Testbench writing

Speaker: Lin Yi-Hsien

Graduate Institute of Electronics Engineering
National Taiwan University

Some slides come form CVSD Lecture

Outline

- Design Under Test
- Initialization part
- Clock part
- Timing Control part
- File IO part
- Input Data part
- Output Data part
- Pattern Verification

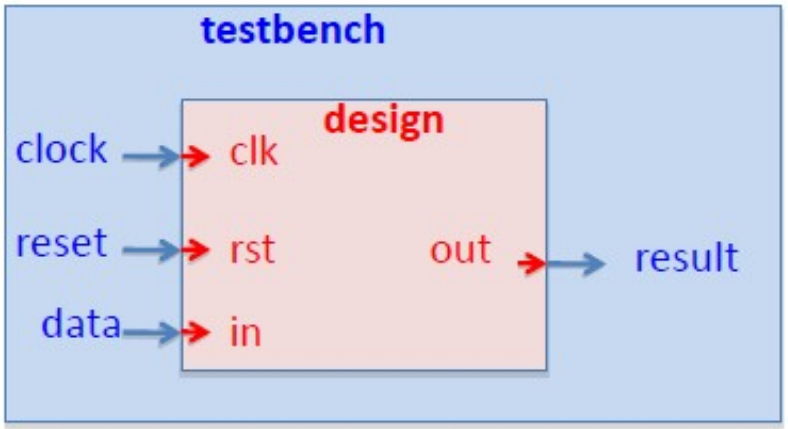
Design Under Test

- You should include the module you want to test in the testbench.

```
module testbench;

reg clock, reset, data;
wire result;

design u_design(
    .clk(clock),
    .rst(reset),
    .in(data),
    .out(result)
);
…
```



Input of DUT: use **reg** for applying stimulus
Output of DUT: use **wire** to capture signals

Initialization part

```
`timescale 1 ns/10 ps
`define CYCLE 10

module bilateral_tb;

reg clock, rst_n, enable_i;
reg [12:0] data_i;
wire [15:0] data_o;
wire enable_o;
wire end_o;

bilateral top(
    .clk(clock),
    .reset(rst_n),
    .enable_i(enable_i),
    .data_i(data_i),
    .data_o(data_o),
    .enable_o(enable_o),
    .end_o(end_o),
);
```

Timescale

- **`timescale 1 ns/10 ps**
- which is declared as `timescale time_unit base/precision base
- time_unit is the amount of time a delay of #1 represents. The time unit must be 1, 10, or 100.
- base is the time base for each unit, ranging from seconds to femtoseconds, and must be: s, ms, us, ns, ps or fs
- precision and base represent how many decimal points of precision to use relative to the time units.

Clock part

```
`timescale 1 ns/10 ps
`define CYCLE 10

[Initialization part]

initial begin
  clk = 0;
end

always #(`CYCLE/2) clk = ~clk;
```

Wave dump part

```
`timescale 1 ns/10 ps
`define CYCLE 10

[Initialization part]
[Clock part]

initial begin
   //FSDB
   $fsdbDumpfile("bilateral.fsdb");
   $fsdbDumpvars(0, top, "+mda");
   //VCD
   $dumpfile(“filename”);
   $dumpvars();
end
```

Wave dump part

- Value Change Dump (VCD) format

- Indigenously supported by most simulators

- Using ASCII text for waveform recording, extremely huge file size

- Fast Signal Database (FSDB) format

  - Defined by *SpringSoft Verdi debugging system*

  - More compact format, small file size

```
//for memory array dump
$fsdbDumpvars(0, top, "+mda");
```

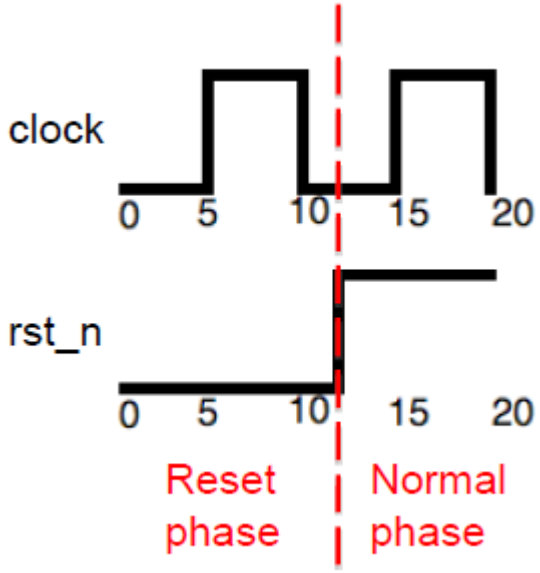
Time control part

```
`timescale 1 ns/10 ps
`define CYCLE 10

[Initialization part]
[Clock part]
[Wave Dump part]

initial begin
    rst_n = 1'b0;
    #12 rst_n = 1'b1;
    #(`CYCLE)rst_n = 1'b0;
    enable_i = 1;

    #400000000
    $finish;
end
```



File IO part

```
`timescale 1 ns/10 ps
`define CYCLE 10
[Initialization part]
[Clock part]
[Wave Dump part]
[Time control part]

reg [12:0] HDRlog [0:85439];
integer i,outfile;
initial begin
   //Input file
   for (i=0; i<85440; i=i+1) begin
      HDRlog[i] = 13'b0;
   end
   $readmemh("log_hdr_hex.txt", HDRlog);

   //Open output file
   outfile = $fopen("out_log_hdr.txt");
end
```

File IO part

- Verilog support two methods to load data into a *reg* array
- Read binary data:
- **`$readmemb("filename", reg_array_name);`**
- Read hexadecimal data
- **`$readmemh("filename", reg_array_name);`**

Input data part

- Looping Style
  - Pros: testbench may be compact
  - Cons: only adequate for patterns with regular timing & values

```
`timescale 1 ns/10 ps
`define CYCLE 10
[Initialization part]
[Clock part]
[Wave Dump part]
[Time control part]
[File IO part]

initial begin
   for (i = 0; i < 85440; i = i + 1)
      @(negedge clk) data_in= HDRlog[i];
   #20
   $finish;
end
```

Output data part

```
`timescale 1 ns/10 ps
`define CYCLE 10
[Initialization part]
[Clock part]
[Wave Dump part]
[Time control part]
[File IO part]
[Input Data part]

always@(*) begin
    if (enable_o)
      $fdisplay(outfile, "%h", data_o);
    if (end_o) begin
      #(`CYCLE)
      $finish;
    end
end
```

Complete module

- Add a Time-Out Condition

 - Because termination condition may never be reached when design is not correct.

```
`timescale 1 ns/10 ps
`define CYCLE 10
`define TIME_OUT 10000

[Initialization part]
[Clock part]
[Wave Dump part]
[Time control part]
[File IO part]
[Input Data part]
[Output Data part]

initial #(`TIME_OUT)
$finish;
```