14:332:231
DIGITAL LOGIC DESIGN

Ivan Marsic, Rutgers University
Electrical & Computer Engineering
Fall 2013

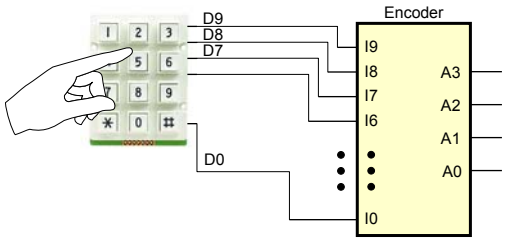
Lecture #11: Encoders

Encoders versus Decoders

- An encoder performs the inverse function as a decoder
- The simplest encoder to build is a 2^n-to-n (**binary encoder**)

Example: A decimal-to-BCD encoder

- A decimal-to-BCD encoder
  - Inputs: 10 bits corresponding to decimal digits 0 through 9, (D_0, …, D_9)
  - Outputs: 4 bits with BCD codes
  - Function: If input bit D_i is a "1", then the output (A_3, A_2, A_1, A_0) is the BCD code for *i*



Example: "7", "3", "8" ➡ "0111", "0011", "1000"

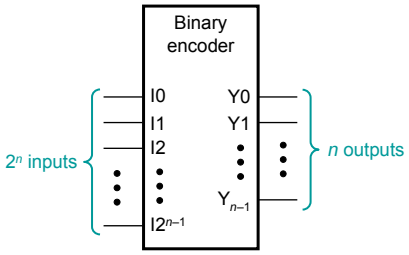
Truth table of the decimal-to-BCD encoder

- From the truth table, encoder outputs:
  A3 = D8 + D9
  A2 = D4 + D5 + D6 + D7
  A1 = D2 + D3 + D6 + D7
  A0 = D1 + D3 + D5 + D7 + D9
- We made use of the fact that only one input can be "1" at one time
- Note that if none button is pushed, output is also "0000"
- What if two buttons are pushed simultaneously? —E.g. D1 and D2 together: A0=A1=1 and A2=A3=0 (0011) which is the same as if D3 were pushed!

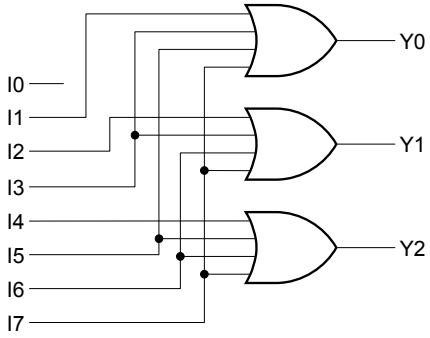
Inputs										*Outputs*			
D9	**D8**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	**A3**	**A2**	**A1**	**A0**
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

Binary Encoders

- General structure:



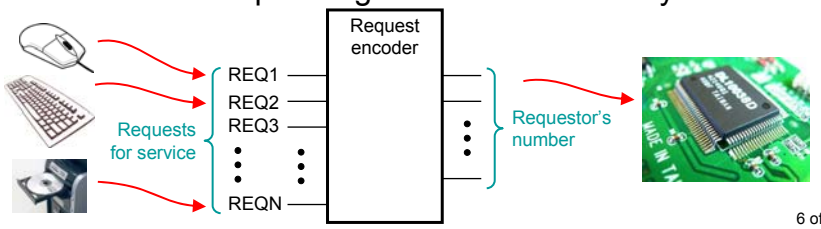
- 8-to-3 encoder:



Y0 = I1 + I3 + I5 + I7
Y1 = I2 + I3 + I6 + I7
Y3 = I4 + I5 + I6 + I7

Priority Encoders

- If **more than one input value is "1"**, then the encoder just designed does not work properly
- An encoder that can accept all possible combinations of input values and produce a meaningful result is a ***priority encoder***
- Among the "1"s that appear, it **selects the most significant input position** (or the least significant input position) containing a "1" and produces the corresponding binary code for that position
- A system with 2^n requestors and a "request encoder" that indicates which request signal is asserted at any time:



Exercise: Design a 4-input priority encoder with active low inputs

- Highest priority is given to most significant input "1" present (I3 … I0)
- Code outputs: A1, A0 and IDLE (indicates no input present)

Intermediate variables:

$H3 = I3'$

$H2 = I2' \cdot I3$

$H1 = I1' \cdot I2 \cdot I3$

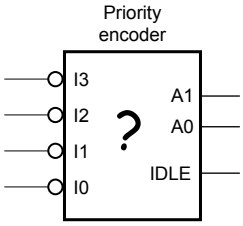
$H0 = I0' \cdot I1 \cdot I2 \cdot I3$

Encoder outputs:

$A1 = H2 + H3$

$A0 = H1 + H3$

$IDLE = I3 \cdot I2 \cdot I1 \cdot I0$



Truth table:

Inputs				Outputs		
I3	I2	I1	I0	A1	A0	IDLE
1	1	1	1	x	x	1
1	1	1	0	0	0	0
1	1	0	x	0	1	0
1	0	x	x	1	0	0
0	x	x	x	1	1	0

Exercise: Design a 4-input priority encoder with active low inputs

- Highest priority is given to most significant input "1" present (I3 … I0)
- Code outputs: A1, A0 and IDLE (indicates no input present)
- We could use a Karnaugh map to get equations, but can be read directly from the truth table and manually optimized if careful:

$A1 = I3 \cdot I2' + I3' = I2' + I3'$

$A0 = I3 \cdot I2 \cdot I1' + I3' = I2 \cdot I1' + I3'$

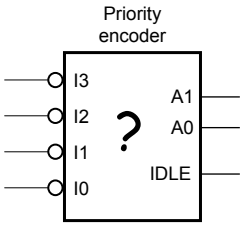
$IDLE = I3 \cdot I2 \cdot I1 \cdot I0$

$X \cdot Y' + X' = [\,(T2)\ X + 1 = 1\,]$

$X \cdot Y' + X' \cdot (Y'+1) =$

$X \cdot Y' + X' \cdot Y' + X' = [\,(T10)\ X \cdot Y + X \cdot Y' = X\,]$

$Y' + X'$

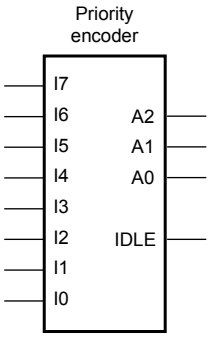


Truth table:

Inputs				Outputs		
I3	I2	I1	I0	A1	A0	IDLE
1	1	1	1	x	x	1
1	1	1	0	0	0	0
1	1	0	x	0	1	0
1	0	x	x	1	0	0
0	x	x	x	1	1	0

8-input Priority Encoder

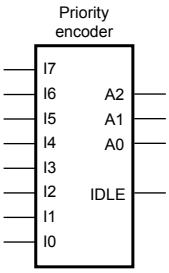
- Logic symbol for a generic 8-input priority encoder



9 of 18

A Generic 8-input Priority Encoder

- Truth table



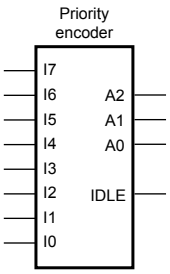
Inputs								*Outputs*			
I7	**I6**	**I5**	**I4**	**I3**	**I2**	**I1**	**I0**	**A2**	**A1**	**A0**	**IDLE**
0	0	0	0	0	0	0	0	0	0	0	1
1	x	x	x	x	x	x	x	1	1	1	0
0	1	x	x	x	x	x	x	1	1	0	0
0	0	1	x	x	x	x	x	1	0	1	0
0	0	0	1	x	x	x	x	1	0	0	0
0	0	0	0	1	x	x	x	0	1	1	0
0	0	0	0	0	1	x	x	0	1	0	0
0	0	0	0	0	0	1	x	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	0

10 of 18

Priority-Encoder Logic Equations

Define intermediate variables, s.t. Hi is "1" if and only if Ii is the highest-priority input:

$$H7 = I7$$
$$H6 = I6 \cdot I7'$$
$$H5 = I5 \cdot I6' \cdot I7'$$
$$\ldots$$
$$H0 = I0 \cdot I1' \cdot I2' \cdot I3' \cdot I4' \cdot I5' \cdot I6' \cdot I7'$$



Encoder outputs:

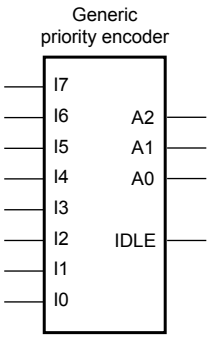
$$A2 = H4 + H5 + H6 + H7$$
$$A1 = H2 + H3 + H6 + H7$$
$$A0 = H1 + H3 + H5 + H7$$

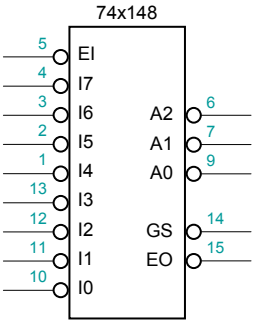
The IDLE output is "1" if no inputs are "1":

$$IDLE = (I0 + I1 + I2 + I3 + I4 + I5 + I6 + I7)'$$
$$= I0' \cdot I1' \cdot I2' \cdot I3' \cdot I4' \cdot I5' \cdot I6' \cdot I7'$$

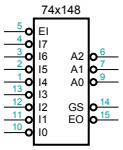
74x148 8-input Priority Encoder

- Active-low I/O
- EI = Enable Input
- GS = Group Select ("Got Something", as opposed to IDLE="Got Nothing")
  - One or more of the request inputs are asserted and EI is asserted
- EO = Enable Output
  - No request input is asserted and EI is asserted (corresponds to IDLE)





Truth Table for a 74x148 Encoder



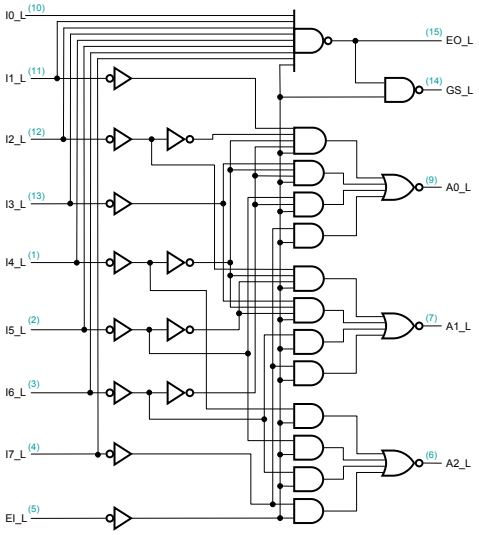
pass

Inputs									Outputs				
EI_L	I0_L	I1_L	I2_L	I3_L	I4_L	I5_L	I6_L	I7_L	A2_L	A1_L	A0_L	GS_L	EO_L
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	x	x	x	x	x	x	x	0	0	0	0	0	1
0	x	x	x	x	x	x	0	1	0	0	1	0	1
0	x	x	x	x	x	0	1	1	0	1	0	0	1
0	x	x	x	x	0	1	1	1	0	1	1	0	1
0	x	x	x	0	1	1	1	1	1	0	0	0	1
0	x	x	0	1	1	1	1	1	1	0	1	0	1
0	x	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0

selected

74x148 Logic Circuit

- 74x148:
  - Enable Input (**EI**) must be asserted for any output to be asserted
  - Group Select (**GS**) is asserted if ≥1 inputs are asserted and EI is asserted
  - Enable Output (**EO**) is asserted if *no input* is asserted and EI is asserted



Example 74x148 Outputs

- 74x148 the **A0_L** output:

  A0_L = 0 for I7_L, I5_L, I3_L, I1_L

  A0_L =

  (EI·I7 + EI·I6_L·I5 + EI·I6_L·I4_L·I3 +
  + EI·I6_L·I4_L·I2_L·I1)′

- 74x148 the **A1_L** output:

  A1_L = 0 for I7_L, I6_L, I3_L, I2_L

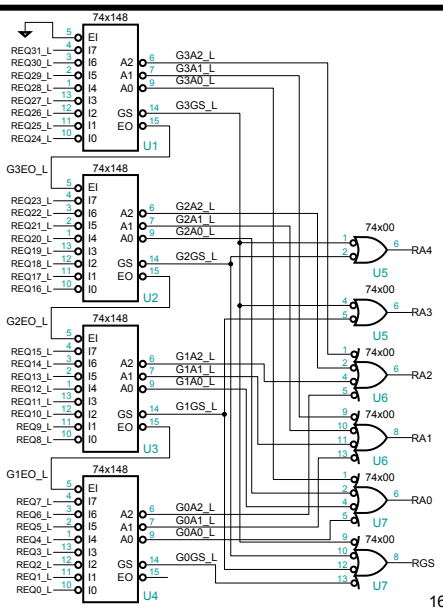
  A1_L =

  (EI·I7 + EI·I6 + EI·I5_L·I4_L·I3 +
  + EI·I5_L·I4_L·I2)′

15 of 18

Cascading Priority Encoders

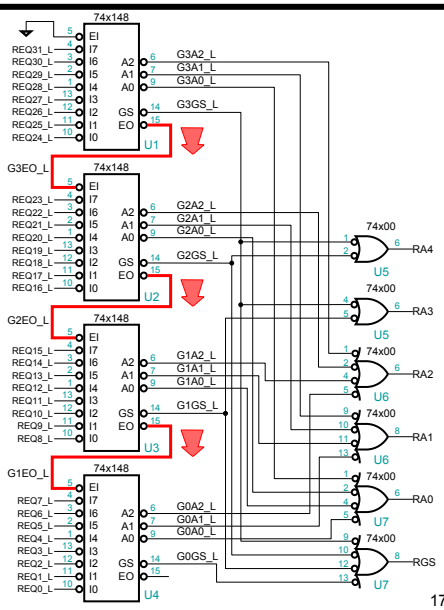
- 32-input priority encoder using four cascaded 74x148s



16 of 18

Cascading Priority Encoders

- EO_L is used for cascading to Enable Input of a lower-priority 74x148
- If none of the requests on the highest priority encoder is asserted (i.e., it's IDLE), the next decoder in the cascade is enabled, etc.
- RA4-RA0 encode the highest-priority requestor



17 of 18

Some outputs of the four 74x148 cascade

- RA3 is "1" when 8–15 and 24–31 lines are requesting

  RA3 = G1·GS + G3·GS

- RA1 is "1" when the second digit of the request is "1"

  RA1 = G0·A1 + G1·A1 + G2·A1 + G3·A1

- RCS is "1" when there is a request

  RGS = G0·GS + G1·GS + G2·GS + G3·GS

18 of 18