



Проектиране на вградени автомобилни електронни системи

Лабораторно упражнение №18

Работа с Xilinx Vivado и Vitis. Makefile и Menuconfig. Йерархични Makefile-ове. Документиране на сорс код с Doxygen.

=====

1. Превключете джъмпера вдясно на платката на позиция JTAG. Свържете μ USB кабел към PROG/UART USB куплунга. Включете платката от ключа ON/OFF. Включете USB-към-UART конвертор към сигнали JB1_P (T20), отговарящ на Uartlite_TxD, и JB1_N (U20), отговарящ на Uartlite_RxD. Модулът Uartlite ще бъде свързан към един MicroBlaze.

ЗАБЕЛЕЖКА: работната маса с платка Zybo Z7-10 трябва да използва куплунг JC и съответно сигналите Uartlite_TxD (V15), Uartlite_RxD (W15).

2. Използвайте предварително-синтезираната хардуерна платформа, дадена в директория 18_2 на настоящото лабораторно упражнение. Нейната блокова схема е показана на следващата страница, а картата на паметта – на последващата. Копирайте файловете в произволна работна директория, например /home/user/workspaces/xilinx_workspace, всички файлове от 18_2, които са:

- *bitstream.bit
- *cortex.elf
- *mb.elf
- *ps7_init.tcl
- *fsbl.elf

3. Стартирайте две копия на Cútecom (или gútkterm) и отворете портовете, отговарящи на printf канала на дебъгера и Стартирайте терминал с CTRL + ALT + T и изпълнете командите [1]:

```
cd /home/user/workspaces/xilinx_workspace
source /home/user/programs/Xilinx/Vitis/2022.2/settings64.sh
```

```
connect
targets
targets 4
fpga bitstream.bit
targets 2
source ps7_init.tcl
ps7_init
```

```
ps7_post_config
dow cortex.elf
con
targets 6
dow mb.elf
con
```

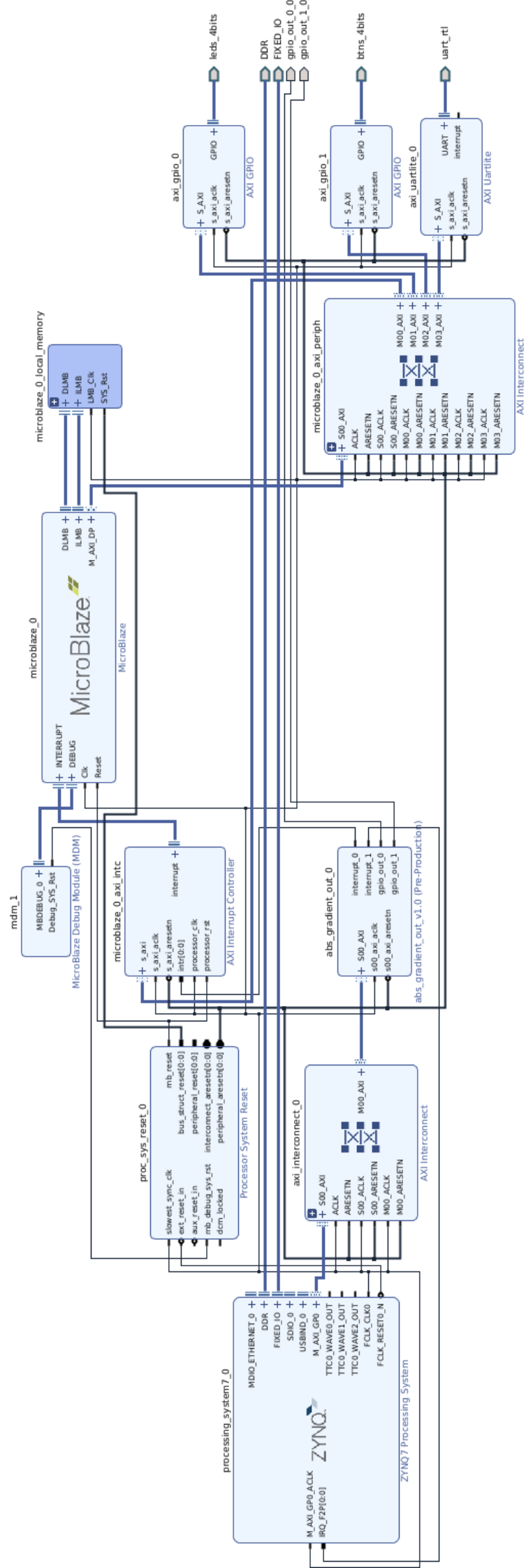
Ако всичко е минало успешно, светодиоди LD2 и LD3 трябва да мигат последователно един след друг, а светодиод LD1 веднъж в секунда (този диод показва постъпили прекъсвания в MicroBlaze от ARM Cortex A9). Отворете Cutecom (или gtkterm) на съответните виртуални COM портове и наблюдавайте съобщенията от двата микропроцесора.

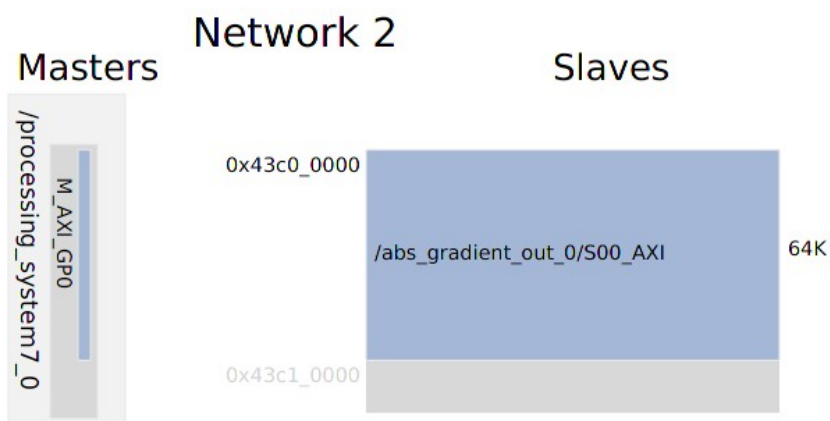
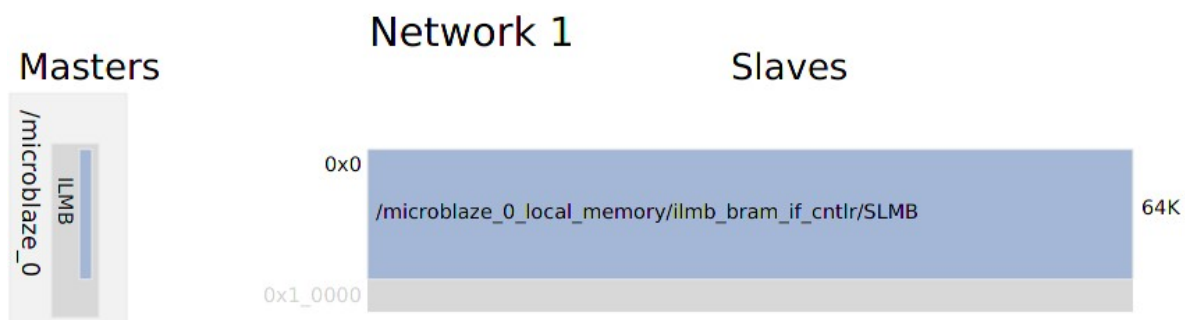
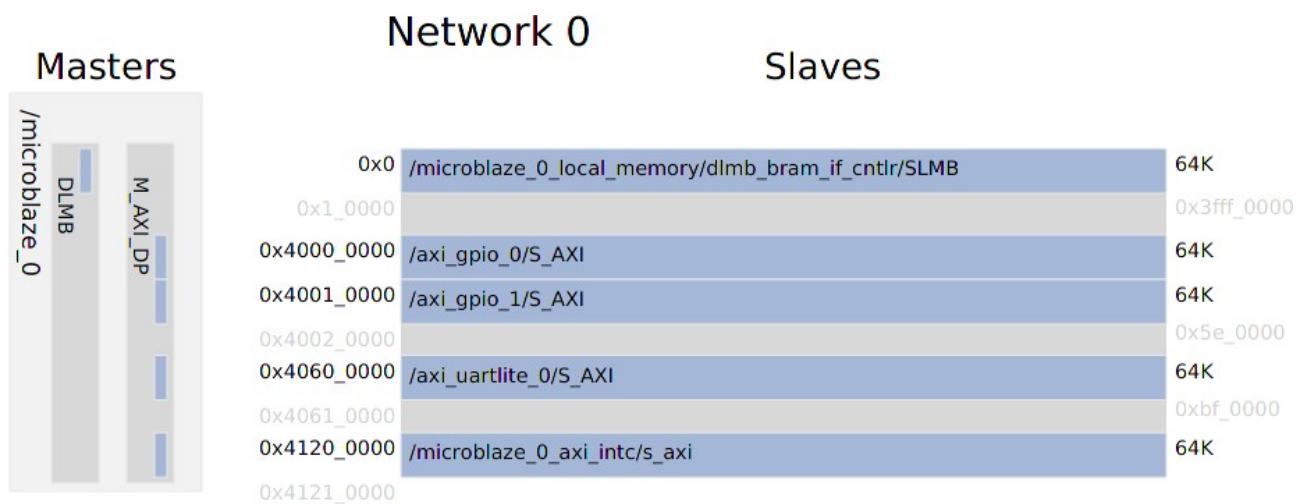
4. В миналата точка се разчита на скрипт (ps7_init.tcl), който чрез дебъгерни команди инициализира външната DDR памет, настройва PLL от модула за тактови сигнали, инициализира важни GPIO изводи, както и пуска таймер за SCU (Snoop Control Unit) модула (свързан с работата на кеша). На практика, обаче, в крайния вариант на системата тези операции ще се извършват от FSBL (First Stage Boot Loader) – програма за първо ниво на начално установяване (която всъщност се изпълнява втора след Boot ROM).

За да стартирате системата с FSBL, в терминала на XSCT напишете disconnect, изключете и пак включете платката. Въведете следните команди:

```
connect
targets
targets 4
fpga bitstream.bit
targets 2
dow fsbl.elf
con
dow cortex.elf
con
targets 6
dow mb.elf
con
```

ВНИМАНИЕ! Командите от предните две точки са валидни, ако в XSCT терминала излиза следния текст, след програмиране на bitstream.bit и въвеждане на команда targets:





xsct% targets

- 1 APU
- 2 ARM Cortex-A9 MPCore #0 (Running)
- 3 ARM Cortex-A9 MPCore #1 (Running)
- 4 xc7z010
- 5 MicroBlaze Debug Module at USER2
- 6* MicroBlaze #0 (Running)

Звезда до цифрата означава “процесор, който е избран в момента и може да му се прилагат дебъг команди”.

5. Затворете XSCT като в терминала напишете:

```
disconnect  
exit
```

Изключете и включете демо платката Zybo. След това напишете tcl скрипт, който да съдържа командите от миналата точка но със закъснение от 1 секунда след пускането на FSBL. Командата за закъснение в tcl има следния синтаксис:

```
after [num]
```

където [num] е число с размерност милисекунди, ms. Извикайте го по следния начин:

```
xsct <име-на-скриптов-файл>
```

6. Копирайте сорс файловете от директория 18_6 в работната ви директория. Компилирайте и линкнете тестовата програма за мигане на светодиода LD3 чрез Makefile. Нека обектовите и двоичните файлове да се преместят в отделна директория с име “debug”. За компилация и линкуване използвайте командата:

```
mb-gcc -Wl,-T -Wl,lscrip.ld -Wl,--no-relax -Wl,--gc-sections -I./include -L./ -lxil -  
mlittle-endian -mxl-barrel-shift -mxl-pattern-compare -mcpu=v11.0 -mno-xl-soft-mul  
main.c -o main.elf
```

Имплементирайте следните Makefile target-и:

```
make  
make prog  
make clean
```

които съответно правят следното:

- *създават debug директория и build-ват програмата;
- *програмират матрицата;
- *изтриват debug директорията.

7. Използвайки файловете от директория 18_7, разделете кода на четири части:

- *main.c - с основния алгоритъм;
- *led/led.c - с функция за инициализация и включване/изключване на светодиода;
- *uart/uart.c - с функция за инициализация на Uartlite и изпращане на символ;
- *printf/printf.c – с имплементация на минималистична printf функция.

Нека всяка директория си има Makefile. Създайте един основен Makefile в top-level директорията, който да вика отделните Makefile-ове [2]. Препоръчително е да използвате променливата \${MAKE} вместо командата make за рекурсивните

target-и [3].

За да спрете GCC на ниво компилатор (без да линкувате), използвайте аргументът -с. Изходният файл тогава трябва да бъде обектов файл с разширение “.o”, като например:

```
mb-gcc -I../include -mlittle-endian -mxl-barrel-shift -mxl-  
pattern-compare -mcpu=v11.0 -mno-xl-soft-mul -c led.c -o  
../debug/led.o
```

Добавете пътищата до отделните C файлове посредством аргумента -I, когато компилирате main.c:

```
./debug/main.elf: ./debug/led.o ./debug/uart.o ./debug/print.o  
main.c  
mb-gcc -Wl,-T -Wl,ldscript.ld -Wl,--no-relax -Wl,--gc-sections  
-I../include -I../led -I../uart -I../print -L./ -lxil -mlittle-endian  
-mxl-barrel-shift -mxl-pattern-compare -mcpu=v11.0 -mno-xl-soft-  
mul main.c ./debug/led.o ./debug/uart.o ./debug/print.o -o  
./debug/main.elf
```

8. Направете примерът ви да може да се конфигурира от графичната програма (от командния ред) – lxdialog (mconf), която използва ncurses библиотеката. Тя създава конфигурационен файл .config, който се преобразува в хедърен файл от програмата conf. Този хедър ще се използва, за да конфигурира програмата ви. За целта първо трябва да компилирате mconf и conf. Копирайте директорията menuconfig [4] от директорията 18_8 в top-level директорията на проекта ви. След това изпълнете командите:

```
cd menuconfig/  
mkdir build  
cd ./build  
cmake ../  
make
```

(ако инсталацията ви е нова може да се наложи да инсталирате ncurses: sudo apt-get install libncurses-dev)

Ако всичко е минало без грешки изпробвайте програмата като изпълните командата:

```
./mconf ../test/rootconf
```

Преобразуването на автоматично генерираният файл menuconfig/build/.config в хедърен файл става посредством командата (прието е този хедър да се нарича autoconf.h) [5]:

```
./conf ../../Kconfig --silentoldconfig autoconf.h
```

9. Разгледайте структурата на файла `menuconfig/test/rootconf`. Опитайте се да създадете аналогичен файл за вашия проект с име `Kconfig`, който да конфигурира следните модули:

`*led` – възможност за контрол на паузите между премигванията (50000, 500000, 2000000 микросекунди);

`*print` – възможност за промяна на съобщение (по ваш избор) в `main`.

За целта направете параметрите на `usleep()` и `printf()` с макроси. Имената на тези макроси трябва да съответстват на имената в генерирания `autoconf.h`. Не забравяйте да включите:

```
#include "autoconf.h"
```

във вашата програма.

ВНИМАНИЕ: понеже `menuconfig target-a` е команда, която не зависи от друг файл и трябва да се изпълнява винаги, направете я `"PHONY"` [6]:

```
.PHONY: menuconfig
```

Без този ред в `Makefile`-а винаги ще получавате следното съобщение:

```
user@computer:~/01_09$ make menuconfig
make: `menuconfig' is up to date.
```

10. Документирайте кода с `Doxygen` коментари (програмата `Doxygen` може да се инсталира в `Ubuntu Linux` със: `sudo apt-get install doxygen texlive-latex-base texlive-latex-recommended texlive-fonts-recommended doxygen-latex`) [7]. Основната структура на един такъв коментар изглежда така:

```
/*!
 * \brief Това е кратко описание на функцията.
 *
 * През един ред следва детайлно описание на функцията
 * заедно с всичките ѝ особености и тънкости. Ако тази
 * функция засяга други, те могат да бъдат изредени тук.
 * Ако тази функция засяга глобални променливи, добре е
 * да се опише това. Може да се използват и HTML тагове.
 * Например \bтази дума ще е засветена.
 *
 * \param my_value – кратко описание на параметъра
 *
```

```

* \return 1 - успех, 0 - неуспех.
*/
int my_function(char my_value){
    ...
}

/*!
* \struct my_structure
* \brief Това е кратко описание на променлива от тип
* структурата
*
* През един ред следва детайлно описание.
*
*/
struct my_structure;

```

Други типове, файлове, функции и полета също могат да се документират: `\struct`, `\union`, `\enum`, `\fn`, `\var`, `\def`, `\typedef`, `\file`, `\namespace`.

След като документирате всички променливи, функции и макроси, отворете команден ред, преместете се в top-level директорията на проекта ви и изпълнете:

```

doxygen -g
doxygen Doxyfile
cd ./latex
make

```

Документацията ще се генерира в директорията `latex` и ще е с име `refman.pdf`. На този етап това ще е един празен файл.

11. Използвайте графичната програма Doxywizard (може да се инсталира в Ubuntu Linux със: `sudo apt-get install doxygen-gui`), за да конфигурирате форматирането на вашия PDF файл. За целта влезте в top-level директорията ви и напишете:

```
doxywizard Doxyfile
```

Следвайте менютата на doxywizard. Добавете логото на ТУ-София от директория `18_11`. Като изходна директория за документацията изберете `doxygen_gui`. Когато сте готови, на таб-а Run натиснете Run doxygen. Затворете графичното приложение, влезте в съответната директория и стартирайте Make:

```

cd doxygen_gui/latex/
make

```


Исходният файл е с име refman.pdf. Ако е избрана опцията за рисуване на дърво на извикванията на функциите, и ако е избрана програмата Graphviz, последната трябва да бъде инсталирана със `sudo apt install graphviz`.

В директорията doxygen_gui/html можете да намерите уеб страница (щракнете двукратно върху index.html), аналогична на PDF файлът refman.pdf. Тази страница може да бъде предоставена за ползване в Интернет, ако библиотеката ви е с отворен код.

*

*

*

[1] “Xilinx Software Command-Line Tool (XSCT)”, Reference Guide, UG1208, (v2016.4), 2016.

[2] https://www.gnu.org/software/make/manual/html_node/Recursion.html

[3] https://www.gnu.org/software/make/manual/html_node/MAKE-Variable.html#MAKE-Variable

[4] <https://github.com/TheGeorge/menuconfig>

[5] <https://johnvidler.co.uk/linux-journal/LJ/222/11333.html>

[6] https://www.gnu.org/software/make/manual/html_node/Phony-Targets.html

[7] <https://www.doxygen.nl/manual/commands.html#cmdcond>

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