



Проектиране на вградени автомобилни електронни системи

Лабораторно упражнение №8

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

=====

1. Превключете джъмпера вдясно на платката на позиция JTAG. Свържете μ 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: 08_eth → 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.

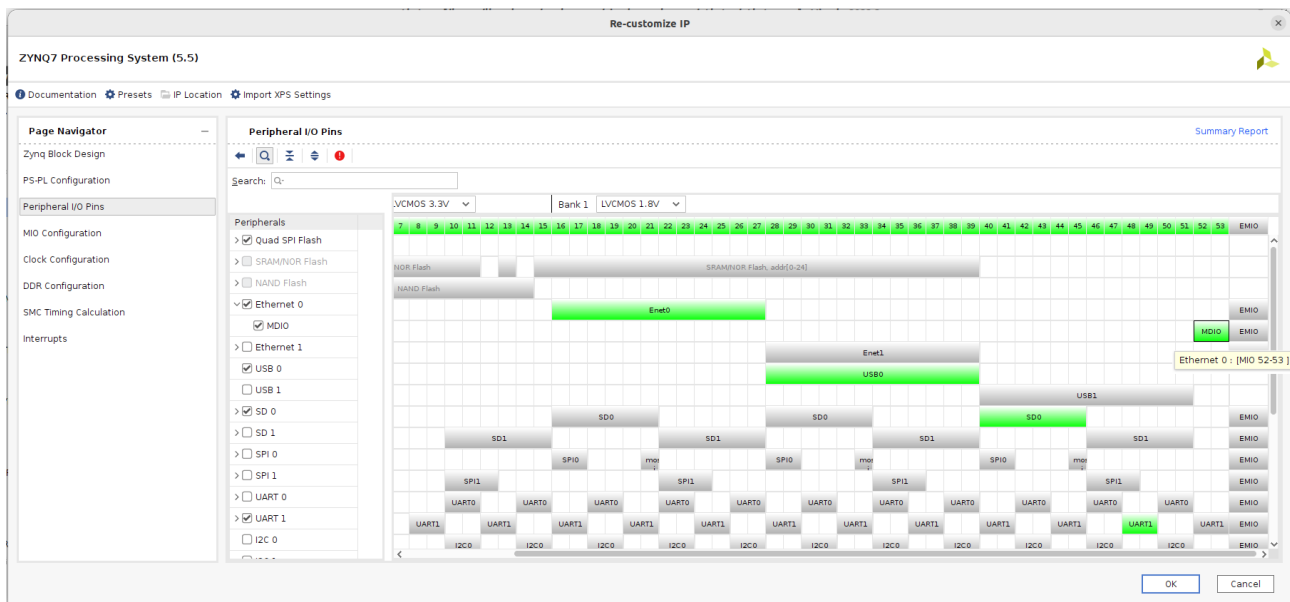
6. Вдясно → Diagram → натиска се и се задържа ляв бутон върху FCLK_CLK0 сигнала и се свързва с M_AXI_GP0_ACLK, след това се пуска левия бутон.

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" → Peripheral I/O pins → натиска се стрелката на Ethernet 0 → проследява се редът на MDIO → вдясно е избрано EMIO → **избира се MDIO** → **OK**.



В “Page navigator” → MIO Configuration → I/O Peripherals → маха се отметката на → USB0. Проверяват се връзките на MII интерфейса:

MIO52 ↔ mdc

MIO53 ↔ mdio

MIO16 ↔ tx_clk

MIO17 ↔ txd[0]

MIO18 ↔ txd[1]

MIO19 ↔ txd[2]

MIO20 ↔ txd[3]

MIO21 ↔ tx_ctl

MIO22 ↔ rx_clk

MIO23 ↔ rxd[0]

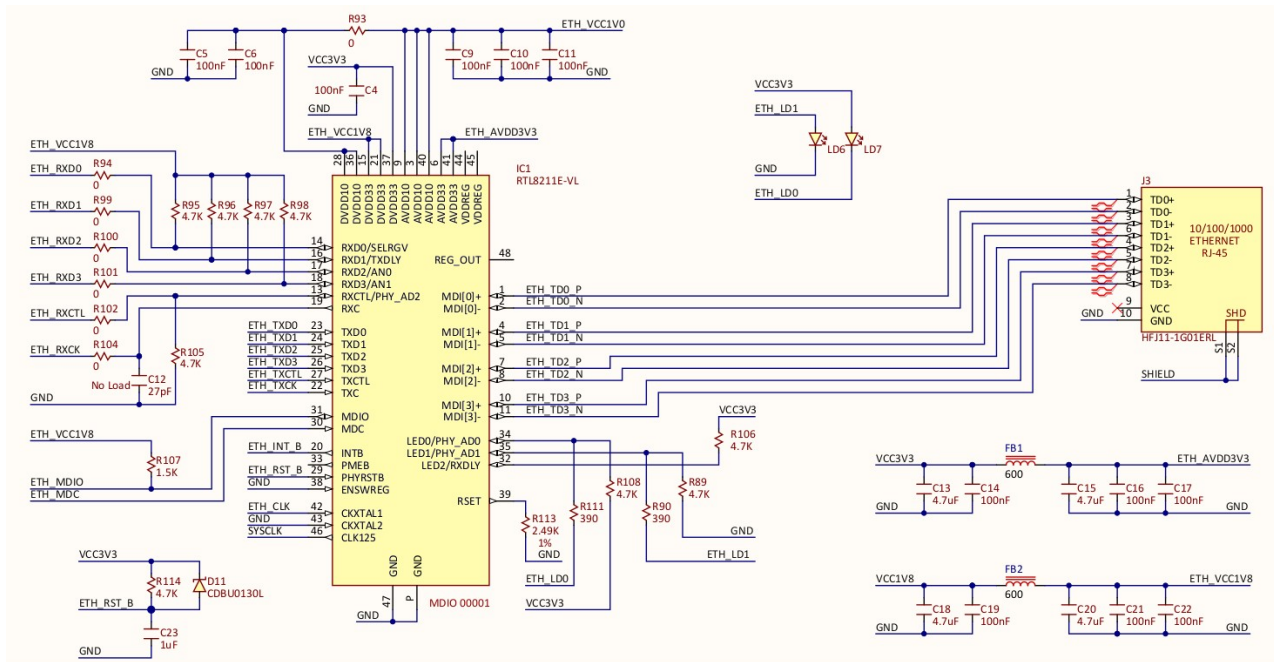
MIO24 ↔ rxd[1]

MIO25 ↔ rxd[2]

MIO26 ↔ rxd[3]

MIO27 ↔ rx_ctl

Забележете IC1 – RTL8211E-VL, който реализира физическия слой (PHY) на Етернет интерфейса. Той включва транслиране и кодиране на сигналите, разбъркване на данните (scramble) и договаряне на връзката.



VCC1V8	
Bank 501	
PS_SRST_B_501	B10 PS_RST
PS_MIO_VREF_501	F11 VREF0V9
PS_MIO16_501	A19 ETH_TXCK
PS_MIO17_501	F14 ETH_TXD0
PS_MIO18_501	B18 ETH_TXD1
PS_MIO19_501	D10 ETH_TXD2
PS_MIO20_501	A17 ETH_TXD3
PS_MIO21_501	F14 ETH_TXCTL
PS_MIO22_501	B17 ETH_RXCK
PS_MIO23_501	D11 ETH_RXD0
PS_MIO24_501	A16 ETH_RXD1
PS_MIO25_501	F15 ETH_RXD2
PS_MIO26_501	A15 ETH_RXD3
PS_MIO27_501	D13 ETH_RXCTL
PS_MIO28_501	C16 OTG_DATA4
PS_MIO29_501	C13 OTG_DIR
PS_MIO30_501	C15 OTG_STP
PS_MIO31_501	F16 OTG_NXT
PS_MIO32_501	A14 OTG_DATA0
PS_MIO33_501	D15 OTG_DATA1
PS_MIO34_501	A12 OTG_DATA2
PS_MIO35_501	F12 OTG_DATA3
PS_MIO36_501	A11 OTG_CLK
PS_MIO37_501	A10 OTG_DATA5
PS_MIO38_501	F13 OTG_DATA6

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

MIO48 ↔ tx

MIO49 ↔ rx

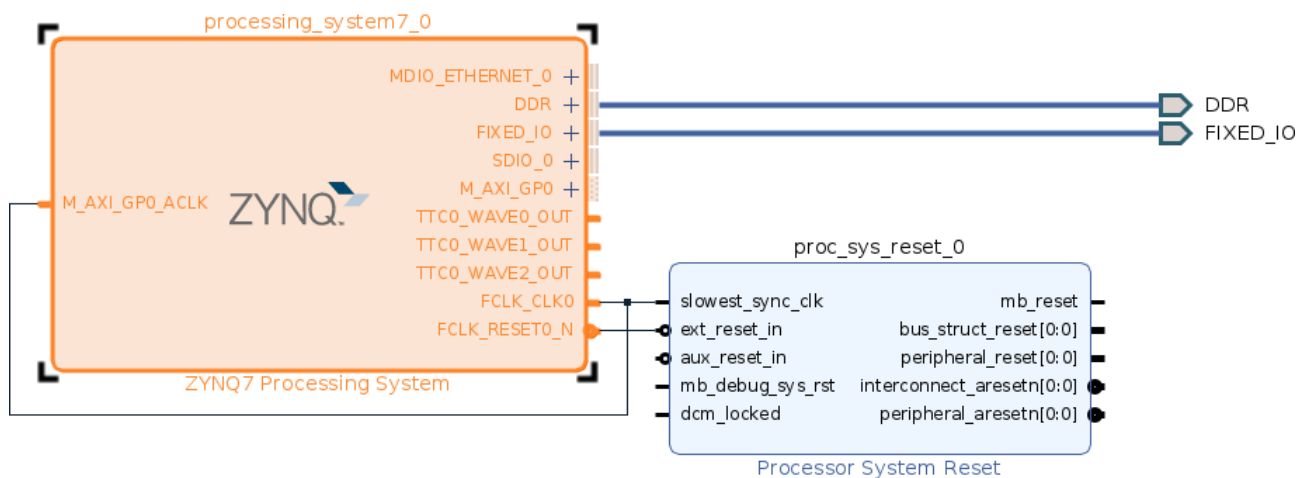
12. Подрежда се блоковата схема с бутон Regenerate Layout.

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 наблюдават 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: 08_eth_pla → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 08_eth, създаден от Vivado → design_1_wrapper.xsa → Open → Finish.

20. Ще се използва библиотеката с отворен сорс код lwIP, която реализира TCP/IP протокола. Тази библиотека не е добавена по подразбиране към проекта. Затова: вляво на средата до таб Explorer има друга таб, Assistant → избира се проекта 08_eth_pla [Platform] → десен бутон → Open Platform Editor → избира се “standalone on ps7_cortexa9_0 → Board Support Package → централно ще се появи таб с бутон “Modify BSP Settings” → натиска се този бутон → слага се отметка на Supported libraries / lwip211.

21. В същия прозорец отдясно в графа Overview трябва да се появи standalone/lwip211 → dhcp_options → lwip_dhcp → Value = true.

ВНИМАНИЕ: ако “autonegotiation link speed” не минава успешно, трябва в този прозорец да изберете темас_adapter_options/phy_link_speed → Value = 1000 Mbps [1] → ОК.

ВНИМАНИЕ: ако “autonegotiation link speed” не минава успешно, и ако рутерът ви е с 100-мегабитови портове, изберете phy_link_speed = 100 → ОК.

22. Отворете във Vitis файла:

```
workspace_vitis/08_eth_pla/export/08_eth_pla/sw/08_eth_pla/standalone_domain/  
bspinclude/include/lwipopts.h
```

и проверете дали макроса LWIP_DHCP е равен на 1:

```
#define LWIP_DHCP 1
```

23. Проверете дали макроса CONFIG_LINKSPEED_AUTODETECT в

```
workspace_vitis/08_eth_pla/export/08_eth_pla/sw/08_eth_pla/standalone_domain/  
bspinclude/include/lwipopts.h
```

е равен на 1:

```
#define CONFIG_LINKSPEED_AUTODETECT 1
```

24. Вляво → Project explorer → избира се 08_eth_pla (Out-of-date) → right-click → Build Project.

25. File → New → Application project → Next → "Select a platform from repository" → Избира се 08_eth → Next → Application project name: 08_eth_app → Next → Next → “lwIP Echo Server” → Finish.

26. Щраква се двукратно с ляв бутон върху директорията src в проекта 08_eth_app_system/08_eth_app → src → echo.c

27. В текстовия редактор на Vitis ще има заредена примерна програма, използваща socket-и и TCP/IP комуникация, за да направи ехо на TCP/IP терминал.

28. Очаква се ехо сървърът да има **IP дадено от рутера** (вижте UART терминала) и **порт номер 7**.

29. Вляво, Project explorer → избира се 08_eth_app_system → right-click → Build project.

30. Вляво, Project explorer → избира се 08_eth_app_system → right-click → Debug as → Launch Hardware.

31. Свържете LAN кабел към куплунга J3 на Zybo от едната страна и мрежа с DHCP сървър от другата страна (рутер).

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:

[12:27:07:041] c_R-----lwIP TCP echo server ----- l_F

[12:27:07:041] c_R TCP packets sent to port 6001 will be echoed back l_F

[12:27:07:057] c_R Start PHY autonegotiation c_R l_F

[12:27:07:057] Waiting for PHY to complete autonegotiation. c_R l_F

[12:27:11:080] autonegotiation complete c_R l_F

[12:27:11:080] link speed for phy address 1: 1000 c_R l_F

[12:27:11:080] unable to determine type of EMAC with baseaddress 0xE000B000 c_R l_F

[12:27:12:257] Board IP: 192.168.0.108 l_F

[12:27:12:257] c_R Netmask : 255.255.255.0 l_F

[12:27:12:257] ^{C_R} Gateway : 192.168.0.1 ^{L_F}

[12:27:12:257] ^{C_R} TCP echo server started @ port 7 ^{L_F}

[12:27:12:257] ^{C_R}

35. Отворете терминал с CTRL + ALT + t и напишете команда за стартиране на TCP/IP терминал, например:

```
telnet 192.168.0.108 7
```

след което въведете произволен текст. Ако примерът се е заредил успешно, трябва да виждате буквите на изречението, което пишете (ехо). С Enter затваряте socket-a.

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

37. Припомнете си ученическите години и напишете уеб страница на HTML, която да се поддържа от сървър на FPGA. Опитайте да достъпите страницата с уеб браузър от локалната мрежа (от външни IP-та няма да стане, защото портовете са затворени). Напомняне: уеб-браузърите използват **порт 80** и изобразяват низовете на HTML чак след **като се затвори socket-a** на връзката.



[1] <https://digilent.com/reference/learn/programmable-logic/tutorials/zybo-getting-started-with-zynq-server/start>

[2] <https://github.com/Xilinx/embeddedsw/issues/70>

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