ОС Linux для систем на кристаллах ПЛИС фирмы Xilinx

(часть 1)

Алексей Шматок (Москва)

В статье обсуждаются особенности разработки и реализации системного и прикладного программного обеспечения для систем на кристаллах ПЛИС, работающих под управлением ОС Linux.

Системы на кристалле (СнК, SoC) ПЛИС получили широкое распространение и используются для решения многих задач. Во всём их многообразии можно выделить СнК, в которых присутствуют микропроцессорные ядра и типовые периферийные устройства: внешняя память, сетевой адаптер, последовательный порт и т.п. Для таких систем в качестве базовой платформы для построения приложений как нельзя лучше подходит операционная система (ОС) с открытым кодом – Linux [1].

Цель данной статьи состоит в том, чтобы поделиться опытом разработки СнК на платах RMB-411 [2] и ML401 [3], дать обзор необходимых инструментальных средств разработки СнК ПЛИС Xilinx, генерации BSP, кросс-компиляции под микропроцессорную архитектуру PPC405/MicroBlaze, сборки ядра Linux, загрузчика U-boot и RootFS, а также разработки системного ПО для собственных IP-ядер (custom cores).

Конфигурация средств разработки и оборудования

На рисунке 1 показана типовая схема подключения используемого для разработки оборудования. К рабочей станции через коммутатор подсоединяются рабочие платы (модули ПЛИС). При помощи технологии виртуализации Linux-сервер и рабочая станция могут быть совмещены на одном физическом компьютере.

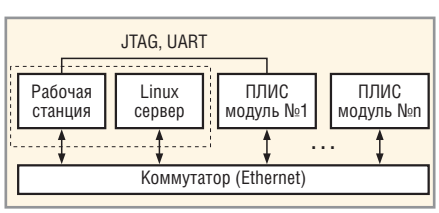


Рис. 1. Схема подключения оборудования

Для отладки применяются кабели JTAG и UART.

На рабочей станции должно быть установлено программное обеспечение для разработки систем на кристалле – Xilinx ISE, XPS. Linux-сервер обеспечивает работу служб DHCP, TFTP, NFS, а также служит основой для разработки (кросс-компиляции) приложений под архитектуру микропроцессорных ядер PPC405, Micro Blaze и т.п.

Система на кристалле ПЛИС

Минимальная тестовая конфигурация для отладочной загрузки ядра Linux должна включать:

- микропроцессорное ядро, контроллер прерываний;
- внешнюю память;
- UART;
- JTAG.

Такая конфигурация позволяет выполнить необходимый минимум: через JTAG загрузить ядро (*zImage.initrd*), создать в памяти виртуальный диск с *rootfs* и подключиться к системе через терминал (UART).

Желательно наличие дополнительных устройств:

- контроллера Ethernet;
- Flash-памяти;
- последовательного ПЗУ (SPROM);
- пользовательского ядра Custom Core.

Начальная конфигурация ПЛИС загружается из SPROM при включении питания. Flash-память содержит либо программный загрузчик (например, U-Boot), либо образ RAM-диска или даже часть *rootfs*.

Через сеть Ethernet производится подключение к удалённому серверу по TFTP, выполняется монтирование сетевых каталогов с NFS; возможно монтирование root. В процессе разработки удобнее загружать ядро и образ RAM-диска с сервера через TFTP, а окончательный вариант записать во flash-память.

Пользовательское ядро решает прикладные задачи, при этом микропроцессорная система под управлением ОС Linux выполняет сервисные функции, служит платформой для разработки и вместе с тем обеспечивает взаимодействие с различными распространёнными интерфейсами.

На рисунке 2 схематично представлена типовая рабочая конфигурации СнК ПЛИС с микропроцессорным ядром PPC 405. Подобные конфигурации будут реализованы и для других типов микропроцессоров, например MicroBlaze, и даже для вашего собственного микропроцессорного ядра (soft core), если в таковом возникнет необходимость.

Использование единственного кристалла ПЛИС не всегда эффективно и возможно. Многое зависит от особенностей прикладного ядра и ресурсов, необходимых для его реализации. Использование кристалла большей ёмкости значительно увеличивает стоимость системы, вместе с тем определённых ресурсов самого ёмкого кристалла может быть недостаточно. Более гибким представляется решение с использованием нескольких кристаллов ПЛИС: «системного» и «прикладного». При таком подходе для реализации прикладного ядра и микропроцессорной системы подбираются адекватные задачам кристаллы ПЛИС. В системном кристалле собирается СнК под управлением Linux, что обеспечивает работу стандартных интерфейсов и протоколов. Пользовательская ПЛИС отводится для решения прикладных задач. Оба кристалла связываются локальной шиной, как показано на рисунке 3.

Также следует отметить проект [4], в котором ПЛИС используется только

для реализации прикладной задачи, а микропроцессорная система под управлением Linux собрана из отдельных микросхем.

Генерация BSP для ядра Linux и загрузчика U-Boot

Процесс генерации BSP в EDK, как для различных версий ядра, так и для микропроцессоров PPC405/Microblaze, сводится к указанию определённых параметров СнК: используемого объёма памяти, перечислению периферийных устройств, рабочего каталога и т.п. Следует обратить внимание на возможные ошибки и несоответствия в коде BSP и в коде используемого ядра. Как правило, они устраняются, но требуют определённых усилий со стороны разработчика.

Генерация BSP для загрузчика U-Boot [5] выполняется на основе соответствующей библиотеки; её необходимо подключить в проект, провести необходимые настройки, аналогичные настройкам при генерации BSP для ядра Linux, и затем выполнить команду *Generate Libraries and BSPs*.

Средства разработки для PPC405

Широкую известность получил скрипт crosstool [6], используя который, можно скомпилировать gcc под заданную микропроцессорную архитектуру.

Использование скрипта может быть следующим:

```
$ su root
$ mkdir /opt/crosstool
$ chown alex:users
/opt/crosstool/
$ su alex
$ cd ~
$ wget
http://kegel.com/crosstool/crosst
ool-0.43.tar.gz
$ tar -xvzf crosstool-0.43.tar.gz
$ cd crosstool-0.43
```

В файле *demo-ppc405.sh* выбираем требуемую версию *gcc* и *glibc*: *$ sh demo-ppc405.sh*. Выполнение скрипта занимает довольно много времени, путь к GCC будет таким: */opt/crosstool/gcc-3.4.4-glibc-2.3.3/powerpc-405-linux-gnu/bin/*.

Вместо Crosstool можно использовать Embedded Linux Development Kit (ELDK) [7] – набор средств разработки для встраиваемых систем. Его удобство заключается в том, что ELDK поставляется в виде отдельного образа инсталляционного диска, на котором имеются варианты установки как под Linux, так и под Windows, с использованием Cooperative Linux.

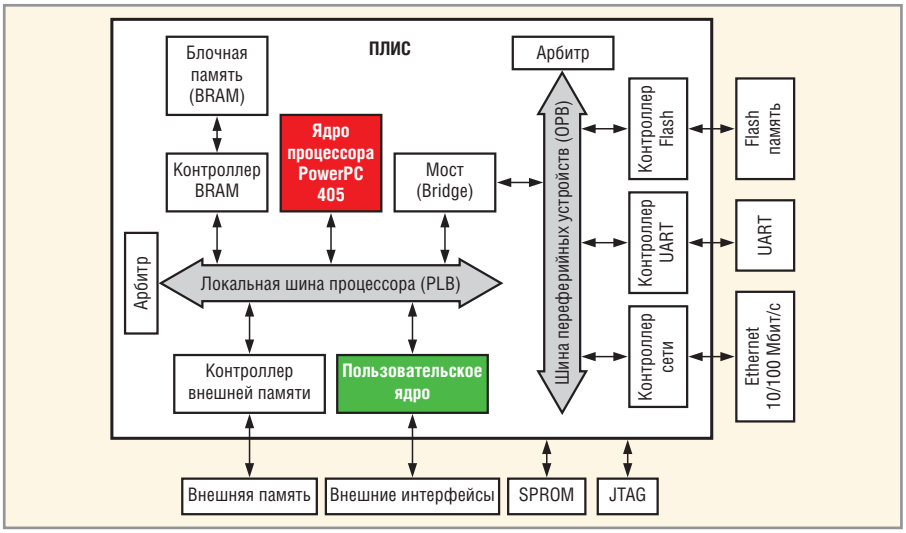


Рис. 2. Система на кристалле ПЛИС с PPC-405

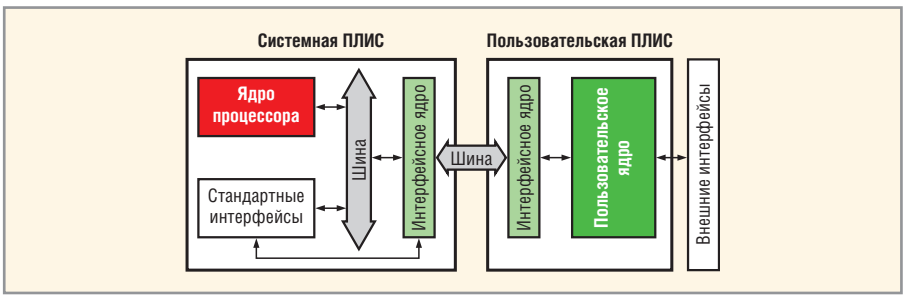


Рис. 3. Система на двух кристаллах ПЛИС

Ядро Linux для PPC405/Mcoroblaze

Основным сайтом разработки ядра Linux является www.kernel.org. Для архитектур PPC405/Mcoroblaze развиваются отдельные ветви, основанные на определённых версиях основного ядра. Со временем изменения вносятся и в основную ветвь ядра, причём данный процесс несколько затянут и несогласован, в результате существует несколько версий ядра со своими особенностями и заплатками (patches). На этом пытаются заработать некоторые компании, предлагая свою поддержку, средства разработки и т.п. Несмотря на сложности, предпочтительно вести собственную разработку, чтобы иметь свободу действий и контролировать проект.

Итак, помимо основной ветки (www.kernel.org), версии ядра Linux 2.4/2.6 для PPC405 можно скачать следующим образом:

```
$ rsync -avz --delete
source.mvista.com::linuxppc_2_4_d
evel linuxppc_2_4_devel
$ cg-clone
git://source.mvista.com/git/linux
-xilinx-26.git
```

Для компиляции в make-файл необходимо прописать переменные *ARCH* и *CROSS_COMPILE*, далее всё как обычно. Настройка параметров ядра выполняется через вызов меню: *$ make menuconfig*.

Объём статьи не позволяет привести список настроек; типовой конфигурационный файл, используемый для плат серии ML, можно получить с сайта Xilinx.

Настроив параметры ядра и скопировав в него BSP для вашей платы, следует выполнить компиляцию *$ make dep && make zImage*.

Чтобы скомпилировать ядро с образом RAM-диска, содержащего *rootfs*, в каталог *arch\ppc\boot\images* помещается упакованный образ диска в файле *ramdisk.image.gz*, далее выполняется команда *make zImage.initrd*.

В отладочном режиме файл *zImage.initrd* может быть загружен через JTAG. Для СнК ПЛИС на базе Microblaze используется uClinux [8] –

дистрибутив, в который входит сразу несколько версий ядра (2.4, 2.6).

Средства кросс-компиляции и генерации BSP доступны по ссылке [9]. Принцип работы с uClinux аналогичный: генерация BSP, настройка параметров ядра, кросс-компиляция и загрузка.

Загрузчик U-Boot

Код загрузчика U-Boot доступен по адресу http://sourceforge.net/projects/u-boot. Генерация BSP для u-boot аналогична генерации BSP для ядра Linux. В случае необходимости в BSP можно добавить код инициализации, например, считывание MAC-адреса для сетевого адаптера из flash-памяти.

В каталоге *include/configs/* должен быть создан заголовочный файл рабочей платы, в котором определяются поддерживаемые команды и скрипты. Проверить полученный код U-Boot можно, загрузив образ через JTAG. В рабочем режиме код загрузчика хранится во flash-памяти.

Загрузчик/отладчик U-Boot может быть использован как самостоятельная среда для отладки и тестирования, в которой есть командный интерпретатор и поддерживается работа с различными периферийными устройствами. Загрузка Linux с использованием U-Boot может быть выполнена следующим образом: из flash-памяти считывается MAC-адрес сетевого адаптера, DHCP выделяет IP-адрес, через TFTP скачивается ядро Linux и образ RAM-диска, автоматически подставляются параметры загрузки ядра, и затем выполняется его запуск.

RootFS

Наиболее простой путь создания *rootfs* заключается в использовании готового набора базовых пакетов для Linux-системы, такого как BusyBox [10], а также в ручной настройке системных файлов и скриптов.

Если существует необходимость в более мощных средствах работы с дистрибутивом Linux, используйте ScratchBox [11]. Хорошим примером проекта на его основе является Nokia Internet Tablet [12].

Разработка системного ПО для пользовательских IP-ядер

Разработав собственное IP-ядро и подключив его к системной шине, можно быстро провести отладку и тестирование под Linux, используя режим user mode, при этом не потребуется написание драйвера, а только тестовое приложение. Типовой make-файл для компиляции таких приложений следующий:

```
# Program
PROGRAM=test-app
SRCS=$(PROGRAM).c
CC=/opt/eldk/usr/bin/ppc_4xx-gcc
TOPDIR=/opt/eldk/ppc_4xx/usr
LIBDIR=$(TOPDIR)/lib
INCDIR=$(TOPDIR)/include
CFLAGS=-I$(INCDIR)
LDFLAGS=-L$(LIBDIR)
OBJS=$(SRCS:.c=.o)
# Rules
.SUFFIXES: .c .o
.c.o:
      $(CC) -c $(CFLAGS) $<
default: all
all: $(PROGRAM)
$(PROGRAM): $(OBJS)
      $(CC) $(OBJS) -o $(PROGRAM)
$(LDFLAGS)
clean:
      rm -f $(OBJS) $(PROGRAM)
```

Само приложение должно содержать код инициализации, в котором указанный физический адрес устройства (custom core) проецируется на виртуальный адрес, с которым можно работать непосредственно из приложения:

```
int core_fd=-1;
int *core_ptr=NULL;
off_t core_target=CORE_BASEADDR;
/* map custom core
to user space
*/
fd = open("/dev/mem",
O_RDWR | O_SYNC);
if(fd==-1){
/*failed*/
goto __exit_and_clean;
}
// get memory pointer
core_ptr = (int *)mmap(0,
CORE_MAP_SIZE,
PROT_READ|PROT_WRITE,
MAP_SHARED,
core_fd,
core_target &
~CORE_MAP_MASK);
if(core_ptr == MAP_FAILED){
/*failed*/
goto __exit_and_clean;
}
core_virt_addr=core_ptr+
(core_target & CORE_MAP_MASK);
// place here test code…
__exit_and_clean:
if(core_ptr!=MAP_FAILED)
munmap(core_ptr,CORE_MAP_SIZE);
if(core_fd!=-1)close(core_fd);
```

Написание драйвера займёт больше времени. Для компиляции кода можно использовать следующий make-файл:

```
# Driver
DRIVER=my_core
SRC=$(DRIVER).c
OBJ=$(DRIVER).o
DEPLOY=$(deploy)
CC=/opt/eldk/usr/bin/ppc_4xx-gcc
TOPDIR=/opt/eldk/ppc_4xx/usr
KINCDIR=/opt/linuxppc_2_4/include
LIBDIR=$(TOPDIR)/lib
INCDIR=$(TOPDIR)/include
CFLAGS=-c -O -DMODULE -D__KERNEL__
-I$(KINCDIR)
COMPILE=$(CC) $(CFLAGS) $(SRC)
#Rules
default: $(SRC)
        $(COMPILE)
clean:
        rm -f $(OBJ)
```

Для отладки лучше использовать динамическую загрузку:

```
$ insmod my_core.o
$ lsmod
$ rmmod my_core
```

При необходимости драйвер можно скомпилировать с ядром или включить в процесс генерации BSP.

Выводы и рекомендации

Для СнК ПЛИС операционная система Linux является наиболее адекватной платформой, которая обеспечивает поддержку множества периферийных устройств, различных интерфейсов и протоколов. В распоряжении разработчика имеется огромное разнообразие ПО с открытым исходным кодом, которое может быть кросс-компилировано под вашу платформу.

В качестве рекомендации для разработчиков хочется процитировать слова Дена Кегеля: «Создание средств разработки gcc / glibc для программирования встраиваемых систем бывало настолько трудным, что просто требовало железной воли, долгих дней, если не недель тяжелых усилий,

глубоких профессиональных знаний Unix и GNU, а иногда просто изворотливости».

Именно наличие «железной воли» позволяет добиваться результатов. Когда вы сталкиваетесь с ошибками компиляции, работая с кодом Linux, BSP и т.д., не отчаивайтесь – копируйте текст ошибки в поисковый запрос, и в большинстве случаев найдётся описание того, как решается данная проблема.

Продолжение следует

Литература

1. ОС Linux http://en.wikipedia.org/wiki/Linux.
2. RMB-411 www.rosta.ru.
3. ML401 www.xilinx.com\ml401.
4. http://www.elphel.com/3fhlo/index.html.
5. Загрузчик U-boot http://u-boot.sourceforge.net.
6. http://kegel.com/crosstool.
7. ELDK www.denx.de.
8. uClinux http://www.uclinux.org.
9. Microblaze toolchain http://www.petalogix.com/resources/downloads.
10. BusyBox http://www.busybox.net.
11. ScratchBox http://www.scratchbox.org.
12. Nokia Internet Tablet www.maemo.org.

Новости мира News of the World Новости мира

Sony: демонстрация сверхтонких OLED-телевизоров

На январской выставке потребительской электроники CES 2007 посетители стенда компании Sony могли полюбоваться интересными моделями OLED-телевизоров с диагоналями 11 и 27 дюймов. Тогда информации о новинках было немного, сегодня же появилась возможность восполнить этот пробел.

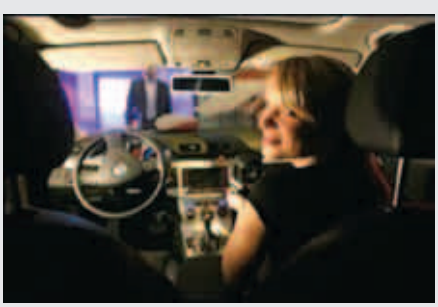
На японской выставке Display 2007, которая проходила в этом году с 11 по 13 апреля, компания ещё раз продемонстрировала публике свои новые телевизоры. Как стало известно, 11-дюймовая модель имеет разрешение 1024 × 600 пикселов, а 27" оснащена Full HD-матрицей (1920 × 1080 пикселов). При этом обе модели имеют поразительно высокую контрастность 1 000 000 : 1 (вероятно, речь идет о динамической контрастности), цветовое пространство по шкале NTSC более 100%, яркость 600 нит, интерфейс HDMI.

Особо изумляет толщина панелей: всего 3 мм для 11" панели и 9 мм – для 27". Кроме этого, OLED-телевизоры Sony потребляют гораздо меньшую мощность, чем аналогичные плазменные и жидкокристаллические. Разработки 11-дюймовой модели уже находятся на стадии, близкой к массовому производству. О сроках начала коммерциализации 27" телевизоров пока не сообщается.

dailytech.com

Беспилотный автомобиль Volkswagen Passat 2.0 TDI

Компания Volkswagen представила автоматизированный автомобиль Volkswagen Passat 2.0 TDI, который абсолютно не нуждается в водителе. Вместо него в корпус автомобиля вмонтировано два лазерных сенсора (один спереди, а другой сзади), которые играют роль технологичных глаз – следят за дорогой, оценивают расположение зданий, пешеходов в определённый момент времени на расстоянии до 200 м. Вся информация, собранная этими сенсорами, обрабатывается специальным компьютером, работающим под управлением ПО, которое разработала немецкая компания Ibeo Automobile Sensor.



В ноябре этот автомобиль будет соревноваться с другими роботизированными системами в небезызвестном шоу – гонках беспилотных автомобилей DARPA 2007, когда машинам придётся проехать самостоятельно более 60 миль.

По словам разработчиков, «коньком» проекта является совершенная лазерная технология построения моделей окружающей среды и ориентирования. Интересно отметить, что развернуть масштабное производство таких систем разработчики планируют уже в следующем году, хотя американские учёные предсказывали первые коммерческие беспилотные авто только через 25 лет.

physorg.com

IBM: третье измерение продлит жизнь закону Мура

По мнению компании IBM, будущие суперкомпьютеры заметно выиграют в случае использования так называемой трёхмерной системы чипов. Сегодня чипы размещаются в двухмерной плоскости, а сообщение между ними организуется посредством проводных соединений, которые обладают далеко не идеальной пропускной способностью.

История такой технологии уходит в середину 90-х годов прошлого века. Инженерам потребовалось десятилетие на то, чтобы подготовить технологию к современному массовому производству. В соответствии с новой технологией, предполагается размещать чипы непосредственно друг на друге, а соединения между чипами осуществлять через каналы в кремниевом корпусе чипов. Данная технология получила название through-silicon vias, т.е. «соединения сквозь кремний». Преимущество достигается за счёт уменьшения расстояние между чипами в 1000 раз. Оно заключается в увеличении пропускной способности межчиповых соединений в 100 раз. Таким образом, по мнению сотрудников IBM, эффективность чипов повышается на 40%, что открывает путь новому поколению суперкомпьютеров.

Причём описанная технология, очевидно, не задержится в тесном кругу инженеров и исследователей. Ещё в прошлом году компания представила инновационный «многоэтажный» чип, оснащённый 80 ядрами и способный совершать более триллиона операций в секунду. Массовое же производство «многоэтажных» чипов IBM начнёт в 2008 г., представив предварительно несколько пробных вариантов уже в этом году. Более того, компания IBM не является единственной компанией, работающей в данном направлении. В то время как Intel пока только разрабатывает схожую технологию вертикального размещения чипов, компания Tru-Si уже обладает таковой.

news.bbc.co.uk