Версия документа: 0.3

Процессорное ядро schoolMIPS

Руководство пользователя

Станислав Жельнио

2017

Оглавление

Список таблиц

1. Благодарности

Автор выражает благодарность коллективу переводчиков учебника Дэвида Харриса и Сары Харрис «Цифровая схемотехника и архитектура компьютера», всем тем, кто принимает активное участие в подготовке учебного курса (группа "Young Russian Chip Architects": yrca@googlegroups.com), а также персонально Юрию Панчулу за его образовательные инициативы на территории стран СНГ.

2. Введение

schoolMIPS - это простейшее процессорное ядро, разработанное в рамках инициативы по преподаванию школьникам основ цифровой схемотехники, языков описания аппаратуры и использования ПЛИС[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;

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

3. Уровень поддержки MIPS архитектуры

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

Таблица 1. Поддерживаемые коды поля Opcode

opcode		bits 28..26							
		0	1	2	3	4	5	6	7
bits 31..29		000	001	010	011	100	101	110	111
0	000	Special	RegImm	J	JAL	BEQ	BNE	BLEZ	BGTZ
1	001	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2	010	COP0	β	COP2	β	BEQL	BNEL	BLEZL	BGTZL
3	011	α	α	α	α	Special2	ϑΑΛΞ	α	Σπεχιαλ3
4	100	LB	LH	LWL	LW	LBU	LHU	LWR	α
5	101	SB	SH	SWL	SW	α	α	SWR	CACHE
6	110	LL	β	LWC2	PREF	α	β	α	α
7	111	SC	β	SWC2	α	α	β	α	α

Таблица 2. Поддерживаемые коды поля Function

function		bits 2..0							
		0	1	2	3	4	5	6	7
bits 5..3		000	001	010	011	100	101	110	111
0	000	SLL	β	SRL/ ROTR	SRA	SLLV	α	SRLV/ ROTRV	SRAV
1	001	JR	JALR	MOVZ	MOVN	SYSCALL	BREAK	α	SYNC
2	010	MFHI	MTHI	MFLO	MTLO	α	α	α	α
3	011	MULT	MULTU	DIV	DIVU	α	α	α	α
4	100	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
5	101	α	α	SLT	SLTU	α	α	α	α
6	110	TGE	TGEU	TLT	TLTU	TEQ	α	TNE	α
7	111	α	α	α	α	α	α	α	α

[1] https://geektimes.ru/post/289827/

[2] Таблицы приведены в соответствии с документом: MIPS32 microAptiv UP Processor Core Family Software User's Manual

4. Текущий статус поддержки отладочных плат

Таблица 3. Текущий статус поддержки отладочных плат

№ п/п	Наименование платы	ПЛИС	Статус
1	2	3	4
1.	Terasic DE10-Lite	Altera MAX10	Поддерживается в полном объеме
2.	Terasic DE1-SoC	Altera Cyclone V SoC	Поддерживается в полном объеме

5. Структура проекта

Структура каталогов проекта, его модульный состав, а также перечень интерфейсных сигналов основного модуля приведены в таблицах ниже.

Таблица 4. Структура каталогов проекта

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

Таблица 5. Модульный состав проекта

№ п/п	Наименование модуля	Описание
1	2	3
Каталог src		
1.	sm_cpu	Основной модуль процессорного ядра
2.	sm_control	Процессорное ядро. Устройство управления
3.	sm_alu	Процессорное ядро. Арифметико-логическое устройство
4.	sm_register_file	Процессорное ядро. Регистры общего назначения
5.	sm_rom	Процессорное ядро. Память инструкций
6.	sm_register	32-битный регистр
7.	sm_register_we	32-битный регистр с вводом разрешения записи
8.	sm_hex_display	Адаптер вывода числа на HEX индикатор
9.	sm_clk_divider	Делитель тактовой частоты
Каталог testbench		
10.	sm_testbench	Основной модуль запуска в режиме симуляции
Каталог board / <наименование платы>		
11.	<top>	Модуль верхнего уровня, специфический для отладочной платы. Имя не регламентируется

Таблица 6. Интерфейсные сигналы основного модуля процессорного ядра (sm_cpu)

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

6. Порядок развертывания и запуска

6.1.Программное окружение

Перечень программных продуктов, необходимых (рекомендуемых) для работы с schoolMIPS, с указанием их версий, работоспособность на которых подтверждена, приведен ниже.

Таблица 7. Программное окружение

№ п/п	Наименование	Версия	Примечание
1	2	3	4
1.	ОС Windows	7	Работа на других версиях ОС - не тестировалась. Для миграции на ОС Linux необходима переработка скриптов, входящие в состав примеров программ
2.	Quartus Prime	16.1	Используется для синтеза и последующего запуска на отладочных платах
3.	ModelSim	10.5b	Используется для запуска в режиме симуляции. Задействована версия INTEL FPGA STARTER EDITION (Revision: 2016.10), входящая в состав Quartus Prime
4.	Codescape MIPS SDK	1.4.1.07	Используется для компиляции примеров программ и формирования hex-образов памяти инструкций
5.	Visual Studio Code	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"

6.2.Структура каталога тестовой программы

В таблице ниже приведена типовая структура каталога тестовой программы (program / <имя программы> / *) с описанием вложенных каталогов, файлов, их назначения и отдельных особенностей использования.

Таблица 8. Типовая структура каталога тестовой программы

№ п/п	Имя	Описание
1	2	4
1.	main.S	Файл с исходным кодом тестовой программы
2.	program.elf	Бинарный файл тестовой программы. Формируется компилятором в результате выполнения скрипта 01_compile_and_link.bat
3.	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
4.	program.ld	Скрипт компоновщика, определяющий состав program.elf. Используется скриптом 01_compile_and_link.bat
5.	program.dis	Результат дисассемблирования program.elf. Является результатом работы скрипта 02_disassemble.bat
6.	modelsim_script.tcl	Скрипт, задающий конфигурацию ModelSim и обеспечивающий запуск тестового примера в режиме симуляции. Используется скриптом 04_simulate_with_modelsim.bat
7.	sim	Временный каталог, внутри которого запускается ModelSim для последующей симуляции. Создается скриптом 04_simulate_with_modelsim.bat
8.	00_clean_all.bat	Скрипт. Выполняет удаление временных файлов и каталогов.

9.	01_compile_and_link.bat	Скрипт. Осуществляет сборку программы main.S и формирование program.elf. В своей работе использует program.ld
10.	02_disassemble.bat	Скрипт. Выполняет дизассемблирование program.elf, результатом работы является program.dis
11.	03_generate_verilog_readmemh_file.bat	Скрипт. Выполняет формирование program.hex из program.elf
12.	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
13.	05_copy_program_to_board.bat	Скрипт. Выполняет копирование program.hex в каталог board / program для его последующего использования при синтезе

6.3.Запуск в режиме симуляции

Для запуска в режиме симуляции 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
  ```
- запустить программу в симуляторе
  ```
  04_simulate_with_modelsim.bat
  ```

6.4.Синтез проекта и программирование ПЛИС

Для синтеза конфигурации ПЛИС и ее программирования 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
  ```
- 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).

Дальнейшие действия, выполняются с использованием кнопок и переключателей, доступных на плате. Порядок работы с ними описан в следующем разделе.

6.5.Интерфейс пользователя

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

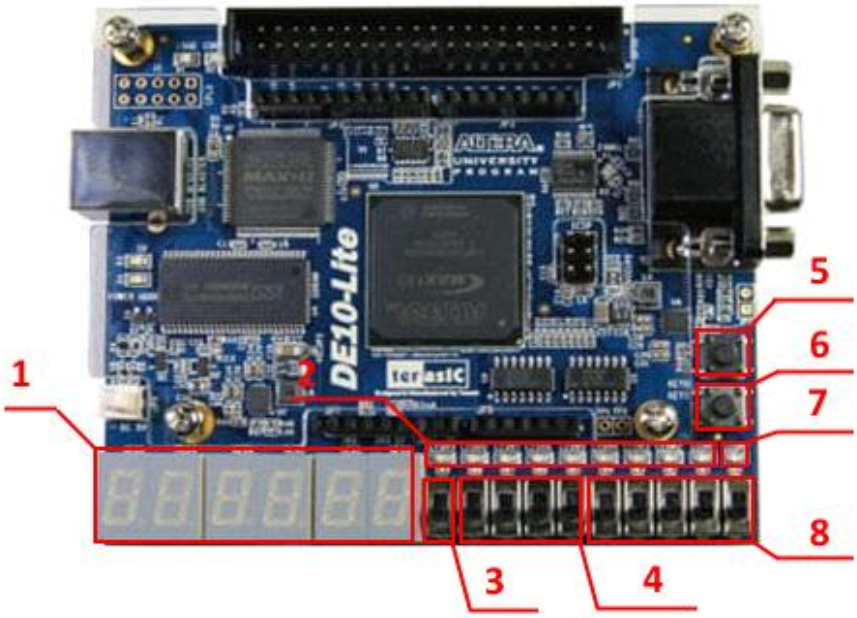

Рисунок 1. Отладочная плата DE10-Lite

Таблица 9. Элементы управления отладочной платы

№ на схеме	Обозначение на плате	Тип	Назначение
1	**2**	**3**	**4**
1.	HEX5-HEX0	HEX-индикатор	Отображение значения текущего выбранного регистра (младшие разряды)
2.	LED9-LED1	LED-индикаторы	Отображение значения текущего выбранного регистра (младшие разряды)
3.	SW9	переключатель	Разрешить выполнение программы. При включенном переключателе сигнал с делителя тактового сигнала поступает в процессорное ядро
4.	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
5.	KEY0	кнопка	Сигнал системного сброса
6.	KEY1	кнопка	Разрешить выполнение программы. При нажатой кнопке сигнал с делителя тактового сигнала поступает в процессорное ядро
7.	LED0	LED-индикатор	Индикатор тактового сигнала
8.	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

Имя	Номер	Использование
`$0`	0	Регистр нуля (всегда возвращает 0)
`$at`	1	Временный регистр для нужд ассемблера
`$v0-$v1`	2-3	Возвращаемые функциями значения
`$a0-$a3`	4-7	Аргументы функций
`$t0-$t7`	8-15	Временные переменные
`$s0-$s7`	16-23	Сохраняемые (локальные) переменные
`$t8-$t9`	24-25	Временные переменные
`$k0-$k1`	26-27	Временные переменные ОС
`$gp`	28	Глобальный указатель
`$sp`	29	Указатель стека
`$fp`	30	Указатель кадра стека
`$ra`	31	Адрес возврата из функции

7. 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.