Introduction to Computer Engineering – EECS 203
http://ziyang.eecs.northwestern.edu/~dickrp/eecs203/

Instructor: Robert Dick
Office: L477 Tech
Email: dickrp@northwestern.edu
Phone: 847-467-2298

TA: Neal Oza
Office: Tech. Inst. L375
Phone: 847-467-0033
Email: nealoza@u.northwestern.edu

TT: David Bild
Office: Tech. Inst. L470
Phone: 847-491-2083
Email: d-bild@northwestern.edu

NORTHWESTERN UNIVERSITY

Encoders

- Assume you have n one-bit signals
- Only one signal can be 1 at a time
- How many states can you be in?
- How many signals are required to encode all those states?

Encoder example

Pressed (i_2, i_1, i_0)	Code (o_1, o_0)
000	00
001	01
010	10
011	XX
100	11
101	XX
110	XX
111	XX

Implementation?

Priority encoder

What if we want the highest-order high signal to dominate?

Pressed (i_3, i_2, i_1)	Code (o_1, o_0)
000	00
001	01
010	10
011	10
100	11
101	11
110	11
111	11

What impact on implementation efficiency?

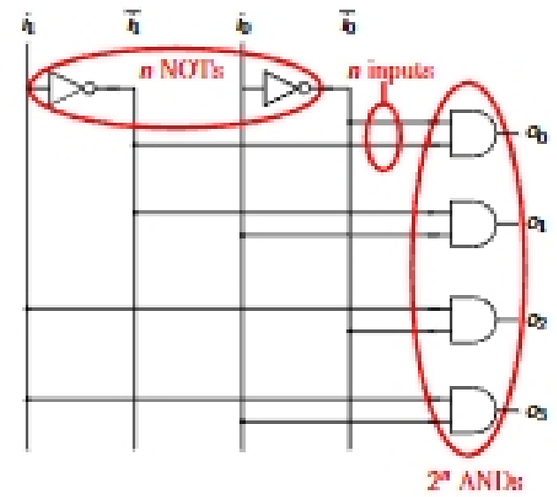
Decoders

Need to map back from encoded signal to state

Pressed (i_1, i_0)	Code (o_3, o_2, o_1, o_0)
00	0001
01	0010
10	0100
11	1000

o_0 isn't always used. Why?
Most straightforward implementation?

Straight-forward decoder implementation



Decoder implementation efficiency

- n NOTs
- n^2 n-input ANDS
- $\mathcal{O}(n^2)$
- Can't do this for large number of inputs!
- Instead, decompose into multi-level implementation

Multilevel decoder implementation

Starting point

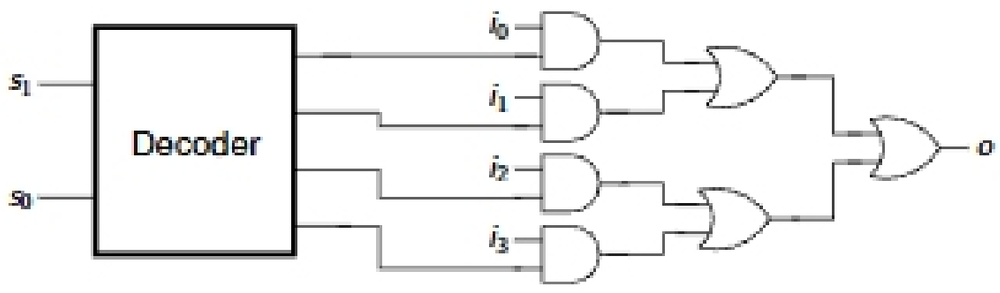
$$o_0 = \bar{i}_2 \bar{i}_1 \bar{i}_0$$
$$o_1 = \bar{i}_2 \bar{i}_1 i_0$$
$$o_2 = \bar{i}_2 i_1 \bar{i}_0$$
$$o_3 = \bar{i}_2 i_1 i_0$$
$$o_4 = i_2 \bar{i}_1 \bar{i}_0$$
$$o_5 = i_2 \bar{i}_1 i_0$$
$$o_6 = i_2 i_1 \bar{i}_0$$
$$o_7 = i_2 i_1 i_0$$

Multilevel decoder implementation

$$o_0 = \overline{i_2}\,(\overline{i_1}\,\overline{i_0})$$
$$o_1 = \overline{i_2}\,(\overline{i_1}\,i_0)$$
$$o_2 = \overline{i_2}\,(i_1\,\overline{i_0})$$
$$o_3 = \overline{i_2}\,(i_1\,i_0)$$
$$o_4 = i_2(\overline{i_1}\,\overline{i_0})$$
$$o_5 = i_2(\overline{i_1}\,i_0)$$
$$o_6 = i_2(i_1\,\overline{i_0})$$
$$o_7 = i_2(i_1\,i_0)$$

Reuse terms! Schematic?

Logic gate MUX

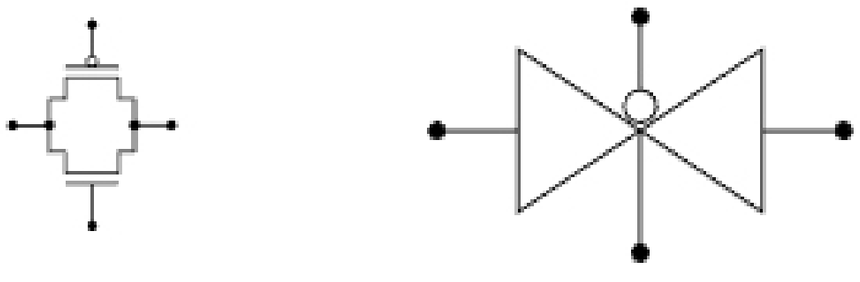


However, there is another way...

MUX truth table

I_1	I_0	C	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

Review: Other TG diagram



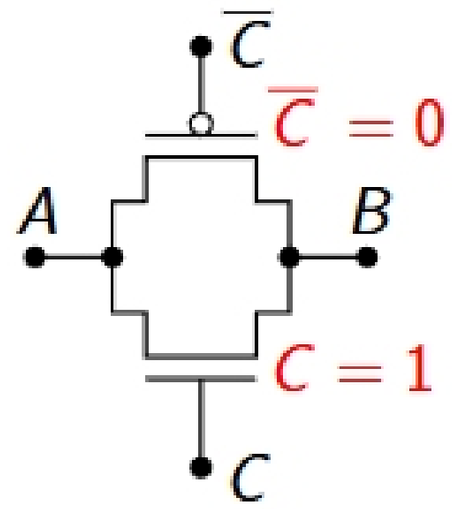
Multiplexers or selectors

- Routes one of 2^n inputs to one output
- n control lines
- Can implement with logic gates

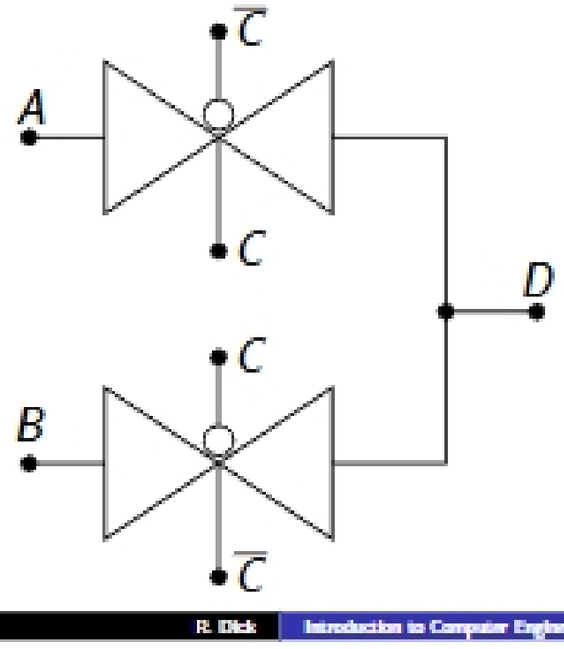
MUX functional table

C	Z
0	I_0
1	I_1

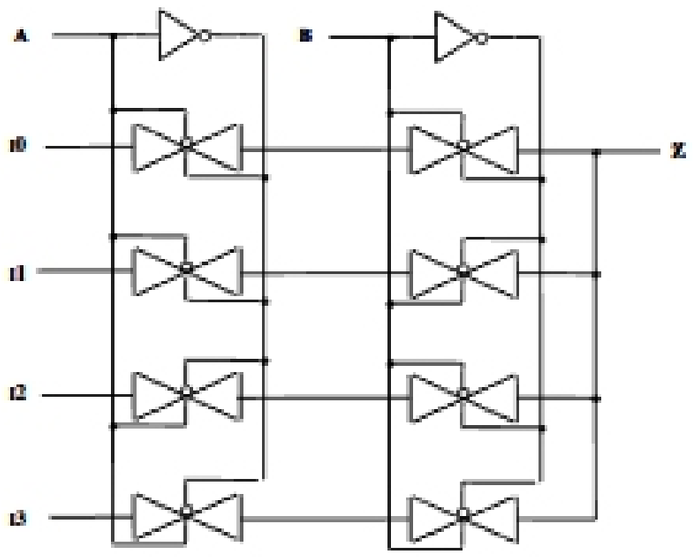
Review: CMOS transmission gate (TG)



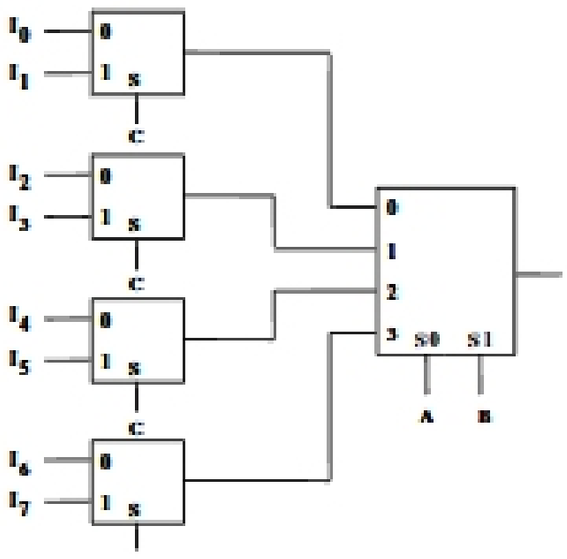
MUX



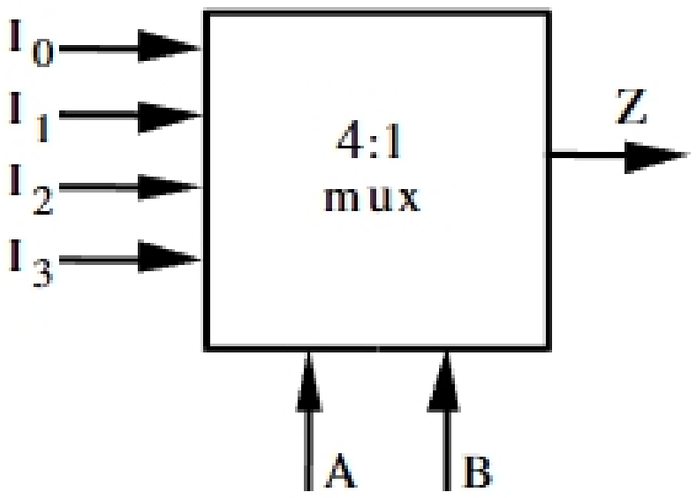
MUX using TGs



Alternative hierarchical MUX implementation



MUX examples

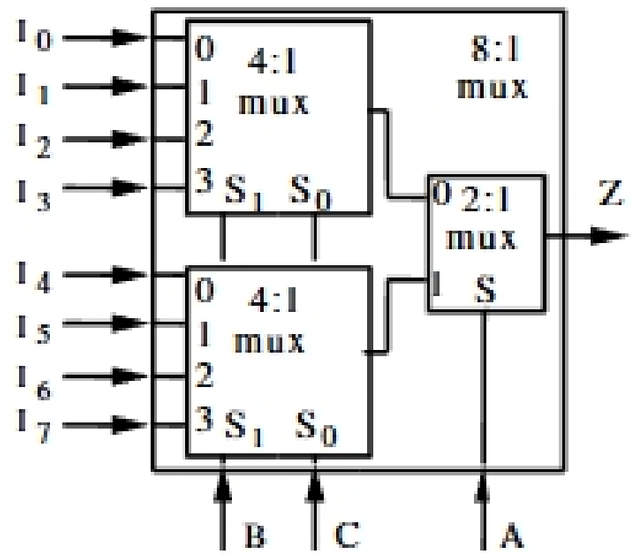


$$Z = \overline{A}\,\overline{B}\,I_0 + \overline{A}\,BI_1 + A\overline{B}\,I_2 + ABI_3$$

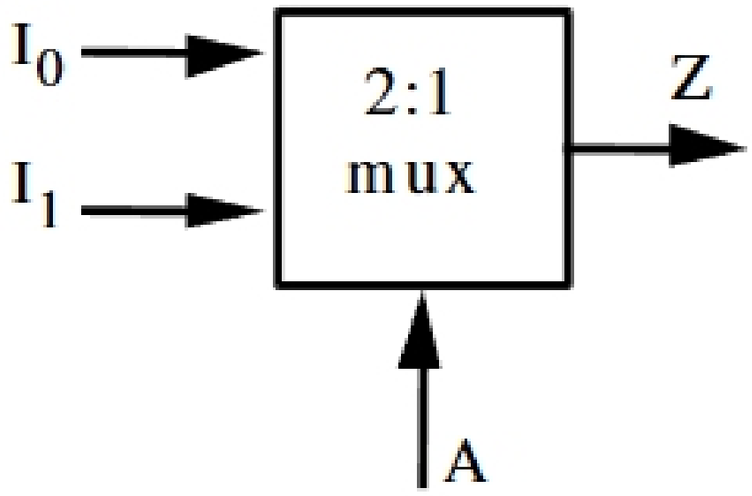
MUX properties

- A $2^n : 1$ MUX can implement any function of n variables
- A $2^{n-1} : 1$ can also be used
  - Use remaining variable as an input to the MUX

Hierarchical MUX implementation

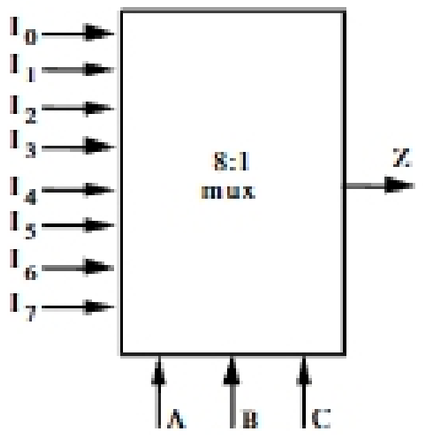


MUX examples



$$Z = \overline{A}I_0 + AI_1$$

MUX examples



$$Z = \overline{A}\,\overline{B}\,\overline{C}\,I_0 + \overline{A}\,\overline{B}\,CI_1 + \overline{A}\,B\overline{C}\,I_2 + \overline{A}\,BCI_3 + A\overline{B}\,\overline{C}\,I_4 + A\overline{B}\,CI_5 + AB\overline{C}\,I_6 + ABCI_7$$

MUX example

$$F(A,B,C) = \sum(0,2,6,7)$$
$$= \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\overline{C} + AB\overline{C} + ABC$$