# 线程同步,通信与虚方法

# 目录

- 线程同步,通信与虚方法
  - 。 进程同步,通信
    - 事件event
    - 旗语semaphore
    - 信箱mailbox
  - 。 虚方法
    - 实例理解
      - 将子类句柄赋值成父类句柄
      - 将父类句柄赋值成子类句柄
      - 使用系统函数\$cast()
    - 结论

# 进程同步,通信

测试平台的的线程之间需要进行同步与数据交换,所有的数据交换被称为线程间的通信

# 事件event

- 使用关键词声明一个事件,不需要new;
- 使用 -> 触发一个事件;
- 使用wait (电平敏感) 或者是@ (边沿敏感) 来进行等待,一般先等待事件,再触发事件。

实例如下: event创建了两个对象,不需要new。触发e1时等待e2,触发e2时等待e1。由于都是同时触发,可能存在等不到的情况。

```
event e1,e2;
  initial begin
    $display("@%0t: 1:before trigger", $time);
    -> e1;
    @e2;
    $display("@%0t: 1:after trigger", $time);
end
initial begin
    $display("@%0t: 2:before trigger", $time);
    ->e2;
    @e1;
    $display("@%0t: 2:after trigger", $time);
end
```

#### 打印结果为

```
@0:1: before trigger
@0:2: before trigger
@0:1: after trigger
```

增加triggered, 使用wait进行等待。

```
event e1,e2;
  initial begin
        $display("@%0t: 1:before trigger", $time);
        -> e1;
        wait (e2.tregger);
        $display("@%0t: 1:after trigger", $time);
end
  initial begin
        $display("@%0t: 2:before trigger", $time);
        ->e2;
        wait (e1.tregger);
        $display("@%0t: 2:after trigger", $time);
end
```

此时输出观察结果。两个事件均被触发

```
@0:1: before trigger
@0:2: before trigger
@0:1: after trigger
@0:2: after trigger
```

# 旗语semaphore

实现对同一个资源的访问控制功能。

想象一下你和你爱人共享一辆汽车的情形。显然,每次只能有一个人可以开车。为应对这种情况,你们可以约定谁持有钥匙谁开车。当你用完车以后,你会让出车子以便对方使用。车钥匙就是旗语,它确保了只有一个人可以使用汽车。在操作系统的术语里,这就是大家所熟知的"互斥访问",所以旗语可被视为一个互斥体,用于实现对同一资源的访问控制。

- 使用关键词semaphore 声明
- 使用前必需要使用new()进行初始化。new(1)为放入一把钥匙
- get()和put()可以对钥匙进行获取或是归还,进行等待
- try\_get()获取一个旗语不被阻塞。返回0表示要是不够,不进行等待

semaphore初始化可以初始化0,即new()无参数,可以不停的换钥匙。 semaphore中get()/put()函数中没有传递参数,即 默认他们在等待和归还钥匙的数量为1

# 信箱mailbox

sv里的FIFO, 线程内部数据通信或者是数据的缓存

- 必须使用new进行初始化,使用size来限定存储的最大数量。function new (int bound =0);表示不限制大小
- 使用put放入数据, get获取数据
- 信箱满,那么put阻塞;信箱空,get会阻塞
- peek拷贝数据不移除信箱里的数据
- num()获取信息的数目
- 默认信箱没有存储类型。可以使用#()来指定存储的形式

# 虚方法

使用虚方法目的:通过在父类里定义虚方法(task or function),可以在当父类句柄调用一个方法时候,前提是若是这个句柄指向了子类对象,则调用的方法为子类的方法而不是父类的方法。

# 实例理解

#### 将子类句柄赋值成父类句柄

```
module tb virtual();
class Transaction;
bit [31:0] src = 100;
function void display();
   $display("Transaction src = %0d", src);
endfunction
endclass
class BadTr extends Transaction;
bit [31:0] bad src = 200;
function void display();
   super.display();
   $display("BadTr src = %0d",bad_src);
endfunction
endclass
initial begin
   Transaction tr;
   BadTr bad, bad2;
   bad = new(); //构建了一个子类的对象
   tr = bad; //父类的句柄指向子类的对象
   $display(tr.src);//显示父类的变量的内容
   tr.display; //调用的是父类的函数
end
endmodule
```

### 仿真结果为

```
# 100
# Transaction src = 100
```

### 使用虚方法后可以看到打印的输出发生了变化。调用了子类的函数。

```
Library × Project × sim ×
                                                    module tb virtual();
                                                   □class Transaction;
Transcript =
                                                    bit [31:0] src = 100;
# ** Warning: (vlib-34) Library already exists at "w
                                                   □virtual function void display();//使用虚方法定义函数
# QuestaSim-64 vmap 10.6c Lib Mapping Utility 2017.0
                                                         $display("Transaction src = %0d", src);
# vmap work work
# Modifying E:/code/QuestaSim/my_testbeach/tb.mpf
                                                    endfunction
# QuestaSim-64 vlog 10.6c Compiler 2017.07 Jul 26
                                                    endclass
# Start time: 17:59:09 on Dec 03,2021
# vlog -reportprogress 300 tb_virtual.sv
                                                9
                                                  □class BadTr extends Transaction;
# -- Compiling module tb_virtual
                                                  bit [31:0] bad src = 200;
# Top level modules:
                                                  function void display();
      tb_virtual
                                                         super.display();
# End time: 17:59:09 on Dec 03,2021, Elapsed time: 0
                                                         $display("BadTr src = %0d",bad src);
# Errors: 0, Warnings: 0
                                                   endfunction
                                               14
# End time: 17:59:12 on Dec 03,2021, Elapsed time: 0
                                                    endclass
# Errors: 0, Warnings: 1
vsim -t ns -novopt "+notimingchecks" work.tb_virtu
                                              16
# Start time: 17:59:12 on Dec 03,2021
                                                  ∃initial begin
                                               17
 ** Warning: (vsim-8891) All optimizations are turn
                                                    Transaction tr;
e the User's Manual section on Preserving Object Vis
                                                    BadTr bad, bad2;
# Refreshing E:/code/QuestaSim/my_testbeach/work.tb_
                                                   bad = new(); //构建了一个子类的对象
                                              20
# Loading sv std.std
                                                    tr = bad; //父类的句柄指向子类的对象
# Loading work.tb virtual
                                               21
       100
                                                    $display(tr.src);//显示父类的变量的内容
  Transaction src
                                                    tr.display; //调用的是父类的函数
# BadTr src = 200
                                               24
                                                   end
                                                    endmodule
VSIM 2>
```

#### 将父类句柄赋值成子类句柄

```
initial begin
Transaction tr;
BadTr bad,bad2;
tr = new();//创建一个父类对象
bad = tr;//将父类对象赋值给子类句柄,ERROR不会执行
```

```
$display(bad.bad_src);//父类对象不存在该成员变量
end
```

# 我的编译环境会报错

```
# ** Error: (vlog-13216) tb_virtual.sv(21): Illegal assignment to type 'class tb_virtual.BadTr' from type 'cla
```

#### 使用系统函数\$cast()

```
initial begin
    Transaction tr;
BadTr bad,bad2;
bad = new();
tr = bad; //父类的句柄指向子类的对象
    $cast(bad2,tr);
    if(!$cast(bad2,tr))
        $display("cannot assign tr to bad2");
$display(bad2.bad_src);
bad2.display();
end
```

#### 打印结果为

```
# 200
# Transaction src = 100
# BadTr src = 200
```

# 结论

- 通过在父类里定义虚方法(task or function),可以在当父类句柄调用一个方法时候,**前提是若是这个句柄指向了子 类对象**,则调用的方法为子类的方法而不是父类的方法。
- 将父类对象赋值给子类句柄,ERROR不会执行
- 父类的句柄指向子类的对象,但是不能访问子类成员,使用虚方法可以访问子类的函数或者任务
- 使用系统函数\$cast()进行类型转换,转换之后的新句柄可以访问函数与变量