Инструменты командной строки Altera для разработки устройств

В этой секции описываются инструменты командной строки для разработки аппаратного проекта. Они подходят для всех проектов, созданных в SOPC Builder с включением или без включения процессора Nios II.

quartus_cmd и sopc_builder

Эти команды создают скрипты, автоматизирующие генерирование систем SOPC Builder и компиляцию соответствующих проектов Quartus II.

Вы можете использовать эти команды для создания процесса, задействующего минимальное количество исходных файлов, требуемых для сборки вашего проекта Quartus II. Если вы копируете существующий проект, чтобы использовать его в качестве базы для разработки нового проекта, вы можете скопировать только этот минимальный набор исходных файлов. Проще говоря, когда вы контролируете версию системы в файлах, вы хотите контролировать её в минимальном наборе, необходимом для реконструкции проекта.

Для реконструкции системы SOPC Builder необходимы следующие файлы:

- **<**project**>.qpf** (файл проекта Quartus II);
- **<**project**>.qsf** (фай настроек Quartus II);
- **<**SOPC Builder system**>.sopc** (файл описания системы SOPC Builder);
- дополнительные HDL, BDF или BSF файлы имеющегося проекта.

Если вы работаете с примерами аппаратных проектов, которые предлагаются к инсталляции Quartus II, Altera рекомендует вам скопировать каждый набор исходных файлов в рабочую директорию, чтобы избежать неумышленных изменений в оригинальных исходных файлах. Запускайте скрипт в новой рабочей директории.

Для создания процесса, обслуживающего минимум исходных файлов, выполните следующие пункты:

1. Скопируйте необходимые исходные файлы в рабочую директорию, проследите за корректностью каждого копируемого файла.
2. Смените рабочую директорию на эту.
3. Для генерирования **.sof** файла для конфигурирования FPGA, введите следующую последовательность команд:

```
sopc_builder --no_splash -s --generate ↵
quartus_cmd <project>.qpf -c <project>.qsf ↵
```

Оболочка скрипта в примере 4-2 отображает эти команды. Этот скрипт автоматизирует процесс генерирования систем SOPC Builder и компиляцию проектов Quartus II через любое количество поддиректорий. Этот скрипт приведён в качестве примера, он должен быть изменён под ваш проект. Если вы хотите скомпилировать проекты Quartus II установите переменную COMPILE_QUARTUS в скрипте в 1.

Example 4–2. Script to Generate SOPC Builder System and Compile Quartus II Projects (Part 1 of 2)

```
#!/bin/sh
COMPILE_QUARTUS=0
#
# Resolve TOP_LEVEL_DIR, default to PWD if no path provided.
#
if [ $# -eq 0 ]; then
   TOP_LEVEL_DIR=$PWD
else
   TOP_LEVEL_DIR=$1
fi
echo "TOP_LEVEL_DIR is $TOP_LEVEL_DIR"
echo
#
# Generate SOPC list...
#
SOPC_LIST=`find $TOP_LEVEL_DIR -name "*.sopc"`
#
# Generate Quartus II project list.
#
PROJ_LIST=`find $TOP_LEVEL_DIR -name "*.qpf" | sed s/\.qpf//g`
#
# Main body of the script.  First "generate" all of the SOPC Builder
# systems that are found, then compile the Quartus II projects.
#
#
# Run SOPC Builder to "generate" all of the systems that were found.
#
for SOPC_FN in $SOPC_LIST
do
   cd `dirname $SOPC_FN`
   if [ ! -e `basename $SOPC_FN .sopc`.vhd -a ! -e `basename $SOPC_FN .sopc`.v ]; then
      echo; echo
      echo "INFO:  Generating $SOPC_FN SOPC Builder system."
      sopc_builder -s --generate=1 --no_splash
      if [ $? -ne 4 ]; then
         echo; echo
         echo "ERROR:  SOPC Builder generation for $SOPC_FN has failed!!!"
         echo "ERROR:  Please check the SOPC file and data " \
                   "in the directory `dirname $SOPC_FN` for errors."
      fi
   else
      echo; echo
      echo "INFO:  HDL already exists for $SOPC_FN, skipping Generation!!!"
   fi
   cd $TOP_LEVEL_DIR
done
#
# Continued...
#
```

Example 4–2. Script to Generate SOPC Builder System and Compile Quartus II Projects (Part 2 of 2)

```
#
# Now, generate all of the Quartus II projects that were found.
#
if [ $COMPILE_QUARTUS ]; then
    for PROJ in $PROJ_LIST
    do
        cd `dirname $PROJ`
        if [ ! -e `basename $PROJ`.sof ]; then
            echo; echo
            echo "INFO:  Compiling $PROJ Quartus II Project."
            quartus_cmd `basename $PROJ`.qpf -c `basename $PROJ`.qsf
            if [ $? -ne 4]; then
                echo; echo
                echo "ERROR:  Quartus II compilation for $PROJ has failed!!!."
                echo "ERROR:  Please check the Quartus II project " \
                             "in `dirname $PROJ` for details."
            fi
        else
            echo; echo
            echo "INFO:  SOF already exists for $PROJ, skipping compilation."
        fi
        cd $TOP_LEVEL_DIR
    done
fi
```

Команды и скрипты из примера 4-2 приводятся только в целях ознакомления. Altera не гарантирует их работу в вашем конкретном случае.

Инструменты командной строки Altera для программирования флеш памяти

В этой секции описываются инструменты командной строки для программирования вашего проекта Nios II во флеш память.

Когда вы используете Nios II IDE для программирования флеш памяти, Nios II IDE генерирует скрипт, состоящий из команд конвертирования и команд программирования. Вы можете использовать этот скрипт в качестве основы для разработки собственного процесса программирования флеш памяти.

За подробной информацией об Nios II IDE, и использовании командной строки флеш программатора Nios II и сопутствующих инструментах, обратитесь к руководству пользователя флеш программатором Nios II.

nios2-flash-programmer

Эта команда программирует CFI флеш память. Поскольку флеш программатор Nios II использует JTAG интерфейс, команда nios2-flash-programmer имеет те же самые опции, как и другие команды. Вы можете получить информацию об опциях командной строки для этой команды с помощью опции --help.

Пример использования nios2-flash-programmer

Вам необходимо выполнить следующие пункты для программирования CFI флеш памяти:

1. Следуйте пунктам в секции “nios2-download” на стр. 4–9, или используйте Nios II IDE для программирования вашего FPGA проектом, организующим интерфейс с вашим чипом CFI.

2. Введите следующую команду для проверки корректности детектирования вашего чипа:

nios2-flash-programmer –debug –base=<*base address*> ↵

где <*base address*> - это базовый адрес (офсет) вашего флеш чипа. Базовый адрес каждого компонента отображается в SOPC Builder. Если флеш устройство детектируется, отображается таблица содержимого CFI.

3. Сконвертируйте ваш файл в флеш формат (**.flash**), используя одну из утилит elf2flash, bin2flash или sof2flash, описанных в секции “elf2flash, bin2flash и sof2flash”.

4. Введите следующую команду для программирования результирующего **.flash** файла в чип CFI

nios2-flash-programmer –base=<*base address*> <*file*>.flash ↵

5. В дополнение введите следующую команду для сброса и запуска процессора по его адресу сброса:

nios2-download –g –r ↵

elf2flash, bin2flash и sof2flash

Три этих команды часто используются вместе с командой nios2-flash-programmer. Выходной **.flash** файл имеет стандарт **.srec** файла.

Следующие две важные опции командной строки доступны для команды elf2flash:

- Опция -boot=<*boot copier file*>.srec указывает команде elf2flash присоединить S-record файл загрузчика вначале конвертируемого ELF файла.
- Опция -after=<*flash file*>.flash помещает сгенерированный **.flash** файл - сконвертированный ELF файл - сразу после заданного **.flash** файла во флеш памяти.

Опция -after часто для того, чтобы разместить **.elf** файл сразу за **.sof** файлом в чипе EPCS.

Если вы используете чип EPCS, вы должны запрограммировать аппаратный образ в чип перед программным образом. Изменение этого порядка может повредить вашу программу.

Перед тем, как записать в любой флеш чип, флеш программатор Nios II стирает весь сектор, в который он будет записывать. Однако для чипов EPCS, если вы генерируете программный образ, используя опцию elf2flash -after, флеш программатор Nios II помещает программный образ сразу за аппаратным образом, а не на границе следующего сектора. Поэтому флеш программатор Nios II не стирает текущий сектор перед тем, как разместить в нём образ программы. Однако он стирает текущий сектор перед размещением аппаратного образа.

Когда вы используете флеш программатор из Nios II IDE, вы автоматически создаёте скрипт, состоящий из этих команд. Запуск флеш программатора создаёт скрипт (**.sh**) в директории **Debug** или **Release** вашего проекта. Этот скрипт состоит из подробных шагов команд, используемых для программирования вашей флеш памяти.

В примере 4-3 показан простой авто генерируемый скрипт.

Example 4–3. Sample Auto-Generated Script:

```
#!/bin/sh
#
# This file was automatically generated by the Nios II IDE Flash Programmer.
#
# It will be overwritten when the flash programmer options change.
#

cd <full path to your project>/Debug

# Creating .flash file for the FPGA configuration
#"$SOPC_KIT_NIOS2/bin/sof2flash" --offset=0x400000 --input="full path to your SOF" \
                --output="<your design>.flash"

# Programming flash with the FPGA configuration
#"$SOPC_KIT_NIOS2/bin/nios2-flash-programmer" --base=0x00000000 --sidp=0x00810828 \
                --id=1436046714 --timestamp=1169569475 --instance=0 "<your design>.flash"
#
# Creating .flash file for the project
"$SOPC_KIT_NIOS2/bin/elf2flash" --base=0x00000000 --end=0x7fffff --reset=0x0 \
                --input="<your project name>.elf" --output="ext_flash.flash" \
                --boot="<path to the bootloader>/boot_loader_cfi.srec"

# Programming flash with the project
"$SOPC_KIT_NIOS2/bin/nios2-flash-programmer" --base=0x00000000 --sidp=0x00810828 \
                --id=1436046714 --timestamp=1169569475 --instance=0 "ext_flash.flash"

# Creating .flash file for the read only zip file system
"$SOPC_KIT_NIOS2/bin/bin2flash" --base=0x00000000 --location=0x100000\
                --input="<full path to your binary file>" --output="<filename>.flash"

# Programming flash with the read only zip file system
"$SOPC_KIT_NIOS2/bin/nios2-flash-programmer" --base=0x00000000 --sidp=0x00810828 \
                --id=1436046714 --timestamp=1169569475 --instance=0 "<filename>.flash"
```

Пути, имена файлов и адреса при модификации авто-генерируемого скрипта зависят от имён и расположения файлов, которые вы конвертируете при конфигурировании вашего аппаратного проекта.

Пример использования bin2flash

Для программирования произвольного бинарного файла во флеш память, выполните следующие пункты:

1. Введите следующую команду для генерирования вашего **.flash** файла:

   bin2flash --location=<*offset from the base address*> \
   -input=<*your file*> --output=<*your file*>.flash ↵

2. Введите следующую команду для программирования только что созданного файла во флеш память:

   nios2-flash-programmer -base=<*base address*> <*your file*>.flash ↵