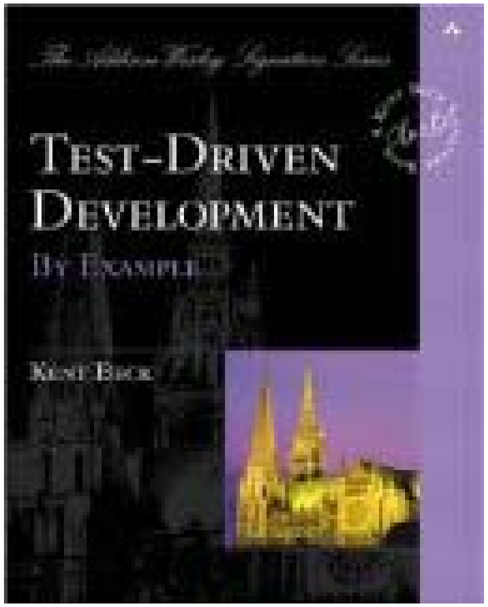
Lecture 5. Test-Driven Development for Analog Circuit Design

Jaeha Kim
Mixed-Signal IC and System Group (MICS)
Seoul National University
jaeha@ieee.org

Test-Driven Development (TDD)

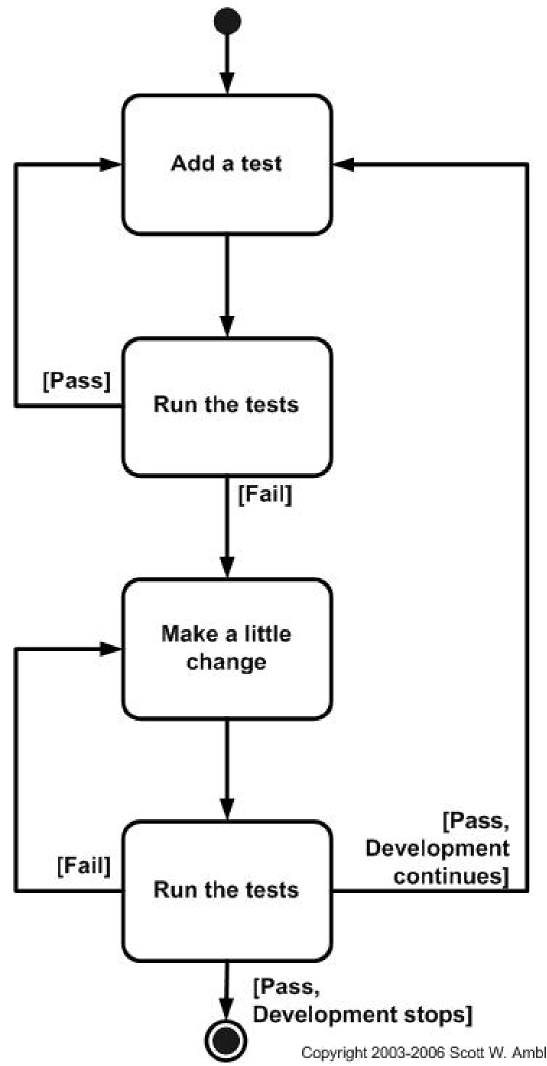
- TDD: an evolutionary approach to software engineering that combines "Test-First Development (TFD)" and Refactoring
  - Software codes are developed in small incremental steps rather than in one big step



- Kent Beck's Two Simple Rules:
  - (TFD) you should write new business code only when an automated test has failed
  - (Refactoring) you should eliminate any duplication that you find.

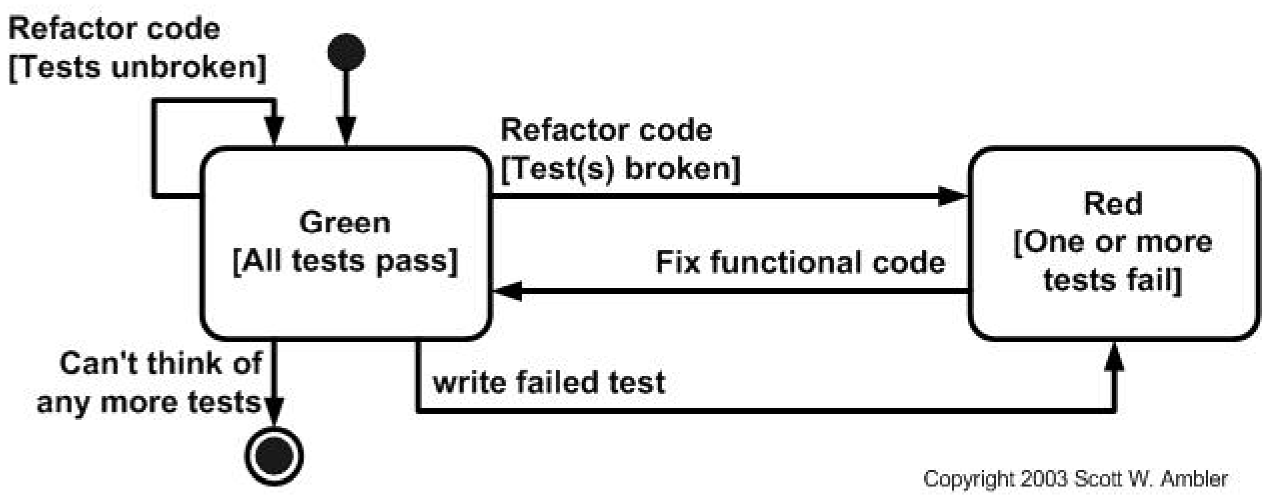
Test-First Development (TFD)

- TFD writes the "tests" first before writing the production codes
  - A programmer taking a TDD approach refuses to write a new function until there is first a test that fails because that function isn't present
  - In fact, they refuse to add even a single line of code until a test exists
  - Once the test is in place they then do the work required to ensure that the test suite now passes



Refactoring

- When implementing a new feature, ask whether the existing design is the best one possible to implement that functionality
  - If yes, proceed via a TFD approach
  - If no, refactor it locally to change the portion of the design affected by the new feature, enabling to add that feature as easy as possible
  - As a result, you will always be improving the quality of your design, thereby making it easier to work with in the future



TDD in Practice

- You design organically, with the running code providing feedback between decisions
- You write your own tests because you can't wait 20 times per day for someone else to write them for you
- Your development environment must provide rapid response to small changes
  - (e.g. you need a fast compiler and regression test suite)
- Your designs must consist of highly cohesive, loosely coupled components to make testing easier
  - This also makes evolution and maintenance easier

Good Unit Tests

- Run fast (they have short setups, run times, and break downs)
- Run in isolation (you should be able to reorder them)
- Use data that makes them easy to read and to understand
- Use real data (e.g. copies of production data) when they need to
- Represent one step towards your overall goal

TDD vs. Traditional Testing

- With TDD, you have *made progress* when a test fails
  - Since it means you have successfully identified what needs to be done – you haven't if the test works with the existing code
  - With traditional testing, tests uncover "problems or defects"
- With TDD, you have a clear measure of success when the test no longer fails
  - Increasing your confidence that your system actually meets the requirements defined for it, that your system actually works and therefore you can proceed with confidence
  - With traditional testing, you don't have confidence that you have tested all the features desired

Test with a Purpose

- TDD is primarily a specification technique in a sense that
  - Think through your requirements or design before your write your functional code
  - You must know why you are testing something and to what level it needs to be tested
- Yet, its valuable side effect is that every single line of code will be tested
  - Since you do not write codes unless there is a failing test
  - Hence 100% coverage test is achieved
  - Something that traditional testing doesn't guarantee, although it does recommend it

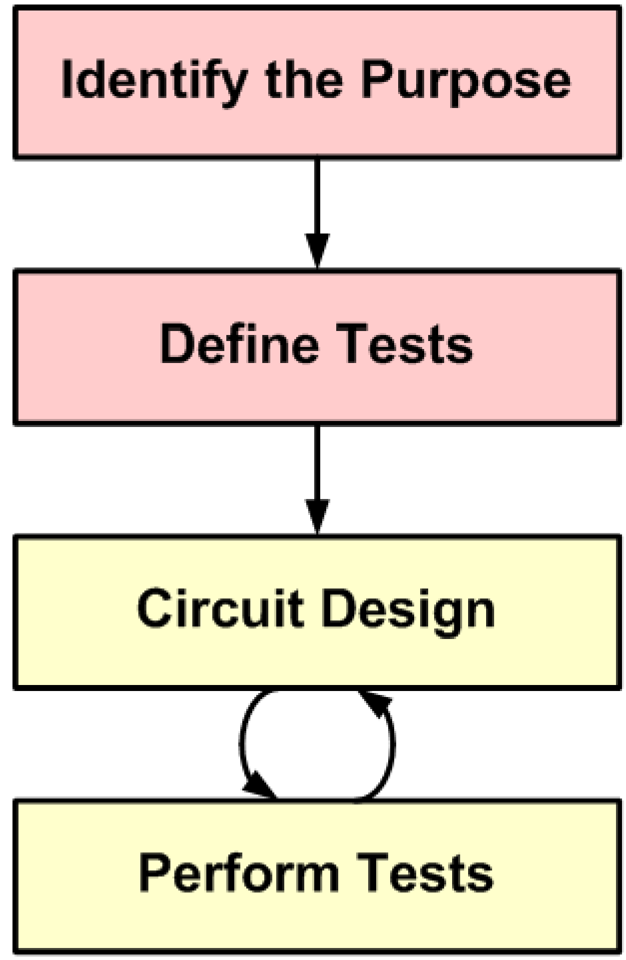
TDD in Short:

If it's worth building, it's worth testing.

If it's not worth testing, why are you wasting your time working on it?

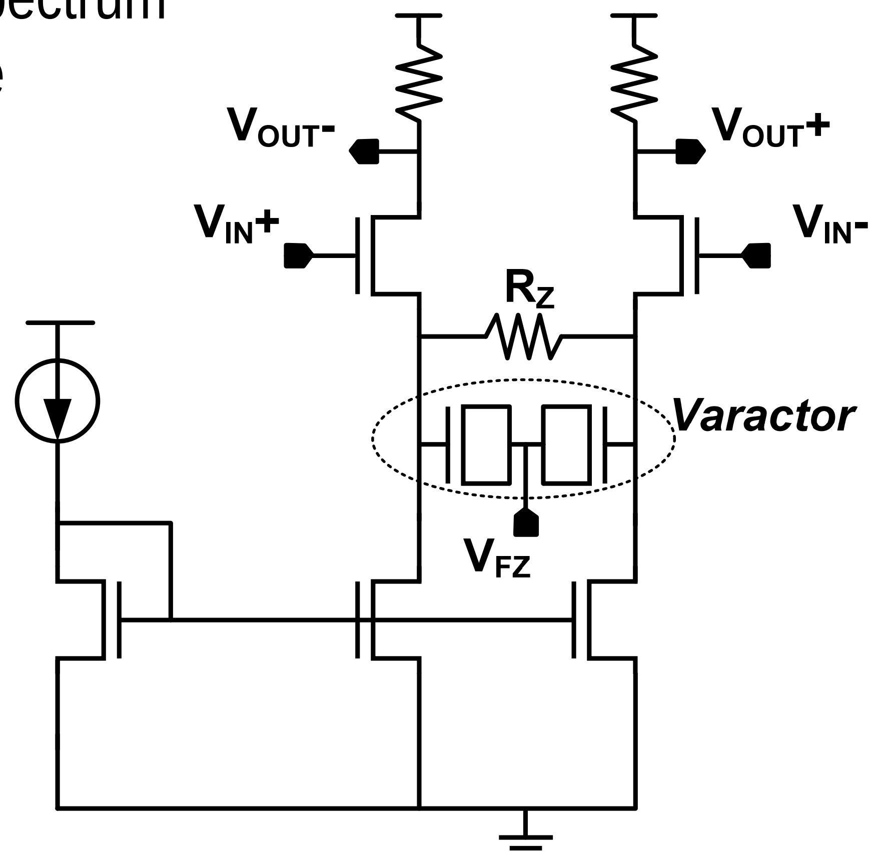
TDD for IC Design

- Know the purpose of the circuit before you design it
  - Recall that a typical purpose of a circuit is to realize a linear system
  - Or, an easy nonlinear system
- The purpose will guide what tests are necessary
  - Does it have the correct functionality?
  - Does it have good performance?



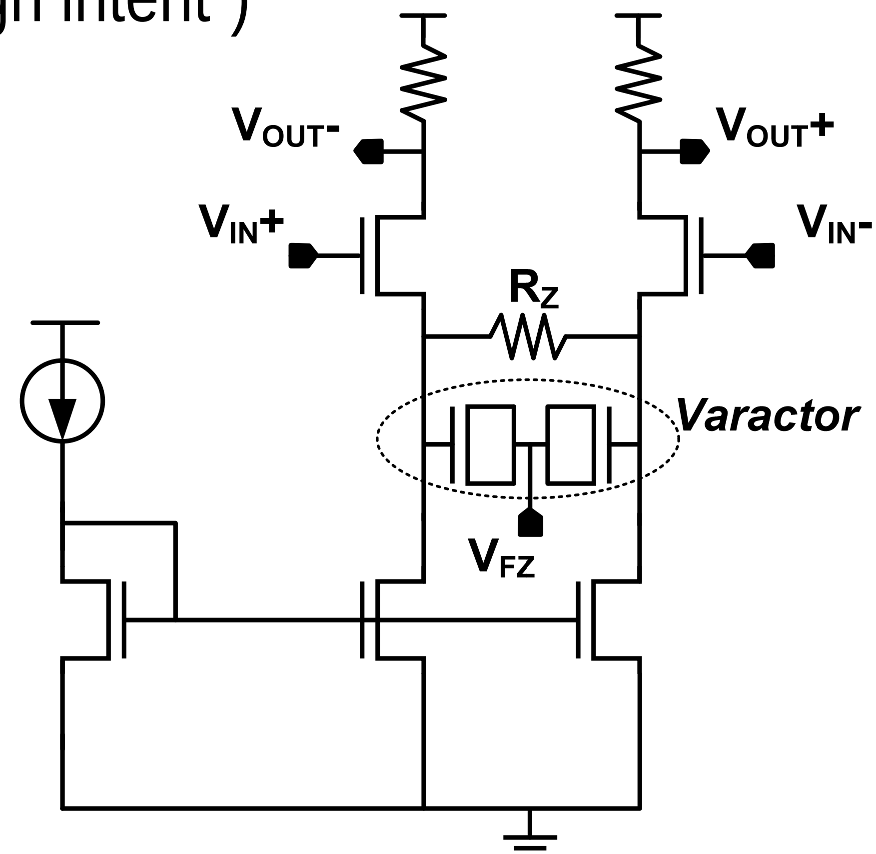
Example: A Linear Equalizer

- Purpose
  - To amplify high-frequency spectrum of the input signal so that the effective channel response becomes equalized
    - A linear filter between input V and output V
  - To make its transfer function characteristic adjustable
    - V_{FZ} controls the zero position (a secondary system)



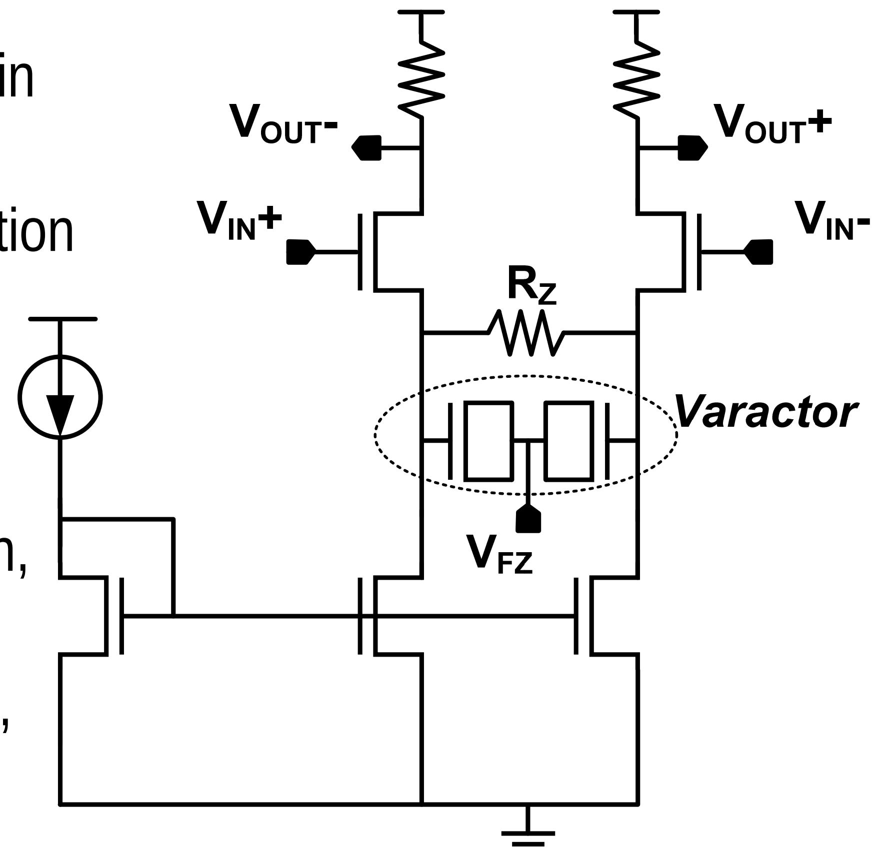
Example: A Linear Equalizer (2)

- Description (how the purpose is being realized in this particular circuit – the “design intent”)
  - The source-degeneration resistor-capacitor pair lowers the low-frequency gain without affecting the high-frequency gain
  - The boundary between the low- and high-frequency is set by their RC product



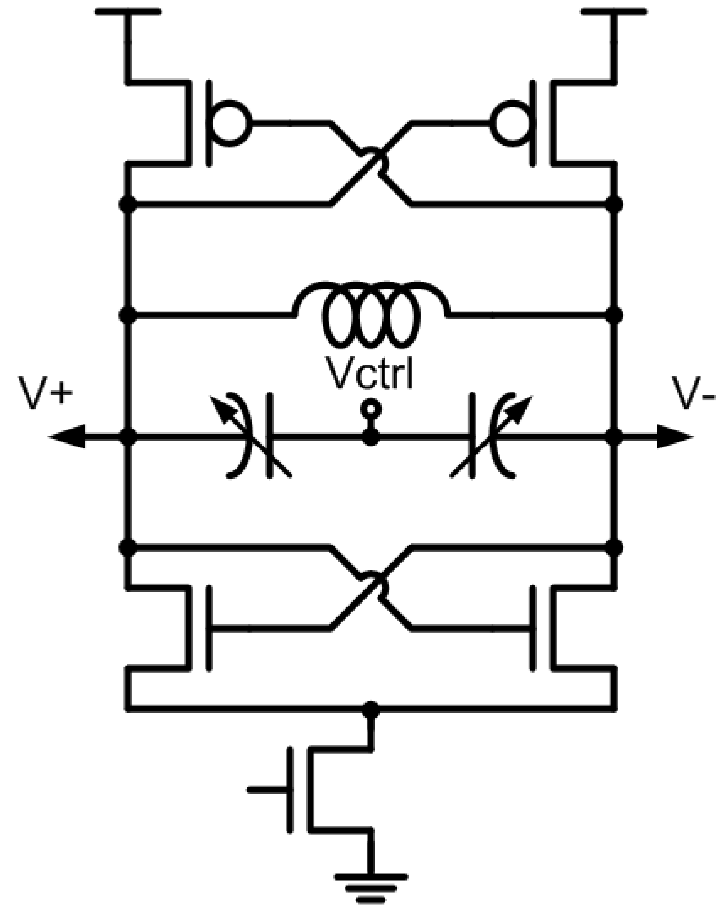
Example: A Linear Equalizer (3)

- Tests
  - Functionality tests
    - Vin-to-Vout frequency-domain transfer function
    - V_{FZ}-to-zero DC transfer function
  - Performance tests
    - Linear: gain, -3dB BW, CMRR, PSRR
    - Nonlinear: -1dB compression, input common-mode range
    - Resource: power dissipation, area



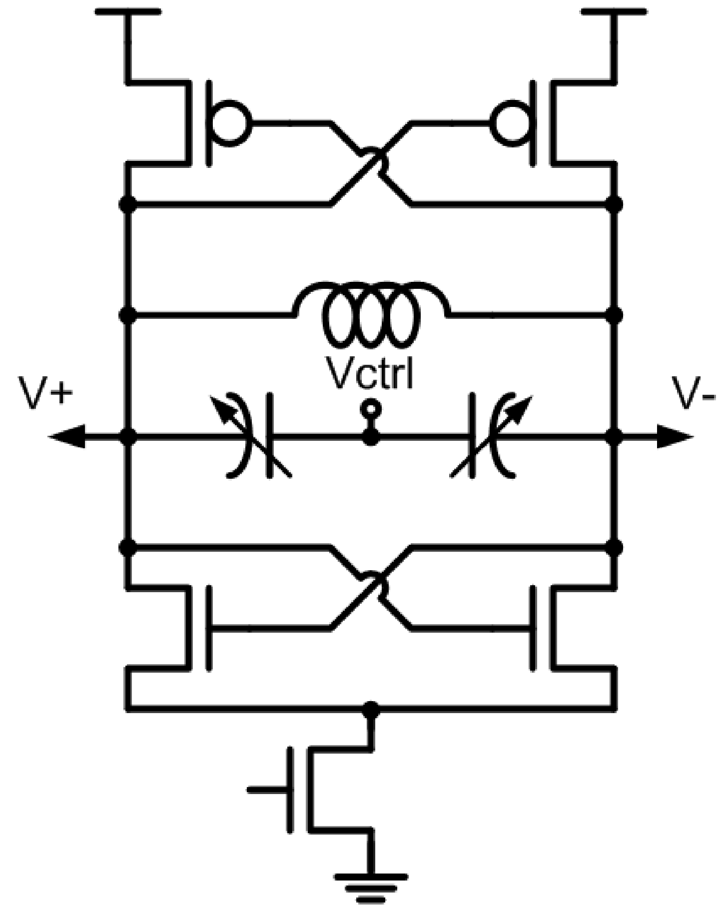
Example: Voltage-Controlled Oscillator

- Purpose
  - To provide a periodic clock whose frequency can be controlled by a voltage input Vctrl
- Description
  - A clock is generated by an LC resonant tank whose loss is compensated by active –Gm circuit
  - Frequency is controlled by varying the capacitance (varactor)



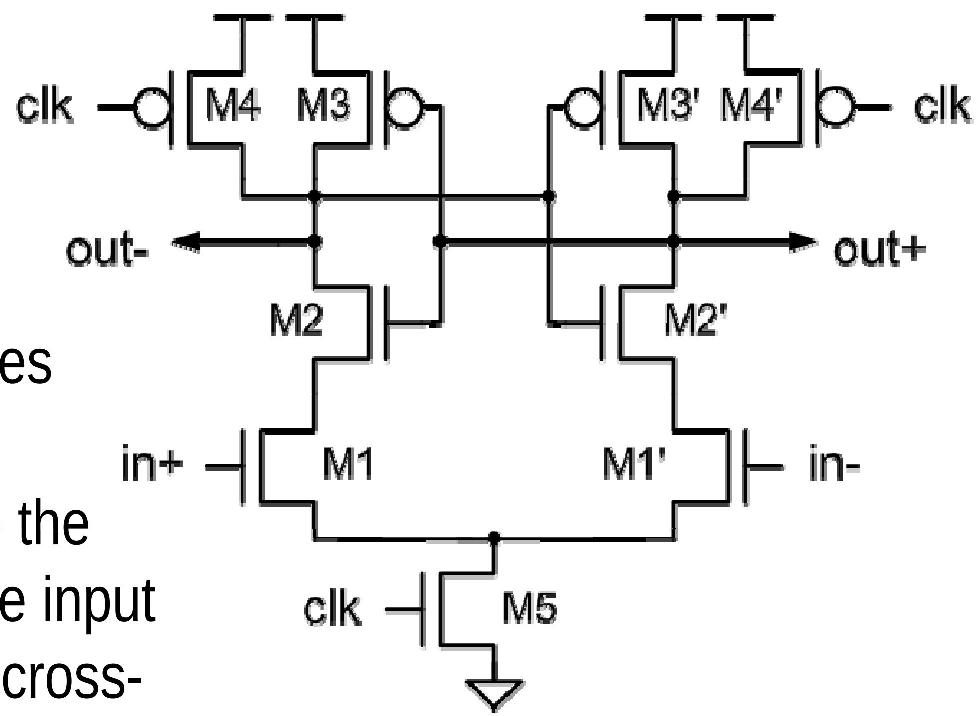
Example: Voltage-Controlled Oscillator

- Tests
  - Functionality tests
    - See if the output is a clock
    - Vctrl-to-frequency transfer function
  - Performance tests
    - Phase noise/jitter
    - Frequency tuning range
    - Output swing
    - Design guides: ISF/PPV, NMF, start-up margin



Example: Voltage Comparator

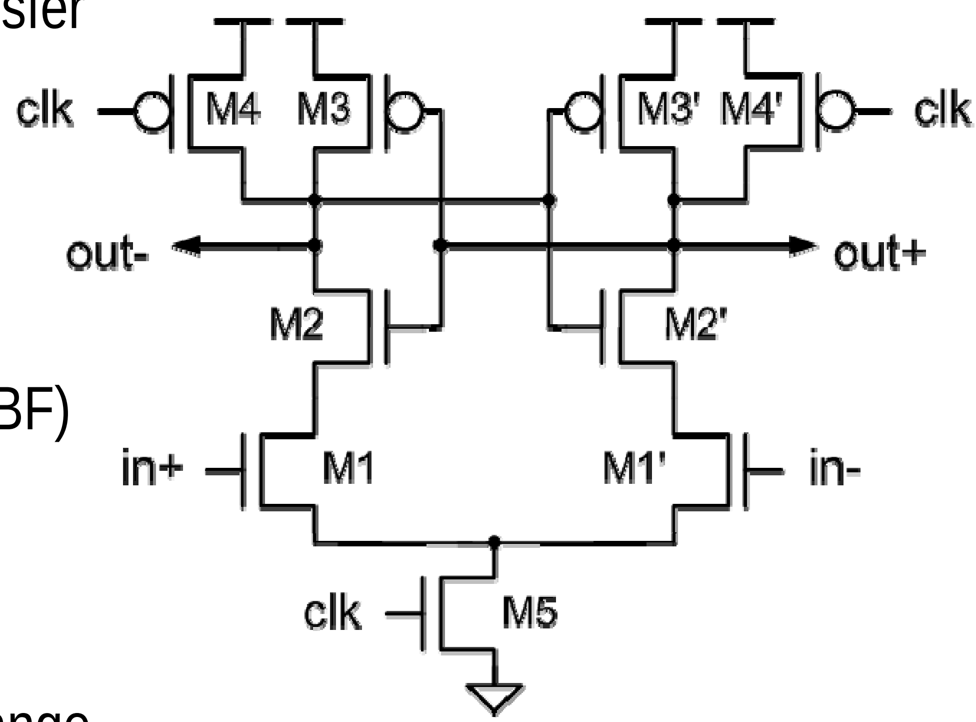
- Purpose
  - Sample the input voltage difference at the rising edge of clk and determine whether it's positive or negative



- Description
  - Precharge the output nodes while clk is low
  - When clk rises, discharge the output nodes based on the input voltages and activate the cross-coupled inverters to regenerate the difference

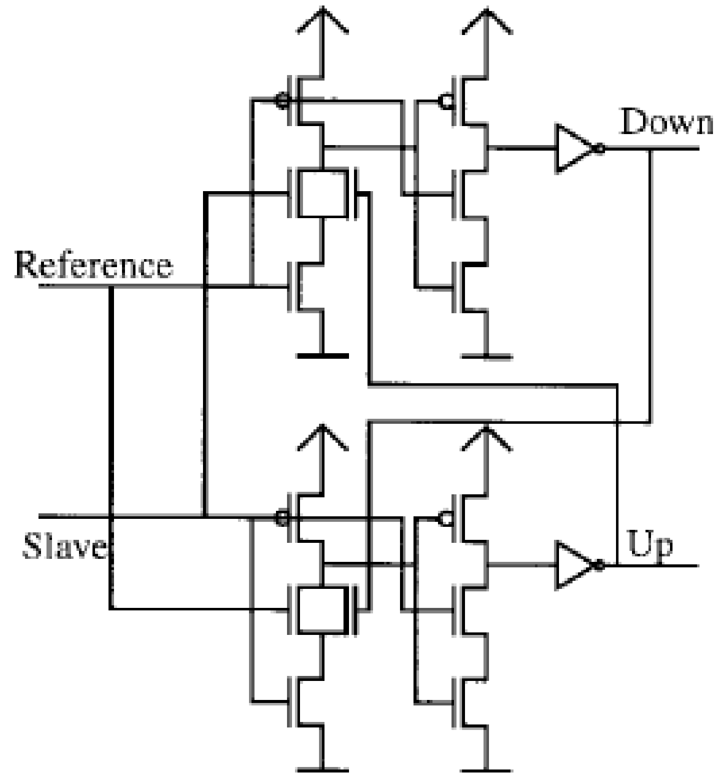
Example: Voltage Comparator (2)

- Tests
  - Functionality tests:
    - Input-to-output DC transfer
  - Performance tests:
    - Input-referred offset
    - Hysteresis
    - Regeneration gain, metastability error (MTBF)
    - Sampling aperture and bandwidth
    - Input referred noise
    - Input common-mode range
    - Power, area, …



Example: Phase-Frequency Detector

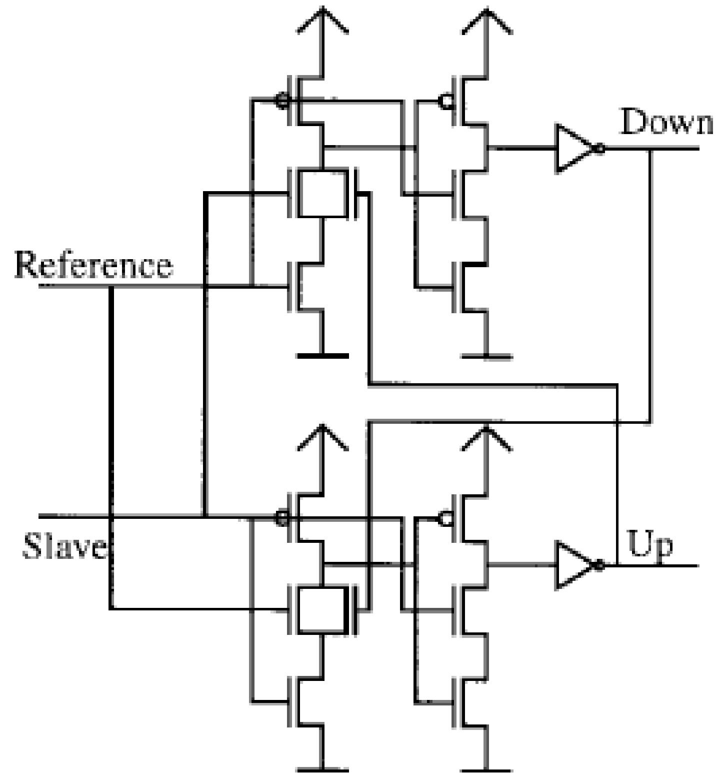
- Purpose
  - Measure the phase (or time) difference between the two input clock events
  - Also detect the frequency difference (i.e. accumulated phase errors)
- Is PFD an analog circuit or digital?
  - Analog functionality
  - Digital implementation



H. Kondoh, H. Notani, T. Yoshimura, H. Shibata, and Y. Matsuda, "A 1.5-V 250-MHz to 3.0-V 622-MHz operation CMOS phase-locked loop with precharge type phase-detector," *IEICE Trans. Electron.,* Apr. 1995.

Example: Phase-Frequency Detector (2)

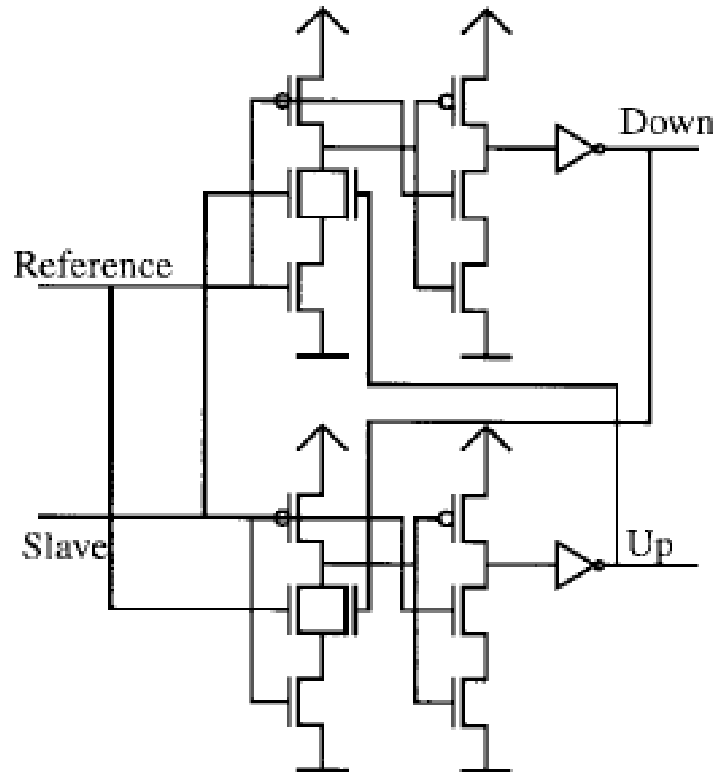
- Description
  - Expresses the output by the Up/Down pulse widths
  - The second dynamic gate: the output rises when the input's rising edge arrives
  - The first dynamic gate
    - resets the outputs when both the input edges have arrived
    - gets ready for the next rising edge while the input is low



H. Kondoh, H. Notani, T. Yoshimura, H. Shibata, and Y. Matsuda, "A 1.5-V 250-MHz to 3.0-V 622-MHz operation CMOS phase-locked loop with precharge type phase-detector," *IEICE Trans. Electron.,* Apr. 1995.

Example: Phase-Frequency Detector (3)

- Tests
  - Functionality tests
    - Phase error to output pulse width transfer function
    - Frequency error to average output transfer function
  - Performance tests
    - Phase detector offset
    - Phase detector hysteresis
    - Phase detector deadzone
    - Max operating frequency
    - Min operating frequency
    - Power, area, …



H. Kondoh, H. Notani, T. Yoshimura, H. Shibata, and Y. Matsuda, "A 1.5-V 250-MHz to 3.0-V 622-MHz operation CMOS phase-locked loop with precharge type phase-detector," ***IEICE Trans. Electron.,*** **Apr. 1995.**

Assignment #2

1. Choose a class of circuit components, for example:
   - Amplifiers or filters
   - Voltage/current references
   - Nonlinear circuits: mixers, peak detectors, absolute detectors, ...
   - Multi-domain circuits: oscillators, delay lines, phase interpolators, phase detectors, …
   - A/D and D/A converters: comparators
   - Choose a small, simple unit block (not a PLL or ΣΔ ADC)

2. Identify the "purpose" of the circuit
   - Note that a single circuit have multiple purposes (e.g. a diff pair may be used both as a linear amplifier as an logic buffer)
   - As the test suites will differ for each purpose, select one only

Assignment #2 – Cont'd

3. List a set of tests that is required to evaluate the circuit
   - Some tests validate the "functionality" of the circuit
   - Some tests validate the "performance" of the circuit
   - Some tests check if the "non-idealities" are within bounds
   - For each test, describe its "purpose" and "procedures"
   - Literature survey will help – investigate what people measure, analyze, and report in order to claim that their circuit is good

4. Prepare a written report and oral presentation in class
   - The reports will be posted on the website so everyone sees it
   - Presentation should be short and concise (5~10min)
   - Active discussion is strongly encouraged