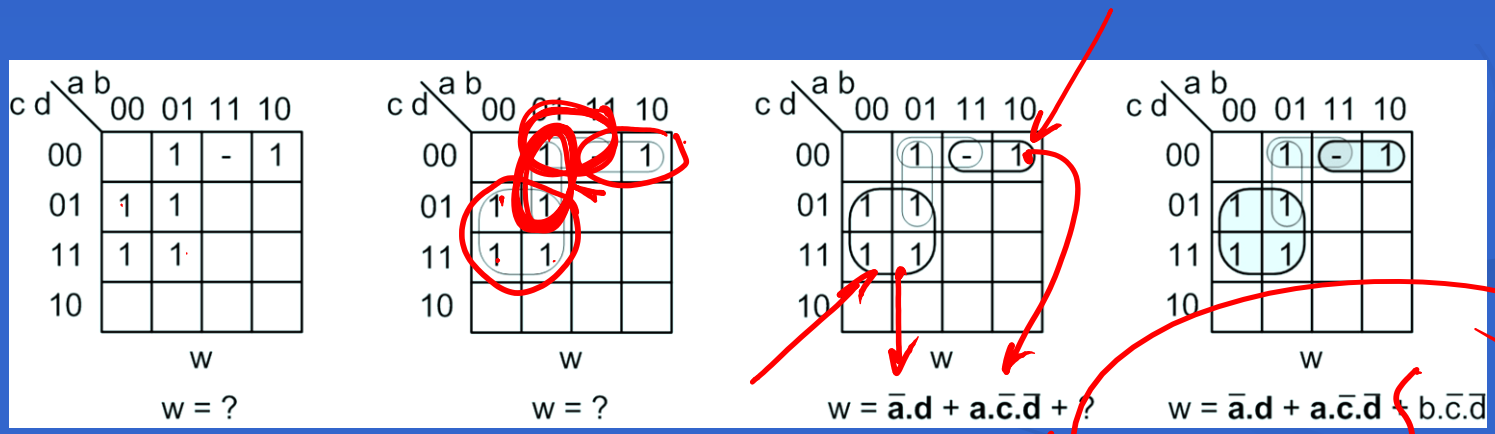
Designing Combinational Circuits

- Minimal Coverage
  - **Implicant.**
  - **Prime Implicant.**
  - **Essential Prime Implicant.**



Designing Combinational Circuits

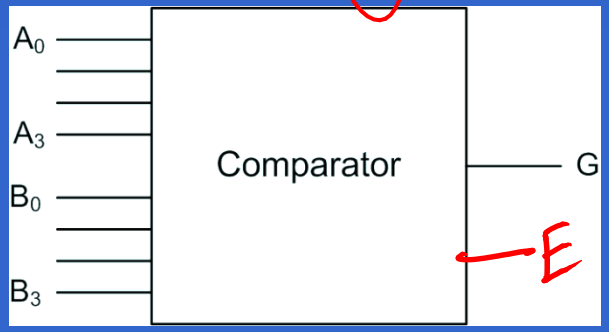
- Minimal Coverage

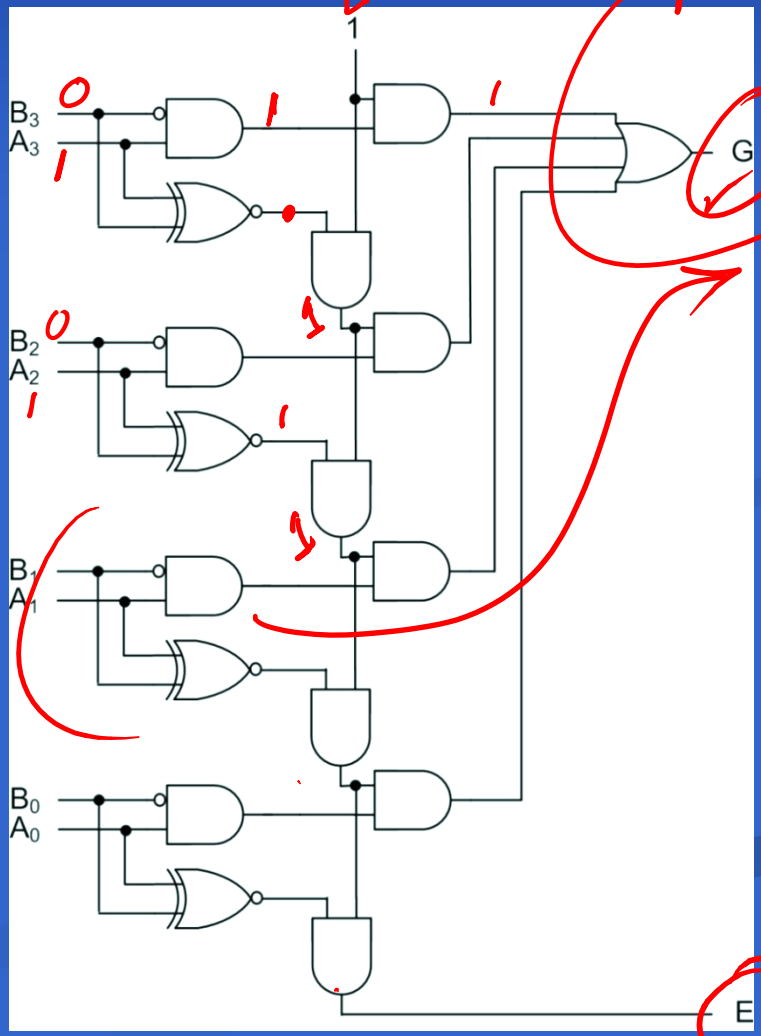


- **Finding Minimal Coverage**

Designing Combinational Circuits

- Iterative Hardware

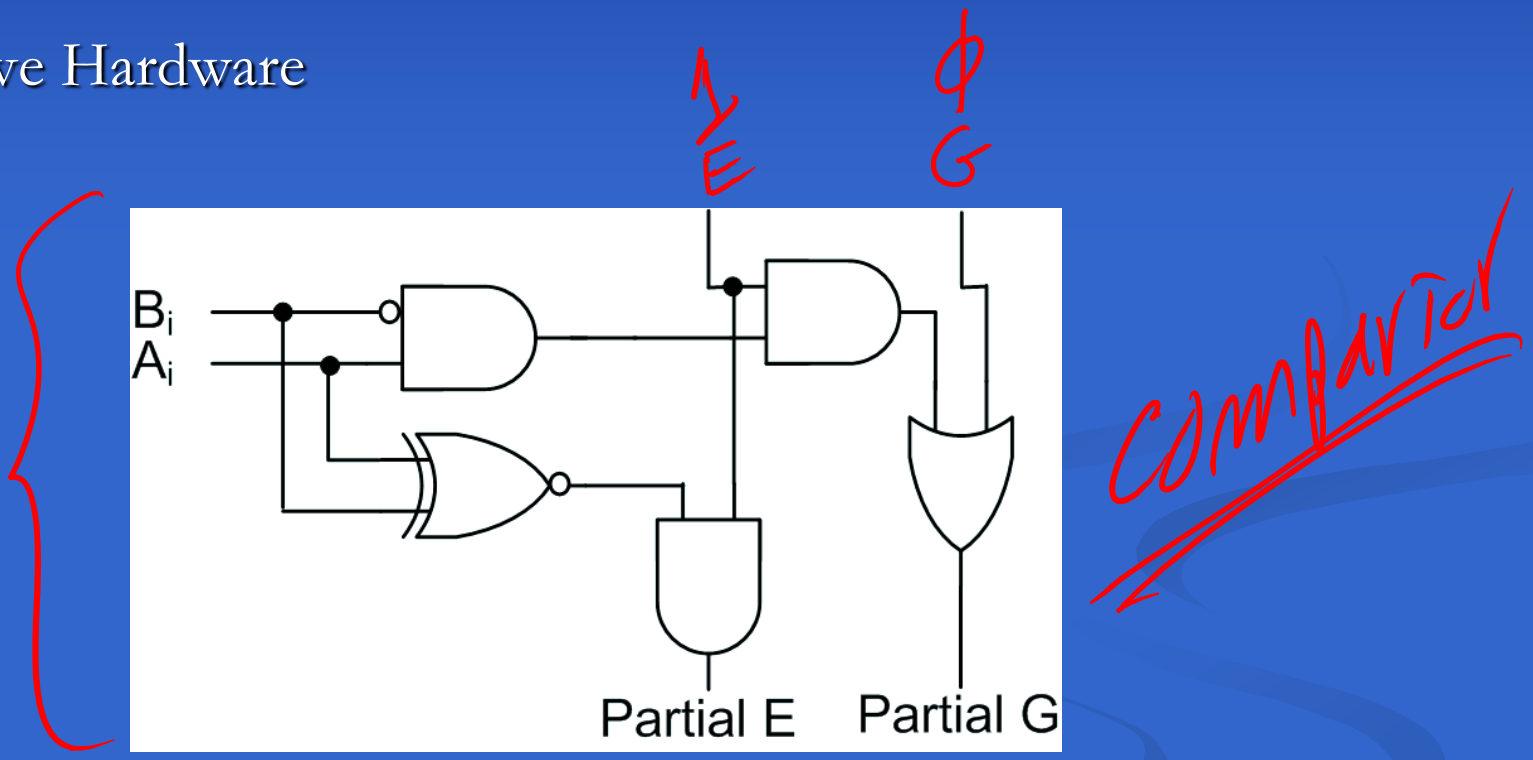




- An Iterative Comparator

Designing Combinational Circuits

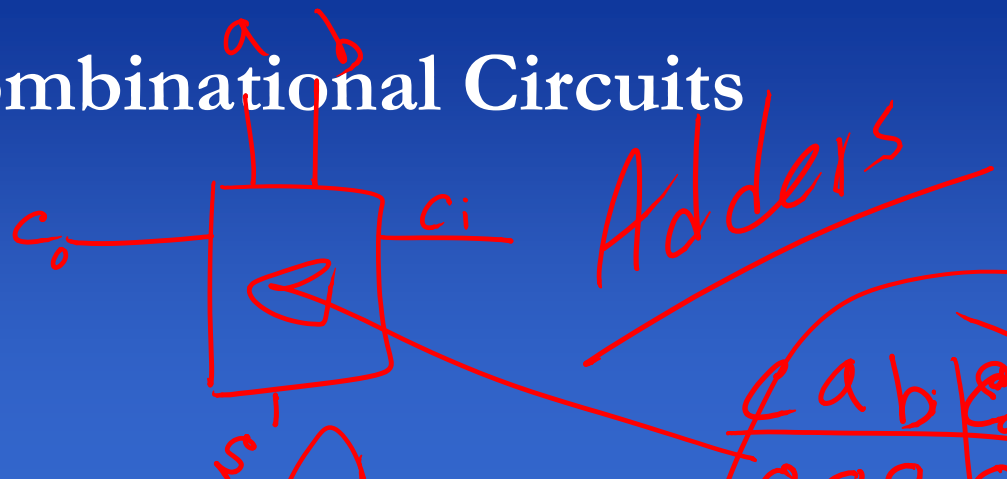
- Iterative Hardware



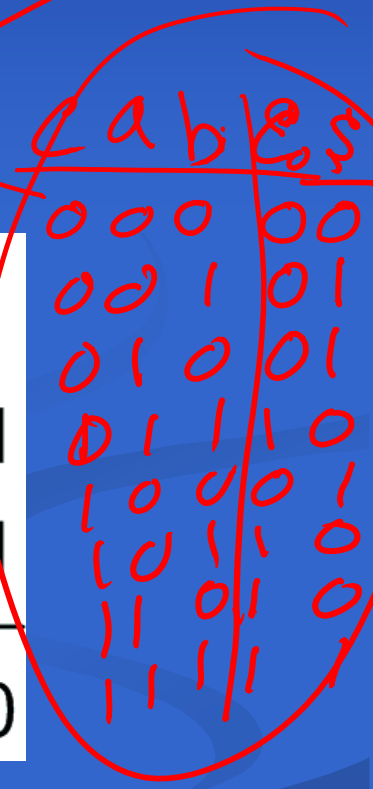
- **A Comparator Bit for a Regular Structure**

Designing Combinational Circuits

- Iterative Hardware



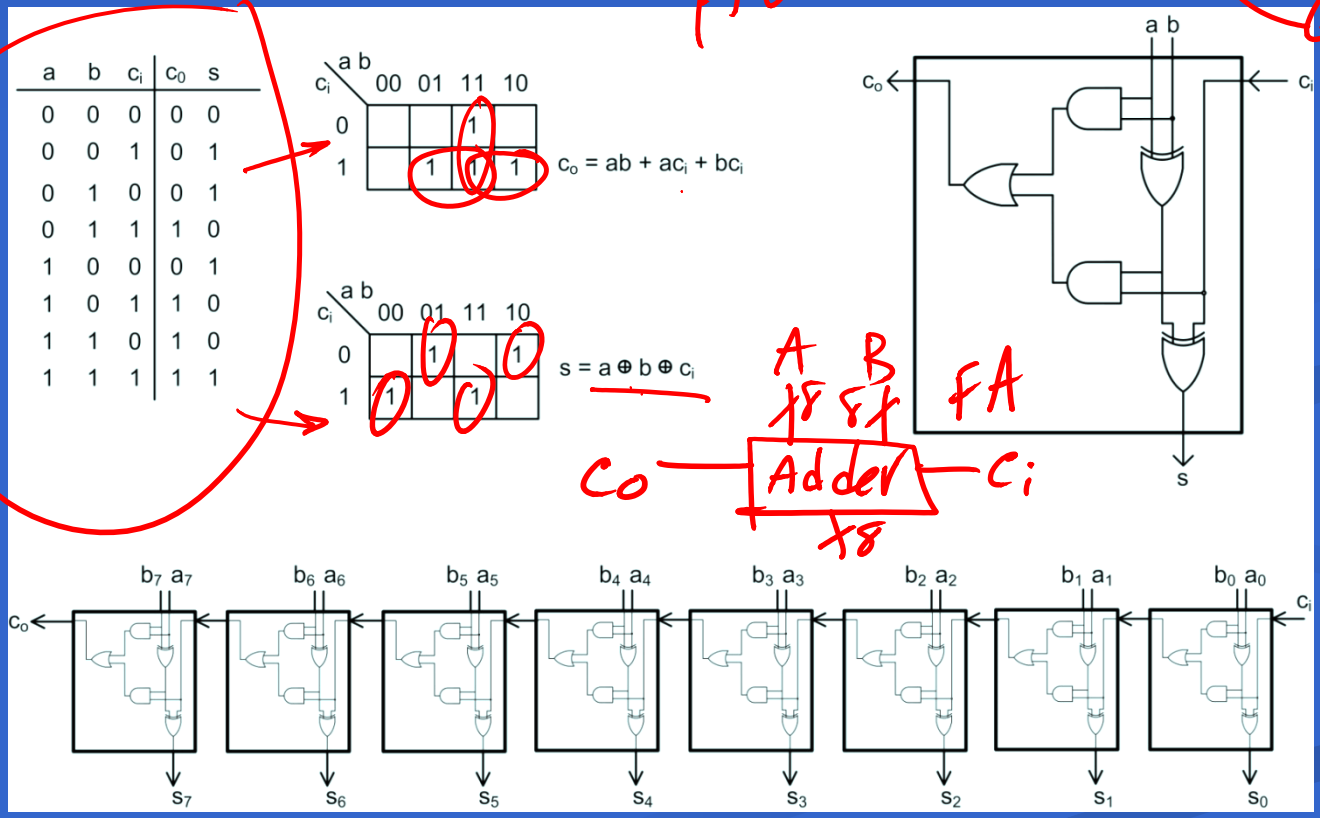
	0	1	0	1	1	1	1	
A:	0	0	1	0	1	0	1	1
B:	0	0	1	0	1	1	0	1
	0	1	0	1	1	0	0	0



  - **The process of adding repeats for every bit position.**

Designing Combinational Circuits

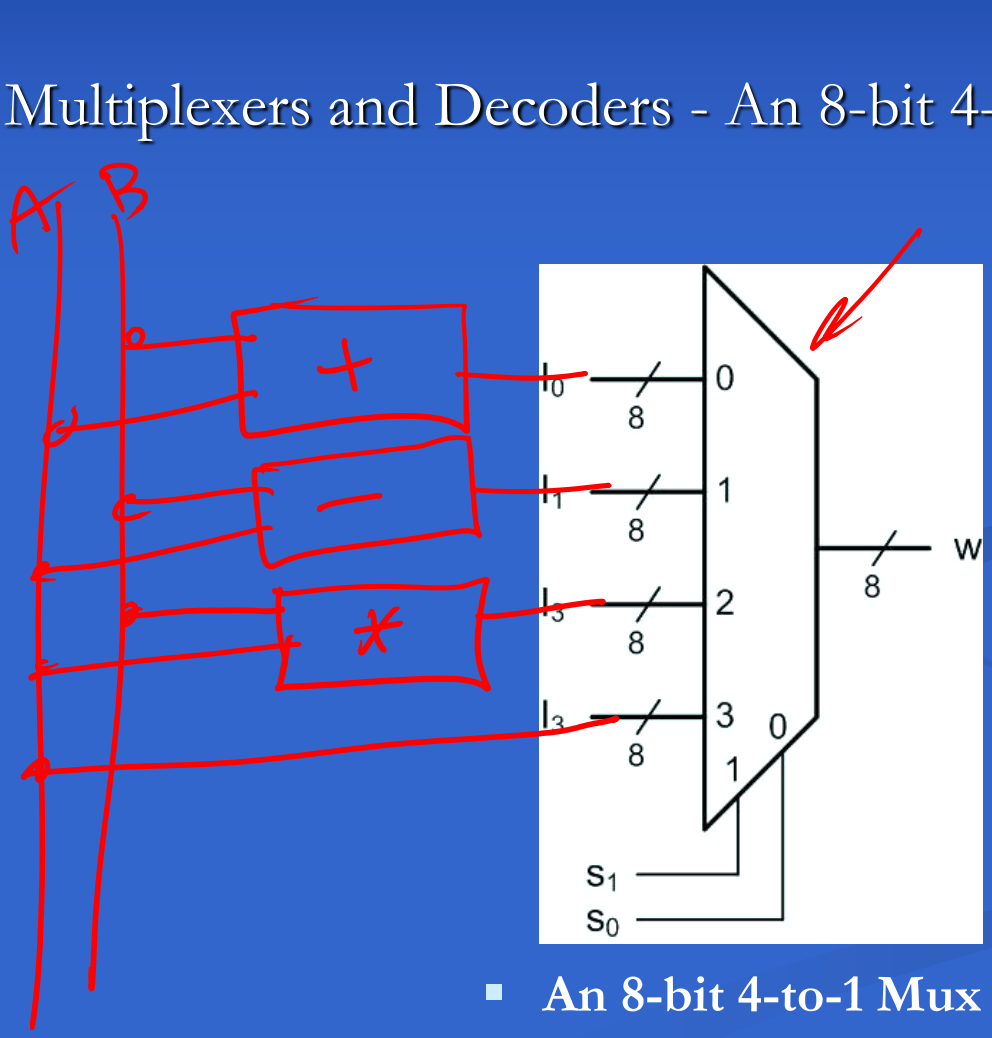
- Iterative Hardware

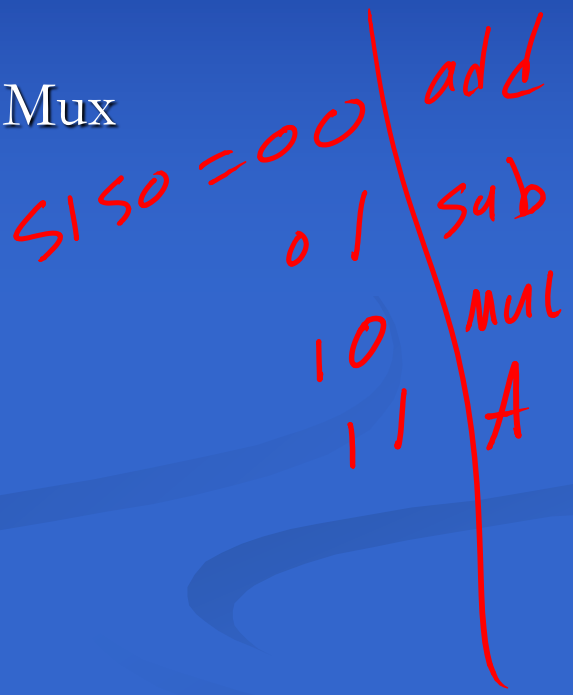


- An 8-bit Ripple Carry Adder

Designing Combinational Circuits

- Multiplexers and Decoders - An 8-bit 4-to-1 Mux

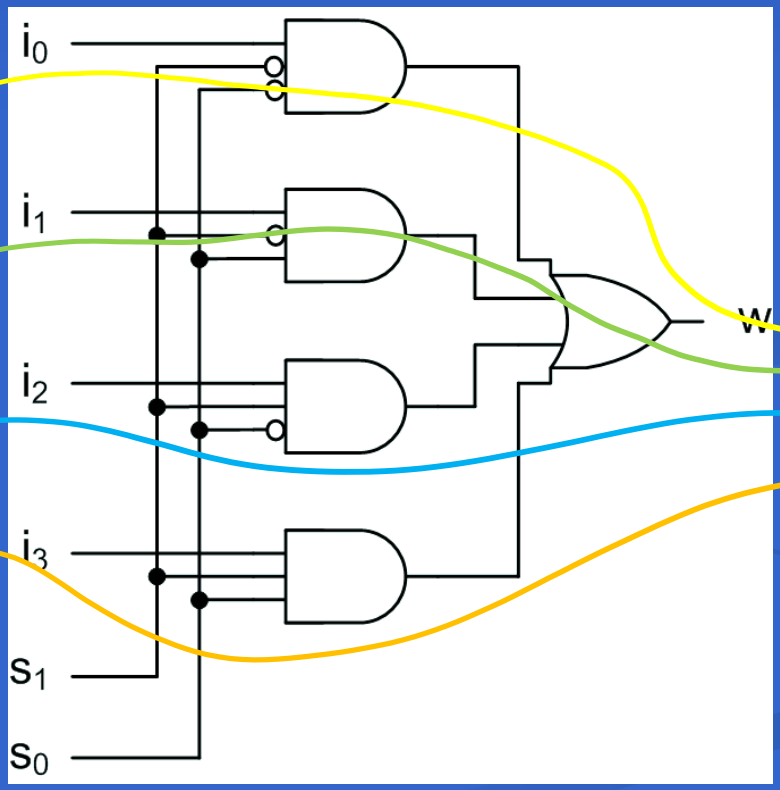




- **An 8-bit 4-to-1 Mux**

Designing Combinational Circuits

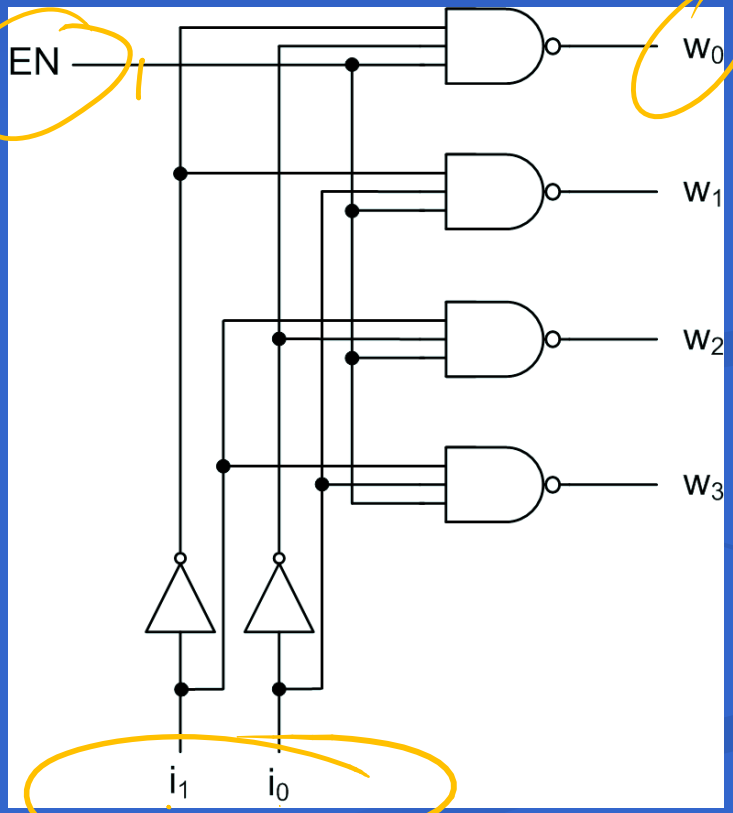
- Multiplexers and Decoders



- **A 4-to-1 Mux**

Designing Combinational Circuits

- Multiplexers and Decoders - A 2-to-4 Decoder

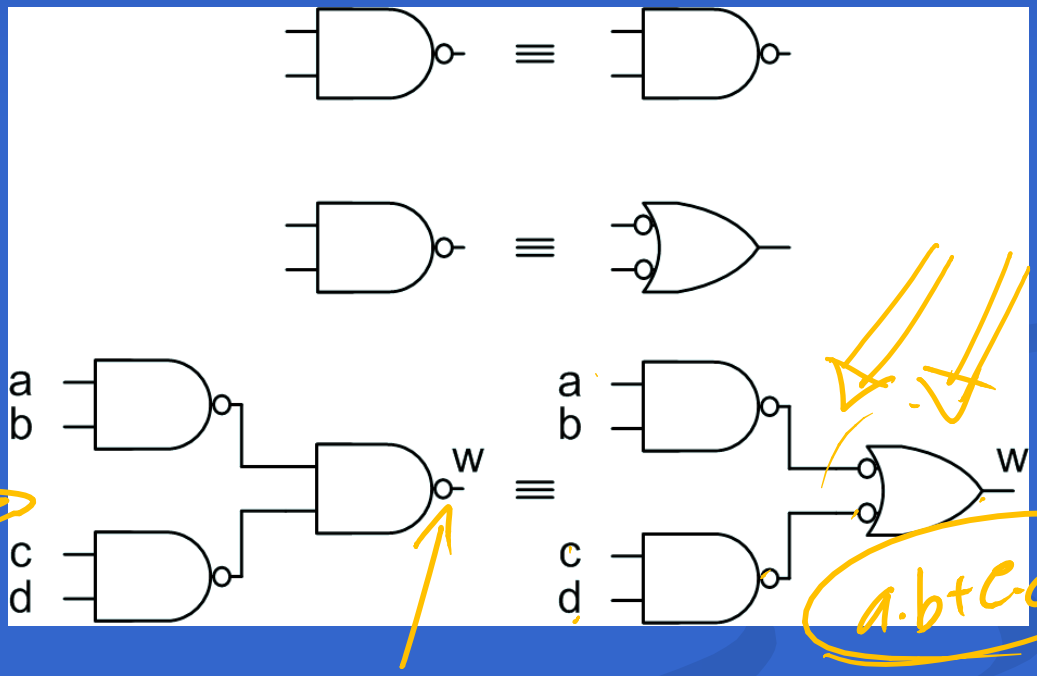


- A 2-to-4 Decoder

Designing Combinational Circuits

- Activity Levels

$$\overline{a \cdot b} = -(a \cdot b)$$
$$\overline{a \cdot b} = (\bar{a} + \bar{b})$$

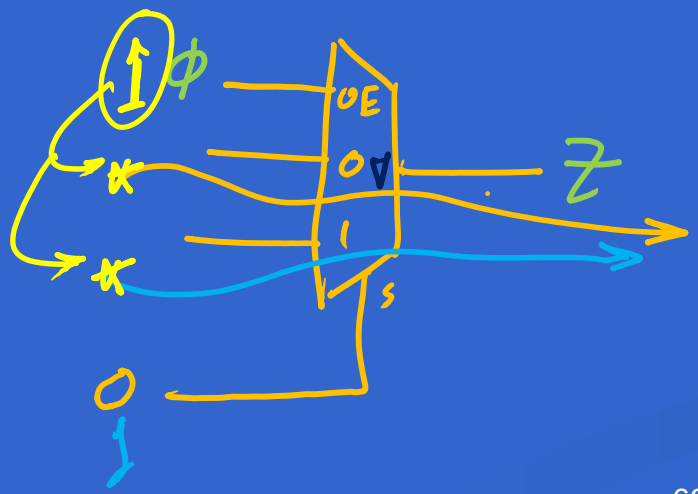


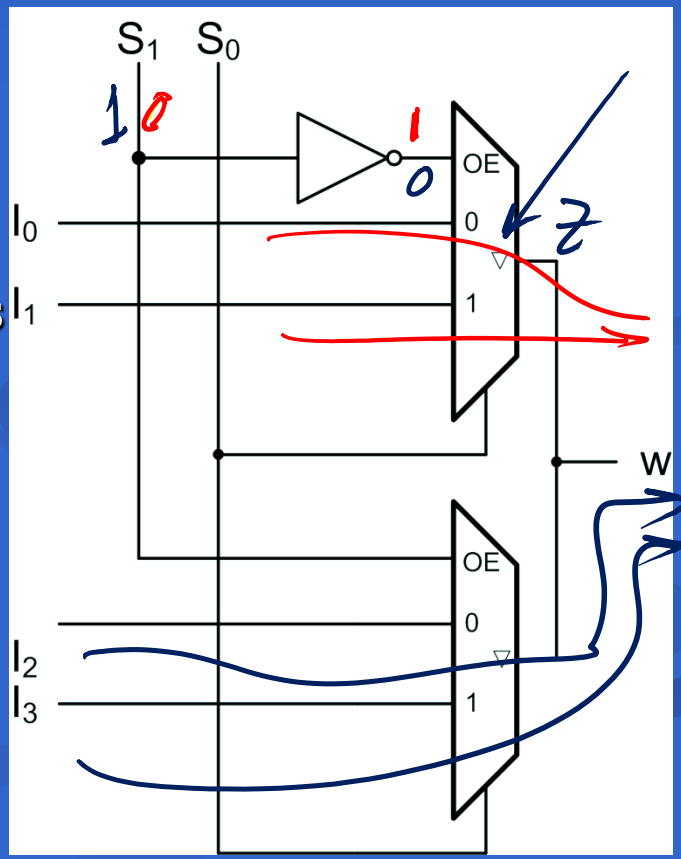
- **A NAND gate can also be looked at as an OR gate with active-low inputs and active-high output.**

Designing Combinational Circuits

- Enable / Disable Inputs

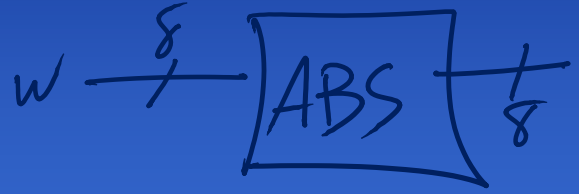
- **Wiring Circuits with OE Control Inputs**

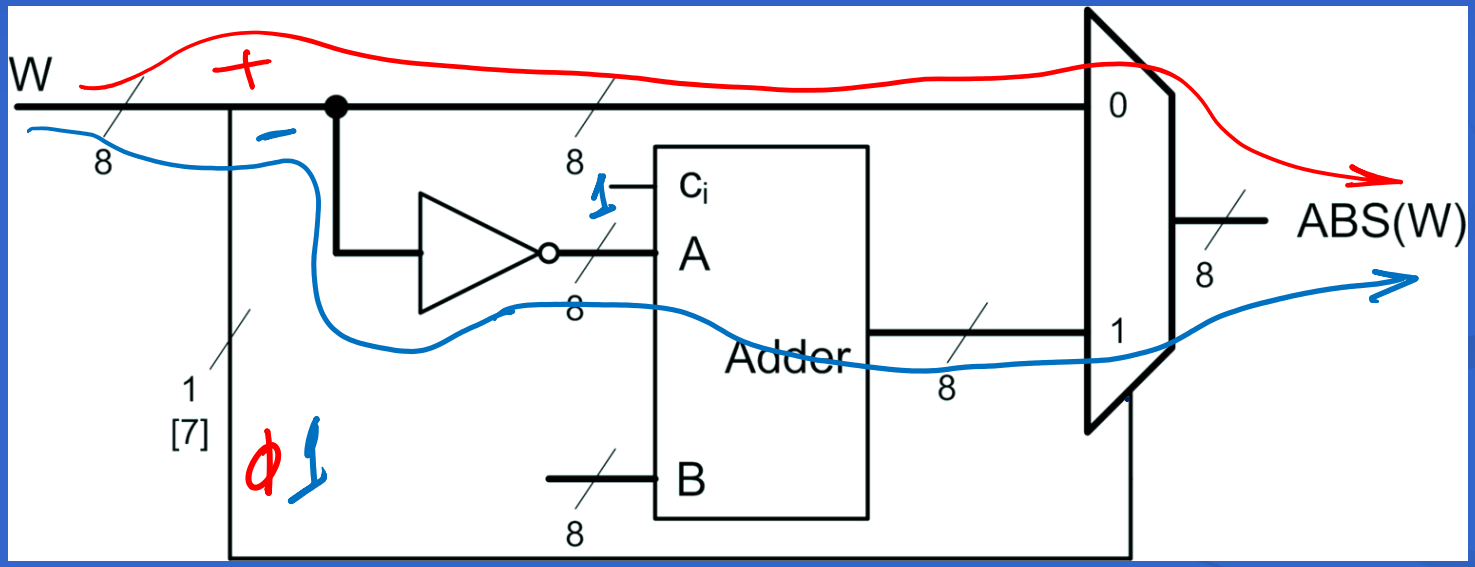




Designing Combinational Circuits

- A High-Level Design





- An Absolute Value Circuit

Logic Design Concepts

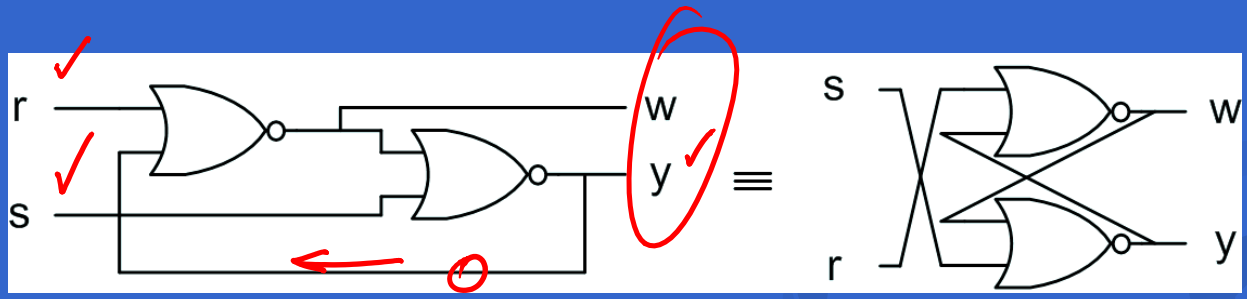
- Number Systems
- Binary Arithmetic
- Basic Logic Gates and Structures
- Designing Combinational Circuits
- Storage Elements
- Sequential Circuit Design
- Memories
- Bidirectional IO
- A Comprehensive Example: Serial Adder
- Summary

Storage Elements

- The Basic Latch
- Clocked D Latch
- Flip-Flops
- Flip-Flop Control
- Registers

Storage Elements

- The Basic Latch



- **The Basic latch (Two Equivalent Circuits)**

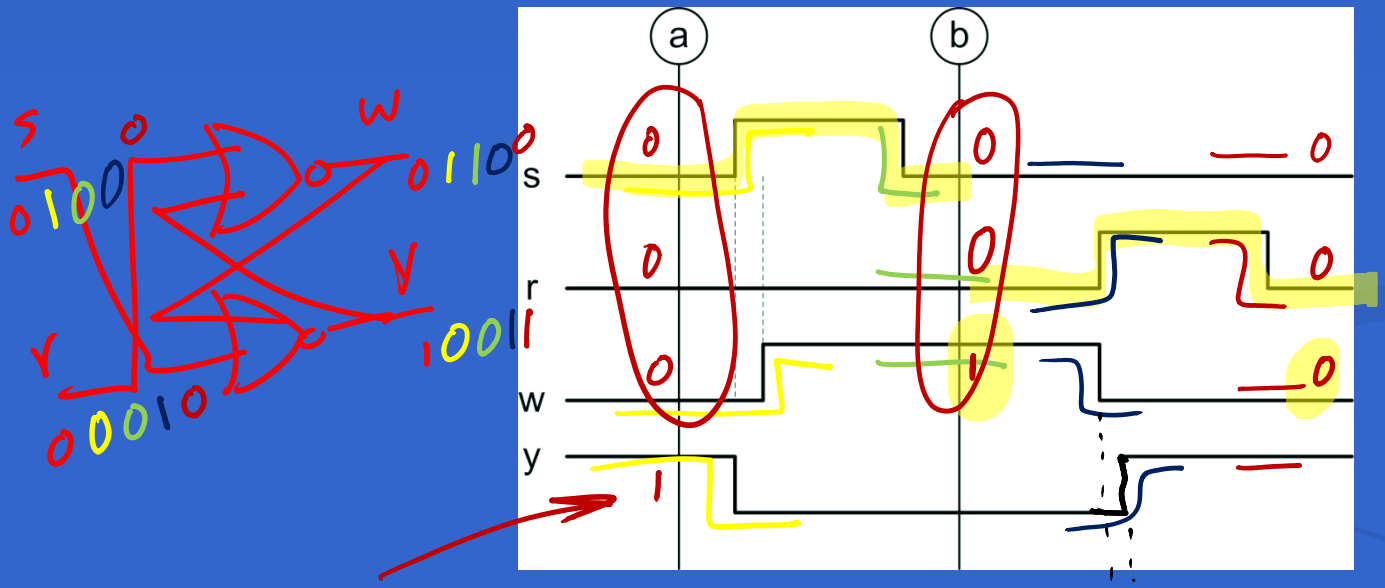
$$g(t) = f(a(t), g(t-1), \ldots)$$

$$g(t) = f(a(t), s(t))$$

$$s(t) = g(a(t), s(t-1))$$

Storage Elements

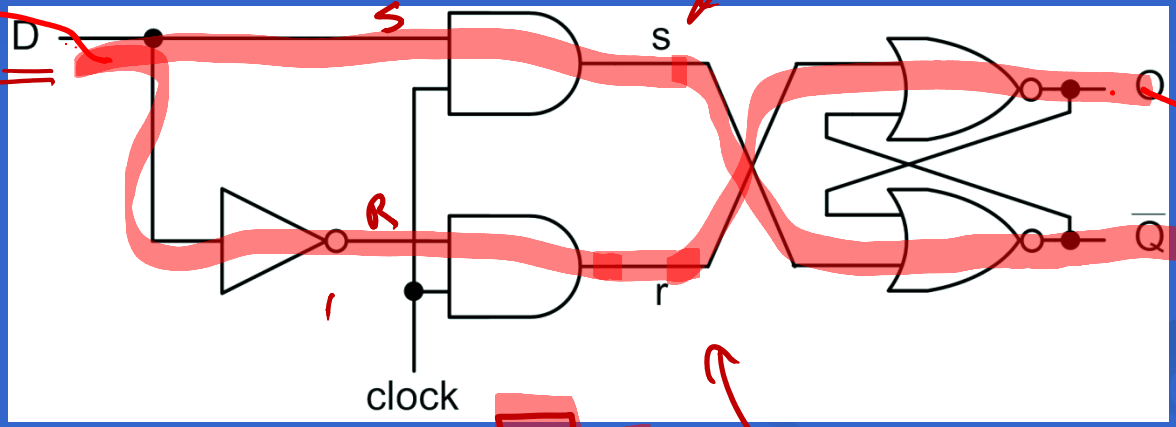
- The Basic Latch



- **Setting and Resetting the Basic Latch**

Storage Elements

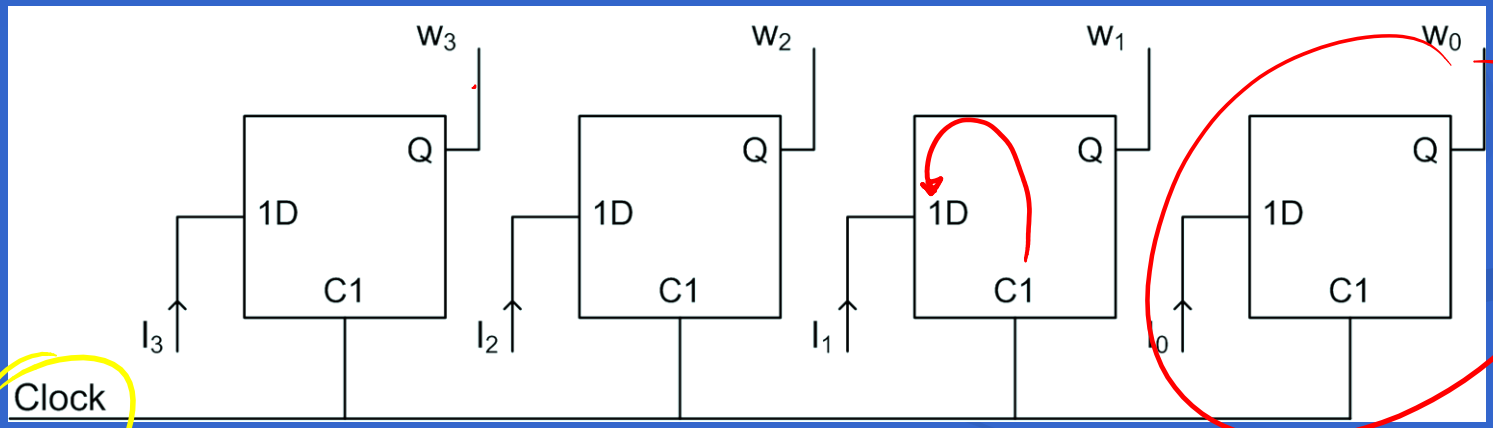
- Clocked D Latch



- When clock becomes 1, the value of D will be stored until the next time that clock becomes 1.

Storage Elements

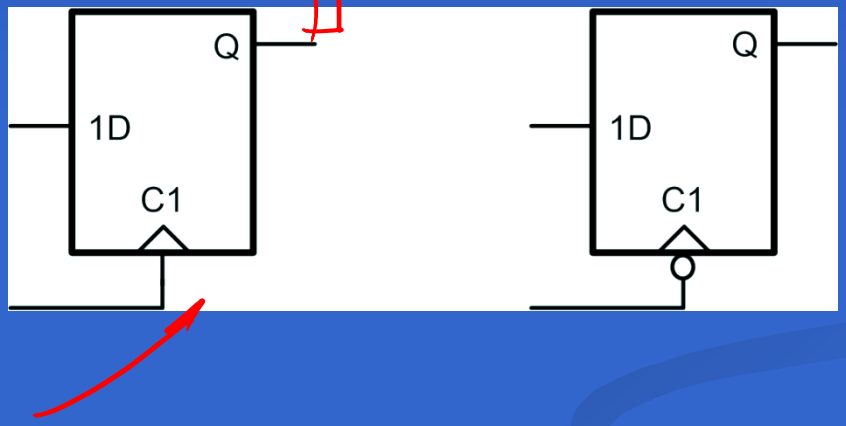
- Clocked D Latch



- **A Quad Latch Using a Symbolic Representation of a Latch**

Storage Elements

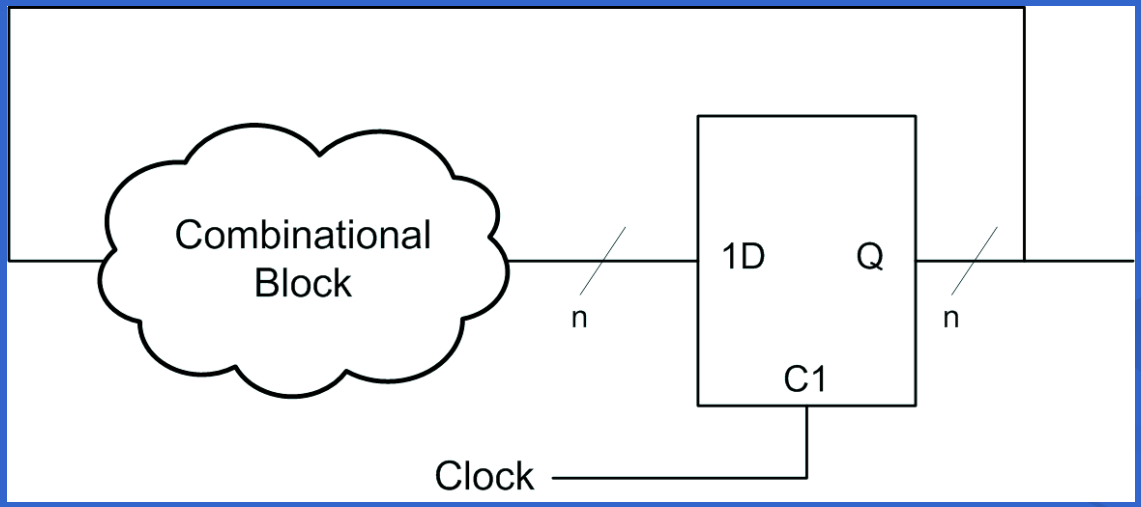
- Flip-Flops



  - **Edge Trigger Flip-flops**

Storage Elements

- Flip-Flops

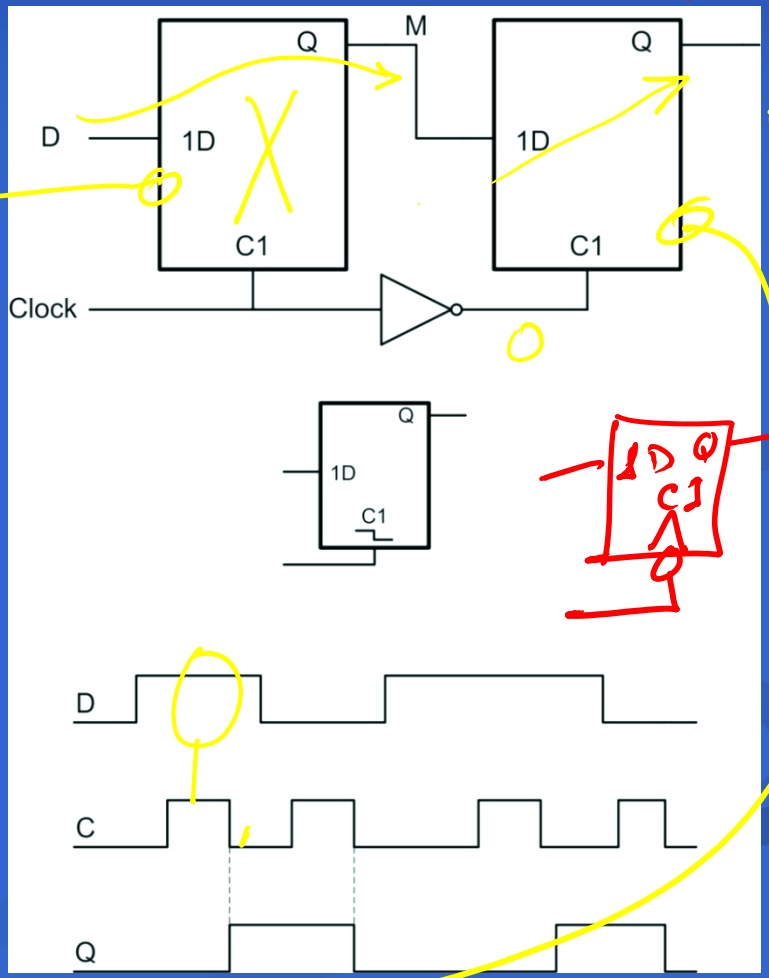


- **Latch Feedback Causes Unpredictable Results**

Storage Elements

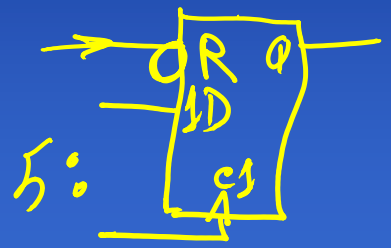
- Flip-Flops

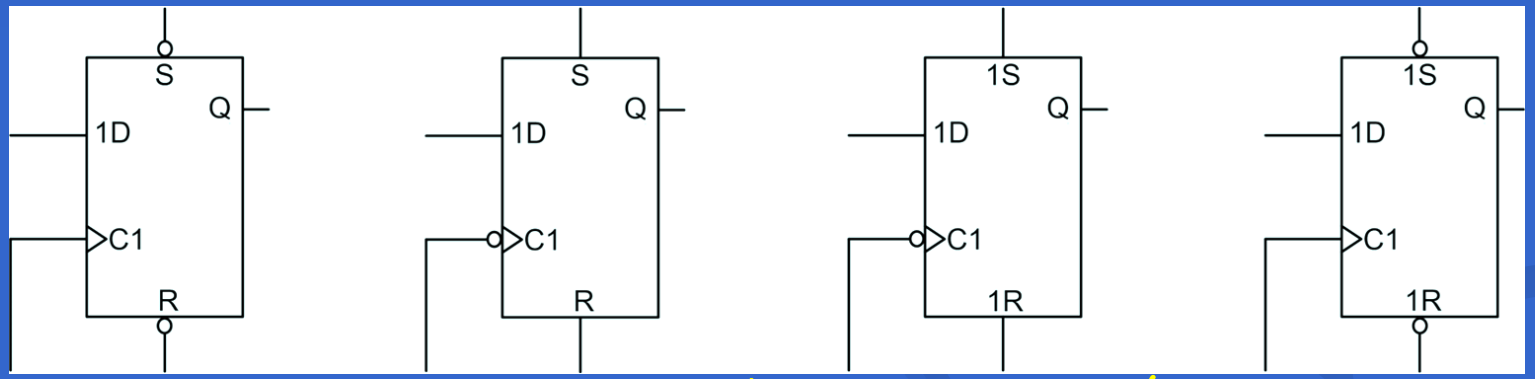
- **A D-Flip-Flop, Its Structure, Symbolic Notation and Waveform**



Storage Elements

- Flip-Flop Control





1: rising active low asynch set & reset

2: falling edge active hi asynch set & reset

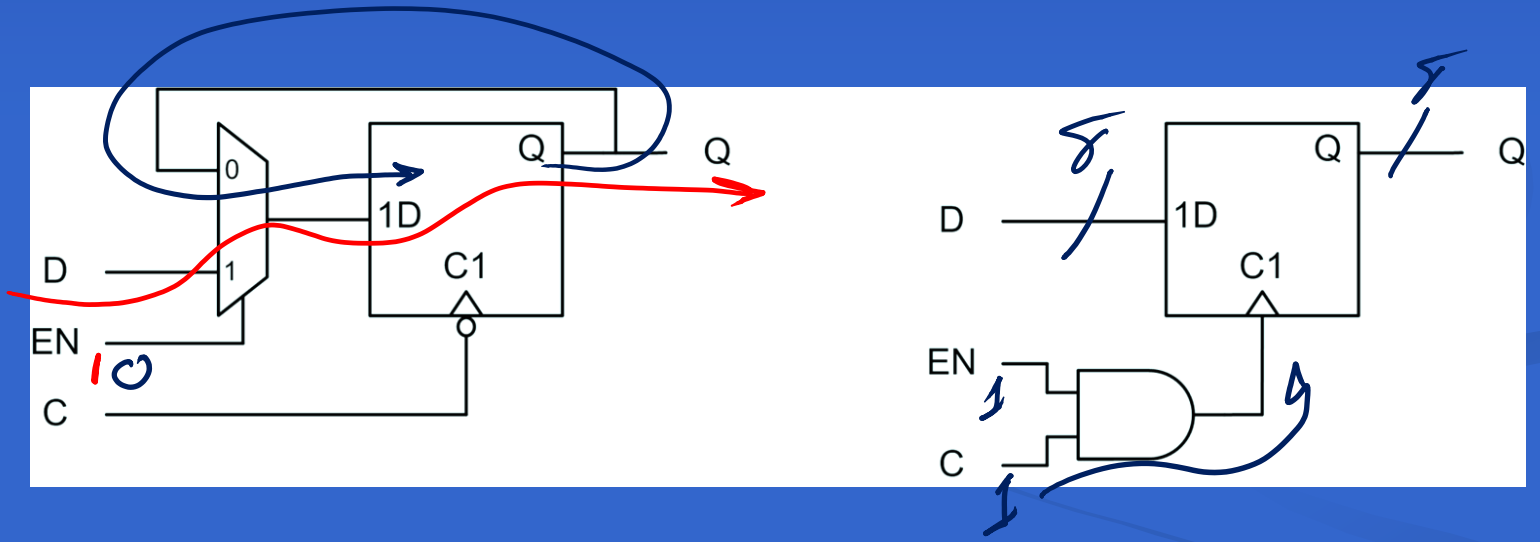
3: falling edge active high synchron set & reset

4: rising edge active low synchron set & reset

- **Flip-flops with Synchronous and Asynchronous Control**

Storage Elements

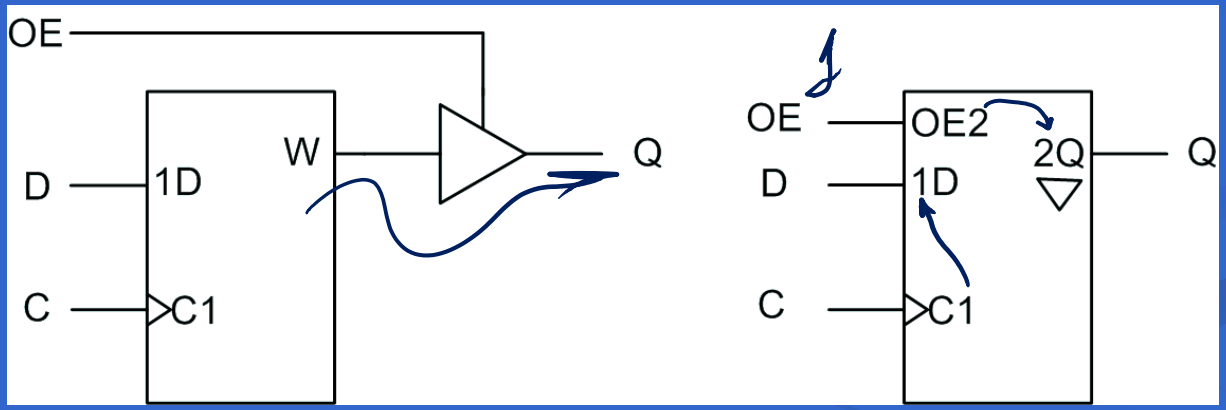
- Flip-Flop Control



- **Clock Enabling**

Storage Elements

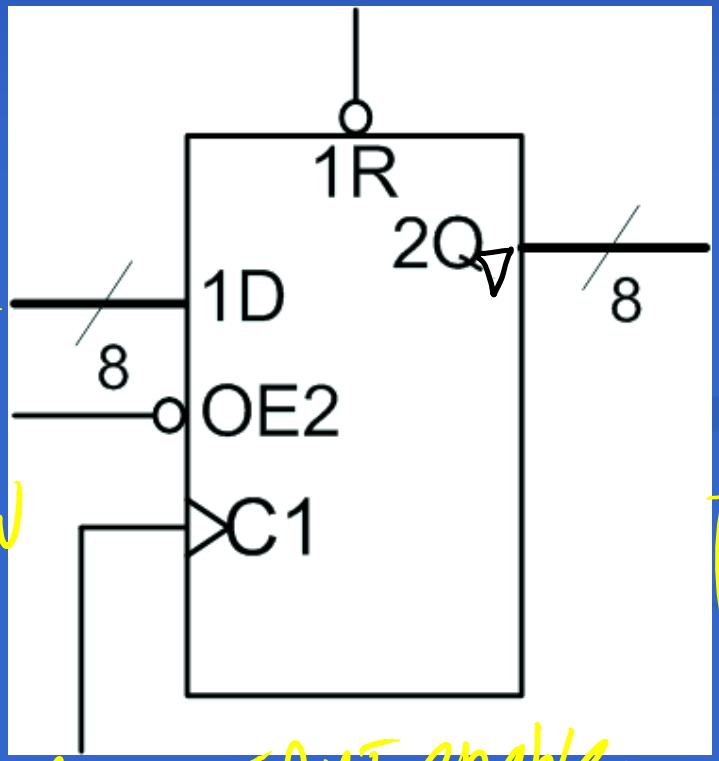
- Flip-Flop Control

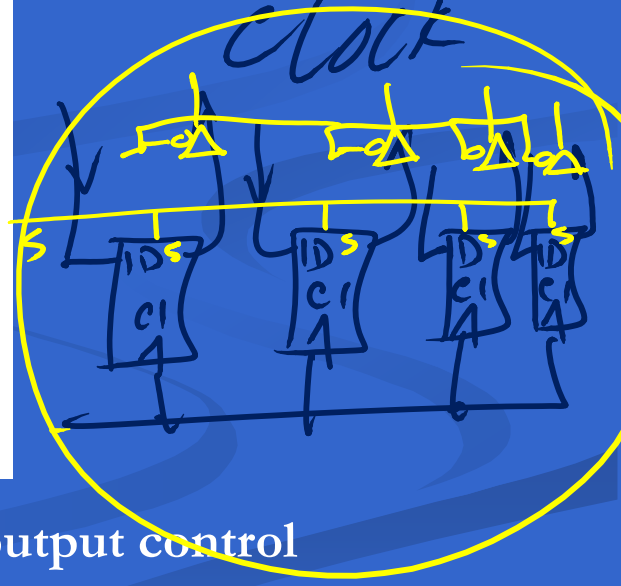


- **Three-State Control**

Storage Elements

- Registers





- **An 8-bit register with an active-low three-state output control and a synchronous active low reset.**

Logic Design Concepts

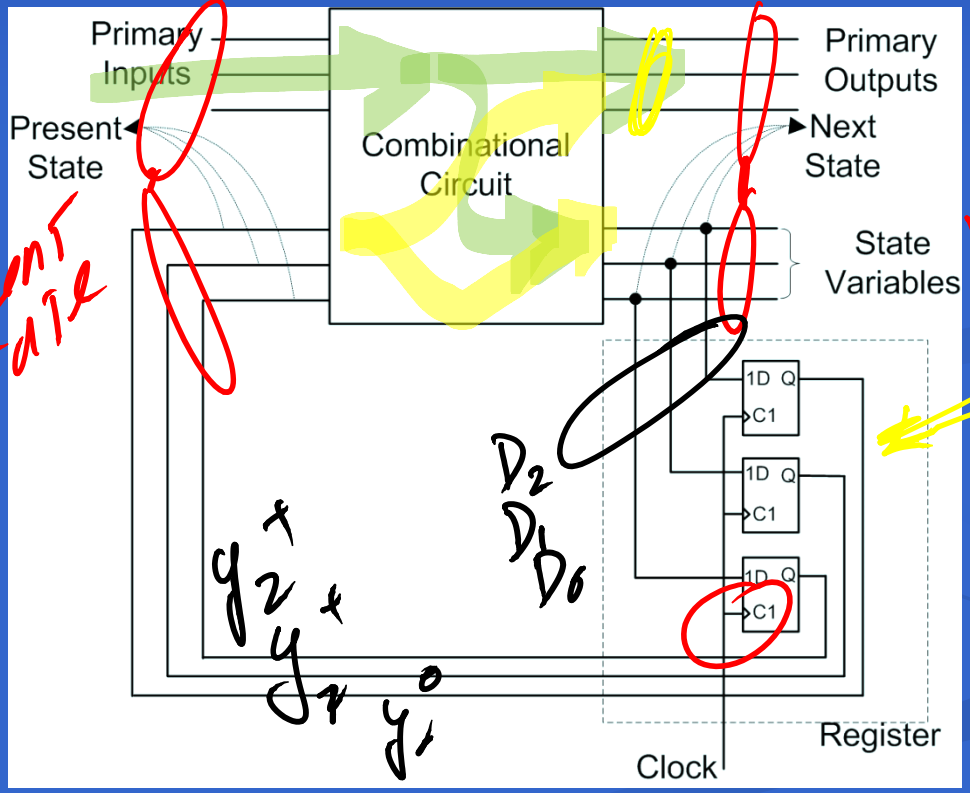
- Number Systems
- Binary Arithmetic
- Basic Logic Gates and Structures
- Designing Combinational Circuits
- Storage Elements
- Sequential Circuit Design
- Memories
- Bidirectional IO
- A Comprehensive Example: Serial Adder
- Summary

Sequential Circuit Design

- Finite State Machines
- Designing State Machines
- Mealy and Moore Machines
- One-Hot Realization
- Sequential Packages

Sequential Circuit Design

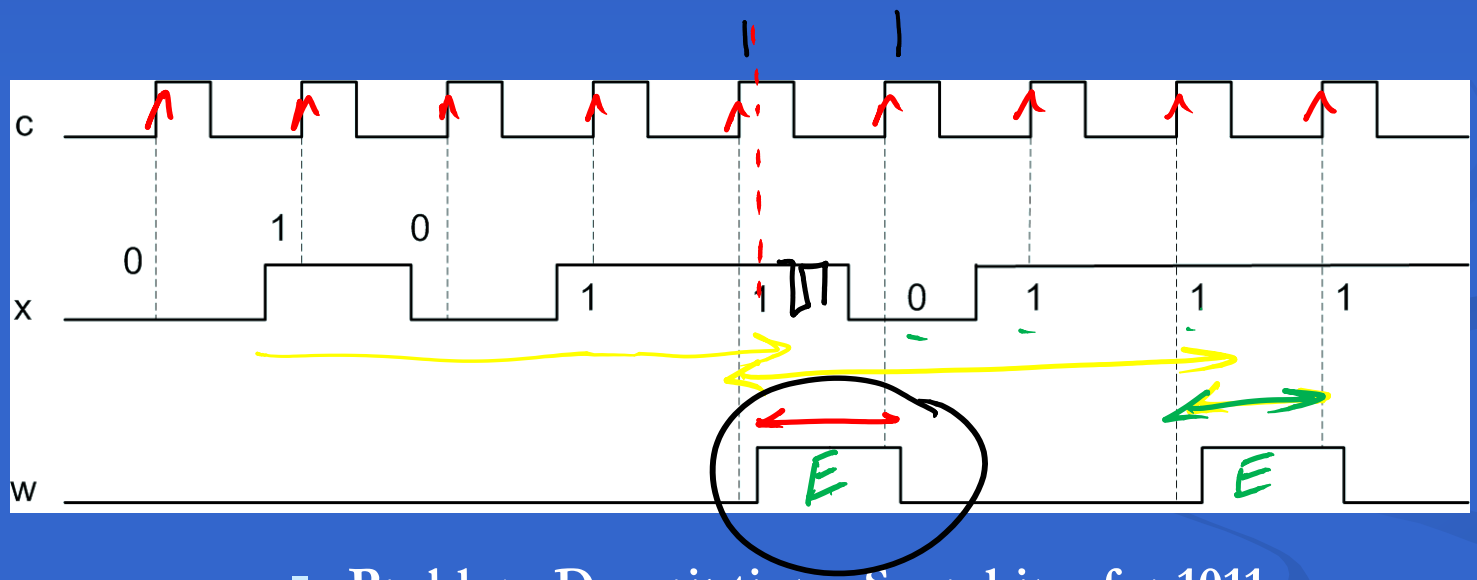
- Finite State Machines



- **Huffman Model of a Sequential Circuit**

Sequential Circuit Design

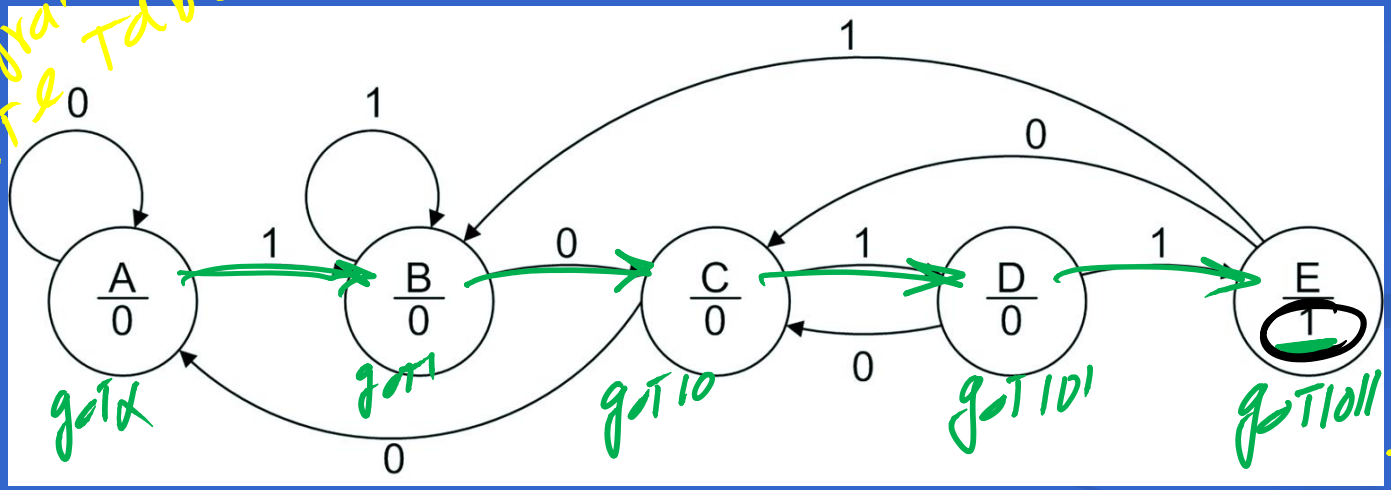
- Designing State Machines



- **Problem Description - Searching for 1011**

Sequential Circuit Design

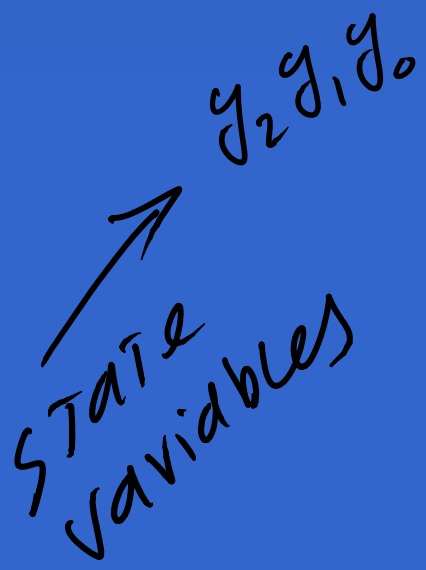
- Designing State Machines



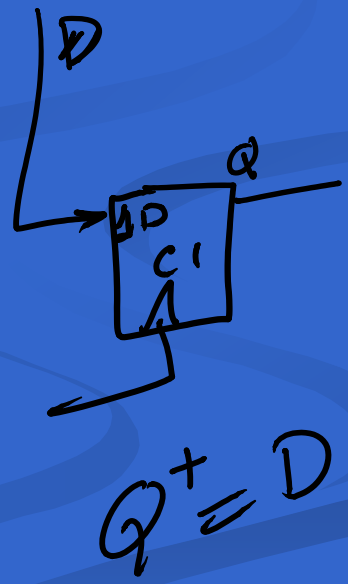
- **State Diagram for the 1011 Detector**

Sequential Circuit Design

- Designing State Machines



State	x = 0	x = 1	
000 A	A 000	001 B	0
001 B	C 010	001 B	0
010 C	A 000	011 D	0
011 D	C 010	100 E	0
100 E	C 010	001 B	1
	State$^+$		w



- **State Table of the 1011 Detector**

Sequential Circuit Design

- Designing State Machines

State	y_2	y_1	y_0
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0

State Assignment

- State Assignment

101
110
111
UNUSED STATES

Sequential Circuit Design

- Designing State Machines

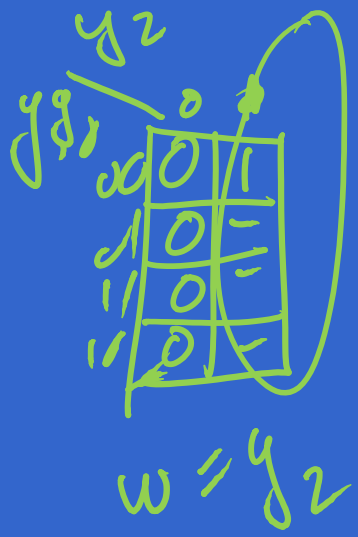
				x		
	y_2	y_1	y_0	0	1	
A:	0	0	0	0 0 0	0 0 1	0
B:	0	0	1	0 1 0	0 0 1	0
C:	0	1	0	0 0 0	0 1 1	0
D:	0	1	1	0 1 0	1 0 0	0
E:	1	0	0	0 1 0	0 0 1	1
				y_2^+ y_1^+ y_0^+		w

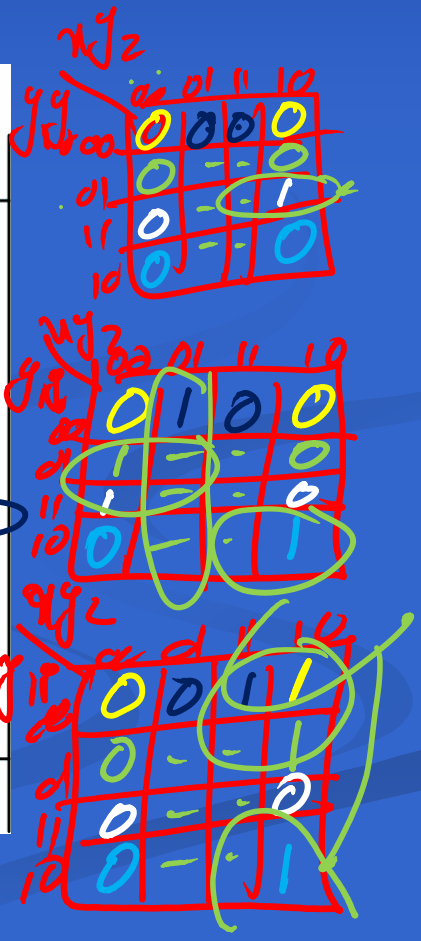
- Transition Table for the 1011 Detector

Sequential Circuit Design

- Designing State Machines

			x = 0	x = 1	
y_2	y_1	y_0			
0	0	0	0 0 0	0 0 1	0
0	0	1	0 1 0	0 0 1	0
0	1	0	0 0 0	0 1 1	0
0	1	1	0 1 0	1 0 0	0
1	0	0	0 1 0	0 0 1	1
1	0	1	- - -	- - -	-
1	1	0	- - -	- - -	-
1	1	1	- - -	- - -	-
			D_2 D_1 D_0		w

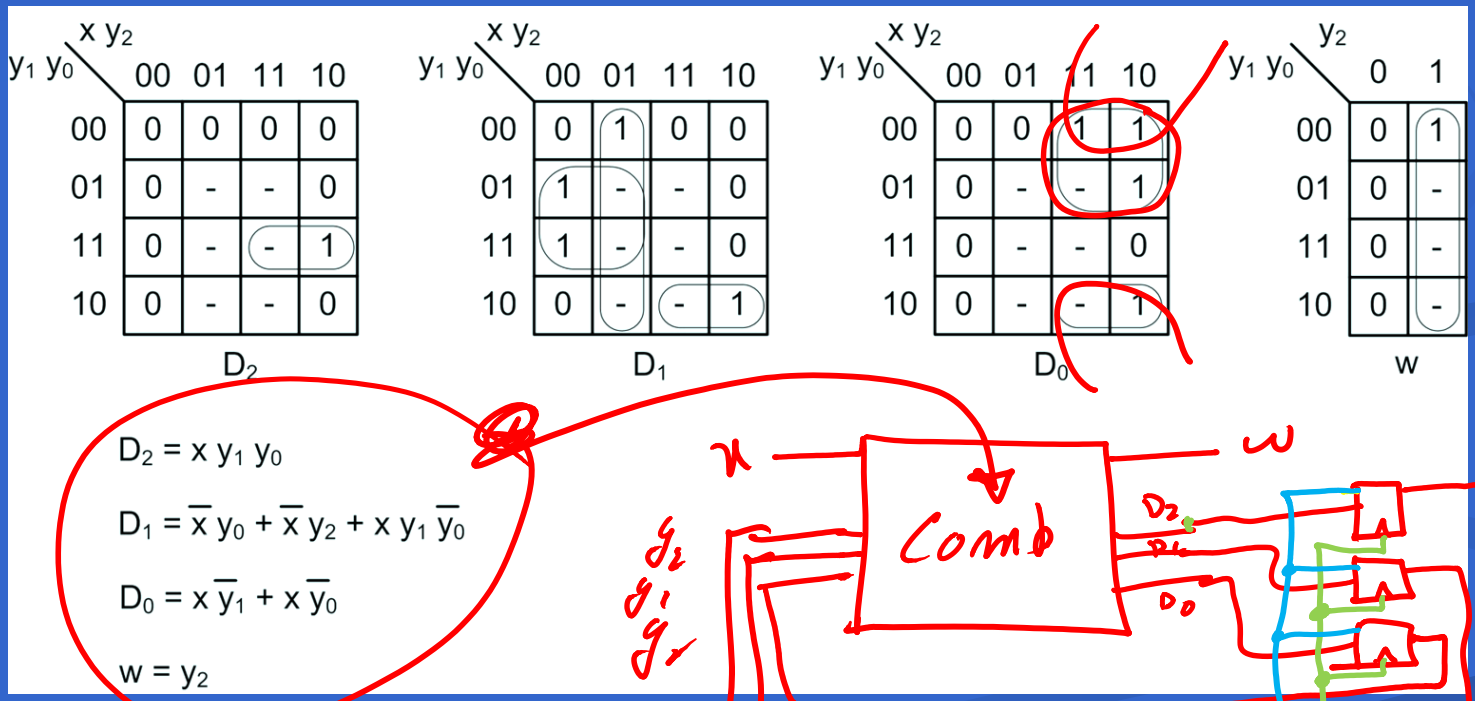




- **Flip-Flop Excitation Tables**

Sequential Circuit Design

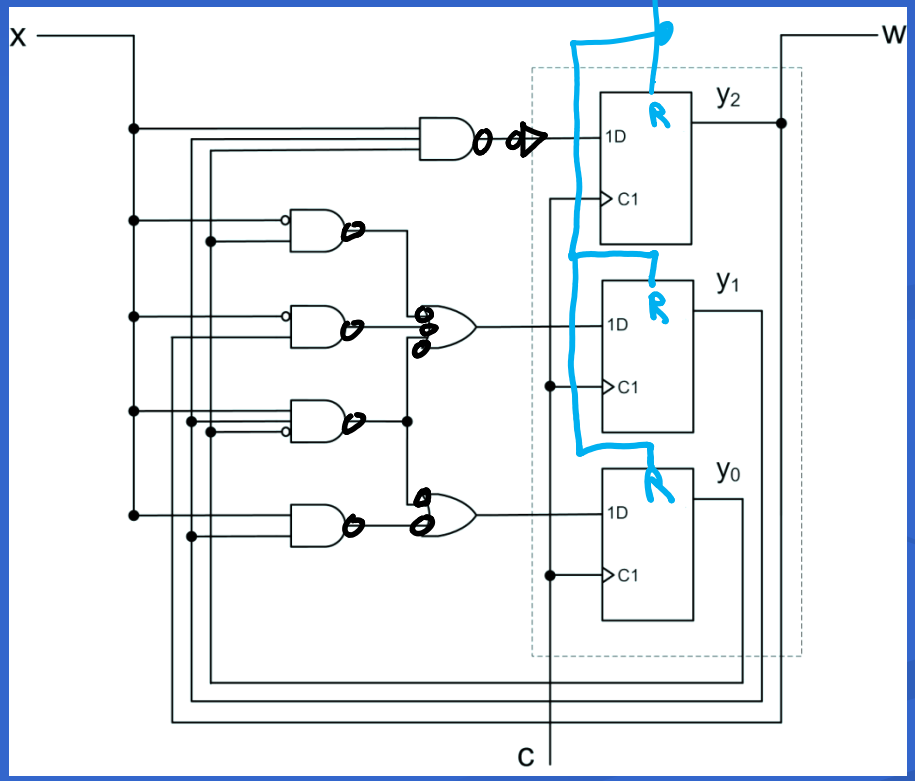
- Designing State Machines



- Implementing Combinational Part

Sequential Circuit Design

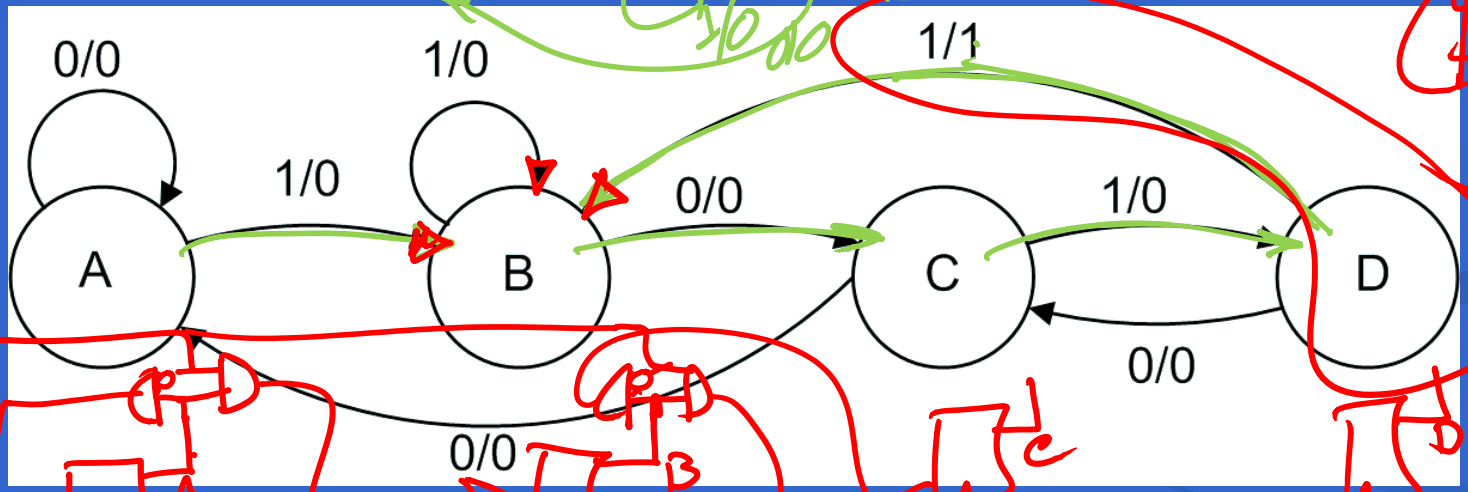
- Designing State Machines



- **Logic Block Diagram of the 1011 Detector**

Sequential Circuit Design

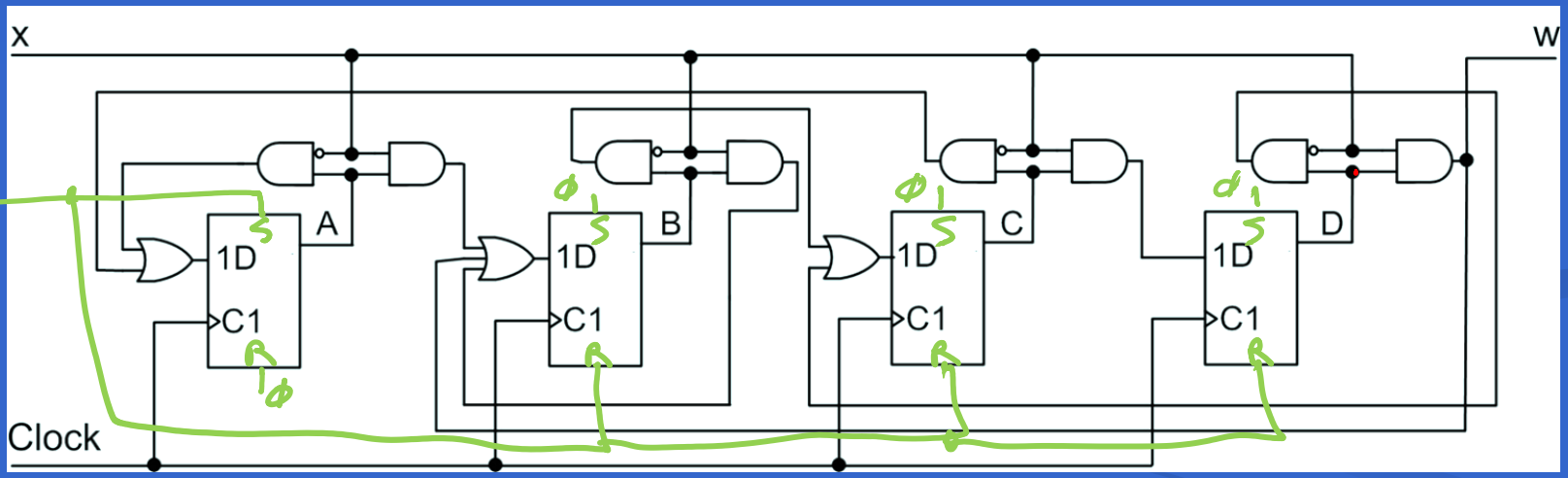
- Mealy and Moore Machines



Mealy State Diagram

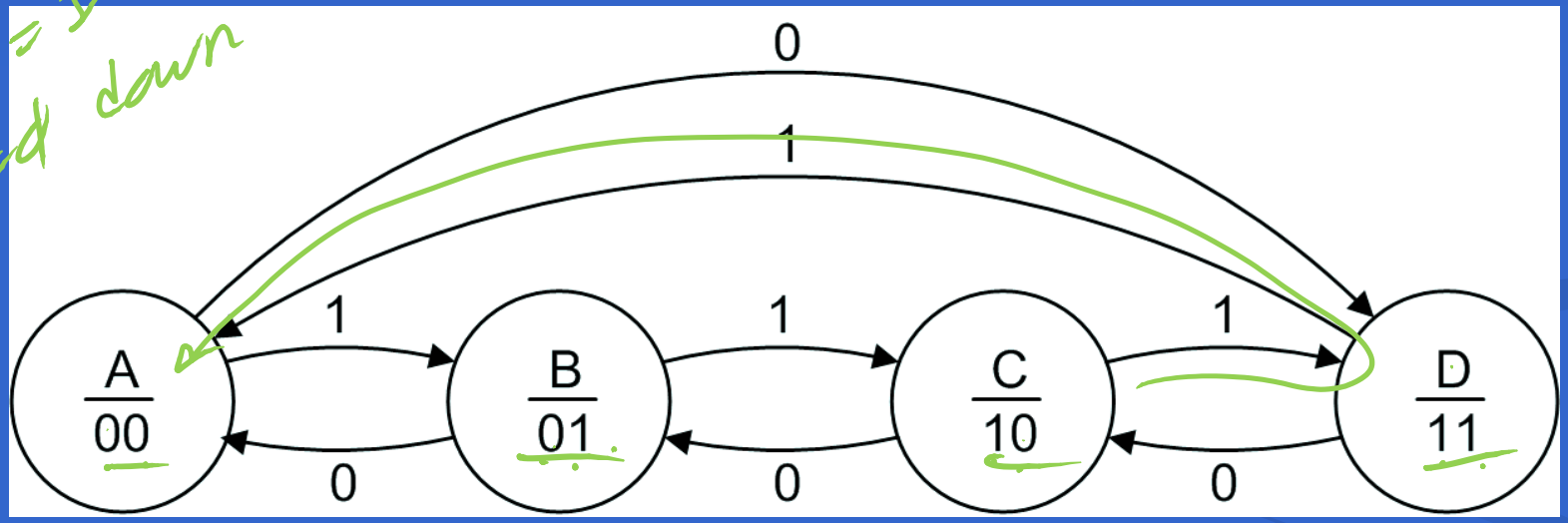
Sequential Circuit Design

- One-Hot Realization



Sequential Circuit Design

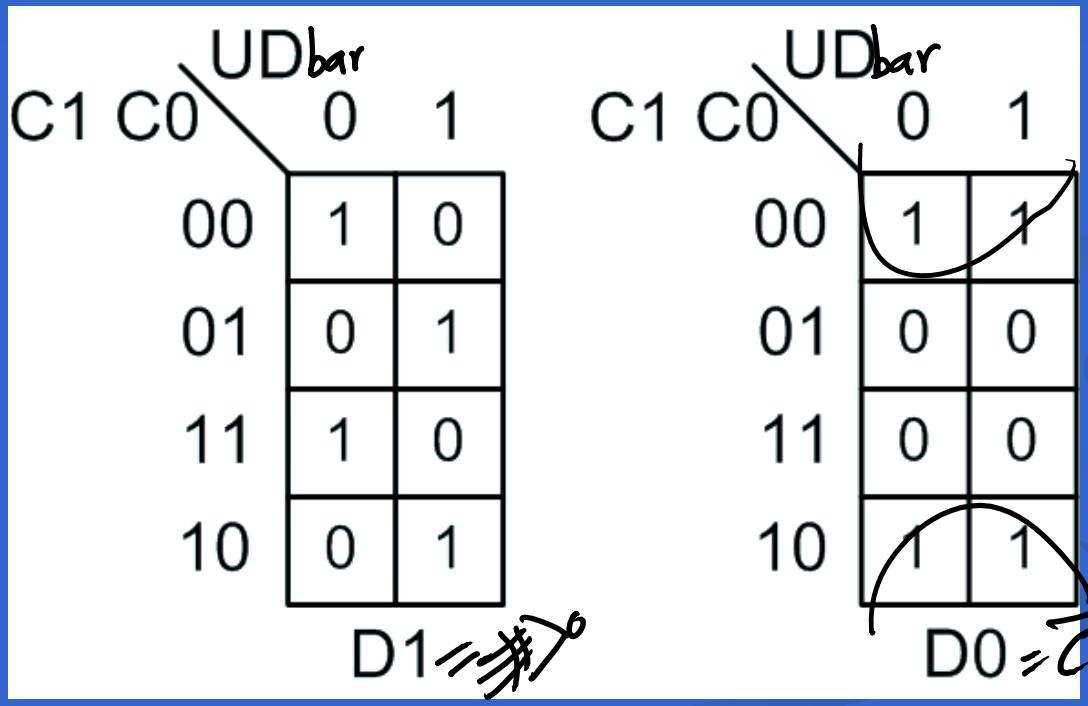
- Sequential Packages - Counter



- State Diagram of a 2-bit Counter

Sequential Circuit Design

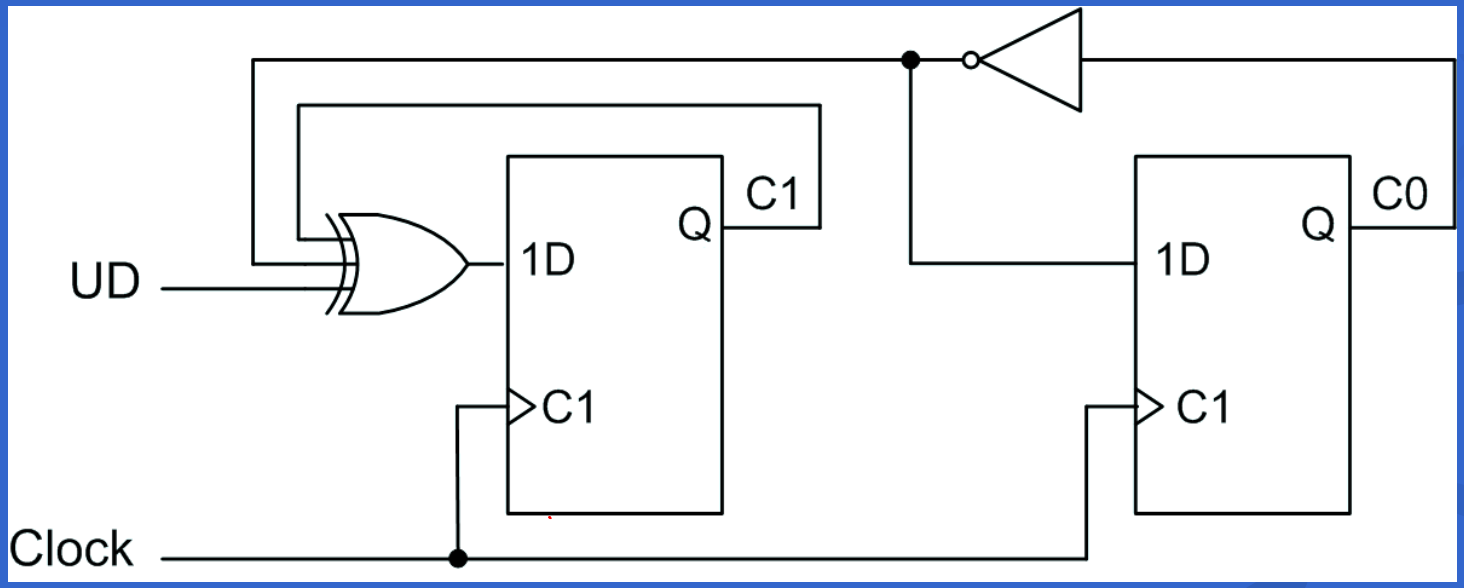
- Sequential Packages - Counter



- **Excitation K-maps for a 2-Bit Up-Down Counter**

Sequential Circuit Design

- Sequential Packages - Counter

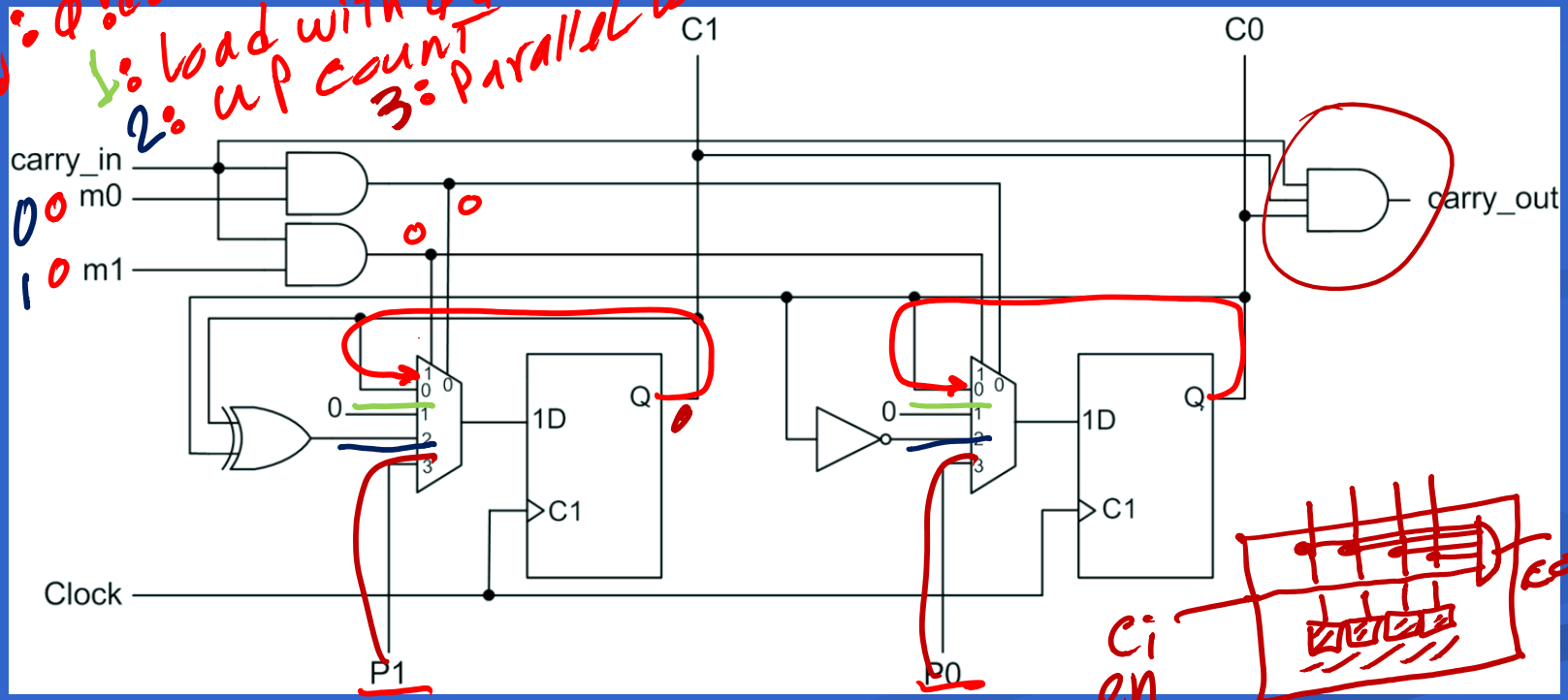


- A Two-Bit Up-Down Counter

binary

Sequential Circuit Design

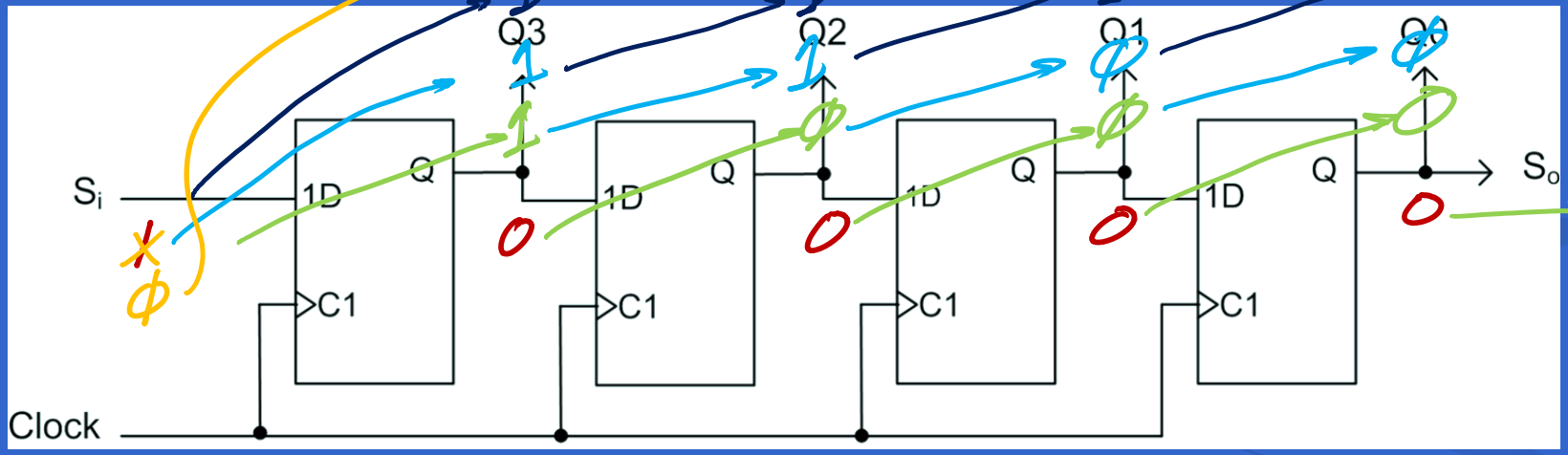
- Sequential Packages - Counter



- **Two-Bit Up-Counter with Added Features**

Sequential Circuit Design

- Sequential Packages – Shift Register



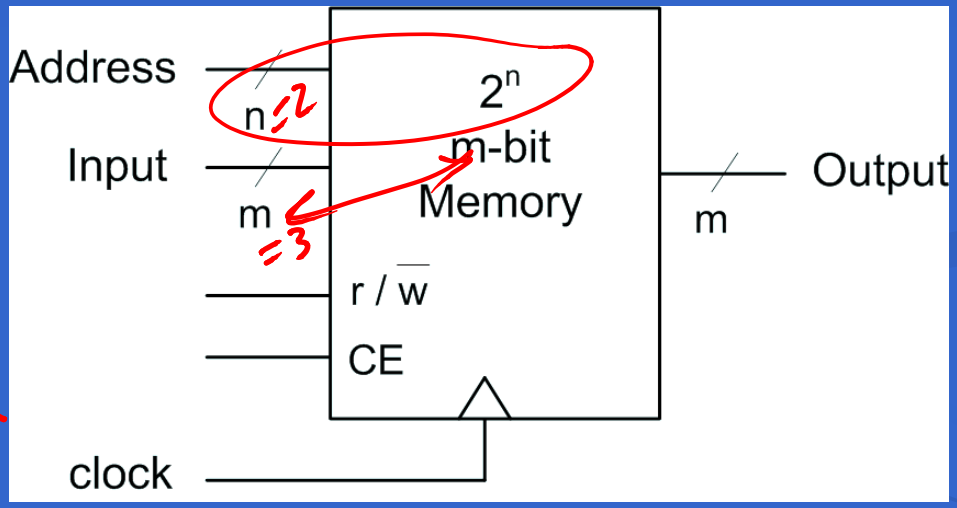
- A 4-bit Shift Register

Logic Design Concepts

- Number Systems
- Binary Arithmetic
- Basic Logic Gates and Structures
- Designing Combinational Circuits
- Storage Elements
- Sequential Circuit Design
- Memories
- Bidirectional IO
- A Comprehensive Example: Serial Adder
- Summary

Memories

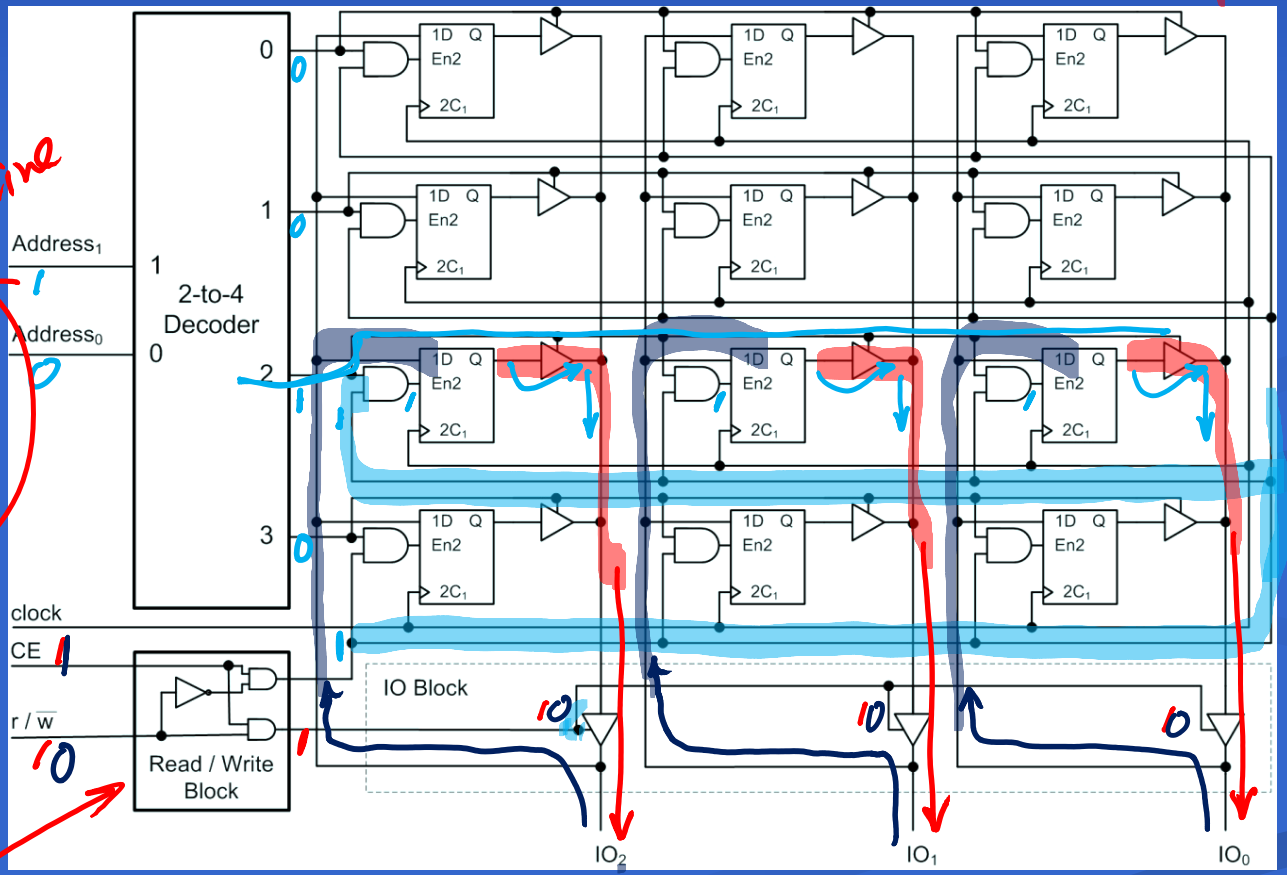
- Static RAM Structure



- A 2n m-bit Memory

Memories

- Static RAM Structure



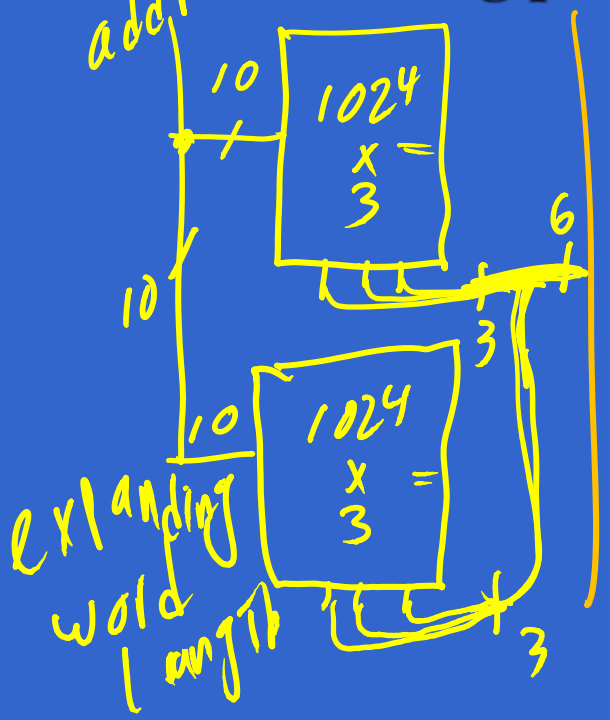
- **SRAM Structure**

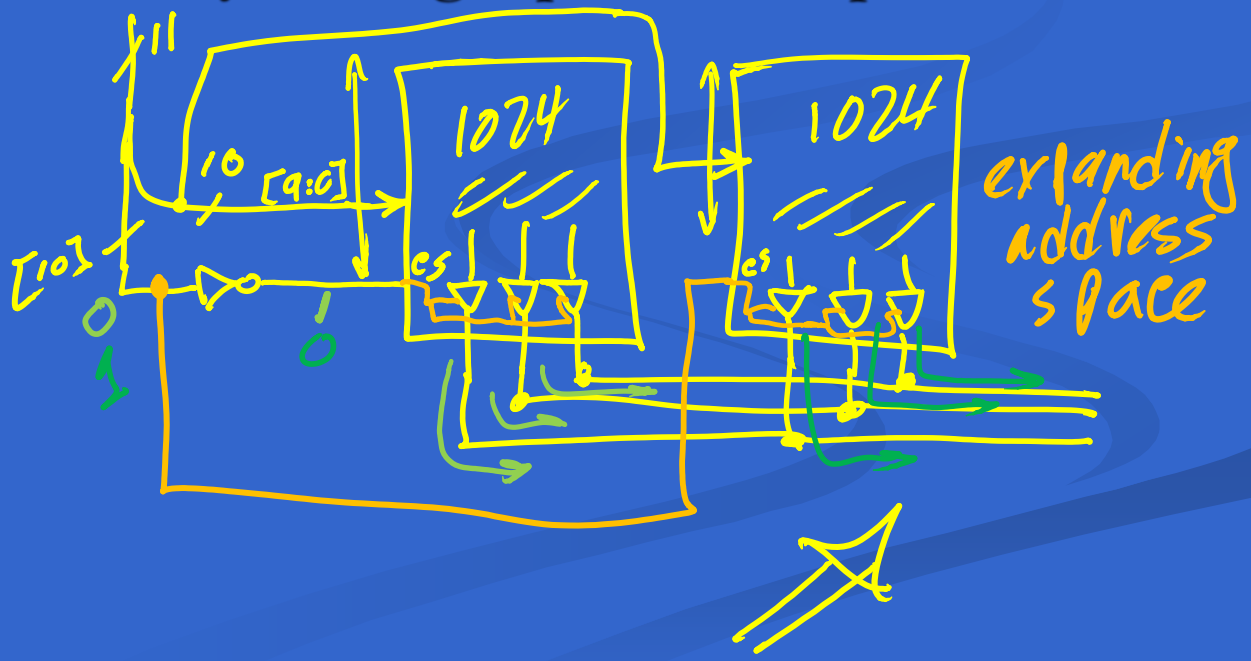
Logic Design Concepts

- Number Systems
- Binary Arithmetic
- Basic Logic Gates and Structures
- Designing Combinational Circuits
- Storage Elements
- Sequential Circuit Design
- Memories
- Bidirectional IO
- A Comprehensive Example: Serial Adder
- Summary

Bidirectional IO

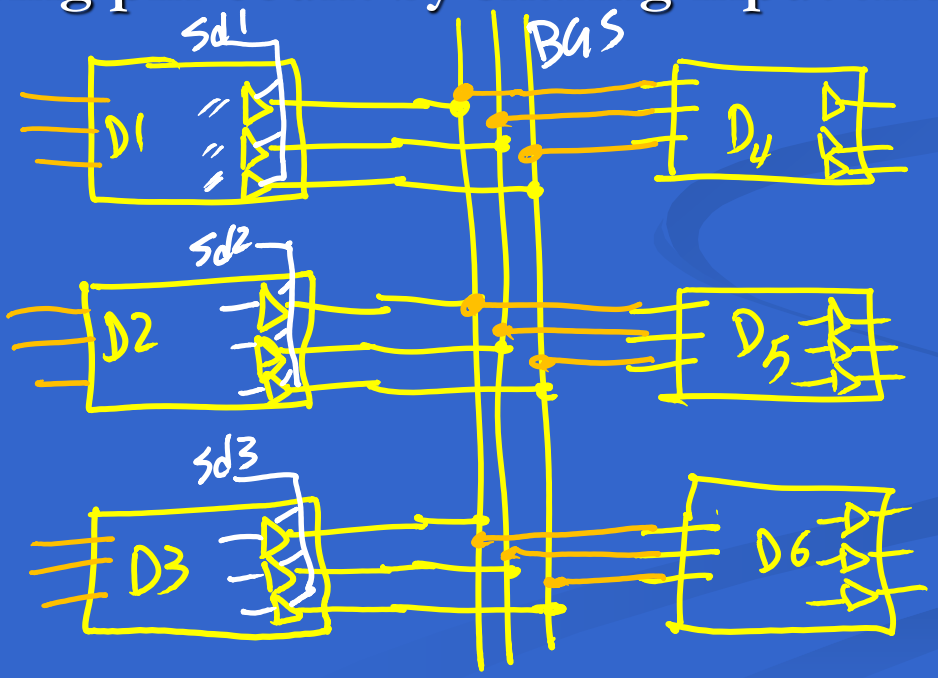
- Bidirectional inout (input output) lines are useful for:
  - Cascading memories (RAM, ROM, etc)
  - Bus source selection
  - Reducing pin-count by sharing input and output lines





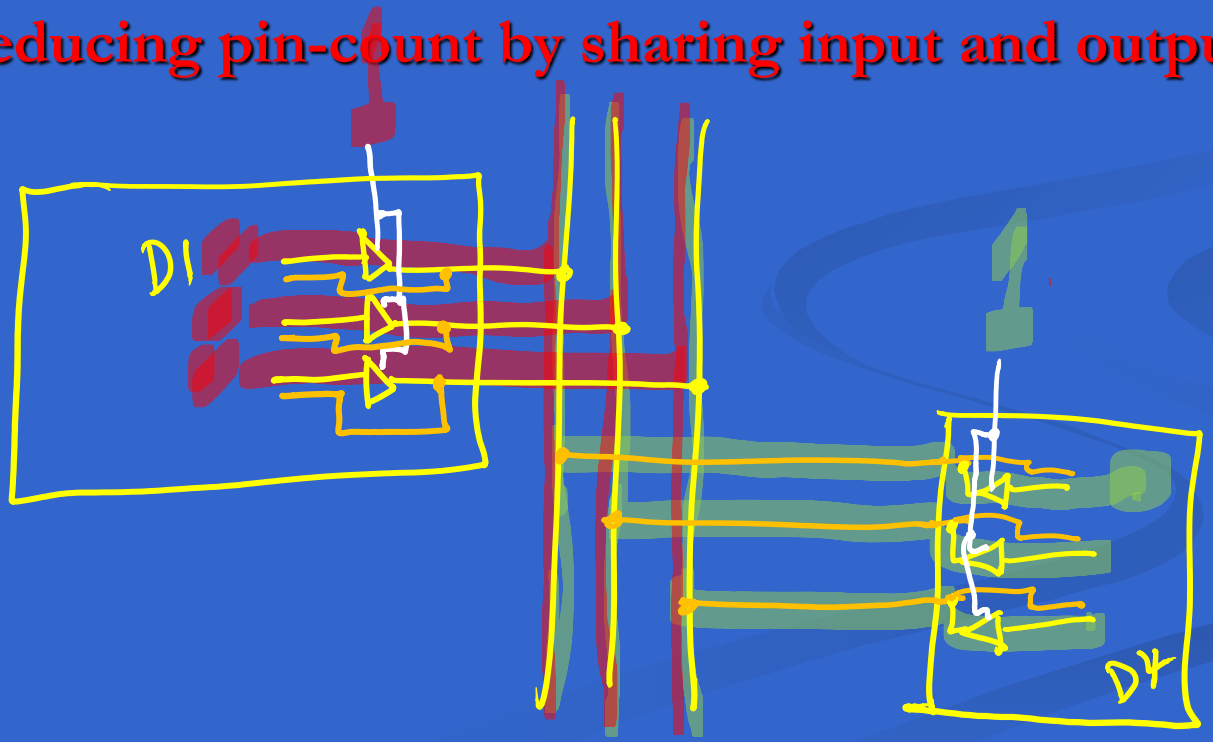
Bidirectional IO

- Bidirectional inout (input output) lines are useful for:
  - Cascading memories (RAM, ROM, etc)
  - Bus source selection
  - Reducing pin-count by sharing input and output lines



Bidirectional IO

- Bidirectional inout (input output) lines are useful for:
  - Cascading memories (RAM, ROM, etc)
  - Bus source selection
  - Reducing pin-count by sharing input and output lines

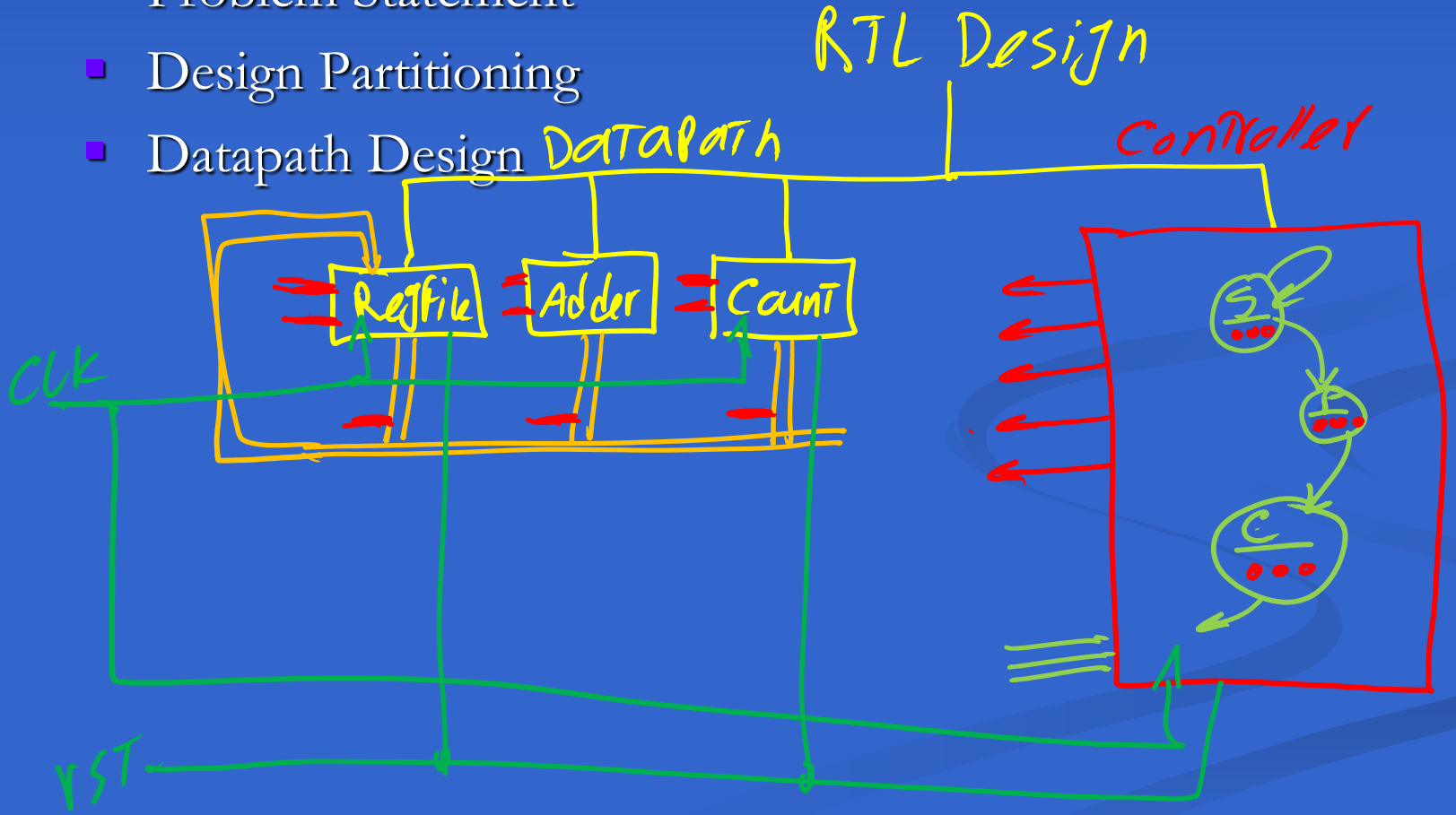


Logic Design Concepts

- Number Systems
- Binary Arithmetic
- Basic Logic Gates and Structures
- Designing Combinational Circuits
- Storage Elements
- Sequential Circuit Design
- Memories
- Bidirectional IO
- A Comprehensive Example: Serial Adder
- Summary

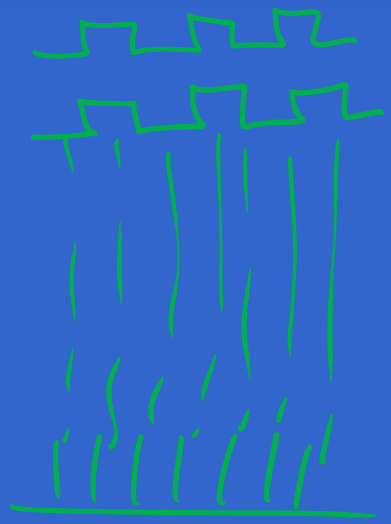
A Comprehensive Example: Serial Adder

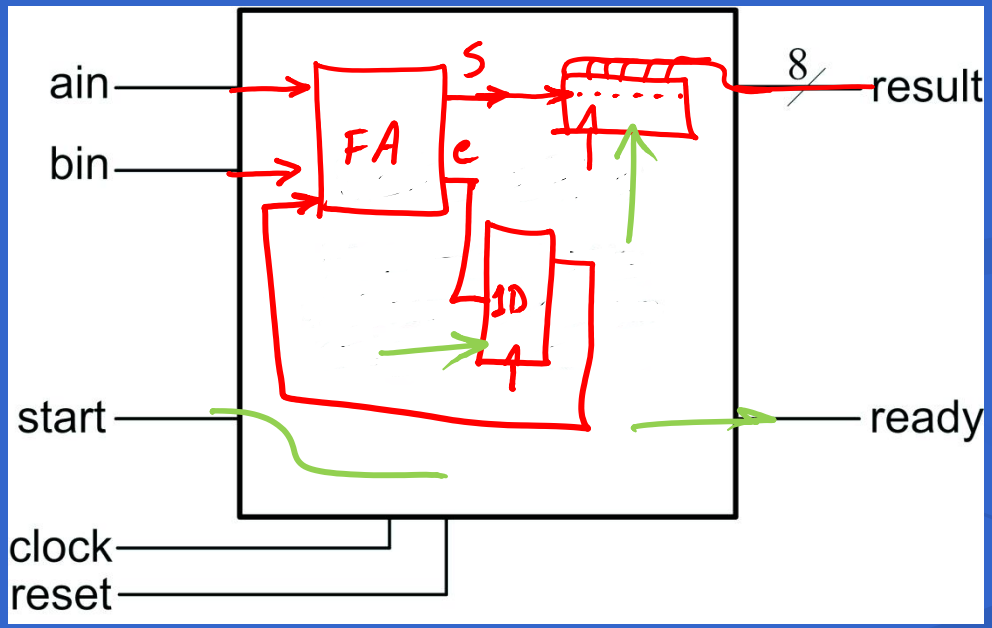
- Problem Statement
- Design Partitioning
- Datapath Design



A Comprehensive Example: Serial Adder

- Problem Statement

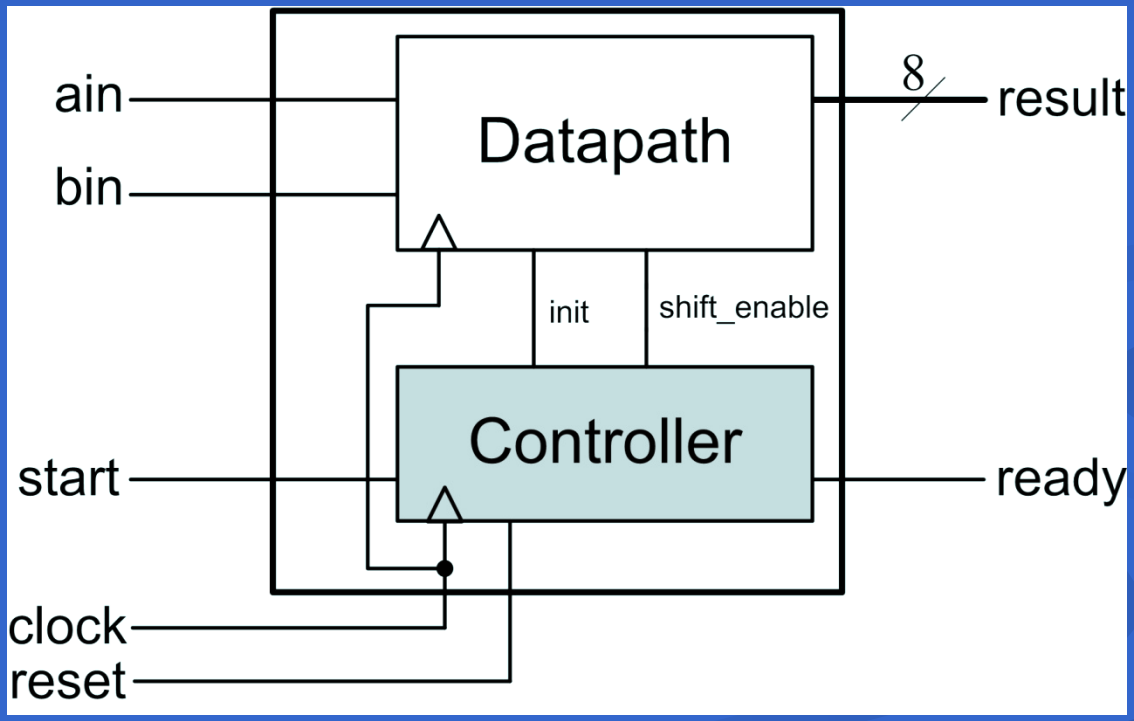




- **Serial Adder Block Diagram**

A Comprehensive Example: Serial Adder

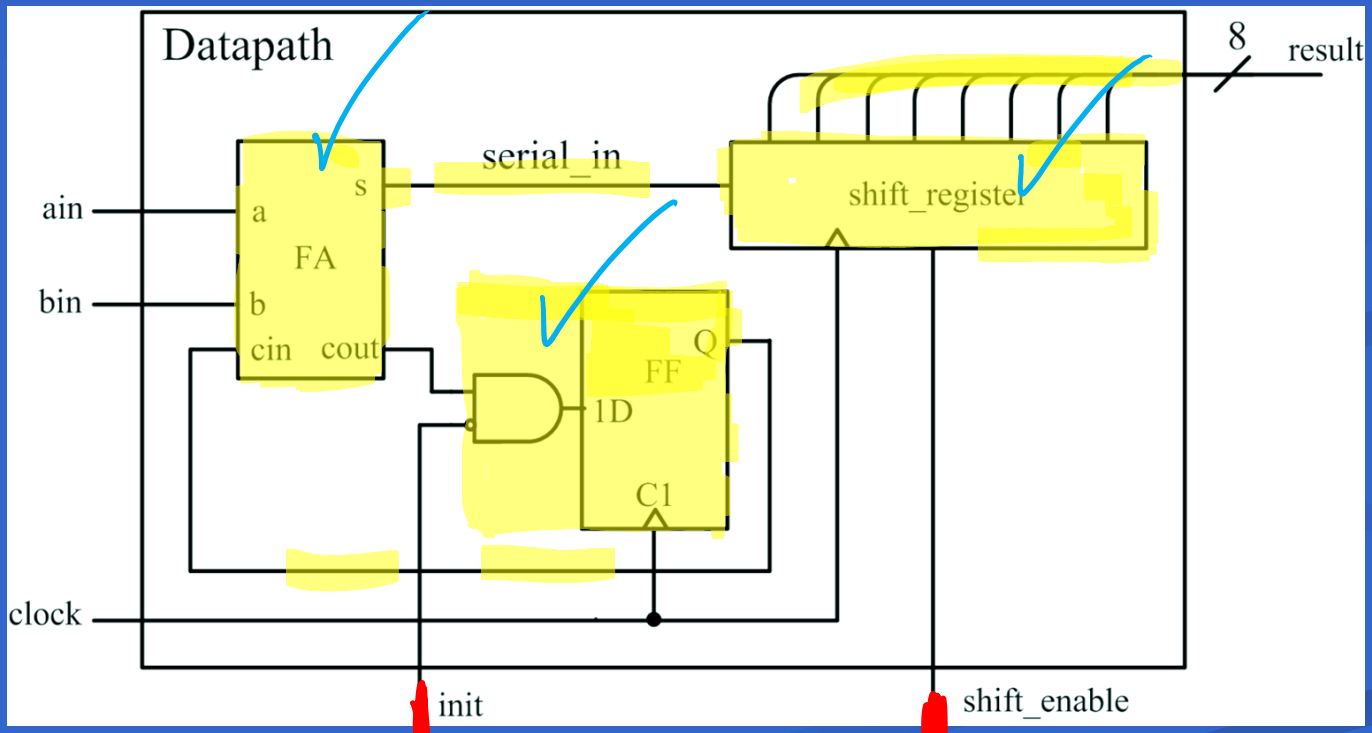
- Design Partitioning



- **Serial Adder Data/Control Partitioning**

A Comprehensive Example: Serial Adder

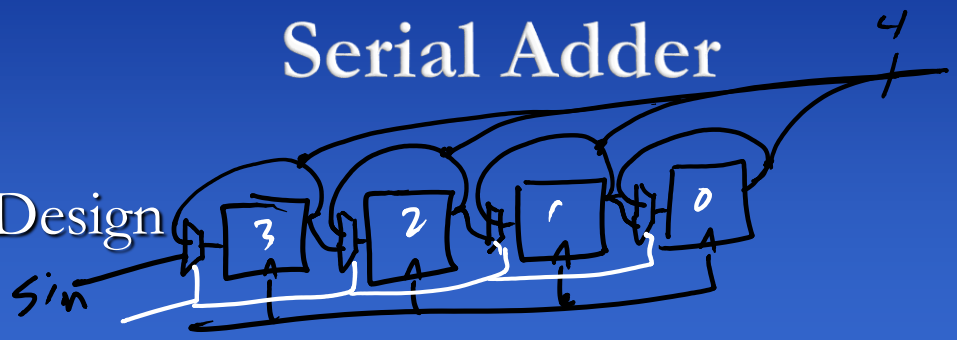
- Datapath Design

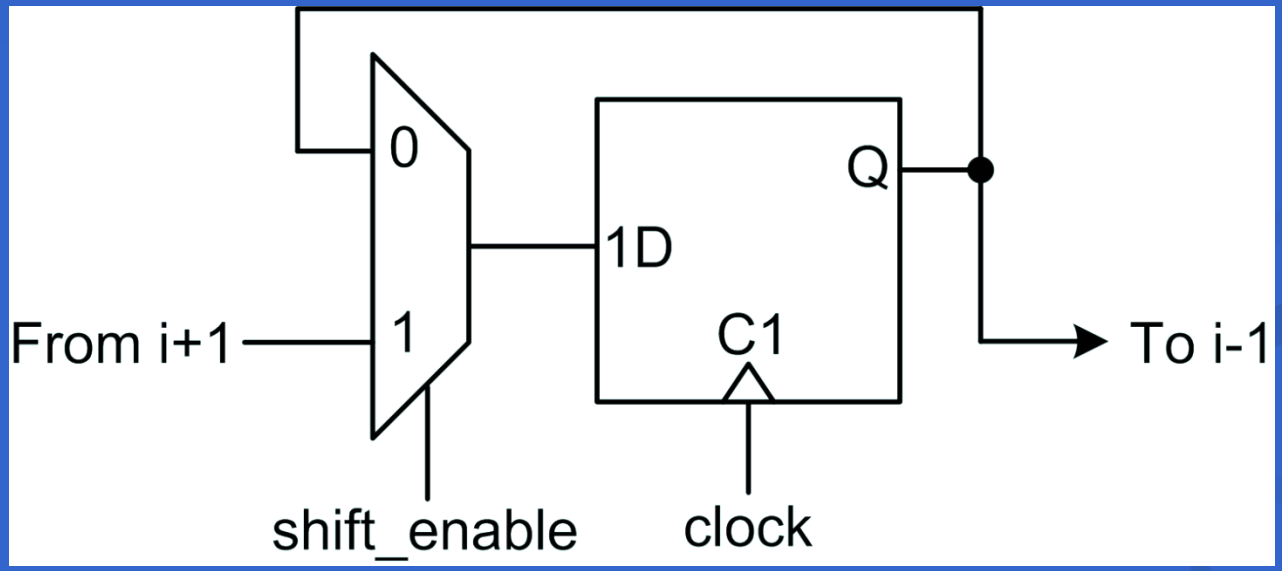


Serial Adder Datapath

A Comprehensive Example: Serial Adder

- Datapath Design

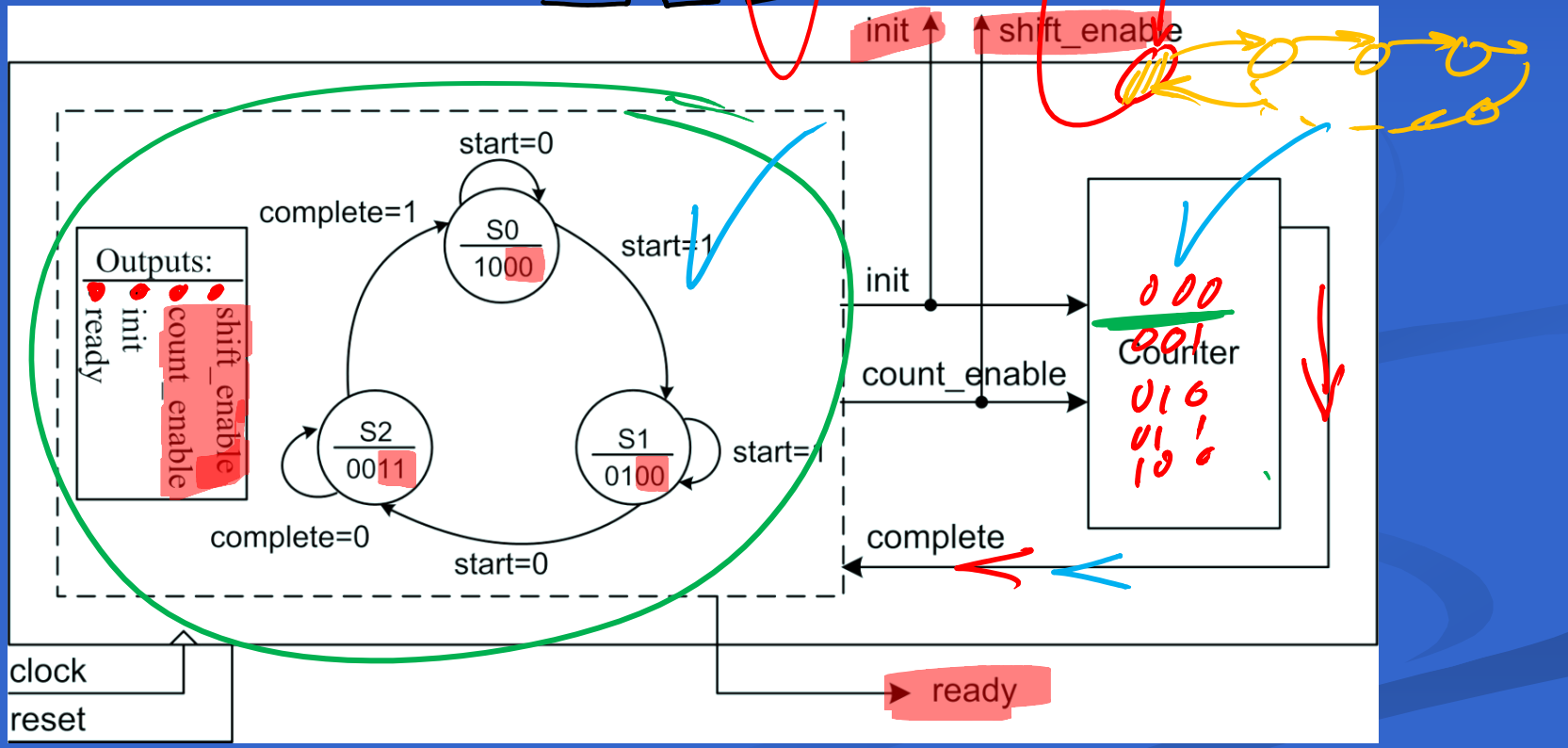




- **Cascadable Shift-register Bit with an Enable Input**

A Comprehensive Example: Serial Adder

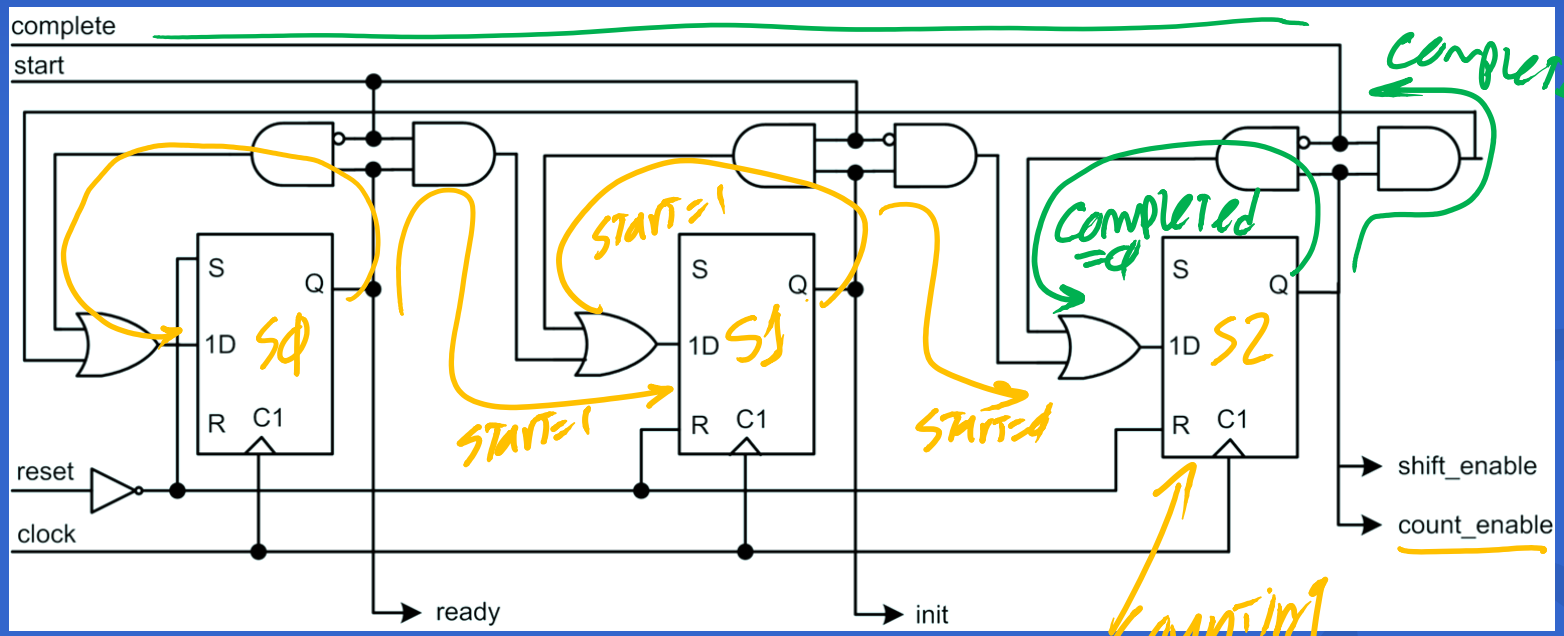
- Serial Adder Controller



- **Serial Adder Controller Diagram**

A Comprehensive Example: Serial Adder

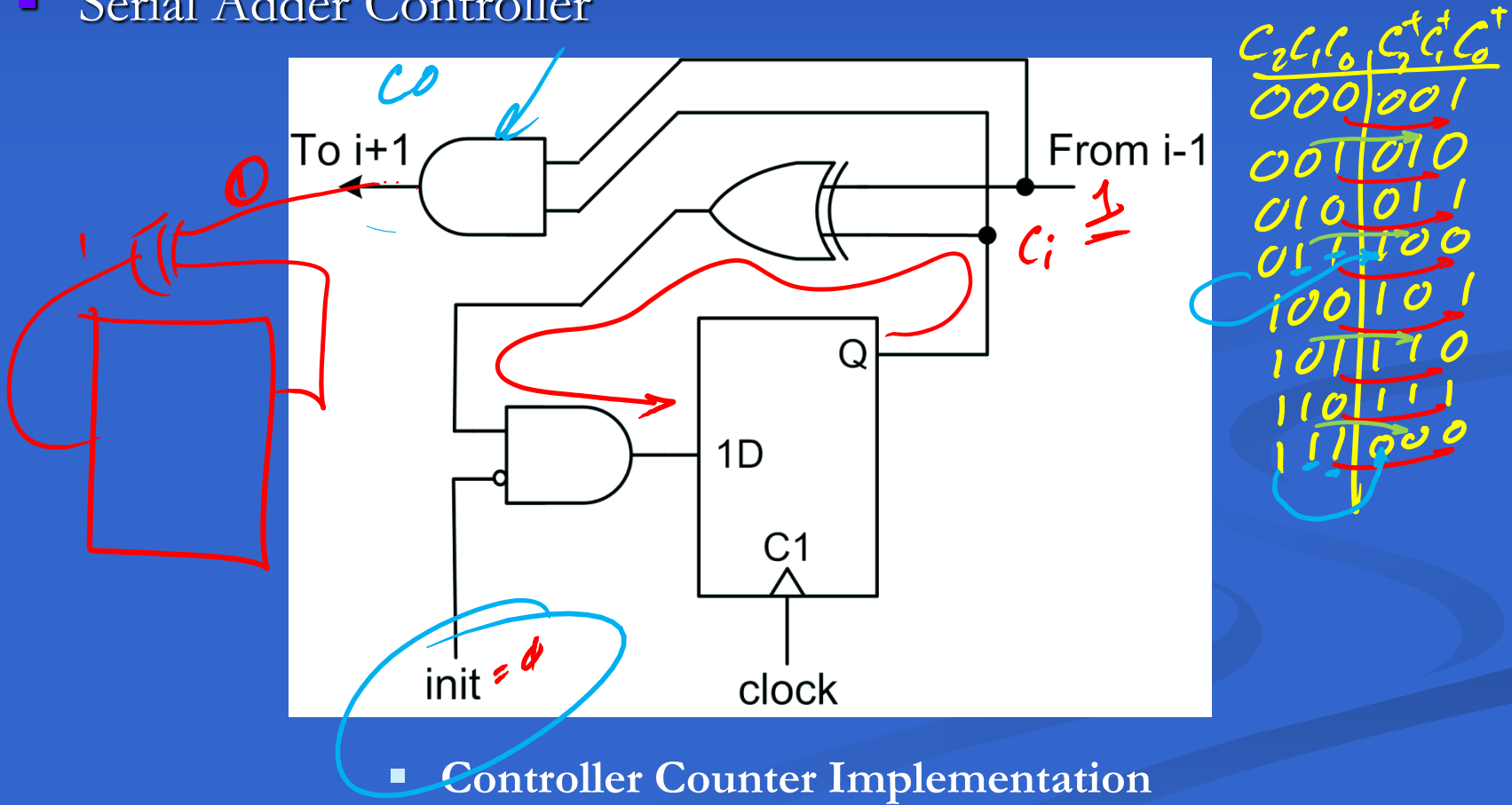
- Serial Adder Controller



- Controller State Machine Implementation

A Comprehensive Example: Serial Adder

- Serial Adder Controller



- Controller Counter Implementation

Logic Design Concepts

- Number Systems
- Binary Arithmetic
- Basic Logic Gates and Structures
- Designing Combinational Circuits
- Storage Elements
- Sequential Circuit Design
- Memories
- Bidirectional IO
- A Comprehensive Example: Serial Adder
- Summary

Summary

- Summary
- Overview
- Combinational
- Sequential
- RT Level
- Complete design flow