

Московский физико-технический институт (государственный университет)

Физтех-школа радиотехники и компьютерных технологий

Кафедра информатики и вычислительной техники

Разработка модуля сценарного программирования ТАР-контроллера ядер микропроцессора R2000+ для инженерной консоли

Выпускная квалификационная работа
(бакалаврская работа)

студент: Романов С. М.
научный руководитель: Фролов П. В.

Москва, 2021

Введение

В состав микропроцессора R2000+ входит TAP-контроллер кластера, позволяющий взаимодействовать с процессорными ядрами.

Тестовые регистры данных TAP-контроллера

- DICR (66 бит) — запись теневой инструкции; чтение PC и некоторых системных флагов.
- DSR (12 бит) — командный регистр: останов и продолжение дешифрации, выполнение текущей инструкции и теневой инструкции.
- DCUIOR (64 бит) — используется для передачи данных между JTAG интерфейсом и ядром.

Для доступа к TAP-контроллеру используется отладочное ПО — инженерная консоль.

Необходимо обеспечить доступ из инженерной консоли к тестовым регистрам с помощью языка программирования Python.

Цель работы

Разработать модуль сценарного программирования TAP-контроллера ядер микропроцессора R2000+ для инженерной консоли

Задачи

- Реализовать прототип модуля для взаимодействия с RTL-моделью:
 - определить набор функций интерфейса доступа к регистрам TAP-контроллера;
 - реализовать функции на языке C++;
 - сформировать связывающий код между языками программирования Python и C++.
- Разработать модуль для инженерной консоли:
 - сформировать связывающий код для функций, реализованных в инженерной консоли;
 - встроить интерпретатор языка программирования Python в инженерную консоль и импортировать в него сформированный модуль.

Функции интерфейса доступа к регистрам TAP-контроллера

- void fetch_off (int core_id) - останов выполнения
- void fetch_on (int core_id) - продолжение после останова
- void exec_IR (int core_id) - исполнение очередной инструкции из буфера команд
- void exec_shadow (int core_id, uint32_t instruction) - исполнение команды с теневого регистра DICR
- uint64_t get_DCUIOR (int core_id) - чтение регистра DCUIOR
- void put_DCUIOR (int core_id, uint64_t data) - запись регистра DCUIOR
- void get_DICR (int core_id) - чтение и расшифровка DICR

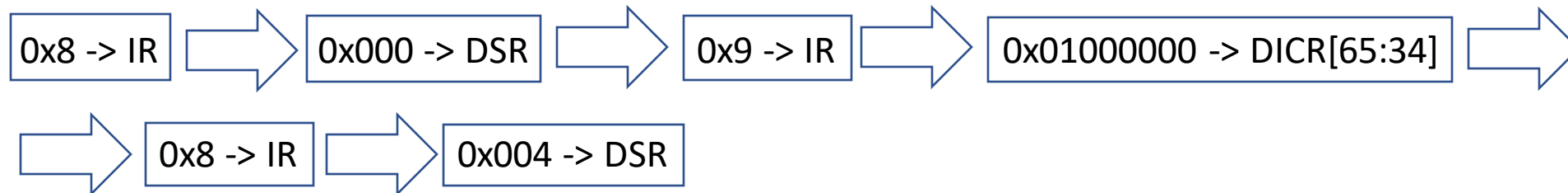
core_id - номер вычислительного ядра

Пример реализации интерфейсной функции

```
void exec_shadow(int core_id, uint32_t instruction)
{
    Код регистра DSR-----> IR (4 бит)
    Номер ядра-----> DR (12 бит)
    Код регистра DICR-----> IR (4 бит)
    Теневая инструкция-----> DR (66 бит)
    Код регистра DSR-----> IR (4 бит)
    Команда исполнения теневой инструкции, номер ядра-----> DR (12 бит)
}
```

\\ Instruction Register (IR), Data Register (DR) - стандартные регистры JTAG-интерфейса

Пример исполнения функции для команды NOP:



0x9 - код регистра DICR 0x000 - выбор нулевого ядра

0x8 - код регистра DSR 0x004 - исполнение теневой инструкции на нулевом ядре

0x01000000 – код команды NOP

Формирование связывающего кода

SWIG (Simplified Wrapper and Interface Generator) — свободное ПО, формирующее программный код для связывания C++ библиотек/программ с другими языками.

На вход SWIG подается интерфейсный файл.

На выходе генерируется связывающий код.

RequiredFunctions.i

```
%module python_jtag_emu
%{
extern void fetch_off(int core_id);
%}
```

Интерфейсный файл

содержит прототипы экспортируемых функций на языке C++.

SWIG

Генерирует обертку для трансформации объектов C++ в объекты Python.

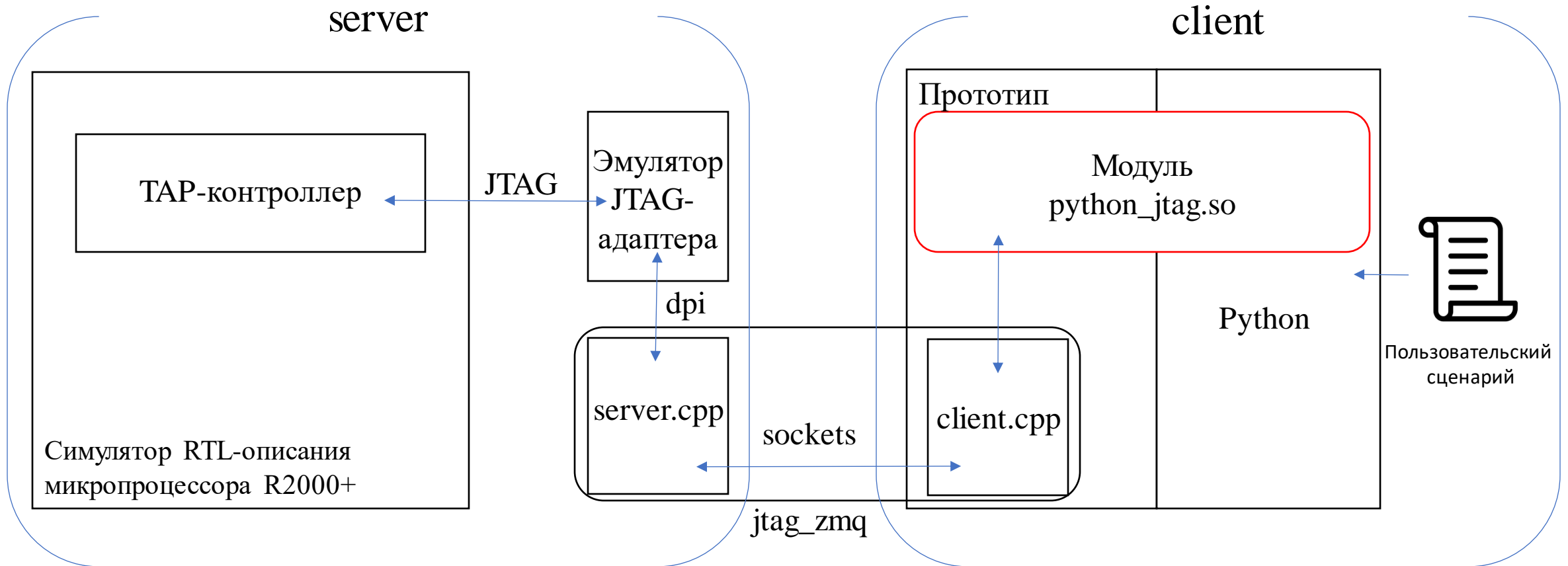
PythonWrappedFunctions.cpp

```
#include <python.h>
...
PyObject* wrap_fetch_off()
{
...
}
```

Сгенерированная SWIGом обёртка экспортируемых функций.

Из исходного кода функций и сгенерированной обертки собирается динамическая библиотека, которая является импортируемым модулем Python.

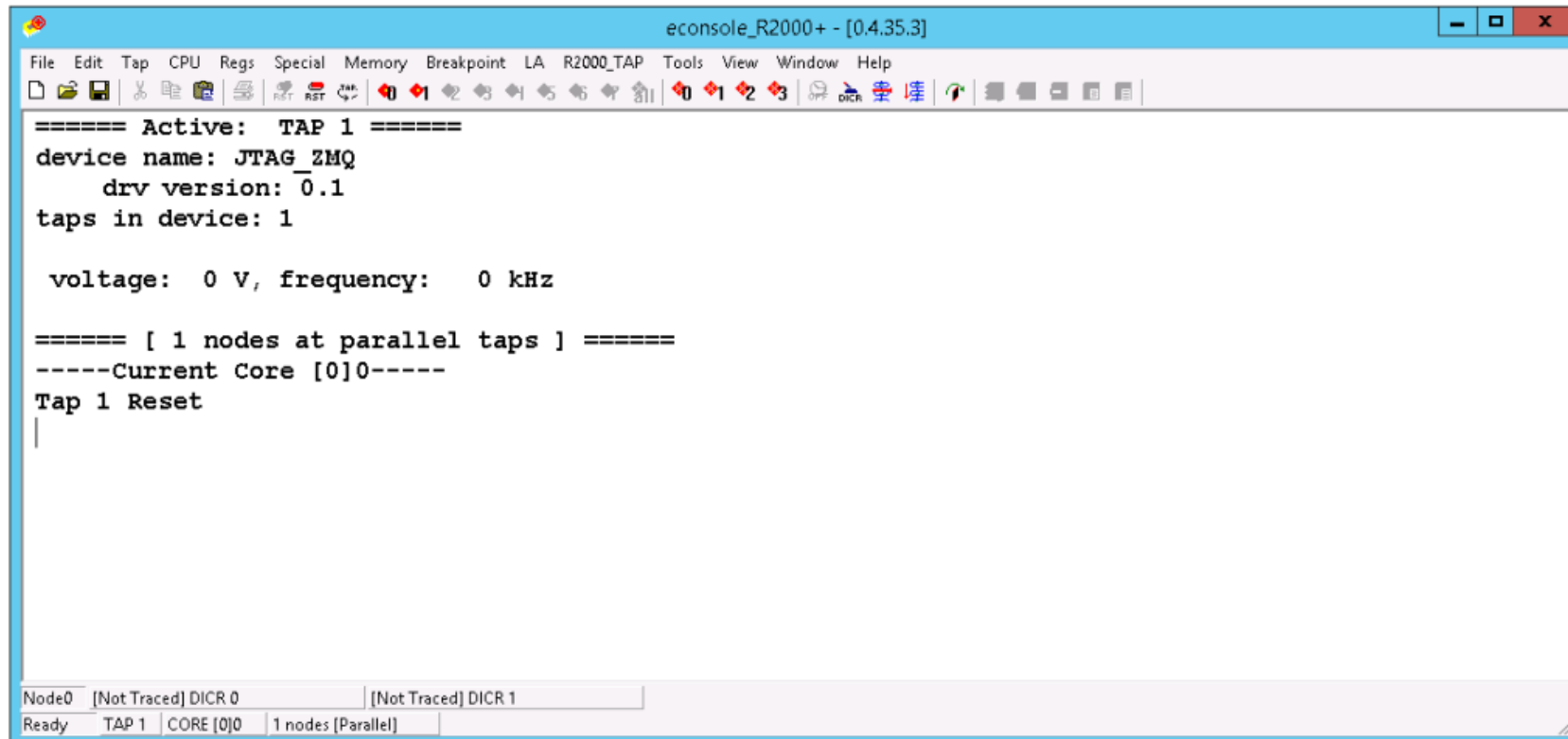
Тестовый стенд



jtag_zmq — библиотека, работающая по принципу клиент-сервер, для соединения с симулятором RTL. Разработана в АО МЦСТ.

Инженерная консоль

Инженерная консоль — программа с графическим интерфейсом пользователя под ОС Windows, предоставляющая доступ к отладочным средствам R2000+



```
econsole_R2000+ - [0.4.35.3]
File Edit Tap CPU Regs Special Memory Breakpoint LA R2000_TAP Tools View Window Help
===== Active: TAP 1 =====
device name: JTAG_ZMQ
drv version: 0.1
taps in device: 1

voltage: 0 V, frequency: 0 kHz

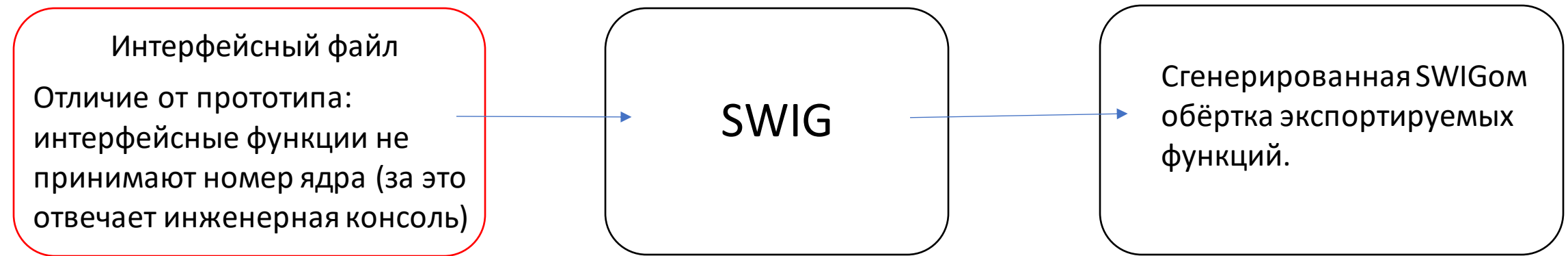
===== [ 1 nodes at parallel taps ] =====
-----Current Core [0]0-----
Tap 1 Reset
|

Node0 [Not Traced] DICR 0 [Not Traced] DICR 1
Ready TAP 1 CORE [0]0 1 nodes [Paralle]
```

Необходимо добавить возможность исполнения пользовательских сценариев на языке Python для работы с регистрами TAP-контроллера ядер.

Формирование связывающего кода

Реализовывать функции взаимодействия с регистрами TAP-контроллера не надо, они уже присутствуют в инженерной консоли.



Доработана система сборки инженерной консоли:

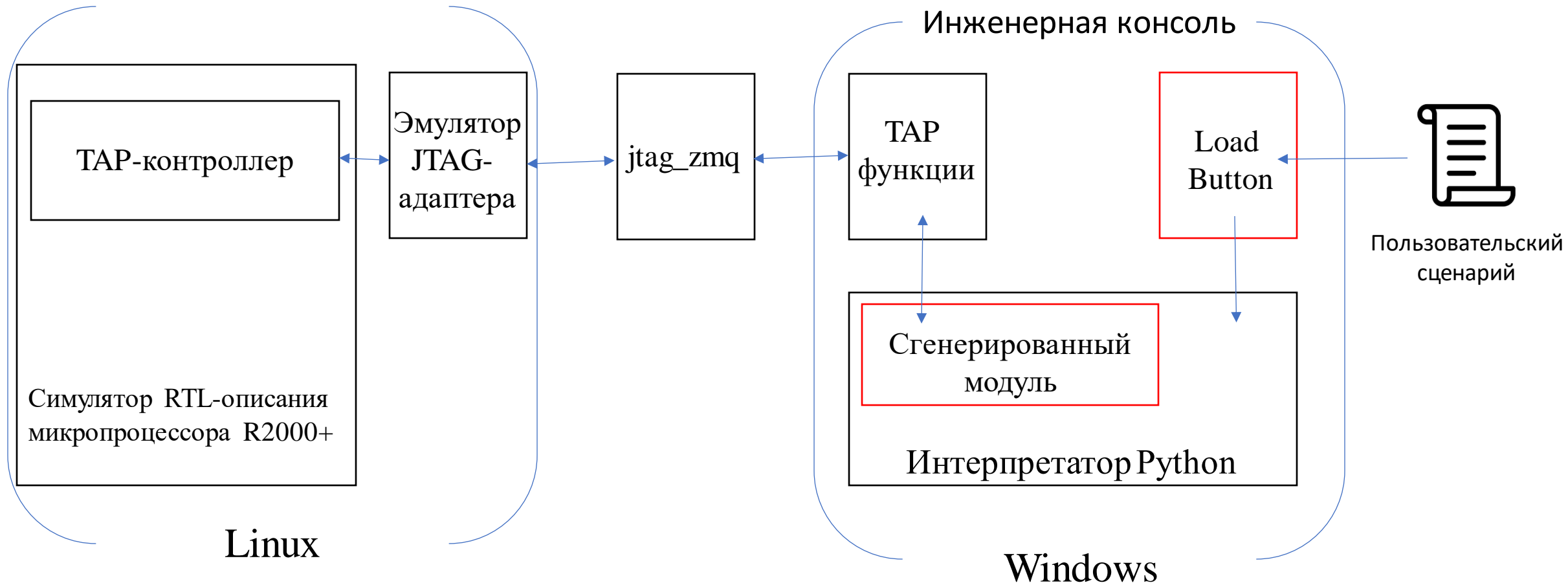
- связывающий код формируется SWIG;
- полученный код компилируется в статическую библиотеку, являющуюся модулем Python.

Символы библиотеки необходимо импортировать в встроенный в консоль интерпретатор Python.

Встраивание интерпретатора в инженерную консоль и импортирование в него реализованного модуля

```
void OnLoadPy() { // метод вызываемый при нажатии клавиши LoadPy
    ... //
    PyImport_Inittab("python_jtag", PyInit__python_jtag); // Добавление python_jtag в таблицу
    // встроенных модулей интерпретатора
    // PyInit__python_jtag - функция инициализации
    // модуля, сгенерированного с помощью SWIG
    //
    PyInitialize(); // Инициализация интерпретатора
    PyRun_SimpleString("from python_jtag import *"); // Импорт модуля
    ... //
    PyRun_SimpleFile(my_script); // Исполнение пользовательского сценария my_script
    //
    PyFinalize(); // Завершение интерпретатора
    ...
}
```

Тестовый стенд



Результаты

- Реализован прототип модуля
 - Определен и реализован набор функций интерфейса доступа к регистрам TAP-контроллера ядер
 - Сгенерирован связывающий код
- Реализован модуль для инженерной консоли
 - Сгенерирован связывающий код, реализовавший модуль
 - Встроен интерпретатор языка Python
 - В интерпретатор импортирован сгенерированный модуль
 - Начата отладка модуля