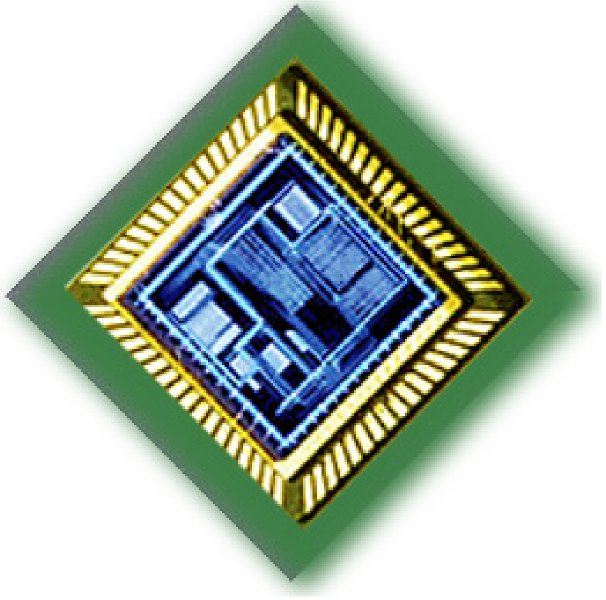


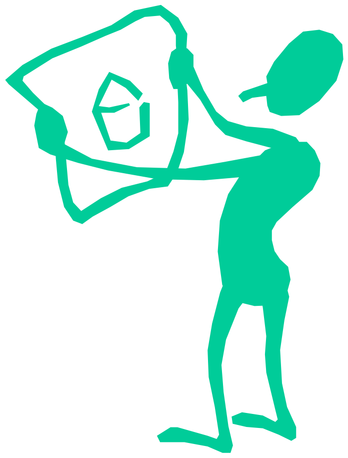
SystemVerilog interface

Curso do Brazil-IP

Felipe Gonçalves Assis
UFCG
fgassis@lad.dsc.ufcg.edu.br

Roteiro

1)O que é
2)Declaração
3)Uso em modules
4)Instanciação e conexão
5)Virtual interface
6)Uso



Roteiro

7) Conexão
8) Modports
9) Tasks
10) Tasks e modports
11) Nossas macros

O que é

- Grupo de variáveis e interfaces.
- Pode ter portas.
- Pode ser uma porta.

- Apenas variáveis em portas podem ser compartilhadas com o mundo exterior.

Declaração

```
interface my_if(input clk, reset);
   logic[7:0] data;
    other_if inner_interface;
endinterface
```

Uso em modules

```
module duv(my_if ports);
    ...
    @(posedge ports.clk)
        if (ports.reset)
            ports.data <= '0;
    ...
endmodule
```

Instanciação e Conexão

```
module tb();
   logic clk, reset;
   my_if my_rif(.clk(clk), .reset(reset));
   duv m_duv(.ports(my_rif));
   ...
endmodule
```

virtual interface

- Referencia uma interface.
- Pode ser usada em classes!

Uso

```
class driver extends uvm_component;
   virtual my_if ports;
   ...
      while (!ports.reset)
         @(posedge ports.clk);
   ...
endclass
```

Conexão

```
module tb();
   logic clk, reset;
   my_if my_rif(.clk(clk), .reset(reset));
   driver driver_h = new(...);
   ...
      driver_h.ports = my_rif;
   ...
endmodule
```

modports

- Determinam como um componente pode acessar a interface.

modports

```
interface my_if(input clk, reset);
   logic[7:0] data;
   logic valid;
   logic ready

   modport transmitter(
      output data, valid,
      input ready, clk, reset);

   modport receiver ( ... );
endinterface
```

modports

```
module duv(my_if.transmitter ports);
   ...
   @(posedge ports.clk)
      if (ports.reset)
         ports.data <= '0;
   ...
endmodule
```

modports

```
module duv(my_if ports);
    ...
    @(posedge ports.transmitter.clk)
        if (ports.transmitter.reset)
            ports.transmitter.data <= '0;
    ...
endmodule
```

tasks

```
interface my_if(input clk, reset);
   ...
   task put(logic[7:0] d);
      ...
   endtask

   task get(output logic[7:0] d);
      ...
   endtask
endinterface
```

tasks

```
class driver extends uvm_component;
   virtual my_if ports;
   ...
      ports.put(sample.data);
   ...
endclass
```

tasks e modports

```
interface my_if(input clk, reset);
   ...
   task put(input[7:0] d);
      ...
   endtask

   modport driver(
      ..., import put);
endinterface
```

Nossas Macros

- Arquivo bvm_macros.svh.
- Declaração automática de Driver, Monitor e Responder.
  - Encapsulamento das funções essenciais na interface.
- Exigências
  - Tasks {driver,monitor,responder}_initial(), do_{drive,monitor,respond}().
  - Modports driver, responder, monitor, com imports das devidas tasks.

Nossas Macros

```
task driver_initial();
   ...
endtask

task monitor_initial();
   ...
endtask

task responder_initial();
   ...
endtask
```

Nossas Macros

```
task do_drive(packet sample);
...

task automatic do_respond(
   ref packet sample);
...

task automatic do_monitor(
   ref packet sample);
...
```

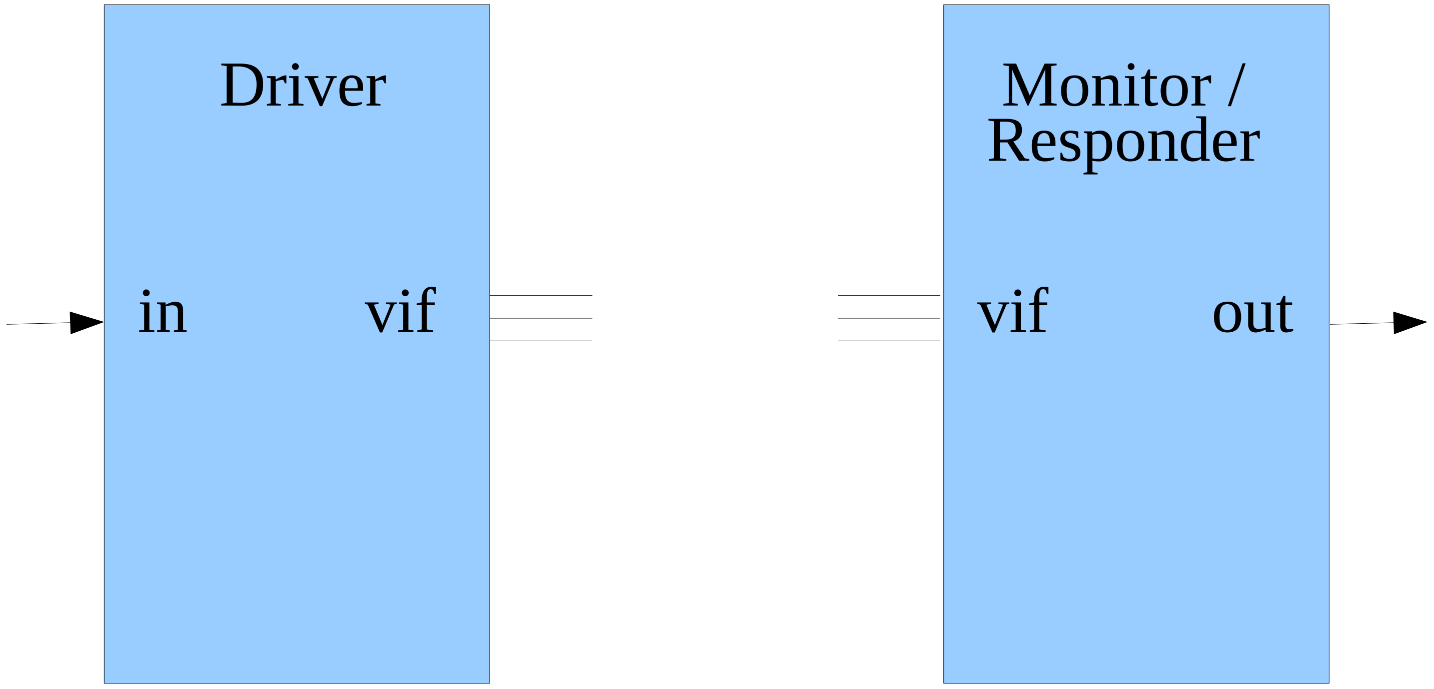
Nossas Macros

```
`include “./bvm_macros.svh”

interface my_if(...);
   ...
endinterface

`bvm_interface_utils(my_if, packet)
```

Interface dos Módulos



Instanciação e conexão

```
my_if_driver driver_h = ...;
my_if_monitor monitor_h = ...;
my_if_responder responder_h = ...;

driver_h.vif = my_rif;
driver_h.in.connect(...);
monitor_h.out.connect(...);
responder_h.out.connect(...);
```

Referência

IEEE Standard for SystemVerilog—
Unified Hardware Design, Specification,
and Verification Language, 2005. **Chapter 20**.