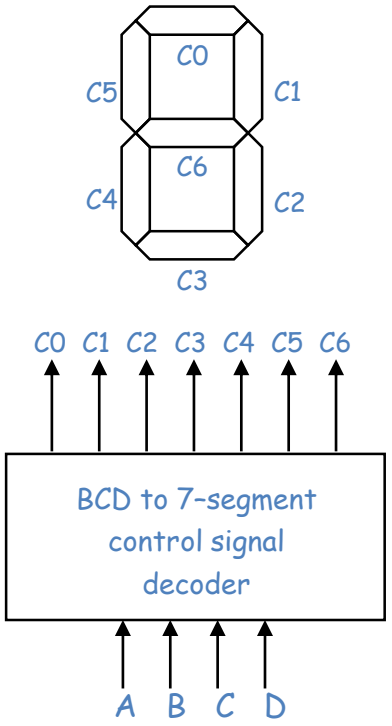
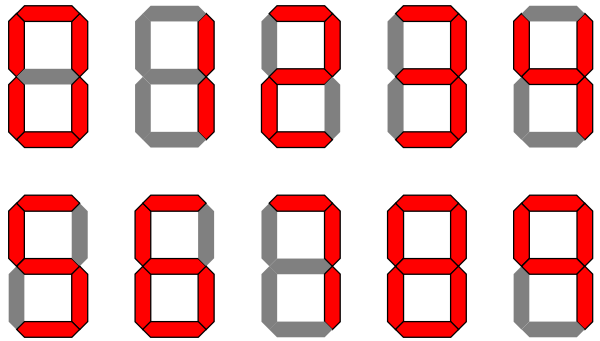
BCD to 7-segment display controller

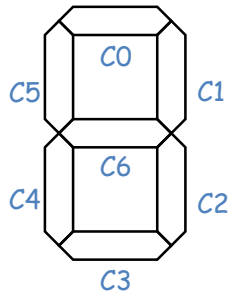
- ✓ Specification
  - Digital readouts on many digital products often use LED seven-segment displays.
  - Input is a 4 bit BCD digit (A, B, C, D)
  - Output is the control signals for the display (7 outputs C0 – C6)





Formulation

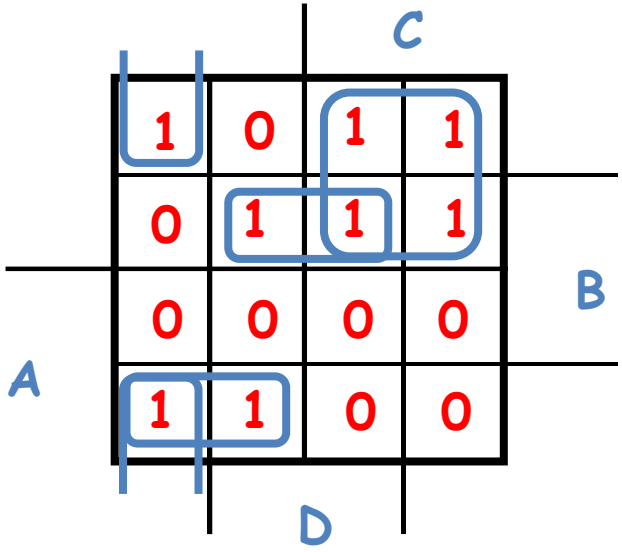
- ✓ Construct a truth table



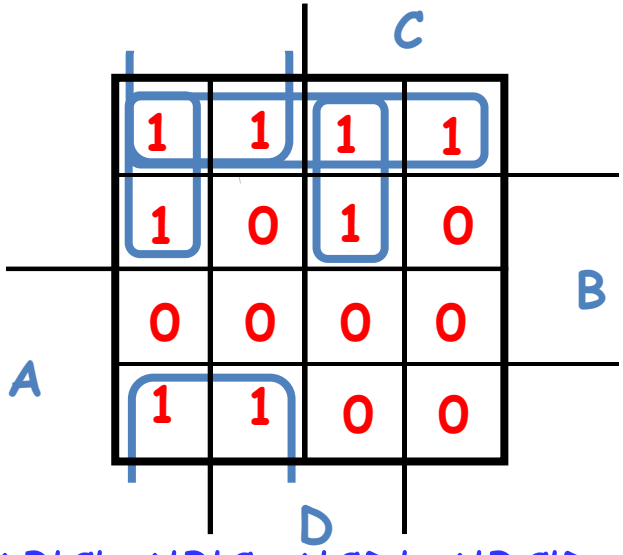
Decimal Digit	Input BCD	Seven-Segment Decoder Outputs C0 C1 C2 C3 C4 C5 C6
0	0 0 0 0	1 1 1 1 1 1 0
1	0 0 0 1	0 1 1 0 0 0 0
2	0 0 1 0	1 1 0 1 1 0 1
3	0 0 1 1	1 1 1 1 0 0 1
4	0 1 0 0	0 1 1 0 0 1 1
5	0 1 0 1	1 0 1 1 0 1 1
6	0 1 1 0	1 0 1 1 1 1 1
7	0 1 1 1	1 1 1 0 0 0 0
8	1 0 0 0	1 1 1 1 1 1 1
9	1 0 0 1	1 1 1 1 0 1 1
All other inputs		0 0 0 0 0 0 0

Optimization

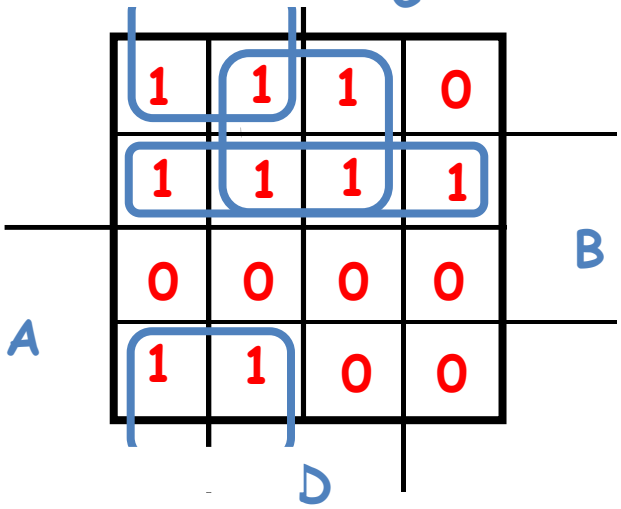
C0 = A'C+AB'C'+A'BD+B'C'D'



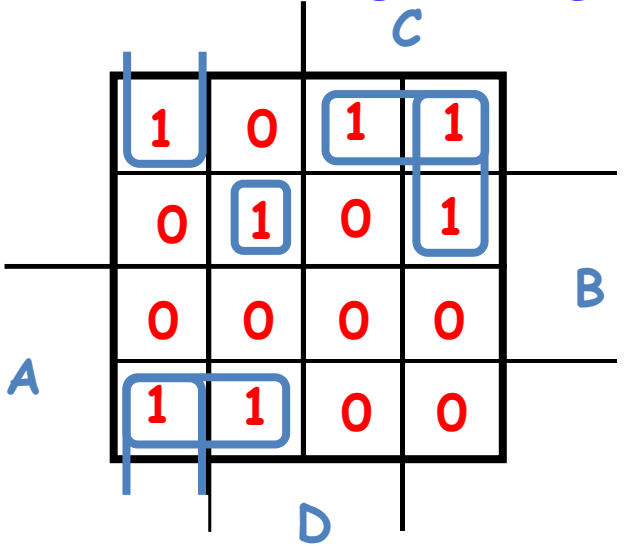
C1=A'B'+B'C'+A'C'D'+A'CD



C2=A'B+A'D+B'C'

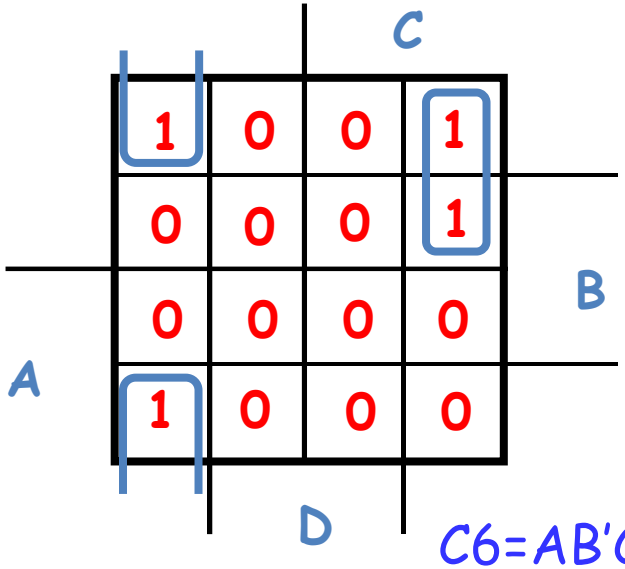


C3=AB'C'+A'B'C+A'CD'+A'BC'D+B'C'D'

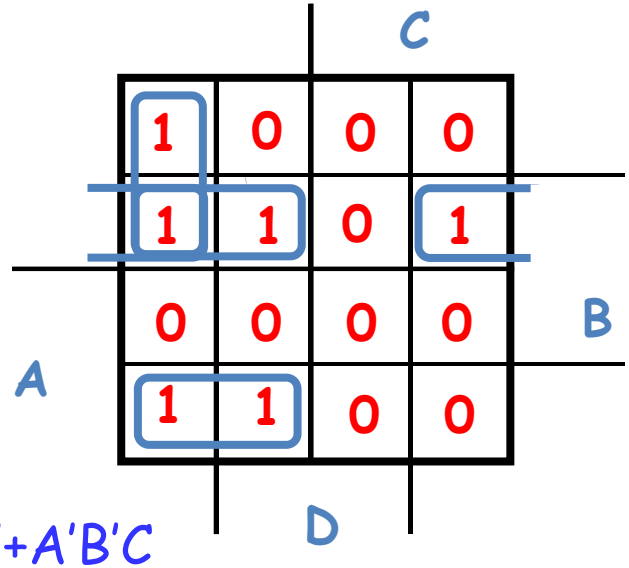


Optimization

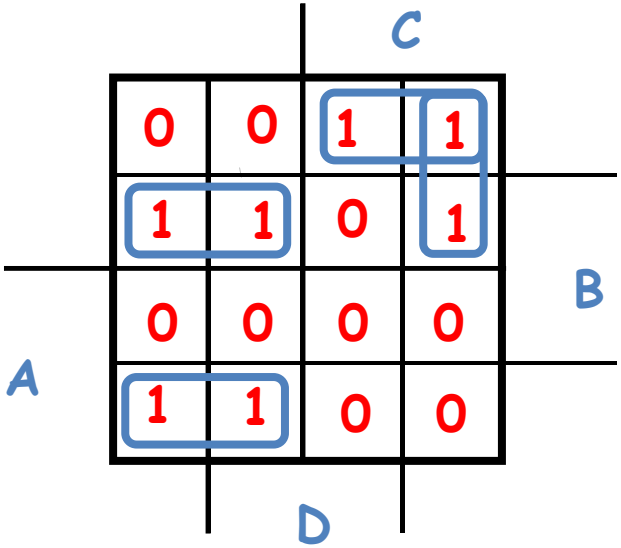
C4=A'CD'+B'C'D'



C5=AB'C'+A'BC'+A'BD'+A'C'D'



C6=AB'C'+A'BC'+A'CD'+A'B'C



Optimization

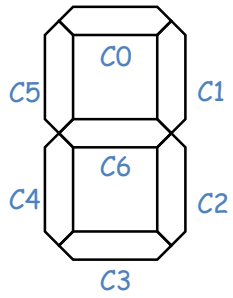
- ✓ Create a K-map for each output and get:

$C0 = A'C + AB'C' + A'BD + B'C'D'$
$C1 = A'B' + B'C' + A'C'D' + A'CD$
$C2 = A'B + A'D + B'C'$
$C3 = AB'C' + A'B'C + A'CD' + A'BC'D + B'C'D'$
$C4 = A'CD' + B'C'D'$
$C5 = AB'C' + A'BC' + A'BD' + A'C'D'$
$C6 = AB'C' + A'BC' + A'CD' + A'B'C$

- ✓ Direct implementation would require 26 AND gates and 7 OR gates.
- ✓ By sharing terms, can actualize with 11 less AND gates maintaining just 2 levels.

Formulation

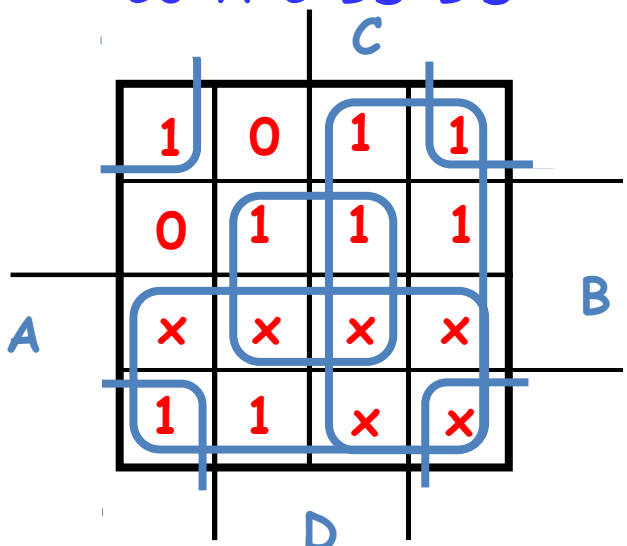
- ✓ Construct a truth table with don't care



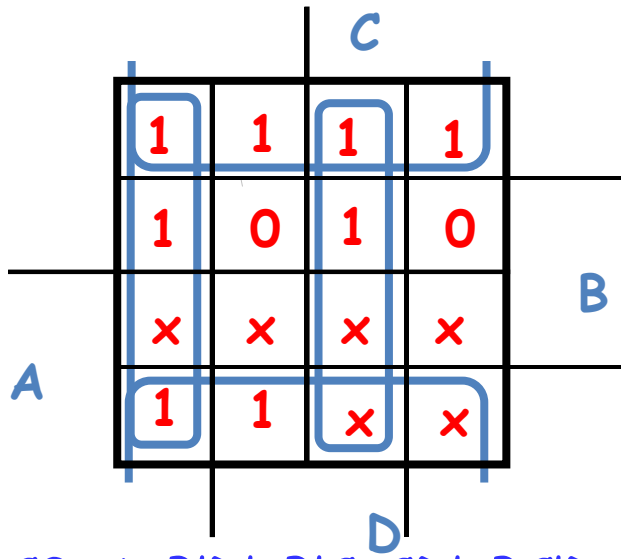
Decimal Digit	Input BCD	Seven-Segment Decoder Outputs C0 C1 C2 C3 C4 C5 C6
0	0 0 0 0	1 1 1 1 1 1 0
1	0 0 0 1	0 1 1 0 0 0 0
2	0 0 1 0	1 1 0 1 1 0 1
3	0 0 1 1	1 1 1 1 0 0 1
4	0 1 0 0	0 1 1 0 0 1 1
5	0 1 0 1	1 0 1 1 0 1 1
6	0 1 1 0	1 0 1 1 1 1 1
7	0 1 1 1	1 1 1 0 0 0 0
8	1 0 0 0	1 1 1 1 1 1 1
9	1 0 0 1	1 1 1 1 0 1 1
All other inputs		X X X X X X X

Optimization

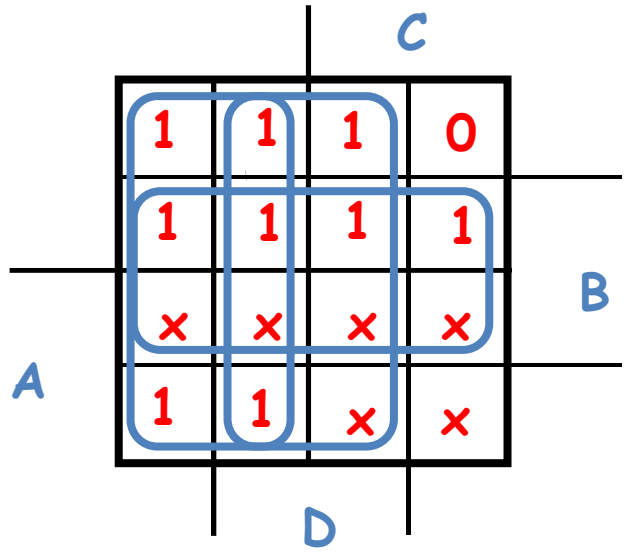
C0=A+C+BD+B'D'



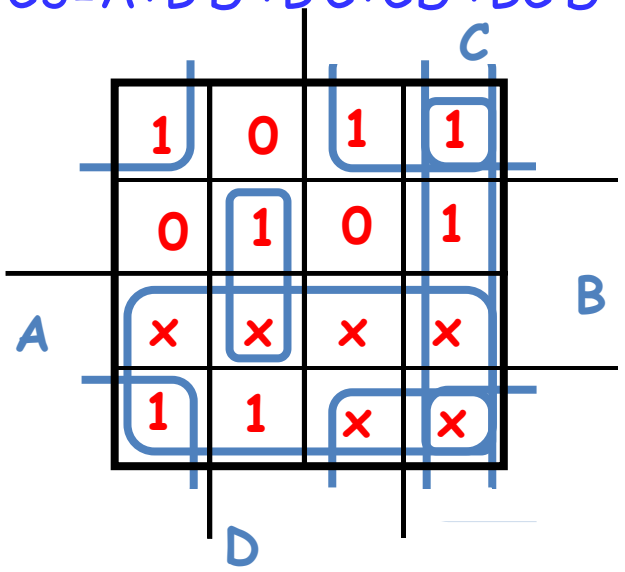
C1=B'+C'D'+CD



C2=B+C'+D

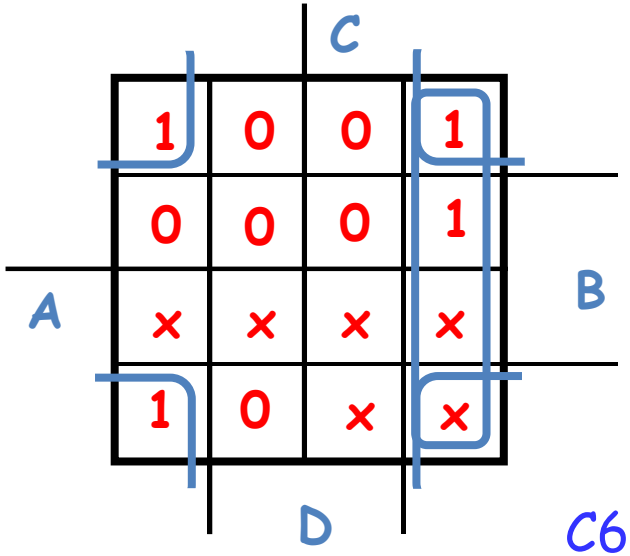


C3=A+B'D'+B'C+CD'+BC'D

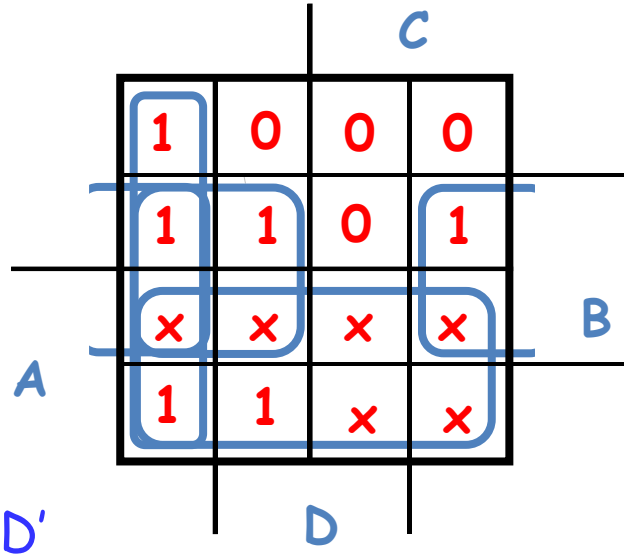


Optimization

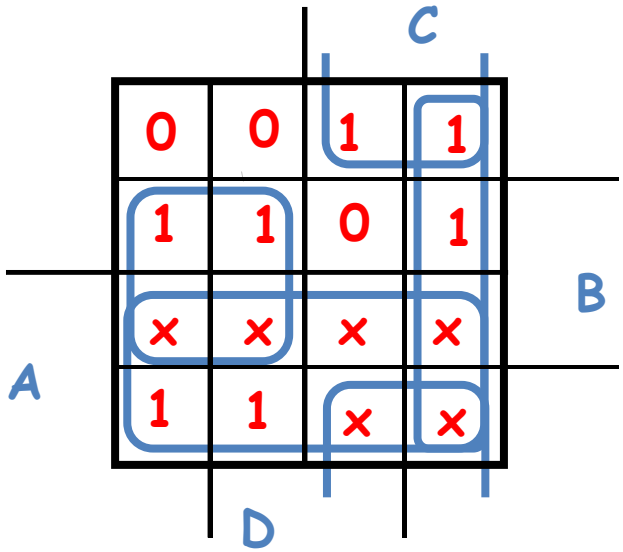
C4=B'D'+CD'



C5=A+BC'+BD'+C'D'



C6=A+BC'+B'C+CD'



Optimization Exploiting DC

- ✓ Create a K-map for each output and get
  - C0 = A+C+BD+B'D'
  - C1 = B'+C'D'+CD
  - C2 = B+C'+D
  - C3 = A+B'C+B'D'+CD'+BC'D
  - C4 = B'D'+CD'
  - C5 = A+C'D'+BD'+BC'
  - C6 = A+B'C+BC'+CD'
- ✓ Direct implementation would require 16 AND gates and 7 OR gates.
- ✓ By sharing terms, can actualize with 7 less AND gates maintaining just 2 levels.