Проектиране на вградени автомобилни електронни системи
Лабораторно упражнение №10

Работа с Xilinx Vivado и Vitis. Синтезиране на микропроцесорна система върху FPGA. Работа с персонализиран IP 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. Стартирайте терминал с CTRL + ALT + T и изпълнете командите:

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

3. Create Project → Next → Project name: 10_custom_ip → 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. Tools → Settings → Project Settings → General → Target language → сменя се на VHDL → OK.

5. Tools → Create and Package New IP → Next → Create a new AXI4 peripheral → Next → Name: my_multiplier →
Name: S00_AXI
Interface Type: Lite
Interface Mode: Slave
Data Width (Bits): 32
Memory Size (Bytes): 64
Number of Registers: 4

→ Next → Edit IP → Finish

6. Ще се използва примерен IP модул на умножител от сайта FPGAdeveloper [1], [2].

7. Вляво → Add sources → Add or create design sources → Next → Add files → укажете пътя до 00_multiplier.vhd (има го в директорията на настоящото упражнение)→ слага се отметка на “Copy sources into IP directory” и се маха отметката “Scan and add RTL include files into project” → Finish.

8. В Sources → Design Sources → щраква се двукратно върху първото синьо кръгче my_multiplier_v1_0 (arch_imp) (my_multiplier_v1_0.vhd) (1), за да се отвори второ синьо подкръгче “my_multiplier_v1_0_S00_AXI_inst : my_multiplier_v1_0_S00_AXI (my_multiplier_v1_0_S00_AXI.vhd)”, което също трябва да се натисне двукратно, за да се отвори редактор за VHDL.

9. Преди ключовата дума begin се добавя:

```
signal multiplier_out : std_logic_vector(31 downto 0);

component multiplier
port (
  clk: in std_logic;
  a: in std_logic_VECTOR(15 downto 0);
  b: in std_logic_VECTOR(15 downto 0);
  p: out std_logic_VECTOR(31 downto 0));
end component;
```

10. Потърсете редът, на който пише “-- Add user logic here” и добавете кода:

```
  multiplier_0 : multiplier
  port map (
  clk => S_AXI_ACLK,
  a => slv_reg0(31 downto 16),
  b => slv_reg0(15 downto 0),
  p => multiplier_out);
```

11. Потърсете редът, на който пише “reg_data_out <= slv_reg1;” и го заместете с:

reg_data_out <= multiplier_out;

12. Малко по-нагоре от този ред, намерете “slv_reg1” и го заместете с “multiplier_out” → Save.

13. До редактора на VHDL има таб Package IP – my_multiplier → File Groups → натиска се “Merge changes from IP File Groups Wizard”. Сега “File Groups” трябва да има зелено тикче вляво на него.

14. Review and package IP → Re-package IP → “Finished packaging ‘my_multiplier’ v1.0 successfully … “ → Yes.

15. Вляво → Flow navigator → Create block design → OK.

16. Вдясно → Diagram → right-click → Add IP → Search → ZYNQ7 Processing System → double click.

17. 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.

18. Вдясно → Diagram → right-click → Add IP → Search → Processor System Reset → double click.

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

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

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

22. 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.

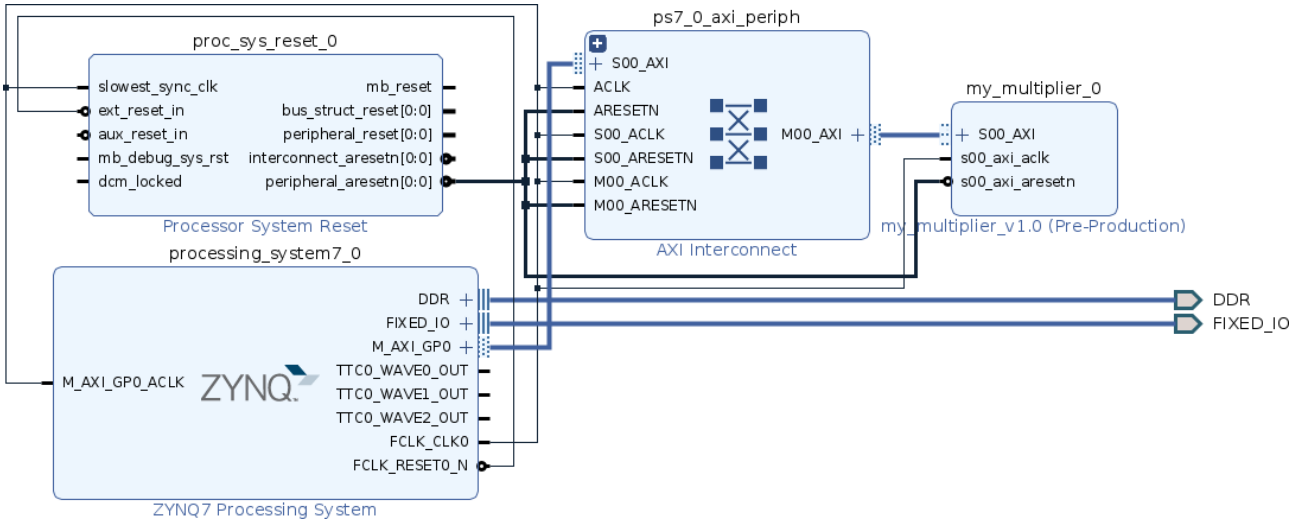
23. В същия прозорец → “Page navigator” → MIO Configuration → маха се отметката на I/O Peripherals → ENET0, USB0 и SD0.

24. Подрежда се блоковата схема с бутон Regenerate Layout.

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

26. 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:



27. 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.

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

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

29. Tools → Launch Vitis IDE

30. 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.

31. File → New → Platform project → Platform project name: 10_custom_ip → Next → таб "Create new platform from hardware" → Browse → избира се пътя до проекта 10_custom_ip, създаден от Vivado → design_1_wrapper.xsa → Open → Finish

32. Вляво → Project explorer → избира се 10_custom_ip → right-click → Build Project.

33. File → New → Application project → Next → "Select a platform from repository" → Избира се 10_custom_ip → Next → Application project name: 10_custom_ip_app → Next → Next → “Hello World” → Finish.

34. Щраква се двукратно с ляв бутон върху директорията src в проекта 10_custom_ip_app_system/10_custom_ip_app/src/helloworld.c

35. Щракнете двукратно върху:
10_custom_ip_app_system/10_custom_ip_app/10_custom_ip_app.prj
и в ново-отворилият се таб “Application Project Settings” натиснете “Hardware Specification: …”. Потърсете в картата на паметта вашия умножител my_multiplier. Трябва да изглежда по този начин:

	Base Address	High Address
my_multiplier	0x43c0 0000	0x43c0 ffff

Това са началният и крайният адрес на вашето IP. Регистър с офсет 0x00 ви е входния регистър, който приема две 16-битови числа, които ще бъдат умножавани, а регистър с офсет 0x04 е изходният регистър, съдържащ резултата.

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

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

#include "xparameters.h"

int main(){
    volatile u32 *my_mult_in = (volatile u32 *)0x43c00000;
    volatile u32 *my_mult_out = (volatile u32 *)0x43c00004;

    init_platform();

    print("Starting my_multiplier example ...\n\r");

    *my_mult_in = 0x00020003;

    xil_printf("Result: %d", *my_mult_out);

    while(1){ }

    cleanup_platform();

    return 0;
}
```

36. Вляво, Project explorer -> избира се 10_custom_ip_app_system -> right-click -> Build project.

37. Вляво, Project explorer -> избира се 10_custom_app_system -> right-click -> Debug as -> Launch Hardware.

38. 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
```

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

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

```
Starting my_multiplier example ...
[20:45:02:173] Result: 6
```

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

42. Напишете програма, която съпоставя структура към началния базов адрес и описва всички (всичките два) регистри на модула. Прочетете извадката от лекциите по “Микропроцесорна схемотехника” 06_struct_map.pdf как може да стане това. Заредете и тествайте кода.

* * *

[1] https://www.fpgadeveloper.com/2014/08/creating-a-custom-ip-block-in-vivado.html/
[2] https://github.com/fpgadeveloper/microzed-custom-ip/blob/master/Vivado/ip_repo/my_multiplier_1.0/src/multiplier.vhd

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