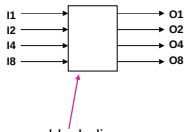
Lecture 8

- K-map minimization examples
- POS minimization with K-map
- Design example
- "Switching network" logic blocks (multiplexers/demultiplexers)

Example: BCD increment-by-1



block diagram

truth table

I8	I4	I2	I1	O8	O4	O2	O1
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	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Need a 4-variable Karnaugh map for each of the 4 output functions

BCD increment-by-1: K-maps

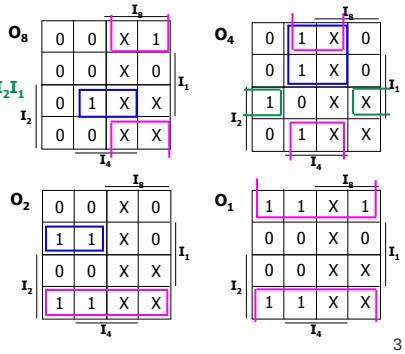
$O_8 = I_4I_2I_1 + I_8I_1'$

$O_4 = I_4I_2' + I_4I_1' + I_4'I_2I_1$

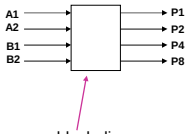
$O_2 = I_8'I_2'I_1 + I_2I_1'$

$O_1 = I_1'$

We greatly simplify the logic by using the don't cares



Example: Two-bit multiplier

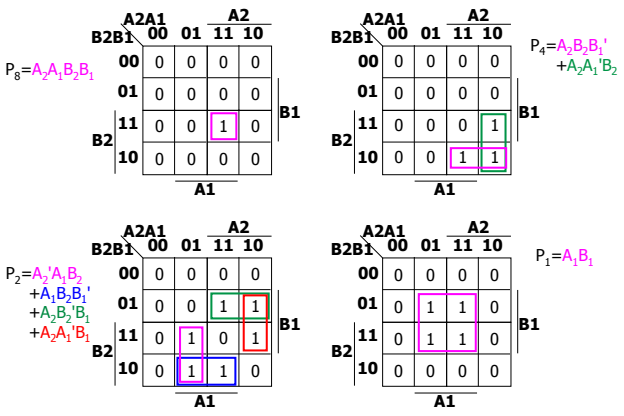


block diagram

truth table

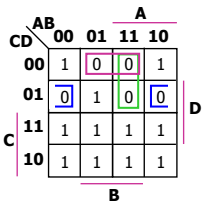
A2	A1	B2	B1	P8	P4	P2	P1
0	0	0	0	0	0	0	0
		0	1	0	0	0	0
		1	0	0	0	0	0
		1	1	0	0	0	0
0	1	0	0	0	0	0	0
		0	1	0	0	0	1
		1	0	0	0	1	0
		1	1	0	0	1	1
1	0	0	0	0	0	0	0
		0	1	0	0	1	0
		1	0	0	1	0	0
		1	1	0	1	1	0
1	1	0	0	0	0	0	0
		0	1	0	0	1	1
		1	0	0	1	1	0
		1	1	1	0	0	1

Need a 4-variable Karnaugh map for each of the 4 output functions



POS minimization with K-maps

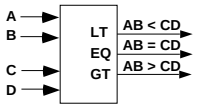
- Encircle the zeros in the map
- Interpret indices complementary to SOP form



F = (B'+C+D)(B+C+D')(A'+B'+C)

Same idea as with truth tables

Design example: Comparator



block diagram

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

Need a 4-variable Karnaugh map for each of the 3 output functions

Comparator: K-maps

K-map for LT

CD \ AB	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

K-map for EQ

CD \ AB	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

K-map for GT

CD \ AB	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

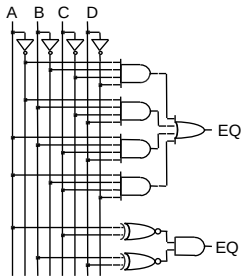
LT = A'B'D+A'C+B'CD

GT = BC'D'+AC'+ABD'

EQ = A'B'C'D'+A'BC'D+ABCD+AB'CD'

= (A xnor C)•(B xnor D)

Comparator: Implementing EQ

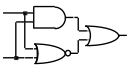


Option 1:
EQ = A'B'C'D'+A'BC'D+ABCD+AB'CD'

5 gates but they require lots of inputs

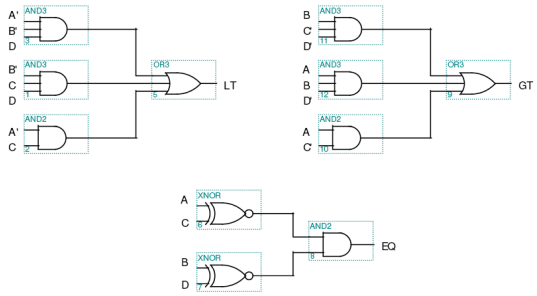
Option 2
EQ = (A xnor C) •(B xnor D)

XNOR is constructed from 3 simple gates



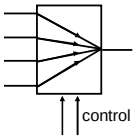
7 gates but they all have 2 inputs each

Comparator: Circuit schematics

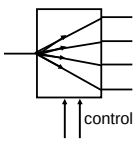


Switching networks logic blocks

- Multiplexer (MUX)
  - Routes one of many inputs to a single output
  - Also called a *selector*



- Demultiplexer (DEMUX)
  - Routes a single input to one of many outputs
  - Also called a *decoder*



Multiplexers

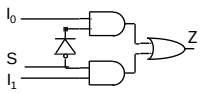
- Basic concept
  - 2^n data inputs; n control inputs ("selects"); 1 output
  - Connects one of 2^n inputs to the output
  - "Selects" decide which input connects to output

Multiplexers: Truth tables

- Two alternative truth-tables: Functional and Logical

Example: A 2:1 Mux

$Z = SIn_1 + S'In_0$



Functional truth table

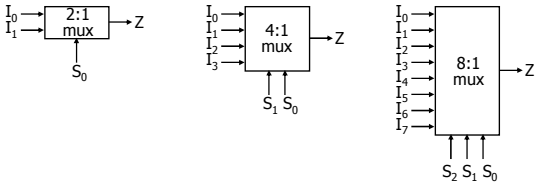
S	Z
0	In_0
1	In_1

Logical truth table

In_1	In_0	S	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

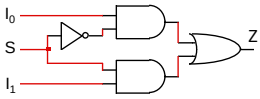
Multiplexers

- 2:1 mux: $Z = S_0'In_0 + S_0In_1$
- 4:1 mux: $Z = S_1'S_0'In_0 + S_1'S_0In_1 + S_1S_0'In_2 + S_1S_0In_3$
- 8:1 mux: $Z = S_2'S_1'S_0'In_0 + S_2'S_1'S_0In_1 + \ldots$

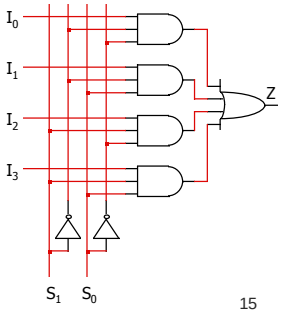


Multiplexers: Implementation

2:1 mux



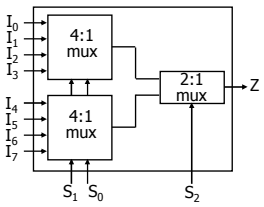
4:1 mux



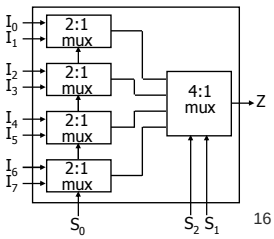
Cascading multiplexers

- Can form large multiplexers from smaller ones (many implementation options)

8:1 mux



8:1 mux

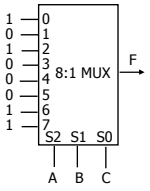


Multiplexer as logic block

- A 2^n:1 mux can implement any function of n variables as a lookup table

F(A,B,C) = m0 + m2 + m6 + m7
= A'B'C' + A'BC' + ABC' + ABC

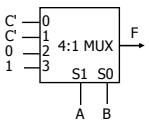
A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



Multiplexer as logic block

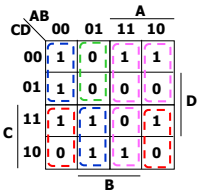
- Can also use a 2^{n-1}:1 mux to implement a function of n variables
  - (n-1) mux control variables $S_0 - S_{n-2}$
  - One data variable S_{n-1}
  - Four possible values for each data input: 0, 1, S_{n-1}, S_{n-1}'

A	B	C	F	
0	0	0	1	C'
0	0	1	0	
0	1	0	1	C'
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	



Multiplexer as logic block

- F(A,B,C,D) implemented using an 8:1 mux



Choose A,B,C as control variables

Choose D as a data variable