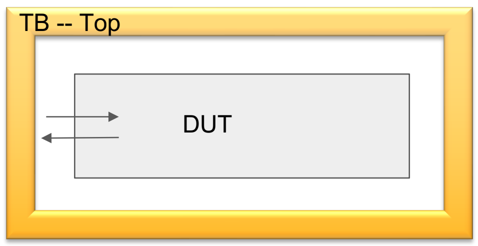
CSCE 616 : Hardware Design Verification

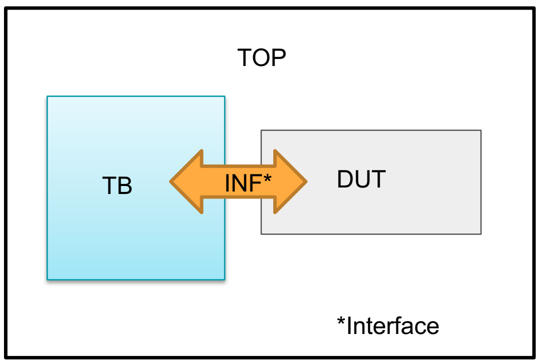
Lab Lecture 2
SV Interface and Randomization Methods

Lab 1 vs Lab 2 implementation

LAB -1



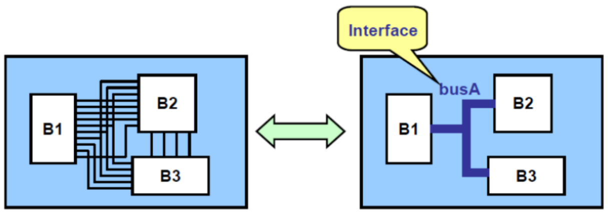
LAB -2



SV-Interface

System Verilog provides interface construct to:

- Encapsulate communication between two hardware blocks.
- Provides a mechanism for grouping together multiple signals into single unit that can be passed around the design hierarchy.



TB – without Interface

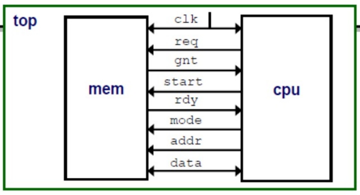
```
module memMod (
  input  logic clk, req, start,
         logic [1:0] mode,
         logic [7:0] addr,
  inout  wire [7:0] data,
  output logic gnt, rdy
  );
...
endmodule
```

```
module cpuMod (
  input  logic clk, gnt, rdy,
  inout  wire [7:0] data,
  output logic req, start,
         logic [7:0] addr,
         logic [1:0] mode
  );
...
endmodule
```

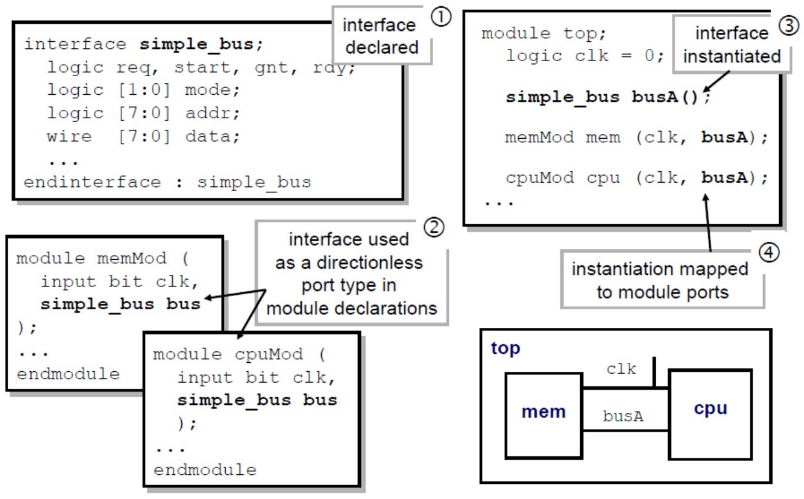
```
module top;
  logic req, gnt, start, rdy;
  logic clk = 0;
  logic [1:0] mode;
  logic [7:0] addr, data;

memMod mem (clk, req, start,
   mode, addr, data, gnt, rdy);

cpuMod cpu (clk, gnt, rdy, data,
   req, start, addr, mode);
.
```



TB – with Interface


interface declared ①
interface used as a directionless port type in module declarations ②
interface instantiated ③
instantiation mapped to module ports ④
top
mem
cpu
clk
busA

```
interface simple_bus;
  logic req, start, gnt, rdy;
  logic [1:0] mode;
  logic [7:0] addr;
  wire  [7:0] data;
  ...
endinterface : simple_bus

module memMod (
  input bit clk,
  simple_bus bus
);
...
endmodule

module cpuMod (
  input bit clk,
  simple_bus bus
  );
...
endmodule

module top;
  logic clk = 0;

  simple_bus busA();

  memMod mem (clk, busA);

  cpuMod cpu (clk, busA);
...
```

Mod-ports

- Modports are used to specify the direction of signals w.r.t the module which uses interface.
  - e.g. If dut is using interface, use direction of interface ports as according to dut.

```
interface intf (input clk);
logic read, enable;
logic [7:0] addr, data;

modport dut (input read, enable, addr, output data);
modport tb (output read, enable, addr, input data);

endinterface: intf
```

Mod-ports (import task)

```
interface intf_AB (input bit clk);
logic ack;
logic ready;
logic send;
logic [31:0] data;

modport source ( input ack, ready, output send, data, import send_data);
 ...
task send_data (input logic send_signal, input logic [31:0] data_bus);
            ...
            // actual task definition

endtask
...

endinterface
```

Random Number Generation

- **$urandom_range(i,j)** : This function returns an unsigned integer in range (i,j)

e.g. // Randomize number between 0 and 100
$display ("Value is %0d",$urandom_range(0,100));

- **$urandom(i)** : This function provides a mechanism to generate pseudo random numbers for a given seed.

e.g. // Randomize number with seed 10
$display ("Value is %0d",$urandom(10));

There are other randomization mechanism which we shall learn in later labs.

Lab 2

Cache Memory Specifications

- Cache-Memory is 5-bits wide and address range is 0 to 31
- Cache-Memory access is synchronous with asynchronous reset
- Write data_in into the cache-memory on posedge of clk only when wr_en=1
- Place cache-memory data onto data_out bus on posedge of clk only when rd_en=1
- Read and Write request cannot be placed simultaneously
- Cache has 4 entries (0-3); there is an entry corresponding to set of 8 addresses, total 32 addresses. i.e. entry 0 incache corresponds to address range 0-7, entry 0: 8-15 and likewise
- Cache Hit - For a read/write request if the cache entry address is same as requested address, read/write requestis served from cache
- Cache Miss - For a read/write request if the cache entry address is different than requested address, read/writerequest is served from main memory
- Cache writeback - For a cache miss, the stored cache entry is written back to main memory and read/writerequest is stored into cache entry

<u>**Objective**</u> -- Create a testbench to verify design and find functional bug