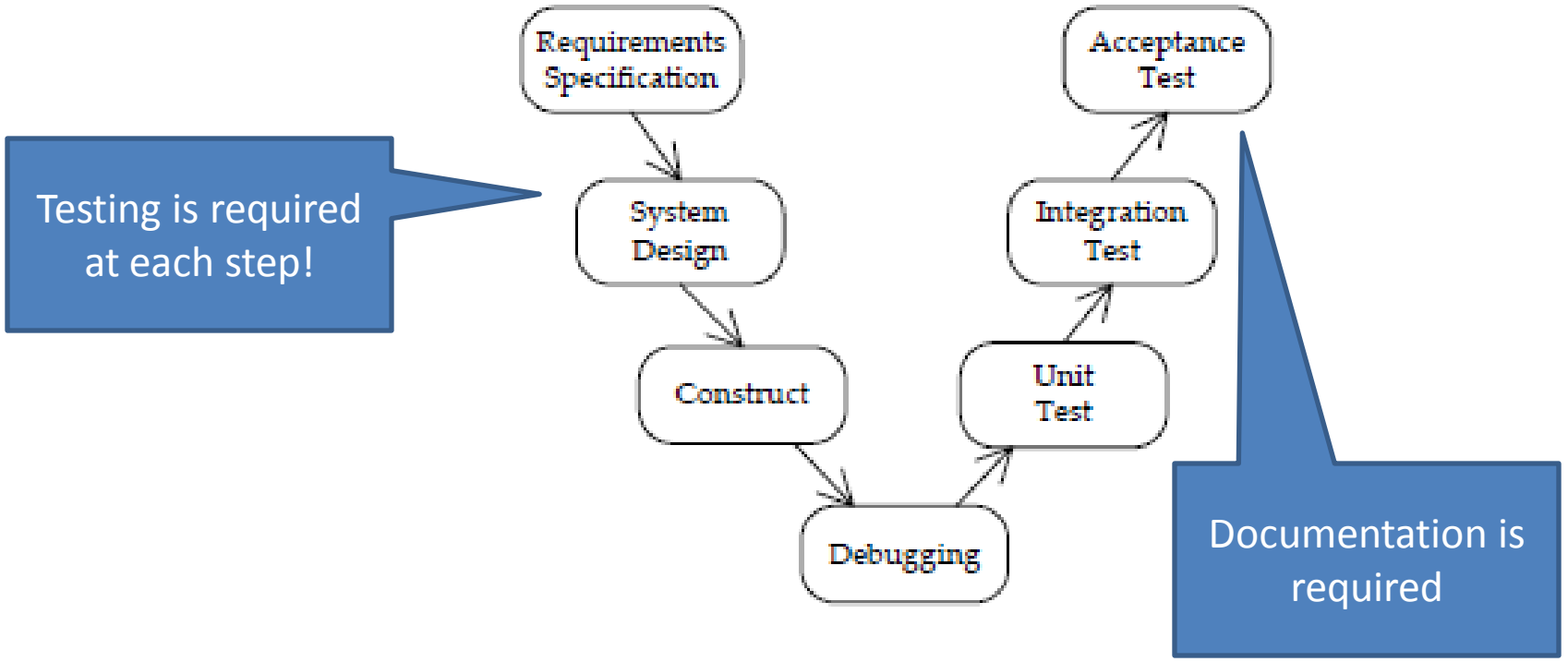
Testing

Motivation

- Development is accompanied by “bugs.”
- Catching bugs early saves money
  - The further a bug progresses the more impact it has on the system
    - A bug fix requires all related modules to be retested.
    - A bug fix may require redesigning related modules.
  - For example
    - PCB design flaw
    - VLSI layout error
    - Subtle coding flaw
- Testing doesn’t remove bugs, it just makes it less likely they exist.

Testing Principles

- Testing proceeds with design process
  - Write tests while designing modules
  - Perform tests while implementing modules
- The Test-Vee illustrates this process



Testing

- Don’t just rely on *Smoke Test*
- Testing
  - Functionality
  - Prevents *Feature Creep*
  - Immediate feedback
  - Extreme Cases
  - Forming a document (describing behavior)
  - Check your design

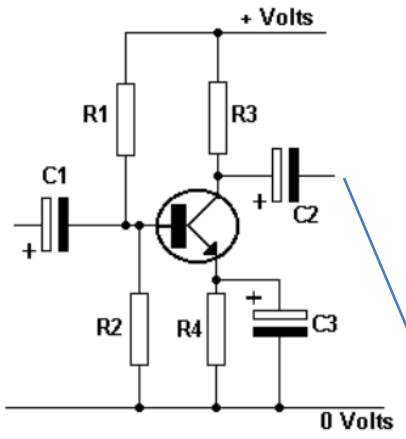
Test Types

- Black box
  - No knowledge of internal organization
  - Only access input and outputs
  - Change inputs and observe outputs
- White box
  - Knowledge of internal organization
  - Might have expectation of fault model
  - Create test instance which reveal physical or logical errors

Example of Testing an Amplifier Using Stub

Black Box Test:
Temp, Vcc, GND

White Box Test:
Temp, Vcc, GND,
Bias Voltages

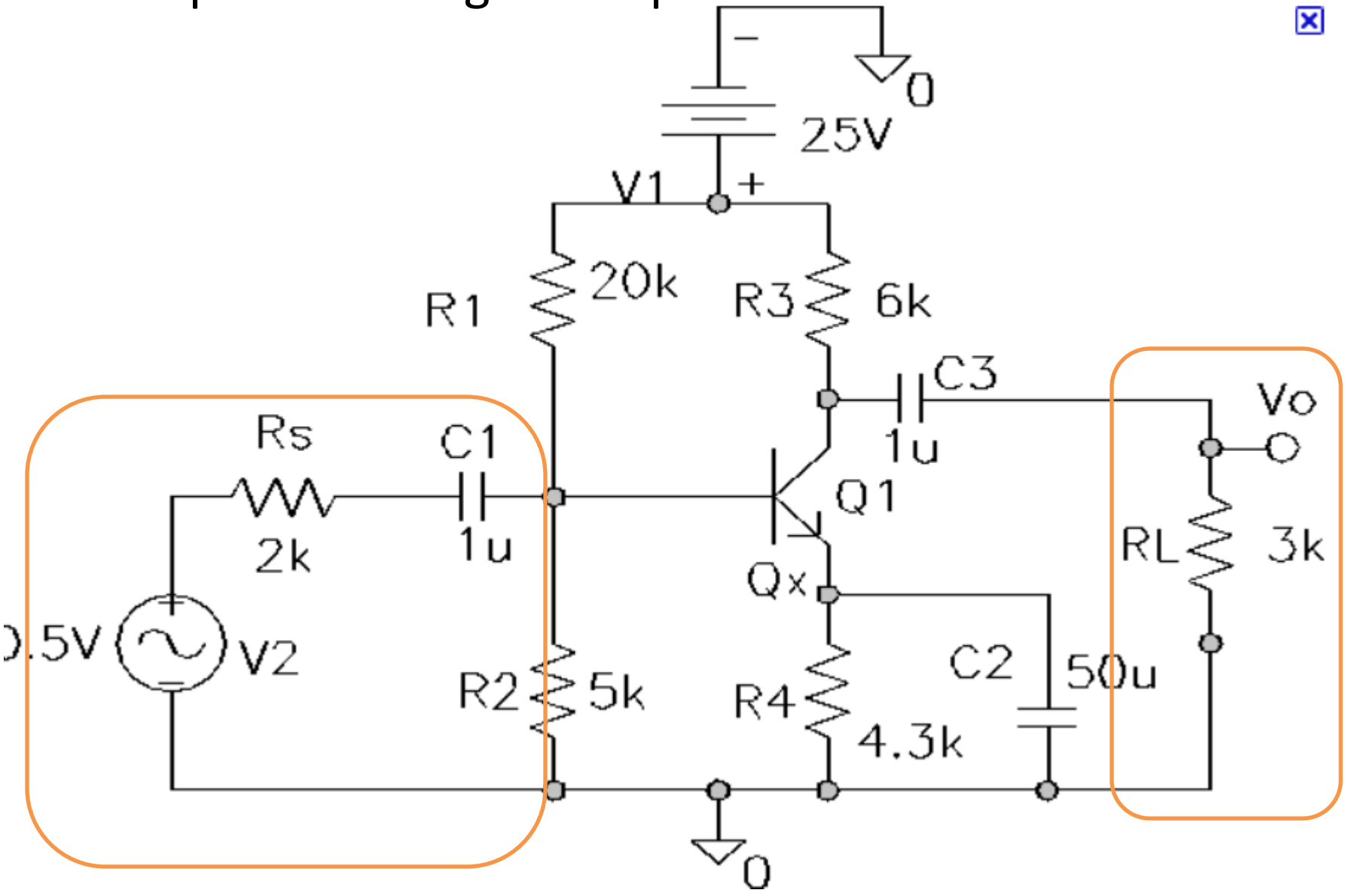


Common Emitter Amplifier
Vo=A x Vi

STUB:
Signal Generator

STUB:
Load

Example of Testing an Amplifier



Test Case Properties

- *Accurate* - The test should check what it is supposed to and exercise an area of intent.
- *Economical* - The test should be performed in a minimal number of steps.
- *Limited in complexity* - Tests should consist of a moderate number (10-15) of steps.
- *Repeatable* - The test should be able to be performed and repeated by another person.
- *Appropriate* - The complexity of the test should be such that it is able to be performed by other individuals who are assigned the testing task.
- *Traceable* - The test should verify a specific requirement.
- *Self cleaning* - The system should return to the pre-test state after the test is complete.

Constructing Tests

- Bugs
  - Bohrbugs (error is in the same place)
  - Heisenbugs (errors move)
- Unit Test
- Integration Test
- Acceptance Test

Testing Process

- Check easiest problems first
  - You can perform more in a given time
- Start at lowest levels of abstraction
  - Upper levels rely on lower level
- Example
  - Is the system powered up?
  - Is the testing equipment adjusted properly?
  - Are the bus lines being correctly manipulated?
  - Have you initialized the system?
  - Are you printing out the right variable/type?

Unit Test

- A ***unit test*** is a test of the functionality of a system module in isolation
- Should be traceable to the detailed design.
- Consists of a set of test cases
- Each test case establish that a subsystem performs a single unit of functionality to some specification.
- Test cases should be written with the express intent of uncovering undiscovered defects.

Test Write-up:
•Matrix
•Automated Scripts
•Step-by-Step

Test Matrix

Test Writer: Sue L. Engineer							
Test Case Name:	ADC function test			**Test ID #:**	ADC-FT-01		
Description:	Verify conversion range and clock frequency. Output goes to 0 in presence of null clock.			**Type:**	☐ white box ☑ black box		
Tester Information							
Name of Tester:				**Date:**			
Hardware Ver:	1.0			**Time:**			
Setup:	Isolate the ADC from the system by removing configuration jumpers.						
Test	**V_T**	**Clock**	**Expected output**		**Pass**	**Fail**	**N/A**	**Comments**
			Decimal	**Hexadecimal**				
1	0.0V	10kHz	0	0x000				
5	2.0V	0Hz	0	0x000				
Overall test result:								

Integration Testing

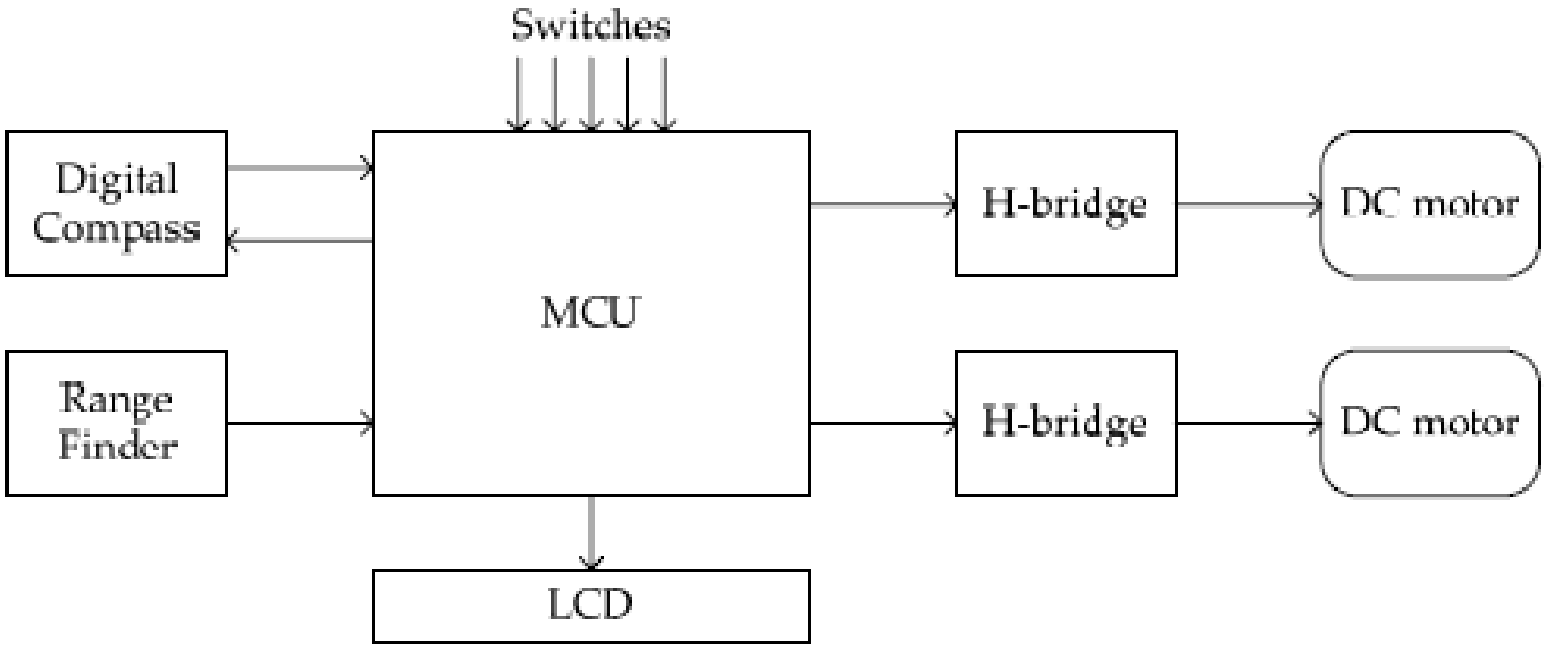
- Write integration test
  - Help insure requirements are being met.
  - Help to firm-up design
  - Requires the designer think about the expected behavior of the subsystems.
  - Requires designer to think about extreme behaviors of subsystems.

Acceptance Test

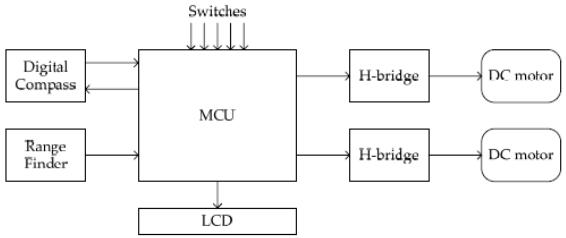
- Formal legal document
- Written along with requirements
- Traceable to engineering requirements
- Identifies
  - Scope – how much of the system is tested?
  - Level – how deep will testing be performed?

Example: Robot Architecture

- Autonomous navigating robot
- Engineering requirements
  - It moves parallel to wall within 12' to 18'
  - Heading should not deviate 10 degrees from wall
  - It should operate under a variety of ambient lighting conditions



Example



Unit Test

- MCU (hardware)
- LCD
- Switches
- Compass
- Range finder
- H-bridge
- Motors
- Chassis
- MCU (software)

Integration Test

- MCU + motors + bridge + switches
- Chassis + digital compass + MCU + motors + bridge +LCD
- Chassis + range finder + MCU + motors + bridge

Acceptance Test

- Autonomous navigating robot
- Engineering requirements
  - It moves parallel to wall within 12' to 18'
  - Heading should not deviate 10 degrees from wa
  - It should operate under a variety of ambient lighting conditions

→

- Verification of engineering requirements
  - Jack chassis off ground
    - Vary distance of cardboard wall
    - Vary angle of cardboard wall

Assignment (Use the Test Matrix form)

- Develop an acceptance test suite for your project – you must target engineering requirements
- Develop an integration test suite for your project – you must target various high levels of design architecture
- Develop a unit test suite for your project – you must target lowest levels of design