|  |
| --- |
| Версия документа: 0.10 |
| Процессорное ядро schoolMIPS |
| Руководство пользователя |
|  |
| Станислав Жельнио |
| 2017 |

Оглавление

[1. Благодарности 3](#_Toc493028898)

[2. Введение 3](#_Toc493028899)

[3. Уровень поддержки MIPS архитектуры 3](#_Toc493028900)

[4. Аппаратное и программное обеспечение 4](#_Toc493028901)

[5. Структура проекта 5](#_Toc493028902)

[6. Порядок развертывания 7](#_Toc493028903)

[6.1. Развертывание программного окружения 7](#_Toc493028904)

[6.2. Получение schoolMIPS 7](#_Toc493028905)

[7. Порядок запуска 7](#_Toc493028906)

[7.1. Порядок сборки программы 7](#_Toc493028907)

[7.2. Запуск в режиме симуляции 7](#_Toc493028908)

[7.3. Синтез проекта и программирование ПЛИС Altera 7](#_Toc493028909)

[7.4. Синтез проекта и программирование ПЛИС Xilinx 8](#_Toc493028910)

[7.5. Интерфейс пользователя 9](#_Toc493028911)

[8. Миграция и добавление кода 10](#_Toc493028912)

[ПРИЛОЖЕНИЕ 1. Пример синтеза и конфигурации ПЛИС Altera 11](#_Toc493028913)

[ПРИЛОЖЕНИЕ 2. Пример синтеза и конфигурации ПЛИС Xilinx 13](#_Toc493028914)

**Список таблиц**

[Таблица 1. Поддерживаемые коды поля Opcode 3](#_Toc493028915)

[Таблица 2. Поддерживаемые коды поля Function 4](#_Toc493028916)

[Таблица 3. Программное окружение 4](#_Toc493028917)

[Таблица 4. Текущий статус поддержки отладочных плат 5](#_Toc493028918)

[Таблица 5. Структура каталогов проекта 5](#_Toc493028919)

[Таблица 6. Типовая структура каталога тестовой программы 5](#_Toc493028920)

[Таблица 7. Модульный состав проекта 6](#_Toc493028921)

[Таблица 8. Интерфейсные сигналы основного модуля процессорного ядра (sm\_cpu) 6](#_Toc493028922)

[Таблица 9. Элементы управления отладочной платы 9](#_Toc493028923)

[Таблица 10. Назначение и номера регистров MIPS 10](#_Toc493028924)

## Благодарности

Этот проект появился на свет благодаря коллективу переводчиков [учебника Дэвида Харриса и Сары Харрис «Цифровая схемотехника и архитектура компьютера»](https://habrahabr.ru/post/259505/), участникам конференции "Young Russian Chip Architects" (yrca@googlegroups.com), а также персонально:

|  |  |
| --- | --- |
| Юрий Панчул, [@yuri-panchul](https://github.com/yuri-panchul) | идея проекта, архитектура |
| Станислав Жельнио, [@zhelnio](https://github.com/zhelnio) | архитектура, кодирование |
| Александр Романов, [@RomeoMe5](https://github.com/RomeoMe5) | архитектура, поддержка Terasic MAX10 Neek, Terasic DE10-Standard, Марсоход 3 |
| Oleg Lyovin, [@olegartys](https://github.com/olegartys) | багфикс, подержка Marsohod3b |
| Dmitry Dluzhnevsky, [@Dluzhnevsky](https://github.com/Dluzhnevsky) | поддержка RZ-EasyFPGA A2.1 |
| [@woodywitch](https://github.com/woodywitch) | поддержка RZ-EasyFPGA A2.1 |

## Введение

schoolMIPS - это простейшее процессорное ядро, разработанное в рамках инициативы по преподаванию школьникам основ цифровой схемотехники, языков описания аппаратуры и использования ПЛИС[[1]](#footnote-1).

Основные особенности:

* язык описания аппаратуры Verilog;
* подмножество архитектуры MIPS с памятью инструкций, с регистрами общего назначения, но без памяти данных;
* однотактовая микроархитектура;
* минимальный набор инструкций, первоначально достаточный для вычисления числа Фибоначчи и целочисленного квадратного корня итеративным способом;
* максимально упрощенная в целях преподавания микроархитектура;
* является упрощенной версией процессорного ядра MIPS написанного Сарой Харрис.

В состав schoolMIPS входят:

* исходные коды процессорного ядра;
* примеры запускаемых на выполнение программ (тестовые программы);
* проекты средств синтеза, необходимые для запуска schoolMIPS на отладочных платах;
* набор скриптов, осуществляющих компиляцию примеров, симуляцию и подготовку к синтезу;
* документация (включая настоящий документ).

Проект schoolMIPS доступен для загрузки по адресу: <https://github.com/MIPSfpga/schoolMIPS>

## Уровень поддержки MIPS архитектуры

Поддерживаемы по состоянию на 01.07.2017 инструкции приведены в таблицах[[2]](#footnote-2) ниже (отмечены цветом).

Таблица 1. Поддерживаемые коды поля Opcode

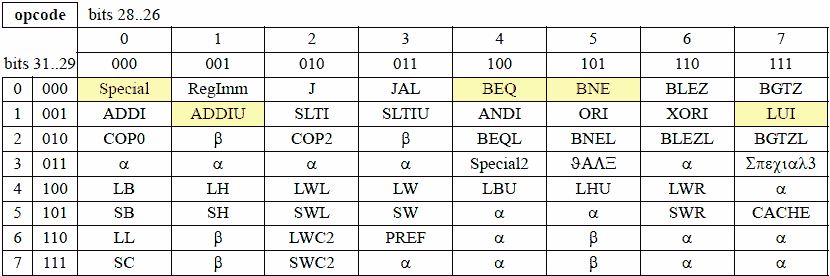
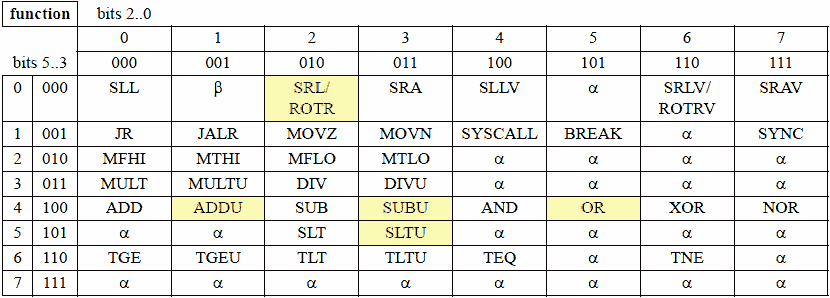


Таблица 2. Поддерживаемые коды поля Function



## Аппаратное и программное обеспечение

Минимально необходимый для работы со schoolMIPS набор аппаратного и программного обеспечения, который позволит изучить работу процессорного ядра в режиме симуляции включает:

* компьютер и операционная система;
* средства симуляции;
* средства сборки программ;
* редактор кода

Для того, чтобы развернуть schoolMIPS на ПЛИС и изучить его работу "на железе" дополнительно потребуется:

* отладочная плата ПЛИС;
* средства синтеза

Перечень поддерживаемого аппаратного и программного обеспечения приведен в таблицах ниже.

Таблица 3. Программное окружение

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Наименование** | **Версия** | **Примечание** |
| **1** | **2** | **3** | **4** |
| Операционные системы | | | |
|  | ОС Windows | 7 | Работа на других версиях ОС - не тестировалась |
|  | ОС Ubuntu Linux | 16.10 | Необходимо выполнить замену \*.bat скриптов в каталогах проекта на их \*.sh аналоги. (подробнее - в Разделе "Развертывание программного окружения") |
| Инструменты синтеза | | | |
|  | [Quartus Prime](http://dl.altera.com/?edition=lite) | 16.1 | Используется для синтеза и последующего запуска на отладочных платах с ПЛИС Altera (Intel) |
|  | Xilinx Vivado | 2017.2 | Используется для синтеза и последующего запуска на отладочных платах с ПЛИС Xilinx |
| Средства симуляции | | | |
|  | ModelSim | 10.5b | Используется для запуска в режиме симуляции. Задействована версия INTEL FPGA STARTER EDITION (Revision: 2016.10), входящая в состав Quartus Prime |
|  | [Icarus Verilog](http://iverilog.icarus.com/) | 10.1.1 | Используется для запуска в режиме симуляции |
| Средства сборки ПО | | | |
|  | [Codescape MIPS SDK](https://community.imgtec.com/developers/mips/tools/codescape-mips-sdk/download-codescape-mips-sdk-essentials/) | 1.4.1.07 | Используется для компиляции примеров программ и формирования hex-образов памяти инструкций |
| Редактор кода | | | |
|  | [Visual Studio Code](https://code.visualstudio.com/) | 1.13.1 | Рекомендуется к использованию для навигации по каталогам проекта, а также в качестве текстового редактора разработчиков (обучаемых). Перечень рекомендуемых расширений: "Git History(git log)", "C/C++", "hexdump for VSCode", "MIPS Support", "SystemVerilog", "TCL", "VerilogHDLs" |

Таблица 4. Текущий статус поддержки отладочных плат

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Наименование платы** | **ПЛИС** | **Статус** |
| **1** | **2** | **3** | **4** |
|  | [Terasic DE10-Lite](http://de10-lite.terasic.com) | [Altera MAX10](https://www.altera.com/products/fpga/max-series/max-10/overview.html) | Поддерживается в полном объеме |
|  | [Terasic DE1-SoC](http://de1-soc.terasic.com.tw/) | [Altera Cyclone V SoC](https://www.altera.com/products/soc/portfolio/cyclone-v-soc/) | Поддерживается в полном объеме |
|  | [Terasic MAX10 Neek](http://max10.terasic.com) | [Altera MAX10](https://www.altera.com/products/fpga/max-series/max-10/overview.html) | Поддерживается в полном объеме |
|  | [Марсоход 3](https://marsohod.org/howtostart/plata-marsokhod3) | [Altera MAX10](https://www.altera.com/products/fpga/max-series/max-10/overview.html) | На плате отсутствуют HEX-индикаторы и переключатели. |
|  | [Terasic DE10-Standard](http://de10-standard.terasic.com) | [Altera Cyclone V SoC](https://www.altera.com/products/soc/portfolio/cyclone-v-soc/) | Поддерживается в полном объеме |
|  | [Terasic DE0](http://de0.terasic.com) | [Altera Cyclone III](https://www.altera.com/products/fpga/cyclone-series/cyclone-iii/overview.html) | Поддерживается в полном объеме |
|  | [Terasic DE10-Nano](http://de10-nano.terasic.com/) | [Altera Cyclone V SoC](https://www.altera.com/products/soc/portfolio/cyclone-v-soc/) | На плате отсутствуют HEX-индикаторы |
|  | [Terasic DE0-CV](http://de0-cv.terasic.com.tw) | [Altera Cyclone V](https://www.altera.com/products/fpga/cyclone-series/cyclone-v/overview.html) | Поддерживается в полном объеме |
|  | [Digilent Nexys4](http://store.digilentinc.com/nexys-4-artix-7-fpga-trainer-board-limited-time-see-nexys4-ddr/) | [Xilinx Artix-7](http://www.xilinx.com/products/silicon-devices/fpga/artix-7.html) | Поддерживается в полном объеме |
|  | [Digilent Nexys4-DDR](http://store.digilentinc.com/nexys-4-ddr-artix-7-fpga-trainer-board-recommended-for-ece-curriculum/) | [Xilinx Artix-7](http://www.xilinx.com/products/silicon-devices/fpga/artix-7.html) | Поддерживается в полном объеме |
|  | [RZ-EasyFPGA A2.1](https://github.com/MIPSfpga/schoolMIPS/blob/master/board/rz_easyFPGA_A2.1/README.md) | [Altera Cyclone IV](https://www.altera.com/products/fpga/cyclone-series/cyclone-iv/overview.html) | На плате отсутствуют HEX-индикаторы |

## Структура проекта

Структура каталогов проекта, его модульный состав, типовая структура каталога тестовой программы  
([program](https://github.com/MIPSfpga/schoolMIPS/tree/master/program) / <имя программы> / \*) с описанием вложенных каталогов, файлов, их назначения и отдельных особенностей использования, а также перечень интерфейсных сигналов основного модуля приведены в таблицах ниже.

Таблица 5. Структура каталогов проекта

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Наименование каталога** | **Описание** |
| **1** | **2** | **3** |
|  | [doc](https://github.com/MIPSfpga/schoolMIPS/tree/master/doc) | Документация проекта, включая настоящий документ |
|  | [src](https://github.com/MIPSfpga/schoolMIPS/tree/master/src) | Платформонезависимый исходный код процессорного ядра schoolMIPS |
|  | [board](https://github.com/MIPSfpga/schoolMIPS/tree/master/board) | Платформозависимый исходный код. Проекты средств синтеза (Quartus) и модули верхнего уровня специфические для отладочных плат |
|  | [board](https://github.com/MIPSfpga/schoolMIPS/tree/master/board) / [program](https://github.com/MIPSfpga/schoolMIPS/tree/master/board/program) | Hex-файлы памяти программ, используемые при синтезе для последующей инициализации блоков памяти ПЛИС |
|  | [board](https://github.com/MIPSfpga/schoolMIPS/tree/master/board) / <наименование платы> | Проект средства синтеза (Quartus), включая модуль верхнего уровня специфический для конкретной отладочной платы ПЛИС |
|  | [program](https://github.com/MIPSfpga/schoolMIPS/tree/master/program) | Примеры программ (тестовые программы), включая исходные коды, скрипты компиляции и симуляции. Работа каждого примера проверена в режиме симуляции и подтверждена на отладочной плате |
|  | [program](https://github.com/MIPSfpga/schoolMIPS/tree/master/program) / [00\_counter](https://github.com/MIPSfpga/schoolMIPS/tree/master/program/00_counter) | Простейший инкрементальный счетчик |
|  | [program](https://github.com/MIPSfpga/schoolMIPS/tree/master/program) / [01\_fibonacci](https://github.com/MIPSfpga/schoolMIPS/tree/master/program/01_fibonacci) | Вычисление последовательности чисел Фибоначчи |
|  | [program](https://github.com/MIPSfpga/schoolMIPS/tree/master/program) / [02\_sqrt](https://github.com/MIPSfpga/schoolMIPS/tree/master/program/02_sqrt) | Вычисление квадратного корня итеративным способом |
|  | [testbench](https://github.com/MIPSfpga/schoolMIPS/tree/master/testbench) | Verilog модули для тестирования. Используются только в режиме симуляции |
|  | [scripts](https://github.com/MIPSfpga/schoolMIPS/tree/master/scripts) | Служебные скрипты и утилиты |

Таблица 6. Типовая структура каталога тестовой программы

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Имя** | **Описание** |
| **1** | **2** | **4** |
|  | main.S | Файл с исходным кодом тестовой программы |
|  | program.elf | Бинарный файл тестовой программы. Формируется компилятором в результате выполнения скрипта 01\_compile\_and\_link.bat |
|  | program.hex | Текстовый файл памяти инструкций тестовой программы. Используется для инициализации памяти при симуляции и синтезе. Является результатом работы скрипта 03\_generate\_verilog\_readmemh\_file.bat |
|  | program.ld | Скрипт компоновщика, определяющий состав program.elf. Используется скриптом 01\_compile\_and\_link.bat |
|  | program.dis | Результат дисассемблирования program.elf. Является результатом работы скрипта 02\_disassemble.bat |
|  | modelsim\_script.tcl | Скрипт, задающий конфигурацию ModelSim и обеспечивающий запуск тестового примера в режиме симуляции. Используется скриптом 04\_simulate\_with\_modelsim.bat |
|  | sim | Временный каталог, внутри которого запускается ModelSim для последующей симуляции. Создается скриптом 04\_simulate\_with\_modelsim.bat |
|  | 00\_clean\_all.bat | Скрипт. Выполняет удаление временных файлов и каталогов. |
|  | 01\_compile\_and\_link.bat | Скрипт. Осуществляет сборку программы main.S и формирование program.elf. В своей работе использует program.ld |
|  | 02\_disassemble.bat | Скрипт. Выполняет дизассемблирование program.elf, результатом работы является program.dis |
|  | 03\_generate\_verilog\_readmemh\_file.bat | Скрипт. Выполняет формирование program.hex из program.elf |
|  | 04\_simulate\_with\_modelsim.bat | Скрипт. Осуществляет запуск тестовой программы в симуляторе ModelSim. В своей работе использует modelsim\_script.tcl и создает временный каталог sim |
|  | 05\_copy\_program\_to\_board.bat | Скрипт. Выполняет копирование program.hex в каталог [board](https://github.com/MIPSfpga/schoolMIPS/tree/master/board) / [program](https://github.com/MIPSfpga/schoolMIPS/tree/master/board/program) для его последующего использования при синтезе |
|  | 06\_simulate\_with\_icarus.bat | Скрипт Осуществляет запуск тестовой программы в симуляторе Icarus Verilog |

Таблица 7. Модульный состав проекта

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Наименование модуля** | **Описание** |
| **1** | **2** | **3** |
| Каталог [src](https://github.com/MIPSfpga/schoolMIPS/tree/master/src) | | |
|  | sm\_cpu | Основной модуль процессорного ядра |
|  | sm\_control | Процессорное ядро. Устройство управления |
|  | sm\_alu | Процессорное ядро. Арифметико-логическое устройство |
|  | sm\_register\_file | Процессорное ядро. Регистры общего назначения |
|  | sm\_rom | Процессорное ядро. Память инструкций |
|  | sm\_register | 32-битный регистр |
|  | sm\_register\_we | 32-битный регистр с вводом разрешения записи |
|  | sm\_hex\_display | Адаптер вывода числа на HEX индикатор |
|  | sm\_clk\_divider | Делитель тактовой частоты |
|  | sm\_top | Основной платформонезависимый модуль верхнего уровня |
|  | sm\_metafilter | Фильтр входных сигналов |
| Каталог [testbench](https://github.com/MIPSfpga/schoolMIPS/tree/master/testbench) | | |
|  | sm\_testbench | Основной модуль запуска в режиме симуляции |
| Каталог [board](https://github.com/MIPSfpga/schoolMIPS/tree/master/board) / <наименование платы> | | |
|  | <top> | Модуль верхнего уровня, специфический для отладочной платы. Имя не регламентируется |

Таблица 8. Интерфейсные сигналы основного модуля процессорного ядра (sm\_cpu)

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Наименование сигнала** | **Направление** | **Описание** |
| **1** | **2** | **3** | **4** |
|  | clk | input | Тактовый сигнал |
|  | rst\_n | input | Сигнал системного сброса |
|  | regAddr [4:0] | input | Номер регистра MIPS32, значение которого необходимо вывести в regData. Вывод осуществляется в этом же такте |
|  | regData [31:0] | output | Значение регистра, номер которого задан на regAddr |

## Порядок развертывания

## Развертывание программного окружения

При установке программных пакетов для корректного функционирования скриптов необходимо добавить в переменную системного окружения $PATH следующие каталоги (на примере Windows):

* <Codescape MIPS SDK>\Toolchains\mips-mti-elf\2016.05-03\bin
* <Quartus или Modelsim>\modelsim\_ase\win32aloem)
* <Icarus Verilog>\bin и <Icarus Verilog>\gtkwave\bin

Критерием успешного изменения настроек является доступность для запуска в командной строке вне зависимости от текущего каталога следующих утилит: mips-mti-elf-gcc (компилятор, входящий в состав Codescape MIPS SDK), vsim (симулятор Modelsim), iverilog и gtkwave (симулятор Icarus Verilog).

## Получение schoolMIPS

Развертывание schoolMIPS может быть выполнено:

* путем загрузки архива с текущей версией по адресу: <https://github.com/MIPSfpga/schoolMIPS>
* средствами системы контроля версией git

git clone https://github.com/MIPSfpga/schoolMIPS.git

В случае, если для работы используется Linux, после развертывания необходимо выполнить замену \*.bat скриптов в каталогах проекта на их \*.sh аналоги. Для этого используется scripts/[init\_linux.sh](https://github.com/MIPSfpga/schoolMIPS/blob/master/scripts/init_linux.sh)

## Порядок запуска

В последующих раздела приведено описание порядка сборки программы, а также ее запуска на schoolMIPS в режиме симуляции и на отладочной плате ПЛИС. Имена скриптов приведены в их Windows варианте (\*.bat), для Linux порядок действий остается таким же, при этом используются скрипты (\*.sh).

## Порядок сборки программы

Для сборки программы и формирования дампа памяти программ необходимо:

* запустить интерпретатор командной строки (cmd.exe). При использовании Visual Studio Code это можно сделать с помощью комбинации клавиш **Ctrl+`**.
* перейти в каталог с тестовой программой (на примере программы 00\_counter)

cd program\00\_counter

* выполнить удаление временных файлов, оставшихся от предыдущего запуска (при необходимости)

00\_clean\_all.bat

* выполнить сборку тестовой программы

01\_compile\_and\_link.bat

* выполнить дисассемблирование полученного бинарного файла (при необходимости)

03\_generate\_verilog\_readmemh\_file.bat

## Запуск в режиме симуляции

Для запуска в режиме симуляции необходимо:

* при использовании ModelSim:

04\_simulate\_with\_modelsim.bat

* при использовании Icarus Verilog:

06\_simulate\_with\_icarus.bat

## Синтез проекта и программирование ПЛИС Altera

Пример синтеза конфигурации ПЛИС и ее программирования (со скриншотами интерфейса) приведен в Приложении 1. В общем случае для синтеза конфигурации ПЛИС Altera и ее программирования необходимо:

* выполнить сборку программы и формирование дампа памяти программ;
* скопировать текстовый файл памяти инструкций в каталог board \ program

05\_copy\_program\_to\_board.bat

* перейти в каталог с файлами, специфическими для отладочной платы (на примере DE10-Lite):

cd ..\..\board\de10\_lite

* сформировать каталог с проектом синтеза

make\_project.bat

* запустить программный пакет для синтеза: Quartus Prime 16.1
* открыть в запущенном пакете проект board\de10\_lite\project\de10\_lite.qpf. Для этого можно использовать пункт меню File -> Open Project;
* выполнить компиляцию проекта: Processing -> Start Compilation;
* подключить отладочную плату к рабочей станции;
* запустить интерфейс программатора: Tools -> Programmer;
* в интерфейсе программатора при необходимости выполнить настройку подключения платы (кнопка Hardware Setup), после чего осуществить программирование ПЛИС (кнопка Start).

## Синтез проекта и программирование ПЛИС Xilinx

Пример синтеза конфигурации ПЛИС и ее программирования (со скриншотами интерфейса) приведен в Приложении 2. В общем случае для синтеза конфигурации ПЛИС Xilinx и ее программирования необходимо:

* выполнить сборку программы и формирование дампа памяти программ;
* скопировать текстовый файл памяти инструкций в каталог board \ program

05\_copy\_program\_to\_board.bat

* перейти в каталог с файлами, специфическими для отладочной платы (на примере Nexys4):

cd ..\..\board\nexys4

* сформировать каталог для проекта синтеза

make\_project.bat

* запустить программный пакет для синтеза: Vivado 2017.2
* создать проект: Quick Start -> Create Project

дальнейшие действия выполняются в мастере создания проекта

* раздел Project Name:

имя: nexys4

путь: <Путь к schoolMIPS>\board\nexys4\project

снять отметку Create Subdirectory

* раздел Project Type, выбрать RTL-project
* раздел Add Sources:

Add Directories -> <Путь к schoolMIPS>\board\nexys4

Add Directories -> <Путь к schoolMIPS>\src

Add Files -> <Путь к schoolMIPS>\board\program\program.hex

* раздел Add Constraints

Add Files -> <Путь к schoolMIPS>\board\nexys4\nexys4.xdc

* раздел Default Part

Family: Artix-7

Package: csg324

Speed Grade: -1

Part: xc7a100tcsg324-1

* после завершения работы мастера создания проекта установить для файла program.hex тип

"Memory Initialization Files"

* выполнить сборку проекта: Run Synthesis, Run Implementation, Generate Bitstream
* выполнить программирование ПЛИС

## Интерфейс пользователя

Основные элементы управления и их назначение при работе с schoolMIPS приведены ниже   
(на примере платы DE10-Lite):

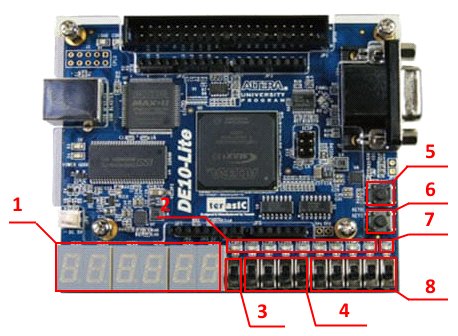


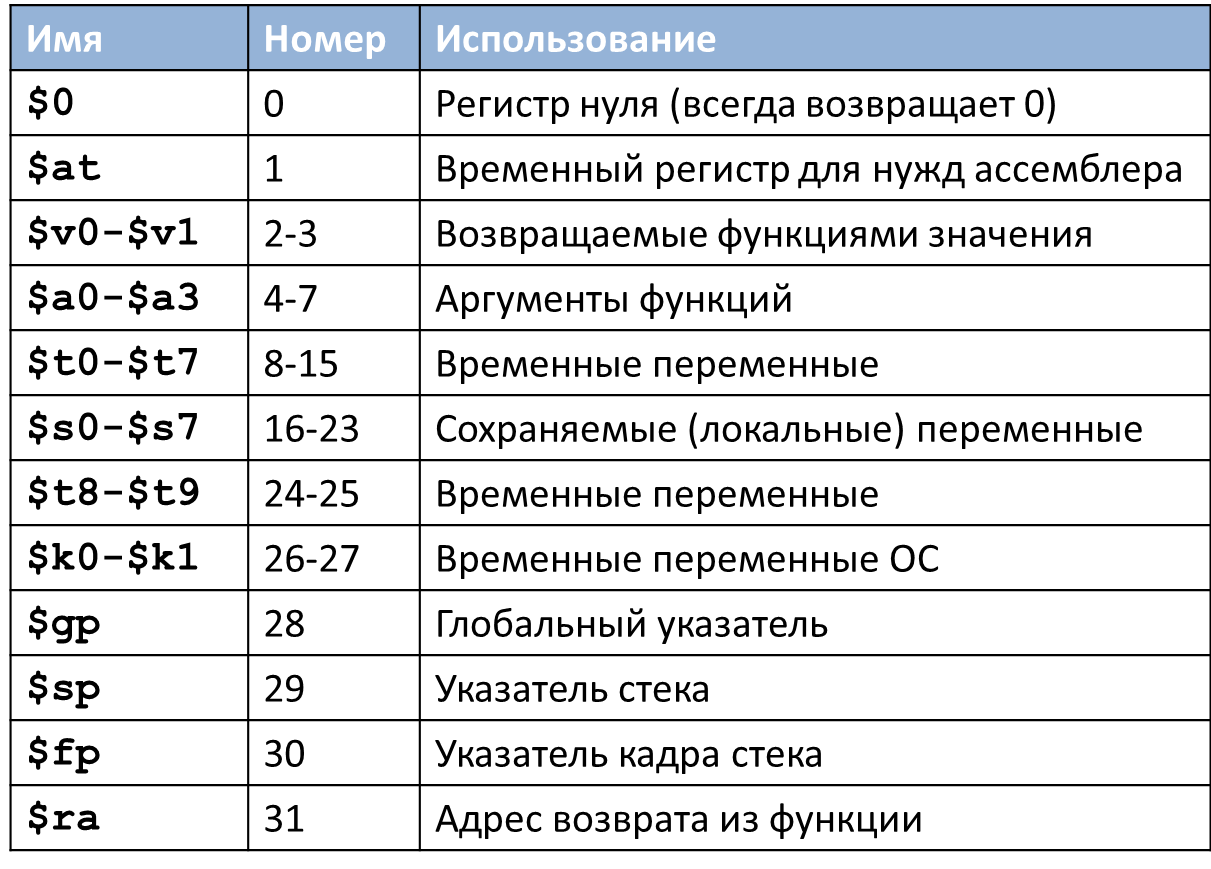
Рисунок 1. Отладочная плата DE10-Lite

Таблица 9. Элементы управления отладочной платы

|  |  |  |  |
| --- | --- | --- | --- |
| **№ на схеме** | **Обозначение на плате** | **Тип** | **Назначение** |
| **1** | **2** | **3** | **4** |
|  | HEX5-HEX0 | HEX-индикатор | Отображение значения текущего выбранного регистра (младшие разряды) |
|  | LED9-LED1 | LED-индикаторы | Отображение значения текущего выбранного регистра (младшие разряды) |
|  | SW9 | переключатель | Разрешить выполнение программы. При включенном переключателе сигнал с делителя тактового сигнала поступает в процессорное ядро |
|  | SW8- SW5 | переключатели | Группа переключателей настройки делителя тактового сигнала. Для плат с недостаточным количеством переключателей данная настройка может задаваться в коде специфического для платы модуля верхнего уровня |
|  | KEY0 | кнопка | Сигнал системного сброса |
|  | KEY1 | кнопка | Разрешить выполнение программы. При нажатой кнопке сигнал с делителя тактового сигнала поступает в процессорное ядро |
|  | LED0 | LED-индикатор | Индикатор тактового сигнала |
|  | SW4- SW0 | переключатели | Группа переключателей выбора текущего регистра общего назначения (0-31). Значение, хранящиеся выбранном регистре, выводится на HEX и LED-индикаторы. При выборе 0 - отображается значение счетчика команд |

Например, для того, чтобы вывести значение, хранящиеся в регистре $v0 необходимо установить на переключателях SW4- SW0 значение двоичного числа 5'b00010 (SW1 - в верхнем положении, SW4- SW2,SW0 - в нижнем). Что будет соответствовать 2 - порядковому номеру регистра $v0. Перечень регистров MIPS с указанием их номеров и назначения приведен ниже.

Таблица 10. Назначение и номера регистров MIPS



## Миграция и добавление кода

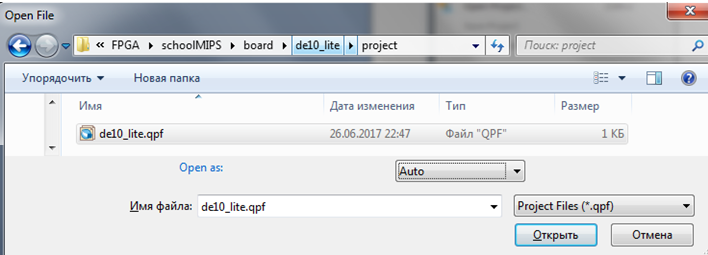
Проект открыт для добавления Вашего кода, вместе с тем, его использование в образовательном процессе накладывает определенные ограничения, выраженные в следующих принципах:

* простота и очевидность используемых языковых и архитектурных конструкций;
* бережное оформление кода и осмысленный подход к именованию сущностей;
* сохранение простора для самостоятельной работы обучаемых;
* разделение платформозависимого и платформонезависмого кода;
* предпочтение кода на Verilog графическому проектированию.

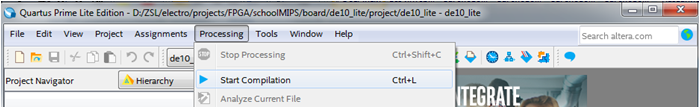
## ПРИЛОЖЕНИЕ 1. Пример синтеза и конфигурации ПЛИС Altera

1. Открытие проекта

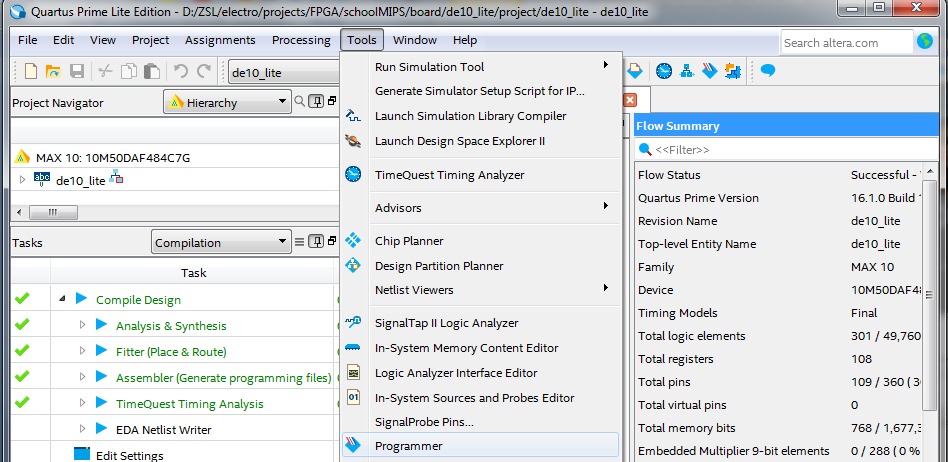




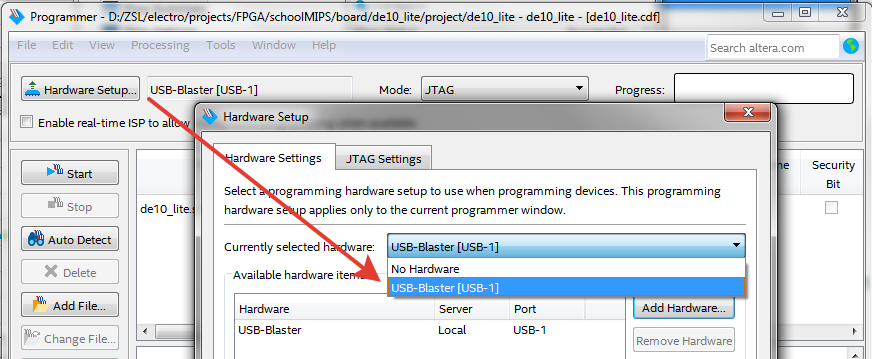
1. Запуск синтеза



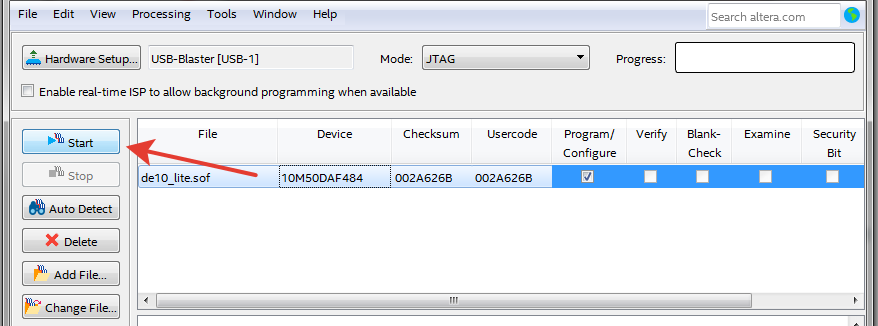
1. Запуск программатора



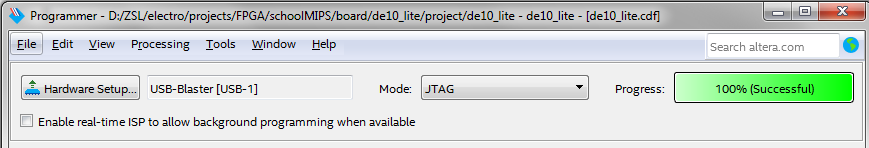
1. Настройка программатора



1. Выбор конфигурации для программирования ПЛИС, запуск процедуры



1. Процедура завершена

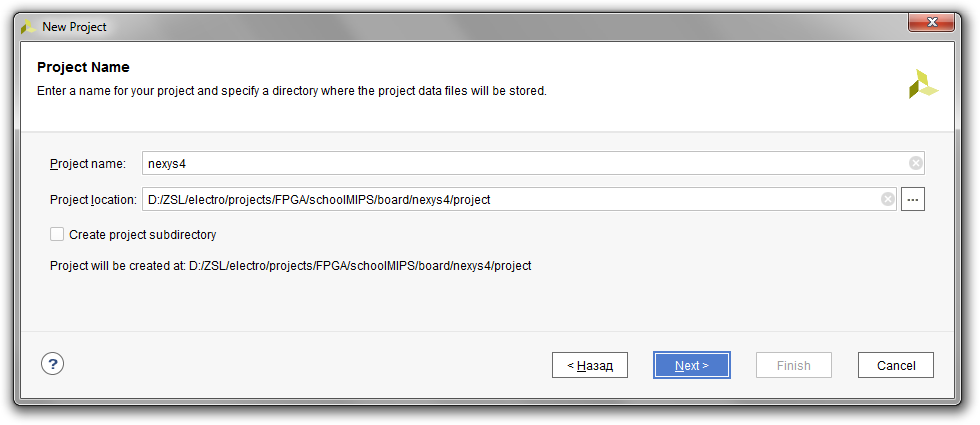


## ПРИЛОЖЕНИЕ 2. Пример синтеза и конфигурации ПЛИС Xilinx

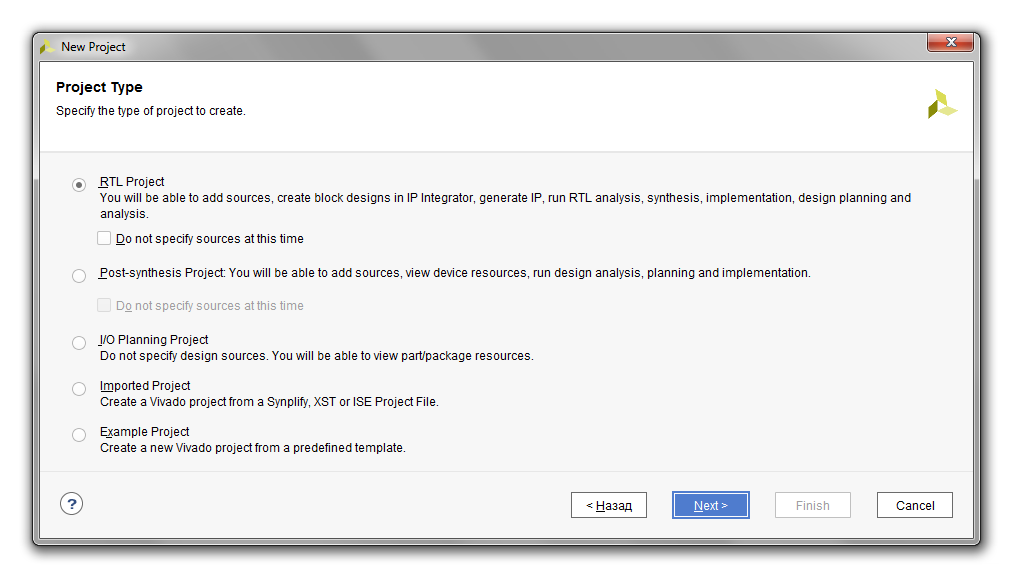
1. Создание проекта



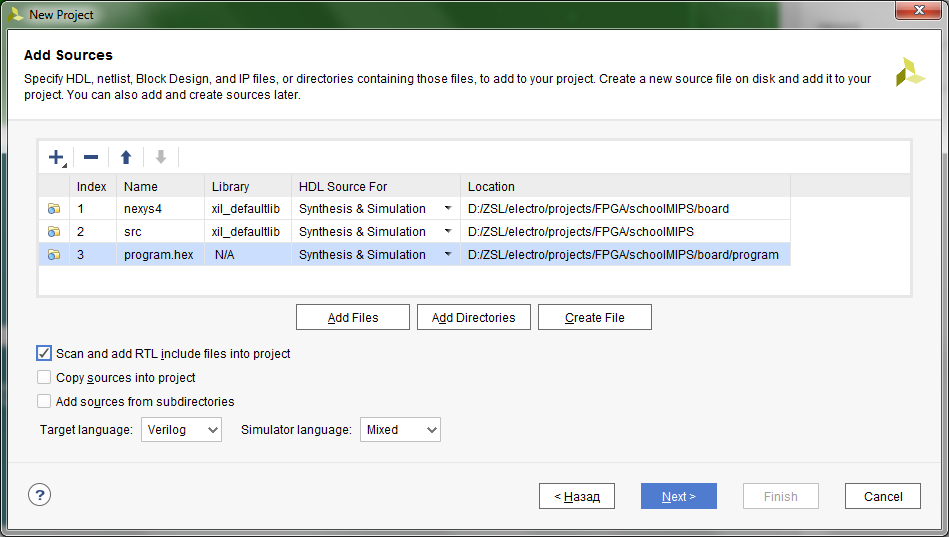
1. Установка имени и расположения



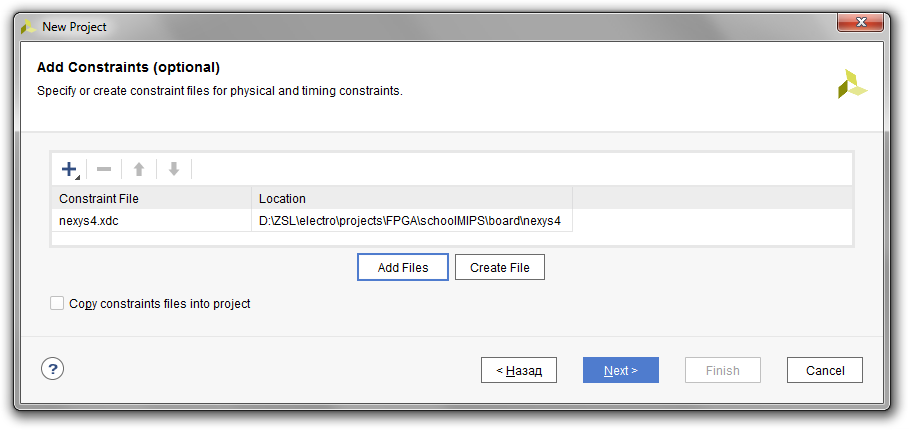
1. Установка типа проекта



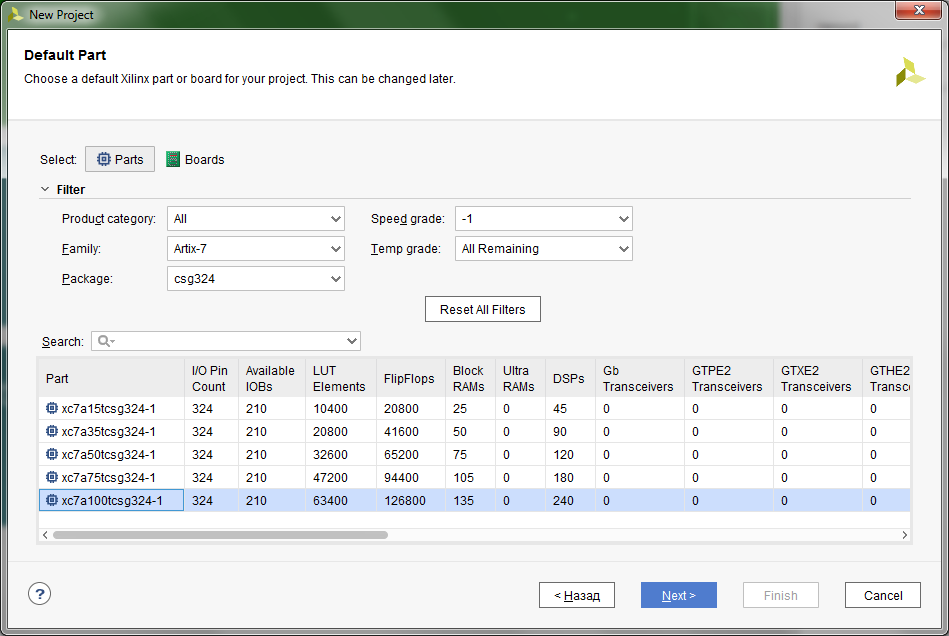
1. Настройка расположения файлов с исходным кодом



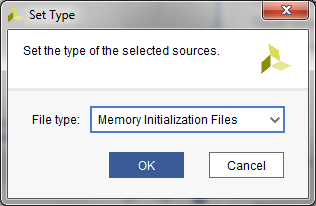
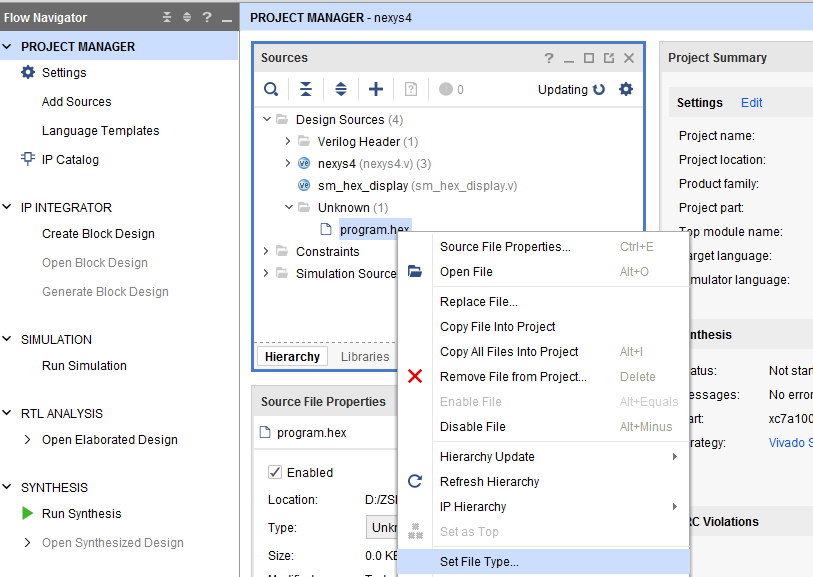
1. Настройка ограничений



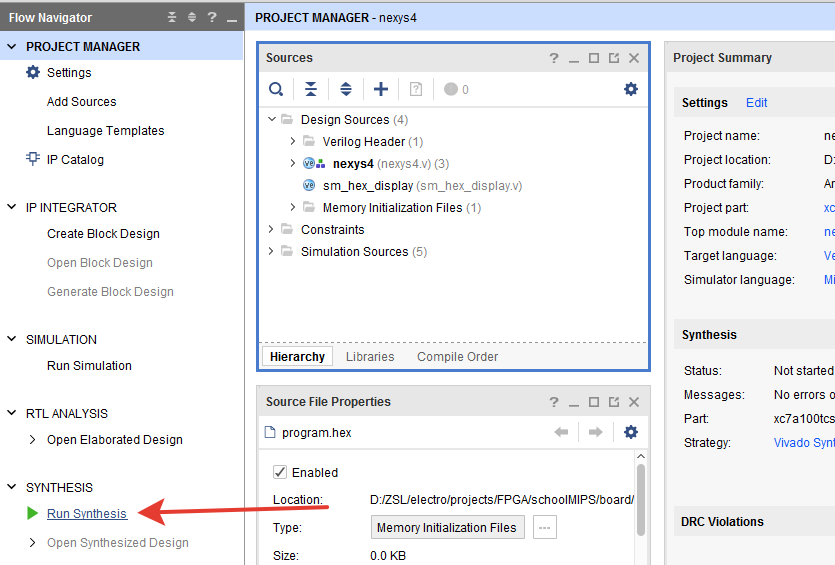
1. Настройка используемой ПЛИС

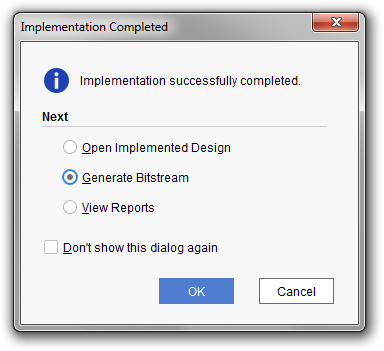
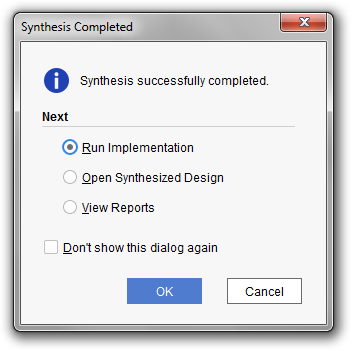


1. Установка типа для файла с дампом памяти

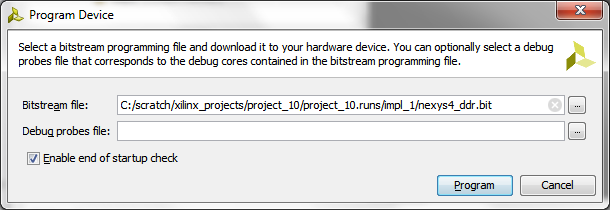


1. Запуск процедуры синтеза





1. Запуск процедуры программирования ПЛИС



1. https://geektimes.ru/post/289827/ [↑](#footnote-ref-1)
2. Таблицы приведены в соответствии с документом: MIPS32 microAptiv UP Processor Core Family Software User’s Manual [↑](#footnote-ref-2)