Combinational Logic DDCA 1, 2

Monday, January 11, 2021 1:17 PM

2.1 Intro

- A circuit is combinational if consist of interconnected circuit elements such that
  - every circuit element is combinational
  - every node of circuit is designated as an input to the circuit or connects to exactly one output terminal of a circuit element
  - circuit contains no cyclic paths: every path through the circuit visits each circuit node at most once

2.2 Boolean Equations

- vars TRUE or FALSE
- complement of A is its inverse (A w/ - on top, or A')
- AND of one or more literals is a product or implicant
  - A'B, AB'C' and B are all implicants for a function of 3 vars
- minterm is a product involving all of the inputs to the function
  - AB'C' is a minterm for a func of 3 var A,B,C, but A'B is not, does not involve C
- OR of on or more literals is called a sum
- maxterm is a sum involving all of the inputs to the function
  - A+B'+C is a maxterm for a func of 3 var A,B,C
- **Order of Operations**
  - NOT has highest precedence, AND, OR

    $\overline{A}B + BC\overline{D} = ((\overline{A})B) + (BC(\overline{D}))$
- **Sum of Products Form**
  - **Truth Tables**

A	B	Y	minterm	minterm name
0	0	0	$\overline{A}\,\overline{B}$	m_0
0	1	1	$\overline{A}\,B$	m_1
1	0	0	$A\,\overline{B}$	m_2
1	1	0	$A\,B$	m_3

A	B	Y	minterm	minterm name
0	0	0	$\overline{A}\,\overline{B}$	m_0
0	1	1	$\overline{A}\,B$	m_1
1	0	0	$A\,\overline{B}$	m_2
1	1	1	$A\,B$	m_3

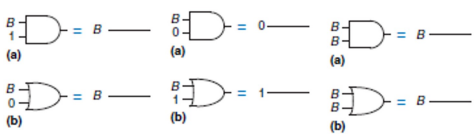
  - Write Boolean equation for any truth table by summing each fo the minterms for which output Y is TRUE
    - Y=A'B
    - Y=A'B+AB
  - Sum of products canonical form (sum OR of products ANDs forming minterms
- **Product of Sums**
  - Each row of a truth table corresponds to a maxterm that is FALSE for that row

A	B	Y	maxterm	maxterm name
0	0	0	$A + B$	M_0
0	1	1	$A + \overline{B}$	M_1
1	0	0	$\overline{A} + B$	M_2
1	1	1	$\overline{A} + \overline{B}$	M_3

  - 2 rows in which the output is FALSE
  - Product of sums form: Y = (A+B)(A'+B)

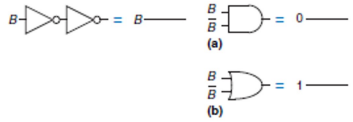
2.3 Boolean Algebra

	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



- **Theorems of One Variable**

	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



- **Theorems of Several Variables**

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

- **De Morgan's Theorem**
  - complement of product of all terms is equal to sum of complement of each term
  - NAND gate = OR gate with inverted inputs

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

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

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

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

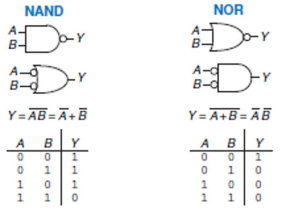


Figure 2.19 De Morgan equivalent gates

- **Simplifying Equations**
  - use theorems to simplify Boolean eqns
  - Y=A'B'+AB' simplifies to Y=B'
  - An eqn is minimized if it uses the fewest possible implicants
  - An implicant is called a prime implicant if it cannot be combined with any other implicants
    - A'B'C'+AB'C'+AB'C can be simplified to two prime implicants: B'C'+AB'
      - B'C' and AB' share the minterm AB'C'

Table 2.4 Equation minimization

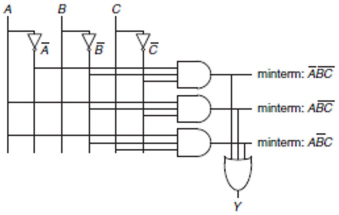
Step	Equation	Justification
	$\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$	
1	$\overline{B}\,\overline{C}(\overline{A} + A) + A\overline{B}C$	T8: Distributivity
2	$\overline{B}\,\overline{C}(1) + A\overline{B}C$	T5: Complements
3	$\overline{B}\,\overline{C} + A\overline{B}C$	T1: Identity

Table 2.5 Improved equation minimization

Step	Equation	Justification
	$\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$	
1	$\overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}C$	T3: Idempotency
2	$\overline{B}\,\overline{C}(\overline{A} + A) + A\overline{B}(\overline{C} + C)$	T8: Distributivity
3	$\overline{B}\,\overline{C}(1) + A\overline{B}(1)$	T5: Complements
4	$\overline{B}\,\overline{C} + A\overline{B}$	T1: Identity

2.4 From Logic to Gates

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



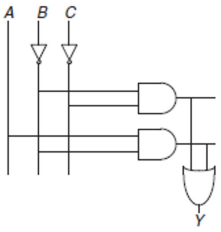


Figure 2.25 Schematic of $Y = \overline{B}\,\overline{C} + A\overline{B}$

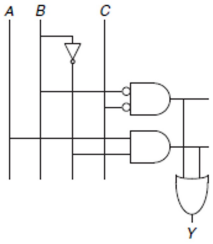
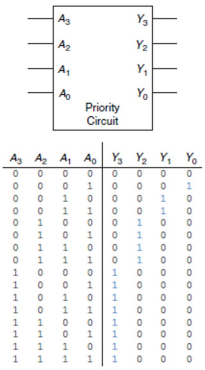


Figure 2.26 Schematic using fewer gates

- schematics consistant drawings:
  - inputs left or top side
  - output on right or bottom side
  - gates flow left to right
  - use straight wires
  - wires connect at T junction
  - a dot where wire cross = connection bw wires
  - wires without dot = no connection

- Priority circuit
  - four inputs A3,…,A0 and four outputs, Y3,…,Y0



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

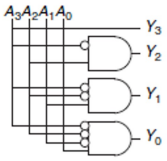


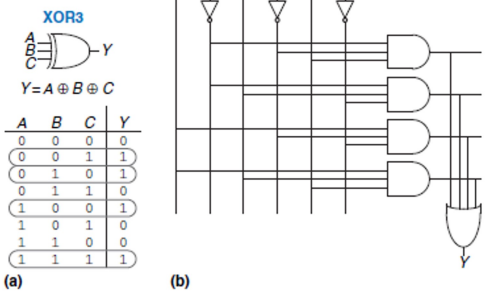
Figure 2.28 Priority circuit schematic

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

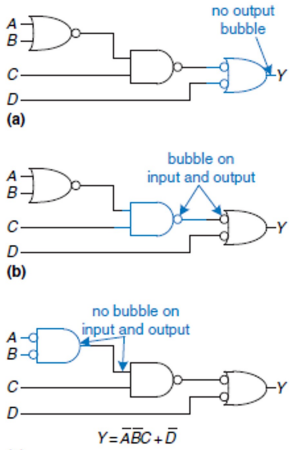
Figure 2.29 Priority circuit truth table with don't cares (X's)

2.5 Multilevel Combinational Logic

- Hardware reduction
  - XOR function: TRUE when odd number of inputs TRUE
  - No way to simplify
    $Y = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC$
    into fewer implicants



- **Bubble Pushing**
  - CMOS circuits prefer NANDs and NORs over ANDs and Ors
  - redraw crcutis so that bubbles cancel out and func can be more easily determined
    - Begin output of circuit and work towards inputs
    - push abubbles on final output back toward the inputs so that can read equation in terms fo output
    - working backward, draw gate in form so that bubbles cancel



2.6 X's and Z's, OH MY

- illegal and floating values
- **Illegal value X**
  - unknown or illegal value
  - if being driven to both 0 and 1 at the same time
  - contention, error to be avoided
  - voltage between 0 and V_DD

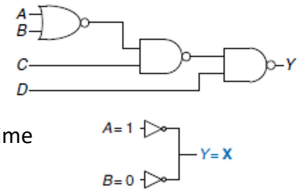


Figure 2.39 Circuit with contention

- **Floating Value Z**
  - node neither HIGH nor LOW
  - floating, high impedance, or high Z
  - might be 0 or might be 1 or voltage in between depending on history of sys
  - if forget to connect a voltage to circuit input or assume that unconnected input is same as input with value of 0
  - **Tristate buffer**
    - 3 possible output states: HIGH (1), LOW (0) and floating (Z)
    - input A, output Y, enable E
    - when enable TRUE, buffer is simple buffer, transferring input value to output
    - When enable is FALSE, output allowed to float Z
    - for active high enable, when HIGH (1), buffer enabled
    - for active low enable, when LOW (0), buffer enabled

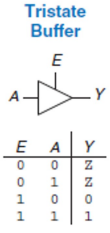
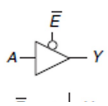


Figure 2.40 Tristate buffer

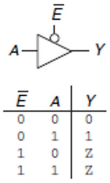
2.7 Karnaugh Maps (K-maps)

- truth table and K map for three input function
- top row = 4 possible values for A and B inputs



2.7 Karnaugh Maps (K-maps)

- truth table and K map for three input function
- top row = 4 possible values for A and B inputs
- left col = 2 possible values for C input
- each square corresponds to row in truth table and contains value of output Y=1 when ABC=000
- each square in K-map = single minterm
- K-map wraps around, square far right adjacent to squares far left, differ only in one var A



$\overline{E}$	A	Y
0	0	0
0	1	1
1	0	Z
1	1	Z

Figure 2.41 Tristate buffer with active low enable

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

(a)

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

(b)

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

(c)

Figure 2.43 Three-input function: (a) truth table, (b) K-map, (c) K-map showing minterms

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

Figure 2

- **Circular Thinking**
  - can use Boolean algebra to minimize equations in sum-of-products form
    $Y = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C = \overline{A}\,\overline{B}(\overline{C} + C) = \overline{A}\,\overline{B}$
  - K-maps help simplify graphically by circling 1's in adjacent squares
  - for each circle, write corresponding implicant (product of one or more literals)
- **Logic Minimization with K-Maps**
  - largest possible circles are prime implicants
  - **Rules**
    - use fewest circles necessary to cover all 1's
    - All squares in each circle must containt 1's
    - each circle must span a rectangular block that is a power of 2 (1, 2, 4, …) squares in each direction
    - each circle should be as large as possible
    - a circle may wrap around edges of K-map
    - A I in a K-map may be circled multiple times if doing so allows fewer circles

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

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

$Y = A\overline{C} + \overline{B}$

Table 2.6 Seven-segment display decoder truth table

$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
others	0	0	0	0	0	0	0

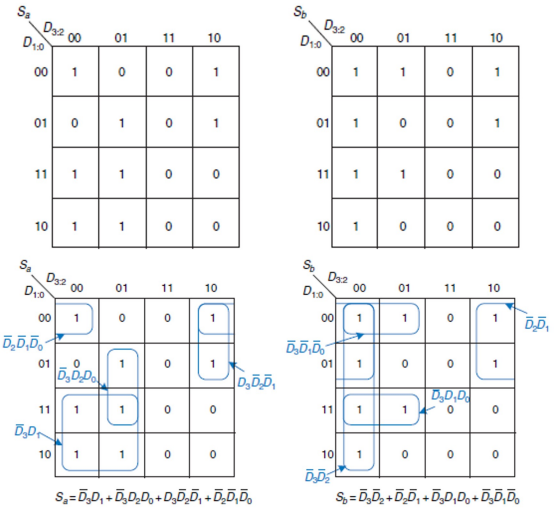


Figure 2.50 K-map solution for Example 2.10

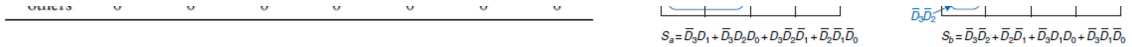
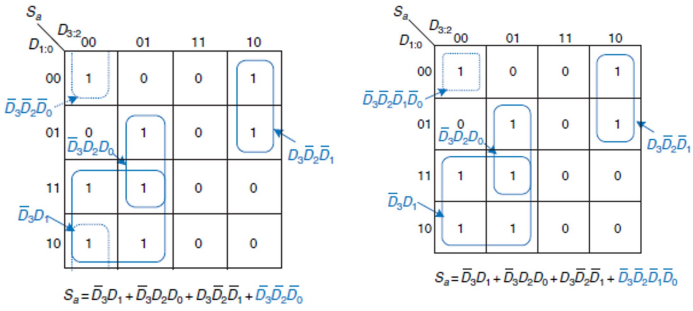
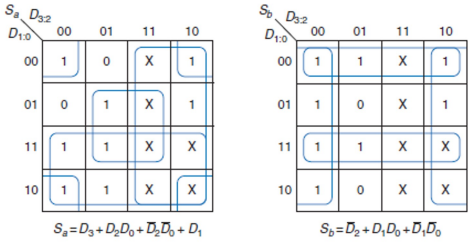


Figure 2.50 K-map solution for Example 2.10



- **Don't Cares**
  - use X
  - entry can be 0 or 1
  - output value unimportant or corresponding input combo can never happen



2.8 Combinational Building Blocks

- **Multiplexers**
  - choose an output from among several possible inputs based on the value of a select signal
  - MUX
    - 2:1 multiplexer two data inputs D0 D1, Select input S, output Y
  - S is a control signal, controls what multiplexer does
- **Wider Multiplexers**
  - 4:1 multiplexer has four data inputs and one output
  - 2 select signals needed to choose mong 4 data inputs
  - N:1 mux needs log_2(N) select lines

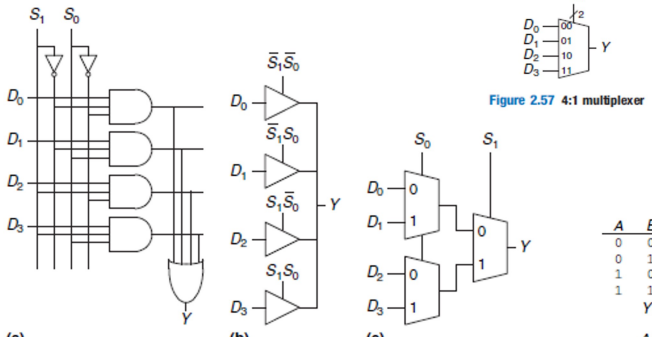


Figure 2.57 4:1 multiplexer

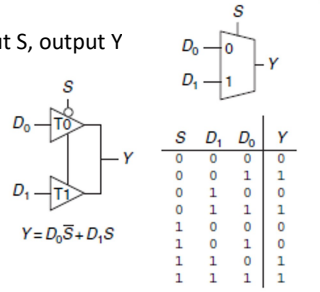
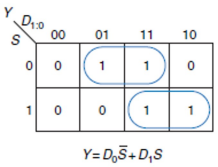
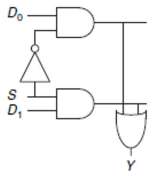


Figure 2.54 2:1 multiplexer symbol and truth table





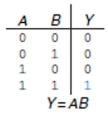
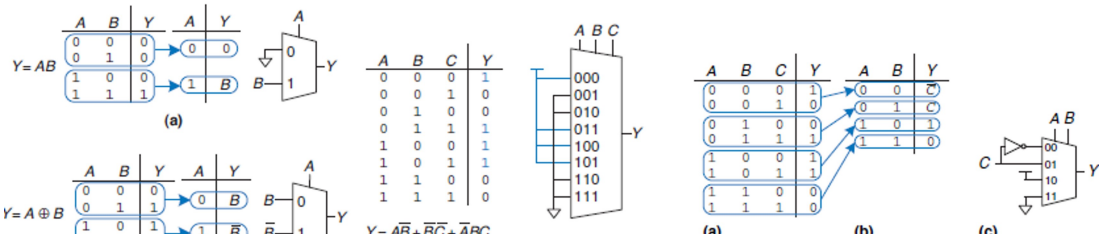
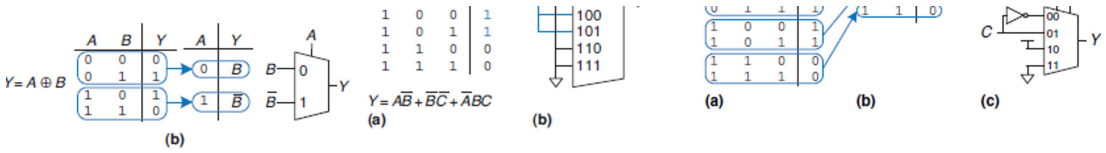


Figure 2.59 4:1 multiplexer implementation of two-input AND function

- **Multiplexer Logic**
  - can be used as lookup tables
  - mux can be used to make a two-input AND gate
    A and B inputs serve as select lines
    2^N input mux can be programmed to perform N-input logic function by applying 0s and 1s
  - can cut mx size in half using 2^(N-1) input mux to perform N-input logic func





- **Decoders**
  - has N inputs and 2^N outputs
  - asserts one of its outptus depending on input combination
  - when A_1:0=00, Y0=1, When A1:0=01, Y1=1
  - outputs are called one-hot bc exactly one is "hot" (HIGH) at given time
  - 2:4 decoder with AND, OR and NOT gates
  - **Decoder Logic**
    - can be combined with OR gates to build logic funcs
    - two-input XNOR func using 2:4 decoder and OR gate
    - each output of decoder=single minterm, built as OR of minterms

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

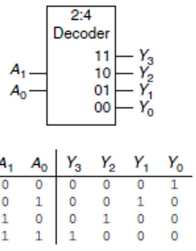


Figure 2.63 2:4 decoder

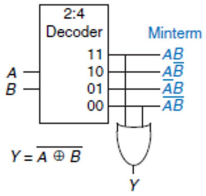


Figure 2.65 Logic function using decoder

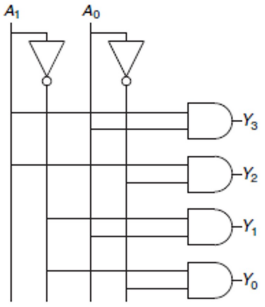
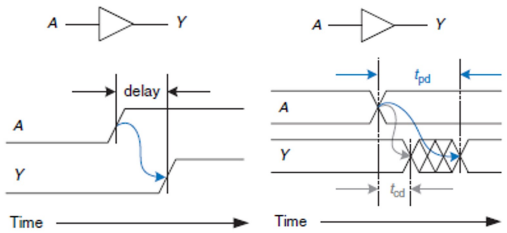
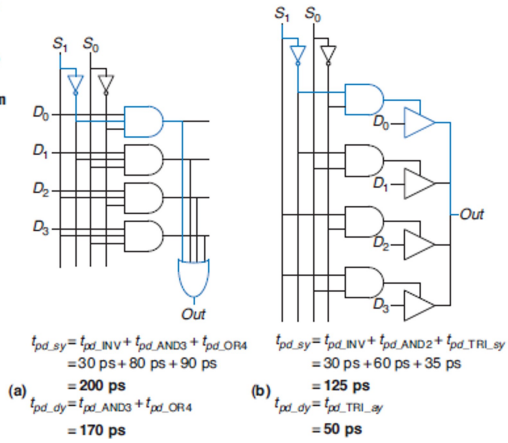


Figure 2.64 2:4 decoder implementation

2.9 Timing

- **Propagation and Contamination Delay**
  - Combinational logic characterized by prop and cont delays
  - t_pd is max time when input changes utnil output or outputs reach their final value
  - t_cd min time from when input changes until any output starts to change

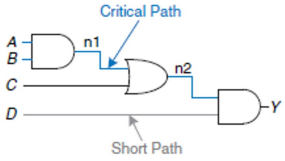




$t_{pd\_sy} = t_{pd\_INV} + t_{pd\_AND3} + t_{pd\_OR4}$
$= 30\text{ ps} + 80\text{ ps} + 90\text{ ps}$
$= 200\text{ ps}$ (a)
$t_{pd\_dy} = t_{pd\_AND3} + t_{pd\_OR4}$
$= 170\text{ ps}$

$t_{pd\_sy} = t_{pd\_INV} + t_{pd\_AND2} + t_{pd\_TRI\_sy}$
$= 30\text{ ps} + 60\text{ ps} + 35\text{ ps}$
$= 125\text{ ps}$ (b)
$t_{pd\_dy} = t_{pd\_TRI\_ay}$
$= 50\text{ ps}$

  - buffer propagation delay and contamination delay in blue and gray
  - A is HIGH or LOW, changes to other state at time
  - differences in delay may be different
    - different rising and falling delays
    - multiple inputs and outputs, som faster than others
    - circuits slowing down when hot and speeding up when cold



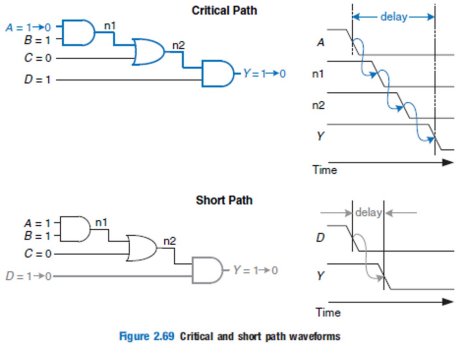
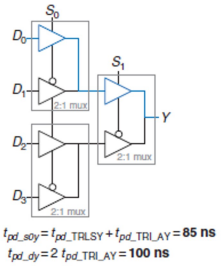


Figure 2.69 Critical and short path waveforms

- **Glitches**
  - possible that single input transition can cause multiple output transitions
  - called glitches or hazards



$t_{pd\_soy} = t_{pd\_TRLSY} + t_{pd\_TRI\_AY} = 85\text{ ns}$
$t_{pd\_dy} = 2\,t_{pd\_TRI\_AY} = 100\text{ ns}$

Figure 2.74 4:1 multiplexer propagation delays: hierarchical using 2:1 multiplexers

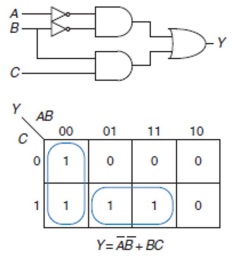
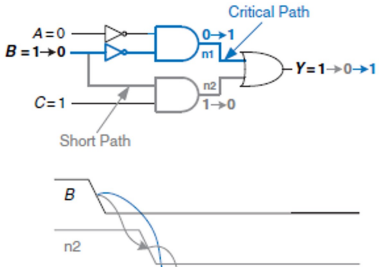


Figure 2.75 Circuit with a glitch



  - if A=0, C=1, B transitions from 1 to 0

$t_{pd\_s0y} = t_{pd\_TRLSY} + t_{pd\_TRI\_AY} = 85$ ns
$t_{pd\_dy} = 2\,t_{pd\_TRI\_AY} = 100$ ns

Figure 2.74 4:1 multiplexer propagation delays: hierarchical using 2:1 multiplexers

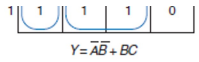
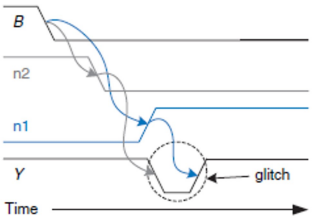


Figure 2.75 Circuit with a glitch



- if A=0, C=1, B transitions from 1 to 0
  - short path goes through AND and OR
  - critical path in blue goes through inverter and AND and OR
  - as B->0, n2 on short path falls before n1 on critical path rises
  - until n1 rises, Y drops to 0
  - when n1 rises, Y returns to 1
  - Y starts at 1, and ends at 1 but momentarily glitches to 0