国立大学法人
電気通信大学
The University of Electro-Communications

Pham Laboratory
Integrated circuit design laboratory

「Course」
RISC-V Computer System Integration

「Lecture 3」Rocket Computer System: Introduction and System Modification

Hoang Trong Thuc

2022/11

Outline

1. Introduction
2. System architecture
3. Git clone and prepare
4. Make the system
5. Program Arty-A7
6. Using IntelliJ IDEA-IC
7. Modifying system by Scala config
8. Practice: system modification

Outline

1. Introduction
2. System architecture
3. Git clone and prepare
4. Make the system
5. Program Arty-A7
6. Using IntelliJ IDEA-IC
7. Modifying system by Scala config
8. Practice: system modification

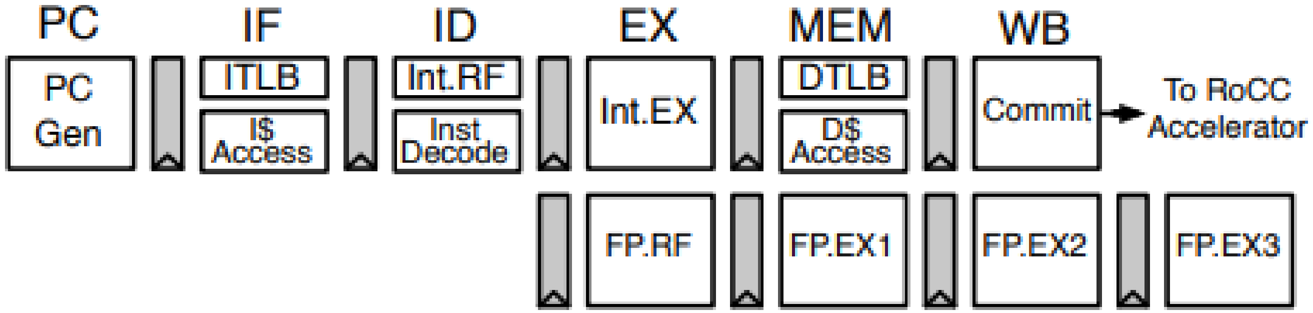
1. Introduction (1/4) Rocket core

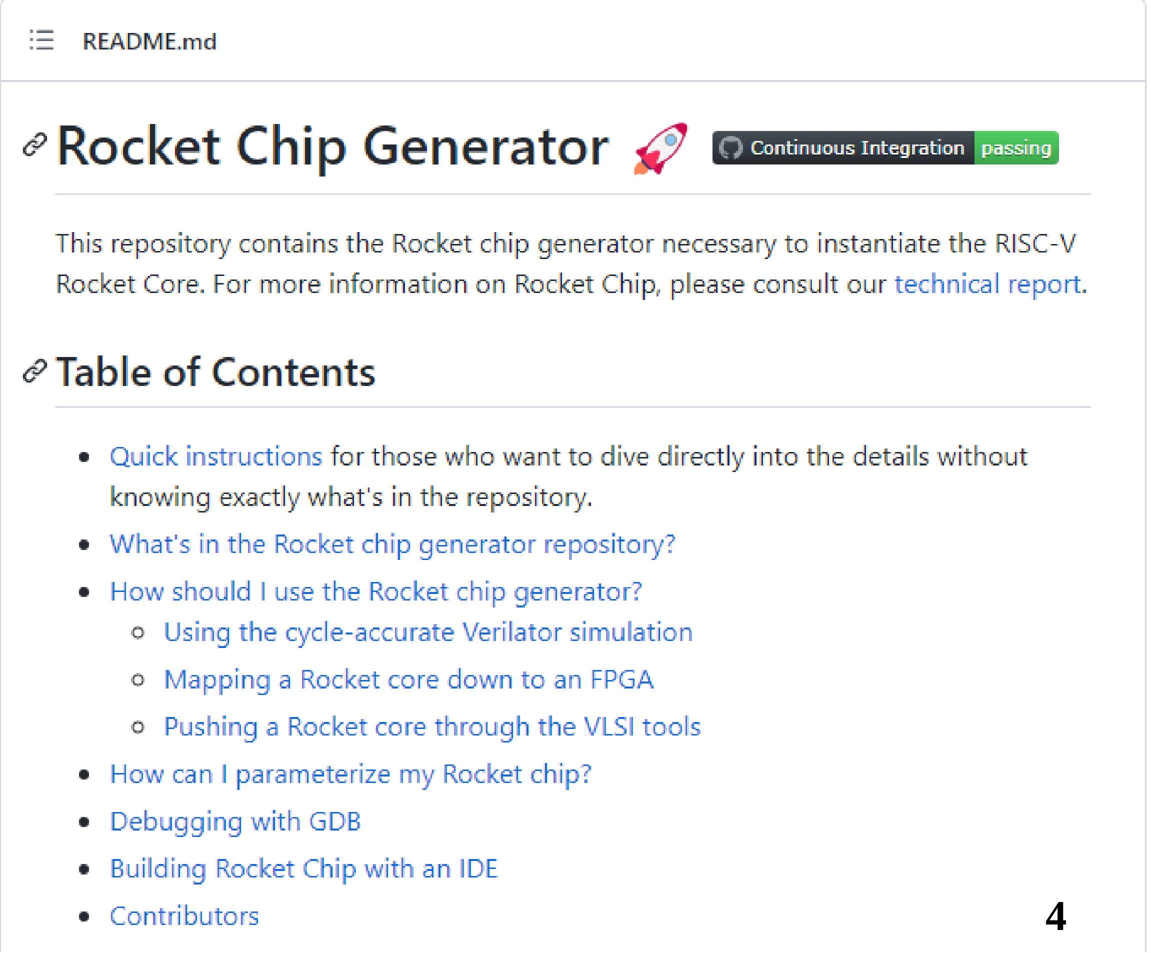
Rocket core

- Github: https://github.com/chipsalliance/rocket-chip
- Document: https://chipyard.readthedocs.io/en/stable/Generators/Rocket-Chip.html
  https://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.pdf

Rocket is the most popular processor in RISC-V community.

Rocket is a 5-stage *in-of-order* processor. Its pipeline architecture:





README.md

Rocket Chip Generator 🚀 Continuous Integration passing

This repository contains the Rocket chip generator necessary to instantiate the RISC-V Rocket Core. For more information on Rocket Chip, please consult our technical report.

Table of Contents

- Quick instructions for those who want to dive directly into the details without knowing exactly what's in the repository.
- What's in the Rocket chip generator repository?
- How should I use the Rocket chip generator?
  - Using the cycle-accurate Verilator simulation
  - Mapping a Rocket core down to an FPGA
  - Pushing a Rocket core through the VLSI tools
- How can I parameterize my Rocket chip?
- Debugging with GDB
- Building Rocket Chip with an IDE
- Contributors

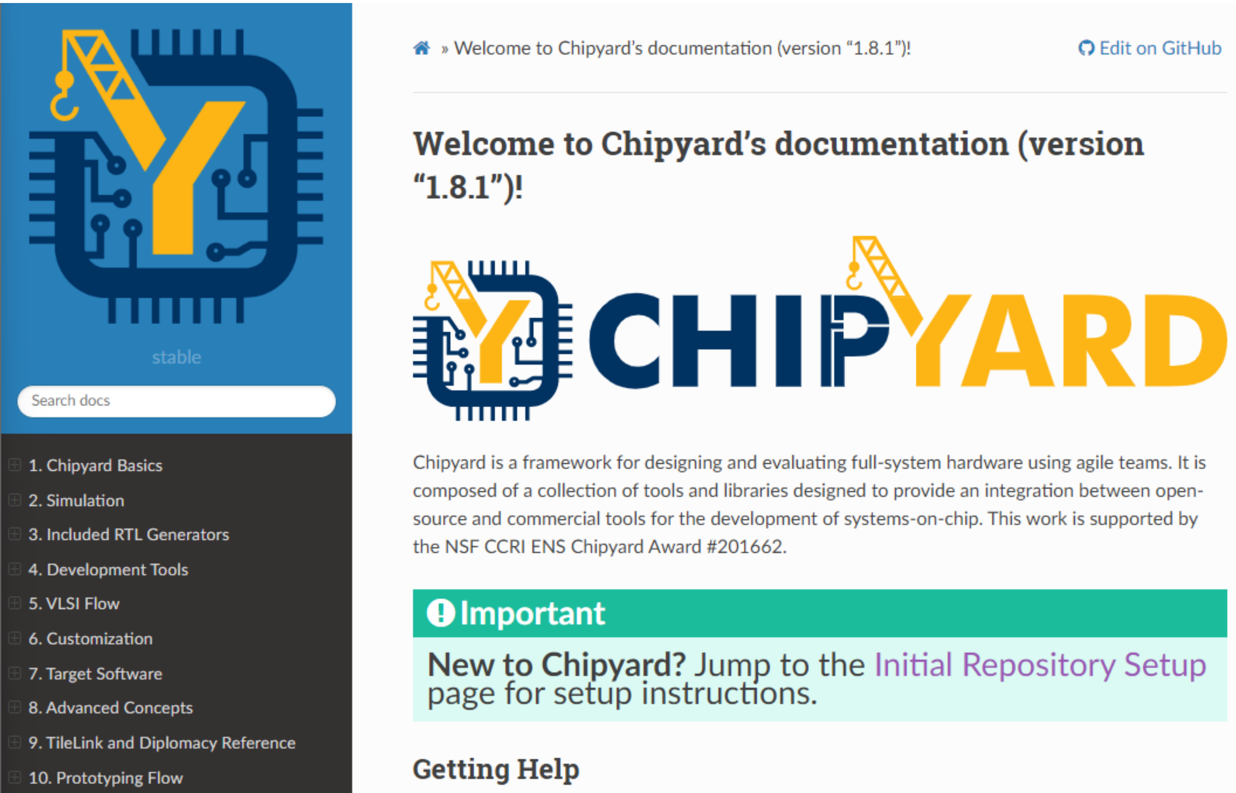
1. Introduction (2/4) Chipyard

Chipyard library

- Github: https://github.com/ucb-bar/chipyard
- Document: https://chipyard.readthedocs.io/en/stable/

You can see that there are many processors ready for use.

We will use the Rocket core in here.



stable

Search docs

1. Chipyard Basics
2. Simulation
3. Included RTL Generators
4. Development Tools
5. VLSI Flow
6. Customization
7. Target Software
8. Advanced Concepts
9. TileLink and Diplomacy Reference
10. Prototyping Flow

» Welcome to Chipyard's documentation (version "1.8.1")!

Edit on GitHub

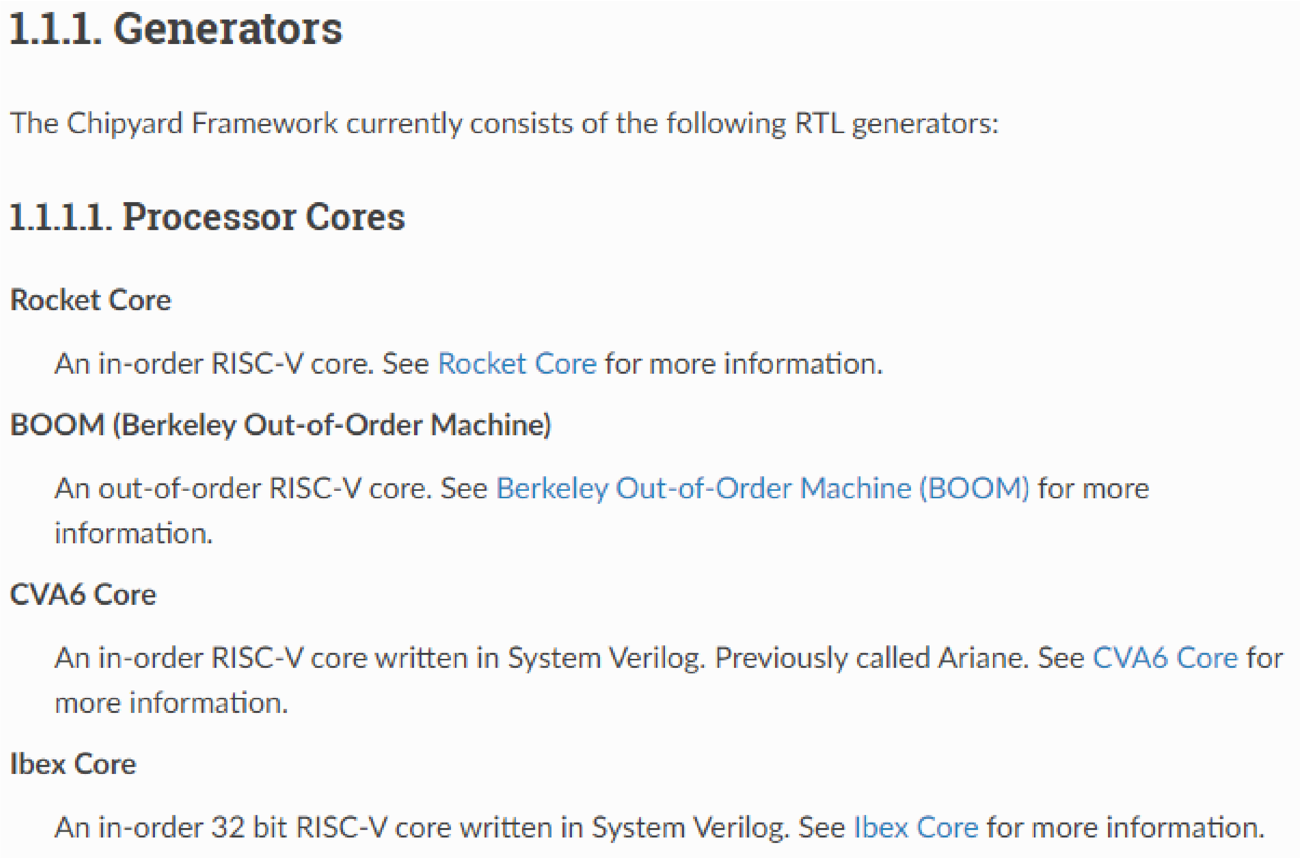
Welcome to Chipyard's documentation (version "1.8.1")!

Chipyard is a framework for designing and evaluating full-system hardware using agile teams. It is composed of a collection of tools and libraries designed to provide an integration between open-source and commercial tools for the development of systems-on-chip. This work is supported by the NSF CCRI ENS Chipyard Award #201662.

Important

New to Chipyard? Jump to the Initial Repository Setup page for setup instructions.

Getting Help



1.1.1. Generators

The Chipyard Framework currently consists of the following RTL generators:

1.1.1.1. Processor Cores

Rocket Core

An in-order RISC-V core. See Rocket Core for more information.

BOOM (Berkeley Out-of-Order Machine)

An out-of-order RISC-V core. See Berkeley Out-of-Order Machine (BOOM) for more information.

CVA6 Core

An in-order RISC-V core written in System Verilog. Previously called Ariane. See CVA6 Core for more information.

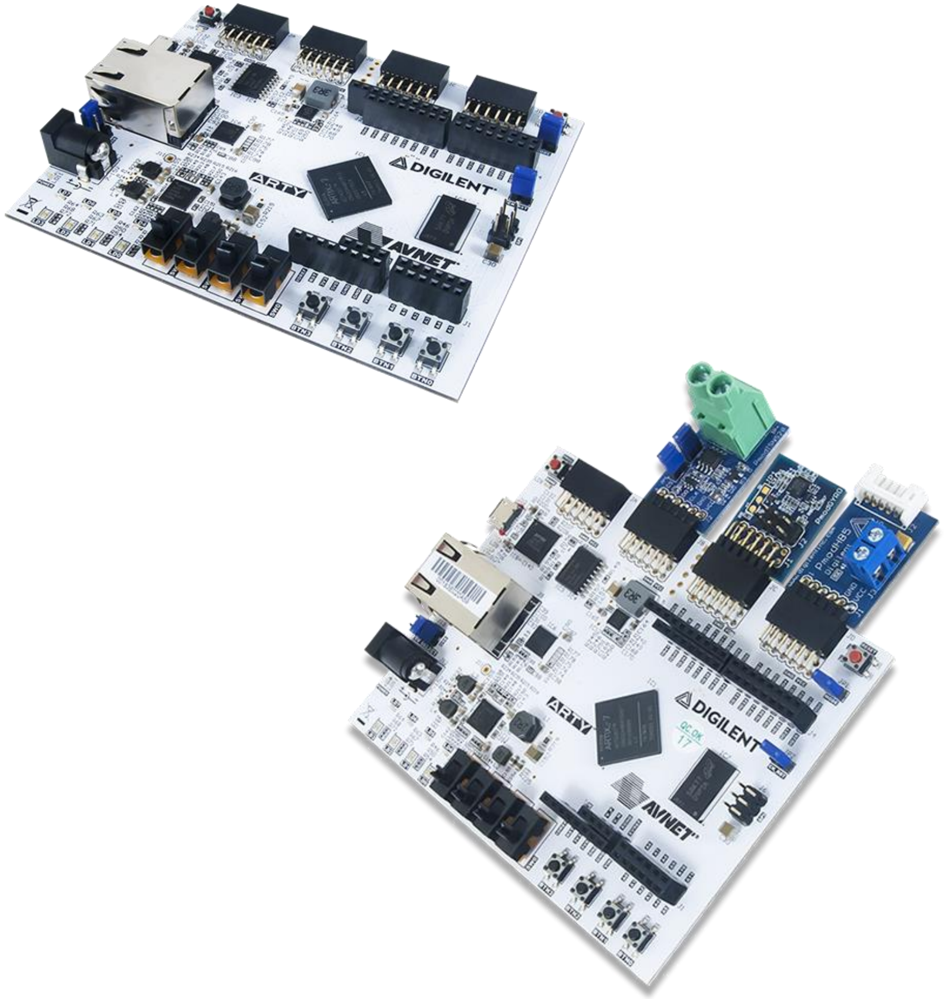
Ibex Core

An in-order 32 bit RISC-V core written in System Verilog. See Ibex Core for more information.

1. Introduction (3/4) Arty-A7 FPGA

Arty-A7 FPGA

- Link: https://digilent.com/reference/programmable-logic/arty-a7/start



We will use Arty-A7 FPGA for this course

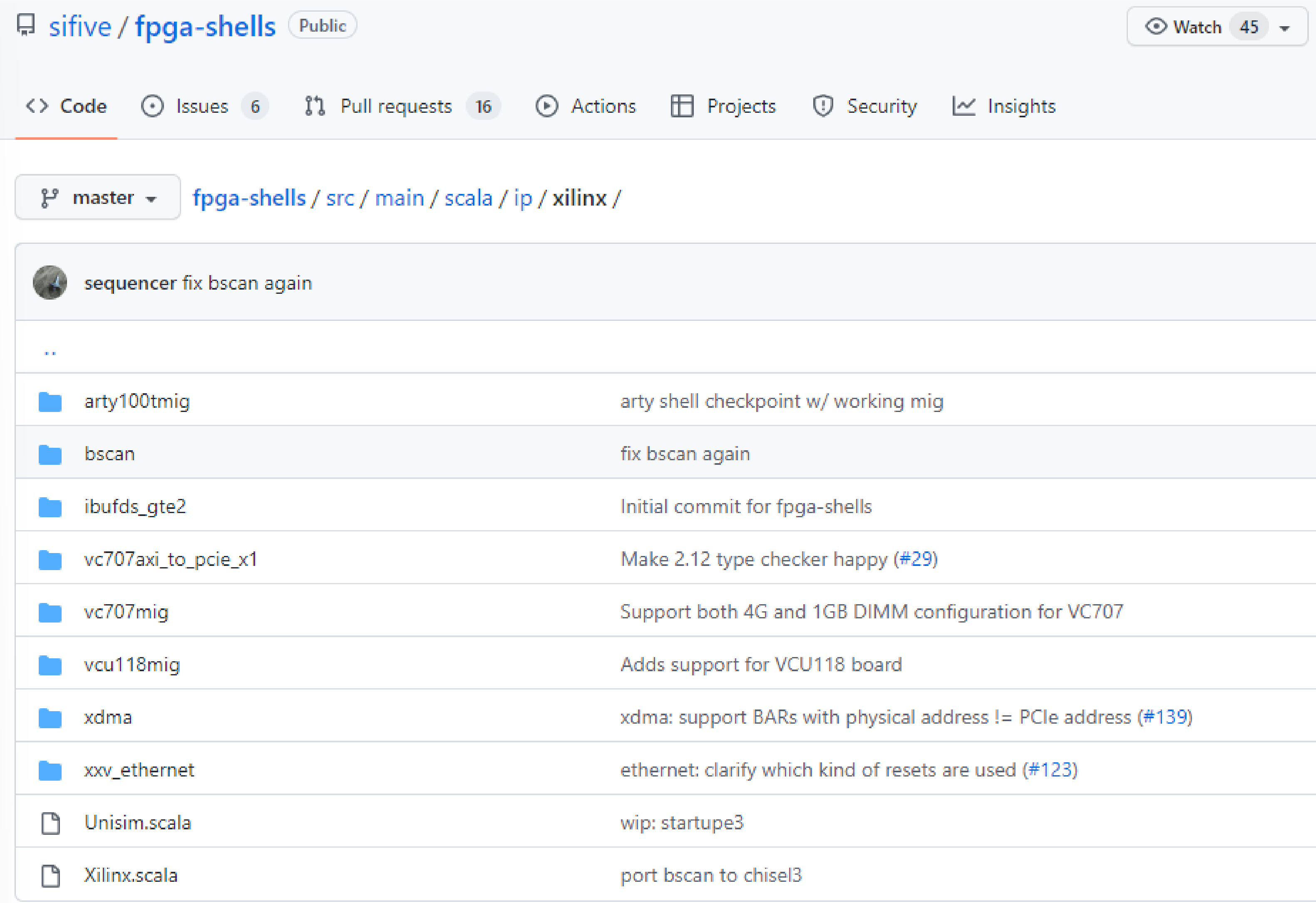
- A Xilinx FPGA
- Relatively cheap
- License-free to build *(license is provided by Digilent, not Xilinx)*
- Convenient with PMOD headers *(easy-to-use SD-card, Flash, etc.)*
- Has two versions of FPGA: **35T** and **100T** *(be careful, you have to check the FPGA version before compiling)*

1. Introduction (4/4) FPGA-shells

FPGA-shells library

- Github: https://github.com/sifive/fpga-shells

Common FPGA IPs can be found in here. *We will use the Arty's memory IP in here.*



sifive / fpga-shells Public

fpga-shells / src / main / scala / ip / xilinx /

sequencer fix bscan again

..

arty100tmig	arty shell checkpoint w/ working mig
bscan	fix bscan again
ibufds_gte2	Initial commit for fpga-shells
vc707axi_to_pcie_x1	Make 2.12 type checker happy (#29)
vc707mig	Support both 4G and 1GB DIMM configuration for VC707
vcu118mig	Adds support for VCU118 board
xdma	xdma: support BARs with physical address != PCIe address (#139)
xxv_ethernet	ethernet: clarify which kind of resets are used (#123)
Unisim.scala	wip: startupe3
Xilinx.scala	port bscan to chisel3

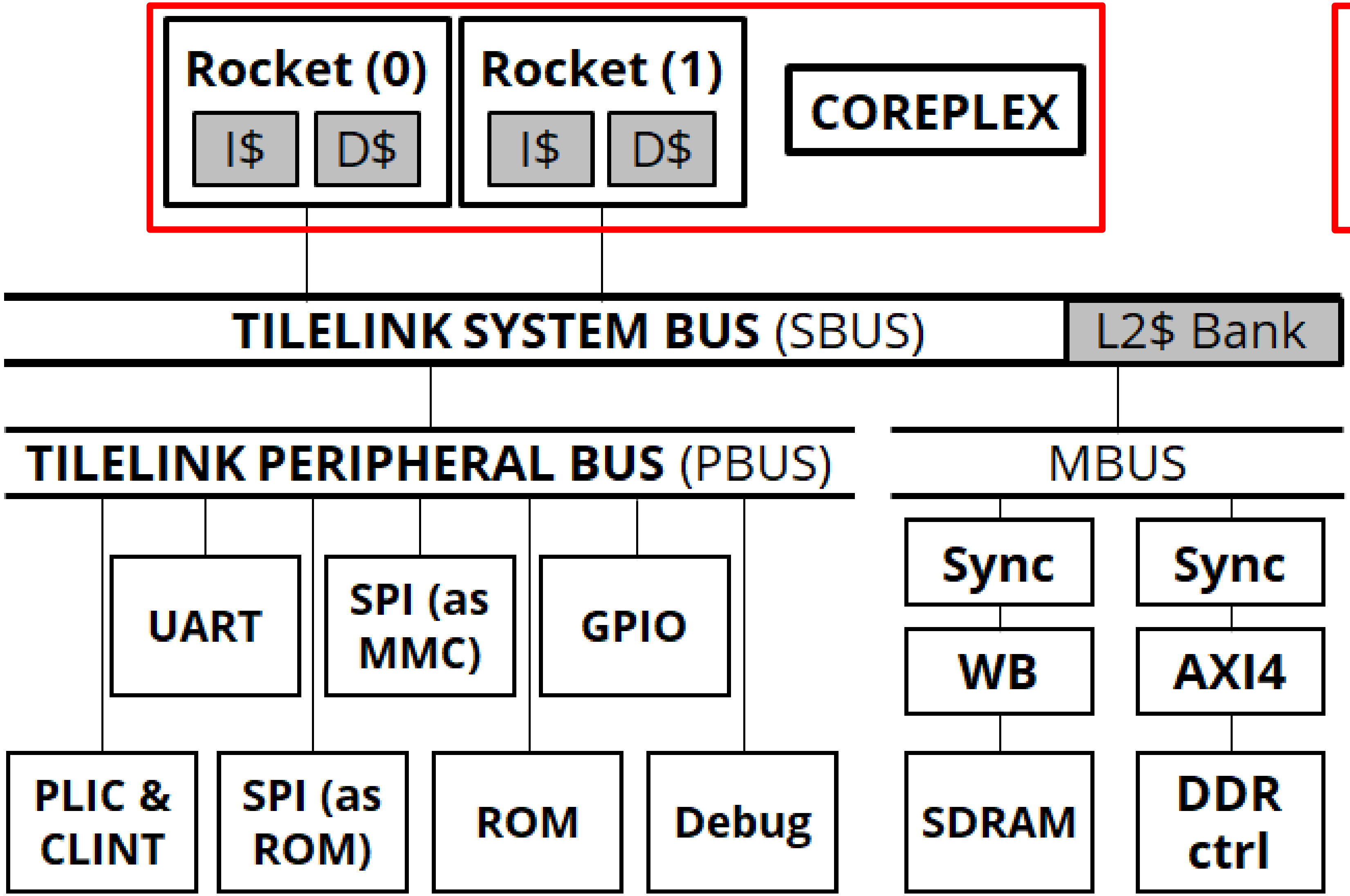
Outline

1. Introduction
2. System architecture
3. Git clone and prepare
4. Make the system
5. Program Arty-A7
6. Using IntelliJ IDEA-IC
7. Modifying system by Scala config
8. Practice: system modification

2. System architecture (1/9) Processor

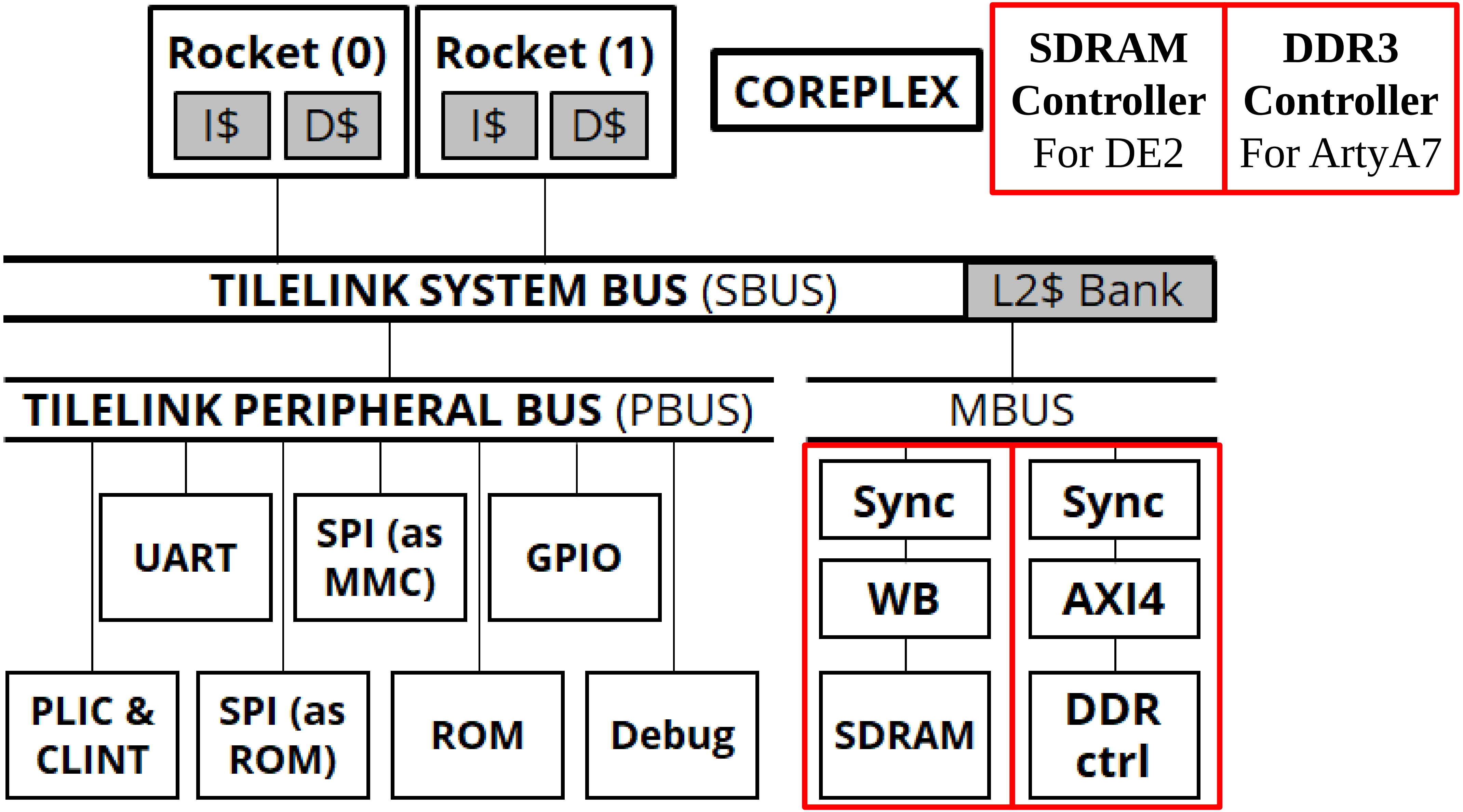
In this course, we will working with an example of Rocket computer system at here:

- Github: https://github.com/uec-hanken/RISCVConsole

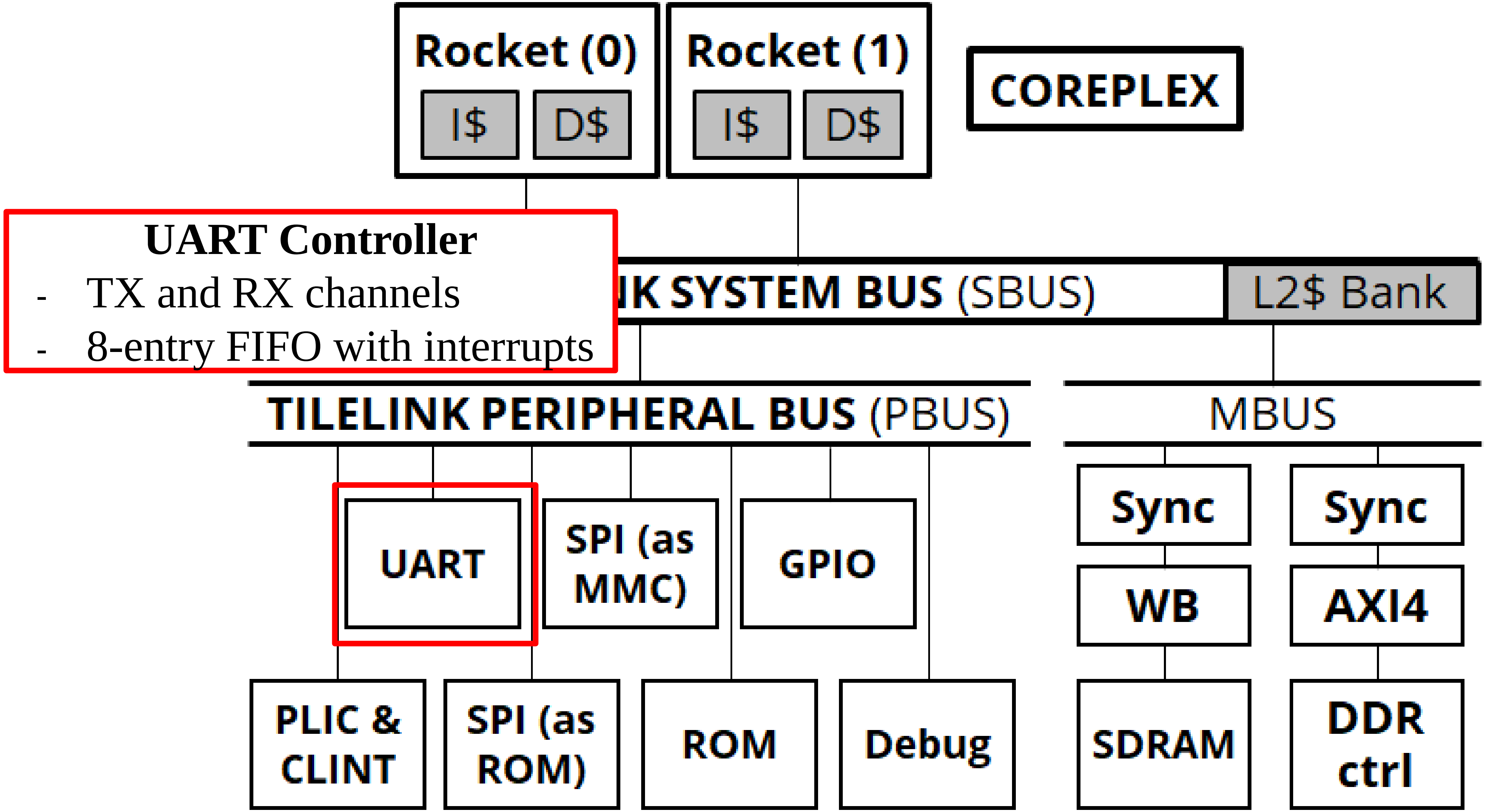


Coreplex
RISC-V 32/64
IMAFDC

2. System architecture (2/9) Memory



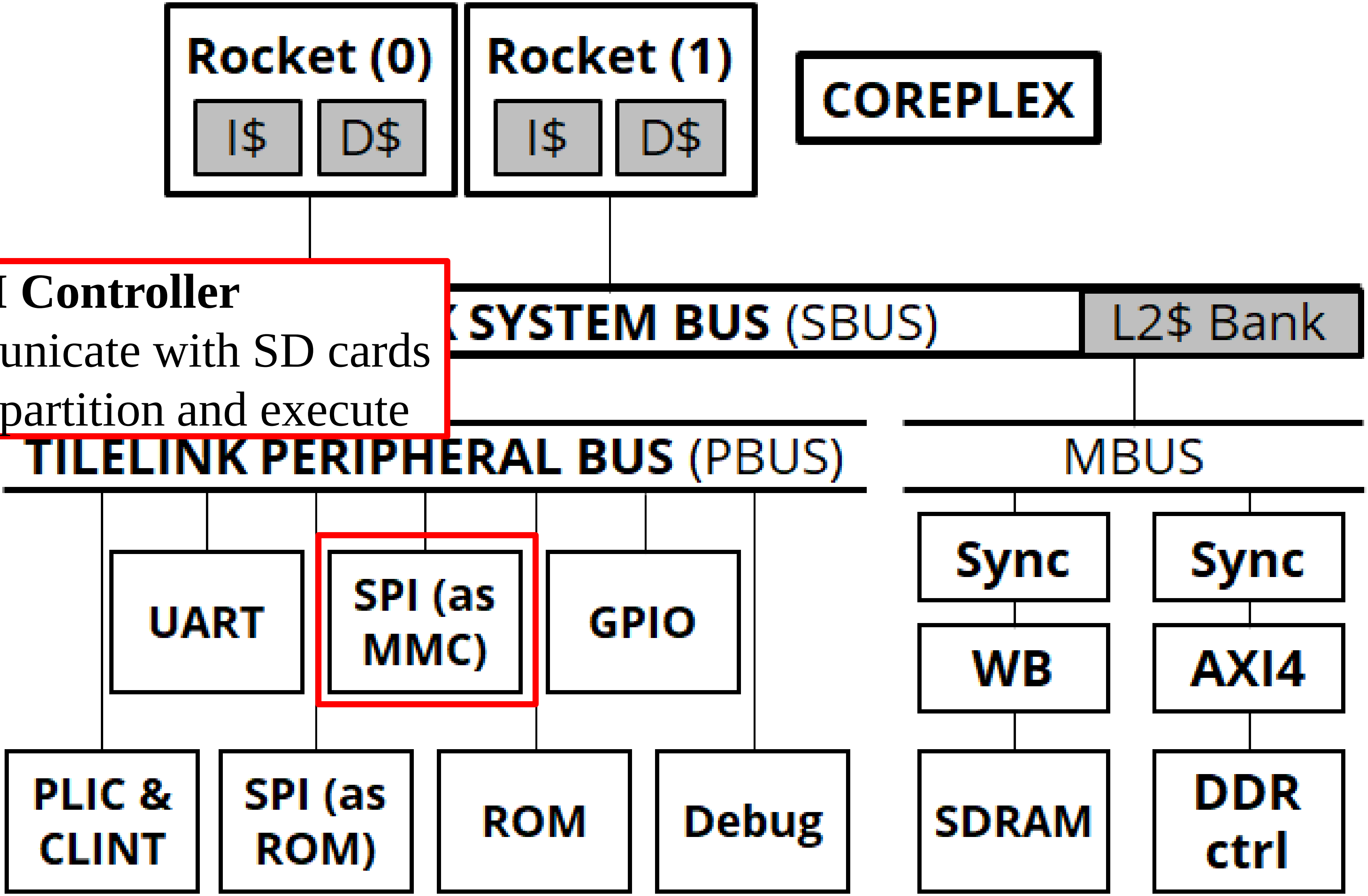
2. System architecture (3/9) UART



UART Controller

- TX and RX channels
- 8-entry FIFO with interrupts

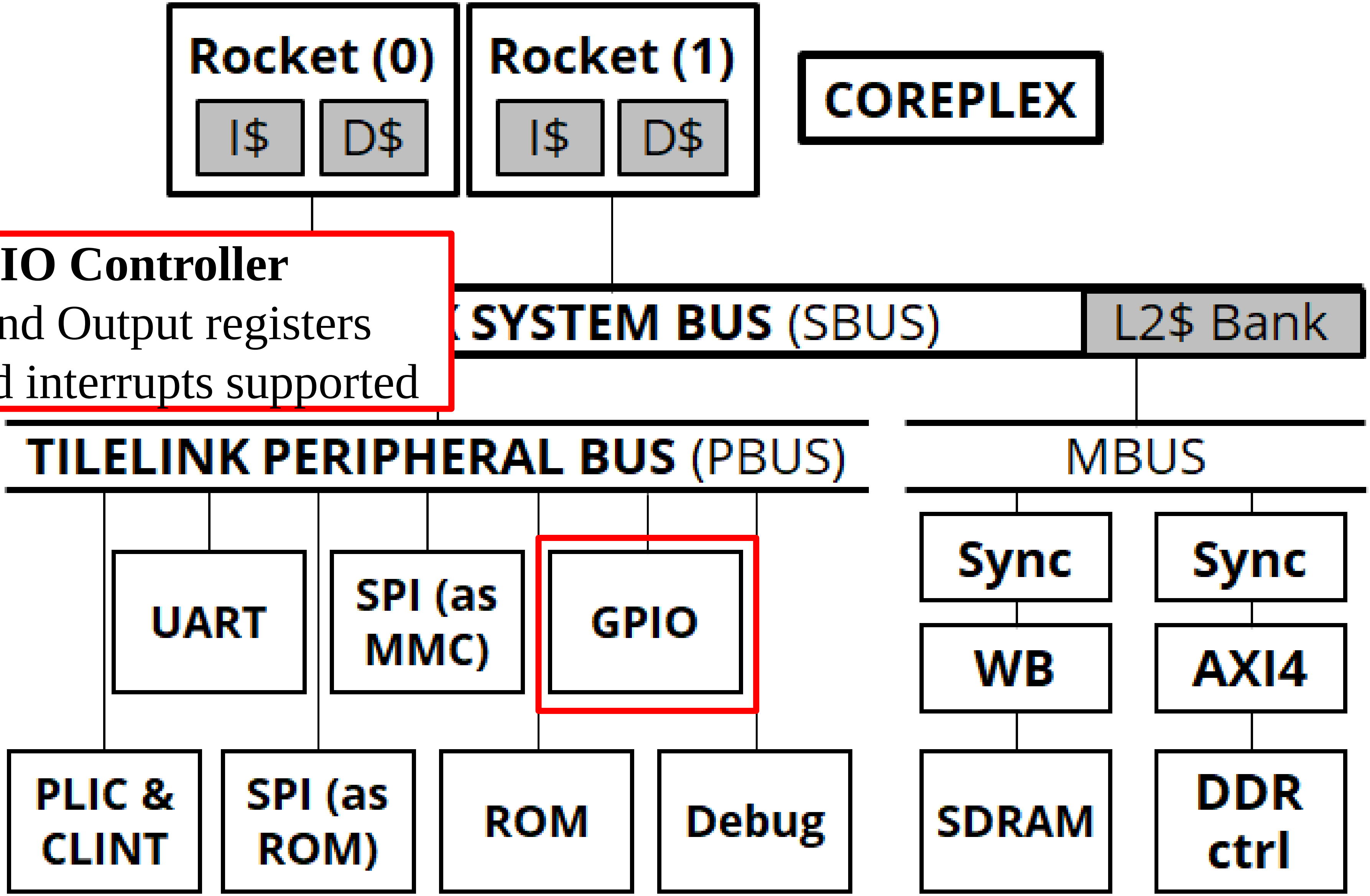
2. System architecture (4/9) SD-card



SPI Controller

- To communicate with SD cards
- Read the partition and execute

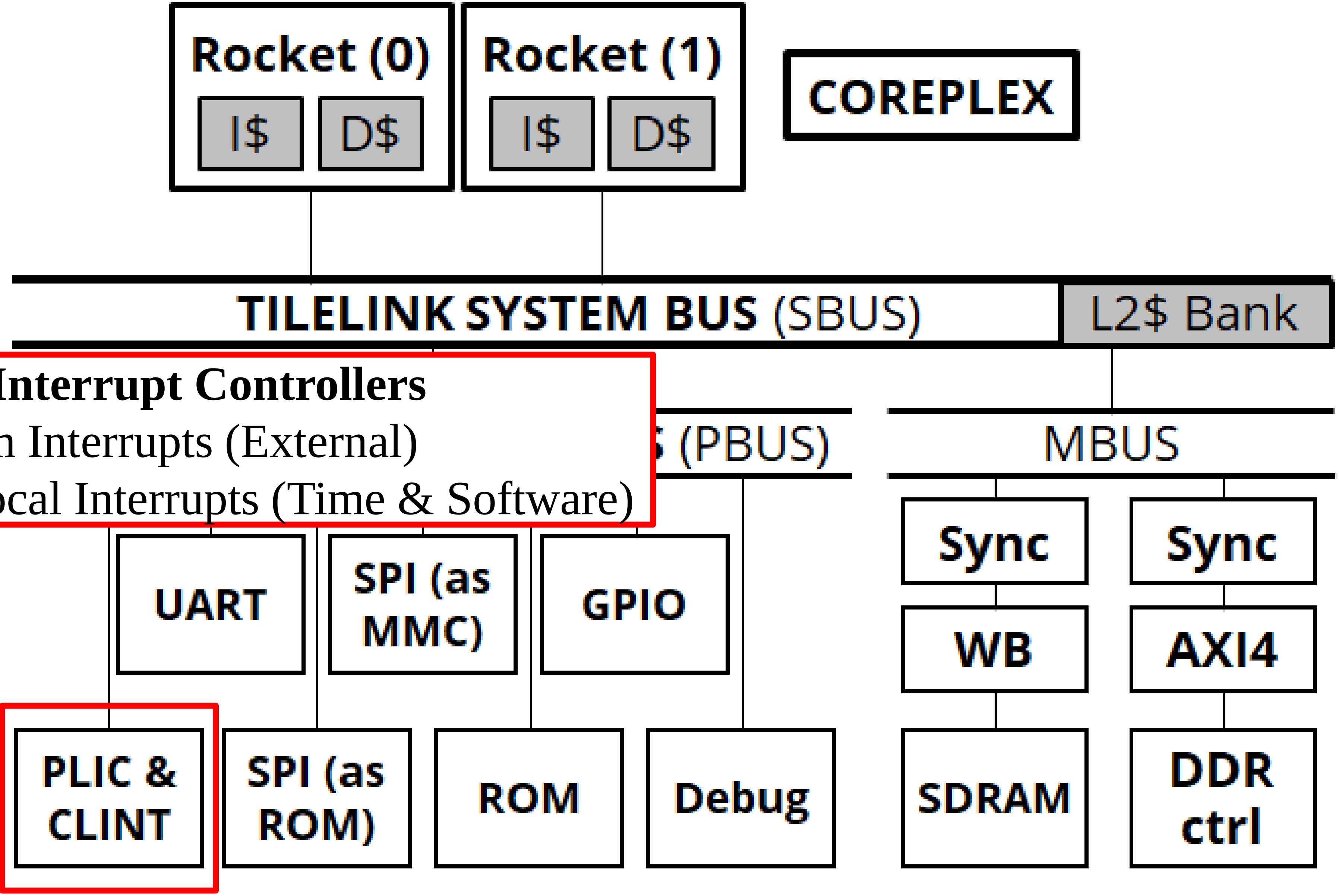
2. System architecture (5/9) GPIO



GPIO Controller

- Input and Output registers
- Masked interrupts supported

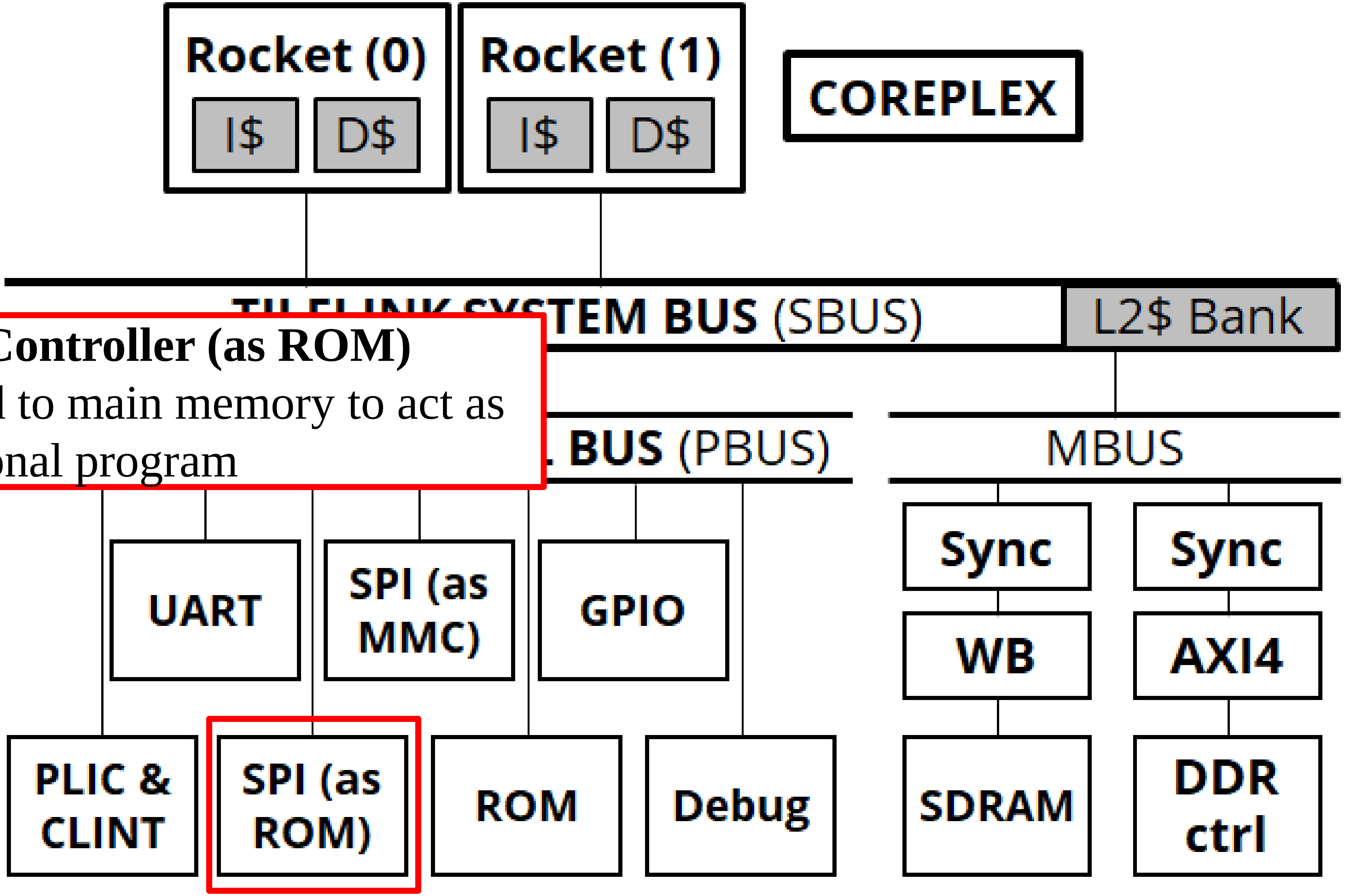
2. System architecture (6/9) Interrupt



Interrupt Controllers

- Platform Interrupts (External)
- Core Local Interrupts (Time & Software)

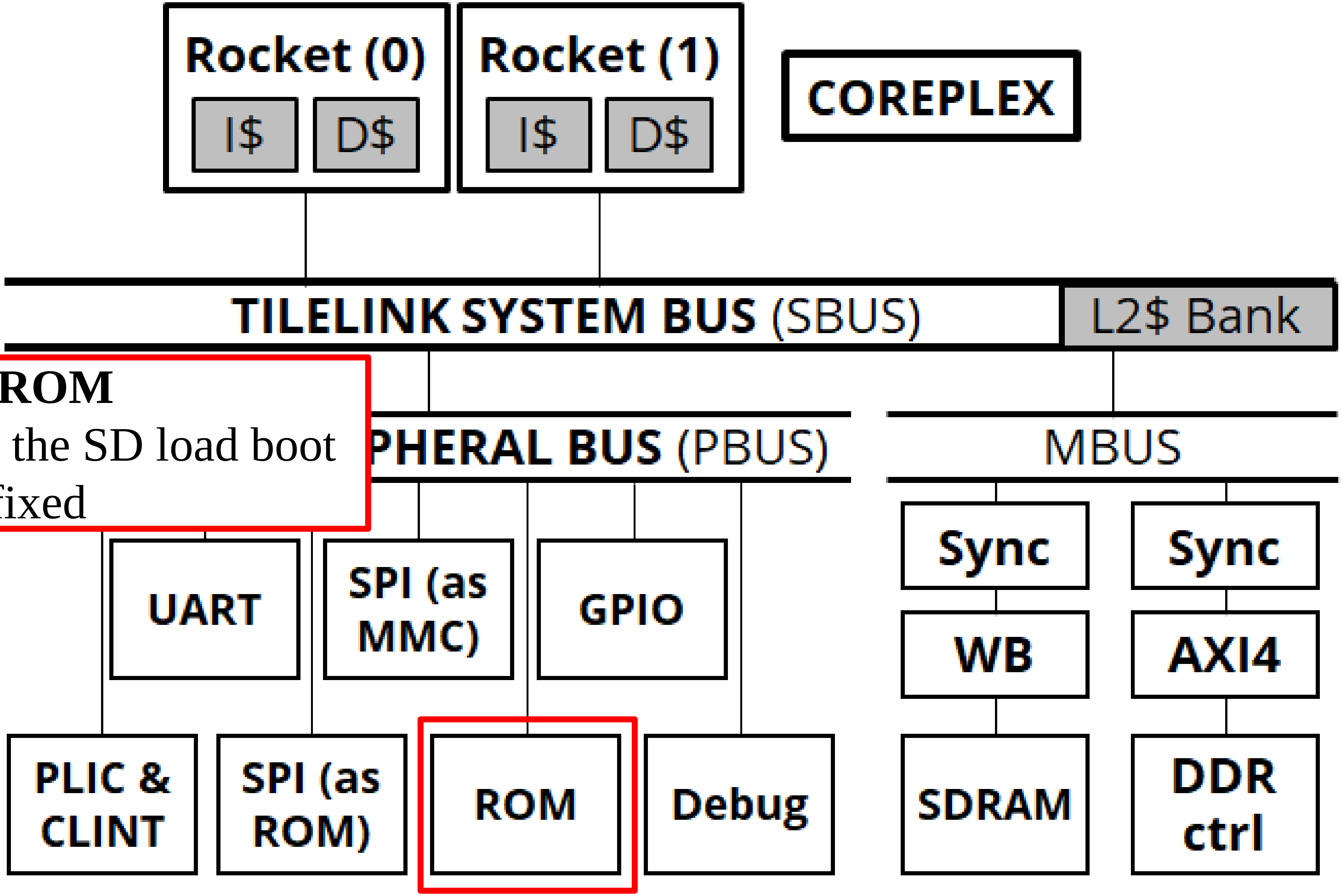
2. System architecture (7/9) SPI



SPI Controller (as ROM)

- Mapped to main memory to act as an optional program

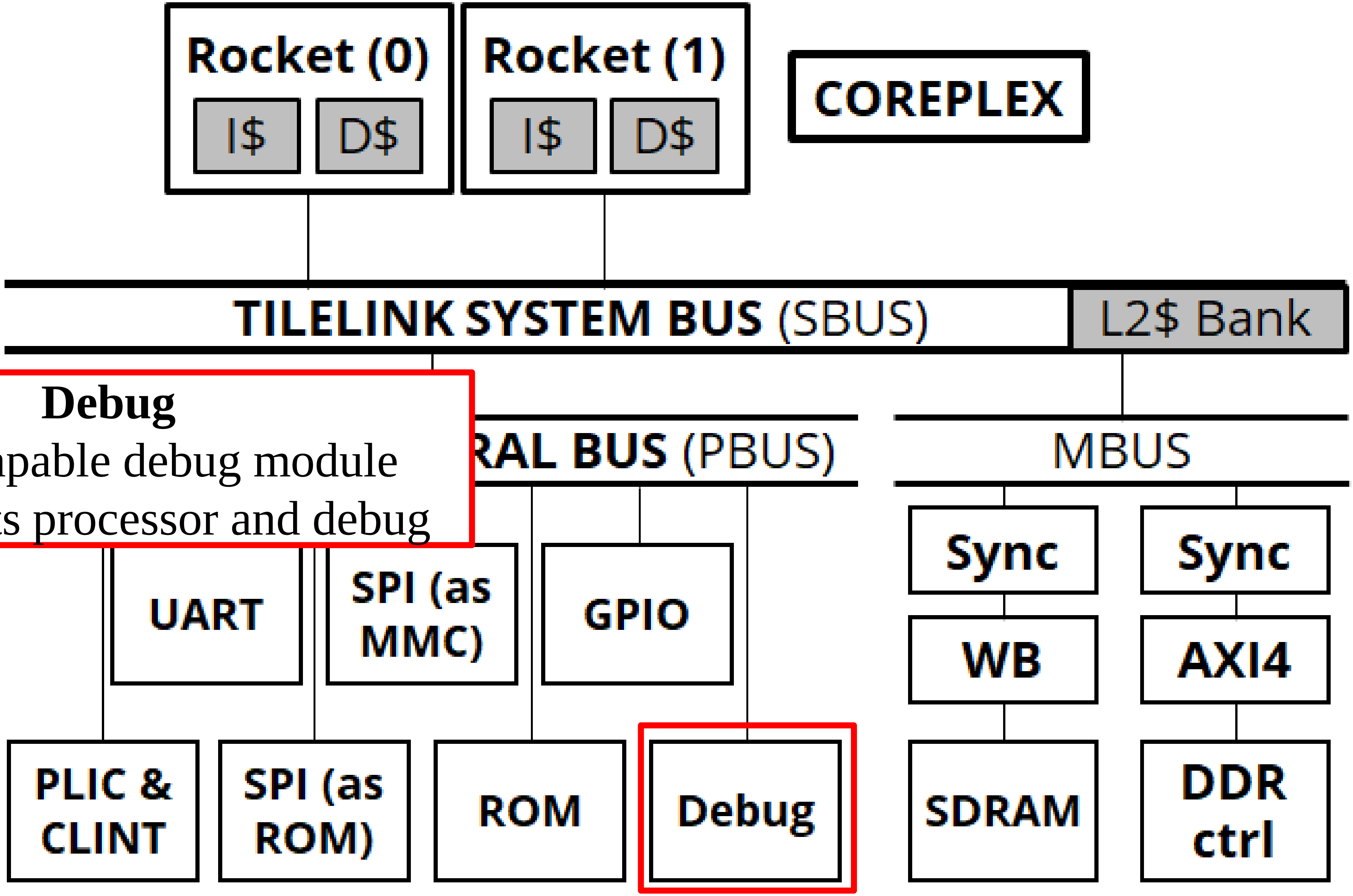
2. System architecture (8/9) Boot ROM



ROM

- Contains the SD load boot
- Always fixed

2. System architecture (9/9) Debug module



Debug

- JTAG-capable debug module
- Interrupts processor and debug

Outline

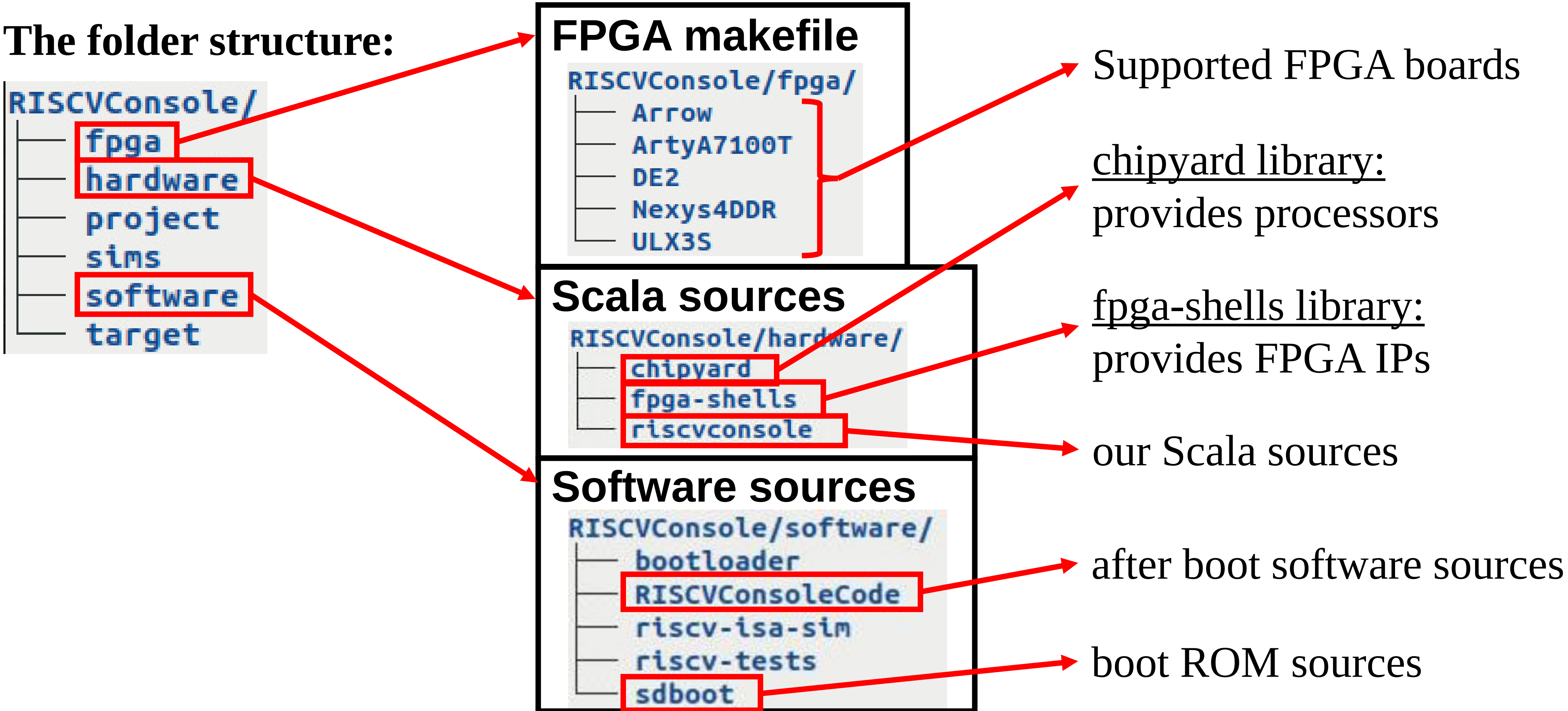
1. Introduction
2. System architecture
3. Git clone and prepare
4. Make the system
5. Program Arty-A7
6. Using IntelliJ IDEA-IC
7. Modifying system by Scala config
8. Practice: system modification

3. Git clone and prepare (1/4) `$ git clone` and folder structure

To clone the project:

```
$ git clone https://github.com/uec-hanken/RISCVConsole.git
$ cd RISCVConsole/
$ ./update.sh
```

The folder structure:



```
RISCVConsole/
├── fpga
├── hardware
├── project
├── sims
├── software
└── target
```

FPGA makefile

```
RISCVConsole/fpga/
├── Arrow
├── ArtyA7100T
├── DE2
├── Nexys4DDR
└── ULX3S
```

Supported FPGA boards

Scala sources

```
RISCVConsole/hardware/
├── chipyard
├── fpga-shells
└── riscvconsole
```

<u>chipyard library:</u>
provides processors

<u>fpga-shells library:</u>
provides FPGA IPs

our Scala sources

Software sources

```
RISCVConsole/software/
├── bootloader
├── RISCVConsoleCode
├── riscv-isa-sim
├── riscv-tests
└── sdboot
```

after boot software sources

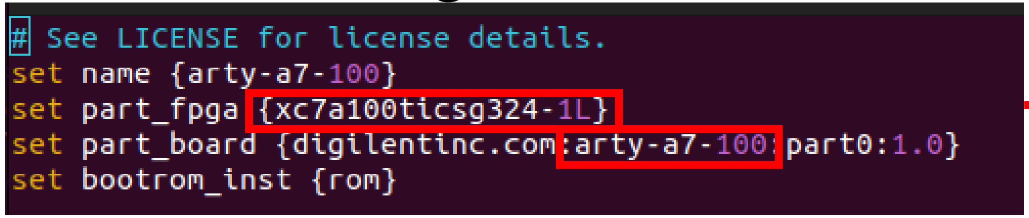
boot ROM sources

3. Git clone and prepare (2/4) Prepare for Arty-A7-35T

If you use the Arty-A7-**100T** version, please skip the next two steps.
If you use the Arty-A7-**35T** version, please do the following:

```
$ vi hardware/fpga-shells/xilinx/arty_a7_100/tcl/board.tcl
```

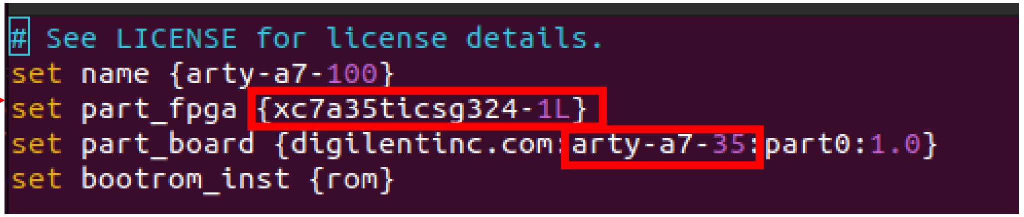
Change from here:



```
# See LICENSE for license details.
set name {arty-a7-100}
set part_fpga {xc7a100ticsg324-1L}
set part_board {digilentinc.com:arty-a7-100:part0:1.0}
set bootrom_inst {rom}
```

(type `i` to write and `esc` to release)

To here:



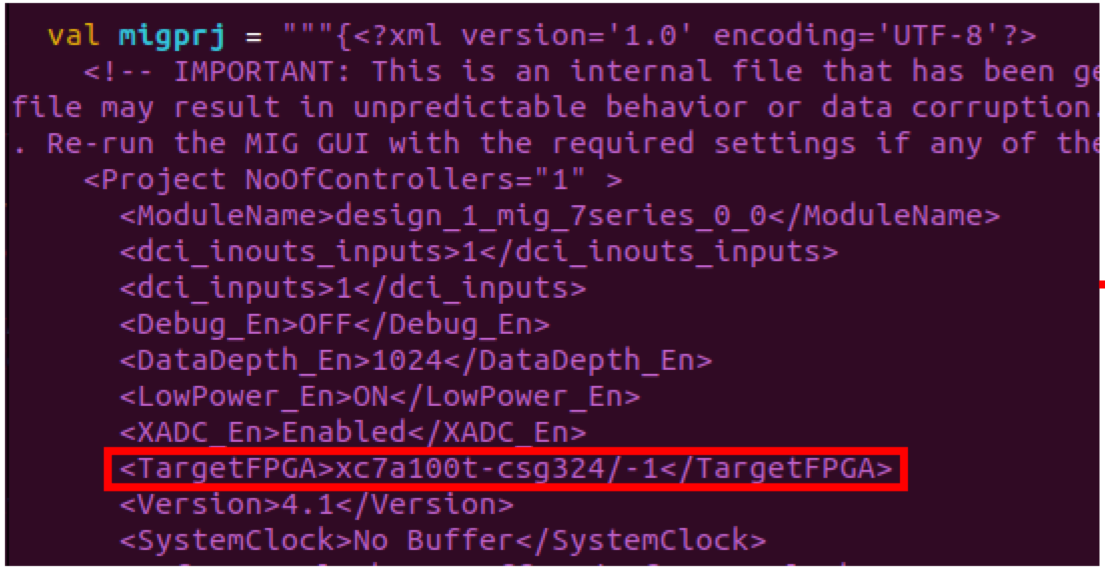
```
# See LICENSE for license details.
set name {arty-a7-100}
set part_fpga {xc7a35ticsg324-1L}
set part_board {digilentinc.com:arty-a7-35:part0:1.0}
set bootrom_inst {rom}
```

(type `:wq` to save and exit)

3. Git clone and prepare (3/4) Prepare for Arty-A7-35T

```
$ vi hardware/fpga-shells/src/main/scala/ip/xilinx/arty100tmig/arty100tmig.scala
```

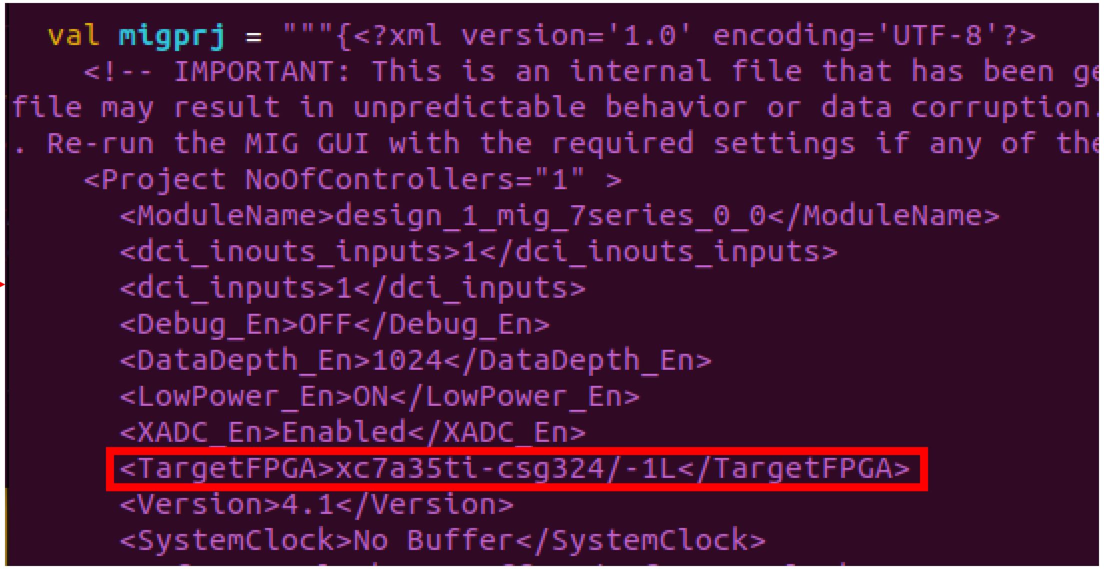
Change from here:



```
  val migprj = """{<?xml version='1.0' encoding='UTF-8'?>
    <!-- IMPORTANT: This is an internal file that has been ge
file may result in unpredictable behavior or data corruption.
. Re-run the MIG GUI with the required settings if any of the
    <Project NoOfControllers="1" >
      <ModuleName>design_1_mig_7series_0_0</ModuleName>
      <dci_inouts_inputs>1</dci_inouts_inputs>
      <dci_inputs>1</dci_inputs>
      <Debug_En>OFF</Debug_En>
      <DataDepth_En>1024</DataDepth_En>
      <LowPower_En>ON</LowPower_En>
      <XADC_En>Enabled</XADC_En>
      <TargetFPGA>xc7a100t-csg324/-1</TargetFPGA>
      <Version>4.1</Version>
      <SystemClock>No Buffer</SystemClock>
```

(type `i` *to write and* `esc` *to release)*

To here:

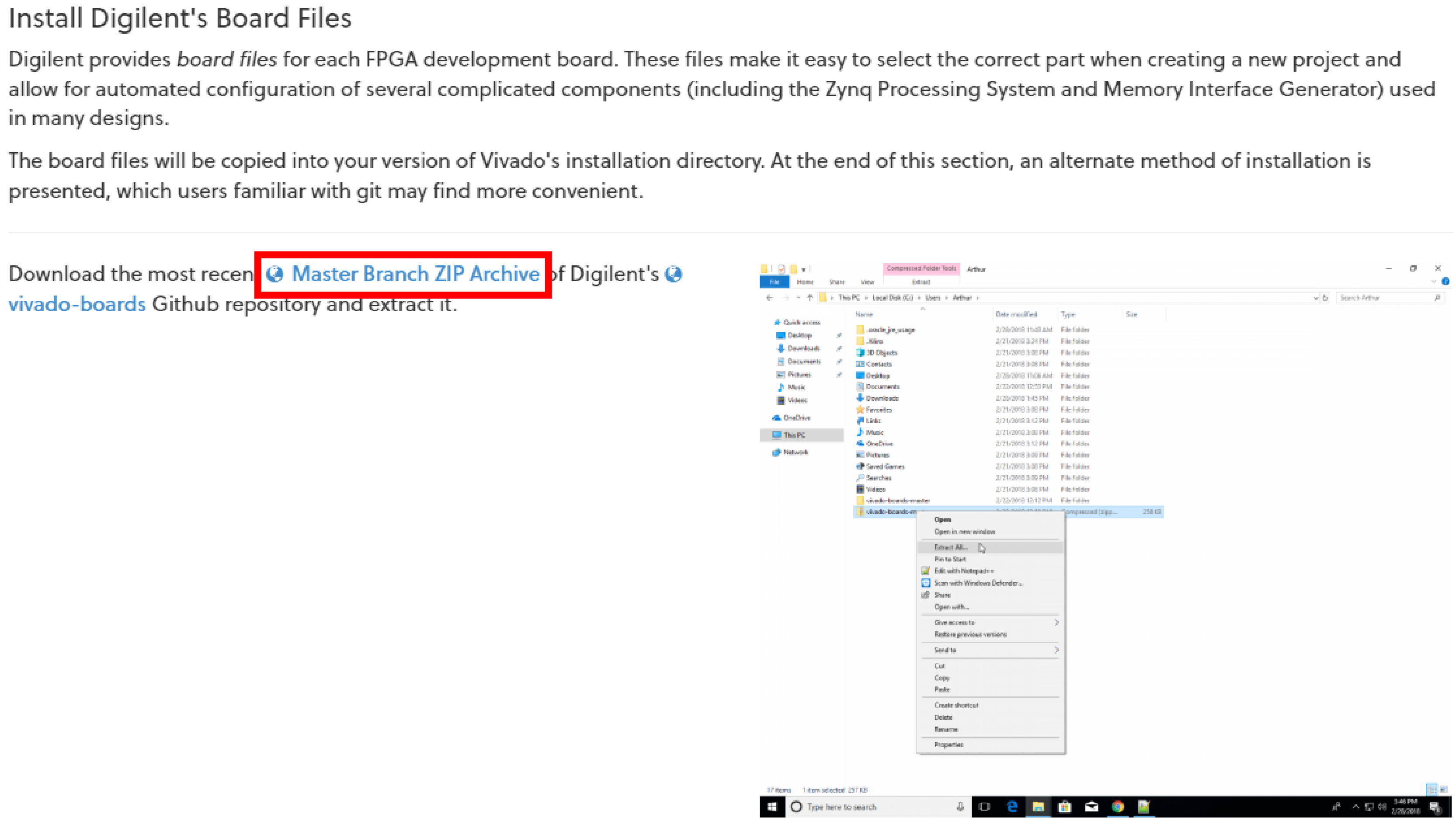


```
  val migprj = """{<?xml version='1.0' encoding='UTF-8'?>
    <!-- IMPORTANT: This is an internal file that has been ge
file may result in unpredictable behavior or data corruption.
. Re-run the MIG GUI with the required settings if any of the
    <Project NoOfControllers="1" >
      <ModuleName>design_1_mig_7series_0_0</ModuleName>
      <dci_inouts_inputs>1</dci_inouts_inputs>
      <dci_inputs>1</dci_inputs>
      <Debug_En>OFF</Debug_En>
      <DataDepth_En>1024</DataDepth_En>
      <LowPower_En>ON</LowPower_En>
      <XADC_En>Enabled</XADC_En>
      <TargetFPGA>xc7a35ti-csg324/-1L</TargetFPGA>
      <Version>4.1</Version>
      <SystemClock>No Buffer</SystemClock>
```

(type `:wq` *to save and exit)*

3. Git clone and prepare (4/4) Prepare Arty-A7 license

Arty-A7 license is free. You can download and install the Arty-A7 license to Vivado.
Download the link from the Digilent website.
Guide to install the license:
https://digilent.com/reference/programmable-logic/guides/installing-vivado-and-vitis



Install Digilent's Board Files

Digilent provides *board files* for each FPGA development board. These files make it easy to select the correct part when creating a new project and allow for automated configuration of several complicated components (including the Zynq Processing System and Memory Interface Generator) used in many designs.

The board files will be copied into your version of Vivado's installation directory. At the end of this section, an alternate method of installation is presented, which users familiar with git may find more convenient.

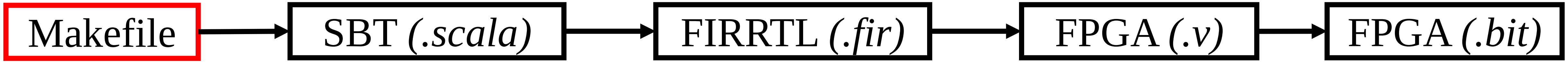
Download the most recent Master Branch ZIP Archive of Digilent's vivado-boards Github repository and extract it.

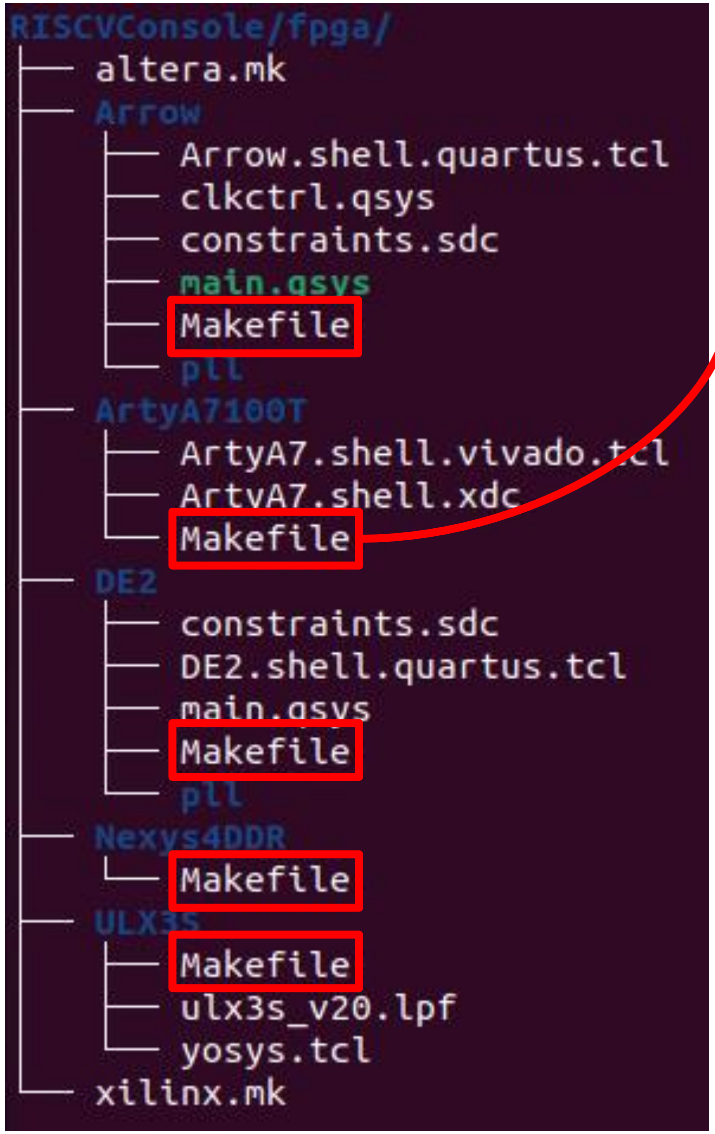
1. Extract the ZIP file
2. Copy all the content in `new/board_files`
3. Paste them to `Xilinx/Vivado/2022.1/data/boards/board_files/`

Outline

1. Introduction
2. System architecture
3. Git clone and prepare
4. **Make the system**
5. Program Arty-A7
6. Using IntelliJ IDEA-IC
7. Modifying system by Scala config
8. Practice: system modification

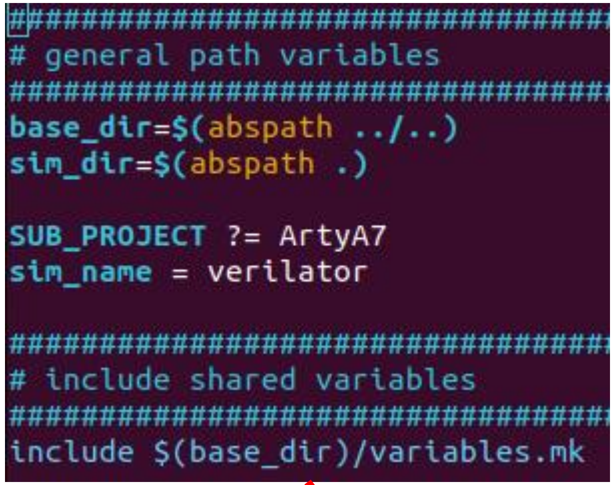
4. Make the system (1/9) Makefile in fpga folder





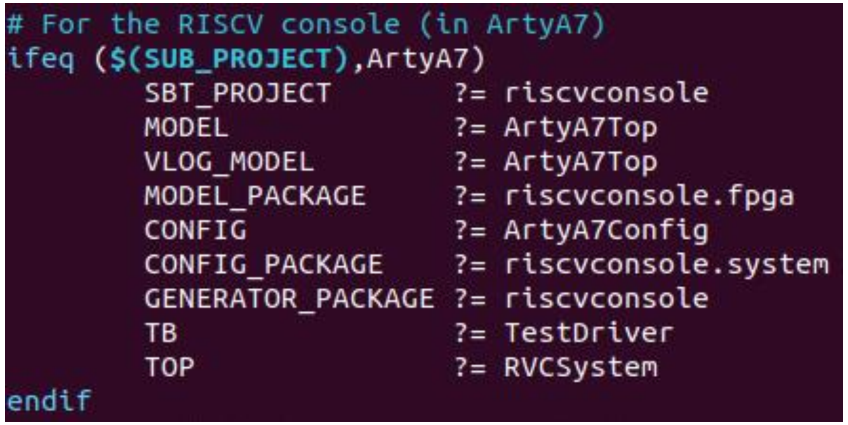
```
RISCVConsole/fpga/
├── altera.mk
├── Arrow
│   ├── Arrow.shell.quartus.tcl
│   ├── clkctrl.qsys
│   ├── constraints.sdc
│   ├── main.qsys
│   ├── Makefile
│   └── pll
├── ArtyA7100T
│   ├── ArtyA7.shell.vivado.tcl
│   ├── ArtyA7.shell.xdc
│   └── Makefile
├── DE2
│   ├── constraints.sdc
│   ├── DE2.shell.quartus.tcl
│   ├── main.qsys
│   ├── Makefile
│   └── pll
├── Nexys4DDR
│   └── Makefile
├── ULX3S
│   ├── Makefile
│   ├── ulx3s_v20.lpf
│   └── yosys.tcl
└── xilinx.mk
```

At `RISCVConsole/fpga/ArtyA7100T/Makefile`:



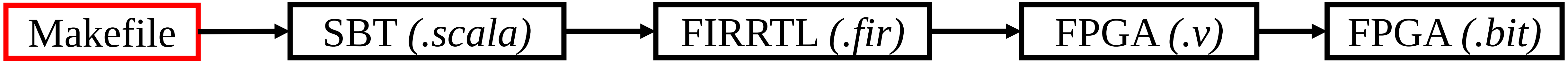
```
#################################
# general path variables
#################################
base_dir=$(abspath ../..)
sim_dir=$(abspath .)

SUB_PROJECT ?= ArtyA7
sim_name = verilator

#################################
# include shared variables
#################################
include $(base_dir)/variables.mk
```

At `RISCVConsole/variables.mk`:



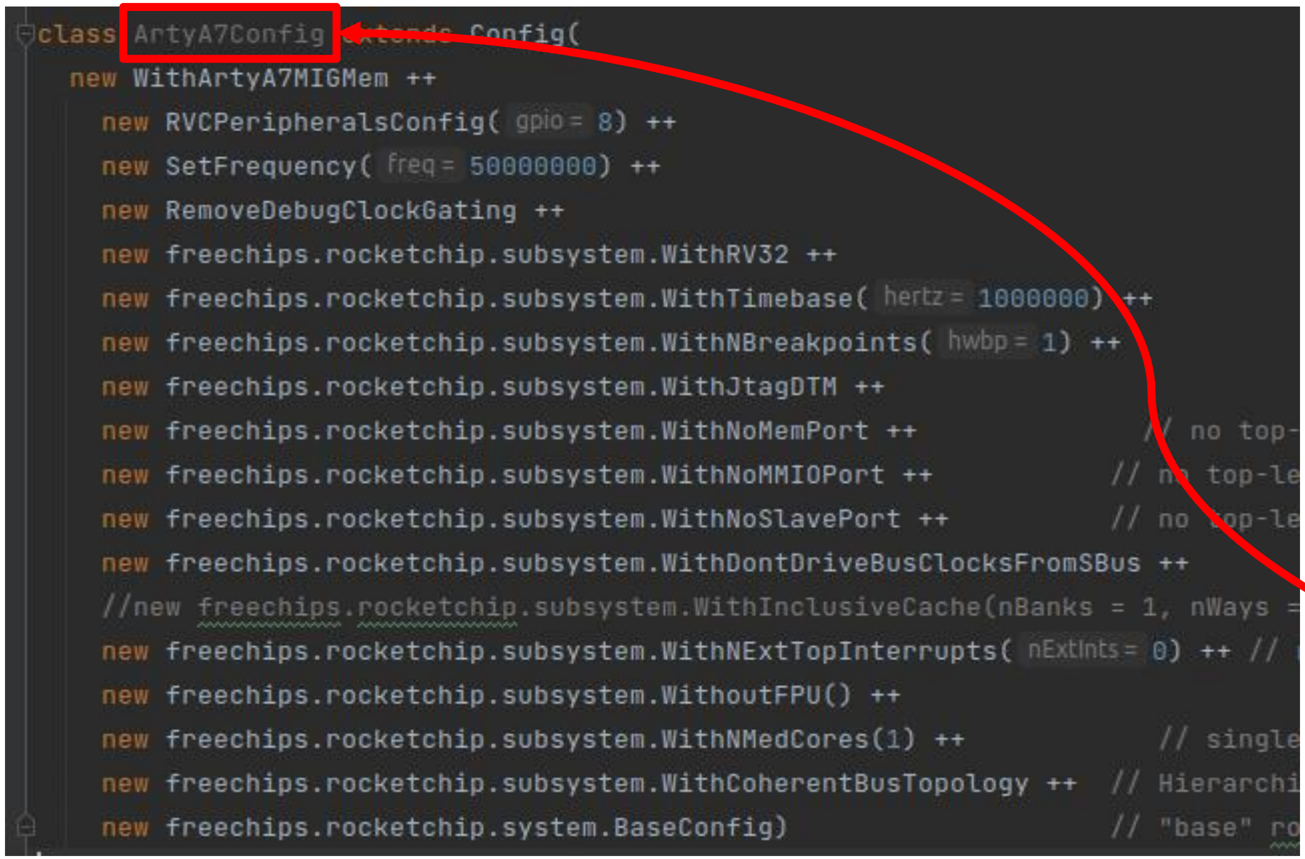
```
# For the RISCV console (in ArtyA7)
ifeq ($(SUB_PROJECT),ArtyA7)
	SBT_PROJECT       ?= riscvconsole
	MODEL             ?= ArtyA7Top
	VLOG_MODEL        ?= ArtyA7Top
	MODEL_PACKAGE     ?= riscvconsole.fpga
	CONFIG            ?= ArtyA7Config
	CONFIG_PACKAGE    ?= riscvconsole.system
	GENERATOR_PACKAGE ?= riscvconsole
	TB                ?= TestDriver
	TOP               ?= RVCSystem
endif
```

4. **Make the system** (2/9) Equivalent in **Scala** config



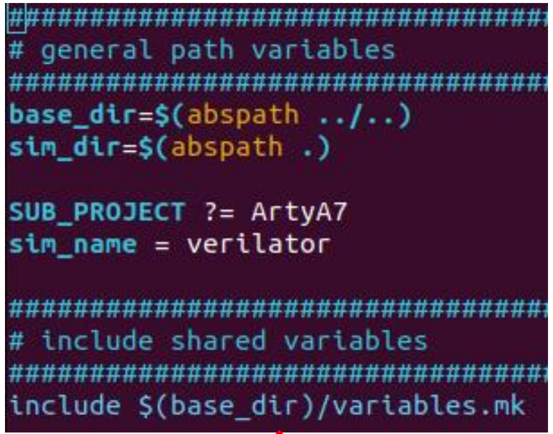
Makefile → SBT (*.scala*) → FIRRTL (*.fir*) → FPGA (*.v*) → FPGA (*.bit*)

At `RISCVConsole/hardware/riscvconsole/src/main/scala/riscvconsole/RVCConfig.scala`:



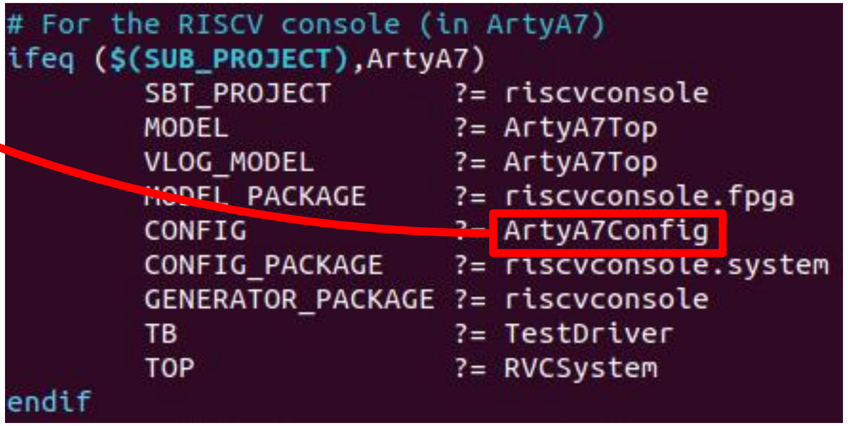
```
class ArtyA7Config extends Config(
  new WithArtyA7MIGMem ++
    new RVCPeripheralsConfig( gpio = 8) ++
    new SetFrequency( freq = 50000000) ++
    new RemoveDebugClockGating ++
    new freechips.rocketchip.subsystem.WithRV32 ++
    new freechips.rocketchip.subsystem.WithTimebase( hertz = 1000000) ++
    new freechips.rocketchip.subsystem.WithNBreakpoints( hwbp = 1) ++
    new freechips.rocketchip.subsystem.WithJtagDTM ++
    new freechips.rocketchip.subsystem.WithNoMemPort ++                // no top-
    new freechips.rocketchip.subsystem.WithNoMMIOPort ++              // no top-le
    new freechips.rocketchip.subsystem.WithNoSlavePort ++             // no top-le
    new freechips.rocketchip.subsystem.WithDontDriveBusClocksFromSBus ++
    //new freechips.rocketchip.subsystem.WithInclusiveCache(nBanks = 1, nWays =
    new freechips.rocketchip.subsystem.WithNExtTopInterrupts( nExtInts = 0) ++ //
    new freechips.rocketchip.subsystem.WithoutFPU() ++
    new freechips.rocketchip.subsystem.WithNMedCores(1) ++              // single
    new freechips.rocketchip.subsystem.WithCoherentBusTopology ++  // Hierarchi
    new freechips.rocketchip.system.BaseConfig)                     // "base" ro
```

At `RISCVConsole/fpga/ArtyA7100T/Makefile`:



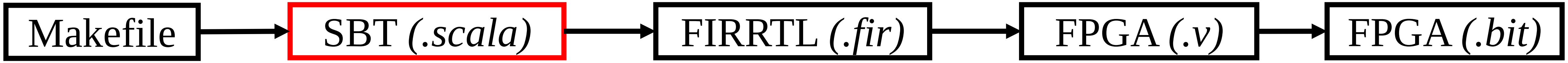
```
#####################################
# general path variables
#####################################
base_dir=$(abspath ../..)
sim_dir=$(abspath .)

SUB_PROJECT ?= ArtyA7
sim_name = verilator

#####################################
# include shared variables
#####################################
include $(base_dir)/variables.mk
```

At `RISCVConsole/variables.mk`:



```
# For the RISCV console (in ArtyA7)
ifeq ($(SUB_PROJECT),ArtyA7)
	SBT_PROJECT       ?= riscvconsole
	MODEL             ?= ArtyA7Top
	VLOG_MODEL        ?= ArtyA7Top
	MODEL_PACKAGE     ?= riscvconsole.fpga
	CONFIG            ?= ArtyA7Config
	CONFIG_PACKAGE    ?= riscvconsole.system
	GENERATOR_PACKAGE ?= riscvconsole
	TB                ?= TestDriver
	TOP               ?= RVCSystem
endif
```

4. Make the system (3/9) Scala structure



Makefile → SBT *(.scala)* → FIRRTL *(.fir)* → FPGA *(.v)* → FPGA *(.bit)*

FPGA Shell folder

GPIO Pins
UART Pins
SPI Pins
I2C Pins
DDR Ports
SDRAM Ports
CODEC Ports
JTAG Pins
Other Ports

FPGA folder

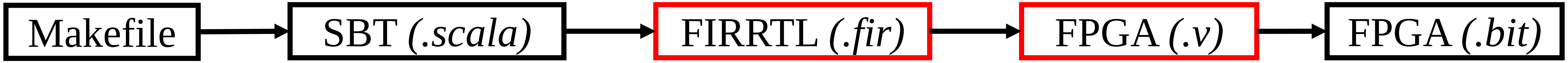
RVC. System

- General Purpose IO
- UART
- SPI Flash
- I2C
- SDRAM/DDR
- TL Serial (For simulations)
- FFT/CODEC
- Any additional peripherals

RVC. Subsystem

- TileLink Buses
- Debug Module
- Boot ROM
- Core Local Interrupts
- Platform Level Interrupt Controller
- Coreplex
  - Rocket
  - BOOM

4. Make the system (4/9) `$ make default`



Now, to make the system, from the RISCVConsole, go to the Arty build folder:

```
$ cd fpga/ArtyA7100T/
```

Export the RISC-V toolchain to the **PATH**:

```
$ export RISCV=/opt/riscv
$ export PATH=$RISCV/bin/:$PATH
```

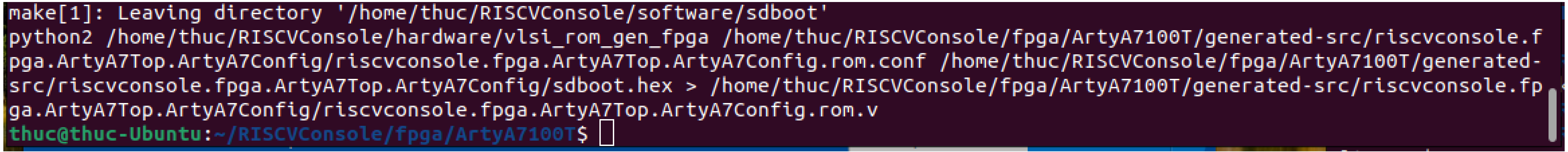
Export Vivado to the **PATH**:

```
$ export PATH=/opt/Xilinx/Vivado/2022.1/bin/:$PATH
```

For the compilation:

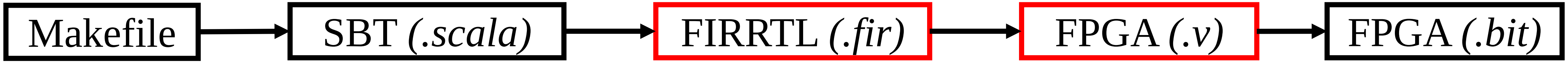
```
$ make default
```

This will compile Scala to Verilog *(also compile the boot ROM C/C++ sources)*

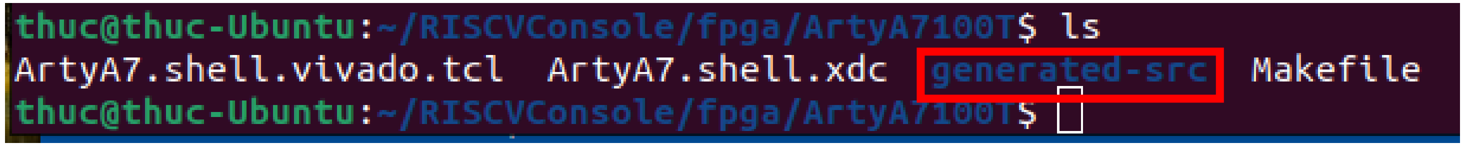


```
make[1]: Leaving directory '/home/thuc/RISCVConsole/software/sdboot'
python2 /home/thuc/RISCVConsole/hardware/vlsi_rom_gen_fpga /home/thuc/RISCVConsole/fpga/ArtyA7100T/generated-src/riscvconsole.f
pga.ArtyA7Top.ArtyA7Config/riscvconsole.fpga.ArtyA7Top.ArtyA7Config.rom.conf /home/thuc/RISCVConsole/fpga/ArtyA7100T/generated-
src/riscvconsole.fpga.ArtyA7Top.ArtyA7Config/sdboot.hex > /home/thuc/RISCVConsole/fpga/ArtyA7100T/generated-src/riscvconsole.fp
ga.ArtyA7Top.ArtyA7Config/riscvconsole.fpga.ArtyA7Top.ArtyA7Config.rom.v
thuc@thuc-Ubuntu:~/RISCVConsole/fpga/ArtyA7100T$
```

4. Make the system (5/9) `$ make default`



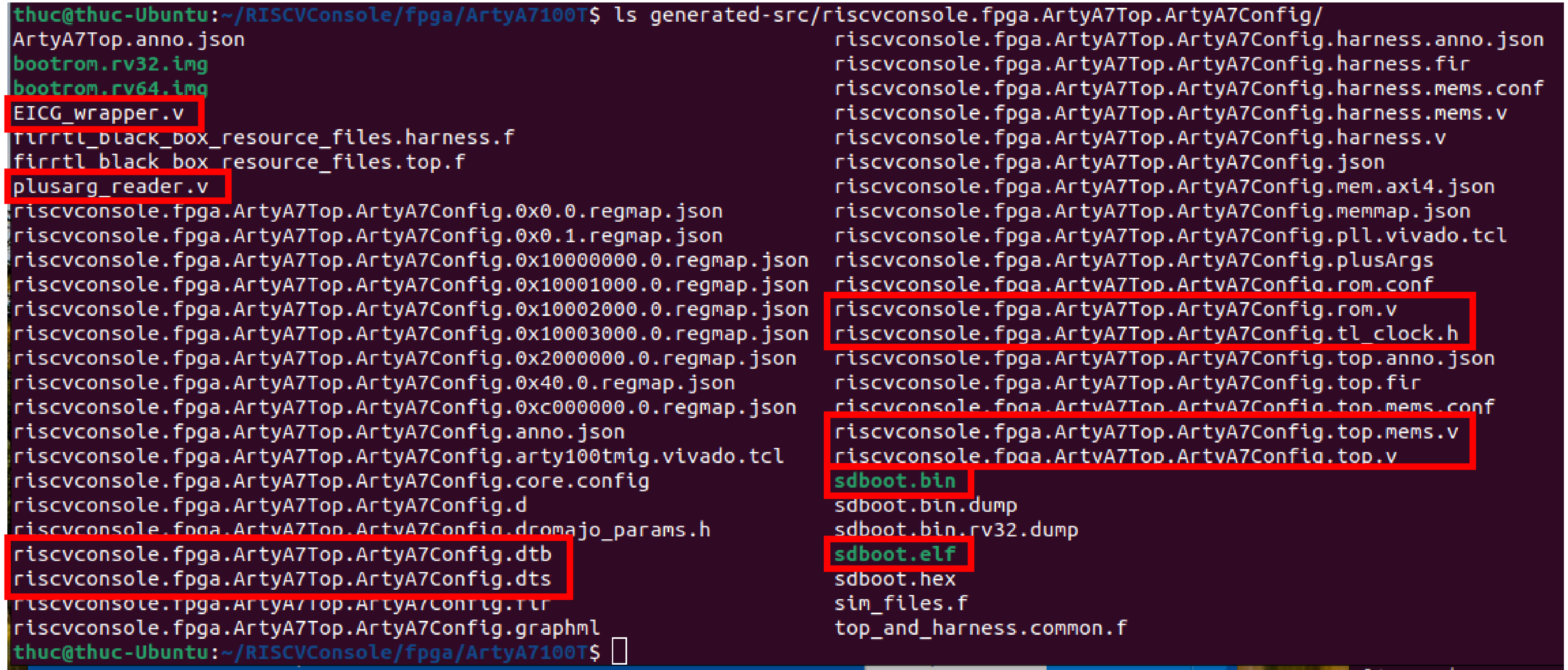
After `$ make default`, the **generated-src** folder is created:



```
thuc@thuc-Ubuntu:~/RISCVConsole/fpga/ArtyA7100T$ ls
ArtyA7.shell.vivado.tcl  ArtyA7.shell.xdc  generated-src  Makefile
thuc@thuc-Ubuntu:~/RISCVConsole/fpga/ArtyA7100T$
```

Inside the **generated-src** folder, there are many files:

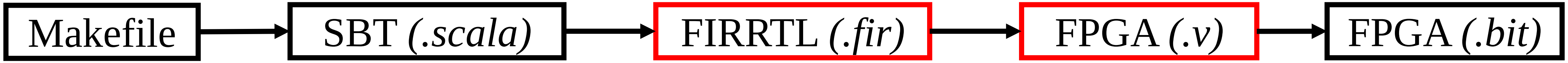
Verilog files, FIRRTL files, temporary Java files, boot ROM files, device tree, etc.

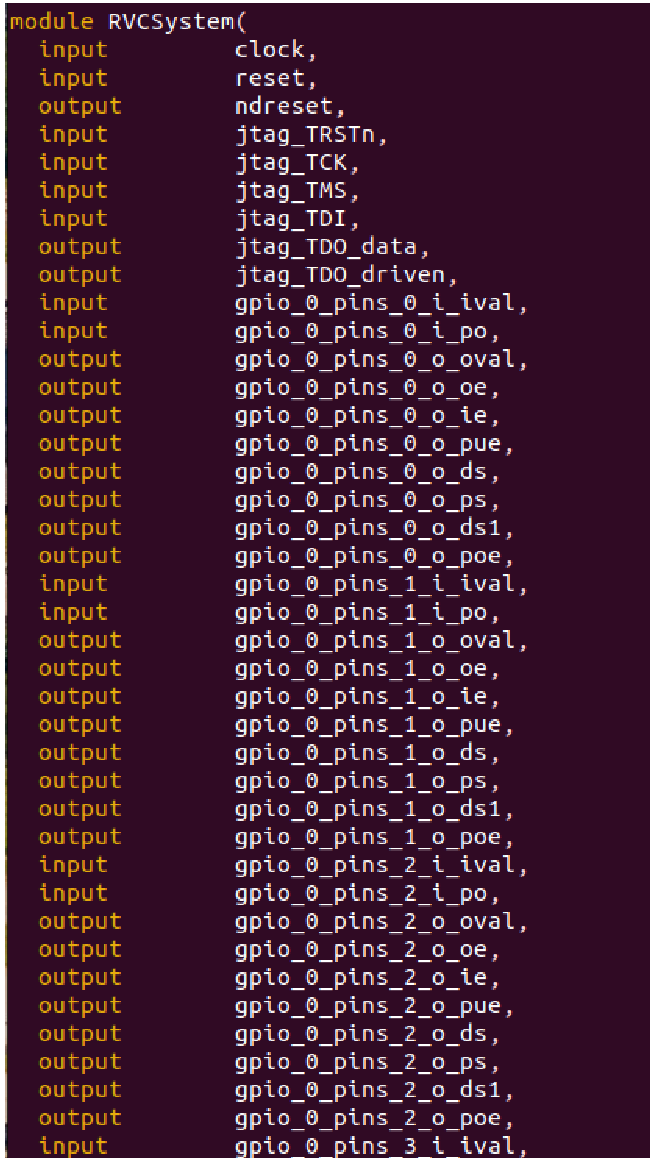


```
thuc@thuc-Ubuntu:~/RISCVConsole/fpga/ArtyA7100T$ ls generated-src/riscvconsole.fpga.ArtyA7Top.ArtyA7Config/
ArtyA7Top.anno.json                                                 riscvconsole.fpga.ArtyA7Top.ArtyA7Config.harness.anno.json
bootrom.rv32.img                                                    riscvconsole.fpga.ArtyA7Top.ArtyA7Config.harness.fir
bootrom.rv64.img                                                    riscvconsole.fpga.ArtyA7Top.ArtyA7Config.harness.mems.conf
EICG_wrapper.v                                                      riscvconsole.fpga.ArtyA7Top.ArtyA7Config.harness.mems.v
firrtl_black_box_resource_files.harness.f                           riscvconsole.fpga.ArtyA7Top.ArtyA7Config.harness.v
firrtl_black_box_resource_files.top.f                               riscvconsole.fpga.ArtyA7Top.ArtyA7Config.json
plusarg_reader.v                                                    riscvconsole.fpga.ArtyA7Top.ArtyA7Config.mem.axi4.json
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.0x0.0.regmap.json          riscvconsole.fpga.ArtyA7Top.ArtyA7Config.memmap.json
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.0x0.1.regmap.json          riscvconsole.fpga.ArtyA7Top.ArtyA7Config.pll.vivado.tcl
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.0x10000000.0.regmap.json   riscvconsole.fpga.ArtyA7Top.ArtyA7Config.plusArgs
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.0x10001000.0.regmap.json   riscvconsole.fpga.ArtyA7Top.ArtyA7Config.rom.conf
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.0x10002000.0.regmap.json   riscvconsole.fpga.ArtyA7Top.ArtyA7Config.rom.v
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.0x10003000.0.regmap.json   riscvconsole.fpga.ArtyA7Top.ArtyA7Config.tl_clock.h
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.0x2000000.0.regmap.json    riscvconsole.fpga.ArtyA7Top.ArtyA7Config.top.anno.json
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.0x40.0.regmap.json         riscvconsole.fpga.ArtyA7Top.ArtyA7Config.top.fir
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.0xc000000.0.regmap.json    riscvconsole.fpga.ArtyA7Top.ArtyA7Config.top.mems.conf
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.anno.json                  riscvconsole.fpga.ArtyA7Top.ArtyA7Config.top.mems.v
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.arty100tmig.vivado.tcl     riscvconsole.fpga.ArtyA7Top.ArtyA7Config.top.v
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.core.config                sdboot.bin
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.d                          sdboot.bin.dump
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.dromajo_params.h           sdboot.bin.rv32.dump
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.dtb                        sdboot.elf
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.dts                        sdboot.hex
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.fir                        sim_files.f
riscvconsole.fpga.ArtyA7Top.ArtyA7Config.graphml                    top_and_harness.common.f
thuc@thuc-Ubuntu:~/RISCVConsole/fpga/ArtyA7100T$
```

Some important files

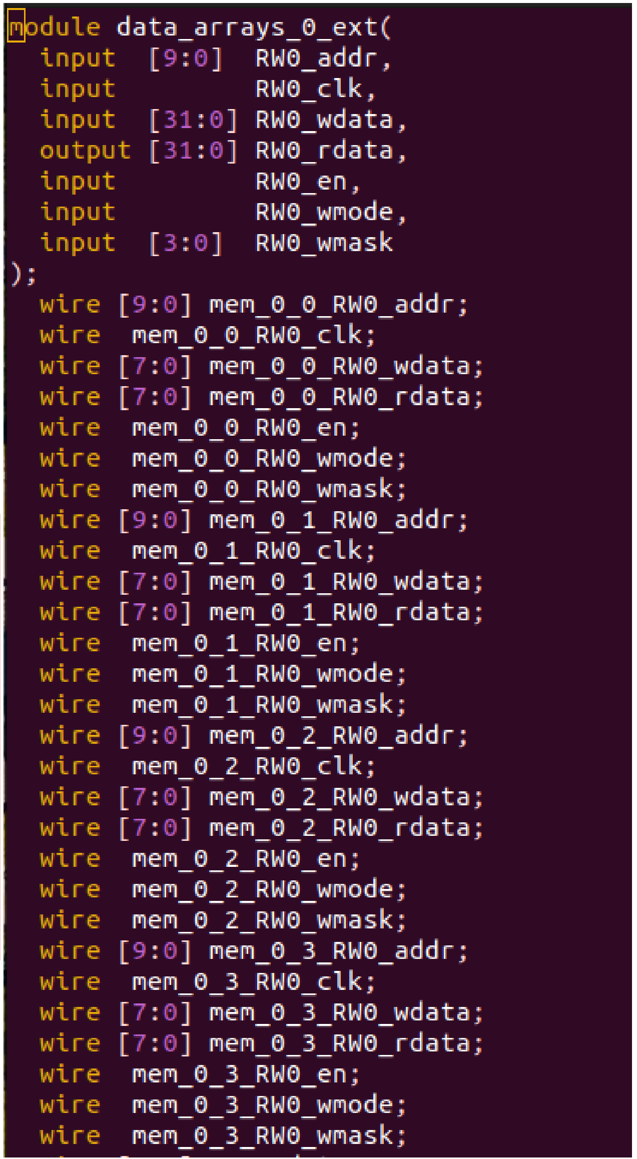
4. Make the system (6/9) Verilog files


Makefile → SBT (*.scala*) → FIRRTL (*.fir*) → FPGA (*.v*) → FPGA (*.bit*)

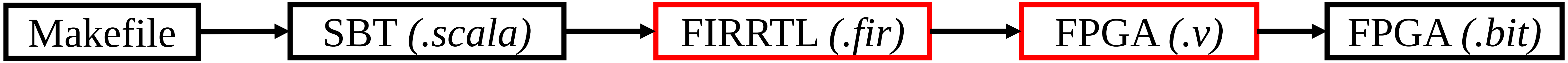

```
module RVCSystem(
  input          clock,
  input          reset,
  output         ndreset,
  input          jtag_TRSTn,
  input          jtag_TCK,
  input          jtag_TMS,
  input          jtag_TDI,
  output         jtag_TDO_data,
  output         jtag_TDO_driven,
  input          gpio_0_pins_0_i_ival,
  input          gpio_0_pins_0_i_po,
  output         gpio_0_pins_0_o_oval,
  output         gpio_0_pins_0_o_oe,
  output         gpio_0_pins_0_o_ie,
  output         gpio_0_pins_0_o_pue,
  output         gpio_0_pins_0_o_ds,
  output         gpio_0_pins_0_o_ps,
  output         gpio_0_pins_0_o_ds1,
  output         gpio_0_pins_0_o_poe,
  input          gpio_0_pins_1_i_ival,
  input          gpio_0_pins_1_i_po,
  output         gpio_0_pins_1_o_oval,
  output         gpio_0_pins_1_o_oe,
  output         gpio_0_pins_1_o_ie,
  output         gpio_0_pins_1_o_pue,
  output         gpio_0_pins_1_o_ds,
  output         gpio_0_pins_1_o_ps,
  output         gpio_0_pins_1_o_ds1,
  output         gpio_0_pins_1_o_poe,
  input          gpio_0_pins_2_i_ival,
  input          gpio_0_pins_2_i_po,
  output         gpio_0_pins_2_o_oval,
  output         gpio_0_pins_2_o_oe,
  output         gpio_0_pins_2_o_ie,
  output         gpio_0_pins_2_o_pue,
  output         gpio_0_pins_2_o_ds,
  output         gpio_0_pins_2_o_ps,
  output         gpio_0_pins_2_o_ds1,
  output         gpio_0_pins_2_o_poe,
  input          gpio_0_pins_3_i_ival,
```

Top file:
`riscvconsole.fpga.ArtyA7Top.ArtyA7Config.top.v`

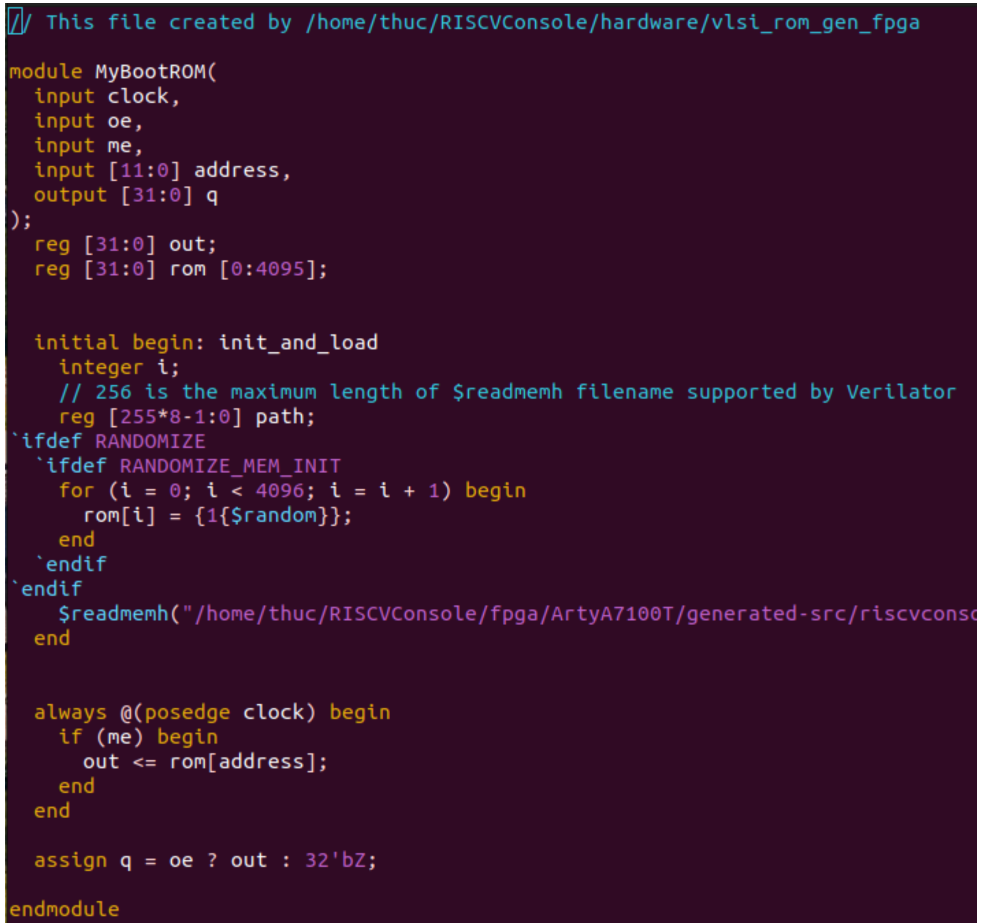
File that contains all the memories used in the system:
`riscvconsole.fpga.ArtyA7Top.ArtyA7Config.top.mems.v`


```
module data_arrays_0_ext(
  input  [9:0]  RW0_addr,
  input         RW0_clk,
  input  [31:0] RW0_wdata,
  output [31:0] RW0_rdata,
  input         RW0_en,
  input         RW0_wmode,
  input  [3:0]  RW0_wmask
);
  wire [9:0] mem_0_0_RW0_addr;
  wire  mem_0_0_RW0_clk;
  wire [7:0] mem_0_0_RW0_wdata;
  wire [7:0] mem_0_0_RW0_rdata;
  wire  mem_0_0_RW0_en;
  wire  mem_0_0_RW0_wmode;
  wire  mem_0_0_RW0_wmask;
  wire [9:0] mem_0_1_RW0_addr;
  wire  mem_0_1_RW0_clk;
  wire [7:0] mem_0_1_RW0_wdata;
  wire [7:0] mem_0_1_RW0_rdata;
  wire  mem_0_1_RW0_en;
  wire  mem_0_1_RW0_wmode;
  wire  mem_0_1_RW0_wmask;
  wire [9:0] mem_0_2_RW0_addr;
  wire  mem_0_2_RW0_clk;
  wire [7:0] mem_0_2_RW0_wdata;
  wire [7:0] mem_0_2_RW0_rdata;
  wire  mem_0_2_RW0_en;
  wire  mem_0_2_RW0_wmode;
  wire  mem_0_2_RW0_wmask;
  wire [9:0] mem_0_3_RW0_addr;
  wire  mem_0_3_RW0_clk;
  wire [7:0] mem_0_3_RW0_wdata;
  wire [7:0] mem_0_3_RW0_rdata;
  wire  mem_0_3_RW0_en;
  wire  mem_0_3_RW0_wmode;
  wire  mem_0_3_RW0_wmask;
```

4. Make the system (7/9) Verilog files



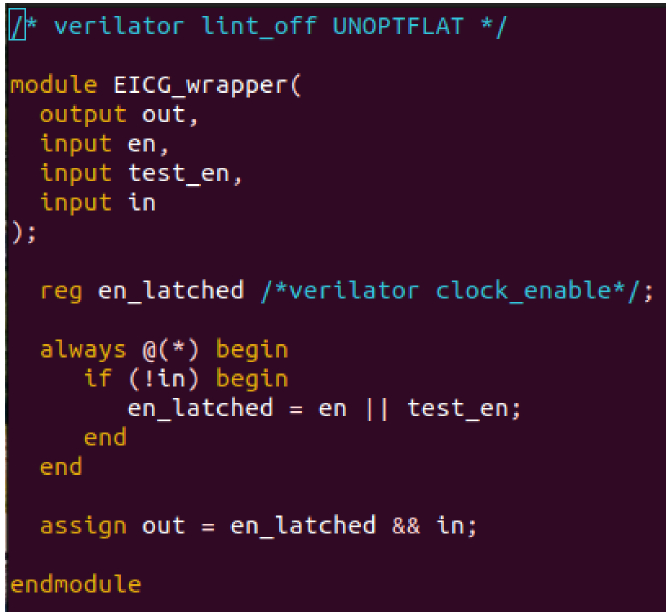
Boot ROM file:
`riscvconsole.fpga.ArtyA7Top.ArtyA7Config.rom.v`



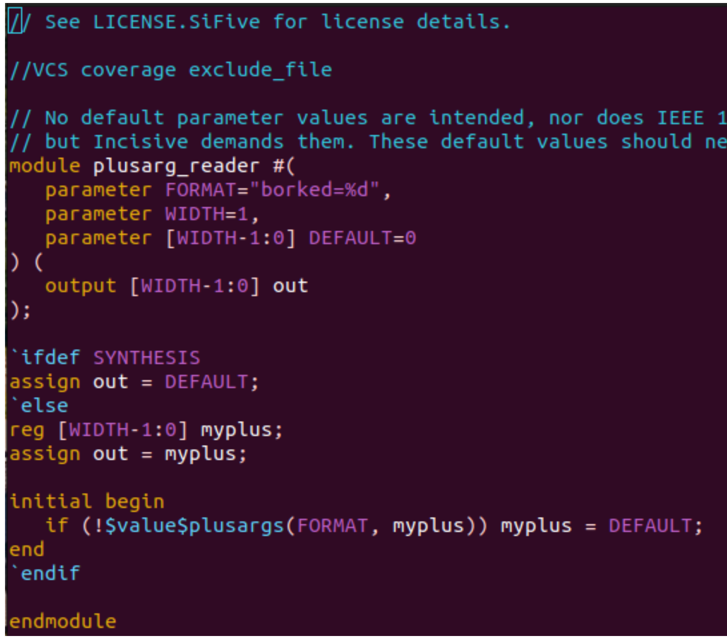
```
// This file created by /home/thuc/RISCVConsole/hardware/vlsi_rom_gen_fpga

module MyBootROM(
  input clock,
  input oe,
  input me,
  input [11:0] address,
  output [31:0] q
);
  reg [31:0] out;
  reg [31:0] rom [0:4095];


  initial begin: init_and_load
    integer i;
    // 256 is the maximum length of $readmemh filename supported by Verilator
    reg [255*8-1:0] path;
`ifdef RANDOMIZE
  `ifdef RANDOMIZE_MEM_INIT
    for (i = 0; i < 4096; i = i + 1) begin
      rom[i] = {1{$random}};
    end
  `endif
`endif
    $readmemh("/home/thuc/RISCVConsole/fpga/ArtyA7100T/generated-src/riscvcons
  end


  always @(posedge clock) begin
    if (me) begin
      out <= rom[address];
    end
  end

  assign q = oe ? out : 32'bZ;

endmodule
```

Other Verilog files:

- `EICG_wrapper.v`
- `plusarg_reader.v`

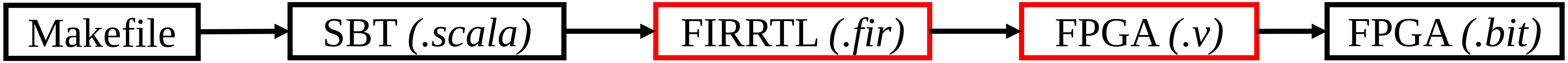


```
/* verilator lint_off UNOPTFLAT */

module EICG_wrapper(
  output out,
  input en,
  input test_en,
  input in
);

  reg en_latched /*verilator clock_enable*/;

  always @(*) begin
     if (!in) begin
        en_latched = en || test_en;
     end
  end

  assign out = en_latched && in;

endmodule
```



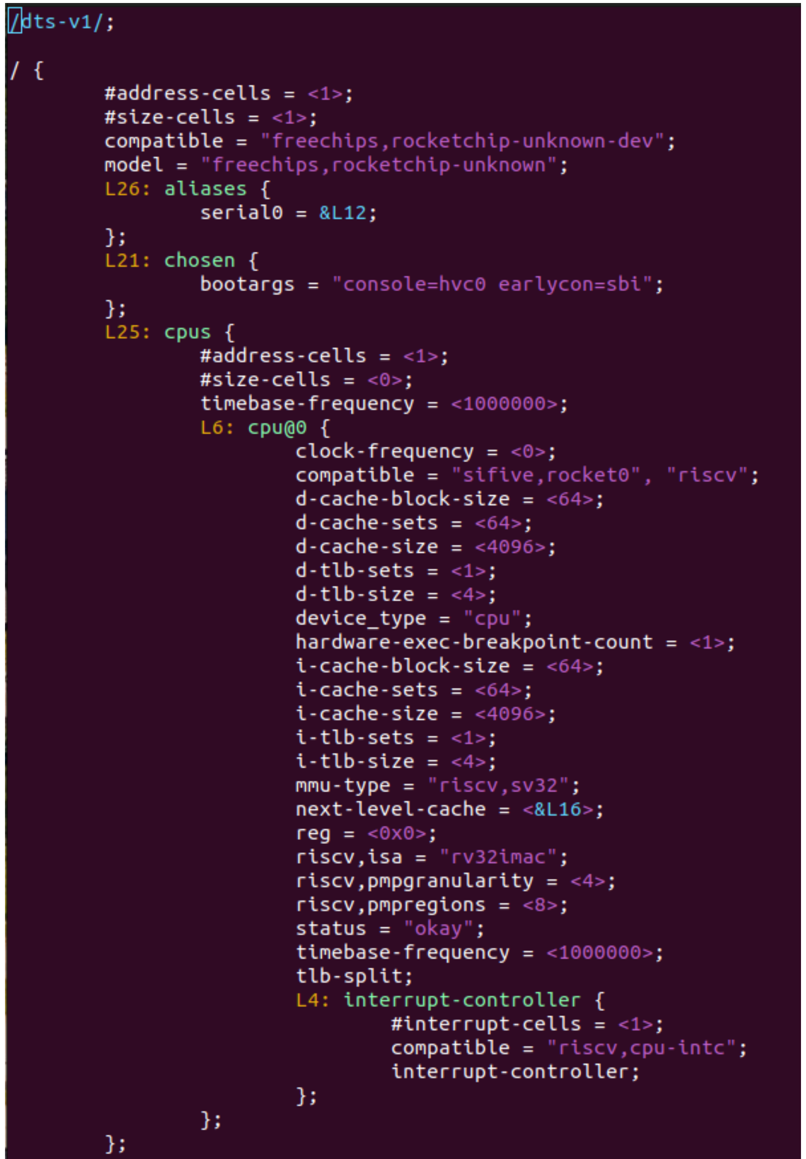
```
// See LICENSE.SiFive for license details.

//VCS coverage exclude_file

// No default parameter values are intended, nor does IEEE 1
// but Incisive demands them. These default values should ne
module plusarg_reader #(
   parameter FORMAT="borked=%d",
   parameter WIDTH=1,
   parameter [WIDTH-1:0] DEFAULT=0
) (
   output [WIDTH-1:0] out
);

`ifdef SYNTHESIS
assign out = DEFAULT;
`else
reg [WIDTH-1:0] myplus;
assign out = myplus;

initial begin
   if (!$value$plusargs(FORMAT, myplus)) myplus = DEFAULT;
end
`endif

endmodule
```

4. Make the system (8/9) Device tree file

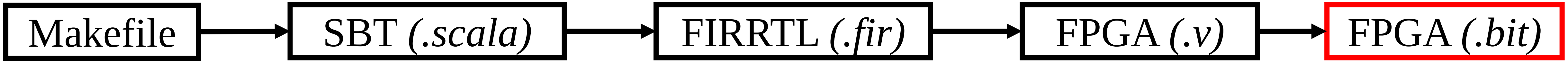

Makefile
SBT *(.scala)*
FIRRTL *(.fir)*
FPGA *(.v)*
FPGA *(.bit)*

Device tree file:
(needed for software)
`riscvconsole.fpga.ArtyA7Top.ArtyA7Config.dts`

Its binary version:
`riscvconsole.fpga.ArtyA7Top.ArtyA7Config.dtb`


```
/dts-v1/;

/ {
        #address-cells = <1>;
        #size-cells = <1>;
        compatible = "freechips,rocketchip-unknown-dev";
        model = "freechips,rocketchip-unknown";
        L26: aliases {
                serial0 = &L12;
        };
        L21: chosen {
                bootargs = "console=hvc0 earlycon=sbi";
        };
        L25: cpus {
                #address-cells = <1>;
                #size-cells = <0>;
                timebase-frequency = <1000000>;
                L6: cpu@0 {
                        clock-frequency = <0>;
                        compatible = "sifive,rocket0", "riscv";
                        d-cache-block-size = <64>;
                        d-cache-sets = <64>;
                        d-cache-size = <4096>;
                        d-tlb-sets = <1>;
                        d-tlb-size = <4>;
                        device_type = "cpu";
                        hardware-exec-breakpoint-count = <1>;
                        i-cache-block-size = <64>;
                        i-cache-sets = <64>;
                        i-cache-size = <4096>;
                        i-tlb-sets = <1>;
                        i-tlb-size = <4>;
                        mmu-type = "riscv,sv32";
                        next-level-cache = <&L16>;
                        reg = <0x0>;
                        riscv,isa = "rv32imac";
                        riscv,pmpgranularity = <4>;
                        riscv,pmpregions = <8>;
                        status = "okay";
                        timebase-frequency = <1000000>;
                        tlb-split;
                        L4: interrupt-controller {
                                #interrupt-cells = <1>;
                                compatible = "riscv,cpu-intc";
                                interrupt-controller;
                        };
                };
        };
```

4. Make the system (9/9) `$ make bit`

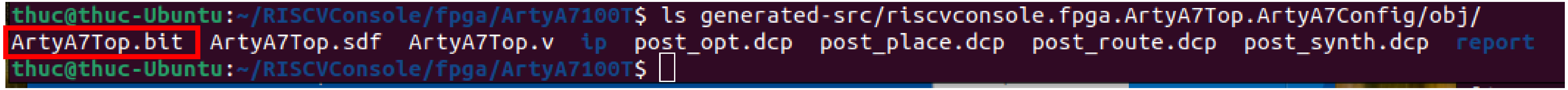


The `$ make default` is just for generating Verilog.
Now, to compile the FPGA:

```
$ make bit
```

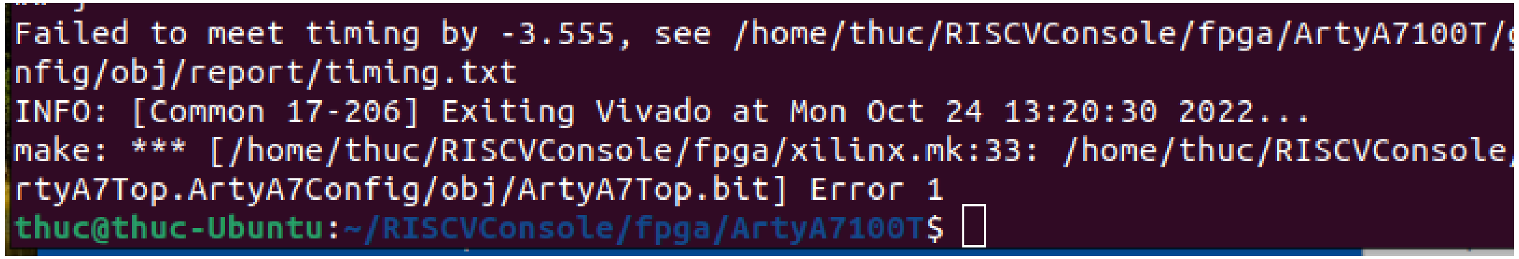
This will compile Verilog to **.bit** file for programming the FPGA

After `$ make bit`, you can find the **.bit** file for programming the FPGA in:
`generated-src/riscvconsole.fpga.ArtyA7Top.ArtyA7Config/obj/`



```
thuc@thuc-Ubuntu:~/RISCVConsole/fpga/ArtyA7100T$ ls generated-src/riscvconsole.fpga.ArtyA7Top.ArtyA7Config/obj/
ArtyA7Top.bit  ArtyA7Top.sdf  ArtyA7Top.v  ip  post_opt.dcp  post_place.dcp  post_route.dcp  post_synth.dcp  report
thuc@thuc-Ubuntu:~/RISCVConsole/fpga/ArtyA7100T$ 
```

***Note:** if the `$ make bit` has an error related to timing, it is fine as long as the **.bit** file was generated.



```
Failed to meet timing by -3.555, see /home/thuc/RISCVConsole/fpga/ArtyA7100T/
nfig/obj/report/timing.txt
INFO: [Common 17-206] Exiting Vivado at Mon Oct 24 13:20:30 2022...
make: *** [/home/thuc/RISCVConsole/fpga/xilinx.mk:33: /home/thuc/RISCVConsole/
rtyA7Top.ArtyA7Config/obj/ArtyA7Top.bit] Error 1
thuc@thuc-Ubuntu:~/RISCVConsole/fpga/ArtyA7100T$ 
```

Outline

1. Introduction
2. System architecture
3. Git clone and prepare
4. Make the system
5. Program Arty-A7
6. Using IntelliJ IDEA-IC
7. Modifying system by Scala config
8. Practice: system modification

5. **Program Arty-A7** (1/10) Prepare SD-card

Before programming the Arty-A7 FPGA board,
you must prepare the software on the SD card.

First, you'll need the **gptfdisk** tool to format the SD card.
If you don't have it already, do the following to install it:

From your home folder:

```
$ git clone https://github.com/tmagik/gptfdisk.git
$ cd gptfdisk/
$ make -j`nproc`
```

Put the SD-card to your PC, then:

Go to gptfdisk folder:

```
$ cd gptfdisk/
$ sudo ./gdisk /dev/sd?
```

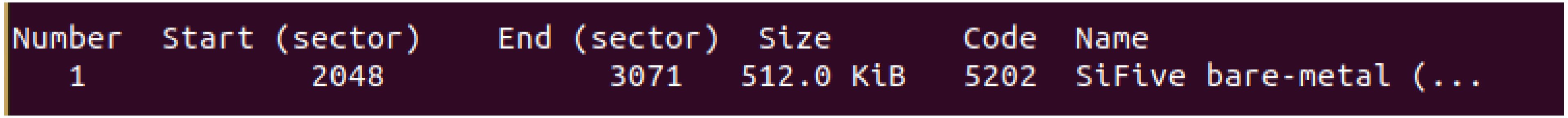
The `?` points to the SD card. For example: `/dev/sdb`

5. Program Arty-A7 (2/10) Prepare SD-card

Some commands to use while in the **gptfdisk** tool:

`$ p`	:	print partitions information
`$ d`	:	delete partition
`$ n`	:	create new partition
`$ w`	:	write partition
`$ q`	:	exit gptfdisk

You’ll need to format the SD card to look like this:

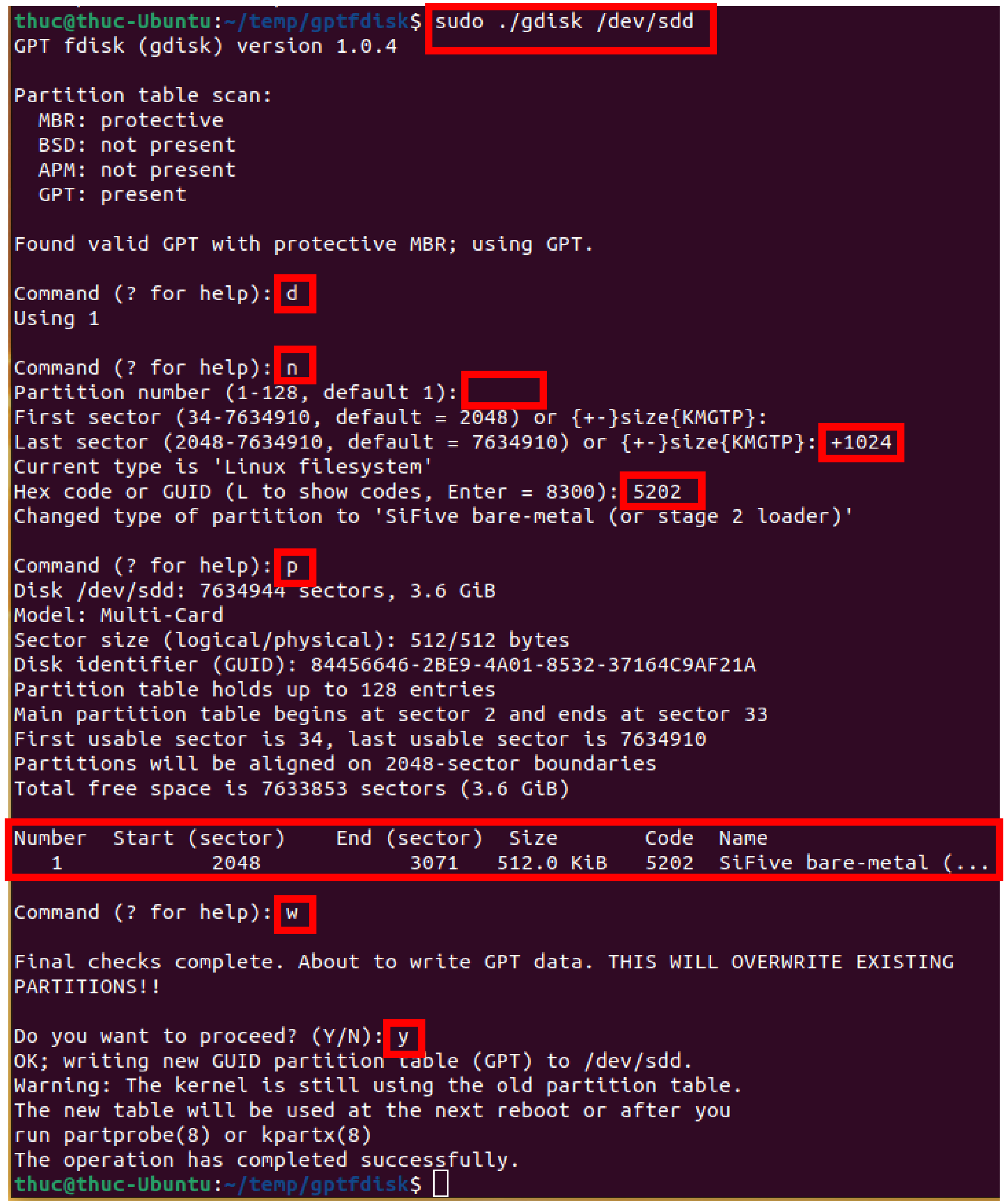


```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048            3071   512.0 KiB   5202  SiFive bare-metal (...
```

5. Program Arty-A7 (3/10) Prepare SD-card

Example commands:

```
$ sudo ./gdisk /dev/sdb
$ d
$ n
$ (Enter)
$ +1024
$ 5202
$ p
$ w
$ y
```



```
thuc@thuc-Ubuntu:~/temp/gptfdisk$ sudo ./gdisk /dev/sdd
GPT fdisk (gdisk) version 1.0.4

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help): d
Using 1

Command (? for help): n
Partition number (1-128, default 1):
First sector (34-7634910, default = 2048) or {+-}size{KMGTP}:
Last sector (2048-7634910, default = 7634910) or {+-}size{KMGTP}: +1024
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300): 5202
Changed type of partition to 'SiFive bare-metal (or stage 2 loader)'

Command (? for help): p
Disk /dev/sdd: 7634944 sectors, 3.6 GiB
Model: Multi-Card
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 84456646-2BE9-4A01-8532-37164C9AF21A
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 7634910
Partitions will be aligned on 2048-sector boundaries
Total free space is 7633853 sectors (3.6 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048            3071   512.0 KiB   5202  SiFive bare-metal (...

Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!

Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/sdd.
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.
thuc@thuc-Ubuntu:~/temp/gptfdisk$
```

5. Program Arty-A7 (4/10) Prepare software

Now, prepare the after-boot software:

From your RISCVConsole folder:

```
$ cd RISCVConsole/
```

Go to:

```
$ cd software/RISCVConsoleCode/
```

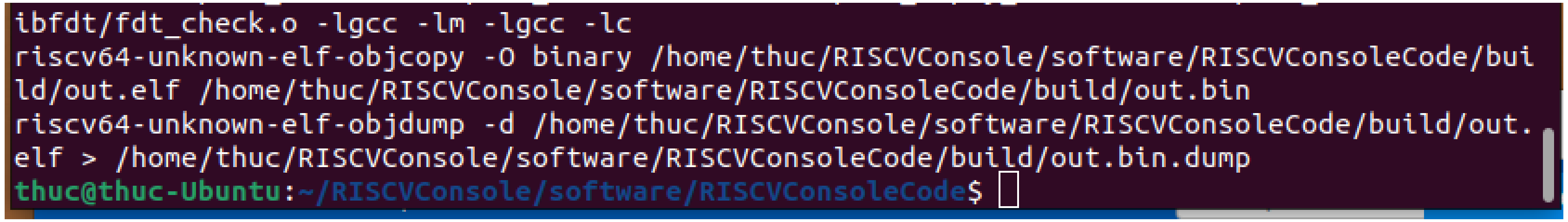
Remember to have the RISC-V toolchain available in the **PATH**:

```
$ export PATH=/opt/riscv/bin/:$PATH
```

Finally, compile the software:

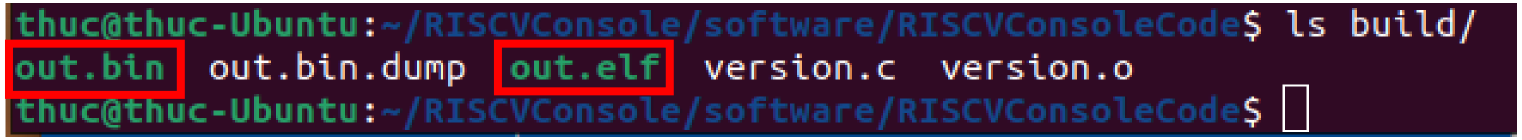
```
$ make bin
```

Terminal after `$ make bin` :



```
ibfdt/fdt_check.o -lgcc -lm -lgcc -lc
riscv64-unknown-elf-objcopy -O binary /home/thuc/RISCVConsole/software/RISCVConsoleCode/bui
ld/out.elf /home/thuc/RISCVConsole/software/RISCVConsoleCode/build/out.bin
riscv64-unknown-elf-objdump -d /home/thuc/RISCVConsole/software/RISCVConsoleCode/build/out.
elf > /home/thuc/RISCVConsole/software/RISCVConsoleCode/build/out.bin.dump
thuc@thuc-Ubuntu:~/RISCVConsole/software/RISCVConsoleCode$
```

After `$ make bin`, the compiled software are under the **build** folder:



```
thuc@thuc-Ubuntu:~/RISCVConsole/software/RISCVConsoleCode$ ls build/
out.bin  out.bin.dump  out.elf  version.c  version.o
thuc@thuc-Ubuntu:~/RISCVConsole/software/RISCVConsoleCode$
```

5. Program Arty-A7 (5/10) Prepare software

Now, to write the compiled software to the SD card:

From your RISCVConsole/software/RISCVConsoleCode folder:

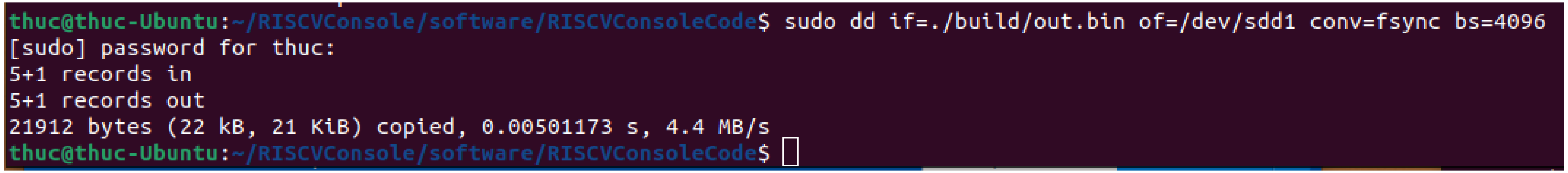
```
$ sudo dd if=./build/out.bin of=/dev/sd?1 conv=fsync bs=4096
```

Again, the **?** points to the SD card.

For example:

```
$ sudo dd if=./build/out.bin of=/dev/sdb1 conv=fsync bs=4096
```

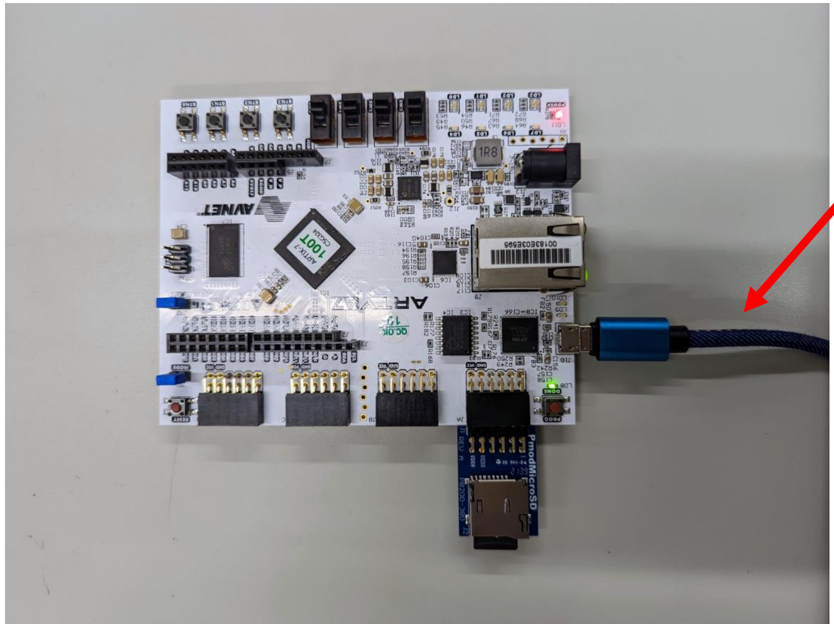
After `$ sudo dd` :



```
thuc@thuc-Ubuntu:~/RISCVConsole/software/RISCVConsoleCode$ sudo dd if=./build/out.bin of=/dev/sdd1 conv=fsync bs=4096
[sudo] password for thuc:
5+1 records in
5+1 records out
21912 bytes (22 kB, 21 KiB) copied, 0.00501173 s, 4.4 MB/s
thuc@thuc-Ubuntu:~/RISCVConsole/software/RISCVConsoleCode$
```

Now, remove the SD card from your PC.
Put it in the PMOD-SD-card and connect to the Arty-A7.

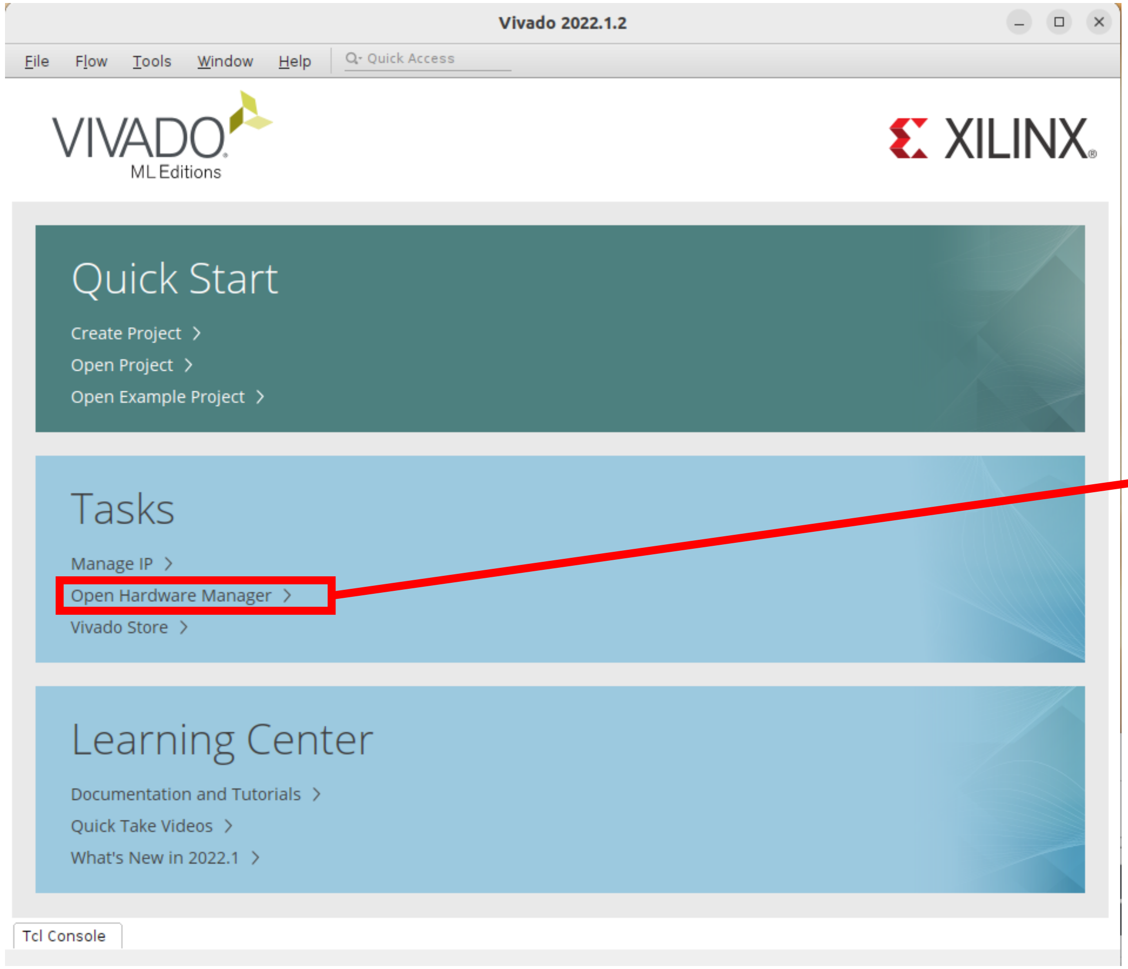
Like this:



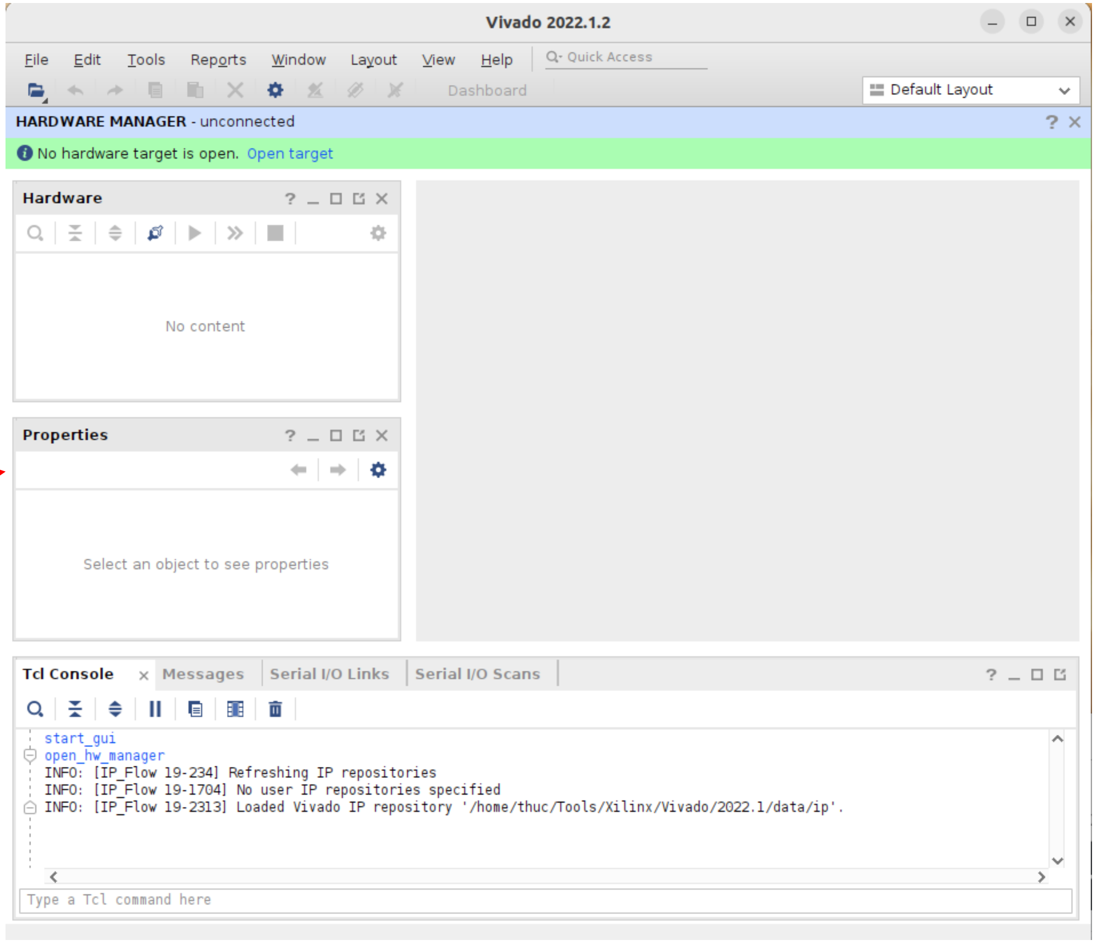
Remember to connect the microUSB-to-USB to your PC.

5. Program Arty-A7 (6/10) Program the Arty-A7

To program the board, open Vivado:



Open hardware manager:



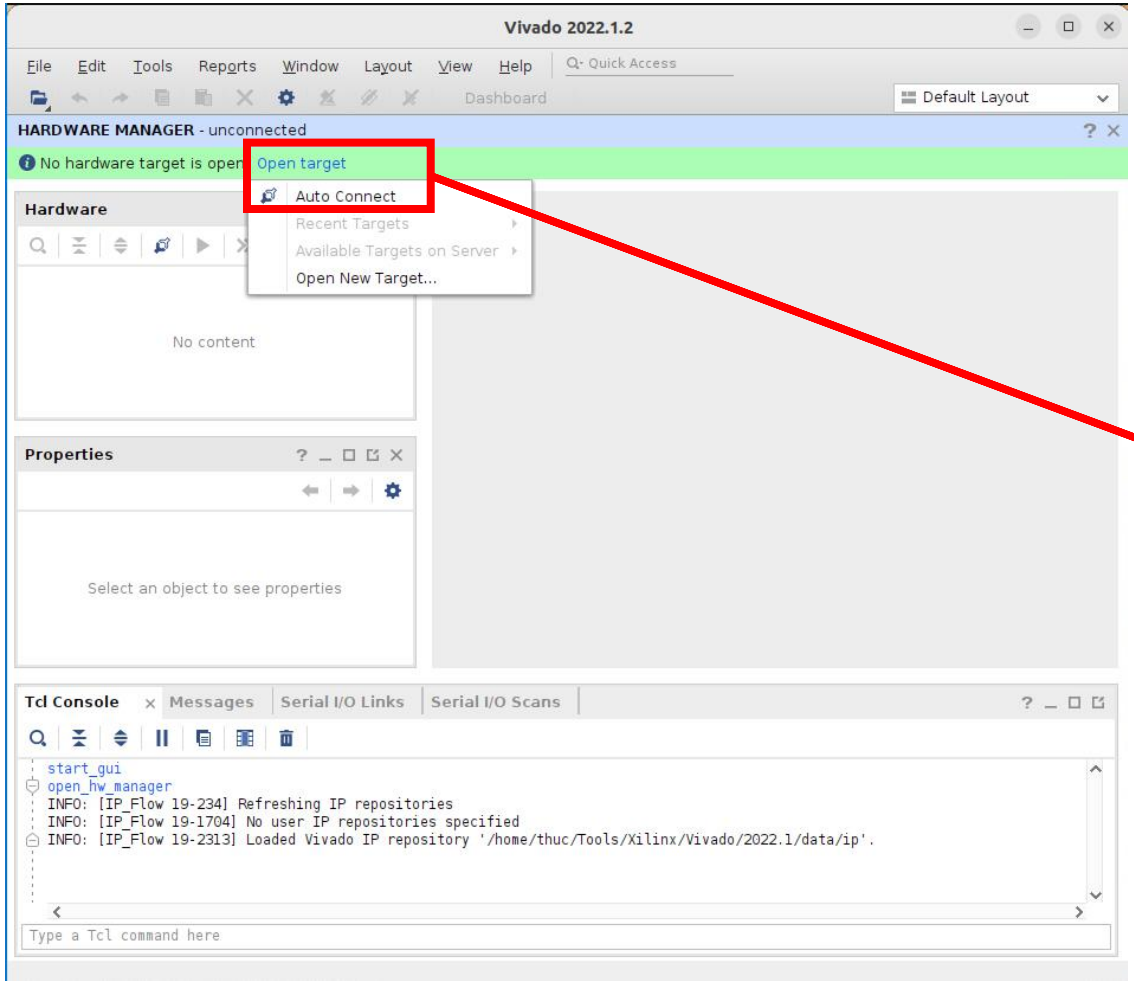
No hardware target is open. Open target

Select an object to see properties

```
start_gui
open_hw_manager
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1704] No user IP repositories specified
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository '/home/thuc/Tools/Xilinx/Vivado/2022.1/data/ip'.
```

5. Program Arty-A7 (7/10) Program the Arty-A7

Open target → Auto connect :



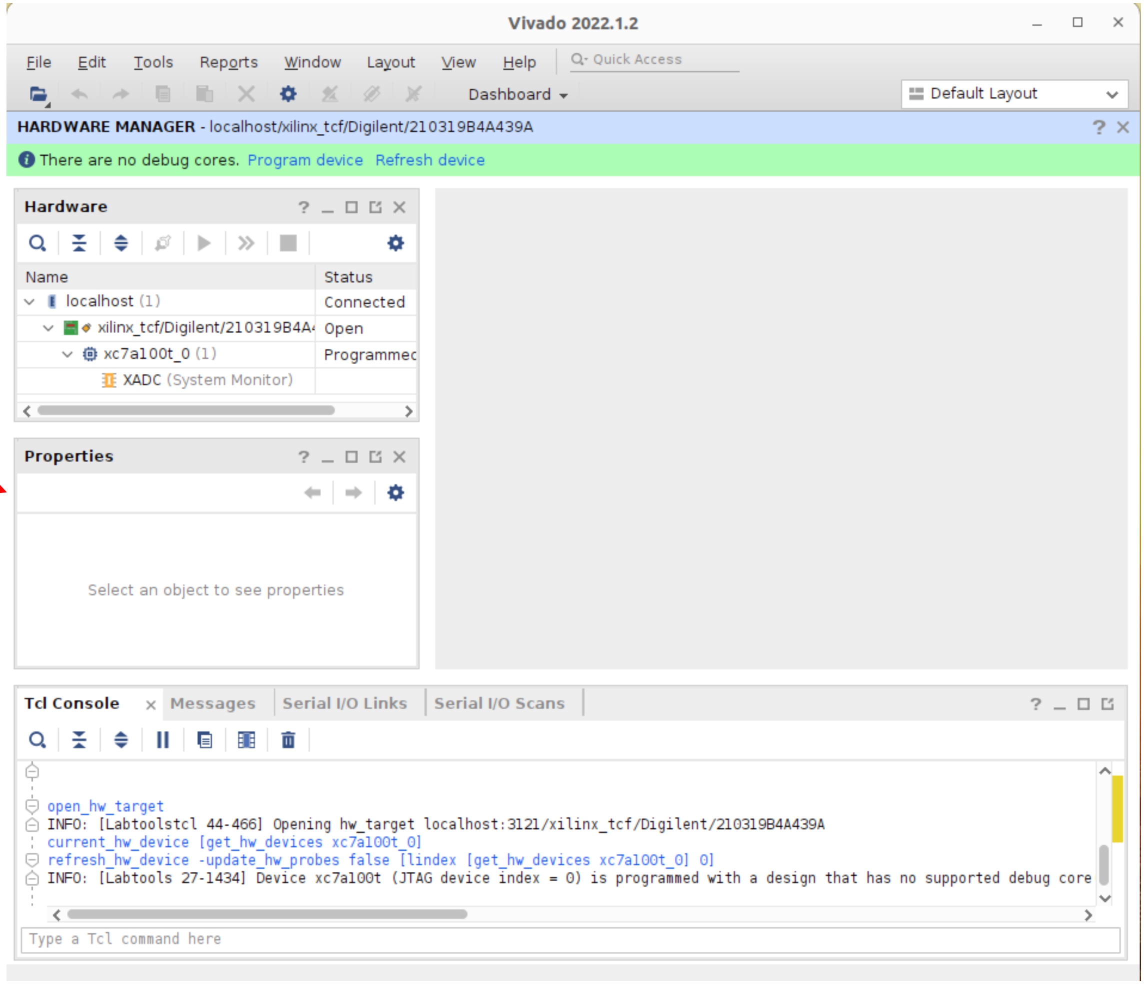
HARDWARE MANAGER - unconnected

No hardware target is open Open target

No content

Select an object to see properties

```
start_gui
open_hw_manager
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1704] No user IP repositories specified
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository '/home/thuc/Tools/Xilinx/Vivado/2022.1/data/ip'.
```



HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210319B4A439A

There are no debug cores. Program device Refresh device

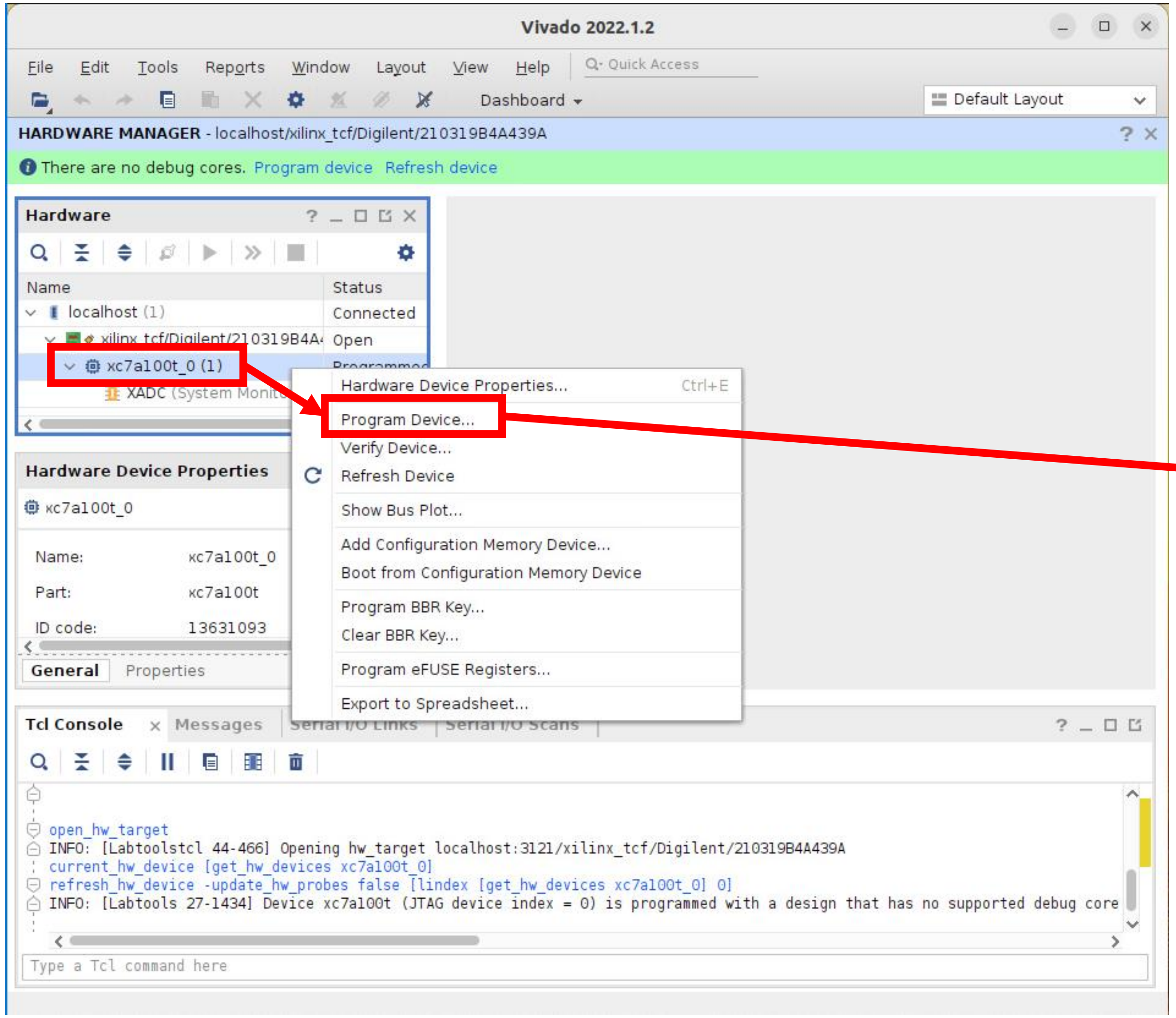
Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210319B4A4	Open
xc7a100t_0 (1)	Programmec
XADC (System Monitor)	

Select an object to see properties

```
open_hw_target
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210319B4A439A
current_hw_device [get_hw_devices xc7a100t_0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a100t_0] 0]
INFO: [Labtools 27-1434] Device xc7a100t (JTAG device index = 0) is programmed with a design that has no supported debug core
```

5. Program Arty-A7 (8/10) Program the Arty-A7

Right-click on the **xc7a100t (1)** → Program device…



Vivado 2022.1.2

HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210319B4A439A

There are no debug cores. Program device Refresh device

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210319B4A4	Open
xc7a100t_0 (1)	Programmed
XADC (System Monit	

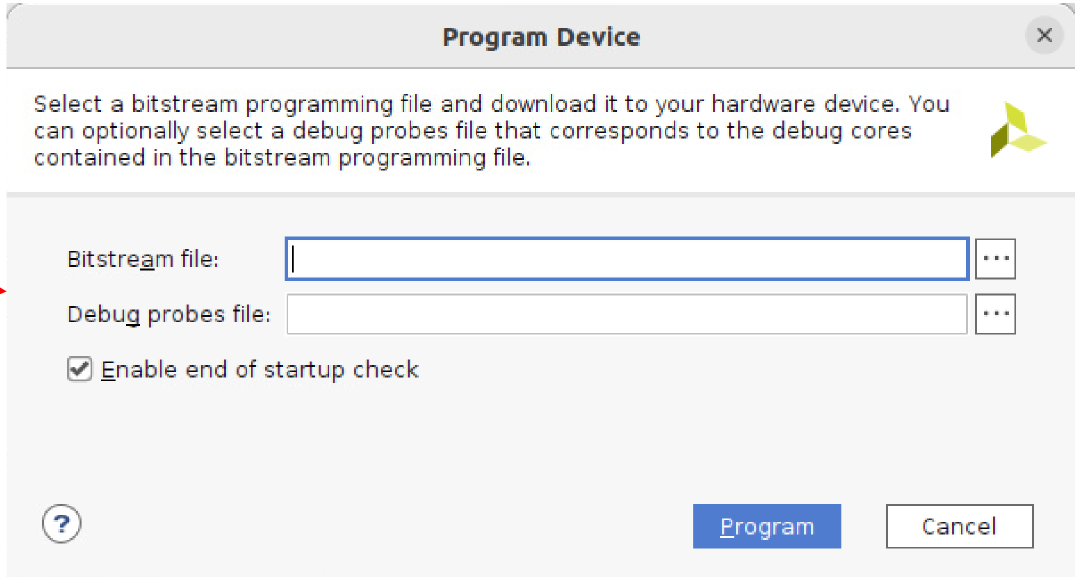
Hardware Device Properties

xc7a100t_0

Name:	xc7a100t_0
Part:	xc7a100t
ID code:	13631093

```
open_hw_target
INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210319B4A439A
current_hw_device [get_hw_devices xc7a100t_0]
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices xc7a100t_0] 0]
INFO: [Labtools 27-1434] Device xc7a100t (JTAG device index = 0) is programmed with a design that has no supported debug core
```

Type a Tcl command here



Program Device

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

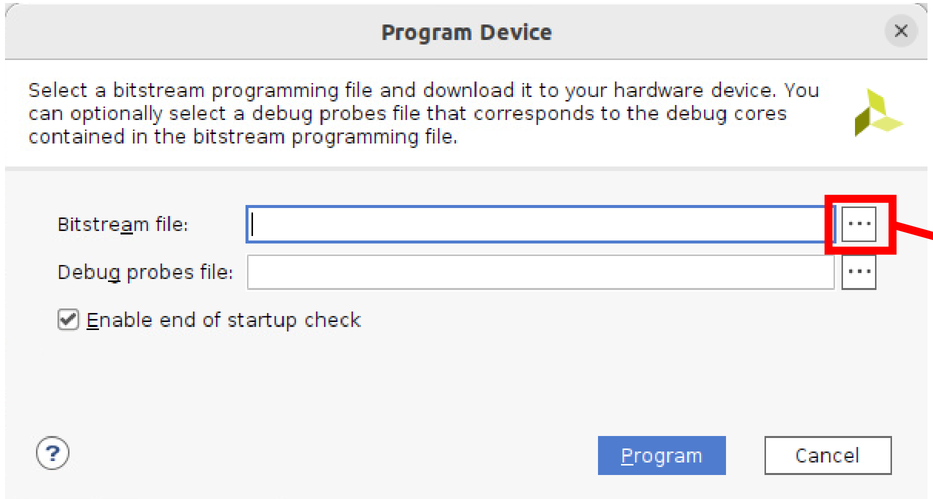
Bitstream file:

Debug probes file:

☑ Enable end of startup check

5. Program Arty-A7 (9/10) Program the Arty-A7

Browse to the **ArtyA7Top.bit** under `RISCVConsole/fpga/ArtyA7100T/generated-src/riscvconsole.fpga.ArtyA7Top.ArtyA7Config/obj` folder



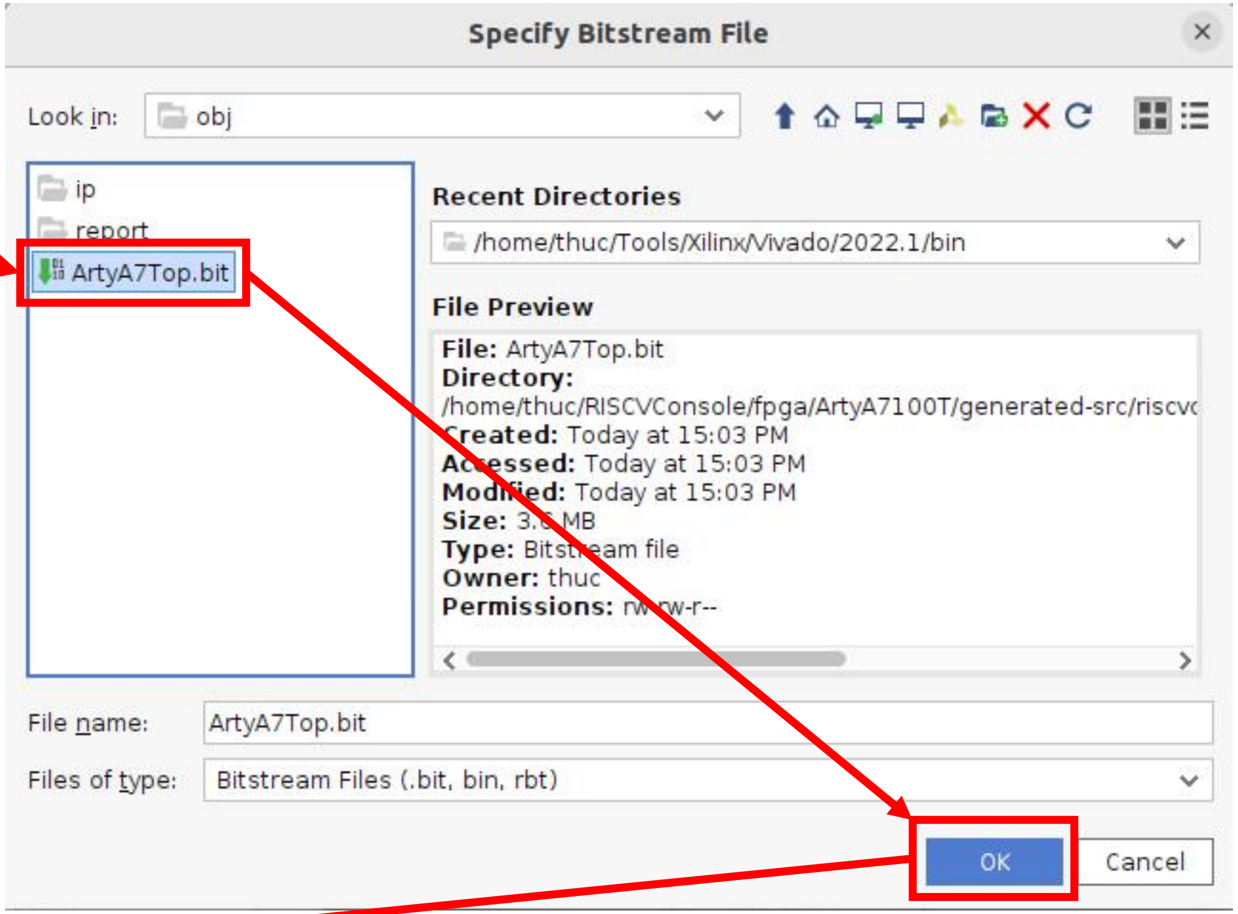
Program Device

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file:

Debug probes file:

☑ Enable end of startup check



Specify Bitstream File

Look in: obj

- ip
- report
- ArtyA7Top.bit

Recent Directories

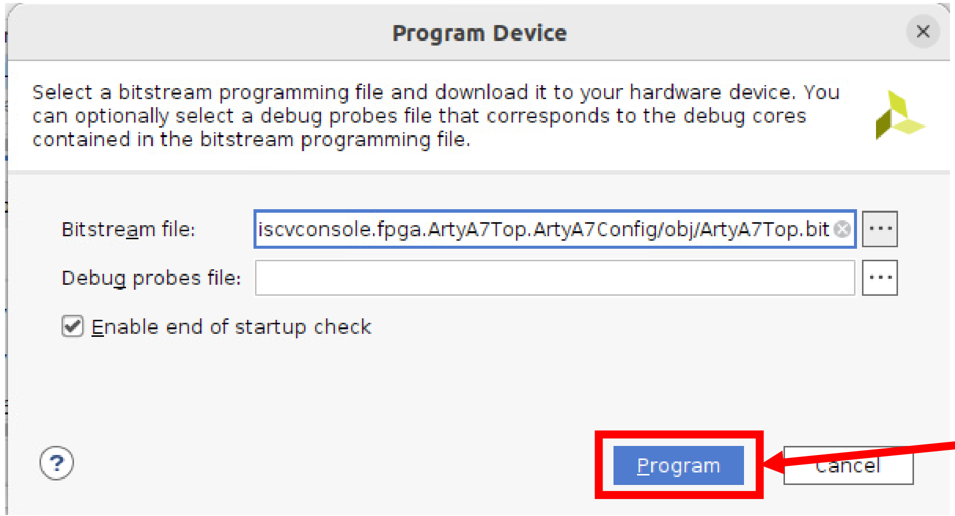
/home/thuc/Tools/Xilinx/Vivado/2022.1/bin

File Preview

File: ArtyA7Top.bit
Directory:
/home/thuc/RISCVConsole/fpga/ArtyA7100T/generated-src/riscvc
Created: Today at 15:03 PM
Accessed: Today at 15:03 PM
Modified: Today at 15:03 PM
Size: 3.6 MB
Type: Bitstream file
Owner: thuc
Permissions: rw-rw-r--

File name: ArtyA7Top.bit

Files of type: Bitstream Files (.bit, bin, rbt)



Program Device

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: iscvconsole.fpga.ArtyA7Top.ArtyA7Config/obj/ArtyA7Top.bit

Debug probes file:

☑ Enable end of startup check

Finally, **OK** then **Program**.

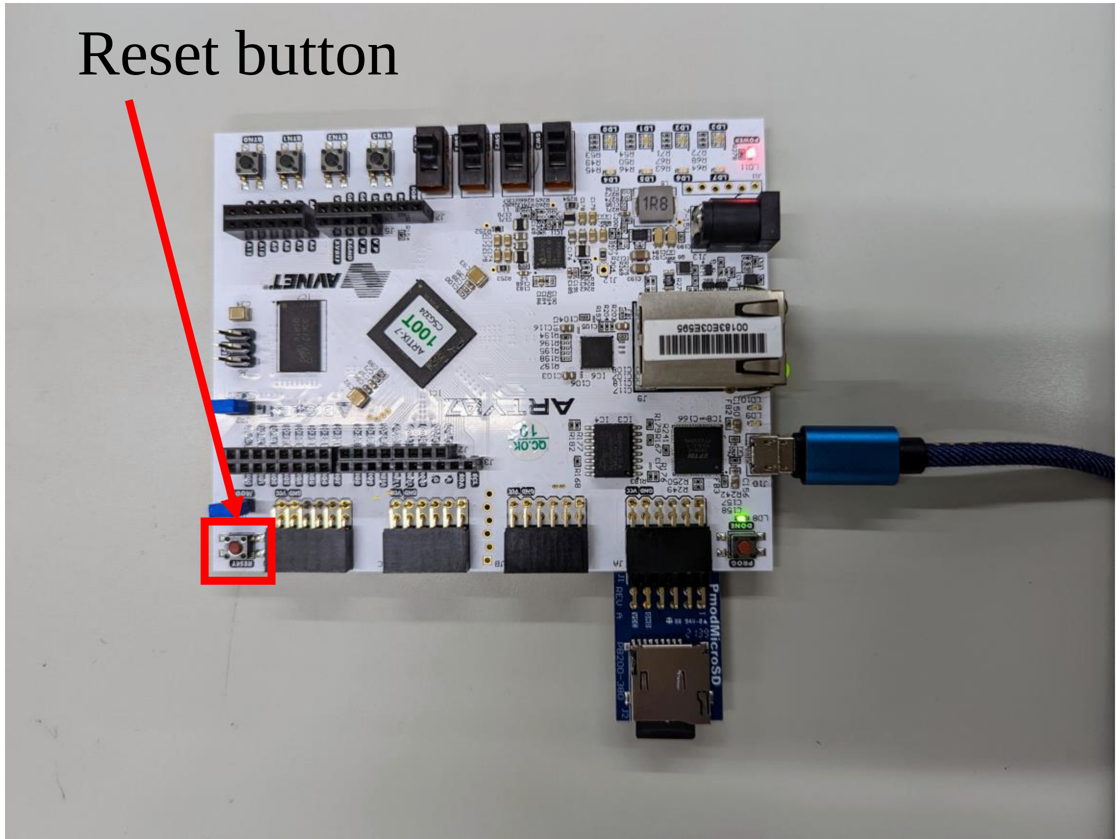
5. Program Arty-A7 (10/10) UART terminal

To open the UART terminal:

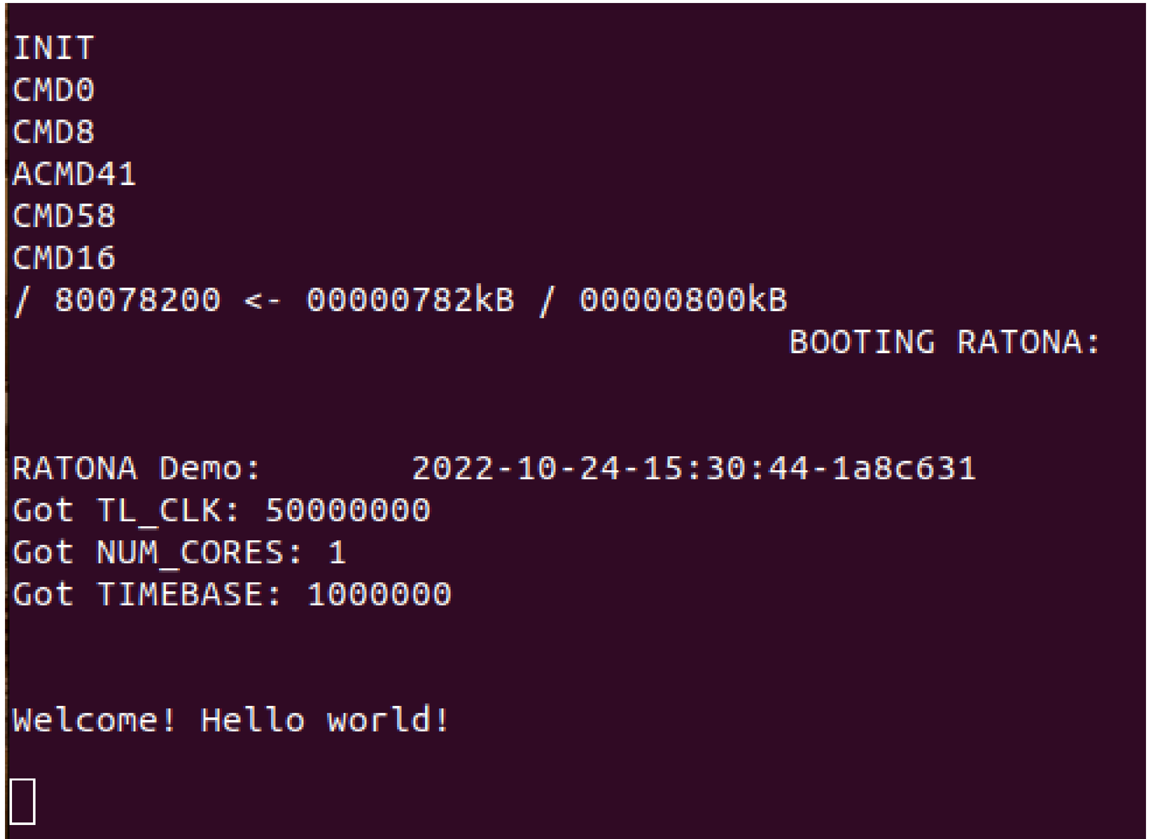
```
$ sudo minicom -b 115200 -D /dev/ttyUSB?
```

Where ? points to the Arty-A7 microUSB connection.

For example: `$ sudo minicom -b 115200 -D /dev/ttyUSB1`



Printing in the UART terminal:



```
INIT
CMD0
CMD8
ACMD41
CMD58
CMD16
/ 80078200 <- 00000782kB / 00000800kB
                                             BOOTING RATONA:


RATONA Demo:       2022-10-24-15:30:44-1a8c631
Got TL_CLK: 50000000
Got NUM_CORES: 1
Got TIMEBASE: 1000000


Welcome! Hello world!
```

Outline

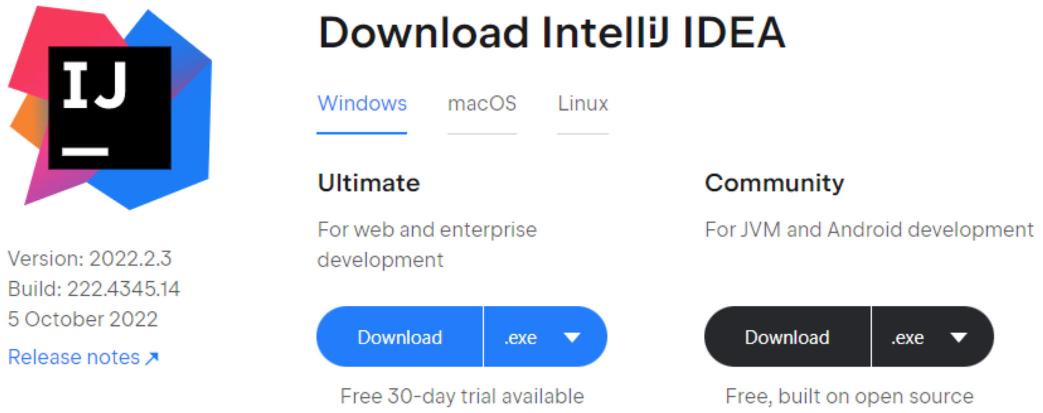
1. Introduction
2. System architecture
3. Git clone and prepare
4. Make the system
5. Program Arty-A7
6. Using IntelliJ IDEA-IC
7. Modifying system by Scala config
8. Practice: system modification

6. Using IntelliJ IDEA-IC (1/3) Download and install

To write/modify Scala sources efficiently,
we need an IDE *(Integrated Development Environment)* tool.

- *SBT* is the compiler for *Scala*
- **IntelliJ IDEA-IC** is the GUI for *SBT*

→ **IntelliJ IDEA-IC** for *Scala* is like Visual Studio for *C++*



Download IntelliJ IDEA

Version: 2022.2.3
Build: 222.4345.14
5 October 2022

Ultimate

For web and enterprise development

Free 30-day trial available

Community

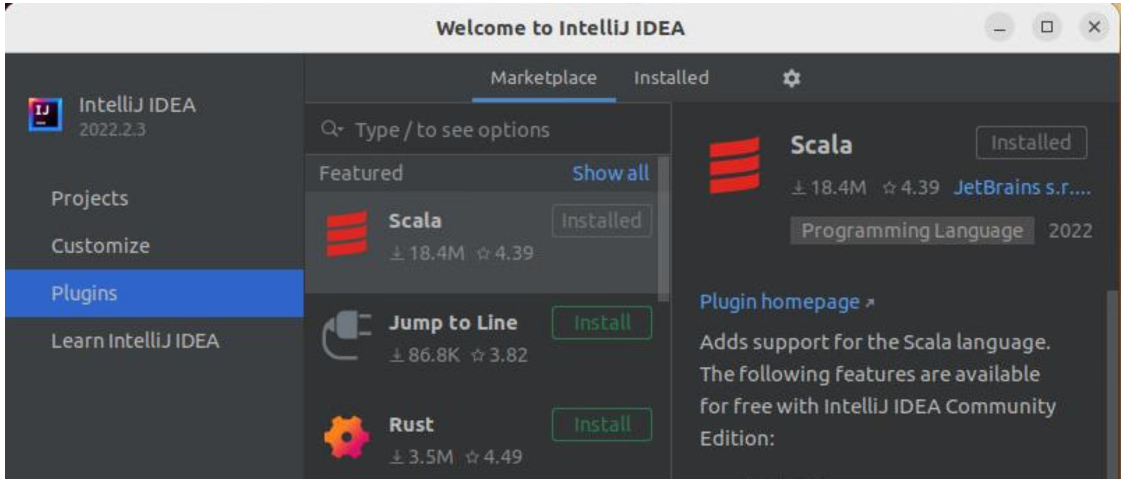
For JVM and Android development

Free, built on open source

To install **IntelliJ IDEA-IC**,
just follow their website:
https://www.jetbrains.com/idea/

They have a free version: *Community*

The first time it opens,
remember to install
the *Scala plugin*.



Welcome to IntelliJ IDEA

IntelliJ IDEA
2022.2.3

Featured

Scala
18.4M ☆4.39

Jump to Line
86.8K ☆3.82

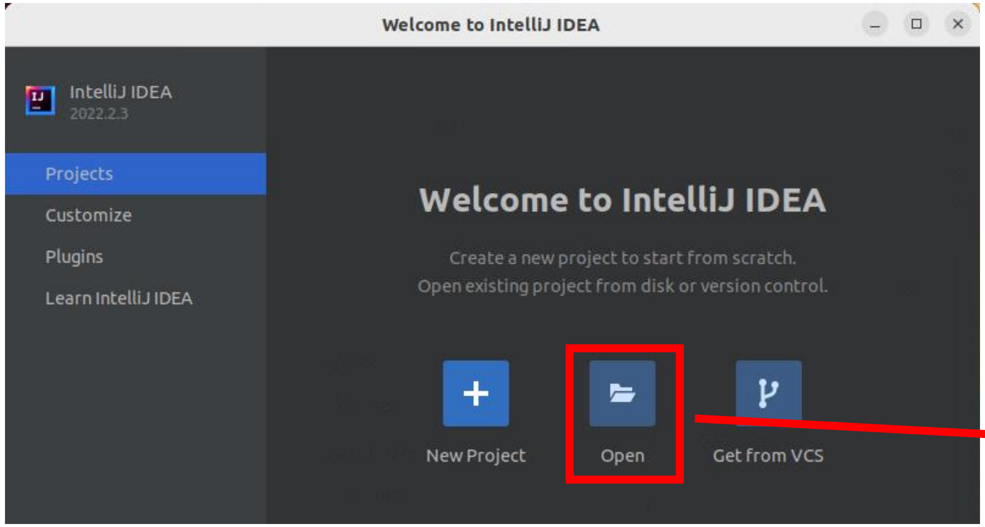
Rust
3.5M ☆4.49

Scala
18.4M ☆4.39 JetBrains s.r....
Programming Language 2022

Adds support for the Scala language. The following features are available for free with IntelliJ IDEA Community Edition:

6. Using IntelliJ IDEA-IC (2/3) Import project

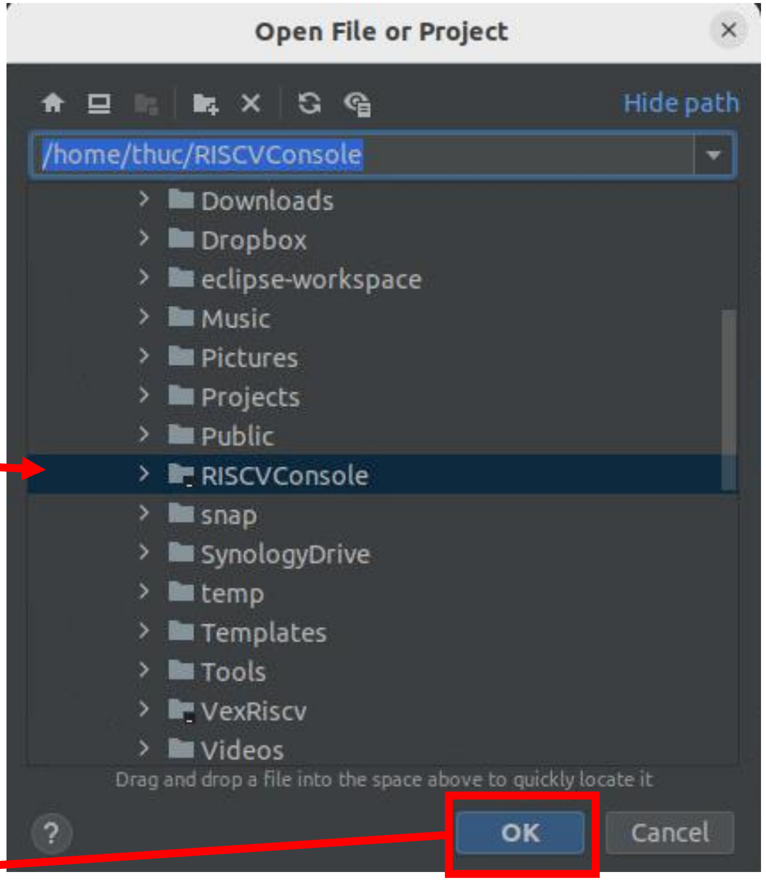
To import existing project, *Open*:



Welcome to IntelliJ IDEA

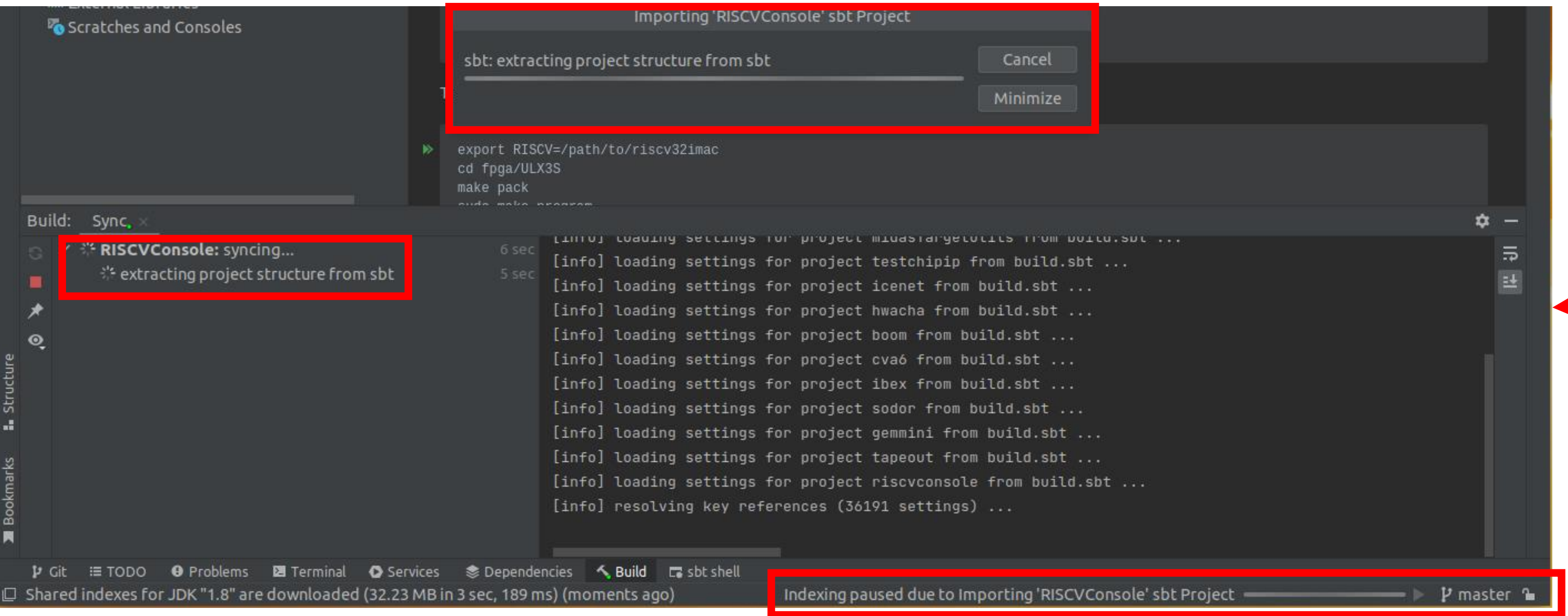
Create a new project to start from scratch.
Open existing project from disk or version control.

Choose the project to import:



/home/thuc/RISCVConsole

Drag and drop a file into the space above to quickly locate it



sbt: extracting project structure from sbt

```
export RISCV=/path/to/riscv32imac
cd fpga/ULX3S
make pack
sudo make program
```

RISCVConsole: syncing...
extracting project structure from sbt

```
[info] loading settings for project midasTargetUtils from build.sbt ...
[info] loading settings for project testchipip from build.sbt ...
[info] loading settings for project icenet from build.sbt ...
[info] loading settings for project hwacha from build.sbt ...
[info] loading settings for project boom from build.sbt ...
[info] loading settings for project cva6 from build.sbt ...
[info] loading settings for project ibex from build.sbt ...
[info] loading settings for project sodor from build.sbt ...
[info] loading settings for project gemmini from build.sbt ...
[info] loading settings for project tapeout from build.sbt ...
[info] loading settings for project riscvconsole from build.sbt ...
[info] resolving key references (36191 settings) ...
```

Shared indexes for JDK "1.8" are downloaded (32.23 MB in 3 sec, 189 ms) (moments ago)

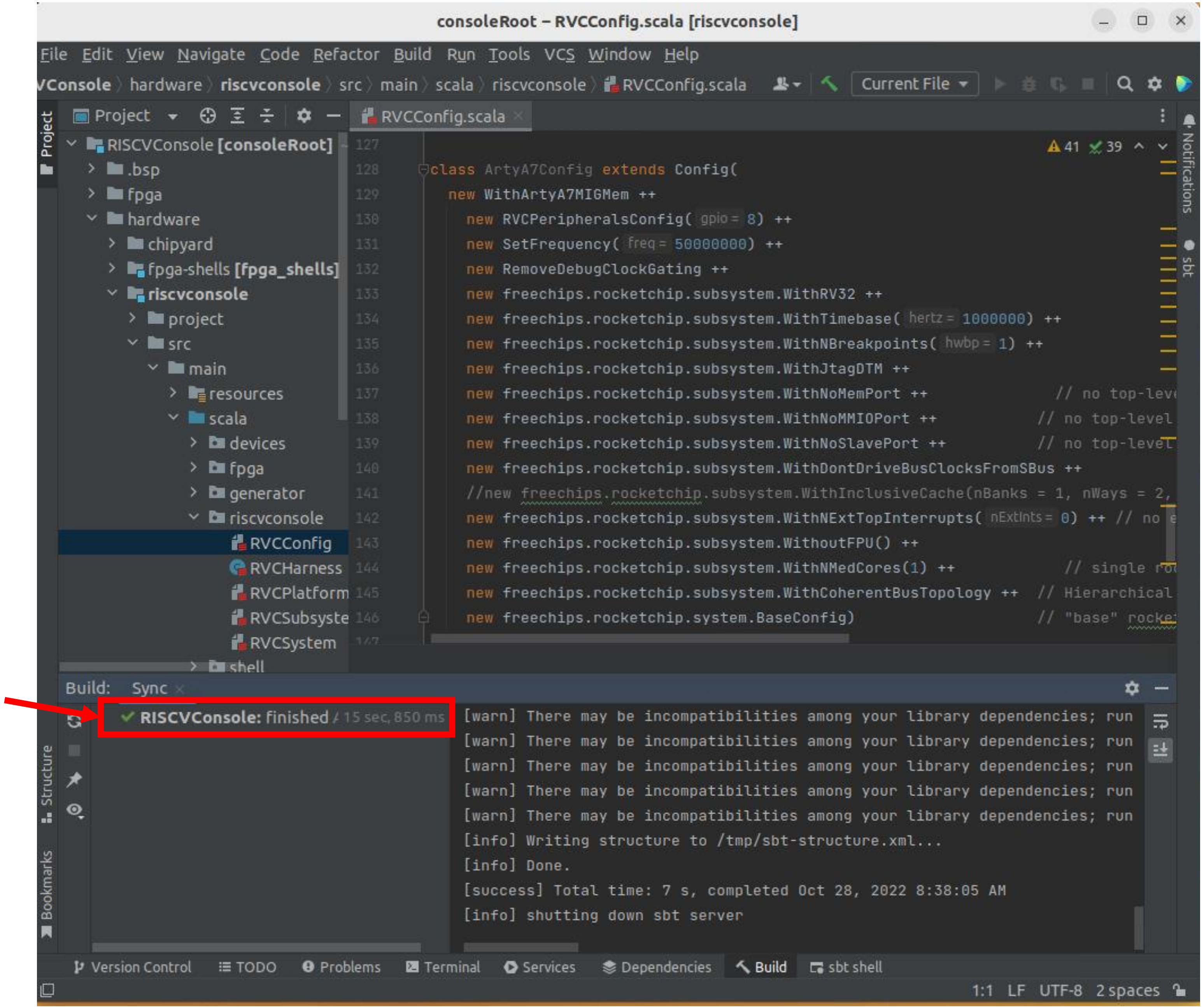
Indexing paused due to Importing 'RISCVConsole' sbt Project

It could take a while for the importing to finish.

6. Using IntelliJ IDEA-IC (3/3) Import project

Now all the variables are properly linked.
You can:

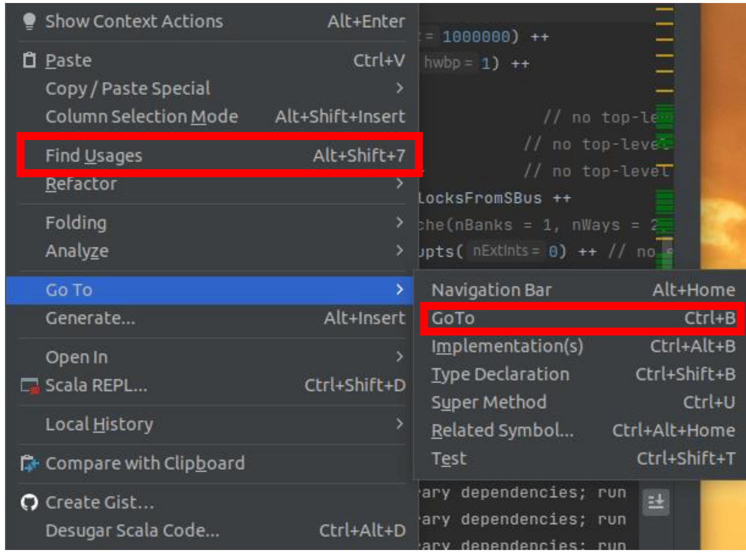
- Find usages
- Go to



```
class ArtyA7Config extends Config(
  new WithArtyA7MIGMem ++
    new RVCPeripheralsConfig( gpio = 8) ++
    new SetFrequency( freq = 50000000) ++
    new RemoveDebugClockGating ++
    new freechips.rocketchip.subsystem.WithRV32 ++
    new freechips.rocketchip.subsystem.WithTimebase( hertz = 1000000) ++
    new freechips.rocketchip.subsystem.WithNBreakpoints( hwbp = 1) ++
    new freechips.rocketchip.subsystem.WithJtagDTM ++
    new freechips.rocketchip.subsystem.WithNoMemPort ++            // no top-leve
    new freechips.rocketchip.subsystem.WithNoMMIOPort ++         // no top-level
    new freechips.rocketchip.subsystem.WithNoSlavePort ++        // no top-level
    new freechips.rocketchip.subsystem.WithDontDriveBusClocksFromSBus ++
    //new freechips.rocketchip.subsystem.WithInclusiveCache(nBanks = 1, nWays = 2,
    new freechips.rocketchip.subsystem.WithNExtTopInterrupts( nExtInts = 0) ++ // no e
    new freechips.rocketchip.subsystem.WithoutFPU() ++
    new freechips.rocketchip.subsystem.WithNMedCores(1) ++          // single ro
    new freechips.rocketchip.subsystem.WithCoherentBusTopology ++  // Hierarchical
    new freechips.rocketchip.system.BaseConfig)                   // "base" rocke
```

```
[warn] There may be incompatibilities among your library dependencies; run
[warn] There may be incompatibilities among your library dependencies; run
[warn] There may be incompatibilities among your library dependencies; run
[warn] There may be incompatibilities among your library dependencies; run
[warn] There may be incompatibilities among your library dependencies; run
[info] Writing structure to /tmp/sbt-structure.xml...
[info] Done.
[success] Total time: 7 s, completed Oct 28, 2022 8:38:05 AM
[info] shutting down sbt server
```

When it finishes.

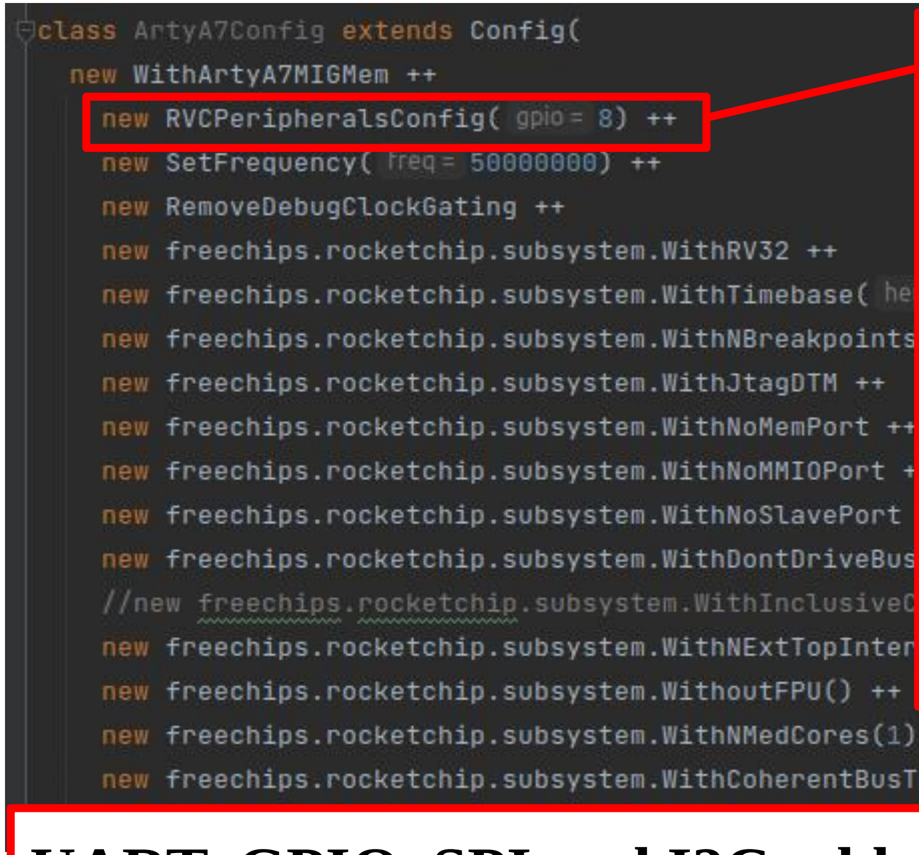


Outline

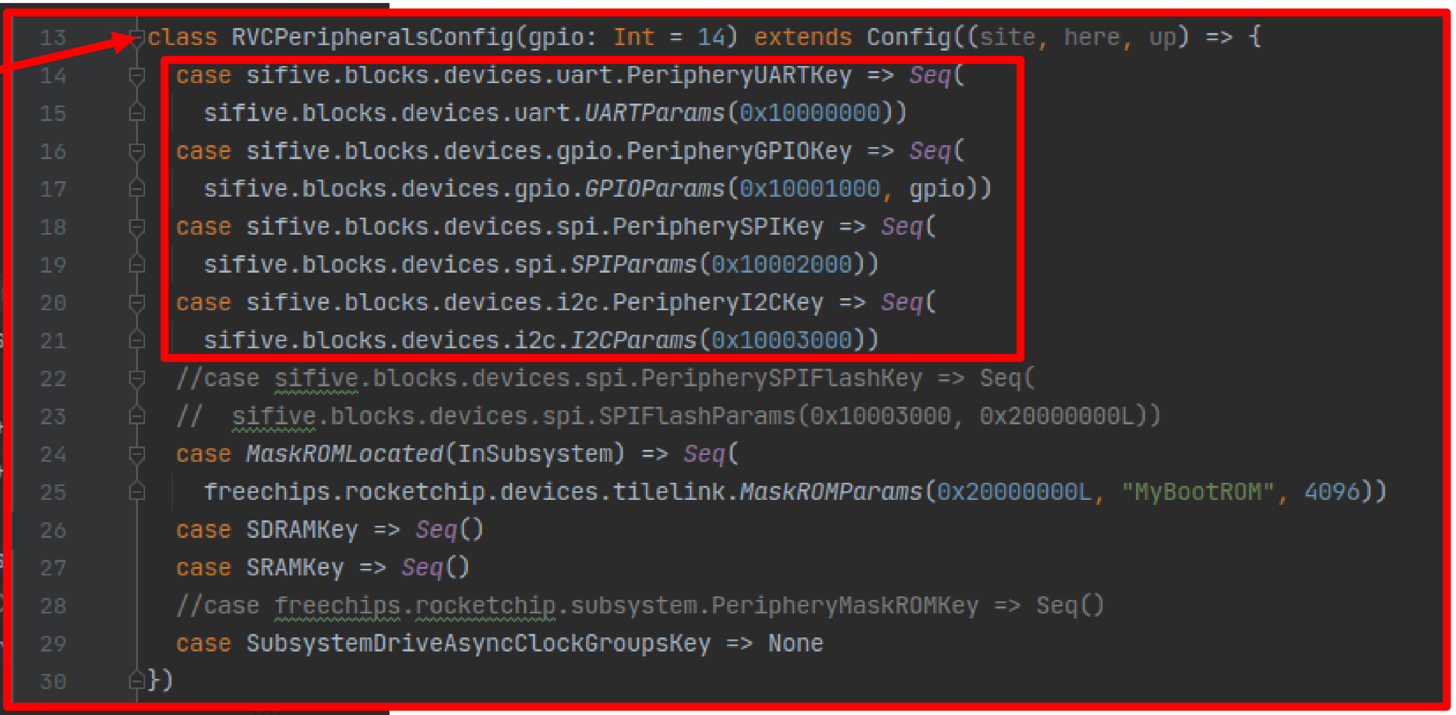
1. Introduction
2. System architecture
3. Git clone and prepare
4. Make the system
5. Program Arty-A7
6. Using IntelliJ IDEA-IC
7. Modifying system by Scala config
8. Practice: system modification

7. Modifying system by Scala config (1/10) Peripheral

At `RISCVConsole/hardware/riscvconsole/src/main/scala/riscvconsole/RVCConfig.scala`:

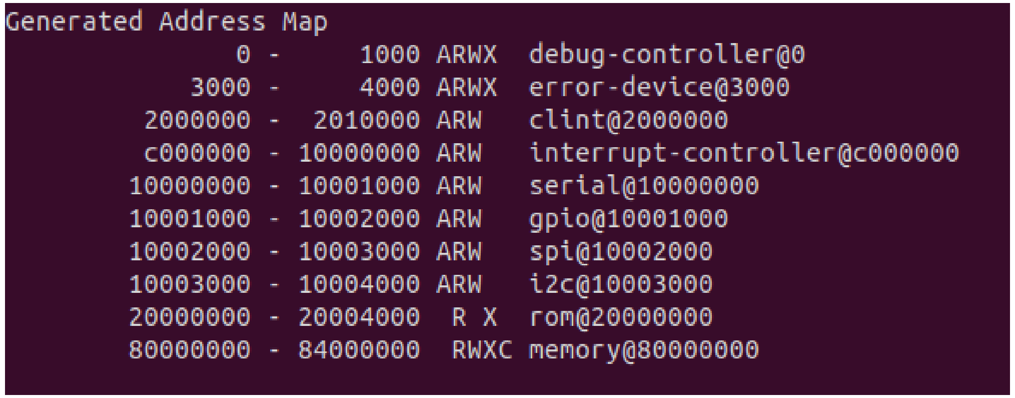


```
class ArtyA7Config extends Config(
  new WithArtyA7MIGMem ++
    new RVCPeripheralsConfig( gpio = 8) ++
    new SetFrequency( freq = 50000000) ++
    new RemoveDebugClockGating ++
    new freechips.rocketchip.subsystem.WithRV32 ++
    new freechips.rocketchip.subsystem.WithTimebase( he
    new freechips.rocketchip.subsystem.WithNBreakpoints
    new freechips.rocketchip.subsystem.WithJtagDTM ++
    new freechips.rocketchip.subsystem.WithNoMemPort ++
    new freechips.rocketchip.subsystem.WithNoMMIOPort +
    new freechips.rocketchip.subsystem.WithNoSlavePort
    new freechips.rocketchip.subsystem.WithDontDriveBus
    //new freechips.rocketchip.subsystem.WithInclusiveC
    new freechips.rocketchip.subsystem.WithNExtTopInter
    new freechips.rocketchip.subsystem.WithoutFPU() ++
    new freechips.rocketchip.subsystem.WithNMedCores(1) ++            // single
    new freechips.rocketchip.subsystem.WithCoherentBusTopology ++  // Hierarchi
                                                                   // "base" ro
```



```
class RVCPeripheralsConfig(gpio: Int = 14) extends Config((site, here, up) => {
  case sifive.blocks.devices.uart.PeripheryUARTKey => Seq(
    sifive.blocks.devices.uart.UARTParams(0x10000000))
  case sifive.blocks.devices.gpio.PeripheryGPIOKey => Seq(
    sifive.blocks.devices.gpio.GPIOParams(0x10001000, gpio))
  case sifive.blocks.devices.spi.PeripherySPIKey => Seq(
    sifive.blocks.devices.spi.SPIParams(0x10002000))
  case sifive.blocks.devices.i2c.PeripheryI2CKey => Seq(
    sifive.blocks.devices.i2c.I2CParams(0x10003000))
  //case sifive.blocks.devices.spi.PeripherySPIFlashKey => Seq(
  //  sifive.blocks.devices.spi.SPIFlashParams(0x10003000, 0x20000000L))
  case MaskROMLocated(InSubsystem) => Seq(
    freechips.rocketchip.devices.tilelink.MaskROMParams(0x20000000L, "MyBootROM", 4096))
  case SDRAMKey => Seq()
  case SRAMKey => Seq()
  //case freechips.rocketchip.subsystem.PeripheryMaskROMKey => Seq()
  case SubsystemDriveAsyncClockGroupsKey => None
})
```

UART, GPIO, SPI and I2C added
(Debug, CLINT, PLIC, error devices and ROM are mandatory for the system)

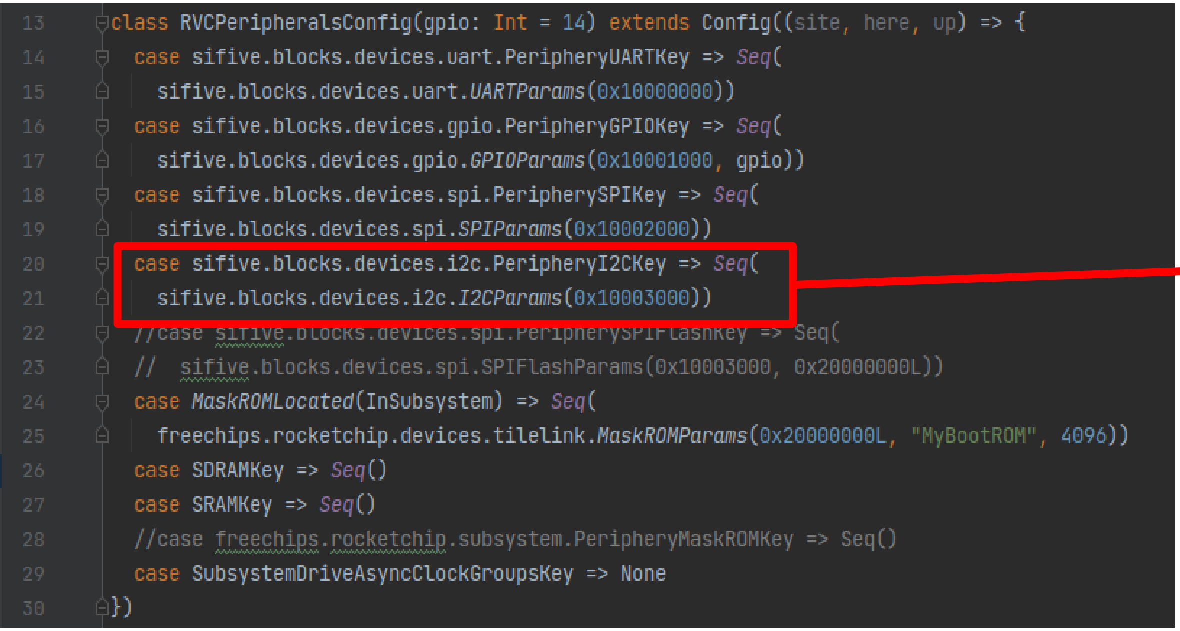


```
Generated Address Map
               0 -     1000 ARWX  debug-controller@0
            3000 -     4000 ARWX  error-device@3000
         2000000 -  2010000 ARW   clint@2000000
         c000000 - 10000000 ARW   interrupt-controller@c000000
        10000000 - 10001000 ARW   serial@10000000
        10001000 - 10002000 ARW   gpio@10001000
        10002000 - 10003000 ARW   spi@10002000
        10003000 - 10004000 ARW   i2c@10003000
        20000000 - 20004000  R X  rom@20000000
        80000000 - 84000000  RWXC memory@80000000
```

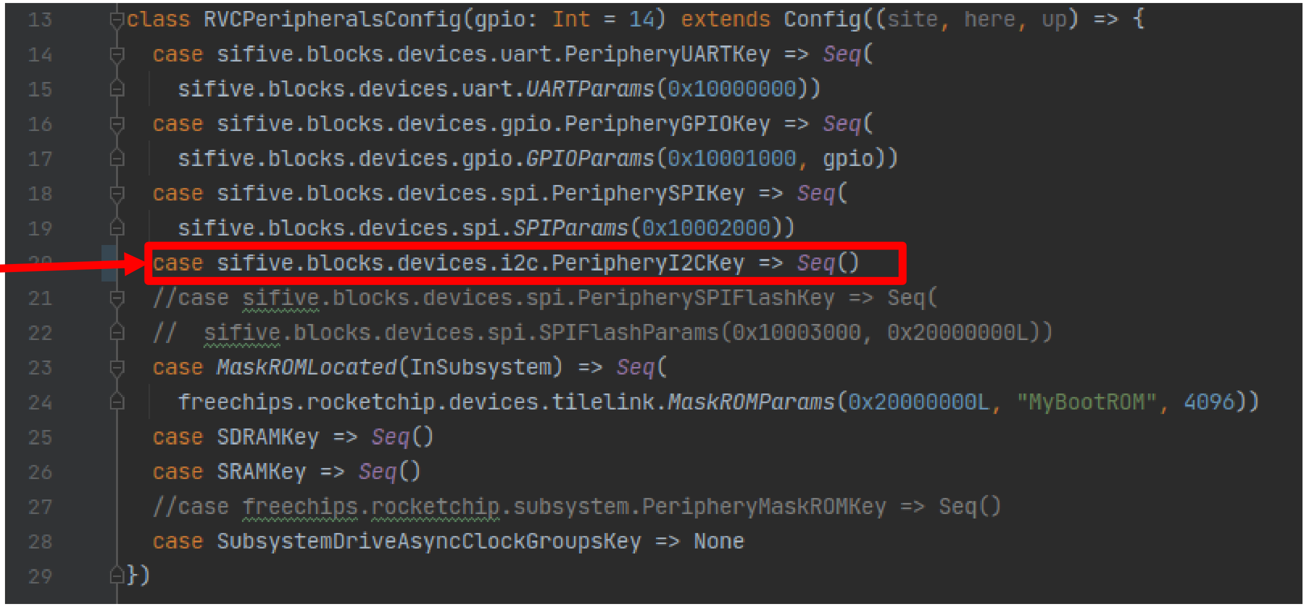
7. Modifying system by Scala config (2/10) Peripheral

At `RISCVConsole/hardware/riscvconsole/src/main/scala/riscvconsole/RVCConfig.scala`:

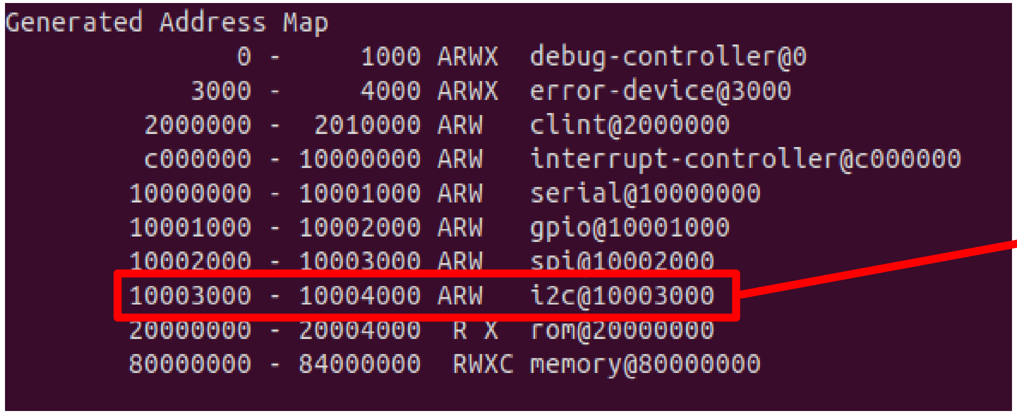
For example: remove I2C from the system.



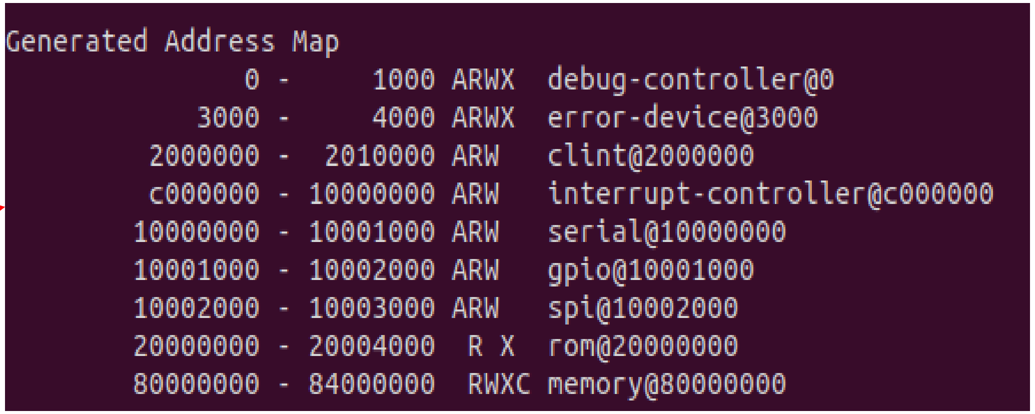
```
class RVCPeripheralsConfig(gpio: Int = 14) extends Config((site, here, up) => {
  case sifive.blocks.devices.uart.PeripheryUARTKey => Seq(
    sifive.blocks.devices.uart.UARTParams(0x10000000))
  case sifive.blocks.devices.gpio.PeripheryGPIOKey => Seq(
    sifive.blocks.devices.gpio.GPIOParams(0x10001000, gpio))
  case sifive.blocks.devices.spi.PeripherySPIKey => Seq(
    sifive.blocks.devices.spi.SPIParams(0x10002000))
  case sifive.blocks.devices.i2c.PeripheryI2CKey => Seq(
    sifive.blocks.devices.i2c.I2CParams(0x10003000))
  //case sifive.blocks.devices.spi.PeripherySPIFlashKey => Seq(
  //  sifive.blocks.devices.spi.SPIFlashParams(0x10003000, 0x20000000L))
  case MaskROMLocated(InSubsystem) => Seq(
    freechips.rocketchip.devices.tilelink.MaskROMParams(0x20000000L, "MyBootROM", 4096))
  case SDRAMKey => Seq()
  case SRAMKey => Seq()
  //case freechips.rocketchip.subsystem.PeripheryMaskROMKey => Seq()
  case SubsystemDriveAsyncClockGroupsKey => None
})
```



```
class RVCPeripheralsConfig(gpio: Int = 14) extends Config((site, here, up) => {
  case sifive.blocks.devices.uart.PeripheryUARTKey => Seq(
    sifive.blocks.devices.uart.UARTParams(0x10000000))
  case sifive.blocks.devices.gpio.PeripheryGPIOKey => Seq(
    sifive.blocks.devices.gpio.GPIOParams(0x10001000, gpio))
  case sifive.blocks.devices.spi.PeripherySPIKey => Seq(
    sifive.blocks.devices.spi.SPIParams(0x10002000))
  case sifive.blocks.devices.i2c.PeripheryI2CKey => Seq()
  //case sifive.blocks.devices.spi.PeripherySPIFlashKey => Seq(
  //  sifive.blocks.devices.spi.SPIFlashParams(0x10003000, 0x20000000L))
  case MaskROMLocated(InSubsystem) => Seq(
    freechips.rocketchip.devices.tilelink.MaskROMParams(0x20000000L, "MyBootROM", 4096))
  case SDRAMKey => Seq()
  case SRAMKey => Seq()
  //case freechips.rocketchip.subsystem.PeripheryMaskROMKey => Seq()
  case SubsystemDriveAsyncClockGroupsKey => None
})
```



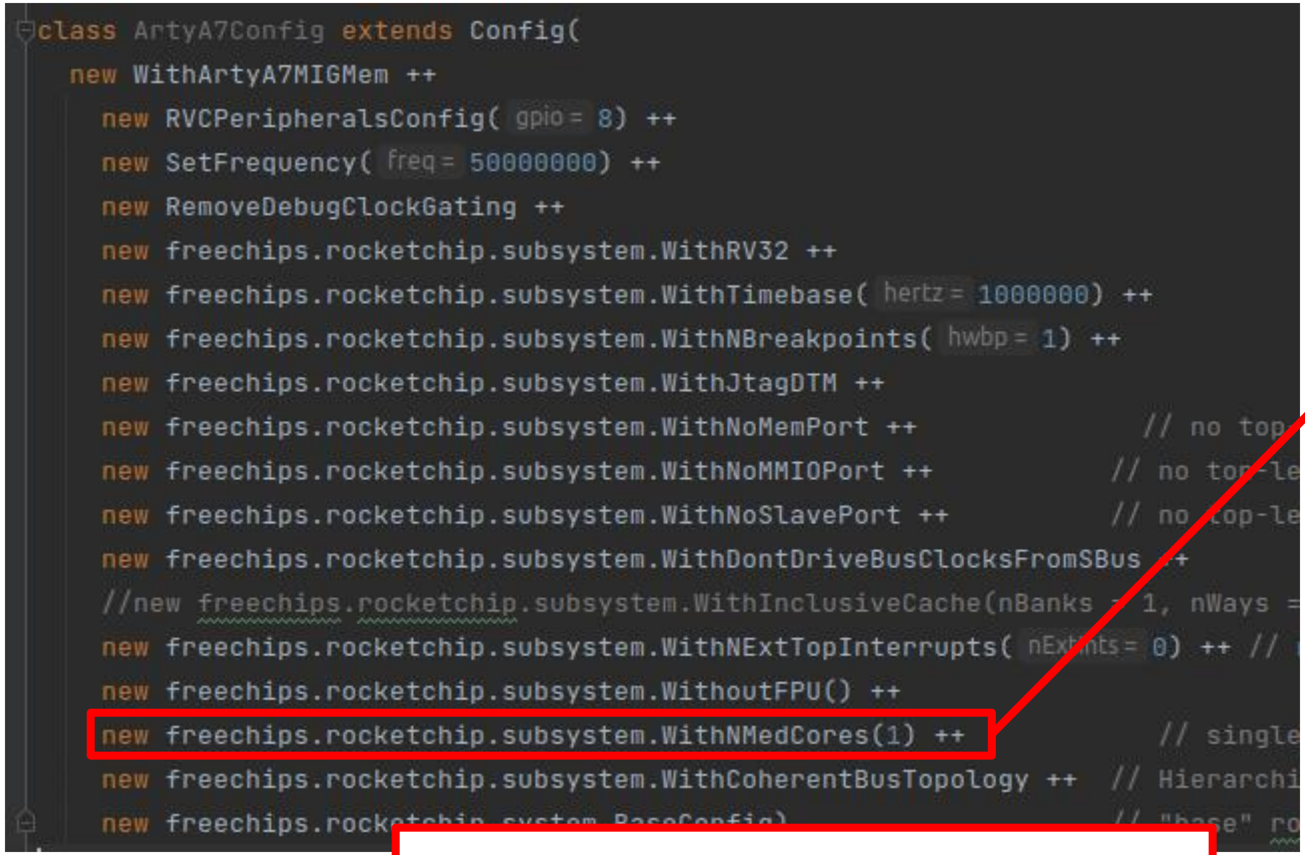
```
Generated Address Map
               0 -        1000 ARWX  debug-controller@0
            3000 -        4000 ARWX  error-device@3000
         2000000 -     2010000 ARW   clint@2000000
         c000000 -    10000000 ARW   interrupt-controller@c000000
        10000000 -    10001000 ARW   serial@10000000
        10001000 -    10002000 ARW   gpio@10001000
        10002000 -    10003000 ARW   spi@10002000
        10003000 -    10004000 ARW   i2c@10003000
        20000000 -    20004000  R X  rom@20000000
        80000000 -    84000000  RWXC memory@80000000
```



```
Generated Address Map
               0 -        1000 ARWX  debug-controller@0
            3000 -        4000 ARWX  error-device@3000
         2000000 -     2010000 ARW   clint@2000000
         c000000 -    10000000 ARW   interrupt-controller@c000000
        10000000 -    10001000 ARW   serial@10000000
        10001000 -    10002000 ARW   gpio@10001000
        10002000 -    10003000 ARW   spi@10002000
        20000000 -    20004000  R X  rom@20000000
        80000000 -    84000000  RWXC memory@80000000
```

7. Modifying system by Scala config (3/10) Processor

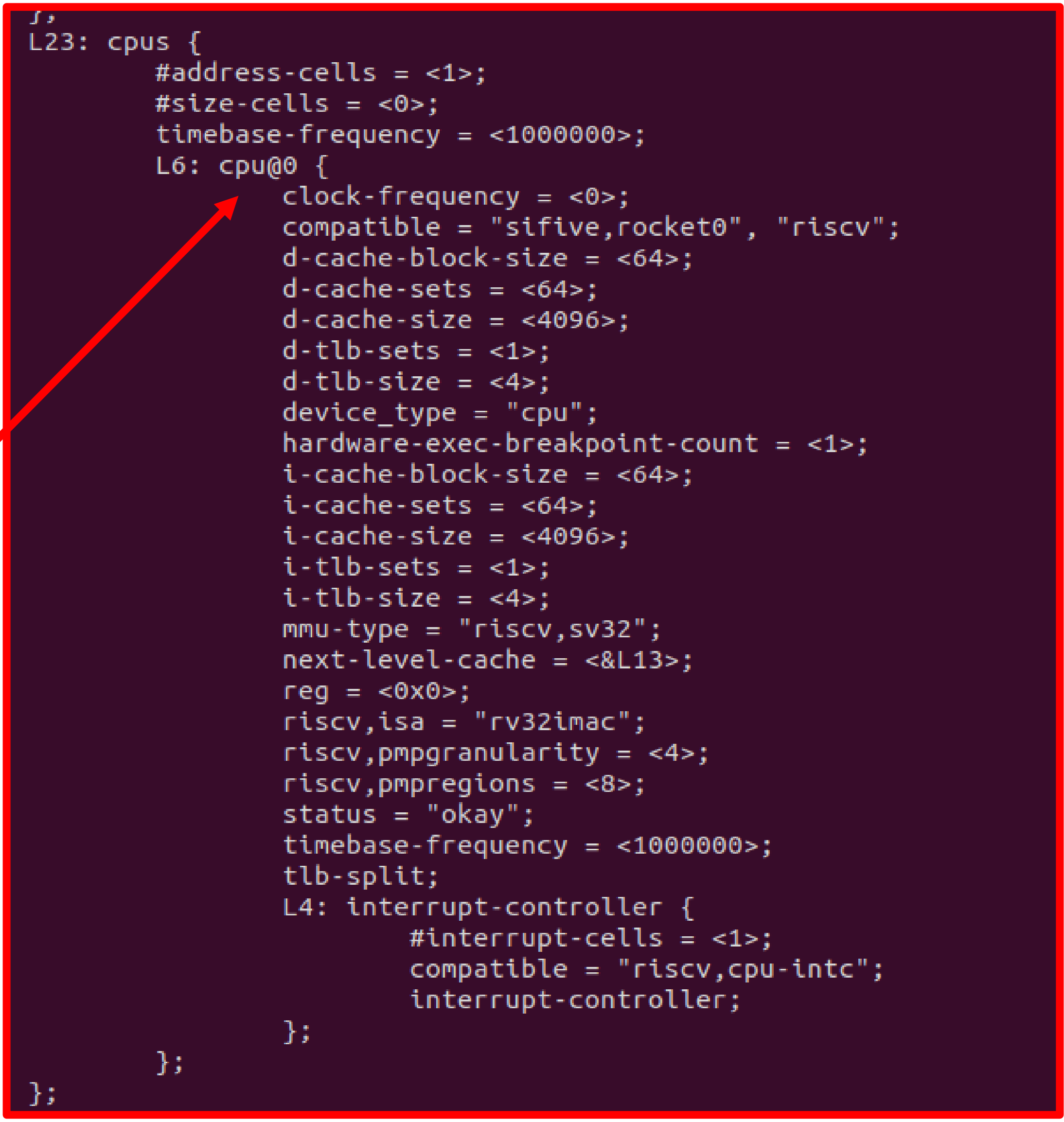
At `RISCVConsole/hardware/riscvconsole/src/main/scala/riscvconsole/RVCConfig.scala`:



```
class ArtyA7Config extends Config(
  new WithArtyA7MIGMem ++
    new RVCPeripheralsConfig( gpio = 8) ++
    new SetFrequency( freq = 50000000) ++
    new RemoveDebugClockGating ++
    new freechips.rocketchip.subsystem.WithRV32 ++
    new freechips.rocketchip.subsystem.WithTimebase( hertz = 1000000) ++
    new freechips.rocketchip.subsystem.WithNBreakpoints( hwbp = 1) ++
    new freechips.rocketchip.subsystem.WithJtagDTM ++
    new freechips.rocketchip.subsystem.WithNoMemPort ++                 // no top
    new freechips.rocketchip.subsystem.WithNoMMIOPort ++               // no top-le
    new freechips.rocketchip.subsystem.WithNoSlavePort ++              // no top-le
    new freechips.rocketchip.subsystem.WithDontDriveBusClocksFromSBus ++
    //new freechips.rocketchip.subsystem.WithInclusiveCache(nBanks = 1, nWays =
    new freechips.rocketchip.subsystem.WithNExtTopInterrupts( nExtInts = 0) ++ //
    new freechips.rocketchip.subsystem.WithoutFPU() ++
    new freechips.rocketchip.subsystem.WithNMedCores(1) ++             // single
    new freechips.rocketchip.subsystem.WithCoherentBusTopology ++  // Hierarchi
    new freechips.rocketchip.system.BaseConfig)                    // "base" ro
```

`WithNMedCores(1) ++`

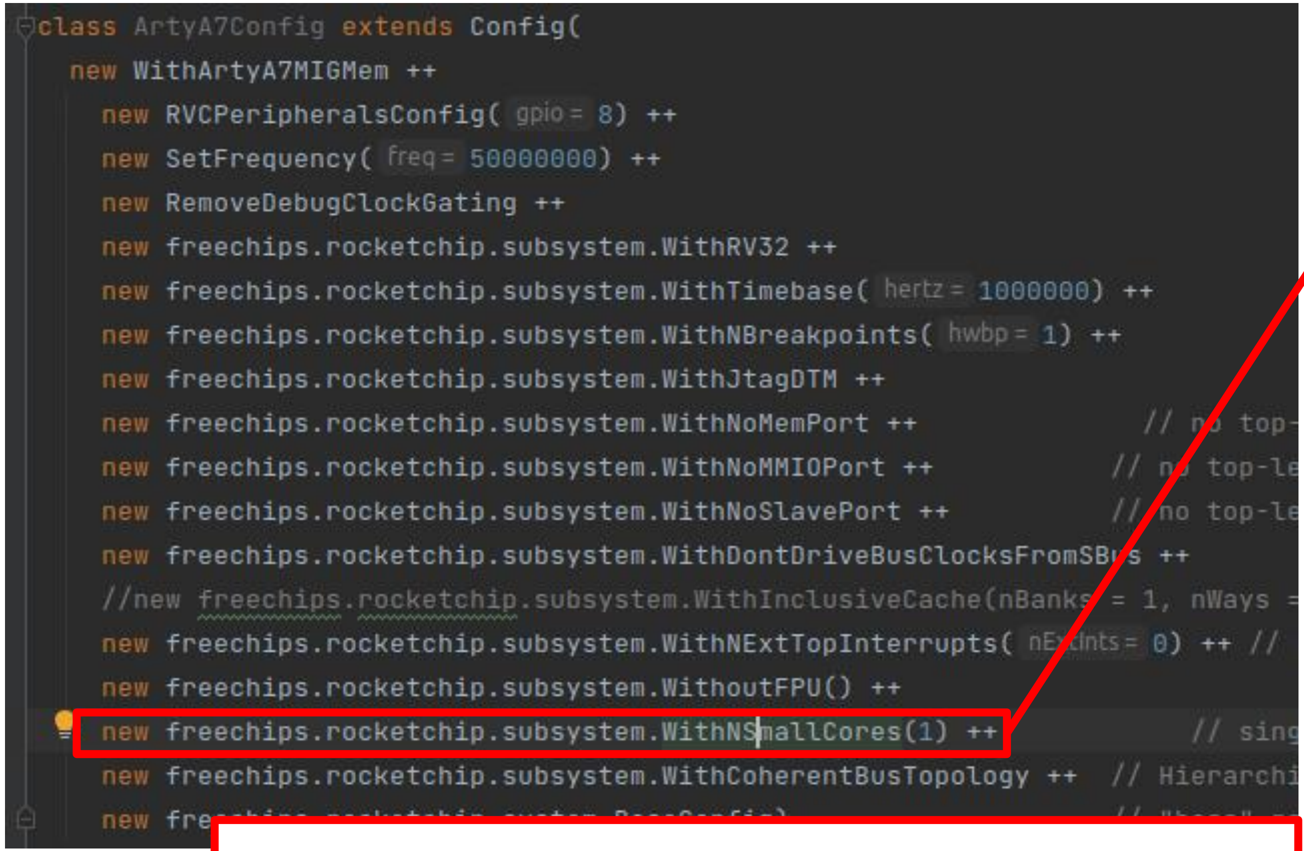
- Supports VM
- Suitable for Linux booting



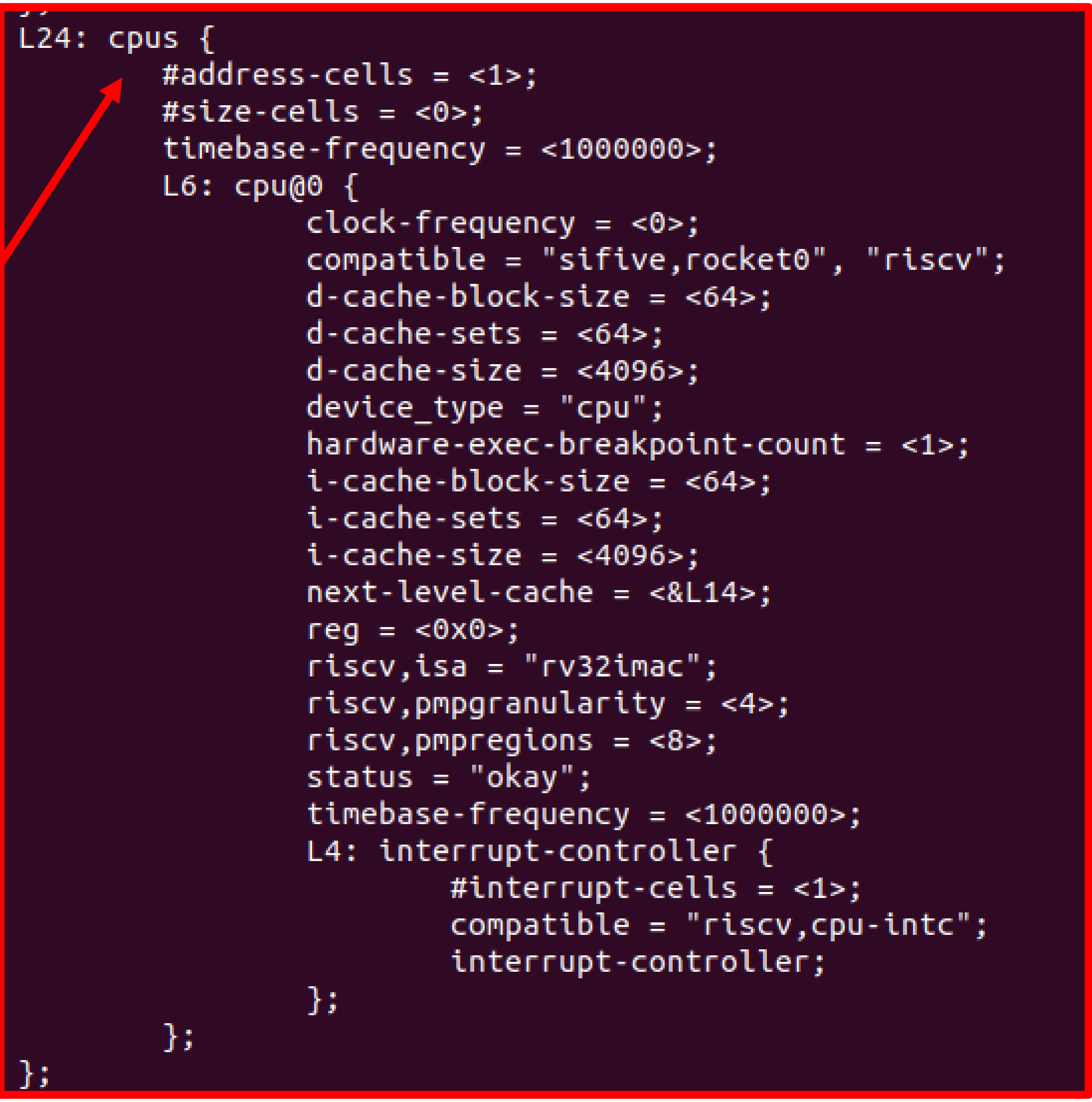
```
};
L23: cpus {
        #address-cells = <1>;
        #size-cells = <0>;
        timebase-frequency = <1000000>;
        L6: cpu@0 {
                clock-frequency = <0>;
                compatible = "sifive,rocket0", "riscv";
                d-cache-block-size = <64>;
                d-cache-sets = <64>;
                d-cache-size = <4096>;
                d-tlb-sets = <1>;
                d-tlb-size = <4>;
                device_type = "cpu";
                hardware-exec-breakpoint-count = <1>;
                i-cache-block-size = <64>;
                i-cache-sets = <64>;
                i-cache-size = <4096>;
                i-tlb-sets = <1>;
                i-tlb-size = <4>;
                mmu-type = "riscv,sv32";
                next-level-cache = <&L13>;
                reg = <0x0>;
                riscv,isa = "rv32imac";
                riscv,pmpgranularity = <4>;
                riscv,pmpregions = <8>;
                status = "okay";
                timebase-frequency = <1000000>;
                tlb-split;
                L4: interrupt-controller {
                        #interrupt-cells = <1>;
                        compatible = "riscv,cpu-intc";
                        interrupt-controller;
                };
        };
};
```

7. Modifying system by Scala config (4/10) Processor

At `RISCVConsole/hardware/riscvconsole/src/main/scala/riscvconsole/RVCConfig.scala`:



```
class ArtyA7Config extends Config(
  new WithArtyA7MIGMem ++
    new RVCPeripheralsConfig( gpio = 8) ++
    new SetFrequency( freq = 50000000) ++
    new RemoveDebugClockGating ++
    new freechips.rocketchip.subsystem.WithRV32 ++
    new freechips.rocketchip.subsystem.WithTimebase( hertz = 1000000) ++
    new freechips.rocketchip.subsystem.WithNBreakpoints( hwbp = 1) ++
    new freechips.rocketchip.subsystem.WithJtagDTM ++
    new freechips.rocketchip.subsystem.WithNoMemPort ++            // no top-
    new freechips.rocketchip.subsystem.WithNoMMIOPort ++          // no top-le
    new freechips.rocketchip.subsystem.WithNoSlavePort ++         // no top-le
    new freechips.rocketchip.subsystem.WithDontDriveBusClocksFromSBus ++
    //new freechips.rocketchip.subsystem.WithInclusiveCache(nBanks = 1, nWays =
    new freechips.rocketchip.subsystem.WithNExtTopInterrupts( nExtInts = 0) ++ //
    new freechips.rocketchip.subsystem.WithoutFPU() ++
    new freechips.rocketchip.subsystem.WithNSmallCores(1) ++           // sing
    new freechips.rocketchip.subsystem.WithCoherentBusTopology ++  // Hierarchi
    new fre
```



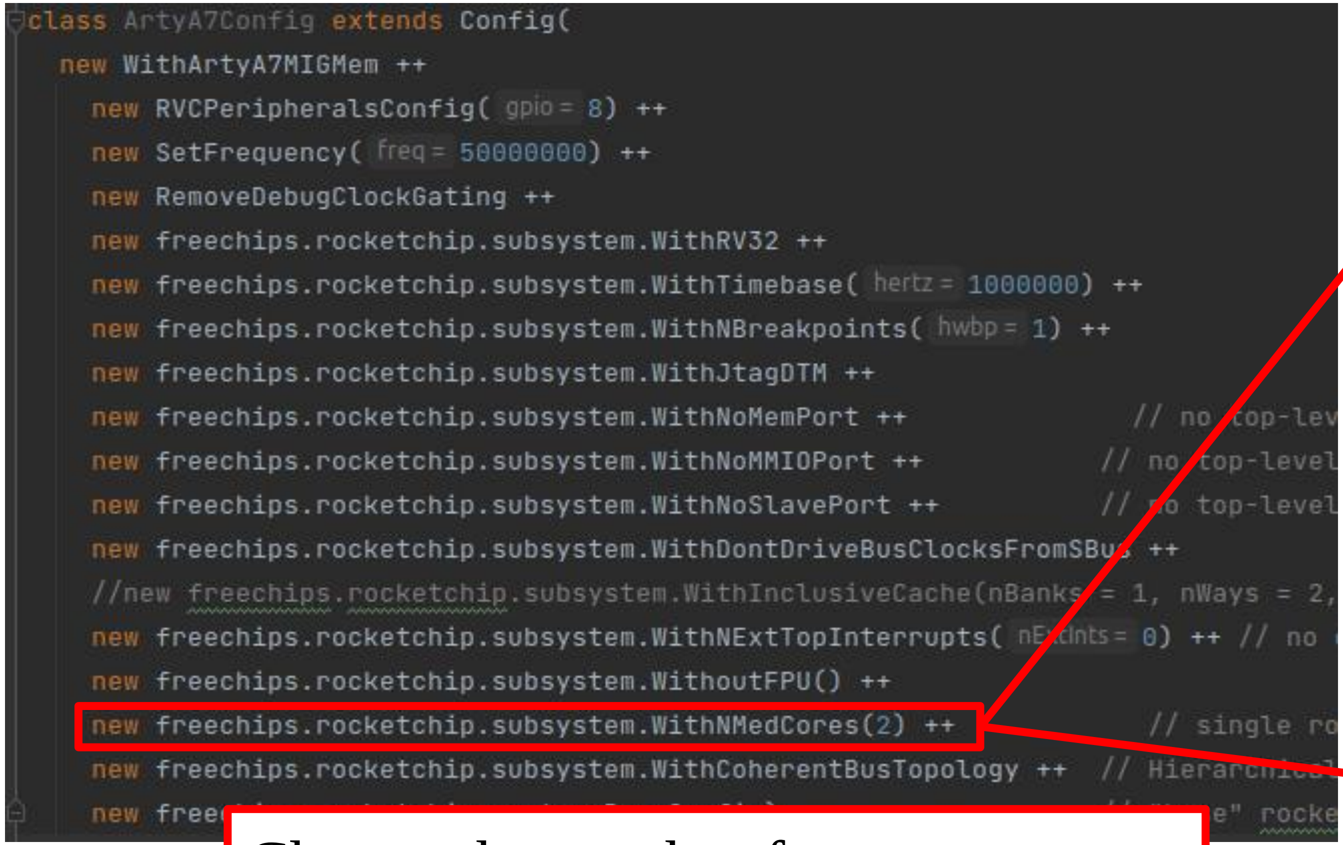
```
L24: cpus {
        #address-cells = <1>;
        #size-cells = <0>;
        timebase-frequency = <1000000>;
        L6: cpu@0 {
                clock-frequency = <0>;
                compatible = "sifive,rocket0", "riscv";
                d-cache-block-size = <64>;
                d-cache-sets = <64>;
                d-cache-size = <4096>;
                device_type = "cpu";
                hardware-exec-breakpoint-count = <1>;
                i-cache-block-size = <64>;
                i-cache-sets = <64>;
                i-cache-size = <4096>;
                next-level-cache = <&L14>;
                reg = <0x0>;
                riscv,isa = "rv32imac";
                riscv,pmpgranularity = <4>;
                riscv,pmpregions = <8>;
                status = "okay";
                timebase-frequency = <1000000>;
                L4: interrupt-controller {
                        #interrupt-cells = <1>;
                        compatible = "riscv,cpu-intc";
                        interrupt-controller;
                };
        };
};
```

Change to: `WithNSmallCores (1) ++`

- Does not support VM
- Usually for Micontrollers

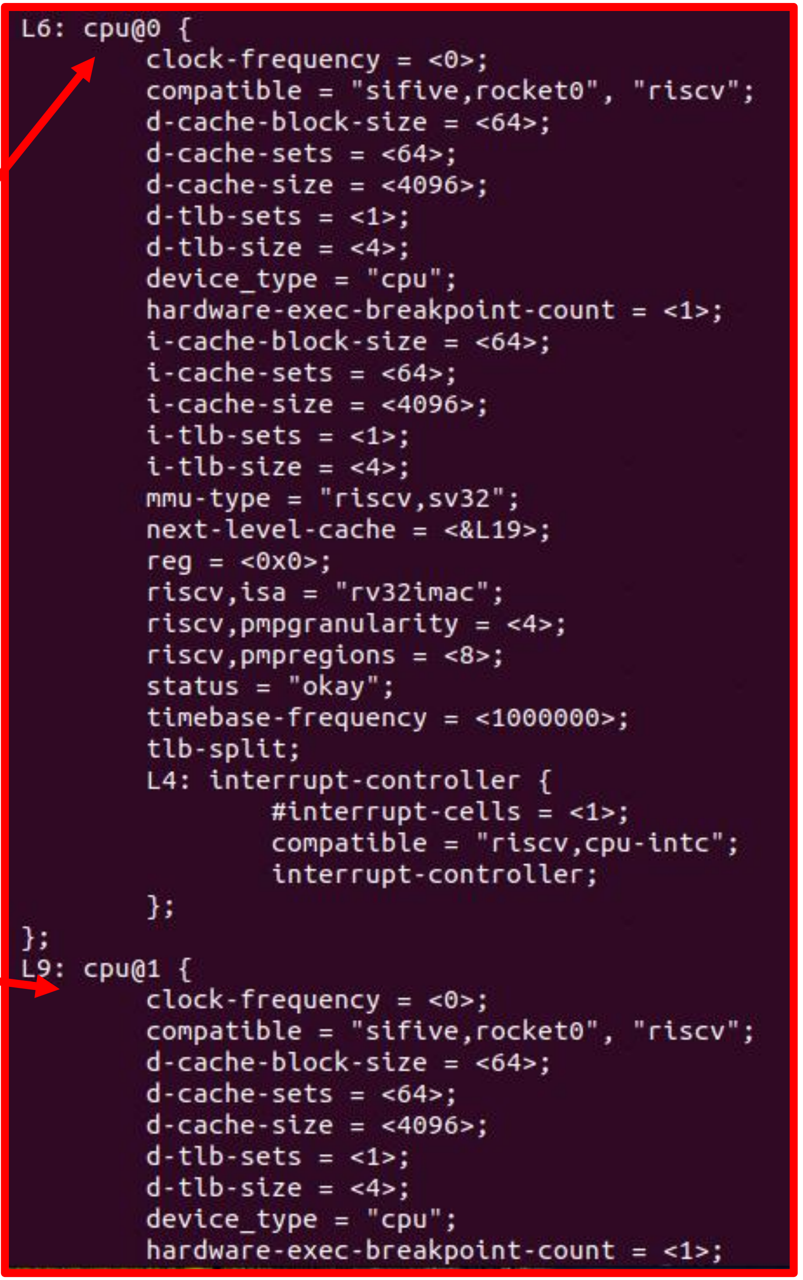
7. Modifying system by Scala config (5/10) Processor

At `RISCVConsole/hardware/riscvconsole/src/main/scala/riscvconsole/RVCConfig.scala`:



```
class ArtyA7Config extends Config(
  new WithArtyA7MIGMem ++
    new RVCPeripheralsConfig( gpio = 8) ++
    new SetFrequency( freq = 50000000) ++
    new RemoveDebugClockGating ++
    new freechips.rocketchip.subsystem.WithRV32 ++
    new freechips.rocketchip.subsystem.WithTimebase( hertz = 1000000) ++
    new freechips.rocketchip.subsystem.WithNBreakpoints( hwbp = 1) ++
    new freechips.rocketchip.subsystem.WithJtagDTM ++
    new freechips.rocketchip.subsystem.WithNoMemPort ++             // no top-lev
    new freechips.rocketchip.subsystem.WithNoMMIOPort ++          // no top-level
    new freechips.rocketchip.subsystem.WithNoSlavePort ++         // no top-level
    new freechips.rocketchip.subsystem.WithDontDriveBusClocksFromSBus ++
    //new freechips.rocketchip.subsystem.WithInclusiveCache(nBanks = 1, nWays = 2,
    new freechips.rocketchip.subsystem.WithNExtTopInterrupts( nExtInts = 0) ++ // no
    new freechips.rocketchip.subsystem.WithoutFPU() ++
    new freechips.rocketchip.subsystem.WithNMedCores(2) ++           // single ro
    new freechips.rocketchip.subsystem.WithCoherentBusTopology ++ // Hierarchical
    new free                                                  e" rocke
```

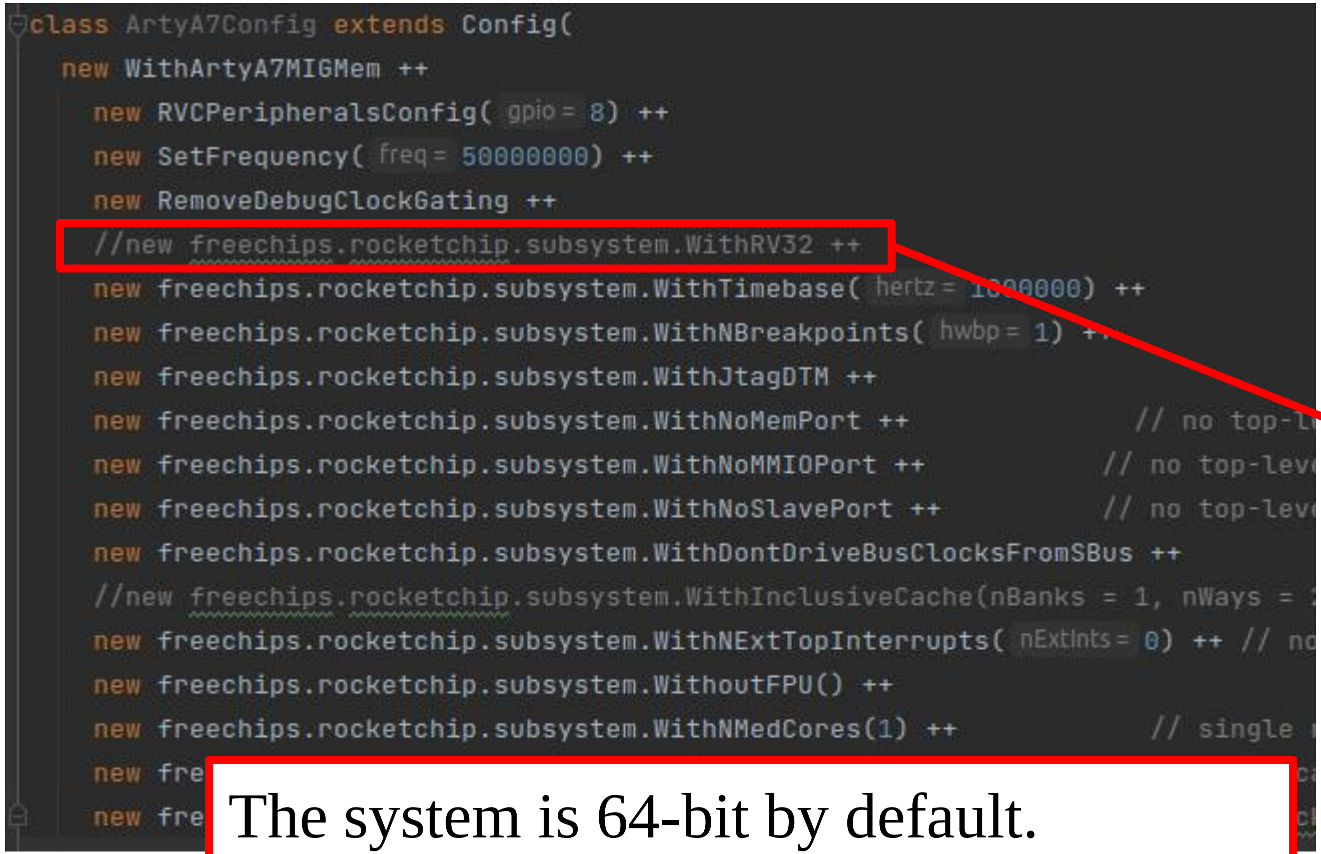
Change the number from `(1)` to `(2)` will increase the number of processors accordingly



```
L6: cpu@0 {
        clock-frequency = <0>;
        compatible = "sifive,rocket0", "riscv";
        d-cache-block-size = <64>;
        d-cache-sets = <64>;
        d-cache-size = <4096>;
        d-tlb-sets = <1>;
        d-tlb-size = <4>;
        device_type = "cpu";
        hardware-exec-breakpoint-count = <1>;
        i-cache-block-size = <64>;
        i-cache-sets = <64>;
        i-cache-size = <4096>;
        i-tlb-sets = <1>;
        i-tlb-size = <4>;
        mmu-type = "riscv,sv32";
        next-level-cache = <&L19>;
        reg = <0x0>;
        riscv,isa = "rv32imac";
        riscv,pmpgranularity = <4>;
        riscv,pmpregions = <8>;
        status = "okay";
        timebase-frequency = <1000000>;
        tlb-split;
        L4: interrupt-controller {
                #interrupt-cells = <1>;
                compatible = "riscv,cpu-intc";
                interrupt-controller;
        };
};
L9: cpu@1 {
        clock-frequency = <0>;
        compatible = "sifive,rocket0", "riscv";
        d-cache-block-size = <64>;
        d-cache-sets = <64>;
        d-cache-size = <4096>;
        d-tlb-sets = <1>;
        d-tlb-size = <4>;
        device_type = "cpu";
        hardware-exec-breakpoint-count = <1>;
```

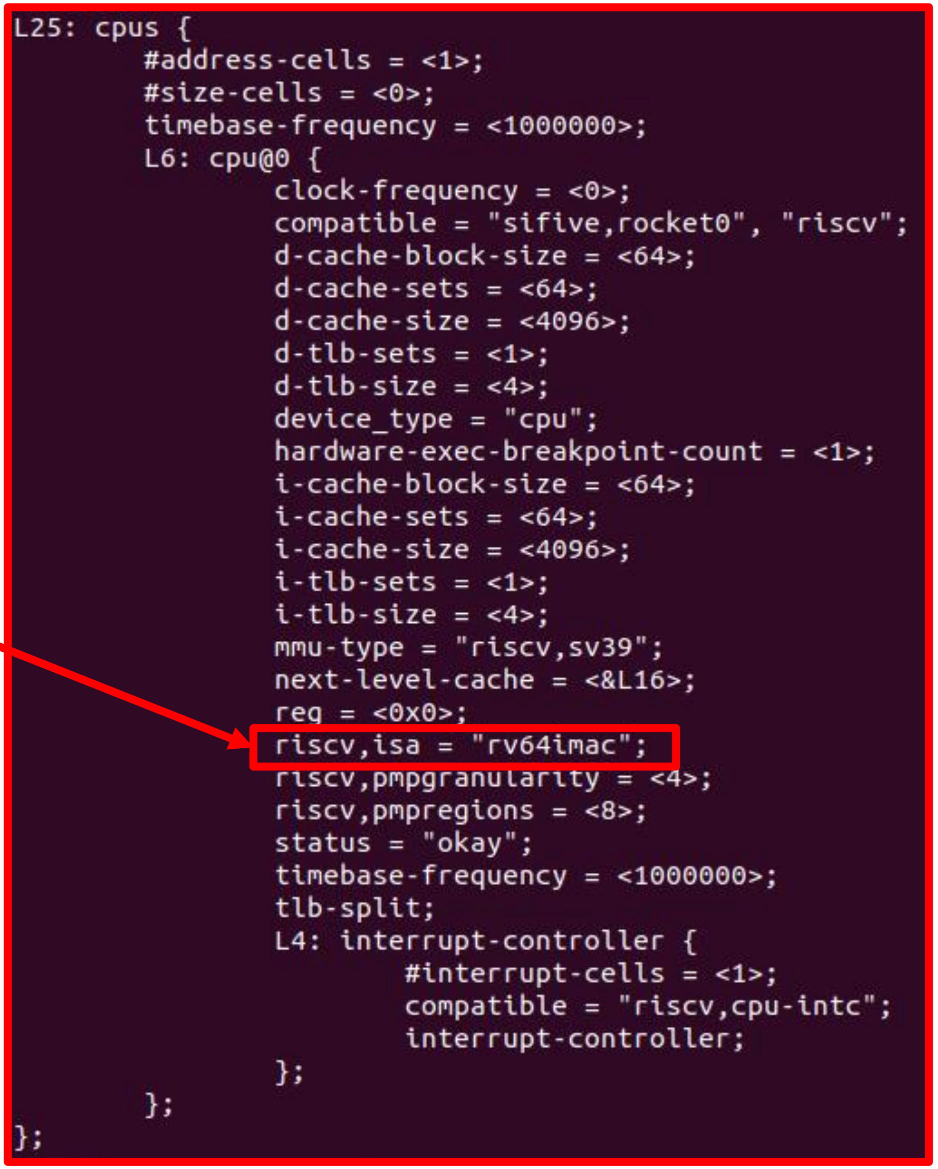
7. Modifying system by Scala config (6/10) Addressing

At `RISCVConsole/hardware/riscvconsole/src/main/scala/riscvconsole/RVCConfig.scala`:



```
class ArtyA7Config extends Config(
  new WithArtyA7MIGMem ++
    new RVCPeripheralsConfig( gpio = 8) ++
    new SetFrequency( freq = 50000000) ++
    new RemoveDebugClockGating ++
    //new freechips.rocketchip.subsystem.WithRV32 ++
    new freechips.rocketchip.subsystem.WithTimebase( hertz = 1000000) ++
    new freechips.rocketchip.subsystem.WithNBreakpoints( hwbp = 1) ++
    new freechips.rocketchip.subsystem.WithJtagDTM ++
    new freechips.rocketchip.subsystem.WithNoMemPort ++              // no top-le
    new freechips.rocketchip.subsystem.WithNoMMIOPort ++           // no top-leve
    new freechips.rocketchip.subsystem.WithNoSlavePort ++          // no top-leve
    new freechips.rocketchip.subsystem.WithDontDriveBusClocksFromSBus ++
    //new freechips.rocketchip.subsystem.WithInclusiveCache(nBanks = 1, nWays = 
    new freechips.rocketchip.subsystem.WithNExtTopInterrupts( nExtInts = 0) ++ // no
    new freechips.rocketchip.subsystem.WithoutFPU() ++
    new freechips.rocketchip.subsystem.WithNMedCores(1) ++            // single 
    new fre
    new fre
```

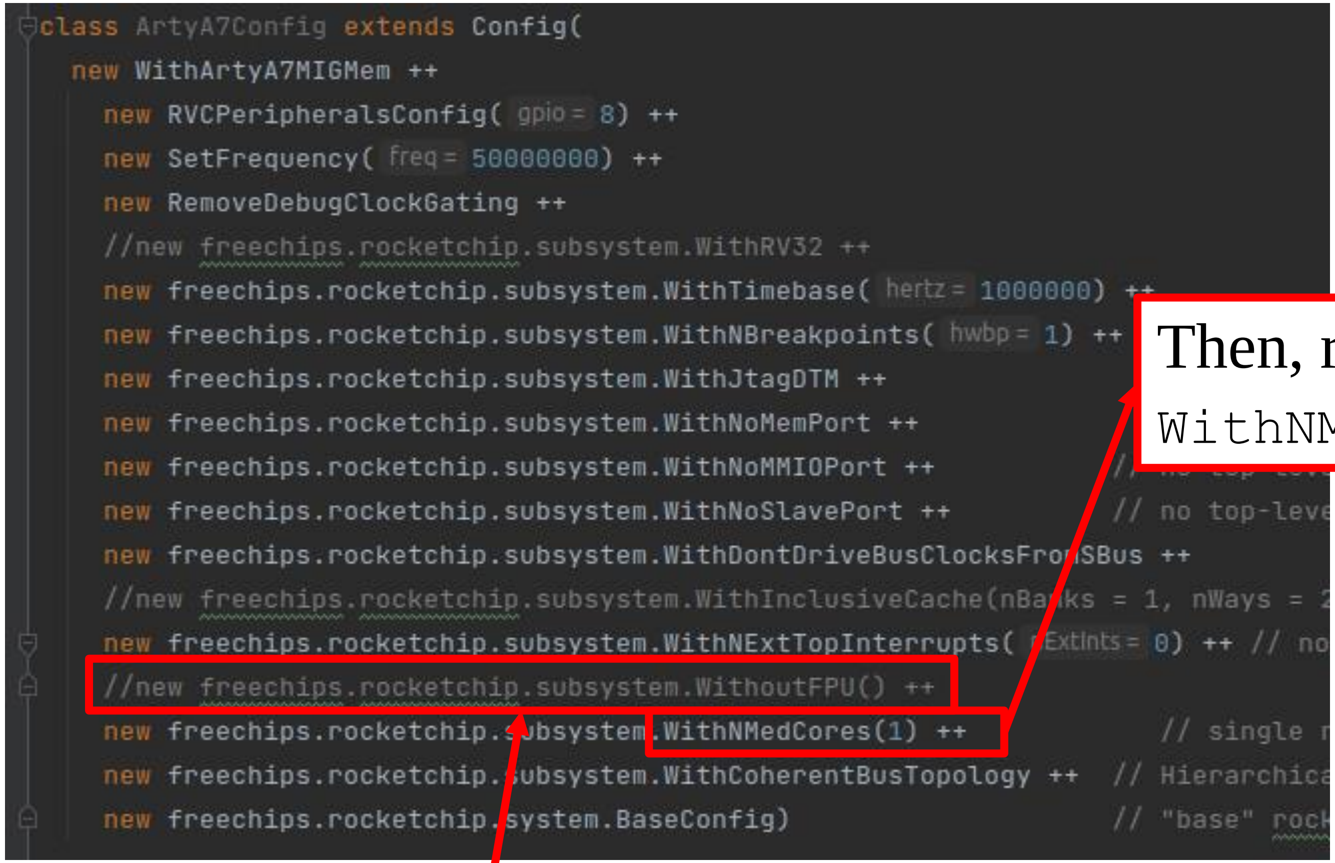
The system is 64-bit by default.
So if you disable the `WithRV32 ++` line,
you'll get the 64-bit system back.



```
L25: cpus {
        #address-cells = <1>;
        #size-cells = <0>;
        timebase-frequency = <1000000>;
        L6: cpu@0 {
                clock-frequency = <0>;
                compatible = "sifive,rocket0", "riscv";
                d-cache-block-size = <64>;
                d-cache-sets = <64>;
                d-cache-size = <4096>;
                d-tlb-sets = <1>;
                d-tlb-size = <4>;
                device_type = "cpu";
                hardware-exec-breakpoint-count = <1>;
                i-cache-block-size = <64>;
                i-cache-sets = <64>;
                i-cache-size = <4096>;
                i-tlb-sets = <1>;
                i-tlb-size = <4>;
                mmu-type = "riscv,sv39";
                next-level-cache = <&L16>;
                reg = <0x0>;
                riscv,isa = "rv64imac";
                riscv,pmpgranularity = <4>;
                riscv,pmpregions = <8>;
                status = "okay";
                timebase-frequency = <1000000>;
                tlb-split;
                L4: interrupt-controller {
                        #interrupt-cells = <1>;
                        compatible = "riscv,cpu-intc";
                        interrupt-controller;
                };
        };
};
```

7. Modifying system by Scala config (7/10) FPU

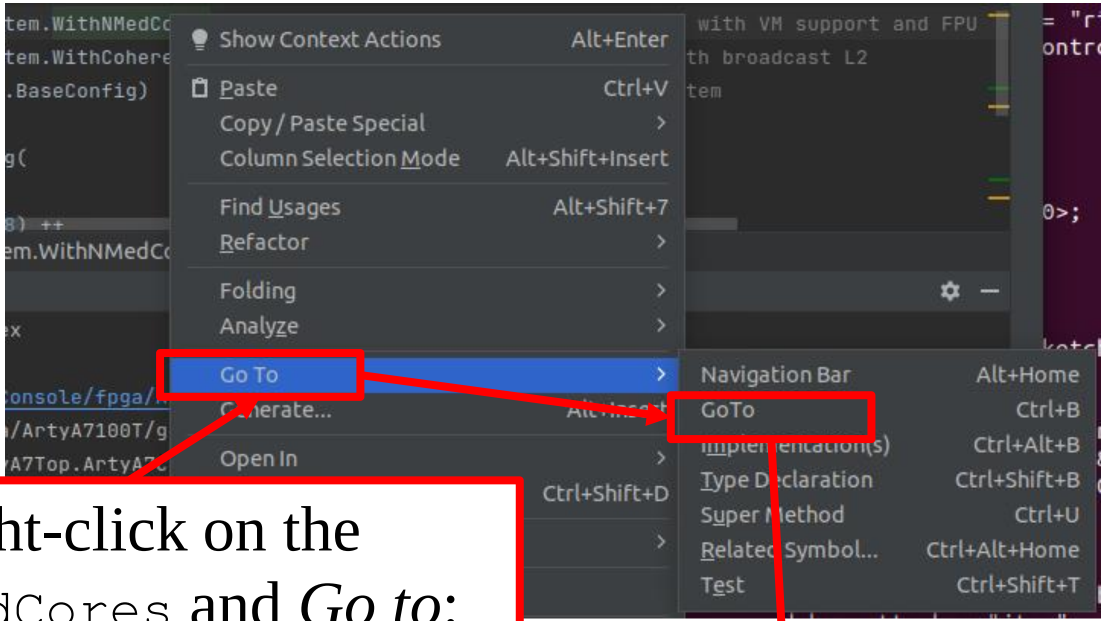
At `RISCVConsole/hardware/riscvconsole/src/main/scala/riscvconsole/RVCConfig.scala`:

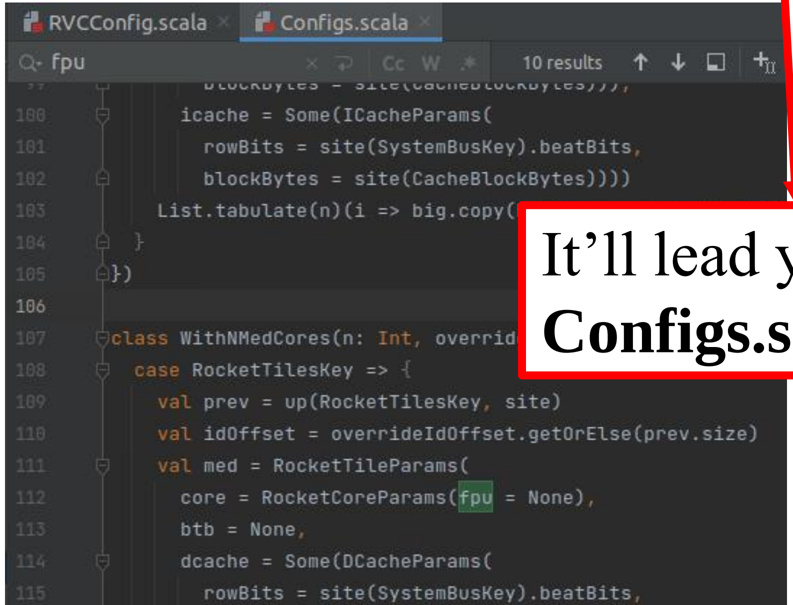


```
class ArtyA7Config extends Config(
  new WithArtyA7MIGMem ++
    new RVCPeripheralsConfig( gpio = 8) ++
    new SetFrequency( freq = 50000000) ++
    new RemoveDebugClockGating ++
    //new freechips.rocketchip.subsystem.WithRV32 ++
    new freechips.rocketchip.subsystem.WithTimebase( hertz = 1000000) ++
    new freechips.rocketchip.subsystem.WithNBreakpoints( hwbp = 1) ++
    new freechips.rocketchip.subsystem.WithJtagDTM ++
    new freechips.rocketchip.subsystem.WithNoMemPort ++
    new freechips.rocketchip.subsystem.WithNoMMIOPort ++
    new freechips.rocketchip.subsystem.WithNoSlavePort ++          // no top-leve
    new freechips.rocketchip.subsystem.WithDontDriveBusClocksFromSBus ++
    //new freechips.rocketchip.subsystem.WithInclusiveCache(nBanks = 1, nWays = 2
    new freechips.rocketchip.subsystem.WithNExtTopInterrupts( nExtInts = 0) ++ // no
    //new freechips.rocketchip.subsystem.WithoutFPU() ++
    new freechips.rocketchip.subsystem.WithNMedCores(1) ++           // single r
    new freechips.rocketchip.subsystem.WithCoherentBusTopology ++  // Hierarchica
    new freechips.rocketchip.system.BaseConfig)                    // "base" rock
```

To bring back the FPU, first, disable the `WithoutFPU() ++` line.

Then, right-click on the `WithNMedCores` and *Go to*:

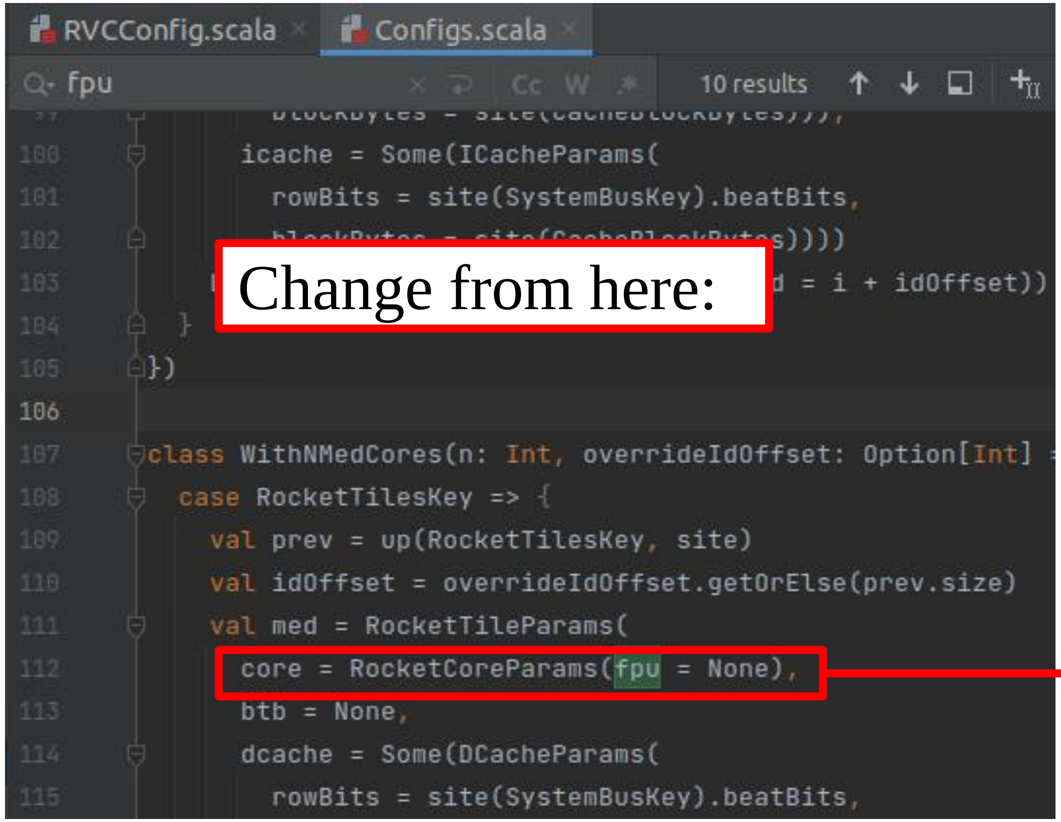




```
          blockBytes = site(CacheBlockBytes))),
        icache = Some(ICacheParams(
          rowBits = site(SystemBusKey).beatBits,
          blockBytes = site(CacheBlockBytes))))
      List.tabulate(n)(i => big.copy(
    }
})

class WithNMedCores(n: Int, overrid
  case RocketTilesKey => {
    val prev = up(RocketTilesKey, site)
    val idOffset = overrideIdOffset.getOrElse(prev.size)
    val med = RocketTileParams(
      core = RocketCoreParams(fpu = None),
      btb = None,
      dcache = Some(DCacheParams(
        rowBits = site(SystemBusKey).beatBits,
```

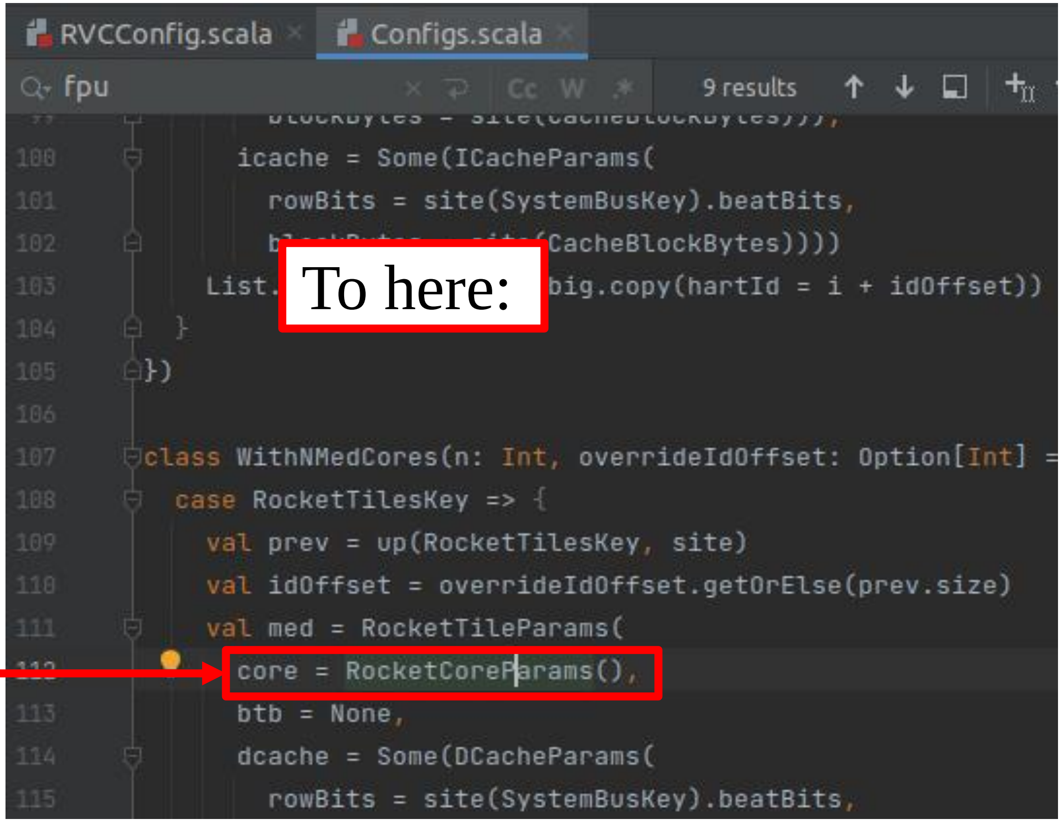
It'll lead you to the **Configs.scala** file:

7. Modifying system by Scala config (8/10) FPU



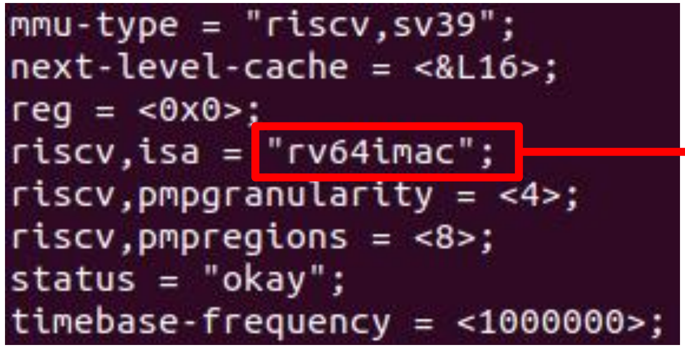
Change from here:

```
        blockBytes = site(CacheBlockBytes))),
      icache = Some(ICacheParams(
        rowBits = site(SystemBusKey).beatBits,
        blockBytes = site(CacheBlockBytes))))
    L                          d = i + idOffset))
  }
})

class WithNMedCores(n: Int, overrideIdOffset: Option[Int] =
  case RocketTilesKey => {
    val prev = up(RocketTilesKey, site)
    val idOffset = overrideIdOffset.getOrElse(prev.size)
    val med = RocketTileParams(
      core = RocketCoreParams(fpu = None),
      btb = None,
      dcache = Some(DCacheParams(
        rowBits = site(SystemBusKey).beatBits,
```

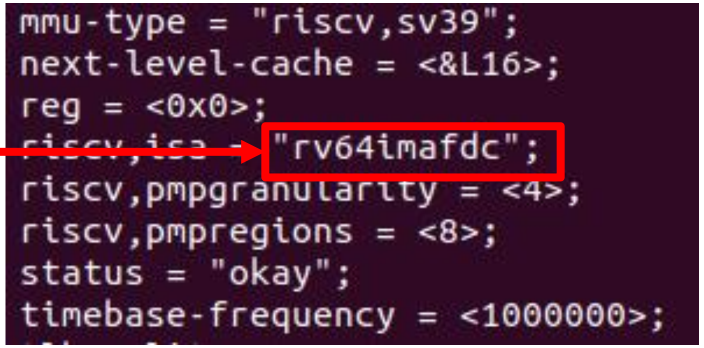


To here:

```
        blockBytes = site(CacheBlockBytes))),
      icache = Some(ICacheParams(
        rowBits = site(SystemBusKey).beatBits,
        blockBytes = site(CacheBlockBytes))))
    List.              big.copy(hartId = i + idOffset))
  }
})

class WithNMedCores(n: Int, overrideIdOffset: Option[Int] =
  case RocketTilesKey => {
    val prev = up(RocketTilesKey, site)
    val idOffset = overrideIdOffset.getOrElse(prev.size)
    val med = RocketTileParams(
      core = RocketCoreParams(),
      btb = None,
      dcache = Some(DCacheParams(
        rowBits = site(SystemBusKey).beatBits,
```



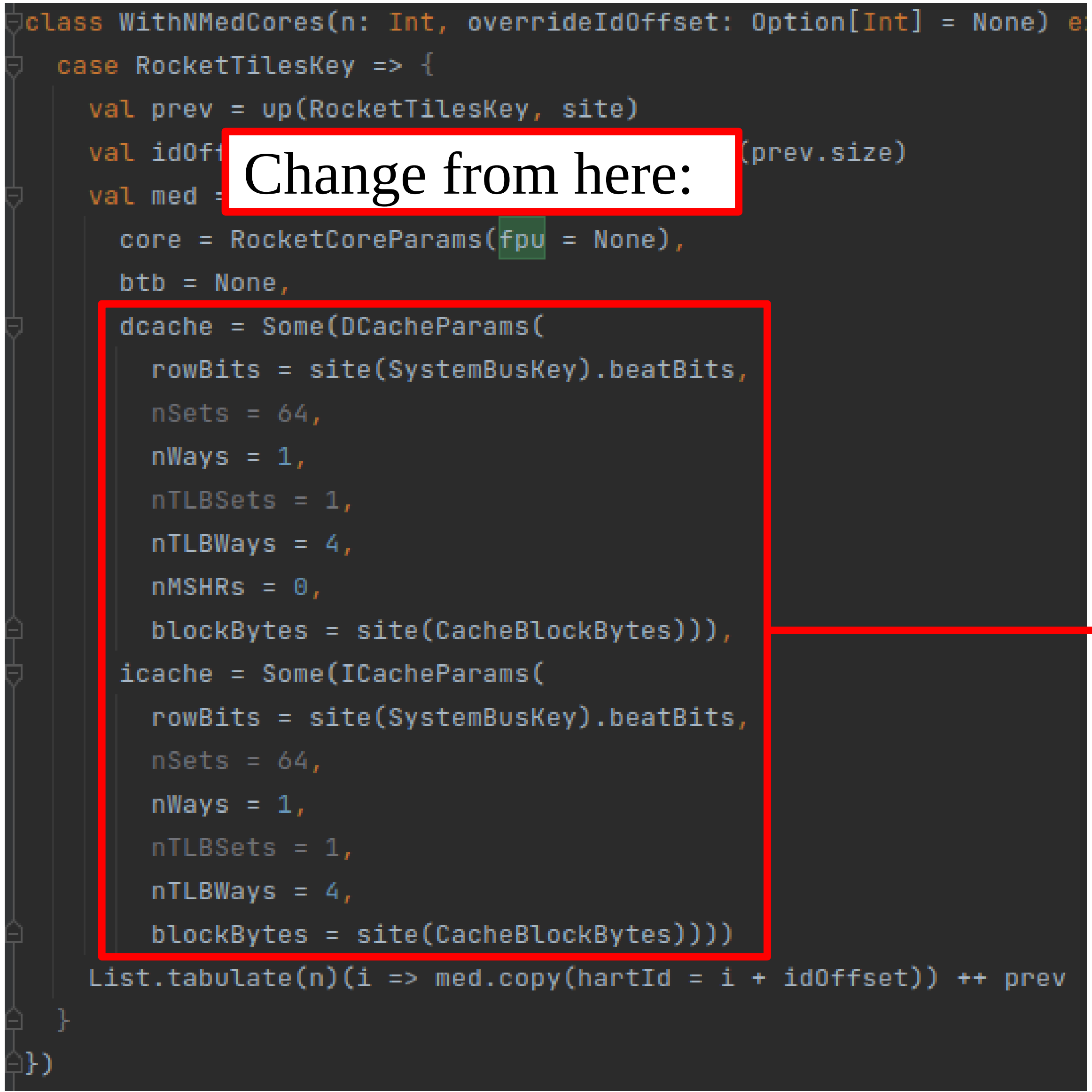
```
mmu-type = "riscv,sv39";
next-level-cache = <&L16>;
reg = <0x0>;
riscv,isa = "rv64imac";
riscv,pmpgranularity = <4>;
riscv,pmpregions = <8>;
status = "okay";
timebase-frequency = <1000000>;
```



```
mmu-type = "riscv,sv39";
next-level-cache = <&L16>;
reg = <0x0>;
riscv,isa = "rv64imafdc";
riscv,pmpgranularity = <4>;
riscv,pmpregions = <8>;
status = "okay";
timebase-frequency = <1000000>;
```

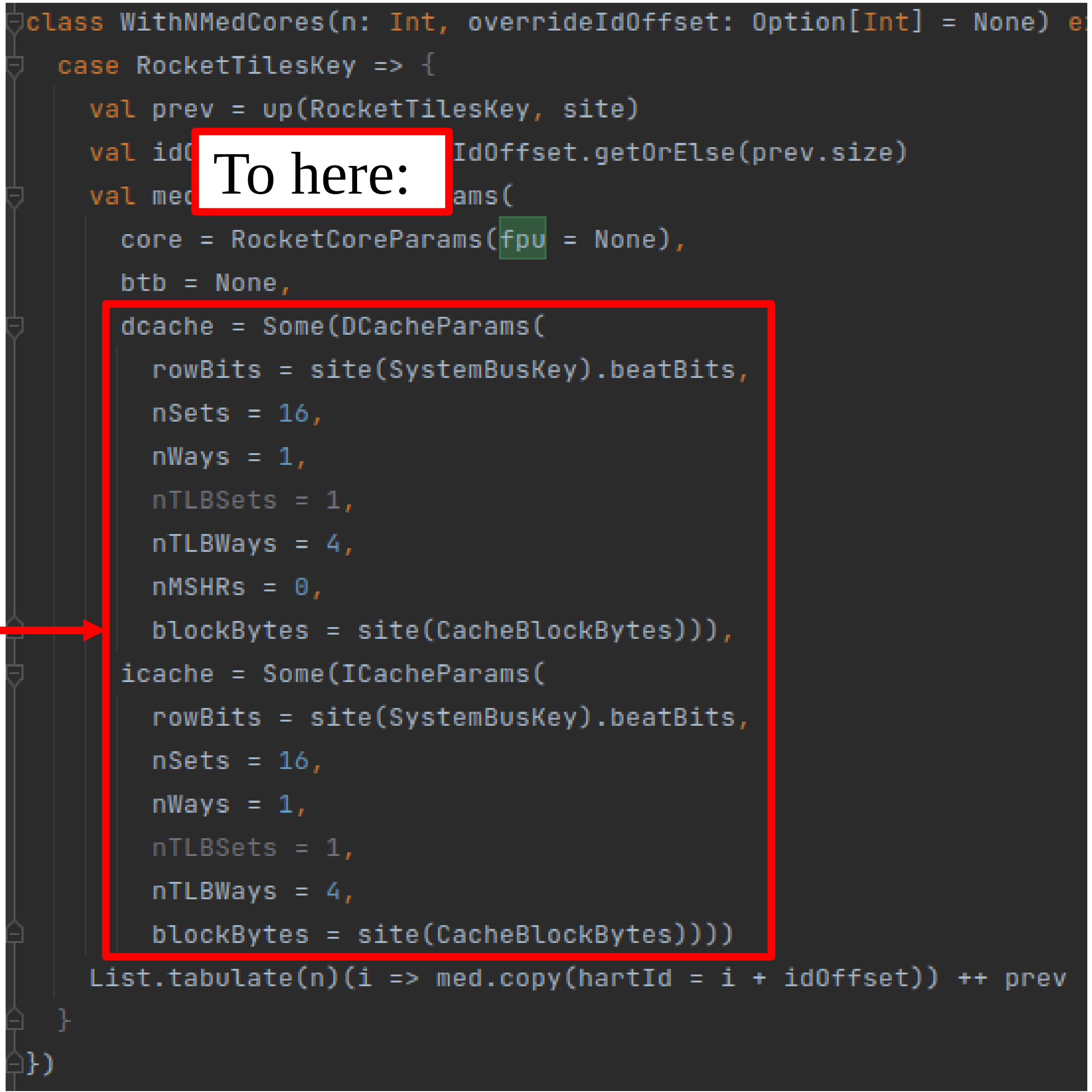
7. Modifying system by Scala config (9/10) L1 caches

To reduce the L1 caches, for example, from the **Configs.scala** file:



Change from here:

```
class WithNMedCores(n: Int, overrideIdOffset: Option[Int] = None) e
  case RocketTilesKey => {
    val prev = up(RocketTilesKey, site)
    val idOff                                  (prev.size)
    val med =
      core = RocketCoreParams(fpu = None),
      btb = None,
      dcache = Some(DCacheParams(
        rowBits = site(SystemBusKey).beatBits,
        nSets = 64,
        nWays = 1,
        nTLBSets = 1,
        nTLBWays = 4,
        nMSHRs = 0,
        blockBytes = site(CacheBlockBytes))),
      icache = Some(ICacheParams(
        rowBits = site(SystemBusKey).beatBits,
        nSets = 64,
        nWays = 1,
        nTLBSets = 1,
        nTLBWays = 4,
        blockBytes = site(CacheBlockBytes)))
    List.tabulate(n)(i => med.copy(hartId = i + idOffset)) ++ prev
  }
})
```

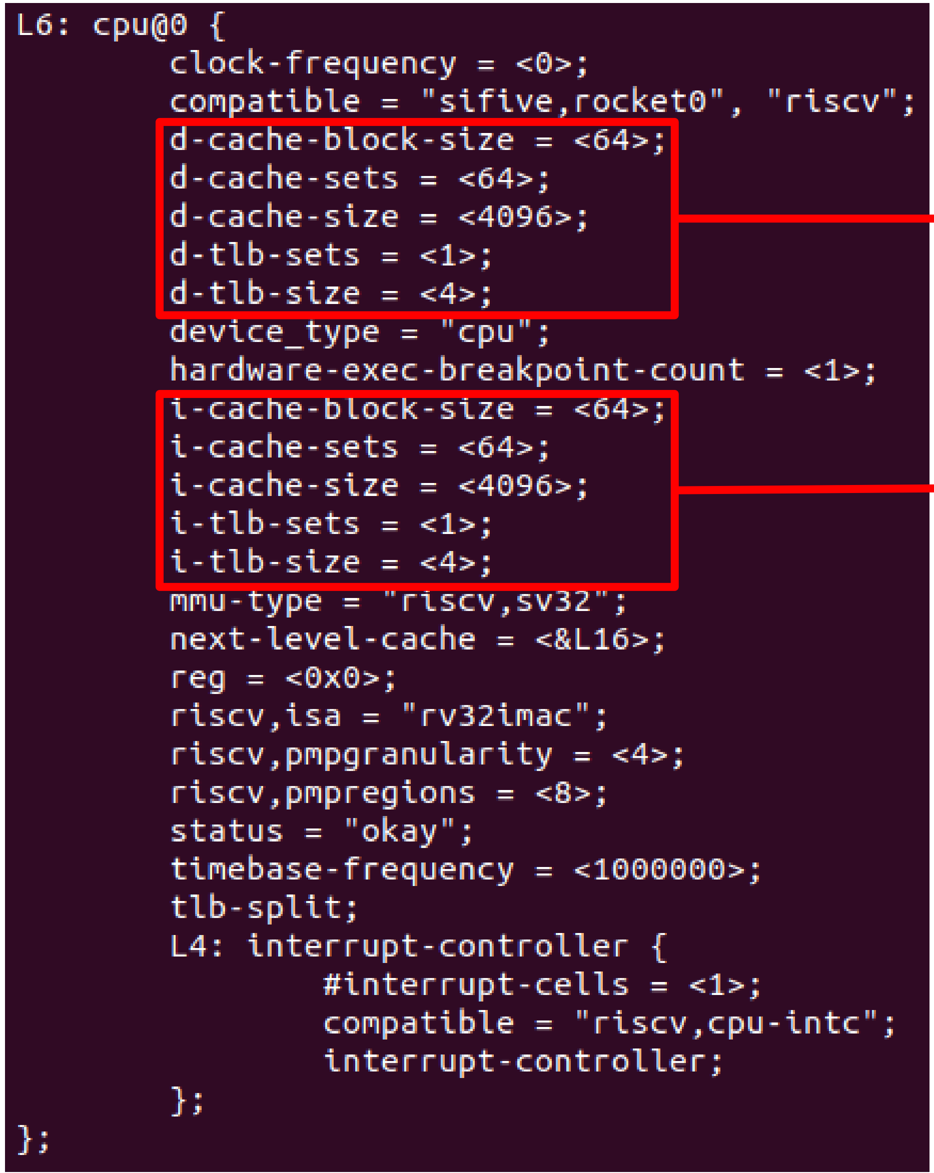


To here:

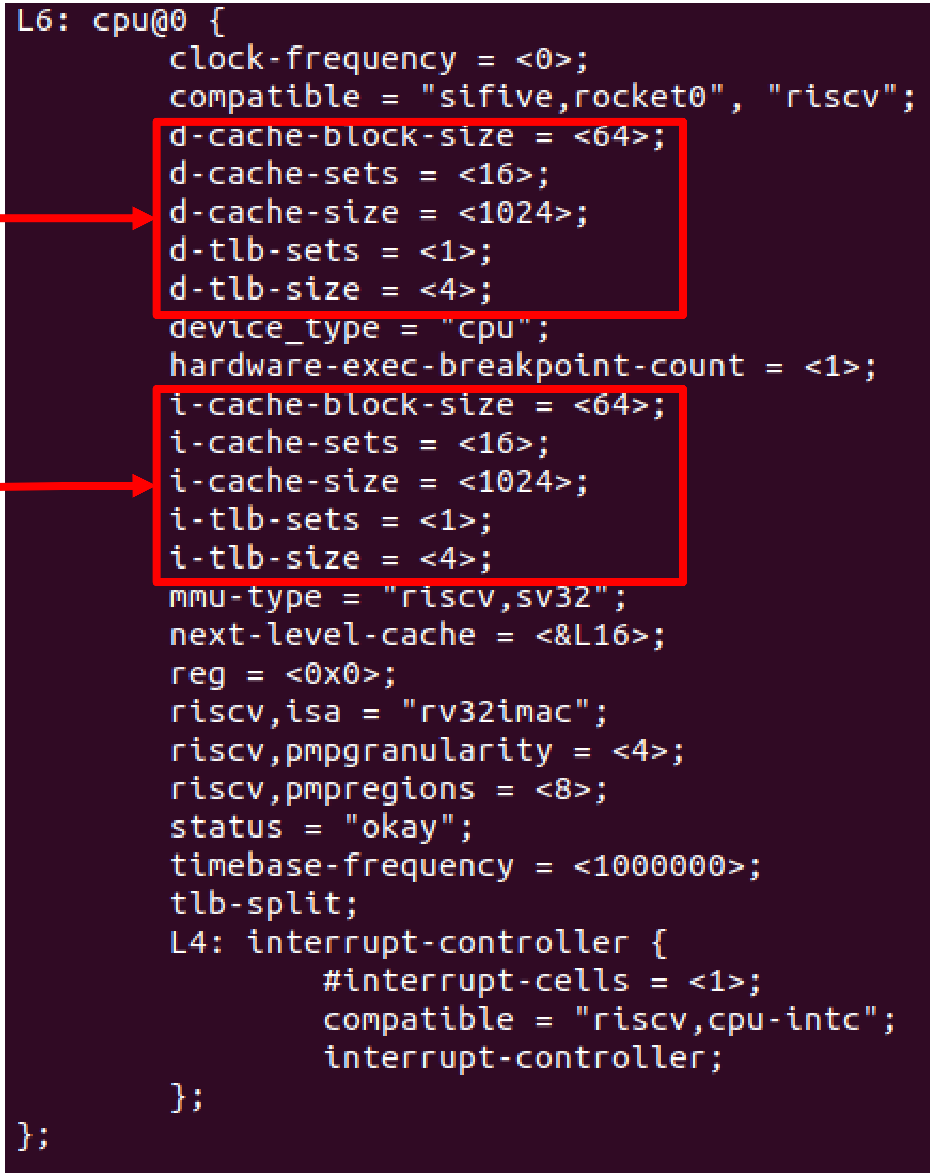
```
class WithNMedCores(n: Int, overrideIdOffset: Option[Int] = None) e
  case RocketTilesKey => {
    val prev = up(RocketTilesKey, site)
    val id                 IdOffset.getOrElse(prev.size)
    val med                ams(
      core = RocketCoreParams(fpu = None),
      btb = None,
      dcache = Some(DCacheParams(
        rowBits = site(SystemBusKey).beatBits,
        nSets = 16,
        nWays = 1,
        nTLBSets = 1,
        nTLBWays = 4,
        nMSHRs = 0,
        blockBytes = site(CacheBlockBytes))),
      icache = Some(ICacheParams(
        rowBits = site(SystemBusKey).beatBits,
        nSets = 16,
        nWays = 1,
        nTLBSets = 1,
        nTLBWays = 4,
        blockBytes = site(CacheBlockBytes)))
    List.tabulate(n)(i => med.copy(hartId = i + idOffset)) ++ prev
  }
})
```

7. Modifying system by Scala config (10/10) L1 caches

The result after that:



```
L6: cpu@0 {
        clock-frequency = <0>;
        compatible = "sifive,rocket0", "riscv";
        d-cache-block-size = <64>;
        d-cache-sets = <64>;
        d-cache-size = <4096>;
        d-tlb-sets = <1>;
        d-tlb-size = <4>;
        device_type = "cpu";
        hardware-exec-breakpoint-count = <1>;
        i-cache-block-size = <64>;
        i-cache-sets = <64>;
        i-cache-size = <4096>;
        i-tlb-sets = <1>;
        i-tlb-size = <4>;
        mmu-type = "riscv,sv32";
        next-level-cache = <&L16>;
        reg = <0x0>;
        riscv,isa = "rv32imac";
        riscv,pmpgranularity = <4>;
        riscv,pmpregions = <8>;
        status = "okay";
        timebase-frequency = <1000000>;
        tlb-split;
        L4: interrupt-controller {
                #interrupt-cells = <1>;
                compatible = "riscv,cpu-intc";
                interrupt-controller;
        };
};
```



```
L6: cpu@0 {
        clock-frequency = <0>;
        compatible = "sifive,rocket0", "riscv";
        d-cache-block-size = <64>;
        d-cache-sets = <16>;
        d-cache-size = <1024>;
        d-tlb-sets = <1>;
        d-tlb-size = <4>;
        device_type = "cpu";
        hardware-exec-breakpoint-count = <1>;
        i-cache-block-size = <64>;
        i-cache-sets = <16>;
        i-cache-size = <1024>;
        i-tlb-sets = <1>;
        i-tlb-size = <4>;
        mmu-type = "riscv,sv32";
        next-level-cache = <&L16>;
        reg = <0x0>;
        riscv,isa = "rv32imac";
        riscv,pmpgranularity = <4>;
        riscv,pmpregions = <8>;
        status = "okay";
        timebase-frequency = <1000000>;
        tlb-split;
        L4: interrupt-controller {
                #interrupt-cells = <1>;
                compatible = "riscv,cpu-intc";
                interrupt-controller;
        };
};
```

Outline

1. Introduction
2. System architecture
3. Git clone and prepare
4. Make the system
5. Program Arty-A7
6. Using IntelliJ IDEA-IC
7. Modifying system by Scala config
8. Practice: system modification

8. Practice: system modification (1/1)

Exercise 1:

Try these combinations (`$ make default` → `$ make bit`) and report the resources in Arty-A7:

- RV32IMAC small single-core Rocket
- RV64GC small single-core Rocket
- RV32IMAC medium single-core Rocket
- RV64GC medium single-core Rocket
- RV32IMAC small single-core Rocket with reduced caches ($I=1KB, $D=1KB)
- RV64GC medium single-core Rocket with increased caches ($I=64KB, $D=64KB)

Exercise 2:

Try multiple configurations to see if the Arty-A7 can support dual-core Rocket.
And check on both **35T** and **100T** versions of the Arty-A7. So the questions are:

1. Which configuration will use the most of the **Arty-A7-35T**?
2. Which configuration will use the most of the **Arty-A7-100T**?

THANK YOU

2022/11