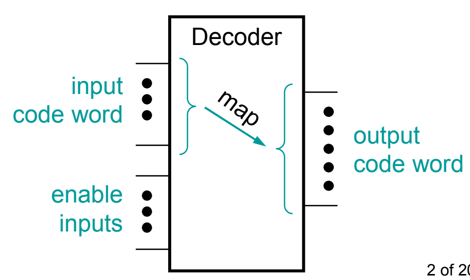
14:332:231
DIGITAL LOGIC DESIGN

Ivan Marsic, Rutgers University
Electrical & Computer Engineering
Fall 2013

Lecture #10: Decoders

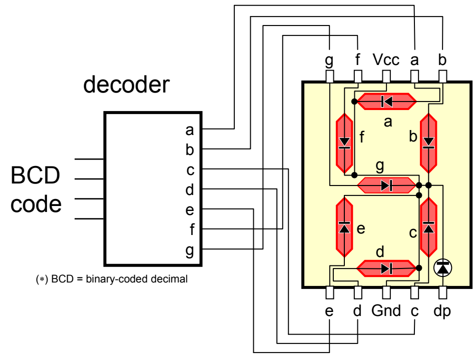
General Decoder Structure

- A decoder is a logic circuit that converts coded inputs into coded outputs.
- Each input code word produces a different output code word (there is a one-to-one mapping between inputs and outputs)



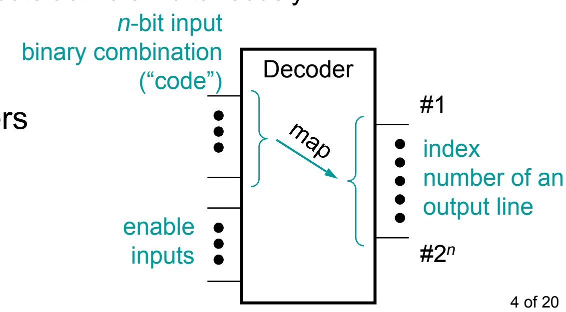
Decoder Example

- BCD to seven-segment decoder
  - has 4-bit BCD as input code and the "seven-segment code" as its output code



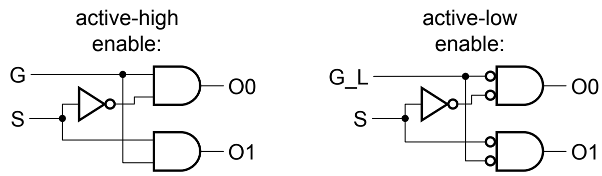
Binary Decoder

- Accepts a *n*-bit binary input code and generates a 1-out-of-2^n output code
- Used to activate exactly one of 2^n outputs based on *n*-bit input value
- Examples: 2-to-4, 3-to-8, 4-to-16, etc.
- Note: BCD to seven-segment decoder is NOT a binary decoder
  - Because multiple outputs active simultaneously
- Binary decoders are simple and general; can be used to build general decoders (shown later)

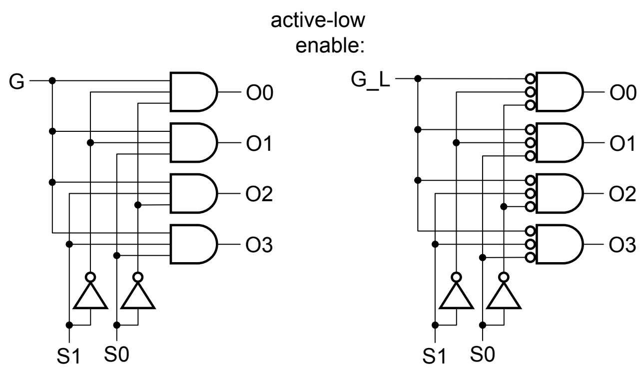


Gate Level Implementation of Decoders

- 1:2 decoders

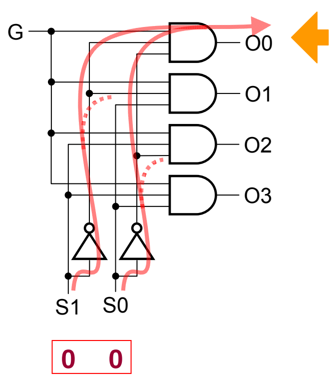


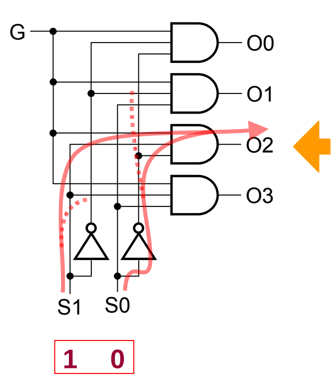
- 2:4 decoders



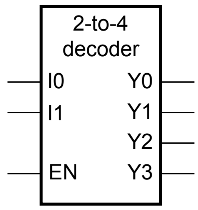
How It Works

- 2:4 decoder:
  - input combination: "00"
  - output: O_0
  - input combination: "10"
  - output: O_2





Binary 2-to-4 Decoder



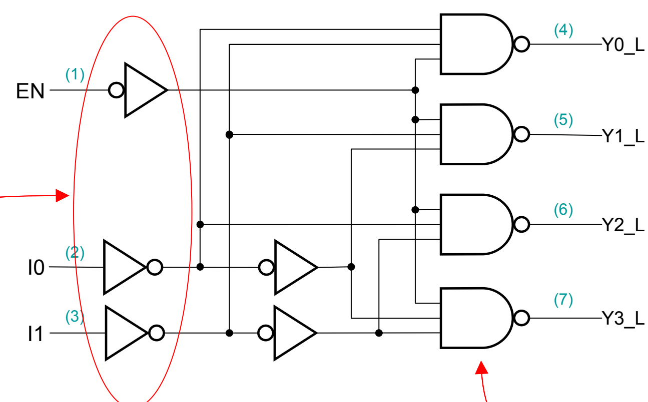
Inputs			*Outputs*			
EN	I1	I0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Note "x" (don't care) notation.

- Note that the outputs of the decoder correspond to the *minterms*: $Y_i = m_i$
  - e.g., Y0 = I1′ · I0′
  - Y1 = I1′ · I0 etc.

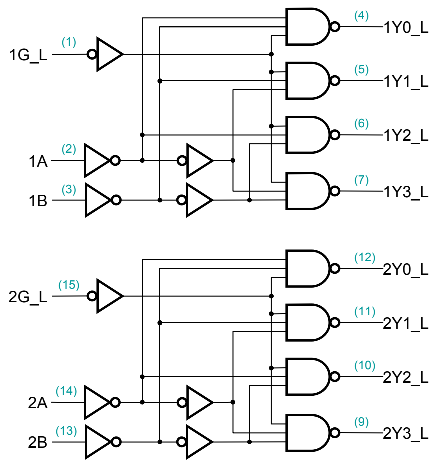
MSI 2-to-4 Decoder

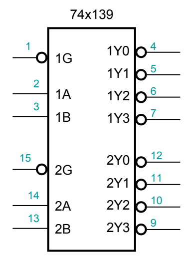
(* COMPARE TO Wakerly, 4th edition, Figure 6-32(b), page 385)



- Input buffering (less load on input circuit)
- NAND gates (faster operation)

Complete 74x139 Decoder



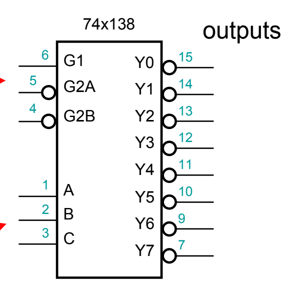


Two 2-to-4 decoders in a single packaging

(* COMPARE TO 74x138 3-to-8 decoder, described next)

74x138: 3-to-8 decoder

- Commercially available MSI 3-to-8 decoder
  - Note that its outputs are active low
    - » because TTL and CMOS *inverting* gates are faster than non-inverting gates



- Logic equations for internal output signals include “enable” signals.
- Example:

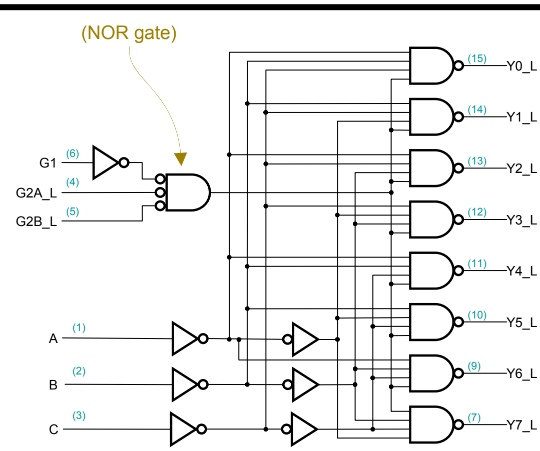
$$Y5 = \underbrace{G1 \cdot G2A \cdot G2B}_{\text{enable}} \cdot \underbrace{C \cdot B' \cdot A}_{\text{select}}$$

Truth Table for a 3-to-8 Decoder

Inputs						*Outputs*							
G1	G2A_L	G2B_L	C	B	A	Y7_L	Y6_L	Y5_L	Y4_L	Y3_L	Y2_L	Y1_L	Y0_L
0	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

- Because of the inversion bubbles, we have the following relations between internal and external signals
  G2A = G2A_L′
  Y5 = Y5_L′ etc.

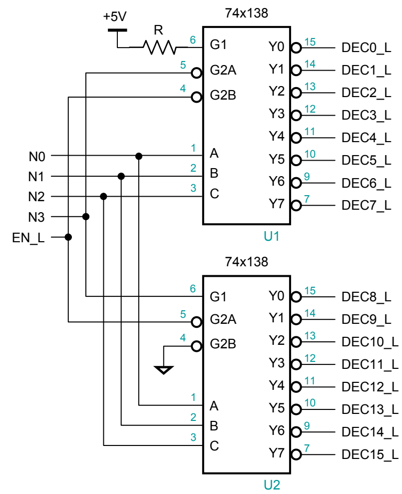
3-to-8 Decoder Logic Diagram



Decoder Cascading

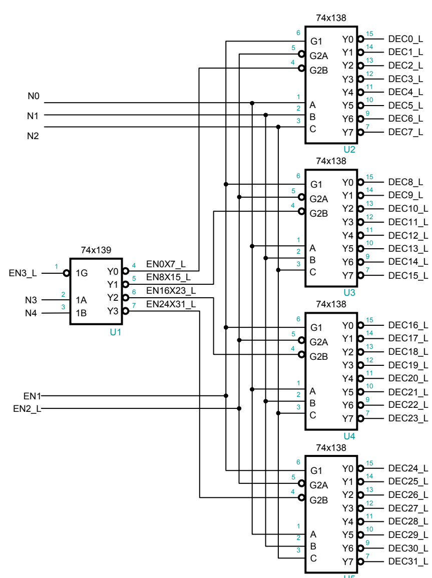
- Decoders can be cascaded hierarchically to decode larger code words

- Example:
  Design a 4-to-16 decoder using 74x128s (3-to-8 decoders)



More Cascading

5-to-32 decoder

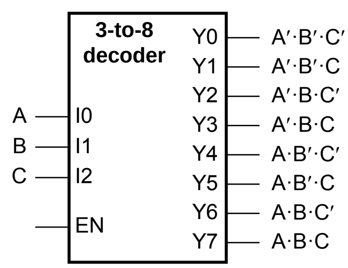


Decoder Applications

- Microprocessor memory systems
  - Selecting different banks of memory
- Microprocessor input/output systems
  - Selecting different devices
- Microprocessor instruction decoding
  - Enabling different functional units
- Memory chips
  - Enabling different rows of memory depending on address
- Lots of other applications

Decoders as General-Purpose Logic

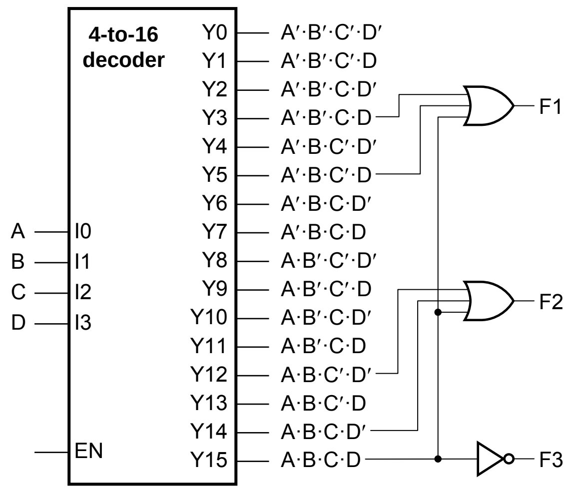
- n-to-2^n decoders can implement any function of n variables
  - with the variables used as control inputs
  - the appropriate *minterms* summed to form the function



decoder generates appropriate *minterm* based on *control signals* (it “decodes” control signals)

Decoders as General-Purpose Logic

- F1 = A′·B·C′·D + A′·B′·C·D + A·B·C·D
- F2 = A·B·C′·D′ + A·B·C
- F3 = A′ + B′ + C′ + D′

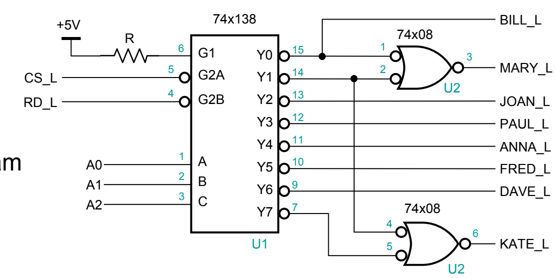


Customized Decoder Circuit

CS_L	RD_L	A2	A1	A0	***Output(s) to Assert***
1	x	x	x	x	none
x	1	x	x	x	none
0	0	0	0	0	BILL_L, MARY_L
0	0	0	0	1	MARY_L, KATE_L
0	0	0	1	0	JOAN_L
0	0	0	1	1	PAUL_L
0	0	1	0	0	ANNA_L
0	0	1	0	1	FRED_L
0	0	1	1	0	DAVE_L
0	0	1	1	1	KATE_L

Truth table

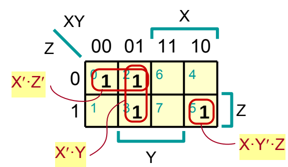
Circuit diagram

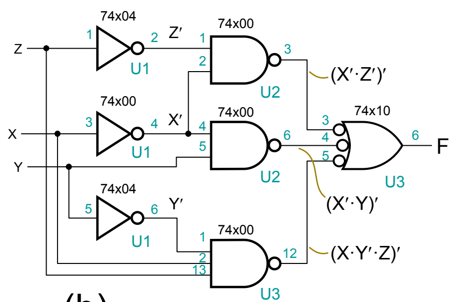


Decoder-Based Circuits

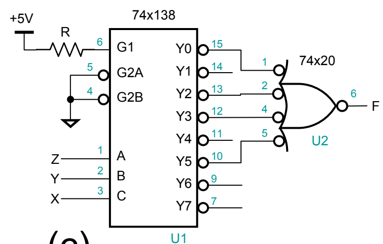
Designing a circuit for the logic function
$F = \sum_{X,Y,Z}(0,2,3,5)$:
(a) Karnaugh map;
(b) NAND-based minimal sum-of-products;
(c) decoder-based canonical sum.

(a)





(b)



(c)

Multiple Decoding w/ a Single Decoder