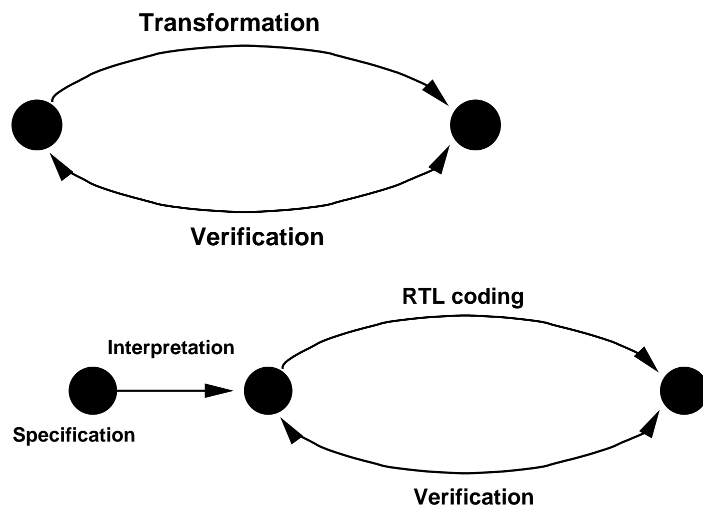
Simulation, Verification and Testing: 0

Verification :

- The process of ensuring that your VHDL model, or design, behaves in a way that matches the initial specification and requirements, ie. proving the functional correctness of the design.

Reconvergence Model:

- The representation of the verification process. What are you verifying?



A transformation from a starting point and a return path for verification, idealized model, common starting point.

Introduction of the Human Factor, common point

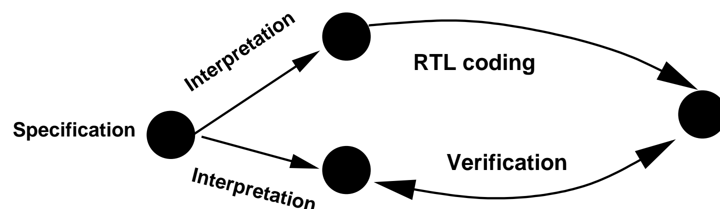
is now the interpretation of the specification.

Sources: Bergeron, J. Writing Testbenches.Kluwer Academic Publishers; Zwolinski, MDigital Design with VHDL.Prentice Hall.

Simulation, Verification and Testing: 1

Solutions?

- Automation:
  - The ideal way to eliminate human introduced errors. Hardware design usually requires human intervention.
- Poka-Yoke:
  - "inadvertent mistake prevention", A technique from TQM to reduce a process to a set of small and foolproof steps. It requires a well defined process.
- Redundancy:
  - Simple but costly, every transformation is verified by an independent team or route. Essential for high-reliability or high-cost projects such as aircraft or ASICS.

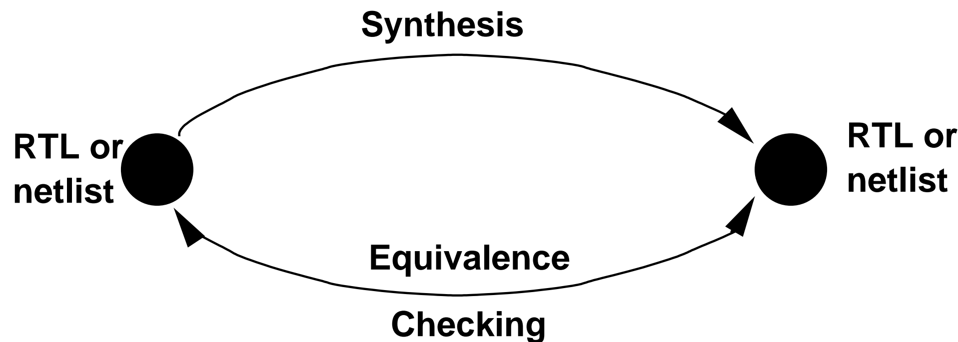


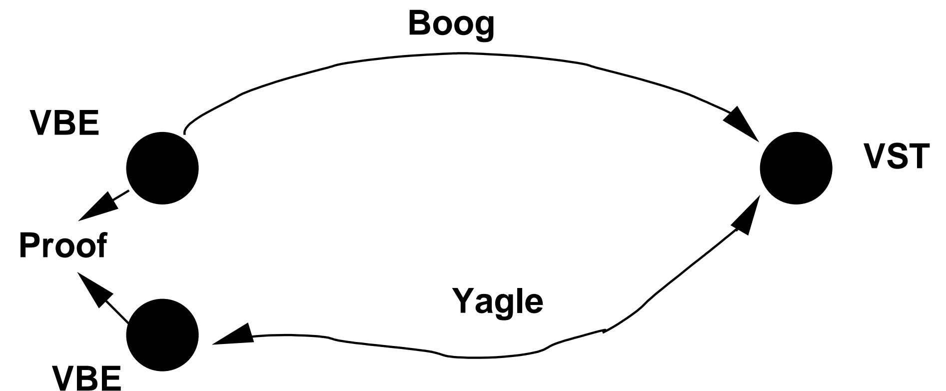
Use a different engineer for the verification especially if the specification is ambiguous.

Formal Verification: 2

How are you verifying?

- Formal verification:
  - Equivalence checking: A formal process that mathematically proves that the origin and the output are logically equivalent. The Alliance software provides tools for equivalence checking

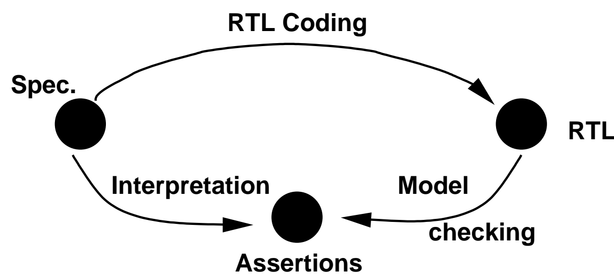




Model Checking : 3

- Model Checking:

■ The formal proving of assertions or characteristics of a design. For example:
given that mem_req is asserted then m_ack must be asserted eventually.



Problem is :
deciding which assertions to formally prove,
describing them in formal language,
learning to use the tools, PVS > 6 months

There is still a lot of work to be done in this field, especially wrt automating the identification of assertions to be proven.

Model Checking can also apply to checking state machines for unreachable or isolated states.

Functional Verification :

Does the design implement the intended functionality?

Without functional verification you have to trust that the implementation is correct.

With functional verification you can

- prove that the implementation does NOT meet the specification.
- SHOW that the implementation appears to meet the specification.
- NOT prove that the implementation meets the specification.

Even assuming a formal specification language you cannot prove the absence of design errors. You cannot prove the absence of flying pigs!

Verification :

Black-Box Verification:

- Independent of the implementation details.
- Cannot look at or know about design internals.
- Difficult to control and observe specific features.

White-Box Verification:

- Intimate knowledge and control of design internals.
- Implementation (target architecture) specific.
- Needs reworking when implementation changes.

Grey-Box Verification:

- Is a compromise between the above.
- Black-box test-case written with knowledge of internals
- Tests implementation specific features via top-level interfaces
- Will need rewriting for different implementations.

Testing is NOT verification!

Testing is to verify that the design was manufactured correctly.

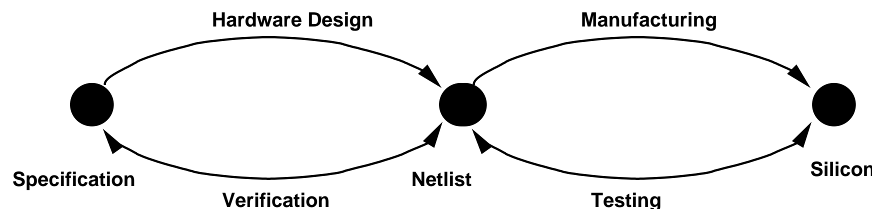
Do all locations in the design switch from ’1’ to ’0’ and from ’0’ to ’1’?

Test Coverage:

- the ratio of physical locations tested to the total number of locations.

Problem:

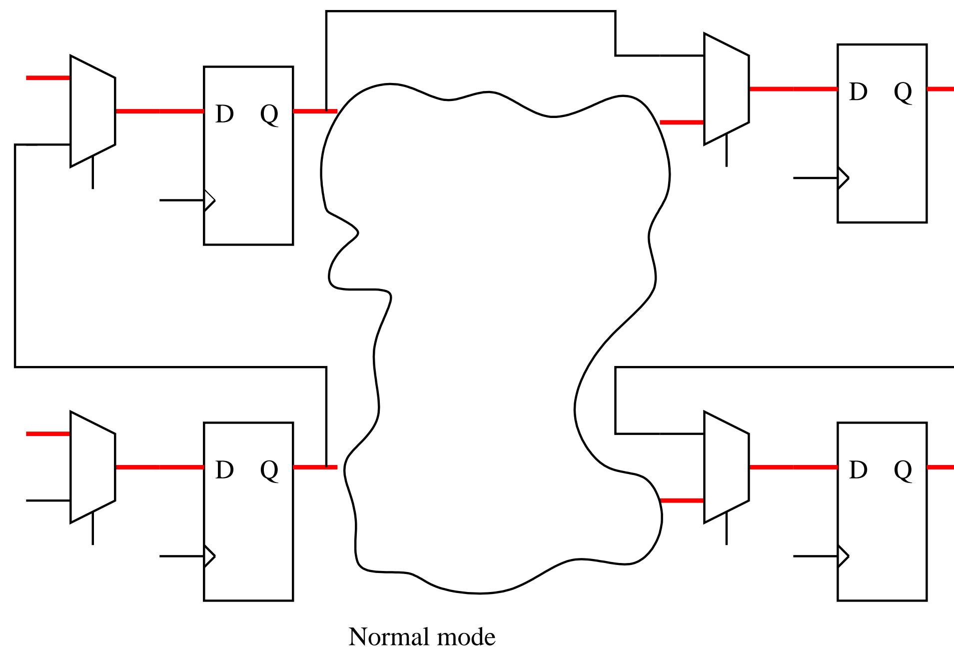
- Can we set all internal physical locations to ’0’ or ’1’? A wrist-watch has few inputs and outputs but many billions of states if it has a calendar function (milliseconds in 100 years?)



Testing techniques:

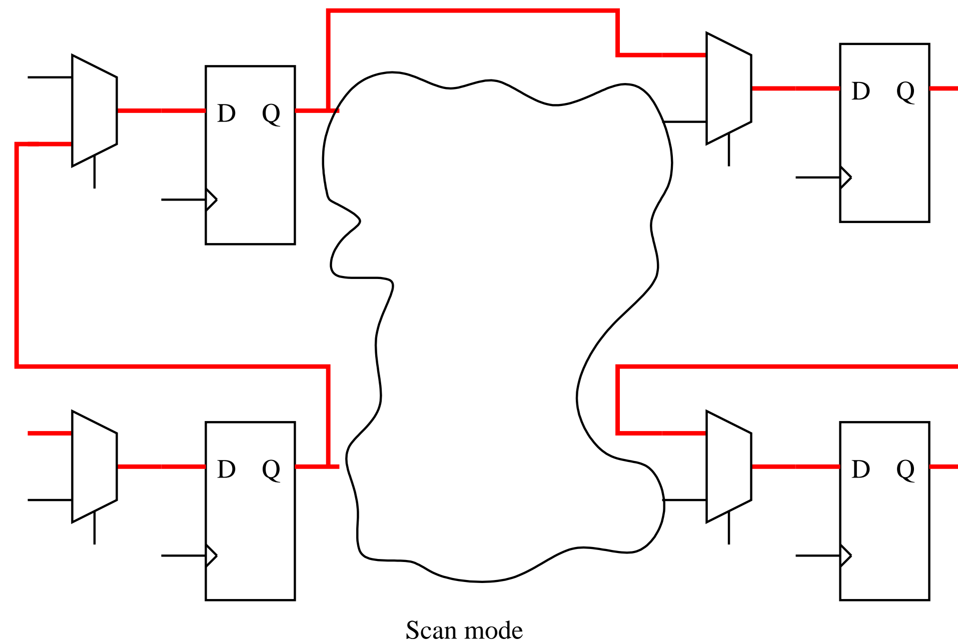
Scan-Based testing:

- All registers are linked in a long serial chain.
- 2 operating modes.
  - normal-mode when the registers operate as normal



Normal mode

- scan-mode when the registers act as a long shift register



Scan mode

Usage:

- set registers into scan-mode
- clock in a known bit-string, length = No. of reg.
- set registers to normal mode
- apply a clock pulse
- set registers to scan-mode
- clock out the resulting string and inspect.

Problems:

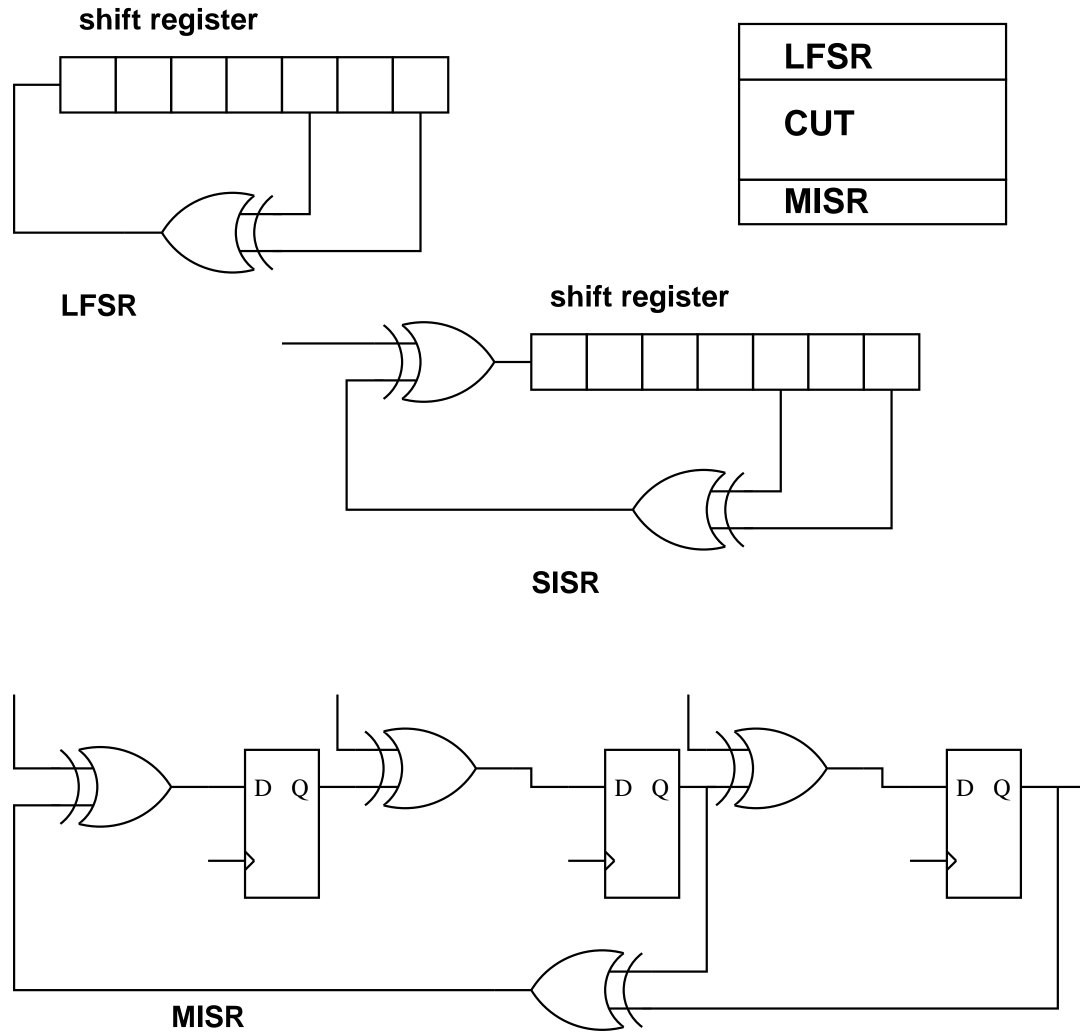
- Need to modify your design to incorporate modified registers and extra inputs.
- Therefore you need to reverify your design.
- Must be a fully synchronous design
- No derived or gated clocks
- Use only one clock edge.
- increased area of chip (power, heating, packaging)

Testing, other techniques :

BIST (BIT)

- Built In Self Test or Built In Test.
  - Specific hardware is included in the IC or circuit board to provide for, and perform, some of the testing.
  - Store test vectors and responses in look-up tables for example.
  - Alternatively, use Linear Feedback Shift Registers (LFSR) to generate the test vectors and a Single or Multiple Input Signature Register (S/MISR) to test the result.
  - Principle is to feed a set of test vectors into the Circuit Under Test (CUT) and feed the output vectors into the MISR to generate a go/no-go signature.

Testing: BIST



Problems:

- Sequence has to be designed when circuit is built.
- Additional, specialised hardware.
- Faults in the LFSR or MISR
- false positives in the MISR, although this tends to 2^-n.

Solution?

- use BILBO!
- Built In Logic Block Observation.
- Uses registers in a similar way to scan-path testing.
- Requires registers with 3 control lines. requires approx 4 extra gates per register.
- extra hardware = extra faults + extra cost + increased propagation delay.
- So do we need to include a tester tester?
- And what about CPLDs or FPGAs, how do we test those?

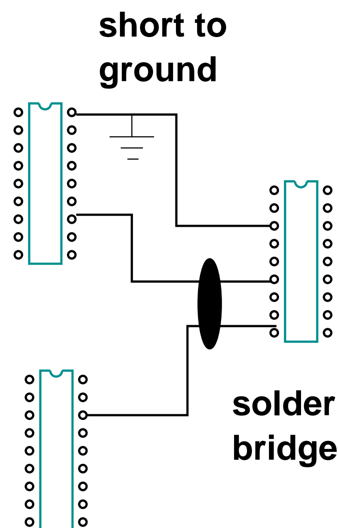
Testing: Boundary scan

Testing the PCB :

- Used to use a 'bed of nails' approach... but
  - may have a 20 layer board so unreachable layers
  - package density may cause access problems
  - Multi-chip modules using unpackaged ICs

Boundary Scan:

- Tests the interconnect on PCBs and mounted ICs.
- Allows the outputs of the ICs to be controlled and the inputs to be observed.



ICs need 2 special pins, TDI and TDO, test data in & out, plus control pins collectively known as the TAP (Test Access Port)