Lecture 13
Magnitude Comparator

Magnitude Comparator

- It is a combinational circuit that compares two numbers and determines their relative magnitude
- The output of comparator is usually 3 binary variables indicating: A>B

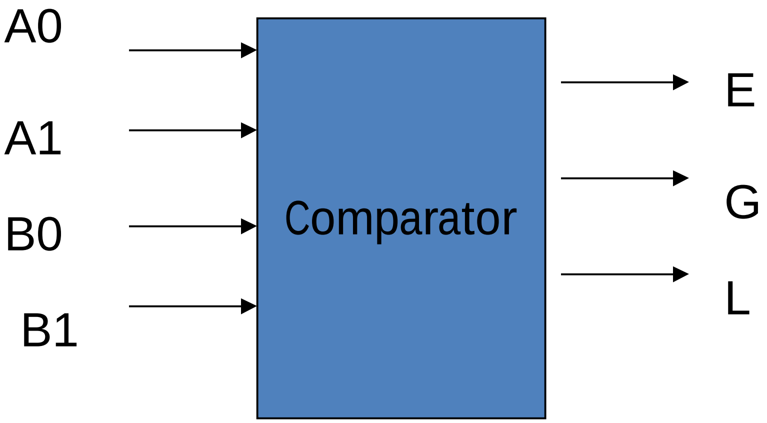
  A=B

  A<B
- For example to design a comparator for 2 bit binary numbers A (A1A0) and B (B1B0) we do the following steps:

Comparators

- **For a 2-bit comparator we have four inputs A1A0 and B1B0 and three output E (is 1 if two numbers are equal), G (is 1 when A > B) and L (is 1 when A < B) If we use truth table and KMAP the result is**
- **$E= A'1A'0B'1B'0 + A'1A0B'1B0 + A1A0B1B0 + A1A'0B1B'0$**

  or $E=((A0 \oplus B0) + (A1 \oplus B1))'$ (see next slide)
- **$G = A1B'1 + A0B'1B'0 + A1A0B'0$**
- **$L= A'1B1 + A'1A'0B0 + A'0B1B0$**



Truth Table

A0	A1	B0	B1	E	L	G
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	1	0	0

Magnitude Comparator

- From the truth table:

E = (0,5,10,15)

= A1’A0’B1’B0’ + A1’A0B1’B0 + A1A0’B1B0’ +A1A0B1B0

Magnitude Comparator

- A>B means

A1	B1	Y1
0	0	0
0	1	0
1	0	1
1	1	0

if A1=B1 (X1=1) then A0 should be 1 and B0 should be 0

A0	B0	Y0
0	0	1
0	1	0
1	0	0
1	1	0

For A> B: A1 > B1 or A1 =B1 and A0 > B0

It means Y= A1B’1 + X1A0B’0 should be 1 for A>B

Magnitude Comparator

- For $B>A$ $B1 > A1$

  or

  $A1=B1$ and $B0> A0$

  $z= A'1B1 + X1A'0B0$
- The procedure for binary numbers with more than 2 bits can also be found in the similar way. For example next slide shows the 4-bit magnitude comparator, in which

$(A= B) = x3x2x1x0$

$(A> B) = A3B'3 + x3A2B'2 + x3x2A1B'1+ x3x2x1A0B'0$

$(A< B) = A'3B3 + x3A'2B2 + x3x2A'1B1+ x3x2x1A'0B0$

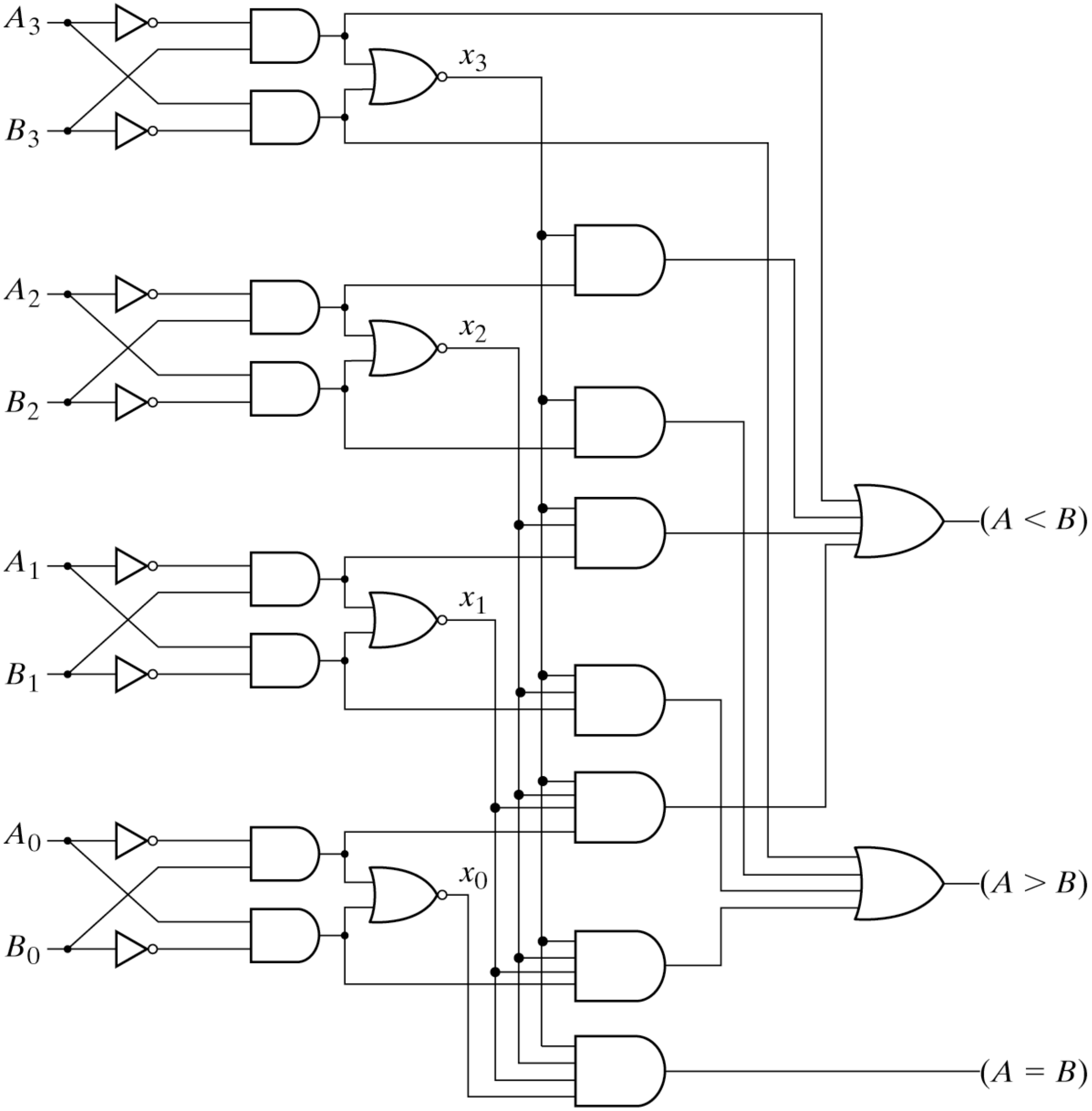
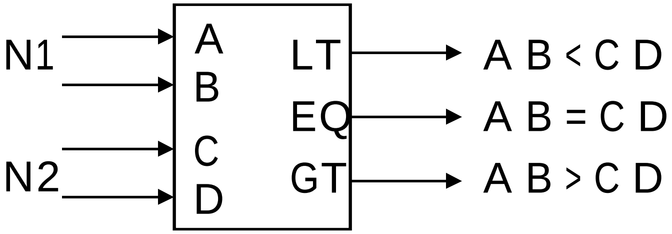


Fig. 4-17 4-Bit Magnitude Comparator

Design example: two-bit comparator

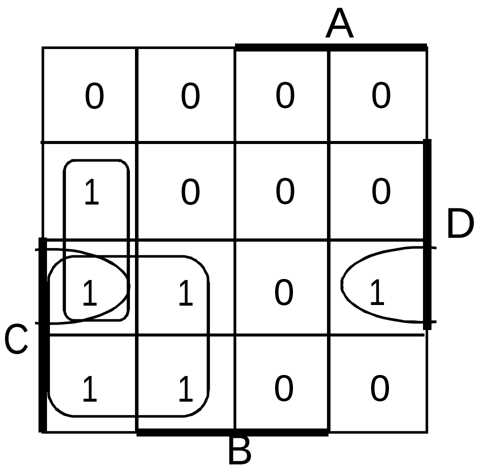


block diagram
and
truth table

A	B	C	D	LT	EQ	GT
0	0	0	0	0	1	0
		0	1	1	0	0
		1	0	1	0	0
		1	1	1	0	0
0	1	0	0	0	0	1
		0	1	0	1	0
		1	0	1	0	0
		1	1	1	0	0
1	0	0	0	0	0	1
		0	1	0	0	1
		1	0	0	1	0
		1	1	1	0	0
1	1	0	0	0	0	1
		0	1	0	0	1
		1	0	0	0	1
		1	1	0	1	0

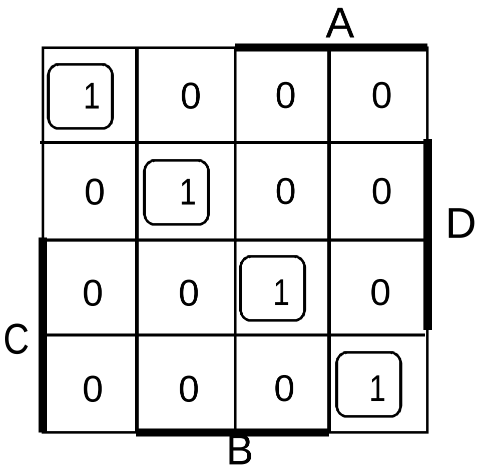
we'll need a 4-variable Karnaugh map
for each of the 3 output functions

Design example: two-bit comparator (cont’d)



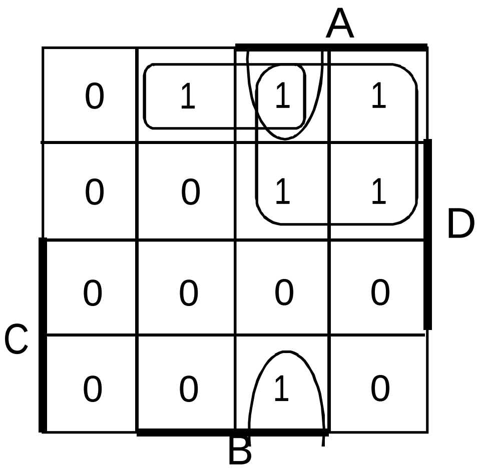
0	0	0	0
1	0	0	0
1	1	0	1
1	1	0	0

K-map for LT



1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

K-map for EQ



0	1	1	1
0	0	1	1
0	0	0	0
0	0	1	0

K-map for GT

LT = A' B' D + A' C + B' C D

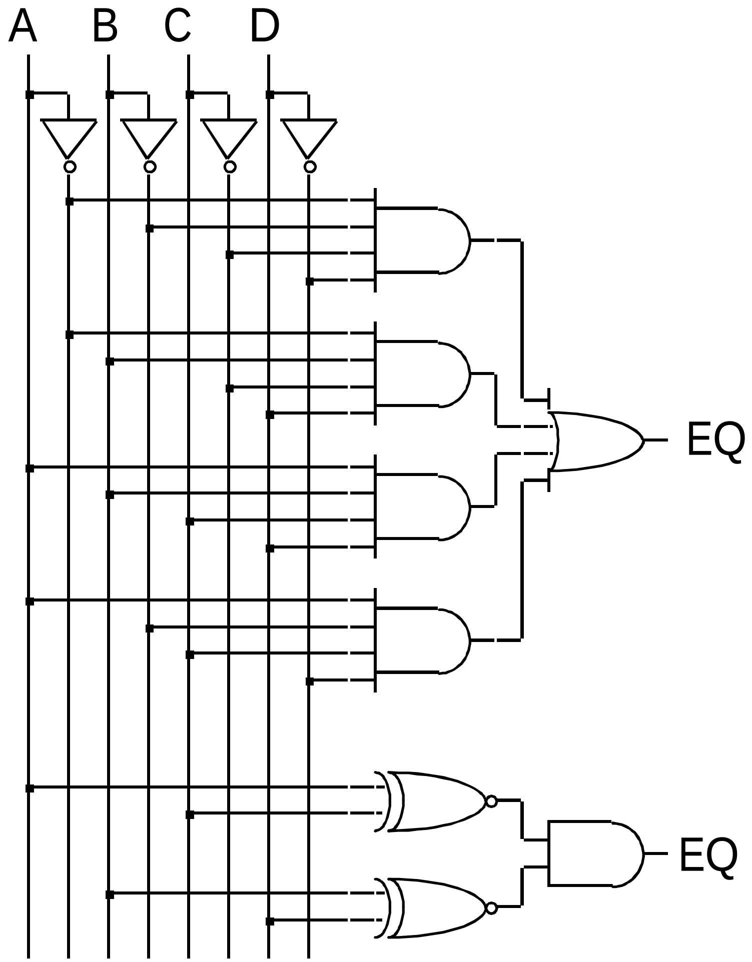
EQ = A'B'C'D' + A'BC'D + ABCD + AB'CD’ = (A xnor C) • (B xnor D)

GT = B C' D' + A C' + A B D'

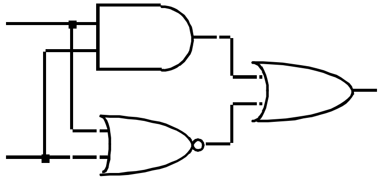
Canonical PofS vs minimal?

LT and GT are similar (flip A/C and B/D)

Design example: two-bit comparator (cont'd)



two alternative implementations of EQ with and without XOR



XNOR is implemented with at least 3 simple gates