Проектиране на вградени автомобилни електронни системи
Лабораторно упражнение №17

Работа с Petalinux. Драйвери за персонализиран модул в системи с Линукс.

==========

1. В настоящото лабораторно упражнение ще се използва Линукс ядрото, компилирано в предходното лабораторно [1]. Стартирайте терминал и се преместете в директорията на PetaLinux:

```
cd /home/user/workspaces/petalinux_workspace/16_linux_petalinux
```

след което инициализирайте PetaLinux средата:

```
source /home/user/programs/PetaLinux/settings.sh
```

2. Изпълнете командата (не трябва да има знак за подчертаване _ в името):

```
petalinux-create -t modules --name absgradientout --enable
```

която трябва да ви върне:

```
INFO: Create modules: absgradientout
INFO:      New      modules      successfully      created      in
/home/user/workspaces/petalinux_workspace/16_linux_petalinux/proje
ct-spec/meta-user/recipes-modules/absgradientout
INFO: Enabling created component...
INFO: absgradientout has been enabled
```

От тук се вижда, че темплейт сорс файл на драйвера е създаден в директорията:

/home/user/workspaces/petalinux_workspace/16_linux_petalinux/project-spec/meta-user/recipes-modules/absgradientout/files/**absgradientout.c**

където също има и Makefile.

3. Компилирайте Линукс ядрото, заедно с вашия (out-of-tree) драйвер:

```
petalinux-build
```

След изпълнението на тази команда, драйверът absgradientout.ko може да бъде намерен в директорията:

/home/user/workspaces/petalinux_workspace/16_linux_petalinux/build/tmp/sysroots-components/zynq_generic_7z010/absgradientout/lib/modules/6.1.5-xilinx-v2023.1/extra/**absgradientout.ko**

Командата petalinux-build извършва следните операции:

`petalinux-build -c kernel` (компилира Линукс ядрото)
`petalinux-build -c absgradientout` (компилира Линукс драйвера)
`petalinux-build -c rootfs` (добавя драйвера във файловата система и създава наново файловете за стартиране на Линукс)

4. Заредете Линукс върху FAT32 партишъна на SD карта като копирате следните файлове:

Partition 1 (FAT32)
images/linux/**boot.scr**
images/linux/**image.ub**
build/tmp/sysroots-components/zynq_generic_7z010/xilinx-bootbin/boot/**BOOT.BIN**

След което се влезте с потребител:

user: petalinux
password: въведете нова парола

Преместете се в следната директория:

```
cd /lib/modules/6.1.5-xilinx-v2023.1/extra
```

където 6.1.5-xilinx-v2023.1 може да се променя с времето. Заредете драйвера с командата modprobe:

```
sudo modprobe absgradientout.ko
```

при което би трябвало да видите съобщението:

```
bsgradientout: loading out-of-tree module taints kernel.
<1>Hello module world.
<1>Module parameters were (0xdeadbeef) and "default"
```

Със следната команда може да се види дали модулът е зареден в Линукс ядрото:

```
lsmod
```

Със следната команда може да се премахне модулът от Линукс ядрото:

```
rmmod absgradientout
```

5. Преди да се премине към разработката на кода за драйвера, трябва да се разработи приложна програма, която работи във виртуалното адресно поле user space (драйверите работят в kernel space) [2][3]. За целта, стартира се развойната среда Vitis Classic → File → New → Platform project → дава се име на проект: 17_kernel_modules_pla → Next → Browse → указва се пътя до настоящото лабораторно, където в директория 17_5 има файл design_1_wrapper.xsa → избира се файлът design_1_wrapper.xsa → в падащото меню Processor: изберете ps7_cortexa9_0 → Next → Finish.

Вляво → таб Explorer → двукратно щракване върху platform.spr → в новоотвореният таб 17_kernel_modules_pla → натиснете зеленият символ плюс + (Add domain)→в новоотвореният прозорец New domain in “17_kernel_modules_pla” въведете следната информация:

Name: linux_domain
Display Name: linux_domain
OS: linux
Processor: ps7_cortexa9
Supported Runtimes: C/C++
Architecture: 32-bit
Bif file: оставете празно
Boot Components Directory: оставете празно
FAT32 Partition Directory: оставете празно

и натиснете OK. Натиснете бутонът Build.

Създайте приложен проект File → New → Application project → Next →избира се 17_kernel_modules_pla за платформа → Next → Application project name: 17_kernel_modules_app, както и Select target processor for the Application project: ps7_cortexa9_SMP → Next → Next → Linux Hello World → Finish.

Отваря се helloworld.c от Explorer на Vitis → 17_kernel_modules_app_system → 17_kernel_modules_app → src → helloworld.c и се въвежда следната програма:

```c
#include <stdio.h>

int main(){
      int i;

      for(i = 0; i < 10; i++){
            printf("Hello, World, from user space! (%d)\n", i);
      }

    return 0;
}
```

след което се натиска бутон Build. Забележете, че кроскомпилаторът се е променил (спрямо baremetal фърмуера, който използва arm-none-eabi-gcc):

```
arm-linux-gnueabihf-gcc
```

Добавете приложната програма във файловата система на Линукс. За целта, в терминала на PetaLinux напишете:

```
petalinux-create -t apps --template install --name hellow --enable
```

където hellow е името на приложната програма. След това изтрийте темплейт приложението:

```
rm project-spec/meta-user/recipes-apps/hellow/files/hellow
```

Копирайте и преименувайте вашето приложение от проекта на Vitis:

/home/user/workspaces/vitis_workspace/17_kernel_modules_app/Debug/17_kernel_modules_app.elf

в директорията на PetaLinux:

/home/user/workspaces/petalinux_workspace/project-spec/meta-user/recipes-apps/hellow/files/hellow

Забележете, че сме преименували:

17_kernel_modules_app.elf → hellow

В терминала на PetaLinux въведете командата:

```
petalinux-build
```

Заредете Линукс на SD картата и след логване с petalinux потребителското име, стартирайте приложението:

```
16_linux_petalinux:~$ hellow
Hello, World, from user space! (0)
Hello, World, from user space! (1)
Hello, World, from user space! (2)
Hello, World, from user space! (3)
Hello, World, from user space! (4)
Hello, World, from user space! (5)
Hello, World, from user space! (6)
Hello, World, from user space! (7)
Hello, World, from user space! (8)
Hello, World, from user space! (9)
```

Забележка: операциите по сваляне, изпълнение и дебъгване на user space програмата може да се направи и от Vitis [1]. За целта трябва да се направи връзка с Zybo платката. Това става посредством сървърно приложение, което се нарича TCF Agent. Това приложение по подразбиране е включено в инсталацията на Линукс, когато сме създали проект с PetaLinux, и то се пуска при стартирането на системата. Връзката с TCF Agent става по Ethernet. Компютърът, на който е пуснат Vitis и Zybo трябва да са в локална мрежа. Zybo платката трябва да се включи с Ethernet кабел включен предварително!

ВНИМАНИЕ! Програмируемата матрица с Линукс и Ethernet изискват стабилно захранващо напрежение от мощен източник. Платката трябва да е включена директно в компютъра посредством само един USB кабел и/или трябва да се включи DC адаптер към куплунг J15.

Забележка: ако се работи само в PetaLinux, без да се използва Vitis, командите са следните (предполага се, че името на приложението е myapp):

```
petalinux-build -c myapp
petalinux-build -c rootfs
petalinux-build -x package
```

а изтриване на приложението става с:

```
petalinux-build -c myapp -x do_clean
```

6. Драйверите за Линукс може да се зареждат при стартиране на системата, ако дървесното им двоично описание е добавено към останалите. Тогава след init( ) функцията, ще се извика функцията probe( ) която свързва физическите адреси (в нашия случай 0x43c00000) с адреси във виртуалното поле на системата (например 0xe0970000).

За целта, “compatible” низът в дървесното двоично описание трябва да е еквивалентно на “compatible” низът в драйвера.

Автоматично генерираните дървесни описания на потребителски персонализирани модули (custom IP) се намират в:

/home/user/workspaces/petalinux_workspace/16_linux/components/plnx_workspace/device-tree/device-tree/pl.dtsi

Ето така изглежда описанието на персонализираният модул:

```
/*
 * CAUTION: This file is automatically generated by Xilinx.
 * Version: XSCT
 * Today is: Sun Nov 26 13:17:15 2023
 */


/ {
     amba_pl: amba_pl {
          #address-cells = <1>;
          #size-cells = <1>;
          compatible = "simple-bus";
          ranges ;
          abs_gradient_out_0: abs_gradient_out@43c00000 {
               clock-names = "s00_axi_aclk";
               clocks = <&clkc 15>;
               compatible = "xlnx,abs-gradient-out-1.0";
               interrupt-names = "interrupt_0", "interrupt_1";
               interrupts = <0 29 4 0 2>;
               reg = <0x43c00000 0x10000>;
               xlnx,s00-axi-addr-width = <0x6>;
               xlnx,s00-axi-data-width = <0x20>;
          };
     };
};
```

В случая низът “xlnx,abs-gradient-out-1.0” трябва да се въведе в драйвера. Номерът на прекъсването е 61 – 32 = 29, където 61 е номерът даден от Vivado, а 32 е число, което Линукс изважда, за да се получи крайният номер на прекъсването. Числото 4 означава, че имаме прекъсване по нарастващ фронт.

Дървесните двоични описания, които потребителя иска да добави към автоматично генерираните, се намират в директорията [4]:

/home/user/workspaces/petalinux_workspace/16_linux/project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi

Напишете приложна програма за Линукс, която отваря файл, отговарящ за комуникацията с персонализирания модул, и записващ низа “1”:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int main(int argc, char **argv){
	int i;
	int fd;

	fd = open("/dev/absgradientout", O_RDWR);

	if ( fd == -1 ) {
		perror("open failed");
		exit(-1);
	}

	printf("open: successful\n");

	for(i = 0; i < 10; i++){
		write(fd, "1", 1);
		usleep(1000000);
	}

	close(fd);

	return 0;
}
```

след което вградете програмата във файловата система, както в предходната точка, чрез копиране и преименуване на .elf файла от Vitis в:

/home/user/workspaces/petalinux_workspace/project-spec/meta-user/recipes-apps/hellow/files/**hellow**

Въведете кода на драйвера, даден в директория **17_6** на настоящото лабораторно.

ВНИМАНИЕ! Проверете compatible низът от pl.dtsi. Ако той се различава от “xlnx,abs-gradient-out-1.0”, въведете новата стойност в драйвера ви.

Компилирайте Линукс и го заредете на SD картата. След като се въведе логин името, трябва да се разреши достъпа до dev файла absgradientout:

```
sudo chmod 777 /dev/absgradientout
```

След това можете да стартирате приложението:

```
hellow
```

7. Модифицирайте приложната програма, така че да се генерират 10 правоъгълни импулса на изводите get_ports gpio_out_0_0 (V20), get_ports gpio_out_1_0 (W20). Свържете осцилоскоп на JB конектора към изводи JB1_P и JB1_N. Пуснете приложната програма и наблюдавайте осцилоскопа.

ЗАБЕЛЕЖКА: работната маса с платка Zybo Z7-10 трябва да използва куплунг JC и съответно сигналите gpio_out_0_0 (T11), gpio_out_1_0 (T10).

* * *

[1] https://xilinx.github.io/Embedded-Design-Tutorials/docs/2021.1/build/html/docs/Introduction/Zynq7000-EDT/4-linux-for-zynq.html

[2] https://xilinx.github.io/Embedded-Design-Tutorials/docs/2021.1/build/html/docs/Introduction/Zynq7000-EDT/8-custom-ip-driver-linux.html

[3] https://github.com/LubomirBogdanov/PVS/tree/master/05_linux

[4] https://docs.xilinx.com/r/en-US/ug1144-petalinux-tools-reference-guide/Configuring-Device-Tree

[5] https://support.xilinx.com/s/article/61117?language=en_US

доц. д-р инж. Любомир Богданов, 2023 г.