ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ - VIỄN THÔNG

DESLAB

RISC-V Day Vietnam 2022

IMPLEMENTATION OF 32-BIT AND 64-BIT RISC-V ON HARDWARE

DUC-HUNG LE

15 January 2022

CONTENT

- Introduction to HCMUS, FETEL and DESLAB
- Sharing some RISC-V implementation results
- Ongoing Plans

HISTORY

1941	**Division of Indochina College of Science**
1956	**Faculty of Science** *the University of Saigon*
1977	**Ho Chi Minh City University**
1996	**University of Natural Sciences**
2007 until Now	**University of Science** - Vietnam National University HCMC (http://www.hcmus.edu.vn)

> 14.000 students
- ~13.000 undergraduates
- ~1.500 graduates

120 programs
- 68 graduate
- 52 undergraduate

Information technology

Biology and biotechnology

Chemistry

Electronics and telecommunications

Mathematics and computer science

Material science

Physics and engineering physics

Environmental science

Geology

*"**Leading key institution** of undegraduate and post-graduate education, research, and technology transfer in the fields of natural sciences, applied sciences and technologies **of VNUHCM and southern Vietnam.**"*

3

INTRODUCTION TO FETEL

Faculty of Electronics and Telecommunications (FETEL)

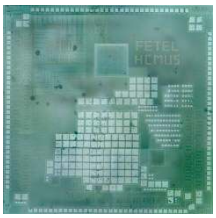
- Undergraduate (Bachelor of Science)
  - Department of Electronics (IC Design, Biomedical Electronics, Automation Control)
  - Department of Computer and Embedded Systems (Computer Engineering, Embedded Programming)
  - Department of Network and Telecommunications (Telecommunication Engineering, Telecommunication Systems)
- Graduate
  - Master of Electronics Engineering (Master of Science)
    - Major of IC Design and Microelectronics
    - Major of Computer – Electronics and Telecommunications
  - Ph.D. of Radio and Electronics

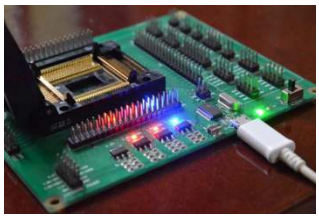
4

INTRODUCTION TO DESLAB

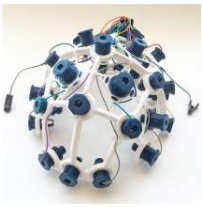
Current research themes

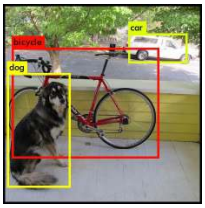
- Low power analog and digital IC design, Embedded designs (FPGA, MCU, CPU) and chip verification
- Digital Signal Processing and Biomedical Signal Processing
- Smart Integration Systems


IC Design


Test chip and verification


FPGA Design

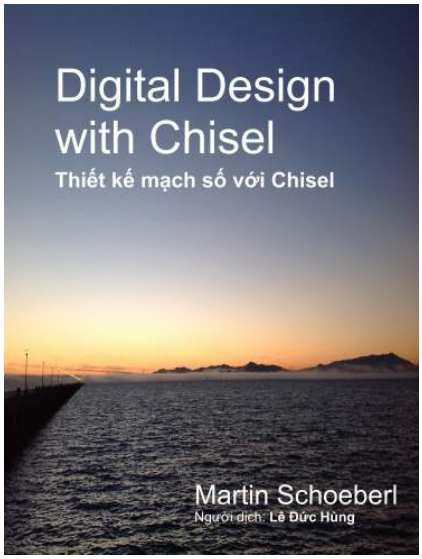

EEG Signal Processing



GPU-based AI system

BOOK TRANSLATION

- A little contribution to community: Translate “Digital Design with Chisel” of Martin Schoeberl into Vietnamese.
- The translated book is free, open-access and has been published since Q1/2021.
- Vietnamese version is available at page of Martin Schoeberl: http://www.imm.dtu.dk/~masca/chisel-book.html



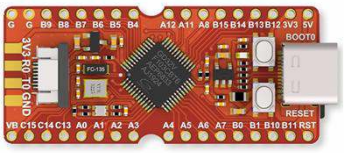
INTRODUCTION TO RISC-V

- RISC-V ISA defines the software interface, not hardware implementation.
- In a RISC-V based SoC, RISC-V compatible core(s) will be used to run software compiled for RISC-V ISA.
- Some open-source RISC-V 32-bit and 64-bit cores:
  - RISC-V microcontroller (FE310-G003, GD32VF103, etc).
  - **Rocketchip** (32-bit, 64-bit RISC-V)
  - **VexRiscV** (32-bit, with MMU)
  - PicoRV32 (32-bit, very small)
  - XuanTie C910 (64-bit, high performance)
  - Berkeley Out-of-Order Machine (BOOM) (64-bit RISC-V)

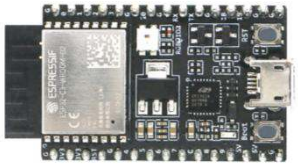
7

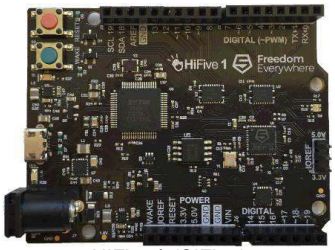
INTRODUCTION TO RISC-V

- Some commercial products using RISC-V



GD32VF103CBT6 (GigaDevice)


ESP32-C2 (Expressif)



HiFive1 (SiFive)

GD32 RISC-V Dev Board (SeeedStudio)

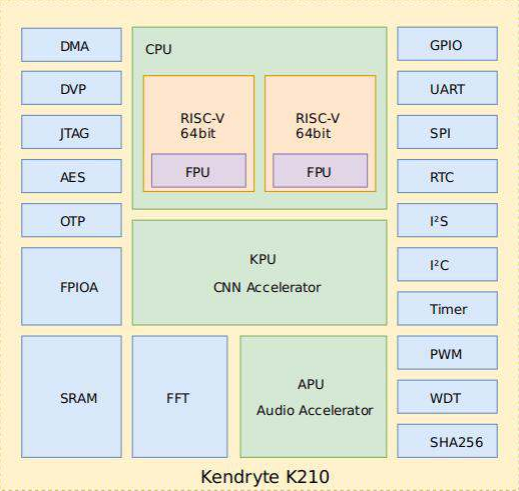
8

IMPLEMENTATION OF RISC-V ON HARDWARE

RISC-V SoC Design Flow

- A SoC needs more than just CPU cores!
- Additional IPs needed:
  - Peripherals: GPIO, Timer, Accelerators, JTAG, etc.
  - Bus(es): AXI, Avalon, TileLink, Wishbone, etc.
  - Bus bridge, synchronizer, power management, etc.

→ Long time to develop.



K210 SoC from Kendryte.

9

IMPLEMENTATION OF RISC-V ON HARDWARE

RISC-V SoC Design Flow

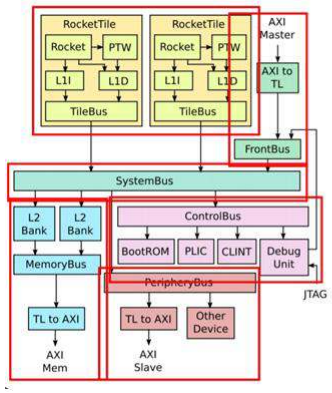
- Chipyard is a framework for designing and evaluating full-system hardware using agile teams. It is composed of a collection of tools and libraries designed to provide an integration between open-source and commercial tools for the development of systems-on-chip. [Source: Chipyard]
- Chipyard is a generator solution from UC-Berkeley → Faster time to develop
- Chipyard supports Rocket and BOOM cores
- Chipyard uses **Chisel** and mix-in **Configs** to generate Verilog RTL

10

IMPLEMENTATION OF RISC-V ON HARDWARE

RISC-V SoC Design Flow

- Example: Internal architecture of Rocketchip (Chipyard Rocket core generator):



Tiles: unit of replication for a core
- CPU
- L1 Caches
- Page-table walker

L2 banks:
- Receive memory requests

FrontBus:
- Connects to DMA devices

ControlBus:
- Connects to core-complex devices

PeripheryBus:
- Connects to other devices

SystemBus:
- Ties everything together

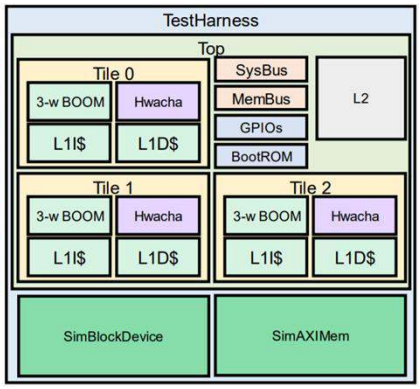
11

IMPLEMENTATION OF RISC-V ON HARDWARE

RISC-V SoC Design Flow

- Example: Configuration using existing module

```
class MyCustomConfig extends Config(
  new WithExtMemSize((1<<30) * 2L)          ++
  new WithBlockDevice                       ++
  new WithGPIO                              ++
  new WithBootROM                           ++
  new hwacha.DefaultHwachaConfig            ++
  new WithInclusiveCache(capacityKB=1024)   ++
  new boom.common.WithLargeBooms            ++
  new boom.system.WithNBoomCores(3)         ++
  new WithNormalBoomRocketTop               ++
  new rocketchip.system.BaseConfig)
```



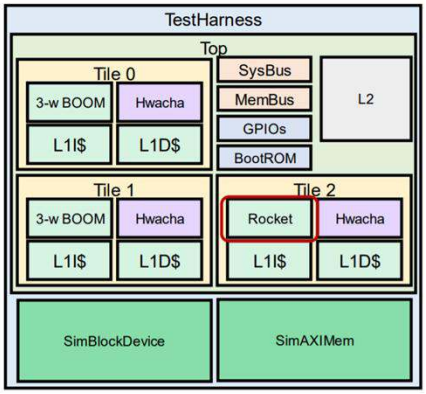
12

IMPLEMENTATION OF RISC-V ON HARDWARE

RISC-V SoC Design Flow

- Example: Allow flexible configuration by modification:

```
class MyCustomConfig extends Config(
  new WithExtMemSize((1<<30) * 2L)          ++
  new WithBlockDevice                       ++
  new WithGPIO                              ++
  new WithBootROM                           ++
  new hwacha.DefaultHwachaConfig            ++
  new WithInclusiveCache(capacityKB=1024)   ++
  new boom.common.WithLargeBooms            ++
  new boom.system.WithNBoomCores(2)         ++
  new rocketchip.subsystem.WithNBigCores(1)++
  new WithNormalBoomRocketTop               ++
  new rocketchip.system.BaseConfig)
```



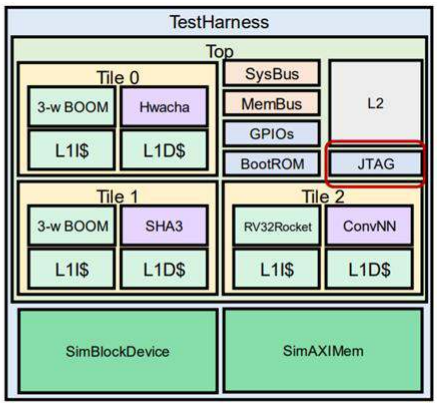
13

IMPLEMENTATION OF RISC-V ON HARDWARE

RISC-V SoC Design Flow

- Example: Allow flexible configuration by modification:

```
class MyCustomConfig extends Config(
  new WithExtMemSize((1<<30) * 2L)          ++
  new WithBlockDevice                       ++
  new WithGPIO                              ++
  new WithJtagDTM                           ++
  new WithBootROM                           ++
  new WithMultiRoCCConvAccel(2)             ++
  new WithMultiRoCCSha3(1)                  ++
  new WithMultiRoCCHwacha(0)                ++
  new WithInclusiveCache(capacityKB=1024)   ++
  new boom.common.WithLargeBooms            ++
  new boom.system.WithNBoomCores(2)         ++
  new rocketchip.subsystem.WithRV32         ++
  new rocketchip.subsystem.WithNBigCores(1)++
  new WithNormalBoomRocketTop               ++
  new rocketchip.system.BaseConfig)
```

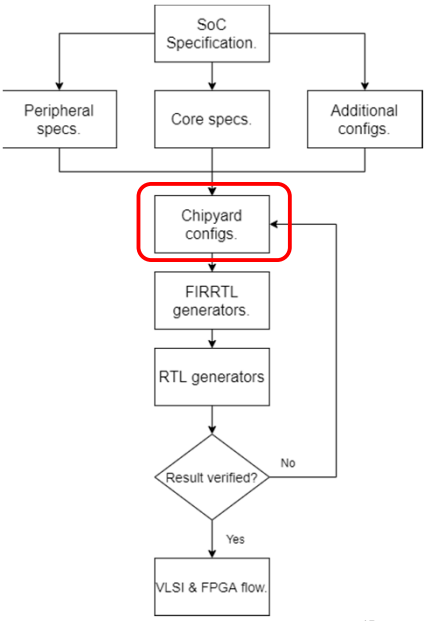


14

IMPLEMENTATION OF RISC-V ON HARDWARE

RISC-V SoC Design Flow

- Design flow using Chipyard:
  - SoC specifications: Number of cores, 32-bit/64-bit, peripheral, accelerators, L1 cache, etc.
  - Chipyard configs: configuration using Scala and Chisel.
  - FIRRTL generators: Generate FIRRTL from Chisel/Scala to feed into RTL generators.
  - RTL generators: Generate Verilog RTL from FIRRTL, along with other necessities.

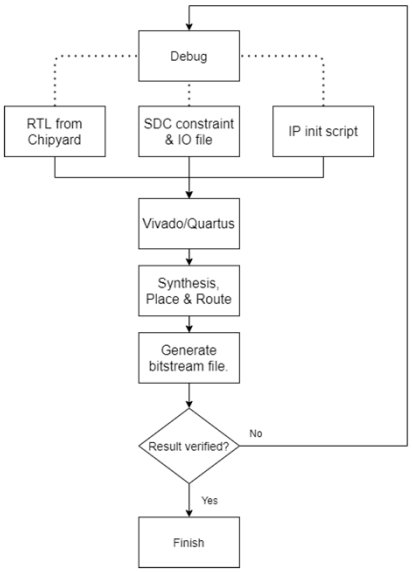


15

IMPLEMENTATION OF 32-BIT RISC-V ON FPGA

RISC-V SoC FPGA Design Flow

- **Input:** Verilog code from RTL generators
- Verilog RTL codes is fed into FPGA Design Flow
- Faster prototype implementation and check functionality
- Develop demo results
- **Output:** Bitstream file to program on FPGA

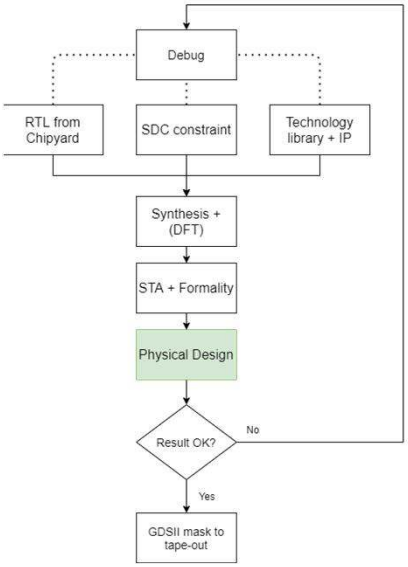


16

IMPLEMENTATION OF 32-BIT RISC-V ON FPGA

RISC-V SoC VLSI Design Flow

- **Input:** Verilog code from FPGA design flow
- Verilog codes is fed into ASIC Design Flow
- Optimized results on physical design
- Redundant codes must be removed in Verilog RTL
- **Output:** GDS stream file for tape-out

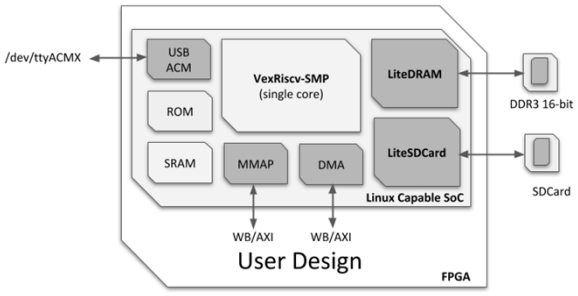


17

IMPLEMENTATION OF 32-BIT RISC-V ON FPGA

RISC-V SoC FPGA/VLSI Flow

- Example result (using **LiteX** framework with **VexRiscV** core):
  - Running Linux on a RISC-V SoC on FPGA (**Xilinx Nexys4DDR**).
  - SoC specification:
    - VexRiscV 32-bit with MMU + SMP.
    - DRAM and SDCard (in FPGA).
    - Ethernet interface (not shown).
    - AES-128, RSA-2048, SHA-1 accelerator (in CPU core).
    - Exported MMAP + DMA WB/AXI bus for user extension.
    - Internal ROM (for bootloader) + SRAM.
  - Load Linux image through UART connection.



18

IMPLEMENTATION OF 32-BIT RISC-V ON FPGA

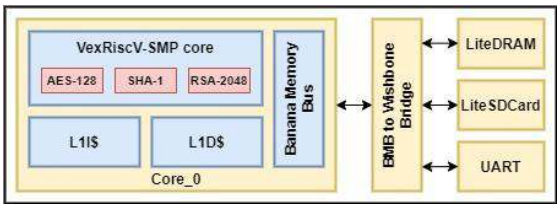
RISC-V SoC FPGA/VLSI Flow

- Example result (using **LiteX** framework with **VexRiscV** core) on FPGA:

IMPLEMENTATION OF 32-BIT RISC-V ON FPGA

RISC-V SoC FPGA/VLSI Flow

- Develop some basic and traditional cryptography cores as accelerators on this Litex on FPGA
  - AES-128, SHA-1, RSA-2048 cores



Resource Utilization	Base	AES-128	AES + SHA-1	AES-128 + SHA-1 + RSA-2048
LUT	8697	8827	10263	33595
LUTRAM	421	421	421	437
FF	7577	7643	9206	64389
BRAM	44	44.50	44.50	44.50
DSP	4	4	4	4
IO	98	98	98	98
BUFG	7	7	7	10
MMCM	1	1	1	1

IMPLEMENTATION OF 32-BIT RISC-V ON FPGA

RISC-V SoC FPGA/VLSI Flow

- Develop some basic and traditional cryptography cores as accelerators on this Litex on FPGA
  - SHA-1 accelerator functional verification

```
Input string: riscv
SHA1 core output: 2f60debd470b0d4fd4f0671f879e6304890e1ab0
Software output (OpenSSL): 2f60debd470b0d4fd4f0671f879e6304890e1ab0
```

  - RSA-2048 accelerator functional verification

```
RSA TEST
Public key (N): 60af4d6e17cd4b9129bc8f6133be4da550392cb631e9b2b6f4cbd5b32ba219a3c7d958
26d14a3584d4d9a7260fad46958f035d328e7d183c2817cfc47785f8f3fb90ee631ce6c
63cba652967d00ae16d4d7010adebc23a0e9f5fe82942f5a3f8a44e842cd2f7445e1ea0
8f0452ee965f81069a548de88b4ed10a2614f804b417
Private key (D): 7ee04d29f60e2fe634f0feeebdaa33d6866a21248e677160e9e93b6c3e0c20bd6e9ae73
5395cbad2af475833784b18df56a7417c00c464333d21d242adde9f8c5848988a08815
669121cc1d1b186e7b1a479e71c3b1507fb5aa9d8b8b025c08c088191762d57c08cc2
dbfddf9b43ad7cf54a122f634ac635a7c3fd8ccdfd7221
Text in (Org): 5851f42d40b18ccf4bb5f6464703312930705b0420fd5db41a8b7f78502959d82b8948
686c0356a708cdb7ff3477d43f70a3a52b28e4baf17d8341fc0ae16fd9742d2f7a0d1f0
79676035e0940f7702c6fa72ca52aa8415758a0df74474a03642e533cc404185faf6de3
b1150cab86287043bfa4398150e9375216574d9a2fdb
Text out (Encrypted): 489e02ff5abbdaece6f38b0a5bcb97f3ecee25b4543f008de8812bd3013160d7a1559d1
442276a91709f578d74293503f38813c938df343739e9229c08edc26d52d478cac468db85c0b077408864e2c547d4b3c33b
9ab8eb873d1fb0e1d14de3e99e185a3a0fdebb3e6f6c96e8fd397f0ef641e120bab2bf5c221979d242657d
Text out (Decrypted): 5851f42d40b18ccf4bb5f6464703312930705b0420fd5db41a8b7f78502959d82b8948
686c0356a708cdb7ff3477d43f70a3a52b28e4baf17d8341fc0ae16fd9742d2f7a0d1f0
79676035e0940f7702c6fa72ca52aa8415758a0df74474a03642e533cc404185faf6de3
b1150cab86287043bfa4398150e9375216574d9a2fdb
Text in (Org) = Text out (Decrypted)
```


IMPLEMENTATION OF 32-BIT RISC-V ON ASIC

RISC-V SoC FPGA/VLSI Flow

- Results (using **LiteX** framework with **VexRiscV** core) on ASIC (PnR results):

```
Cell Count
----------------------------------------
Hierarchical Cell Count:          41847
Hierarchical Port Count:         303701
Leaf Cell Count:                 821304
Buf/Inv Cell Count:              100447
Buf Cell Count:                   24374
Inv Cell Count:                   76073
CT Buf/Inv Cell Count:                0
Combinational Cell Count:        420496
Sequential Cell Count:           400808
Macro Count:                          0
----------------------------------------


Area
----------------------------------------
Combinational Area:         1663492.92
Noncombinational Area:      6249090.54
Buf/Inv Area:                477995.35
Total Buffer Area:            81037.93
Total Inverter Area:         396957.42
Macro/Black Box Area:             0.00
Net Area:                            0
Net XLength:                      0.00
Net YLength:                      0.00
----------------------------------------
Cell Area (netlist):                     7912583.47
Cell Area (netlist and physical only):   8158254.40
Net Length:                       0.00
```

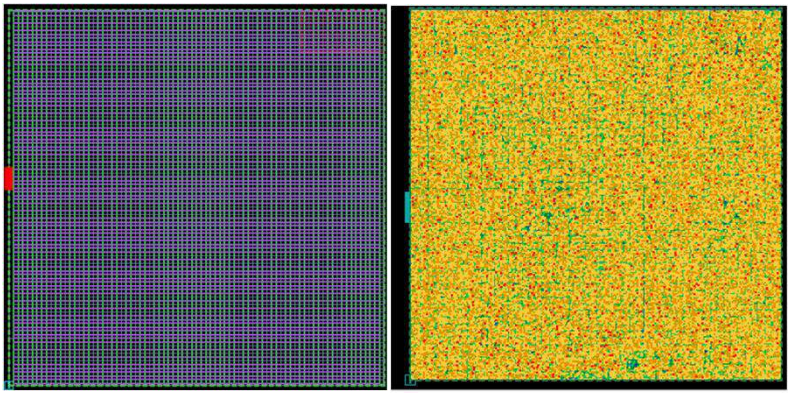
```
Utilization Ratio:                    0.5078
Utilization options:
 - Area calculation based on:         site_row of block vlsi
 - Categories of objects excluded:    hard_macros macro_keepouts soft_macros io_cells hard_blockages
Total Area:                           15963580.6290
Total Capacity Area:                  15582053.1330
Total Area of cells:                  7912595.8449
Area of excluded objects:
 - hard_macros         :              125309.2223
 - macro_keepouts      :              3134.9253
 - soft_macros         :              0.0000
 - io_cells            :              0.0000
 - hard_blockages      :              381527.4960

Utilization of site-rows with:
 - Site 'unit':                       0.5078
```

IMPLEMENTATION OF 32-BIT RISC-V ON ASIC

RISC-V SoC FPGA/VLSI Flow

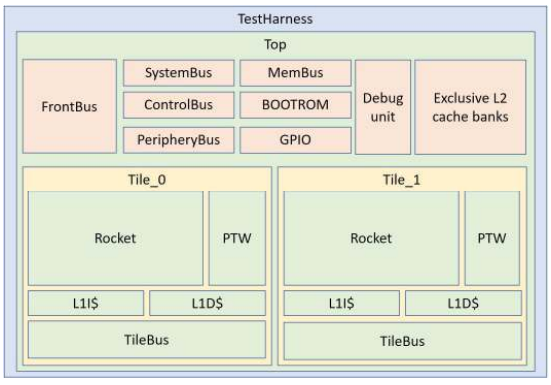
- Results (using **LiteX** framework with **VexRiscV** core) on ASIC (PnR results):



Specifications	Values
Frequency	50 MHz
Area	15,963,580 μm^2
Dimension	3.9 x 3.9 mm
Power	42.1 mW (@50MHz)
Process	CMOS 65nm
DRC	Clean
LVS	Pass

IMPLEMENTATION OF 64-BIT RISC-V ON ASIC

- Architecture of Dual-core RISC-V 64-bit
  - 2-core RV64GC Rocket CPU
  - L1, L2 Cache
  - Control Bus
  - System Bus
  - Peripheral Bus
  - BOOTROM, GPIO, etc.

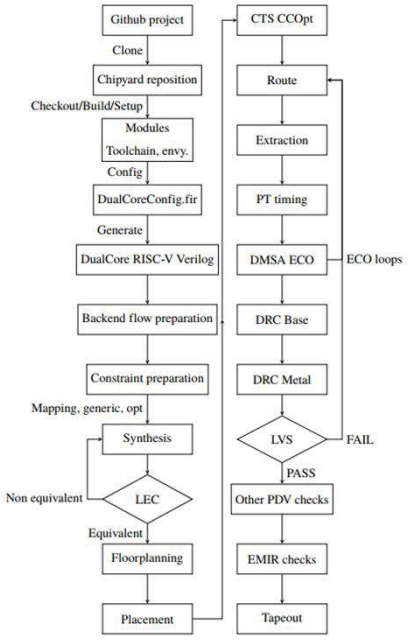


24

IMPLEMENTATION OF 64-BIT RISC-V ON ASIC

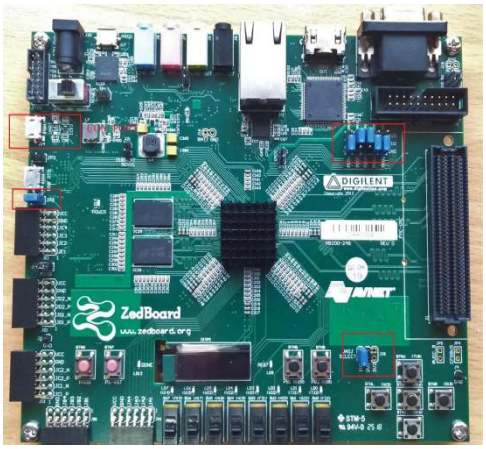
- **Digital design flow with Chisel**

1. Clone Chipyard
2. Build environments, tools, libraries
3. Config DualCoreConfig.fir
4. Generate RTL codes using FIRRTL and VerilogRTL tool
5. Setup timing constraints, library, scripts
6. Run synthesis to generate gate-level netlist
7. Check LEC Post-Synthesis
8. Generate floorplan, macro, blockages
9. Run PnR with gate-level netlist and timing constraints
10. Run LEC Post-Route verification
11. Run parasitic RC extraction
12. Run ECO Primetime ECO to fix timing Sign-off
13. Run DRC, LVS, antenna, boundary, ERC verification
14. Run EMIR static, dynamic and signal EM verification
15. Run final Sign-off verification
16. Tape-out



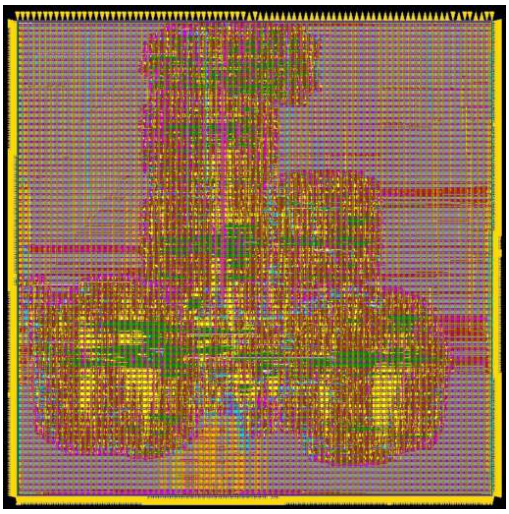
IMPLEMENTATION OF 64-BIT RISC-V ON FPGA

RISC-V SoC FPGA/VLSI Flow



IMPLEMENTATION OF 64-BIT RISC-V ON ASIC

Specifications	Values
Frequency	500 MHz
Area	1,384,255 μm^2
Dimension	1.17 x 1.17 mm
Power	493.6 mW (@500MHz)
Process	FinFET 7nm
DRC, Antenna, boundary, ERC	Clean
LVS	Pass
EM and IR drop	Clean
LEC from RTL to Post-Routing	Design Equal



27

IMPLEMENTATION OF 64-BIT RISC-V ON ASIC

- **ERC and Antenna Verification**

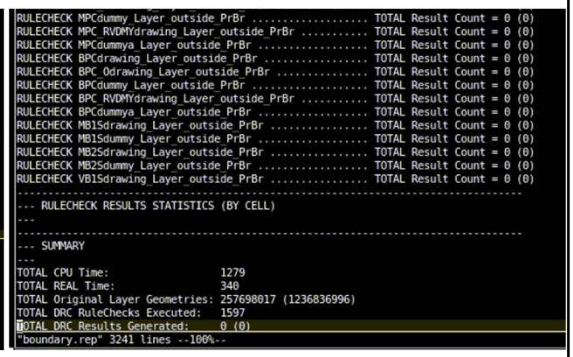
```
RULECHECK MPC_Odrawing_Layer_outside_PrBr ..... TOTAL Result Count = 0 (0)
RULECHECK MPCdummy_Layer_outside_PrBr ......... TOTAL Result Count = 0 (0)
RULECHECK MPC_RVDMYdrawing_Layer_outside_PrBr .. TOTAL Result Count = 0 (0)
RULECHECK MPCdummya_Layer_outside_PrBr ........ TOTAL Result Count = 0 (0)
RULECHECK BPCdrawing_Layer_outside_PrBr ....... TOTAL Result Count = 0 (0)
RULECHECK BPC_Odrawing_Layer_outside_PrBr ..... TOTAL Result Count = 0 (0)
RULECHECK BPCdummy_Layer_outside_PrBr ......... TOTAL Result Count = 0 (0)
RULECHECK BPC_RVDMYdrawing_Layer_outside_PrBr .. TOTAL Result Count = 0 (0)
RULECHECK BPCdummya_Layer_outside_PrBr ........ TOTAL Result Count = 0 (0)
RULECHECK MB1Sdrawing_Layer_outside_PrBr ...... TOTAL Result Count = 0 (0)
RULECHECK MB1Sdummy_Layer_outside_PrBr ........ TOTAL Result Count = 0 (0)
RULECHECK MB2Sdrawing_Layer_outside_PrBr ...... TOTAL Result Count = 0 (0)
RULECHECK MB2Sdummy_Layer_outside_PrBr ........ TOTAL Result Count = 0 (0)
RULECHECK VB1Sdrawing_Layer_outside_PrBr ...... TOTAL Result Count = 0 (0)
--------------------------------------------------------------------------------
--- RULECHECK RESULTS STATISTICS (BY CELL)
---
--------------------------------------------------------------------------------
--- SUMMARY
---
TOTAL CPU Time:                 1219
TOTAL REAL Time:                345
TOTAL Original Layer Geometries: 257679057 (1236818036)
TOTAL DRC RuleChecks Executed:   1597
TOTAL DRC Results Generated:     0 (0)
"boundary.rep" 3242L, 267880C
```

```
--------------------------------------------------------------------------------
--- RULECHECK RESULTS STATISTICS
---
RULECHECK pickup.p_to_p ........... TOTAL Result Count = 0 (0)
RULECHECK pickup.n_to_n ........... TOTAL Result Count = 0 (0)
RULECHECK pickup.n_to_p ........... TOTAL Result Count = 0 (0)
RULECHECK mnpg .................... TOTAL Result Count = 0 (0)
RULECHECK mppg .................... TOTAL Result Count = 0 (0)
RULECHECK mnpgldd ................. TOTAL Result Count = 0 (0)
RULECHECK floating.nxwell_float ... TOTAL Result Count = 0 (0)
RULECHECK floating.psub ........... TOTAL Result Count = 0 (0)
RULECHECK npvss150 ................ TOTAL Result Count = 0 (0)
RULECHECK npvss49 ................. TOTAL Result Count = 0 (0)
RULECHECK ppvdd150 ................ TOTAL Result Count = 0 (0)
RULECHECK ppvdd49 ................. TOTAL Result Count = 0 (0)
RULECHECK LVSDMY4_DNW_CHECK ....... TOTAL Result Count = 0 (0)
--------------------------------------------------------------------------------
--- ERC RULECHECK RESULTS STATISTICS (BY CELL)
---
--------------------------------------------------------------------------------
--- SUMMARY
---
TOTAL CPU Time:                 1511
TOTAL REAL Time:                688
TOTAL Original Layer Geometries: 246392023 (1045410547)
TOTAL ERC RuleChecks Executed:   13
TOTAL ERC RuleCheck Results Generated:  0 (0)
"erc.sum" 837 lines --100%--
```

28

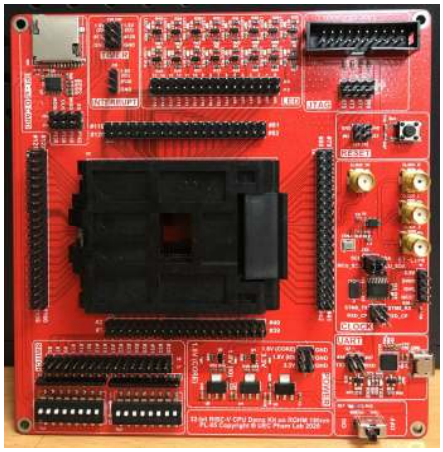
IMPLEMENTATION OF 64-BIT RISC-V ON ASIC

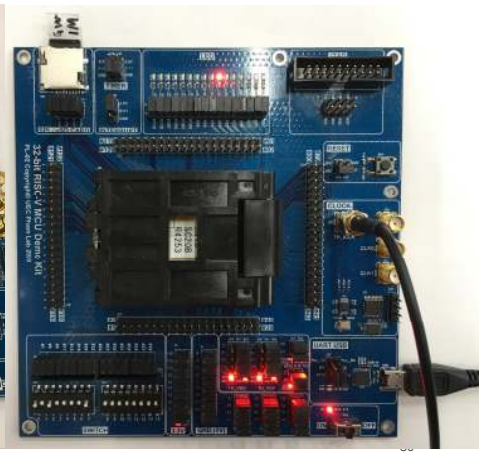
- **DRC and LVS Verification**



PCB DESIGN FOR RISC-V CHIP TEST

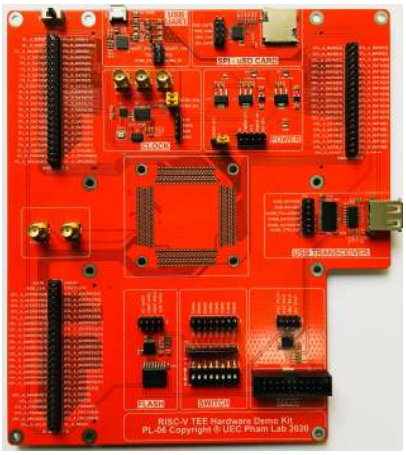
- PCB Design for 32-bit RISC-V Chip (Dedicated to Pham's Lab, UEC, Japan)





PCB DESIGN FOR RISC-V CHIP TEST

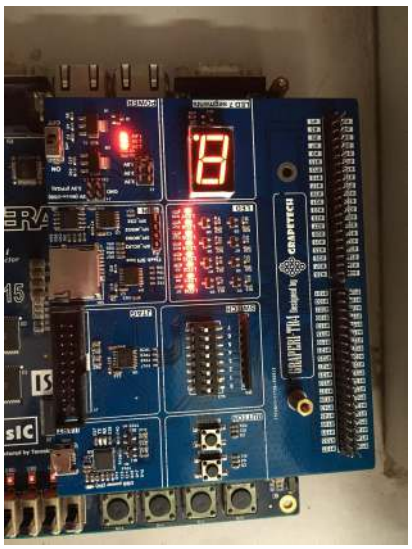
- PCB Design for 64-bit RISC-V Chip (Dedicated to Pham's Lab, UEC, Japan)



31

PCB DESIGN FOR RISC-V CHIP TEST

- Peripheral board for RISC-V chip test
  - Peripheral boards for CPU testing
  - UART
  - JTAG
  - SPI (Flash, SD Card)
  - I/O: Switch, Buttons, LEDs
  - Compatible with FPGA boards or other CPUs/MCUs

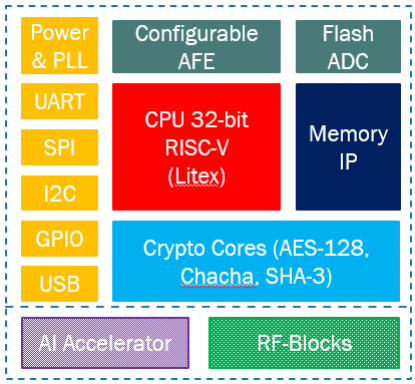


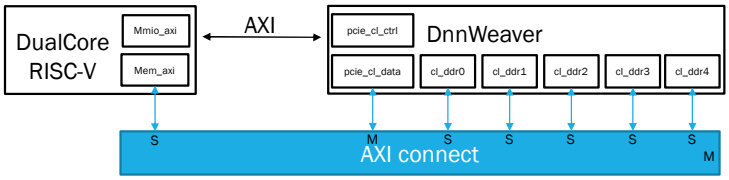
32

ONGOING PLANS WITH RISC-V

- H-23 Chip: DAQ Chip based on RISC-V
  - CPU 32-bit RISC-V (Litex): In progress
  - Flash ADC 8-bit (Completed)
  - Analog Front End (AFE): In progress
  - Traditional and Lightweight Cryptos: In Progress
- AI accelerator with RISC-V
  - DNN Weaver (open-source)
  - 2-core RISC-V 64-bit

All codes and designs will be open and free-access after completion





CONCLUSION

- The objective of this presentation is to share some our RISC-V implementation results on FPGA and ASIC.
- We are the newbie in RISC-V implementation → no new or state-of-the-art techniques → We hope to give a specific system on RISC-V in the next year.
- We look forward to collaborate with persons or organizations in IC design and SoC design.

THE END
THANK YOU

LDHUNG@HCMUS.EDU.VN