

## Реализация примера расширенного копировщика загрузки

В этой секции описаны шаги, необходимые для создания и запуска примера расширенного копировщика загрузки на примере проекта Nios II Ethernet Standard.

### Установки программных инструментов и платы разработки

Для создания и запуска примера расширенного копировщика загрузки, вы должны выполнить следующие пункты:

1. Убедиться, что на компьютере установлены Nios II EDS версии 10.0 (или старше) и программа Quartus II версии 10.0 (или старше).
2. Питание и кабель USB-Blaster™ подключены к плате разработки Altera.

### Создание соответствующего аппаратного проекта

В следующих пунктах вы открываете, модифицируете и генерируете систему Nios II, на которой вы собираетесь запускать пример расширенного копировщика загрузки. Также вы должны решить, какой метод загрузки вы хотите реализовать. В нескольких следующих пунктах вам потребуется слегка изменить действия, в зависимости от используемого метода.

Для открытия примера проекта:

1. Проверьте, что вы загрузили и разархивировали файлы, описанные в секции "Файлы аппаратного проекта" на стр. 2, в вашу директорию *<project>*.
2. В программе Quartus II кликните на меню **File > Open Project** и откройте файл проекта *<project>\niosii\_ethernet\_standard\_<board>.qpf*. Если вы планируете загружаться прямо из CFI флеш, пример стандартного проекта будет работать без дополнительной памяти, вы можете пропустить секцию "Сборка расширенного копировщика загрузки" на стр. 12.

Для добавления внутри чиповой загрузочной ROM в систему:

1. Кликните **Tools > SOPC Builder** для запуска SOPC Builder.
2. В SOPC Builder на вкладке **System Contents** раскройте **Memories and Memory Controllers**, раскройте **On-Chip** и выберите **On-Chip Memory (RAM or ROM)**.

3. Кликните **Add** для добавления компонента в систему. Используйте следующие настройки для спецификации памяти:

- **Memory Type: RAM (Writable)** (не **ROM (Read-only)**)
- **Single-port access**
- **Data width: 32 бита**
- **Total memory size: 8 kBytes**

Заданный размер периферии означает, что вы резервируете место под код образа для следующих версий примера копировщика загрузки. Этот образ содержит следующий код:

- Код сброса в секции `.entry`
- Код запуска `crt0.s`
- Секция `.text`, содержащая точку входа `alt_main`
- Секция `.rodata`, хранящая любые доступные только для чтения данные инициализации
- Секция `.rdata` хранящая данные инициализации для чтения и записи
- Секция `.bss`, хранящая переменные инициализации и статические
- Обработчик исключений, размещённый в секции `.exception`

Некоторые эти секции скопированы в RAM исключений – RAM, которая содержит вектор исключений, - при исполнении кода запуска `crt0.s`, но все эти секции изначально хранятся в этой внутри чиповой памяти.

4. В матрице соединений SOPC Builder проследите за тем, чтобы слейв порт внутри чиповой памяти был подключен к мастеру инструкций Nios II и к мастеру данных Nios II.
5. Если в нижнем окне SOPC Builder рапортует об ошибке, вызванной перекрытием адресов новой внутри чиповой памяти с другой периферией, выберите соответствующий базовый адрес для внутри чиповой памяти, чтобы ни с чем не перекрываться.
6. Модифицируйте тактовый вход для новой внутри чиповой памяти, чтобы сделать её тактируемой одинаковым тактом с компонентом **cpu**.
7. Правым кликом на компонент **On-Chip Memory** выберите **Rename**. Переименуйте компонент на описательное имя `boot_rom`.
8. Для разрешения запуска копировщика загрузки из внутри чиповой памяти, правым кликом на компонент **cpu** вашей системы, выберите **Edit**.
9. В окне настроек процессора Nios II установите **Reset Vector Memory** на **boot\_rom** с **Offset** `0x00000000`.
10. Кликните **Finish** для выхода из окна настроек Nios II.
11. Кликните **Generate** для генерирования системы SOPC Builder.

### Сборка расширенного копировщика загрузки

Для сборки примера расширенного копировщика загрузки в новой директории проекта Quartus II, выполните следующие пункты:

1. Проверьте, что вы загрузили и разархивировали файлы, описанные в секции "Файлы аппаратного проекта" на стр. 2, в вашу директорию `<project>`.

2. Откройте файл `<project>/boot_copier_sw/bsp/advanced_boot_copier_bsp/bootcopier_bsp_settings.tcl` в текстовом редакторе.
3. Чтобы гарантировать точные настройки bsp и размещения компилятора в памяти, отредактируйте следующие настройки в `bootcopier_bsp_settings.tcl`:
  - a. Установите Set ONCHIP в 1, если вы загружаетесь из внутри чиповой памяти и 0 в обратном случае.
  - b. Отредактируйте EXCEPTION\_OFFSET и RESET\_OFFSET, чтобы сделать их значения равными значениям компонента CPU в SOPC Builder.
  - c. Обновите SDRAM\_SIZE и FLASH\_SIZE, чтобы сделать их размер равным размеру памяти в SOPC Builder.
4. Откройте файл `<project>/boot_copier_sw/app/advanced_boot_copier/advanced_boot_copier.c` в текстовом редакторе.
5. Отредактируйте строку:

```
#define BOOT_METHOD <boot_method>
```

Чтобы она отображала используемый вами метод. Следующие опции доступны для `<boot_method>`:

- BOOT\_FROM\_CFI\_FLASH
- BOOT\_CFI\_FROM\_ONCHIP\_ROM
- BOOT\_EPCS\_FROM\_ONCHIP\_ROM

Директива `#define` указывает компилятору собирать копировщик загрузки в соответствии с выбранным вами методом.

6. Чтобы запретить приложению выводить сообщения во время загрузки в JTAG UART, отредактируйте строку:

```
#define USING_JTAG_UART 1
```

на:

```
#define USING_JTAG_UART 0
```

Директива `#define` указывает компилятору собирать копировщик загрузки совсем без кода JTAG UART.

7. Откройте командную строку Nios II. (В Windows кликните **Пуск > Все программы > Altera > Nios II EDS > Nios II Command Shell**).
8. Смените директорию на `<project>/boot_copier_sw/app/advanced_boot_copier`.
9. Для создания и сборки BSP и проекта приложения, введите следующую команду:

```
./create-this-app ↵
```

Сейчас вы создали исполняемый файл копировщика загрузки, который готов к запуску на процессоре Nios II. Далее вы должны создать приложение для загрузки, используя новый копировщик загрузки.

### Сборка тестового приложения для загрузки

В этой секции вы соберёте тестовое приложение для загрузки. Для сборки тестового приложения для загрузки, используя расширенный копировщик загрузки, выполните следующие пункты:

1. Откройте командную строку Nios II. (В Windows кликните **Пуск > Все программы > Altera > Nios II EDS > Nios II Command Shell**).

2. Смените на директорию `<project>/boot_copier_sw/app/hello_world`.
3. Для создания и сборки тестового приложения BSP и проекта приложения, а также для генерирования исполняемого файла **hello\_world.elf**, введите следующую команду:

```
./create-this-app ↵
```

### Упаковка тестового приложения в запись загрузки

В этой секции вы упакуете тестовое приложение для загрузки в запись загрузки, которую понимает копировщик загрузки. Для упаковки приложения, вы запускаете скрипт в командной строке Nios II. Следующие скрипты находятся с файлами проекта:

- **make\_flash\_image\_script.sh**
- **make\_header.pl**
- **read\_flash\_image.pl**

Чтобы упаковать тестовое приложение для загрузки, используя расширенный копировщик загрузки, выполните следующие пункты:

1. Откройте в текстовом редакторе файл скрипта `<project>/boot_copier_sw/app/hello_world/make_flash_image_script.sh` и обновите параметры `flash_base` и `flash_end` на те, что вы сделали в системе.
2. В командной строке Nios II, запустите скрипт **make\_flash\_image\_script.sh**, чтобы упаковать файл **.elf** в записи загрузки, введя следующую команду:

```
./make_flash_image_script.sh hello_world.elf ↵
```

Запустив этот скрипт, вы можете получить предупреждения о пустом загрузочном сегменте и увидеть имя промежуточного файла **fake\_flash\_copier.srec**. Вы можете спокойно проигнорировать эти предупреждения.

Скрипт создаёт файлы **hello\_world.elf.flash.bin** и **hello\_world.elf.flash.srec** в текущей директории. Теперь у вас есть все бинарные образы, необходимые для загрузки тестового приложения с примером копировщика загрузки. Далее вы программируете эти образы в соответствующие области.

### Загрузка прямо из CFI флеш памяти

В этой секции вы используете флеш программатор Nios II для программирования копировщика загрузки и тестового приложения в CFI флеш память.

Если вы выбрали загрузку из внутри чиповой памяти, эта секция не применяется. Перейдите сразу на секцию "Загрузка CFI или EPCS флеш из внутри чиповой памяти" на стр. 15.

1. В программе Quartus II выберите Tools > **Programmer**.
2. Убедитесь, что файл `<project>\niosii_ethernet_standard_<board>.sof` находится в столбце **File**.
3. Убедитесь, что опция **Program/Configure** включена.
4. Кликните **Start** для конфигурирования вашего FPGA файлом **.sof**.

5. В командной строке Nios II смените директорию на:  
`<project>/boot_copier_sw/app/hello_world`
6. Установите офсет во флеш памяти, по которому находится образ загрузки `hello_world`, введя следующую команду:

```
bin2flash --input=hello_world.elf.flash.bin \  
--output=hello_world.flash \  
--location=0x00240000 ↵
```

Установите офсет на 0x00240000 или 0x00440000, поскольку режим загрузки из CFI флеш подразумевает возможность двойного размещения, когда копировщик загрузки выбирает образ загрузки 1 и 2 соответственно. Два этих адреса работают одинаково.

Вы можете также изменить размещение по умолчанию, редактируя оператор `#define` для `BOOT_IMAGE_1_OFFSET` и `BOOT_IMAGE_2_OFFSET` в файле `<project>/boot_copier_sw/app/advanced_boot_copier/advanced_boot_copier.c`, а затем заново собрать копировщик загрузки.

7. Запрограммируйте загрузочный образ `hello_world` во флеш память, введя следующую команду:

```
nios2-flash-programmer --base=<flash_base> \  
hello_world.flash ↵
```

где `<flash_base>` - это базовый адрес компонента CFI флеш в вашей системе SOPC Builder.

8. В командной строке Nios II смените директорию `<project>/boot_copier_sw/app/advanced_boot_copier`.
9. Создайте файл флеш памяти для копировщика загрузки, введя следующую команду:

```
make flash ↵
```

Эта команда создаёт файл `<flash_component>.flash`, где `<flash_component>` - это имя CFI флеш компонента в вашей системе SOPC Builder.

10. Запрограммируйте копировщик загрузки во флеш, введя следующую команду:

```
nios2-flash-programmer --base=<flash_base> \  
<flash_component>.flash ↵
```

11. Продолжите выполнение с секции "Запуск примера расширенного копировщика загрузки" на стр. 17.

### Загрузка CFI или EPCS флеш из внутри чиповой памяти

В этой секции вы используете программу Quartus II для программирования копировщика загрузки в память FPGA `boot_rom`, и затем используете флеш программатор Nios II для программирования записи загрузки тестового приложения в CFI или EPCS флеш память.

Если вы выбрали загрузку прямо из CFI флеш памяти, эта секция не применяется. Перейдите сразу на секцию "Загрузка прямо из CFI флеш памяти" на стр. 14.

Для программирования копировщика загрузки в память FPGA `boot_rom`, выполните следующие пункты:

1. В командной строке Nios II смените поддиректорию вашего проекта Quartus II `<project>/boot_copier_sw/app/advanced_boot_copier`.
2. Для генерирования файла инициализации для памяти **boot\_rom**, содержащей копировщик загрузки, введите следующую команду:

```
elf2hex advanced_boot_copier.elf <boot_rom_start_address>
<boot_rom_end_address> --width=32 --create-lanes=0
../../../../boot_rom.hex ↵
```

3. Если SOPC Builder ещё открыт, войдите в него и крикните **Exit** для его закрытия.
4. В программе Quartus II кликните **Processing > Start Compilation** для компиляции проекта.
5. Когда компиляция закончится, кликните **Tools > Programmer**.
6. Убедитесь, что файл `<project>\niosii_ethernet_standard_<board>.sof` находится в столбце **File**.
7. Убедитесь, что опции **Program/Configure** включены.
8. Кликните **Start** для конфигурирования вашего FPGA .sof файлом.

Память **boot\_rom** в FPGA теперь содержит исполняемый образ примера копировщика загрузки.

Для программирования тестового приложения во флеш память, выполните следующие пункты:

9. В командной строке Nios II смените поддиректорию вашего проекта Quartus II `<project>/boot_copier_sw/app/hello_world`.
10. Установите офсет во флеш памяти, где будет расположен образ загрузки `hello_world`, введя следующие команды:
  - Если вы загружаетесь из CFI флеш памяти, введите следующую команду:

```
bin2flash --input=hello_world.elf.flash.bin \
--output=hello_world.flash \
--location=0x00240000 ↵
```

- Если вы загружаетесь из EPCS чипа, введите следующую команду:

```
bin2flash --input=hello_world.elf.flash.bin \
--output=hello_world.flash \
--location=0x00060000 ↵
```

Установите офсет 0x00240000 или 0x00440000, если загружаетесь из CFI флеш памяти, и 0x00060000 или 0x00080000, если из чипа EPCS, поскольку режим загрузки из соответствующей флеш памяти подразумевает возможность двойного размещения, когда копировщик загрузки выбирает образ загрузки 1 и 2 соответственно. В большинстве случаев, два этих адреса работают одинаково.

Вы можете также изменить размещение по умолчанию, редактируя оператор `#define` для `BOOT_IMAGE_1_OFFSET` и `BOOT_IMAGE_2_OFFSET` в файле `<project>/boot_copier_sw/app/advanced_boot_copier/advanced_boot_copier.c`, а затем заново собрать копировщик загрузки.

Если вы редактируете офсеты флеш образов в файле `advanced_boot_copier.c`, задайте значение `-location` одним из определённых вами офсетов образов, а не используйте приведённые здесь значения офсетов.

11. Запрограммируйте образ загрузки `hello_world` во флеш память, введя одну из следующих команд:



- Если вы загружаетесь из CFI флеш памяти, введите следующую команду:

```
nios2-flash-programmer --base=<flash_base> \  
hello_world.flash ↵
```

- Если вы загружаетесь из EPCS чипа, введите следующую команду:

```
nios2-flash-programmer --base=<flash_base> \  
--epcs hello_world.flash ↵
```

где <flash\_base> - это базовый адрес CFI или EPCS флеш компонента в вашей системе SOPC Builder.

### Запуск примера расширенного копировщика загрузки

Для запуска примера расширенного копировщика загрузки в вашей плате разработчика, выполните следующие пункты:

- После завершения работы флеш программатора, в командной строке Nios II введите следующую команду для сброса процессора Nios II:

```
nios2-download -r -g ↵
```

Загрузчик и тестовое приложение выведут статусные сообщения в JTAG UART, если он разрешён. Если JTAG UART и SYS\_CLK\_TIMER инициализируются в файле **bsp/alt\_sys\_init.c**, а USING\_JTAG\_UART сохранило значение 1 в файле **<project>/boot\_copier\_sw/app/advanced\_boot\_copier/advanced\_boot\_copier.c**, вы сможете увидеть эти сообщения.

Для просмотра сообщений, выполните следующие пункты:

- В командной строке Nios II запустите утилиту **nios2-terminal**, введя следующую команду:

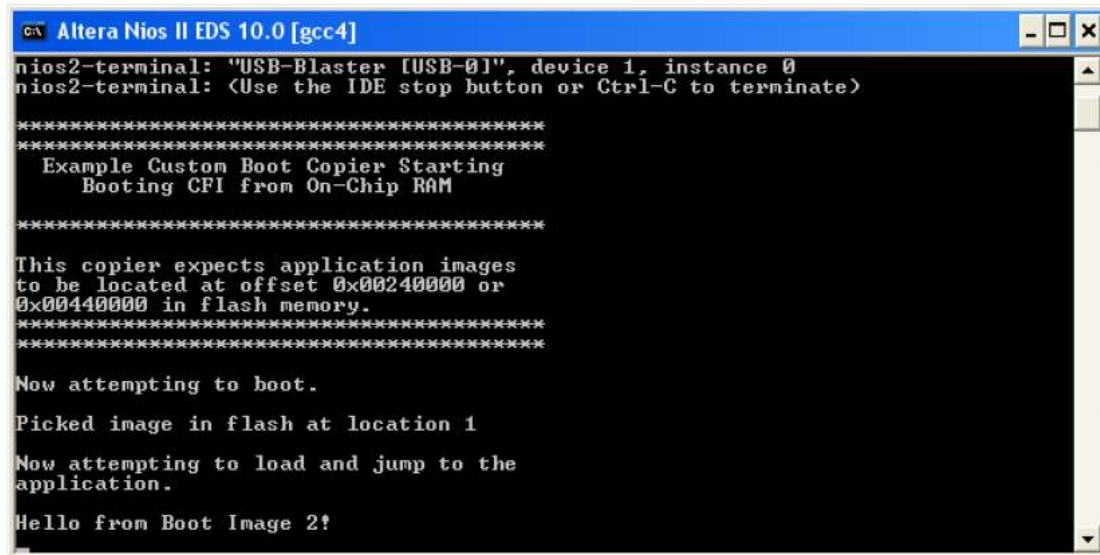
```
nios2-terminal ↵
```

Если **nios2-terminal** не может подключиться к JTAG UART с помощью настроек по умолчанию, запустите его с опцией --help, чтобы посмотреть нужные переключения в командной строке.

Если ваш **nios2-terminal** отображает неполный результат копировщика загрузки, следующий на выходе образа загрузки, нажмите кнопку сброса CPU на вашей плате разработчика, чтобы повторить процесс загрузки и увидеть полный результат.

Если копировщик загрузки запущен успешно, вы увидите результат в **nios2-terminal**, как показано на рис. 3. Ваш результат может незначительно отличаться, когда вы загружаетесь из внешней памяти или из EPCS флеш.

**Figure 3. Advanced Boot Copier Output**



```
Altera Nios II EDS 10.0 [gcc4]
nios2-terminal: "USB-Blaster [USB-0]", device 1, instance 0
nios2-terminal: <Use the IDE stop button or Ctrl-C to terminate>

*****
Example Custom Boot Copier Starting
Booting CFI from On-Chip RAM
*****

This copier expects application images
to be located at offset 0x00240000 or
0x00440000 in flash memory.
*****

Now attempting to boot.

Picked image in flash at location 1

Now attempting to load and jump to the
application.

Hello from Boot Image 2!
```