Chapter 2 :: Combinational Logic Design

Digital Design and Computer Architecture

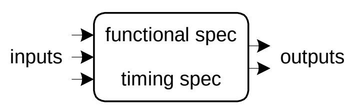
David Money Harris and Sarah L. Harris

Chapter 2 :: Topics

Introduction

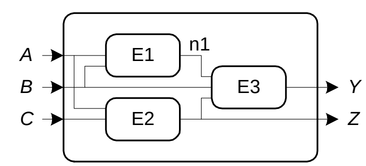
A logic circuit is composed of:

- Inputs
- Outputs
- Functional specification
- Timing specification



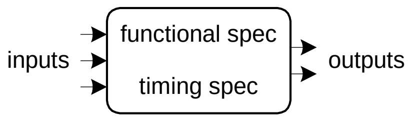
Circuits

- Nodes
  - Inputs: *A*, *B*, *C*
  - Outputs: *Y*, *Z*
  - Internal: n1
- Circuit elements
  - E1, E2, E3
  - Each a circuit



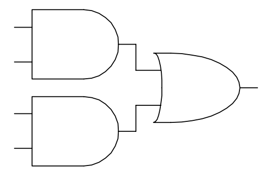
Types of Logic Circuits

- Combinational Logic
  - Memoryless
  - Outputs determined by current values of inputs
- Sequential Logic
  - Has memory
  - Outputs determined by previous and current values of inputs



Rules of Combinational Composition

- Every circuit element is itself combinational
- Every node of the circuit is either designated as an input to the circuit or connects to exactly one output terminal of a circuit element
- The circuit contains no cyclic paths: every path through the circuit visits each circuit node at most once
- Example:

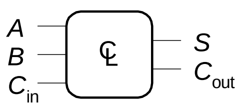


Boolean Equations

- Functional specification of outputs in terms of inputs
- Example:

$S = \mathrm{F}(A, B, C_{in})$

$C_{out} = \mathrm{F}(A, B, C_{in})$



$S = A \oplus B \oplus C_{in}$

$C_{out} = AB + AC_{in} + BC_{in}$

Sum-of-Products (SOP) Form

- All Boolean equations can be written in SOP form
- Each row in a truth table has a minterm
- A minterm is a product (AND) of literals
- Each minterm is TRUE for that row (and only that row)
- The function is formed by ORing the minterms for which the output is TRUE
- Thus, a sum (OR) of products (AND terms)

A	B	Y	minterm
0	0	0	$\overline{A}\,\overline{B}$
0	1	1	$\overline{A}\,B$
1	0	0	$A\,\overline{B}$
1	1	1	$A\,B$

$Y = \mathrm{F}(A, B) =$

Product-of-Sums (POS) Form

- All Boolean equations can be written in POS form
- Each row in a truth table has a maxterm
- A maxterm is a sum (OR) of literals
- Each maxterm is FALSE for that row (and only that row)
- The function is formed by ANDing the maxterms for which the output is FALSE
- Thus, a product (AND) of sums (OR terms)

A	B	Y	maxterm
0	0	0	$A + B$
0	1	1	$A + \overline{B}$
1	0	0	$\overline{A} + B$
1	1	1	$\overline{A} + \overline{B}$

$$Y = F(A, B) = (A + B)(\overline{A} + B)$$

Boolean Equations Example

- You are going to the cafeteria for lunch
  - You won't eat lunch ($\overline{E}$)
  - If it's not open ($\overline{O}$) or
  - If they only serve corndogs (C)
- Write a truth table for determining if you will eat lunch (E).

O	C	E
0	0	
0	1	
1	0	
1	1	

SOP & POS Form

- SOP – sum-of-products

O	C	E	minterm
0	0	0	$\overline{O}\ \overline{C}$
0	1	0	$\overline{O}\ C$
1	0	1	$O\ \overline{C}$
1	1	0	$O\ C$

$$Y = O\overline{C}$$

- POS – product-of-sums

O	C	E	maxterm
0	0	0	$O + C$
0	1	0	$O + \overline{C}$
1	0	1	$\overline{O} + C$
1	1	0	$\overline{O} + \overline{C}$

$$Y = (O + C)(O + \overline{C})(\overline{O} + \overline{C})$$

Boolean Algebra

- Set of axioms and theorems to simplify Boolean equations
- Like regular algebra, but in some cases simpler because variables can have only two values (1 or 0)
- Axioms and theorems obey the principles of duality:
  - ANDs and ORs interchanged, 0's and 1's interchanged

Boolean Axioms

	Axiom		Dual	Name
A1	$B = 0$ if $B \neq 1$	A1′	$B = 1$ if $B \neq 0$	Binary field
A2	$\overline{0} = 1$	A2′	$\overline{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	A3′	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	A4′	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5′	$1 + 0 = 0 + 1 = 1$	AND/OR

	Theorem		Dual	Name
T1	$B \bullet 1 = B$	T1′	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2′	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3′	$B + B = B$	Idempotency
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	T5′	$B + \overline{B} = 1$	Complements

T1: Identity Theorem

- B • 1 =
- B + 0 =

T2: Null Element Theorem

- B • 0 =
- B + 1 =

T3: Idempotency Theorem

- B • B =
- B + B =

T4: Identity Theorem

- $\overline{\overline{B}} =$

T5: Complement Theorem

- $B \bullet \overline{B} =$
- $B + \overline{B} =$

Boolean Theorems: Summary

	Theorem		Dual	Name
T1	$B \bullet 1 = B$	T1′	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2′	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3′	$B + B = B$	Idempotency
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	T5′	$B + \overline{B} = 1$	Complements

Boolean Theorems of Several Variables

	Theorem		Dual	Name
T6	$B \bullet C = C \bullet B$	T6′	$B + C = C + B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	T7′	$(B + C) + D = B + (C + D)$	Associativity
T8	$(B \bullet C) + B \bullet D = B \bullet (C + D)$	T8′	$(B + C) \bullet (B + D) = B + (C \bullet D)$	Distributivity
T9	$B \bullet (B + C) = B$	T9′	$B + (B \bullet C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	T10′	$(B + C) \bullet (B + \overline{C}) = B$	Combining
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D)$ $= B \bullet C + \overline{B} \bullet D$	T11′	$(B + C) \bullet (\overline{B} + D) \bullet (C + D)$ $= (B + C) \bullet (\overline{B} + D)$	Consensus
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots}$ $= (\overline{B_0} + \overline{B_1} + \overline{B_2} \ldots)$	T12′	$\overline{B_0 + B_1 + B_2 \ldots}$ $= (\overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2})$	De Morgan's Theorem

Simplifying Boolean Expressions: Example 1

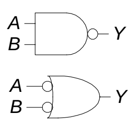
- $Y = \bar{A}B + AB$

Simplifying Boolean Expressions: Example 2

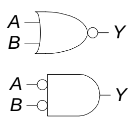
- $Y = A(AB + ABC)$

DeMorgan's Theorem

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

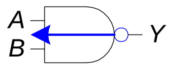


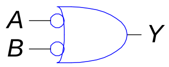
- $Y = \overline{A + B} = \bar{A} \bullet \bar{B}$



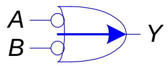
Bubble Pushing

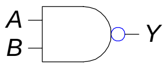
- Pushing bubbles backward (from the output) or forward (from the inputs) changes the body of the gate from AND to OR or vice versa.
- Pushing a bubble from the output back to the inputs puts bubbles on all gate inputs.





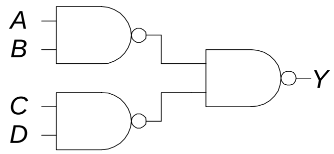
- Pushing bubbles on *all* gate inputs forward toward the output puts a bubble on the output and changes the gate body.





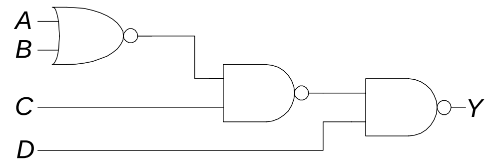
Bubble Pushing

- What is the Boolean expression for this circuit?

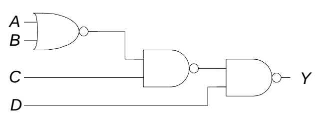


Bubble Pushing Rules

- Begin at the output of the circuit and work toward the inputs.
- Push any bubbles on the final output back toward the inputs.
- Draw each gate in a form so that bubbles cancel.

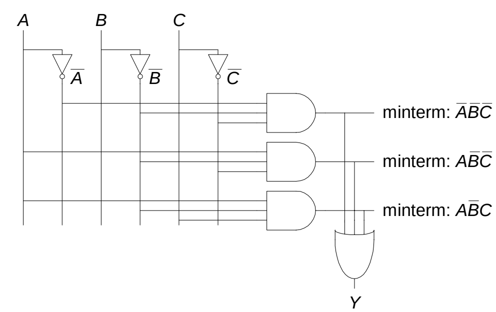


Bubble Pushing Example



From Logic to Gates

- Two-level logic: ANDs followed by ORs
- Example: $Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$

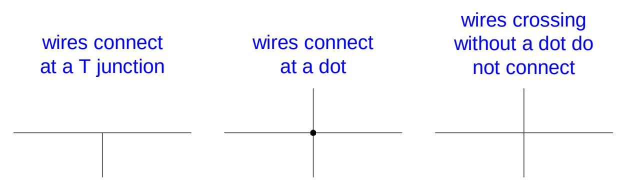


Circuit Schematics with Style

- Inputs are on the left (or top) side of a schematic
- Outputs are on the right (or bottom) side of a schematic
- Whenever possible, gates should flow from left to right
- Straight wires are better to use than wires with multiple corners

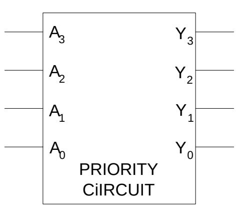
Circuit Schematic Rules (cont.)

- Wires always connect at a T junction
- A dot where wires cross indicates a connection between the wires
- Wires crossing *without* a dot make no connection



Multiple Output Circuits

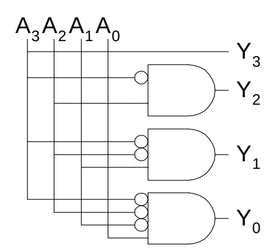
- Output asserted corresponding to most significant TRUE input



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	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				

Priority Encoder Hardware

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0



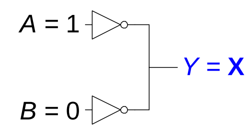
Don't Cares

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0

Contention: X

- Contention: circuit tries to drive the output to 1 and 0
  - Actual value may be somewhere in between
  - Could be a legal 0, a legal 1, or in the forbidden zone
  - Might change with voltage, temperature, time, noise
  - Often causes excessive power dissipation

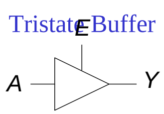


- Contention usually indicates a bug.
  - Fix it unless you are sure you know what you are doing.
- Warning: X is used for "don't care" and contention
  - Note the same thing
  - Look at the context to tell them apart

Floating: Z

- Floating, high impedance, open, high Z
- Floating output might be 0, 1, or somewhere in between
  - A voltmeter won't indicate whether a node is floating

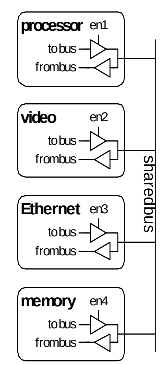
Tristate Buffer



E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

Tristate Busses

- Floating nodes are used in tristate busses
  - Many different drivers
  - Exactly one is active at any time



Karnaugh Maps (K-Maps)

- Boolean expressions can be minimized by combining terms
- K-maps minimize equations graphically
- $PA + P\bar{A} = P$

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

Y: C \ AB	00	01	11	10
0	1	0	0	0
1	1	0	0	0

Y: C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

K-map

- Circle 1's in adjacent squares
- In the Boolean expression, include only the literals whose true and complement form are *not* in the circle

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

Y: C \ AB	00	01	11	10
0	1	0	0	0
1	1	0	0	0

$$Y = \bar{A}\bar{B}$$

3-input K-map

Y: C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

Truth Table

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

K-Map

Y: C \ AB	00	01	11	10
0				
1				

K-map Definitions

- Complement: variable with a bar over it
  $\bar{A}, \bar{B}, \bar{C}$
- Literal: variable or its complement
  $A, \bar{A}, B, \bar{B}, C, \bar{C}$
- Implicant: product of literals
  $AB\bar{C}, \bar{A}C, BC$
- Prime implicant: implicant corresponding to the largest circle in a K-map

K-map Rules

- Every 1 in a K-map must be circled at least once
- Each circle must span a power of 2 (i.e. 1, 2, 4) squares in each direction
- Each circle must be as large as possible
- A circle may wrap around the edges of the K-map
- A “don't care” (X) is circled only if it helps minimize the equation

4-input K-map

Y

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

$$Y = \overline{A}C + \overline{A}BD + A\overline{B}\,\overline{C} + \overline{B}\,\overline{D}$$

4-input K-map

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Y

CD \ AB	00	01	11	10
00				
01				
11				
10				

K-maps with Don’t Cares

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Y

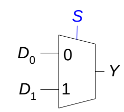
CD \ AB	00	01	11	10
00				
01				
11				
10				

Combinational Building Blocks

- Multiplexers
- Decoders

Multiplexer (Mux)

- Selects between one of N inputs to connect to the output.
- $\log_2 N$-bit select input – control input
- Example: 2:1 Mux



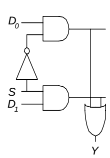
S	D1	D0	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Multiplexer Implementations

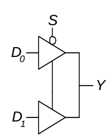
- Logic gates
  - Sum-of-products form

S \ D_0D_1	00	01	11	10
0	0	0	1	1
1	0	1	1	0

$Y = D_0\overline{S} + D_1S$



- Tristates
  - For an N-input mux, use N tristates
  - Turn on exactly one to select the appropriate input

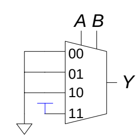


Logic using Multiplexers

- Using the mux as a lookup table

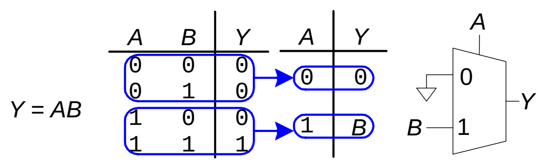
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = AB$



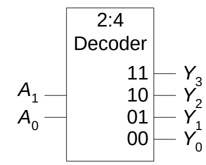
Logic using Multiplexers

- Reducing the size of the mux



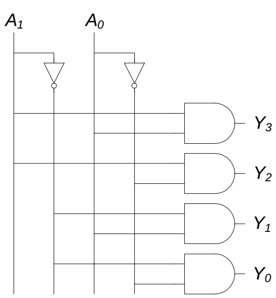
Decoders

- N inputs, 2^N outputs
- One-hot outputs: only one output HIGH at once



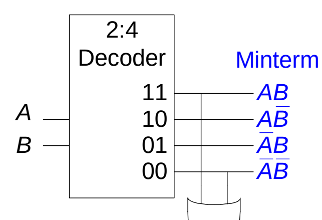
A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Decoder Implementation



Logic using Decoders

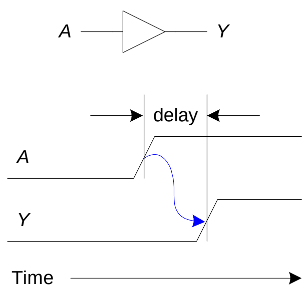
- OR minterms



$$Y = AB + \overline{A}\,\overline{B} = \overline{A \oplus B}$$

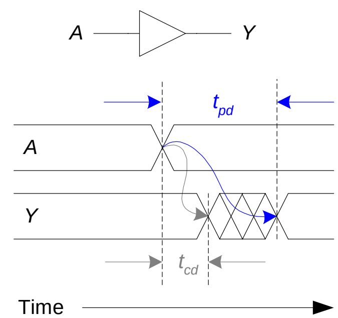
Timing

- Delay between input change and output changing
- How to build fast circuits?



Propagation & Contamination Delay

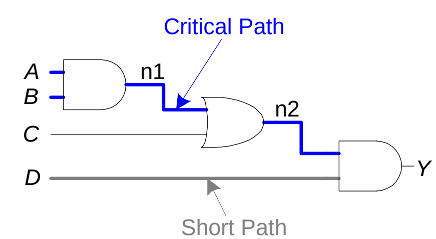
- Propagation delay: t_{pd} = max delay from input to output
- Contamination delay: t_{cd} = min delay from input to output



Propagation & Contamination Delay

- Delay is caused by
  - Capacitance and resistance in a circuit
  - Speed of light limitation
- Reasons why t_{pd} and t_{cd} may be different:
  - Different rising and falling delays
  - Multiple inputs and outputs, some of which are faster than others
  - Circuits slow down when hot and speed up when cold

Critical and Short Paths



Critical (Long) Path: $t_{pd} = 2t_{pd\_AND} + t_{pd\_OR}$

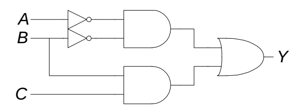
Short Path: $t_{cd} = t_{cd\_AND}$

Glitches

- A *glitch* occurs when a single input change causes multiple output changes

Glitch Example

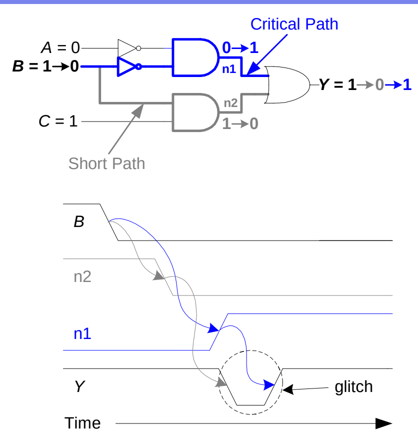
- What happens when A = 0, C = 1, B falls?



Y (C \ AB)	00	01	11	10
0	1	0	0	0
1	1	1	1	0

$Y = \bar{A}\bar{B} + BC$

Glitch Example (cont.)

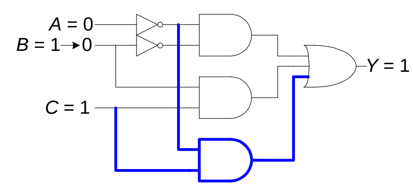


Fixing the Glitch

Y (C \ AB)	00	01	11	10
0	1	0	0	0
1	1	1	1	0

$\bar{A}C$

$Y = \bar{A}\bar{B} + BC + \bar{A}C$



Why Understand Glitches?

- Glitches don't cause problems because of synchronous design conventions (which we'll talk about in Chapter 3)
- But it's important to recognize a glitch when you see one in simulations or on an oscilloscope
- Can't get rid of all glitches – simultaneous transitions on multiple inputs can also cause glitches