Chapter 7 – System Test

1. **Explain the difference between black box and white box testing. [R]**

   The major difference between these types of testing is knowledge of the internal structure of the circuit. A black box test is performed without knowledge of the system, while white box testing is performed with knowledge of the systems construction. Consequently, black box tests are generally written to check extreme inputs and outputs, and transitional behaviors. White box tests are generally written to force internal nodes of the system to specific values.

2. **Identify a circuit simulator (analog or digital) that you are familiar with. Explain the features of this simulator which increase the observability and controllability of the circuit being simulated. [A]**

   In the VHDL simulator Active HDL a user can explore the design hierarchy during run time and set internal nodes to user defined values. For example, when simulating an arithmetic and logic unit (ALU), the user can supply the top level design unit with data input and control. Then while the ALU is operating on the data, the user can halt the simulator, effectively stopping time. This enhances observability because the user can take their time to observe the state of the system at any point in time. At each level of the design hierarchy, the user can examine the state of any node at that design level; giving the user maximum observability. In addition, the users can override a nodes derived value, giving the user controllability.

3. **A mobile robot is being built. It uses two DC motors in a differential drive configuration, a microcontroller to control movement and an ultrasonic sensor to detect obstacles. The robot is built to wander around without bumping into objects. Explain how stubs could be used in testing to take the place of incomplete subsystems. Be specific. [A]**

   If the DC motors are being controlled using a PWM schema, then when checking the motor control, an O-scope can take the place of the DC motors. Likewise an O-scope can be used to unit test the ultrasonic range finder. Depending on the complexity of the range finder, a function generator could supply a suitable signal to the MCU in order to test the MCU algorithms.

4. **Consider that you have an op amp integrated circuit package, such as the LM741 in Appendix C. What type of testing would be appropriate for testing this device? Write a short test plan for doing so. [A]**

   This question is asking you to test an individual 741 op-amp, not a circuit which might use a LM741. The LM741 is encapsulated inside a ceramic package, consequently its internal nodes are not available for inspection or alternation; it certainly has a low observability. However, its internal organization is given in the technical documentation. Consequently, you could design tests to assert internal nodes of the circuit to particular values and check to see if the output behaves the way that it should for this forced value. For example, the Darlington pair consisting of BJTs Q15 and Q17 can be controlled by manipulating a combination of the inverting input (pin 2) and the offset null (pin 5). The state of the Darlington pair can then be

checked to see if it influences the output (via BJT Q20) in the expected way. Consequently, you would be writing white-box tests for the circuit.

5. Explain under what situations a matrix test is appropriate. [R]

A matrix test is appropriate when the tests are structurally similar but differ in their values. For example, imagine testing a digital adder circuit. You might want to check a few hundred different input combinations. Clearly avoiding any unnecessary redundancy in the explanation of the tests would be welcome. This can be achieved with a matrix test.

6. Explain under what situations a step-by-step test is appropriate. [R]

A step-by-step test is appropriate when the set of tests are structurally different. That is a different procedure must be carried out for each test. For example, you might want to test the avoidance behavior of an autonomous robot using different obstacles. Each obstacle might require a unique setup of the robot and the obstacles. In addition, the robot may be designed to respond differently to the various obstacles. In this case it would be wise to give an explicit step-by-step set of instructions for each test.

7. Consider the stages of unit testing, integration testing, and acceptance testing. For each of these stages, identify the corresponding requirements that each test should be traceable to. [A]

Each stage of testing is associated with its own portion of the design. This can be seen in the Test Vee shown in Figure 7.1. For example acceptance testing is associated with the requirements specification. Consequently, the acceptance test must trace back to the requirements specification developed in section 3.8. Integration testing is associated with the systems design phase of development. Consequently, the high level design defines the functional and behavioral requirements for the modules. The unit test is associated with the construction of the individual units of the system. The requirements for each module are defined in the low level architectural description of the modules in the functional decomposition.

8. Consider the case study robot design in Section 7.3, which presents an acceptance test for the first system requirement. Develop an acceptance test for the second system requirement.

Test Writer: Sue L. Engineer			
Test Case Name:	Robot acceptance test #2	**Test ID #:**	Robot-AT-02
Description:	Checks the engineering requirement: *The robot's heading should never deviate no more than 10 degrees from the wall's axis, while traveling parallel to a straight wall over a 3 meter course.*	**Type:**	☐ white box ☑ black box
Tester Information			
Name of Tester:		**Date:**	
Hardware Ver:	Robot 1.0	**Time:**	
Setup:	Completed robot should be fully charged and placed on 3 meter test track. The track should have a fixed and calibrated orientation.		

Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Write a program to log the robots orientation.	Program should be statically tested to verify accuracy. Should sample wall at a sufficient rate depending on speed.				
2	Put robot on test track, run test, and download data.	The robot should travel down the entire length of the test track and then stop.				
3	Plot test data in a spreadsheet program.	Plot of orientation vs. time should be within 10 degrees 90% of the time.				
	Overall test result:					

9. Consider the case study robot design in Section 7.3. Develop an integration test that demonstrates the combined operation of the DC motors, MCU, and range finder.

Test Writer: Sue L. Engineer			
Test Case Name:	Robot integration test #2	**Test ID #:**	Robot-IT-02
Description:	Checks interaction of DC motors, MCU and the range finder to verify that the robot can be controlled to stay within 12-18″ of a wall. This test is traceable to the 1st requirement.	**Type:**	☐ white box ☑ black box
Tester Information			
Name of Tester:		**Date:**	
Hardware Ver:	Robot 1.0	**Time:**	
Setup:	Lay down two strips of tape parallel to wall, one 12″ and one 18″		

Step	Action	Expected Result	Pass	Fail	N/A	Comments
1	Write program to drive motors to keep robot within 12-18″ from the wall.	Program should be statically tested by running the program and placing the .				
2	Run the test program and visually verify conformance with requirement.	The robot should spend a significant amount of time within the stripes.				
	Overall test result:					

Note, that this test is not very formal. You really don't want to burden your design team with reams of testing especially when the tests may need to be rerun as a result of design changes (e.g. regression testing). This test is to make sure there are no surprises later when the acceptance test for the 1st requirement is run.

10. Consider the case study robot design in Section 7.3. Develop an integration test that demonstrates the combined operation of the digital compass, MCU, and LCD.

Test Writer: Sue L. Engineer						
	Test Case Name:	Robot integration test #3			**Test ID #:**	Robot-IT-03
	Description:	Checks interaction of digital compass, MCU and the LCD to verify that the correct orientation can be displayed within +/- 10 degrees. Traceable to second engineering requirement.			**Type:**	☐ white box ☑ black box
Tester Information						
	Name of Tester:				**Date:**	
	Hardware Ver:	Robot 1.0			**Time:**	
	Setup:	Place robot on turntable from integration test #1.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Write program to display heading information on LCD.	Program should be built from existing unit test code and compile successfully.				
2	Orientate the robot on turntable from 0 to 360 degrees in 10 degree increments.	Record the expected angles against displayed angles looking for significant or systematic deviations.				
	Overall test result:					

11. Consider the case study robot design in Section 7.3. Develop unit tests for range finder, the DC motors, the H-bridges, and the LCD.

Test Writer: Sue L. Engineer						
	Test Case Name:	DC motors unit test #1			**Test ID #:**	Motor-UT-01
	Description:	Checks that the DC motors are functional.			**Type:**	☐ white box ☑ black box
Tester Information						
	Name of Tester:				**Date:**	
	Hardware Ver:	DC motor			**Time:**	
	Setup:	DC motors should be placed in test harness (robot chassis if available) or attached to lab bench.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Turn power supply off, set voltage to zero, connect leads to motor.	Wires should be connected to ensure no shorts occur.				
2	Turn on power supply.	Supply should not overload. Motor should not run.				
3	Increase voltage	Motor speed should increase				

	while checking current.	with voltage. Supply current should remain below 200mA.				
4	Switch off power supply, set voltage to zero, and swap polarity of supply.	Load current on the supply should increase				
5	Turn on power supply.	Supply should not overload. Motor should not run.				
6	Increase voltage while checking current.	Motor speed should increase with voltage in opposite direction from step 3. Supply current should remain below 200mA.				
	Overall test result:					

This test merely checks if the motors are functioning correctly. Students could also check if the torque were sufficient for their application or document the power consumption under different loads in order to test if their H-bridges were going to be able to handle the loads the motors would be putting them under.

Test Writer: Sue L. Engineer						
	Test Case Name:	H-bridge unit test #1			**Test ID #:**	HB-UT-01
	Description:	Checks that the H-bridges are functional.			**Type:**	☐ white box ☑ black box
Tester Information						
	Name of Tester:				**Date:**	
	Hardware Ver:	LMD18200 H-bridge			**Time:**	
	Setup:	H-bridge should be plugged into a bread board and fitted with heat-sink. H-bridge should be powered with +25V supply. The PWM input should be connected to a function generator with set to a 0-5V variable duty cycle 100Hz square wave. This output should be traced with Oscope. The H-bridge output should be connected to a 10Watt 25Ohm power resistor through an amp meter.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Turn on the Oscope, then the function generator. Set duty cycle to 0%.	Verify that the PWM signal is 0-5V, 100Hz and has a 0% duty-cycle.				
2	Turn on power supply.	Current draw from power supply should be less than 20mA and the current through power resistor should be less than 10mA.				
3	Increase duty cycle from 0 to 100% by 25% increments.	Current measured at output should increase by 250mA at each step.				
	Overall test result:					

I often refer to these as "hello world" tests. Basically you are doing a quickie test of the chips functionality. This is as much for testing the component as it is about learning how the component works.

Test Writer: Sue L. Engineer						
	Test Case Name:	LCD unit test #1			**Test ID #:**	LCD-UT-01
	Description:	Checks that the LCD is functional and that the MCU can configure and write to LCD.			**Type:**	☐ white box ☑ black box
Tester Information						
	Name of Tester:				**Date:**	
	Hardware Ver:	Optrex LCD			**Time:**	
	Setup:	The LCD should be wired to the MCU using an 8-bit interface.				
Step	**Action**	**Expected Result**	**Pass**	**Fail**	**N/A**	**Comments**
1	Write a program to configure the LCD automatically and write "hello"					
	Overall test result:					

Its unlikely that this test really needs to be documented. This is a task that would be performed in the process of building the software LCD. The more important issue with this test would be writing an effective reusable routines to read and write the LCD in both command and data mode.

12. Project Application. Develop a unit test for your project. The unit tests should apply to the lowest level units in the design. [P]

Getting students to design good unit tests is difficult. They would much rather start in and build the circuit, considering the time spent up front in designing a test as wasted. However, by requiring students to write a set of unit tests before they start to build the circuits you can eliminate time spent building the wrong part. Its important that the students correctly document the unit tests using the forms shown in Table 7.1 or 7.2. These tests should be written hand-in-hand with the design of the individual units. Since the students have knowledge of the internal organization of the units, these tests should be white-box tests.

13. Project Application. Develop an integration test for your project. The integrations tests should apply to the higher levels of the design architecture. [P]

Integration tests should be written at the same time as the high level architecture is being defined. Consequently, the students are writing black-box tests. These tests should be checking that the interfaces operate correctly with extreme inputs, transitional inputs, and erroneous inputs. We frequently find that groups have to make significant design changes mid-way through the term, long after completing their high level design. Having the students work their top-level design by writing integrations tests helps catch these design snafus earlier in the design cycle.

14. Project Application. Develop an acceptance test for your project. The acceptance tests should apply to the engineering requirements developed for the system. [P]

The acceptance test should be one of the first steps of the design process because they are the requirements documentation. However, developing explicit tests on how the students groups will demonstrate these requirements is an overlooked portion of the design process.