Проектиране на вградени автомобилни електронни системи
Лабораторно упражнение №16

Работа с Xilinx Vivado и Vitis. Асиметрична мултипроцесорна система (AMP) с Linux и с baremetal фърмуер.

==========

1. В настоящото лабораторно ще се използва пакетът от програми PetaLinux на Xilinx, който има за цел да улесни компилацията на Линукс за вградени многопроцесорни системи върху чип и да модифицира ядрото, така че да се изпълнява на конкретния дизайн.

Изискванията към хардуерът са следните:
* един модул от вида троен таймер (triple timer counter – TTC);
* един модул от вида UART;
* контролер за външна оперативна памет ;
* външна оперативна памет с минимален обем 512 MB;
* енергонезависима памет (QSPI Flash, SD/MMC);
* модул Етернет (незадължителен).

За щастие матрицата XC7Z010 поддържа TTC и Етернет, има контролер за външна DDR2 и DDR3 памети. Този чип позволява начално стартиране (boot) от външна QSPI Flash или SD карта. Демо платката Zybo осигурява както външна 512 MB DDR3, така и чип, реализиращ физическия слой на Етернет, а също и куплунг за SD карта. С други думи, **платката Zybo е съвместима с PetaLinux**.

2. Инсталацията на PetaLinux минава през следните етапи (за Ubuntu 20.04 64-bit):

*От сайта на Xilinx се сваля последната версия, която за настоящото упражнение е 2023.1. Инсталационният файл се казва:

petalinux-v2023.1-05012318-installer.run

*Създава се инсталационна директория и се инсталира в нея PetaLinux:

`sudo apt install gawk xterm autoconf libtool texinfo` (възможно е да има и други, инсталаторът ще ги покаже)

```
mkdir -p /home/user/programs/PetaLinux
```

```
sudo chmod 755 ./petalinux-v2023.1-05012318-installer.run
md5sum petalinux-v2023.1-05012318-installer.run
```
(→ 78fd08837e2d30541190a7ff20988e0f)

```
./petalinux-v2023.1-05012318-installer.run -d /home/user/programs/PetaLinux
```

Използват се бутони Enter, q, y на клавиатурата за съответно показване, затваряне и приемане на софтуерните споразумения. Ето примерно изпълнение:

```
INFO: Installing PetaLinux...
INFO: Checking PetaLinux installer integrity...
INFO: Installing PetaLinux SDK to "/home/user/programs/PetaLinux/."
INFO:                   Installing                  buildtools                  in
/home/user/programs/PetaLinux/./components/yocto/buildtools
INFO:                Installing                buildtools-extended              in
/home/user/programs/PetaLinux/./components/yocto/buildtools_extended
INFO: PetaLinux SDK has been installed to /home/user/programs/PetaLinux/.
```

* Експортват се настройките на пакета в терминала:

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

* Проверява се експорта чрез показване на променливата $PETALINUX

```
echo $PETALINUX
```

Тази команда трябва да върне:

```
/home/user/programs/PetaLinux
```

3. Превключете джъмпера вдясно на платката **на позиция SD**. Свържете μUSB кабел към PROG/UART USB куплунга. Включете платката от ключа ON/OFF.

4. Стартирайте терминал с CTRL + ALT + T и изпълнете командите:

```
source ~/programs/xilinx/Vivado/2022.2/settings64.sh
vivado
```

5. Създава се системата от лабораторно упражнение “Многопроцесорна система с персонализиран IP модул на Verilog. Синхронизация чрез прекъсвания.”, но със следните разлики:

* В блок Zynq7 се оставят отметките на ENET0, USB0 и SD0.
* В блок Zynq7 се поставя отметка на Fabric interrupts → PL-PS Interrupt Ports → отметка на IRQ_F2P.
* Изтрийте concat блока на контролера на прекъсванията на MicroBlaze.
* Свържете interrupt_0 на входа към IRQ_F2P на ZYNQ7.

* Свържете interrupt_1 на входа към intr[0:0] на AXI Interrupt Controller на MicroBlaze.
* Във Vivado → Tools → Settings → IP → Repository → вдясно → IP Repositories → Add → vivado_workspace/ip_repo (тази директория се е създала автоматично в миналото упражнение) → би трябвало да се вижда персонализираният модул abs_gradient_out_v1.0 → OK → OK. Сега модулът abs_gradient_out ще е достъпен в редактора на блоковата схема.

Блоковата схема е показана на следващата страница, а промяната в свързването е показана с оранжево.

6. Вдясно→Diagram→лента с бутони → Validate Design (F6) → "Validation successful. There are no errors or critical warnings in this design." → OK.

В някои версии на Vivado е възможно да се появят предупредителни съобщения, относно отрицателни стойности на параметрите DDR_DQS_TO_CLK_DELAY_x, но те могат да се игнорират в конкретния дизайн.

7. Централно→в Block design прозореца, натиска се таб-а Sources → Design sources → right-click на design_1.bd → Create HDL Wrapper (създава VHDL описание на новосъздадената система) → Let Vivado manage wrapper and auto-update → OK

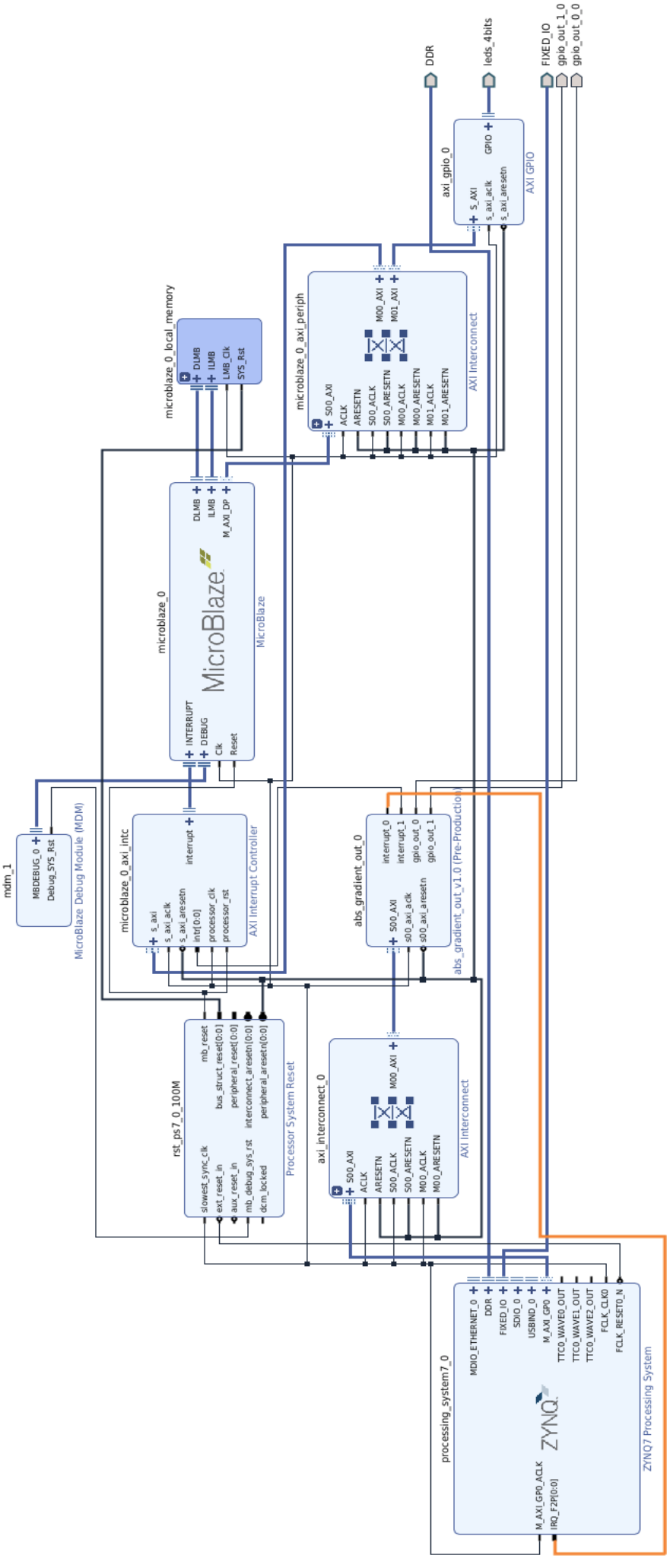
8. Добавя се constraints file, в който ще се опише извеждането на сигналите gpio_out_0 и gpio_out_1 на изводи съответно V20 и W20, които излизат на JB конектора под имената JB2_P и JB2_N.

Вляво → таб Sources → десен бутон върху категорията Constraints → Add Sources → Add or create constraints → Next → Create file → File name: abs_gradient_out → OK → Finish.
Отворете категорията Constraints → constrs_1 (1) → двукратно щракване на abs_gradient_out.xdc →въвежда се:

```
#GPIO 0
set_property PACKAGE_PIN V20 [get_ports gpio_out_0_0]
set_property IOSTANDARD LVCMOS33 [get_ports gpio_out_0_0]

#GPIO 1
set_property PACKAGE_PIN W20 [get_ports gpio_out_1_0]
set_property IOSTANDARD LVCMOS33 [get_ports gpio_out_1_0]
```

Проверете в design_1_wrapper.v дали наистина имената на сигналите са gpio_out_0_0 и gpio_out_0_1.


DDR
leds_4bits
FIXED_IO
gpio_out_1_0
gpio_out_0_0
axi_gpio_0
S_AXI
s_axi_aclk
s_axi_aresetn
GPIO
AXI GPIO
microblaze_0_axi_periph
S00_AXI
ACLK
ARESETN
S00_ACLK
S00_ARESETN
M00_ACLK
M00_ARESETN
M01_ACLK
M01_ARESETN
M00_AXI
M01_AXI
AXI Interconnect
microblaze_0_local_memory
DLMB
ILMB
LMB_Clk
SYS_Rst
microblaze_0
MicroBlaze
INTERRUPT
DEBUG
Clk
Reset
M_AXI_DP
MicroBlaze
mdm_1
MBDEBUG_0
Debug_SYS_Rst
MicroBlaze Debug Module (MDM)
microblaze_0_axi_intc
s_axi
s_axi_aclk
s_axi_aresetn
interrupt
intr[0:0]
processor_clk
processor_rst
AXI Interrupt Controller
abs_gradient_out_0
S00_AXI
s00_axi_aclk
s00_axi_aresetn
interrupt_0
interrupt_1
gpio_out_0
gpio_out_1
abs_gradient_out_v1.0 (Pre-Production)
rst_ps7_0_100M
slowest_sync_clk
ext_reset_in
aux_reset_in
mb_debug_sys_rst
dcm_locked
mb_reset
bus_struct_reset[0:0]
peripheral_reset[0:0]
interconnect_aresetn[0:0]
peripheral_aresetn[0:0]
Processor System Reset
axi_interconnect_0
S00_AXI
ACLK
ARESETN
S00_ACLK
S00_ARESETN
M00_ACLK
M00_ARESETN
M00_AXI
AXI Interconnect
processing_system7_0
MDIO_ETHERNET_0
DDR
FIXED_IO
SDIO_0
USBIND_0
M_AXI_GP0
TTC0_WAVE0_OUT
TTC0_WAVE1_OUT
TTC0_WAVE2_OUT
FCLK_CLK0
FCLK_RESET0_N
M_AXI_GP0_ACLK
IRQ_F2P[0:0]
ZYNQ
ZYNQ7 Processing System

9. Вляво → Flow navigator → Generate bitstream → Yes → OK → изчаква се няколко минути (докато завърши синтеза) → View reports → OK

ВНИМАНИЕ: долу, централно, в таб Log може да наблюдавата съобщенията от синтеза. Най-горе, вдясно на Vivado прозореца ще видите иконка на въртящ се зелен часовник. Докато тя е видима, значи трябва да се изчака.

10. File → Export → Export hardware → Next → Include bitstream → Next → Next → Finish

==========
==========

11. Отворете терминал и създайте нова директория:

```
cd workspaces
mkdir petalinux_workspace
cd petalinux_workspace
```

Изпълнете командите:

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

petalinux-create --type project --template zynq --name 16_linux_petalinux
```

Последната команда трябва да ви върне:

```
INFO: Create project: 16_linux_petalinux
INFO:         New         project         successfully         created         in
/home/user/workspaces/petalinux_workspace/16_linux_petalinux
```

След това влезте в директорията:

```
cd 16_linux_petalinux
```

12. Създайте нова конфигурация на Линукс за вашият дизайн, подавайки хардуерното описание design_1_wrapper.xsa като параметър:

```
petalinux-config                                                --get-hw-description
/home/user/workspaces/vivado_workspace/16_linux/design_1_wrapper.xsa
```

В отворилото се меню на menuconfig (син екран) с дясна стрелка изберете Exit → Yes

В това меню се настройват Линукс ядрото, програмата за начално установяване (bootloader), програмите, които ще се включат в Линукс дистрибуцията, build

системата и др.
Build системата е Yocto, аналогична на Buildroot.

```
petalinux-build
```

Въоръжете се с търпение – на компютър със 70-Mbit връзка с Интернет, процесор Intel Core i3-10100 (8 x 3.6 GHz), 16 GB DDR3, тази команда отнема 20 минути.

Примерен отговор на командата е:

```
[INFO] Sourcing buildtools
[INFO] Building project
[INFO] Silentconfig project
[INFO] Silentconfig rootfs
[INFO] Generating workspace directory
INFO: bitbake petalinux-image-minimal
NOTE:            Started            PRServer            with            DBfile:
/home/user/workspaces/petalinux_workspace/16_linux_petalinux/build/cache/
prserv.sqlite3, Address: 127.0.0.1:46541, PID: 36742
Loading                     cache:                     100%                    |
| ETA:  --:--:--
Loaded 0 entries from dependency cache.
Parsing                    recipes:                     100%                    |
##################################################################################
##################################################################################
#####| Time: 0:02:13
Parsing of 4344 .bb files complete (0 cached, 4344 parsed). 6275 targets, 357
skipped, 1 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies
NOTE:            Fetching            uninative            binary            shim
file:///home/user/workspaces/petalinux_workspace/16_linux_petalinux/components/
yocto/downloads/uninative/
5fab9a5c97fc73a21134e5a81f74498cbaecda75d56aab971c934e0b803bcc00/x86_64-
nativesdk-libc-
3.8.1.tar.xz;sha256sum=5fab9a5c97fc73a21134e5a81f74498cbaecda75d56aab971c934e0b8
03bcc00 (will check PREMIRRORS first)
Initialising                  tasks:                    100%                    |
##################################################################################
##################################################################################
##| Time: 0:00:06
Checking      sstate      mirror      object      availability:      100%      |
##################################################################################
##########################################################| Time: 0:00:31
Sstate summary: Wanted 1550 Local 0 Mirrors 1396 Missed 154 Current 0 (90%
match, 0% complete)
WARNING:     The     busybox:do_fetch     sig     is     computed     to     be
16146b24d18a0700bfca13cdb5b9e2b8f48fd710c6a24ce7f779c35c33ad65c0, but the sig is
locked  to  c2ac5f0c29da84190d0f46f4f48dcf67ef7255138a4424d0a00004f7f614a942  in
SIGGEN_LOCKEDSIGS_t-cortexa9t2hf-neon
NOTE: Executing Tasks
NOTE: Tasks Summary: Attempted 4139 tasks of which 3501 didn't need to be rerun
and all succeeded.

Summary: There was 1 WARNING message.
INFO: Failed to copy built images to tftp dir: /tftpboot
```

```
[INFO] Successfully built project
```

Последният warning може да се пренебрегне (Линуксът ще се стартира от SD карта, а не от мрежата).

13. Разделя се и се форматира картата. Копират се файловете (от top-level директория petalinux_workspace/16_linux_petalinux):

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

Partition 2 (ext4)
images/linux/**rootfs.tar.gz**

на SD карта. Дава се cut на всички файлове от Partition 2, директория rootfs, и се копират с cut/paste едно ниво по-нагоре. Директории proc и sys няма да може да ги преместите, затова направете празни такива. Изтриите папката rootfs от Partition 2.

Файлът BOOT.BIN съдържа:
*First Stage Boot Loader (FSBL), който всъчщност се изпълнява втори след BootROM;
*Bistream за програмируемата логика;
*Потребителска програма за ARM Cortex A9(0) – това може да е U-boot – програма за вторично установяване (Second Stage Boot Loader, SSBL) или baremetal фърмуер (.elf) в случай, че не се използва Линукс.

Файлът image.ub съдържа:
*Линукс ядро;
*дървесни двоични описания (dtb);
*минималистична rootfs файлова система.

Файлът boot.scr е скрипт с команди, които трябва да се изпълнят от U-boot, за да стартират Линукс ядрото.

Файлът rootfs.tar.gz е архив с файловата система, която се използва от потребителя след инициализацията на Линукс.

ВНИМАНИЕ! За да може FSBL да зареди U-boot / потребителски фърмуер, във Vivado трябва да е избрана опцията SD0 (двукратно щракване върху Zynq7 → Page navigator → MIO Configuration → I/O peripherals → SD0).

14. Поставя се SD картата в слот J4 (SD MICRO) от обратната страна на Zybo. Картата трябва да е с пиновете нагоре (обърната).

15. От менюто на Ubuntu се избира Cutecom (или gtkterm) и се настройва връзка на /dev/ttyUSB1 с параметри 115200-8-N-1. Рестартира се XC7Z010 с бутон BTN7 на Zybo. Ако компилацията е била успешна, трябва да се види менюто на U-boot, последвано от стартирането на Линукс и команден ред, подканващ за потребителско име. Понеже това е първо стартиране, трябва да се избере парола за администратор на системата.

```
User: petalinux
Pass: <по избор>
```

16. Разгледайте файловата система с командите:

```
cd ../../
ls
```

както и инсталираните програми с:

```
help
```

За да загасите платката, напишете следната команда преди да премахнете захранването:

```
sudo shutdown -h now
```

17. Тествайте персонализираният IP модул без Линукс (върнете джъмпера JP5 в позиция JTAG) – с baremetal фърмуер.

Въведете в main_mb.c на микропроцесора MicroBlaze следната програма (този път има само едно прекъсване – от abs_gradient_out, сигнал interrupt_1):

```
#include <stdio.h>
#include <xgpio.h>
#include "xparameters.h"
#include "sleep.h"
#include "xil_exception.h"
#include "xintc.h"

typedef struct {
      UINTPTR BaseAddress;     /**< Device base address */
      u32 IsReady;             /**< Device is initialized and ready */
      int InterruptPresent;    /**< Are interrupts supported in h/w */
      int IsDual;       /**< Are 2 channels supported in h/w */
} abs_gradient_out_t;

XGpio output;
abs_gradient_out_t abs_module_0;
```

```
XIntc intc_0;

volatile u32 leds_state = 0;

void abs_gradient_interrupt_1(void){
	static int flag = 1;

	if(flag){
		flag = 0;
		leds_state |= 0x02UL;
		XGpio_DiscreteWrite(&output, 1, leds_state);
	}
	else{
		flag = 1;
		leds_state &= ~0x02UL;
		XGpio_DiscreteWrite(&output, 1, leds_state);
	}
}

int main(void){
	abs_module_0.BaseAddress = (volatile uint32_t)0x43C00000;
	abs_module_0.IsReady = 1;
	abs_module_0.IsDual = 0;
	abs_module_0.InterruptPresent = 1;

	XGpio_Initialize(&output, XPAR_AXI_GPIO_0_DEVICE_ID);
	XGpio_SetDataDirection(&output, 1, 0x0);
	XGpio_DiscreteWrite(&output, 1, 0x0f);

	XIntc_Initialize(&intc_0, XPAR_INTC_0_DEVICE_ID);
	XIntc_SelfTest(&intc_0);
	XIntc_Connect(&intc_0,
XPAR_MICROBLAZE_0_AXI_INTC_ABS_GRADIENT_OUT_0_INTERRUPT_1_INTR,
(XInterruptHandler)abs_gradient_interrupt_1, &abs_module_0);
	XIntc_Start(&intc_0, XIN_REAL_MODE);
	XIntc_Enable(&intc_0,
XPAR_MICROBLAZE_0_AXI_INTC_ABS_GRADIENT_OUT_0_INTERRUPT_1_INTR);

	Xil_ExceptionInit();
	Xil_ExceptionEnable();
	Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_INT,
(Xil_ExceptionHandler)XIntc_InterruptHandler, &intc_0);

	//Clear the AXI interrupt controller's pending flag
	XIntc_Acknowledge(&intc_0,
XPAR_MICROBLAZE_0_AXI_INTC_ABS_GRADIENT_OUT_0_INTERRUPT_1_INTR);

	usleep(1000000);

	XGpio_DiscreteWrite(&output, 1, 0x00);

	while(1){
		leds_state |= 0x04UL;
		leds_state &= ~0x08UL;
		XGpio_DiscreteWrite(&output, 1, leds_state);
		usleep(100000);
		leds_state &= ~0x04UL;
		leds_state |= 0x08UL;
		XGpio_DiscreteWrite(&output, 1, leds_state);
```

```
            usleep(100000);
      }

      return 0;
}
```

За ARM Cortex A9 програмата е:

```
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "sleep.h"

volatile uint32_t *abs_interrupt = ((volatile uint32_t *)0x43C00000)+11;

int main(){
    init_platform();

    print("Starting ...\n\r");

    while(1){
      print("Send interrupt\n\r");
      *abs_interrupt = 0x02;
      *abs_interrupt = 0x00;
      usleep(1000000);
    }

    cleanup_platform();

    return 0;
}
```

18. Заредете Линукс на ARM Cortex A9 и фърмуерът на MicroBlaze едновременно без да използвате Linux firmware опцията [1]. Това означава, че .elf файлът на MicroBlaze трябва да се внедри в bitstream файлът на системата [2]. За целта:

* върнете джъмпера JP5 в позиция SD;
* във Vivado → вляво → таб Sources → Design sources → десен бутон → Add sources → Add or create design sources → Next → Add files → укажете пътя до .elf файлът на MicroBlaze (/home/user/workspaces/vitis_workspace/16_linux_mb_app/Debug/16_linux_mb_app.elf) → махнете отметката copy sources into project → Finish;
* под design_1_wrapper.elf трябва да се появи нова категория ELF, а в нея трябва да се вижда 16_linux_mb_app.elf;
* във Vivado → вдясно → Diagram → десен бутон върху блок MicroBlaze → Associate ELF files … → Design sources → design_1 → microblaze_0 → вдясно има бутон с три точки "…" → избира се 16_linux_mb_app.elf (а не mb_bootloop_le.elf)→ OK → OK;
* във Vivado → вляво → Flow navigator → Generate bitstream → OK → Bitstream generation completed → View Reports → OK;

* във Vivado File → Export → Export hardware → Next → Include bitstream → Next → Next → An exported file for this module was found at this location. Do you want to overwrite it? → Yes → Finish;
* в терминала на PetaLinux →

```
petalinux-config
--get-hw-description=/home/user/workspaces/vivado_workspace/16_linux/
design_1_wrapper.xsa
```

дава се стрелка надясно в menuconfig и се избира Exit.
* в терминала на PetaLinux →

```
petalinux-build
```

Ако компилацията на Линукс първоначално е отнела 20 минути, то тази стъпка отнема около 3 минути. Копирайте и разархивирайте файловете **boot.scr**, **image.ub**, **rootfs.tar.gz** на SD картата, както преди, след което я върнете в Zybo платката. Рестартирайте или изключете и след това пак включете (power-cycle) платката. Ако всичко е минало успешно, би трябвало да се стартира Линукс и да се виждат мигащите светодиоди от MicroBlaze-а.

* * *

[1]https://github.com/bedik16/PetaLinux
[2]https://www.instructables.com/Flashing-a-MicroBlaze-Program/

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