



БУДУЩЕЕ  
В НАШИХ  
РУКАХ

# Как разработать библиотеку верификационных компонентов и сохранить рассудок

Андрей Кущенко



# Андрей Куценко

Руководитель группы модульной верификации



## План доклада

**01** Требования для качественной библиотеки переиспользуемых компонентов

**02** Unit-тесты

**03** Быстрый старт с SVUnit

**04** Структура тестового окружения

**05** Тестирование UVM агента

**06** Тестирование негативных сценариев

**07** Wavedrom для написания тестов

**08** Выводы



## Обычная библиотека

Хотим библиотеку верификационных компонентов (VIP?)

Пишем код для пары десятков компонентов

Отлаживаем вместе с RTL (много-много раз)

Начинаем верифицировать новый модуль и в очередной раз  
копипастим reset агент с исправлением 3 строк кода

<Вы находитесь здесь>



# Обычная библиотека

## Хотим библиотеку верификационных компонентов (VIP?)

Пишем код для пары десятков компонентов

Отлаживаем вместе с RTL (много-много раз)

Начинаем верифицировать новый модуль и в очередной раз  
копипастим reset агент с исправлением 3 строк кода

<Вы находитесь здесь>



# Обычная библиотека

Хотим библиотеку верификационных компонентов (VIP?)

## Пишем код для пары десятков компонентов

---

Отлаживаем вместе с RTL (много-много раз)

Начинаем верифицировать новый модуль и в очередной раз копируем reset агент с исправлением 3 строк кода

<Вы находитесь здесь>



## Обычная библиотека

Хотим библиотеку верификационных компонентов (VIP?)

Пишем код для пары десятков компонентов

### Отлаживаем вместе с RTL (много-много раз)

---

Начинаем верифицировать новый модуль и в очередной раз копируем reset агент с исправлением 3 строк кода

<Вы находитесь здесь>



## Обычная библиотека

Хотим библиотеку верификационных компонентов (VIP?)

Пишем код для пары десятков компонентов

Отлаживаем вместе с RTL (много-много раз)

**Начинаем верифицировать новый модуль и в очередной раз  
копипастим `reset` агент с исправлением 3 строк кода**

---

<Вы находитесь здесь>



## Обычная библиотека

Хотим библиотеку верификационных компонентов (VIP?)

Пишем код для пары десятков компонентов

Отлаживаем вместе с RTL (много-много раз)

Начинаем верифицировать новый модуль и в очередной раз  
копипастим reset агент с исправлением 3 строк кода

**<Вы находитесь здесь>**

---



# Библиотека у сына маминей подруги

Документация на каждый агент, хранящаяся в общем доступе

Непосредственно код компонента

Тесты на компонент

Lint проверки

CI



# Библиотека у сына маминей подруги

Документация на каждый агент, хранящаяся в общем доступе

Непосредственно код компонента

Тесты на компонент

Lint проверки

CI



# Библиотека у сына маминей подружки

Документация на каждый агент, хранящаяся в общем доступе

## Непосредственно код компонента

---

Тесты на компонент

Lint проверки

CI

---



# Библиотека у сына маминей подружки

Документация на каждый агент, хранящаяся в общем доступе

Непосредственно код компонента

## Тесты на компонент

---

Lint проверки

CI

---



# Библиотека у сына маминой подруги

Документация на каждый агент, хранящаяся в общем доступе

Непосредственно код компонента

Тесты на компонент

**Lint проверки**

---

CI

---



# Библиотека у сына маминей подруги

Документация на каждый агент, хранящаяся в общем доступе

Непосредственно код компонента

Тесты на компонент

Lint проверки

CI

---



# Библиотека у сына маминей подруги

Документация на каждый агент, хранящаяся в общем доступе

Непосредственно код компонента

## Тесты на компонент

---

Lint проверки

CI

---



# SVUnit



Open-source библиотека  
для написания тестов  
на языке **SystemVerilog**





# Основные возможности SVUnit

**01**

Скрипты  
для запуска

**02**

Набор удобных  
макросов

**03**

Красивые принты

**04**

Встроенная логика для  
запуска, остановки теста

**05**

Поддержка wavedrom  
для написания тестов



# Поддерживаемые симуляторы



IUS



Xcellium



Questa



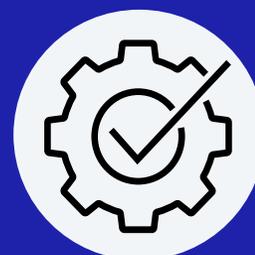
Modelsim



Riviera



VCS



dsim



Verilator



## Быстрый старт с SVUnit

```
> git clone https://github.com/svunit/svunit.git  
  
> export SVUNIT_INSTALL=`pwd`  
  
> export PATH=$PATH:$SVUNIT_INSTALL"/bin"  
  
> create_unit_test.pl bogus.sv  
  
> runSVUnit -s <simulator>
```



## Быстрый старт с SVUnit

```
> git clone https://github.com/svunit/svunit.git  
> export SVUNIT_INSTALL=`pwd`  
> export PATH=$PATH:$SVUNIT_INSTALL"/bin"  
> create_unit_test.pl bogus.sv  
> runSVUnit -s <simulator>
```



## Быстрый старт с SVUnit

```
> git clone https://github.com/svunit/svunit.git  
> export SVUNIT_INSTALL=`pwd`  
> export PATH=$PATH:$SVUNIT_INSTALL"/bin"  
> create_unit_test.pl bogus.sv  
> runSVUnit -s <simulator>
```

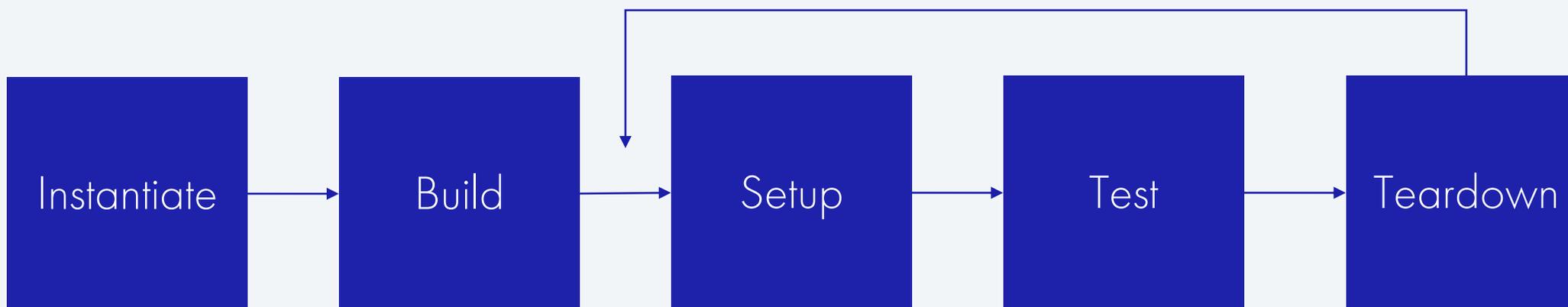


## Быстрый старт с SVUnit

```
> git clone https://github.com/svunit/svunit.git  
> export SVUNIT_INSTALL=`pwd`  
> export PATH=$PATH:$SVUNIT_INSTALL"/bin"  
> create_unit_test.pl bogus.sv  
> runSVUnit -s <simulator>
```



# Структура тестового окружения





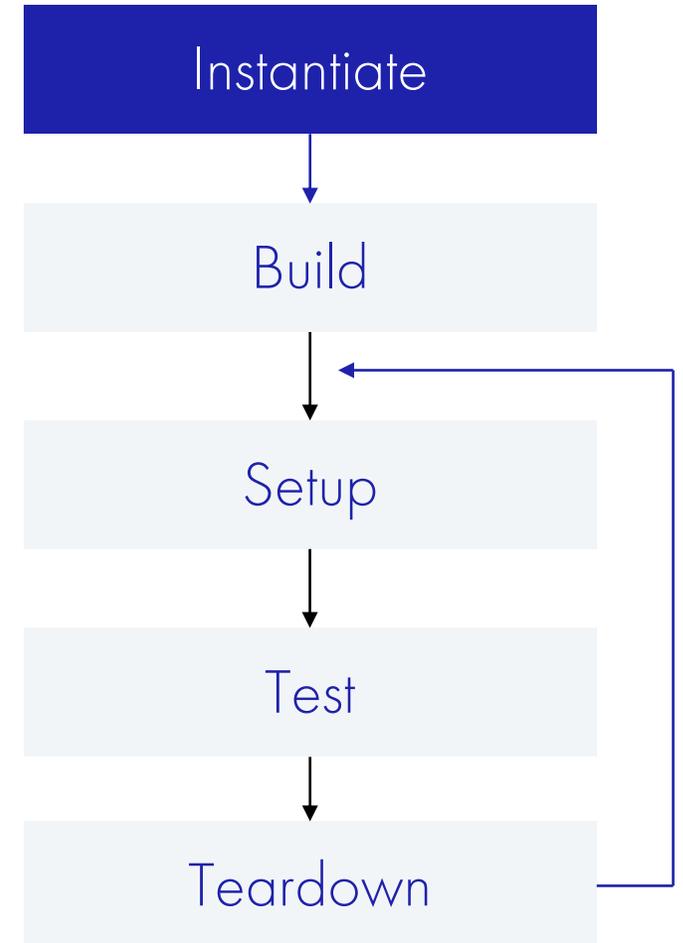
# Instantiate

```
import svunit_pkg::svunit_testcase;

string name = "apb_slave_ut";
svunit_testcase svunit_ut;

initial begin
    clk = 0;
    forever
        #5 clk = ~clk;
end

apb_slave my_apb_slave(.*);
```





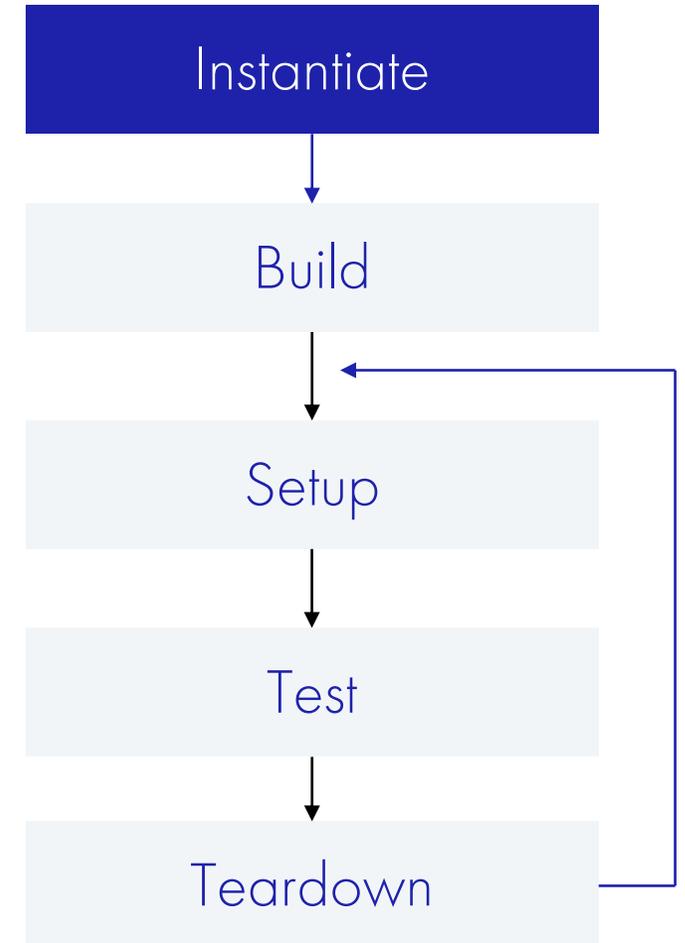
# Instantiate

```
import svunit_pkg::svunit_testcase;

string name = "apb_slave_ut";
svunit_testcase svunit_ut;

initial begin
    clk = 0;
    forever
        #5 clk = ~clk;
end

apb_slave my_apb_slave(.*);
```





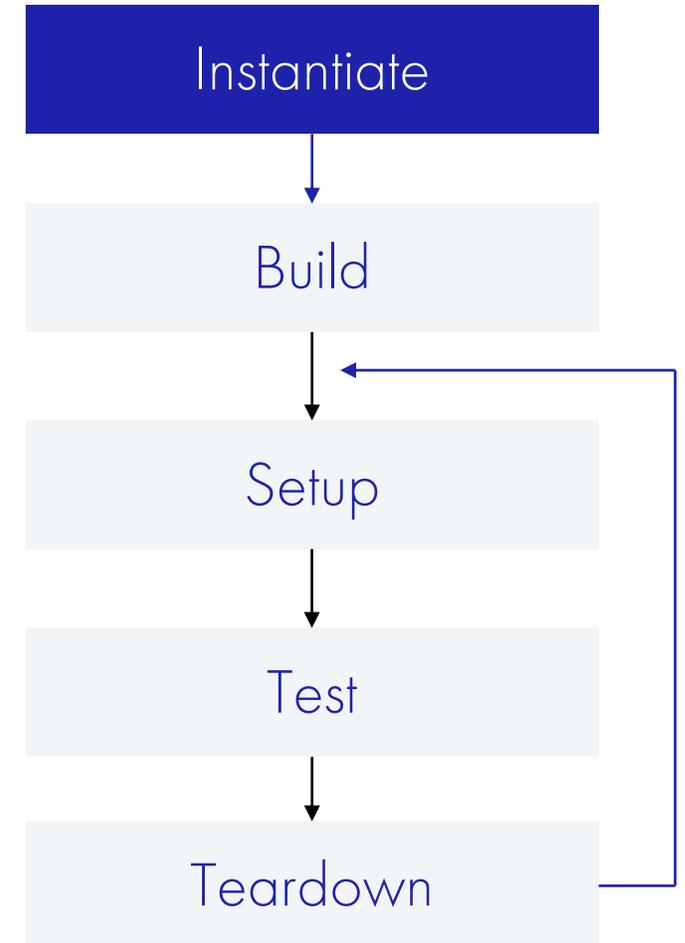
# Instantiate

```
import svunit_pkg::svunit_testcase;

string name = "apb_slave_ut";
svunit_testcase svunit_ut;

initial begin
    clk = 0;
    forever
        #5 clk = ~clk;
end

apb_slave my_apb_slave(.*);
```





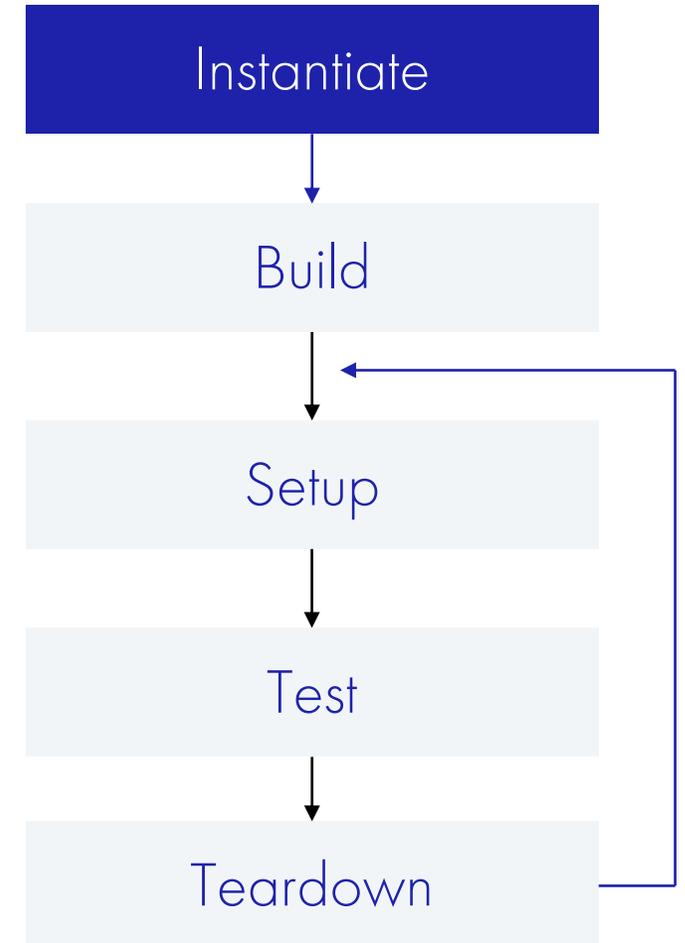
# Instantiate

```
import svunit_pkg::svunit_testcase;

string name = "apb_slave_ut";
svunit_testcase svunit_ut;

initial begin
    clk = 0;
    forever
        #5 clk = ~clk;
end

apb_slave my_apb_slave(.*);
```





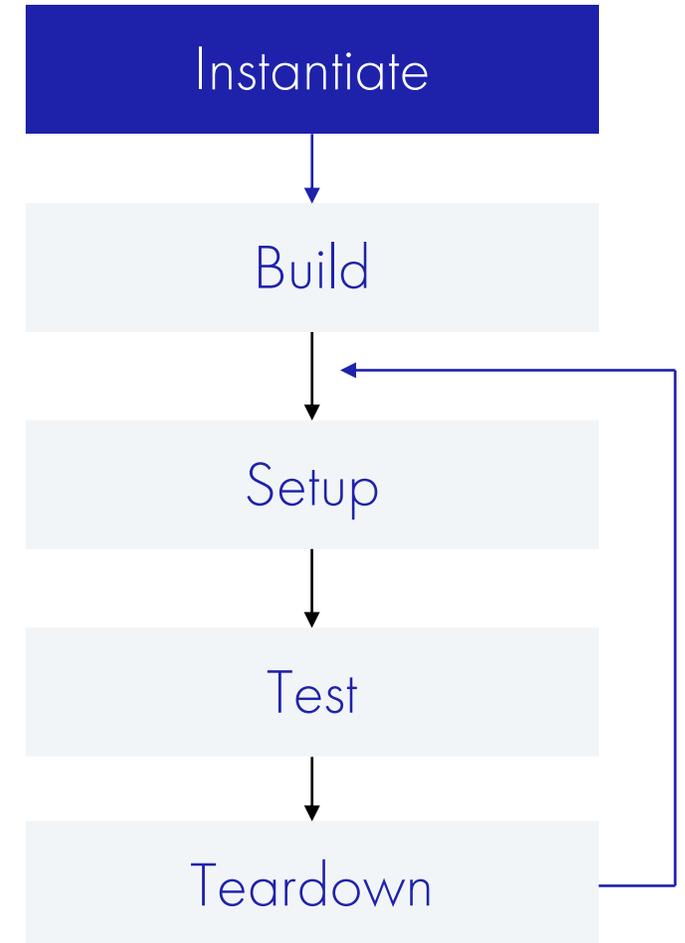
# Instantiate

```
import svunit_pkg::svunit_testcase;

string name = "apb_slave_ut";
svunit_testcase svunit_ut;

initial begin
    clk = 0;
    forever
        #5 clk = ~clk;
end

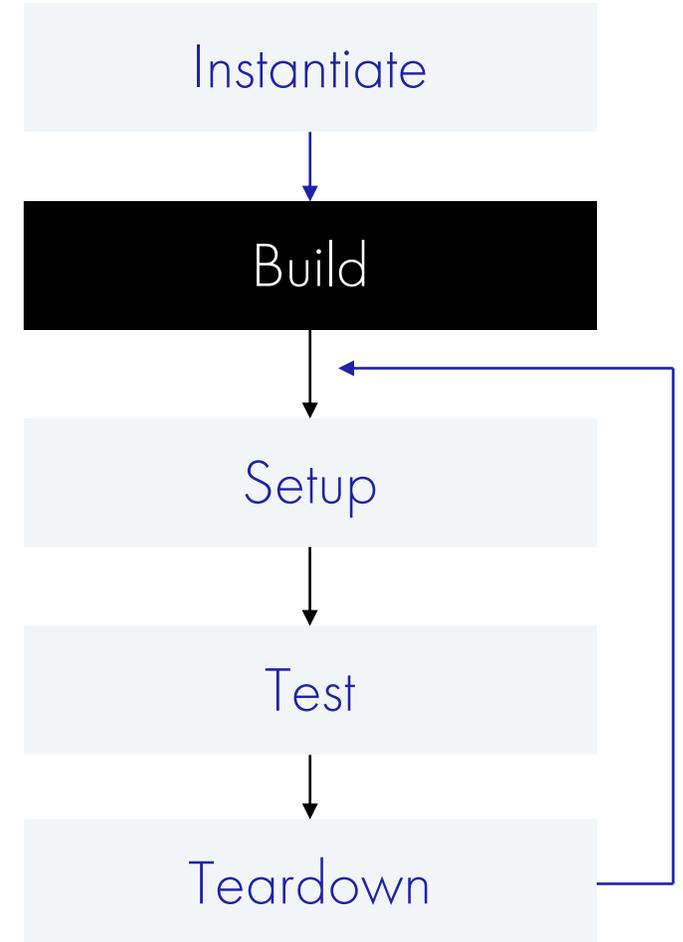
apb_slave my_apb_slave(.*);
```





# Build

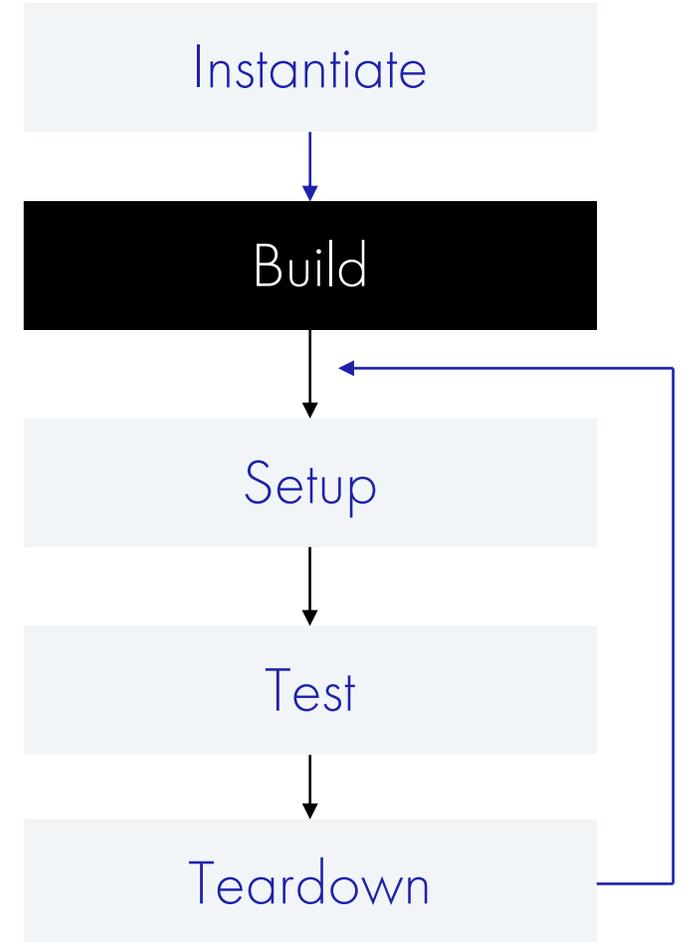
```
function void build();  
    svunit_ut = new(name);  
  
    checker = new();  
endfunction
```





# Build

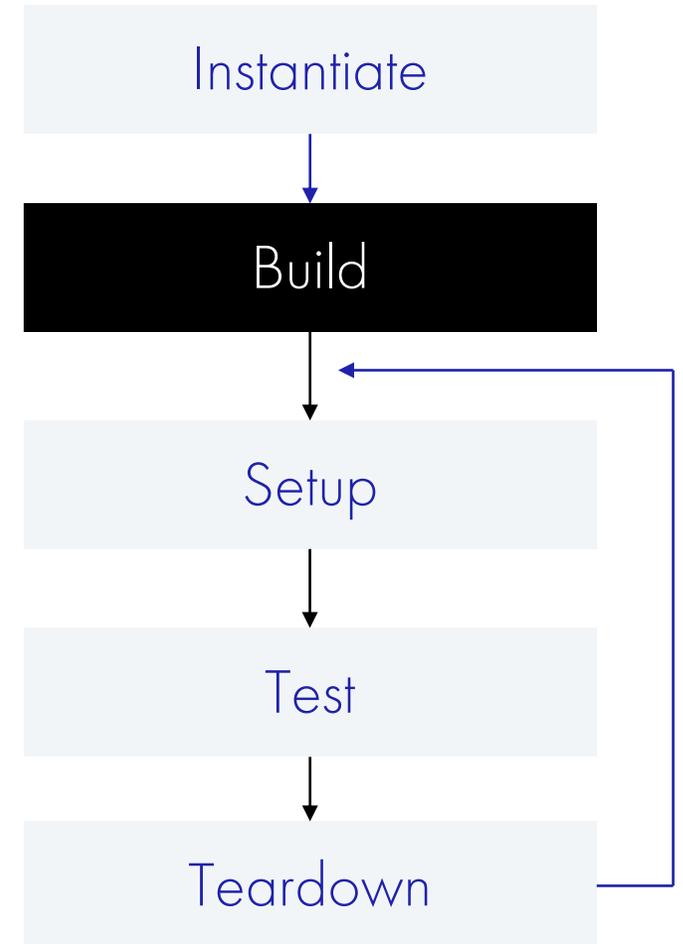
```
function void build();  
    svunit_ut = new(name);  
  
    checker = new();  
endfunction
```





# Build

```
function void build();  
    svunit_ut = new(name);  
  
    checker = new();  
endfunction
```



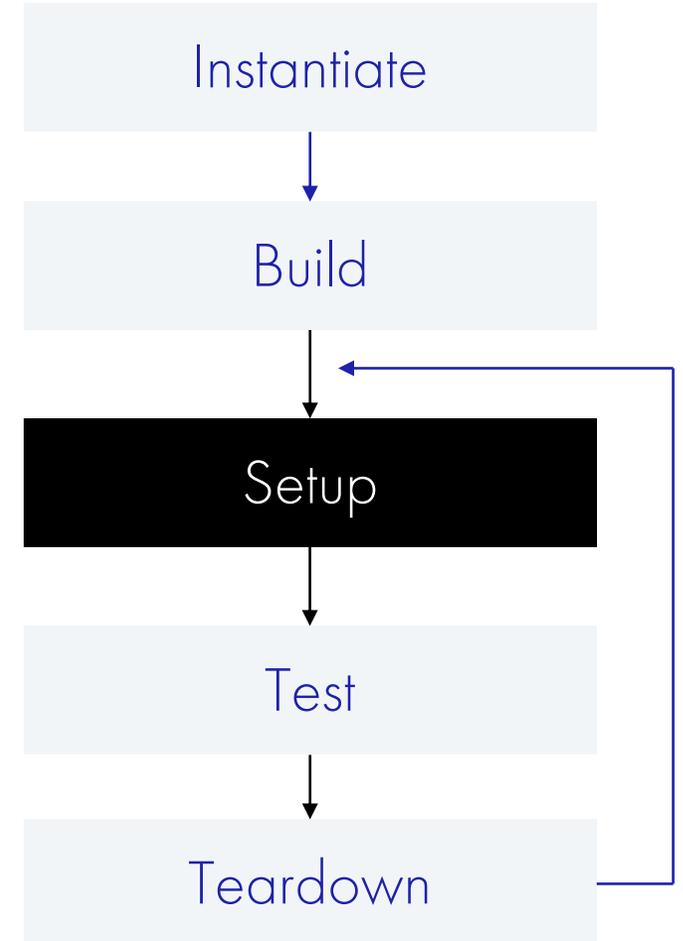


# Setup

```
task setup();
    svunit_ut.setup();

    idle();

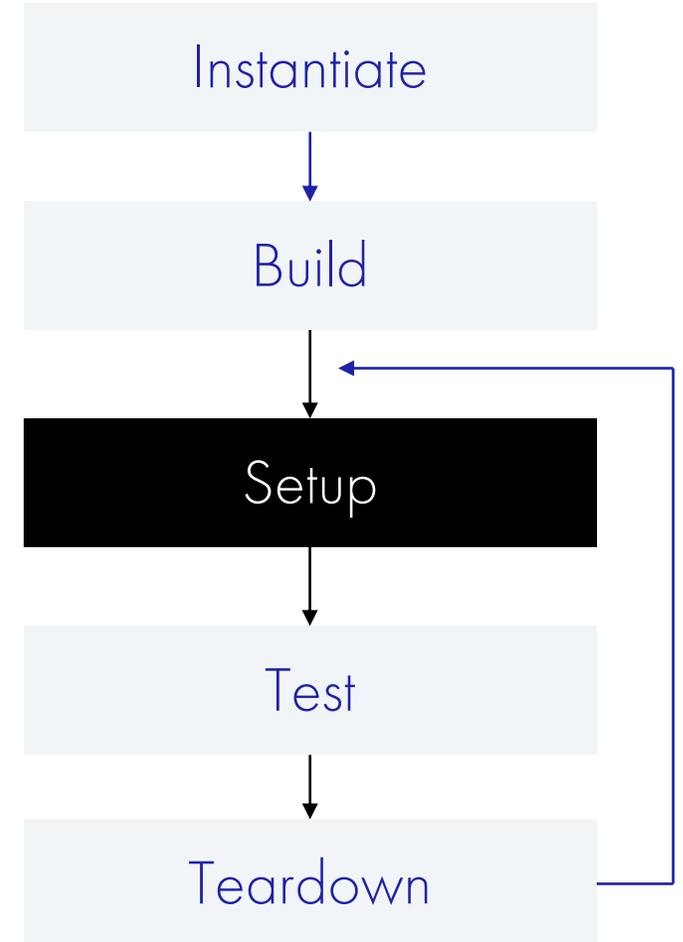
    rst_n = 0;
    repeat (8) @(posedge clk);
    rst_n = 1;
endtask
```





# Setup

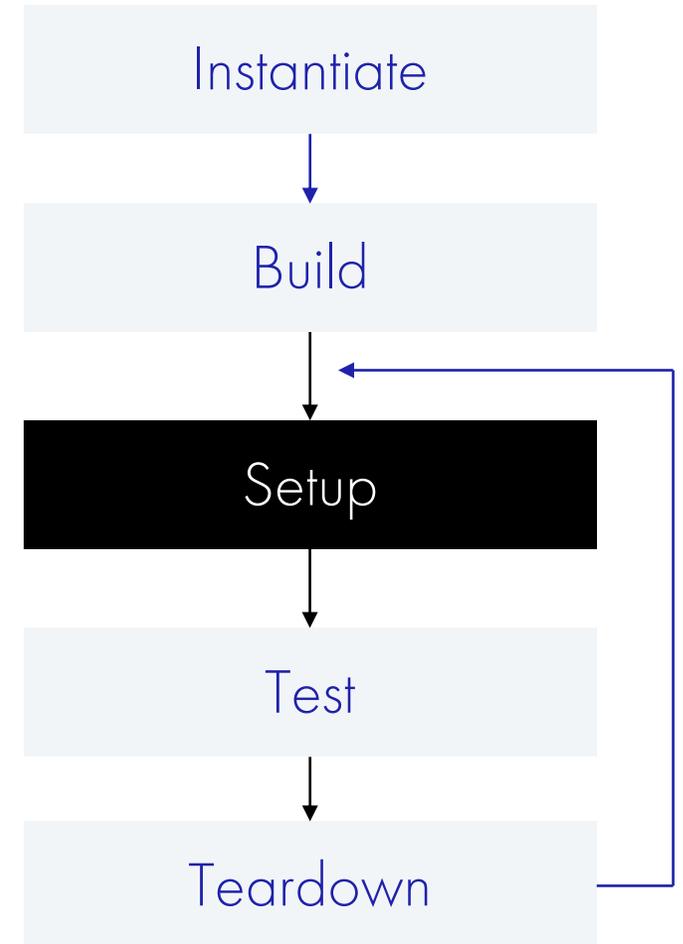
```
task setup();  
    svunit_ut.setup();  
  
    idle();  
  
    rst_n = 0;  
    repeat (8) @(posedge clk);  
    rst_n = 1;  
endtask
```





# Setup

```
task setup();  
    svunit_ut.setup();  
  
    idle();  
  
    rst_n = 0;  
    repeat (8) @(posedge clk);  
    rst_n = 1;  
endtask
```



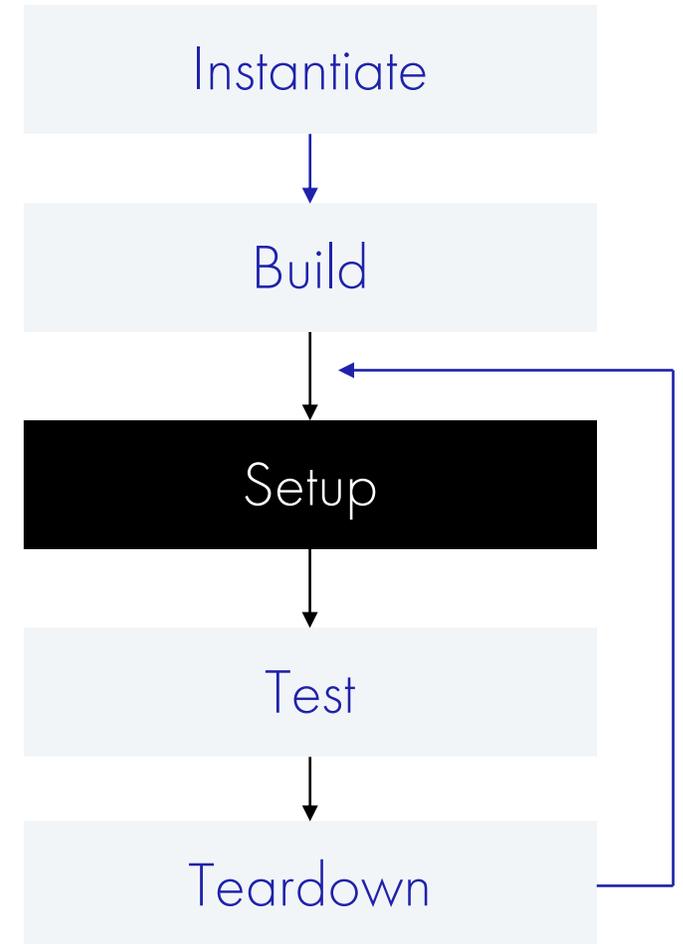


# Setup

```
task setup();
    svunit_ut.setup();

    idle();

    rst_n = 0;
    repeat (8) @(posedge clk);
    rst_n = 1;
endtask
```

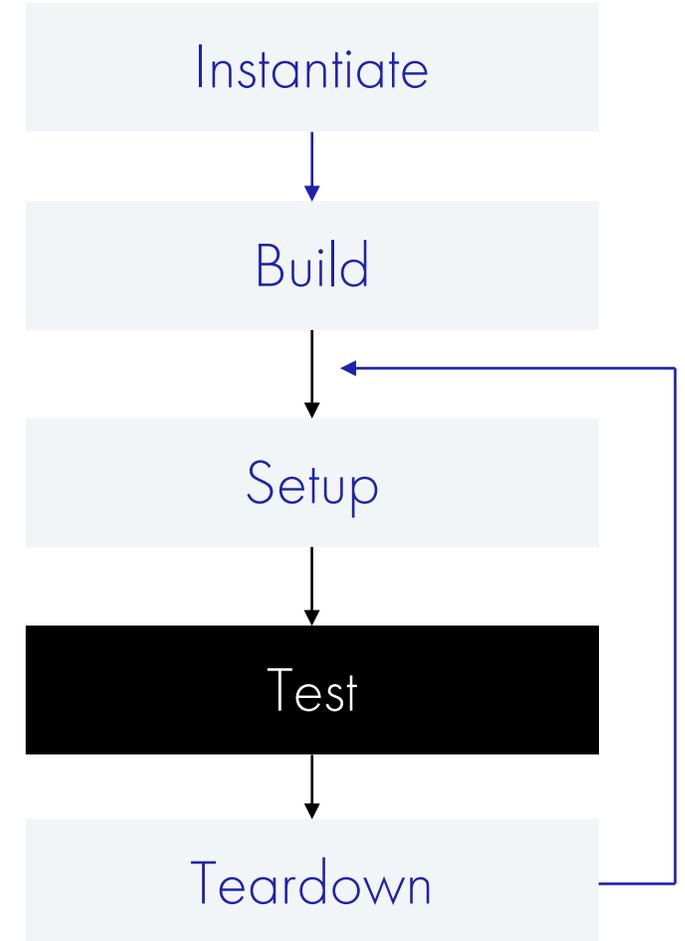




# Test

```
/**
 * *****
 * // Test:
 * //   single_write_then_read
 * //
 * // Desc:
 * //   do a write then a read at the same address
 * // *****
 `SVTEST(single_write_then_read)
   addr = 'h32;
   data = 'h61;

   write(addr, data);
   read(addr, rdata);
   `FAIL_IF(data != rdata);
 `SVTEST_END
```

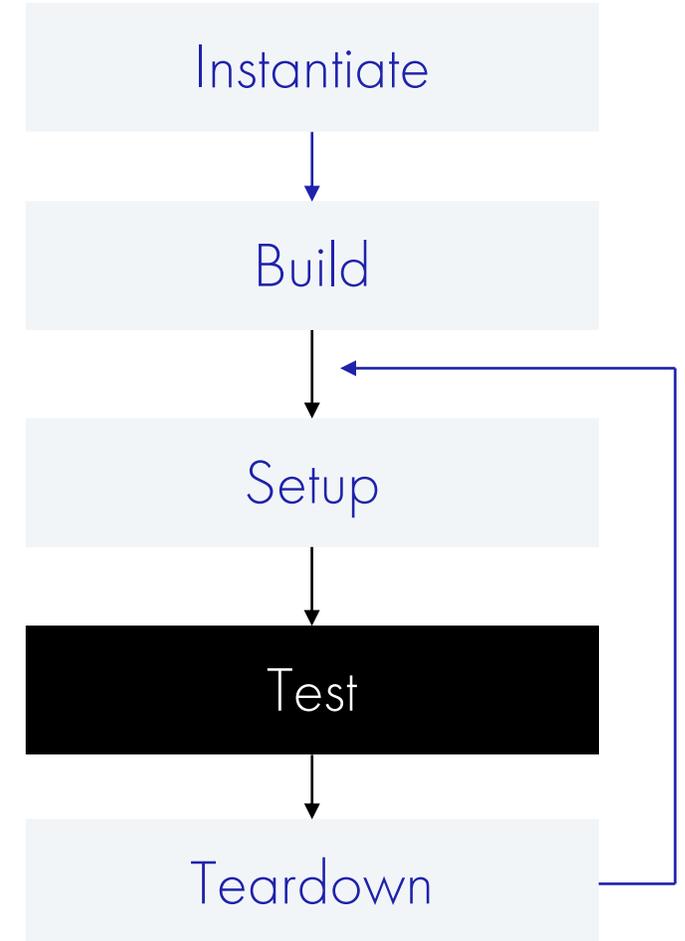




# Test

```
/**
 * *****
 * // Test:
 * //   single_write_then_read
 * //
 * // Desc:
 * //   do a write then a read at the same address
 * // *****
 `SVTEST(single_write_then_read)
   addr = 'h32;
   data = 'h61;

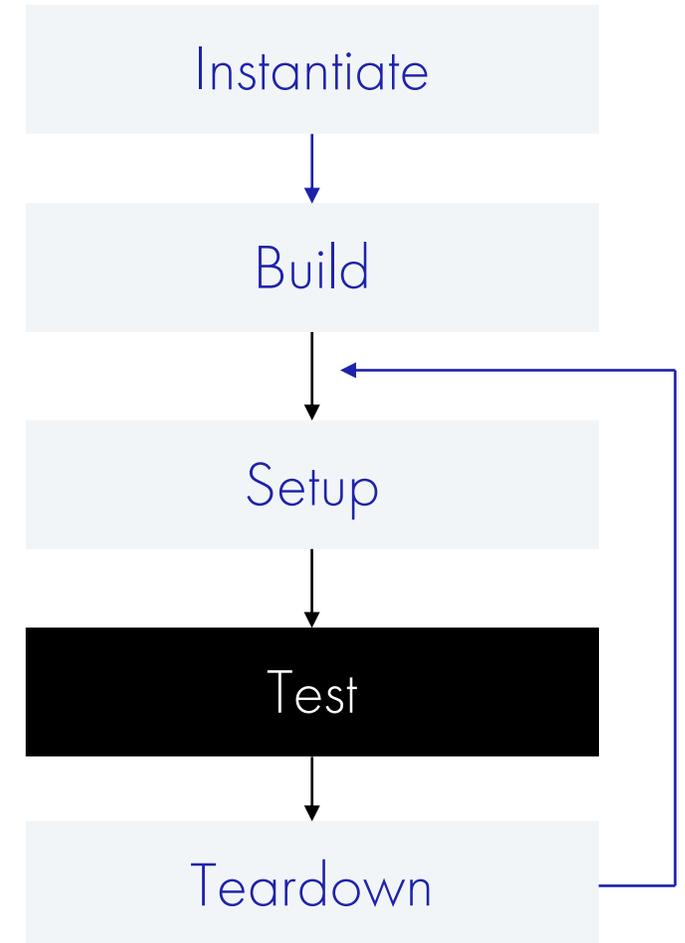
   write(addr, data);
   read(addr, rdata);
   `FAIL_IF(data != rdata);
 `SVTEST_END
```





# Test

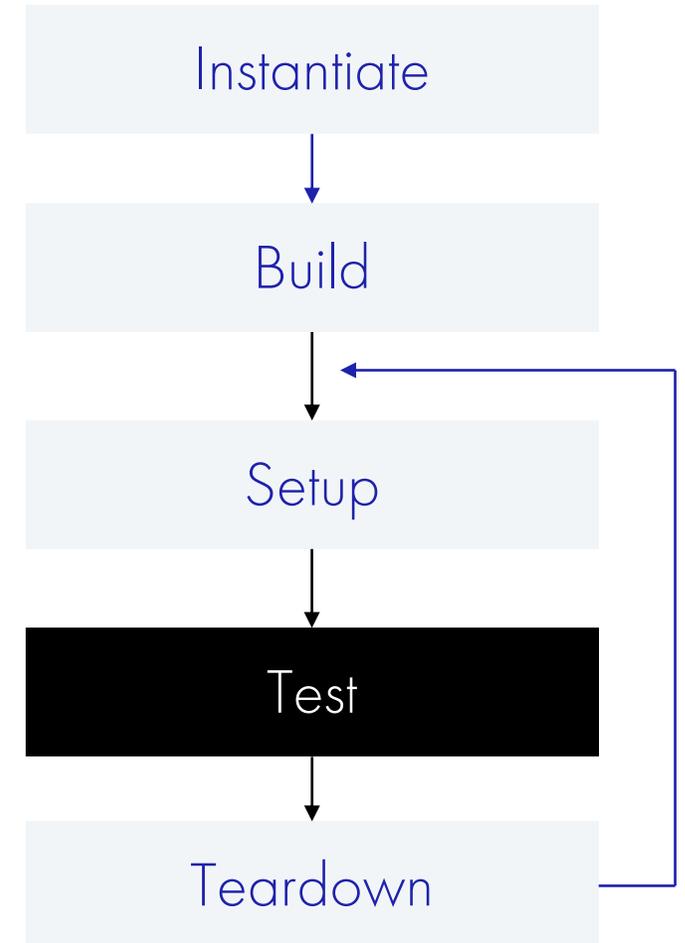
```
//*****  
// Test:  
//   single_write_then_read  
//  
// Desc:  
//   do a write then a read at the same address  
//*****  
`SVTEST(single_write_then_read)  
  addr = 'h32;  
  data = 'h61;  
  
  write(addr, data);  
  read(addr, rdata);  
  `FAIL_IF(data != rdata);  
`SVTEST_END
```





# Test

```
//*****  
// Test:  
//   single_write_then_read  
//  
// Desc:  
//   do a write then a read at the same address  
//*****  
`SVTEST(single_write_then_read)  
  addr = 'h32;  
  data = 'h61;  
  
  write(addr, data);  
  read(addr, rdata);  
  `FAIL_IF(data != rdata);  
`SVTEST_END
```

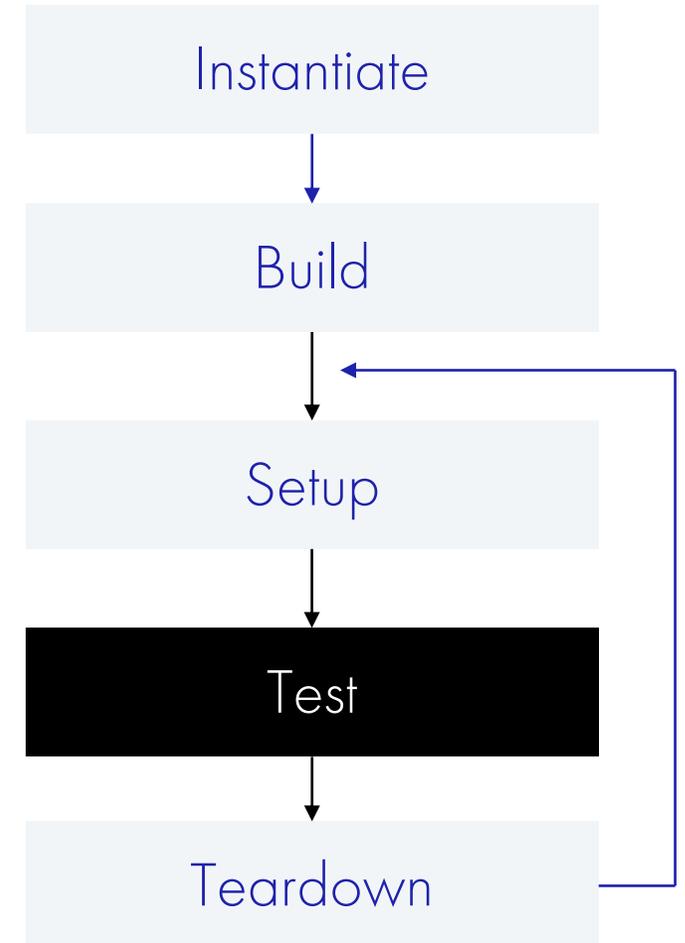




# Test

```
/**
 * *****
 * // Test:
 * //   single_write_then_read
 * //
 * // Desc:
 * //   do a write then a read at the same address
 * // *****
 `SVTEST(single_write_then_read)
   addr = 'h32;
   data = 'h61;

   write(addr, data);
   read(addr, rdata);
   `FAIL_IF(data != rdata);
 `SVTEST_END
```

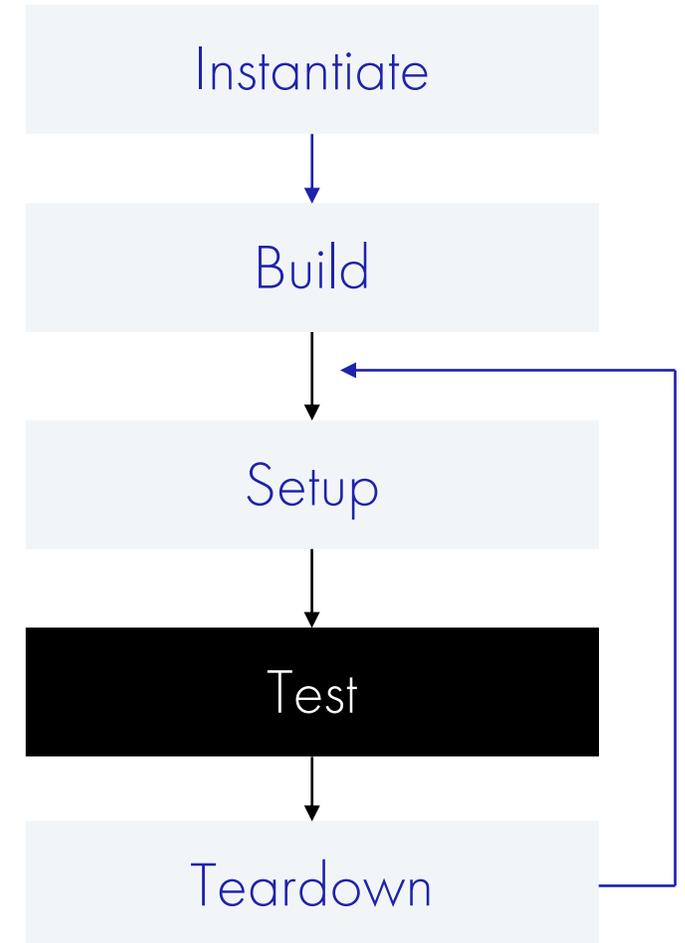




# Test

```
/**
 * *****
 * // Test:
 * //   single_write_then_read
 * //
 * // Desc:
 * //   do a write then a read at the same address
 * // *****
 `SVTEST(single_write_then_read)
   addr = 'h32;
   data = 'h61;

   write(addr, data);
   read(addr, rdata);
   `FAIL_IF(data != rdata);
`SVTEST_END
```

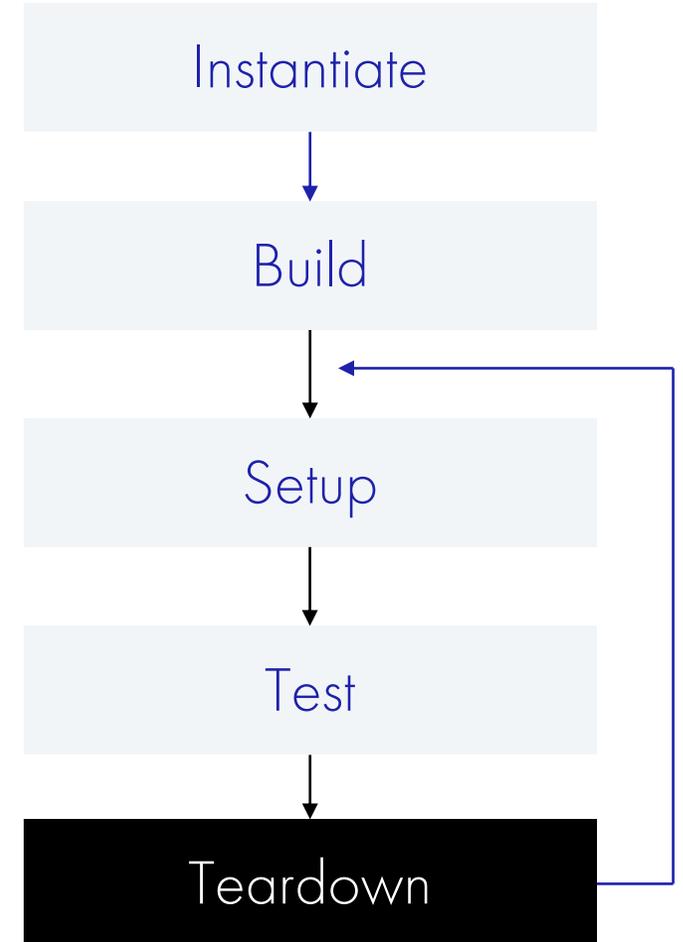




# Teardown

```
task teardown();
    svunit_ut.teardown();

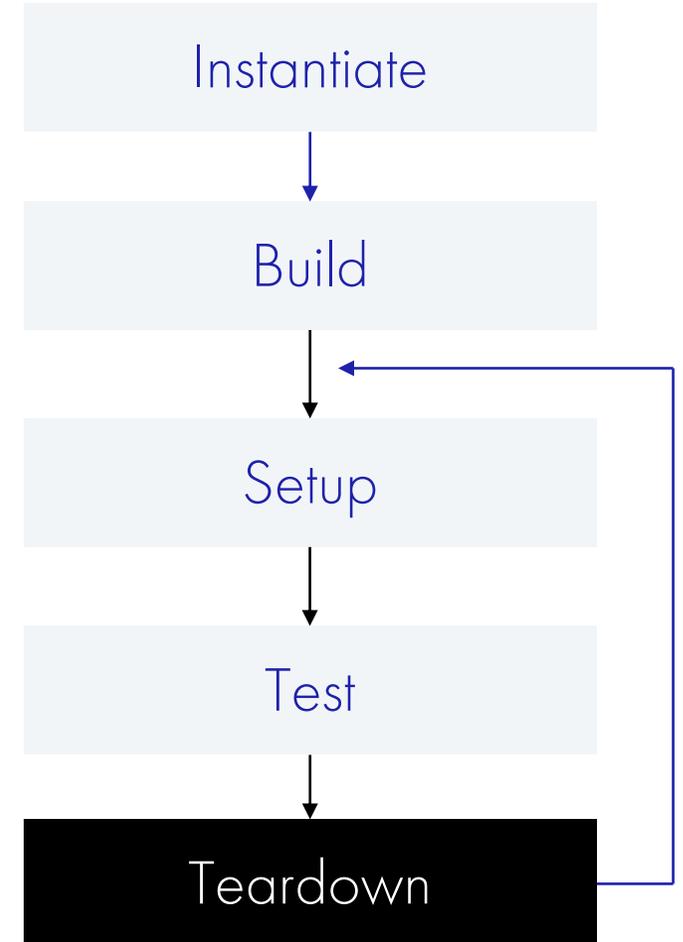
    checker.check_registers_state();
endtask
```





# Teardown

```
task teardown();  
    svunit_ut.teardown();  
  
    checker.check_registers_state();  
endtask
```

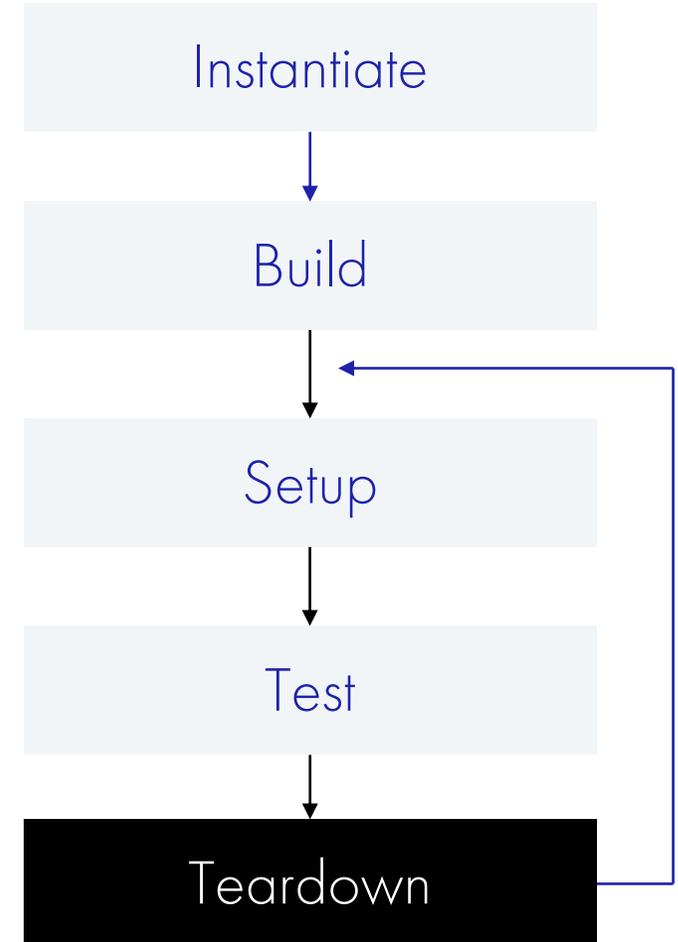




# Teardown

```
task teardown();
    svunit_ut.teardown();

    checker.check_registers_state();
endtask
```





## Макросы сравнения

```
`define FAIL_IF(exp)
`define FAIL_UNLESS(exp)

`define FAIL_IF_EQUAL(a,b)
`define FAIL_UNLESS_EQUAL(a,b)

`define FAIL_IF_STR_EQUAL(a,b)
`define FAIL_UNLESS_STR_EQUAL(a,b)
```

# Тестирование UVM агента



01

BFM/Driver

02

Монитор

03

Монитор функционального покрытия

04

Агент в сборе

# Тестирование UVM агента



01

BFM/Driver

02

Монитор

03

Монитор функционального покрытия

04

Агент в сборе

# Тестирование UVM агента



01

BFM/Driver

02

Монитор

03

Монитор функционального покрытия

04

Агент в сборе

# Тестирование UVM агента



01

BFM/Driver

02

Монитор

03

Монитор функционального покрытия

04

Агент в сборе

# Тестирование UVM агента



01

BFM/Driver

02

Монитор

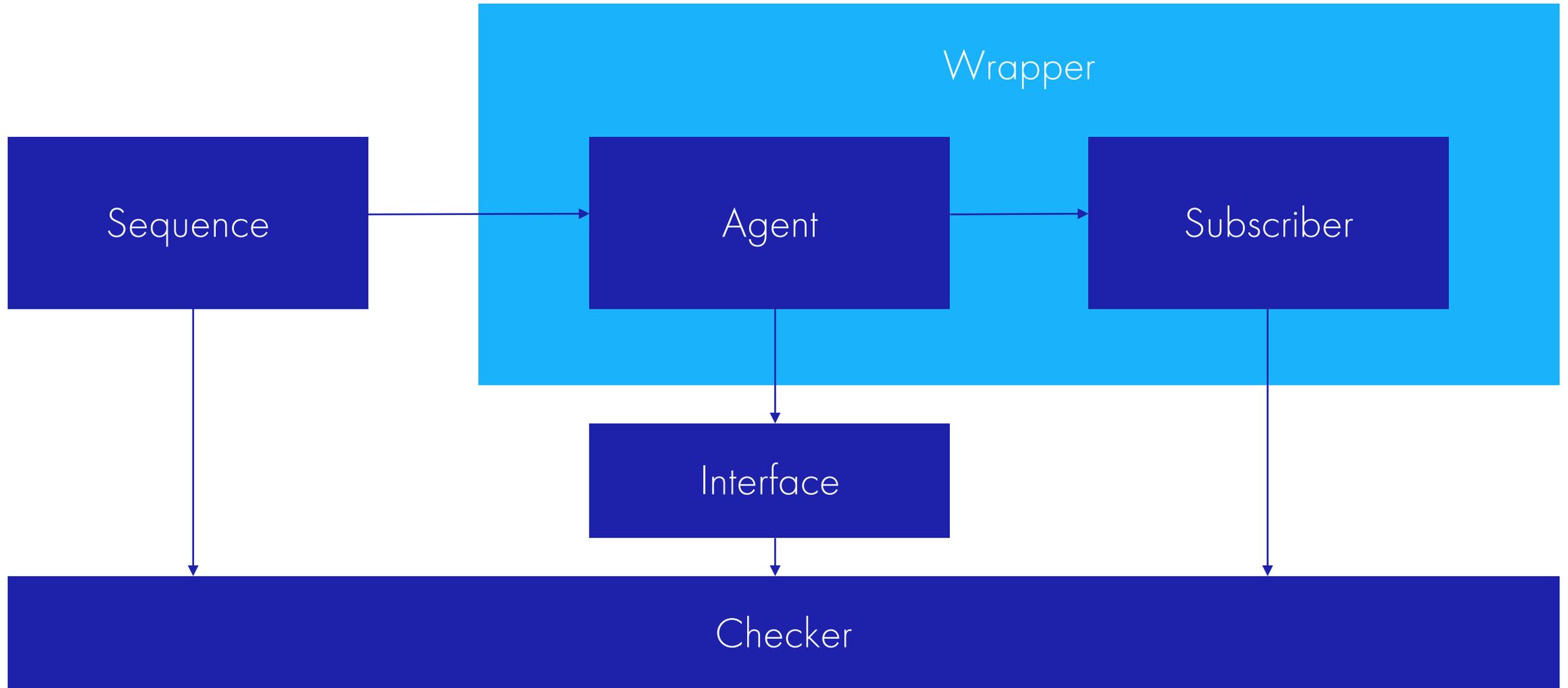
03

Монитор функционального покрытия

04

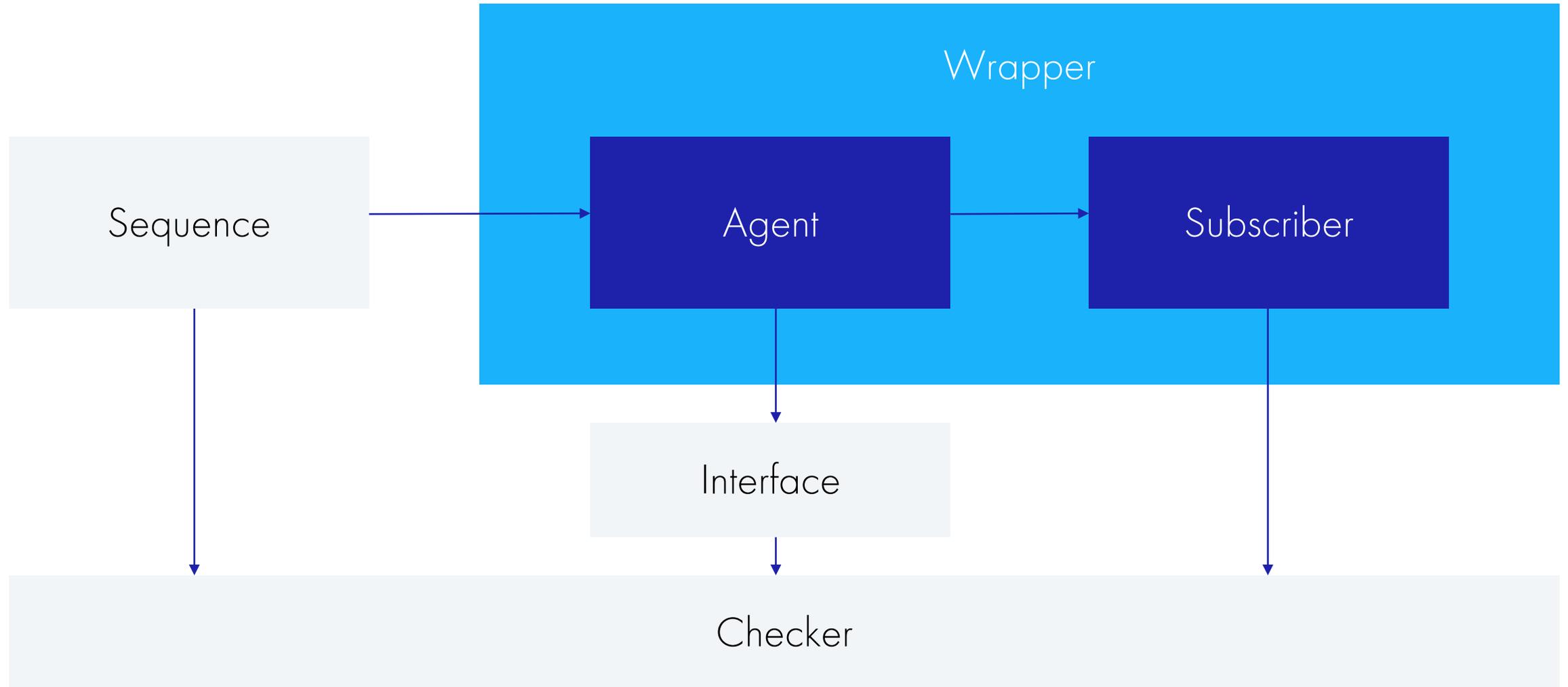
Агент в сборе

# Структурная схема теста UVM агента



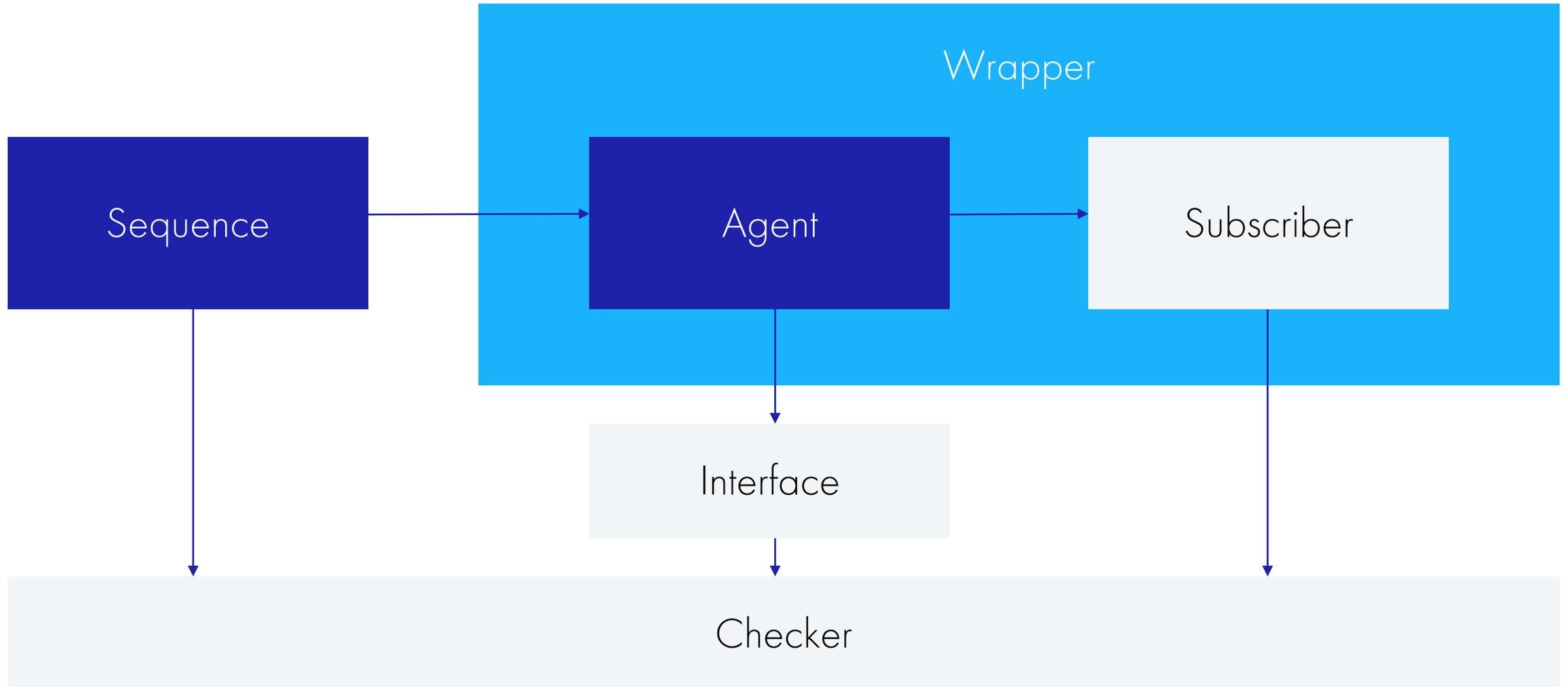


# Структурная схема теста UVM агента



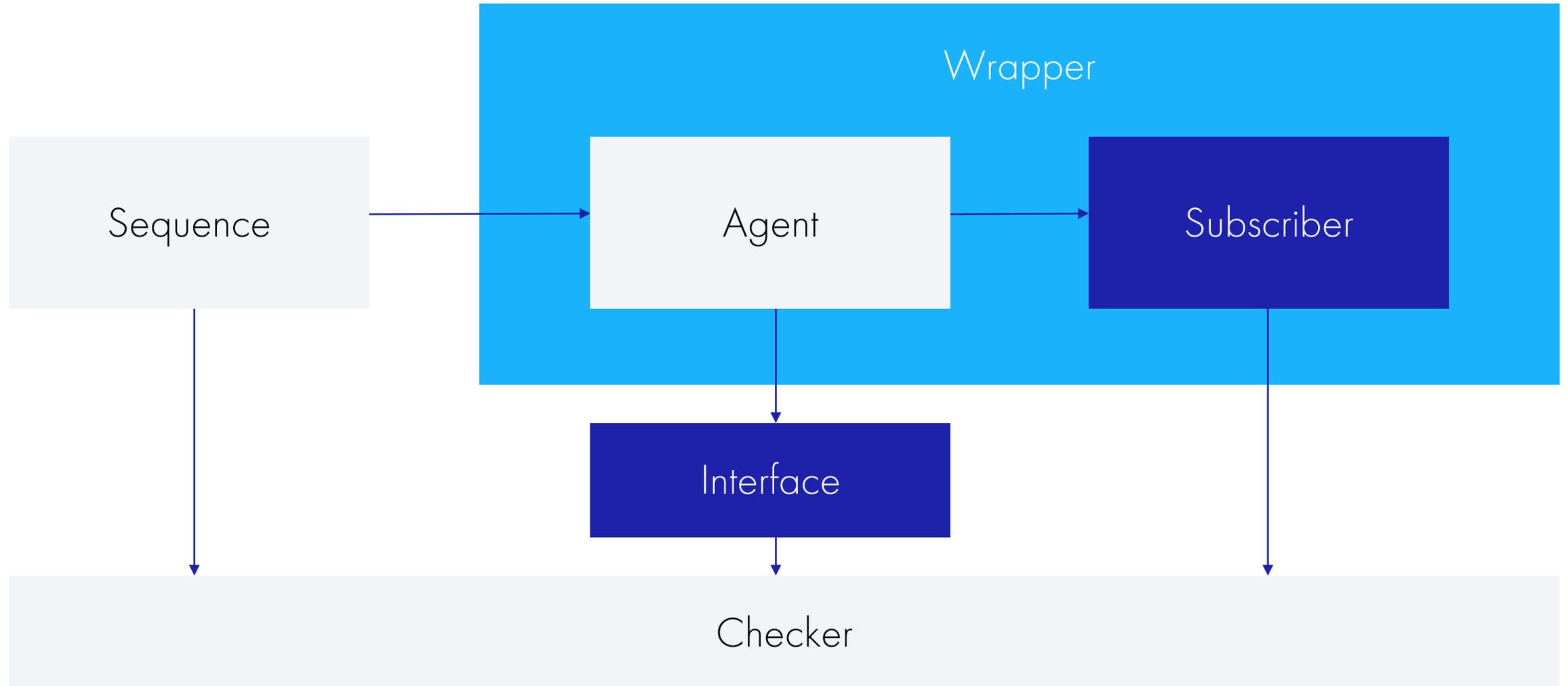


# Структурная схема теста UVM агента

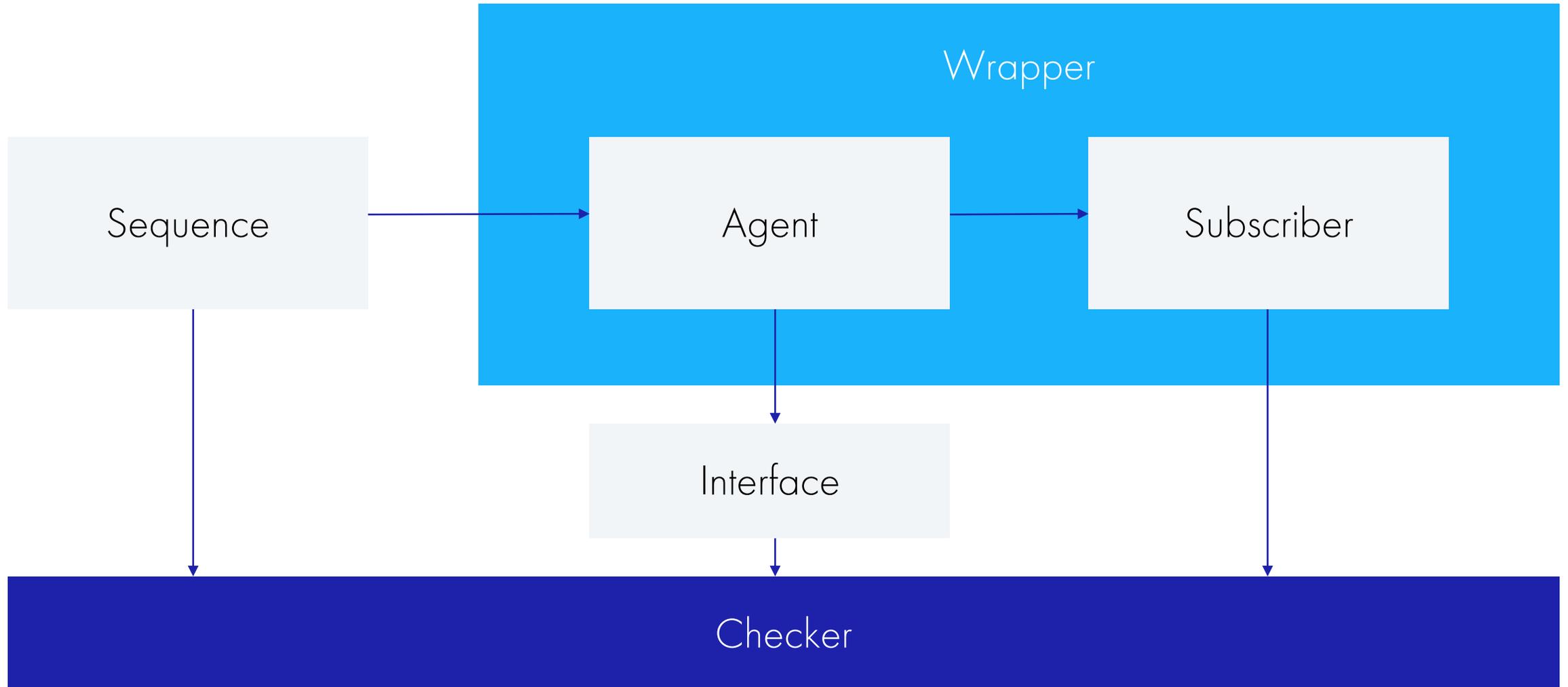




# Структурная схема теста UVM агента



# Структурная схема теста UVM агента





## Негативные сценарии

```
uvm_report_mock::setup();  
uvm_report_mock::expect_fatal();  
uvm_report_mock::expect_error();  
uvm_report_mock::expect_warning();  
uvm_report_mock::verify_complete();
```



## Негативные сценарии

```
uvm_report_mock::setup();  
uvm_report_mock::expect_fatal();  
uvm_report_mock::expect_error();  
uvm_report_mock::expect_warning();  
uvm_report_mock::verify_complete();
```



## Негативные сценарии

```
uvm_report_mock::setup();  
uvm_report_mock::expect_fatal();  
uvm_report_mock::expect_error();  
uvm_report_mock::expect_warning();  
uvm_report_mock::verify_complete();
```



## Негативные сценарии

```
uvm_report_mock::setup();  
uvm_report_mock::expect_fatal();  
uvm_report_mock::expect_error();  
uvm_report_mock::expect_warning();  
uvm_report_mock::verify_complete();
```

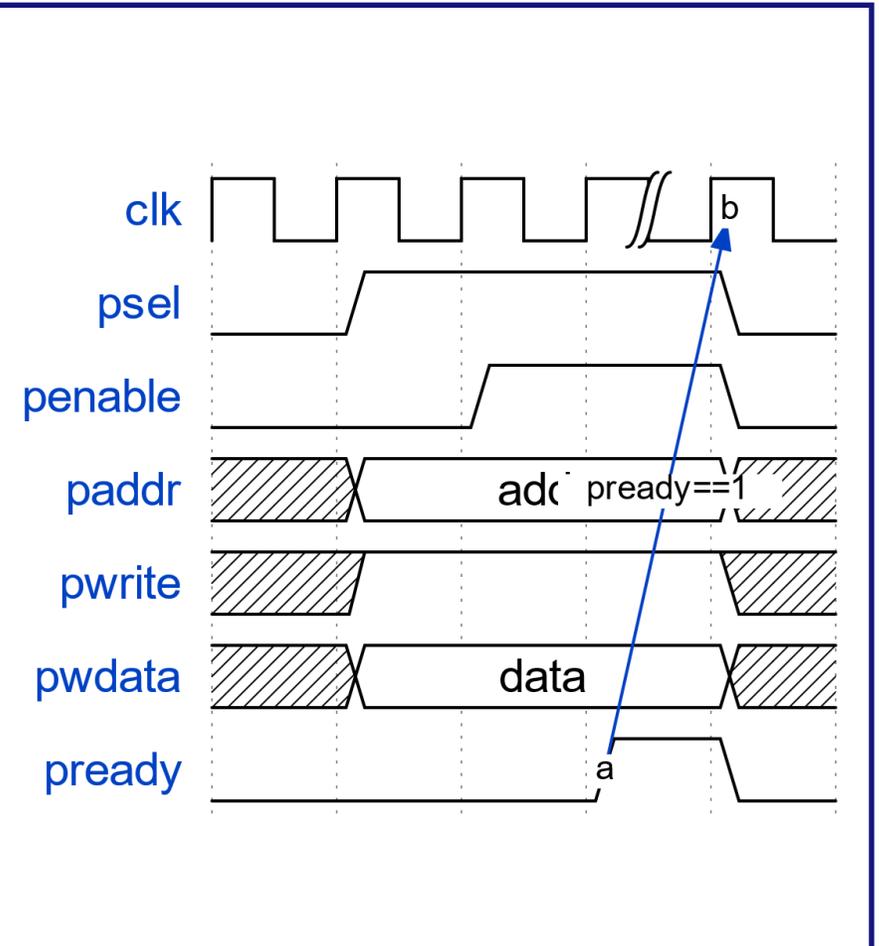


# Wavedrom

```

task write(input logic [7:0] addr,
          input logic [31:0] data);
    step(); nextSamplePoint(); psel = 'h0; ...
    step(); nextSamplePoint(); psel = 'h1; ...
    step(); nextSamplePoint(); penable = 'h1;
    while (!(pready==1)) begin
        step(); nextSamplePoint();
    end
    psel = 'h0; penable = 'h0; ...
endtask

```



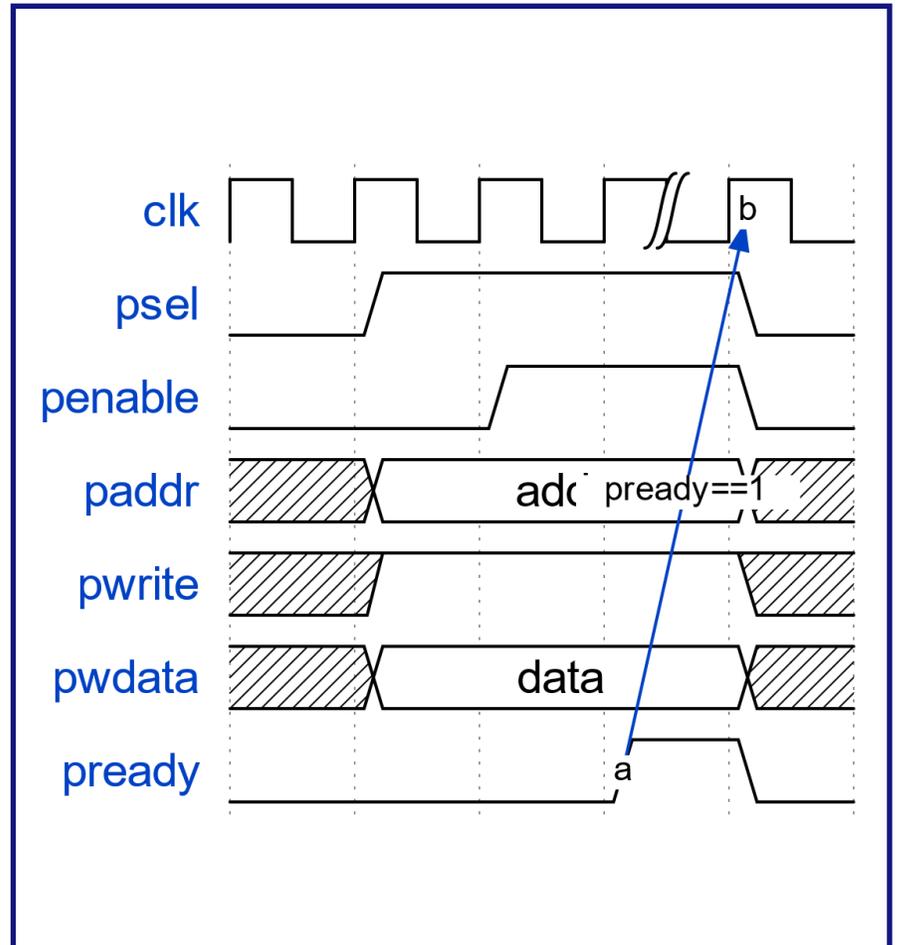


# Wavedrom

```

task write(input logic [7:0] addr,
          input logic [31:0] data);
    step(); nextSamplePoint(); psel = 'h0; ...
    step(); nextSamplePoint(); psel = 'h1; ...
    step(); nextSamplePoint(); penable = 'h1;
    while (!(pready==1)) begin
        step(); nextSamplePoint();
    end
    psel = 'h0; penable = 'h0; ...
endtask

```



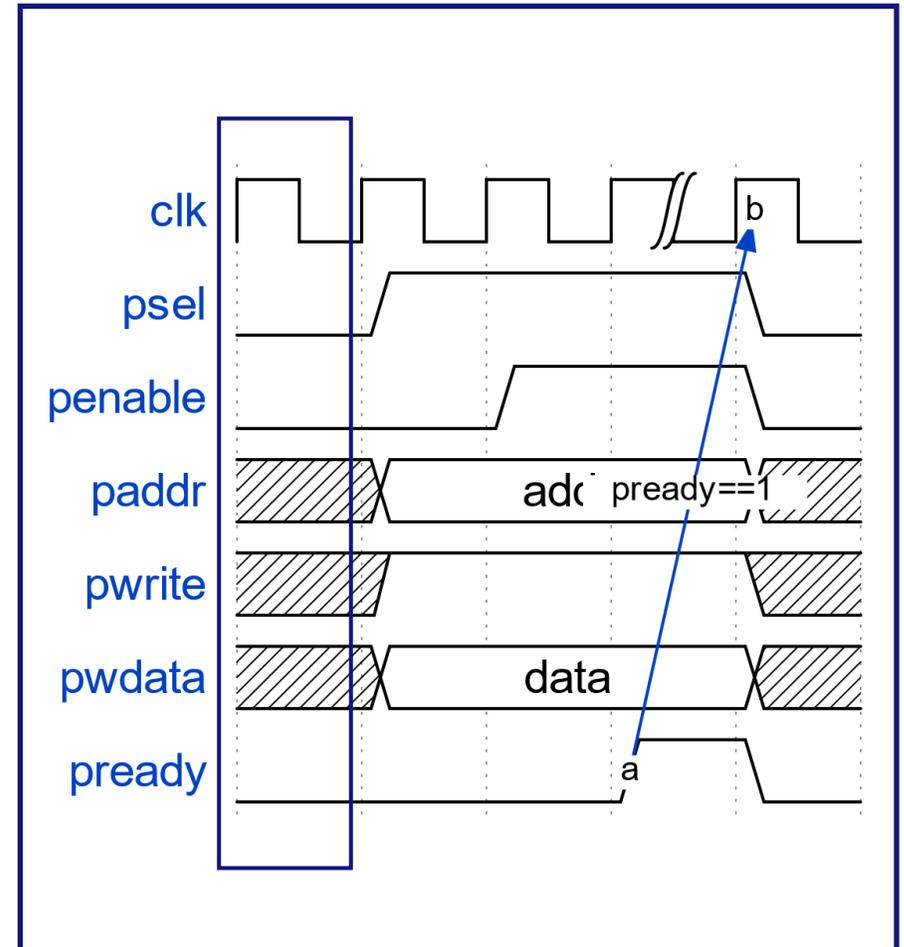


# Wavedrom

```

task write(input logic [7:0] addr,
          input logic [31:0] data);
    step(); nextSamplePoint(); psel = 'h0; ...
    step(); nextSamplePoint(); psel = 'h1; ...
    step(); nextSamplePoint(); penable = 'h1;
    while (!(pready==1)) begin
        step(); nextSamplePoint();
    end
    psel = 'h0; penable = 'h0; ...
endtask

```



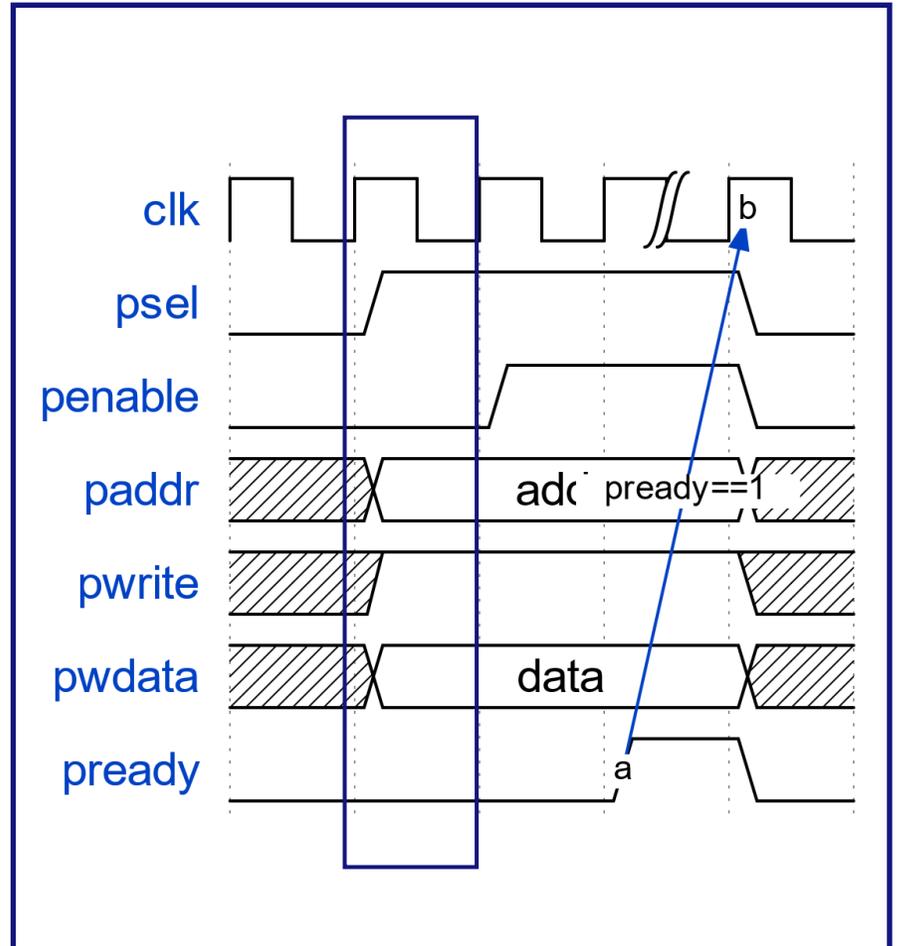


# Wavedrom

```

task write(input logic [7:0] addr,
          input logic [31:0] data);
    step(); nextSamplePoint(); psel = 'h0; ...
    step(); nextSamplePoint(); psel = 'h1; ...
    step(); nextSamplePoint(); penable = 'h1;
    while (!(pready==1)) begin
        step(); nextSamplePoint();
    end
    psel = 'h0; penable = 'h0; ...
endtask

```



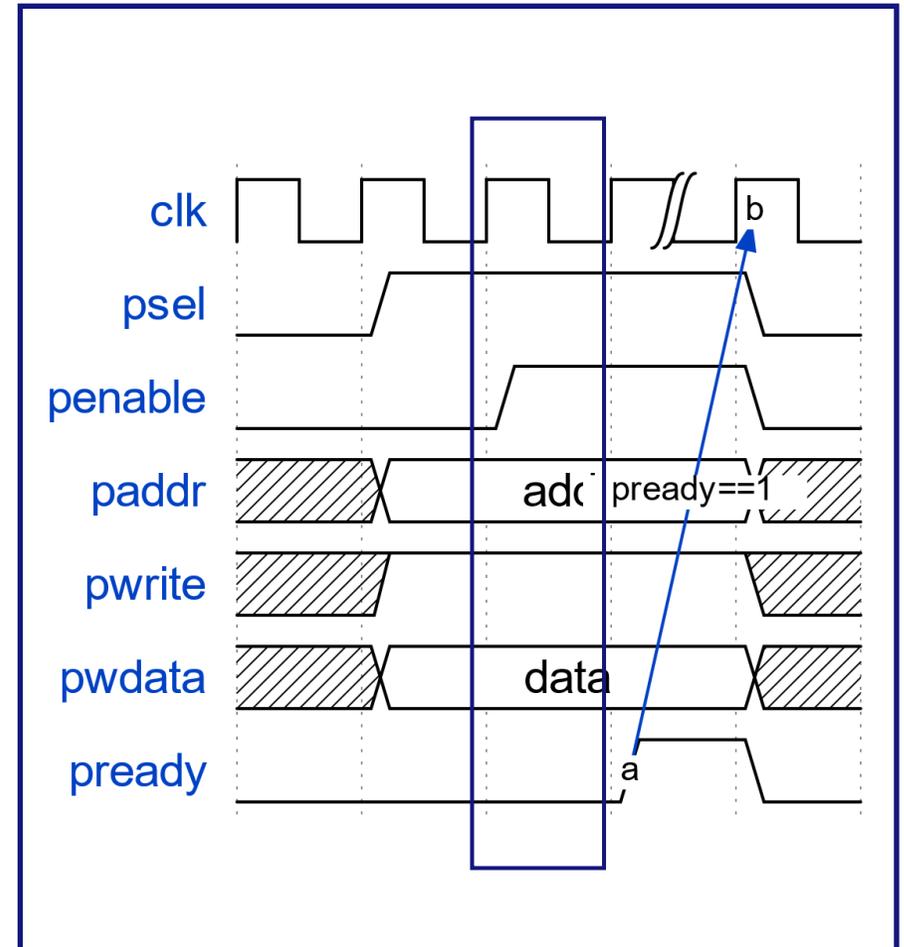


# Wavedrom

```

task write(input logic [7:0] addr,
          input logic [31:0] data);
    step(); nextSamplePoint(); psel = 'h0; ...
    step(); nextSamplePoint(); psel = 'h1; ...
    step(); nextSamplePoint(); penable = 'h1;
    while (!(pready==1)) begin
        step(); nextSamplePoint();
    end
    psel = 'h0; penable = 'h0; ...
endtask

```



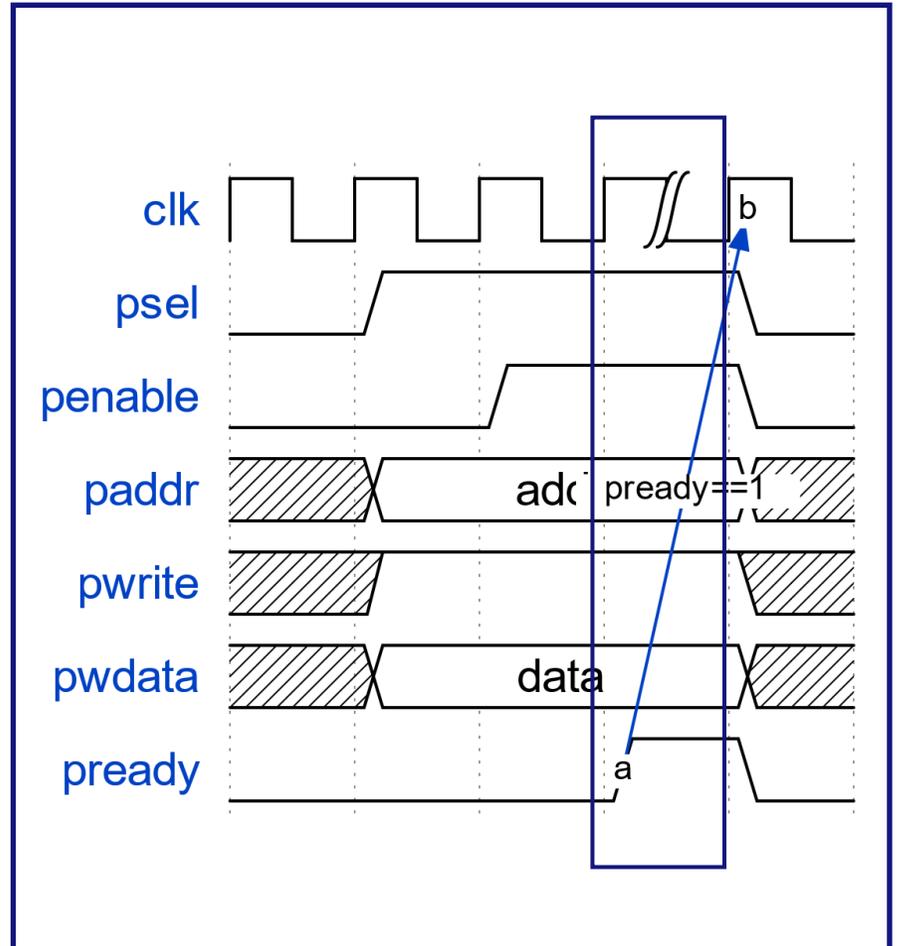


# Wavedrom

```

task write(input logic [7:0] addr,
          input logic [31:0] data);
    step(); nextSamplePoint(); psel = 'h0; ...
    step(); nextSamplePoint(); psel = 'h1; ...
    step(); nextSamplePoint(); penable = 'h1;
    while (!(pready==1)) begin
        step(); nextSamplePoint();
    end
    psel = 'h0; penable = 'h0; ...
endtask

```



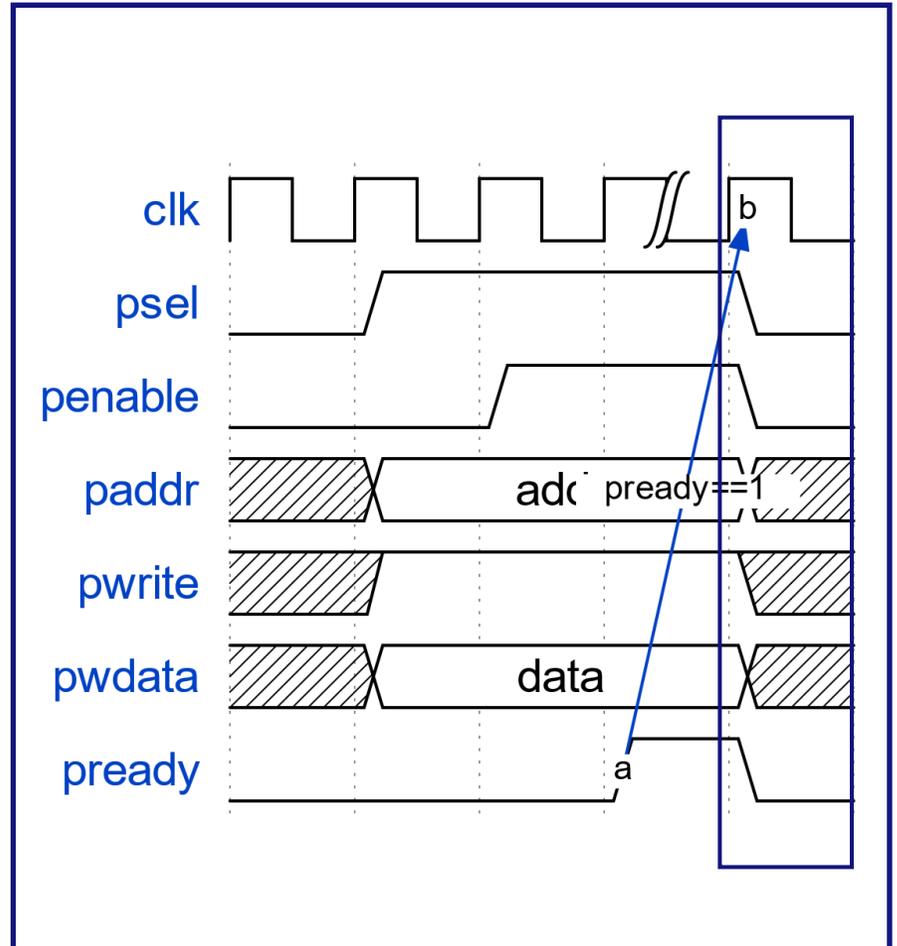


# Wavedrom

```

task write(input logic [7:0] addr,
          input logic [31:0] data);
    step(); nextSamplePoint(); psel = 'h0; ...
    step(); nextSamplePoint(); psel = 'h1; ...
    step(); nextSamplePoint(); penable = 'h1;
    while (!(pready==1)) begin
        step(); nextSamplePoint();
    end
    psel = 'h0; penable = 'h0; ...
endtask

```





## Результаты внедрения

Поставка переиспользуемых  
верификационных  
компонентов продуктового  
качества

Легкое изменение  
компонентов

Высокая скорость  
проверки и отладки

Одинаковые скрипты  
для запуска тестов  
всех компонентов

Не нужен рабочий RTL  
для отладки  
компонентов

Повторяйте это упражнение каждый день —  
и ~~спина болят~~ багов в тестбенчах не будет



## Полезные ссылки



SVUnit



Examples



Modules



UVM

# Вопросы и ответы





БУДУЩЕЕ  
В НАШИХ  
РУКАХ

# Партнеры конференции



# Наши ресурсы



Как найти  
сообщество

[FPGA-Systems.ru](http://FPGA-Systems.ru)

[FPGA-Systems Magazine \(FSM\)](#)

[@fpgasystems](#)

[admin@fpga-systems.ru](mailto:admin@fpga-systems.ru)

[Youtube](#)

[@fpgasystems](#)