Lecture Outline

Reminders to self:

- ❑ Turn on lecture recording to Cloud
- ❑ Turn on Zoom microphone

- Last Lecture
  - Continued Three-state buffers / Tri-state outputs
  - Decoders and Encoders
  - Started ROM
- Today's Lecture
  - Continue ROM
  - Programmable Logic Arrays
  - Start Latches & Flip-Flops (primarily latches today)

Handouts and Announcements

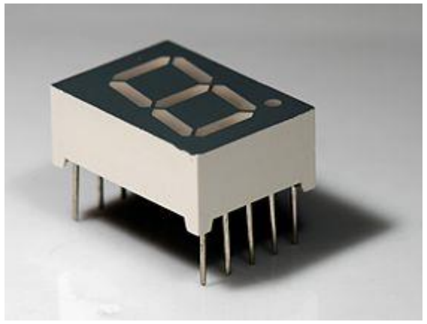
- Announcements
  - Homework Problem 9-3
    - Posted on Carmen today
    - Due: 11:59pm Thursday 2/23
  - Homework Reminder:
    - HW 9-1 due: 11:25am Wednesday 2/22
    - HW 9-2 due: 11:59pm Thursday 2/23
  - Read for Wednesday: pages 342-352

Handouts and Announcements

- Announcements
  - Mini-Exam 3 Reminder
    - Available 5pm Monday 2/20 through 5:00pm Tuesday 2/21
    - Due in Carmen PROMPTLY at 5:00pm on 2/21
    - Designed to be completed in ~36 min, but you may use more
    - When planning your schedule:
      - I recommend building in 10-15 min extra
      - To allow for downloading exam, signing and dating honor pledge, saving solution as pdf, and uploading to Carmen
    - I also recommend not procrastinating
  - Exam review topics available on Carmen
  - Sample Mini-Exams 3 and 4 from Au20 also available

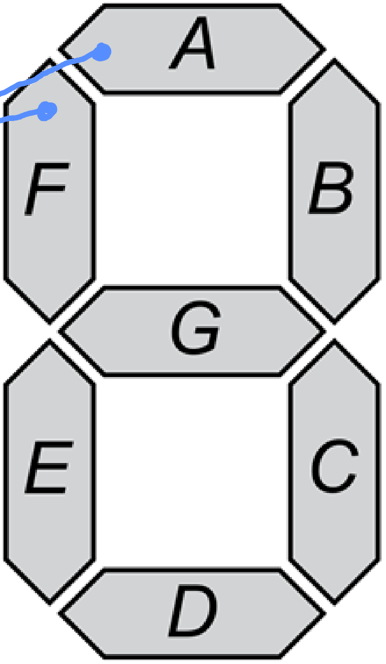
Read-Only Memory

Example: ROM for seven-segment display driver



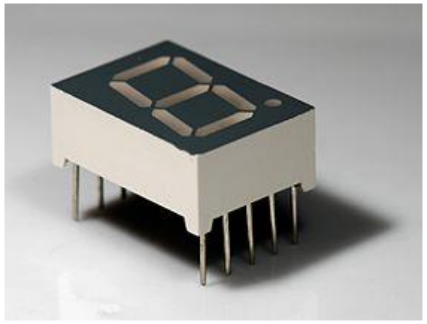
- Seven-segment display: an electronic display device for displaying decimal numerals
- Alternative to the more complex dot matrix displays
- Used in things such as digital clocks, electronic meters, basic calculators, etc.
    - Light up B and C → Displays a 1
    - Light up B, C, F and G → Displays a 4
    - Etc.
- Goal: ROM LUT for BCD to seven-segment driver

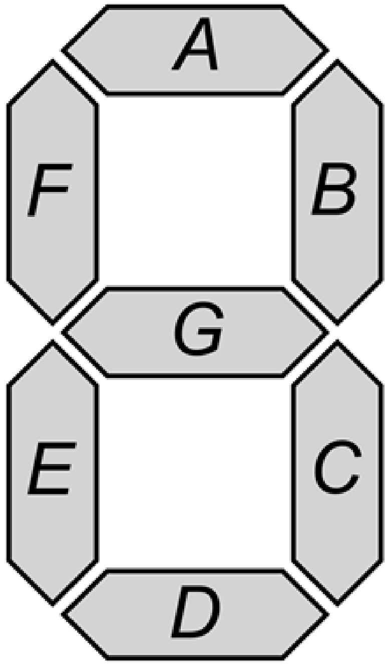
Each segment an LED



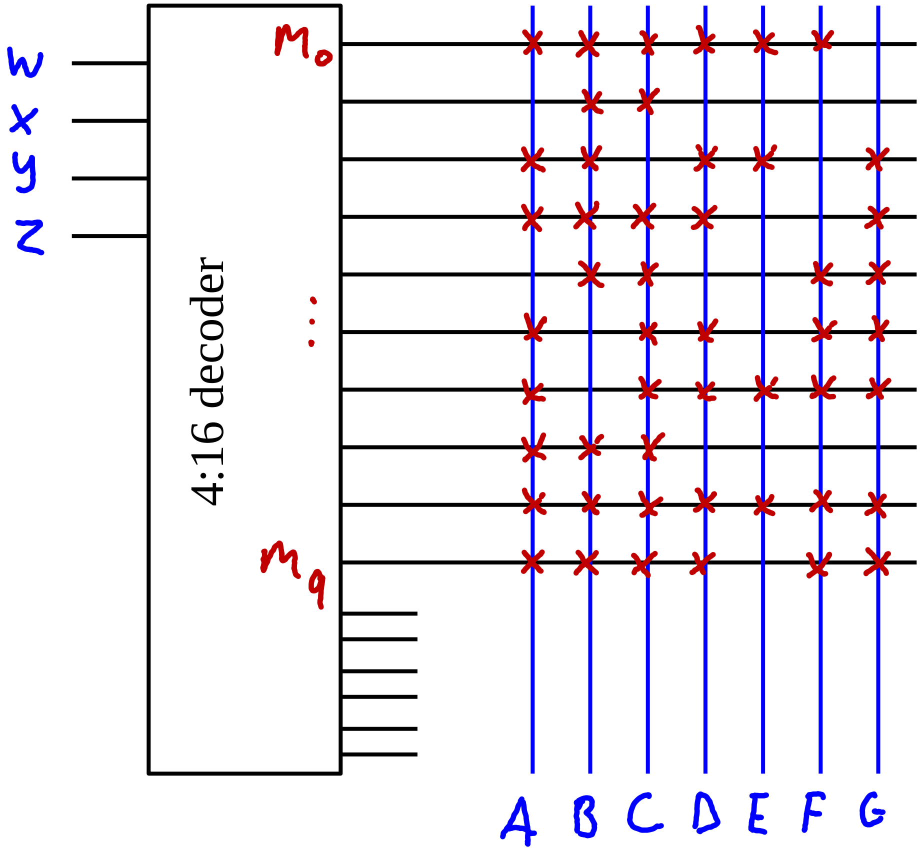
Read-Only Memory

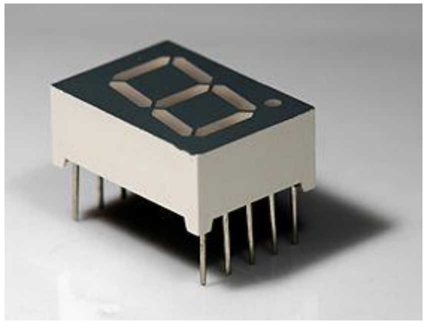
W	*X*	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	*F*	*G*
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

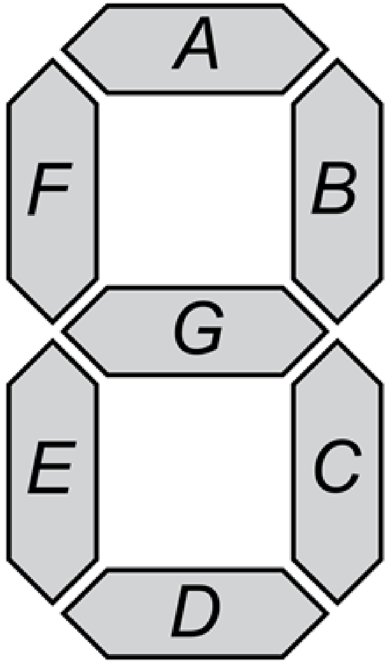




Read-Only Memory







Read-Only Memory

Types of ROM (Non-volatile memory)

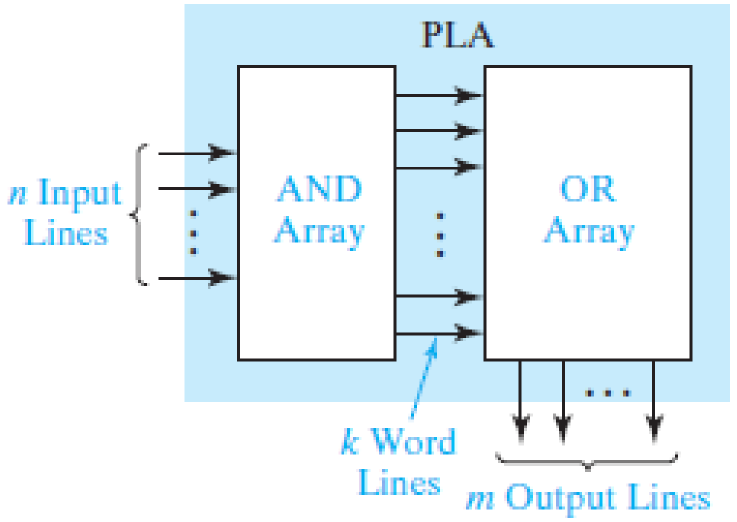
- Standard ROM
  - Permanent
  - Switches between word and output lines hardwired in
  - Either the switch is there, or it is not
  - Done at "Mask-level" during fabrication of IC
- Programmable ROM (PROM)
  - Can electronically set the locations of the "on" switches
  - **<u>Once</u>**
- Electronically Erasable PROM (EEPROM)
  - Can electronically reprogram
  - On order of 100 to 1000 times
  - Uses a special type of FET for the switches
  - Takes large voltages to erase/reprogram
  - Limited number of times before FET fails

Programmable Logic Arrays

- A Programmable Logic Array (PLA) performs the same basic function as a ROM.
- A PLA with n inputs and m outputs can realize m functions of n variables
- But internal organization of PLA is different than that of ROM
  - ROM directly implements a truth table
  - PLA uses arrays of ANDs and ORs to implement an SOP expression
- # Input lines: n
- # Output lines: m
- Between AND array and OR array: k Word Lines

FIGURE 9-28 Programmable Logic Array Structure

© Cengage Learning 2014



THE OHIO STATE UNIVERSITY
COLLEGE OF ENGINEERING

Programmable Logic Arrays

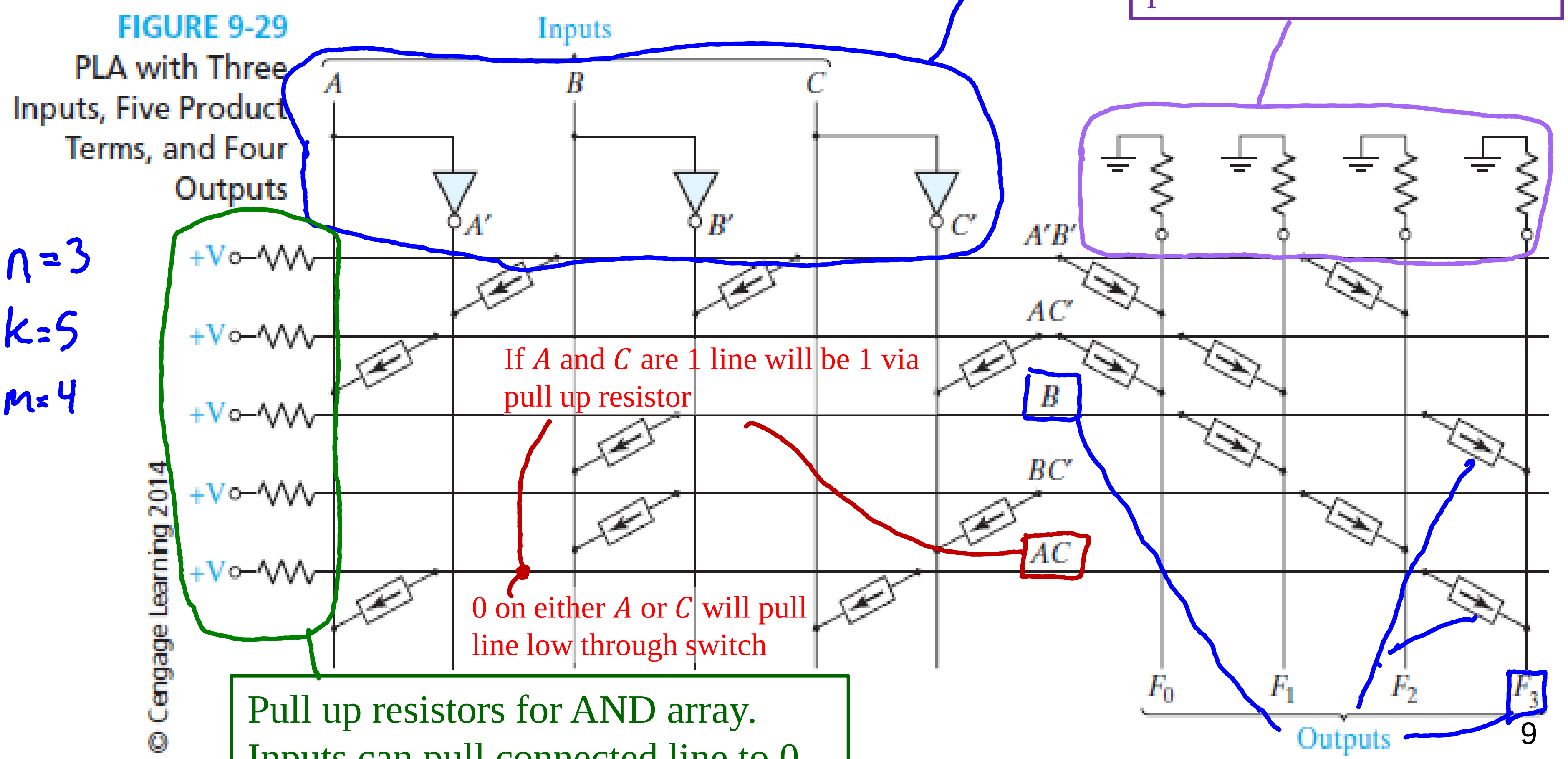
- PLA with AND and OR arrays programmed
  - Switch connections in AND array on left
  - OR array on right

Inputs and complements both available

Pull down resistors for OR array. Any 1 output from AND array can pull connected line to 1.

FIGURE 9-29
PLA with Three Inputs, Five Product Terms, and Four Outputs

n=3
k=5
m=4



If A and C are 1 line will be 1 via pull up resistor

0 on either A or C will pull line low through switch

Pull up resistors for AND array. Inputs can pull connected line to 0.

Programmable Logic Arrays

PLA Tables

- Symbols 0, l, and – indicate whether a variable is
  - Complemented,
  - Not complemented, or
  - Not present in the corresponding product term
- Output side of table specifies which product terms appear in each output function
- A 1 or 0 indicates whether a given product term is present or not present in the corresponding output function

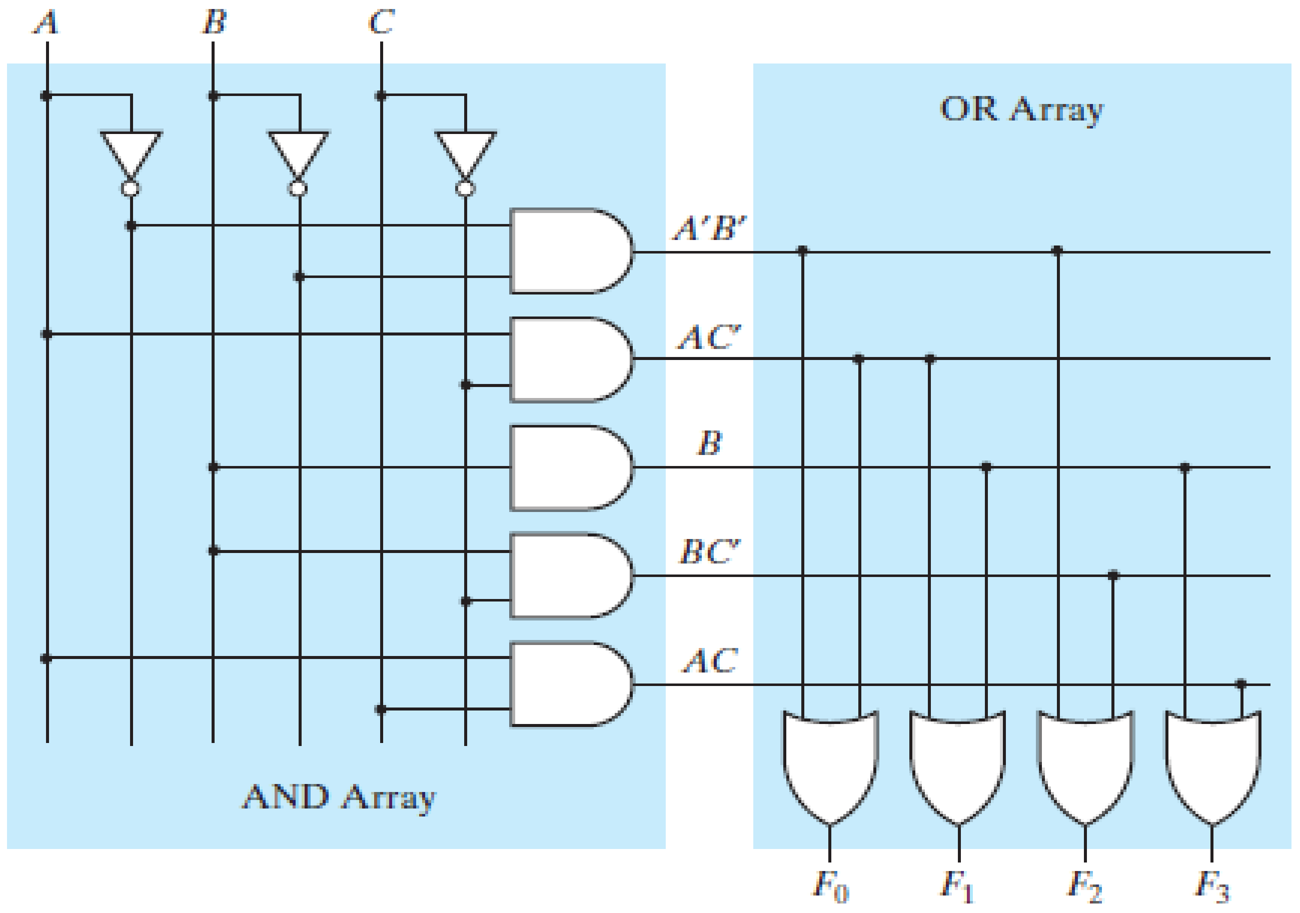
AND array　　OR array

PLA table for entire example array on previous slide

Product Term	Inputs A B C	Outputs F_0	F_1	F_2	F_3	
$A'B'$	0 0 –	1	0	1	0	$F_0 = A'B' + AC'$
AC'	1 – 0	1	1	0	0	$F_1 = AC' + B$
B	– 1 –	0	1	0	1	$F_2 = A'B' + BC'$
BC'	– 1 0	0	0	1	0	$F_3 = B + AC$
AC	1 – 1	0	0	0	1	

Programmable Logic Arrays

FIGURE 9-30
AND-OR Array
Equivalent to
Figure 9-29

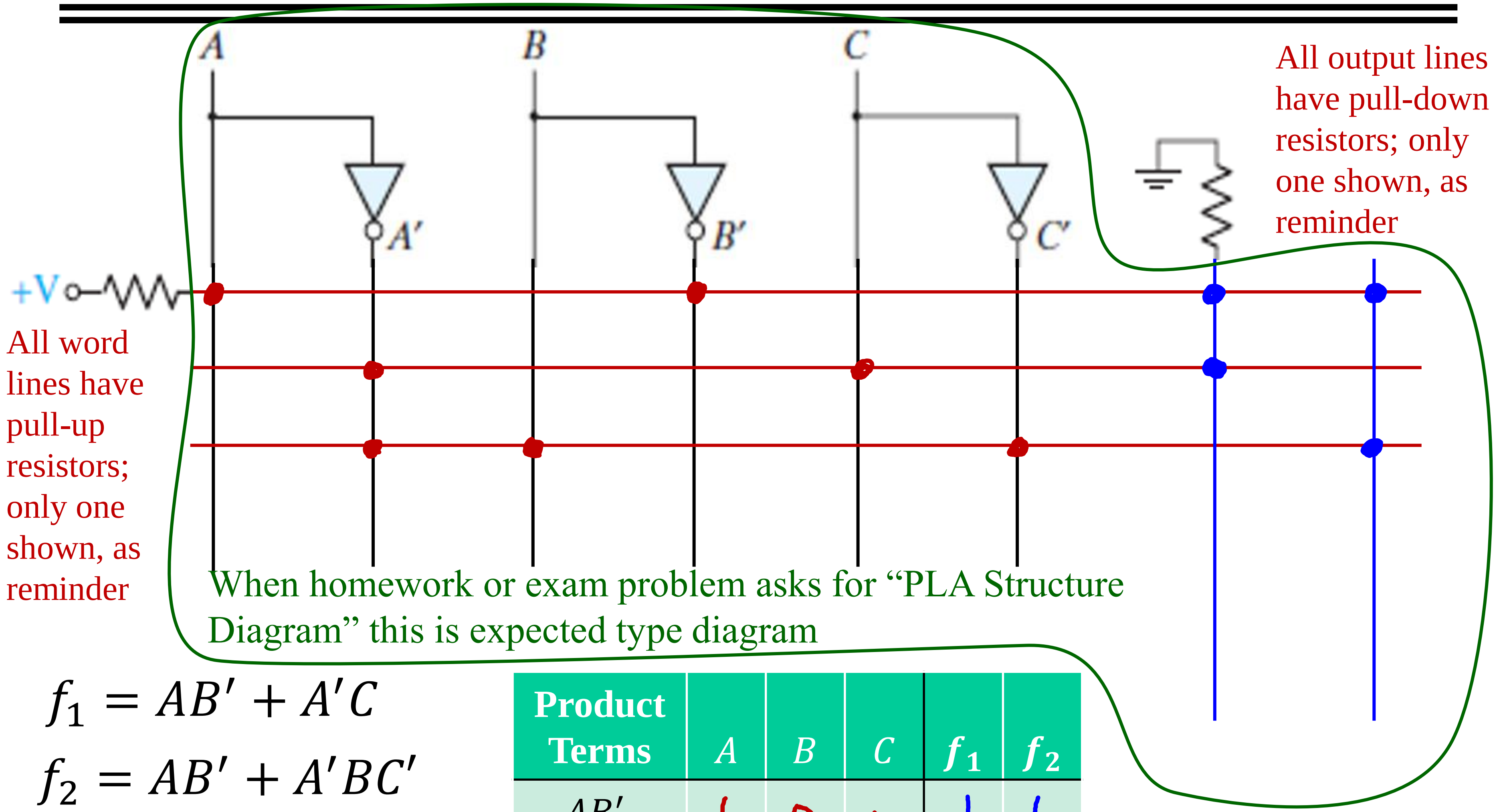


AND-OR gate array for entire example array on previous slides

Programmable Logic Arrays

- ROM
  - Decoder iterate all minterms
  - 24 inputs $\rightarrow 2^{24}$ word lines !!!
- PLA
  - Reduce expression first
  - Implement only those word lines that are needed

PLA Example



All output lines have pull-down resistors; only one shown, as reminder

All word lines have pull-up resistors; only one shown, as reminder

When homework or exam problem asks for "PLA Structure Diagram" this is expected type diagram

$$f_1 = AB' + A'C$$

$$f_2 = AB' + A'BC'$$

When homework or exam problem asks for "PLA Table" this is expected type of table

Product Terms	A	B	C	f_1	f_2
AB'	1	0	-	1	1
$A'C$	0	-	1	1	
$A'BC'$	0	1	0		1

Programmable Array Logic

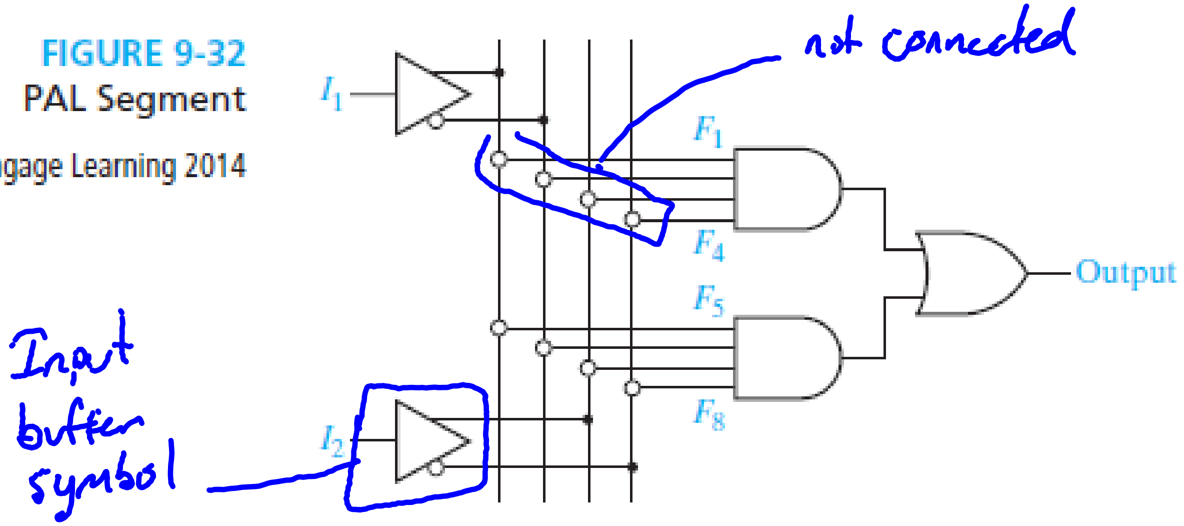
- Programmable Array Logic (PAL) is a special case of PLAs
- Same AND-array and OR-array structure, but
  - Only the AND array is programmable
  - The OR array is fixed
- Less expensive than the more general PLA
- Easier to program
- Input buffers with both inverting and non-inverting outputs used
  - Need both inverting and non-inverting
  - And each typically must drive many AND gates
- Limited Fan-in
- Different types available, with different Fan-in limits

ECE2060

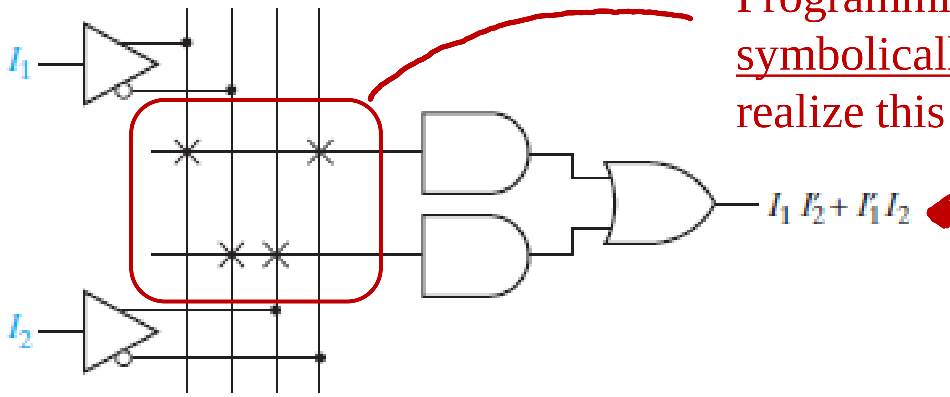
Programmable Array Logic

- Segment of an unprogrammed PAL
- This type has 2-input ORs

FIGURE 9-32
PAL Segment



(a) Unprogrammed



Programming of AND-array symbolically marked, to realize this function

(b) Programmed

Latches & Flip-Flops

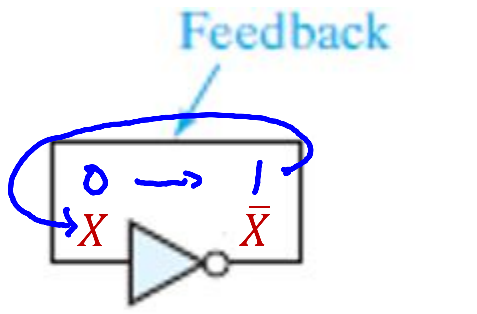
- Property of sequential switching circuits that output depends
    - Not only on present input, but
    - Also on past sequence of inputs.
- Circuits must be able to "remember" something about past history of the inputs in order to produce present output
- Latches and flip-flops are commonly used in sequential circuits
- Latches and flip-flops are memory devices that
    - Can assume one of two stable   , and
    - Have one or more inputs that can cause the output state to change

Basic difference between latches and flip-flops

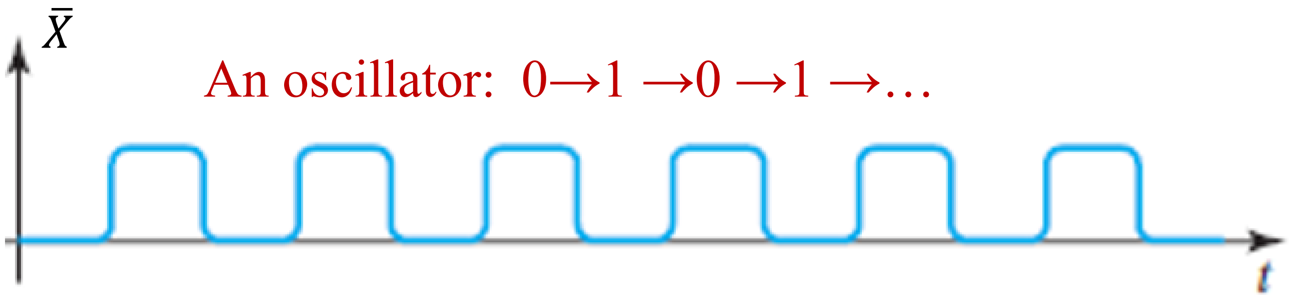
- For synchronous digital systems: common practice is to synchronize operation of all flip-flops by a common pulse signal
- Each flip-flop has a clock input
- Flip-flops are memory devices that can only change output in response to a clock input, not data inputs
  - Data inputs determine what the output state will be
  - Clock triggers when the change to that output state happens
- A memory element that has no clock input is typically called a latch

Feedback

- Logic circuits we have studied so far have not had feedback
- By feedback we mean that output of one of the gates is connected back into the input of another gate so as to form a closed loop



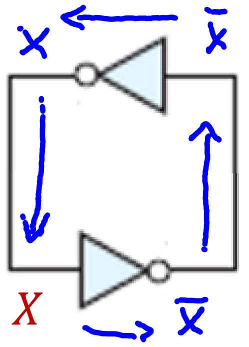
Inverter with feedback



An oscillator: $0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow \ldots$

- Known as a ring-oscillator
- Any odd number n of inverters with output of last fed back to input of first
- Oscillation period $\sim n \times$ (average propagation time)

Latches



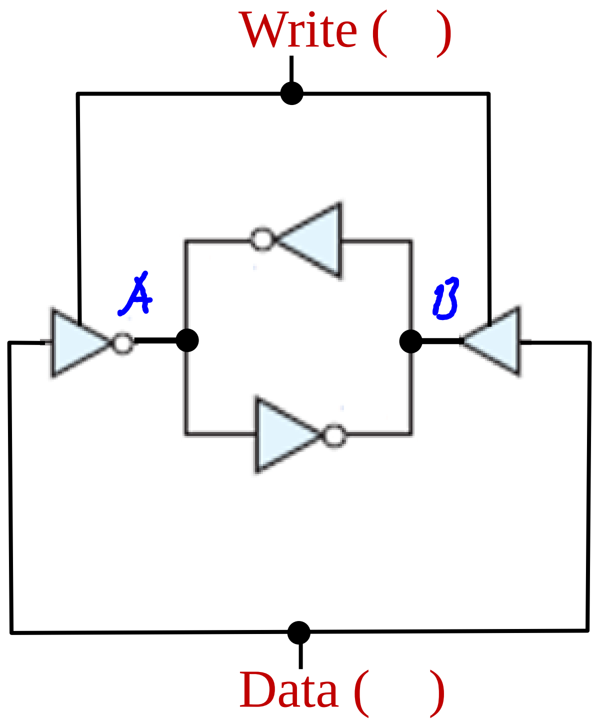
Two stable conditions

- $X = 1$ or $X = 0$
- Also known as stable states
- But need a way to change the state

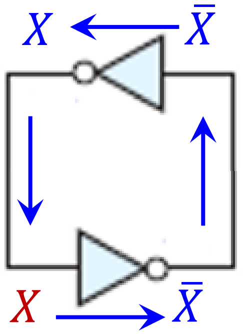
Write-Enable Latch

$$WRITE = 1 \begin{Bmatrix} \bar{D} \rightarrow A \\ D \rightarrow B \end{Bmatrix}$$

$$WRITE = 0 \begin{Bmatrix} \bar{A} \rightarrow B \\ \bar{B} \rightarrow A \end{Bmatrix}$$



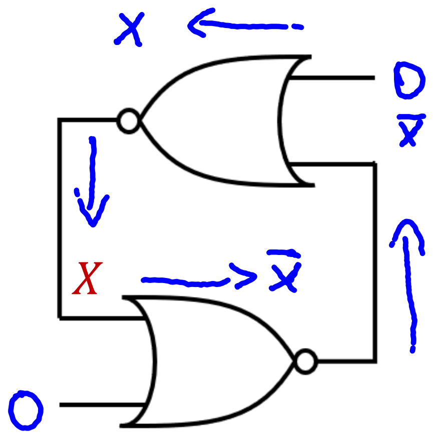
Latches



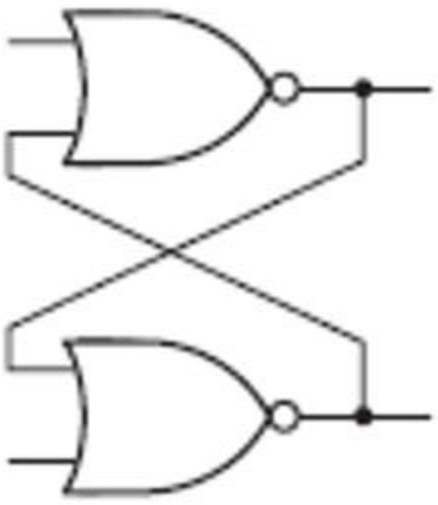
Two stable conditions

- $X = 1$ or $X = 0$
- Also known as stable states
- But need a way to change the state

Replace Inverters with NOR gates



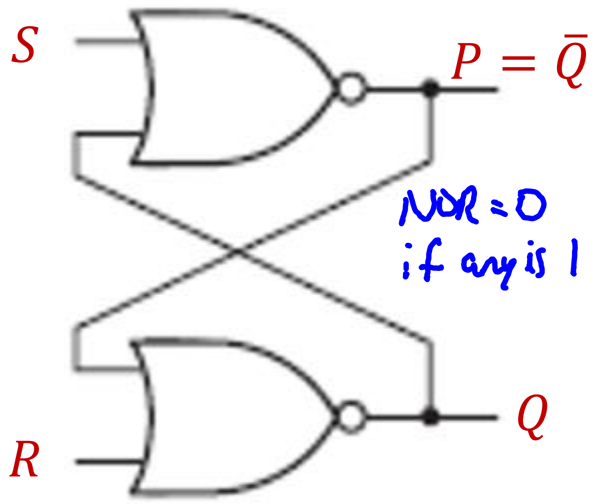
- NOR operates as inverter when one input wired to 0
- Flip orientation of top NOR
- And what happens if we change the values at extra inputs?



Set-Reset Latch

Set-Reset Latch (SR Latch)

- Can be made from two NOR gates with feedback in Cross-coupled form
- Term [$Q(t)$] used to denote state of Q output of the latch or flip-flop at time any input signal changes
- Term [$Q(t+\varepsilon)$] denotes state of Q output after latch or flip-flop has reacted to input change and stabilized
- Normally we will write next state with + superscript, without explicitly showing $t+\varepsilon$

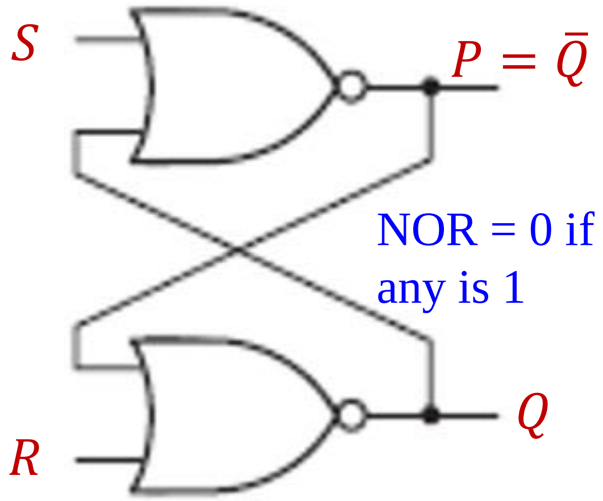


Present State				Next State	
S	R	Q	P	Q^+	P^+
0	0	0	1		
0	0	1	0		
0	1	0	1		
0	1	1	0		
1	0	0	1		
1	0	1	0		

Set-Reset Latch

Set-Reset Latch (SR Latch)

$$\left.\begin{aligned} S &= \\ R &= 1 \end{aligned}\right\}$$ BAD!



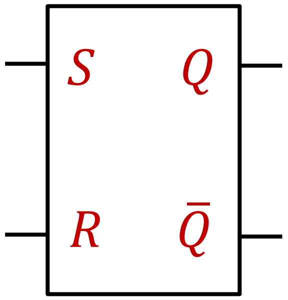
- Makes both $P = Q = 0$, but then $P \neq \bar{Q}$
- If S and R are simultaneously changed to 0, both P and Q may both change to 1
- The 1s propagate through the NOR gates changing P and Q to 0
- Latch may continue to oscillate if the gate delays are equal
- : **$S = R = 1$ must be avoided**

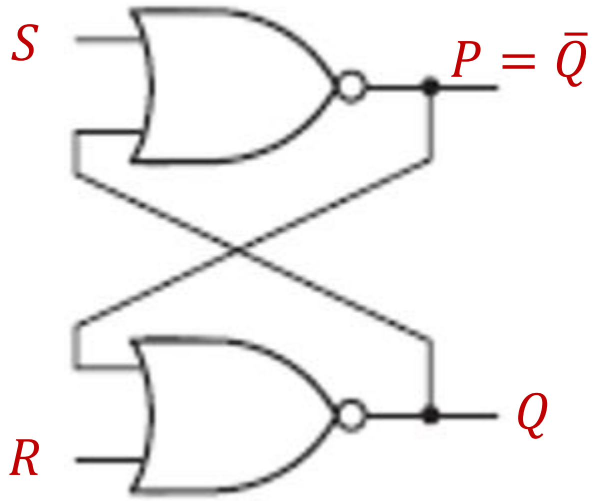
The designer of the circuit using the SR Latch is responsible for avoiding it!

Set-Reset Latch

Set-Reset Latch (SR Latch)

SR Latch Component





- $S = 0, R = 0$
- $S = 0, R = 1$
- $S = 1, R = 0$
- $S = 1, R = 1$ Do Not Allow

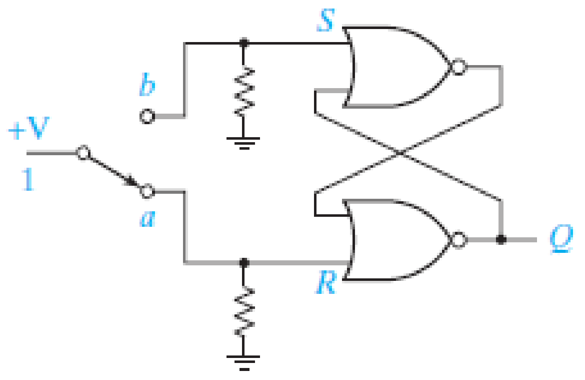
- Note: No clock input in component figure
- We will later use SR flip-flops, which will have a clock input
- That is how you can tell them apart

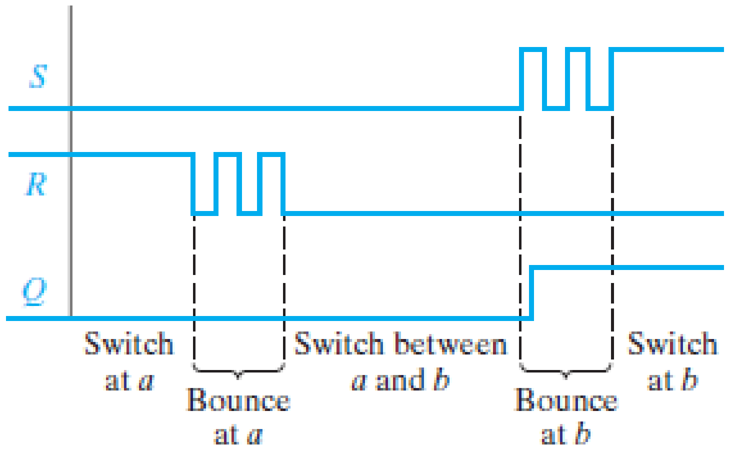
Set-Reset Latch

Set-Reset Latch (SR Latch)

- A useful application of the SR Latch is switch debouncing
- Switch contacts tend to vibrate open and closed several times before settling to their final position
- Produces an electronically noisy transition
- This noise can interfere with proper operation of a logic circuit

FIGURE 11-9
Switch Debouncing with an S-R Latch



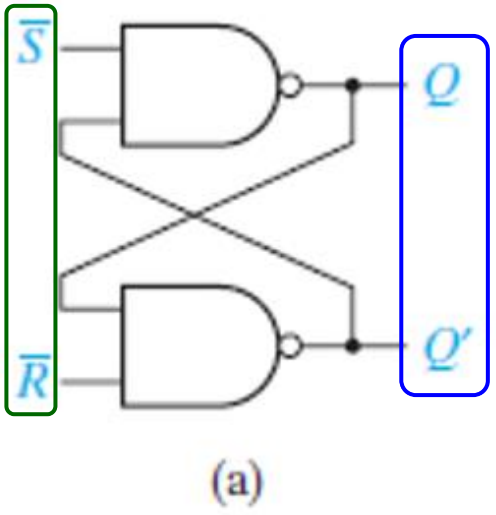


- This debouncing circuit works for double-throw switches (position *a* or position *b*)
- Debouncing a single-throw switch requires a different circuit

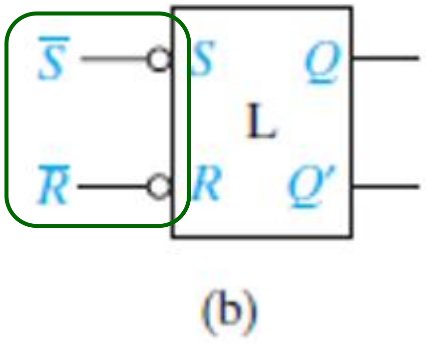
$\bar{S}$-$\bar{R}$ Latch

A NAND version is also possible

$\bar{S}$– $\bar{R}$ Latch Circuit



SR Latch Component



Note swapped outputs and complemented inputs

Present State → Q; Next State → Q^+

$\bar{S}$	$\bar{R}$	Q	Q^+
1	1	0	0
1	1	1	1
1	0	0	0
1	0	1	0
0	1	0	1
0	1	1	1
0	0	0	– } Inputs not allowed
0	0	1	–

- $\bar{S} = 1, \bar{R} = 1$
- $\bar{S} = 0, \bar{R} = 1$
- $\bar{S} = 1, \bar{R} = 0$
- $\bar{S} = 0, \bar{R} = 0$ Unstable: Do Not Allow