Integrated AI and Wireless for Sustainable Mobile Network Evolutions

--- An Open-Source RISC-V Domain Specific Architecture Approach

Presenter: Zhiyuan Jiang

Contributor: Limin Jiang, Yi Shi, Yintao Liu, Qingyu Deng, Siyu Xu, Yihao Shen, Feng Yuan, Si Wang, Bo Ruan, Haiqin Hu, Meiling Yang, Shan Cao, Ting Zhou and Sheng Zhou

2025.7.8@Geneva

先进通信与计算芯片实验室
Advanced Communication and Computing Electronics Lab

Our Team

ACE-Lab

Faculty Members:

Zhiyuan Jiang
Professor@SHU

Shan Cao
Associate Prof.@SHU

Zeyu Hu
Lecturer@SHU

Ting Zhou
Professor@SHU

Sheng Zhou
Associate Prof.@THU

Our Team

Advanced Communication and Computing Electronics Lab (ACE-Lab) was founded in 2019 at Shanghai University in Shanghai, China.

深轶科技 DEEP TRANSCEND Deep Transcend Ltd., founded in Shanghai in 2023, is a start-up with 6 full-time employees.

Outline

- Background & Motivation
- Related Works
- Echo: An Open-Source 5G/4G/GNSS/LoRa/AI Library
- Venus: A Multi-Core Dataflow-Driven RISC-V Domain Specific Architecture and Implementation on 40nm CMOS
- Zoozve: A Strip-Mining-Free RISC-V Vector Extension Compiler
- Conclusion

Mobile Network Evolution

1G (1980s) —— Analog communication (Ericsson & Motorola)

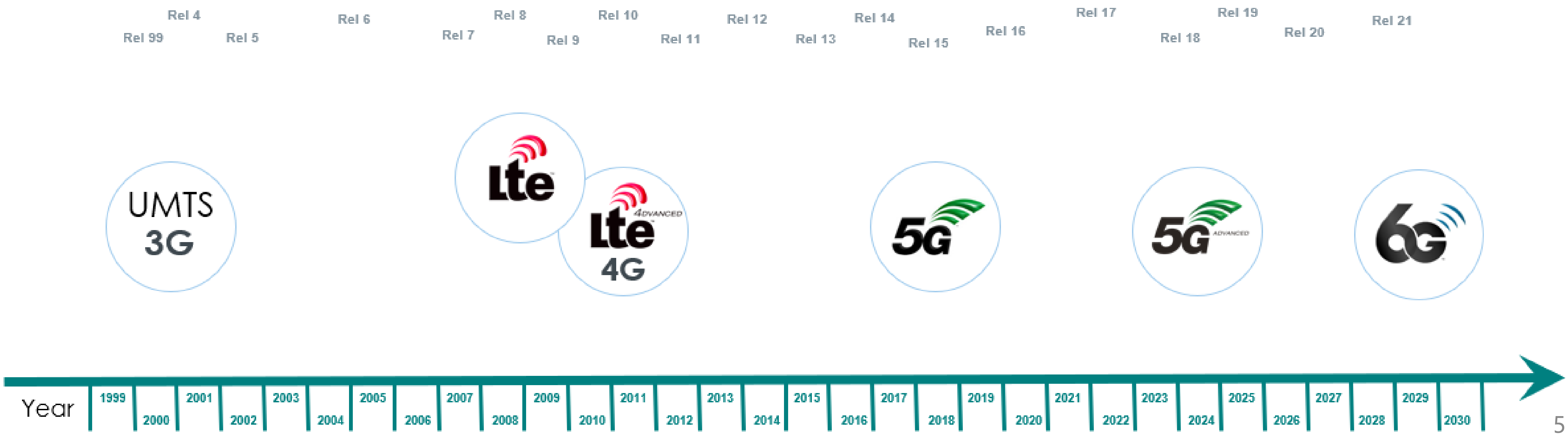
2G (1990s) —— Digital communication (Nokia)

3G (2000s) —— CDMA

4G (2010s) —— OFDM and MIMO → 5G (2020s) → 6G (2030s)

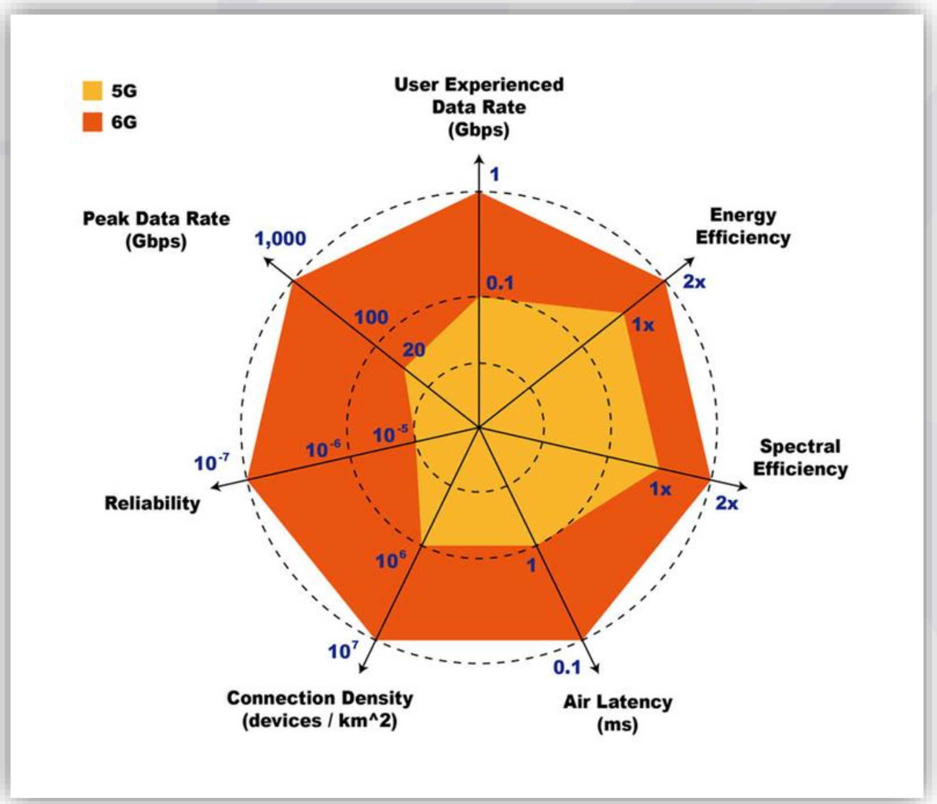
3GPP Releases and Generations

Dates indicated reflect when the work is maturing in the groups. It does not indicate the release's freeze date or the finish of work.

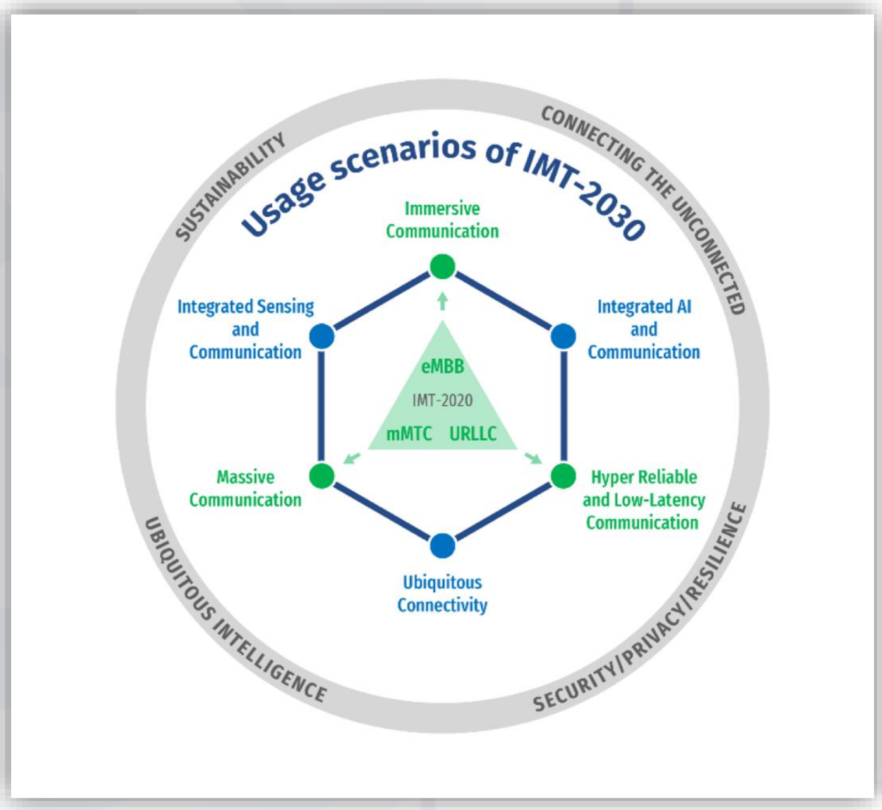


5G & 6G

- From Performance Improvement to Scenario Driven
- From IoT (Internet of Everything) to IIoT (Intelligent Internet of Things)



Enhancements of Communication Performance

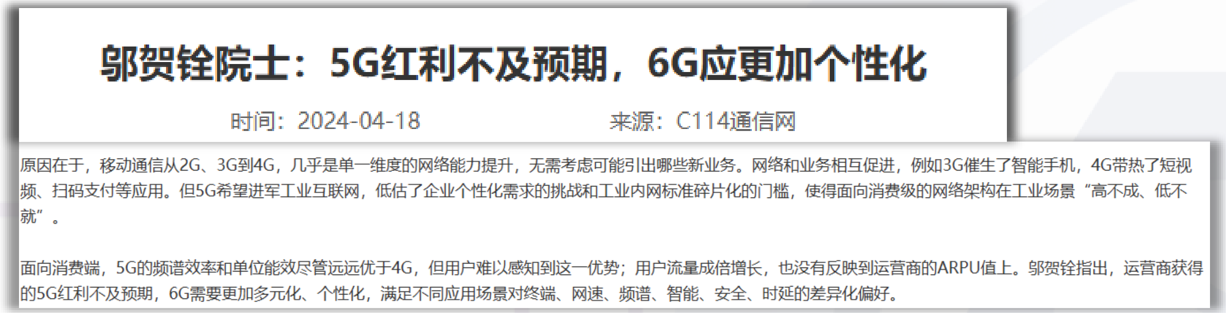


Usage scenarios of IMT-2030
SUSTAINABILITY
CONNECTING THE UNCONNECTED
UBIQUITOUS INTELLIGENCE
SECURITY/PRIVACY/RESILIENCE
Immersive Communication
Integrated Sensing and Communication
Integrated AI and Communication
eMBB
IMT-2020
mMTC
URLLC
Massive Communication
Hyper Reliable and Low-Latency Communication
Ubiquitous Connectivity

Increase in application scenarios

Sustainable 6G Evolution Dilemma

From Users: 5G is not good enough

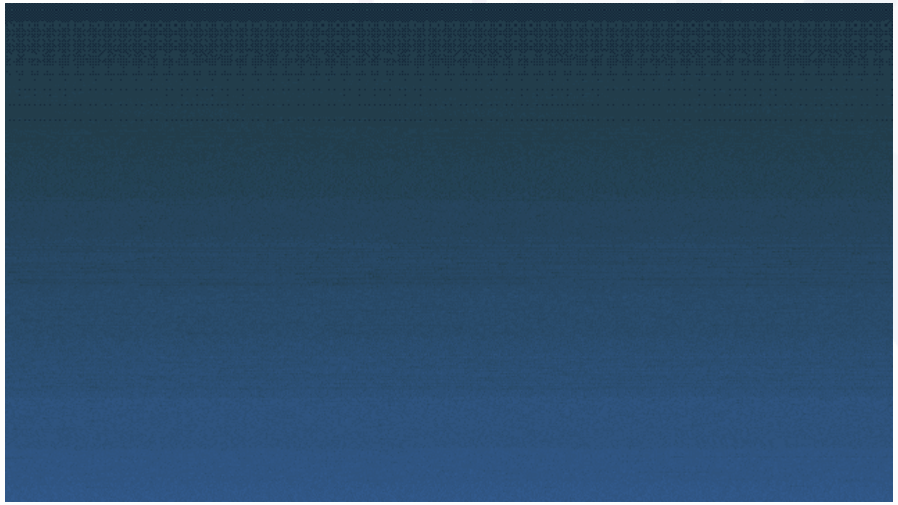


邬贺铨院士：5G红利不及预期，6G应更加个性化

时间：2024-04-18　　来源：C114通信网

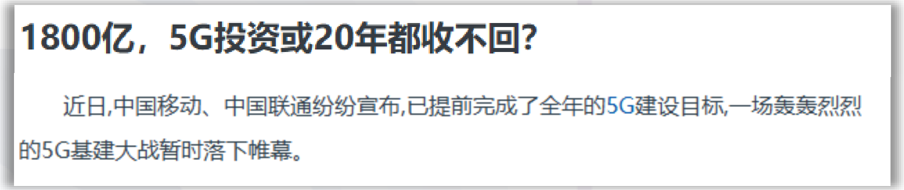
原因在于，移动通信从2G、3G到4G，几乎是单一维度的网络能力提升，无需考虑可能引出哪些新业务。网络和业务相互促进，例如3G催生了智能手机，4G带热了短视频、扫码支付等应用。但5G希望进军工业互联网，低估了企业个性化需求的挑战和工业内网标准碎片化的门槛，使得面向消费级的网络架构在工业场景“高不成、低不就”。

面向消费端，5G的频谱效率和单位能效尽管远远优于4G，但用户难以感知到这一优势；用户流量成倍增长，也没有反映到运营商的ARPU值上。邬贺铨指出，运营商获得的5G红利不及预期，6G需要更加多元化、个性化，满足不同应用场景对终端、网速、频谱、智能、安全、时延的差异化偏好。



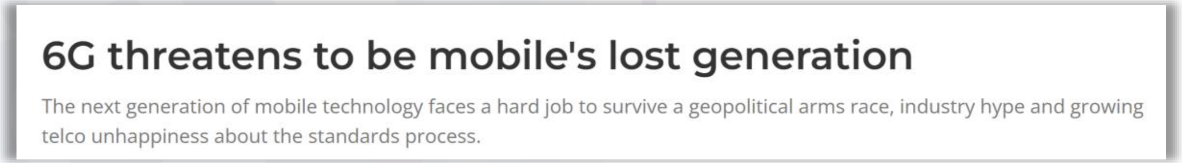
Without killer application, not app-driven

From Operators: Not economically sustainable



1800亿，5G投资或20年都收不回?

近日,中国移动、中国联通纷纷宣布,已提前完成了全年的5G建设目标,一场轰轰烈烈的5G基建大战暂时落下帷幕。



6G threatens to be mobile's lost generation

The next generation of mobile technology faces a hard job to survive a geopolitical arms race, industry hype and growing telco unhappiness about the standards process.

Vodafone Group's Network Strategy Director Santiago Tenorio said in an interview: "Nobody needs 6G. The industry should make 6G a 'No-G'." BT Group's Chief Architect Neil McRae also said in 2021: "I hope that 5G will be the best and the last generation of mobile communication technology in history. I hope we don't need 6G."

6G with Native AI

Main Features:

- **Native AI：** AI will be both a service and a native feature in the 6G communication system, and 6G will be an E2E system that supports AI-based services and applications.
- **Networked sensing (ISAC)**
- **Extreme Connectivity**
- **Integrated NTN：** Ubiquitous Connectivity
- **Native Trustworthiness**
- **Sustainability**

  The potential technologies to realize energy efficiency span architectures, materials, hardware components, algorithms, software, and protocols.

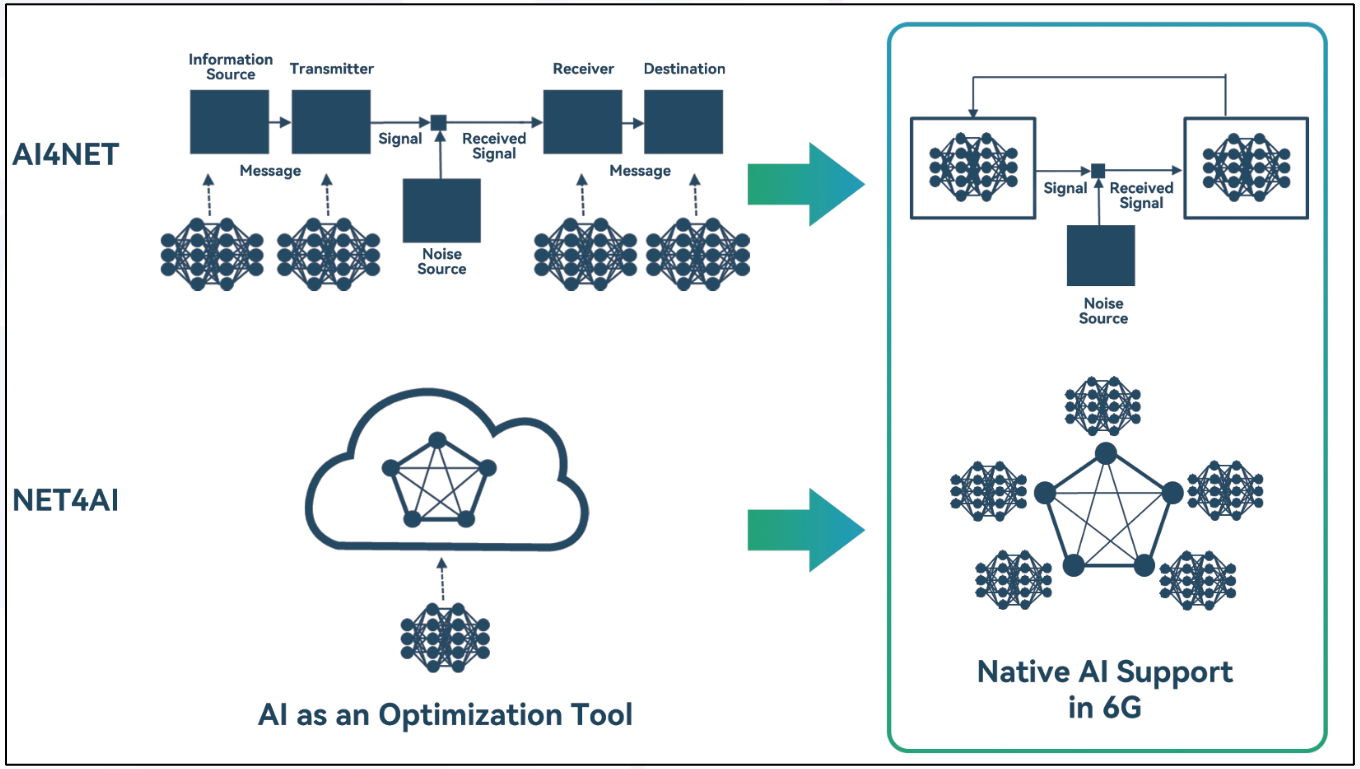
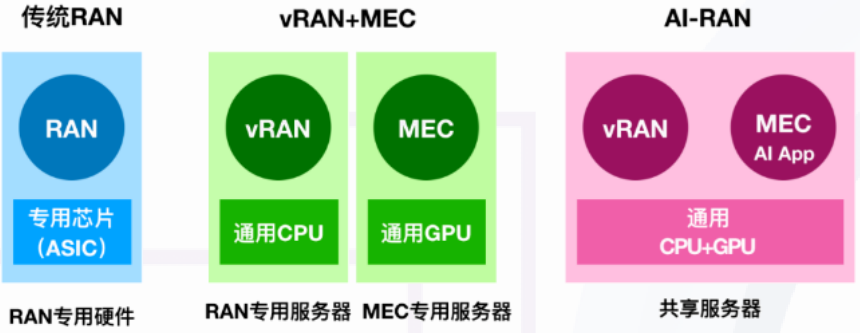


Figure: AI for Network & Network for AI

AI-RAN Alliance



传统RAN

RAN

专用芯片（ASIC）

RAN专用硬件

vRAN+MEC

vRAN

通用CPU

MEC

通用GPU

RAN专用服务器 MEC专用服务器

AI-RAN

vRAN

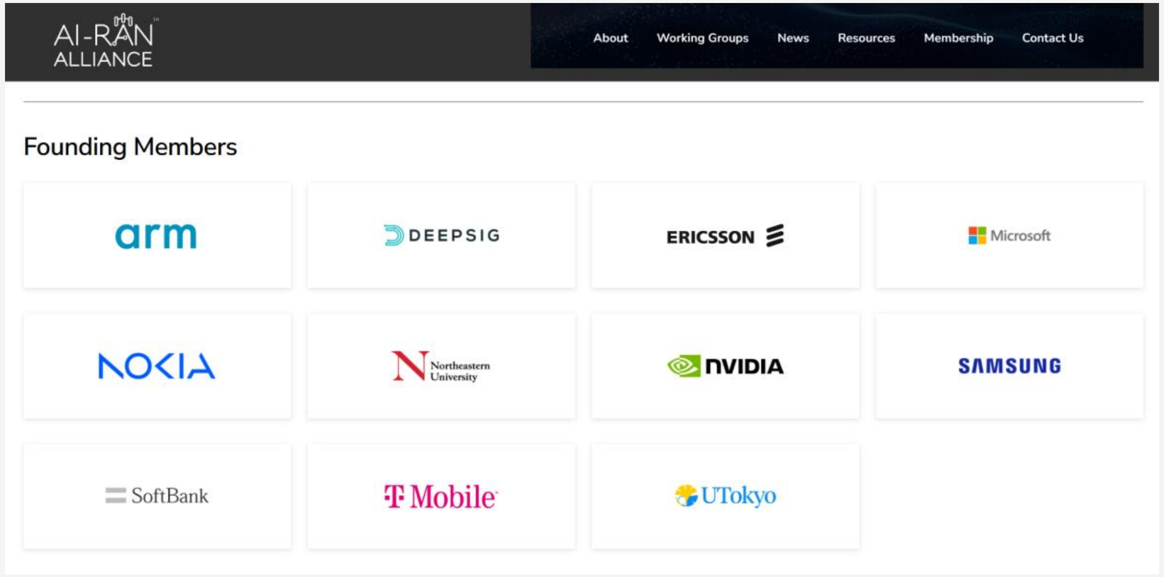
MEC AI App

通用 CPU+GPU

共享服务器

AI-RAN is an architecture that integrates AI and RAN. It uses GPUs for baseband signal processing and AI computing, thereby disrupting the traditional architecture as well as the O-RAN architecture.

AI-RAN Alliance was established on Feb. 2024



AI-RAN ALLIANCE

About | Working Groups | News | Resources | Membership | Contact Us

Founding Members

arm

DEEPSIG

ERICSSON

Microsoft

NOKIA

Northeastern University

NVIDIA

SAMSUNG

SoftBank

T Mobile

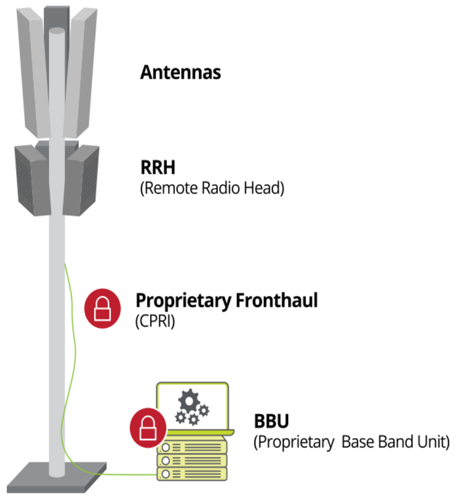
UTokyo

"We will install AITRAS in nearly 200,000 base stations across the country and fully rebuild the communication network using AI-RAN technology." (Nvidia and SoftBank)

Traditional RAN Architecture

Traditional RAN:

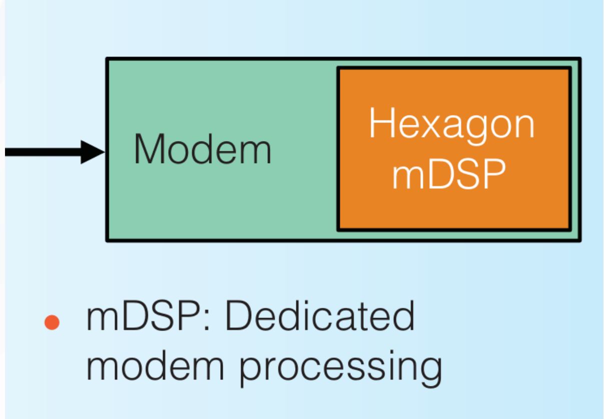
- ➢ Protocol stack that runs on proprietary hardware.
- ➢ Radio Unit and BBU are connected via proprietary interfaces.
- ➢ Single vendor provides both Radio Unit and BBU.



Traditional RAN structure

Traditional Baseband：（DSP and ASIC modem)

- ➢ The core processing unit of the baseband is solidified by hardware.
- ➢ Closed Ecosystem: Neither the software nor hardware implementations of the baseband are open.



- mDSP: Dedicated modem processing

Qualcomm’s baseband

Background Summary

Industry

- ➢ Economically unsustainable 6G
- ➢ Operators is vendor-locked
- ➢ limited flexibility / scalability in RAN

High upgrade and maintenance costs

AI-Driven

- ➢ Native-AI mobile network
- ➢ Hardware choice is limited
- ➢ lack open-source platforms similar with AI community

Hinder hardware efficiency and cross-domain innovation

Academia

- ➢ lack open-source reference designs and collaborative testbeds
- ➢ Long time-to-market for new technology

Hard to integrate cutting-edge research and industry feedback

Sustainable mobile network evolution is a major challenge !

Open-source, Decoupled Software and Hardware on Integrated AI and Wireless Chipsets

Outline

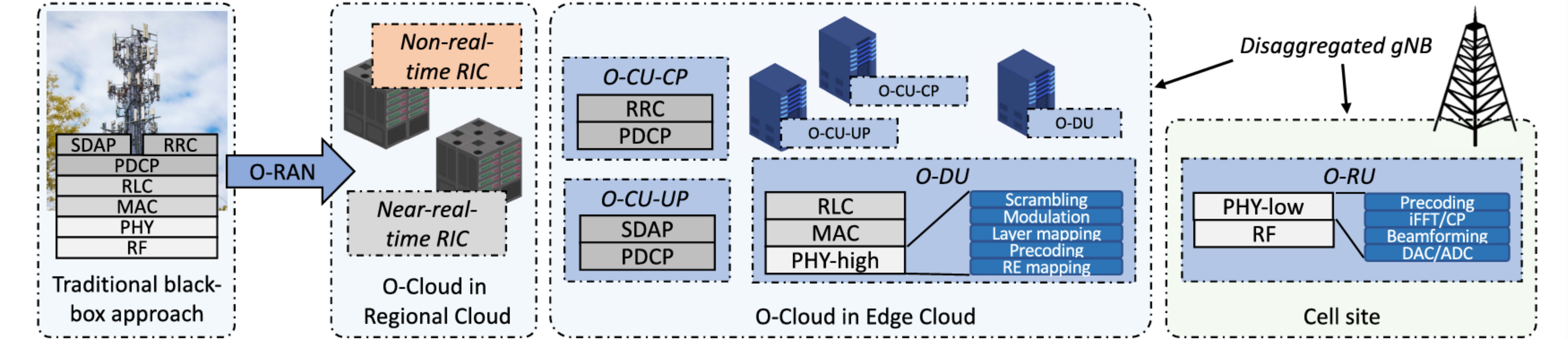
ACE-Lab

- Background & Motivation
- **Related Works**
- Echo: An Open-Source 5G/4G/GNSS/LoRa/AI Library
- Venus: A Multi-Core Dataflow-Driven RISC-V Domain Specific Architecture and Implementation on 40nm CMOS
- Zoozve: A Strip-Mining-Free RISC-V Vector Extension Compiler
- Conclusion

Open RAN / V-RAN

ACE-Lab

Evolution of the traditional black-box base station architecture toward a virtualized gNB with a functional split



CU: RRC, SDAP, and PDCP **DU:** RLC, MAC, frequency-domain functionalities **RU:** time-domain functionalities

Features:

Decoupling of hardware and software

- The non-real-time application software of the access network is separated from dedicated hardware and runs on general-purpose x86 servers.

Unified interfaces

- Standardized set of communication protocols and data models that allow for seamless interoperability between the various components of a RAN.

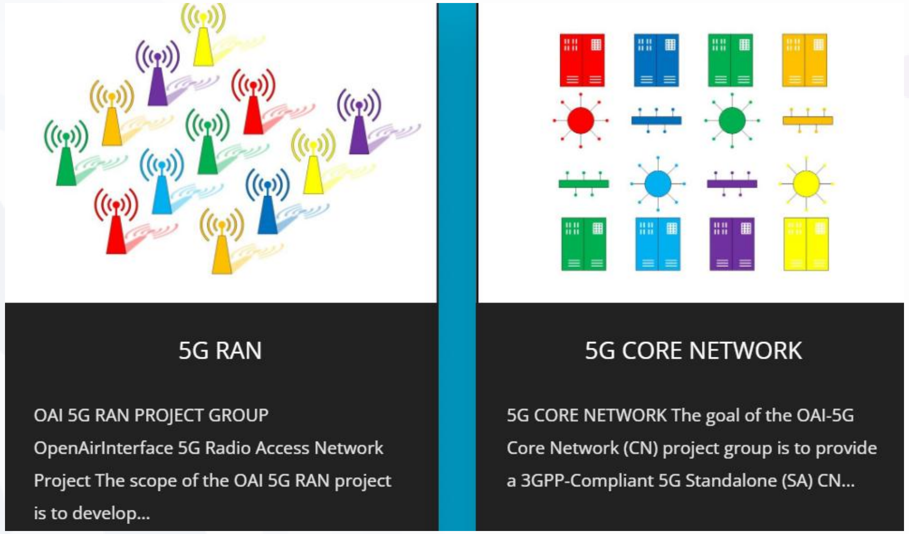
OAI

ACE-Lab

OpenAirInterface (OAI)

5G software alliance for democratizing wireless innovation

- **Main Projects:** 5G RAN Project & 5G Core Network Project
- **Main application:** Simulation testing of base stations and core networks, and deployment of temporary base stations
- **Deployment platform: CPU + FPGA/GPU**
- **Target Stakeholders:** Researcher and Special application scenarios (Disaster relief)



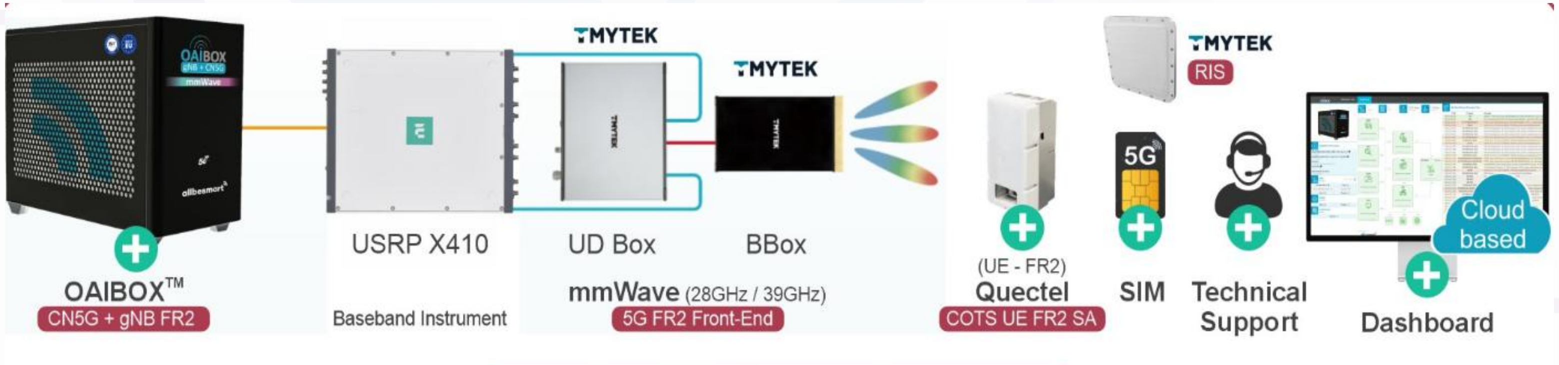
5G RAN

OAI 5G RAN PROJECT GROUP OpenAirInterface 5G Radio Access Network Project The scope of the OAI 5G RAN project is to develop...

5G CORE NETWORK

5G CORE NETWORK The goal of the OAI-5G Core Network (CN) project group is to provide a 3GPP-Compliant 5G Standalone (SA) CN...

OAIBOX™



OAIBOX^{TM} mmWave

5G-SA FR2 end-to-end Research Platform

srsRAN

srsRAN Project

Open-source 4G and 5G software radio suites developed by Software Radio Systems

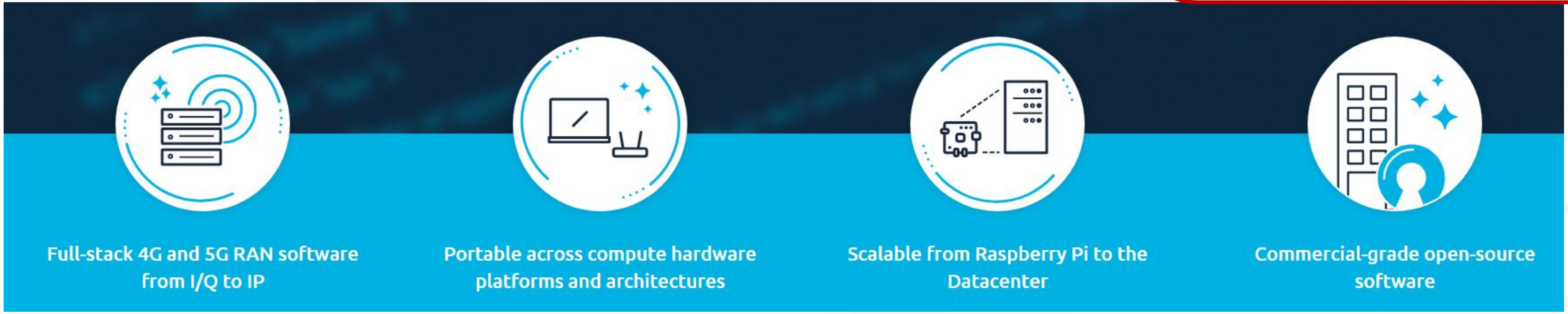
- **Implementation Language: C++**
- **Deployment platform:** x86, ARM and AMD processors with SIMD
- **Target Stakeholders:** Researcher and 5G private network deployment

  (Factory automation, smart industrial parks)
- **Main Features:**

Compare to OAI:

✓ **srsRAN:** more lightweight and easier to deploy, suitable for quickly verify communication solutions or build low-cost testing environments.

✓ **OAI:** more focused on in-depth protocol research and the simulation of complex scenarios.

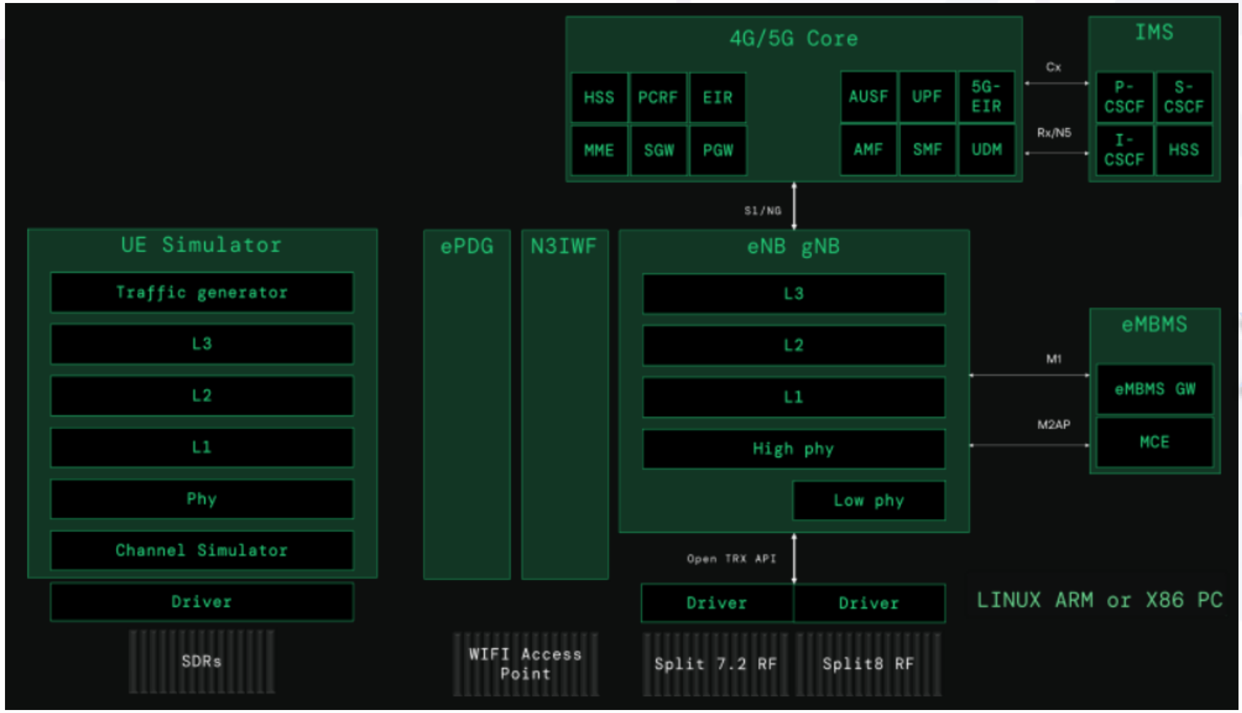

Full-stack 4G and 5G RAN software from I/Q to IP
Portable across compute hardware platforms and architectures
Scalable from Raspberry Pi to the Datacenter
Commercial-grade open-source software

Proprietary Solutions

amarisoft

Amarisoft Technology

Deliver distinctive eNB, gNB, and UE simulator software to the wireless industry, compatible with readily available off-the-shelf hardware, including the physical layer.



Amarisoft's Projects overview

Target Stakeholders: Researcher and Industry

- **For Labs:** Offer cost-effective and high-quality test equipment for both device and base station testing.
- **For Industry:** Provide the network deployment market with comprehensive products and services that address the requirements of both public and private networks.

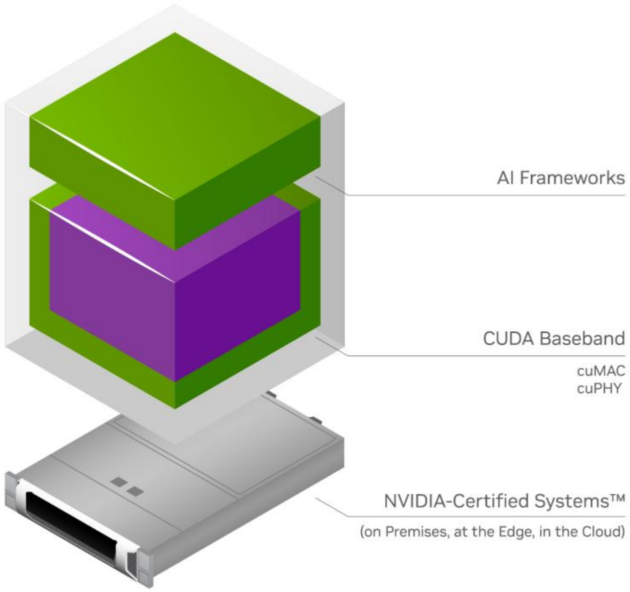
Problems: The ecosystem is closed, and AI is not supported.

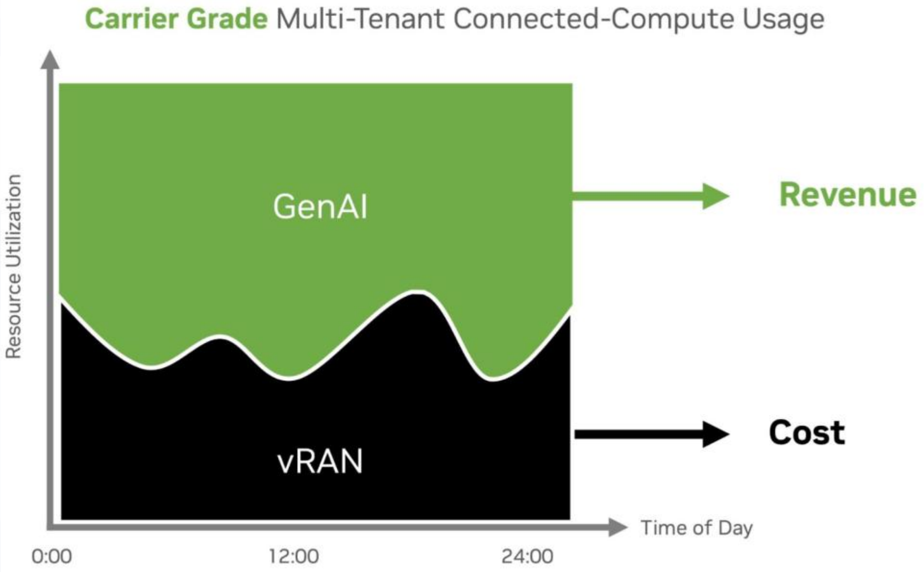
AI-RAN

Feature: One Accelerated Infrastructure (GPU) for AI and RAN

Benefits:

- Unlock New AI Applications and Services
- Maximize Resource Utilization
- Boost Radio Performance
- Prepare for 6G
- One Extensible Architecture


AI Frameworks
CUDA Baseband
cuMAC
cuPHY
NVIDIA-Certified Systems™
(on Premises, at the Edge, in the Cloud)



Three Aerial-based platforms:

- Aerial CUDA Accelerated RAN
- Sionna Neural Radio Framework
- NVIDIA Aerial Omniverse Digital Twin

Compared to the previous projects:

✓ Support AI-native mobile network.

✓ Lower power consumption than RAN implemented by x86.

Target Stakeholders: Researchers and operators

Comparisons and Preview of Our Solution

		OAI	srsRAN	Amarisoft	AI-RAN	Modem+NPU	Our Solution
For Researchers	UE Simulation	✓	✓	✓	✓		✓
	RAN Simulation	✓	✓	✓	✓		✓
For Commercial	Private Network	✓	✓	✓	✓	✓	✓
	RAN (For operators)				✓	✓	✓
	UE					✓	✓
AI Performance		╲	╲	╲	High	High	Almost High
Baseband Performance		Low	Low	Low	Medium	High	Almost High
Energy Efficiency		Low	Low	Low	Medium	High	High
Cost		x86 (High)	x86 (High)	x86 (High)	GPU (High)	ASIC (Low)	Venus (Low)
Ecosystem		Open	Open	Closed	Open	Closed	Open

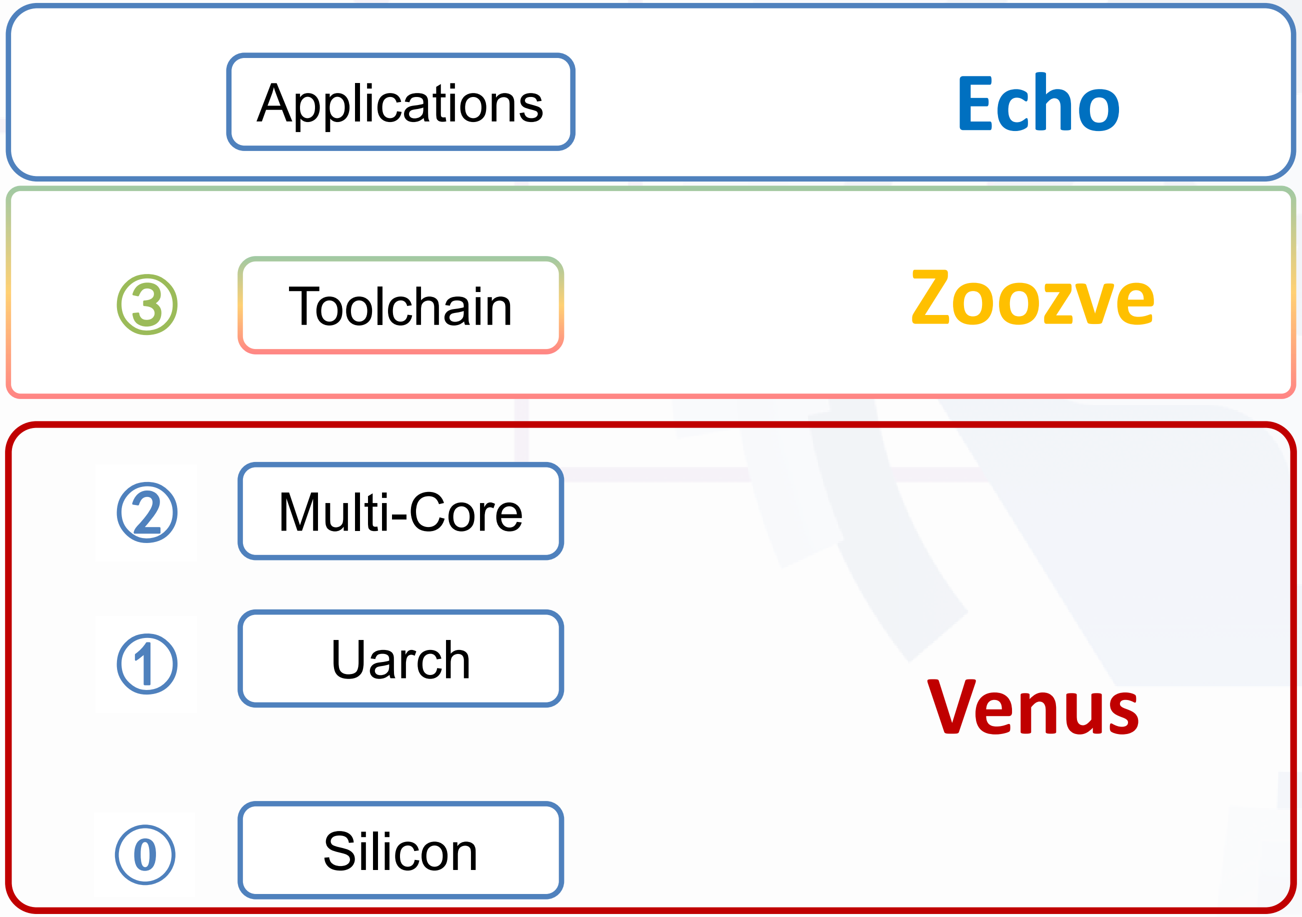
OAI & srsRAN: run on x86/arm CPUs, poor performance, mainly used for research simulation, do not support AI-native.

AI-RAN: run on GPU, higher energy consumption of communication than ASIC, can't be used on UE side.

Modem+NPU: Inflexible for academic research and high latency caused by inter-chip communication.

Our Solution

- **To build a open-source, fully software-defined and evolvable, AI-integrated mobile network baseband architecture**



Applications

Echo

③ Toolchain

Zoozve

② Multi-Core

① Uarch

⓪ Silicon

Venus

An Open-Source library for 5G, 4G, GNSS, LoRa and etc.

AI and wireless baseband processing (WBP) integrated toolchain

Multi-Core Dataflow-Driven RISC-V Domain Specific Architecture for Integrated AI and WBP

Outline

ACE-Lab

- Background & Motivation
- Related Works
- Echo: An Open-Source 5G/4G/GNSS/LoRa/AI Library
- Venus: A Multi-Core Dataflow-Driven RISC-V Domain Specific Architecture and Implementation on 40nm CMOS
- Zoozve: A Strip-Mining-Free RISC-V Vector Extension Compiler
- Conclusion

Integrated AI and Wireless with Software & Hardware Joint Opt.

Venus & Echo

A software-hardware fully decoupled solution designed for AI-Native wireless baseband

Venus: **The first domain-specific RISC-V instruction set architecture integrating AI and communication.**

- AI-Communication converges instruction sets
- Fully decoupled software and hardware design
- Programmable DSA accelerator

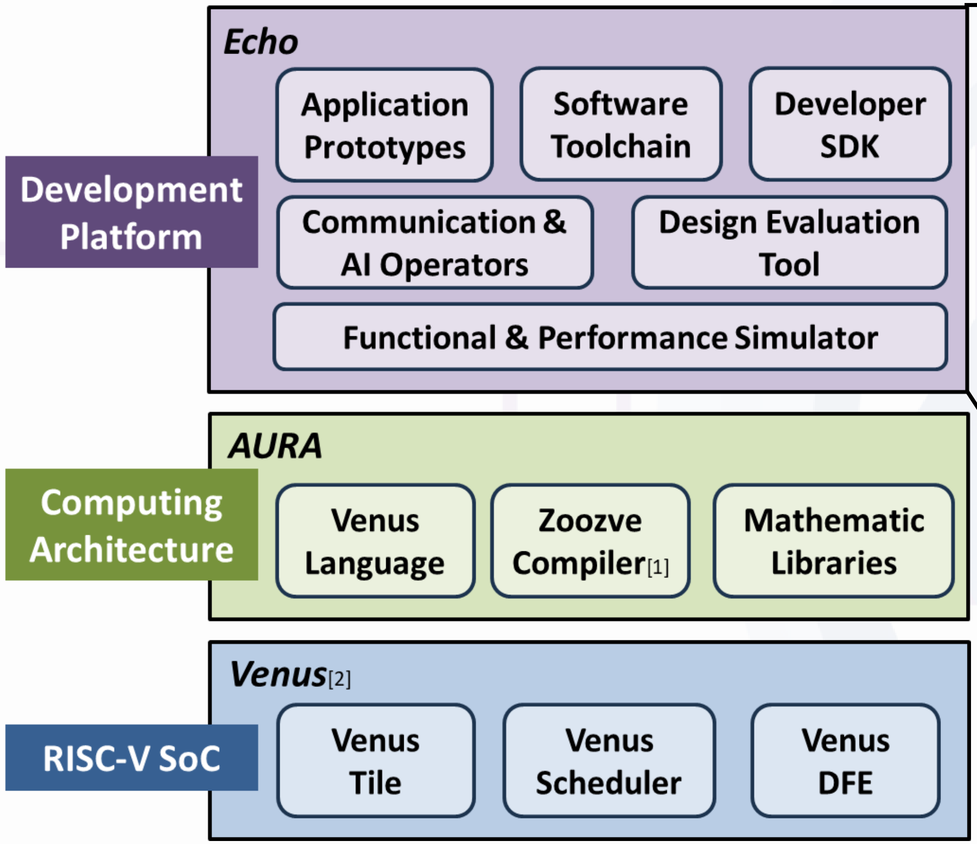
Echo: **The open-source wireless Testbench built upon the Venus instruction set.**

- Full-stack emulation environment
- Communication performance evaluation
- High-performance wireless operator library

	Flexibility	*Wireless Performance*	*Wireless EE*	*AI Performance*	*AI EE*	*Cost*
Venus	High	High	High	Almost high	High	Low
AI-RAN	High	High	Low	High	High	High
DSP+NPU	Low	Medium	High	Medium	High	Low

What is *Echo* ?

- The open-source ComAI development platform build on ***AURA*** architecture and ***Venus*** RISC-V SoC.



Echo

Development Platform

Application Prototypes

Software Toolchain

Developer SDK

Communication & AI Operators

Design Evaluation Tool

Functional & Performance Simulator

AURA

Computing Architecture

Venus Language

Zoozve Compiler[1]

Mathematic Libraries

Venus[2]

RISC-V SoC

Venus Tile

Venus Scheduler

Venus DFE

Echo includes:

Application Prototypes: 5G/LTE, AI-Based Channel Estimation, GNSS,LORA

Communication & AI Operators: FFT, decoder, Conv2D/3D, GELU/SiLU

Software Toolchain: Compiler, debugger, ISA support

Developer SDK: Libraries, APIs, documentation

Design Evaluation Tool: Test application performance at different hardware scales

Functional & Performance Simulator: Simulates Venus workloads and output Latency & throughput estimation

Project Info：

- GitHub: *https://github.com/ACELab-SHU/ACE-Echo*
- Website: *https://acelab-shu.github.io/ACE-Echo*

[1] Xu, Siyi, et al. "Zoozve: A Strip-Mining-Free RISC-V Vector Extension with Arbitrary Register Grouping Compilation Support (WIP)." *Proceedings of the 26th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*. 2025.

[2] Jiang, Limin, et al. "A hierarchical dataflow-driven heterogeneous architecture for wireless baseband processing." *Proceedings of the 30th Asia and South Pacific Design Automation Conference.* 2025.

What will *Echo* do & Who is *Echo* for ?

What Echo can do ?

- **Unified Com & AI Development**

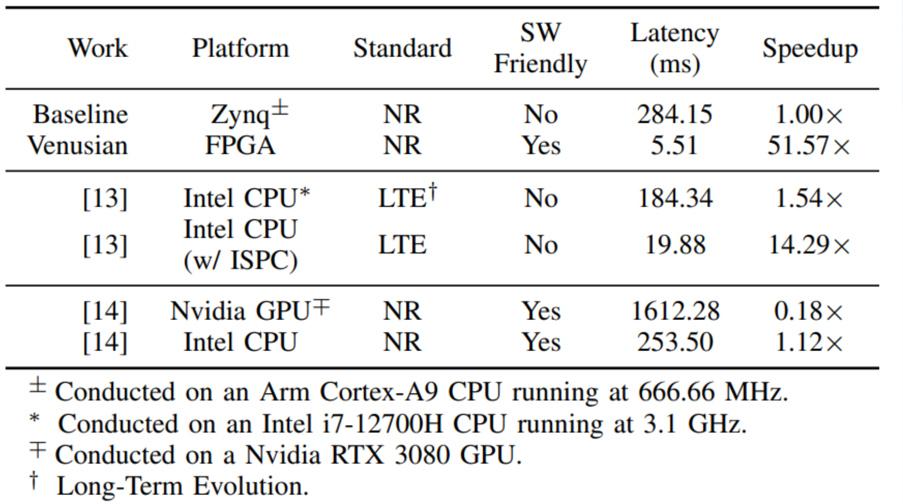
  Simplifies joint AI and communication system design through one unified programming model.

- **Dual-Precision Execution**

  One codebase, dual targets — generate both floating-point and fixed-point versions for fast deployment.

- **Software-Hardware co-design & cycle-accurate simulation**
- **Higher performance and better energy efficiency than GPPs**

Table.1 Comparison with earlier works for performance and software-friendliness



Work	Platform	Standard	SW Friendly	Latency (ms)	Speedup
Baseline	Zynq±	NR	No	284.15	1.00×
Venusian	FPGA	NR	Yes	5.51	51.57×
[13]	Intel CPU*	LTE†	No	184.34	1.54×
[13]	Intel CPU (w/ ISPC)	LTE	No	19.88	14.29×
[14]	Nvidia GPU∓	NR	Yes	1612.28	0.18×
[14]	Intel CPU	NR	Yes	253.50	1.12×

± Conducted on an Arm Cortex-A9 CPU running at 666.66 MHz.
* Conducted on an Intel i7-12700H CPU running at 3.1 GHz.
∓ Conducted on a Nvidia RTX 3080 GPU.
† Long-Term Evolution.

🎓 Academia & Researchers

→ Open-Source Platform for Communication-AI Research

Echo provides a low-cost, low-power environment to prototype and validate communication algorithms with real-world performance. Ideal for academic research and rapid innovation.

🏭 Industry

→ Decoupled Software-Hardware Baseband Chip Solution

Accelerate your baseband chip development with a modular, software-first approach.

- **R&D Cycle Reduced:** From 12–18 months to just 3–6 months.

🧩 Standards Organizations

→ Fast-Track 6G Technology Validation

Streamline the path to 6G standardization with efficient tools and full-stack communication-AI libraries.

- Cut traditional prototype cycles (3–5 years) down to months.

Current Features of *Echo v0.1*

ACE-Lab

Echo v0.1 : (Current Features)

- ✓ **Complete Application Development Support**
  Includes compiler, libraries, debugging, and simulation tools.
- ✓ **Hardware-Consistent Fixed-Point Simulation and register usage analysis**
- ✓ **3GPP-Compliant 5G/LTE PHY Library**
  Such as channel coding (Polar/Turbo/LDPC), OFDM, channel estimation, modulation…
- ✓ **5G Cell Search Demo Implemented**
  End-to-end demo showcasing Venus-based baseband algorithm implementation.
- ✓ **A few AI operators**

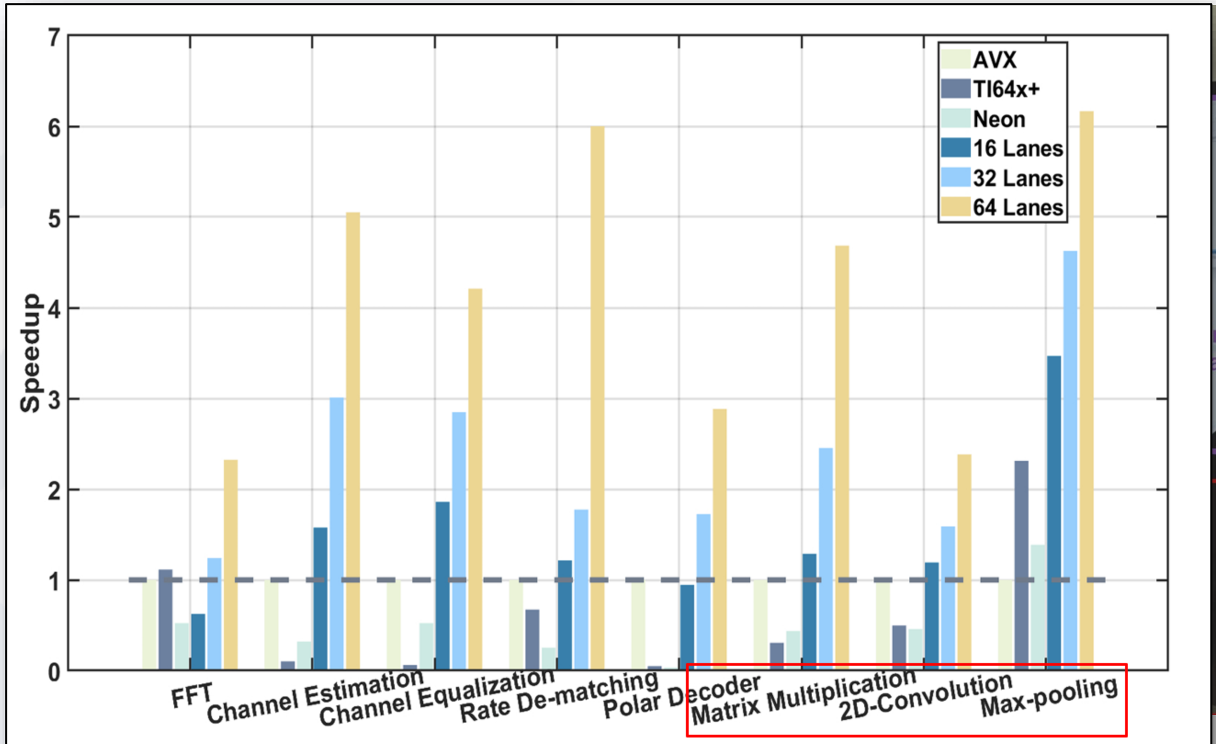
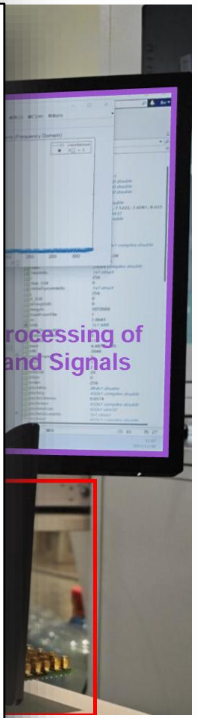


Fig.2 Single Venus tile performance compared with Intel AVX, Arm Neon and TI C64x+ DSP.



uccessfully

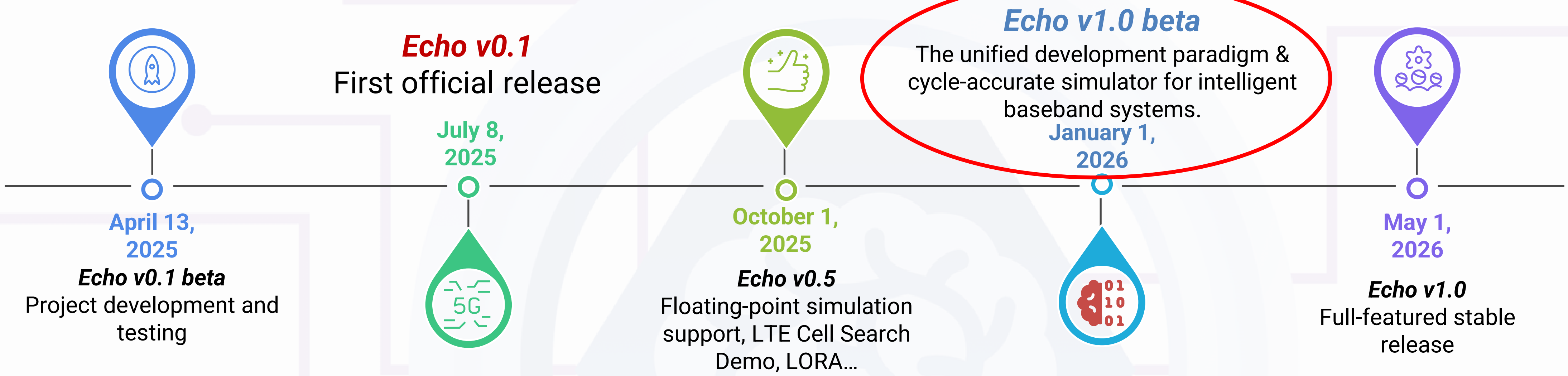
Try *Echo* Today !!!

- We invite you to download and experience the ***Echo*** platform.
- Our simulator can perform full cell search and decode MIB/SIB now.
- The same application can also be deployed directly on the Venus chip for real-time execution.

Home Page:
https://acelab-shu.github.io/ACE-Echo

Future Roadmap of *Echo*

Echo Project Release Timeline:



April 13, 2025
Echo v0.1 beta
Project development and testing

July 8, 2025
Echo v0.1
First official release

October 1, 2025
Echo v0.5
Floating-point simulation support, LTE Cell Search Demo, LORA…

January 1, 2026
Echo v1.0 beta
The unified development paradigm & cycle-accurate simulator for intelligent baseband systems.

May 1, 2026
Echo v1.0
Full-featured stable release

🧠 *Echo v1.0 beta*

Scheduled for release before January 1, 2026

A major milestone in our open-source journey —— officially introducing a unified programming paradigm for Communication-AI fusion.

Main Features:

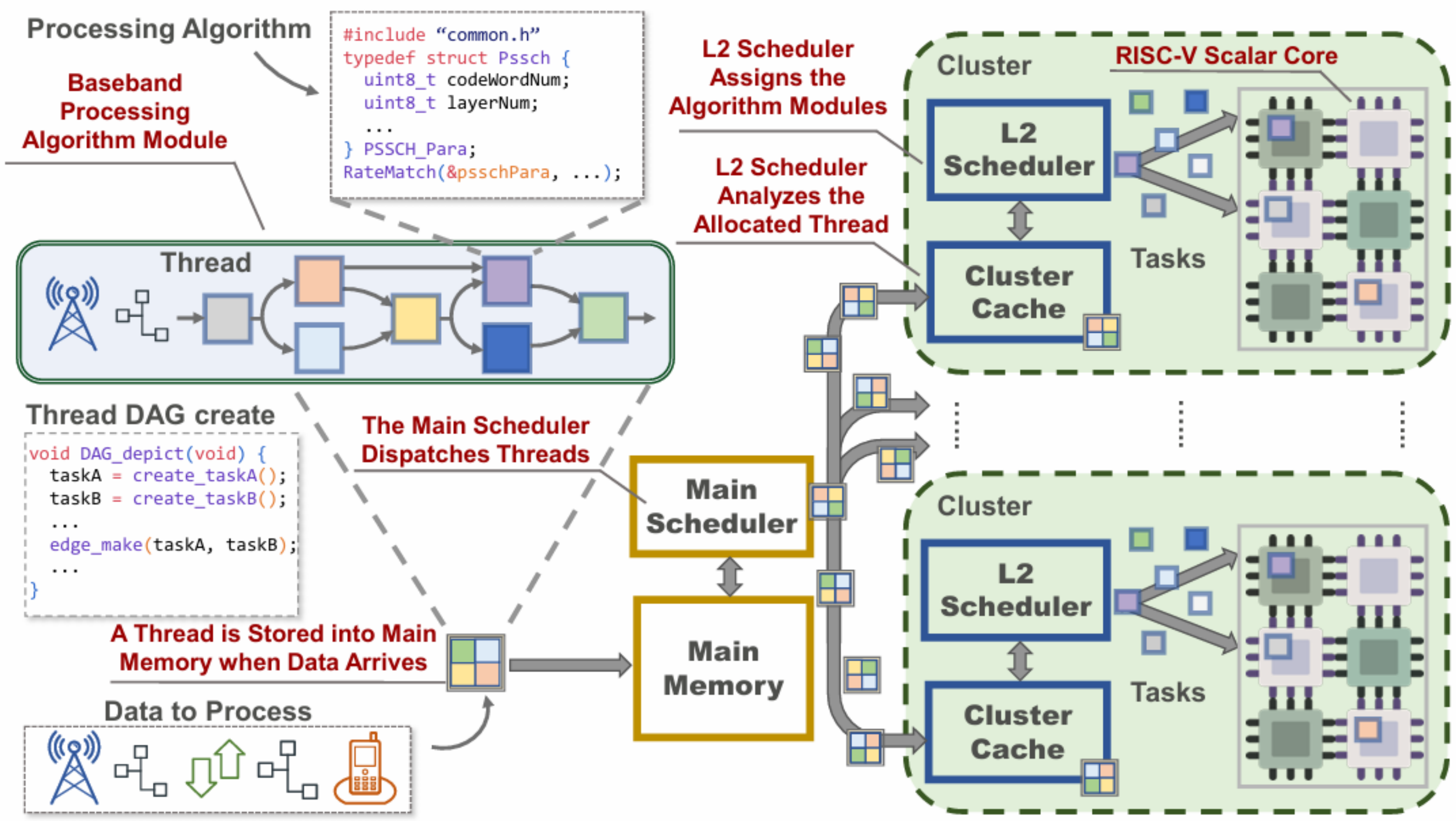
- Unified programming model for Com & AI
- Cycle-Accurate Simulator for software-hardware co-design
- AI Operator Library
- Richer Operator Library of signal processing

With ***Echo***, we aim to **lower the barrier to intelligent baseband innovation** — enabling a broader community to explore, simulate, and accelerate next-generation wireless systems.

We warmly welcome more developers, researchers, and collaborators to join us on this journey.

With Echo —— Rapid Prototyping AI-native Wireless Networks

Echo provides cycle-accurate execution time estimation for programs when deployed on Venus.



```
#include "common.h"
typedef struct Pssch {
  uint8_t codeWordNum;
  uint8_t layerNum;
  ...
} PSSCH_Para;
RateMatch(&psschPara, ...);
```

```
void DAG_depict(void) {
  taskA = create_taskA();
  taskB = create_taskB();
  ...
  edge_make(taskA, taskB);
  ...
}
```

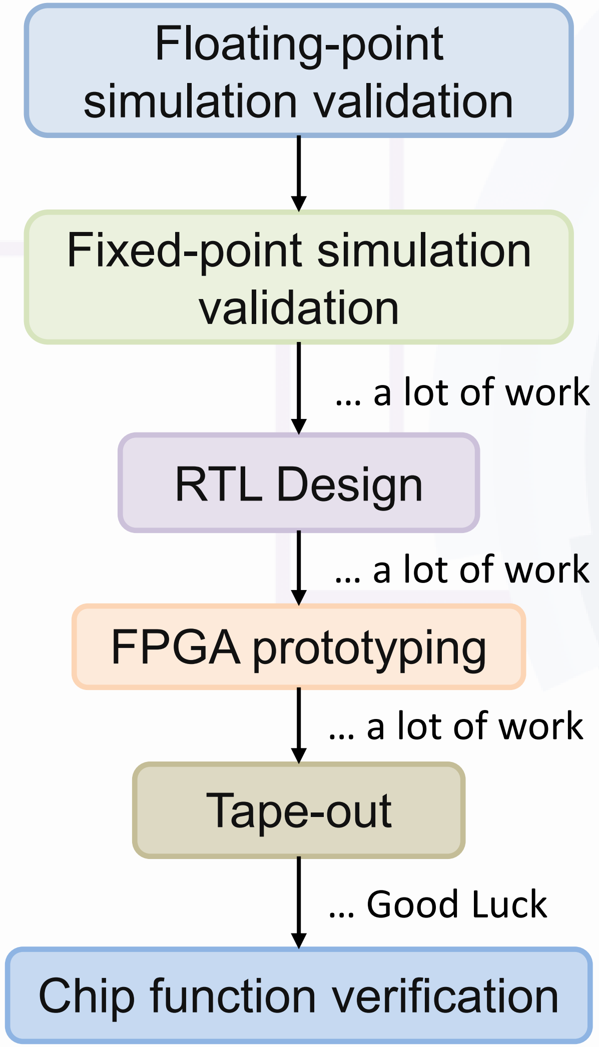
Processing Algorithm
Baseband Processing Algorithm Module
Thread
Thread DAG create
The Main Scheduler Dispatches Threads
A Thread is Stored into Main Memory when Data Arrives
Data to Process
Main Scheduler
Main Memory
L2 Scheduler Assigns the Algorithm Modules
L2 Scheduler Analyzes the Allocated Thread
Cluster
RISC-V Scalar Core
L2 Scheduler
Cluster Cache
Tasks
Cluster
L2 Scheduler
Cluster Cache
Tasks

Programming model description, including multi-level dataflow scheduling

From programming languages to hardware design, our **full-stack vertical integration** enables us have a clear understanding of the mapping between each Venus language and hardware, so we can directly give the actual execution time of the program written by the developer on the Venus chip of the selected size and configuration.

With Venus & Echo —— Rapid Product Commercialization

Traditional baseband development process



Floating-point simulation validation

Fixed-point simulation validation

... a lot of work

RTL Design

... a lot of work

FPGA prototyping

... a lot of work

Tape-out

... Good Luck

Chip function verification

With the Help of Venus & Echo

Develop your program by the ***Echo***

Echo provides comprehensive simulation reports encompassing both floating-point and fixed-point results. Developers can directly inspect the fixed-point implementation outcomes, while leveraging the floating-point reference data to isolate error sources - determining whether discrepancies stem from algorithmic flaws or excessive quantization errors.

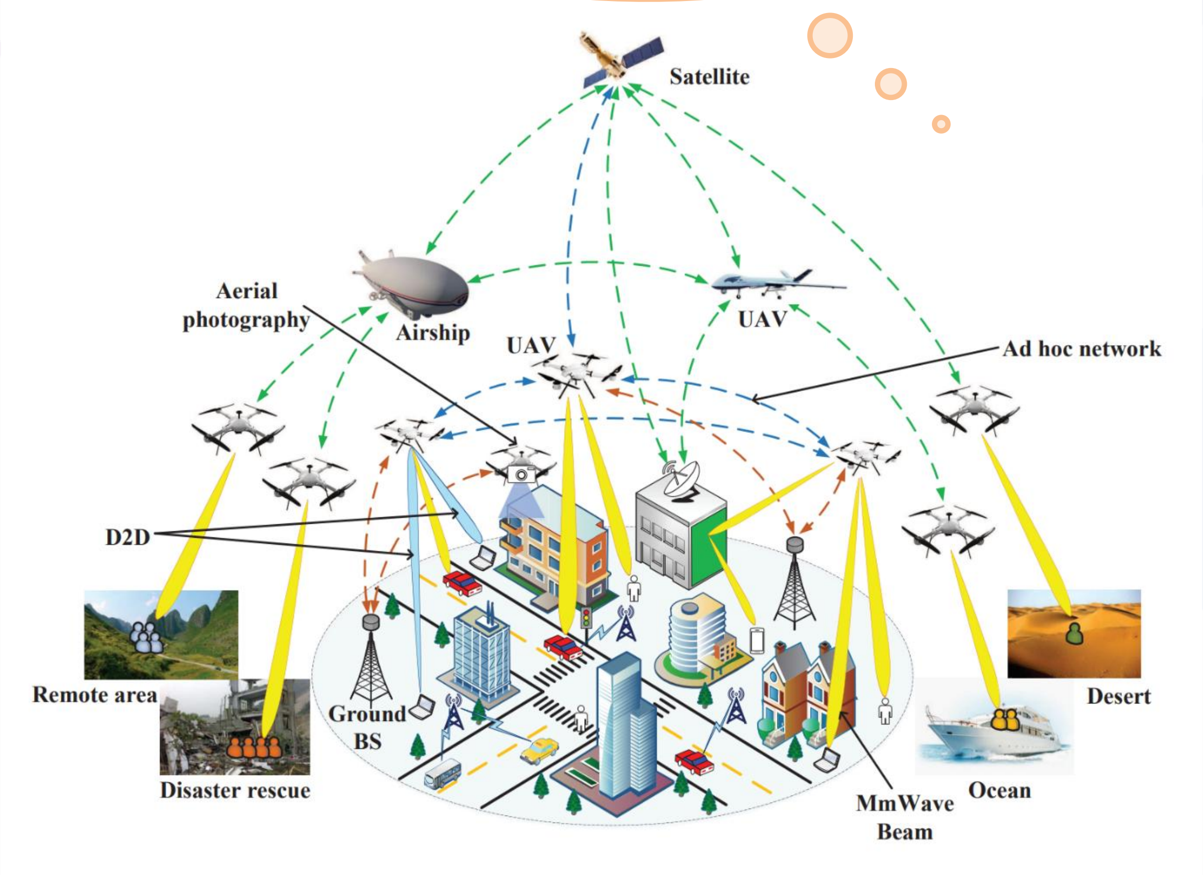
Concurrently, ***Echo*** generates hardware deployment runtime projections, enabling engineers to verify real-time performance compliance with system requirements.

Program the software into the ***Venus*** chip

With Venus & Echo —— Low-cost & Reliable Communication Testing

Many new scenarios !!! Impossible to create simulation environments that perfectly replicate real-world.

Just software simulation cannot guarantee the reliability of communication algorithms in actual deployments.



Comparison of different platforms in real scenario testing

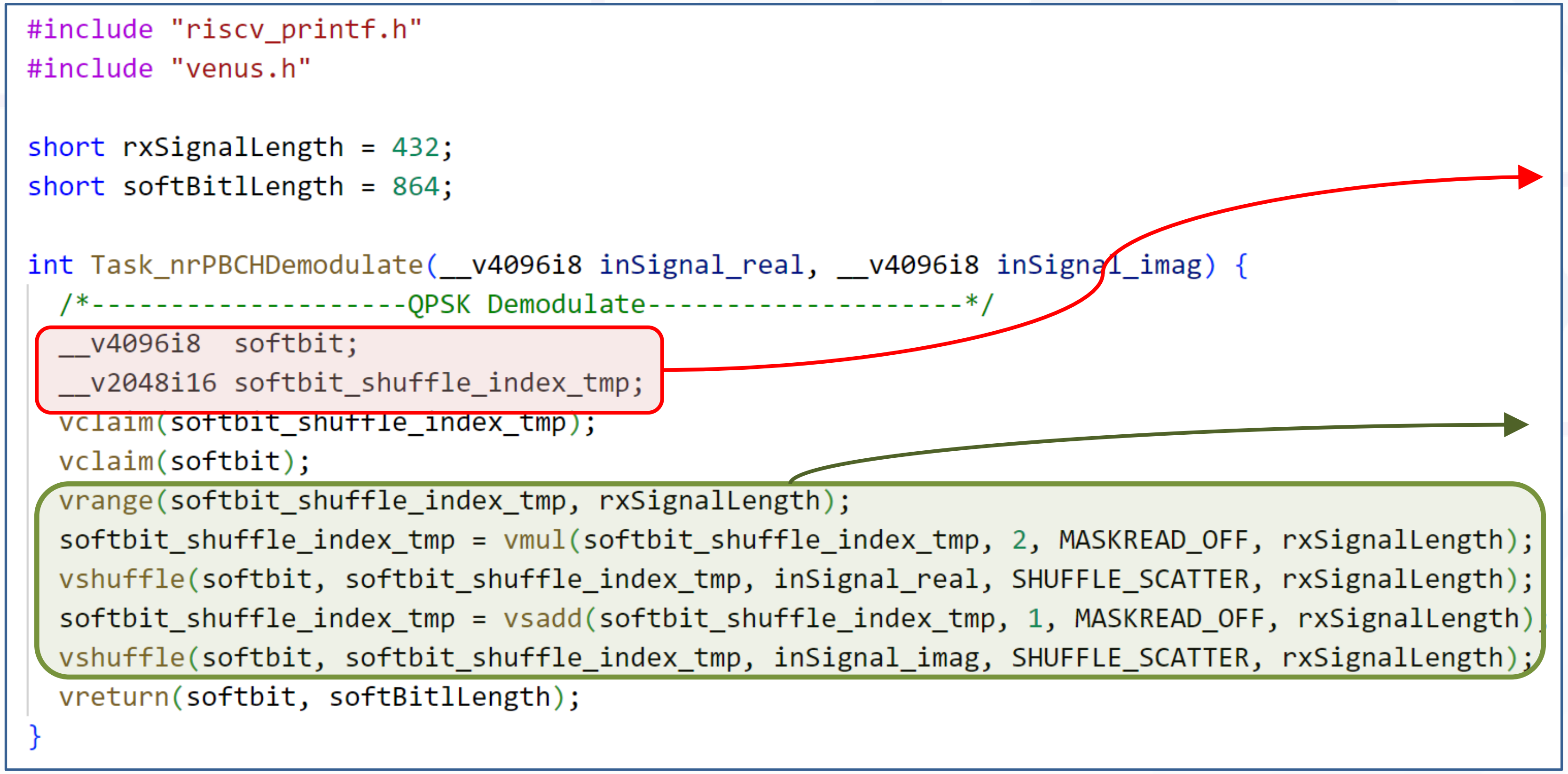
	Venus	GPU	ASIC	x86/arm
Power	Low	High	Low	High
Cost	Low	Medium	High	Medium
Time	2-3 days	2-3 days	1-2 years	2-3 days

For instance, when validating UAV (Unmanned Aerial Vehicle) communication algorithms, power-hungry GPU and x86 architectures prove inadequate for sustained aerial operations due to excessive energy consumption. Meanwhile, ASIC solutions require a 12-month tape-out cycle with costs exceeding $1million for advanced node implementations.

Echo: Programming examples (1/3)

Three steps Complete the application development of Venus

Step1: Write functions (tasks) based on the Venus language



```
#include "riscv_printf.h"
#include "venus.h"

short rxSignalLength = 432;
short softBitlLength = 864;

int Task_nrPBCHDemodulate(__v4096i8 inSignal_real, __v4096i8 inSignal_imag) {
  /*-------------------QPSK Demodulate-------------------*/
  __v4096i8  softbit;
  __v2048i16 softbit_shuffle_index_tmp;
  vclaim(softbit_shuffle_index_tmp);
  vclaim(softbit);
  vrange(softbit_shuffle_index_tmp, rxSignalLength);
  softbit_shuffle_index_tmp = vmul(softbit_shuffle_index_tmp, 2, MASKREAD_OFF, rxSignalLength);
  vshuffle(softbit, softbit_shuffle_index_tmp, inSignal_real, SHUFFLE_SCATTER, rxSignalLength);
  softbit_shuffle_index_tmp = vsadd(softbit_shuffle_index_tmp, 1, MASKREAD_OFF, rxSignalLength);
  vshuffle(softbit, softbit_shuffle_index_tmp, inSignal_imag, SHUFFLE_SCATTER, rxSignalLength);
  vreturn(softbit, softBitlLength);
}
```

QPSK Demodulation task based on the Venus language

- ✓ Similar to C
- ✓ Added a new variable type

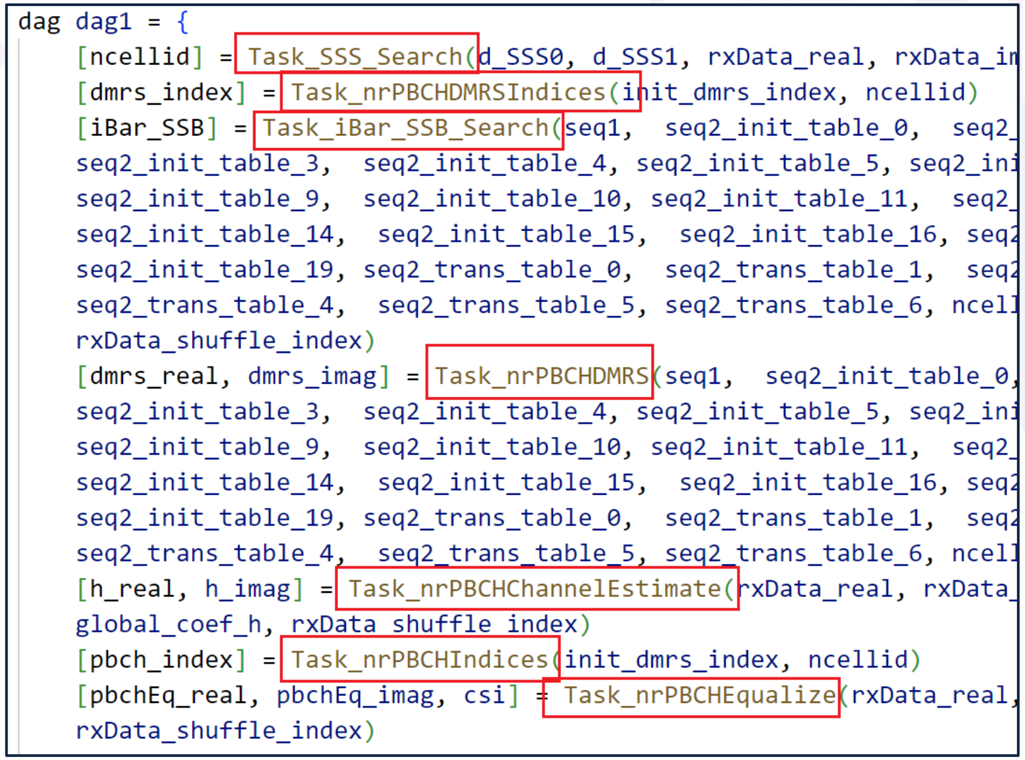
  ——Vectors
- ✓ Added vector calculation

  instructions for Venus, such as

  vshuffle, vmul, vsadd ……

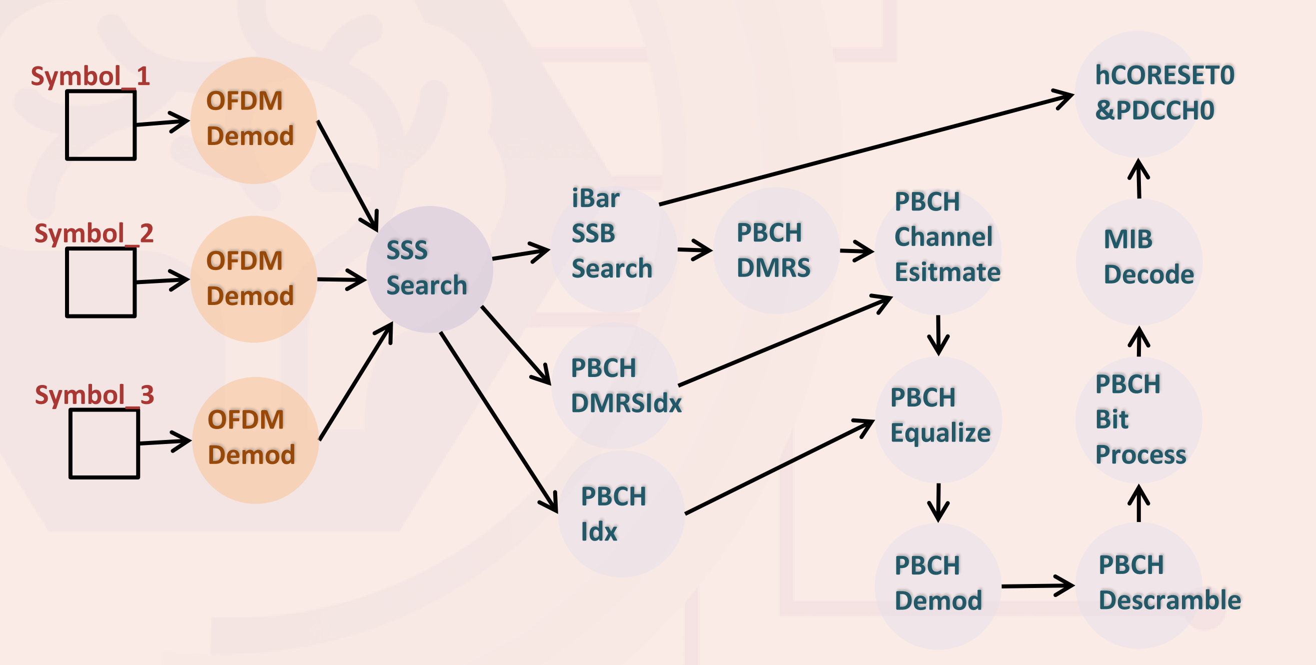
Echo: Programming examples (2/3)

Step2: Write a .bas file to connect the functions (tasks) you developed into a DAG.



```
dag dag1 = {
    [ncellid] = Task_SSS_Search(d_SSS0, d_SSS1, rxData_real, rxData_im
    [dmrs_index] = Task_nrPBCHDMRSIndices(init_dmrs_index, ncellid)
    [iBar_SSB] = Task_iBar_SSB_Search(seq1,  seq2_init_table_0,  seq2_
    seq2_init_table_3,  seq2_init_table_4, seq2_init_table_5, seq2_ini
    seq2_init_table_9,  seq2_init_table_10, seq2_init_table_11,  seq2_
    seq2_init_table_14,  seq2_init_table_15,  seq2_init_table_16, seq2
    seq2_init_table_19, seq2_trans_table_0,  seq2_trans_table_1,  seq2
    seq2_trans_table_4,  seq2_trans_table_5, seq2_trans_table_6, ncell
    rxData_shuffle_index)
    [dmrs_real, dmrs_imag] = Task_nrPBCHDMRS(seq1,  seq2_init_table_0,
    seq2_init_table_3,  seq2_init_table_4, seq2_init_table_5, seq2_ini
    seq2_init_table_9,  seq2_init_table_10, seq2_init_table_11,  seq2_
    seq2_init_table_14,  seq2_init_table_15,  seq2_init_table_16, seq2
    seq2_init_table_19, seq2_trans_table_0,  seq2_trans_table_1,  seq2
    seq2_trans_table_4,  seq2_trans_table_5, seq2_trans_table_6, ncell
    [h_real, h_imag] = Task_nrPBCHChannelEstimate(rxData_real, rxData_
    global_coef_h, rxData_shuffle_index)
    [pbch_index] = Task_nrPBCHIndices(init_dmrs_index, ncellid)
    [pbchEq_real, pbchEq_imag, csi] = Task_nrPBCHEqualize(rxData_real,
    rxData_shuffle_index)
```

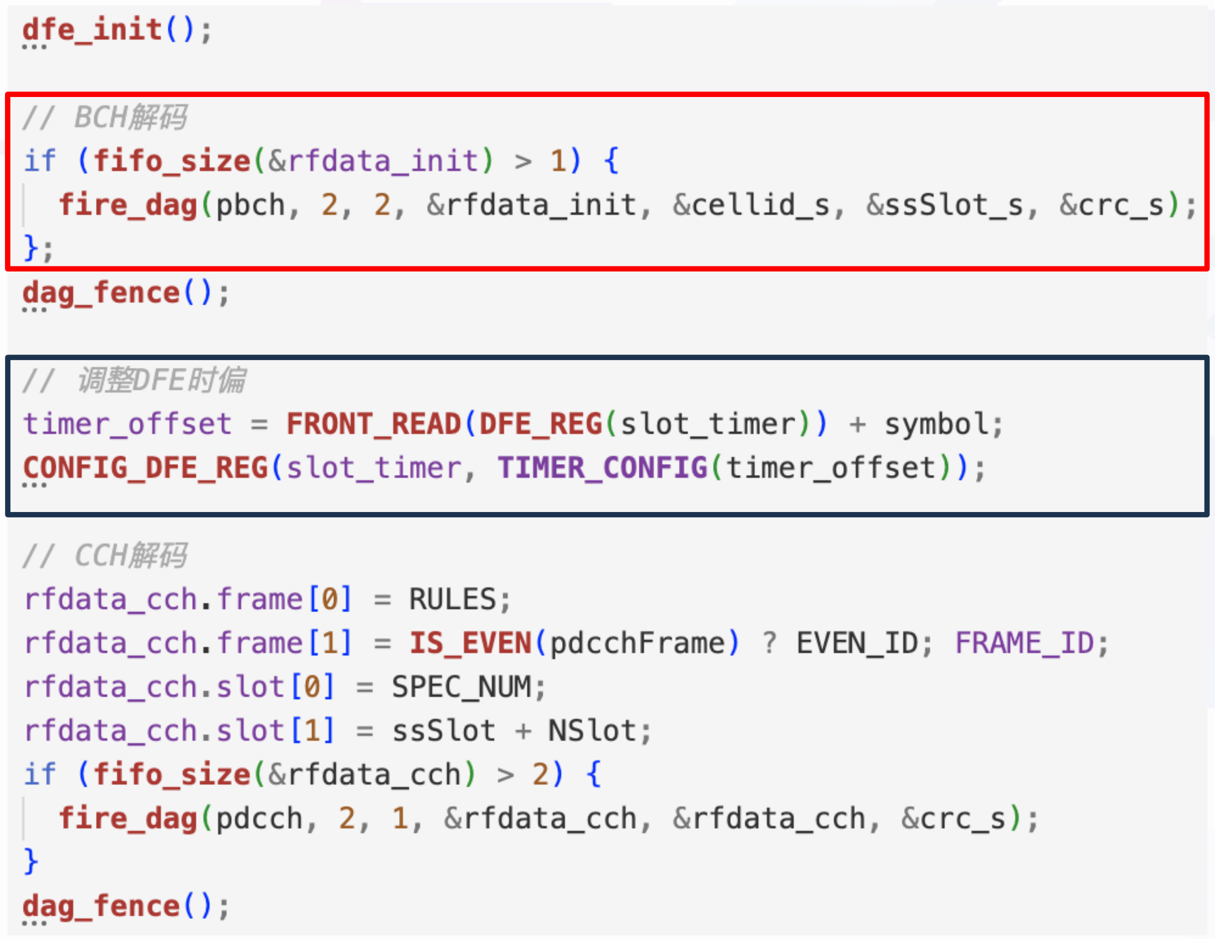
Screenshot of a portion of NR PBCH.bas



Directed Acyclic Graph (DAG) of NR PBCH

Echo: Programming examples (3/3)

Step3: Write L1 scheduler files to delivery Dag-level tasks, to control hardware interrupts and data transmission with the DFE (Digital Front-End)



```
dfe_init();

// BCH解码
if (fifo_size(&rfdata_init) > 1) {
  fire_dag(pbch, 2, 2, &rfdata_init, &cellid_s, &ssSlot_s, &crc_s);
};
dag_fence();

// 调整DFE时偏
timer_offset = FRONT_READ(DFE_REG(slot_timer)) + symbol;
CONFIG_DFE_REG(slot_timer, TIMER_CONFIG(timer_offset));

// CCH解码
rfdata_cch.frame[0] = RULES;
rfdata_cch.frame[1] = IS_EVEN(pdcchFrame) ? EVEN_ID; FRAME_ID;
rfdata_cch.slot[0] = SPEC_NUM;
rfdata_cch.slot[1] = ssSlot + NSlot;
if (fifo_size(&rfdata_cch) > 2) {
  fire_dag(pdcch, 2, 1, &rfdata_cch, &rfdata_cch, &crc_s);
}
dag_fence();
```

Specify the Dag to be calculated, its trigger conditions, and data inputs

Control and configure the digital RF front-end

Other features:

1. Handling peripheral interrupts
2. Handling DMA interrupts
3. Analyze DAG return values
4. Controlling DMA data transfers
5. ……

Echo -- Basic Framework is Ready

- ✓ **L1 Scheduler：** Interact with hardware peripherals, interrupt handling, and Dag-level task delivery
- ✓ **L2 Scheduler：** Dag analyze and task delivery
- ✓ **Venus Compiler:** The whole process of compilation - preprocessing, compilation, assembly, linking

Emulator and open-source repositories are coming soon……

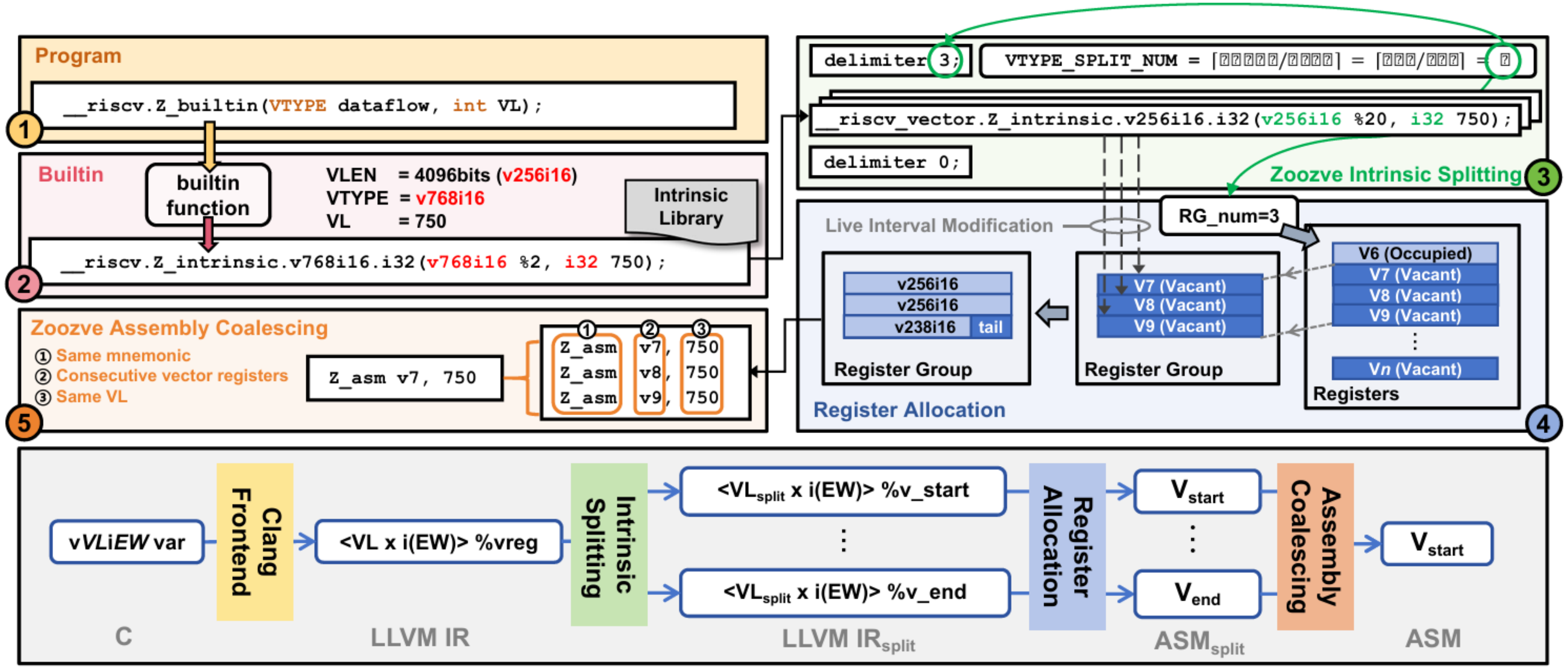


Figure - Compilation workflow for Venus Compiler (***Zoozve***) in LLVM

LTE/NR & AI High-performance Operator Library

✓ Almost all modules in the LTE/NR physical layer are already supported.

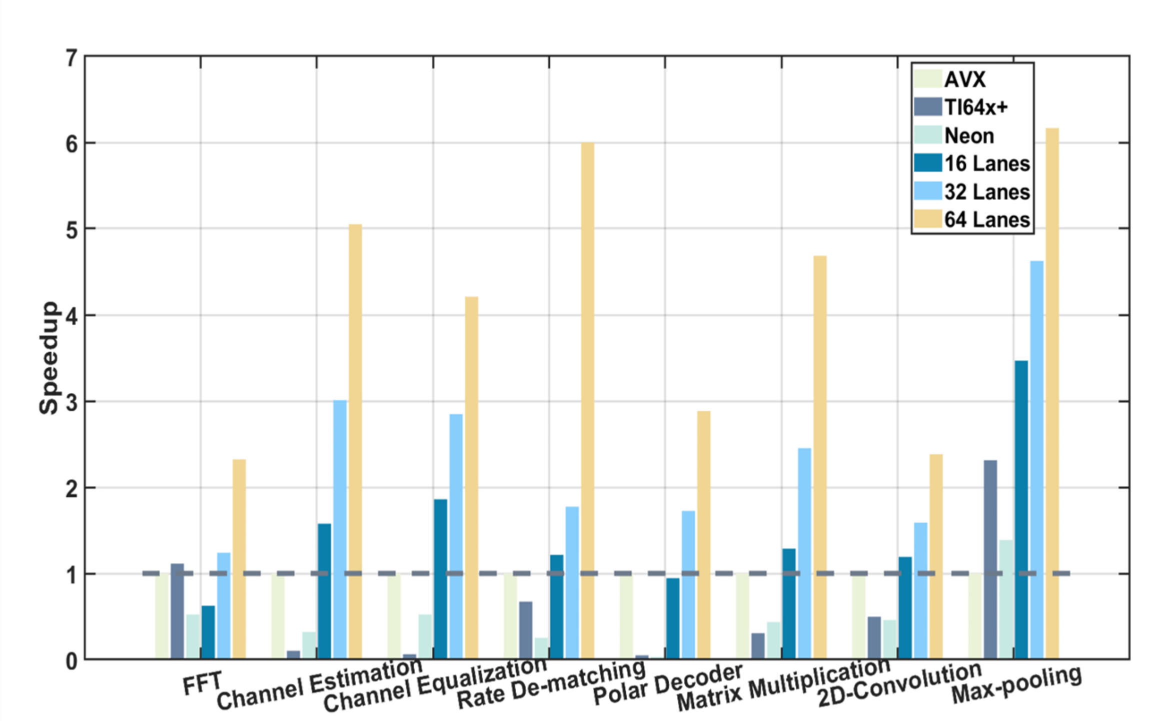
✓ Some of the basic operators of AI.

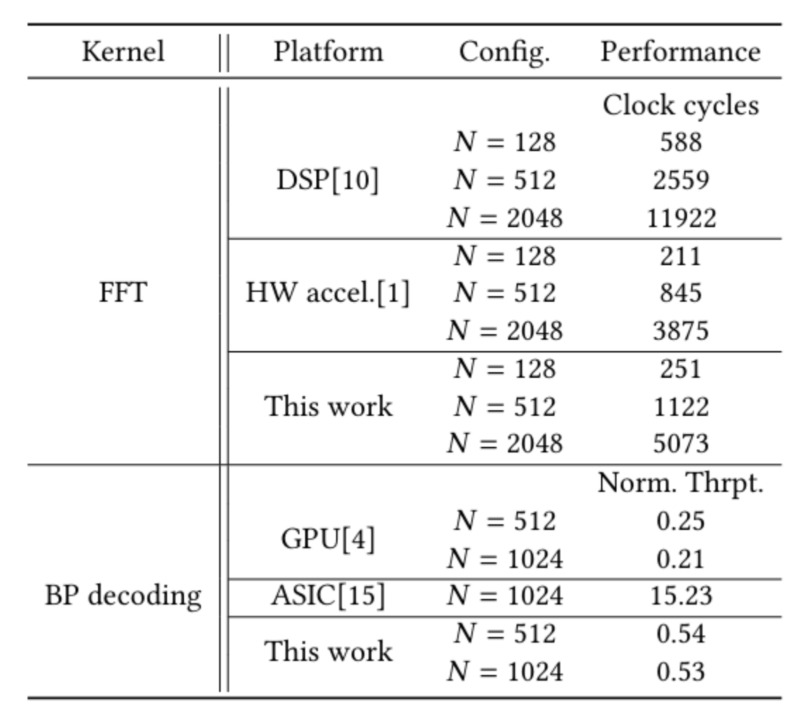
✓ Well-optimized modules such as FFT, LDPC, Polar, channel estimation, etc.

More high-performance modules will be available soon ...

Echo Toolbox

FFT, Channel Estimation, Polar, LDPC, Turbo…
Matrix Multiplication,2D-Convolution, Max-pooling…



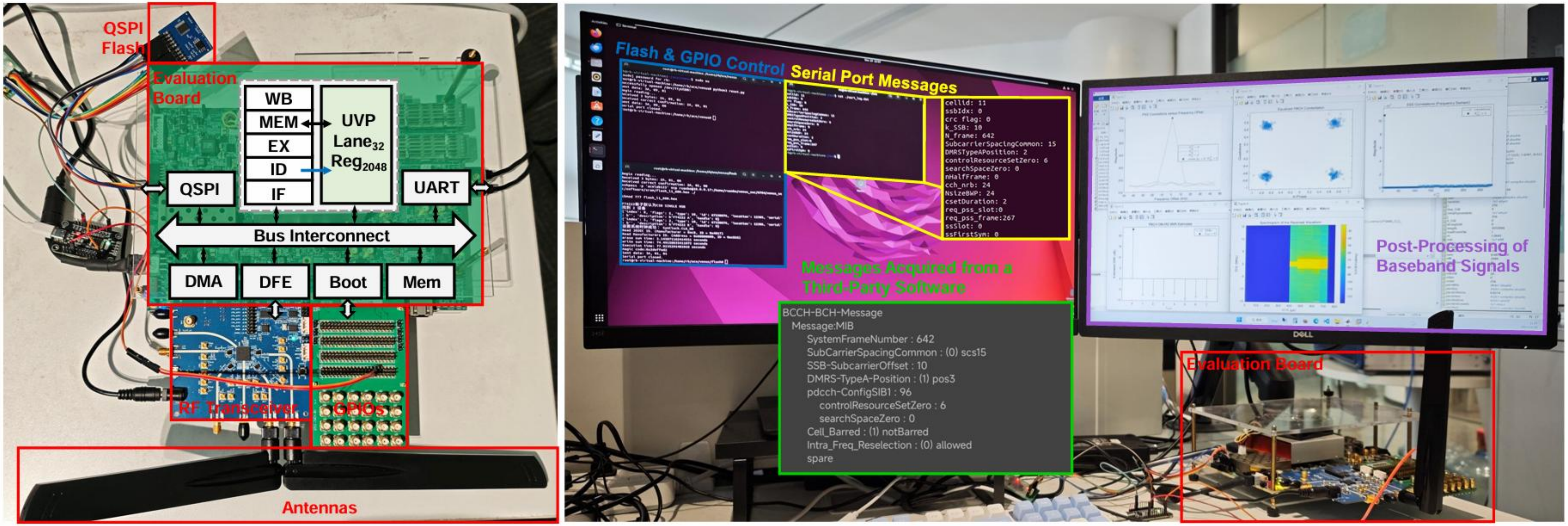


Kernel	Platform	Config.	Performance
FFT			Clock cycles
	DSP[10]	$N = 128$	588
		$N = 512$	2559
		$N = 2048$	11922
	HW accel.[1]	$N = 128$	211
		$N = 512$	845
		$N = 2048$	3875
	This work	$N = 128$	251
		$N = 512$	1122
		$N = 2048$	5073
BP decoding			Norm. Thrpt.
	GPU[4]	$N = 512$	0.25
		$N = 1024$	0.21
	ASIC[15]	$N = 1024$	15.23
	This work	$N = 512$	0.54
		$N = 1024$	0.53

Figure – Performance demonstration of the main modules

NR/LTE Cell Search Demo

We used the ***Venus*** instruction set to complete the NR/LTE cell search process on the ***Echo***, and through FPGA prototyping, we realized the communication with commercial base stations.



```
cellid: 11
ssbIdx: 0
crc flag: 0
k_SSB: 10
N_frame: 642
SubcarrierSpacingCommon: 15
DMRSTypeAPosition: 2
controlResourceSetZero: 6
searchSpaceZero: 0
nHalfFrame: 0
cch_nrb: 24
NsizeBWP: 24
csetDuration: 2
req_pss_slot:0
req_pss_frame:267
ssSlot: 0
ssFirstSym: 0
```

```
BCCH-BCH-Message
  Message:MIB
     SystemFrameNumber : 642
     SubCarrierSpacingCommon : (0) scs15
     SSB-SubcarrierOffset : 10
     DMRS-TypeA-Position : (1) pos3
     pdcch-ConfigSIB1 : 96
        controlResourceSetZero : 6
        searchSpaceZero : 0
     Cell_Barred : (1) notBarred
     Intra_Freq_Reselection : (0) allowed
     spare
```

Figure -- Evaluation platform for proof of our solution.

Outline

- Background & Motivation
- Related Works
- Echo: An Open-Source 5G/4G/GNSS/LoRa/AI Library
- **Venus: A Multi-Core Dataflow-Driven RISC-V Domain Specific Architecture and Implementation on 40nm CMOS**
- Zoozve: A Strip-Mining-Free RISC-V Vector Extension Compiler
- Conclusion

Overview of Venus

- A **cache-free manycore** architecture is proposed to increase energy and area efficiency without compromising performance due to the predictable data processing nature of WBP.
- We develop a ***pack-and-ship*** data dispatch system to enable the tiles to operate in a **bundled access and execution** style, which can drastically reduce the cost of data movement. **Modular/Decoupled**
- A **hierarchical dataflow** task scheduling scheme is designed and two strategies, namely multi-threading and lazy-deletion, are proposed to fully utilize the hardware resources.

Cyclical/Predictable

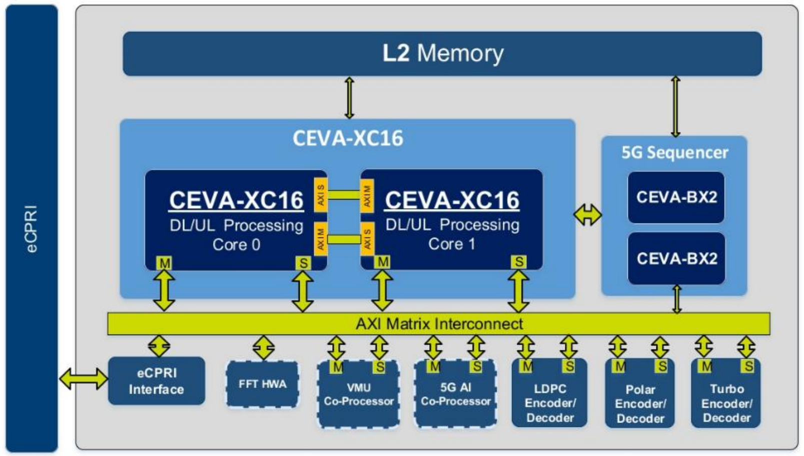
Wireless Baseband Processing (WBP) Architecture

WBP: How?

□DSP:

✓VLIW boosts ILP

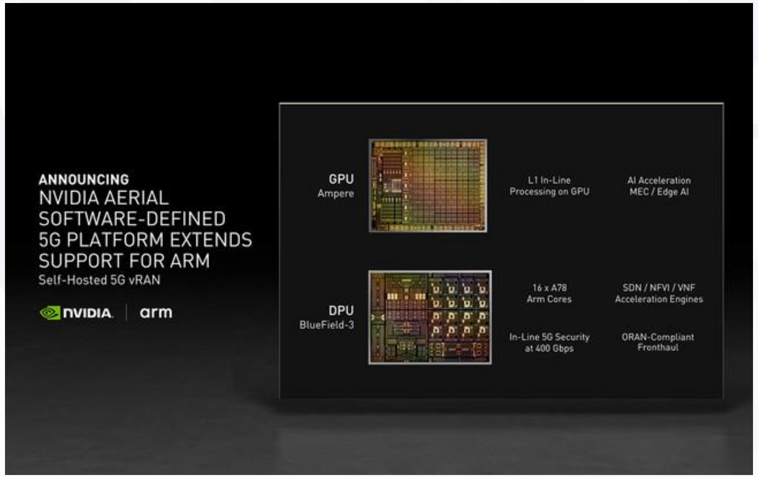
✓High control overhead limits scalability



□X86 Server/GPGPU:

✓ Massive computing capability

✓High energy consumption

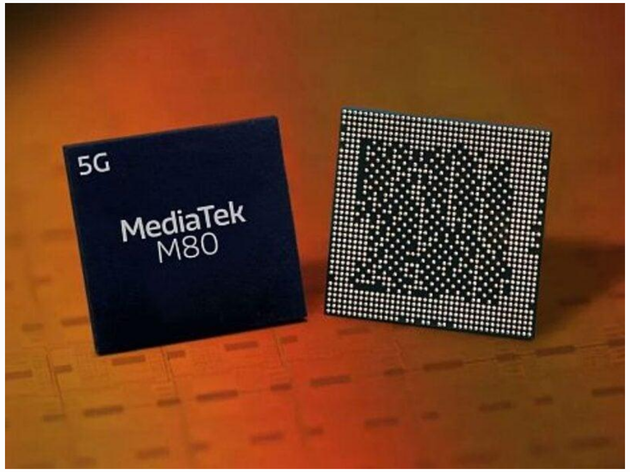


ANNOUNCING
NVIDIA AERIAL
SOFTWARE-DEFINED
5G PLATFORM EXTENDS
SUPPORT FOR ARM
Self-Hosted 5G vRAN

□ASIC:

✓Best PPA

✓Long time to market



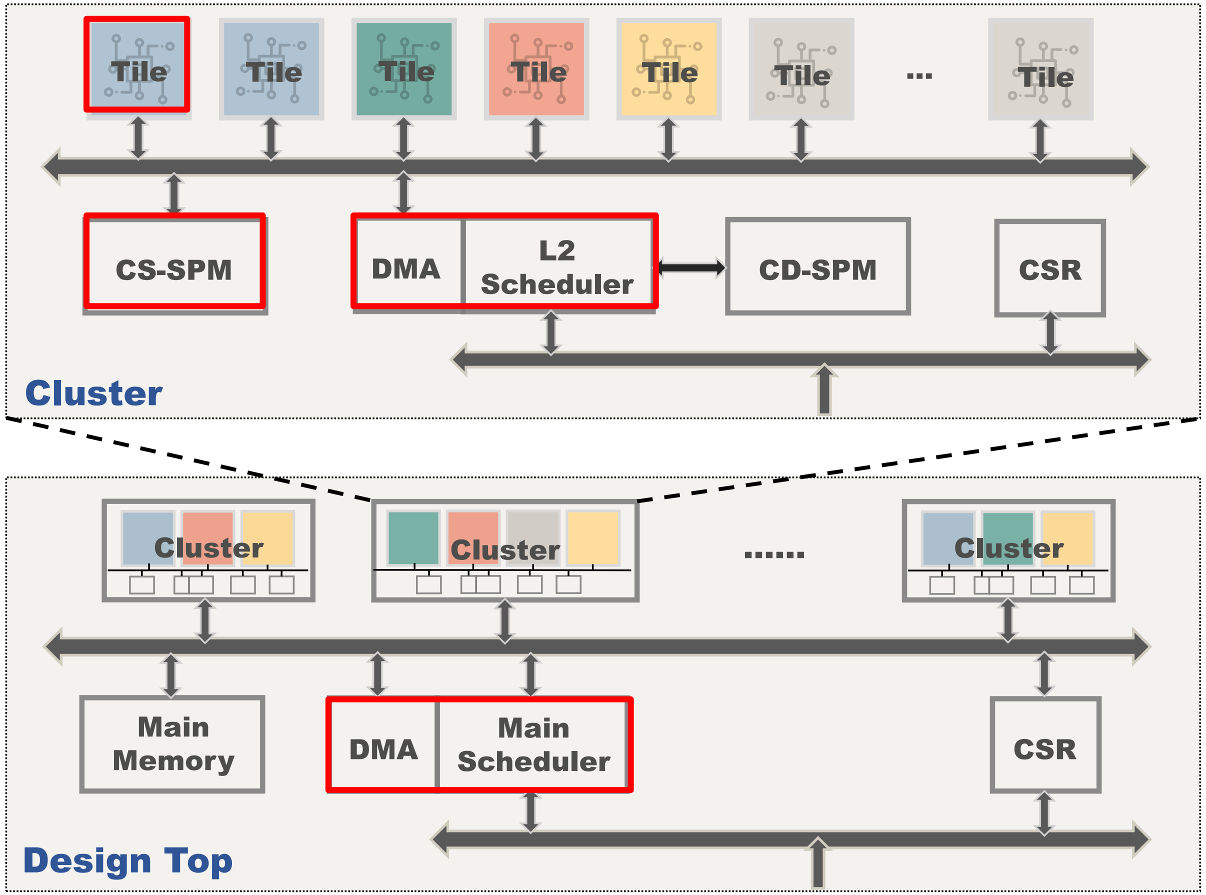
Related Works

- Various works have been presented in academia seeking a way towards manycore parallel computing for WBP.

	Work	Core Heterogeneity	Scalability	DLP	TLP	HW/SW Co-design
GPPs	Sora	☹	🙂	🙂	🙂	☹
Sys-level Analyses	TeraPool	☹	🙂	🙂	-	🙂
	SPECTRUM	☹	🙂	☹	🙂	☹
NoCs	MACRON	🙂	🙂	🙂	-	🙂
	MAGALI	🙂	🙂	☹	-	☹
ASIP	DXT501	🙂	☹	🙂	🙂	🙂
	Ours	🙂	🙂	🙂	🙂	🙂

System Design: Architecture

- **Tile:** RV32IM core with customized vector extension & local scratchpad memory.
- **L2 DMA:** Orchestrating Tiles via a scalar scheduler.
- **CS-SPM:** Swap space for cluster.
- **Main scheduler:** Managing high-level scheduling; Directing the main DMA engine to transfer data between main memory and the clusters.



System Design: *Pack-n-Ship*

ACE-Lab

■ NUMA Approach:

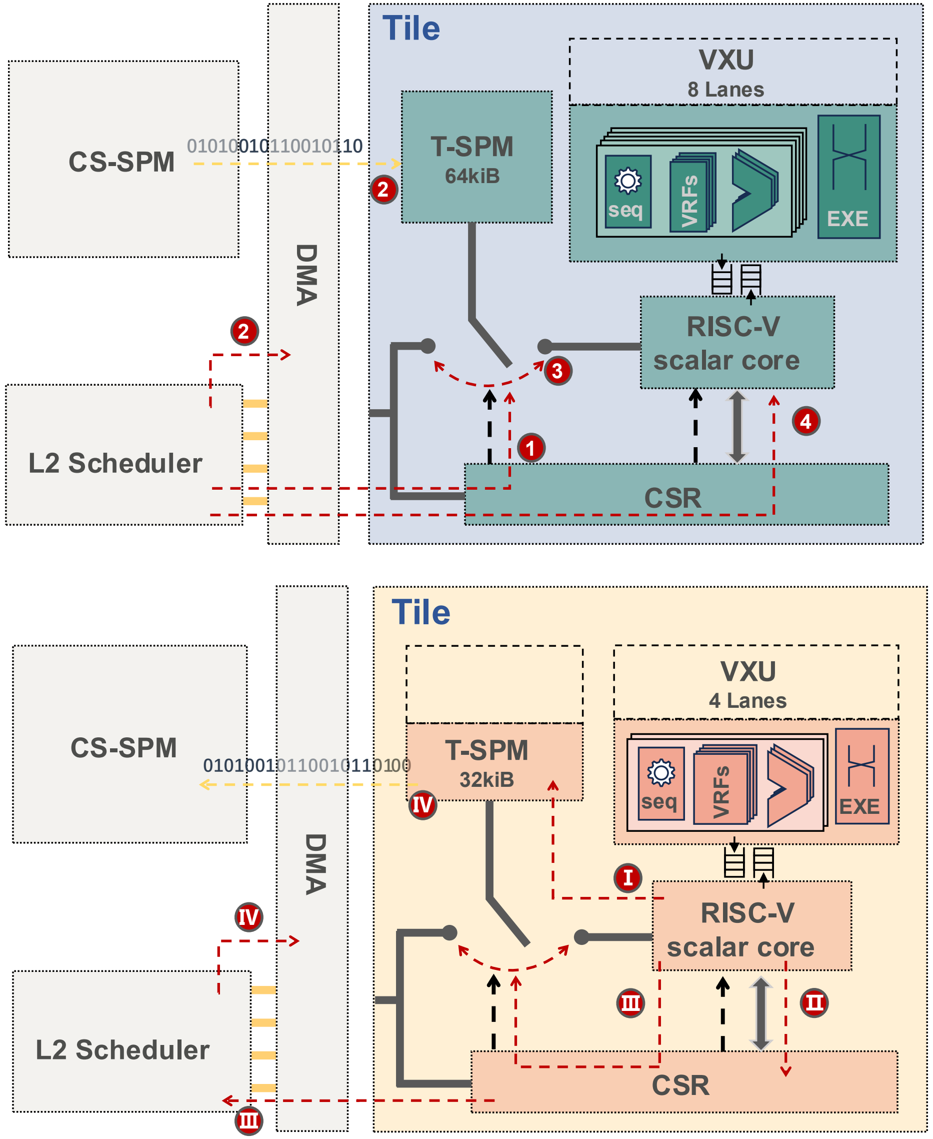
- □ Each tile and cluster has its own SPM, accessed by *outside* DMA.
- □ Eliminating fragment memory access.

■ Before Execution

- □ Alter T-SPM direction by atomic instructions.
- □ DMA moves data from CS-SPM to T-SPM.
- □ Change back T-SPM direction to the RV core.
- □ De-assert core reset.

■ After Execution

- □ Store results in T-SPM.
- □ Notify attributes of return value to CSRs.
- □ Alter T-SPM direction & Issue an interrupt.
- □ DMA retrieve data back to CS-SPM.



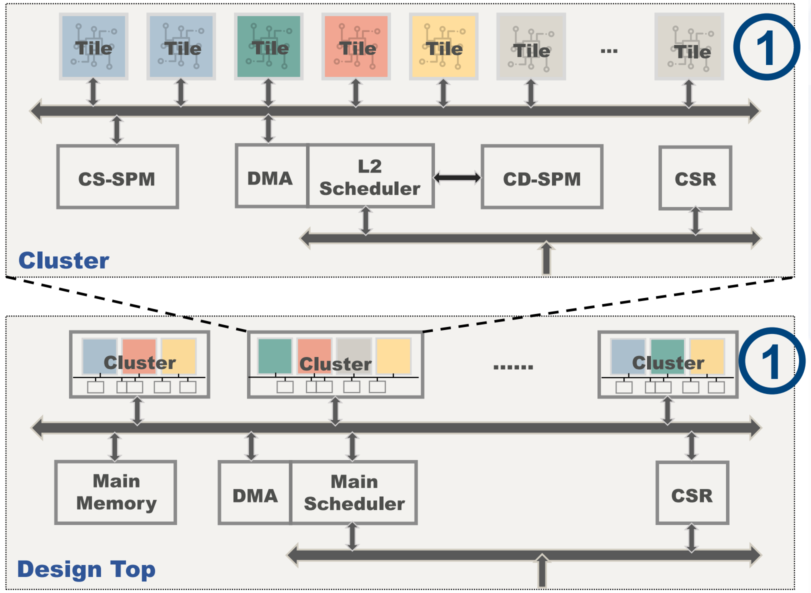
System Design: Heterogeneous Configuration

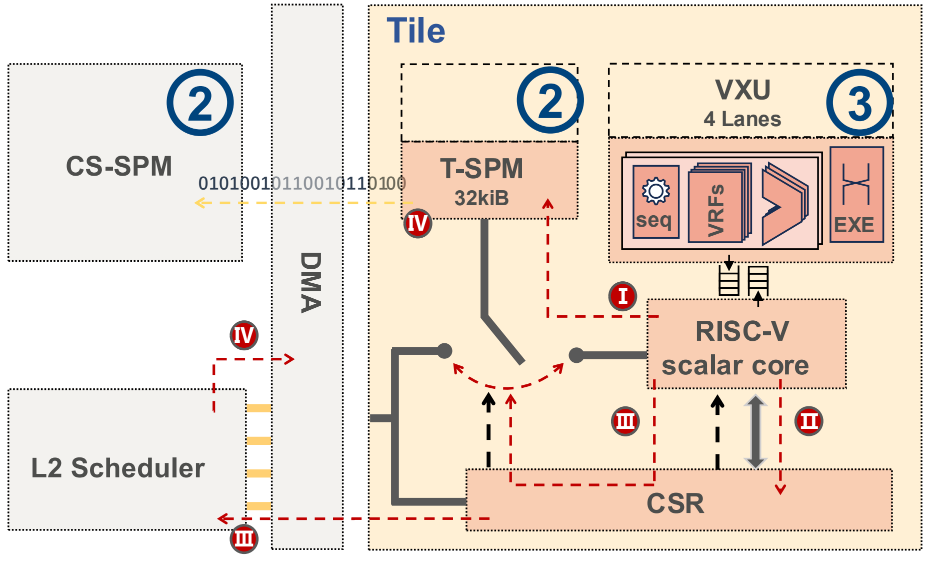
■ Configurable dimensions for WBP

① **# of clusters & tiles:** Enhancing thread- & task-level parallelism of processing Tx & Rx in consecutive time slots – dependent on protocol throughput.

② **SPM footprint:** Dependent on computation type: FFT, Polar decoding; Multi-threading capability.

③ **# of lanes and VRFs:** Enhancing DLP capabilities.



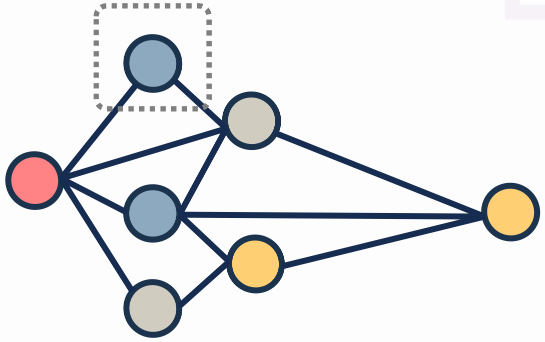


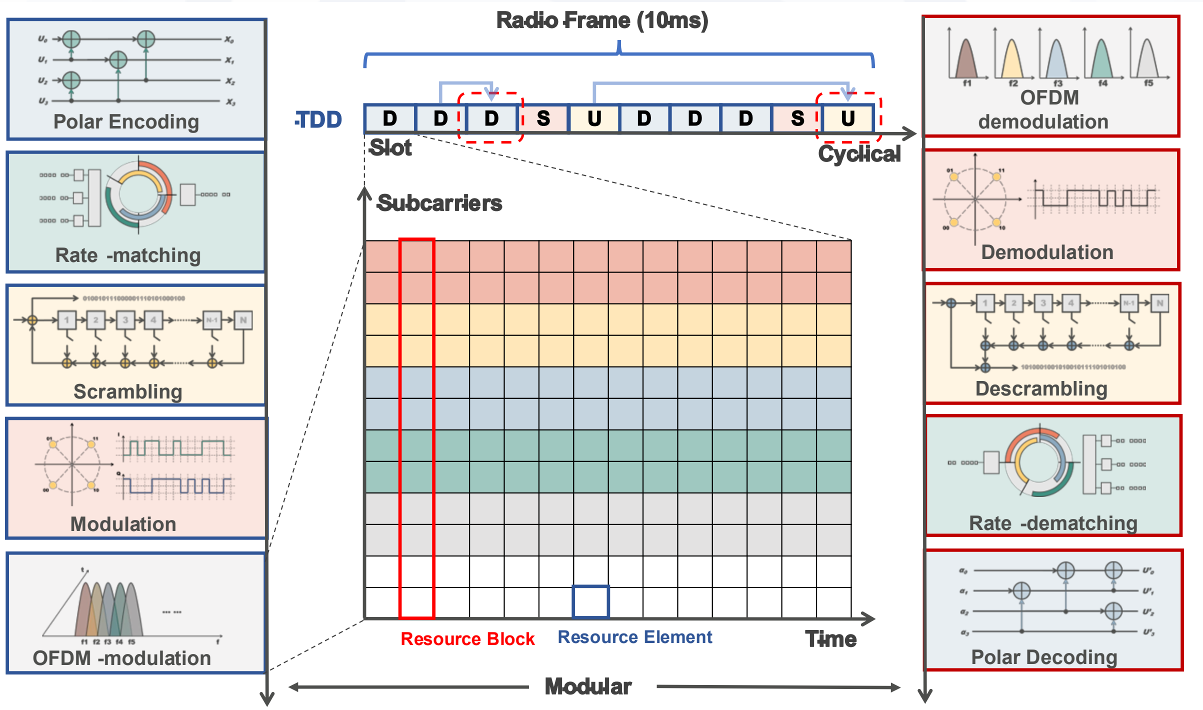
Execution Model: Dataflow Model

■ WBP interpreted as DAGs

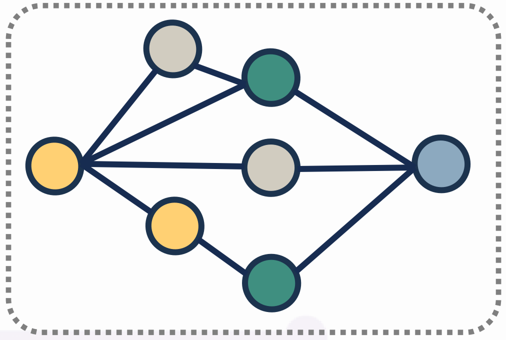
- □A subsequent module is activated only when all preceding tasks are complete.
- □WBP follows a consistent flow over time, enhancing data locality as the DAG information is unlikely to be reconfigured on the hardware.
- □Less scheduler-bounded

Task: A module running on a tile.



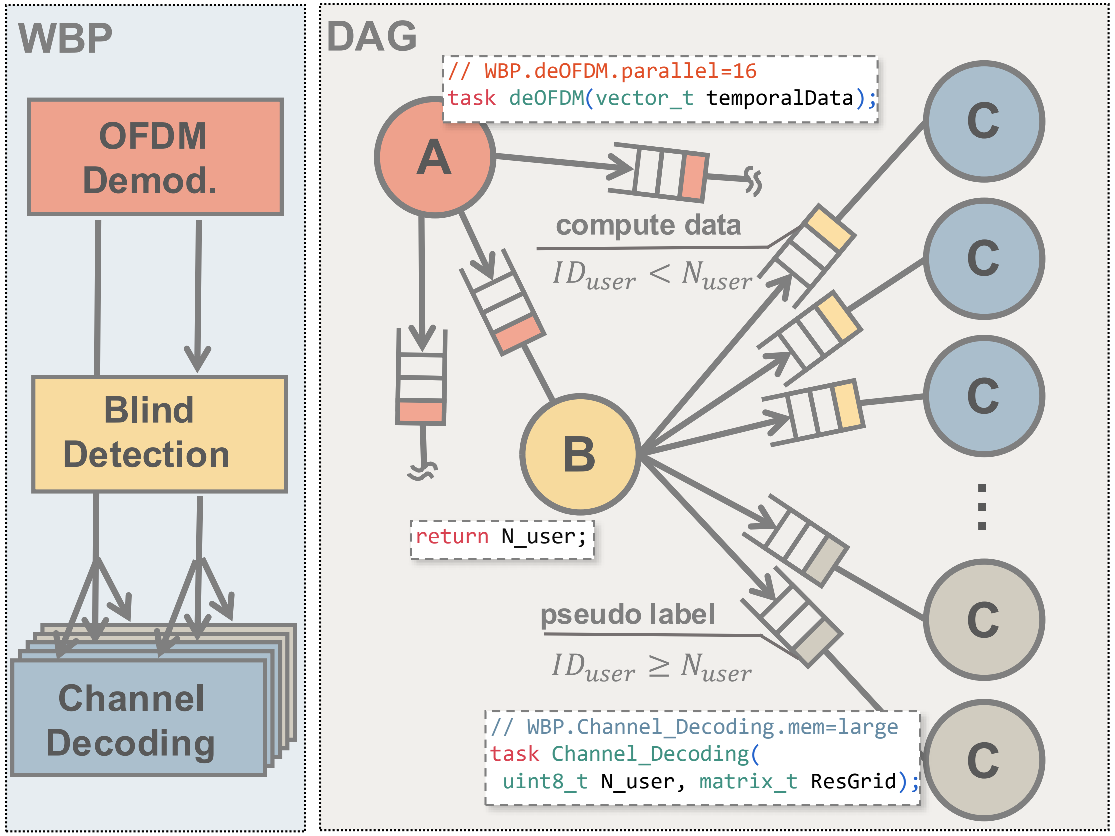


Thread: Several related tasks, representing a complete transmit or receive processing flow.



Execution Model: Dataflow Model

- ■ Attributes guide the scheduler in selecting the most suitable tile for deployment.
- ■ Runtime adjustment of DAGs
  - □ Tasks can be dismissed on-the-fly once the worst-case DAG is determined.
  - □ If the blind detection task detects fewer users, the computational burden can be reduced.



```
// WBP.deOFDM.parallel=16
task deOFDM(vector_t temporalData);
```

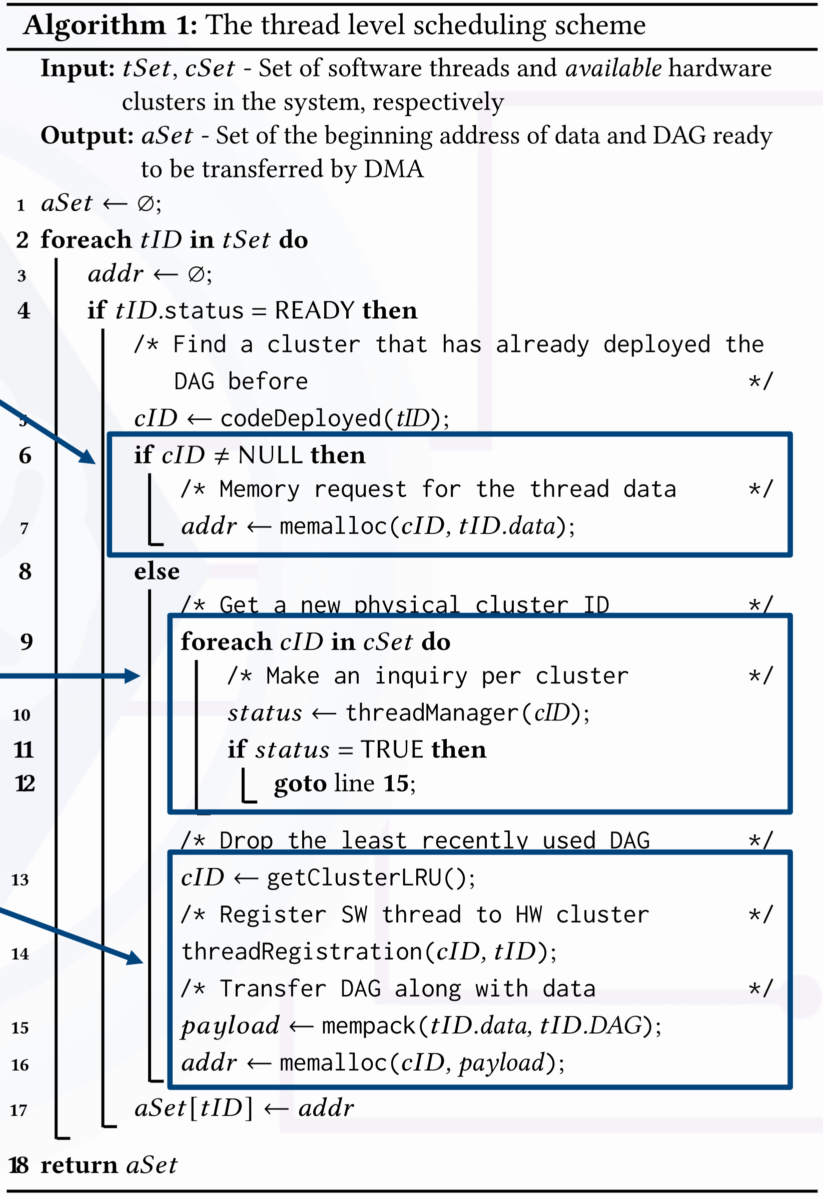
```
return N_user;
```

```
// WBP.Channel_Decoding.mem=large
task Channel_Decoding(
 uint8_t N_user, matrix_t ResGrid);
```

Execution Model: Multi-Level Scheduling

■ Thread-Level Lazy-Deletion

- □ Does not immediately free up DAG memory. Checks whether the DAG has already been deployed to a cluster and only transfers the data for the next thread issue.
- □ Checks the validity for running multiple threads within a single cluster.
- □ Drop the least recent used DAG when all available computing resources are occupied.



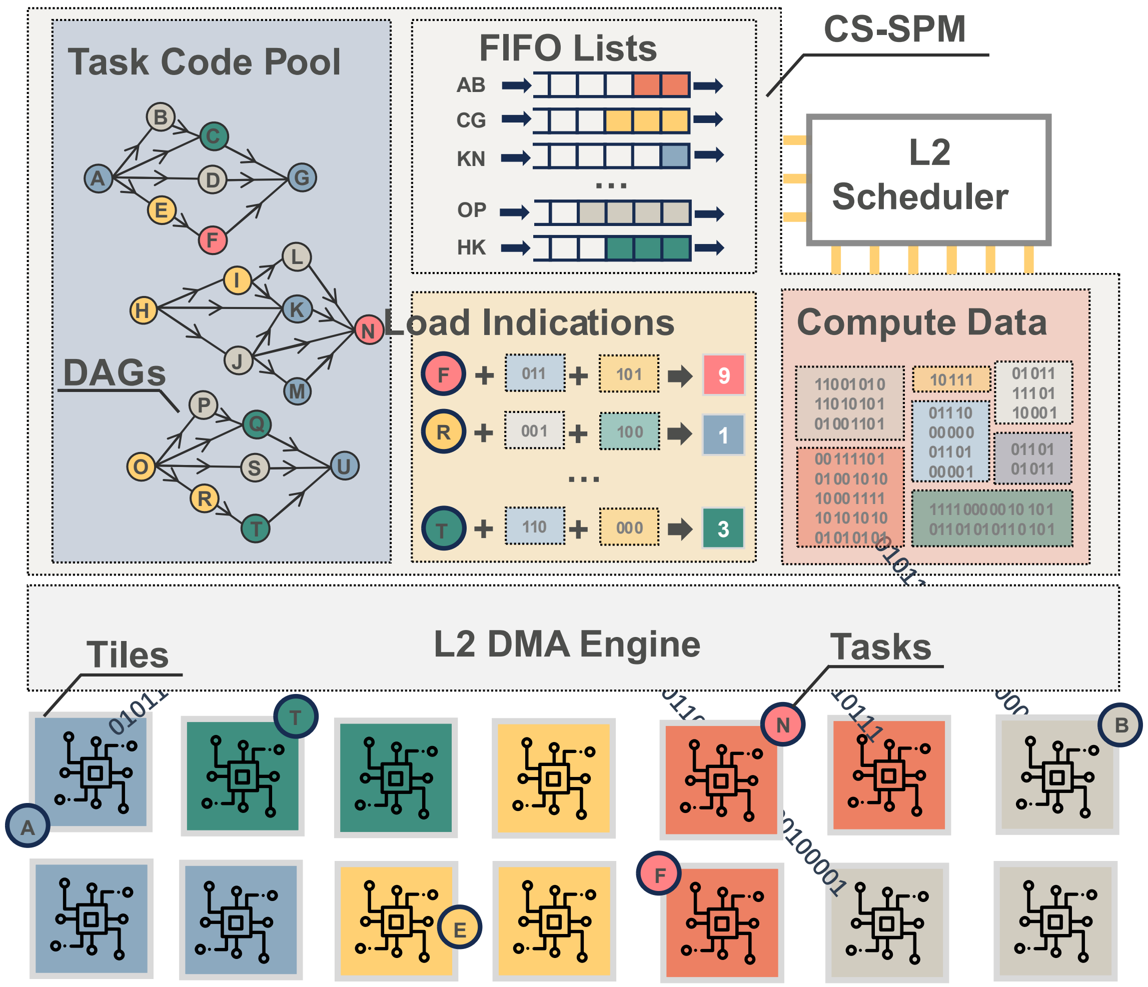
Algorithm 1: The thread level scheduling scheme

```
Input: tSet, cSet - Set of software threads and available hardware
       clusters in the system, respectively
Output: aSet - Set of the beginning address of data and DAG ready
        to be transferred by DMA
1  aSet ← ∅;
2  foreach tID in tSet do
3      addr ← ∅;
4      if tID.status = READY then
           /* Find a cluster that has already deployed the
              DAG before                                        */
5          cID ← codeDeployed(tID);
6          if cID ≠ NULL then
               /* Memory request for the thread data            */
7              addr ← memalloc(cID, tID.data);
8          else
               /* Get a new physical cluster ID                 */
9              foreach cID in cSet do
                   /* Make an inquiry per cluster               */
10                 status ← threadManager(cID);
11                 if status = TRUE then
12                     goto line 15;
               /* Drop the least recently used DAG              */
13             cID ← getClusterLRU();
               /* Register SW thread to HW cluster              */
14             threadRegistration(cID, tID);
               /* Transfer DAG along with data                  */
15             payload ← mempack(tID.data, tID.DAG);
16             addr ← memalloc(cID, payload);
17         aSet[tID] ← addr
18 return aSet
```

Execution Model: Multi-Level Scheduling

■ Tile-Level Scheduling

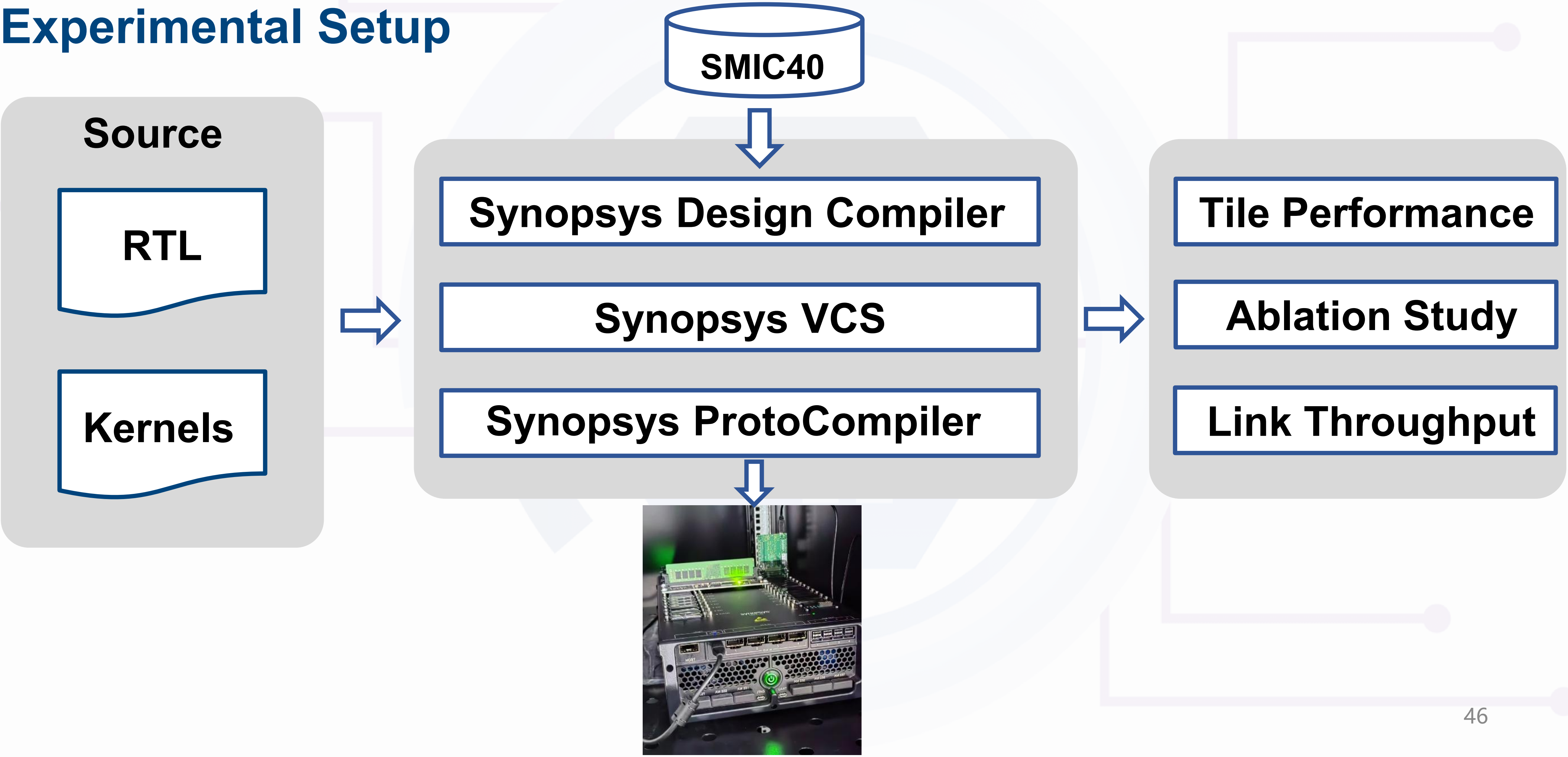
- □ **L2 Scheduler:** Processes the nodes (tasks) in the **task code pool** and checks their readiness through the FIFO queues.
- □ **FIFO Lists:** Track edges between the DAG nodes.
- □ **Load Indications:** Task to be processed and the preferred tile.
- □ **L2 DMA:** Transfers data from the **compute data** section to the heterogeneous tiles



Evaluation

ACE-Lab

■ Experimental Setup



SMIC40

Source

RTL

Kernels

Synopsys Design Compiler

Synopsys VCS

Synopsys ProtoCompiler

Tile Performance

Ablation Study

Link Throughput

Evaluation

Single-Tile Performance

- @ 500 MHz
- Lies between commercial hardware and ASICs
  - 2.3x in FFT & 2x in BP

Config: A 64-lane, 50.1 GOPS VXU

Kernel	Platform	FFT Length	Clock Cycles
FFT	DSP	128	588
		512	2559
		2048	11922
	HW Accel.	128	211
		512	845
		2048	3875
	Ours	128	251
		512	1122
		2048	5073

Kernel	Platform	Dec. Length	Norm. Thrpt.
BP Decode	GPU	512	0.25
		1024	0.21
	ASIC	1024	15.23
	Ours	512	0.54
		1024	0.53

Evaluation

■ Ablation Study

- □ 12T vs. 3C4T-2L2S
- □ 6.5% power increase; 1.3x throughput
- □ Under-utilization in single-level arch.
- □ 6.4% and 9.5% gain under lazy-deletion

Config:

Large (L) Tile (T): w/ 64-lane VXU, 32 VRFs

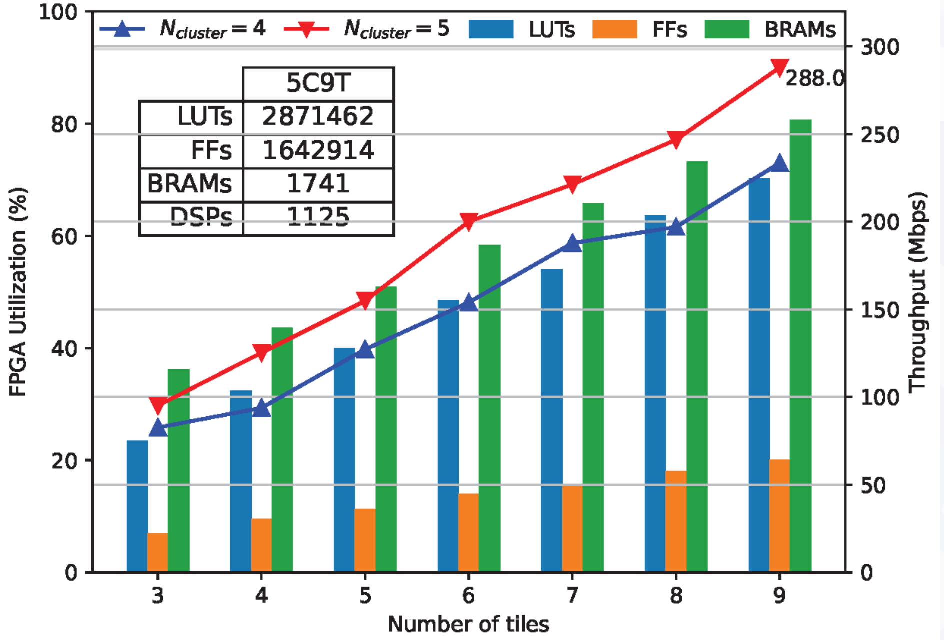
Small (S) Tile: w/ 8-lane VXU, 64VRFs

Baseline + extra features	**Power (W)**		**Throughput (Mbps)**	
	12T	**3C4T**	**Single-Level**	**Multi-Level**
12T / 3C4T arch.	3.24	3.45	8.5	21.2
+ Multi-Threading			64.1	84.7
+ Lazy-Deletion			68.5	93.6

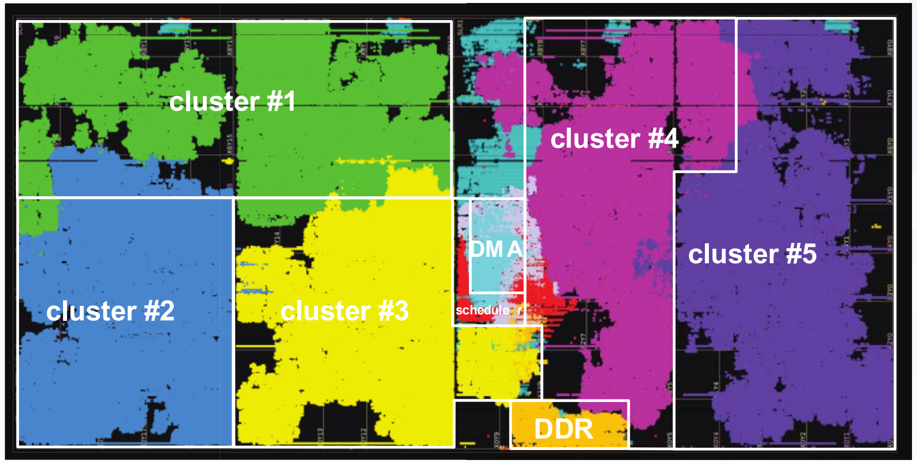
Evaluation

■ Link throughput experiment on prototype

- □5C9T
- □288Mbps



	5C9T
LUTs	2871462
FFs	1642914
BRAMs	1741
DSPs	1125



Module	Configuration
Channel Coding	Polar Codes
Rate-Matching	RV0
Scrambling	Gold Sequence
Modulation	QPSK
OFDM	128 subcarriers
Channel Estimation	Least Squares
Channel Equalization	Zero Forcing
Channel Decoding	Min-Sum BP

Outline

- Background & Motivation
- Related Works
- Echo: An Open-Source 5G/4G/GNSS/LoRa/AI Library
- Venus: A Multi-Core Dataflow-Driven RISC-V Domain Specific Architecture and Implementation on 40nm CMOS
- **Zoozve: A Strip-Mining-Free RISC-V Vector Extension Compiler**
- Conclusion

Zoozve: A Strip-Mining-Free RISC-V Vector Extension

ACE-Lab

□ Strip-Mining-Free:

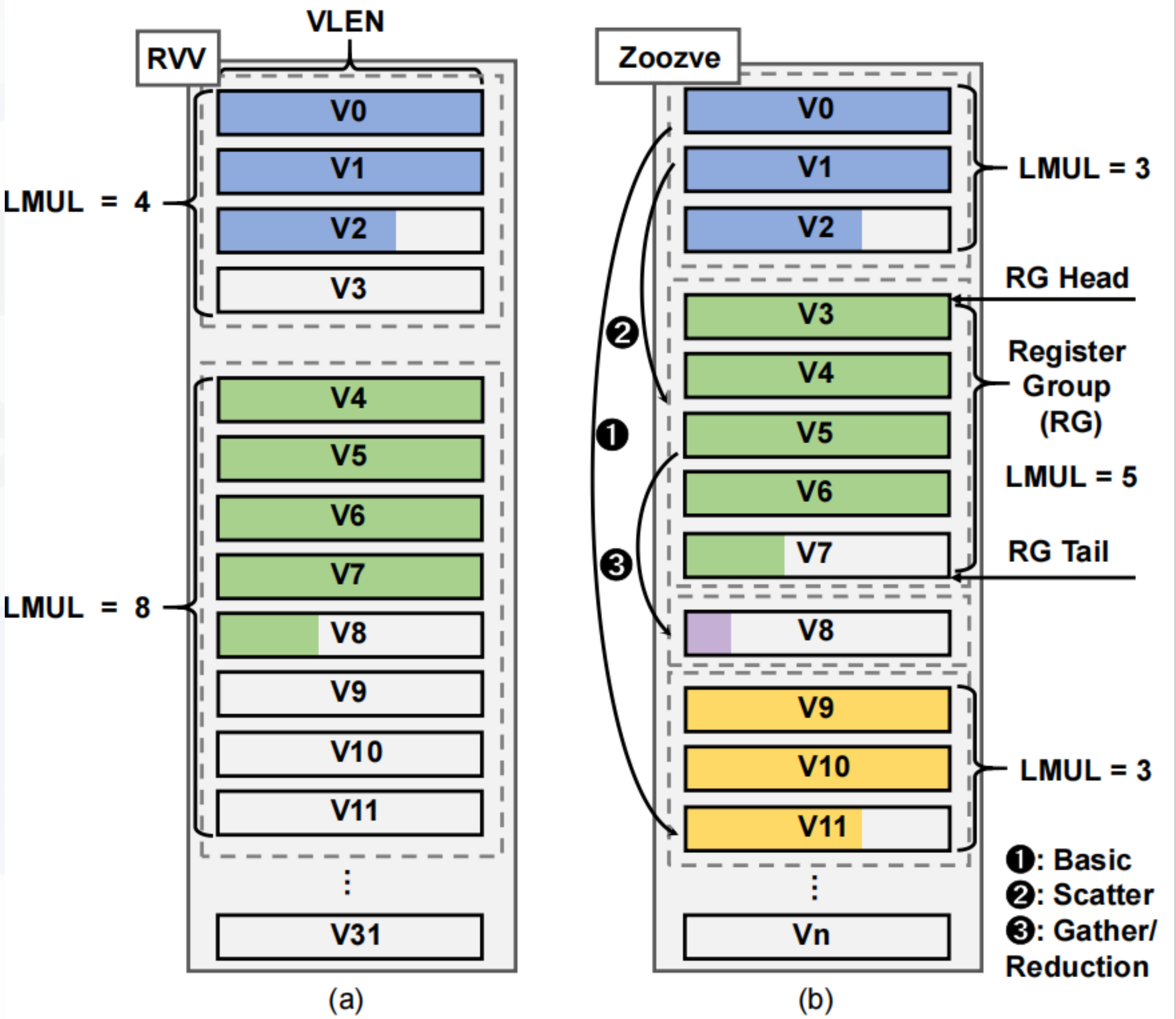
Introduce **asymmetric** instructions for efficient ultra-long vector operations.

□ Arbitrary Register Grouping:

Overcome the limitations of fixed register groups in RVV.

□ Compilation Support, LLVM:

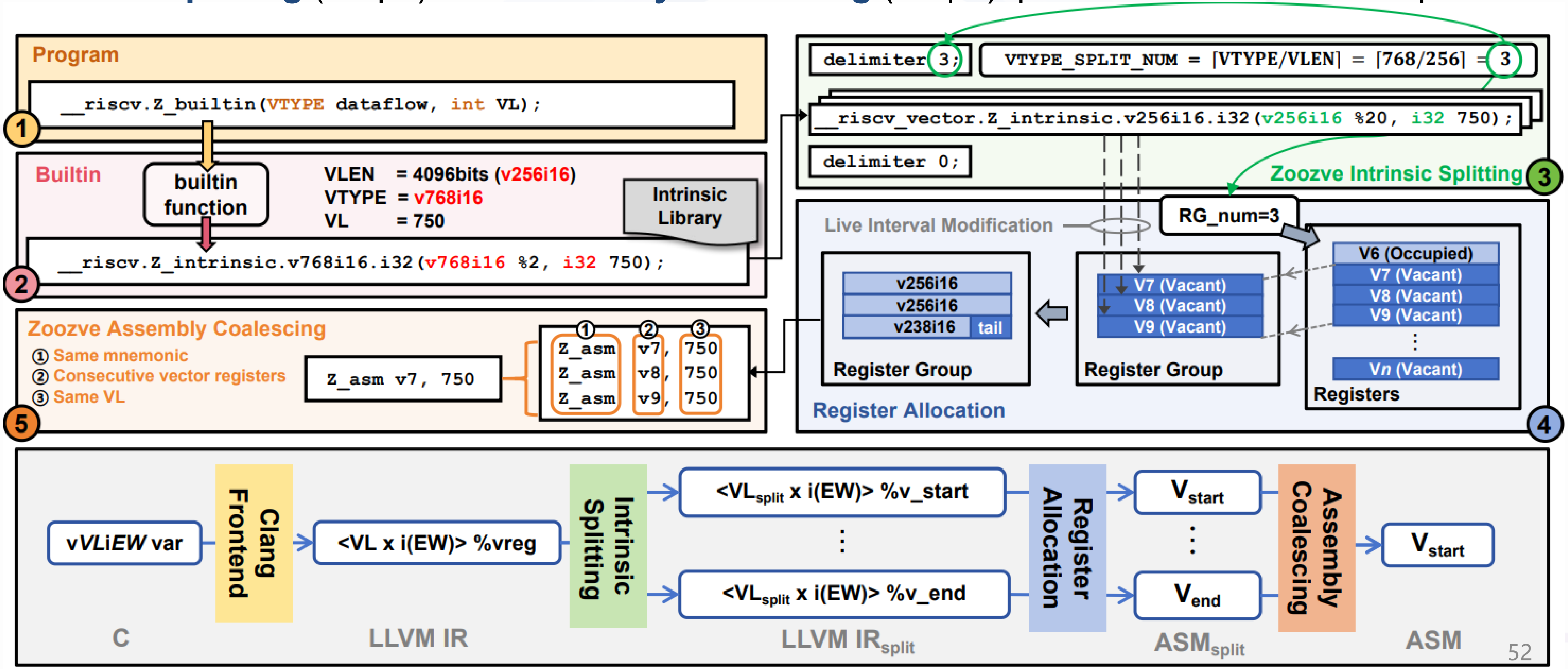
Address the **constraint of the definition of vector types** and add passes to support Zoozve.



Compiler Implementation

■ Compilation Stages

Intrinsic Splitting (Step3) and **Assembly Coalescing** (Step5) passes have been developed.



```
__riscv.Z_builtin(VTYPE dataflow, int VL);
```

```
__riscv.Z_intrinsic.v768i16.i32(v768i16 %2, i32 750);
```

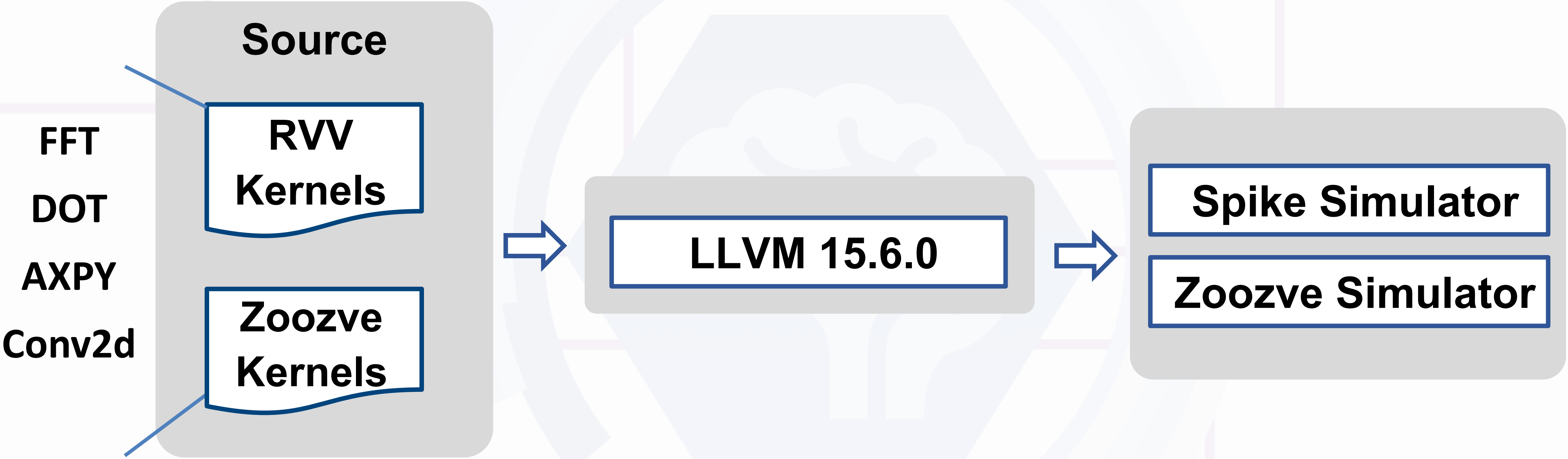
```
delimiter 3;
VTYPE_SPLIT_NUM = ⌈VTYPE/VLEN⌉ = ⌈768/256⌉ = 3
__riscv_vector.Z_intrinsic.v256i16.i32(v256i16 %20, i32 750);
delimiter 0;
```

```
Z_asm v7, 750
```

```
Z_asm v7, 750
Z_asm v8, 750
Z_asm v9, 750
```

Evaluation

■ Experimental Setup



Evaluation

■ Comparison with RVV

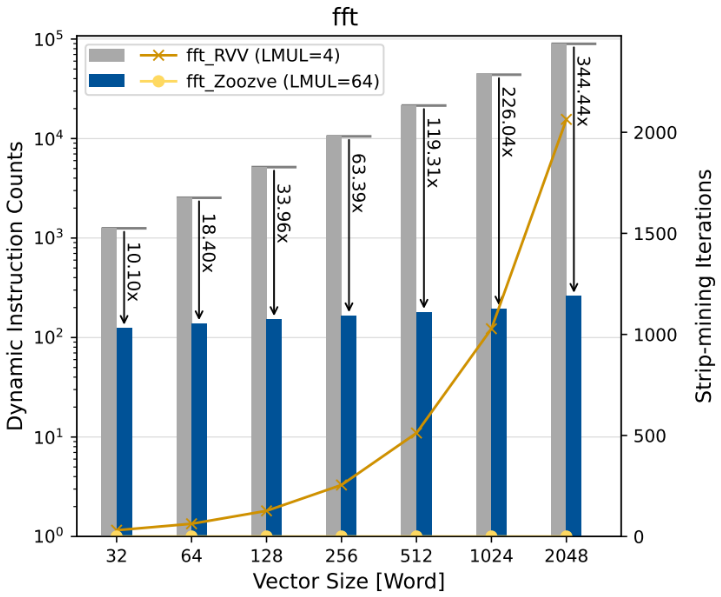
Metric:

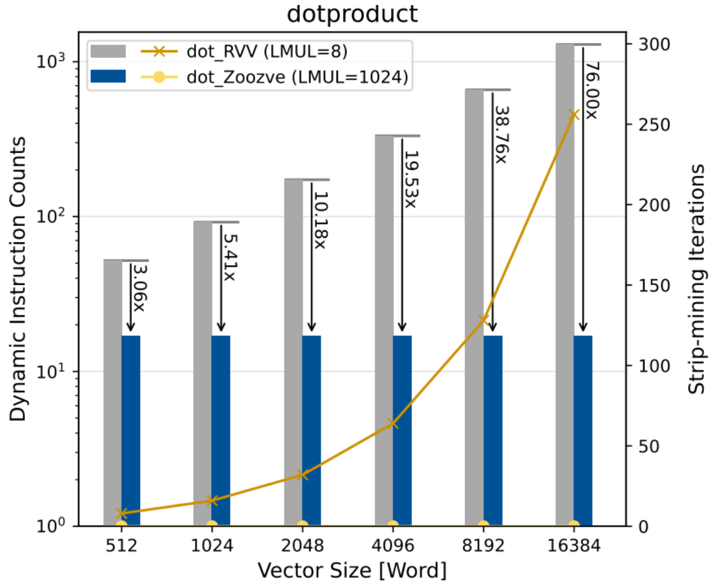
- **Dynamic Instruction Counts**
- **Strip-mining Iterations**

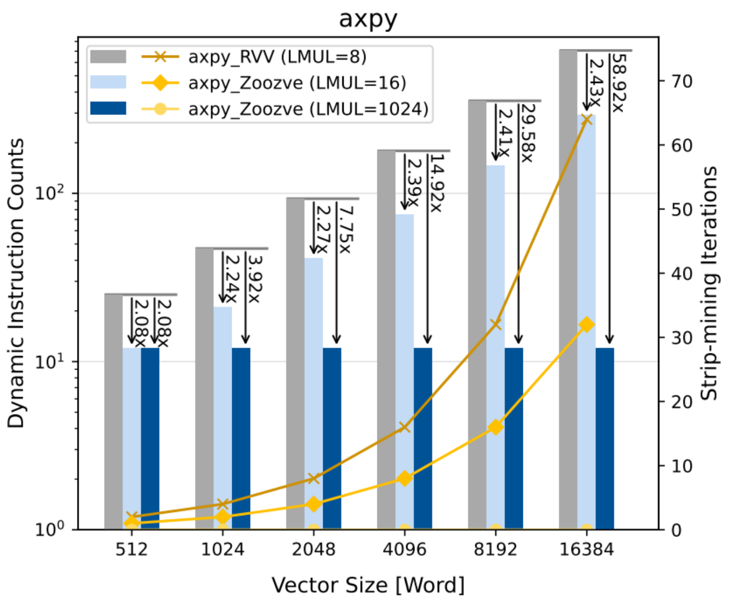
FFT : 344.44× speedup

DOT : 76× speedup

AXPY : 58.92× speedup







Evaluation

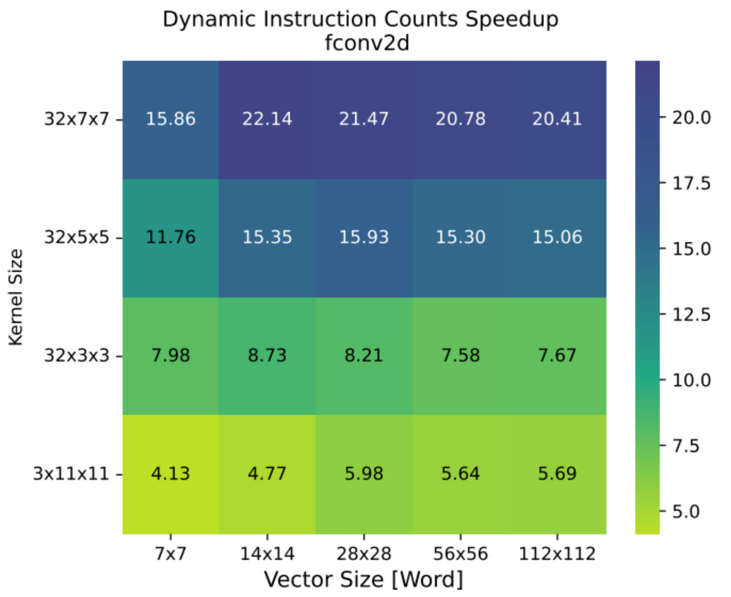
■ Comparison with RVV

Metric:

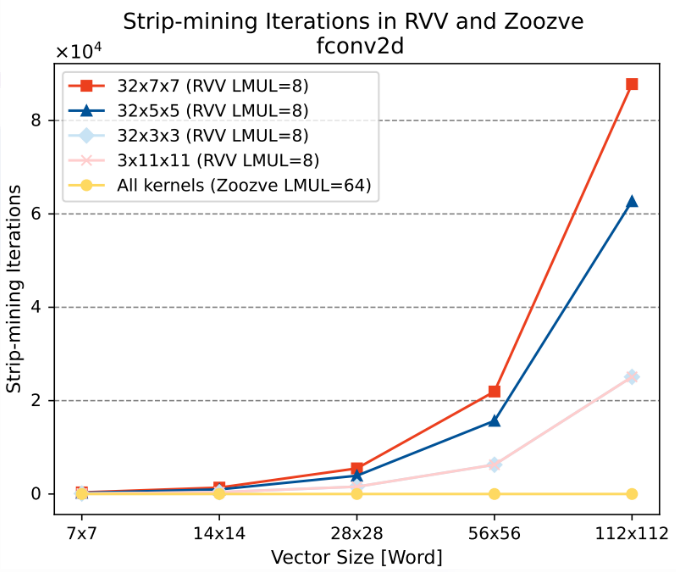
- **Dynamic Instruction Counts**
- **Strip-mining Iterations**
- **Optimal Register Utilization**

2D Convolution: 32×7×7

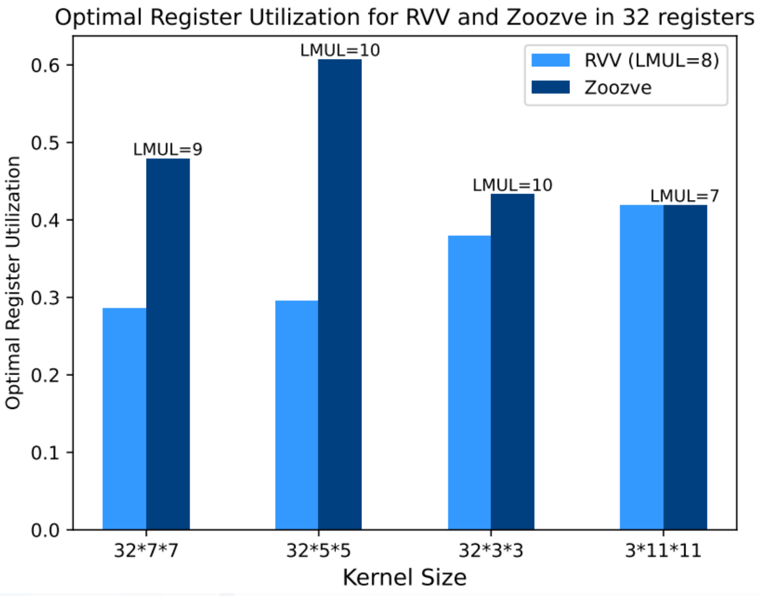
20.41× speedup



Dynamic Instruction Counts Speedup
fconv2d



Strip-mining Iterations in RVV and Zoozve
fconv2d

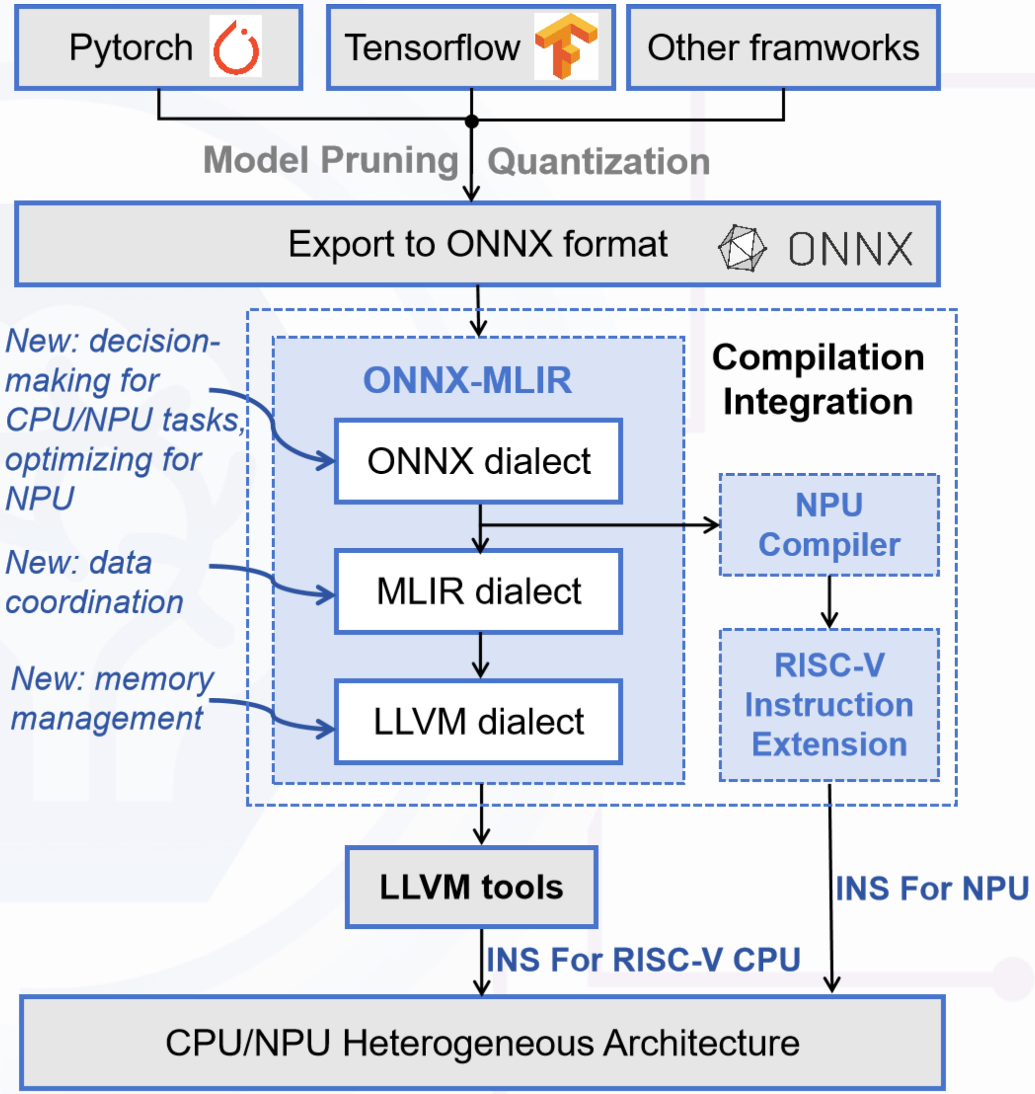


Optimal Register Utilization for RVV and Zoozve in 32 registers

Heterogeneous Compilation: overview

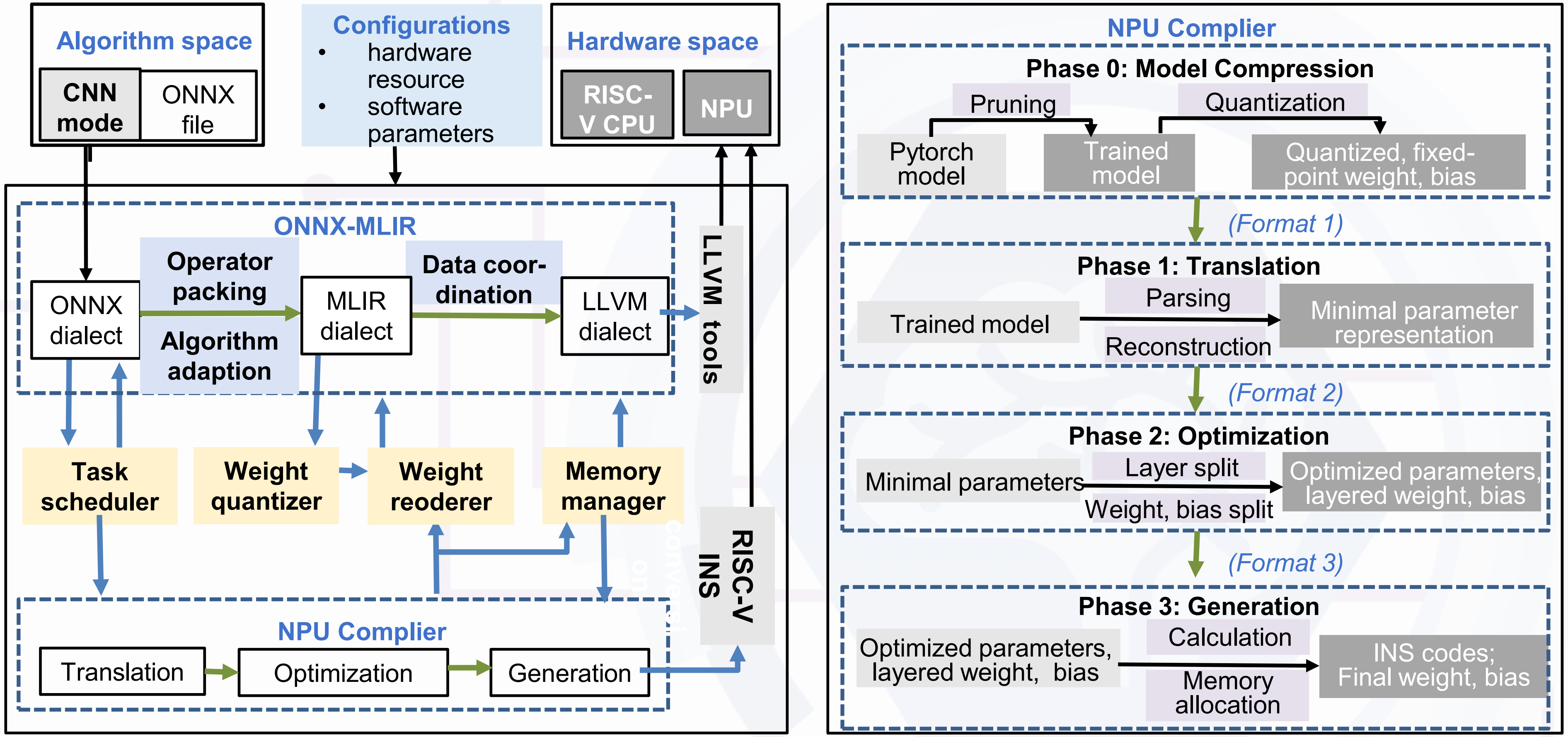
■ Muilt-Level Integration

- □ **Task Scheduler:** Takes the ONNX dialect as input, partitions tasks between the CPU and NPU based on **hardware constraints** or other specific requirements.
- □ **Data Coordination:** CPU/NPU storage format unification.
- □ **Memory Manager:** Unifies the management of CPU and NPU data spaces, ensuring the correctness of data access across layers while optimizing memory usage.



Heterogeneous Compilation: Framework

ACE-Lab



(a)

(b)

Format 1 ORIGIN

(0) **Conv2d1**(14, 14, 256, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), BN=False, ReLU=relu, MaxPooling=(2,2,2),dilation=0, GAP=(7,7), Conv2d1)
(1)**FC1**(1, 1, 64, 1000, BN=False, ReLU=relu, Conv2d1)
(2)**Conv2d1 + Conv2d2**

Format 2 MINIMAL

(0) **Conv(0**, row, col, in_ch, out_ch, k_x, k_y, pooling, para_in, para_out, data_flow,padding, **BN**, relu, **dilation**, 0, in_id, stride)
(1) **FC(1**, row, col, in_ch, **BN**, relu, in_id)
(2) **Conv(2**, row, col, in_ch, out_ch, k_x, k_y, pooling, para_in, para_out, data_flow, padding, **BN**, relu, **dilation**, 0, in_id, stride, add_id)

Format 3 OPTIMIZED

(0) **Conv(0**, row, col, in_ch, out_ch, k_x, k_y, pooling, para_in, para_out, data_flow, padding, relu, 0, in_id, stride, **split_layers**,**repeat_layers**)
(1) **FC(1**, row, col, in_ch, relu, in_id)
(2) **Conv(2**, row, col, in_ch, out_ch, k_x, k_y, pooling, para_in, para_out, data_flow, padding, relu, 0, in_id, stride, **split_layers**, **repeat_layers**, add_id)

(c)

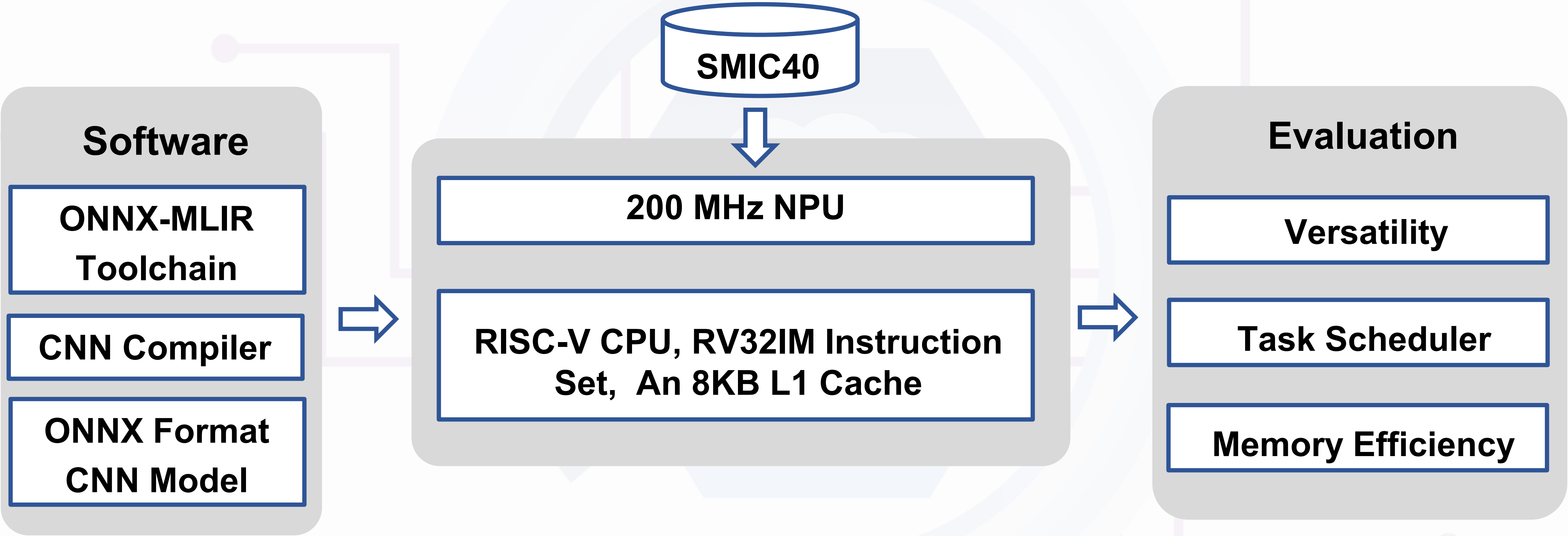
(a) The workflow of our compilation framework;

(b) The workflow of our previous NPU compiler, including translation, optimization, and execution;

(c) The illustration of the multi-level intermediate representation.

Evaluation

■ Experimental Setup



Software

ONNX-MLIR Toolchain

CNN Compiler

ONNX Format CNN Model

SMIC40

200 MHz NPU

RISC-V CPU, RV32IM Instruction Set, An 8KB L1 Cache

Evaluation

Versatility

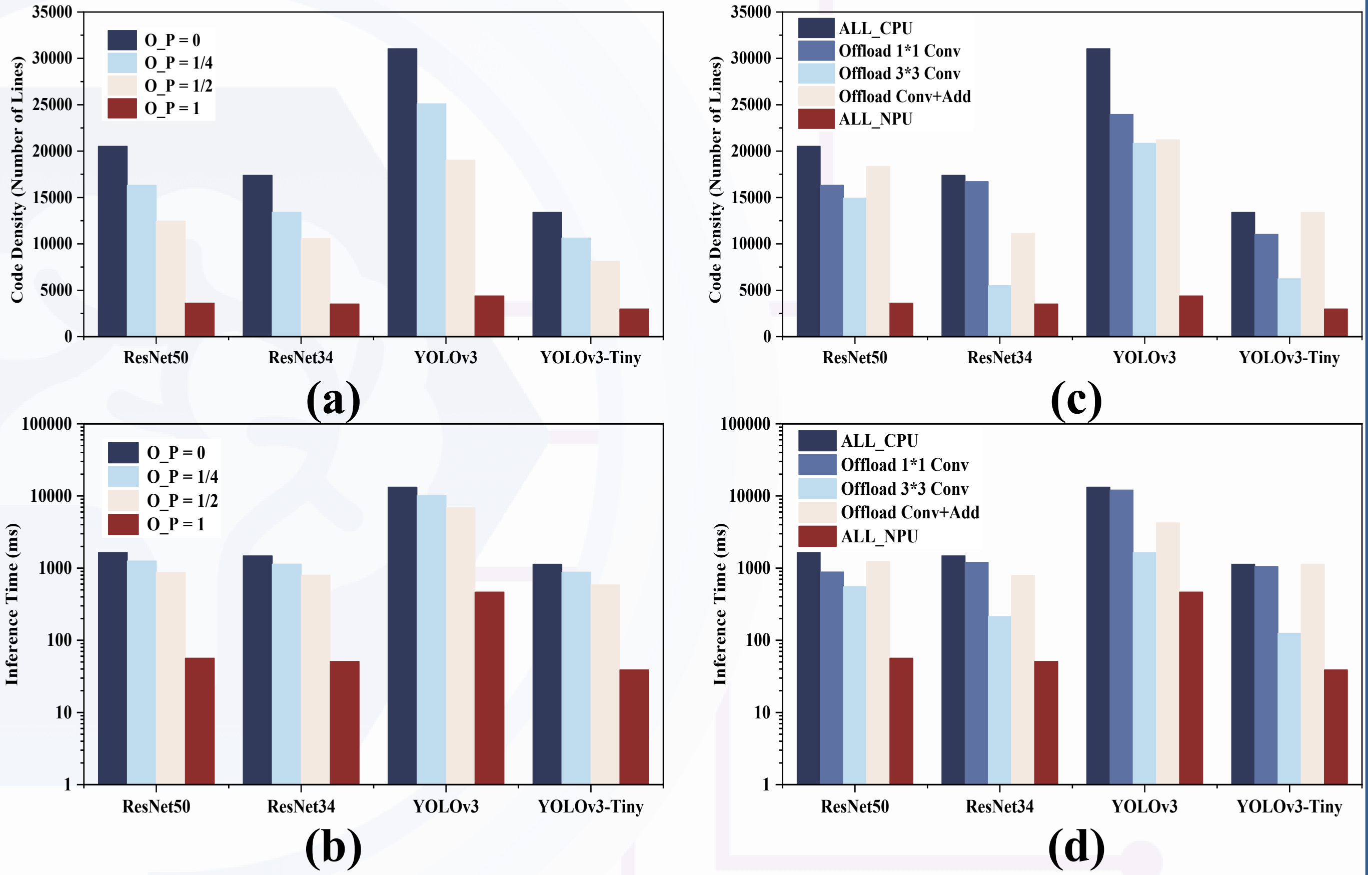
Task Scheduler

Memory Efficiency

Evaluation

■ Experimental Results

- □ **Versatility:** Effectively generates optimized instructions for various models, significantly enhancing the efficiency and performance of heterogeneous architectures.
- □ **Performance:** Achieves up to a **7.06×** reduction in **code density** and up to a **5.58×** improvement in **memory usage**, bridging the computational gap between CPUs and NPUs.



(a) (c)

(b) (d)

Outline

- Background & Motivation
- Related Works
- Echo: An Open-Source 5G/4G/GNSS/LoRa/AI Library
- Venus: A Multi-Core Dataflow-Driven RISC-V Domain Specific Architecture and Implementation on 40nm CMOS
- Zoozve: A Strip-Mining-Free RISC-V Vector Extension Compiler
- **Conclusion**

Conclusions

- □ Huge concerns over inefficient investment on mobile networks going forward.
- □ The integration of AI and communication has become a consensus for future development.
- □ The limitations of baseband chip architecture and ecosystem have further intensified concerns about the ability to support the continuous evolution of mobile networks.

- □ This study proposes:
  - ✓ A scalable, software-defined, AI-integrated, and sustainable baseband chip architecture built specifically for sustainable evolution of mobile networks
  - ✓ AI and wireless baseband functions can share resources on the computing units level, with short time-to-market and user-friendly programming model
  - ✓ Open-source and complete 5G/GNSS/etc. protocol stack and AI models are available soon…

Thank You

Email: jiangzhiyuan@shu.edu.cn

清華大學
Tsinghua University

上海大学
SHANGHAI UNIVERSITY

先进通信与计算芯片实验室
Advanced Communication and Computing Electronics Lab
(ACE-Lab)

深铁科技
DEEPTRANSCEND