OOP的句柄和cast

2025年9月21日 17:29

代码路径，使用EDA playground进行仿真

https://www.edaplayground.com/x/Sf6m

参考绿皮书的OOP章节进行学习

一、第一个小实验：tr=bad

1、申明两个class，其中有继承关系

```
52
53 class   Transaction extends uvm_object;
54     rand bit[31:0]  src;
55     virtual function void display(input string prefix="");
56       $display("%sTransaction: src=%0d", prefix, src);
57     endfunction
58
59   endclass
60
61   class BadTr extends Transaction;
62     int bad_crc;
63     virtual function void display(input string prefix="");
64       $display("%sTransaction: bad_crc=%0d", prefix, bad_crc);
65       super.display(prefix);
66     endfunction
67
68   endclass
69
```

2、申明两个句柄出来

```
75
76
77   Transaction   tr;
78   BadTr bad,bad2;
79
```

3、实例化

```
initial begin
  environment = new("env");

  //oop study====================
  //tr = new();
  bad = new();
  `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
  `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);

  bad.src = 32'h5555;
  bad.bad_crc = 32'hAAAA;

  `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
  `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);

  tr = bad;
  `uvm_info("FHZH_tr",$sformatf("tr.src=%0h",tr.src),UVM_NONE);
  //`uvm_info("FHZH_tr",$sformatf("tr.bad_crc=%0h",tr.bad_crc),UVM_NONE);
  tr.display;

  `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
  `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);
  bad.display;

  tr=new();
  `uvm_info("FHZH_tr",$sformatf("tr.src=%0h",tr.src),UVM_NONE);
  //`uvm_info("FHZH_tr",$sformatf("tr.bad_crc=%0h",tr.bad_crc),UVM_NONE);
  tr.display;


  //bad.display;
  //==============================

  run_test();
end
```

仿真结果：

```
# KERNEL: ASDB file was created in location /home/runner/dataset.asdb
# KERNEL: UVM_INFO /home/runner/testbench.sv(87) @ 0: reporter [FHZH_bad] bad.src=0
# KERNEL: UVM_INFO /home/runner/testbench.sv(88) @ 0: reporter [FHZH_bad] bad.bad_crc=0
# KERNEL: UVM_INFO /home/runner/testbench.sv(93) @ 0: reporter [FHZH_bad] bad.src=5555
# KERNEL: UVM_INFO /home/runner/testbench.sv(94) @ 0: reporter [FHZH_bad] bad.bad_crc=aaaa
# KERNEL: UVM_INFO /home/runner/testbench.sv(97) @ 0: reporter [FHZH_tr] tr.src=5555
# KERNEL: Transaction: bad_crc=aaaa
# KERNEL: Transaction: src=5555
# KERNEL: UVM_INFO /home/runner/testbench.sv(101) @ 0: reporter [FHZH_bad] bad.src=5555
# KERNEL: UVM_INFO /home/runner/testbench.sv(102) @ 0: reporter [FHZH_bad] bad.bad_crc=aaaa
# KERNEL: Transaction: bad_crc=aaaa
# KERNEL: Transaction: src=5555
# KERNEL: UVM_INFO /home/runner/testbench.sv(106) @ 0: reporter [FHZH_tr] tr.src=0
# KERNEL: Transaction: src=0
```

总结：

1、申明句柄，并不会开辟内存进行实例化，只有直接new时，才会有实例化空间；

2、当bad进行new时，bad有实例化空间；然后使用tr=bad，此时两个句柄都指向了同一个内存空间，但是tr的句柄只能访问到Transaction内的成员，如果想要打印出tr.bad_crc，就会报错，因为Transaction中没有bad_crc的member：

```
86      bad = new();
87      `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
88      `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);
89
90      bad.src = 32'h5555;
91      bad.bad_crc = 32'hAAAA;
92
93      `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
94      `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);
95
96      tr = bad;
97      `uvm_info("FHZH_tr",$sformatf("tr.src=%0h",tr.src),UVM_NONE);
98      `uvm_info("FHZH_tr",$sformatf("tr.bad_crc=%0h",tr.bad_crc),UVM_NONE);
99      tr.display;
100
101     `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
102     `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);
103     bad.display;
104
105     tr=new();
106     `uvm_info("FHZH_tr",$sformatf("tr.src=%0h",tr.src),UVM_NONE);
107     //`uvm_info("FHZH_tr",$sformatf("tr.bad_crc=%0h",tr.bad_crc),UVM_NONE);
108     tr.display;
```

```
[2025-09-21 09:36:44 UTC] vlib work && vlog '-timescale' '1ns/1ns' '-sv2k9' +incdir+$RIVIERA_HOME/vlib/uvm-1.2/src -l uvm_1_2 -err VC
VSIMSA: Configuration file changed: `/home/runner/library.cfg'
ALIB: Library "work" attached.
work = /home/runner/work/work.lib
MESSAGE_SP VCP2124 "Package uvm_pkg found in library uvm_1_2."
ERROR VCP5274 "Member ""bad_crc"" not found in ""tr"". Use ""-err VCP5274 W1"" to suppress this error." "testbench.sv" 98  166
FAILURE "Compile failure 1 Errors 0 Warnings  Analysis time: 2[s]."
Exit code expected: 0, received: 255
Done
```

3、但是若此时使用方法时，即tr.display，打印出来的是bad中重载过后的方法！

二、第二个小实验:bad=tr

```
84      //oop study=====================
85      //tr = new();
86      //第一个小实验：tr=bad
87      bad = new();
88      `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
89      `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);
90
91      bad.src = 32'h5555;
92      bad.bad_crc = 32'hAAAA;
93
94      `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
95      `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);
96
97      tr = bad;
98      `uvm_info("FHZH_tr",$sformatf("tr.src=%0h",tr.src),UVM_NONE);
99      //`uvm_info("FHZH_tr",$sformatf("tr.bad_crc=%0h",tr.bad_crc),UVM_NONE);
100     tr.display;
101
102     `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
103     `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);
104     bad.display;
105
106     tr=new();
107     `uvm_info("FHZH_tr",$sformatf("tr.src=%0h",tr.src),UVM_NONE);
108     //`uvm_info("FHZH_tr",$sformatf("tr.bad_crc=%0h",tr.bad_crc),UVM_NONE);
109     tr.display;
110
111     //第二个小实验：bad=tr;
112     tr.src = 32'h7777;
113     `uvm_info("FHZH_tr",$sformatf("tr.src=%0h",tr.src),UVM_NONE);
114     //`uvm_info("FHZH_tr",$sformatf("tr.bad_crc=%0h",tr.bad_crc),UVM_NONE);
115     tr.display;
116
117     bad.display;
118     bad=tr;
119     `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
120     `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);
121
```

仿真结果

```
[2025-09-21 09:44:29 UTC] vlib work && vlog '-timescale' '1ns/1ns' '-sv2k9' +incdir+$RIVIERA_HOME/vlib/uvm-1
VSIMSA: Configuration file changed: `/home/runner/library.cfg'
ALIB: Library "work" attached.
work = /home/runner/work/work.lib
MESSAGE_SP VCP2124 "Package uvm_pkg found in library uvm_1_2."
WARNING VCP5228 "Input port a<wire> is used as lvalue." "design.sv" 33  16
WARNING VCP5228 "Input port b<wire> is used as lvalue." "design.sv" 34  16
WARNING VCP5228 "Input port doAdd<wire> is used as lvalue." "design.sv" 35  20
ERROR VCP2852 "Incompatible types at assignment: .bad<BadTr> <- tr<Transaction>." "testbench.sv" 118  8
FAILURE "Compile failure 1 Errors 3 Warnings  Analysis time: 2[s]."
Exit code expected: 0, received: 255
Done
```

总结：结果符合书中描述，如果将基类的实例赋给子类的句柄，编译直接报错

但是如例 8.13 所示，当你试图做反方向的赋值，即将一个基类对象拷贝到一个扩展

226 第8章 面向对象编程的高级技巧指南

类的句柄中时，会发生什么呢？这种操作会失败，因为有些属性仅存在于扩展类中，基类并不具备，例如 bad_crc。SystemVerilog 编译器对句柄类型作静态检查，因此第二行不会被编译。

例 8.13 将一个基类句柄拷贝到一个扩展类句柄

```
tr=new();                  // 创建一个基类对象
bad=tr;                    // ERROR:这一行不会被编译
$display(bad.bad_crc);     // 基类对象不存在 bad_crc 成员
```

三、第三个小实验：cast

1、class类的申明和句柄的申明别无两样，但需要注意这个bad2，这次会用到

```
52
53 class   Transaction extends uvm_object;
54     rand bit[31:0]  src;
55     virtual function void display(input string prefix="");
56       $display("%sTransaction: src=%0h", prefix, src);
57     endfunction
58
59   endclass
60
61   class BadTr extends Transaction;
62     int bad_crc;
63     virtual function void display(input string prefix="");
64       $display("%sTransaction: bad_crc=%0h", prefix, bad_crc);
65       super.display(prefix);
66     endfunction
67
68   endclass
69
70
71 module top;
72
73   bit clk;
74   env environment;
75
76
77   Transaction   tr;
78   BadTr bad,bad2;
79
```

2、这边虽然中间将tr改来改去，最后仍然将tr句柄挂在了bad上， bad自始至终没变过哈，但是bad2甚至没有申明实例化，就直接将tr作为源端，bad2作为目的端进行了cast操作

```
initial begin
  environment = new("env");

  //oop study====================
  //tr = new();
  //第一个小实验：tr=bad
  bad = new();
  `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
  `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);

  bad.src = 32'h5555;
  bad.bad_crc = 32'hAAAA;

  `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
  `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);

  tr = bad;
  `uvm_info("FHZH_tr",$sformatf("tr.src=%0h",tr.src),UVM_NONE);
  //`uvm_info("FHZH_tr",$sformatf("tr.bad_crc=%0h",tr.bad_crc),UVM_NONE);
  tr.display;

  `uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
  `uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);
  bad.display;

  tr=new();
  `uvm_info("FHZH_tr",$sformatf("tr.src=%0h",tr.src),UVM_NONE);
  //`uvm_info("FHZH_tr",$sformatf("tr.bad_crc=%0h",tr.bad_crc),UVM_NONE);
  tr.display;

  //第二个小实验：bad=tr;
  tr.src = 32'h7777;
  `uvm_info("FHZH_tr",$sformatf("tr.src=%0h",tr.src),UVM_NONE);
  //`uvm_info("FHZH_tr",$sformatf("tr.bad_crc=%0h",tr.bad_crc),UVM_NONE);
  tr.display;

  bad.display;
  //bad=tr;
  //`uvm_info("FHZH_bad",$sformatf("bad.src=%0h",bad.src),UVM_NONE);
  //`uvm_info("FHZH_bad",$sformatf("bad.bad_crc=%0h",bad.bad_crc),UVM_NONE);

  //第三个小实验：cast的使用
  tr=bad;
  tr.display;
  $cast(bad2,tr);
  `uvm_info("FHZH_bad2",$sformatf("bad2.src=%0h",bad2.src),UVM_NONE);
  `uvm_info("FHZH_bad2",$sformatf("bad2.bad_crc=%0h",bad2.bad_crc),UVM_NONE);
  bad2.display;
```

3、仿真结果：可以看到bad2拥有了值，且和bad一样，因为他两就是同一类型的，所以cast成功

```
# KERNEL: ASDB file was created in location /home/runner/dataset.asdb
# KERNEL: UVM_INFO /home/runner/testbench.sv(88) @ 0: reporter [FHZH_bad] bad.src=0
# KERNEL: UVM_INFO /home/runner/testbench.sv(89) @ 0: reporter [FHZH_bad] bad.bad_crc=0
# KERNEL: UVM_INFO /home/runner/testbench.sv(94) @ 0: reporter [FHZH_bad] bad.src=5555
# KERNEL: UVM_INFO /home/runner/testbench.sv(95) @ 0: reporter [FHZH_bad] bad.bad_crc=aaaa
# KERNEL: UVM_INFO /home/runner/testbench.sv(98) @ 0: reporter [FHZH_tr] tr.src=5555
# KERNEL: Transaction: bad_crc=aaaa
# KERNEL: Transaction: src=5555
# KERNEL: UVM_INFO /home/runner/testbench.sv(102) @ 0: reporter [FHZH_bad] bad.src=5555
# KERNEL: UVM_INFO /home/runner/testbench.sv(103) @ 0: reporter [FHZH_bad] bad.bad_crc=aaaa
# KERNEL: Transaction: bad_crc=aaaa
# KERNEL: Transaction: src=5555
# KERNEL: UVM_INFO /home/runner/testbench.sv(107) @ 0: reporter [FHZH_tr] tr.src=0
# KERNEL: Transaction: src=0
# KERNEL: UVM_INFO /home/runner/testbench.sv(113) @ 0: reporter [FHZH_tr] tr.src=7777
# KERNEL: Transaction: src=7777
# KERNEL: Transaction: bad_crc=aaaa
# KERNEL: Transaction: src=5555
# KERNEL: Transaction: bad_crc=aaaa
# KERNEL: Transaction: src=5555
# KERNEL: UVM_INFO /home/runner/testbench.sv(126) @ 0: reporter [FHZH_bad2] bad2.src=5555
# KERNEL: UVM_INFO /home/runner/testbench.sv(127) @ 0: reporter [FHZH_bad2] bad2.bad_crc=aaaa
# KERNEL: Transaction: bad_crc=aaaa
# KERNEL: Transaction: src=5555
# KERNEL: UVM_INFO @ 0: reporter [RNTST] Running test ...
```

4、总结：结合书本的描述，可以看到在cast的过程中tr就是个中间工具人：

4.1、tr确实指向了一个对象实例，即bad的实例，但结合小实验一可以看到，tr只能访问当bad实例中的一部分（src），如果访问bad_crc就不得行；

4.2、当进行cast操作是，其实检查的就是tr所指向的实例化本身的对象类型和目的端的对象类型，此处两个类型为一致的；

4.3、畅享：根据描述，如果tr所指向的对象类型bad是bad2类型的拓展类，那应该也能成功，只不过cast过去的只有部分了，因为bad2只能访问自己类型的成员。（此处就没有做试验了，因为用的少吧）

4.4、工作中遇到的cast例子有一个就是sequence和driver之间的握手了，在sqr中即将将包传递给driver时，会将内部的包，就像此处的bad，送给一个uvm_object基类tr，然后将tr传递给driver；而在driver侧收到tr时，我们可以自己申明一个想要的包的句柄，比如此处是bad2，然后$(bad2,tr)，这样sqr和driver之间只需要传递基类tr即可，不需要传递其他类型的包。

将一个基类句柄赋值给一个扩展类句柄并不总是非法的。当基类句柄确实指向一个派生类对象时是允许的。$cast 子程序会检查句柄所指向的对象类型，而不仅仅检查句柄本身。一旦源对象跟目的对象是同一类型，或者是目的类的扩展类，你就可以从基类句柄 tr 中拷贝扩展对象的地址给扩展对象的句柄 bad2 了。

例 8.14 使用$cast 拷贝句柄

```
bad=new();                    // 构建 BadTr 扩展对象
tr=bad;                       // 基类句柄指向扩展对象

// 检查对象类型并且拷贝。如果类型失配则在仿真时报错
//如果成功，bad2 就指向 tr 所引用的对象
$cast(bad2,tr);
```