

Logic Design

A Review

Boolean Algebra

- ❖ Two Values: zero and one
- ❖ Three Basic Functions: And, Or, Not
- ❖ Any Boolean Function Can be Constructed from These Three

| And | 0 | 1 |
|------------|----------|----------|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

| Or | 0 | 1 |
|-----------|----------|----------|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

| Not | |
|------------|----------|
| 0 | 1 |
| 1 | 0 |



Algebraic Laws

| Classification | Law |
|------------------------|--|
| Identity | $a1=1a=a$ $a+0=0+a=a$ |
| Dominance | $a0=0a=0$ $1+a=a+1=1$ |
| Commutativity | $a+b=b+a$ $ab=ba$ |
| Associativity | $a(bc)=(ab)c$ $a+(b+c)=(a+b)+c$ |
| Distributive | $a(b+c)=ab+ac$ $a+bc=(a+b)(a+c)$ |
| Demorgan's Laws | $(a + b)' = a'b'$ $(ab)' = a' + b'$ |



Boolean Expressions

- ❖ Addition represents OR
- ❖ Multiplication represents AND
- ❖ Not is represented by a prime a' or an overbar \bar{a}
- ❖ Examples:
 - ❖ $s = a'bc + ab'c + abc' + a'b'c'$
 - ❖ $q = ab + bc + ac + abc$

Superfluous Terms

- ❖ The following Two Equations Represent The Same Function.

$$q = ab + bc + ac + abc$$

$$q = ab + bc + ac$$

| a | b | c | q |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |



Prime Implicants

- ❖ A Prime Implicant is a Product of Variables or Their Complements, eg. $ab'cd'$
- ❖ If a Prime Implicant has the Value 1, then the Function has the Value 1
- ❖ A Minimal Equation is a Sum of Prime Implicants



Minimization and Minterms

- ❖ Minimization Reduces the Size and Number of Prime Implicants
- ❖ A MinTerm is a Prime Implicant with the Maximum Number of Variables
- ❖ For a 3-input Function $a'bc$ is a MinTerm, while ab is not.
- ❖ Prime Implicants can be Combined to Eliminate Variables, $abc' + abc = ab$

Minimization with Maps

❖ A Karnaugh Map

| | 00 | 01 | 11 | 10 |
|---|----|----|----|----|
| 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 |

A {

C {

B {



Procedure

- ❖ Select Regions Containing All 1's
- ❖ Regions should be as Large as Possible
- ❖ Regions must contain 2^k cells
- ❖ Regions should overlap as little as possible
- ❖ The complete set of regions must contain all 1's in the map



Procedure 2

- ❖ Top and Bottom of Map are Contiguous
- ❖ Left and Right of Map are Contiguous
- ❖ Regions represent Prime Implicants
- ❖ Use Variable name guides to construct equation
 - Completely inside the region of a variable means prime implicant contains variable
 - Completely outside the region of a variable means prime implicant contains negation



Applied to Previous Map

| | C | | B | |
|---|----|----|----|----|
| | 00 | 01 | 11 | 10 |
| A | 0 | 0 | 0 | 1 |
| A | 1 | 0 | 0 | 0 |

The table shows a Karnaugh map with variables A, B, and C. The columns are labeled C (00, 01, 11, 10) and the rows are labeled A (0, 1). The cells containing '1' are at (A=0, C=00) and (A=0, C=10). Hand-drawn black circles group these two '1's, and a bracket labeled 'A' is placed to the left of the first row.

$$q = c'b' + c'a'$$



A 4-Variable Karnaugh Map

| | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 1 | 1 |

Diagram annotations:

- Variable **D** is indicated by a bracket above the columns 11 and 10.
- Variable **C** is indicated by a bracket above the columns 11 and 10.
- Variable **B** is indicated by a bracket to the left of the rows 01 and 11.
- Variable **A** is indicated by a bracket to the left of the rows 11 and 10.

First Minimization

| | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 1 | 1 |

Diagram illustrating the first minimization step of a Karnaugh map. The map is a 4x4 grid with columns labeled 00, 01, 11, and 10, and rows labeled 00, 01, 11, and 10. The cells containing 1s are circled, indicating the first minimization step. The circled 1s are located at (01, 01), (01, 11), (11, 01), (11, 11), (10, 01), (10, 11), (10, 10), and (10, 10). The circled 1s are grouped into two groups: Group D (top row of 1s) and Group C (right column of 1s). Group A (left column of 1s) and Group B (middle column of 1s) are also indicated by brackets on the left side of the map.



Second Minimization

| | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 1 | 1 |

Diagram annotations:

- Group D: Brackets above columns 11 and 10.
- Group C: Brackets above columns 11 and 10, overlapping with D.
- Group B: Brackets to the left of rows 01 and 11.
- Group A: Brackets to the left of rows 11 and 10.
- Groupings: Circles around (01,1), (11,1), (10,1), (11,1), and (10,1). An oval around (11,1) and (10,1).



Minimal Forms for Previous Slides:

- ❖ $ab'd + bc'd + a'bc + acd'$

- ❖ $ac'd + a'bd + bcd' + ab'c$

- ❖ Moral: A Boolean Function May Have Several Different Minimal Forms

- ❖ Karnaugh Maps are Ineffective for Functions with More than Six Inputs.



Quine McClusky Minimization

- ❖ Amenable to Machine Implementation
- ❖ Applicable to Circuits with an Arbitrary Number of Inputs
- ❖ Effective Procedure for Finding Prime Implicants, but ...
- ❖ Can Require an Exponential Amount of Time for Some Circuits



Quine-McClusky Procedure

- ❖ Start with The Function Truth Table
- ❖ Extract All Input Combinations that Produce a TRUE Output (MinTerms)
- ❖ Group All MinTerms by The Number of Ones They Contain
- ❖ Combine Minterms from Adjacent Groups



More Quine-McClusky

- ❖ Two Min-Terms Combine If They Differ by Only One Bit
- ❖ The Combined MinTerm has an x in the Differing Position
- ❖ Create New Groups From Combined Min-Terms
- ❖ Each Member of A New Group Must Have the Same Number of 1's and x's



Yet More Quine-McClusky

- ❖ Each Member of A Group Must Have x's in The Same Position.
- ❖ Combine Members of the New Groups To Create More New Groups
- ❖ Combined Terms Must Differ By One Bit, and Have x's in the Same Positions
- ❖ Combine as Much as Possible
- ❖ Select Prime Implicants to “Cover” All Ones in the Function



Quine-McClusky Example 1

Numbers in Parentheses are Truth-Table Positions.

| | |
|----------------|-----------------|
| 0011(3) | 1100(12) |
|----------------|-----------------|

| | | |
|----------------|-----------------|-----------------|
| 0111(7) | 1011(11) | 1101(13) |
|----------------|-----------------|-----------------|

| |
|-----------------|
| 1110(14) |
|-----------------|

| |
|-----------------|
| 1111(15) |
|-----------------|



Quine-McClusky Example 2

New Groups After Combining MinTerms

| | |
|---------------|---------------|
| $0x11(3,7)$ | $110x(12,13)$ |
| $1x11(11,15)$ | $111x(14,15)$ |
| $x011(3,11)$ | $11x0(12,14)$ |
| $x111(7,15)$ | $11x1(13,15)$ |

Quine-McClusky Example 3

The Final Two Groups

Note That These Two Elements Cover
All Truth-Table Positions

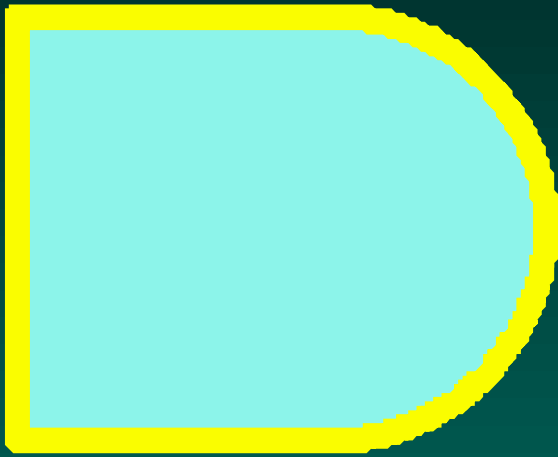
$xx11(3,7,11,15)$

$11xx(12,13,14,15)$

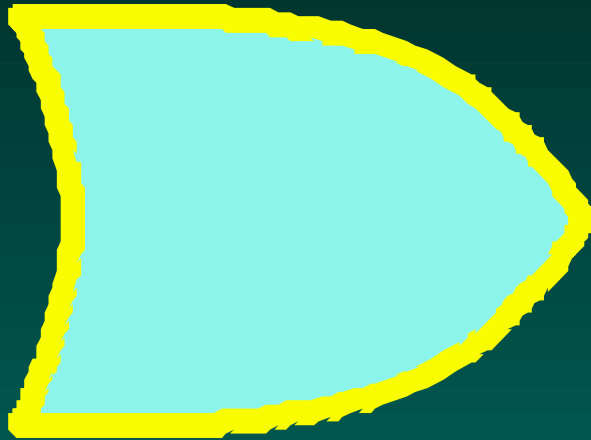
Quine-McClusky Example 4

- ❖ Each Group Element Represents a Prime Implicant
- ❖ It is Necessary to Select Group Elements to Cover All Truth-Table Positions.
- ❖ In This Case, $ab+cd$ is the Minimal Formula.
- ❖ In General, Selecting a Minimal Number of Prime Implicants is NP-Complete

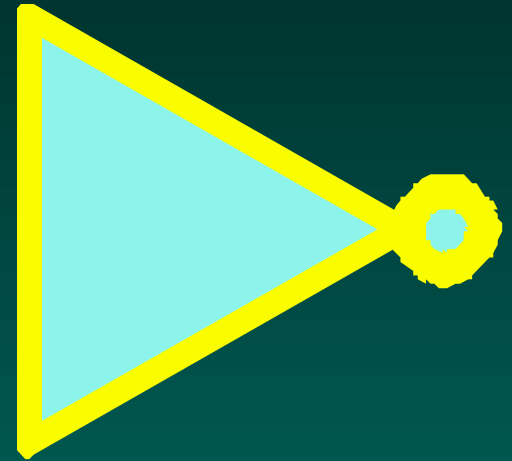
Basic Logic Symbols



And



Or

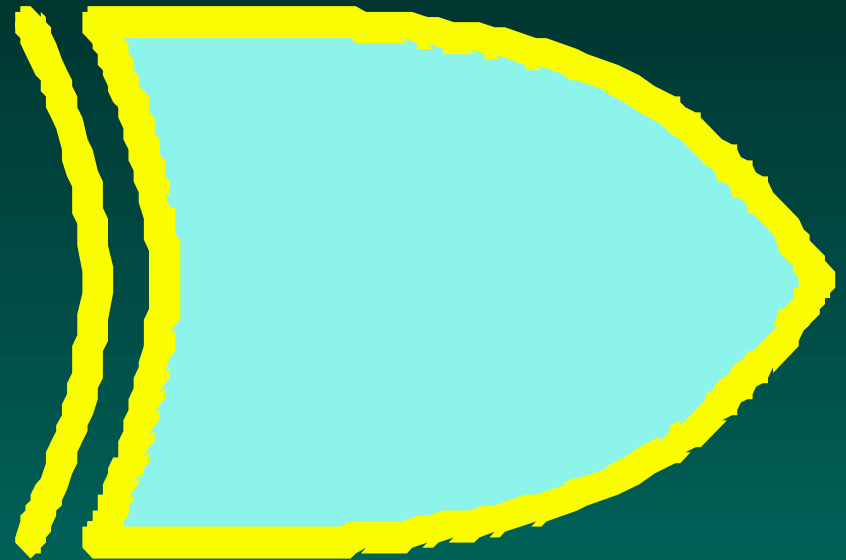


Not



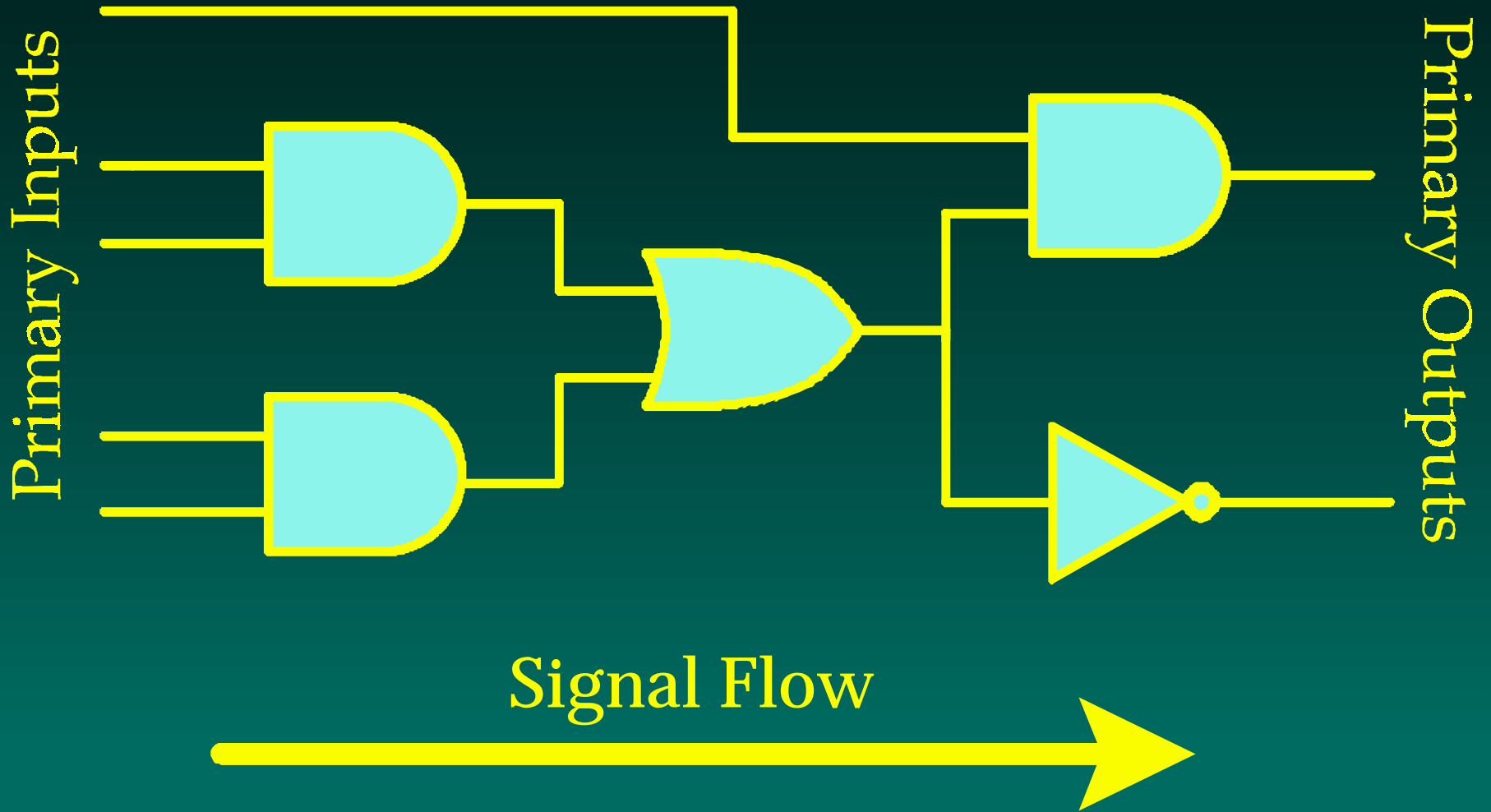
The Exclusive Or Function

| Xor | 0 | 1 |
|------------|----------|----------|
| 0 | 0 | 1 |
| 1 | 1 | 0 |



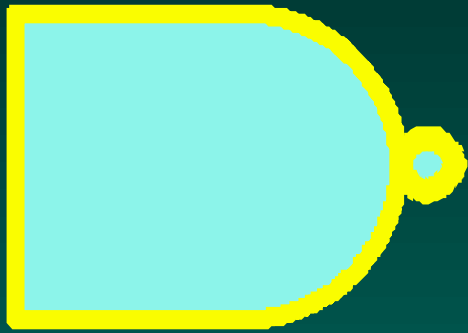


A Simple Logic Diagram

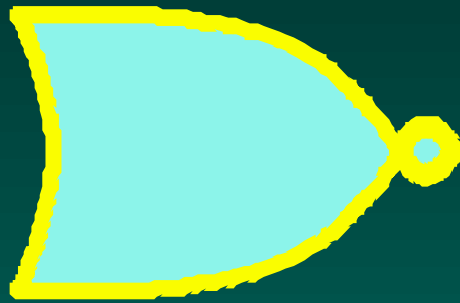




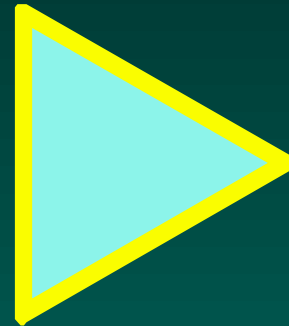
Additional Logic Symbols



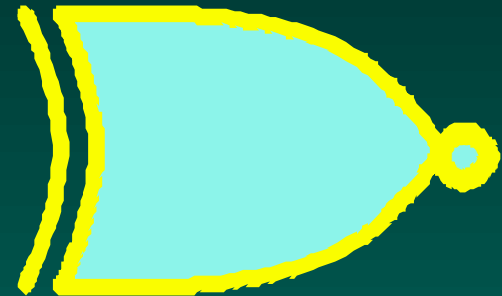
Nand



Nor



Buffer



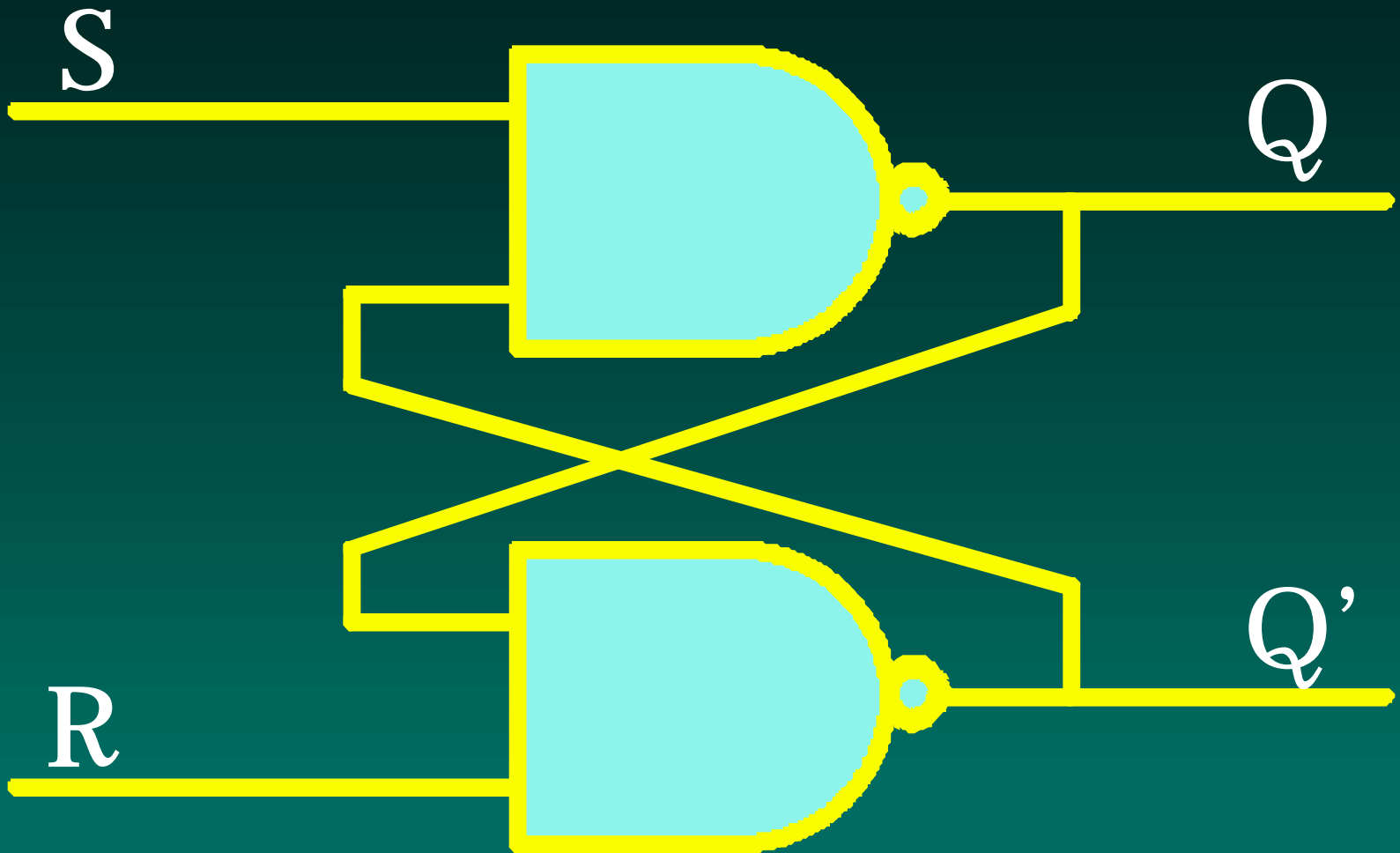
Xnor



Sequential Logic

- ❖ Contains Memory Elements
- ❖ Memory is Provided by Feedback
- ❖ Circuit diagrams generally have implicit or explicit cycles
- ❖ Two Styles: Synchronous and Asynchronous

An RS Flip-Flop



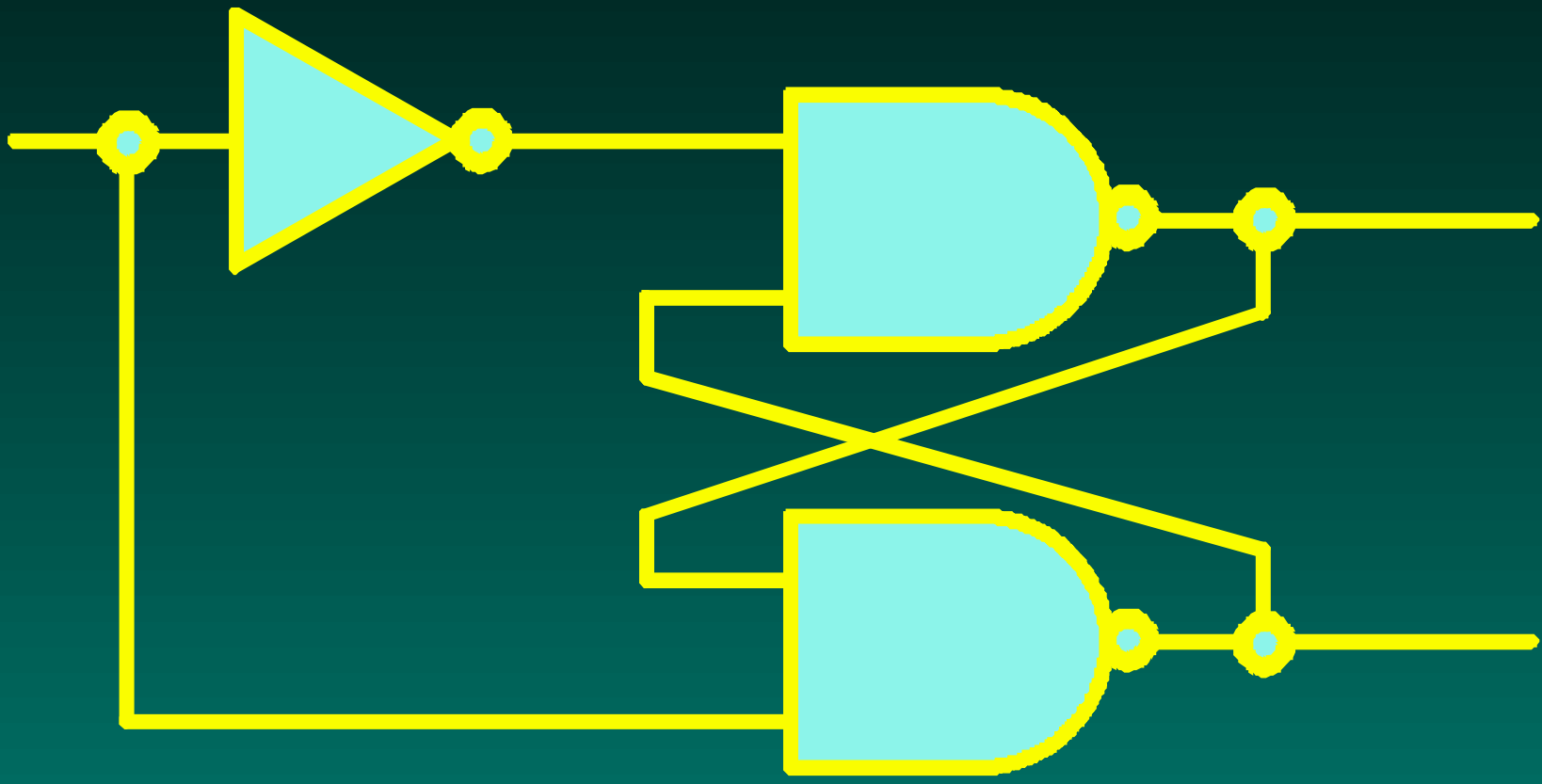
RS Characteristics

- ❖ If $S=0$ and $R=1$, Q is set to 1, and Q' is reset to 0
- ❖ If $R=0$ and $S=1$, Q is reset to 0, and Q' is set to 1
- ❖ If $S=1$ and $R=1$, Q and Q' maintain their previous state.
- ❖ If $S=0$ and $R=0$, a transition to $S=1$, $R=1$ will cause oscillation.

Instability

- ❖ RS flip-flops can become unstable if both R and S are set to zero.
- ❖ All Sequential elements are fundamentally unstable under certain conditions
 - Invalid Transitions
 - Transitions too close together
 - Transitions at the wrong time

D Flip-Flops



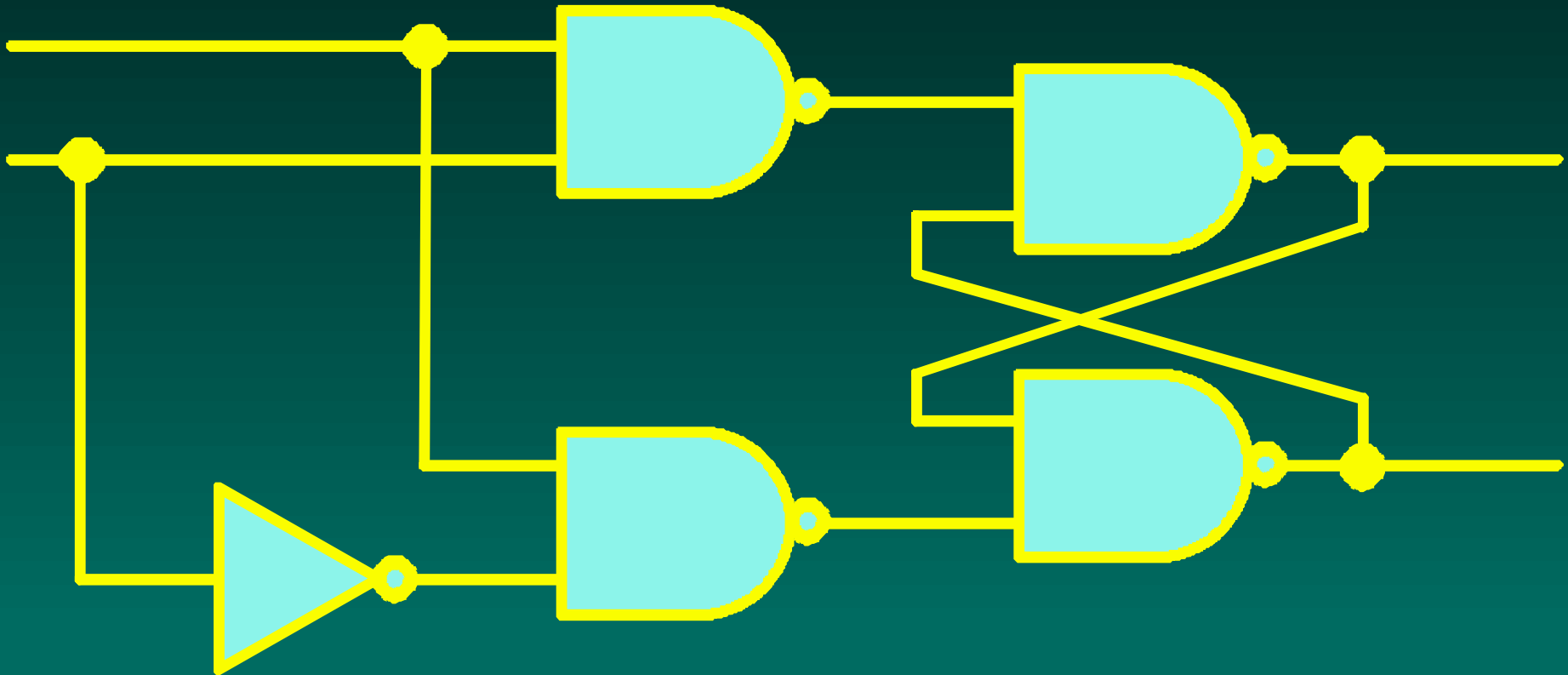


D-Flip Flop Characteristics

- ❖ Avoids the instability of the RS flip-flop
- ❖ Retains its last input value
- ❖ Formally known as a “Delay” flip-flop
- ❖ May become unstable if transisions are too close together
- ❖ Is generally implemented as a special circuit, *not* as pictured here.



A Clocked D Flip-Flop

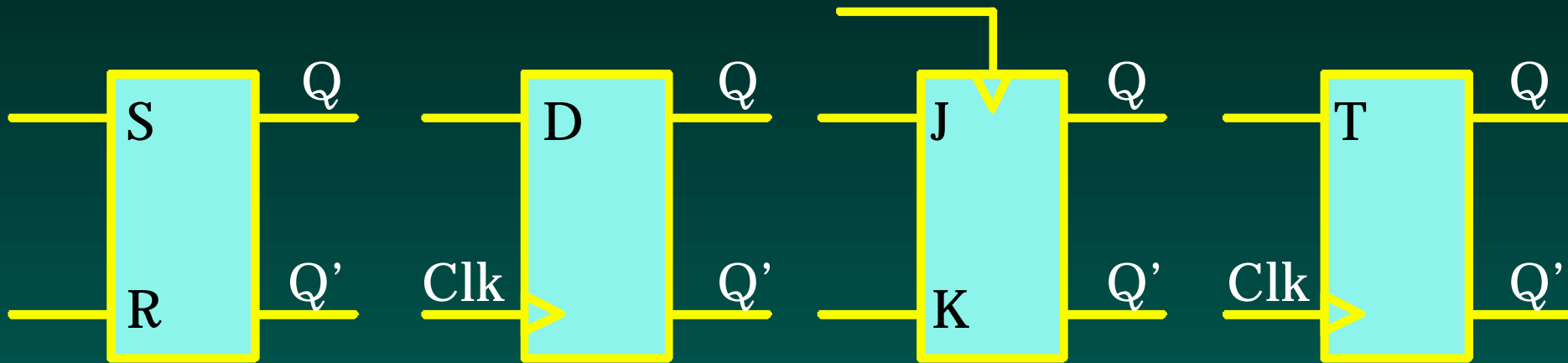




Clocked D-Flip Flop Characteristics

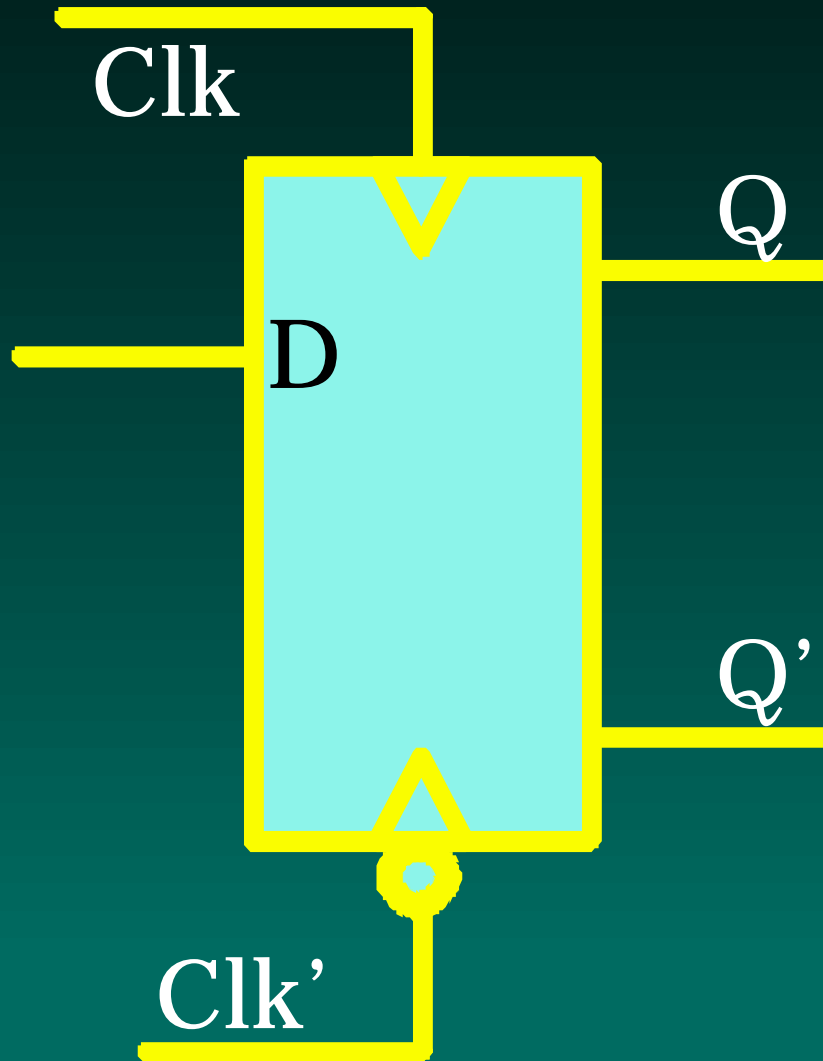
- ❖ Synchronizes transitions with a clock
- ❖ Input should remain stable while clock is active
- ❖ Transition at the wrong time can cause instability
 - Changes while clock is active
 - Changes simultaneous with clock

Flip-Flop Symbols



Flip-Flop Symbols Contain Implicit Feedback Loops

A CMOS Flip-Flop

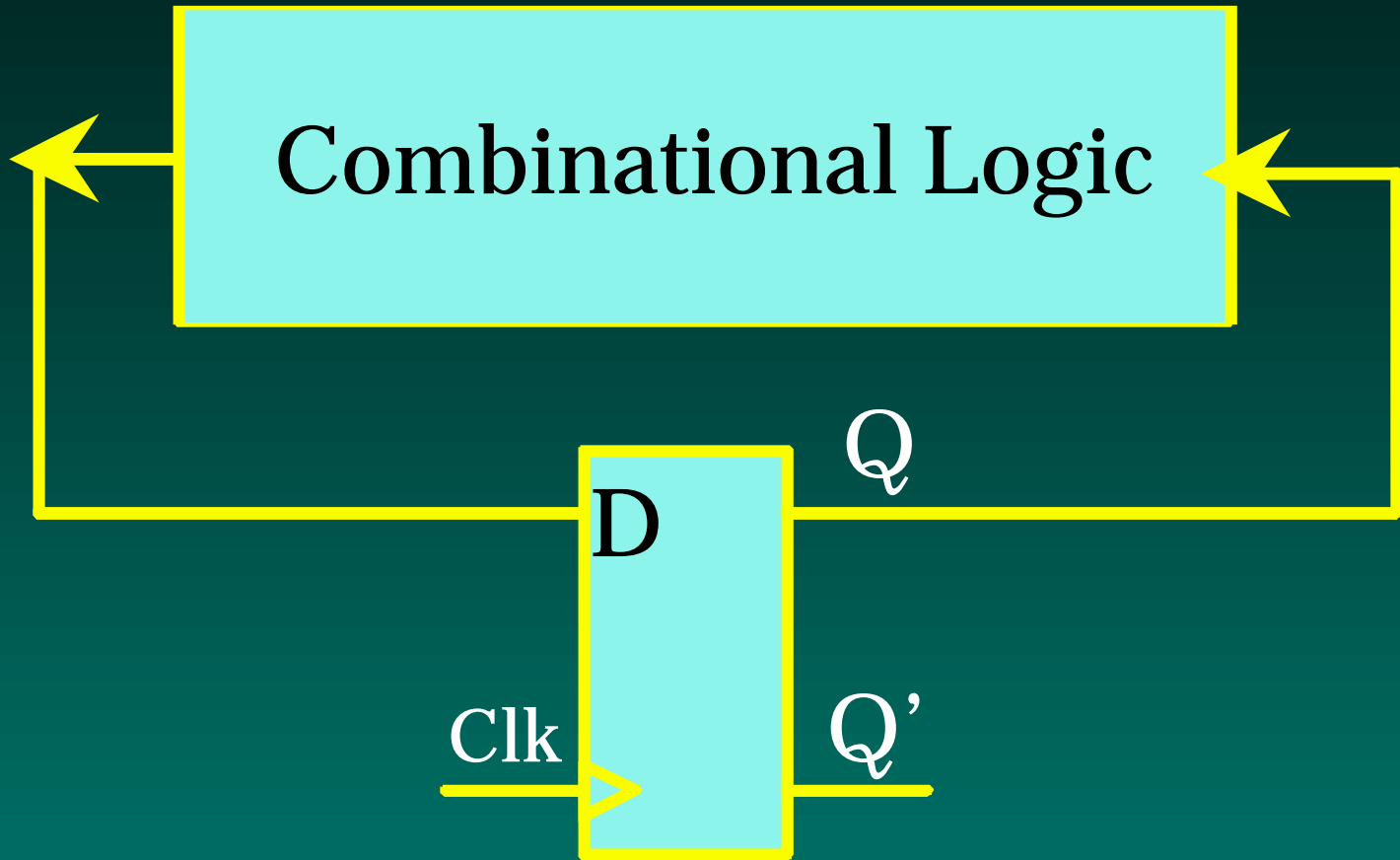




CMOS Logic Elements

- ❖ CMOS = Complementary MOS
- ❖ CMOS Elements Often Require 2 Clocks or 2 Controls
- ❖ Clocks or Controls must be Complements of One another
- ❖ Clock-Skew (Non-Simultaneous changes in both clocks) can cause problems

An Asynchronous Sequential Circuit

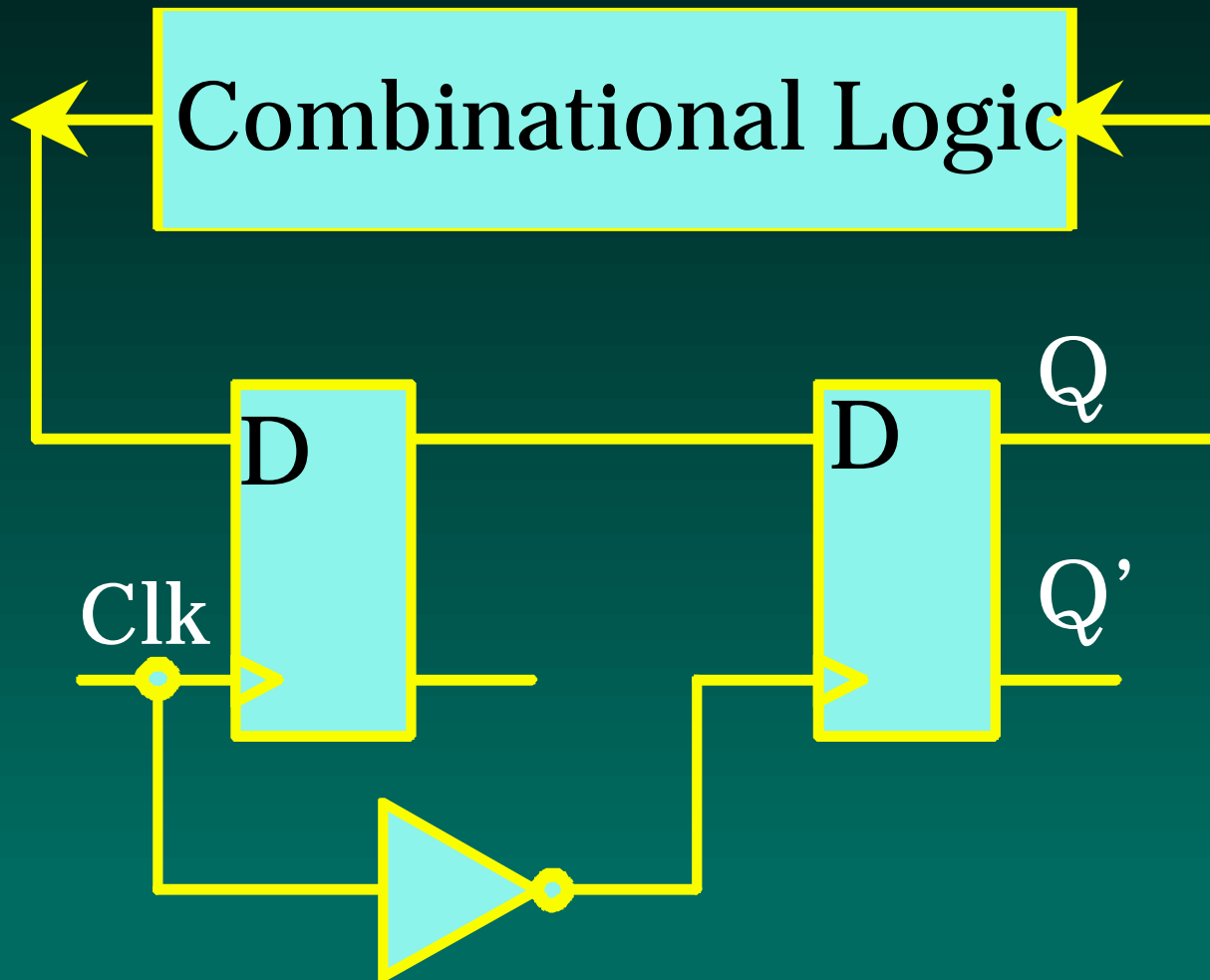




Asynchronous Circuits

- ❖ **Combinational Logic is used:**
 - To Compute New States
 - To Compute Outputs
- ❖ **State is maintained in Asynchronous Circuit Elements**
- ❖ **Care must be used to avoid oscillations**

A Synchronous Sequential Circuit

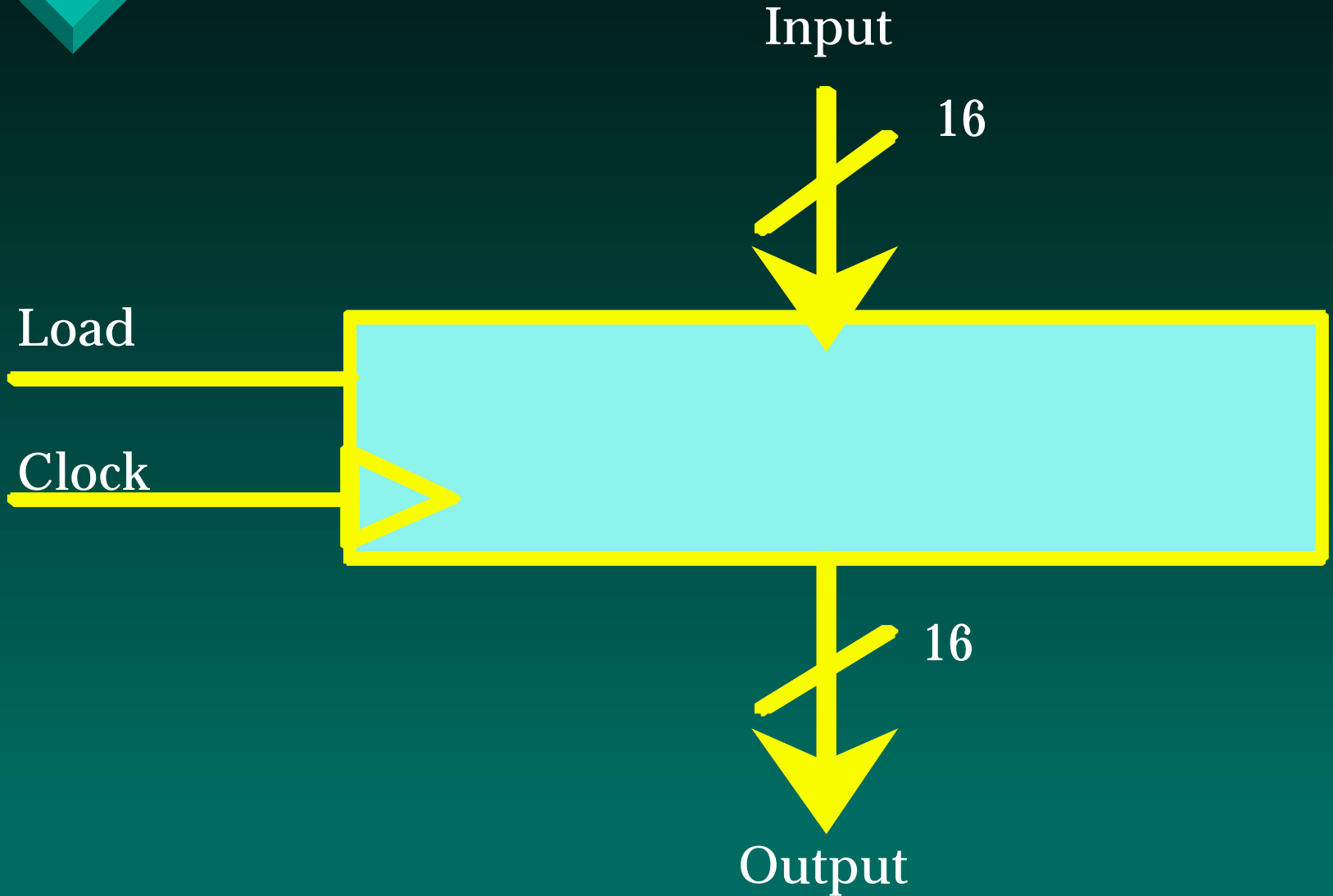




Synchronous Circuits

- ❖ Combinational Logic is used to:
 - Compute New States
 - Compute Outputs
- ❖ State is maintained in Synchronous Flip-Flops
- ❖ State Changes can be made only when clock changes
- ❖ Combinational Logic Must be Stable when Clock is Active

Register Symbol





Register Issues

- ❖ Generally A Collection of D Flip-Flops
- ❖ Can be Synchronous or Asynchronous
- ❖ Default is Assumption is **Synchronous**
- ❖ May have internal wiring to:
 - Perform Shifts
 - Set/Clear
 - All-Zero Status Flag



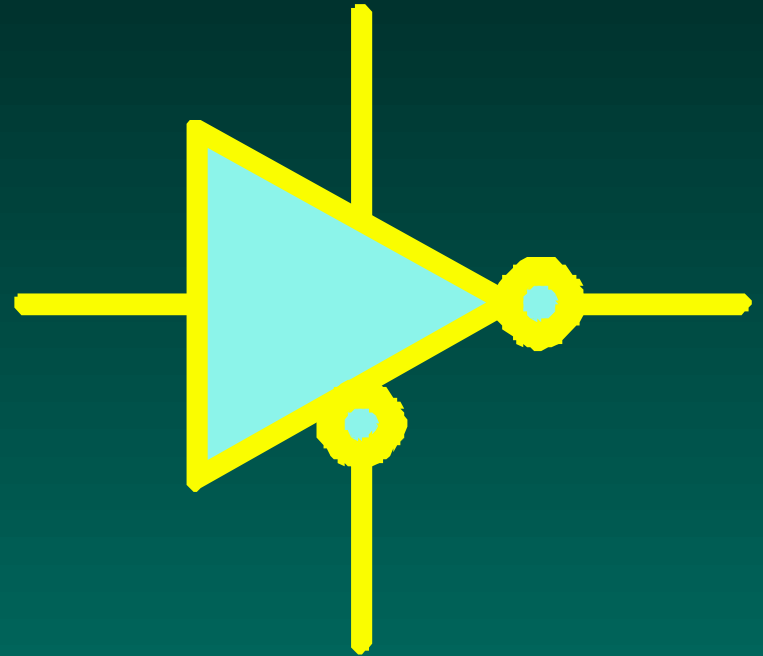
Tristate Elements

- ❖ Three States:
 - Zero (Output is grounded)
 - One (Output connected to Power Terminal)
 - High-Impedance (Output Not Connected to Either Power Or Ground)
- ❖ Can be Used to Construct Cheap Multiplexors

CMOS Tri-state Buffers



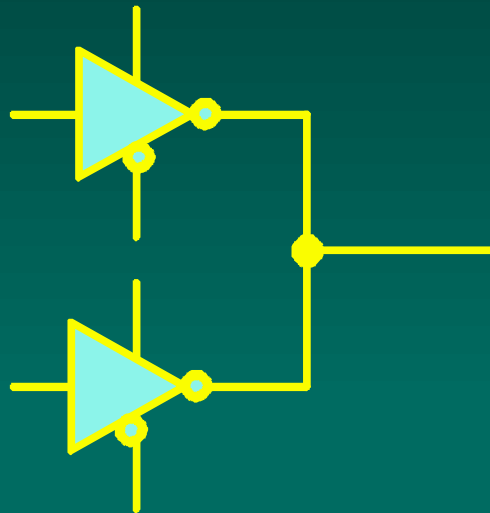
Non-Inverting



Inverting

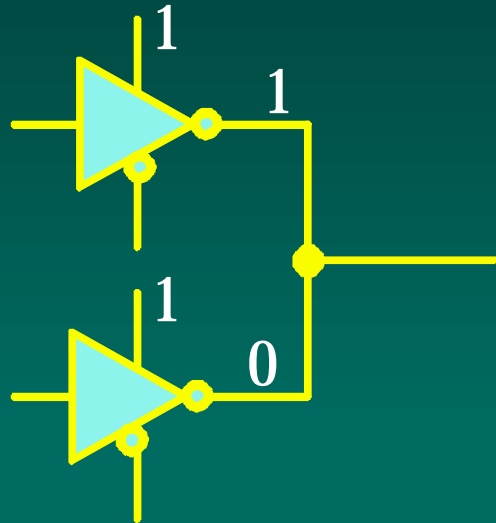
Tri-State Buffer Issues

- ❖ The Gate Amplifies its Signal
- ❖ May be Inverting or Non-Inverting
- ❖ Often used to Construct Multiplexors Using Wired-Or Connections



More Tri-State Issues

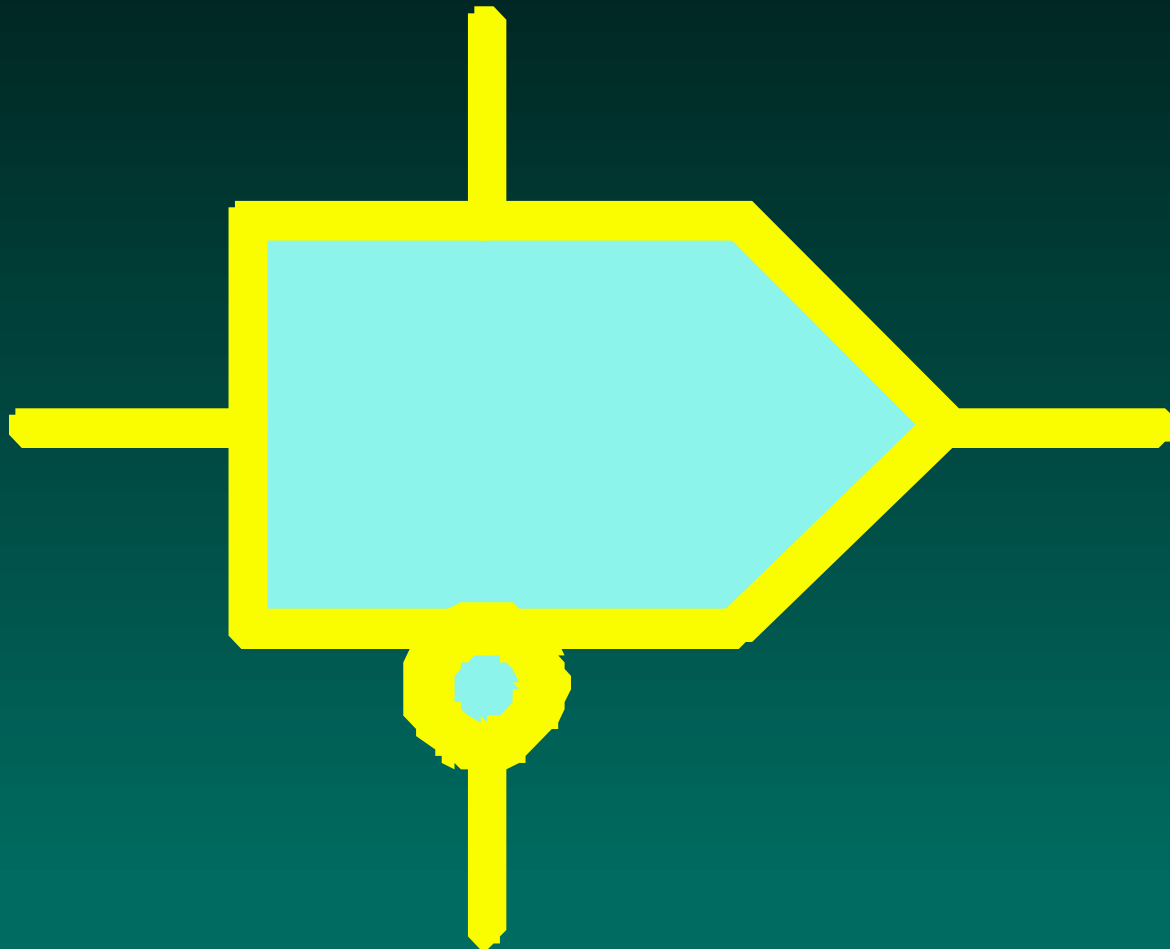
- ❖ In a Wired-Or Connection, Only One Buffer can be in Non-Tristate State
- ❖ Violating This Rule Can Destroy The Circuit Due a Power/Ground Short



DANGER!



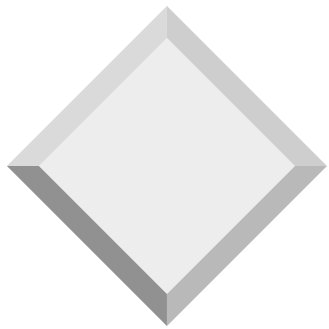
The CMOS Transmission Gate





Transmission Gate Issues

- ❖ Similar to Tristate Buffer
- ❖ Has No Amplification
- ❖ Number of Consecutive Transmission Gates is Limited
- ❖ Similar Problems With Wired-Or Connections



Logic Design

A Review

Boolean Algebra

- ❖ Two Values: zero and one
- ❖ Three Basic Functions: And, Or, Not
- ❖ Any Boolean Function Can be Constructed from These Three

| And | 0 | 1 |
|------------|----------|----------|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

| Or | 0 | 1 |
|-----------|----------|----------|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

| Not | |
|------------|----------|
| 0 | 1 |
| 1 | 0 |

Algebraic Laws

| Classification | Law |
|-----------------|--|
| Identity | $a1=1a=a$ $a+0=0+a=a$ |
| Dominance | $a0=0a=0$ $1+a=a+1=1$ |
| Commutativity | $a+b=b+a$ $ab=ba$ |
| Associativity | $a(bc)=(ab)c$ $a+(b+c)=(a+b)+c$ |
| Distributive | $a(b+c)=ab+ac$ $a+bc=(a+b)(a+c)$ |
| Demorgan's Laws | $(a + b)' = a'b'$ $(ab)' = a' + b'$ |



Boolean Expressions

- ❖ Addition represents OR
- ❖ Multiplication represents AND
- ❖ Not is represented by a prime a' or an overbar \overline{a}
- ❖ Examples:
 - ❖ $s = a'bc + ab'c + abc' + a'b'c'$
 - ❖ $q = ab + bc + ac + abc$

Superfluous Terms

- ❖ The following Two Equations Represent The Same Function.

$$q = ab + bc + ac + abc$$

$$q = ab + bc + ac$$

| a | b | c | q |
|----------|----------|----------|----------|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |



Prime Implicants

- ❖ A Prime Implicant is a Product of Variables or Their Complements, eg. $ab'cd'$
- ❖ If a Prime Implicant has the Value 1, then the Function has the Value 1
- ❖ A Minimal Equation is a Sum of Prime Implicants



Minimization and Minterms

- ❖ Minimization Reduces the Size and Number of Prime Implicants
- ❖ A MinTerm is a Prime Implicant with the Maximum Number of Variables
- ❖ For a 3-input Function $a'bc$ is a MinTerm, while ab is not.
- ❖ Prime Implicants can be Combined to Eliminate Variables, $abc' + abc = ab$

Minimization with Maps

❖ A Karnaugh Map

| | 00 | 01 | 11 | 10 |
|---|----|----|----|----|
| 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 |

A {

C {

B {



Procedure

- ❖ Select Regions Containing All 1's
- ❖ Regions should be as Large as Possible
- ❖ Regions must contain 2^k cells
- ❖ Regions should overlap as little as possible
- ❖ The complete set of regions must contain all 1's in the map



Procedure 2

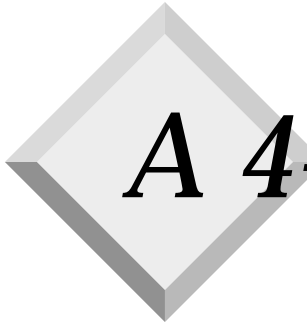
- ❖ Top and Bottom of Map are Contiguous
- ❖ Left and Right of Map are Contiguous
- ❖ Regions represent Prime Implicants
- ❖ Use Variable name guides to construct equation
 - Completely inside the region of a variable means prime implicant contains variable
 - Completely outside the region of a variable means prime implicant contains negation



Applied to Previous Map

| | | C | | B | |
|-----|---|----|----|----|----|
| | | 00 | 01 | 11 | 10 |
| A { | 0 | 1 | 0 | 0 | 1 |
| | 1 | 1 | 0 | 0 | 0 |

$$q = c'b' + c'a'$$



A 4-Variable Karnaugh Map

| | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 1 | 1 |

Diagram annotations:

- Variable **D** is indicated by a bracket above the columns 11 and 10.
- Variable **C** is indicated by a bracket above the columns 11 and 10.
- Variable **B** is indicated by a bracket to the left of the rows 01 and 11.
- Variable **A** is indicated by a bracket to the left of the rows 11 and 10.



First Minimization

| | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 1 | 1 |

Diagram illustrating the first minimization step of a Karnaugh map. The map is a 4x4 grid with columns labeled 00, 01, 11, and 10, and rows labeled 00, 01, 11, and 10. The top row (00) and the first column (00) are shaded gray. The map contains several 1s, which are grouped into four pairs:

- Group D: A horizontal pair of 1s in the 01 and 11 columns of the 01 row.
- Group C: A horizontal pair of 1s in the 11 and 10 columns of the 10 row.
- Group B: A vertical pair of 1s in the 01 and 11 columns of the 11 row.
- Group A: A vertical pair of 1s in the 01 and 11 columns of the 10 row.

Labels A, B, C, and D are placed to the left and above the grid, with curly braces indicating the groups. The 1s in the 01 and 11 columns of the 01, 11, and 10 rows are also circled.



Second Minimization

| | 00 | 01 | 11 | 10 |
|----|----|----|----|----|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 0 | 1 | 1 | 1 |

Diagram illustrating the second minimization step of a Karnaugh map. The map is a 4x4 grid with columns labeled 00, 01, 11, and 10, and rows labeled 00, 01, 11, and 10. The top row (00) and the first column (00) are shaded gray. The variables A, B, C, and D are indicated by brackets above and to the left of the grid. The 1s in the map are circled, and the 1s in the 01 and 11 rows are grouped together by a horizontal oval, representing a prime implicant.



Minimal Forms for Previous Slides:

❖ $ab'd + bc'd + a'bc + acd'$

❖ $ac'd + a'bd + bcd' + ab'c$

❖ Moral: A Boolean Function May Have Several Different Minimal Forms

❖ Karnaugh Maps are Ineffective for Functions with More than Six Inputs.



Quine McClusky Minimization

- ❖ Amenable to Machine Implementation
- ❖ Applicable to Circuits with an Arbitrary Number of Inputs
- ❖ Effective Procedure for Finding Prime Implicants, but ...
- ❖ Can Require an Exponential Amount of Time for Some Circuits



Quine-McClusky Procedure

- ❖ Start with The Function Truth Table
- ❖ Extract All Input Combinations that Produce a TRUE Output (MinTerms)
- ❖ Group All MinTerms by The Number of Ones They Contain
- ❖ Combine Minterms from Adjacent Groups



More Quine-McClusky

- ❖ Two Min-Terms Combine If They Differ by Only One Bit
- ❖ The Combined MinTerm has an x in the Differing Position
- ❖ Create New Groups From Combined Min-Terms
- ❖ Each Member of A New Group Must Have the Same Number of 1's and x's



Yet More Quine-McClusky

- ❖ Each Member of A Group Must Have x's in The Same Position.
- ❖ Combine Members of the New Groups To Create More New Groups
- ❖ Combined Terms Must Differ By One Bit, and Have x's in the Same Positions
- ❖ Combine as Much as Possible
- ❖ Select Prime Implicants to “Cover” All Ones in the Function



Quine-McClusky Example 1

Numbers in Parentheses are Truth-Table Positions.

| | | |
|-----------------|-----------------|-----------------|
| 0011(3) | 1100(12) | |
| 0111(7) | 1011(11) | 1101(13) |
| 1110(14) | | |
| 1111(15) | | |

Quine-McClusky Example 2

New Groups After Combining MinTerms

| | |
|---------------|---------------|
| $0x11(3,7)$ | $110x(12,13)$ |
| $1x11(11,15)$ | $111x(14,15)$ |
| $x011(3,11)$ | $11x0(12,14)$ |
| $x111(7,15)$ | $11x1(13,15)$ |



Quine-McClusky Example 3

The Final Two Groups

Note That These Two Elements Cover
All Truth-Table Positions

| |
|---------------------|
| $xx11(3,7,11,15)$ |
| $11xx(12,13,14,15)$ |

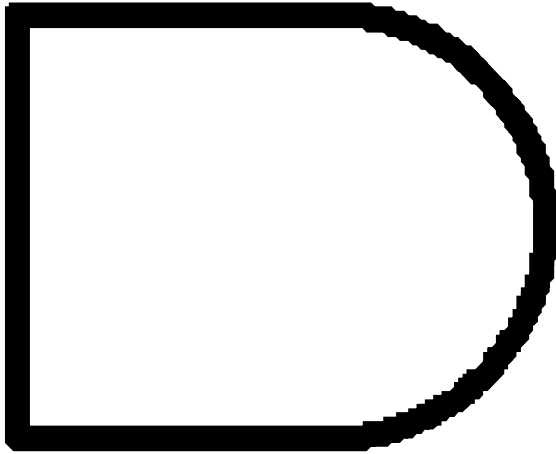


Quine-McClusky Example 4

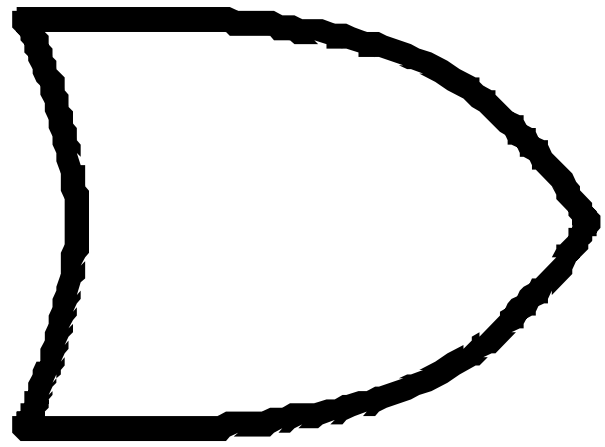
- ❖ Each Group Element Represents a Prime Implicant
- ❖ It is Necessary to Select Group Elements to Cover All Truth-Table Positions.
- ❖ In This Case, $ab+cd$ is the Minimal Formula.
- ❖ In General, Selecting a Minimal Number of Prime Implicants is NP-Complete



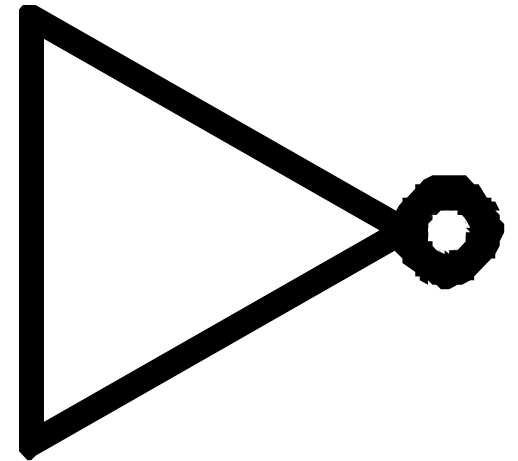
Basic Logic Symbols



And



Or

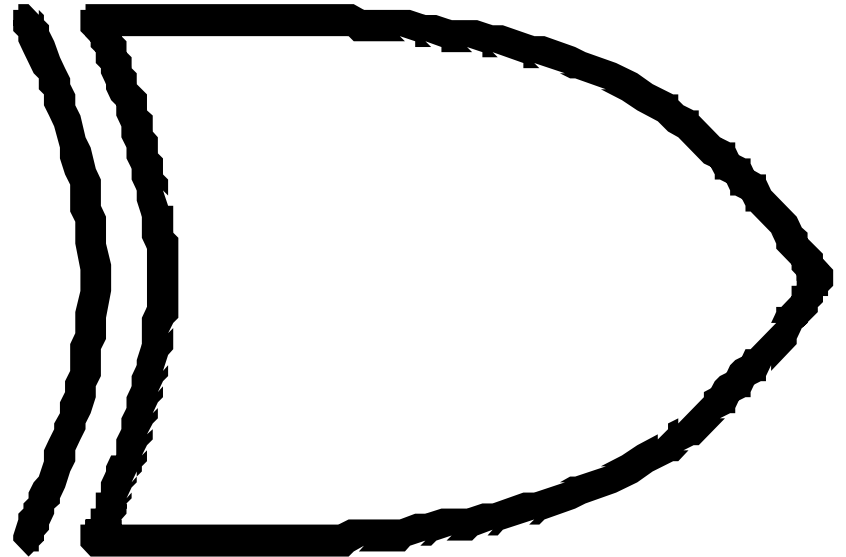


Not

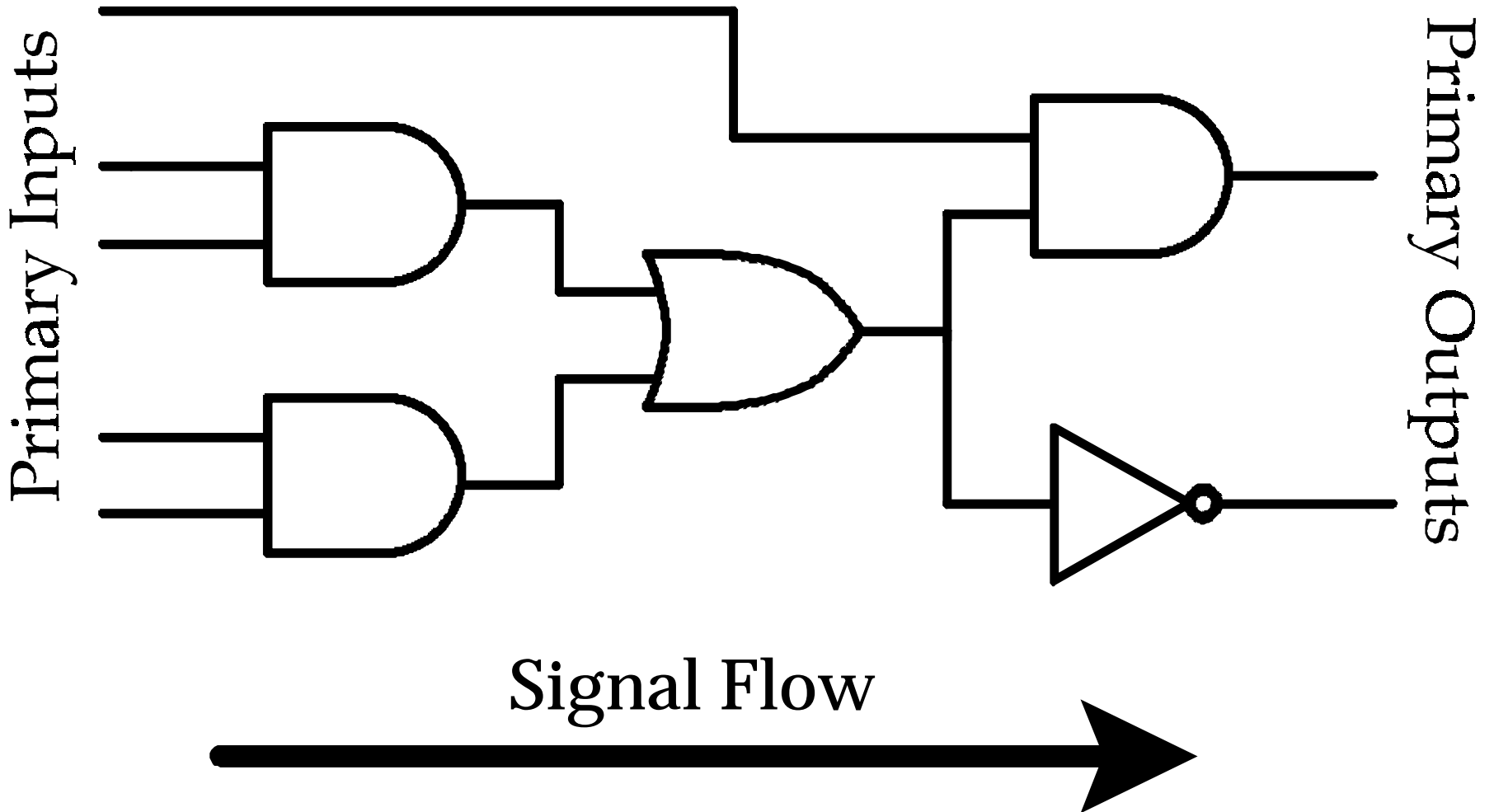


The Exclusive Or Function

| Xor | 0 | 1 |
|------------|----------|----------|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

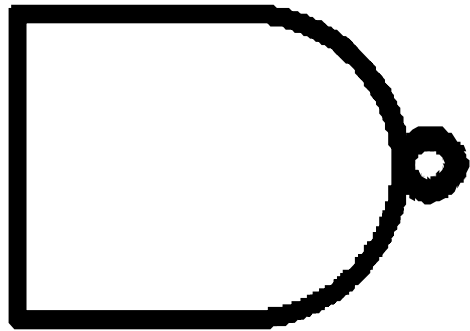


A Simple Logic Diagram

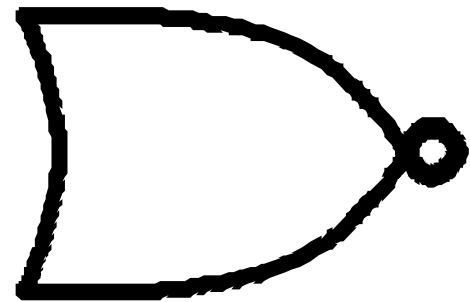




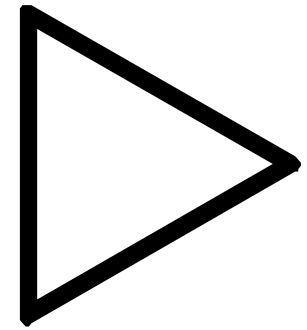
Additional Logic Symbols



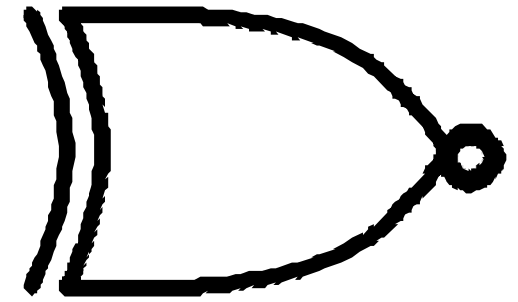
Nand



Nor



Buffer



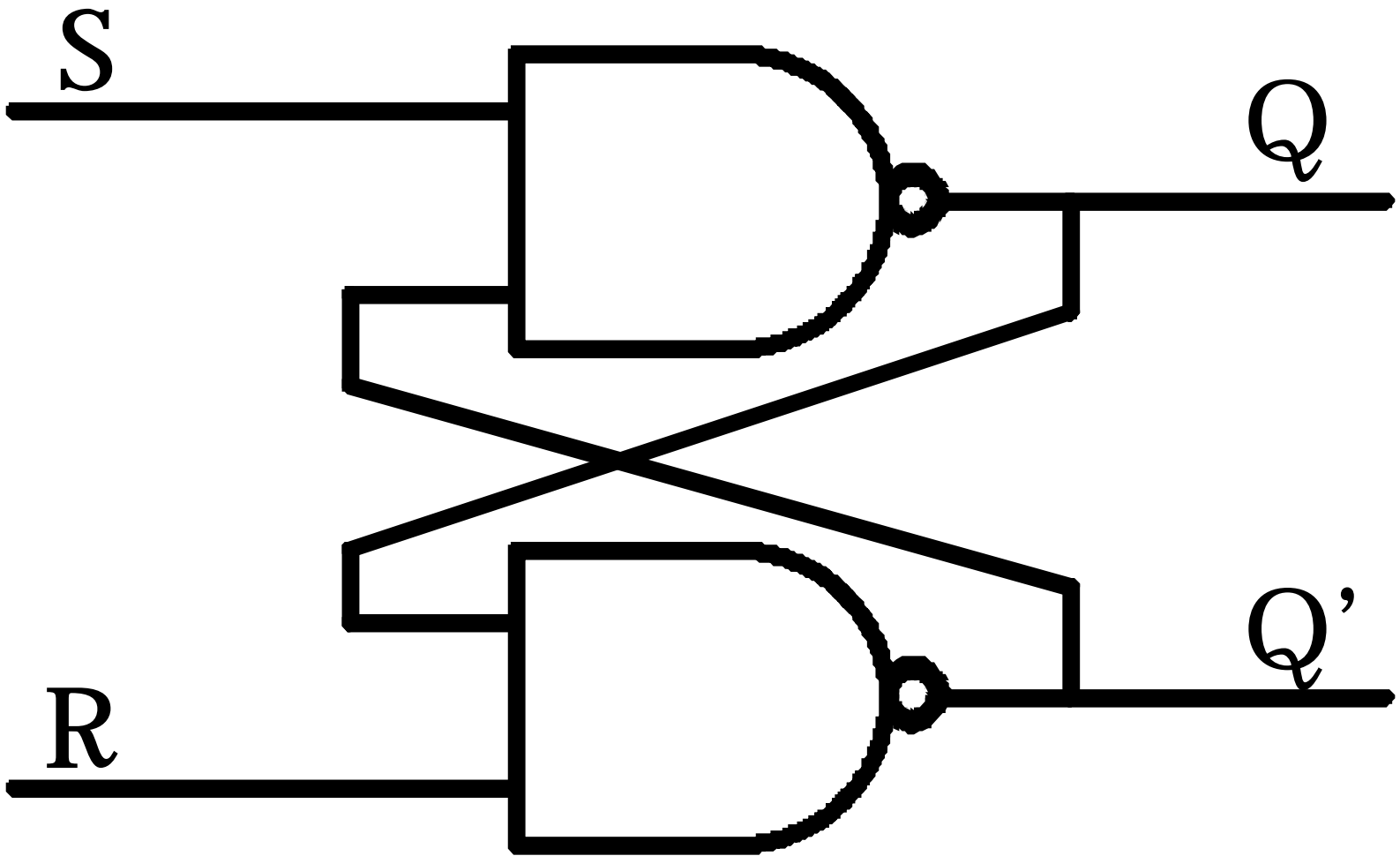
Xnor



Sequential Logic

- ❖ Contains Memory Elements
- ❖ Memory is Provided by Feedback
- ❖ Circuit diagrams generally have implicit or explicit cycles
- ❖ Two Styles: Synchronous and Asynchronous

An RS Flip-Flop





RS Characteristics

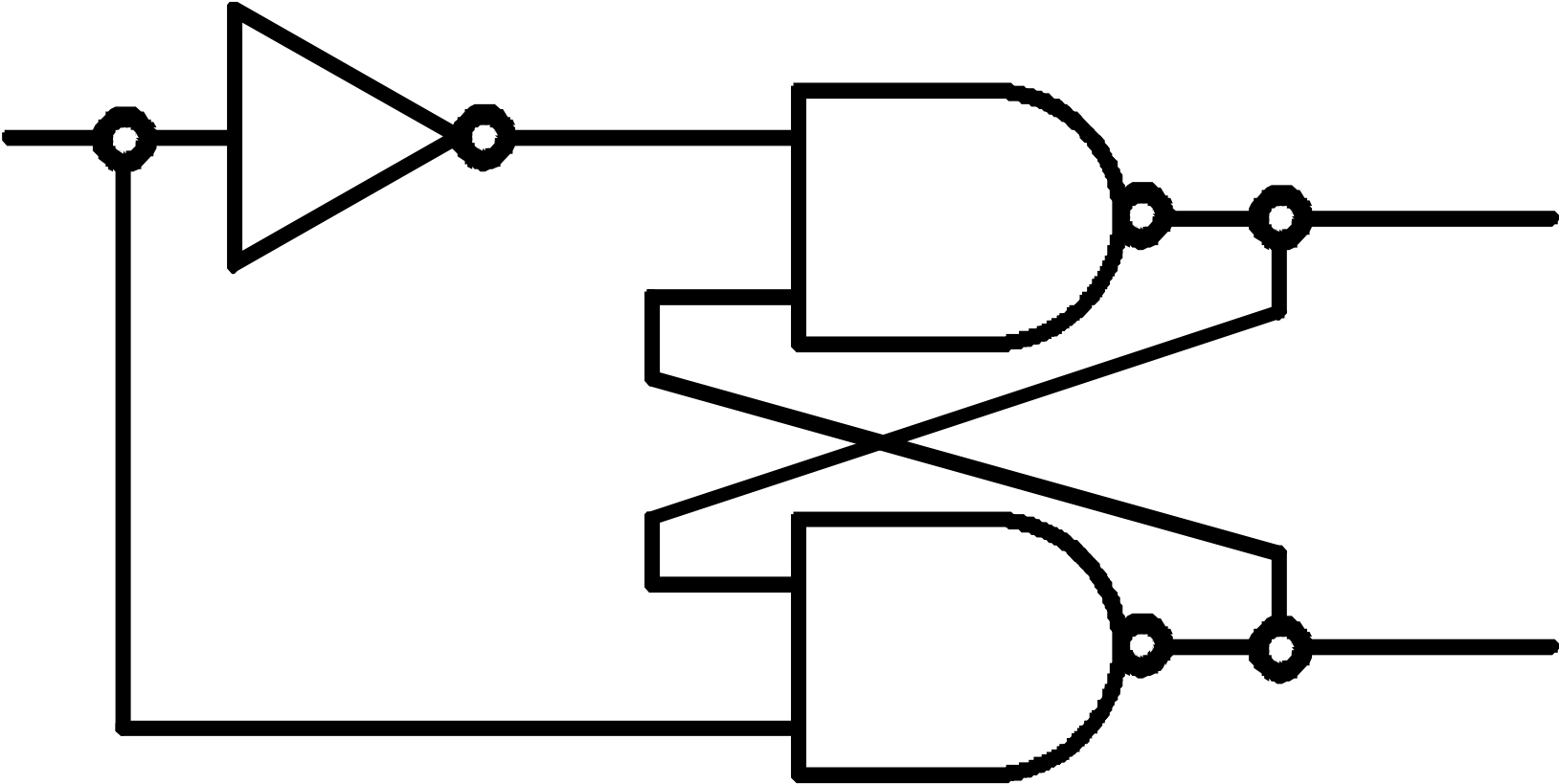
- ❖ If $S=0$ and $R=1$, Q is set to 1, and Q' is reset to 0
- ❖ If $R=0$ and $S=1$, Q is reset to 0, and Q' is set to 1
- ❖ If $S=1$ and $R=1$, Q and Q' maintain their previous state.
- ❖ If $S=0$ and $R=0$, a transition to $S=1$, $R=1$ will cause oscillation.



Instability

- ❖ RS flip-flops can become unstable if both R and S are set to zero.
- ❖ All Sequential elements are fundamentally unstable under certain conditions
 - Invalid Transitions
 - Transitions too close together
 - Transitions at the wrong time

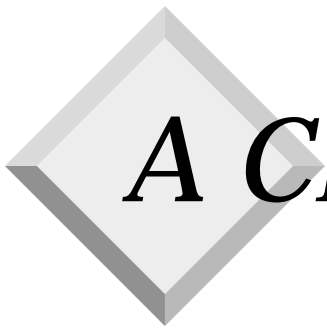
D Flip-Flops



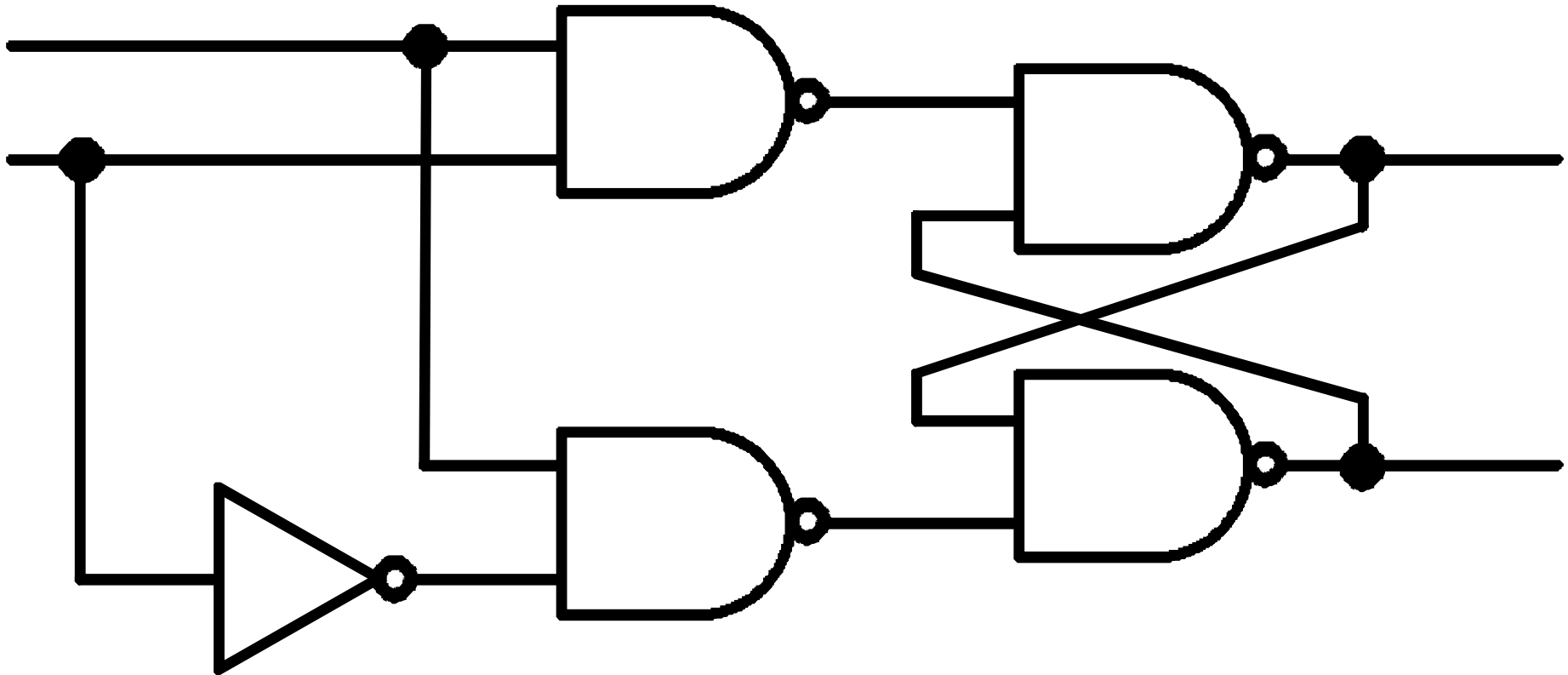


D-Flip Flop Characteristics

- ❖ Avoids the instability of the RS flip-flop
- ❖ Retains its last input value
- ❖ Formally known as a “Delay” flip-flop
- ❖ May become unstable if transisions are too close together
- ❖ Is generally implemented as a special circuit, *not* as pictured here.



A Clocked D Flip-Flop

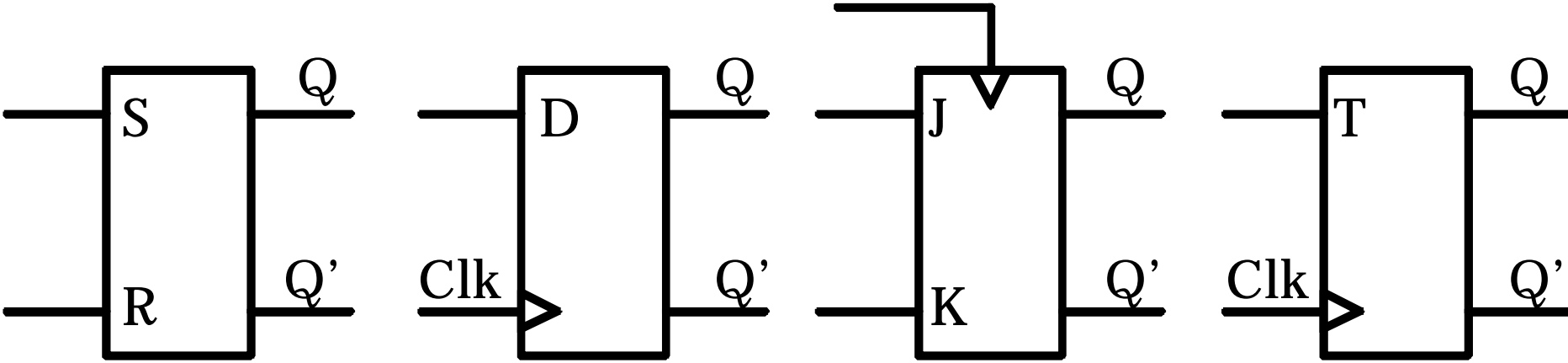




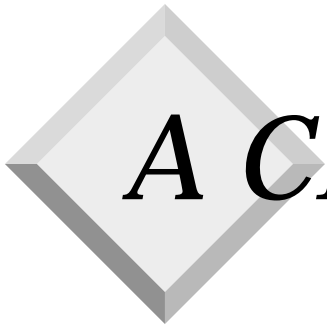
Clocked D-Flip Flop Characteristics

- ❖ Synchronizes transitions with a clock
- ❖ Input should remain stable while clock is active
- ❖ Transition at the wrong time can cause instability
 - Changes while clock is active
 - Changes simultaneous with clock

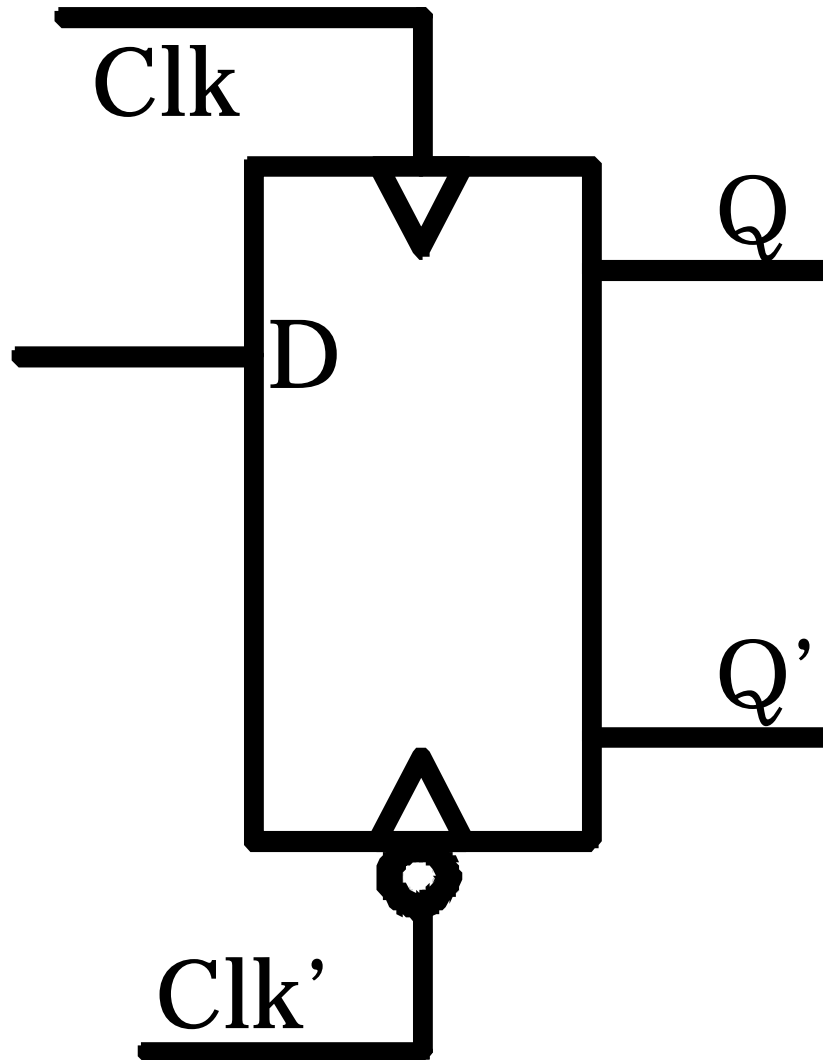
Flip-Flop Symbols



Flip-Flop Symbols Contain Implicit Feedback Loops



A CMOS Flip-Flop

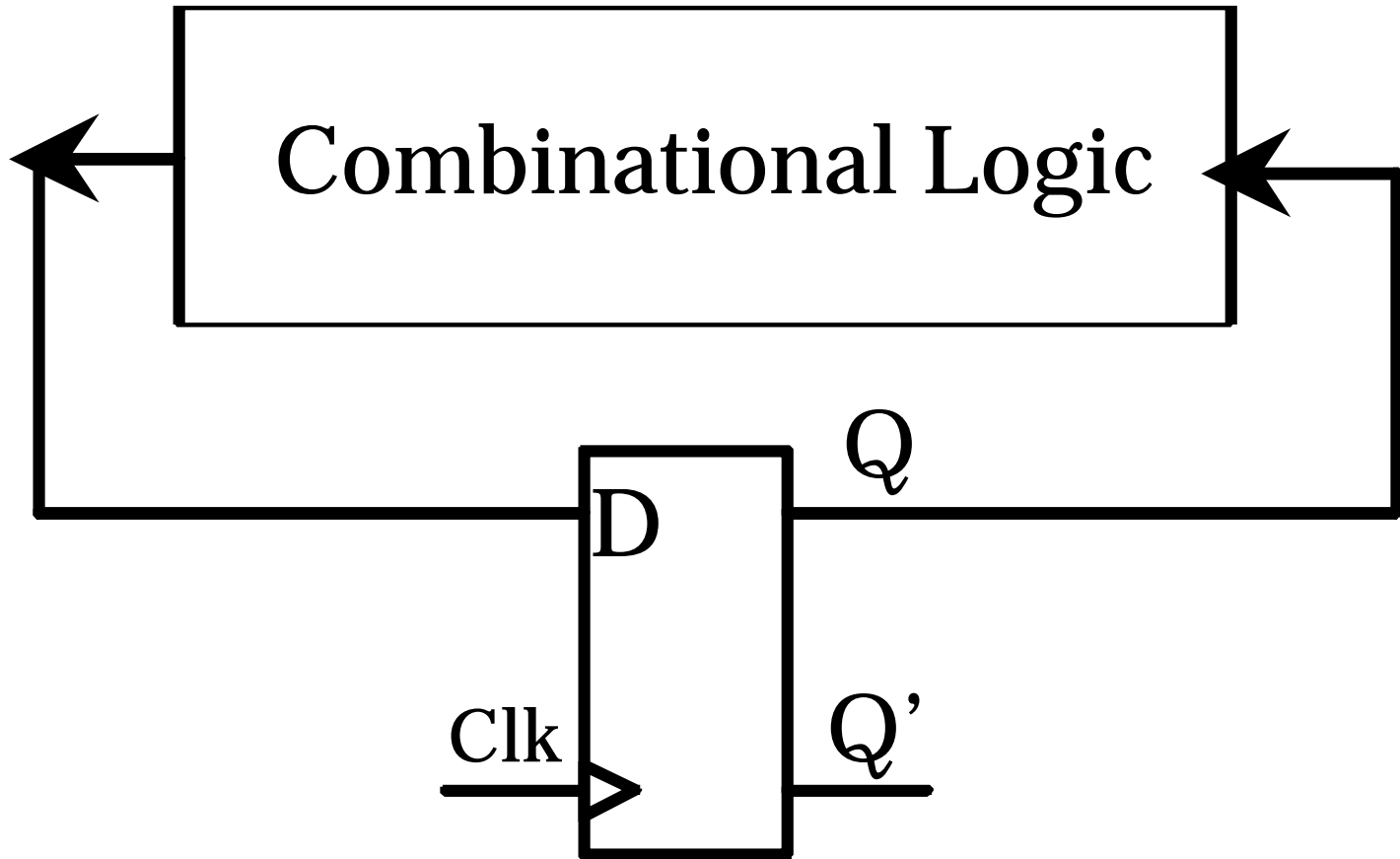




CMOS Logic Elements

- ❖ CMOS = Complementary MOS
- ❖ CMOS Elements Often Require 2 Clocks or 2 Controls
- ❖ Clocks or Controls must be Complements of One another
- ❖ Clock-Skew (Non-Simultaneous changes in both clocks) can cause problems

An Asynchronous Sequential Circuit

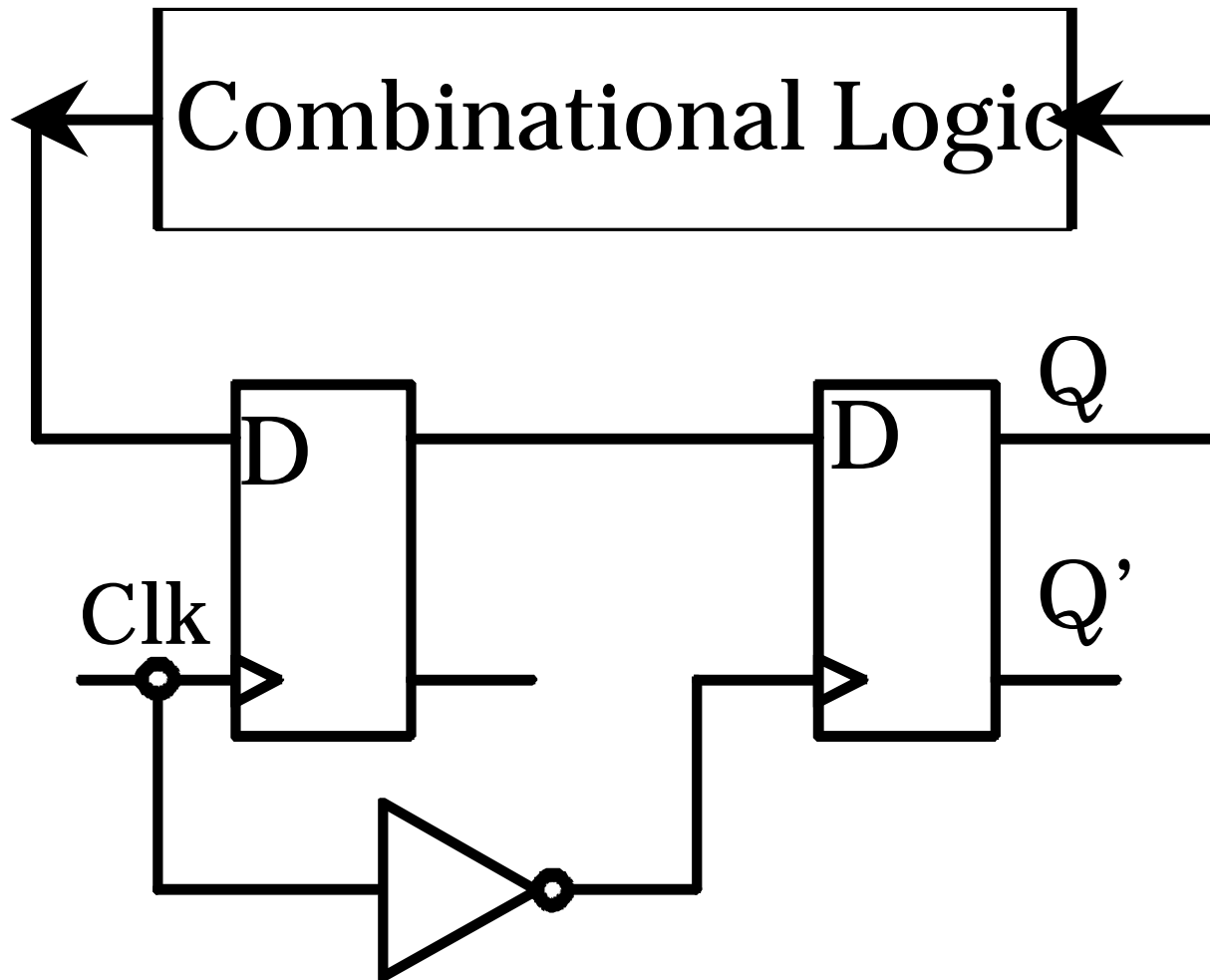




Asynchronous Circuits

- ❖ **Combinational Logic is used:**
 - To Compute New States
 - To Compute Outputs
- ❖ **State is maintained in Asynchronous Circuit Elements**
- ❖ **Care must be used to avoid oscillations**

A Synchronous Sequential Circuit

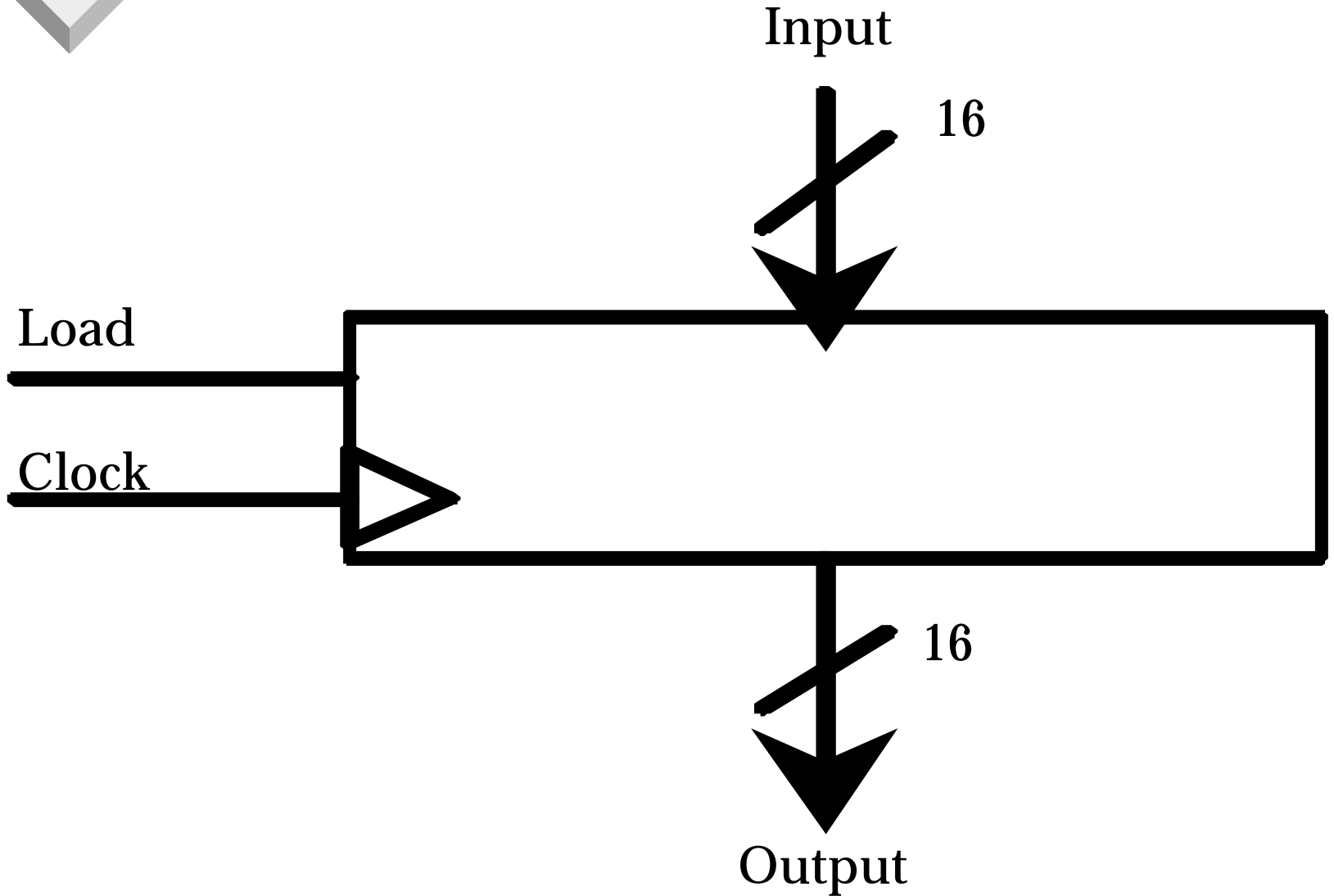




Synchronous Circuits

- ❖ **Combinational Logic is used to:**
 - Compute New States
 - Compute Outputs
- ❖ **State is maintained in Synchronous Flip-Flops**
- ❖ **State Changes can be made only when clock changes**
- ❖ **Combinational Logic Must be Stable when Clock is Active**

Register Symbol





Register Issues

- ❖ Generally A Collection of D Flip-Flops
- ❖ Can be Synchronous or Asynchronous
- ❖ Default is Assumption is Synchronous
- ❖ May have internal wiring to:
 - Perform Shifts
 - Set/Clear
 - All-Zero Status Flag

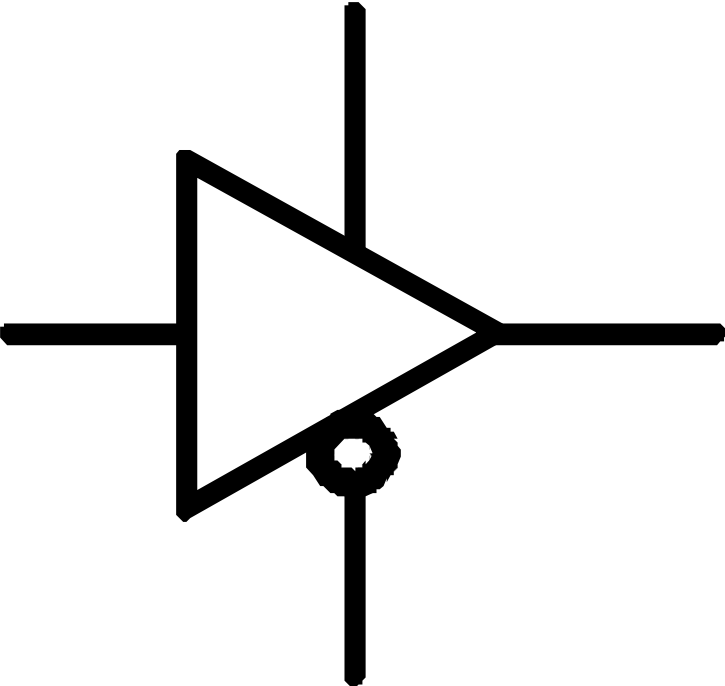


Tristate Elements

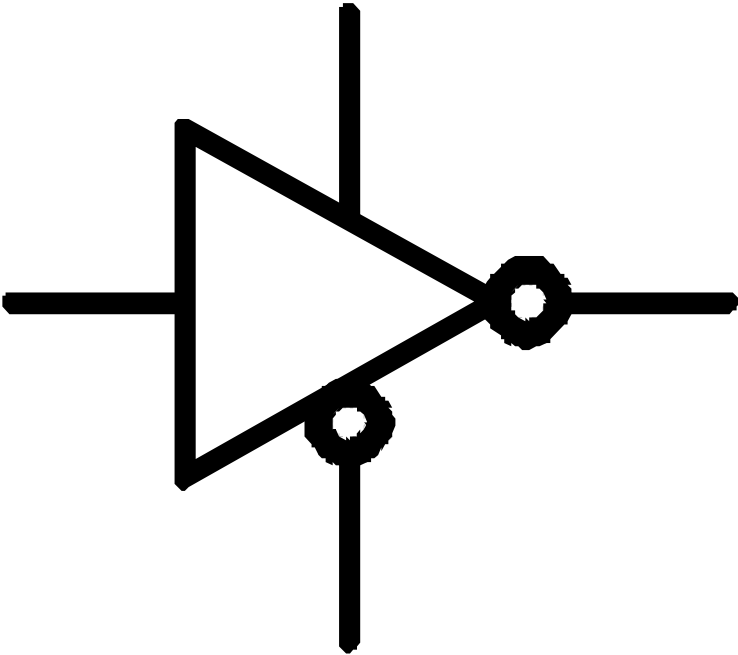
- ❖ Three States:
 - Zero (Output is grounded)
 - One (Output connected to Power Terminal)
 - High-Impedance (Output Not Connected to Either Power Or Ground)
- ❖ Can be Used to Construct Cheap Multiplexors



CMOS Tri-state Buffers



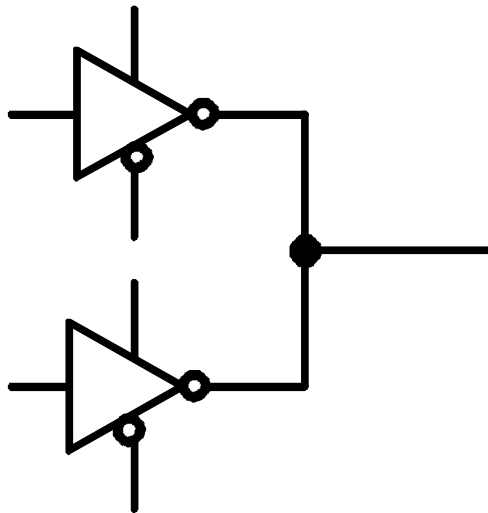
Non-Inverting



Inverting

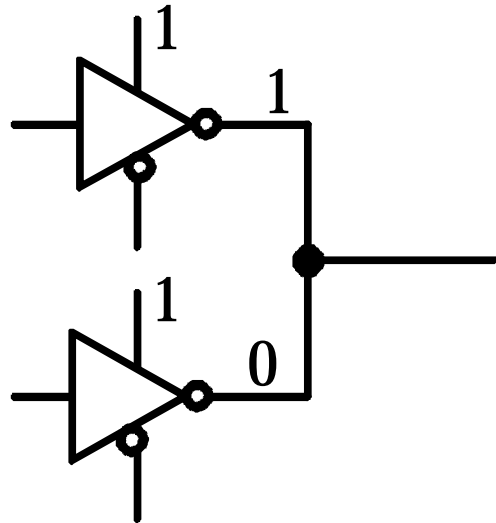
Tri-State Buffer Issues

- ❖ The Gate Amplifies its Signal
- ❖ May be Inverting or Non-Inverting
- ❖ Often used to Construct Multiplexors Using Wired-Or Connections



More Tri-State Issues

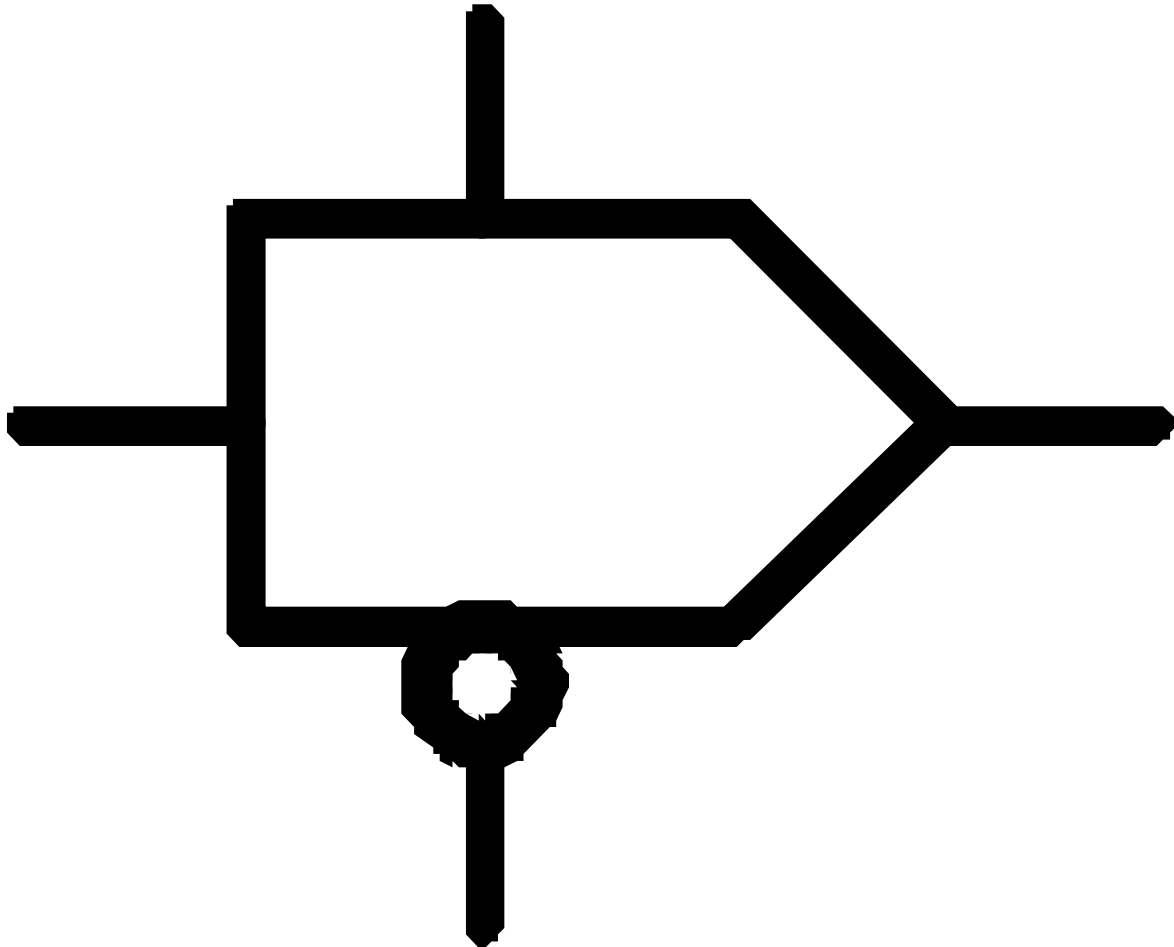
- ❖ In a Wired-Or Connection, Only One Buffer can be in Non-Tristate State
- ❖ Violating This Rule Can Destroy The Circuit Due a Power/Ground Short



DANGER!



The CMOS Transmission Gate





Transmission Gate Issues

- ❖ Similar to Tristate Buffer
- ❖ Has No Amplification
- ❖ Number of Consecutive Transmission Gates is Limited
- ❖ Similar Problems With Wired-Or Connections