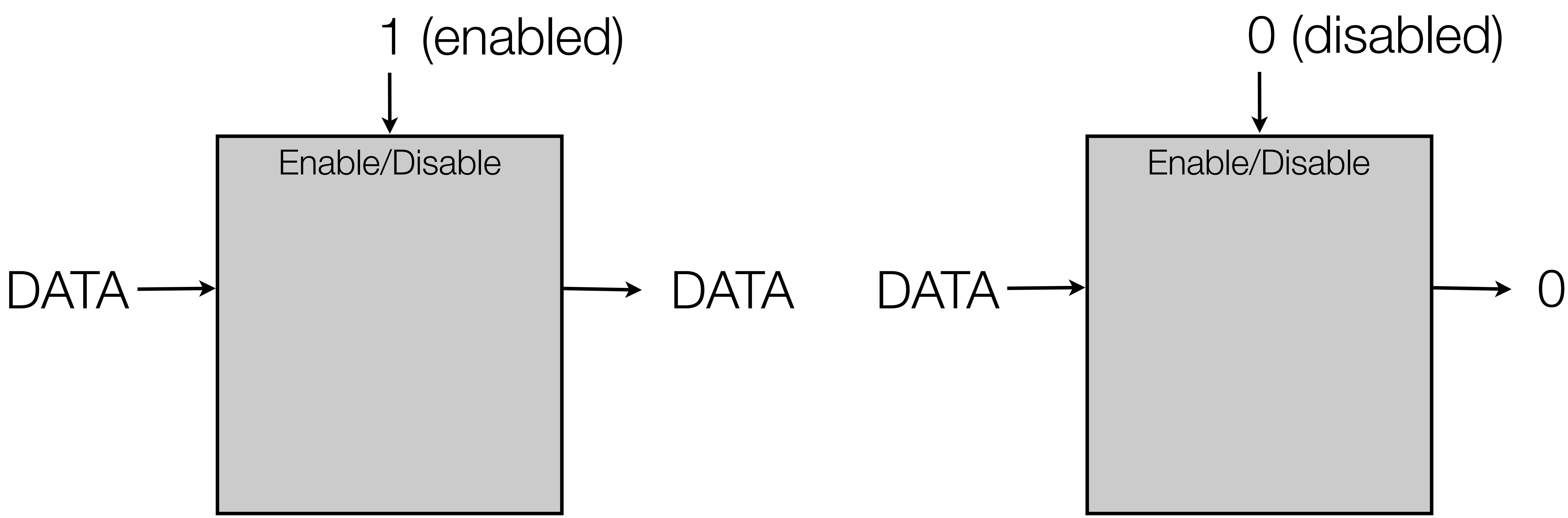
Combinational circuits

- Combinational circuits are stateless
- The outputs are functions only of the inputs

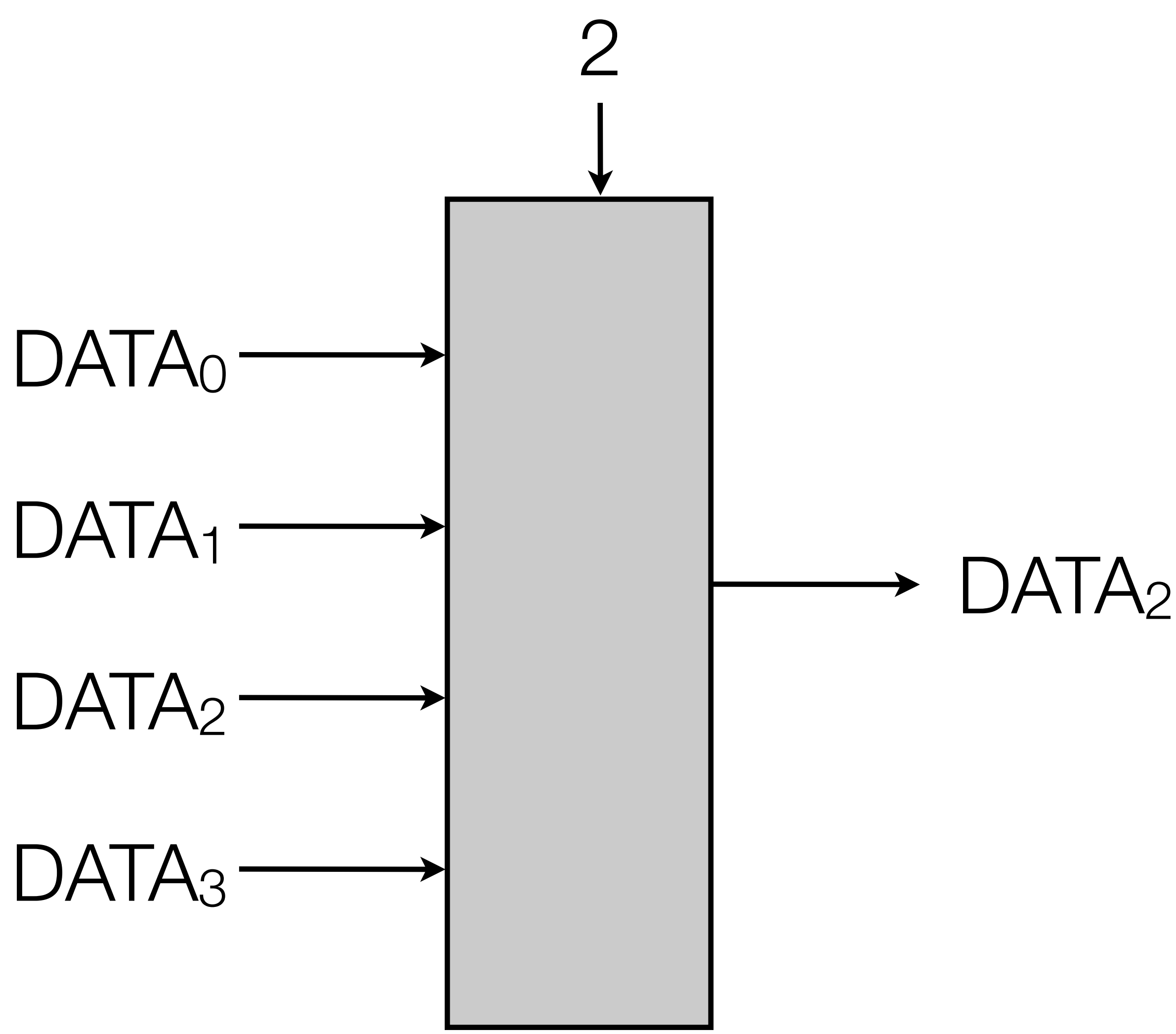
Enabler Circuit (High-level view)

- Enabler circuit has 2 inputs
  - data (can be several bits, but 1 bit examples for now)
  - enable/disable (i.e., on/off)
- Enable circuit “on”: output = data, Enable circuit “off”: output is “zeroed” (e.g., output signal is all 0’s)



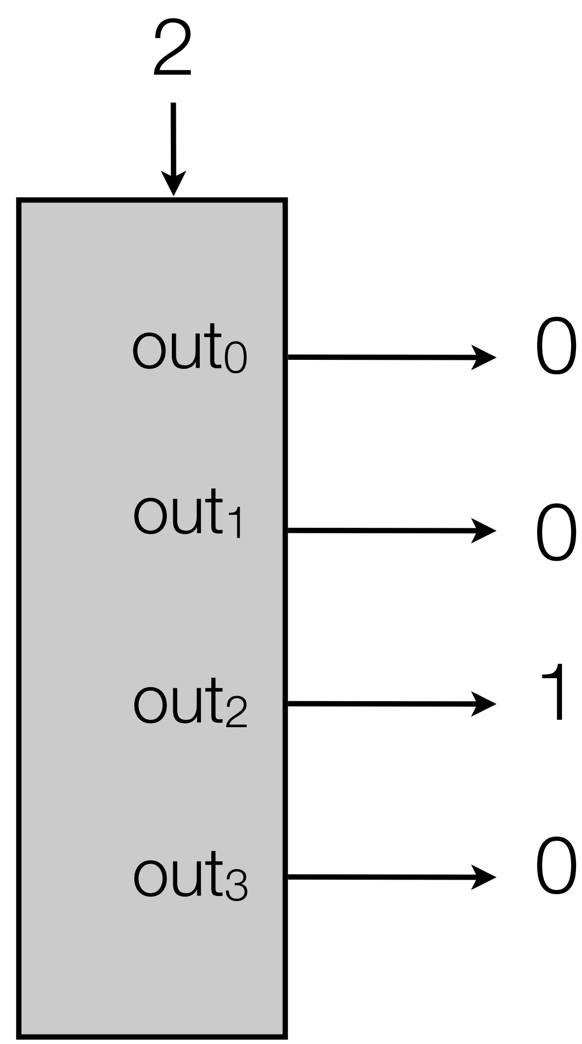
MUX Circuit (High-level)

- k Data values enter as input
- Selector chooses which one comes out



Decoder Circuit (high-level view)

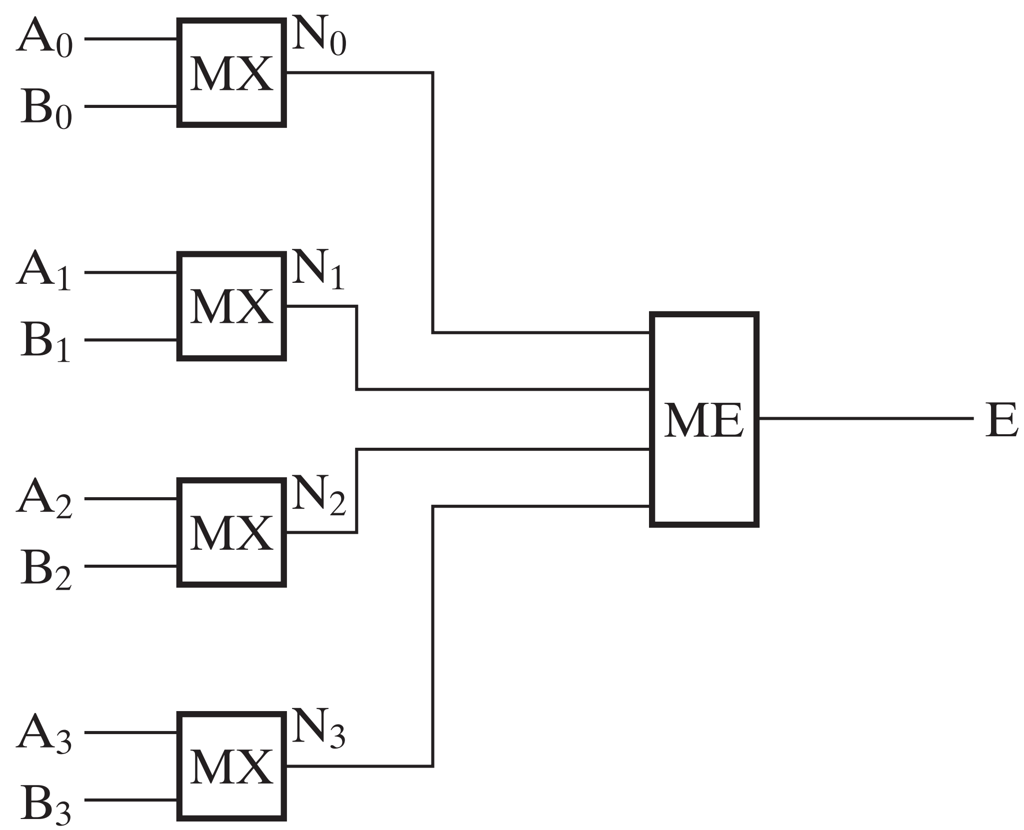
- No DATA inputs
- 2^k 1-bit outputs
- Selector input chooses which output = 1, all other outputs = 0



Building “big” circuits: Hierarchical design

3-4

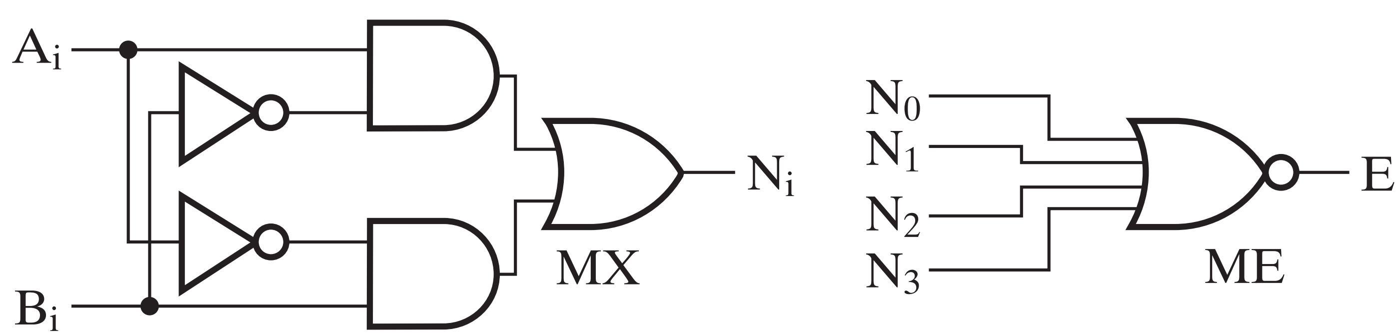
“Big”Circuit



(a)

Design small circuits to be used in a bigger circuit

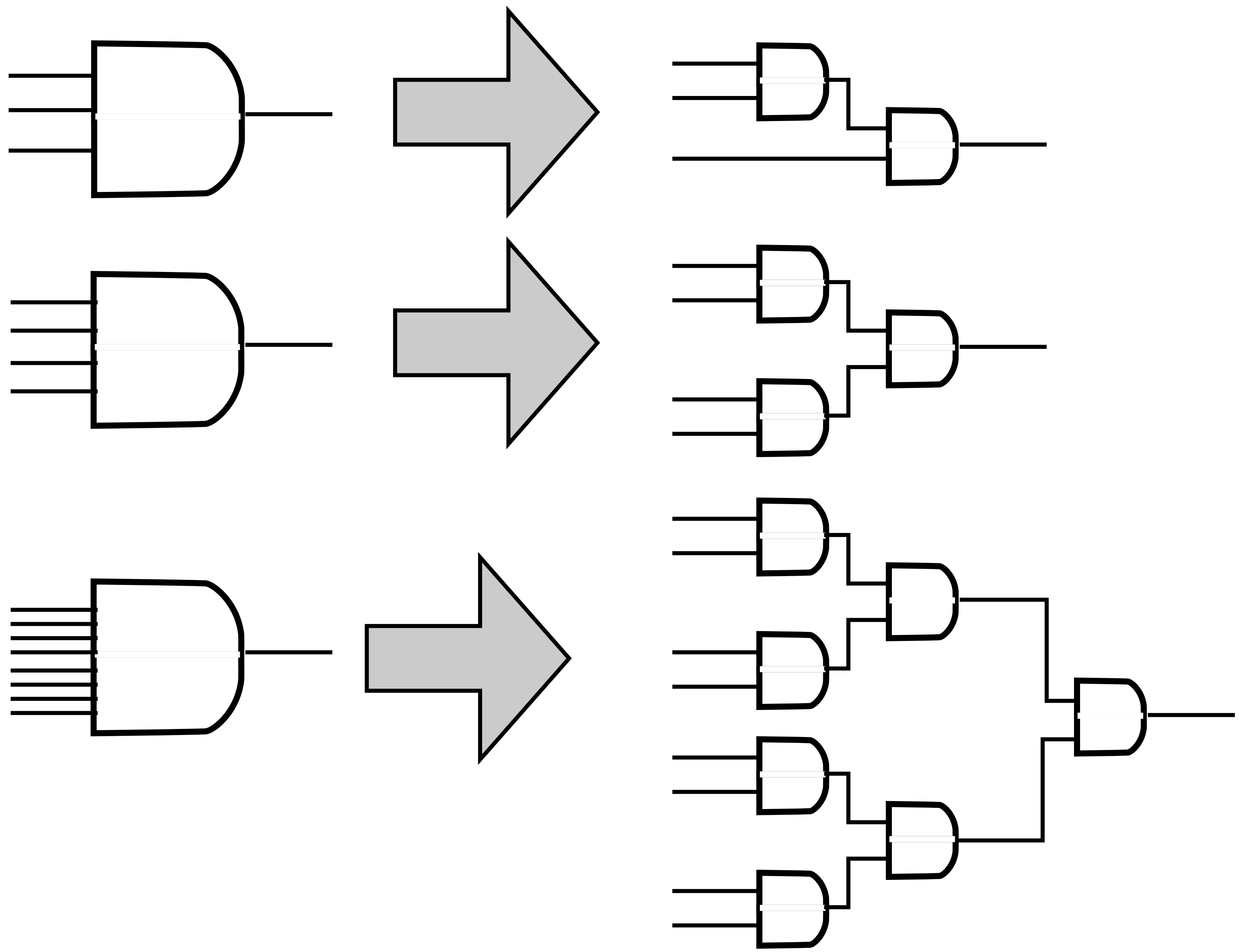
Smaller Circuits



(b) (c)

M. Morris Mano & Charles R. Kime
LOGIC AND COMPUTER DESIGN FUNDAMENTALS, 4e

Notation: Emulating a k-input gate via 2-input

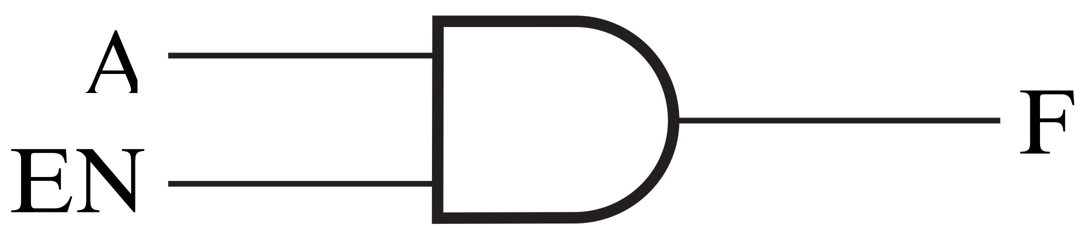


- Each stage in the circuit cuts # of gates by half
- k input gate emulated with $\log_2 k$ depth 2-input gate circuits
- Same process works for OR, XOR as well

Enabler circuits: 1 bit Data input “A”

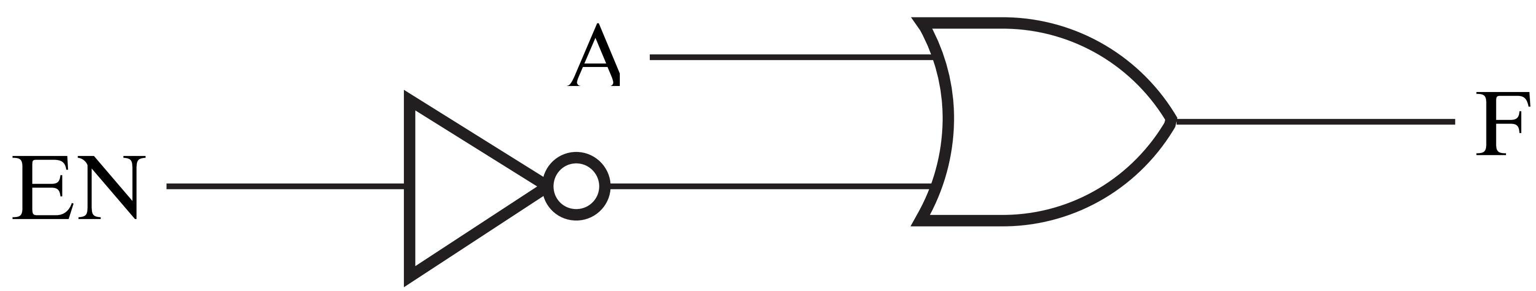
3-15

Abbreviated truth table (input listed in the output)



(a)

EN	F
0	0
1	A



(b)

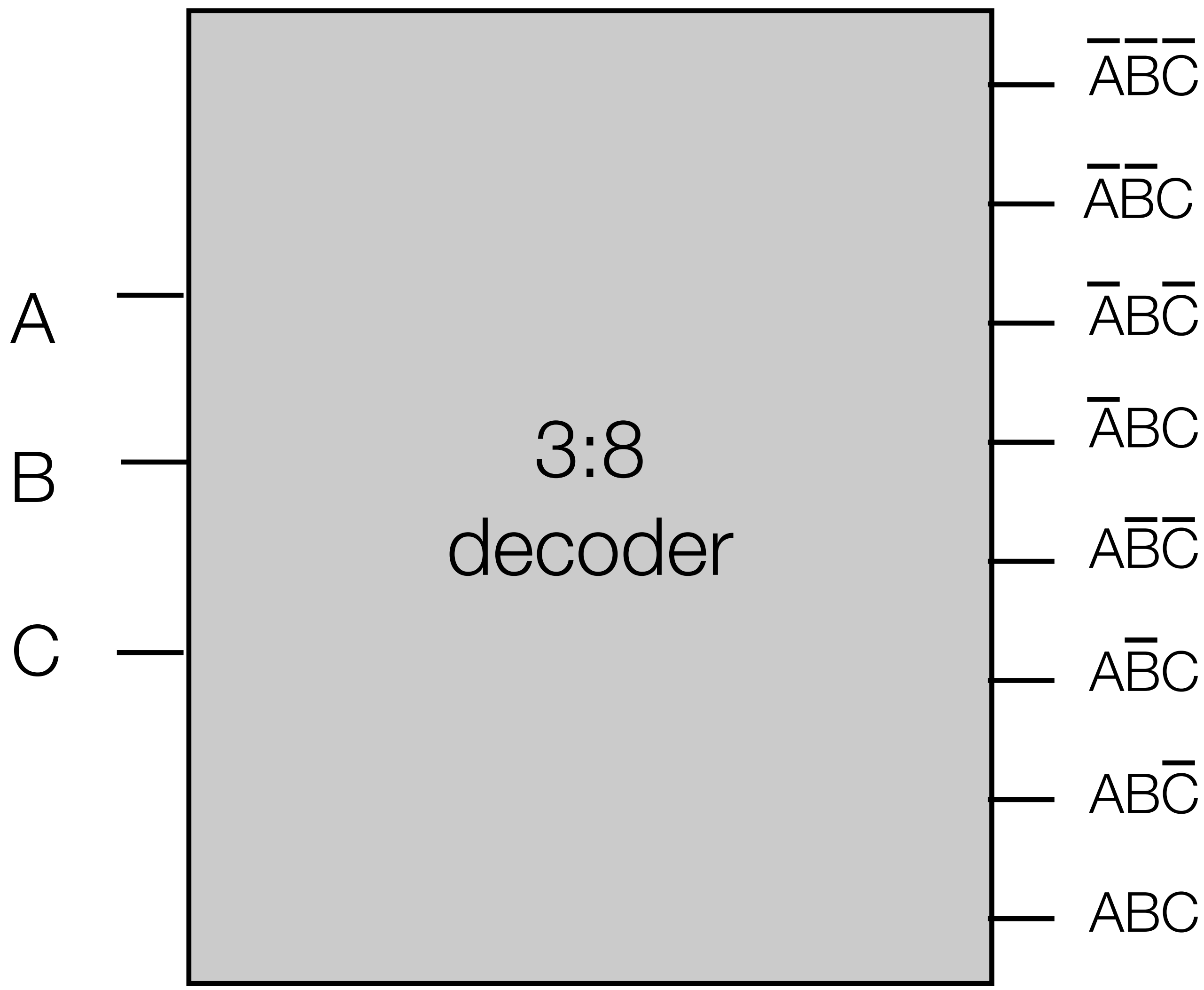
EN	F
0	1
1	A

For both enabler circuits above, output is “enabled” (F=X) only when input ‘ENABLE’ signal is asserted (EN=1). Note the different output when DISABLED

M. Morris Mano & Charles R. Kime
LOGIC AND COMPUTER DESIGN FUNDAMENTALS, 4e

Decoder-based circuits

Converts n-bit input to m-bit output, where n <= m <= 2^n



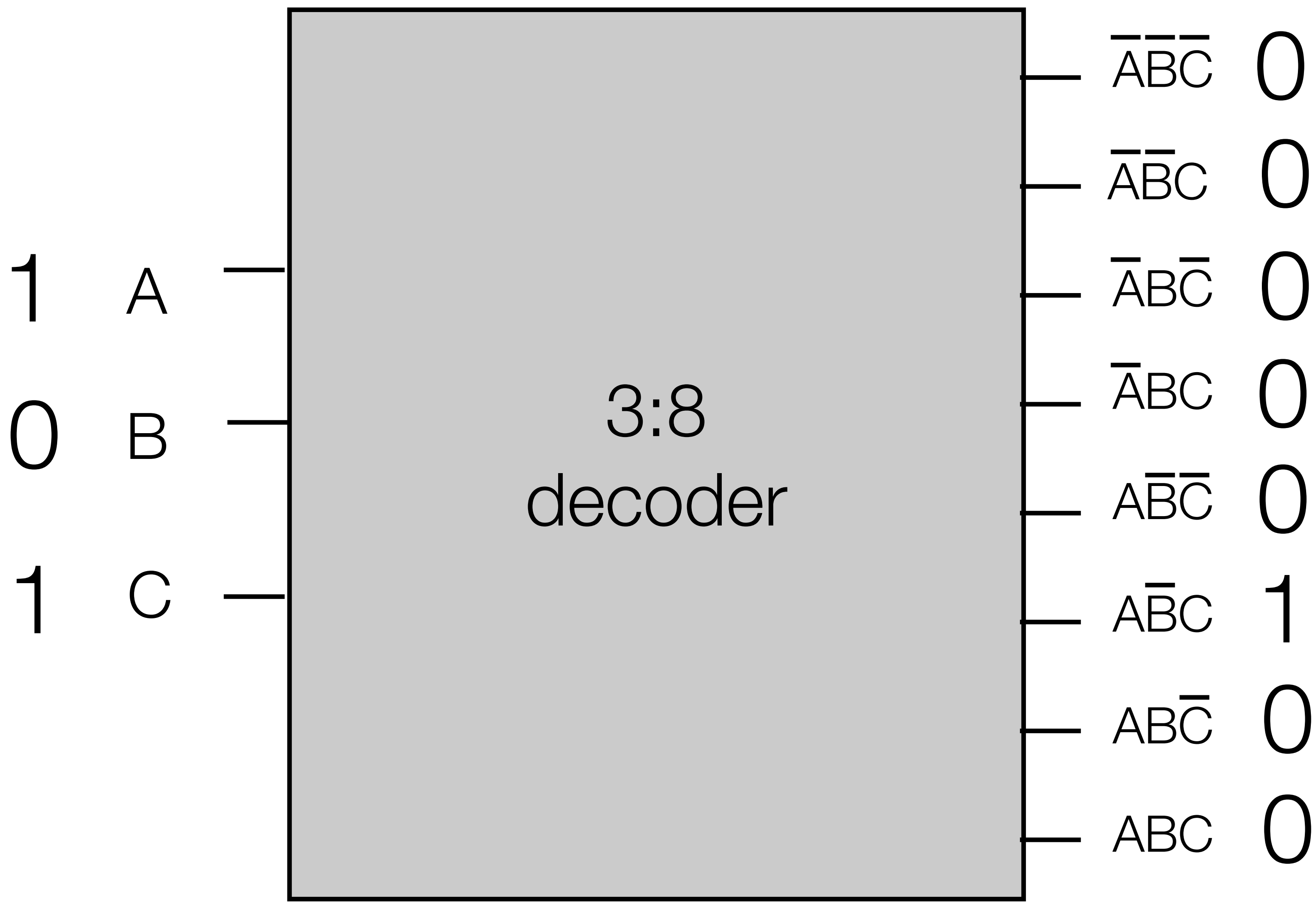
“Standard” Decoder: i^{th} output = 1, all others = 0,

where i is the binary representation of the input (ABC)

Decoder-based circuits

Converts n-bit input to m-bit output, where n <= m <= 2^n

e.g., ABC = 101 (i=5)



"Standard" Decoder: i^{th} output = 1, all others = 0,

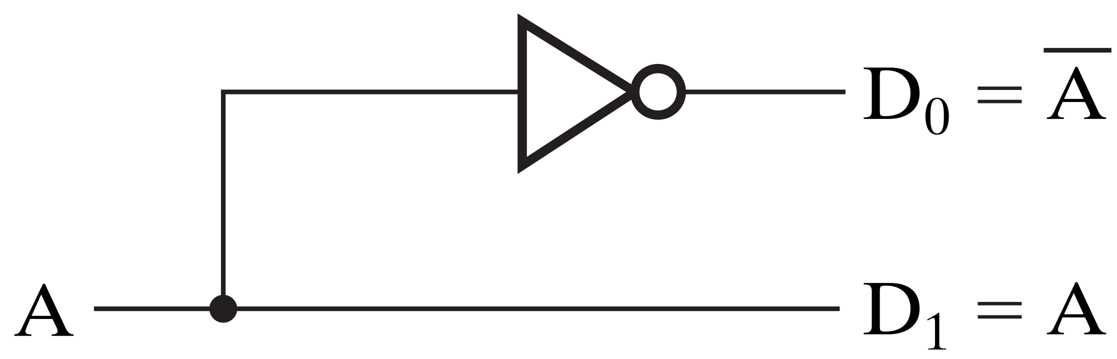
where i is the binary representation of the input (ABC)

Decoder (1:2) Internal Design

3-17

A	**D_0**	**D_1**
0	1	0
1	0	1

(a)



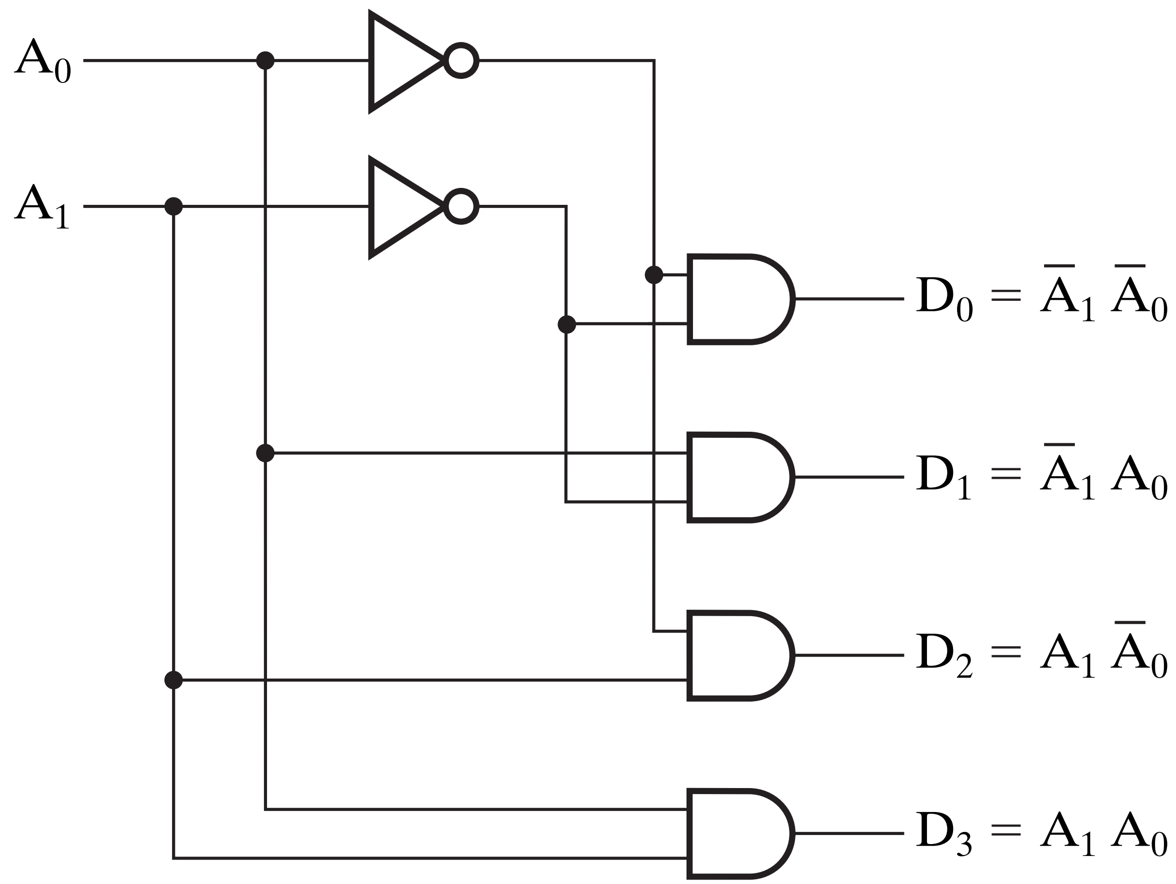
(b)

Decoder (2:4)

3-18

A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a)

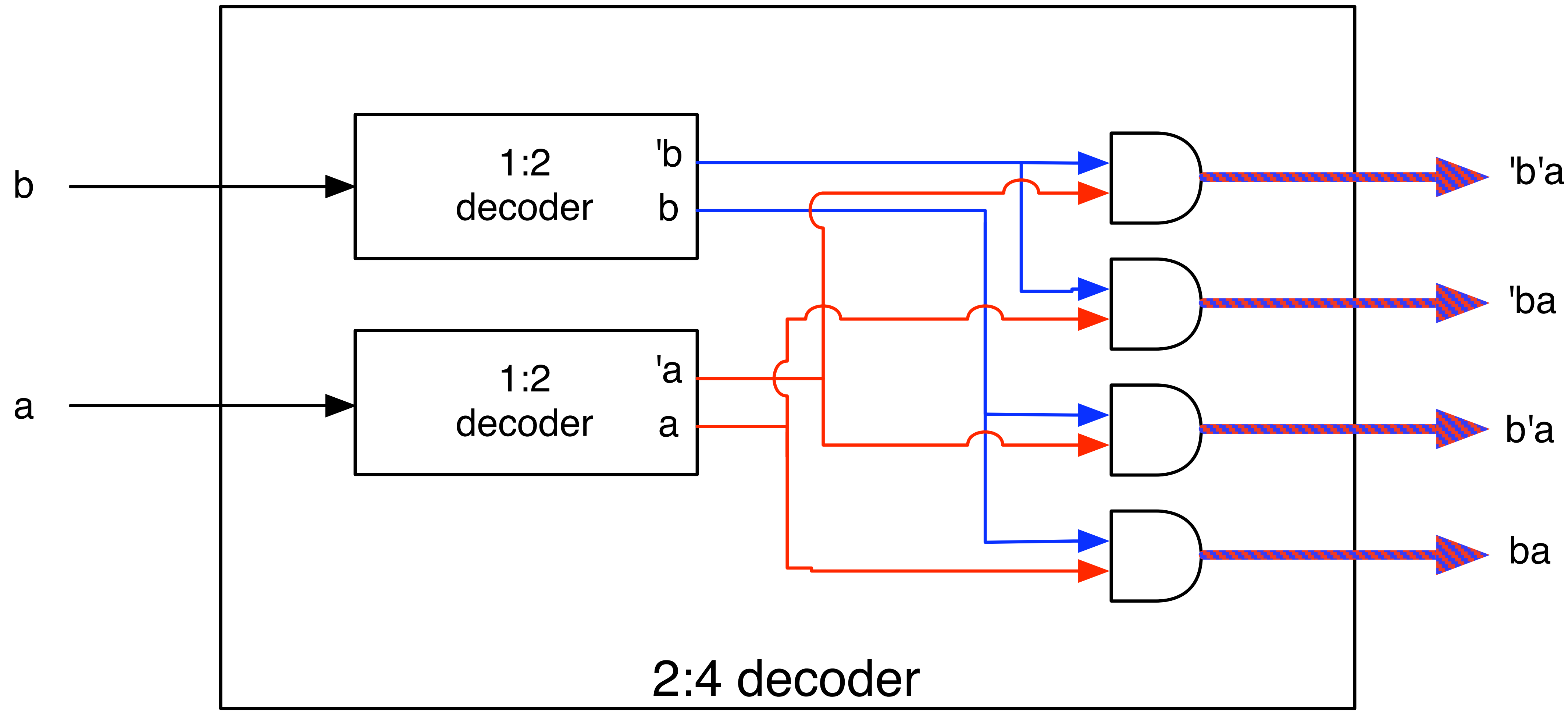


(b)

"Standard" Decoder: i^{th} output = 1, all others = 0, where i is the binary representation of the input

M. Morris Mano & Charles R. Kime
LOGIC AND COMPUTER DESIGN FUNDAMENTALS, 4e

Hierarchical design of decoder (2:4 decoder)

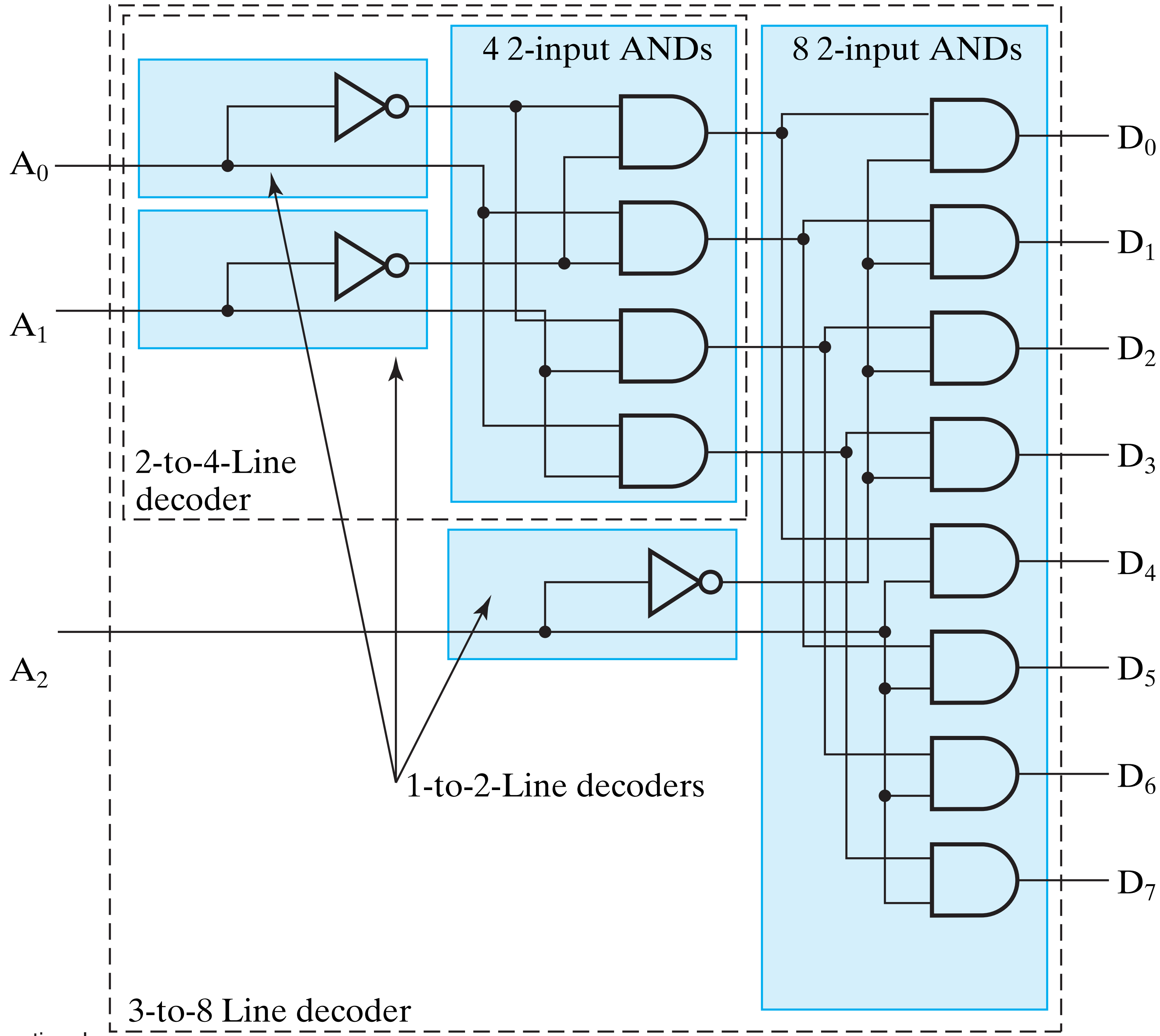


Can build 2:4 decoder out of two 1:2 decoders (and some additional circuitry)

Decoder (3:8)

Hierarchical design: use small decoders to build bigger decoder

3-19



Note: A_2 "selects" whether the 2-to-4 line decoder is active in the top half (A_2=0) or the bottom (A_2=1)

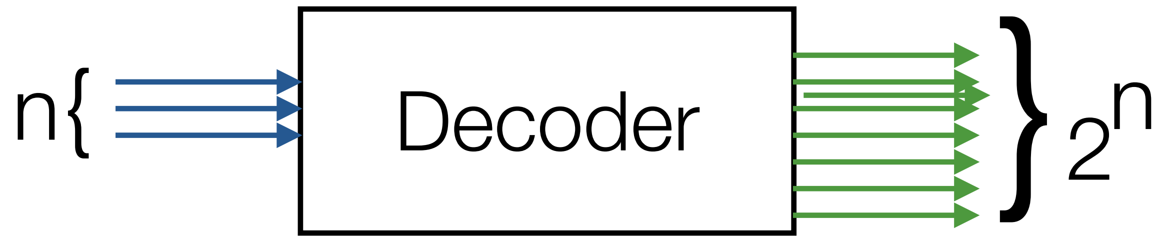
Encoders

Inverse of a decoder: converts m-bit input to n-bit output, where $n \le m \le 2^n$

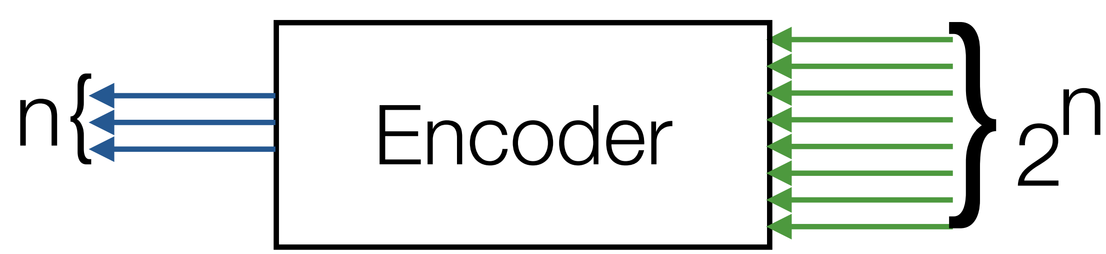
TABLE 3-7
Truth Table for Octal-to-Binary Encoder

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Decoders and encoders



BCD values			One-hot encoding							
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0



Note: for Encoders - input is assumed to be just one 1, the rest 0’s

Priority Encoder

T 3-8

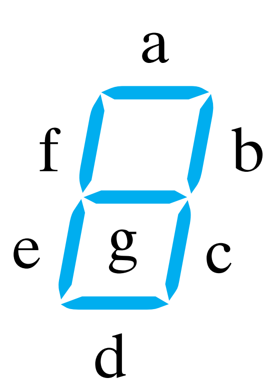
- Designed for any combination of inputs

TABLE 3-8
Truth Table of Priority Encoder

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

General code conversion: a circuit that coverts some inputs to outputs

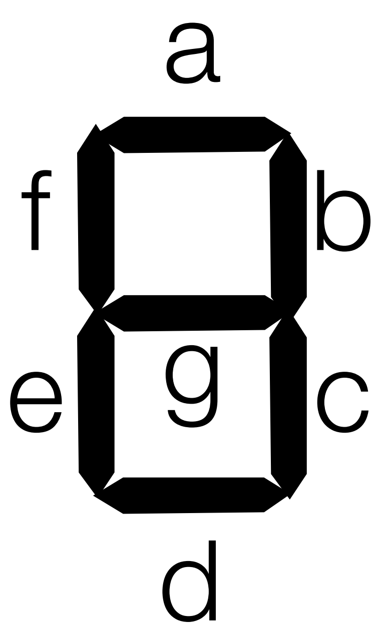
3-3



(a) Segment designation

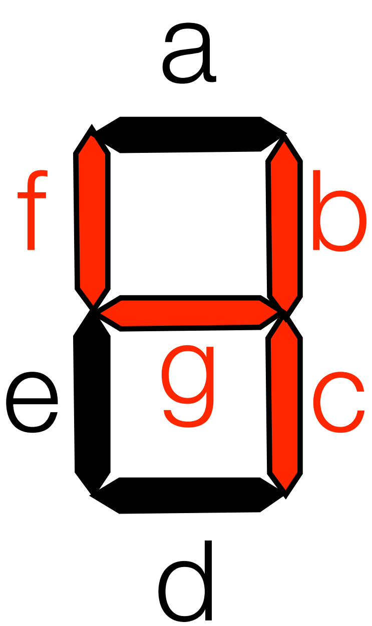
(b) Numeric designation for display

Code conversion



	Input				Output						
Va	W	X	Y	Z	a	b	c	d	e	f	g
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	0	0	1	1
X	1	0	1	0	X	X	X	X	X	X	X
X	1	0	1	1	X	X	X	X	X	X	X
X	1	1	0	0	X	X	X	X	X	X	X
X	1	1	0	1	X	X	X	X	X	X	X
X	1	1	1	0	X	X	X	X	X	X	X
X	1	1	1	1	X	X	X	X	X	X	X

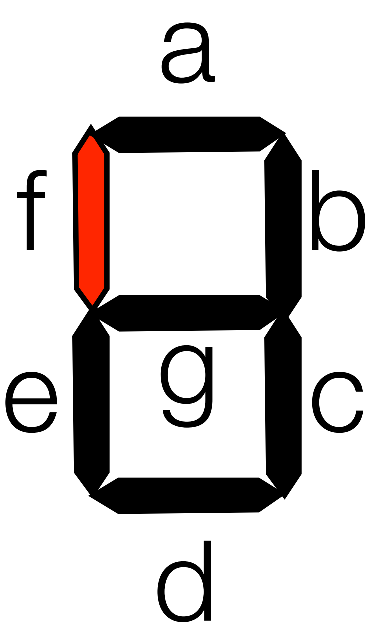
Code conversion



Input					Output						
Va	W	X	Y	Z	a	b	c	d	e	f	g
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	0	0	1	1
X	1	0	1	0	X	X	X	X	X	X	X
X	1	0	1	1	X	X	X	X	X	X	X
X	1	1	0	0	X	X	X	X	X	X	X
X	1	1	0	1	X	X	X	X	X	X	X
X	1	1	1	0	X	X	X	X	X	X	X
X	1	1	1	1	X	X	X	X	X	X	X

e.g., what outputs "lights up" when input V=4?

Code conversion



For what values does output f “light up” for?

Y=1

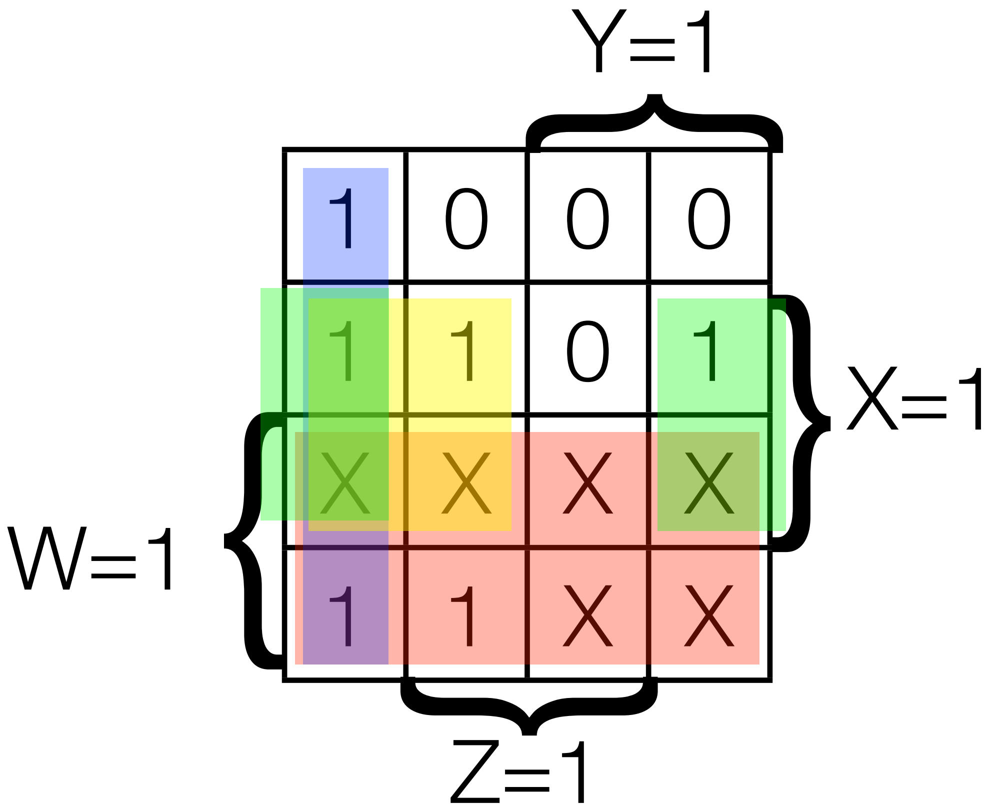
	1	0	0	0
	1	1	0	1
W=1	X	X	X	X
W=1	1	1	X	X

X=1

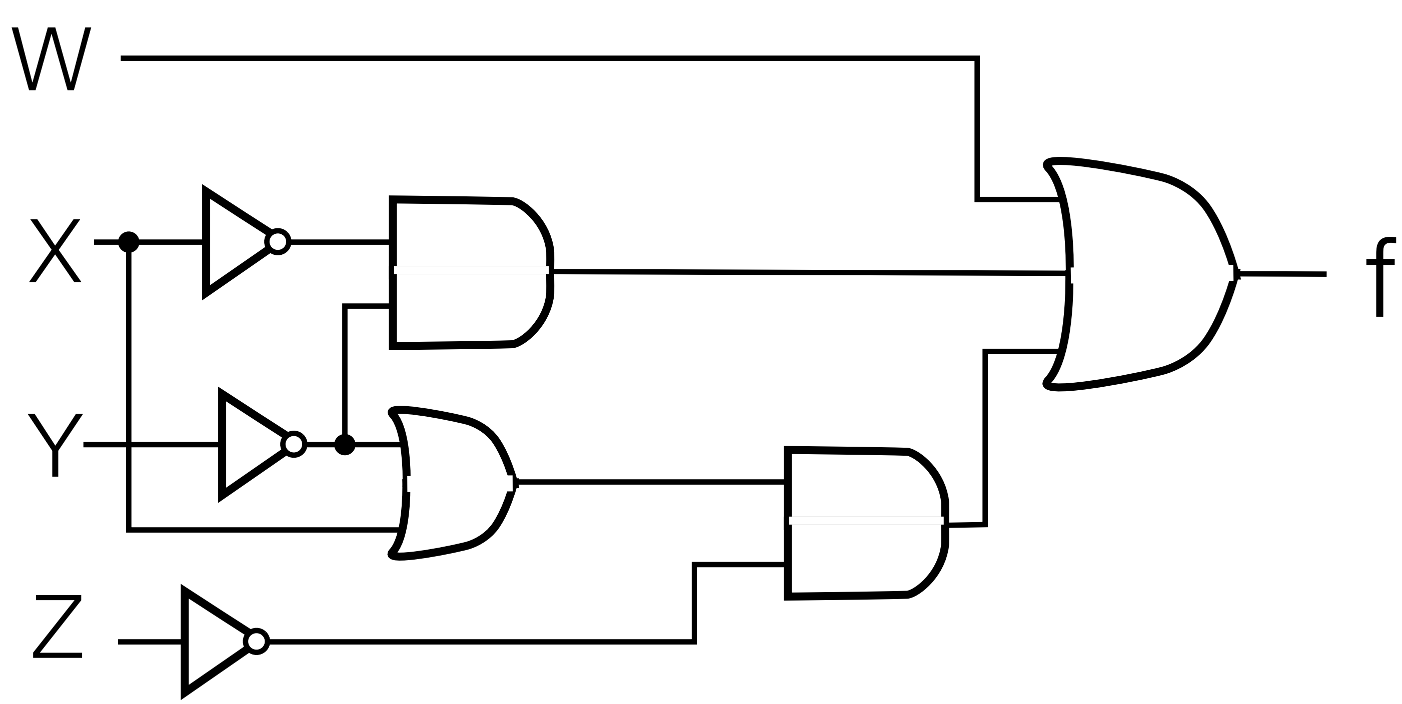
Z=1

Input					Output						
Va	W	X	Y	Z	a	b	c	d	e	f	g
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	0	0	1	1
X	1	0	1	0	X	X	X	X	X	X	X
X	1	0	1	1	X	X	X	X	X	X	X
X	1	1	0	0	X	X	X	X	X	X	X
X	1	1	0	1	X	X	X	X	X	X	X
X	1	1	1	0	X	X	X	X	X	X	X
X	1	1	1	1	X	X	X	X	X	X	X

Algebra and Circuit for “f”



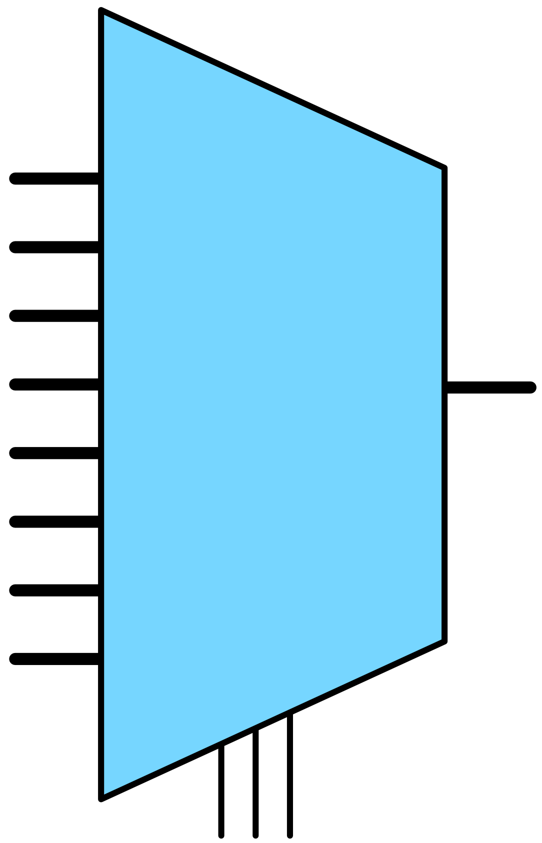
$$f = W + \overline{Y}\,\overline{Z} + X\overline{Z} + \overline{X}\,\overline{Y} = W + (X+\overline{Y})\overline{Z} + \overline{X}\,\overline{Y}$$



Multiplexers

- Combinational circuit that **selects** binary information from one of many input lines and directs it to one output line

2^n inputs

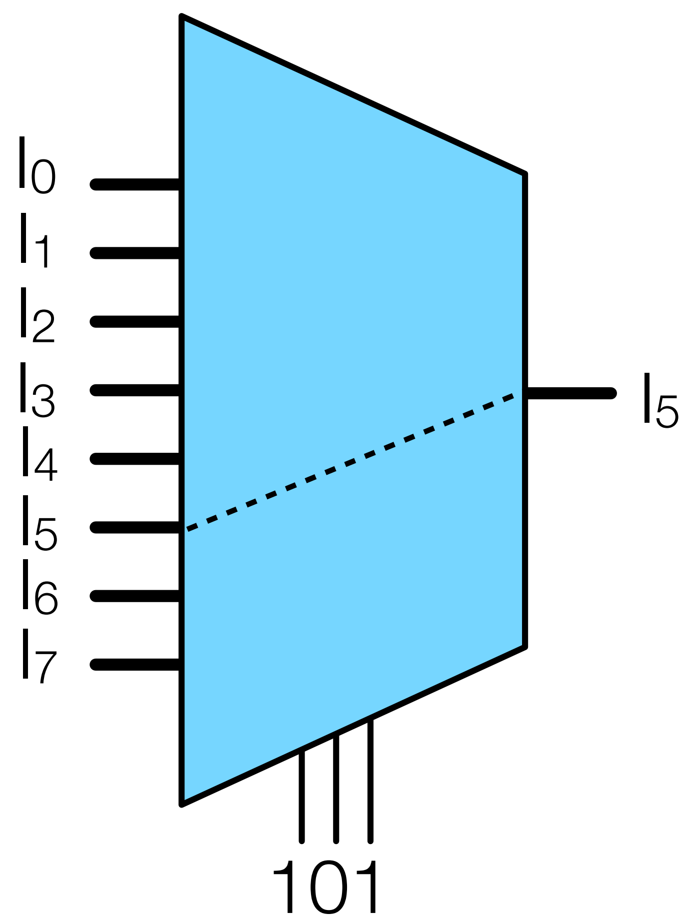


1 output

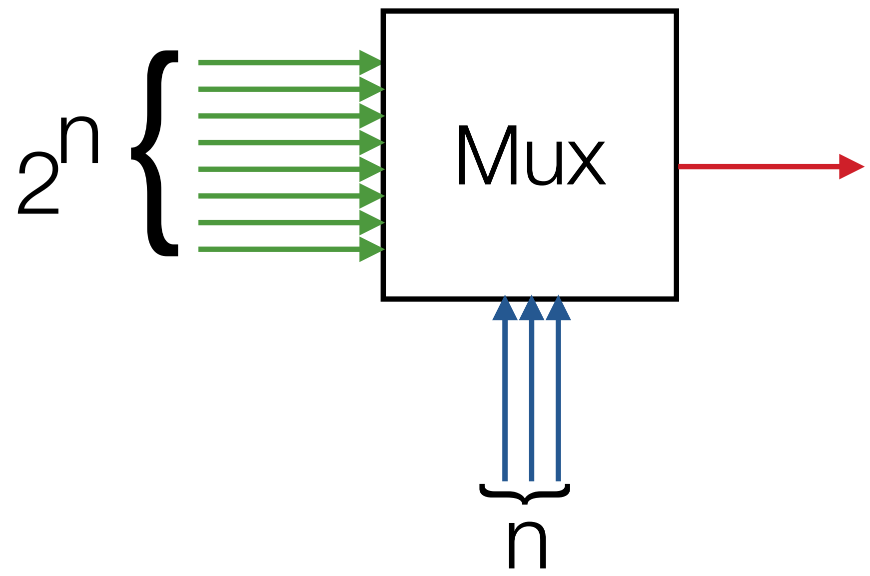
n selection bits

indicate (in binary) which input feeds to the output

Multiplexer example

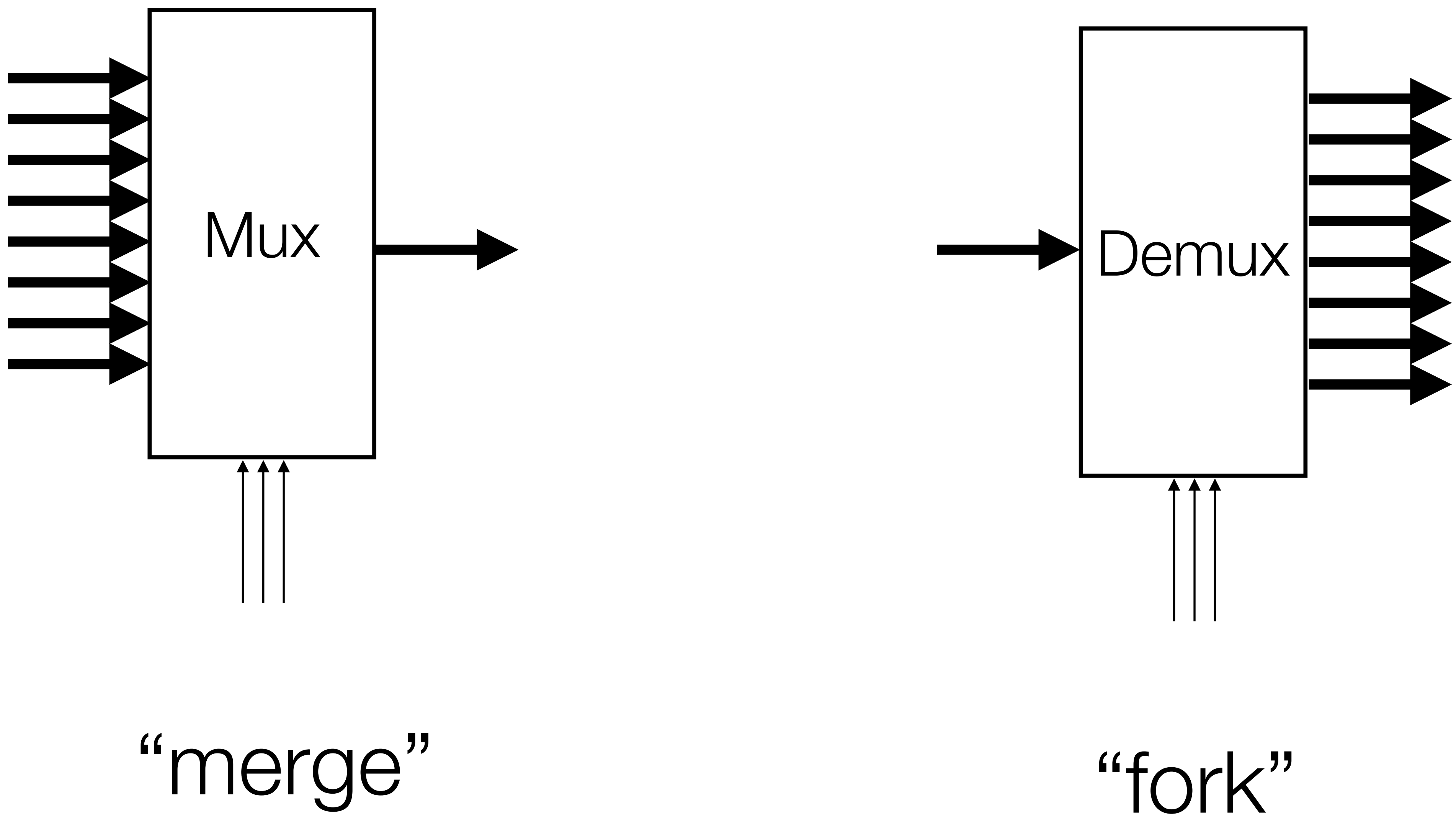


Multiplexers & Demultiplexers

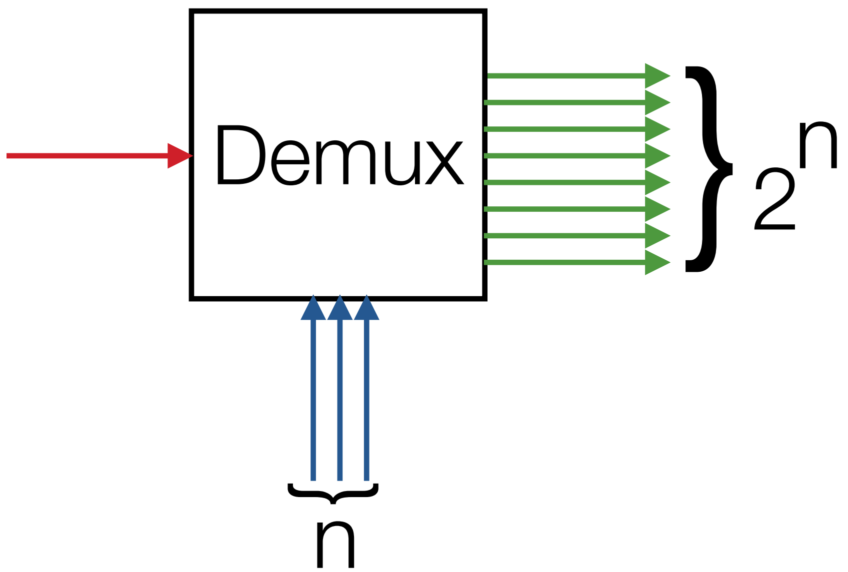


2^n inputs								n-bit BCD value			1 output
a	x	x	x	x	x	x	x	0	0	0	a
x	b	x	x	x	x	x	x	0	0	1	b
x	x	c	x	x	x	x	x	0	1	0	c
x	x	x	d	x	x	x	x	0	1	1	d
x	x	x	x	e	x	x	x	1	0	0	e
x	x	x	x	x	f	x	x	1	0	1	f
x	x	x	x	x	x	g	x	1	1	0	g
x	x	x	x	x	x	x	h	1	1	1	h

Muxes and demuxes called “steering logic”



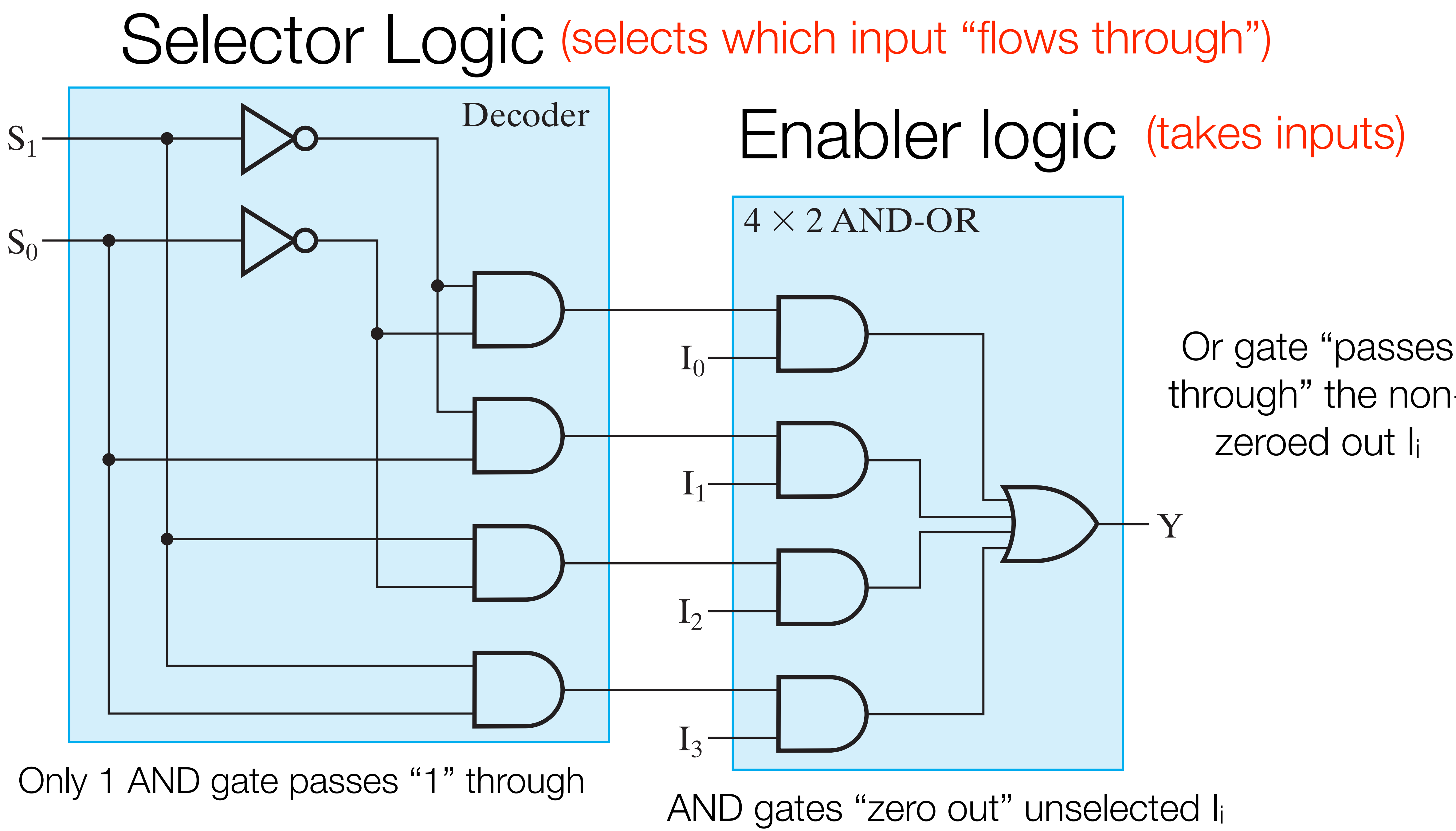
Demultiplexers



1 input	n-bit BCD value			2^n outputs							
a	0	0	0	a	0	0	0	0	0	0	0
b	0	0	1	0	b	0	0	0	0	0	0
c	0	1	0	0	0	c	0	0	0	0	0
d	0	1	1	0	0	0	d	0	0	0	0
e	1	0	0	0	0	0	0	e	0	0	0
f	1	0	1	0	0	0	0	0	f	0	0
g	1	1	0	0	0	0	0	0	0	g	0
h	1	1	1	0	0	0	0	0	0	0	h

Internal mux organization

3-26

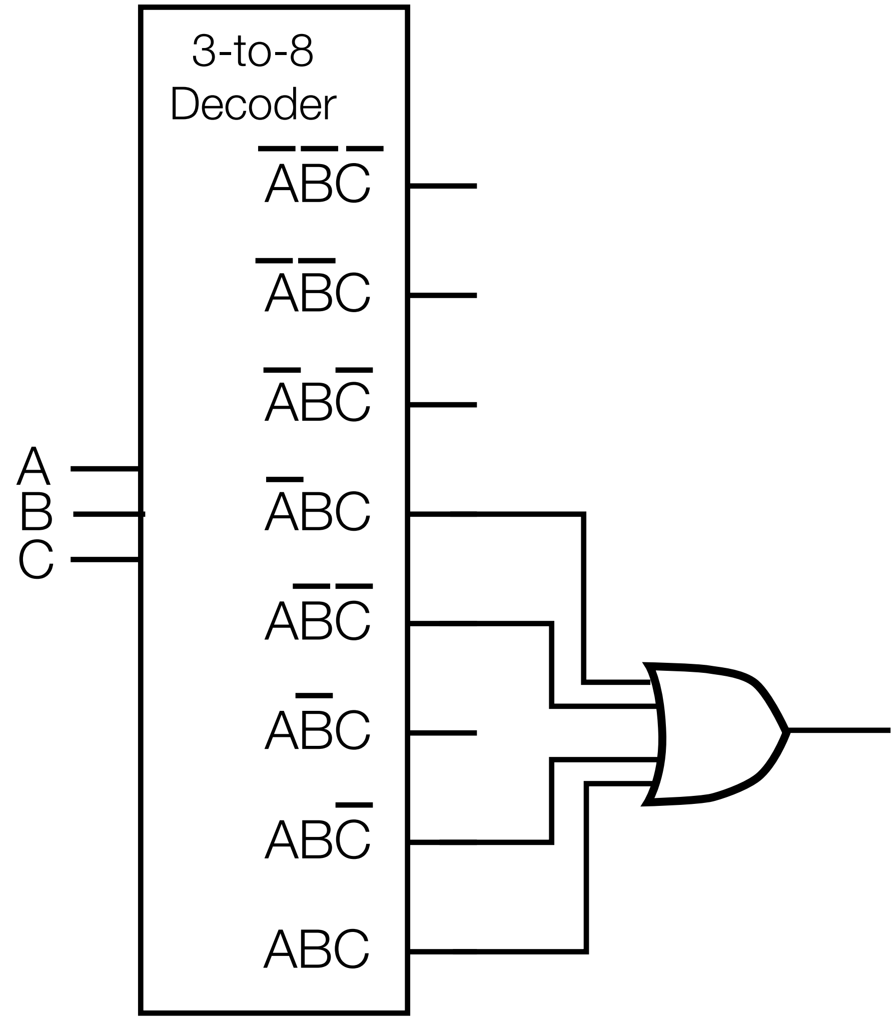


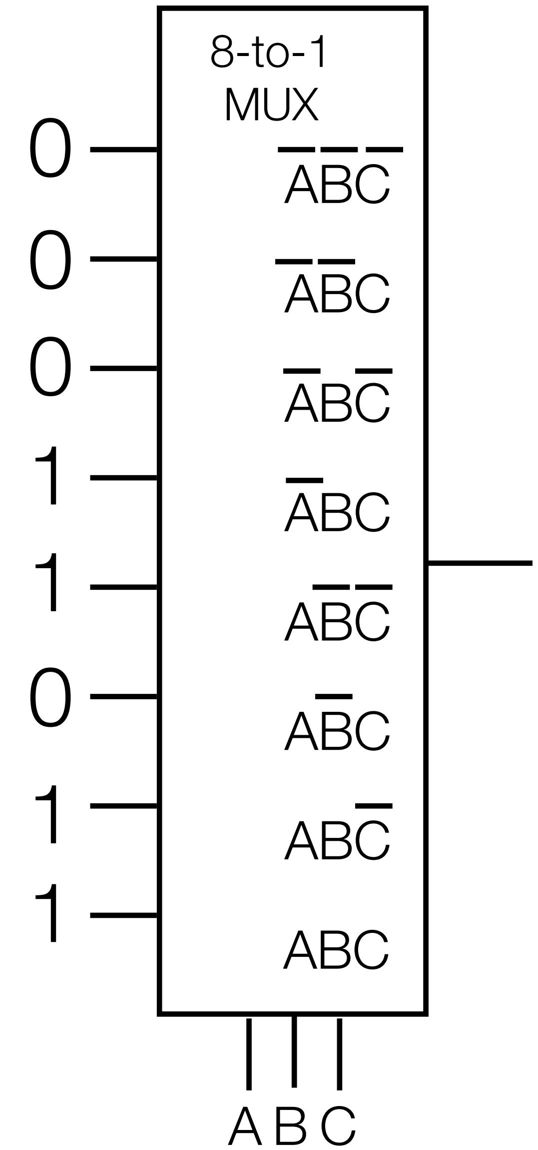
M. Morris Mano & Charles R. Kime
LOGIC AND COMPUTER DESIGN FUNDAMENTALS, 4e

Representing Functions with Decoders and MUXes

- e.g., $F = A\overline{C} + BC$

A	B	C	minterm	F
0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	0
0	0	1	$\overline{A}\,\overline{B}C$	0
0	1	0	$\overline{A}B\overline{C}$	0
0	1	1	$\overline{A}BC$	1
1	0	0	$A\overline{B}\,\overline{C}$	1
1	0	1	$A\overline{B}C$	0
1	1	0	$AB\overline{C}$	1
1	1	1	ABC	1



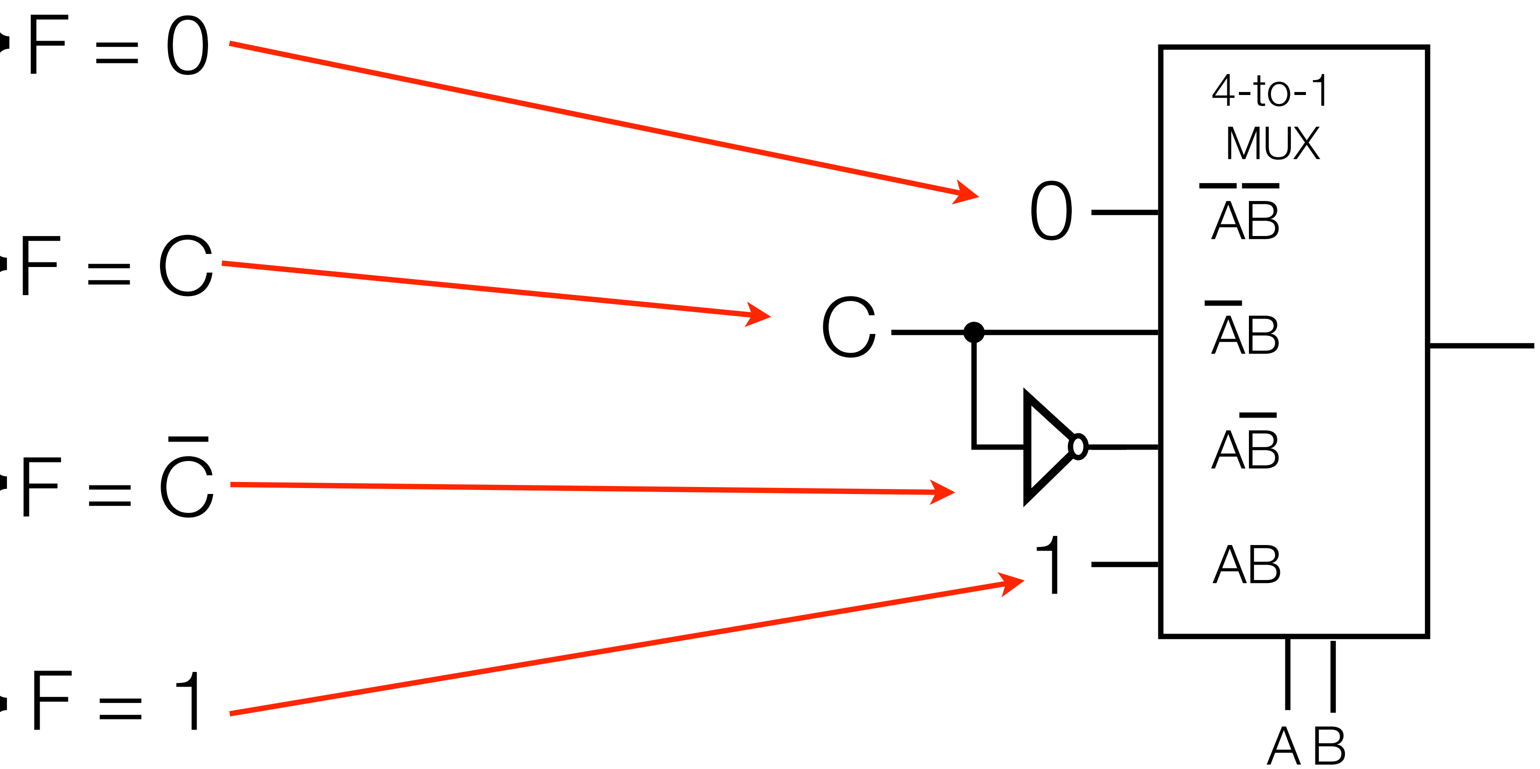


- Decoder: OR minterms for which F should evaluate to 1
- MUX: Feed in the value of F for each minterm

A Slick MUX trick

- Can use a smaller MUX with a little trick e.g., $F = AC + B\overline{C}$
- Note for rows paired below, A&B have same values, C iterates between 0&1
- For the pair of rows, F either equals 0, 1, C or $\overline{C}$

A	B	C	minterm	F
0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	0
0	0	1	$\overline{A}\,\overline{B}C$	0
0	1	0	$\overline{A}B\overline{C}$	0
0	1	1	$\overline{A}BC$	1
1	0	0	$A\overline{B}\,\overline{C}$	1
1	0	1	$A\overline{B}C$	0
1	1	0	$AB\overline{C}$	1
1	1	1	ABC	1



Slick MUX trick: Example

- e.g., $F = \overline{A}C + \overline{B}\,\overline{C} + A\overline{C}$

A	B	C	minterm	F
0	0	0	$\overline{A}\,\overline{B}\,\overline{C}$	1
0	0	1	$\overline{A}\,\overline{B}C$	1
0	1	0	$\overline{A}B\overline{C}$	0
0	1	1	$\overline{A}BC$	1
1	0	0	$A\overline{B}\,\overline{C}$	1
1	0	1	$A\overline{B}C$	0
1	1	0	$AB\overline{C}$	1
1	1	1	ABC	0

$F = 1$

$F = C$

$F = \overline{C}$

$F = \overline{C}$