

**ПРОГРАММИРОВАНИЕ ЗАГРУЗОЧНОЙ FLASH ДЛЯ ЗАПУСКА
MICROBLAZE**

Автор:

KeisN13

Рецензенты:

Aspect

Материал подготовлен при поддержке:

КТЦ «Инлайн групп» - дистрибьютор фирмы Xilinx (www.plis.ru)

АТР 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 софтверного процессора 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 т.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 какой-либо самопальный скрипт для выполнения тех или иных действий. Призываю всех познакомиться с Tcl консолью и языком Tcl. Синтаксис языка не сложный, а

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:

- Битрим 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 указана 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 эта информация пока избыточна, что нам точно надо знать так это то, что у нас стоит QSPI FLASH с возможностью 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).

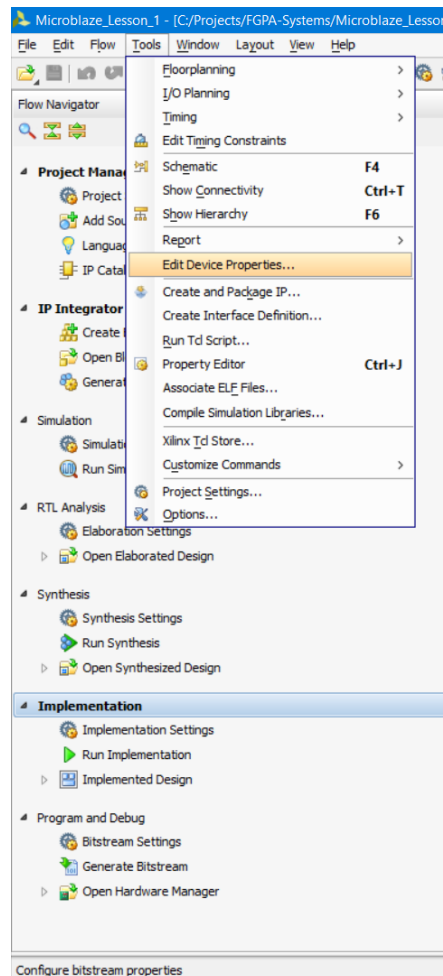


Рисунок 1 Вызов мастера настроек конфигурационного файла (д.б открыт имплементированный проект)

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

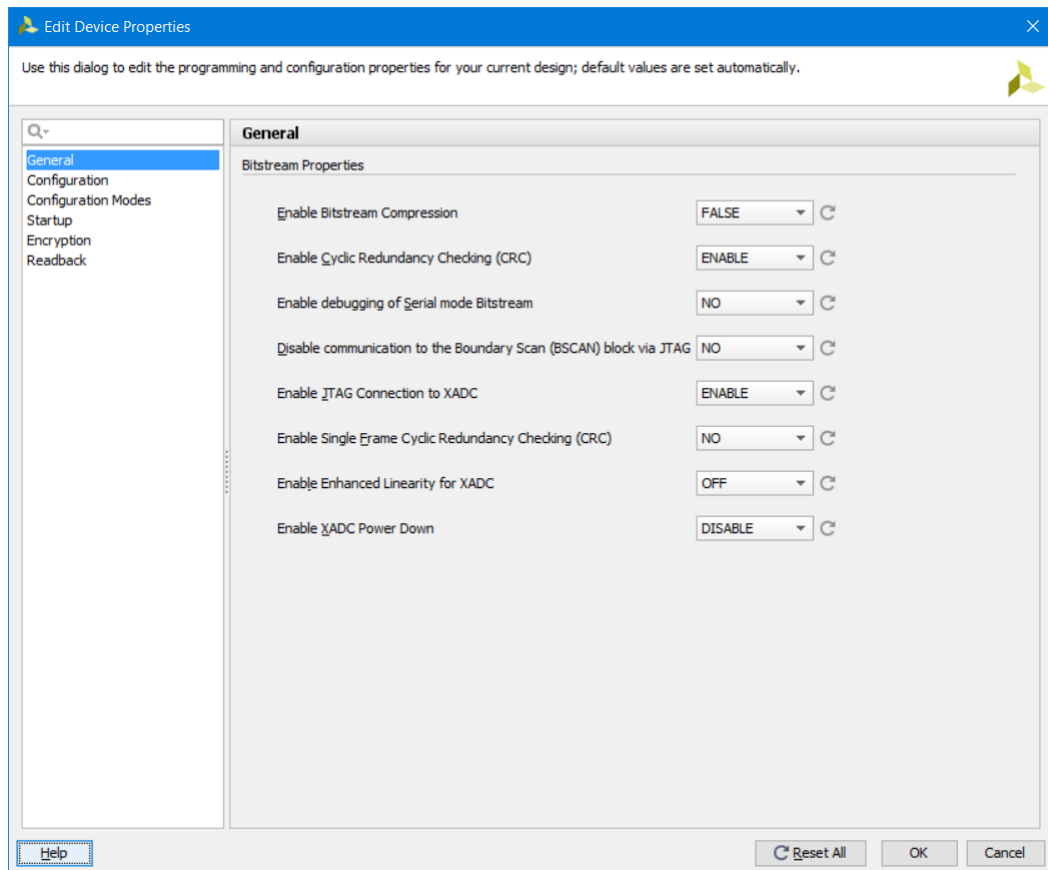


Рисунок 2 Окно настроек конфигурации

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

Configuration Rate (MHz) 33

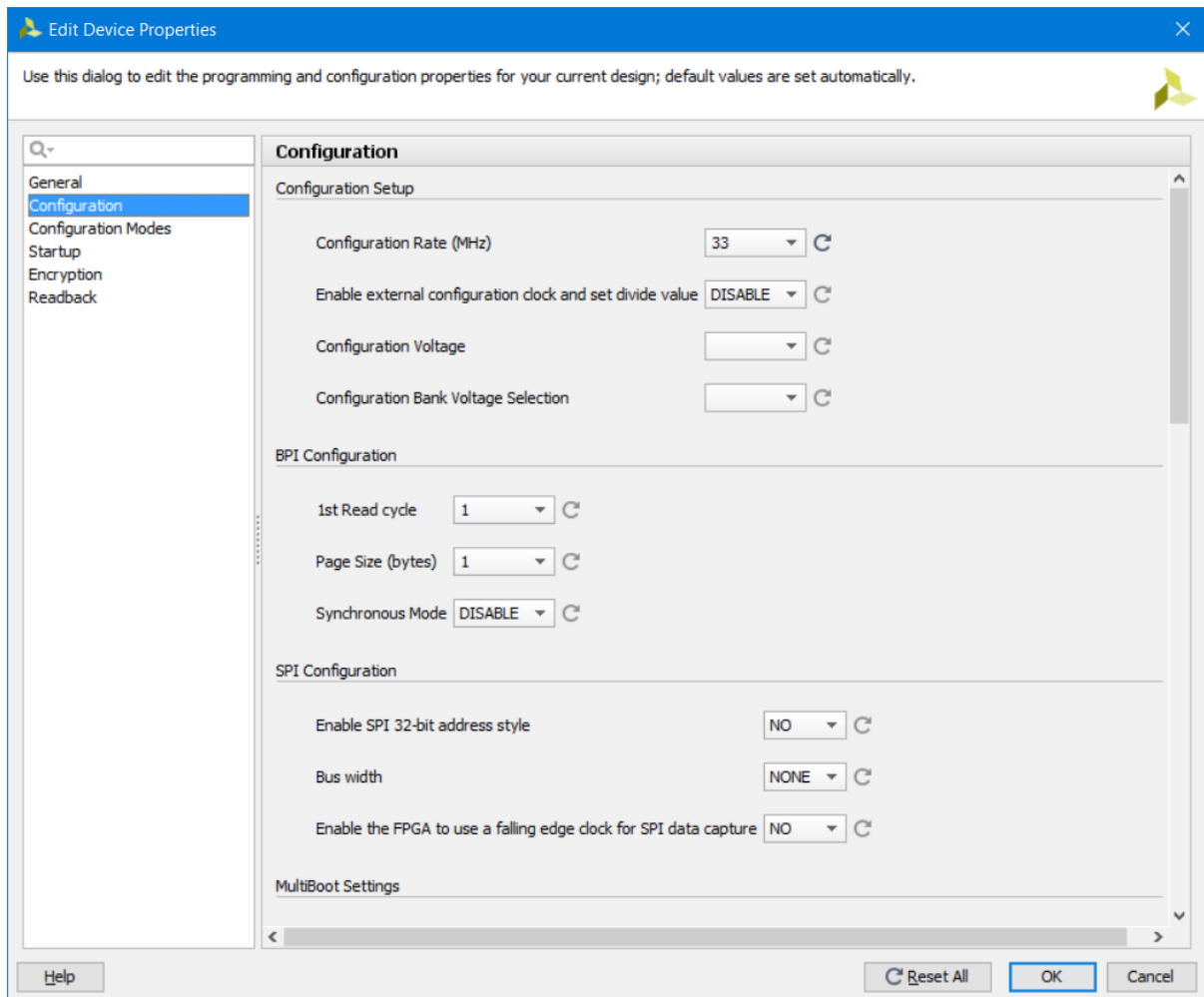


Рисунок 3 Окно настройки конфигурации: Configuration

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

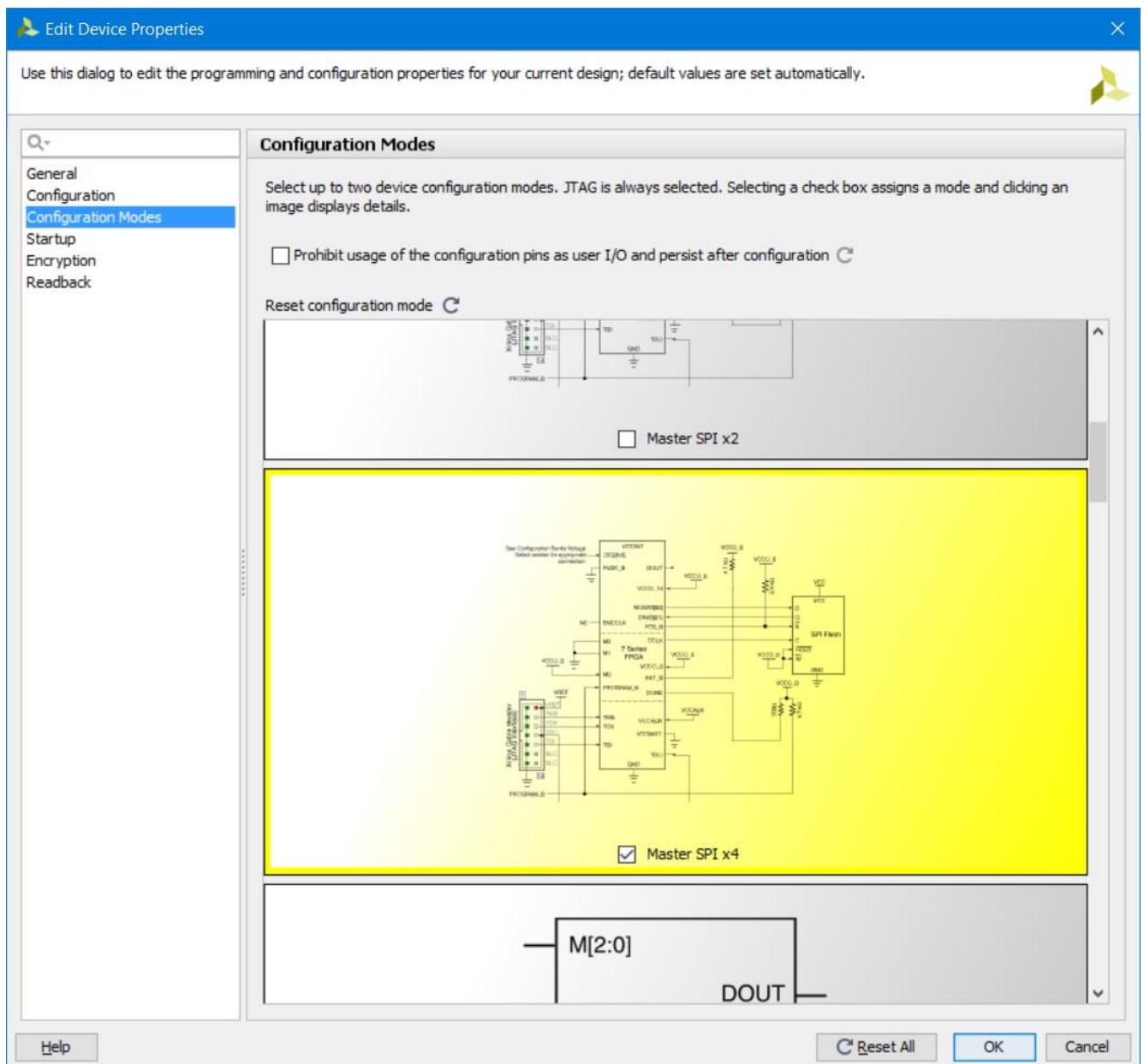


Рисунок 4 Окно настройки конфигурации: Configuration Modes

Все остальное оставим по умолчанию и нажимаем кнопку ОК.

Запускаем генерацию битстрима 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)

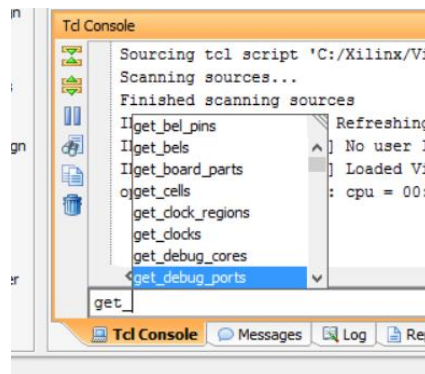


Рисунок 5 Интерактивное дополнение команд по введённым символам “get_”

Однако если Вы будете вводить команду `updatemem`, то она не появится в списке доступных. Но это не должно Вас останавливать.

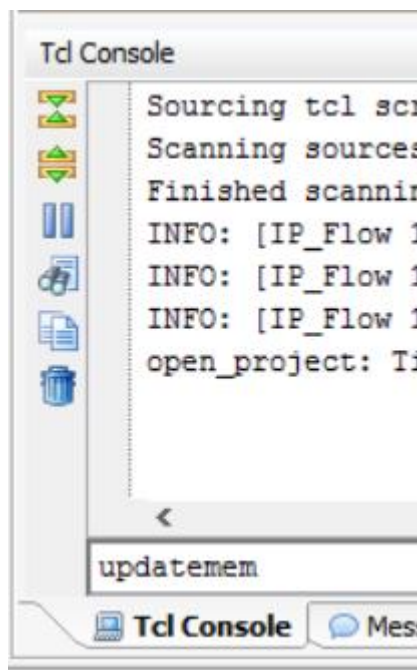


Рисунок 6 Отсутствие команды `updatemem` в консоли

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

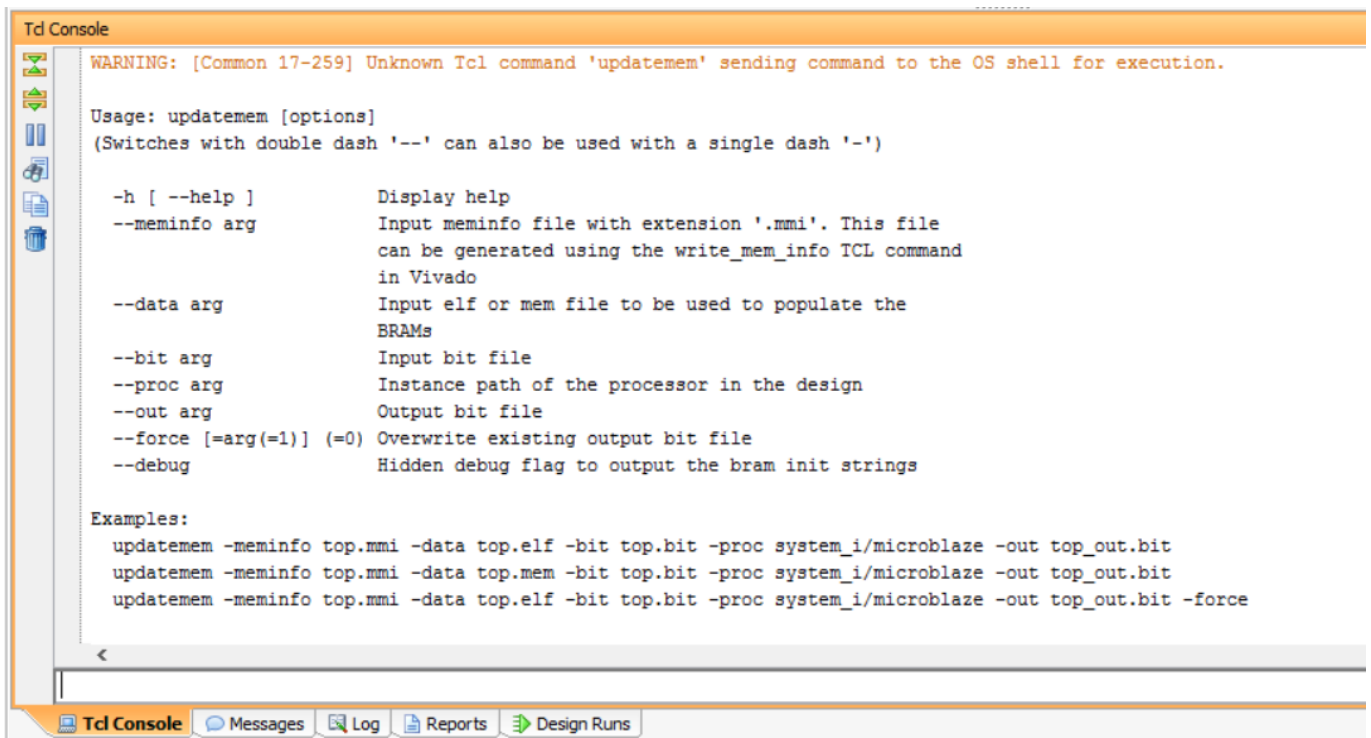


Рисунок 7 Справка по команде updatemem

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)

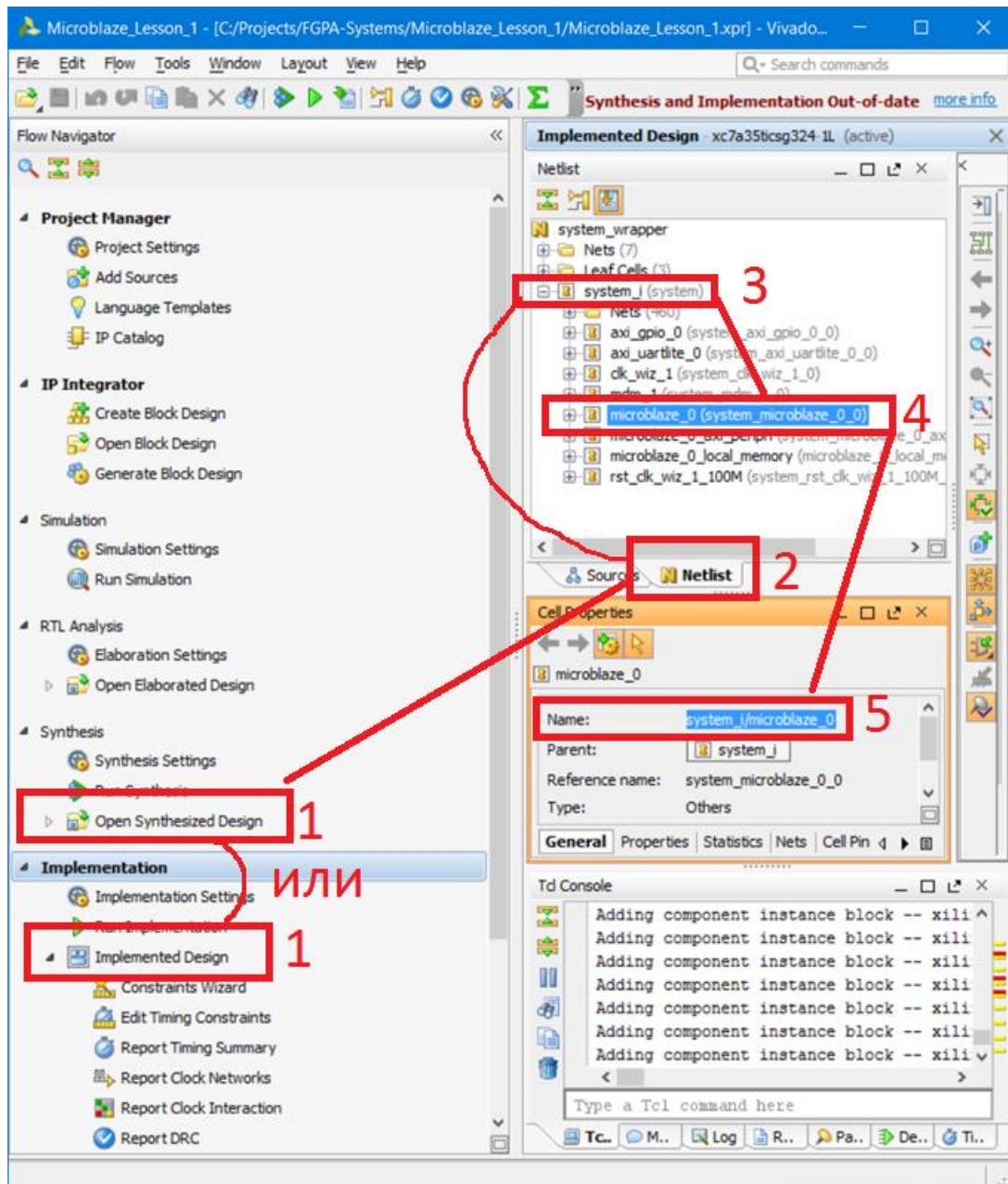


Рисунок 8 Поиск названия процессора в иерархии проекта

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.

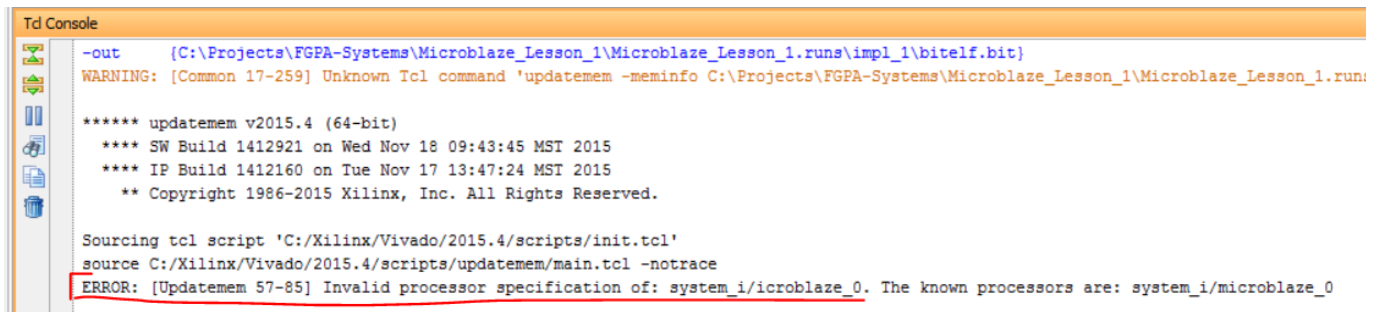
```

1 updatemem\
2 -meminfo {C:\Projects\FPGA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\system_wrapper.mmi}\
3 -data {C:\Projects\FPGA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.sdk\microblaze_lesson_1\Debug\microblaze_lesson_1.elf}\
4 -bit {C:\Projects\FPGA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\system_wrapper.bit}\
5 -proc {system_j/microblaze_0}\
6 -out {C:\Projects\FPGA-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)



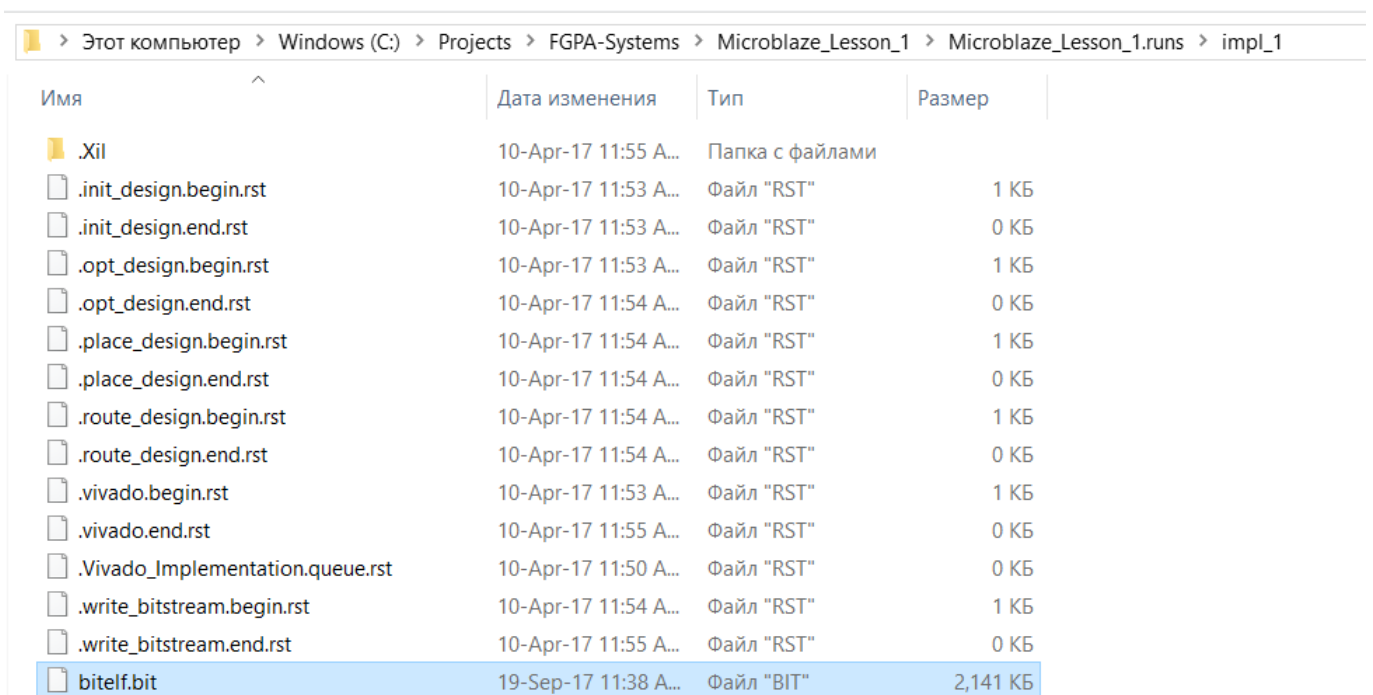
```
Tcl Console
-out (C:\Projects\FPGA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\bitelf.bit)
WARNING: [Common 17-259] Unknown Tcl command 'updatemem -meminfo C:\Projects\FPGA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\bitelf.bit'

***** updatemem v2015.4 (64-bit)
**** SW Build 1412921 on Wed Nov 18 09:43:45 MST 2015
**** IP Build 1412160 on Tue Nov 17 13:47:24 MST 2015
** 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/microblaze_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 появился в соответствующей директории (рис.11)



Имя	Дата изменения	Тип	Размер
.Xil	10-Апр-17 11:55 A...	Папка с файлами	
.init_design.begin.rst	10-Апр-17 11:53 A...	Файл "RST"	1 КБ
.init_design.end.rst	10-Апр-17 11:53 A...	Файл "RST"	0 КБ
.opt_design.begin.rst	10-Апр-17 11:53 A...	Файл "RST"	1 КБ
.opt_design.end.rst	10-Апр-17 11:54 A...	Файл "RST"	0 КБ
.place_design.begin.rst	10-Апр-17 11:54 A...	Файл "RST"	1 КБ
.place_design.end.rst	10-Апр-17 11:54 A...	Файл "RST"	0 КБ
.route_design.begin.rst	10-Апр-17 11:54 A...	Файл "RST"	1 КБ
.route_design.end.rst	10-Апр-17 11:54 A...	Файл "RST"	0 КБ
.vivado.begin.rst	10-Апр-17 11:53 A...	Файл "RST"	1 КБ
.vivado.end.rst	10-Апр-17 11:55 A...	Файл "RST"	0 КБ
.Vivado_Implementation.queue.rst	10-Апр-17 11:50 A...	Файл "RST"	0 КБ
.write_bitstream.begin.rst	10-Апр-17 11:54 A...	Файл "RST"	1 КБ
.write_bitstream.end.rst	10-Апр-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 mcs, которым прошьём FLASH.

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

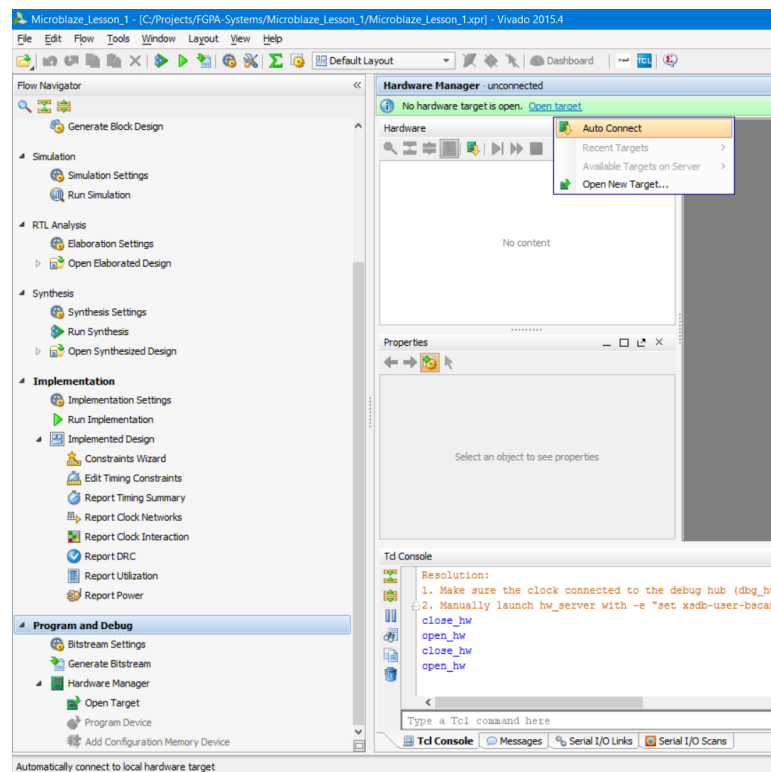


Рисунок 12 Подключение Arty

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

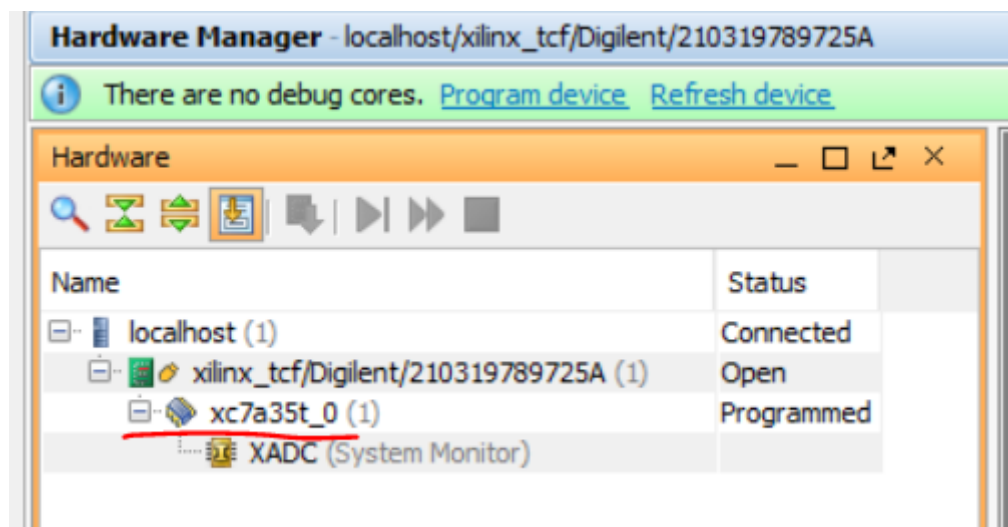


Рисунок 13 Список подключённых устройств

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

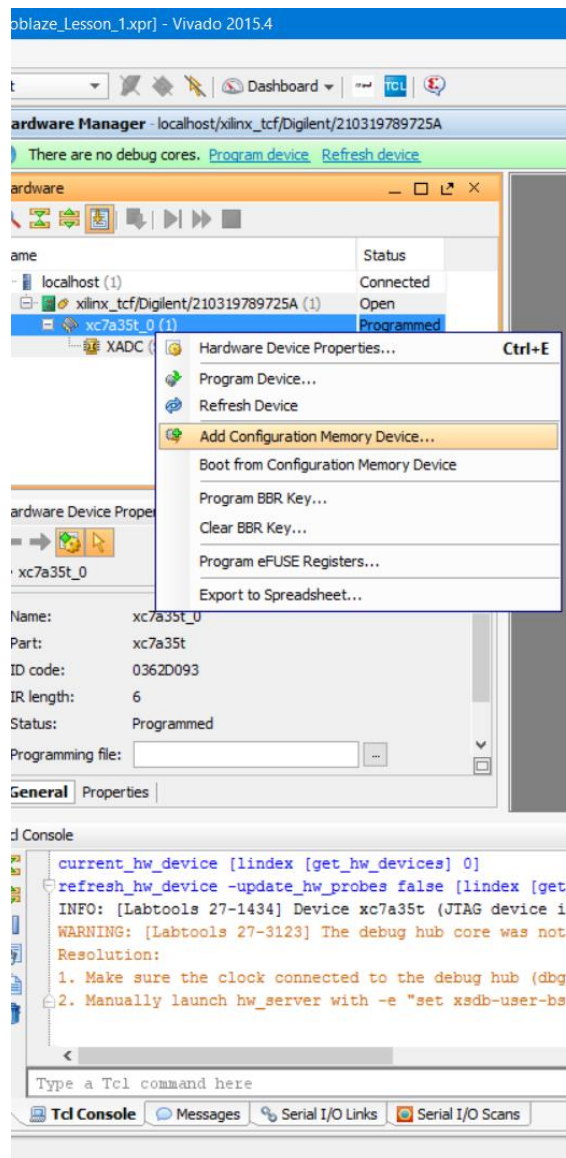


Рисунок 14 Добавление конфигурационного устройства, вызов мастера

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

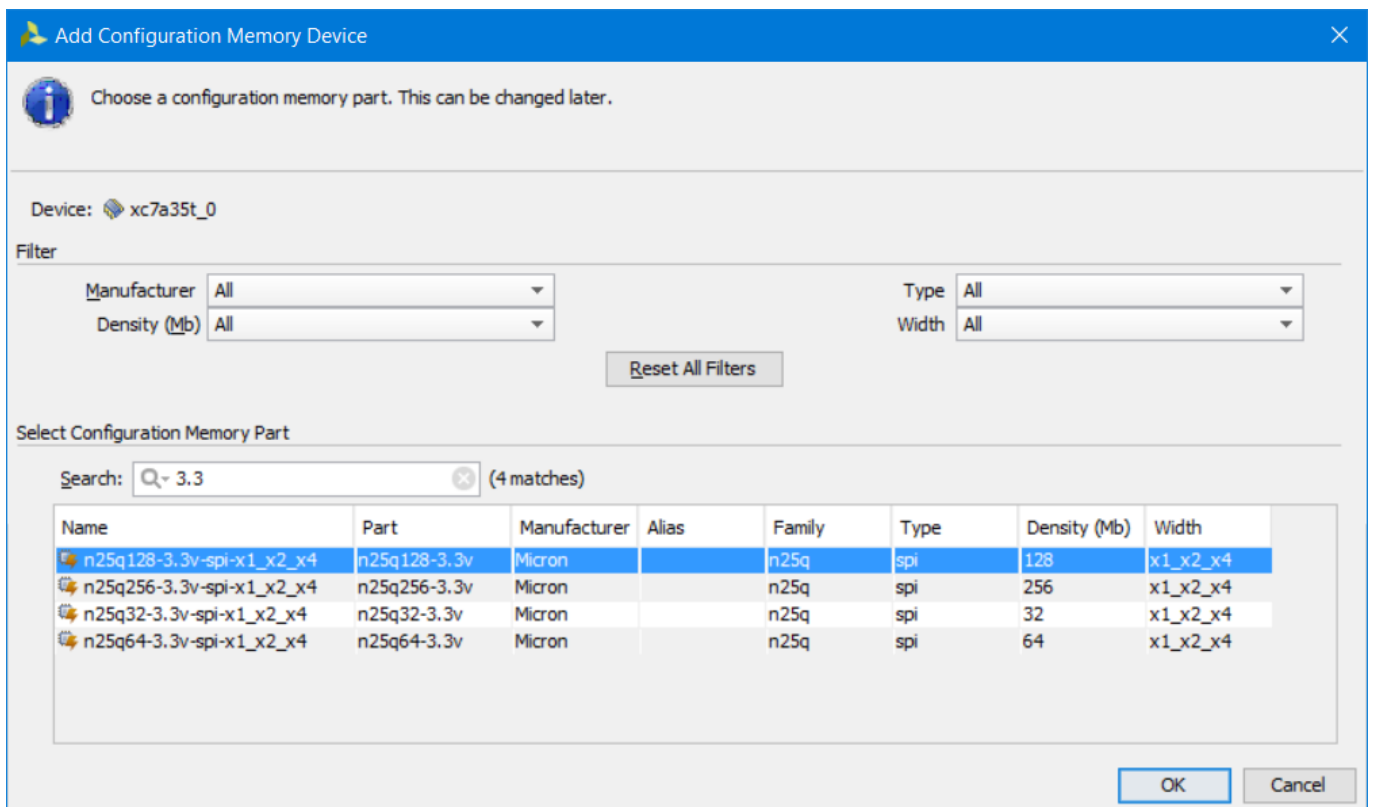


Рисунок 15 Выбор конфигурационной FLASH

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

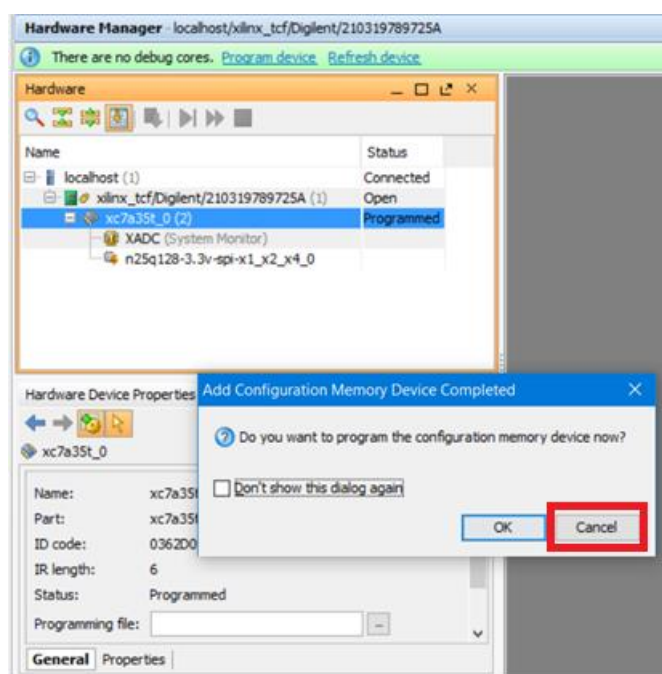


Рисунок 16 Окна запуска мастера создания mcs файла

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]

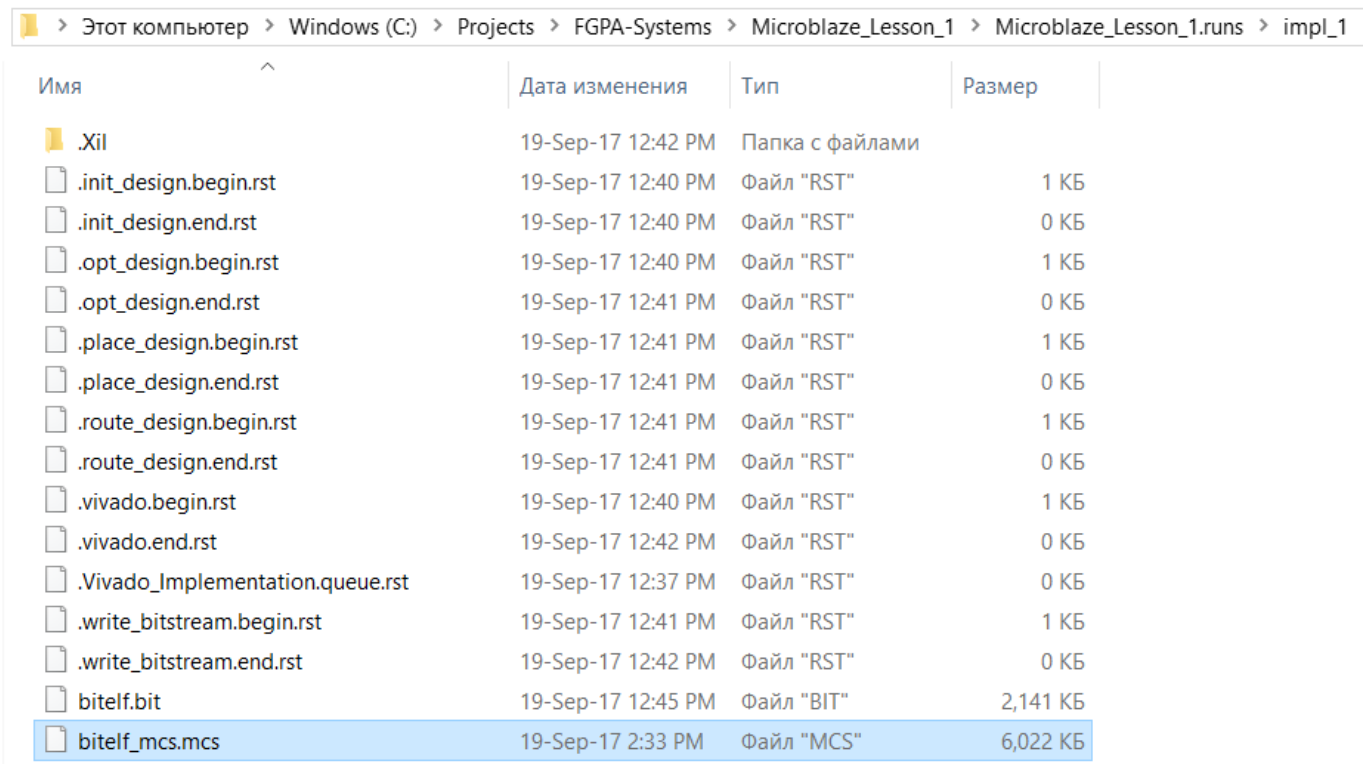
Я не буду Вас томить описанием всех параметров. Скажу, что для нашего случая достаточно указать «правильный» bit файл (bitelf.bit) и куда сохранить сгенерированный mcs (bitelf_mcs.mcs) файл. Размер нашей FLASH 16MB. Интерфейс мы выставили spi_x4. Опция –force нужна, чтобы можно было переписать файл при 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\FPGA-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.



Имя	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 mcs 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).

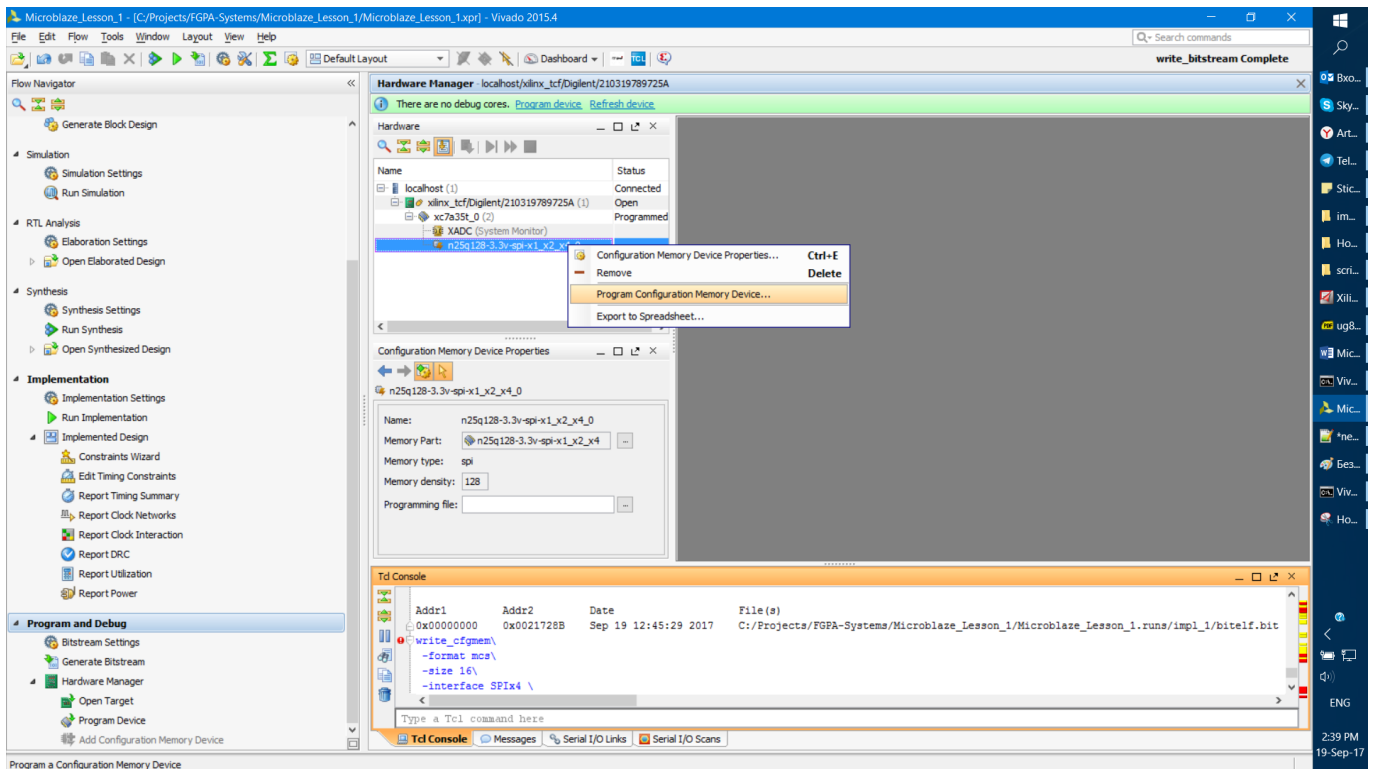


Рисунок 19 Запуск мастера программирования конфигурационной FLASH

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

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

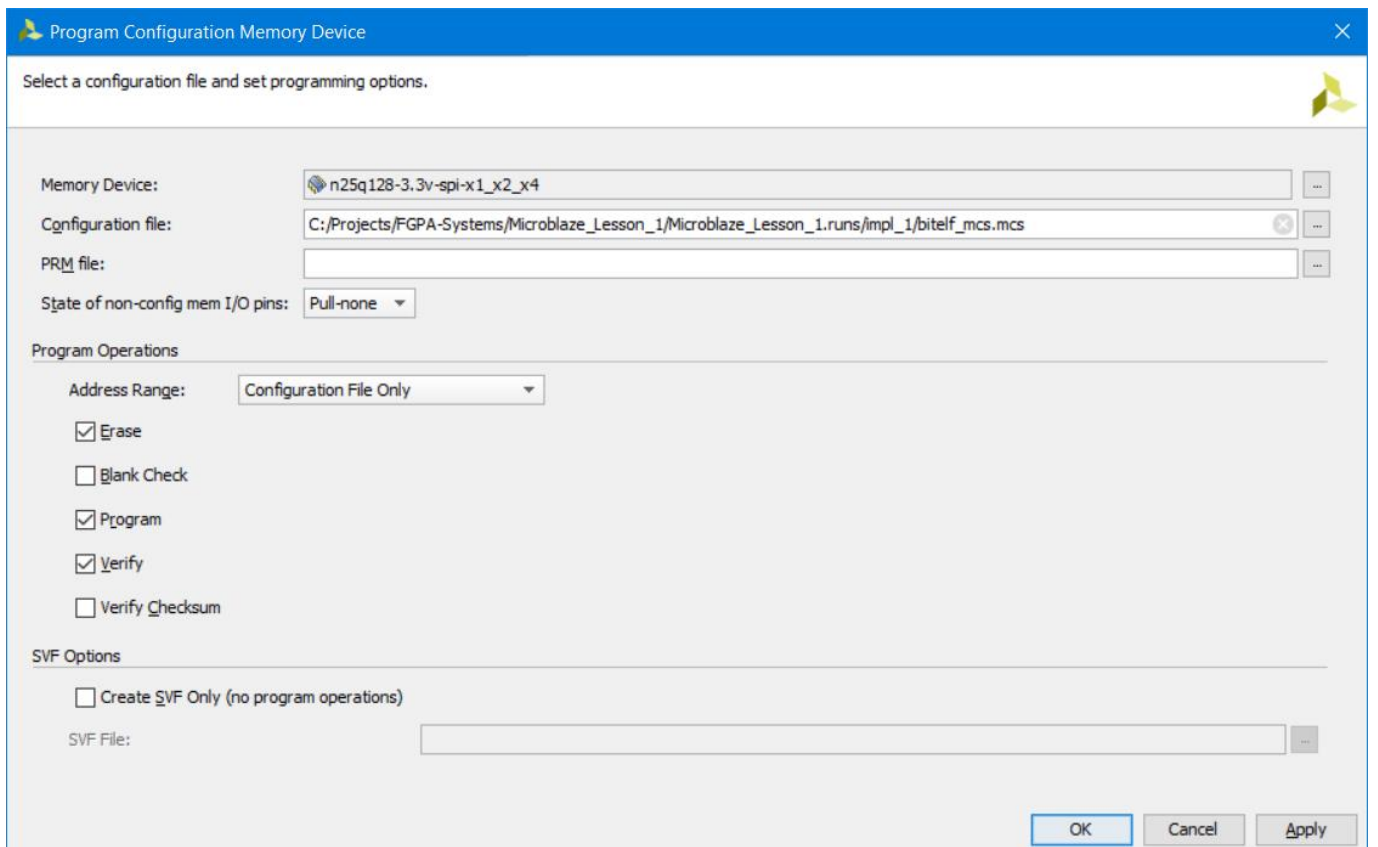


Рисунок 20 Мастер настроек конфигурации

После окончания программирования FLASH можно либо переподключить USB кабель, либо нажать кнопку 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

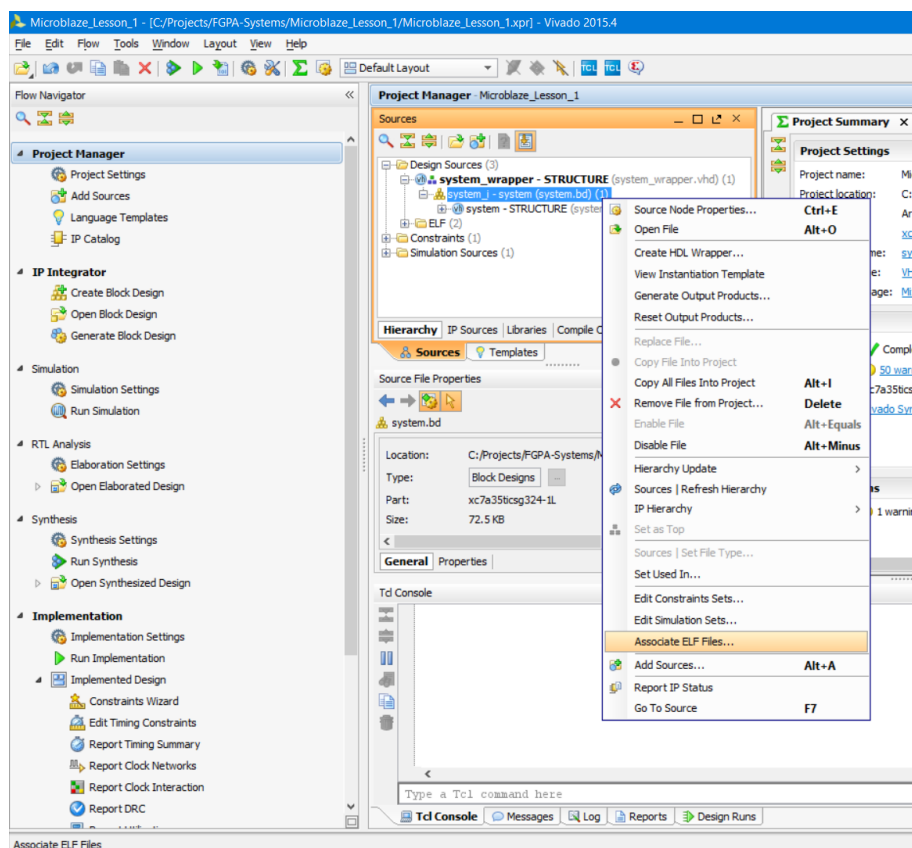


Рисунок 21 Запуск мастера присоединения исполняемых файлов

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.

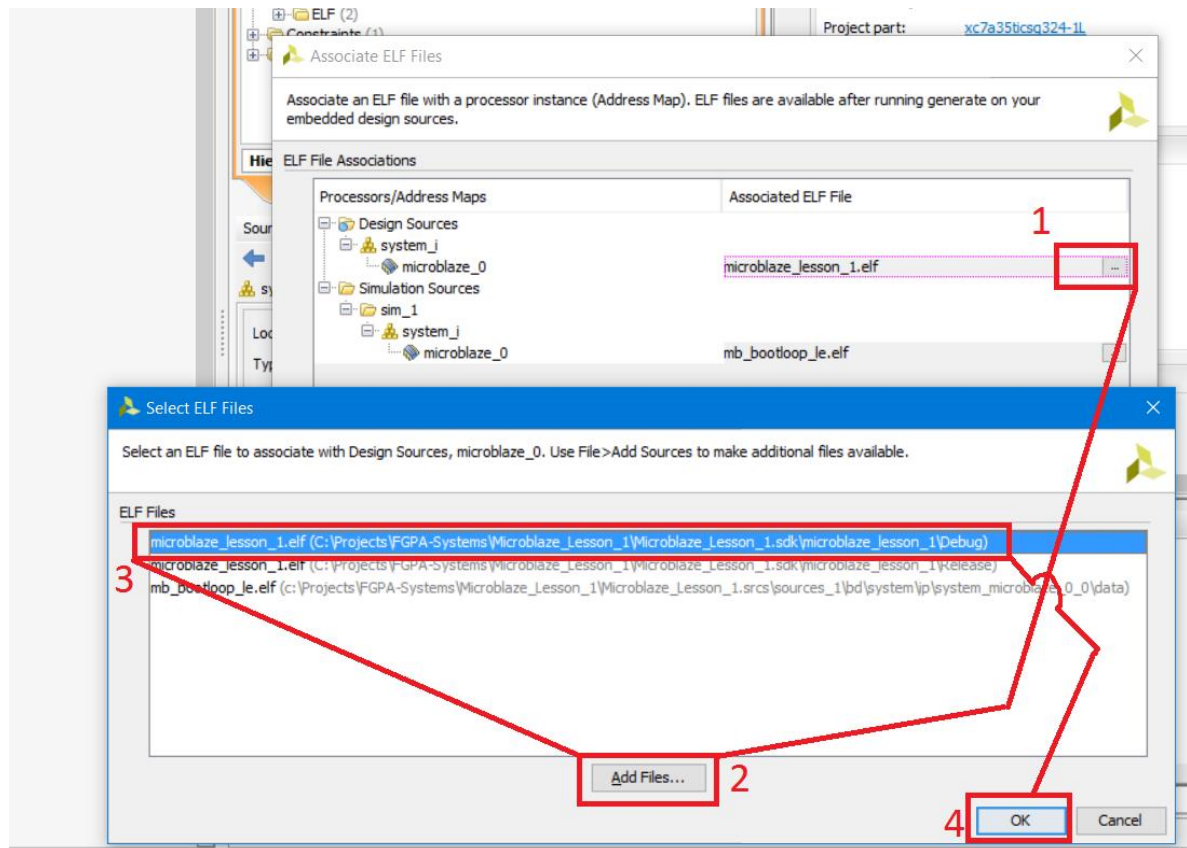


Рисунок 22 Окно добавления файла исполняемой программы elf

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).

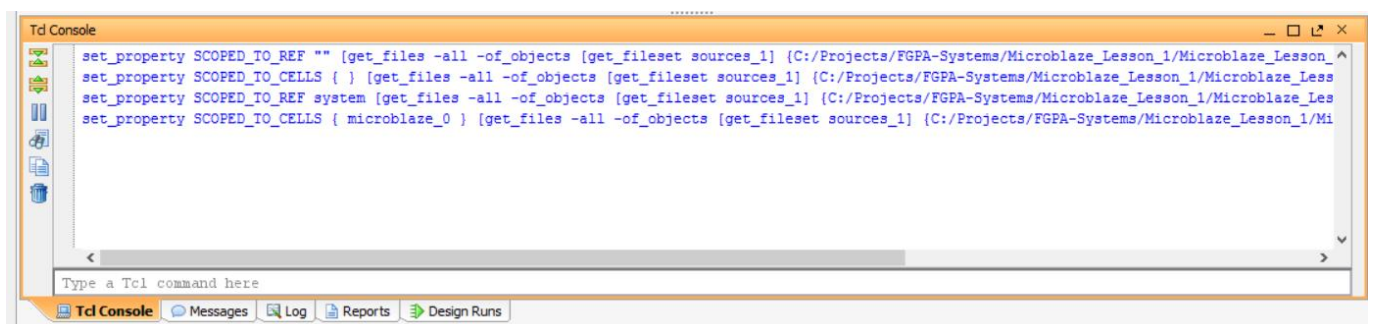


Рисунок 23 Команды для выполнения ассоциации elf и процессора. Сохраните их в текстовом файле. Они пригодятся позднее.

После ассоциирования, нужно регенерировать bit файл. Нажмите кнопку Generate Bitstream и дождитесь окончания операции. После нажатия на кнопку

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

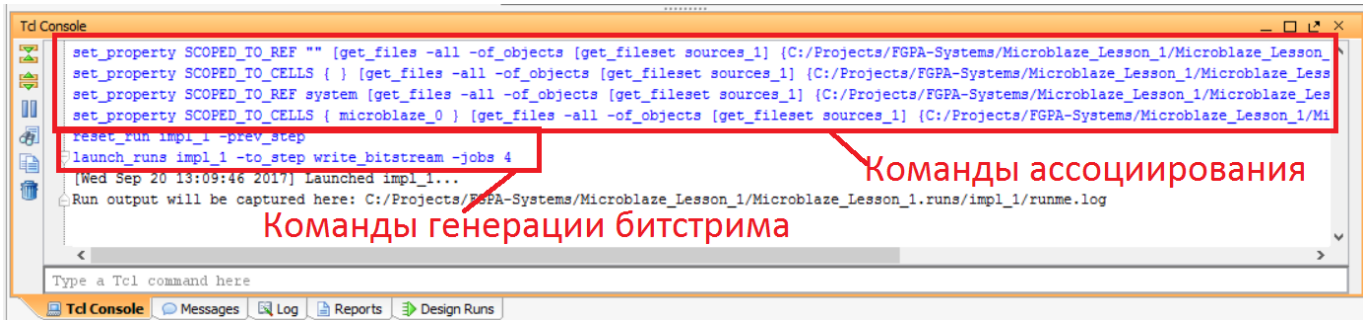


Рисунок 24 Команды ассоциирования и генерации битстрима в Tcl консоли. Добавьте команды генерации в текстовый файл ниже команд ассоциирования

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)

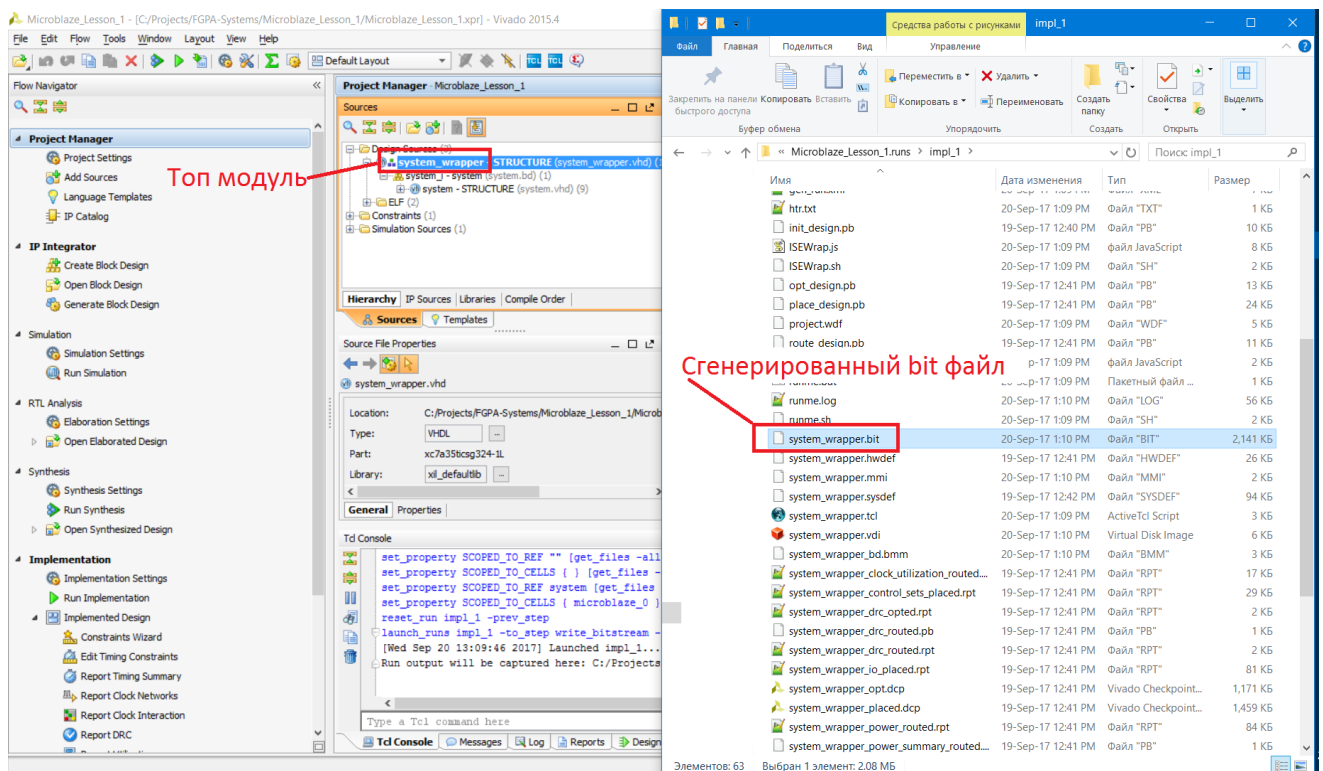


Рисунок 25 Топ модуль и сгенерированный bit файл

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/FPGA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.runs/impl_1/system_wrapper.bit"}\
-force\
-file {C:\Projects\FPGA-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 битсрима. 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 #ассоциирование
2 set_property SCOPED_TO_REF "" [get_files -all -of_objects [get_fileset sources_1] {C:/Projects/FPGA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.runs/impl_1/system_wrapper.bit}]
3 set_property SCOPED_TO_CELLS { } [get_files -all -of_objects [get_fileset sources_1] {C:/Projects/FPGA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.runs/impl_1/system_wrapper.bit}]
4 set_property SCOPED_TO_REF system [get_files -all -of_objects [get_fileset sources_1] {C:/Projects/FPGA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.runs/impl_1/system_wrapper.bit}]
5 set_property SCOPED_TO_CELLS { microblaze_0 } [get_files -all -of_objects [get_fileset sources_1] {C:/Projects/FPGA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.runs/impl_1/system_wrapper.bit}]
6
7 #Генерация Bitstream
8 reset_run impl_1 -prev_step
9 launch_runs impl_1 -to_step write_bitstream -jobs 4
10
11 #Создание mcs файла
12 write_cfgmem\
13 -format mcs\
14 -size 16\
15 -interface SPIx4 \
16 -loadbit {up 0x00000000 "C:/Projects/FPGA-Systems/Microblaze_Lesson_1/Microblaze_Lesson_1.runs/impl_1/system_wrapper.bit"}\
17 -force\
18 -file {C:\Projects\FPGA-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 Artu 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 нечего. Но, вот тут нас может выручить одна полезная фишка 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 битсима 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_Lesson_1}
4 set_property SCOPED_TO_REF system [get_files -all -of_objects [get_fileset sources_1] {C:/Projects/FGPA-Systems/Microblaze_Lesson_1}
5 set_property SCOPED_TO_CELLS { microblaze_0 } [get_files -all -of_objects [get_fileset sources_1] {C:/Projects/FGPA-Systems/Microblaze_Lesson_1}
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 -file {C:\Projects\FGPA-Systems\Microblaze_Lesson_1\Microblaze_Lesson_1.runs\impl_1\bitelf_mcs.mcs}

```

Рисунок 28 Окончательный листинг файла mem_gen.tcl

Открываем Vivado, нажимаем Tools→Customize Commands→ Customize Commands.

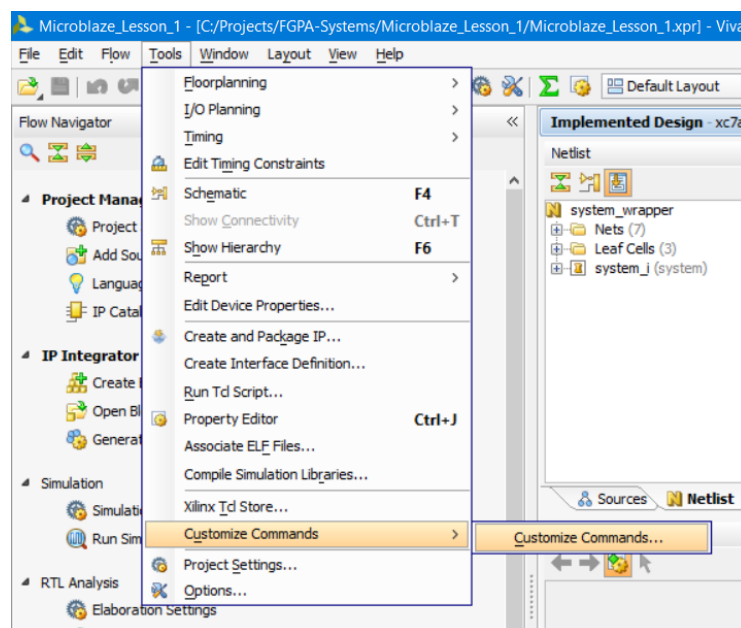


Рисунок 29 Листинг команд для генерирования склеенного bit+elf и mcs файлов

Откроется окно настройки кнопки, нажмите на плюсики, чтобы добавить кнопку, название кнопки «mem_gen», в качестве источника укажите файл

mem_gen.tcl, при необходимости можете добавить иконку и сделать описание (рис.30).

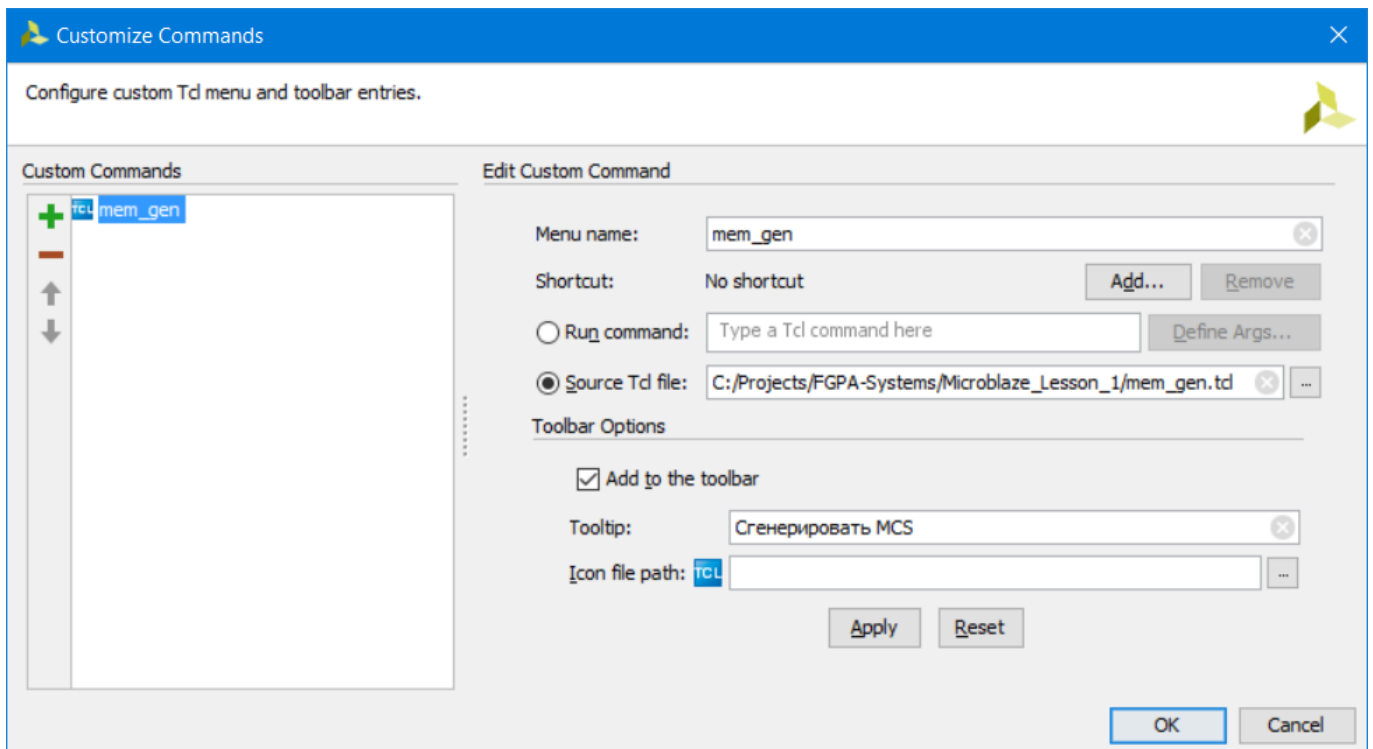


Рисунок 30 Меню создания кнопки в Vivado

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

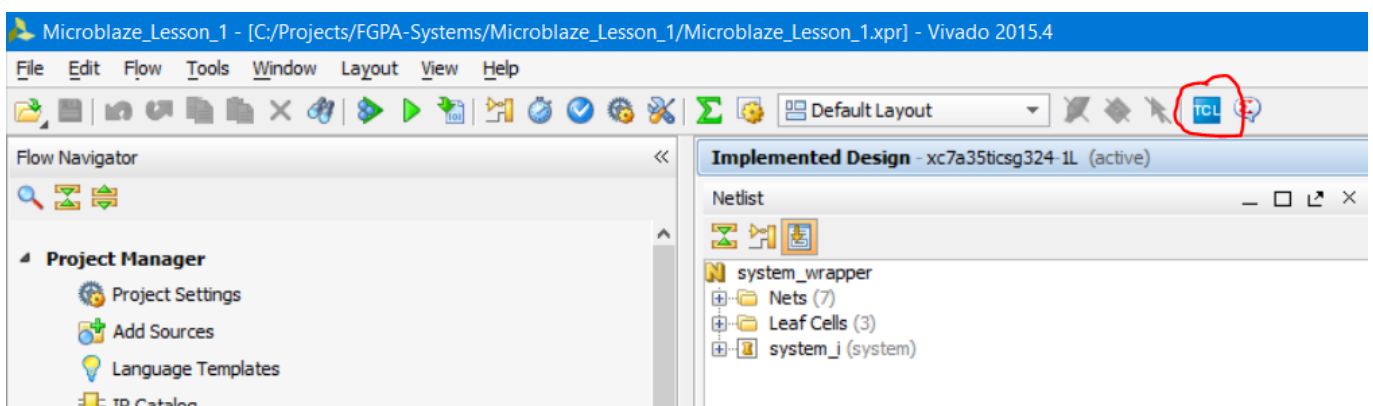


Рисунок 31 Кнопка пользователя в панели Vivado

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

Не забывайте обновлять mcs 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 [Aspect](#) 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. [UG894](#) Using Tcl Scripting
4. [UG898](#) Embedded Processor Hardware Design
5. [UG984](#). MicroBlaze Processor Reference Guide. Xilinx Inc.
6. [MicroBlaze на сайте Xilinx](#)
7. [Vivado на сайте Xilinx](#)
8. [Описание](#) 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](#)