Interfaces

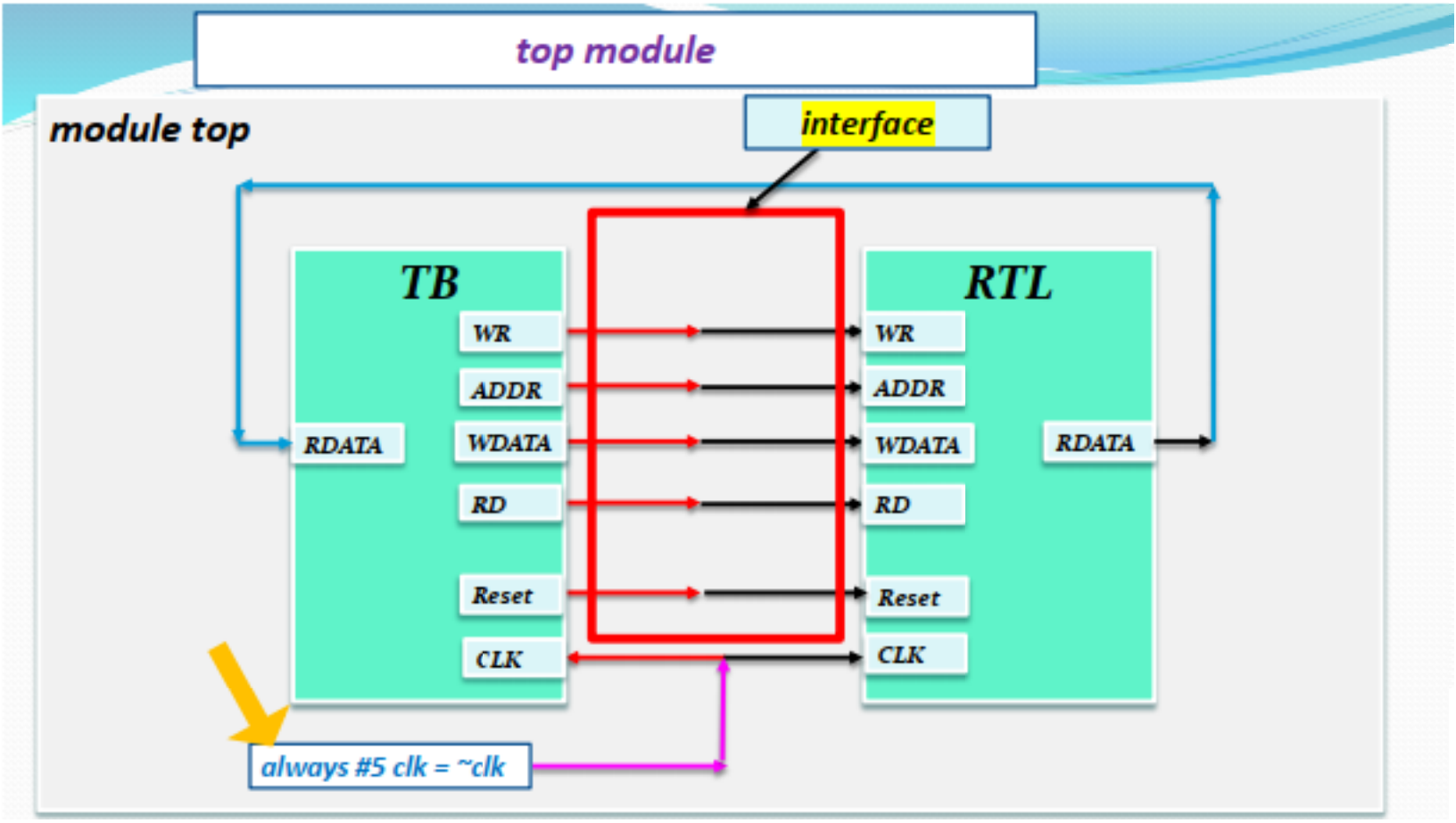
Interface

- *Interface represents a bundle of nets or variables, with intelligence such as synchronization, and functional code.*
- *An interface can be instantiated like a module but also connected to ports like a signal.*

interface *interface_name (arguments);*

<interface_items >

endinterface

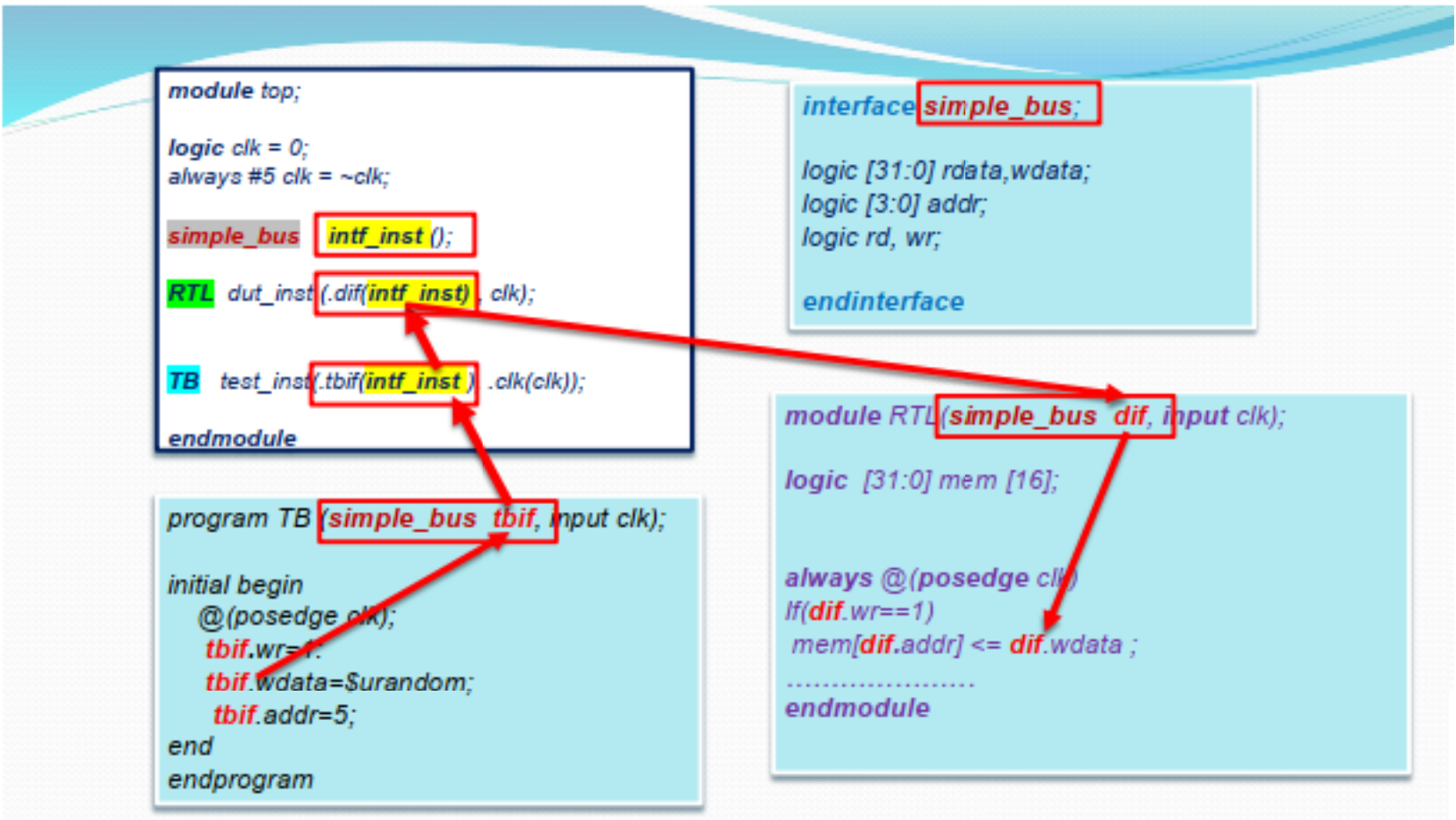

top module
module top
interface
TB
RTL
WR
ADDR
WDATA
RD
Reset
CLK
RDATA
WR
ADDR
WDATA
RD
Reset
CLK
RDATA
always #5 clk = ~clk

Interface as a port

```
interface simple_bus; // Define the interface
logic [31:0] rdata,wdata;
logic [3:0] addr;
logic wr,rd
endinterface
```

```
module RTL (clk,reset,wr,rd,addr,wdata,rdata,response);
```

```
module RTL (simple_bus intf, input clk);
logic [31:0] mem [16];
always @(posedge clk)
If(intf.wr==1)
mem[intf.addr] <= intf.wdata ;
.....
endmodule
```


module top;
logic clk = 0;
always #5 clk = ~clk;
simple_bus intf_inst ();
RTL dut_inst (.dif(intf_inst) , clk);
TB test_inst(.tbif(intf_inst) ,.clk(clk));
endmodule
interface simple_bus;
logic [31:0] rdata,wdata;
logic [3:0] addr;
logic rd, wr;
endinterface
program TB (simple_bus tbif, input clk);
initial begin
@(posedge clk);
tbif.wr=1;
tbif.wdata=$urandom;
tbif.addr=5;
end
endprogram
module RTL(simple_bus dif, input clk);
logic [31:0] mem [16];
always @(posedge clk)
If(dif.wr==1)
mem[dif.addr] <= dif.wdata ;
.....
endmodule

Legacy RTL code

```
module top;
logic clk=0;
always #5 clk = ~clk;

simple_bus intf ();

mem_rtl   dut_inst  (.clk(clk),
                     .rdata(intf.rdata) ,
                     .wdata(intf.wdata),
                     .wr(intf.wr),.rd(intf.rd),
                     .addr(intf.addr));

TB   test_inst   (.tbif(intf ), .clk(clk));

endmodule
```

```
interface simple_bus;
logic [31:0] rdata,wdata;
logic [3:0] addr;
logic rd, wr;
endinterface
```

Modports : Signal directions

- When an **interface** is referenced **as a port**,
  - All **nets in the interface** are assumed to have a **inout direction**
  - All variables in the interface are assumed to be of **ref direction**

- A **modport defines the port direction** that the module sees for the signals in the interface
- **The modport definitions do not contain vector sizes or types.**
- The modport declaration only defines whether the connecting module sees a signal as an **input**, **output**, **inout**, or **ref** port.

```
interface simple_bus (input clk); // Define the interface
logic [31:0] rdata,wdata;
logic [3:0] addr;
logic rd,wr;

modport   dut_ports  (input   addr,wr,rd,wdata ,clk,  output rdata);
modport    tb_ports  (output  addr,wr,rd,wdata,       input rdata,clk);

endinterface: simple_bus
```

Specifying which ***modport*** view to use

- *SystemVerilog provides two methods for specifying which modport view a module interface port should use*

1) ***As part of the interface connection to a module instance***

```
simple_bus    sb_intf(clk);
RTL  rtl_inst    (sb_intf.dut_ports);
TB   test_inst   (sb_intf.tb_ports);
```

2) ***As part of the module port declaration in the module definition***

```
module RTL  (simple_bus.dut_ports  intf);
program TB  (simple_bus.tb_ports  tbif);
```

Modport as part of connection to module

```
module top;
logic clk = 0;

simple_bus  intf_inst ();

RTL  dut_inst (intf_inst.dut_ports, clk);

TB   test_inst (intf_inst.tb_ports,   clk);

endmodule
```

```
interface simple_bus;
logic [31:0] rdata,wdata;
logic [3:0] addr;
logic rd,wr;
modport  dut_ports (input  addr,wr,rd,wdata ,clk,  output rdata);

modport  tb_ports  (output  addr,wr,rd,wdata,  input rdata,clk);

endinterface
```

```
program TB (simple_bus  tbif, input clk);

initial begin
   @(posedge clk);
    tbif.rd=1:
    tbif.addr=5;
end
endprogram
```

```
module RTL ( simple_bus  intf,  input  clk);
logic  [31:0] mem [16];

always @(posedge clk)
If(intf.rd==1)
intf.rdata <= mem[intf.addr];
.....................
endmodule
```

Modport is part of module port declaration

```
module top;
logic clk = 0;

simple_bus  intf_inst ();

RTL  dut_inst (intf_inst  clk);

TB   test_inst (intf_inst,  clk);

endmodule
```

```
interface simple_bus;
logic [31:0] rdata,wdata;
logic [3:0] addr;
logic rd,wr;
modport  dut_ports (input  addr,wr,rd,wdata ,clk,  output rdata);

modport  tb_ports  (output  addr,wr,rd,wdata,  input rdata,clk);

endinterface
```

```
program TB (simple_bus.tb_ports  tbif
input clk);

initial begin
   @(posedge clk);
    tbif.rd=1:
    tbif.addr=5;
end
endprogram
```

```
module RTL ( simple_bus.dut_ports intf,  input  clk);
logic  [31:0] mem [16];

always @(posedge clk)
If(intf.rd==1)
intf.rdata <= mem[intf.addr];
.....
endmodule
```

Parameterized interfaces

```
interface simple_bus #(parameter DWIDTH = 32, AWIDTH  = 4) ;
logic [DWIDTH-1:0] rdata,wdata;
logic [AWIDTH-1:0] addr;
logic wr,rd;
endinterface

module top;
simple_bus   intf_inst1();  DWIDTH=32 AWIDTH=4

simple_bus  #(64,8)  intf_inst2();  DWIDTH=64 AWIDTH=8

simple_bus  #(.DWIDTH(16), .AWIDTH(3) ) intf_inst3() ;  DWIDTH=16 AWIDTH=3
endmodule
```