# Проектиране на вградени автомобилни електронни системи
# Лабораторно упражнение №4

Работа с Xilinx Vivado и Vitis. Синтезиране на микропроцесорна система върху FPGA. Работа с таймерен модул и контролер на прекъсвания.

==========

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: 04_timer → 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

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

10. Вдясно → Diagram → right-click → Add IP → Search → AXI Timer → double click

11. Вдясно → Diagram → right-click → Add IP → Search → AXI Timer → double click, за да добавите втори таймерен модул.

12. Има два таймера, които ще генерират прекъсвания. Как ще се свържат към

един вход за прекъсване на микропроцесора? Отговор: чрез блок, който обединява сигналите [1], [2].

13. Вдясно → Diagram → right-click → Add IP → Search → concat→ double click
Ако трябват повече от 2 входа, трябва да се щракне два пъти върху concat блока и в полето “Number of ports” да се попълни с числото, което е необходимо.

14. 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 [15:0]” → OK

15. Свързват се “interrupt” изходите на таймерите с двата входа на Concat модула (In0[0:0] и In1[0:0]).
*Натиска се ляв бутон върху извода и се задържа, докато се тегли връзката. Когато се свърже с другия пин, трябва да се пусне левия бутон.*

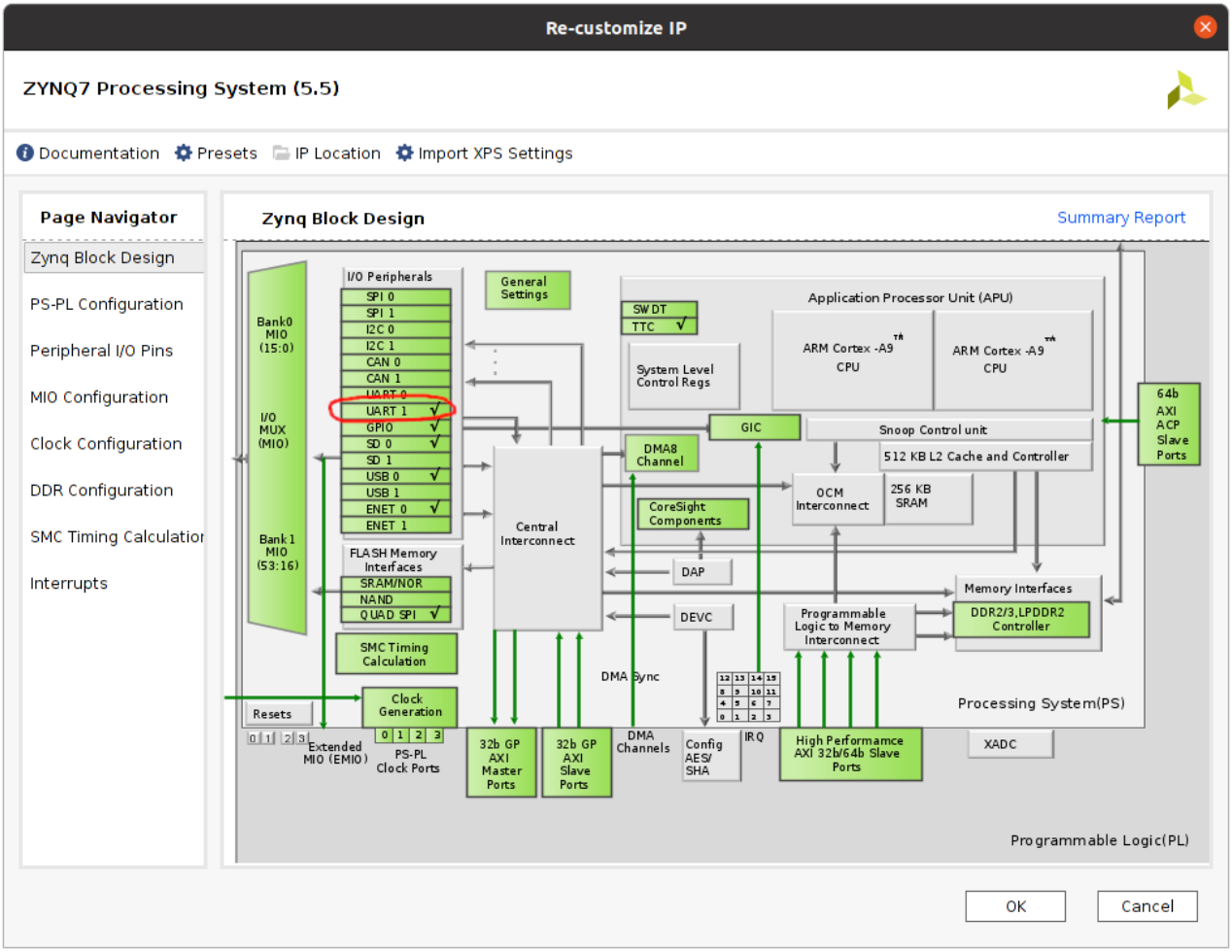
16. Свързва се изхода dout[1:0] на “concat” блока с входа IRQ_F2P[0:0] на “ZYNQ7 Processing System” блока.

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

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

19. Подрежда се блоковата схема с бутон Regenerate Layout.

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

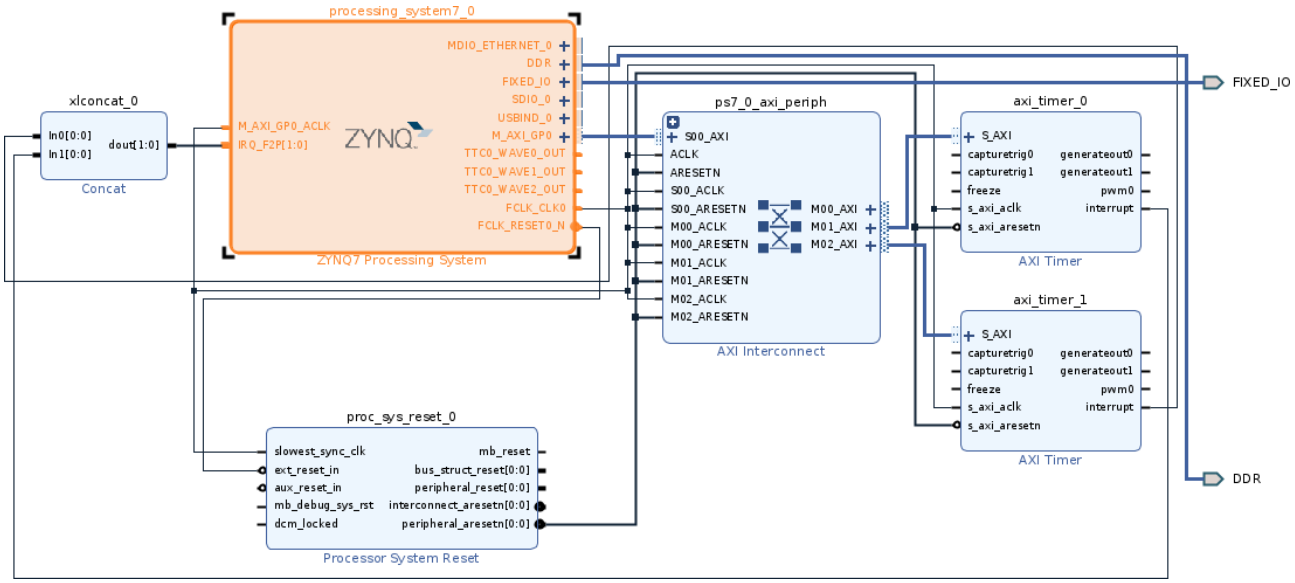



Проверява се дали сигналите на UART блока са свързани към правилните изводи. Това са MIO48 и MIO49, които са свързани към виртуалния сериен порт на дебъгера и не се нуждаят от USB-към-UART конвертор. Сигналите може да се видят от таб MIO Configuration → I/O Peripherals → UART1 (или UART0) → MIO48 ↔ tx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

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

22. 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

23. 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.

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

==========

25. 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.

26. 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.

27. File → New → Platform project → Platform project name: 04_timer → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 04_timer, създаден от Vivado → design_1_wrapper.xsa → Open → Finish

28. Вляво → Project explorer → избира се 04_timer → right-click → Build Project

29. File → New → Application project → Next → "Select a platform from repository" → Избира се 04_timer → Next → Application project name: 04_timer_app → Next → Next → “Hello World” → Finish

30. Щраква се двукратно с ляв бутон върху директорията src в проекта 04_timer_app_system/04_timer_app → src → helloworld.c

31. В текстовия редактор на Vitis се въвежда следната програма:

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

#include "xparameters.h"
#include "xtmrctr.h"
#include "xscugic.h"
#include "xil_exception.h"

XScuGic INTC0;
XTmrCtr TIM0;

void timer0_int_handler(void *data){
	print("Hello World\n\r");
	XTmrCtr_Reset(&TIM0, 0);
}

int main(){
	XScuGic_Config *intc_config;
	init_platform();

	intc_config = XScuGic_LookupConfig(XPAR_PS7_SCUGIC_0_DEVICE_ID);
	XScuGic_CfgInitialize(&INTC0, intc_config, intc_config->CpuBaseAddress);
	XScuGic_Connect(&INTC0, XPAR_FABRIC_AXI_TIMER_0_INTERRUPT_INTR,
			(Xil_ExceptionHandler) timer0_int_handler, &TIM0);
	XScuGic_Enable(&INTC0, XPAR_FABRIC_AXI_TIMER_0_INTERRUPT_INTR);
	Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_INT, (Xil_ExceptionHandler)
			XScuGic_InterruptHandler, &INTC0);
	Xil_ExceptionEnable();

	XTmrCtr_Initialize(&TIM0, XPAR_AXI_TIMER_0_DEVICE_ID);
	XTmrCtr_SetHandler(&TIM0, (XTmrCtr_Handler)timer0_int_handler, &TIM0);
	XTmrCtr_SetResetValue(&TIM0, 0, 100000000U);
	XTmrCtr_SetOptions(&TIM0, 0, XTC_INT_MODE_OPTION | XTC_AUTO_RELOAD_OPTION
			| XTC_DOWN_COUNT_OPTION);
	XTmrCtr_Start(&TIM0, 0);

    print("Starting timer with interrupts...\n\r");

    while(1){

    }

    cleanup_platform();

    return 0;
}
```

32. Вляво, Project explorer -> избира се 04_timer_app_system -> right-click -> Build project

33. Вляво, Project explorer -> избира се 04_timer_app_system -> right-click -> Debug as -> Launch Hardware

34. 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

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

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

```
Starting timer with interrupts...
[19:47:49:763] Hello World
[19:47:50:771] Hello World
[19:47:51:763] Hello World
```

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

38. Напишете програма, която пуска и втория таймер, работещ с два пъти по-кратък период.

39. В прекъсванията не трябва да се изпълнява много код, а printf е сложна функция. Прехвърлете принтирането в main( ), нека в хендлера само да се известява за препълването на таймера.

* * *

[1] Abhin Ayp, "How to handle more that 16 interrupts using the AXI Interrupt Controller", online, 2021.
https://support.xilinx.com/s/article/1165154?language=en_US

[2] https://support.xilinx.com/s/article/58942?language=en_US

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