Driving into wires

Driving into signals

(Driving_into_wires_md.sv)

```
1  module ld_reg #(SIZE=8)
2      (input logic clk, ld,
3          input logic[SIZE-1:0] d_in,
4          output logic[SIZE-1:0] r_out, d_out,k_out,
5          inout logic[SIZE-1:0] data1, data2, data3);
6
```

- 1) module ports defined as direction "**output**" can be written from an
  - always block or
  - A single "assign statement within or outside the always

```
always @ (posedge clk) begin : FF_LD
    r_out <= d_in;
    assign d_out=8'b10X_XZZ0;
end : FF_LD
```

- 2) module ports defined as direction **"inout"**
  - must be driven by the "**assign**" statement.

```
19      assign data1 = ~ r_out;
20      assign k_out=8'b10X_XZZ0;
21  endmodule : ld_reg
```

Driving_into_wires_if.sv | Driving_into_wires_md.sv

```
1  module ld_reg #(SIZE=8)
2      (input logic clk, ld,
3          input logic[SIZE-1:0] d_in,
4          output logic[SIZE-1:0] r_out, d_out,k_out,
5          inout logic[SIZE-1:0] data1, data2, data3);
6
7      logic a, b=1'b1; // local variable
8      wire[SIZE-1:0] wdata1, wdata2, wdata3;
9      always @ (posedge clk) begin : FF_LD
10         // The begin statement is only needed
11         // if multiple statements in body of always
12         r_out <= d_in;
13         // data1 <= d_in; // illegal
14         // line above:  A net is not a legal lvalue in this context
15         // wdata1 <= d_in; // illegal
16         // line above:  A net is not a legal lvalue in this context
17         assign d_out=8'b10X_XZZ0;
18     end : FF_LD
19     assign data1 = ~ r_out;
20     assign k_out=8'b10X_XZZ0;
21 endmodule : ld_reg
22
```

Not recommended as a style from an `always` or `always_ff` with clocking

OK from `always_comb` or `always` used as combinational

SV Interface

Driving_into_wires_if.sv

- Nets (declared as wire) behave the as wires, same as in modules.

```
2  parameter SIZE=8;
3  interface test_if (input logic clk);
4      logic ld;
5      logic[SIZE-1:0] d_in;
6      logic[SIZE-1:0] r_out, r_out2;
7      wire logic[SIZE-1:0] data1, data2, data3;
```

```
42 module top2;
43     bit clk;
44     test_if m_if(clk);
45     initial forever #10 clk=!clk;
48     task t();
49         @ (posedge clk) begin
50             m_if.r_out <= m_if.d_in;
51             //m_if.data1 <= m_if.d_in;
52             // A net is not a legal lvalue in this context
53         end
54
55     endtask : t
```

SV interface and classes

- Tasks in classes have the same rules as tasks or always blocks from modules when driving into interface variables.
  - Exception for clocking blocks (see next slide)

```
2  parameter SIZE=8;
3  interface test_if (input logic clk);
4      logic ld;
5      logic[SIZE-1:0] d_in;
6      logic[SIZE-1:0] r_out, r_out2;
7      wire logic[SIZE-1:0] data1, data2, data3;
8      clocking driver_cb @ (posedge clk);
```

```
class c;

    virtual interface test_if vif_bdl;
    virtual task t();
        @ vif.driver_cb begin
            // vif_bdl.data3 <= 8'b10ZZ_ZZ01; // illegal
            //  line above: A net is not a legal lvalue in this context
        end
    endtask : t
```

SV interface and clocking blocks

- **1800-2012 14.3 Clocking block declaration**

  * A *clockvar* whose clocking_direction is **inout** shall behave as if it were two *clockvars*, one **input** and one **output**, having the same name and the same clocking_signal.

  * Reading the value of such an **inout** *clockvar* shall be equivalent to reading the corresponding **input** *clockvar*.

  * Writing to such an **inout** *clockvar* shall be equivalent to writing to the corresponding **output** *clockvar*.

```
2  parameter SIZE=8;
3  interface test_if (input logic clk);
4      logic ld;
5      logic[SIZE-1:0] d_in;
6      logic[SIZE-1:0] r_out, r_out2;
7      wire logic[SIZE-1:0] data1, data2, data3;
8      clocking driver_cb @ (posedge clk);
9          default input #2 output #3;
10         output data1,r_out2;
11         input d_in;
12         inout data2;
13     endclocking : driver_cb
14     modport drvr_if_mp (clocking driver_cb);
15 endinterface :test_if
16
```

```
2  parameter SIZE=8;
3  interface test_if (input logic clk);
4      logic ld;
5      logic[SIZE-1:0] d_in;
6      logic[SIZE-1:0] r_out, r_out2;
7      wire logic[SIZE-1:0] data1, data2, data3;
8      clocking driver_cb @ (posedge clk);
9          default input #2 output #3;
10         output data1,r_out2;
11         input d_in;
12         inout data2;
13     endclocking : driver_cb
14     modport drvr_if_mp (clocking driver_cb);
15 endinterface :test_if
16
```

```
17 class c;
18     virtual interface test_if.drvr_if_mp vif;
19     virtual interface test_if vif_bdl;
20     virtual task t();
21         @ vif.driver_cb begin
22             vif.driver_cb.r_out2 <= vif.driver_cb.d_in;
23             vif.driver_cb.data2 <= vif.driver_cb.d_in;
37         end
38     endtask : t
39
40 endclass : c
```

✓

```
2  parameter SIZE=8;
3  interface test_if (input logic clk);
4      logic ld;
5      logic[SIZE-1:0] d_in;
6      logic[SIZE-1:0] r_out, r_out2;
7      wire logic[SIZE-1:0] data1, data2, data3;
8      clocking driver_cb @ (posedge clk);
```

```
42 module top2;
43     bit clk;
44     test_if m_if(clk);
45     initial forever #10 clk=!clk;
46     c c_h=new();
47
48     task t();
49         @ (posedge clk) begin
50             m_if.r_out <= m_if.d_in;
51             //m_if.data1 <= m_if.d_in;
52             // A net is not a legal lvalue in this context
53         end
54
55     endtask : t
56
```