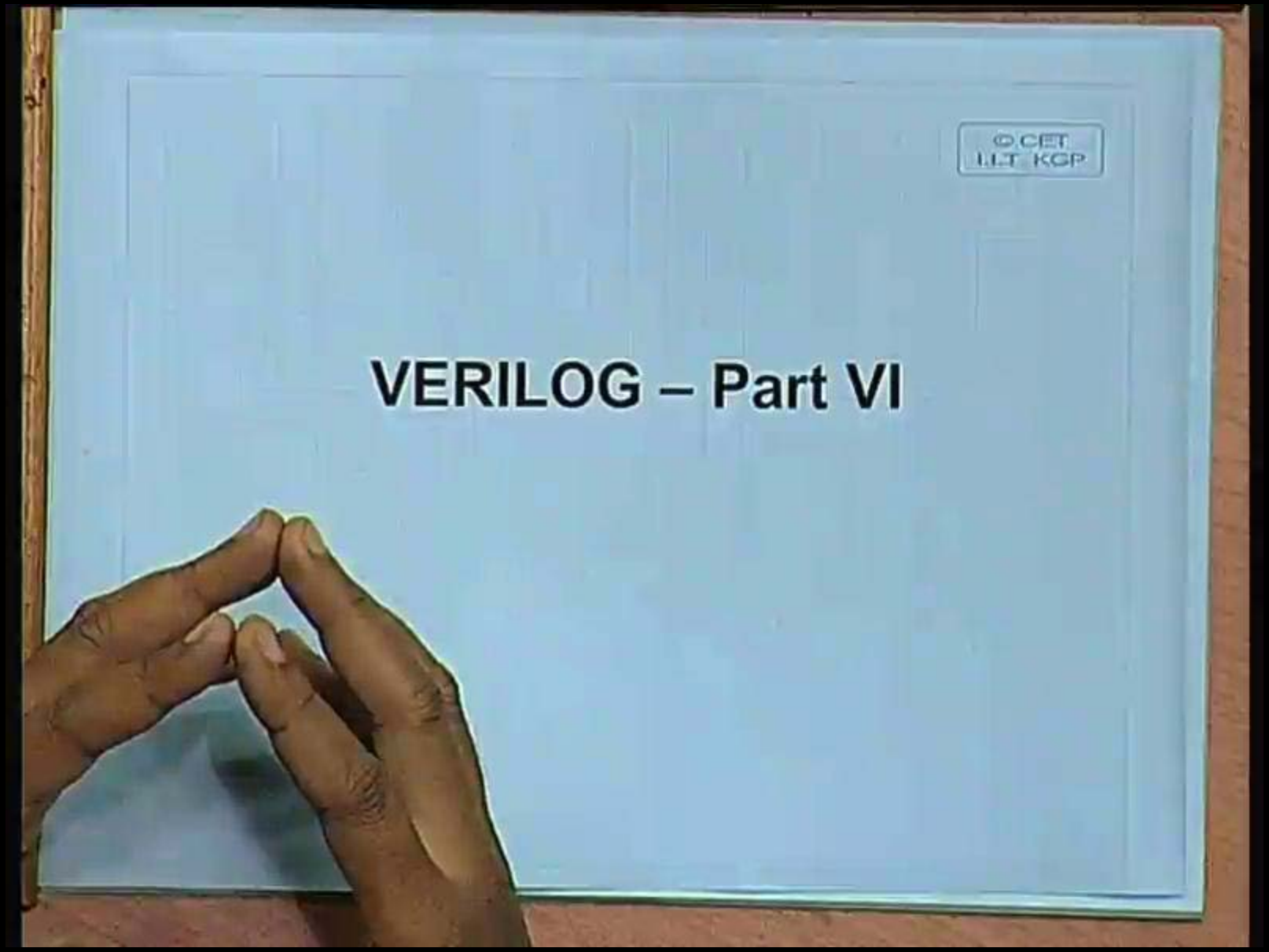
CSE-308: Digital System Design

Lecture 6


©CET
I.I.T. KGP
VERILOG – Part VI

Memory Modeling Revisited

- Memory is typically included by instantiating a pre-designed module.
- Alternatively, we can model memories using two-dimensional arrays.
  - Array of register variables.
    - Behavioral model of memory
  - Mostly used for simulation purposes.
  - For small memories, even for synthesis.

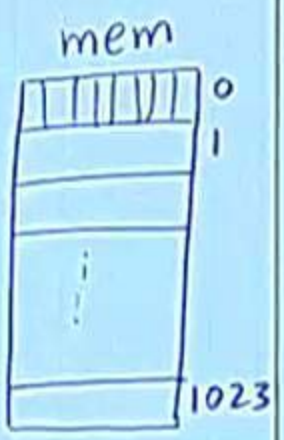
Typical Example


```
module  memory_model ( ..... )

reg [7:0]   mem [0:1023];

endmodule
```

mem[i]



How to Initialize memory

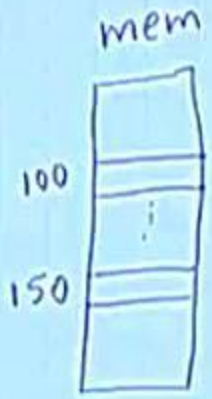
- By reading memory data patterns from a specified disk file.
  - Used for simulation.
  - Used in test benches.
- Two Verilog functions are available:

1. $readmemb (filename, memname, [startaddr, stopaddr])

   Data read in binary format.

2. $readmemh (filename, memname, startaddr, stopaddr)

   Data read in hexadecimal format.



* File data are insufficient.

* Read
  start = 100
  stop = 150

An Example


```
module  memory_model ( …….. )
    reg [7:0]   mem [0:1023];
    initial
      begin
        $readmemh (“mem.dat”, mem);
      end
endmodule
```

A Specific Example :: Single Port RAM with Synchronous Read-Write

```
module  ram_1 (addr, data, clk, rd, wr, cs)
        input  [9:0] addr;    input  clk, rd, wr, cs;
        inout  [7:0] data;
        reg [7:0]  mem [1023:0];    reg [7:0] d_out;

        assign   data = (cs && rd)  ?  d_out ; 8'bz;
        always  @ (posedge clk)
                if (cs && wr && !rd)  mem [addr]  =  data;
        always  @ (posedge clk)
                if (cs && rd && !wr)  d_out  =  mem [addr];
endmodule
```

A Specific Example :: Single Port RAM with Asynchronous Read-Write

```
module  ram_2 (addr, data, rd, wr, cs)
        input  [9:0] addr;    input  rd, wr, cs;
        inout  [7:0] data;
        reg [7:0]  mem [1023..0];    reg [7:0] d_out;

        assign   data = (cs && rd)  ?  d_out ; 8'bz;
        always  @ (addr or data or rd or wr or cs)
                if (cs && wr && !rd)  mem [addr]  =  data;
        always  @ (addr or rd or wr or cs)
                if (cs && rd && !wr)  d_out  =  mem [addr];
endmodule
```

A Specific Example :: ROM/EPROM

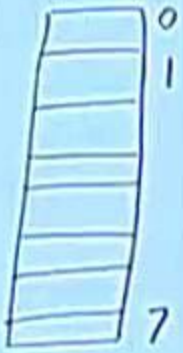
```
module  rom (addr, data, rd_en, cs)
        input  [2:0] addr;   input  rd_en, cs;
        output  [7:0] data;
        reg [7:0]  data;
        always  @ (addr or rd_en or cs)
                case (addr)
                   0:  22;
                   1:  45;
                   .....
                   7:  12;
                endcase
endmodule
```

(rd_en and cs)

data = 22

data = 45

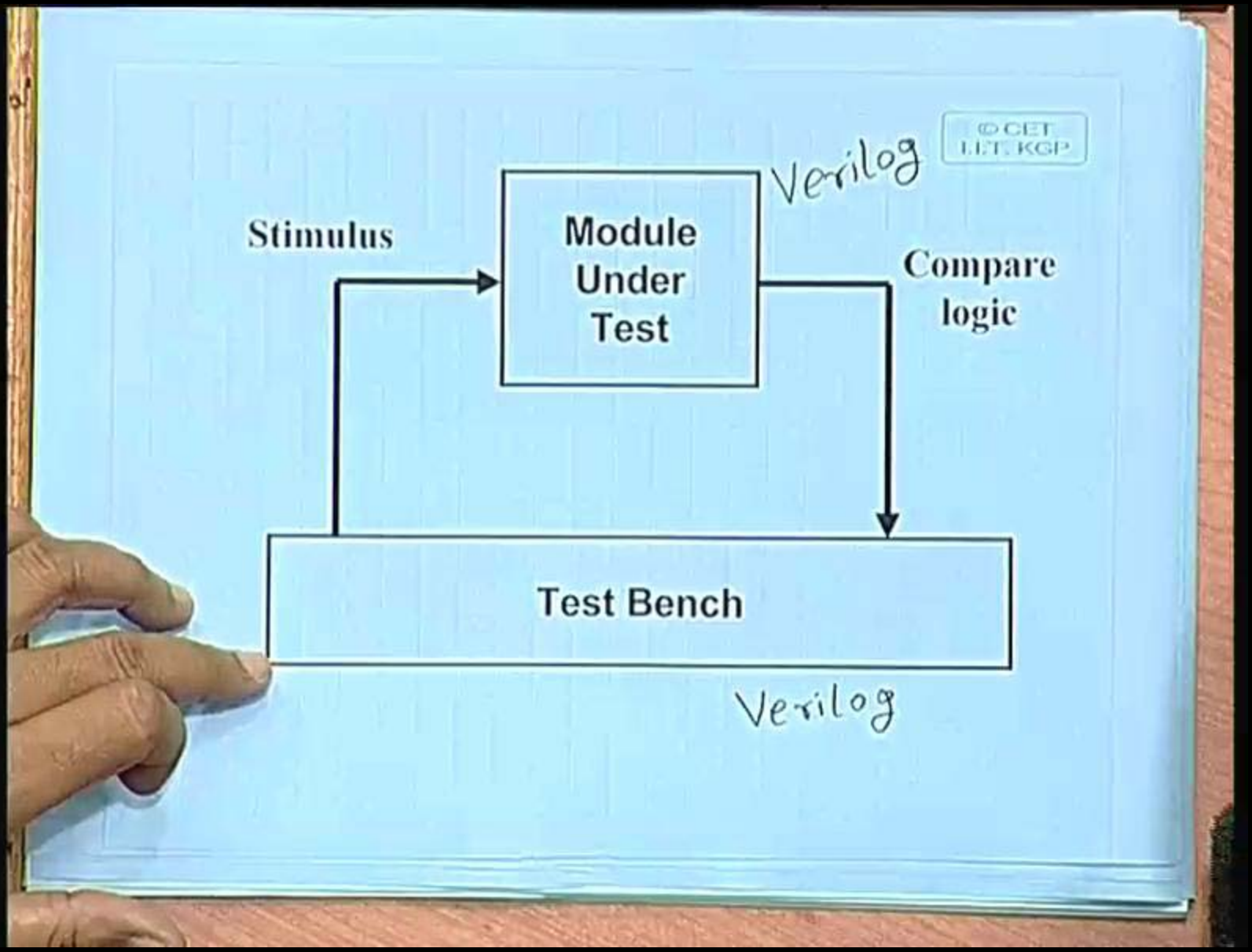
data = 12

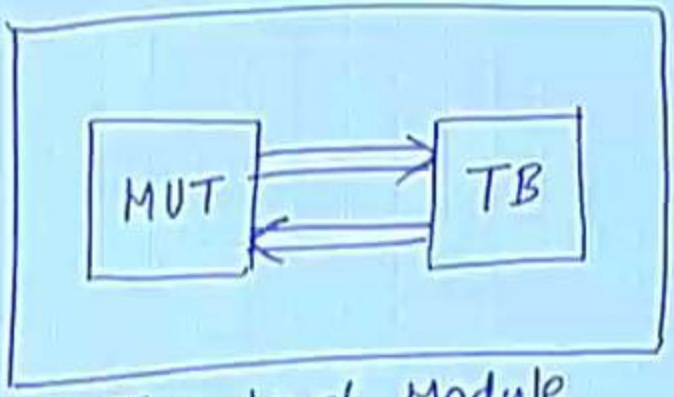


Verilog Test Bench

- What is test bench?
  - A Verilog procedural block which executes only once.
  - Used for simulation.
  - Testbench generates clock, reset, and the required test vectors.

Proc. block — { always, initial


© CET
I.I.T. KGP
Verilog
Stimulus
Module
Under
Test
Compare
logic
Test Bench
Verilog



Top-level Module

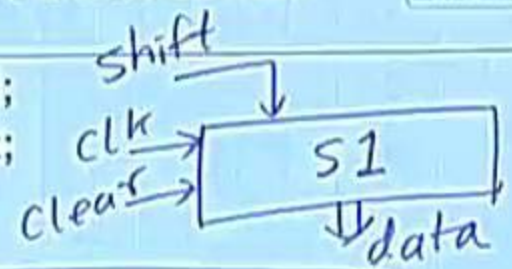
How to Write Testbench?

- Create a dummy template
  - Declare inputs to the module-under-test (MUT) as "reg", and the outputs as "wire"
  - Instantiate the MUT.

- Initialization
  - Assign some known values to the MUT inputs.
- Clock generation logic
  - Various ways to do so.
- May include several simulator directives
  - Like $display, $monitor, $dumpfile, $dumpvars, $finish.

- $display
  - Prints text or variables to stdout.
  - Syntax same as "printf".
- $monitor
  - Similar to $display, but prints the value whenever the value of some variable in the given list changes.
- $finish
  - Terminates the simulation process.
- $dumpfile
  - Specify the file that will be used for storing the waveform.
- $dumpvars
  - Starts dumping all the signals to the specified file

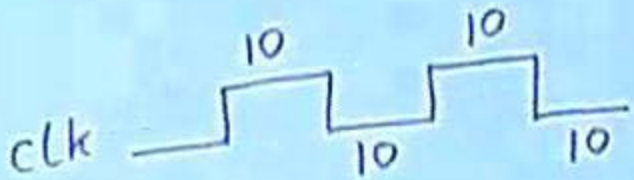
Example Testbench



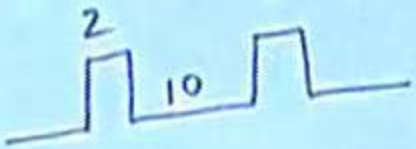

```
module shifter_toplevel;
        reg clk, clear, shift;
        wire [7:0] data;

    shift_register S1 (clk, clear, shift, data);
    initial
      begin
            clk = 0;  clear = 0;  shift = 0;
      end
    always
            #10 clk = !clk;
endmodule
```

// initialization




```
always
#10 clk = !clk;
```




```
always
  begin
    #2  clk = 0;
    #10 clk = 1;
  end
```

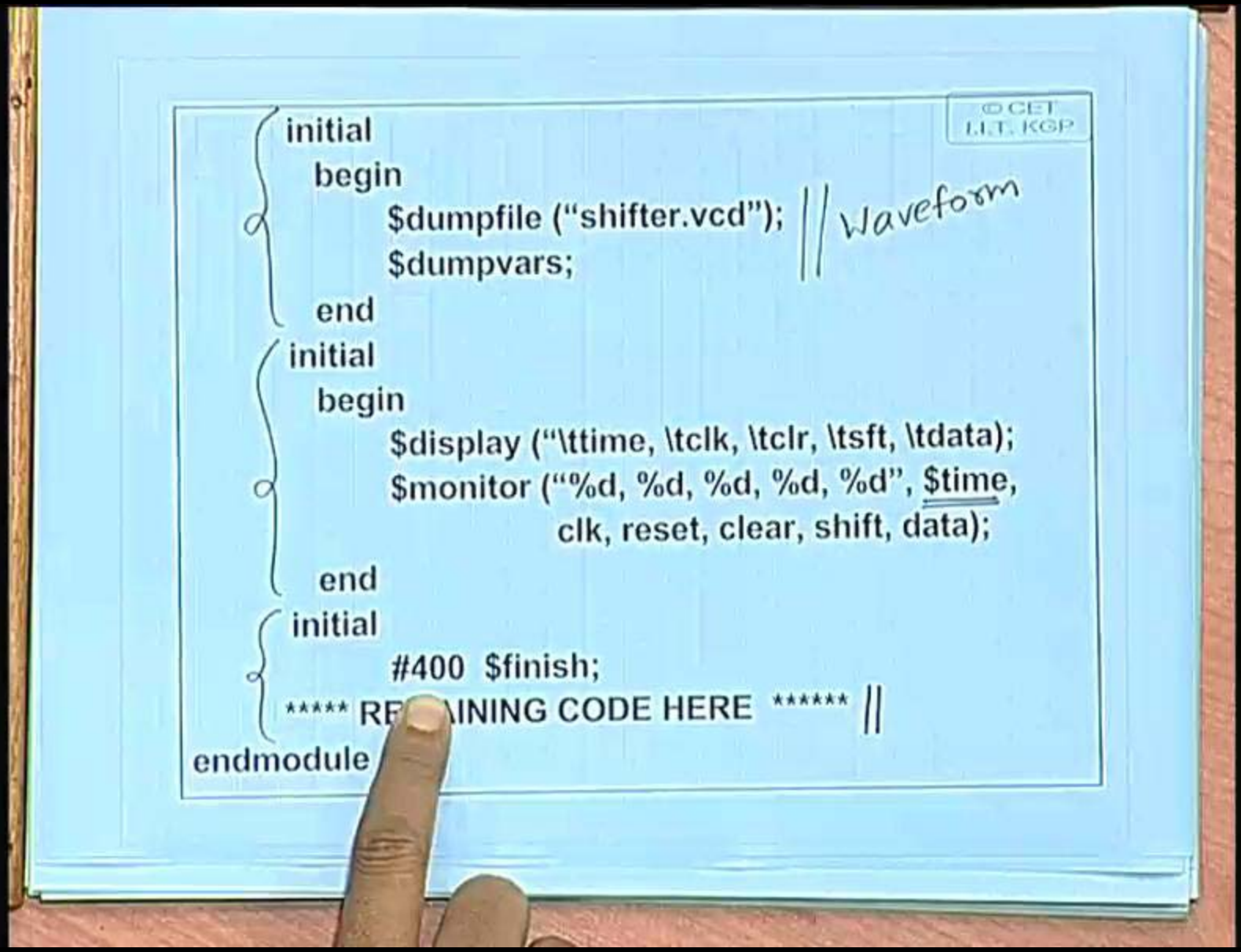
`timescale

A More Complete Version


```
module  shifter_toplevel;
        reg  clk, clear, shift;
        wire [7:0]  data;

        shift_register S1 (clk, clear, shift, data);
        initial
         begin
                clk = 0;   clear = 0;   shift = 0;
         end
        always
                #10 clk = !clk;
```

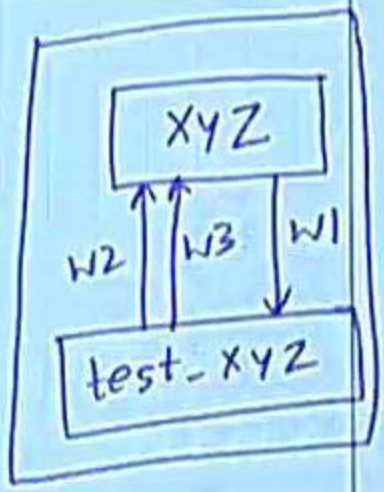
contd..

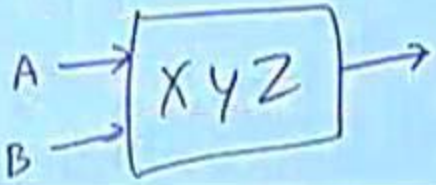

© CET
I.I.T. KGP
initial
begin
$dumpfile ("shifter.vcd");
$dumpvars;
end
initial
begin
$display ("\ttime, \tclk, \tclr, \tsft, \tdata);
$monitor ("%d, %d, %d, %d, %d", $time,
clk, reset, clear, shift, data);
end
initial
#400 $finish;
***** RE INING CODE HERE ******
endmodule
Waveform

A Complete Example


```
module testbench;
   wire w1, w2, w3;
   xyz m1 (w1, w2, w3);
   test_xyz m2 (w1, w2, w3);
endmodule

module xyz (f, A, B);
   input A, B;  output f;
   nor #1 (f, A, B);
ndmodule
```





contd..

```
module test_xyz (f, A, B);
   input f;
   output A, B;
   reg A, B;
   initial
      begin
         $monitor ($time, "A=%b", "B=%b", f=%b",
                                    A, B, f);
         #10 A = 0; B = 0;
         #10 A = 1; B = 0;
         #10 A = 1; B = 1;
         #10 $finish;
      end
endmodule
```