

LLM-Guided Generation of Power-Efficient RISC-V Cores

Using Predictive Feedback

Team Members & Name

Mekala Bindu Bhargavi & Ameya Ramateke

AutoCore AI (BITS-Pilani, Hyderabad Campus)



Cognichip Hackathon 2026

AGENDA

- ☐ Problem Statement
- ☐ Introduction
- ☐ Motivation
- ☐ Architecture Overview
- ☐ Results
- ☐ Conclusion
- ☐ Future Scope

Introduction

RISC–

- **V**RISC–V is an open standard, *load store* instruction set architecture.
- RISC–V is available freely under a permissive license.
- RISC–V is not...
 - A Company
 - A CPU implementation
- RISC–V uses a standard naming convention to describe the ISAs supported in a given implementation [1].
- ISA Name format: **RV[###][abc.....xyz]**
 - RV – Indicates a RISC–V architecture



Reference:

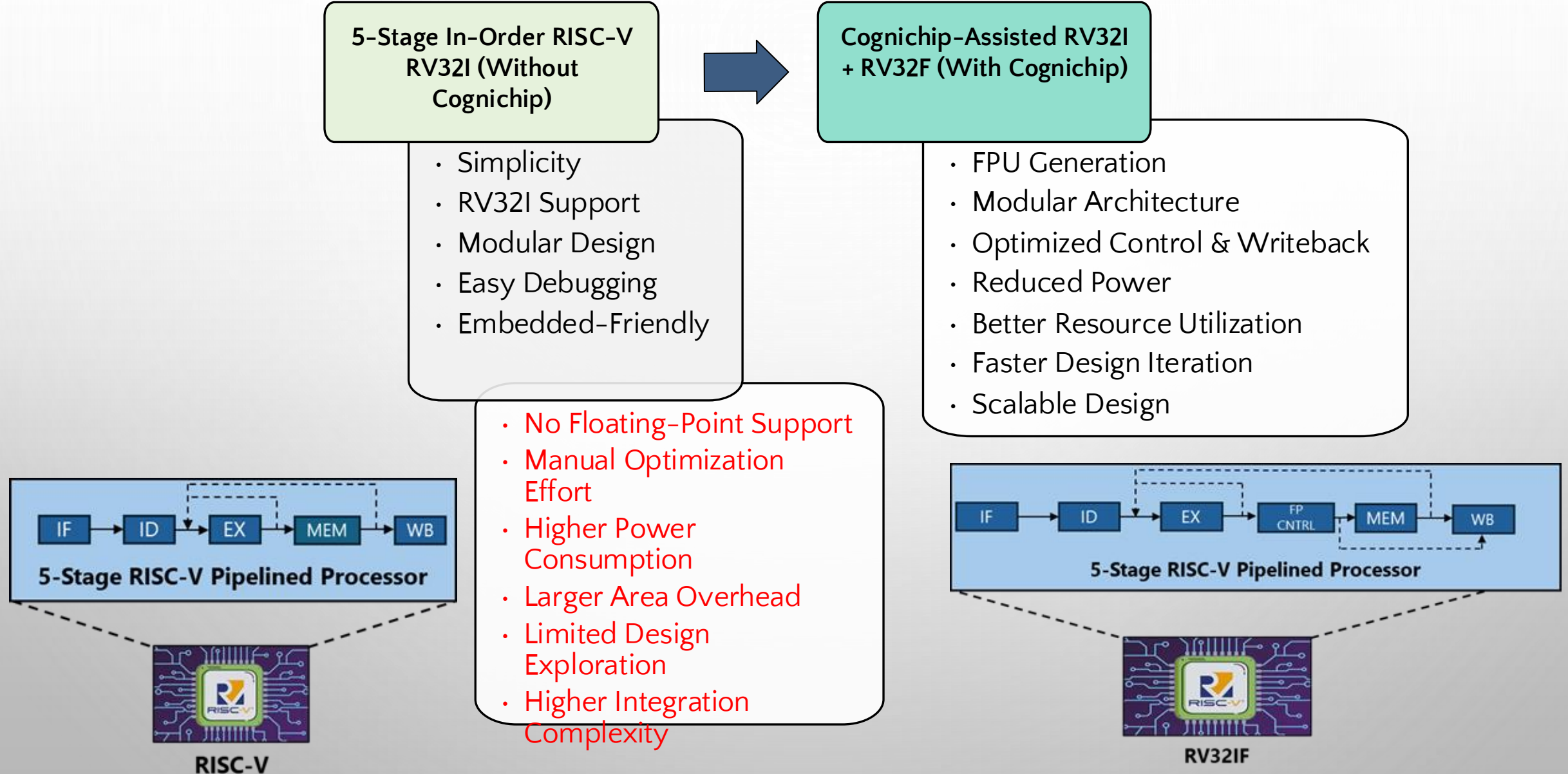
[1] E. Cui, T. Li, and Q. Wei, “Risc-v instruction set architecture extensions:A survey,” IEEE Access, vol. 11, pp. 24 696–24 711, 2023

INTRODUCTION Cntd..

- [###] – {32, 64, 128} indicate the width of the integer register file and the size of the user address space
- [abc...xyz] – Used to indicate the set of extensions supported by an implementation.
- RISC-V allows for custom, “Non-Standard”, extensions in an implementation.
- Putting it all together (examples)
 - RV32I – The most basic RISC-V implementation
 - RV32IMAC – Integer + Multiply + Atomic + Compressed

Extension	Description
I	Integer
M	Integer Multiplication and Division
A	Atomics
F	Single-Precision Floating Point
D	Double-Precision Floating Point
G	General Purpose = IMAFD
C	16-bit Compressed Instructions
Non-Standard User-Level Extensions	
Xext	Non-standard extension “ext”

Motivation



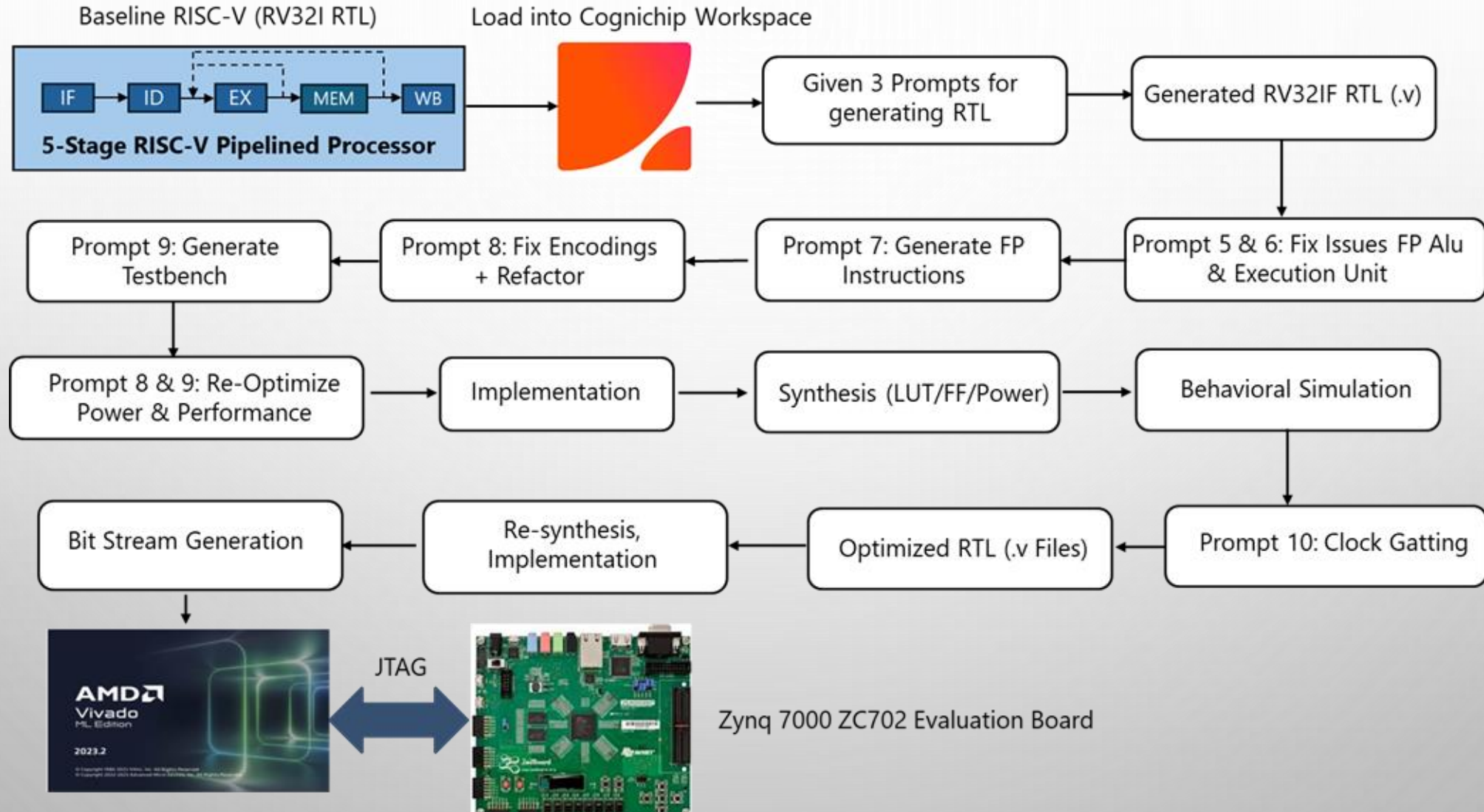
Problem Statement

- Embedded and edge systems increasingly need floating-point computation, but adding an FPU (RV32F) to a lightweight RISC-V core causes a significant increase in power and hardware usage.
- Manual design and optimization of such processors is slow and error-prone and limits exploration of better low-power architectures.

Goal

Use Cognichip as an AI-assisted co-designer to automatically generate and optimize an RV32IF-enabled RISC-V core that reduces power and area overhead while maintaining correct functionality and good performance.

End-to-End Design Flow for Cognichip-Assisted RV32IF Processor



Architecture Comparison: Baseline RV32I vs Cognichip-Optimized RV32IF

Aspect	Baseline RISC-V (RV32I)	Cognichip-Optimized RISC-V (RV32IF)
Pipeline	5-stage in-order (IF, ID, EX, MEM, WB)	5-stage in-order with FPU integration
ISA Support	RV32I (Integer only)	RV32I + RV32F (Floating Point)
FPU	Not present	AI-generated multi-cycle FPU
Design Method	Manual RTL development	AI-assisted RTL generation with prompts
Iterations	Manual iterations	8–9 guided prompt iterations
Instruction Encoding	Manual, error-prone	Generated and verified with fixes
Verification	Behavioral simulation	Behavioral simulation + iterative fixes
Resource Utilization	Baseline LUT/FF usage	Reduced LUT/FF after optimization
Power	Baseline power consumption	Re-optimized for lower power
Performance	Baseline performance	Optimized control and data path
Writeback & Control	Simple control path	Cleaner, optimized control path
Scalability	Limited	More modular and extensible
FPGA Validation	Tested on Zynq-7000 ZC702	Tested on Zynq-7000 ZC702
Overall Outcome	Integer-only functional core	RV32IF core optimized for Power and Performance

Evaluation Framework

Tools Used for Baseline Architecture RV32I:

- **RISC-V Encoder and Decoder rvcodec.js Tool[2]:** To generate I type Instructions.
- **AMD Vivado[3]:** Synthesized and implemented the RISC-V processor on the Zynq 7000 ZC702 Evaluation Board (xc7z020clg484-1).

System Configuration

```
graph TD; SC[System Configuration] --> S[Software]; SC --> H[Hardware];
```

Software

AMD Vivado Version: 2023.2.

rvcodec.js is an open-source, online RISC-V instruction encoder- decoder converter

Hardware

FPGA Platform: Zynq 7000 ZC702 Evaluation Board (xc7z020clg484-1)

Processor: 5-stage pipelined, 32-bit RISC-V processor

References:

[2] <https://luplab.gitlab.io/rvcodecjs/>

[3] <https://docs.amd.com/r/2023.2-English/ug973-vivado-release-notes-install-license/Licensing>

Evaluation Framework

Tools Used for Cognichip-Assisted RV32IF:

- **RISC-V Encoder and Decoder rvcodec.js Tool[2]:** To generate I and F type Instructions.
- **AMD Vivado[3]:** Synthesized and implemented the RISC-V processor on the Zynq 7000 ZC702 Evaluation Board (xc7z020clg484-1).

System Configuration

```
graph TD; SC[System Configuration] --> S[Software]; SC --> H[Hardware];
```

Software

AMD Vivado Version: 2023.2.
rvcodec.js is an open-source, online RISC-V
instruction encoder- decoder converter

Hardware

FPGA Platform: Zynq 7000 ZC702 Evaluation Board
(xc7z020clg484-1)
Processor: 5-stage pipelined, 32-bit RISC-V processor

References:

[2] <https://luplab.gitlab.io/rvcodecjs/>

[3] <https://docs.amd.com/r/2023.2-English/ug973-vivado-release-notes-install-license/Licensing>

RTL Schematic for Baseline Architecture RV32I

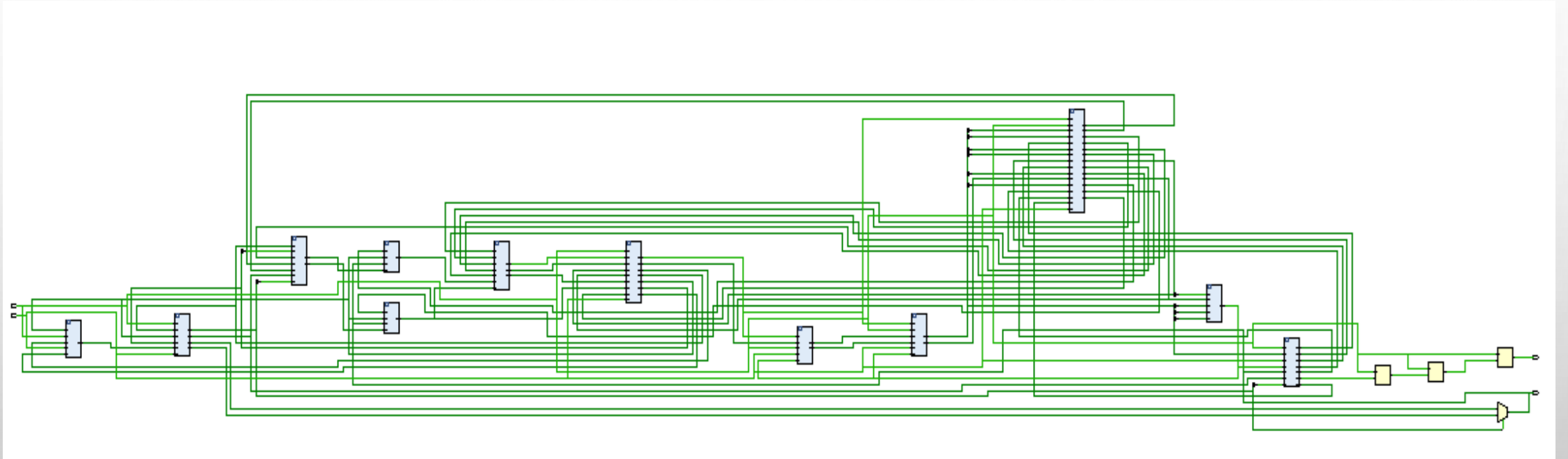


Fig: Schematic for Baseline Architecture RV32I

Results for Baseline Architecture RV32I

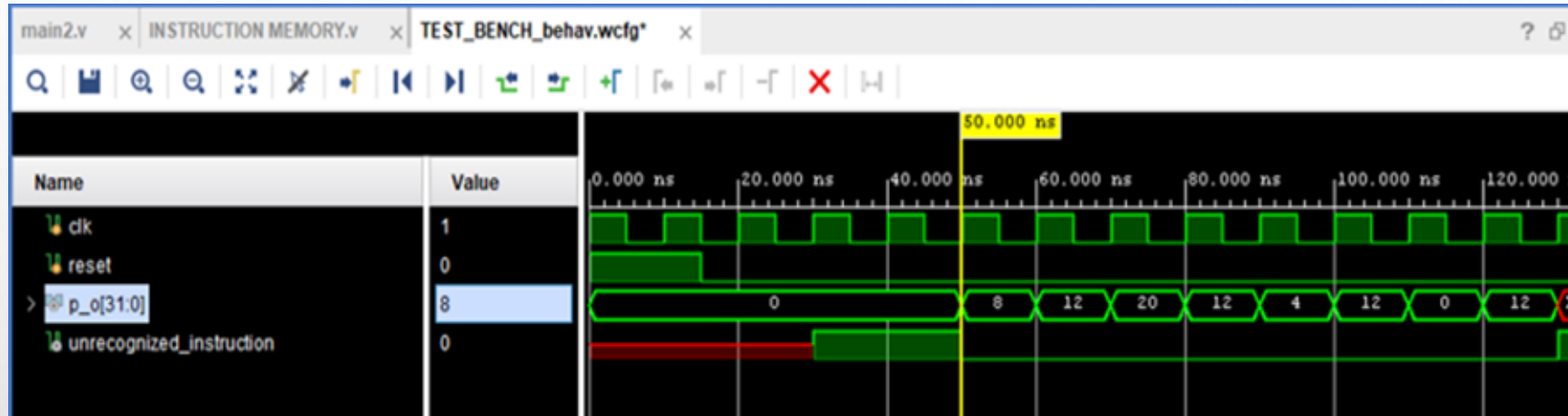


Fig. Behavioural Simulation Waveform



Fig. ILA Waveform

- Behavioral simulation used to verify functional correctness
- Waveform confirms correct instruction flow through pipeline
- Hardware validation performed using Integrated Logic Analyzer (ILA)
- Writeback data observed directly on FPGA
- Confirms correct runtime behavior of baseline processor
- Matches results seen in behavioral simulation

Results for Baseline Architecture RV32I

Name	^1	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	DSPs (220)	Bonded IOB (200)	BUFGCTRL (32)
▼ N RISC_V_PROCESSOR		19355	9930	4633	2112	3	35	4
> I dc_s (DECODE)		576	993	256	0	0	0	0
> I ex_s (EXECUTE_STAGE)		2600	101	25	0	3	0	0
I forwarding_unit (FORWARDING_UNIT)		64	4	0	0	0	0	0
> I if_s (INSTRUCTION_FETCH)		55	32	0	0	0	0	0
I mr_s (MEM_STAGE)		8736	8192	4352	2112	0	0	0
I p1 (IF_ID)		49	70	0	0	0	0	0
I p2 (ID_EX)		116	169	0	0	0	0	0
I p3 (EX_MEM)		7106	297	0	0	0	0	0
I P4 (MEM_WB)		52	71	0	0	0	0	0

Fig: On-Chip Performance with Base Architecture RV32I

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power: 0.243 W
Design Power Budget: Not Specified
Process: typical
Power Budget Margin: N/A
Junction Temperature: 27.8°C
Thermal Margin: 57.2°C (4.8 W)
Ambient Temperature: 25.0 °C
Effective θ_{JA} : 11.5°C/W
Power supplied to off-chip devices: 0 W
Confidence level: Medium
[Launch Power Constraint Advisor](#) to find and fix invalid switching activity

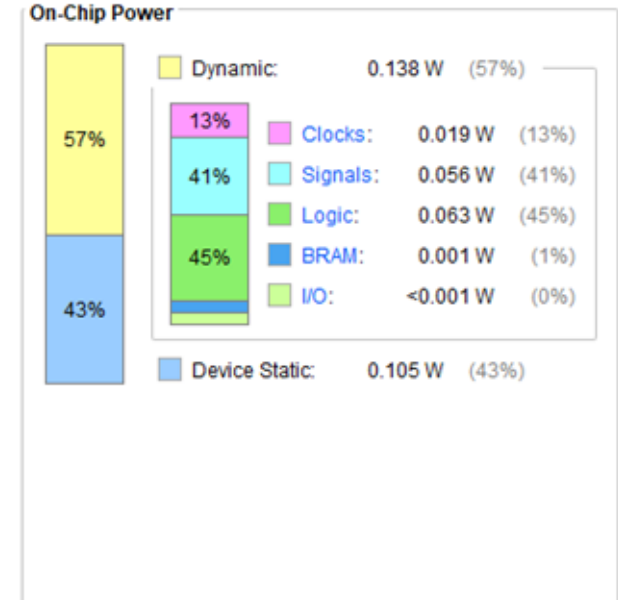


Fig: On-Chip power with Baseline Architecture RV32I

RTL Schematic for Cognichip-Assisted Optimized RV32IF

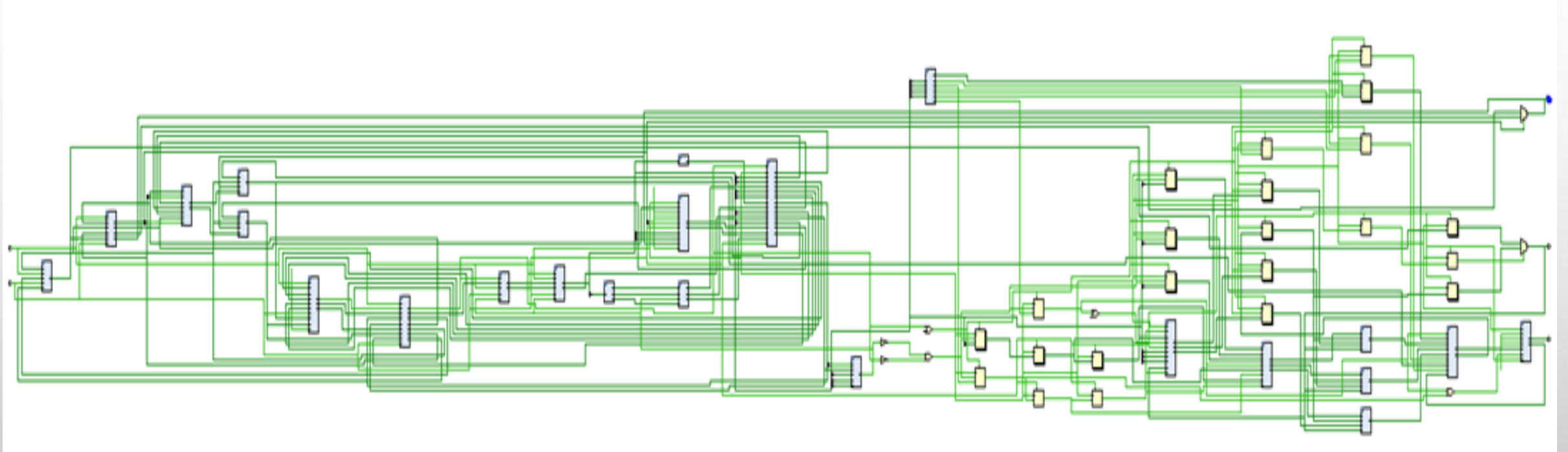


Fig: Schematic for Cognichip Optimized RV32IF

Results for Cognichip-Assisted Optimized RV32IF

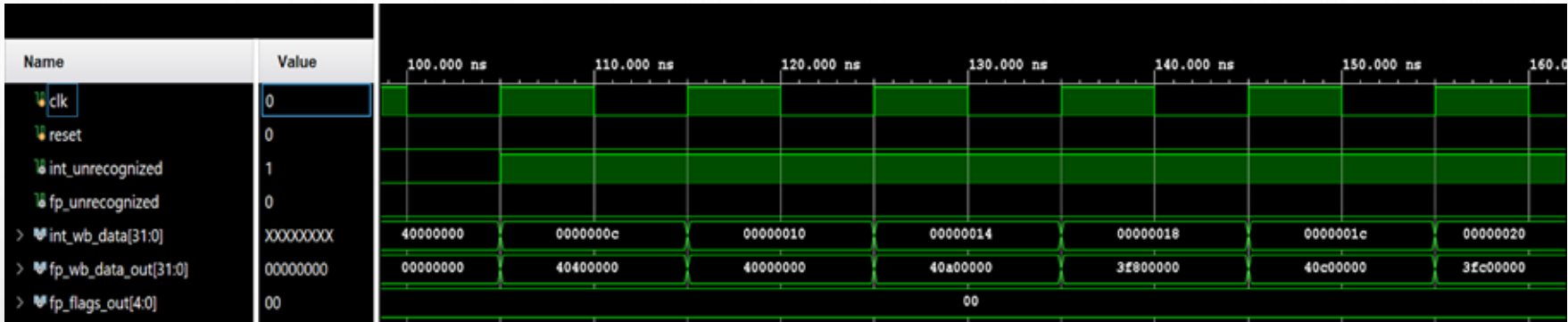


Fig. Behavioural Simulation Waveform

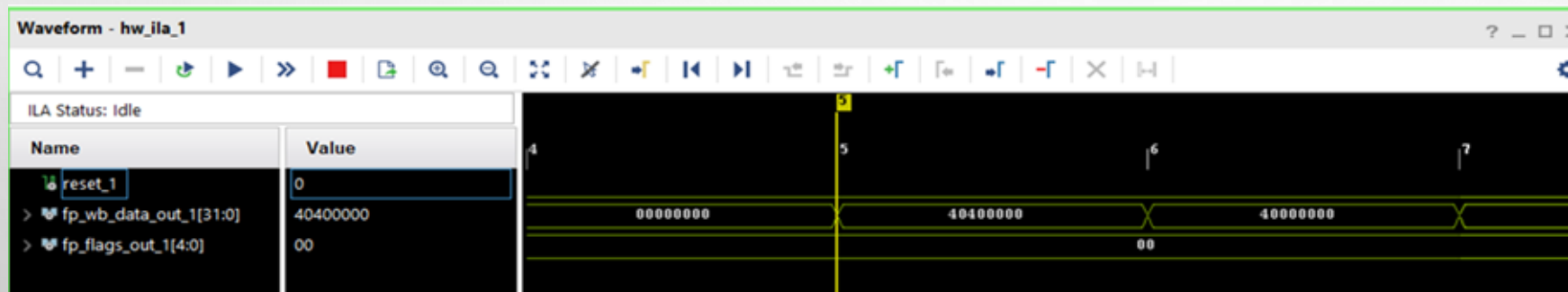


Fig. ILA Waveform

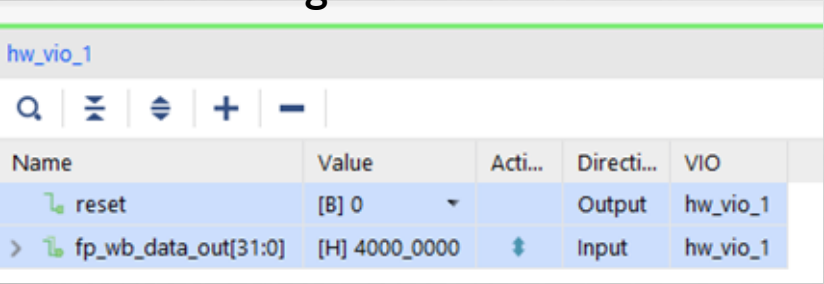


Fig. VIO Output

- Behavioral simulation confirms correct execution of RV32F instructions.
- Floating-point writeback data (fp_wb_data_out) shows expected IEEE-754 values.
- No unrecognized integer or floating-point instructions during execution.
- Waveforms show correct sequencing of FP operations: add, sub, mul, div, sqrt, min, max, compare, convert, and fused multiply-add.
- FP flags remain valid and stable during execution.

Results for Cognichip-Assisted Optimized RV32IF

Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)	Bonded IOB (200)	BUFGCTRL (32)
▼ N RISC_V_RV32F_PROCESSOR_POWER_OPT	17911	11019	4770	2145	6580	17911	71	1
ex_mem_reg (EX_MEM)	7083	296	0	0	2872	7083	0	0
fp_fwd_mux1 (FP_FORWARDING_MUXES)	42	0	0	0	35	42	0	0
fp_fwd_mux2 (FP_FORWARDING_MUXES_0)	7	0	0	0	3	7	0	0
fp_regfile (FP_REGFILE_POWER_OPT)	296	1024	132	33	330	296	0	0
id_ex_reg (ID_EX)	764	170	0	0	289	764	0	0
if_id_reg (IF_ID)	125	75	0	0	77	125	0	0
> if_stage (INSTRUCTION_FETCH_POWER_OPT)	87	64	0	0	31	87	0	0
> int_execute (EXECUTE_STAGE_POWER_OPT)	0	0	0	0	28	0	0	0
int_fwd_mux1 (FORWARDING_MUXES)	32	0	0	0	25	32	0	0
int_fwd_mux2 (FORWARDING_MUXES_1)	4	0	0	0	4	4	0	0
int_regfile (REGFILE_POWER_OPT)	576	992	256	0	520	576	0	0
mem_stage (MEM_STAGE)	8736	8192	4352	2112	5133	8736	0	0
mem_wb_reg (MEM_WB)	159	71	30	0	94	159	0	0

Fig: On-Chip Performance with Cognichip-Assisted Optimized RV32IF

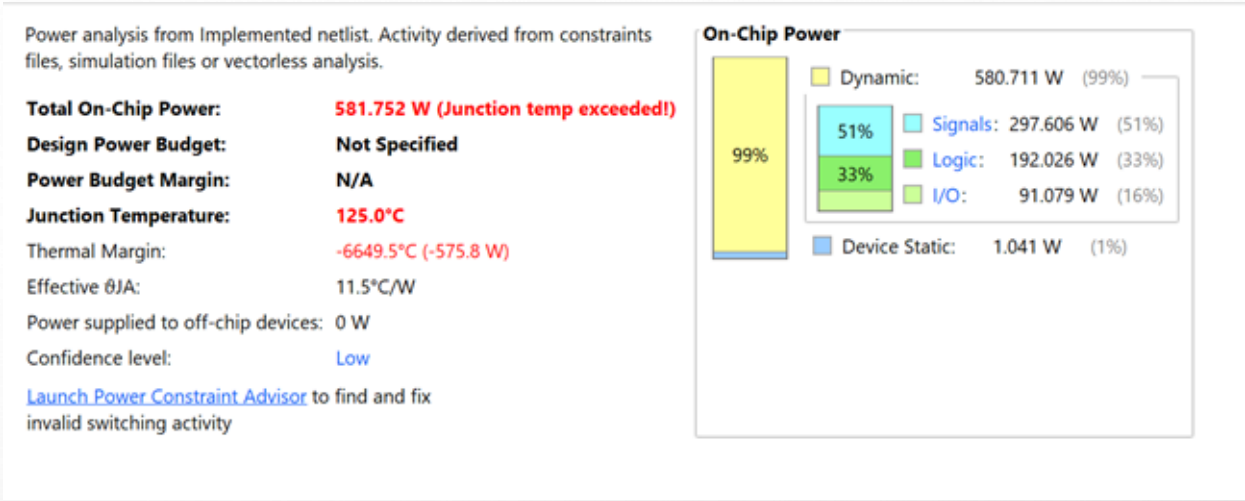


Fig: On-Chip power with Cognichip-Assisted RV32IF (Without Optimized Power Modules)

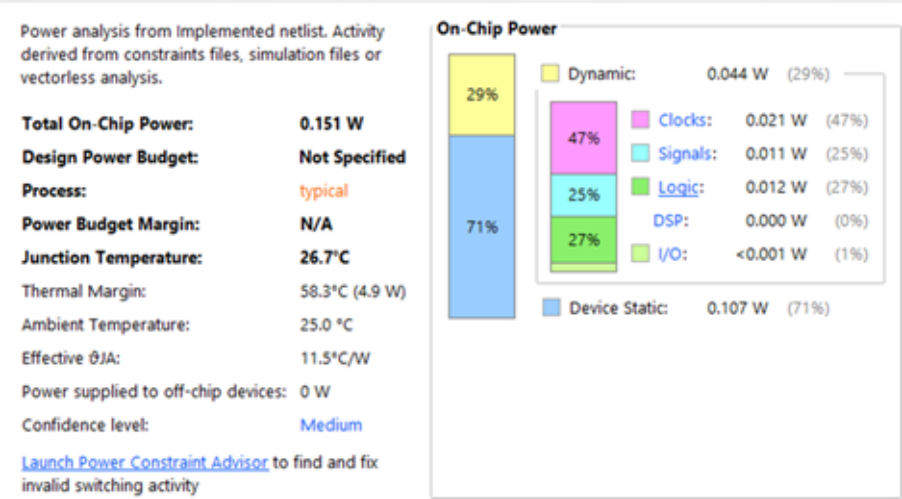


Fig: On-Chip power with Cognichip-Assisted RV32IF (With Optimized Power Modules)

Performance & Resource Optimization : Baseline vs Cognichip

Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	DSPs (220)	Bonded IOB (200)	BUFGCTRL (32)
▼ N RISC_V_PROCESSOR	19355	9930	4633	2112	3	35	4
> [I] dc_s (DECODE)	576	993	256	0	0	0	0
> [I] ex_s (EXECUTE_STAGE)	2600	101	25	0	3	0	0
[I] forwarding_unit (FORWARDING_UNIT)	64	4	0	0	0	0	0
> [I] if_s (INSTRUCTION_FETCH)	55	32	0	0	0	0	0
[I] mr_s (MEM_STAGE)	8736	8192	4352	2112	0	0	0
[I] p1 (IF_ID)	49	70	0	0	0	0	0
[I] p2 (ID_EX)	116	169	0	0	0	0	0
[I] p3 (EX_MEM)	7106	297	0	0	0	0	0
[I] p4 (MEM_WB)	52	71	0	0	0	0	0

Fig: On-Chip Performance with Base Architecture RV32I

Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)	Bonded IOB (200)	BUFGCTRL (32)
▼ N RISC_V_RV32F_PROCESSOR_POWER_OPT	17911	11019	4770	2145	6580	17911	71	1
[I] ex_mem_reg (EX_MEM)	7083	296	0	0	2872	7083	0	0
[I] fp_fwd_mux1 (FP_FORWARDING_MUXES)	42	0	0	0	35	42	0	0
[I] fp_fwd_mux2 (FP_FORWARDING_MUXES_0)	7	0	0	0	3	7	0	0
[I] fp_regfile (FP_REGFILE_POWER_OPT)	296	1024	132	33	330	296	0	0
[I] id_ex_reg (ID_EX)	764	170	0	0	289	764	0	0
[I] if_id_reg (IF_ID)	125	75	0	0	77	125	0	0
> [I] if_stage (INSTRUCTION_FETCH_POWER_OPT)	87	64	0	0	31	87	0	0
> [I] int_execute (EXECUTE_STAGE_POWER_OPT)	0	0	0	0	28	0	0	0
[I] int_fwd_mux1 (FORWARDING_MUXES)	32	0	0	0	25	32	0	0
[I] int_fwd_mux2 (FORWARDING_MUXES_1)	4	0	0	0	4	4	0	0
[I] int_regfile (REGFILE_POWER_OPT)	576	992	256	0	520	576	0	0
[I] mem_stage (MEM_STAGE)	8736	8192	4352	2112	5133	8736	0	0
[I] mem_wb_reg (MEM_WB)	159	71	30	0	94	159	0	0

Fig: Performance with Cognichip-Assisted Optimized RV32IF

- Slice LUTs reduced from ~19,355 to ~17,911
✔ -7.5% reduction in LUT usage
- Logic complexity reduced through RTL refactoring and gating
✔ More efficient datapath and control logic
- Better module-level optimization observed in execute and control paths
✔ Less redundant logic after Cognichip-guided cleanup
- Registers slightly increased due to added FPU control and pipeline tracking
⚠ Acceptable trade-off for added RV32F functionality
- Overall hardware efficiency improved
✔ Fewer LUTs for a more capable (RV32IF) processor
- Performance-per-area improved
✔ More features (FPU + optimizations) with -7-8% less LUT cost

Power Optimization : Baseline vs Cognichip

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power: 0.243 W
Design Power Budget: Not Specified
Process: typical
Power Budget Margin: N/A
Junction Temperature: 27.8°C
Thermal Margin: 57.2°C (4.8 W)
Ambient Temperature: 25.0 °C
Effective θ_{JA} : 11.5°C/W
Power supplied to off-chip devices: 0 W
Confidence level: Medium
[Launch Power Constraint Advisor](#) to find and fix invalid switching activity

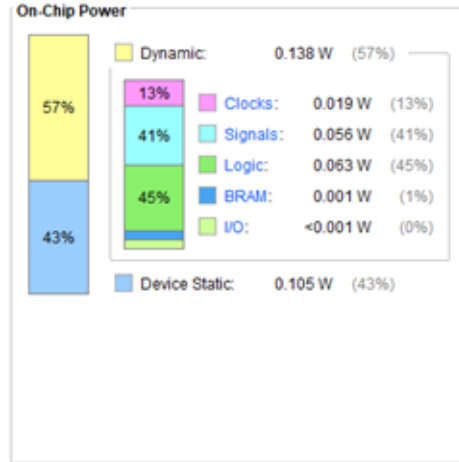
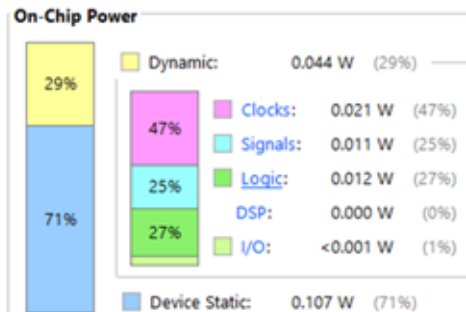


Fig: On-Chip power with Baseline Architecture RV32I

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

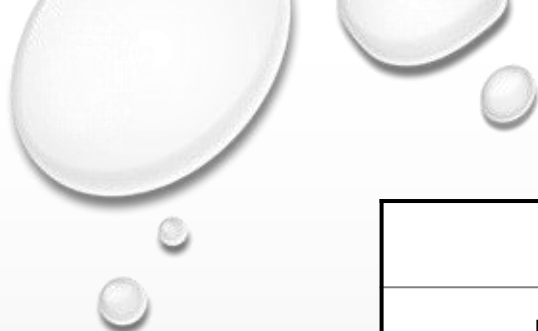
Total On-Chip Power: 0.151 W
Design Power Budget: Not Specified
Process: typical
Power Budget Margin: N/A
Junction Temperature: 26.7°C
Thermal Margin: 58.3°C (4.9 W)
Ambient Temperature: 25.0 °C
Effective θ_{JA} : 11.5°C/W
Power supplied to off-chip devices: 0 W
Confidence level: Medium
[Launch Power Constraint Advisor](#) to find and fix invalid switching activity



- Total on-chip power reduced: $-0.243 \text{ W} \rightarrow -0.151 \text{ W}$ ($\approx 38\%$ reduction)
- Dynamic power significantly lowered using clock gating and enable-based execution
- FP units and execution blocks gated when not in use \rightarrow less switching activity
- Signal and logic power reduced due to operand isolation and control gating
- Overall result: More features (RV32IF + FPU) with lower power than baseline



Fig: On-Chip power with Cognichip-Assisted RV32IF (With Optimized Power Modules)



Pros	Cons (Areas to Improve)
Faster RTL-to-FPGA development	Needs access to previous chat history
Reduces manual coding effort	Need support for uploading images
Enables rapid design iterations	Cognichip terminal needs improvement
Helps generate and refactor RTL	Instruction encoding can be inaccurate for complex ISAs
Supports design space exploration	Better awareness of existing RTL pipelines needed
Improves productivity for complex designs	Power-aware optimizations not always suggested by default
Speeds up prototyping and validation	Verification/testbench generation can be improved

Cognichip: Strengths and Improvement Areas



Conclusion

Baseline RV32I
successfully
extended to RV32IF
using Cognichip.

Functional
correctness verified
using simulation,
VIO, and ILA on
FPGA.

~7.5% LUT reduction
achieved after
Cognichip
optimization.

~38% on-chip power
reduction using clock
gating and RTL
refactoring.

Added floating-point
support with better
performance-per-
area.

Design-to-FPGA
completed within
half a day using
Cognichip.

Demonstrates
Cognichip as an
effective AI co-
designer for
hardware.

Future Scope

- Add PPA prediction model for fast feedback without full synthesis.
- Closed-loop LLM optimization using PPA feedback for better designs.
- Explore more microarchitectures (multi-cycle FPU, shared resources).
- Improve automatic verification and testbench generation.

GitHub Repo

<https://github.com/p20230031/Cognichip-RV32IF-PowerOptimized-RISC-V>



Thank you...