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

Outline

Grading scheme

- 15% homeworks
- 35% labs
- 20% midterm exam
- 30% final exam

Planned schedule

- Mondays: Labs assigned and collected
- Wednesdays: Homeworks collected and assigned
- Friday's class will normally focus on the lab and homework of the week, and will be given by Neal Oza

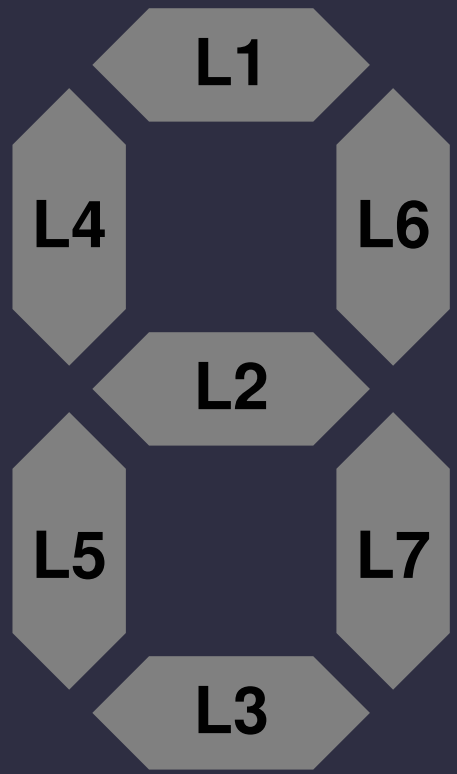
Outline

Review

- What is a truth table?
- Combinational vs. sequential logic?
- Symbol and notation for AND, OR, NOT?
- Other gates also exist, e.g., NAND, NOR, XOR, XNOR

Case study of simple combinational logic design
Seven-segment display

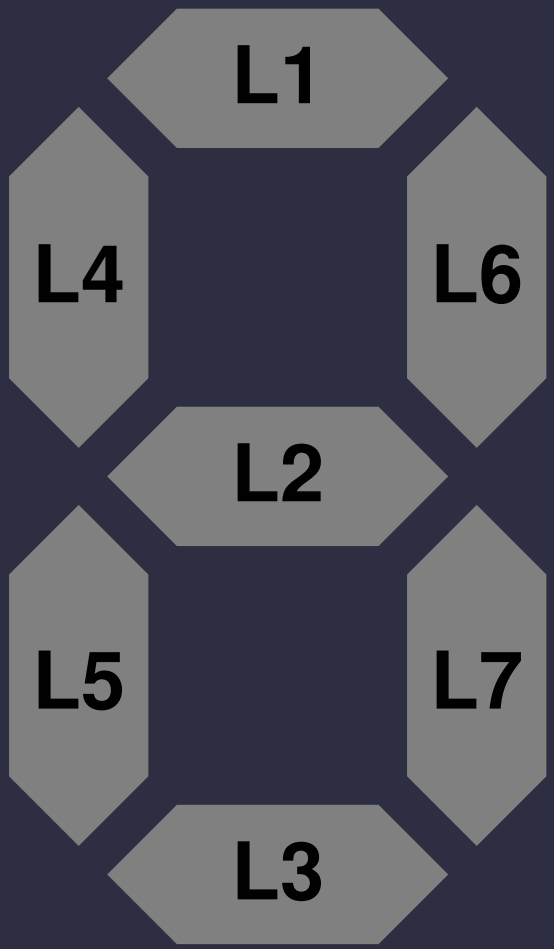
- Given: A four-bit binary input
- Display a decimal digit ranging from zero to nine
- Use a seven-segment display



Case study – Seven-segment

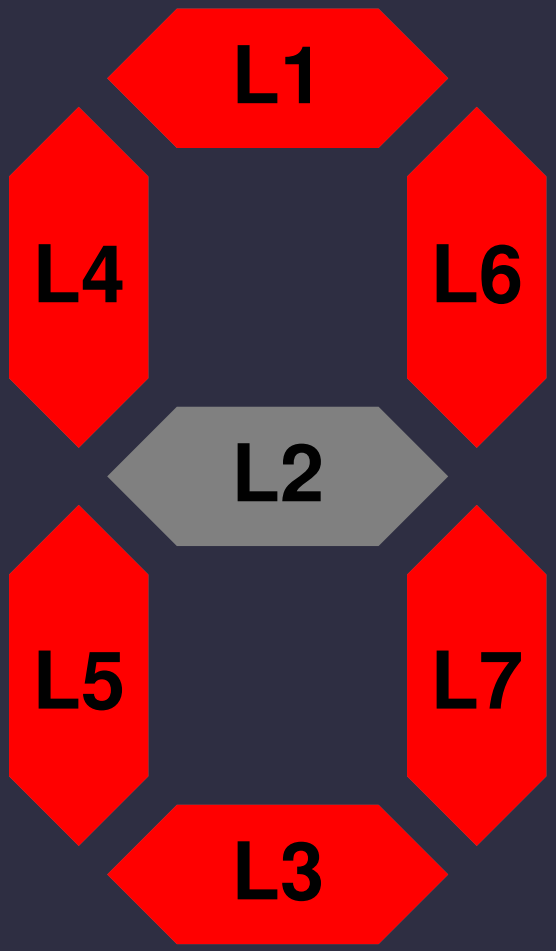
i_3	i_2	i_1	i_0	dec
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9

Case study – Seven-segment



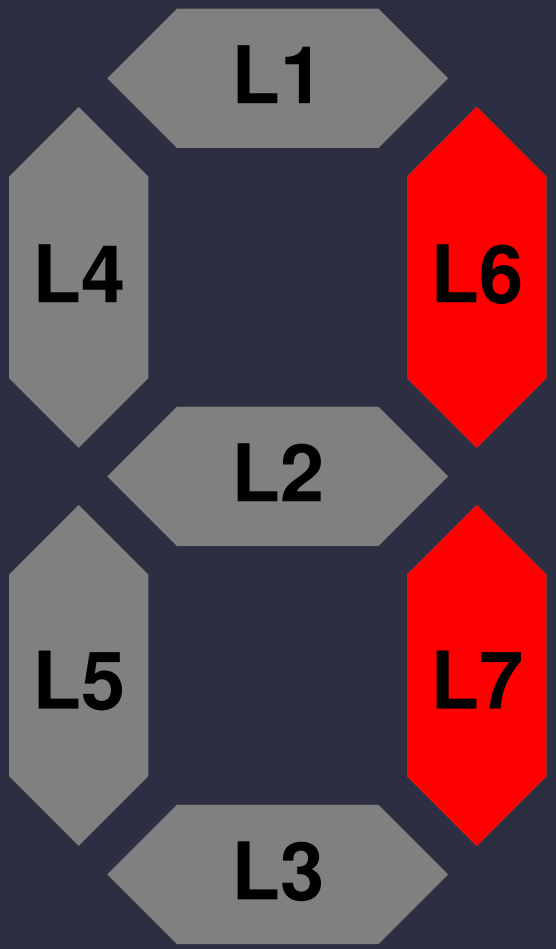
i_3	i_2	i_1	i_0	dec	L1	L2	L3
0	0	0	0	0	1	0	1
0	0	0	1	1	0	0	0
0	0	1	0	2	1	1	1
0	0	1	1	3	1	1	1
0	1	0	0	4	0	1	0
0	1	0	1	5	1	1	1
0	1	1	0	6	1	1	1
0	1	1	1	7	1	0	0
1	0	0	0	8	1	1	1
1	0	0	1	9	1	1	0

Case study – Seven-segment



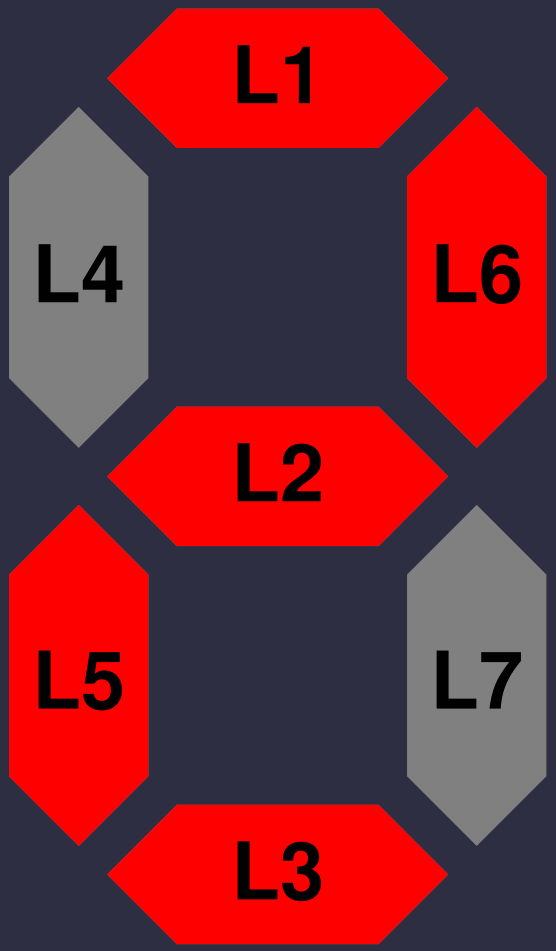
i_3	i_2	i_1	i_0	dec	L1	L2	L3
0	0	0	0	0	1	0	1
0	0	0	1	1	0	0	0
0	0	1	0	2	1	1	1
0	0	1	1	3	1	1	1
0	1	0	0	4	0	1	0
0	1	0	1	5	1	1	1
0	1	1	0	6	1	1	1
0	1	1	1	7	1	0	0
1	0	0	0	8	1	1	1
1	0	0	1	9	1	1	0

Case study – Seven-segment



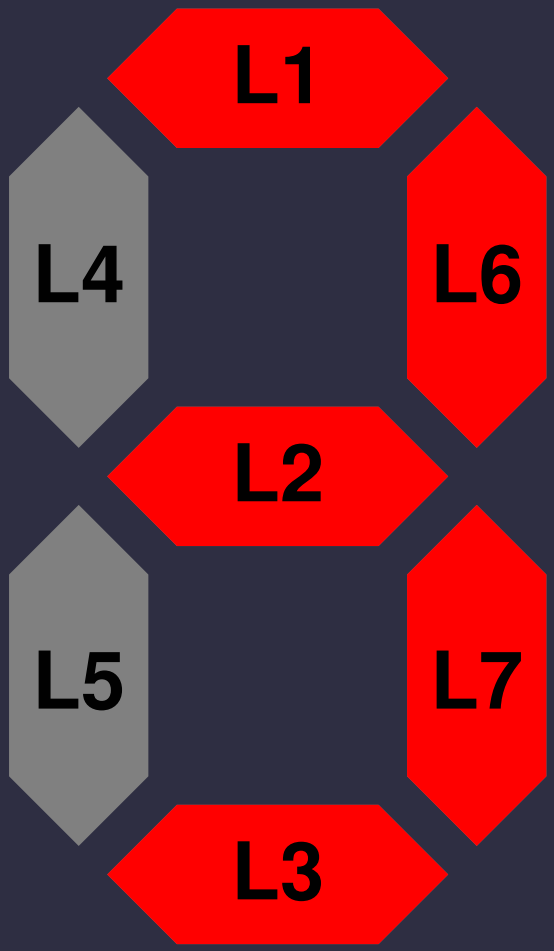
i_3	i_2	i_1	i_0	dec	L1	L2	L3
0	0	0	0	0	1	0	1
0	0	0	1	1	0	0	0
0	0	1	0	2	1	1	1
0	0	1	1	3	1	1	1
0	1	0	0	4	0	1	0
0	1	0	1	5	1	1	1
0	1	1	0	6	1	1	1
0	1	1	1	7	1	0	0
1	0	0	0	8	1	1	1
1	0	0	1	9	1	1	0

Case study – Seven-segment



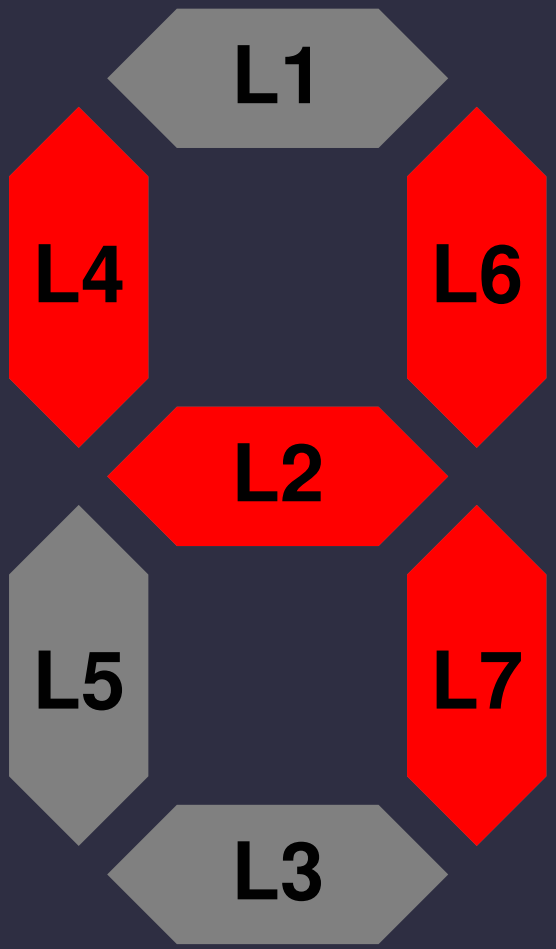
i_3	i_2	i_1	i_0	dec	L1	L2	L3
0	0	0	0	0	1	0	1
0	0	0	1	1	0	0	0
0	0	1	0	2	1	1	1
0	0	1	1	3	1	1	1
0	1	0	0	4	0	1	0
0	1	0	1	5	1	1	1
0	1	1	0	6	1	1	1
0	1	1	1	7	1	0	0
1	0	0	0	8	1	1	1
1	0	0	1	9	1	1	0

Case study – Seven-segment



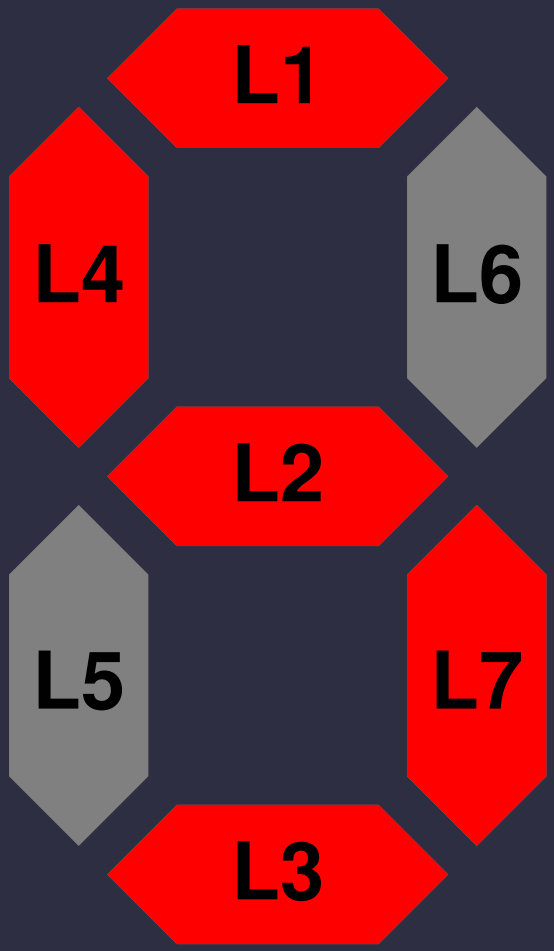
i_3	i_2	i_1	i_0	dec	L1	L2	L3
0	0	0	0	0	1	0	1
0	0	0	1	1	0	0	0
0	0	1	0	2	1	1	1
0	0	1	1	3	1	1	1
0	1	0	0	4	0	1	0
0	1	0	1	5	1	1	1
0	1	1	0	6	1	1	1
0	1	1	1	7	1	0	0
1	0	0	0	8	1	1	1
1	0	0	1	9	1	1	0

Case study – Seven-segment



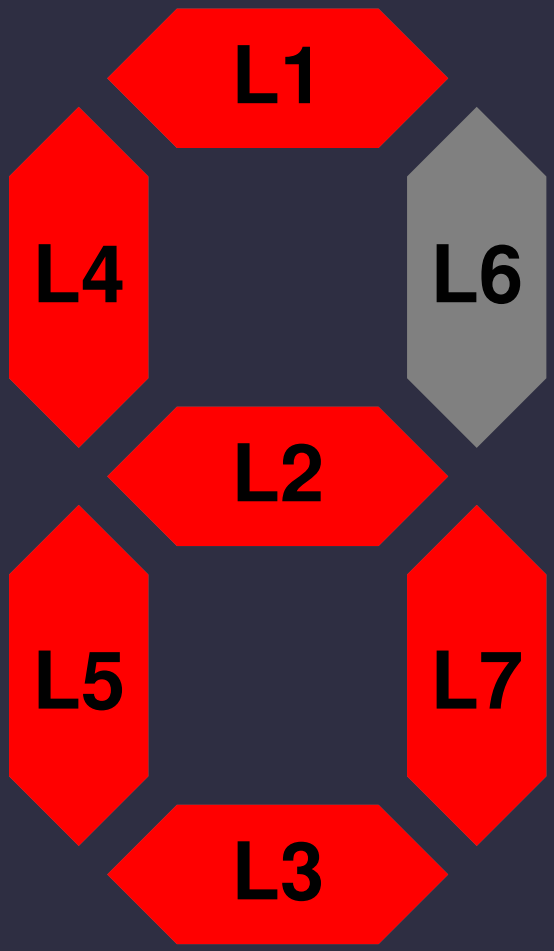
i_3	i_2	i_1	i_0	dec	L1	L2	L3
0	0	0	0	0	1	0	1
0	0	0	1	1	0	0	0
0	0	1	0	2	1	1	1
0	0	1	1	3	1	1	1
0	1	0	0	4	0	1	0
0	1	0	1	5	1	1	1
0	1	1	0	6	1	1	1
0	1	1	1	7	1	0	0
1	0	0	0	8	1	1	1
1	0	0	1	9	1	1	0

Case study – Seven-segment



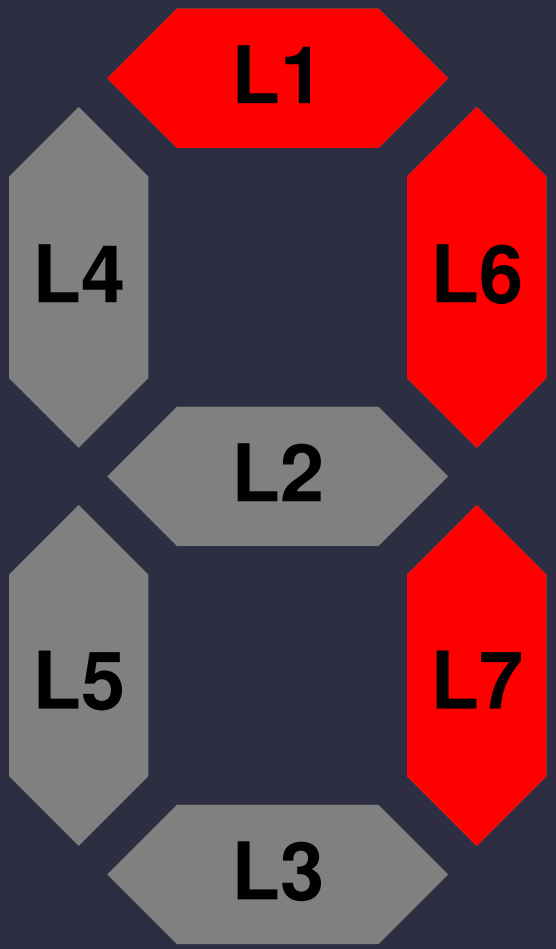
i_3	i_2	i_1	i_0	dec	L1	L2	L3
0	0	0	0	0	1	0	1
0	0	0	1	1	0	0	0
0	0	1	0	2	1	1	1
0	0	1	1	3	1	1	1
0	1	0	0	4	0	1	0
0	1	0	1	5	1	1	1
0	1	1	0	6	1	1	1
0	1	1	1	7	1	0	0
1	0	0	0	8	1	1	1
1	0	0	1	9	1	1	0

Case study – Seven-segment



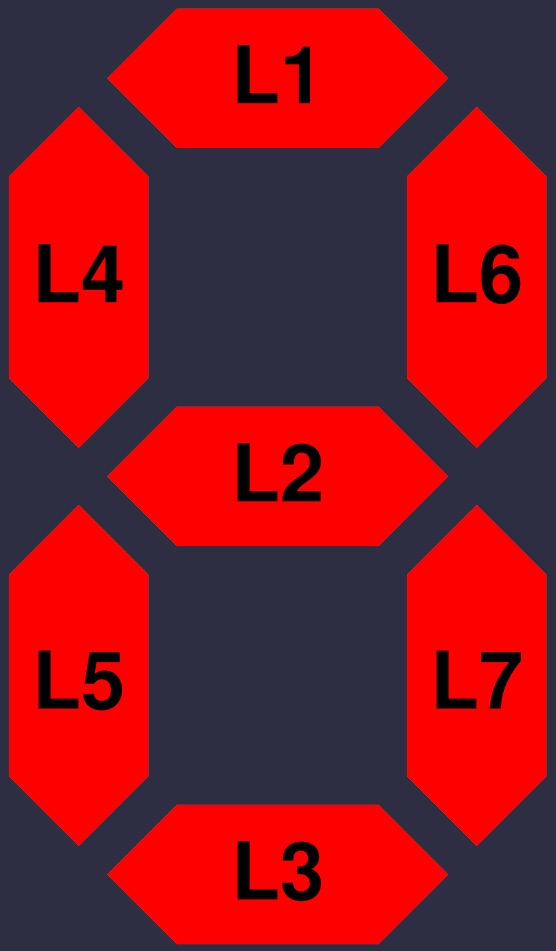
i_3	i_2	i_1	i_0	dec	L1	L2	L3
0	0	0	0	0	1	0	1
0	0	0	1	1	0	0	0
0	0	1	0	2	1	1	1
0	0	1	1	3	1	1	1
0	1	0	0	4	0	1	0
0	1	0	1	5	1	1	1
0	1	1	0	6	1	1	1
0	1	1	1	7	1	0	0
1	0	0	0	8	1	1	1
1	0	0	1	9	1	1	0

Case study – Seven-segment



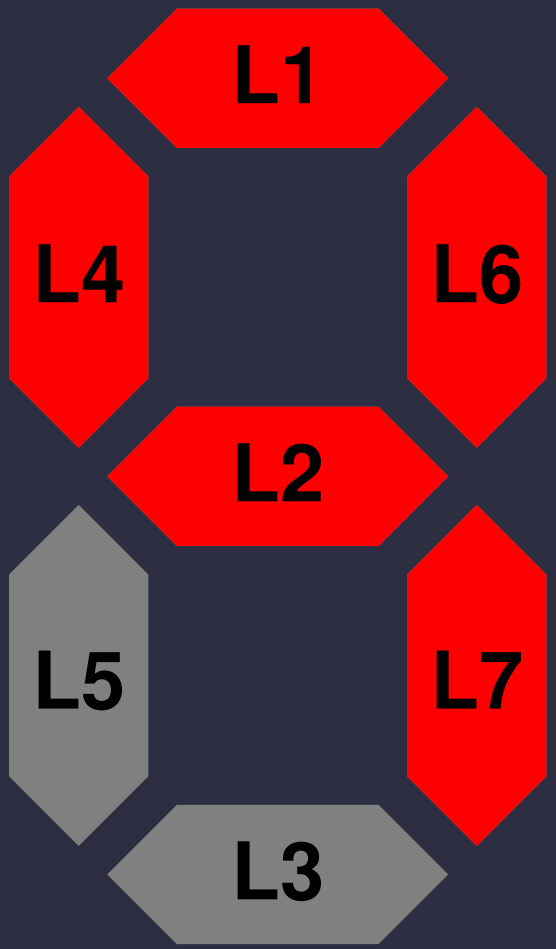
i_3	i_2	i_1	i_0	dec	L1	L2	L3
0	0	0	0	0	1	0	1
0	0	0	1	1	0	0	0
0	0	1	0	2	1	1	1
0	0	1	1	3	1	1	1
0	1	0	0	4	0	1	0
0	1	0	1	5	1	1	1
0	1	1	0	6	1	1	1
0	1	1	1	7	1	0	0
1	0	0	0	8	1	1	1
1	0	0	1	9	1	1	0

Case study – Seven-segment



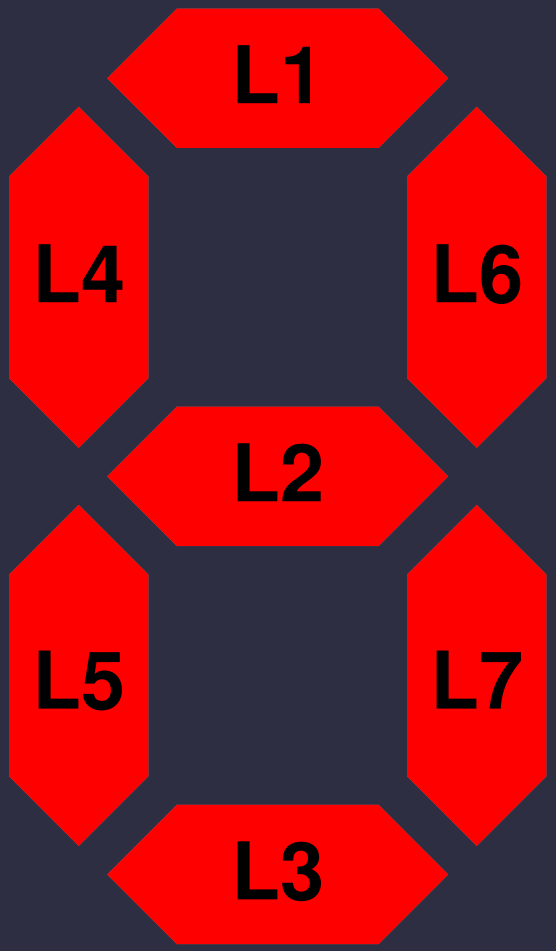
i_3	i_2	i_1	i_0	dec	L1	L2	L3
0	0	0	0	0	1	0	1
0	0	0	1	1	0	0	0
0	0	1	0	2	1	1	1
0	0	1	1	3	1	1	1
0	1	0	0	4	0	1	0
0	1	0	1	5	1	1	1
0	1	1	0	6	1	1	1
0	1	1	1	7	1	0	0
1	0	0	0	8	1	1	1
1	0	0	1	9	1	1	0

Case study – Seven-segment



i_3	i_2	i_1	i_0	dec	L1	L2	L3
0	0	0	0	0	1	0	1
0	0	0	1	1	0	0	0
0	0	1	0	2	1	1	1
0	0	1	1	3	1	1	1
0	1	0	0	4	0	1	0
0	1	0	1	5	1	1	1
0	1	1	0	6	1	1	1
0	1	1	1	7	1	0	0
1	0	0	0	8	1	1	1
1	0	0	1	9	1	1	0

Case study – Seven-segment

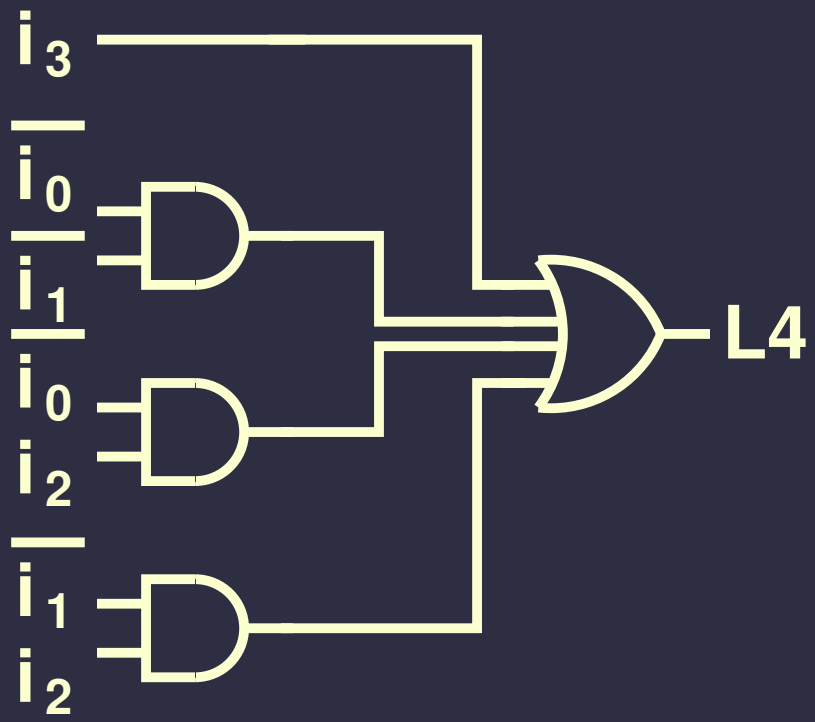


i_3	i_2	i_1	i_0	dec	L1	L2	L3
0	0	0	0	0	1	0	1
0	0	0	1	1	0	0	0
0	0	1	0	2	1	1	1
0	0	1	1	3	1	1	1
0	1	0	0	4	0	1	0
0	1	0	1	5	1	1	1
0	1	1	0	6	1	1	1
0	1	1	1	7	1	0	0
1	0	0	0	8	1	1	1
1	0	0	1	9	1	1	0

Implement L4

i_3	i_2	i_1	i_0	dec	L4
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	2	0
0	0	1	1	3	0
0	1	0	0	4	1
0	1	0	1	5	1
0	1	1	0	6	1
0	1	1	1	7	0
1	0	0	0	8	1
1	0	0	1	9	1

L4 implementation



In a future lecture, I'll explain how to do this sort of design.

Outline

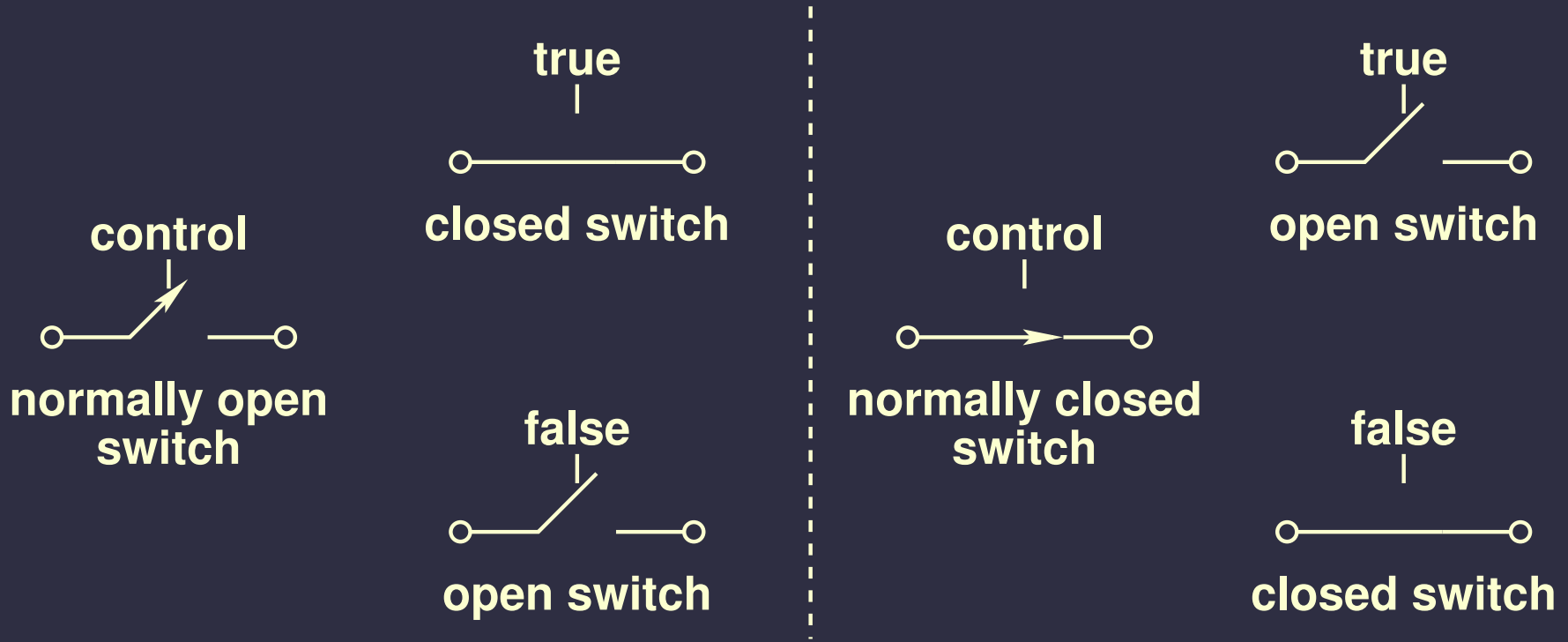
Switch-based design representation

- A switch shorts or opens two points dependant on a control signal
- Used as models for digital transistors
- Why is using normally open and normally closed particularly useful for CMOS?

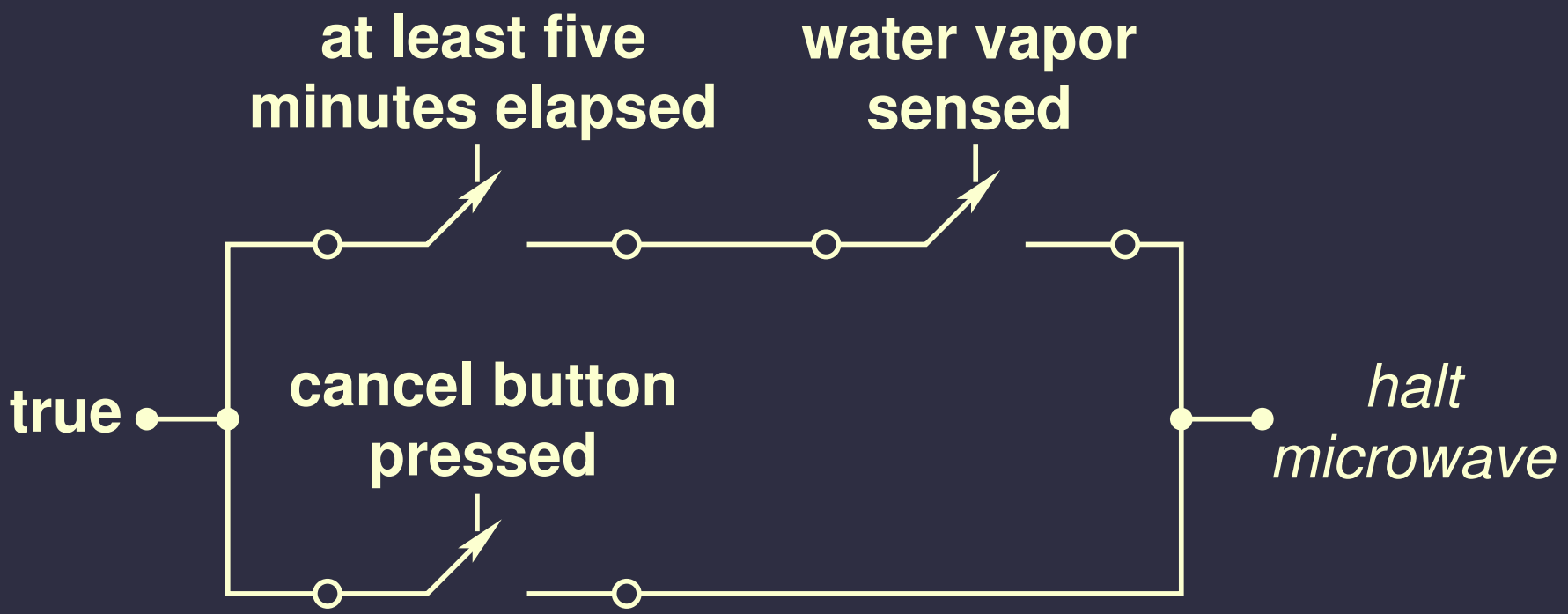
Switch-based design representation

- A switch shorts or opens two points dependant on a control signal
- Used as models for digital transistors
- Why is using normally open and normally closed particularly useful for CMOS?
  - NMOS and PMOS transistors easy to model

Switch-based definitions

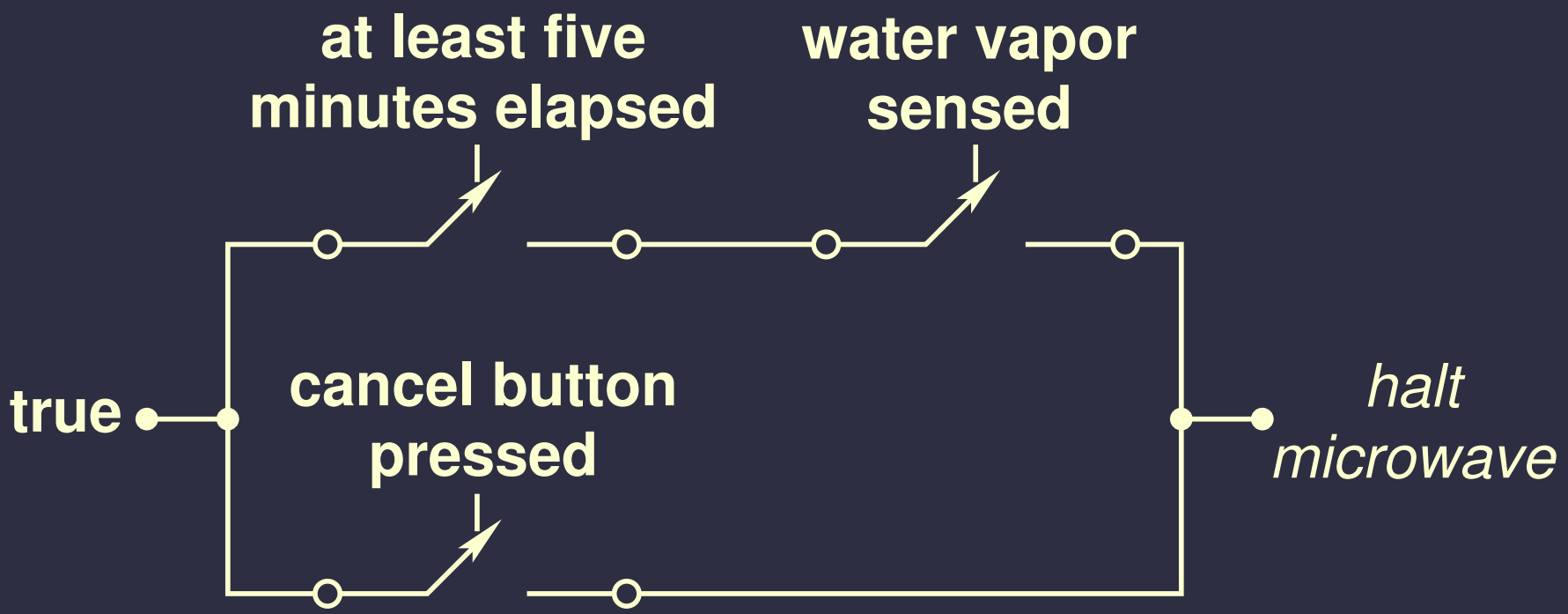


Microwave control example



- What happens if the cancel button is not pressed and five minutes haven't yet passed?

Microwave control example



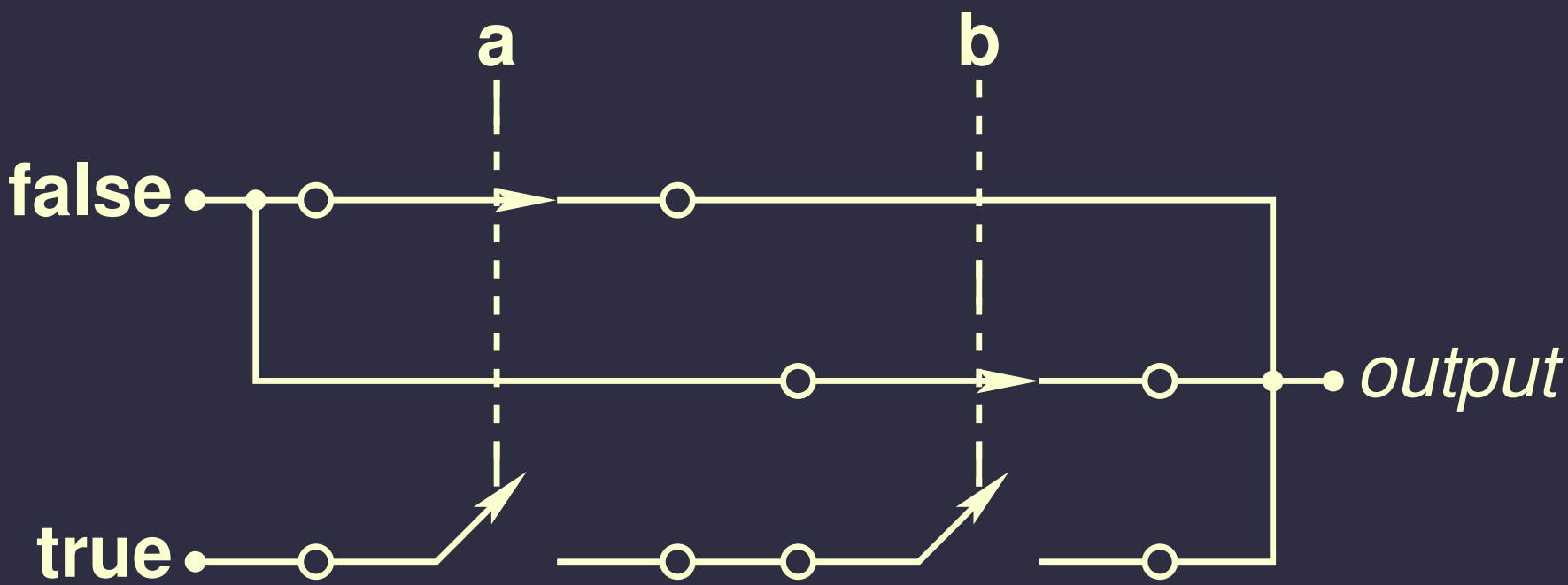
- What happens if the cancel button is not pressed and five minutes haven't yet passed?
  - The output value is undefined.

Constraints on network output

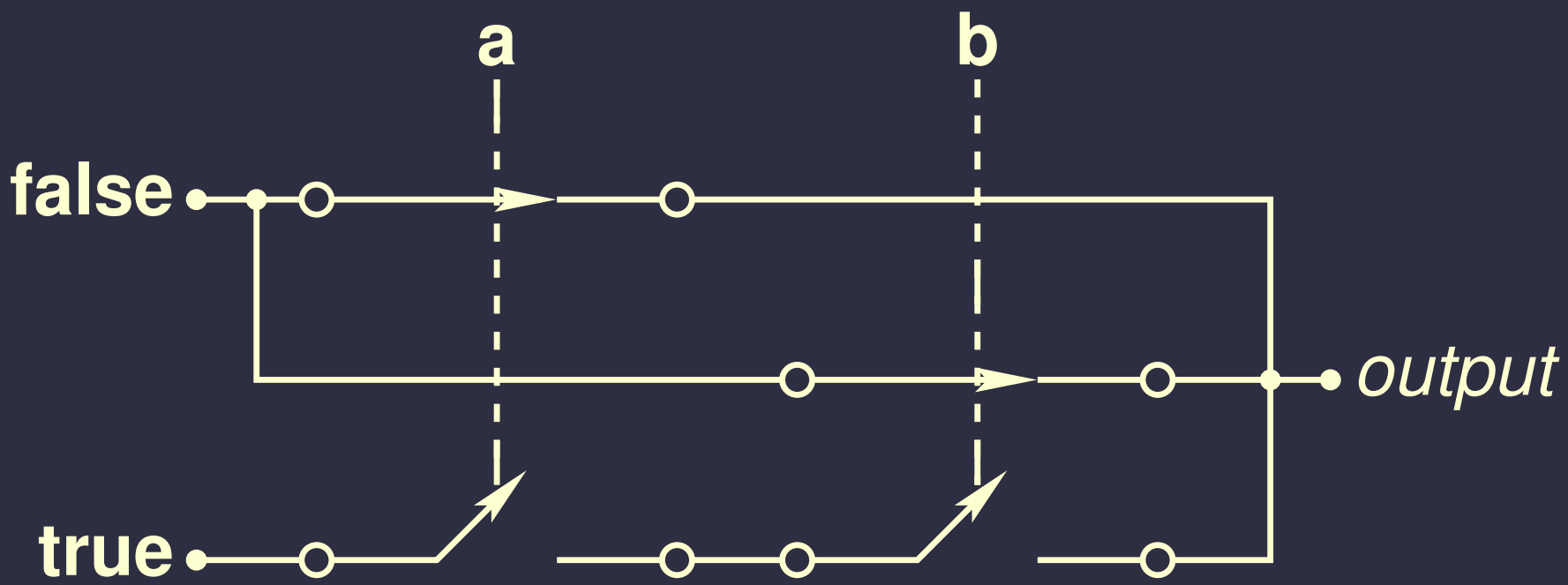
Under all possible combinations of input values

- Each output must be connected to an input value
- No output may be connected to conflicting input values

Switch-based *AND*



Switch-based *AND*



Note that this requires

- Normally closed switches that transmit false signals well
- Normally open switches that transmit true signals well

Outline

Relationship with CMOS

- Metal Oxide Semiconductor
- Positive and negative carriers
- Complimentary MOS
- PMOS gates are like normally closed switches that are good at transmitting only true (high) signals
- NMOS gates are like normally open switches that are good at transmitting only false (low) signals

Relationship with CMOS

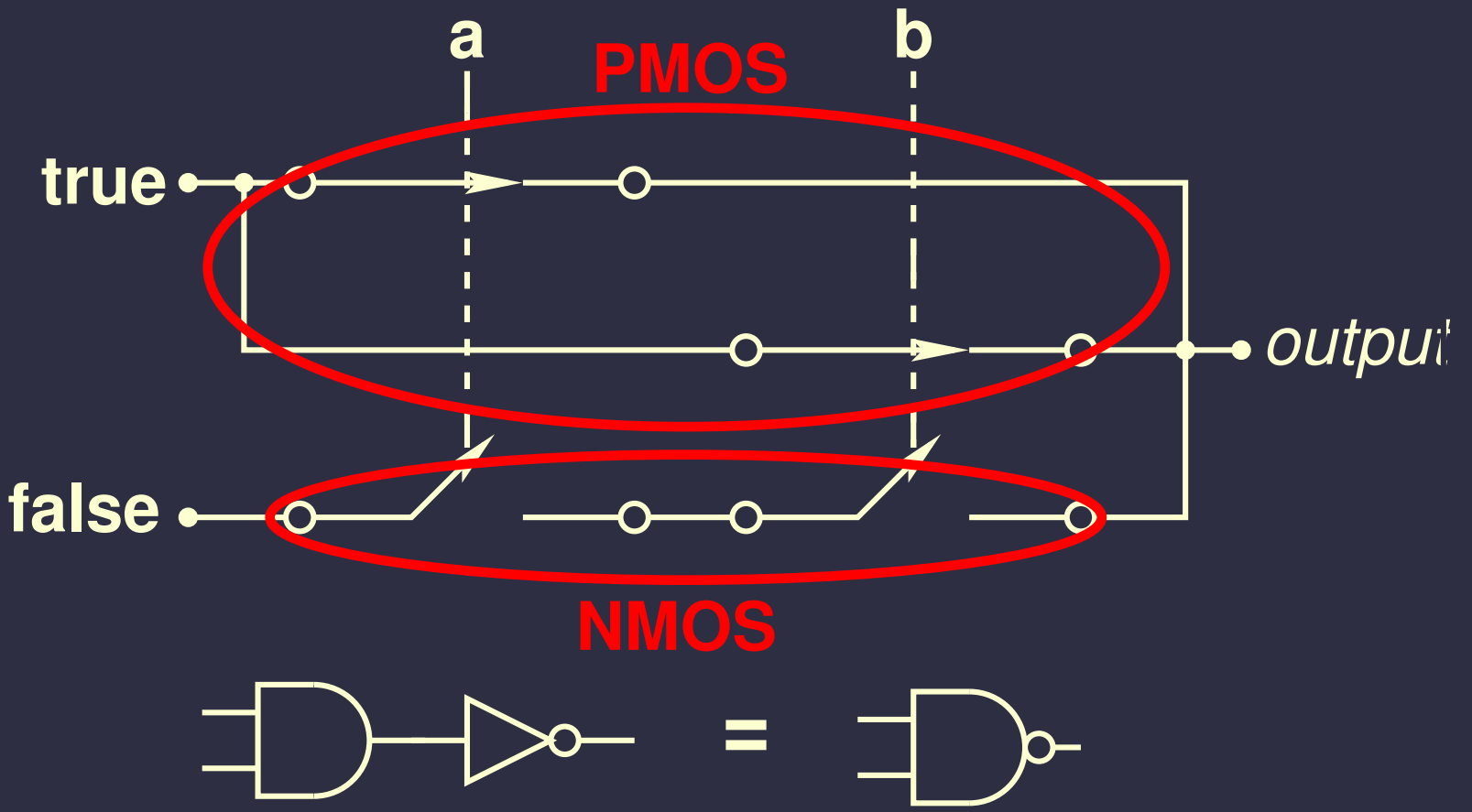
- Metal Oxide Semiconductor
- Positive and negative carriers
- Complimentary MOS
- PMOS gates are like normally closed switches that are good at transmitting only true (high) signals
- NMOS gates are like normally open switches that are good at transmitting only false (low) signals

Relationship with CMOS

- Metal Oxide Semiconductor
- Positive and negative carriers
- Complimentary MOS
- PMOS gates are like normally closed switches that are good at transmitting only true (high) signals
- NMOS gates are like normally open switches that are good at transmitting only false (low) signals

NAND gate

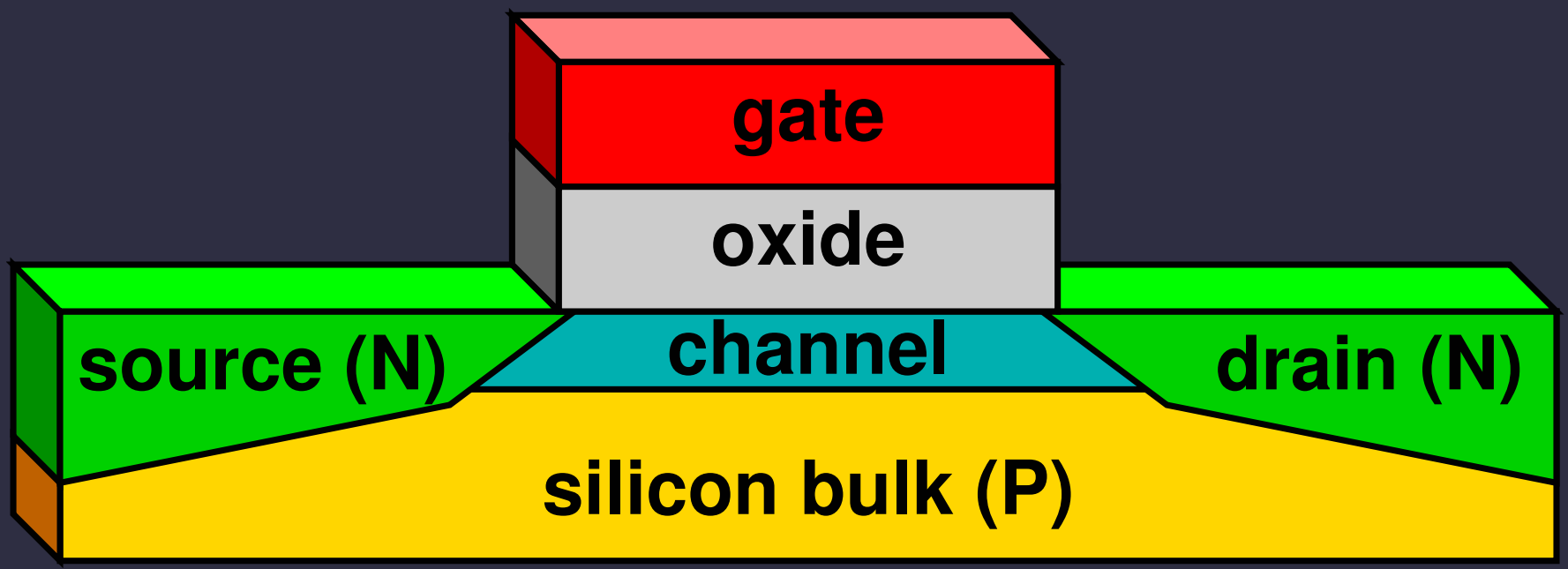
Therefore, *NAND* and *NOR* gates are used in CMOS design instead of *AND* and *OR* gates



Transistors

- Basic device in NMOS and PMOS (CMOS) technologies
- Can be used to construct any logic gate

NMOS transistor



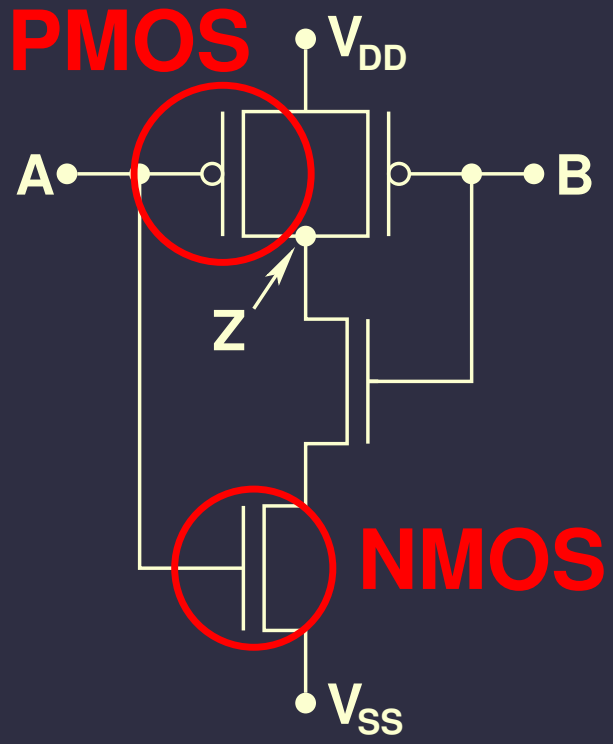
NMOS transistor

- Metal, oxide, semiconductor (MOS)
  - Then it was polysilicon, oxide, semiconductor
  - Now it is metal, hafnium-based low-k dielectric, semiconductor
- P-type bulk silicon doped with positively charged ions
- N-type diffusion regions doped with negatively charged ions
- Gate can be used to pull a few electrons near the oxide
  - Forms channel region, conduction from source to drain starts

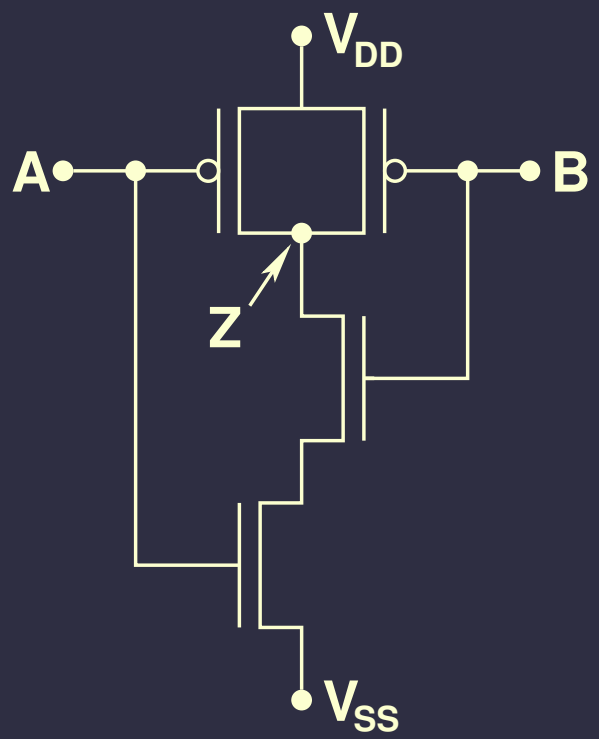
CMOS

- NMOS turns on when the gate is high
- PMOS just like NMOS, with N and P regions swapped
- PMOS turns on when the gate is low
- NMOS good at conducting low (0s)
- PMOS good at conducting high (1s)
- Use NMOS and PMOS transistors together to build circuits
  - Complementary metal oxide silicon (CMOS)

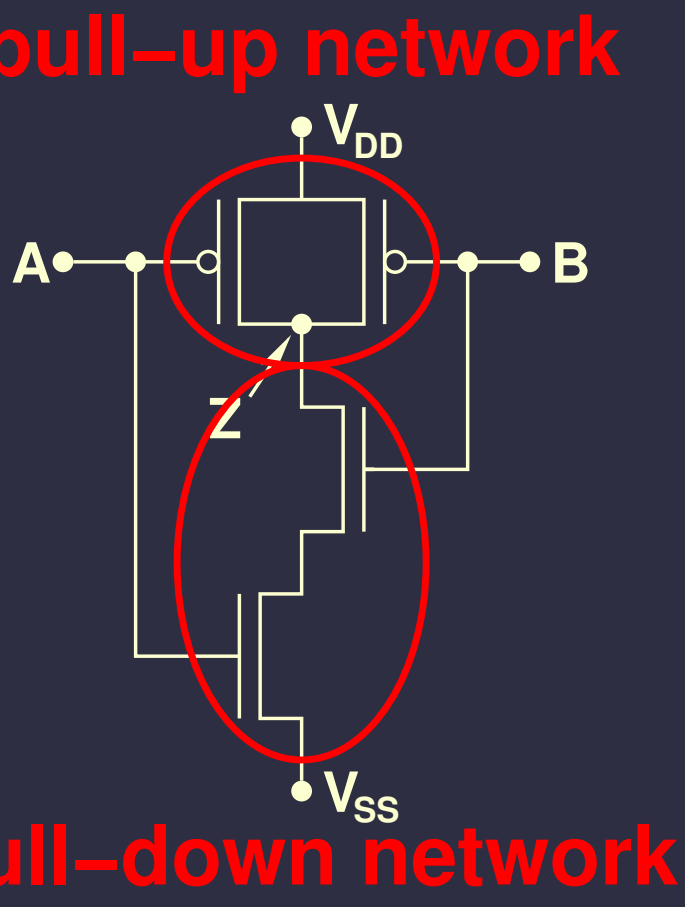
CMOS NAND gate



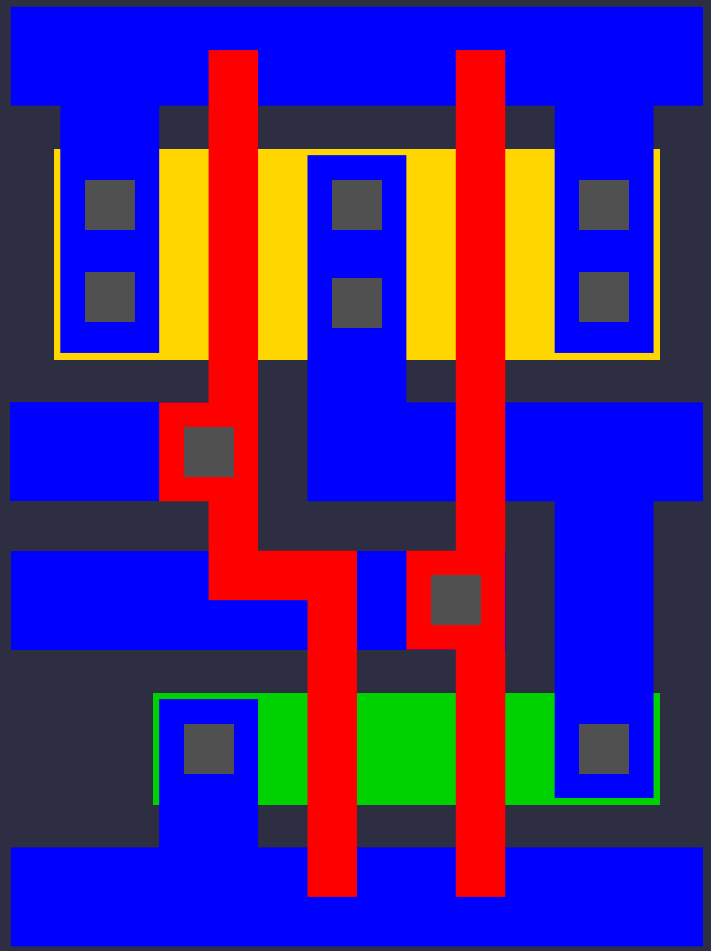
CMOS NAND gate



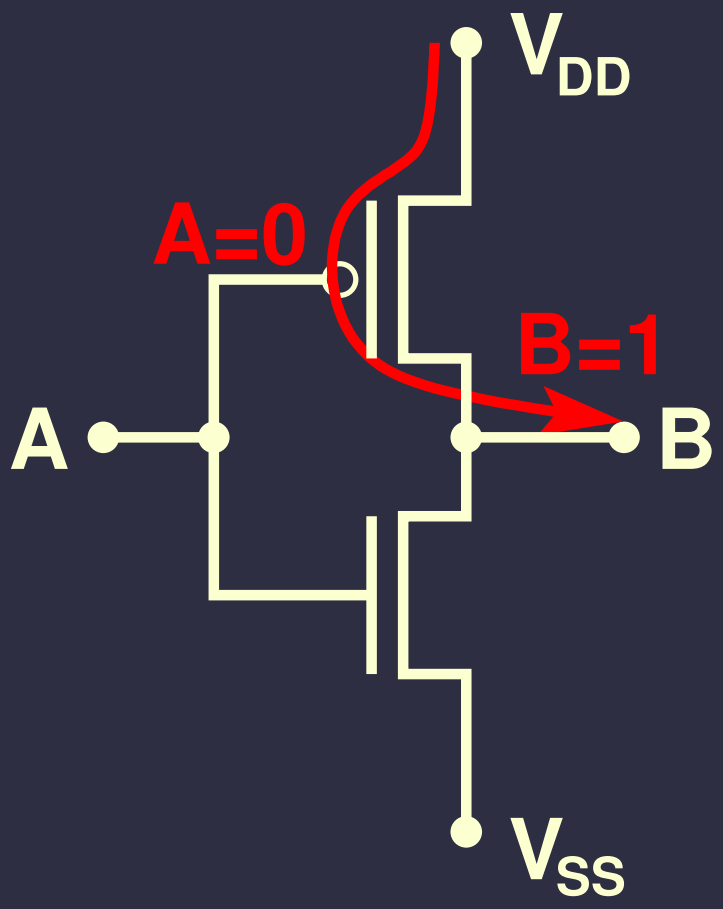
CMOS NAND gate



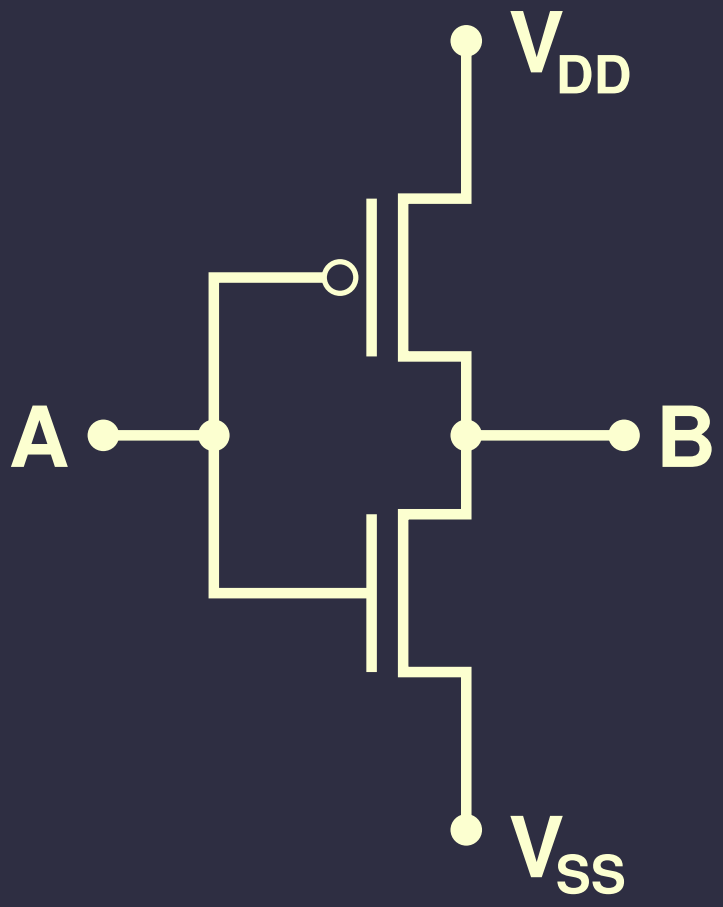
CMOS NAND gate layout



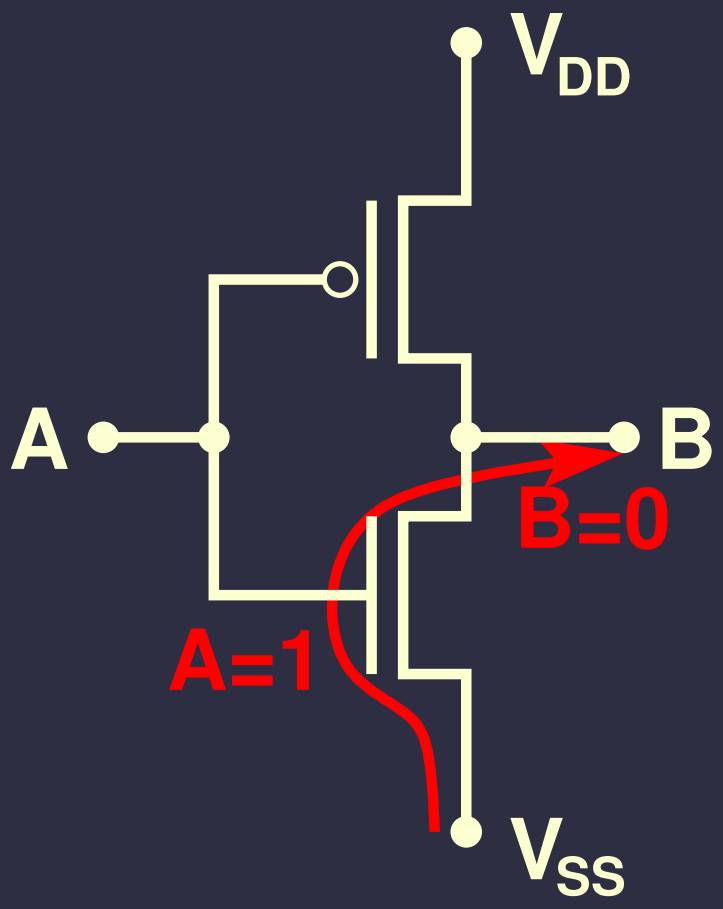
CMOS inverter operation



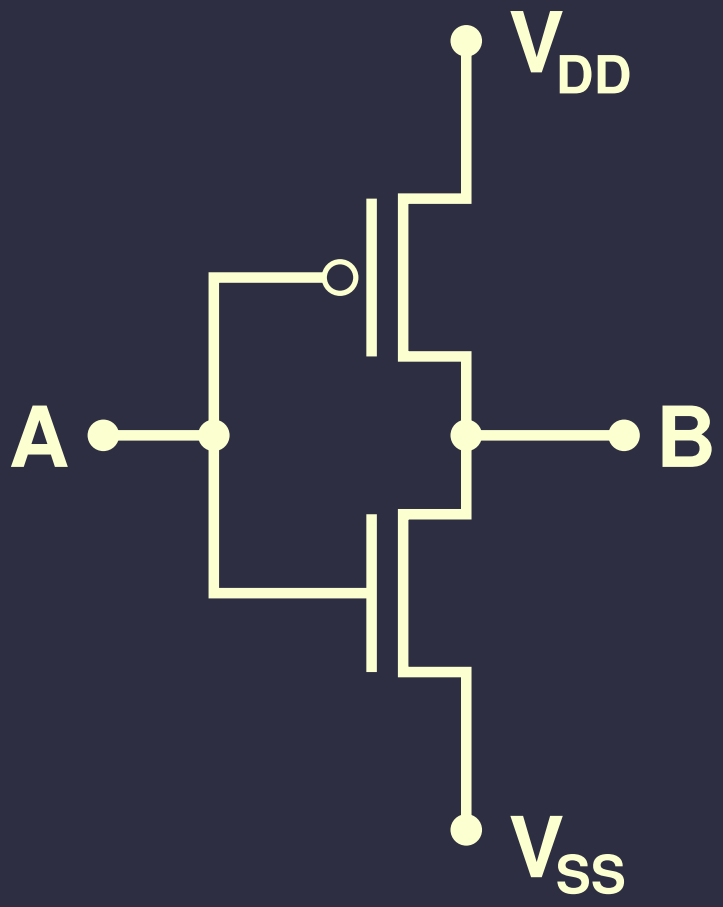
CMOS inverter operation



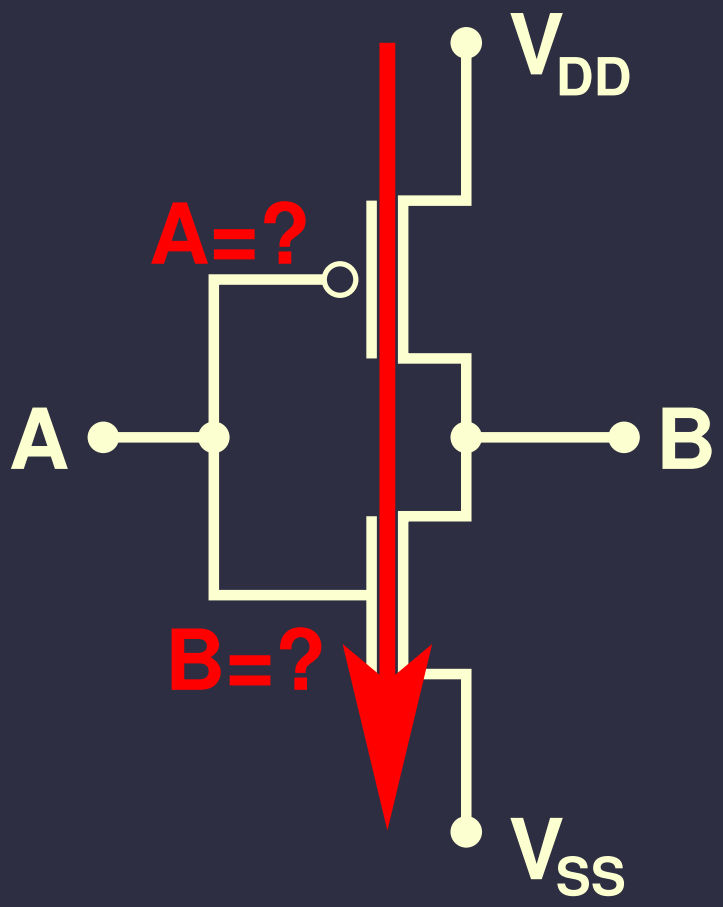
CMOS inverter operation



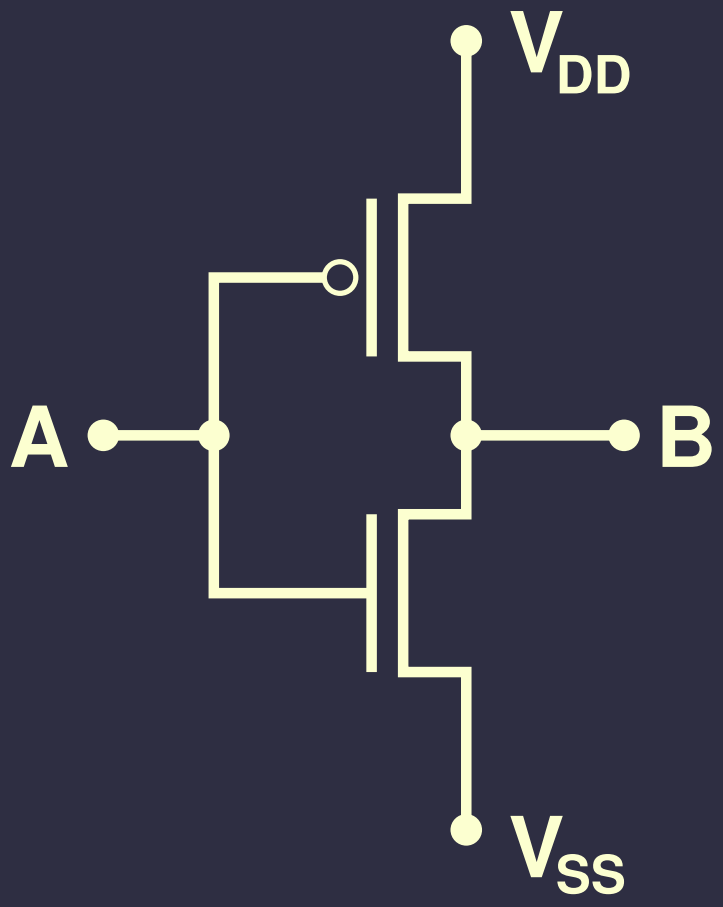
CMOS inverter operation



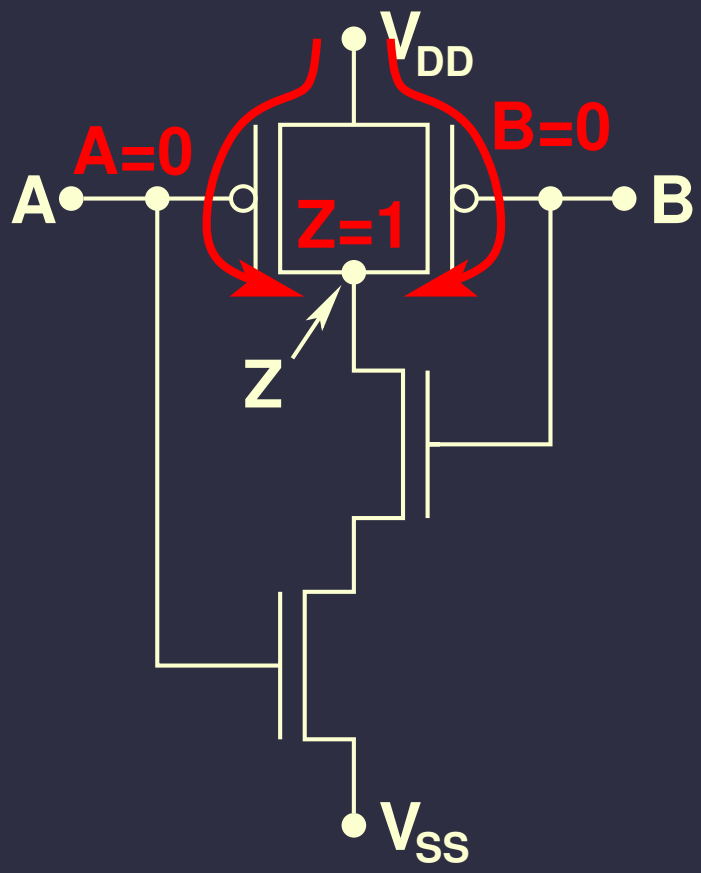
CMOS inverter operation



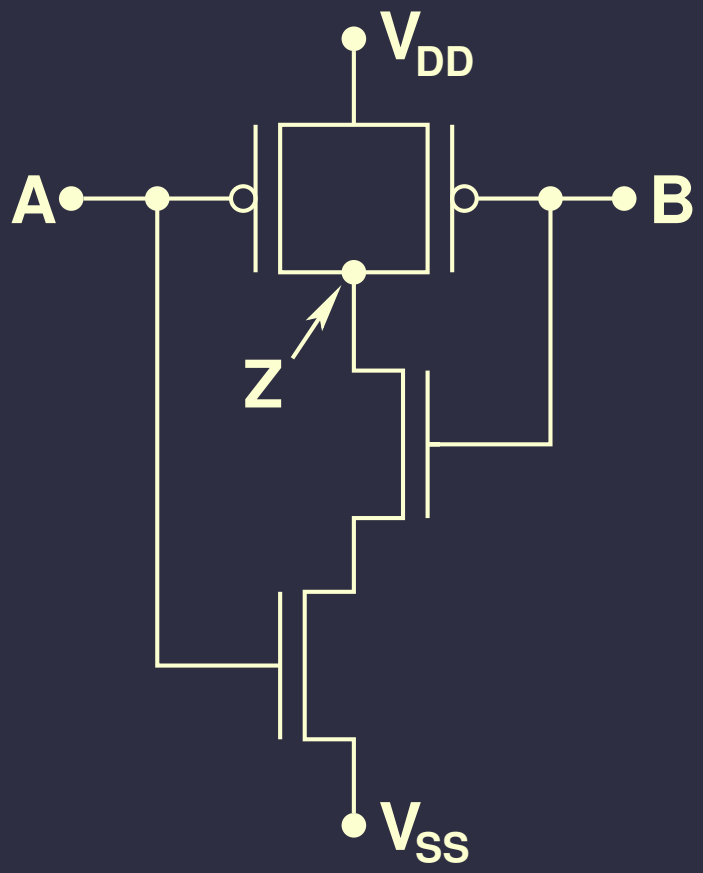
CMOS inverter operation



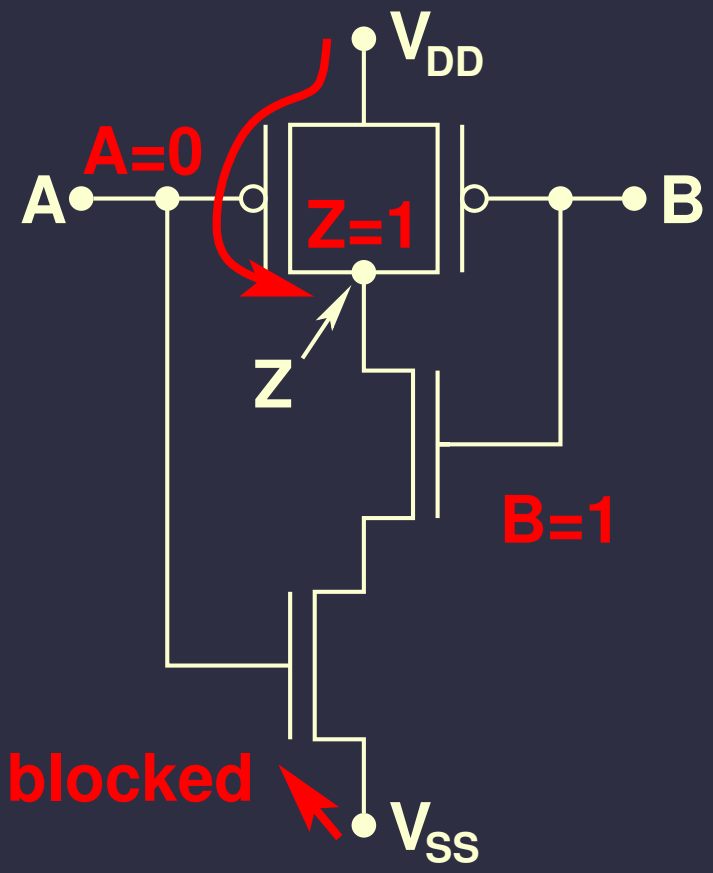
NAND operation



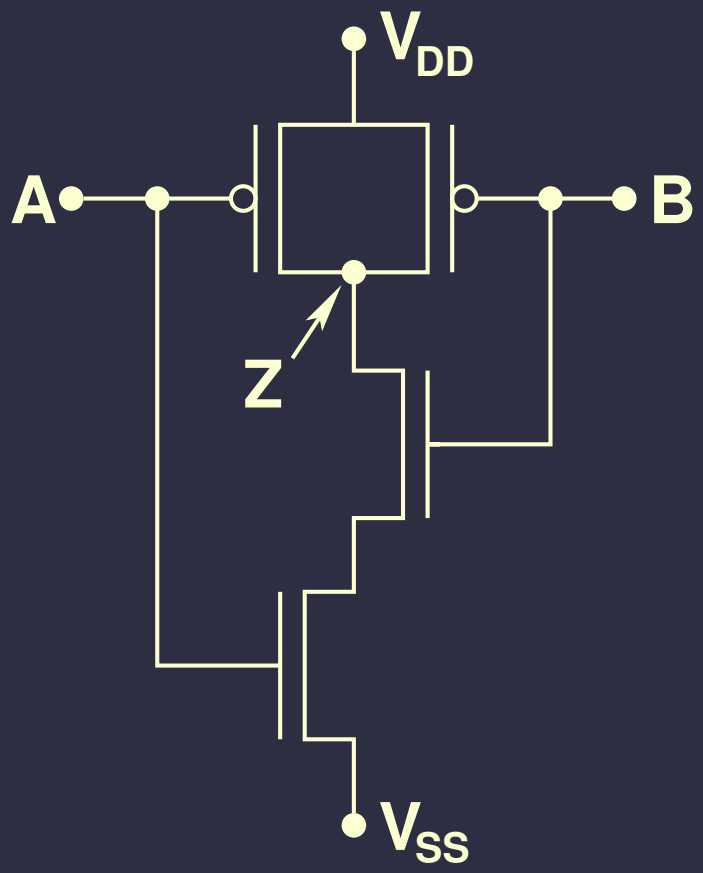
NAND operation



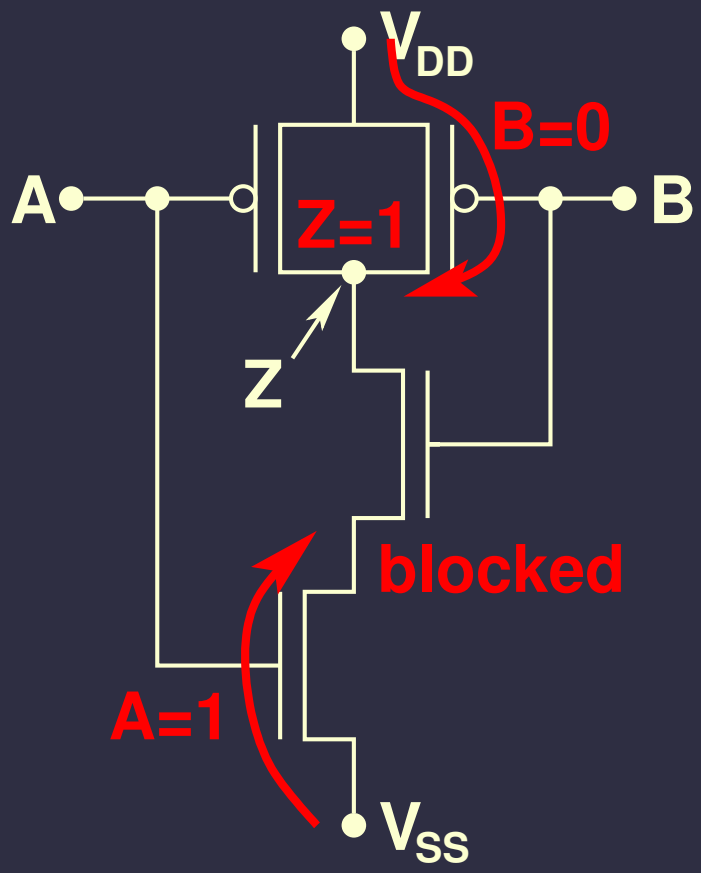
NAND operation



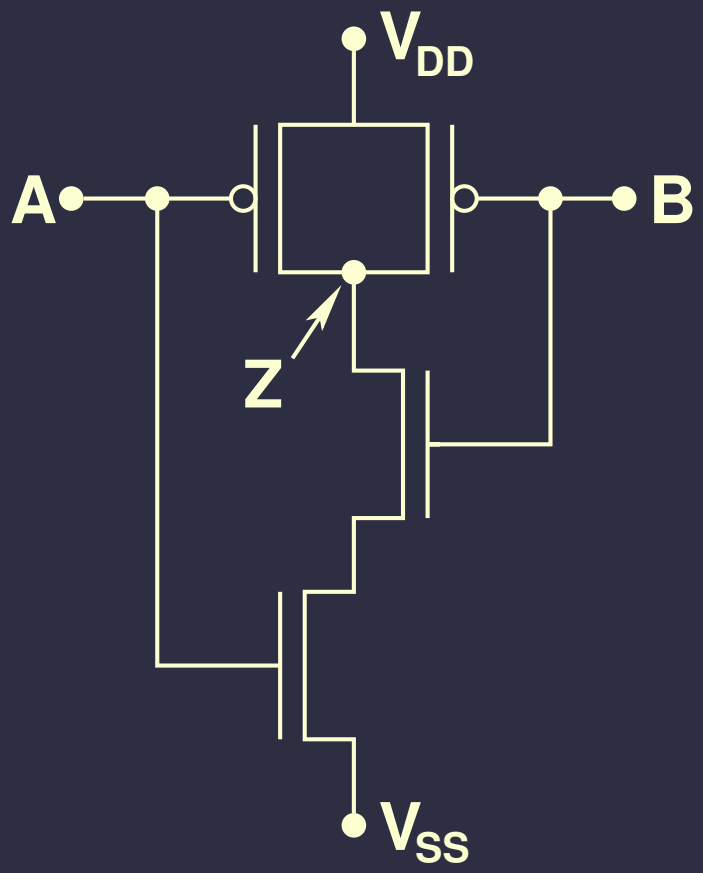
NAND operation



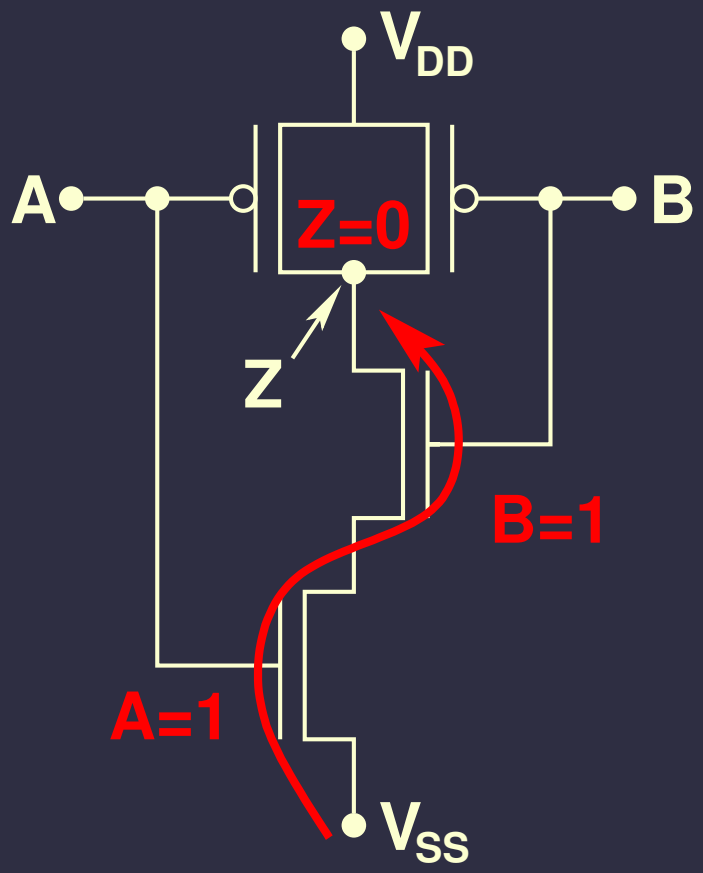
NAND operation



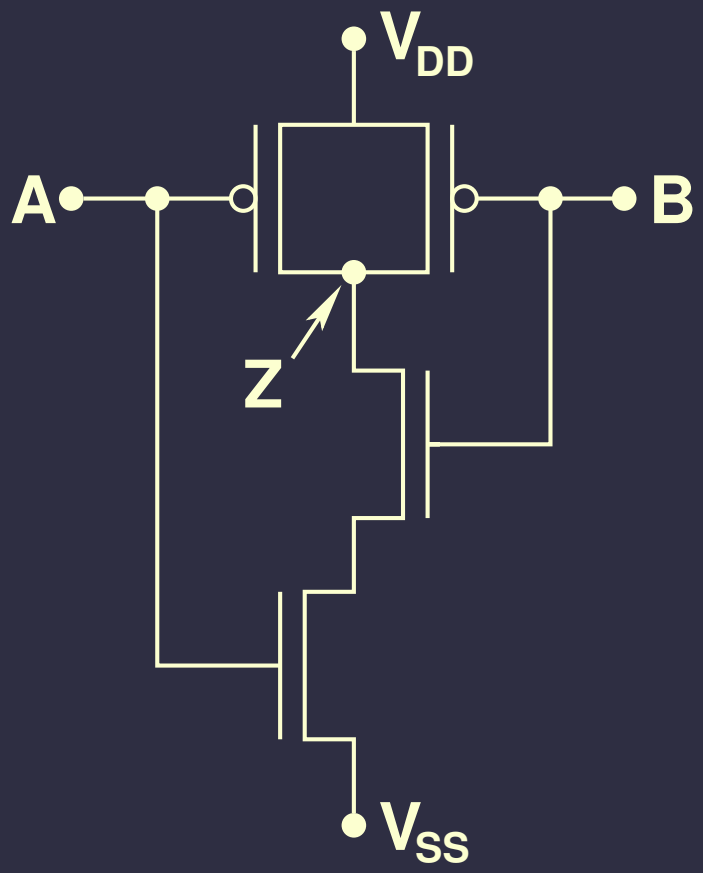
NAND operation



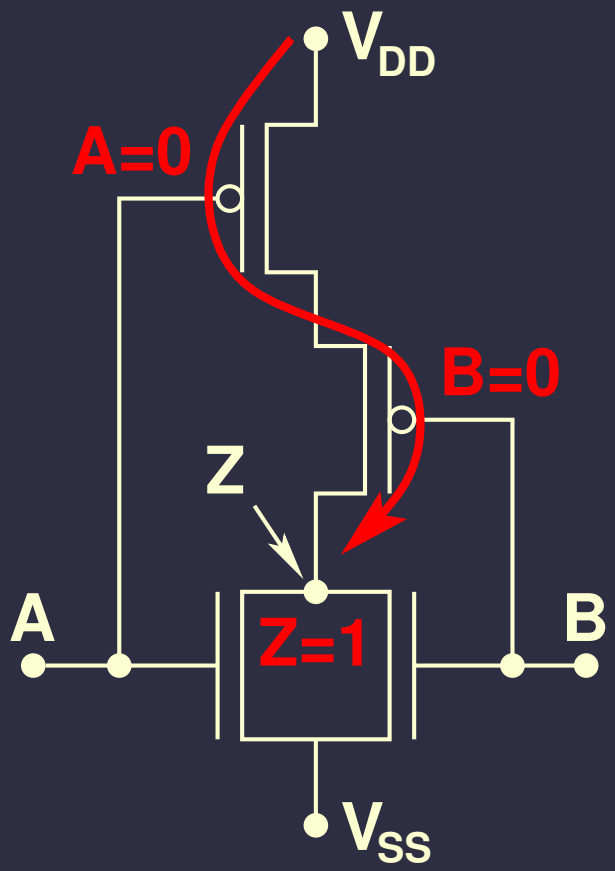
NAND operation



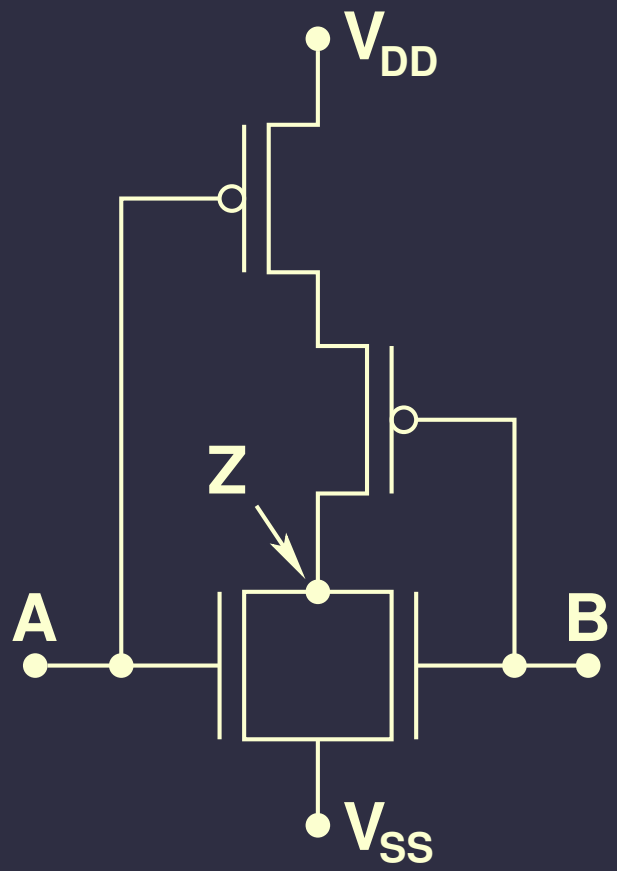
NAND operation



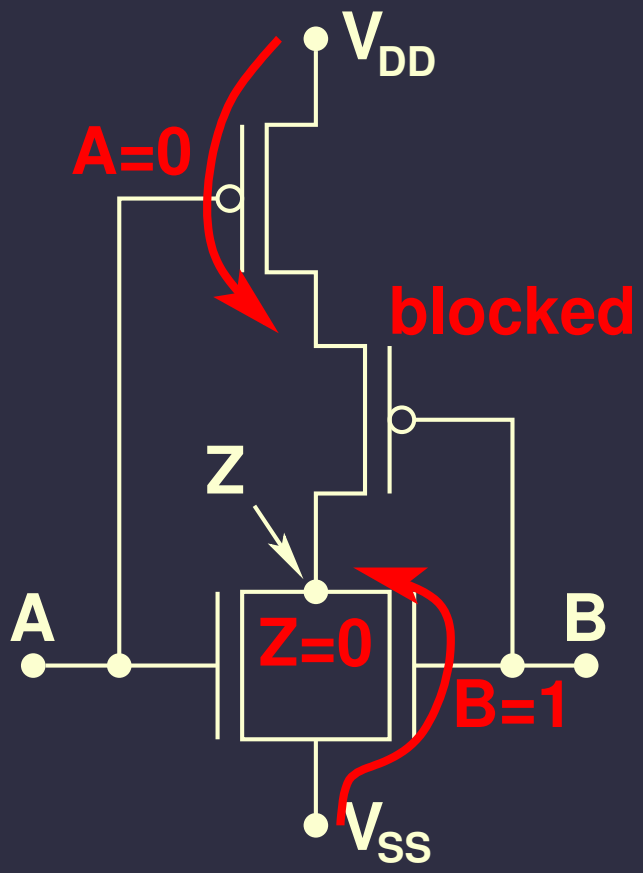
NOR operation



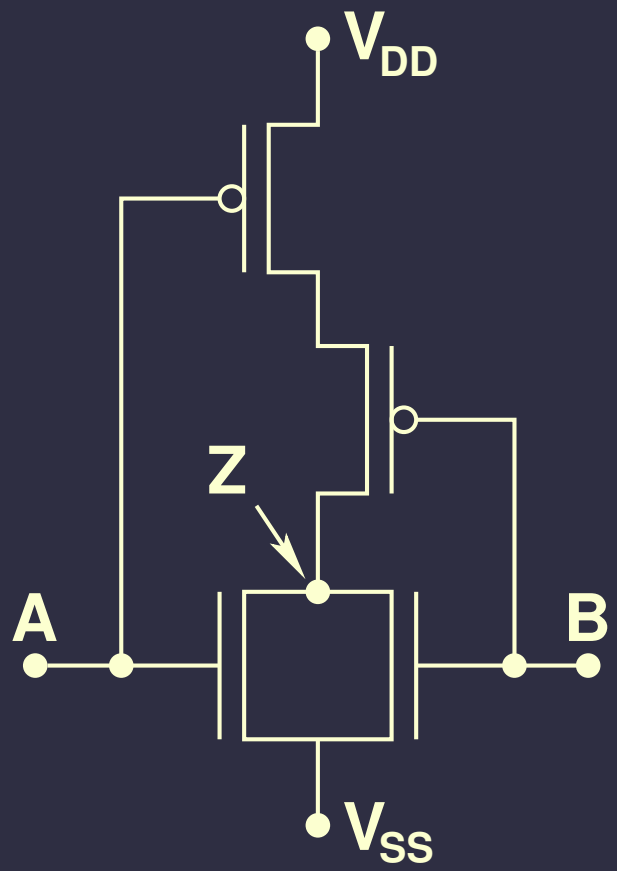
NOR operation



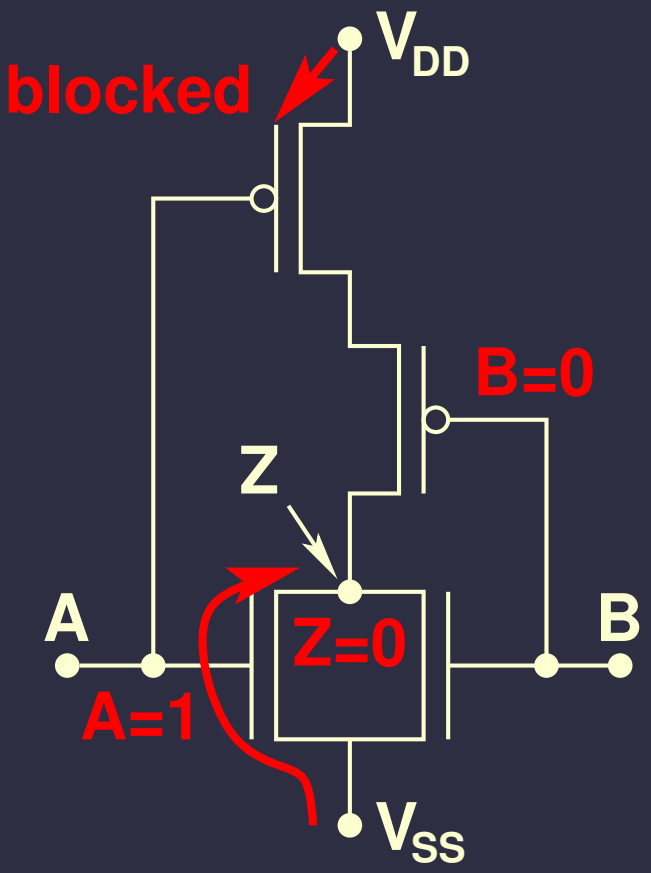
NOR operation



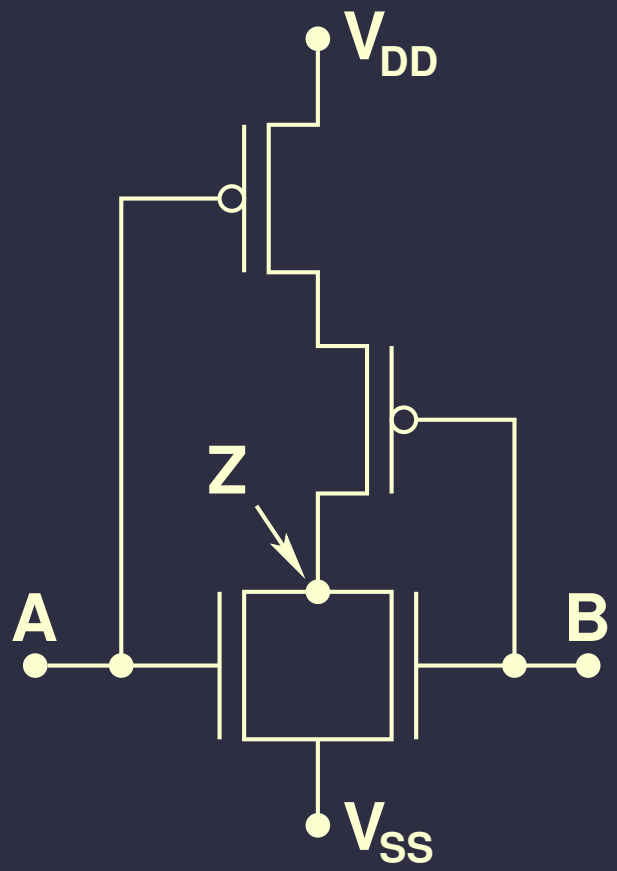
NOR operation



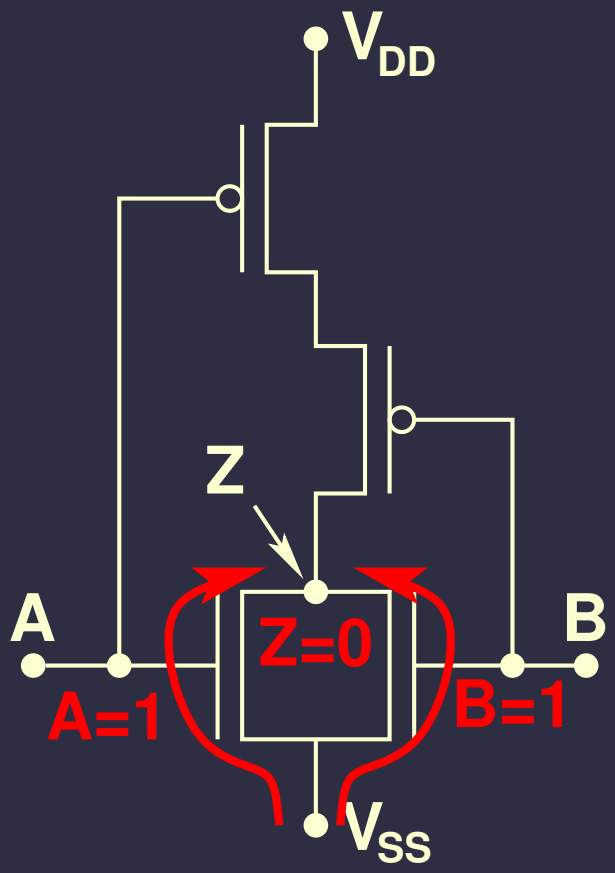
NOR operation



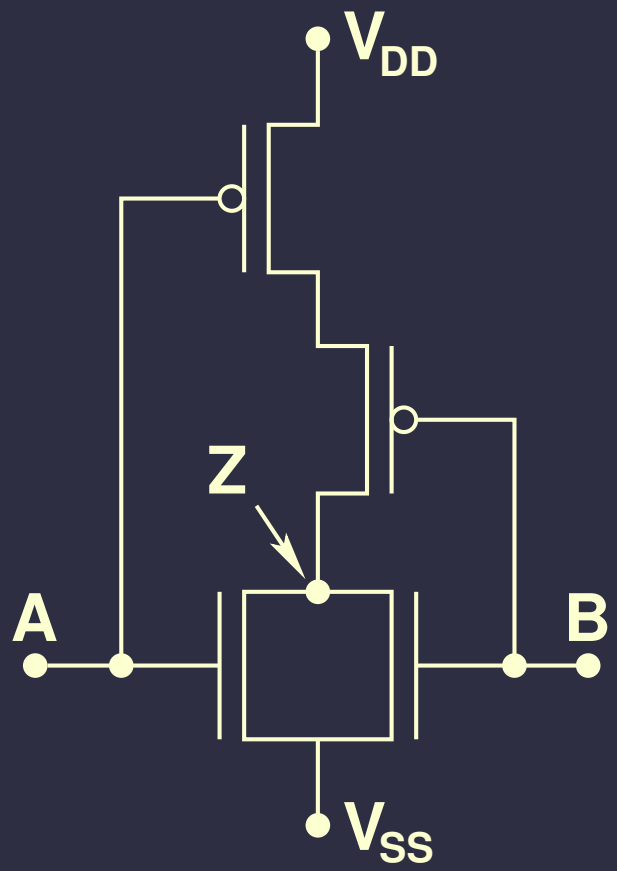
NOR operation



NOR operation



NOR operation



Section outline

4. Transistors and CMOS Design
    - AND and OR are harder than NAND and NOR
    - Transmission gates

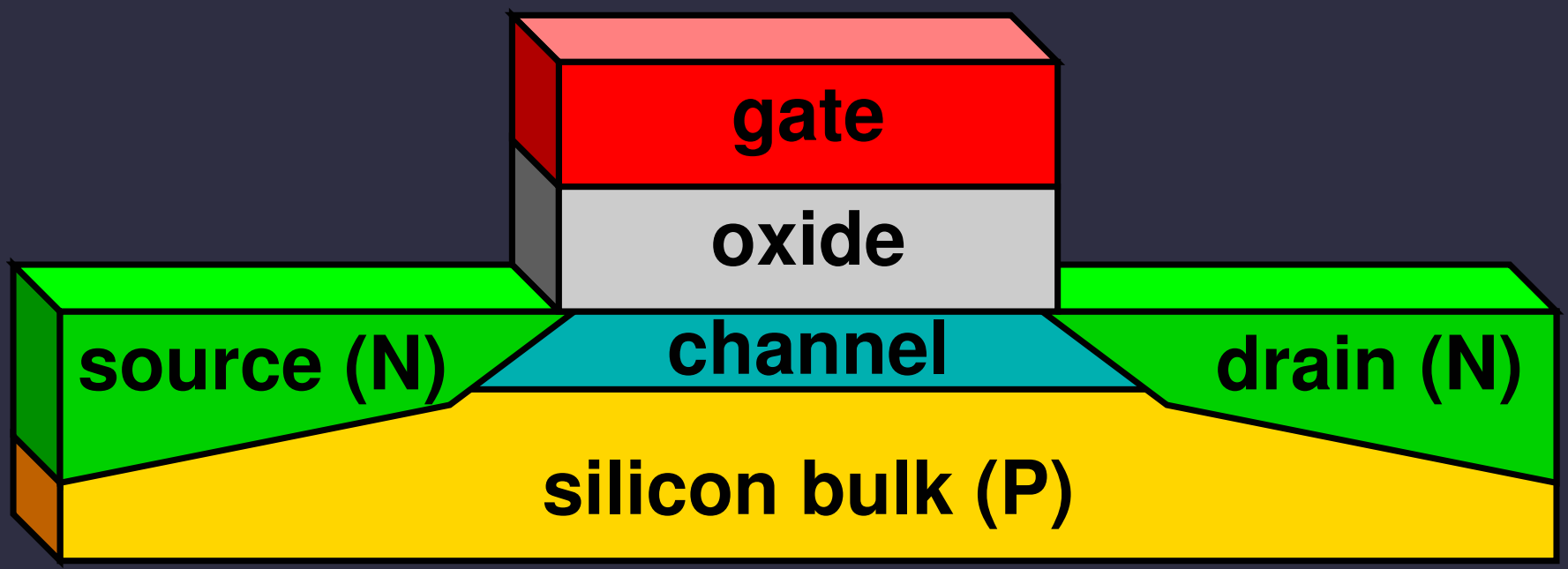
NMOS/PMOS transistors for AND/OR

- Recall that NMOS transmits low values easily...
- ...transmits high values poorly
- PMOS transmits high values easily...
- ...transmits low values poorly
- This is due to the effect of the transistors' *threshold* definitions

NMOS/PMOS transistors for AND/OR

- V_T, or threshold voltage, is commonly 0.7 V
- NMOS conducts when $V_{GS} > V_T$
- PMOS conducts when $V_{GS} < -V_T$
- What happens if an NMOS transistor's source is high?
- Or a PMOS transistor's source is low?
- Alternatively, if one states that $V_{TN} = 0.7$ V and $V_{TP} = -0.7$ V then NMOS conducts when $V_{GS} > V_{TN}$ and PMOS conducts when $V_{GS} < V_{TP}$

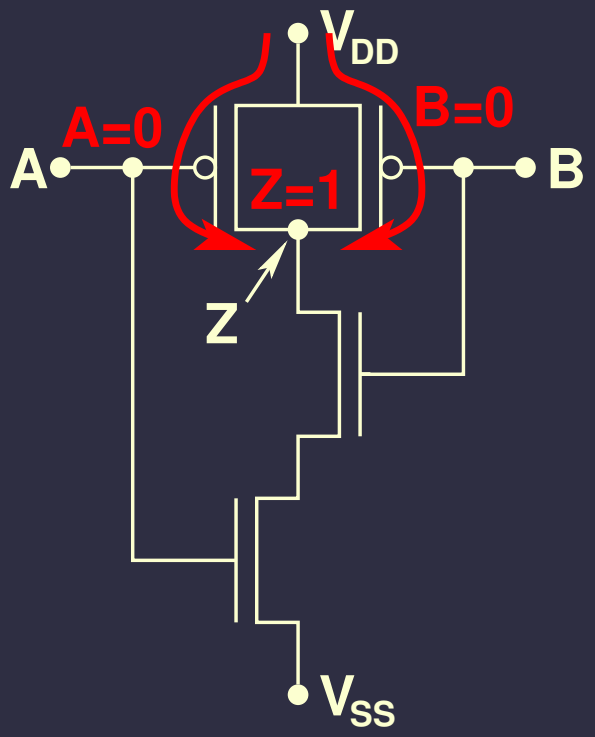
NMOS transistor



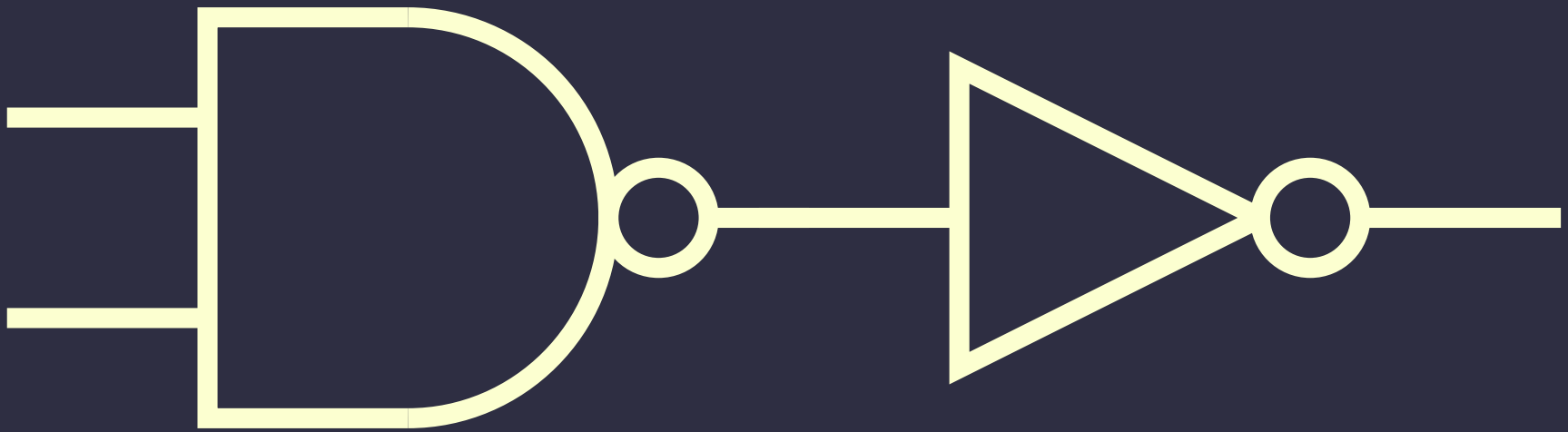
NMOS/PMOS transistors for AND/OR

- If an NMOS transistor's input were V_{DD} (high), for $V_{GS} > V_{TN}$, the gate would require a higher voltage than V_{DD}
- If an PMOS transistor's input were V_{SS} (low), for $V_{GS} < V_{TP}$, the gate would require a lower voltage than V_{SS}

NAND/NOR easy to build in CMOS



AND/OR requires more area, power, time



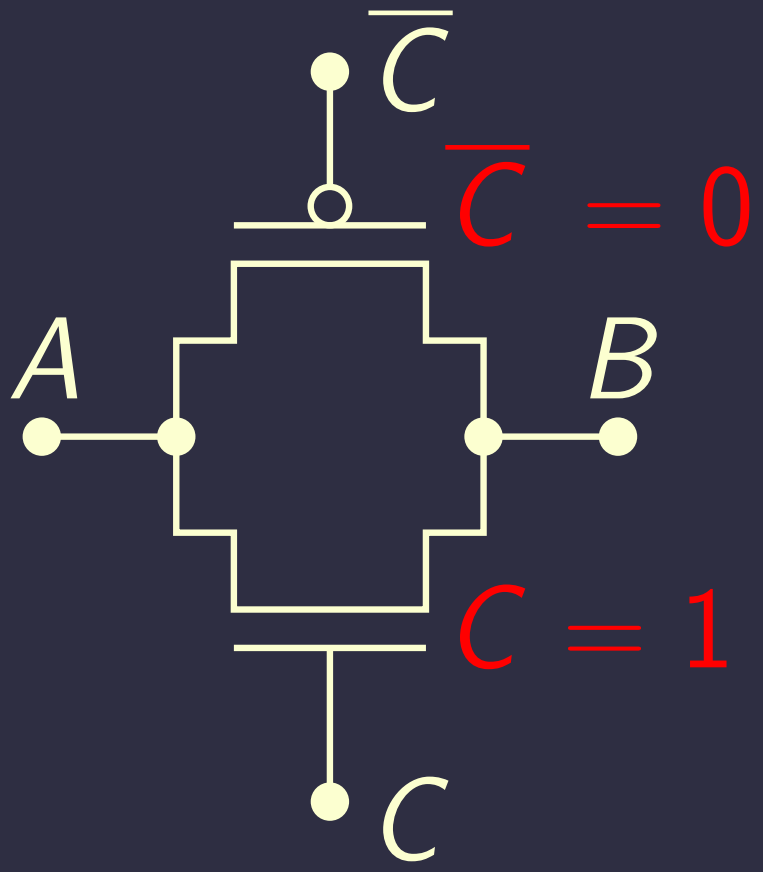
Section outline

4. Transistors and CMOS Design
    - AND and OR are harder than NAND and NOR
    - Transmission gates

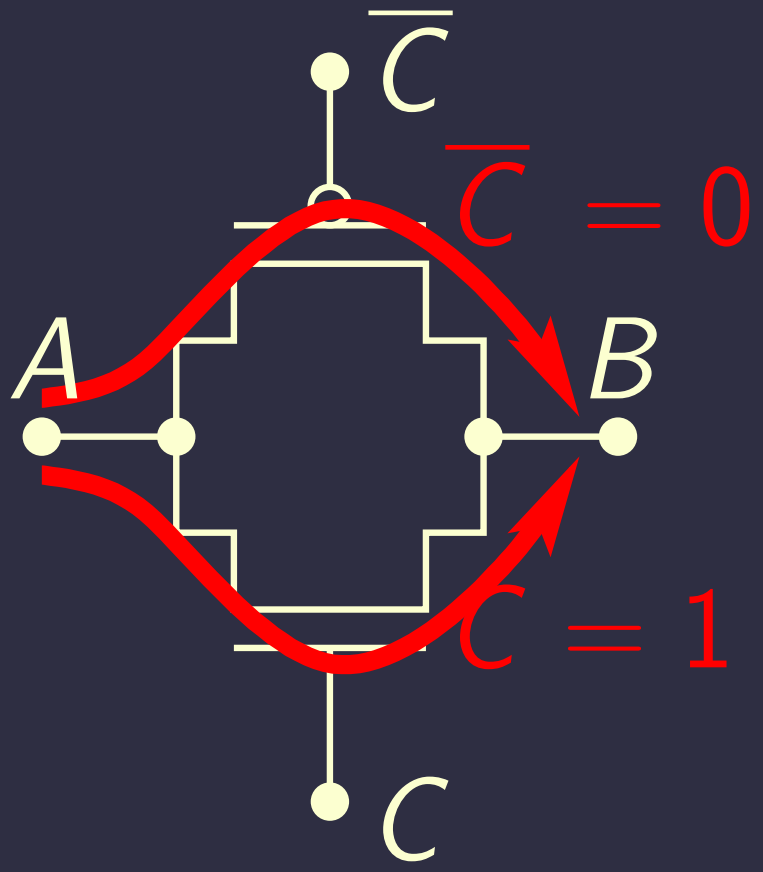
CMOS transmission gates (switches)

- NMOS is good at transmitting 0s
  - Bad at transmitting 1s
- PMOS is good at transmitting 1s
  - Bad at transmitting 0s
- To build a switch, use both: CMOS

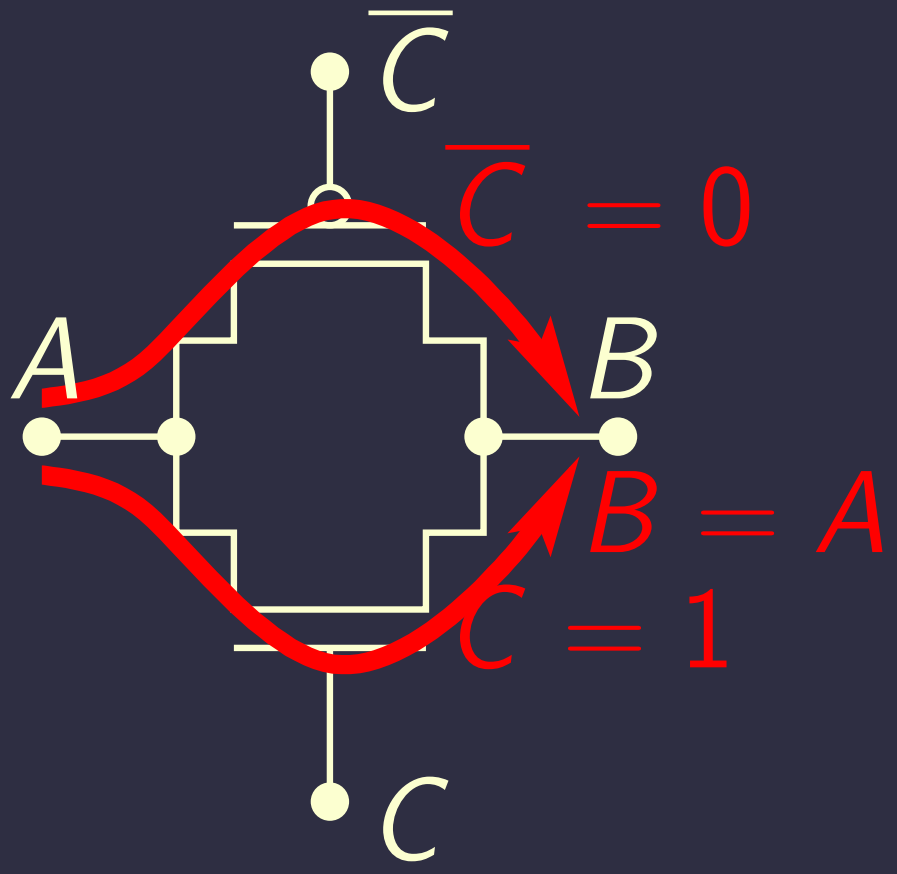
CMOS transmission gate (TG)



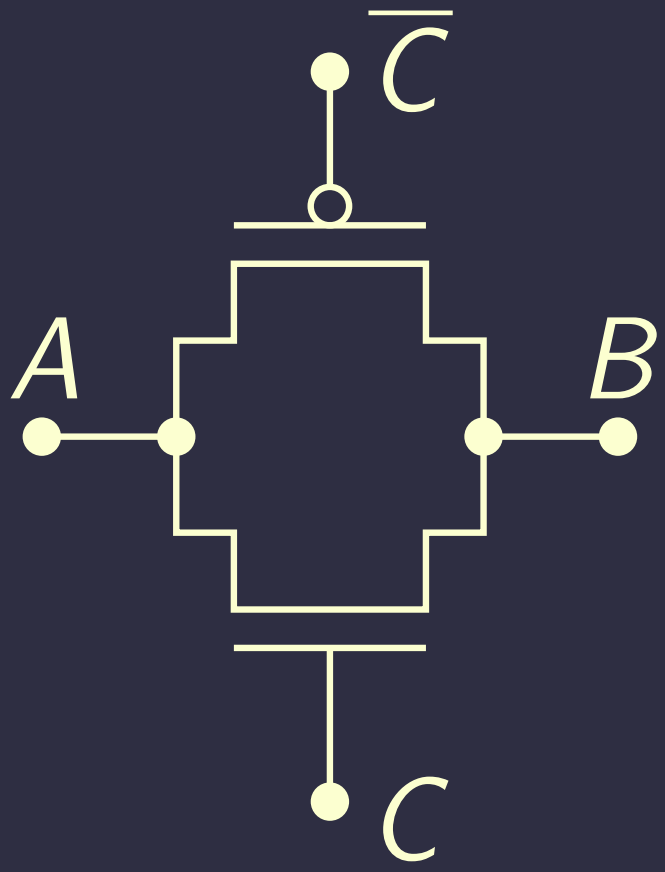
CMOS transmission gate (TG)



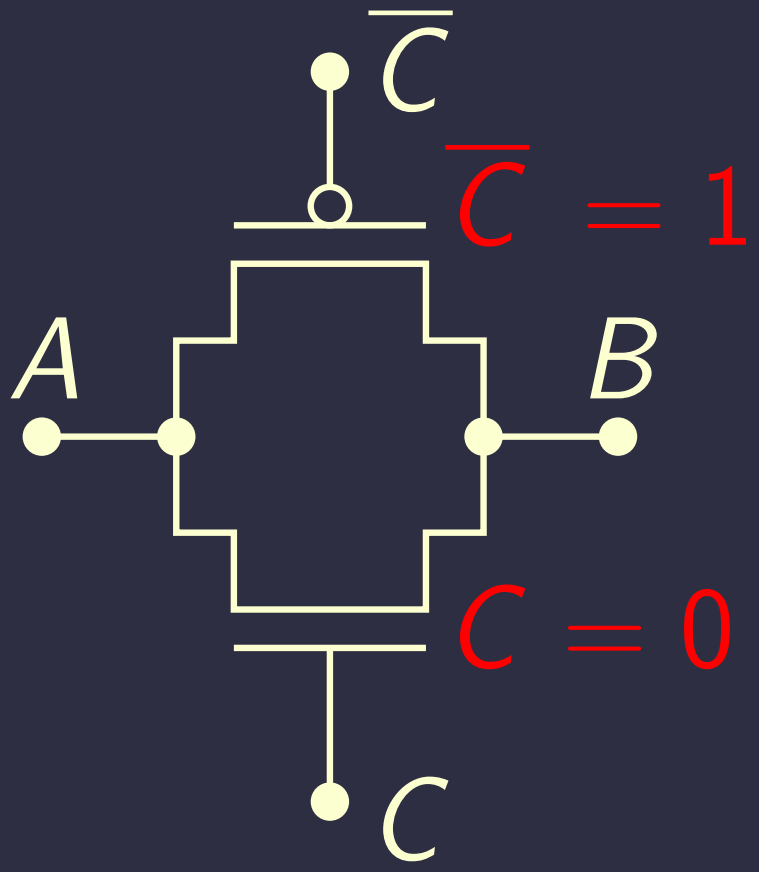
CMOS transmission gate (TG)



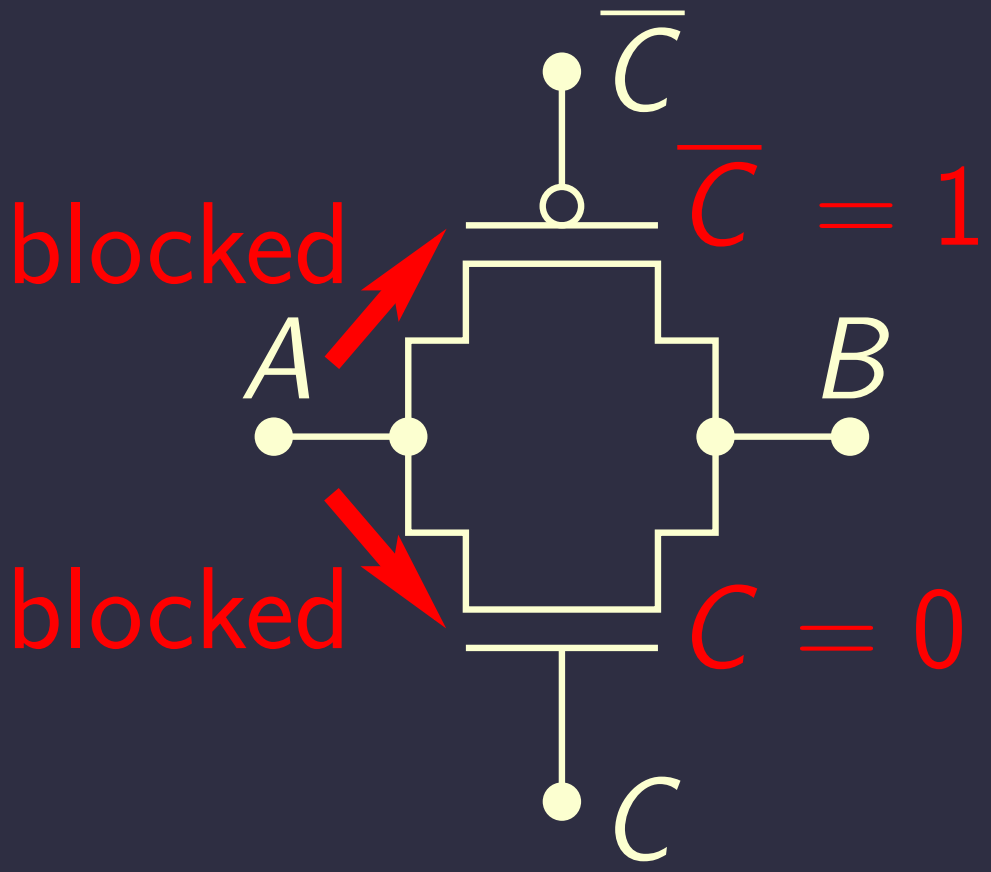
CMOS transmission gate (TG)



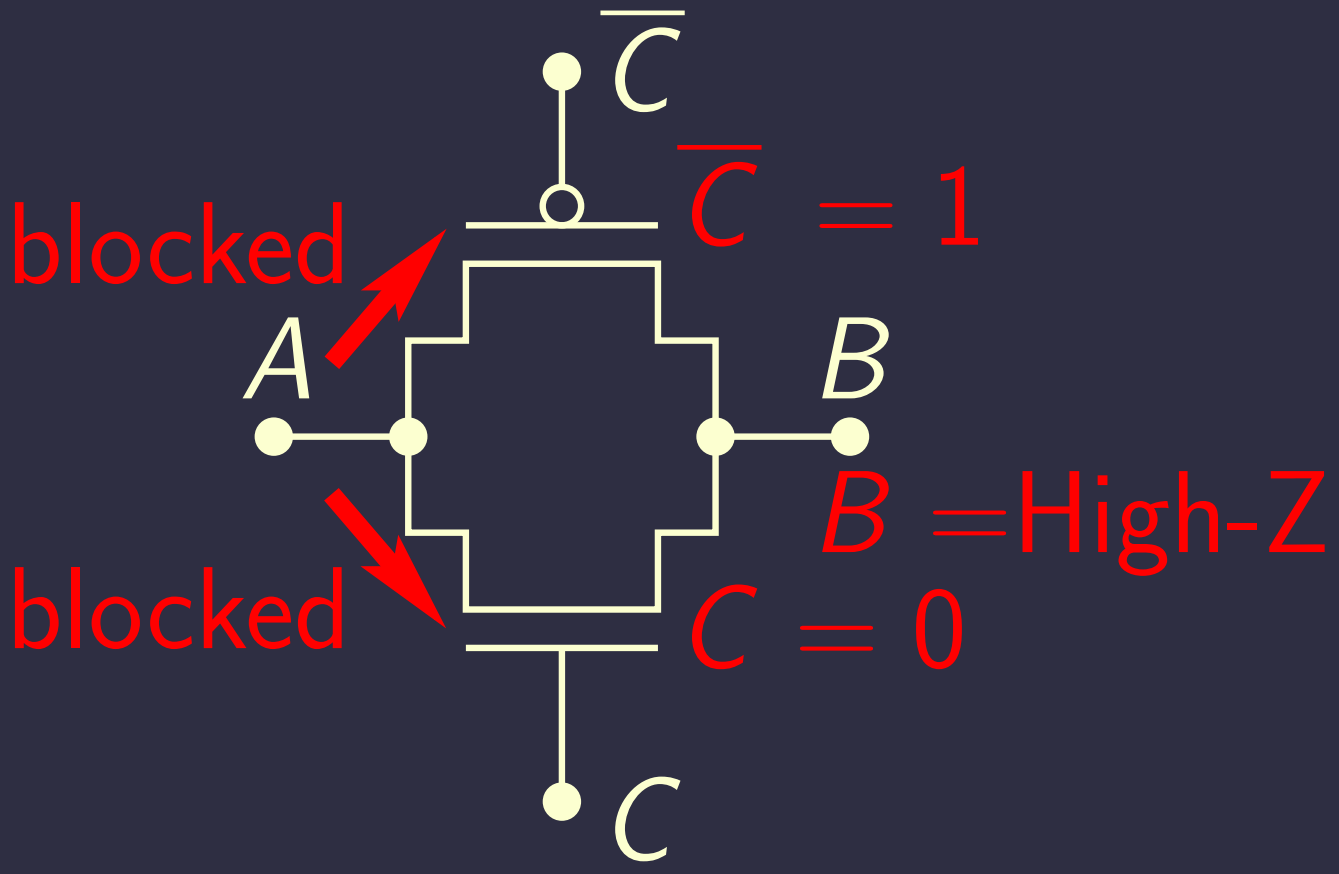
CMOS transmission gate (TG)



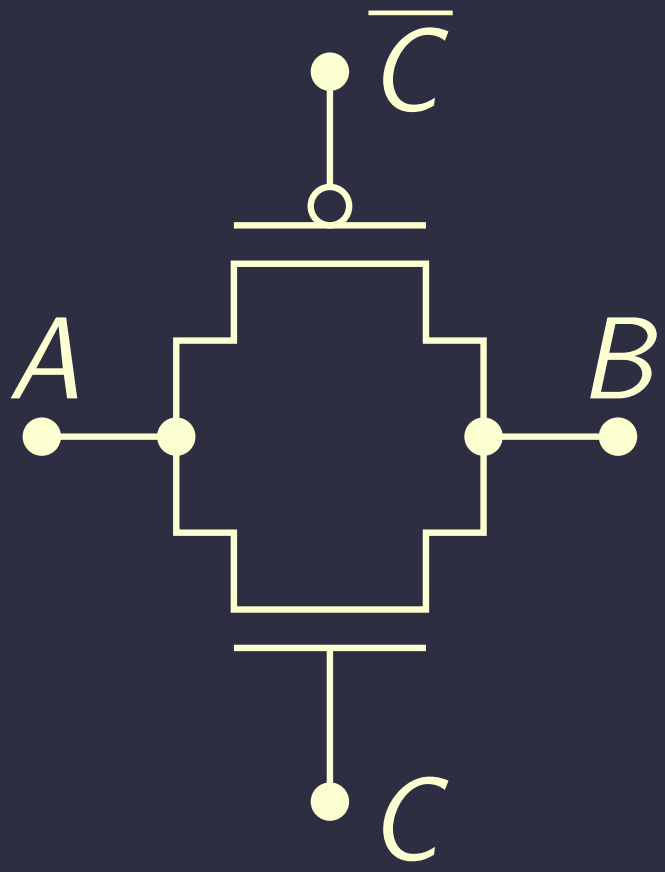
CMOS transmission gate (TG)



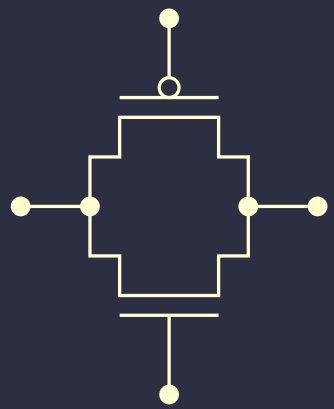
CMOS transmission gate (TG)

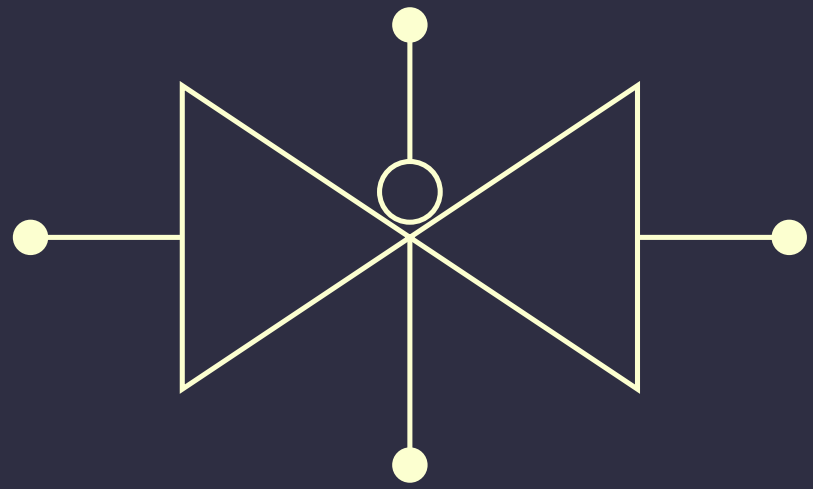


CMOS transmission gate (TG)



Other TG diagram





What can we build with TGs?

- Anything. . . try some examples.

Computer geek culture reference

- http://slashdot.org/
- http://www.python.org/

Outline

Reading assignment

- M. Morris Mano and Charles R. Kime. *Logic and Computer Design Fundamentals*. Prentice-Hall, NJ, fourth edition, 2008
- Sections 2.3–2.5