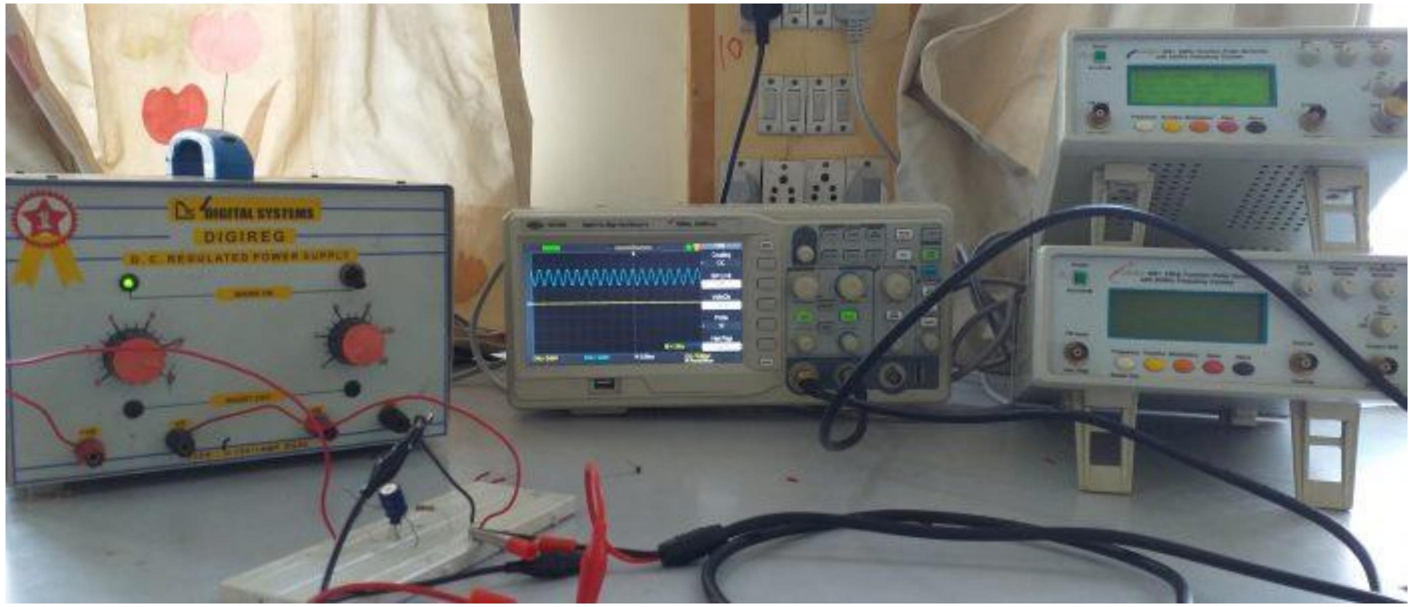
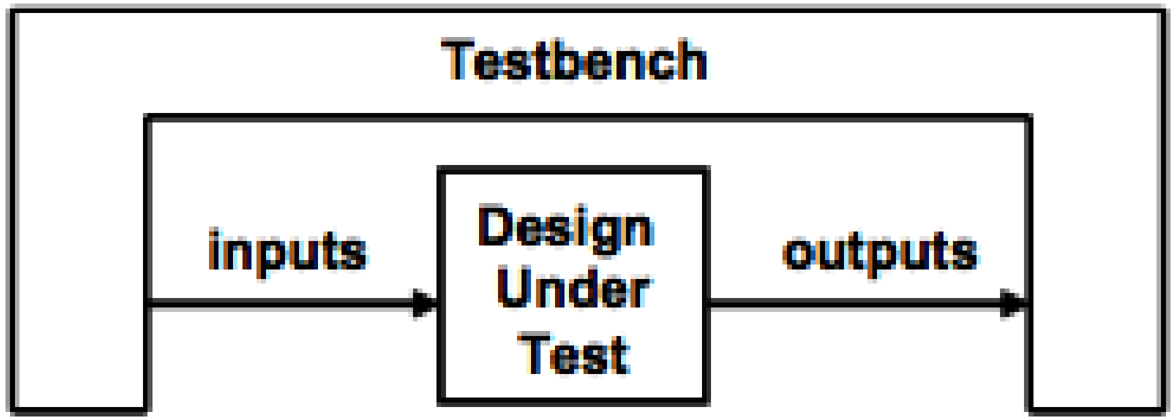
TestBench Components





Design and Testbench

- “Design” ….HDL code….converted as RTL……goes into IC or SoC
  - Design Engineer
- “Test Bench” or “TB” ….HDL (or any other) code…used to verify design
  - Verification Engineer
- For Design Engineer….Verilog and SystemVerilog is same
- SystemVerilog is preferred by Verification Engineer
- SystemVerilog = Verilog + *A lot of verification supporting features.*
- The design code might contain hierarchical level of modules.
- There will be always a top level module in the design which instantiate the hierarchical modules.
- The Test Bench is again another ‘module’ in SV.
- The ‘Top-level module’ of the design will be instantiated in the TB module.

Design and Testbench

```
module a();
endmodule: a

module b();
endmodule: b

module c();
endmodule: c

module d();
endmodule: d

module cluster();
            c   c1(.*)
            d   d1(.*)
endmodule: cluster
```

```
module soc_top();
            cluster cluster1(.*)
            a     a1(.*)
            b     b1(.*)
endmodule: soc_top
```

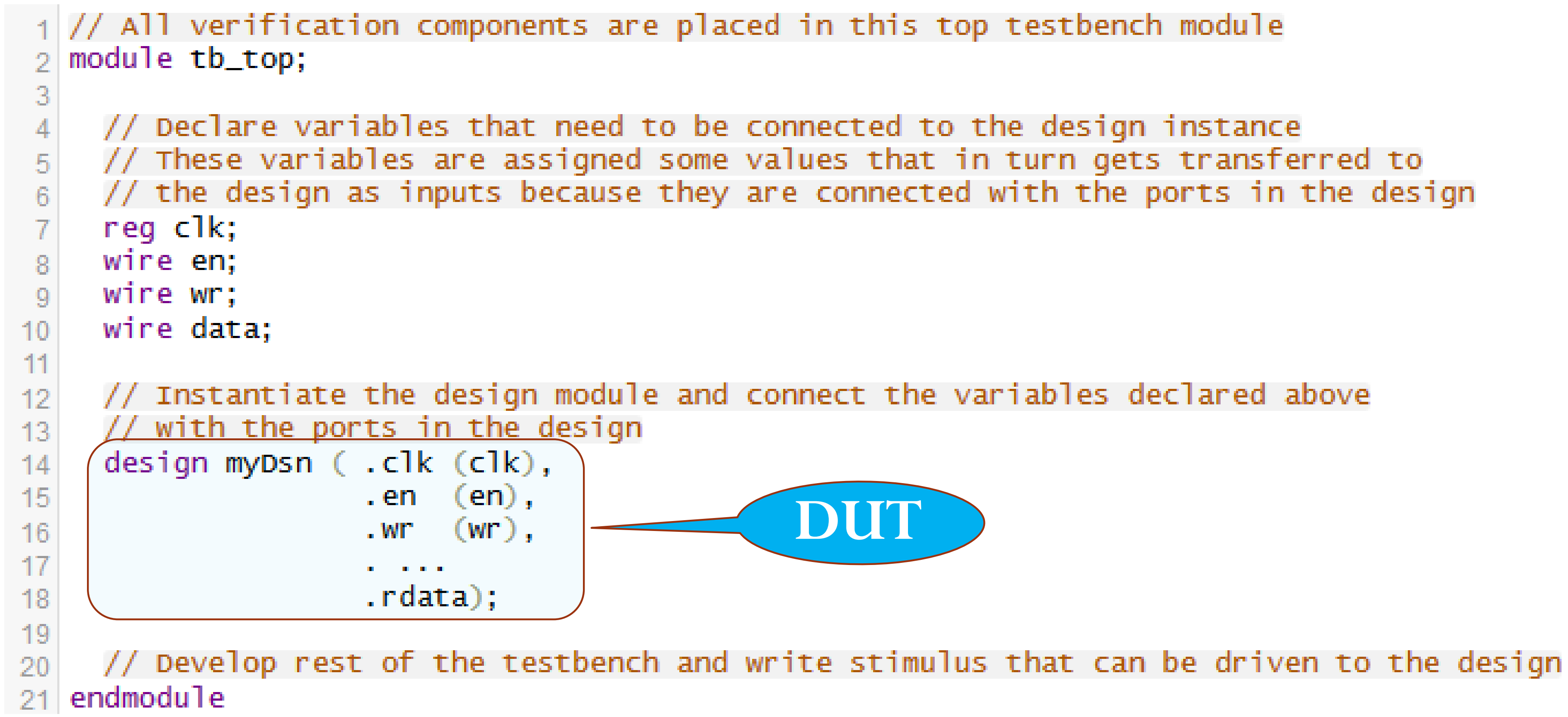
```
module test_bench();
            soc_top soc_top1(.*);

            //TB Functionality
endmodule: test_bench
```

OR

```
module test_bench();
            //TB Functionality
endmodule: test_bench
```

```
module top();
            soc_top soc_top1(.*);
            test_bench test_bench1(.*);
endmodule: test_bench
```

What is DUT?



```
// All verification components are placed in this top testbench module
module tb_top;

  // Declare variables that need to be connected to the design instance
  // These variables are assigned some values that in turn gets transferred to
  // the design as inputs because they are connected with the ports in the design
  reg clk;
  wire en;
  wire wr;
  wire data;

  // Instantiate the design module and connect the variables declared above
  // with the ports in the design
  design myDsn ( .clk (clk),
                 .en  (en),
                 .wr  (wr),
                 . ...
                 .rdata);

  // Develop rest of the testbench and write stimulus that can be driven to the design
endmodule
```

Linear Testbench

- ✓ Linear TestBench is the simplest, fastest and easiest way of writing testbenchs.
- ✓ This became novice verification engineer choice.
- ✓ It is also slowest way to execute stimulus.
- ✓ Typically, linear testbenchs are written in the VHDL or Verilog. In this TestBench, simple linear sequence of test vectors is mentioned.
- ✓ Stimulus code is easy to generate.
- ✓ Small models like simple state machine or adder can be verified with this approach.

SystemVerilog Testbench Features

- ✓ Constrained random generation – built on class infrastructure
- ✓ Universal ***randomize***() method
- ✓ In-line random generation (***randomize***() with)
- ✓ Spawn threads (*fork….join/join_any/join_none*)
- ✓ Control threads (*process* class methods)
- ✓ Inter-process communication (*mailboxes, semaphores* etc)