Advanced Digital Logic Design – EECS 303

http://ziyang.eecs.northwestern.edu/eecs303/

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

NORTHWESTERN
UNIVERSITY

Outline

Section outline

1. Transistors in digital systems
   - Switches
   - CMOS
   - AND and OR are harder than NAND and NOR
   - Transmission gates

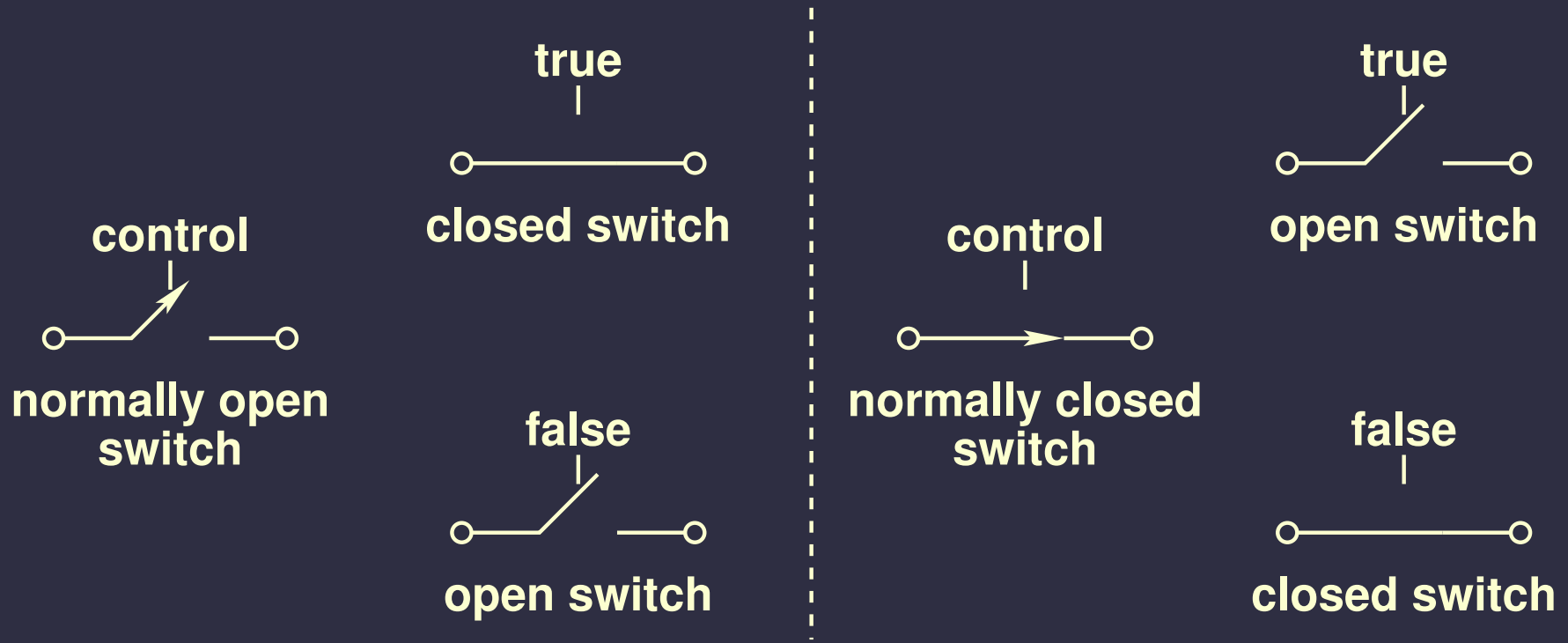
Switch-based design representation

- 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?

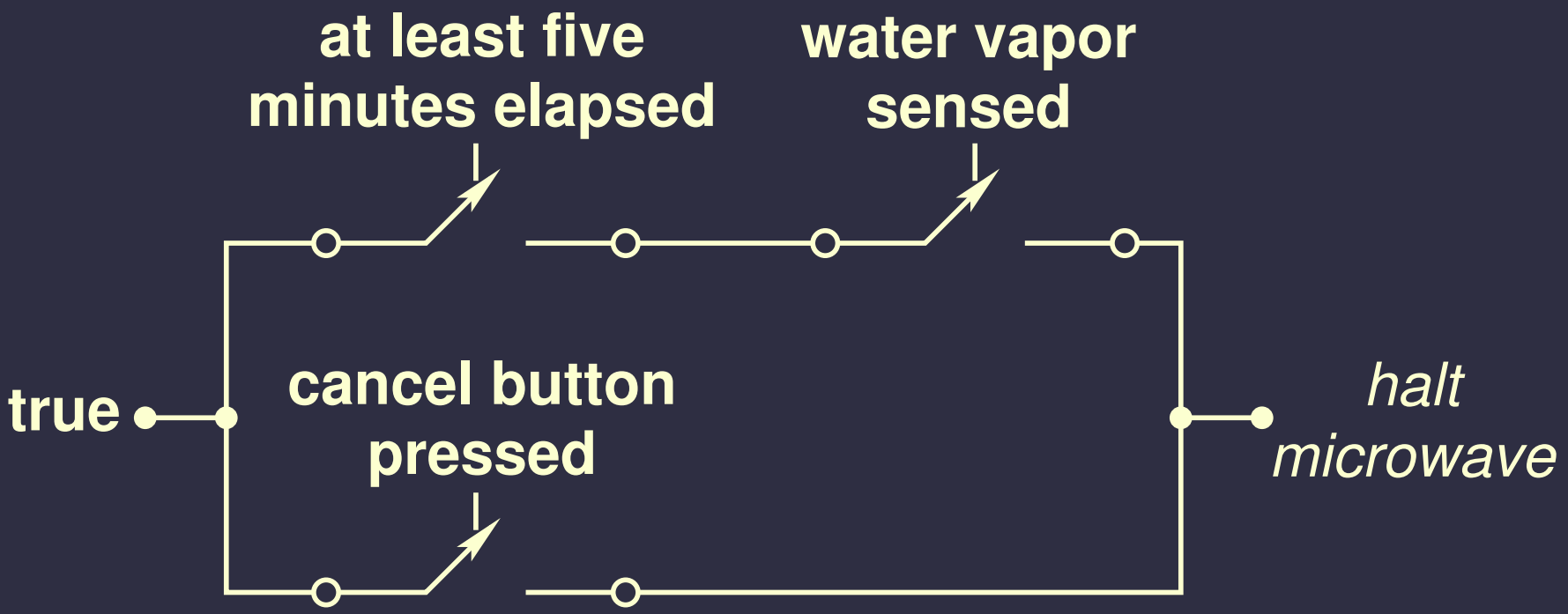
Switch-based design representation

- 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

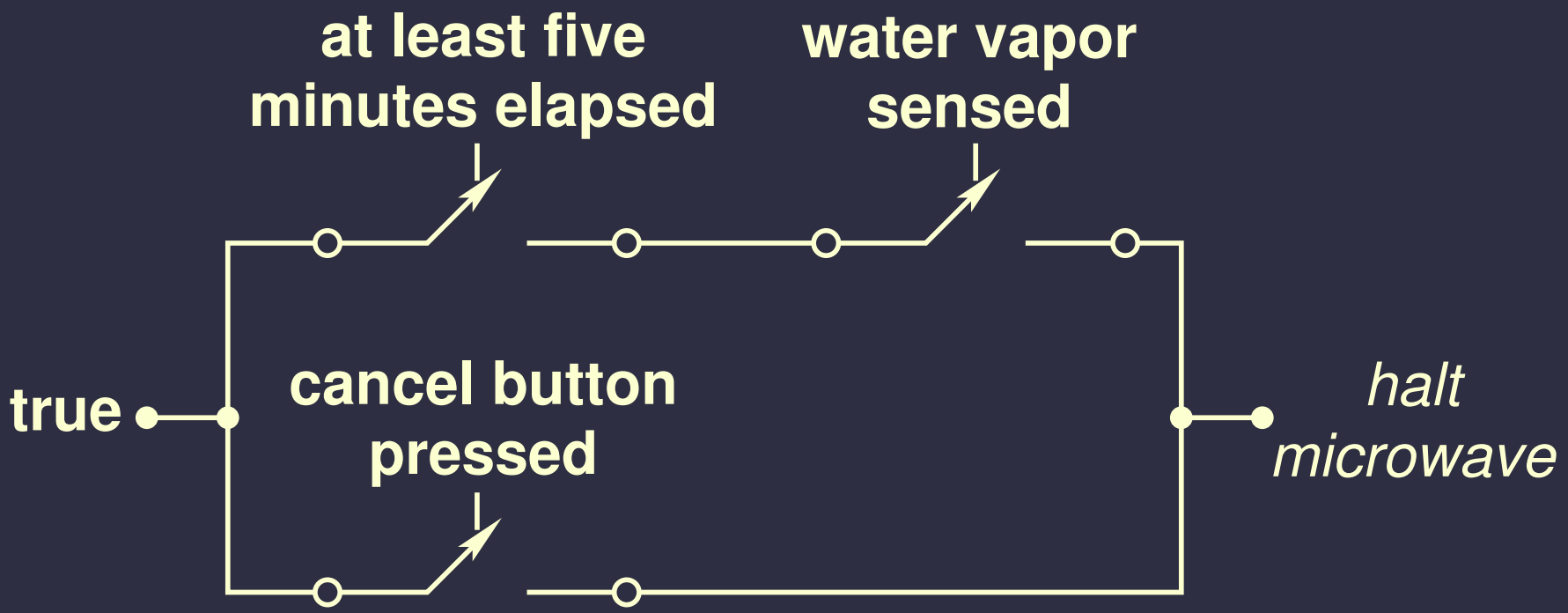


Microwave control example



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

Microwave control example



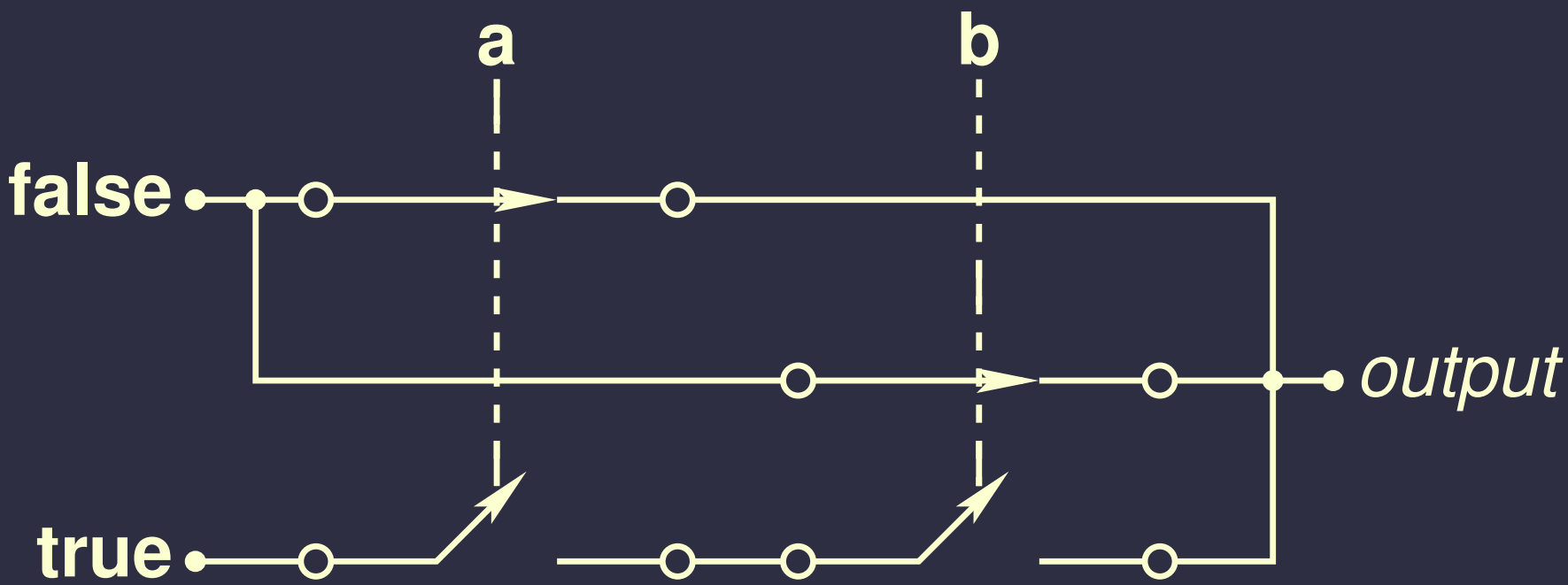
- What happens if the cancel button is not pressed and five minutes haven't yet passed?
  - The output value is undefined.

Constraints on network output

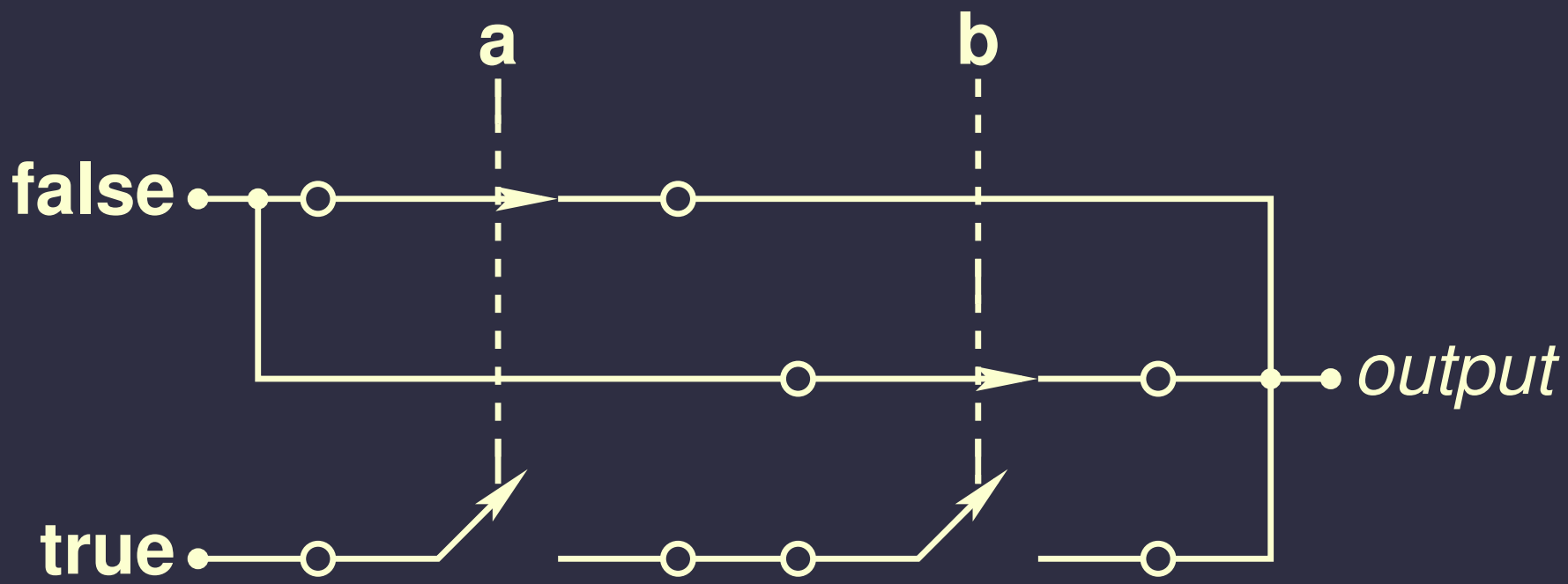
Under all possible combinations of input values

- 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

Section outline

1. Transistors in digital systems
   - Switches
   - CMOS
   - AND and OR are harder than NAND and NOR
   - Transmission gates

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

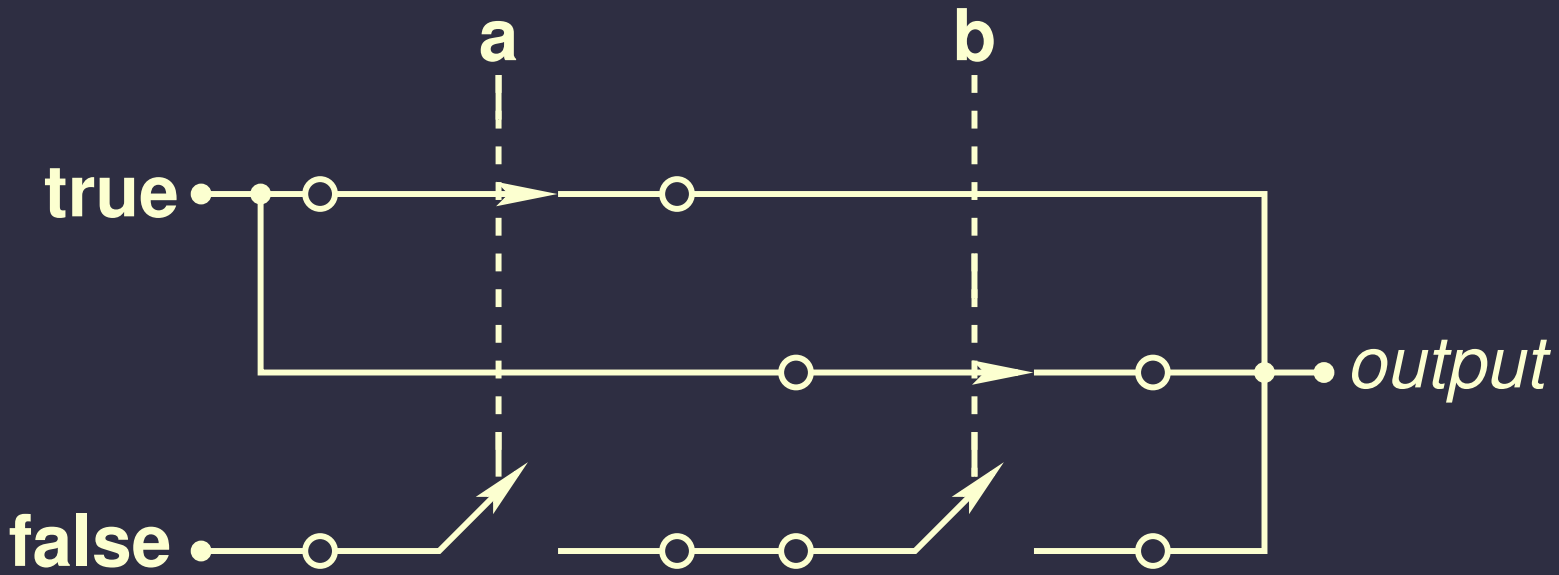
- Metal Oxide Semiconductor
- Positive and negative carriers
- Complimentary MOS
- PMOS gates are like normally closed switches that are good at transmitting only true (high) signals
- NMOS gates are like normally open switches that are good at transmitting only false (low) signals

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

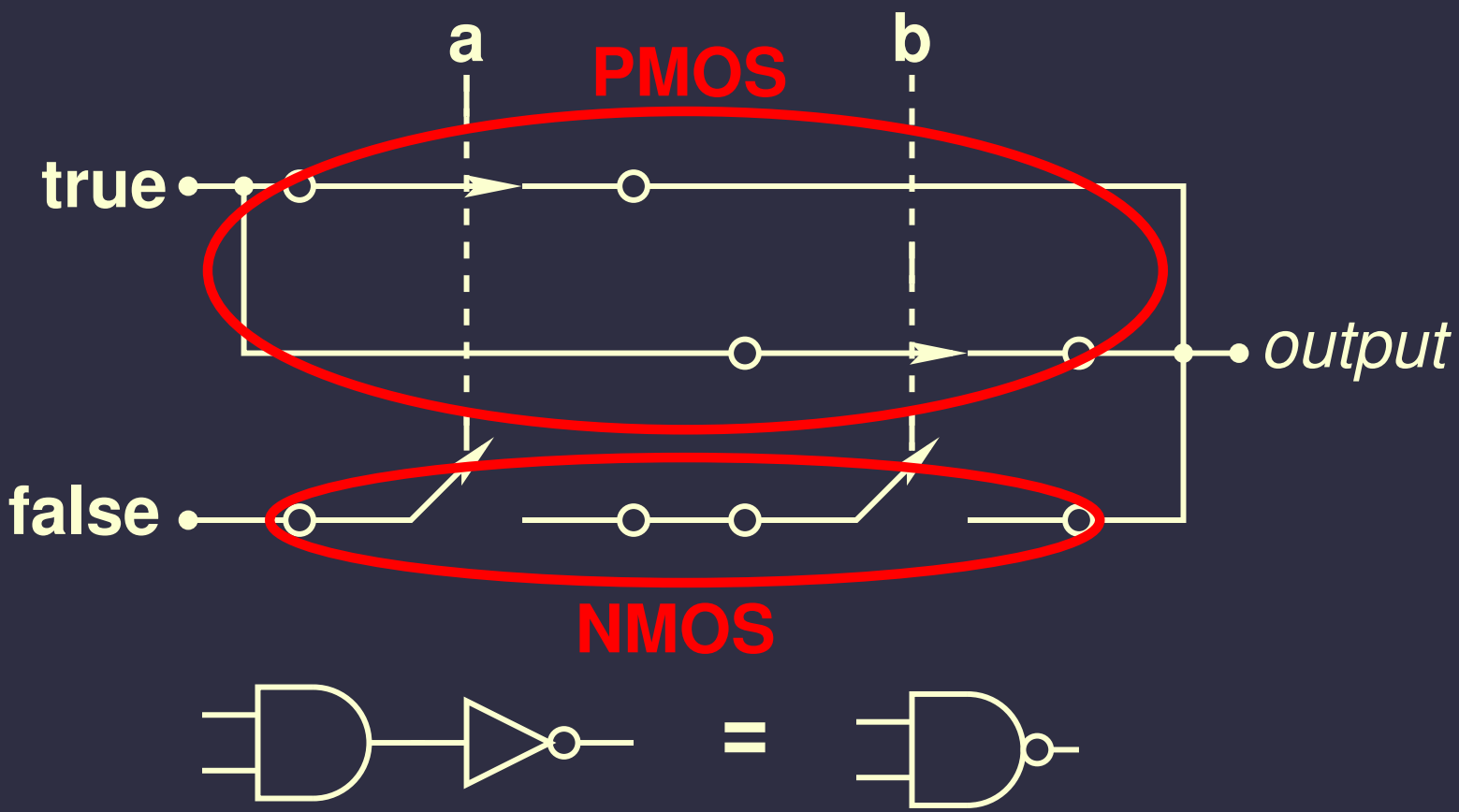
NAND gate

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



NAND gate

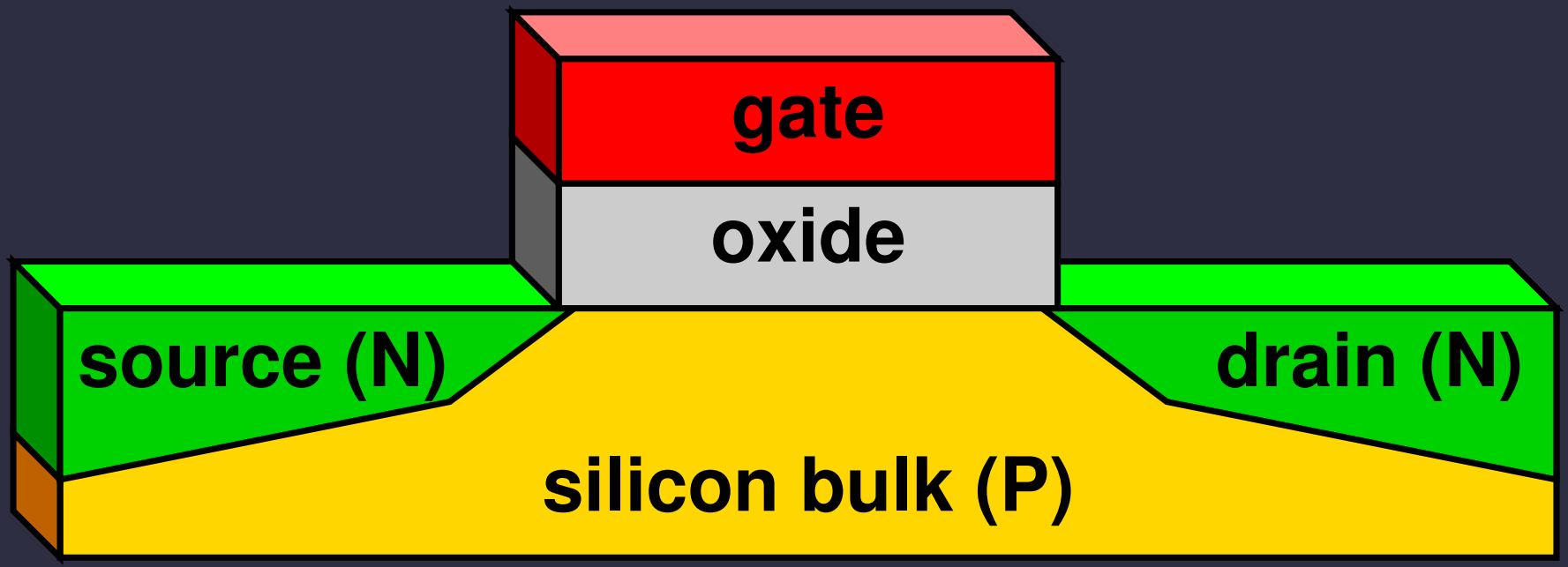
Therefore, *NAND* and *NOR* gates are used in CMOS design instead of *AND* and *OR* gates



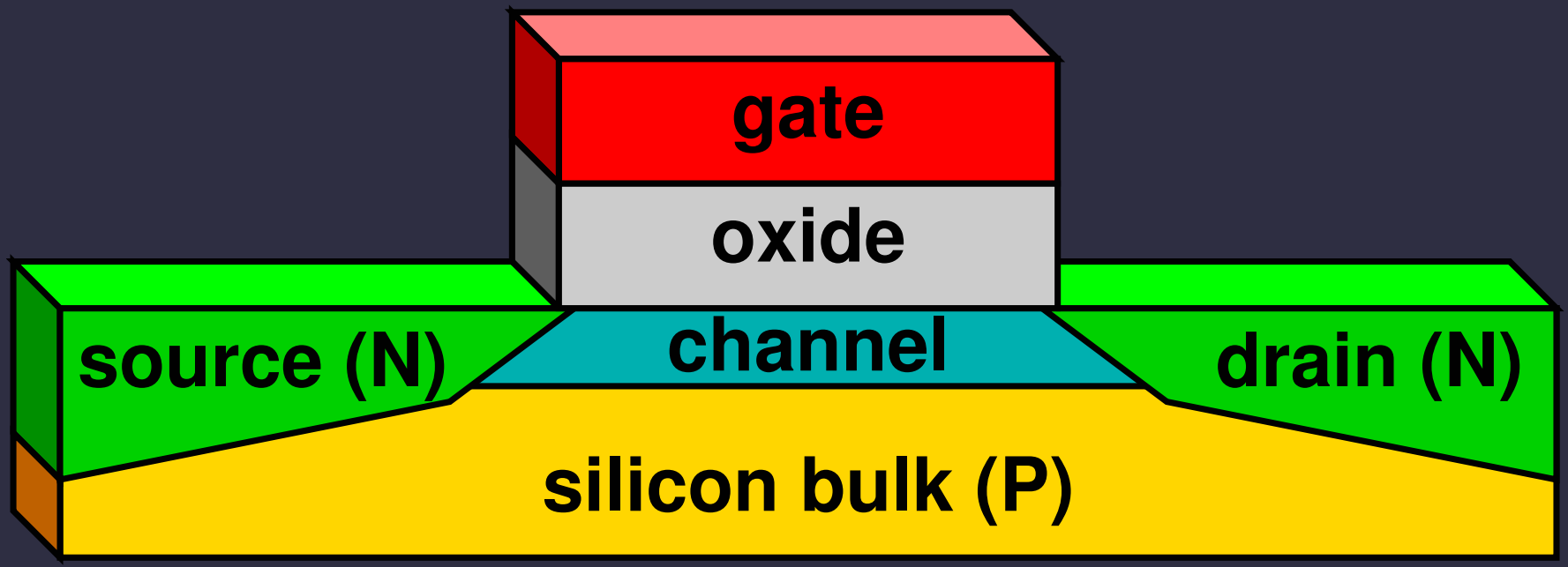
Transistors

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

NMOS transistor



NMOS transistor



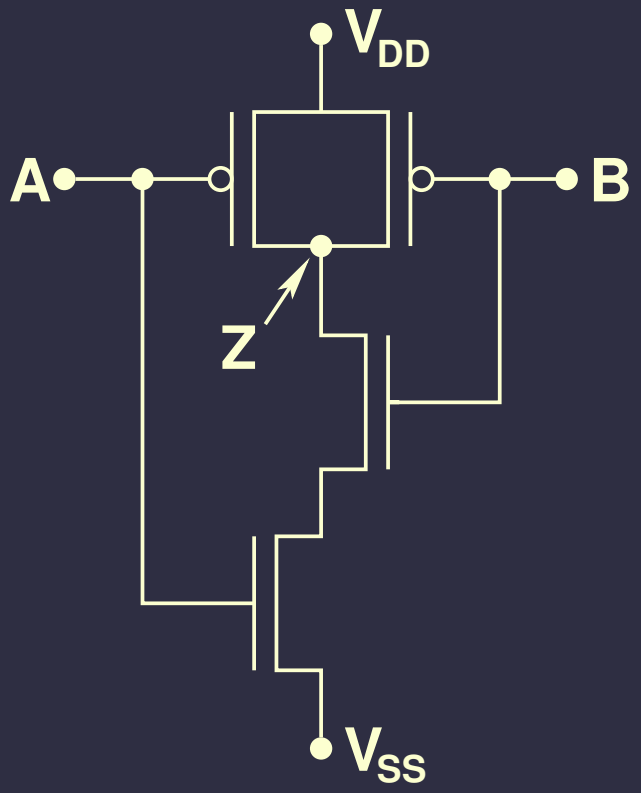
NMOS transistor

- 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

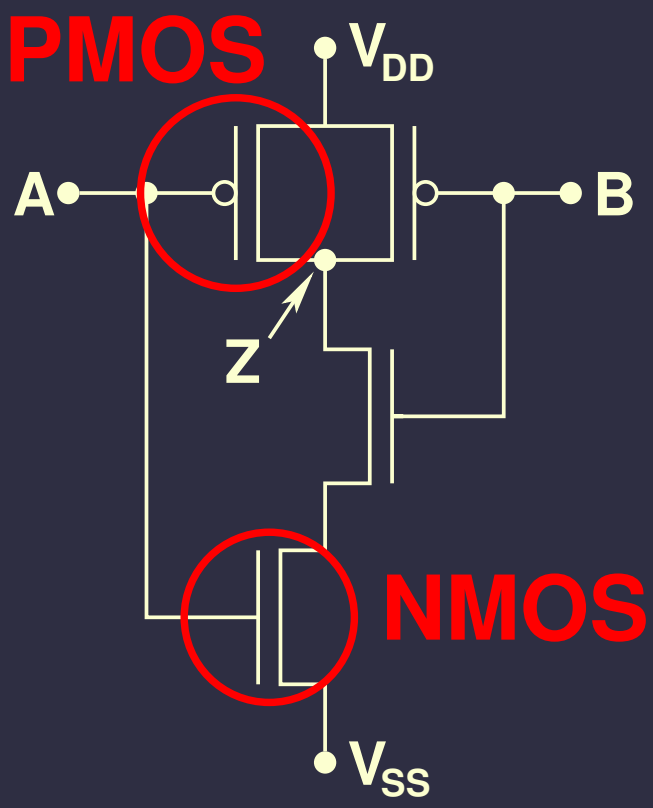
CMOS

- 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)

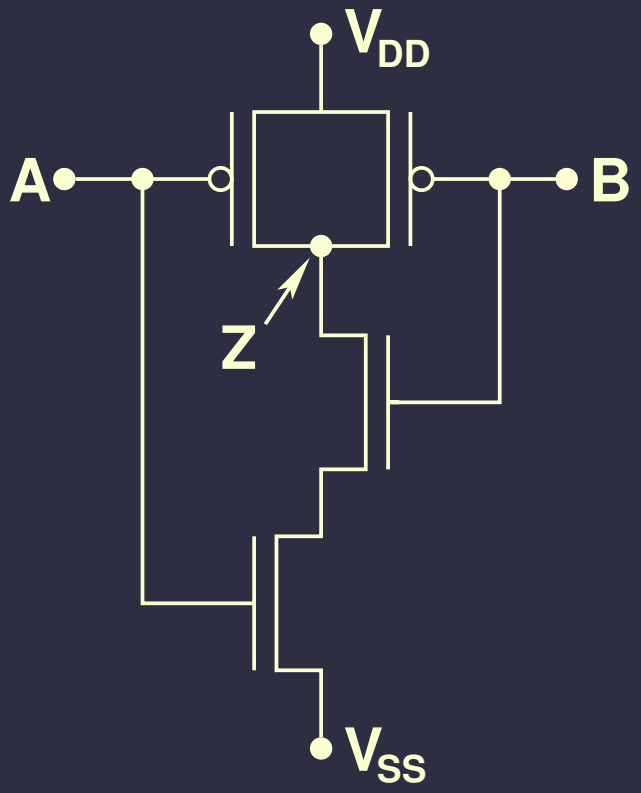
CMOS NAND gate



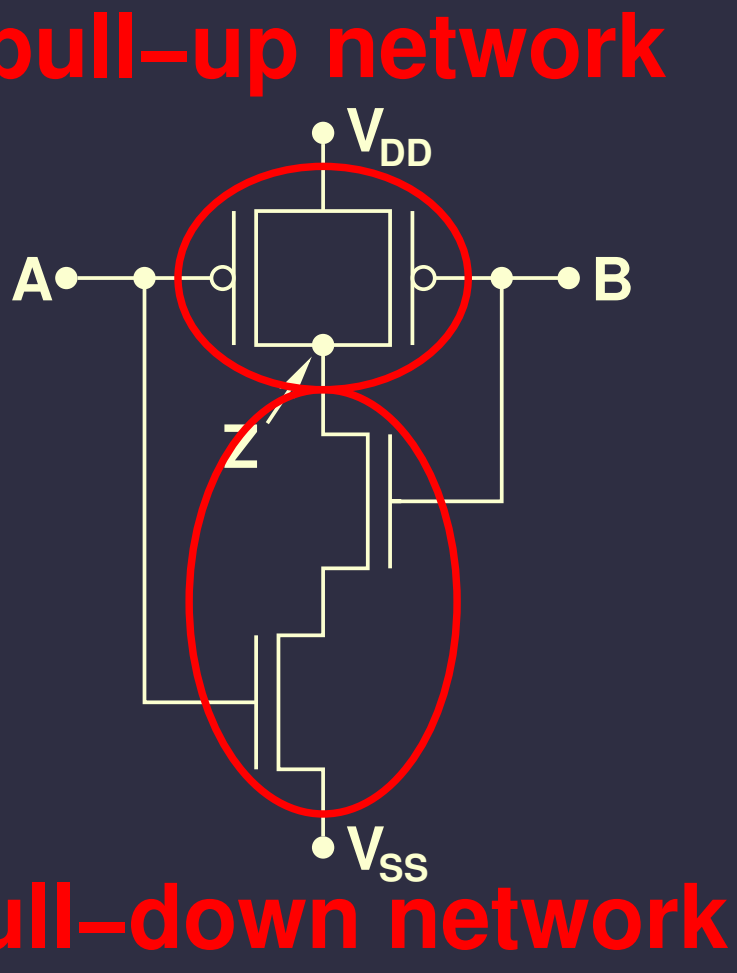
CMOS NAND gate



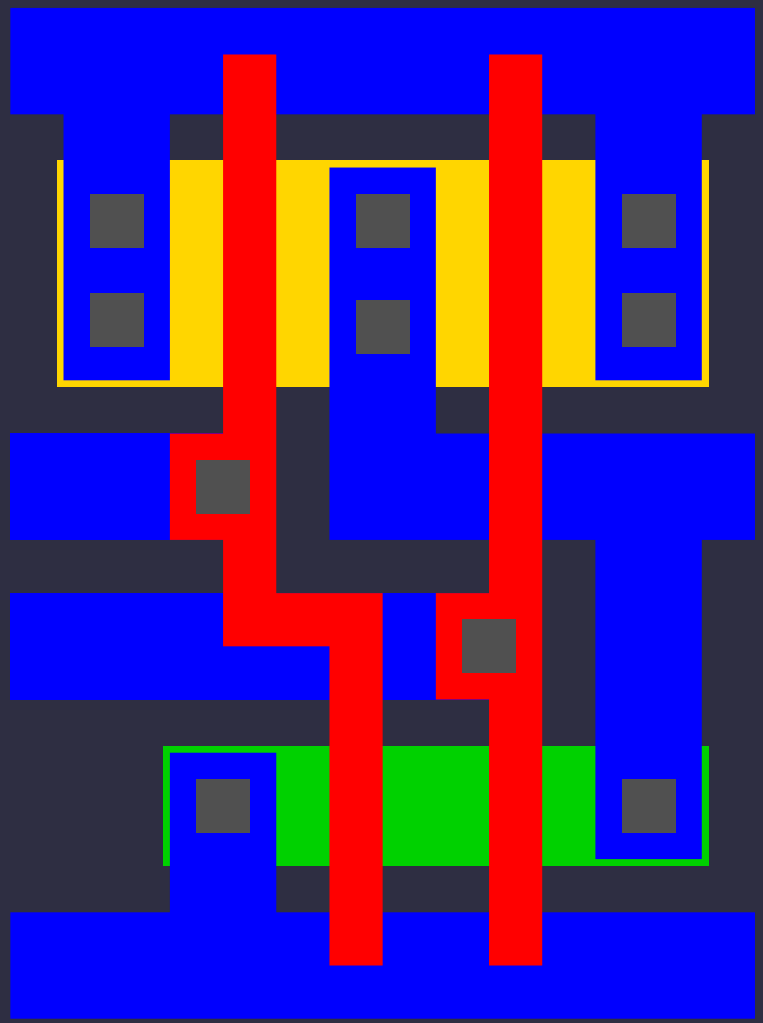
CMOS NAND gate



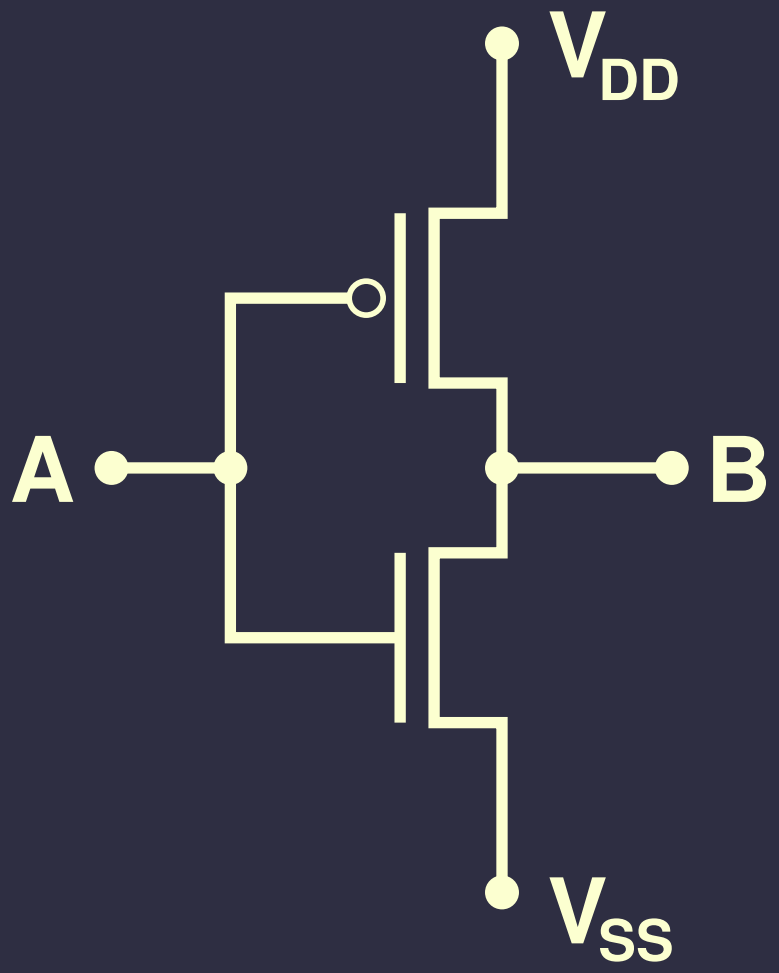
CMOS NAND gate



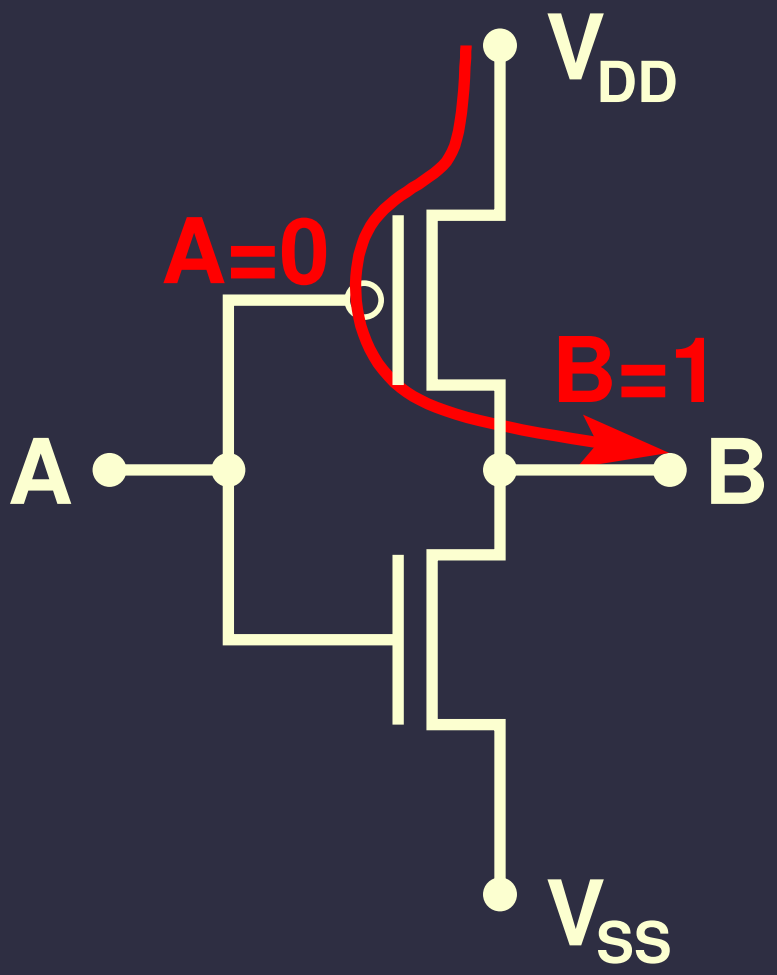
CMOS NAND gate layout



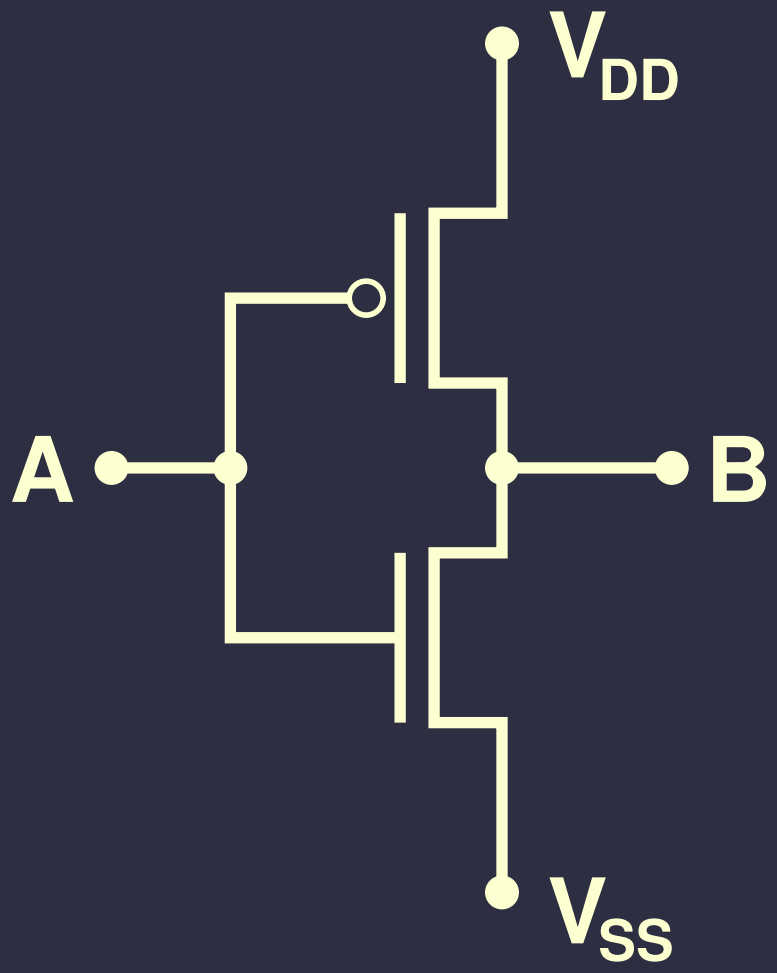
CMOS inverter operation



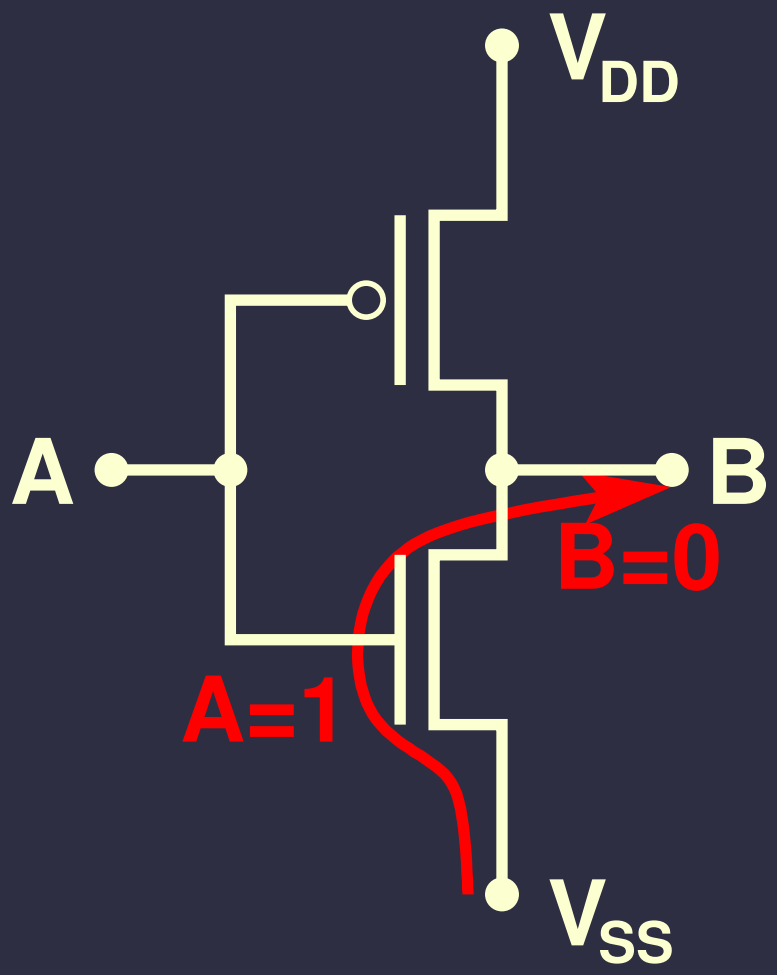
CMOS inverter operation



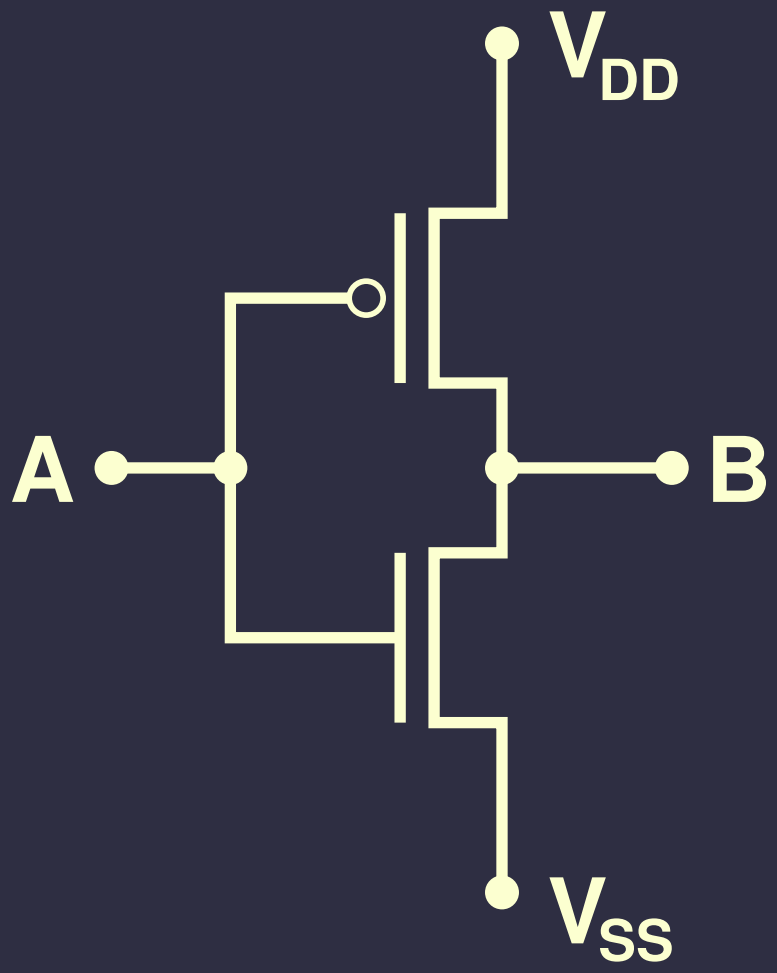
CMOS inverter operation



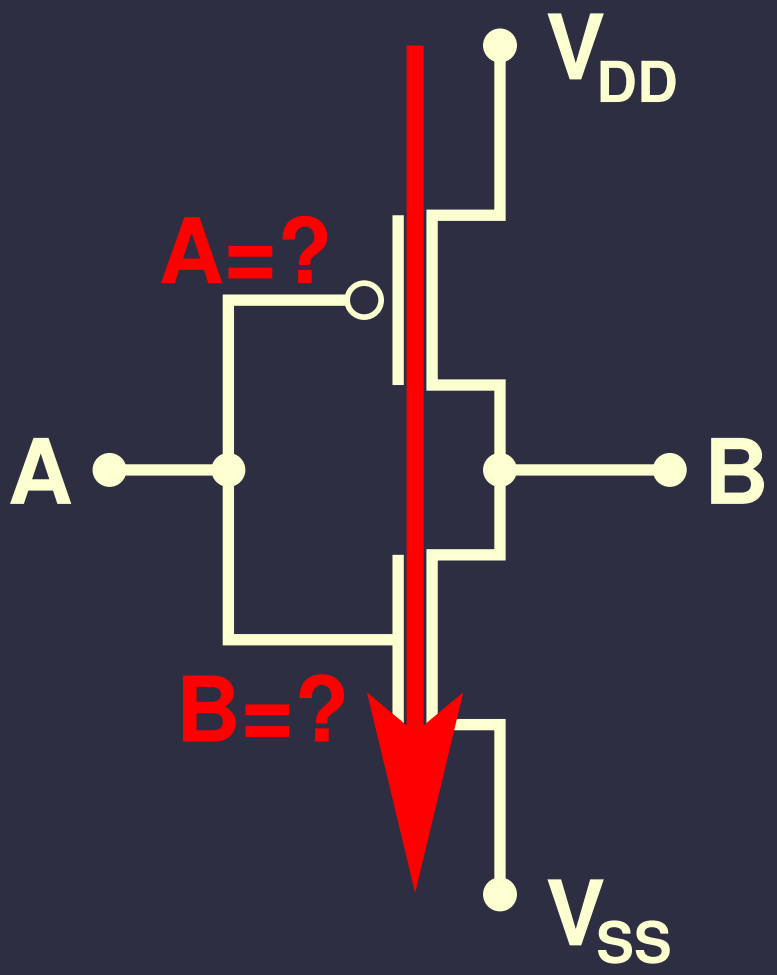
CMOS inverter operation



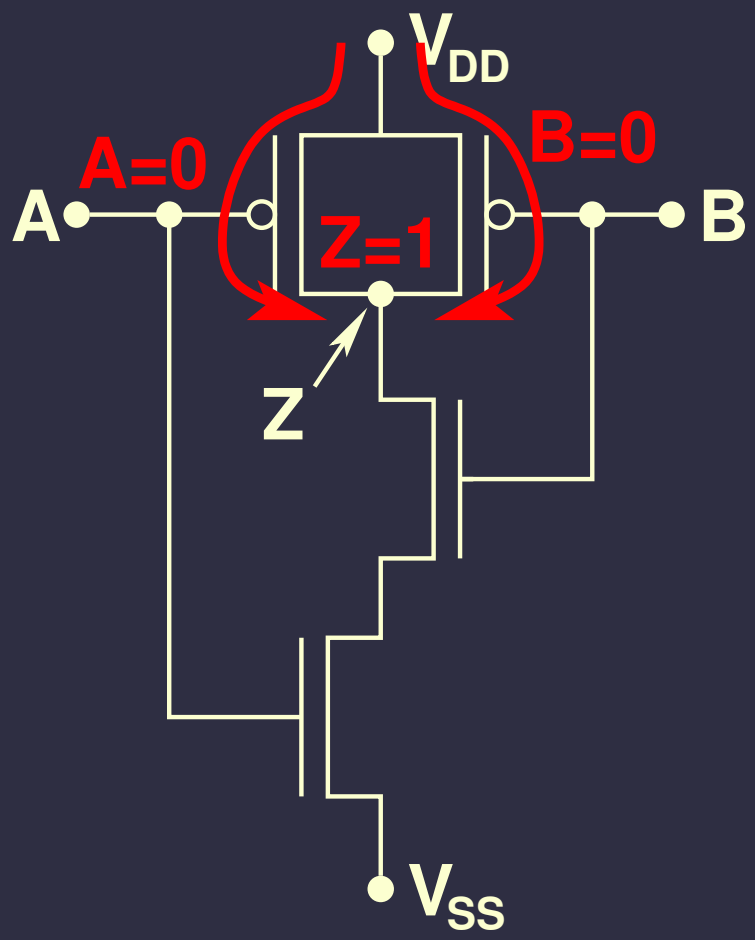
CMOS inverter operation



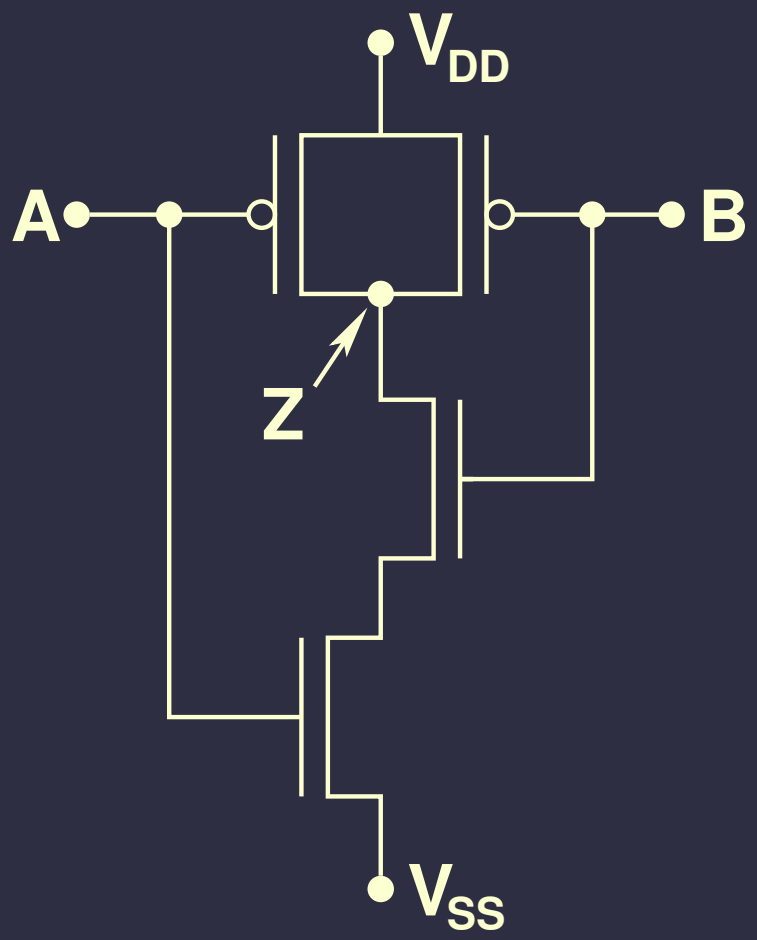
CMOS inverter operation



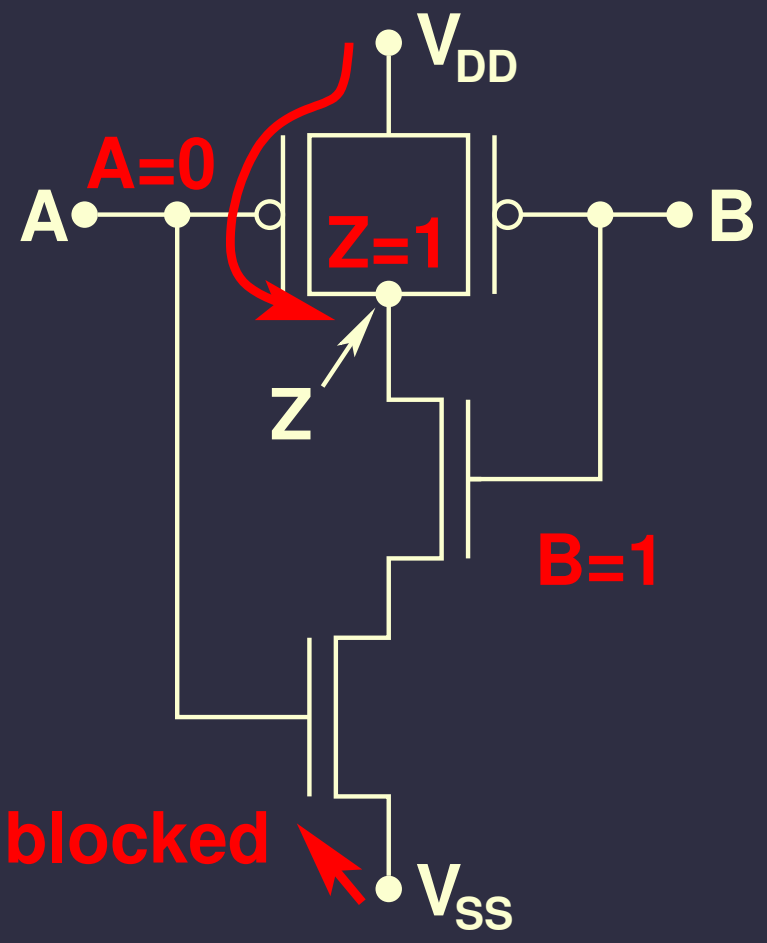
NAND operation



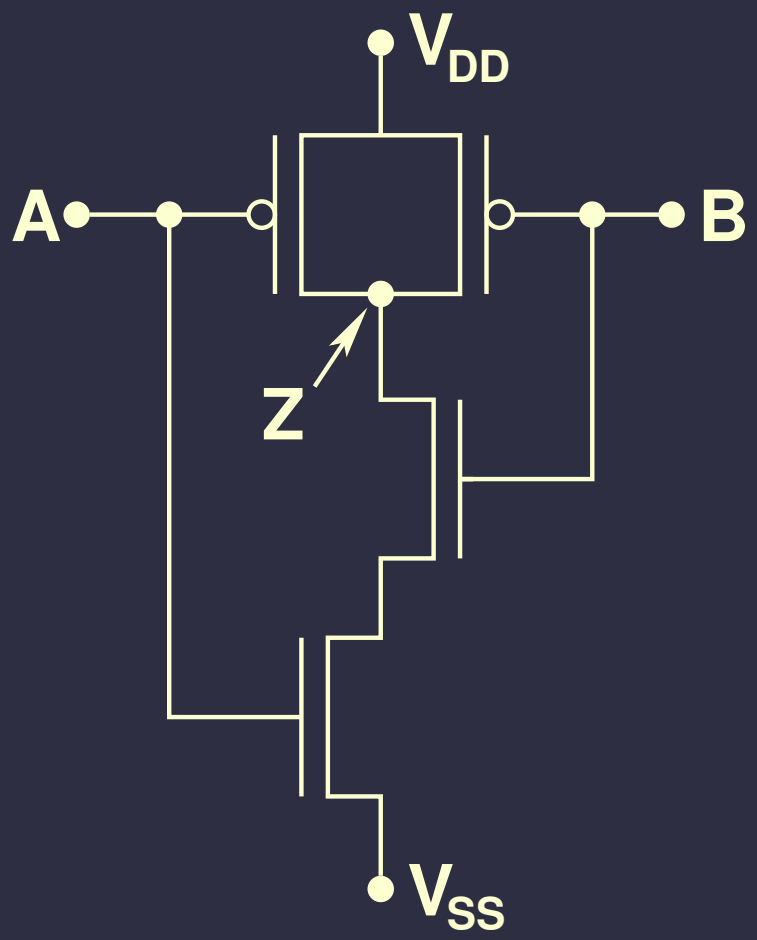
NAND operation



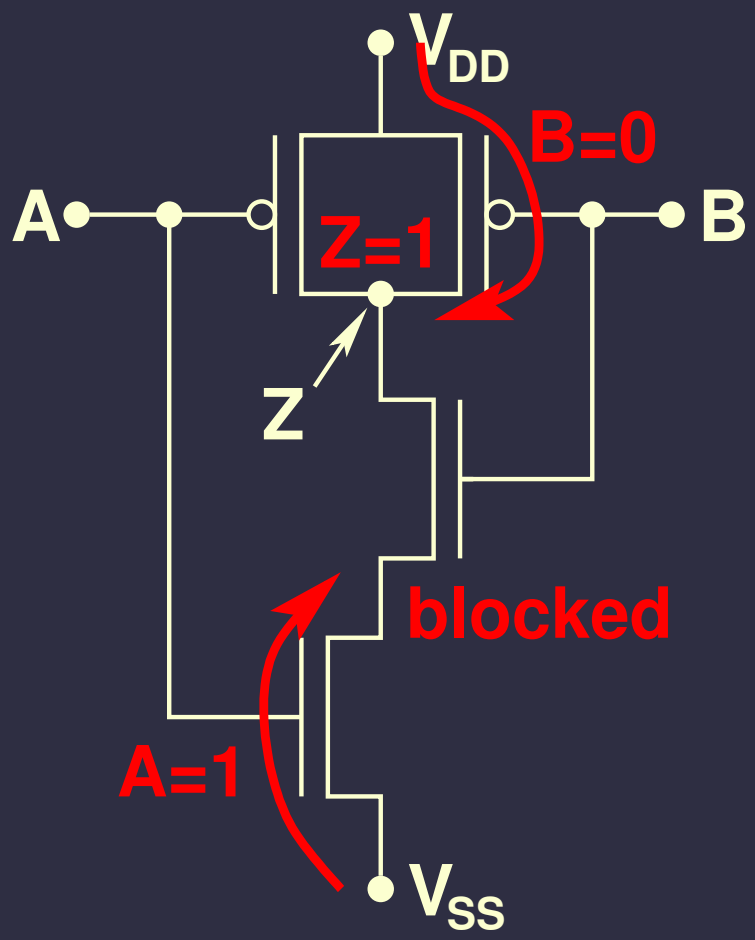
NAND operation



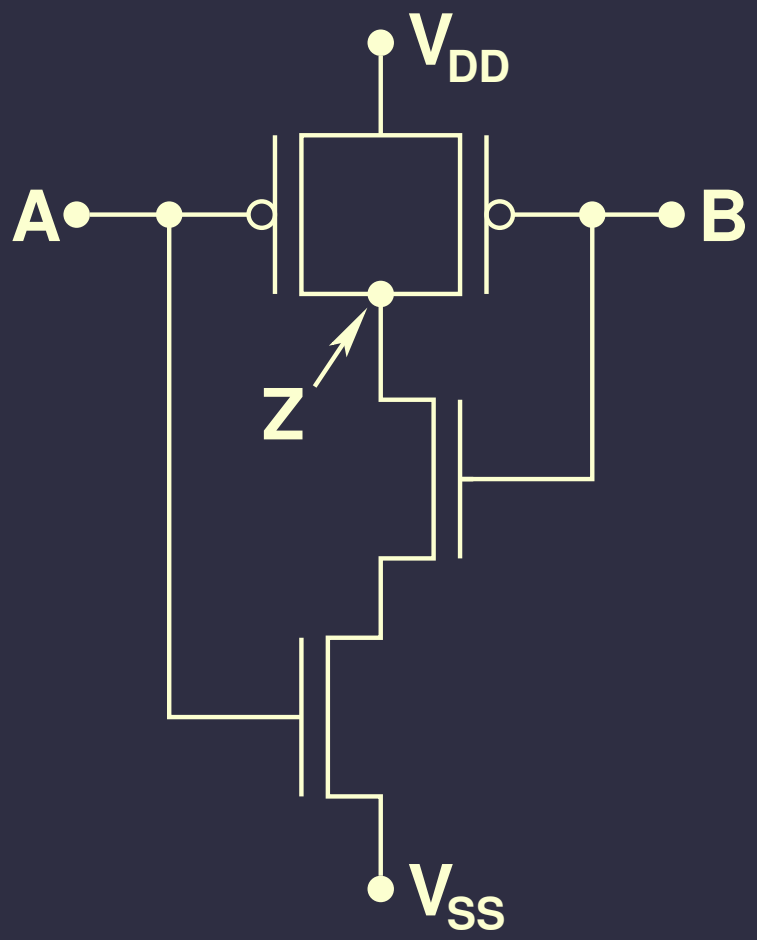
NAND operation



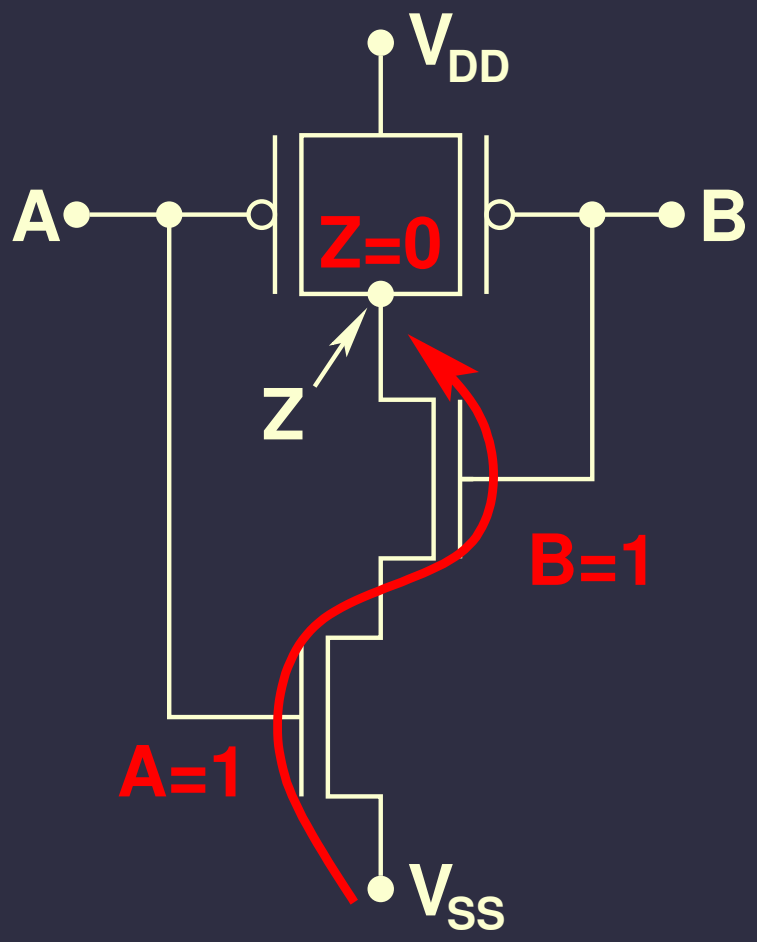
NAND operation



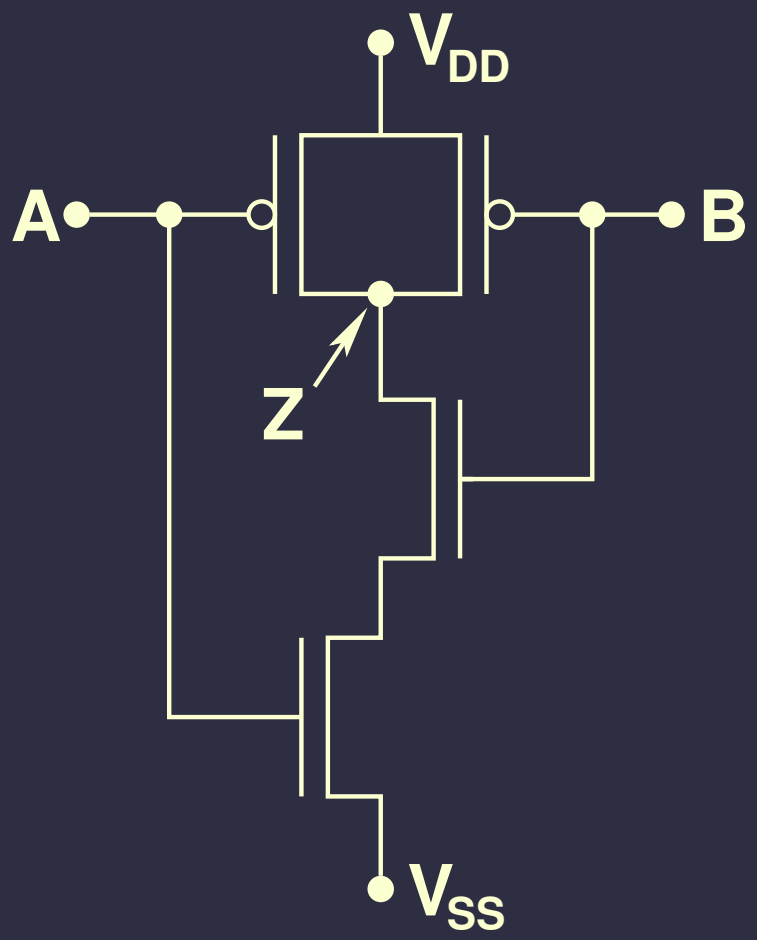
NAND operation



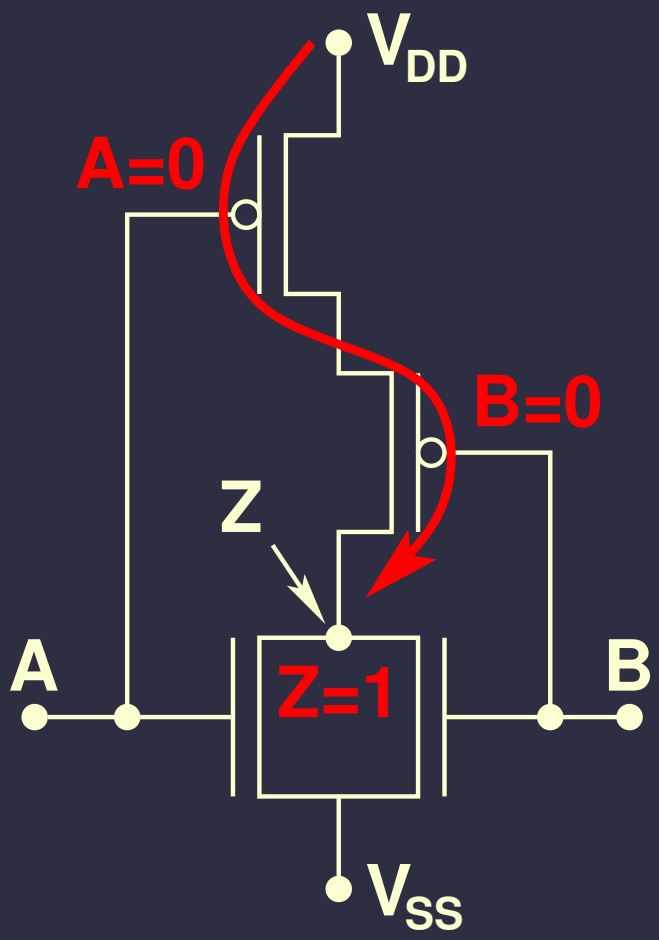
NAND operation



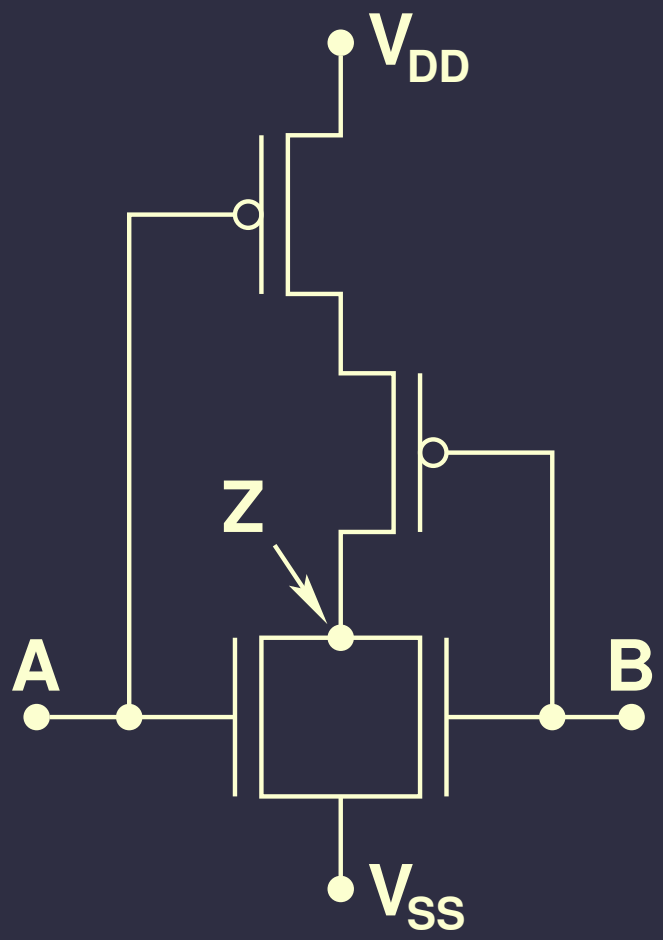
NAND operation



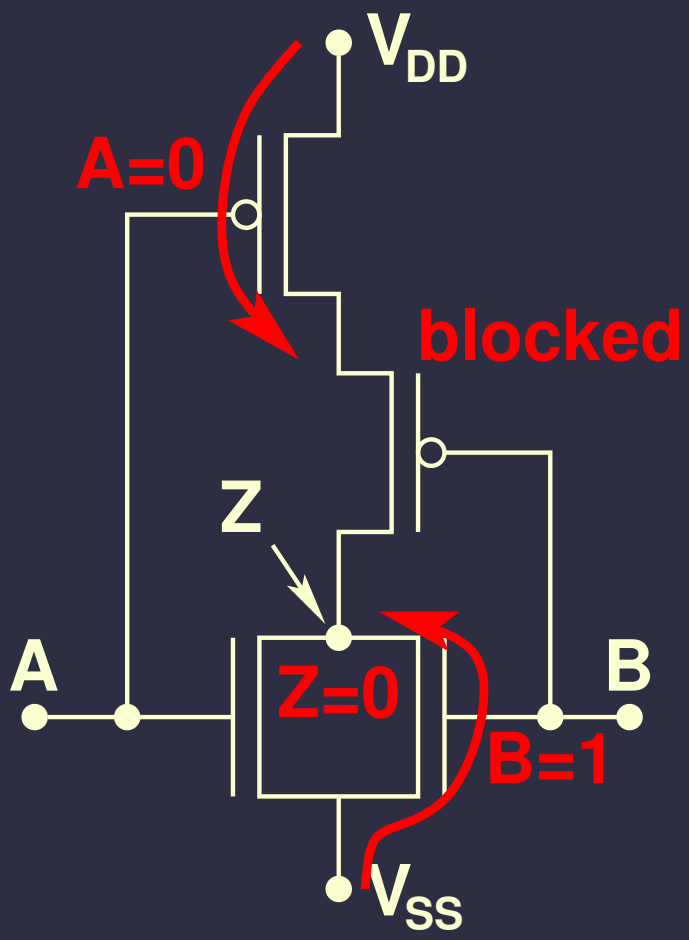
NOR operation



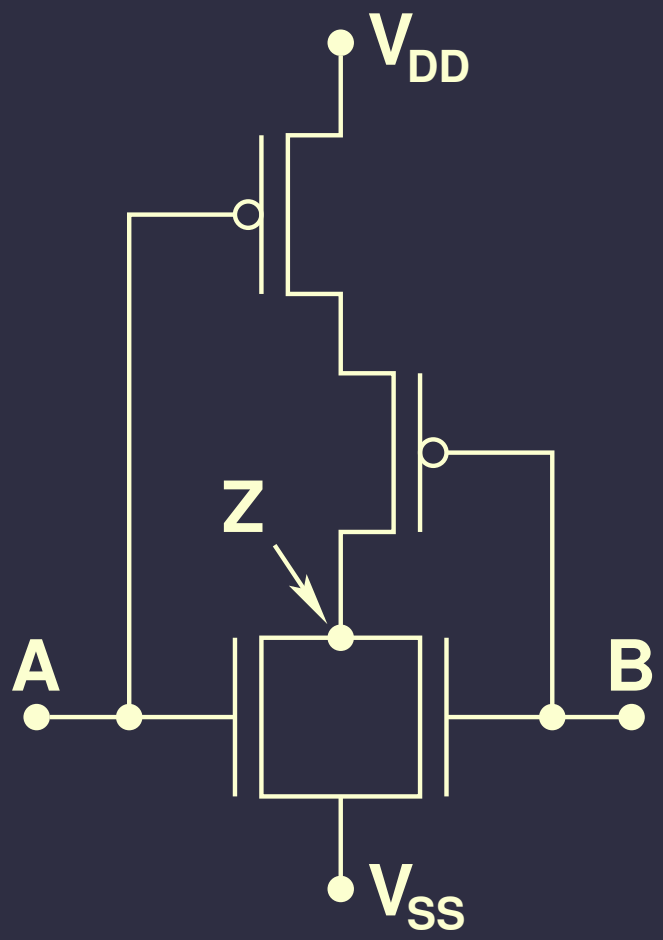
NOR operation



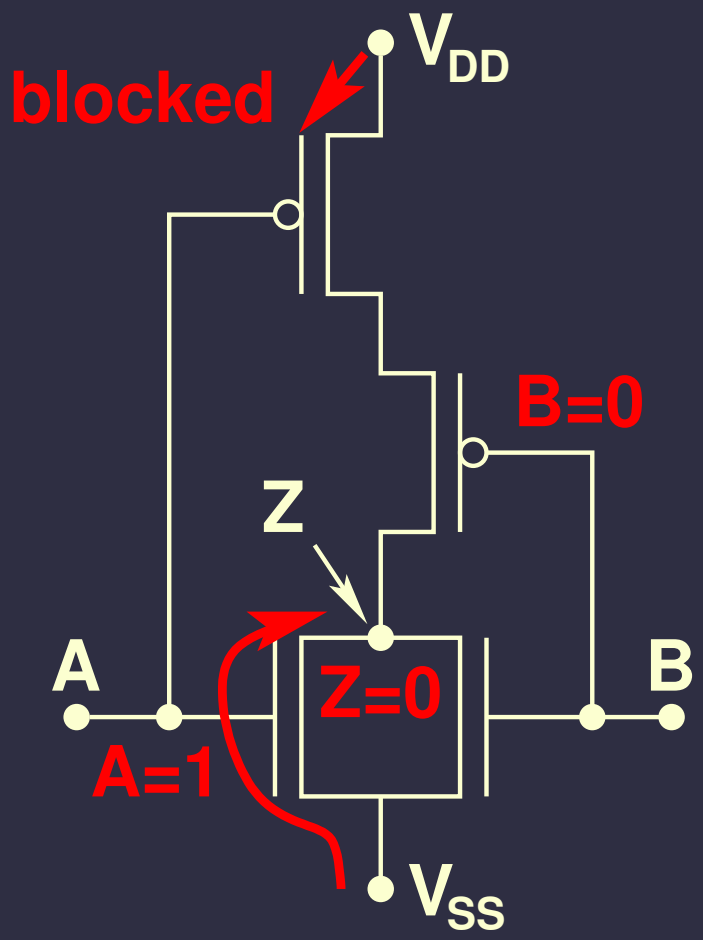
NOR operation



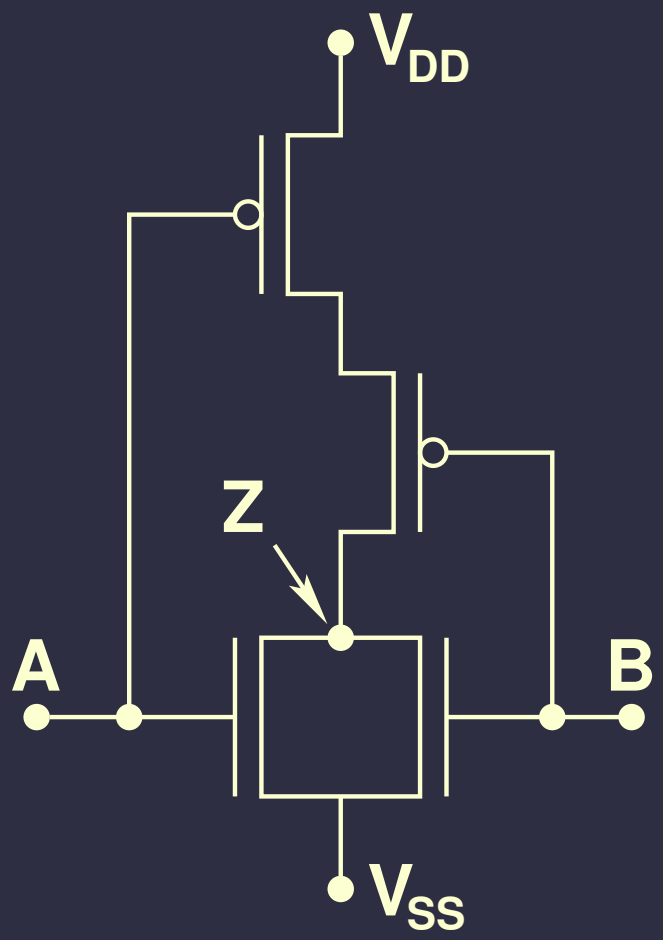
NOR operation



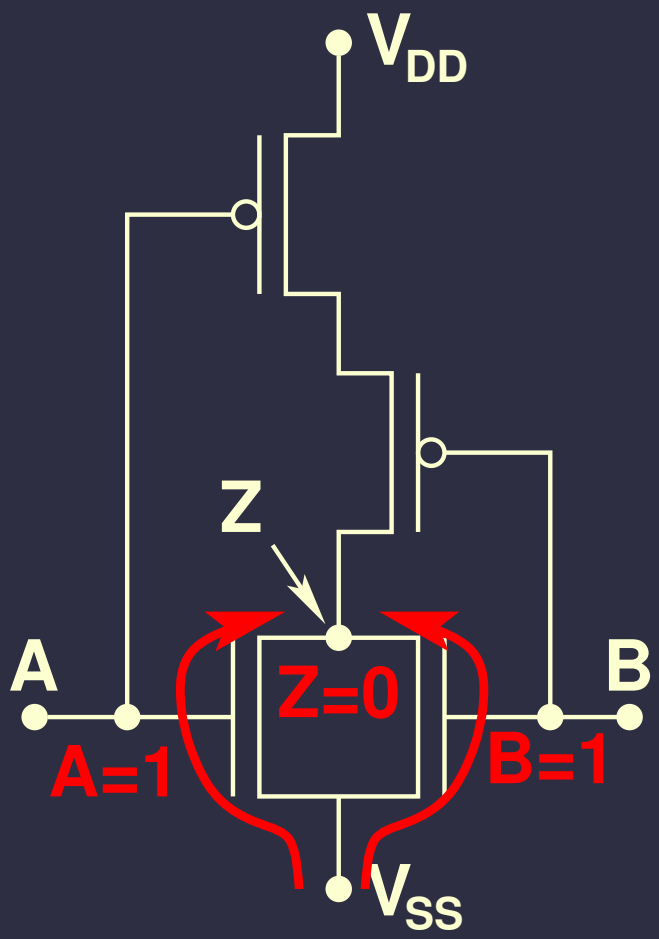
NOR operation



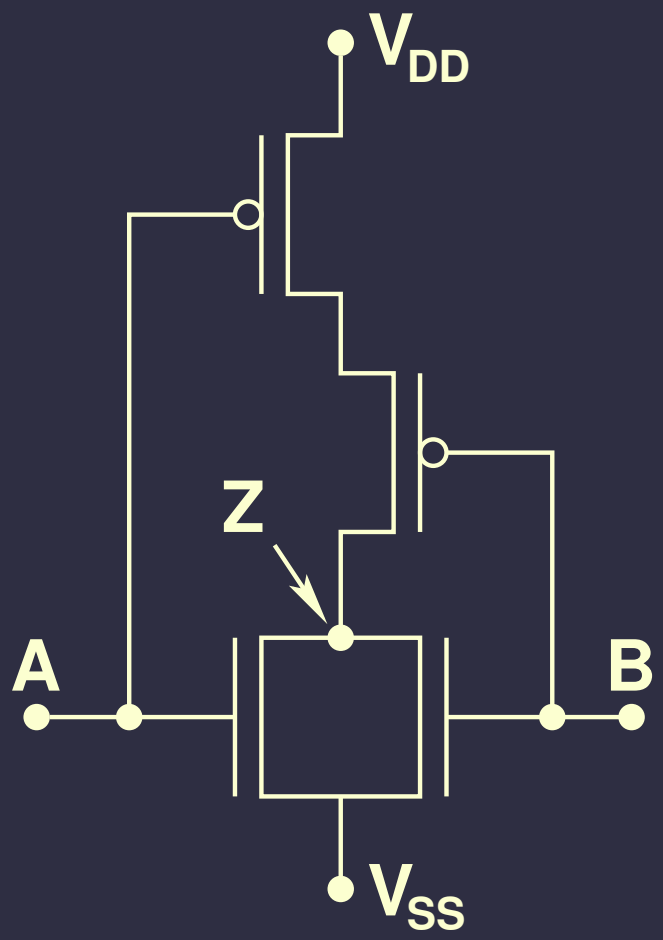
NOR operation



NOR operation



NOR operation



Section outline

1. Transistors in digital systems
   - Switches
   - CMOS
   - AND and OR are harder than NAND and NOR
   - Transmission gates

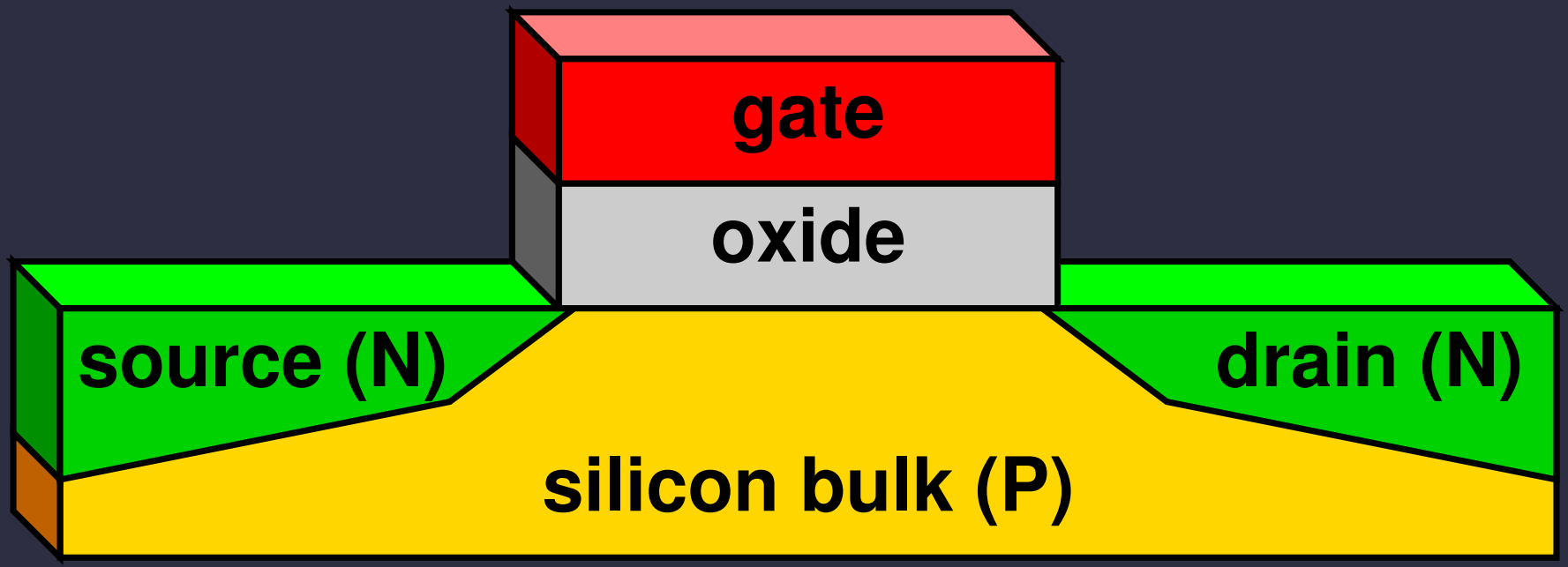
Non-ideality of NMOS/PMOS transistors

- Recall that NMOS transmits low values easily...
- ...transmits high values poorly
- PMOS transmits high values easily...
- ...transmits low values poorly

Non-ideality of NMOS/PMOS transistors

- 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 $V_{GS} < V_{TP}$

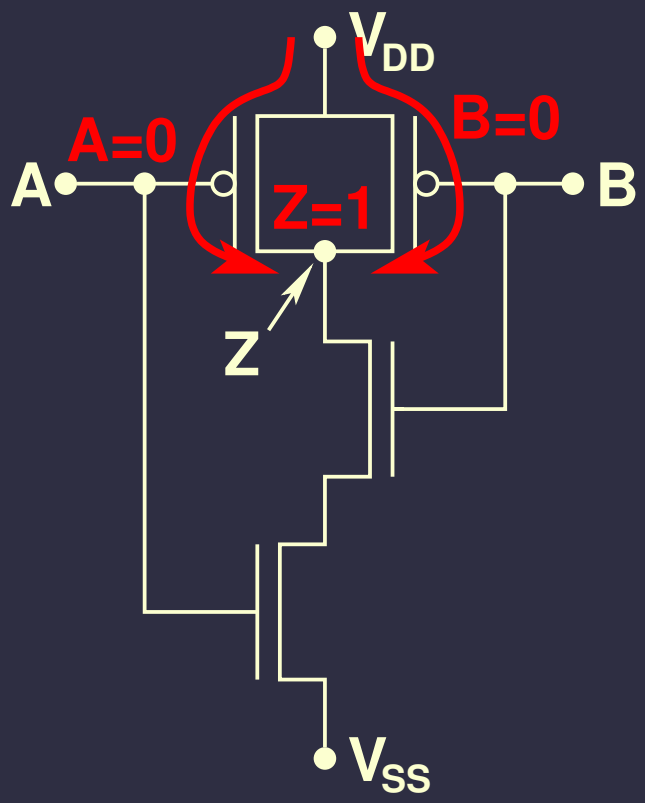
NMOS transistor



Non-ideality of NMOS/PMOS transistors

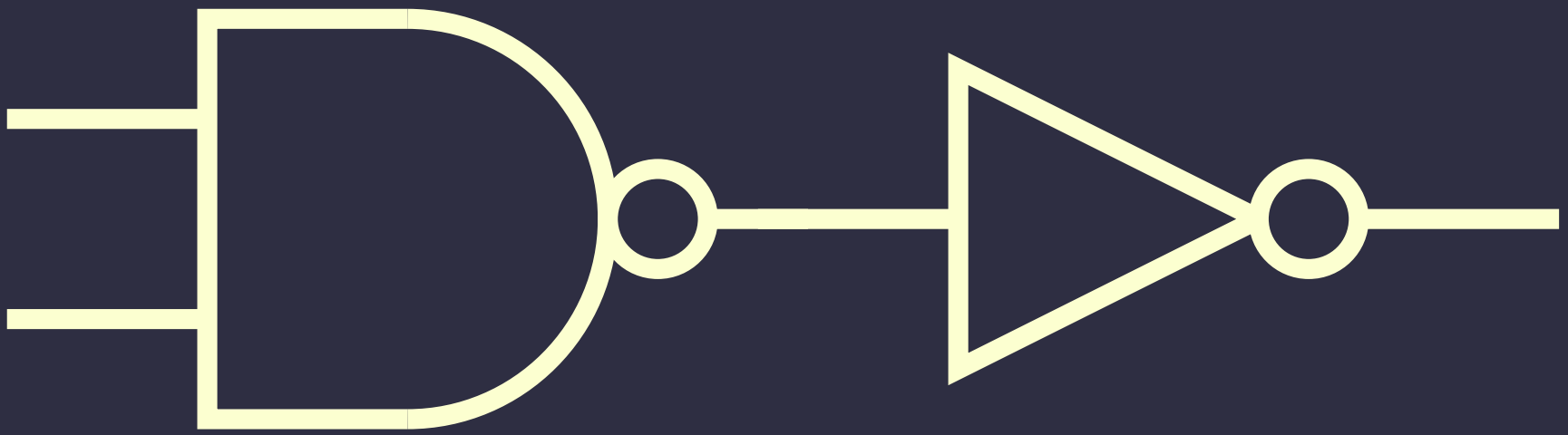
- If an NMOS transistor's input were V_{DD} (high), for $V_{GS} > V_{TN}$, the gate would require a higher voltage than V_{DD}
- If an PMOS transistor's input were V_{SS} (low), for $V_{GS} < V_{TP}$, the gate would require a lower voltage than V_{SS}

Implications of non-ideality



NAND/NOR easy to build in CMOS

Implications of non-ideality



AND/OR requires more area, power, time

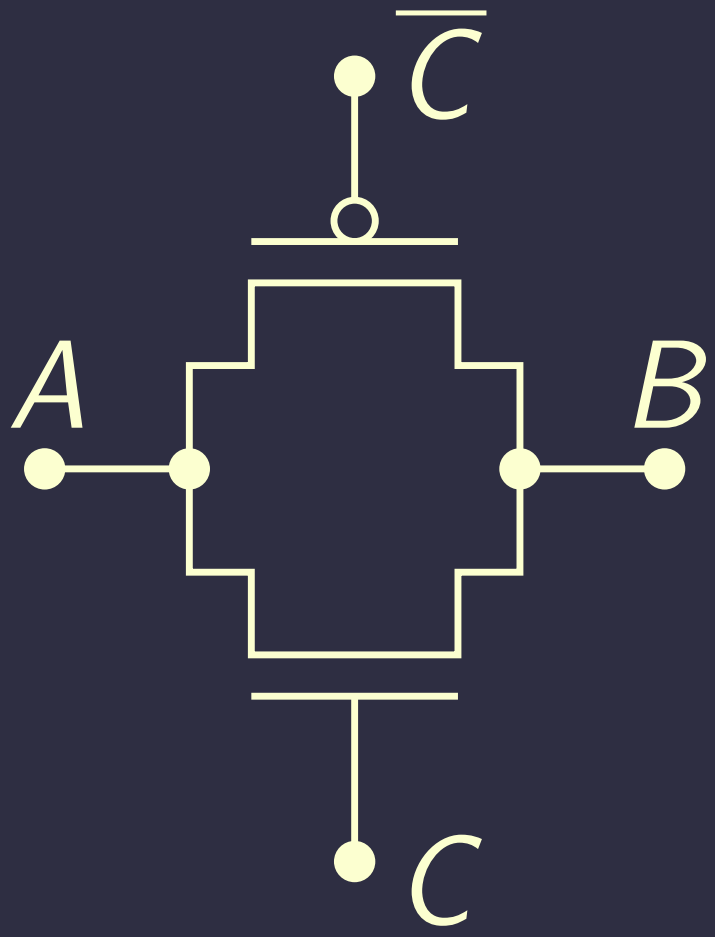
Section outline

1. Transistors in digital systems
   - Switches
   - CMOS
   - AND and OR are harder than NAND and NOR
   - Transmission gates

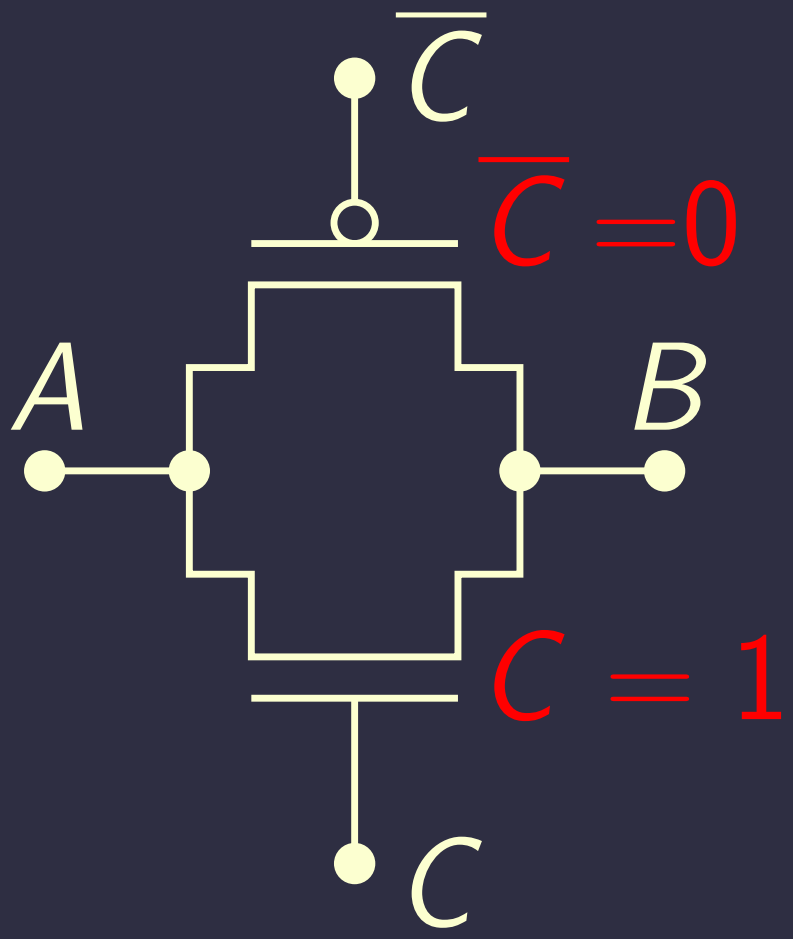
CMOS transmission gates (switches)

- 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

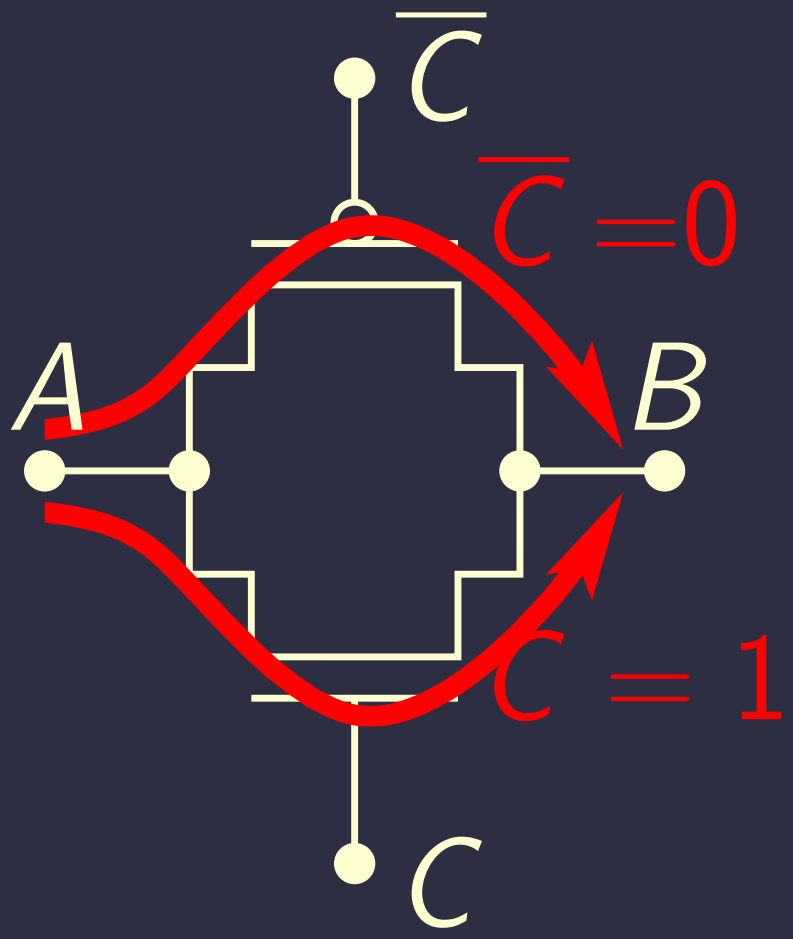
CMOS transmission gate (TG)



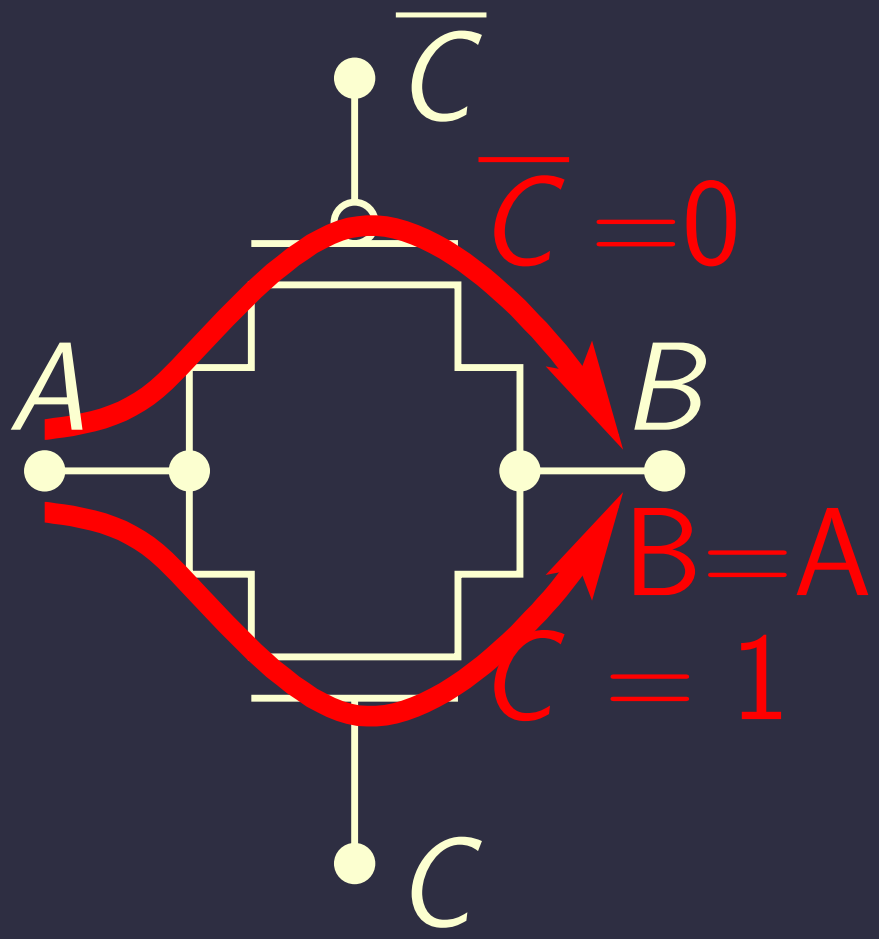
CMOS transmission gate (TG)



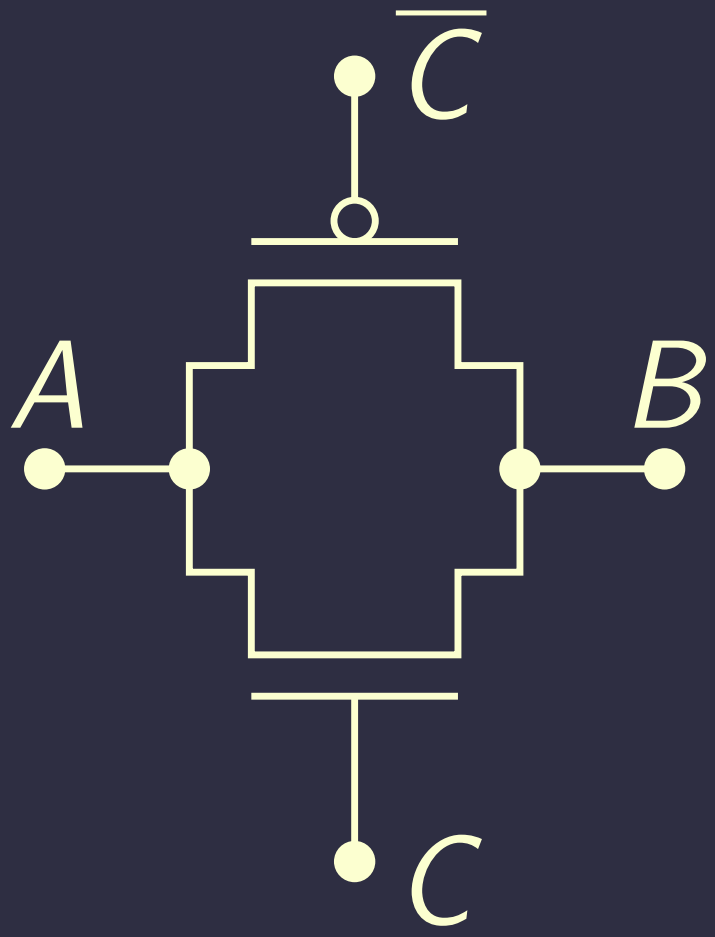
CMOS transmission gate (TG)



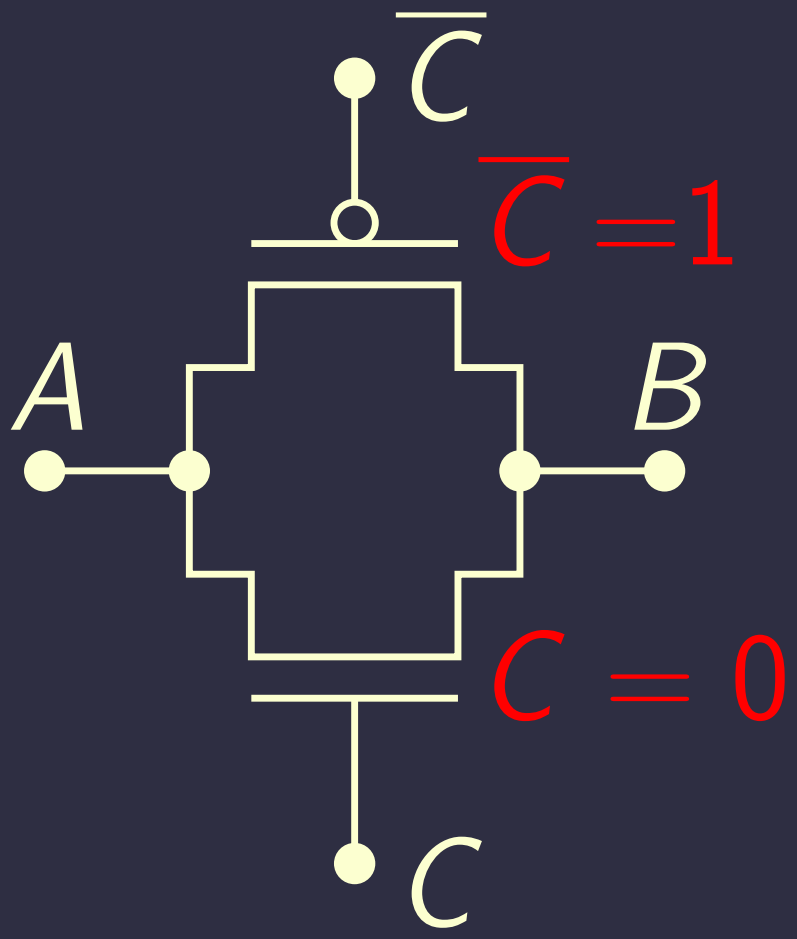
CMOS transmission gate (TG)



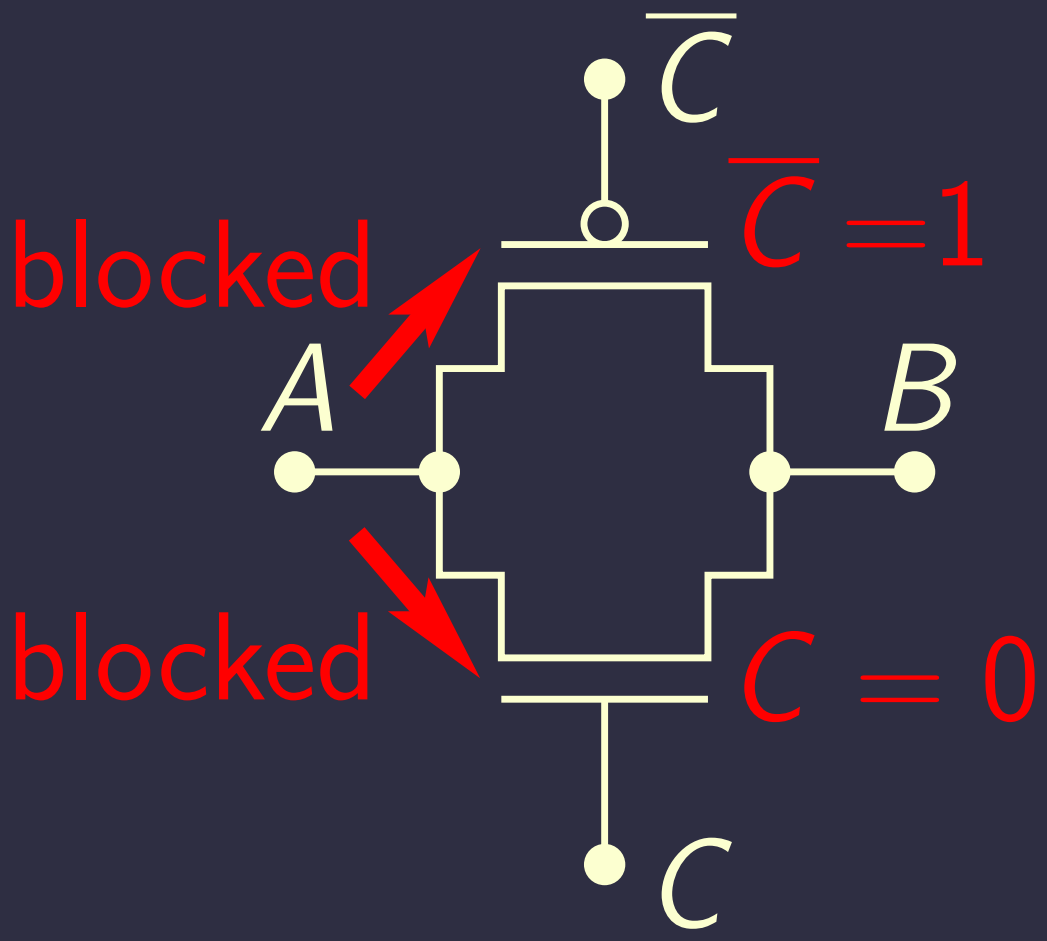
CMOS transmission gate (TG)



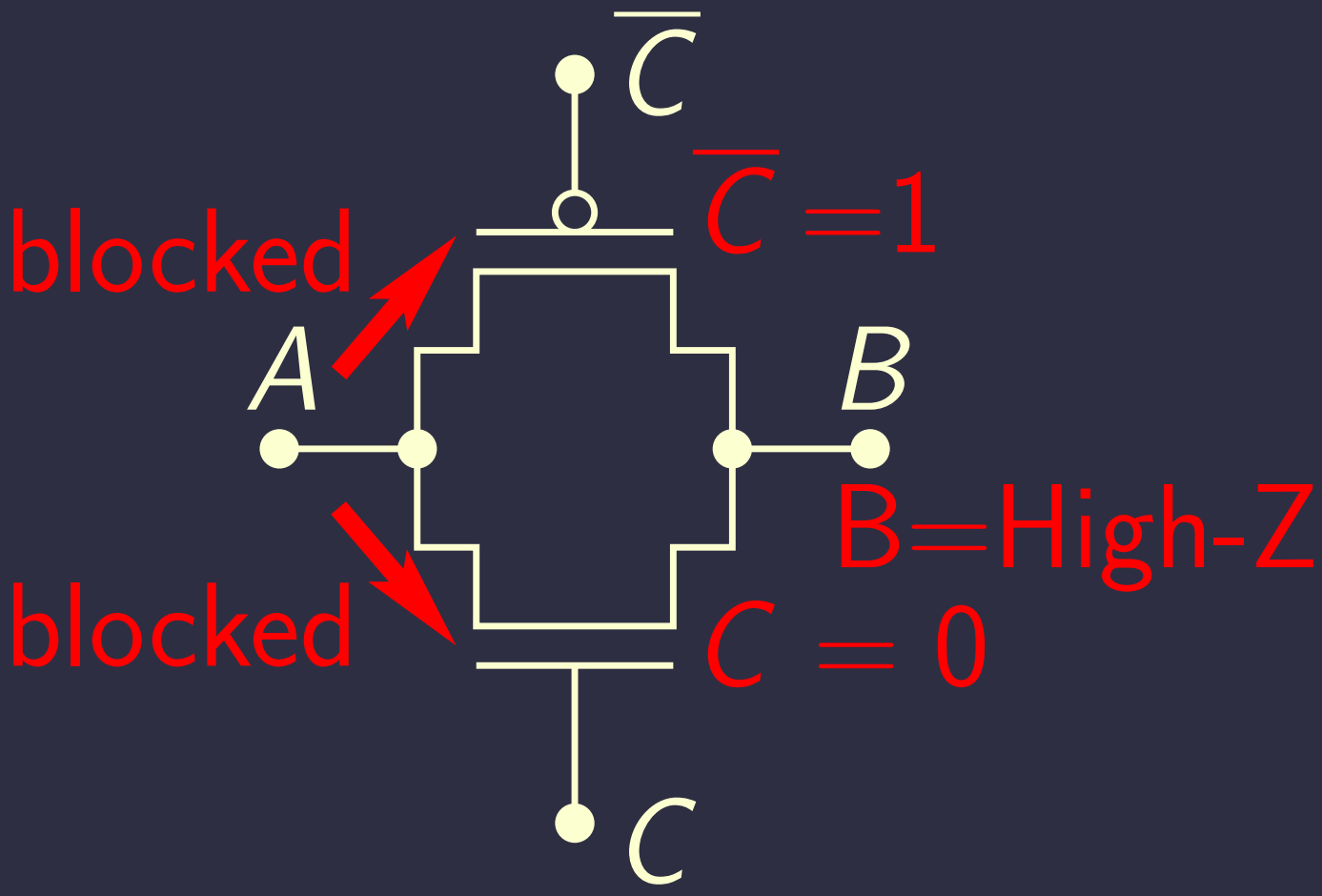
CMOS transmission gate (TG)



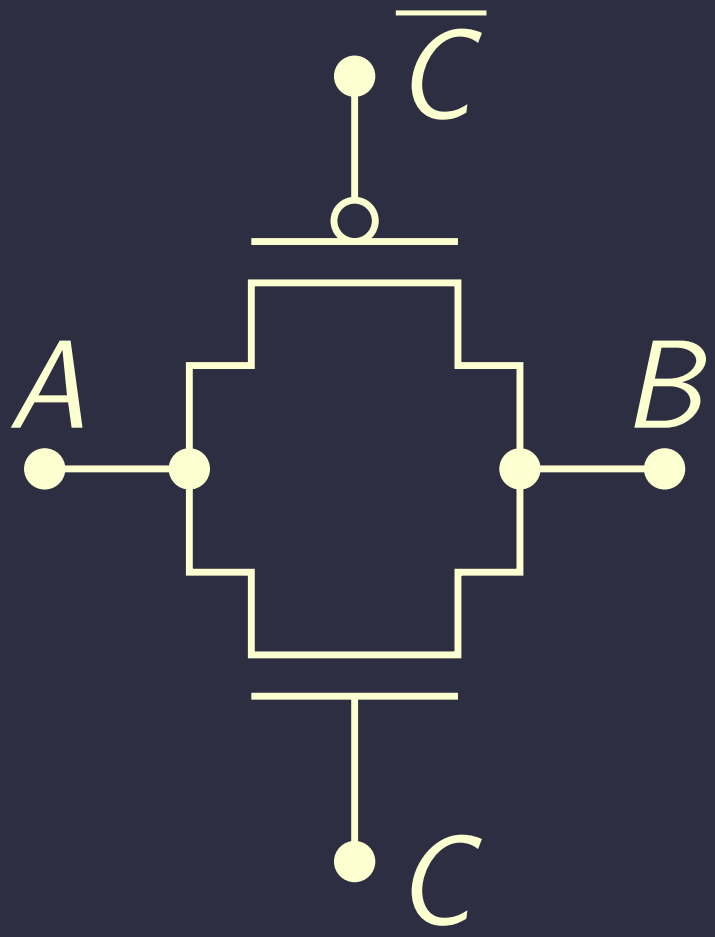
CMOS transmission gate (TG)



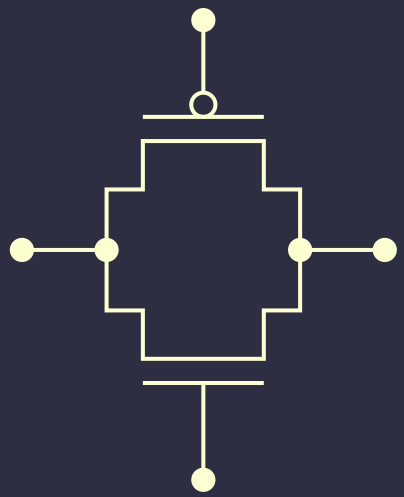
CMOS transmission gate (TG)

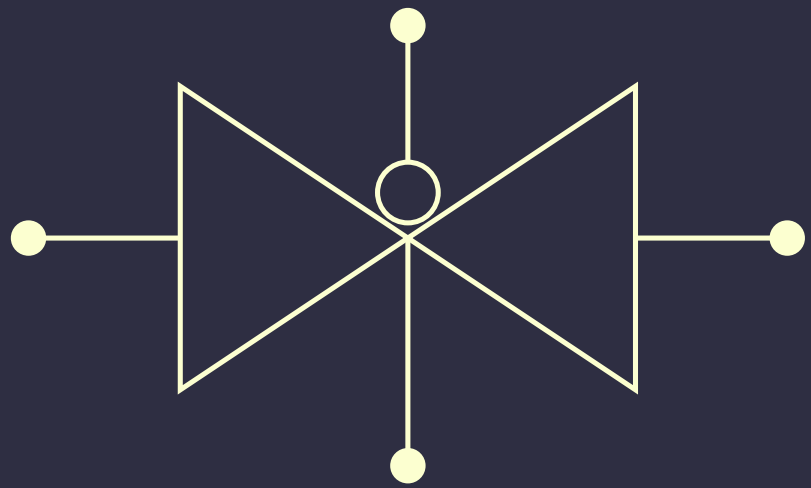


CMOS transmission gate (TG)



Other TG diagram





What can we build with TGs?

- Anything. . . try some examples.

Outline

Section outline

2. Homework
   - Assignment
   - Details

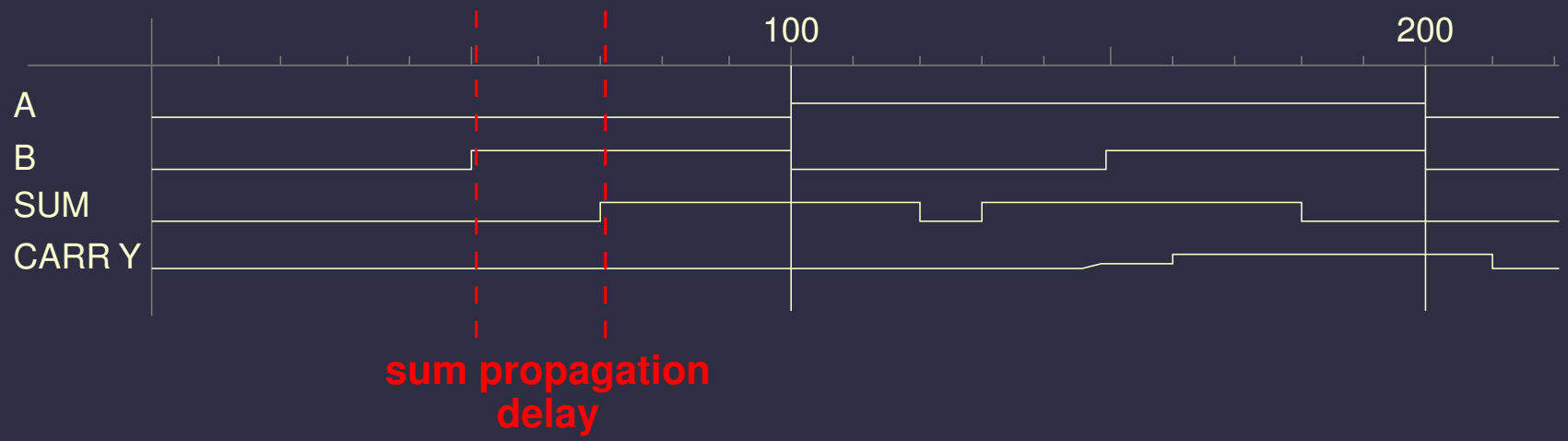
Lab one

- Walks you through the design and simulation of an exclusive-or (XOR) gate
- Due on 2 October
- Start early, especially if you are not familiar with Unix
- This lab requires a lot of tasks that are extremely easy the second time you do them, but slow and error-prone the first time through

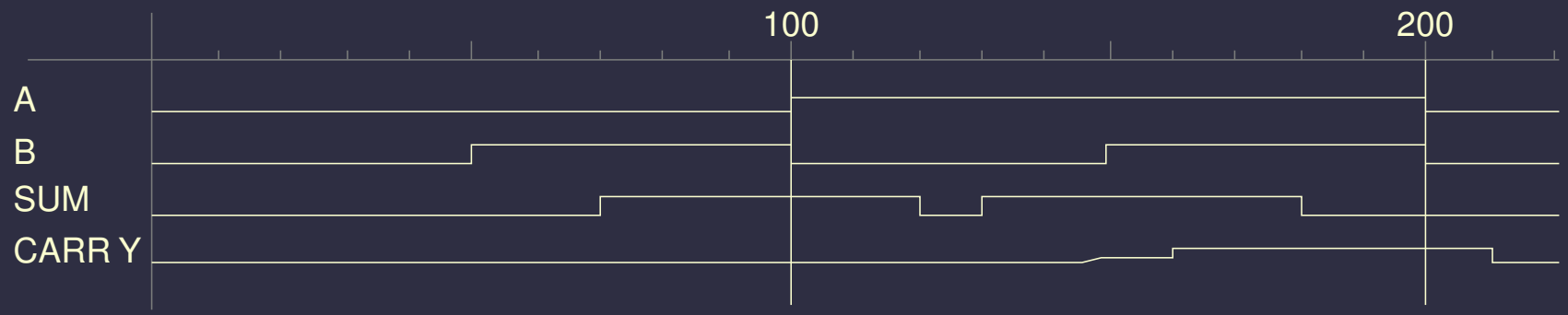
Section outline

2. Homework
    - Assignment
    - Details

Timing diagram for half adder

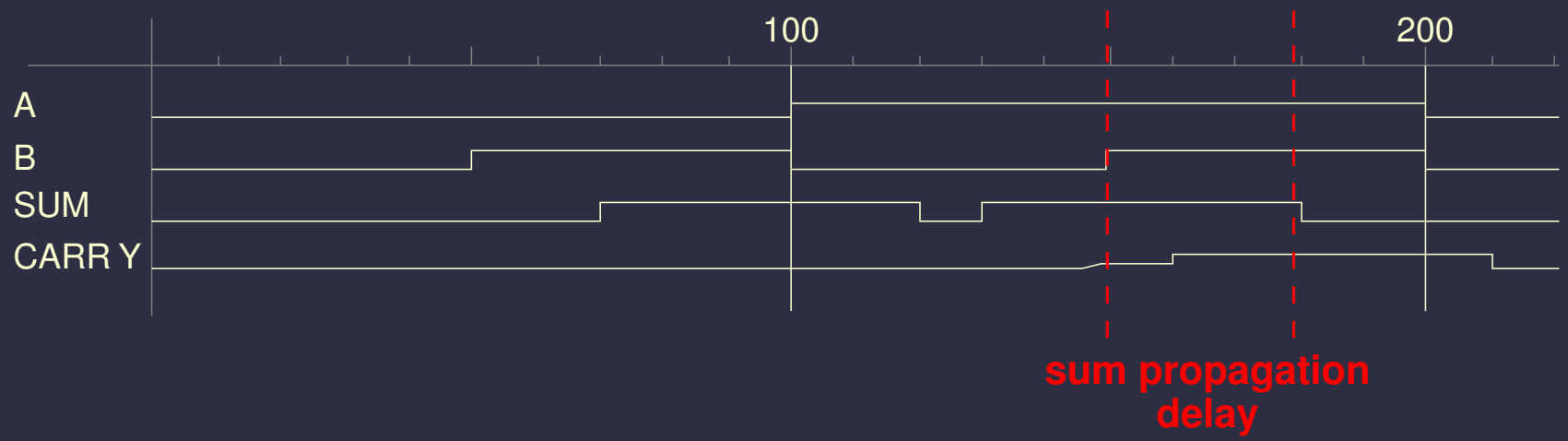


Timing diagram for half adder

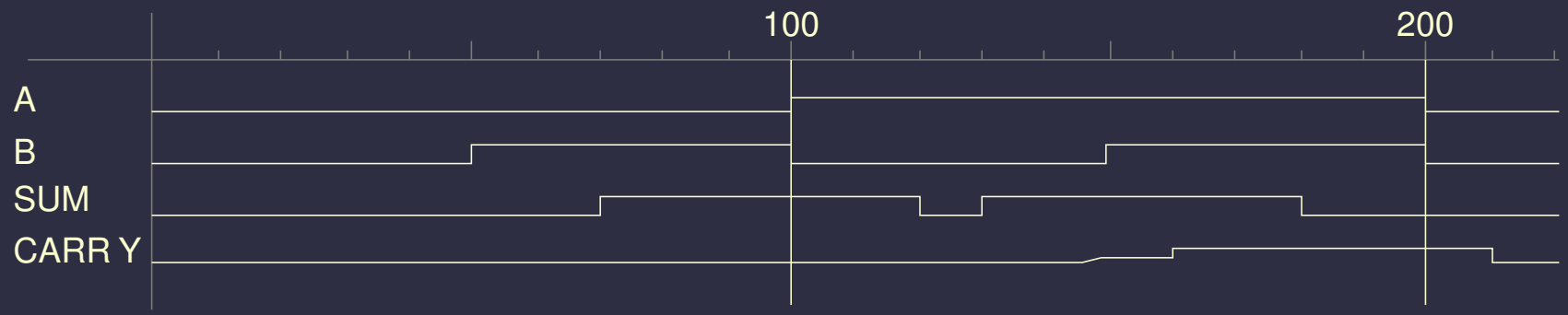


- Delay between input and output changes
- Delay is sensitive to circuit path

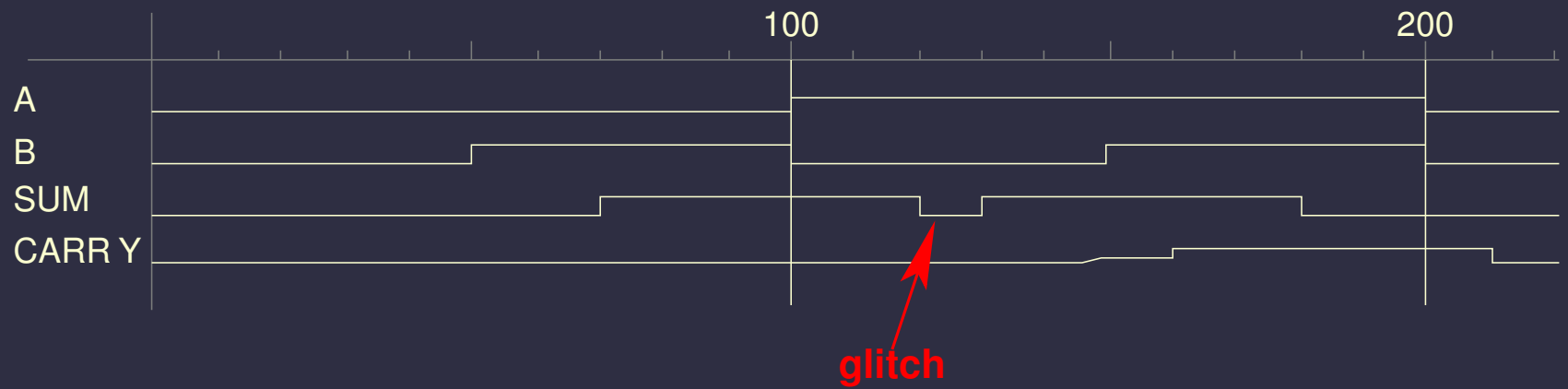
Timing diagram for half adder



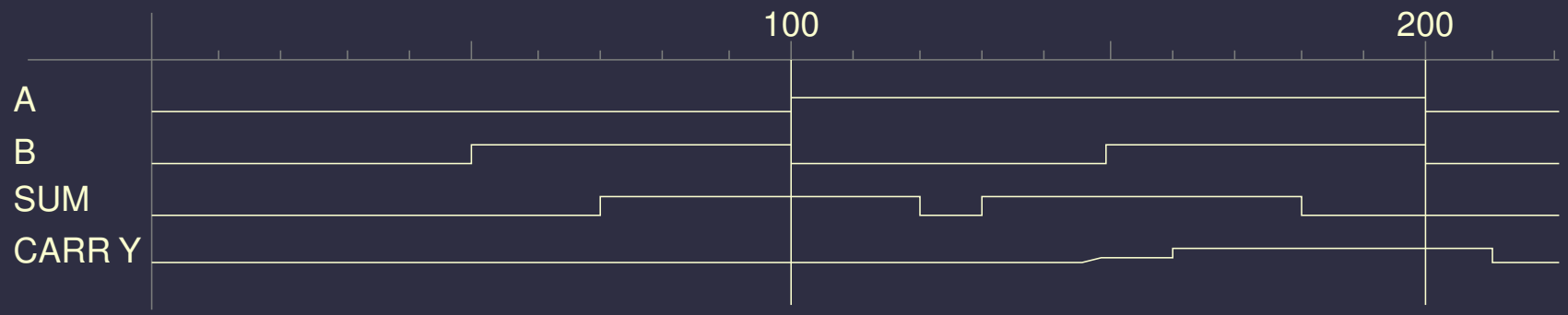
Timing diagram for half adder



Timing diagram for half adder

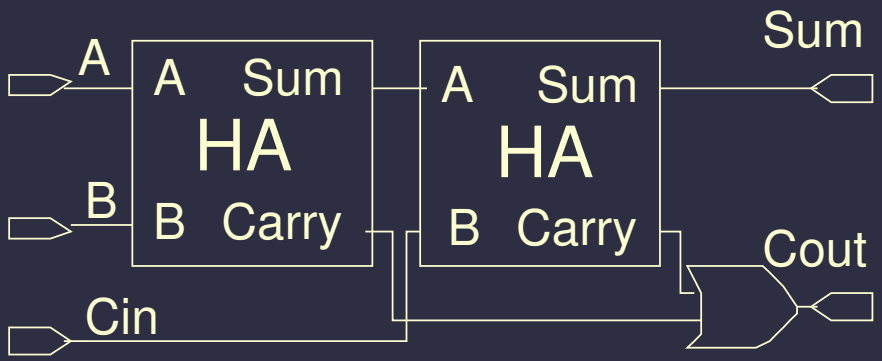


Timing diagram for half adder

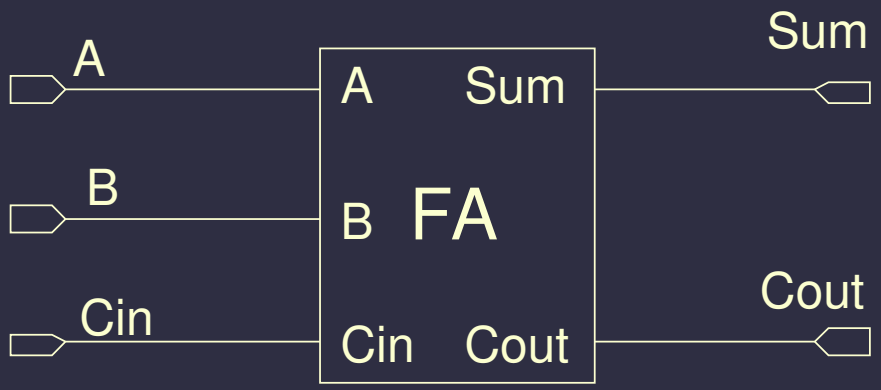


- Outputs may temporarily be incorrect before stabilizing
  - Glitches – caused by hazards

Block diagrams



Full adder composed of half adder blocks



Full adder block

- Structural organization of the design
- Hierarchical functional black boxes with input/output connections
- Concentrates on how the components are organized by wiring

Mentor Graphics tools introduction

- The Mentor Graphics CAD system has many components
- You will use a portion of the tools in this course
  - Falcon Design Framework
  - Design Architect for entering logic designs
  - Quicksim for simulating the designs
  - QuickHDL for entering and simulating the VHDL designs
- You will soon be working on lab 1, a Mentor Graphics tutorial

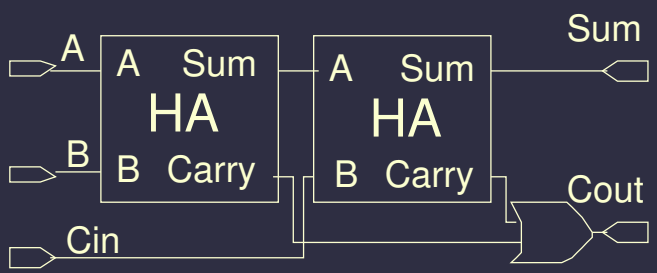
Mentor Graphics introduction

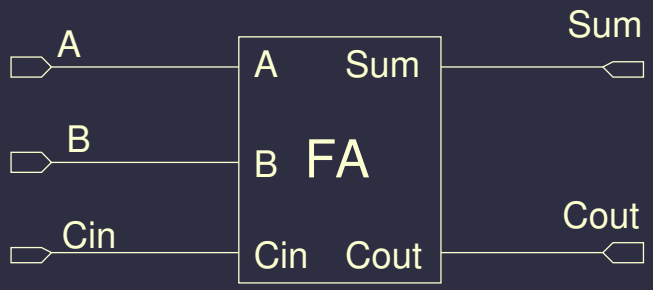
- Typing "source /vol/ece303/mgc.env" in the ECE filesystem will set up environment for ECE 303 labs
- Typing "dmgr" for Design Manager will open a window allowing several other Mentor Graphics to be run
- Mentor Graphics is not a single tool tool but a series of design tools that uses object oriented data representation to simplify the design process

Mentor Graphics introduction

- Data created in one tool (e.g., Design Architect) can be exported to another tool (e.g., Quicksim) for simulation
- A schematic is a diagram of a circuit
- Warning: Don't use OS commands to move directories or files
  - Design Manager needs to update other files when things are moved

Component definition





Data created by Design Architect is saved as components

- Models describing functional and graphical aspects
- Component data is composed of a schematic and a symbol
- A symbol is a graphical model with input and output pins
- A schematic is a functional model describing the relationship between output and input values

Viewpoint definition

- A viewpoint is a set of rules specifying a design's configuration
- Specifies the subset of data to use as input for a specific tool
- Allows a block to be described in different ways within different viewpoints
  - One can evaluate the impact of a low-level design decision on high-level design
  - For now, one viewpoint per design should be sufficient

Outline

Section outline

3. Two-level logic
   - Boolean algebra
   - Simplification
   - Karnaugh maps

Boolean algebra

- Set of elements, B
- Binary operators, { [AND, $\wedge$, *, $\cdot$], [OR, $\vee$, $+$] }
  - We'll prefer $\cdot$ and $+$
  - $\cdot$ frequently omitted
- Unary operator, [NOT, ', $\overline{o}$]

Axioms of Boolean algebra

$$\exists x, y \in B \text{ s.t. } x \neq y$$

closure	$\forall x, y \in B$	$xy \in B$
		$x + y \in B$
commutative laws	$\forall x, y \in B$	$xy = yx$
		$x + y = y + x$
identities	$0, 1 \in B, \forall x \in B$	$x1 = x$
		$x + 0 = x$

Axioms of Boolean algebra

$$\exists x, y \in B \text{ s.t. } x \neq y$$

distributive laws	$\forall x, y, z \in B$	$x + (yz) = (x + y)(x + z)$
		$x(y + z) = xy + xz$
compliment	$x \in B$	$x\overline{x} = 0$
		$x + \overline{x} = 1$

DeMorgan's laws

$$\overline{(a+b)} = \overline{a}\ \overline{b}$$
$$\overline{ab} = \overline{a} + \overline{b}$$
$$\overline{f(x_1, x_2, \ldots, x_n, \cdot, +)} = f(\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}, +, \cdot)$$

- Those xs could be functions
- Apply in stages
  - Top-down

DeMorgan's laws example

$$\overline{a + bc}$$

$$\overline{a} \cdot \overline{(bc)}$$

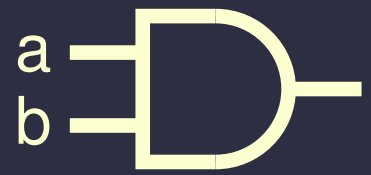
$$\overline{a} \cdot (\overline{b} + \overline{c})$$

Representations of Boolean functions

- Truth table
- Expression using only $\cdot$, $+$, and $'$
- Symbolic
- Karnaugh map
  - More useful as visualization and optimization tool

AND

a	b	a b
0	0	0
0	1	0
1	0	0
1	1	1



a *AND* b = a b

Will show Karnaugh map later

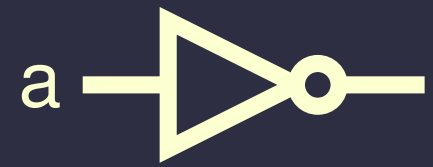
OR

a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1

$$a \; \mathit{OR} \; b = a + b$$

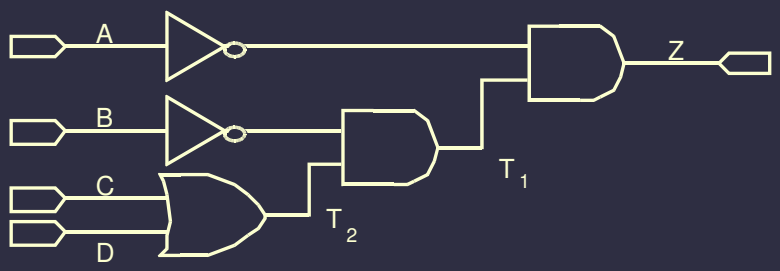
NOT

a	$\bar{a}$
0	1
1	0

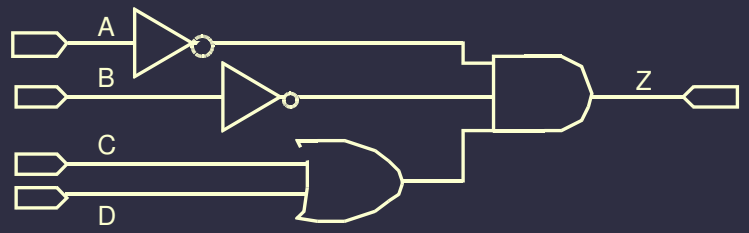


$$NOT\ \mathrm{a} = \bar{\mathrm{a}}$$

Different representations possible

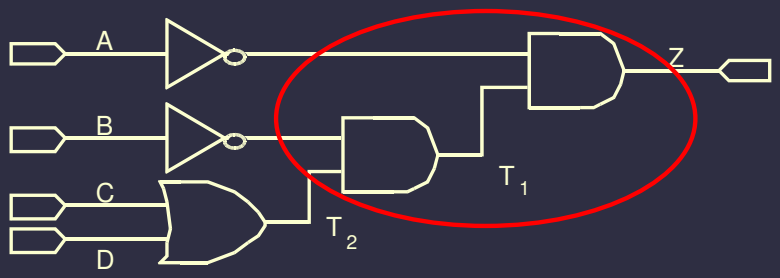


$$Z = ((C + D)\,\overline{B}\,)\,\overline{A}$$

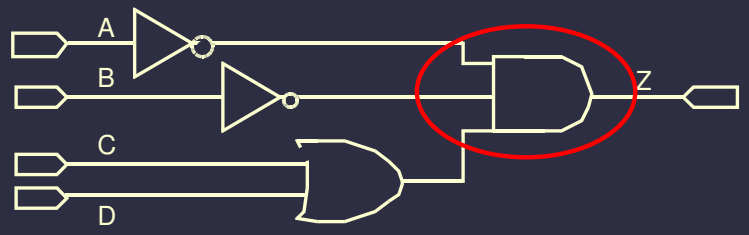


$$Z = (C + D)\,\overline{A}\;\overline{B}$$

Different representations possible



$$Z = ((C + D)\,\overline{B}\,)\,\overline{A}$$



$$Z = (C + D)\,\overline{A}\;\overline{B}$$

Section outline

3. Two-level logic
   - Boolean algebra
   - Simplification
   - Karnaugh maps

Simplifying logic functions

- Minimize literal count (related to gate count, delay)
- Minimize gate count
- Minimize levels (delay)
- Trade off delay for area
  - Sometimes no real cost

Simplifying logic functions

- Minimize literal count (related to gate count, delay)
- Minimize gate count
- Minimize levels (delay)
- Trade off delay for area
  - Sometimes no real cost

Proving theorems = simplification

Prove $XY + X\overline{Y} = X$

$$XY + X\overline{Y} = X(Y + \overline{Y}) \quad \text{distributive law}$$
$$X(Y + \overline{Y}) = X(1) \quad \text{complementary law}$$
$$X(1) = X \quad \text{identity law}$$

Proving theorems = simplification

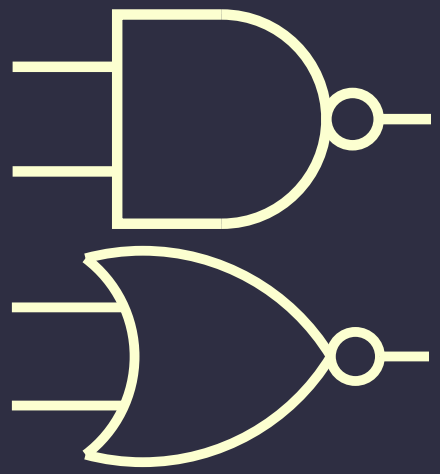
Prove $X + XY = X$

$$X + XY = X1 + XY \quad \text{identity law}$$
$$X1 + XY = X(1 + Y) \quad \text{distributive law}$$
$$X(1 + Y) = X1 \quad \text{identity law}$$
$$X1 = X \quad \text{identity law}$$

Literals

- Each appearance of a variable (complement) in expression
- Fewer literals usually implies simpler to implement
- E.g., $Z = A\overline{B}\,C + \overline{A}\,B + \overline{A}\,B\overline{C} + \overline{B}\,C$
  - Three variables, ten literals

NANDs and NORs



- Can be implemented in CMOS
  - More on this later
- X NAND $Y = \overline{XY}$
- X NOR $Y = \overline{X + Y}$
- Do we need inverters?

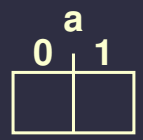
Section outline

3. Two-level logic
   - Boolean algebra
   - Simplification
   - Karnaugh maps

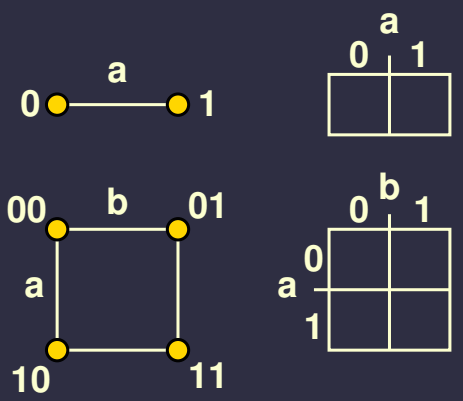
Karnaugh maps (K-maps)

- Fundamental attribute is adjacency
- Useful for logic synthesis
- Helps logic function visualization

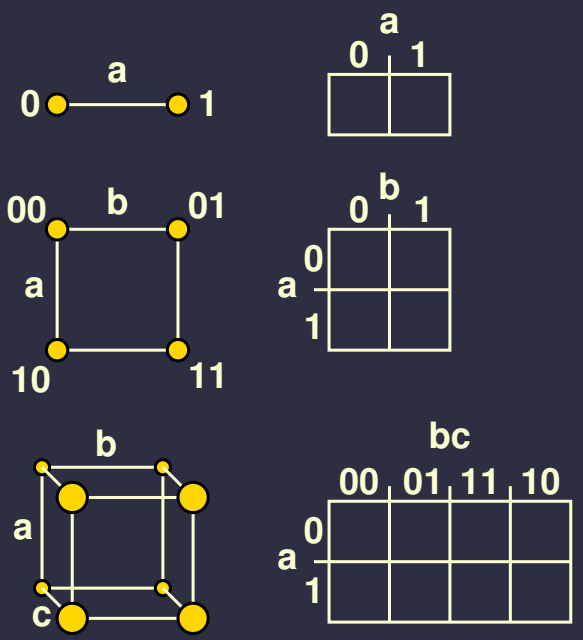
Karnaugh maps



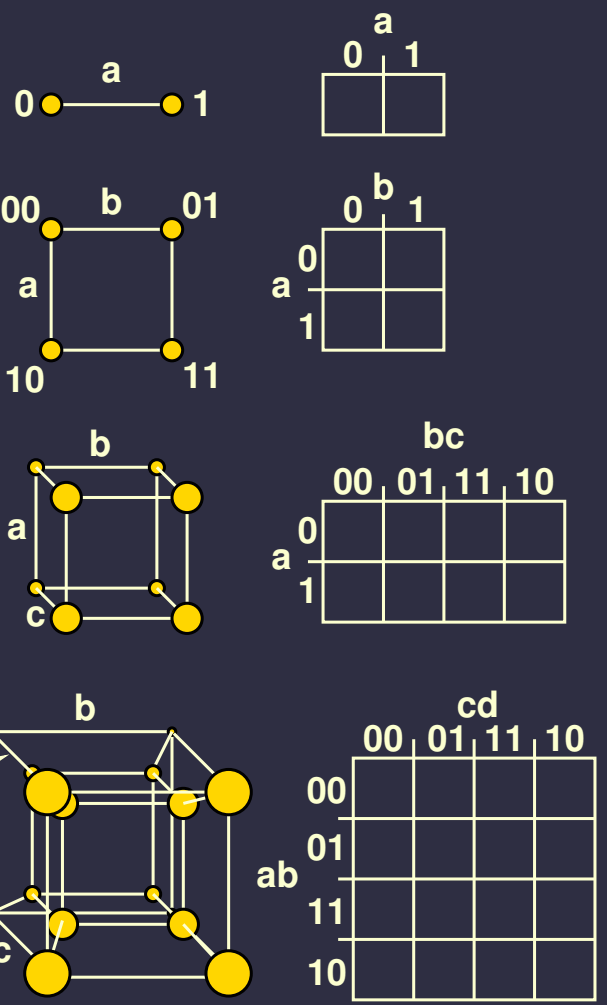
Karnaugh maps



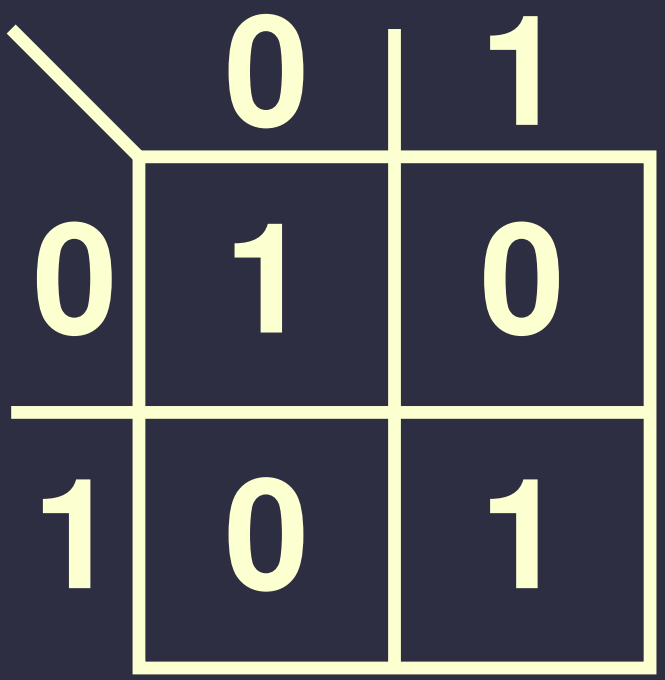
Karnaugh maps



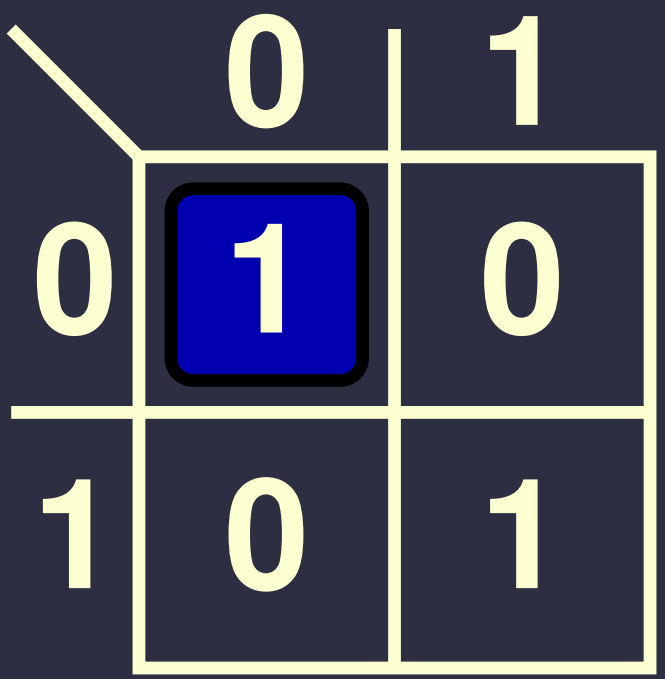
Karnaugh maps



Sum of products (SOP)

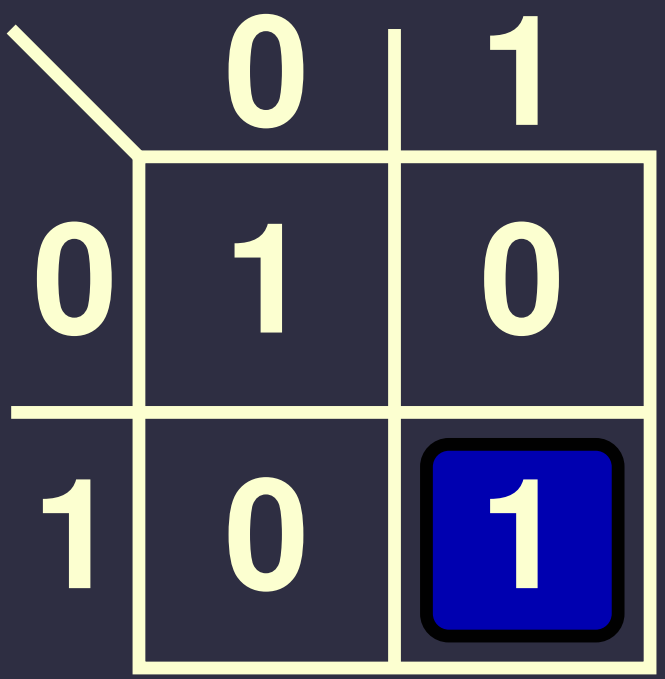


Sum of products (SOP)



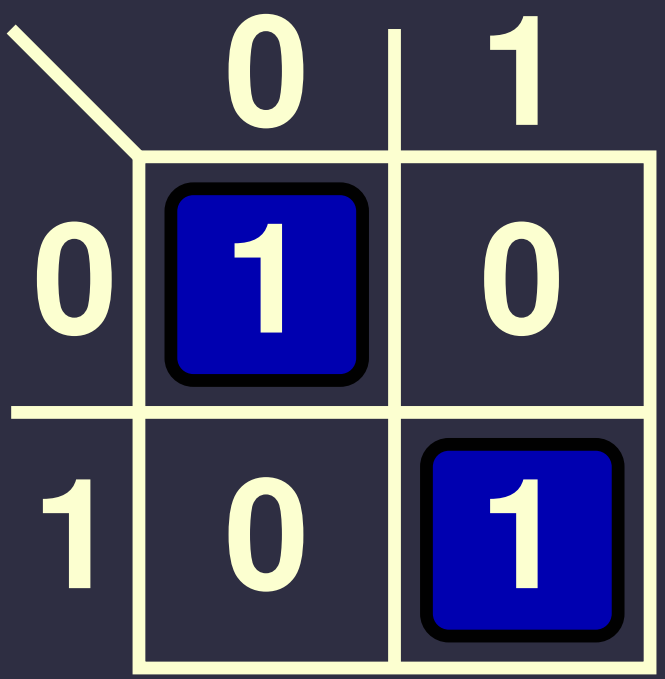
$(\overline{a}\ \overline{b})$

Sum of products (SOP)



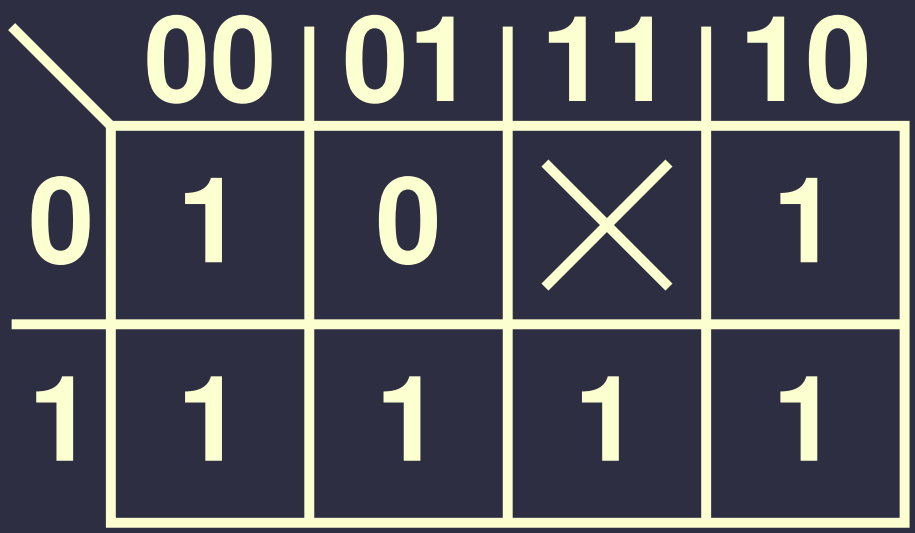
$(\overline{a}\ \overline{b})$ $(a\ b)$

Sum of products (SOP)

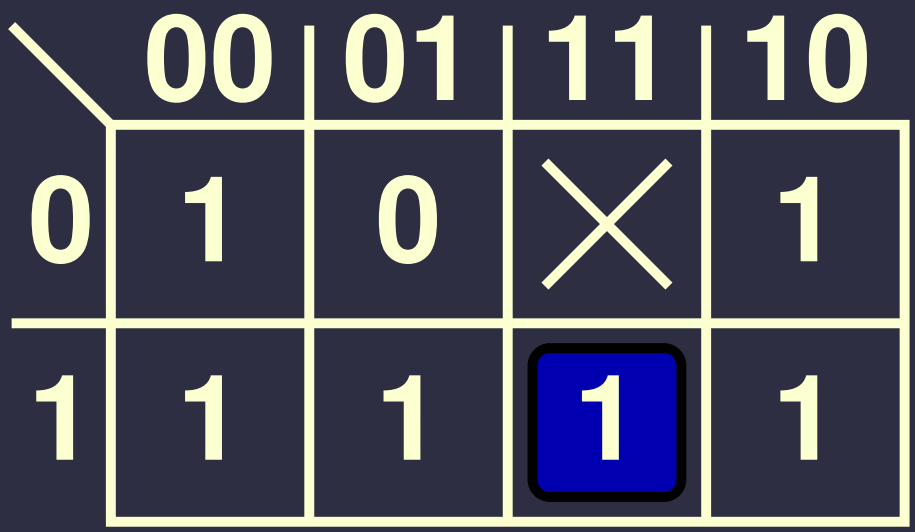


$$(\overline{a}\ \overline{b}) + (a\ b)$$

Implicants

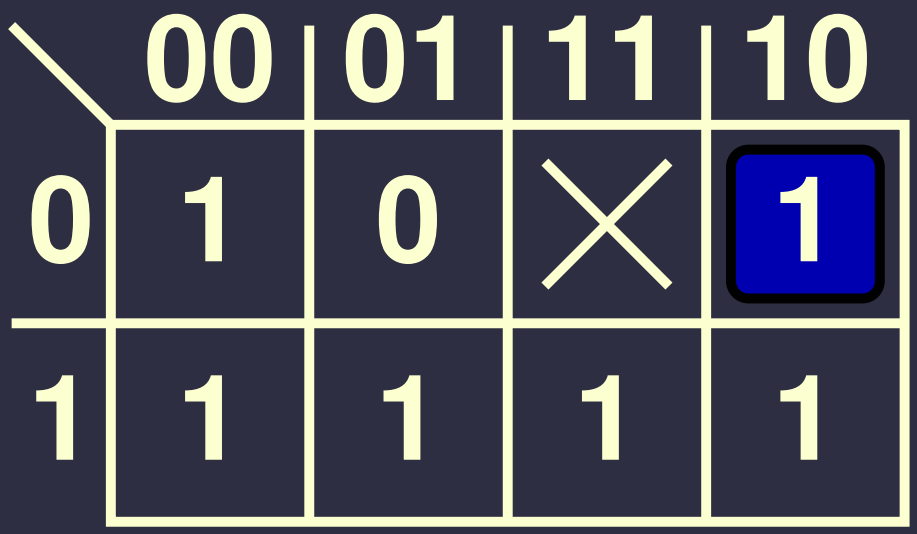


Implicants



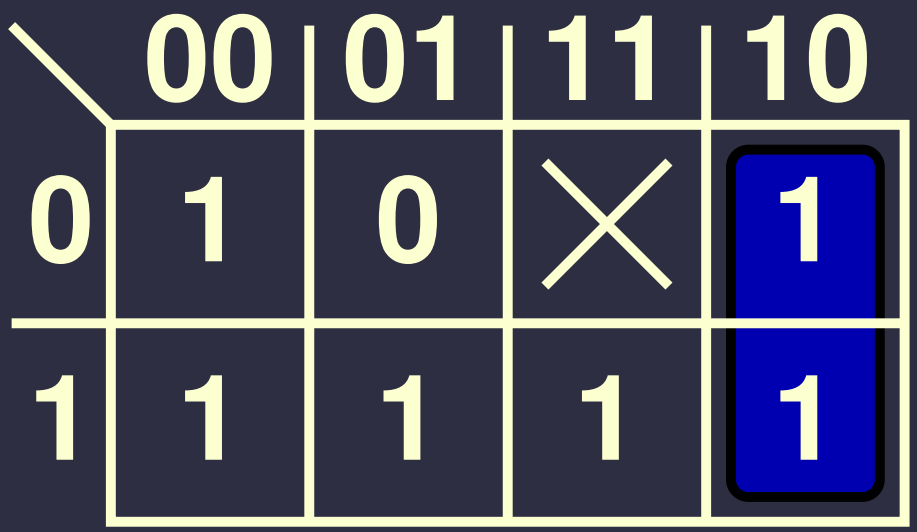
implicant

Implicants



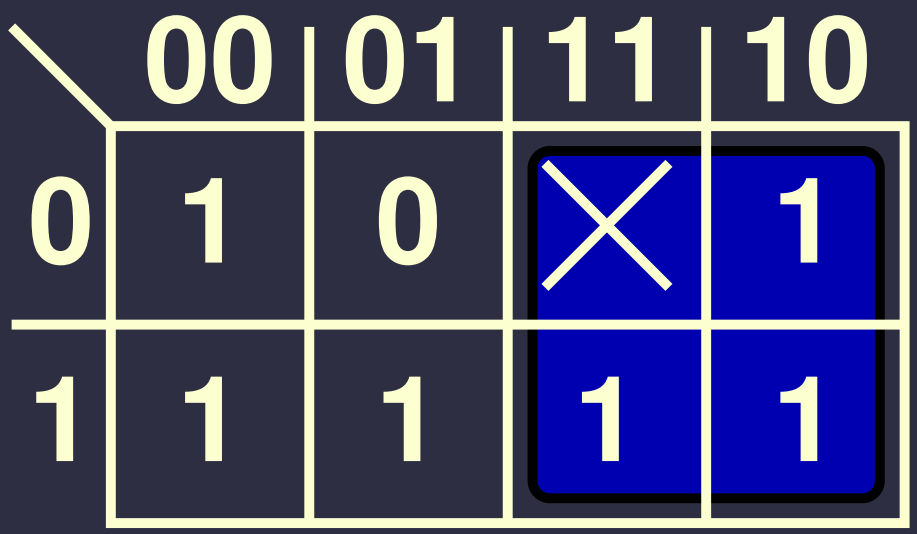
implicant

Implicants



implicant

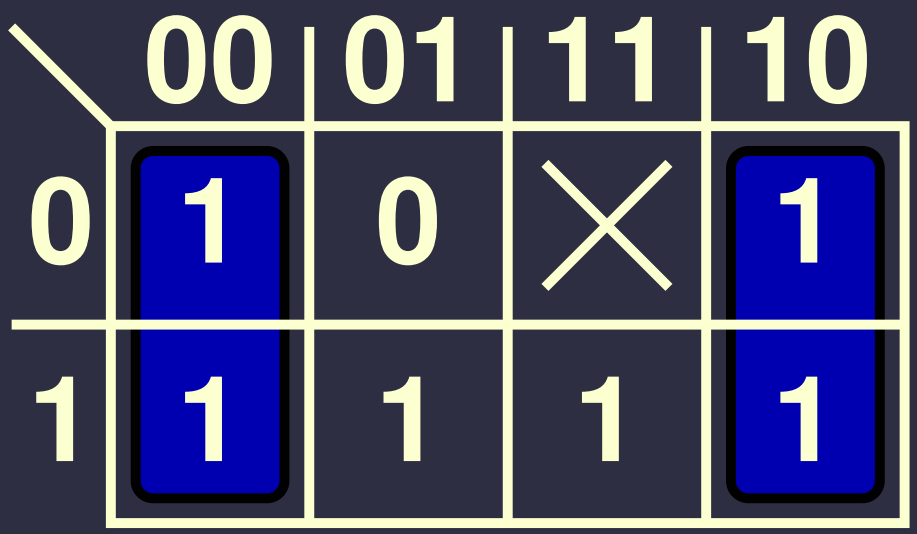
Implicants



prime implicant

Prime implicants are not covered by other implicants

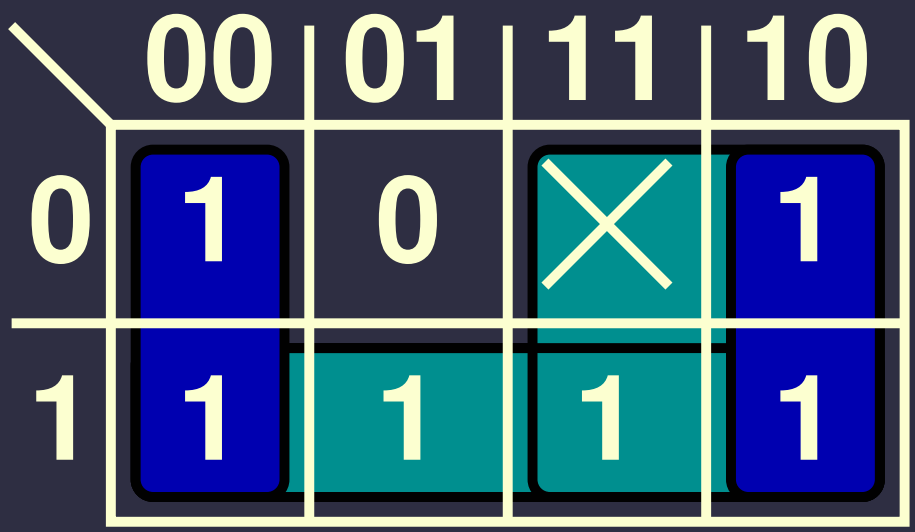
Implicants



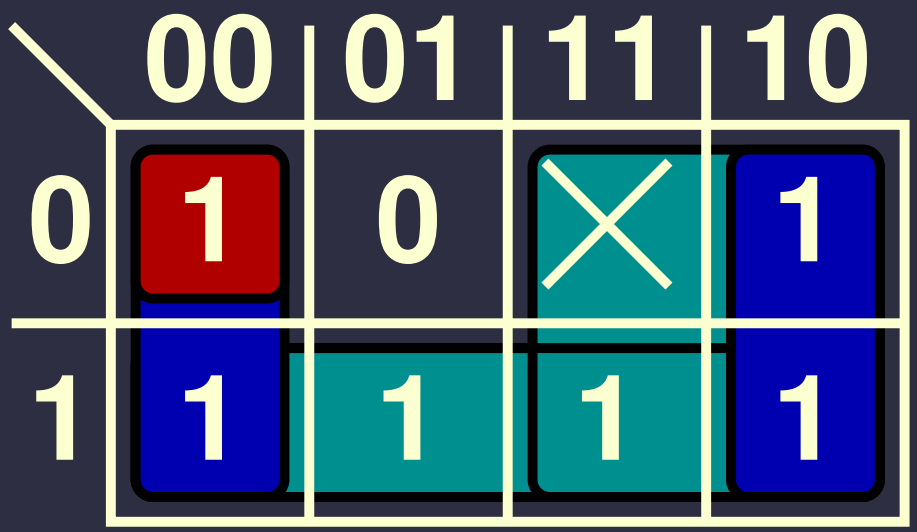
essential prime implicant

Essential prime implicants uniquely cover minterms

Implicants



Implicants



K-map example

- Minimize $f(a, b, c, d) = \sum(1, 3, 8, 9, 10, 11, 13)$

K-map example

- Minimize $f(a, b, c, d) = \sum(1, 3, 8, 9, 10, 11, 13)$
- f(a, b, c, d) = $a\ \overline{b} + \overline{b}\ \ d + a\ \overline{c}\ \ d$

K-map simplification technique

For all minterms

- Find maximal groupings of 1's and X's adjacent to that minterm.
- Remember to consider top/bottom row, left/right column, and corner adjacencies.
- These are the prime implicants.

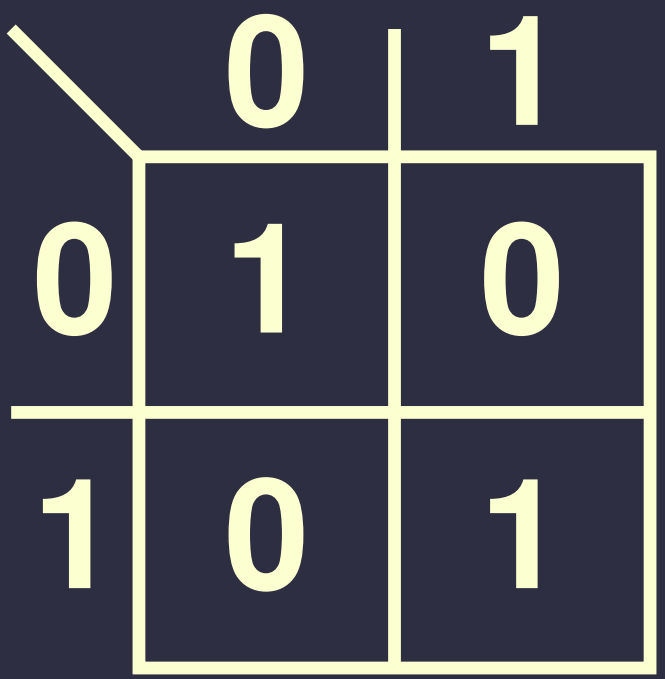
K-map simplification technique

- Revisit the 1's elements in the K-map.
- If covered by single prime implicant, the prime is essential, and participates in final cover.
- The 1's it covers do not need to be revisited.

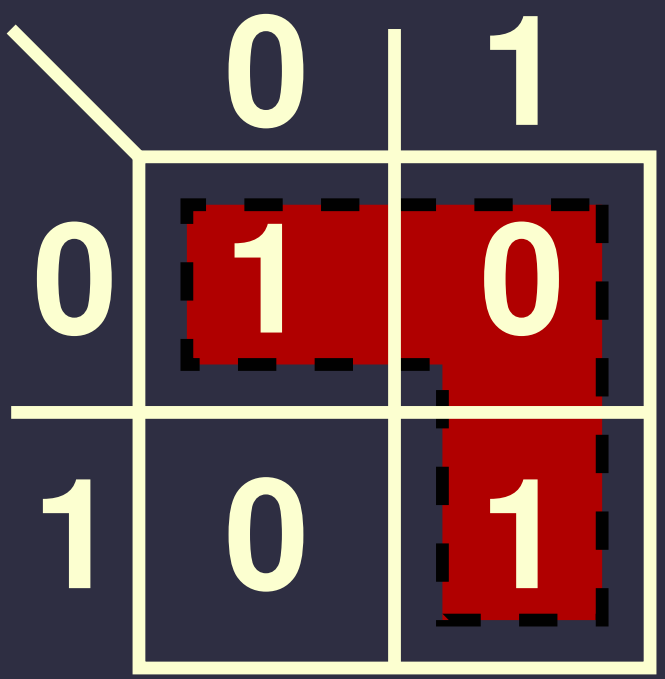
K-map simplification technique

- If there remain 1's not covered by essential prime implicants,
- Then select the smallest number of prime implicants that cover the remaining 1's.
- This can be difficult for complicated functions.
- Will present an algorithm for this in a future lecture.

Product of sums (POS)

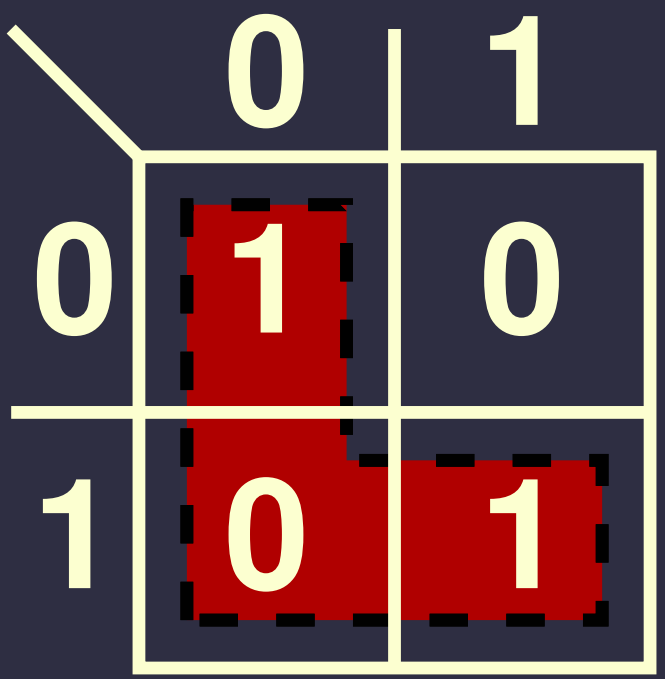


Product of sums (POS)



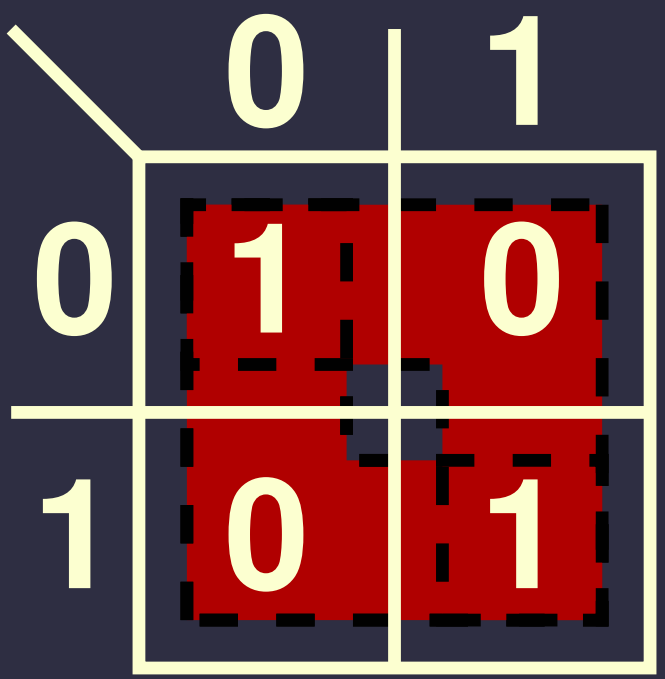
$(\overline{a} + b)$

Product of sums (POS)

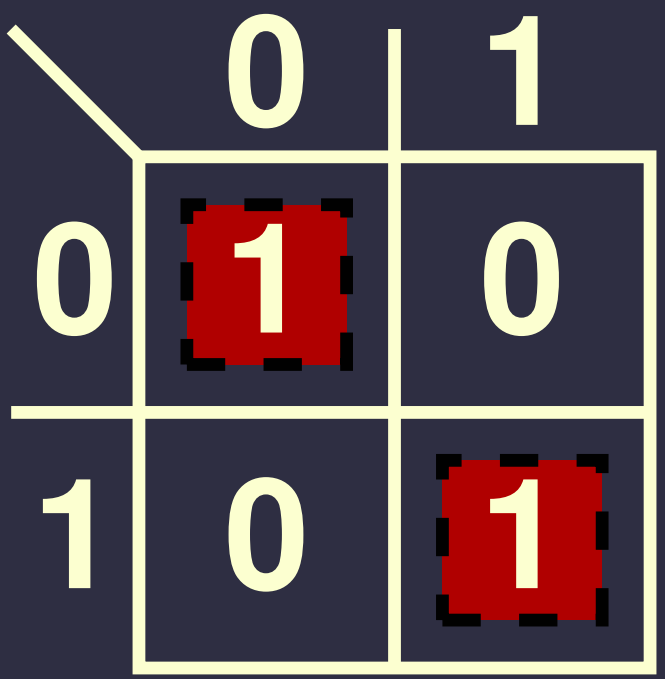


$$(a + \overline{b})$$

Product of sums (POS)



Product of sums (POS)



$$(\overline{a} + b) \cdot (a + \overline{b})$$

POS K-map techniques

- Direct reading by covering zeros and inverting variables

Or

- Invert function
- Do SOP
- Invert again
- Apply DeMorgan's laws

POS K-map example

- Minimize $f(a, b, c) = \prod(2, 4, 5, 6)$

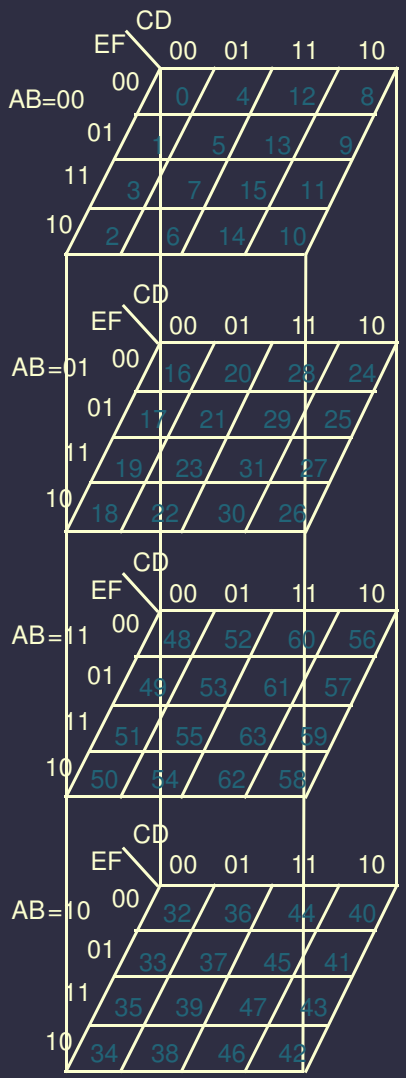
POS K-map example

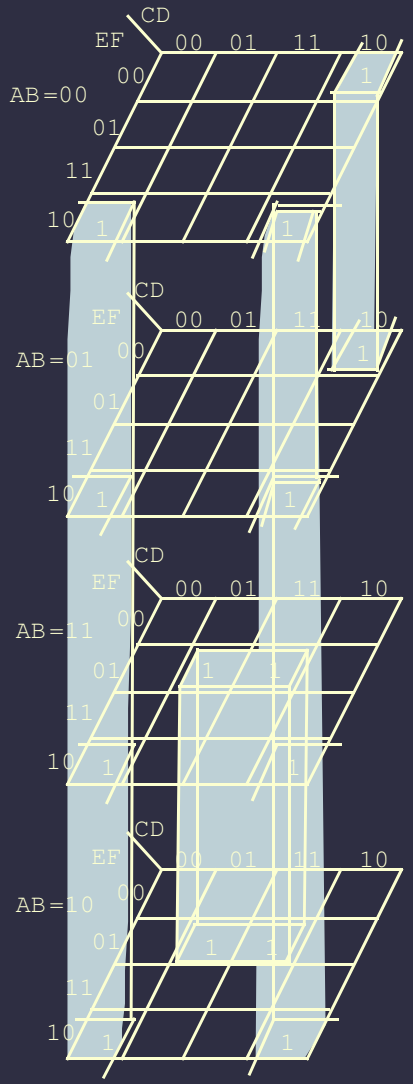
- Minimize $f(a, b, c) = \prod(2, 4, 5, 6)$
- $f(a, b, c) = (\overline{b} + c)(\overline{a} + b)$

Six-variable K-map example

$$z(a, b, c, d, e, f) = \sum(2, 8, 10, 18, 24, 26, 34, 37, 42, 45, 50, 53, 58, 61)$$

Six-variable K-map example





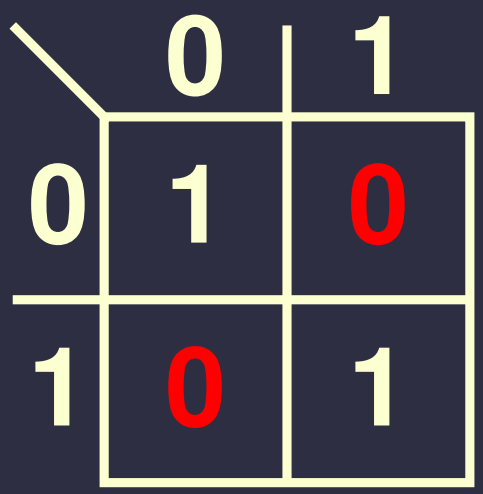
Six-variable K-map example

$$z(a,b,c,d,e,f) = \overline{d}\ e\,\overline{f} + ad\,\overline{e}\ f + \overline{a}\ C\,\overline{d}\ \overline{f}$$

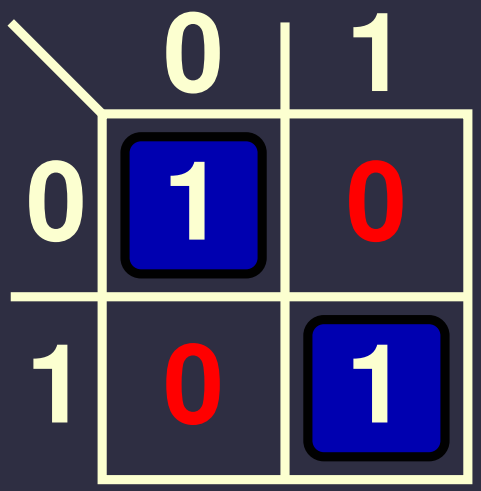
Don't Care logic

- All specified Boolean values are 0 or 1
- However, during design some values may be unspecified
  - Don't care values ($\times$)
- At $\times$s allow circuit optimization
  - Incompletely specified functions allow optimization

Don't Care values



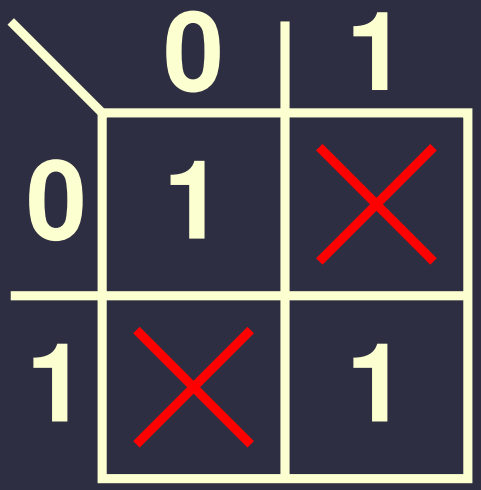
Don't Care values



Can let the undefined values be zero

- Correct...
- ...however, complicated
- $(\overline{a}\ \overline{b}) + (a\ b)$

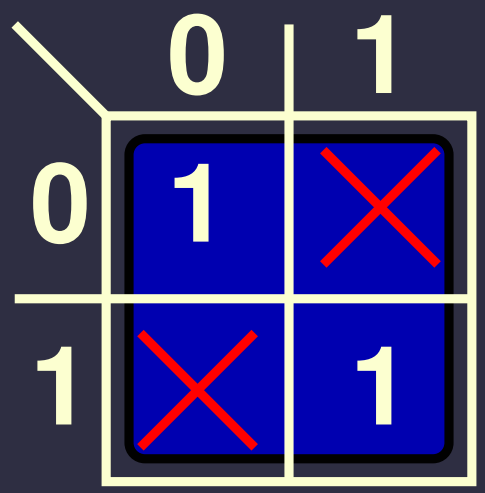
Don't Care values



Can let the undefined values be zero

- Correct...
- ...however, complicated
- $(\overline{a}\ \overline{b}) + (a\ b)$

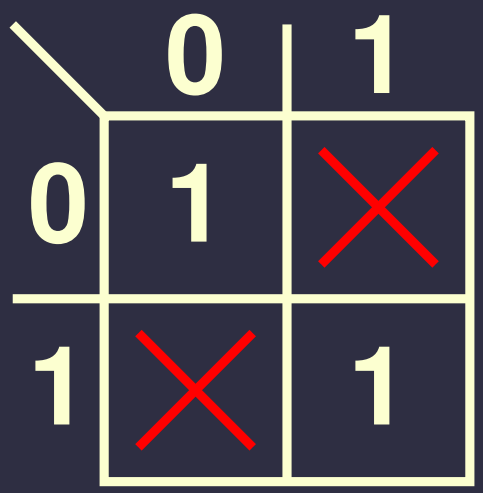
Don't Care values



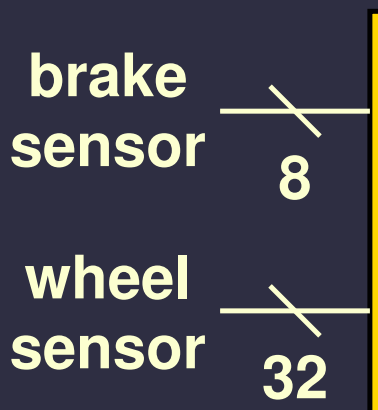
Instead, leave these values undefined ($\times$)

- Also called Don't Care values
- Allows any function implementing the specified values to be used
- E.g., could use $(\overline{a} \cdot \overline{b}) + (a \cdot b)$

Don't Care values



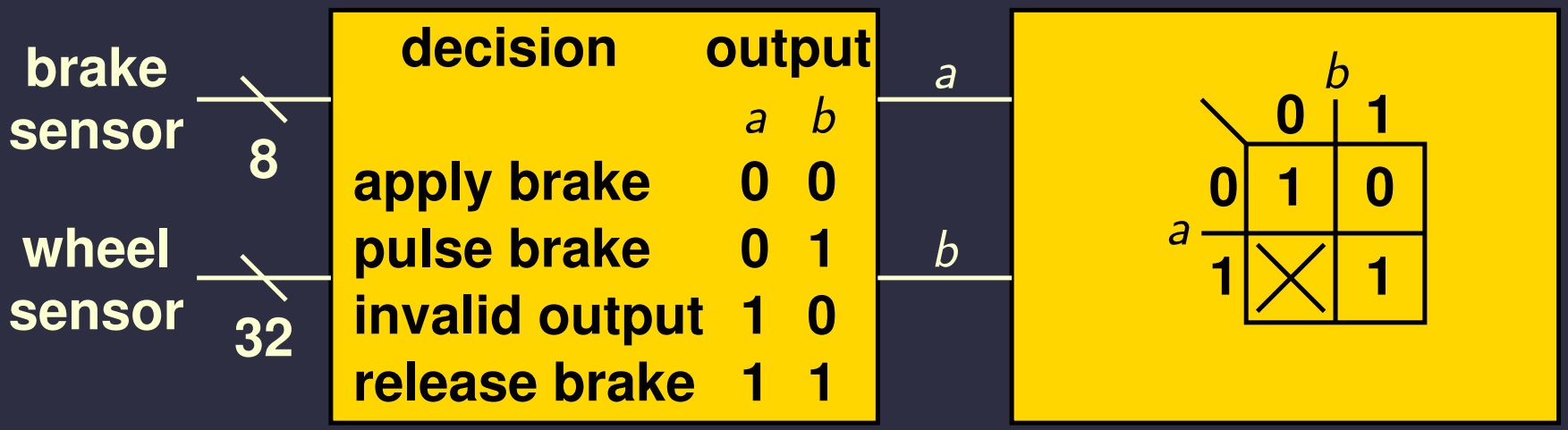
Satisfiability Don't Cares



decision	output	
	a	b
apply brake	0	0
pulse brake	0	1
invalid output	1	0
release brake	1	1

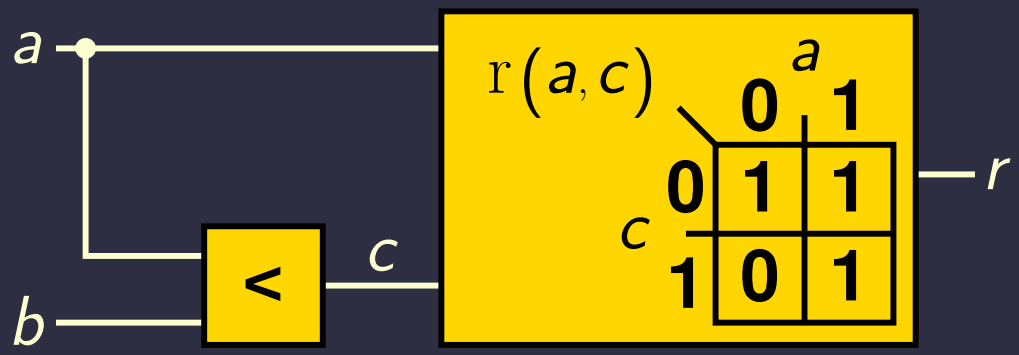
- Input can never occur
- This can happen within a circuit
- Some modules will not be capable of producing certain outputs

Satisfiability Don't Cares



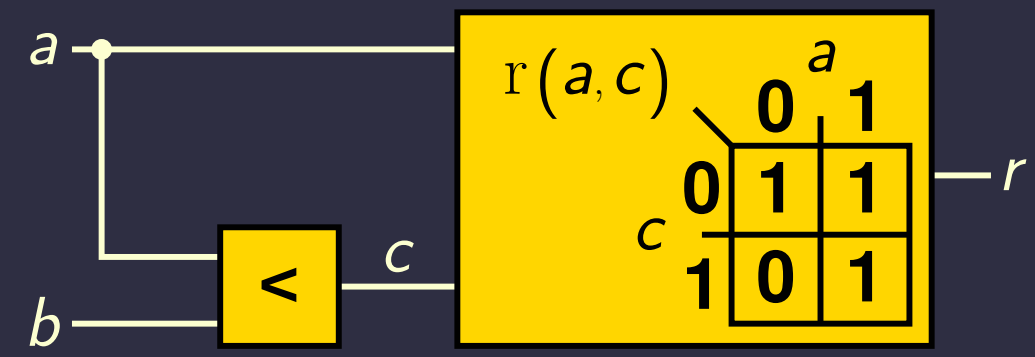
- Input can never occur
- This can happen within a circuit
- Some modules will not be capable of producing certain outputs

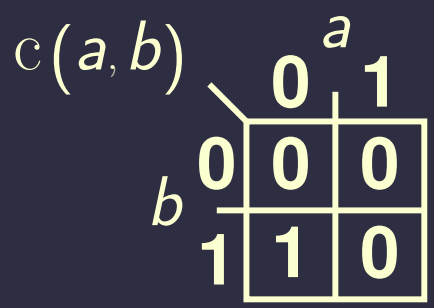
Observability Don't Cares



Output will be ignored for certain inputs

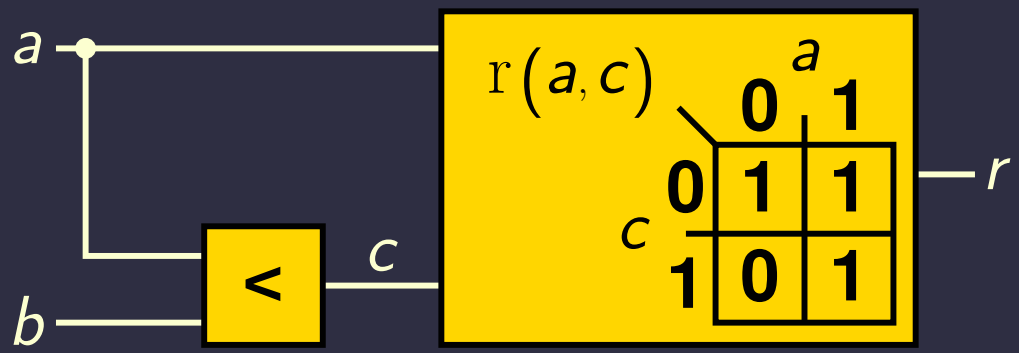
Observability Don't Cares

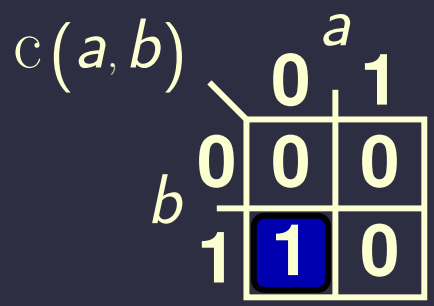




Output will be ignored for certain inputs

Observability Don't Cares

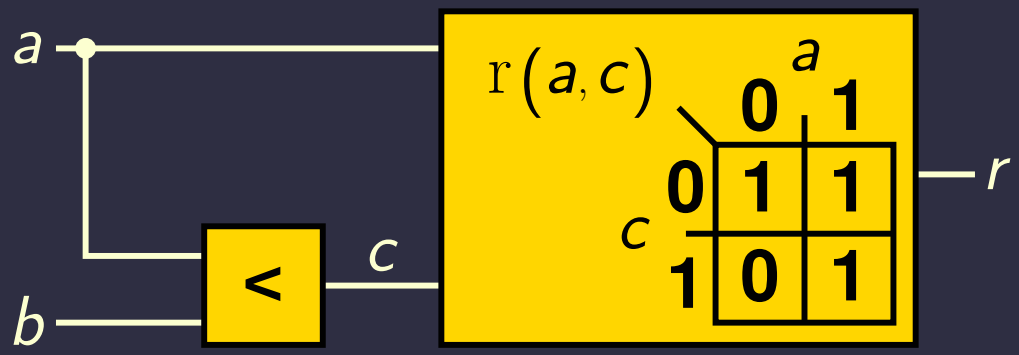


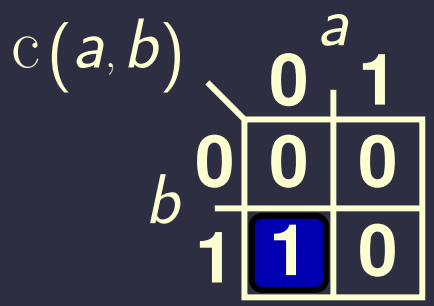


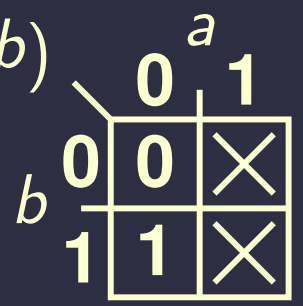
$$c(a,b) = \overline{a} \wedge b$$

Output will be ignored for certain inputs

Observability Don't Cares



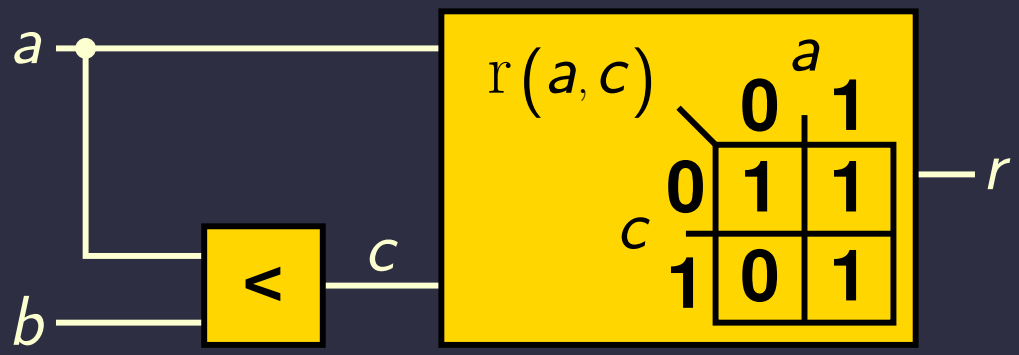




$$c(a,b) = \overline{a} \wedge b$$

Output will be ignored for certain inputs

Observability Don't Cares



$c(a,b)$

b \ a	0	1
0	0	0
1	1	0

$$c(a,b) = \overline{a} \wedge b$$

$c'(a,b)$

b \ a	0	1
0	0	X
1	1	X

$$c'(a,b) = b$$

Output will be ignored for certain inputs

Don't-care K-map example

- Minimize $f(a, b, c, d) = \sum(1, 3, 8, 9, 10, 11, 13) + d(5, 7, 15)$

Don't-care K-map example

- Minimize $f(a, b, c, d) = \sum(1, 3, 8, 9, 10, 11, 13) + d(5, 7, 15)$
- $f(a, b, c, d) = a\ \overline{b} + d$

CMOS

- Refer to M. Morris Mano and Charles R. Kime. *Logic and Computer Design Fundamentals*. Prentice-Hall, NJ, fourth edition, 2008
- http://www.writphotec.com/mano/
- CMOS supplement
- Optimization supplement
- qm.py at http://ziyang.eecs.northwestern.edu/~dickrp/tools.html

Outline

Intractable problems reading assignment

- Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Company, NY, 1979
- Chapter 1, Sections 1–5
- Introduces the concept of intractable problems
- Many problems in digital design are intractable
  - Too hard to solve optimally in a reasonable amount of time
- Use heuristics

Computer Geek Culture

- Python and perl