

Subject: errata/13: PROPOSAL - 12.2.2: bad documentation

ISSUE 13 notes that the current example at the end of section 12.2.2.1 shows named parameter passing before it is described (in the next section). I agree that the example should be modified. I also agree that additional examples should be added to section 12.2.2.2. Proposals below (better formatting also in the proposed examples).

SEE: Example at the end of section 12.2.2.1 and the paragraph just before the example.

WAS: Consider the following example, where the parameters within module instance `mod_a` are changed during instantiation. (*and replace the example*)

PROPOSED: Consider the following example, where the parameters within module instances `mod_a` and `mod_c` are changed during instantiation.

```
module tb1;
  wire [9:0] out_a;
  wire [4:0] out_b, out_c;
  reg [9:0] in_a;
  reg [4:0] in_b, in_c;
  reg      clk;

  // testbench clock & stimulus generation code ...

  // Three instances of vdff with positional parameters
  // mod_a has new parameter values size=10 and delay=15
  // mod_b has default parameters (size=5, delay=1)
  // mod_c has one default size=5 and one new delay=12
  vdff #(10,15) mod_a (.out(out_a), .in(in_a), .clk(clk));
  vdff          mod_b (.out(out_b), .in(in_b), .clk(clk));
  vdff #( 5,12) mod_c (.out(out_c), .in(in_c), .clk(clk));
endmodule

module vdff (out, in, clk);
  parameter size=5, delay=1;
  output [size-1:0] out;
  input [size-1:0] in;
  input      clk;
  reg [size-1:0] out;

  always @(posedge clk)
    #delay out = in;
endmodule
```

SEE: Section 12.2.2.2. Add the following to the end of section 12.2.2.2

PROPOSED:

Consider the following example, where both parameters of `mod_a` and only the second parameter of `mod_c` are changed during instantiation.

```
module tb2;
  wire [9:0] out_a;
  wire [4:0] out_b, out_c;
  reg [9:0] in_a;
  reg [4:0] in_b, in_c;
  reg      clk;
```

```

// testbench clock & stimulus generation code ...

// Three instances of vdff with named parameters
// mod_a has new parameter values size=10 and delay=15
// mod_b has default parameters (size=5, delay=1)
// mod_c has one default size=5 and one new delay=12
vdff #(.size(10), .delay(15))
        mod_a (.out(out_a), .in(in_a), .clk(clk));
vdff      mod_b (.out(out_b), .in(in_b), .clk(clk));
vdff #(.delay(12)) mod_c (.out(out_c), .in(in_c), .clk(clk));
endmodule

module vdff #(parameter size=5, delay=1)
(output reg [size-1:0] out,
 input      [size-1:0] in,
 input      clk);

    always @(posedge clk)
        #delay out = in;
endmodule

```

It shall be legal to instantiate modules using different types of parameter redefinition in the same top-level module. Consider the following example, where the parameters of **mod_a** are changed using positional parameter redefinition and only the second parameter of **mod_c** is changed using named parameter redefinition during instantiation.

```

module tb3;
// declarations & code ...

// legal mixture of instance with positional parameters and
// another instance with named parameters
vdff #(10, 15)      mod_a (.out(out_a), .in(in_a), .clk(clk));
vdff      mod_b (.out(out_b), .in(in_b), .clk(clk));
vdff #(.delay(12)) mod_c (.out(out_c), .in(in_c), .clk(clk));
endmodule

```

It shall be illegal to instantiate any module using a mixture of positional and named parameter redefinitions as shown in the instantiation of **mod_a** below.

```

module tb4;
// declarations & code ...

// mod_a instance with ILLEGAL mixture of both positional and
// named parameters
vdff #(10, .delay(15)) mod_a (.out(out_a), .in(in_a), .clk(clk));
vdff      mod_b (.out(out_b), .in(in_b), .clk(clk));
vdff #(.delay(12))      mod_c (.out(out_c), .in(in_c), .clk(clk));
endmodule

```