SiFive-OE-poPingUI-P4

Для экспериментов с применением языка P4 (p4.org) в сетевых устройствах была предпринята попытка сборки прототипа компилятора P4C на платформе HiFive Unleashed U540 компании SiFive. В качестве среды разработки использовалась система OpenEmbedded (www.yoctoproject.org) и базовый репозиторий SiFive (github.com/sifive/meta-sifive/tree/master). Для работы также требуется пакет repo, установка которого описана по ссылке.

Для сборки образов потребуется достаточно большое пространство на диске (после сборки образа на диске было занято 94 Гбайт), а сам диск должен быть быстрым (SSD или SAS).

1. Создаём пустой каталог и переходим в него

   ```
   mkdir sifive-master
   cd sifive-master
   ```
   Далее в тексте этот каталог будет именоваться корневым и все ссылки на каталоги и файлы будут отсчитываться от него.

2. Создаём и заполняем репозиторий

   ```
   $ repo init -u git://github.com/sifive/meta-sifive -b master -m tools/manifests/sifive.xml
   $ repo sync
   ```
3. Создаём уровень meta-poPingUI

   ```
   $ bitbake-layers create-layer meta-poPingUI
   NOTE: Starting bitbake server...
   Add your new layer with 'bitbake-layers add-layer meta-poPingUI'
   $ bitbake-layers add-layer meta-poPingUI
   NOTE: Starting bitbake server...
   Unable to find bblayers.conf
   ```
4. Копируем файл установки окружения из каталога meta-sifive

   ```
   $ cp meta-sifive/setup.sh meta-poPingUI/setup.sh
   ```
5. Редактируем созданную копию, добавляя строку

   ```
   $ bitbake-layers add-layer ../meta-poPingUI
   ```
6. Из строки DISTRO_FEATURES_append = " largefile opengl ptest multiarch pam systemd vulkan " удаляем opengl и vulkan и получаем в результате

   ```
   DISTRO_FEATURES_append = " largefile ptest multiarch pam systemd "
   ```
7. В начале файла меняем имя создаваемого образа, как показано ниже

   ```
   BITBAKEIMAGE="coreip-cli"
   ```
8. В конце файла помещаем символ комментария в показанные ниже строки

   ```
   # Add r600 drivers for AMD GPU
   #PACKAGECONFIG_append_pn-mesa = " r600"

   # Add support for modern AMD GPU (e.g. RX550 / POLARIS)
   #PACKAGECONFIG_append_pn-mesa = " radeonsi"
   #PACKAGECONFIG_append_pn-mesa = " gallium-llvm"
   ```
9. Копируем каталог meta-sifive/conf в meta-poPingUI/conf

10. Редактируем файл meta-poPingUI/conf/layer.conf, заменяя sifive на poPingUI, а также устанавливаем для уровня высокий приоритет, чтобы при совпадении имён задания выбирались из нашего уровня

    ```
    BBFILE_PRIORITY_meta-poPingUI = "9"
    ```
11. Копируем каталог meta-sifive/recipes-sifive/images в meta-poPingUI/recipes-poPingUI/images

    Поскольку планируется работа лишь с платой HiFive без модулей расширения удаляем файлы с поддержкой графического интерфейса (demo-coreip-xfce4.bb и demo-coreip-xfce4-debug.bb), а файлы demo-coreip-cli.bb и demo-coreip-cli-debug.bb переименовываем в coreip-cli.bb и coreip-cli-debug.bb. Это позволит избежать путаницы и при необходимости воспользоваться образами уровня meta-sifive.

12. Переходим в корневой каталог репозитория и вводим команду

    ```
    $ . ./meta-poPingUI/setup.sh
    Init OE
    You had no conf/local.conf file. This configuration file has therefore been
    created for you with some default values. You may wish to edit it to, for
    example, select a different MACHINE (target hardware). See conf/local.conf
    for more information as common configuration options are commented.

    You had no conf/bblayers.conf file. This configuration file has therefore been
    created for you with some default values. To add additional metadata layers
    into your configuration please add entries to conf/bblayers.conf.

    The Yocto Project has extensive documentation about OE including a reference
    manual which can be found at:
       http://yoctoproject.org/documentation

    For more information about OpenEmbedded see their website:
       http://www.openembedded.org/


    ### Shell environment set up for builds. ###

    You can now run 'bitbake <target>'

    Common targets are:
       core-image-minimal
       core-image-sato
       meta-toolchain
       meta-ide-support
    ```

```
You can also run generated qemu images with a command like 'runqemu qemux86'.

Other commonly useful commands are:
- 'devtool' and 'recipetool' handle common recipe tasks
- 'bitbake-layers' handles common layer tasks
- 'oe-pkgdata-util' handles common target package tasks
Adding layers
NOTE: Starting bitbake server...
NOTE: Starting bitbake server...
NOTE: Starting bitbake server...
NOTE: Starting bitbake server...
NOTE: Starting bitbake server...
NOTE: Starting bitbake server...
NOTE: Starting bitbake server...
NOTE: Starting bitbake server...
NOTE: Starting bitbake server...
Creating auto.conf
---------------------------------------------------
MACHINE=freedom-u540 bitbake coreip-cli
---------------------------------------------------

Buildable machine info
---------------------------------------------------
* freedom-u540: The SiFive HiFive Unleashed board
* qemuriscv64: The 64-bit RISC-V machine
---------------------------------------------------
```

13. Запускаем сборку образа командой

```
$ MACHINE=freedom-u540 bitbake coreip-cli
Parsing recipes: 100% |
##########################################################################################################################################################################| Time: 0:00:18
Parsing of 2346 .bb files complete (0 cached, 2346 parsed). 3445 targets, 174 skipped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION           = "1.46.0"
BUILD_SYS            = "x86_64-linux"
NATIVELSBSTRING      = "mageia-7"
TARGET_SYS           = "riscv64-oe-linux"
MACHINE              = "freedom-u540"
DISTRO               = "nodistro"
DISTRO_VERSION       = "nodistro.0"
TUNE_FEATURES        = "riscv64"
meta                 = "HEAD:57ccf1e3bb320bd28a2d106c98f4706434c3075a"
meta-oe
meta-python
meta-multimedia
meta-networking
meta-gnome
meta-xfce            = "HEAD:920161113533074d27dc93c521815380fdf20275"
meta-riscv           = "HEAD:8bd3402c76f897189842f65e521f1388777660ca"
meta-sifive          = "HEAD:4c97a625e70558fbca9dc210616b13115e22dbee"
meta-poPingUI        = "<unknown>:<unknown>"
```

14. Процесс загрузки исходных кодов[1] и сборки компонент достаточно долог и можно заняться другими делами. В моей системе Mageia 7.1 процесс завершается ошибкой

```
virtual:native:/OE/sifive-master/openembedded-core/meta/recipes-devtools/perl/libxml-parser-perl_2.46.bb:do_configure
```

Для решения этой проблемы создаем в каталоге уровня файл meta-poPingUI/recipes-devtools/libxml-parser-perl/libxml-parser-perl_2.46.bbappend, показанный ниже

```
do_configure_prepend_class-native () {
   cd ${S}
   ${HOSTTOOLS_DIR}/perl Makefile.PL
   cd -
}
```

В новой версии почему-то в Makefile указывалось полное имя компилятора x86_64-mageia-linux-gnu-gcc-10, который по непонятным причинам система сборке не находила. Помогла банальная замена на gcc

15. Проблема с networkmanager решается командой LC_ALL=C ../recipe-sysroot-native/usr/bin/intltool-merge -x -u -c ./po/.intltool-merge-cache ../NetworkManager-1.22.10/po data/org.freedesktop.NetworkManager.policy.in data/org.freedesktop.NetworkManager.policy из каталога build/tmp-glibc/work/riscv64-oe-linux/networkmanager/1.22.10-r0/build и создания символьной ссылки /usr/bin/nativeperl на локальный перл хоста[2].

16. В новой версии (3.2.1 gatesgarth) возникает непонятная проблема при сборке libxml2 - python не находит библиотеку libintl. Разбираться не стал, а просто поправил файл build/python/Makefile, убрав из строки опцию -lintl

17. На этом первичные правки завершаются и образ собирается нормально. Можно начинать внесение своих правок.

18. Сначала добавим Judy, поскольку от этого пакета зависит p4c и связанные пакеты.

```
$ devtool add https://github.com/multiSnow/judy.git
$ bitbake judy
```

Процесс завершается ошибкой /lib/ld-linux-riscv64-lp64d.so.1: No such file or directory. Это связано с попыткой запуска двоичного файла генерации таблиц, собранного для RISCV, из среды сборки x86. Найти способ обхода этой проблемы мне не удалось, поэтому были просто взяты файлы, созданные на платформе HiFive

[1]Время от времени при загрузке кода могут возникать ошибки, связанные с недоступностью сетевых ресурсов Internet. В таких случаях следует просто повторить команду.
[2]Попытка выполнить команду из файла дополнения даёт ошибку «You must have XML::Parser installed to run ../recipe-sysroot-native/usr/bin/intltool-merge». Добавление зависимостей проблему не решает.

Unleashed, и помещены в каталоги исходного кода judy. При этом запуск программы генерации (JudyLTablesGen и Judy1TablesGen) из Makefile был удален путем простого редактирования.

Копируем файл JudyLTables.c в каталог build/tmp-glibc/work/riscv64-oe-linux/judy/1.0.5+git999-r0/judy-1.0.5+git999/src/JudyL и удаляем конец строки 795[1] ./JudyLTablesGen. Затем повторяем команду bitbake judy и выполняем аналогичные процедуры в каталоге Judy1 с файлом Judy1Tables.c. Следующий запуск команды bitbake judy обеспечивает сборку пакета без ошибок.

19. Добавляем созданное задание на уровень meta-poPingUI командой

```
$ devtool finish -f judy meta-poPingUI/recipes-devtools
```

из корневого каталога системы сборки. В результате готовое задание помещается на нужный уровень в каталог recipes-devtools. После этого следует повторить сборку задания, поскольку при переносе внесенные изменения теряются (см. 14). Операции выполняются с файлами в каталоге build/tmp-glibc/work/riscv64-oe-linux/judy/1.0.5+gitAUTOINC+a5971f3ee4-r0/build/src.

20. Следующим добавляем пакет PI[2]

```
$ devtool add https://github.com/p4lang/PI.git
```

В файле задания pi_git.bb меняем строку DEPENDS, как показано ниже

```
DEPENDS = "readline judy nanomsg protobuf protobuf-c"
```

И добавляем опции настройки

```
EXTRA_OECONF = " --with-fe-cpp --with-proto=Protobuf --with-internal-rpc --with-cli "
```

Между пакетами PI и bmv2 имеются циклические зависимости, поэтому PI придётся собрать без поддержки bmv2 (без опции --with-bmv2). Включаем задание на уровень meta-poPingUI командами из корневого каталога

```
$ mkdir meta-poPingUI/recipes-p4
$ devtool finish pi meta-poPingUI/recipes-p4
$ bitbake pi
```

Система QA выдает ошибки, но в них нет ничего страшного и можно просто удалить файл build/tmp-glibc/work/riscv64-oe-linux/pi/0.1+gitAUTOINC+0fbdac2561-r0/temp/log.do_compile

21. В новой версии возникла ошибка с libxml2 (не находила библиотеку libintl при сборке). Добавил файл .bbappend с единственной строкой

```
DEPENDS = "intltool"
```

22. Затем добавляем пакет bmv2 (behavioral model version 2), содержащий прототип коммутатора P4.

```
$ devtool add https://github.com/p4lang/behavioral-model.git
```

Система сборки создает для него задание с именем bm. Редактируем в файле задания bm_git.bb приведённые ниже строки

```
DEPENDS = "gmp judy nanomsg libpcap pi boost"
EXTRA_OECONF = " --with-nanomsg --with-pi --enable-modules --disable-dependency-tracking --without-thrift "
```

Затем переносим задание на уровень meta-poPingUI командой из корневого каталога

```
$ devtool finish bm meta-poPingUI/recipes-p4
```

23. Начинаем сборку компилятора P4C

```
$ devtool add https://github.com/p4lang/p4c.git
```

Файл задания корректируем как показано ниже

```
DEPENDS = "flex-native bison-native boost protobuf protobuf-native protobuf-c protobuf-c-native doxygen bm bdwgc gmp judy"
EXTRA_OECMAKE = "-DENABLE_GC=OFF -DENABLE_EBPF=OFF -DENABLE_PROTOBUF_STATIC=OFF "
```

Сборка завершается ошибкой, связанной с запуском исполняемого файла RISCV (tools/irgenerator) в среде сборки x86. Для решения проблемы помещаем символ комментария в начало строки с вызовом команды irgenerator (строка 319[3]) в файле build.ninja каталога build/workspace/sources/p4c/oe-workdir/p4c-1.0+git999. Затем нужно скопировать в каталог build/workspace/sources/p4c/oe-workdir/p4c-1.0+git999/ir заранее подготовленные на целевой платформе файлы, перечисленные ниже

```
gen-tree-macro.h
gen-tree-macro.h.fixup
gen-tree-macro.h.tmp
ir-generated.cpp
ir-generated.cpp.fixup
ir-generated.cpp.tmp
ir-generated.h
ir-generated.h.fixup
ir-generated.h.tmp
libir.a
```

Копируем задание на уровень meta-poPingUI командой из корневого каталога

```
$ devtool finish bm meta-poPingUI/recipes-p4
```

После чего повторяем сборку и после получения ошибки внесим описанные выше правки в каталоге build/tmp-glibc/work/riscv64-oe-linux/p4c/1.0+gitAUTOINC+f79af56ea9-r0/build и еще раз собираем пакет командой

```
$ bitbake p4c
```

24. Сборка завершается без ошибок и можно внести задание в образ, редактируя файл meta-poPingUI/recipes-poPingUI/images/coreip-cli.bb. Для этого добавляем строку в конце переменной IMAGE_INSTALL, как показано ниже.

[1]Номер строки может отличаться, поэтому лучше воспользоваться контекстным поиском в файле.
[2]При очередной сборке образа вдруг пошли сообщения QA об использовании при компиляции каталога хоста /usr/include. Исправить не сумел. Решил проблему удалением содержимого каталога temp в дереве пакета после компиляции задания.
[3]Можно найти строку по контексту Generating IR class files. Искомая строка будет предыдущей.

```
    p4c \
    ${CORE_IMAGE_EXTRA_INSTALL} \
    "

IMAGE_INSTALL_append_freedom-u540 = "\
    unleashed-udev-rules \
    "
```

25. Далее создаем образ командой

    ```
    $ MACHINE=freedom-u540 bitbake coreip-cli
    ```

    и по завершении его создания переносим на SD-карту командой

    ```
    $ zcat build/tmp-glibc/deploy/images/freedom-u540/coreip-cli-freedom-u540.wic.gz | sudo dd of=/dev/sdX bs=512K iflag=fullblock oflag=direct conv=fsync status=progress
    ```

    где sdX - имя устройства (SD-карта).

26. По завершении записи можно отмонтировать SD-карту и перенести ее на плату HiFive Unleased. Для входа в систему служит имя пользователя root с паролем sifive.

В следующих статьях будет описана оптимизация образа и добавление функций трассировки, тестирования и мониторинга сетевых функций.

Работа выполнена в рамках проекта «Орион».

Николай Малых

nmalykh@protokols.ru