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

1. Administration
2. Overview of course
3. Homework
4. Misc.

Today's goals

1. Know how to get access to the resources you'll need for this course
   - Books, computer lab, website, newsgroup
2. Understand work and grading policies
3. Have a rough understanding of the topics we will cover
4. Have a rough understanding of an example design
   - You'll soon be designing similar systems on your own

Administration

- Lecture notes handed out before class
- PDF files posted after lectures
- http://ziyang.eecs.northwestern.edu/~dickrp/eecs303/
- If something isn't clear and you ask about it in class, I'll sometimes add more detail to the slides before posting

Class prerequisites

- ECE 203: Introduction to Computer Engineering
  - Need to have basic understanding of digital systems, logic gates, combinational logic, and sequential logic
- Need Unix experience (or need to catch up) since we will use the Mentor Graphics tools on Sun workstations
- Expect you to familiarize yourself with the basics of using this OS on your own but will give some hints
  - Use search engine, e.g., google: "unix beginners"
  - http://www.ee.surrey.ac.uk/Teaching/Unix/ not a bad place to start

Class foundation for

- EECS 347: Microprocessor System Projects
- EECS 357: Introduction to VLSI CAD
- EECS 361: Computer Architecture
- EECS 362: Computer Architecture Projects
- EECS 391: Introduction to VLSI Design
- EECS 392: VLSI Design Projects
- EECS 393: Design and Analysis of High-Speed Integrated Circuits

Required book

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

Reference books

- Allen Dewey. *Analysis and Design of Digital Systems With VHDL*. PWS Publishing Company, International Thompson Publishing, 1997
- Zvi Kohavi. *Switching and Finite Automata Theory*. McGraw-Hill Book Company, NY, 1978
- A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers principles, techniques, and tools*. Addison-Wesley, MA, 1986
- Randy H. Katz. *Contemporary Logic Design*. The Benjamin/Cummings Publishing Company, Inc., 1994

Grading policies

Homeworks:	25% of grade
Labs:	25% of grade
Midterm exam:	20% of grade
Final exam:	30% of grade

- Homeworks and labs due at beginning of class on due date
- 5% penalty for handing after start of class but still on due date
- 10% penalty per late working day
- No credit if more than three working days late

Grading style

- Homeworks and some labs will be graded quite strictly
  - Learn from the feedback
  - See the TA or me if something doesn't make sense
  - Don't assume a 75% grade on the homework implies a C in the course – it doesn't
- Will cover a limited amount material in lectures that does not appear in the course textbook
  - However, you'll have access to the full set of lecture notes
- I will do my best not to make exams surprising
  - However, they won't be easy and the best students in the class probably won't get 100% on the exams

Lab assignments

Tried to make lab assignments get to the point w.o. wasting time

- In this area, lab assignments necessarily require some time
- May take you much longer than some other students if you need to refresh your memory or fill in gaps in your background
- You probably will not be able to finish labs on time if you start them the day before they're due

Lab assignments

Tried to make lab assignments get to the point w.o. wasting time

- In this area, lab assignments necessarily require some time
- May take you much longer than some other students if you need to refresh your memory or fill in gaps in your background
- You probably will not be able to finish labs on time if you start them the day before they're due

Lab work

- Computer aided-design (CAD) software from Mentor Graphics
- Sun workstations in the Wilkinson Lab (M338 Tech)
- Lab Hours: Open
- Topics
  - Tutorial on Mentor Graphics (simple logic)
  - Design of combinational logic
  - Design of sequential logic
  - Use of VHDL for combinational and sequential design

Decide office hours

We can reschedule office hours based on your comments

Person	Day	Time	Room
Robert Dick	Tuesday	5:00–6:00	L477 Tech
Robert Dick	Thursday	5:00–6:00	L477 Tech

We'll go to the Wilkinson Lab (M338 Tech) when requested.

Subscribe to mailing list

- Very useful for getting questions rapidly answered
- If you email an academic question to the TA or me, we will post the question and the answer to the newsgroup/mailing list but remove your name
- Send mail to "listserv@listserv.it.northwestern.edu"
- No subject
- Body of SUBSCRIBE ADLD [Firstname] [Lastname]
- Send mail to "adld@listserv.it.northwestern.edu" to post
- I will archive posts and make them available via the course web page

Outline

1. Administration
2. Overview of course
3. Homework
4. Misc.

Section outline

2. Overview of course
   - Topics
   - Goals
   - Overview and review
   - Case study

Course topics in context I

1. Boolean algebra (brief review)
   - Formulating problems as Boolean expressions
   - Can use to solve problems in many fields of engineering
2. Karnaugh maps (brief review)
   - Helps visualize problem in which adjacency is important
3. Quine–McCluskey (fairly quick coverage, depending on background)
   - Covering

Course topics in context II

4. Heuristic logic minimization
   - Complexity and algorithms
5. Implementation technologies
   - Useful starting point for prototyping designs
   - Implementation technologies are constantly changing
6. Graph definitions, critical path, and topological sort
   - Basic understanding of graph algorithms
7. Number systems, binary arithmetic (more detail, advanced operations)
   - Fundamental meaning of mathematical operations

Course topics in context III

8. Technology mapping
   - Covering
9. FSM design, non-deterministic intermediate representations
   - Compiler, languages, CS theory
10. Incompletely specified FSM state minimization
    - Covering
11. CAD software
    - Testing ideas in other fields, e.g., computer architecture
12. Testing (if time permits)

Section outline

2. Overview of course
   - Topics
   - Goals
   - Overview and review
   - Case study

Course goals I

1. Learn to manually design, optimize, and implement small digital combinational circuits.
2. Have a basic understanding of the building blocks and implementation technologies available to digital designers.
3. Understand how to use schematic capture software to design digital circuits.
4. Be capable of doing automatic and manual timing analysis of combinational circuits.
5. Be capable of using CAD software to automatically optimize large digital combinational circuits and map them to a target technology.

Course goals II

6. Have a high-level understanding of the algorithms such synthesis software uses (e.g., logic optimization and technology mapping). This first portion of the course was dedicated to combinational design. I went into depth on a few more advanced topics because I wanted you to see some of the beauty of the algorithms used to automatically design circuits, e.g., my description of portions of the Espresso algorithm.
7. Understand how to design, optimize, and implement finite state machines.
8. Understand that sequential behavior can be specified in different ways and have a reasonably good understanding of how to start from a few different types of specifications and end up with working logic.

Course goals III

9. Understand the differences between synchronous and asynchronous finite state machines and know the advantages of each.
10. Be capable of doing simple VHDL designs.

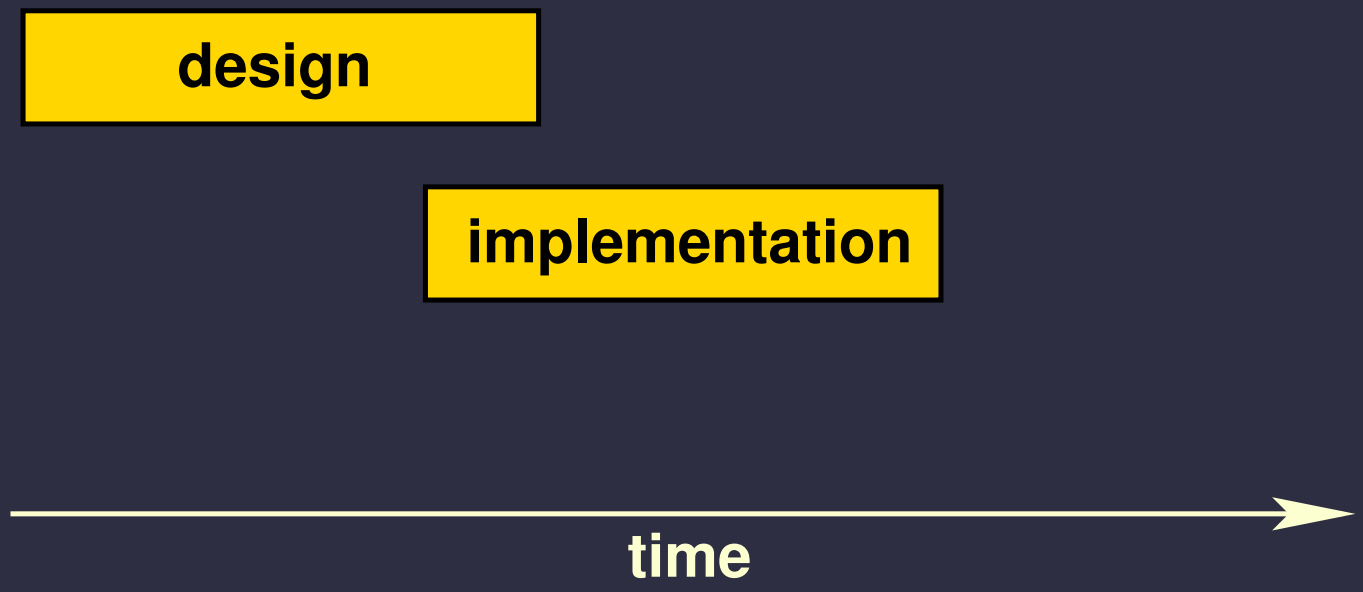
Section outline

2. Overview of course
   - Topics
   - Goals
   - Overview and review
   - Case study

Design, implementation, and debugging

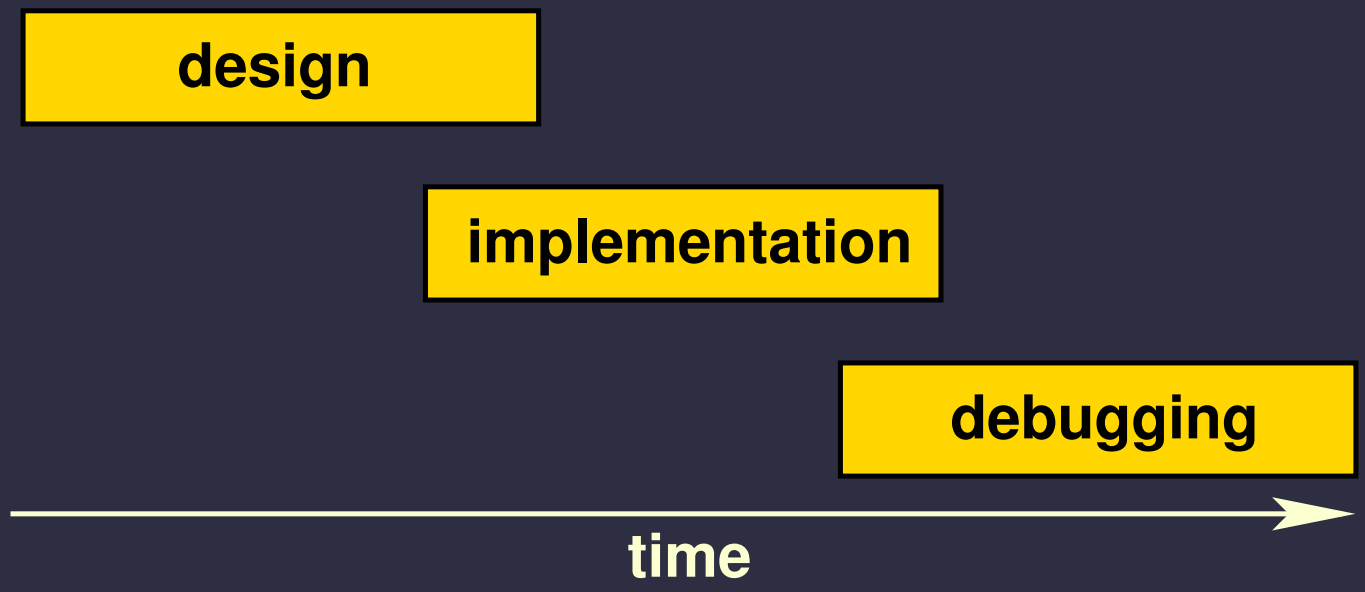
- Functionality
- Constraints: Timing, area, power, price
- Formally define abstract blocks

Design, implementation, and debugging



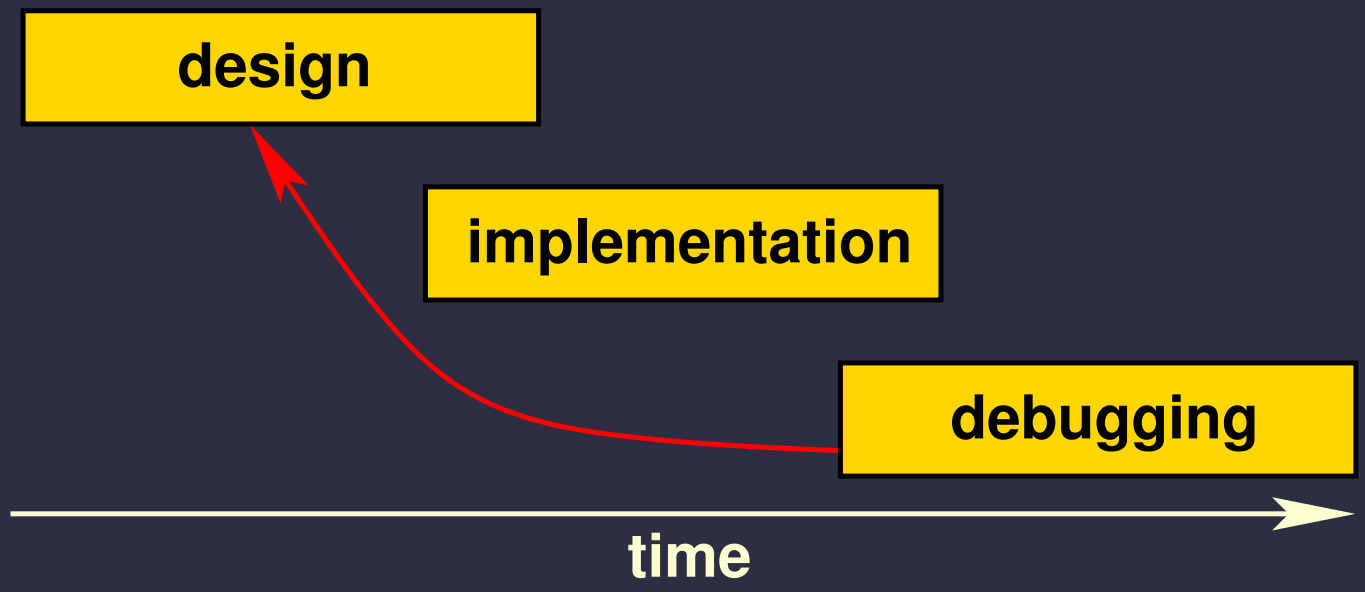
- Assemble primitive building blocks into hierarchical system
- Choose among design alternatives after impacts explored

Design, implementation, and debugging

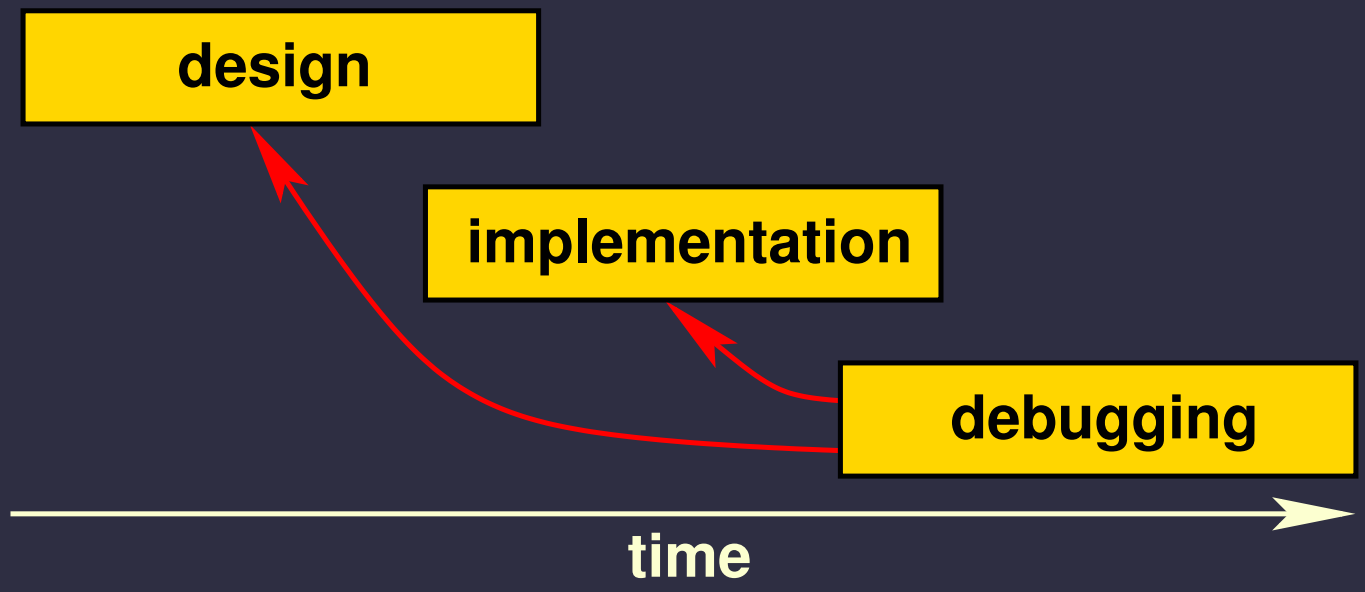


- Fault isolation: Design flaws, implementation flaws, component flaws
- Hypothesis formation and testing
- Good design and implementation make debugging easier
  - 4–5 hours $\longrightarrow$ 12–14 hours

Design, implementation, and debugging



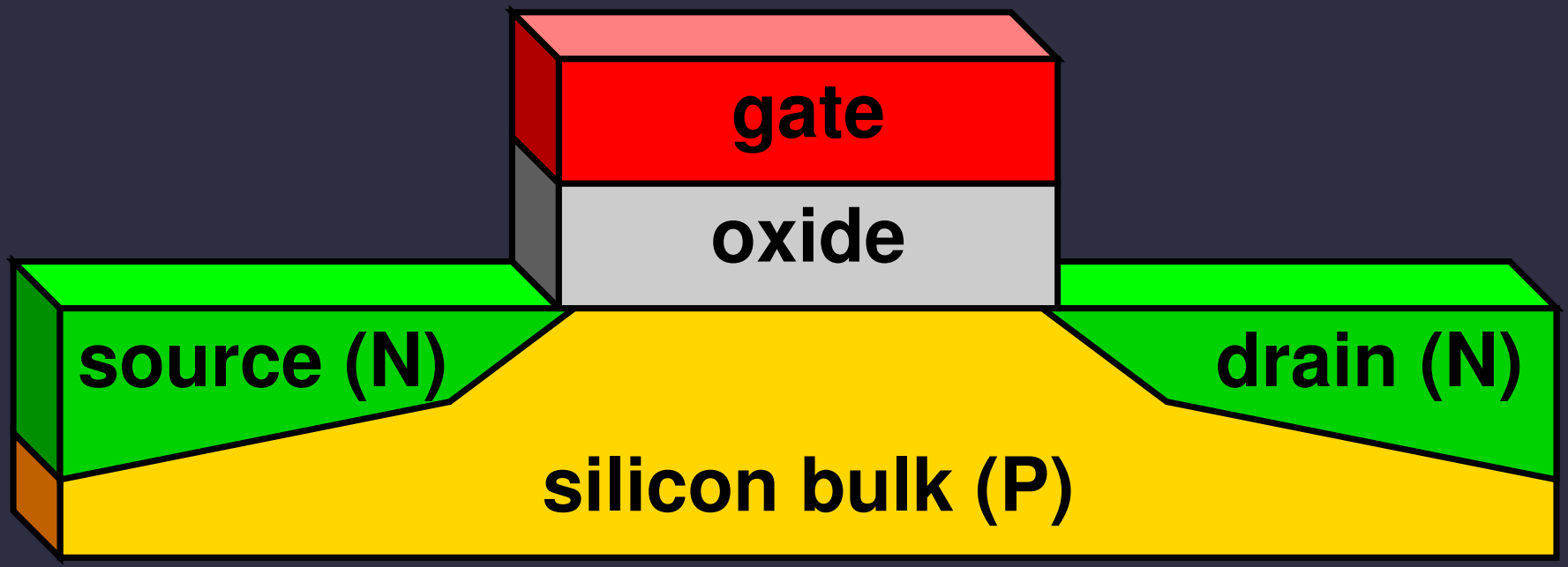
Design, implementation, and debugging



Ambitious goal for synthesis

- Start from single system-level description
- Automatically build all hardware
- Where do we start?

Review: MOSFETs



Relationship with CMOS

- Metal Oxide Silicon
- 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

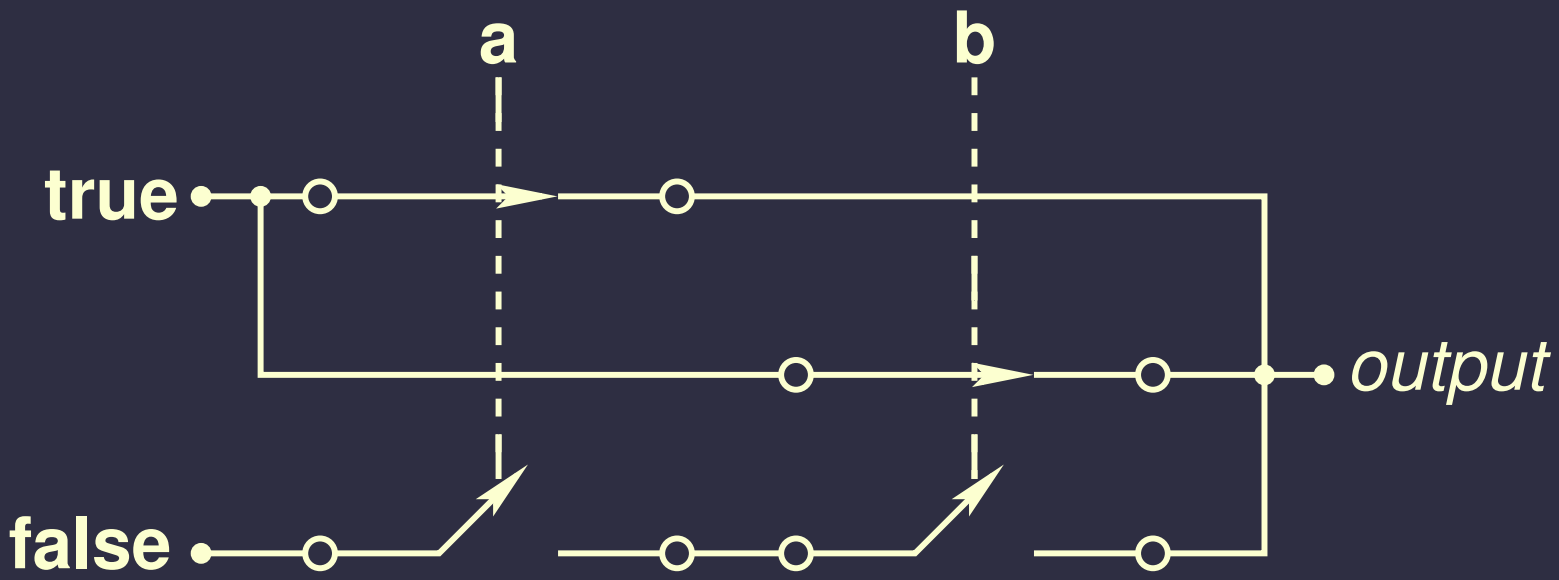
- Metal Oxide Silicon
- Positive and negative carriers
- Complimentary MOS
- PMOS gates are like normally closed switches that are good at transmitting only true (high) signals
- NMOS gates are like normally open switches that are good at transmitting only false (low) signals

Relationship with CMOS

- Metal Oxide Silicon
- 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

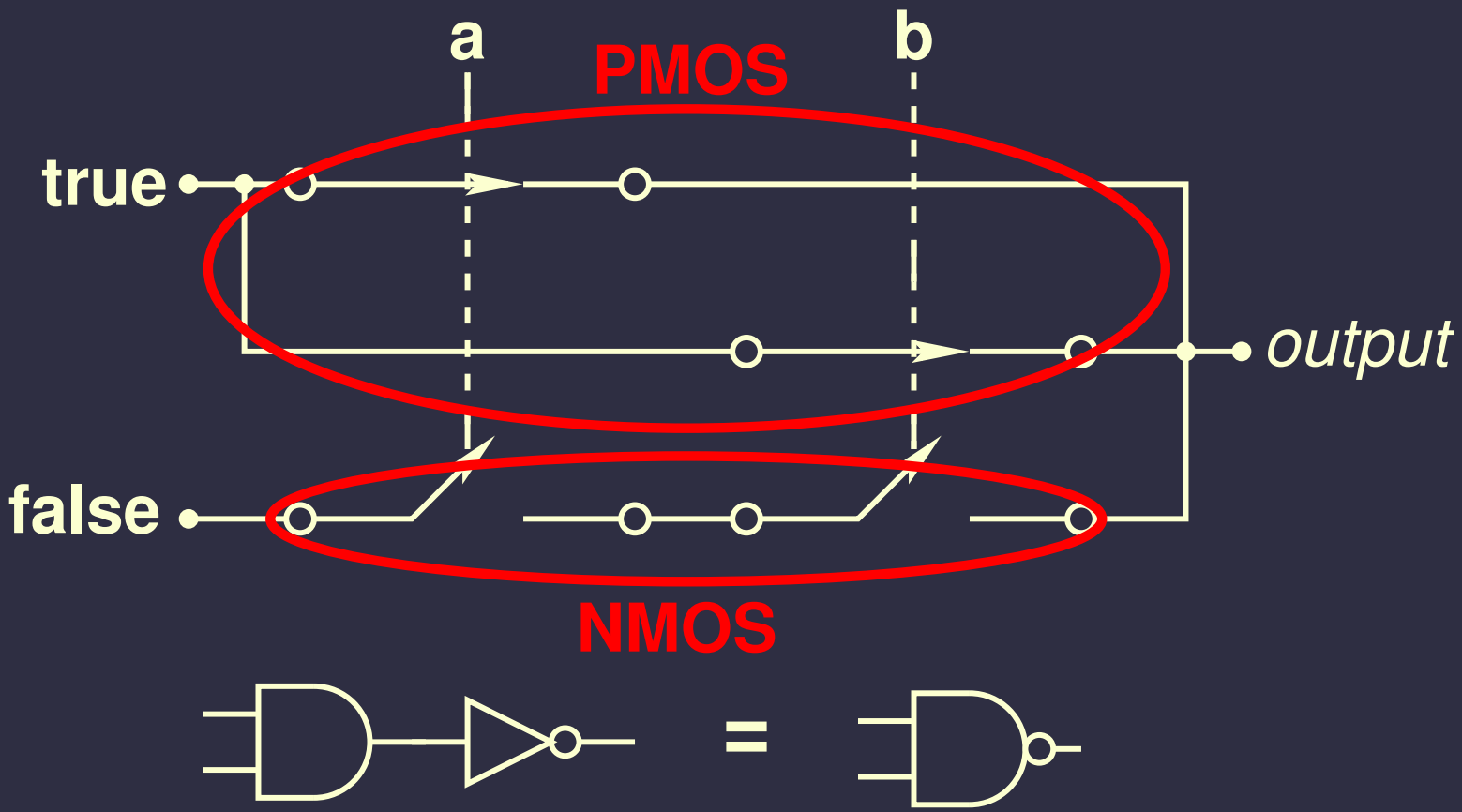
NAND gate

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

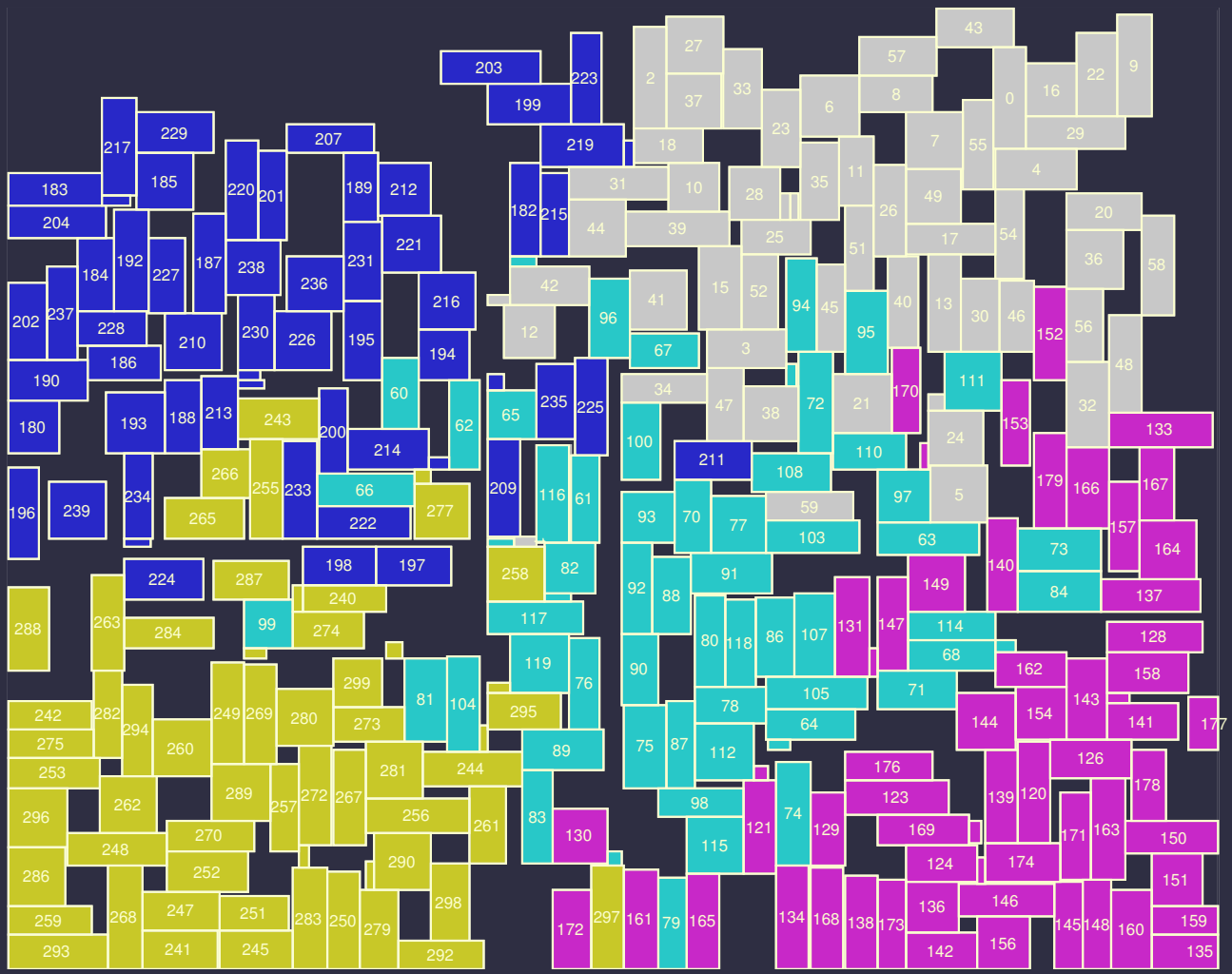


NAND gate

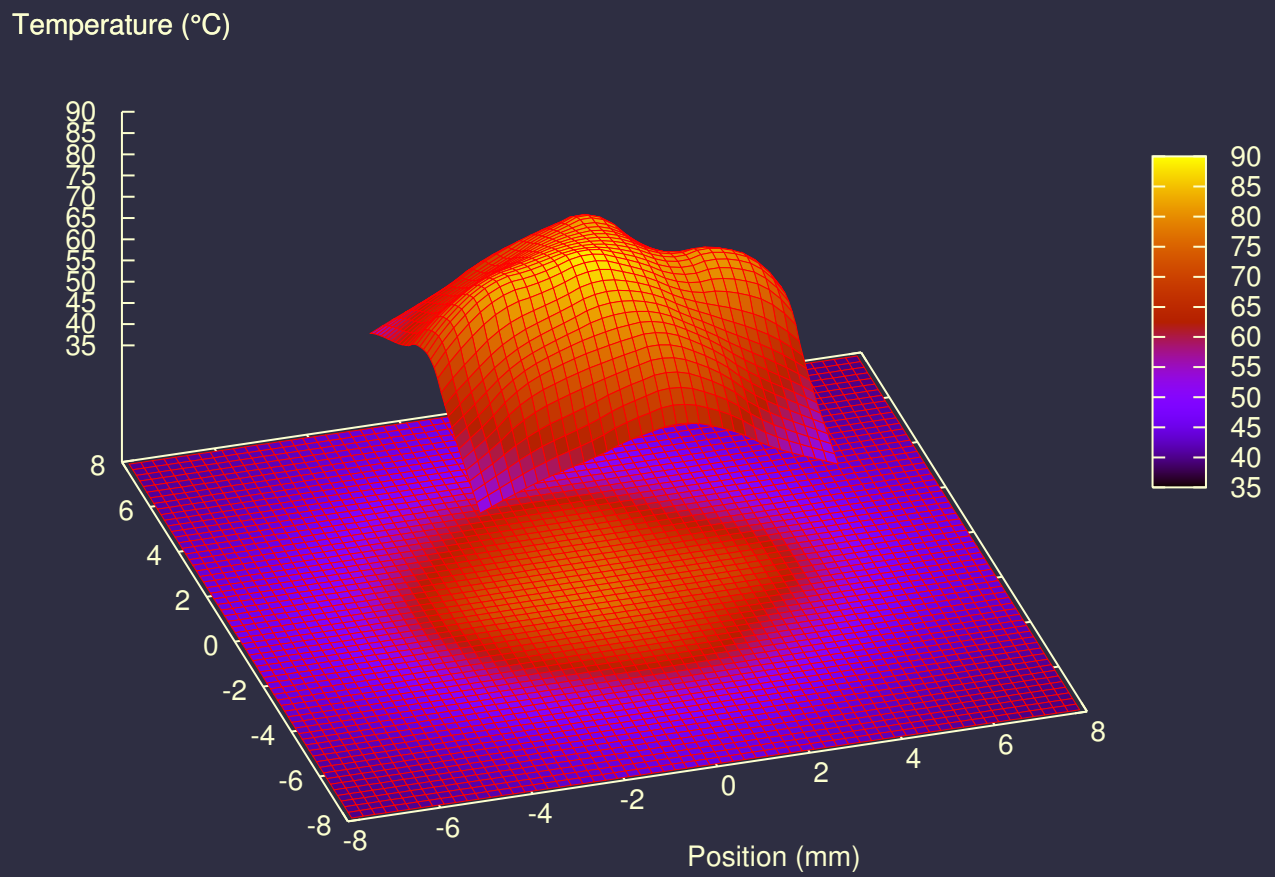
- Therefore, *NAND* and *NOR* gates are used in CMOS design instead of *AND* and *OR* gates



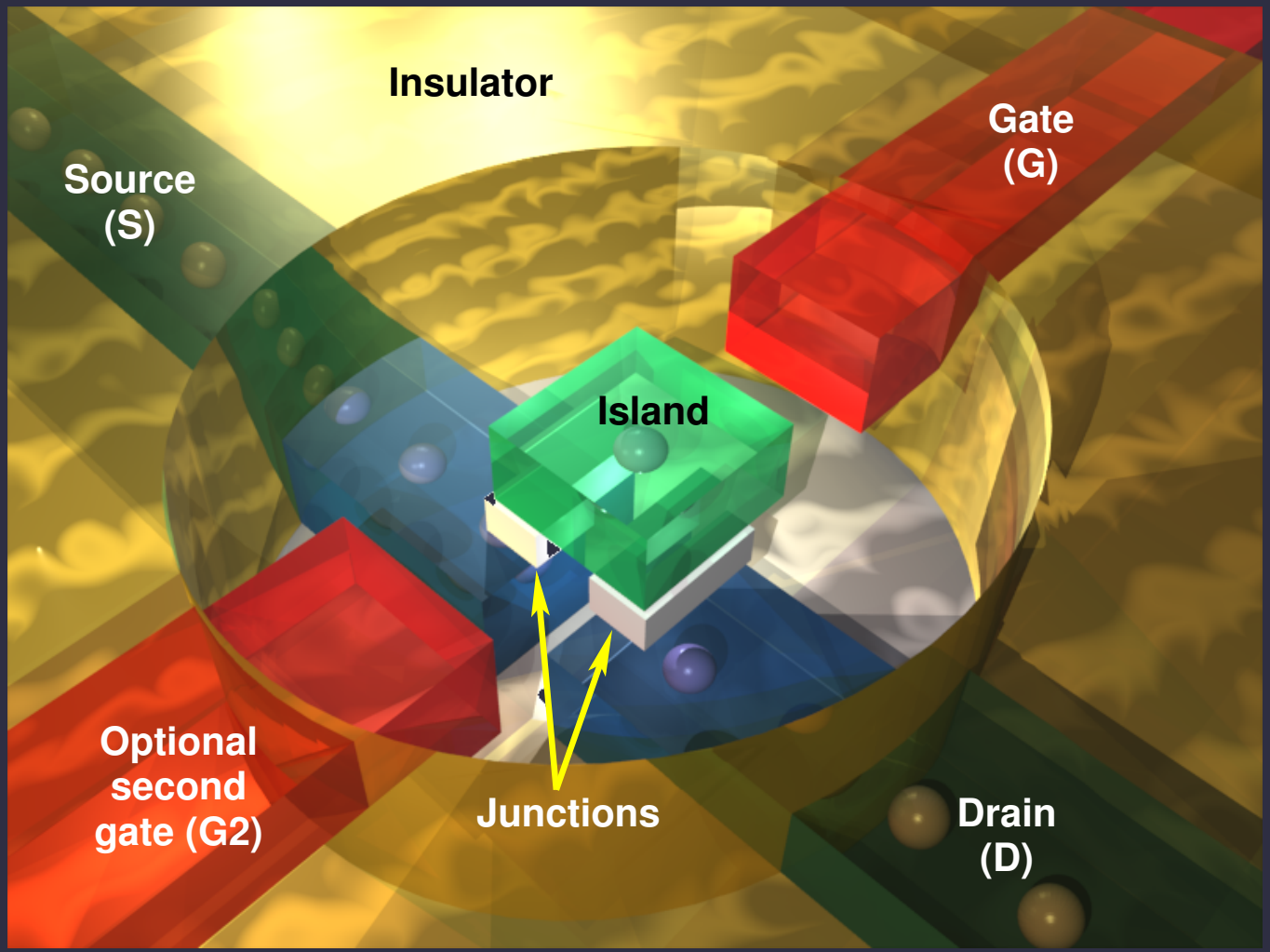
Floorplanning



Thermal analysis



New technologies

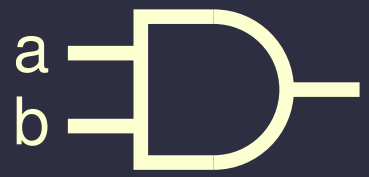


Review: Boolean algebra

- The only values are 0 (or false) and 1 (or true)
- One can define operations/functions/gates
  - Boolean values as input and output
- A truth table enumerates output values for all input value combinations

AND

a	b	a ∧ b
0	0	0
0	1	0
1	0	0
1	1	1



$$a \; \textit{AND} \; b = a \wedge b = a \; b$$

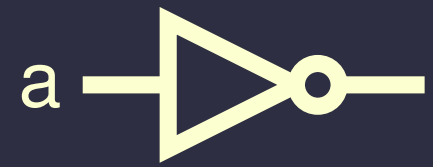
OR

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

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

NOT

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



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

Section outline

2. Overview of course
    - Topics
    - Goals
    - Overview and review
    - Case study

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

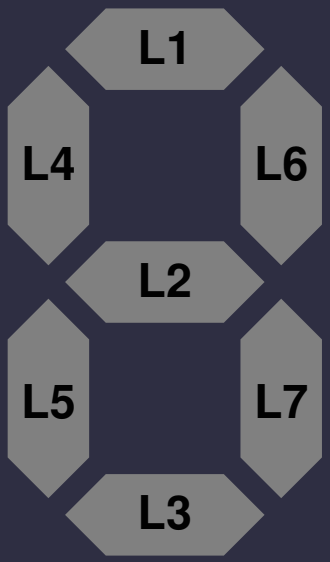
Case study – seven-segment display

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

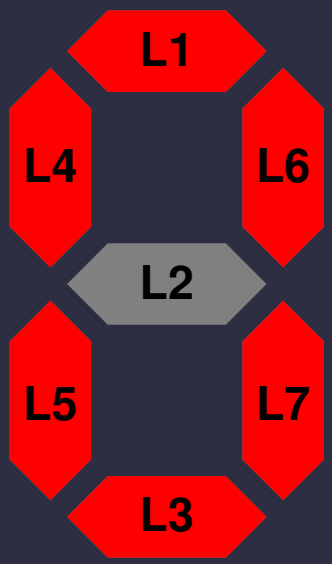
i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Case study – Seven-segment



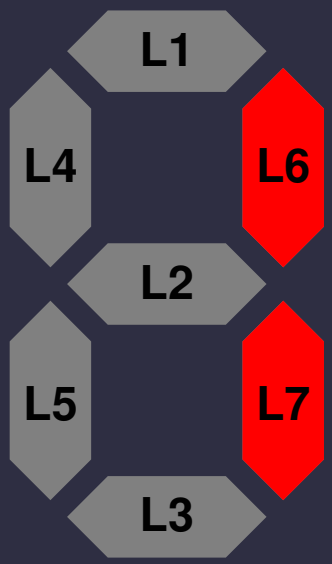
i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Case study – Seven-segment



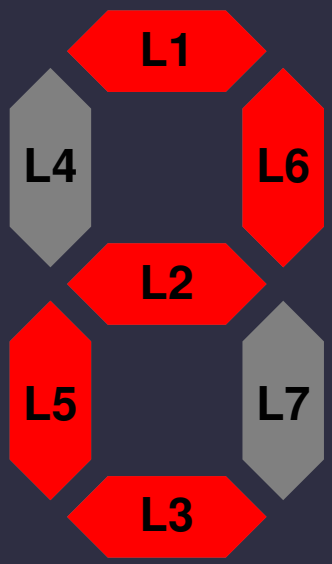
i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Case study – Seven-segment



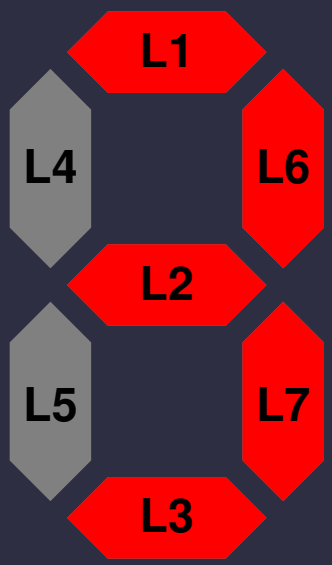
i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Case study – Seven-segment



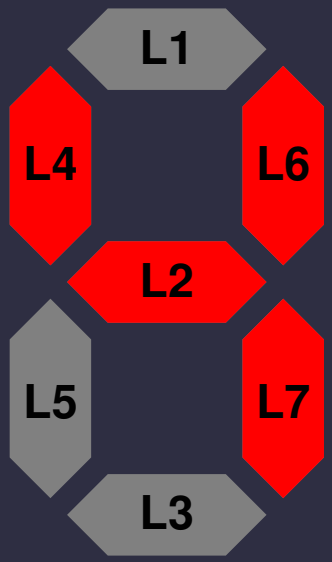
i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Case study – Seven-segment



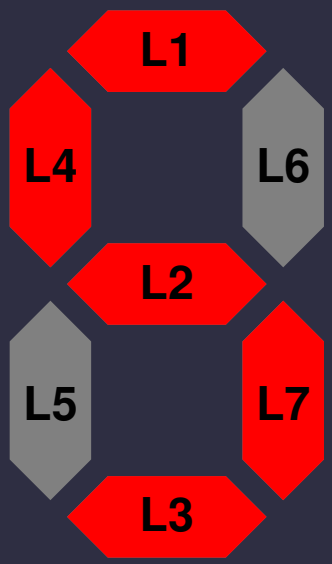
i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Case study – Seven-segment



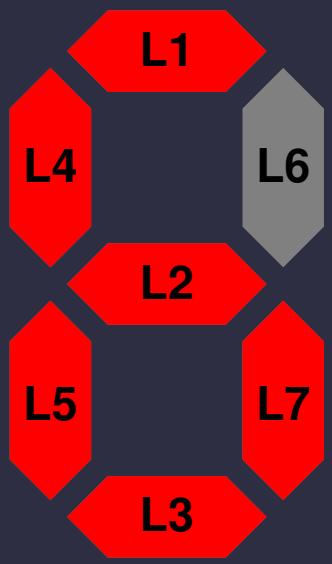
i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Case study – Seven-segment



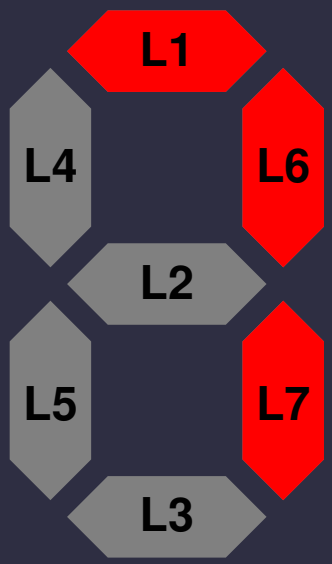
i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Case study – Seven-segment



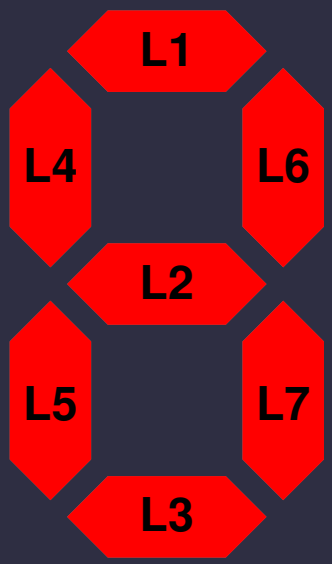
i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Case study – Seven-segment



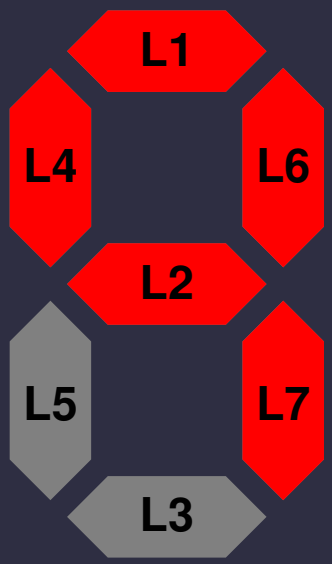
i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Case study – Seven-segment



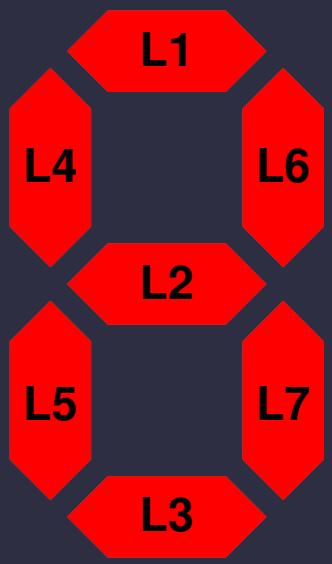
i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Case study – Seven-segment



i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Case study – Seven-segment

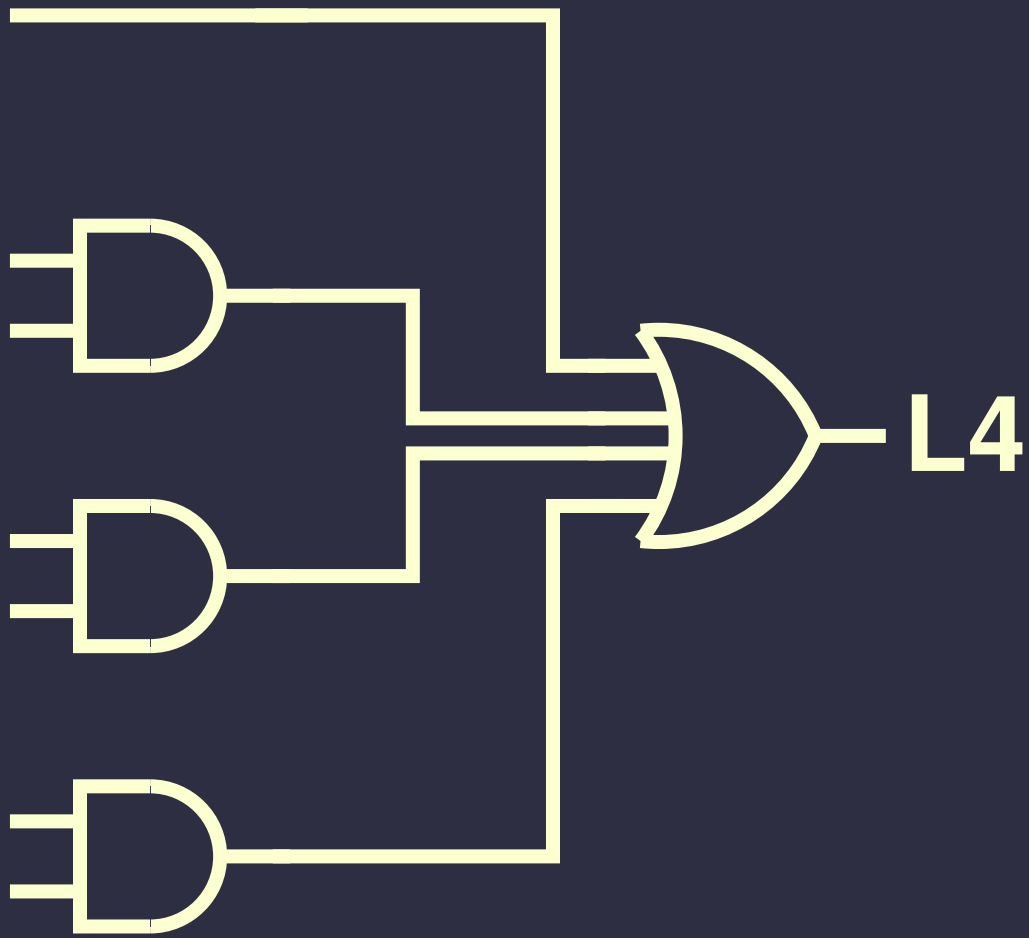


i_3	i_2	i_1	i_0	dec	L1	L2	L3	L4	L5	L6	L7
0	0	0	0	0	1	0	1	1	1	1	1
0	0	0	1	1	0	0	0	0	0	1	1
0	0	1	0	2	1	1	1	0	1	1	0
0	0	1	1	3	1	1	1	0	0	1	1
0	1	0	0	4	0	1	0	1	0	1	1
0	1	0	1	5	1	1	1	1	0	0	1
0	1	1	0	6	1	1	1	1	1	0	1
0	1	1	1	7	1	0	0	0	0	1	1
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	0	1	0	1	1

Implement L4

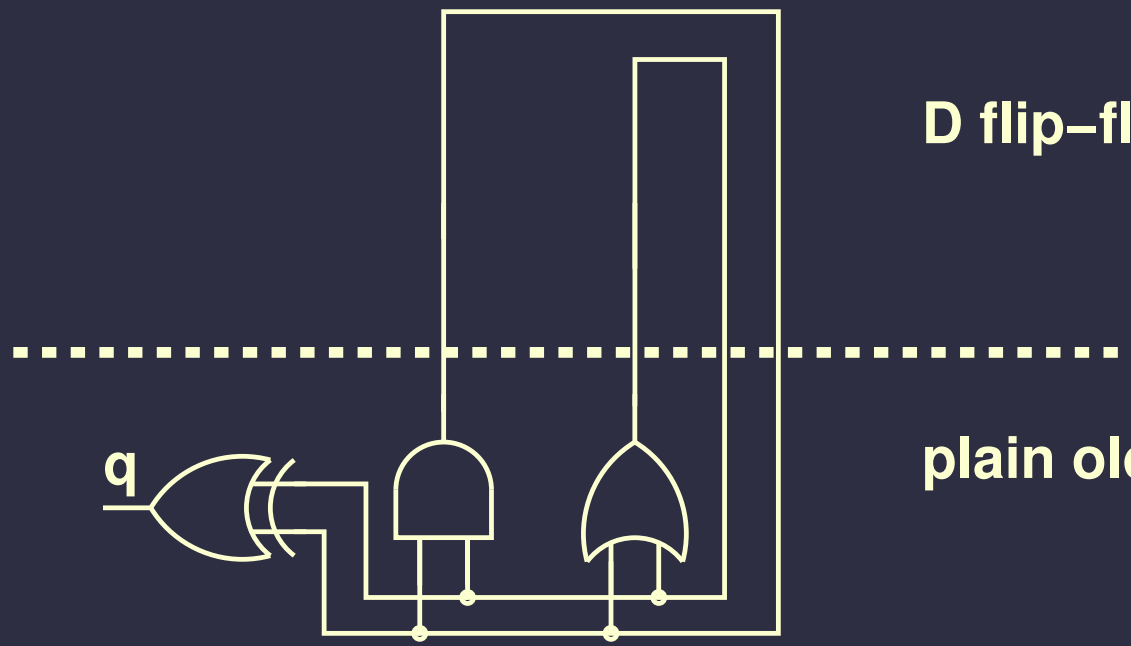
i_3	i_2	i_1	i_0	dec	L4
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	2	0
0	0	1	1	3	0
0	1	0	0	4	1
0	1	0	1	5	1
0	1	1	0	6	1
0	1	1	1	7	0
1	0	0	0	8	1
1	0	0	1	9	1

L4 implementation



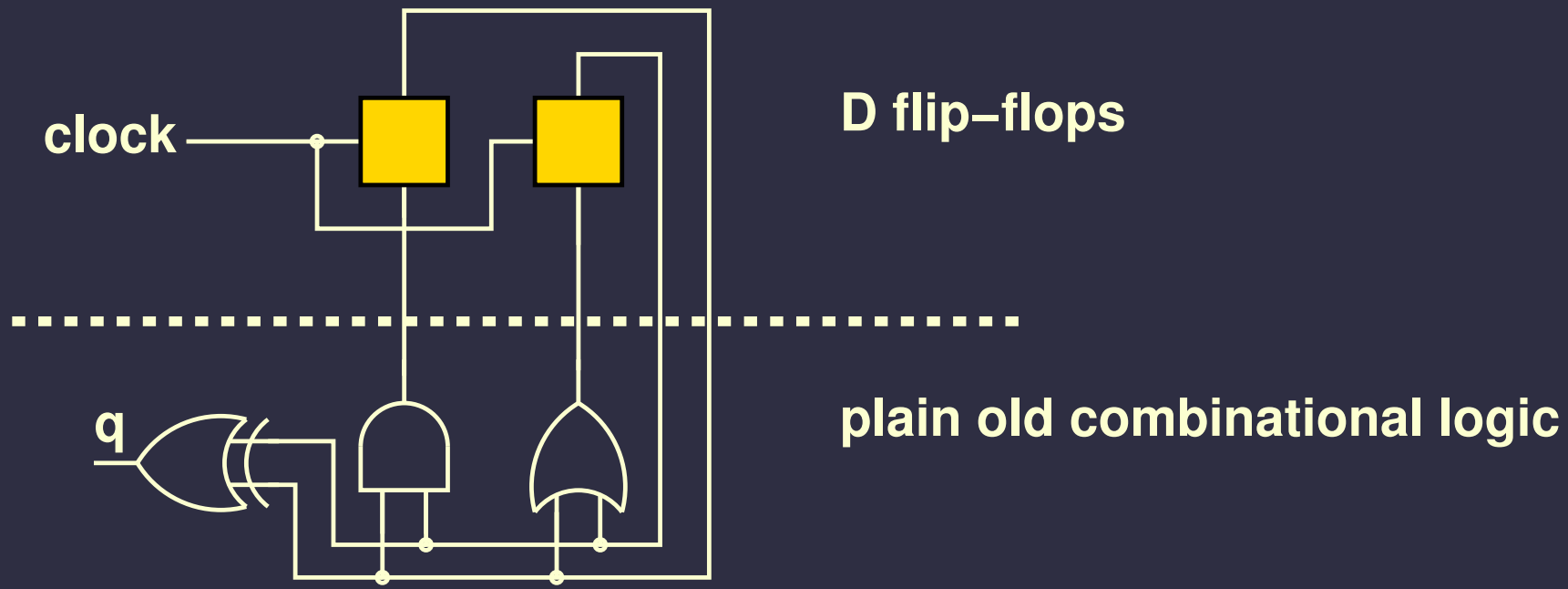
Combinational vs. sequential logic

- No feedback between inputs and outputs – combinational
  - Outputs a function of the current inputs, only



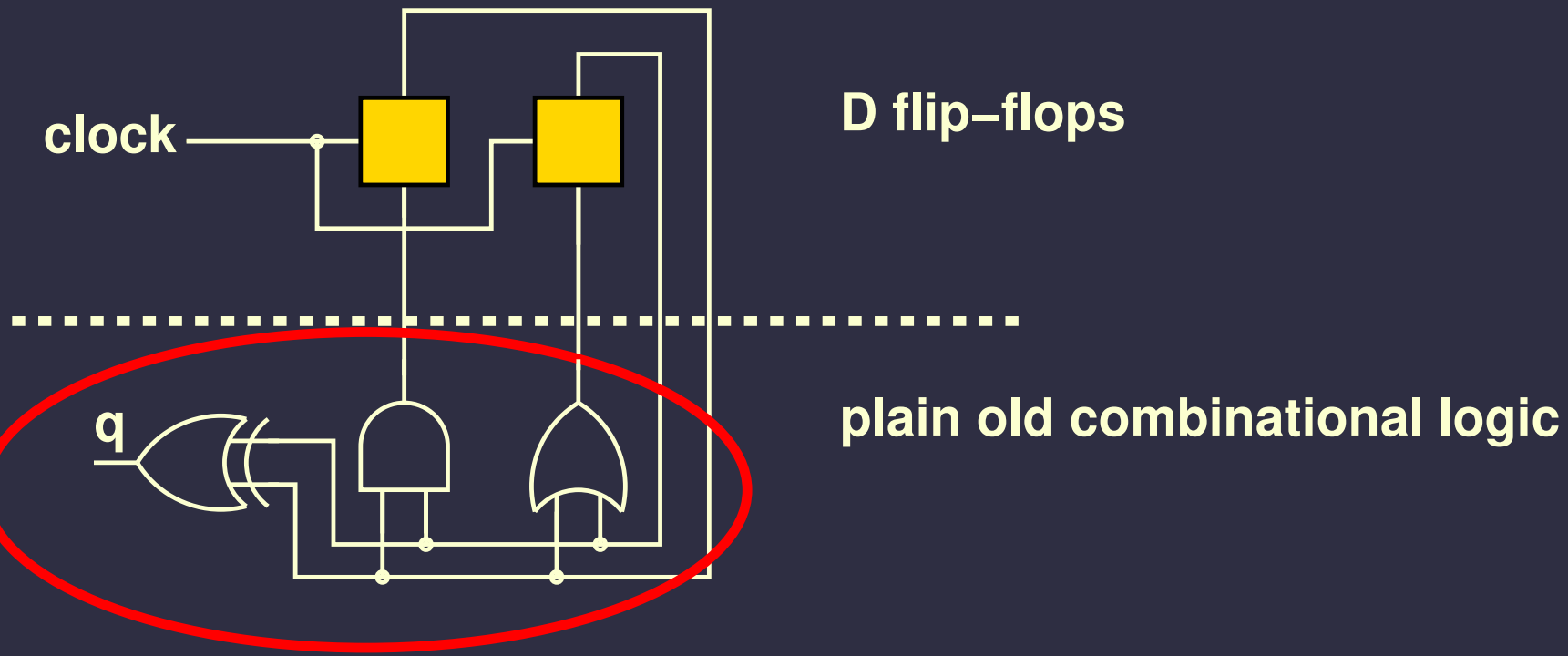
Combinational vs. sequential logic

- No feedback between inputs and outputs – combinational
  - Outputs a function of the current inputs, only
- Feedback – sequential



Combinational vs. sequential logic

- No feedback between inputs and outputs – combinational
  - Outputs a function of the current inputs, only



Sequential logic

- Outputs depend on current state and (maybe) current inputs
- Next state depends on current state and input
- For implementable machines, there are a finite number of states
- Synchronous
  - State changes upon clock event (transition) occurs
- Asynchronous
  - State changes upon inputs change, subject to circuit delays

What can we do?

- Finite state machine design

What can we do?

- Finite state machine design
- Logic minimization

What can we do?

- Finite state machine design
- Logic minimization
- Implementation with gates

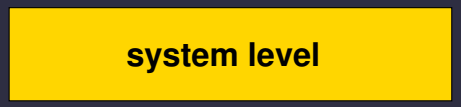
What can we do?

- Finite state machine design
- Logic minimization
- Implementation with gates
- Need a lot more depth, and a lot more detail!

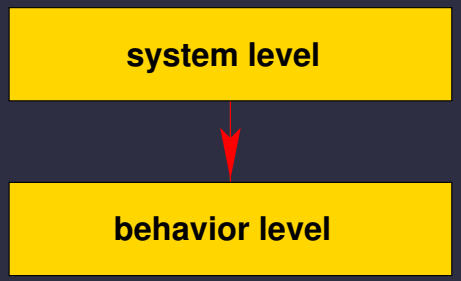
Ambitious goal for synthesis

- Start from single system-level description
- Automatically build all hardware
- Where do we start?
- What are the fundamental barriers?
- What new discoveries are necessary?

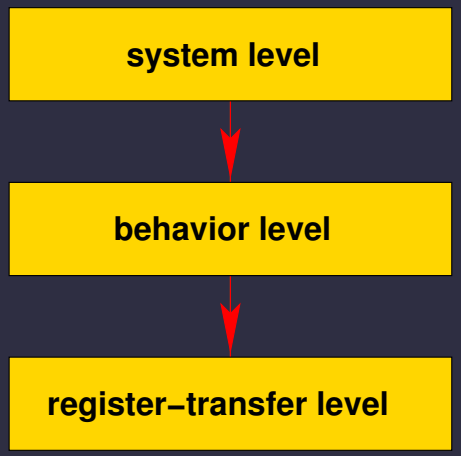
Conventional synthesis



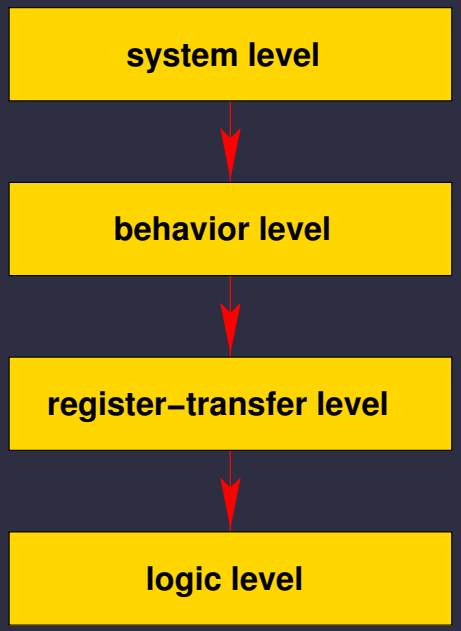
Conventional synthesis



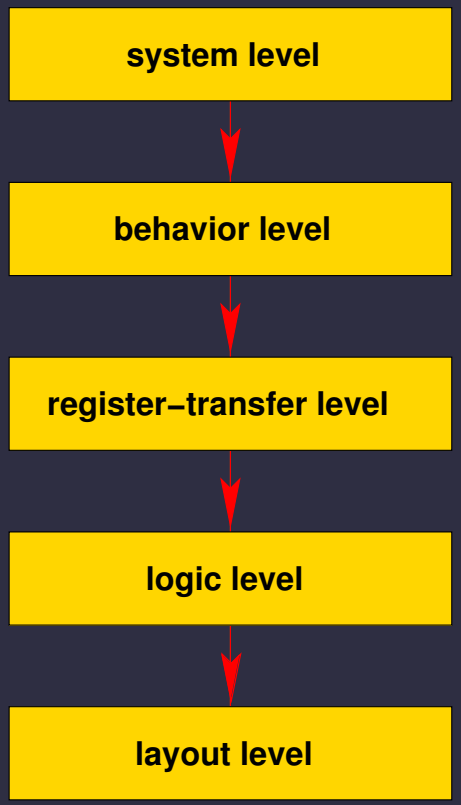
Conventional synthesis



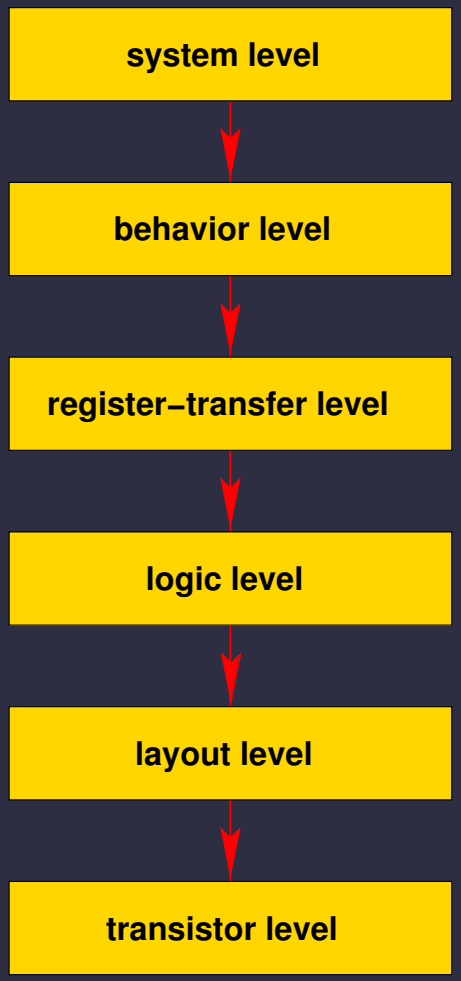
Conventional synthesis



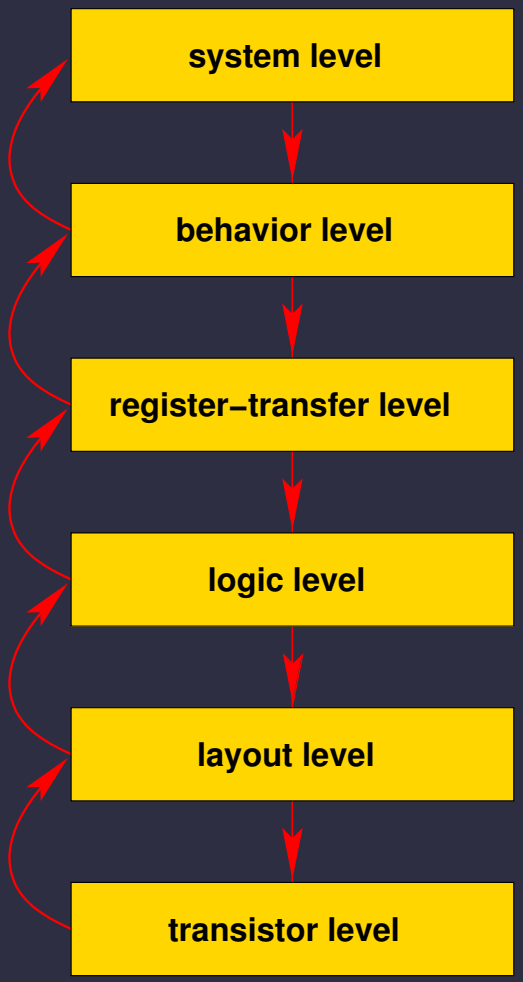
Conventional synthesis



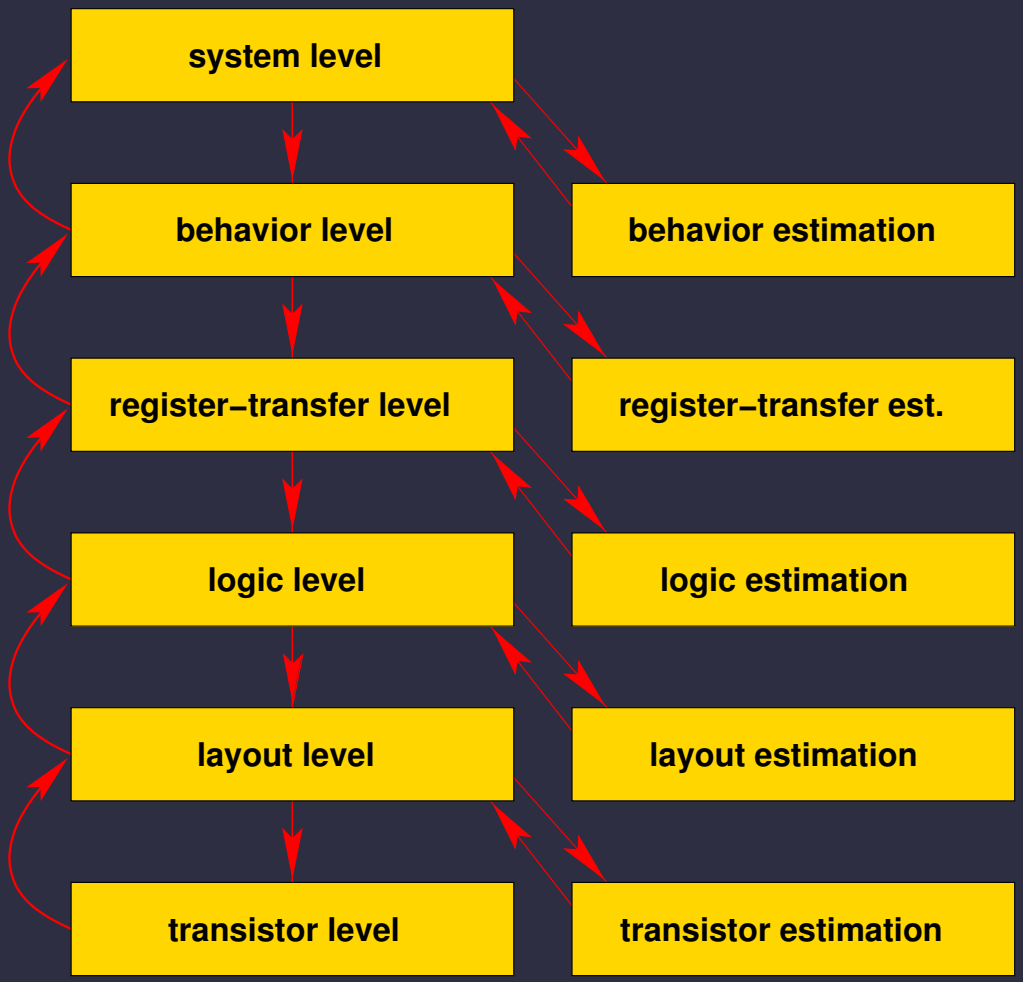
Conventional synthesis



Conventional synthesis



Conventional synthesis



Status and approach

- Understand a few combinational logic design techniques
  - Much left to learn
- Have scratched the surface of sequential logic design
  - Much left to learn
- Little knowledge of automation and its fundamental barriers
- Approach: Start from the core we learned in EECS 203
  - Build breadth and depth

Combinational design

- Let's start by reviewing combinational design
- A lot of amazing stuff will build upon this later

Outline

1. Administration
2. Overview of course
3. Homework
4. Misc.

Get Wilkinson lab account

- Confirm that you are registered for the course at http://courses.northwestern.edu/
- The administrators have a list of students.
- They will create accounts and add physical access to M334 to your card.

Introductory reading assignments

- In general, reading assignments will cover material that will be presented in the next class.
- It may seem like a lot but most should be review from EECS 203.
- Even if you think you remember the material from EECS 203, spend a few minutes with the book to confirm.
- M. Morris Mano and Charles R. Kime. *Logic and Computer Design Fundamentals*. Prentice-Hall, NJ, fourth edition, 2008
- Chapters 2, 3, and 4

Outline

1. Administration
2. Overview of course
3. Homework
4. Misc.

Computer geek culture

- RLE
- Compression geek culture: compression = prediction = classification