Implementation of a Dual-core 64-bit RISC-V on 7nm FinFET process

Van-Ninh Ho, Duc-Hung Le[1*]
[1] University of Science, Vietnam National University Ho Chi Minh City, Vietnam
[*] Email: ldhung@hcmus.edu.vn

***Abstract*— This paper presents a back-end implementation of a Dual-core 64-bit RISC-V using the proposed digital ASIC design flow with hardware construction language Chisel. A design flow started from the Chisel code of Dual-core 64-bit RISC-V generated from Chipyard. After that, Verilog source code was converted from Chisel with the configured Dual-core 64-bit RISC-V architecture. This design was successfully implemented on TSMC 7nm FinFET process with the proposed ASIC design flow and design techniques for complex CPU designs. The Dual-core 64-bit RISC-V has a core size of 1.17 x 1.17mm, operating at maximum frequency 500MHz, and consuming power 493.7mW.**

***Keywords*— RISC-V, Dual-core, 7nm FinFET.**

I. Introduction

RISC-V is not a microprocessor (CPU), but a free, open-source set of Instruction Set Architecture (ISA) that is based on the principles of the RISC architecture. With the fixed ISA specification, designers can build their own ISA sets. A study by Semico company forecasts that the market will consume 62.4 billion cores of CPU cores RISC-V by 2025, of which the industrial array will account for 16.7 billion cores. Therefore, the RISC-V will gradually be popular and gain more market sharing in the near future. In addition, the fact that NVidia has officially acquired ARM Holdings will make the battle of CPU cores between brands even more fierce, as chip design companies now have to consider choosing between the costs. out and bring efficiency to deliver customers the best performance products. ARM's acquisition by Nvidia also questioned the designers of the early ARM's advanced customization.

The goal of this paper is to study the complete RISC-V architecture along with the chip design process from Chisel (open-source) to RTL and from RTL to GDSII with EDA tools and back-end design techniques. More specifically, a multi-core processor, with open-source ISA from RISC-V, high frequency clock speed, and complete full set of tests was successful implemented for tape-out. First, Verilog codes would be configured and generated by the open-source Chisel language, a scalar language set, with a variety of highly customizable CPU core configurations. Then, RTL would be synthesized using the Cadence Genus tool. Next, the netlist will be imported for place-and-route (PnR) using the Cadence Innovus tool to generate the physical layout design. Finally, Physical Design Verification (PDV) tests would be used for verification by using the Mentor Graphic Caliber rule deck tool. Furthermore, during the implementation, advanced techniques in synthesis, in PnR, in PDV such as: using ULVT cells, useful skew technique, multi power cell library, shielding clock nets, non-default rule, etc. would be also used to increase design efficiency and clock speed.

II. FinFET technology

Fig. 1 shows the structure of a FinFET device. The FinFET device consists of a thin silicon body, with thickness of t_{fin} at is wrapped by gate electrodes. The device is termed quasi-planar as the current flows parallel to the wafer plane, and the channel is formed perpendicular to the plane. The effective gate length L_G twice as large as the fin height h_{fin} that we define a FinFET technology by their effective gate lengths. Therefore, L_G set to be 7nm in 7nm FinFET device models. In this work, we focus on the shorted-gate FinFET devices because they provide better driving strength. The 7nm FinFET device models are adopted from [1].

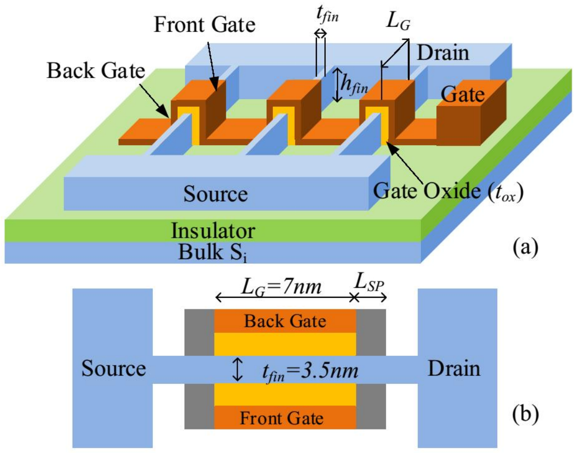

Fig. 1. (a) Perspective view and (b) top view of the 7nm FinFET device.

7nm FinFET standard cell libraries contain all typical types of combinational cells and sequential cells. Each cell is carefully sized to achieve equal rise and fall times at the characterization supply voltage level. The standard cell libraries are built in the Synopsys Liberty format [2], which is widely used for logic synthesis and static timing analysis. With the presented libraries, various benchmark circuits were synthesized; and their dynamic and static power consumption results were also reported. Comparisons between 7nm FinFET standard cell libraries and conventional CMOS libraries such as 14nm and 45nm, are carried out for same benchmark circuits. Synthesis results demonstrate that 7nm FinFET technology can achieve 10X and 1000X energy reductions on average in the super-threshold regime, and 16X and 3000X energy reductions on average in the near-threshold regime, compared to those results of 14nm and 45nm bulk CMOS technology nodes. This work forecasts the power consumptions of 7nm FinFET technology, while an analysis of process-induced variations at this technology node can be found in [3].

III. DUAL-CORE 64-BIT RISC-V IMPLEMENTATION

A. *Chisel and RISC-V Introduction*

Chisel is an open-source hardware construction language embedded in Scala [4]. It supports advanced hardware design using highly parameterized generators and supports RISC-V cores such as Rocket Chip and BOOM.

RISC-V is an open standard instruction set architecture based on established RISC (Reduced Instruction Set Computer) principles [5, 6]. The project RISC-V started in 2010 at UC Berkeley. Comparing to ARM and x86, RISC-V has the following advantages:

– Free: RISC-V is open-source.
– Simple: RISC-V is much smaller than other commercial ISAs.
– Modular: RISC-V has a small standard base ISA, with multiple standard extensions.
– Stable: Base and first standard extensions are already frozen. There is no need to worry about major updates.
– Extensibility: Specific functions can be added based on extensions. There are many more extensions are under development.

RISC-V base with standard extensions:

– Four base integer ISAs:
  - RV32E, RV32I, RV64I, RV128I
  - RV32E is 16-register subset of RV32I.
  - Only < 50 hardware instructions needed for base
– Standard extensions:
  - M: integer multiply/divide.
  - A: Atomic memory operations (AMO + LR/SC)
  - F: single-precision floating-point.
  - D: double-precision floating-point.
  - G = IMAFD, "general purpose" ISA.
  - Q: quad-precision floating-point.

B. *Dual-core 64-bit RISC-V Architecture*

The Dual-core 64-bit RISC-V source code was configured and generated from Chipyard. Block diagram and components of the Dual-core 64-bit RISC-V based on Chisel are configured in Fig. 2. The modules of design Dual Core RISC-V as below.

– 2 rocket cores RV64GC, each core has:
  - CPU
  - L1 caches
  - Page-table walker
– L2 banks:
  - Receive memory requests
– Front bus:
  - Connects to DMA devices
– Control bus:
  - Connects to Core-Complex devices
– Periphery bus:
  - Connects to other devices
– System bus:
  - Ties everything together (as known as fabric bus)

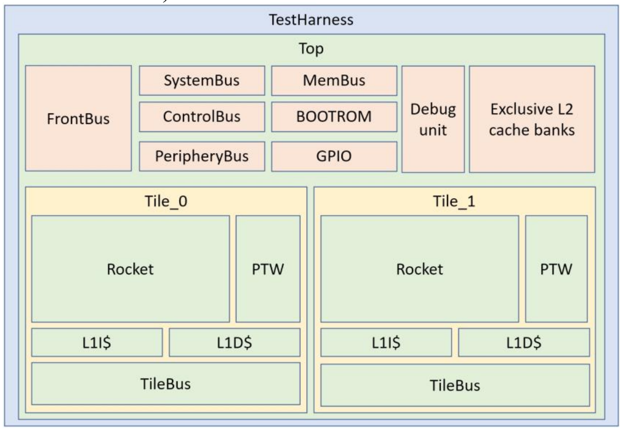

Fig. 2. The block diagram of Dual Core RISC-V design.

C. *Dual-core 64-bit RISC-V Implementation*

C.1. The proposed digital IC design flow with Chisel.

The digital design flow is proposed to generate GDS using hardware construction language Chisel. The Chisel code was converted to FIR [7], then from FIR to Verilog, and from Verilog to GDS. The proposed flow, which shown in Fig. 3, is described as below.

– Step 1: Clone Chipyard repository with Chisel from Github.
– Step 2: Checkout all modules of Chipyard on Linux.
– Step 3: Build RISC-V tools-chain from checked modules.
– Step 4: Build environment, paths, libraries on Linux.
– Step 5: Config DualCoreConfig.fir by Chisel in order to generate the synthesizable RTL source.
– Step 6: Generate RTL by FIRRTL.
– Step 7: Start to build the physical design flow, prepare libraries, technology files, design inputs, scripting…
– Step 8: Prepare timing constraint files for Synthesis step.
– Step 9: Synthesis RTL source code generated from FIRRTL.
– Step 10: Check Formal Verification.
– Step 11: Create chip level floorplan with taking care about the base-layer from manufactory.
– Step 12: Arrange modules, macros based on constraints about area, density and timing.
– Step 13: Place and Route with synthesized netlist and synthesized timing constraints generated from Synthesis.

- Step 14: Check Formal Verification.
- Step 15: Extract RC delay and perform PrimeTime ECO DMSA loops.
- Step 16: Check LVS, DRC, metal, antenna, boundary, bump, ERC.
- Step 17: Check EMIR static, signal EM.
- Step 18: Check Sign-off.
- Step 19: Tape-out.

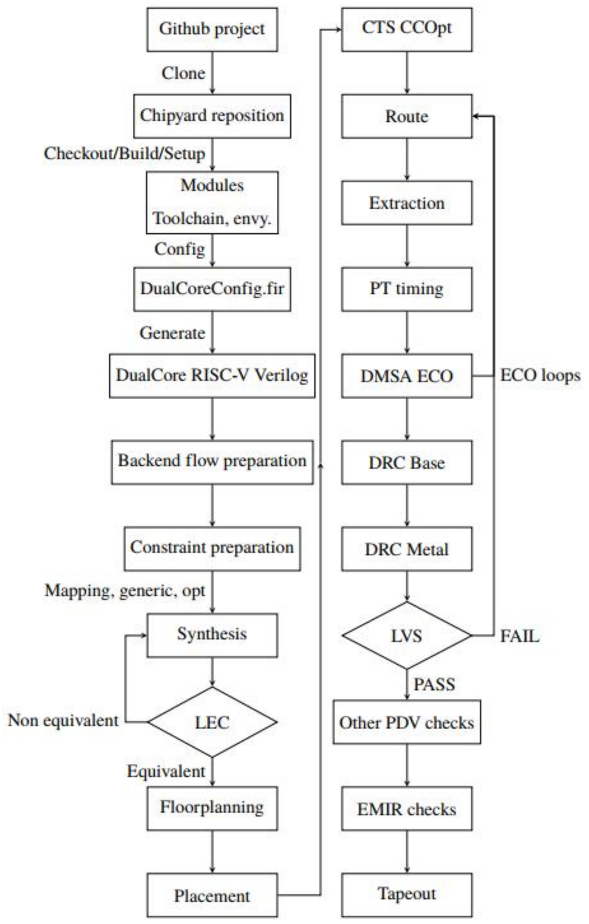

Fig. 3. The proposed ASIC design flow from RTL to GDS with Chipyard using HCL Chisel.

C.2. Function Verification

In order to verify the functional design throughout the design flow and make sure that design changes such as tools error, human error, mapping, optimization, and ECO changes do not affect the functionality of the RTL source code, we used Formal Verification (as known as Logic Equivalent Check) as a method that replaced the traditional simulation without using input vectors so that it is more efficient. The flow of LEC process is shown in Fig. 4. The steps of LEC check are described as below:

- Step 1: Read all design files, library elements, netlist, RTL for both golden design and revised design.
- Step 2; Mapping, LEC will process design into many small parts called design cones and starting mapping them between golden design and revised design.
- Step 3: Compare, LEC will compare functional equivalence between the logical design cones.
- Step 4: Debug, after comparing, LEC will generate reports to debug and resolve the LEC problems that can happen.

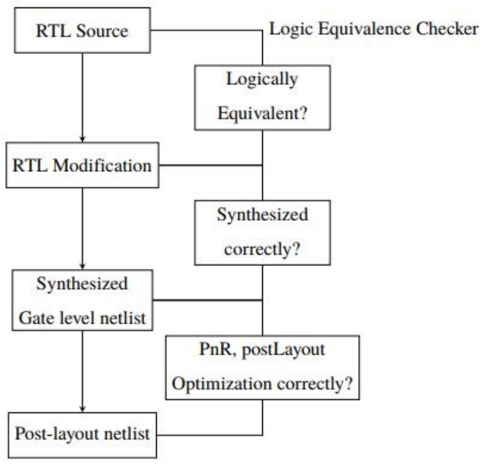
RTL Source
Logic Equivalence Checker
Logically Equivalent?
RTL Modification
Synthesized correctly?
Synthesized Gate level netlist
PnR, postLayout Optimization correctly?
Post-layout netlist

Fig. 4. The flow of Logic Equivalent Checking process.

C.3. Design techniques of Dual-core 64-bit RISC-V

With the generator tool of Chisel, we can generate many different configurations for a SoC. These configurations are specified by the parameters of Chisel, they can be changed depend on designer purposes.

```
  class DualCoreConfig extends Config(
  new WithTop ++
  new WithGPIO ++
  new WithBootROM ++
  new WithJtagDTM ++
  new
freechips.rocketchip.subsystem.WithInclusiveC
ache ++
  new boom.common.WithRenumberHarts ++
  new
freechips.rocketchip.subsystem.WithNBigCores(
2) ++
  new freechips.rocketchip.system.BaseConfig)
```

Timing constraints and timing analysis of the Dual-core 64-bit RISC-V are important parts beside physical verification and formal verification. Generally, a design is checked for functionalities by verification methods and is made sure that it will behave correctly after manufacturing by timing analysis.

STA (Static Timing Analysis) as well as Logic Equivalence Checking (LEC) are performed in many steps in the digital design flow, after synthesis, placement, clock tree and routing. Constraints contain period, frequency, net skew, maximum delay between endpoint or maximum net delay.

Below are some initial timing constraints of the design Dual-core 64-bit RISC-V with 2ns clock period.

```
  # Clock definition
  create_clock -name main_clk -period 2 -
waveform { 0.000 1 } [get_ports {
  clock}]
  # Clock transition
  set_max_transition 0.2 -clock_path
[get_clocks {main_clk}];
  set_max_transition 0.4 -data_path
[get_clocks {main_clk}];
  set_clock_transition 0.03 [get_clocks
{main_clk}]
  # set input/output delay
  set_input_delay -max -add_delay 0.6 -clock
[get_clocks main_clk] [
  get_ports -quiet * -filter "direction ==
in"]
  set_input_delay -min -add_delay 0 -clock
[get_clocks main_clk] [get_ports
  -quiet * -filter "direction == in"]
  set_output_delay -max -add_delay 0.6 -clock
[get_clocks main_clk] [
  get_ports -quiet * -filter "direction ==
out"]
  set_output_delay -min -add_delay 0 -clock
[get_clocks main_clk] [get_ports
  -quiet * -filter "direction == out"]
  # set drive cell and load capacitance
  set_load 0.01 [all_outputs]
  set_driving_cell -no_design_rule -lib_cell
$LIB(driving_gate) -pin X
  main_clk
  # set clock group
  set_clock_groups -asynchronous -name
MAIN_CLK -group [get_clocks main_clk]
  # set clock uncertainty.
  set_clock_uncertainty -setup 0.105
[get_clocks *]
```

With the above constraints, the Dual-core 64-bit RISC-V was performed at clock speed 500MHz, max clock transition is limited at 200ps for clock paths and 400ps for data paths. Besides, the clock transitions on all flipflops of main_clk limited at only 30ps. The clock uncertainty was set to make sure the clock correlation within the allowed margin when fabrication, by 105ps@500MHz. The timing optimization should be done at every steps of the design flow:

- Applying path delay adjustment to force the synthesis and push more effort on optimizing the timing.
  `path_adjust -delay -100 -from tile/dcache -to tile/dcache -setup`
- Floorplan size multiples of manufacturing grid (0.064 um) and technology cell height (0.024 um).
- Placement techniques: timing driven high effort, reclaim area while optimizing preCTS, early estimate trial route based on routing tracks for better correlation delay at CTS/route.
- CTS techniques: apply advanced route types for top/trunk/leaf clock nets and multi-cut via, apply shielding for clock nets to improve SI and crosstalk.
- Optimize power consumption at routing steps for dynamic power and leakage power.
  `setOptMode -powerEffort high -leakageToDynamicRatio 0.0.`

IV. Experiment results

The proposed ASIC design flow from Chisel to GDSII applied to Dual-core 64-bit RISC-V were successfully implemented on 7nm FinFET process. The layout view of the design is shown in Fig. 5. The final design Dual-core 64-bit RISC-V with bump connectivity is shown in Fig. 7. The design was implemented and optimized as flatten design style. The physical issue and inter-connection timing of whole chip was taken care better than hierarchical style.

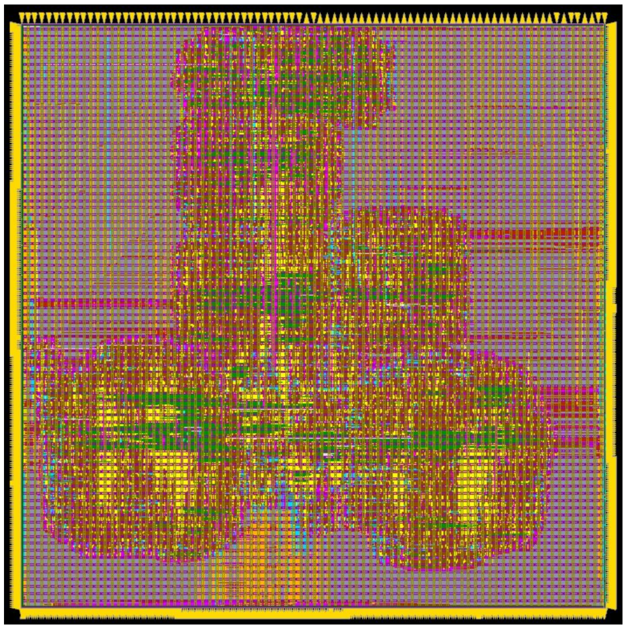
Fig. 5. Layout view of Dual-core 64-bit RISC-V with all routings and PG.

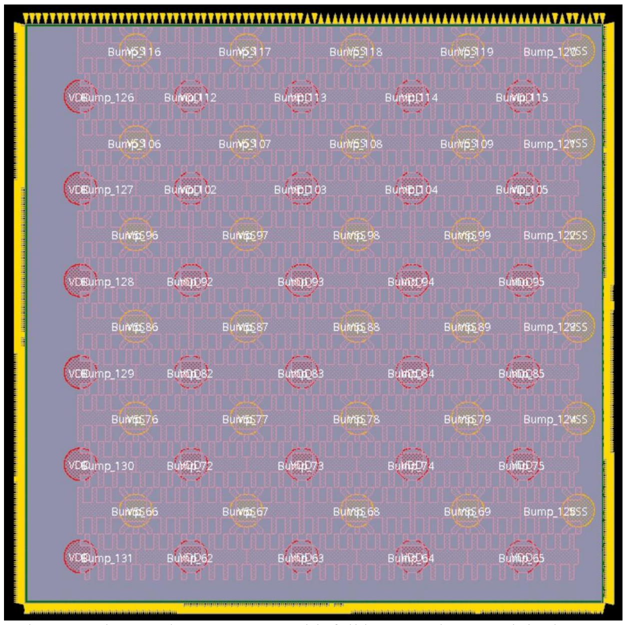
Fig. 6. Design Dual core RISC-V with full bump and connectivity bumps.

The overall results of Dual-core 64-bit RISC-V is shown in Table I:

TABLE I. SUMMARY OF DUAL-CORE 64-BIT RISC-V ON ASIC

Specifications	**Value**
Frequency	500 MHz
Area	1,384,255μm^2
Size	1.17 × 1.17 mm
Power	493,7mW (@500MHz)
Process	FinFET 7nm
DRC, ERC, bump, boundary	Clean
LVS	Passed
EMIR, EM	Clean

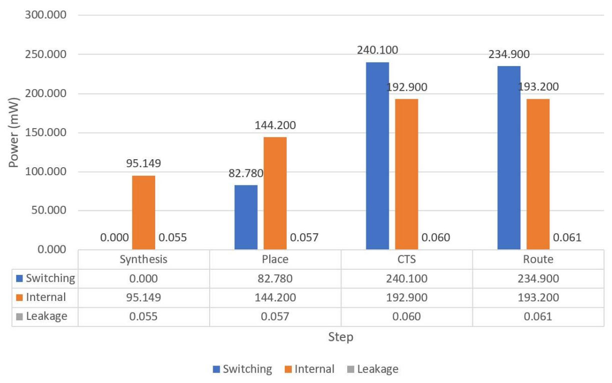

Fig. 7. Summary of total dissipation power.

The total dissipation switching power before CTS was low, because of the clock elements, which consumed the most power in design, was not built. Based on Fig. 7, the switching power at synthesis and placement was 0 and 83 mW, and at CTS and Route was increased by 240 mW and 234 mW respectively.

Timing sign-off will be verified by PrimeTime for most of scenarios, the scenarios are based on conditions, temperature, voltage threshold and process for each corner of wafer while fabrication and real-time usage by end-users. Below are the most dominant scenarios.

```
 func2C_setup1_ssgnp_0p675v_m40c__cworst-
t_m40c
 func2C_setup2_ssgnp_0p675v_m40c__rcworst-
t_m40c
 func2C_hold1_ffgnp_0p825v_m40c__cbest_m40c
 func2C_hold6_ffgnp_0p825v_m40c__rcworst_m40c
```

The timing result of PrimeTime is shown as the Table II. After post ECO PrimeTime, the setup timing is clean, the hold timing is violated 7 paths with worst slack is -13ps, total slack is -35ps.

TABLE II. THE RESULTS OF POST-LAYOUT TIMING QUALITY BY PRIMETIME.

Mode	**Setup (ns)**	**Hold (ns)**
WNS (reg2reg)	0	-0.0135
TNS (reg2reg)	0	-0.0348
NUM (reg2reg)	0	7
Transition	0	-0.007
Capacitance	0	0
Noise	0	0

V. CONCLUSION

The design Dual-core 64-bit RISC-V was successfully generated based on ISA GV64GC RISC-V. The Dual-core 64-bit RISC-V core was implemented on TSMC 7nm FinFET process using the proposed ASIC design flow with hardware construction language Chisel. The clock frequency is 500MHz. The die-dimension of the design is 1.17 mm × 1.17 mm, die area is around 1.38 mm^2. The core may consume 493.7 mW dissipation power at 500MHz.

The final GDS file was designed by using Genus and Innovus tools. The physical verification and timing verification are clean. The layout results are verified and guaranteed by Formal Verification LEC, PrimeTime, Calibre and Redhawk. Therefore, the GDS layout file meets all requirements for successful tape-out.

REFERENCES

[1] Q. Xie, X. Lin, Y. Wang, S. Chen, "Performance Comparisons between 7nm FinFET and Conventional Bulk CMOS Standard Cell Libraries," IEEE Transactions on Circuits and Systems II: Express Briefs, Vol. 62, Iss. 8, pp. 761 - 765, 2015.

[2] Liberty Library Modeling, Synopsys Inc., [online] http://www.synopsys.com/community/interoperability/pages/libertylibmodel.aspx.

[3] Q. Xie, Y. Wang, S. Chen, and M. Pedram. "Variation-aware joint optimization of supply voltage and sleep transistor size for 10nm FinFET technology," in ICCD, Oct. 2014.

[4] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanovi, "Chisel: Constructing hardware in a scala embedded language," in Proc. 49th Annu. Design Autom. Conf., pp. 1212–1221, Jun. 2012.

[5] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanoviä, "The RISC-V instruction set manual, volume I: User-level ISA, version 2.0," EECS Dept., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2014-54, May 2014.

[6] RISC-V Foundation. (2019). Rocket Chip Generator. [Online]. Available: https://github.com/chipsalliance/rocket-chip

[7] P. S. Li, A. M. Izraelevitz, and J. Bachrach, "Specification for the FIRRTL Language," EECS Dept., Univ. California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2016-9, Feb. 2016.