Embedded Systems - Unit 5
Validation and Debugging

1 Host and Target Systems

Host

The host system refers to the PC or workstation where development tools are installed and executed.

Target

The target system is the actual embedded hardware on which the software runs.

Testing Overview

- Host: for hardware-independent code and simulation.
- Target: for hardware-dependent testing.

2 Host Testing

Methods

- Manual Testing
- Automated Testing
- Regression Testing
- White Box Testing
- Functional Testing

Concept

Testing occurs on the host by simulating the SUT environment. Hardware dependencies are replaced by test stubs.

3 Manual Testing

- Human testers execute test cases.
- Useful for quick, exploratory feedback.
- Prone to human error.

Advantages

- Quick feedback
- Flexibility
- Exploratory edge-case discovery

4 Automated Testing

- Uses tools/scripts to execute tests.
- Requires setup effort but allows consistency.
- Supports regression testing.

Advantages

- Reduced test time
- Increased coverage
- Repeatable results

Example

Given $n = 4$, 2401 input combinations (e.g., $0, 1, 2, 100/2, 100 - 1, 100, 101$). Automating saves time.

5 Regression Testing

- Verifies past bugs do not reappear.
- Performed after bug fixes or new releases.
- Enables sanity checks and detecting recurring issues.

6 White Box Testing

- Code-level testing focused on internal paths.
- Requires understanding of code.
- Performed post-unit testing.

Limitations

- May miss missing-code faults.
- Doesn't suit timing/hardware/load testing.

7 Functional (Black Box) Testing

- Based on system requirements.
- Independent of code structure.
- Suitable for nonfunctional testing.

Limitations

- Ignores code coverage.
- Less effective for detecting deep logic bugs.

8 Limitations of Host Testing

1. No real-time behavior testing.
2. Cannot detect peripheral/memory faults.
3. Shared data and timing issues are hard to detect.

9 Target Testing Tools and Methods

ROM Emulator

Replaces ROM with fast RAM. Allows rapid updates and debugging.

Remote Debugger

- **Frontend:** UI on host to control testing.
- **Backend:** Runs on target to execute commands.

Debug Kernel

Requires:
- Interrupt vector for debugger entry.
- Software interrupt to signal debugger.

Implements run control, breakpoints, memory inspection.

Logic Analyser

- Captures pin signals.
- Triggered tracing.
- Measures logical states only (1/0).

BDM (Background Debug Mode)

- Motorola's on-chip debug interface.
- Uses wiggler to connect host and target.
- Allows tracing, breakpoints, and memory access.

PROM Programmer

- Burns software image into (E)PROM.
- Not suitable for frequent updates.
- Erasable types (EPROM) are reusable.

Source-level Debugger

- Debugger runs on host.
- Loads and runs target image in RAM.
- Allows breakpoints, single-step, monitor registers.

JTAG

- Controls boundary pins for testing.
- Requires JTAG-compliant processor.
- Standardized as IEEE 1149.1.

In-Circuit Emulator (ICE)

- Replaces processor with test equipment.
- Allows tracing, breakpoints, overlays.
- GUI support and real-time debug.

10 Advanced Target Testing Features

Bullet-Proof Run Control

- ICE substitutes processor and memory.
- Uses NMI control and shadow memory.
- Maintains debug access despite faulty memory.

Real-Time Trace

- Integrates logic analyzer.
- Uses trigger signals for precise entry.
- Allows synchronized trace and debug.

Hardware Breakpoints

- Trigger breakpoints using logic analyzer.
- Monitor memory access or specific functions.
- Enables ISR debugging, e.g., ‘foo()‘ function corruption.

Overlay Memory

- Replaces ROM with RAM dynamically.
- Allows error detection, test coverage.
- Uses memory-mapping and steering logic.

11 Challenges of Target Testing

1. Incomplete software limits early testing.
2. Hardware and software may not be ready simultaneously.
3. Hard to simulate regression on target.
4. Real-time systems lack trace storage.
5. Needs specialized debug hardware/software.