HPC-GENIE Project HP:
https://www.hpc.itc.nagoya-u.ac.jp/menu/hpc_genie.html

From Vibe Coding to Silicon: Local LLM Agents for AI-Assisted Semiconductor Design

Takahiro Katagiri

Information Technology Center, Nagoya University

Workshop on LLM-Driven Code Generation and Automation for HPC and Scientific Computing
Monday, July 6, 2026, 13:30–18:35
LIP6, Sorbonne University, Paris
Session 2: Hardware Design 15:10-15:35

KATAGIRI Takahiro

Professor

Information Technology Center
Graduate School of Informatics
Nagoya University

Ph.D. in Computer Science
The University of Tokyo, 2001

High-Performance Computing, Auto-Tuning, and AI-Driven Scientific Computing

Research profile based on Nagoya University Faculty Profiles

Research Interests

- High Performance Computing
- Software Auto-tuning
- AI Agents
- Quantum Computing
- Massively Parallel Processing
- Large-Scale Eigenproblem

Core Areas

- Software auto-tuning for high-performance computing
- Massively parallel processing and supercomputing
- Large-scale distributed machine learning / AI agents
- HPC-Quantum Computing and post-Moore numerical algorithms
- Parallel programming education and co-design

Selected Recognition

HPC4HPCAsia2026 Best Paper Award · IEEE PDSEC2023 Best Paper Award · LHAM 2018 Workshop Best Paper Award

Source: profs.provost.nagoya-u.ac.jp/html/100009072_en.html | Updated: 2026/06/22

GPT-5.5 generated this slide on July 6, 2026

Agenda

- **Project Overview**: JST Research and Development Program for Next-generation Edge AI Semiconductors
- **Case Study**: Realtime OS development by LLM agents
- **Summary**

Agenda

- **Project Overview**: JST Research and Development Program for Next-generation Edge AI Semiconductors
- **Case Study**: Realtime OS development by LLM agents
- **Summary**

Financial support:
JST Research and Development Program for Next-generation Edge AI Semiconductors

Creating a Next-Generation High-Efficiency Design Platform for Edge AI Semiconductors Assisted by Local LLMs

R&D Principal Investigator: Toru Ishihara (Nagoya University)

R&D GL:	**Takahiro Katagiri (Nagoya University)**
R&D GL:	Tetsuya Iizuka (The University of Tokyo)
R&D GL:	Hiromitsu Awano (Nagoya University)
R&D GL:	Akihiko Shinya (NTT)
Sub-GL (Ishihara G):	Shinya Honda (Nagoya University)
Sub-GL (Ishihara G):	Jun Shiomi (Osaka University)
Sub-GL (Awano G):	Yusuke Sakemi (Chiba Institute of Technology)
Sub-GL (Awano G):	Shinichi Nishizawa (Hiroshima University)

JST Next-Generation Edge AI Semiconductor Research and Development Project (2025.12-)

Mission of This Project: Building Next-Generation Domestic EDA Technology and Research Organization

R&D Principal Investigator: Prof. Toru Ishihara (Nagoya University)

Period: **2025.12-2030.11**

Total Budget: **Maimum. About 10.37 million US dollars　(16.5億円)**

8 Groups

テーマ	戦略	課題	技術領域	目標・応用	代表者	研究開発課題名
①設計	ASIC	自動設計	システム	ユースケース創出	川原圭博 (東京大学)	ユースケース駆動による機能分化型フィジカルAIチップ設計
			アーキテクチャ アルゴリズム	サイエンス貢献	泰地真弘人 (理化学研究所)	AI for Scienceのためのエッジの知能化加速
			デジタル メモリ	ロボット	小菅敦丈 (東京大学)	AIによるAIのためのAI回路設計自動化技術
			アナログ	センサ信号処理 無線接続	岡田健一 (東京科学大学)	アナデジ混載型エッジAI SoC設計技術の研究開発
			EDA	EDA用LLM	石原亨 (名古屋大学)	ローカルLLM支援によるエッジAI半導体の次世代高効率設計基盤の創出
②実装	3D	抜熱	チップレット	インターポーザあり	田中徹 (東北大学)	エッジAI半導体を実現する3Dヘテロ集積技術
			3D実装	インターポーザなし	井上史大 (横浜国立大学)	環境循環型3D集積半導体製造革新と拠点形成
③デバイス	CFET 配線	構造・材料 低抵抗化	トランジスタ 配線	A5世代以降	多田宗弘 (慶應義塾大学)	新構造・新材料トランジスタと低抵抗配線の研究開発

Issue

Design

Automated Design

LLM for EDA

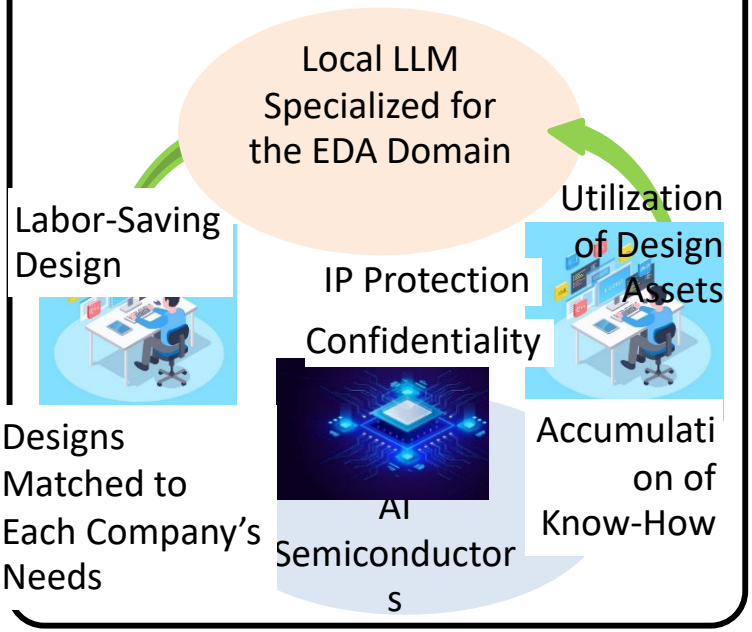
Implementation

Device

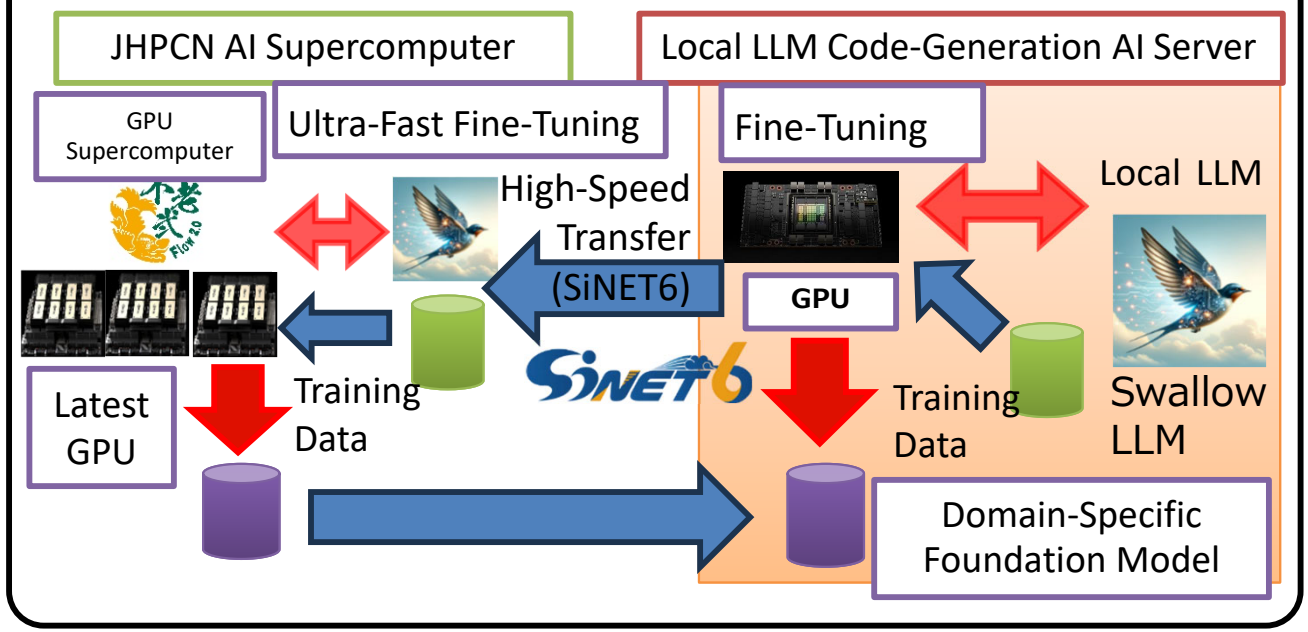
Local LLMs and Their Tuning Technology

- ✓A **local LLM** is a system that runs an LLM in a local environment, such as a **company's on-premises server**, rather than in the cloud. It is effective when **IP protection and strict NDA compliance** are required.
- ✓Tuning technology for EDA local LLMs and multi-AI-agent technology are key.

Development of Local LLMs for EDA



Development of Multi-AI Agents and Their Use in EDA (Katagiri G)



VibeCodeHPC - Multi Agentic Vibe Coding for HPC

https://github.com/Katagiri-Hoshino-Lab/VibeCodeHPC-jp Publicly available

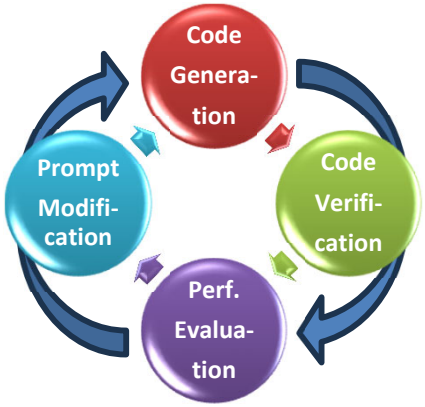
VibeCodeHPC is a multi-agent system that fully automates environment setup and code optimization for HPC. Multiple AI agents collaborate through tmux-based communication in CLI environments such as Claude Code.

⇒ Extend code-generation AI agents to semiconductor circuit design

Publicly Available VibeCodeHPC Agents (Multi-Agent LLM)

Agent	Role	Responsibility
PM (Project Manager)	Project Management	Requirements definition, resource allocation, budget management
SE (System Engineer)	System Design	Agent monitoring, statistical analysis, report generation
PG (Programmer)	Code Generation & Execution	Parallelization implementation, SSH/SFTP connection, job execution, performance measurement, SOTA judgment
CD (Continuous Deployment)	Deployment Management	Publication and anonymization of SOTA-achieving code

Utilizing the Latest AI Agent Technology for Code Generation

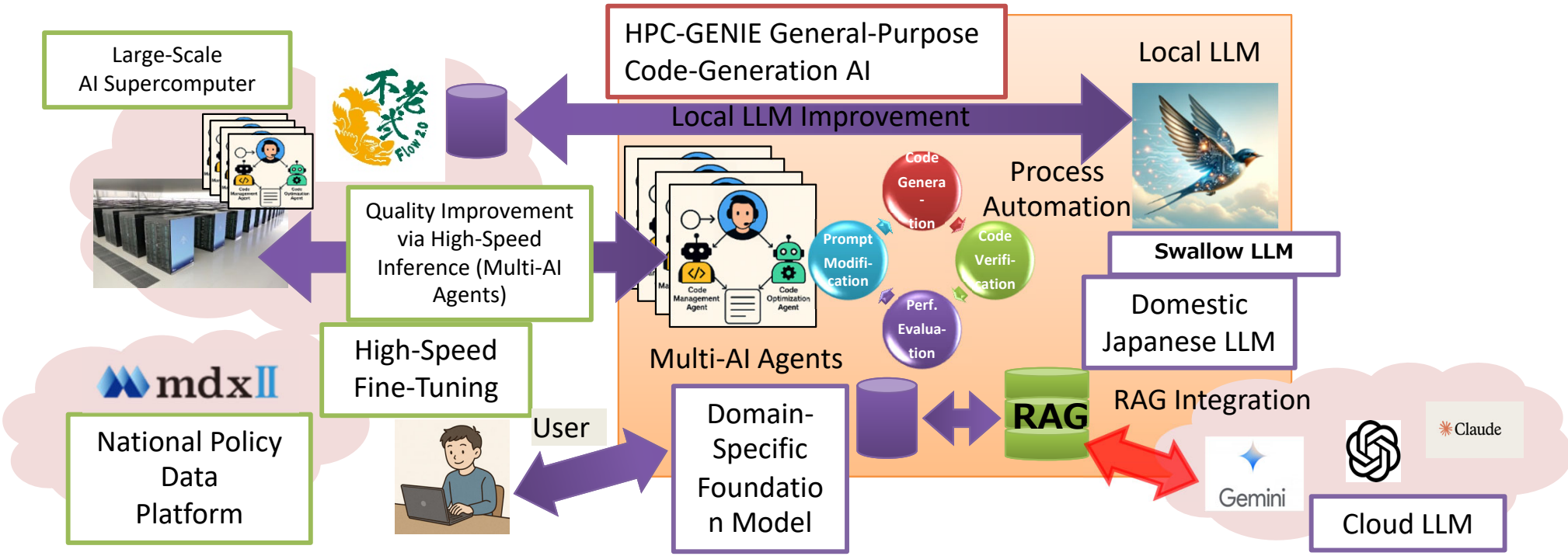


Example Configuration of Circuit-Design Agents (**Multi-Agent LLM**) Developed in This Project

Agent	Role	Responsibility
Circuit Design PM	Circuit-design project management	Requirements definition, resource allocation, budget management
Circuit Design PG	Circuit description generation & execution	Code generation for Verilog, SPICE netlists, etc.
Circuit Quality Audit	Functional verification & performance evaluation	Functional verification, PPA evaluation, inference-accuracy estimation and evaluation
Circuit Design CD	Prompt management and others	Prompt engineering and others

Development of Text-Generation Technology Using Multi-AI Agents

- **Process automation**: iterative prompting for code generation → generated-code verification → performance evaluation → prompt revision
  - ✓ Improving code quality with multi-AI agents (A2A)
- RAG to strengthen local LLMs and **high-speed fine-tuning** using **supercomputers**.
- Improving generated-code quality through **high-speed inference** (multi-AI-agent execution) on **supercomputers**.

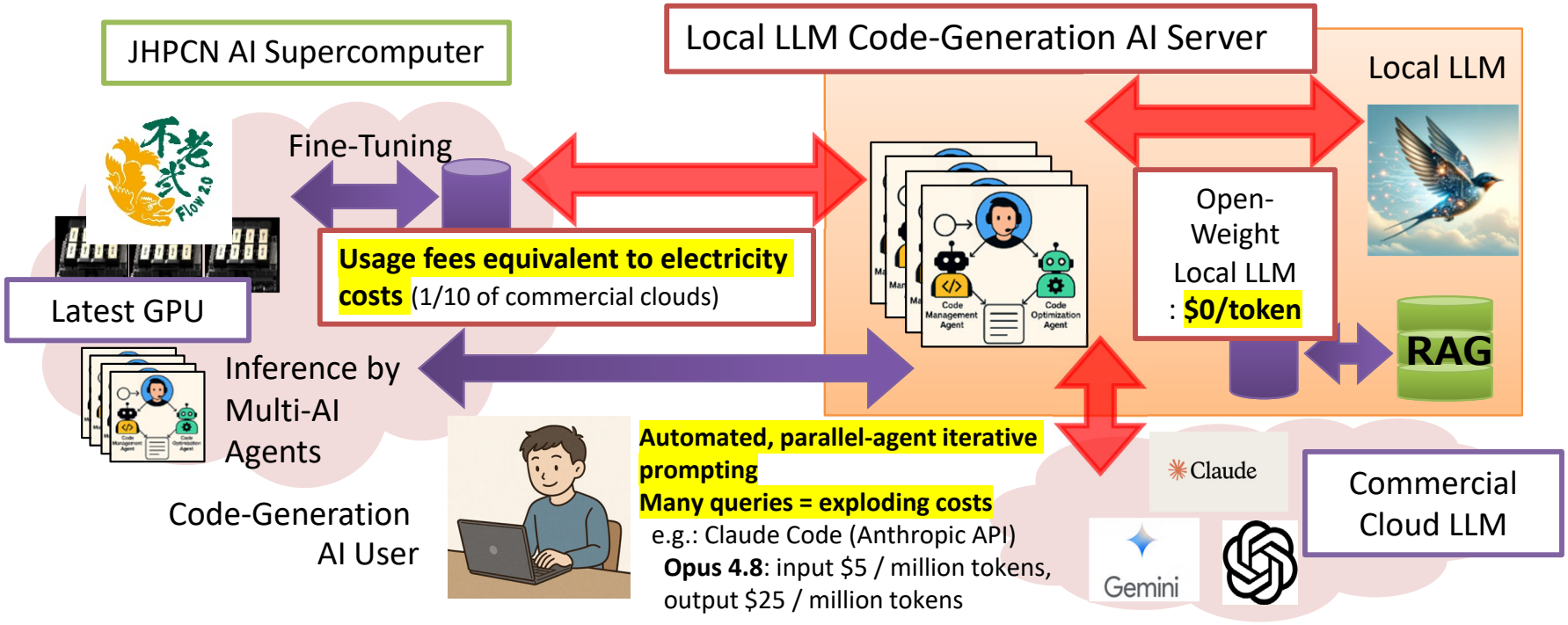


Deployment of Nagoya University's Next-Generation Supercomputer "Flow 2.0"

Goal: Providing academia with a **low-cost, high-performance code-generation AI** environment

- Challenge with cloud LLMs: charges for fine-tuning and parallel inference
  - **Fine-tuning** and parallel inference require many latest GPUs
  - Usage fees for the JHPCN AI supercomputer are **equivalent to electricity costs** (one-tenth of commercial clouds)

→ Nagoya University ITC's next supercomputer "Flow 2.0": planned as a new service starting operation in October 2026



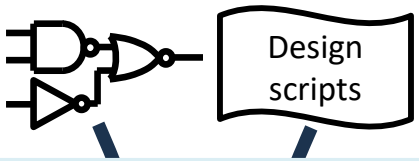
Local LLMs and Their Customization Technologies

- ✓ **A local LLM** is a mechanism for running an LLM in a local environment, such as a **company's on-premises server**, rather than in the cloud. It requires **no usage fees** and is effective for **IP protection and strict NDA compliance**.
- ✓ **Another goal is to build national capability in R&D for proprietary local LLMs and AI systems.**
- ✓ **As academia, we also aim to train students and young researchers capable of developing proprietary local LLMs and AI systems.**

*Generative AI system = LLM + Harness
Harness = the control layer that launches and runs the LLM

Use and training of generative AI systems

Benchmark creation



Use OSS frameworks such as Hugging Face and MS-Swift

Generative AI system specialized for the EDA domain*

Database for EDA-specialized local LLMs and Harness

Collaboration

***Technology for tuning generative AI systems**

Develop reinforcement-learning(RL)-based tuning methods

Use the domestically developed Swallow-LLM as a foundation model
(also use local LLMs such as Gemma and Qwen)

R. Mikasa, S. Hayashi, D. Mukunoki, T. Hoshino, T. Katagiri, "**Improving HPC Code Generation Capability of LLMs via Online Reinforcement Learning with Real-Machine Benchmark Rewards**," Proc. **ISC26 Workshop** (**LLM4HPC**), June 2026.

Deploy LAD trials within the team

Ishihara & Honda Groups: (a) LAD technology for large-scale digital design
Awano & Iizuka Groups: (b) LAD technology for medium-scale mixed-signal design
Shinke & Sawada Groups: (c) LAD technology for small-scale photonic-electronic fusion AI circuits

Strategy for Contributing to the Revival of Japan's Semiconductor Industry

Collaboration with system industries that create end products is key to reviving the semiconductor industry

System industries

Embedded systems industries
[automobiles, robots, games, etc.]

Generative AI (LLM) systems industries
[LAD, internal operations, call centers, etc.]

Collaboration with KMC, NEC, etc.

Nationwide deployment through "Flow2. 0" Type II

✓ **Embedded Consortium**

- **Explore LLM-assisted hardware/software co-development**
- Research and develop LAD technologies for embedded microcontrollers with embedded NPUs
- Collaborate with Jedat on LAD technologies

NVIDIA has an extremely high share in high-performance AI semiconductors

⇒ A security crisis affecting all industries

✓ **LLM customization**

- Promote research using **confidential data and development of local LLMs**
- Promote LAD technology research and **specialization of local LLMs for EDA design**

Chip-design and foundry industries

Fabless chip vendors
[NVIDIA, Broadcom, Renesas, etc.]

Foundries
[TSMC, Samsung, Rapidus, etc.]

Agenda

- **Project Overview**: JST Research and Development Program for Next-generation Edge AI Semiconductors
- **Case Study**: Realtime OS development by LLM agents
- **Summary**

TOPPERS Project Overview

Toyohashi OPen Platform for Embedded Real-time Systems

Japan-born Embedded RTOS

https://www.toppers.jp/project.html

Professor Honda (Nagoya University)

In brief — A practical, Japan-born **open-source embedded RTOS (Real-Time OS)** project rooted in the ITRON tradition

Project positioning

- Operated by the TOPPERS Project NPO
- **Developed through industry–academia–government and individual collaboration**
- Publishes foundational software for embedded systems
- Also emphasizes engineer training and educational materials

Main deliverables

- **Real-time OS kernels: ASP3, HRP3**, etc.
- **Multiprocessor kernels: FMP3, HRMP3**
- Middleware such as TECS, TCP/IP, CAN/LIN
- Simulation environments, specifications, and teaching materials

Suitable use cases

- Control systems requiring real-time performance and reliability
- Industrial equipment, automotive, robotics, and IoT devices
- Small to mid-size embedded software, from tens of KB to about 1 MB
- From research and education to commercial use

Goals — **Promote the definitive ITRON-spec OS, advance next-generation RTOS technology, provide development support tools, and cultivate talent through education.**

License: proprietary TOPPERS License designed with commercial embedded use in mind.

Recent updates — Updates include the 3rd-generation kernel specifications and ASP3/HRP3/HRMP3 releases, plus newsletters and application development contest information.

Sources: Official TOPPERS website (project / kernels / license / news)

Materials provided by Professor Honda (Nagoya University)

Use of AI in TOPPERS Development: OS Kernel Porting (1)

Porting the **TOPPERS/FMP3** kernel to the **Raspberry Pi Pico 2**.

※**TOPPERS/ASP3 OS kernel**

Case 2 Creating ASP3 for PICO2 Based on FMP3 for PICO2

Model
Claude Code Sonnet 4.6

Time required
About 2.5 hours

Details

- Wrote the goal and execution method in CLAUDE.md
- After initialization, it mostly ran autonomously
- Confirmed that the OS tool loop worked effectively

By clarifying the policy with CLAUDE.md + init, development was completed almost autonomously

Materials provided by Professor Honda (Nagoya University)

Use of AI in TOPPERS Development: OS Kernel Porting (2)

※Porting TOPPERS/ATK2, an AUTOSAR-based real-time OS kernel, to run on the Renesas RA6M5 microcontroller / EK-RA6M5 evaluation board.

Case 3 Porting ATK2 to Renesas RA6M5

Same core, using code generated by Renesas Smart Configurator

Model
Claude Code Opus-4.7

Repository
github.com/exshonda/atk2_sc1_shin

Development

- Proposed and carried out development phases
- Once taught how to use the debugger, **it began debugging** (checked device-register values and understood/responded to TrustZone)
- **Proposed tests itself** (running for over 30 minutes)

Materials provided by Professor Honda (Nagoya University)

Use of AI in TOPPERS Development: OS Kernel Porting (2)

Case 3 (cont.) Findings -- Debugging Lessons

Findings

- Indifferent to source-code layout (**auto-generated code should be kept in a separate folder**)
- **Struggles when primary information is wrong** (does not doubt it)
  - **Ambiguous information about the board UART port was provided, causing delays**

Debugging Struggles -> Resolution

- **Because the startup mode differed from STM32, the error was mistaken** for a different one and debugging stalled
- **Analysis began** after the debugger stopped following a double fault, so it was misdiagnosed as **a stack overflow** (same trap as past ASP3 ports)
- **Solved immediately after instructing it to break** in the exception handler

☛Providing export knowledge in this domain by human.

Document all knowledge gained by AI -> the same work goes smoothly next time (stored in docs/)

☛The knowledge is reusable by in-context learning.

Materials provided by Professor Honda (Nagoya University)

Updating Legacy OS Kernel

Case 1 Bringing Topper OS kernel ATK2 Up to Third-Generation Level

Model
Claude Code Sonnet 4.6 / Opus-4-7
Used after upgrading the subscription plan

Time required
About 2.5 hours

Repository: github.com/exshonda/atk2_sc1_shin

Development Details

- **Targeted Nucleo H563ZI for ATK2** (also developed architecture-dependent parts; provided corresponding ASP3 code)
- **Converted binary cfg to Python**: provided C++ source (also showed the earlier ASP3 Python **cfg(configurator) version**)
- **Made the Makefile TOPPER's ASP3-equivalent**: enabled parallel compilation using .d files (dependency file)

In TOPPERS/ASP3, FMP3, ATK2, and similar systems**, tasks, semaphores, event flags, cyclic handlers, interrupt handlers, and other objects used by the application are not all written directly by hand in C code**. Instead, they are described in configuration files. **The tool that reads these settings and generates the C source files and headers used by the kernel is called cfg**.

Materials provided by Professor Honda (Nagoya University)

Updating Legacy OS Kernel

Case 1 (cont.) Splitting Development Phases and Verification

Finding: AI proposed development phases, and each phase was run in a separate session

Phase 1
Port the skeleton
Done in this session

→

Phase 2
pass1: ARXML -> cfg1_out.c
Separate session

→

Phase 3
.tf parser + macro engine
Separate session

→

Phase 4
Integrate pass2 / pass3
Separate session

☑ **Verification: Generated test programs and compared results with cfg.exe**

For large ports, split into phases and separate sessions -> verify quality through tool loops

Updating Legacy OS Kernel: TTSP3 (TOPPER's OS Kernel)

TTSP3 and Its Current Status

- **9,000 API tests** for third-generation OSs
- **Generates test code from abstract YAML notation**
  - –Multiple test cases can be merged
- Development stopped after 2019
  - –Large scale (ASP3 / FMP3 / HRP3 / HRMP3)
  - –The integrated specification version was updated

```
version: "$Id: act_tsk_a.yaml 64 2019-12-20
01:33:52Z fujisft-shigihara $"
ASP_task_manage_act_tsk_a:
  note:  【NGKI1114】
  pre_condition:
    TASK1:
      type    : TASK
      tskstat: running
    TASK2:
      type    : TASK
      tskstat: dormant
    CPU_STATE:
      type    : CPU_STATE
      loc_cpu: true
  do:
    id      : TASK1
    syscall: act_tsk(TASK2)
    ercd    : E_CTX
  post_condition:
```

Test_x000B_Program

Before developing asp3_core, described later, we cleaned up the stalled TTSP3

Updating Legacy OS Kernel : TTSP3

Method -- Consult AI and Divide Work Between AI and Humans

Discussed with AI by chat and split tasks between AI and humans

AI handled Task

- **Self-verification loop**: make it executable and judge pass/fail from execute.log
- **Reference patterns**: prepare results that have passed once on standard ASP3
- **These two are needed for autonomous work**

Humans handled Task

- **Judging specification differences**
- **Policy handling and decision-making**

The key is for humans to prepare the foundation that lets AI run autonomously: a self-verification loop plus reference patterns

Analyzing Debugging Methods for Complex SoCs

ROM Code → Firmware (TF-A) → Bootloader (U-Boot) → Linux Kernel

SoC Boot Challenges and Assumptions

- **Complex boot** involving secure/non-secure state, DDR, and cache initialization
- **Bare-metal development is not assumed** (Linux running is enough / **manuals are not public**)
- **Trial and error is essential** from debugger connection to stable operation

AI Benefits

- **Patiently tries many possibilities**
- Stops at specified points during boot and checks logs
- Analyzes the hardware initialization order

Human Tasks

- **Provide related information and past findings**
- Unplug/replug USB and turn power ON/OFF

☛In-context learning is sufficient; fine-tuning is not necessary.

Lessons on Using Generative AI Effectively

Inputting Prior Knowledge

- Reading PDFs directly is inefficient (takes time) -> use another agent to summarize and file by topic
- Recording pitfalls improves efficiency (tell it to "record findings" when converging or context is full)
- Samples work better than specifications (programs over natural language) -> API documentation
- For TTSP3 and asp3_core, this operation became a loop -> automatic documentation during horizontal rollout -> Development documentation

Creating a Development Plan Is Important

- Smart LLMs can run autonomously, but giving some **upfront structure** makes the work smoother

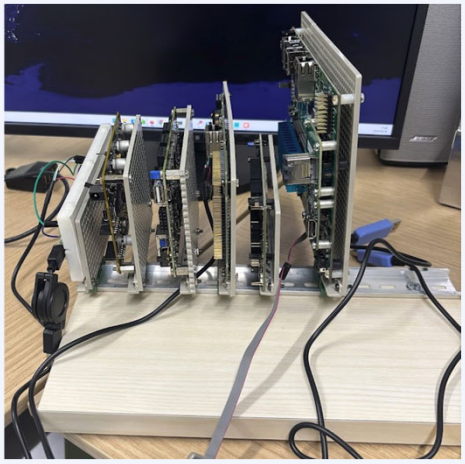
Lessons on Using Generative AI Effectively

AI-Only Tool Loops Are Important

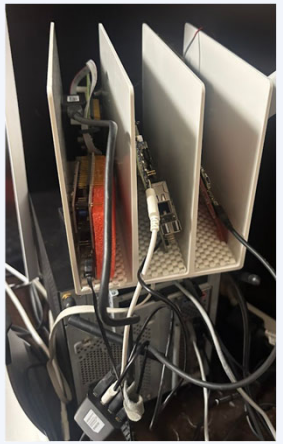
- Efficiency -- **watch out for occasional loops** over trivial issues
- **Use simulation environments** to speed development (they do not hang!)
- For asp3_core, first prepare QEMU for all architecture-dependent parts
- **Environments without a CLI cannot be automated** by AI and may no longer be chosen in the future

Physical Development Environment

- **High-density board setup**
- Multiple USB connections are not very stable
- Boards that cannot be reset via JTAG are troublesome
- **Handle cases requiring USB unplug/replug and power ON/OFF**



DIN Rail



Book Stand

Agenda

- **Project Overview**: JST Research and Development Program for Next-generation Edge AI Semiconductors
- **Case Study**: Realtime OS development by LLM agents
- **Summary**

Summary: From Vibe Coding to Silicon

Local LLM agents for AI-assisted semiconductor and embedded-system design

Core message

The project aims to rebuild Japan's EDA capability by combining local, customizable LLMs, multi-agent automation, and HPC infrastructure to accelerate secure semiconductor design.

1. Project platform

A JST-backed 2025–2030 effort links EDA, embedded systems, local LLMs, HPC, **mixed-signal design**, **photonic-electronic AI circuits**, and combinatorial optimization into a next-generation design platform.

2. Agentic design flow

VibeCodeHPC-style agents are **extended from HPC code generation to circuit design**: project management, code/RTL or netlist generation, verification, PPA evaluation, and prompt refinement operate in tool loops.

3. RTOS case study

TOPPERS/ASP3 work shows practical wins: **AI agents ported and modernized RTOS** components, created QEMU/CI-based verification loops, and improved debugging when humans supplied reliable constraints and records.

Strategic takeaway

Local LLMs are the strategic foundation for trustworthy LAD (LLM-Aided Design) systems: by combining high-quality local data, documented design know-how, self-verification loops, and low-cost GPU access via the supercomputer “Flow 2.0”. We can **protect IP**, **customize models** for each organization, and scale AI-assisted development from embedded software to silicon design.