COMS30026 Design Verification

Verification Hierarchy

Kerstin Eder

(Acknowledgement: Avi Ziv from the IBM Research Labs in Haifa has kindly permitted the re-use of some of his slides.)

University of BRISTOL

Department of COMPUTER SCIENCE

Outline

- Observability and Controllability
  - Black box, white box and grey box testing
- Verification hierarchy
  - Levels at which to perform verification

Observability and Controllability

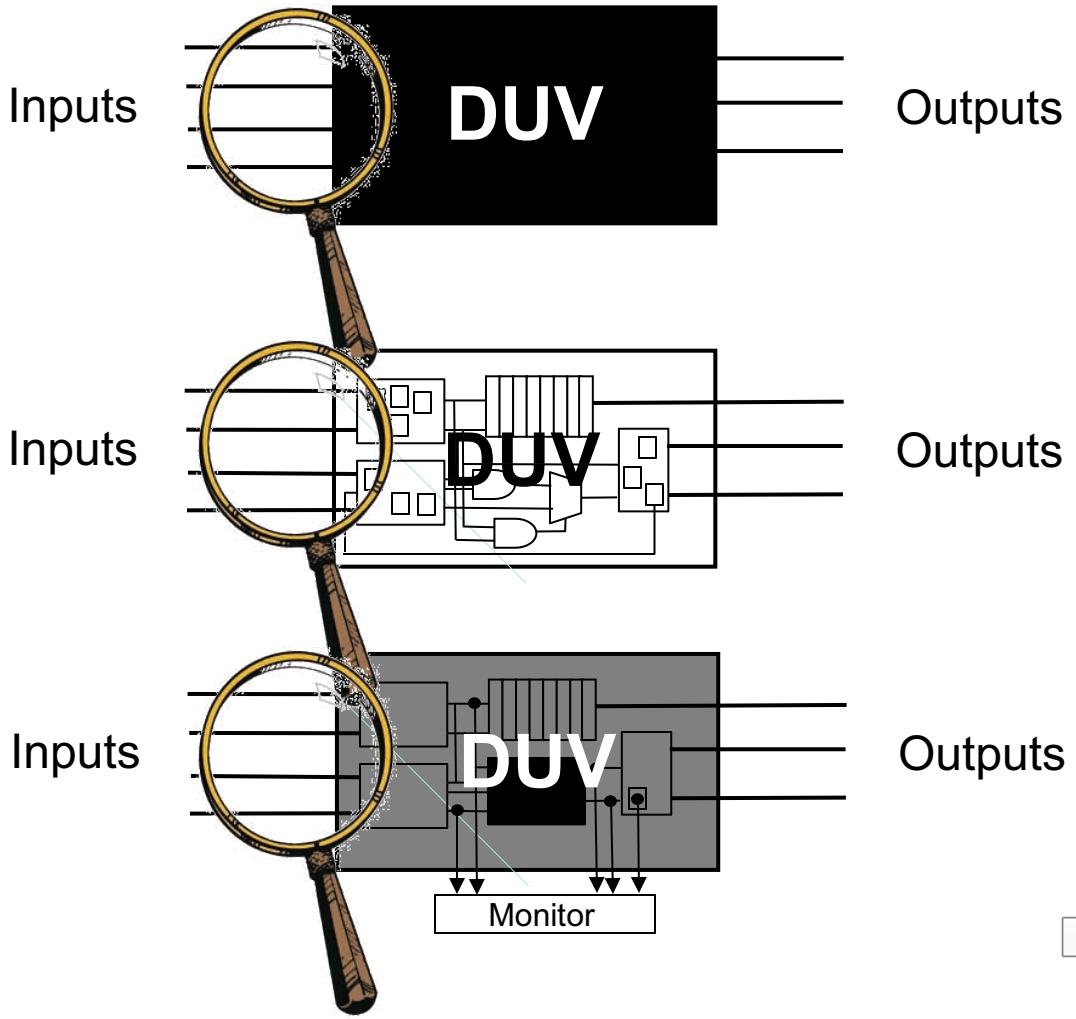
Observability and Controllability

- Observability: Indicates the ease at which the verification engineer can identify when the design acts appropriately versus when it demonstrates incorrect behavior.

- Controllability: Indicates the ease at which the verification engineer creates the specific scenarios that are of interest.

Levels of Observability

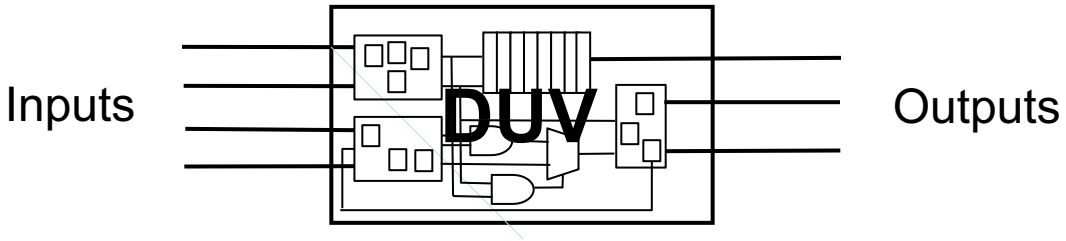
- Black Box
- White Box
- Grey Box



Black Box Verification

- The black box has inputs, outputs, and performs some (well documented) function.
- To verify a black box, you need to **understand the function.**
- The verification code utilizes only the external interfaces.
- The internal signals and state remain in the dark.
- **Pros:**
  - No knowledge of the actual implementation is required.
  - Ability to predict functional results based on inputs alone ensures that the reference model remains independent from the DUV implementation.
  - Verification code is less sensitive to changes inside the DUV.
- **Cons:**
  - Difficult to locate source of problem, only exposes effects. (If at all! Remember, not all bugs propagate to the outputs.)
  - **Lacks controllability and observability.**

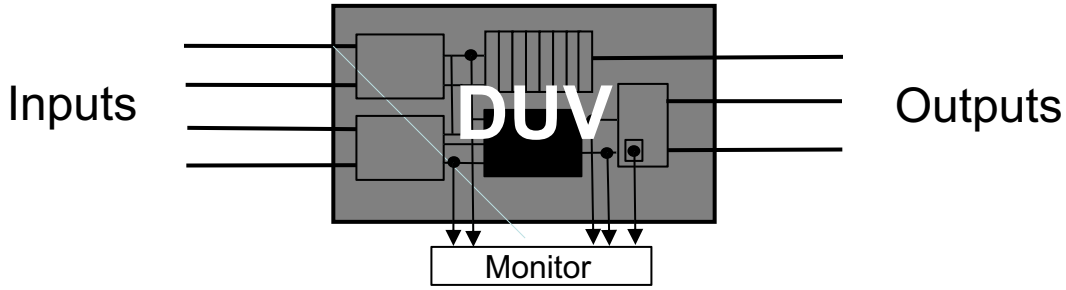
White Box Verification



(Opposite of black-box approach.)

- For white box verification the internal facilities of the DUV are known, visible and utilised for verification.
- **Pros:**
  - **Full visibility and controllability of internal signals.**
    - **Can identify and cover corner cases.**
    - **Can detect bugs as soon as they occur.**
  - Quickly possible to set up interesting conditions, e.g. counter roll-over.
- **Cons:**
  - Danger to follow the implementation/design instead of the specification.
  - Sensitive to changes in the DUV (implementation).
  - Too many details make it hard to create and maintain.

Grey Box Verification



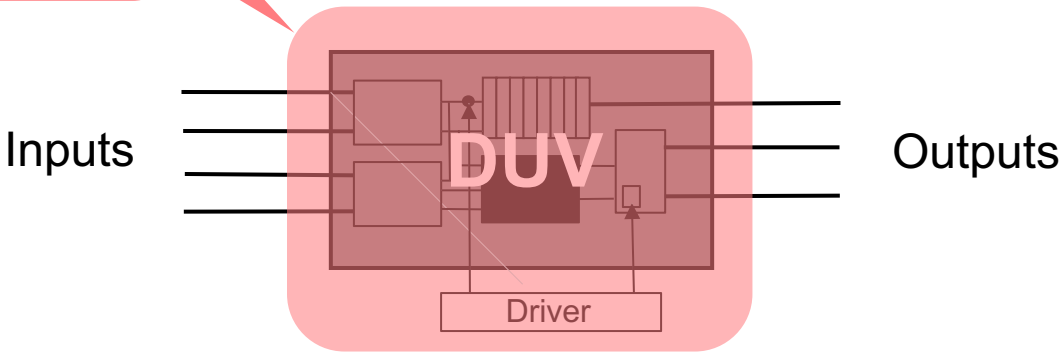
- For grey box verification a limited number of DUV facilities are utilised in a mostly black-box environment.
  - Access important and stable features, the rest is kept in the dark.
- Combines the pros (if done the right way) or the cons (if done the wrong way) of black and white box.
  - Progression from black box to grey box should be carefully planned and started only when the **DUV is sufficiently stable.**
- In practice: **Most verification environments are grey box.**
  - May need to start with black box with planned evolution into grey box.
  - Note: Prediction of correct results on an interface is occasionally impossible without viewing an internal signal.

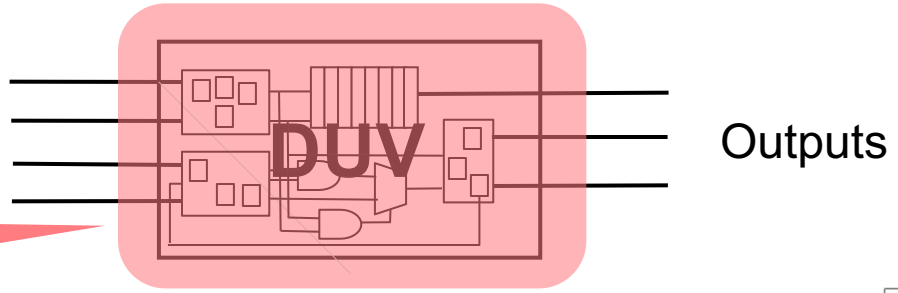
Levels for Controllability

- Black Box

Watch out for unreachable states!

- Grey Box





Watch out for unreachable states!

Be careful with White Box Controllability

- In theory, the same levels as for observability also exist for controllability:
  - black, grey and white box
- In practice:
  - **We seldom control the internals of the DUV.**
  - This may drive the design into a state that is not reachable *under normal circumstances*.
  - It may thus lead to an inconsistent DUV state.
- The main exception: **Warm Loading**
  - Brings the DUV to a predefined interesting state.
    - E.g. cache initialization, almost full buffer
  - Reduces the time needed for reaching this state.

Verification Hierarchy

Verification Hierarchy

- Today's complex chips and systems are divided into logical units
  - Usually determined during specification / high-level design
  - Usually follow the architecture of the system
  - This practice is called hierarchical design
- Hierarchical design allows a designer to subdivide a complex problem into more manageable blocks
  - The design team combines these blocks to form bigger units, and continues to merge/integrate these blocks until the chip or system is complete

Pros and Cons of Hierarchical Design

- Pros
  - Breaks the design into manageable pieces
  - Allow designers to focus on single function / aspect of the design
- Cons
  - More interfaces to specify / design / verify
  - Integration issues

Levels of Verification

- Verification usually adapts to and takes advantage of the hierarchical *design* stages and boundaries
- Common levels of verification
  - Designer level (block level)
  - Unit level
  - (Core level)
  - Chip level
  - System level
  - Hardware / software co-verification

Designer (Block) Level Verification

- Used for verification of single blocks and macros
- Usually, done by the designer him/herself
- Main goal – Sanity checking and certification for a given block
- Ranges from a simple test of basic functionality to complete verification environments
- The common level for formal verification

Unit Level Verification

- A set of blocks that are designed to handle a specific function or aspect of the system
  - E.g., memory controller, floating-point unit
- Usually there is a formalized spec
  - More stable interface and function
- The target of first serious verification effort

Core Level Verification

- A core is a unit or set of units designed to be used across many designs
  - Well defined function
  - Standardized interfaces
- Verification needs to be thorough and complete
  - Address all possible uses of the core
- The verification team can use “Verification IP” for the standardized interfaces

Chip Level Verification

- Verification of a set of units that are *packaged* together in a physical entity
- Main goals of verification
  - Connection and integration of the various units
  - Verify functions that could not be verified at lower levels
- Need verification closure to avoid problems at tape-out

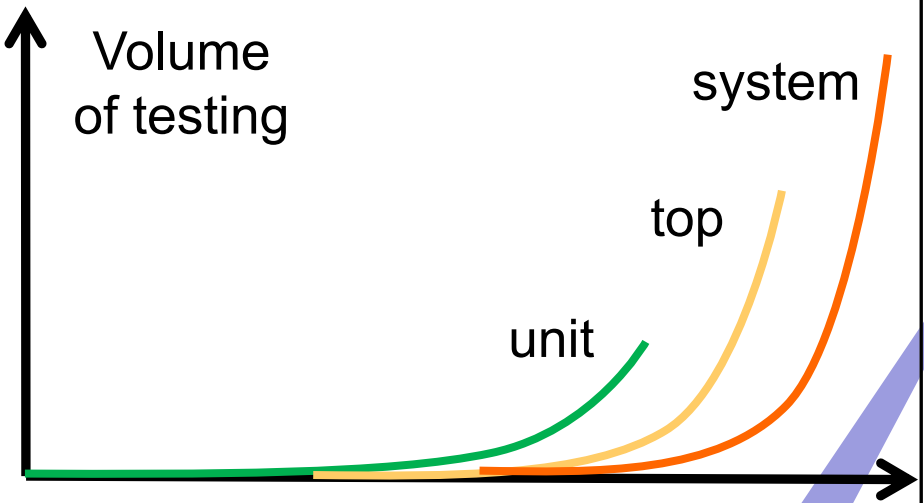
System Level Verification

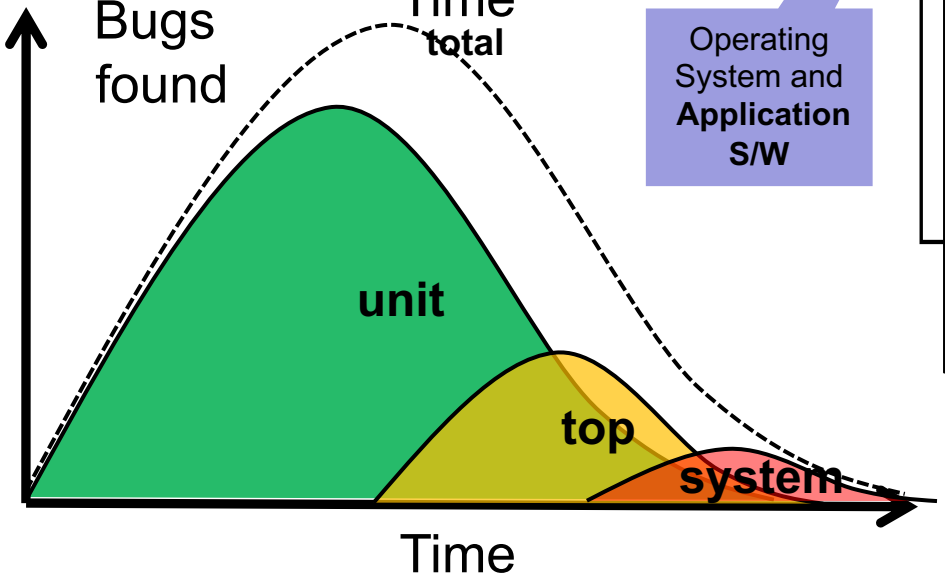
- The purpose of this level of verification is to confirm
  - Interconnection
  - Integration
  - System design
- Verification focuses on the interactions between the components of the system rather then the functionality of each individual component

HW / SW Co-Verification

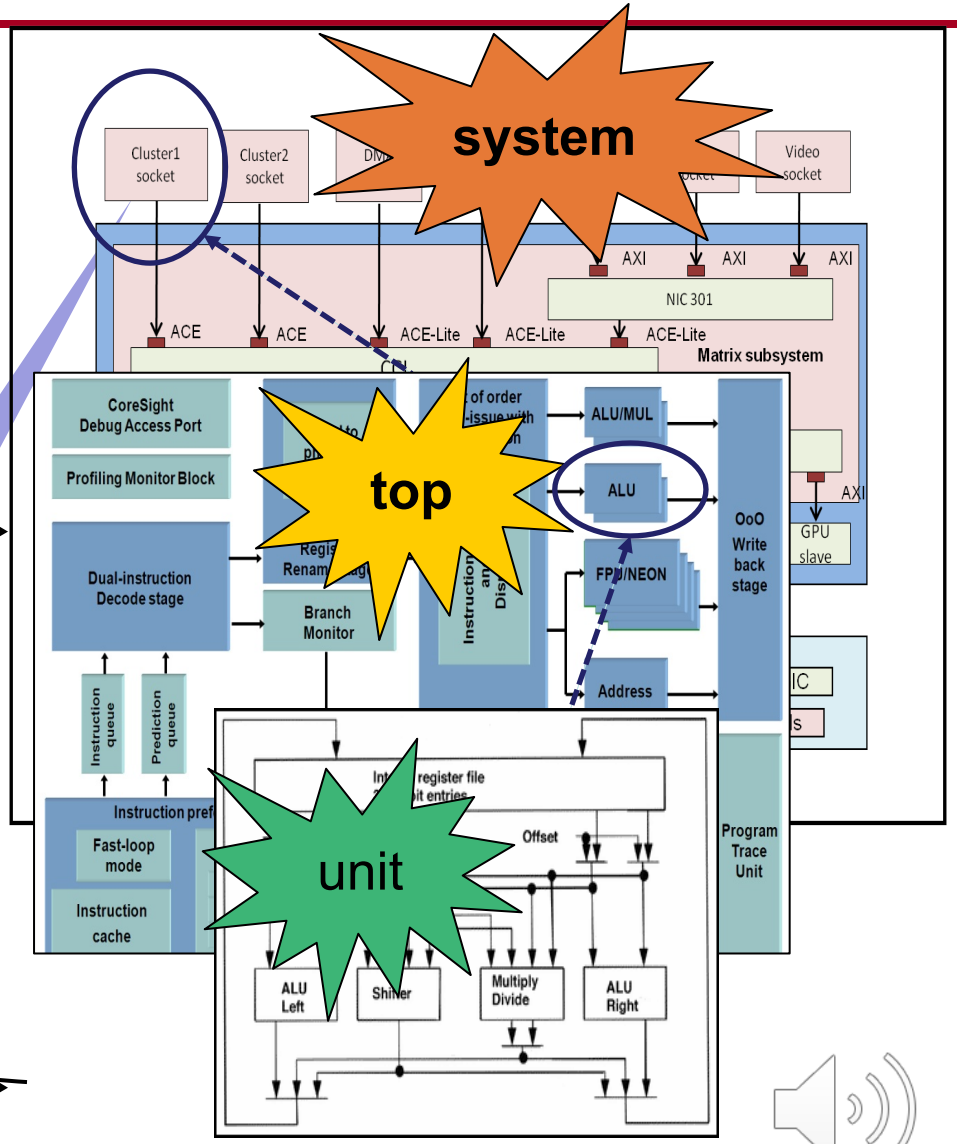
- Marries the system-level hardware with the code that runs on it
- Combines techniques from the hardware verification and software testing domains
- This combination creates many issues
  - Different verification / testing techniques
  - Different modes of operation
  - Different speed
- Beyond the scope of this course

Verification at different Design Levels

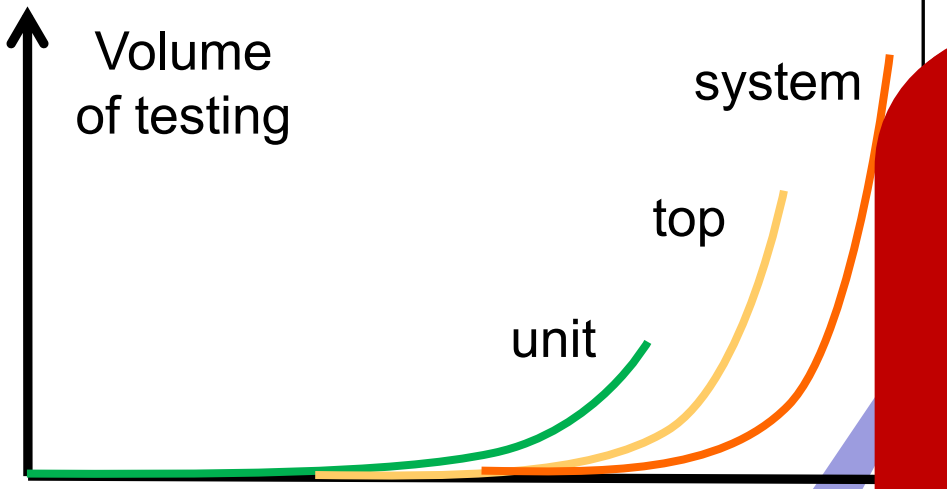


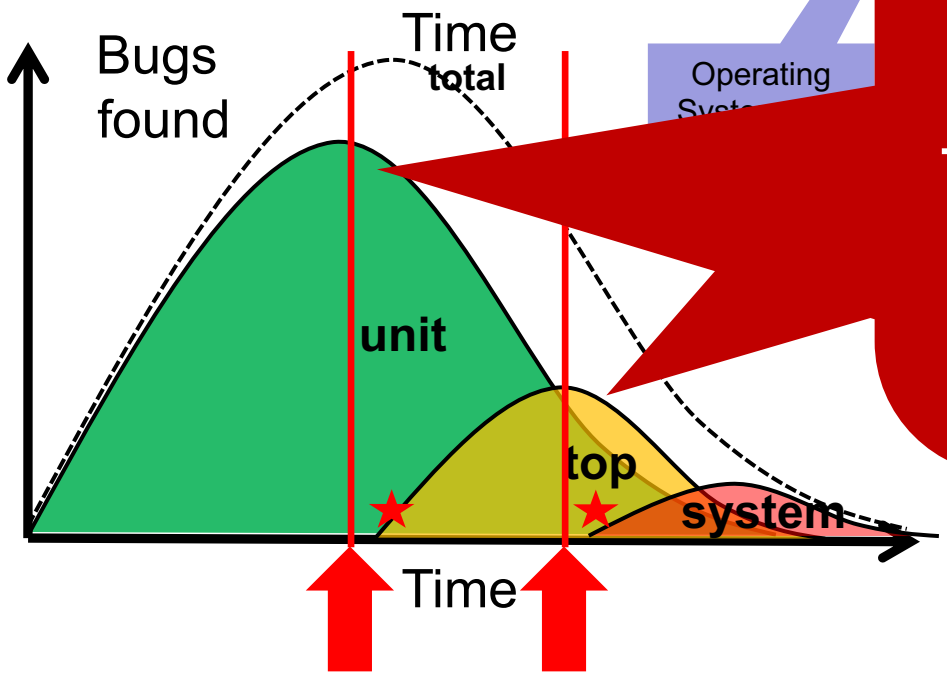


Operating System and **Application S/W**



Verification at different Design Levels





Main lesson:
Only move up to the next level when the number of bugs found at the current, lower, level has started to drop.

Which Level To Choose?

- **Always choose the lowest level that completely contains the target function under verification**
- Each verifiable piece should have its own specification
- Function to be verified may dictate the appropriate level for verification
- The selected level must provide suitable controlability and observability to perform verification

Which Levels to Verify?

- In general, each level that is exposed to the “outside world” is mandatory
  - For example, chip level, system level
- The rest depends on many factors
  - Complexity
  - Risk
  - Schedule
  - Resources

Summary

We have investigated

- Observability and Controllability
  - Black box, white box and grey box testing
- Verification hierarchy
  - Levels of verification
  - Importance of selecting the appropriate level for verification

Always choose the lowest level that completely contains the target function under verification.