# РАЗРАБОТКА ПРОЦЕССОРНОЙ СИСТЕМЫ НА БАЗЕ СОФТ-ПРОЦЕССОРА MICROBLAZE В СРЕДЕ XILINX VIVADO IDE/HLX

Автор: KeisN13

Рецензенты: Intekus Nagornayaod

Материал подготовлен при поддержке:

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

ATP Center Xilinx – Сертифицированный тренинг центр Xilinx (<u>www.plis2.ru</u>)

# Оглавление

Аннотация	3
Введение	3
HW/SW части процессорной системы	4
Создание нового проекта	5
Создание HW-части	12
Разработка SW части	
Заключение	61
Библиографический список	
Список тренингов	

#### Аннотация

В этой статье описаны основные этапы разработки процессорной системы на базе софт-процессора MicroBlaze [1, 2] в среде Xilinx Vivado [3]. Будут рассмотрены только основные моменты и шаги, которые позволят Вам быстро собрать процессорную систему из ресурсов ПЛИС/FPGA, а также получить общее представление о ходе её построения и подключения периферии.

#### Введение

Разработка систем на FPGA давно перестала ограничиваться простым написанием кода на языках описания аппаратуры (HDL); и по мере возрастания количества логических ресурсов и сложности проектов, подходы к проектированию систем на FPGA многократно пересматривались. Одним из витков развития стало внедрение в проекты софт-процессоров – по сути обычных микропроцессоров, но собранных на ресурсах FPGA. Такой подход даёт разработчикам гибкость системы и ещё больше объединяет области сотрудничества HW (hardware – аппаратных) и SW (software – программных) инженеров. К сожалению, несмотря на относительную несложность разработки софт-процессорных систем, многие пытаясь «поднять» эту тему, сталкиваются с трудностями освоения, потому что не знают где найти нужную информацию, а если и находят, то не знают с чего начать.

**Цель** статьи – дать общее представление об этапах сборки процессорной системы на базе софт-процессора Microblaze, используя среду Xilinx Vivado.

К сожалению, в рамках одной статьи сложно описать все многообразие процесса построения софт-процессорных систем на FPGA и детально изложить все сопровождающие его «тонкие» моменты, но с чего-то начать необходимо. В последующих статьях мы рассмотрим подключение Ethernet, различных контроллеров памяти, внешних интерфейсов, разберём работу компонентов,

3

связывающих MicroBlaze с периферией и многое другое. Но это будет чуточку позже, а для начала...

Примечание: надо сказать, что при создании софт-процессорной системы, точнее для её отладки Вам всё же придётся иметь какую-нибудь макетную плату с FPGA Xilinx 7-го семейства [4] или же уметь пользоваться QEMU эмулятором. Простое моделирование через симулятор XSIM здесь не есть решение проблемы. Моделирование запустится, но сколько понадобится времени чтобы посмотреть, что светодиод моргает – я сказать не могу; возможно – дни и недели машинного времени. Поэтому крайне рекомендую обзавестись недорогой макетной платой. экспериментировал на Arty Board от компании Digilent [5], именно на ней и будет проходить текущее и дальнейшее «обучение». Рекомендую читателям обзавестись ею, поскольку кристалл, установленный на ней, поддерживается бесплатной (Web) версией Vivado, но при этом с платой поставляется полноценная лицензия. В России плата обошлась мне в 153 доллара, заказывал через компанию Регион-Вирта [6], поскольку они могут работать с физическими лицами. В интернет-магазинах (включая AliExpress) в то время плата стоила гораздо дороже. Возможно, сейчас ситуация изменилась – проверьте.

#### HW/SW части процессорной системы

Разработка любой процессорной системы, построенной на ресурсах FPGA, состоит из двух фундаментальных частей: сборки аппаратной платформы HW – hardware, и разработки исполняемой программы SW – software.

НW-часть разрабатывается в среде Xilinx Vivado в модуле IP Integrator [7] (Vivado IPI) и представляет собой создание собственно экземпляра (или нескольких – для многопроцессорной системы) ядра MicroBlaze, соединение его с необходимой периферией и распределение адресного пространства. Разработка кода для MicroBlaze выполняется в Xilinx SDK [8] на ассемблере или C/C++.

4

Процесс сборки HW-части в Vivado IPI во многом похож, на аналогичный в предшествующих средах Xilinx – ISE и PlanAhead (где для этого использовалась утилита XPS – Xilinx Platform Studio), но имеет ряд отличий от него. Сохранились общие принципы, различия же встречаются, по большей части, в представлении системы. В одной из последующих статей мы разберём процесс разработки в XPS. Со стороны же программной (SW-части) никаких отличий нет: для работы с программной частью также используется Xilinx SDK.

Разработка аппаратной платформы начинается с запуска Vivado и создания проекта.

#### Создание нового проекта

Воспользуемся одним из способов открытия стартового окна Vivado, используя TCL Shell (Пуск – Xilinx Design Tools – Vivado 201x.x – Vivado 201x.x Tcl Shell (см. рис. 1)).



Рисунок 1 Расположение Tcl Shell в меню Пуск

В этом проекте я использую Vivado 2015.4, потому что в ней нет некоторых ошибок, присутствующих в последующих версиях. Для себя считаю её оптимальным вариантом по занимаемому на жёстком диске объёму, стабильности, функциональности и быстродействию. В более новых версиях этапы разработки HW-части либо полностью, либо почти полностью аналогичны или точь в точь повторяют описанную здесь последовательность действий.

В открывшемся окне пишем команду start\_gui (рис. 2). Нажимаем Enter и ждём окончания выполнения команды.



Рисунок 2 Tcl Shell и запуск Vivado с помощью команды start gui

После завершения выполнения команды появится стандартное окно Vivado (рис. 3). В зависимости от версии интерфейс может несколько отличаться; описание и скриншоты в статье будут соответствовать версии 2015.4.

Приступаем к созданию нового проекта.

Нажимаем Create New project (рис. 3)

🛓 Vivado 2015.4 — 🗖	× 🕂
Be Flow Iods Window Help	0
	O Bxo.
VIVADO, Productivity. Multiplied.	<b>(</b> 7) _
ALE PROGRAMMABLE.	🕓 Vib
Quick Start	Sky_
	FGP.
	Keis
Ureare new motions	Mic-
Tasks	🥠 Без.
	🛃 Xili
	📼 ug8.
Minage IP Open Hardware Manager Xiliux Td Store	CE Viv_
	🝌 Vīv
Information Center	
Documentation and Tutorials Quick Talke Videos Release Notes Guide	
	® 9
1 d Console	× 🖻 🧕
Z Vivadoł start_gui	^ 🔽 🖯
	S (3
\$	
	(1)
	ENG
Type a Tcl command here	8:46 AN
	417201

Рисунок 3 Окно Vivado 2015.4, красным выделена кнопка создания нового проекта

После нажатия кнопки появится мастер создания нового проекта, в котором нажимаем кнопку Next (рис. 4).

≽ New Project	×
	Create a New Vivado Project
	This wizard will guide you through the creation of a new project.
	To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.
lir	
IS	
-	
-	To continue, dick Next.
	< Back Next > Einish Cancel

Рисунок 4 Окно мастера создания нового проекта

На странице Project Name (рис. 5) указываем:

1. Название проекта – в поле Project name пишем Microblaze\_Lesson\_1

2. Место расположения проекта – в поле Project location указываем ту папку, которую Вам будет удобно использовать. *Не используйте* в названии пути русские буквы, пробелы, тире и специальные символы:

C:/Projects/FGPA\_Systems/Microblaze\_Lesson\_1

3. Снимаем галочку Create project subdirectory – иначе по указанному нами пути среда создаст подпапку с введённым нами именем проекта и поместит его файлы уже туда.

4. Нажимаем кнопку Next

A New Project	×
Project Name Enter a name for your project and specify a directory where the project data files will be stored.	
Project name: Microblaze_Lesson_1 2 Project location: C:/Projects/FGPA_Systems/Microblaze_Lesson_1 3	3
Create project subdirectory 3 Project will be created at: C:/Projects/FGPA-Systems/Microblaze_Lesson_1	
4	
< <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cancel	

Рисунок 5 Окно мастера создания нового проекта: название проекта, путь к проекту

На странице Ргојест Туре (рис. 6):

1. Выбираем тип проекта RTL Project, потому что мы будем создавать с нуля обыкновенный проект с добавлением IP-ядер и последующей имплементацией.

2. Ставим галочку Do not specify..., потому что при создании этого проекта никакие файлы мы в него добавлять не будем

3. Нажимаем кнопку Next

A New Project	×
Project Type Specify the type of project to create.	2
RTL Project You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.           Image: Comparison of the planning and analysis	1
<ul> <li>Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.</li> <li>Do not specify sources at this time</li> </ul>	
O I/O Planning Project Do not specify design sources. You will be able to view part/package resources.	
O Imported Project Create a Vivado project from a Synplify, XST or ISE Project File.	
Create a new Vivado project from a predefined template.	
< Back Next > Einish Ca	incel

Рисунок 6 Окно мастера создания нового проекта: выбор типа создаваемого проекта

На странице Default Part выбираем (рис. 7):

1. Вкладку Parts с перечнем FPGA, доступных для текущей версии Vivado.

2. В поле поиска Search пишем название кристалла, стоящего на плате. На Arty

Board установлена FPGA xc7a35ticsg324-1L

3. В списке выбираем нашу FPGA

- 4. Нажимаем кнопку Next
- 9

🝌 New Project										×
Default Part Choose a default Xili	nx part or board	l for your projec	t. This can	be changed la	ter.					
Select: Parts	Boards									
Product category:	All		Ŧ	Spee <u>d</u> gr	ade:	All			<b>*</b>	
Eamily:	All			Temp gra	de:	All			•	
Package:	All		*	]						
			2	Reset All Filter	s					
Search: Q→ xc7a35tic	sg324-1L	⊗ ~ (1	match)							
Part	I/O F Cour	Pin Block	s I	DSPs	FlipFlo	ps	GTPE2 Transceivers	Gb Transceivers	Available IOBs	LU Ele
🔷 xc7a35ticsg324-1L	324	50	9	90	41600	(	D	0	210	208
	5									
<								4		
					<	< <u>B</u> ack	<u>N</u> ext >	Einish	Cance	I

Рисунок 7 Окно мастера создания нового проекта: выбор FPGA

На этом создание нового проекта закончено. В окне New Project Summary (рис. 8) отображается суммарная информация о созданном проекте. Нажимаем кнопку Finish.



Рисунок 8 Окно мастера создания нового проекта: суммарные сведения о создаваемом проекте

Если Вы что-то захотите поменять в проекте, это можно сделать в любой момоен во вкладке Tools – Project settings (рис. 9)



Рисунок 9 Кнопка для изменения настроек проекта

## Создание НW-части

Для разработки HW-части софт-процессорной системы в Vivado используется IP Integrator [7], а сам процесс построения основан на добавлении и соединении готовых IP-ядер, которые могут быть как от Xilinx, так и от других фирм, а также разработанными Вами самостоятельно. Для запуска IP Integrator нужно нажать кнопку создания нового блочного проекта Create Block Design, которая находится в панели Flow Navigator – группа IP Integrator, пункт Create Block Design (рис. 10).



Рисунок 10 Кнопка создания нового блочного проекта

После нажатия на кнопку создания нового блочного проекта появится окно (рис. 11), которое позволяет задать его имя, определить папку для его размещения, если предлагаемая по умолчанию Вас не устраивает, и определить к какой подгруппе основной панели (дизайн или моделирование) будет относиться этот блок. Изменим:

- 1. Имя проекта на system
- 2. Нажимаем кнопку ОК

🝌 Create Block D	esign			×
Please specify name	of block design.			
		1		<b>••</b>
Design name:	system	Ľ.		$\odot$
Directory:	🛜 <local pr<="" td="" to=""><td>oject&gt;</td><td></td><td>-</td></local>	oject>		-
Specify source set:	C Design Sour	ces		-
				2
			ОК	Cancel

Рисунок 11 Окно настроек создаваемого блочного проекта

После нажатия на кнопку OK Vivado перейдёт в режим IP Integrator: (рис. 12).

Microblaze_Lesson_1 - [C:/Projects	/FGPA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1xpr] - Vivado 2015.4	- 0 X
File Edit Flow Tools Window	ayout Yew Hep	Q- Search commands
😂 🖿 🖬 🖬 🖓 🖏	🖻 🕨 🏂 🔞 🛞 🔽 🌀 🔠 Default Layout 💦 🔹 🦎 🗽 🥸	Ready
Flow Navigator	Block Design system	×
९ 🛣 🛱	Design L × Co Diagram X	□ Ŀ×
<ul> <li>Project Hanager         <ul> <li>Project Settings</li> <li>Settings Block Design</li> <li>Simulation</li> <li>Simulation</li> <li>Simulation</li> <li>Rin Simulation</li> <li>Rin Simulation</li> <li>Rind Export Settings</li> <li>Bornet Block Design</li> <li>Simulation</li> </ul> </li> <li>Rind Export Settings</li> <li>Bornet Block Design</li> <li>Simulation</li> <li>Simulation</li> <li>Simulation</li> </ul> <li>Rind Export Settings</li> <li>Bornet Block Design</li> <li>Simulation</li> <li>Simulation Settings</li> <li>Bornet Block Design</li> <li>Simulation Settings</li> <li>Bornet Block Design</li> <li>Simulation Settings</li> <li>Simulation Settings</li> <li>Simulation Settings</li> <li>Simulation Settings</li>	Image: System	
Run Synthesis Doen Synthesized Design	Tid Console	- D 12 X
Implementation     G Implementation     Implementation     Implementation     Implementation     Implemented Design     Program and Debug     G Istream Settings     G Intervent Bitshream     Implemented Bitshream     Implemented Design     Implemented Design	<pre>Vivadot start_gui Vivadot start_gui Vivadot</pre>	
	Type a Tcl command here	
	📓 Tcl Console 💭 Messages 🛛 🕄 Log 🗋 Reports 🐊 Design Runs	

Рисунок 12 Окно Block Design с различными доступными вкладками

Теперь мы можем начать собирать процессорную систему. Процесс этот во многом автоматизирован, и мы этим воспользуемся. Первое, что нам необходимо добавить в софт-процессорную систему – это сам софт-процессор MicroBlaze. Есть

несколько вариантов его исполнения и добавления, но для проектирования таких простых систем, как наша, обычно используется мастер настройки. Для добавления IP-блоков на схему проекта (вкладка Diagram) можно воспользоваться кнопкой Add IP <sup>Ф</sup>, расположенной в панели инструментов слева во вкладке Diagram или нажать Ctrl+I. После нажатия на кнопку откроется каталог блоков, которые можно добавить на поле Diagram (рис. 13).



Рисунок 13 Окно со списком доступных IP блоков, после нажатия на кнопку Add IP

В появившемся окне со списком доступных IP блоков (рис. 14) в поле поиска Search пишем *MicroBlaze* и выбираем из найденных позиций MicroBlaze. Остальные два это MicroBlaze Debug Module (MDM) – отладчик для MicroBlaze, который будет позже добавлен автоматически, и микроконтроллерная система на базе MicroBlaze – фактически, ещё один вариант кастомизации процессорной системы.



LIVIER to select, ESC to calleet, curry for in details

Рисунок 14 Окно со списком доступных IP блоков, с отфильтрованным списком MicroBlaze

Дважды кликаем по MicroBlaze, после чего IP блок появляется на поле (рис. 15). Обратите внимание, что также появилось окно настройки адресов (об этом позже), и «свесилась» зелёная строка помощи, которая появляется, если Vivado может как-то автоматизировать процесс (в нашем случае появилась надпись-ссылка Run Block Automation) и сам модуль MicroBlaze возник на поле Diagram.

4	Micr	oblaze_Lesson_1 - [C:/Projects/FGPA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.x
Ei	e <u>E</u>	dit Flow <u>T</u> ools <u>W</u> indow La <u>v</u> out <u>V</u> iew <u>H</u> elp
	Blo	ск Design - system *
	s	Be Diagram × R Address Editor ×
ō	pertic	→] ♣ system
/igat(	Prof	Q+ 🗔 Designer Assistance available Run Block Automation
low Nav		Можно что-то автоматизировать
Ľ.	~	
	g	
	Des	
		Reset
		MicroBlaze
		🔚 выбранный модуль из каталога

Рисунок 15 Появление дополнительных вкладок, опций автоматизации и IP блока после его добавления

Воспользуемся предлагаемой автоматизацией, нажав на ссылку Run Block Automation, после чего появится окно с мастера экспресс настроек процессорной системы (рис. 16). После вызова мастера доступны именно экспресс-настройки, то есть те, которые используются наиболее часто. Расширенные настройки доступны по двойному щелчку на IP-блоке MicroBlaze, но это выходит за рамки текущей статьи.

Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.	A Run Block Automation
Image: Second state of the selected of the second state of the second stat	Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.
OK Cancel	All Automation (1 out of 1 selected)

Рисунок 16 Мастер экспресс натрое процессорной системы

Из предлагаемых настроек доступны:

- 1. Количество памяти для программы и данных
- 2. Управление механизмом коррекции ошибок
- 3. Конфигурация кэша
- 4. Конфигурация отладчика (тот самый MDM из списка)
- 5. Задействование/отключение порта сопряжения с периферией (АХІ-порта)
- 6. Включение/выключение контроллера прерываний

#### 7. Выбор сигнала тактирования

Оставим все настройки в значениях по умолчанию и нажмём кнопку ОК, т. к. для нас этих параметров достаточно. После окончания работы мастера на поле появится множество новых блоков, которые мы разберём ниже. Для оптимизации рабочего поля нажмите кнопку перестроить 🐼 в панели инструментов (рис. 17).



Рисунок 17 Результат работы мастера экспресс-настроек после оптимизации рабочего поля

Как видите, картина изменилась, и на рабочем поле появились:

1. Кнопка оптимизации рабочего поля

2. Предложение по автоматическому подключению одних блоков к другим (нажмите на неё и посмотрите, что предлагается подключить; после просмотра нажмите Отмена – Cancel)

3. Блок управления тактовой частотой и синхронизацией

- 4. Блок сброса процессорной системы
- 5. Модуль отладки (отладчик)
- 6. Ядро софт-процессора MicroBlaze
- 7. Локальная память данных и программ

Теперь добавим необходимую периферию. В начале статьи мы определились, что будем мигать светодиодом, и выдавать сообщения по UART. Значит, нужно добавить IP-блоки, которые будут обеспечивать этот функционал. Для мигания будем использовать IP-блок ввода/вывода общего назначения GPIO [9], а для вывода сообщений – UART Lite [10].

Добавление IP блоков аналогично добавлению MicroBlaze. Поэтому нажимаем кнопку <sup>9</sup>, и в поле поиска пишем *gpio*, выбираем модуль AXI GPIO и нажимаем Enter (рис. 18)



Рисунок 18 Поиск GPIO в каталоге доступных IP блоков

На схему должен добавиться блок, изображённый на рис. 19.



Рисунок 19 Блок АХІ GPIO

После добавления блока AXI GPIO необходимо его настроить, задать разрядность портов, определить направление и количество каналов. Настройка производится двойным щелчком по блоку либо выбором Customize Block из контекстного меню, открывающегося по щелчку правой кнопкой мыши. Модуль axi\_gpio\_0 будет управлять светодиодом, периодически включая и выключая его. Значит axi\_gpio\_0 должен быть сконфигурирован следующим образом (рис. 20)

Re-customize IP		×
AXI GPIO (2.0)		
W Documentation 📄 IP Location		
Show disabled ports	Component Name system_axi_gpio_0_0	
	GPIO	
	All Inputs	
	All Outputs	
	GPIO Width 1 [1 - 32]	
	Default Output Value 0x00000000 (0x0000000,0xFFFFFFF)	
	Default Tri State Value 0xFFFFFFF () [0x0000000,0xFFFFFFF]	
LAS AVI	Enable Dual Channel 3	
s_axi_ack GPIO ⊕	GPIO 2	
-s_axi_aresetn	All Inputs	
	All Outputs	
	GPIO Width [1 - 32]	
	Default Output Value 0x00000000 (0x0000000,0xFFFFFFF)	
	Enable Interrupt 4	
		-
	ОК	Cancel

Рисунок 20 Настройка блока АХІ GPIO

1. Задаём направление портов блока AXI GPIO – выбираем, что все выводы являются выходами, т. к. мы управляем светодиодом, а не он нами.

2. Ставим разрядность порта 1 – всего будет подключён 1 светодиод.

3. Второй канал нам не нужен, отключаем его.

4. Нажимаем кнопку ОК.

Нажав плюсик около интерфейса GPIO модуля axi\_gpio\_0 увидим, что порт gpio\_io\_o имеет разрядность 1 (вектор [0:0]) – рис. 21.



Рисунок 21 Вид блока АХІ GPIO после настройки

Теперь давайте попробуем подключить наш настроенный блок AXI GPIO к MicroBlaze. Предлагаю довериться автоматике и посмотреть, что нам создаст Vivado. Нажимаем на строку Run connection Automation (см рис. 19) и ждём появления окна помощника доступных подключений (рис. 22)

À Run Connection Automation		×
Automatically make connections in your design by check the right.	king the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on	2
All Automation (1 out of 5 selected)	Description Connect Slave interface (/axi_gpio_0/S_AXI) to a selected Master address space. Options Master: /microblaze_0 (Periph) Clock Connection (for unconnected clks) : Auto	
	3	
	ОК Са	ncel

Рисунок 22 Окно помощника подключений

В окне помощника подключений:

1. Выберите подключение axi\_gpio\_0 и поставьте галочку S\_AXI (далее мы рассмотрим и поясним это подключение).

2. Установите подключение тактового сигнала как Auto.

3. Нажмите ОК.

После завершения подключения нажмите кнопку оптимизации рабочего пространства 🐼 в панели инструментов.

Как вы можете заметить, появился дополнительный модуль AXI Interconnect [11] (рис. 23). Кратко опишу его назначение. Дело в том, что взаимодействие между процессором и периферией происходит по шине AXI [12] (о ней и её видах должна быть отдельная статья, если кто-то захочет в этом помочь – напишите мне). На шине есть мастер (обычно это процессор) и слэйв (Slave), в нашей литературе это ведущий и ведомый соответственно. Мастер отправляет команды слейву. Однако AXI не позволяет подключить к мастеру более одного слейва напрямую. Именно напрямую нельзя, но можно через коммутатор, который и называется AXI Interconnect – назначение этого модуля обеспечить подключение между несколькими мастерами и несколькими слейвами. Пока в нашей системе один мастер – microblaze\_0 и один слейв – axi\_gpio\_0.



Рисунок 23 Результат работы помощника подключений: автоматическая вставка модуля AXI Interconnect, его подключение к MicroBlaze и AXI GPIO

После работы мастера синяя строка сверху не пропала. Это потому что ещё есть что автоматизировать, но делать этого мы пока не будет. Теперь давайте попробуем добавить модуль UART Lite в нашу систему.

Надеюсь, что последовательность запомнили. Нажимаем кнопку <sup>у</sup>, вводим в строке поиска Uartlite, дважды кликаем и ждём. Если все сделали корректно, должен появиться модуль как на рис. 24.





При желании просмотрите доступные настройки для модуля и если есть необходимость, измените. Настройки обычные: скорость (9600), количество стоп бит (1), чётность (нет), количество бит (8).

Для подключения Uartlite к нашей системе, давайте вновь воспользуемся автоматизацией, которую предлагает Vivado, нажав строку Run Connection Automation. Для модуля axi\_uart\_0 выберите S\_AXI и нажмите OK (рис. 25)

📥 Run Connection Automation			×
Automatically make connections in your design by ch the right.	ecking the boxes of the interfaces to connect. Selec	t an interface on the left to display its con	figuration options on
All Automation (1 out of 6 selected)	Description Connect Slave interface (/axi_uartlite_0/S	6_AXI) to a selected Master address space	
	Options		12
	Master:	/microblaze_0 (Periph)	
	Clock Connection (for unconnected clks)	· Auto ·	
⇒ reset			•
·····[_] ≫ ext_reset_in			
			3
		[	OK Cancel

Рисунок 25 Настройка помощника подключений для Uartlite

После завершения подключения нажмите кнопку оптимизации рабочего пространства 🐼 в панели инструментов и обратите внимание на изменения в AXI Interconnect (если не помните что было, обратитесь к рис. 23).

Что произошло и почему? В AXI Interconnect добавился дополнительный мастер-порт для подключения второго слэйва (Uartlite – это ведомое устройство). MicroBlaze имеет один мастер-порт, который называется M\_AXI\_DP. Мастер один, а слейва два, но подключать напрямую можно только один слейв. Для подключения нескольких слейвов мы используем AXI Interconnect. Поэтому в AXI Interconnect добавился ещё один порт (рис. 26). В общем случае, AXI Interconnect может обеспечивать взаимное подключение нескольких мастеров и нескольких слейвов.



Рисунок 26 После окончания работы помощника подключений, количество слейв устройств на шине стало равным двум. Поэтому в АХІ Interconnect добавился ещё один порт

Часть дела сделана. Теперь давайте настроим модуль управления тактовой частотой и синхронизацией clk\_wiz\_1. Для Ваших плат значения могут быть другими – надеюсь, что Вы сможете найти в документации значение частоты системного тактового генератора, тип идущего от него сигнала, и ножку(и) FPGA к которым он подключён. Ниже приведены настройки для Arty Board (рис. 27). Настройка любого модуля производится двойным щелчком по нему.

🚽 Re-customize IP	X
Clocking Wizard (5.2)	4
🖗 Documentation 📄 IP Location	
IP Symbol Resource	Component Name system_dk_wiz_1_0  Clocking Options Output Clocks MMCM Settings Summary  Primitive  Primitive  Clocking Features Jitter Optimization  Generation  Generation  Jitter Optimization  Generation  Gene
	Dynamic Reconfig Dynamic Phase Shift Maximize Input Jitter filtering     Safe Clock Startup  Dynamic Reconfig Interface Options     AXI4Ute DRP Phase Duty Cycle Config  Input Clock Information
	Input Clock     Toput Ecception       Primary     Manual       100.000     100.000       Secondary     Auto       100.000     60.000       120.000     0.010       Single ended dook capable     •
	OK Cancel

Рисунок 27 Окно настройки модуля управления тактовой частотой и синхронизацией: вкладка Clocking Options

В окне настроек clk\_wiz\_1:

- 1. Выберите вкладку Clocking Options.
- 2. Тип модуля ММСМ.
- 3. Включены возможности синтеза частоты и фазовой подстройки.
- 4. Основная тактовая частота 100МГц.
- 5. Тип сигнала с системного тактового генератора на плате однополярный.
- 6. Затем выберите вкладку Output Clocks (рис. 28).
- 7. На ней установите выходное значение частоты (тактовая частота процессора) также 100 МГц.

Пролистайте вниз до конца

- 8. Снимите галочку с сигнала reset.
- 9. Установите галочку locked.
- 10. Нажмите ОК.

🧧 Re-customize IP								×
Clocking Wizard (5.2)								
🍘 Documentation 늖 IP Location								
IP Symbol Resource	Component Name system	_dk_wiz_1_0 6						
Show disabled ports	Clocking Options	Output Clocks M	MCM Settings S	ummary				
	The phase is calculated	relative to the active	input dock.					^
	Output Clock 0	utput Freq (MHz)	Actual	Phase (degrees)	Astual	Duty Cycle (%	6) Actual	Drives
	R dk out1 10		100.000	0.000	0.000	50,000	50.0	BLIEG
	dk out2 10	0.000	N/A	0.000	N/A	50.000	N/A	BUFG
	dk out3 10	0.000	N/A	0.000	N/A	50.000	N/A	BUFG
	dk out4 10	0.000	N/A	0.000	N/A	50.000	N/A	BUFG
	dk_out5	0.000	N/A	0.000	N/A	50.000	N/A	BUFG
-clk_in1 clk_out1 -	dk_out6	0.000	N/A	0.000	N/A	50.000	N/A	BUFG
-reset locked -	clk_out7 10	00.000	N/A	0.000	N/A	50.000	N/A	BUFG
		ENCINC	c	locking Feedback				
				Source		Signaling		
	Output Clock	Sequence Num	ber	Automatic (	ontrol On-Chin	Single	e-ended	
	dk_out1	1			ontrol Off-Chin		rential	
	dk_out2	1				O Dirie	enua	
	dk_out3	1		O User-Contro	lled On-Chip			
	dk_out4	1		O User-Contro	lled Off-Chip			
	<	-						× *
							OK	Cancel

# Пролистайте вниз

Re-customize IP									×
Clocking Wizard (5.2)									2
🍘 Documentation 📄 IP Location									
IP Symbol Resource	Component Name syste	em_dk_wiz_1_0 6							
Show disabled ports	5.2) contain  Tree  Corponent Name system_dk_uiz_1.9  Cociang Optors Output Clock MMCM Settings Summary  Cociang Optors Output Settings Summary  Cociang Optors Output Settings  Cociang Optors Output Settings  Cociang Optors Output Settings  Cociang Optors Output Settings  Cociang Optors  Cociang  Cociang								
	dk_out6	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	^
	dk_out7	100.000	N/A	0.000	N/A	50.000	N/A	BUFG	-
			1	Clocking Feedback		1			
	USE CLOCK SEC	SOElaChag		Source		Signaling			
	Output Clock	Output Clock Sequence Num clk_out1 1		Automatic Control On-Chip		Single-ended			
	ck_out1								
	dk_out2	1		Automatic Control Off-Chip     Differen			ferential		
	dk_out3	1		O User-C	Controlled On-Chip				
clk_out1 -	ck_out4	1		O User-C	Controlled Off-Chip				
	dk_out5	1		Пролиста					
lockeu —	ck_out6	1		пролиста	ите вни.	з до кон	ца		
	ck_out7	1		]					
	Enable Optional Inpu	ts / Outputs		Reset Type					
	reset 8	Doower down	input ck	stopped					
			]	() A	ctive High				
	✓ locked	clkfbstopped		( ) A	ctive Low				
	9								
								~	~
							1		-
							0	K Car	ncel

Рисунок 28 Окно настройки модуля управления тактовой частотой и синхронизацией: вкладка Clocking Options

Установленные настройки могут повлиять на внешний вид clk\_wiz\_1 (для Arty Board он примет вид, как на рис. 29).



Рисунок 29 Изменение внешнего вида модуля после проделанных настроек (для Arty Board)

Если оставить незадействованные выводы модуля сброса неподключенными, то может возникнуть ситуация, при которой процессор будет всегда в состоянии сброса. Потому выполним явное их подключение к логической «1» – неактивному для них логическому уровню. Для этого:

- 1. Добавьте блок constant.
- 2. Установите в нем значение 1 (если по умолчанию оно другое).
- 3. Подключите выход блока Constant к входам aux\_reset\_in и ext\_reset\_in блока rst clk wiz 1 100M.

Подключение осуществляется следующим образом: подведите мышку к выходу блока Constant (указатель должен принять вид карандашика), нажмите на выход и, не отпуская левую кнопку мыши, подведите соединение к входу aux\_reset\_in – а затем повторите те же действия для ext\_reset\_in. В очередной раз нажмите кнопку оптимизации рабочего пространства 🐼 в панели инструментов. Соединение должно принять вид, как на рис. 30.



Рисунок 30 Соединение выхода блока Constant с входами aux\_reset\_in и ext\_reset\_in блока rst\_clk\_wiz\_1\_100M

Теперь необходимо создать внешние входы и выходы для нашей процессорной системы. Если вы обратили внимание, сейчас не подключены clk\_in1 и выходы экземпляров модулей axi\_gpio\_0 и axi\_uartlite\_0. Сейчас мы должны обозначить, какие порты являются внешними и должны выходить из нашей процессорной системы наружу.

Назначим вход clk\_in1 модуля Clocking Wizard внешним. Для этого кликнем по нему правой кнопкой мыши и выберем Make External (рис.31)

## www.FPGA-Systems.ucoz.net



Рисунок 31 Объявление входа clk\_in1 модуля Clocking Wizard внешним

После этого к входу clk\_in1 будет подключён порт, с таким же именем (рис. 32)



Рисунок 32 Внешний порт clk\_in1, созданный автоматически, подключён к входу clk\_in1 модуля clk\_wiz\_1

Аналогично поступаем и с выходом GPIO модуля axi\_gpio\_0 и шиной UART модуля axi\_uartlite\_0. Сделать внешним можно как шину, так и отдельный порт из шины; а если портов в шине несколько (как в шине UART), то направление портов

будет определено автоматически (см. 33). Некоторые незадействованные порты можно оставить неподключёнными – например, interrupt модуля axi\_uartlite\_0.



Рисунок 33 Объявление внешними порт управления светодиодом и шины UART

Помните про вкладку Address Editor (рис.34)? Самое время обратиться к ней и понять её назначение.

Наши модули соединены по шине АХІ, полное название АХІ Метоry Мар, это значит, что все модули, подключённые к шине должны иметь уникальный адрес, чтобы процессор смог к ним обратиться и «не перепутать» один модуль с другим. Назначение уникальных адресов (распределение адресного пространства) как раз и происходит во вкладке Address Editor, где в ручном, полуавтоматическом или автоматическом режиме можно установить диапазон адресов для конкретных модулей.

P	Diagram 🗙 🔣 Address Editor 🗙					
۹,	Cell	Slave Interface	Base Name	Offset Address	Range	High Address
$\mathbf{X}$	⊡ · 📑 microblaze_0					
	🖃 🔛 Data (32 address bits : 4G)					
	microblaze_0_local_memory/dlmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	8K 👻	0x0000_1FFF
矖	axi_gpio_0	S_AXI	Reg	0x4000_0000	64K 🔹	0x4000_FFFF
	🚥 axi_uartlite_0	S_AXI	Reg	0x4060_0000	64K 🔹	0x4060_FFFF
	🖃 🔛 Instruction (32 address bits : 4G)					
	📟 microblaze_0_local_memory/ilmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	8K 👻	0x0000_1FFF

#### Рисунок 34 Вкладка Address Editor

Поскольку мы пользовались помощником соединений и мастерами настроек, адресное пространство уже полностью сконфигурировано, но если бы мы делали подключение шин в ручном режиме, то увидели бы следующую картину (рис. 35).

Эта картинка только для демонстрации: я временно удалил шину AXI, соединяющую экземпляры AXI Interconnect и GPIO, а потом соединил её вручную. Из-за ручного соединения назначения адреса не произошло. Но мы можем назначить адрес, нажав на кнопку автоматического назначения адресов на панели инструментов вкладки Address Editor. После этого, картинка станет прежней (то есть как на рис. 34).

20	Diagram 🗙 🔣 Address Editor 🗙					
٩	Cell	Slave Interface	Base Name	Offset Address	Range	High Address
$\mathbf{Z}$	⊡ 👎 microblaze_0					
	😑 🖽 Data (32 address bits : 4G)					
1. <b>4</b> 1	microblaze_0_local_memory/dlmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	8K	• 0x0000_1FFF
8	🚥 🚥 axi_uartlite_0	S_AXI	Reg	0x4060_0000	64K	• 0x4060_FFFF
-	🖻 🗁 Unmapped Slaves (1)					
	axi_gpio_0	S_AXI	Reg			
	🖮 🎛 Instruction (32 address bits : 4G)					
	🛄 🚥 microblaze_0_local_memory/ilmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	8K	• 0x0000_1FFF
	2 Назначить 1	оявила	сь нер	азмечен	ная	а область
	адрес	oche py	чного	соедиен	ия	АЛІ

Рисунок 35 В случае ручного соединения шин АХІ появляется неразмеченная область. Нажатие кнопки запускает автоматическое распределение адресного пространства

К сожалению, для нашей будущей программы не достаточно памяти, которую мы задали, когда пользовались мастером экспресс-настроек MicroBlaze (рис. 16). Нам необходимо увеличить количество памяти для данных и инструкций с 8К до 16К. Сделать это можно простым выбором из выпадающего списка доступного количества памяти, нажав на стрелочку в соответствующем поле (рис. 36). Изменить необходимо размер памяти и инструкций. В обоих полях должно быть значение 16К

P	Diagram 🗙 🔣 Address Editor 🗙					
۹,	Cell	Slave Interface	Base Name	Offset Address	Range	High Address
$\mathbf{Z}$	⊡ 📲 microblaze_0					
	🖻 🔠 Data (32 address bits : 4G)			_		-
_	microblaze_0_local_memory/dlmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	16K 💌	x0000_3FFF
斷	🚥 axi_uartlite_0	S_AXI	Reg	0x4060_0000	64K 🔻	0x4060_FFFF
	🔤 axi_gpio_0	S_AXI	Reg	0x4003_0000	64K 🔻	0x4000_FFFF
	🖻 🔠 Instruction (32 address bits : 4G)			_		
	🛄 🚥 microblaze_0_local_memory/ilmb_bram_if_cntlr	SLMB	Men	0x0000_0000	8K 🗸	0 t0000_1FFF
					8K 🔺	
		160			16K	
	VISIMENNIE ON HA	TOU			32K	
					64K	

Рисунок 36 Изменение количества памяти, выделяемой для процессорной системе

Мы проделали много действий, закончили собирать систему и назначили все адреса. Но все ли корректно и правильно? В панели инструментов на вкладке Diagram есть одна из самых важных кнопок, называется она Validate Design (2 на рис. 37). Нажатие этой кнопки запускает инструмент проверки ошибок сборки HW-части процессорной системы (рис. 37). Нажмите эту кнопку и дождитесь результата. Если появилось окно, как на рис. 37, то поздравляю Вас, сборка HW-части проекта закончена. Если же появились ошибки – внимательно прочитайте их и постарайтесь самостоятельно исправить, вернувшись по тексту к соответствующим пунктам либо проделав всю последовательность с начала ещё раз, более внимательно и аккуратно.



Рисунок 37 Проверка на ошибки сборки НW части процессорной системы.

Нажмите на основной панели Vivado кнопку сохранения, а затем – кнопку Project Manager, чтобы выйти из IP Integrator и вернуться в основной режим работы Vivado (рис. 38):



Рисунок 38 Сохранение и выход из IP Integrator

Далее следуют стандартные этапы проектирования: синтез, имплементация, генерация файла прошивки (битстрима). Но для блочного проекта обязательно нужно сделать обёртку (wrapper); это показано на рис. 39. Нажмите правой кнопкой на созданном блочном проекте и выберите Create HDL Wrapper. В появившемся окне нажмите OK.

#### www.FPGA-Systems.ucoz.net



Рисунок 39 Создание обёртки для блочного проекта.

Обёртка – это простой VHDL- или Verilog-файл (с расширениями «vhd» или «v» соответственно), в который включена наша процессорная система, как часть иерархии. Таким образом, нашей собранной процессорной системой мы сможем оперировать, как простым модулем, добавляя её в качестве подмодуля в модули верхнего уровня. После создания обёртки наш модуль можно наконец-то запустить на синтез, нажав на кнопу Run Synthesis (рис. 40)

## www.FPGA-Systems.ucoz.net



Рисунок 40 Запуск синтеза проекта

После окончания синтеза появится окно выбора действия (рис. 41):

Synthesis Completed	×
<ul> <li>Synthesis successfully completed.</li> <li>Next</li> </ul>	
<ul> <li><u>Run Implementation</u> 1</li> <li><u>Dpen Synthesized Design</u> 2</li> <li><u>View Reports</u> 3</li> </ul>	
Don't show this dialog again 4 OK Cancel	

Рисунок 41 Действие после синтеза

Окно, появляющееся после синтеза, предлагает выполнить:

- 1. Запустить имплементацию.
- 2. Открыть синтезируемый проект (Выберите этот пункт).
- 3. Просмотреть отчёты.

Выберите второй пункт – открыть синтезированный проект. Если Вы машинально закрыли окно на рис. 41, то можно открыть синтезированный проект другим способом – нажав на кнопку Open Synthesized Design, которая находится под кнопкой запуска синтеза (рис. 42). Обратите внимание, что кнопка Open Synthesized Design активна, если только у Вас есть результат синтеза; если проект не синтезирован, кнопка будет неактивна (см. рис. 40, где результатов синтеза ещё нет)



Рисунок 42 Кнопка открытия синтезированного проекта

Открытие синтезированного проекта нужно для двух вещей:

- 1. Проанализировать проект (если есть желание).
- 2. Выполнить назначение портам нашего проекта физических ножек FPGA.

До текущего момента физически мы ещё не указывали, к каким ножкам подключить нашу HW-часть. Сделаем это сейчас в модуле I/O Planning. Чтобы его запустить, необходимо при открытом уже синтезированном проекте выбрать в меню Layout пункт I/O Planning (рис. 43).



Рисунок 43 Открытие модуля планировщика выводов FPGA (обязательно должен быть открыт синтезированный проект, в противном случае планировщик может быть недоступен).

Если всё сделано корректно, то должен открыться планировщик выводов (рис. 44).

Synthesized Design - xc7a35ticsg324-1L (active)								
Device Constraints	2	× 🔢 Packa	age 🗙 🍥 Device 🗴	۲				
a 🔀 🖨 🖪 🕐 🗕		<b>⇒</b>			12345678	9 10 11 12	13 14 15 16 1	7 18
▼ Internal VREF								
0.675V							+	
		Qt 🚬						
		Q- 1						+
TO Bank 14								
		8				* * * *		
						SS++		1
Drop I/O banks on voltages or the "NONE" folder to set/ VREF.	unset Internal					SS++		
🚴 Sources 🕅 Netlist 🌰 Device Constraint	s	j∰ L				•		
Properties		× 🎽 M				* * * *		
$\leftarrow \rightarrow \bigotimes k$		🐺 N						
Select an object to see properties		Ó					*	4
I/O Ports								
Name	Direction	Neg Diff Pair	Site	Fixed Bank	I/O Std	Vcco Vre	f Drive Strength	Slew Type
□ □ □ All ports (4)	OUT				de fer de (Lucenco no)	1 000	12	0.000
					default (LVCMOS18)	1,800	12	SLOW
	OUT			↓ □	default (LVCMOS18)	<ul> <li>1.800</li> <li>1.800</li> </ul>	12	SLOW
Scalar ports (0)								
UART_43442 (2)	(Multiple)				default (LVCMOS18)	• 1.800	(Multiple)	(Multiple)
🖃 🕞 Scalar ports (2)								
□ UART_rxd	IN			•	default (LVCMOS18)	• 1.800		
V UART_txd	OUT			•	default (LVCMOS18)	• 1.800	12	SLOW
Scalar ports (1)	TN				default (LVCMOE 19)	1 900		
	214			· □	ucrault (LVCI+IOS 10)	1.000		

Рисунок 44 Вид окна планировщика выводов с раскрытым списком портов модуля верхнего уровня нашего проекта

Теперь мы можем выполнить назначение физических ножек FPGA портам нашей процессорной системы. Сделать это можно многими способами. Один из них - явно заполнить необходимые поля. Нас будут интересовать только поля Site название контакта FPGA и I/O Std - стандарт ввода-вывода, определяющей напряжения логических уровней и другие электрические параметры. Обратите внимание, что если Ваша плата – не Arty Board, то содержимое этих двух колонок может быть другим; надеюсь, Вы сможете прочитать принципиальную электрическую схему Вашей платы и указать во всех полях правильные значения. Для Arty Board они показаны на рис. 45.

Synthesized Design * - xc7a35ticsg324-1L	(active)					
Device Constraints	- 0 2	× 🔠 Pac	kage 🗙 🛞 Device	e X		
🔍 🛣 🖨 🛃 🕐 🗕		⇒.		-	12345	5678
▼ Internal VREF		4				Y III
- C 0.75V		A B				
🔁 0.9V						
⊡		Q₹ _				
····· I/O Bank 14						
I/O Bank 15		E State				
I/O Bank 34		V N G				• • • + +
I/O Bank 35		🗆 : 🚬 H				
Drop I/O banks on voltages or the "NONE" fold	ler to set/unset Internal	j				+++
VREF.		к				) 🛛 🛨 🔶
🔏 Sources 🕅 Netlist 🏻 🙈 Device Con	onstraints	<u></u> 派 L				
I/O Port Properties	2	× 🎤 M				
$\leftarrow \rightarrow \bigotimes_{k}$		🥮 🖔				
⊮ dk_in1		P <sub>n</sub>				
		A R				
DIFF TERM						
		<b>_</b> ⊻   °.				
General Properties Configure Power						
Properties Clock Regions					1	
I/O Ports				-		-
🔍 Name	Direction	Neg Diff Pair	Site	Fixed Bank	I/O Std	
All ports (4)						
GPIO_53339 (1)	OUT				3. LVCMOS33*	
	OUT		HS		31 LVCMOS33*	Kc
Scalar ports (0)					5. 21010555	
UART_43442 (2)	(Multiple)				16 LVCMOS33*	
Scalar ports (2)			10	_		
UART_rxd	IN		A9		10 LVCMOS33*	I K N
Scalar ports (1)	001		510		TE LVCMOS33*	
₩ octain ports (1)	IN		E3	· 🖂	3 LVCMOS33*	Кта

Рисунок 45 Подключение портов к ножкам FPGA для Arty Board

Обратите внимание, что значки портов после выполнения назначений стали жёлтыми, а не серыми.

Сохраним результаты назначения, нажав кнопку сохранить (рис. 46). После нажатия на неё появится окно, которое говорит, что сохранение возможно приведёт к необходимости повторного синтеза. Нажимаем ОК.



A Microblaze\_Lesson\_1 - [C:/Projects/FGPA-Systems/Microblaze\_Lesson\_1/Microblaze\_Lesson\_1.xpr] - Vivado 2015.4

Рисунок 46 Сохранение в планировщике выводов произведённых изменений.

Сохранение параметров назначений ножек производится обычно в файл формата (и расширения) xdc – Xilinx Design Constraints. Его мы ешё не создавали, поэтому Vivado предлагает нам создать новый файл для сохранения (рис. 47).

	Asve Constraints		×
r	Select a target file to writ the new constraints.	e new unsaved constraints to. Choosing an existing file will update that file with	2
	• Create a new file		
	<u>F</u> ile type:	Image: Image	
	File name:	constr 2	
	File location:	Si <local project="" to=""> 3</local>	
	Select an existing file	2	
	<select a="" tar<="" td=""><td>get file&gt;</td><td></td></select>	get file>	
		4	
		ОК Са	ancel

Рисунок 47 Окно создания файла для сохранения результата

В окне, показанном на рис. 47:

- 1. Тип файла. Выберите XDC Xilinx Design Constraint файл с настройками для проекта
- 2. Имя файла; наберите constr.
- 3. Расположение файла оставьте по умолчанию.
- 4. Нажмите ОК.

После сохранения результатов необходимо выполнить синтез проекта заново, о чём говорит надпись Synthesis Out-of-Date в правом верхнем углу (рис. 47).

	— (	]	×
	Q - Search commands		
	Synthesis Out-of-date	more	info
			×
			×
16 17 18			

Рисунок 48 Результаты синтеза устарели, синтез необходимо выполнить заново: Synthesis Out-of-date

Мы позволим Vivado заново выполнить синтез, но в этот раз запустим не просто синтез, а сразу генерацию битстрима – bit-файла. При этом Vivado «увидит», что результатов синтеза и имплементации нет, и предложит выполнить их в автоматическом режиме, прежде чем сгенерировать bit-файл (рис. 49).

Нажмите Generate Bitstream и затем, когда Vivado предложит выполнить синтез и имплементацию перед запуском генерации bit-файла – нажмите Yes.

🙏 Microblaze\_Lesson\_1 - [C:/Projects/FGPA-Systems/Microblaze\_Lesson\_1/Microblaze\_Lesson\_1.xpr] - Vivado 2015.4



Рисунок 49 Запуск генерации bit-файла и окно, предупреждающее что сначала будут выполнены синтез и имплементация

Теперь нам нужно ждать окончания генерации bit-файла. Это может занять минут 10-15 в зависимости от Вашего компьютера. По окончании генерации bitфайла появится окно (рис. 50), предлагающее выполнить одно из действий. Нам ничего дальше делать не нужно, поэтому его просто закрываем.

Bitstream Generation Completed	×
Bitstream Generation successfully completed. Next	
Open Implemented Design	
○ <u>V</u> iew Reports	
Open <u>H</u> ardware Manager	
Don't show this dialog again	
OK Cancel	

Рисунок 50 Окно завершения генерации bit-файла. Мы делать дальше ничего не будем, поэтому нажимаем кнопку Cancel.

На этом сборка HW-части закончена... Переходим к SW.

### Разработка SW части

SW-часть (программная) необходима, чтобы «оживить» нашу собранную процессорную систему. Сейчас это просто кусок «железа», который не выполняет никаких действий. Как уже было сказано выше, разработка программной части выполнятся в среде Xilinx SDK, которая есть по сути Eclipse с плагинами от Xilinx. Разумеется, что Xilinx уже автоматизировал часть процесса написания программы и подготовил некоторые исходные файлы и библиотеки, так что писать мы будем не «с нуля». Сейчас нам необходимо передать Xilinx SDK информацию об аппаратной «начинке» нашей процессорной системы: какие использованы устройства, какова их конфигурация и адреса и т.д. В общем – выполнить экспорт нашей аппаратной (HW) части. Сделать это можно, выбрав в левом верхнем углу File-Export-Export Hardware (рис. 51)



Рисунок 51 Экспорт НW-части проекта в Xilinx SDK

После этого появится окно (рис. 52), в котором указывается, нужно ли экспортировать bit-файл, и какую папку выбрать в качестве рабочей. Оставим все настройки в состояниях по умолчанию; bit-файл экспортировать сейчас не надо, мы добавим его позже. Нажимаем ОК.

🝌 Export Hardware	×
Export hardware platform for software development tools.	4
Include bitstream	
Export to: 🛜 <local project="" to=""></local>	•
ОК Са	ncel

Рисунок 52 Параметры экспорта HW части

Теперь запускаем программу Xilinx SDK. Сделать это можно через главное меню Windows или из-под Vivado – просто выберите в левом верхнем углу File-Launch SDK (рис. 53).



Рисунок 53 Запуск Xilinx SDK из-под Vivado

Теперь Vivado просит указать параметры запуска SDK; отставим их по умолчанию и нажмём OK (рис. 54):



Рисунок 54 Параметры запуска Xilinx SDK

Если все сделано правильно, должно появиться окно, как на рис.55. Это и есть Xilinx SDK.

C/C++ - system_wrapper_hw_platform_0/system.hdf - Xilinx SD								o ×
File Edit Source Refactor Navigate Search Project Xili	nx Tools Run Window Help							
📬 🕶 🔛 😳 💌 🔨 🕶 😚 🏭 📝 🖬 🖬 🖉 🕶 😂 🕶 🔮	• @ • 🛛 \$ • Q • Q • X 😕 🔗 • 🗉	1 k • 4 •	\$÷				Quick Access	🖆 🗟 C/C++
Project Explorer 🛛 📄 🖕 🔻 🖓 🖓 🖄	i isystem.hdf ⊠					- 0	🗄 Outline 🛛 🖲 Make Target	
	system_wrapper_hw_platform_0 H	system_wrapper_hw_platform_0 Hardware Platform Specification						
a system.hdf	Design Information							
	Target FPGA Device: 7a35ti Created With: Vivado 2015.4							
	Created On: Sat Apr 08 16:15:20 2017							
	Address Map for processor microblaze_0							
	microblaze_0_local_memory_dlmb_bram_if_cntl	r 0x00000000	0x00001ff 0x4060fff	f				
	axi_gpio_l	0 0x40000000	Ox4000fff	f				
	IP blocks present in the design							
	microblaze_0_local_memory_dlmb_bram_if_cntl	microblaze_0_local_memory_dlmb_bram_if_cntlr lmb_bram_if_cntlr 4.0						
	microblaze_0_local_memory_ilmb_bram_if_cntlr	microblaze_0_local_memory_ilmb_bram_if_cntlr_lmb_bram_if_cntlr 4.0						
	mdm_1	mdm blk mom gon	3.2					
	axi uartiite 0	in orbitazeoca_memory_mo_oram onmemory a.s.						
	microblaze 0 axi periph	axi interconne	ect 2.1					
	rst_clk_wiz_1_100M	proc_sys_reset	5.0					
	clk_wiz_1	clk_wiz	5.2					
	xlconstant_0	xlconstant	1.1					
	microblaze_0	microblaze	9.5					
	axi_gpio_0	axi_gpio	2.0 Reg	<u>iisters</u>				
	microblaze_0_local_memory_dlmb_v10	microblaze_0_local_memory_dlmb_v10 lmb_v10 3.0						
	microblaze_0_local_memory_ilmb_v10	microblaze_0_local_memory_ilmb_v10 lmb_v10 3.0						
	Overview							
👛 Target Connections 🛛 🥔 🖉 🗖	🗄 Problems 🛛 🖉 Tasks 📮 Console 🚍 Properti	es 💷 SDK Term	ninal	~	🗐 SDK Log 🛛			🕞 🕞 🗖 🗖
> 🗁 Hardware Server	0 items				08:32:06 INFO	: Launching XSDB ser	ver: xsdb.bat C:/Xilinx/S	DK/2015.4/s ^
> 🗁 Linux TCF Agent	Description	Reso	ource	Path	08:32:09 INFO	: XSDB server has st	arted successfully.	
> 🗁 QEMU TcfGdbClient					08:32:11 INFO	: Processing command	l line option -hwspec C:/P	rojects/FGP
								~
	<			>	<			>

Рисунок 55 Основное окно Xilinx SDK

Сейчас мы не будем разбирать назначение каждого из внутренних окон системы, поскольку это выходит за рамки данной статьи. Единственное, о чём я

могу упомянуть – так это о том, что у Xilinx SDK очень-очень-очень хорошая справка (рис. 56). Если у Вас возникают вопросы, в основном все ответы лежат в справке. Не поленитесь, ознакомиться с ней, если будет время.

Window	Hel	p 1	
- O -	3	Welcome	
f⊠	?	Help Contents 2	
•	×2	Search	
wrappe		Dynamic Help	
ormation		Key Assist Tips and Tricks	Ctrl+Shift+L
A Device:		Cheat Sheets	
ited With:			
eated On:	$\checkmark$	Enable WebTalk	
ap for pro ≥_0_local_n	<b>≥</b> 3 ⊗	Acquire a License Key Manage License Check for Updates Install New Software	
		About Xilinx SDK	

Рисунок 56 Вызов справки Xilinx SDK

Приступим к созданию проекта. Выбираем File-New-Application project

SOK (	SOK C/C++ - system_wrapper_hw_platform_0/system.hdf - Xilinx SDK												
File	1 dit	Source	Refactor	Navigate	Search	Project	Xilinx Too	ls I	Run	Window	Help	_	
	New		2			Alt+	-Shift+N>	Q	Арр	olication P	roject	3	
	Open	File							SPN	A Project		-	F
	Close						Ctrl+W	lıî ⊨≙	Boa Proi	ird Suppor	t Packag	ge	
	Close	All				Ctrl+	Shift+W		5.00	rse Felder			
	Save						Ctrl+S		30u				

Рисунок 57 Создание нового проекта в Xilinx SDK

Задаём основные параметры проекта (рис. 58):

New Project — 🗆 🗙
Application Project
Create a managed make application project.
Project name: microblaze_lesson_1 Use default location 2 Location: C:\Projects\FGPA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_ Browse Choose file system: default ~
Target Hardware
Hardware Platform: system_wrapper_hw_platform_0 4 Vew
Processor: microblaze_0 5
Target Software
Compiler: 32-bit V
Board Support Package:  Create New microblaze_lesson_1_bsp 7
○ <u>U</u> se existing ~
8
Image: Second

Рисунок 58 Параметры нового проекта

На рис. 58:

- 1. Название проекта.
- Место расположения проекта (выставляем используемое по умолчанию).
- 3. Тип используемой в проекте операционной системы (без ОС standalone).
- 4. Аппаратная конфигурация (если процессоров системе несколько, то будет список; у нас только один).

- 5. Идентификатор процессора (актуально для многопроцессорных систем).
- 6. Язык программирования (чистый С).
- 7. Пакет драйверов (создаём новый на основании информации о HWчасти).
- 8. Нажимаем Next.

Теперь SDK предлагает выбрать один из нескольких готовых вариантов приложения (application), включая Hello, world (рис. 59). Мы же с Вами, для более подробного знакомства со структурой приложения, выберем пустое приложение (Empty Application), нажимаем Finish.

SOK New Project			—	
Templates Create one of the avai project.	lable templates to gene	erate a fully-fu	nctioning applicatior	
Available Templates:				
Dhrystone Empty Application Hello World IwIP Echo Server Memory Tests Peripheral Tests SREC Bootloader SREC SPI Bootloader		A blank C	project.	
			2	
?	< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cancel

Рисунок 59 Выбор типа проекта – пустой проект

После нажатия на кнопку Finish в проект добавляется несколько файлов и папок (рис. 60).



Рисунок 60 После создания проекта добавляются приложение и драйверы.

В нашем приложении ещё нет файла с исполняемой программой. Нам нужно создать его и добавить в проект. Поскольку качестве языка программирования был выбран С, то нужно добавить файл с расширением «.с». Для добавления нового файла в проект нажмите на значку нашего приложения правой кнопкой и выберите New – File (рис. 61).



Рисунок 61 Добавление нового файла в приложение

Теперь нужно указать параметры создаваемого файла (рис. 62). Путь к папке, в которой он будет создан, можно указать в строке 1, или же в дереве папок выбрать папку, в которой он будет создан

- SOK New File	_		×
File Create a new file resource.			
Enter or select the parent folder: microblaze_lesson_1/src 1 microblaze_lesson_1 microblaze_lesson_1 microblaze_lesson_1_bsp RemoteSystemsTempFiles \$\$\$ system_wrapper_hw_platform_0			
File na <u>m</u> e: program.c <b>3</b>			
1	4		
?	<u> </u>	Cancel	

Рисунок 62 Создание и добавление нового файла в проект.

На рис. 62:

- 1. Путь к файлу относительно каталога приложения.
- 2. Путь к файлу в дереве директорий.
- 3. Указываем название файла (program.c).
- 4. Нажмите Finish.

Созданный файл появится в папке src (рис. 63). Чтобы просмотреть его содержимое, дважды кликните по нему. После создания файла появилась ошибка, о чем говорит красный крестик в папке приложения. Дело в том, что файл – пустой, и в приложении нет функции main(). Нам необходимо написать программу в наш файл.



Рисунок 63 Ошибка из-за отсутствия функции main()

Откройте файл program.c и заполните его следующим кодом:

```
#include "xparameters.h" //Библиотека с параметрами IP-блоков
#include "xgpio.h" //Библиотека с функциями GPIO
ХGpio gpio; //<u>Создаем</u> "программную" модель GPIO
int main(){
      u32 i
              = 0; //используем для задержки
      u32 led = 0; //состояние светодиода
      XGpio Initialize(&gpio, XPAR GPIO 0 DEVICE ID);//Находим и инициализируем GPIO
      xil printf("Hello, world!!!\n\r");//Автоматически цепляется Uartlite и выводит
сообщение
      while(1){ //Бесконечный цикл мигания
             i++; //увеличиваем счётчик
             if(i == 1000000){//Если достигнуто значение 1 000 000
                    led = !led;//Инвертируем состояние светодиода
                    XGpio_DiscreteWrite(&gpio, 1, led);//Записываем состояние светодиода в
GPIO
                    і = 0;//Сбрасываем сётчик
             }
      }
      return 0;
}
```

Листинг 1.

Файл храгатеters.h является одним из наиболее важных файлов, который содержит адреса и номера устройств на шине нашей процессорной системы. Именно в нём определён номер (ID) нашего GPIO - XPAR\_GPIO\_0\_DEVICE\_ID. Если бы GPIO модулей было бы несколько, то для каждого из них было бы отдельное описание параметров адреса, ID и т.д. Файл храгатеters.h можно найти (рис. 64) в Project Explorer – название\_приложения\_bsp – процессор – include

SOK	C/C++ -	micro	blaz	e les	son 1	/src/	prog	ram.c -
File	e Edit	Sour	ce l	Refac	tor	Navi	igate	Sear
2	- II II		· @	~ %	-	6	송 🛛	1 🕱 🖬
8	Proje	ct Exp	lorer	53			- 221 -	
ö		icrobl	aze	esso	n 1			
	v 🏙 m	icrobl	aze	esso	n 1 h	sn		
	i	BSP	Docu	ment	ation	1		
		micro	oblaz	te 0				
	1	⊖ co	de		3			
	-	😕 in	clude	2				
		> .h	_pro	ofile_t	timer	_hw.h	E	
		> .h	bsp	confi	g.h			
		> .h	fsl.h	-				
		> .h	mb_	inter	face.	h		
		> .h	mbl	aze_r	nt_typ	bes.h		
		> .h	mic	robla	ze_e	kcept	ions_	g.h
		> .h	mic	robla	ze_e	kcept	ions_	i. <mark>h</mark>
		> .h	mic	robla	ze_in	terru	pts_i.	n
		> .h	mic	robla	ze_sl	eep.h		
		> .h	pro	file.h				
		> .h	pvr.	h				
		> .h	xba	sic_ty	pes.h	n i i		
		> .h	xbra	am_h	w.h			
		> .h	xbra	am.h				
		> .h	xde	bug.h	1			
		> .h	xen	v_star	ndalo	ne.h		
		> .h	xen	v.h				
		> .h	xgp	io_l.h				
		> .h	xgp	io.h				
		> .h	xil_a	assert	.h			
		> .h	xil_0	ache	_vxw	orks.h	n	
		> .h	XII_C	ache	.h			
		> h	XII_6	excep	tion.	h		
		> h	XII_I	nal.n				
		> .h	XII_I	o.n				
	72	> 10	XII_I	nacro	obaci	c.n		
			XII_I	nisc_	psres b	et_ap	1.11	
				octor	ul cho b			
			vil +	estic	h	1		
			vil +	estin	emb			
			xil t	vnes	h			
		5 6	xio	h h				
		> .h	xpa	rame	ters.h			
		> [h]	xpla	tforn	n info	o.h		

Рисунок 64 Расположение файла xparameters.h

Мы создали программу, теперь можем её попробовать запустить. Мы не будем отлаживаться в пошаговом режиме, а сразу посмотрим результат. Но для начала нам необходимо подключить плату к компьютеру, что бы он определил какой СОМ порт будет задействован для UARTa.

Для просмотра сообщений от UART мы будем использовать SDK terminal. Чтобы открыть SDK terminal, необходимо найти его в каталоге доступных окон. Для этого выберите Window – Show View – Others..., затем наберите в строке поиска SDK terminal, выберите его и нажмите OK (рис. 65).



Рисунок 65 Открытие окна SDK terminal

После этого должно появиться окно SDK terminal, которое мы будем использовать, чтобы «ловить» сообщения, посылаемые нашим Uartlite модулем. Теперь давайте настроим SDK terminal (рис. 66). Выберите окно SDK terminal, нажмите кнопку подключения к последовательном порту (зелёный крестик) и установите значения в соответствии с рис. 66, затем нажмите OK.

# www.FPGA-Systems.ucoz.net

	Basic Settings Port: COM - 3 Baud Rate 9600
0 Hardware Platform Specification	<ul> <li>✓ Advance Settings</li> </ul>
2017 f_cntlr 0x00000000 0x00003fff tlite_0 0x40600000 0x4060ffff jpio_0 0x40000000 0x4000ffff	Data Bits: 8   Stop Bits: 1   Parity: None   Flow Control: None   Timeout (sec) OK
f cntlr Imb bram if cntlr 4.0	~ ~
Console 🖉 Task 🔄 🖾 SDK Terminal 🛛 👔 Problems 📮 Conso	le
Click on + button to add a port to the terminal.	
<	~
	Send Clear

Рисунок 66 Настройка терминала SDK

Настройка терминала выполнена. По завершении настройки терминала, в нём может появиться «мусор». Если он Вам мешает нажмите кнопку очистить *с*, которая находится рядом с кнопкой подключения к порту (зелёный крестик 2 на рис. 66).

Теперь необходимо настроить запуск нашей программы: прошить FPGA и загрузить приложение в процессор. Всё это делается в одном окне, но в разных вкладках. Для настройки запуска (рис. 67) нажмите на треугольник рядом с кнопкой Run



Рисунок 67 Вызов настроек запуска

Окно на рис. 68 предназначено для выбора отладчика или инструмента запуска настройки запуска процессорной системы. Дважды кликните на System Debugger.

SON Run Configurations		×
Create, manage, and run configurations		
Image: Second	<ul> <li>Configure launch settings from this dialog:</li> <li>Press the 'New' button to create a configuration of the selected type.</li> <li>Press the 'Duplicate' button to copy the selected configuration.</li> <li>Press the 'Delete' button to remove the selected configuration.</li> <li>Press the 'Filter' button to configure filtering options.</li> <li>Edit or view an existing configuration by selecting it.</li> <li>Configure launch perspective settings from the 'Perspectives' preference page.</li> </ul>	
. (?)		Run Close

Рисунок 68 Окно выбора инструмента запуска

После этого окно должно приобрести вид, соответствующий рис. 69. Теперь можем приступить к настройке запуска.

Son Run Configurations		
Create, manage, and run configurations Run or Debug a program using System Debugger.		
Image: Second Secon	Name:       New_configuration	Browse
Filter matched 6 of 13 items	Apply	Re <u>v</u> ert
. (?)	Run	Close

Рисунок 69 Исходное окно настроек запуска

Измените название наших настроек на system\_run\_1, затем выберите Debug Type: Standalone Application Debug (рис. 70), после чего окно должно измениться.

SOK Run Configurations		×
Create, manage, and run configurations Run or Debug a program using System Debugger.		
Image: Second Secon	Name: system_run_1 1 Target Setup Application I Application Arguments Environment Symbol Files Symbol Files Source & Path Map Common Debug Type: Attach to running target 2 Connection Standalone Application Debug Execute ScAttach to running target Summary Debugger will connect to the target and display current status of all the processors. No resets or initializations will be performed on the target before launching the debugger.	Browse
Filter matched 6 of 13 items	Apply	Re <u>v</u> ert
۲	Bun	Close

Рисунок 70 Переименование настроек и выбор типа отладки

После изменения, заполните содержимое в соответствии с рис. 71

Son Run Configurations		
Create, manage, and run configurations		
Run or Debug a program using System Debugger.	Name: system_run_1  Target Setup Application Application Application Connection: Coal New Hardware platform: system_wrapper_hw_platform_0 Bitstream file: FPGA Device: Auto Detect Select FPGA Device: Auto Detect Select Summary of operations will be performed Following operations operations will be performed Following operat	nerate
Filter matched 6 of 13 items	Apply Re	e <u>v</u> ert
0	<u>R</u> un C	Close

Рисунок 71 Настройка НW параметров запуска

На рис. 71 обозначены:

1 Путь к бит файлу. Должен быть указан полный путь. Находится внутри папки с проектом (название\_проекта – нзвание\_проекта.runs – impl\_1 – .bit)

2 Сброс системы после загрузки

- 3 Запрограммировать FPGA
- 4 Перейдите во вкладку Application

Run Configurations						×
<b>Create, manage, and run configurations</b> Run or Debug a program using System Debugger.						
Y       Image: Second Se	Name: system_run_1  Target Setur Application Available Processors Dow microblaze_0  Proje Appli  Th Relat Re Stu File File	Arguments) The Environment is S noad application 1 at Name: microblaze_lesson_1 2 cation: Debug/microblaze_lesson_1.4 is is a self-relocating application ve address to which the program section set processor up at program entry iles to download before launch	ymbol Files	rce & Path M	ap Common	Browse Browse Add Remove Relative
Filter matched 6 of 13 items				4	Appl <u>y</u>	Re <u>v</u> ert
0				5	<u>R</u> un	Close

Рисунок 72 Настройка параметров запуска приложения

На рис. 72:

- 1 Загрузить приложение после загрузки FPGA
- 2 Название приложения, которое нужно загрузить
- 3 Путь к исполняемому .elf файлу приложения
- 4 Кнопка применить настройки
- 5 Запуск приложения

После нажатия кнопки Run произойдёт загрузка FPGA и запуск приложения, в результате чего светодиод LD4 на плате Arty Board должен начать мигать, а в SDK terminal должно появиться сообщение «Hello, world!», рис. 73.

💷 Console 🖉 Tasks 🖳 SDK Terminal 🛛 🔝 Problems 🖳 Console	🕂 🗶 🖓 🗖
Connected to: Serial ( COM6, 9600, 0, 8 )	
Hello, world!	^
	✓
<	>
	Send Clear

Рисунок 73 Hello, world! В SDK terminal

На этом сборка и запуск процессорной системы, построенной на базе MicroBlaze, закончена.

#### Заключение

Не смотря на то, что материал занимает более 60 страниц, для небольших проектов сборка процессорной системы на базе MicroBlaze занимает не многим больше 10 минут. Во многом этот процесс автоматизирован, что помогает избежать пользовательских ошибок. Ниже, в библиографическом списке, приведена некоторая подборка документов, которые могут послужить В качестве дополнительного материала для более глубокого понимания, как процесса разработки, так и функциональности используемых IP-блоков, а также приведены ссылки на доступные по данной тематике тренинги в сертифицированном тренинг центре компании Xilinx.

Если кто-то захочет попрактиковаться, то вот Вам домашнее задание:

- На Arty Board установлены четыре зелёных светодиода. Сделайте из них бегущие огни или бегущую тень или счётчик. Тоже можно сделать и с трёхцветными светодиодами.
- 2. Настройте GPIO таким образом, что бы можно было считывать состояния кнопок, установленных на плате, и отображать их состояние в SDK terminal.
- 3. Сделайте запуск бегущих огней по кнопке любой на Ваш выбор.
- 61

# Библиографический список

- 1. <u>UG984</u>. MicroBlaze Processor Reference Guide. Xilinx Inc.
- 2. <u>MicroBlaze на сайте Xilinx</u>
- 3. <u>Vivado на сайте Xilinx</u>
- 4. DS180. 7 Series FPGAs Data Sheet: Overview. Xilinx Inc.
- 5. <u>Описание</u> Arty Board на сайте Digilent
- 6. Сайт компании Регион-Вирта
- 7. UG995. Designing IP Subsystems Using IP Integrator. Xilinx Inc.
- 8. UG782. Software Development Kit (SDK) Help
- 9. PG144. AXI GPIO LogiCORE IP Product Guide
- 10. PG142. AXI UART Lite LogiCORE IP Product Guide
- 11. PG059. AXI Interconnect LogiCORE IP Product Guide
- 12. UG1037. Vivado AXI Reference Guide

# 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