Проектиране на вградени автомобилни електронни системи
Лабораторно упражнение №12

Работа с Xilinx Vivado и Vitis. Междупроцесорна комуникация с помощта на пощенска кутия (mailbox).

==========

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:

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

3. Create Project → Next → Project name: 12_mailbox → 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 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 = 3 → 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_0". 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. В полето General Settings се слага отметка на Enable Exceptions.

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 → right-click → Add IP → Search → Mailbox → double click [2].

18. Натиска се и се задържа ляв бутон върху M01_AXI порта на AXI Interconnect блокът и се свързва с порта S00_AXI на mailbox_0 блока, след това се пуска левия бутон.

19. Вдясно → Diagram → right-click → Add IP → Search → AXI Interconnect → double click.

20. Щракнете два пъти върху блока AXI Interconnect → Number of Slave Interfaces = 1 → Number of Master Interfaces = 1 → OK.

21. Натиска се и се задържа ляв бутон върху M_AXI_GP0 порта на Zynq блока и се свързва към токуо-що добавения AXI Interconnect на порт S00_AXI, след това се пуска левия бутон. Аналогично се свързва порт M00_AXI на AXI Interconnect към порт S1_AXI на mailbox_0.

22. За да е ефективна комуникацията между процесорите, трябва да се използват прекъсвания, показващи кога във FIFO буферът на mailbox_0 е постъпила информация. Вдясно → Diagram → right-click → Add IP → Search → Concat → double click. Това е блок, който свързва магистрали с различна разредност. В конкретния случай сигналът за прекъсване е 1 бит, а магистралата на контролера за прекъсвания на ARM Cortex A9 е 16-битова. Добавя се още един такъв блок за MicroBlaze микропроцесора.

23. Щраква се два пъти върху Concat блоковете и в полето Number of Ports се въвежда числото 1.

24. Щраква се два пъти върху блока “ZYNQ7 Processing System” → в “Page navigator” се отива на раздел “Interrupts” → слага се отметка на “Fabric interrupts” → в подраздела “PL-PS Interrupt Ports” се слага отметка на “IRQ_F2P” → OK.

25. Нека блокът xlconcat_0 отговаря за прекъсванията на Zynq блока, а xlconcat_1 отговаря за прекъсванията на MicroBlaze. Натиска се и се задържа ляв бутон върху dout[0:0] порта на xlconcat_0 блока и се свързва с IRQ_F2P[0:0] на Zynq, след това се пуска левия бутон. Аналогично се свързва In0[0:0] порта на xlconcat_0 с Interrupt_1 сигнала на mailbox_0.

26. За да може MicroBlaze да работи с прекъсвания, трябва първо да му се добави контролер на прекъсванията. Вдясно → Diagram → right-click → Add IP → Search → AXI Interrupt Controller → double click. Натиска се и се задържа ляв бутон върху INTERRUPT порта на microblaze_0 и се свързва с interrupt на AXI Interrupt Controller, след това се пуска левия бутон. Аналогично се свързват intr[0:0] на axi_intc_0 с dout[0:0] на xlconcat_1. Също In0[0:0] на xlconcat_1 се свързва с Interrupt_0 на mailbox_0.

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

28. 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.

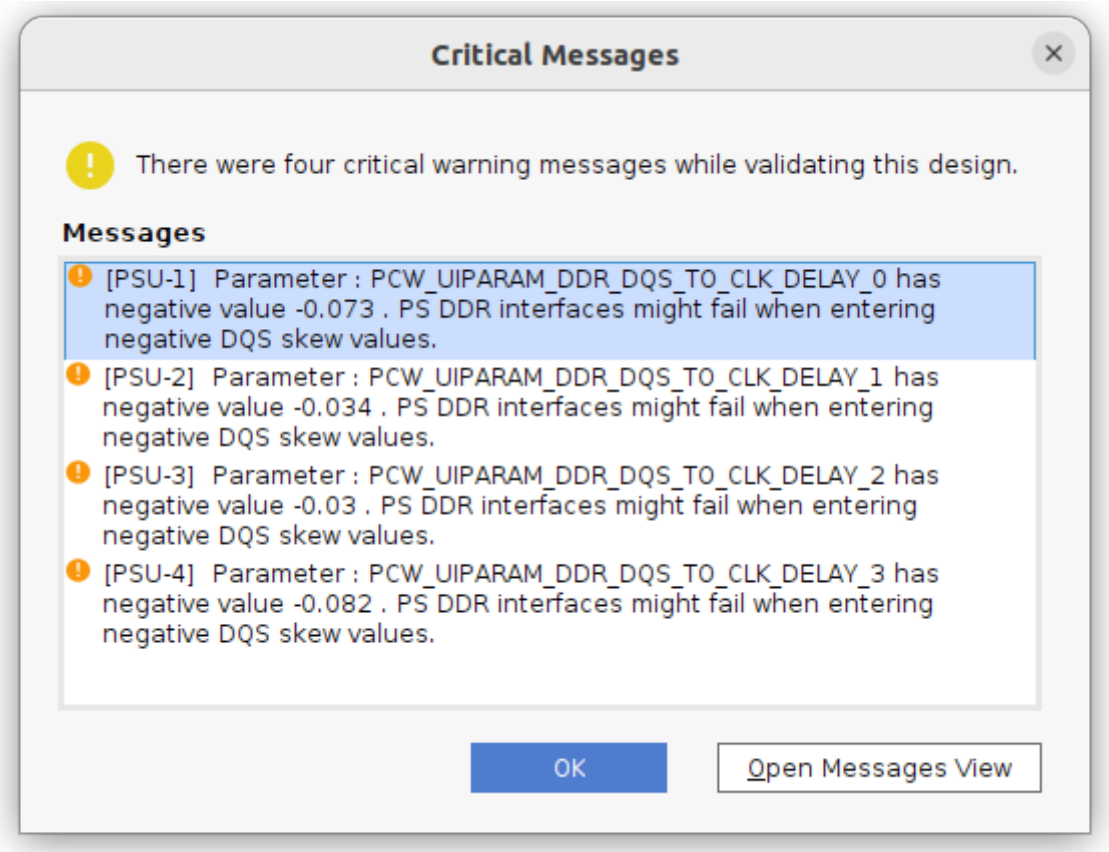
29. В същия прозорец → “Page navigator” → MIO Configuration → маха се отметката на I/O Peripherals → ENET0, USB0 и SD0. Натиска се OK.

30. 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.

31. Подрежда се блоковата схема с бутон Regenerate Layout.

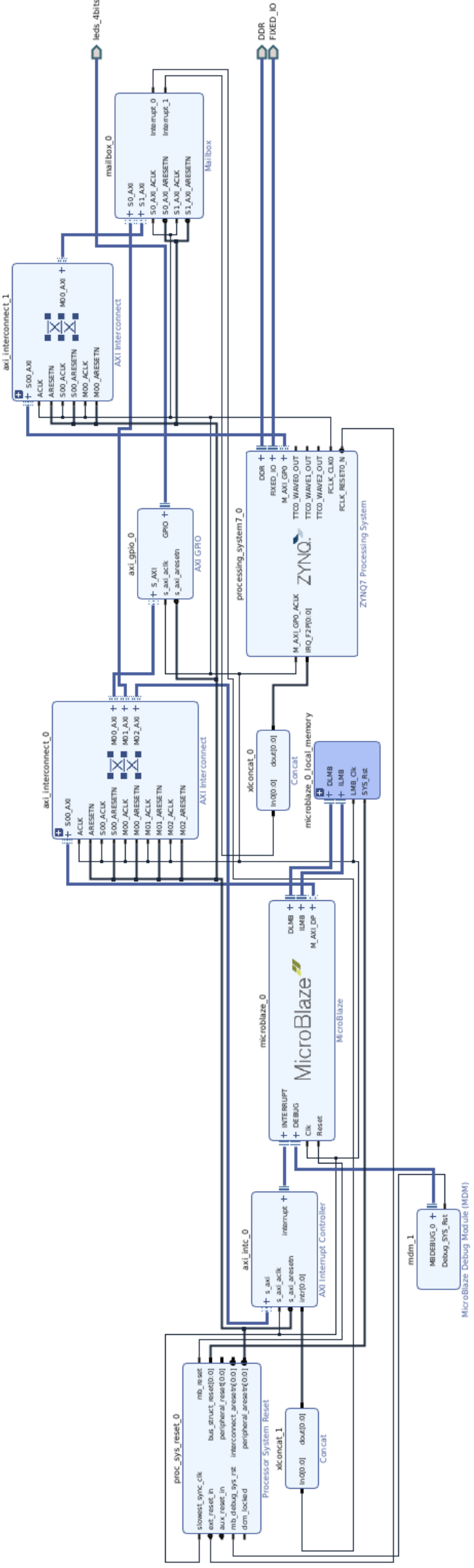
32. Вдясно→Diagram→лента с бутони → Validate Design (F6) → "Validation successful. There are no errors or critical warnings in this design." → OK

В някои версии на Vivado е възможно да се появят предупредителни съобщения, като тези показани по-долу, но те могат да се игнорират в конкретния дизайн.



33. 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 е показана на следващата страница, както и в директорията на упражнението с име на файл 06_design_1.pdf.


leds_4bits
DDR
FIXED_IO
mailbox_0
Mailbox
axi_interconnect_1
AXI Interconnect
axi_gpio_0
AXI GPIO
processing_system7_0
ZYNQ7 Processing System
axi_interconnect_0
AXI Interconnect
xlconcat_0
Concat
microblaze_0_local_memory
microblaze_0
MicroBlaze
axi_intc_0
AXI Interrupt Controller
mdm_1
MicroBlaze Debug Module (MDM)
proc_sys_reset_0
Processor System Reset
xlconcat_1
Concat

34. 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.

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

==========

==========

36. Tools → Launch Vitis IDE

37. 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.

38. File → New → Platform project → Platform project name: 12_mailbox_mb_pla → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 12_mailbox, създаден от Vivado → design_1_wrapper.xsa → Open → Поле Software specification → Operating system: standalone (това означава bare-metal firmware) и Processor: microblaze_0 → Finish.

39. Вляво → Project explorer → избира се 12_mailbox_mb_pla → right-click → Build Project.

40. File → New → Application project → Next → "Select a platform from repository" → Избира се 12_mailbox_mb_pla → Next → Application project name: 12_mailbox_mb_app → Next → Next → “Empty application (C)” → Finish.

41. Вляво в таб Explorer → отваря се 12_mailbox_mb_app_system / 12_mailbox_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.

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

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

int main(void){
      XGpio output;
      XMbox_Config *mailbox_config;
      XMbox mailbox_0;
      u32 bytes_receved = 0;
      char receive_buffer[32];

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

    mailbox_config = XMbox_LookupConfig(XPAR_MBOX_0_DEVICE_ID);
    XMbox_CfgInitialize(&mailbox_0, mailbox_config, mailbox_config-
>BaseAddress);


      while(1){
            XMbox_Read(&mailbox_0, (u32 *)receive_buffer, 4, &bytes_receved);
            if(bytes_receved > 0){
                  if(!strcmp(receive_buffer, "ON  ")){ //multiple of 4 bytes
                        XGpio_DiscreteWrite(&output, 1, 0x01);
                  }

                  if(!strcmp(receive_buffer, "OFF ")){ //multiple of 4 bytes
                        XGpio_DiscreteWrite(&output, 1, 0x00);
                  }
            }
      }

      return 0;
}
```

43. Вляво → Project explorer → избира се 12_mailbox_mb_app → right-click → Build Project.

44. Вляво → Project explorer → избира се 12_mailbox_mb_app_system → right-click → Build Project.

45. File → New → Platform project → Platform project name: 12_mailbox_cortex_pla → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 12_mailbox, създаден от Vivado → design_1_wrapper.xsa → Open → Поле Software specification → Operating system: standalone и Processor: ps7_cortexa9_0 → Finish.

46. Вляво → Project explorer → избира се 12_mailbox_cortex_pla → right-click → Build Project.

47. File → New → Application project → Next → "Select a platform from

repository" → Избира се 12_mailbox_cortex_pla → Next → Application project name: 12_mailbox_cortex_app → Next → Next → “Hello World” → Finish.

48. Щраква се двукратно с ляв бутон върху директорията src в проекта 12_mailbox_cortex_app_system/ 12_mailbox_cortex_app → src → helloworld.c

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

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

int main(void){
    u32 bytes_sent = 0;
    char *on_message = "ON  "; //multiple of 4 bytes
    char *off_message = "OFF "; //multiple of 4 bytes
    XMbox_Config *mailbox_config;
    XMbox mailbox_0;

    init_platform();

    mailbox_config = XMbox_LookupConfig(XPAR_MBOX_0_DEVICE_ID);
    XMbox_CfgInitialize(&mailbox_0, mailbox_config, mailbox_config-
>BaseAddress);



    while(1){
      print("Sending message to MicroBlaze: led ON\n\r");
      XMbox_Write(&mailbox_0, (u32 *)on_message, strlen(on_message),
&bytes_sent);
      usleep(1000000);
      print("Sending message to MicroBlaze: led OFF\n\r");
      XMbox_Write(&mailbox_0, (u32 *)off_message, strlen(off_message),
&bytes_sent);
      usleep(1000000);
    }

    cleanup_platform();

    return 0;
}
```

50. Вляво, Project explorer → избира се 12_mailbox_cortex_app → right-click → Build project.

51. Вляво, Project explorer → избира се 12_mailbox_cortex_app_system → right-click → Build project.

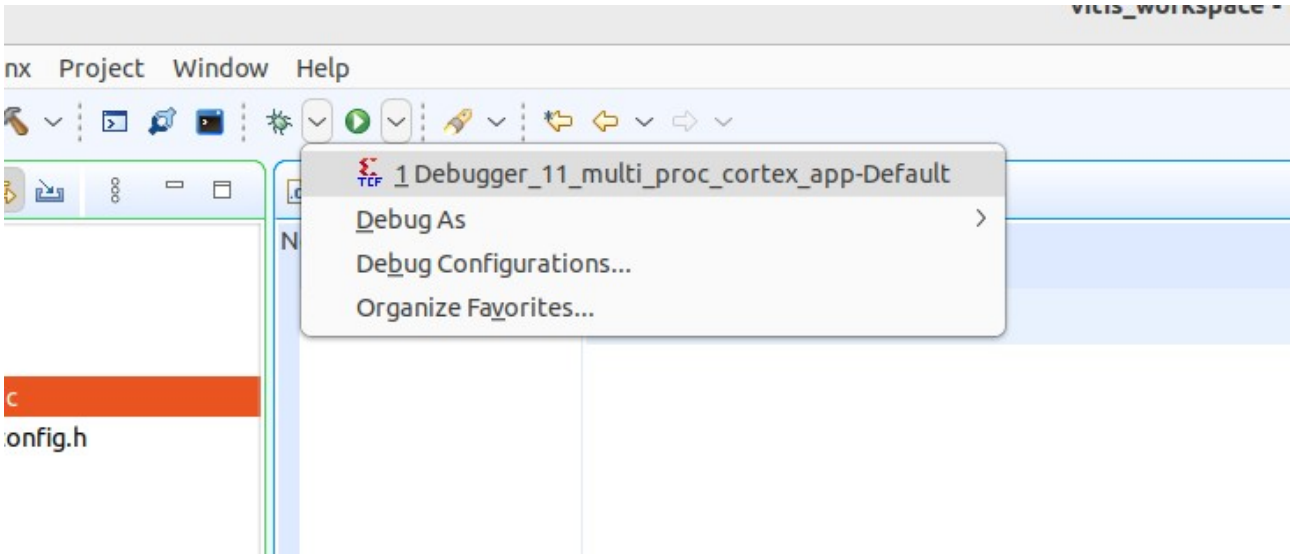
52. В основния прозорец на Vitis до бутонът Debug има стрелка надолу →

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. Затова → 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/12_mailbox_mb_app.elf
ps7_cortexa9_0 Debug/12_mailbox_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.

53. 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.

54. 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
```

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

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

```
Sending message to MicroBlaze: led OFF
Sending message to MicroBlaze: led ON
Sending message to MicroBlaze: led OFF
Sending message to MicroBlaze: led ON
Sending message to MicroBlaze: led OFF
Sending message to MicroBlaze: led ON
```

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

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

59. Напишете програма за MicroBlaze, която използва прекъсвания от mailbox_0 [3]. За улеснение, по-долу е даден кодът, който е свързан с инициализацията на контролерът на прекъсванията за MicroBlaze[4], също и функцията за изчистване на флаговете от прекъсванията. Функциите, свързани с mailbox_0 могат да бъдат намерени в:

12_mailbox_mb_pla/microblaze_0/standalone_domain/bsp/microblaze_0/libsrc/mbox_v4_5/src/xmbox.c

като повече внимание трябва да се обърне на функцията Xmbox_SetReceiveThreshold( ), която има своя особеност в начинът ѝ на работа.

```
	XIntc_Initialize(&intc_0, XPAR_INTC_0_DEVICE_ID);
	XIntc_SelfTest(&intc_0);
	XIntc_Connect(&intc_0, XPAR_AXI_INTC_0_MAILBOX_0_INTERRUPT_0_INTR,
(XInterruptHandler)mailbox_interrupt_handler, &mailbox_0);
	XIntc_Start(&intc_0, XIN_REAL_MODE);
	XIntc_Enable(&intc_0, XPAR_AXI_INTC_0_MAILBOX_0_INTERRUPT_0_INTR);

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

	microblaze_register_handler(mailbox_interrupt_handler, NULL);
	microblaze_enable_interrupts();

	//Clear the AXI interrupt controller's pending flag
	XIntc_Acknowledge(&intc_0, XPAR_AXI_INTC_0_MAILBOX_0_INTERRUPT_0_INTR);
```

* * *

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

[2] Adam Tylor, “MicroZed Chronicles: Inter Processor Communication (Part 2)”, online, https://www.hackster.io/news/microzed-chronicles-inter-processor-communication-part-2-e3f239921d79, 2023.

[3] https://github.com/Xilinx/embeddedsw/blob/master/XilinxProcessorIPLib/drivers/mbox/examples/xmbox_intr_example.c

[4] https://github.com/Xilinx/embeddedsw/blob/master/XilinxProcessorIPLib/drivers/intc/examples/xintc_example.c

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