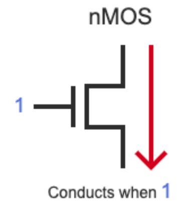
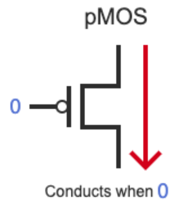
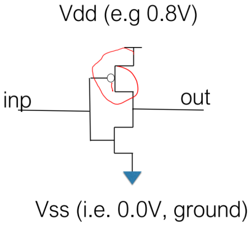
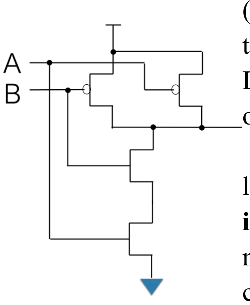




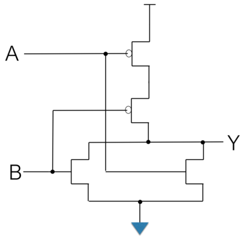




unsigned binary range: 0 to (2^n)-1; 1’s Complement: pos is 0 to 2^(n-1) and to get neg is invert bits of pos 2’s Complement range: -2^(n-1) to 2^(n-1)-1; **CMOS** uses both pMOS and nMOS; **NOT** gate: 2 transistors (pMOS on top, nMOS on bottom) → **AND/OR**: 6 transistors; **Two Level Logic:** **Sum of Products (Disjunctive) and Product of Sums (Conjunctive)**; **Boolean Algebra Properties:** Associative; (a•b)•c = a•(b•c), (a+b)+c=a+(b+c); Commutative: a•b=b•a, a+b=b+a; Distributive: a•(b+c)=a•b+a•c, a+(b•c)=(a+b)•(a+c); Identity: a•1=a, a+0=a; Complement: a•a’=0, a+a’=1; Uniting Theorem: ab+a’b=b, (a’+b)(a+b)=b; Absorption Theorem: a+ab=a, a(1+b)=a; If we can prove a statement using laws of Boolean algebra true, then the dual of the statement is also true: swap (+, ·) and complement all 0’s and 1’s; Distributive OR: X + YZ → (X+Y)(X+Z); Distributive AND: X + YZ → (X+Y)(X+Z); # of literals counts number of unique occurrences of variable **and** complement, # of variables is number of occurrences of literal, # pins = total # of input + output

Complement: variable with a “BAR” over it or ‘ after it⇒A’; **Literal:** variable or its complement; **Implicant:** product of literals⇒ABC; **Implicate:** sum of literals⇒(A+B+C); **Minterm/Maxterm:** implicant/implicate that **includes all the inputs**⇒f(A, B, C, D)=ABCD/(A’+B+C+D); **maxterm** is sum when 0, **minterm** is product when 1; SOP/POS minimizes number of implicants; **Canonical POS/SOP Form:** each term in SOP is minterm, each term in POS is maxterm → to turn canonical, for product multiple by 1 and add missing variable as (A+A’) and for sum add 0 and add in missing variable as AA’ **(UP) NAND; Consensus Theorem:** Given a pair of terms where one the variables appears as a true and complement, the consensus is formed by ANDing (or ORing) the remaining terms together (AB+A’C+**BC** and (A+B)(A’+C)(**B+C**)): WORKS FOR GROUPS OF VARIABLES; USEFUL: A + A’B = A + B ⇒ you can show that adding the consensus term is equivalent to the original through boolean algebra, venn diagrams, or truth tables

Show AB + A’C + BC == AB + A’C
AB + A’C + BC
AB + A’ C + 1BC
AB + A’C + (A+A’)BC
AB + A’C + ABC + A’BC

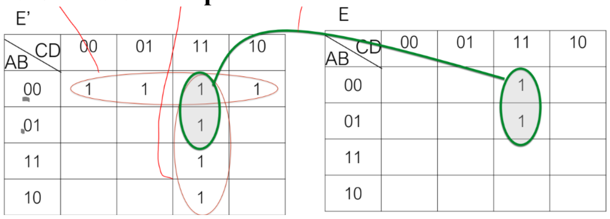


Bubble Pushing: pushing bubbles backward (from the output) or forward (from the inputs) changes the body of the gate from AND to OR or vice versa; **Shannon’s:** f(x, …) = x*f(1, …) + x’*f(0,...) = (x+f(0, …))(x’+f(1,...)) → can apply recursively to more than one term (f(x, y, …) = x*y*f(1, 1, …) + x’*y*f(0,1,...) + x*y’*f(1,0,...) + x*y*f(1,1,...)) ← **NOR; Incompletely Specified Function:** Output does not matter in certain input cases⇒ $\Sigma m(1, 3, 4, 7) + \Sigma d(2, 5)$ means minterms 1, 3, 4, 7 are 1 and 2, 5 are X and rest are 0 while $\Pi M(0, 6)\ \Pi d(2, 5)$ means maxterms 0, 6 are 0 and 2, 5 are X and the rest are 1→if no unspecified, then sum of all minterms is logically equivalent to prod of all maxterms; **Minimal Two Level Logic:** For all expressions with minimum # of implicants/implicates, find the solution with the minimum # of literals; **Karnaugh Map⇒ Covers**: a term covers a minterm if that term evaluates to “1” when the minterms it covers evaluate to 1 (ab covers abc+abc’); **prime implicant/implicate** - the largest implicant/implicate that covers a region of 1’s; **essential prime implicant/implicate** - a prime implicant/implicate that is only prime implicant that includes at least one of its 1’s; **Universal Gates:** any row on a truth table can be expressed with two logical operations, and/or with not and the entire truth table can be expressed with three logic operations: and, not or; NOT(A) = NAND(A, A) = NOR(A, A), AND(A, B) = NAND(NAND(A, B), NAND(A, B)) = NOR(NOR(A, A), NOR(B, B)), OR(A, B) = NAND(NAND(A, A), NAND(B, B)) = NOR(NOR(A, B), NOR(A, B)); **Bubble Pushing:** 1) to convert to NAND network: replace all AND gates with NAND (bubble output) and replace all OR gates with NAND (bubble inputs) and if bubble output drives bubble input, DONE else add inverter 2) to convert to NOR network: replace all OR gates with NOR (bubble output) and replace all AND gates with NOR (bubble input) and if bubble output drives bubble input, DONE else add inverter;

Numbering K-map Example:

ab\cd	00	01	11	10
00	m0	m1	**m3**	m2
01	m4	m5	**m7**	m6
11	**m12**	**m13**	**m15**	**m14**
10	m8	m9	**m11**	m10

5 Variable K-map:



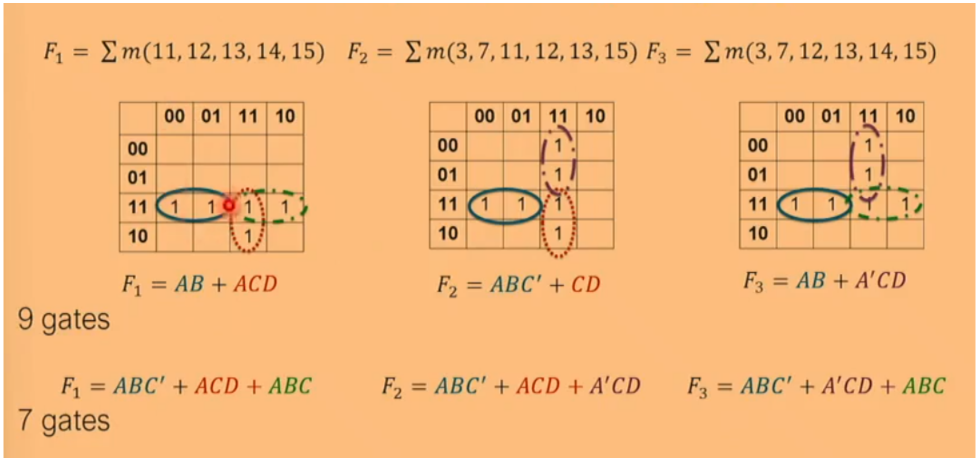
E’

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

E

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

Multi-Output:



$F_1 = \sum m(11, 12, 13, 14, 15)$ $F_2 = \sum m(3, 7, 11, 12, 13, 15)$ $F_3 = \sum m(3, 7, 12, 13, 14, 15)$

	00	01	11	10
00				
01				
11	1	1	1	1
10			1	

$F_1 = AB + ACD$

	00	01	11	10
00			1	
01			1	
11	1	1	1	
10			1	

$F_2 = ABC' + CD$

	00	01	11	10
00			1	
01			1	
11	1	1	1	1
10				

$F_3 = AB + A'CD$

9 gates

$F_1 = ABC' + ACD + ABC$ $F_2 = ABC' + ACD + A'CD$ $F_3 = ABC' + A'CD + ABC$

7 gates

Quine-McCluskey:

- cd’ covers 2, 6, 10, 14

$$f(a,b,c,d) = \sum m(0,1,2,5,6,7,8,9,10,14)$$

0	0000	✓
1	0001	✓
2	0010	✓
8	1000	✓
5	0101	✓
6	0110	✓
9	1001	✓
10	1010	✓
7	0111	✓
14	1110	✓

0,1	000-	✓
0,2	00-0	✓
0,8	-000	✓
1,5	0-01	
1,9	-001	✓
2,6	0-10	✓
2,10	-010	✓
8,9	100-	✓
8,10	10-0	✓
5,7	01-1	
6,7	011-	
6,14	-110	✓
10,14	1-10	✓

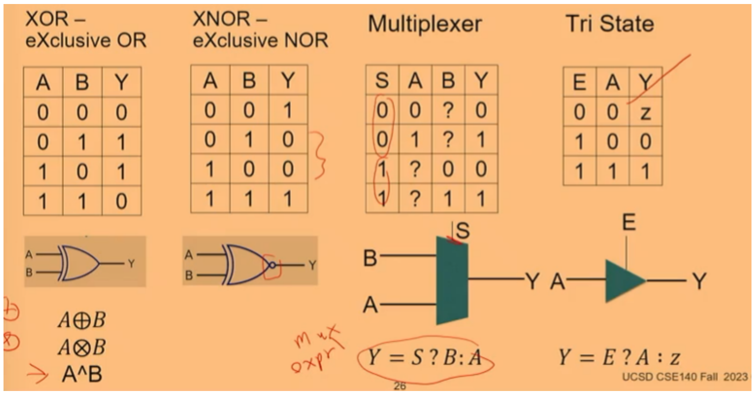
0,1,8,9	-00-	
0,2,8,10	-0-0	
0,8,1,9	-00-	X
0,8,2,10	-0-0	X
2,6,10,14	--10	
2,10,6,14	--10	X

		0	1	2	5	6	7	8	9	10	14
0,1,8,9	b’c’	x	x					x	x		
0,2,8,10	b’d’	x		x				x		x	
2,6,10,14	cd’			x		x				x	x
1,5	a’c’d		x		x						
5,7	a’bd				x		x				
6,7	a’bc					x	x				

a'c'd (5) || a'bd (5,7)

a'bc (7)

If given POS and want to find minimal two level, apply negate on whole thing, do Quine on SOP, then negate that result



XOR – eXclusive OR

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

XNOR – eXclusive NOR

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

Multiplexer

S	A	B	Y
0	0	?	0
0	1	?	1
1	?	0	0
1	?	1	1

Tri State

E	A	Y
0	0	z
1	0	0
1	1	1

$A \oplus B$

$A \otimes B$

→ A^B

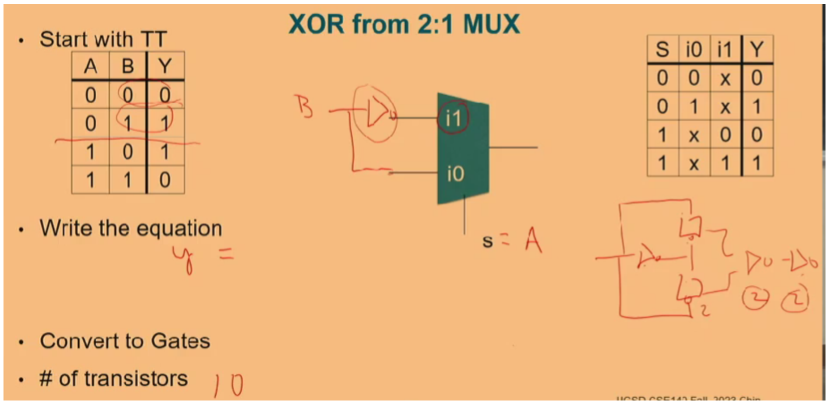
mux expr

$Y = S\,?\,B : A$

$Y = E\,?\,A : z$

UCSD CSE140 Fall 2023

26



XOR from 2:1 MUX

- Start with TT

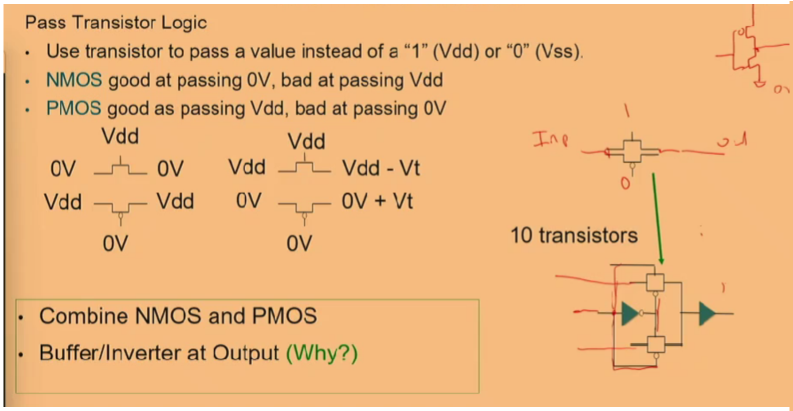
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

- Write the equation

y =

- Convert to Gates
- # of transistors 10

S	i0	i1	Y
0	0	x	0
0	1	x	1
1	x	0	0
1	x	1	1



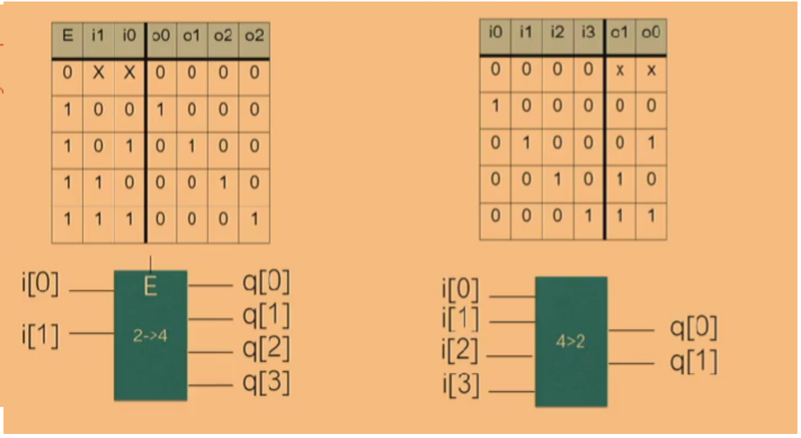
Pass Transistor Logic

- Use transistor to pass a value instead of a “1” (Vdd) or “0” (Vss).
- NMOS good at passing 0V, bad at passing Vdd
- PMOS good as passing Vdd, bad at passing 0V

10 transistors

- Combine NMOS and PMOS
- Buffer/Inverter at Output (Why?)

XOR can be used for parity check and detect odd number of bit errors

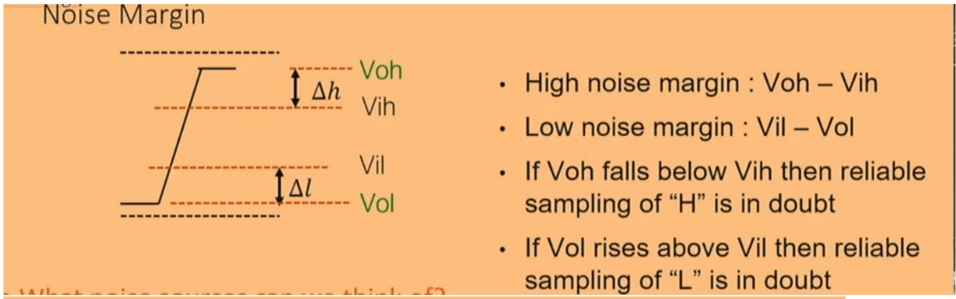


E	i1	i0	o0	o1	o2	o2
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

i0	i1	i2	i3	o1	o0
0	0	0	0	x	x
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

DECODER

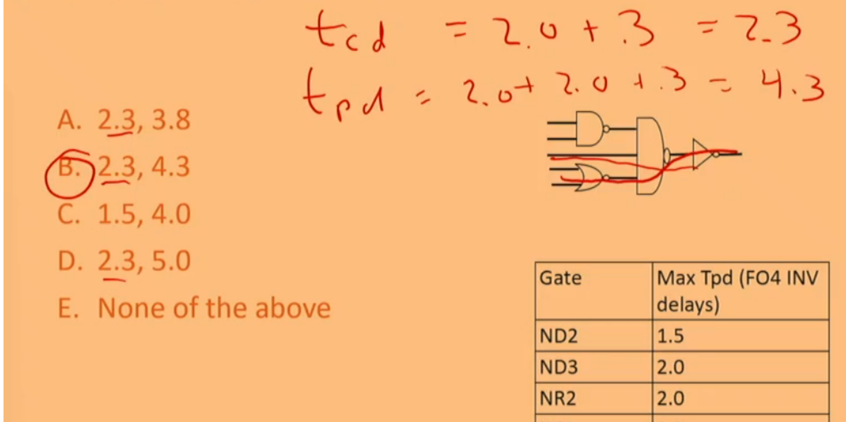
ENCODER



Noise Margin

- High noise margin : Voh – Vih
- Low noise margin : Vil – Vol
- If Voh falls below Vih then reliable sampling of “H” is in doubt
- If Vol rises above Vil then reliable sampling of “L” is in doubt

What noise sources can we think of?



$t_{cd} = 2.0 + .3 = 2.3$

$t_{pd} = 2.0 + 2.0 + .3 = 4.3$

A. 2.3, 3.8

[✓ B. 2.3, 4.3]

C. 1.5, 4.0

D. 2.3, 5.0

E. None of the above

Gate	Max Tpd (FO4 INV delays)
ND2	1.5
ND3	2.0
NR2	2.0