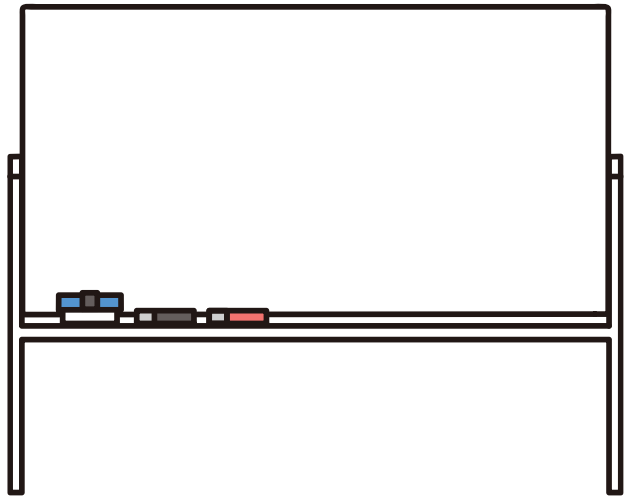
Open Source Step-and-Compare: делаем индустриальный подход к верификации RISC-V доступным каждому

Чусов Сергей Андреевич

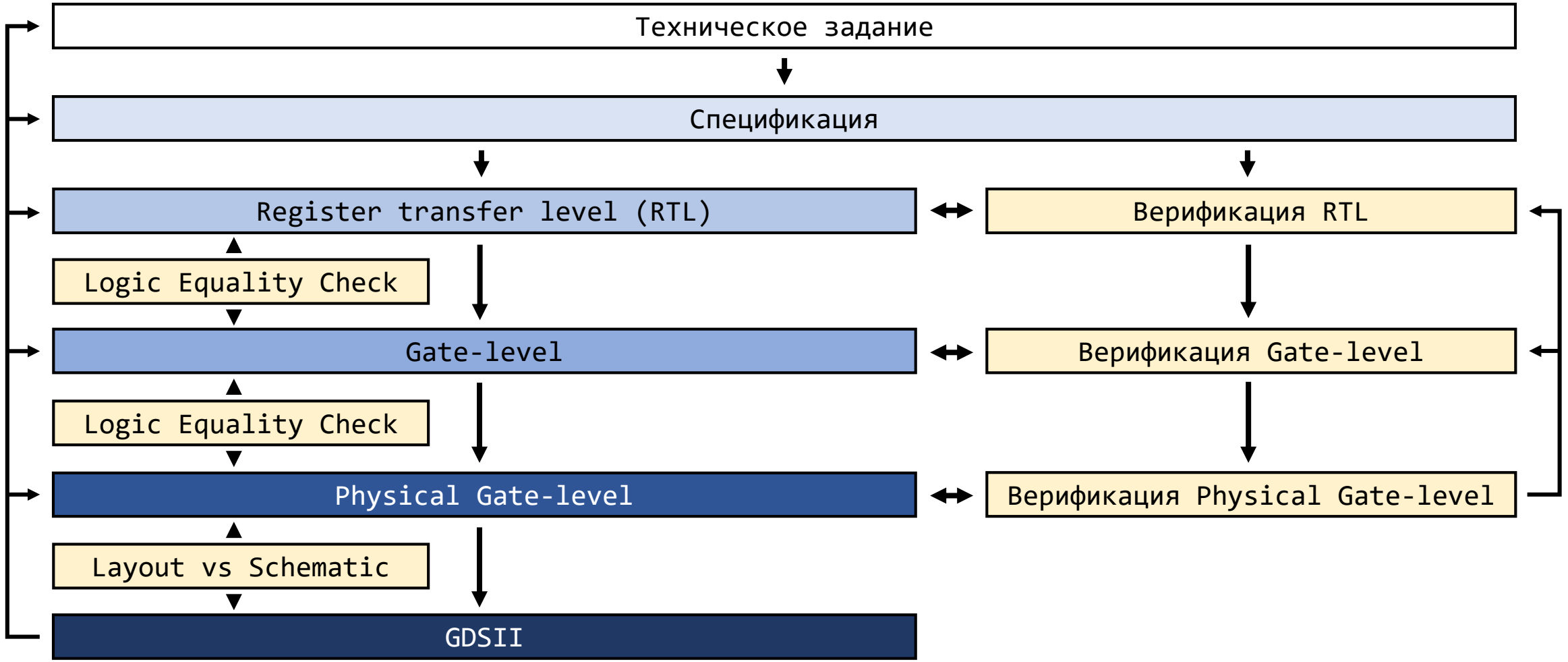
НИЛ энергоэффективных Систем на Кристалле НИУ МИЭТ

План выступления

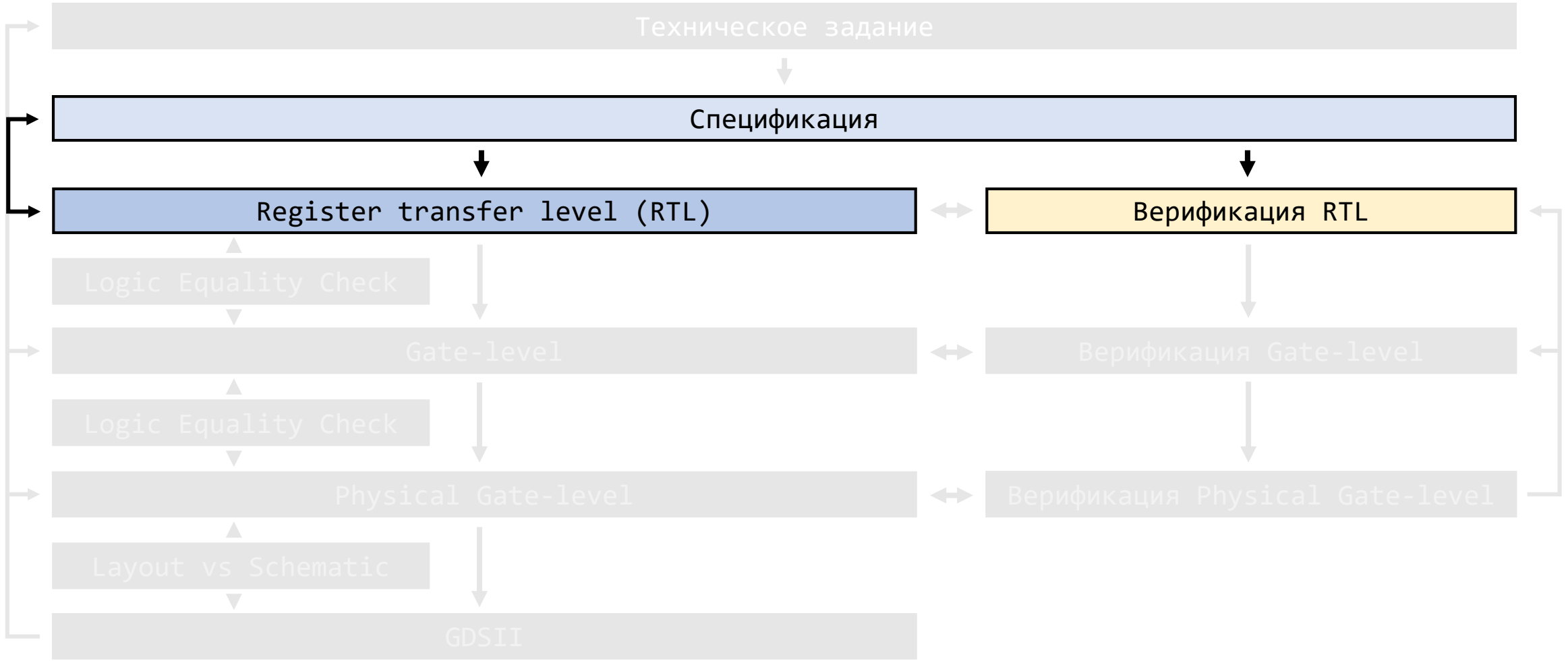
- Верификация и функциональная верификация
- Функциональная верификация процессорных ядер
- Особенности верификации RISC-V ядер
- Существующие подходы к верификации RISC-V ядер
- Step-and-Compare подход к верификации RISC-V ядер
- Step-and-Compare с использованием открытого ПО



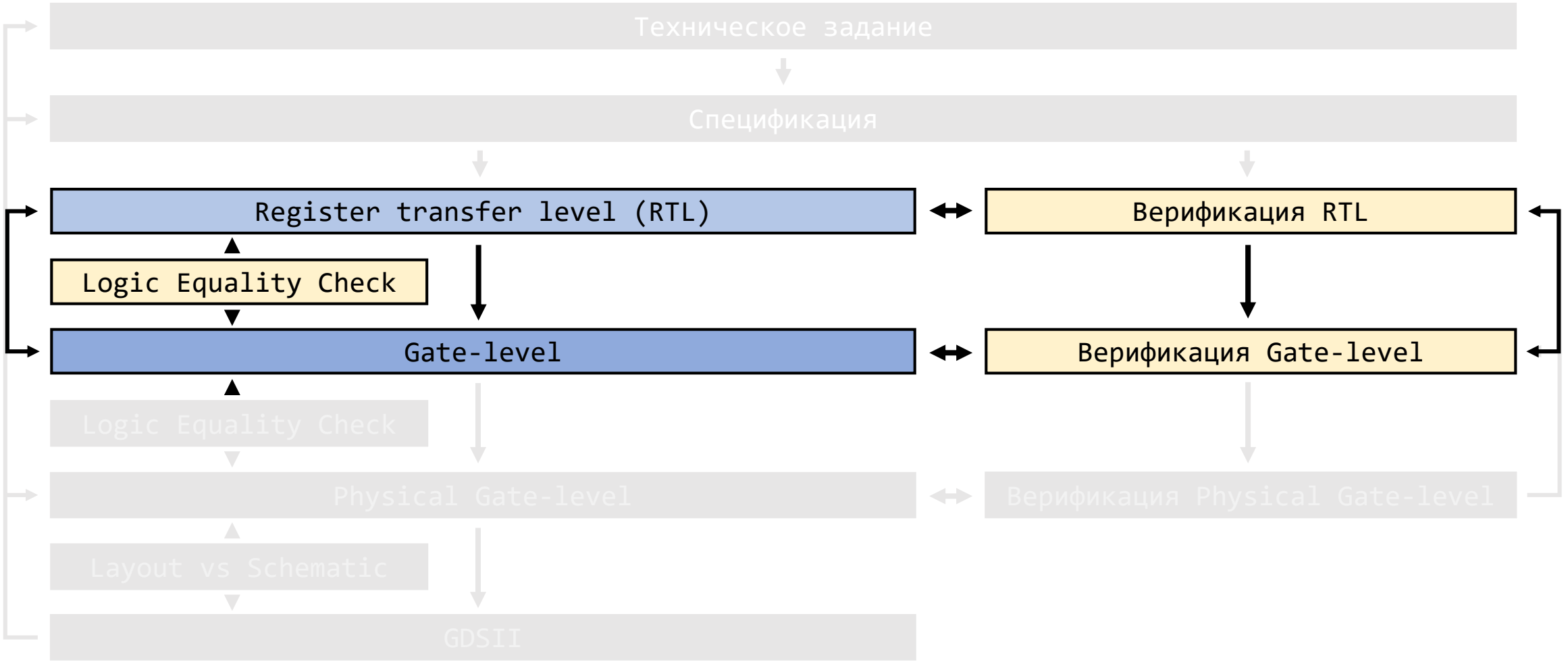
Верификация и функциональная верификация



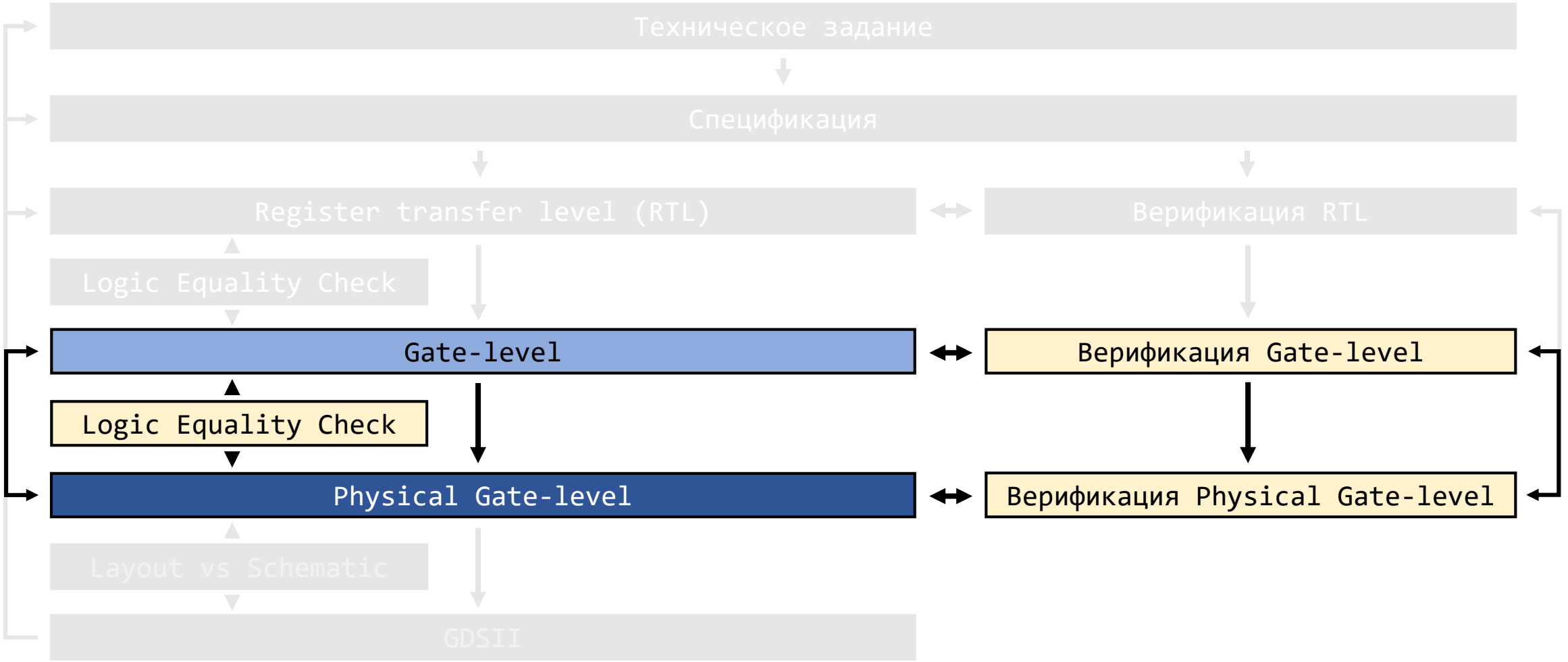
Верификация и функциональная верификация



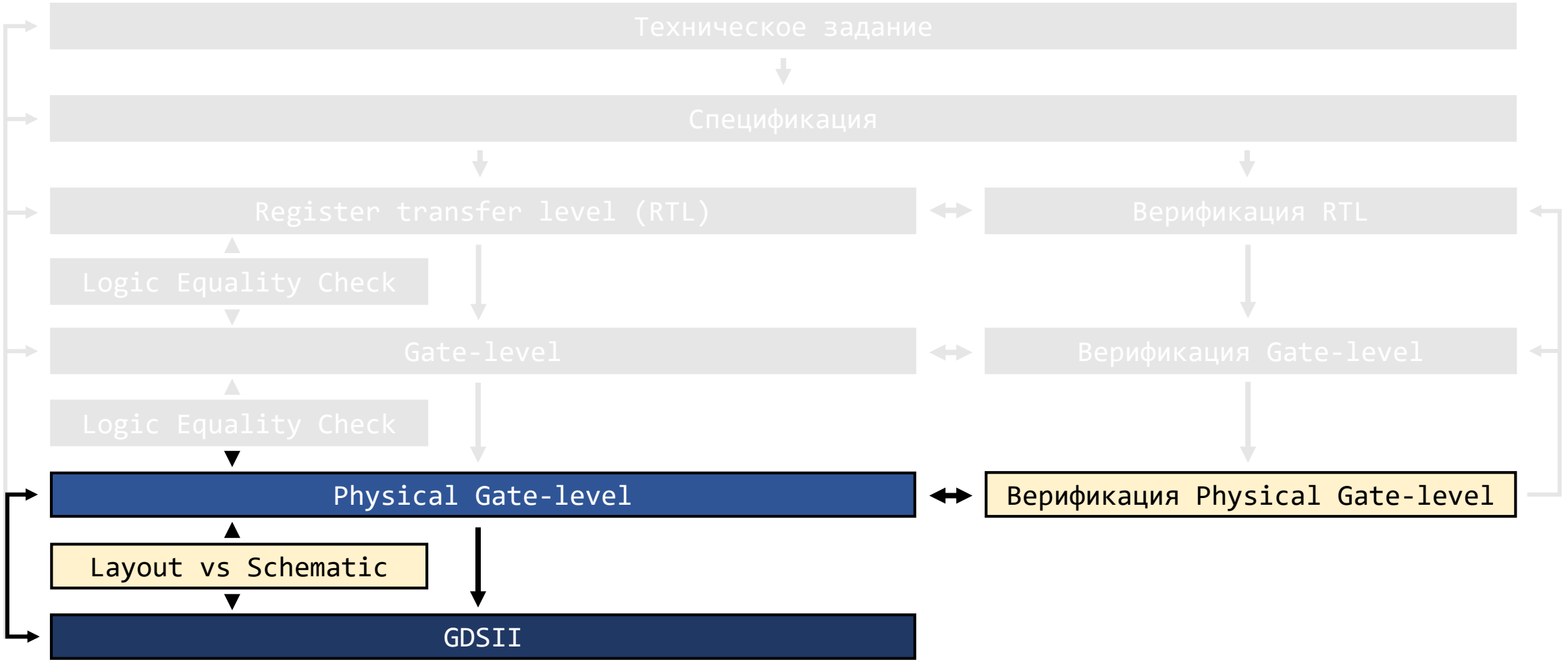
Верификация и функциональная верификация



Верификация и функциональная верификация



Верификация и функциональная верификация



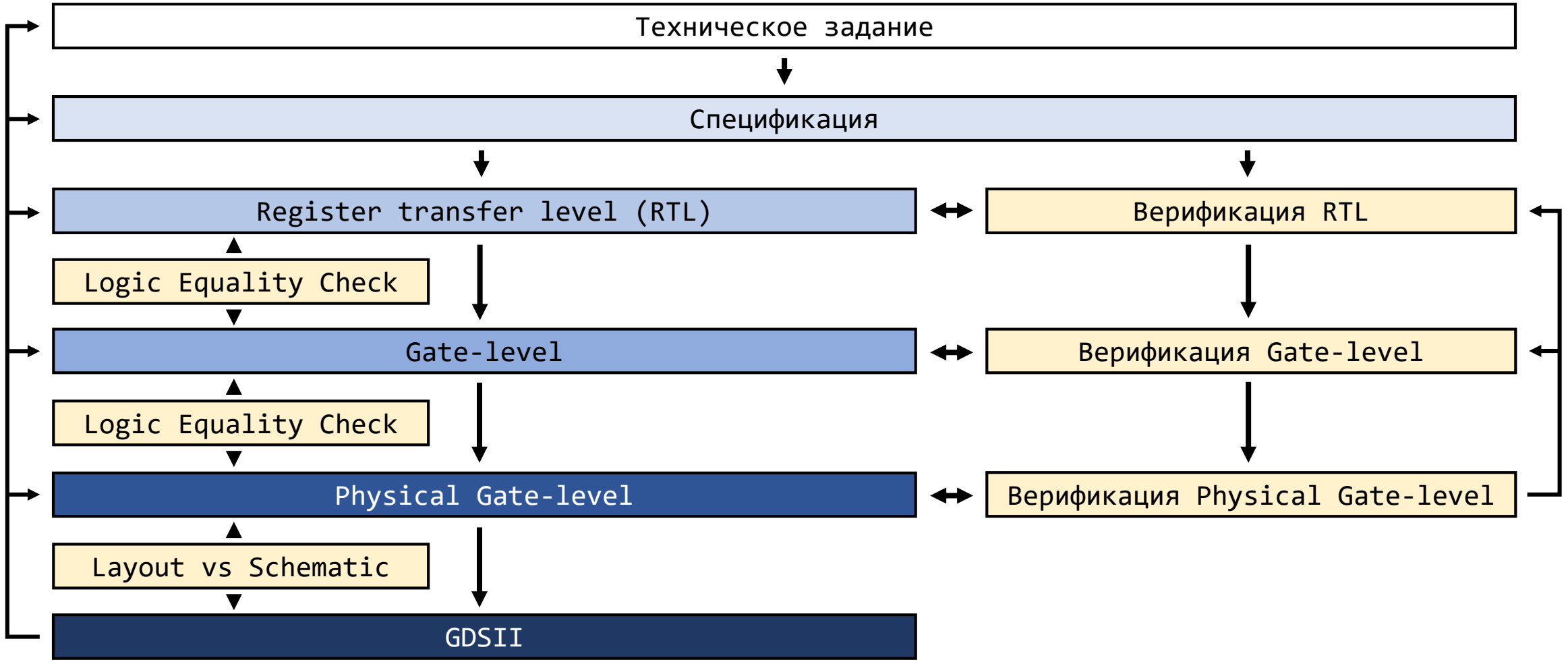
Верификация и функциональная верификация

Верификация – процесс подтверждения эквивалентности представлений.

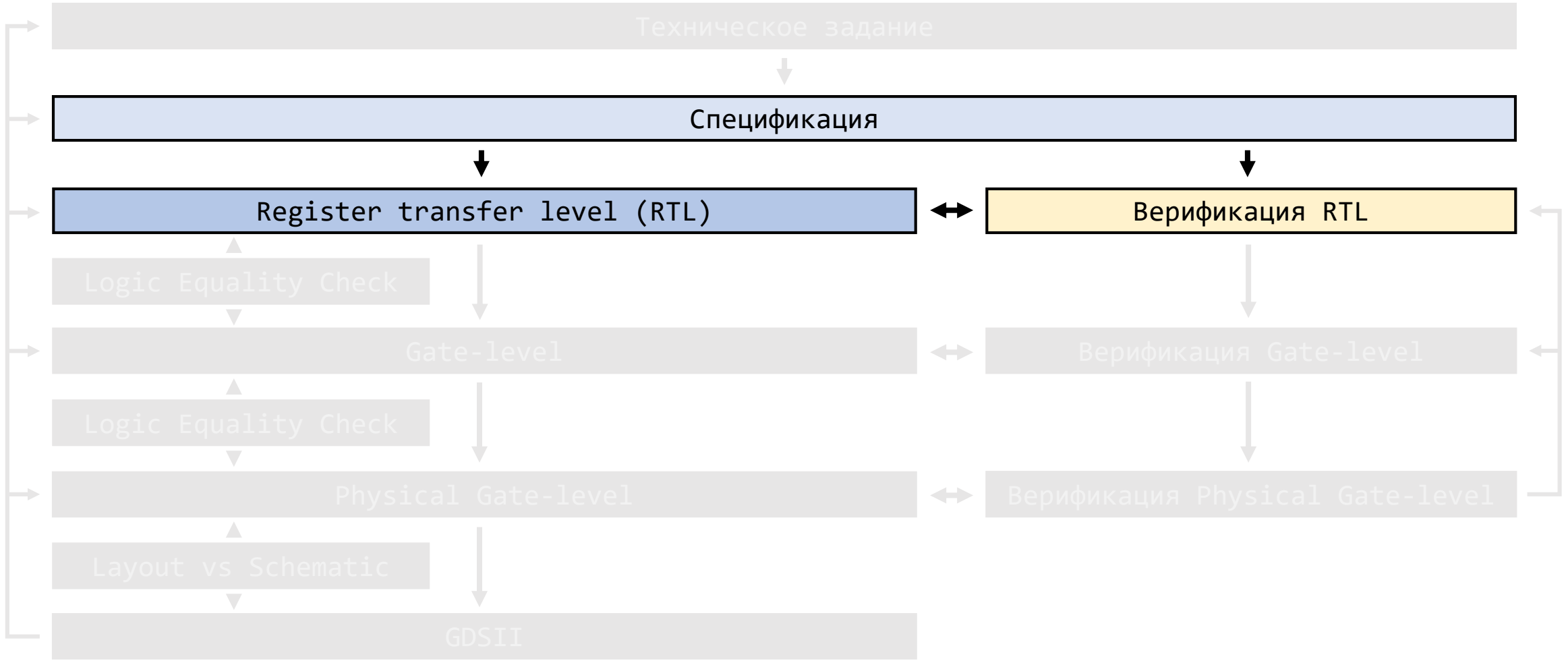
Под представлениями подразумеваются:

- Спецификация;
- RTL;
- Gate-level;
- Physical Gate-level;
- GDSII.

Верификация и функциональная верификация

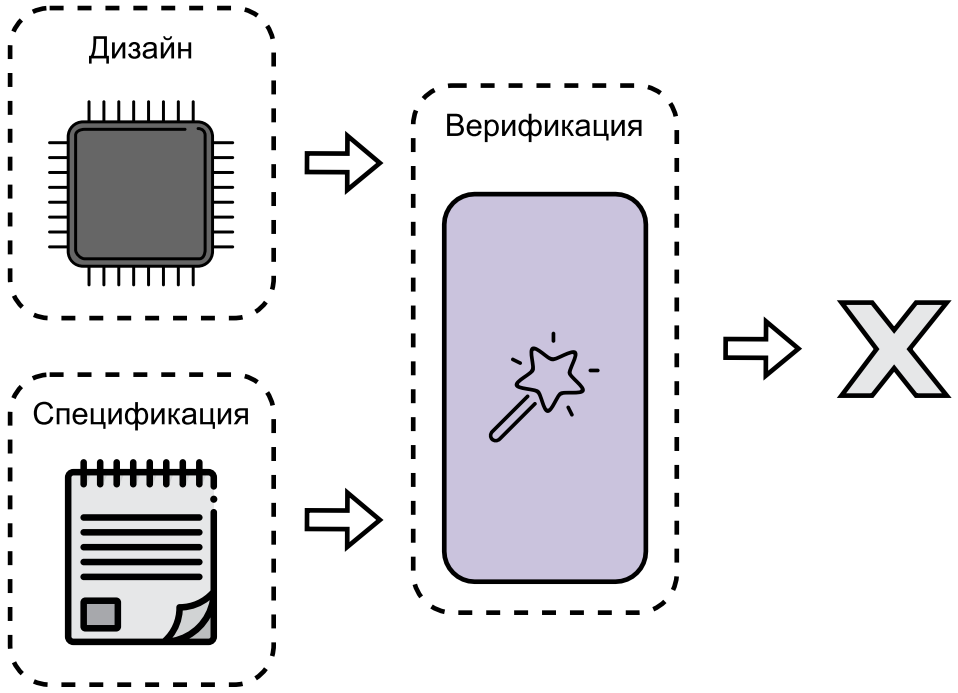


Верификация и функциональная верификация



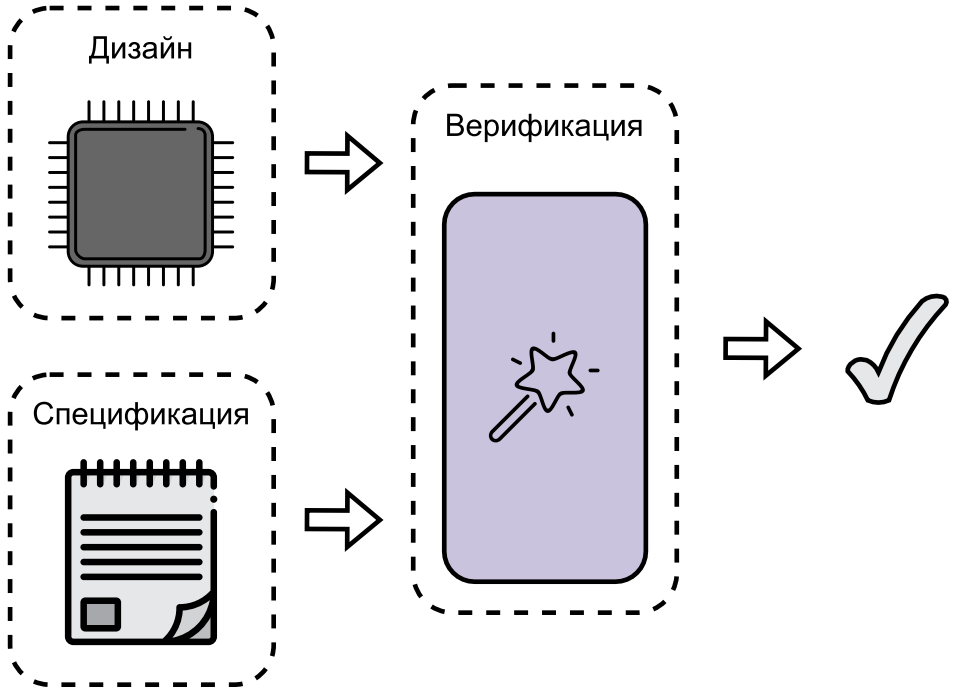
Верификация и функциональная верификация

Функциональная верификация – процесс подтверждения эквивалентности RTL и спецификации.



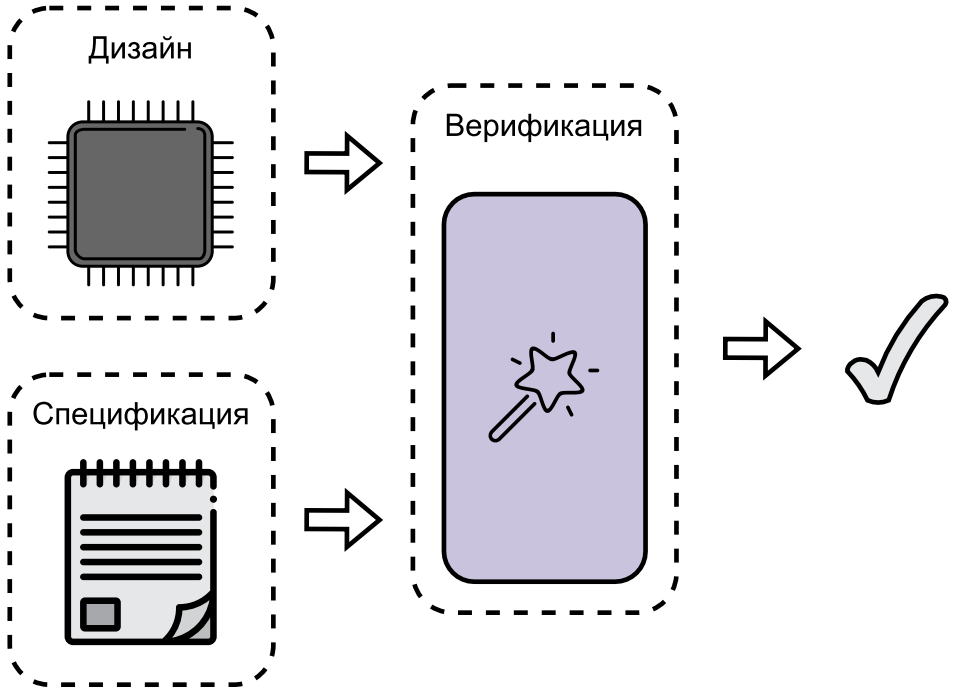
Верификация и функциональная верификация

Функциональная верификация – процесс подтверждения эквивалентности RTL и спецификации.



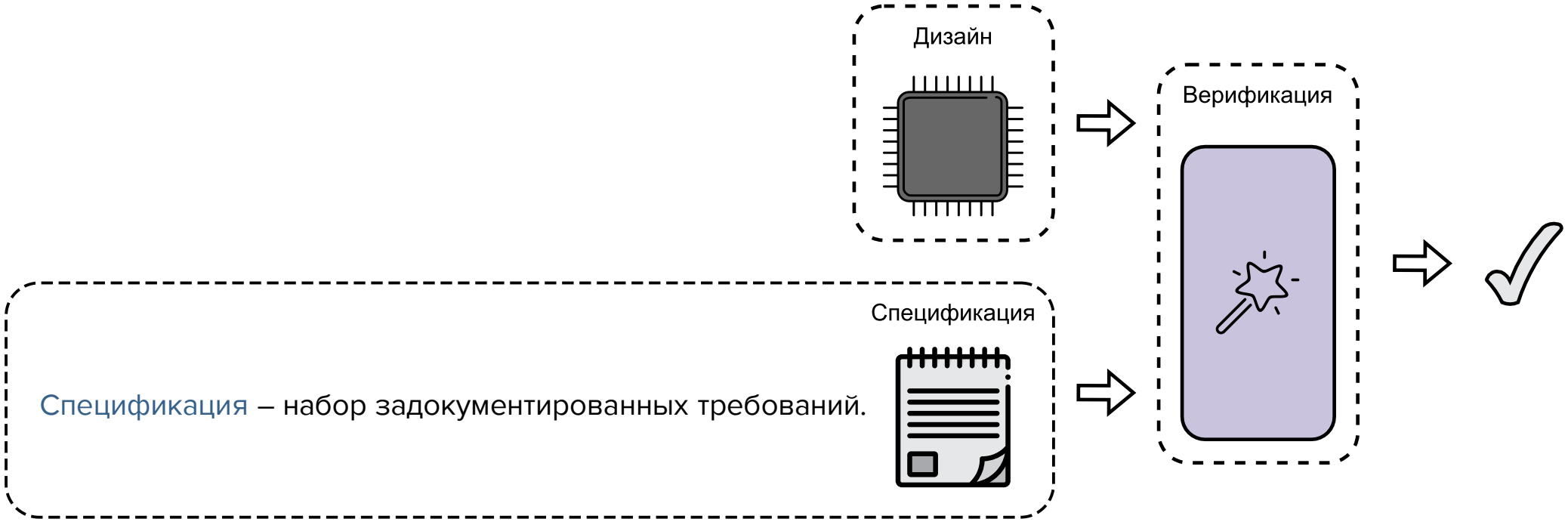
Верификация и функциональная верификация

Функциональная верификация – процесс подтверждения эквивалентности RTL и спецификации.



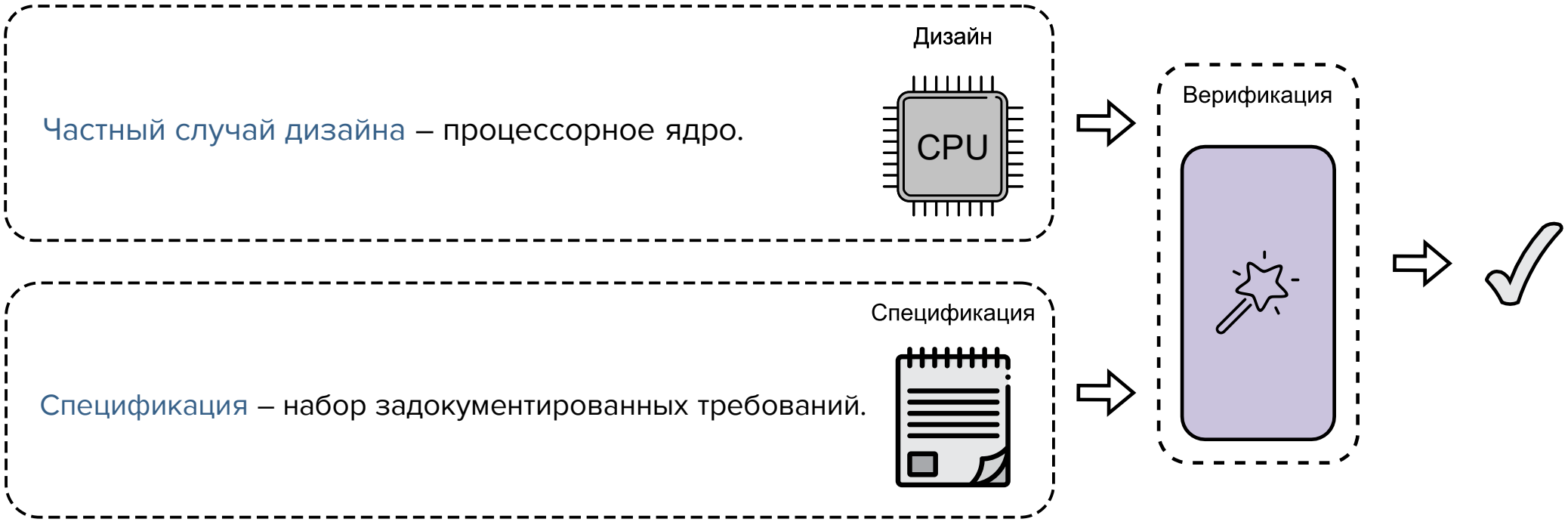
Верификация и функциональная верификация

Функциональная верификация – процесс подтверждения эквивалентности RTL и спецификации.



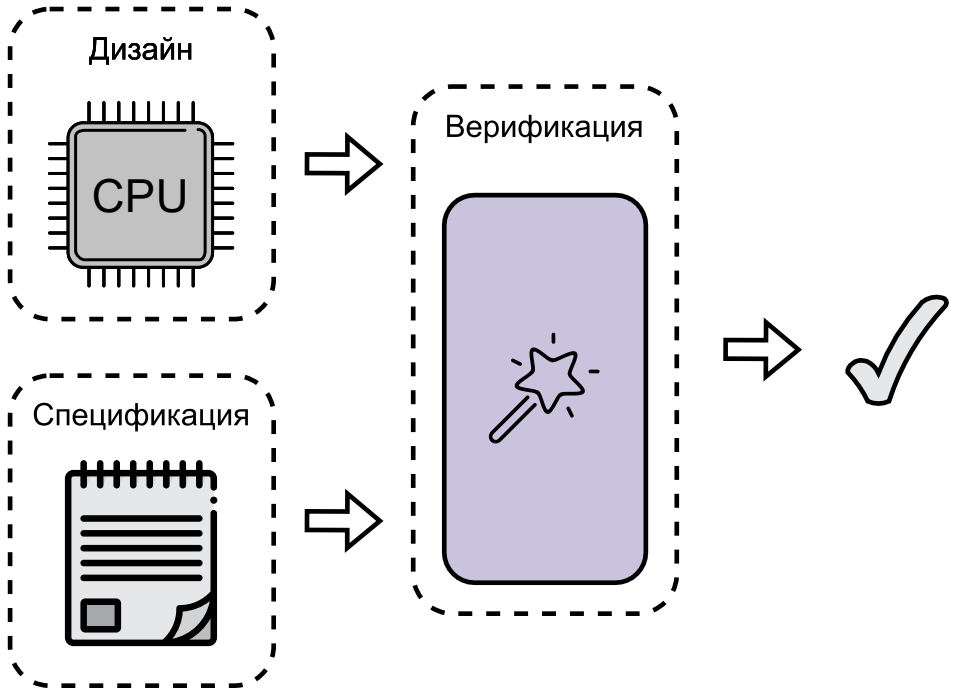
Функциональная верификация процессорных ядер

Функциональная верификация – процесс подтверждения эквивалентности RTL и спецификации.



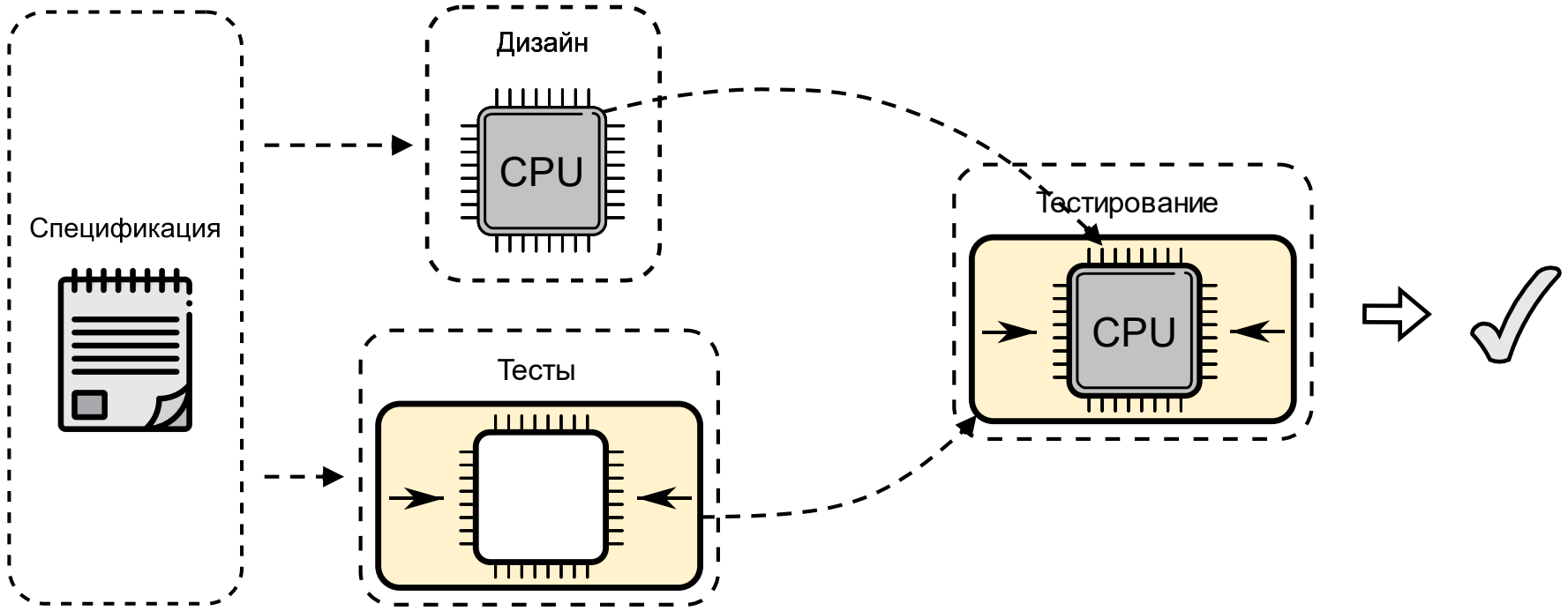
Функциональная верификация процессорных ядер

Функциональная верификация – процесс подтверждения эквивалентности RTL и спецификации.

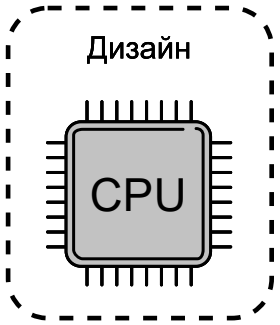


Функциональная верификация процессорных ядер

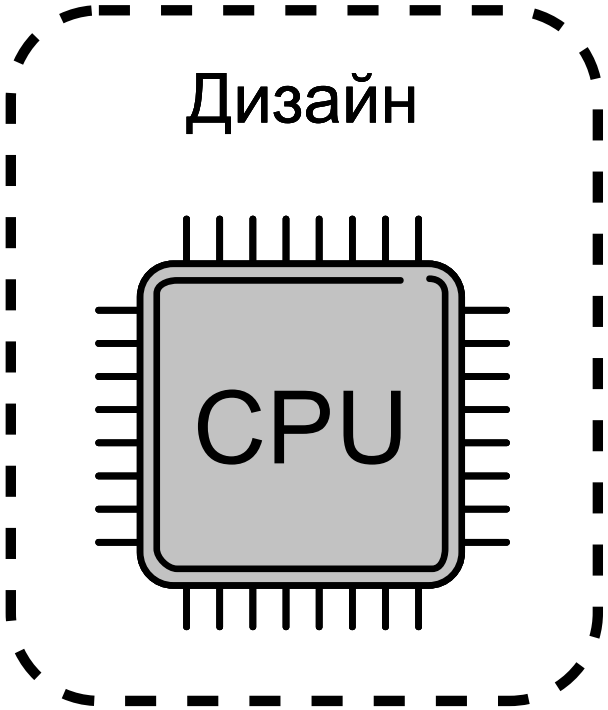
Функциональная верификация – процесс подтверждения эквивалентности RTL и спецификации.



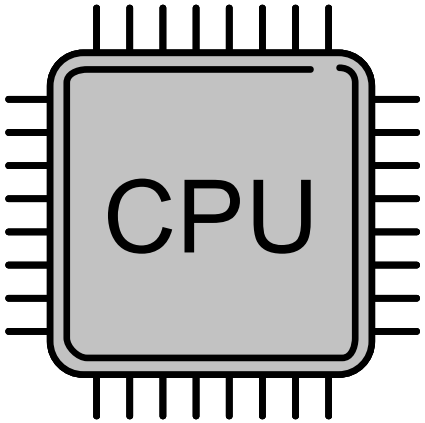
Функциональная верификация процессорных ядер



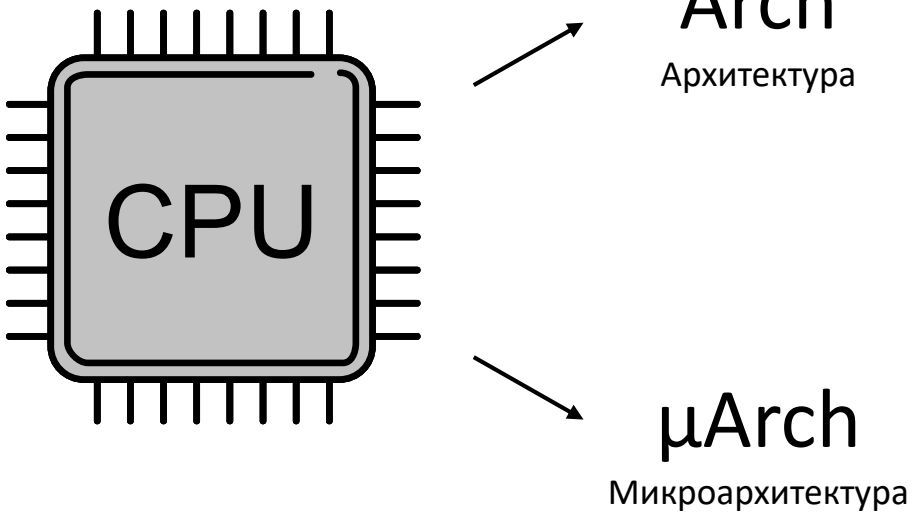
Функциональная верификация процессорных ядер



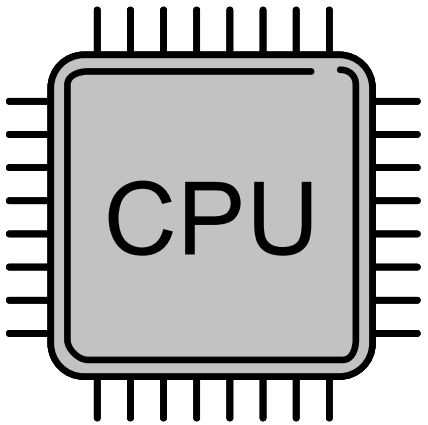
Функциональная верификация процессорных ядер



Функциональная верификация процессорных ядер



Функциональная верификация процессорных ядер



Arch

Архитектура

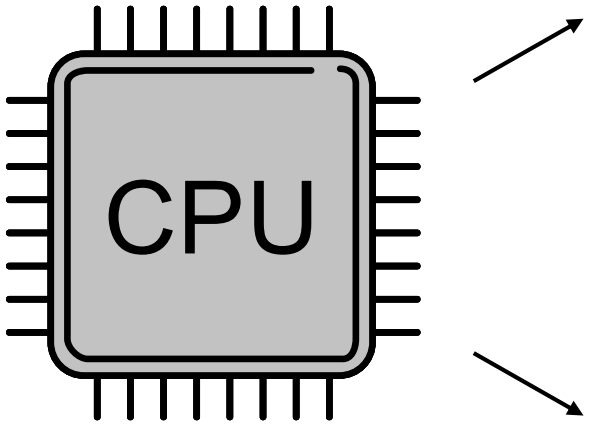
ISA

Архитектура Набора Команд

µArch

Микроархитектура

Функциональная верификация процессорных ядер



ISA

Архитектура Набора Команд

x86

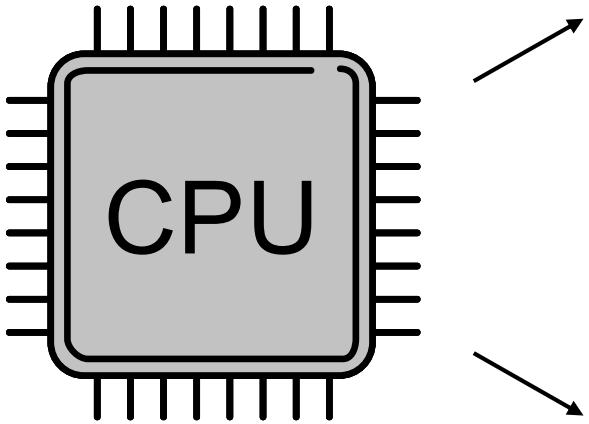
arm

RISC-V®

µArch

Микроархитектура

Функциональная верификация процессорных ядер



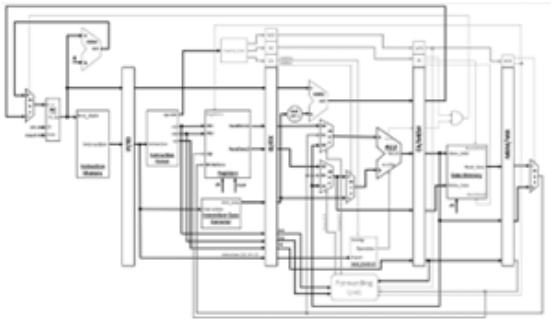
ISA

Архитектура Набора Команд

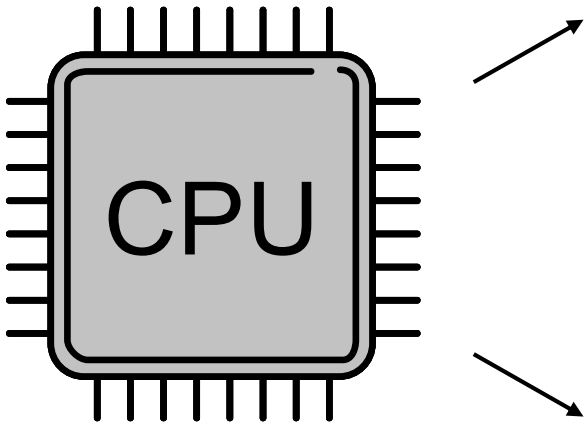
x86 arm RISC-V®

µArch

Микроархитектура



Функциональная верификация процессорных ядер



ISA

Архитектура Набора Команд

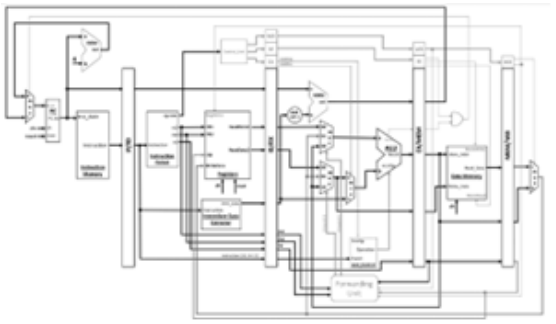
x86

arm

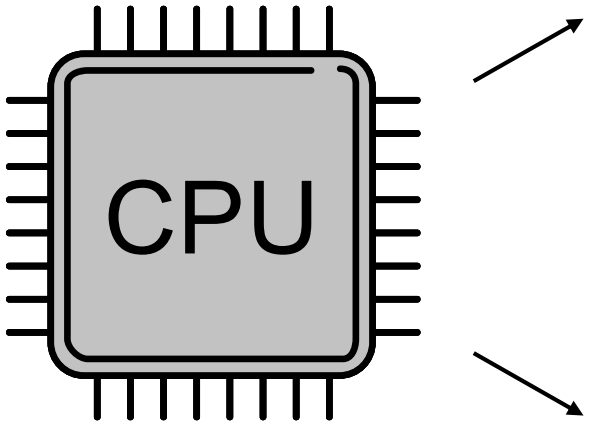
RISC-V®

µArch

Микроархитектура



Функциональная верификация процессорных ядер



ISA

Архитектура Набора Команд

The RISC-V Instruction Set Manual
Volume I: Unprivileged ISA
Document Version 20191213

Intel® 64 and IA-32 Architectures
Software Developer's Manual

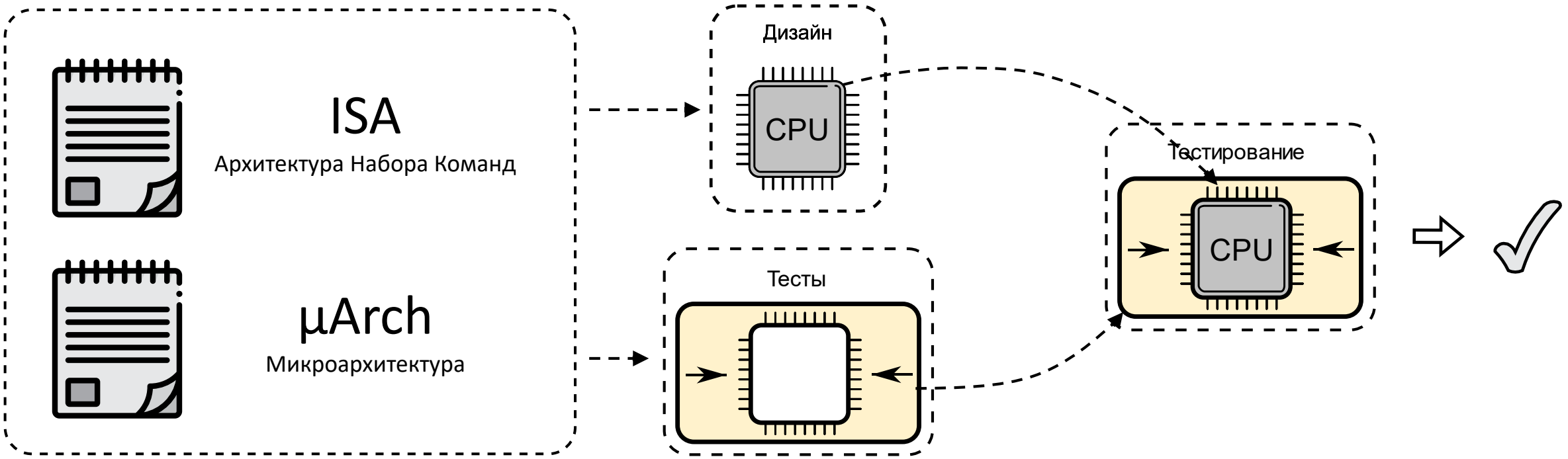
Volume 1:
Basic Architecture

µArch

Микроархитектура

Внутренняя документация

Функциональная верификация процессорных ядер



Особенности верификации RISC-V® ядер

- Открытая и свободно распространяемая.

Особенности верификации RISC-V® ядер

- Открытая и свободно распространяемая.

Особенности верификации RISC-V® ядер

- Большое количество примеров и возможностей для обмена опытом;
- Возможность переиспользования.

Особенности верификации RISC-V® ядер

- Гибкая и расширяемая.

- RV64IMACV_Zicsr_Zba_Zbb_Zbkc_Xbar_Xfoo

Расширение	Описание	Инструкции
RV32I	Базовый 32-битный набор	48
RV32E	RV32I с уменьшенным количеством регистров	RV32I
RV64I	Базовый 64-битный набор	14
RV128I	Базовый 128-битный набор	14

M	Zicsr
A	B
C	X
V	

Особенности верификации RISC-V® ядер

- Гибкая и расширяемая.

- RV64IMACV_Zicsr_Zba_Zbb_Zbkc_Xbar_Xfoo

Особенности верификации RISC-V® ядер

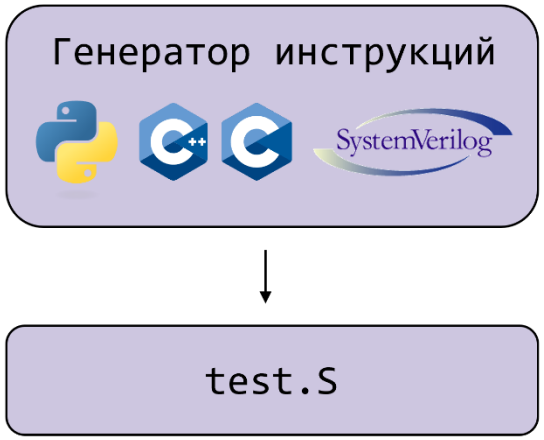
- Гибкая и расширяемая.

- Огромное количество возможных реализаций;
- Каждая реализация имеет:
  - свою микроархитектуру;
  - свой поддерживаемый набор инструкций.

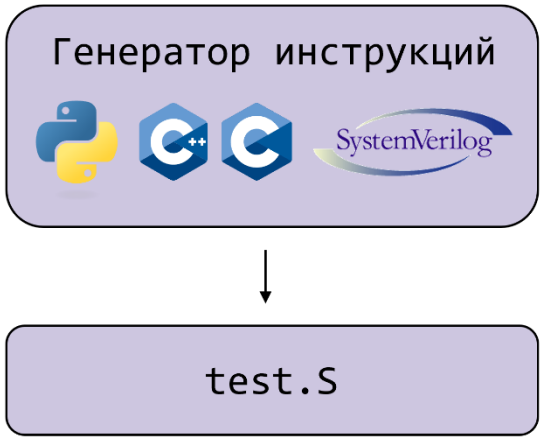
Существующие подходы к верификации RISC-V ядер

- "Hello world!"
- Самопроверка
- Сравнение файлов трассировки
- Step-and-Compare
- Async-Step-and-Compare

Step-and-Compare подход к верификации RISC-V

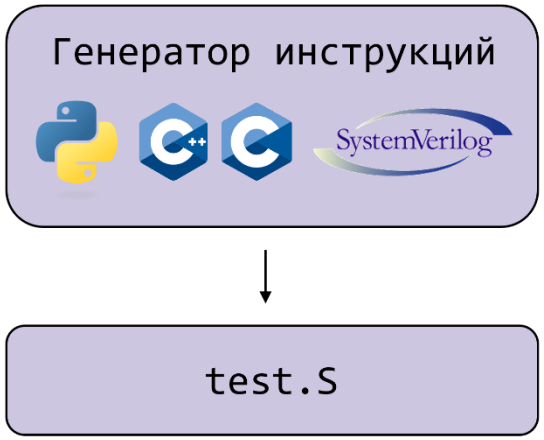


Step-and-Compare подход к верификации RISC-V

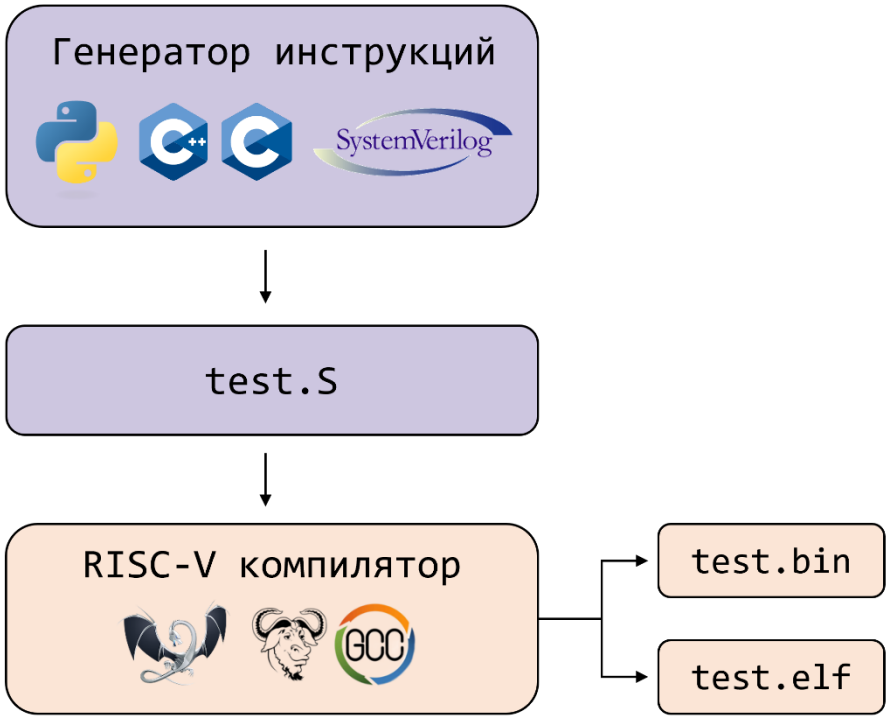


- github.com/chipsalliance/riscv-dv
- github.com/openhwgroup/force-riscv
- forge.ispras.ru/projects/microtesk-riscv
- gitlab.com/shaktiproject/tools/aapg
- github.com/syntacore/snippy

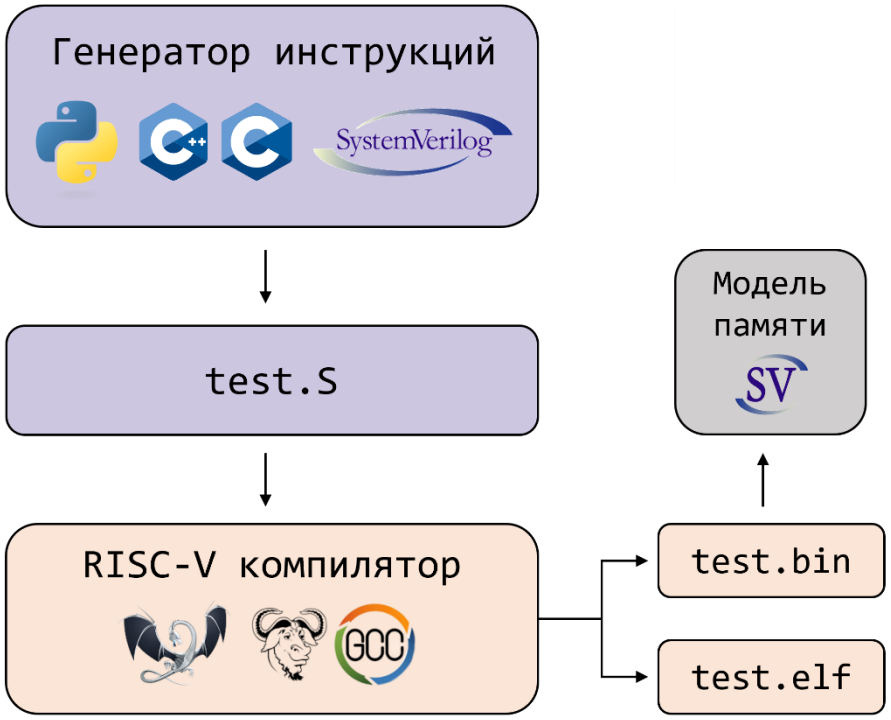
Step-and-Compare подход к верификации RISC-V



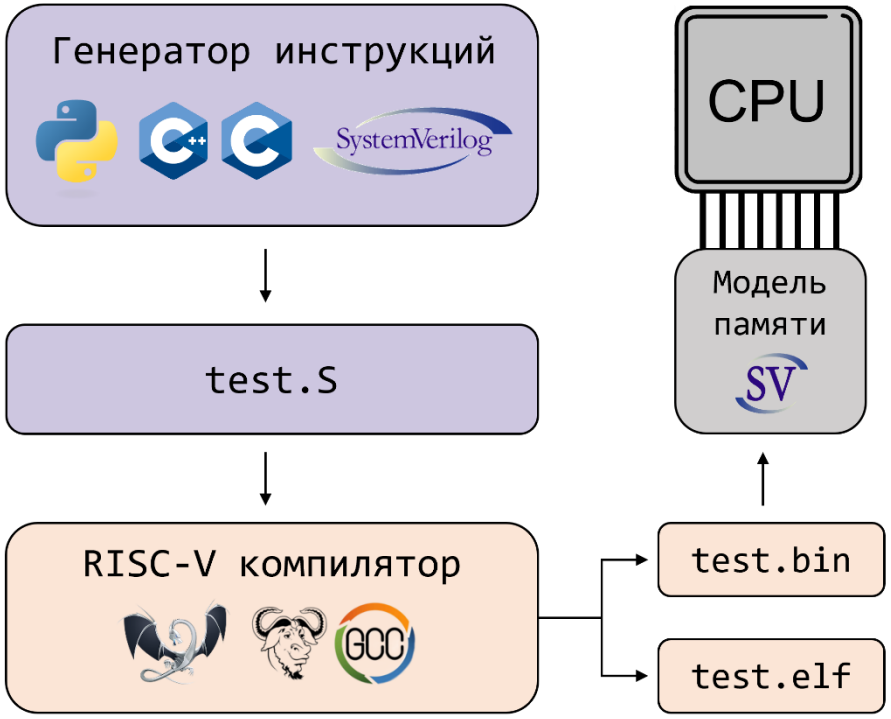
Step-and-Compare подход к верификации RISC-V



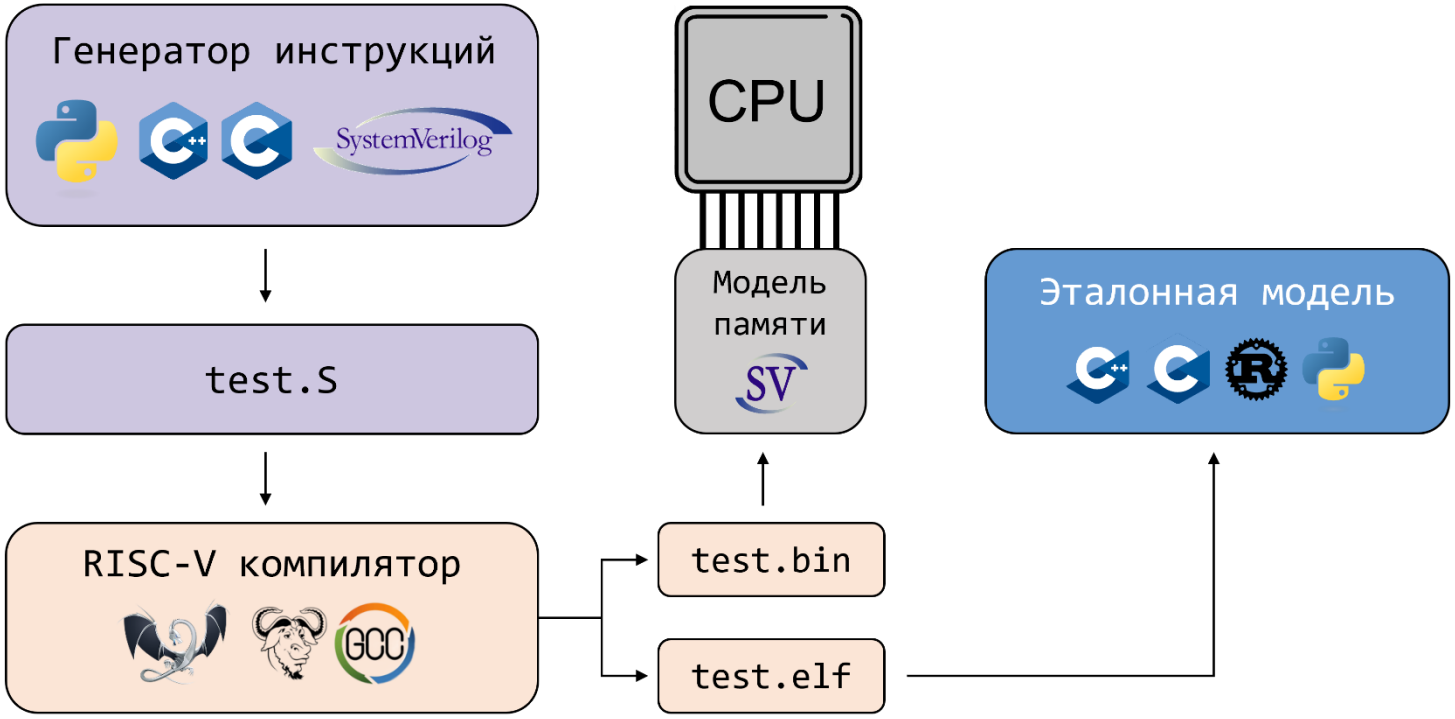
Step-and-Compare подход к верификации RISC-V



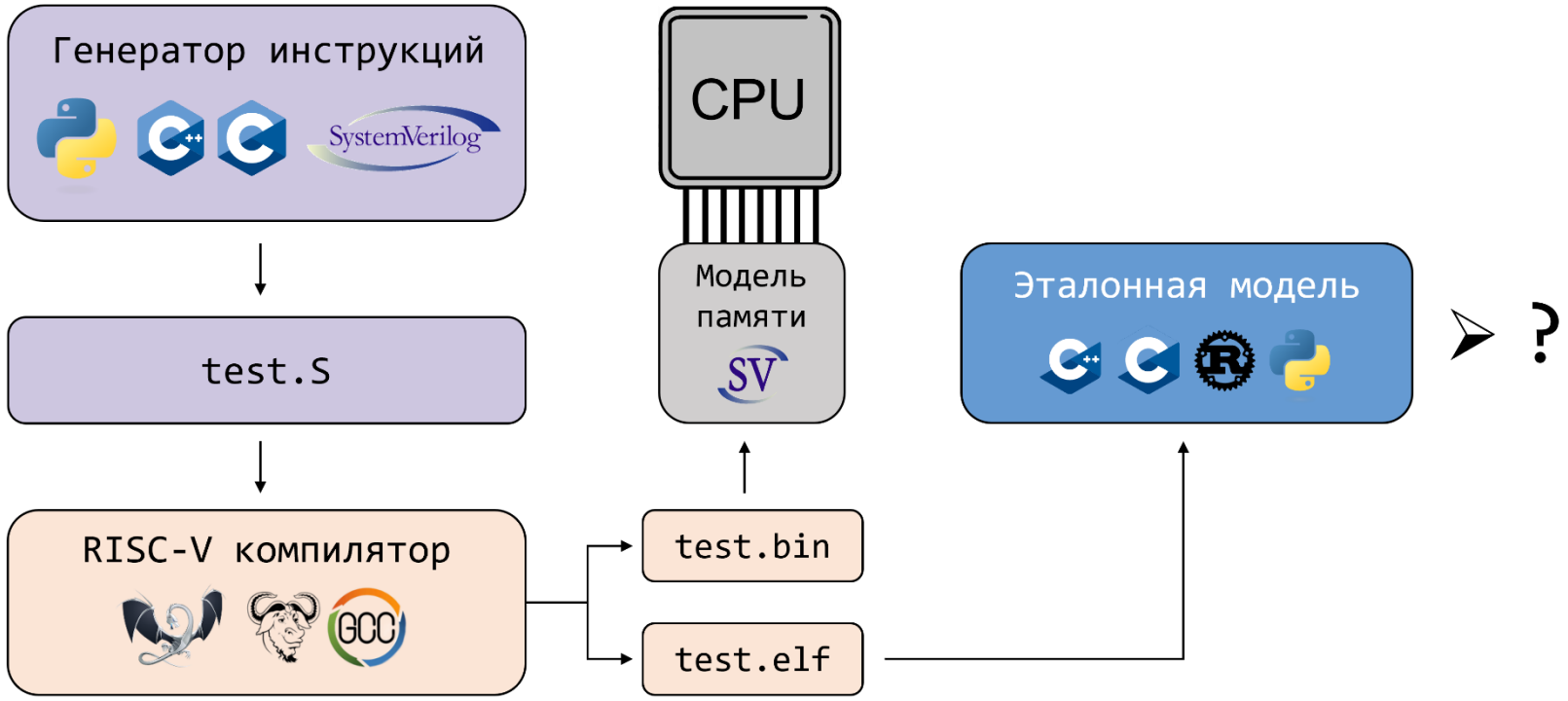
Step-and-Compare подход к верификации RISC-V



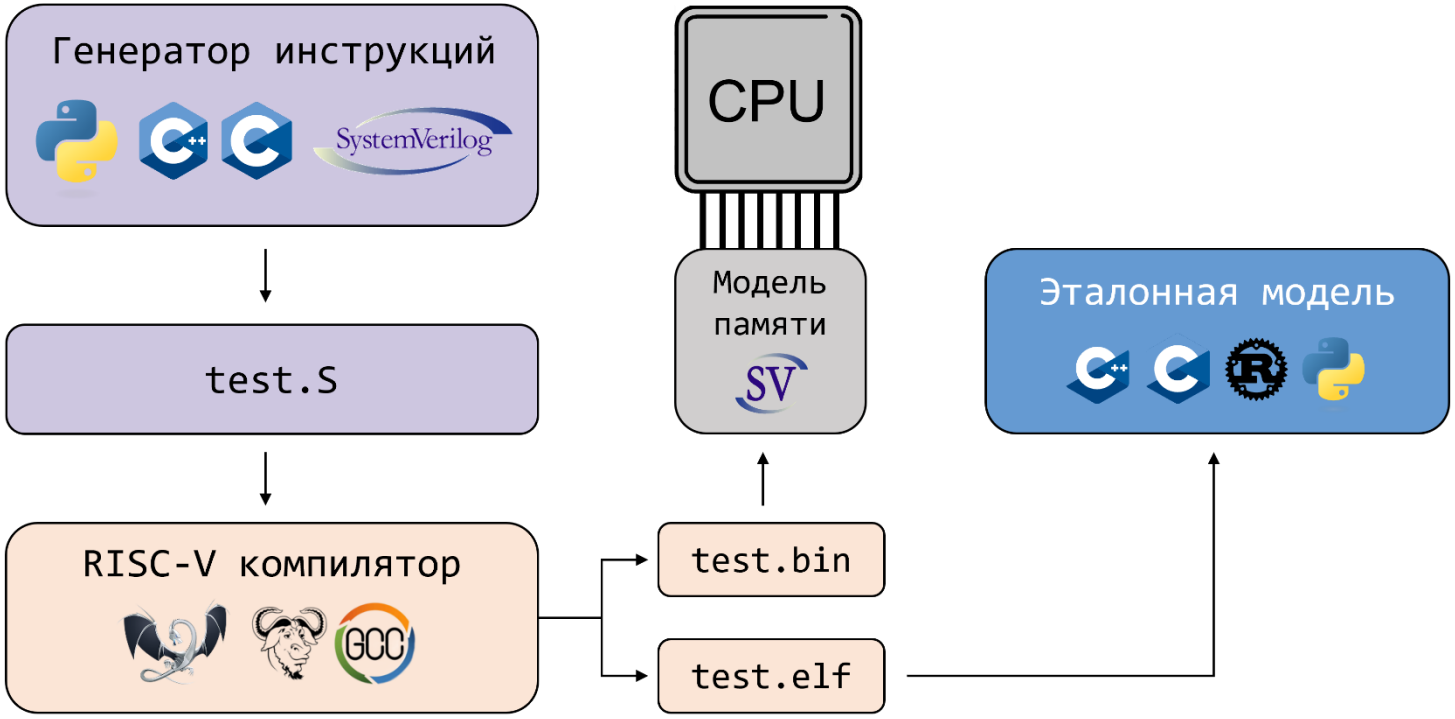
Step-and-Compare подход к верификации RISC-V



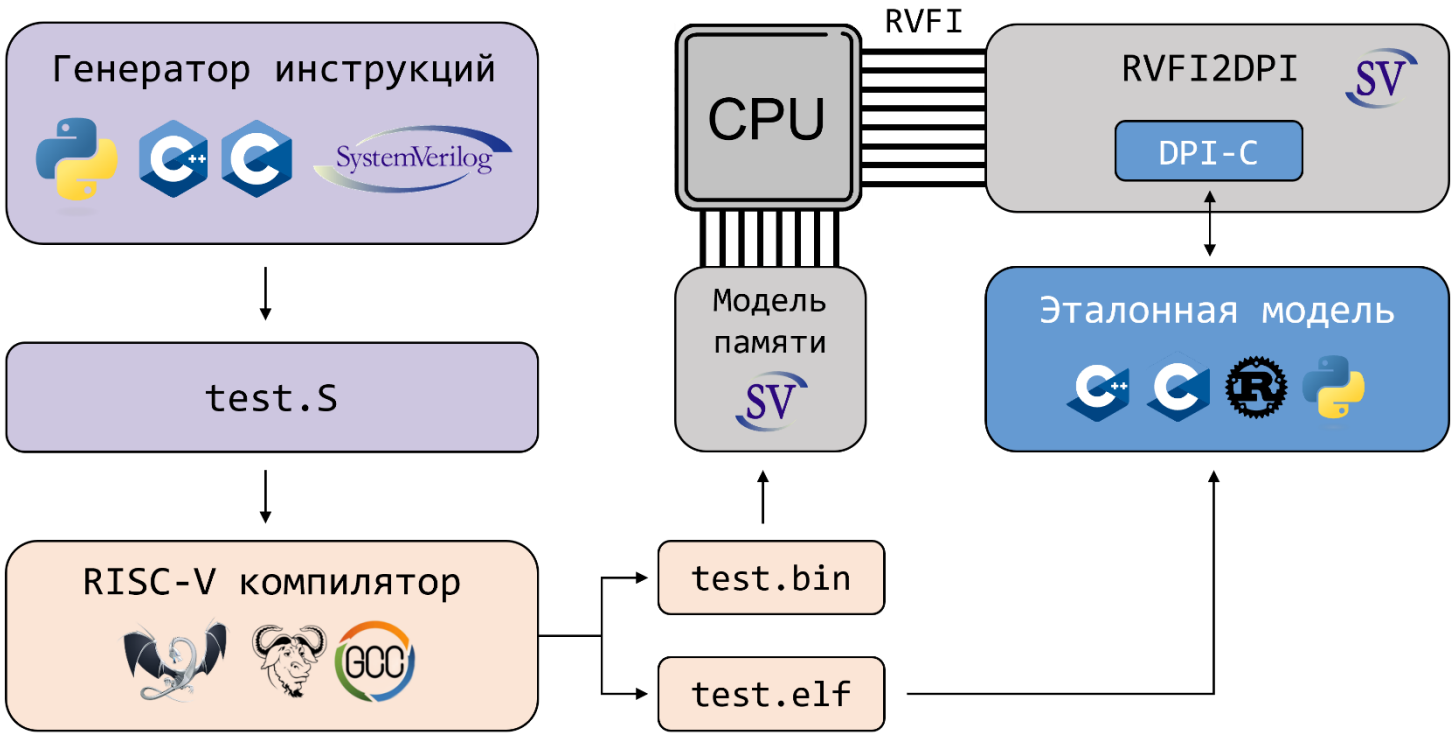
Step-and-Compare подход к верификации RISC-V



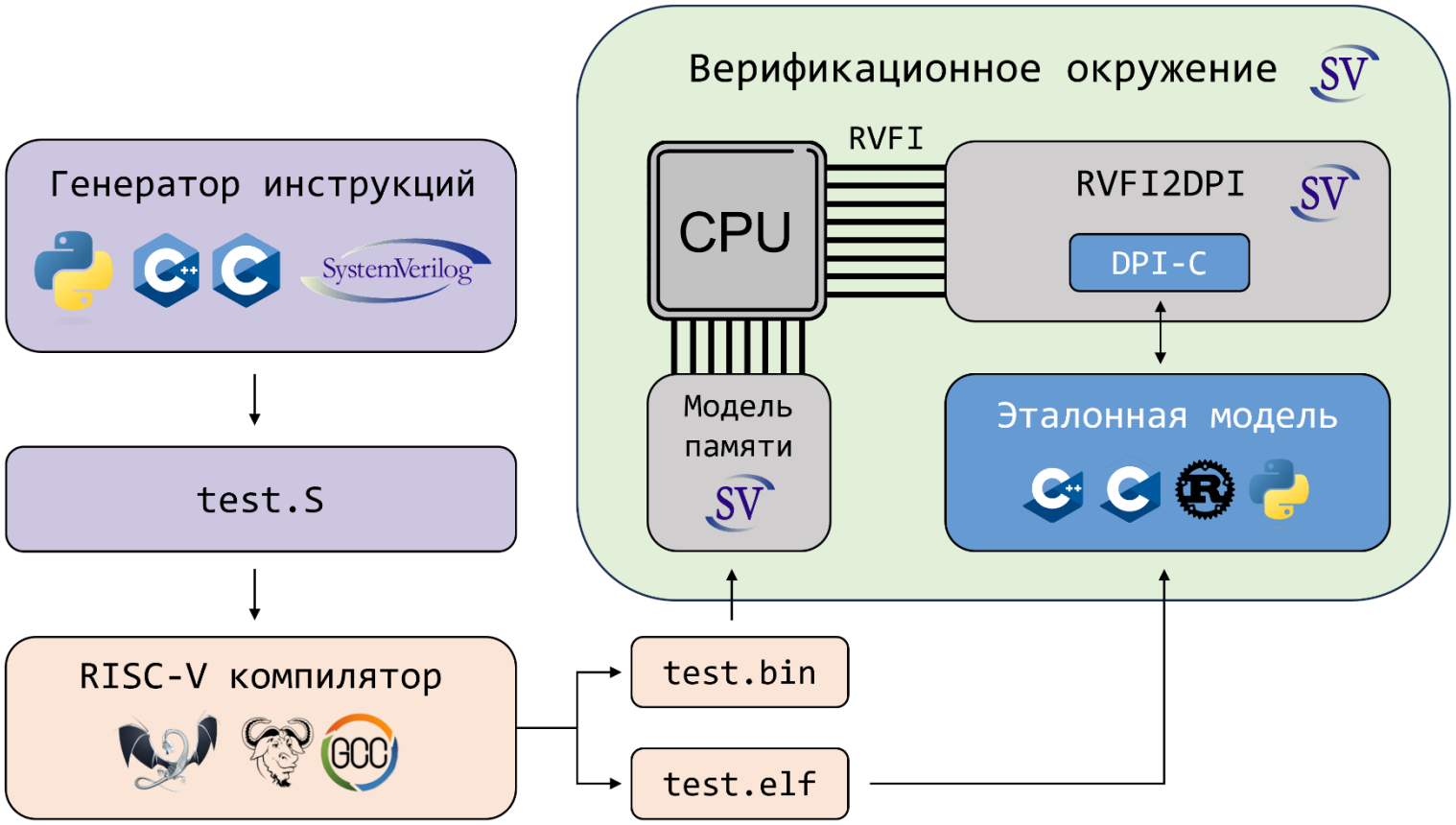
Step-and-Compare подход к верификации RISC-V



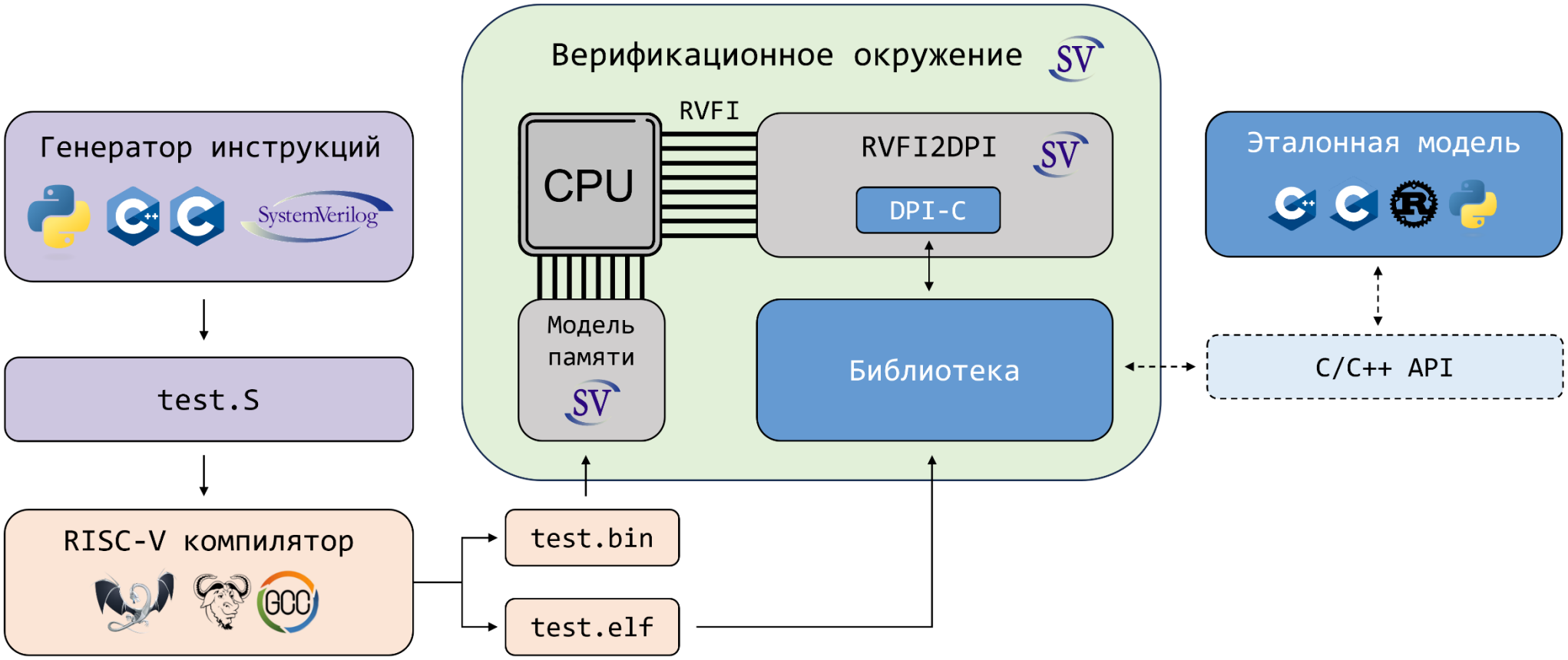
Step-and-Compare подход к верификации RISC-V



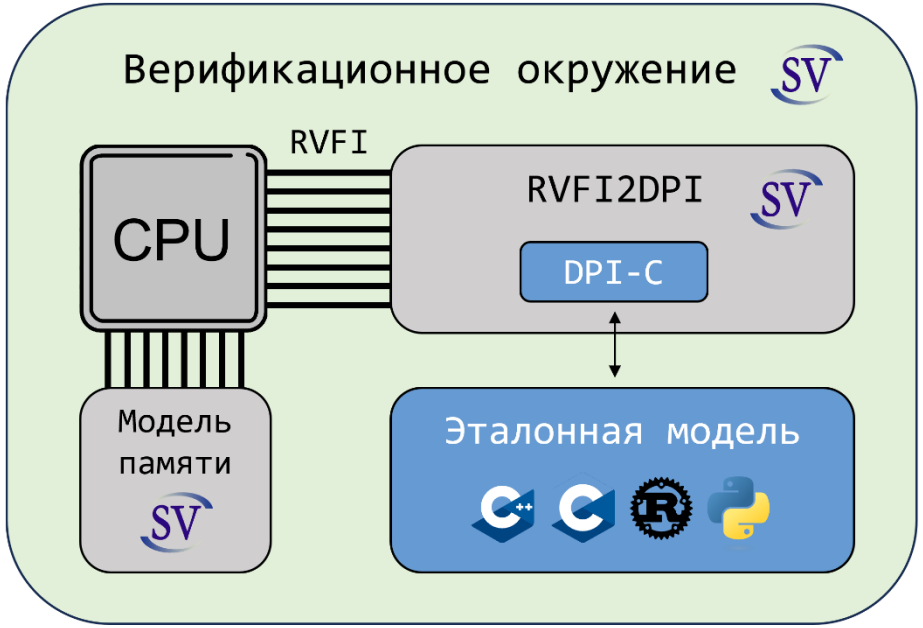
Step-and-Compare подход к верификации RISC-V



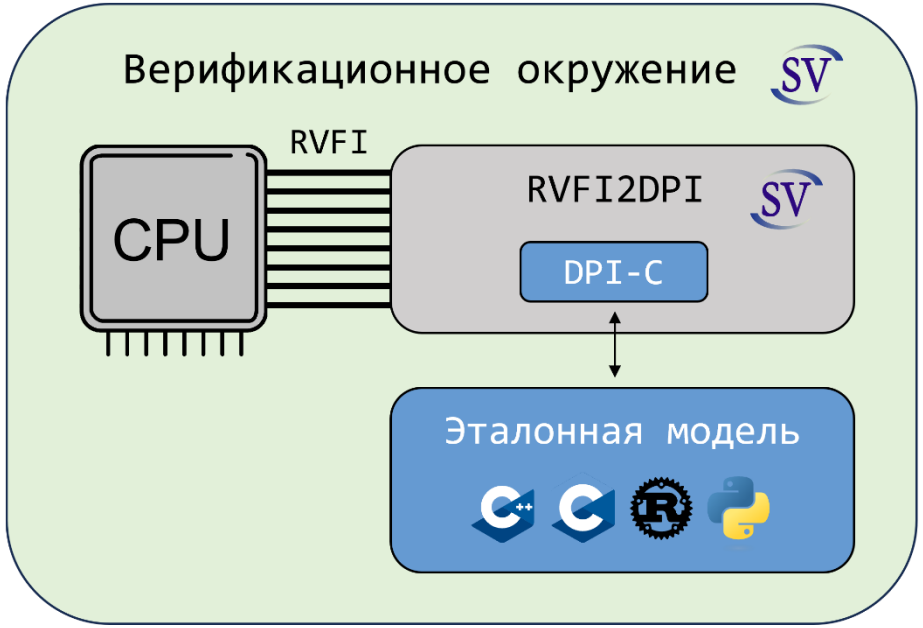
Step-and-Compare подход к верификации RISC-V



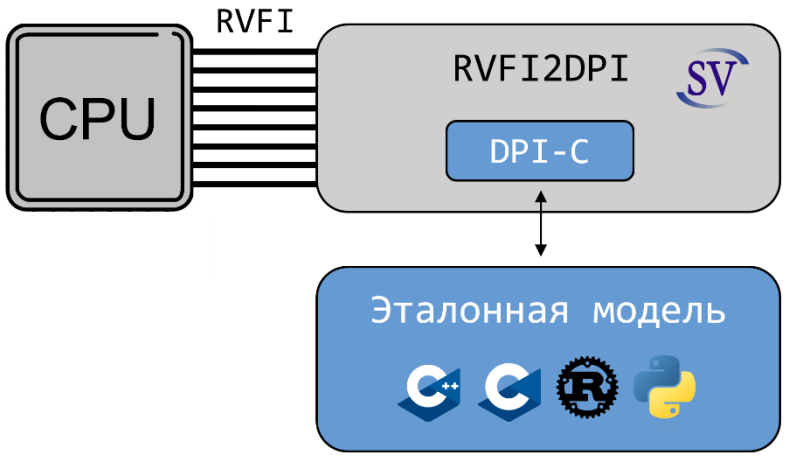
Step-and-Compare подход к верификации RISC-V



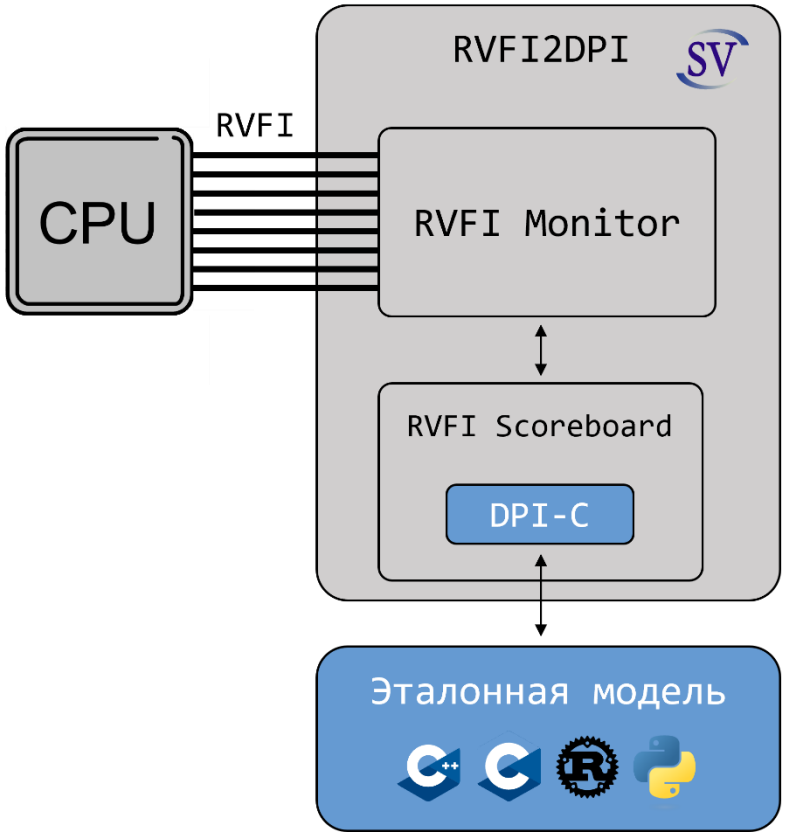
Step-and-Compare подход к верификации RISC-V



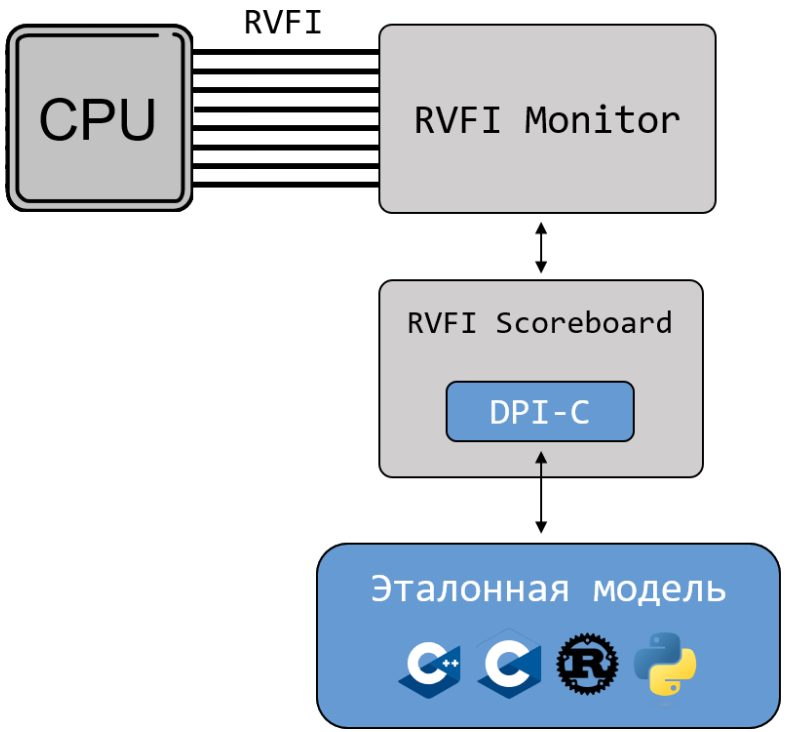
Step-and-Compare подход к верификации RISC-V



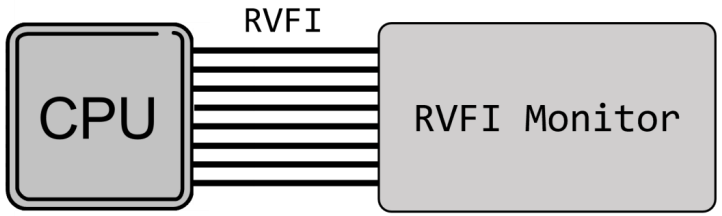
Step-and-Compare подход к верификации RISC-V



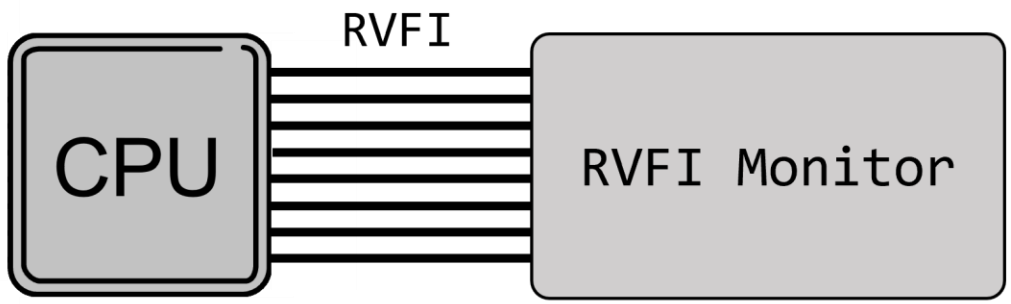
Step-and-Compare подход к верификации RISC-V



Step-and-Compare подход к верификации RISC-V

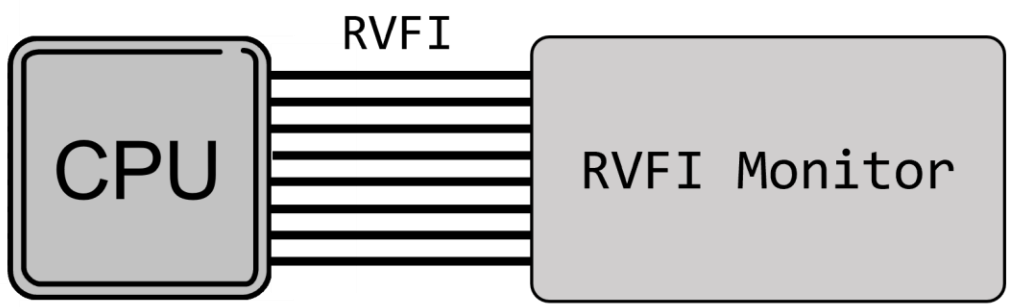


Step-and-Compare подход к верификации RISC-V



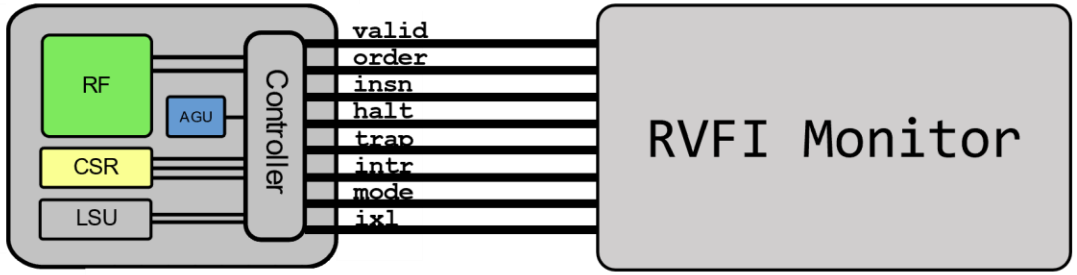
Step-and-Compare подход к верификации RISC-V

- Интерфейс передачи информации о внутреннем состоянии ядра;
- Изначально разработан Symbiotic EDA для формальной верификации.

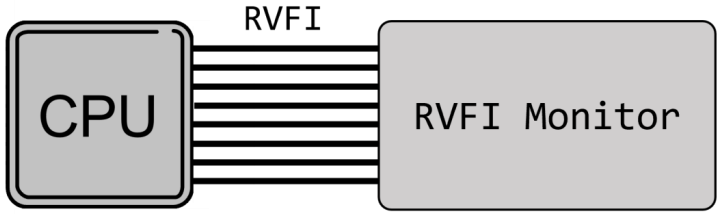


Step-and-Compare подход к верификации RISC-V

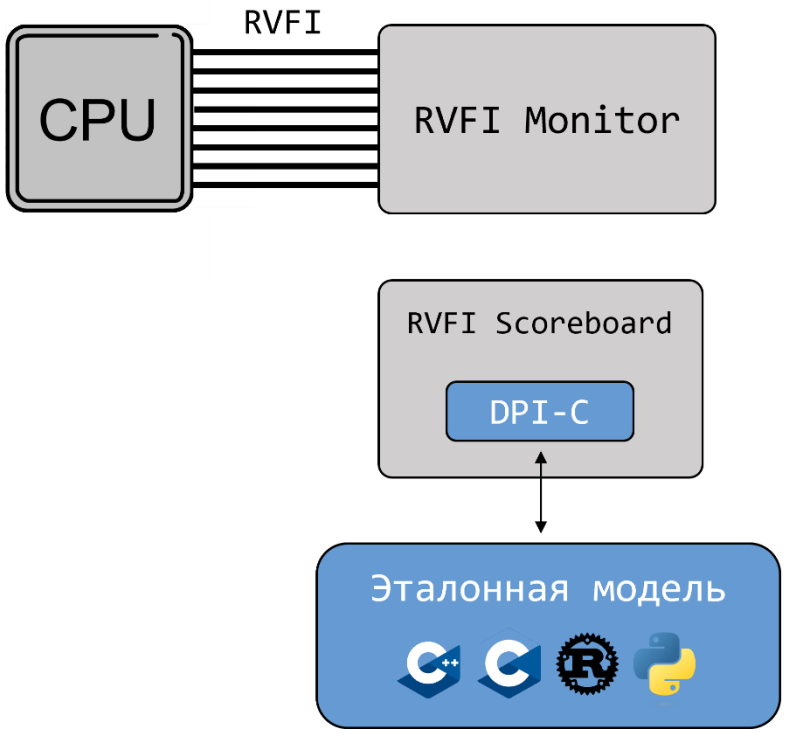
- Интерфейс передачи информации о внутреннем состоянии ядра;
- Изначально разработан Symbiotic EDA для формальной верификации.



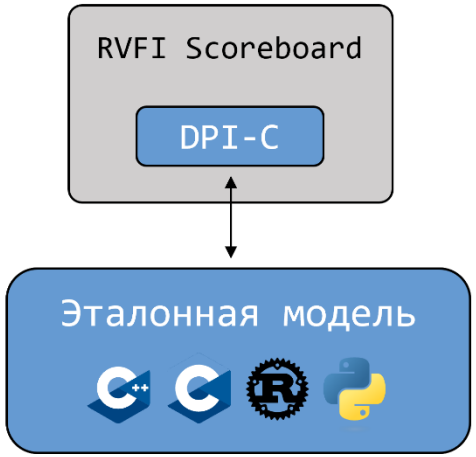
Step-and-Compare подход к верификации RISC-V



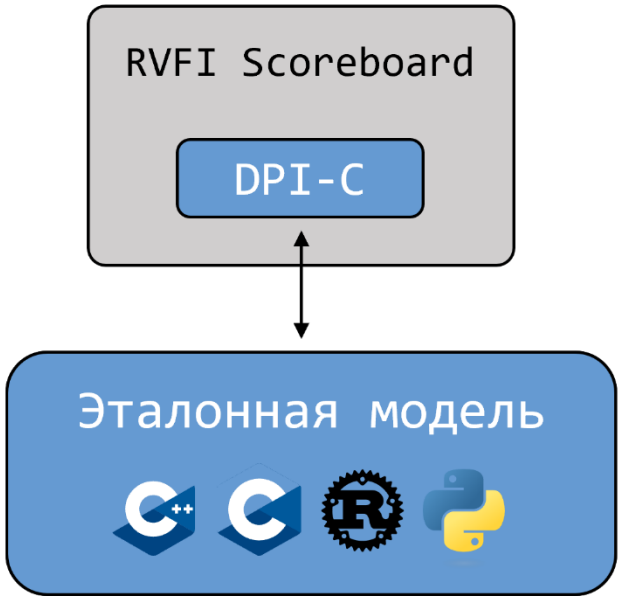
Step-and-Compare подход к верификации RISC-V



Step-and-Compare подход к верификации RISC-V



Step-and-Compare подход к верификации RISC-V

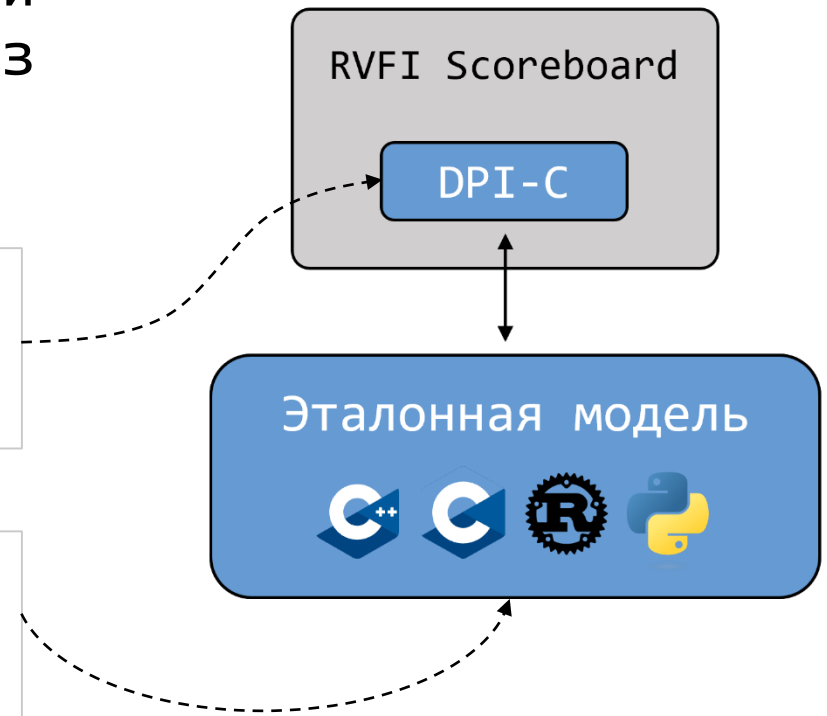


Step-and-Compare подход к верификации RISC-V

- DPI-C/C++ – механизм, позволяющий вызывать C/C++ функции напрямую из SystemVerilog в ходе симуляции.

```
import "DPI-C" function bit [31:0] hammer_get_gpr (
  chandle   hammer,
  bit [4:0] gpr_id
);
```

```
extern svBitVecVal hammer_get_gpr (
        void*        hammer,
  const svBitVecVal* gpr_id
);
```

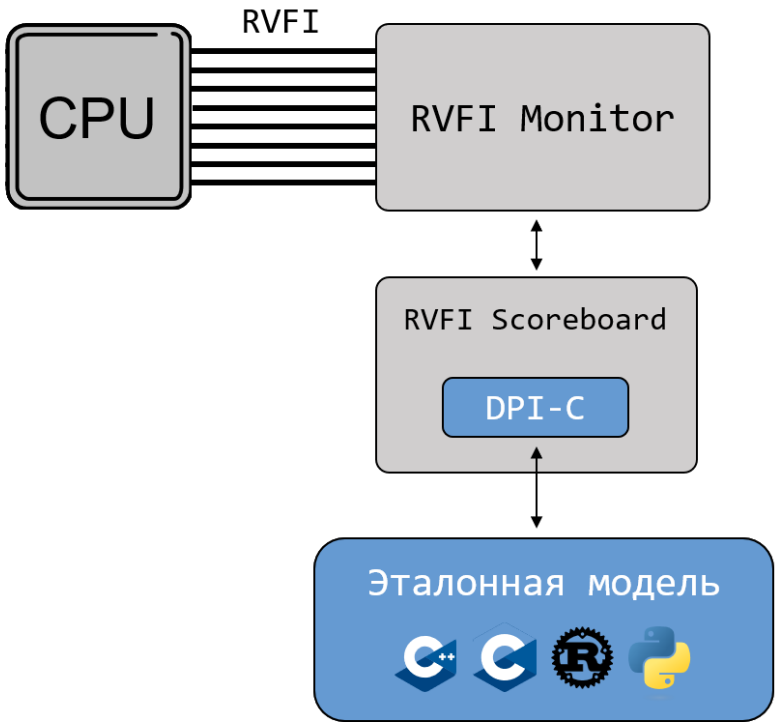


Step-and-Compare подход к верификации RISC-V

- Пример взаимодействия с моделью:

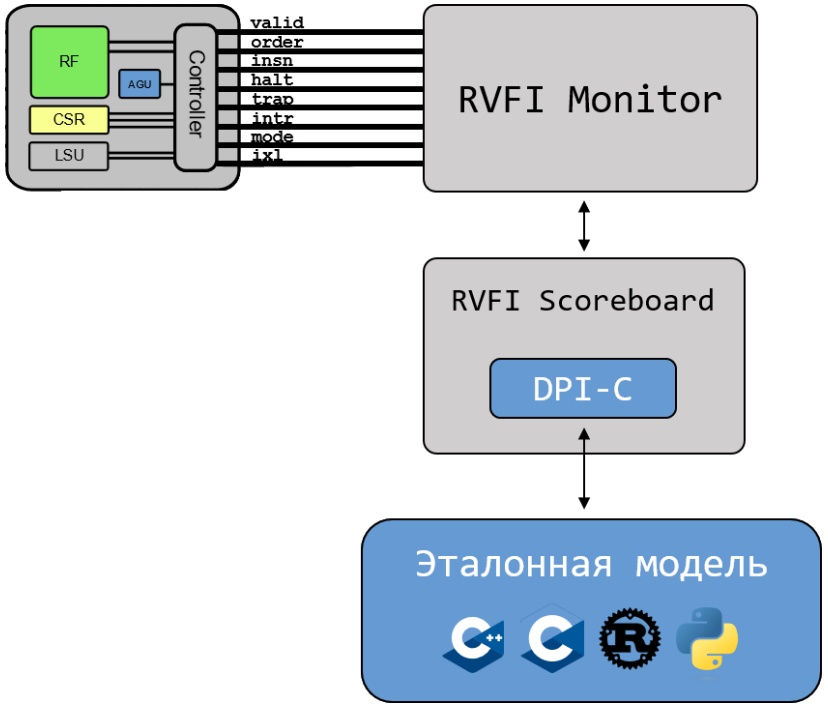
Step-and-Compare подход к верификации RISC-V

- Пример взаимодействия с моделью:



Step-and-Compare подход к верификации RISC-V

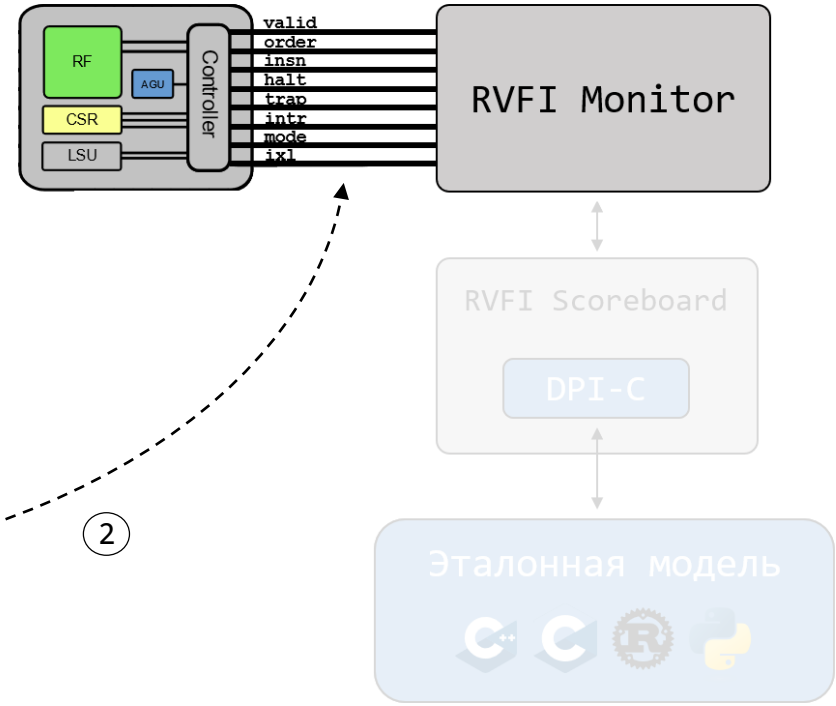
- Пример взаимодействия с моделью:



Step-and-Compare подход к верификации RISC-V

- Пример взаимодействия с моделью:

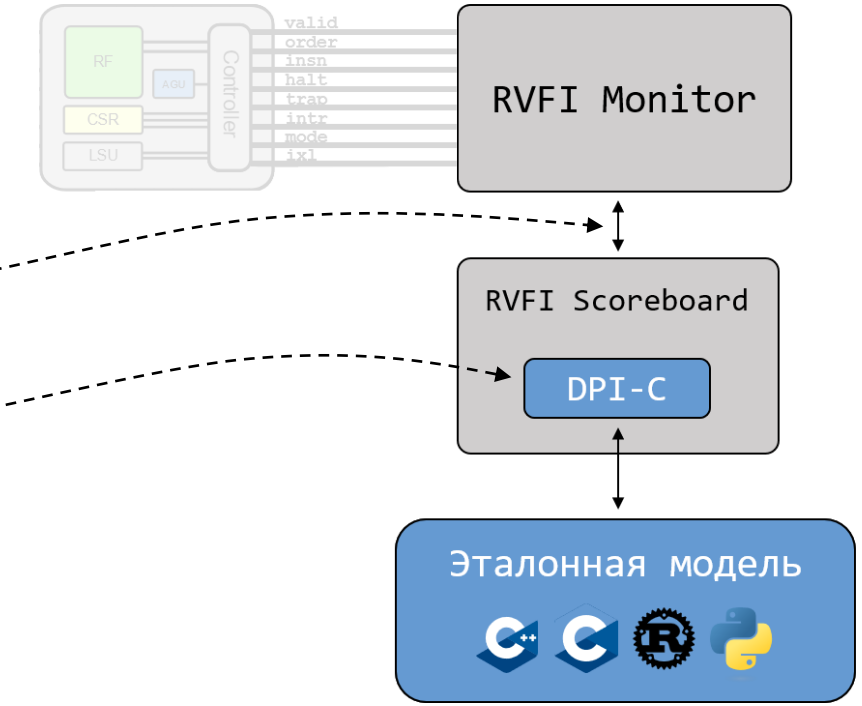
```
virtual task run();
    forever begin
        vif.wait_clks(1);
        get_and_put();
    end
endtask

virtual task get_and_put();
    miriscv_mem_item t = new();
    get_data(t); mbx.put(t);
endtask

virtual task get_data(miriscv_mem_item t);
    vif.get_bus_status (t);
endtask
```



Step-and-Compare подход к верификации RISC-V

- Пример взаимодействия с моделью:

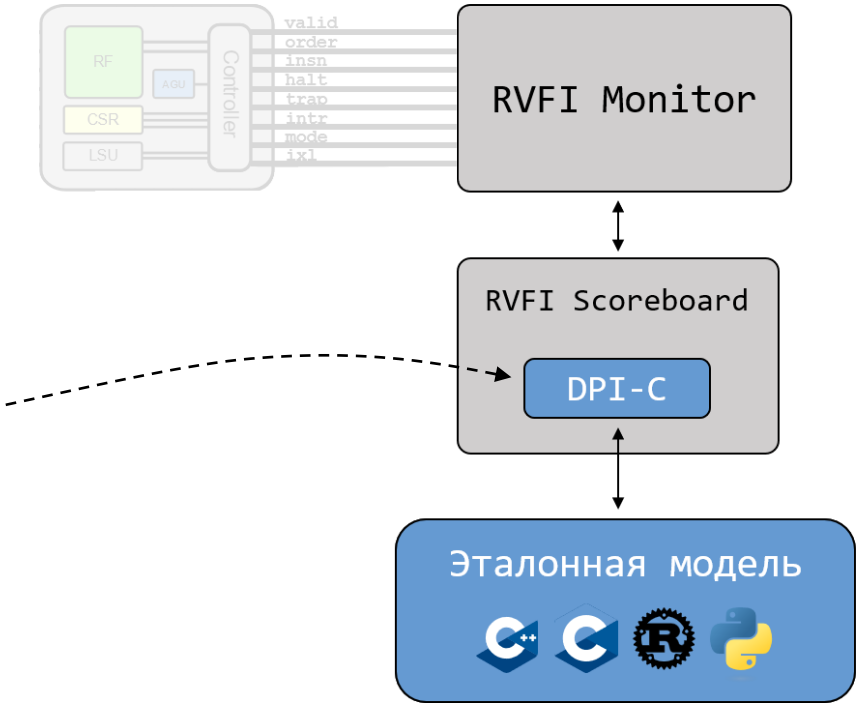
```
virtual task run();
  miriscv_rvfi_item t; bit result;

  forever begin

    rvfi_mbx.get(t);

    result = check_pc_and_instr(t);
    hammer_single_step(hammer);
    if( result ) void'(check_rd(t));

    retire_cnt = retire_cnt + 1;
  end

endtask
```



Step-and-Compare подход к верификации RISC-V

- Пример взаимодействия с моделью:

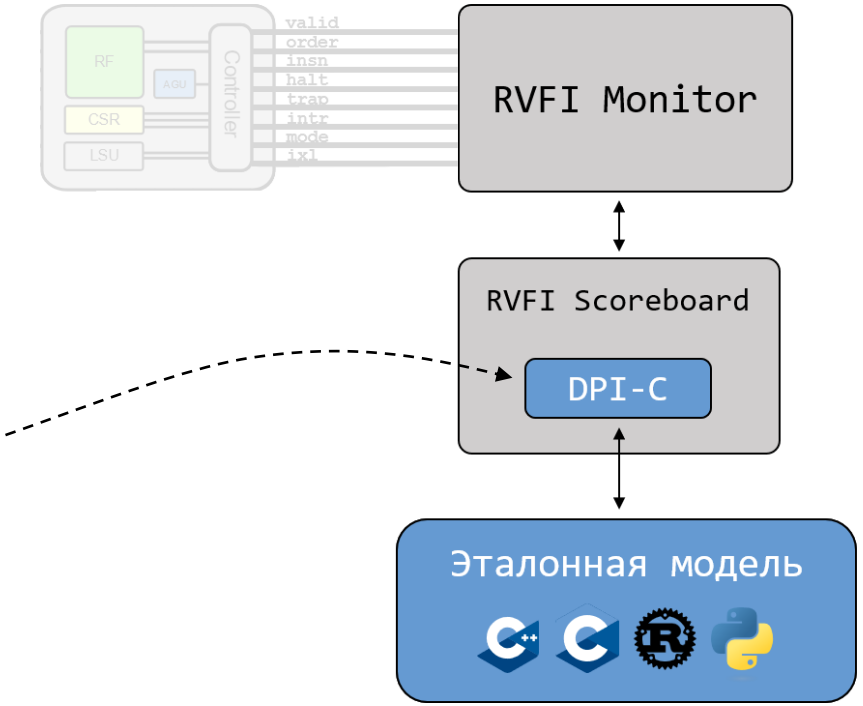
```
virtual task run();
  miriscv_rvfi_item t; bit result;

  forever begin

    rvfi_mbx.get(t);

    result = check_pc_and_instr(t);
    hammer_single_step(hammer);
    if( result ) void'(check_rd(t));

    retire_cnt = retire_cnt + 1;
  end

endtask
```



Step-and-Compare подход к верификации RISC-V

- Пример взаимодействия с моделью:

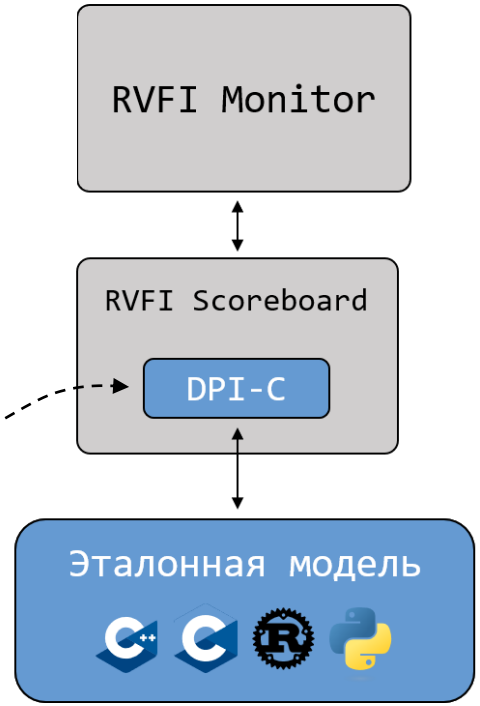
```
virtual task run();
  miriscv_rvfi_item t; bit result;

  forever begin

    rvfi_mbx.get(t);

    result = check_pc_and_instr(t);
    hammer_single_step(hammer);
    if( result ) void'(check_rd(t));

    retire_cnt = retire_cnt + 1;
  end

endtask
```



Step-and-Compare подход к верификации RISC-V

- Пример взаимодействия с моделью:

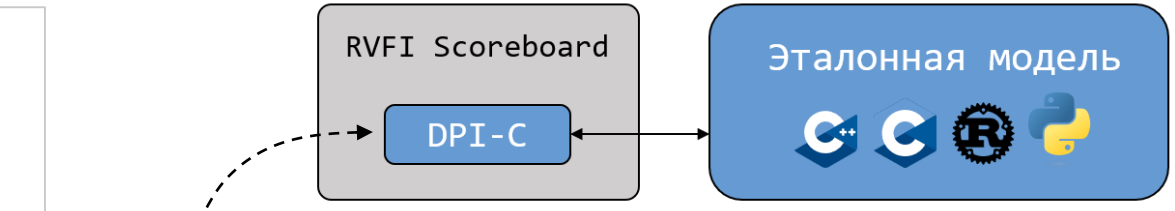
```
virtual task run();
  miriscv_rvfi_item t; bit result;

  forever begin

    rvfi_mbx.get(t);

    result = check_pc_and_instr(t);
    hammer_single_step(hammer);
    if( result ) void'(check_rd(t));

    retire_cnt = retire_cnt + 1;
  end

endtask
```



Step-and-Compare подход к верификации RISC-V

- Пример взаимодействия с моделью:

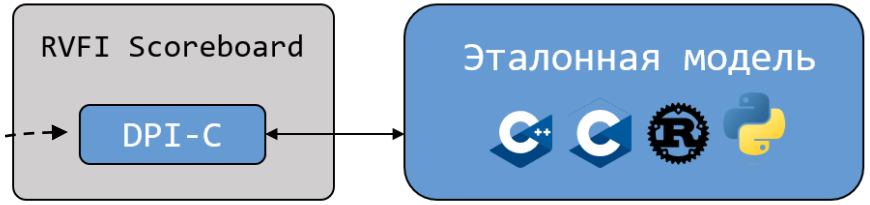
```
virtual task run();
  miriscv_rvfi_item t; bit result;

  forever begin

    rvfi_mbx.get(t);

    result = check_pc_and_instr(t);
    hammer_single_step(hammer);
    if( result ) void'(check_rd(t));

    retire_cnt = retire_cnt + 1;
  end

endtask
```



Step-and-Compare подход к верификации RISC-V

- Пример взаимодействия с моделью:

```
virtual task run();
  miriscv_rvfi_item t; bit result;

  forever begin

    rvfi_mbx.get(t);

    result = check_pc_and_instr(t);
    hammer_single_step(hammer);
    if( result ) void'(check_rd(t));

    retire_cnt = retire_cnt + 1;
  end

endtask
```

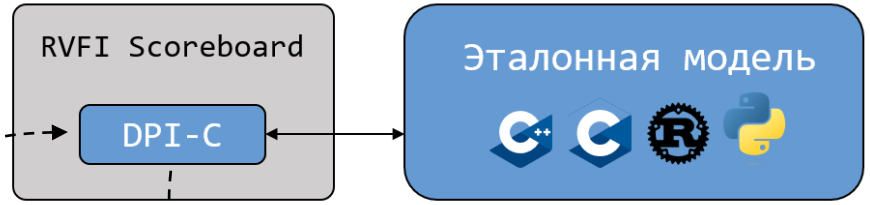



```
virtual function bit check_rd(...);
  bit result = 1; string msg; bit [31:0] rd;

  rd = hammer_get_gpr(hammer, t.rvfi_rd_addr);

  if( hammer_rd !== t.rvfi_rd_wdata ) begin
    msg = "\nRD mismatch! "; ...
    result = 0;...
  end ...
endfunction
```

Step-and-Compare подход к верификации RISC-V

- Пример взаимодействия с моделью:

```
virtual task run();
  miriscv_rvfi_item t; bit result;

  forever begin

    rvfi_mbx.get(t);

    result = check_pc_and_instr(t);
    hammer_single_step(hammer);
    if( result ) void'(check_rd(t));

    retire_cnt = retire_cnt + 1;
  end

endtask
```




```
virtual function bit check_rd(...);
  bit result = 1; string msg; bit [31:0] rd;

  rd = hammer_get_gpr(hammer, t.rvfi_rd_addr);

  if( hammer_rd !== t.rvfi_rd_wdata ) begin
    msg = "\nRD mismatch! "; ...
    result = 0;...
  end ...
endfunction
```

Step-and-Compare с использованием открытого ПО

- Открытый ознакомительный курс по верификации RISC-V ядер:
  - Использование исключительно открытого ПО
  - Предоставление виртуальной машины
  - Теоретическая и практическая части
  - Использование разнообразных методик
  - Наличие эталонных реализаций

Step-and-Compare с использованием открытого ПО

- Идея курса – обучение концепциям функциональной верификации RISC-V ядер, используемым в индустрии в настоящее время.

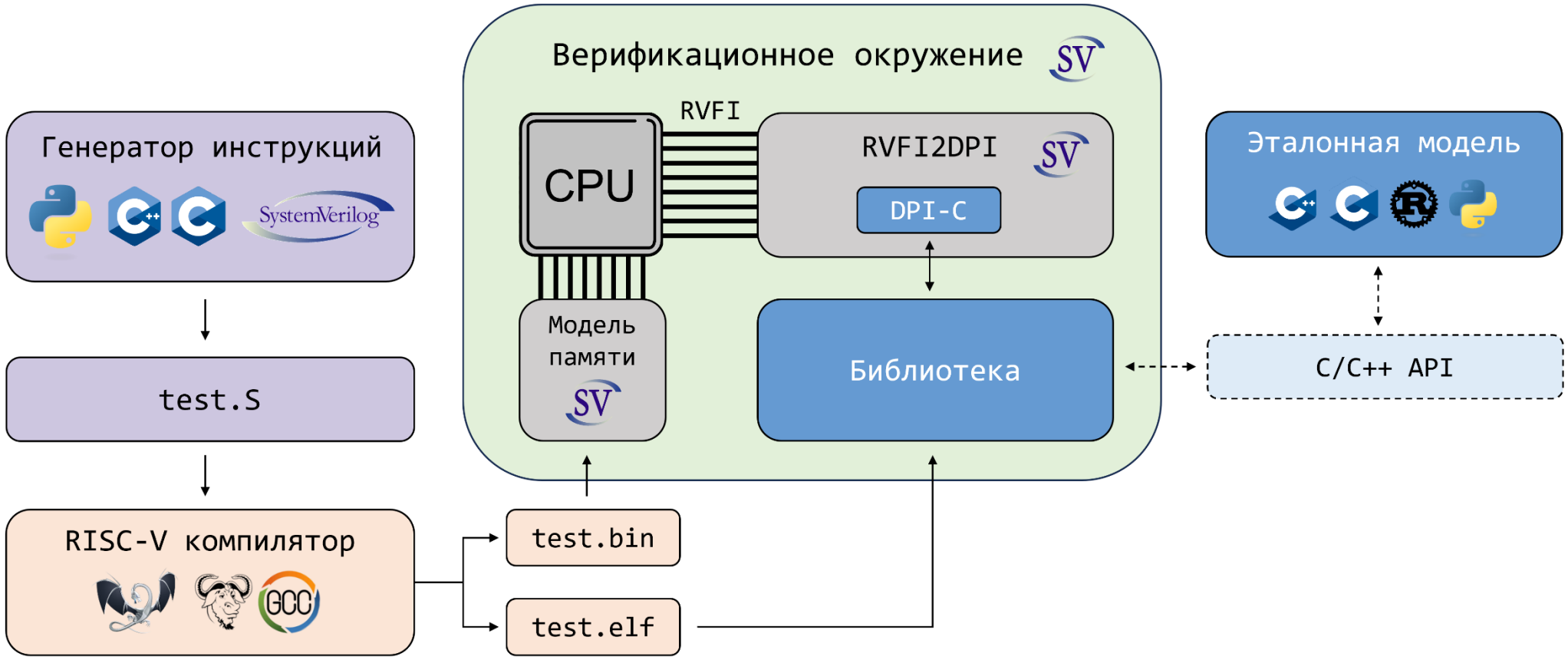
- Необходимо создать простейший пример для Step-and-Compare.
- К существующим примерам в открытом доступе много вопросов.
- В том числе к примерам в англоязычном пространстве.

Step-and-Compare с использованием открытого ПО

- Существующие примеры:
  - github.com/lowRISC/ibex/dv/uvm/core_ibex
  - github.com/openhwgroup/core-v-verif


- Оба излишне сложны для восприятия.
- В обоих используются коммерческие симуляторы.
- Один использует коммерческую эталонную модель от imperas.

Step-and-Compare с использованием открытого ПО

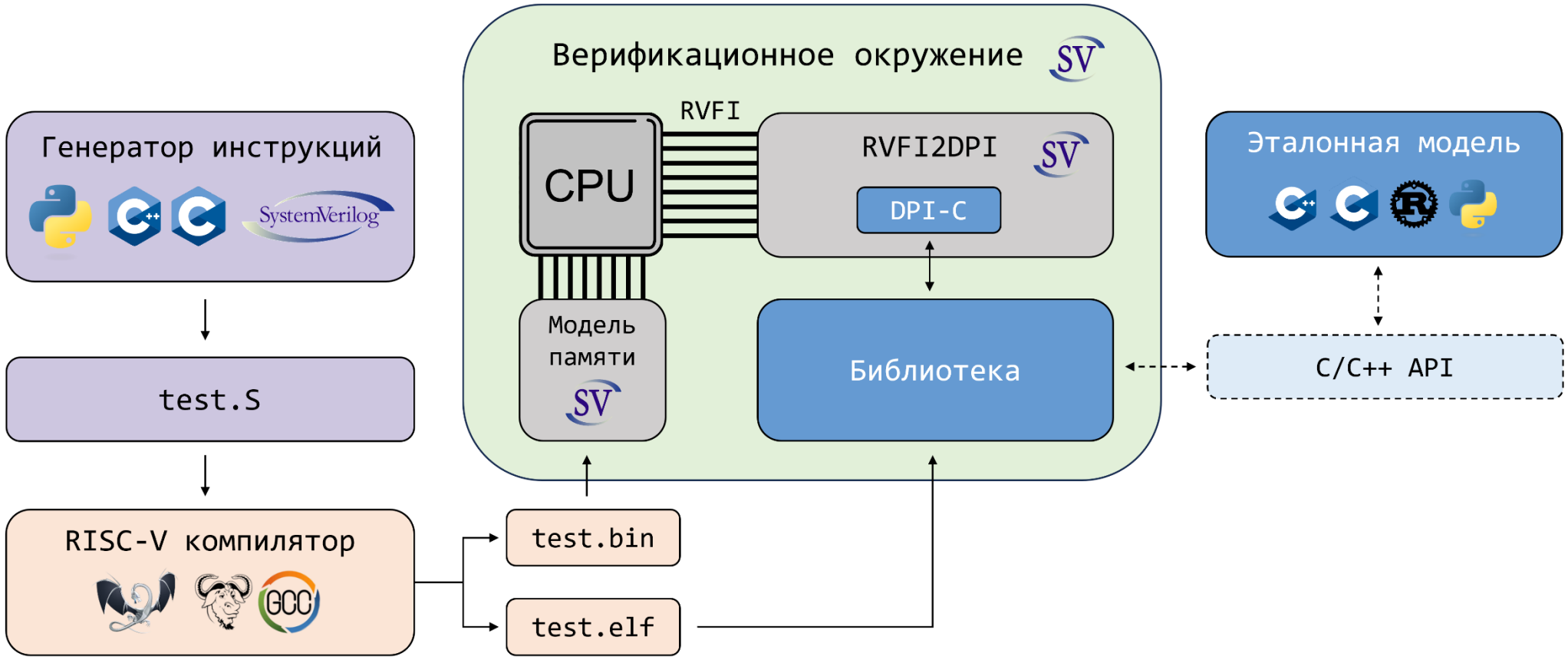


Step-and-Compare с использованием открытого ПО

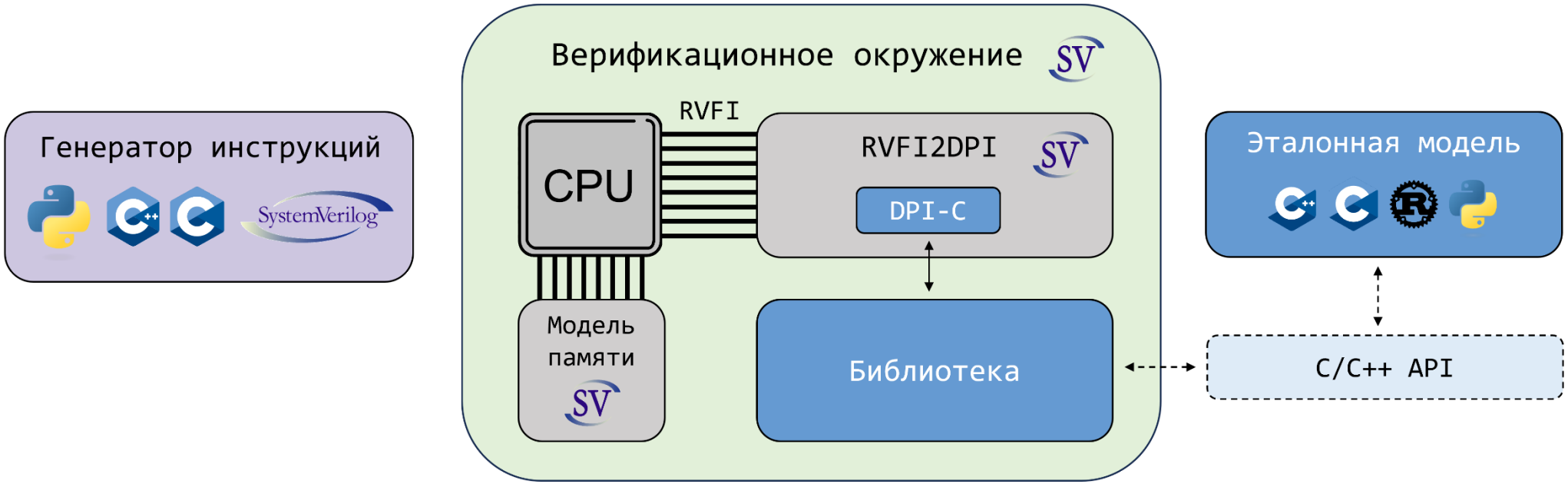
- Симулятор 

:
  - Активно разрабатывается;
  - Используется в индустрии;
  - Поддерживает SystemVerilog (но есть нюансы).

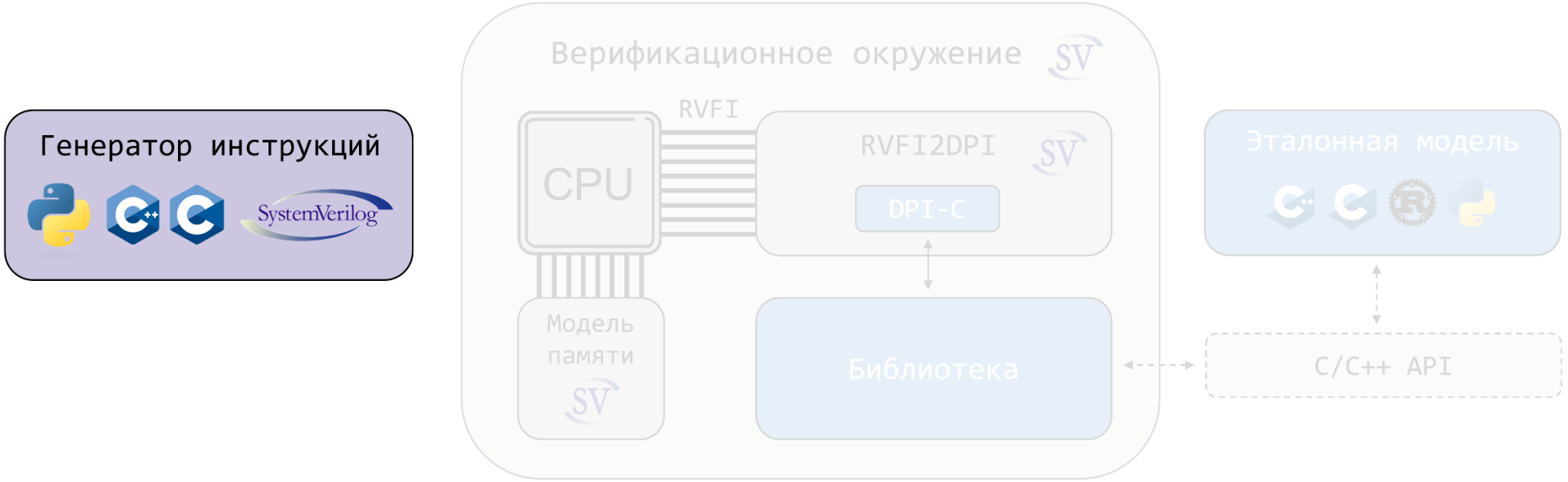
Step-and-Compare с использованием открытого ПО



Step-and-Compare с использованием открытого ПО



Step-and-Compare с использованием открытого ПО



Step-and-Compare с использованием открытого ПО

- Генератор случайных инструкций:
  - github.com/chipsalliance/riscv-dv
  - github.com/openhwgroup/force-riscv
  - forge.ispras.ru/projects/microtesk-riscv
  - gitlab.com/shaktiproject/tools/aapg
  - github.com/syntacore/snippy

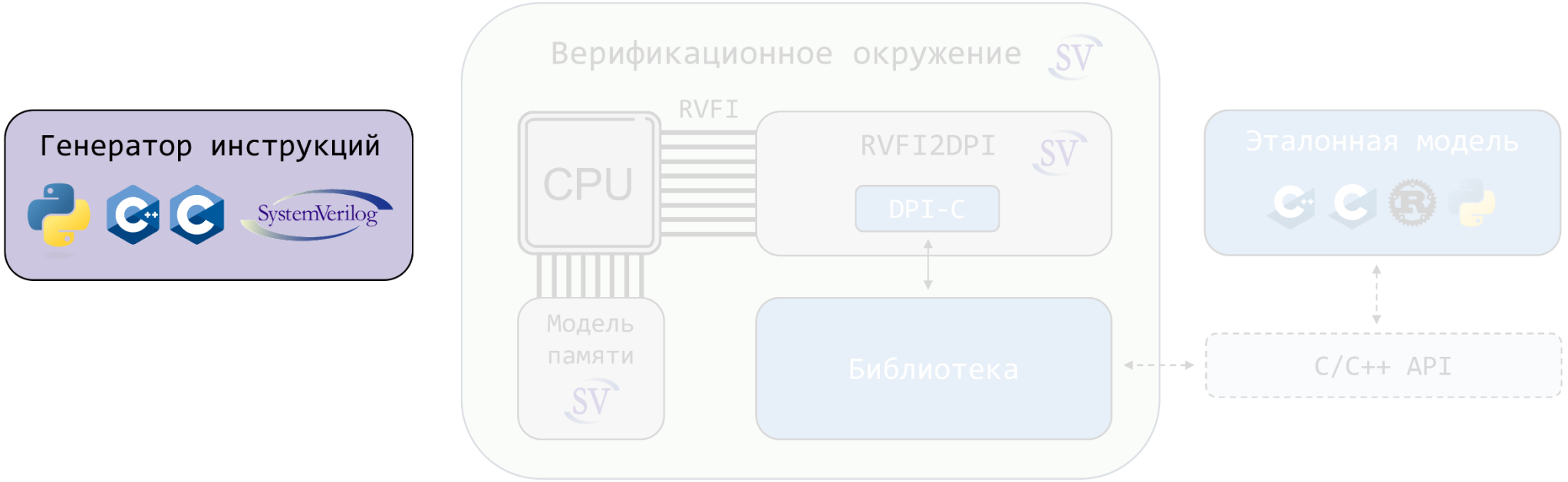
Step-and-Compare с использованием открытого ПО

- Генератор случайных инструкций:
  - github.com/chipsalliance/riscv-dv
  - github.com/openhwgroup/force-riscv
  - forge.ispras.ru/projects/microtesk-riscv
  - gitlab.com/shaktiproject/tools/aapg
  - github.com/syntacore/snippy

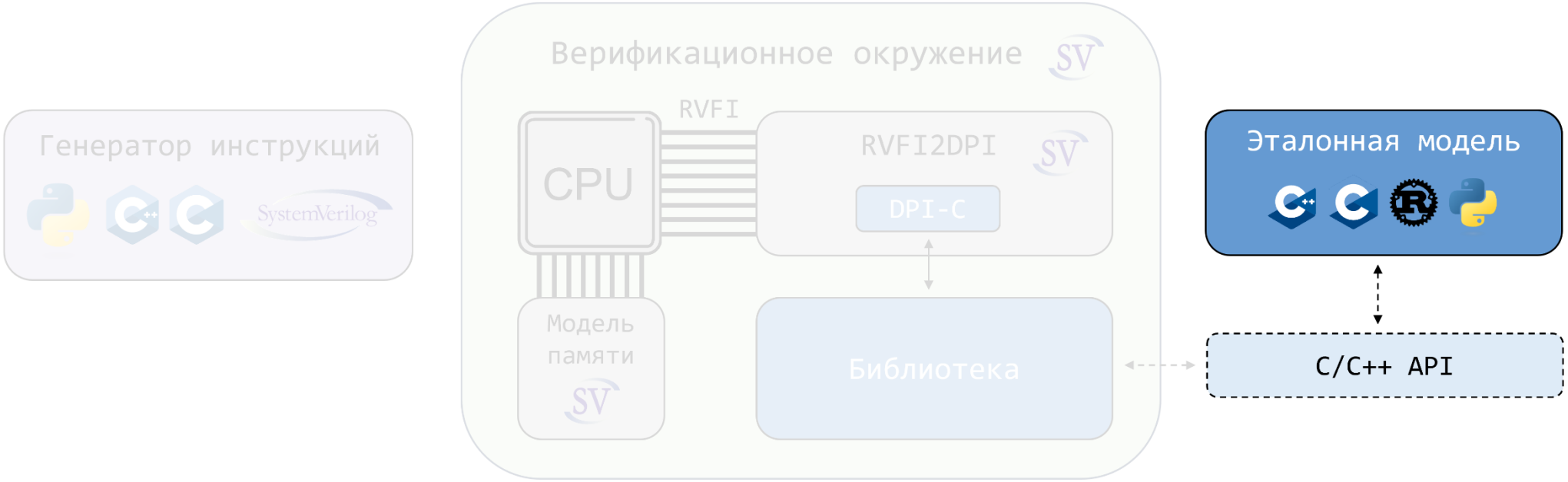
Step-and-Compare с использованием открытого ПО

- Генератор случайных инструкций AAPG:
  - gitlab.com/shaktiproject/tools/aapg
- Однозначно стоит обратить внимание на LLVM Snippy:
  - github.com/syntacore/snippy

Step-and-Compare с использованием открытого ПО



Step-and-Compare с использованием открытого ПО



Step-and-Compare с использованием открытого ПО

- Эталонная модель:
  - github.com/riscv-software-src/riscv-isa-sim
  - github.com/chipsalliance/VeeR-ISS
  - github.com/GregAC/rrs
  - github.com/cornell-brg/pydgin

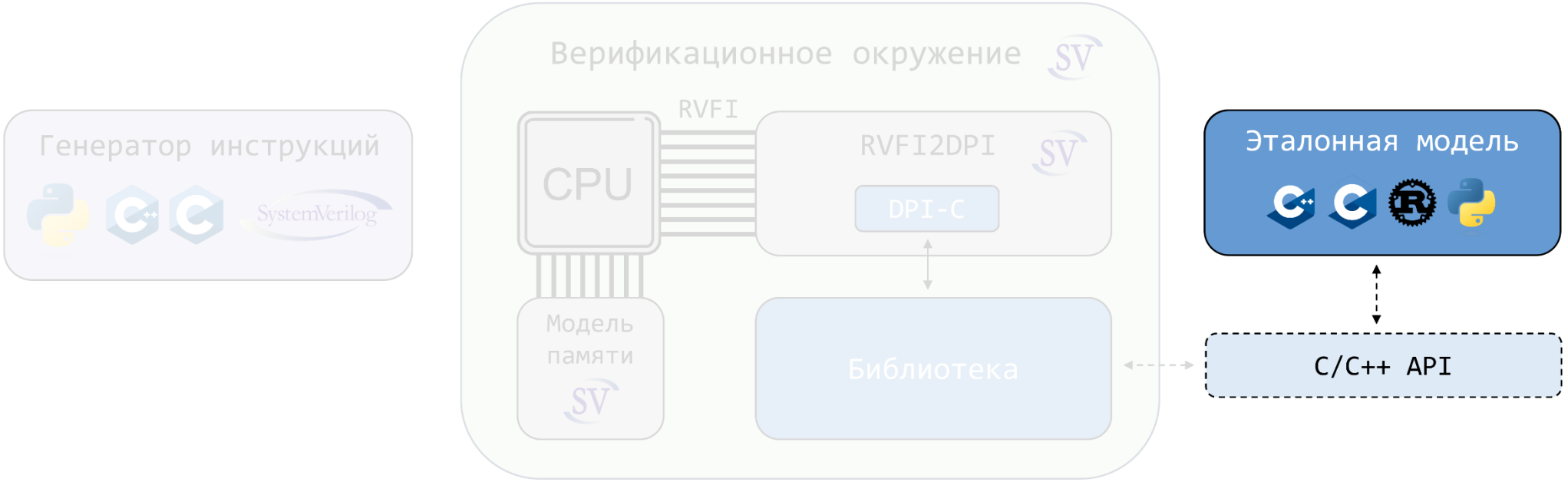
Step-and-Compare с использованием открытого ПО

- Эталонная модель Spike:
  - github.com/riscv-software-src/riscv-isa-sim
- Не поддерживает Step-and-Compare.

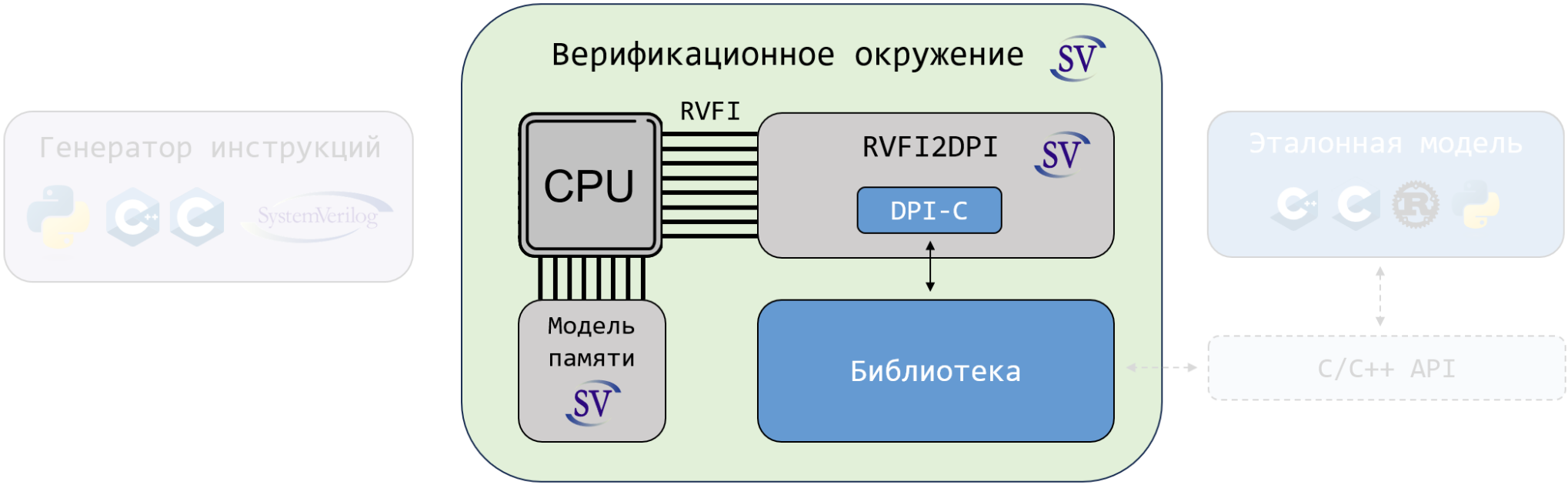
Step-and-Compare с использованием открытого ПО

- Эталонная модель Spike:
  - github.com/riscv-software-src/riscv-isa-sim
- Не поддерживает Step-and-Compare.
- ПО для Step-and-Compare Hammer:
  - https://github.com/rivosinc/hammer

Step-and-Compare с использованием открытого ПО



Step-and-Compare с использованием открытого ПО



Step-and-Compare с использованием открытого ПО

- Верификационное окружение:
  - 

  - DPI-C
  - ООП
- Подробный разбор концепций;
- Простота реализации.

Step-and-Compare с использованием открытого ПО

- Открытый ознакомительный курс по верификации RISC-V ядер:
  - Использование исключительно открытого ПО;
  - Предоставление виртуальной машины;
  - Теоретическая и практическая части;
  - Использование разнообразных методик;
  - Наличие эталонных реализаций.

  - Практика по Step-and-Compare в разработке;
  - Эталонная реализация уже доступна.

Материалы доклада, Q&A и не только

Курс по верификации RISC-V

Telegram канал

```
$display("VFA");
```

Step-and-Compare с использованием открытого ПО