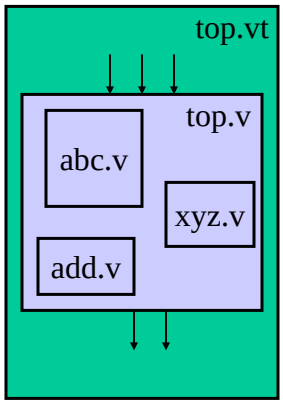
TESTING

Testing

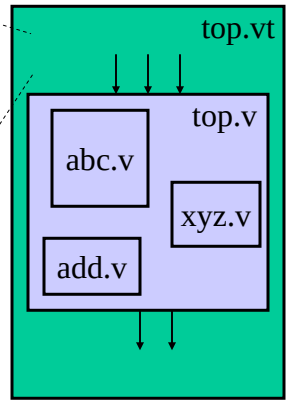
- Logic blocks (*.v)
  - Will be synthesized
  - No "#" delays except for registers
- Testing blocks (*.vt)
  - Pretty much anything goes
  - You'll need "#" delay statements
- Instantiate logic blocks inside testing blocks and drive the inputs and check outputs there



Example test module

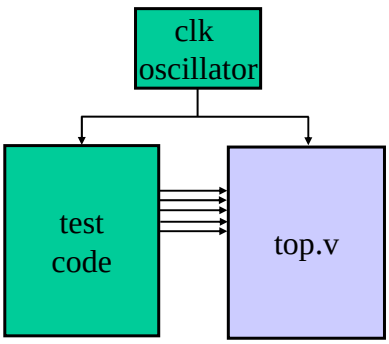
- Basic flow Example 1 (*.vt)

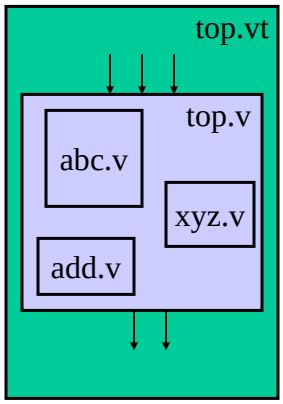
```
in       = 4'b0000;
clk      = 0;
reset    = 1;
#100; clk=1; #100; clk=0;

reset    = 0;
#100; clk=1; #100; clk=0;

in       = 4'b0001;
#100; clk=1; #100; clk=0;

in       = 4'b1010;
#100; clk=1; #100; clk=0;

...
```



Example test module

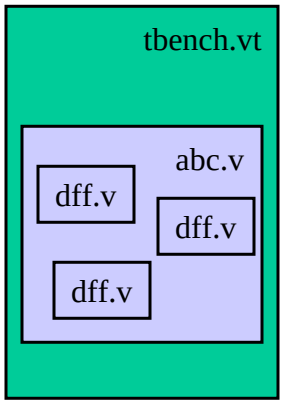
- Basic flow Example 2





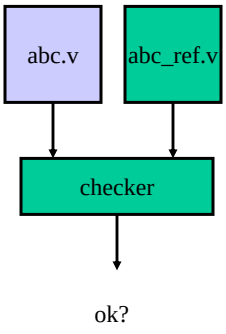
Example Given To You

- Example test bench and modules on web page under "Notes on running verilog"
- Not the most realistic partitioning (e.g., we would normally never put D FFs in their own modules), but it's a good working example to get you started



Verification

- A number of ways to verify designs
  1) Eyeball text printouts
     - Quickest and easiest
  2) Eyeball waveforms
     - Quick and easy
  3) Write reference code and some other code to see if the two are the same. Make sure you temporarily force an error to test your setup.
     - This is the most robust and is what is required for non-trivial designs



Verifying Correctness

- As designs become more complex, verifying it is correct becomes more difficult
- "Golden reference" approach
  - Write an easy to understand simple model in a higher-level language
    - C or matlab commonly
      - Matlab is a natural choice for DSP applications
    - Must be written in *a different way* from the verilog implementation to avoid repeating the same bugs
  - Designers agree this is the correct function (imagine a detailed design review)
  - Many high-level tests run on golden reference
    - Model should be fast

Verifying Correctness

- "Golden reference" approach (continued)
  - Two major approaches to comparing with the Golden Reference
    1) Hardware and Reference must be "close"
       - Possibly compare SNR of hardware vs. reference
       - Inadequate for control hardware which must be a perfect match
    2) Hardware and Reference must be "Bit-accurate"
       - Run a smaller number of tests but now hardware must match golden reference exactly, bit for bit
       - Far easier to automate testing
       - Matlab must now do awkward operations such as rounding and saturation that match hardware
         - » For example, `floor(in + 0.5)` for rounding