

Учреждение Российской академии наук  
Институт системного программирования РАН (ИСП РАН)  
Отдел Технологий программирования

**Утилита *VeriTool*:**  
**Руководство пользователя**  
(версия 0.1-beta)

**Москва 2010**

# Содержание

# Глава 1

## Введение

### 1.1 Назначение и основные возможности

Утилита *VeriTool* предназначена для поддержки разработки тестовых систем для Verilog-моделей аппаратуры. Утилита позволяет автоматически генерировать вспомогательные компоненты тестовой системы, сокращая трудозатраты на разработку тестов. К числу таких компонентов относятся:

- *Verilog-окружение (testbench)* — Verilog-модуль верхнего уровня, содержащий в себе экземпляр тестируемой Verilog-модели. Verilog-окружение, используя VPI-функции, запускает нужный тестовый сценарий и осуществляет в цикле подачу тестовых воздействий и проверку реакций;
- *VPI-медиатор* — компонент тестовой системы, предоставляющий API для задания входных сигналов тестируемой Verilog-модели и получения выходных сигналов. Реализация VPI-медиатора основана на стандартизованном интерфейсе VPI (Verilog Procedural Interface);
- *Шаблоны VPI-функций тестовой системы* — заготовки VPI-функций, используемых Verilog-окружением для запуска тестового сценария, подачи тестовых воздействий и проверки реакций;
- *C-модель интерфейса тестируемой модели* — представление на языке программирования C входов и выходов тестируемой Verilog-модели. C-модель интерфейса используется как часть спецификационной модели данных.

### 1.2 Имеющиеся ограничения

Настоящая версия утилиты *VeriTool* имеет следующие ограничения:

- Утилита не обрабатывает входные и выходные сигналы, разрядность которых превышает 256 бит.
- При генерации Verilog-окружения утилита предполагает, что тестируемый модуль имеет ровно один сигнал синхроимпульса (clock).

## Глава 2

# Системные требования и установка

## 2.1 Системные требования

Для сборки, установки и использования утилиты *VeriTool* требуются базовые утилиты Unix, gcc, make, а также свободно распространяемый симулятор Icarus Verilog, парсер которого используется для анализа входных Verilog-файлов.

Исходный код Icarus Verilog доступен на сайте <http://www.icarus.com/eda/verilog/>.

## 2.2 Установка утилиты

Утилита *VeriTool* распространяется вместе с исходным кодом в архиве, который называется *veritool-номер-версии.tar.gz*. Для установки утилиты необходимо выполнить следующую последовательность действий:

- Установить симулятор Icarus Verilog.
- Присвоить переменной окружения ICARUS\_HOME путь к каталогу установки симулятора. Например, если он установлен в каталоге /usr/local, нужно написать<sup>1</sup>:  
> export ICARUS\_HOME=/usr/local
- Распаковать архив с утилитой:  
> gunzip -c veritool-номер-версии-.tar.gz | tar -xvf -
- Скомпилировать утилиту:  
> cd veritool-номер-версии  
> sh configure  
> make all
- Собственно, установить утилиту:  
> make install

По умолчанию, утилита устанавливается в каталоге /usr/local. Для того чтобы изменить каталог установки, необходимо задать параметр --prefix при запуске скрипта configure:

```
> sh configure --prefix=каталог-установки
```

---

<sup>1</sup>Если компиляция и установка утилиты выполняются разными пользователями, переменная окружения должна быть установлена у них обоих.

## Глава 3

# Параметры утилиты

### 3.1 Опции командной строки

Утилита *VeriTool* имеет следующий формат запуска:

```
> veritool [опции] входные-файлы
```

Ниже перечислены поддерживаемые опции командной строки:

- |  |   |
|--|---|
| <code>--module=<i>module</i></code>    | задает имя тестируемого Verilog-модуля, для которого генерируются компоненты тестовой системы. Если эта опция отсутствует, обрабатывается только первый модуль, определенный во входных файлах.   |
| <code>--clk=<i>signal</i></code>       | задает имя сигнала синхроимпульса. Если эта опция отсутствует, считается, что таким сигналом является <code>clk</code> . Опция <code>--clk</code> имеет смысл, только если генерируется Verilog-окружение (указана опция <code>--all</code> или <code>--testbench</code> ).   |
| <code>--rst=<i>signal</i></code>       | задает имя сигнала сброса. Если эта опция отсутствует, считается, что таким сигналом является <code>rst</code> . Опция <code>--rst</code> имеет смысл, только если генерируется Verilog-окружение (указана опция <code>--all</code> или <code>--testbench</code> ).   |
| <code>--all</code>                     | включает генерацию всех компонентов тестовой системы: Verilog-окружения, VPI-медиатора, VPI-функций и C-модели интерфейса. Установка этой опции эквивалентна одновременной установке <code>--testbench</code> , <code>--vpi-media</code> , <code>--vpi-systf</code> и <code>--interface</code> (без параметров) или отсутствию всех этих опций. |
| <code>--testbench[=<i>file</i>]</code> | включает генерацию Verilog-окружения и может задавать имя соответствующего файла. Если имя файла не указано, Verilog-окружение сохраняется в файле <code>testbench.v</code> .   |
| <code>--vpi-media[=<i>file</i>]</code> | включает генерацию VPI-медиатора и может задавать имя соответствующего файла. Если имя файла не указано, VPI-медиатор сохраняется в файле <code>vpi_media.c</code> .  |

<code>--vpi-media-header=<i>file</i></code>	задает имя заголовочного файла VPI-медиатора. Данная опция учитывается, только если установлена опция <code>--vpi-media</code> . Если опция <code>--vpi-media-header</code> отсутствует (при установленной <code>--vpi-media</code> ), имя заголовочного файла получается из имени основного файла заменой расширения на <code>.h</code> .
<code>--vpi-systf[=<i>file</i>]</code>	включает генерацию шаблонов VPI-функций и может задавать имя соответствующего файла. Если имя файла не указано, шаблоны VPI-функций сохраняются в файле <code>vpi_systf.c</code> .
<code>--vpi-systf-header=<i>file</i></code>	задает имя заголовочного файла для шаблонов VPI-функций. Данная опция учитывается, только если установлена опция <code>--vpi-systf</code> . Если опция <code>--vpi-systf-header</code> отсутствует (при установленной <code>--vpi-systf</code> ), имя заголовочного файла получается из имени основного файла заменой расширения на <code>.h</code> .
<code>--interface[=<i>file</i>]</code>	включает генерацию С-модели интерфейса и может задавать имя соответствующего файла. Если имя файла не указано, С-модель интерфейса сохраняется в файле <code>interface.c</code> .
<code>--interface-header=<i>file</i></code>	задает имя заголовочного файла С-модели интерфейса. Данная опция учитывается, только если установлена опция <code>--interface</code> . Если опция <code>--interface-header</code> отсутствует (при установленной <code>--interface</code> ), имя заголовочного файла получается из имени основного файла заменой расширения на <code>.h</code> .
<code>--destination=<i>dir</i></code>	задает выходной каталог. Если эта опция не указана, файлы сохраняются в текущий каталог.
<code>--version   -v</code>	вывод информации о версии утилиты.
<code>--help   -h</code>	вывод информации о доступных опциях.

## 3.2 Примеры запуска утилиты

```
> veritool --all --testbench=fpu_test.v --vpi-media=fpu_media.c module.v
```

При запуске утилиты *VeriTool* с указанными параметрами будут сгенерированы все компоненты тестовой системы для модуля, определенного в файле `module.v`. При этом Verilog-окружение будет помещено в файле `fpu_test.v`, а VPI-медиатор — в файле `fpu_media.c` (заголовочный файл — `fpu_media.p`). Оставшиеся компоненты тестовой системы будут расположены в файлах по умолчанию: VPI-функции — в файле `vpi_systf.c`, С-модель интерфейса — в файле `interface.c`.

### 3.3 Файл конфигурации

Файл конфигурации позволяет управлять результатом генерации. С помощью файла конфигурации можно задавать названия типов данных в генерируемых файлах (например, типа для C-модели интерфейса) и функций (например, VPI-функций, используемых для запуска тестового сценария, подачи тестовых воздействий и проверки реакций).

В настоящей версии утилиты *VeriTool* данная возможность не поддерживается.