



## Проектиране на вградени автомобилни електронни системи

### Лабораторно упражнение №7

Работа с Xilinx Vivado и Vitis. Синтезиране на микропроцесорна система върху FPGA. Работа със SD карти.

=====

1. Превключете джъмпера вдясно на платката на позиция JTAG. Свържете  $\mu$ USB 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: 07\_sd\_card → 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

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

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

12. Щраква се два пъти върху блока "ZYNQ7 Processing System" → в "Page navigator" → MIO Configuration → маха се отметката на I/O Peripherals → ENET0 и USB0.

13. В същия прозорец → "Page navigator" → MIO Configuration → слага се

отметка на I/O Peripherals → UART1 и се проверяват връзките MIO48 ↔ tx, MIO49 ↔ rx

14. В същия прозорец → “Page navigator” → MIO Configuration → слага се отметка на I/O Peripherals → SD0 и се проверяват връзките:

MIO40 ↔ clk

MIO41 ↔ cmd

MIO42 ↔ data[0]

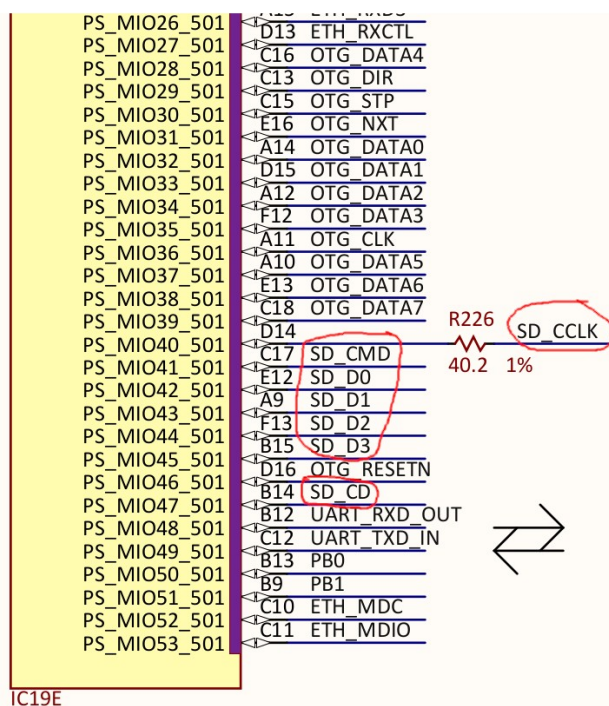
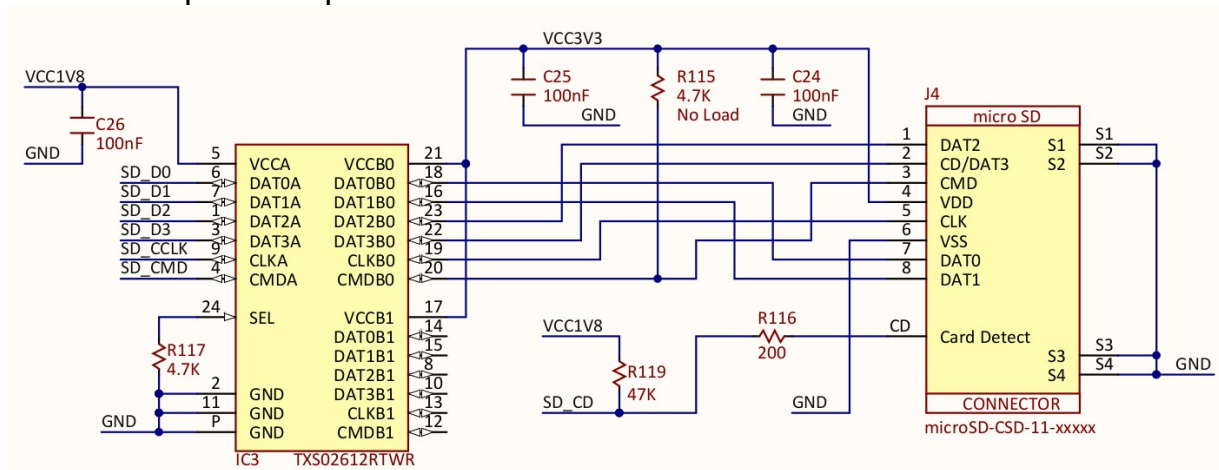
MIO43 ↔ data[1]

MIO44 ↔ data[2]

MIO45 ↔ data[3]

MIO47 ↔ cd (Card Detect, CD, с който се детектира слагането и изваждането на SD картата)

Забележете IC3 – TXS02612RT, който е разширител на порт, но в случая се ползва като транслатор на нива 1V8 ↔ 3V3.

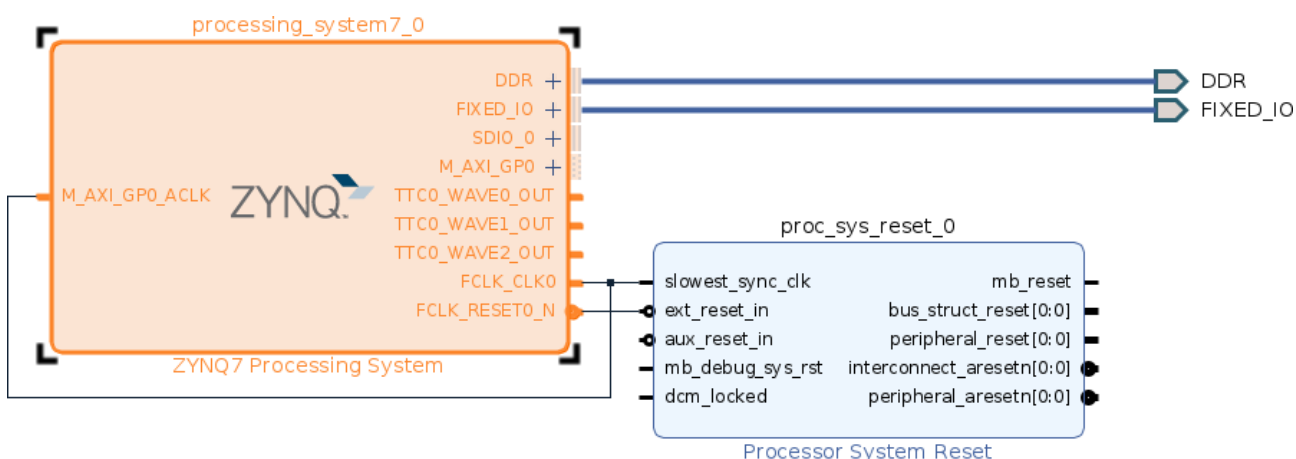


15. Подрежда се блоковата схема с бутон Regenerate Layout.

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 наблюдавате 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: 07\_sd\_card → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 07\_sd\_card, създаден от Vivado → design\_1\_wrapper.xsa → Open → Finish

23. В xsdps\_card.c се намират low-level API функции, които се използват от файловата система xilffs\_v4\_4 (която е създадена като статична библиотека). НО тази библиотека не е добавена по подразбиране към проекта. Затова: вляво на средата до таб Explorer има друга таб, Assistant → избира се проекта 07\_sd\_card [Platform] → десен бутон → Open Platform Editor → избира се “standalone on ps7\_cortexa9\_0 → Board Support Package → централно ще се появи таб с бутон “Modify BSP Settings” → натиска се този бутон → слага се отметка на Supported libraries / xilffs → OK.

24. Вляво → Project explorer → избира се 07\_sd\_card (Out-of-date) → right-click → Build Project.

25. File → New → Application project → Next → "Select a platform from repository" → Избира се 07\_sd\_card → Next → Application project name: 07\_sd\_card\_app → Next → Next → “Hello World” → Finish.

26. Щраква се двукратно с ляв бутон върху директорията src в проекта 07\_sd\_app\_system/07\_sd\_app → src → helloworld.c

27. В текстовия редактор на Vitis се въвежда следната програма [1]:

*Уверете се, че SD картата е форматирана с FAT32 файлова система и че поддържа висока скорост на обмен (125 MHz, CLASS10 UHS-I, GOODRAM 16 GB или 32 GB).*

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

#include "ff.h"

int main(){
    FIL fil;
    FATFS fatfs;
    FRESULT res;
    TCHAR *root_path = "0:/";
    char *file_contents = "Hello, world!";
    UINT bytes_written;
    //BYTE page[FF_MAX_SS];
```

```

init_platform();

printf("Starting SD card example ...\n\r");

res = f_mount(&fatfs, root_path, 0);
xil_printf("f_mount: %d\n", res);

//f_mkfs(root_path, FM_FAT32, 0, page, sizeof(page)); //Format sd card with
FAT32

res = f_open(&fil, "test.txt", FA_CREATE_ALWAYS | FA_WRITE);
xil_printf("f_open: %d\n", res);
res = f_write(&fil, (const void*)file_contents, 13, &bytes_written);
xil_printf("f_write: %d\n", res);
res = f_close(&fil);
xil_printf("f_close: %d\n", res);
res = f_mount(NULL, root_path, 0); // = unmount
xil_printf("f_unmount: %d\n", res);

while(1){ }

cleanup_platform();

return 0;
}

```

28. Вляво, Project explorer -> избира се 07\_sd\_card\_app\_system -> right-click -> Build project.

29. Вляво, Project explorer -> избира се 07\_sd\_card\_app\_system -> right-click -> Debug as -> Launch Hardware.

30. 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

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

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

Starting SD card example ...<sup>C<sub>R</sub></sup> <sup>L<sub>F</sub></sup>

[19:21:48:580] f\_mount: 0<sup>L<sub>F</sub></sup>

[19:21:52:996] f\_open: 0<sup>L<sub>F</sub></sup>

[19:21:52:996] f\_write: 0<sup>L<sub>F</sub></sup>

[19:21:52:996] f\_close: 0<sup>L<sub>F</sub></sup>

[19:21:52:996] f\_unmount: 0<sup>L<sub>F</sub></sup>

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

34. Проверете с четец на карти дали в SD картата е записан файл test.txt със съдържание “Hello, World!”. Поискайте четеца от ръководителя на упражнението.

35. За да не смущавате ръководителя на упражнението повече, напишете програма, която прочита току-що записания файл и принтира съдържанието му в Cutesom терминала [2]. Използвайте xil\_printf с форматни спецификатори.

\*

\*

\*

[1] [https://github.com/Xilinx/embeddedsw/blob/master/lib/sw\\_services/xilffs/examples/xilffs\\_polled\\_example.c](https://github.com/Xilinx/embeddedsw/blob/master/lib/sw_services/xilffs/examples/xilffs_polled_example.c)

[2] [http://elm-chan.org/fsw/ff/00index\\_e.html](http://elm-chan.org/fsw/ff/00index_e.html)

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