Проектиране на вградени автомобилни електронни системи
Лабораторно упражнение №11

Работа с Xilinx Vivado и Vitis. Синтезиране на многопроцесорна система върху FPGA.

==========

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

2. От страничния панел на Ubuntu изберете бутон “Show Applications”, след което в полето “Type to search” напишете Vivado и натиснете с ляв бутон на мишката иконката на програмата.

3. Create Project → Next → Project name: 11_multiproc → 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.

ЗАБЕЛЕЖКА: работната маса с платка Zybo Z7-10 (вдясно на Етернет куплунга трябва да има 2 HDMI конектора; ако има един HDMI и един VGA значи, че е само Zybo) трябва да избере Zybo Z7-10 от това меню.

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 Assitance available → Run Block Automation → Слага се отметка на "All Automation".

9. Вдясно → Diagram → right-click → Add IP → Search → AXI GPIO → double click.

10. Щракнете два пъти върху блока AXI GPIO → Board→ IP Interface: GPIO → Board Interface: leds 4 bits → OK.

11. Вдясно → Diagram → right-click → Add IP → Search → AXI Interconnect → double click.
12. Щракнете два пъти върху блока AXI Interconnect → Number of Slave Interfaces = 1 → Number of Master Interfaces = 1 → OK [1].

13. Вдясно → Diagram → right-click → Add IP → Search → MicroBlaze → double click.

14. Вдясно →Diagram → зелена лента → Designer Assitance available -> Run Block Automation → Слага се отметка на "microblaze_xx". В полето Options се избира Local memory: 64 kB и Cache configuration: None. Натиска се OK.

15. Щракнете два пъти върху блока MicroBlaze → Predefined configurations → Select Configuration: Microcontoller preset → OK. (На микропроцесорът MicroBlaze трябва да се появи M_AXI_DP порт)

16. Натиска се и се задържа ляв бутон върху M_AXI_DP порта и се свързва с порта S00_AXI на AXI Interconnect блокът, след това се пуска левия бутон. Аналогично се свързва портът M00_AXI с порта S_AXI на AXI GPIO модула.

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

18. Щраква се два пъти върху блока “ZYNQ7 Processing System” → в “Page navigator” → MIO Configuration →в раздел I/O Peripherals → UART1 се проверяват връзките MIO48 ↔ tx, MIO49 ↔ rx.

19. В същия прозорец → “Page navigator” → MIO Configuration → маха се отметката на I/O Peripherals → ENET0, USB0 и SD0. Натиска се OK.

20. В основния прозорец на Vivado, до таб Diagram, се избира Address Editor → натиска се бутон Assign All. Тази стъпка разполага периферните модули на системата в адресното поле на съответните микропроцесори. Ако е необходимо, тези адреси могат да се зададат ръчно от проектанта, като се спазва условието да не се застъпват.

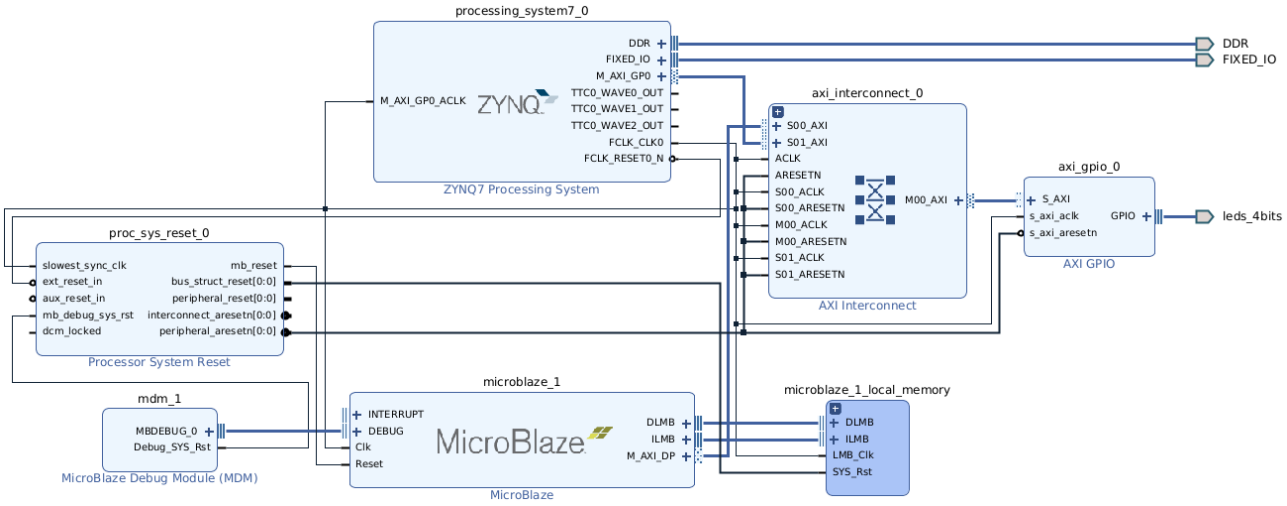
21. Подрежда се блоковата схема с бутон Regenerate Layout.

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

23. Централно→в Block design прозореца, натиска се таб-а Sources → Design sources → right-click на design_1.bd → Create HDL Wrapper (създава Verilog

описание на новосъздадената система) → Let Vivado manage wrapper and auto-update → OK

Блоковата схема на системата трябва да изглежда така:



24. 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.

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

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

26. От страничния панел на Ubuntu изберете бутон “Show Applications”, след което в полето “Type to search” напишете Vitis Classic и натиснете с ляв бутон на мишката иконката на програмата.

27. 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.

28. File → New → Platform project → Platform project name: 11_multi_proc_mb_pla → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 11_multi_proc, създаден от Vivado → design_1_wrapper.xsa → Open → Поле Software specification → Operating system: standalone (това означава bare-metal firmware) и Processor: microblaze_0 → Finish.

29. Вляво → Project explorer → избира се 11_multi_proc_mb_pla → right-click → Build Project.

30. File → New → Application project → Next → "Select a platform from repository" → Избира се 11_multi_proc_mb_pla → Next → Application project name: 11_multi_proc_mb_app → Next → Next → “Empty application (C)” → Finish.

31. Вляво в таб Explorer → отваря се 11_multi_proc_mb_app_system / 11_multi_proc_mb_app / src → върху директорията src се натиска десен бутон → New → Other → C/C++ → Source File → Next → Source file: дава се име на файлът main.c → Finish.

32. В текстовия редактор на Vitis и във файлът main.c на MicroBlaze се въвежда следната програма:

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

int main(void){
	XGpio output;

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

	while(1){
		XGpio_DiscreteWrite(&output, 1, 0x00);
		usleep(200000);
		XGpio_DiscreteWrite(&output, 1, 0x01);
		usleep(200000);
	}

	return 0;
}
```

33. Вляво → Project explorer → избира се 11_multi_proc_mb_app → right-click → Build Project.

34. Вляво → Project explorer → избира се 11_multi_proc_mb_app_system → right-click → Build Project.

35. File → New → Platform project → Platform project name: 11_multi_proc_cortex_pla → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 11_multi_proc, създаден от Vivado → design_1_wrapper.xsa → Open → Поле Software specification → Operating system: standalone и Processor: ps7_cortexa9_0 → Finish.

36. Вляво → Project explorer → избира се 11_multi_proc_cortex_pla → right-click → Build Project.

37. File → New → Application project → Next → "Select a platform from repository" → Избира се 11_multi_proc_cortex_pla → Next → Application project name: 11_multi_proc_cortex_app → Next → Next → “Hello World” → Finish.

38. Щраква се двукратно с ляв бутон върху директорията src в проекта 11_multi_proc_cortex_app_system/ 11_multi_proc_cortex_app → src → helloworld.c

39. В текстовия редактор на Vitis и във файлът main.c на ARM Cortex A9 се въвежда следната програма

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

int main(){
    init_platform();

    while(1){
      print("Running on ARM Cortex A9 ...\n\r");
      usleep(1000000);
    }

    cleanup_platform();

    return 0;
}
```

40. Вляво, Project explorer → избира се 11_multi_proc_cortex_app → right-click → Build project.

41. Вляво, Project explorer → избира се 11_multi_proc_cortex_app_system → right-click → Build project.

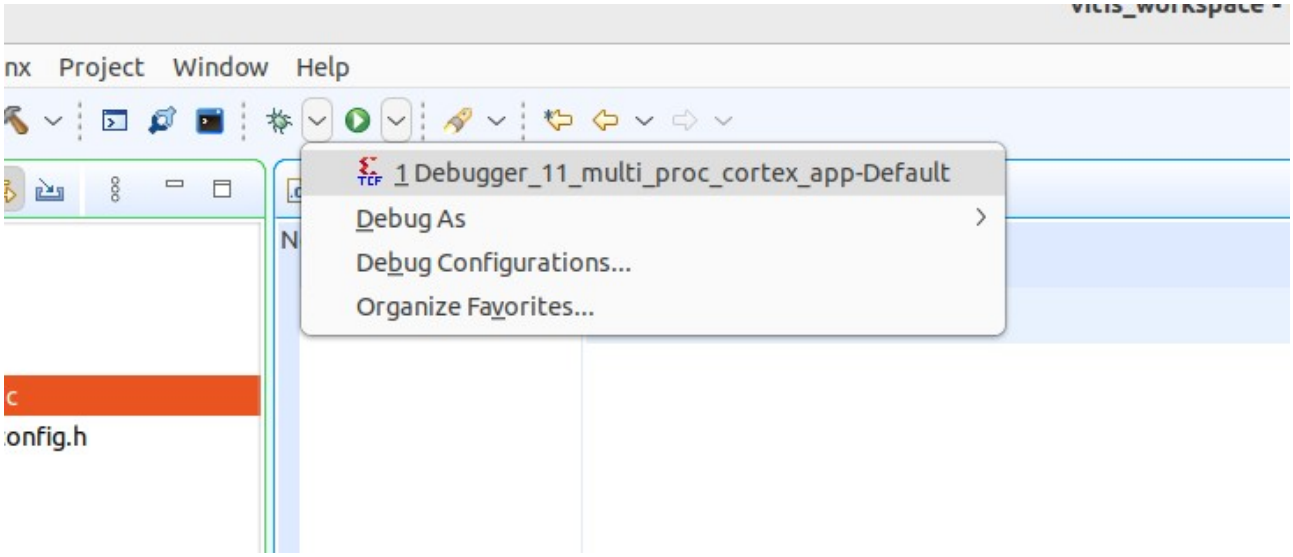
42. В основния прозорец на Vitis до бутонът Debug има стрелка надолу → натиска се → Debug configurations … → щраква се двукратно върху Single

Application Debug → вдясно ще се появи нова конфигурация на дебъг сесия. Избира се таб Application и се слагат отметки на двата процесора: microblaze_0 и ps7_cortexa9_0. Проверяват се полетата Application, указващи фърмуерния .elf файл за всеки процесор. Полето на MicroBlaze може да бъде празно. Ако това е така, с ляв бутон в полето Summary се избира целият ред на microblaze_0 и се натиска бутон Search срещу полето Application. Средата Vitis ще предложи всички .elf файлове, които са достъпни. С ляв бутон се натиска двукратно върху съответния файл на MicroBlaze (11_multi_proc_mb_app.elf). Сега в полетата Application на Summary трябва да се вижда:

microblaze_0 Debug/11_multi_proc_mb_app.elf
ps7_cortexa9_0 Debug/11_multi_proc_cortex_app.elf

Натиска се Apply → Debug

ВНИМАНИЕ: Всяко следващо стартиране на Debug сесия може да стане с бутон надолу до Debug бутона от основния прозорец на Vitis, при условие, че поне веднъж е била стартирана дебъг сесия от Debug configurations… прозореца (в конкретния случай това е станало, когато сме натиснали Apply → Debug).



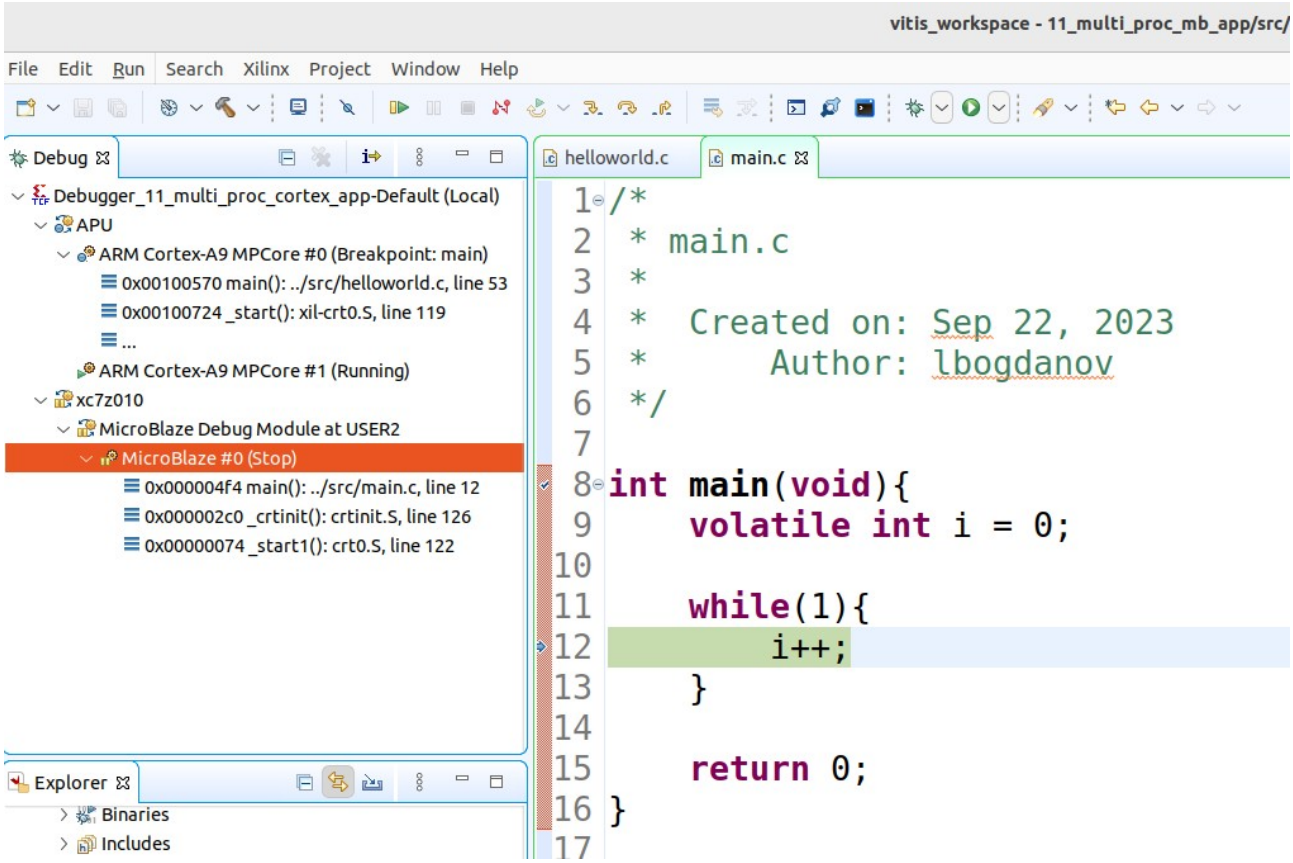
ВНИМАНИЕ: не трябва да се натиска самият бутон Debug понеже това създава нова дебъг сесия, която по подразбиране зарежда фърмуер само на едно Cortex A9 ядро.

ВНИМАНИЕ: при промяна на сорс кода трябва да се натисне Build на фърмуерния проект (_app) и на системния проект (_app_system), иначе дебъг сесията ще зареди старата версия на .elf файлът.

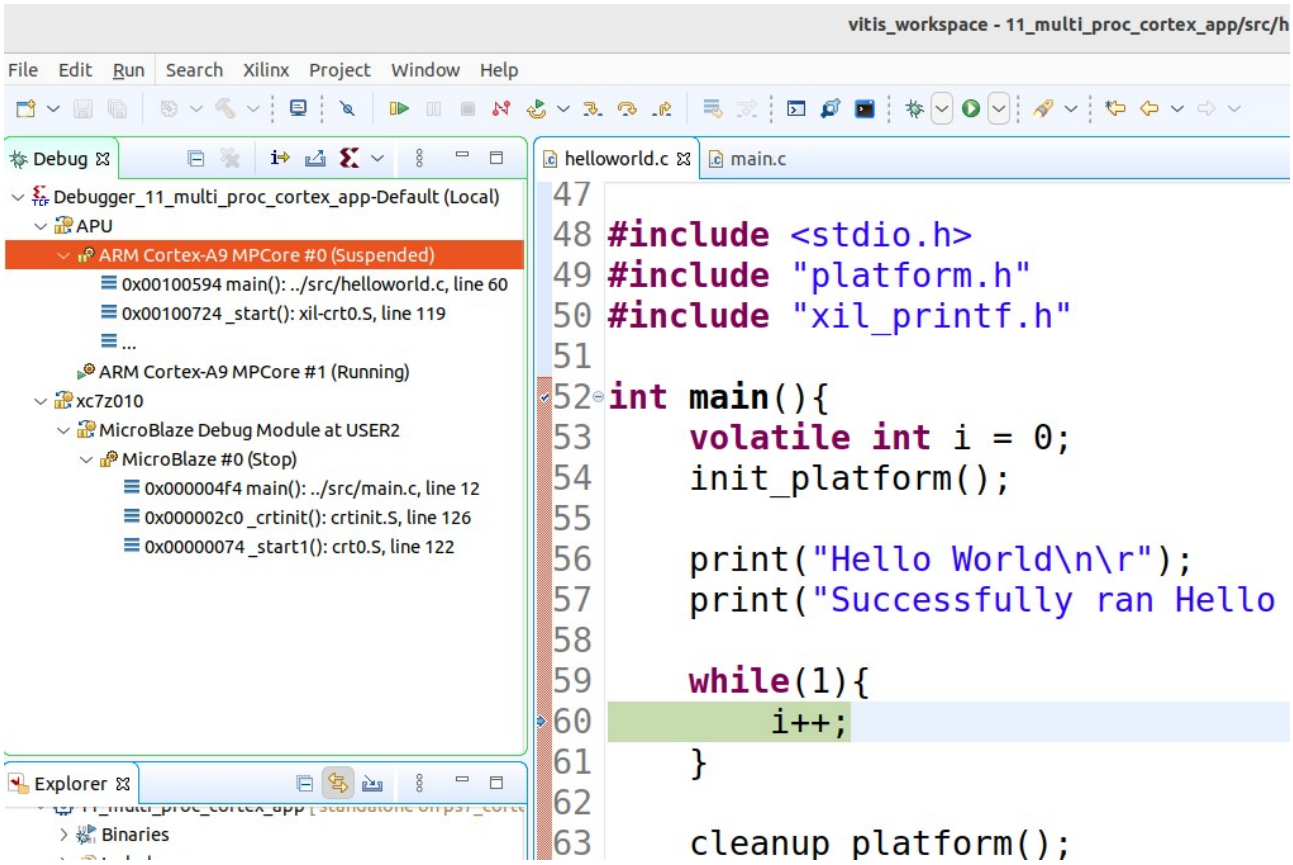
43. Дебъгването на отделните микропроцесори става като се избере с ляв бутон

съответния процесор от таб Debug. Дебъг бутоните и всички дебъг табове се присвояват автоматично на избрания процесор, т.е. въпреки че процесорите са два, наборът от дебъг инструменти е един.

На фигурата по-долу е показано как се дебъгва фърмуерът на MicroBlaze:



На фигурата по-долу е показано как се дебъгва фърмуерът на ARM Cortex A9 (ядро 0):



44. 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
```

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

46. Във Vitis: натиска се бутон Resume (F8) за Cortex A9 (ядро 0). След това в Cutecom трябва да се изпише:

```
Running on ARM Cortex A9 ...
Running on ARM Cortex A9 ...
```

Running on ARM Cortex A9 ...

Running on ARM Cortex A9 ...

Running on ARM Cortex A9 ...

47. Във Vitis: натиска се бутон Resume (F8) за MicroBlaze. След това трябва да започне да мига светодиод LD0 на демо платката Zybo.

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

49. Напишете програма, която мига всички светодиоди на платката (LD0 - LD1).

* * *

[1] Adam Tylor, “MicroZed Chronicles: Inter Processor Communication (Part 1)”, online, https://medium.com/@aptaylorceng/microzed-chronicles-inter-processor-communication-part-1-c1411c1c3053, 2023.

доц. д-р инж. Любомир Богданов, 2024 г.