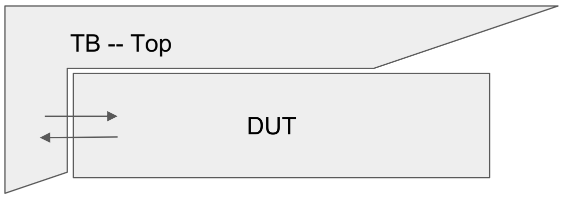
CSCE 616 : Hardware Design Verification

Lab Lecture 1

What is Verification-Testbench

Testbench - HDL code to generate stimuli and verify the DUT(Design under Test).

Consist of driving clock, data, error checking, debug statements and checking.



**In this course, we shall use System Verilog to design out Test-Bench

TB Ports and DUT instantiation

SV Data-types

Structural : logic, wire, reg (Can hold 4 state)

Behavioural : int, bit, real, string, parameter, time (2 state variables)

TB Ports

```
module tb (input logic a, input
logic [7:0] b, output logic [7:0] c);

Or

module tb;
            input logic a;
            input logic [7:0] b;
            output logic [7:0] c;
```

DUT Instantiation

```
module tb (input logic a, input
logic [7:0] b, output logic [7:0] c);

dut dut_inst (.a(a), .b(b), .c(c));
or
dut dut_inst1 (.a, .b, .c);
or
dut dut_inst2 (*);
```

SV Test-Bench

Procedural Blocks

```
initial begin

clk=0;
rst_n=1;
#5 rst_n=0;
#1 rst_n=1;

end
```

```
always begin
#5 clk = ~clk
end

always @(posedge clk)
begin
            a= a+1;
end
```

```
initial begin

fork
#5 a=0;
#2 b=1;
#3 c=3;
join
#5 d=6;
end
```

Blocking statement

```
a=1;b=2; c=3;
b=c; a=b;
```

Non Blocking statement

```
a=1;b=2; c=3;
b <= c; a <= b;
```

SV-Task

Tasks can include timing delays

Tasks have any number of input and output

Task is local to module and can call any other task/function

Model combinational/sequential logic

E.g.

```
task sum (input logic [7:0] a, b, output logic [7:0] c);

#1 c <= a+b;

endtask
```

SV-Function

Function cannot include timing delays

Function can have any number of input and but only one output

Function is local to module and can call any other function but not task

Model combinational logic

E.g.

```
function int sum ( int a, b);
int c;
c = a+b;
return c;

endfunction
```

SV Print Statements

$display -- used to debug as printf in C

E.g. $display (“Hello World”);

$display (“Value of a in binary is %b, in decimal is %d, in hex is %h”, a,a,a);

SV loop

for, while, do...while -- very similar to C

```
E.g. for(int i=0; i <16; i++) begin

          $display(“iter : %d”,i);

     end
```

Lab 1

Simple Memory Specifications

- Simple Memory **is 5-bits wide** and address range is 0 to 31
- Simple Memory access is synchronous with asynchronous reset (**rst_n**)
- Simple Memory **Data Width is 8 bit**
- Write **data_in** into the simple memory on posedge of **clk** only when **wr_en**=1
- Read from simple memory[addr] onto **data_out** bus on posedge of clk only when **rd_en**=1
- Read and Write requests cannot be placed simultaneously
- Reset would write Zeros to all the addresses

Objective -- Create a testbench to verify design and find a bug