SystemVerilog Copying Objects

In a previous(/systemverilog/systemverilog-class-handle-object) post, key topics on class handles and objects were discussed which is essential to understand how shallow copy(#shallow-copy) and deep copy(#deep-copy) works.

Click here to refresh concepts in class handles and objects ! (/systemverilog/systemverilog-class-handle-object)

Shallow Copy

Contents in *pkt* will be copied into *pkt2* when *pkt* is used along with the `new()` constructor for the new object.

```
Packet pkt, pkt2;

pkt = new;
pkt2 = new pkt;
```

This method is known as a *shallow* copy, because all of the variables are copied across integers, strings, instance handles, etc but nested objects are not copied entirely. Only their handles will be assigned to the new object and hence both the packets will point to the same nested object instance. To illustrate this point let's look at an example.

```
class Header;
    int id;
    function new (int id);
        this.id = id;
    endfunction

    function showId();
        $display ("id=0x%0d", id);
    endfunction
endclass

class Packet;
    int     addr;
    int     data;
    Header  hdr;

    function new (int addr, int data, int id);
        hdr = new (id);
        this.addr = addr;
        this.data = data;
    endfunction

    function display (string name);
        $display ("[%s] addr=0x%0h data=0x%0h id=%
    endfunction
endclass

module tb;
    Packet p1, p2;
    initial begin
        // Create a new pkt object called p1
        p1 = new (32'hface_cafe, 32'h1234_5678, 2
        p1.display ("p1");

        // Shallow copy p1 into p2; p2 is a new ob
         p2 = new p1;
        p2.display ("p2");

        // Now let's change the addr and id in p1
        p1.addr = 32'habcd_ef12;
        p1.data = 32'h5a5a_5a5a;
        p1.hdr.id = 17;
        p1.display ("p1");

        // Print p2 and see that hdr.id points to
        // addr and data remain unchanged.
        p2.display ("p2");
    end
endmodule
```

The class **Packet** contains a nested class called **Header**. First we created a packet called *p1* and assigned it with some values. Then *p2* was created as a copy of *p1* using the shallow copy method. To prove that only handles and not the entire object is copied, members of the *p1* packet is

modified including members within the nested class. When the contents in *p2* are printed, we can see that the *id* member in the nested class remains the same.

- OUTPUT

```
ncsim> run
[p1] addr=0xfacecafe data=0x12345678 id=26
[p2] addr=0xfacecafe data=0x12345678 id=26

[p1] addr=0xabcdef12 data=0x5a5a5a5a id=17
[p2] addr=0xfacecafe data=0x12345678  id=17
ncsim: *W,RNQUIE: Simulation is complete.
```

Deep Copy

A *deep* copy is where everything (including nested objects) is copied and typically custom code is required for this purpose.

```
1  Packet p1 = new;
2  Packet p2 = new;
3  p2.copy (p1);
```

Let's add in a custom function called `copy()` within the **Packet** class to the example given above.

```
class Packet;
    ...
   function copy (Packet p);
      this.addr = p.addr;
      this.data = p.data;
      this.hdr.id = p.hdr.id;
   endfunction
    ...
endclass

module tb;
    Packet p1, p2;
    initial begin
        p1 = new (32'hface_cafe, 32'h1234_5678, 32
        p1.display ("p1");

        p2 = new (1,2,3);  // give some values
         p2.copy (p1);
        p2.display ("p2");

        // Now let's change the addr and id in p1
        p1.addr = 32'habcd_ef12;
        p1.data = 32'h5a5a_5a5a;
        p1.hdr.id = 32'h11;
        p1.display ("p1");

        // Now let's print p2 - you'll see the cha
        // but not addr
        p2.display ("p2");
    end
endmodule
```

Note that we have invoked the custom `copy()` function here instead of the *shallow* copy method, and hence Header object contents are expected to be copied into *p2* as well.

- **OUTPUT**

```
ncsim> run
[p1] addr=0xfacecafe data=0x12345678 id=26
[p2] addr=0xfacecafe data=0x12345678 id=26

[p1] addr=0xabcdef12 data=0x5a5a5a5a id=17
[p2] addr=0xfacecafe data=0x12345678  id=26
ncsim: *W,RNQUIE: Simulation is complete.
```

Note that *id* of object *p2* still holds onto the previous value even though *p1*'s *id* field was changed.

INTERVIEW QUESTIONS

- SystemVerilog Interview Set 1 (/systemverilog/systemverilog-interview-questions-set-1)
- SystemVerilog Interview Set 2 (/systemverilog/systemverilog-interview-questions-set-2)
- SystemVerilog Interview Set 3 (/systemverilog/systemverilog-interview-questions-set-3)
- SystemVerilog Interview Set 4 (/systemverilog/systemverilog-interview-questions-set-4)
- SystemVerilog Interview Set 5 (/systemverilog/systemverilog-interview-questions-set-5)
- SystemVerilog Interview Set 6 (/systemverilog/systemverilog-interview-questions-set-6)
- SystemVerilog Interview Set 7 (/systemverilog/systemverilog-interview-questions-set-7)
- SystemVerilog Interview Set 8 (/systemverilog/systemverilog-interview-questions-set-8)
- SystemVerilog Interview Set 9 (/systemverilog/systemverilog-interview-questions-set-9)
- SystemVerilog Interview Set 10 (/systemverilog/systemverilog-interview-questions-set-10)

POLL