ПРОГРАММИРОВАНИЕ ЗАГРУЗОЧНОЙ FLASH ДЛЯ ЗАПУСКА MICROBLAZE

Автор: KeisN13

Рецензенты:

Aspect

Материал подготовлен при поддержке:

КТЦ «Инлайн груп» - дистрибьютор фирмы Xilinx (<u>www.plis.ru</u>)

ATP Center Xilinx – Сертифицированный тренинг центр Xilinx (www.plis2.ru)

Оглавление

Аннотация	3
Введение	3
Способ 1	4
Генерация «правильного» bit файла	5
Склеивание bit+elf = bitelf	9
Создание конфигурационного mcs файла для загрузочной flash	14
Прошиваем FLASH	18
Способ 2	20
Последний штрих	23
Библиографический список	26
Список тренингов	26

Аннотация

В этой статье описаны варианты создания загрузочного файла для FLASH выполняющего запуск MicroBlaze при включении питания.

Введение

В предыдущей статье был рассмотрен процесс создания процессорной системы на базе софтпроцессора MicroBlaze в среде Vivado и Xilinx SDK. Однако, за рамками остался один интересный вопрос: «Как же нам заставить запускаться процессор при включении питания?» Иными словами, какие шаги мы должны предпринять, чтобы соединить вместе файл прошивки FPGA (.bit) и файл исполняемой программы MicroBlaze'a (.elf) в один общий файл прошивки FLASH памяти (.mcs или .bin)?

Как ни парадоксально, но ответ не сложным образом находится простым поиском в DocNav. Но если бы все было так просто

Цель статьи - показать способы создания/слияния bit и elf в единый mcs для Arty Board.

За основу мы с Вами возьмём наш проект из предыдущей статьи. Надеюсь, что у Вас все получилось нормально и всё заработало как следует. Пользоваться мы будем небольшими утилитами Vivado, доступ к которым получим через Tcl консоль.

Лирическое отступление: Tcl – мощнейший инструмент не только Vivado, да и вообще любой среды проектирования, программирования, прототипирования и m.д. И знакомиться Вам с ним придётся в любом случае, если Вы захотите освоить среду в полном объёме, в данном случае Vivado. Несмотря на графический интерфейс, не все возможности Vivado помещаются в него. Например одна из потряснейших вещей «частичная реконфигурация (Partial Reconfiguration)» до версии 2017.1 существовала только в виде консоли и скриптов Tcl. Да и в общем, иногда некоторые вещи удобно делать через консоль или же использовать какойлибо самопальный скрипт для выполнения тех или иных действий. Призываю всех познакомиться с Tcl консолью и языком Tcl. Синтаксис языка не сложный, а

3

литературы в интернетах по нему масса. Ниже приведены ссылки на соответствующие источники [1-3].

В этой статье, мы будем использовать Tcl скрипт, но оформим мы это в виде добавки к графическому интерфейсу Vivado, о которой, как ни странно, мало кто знает. Но об этом позже.

Я покажу как собрать единый mcs файл в Vivado. Это можно сделать и средствами SDK – о чем кто-либо из пользователей должен будет сделать соответствующую заметку на сайте ©.

Способ 1

Для начала нам необходимо прочитать главу «Using UpdateMEM to Update BIT files with MMI and ELF Data» из [4]-. Пожалуйста, обратите внимание, что документы от версии к версии софта меняются. Для конкретного случая глава «Using UpdateMEM to Update BIT files with MMI and ELF Data» в версии 2015.4 идёт под номером 6, а в версии 2017.1 под номером 7. Всегда выбирайте документы именно той же версии, что и Ваш софт.

Как Вы уже, наверное, поняли, из названия главы мы будем использовать утилиту updatemem, для которой нам понадобятся:

- Бистрим файл (.bit), который генерируется Vivado. Команда updatemem берет .bit файл и выдаёт .bit файл.
- Файл с информацией отображения памяти (.mmi) файл. Этот файл описывает то, каким образом ячейки блочной памяти сгруппированы вместе, чтобы создать единое непрерывное адресное пространство. Этот файл автоматически создаётся Vivado и лежит в папке с результатами имплементации (по умолчанию <project>.runs/imp_x) или он может быть сгенерирован, используя команду write_mem_info.
- Файл исполняемой программы MicroBlaze (.elf). Этот файл генерируется SDK.

• Номер процессора в системе (ID). Даже если у вас всего один процессор, путь к его иерархии должен быть указан.

О том, что содержит и для чего нужен каждый тип файла, Вы можете самостоятельно ознакомиться в [4].

Генерация «правильного» bit файла

Правильный bit файл – это такой bit файл, в котором указа информация о конфигурационной FLASH. Для того чтобы такой файл сгенерировать, нам понадобится как минимум информация о FLASH на ArtyBoard. Это можно найти в документации на ArtyBoard и это будет N25Q128A13ESF40 от компании Micron.

Но это информация пока избыточна, что нам точно надо знать так это то, что у нас стоит QSPI FLASH с возможность работы в режимах x1, x2, x4.

Занесём эти настройки в программу. Откройте имплементированный проект (Open Implemented Design) и после этого выберите Tools→Edit Device Properties (рис. 1).

À Microblaze_Les	sson_1	- [C:/Projects/FG	PA-Systen	ns/Microblaze_	Lesson
<u>File Edit Flow</u>	Tools	Window Layo	ut <u>V</u> iew	<u>H</u> elp	_
📄 🔠 🖬 🖓	1	Floorplanning		>	66 g
Flow Navigator	1	I/O Planning		>	<
o 👎 📥		Timing		>	
× 🗠 👳	<u>a</u>	Edit Ti <u>m</u> ing Constr	aints		
▲ Project Mana	24	Sch <u>e</u> matic		F4	
🚳 Project	1	Show Connectivity	(Ctrl+T	
👌 Add Sou	.	Show Hierarchy		F6	
💡 Languag		Report		>	
IP Cata		Edit Device Prope	ties		
	٠	Create and Packa	ge IP		
 IP Integrator 		Create Interface I	Definition		
🚟 Create I		Run Tcl Script			
😚 Open Bl	6	Property Editor		Ctrl+J	
🎨 Generat		Associate EL <u>F</u> File	s		
A Simulation		Compile Simulation	Libraries		
😚 Simulati		Xilinx <u>T</u> d Store			
📖 Run Sim		Customize Comma	nds	>	
4	6	Project Settings			-
RTL Analysis	*	Options			
🏀 Elabora	tion Set	tings			
Den El	aborate	ed Design			
A Synthesis					
Synthesis	is Setti	nas			
Run Svr	nthesis				
Den Si	vnthesi:	zed Design			
p D open of	, nancon	icu besign			
Implementation	ion				
🚳 Impleme	entation	Settings			
🕨 Run Imp	olement	ation			
Impleme	ented D	esign			
4. Program and De	hua				
Program and De Bitstrea	m Settir	206			
Ceneral	te Bitstr	eam			
	ardwar	- Manager			
p pen h	a uwdre	e manager			
Configure bitstream	proper	ties			

Рисунок 1 Вызов мастера настроек конфигурационного файла (д.б открыт имплементированный проект)

Появится окно конфигурации FLASH (рис.2) Настроек тут много, ознакомиться с ними предлагаю Вам самостоятельно, нажав на кнопочку Help внизу слева.

À Edit Device Properties					×
Use this dialog to edit the program	ming and configuration properties for your current design; default values are	set automatically.			4
Q- General Configuration Configuration Modes Startup Encryption Readback	General Bitstream Properties Enable Bitstream Compression Enable Qyclic Redundancy Checking (CRC) Enable debugging of Serial mode Bitstream Disable communication to the Boundary Scan (BSCAN) block via JTAG Enable JTAG Connection to XADC Enable Single Erame Cyclic Redundancy Checking (CRC) Enable Enhanced Linearity for XADC Enable XADC Power Down	FALSE ENABLE NO NO ENABLE NO ENABLE OFF VISABLE] c] c] c] c] c		
Help		CR	eset All	OK	Cancel

Рисунок 2 Окно настроек конфигурации

Нам потребуется только установить режим конфигурирования и скорость/частоту. Перейдите во вкладку Configuration и поставьте следующие значения (рис 3):

Configuration Rate (MHz) 33

🝌 Edit Device Properties		×
Use this dialog to edit the program	mming and configuration properties for your current design; default values are set automatically.	A
Q,-	Configuration	
General	Configuration Setup	^
Configuration Modes Startup	Configuration Rate (MHz) 33 C	
Readback	Enable external configuration clock and set divide value $\square \text{IISABLE} earrow \square C$	
	Configuration Voltage	
	Configuration Bank Voltage Selection	
	BPI Configuration	
	1st Read cycle 1 C	
	Page Size (bytes) 1 - C	
	Synchronous Mode DISABLE 👻 C	
	SPI Configuration	
	Enable SPI 32-bit address style	
	Bus width NONE 🔻 C	
	Enable the FPGA to use a falling edge dock for SPI data capture $\boxed{\text{NO}}$	
	MultiBoot Settings	
	<	>
Help	C' <u>R</u> eset All OK	Cancel

Рисунок 3 Окно настройки конфигурации: Configaration

Перейдите во вкладку Configuration Modes и выберите Master SPI x4 (рис.4)



Рисунок 4 Окно настройки конфигурации: Configaration Modes

Все остальное оставим по умолчанию и нажимаем кнопку ОК.

Запускаем генерацию битсрима Generate Bitstream и ждём окончания генерации.

Склеивание bit+elf = bitelf

Как я упоминал, мы будем использовать команду/утилиту updatemem. Как Вы знаете, Tcl консоль Vivado интерактивная – то есть если Вы набираете команду,

будет выдаваться команды, начало которых совпадает с введёнными символами (рис.5)



Рисунок 5 Интерактивное дополнение команд по введённым символам "get_" Однако если Вы будете вводить команду updatemem, то она не появится в списке доступных. Но это не должно Вас останавливать.



Рисунок 6 Отсутствие команды updatemem в консоли

Всё же если Вы проявите настойчивость и наберёте команду updatemem и нажмёте ввод, то увидите справку по утилите updatemem (рис. 7).

```
Tcl Console
X
     WARNING: [Common 17-259] Unknown Tcl command 'updatemem' sending command to the OS shell for execution.
¢
      Usage: updatemem [options]
(Switches with double dash '--' can also be used with a single dash '-')
æ
       -h [ --help ]
                               Display help
Input meminfo file with extension '.mmi'. This file
        --meminfo arg
痡
                               can be generated using the write_mem_info TCL command
                               in Vivado
        --data arg
                               Input elf or mem file to be used to populate the
                               BRAMs
                               Input bit file
        --bit arg
                              Instance path of the processor in the design
        --proc arg
                               Output bit file
        --out arg
        --force [=arg(=1)] (=0) Overwrite existing output bit file
        --debug
                               Hidden debug flag to output the bram init strings
     Examples:
       updatemem -meminfo top.mmi -data top.elf -bit top.bit -proc system i/microblaze -out top out.bit
       updatemem -meminfo top.mmi -data top.mem -bit top.bit -proc system_i/microblaze -out top_out.bit
       updatemem -meminfo top.mmi -data top.elf -bit top.bit -proc system_i/microblaze -out top_out.bit -force
   🛄 Tcl Console 💭 Messages 🛛 🖼 Log 📑 Reports 📑 Design Runs
```

Рисунок 7 Справка по команде updatemem

Ниже в справке Вы видите примеры. Собственно, их нужно просто повторить

Советую Вам открыть любой текстовый редактор и написать в нём команду и её опции (рис. 9). В качестве аргументов опций должны быть указаны пути к файлам. Я указал абсолютные пути.

Аргументом опции –proc будет положение процессора в иерархии. Чтобы корректно написать к процессору путь, нажмите Open Synthesized Design или Open Implemented Design, перейдите во вкладку Netlist, выберите процессор и скопируйте его имя (поле name в свойствах) (рис. 8)



Рисунок 8 Поиск названия процессора в иерархии проекта

Опция –out это то место, где будет располагаться склеенный bit+elf файл, который называется в данном случае bitelf.bit и располагается в папке impl_1, то есть там же где и исходный bit файл.

Полный листинг команды updatemem с соответствующими опциями показан на рис.9.

1	updatemen	Λ
2	-meminfo	{C:\Projects\FGPA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\system_wrapper.mmi}\
3	-data	{C:\Projects\FGPA-Systems\Microblaze Lesson 1\Microblaze Lesson 1.sdk\microblaze lesson 1\Debug\microblaze lesson 1.elf}
4	-bit	{C:\Projects\FGPA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\system_wrapper.bit}\
5	-proc	{system_i/microblaze_0}
6	-out	{C:\Projects\FGPA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\bitelf.bit}

Рисунок 9 Листинг команды updatemem

Наберите её в текстовом редакторе, после чего скопируйте в Tcl консоль Vivado, и если всё хорошо, то на выходе вы получите файл bitelf.elf, который будет находиться там, где Вы указали (путь – это аргумент опции -out) (рис.9). Пожалуйста, внимательно читайте log, который выдаёт команда updatemem в Tcl консоли Vivado. Ошибки, которые Вы допустили, не будут подсвечены, а просто будет черным текстом написано, что-то вроде этого (рис.10)

Td C	sole
	-out {C:\Projects\FGPA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\bitelf.bit}
(WARNING: [Common 17-259] Unknown Tcl command 'updatemem -meminfo C:\Projects\FGPA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.run:
	****** updatemem v2015.4 (64-bit)
đ	**** SW Build 1412921 on Wed Nov 18 09:43:45 MST 2015
	**** IP Build 1412160 on Tue Nov 17 13:47:24 MST 2015
1	** Copyright 1986-2015 Xilinx, Inc. All Rights Reserved.
	Sourcing tcl script 'C:/Xilinx/Vivado/2015.4/scripts/init.tcl'
	source C:/Xilinx/Vivado/2015.4/scripts/updatemem/main.tcl -notrace
	ERROR: [Updatemem 57-85] Invalid processor specification of: system_i/icroblaze_0. The known processors are: system_i/microblaze_0

Рисунок 10 Вариант вывода ошибки командой updatemem (название процессора указано не верно, не хватает буквы m)

Но в моём случае ошибок не возникло и сформированный склеенный файл

		v		(1	1	Υ.
появился в	соответству	/ющеи ди	ректории (рис. І	LI,)

📙 > Этот компьютер > Windows (C:) > Pro	ojects > FGPA-Systems	> Microblaze_Lesson_	1 > Microblaze_Lesson_1.runs > impl_1
Ами	Дата изменения	Тип	Размер
.Xil	10-Apr-17 11:55 A	Папка с файлами	
.init_design.begin.rst	10-Apr-17 11:53 A	Файл "RST"	1 КБ
.init_design.end.rst	10-Apr-17 11:53 A	Файл "RST"	0 КБ
.opt_design.begin.rst	10-Apr-17 11:53 A	Файл "RST"	1 КБ
.opt_design.end.rst	10-Apr-17 11:54 A	Файл "RST"	0 КБ
] .place_design.begin.rst	10-Apr-17 11:54 A	Файл "RST"	1 КБ
] .place_design.end.rst	10-Apr-17 11:54 A	Файл "RST"	0 КБ
.route_design.begin.rst	10-Apr-17 11:54 A	Файл "RST"	1 КБ
.route_design.end.rst	10-Apr-17 11:54 A	Файл "RST"	0 КБ
.vivado.begin.rst	10-Apr-17 11:53 A	Файл "RST"	1 КБ
.vivado.end.rst	10-Apr-17 11:55 A	Файл "RST"	0 КБ
.Vivado_Implementation.queue.rst	10-Apr-17 11:50 A	Файл "RST"	0 КБ
.write_bitstream.begin.rst	10-Apr-17 11:54 A	Файл "RST"	1 КБ
.write_bitstream.end.rst	10-Apr-17 11:55 A	Файл "RST"	0 КБ
bitelf.bit	19-Sep-17 11:38 A	Файл "BIT"	2,141 КБ

Рисунок 11 Сформированный bitelf.elf файл

Теперь осталось только создать файл прошивки FLASH памяти.

Создание конфигурационного mcs файла для загрузочной flash

Теперь создадим конфигурационный файл msc, которым прошьём FLASH.

Подключаем ArtyBoard к компьютеру. Открываем Hardware Manager (Open Hardware Manager). Выбираем Open Target, затем Auto Connect (рис.12).

Microblaze_Lesson_1 - [C:/Projects/FGPA-Systems/Microblaze_Lesson_1/F	Microblaze_Lesson_1.xpr] - Vivado 2015.4
File Edit Flow Tools Window Layout View Help	
😂 🔊 💷 🦬 🐘 🗙 🗞 🕨 🐂 🚳 🐝 ∑ 🧔 😐 Default La	yout 👻 🎉 🔖 🏌 📾 Dashboard 🛛 🕶 🚾 🛙 🕸
Flow Navigator «	Hardware Manager - unconnected
오, 🖫 🖨	No hardware target is open. <u>Open target</u>
🍪 Generate Block Design 🔷	Hardware Auto Connect
	State Content Targets
Simulation Simulation Settings	Available Targets on Server >
Simulation Settings	Dpen New Target
w Run Sinulation	
 A RTL Analysis 	
🚳 Elaboration Settings	No content
Den Elaborated Design	
4 Synthesis	
Synthesis Settings	
Run Synthesis	
Open Synthesized Design	Properties Z × ··
	← → 😚 k
4 Implementation	
6 Implementation Settings	
Run Implementation	
Implemented Design	
Constraints Wizard	Select an object to see properties
Edit Timing Constraints	
A Report Timing Summary	
Report Clock Networks	
Report Clock Interaction	
Report DRC	I d Console
Report Utilization	1. Make sure the clock connected to the debug hub (dbg hu
gur Report Power	2. Manually launch hw_server with -e "set xsdb-user-bscar
Program and Debug	close_hw
🚱 Bitstream Settings	close hw
🚵 Generate Bitstream	copen_hw
Hardware Manager	
📄 Open Target	<
Program Device	Type a Tcl command here
🗱 Add Configuration Memory Device	🛄 Tcl Console 💭 Messages 💊 Serial I/O Links 📃 Serial I/O Scans
Automatically connect to local hardware target	

Рисунок 12 Подключение Arty

После этого в списке подключённых устройств должна появиться FPGA, установленная на Arty (рис.13)

Hardware Manager - localhost/xilinx_tcf/Digilent/21	.0319789725A
(i) There are no debug cores. Program device Refr	esh device
Hardware	_ 🗆 🖻 ×
States and the states of th	
Name	Status
🖃 📲 localhost (1)	Connected
🖃 🎆 🤌 xilinx_tcf/Digilent/210319789725A (1)	Open
🖻 🚸 xc7a35t_0 (1)	Programmed
XADC (System Monitor)	

Рисунок 13 Список подключённых устройств

Нажимаем правой кнопкой по xc7a35t_0 и выбираем Add Configuration Memory Device ...(рис.14).



Рисунок 14 Добавление конфигурационного устройства, вызов мастера

Откроется очень большой список доступных устройств, в строке поиска наберите «3.3» и выберите «n25q128-3.3v-spi-x1_x2_x4» (puc.15).

👃 Add Configuration Memory Dev	/ice							×
Choose a configuration memor	y part. This can be	changed later.						
Device: 🛞 xc7a35t_0 Filter								
Manufacturer All		-			Type	All		Ŧ
Density (Mb) All		-			Width	All		-
Select Configuration Memory Part		(4 matches)						
Name	Part	Manufacturer	Alias	Family	Туре	Density (Mb)	Width	
🗯 n25q128-3.3v-spi-x1_x2_x4	n25q128-3.3v	Micron		n25q	spi	128	x1_x2_x4	
n25q256-3.3v-spi-x1_x2_x4	n25q256-3.3v	Micron		n25q	spi	256	x1_x2_x4	_
n25q32-3.3v-spi-x1_x2_x4	n25q32-3.3v	Micron		n25q	spi	32	x1_x2_x4	
暽 n25q64-3.3v-spi-x1_x2_x4	n25q64-3.3v	Micron		n25q	spi	64	x1_x2_x4	
							OK	Cancel

Рисунок 15 Выбор конфигурационной FLASH

Нажимаем ОК. После этого появится диалоговое окно для создания mcs файла. Нажимаем Cancel (рис.16).

Hardware Manage	r local	nost/xilinx_tcf/Diglent/	210319789725A		
There are no deb	ug core	s. Program.device. Re	fresh device		
Hardware			- 0 .	* ×	
९ 🛣 🕸 🛃 🗏	1	⇒ ■			
Name			Status		
🖃 📕 localhost (1)			Connected		
□ # Ø xilnx_tcf/	Digient,	210319789725A (1)	Open		
XADO	(Syste	m Monitor)	riogrammeu		
🤹 n25q	128-3.3	v-spi-x1_x2_x4_0			
Hardware Device Prop	perties	Add Configuration M	emory Device (Completed	×
← → >		-			Longer and
xc7a35t_0		O you want to p	rogram the confi	guration memory de	vice now?
1					
Name: x	c7a35	Don't show this dia	sog again		
Part: x	c7a351			OK	Cancel
ID code: 0	36200			- L	
IR length: 6					
Status: P	rogram	ned			
Programming file:			-	~	
General Propertie	s				

Рисунок 16 Окна запуска мастера создания mcs файла

А вот здесь наступает именно тот момент, который всех приводит в недоумение. Создание конфигурационного файла mcs в версиях до 2017.1 возможно только через Tcl консоль. Бред! Не правда ли?

Сделать mcs файл возможно с помощью команды write_cfgmem. У этой команды достаточно большая справка и большое количество аргументов. Вызвать справку можно написав в консоли write_cfgmem –help или обратившись к [2]

Я не буду Вас томить описанием всех параметров. Скажу, что для нашего случая достаточно указать «правильный» bit файл (bitelf.bit) и куда сохранить сгенерированный mcs (bitelf_mcs.mcs) файл. Размер нашей FLASH 16MB. Интерфейс мы выставили spi_x4. Опция –force нужна, чтобы можно было переписать файл при необходимости повторной генерации. Обратите внимание, что для опции –loadbit слэши в пути к файлу bitelf.bit стоят в другом направлении. Если стоит как обычно «\», то при интерпретации команды возможно появление ошибок.

Запишите команду в текстовом редакторе ниже команды updatemem, затем скопируйте команду write_cfgmem и её опции, вставьте в Tcl консоль Vivado и нажмите Enter

write_cfgmem\
-format mcs\
-size 16\
-interface SPIx4 \
-loadbit {up 0x00000000 "C:/Projects/FGPA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.runs/impl_1/bitelf.bit"}\
-force\
-file {C:\Projects\FGPA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\bitelf_mcs.mcs}

Рисунок 17 Листинг для формирования mcs файла

Проверяем папку, которую Вы указали в качестве выходной для write_cfgmem и проверяем наличие файла bitelf_mcs.mcs (рис. 18). В моём случае файл лежит в папке с результатами имплементации.

📙 > Этот компьютер > Windows (C:) >	Projects > FGPA-Systems	> Microblaze_Lessor	n_1 > Microblaze_Lesson_1.runs > imp
^	Дата изменения	Тип	Размер
📕 .Xil	19-Sep-17 12:42 PM	Папка с файлами	
🗋 .init_design.begin.rst	19-Sep-17 12:40 PM	Файл "RST"	1 КБ
init_design.end.rst	19-Sep-17 12:40 PM	Файл "RST"	0 КБ
🗋 .opt_design.begin.rst	19-Sep-17 12:40 PM	Файл "RST"	1 КБ
.opt_design.end.rst	19-Sep-17 12:41 PM	Файл "RST"	0 КБ
] .place_design.begin.rst	19-Sep-17 12:41 PM	Файл "RST"	1 КБ
] .place_design.end.rst	19-Sep-17 12:41 PM	Файл "RST"	0 КБ
.route_design.begin.rst	19-Sep-17 12:41 PM	Файл "RST"	1 КБ
.route_design.end.rst	19-Sep-17 12:41 PM	Файл "RST"	0 КБ
.vivado.begin.rst	19-Sep-17 12:40 PM	Файл "RST"	1 КБ
.vivado.end.rst	19-Sep-17 12:42 PM	Файл "RST"	0 КБ
.Vivado_Implementation.queue.rst	19-Sep-17 12:37 PM	Файл "RST"	0 КБ
.write_bitstream.begin.rst	19-Sep-17 12:41 PM	Файл "RST"	1 КБ
.write_bitstream.end.rst	19-Sep-17 12:42 PM	Файл "RST"	0 КБ
bitelf.bit	19-Sep-17 12:45 PM	Файл "BIT"	2,141 КБ
bitelf_mcs.mcs	19-Sep-17 2:33 PM	Файл "MCS"	6,022 КБ

Рисунок 18 Сгенерированный тся файл

Прошиваем FLASH

Осталось только прошить FLASH сгенерированным bitelf_mcs.mcs файлом.

Для этого нажимаем правой кнопкой мыши на FLASH в списке устройств и выбираем Program Configuration Memory Device (рис.19).

Bit Dit <th>A Microblaze_Lesson_1 - [C:/Projects/FGPA-Systems/Microblaze_Lesso</th> <th>_1/Microblaze_Lesson_1.xpr] - Vivado 2015.4 — 🗇</th> <th>× 📲</th>	A Microblaze_Lesson_1 - [C:/Projects/FGPA-Systems/Microblaze_Lesso	_1/Microblaze_Lesson_1.xpr] - Vivado 2015.4 — 🗇	× 📲
Intervention Intervention	<u>File Edit Flow Tools Window Layout View Help</u>	Q ₇ Search commands	
The Unequer Manager Index Mager	😂 🕼 💷 🗎 📉 🔌 🕨 🚵 🚳 💥 🔽 🎯 😬 Defa	it Layout 🔹 🎉 🗞 🔪 Sashboard 🗸 🔤 🛄 🚳 🛛 write_bitstream Complet	e 2
Image: State State State State State Source State State State State Source State State State <	Flow Navigator	Kardware Manager - localhost/xilinx_tcf/Diglent/210319789725A	🗙 📴 Bxc
Generat Bok Despi Sindaton Generat Bok Despi Sindaton Generat Bok Despi Sindaton Strings Kit Advises Generate Bok Despi Sindaton Strings <td>🔍 🖫 🌐</td> <td>There are no debug cores. Program device. Refresh device</td> <td>S Sky</td>	🔍 🖫 🌐	There are no debug cores. Program device. Refresh device	S Sky
 Sundaton Sundaton Sundaton K.R. Andysis Sundaton Setting: Substaton Setting:<td>🎨 Generate Block Design</td><td>A Hardware _ □ ℓ ×</td><td>🖉 🖉 Art</td>	🎨 Generate Block Design	A Hardware _ □ ℓ ×	🖉 🖉 Art
Sublexit Suble	 Simulation 		
	🔞 Simulation Settings	Name Status	v iei.
• R.R. Analysis • B.R. Analysis • B.R. Analysis • B.Boorston Settings • Sorthersis Settings • Sorthersis Settings • Programmed • Progr	(Run Simulation	Brit localist (1) Connected	🕞 Stic
<pre>* NL Auguss @ Boots to Strugs } WAC (Synthes Notion) @ Cond Bioseties Catings } Wat (Synthesis Strugs } Wat (Synthesis St</pre>	4 DTI Analusia		📙 im.
December allowated Design Synthesis Synt	KIL Analysis Elaboration Sattings	- S XADC (System Monitor)	
Synthesis Syn	Claborated Design	n25q1283.3/spix1_222 G Configuration Memory Device Properties Ctrl+E	HO.
<pre>Synthesis @ Synthesis Sottags @ Ren Synthesis Sottags @ Ren Synthesis Desgn @ Implementation @ Implementation @ Implementation @ Implementation @ Implementation @ Implementation @ Report Thing Constraints @ Report Thing Constraints</pre>	p 🔔 oper caborace beagn	- Remove Delete	📙 scri
Synthesis Settings Pure Synthesis P	 Synthesis 	Program Configuration Memory Device	🛃 Xili
P kun Synthesis P kun Inglementation P kun Ingle	🔞 Synthesis Settings	Export to Spreadsheet	
> Implementation Implementation Memory Device Properties Implementation • Implementation • • • • • • • • • • • • • • • • • • •	Run Synthesis		uga uga
 Implementation Implementation Settings Report Dack Interaction Report	Open Synthesized Design	Configuration Memory Device Properties _ D C × 0	💓 Mie
Implementation Settings Implementat	4 Implementation		The Viv
Rus Implementation Rus Implementation <	🔞 Implementation Settings	[4 r25q128-3.3v-spi-x1_x2_x4_0	
Memory Part: @n25q128-3.3 vspix1, v2_x4 Memory Arit: [n2 - 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0	Run Implementation	Name: n25q128-3.3v-spi-x1_x2_x4_0	A Mic
Constraints Wized Constraints Wized Constraints	a 🔛 Implemented Design	Memory Part: @ n25q128-3.3v-spi-x1_x2_x4 -	📑 *ne
Add Triming Constraints Apport Their Summary By Report Dook Networks Report Dook Repo	🚴 Constraints Wizard	Memory type: spi	🛷 Бег
Report Taning Summary Report Taning Summary Report Clock Interaction Report Clock Interaction Report Utilization Report Utilization Report Utilization Report Utilization Report DRC Report DR	A Edit Timing Constraints	Memory density: 128	
By Report Dock Networks Report Dock Networks <td>🍏 Report Timing Summary</td> <td>Programming file:</td> <td>The Viv.</td>	🍏 Report Timing Summary	Programming file:	The Viv.
Report DRC Pogram and Debug Contract Report DRC Pogram and Debug Contract Report DRC Report DRC Pogram and Debug Contract Report DRC Pogram Device Pog	⊞ ▶ Report Clock Networks		🔍 Ho
Constant and Debug Constant Startem Constartem Constant Startem Constant	Report Clock Interaction		
Report UMization Program and Debug Go Bitware Manager Cover ate Bitware	Report DRC		
<pre>% Report Power Program and Debug Addr1 Addr2 Date File(s) Cx/Projects/FGRA-Systems/Microblase_Lesson_1/Microblase_Lesson_1.runs/impl_//bitelf.bit Addr1 Addr2 Date File(s) Cx/Projects/FGRA-Systems/Microblase_Lesson_1.Microblase_Lesson_1.runs/impl_//bitelf.bit Addr1 Addr2 Date File(s) Open Target Program Device Add Configuration Memory Device Console @ Messages % Serial I/O Link @ Serial I/O Scans Console @ Messages % Serial I/O Scans Console @ Messages</pre>	Report Utilization	Td Console _ D 2	×
Program and Debug Program and Program Program and Program Device	8 Report Power	and the second s	
Image: Status	Program and Debug	Auli Auliz Date rile(3) Auli Auliz Date rile(3) Auli Auliz Date rile(3) Auli Auliz Date rile(3)	– 🤷
Image: Series I/O Scans Image: Series I/O Scans Image: Series I/O Scans Image: Series I/O Scans	🔞 Bitstream Settings	• write_cfgmen\	
Add Configuration Memory Device Add Configuration Memory Device Constant A configuration Memory Device Configuration Memory Device Constant A configuration Memory Device Configuration Memory Device Constant A co	🐏 Generate Bitstream		i e i
Program Device Add Configuration Memory Device Consult (Inc. Consult) Consult (Inc. C	4 📰 Hardware Manager	interface SPIx4 \	Φ ⁽¹⁾
Image: Second Laboration Memory Device Type a Tc1 consuland here 239 P Image: Total Console Image: Second L/O Canas 239 P	📄 Open Target	> · · · · · · · · · · · · · · · · · · ·	ENG
🎼 Add Configuration Memory Device 📄 📜 Tcl Console. 🖉 Messages 🔩 Serial I/O Links 🛄 Serial I/O Cons	Program Device	Type a Tcl command here	
	🕸 Add Configuration Memory Device	E Console Massages Serial I/O Links E reial I/O Scans	2:39 Pt

Рисунок 19 Запуск мастера программирования конфигурационной FLASH

В мастере указываем путь к файлу bitelf_mcs.mcs и нажимаем ОК.

Обратите внимание, что джампер JP1 (mode) на Arty должен быть установлен.

A Program Configuration Memo	ny Device		×
Select a configuration file and set pro	gramming options.		4
Memory Device: Configuration file: PRM file:	n25q128-3.3v-spi-x1_x2_x4 C:/Projects/FGPA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.runs/impl_1/bitelf_mcs.mcs Pull_pope		
Program Operations Address Range: Config Erase Blank Check Program Verify Verify Checksum	uration File Only 💌		
SVF Options Create <u>S</u> VF Only (no programs) SVF File:	am operations)		
	ОК	Cancel	Apply

Рисунок 20 Мастер настроек конфигурации

После окончания программирования FLASH можно либо переподключить USB кабель, либо нажать кнопочку PROG (она находится недалеко от USB разъёма рядом со светодиодом LD8 (DONE)).

Если все сделано корректно, то вы увидите мигающий светодиод, а если ещё и настроите терминал COM порта, то и сообщение Hello World, которое появляется каждый раз, как только загрузится FPGA (либо переподключение питания, либо кнопка PROG).

Способ 2

Рассмотрим другой способ. Он заключается в том, что пользователь сам присоединяет исполняемый файл программы к процессору через графический интерфейс или Tcl команду.

Для этого выберете Project Manager→в иерархии найдите block design→ правой кнопкой→Associate ELF Files



Рисунок 21 Запуск мастера присоединения исполняемых файлов

После этого откроется мастер присоединения исполняемых файлов. Для нашего процессора необходимо указать файл elf. Добавьте его самостоятельно, следуя инструкциям на рис. 22. Пути к файлам у Вас будут соответственно свои.



Рисунок 22 Окно добавления файла исполняемой программы elf

Нажмите ОК, и после этого вы увидите команду в Tcl консоли, набрав которую вы можете выполнить тоже самое, что проделали только что – ассоциирование elf файла с процессорной системой. Сохраните эту команду в новый текстовый файл (рис. 23).



Рисунок 23 Команды для выполнения ассоциации elf и процессора. Сохраните их в текстовом файле. Они пригодятся позднее.

После ассоциирования, нужно перегенерировать bit файл. Нажмите кнопку

Generate Bitstream и дождитесь окончания операции. После нажатия на копку 21

Generate Bitstream появятся команды в Tcl консоли (рис. 24), сохраните их в текстовый файл ниже команд ассоциирования.



Рисунок 24 Команды ассоциирования и генерации битерима в Tcl консоли. Добавьте команды генерации в текстовый файл ниже команд ассоциирования

Наш bit файл был сгенерирован и он уже содержит в себе информацию из elf. Мы выполнили склеивание этих файлов. Теперь повторяем процедуру write_cfgmem, но только вместо файла bitelf.bit нужно указать только что сгенерированный bit файл (system_wrapper.bit), который называется так же, как и топ модуль нашего проекта (system_wrapper) (рис.25)

Microblaze_Lesson_1 - [C:/Projects/FGPA-Systems/Microblaze_Lesson]	📕 🗹 📕 = Средства работы с	рисунками impl_1		- 0	×	
le Edit Flow Tools Window Layout New Help		Файл Главная Поделиться Вид Управлении	è			~ 🔮
🛃 🕼 🖉 🔚 🏬 🗙 🖻 🕨 🛸 🚳 % 🔽 🧐 😬 Det	fault Layout 👻 🛒 🐟 🦎 🚾 🚾 🧐		V Magnuth *	St 🕞 💽	-	
Flow Navigator «	Project Manager - Microblaze_Lesson_1		х удалить	fi- 🚩 🛛		
2、 🖀 🚊	Sources _ 🗆 🖒	Закрепить на панели Копировать Вставить 👔 🤷 Копировать в 🔻	Переименовать Созда паля	зть Своиства ку 👻 🙆	Выделить	
^	🔍 🎞 🛱 😂 📸 📓 🛃	Буфер обмена Упоряд	очить Сс	здать Открыть		
Project Manager Operate Settinger	🖓 🗁 Denign Courses (0)	← → ✓ ↑ 📕 « Microblaze_Lesson_1.runs > impl_1 >		✓ Ů Поиск imp	ol 1	P
	system_wrapper: STRUCTURE (system_wrapper.vhd) ()	<u>^</u>				<u> </u>
	with the system of the sy	RWN	Дата изменения	Тип	Размер	
Language remplates	EF (2)	📈 htr.txt	20-Sep-17 1:09 PM	Файл "ТХТ"	1 KE	
- If Calaby	Gonstantis (a) Gonstantis (a)	init_design.pb	19-Sep-17 12:40 PM	Файл "РВ"	10 KE	
4 IP Integrator		📓 ISEWrap.js	20-Sep-17 1:09 PM	файл JavaScript	8 KB	
ở Create Block Design		ISEWrap.sh	20-Sep-17 1:09 PM	Файл "SH"	2 KB	
😚 Open Block Design		opt_design.pb	19-Sep-17 12:41 PM	Файл "РВ"	13 KE	
🍪 Generate Block Design	Hierarchy IP Sources Libraries Compile Order	place_design.pb	19-Sep-17 12:41 PM	Файл "РВ"	24 KB	
4 Cincipation	& Sources V Templates	project.wdf	20-Sep-17 1:09 PM	Файл "WDF"	5 KB	
Simulation Settings	Source File Properties _ D L	noute design.pb	19-Sep-17 12:41 PM	Файл "РВ"	11 KB	
Run Simulation		Сгенерированный bit фа	ЙЛ ^{р-17 1:09 РМ}	файл JavaScript	2 KB	
	🛞 system_wrapper.vhd		20 Jup-17 1:09 PM	Пакетный файл	1 KE	
RTL Analysis	Location: C: /Projects/EGPA-Systems/Microblaze Lesson 1/Microb	📓 runme.log	20-Sep-17 1:10 PM	Файл "LOG"	56 KB	
🔞 Elaboration Settings		l runme.sh	20-Sep-17 1:09 PM	Файл "SH"	2 КБ	
Open Elaborated Design	Parts vc7a25forca24.1	system_wrapper.bit	20-Sep-17 1:10 PM	Файл "ВІТ"	2,141 KB	
4 Synthesis	Part: Acrossicsystem	System_wrapper.hwdef	19-Sep-17 12:41 PM	Файл "HWDEF"	26 КЬ	
Synthesis Synthesis Settings	Library: xi_detaulub	i system_wrapper.mmi	20-Sep-17 1:10 PM	Файл "ММІ"	2 КБ	
Run Synthesis	General Properties	system_wrapper.sysdef	19-Sep-17 12:42 PM	Файл "SYSDEF"	94 КБ	
Open Synthesized Design		system_wrapper.tcl	20-Sep-17 1:09 PM	ActiveTcl Script	3 Kb	
· · · · · · · · · · · · · · · · · · ·	Td Console	system_wrapper.voi	20-Sep-17 1:10 PM	Virtual Disk image	0 Kb	
4 Implementation	set_property SCOPED_TO_REF "" [get_files -all]	system_wrapper_bd.bmm	20-Sep-17 1:10 PM	Файл "ВММ"	3 KD	
🔞 Implementation Settings	set property SCOPED_TO_REF system [get_files	system_wrapper_clock_utilization_routed	19-Sep-17 12:41 PM	Файл кмі	17 KD	
Run Implementation	<pre>set_property SCOPED_TO_CELLS { microblaze_0 }</pre>	system_wrapper_control_sets_placed.pt	19-Sep-17 12:41 PW	Фаил КРТ	29 ND	
4 🔛 Implemented Design	g reset_run impl_1 -prev_step	gistem wrapper_drc_opted.pt	19-Sep-17 12/41 FW	Файл пр	2 ND 1 VE	
Sconstraints Wizard	[Wed Sep 20 13:09:46 2017] Launched impl 1	System_wrapper_urc_routed int	10-Sep-17 12/41 DM	Файл гр	2 KE	
A Edit Timing Constraints	Run output will be captured here: C:/Projects	system_wrapper_orc_roacca.pc	19-Sep-17 12:41 PM	Файл "RPT"	2 NO 81 KE	
Report Timing Summary		system_wrapper_optices.pt	19-Sep-17 12-41 DM	Vivado Checknoint	1 171 K5	
♣ Report Clock Networks	<	system_wrapper_opcucp	10-Sep-17 12-41 DM	Vivado Checkpoint	1.459 KE	
Report Clock Interaction	Type a Tcl command here	system_wrapper_processory	10-Sep-17 12/41 DM	Main "RDT"	84 KE	
	📟 Tcl Console 💿 Messages 🛛 🕄 Log 🗋 Reports 📑 Design	system wrapper_power_router	19-Sep-17 12:41 PM	Файл "PR"	1 K5	
			1	WORKER FD		_ '

Рисунок 25 Топ модуль и сгенерированный bit файл

Таким образом команда write_cfgmem примет следующий вид (рис. 26).

```
#Создание mcs файла
write_cfgmem\
-format mcs\
-size 16\
-interface SPIx4 \
-loadbit {up 0x00000000 "C:/Projects/FGPA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.runs/impl_1/system_wrapper.bit"}\
-force\
-file {C:\Projects\FGPA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\bitelf_mcs.mcs}
```

Рисунок 26 Команда write_cfgmem для способа 2 генерации mcs

Запустите write_cfgmem команду в Tcl консоли Vivado, и убедитесь, что файл

bitelf_mcs.mcs появился или обновился в соответствующей папке.

Поместите write_cfgmem в текстовый файл ниже команд запуска генерации битерима. Сохраните файл в удобное для Вас место. Название может быть любым без русских букв, но расширение у него должно быть .tcl. Например, mem_gen.tcl. Таким образом должен получиться файл со следующим содержимым.

```
1 #accounupobahue
2 set_property SCOPED_TO_REF "" [get_files -all -of_objects [get_fileset sources_1] {C:/Projects/FGPA-Systems/Microblaze_Lessor
3 set_property SCOPED_TO_CELLS { } [get_files -all -of_objects [get_fileset sources_1] {C:/Projects/FGPA-Systems/Microblaze_Les
4 set_property SCOPED_TO_REF system [get_files -all -of_objects [get_fileset sources_1] {C:/Projects/FGPA-Systems/Microblaze_Les
5 set_property SCOPED_TO_CELLS { microblaze_0 } [get_files -all -of_objects [get_fileset sources_1] {C:/Projects/FGPA-Systems/Microblaze_Les
6 set_run impl_1 -prev_step
9 launch_runs impl_1 -to_step write_bitstream -jobs 4
6 #Cosnanue mcs @aŭna
7 #format mcs\
6 -interface SPIX4 \
6 -loadbit {up 0x0000000 "C:/Projects/FGPA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.runs/impl_1/system_wrapper.bit"}\
6 -force\
7 -force\
6 -file {C:\Projects\FGPA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\bitelf_mcs.mcs}
```

Рисунок 27 Содержимое файла mem_gen.Tcl

Далее можете запрограммировать FLASH память на Arty и убедиться, что все получилось. Долже замигать светодиод, как и в прошлый раз.

Последний штрих

Я не сомневаюсь, что у многих возникла одна мысль: «Это типа так каждый раз что ли делать? Писать этот скрипт и копировать? Хрень какая-то!!!» И тут конечно возразить не чего. Но, вот тут нас может выручить одна полезная фича Vivado – можно создавать собственные кнопки в интерфейсе, по нажатию на которые будет выполняться какой-то скрипт или команда.

Помните я говорил Вам записать команды в текстовый файл? Его нужно дополнить всего одной командой, и вот почему. После команды генерации битсрима, Vivado начнёт его генерировать, но Tcl не будет ждать пока закончится этот процесс, а сразу запустит генерацию mcs файла, что приведёт к ошибке, потому что файл bit ещё не успел сгенерироваться. Поэтому нам нужно сказать Tcl «подожди пока сгенерируется bit и только после этого запускай генерацию mcs». Делается это с помощью команды wait_on_run impl_1. Эту команду нужно поместить между командами генерации битсрима и файла mcs. Таким образом полный листинг файла mem_gen.tcl приведён на рис.28.

1	#ассоциирование
2	set property SCOPED TO REF "" [get files -all -of objects [get fileset sources 1] {C:/Projects/FGPA-Systems/Microblaze Lesson 1
3	set property SCOPED TO CELLS { } [get files -all -of objects [get fileset sources 1] {C:/Projects/FGPA-Systems/Microblaze Lesso
4	set_property SCOPED_TO_REF system [get_files -all -of_objects [get_fileset sources_1] {C:/Projects/FGPA-Systems/Microblaze_Less
5	<pre>set_property SCOPED_TO_CELLS { microblaze_0 } [get_files -all -of_objects [get_fileset sources_1] {C:/Projects/FGPA-Systems/Mic</pre>
6	
7	#Генерация Bitstream
8	reset_run impl_1 -prev_step
9	launch_runs impl_1 -to_step write_bitstream -jobs 4
10	
11	#ждем окончания генерации bit файла
12	wait_on_run impl_1
13	
14	#Создание mcs файла
15	write_cfgmem\
16	-format mcs
17	-size 16\
18	-interface SPIx4 \
19	-loadbit {up 0x00000000 "C:/Projects/FGPA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.runs/impl_1/system_wrapper.bit"}
20	-force\
21	<pre>-file {C:\Projects\FGPA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\bitelf_mcs.mcs}</pre>

Рисунок 28 Окончательный листинг файла mem_gen.tcl

Открываем Vivado, нажимаем Tools→Customize Commands→ Customize

Commands.



Рисунок 29 Листинг команд для генерирования склеенного bit+elf и mcs файлов Откроется окно настройки кнопки, нажмите на плюсик, чтобы добавить кнопку, название кнопки «mem_gen», в качестве источника укажите файл 24 mem_gen.tcl, при необходимости можете добавить иконку и сделать описание (рис.30).

À Customize Commands			×		
Configure custom Tcl menu and toolbar entries.			4		
Custom Commands	Edit Custom Command				
teu mem_gen	Menu name:	mem_gen	۲		
†	Shortcut:	No shortcut	Add Remove		
+	○ Ru <u>n</u> command: ④ <u>S</u> ource Tcl file:	Type a Tcl command here	Define Args		
		: C:/Projects/FGPA-Systems/Microblaze_Lesson_1/mem_gen.td 📀			
	Toolbar Options				
	Add to the	e toolbar			
	Tooltip:	Сгенерировать MCS	8		
	Icon file path:	CL			
		Apply Reset			
			OK Cancel		

Рисунок 30 Меню создания кнопки в Vivado

Нажимаем ОК и теперь кнопка появилась в панели инструментов (рис. 31)

A Microblaze_Lesson_1 - [C:/Projects/FGPA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.xpr] - Vivado 2015.4				
<u>File Edit Flow Tools Window Layout View Help</u>		\sim		
😂, 🖿 📭 💵 🐘 🗙 🛷 🕭 🕨 🐂 🖓 ⊘	6ð 🖗	📡 🚳 😬 Default Layout 🔹 🗶 🗽 🌾		
Flow Navigator	~	Implemented Design - xc7a35ticsg324-1L (active)		
🔍 🔀 🌲		Netlist _ 🗆 🖻 🗶		
	^	🔀 🔄 🛃		
4 Project Manager		System wrapper		
noject Settings		⊕- Can Nets (7)		
🔂 Add Sources		Carl Cells (3)		
🖓 Language Templates		ET a system (system)		
IP Catalog				

Рисунок 31 Кнопка пользователя в панели Vivado

Теперь, нет необходимости каждый раз копировать скрипты, просто нажимаете кнопку и получите mcs файл.

Не забывайте обновлять msc после изменений в elf файле.

Домашнее задание:

- 1. Дополните скрипт таким образом, чтобы запуск программирования FLASH запускался автоматически
- 2. *Измените скрипт таким образом, чтобы команды reset_runs, launch_runs и wait_on_run были привязаны к активному design run, а не к конкретно impl_1

PS: большое спасибо пользователю <u>Aspect</u> чей комментарий мотивировал на написание этого материала.

Библиографический список

- 1. Уэлш Брент, Джонс Кен. Практическое программирование на Tcl и Tk.
- 2. UG835 Vivado Design Suite Tcl Command Reference Guide
- 3. <u>UG894</u> Using Tcl Scripting
- 4. UG898 Embedded Processor Hardware Design
- 5. <u>UG984</u>. MicroBlaze Processor Reference Guide. Xilinx Inc.
- 6. <u>MicroBlaze на сайте Xilinx</u>
- 7. <u>Vivado на сайте Xilinx</u>
- 8. <u>Описание</u> Arty Board на сайте Digilent

Список тренингов

по MicroBlaze в сертифицированном тренинг центре компании Xilinx:

- 1. Построение встраиваемых процессорных систем
- 2. Дополнительный курс по построению встраиваемых процессорных систем
- 3. Разработка ПО для встраиваемых процессорных систем
- 4. Доп. курс по разработке ПО для встраиваемых процессорных систем
- 5. Полный список курсов