Проектиране на вградени автомобилни електронни системи
Лабораторно упражнение №3

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

==========

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

2. Стартирайте терминал с CTRL + ALT + T и изпълнете командите [1]:

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

3. Create Project → Next → Project name: 03_uart → 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

8. 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.

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

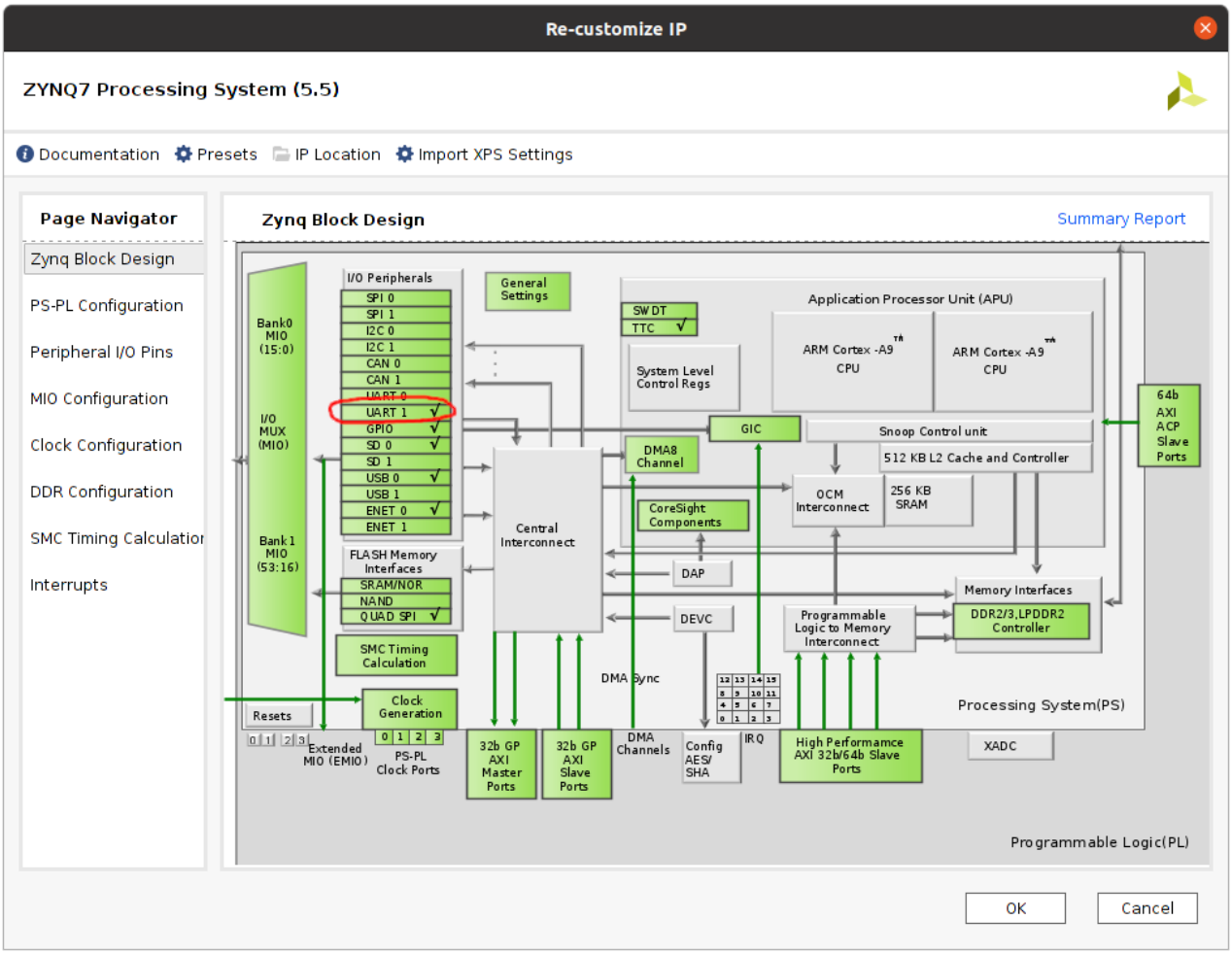
Ще се използва UART модула, който е от микроконтролерния блок на матрицата (наречен PS – Processing System, а FPGA блокът е наречен PL – Programmable Logic). Това означава, че не се налага добавянето на външен IP блок, освен блока за рестарт.

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

11. Вдясно →Diagram → зелена лента → Designer Assitance available -> Run Connection Automation → Слага се отметка на "All Automation".
Натиска се OK.

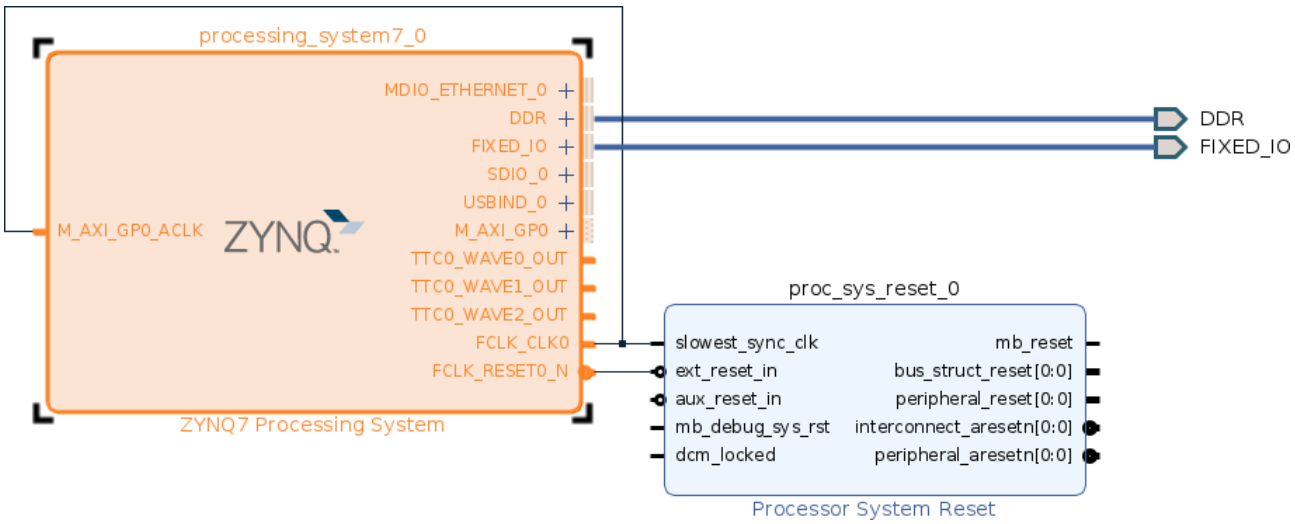
12. Подрежда се блоковата схема с бутон Regenerate Layout.

13. Щраква се двукратно върху блока ZYNQ Processing System. Трябва поне един UART да е избран:



Проверява се дали сигналите на UART блока са свързани към правилните изводи. Това са MIO48 и MIO49, които са свързани към виртуалния сериен порт на дебъгера и не се нуждаят от USB-към-UART конвертор. Сигналите може да се видят от
таб MIO Configuration → I/O Peripherals → UART1 (или UART0) → MIO48 ↔ tx и MIO49 ↔ rx → Натиска се OK.

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



15. 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

16. 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.

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

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

18. Tools → Launch Vitis IDE

19. 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.

20. File → New → Platform project → Platform project name: 03_uart → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 03_uart, създаден от Vivado → design_1_wrapper.xsa → Open → Finish

21. Вляво → Project explorer → избира се 03_uart → right-click → Build Project

22. File → New → Application project → Next → "Select a platform from repository" → Избира се 03_uart → Next → Application project name: 03_uart_app → Next → Next → “Hello World” → Finish

23. Щраква се двукратно с ляв бутон върху директорията src в проекта 03_uart_app_system/03_uart_app → src → helloworld.c

24. В текстовия редактор на Vitis трябва да има готова следната програма [1]:

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


int main()
{
    init_platform();

    print("Hello World\n\r");
    print("Successfully ran Hello World application");
    cleanup_platform();
    return 0;
}
```

25. Вляво, Project explorer -> избира се 03_uart_app_system -> right-click -> Build project

26. Вляво, Project explorer -> избира се 03_uart_app_system -> right-click -> Debug as -> Launch Hardware

27. 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

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

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

Hello World
Successfully ran Hello World application

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

31. Напишете програма, която прави “exo” в терминала, т.е. всеки един въведен символ да се изпраща обратно в терминала. Използвайте F3, за да обходите системните библиотеки и да намерите функцията, която изпраща байтове по UART и функцията, която приема байтове от UART.

32. Напишете програма, която приема команди от RS232 терминала. Нека командите са следните:

hello\n - Изписва низът “Hello\n”
world\n - Изписва низът “World\n”

Използвайте printf и scanf.

* * *

гл. ас. д-р инж. Любомир Богданов, 2021 г.