Testiranje elektronskih sistema (1)
System Design at a Crossroad

• Productivity Crisis
  – 21% Productivity Increase / Year vs.
  – 58% Complexity Increase / Year
Electronic Design Automation (EDA)

- System on a Chip development:
  
  *Yesterday’s chip is today’s functional block!*  

  New design methodologies are needed

---

Source: ICE
Testing and Quality

How much to test?

“The problem of testing can only be contained, not solved”

T. Williams
The Role of Test

Dependability

Security  Safety

Environment

Reliability

Fault-Tolerance  Fault Diagnosis  Test  BIST

System

Design for testability:

Test  Diagnosis
How Much to Test?

**Amusing Test:**

**Paradox 1:**
Digital model is finite, analog model is infinite.

However, the complexity problem was introduced by Digital World

**Paradox 2:**
If I can show that the system works, then it should be not faulty.

But, what does it mean: it works?

32-bit accumulator has $2^{64}$ functions which all should work.

So, you should test all of them!
How Much to Test?

Paradox:
2^{64} input patterns (!)
for 32-bit accumulator
will be not enough.
A short will change the circuit
into sequential one,
and you will need because of that
2^{65} input patterns

Paradox:
Mathematicians counted that Intel 8080
needed for exhaustive testing 37 (!) years
Manufacturer did it by 10 seconds
Majority of functions will never activated
during the lifetime of the system

Time can be your best friend
or your worst enemy
(Ray Charles)
The Problem is Money?

Cost of testing
Cost of quality
Cost of the fault

How to succeed?
Try too hard!
How to fail?
Try too hard!
(From American Wisdom)

Conclusion:
“The problem of testing can only be contained not solved”

T. Williams
Hierarchy

*Paradox:*

To generate a test for a block in a system, the computer needed 2 days and 2 nights. An engineer did it by hand with 15 minutes.

So, why computers?
System testing
System testing

- Testing the system as a whole to validate that it meets its specification and the objectives of its users.
Development testing

- Hardware and software components should be tested as they are developed and as sub-systems are created. These testing activities include:
  - Unit testing.
  - Module testing
  - Sub-system testing
- However, these tests cannot cover:
  - Interactions between components or sub-systems where the interaction causes the system to behave in an unexpected way
  - The emergent properties of the system
System testing

• Intended to test the system as a whole rather than individual system components
  – Integration testing
    • As the system is integrated, it is tested by the system developer for specification compliance
  – Stress testing
    • The behaviour of the system is tested under conditions of load
  – Acceptance testing
    • The system is tested by the customer to check if it conforms to the terms of the development contract

• Inevitably, system testing reveals errors which were undiscovered during component testing
Integration testing
Integration testing

• Concerned with testing the system as it is integrated from its components
• *Integration testing is normally the most expensive activity in the systems integration process*
• Should focus on
  – Interface testing where the interactions between sub-systems and components are tested
  – Property testing where system properties such as reliability, performance and usability are tested
What is integration testing

• Testing in which software components, hardware components, or both together are combined and tested to evaluate interactions between them.

• Integration testing usually go through several realword business scenarios to see whether the system can successfully complete workflow tasks

• *Integration plan specifies the order of combining the modules into partial systems*
Integration test planning

• Integration testing is complex and time-consuming and planning of the process is essential. The larger the system, the earlier this planning must start and the more extensive it must be.

• Integration test planning may be the responsibility of a separate V & V (verification and validation) team. It is good to be undertaken by a group which is separate from the development team.
Test planning activities

- Identify possible system tests using the requirements document
- Prepare test cases and test scenarios to run these system tests
- Plan the development, if required, of tools such as simulators to support system testing
- Prepare, if necessary, operational profiles for the system
- Schedule the testing activities and estimate testing costs
Testing Activities

Subsysstem Code → Unit Test

Subsysstem Code → Unit Test

Subsysstem Code → Unit Test

Unit Test → Integration Test

System Design Document → Integration Test

Tested Subsystem → Integration Test

Requirements Analysis Document → Functional Test

User Manual → Functional Test

Integration Test → Functional Test

Integrated Subsystems → Functional Test

Tested Subsystem → Functional Test

Functioning System → Functional Test

Testing Activities
Component / Module testing

• A unit is the smallest testable piece. In software, a unit is defined as a database trigger, stored procedure, function, report, form, batch load program or PL/SQL program.

• Unit testing is the testing that is performed to validate that the unit does satisfy its functional specification and/or that its implementation structure does match the intended design structure.

• Module /Component consists of units integrated into a module that performs a specific business function
Drivers and Stubs

- **Driver:** A program that calls the interface procedures of the module being tested and reports the results.

  - A driver simulates a module that calls the module currently being tested

- **Stub:** A program that has the same interface procedures as a module that is being called by the module being tested but is simpler.

  - A stub simulates a module called by the module currently being tested
Drivers and Stubs

Driver → Module Under Test → Stub

procedure call → procedure call
How to test this system??

Analysis and Control Software

Embedded Controller

Com Port

Patients Database

Heart support device
Artifacts and Roles for Integration Testing

- Test engineer
  - Use-case model
  - Test plan
  - Test case
  - Test procedure
  - Test evaluation
  - Test component

- Component engineer
- Integration tester
- System tester
- Defect management
What we have here...

- Developers do everything they can to make the software work.
- Testers do everything they can to make the software fail.
Integration testing approaches

• Different approaches to integration testing
  – Bottom-up
  – Top-down
  – Big-bang
  – Sandwich
Bottom-up Integration
Bottom-Up Integration

- Only terminal modules are tested in isolation
- Modules at lower levels are tested using the previously tested higher level modules
- This is done repeatedly until all subsystems are included in the testing
- Requires a module driver for each module to feed the test case input to the interface of the module being tested.
- However, stubs are not needed since we are starting with the terminal modules and use already tested modules when testing modules in the lower levels
- **Disadvantage:** Tests the most important subsystem last!
Top-down Integration
Top-down Integration

• Only modules tested in isolation are the modules which are at the highest level
• After a module is tested, the modules directly called by that module are merged with the already tested module and the combination is tested
• Do this until all subsystems are incorporated into the test
• Requires stub modules to simulate the functions of the missing modules that may be called.
• However, drivers are not needed since we are starting with the modules which is not used by any other module and use already tested modules when testing modules in the higher levels
Sandwich Testing Strategy

• Combines top-down strategy with bottom-up strategy
• *The system is view as having three layers*
  – A target layer in the middle
  – A layer above the target
  – A layer below the target
  – Testing converges at the target layer
• How do you select the target layer if there are more than 3 layers?
  – Heuristic: Try to minimize the number of stubs and drivers
• *Top and Bottom Layer Tests can be done in parallel*
• *Does not test the individual subsystems thoroughly before integration*
Other Approaches to Integration

• **Big Bang Integration**
  – Every module is unit tested in isolation
  – After all of the modules are tested they are all integrated together at once and tested
Implementation Approaches
Advantages and Disadvantages:

• Bottom-up advantages
  – Separately debugged modules
  – System test by integrating previously debugged modules
  – Testing upper-level modules is easier

• Bottom-up disadvantages
  – Drivers must be written
  – Upper-level, critical modules are built last
  – Drivers are more difficult to write than stubs
  – User interfaces are top-level modules
Implementation Approaches
Advantages and Disadvantages:

• Top-down advantages
  – Separately debugged modules
  – System test by integrating previously debugged modules
  – Stubs are easier to code than drivers
  – User interfaces are top-level modules

• Top-down disadvantages
  – Stubs must be written
  – Low-level, critical modules built last
  – Testing upper-level modules is difficult
Implementation Approaches
Advantages and Disadvantages:

• Big bang advantages
  - None

• Big bang disadvantages
  - Difficult to debug
  - Much throwaway code
  - Critical and peripheral modules not distinguished
  - User does not see product until very late in the development cycle
Summary of testing

- Verify operation at normal parameter values
  (a black box test based on the unit’s requirements)
- Verify operation at limit parameter values
  (black box)
- Verify operation outside parameter values
  (black box)
- Ensure that all instructions execute
  (statement coverage)
- Check all paths, including both sides of all branches
  (decision coverage)
Summary of testing

- Check the use of all called objects
- Verify the handling of all data structures
- Verify the handling of all files
- Check normal termination of all loops
  (part of a correctness proof)
- Check abnormal termination of all loops
Summary of testing

• Check normal termination of all recursions
• Check abnormal termination of all recursions
• Verify the handling of all error conditions
• Check timing and synchronization
• Verify all hardware dependencies
Interface testing
What is an interface?

- An agreed mechanism for communication between different parts of the system
- System interface classes
  - Hardware interfaces
    - Involving communicating hardware units
  - Hardware/software interfaces
    - Involving the interaction between hardware and software
  - Software interfaces
    - Involving communicating software components or sub-systems
  - Human/computer interfaces
    - Involving the interaction of people and the system
  - Human interfaces
    - Involving the interactions between people in the process
Hardware interfaces

• Physical-level interfaces
  – Concerned with the physical connection of different parts of the system e.g. plug/socket compatibility, physical space utilisation, wiring correctness, etc.

• Electrical-level interfaces
  – Concerned with the electrical/electronic compatibility of hardware units i.e. can a signal produced by one unit be processed by another unit

• Protocol-level interfaces
  – Concerned with the format of the signals communicated between hardware units.
Software interfaces

- Parameter interfaces
  - Software units communicate by setting pre-defined parameters
- Shared memory interfaces
  - Software units communicate through a shared area of memory
  - Software/hardware interfaces are usually of this type
- Procedural interfaces
  - Software units communicate by calling pre-defined procedures
- Message passing interfaces
  - Software units communicate by passing messages to each other
Parameter interfaces

SS2

Parameter list

SS1
Shared memory interfaces

SS1  SS2  SS3

Shared memory area
Procedural interfaces

Defined procedures (API)
Message passing interfaces

SS2

Exchanged messages

SS1
Interface errors

- Interface misuse
  - A calling component calls another component and makes an error in its use of its interface e.g. parameters in the wrong order

- Interface misunderstanding
  - A calling component embeds assumptions about the behaviour of the called component which are incorrect

- Timing errors
  - The calling and called component operate at different speeds and out-of-date information is accessed
Stress testing
Stress testing

• Exercises the system beyond its maximum design load
  – The argument for stress testing is that system failures are most likely to show themselves at the extremes of the system’s behaviour
  – Tests failure behaviour
  – When a system is overloaded, it should degrade gracefully rather than fail catastrophically
• Particularly relevant to distributed systems
  – As the load on the system increases, so too does the network traffic. At some stage, the network is likely to become swamped and no useful work can be done
Acceptance testing
Acceptance testing

- The process of demonstrating to the customer that the system is acceptable
- Based on real data drawn from customer sources. The system must process this data as required by the customer if it is to be acceptable
- Generally carried out by customer and system developer together
- May be carried out before or after a system has been installed
Performance testing

• Concerned with checking that the system meets its performance requirements
  – Number of transactions processed per second
  – Response time to user interaction
  – Time to complete specified operations
• Generally requires some logging software to be associated with the system to measure its performance
• May be carried out in conjunction with stress testing using simulators developed for stress testing
Reliability testing

- The system is presented with a large number of ‘typical’ inputs and its response to these inputs is observed.
- The reliability of the system is based on the number of incorrect outputs which are generated in response to correct inputs.
- The profile of the inputs (the operational profile) must match the real input probabilities if the reliability estimate is to be valid.
Security testing

- Security testing is concerned with checking that the system and its data are protected from accidental or malicious damage.
- Unlike other types of testing, this cannot really be tested by planning system tests. The system must be secure against unanticipated as well as anticipated attacks.
- Security testing may be carried out by inviting people to try to penetrate the system through security loopholes.
Key points

• System testing involves testing the system as a whole. It is a critical activity in systems integration

• Systems testing includes interface testing, stress testing and acceptance testing

• Advance planning of integration testing is essential and may be the responsibility of a separate group

• Interface testing is the principal type of defect testing carried out at this stage