



## Проектиране на вградени автомобилни електронни системи

### Лабораторно упражнение №9

Работа с Xilinx Vivado и Vitis. Синтезиране на микропроцесорна система върху FPGA. Работа с USB модул.

=====

1. Превключете джъмпера вдясно на платката на позиция JTAG. Свържете  $\mu$ USB кабел към PROG/UART USB куплунга. Включете платката от ключа ON/OFF.

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

```
source ~/programs/xilinx/Vivado/2020.2/settings64.sh
vivado
```

3. Create Project → Next → Project name: 09\_usb → Next → RTL Project + "Do not specify sources at this time" → Next → таб Boards: избира се Zybo (не Zybo Z7-10, не Zybo Z7-20, а само Zybo) → Next → Finish.

4. Вляво → Flow navigator → Create block design → OK.

5. Вдясно → Diagram → right-click → Add IP → Search → ZYNQ7 Processing System → double click.

6. Вдясно → Diagram → натиска се и се задържа ляв бутон върху FCLK\_CLK0 сигнала и се свързва с M\_AXI\_GP0\_ACLK, след това се пуска левия бутон.

7. Вдясно → Diagram → right-click → Add IP → Search → Processor System Reset → double click.

8. Вдясно → Diagram → зелена лента → Designer Assistance available → Run Block Automation → Слага се отметка на "All Automation".

9. Вдясно → Diagram → зелена лента → Designer Assistance available → Run Connection Automation → Слага се отметка на "All Automation". Натиска се OK.

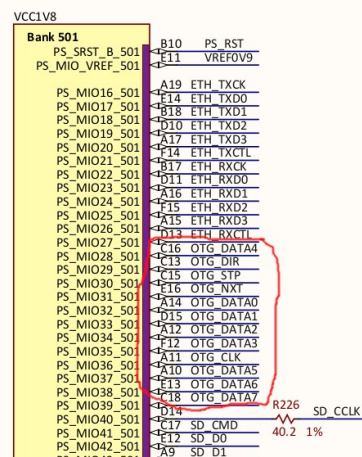
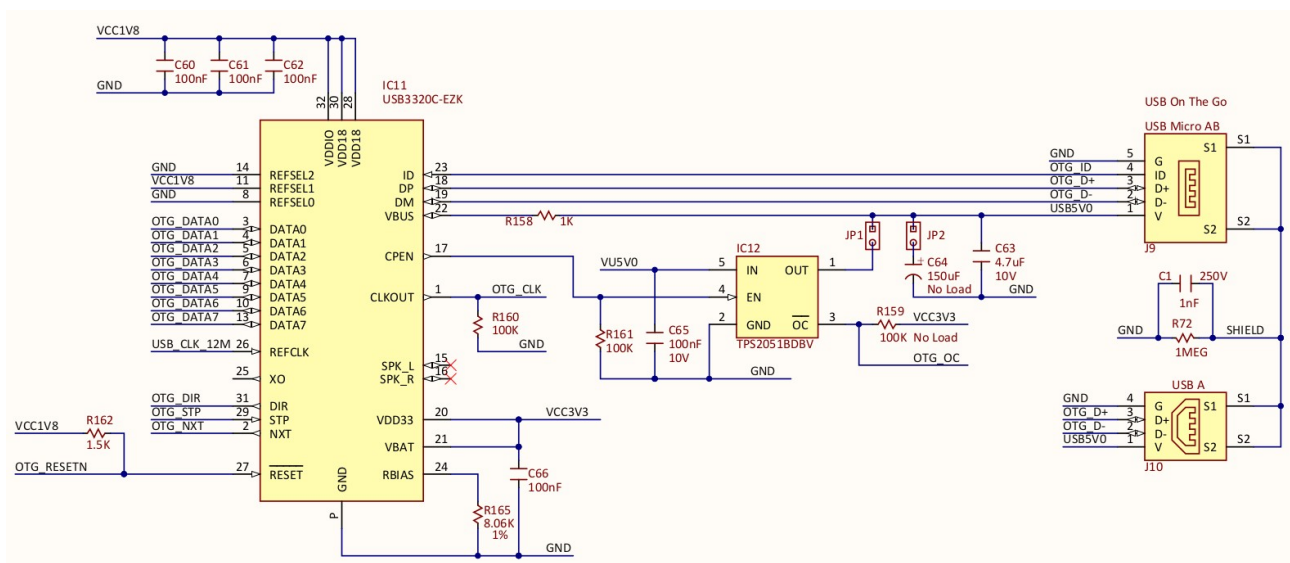
10. Щраква се два пъти върху блока "ZYNQ7 Processing System" → в "Page navigator" → MIO Configuration → I/O Peripherals → маха се отметката на → ENET0. Проверяват се връзките към USB PHY чипа:

MIO28 ↔ data[4]

MIO29 ↔ dir

MIO30 ↔ stp  
 MIO31 ↔ nxt  
 MIO32 ↔ data[0]  
 MIO33 ↔ data[1]  
 MIO34 ↔ data[2]  
 MIO35 ↔ data[3]  
 MIO36 ↔ clk  
 MIO37 ↔ data[5]  
 MIO38 ↔ data[6]  
 MIO39 ↔ data[7]

Забележете IC11 – USB3320C-EZK, който реализира физическия слой (PHY) на USB интерфейса.



11. В същия прозорец → “Page navigator” → MIO Configuration → Проверяват се връзките на UART1 интерфейса:

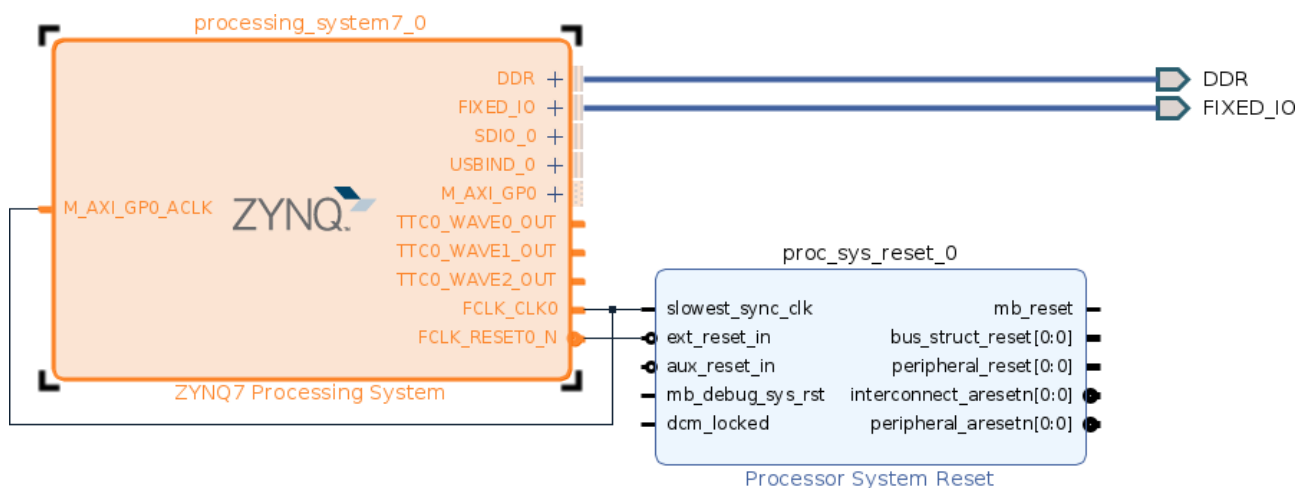
MIO48 ↔ tx  
MIO49 ↔ rx

12. Подрежда се блоковата схема с бутон Regenerate Layout.

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

14. 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

Блоковата схема на системата трябва да изглежда така:



15. 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 наблюдаватa 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.

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

=====

17. Tools → Launch Vitis IDE

18. Избира се път до workspace за фърмуерния проект → Launch

**ВНИМАНИЕ:** възможно е да има останали фърмуерни проекти от минали групи. В таб-а Explorer на средата Vitis със задържане на CTRL от клавиатурата изберете с ляв бутон на мишката всички проекти, след което натиснете десен бутон на мишката и Delete. Ако проектите ще се използват, махнете отметката от “Delete project contents on disk (cannot be undone)” и натиснете OK.

19. File → New → Platform project → Platform project name: 09\_usb → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 09\_usb, създаден от Vivado → design\_1\_wrapper.xsa → Open → Finish.

20. Вляво → Project explorer → избира се 09\_usb (Out-of-date) → right-click → Build Project.

21. File → New → Application project → Next → "Select a platform from repository" → Избира се 09\_usb → Next → Application project name: 09\_usb\_app → Next → Next → “Hello World” → Finish.

22. Щраква се двукратно с ляв бутон върху директорията src в проекта 09\_usb\_app\_system/09\_usb\_app → src → helloworld.c. Копирайте всички файлове от архива 00\_usb\_lib.zip, наличен в директорията на настоящото упражнение, в src директорията на проекта 09\_usb\_app.

23. В текстовия редактор на Vitis се въвежда следната програма [1]:

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

#include "xparameters.h"
#include "xusbps.h"
#include "xscugic.h"
#include "xusbps_cdc.h"

XScuGic INTC;
XUsbPs USB0;

static void reset_usb(void) {
    // Ensure that the PHY is out of reset
    volatile u32 *gpio_base;
    volatile u32 *gpio_oen;
    volatile u32 *gpio_dir;

    /* Ensure that the USB PHY is not in reset */
    gpio_base = (u32 *)0xE000A000;
    gpio_oen = (u32 *)0xE000A208;
    gpio_dir = (u32 *)0xE000A204;

    *(gpio_oen) |= 0x00000080;
```

```

        *(gpio_dir) |= 0x000000080;
        *gpio_base = 0xff7f0080;
    }

    int main(){
        u8 read_buffer[256];
        u32 read_size;
        XScuGic_Config *intc_config;

        init_platform();

        print("Starting the USB example ...\n\r");

        reset_usb();

        intc_config = XScuGic_LookupConfig(XPAR_SCUGIC_SINGLE_DEVICE_ID);
        XScuGic_CfgInitialize(&INTC, intc_config, intc_config->CpuBaseAddress);
        Xil_ExceptionInit();
        Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_IRQ_INT,
(Xil_ExceptionHandler)XScuGic_InterruptHandler, &INTC);
        XScuGic_Connect(&INTC, XPAR_PS7_USB_0_INTR,
(Xil_ExceptionHandler)XUsbPs_IntrHandler, (void *)&USB0);
        XScuGic_Enable(&INTC, XPAR_PS7_USB_0_INTR);
        Xil_ExceptionEnableMask(XIL_EXCEPTION_IRQ);

        xusb_cdc_init(&USB0, XPAR_PS7_USB_0_DEVICE_ID, XPAR_PS7_USB_0_INTR, 64 *
1024);

        while(1){
            read_size = xusb_cdc_rx_bytes_available();
            if (read_size != 0) {
                xusb_cdc_receive_data(read_buffer, read_size);
                read_buffer[read_size] = '\0';
                read_size = 0;
                xil_printf("%s\n", read_buffer);
            }
        }

        cleanup_platform();

        return 0;
    }

```

29. Вляво, Project explorer → избира се 09\_usb\_app\_system → right-click → Build project.

30. Вляво, Project explorer → избира се 09\_usb\_app\_system → right-click → Debug as → Launch Hardware.

31. Свържете микро USB кабел към куплунга J9 на Zybo от едната страна и USB type A към компютър с Linux (Ubuntu) от другата. Използва се USB 2.0.

32. Отваря се терминал в Ubuntu с CTRL + ALT + T → Пише се ls /dev/tty и се натиска tab → "Display all 100 possibilities? (y or n)" въвежда се 'y' → **търси се системния файл, отговарящ на виртуалния RS232 порт** за дебъг съобщения

(обикновено ttyUSB1, **ВНИМАНИЕ** на ttyUSB0 излиза виртуален порт за JTAG дебъгера, който не трябва да бъде отварян).

След като се види номера на виртуалния порт, в същия терминал се стартира RS232 терминал чрез командата:

`cutecom`

33. В cutecom → Device: избира се съответния порт за дебъг съобщения /dev/ttyUSBx → Settings → 115200-8-N-1, no flow control -> Open

34. Във Vitis: натиска се бутон Resume (F8). След това в Cutecom трябва да се изпише:

[20:42:00:909] Starting the USB example ...<sup>L\_F</sup>

[20:42:00:909] <sup>C\_R</sup>

35. Отворете терминал с CTRL + ALT + t и се стартира втори път командата:

`cutecom`

36. Отворете терминал с CTRL + ALT + t и напишете командата:

`dmesg`

Ако USB работи правилно, би трябвало да се види регистрирането на CDC устройство от ядрото на Ubuntu:

[ 6197.480997] usb 1-3: New USB device found, idVendor=03fd, idProduct=0100, bcdDevice= 1.00

[ 6197.481009] usb 1-3: New USB device strings: Mfr=1, Product=2, SerialNumber=3

[ 6197.481015] usb 1-3: Product: CDC ACM Driver

[ 6197.481019] usb 1-3: Manufacturer: Xilinx

[ 6197.481023] usb 1-3: SerialNumber: 2A49876D9CC1AA4

[ 6197.486290] cdc\_acm 1-3:1.0: **tttACM0**: USB ACM device

**ВНИМАНИЕ:** при нова инсталация на Ubuntu Linux, трябва да се добави настоящия потребител към списъка “dialout”, иначе няма да се създаде файлът за виртуалния UART. Това може да стане със следната команда:

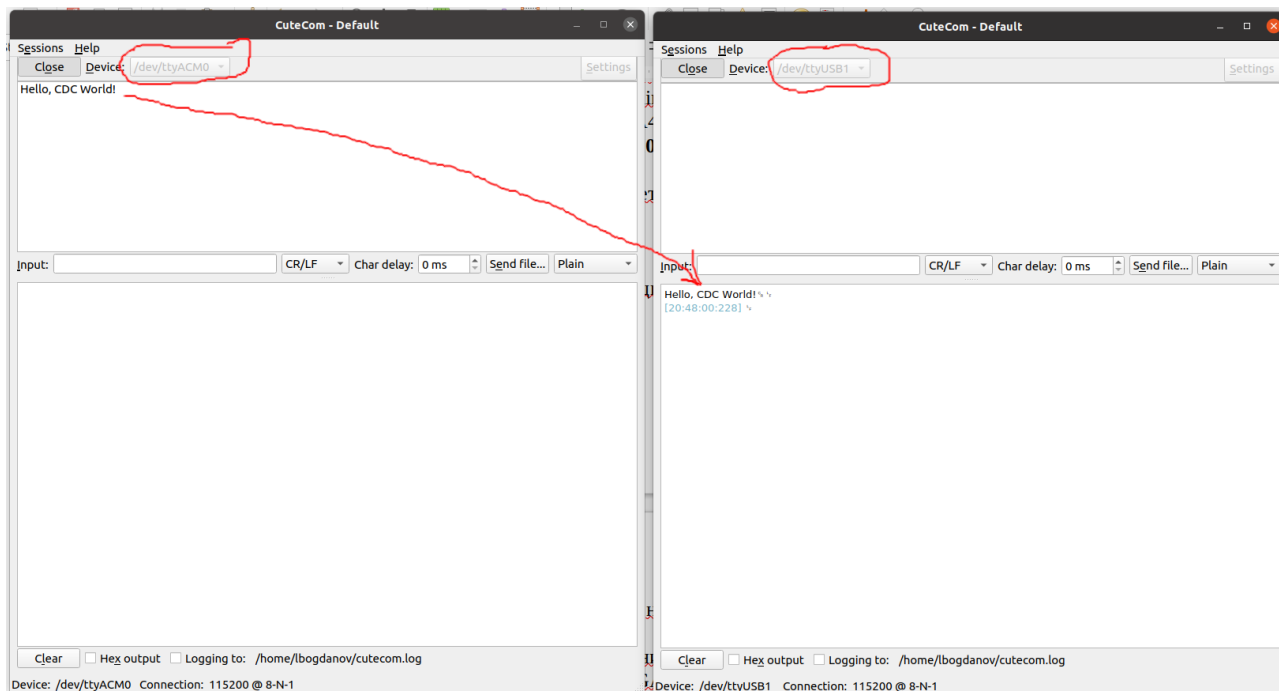
`sudo usermod -a -G dialout $USER`

37. Във втория cutescom терминал отворете връзка:

Device: /dev/ttyACM0

Settings: 115200-8-N-1

и напишете текст в полето Input. Същият този текст трябва да се покаже в първия cutescom терминал.



38. За да спрете debug сесията във Vitis, натиснете Disconnect.

39. Добавете код към примера, така че да съществува и обратната връзка Zybo виртуален терминал → CDC виртуален терминал. Ще се наложи сами да си напишете неблокираща функция за четене на символ от UART-а, аналогична на XUartPs\_RecvByte( ) във файла:

09\_usb/ps7\_cortexa9\_0/standalone\_domain/bsp/ps7\_cortexa9\_0/libsrc/  
uartps\_v3\_10/src/xuartps\_hw.c

\*

\*

\*

[1] <https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841624/Zynq-7000+AP+SoC+USB+CDC+Device+Class+Design+Example+Techtip#Zynq-7000APSoCUSB CDC Device Class Design Example Techtip-Applicationsoftware%3A>

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