Interface/Modport Improvements

Issues:

- LRM weaknesses *must* be fixed!
- We need robust semantics for modport direction
- Failure to support RTL design-by-contract
- Virtual interfaces are a mess
- Poor usability even for current use models

Jonathan Bromley, Verilab jonathan.bromley@verilab.com

SystemVerilog

verilab

Interfaces Are Broken (1): Types

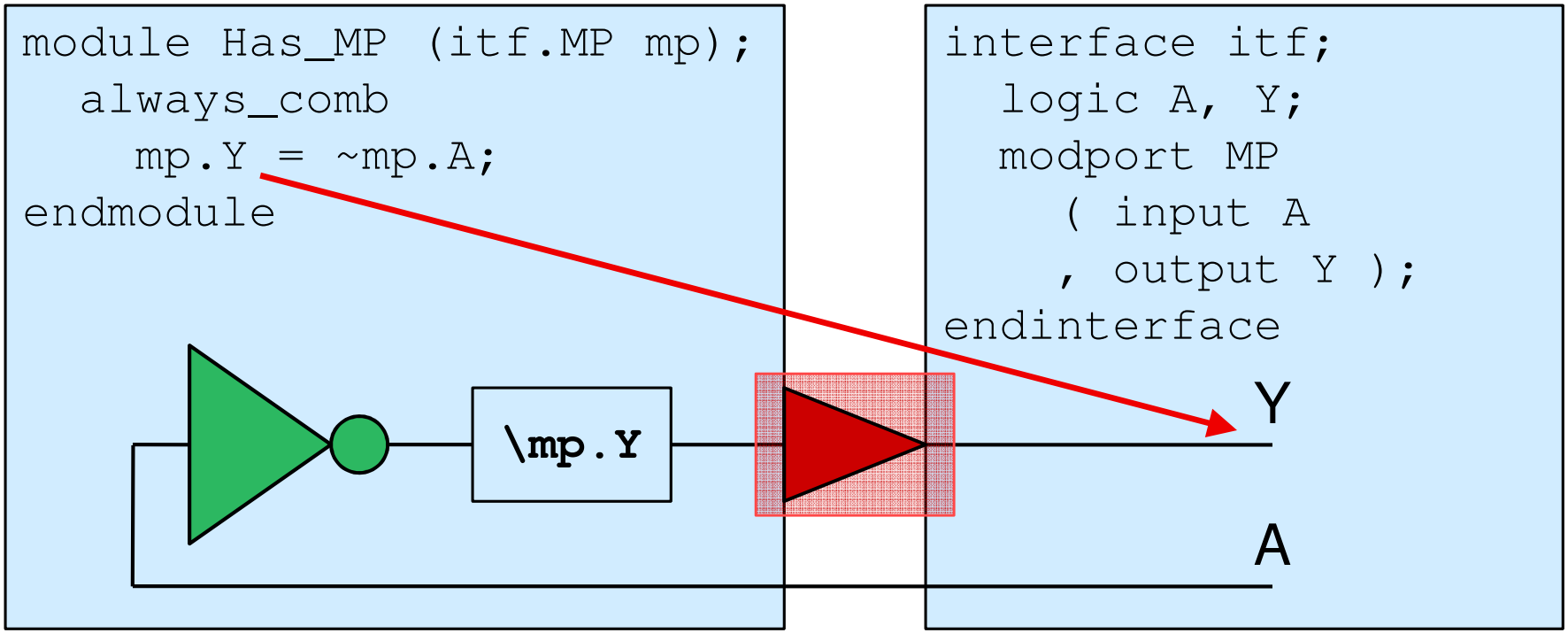
- Interface definition has two roles:
  - data type for virtual interface declaration
  - a "thing" you can instantiate
- Instances can be parameterized
  - This affects the data type
  - Virtual interface needs to know full specification of the interface it references
- SV-2009 changes reflect these issues

SystemVerilog

verilab

Interfaces Are Broken (2): Modports

- What is an *output* modport connection?

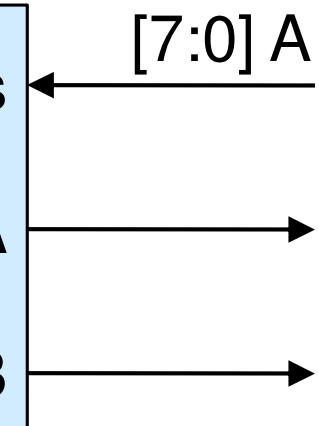


- Possible solution: allow only `ref` access from a virtual interface?

Design by Contract (1): Ports

- Ports are a contract between a module and its environment

```
module Decode
  ( input [7:0] adrs
  , output reg sel_A
  , output reg sel_B
  );
  always @* begin
    sel_A = (adrs == 8'h42);
    sel_B = (adrs < 'h80);
  end
endmodule
```



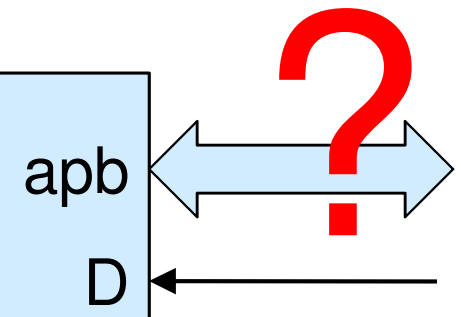
Design by Contract (2): Interfaces?

- What's the contract?

```
module APB_Parallel_Input
  #(parameter ADRS=16'hFF42)
  ( interface.Slave apb
  , input [7:0] D
  );

  always @(posedge apb.PCLK)
    if (apb.PSEL & !apb.PWRITE
        & (apb.PADDR == ADRS) )
      apb.PRDATA <= D;

endmodule
```



any interface with a Slave modport is OK

```
interface Alien;
  int PRDATA; logic Junk;

  modport Slave
    (input PRDATA, inout Junk);
```

Modport as a Data Type

- Making the contract explicit

```
package APB_contract;
  modport Slave
    ( input logic PCLK
    , output logic PREADY
    , output logic [7:0] PRDATA
    , ... );
endpackage
```

```
module APB_Parallel_Input
  #(parameter ADRS=16'hFF42)
  ( APB_contract::Slave apb
  , input [7:0] D );

  always @(posedge apb.PCLK)
    if (apb.PSEL & !apb.PWRITE
        & (apb.PADDR == ADRS) )
      apb.PRDATA <= D;

endmodule
```

SystemVerilog

verilab

Providing the Modport

- Any interface can expose such a modport:

```
package APB_contract;
  modport Slave
    ( input logic PCLK
    , output logic PREADY
    , output logic [7:0] PRDATA
    , ... );
endpackage
```

modport expressions

```
interface APB (input logic clock);
  import APB_contract::*;
  logic PSEL_1, PSEL_2,...
  ...
  Slave slave1 ( .PSEL(PSEL1), .* );
  Slave slave2 ( .PSEL(PSEL2), ....
  ...
```

modport instances

Other Improvements?

- Parameterized modport
- Interface parameters exposed through modport
- Local alias for components of a modport at the point of use
- Renaming of imported/exported subprograms

More details in Mantis 1861 white-paper

SystemVerilog

verilab

Everyone Wants Interface Inheritance

- Array of heterogeneous virtual interfaces? **NO!**
  - Virtual interfaces are strongly typed
    - SV-2009 change, sidesteps many troublesome issues
- Extend an interface or modport with a few extra features? **NO!**
- Connect subset device to superset bus? **NO!**
- A modport to group several other modports? **NO!**

Interface Inheritance Challenges

- Superficially simple, like class inheritance:

```
interface Base;
  logic L;
endinterface
```

```
interface Child extends Base;
  wire W;
endinterface
```

will do duty for (*is a*) Base

- But the connection contract is bidirectional:
  - if I connect a Child to a module that expects a Base...
  - ... W is undriven
- even worse for task/function export

SystemVerilog

verilab