Проектиране на вградени автомобилни електронни системи
Лабораторно упражнение №6

Работа с Xilinx Vivado и Vitis. Синтезиране на микропроцесорна система върху FPGA. Работа със SPI 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. 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: 06_spi → 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. Щраква се два пъти върху блока “ZYNQ7 Processing System” → в “Page navigator” → MIO Configuration → слага се отметка на I/O Peripherals → UART1 и се проверяват връзките MIO48 ↔ tx, MIO49 ↔ rx

11. В същия прозорец → “Page navigator” → MIO Configuration → маха се отметката на I/O Peripherals → ENET0 и USB0.

12. В същия прозорец → “Page navigator” → MIO Configuration → слага се

отметка на Memory Interfaces → Quad SPI Flash → Single SS 4-bit IO → на падащото меню Data mode се избира x1 → проверяват връзките:
MIO1 ↔ qspi0_ss_b
MIO2 ↔ qspi0_io[0]
MIO3 ↔ qspi0_io[1]
MIO5 ↔ qspi0_io[3]/HOLD_B
MIO6 ↔ qspi0_sclk

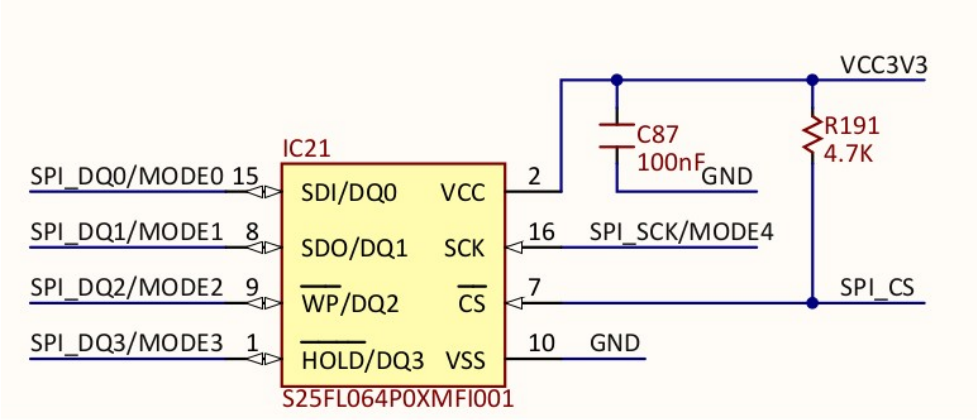
13. Вдясно →Diagram → зелена лента → Designer Assitance available → Run Block Automation → Слага се отметка на "All Automation".

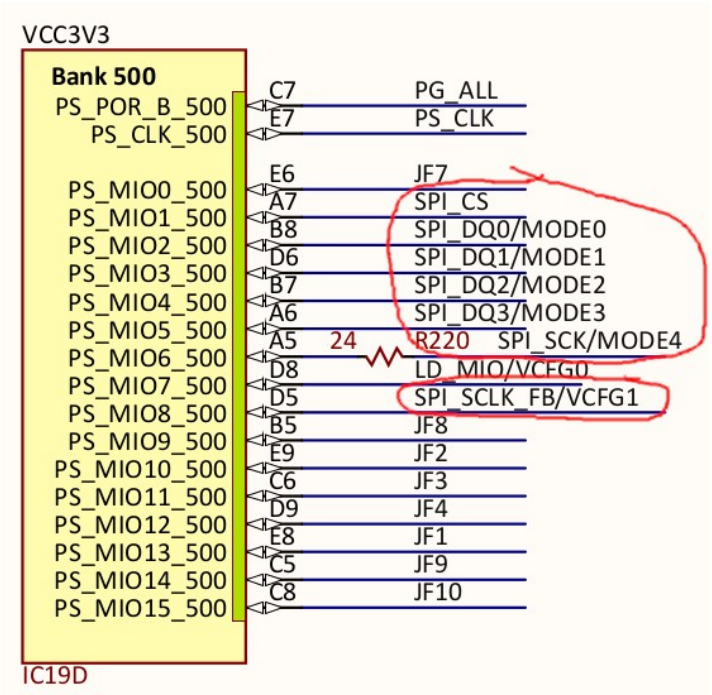
14. Вдясно →Diagram → зелена лента → Designer Assitance available -> Run Connection Automation → Слага се отметка на "All Automation". Натиска се OK.

15. Подрежда се блоковата схема с бутон Regenerate Layout.

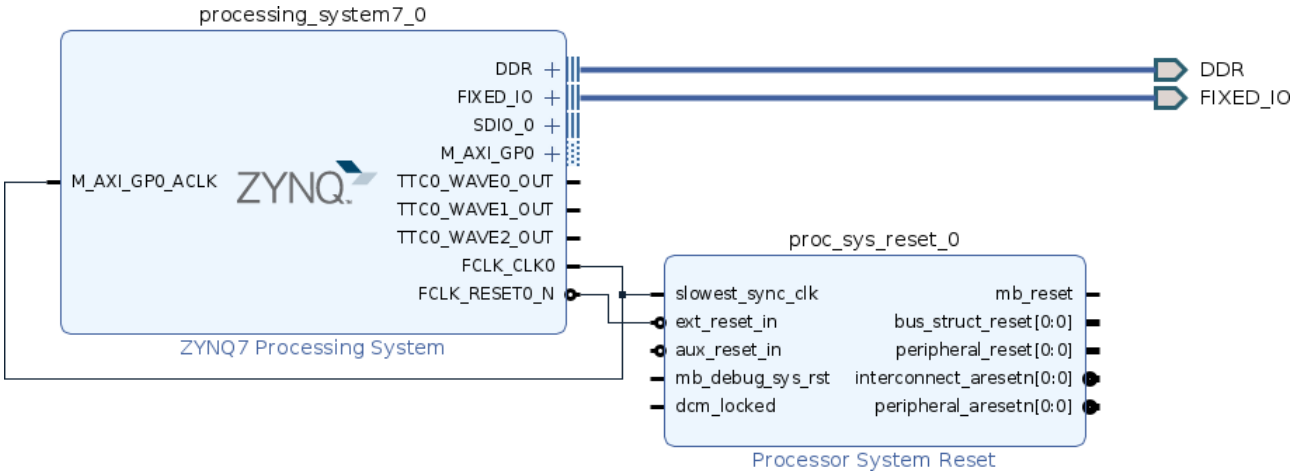
16. Вдясно→Diagram→лента с бутони → Validate Design (F6) → "Validation successful. There are no errors or critical warnings in this design." → OK

17. 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:



18. 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.

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

==========

==========

20. Tools → Launch Vitis IDE

21. 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.

22. File → New → Platform project → Platform project name: 06_spi → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 06_spi, създаден от Vivado → design_1_wrapper.xsa → Open → Finish

23. Вляво→Project explorer → избира се 06_spi → right-click → Build Project.

24. File → New → Application project → Next → "Select a platform from repository" → Избира се 06_spi → Next → Application project name: 06_spi_app

→ Next → Next → “Hello World” → Finish.

25. Щраква се двукратно с ляв бутон върху директорията src в проекта 06_spi_app_system/06_spi_app → src → helloworld.c

26. В текстовия редактор на Vitis се въвежда следната програма [1], [2]:

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

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

XScuGic INTC0;
XQspiPs QSPI0;

volatile u8 transmitting_data;

void qspi_status_handler(void *CallBackRef, u32 StatusEvent, unsigned int
ByteCount){
      transmitting_data = 0;
}

int main(){
      u8 cmd_buffer[16];
      u8 read_buffer[256];
      XScuGic_Config *intc_config;
      XQspiPs_Config *qspi_config;

    init_platform();

    print("Starting the SPI Flash example ...\n\r");

    intc_config = XScuGic_LookupConfig(XPAR_PS7_SCUGIC_0_DEVICE_ID);
    XScuGic_CfgInitialize(&INTC0, intc_config, intc_config->CpuBaseAddress);
    XScuGic_Connect(&INTC0, XPAR_XQSPIPS_0_INTR, (Xil_ExceptionHandler)
XQspiPs_InterruptHandler, &QSPI0);
    XScuGic_Enable(&INTC0, XPAR_XQSPIPS_0_INTR);
    Xil_ExceptionInit();
    Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_INT, (Xil_ExceptionHandler)
XScuGic_InterruptHandler, &INTC0);
    Xil_ExceptionEnable();

    qspi_config = XQspiPs_LookupConfig(XPAR_XQSPIPS_0_DEVICE_ID);
    XQspiPs_CfgInitialize(&QSPI0, qspi_config, qspi_config->BaseAddress);
    XQspiPs_SetStatusHandler(&QSPI0, &QSPI0, (XQspiPs_StatusHandler)
qspi_status_handler);
    XQspiPs_SetOptions(&QSPI0, XQSPIPS_FORCE_SSELECT_OPTION |
XQSPIPS_MANUAL_START_OPTION | XQSPIPS_HOLD_B_DRIVE_OPTION);
    XQspiPs_SetClkPrescaler(&QSPI0, XQSPIPS_CLK_PRESCALE_8);
    XQspiPs_SetSlaveSelect(&QSPI0);

    memset(read_buffer, 0x00, 256);
```

```
    cmd_buffer[0] = 0x90;//Read ID command
    cmd_buffer[1] = 0x00; //Address byte 0
    cmd_buffer[2] = 0x00; //Address byte 1
    cmd_buffer[3] = 0x00; //Address byte 2
    cmd_buffer[4] = 0x00; //Dummy byte
    cmd_buffer[5] = 0x00; //Dummy byte
    cmd_buffer[6] = 0x00; //Dummy byte

    transmitting_data = 1;

    XQspiPs_Transfer(&QSPI0, cmd_buffer, read_buffer, 7);

    while (transmitting_data){ }

    xil_printf("Flash ID = 0x%02X 0x%02X\n\r", read_buffer[5], read_buffer[6]);

    while(1){ }

    cleanup_platform();

    return 0;
}
```

27. Вляво, Project explorer -> избира се 06_spi_app_system -> right-click -> Build project

28. Вляво, Project explorer -> избира се 06_spi_app_system -> right-click -> Debug as -> Launch Hardware

29. 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
```

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

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

```
Starting the SPI Flash example ...
[22:43:31:646] Flash ID = 0x17 0x01
```

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

33. Напишете програма, която прочита JEDEC уникалния номер, 16-битов Device ID и 66-байтов CFI блок с командата RDID.

34. Напишете програма, която прочита 128 байта от адрес 0x00.0000 и ги принтира в следния формат:

```
Starting the SPI Flash example ...
[23:17:21:836] Data block:
[23:17:21:836]
[23:17:21:836] 0xFE 0xFF 0xFF 0xEA 0xFE 0xFF 0xFF 0xEA 0xFE 0xFF 0xFF 0xEA 0xFE 0xFF 0xFF 0xEA
[23:17:21:852] 0xFE 0xFF 0xFF 0xEA 0xFE 0xFF 0xFF 0xEA 0xFE 0xFF 0xFF 0xEA 0xFE 0xFF 0xFF 0xEA
[23:17:21:852] 0x66 0x55 0x99 0xAA 0x58 0x4E 0x4C 0x58 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x01
[23:17:21:868] 0xC0 0x0A 0x00 0x00 0x0C 0x40 0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
[23:17:21:868] 0x0C 0x40 0x01 0x00 0x01 0x00 0x00 0x00 0x68 0xD1 0x16 0xFC 0x00 0x00 0x00 0x00
[23:17:21:868] 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
[23:17:21:884] 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
[23:17:21:884] 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
                              *                    *                    *
```

[1] https://xilinx.github.io/embeddedsw.github.io/qspips/doc/html/api/globals_func.html
[2] https://github.com/Xilinx/embeddedsw/blob/master/XilinxProcessorIPLib/drivers/qspips/examples/xqspips_flash_intr_example.c

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