

Національний університет "Львівська політехніка"  
Кафедра електронних обчислювальних машин (ЕОМ)

---

# Автоматизоване проектування комп'ютерних та кіберфізичних систем

спеціальність 123 "Комп'ютерна інженерія"  
спеціалізація 123.04 "Кіберфізичні системи"  
4-ий курс

Лекція 9. Високорівневі засоби системного проектування (7)

2021

## 01.1. RISC-V.

---

RISC-V (вимовляється "risk-five") — відкрита архітектура інструкцій центрального процесора, що базується на принципах RISC.

Набір процесорних інструкцій DLX з'явився у 1990-му році для першого видання книги Computer Architecture: A Quantitative Approach і позиціонувався в основному для навчальних цілей. Автором розробки був Девід Паттерсон; у академічних колах і серед ентузіастів було здійснено кілька реалізацій DLX для FPGA. Комерційного застосування DLX не мав, але його розвитком став RISC-V.

Існує також дизайн OpenRISC (також базується на DLX), що є продуктом з відкритим кодом і також підтримується гсс. Втім, кількість комерційних реалізацій OpenRISC незначна.

## 01.2. RISC-V.



### RISC-V Ecosystem

#### Software

##### Open-source software:

Gcc, binutils, glibc, Linux, BSD, LLVM, QEMU, FreeRTOS, ZephyrOS, LiteOS, SylixOS, ...

##### Commercial software:

Lauterbach, Segger, IAR, Micrium, ExpressLogic, Ashling, AntMicro, Imperas, UltraSoC ...



ISA specification

Golden Model

Compliance

#### Hardware

##### Open-source cores:

Rocket, BOOM, RI5CY, Ariane, PicoRV32, Piccolo, SCR1, Shakti, Swerv, Hummingbird, ...

##### Commercial core providers:

Andes, Bluespec, Cloudbear, Codaip, Cortus, C-Sky, InCore, Nuclei, SiFive, Syntacore, ...

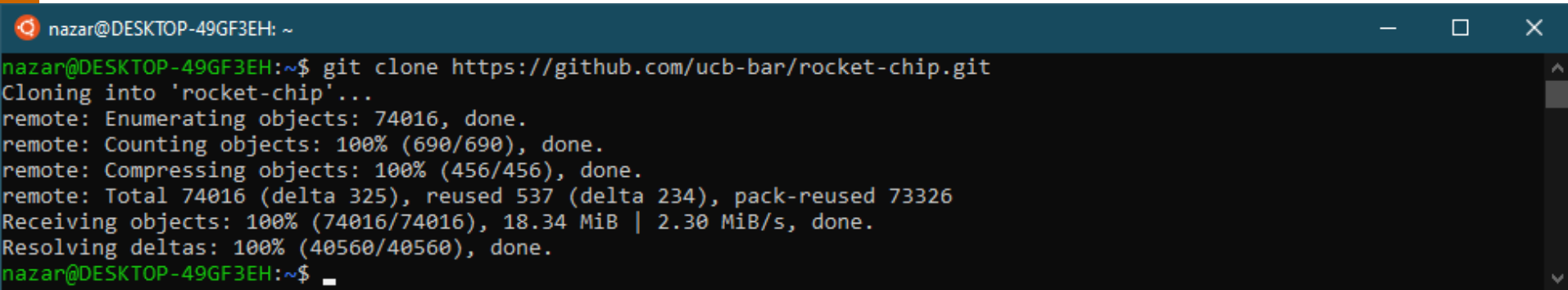
##### Inhouse cores:

Nvidia, +others

## 02.1. Приклад автоматизованого проектування систем на основі RISC-V.

---

# Скачуємо та ініціалізуємо репозиторій Rocket Chip  
**git clone https://github.com/ucb-bar/rocket-chip.git**

A terminal window with a dark background and light green text. The title bar shows 'nazar@DESKTOP-49GF3EH: ~'. The command 'git clone https://github.com/ucb-bar/rocket-chip.git' has been executed. The output shows the cloning process: 'Cloning into 'rocket-chip'...', 'remote: Enumerating objects: 74016, done.', 'remote: Counting objects: 100% (690/690), done.', 'remote: Compressing objects: 100% (456/456), done.', 'remote: Total 74016 (delta 325), reused 537 (delta 234), pack-reused 73326', 'Receiving objects: 100% (74016/74016), 18.34 MiB | 2.30 MiB/s, done.', and 'Resolving deltas: 100% (40560/40560), done.'. The prompt 'nazar@DESKTOP-49GF3EH:~\$' is visible at the bottom.

```
nazar@DESKTOP-49GF3EH: ~  
nazar@DESKTOP-49GF3EH:~$ git clone https://github.com/ucb-bar/rocket-chip.git  
Cloning into 'rocket-chip'...  
remote: Enumerating objects: 74016, done.  
remote: Counting objects: 100% (690/690), done.  
remote: Compressing objects: 100% (456/456), done.  
remote: Total 74016 (delta 325), reused 537 (delta 234), pack-reused 73326  
Receiving objects: 100% (74016/74016), 18.34 MiB | 2.30 MiB/s, done.  
Resolving deltas: 100% (40560/40560), done.  
nazar@DESKTOP-49GF3EH:~$
```

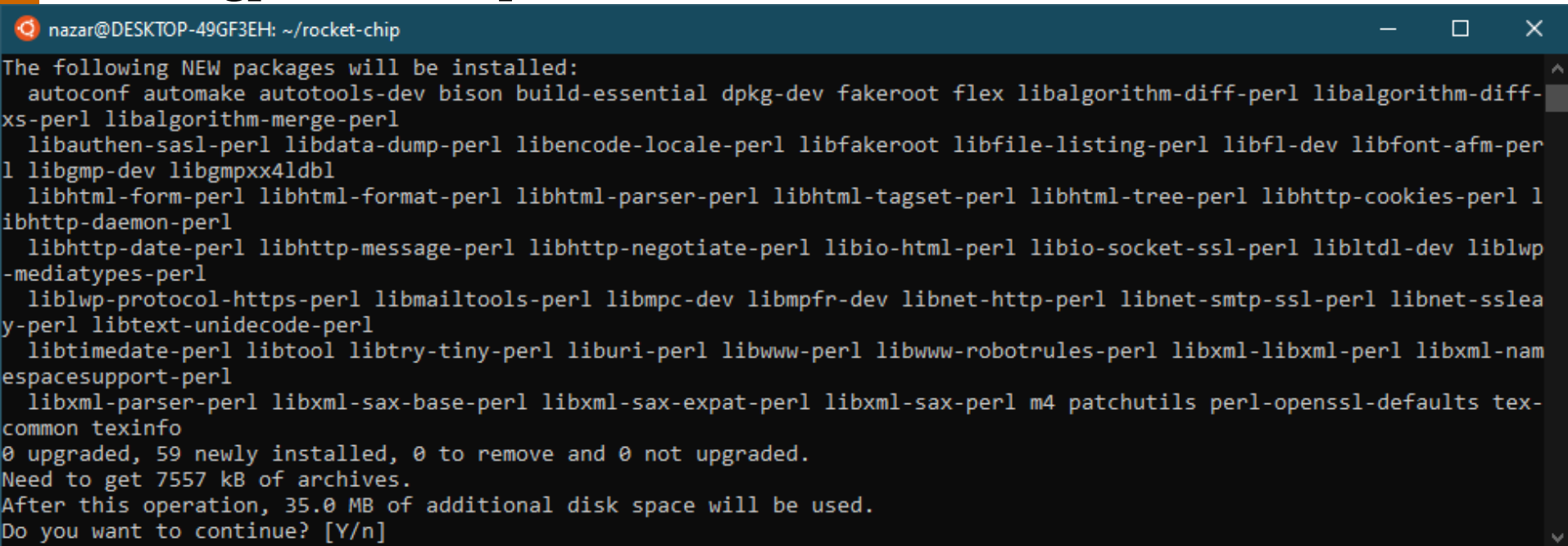
**cd rocket-chip**  
**git submodule update --init**  
**export TOP=\$(pwd)**

## 02.2. Приклад автоматизованого проектування систем на основі RISC-V.

---

# Інсталює додаткові пакети (Ubuntu)

```
sudo apt-get install autoconf automake autotools-dev curl \  
libmpc-dev libmpfr-dev libgmp-dev gawk build-essential bison flex \  
texinfo gperf libtool patchutils bc
```

A terminal window with a dark blue title bar showing the user 'nazar@DESKTOP-49GF3EH' in the directory '~/rocket-chip'. The terminal displays the output of the 'sudo apt-get install' command, listing various packages to be installed, including autoconf, automake, autotools-dev, bison, build-essential, dpkg-dev, fakeroot, flex, libalgorithm-diff-perl, libalgorithm-diff-xs-perl, libalgorithm-merge-perl, libauthen-sasl-perl, libdata-dump-perl, libencode-locale-perl, libfakeroot, libfile-listing-perl, libfl-dev, libfont-afm-perl, libgmp-dev, libgmpxx4ldbl, libhtml-form-perl, libhtml-format-perl, libhtml-parser-perl, libhtml-tagset-perl, libhtml-tree-perl, libhttp-cookies-perl, libhttp-daemon-perl, libhttp-date-perl, libhttp-message-perl, libhttp-negotiate-perl, libio-html-perl, libio-socket-ssl-perl, libltdl-dev, liblwp-mediatypes-perl, liblwp-protocol-https-perl, libmailtools-perl, libmpc-dev, libmpfr-dev, libnet-http-perl, libnet-smtp-ssl-perl, libnet-ssleay-perl, libtext-unidecode-perl, libtimedate-perl, libtool, libtry-tiny-perl, liburi-perl, libwww-perl, libwww-robotrules-perl, libxml-libxml-perl, libxml-namespacesupport-perl, libxml-parser-perl, libxml-sax-base-perl, libxml-sax-expat-perl, libxml-sax-perl, m4, patchutils, perl-openssl-defaults, tex-common, and texinfo. It also shows the disk space requirements and asks for confirmation to continue.

```
nazar@DESKTOP-49GF3EH: ~/rocket-chip  
The following NEW packages will be installed:  
autoconf automake autotools-dev bison build-essential dpkg-dev fakeroot flex libalgorithm-diff-perl libalgorithm-diff-  
xs-perl libalgorithm-merge-perl  
libauthen-sasl-perl libdata-dump-perl libencode-locale-perl libfakeroot libfile-listing-perl libfl-dev libfont-afm-per  
l libgmp-dev libgmpxx4ldbl  
libhtml-form-perl libhtml-format-perl libhtml-parser-perl libhtml-tagset-perl libhtml-tree-perl libhttp-cookies-perl l  
ibhttp-daemon-perl  
libhttp-date-perl libhttp-message-perl libhttp-negotiate-perl libio-html-perl libio-socket-ssl-perl libltdl-dev liblwp  
-mediatypes-perl  
liblwp-protocol-https-perl libmailtools-perl libmpc-dev libmpfr-dev libnet-http-perl libnet-smtp-ssl-perl libnet-sslea  
y-perl libtext-unidecode-perl  
libtimedate-perl libtool libtry-tiny-perl liburi-perl libwww-perl libwww-robotrules-perl libxml-libxml-perl libxml-nam  
espacesupport-perl  
libxml-parser-perl libxml-sax-base-perl libxml-sax-expat-perl libxml-sax-perl m4 patchutils perl-openssl-defaults tex-  
common texinfo  
0 upgraded, 59 newly installed, 0 to remove and 0 not upgraded.  
Need to get 7557 kB of archives.  
After this operation, 35.0 MB of additional disk space will be used.  
Do you want to continue? [Y/n]
```

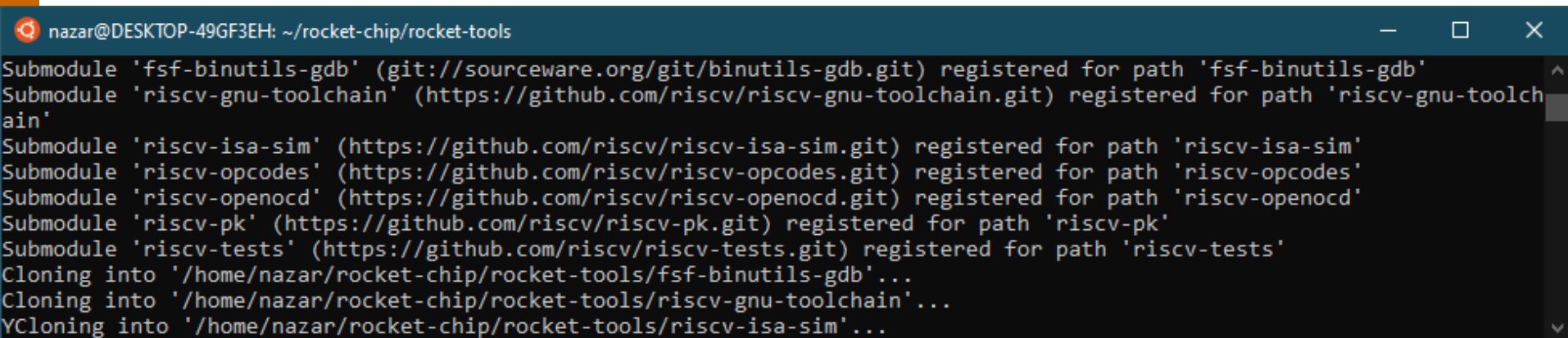
```
sudo apt install default-jre
```

```
sudo apt install python
```

## 02.3. Приклад автоматизованого проектування систем на основі RISC-V.

---

```
# Скачуємо та ініціалізуємо rocket-tools для Rocket Chip
git clone https://github.com/freechipsproject/rocket-tools
cd rocket-tools
git submodule update --init --recursive
```



```
nazar@DESKTOP-49GF3EH: ~/rocket-chip/rocket-tools
Submodule 'fsf-binutils-gdb' (git://sourceware.org/git/binutils-gdb.git) registered for path 'fsf-binutils-gdb'
Submodule 'riscv-gnu-toolchain' (https://github.com/riscv/riscv-gnu-toolchain.git) registered for path 'riscv-gnu-toolchain'
Submodule 'riscv-isa-sim' (https://github.com/riscv/riscv-isa-sim.git) registered for path 'riscv-isa-sim'
Submodule 'riscv-opcodes' (https://github.com/riscv/riscv-opcodes.git) registered for path 'riscv-opcodes'
Submodule 'riscv-openocd' (https://github.com/riscv/riscv-openocd.git) registered for path 'riscv-openocd'
Submodule 'riscv-pk' (https://github.com/riscv/riscv-pk.git) registered for path 'riscv-pk'
Submodule 'riscv-tests' (https://github.com/riscv/riscv-tests.git) registered for path 'riscv-tests'
Cloning into '/home/nazar/rocket-chip/rocket-tools/fsf-binutils-gdb'...
Cloning into '/home/nazar/rocket-chip/rocket-tools/riscv-gnu-toolchain'...
Cloning into '/home/nazar/rocket-chip/rocket-tools/riscv-isa-sim'...
```

## 02.4. Приклад автоматизованого проектування систем на основі RISC-V.

---

```
# Компілюємо тулчейн RISC-V
# (застосовується Newlib для роботи без ОС)
# Редагуємо файл build.common
# JOBS = Кількість ядер/потоків, які дозволяє ваша машина
./build.sh
export RISC_V=$TOP/rocket-tools/riscv-gnu-toolchain
cd $RISC_V
git submodule update --init --recursive
```

## 02.5. Приклад автоматизованого проектування систем на основі RISC-V. Проста програма.

---

```
# За допомогою тулчейну RISC-V компілюємо просту програму
# та запускаємо її на Spike (симулятор ISA)
# використовуючи наявне проксі-ядро (pk)
cd $TOP
echo -e '#include <stdio.h>\n int main(void) \
{ printf("Hello world!\\n"); return 0; }' > hello.c
riscv64-unknown-elf-gcc -o hello hello.c
spike pk hello
```



## 02.6. Приклад автоматизованого проектування систем на основі RISC-V. Компіляція емулятора.

---

```
# Компілюємо емулятор Rocket Chip
# Замінити -j8 на -jN (де N кількість ядер/потоків, що доступні вашій машині)
cd $TOP/emulator
make
# Запускаємо тести
# (може зайняти тривалий проміжок часу)
# make -j8 run-asm-tests
# make -j8 run-bmark-tests
# можна запустити лише один окремий тест:
make output/rv64ui-p-add.out
```

## 02.7. Приклад автоматизованого проектування систем на основі RISC-V. Компіляція емулятора(2).

---

```
# Компілюємо емулятор Rocket Chip
# Замінити -j8 на -jN (де N кількість ядер/потоків, що доступні вашій машині)
cd $TOP/emulator
# Якщо відсутня утиліта vcd2vpd,
# відкриваємо src/main/scala/Testing.scala файл
# та міняємо всі .vpd на .vcd (6 раз)
make debug
# Запускаємо тести
# (може зайняти тривалий проміжок часу та багато місця на диску)
# make -j8 run-asm-tests-debug
# make -j8 run-bmark-tests-debug
# можна запустити лише один окремий тест
# (тоді міняти нічого не треба, якщо відсутня утиліта vcd2vpd):
make output/rv64ui-p-add.vcd
```

## 02.8. Приклад автоматизованого проектування систем на основі RISC-V. Генерація RTL-моделі.

---

```
# Генеруємо Rocket Chip Verilog RTL для FPGAs
# (в fsim/generated-src/)
# Default-конфігурація є "DefaultFPGAConfig",
# яку не можна змінити (наприклад, на "SmallFPGAConfig"),
# наприклад, якщо Default-реалізація з FPU
# не поміщається на кристал.
# Тоді використовуємо make CONFIG=ConfigName verilog
# для застосування бажаної конфігурації
# з набору доступних (src/main/scala/Configs.scala)
# Результируючий вивід Verilog RTL називається Top.ConfigName.v
cd $TOP/vsim
make verilog
```