DESIGN & VERIFICATION DVCON 2014 CONFERENCE & EXHIBITION

Using SystemVerilog Interfaces and Structs for RTL Design

Tom Symons and Nihar Shah
Oracle Labs

Outline

- Why use interfaces in RTL ?
- Problems encountered
- Solutions and work-arounds
- Verification Implications
- Summary
- Recommendations
- Tools Used

Why use interfaces

- Fewer lines of code
  - 25% of lines are for interconnect only
  - Port connections reduced by up to 90%
  - Fewer errors, easier to read, easier to modify
- Ease of managing hierarchy
  - No more duplication of signals up thru hierarchy
- Consistent signal names
  - No need to modify signal names for every instance

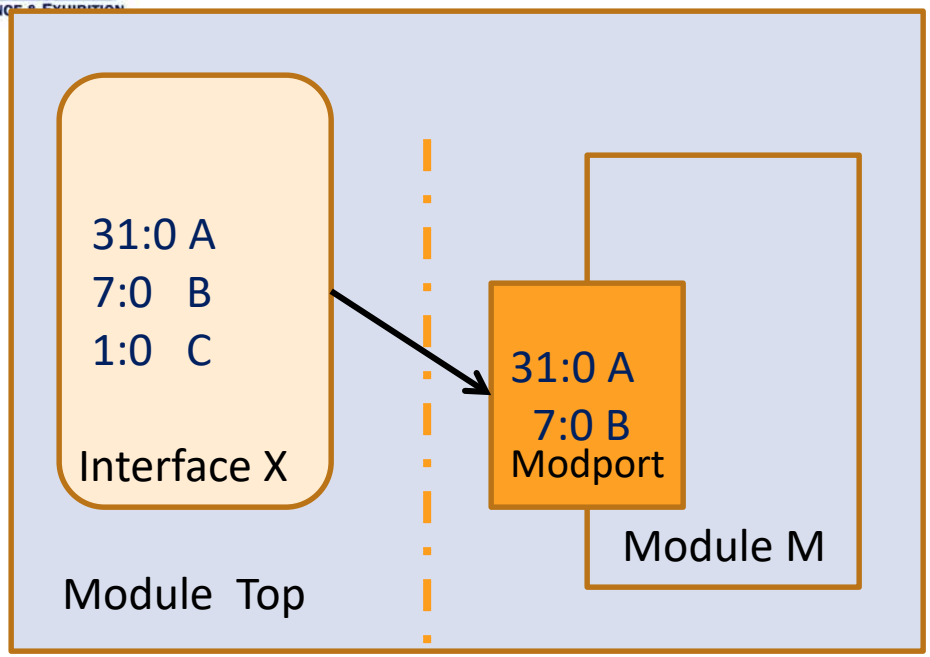
Why use interfaces

- Debug Utility
  - Drag-n-drop interface into wave viewer
  - Easy scripts to format display
  - Add debug signals to help understand activity
  - Interfaces listed in hierarchy viewer
- Assertions
  - Great place for assertions. Automatically added to every instance.

Types of problems

- Interfaces
  - Synthesis boundaries
    - Most problems occurred on synthesis boundaries
    - Solution for synthesis caused downstream tool issues
  - Power domain boundaries
- Structs
  - Only problem occurred when using structs for module ports

Synthesis Boundary Problem

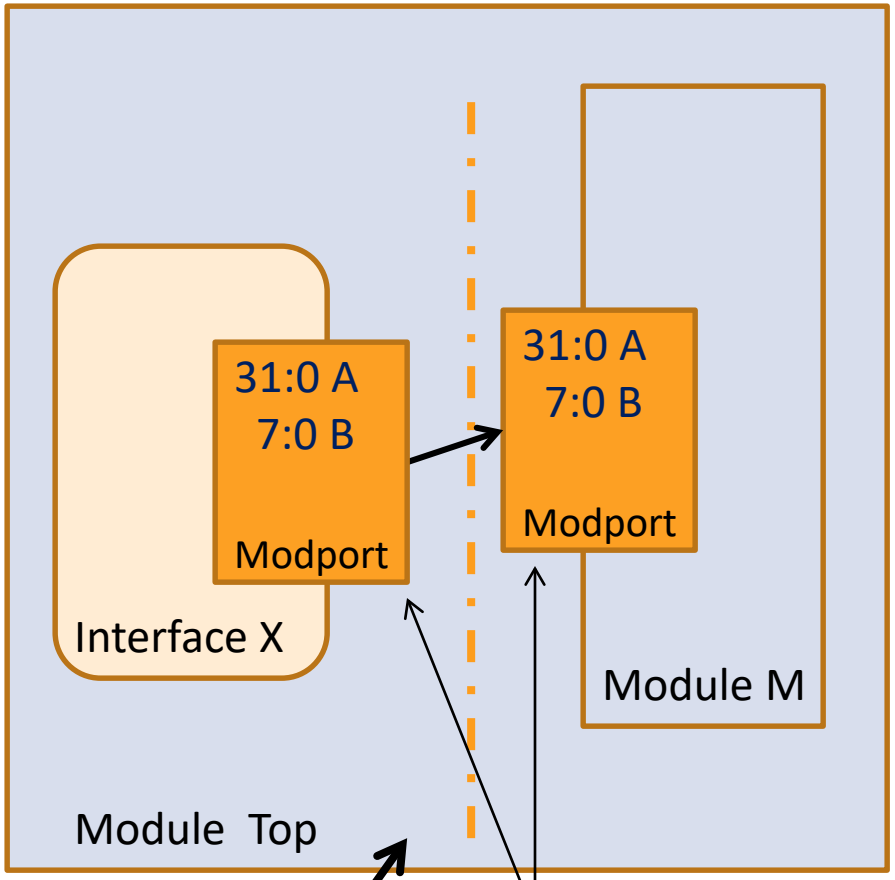



```
module M(
  IntfX xif.mport
);
…
endmodule

module Top;
  IntfX U_if();
  ModA U_A(.xif(U_if));
endmodule
```

Problem! Signal C is used in Top but not M

Synthesis Boundary

Solution




```
module M(
  IntfX xif.mport
);
…
endmodule

module T;
  IntfX U_if();
  ModA U_A(.xif(U_if.mport));
endmodule
```

Specify modport for synthesis.
We call this a source modport reference.

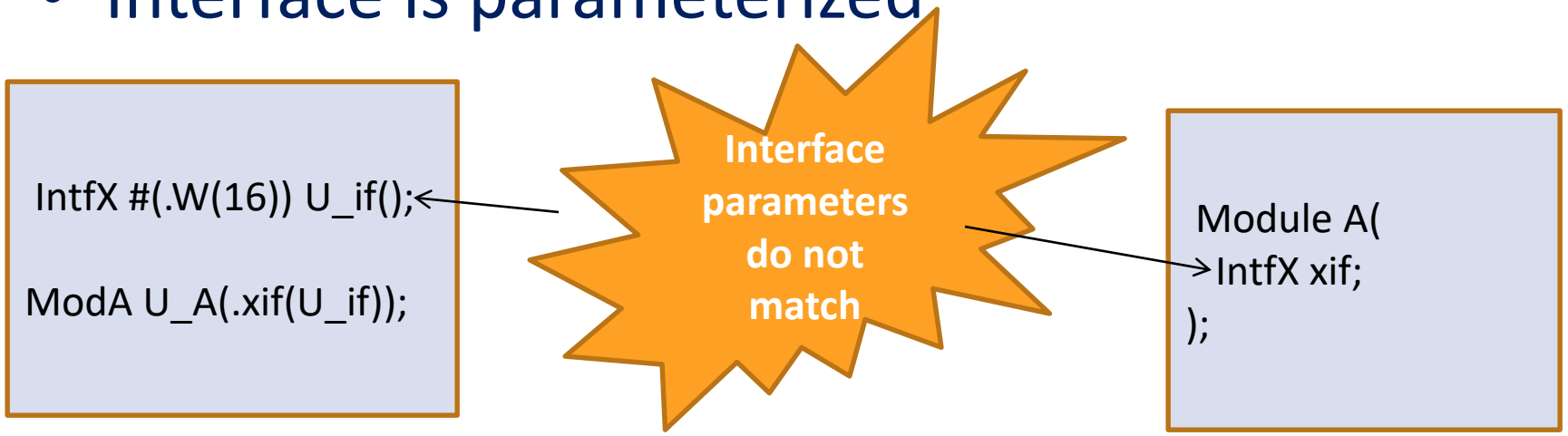
Modport specified on both sides, so synthesis now matches.

Source Modport References

- Downstream tools could not handle source modport references
  - MVSIM
    - Used ifdef to hide modport reference
  - MBIST
    - Used pragma to hide modport reference

More Boundary Problems

- Interface is parameterized



Solution 1: Specify parameter values to DC

Solution 2: Duplicate interface

Duplicating Interfaces

```
Interface IntfX16(
  parameter W = 16
)
`include “intfx.svh”
…
endmodule
```

Duplicate interfaces uses different parameter default values.

Put common implementation in a single include file.

UPF Issues

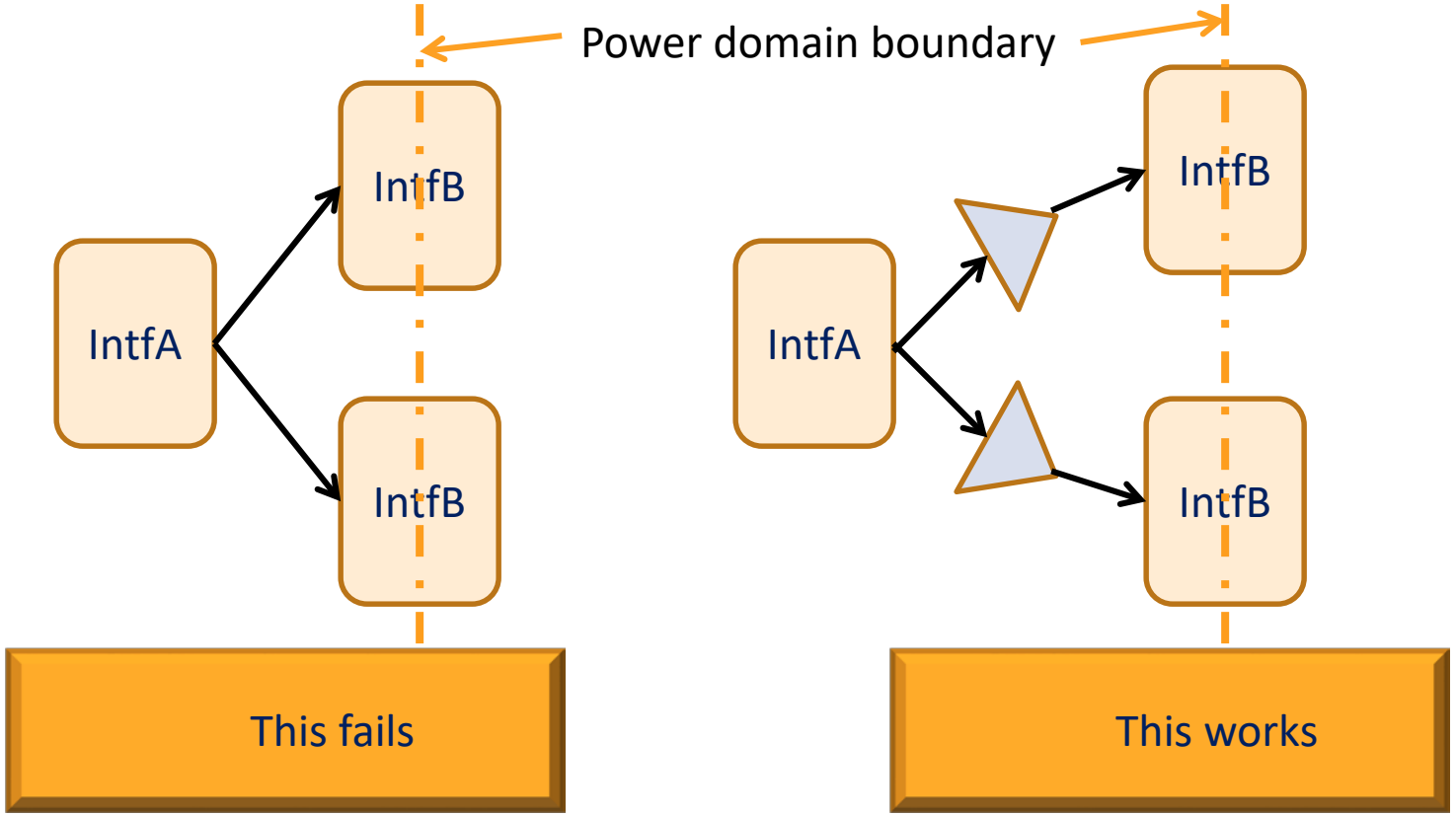
- Period notation for interface.signal
  - Synthesis requires periods here be replaced with underbars
  - But UPF file still required periods
    - Maintain two copies of UPF file, one with periods and one with underbars. Yuck, but it works.

UPF Issues – MVSIM

- No support for interface references in UPF
  - Could not specify default isolation policy for all signals in interface
  - Solution: Specify isolation policies for each signal in interface

More MVSIM Issues

- Fanout from interface to interface fails



Structs

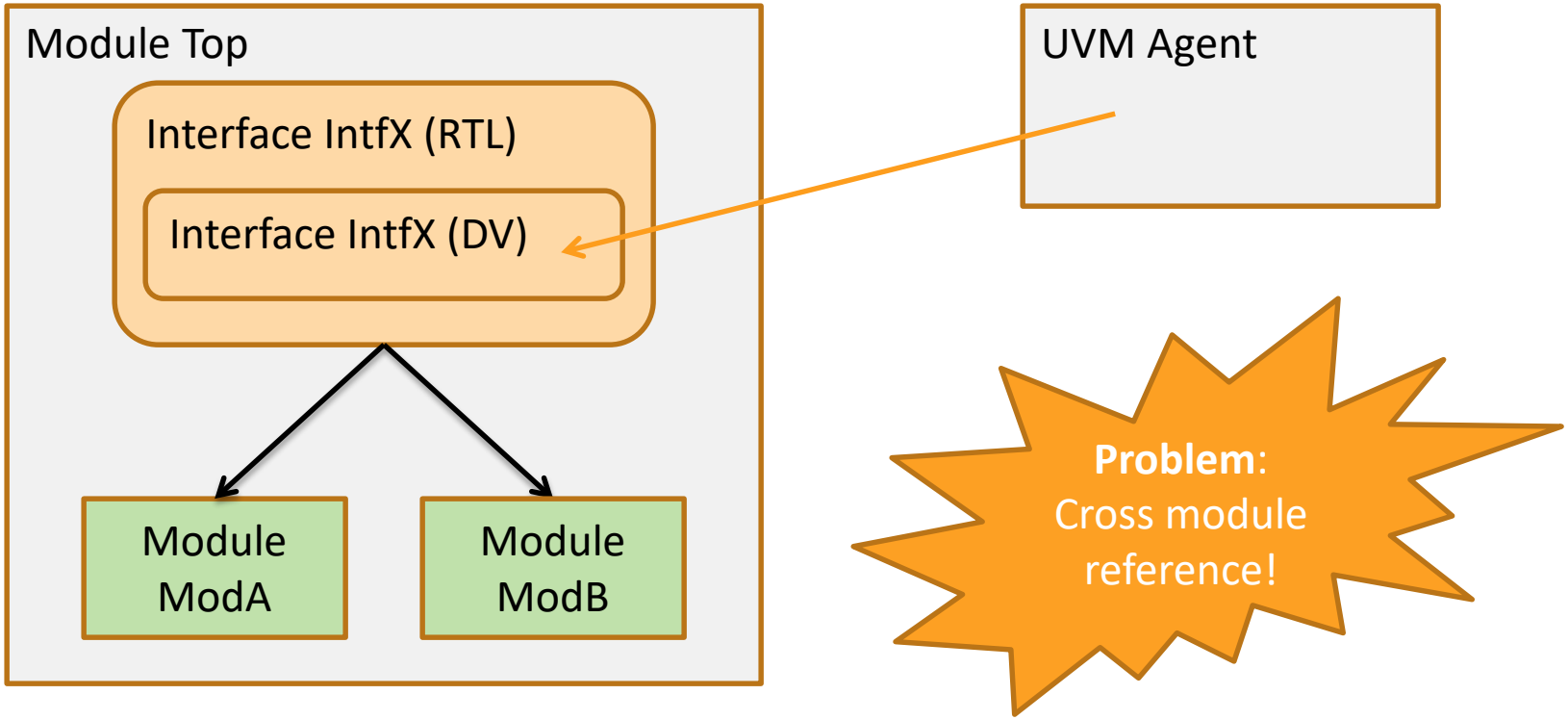
- Great for defining registers
- One struct for each register
- One struct for unit, containing all registers
  - Auto-generated structs ensured consistency across RTL, DV, documentation
- Create output port with entire unit struct
- Only issue was that package had to be imported above module definition

Interfaces for verification

- We CANNOT reuse the same RTL interface for verification!
  - **Parameters**: DV can't live with them, RTL can't live without them!
  - **Cross module references**: Testbench reaches deep into RTL hiearchy to access the interface.
  - **Duplicate RTL interfaces:** DV cannot cope with duplicate interface types for same bus

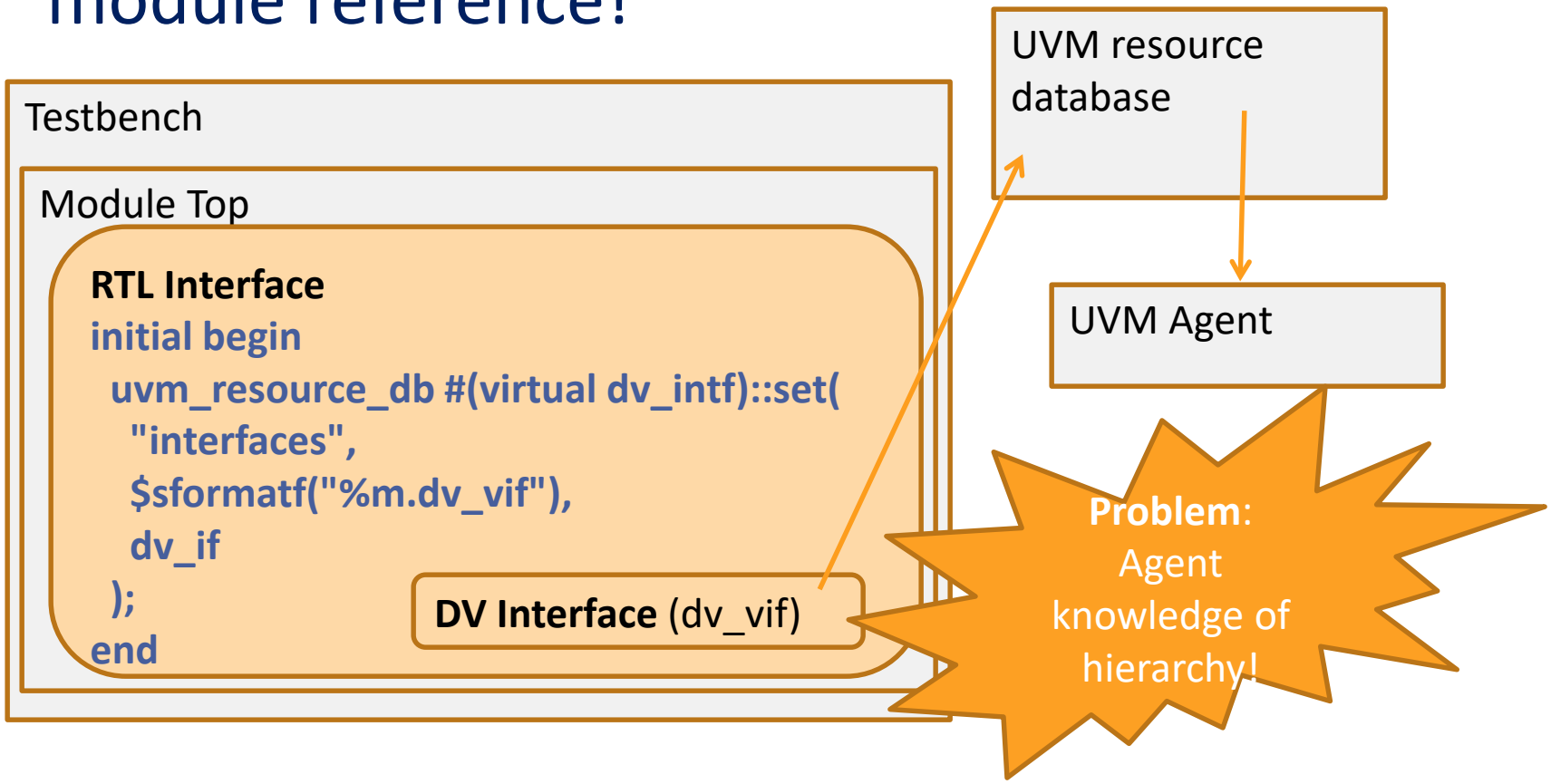
Embedded DV Interface

- Unparameterized DV interface embedded inside parameterized RTL interface



Connecting the DV Interface

- Use the UVM resource db to avoid cross module reference!



Connecting to the Agent

- The testbench will distribute interfaces to the correct agents.

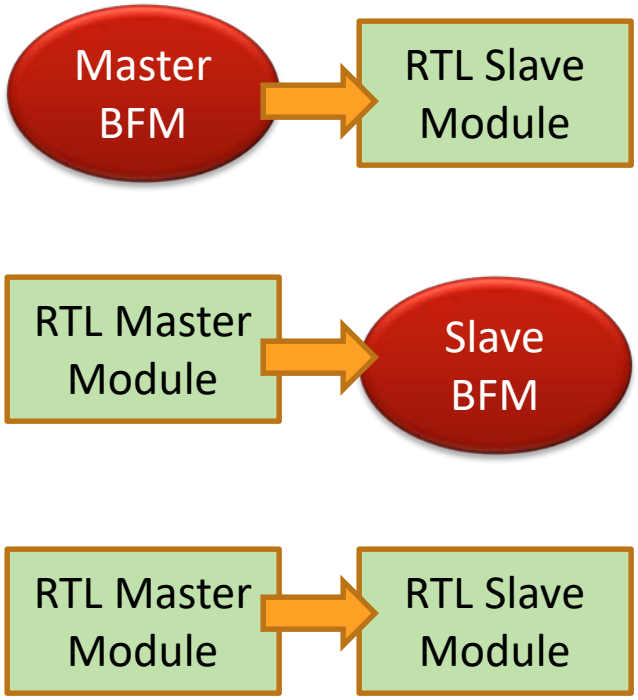
```
Testbench

if (uvm_resource_db #(virtual dv_if)::read_by_name(
    "interfaces“,
    $sformatf("%m.U_dut.rtl_if.dv_vif"),
    dv_vif)
  )
    uvm_config_db #(virtual dv_if)::set(
       null,
       "uvm_test_top.env.dv_agt“,
       "dv_vif",
       dv_vif);
```

UVM resource database

UVM Agent

Using a Master Agent

- What if the RTL interface needs to be driven by the verification environment?




```
interface rtl_intf #(
  parameter DV_DRVR = 0,
  parameter DV_MSTR = 0,
)
generate
if (DV_DRVR == 1) begin
  if (DV_MSTR == 1)
    assign valid_rtl = dv_if.valid_dv;
  else
    assign dv_if.valid_dv = valid_rtl;
end else begin  //DEFAULT!
    assign dv_if.data_dv = valid_rtl;
end
endgenerate
```

Connecting the DV and RTL IF

- DV signals in port list as inouts

RTL Interface

```
interface rtl_intf
#(parameter BUSW = 32)
(input clk, input rst_n);

  logic  [BUSW-1:0] data;
  logic  valid;
  logic  ack;

`ifdef TBBUILD
  wire valid_dv;
  wire [`MAXW-1:0]data_dv;
  wire ack_dv;
  dv_intf dv_if(.*);
…
```

DV Interface

```
interface dv_intf (
  input clk,
  input rst_n,
  inout valid_dv,
  inout [`MAXW-1:0] data_dv,
  inout ack_dv
);

  clocking mcb @(posedge clk);
    …
  endclocking
  modport master_mp (clocking
mcb);
…
```

1. **Avoids cross module reference:** verilog "bind" to another module
2. **Convenience**: Allows for implicit port connection (.* notation)
3. **Master or Slave**: "inout" allows modports to decide the direction.

Summary

- Successful first silicon with modest use of interfaces and structs
  - 180 interface instances
  - 100's of struct instances
- Almost all issues occurred for interfaces on synthesis or power domain boundaries
- Key to accommodate RTL interface solutions was an embedded DV interface.

Recommendations

- User recommendations
  - Add a single interface to your RTL. Pressure tool vendors if you see problems.
  - Avoid arrays of interfaces in TB or RTL if using Verdi
- Recommendations to IEEE committee:
  - Ability to reference a parameterized virtual interface without specifying the parameters
  - Ability to reference itself (“this”)

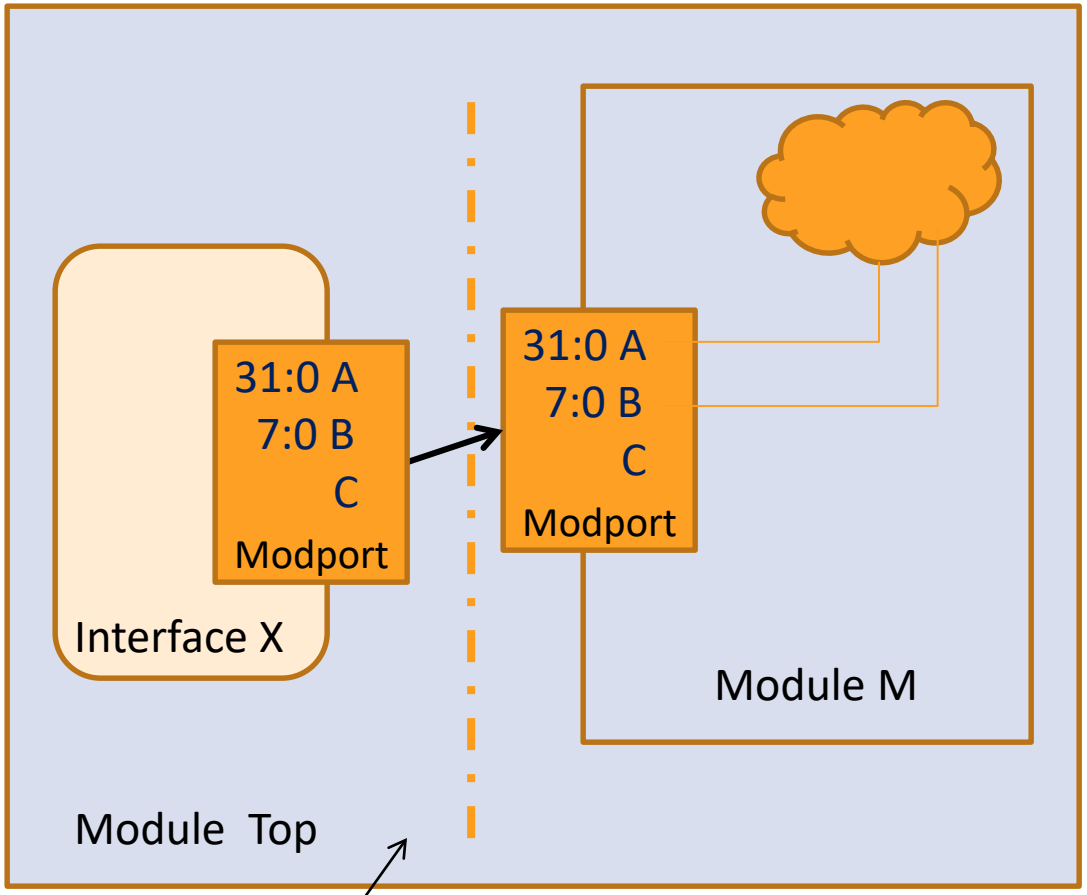
Tools Used

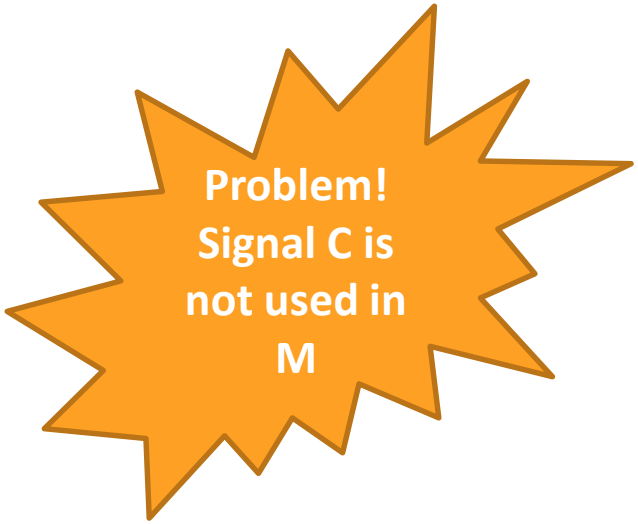
- Tools with no issues
  - VCS from Synopsys
  - LEC from Cadence
- Tools with issues
  - Design Compiler from Synopsys
  - MVSIM from Synopsys for low-power sims
  - Tessent MemoryBist from Mentor
  - LEDA from Synopsys

Questions

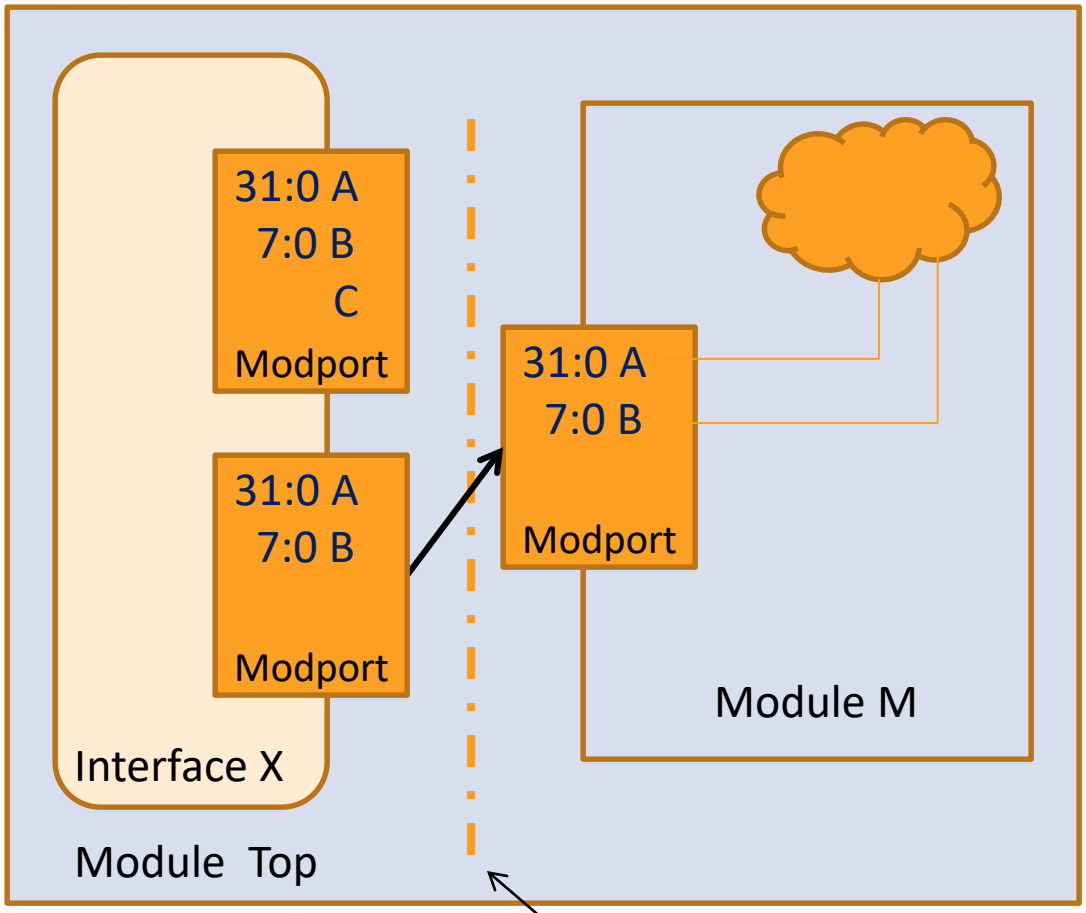
Backup

Synthesis Boundary Problem





Solution



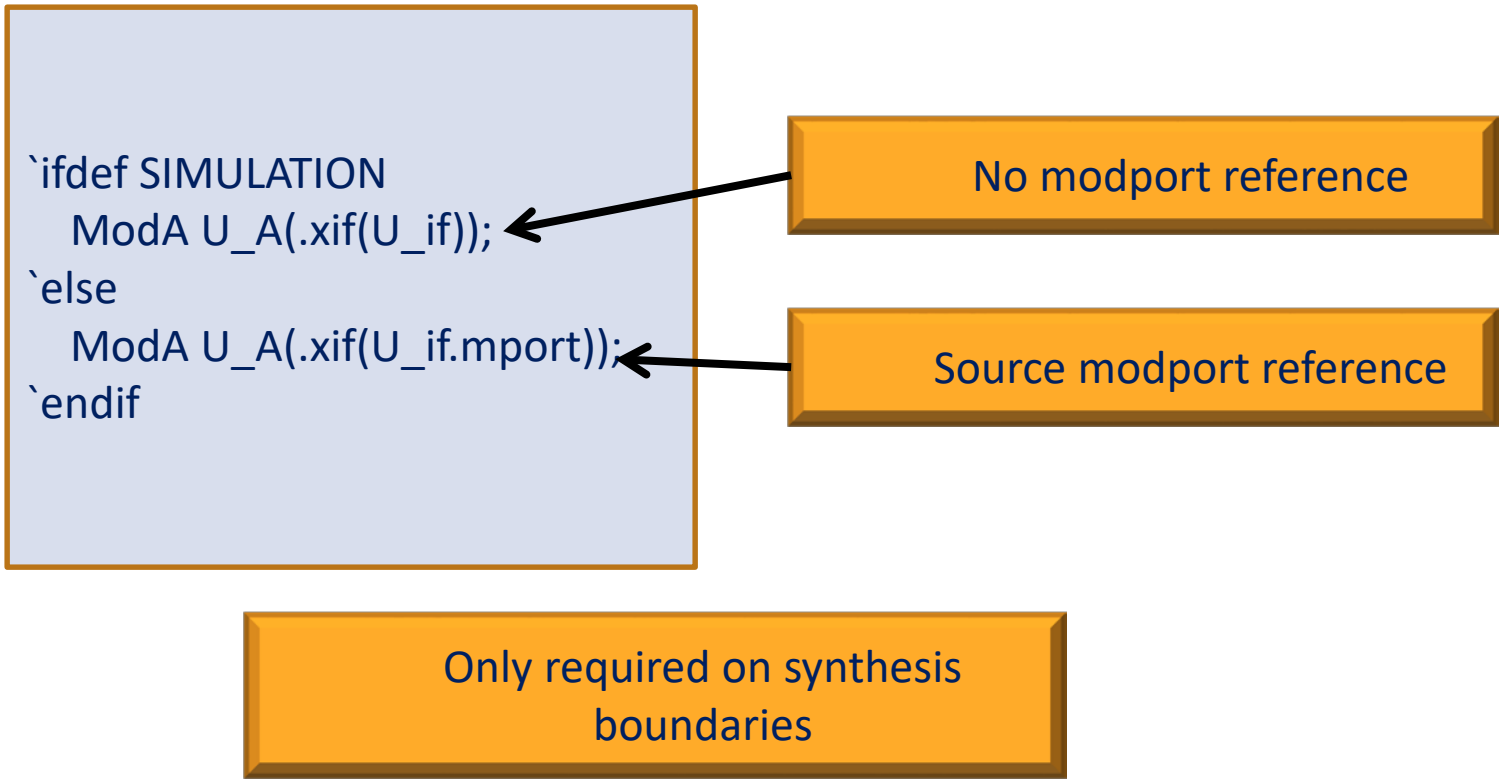
Create a new modport with signal C

Extra modports are free. Add as many as you need to match all your connections.

Use same solution if module M only uses a slice of a bus.

Simulation Issues – MVSIM

- No support for source modport references



DFT Issues

- Mbist tool did not support source modport references
  - Solution: Hide interface port references under pragmas

```
unitX U_dut(
  .clk(clk),
//mbist_etassemble translate_off
    .xif(U_if.mportA),
// mbist_etassemble translate_on
  …
```

```
module unitX(
  input clk,
  …
//mbist_etassemble translate_off
  IntfX xif,
//mbist_etassemble translate_on
);
```