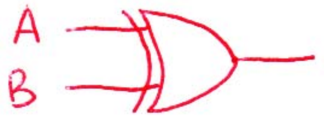
P3360 / AEP 3630

Lecture 25

XOR gate

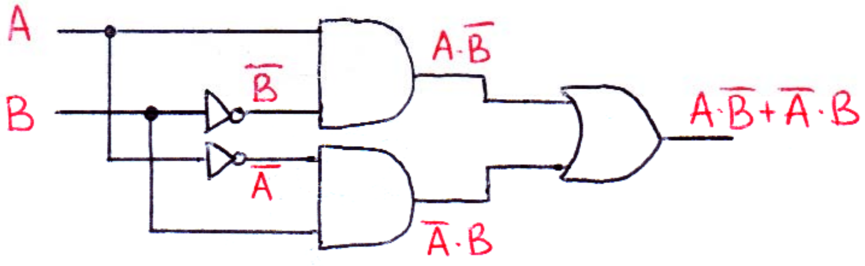
not universal but quite useful ; used for addition, parity checking



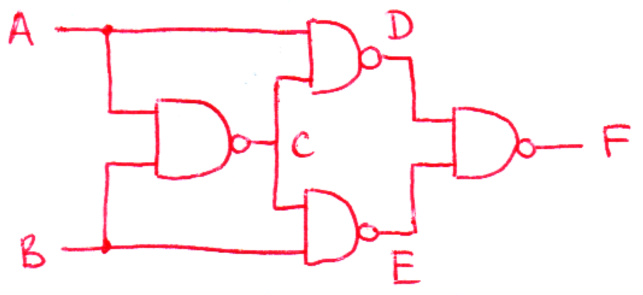
$X = A \oplus B = \bar{A} \cdot B + A \cdot \bar{B}$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Q: how is it made ?



uses 5 gates (inverter costs same as AND, OR)



uses only 4 NAND gates
(20% cost savings)

① de Morgan
$\overline{A \cdot B} = \bar{A} + \bar{B}$
$\overline{A + B} = \bar{A} \cdot \bar{B}$

② $A + B \cdot C = (A+B)(A+C)$
for any A, B, C

$C = \overline{A \cdot B}$

$D = \overline{A \cdot C} = \overline{A \cdot \overline{A \cdot B}} = \bar{A} + \overline{\overline{A \cdot B}} = \bar{A} + A \cdot B = \bar{A} + B$

$E = \overline{B \cdot C} = \overline{B \cdot \overline{A \cdot B}} = \bar{B} + \overline{\overline{A \cdot B}} = \bar{B} + A \cdot B = \bar{B} + A$

Q: prove ?

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

$= \overline{\bar{A} \cdot \bar{B} + \underbrace{B\bar{B}}_{0} + \underbrace{A\bar{A}}_{0} + BA} = \underbrace{\overline{\bar{A} \cdot \bar{B}}}_{A+B} \cdot \underbrace{\overline{B \cdot A}}_{\bar{A}+\bar{B}} = \underbrace{A\bar{A}}_{0} + B\bar{A} + \underbrace{B\bar{B}}_{0} + A\bar{B} = A\bar{B} + B\bar{A} = A \oplus B$

Design procedure for combinational logic circuits

1) assign symbols (bits) to inputs / outputs

2) obtain boolean fcn relating inputs/outputs either

   a) by transforming a verbal description

   b) from input/output truth table

3) [optional] simplify to minimize # of gates

4) draw the schematic using (available) gates

E.g. stay Home if Tired but no Prelim or Sick

$$H = T\cdot\overline{P} + S$$

Q: what if just handed a truth table

T	P	S	H	minterms	maxterms
0	0	0	0		$T+P+S$
0	0	1	1	$\overline{T}\cdot\overline{P}\cdot S$	
0	1	0	0		$T+\overline{P}+S$
0	1	1	1	$\overline{T}\cdot P\cdot S$	
1	0	0	1	$T\cdot\overline{P}\cdot\overline{S}$	
1	0	1	1	$\overline{T}\cdot P\cdot\overline{S}$	
1	1	0	0		$\overline{T}+\overline{P}+S$
1	1	1	1	$T\cdot P\cdot S$	

minterms – "1"'s in truth table for a unique comb. ("$\overset{\text{AND}}{\uparrow}$0" otherw.)

maxterms – "0"'s for a unique comb. ("$\underset{\text{OR}}{\downarrow}$1" otherwise)

output = "1" if any of minterms is "1" $= \sum_{OR}$ minterms

= "sum of products"

$$H = \overline{T}\cdot\overline{P}\cdot S + \overline{T}\cdot P\cdot S + T\cdot\overline{P}\cdot\overline{S} + \overline{T}\cdot P\cdot\overline{S} + T\cdot P\cdot S$$

output = "0" if any of maxterms is "0" $= \prod_{AND}$ maxtems

= "product of sums"

$$H = (T+P+S)(T+\overline{P}+S)(\overline{T}+\overline{P}+S)$$

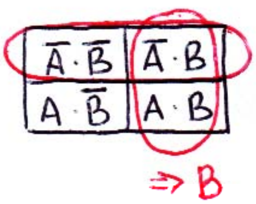
Simplifying boolean expressions with Karnaugh maps

Consider 2-input fcn

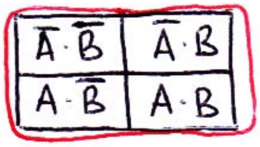
A	B	minterm
0	0	$\bar{A} \cdot \bar{B}$
0	1	$\bar{A} \cdot B$
1	0	$A \cdot \bar{B}$
1	1	$A \cdot B$

		B	
		0	1
A	0	$\bar{A} \cdot \bar{B}$	$\bar{A} \cdot B$
	1	$A \cdot \bar{B}$	$A \cdot B$

rearranged truth table a.k.a. Karnaugh map

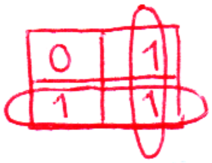


$\Rightarrow \bar{A}$



$\Rightarrow 1$

e.g. Karnaugh map for $A \cdot B + A \cdot \bar{B} + \bar{A} \cdot B$



$\Rightarrow A + B$

3-input fcn $X = f(A,B,C)$

		AB			
		00	01	11	10
C	0				
	1				

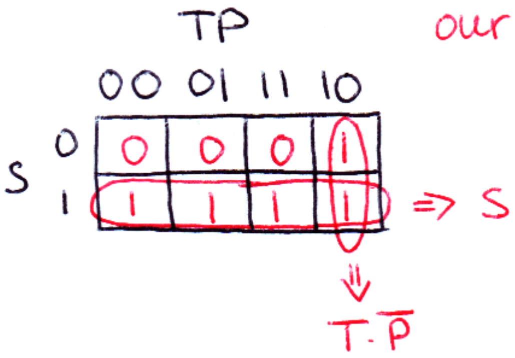
gray code

4-input fcn $X = f(A,B,C,D)$

		AB			
		00	01	11	10
CD	00				
	01				
	11				
	10				

- Karnaugh maps can be used with up to 6 variable
- use computer if more inputs

our example

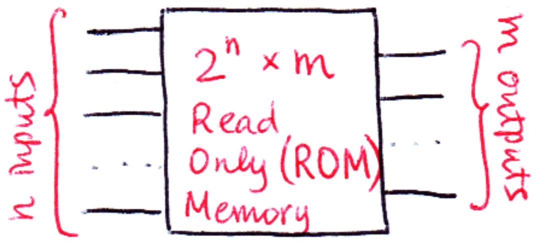


S \ TP	00	01	11	10
0	0	0	0	1
1	1	1	1	1

, ⇒ output = $S + T\cdot\bar{P}$

Programmable logic devices (PLDs)

- used to implement complicated logic (both combinational & sequential)



- can generate arbitrary truth table of a given size by generating the needed minterms

- usually done by breaking the links, can be reversable or irreversable; stored in "memory"
- e.g. EPROM: erasable programmable ROM: uses charge on tiny caps to store info about the logic; non-volatile (preserved between power cycles); erasable by UV light

- some PLD types: PAL - programmable array logic one-time prog. (OTP), few 100 gates

  CPLD - complex PLD; ≲10's of thousand gates

  FPGA - field programmable gate arrays ≲ millions of gates; ext. ROM more complex architecture than "sum of products"

see LTspice Exp. 8.2