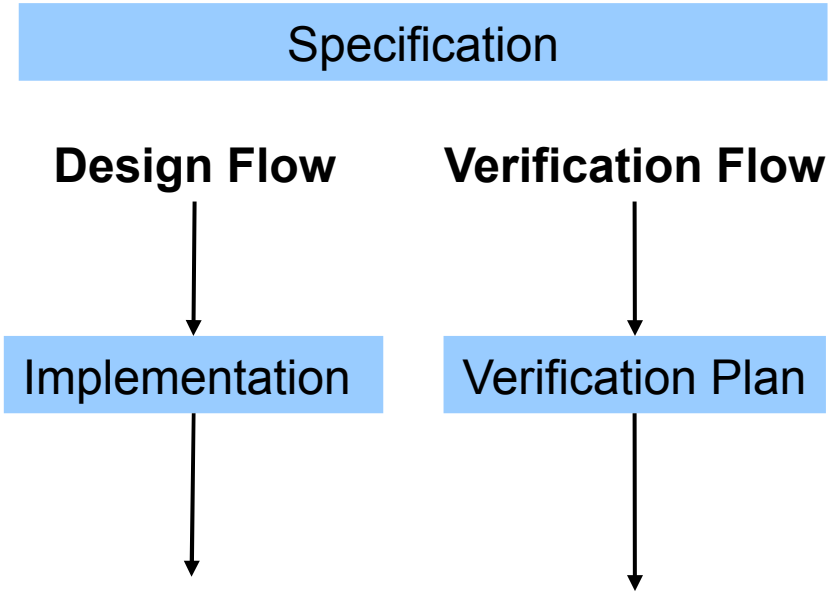
Verification Plan

- Contrast with ad hoc verification:
  - Let each design engineer check their own work
  - As and when time allows
  - Put it all together and hope it works!
  - Implement in FPGAs, so that bugs can be fixed
- For complex systems this is not good enough!
  - Need a plan
    - What is to be verified?
    - When is it done?
    - What resources are needed?
    - How is the plan kept up to date?
    - When have we done enough verification?

Specification



- The verification plan must be started once the functional requirements are known, but before detailed work begins on the implementation.
- This can be repeated at all levels of the design hierarchy.
- The design specification defines the correct behaviour of the system
- Similarly, the verification specification defines the correct behaviour of the verification environment
  - Verification models and software can contain bugs - just the same as the actual system!

Planning the Verification Plan

- What are the goals?
  - What level of confidence is needed?
  - When is verification complete?
- What features need to be verified?
  - Needs input from the designers – may find bugs in the specification
- Plan the verification environment
  - Bottom-up for modules; top-down for system
  - Add reused IP to the plan
- Create the test list
- Specify coverage
  - Metrics
- Determine resources
  - People; software licences; time
- Plan the schedule

What Features to Verify?

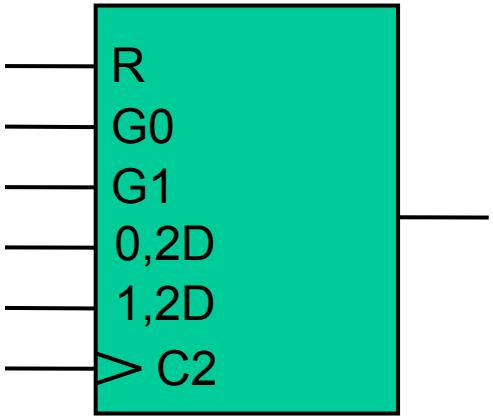
- Need to decide what to verify:
  - Full functionality versus interfaces?
  - Entire design versus sub-blocks?
  - Timing?
    - Static timing verification?
  - Performance?
  - Error rate?
  - Exhaustive testing?
    - If not, then what?
  - Software?
  - Analog blocks and external IP blocks?

Verification Strategy

Block-level example

```
always @(posedge clk)
  if (reset) Q <= 0;
  else
  begin
    if (selA) Q <= QA;
    if (selB) Q <= QB;
  end
```

RTL Verilog



- Latched multiplexer
  - Simple enough to understand in a few minutes
  - Does it meet the specification?
    - Do we care about the implementation?
    - Black-box testing
  - Do we want to debug it?
    - White-box testing
  - Is it reused IP?
    - Can we trust it?
  - How do we verify it in a larger system?

Analysis to Choose Tests

- What is the initial state?
- What is the nature of the input data?
- Does the system have feedback?
- What range of values is allowed in the design?

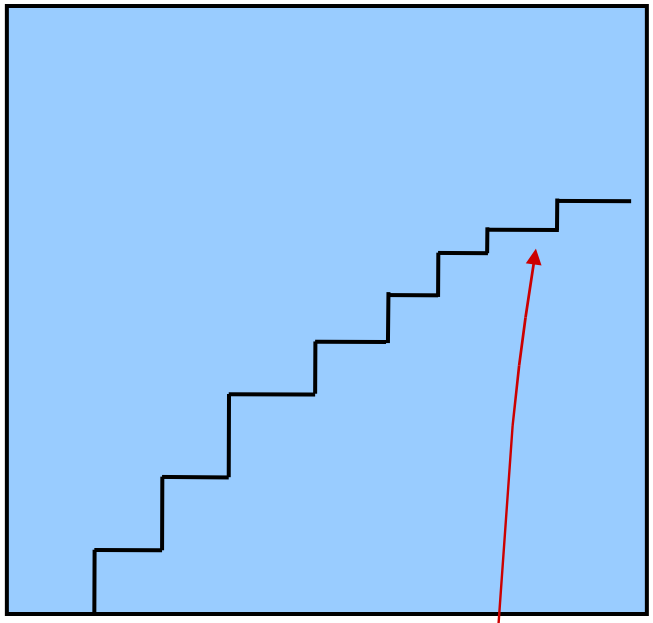
Verification Metrics

- Are we there yet? Code coverage and bug detection rate

```
84:   if A < MAX then
84:     A := A + 1;
      else
 0:     B := B + 1;
      end if;
84:   if A > B then
12:     C := C + B;
      else
72:     C := C - A;
      end if;
```

How to execute this line?

Number of executions



Will this graph reach 100%?

Coverage Metrics Interpretation

- Full code coverage
  - Easy to achieve for statement coverage
  - Difficult for path and expression coverage
- Usefulness of code coverage tools
  - Can help identify hard to find corner cases
  - Low coverage numbers indicate need for more test suites
  - 100% coverage says something about thoroughness of verification, not about correctness of code

Regression Testing

- Bugs have a habit of re-appearing during the development cycle
- A regression test is a large body of tests to ensure that the design still works
- Test whether the DUT has *regressed* after making a change
  - IMPORTANT: add new tests as each bug is found and fixed
- Regression tests must be runnable in batch mode (non-interactive)
- Regression tests must be runnable semi-automatically!
- Regression tests must be self-checking - go/no go test

Bug Metrics

- Track bug reports
  - Which section of code?
  - Which test discovered the bug?
  - Is the bug rate higher than expected?
    - Might indicate a local problem.
- Track bugs found by the verification code
  - An early test may reveal more bugs than a later test
  - A lack of bugs found may suggest a problem with the verification code!
- Track everything
  - Or may end up debugging the same code several times