Akaria

RISC-V Days Tokyo 2022 Autumn

NSITEXE Akaria NS72

RISC-V 64-bit Processor with Vector Extension

Koji Adachi
Manager of Processor Development Sect.
NSITEXE, Inc.

NSI-TEXE

Akaria

Presentation Outline

- NSITEXE Akaria
- Compute Core NS72
- Demo

Akaria

NSITEXE Akaria

Explosion of User Applications

Transition of control trends

Open Loop Cont. → Feedback Cont. → Adaptive Cont. → Full Model Cont.

by mechanism & Analog | by parameters | by Model Predictive Control | by Parallel Processing

Transition of Automotive

Hands-on → Hands-off / Eyes-off → Mind-off

Transition of Smart Factory

Automation → IoT → CPS

Akaria

Application Keywords:

Distributed, Metaverse, Connectivity

Key Technology:

Edge-AI, MPC, Security

Computing requirement:

Flexible Processing
Power Efficiency
Real-time Computing
Functional Safety

NSITEXE Akaria Overview

Akaria Software

Akaria Processor

Standard Processors

RISC-V 32bit (RV32)

RISC-V 64bit (RV64)

\+

Extension Units

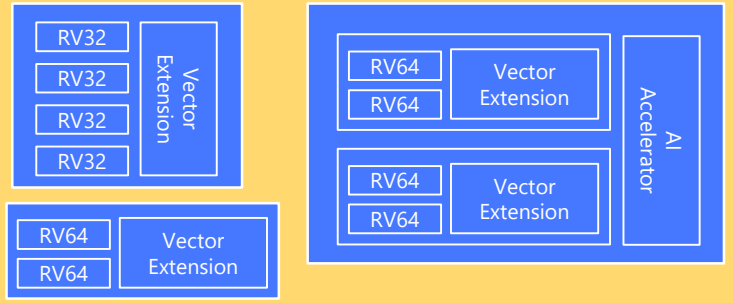
Multi Core — Task Parallelism

Vector Extension — Data Parallelism

AI Accelerator — Neural Network

=

Domain Specific Accelerators (Scalability, Versatility)



Flexible Processing

Power Efficiency

Real-time Computing

Functional Safety

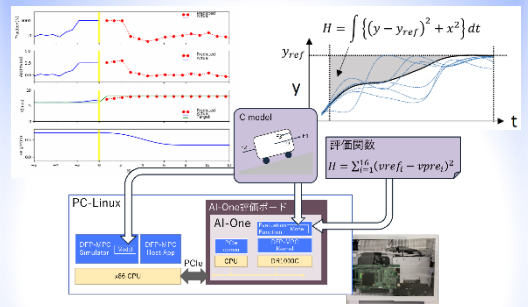
Based on industry demands, Akaria covers various types of applications

RISC-V core is positioned as a key component

Akaria

Akaria Processor and Solution

Advanced Control Model Predictive Control



RV32 RV32 RV32 RV32 Vector

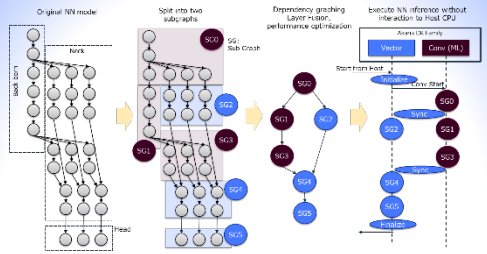
Dedicated MIMD Accelerator

Image Recognition Accelerator Control Plane

RV32 Complex Co-Processor

Control Core for Complex Co-Processor

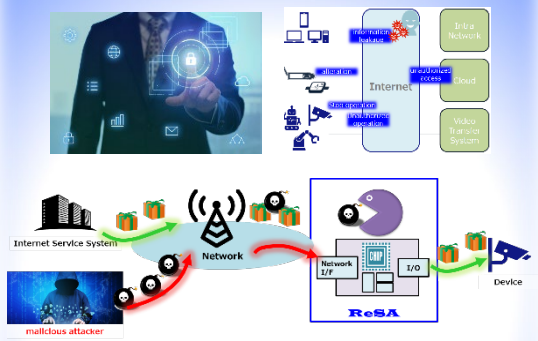
AI accelerator & pre/post-processing



RV64 RV64 Vector AI Acc.

Combination of Vector and AI Acc.

Network Security



Vector NW IP RV32 Secure IPs

Processor Sub-System with 3rd Party IPs

- Applying wide range application by the variety from “control-plane” to “data-plane”
- Already proven in real silicons
- **Each Solution is supported by the standard processor (RV32/RV64)**

Akaria

Akaria RISC-V Core : NS Family

Akaria Standard Processors: NS Family

primitive components to support wide range applications

NS Family Features

- Performance and Efficiency
  - High efficiency with sophisticated μ-architecture refined by excellent architects
  - Variaos line-up and Supporting user customization
- Low-Cost
  - Providing cost-benefits relative to competitors
  - Various tools and software are available to take advantage of the open RISC-V standard
- Quality / Support
  - Functional Safety and high quality supporting Automotive
  - Rapid response due to in-house development

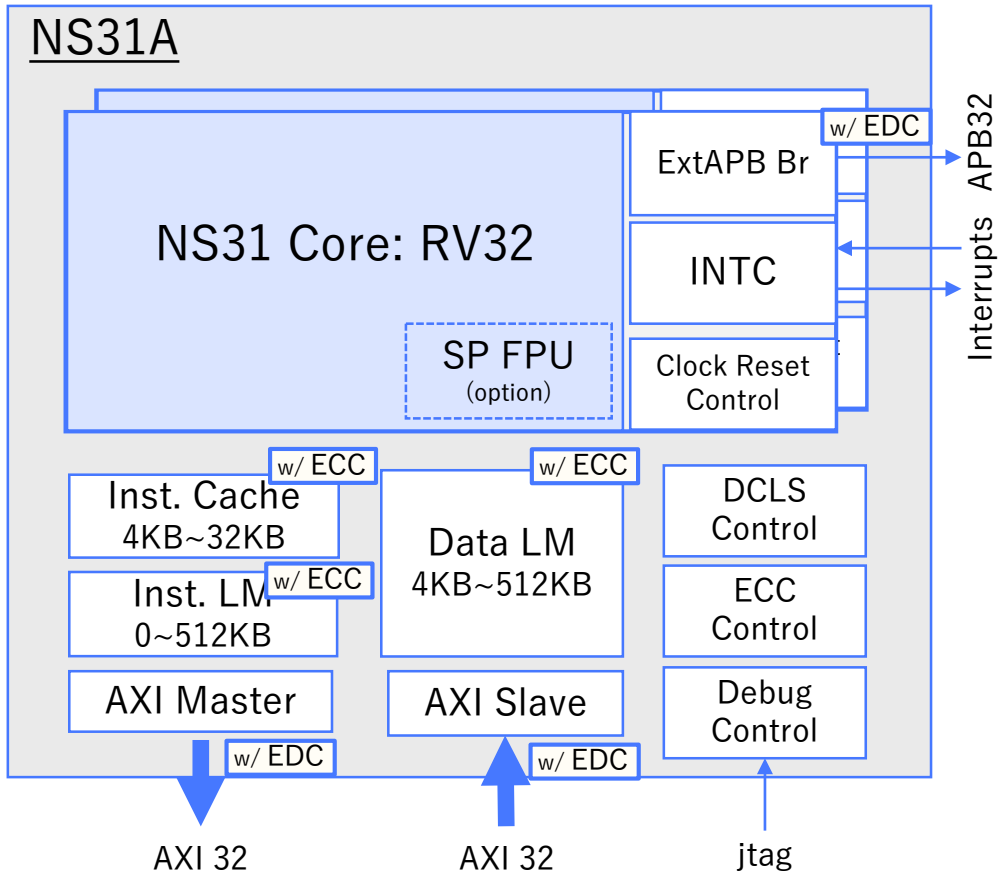
Series of Akaria Standard Processor

- NS7: Performance for data-plane
- NS5: High end for embedded systems
- NS3: High efficiency for control-plane
- NS1: Low power and compact

Series	Base	FuSa	Vector	Rich-OS
NS7	NS72	NS72A	NS72V	NS73
NS5	NS52	Developing	Developing	Developing
NS3	NS31	NS31A		
NS1	NS11	NS11A		

Akaria

NS31 : Efficient Control Core



NS31 features

- Small and efficient microcontroller
  - RISC-V ISA : RV32IMAF
  - 32bit In-order 4-stage pipeline, with BTB
  - Single-Precision Floating Point Unit
- Privilege modes (M-mode and U-mode)
- AMBA compliant Bus Interfaces
  - AXI configuration / AHB configuration

- Internal Memory
  - Instruction Cache
  - Data Local Memory (DLM) *1
  - Instruction Local Memory (ILM) *1

- Interrupt Controller
  - ACLINT/PLIC for a future release
- Debug function

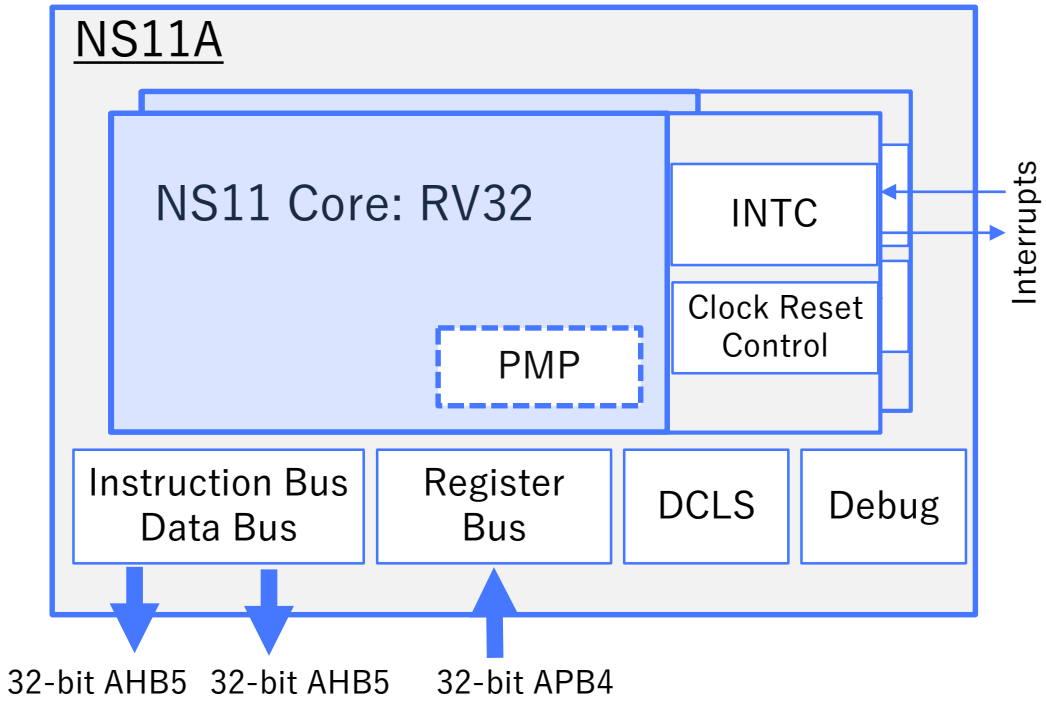
- FuSa
  - Physical Memory Protection
  - Dual-Core Lockstep mechanism
  - ECC memory protection
  - BUS memory protection *1

---

*1 AXI configuration only.

Akaria

NS11 : Low-power Compact Core



NS11 features

- Very small and low power microcontroller
  - RISC-V ISA : RV32IMAC/RV32EMAC
  - 32bit In-order 4-stage pipeline
  - 32bit instruction fetch
- privilege modes (M-mode and U-mode)
- AMBA compliant Bus Interfaces
  - 32bit AHB5 instruction bus interface
  - 32bit AHB5 data bus interface

- Interrupt Controller
  - ACLINT/PLIC for a future release
- Debug function

- FuSa
  - Physical Memory Protection
  - Dual-Core Lockstep mechanism

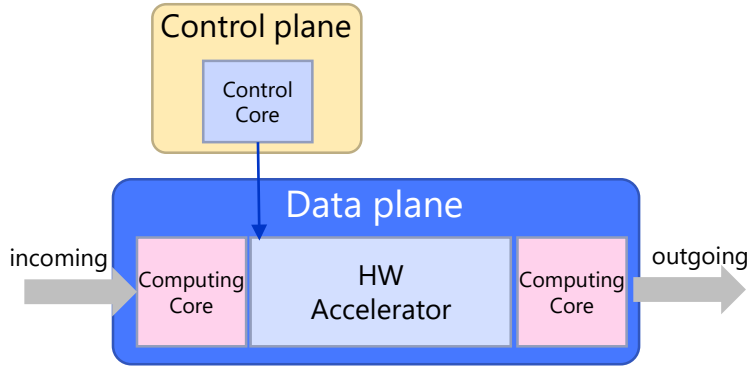
Akaria

Akaria

Computing Core NS72

Compute Core for Data Plane

- Why is a computing core for data plane needed?
- From the controller to the application,
  the software stack becomes complex



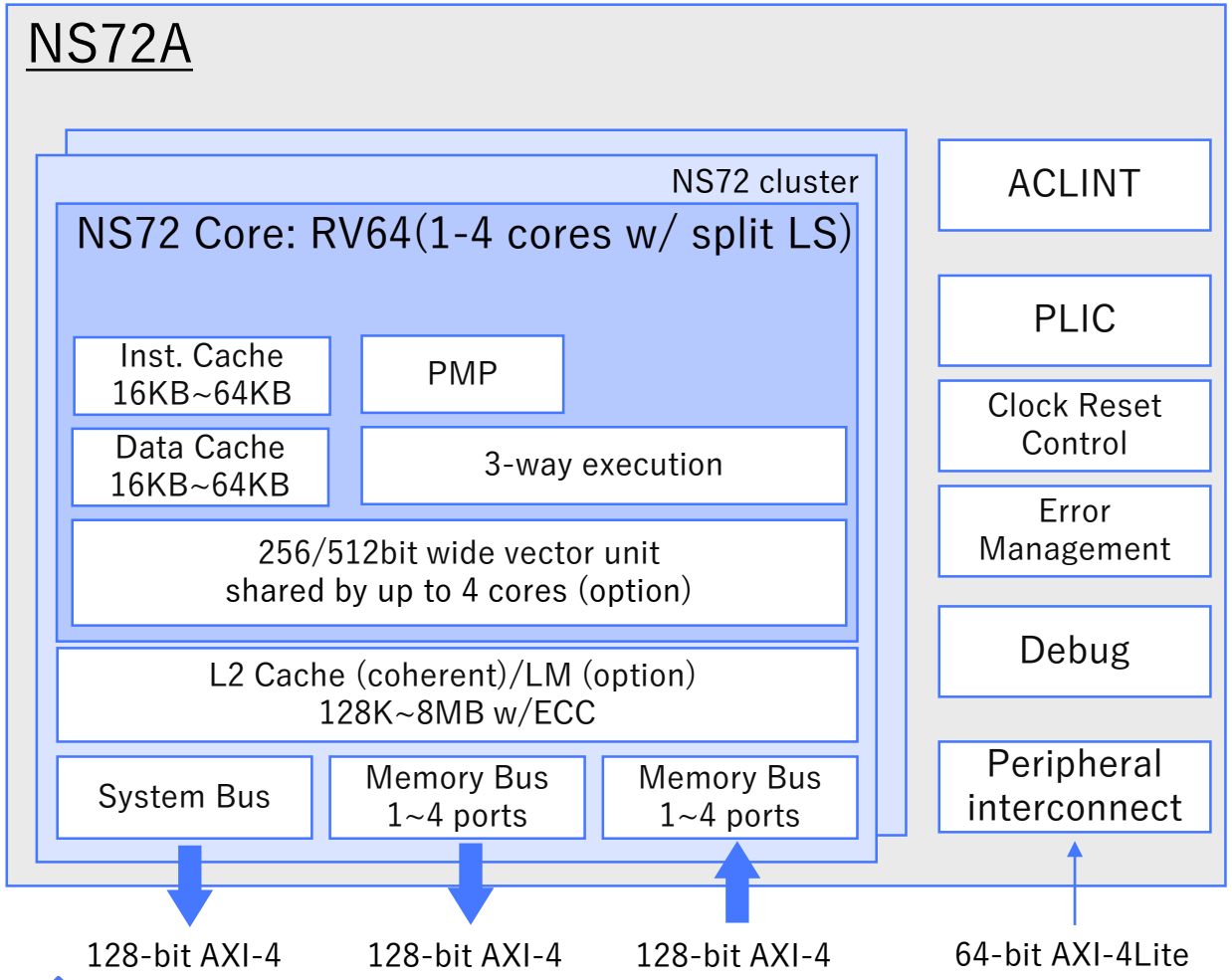
→ Cannot be handled by HW accelerators alone
→ Pre- and post-processing is required to match HW accelerators

- For flexibility, computing core is required
  (e.g. high computing throughput, memory bandwidth, etc.)

For Control Plane	For Data Plane
NS31	NS72
NS11	

Akaria

NS72 : Compute Core for Data Plane

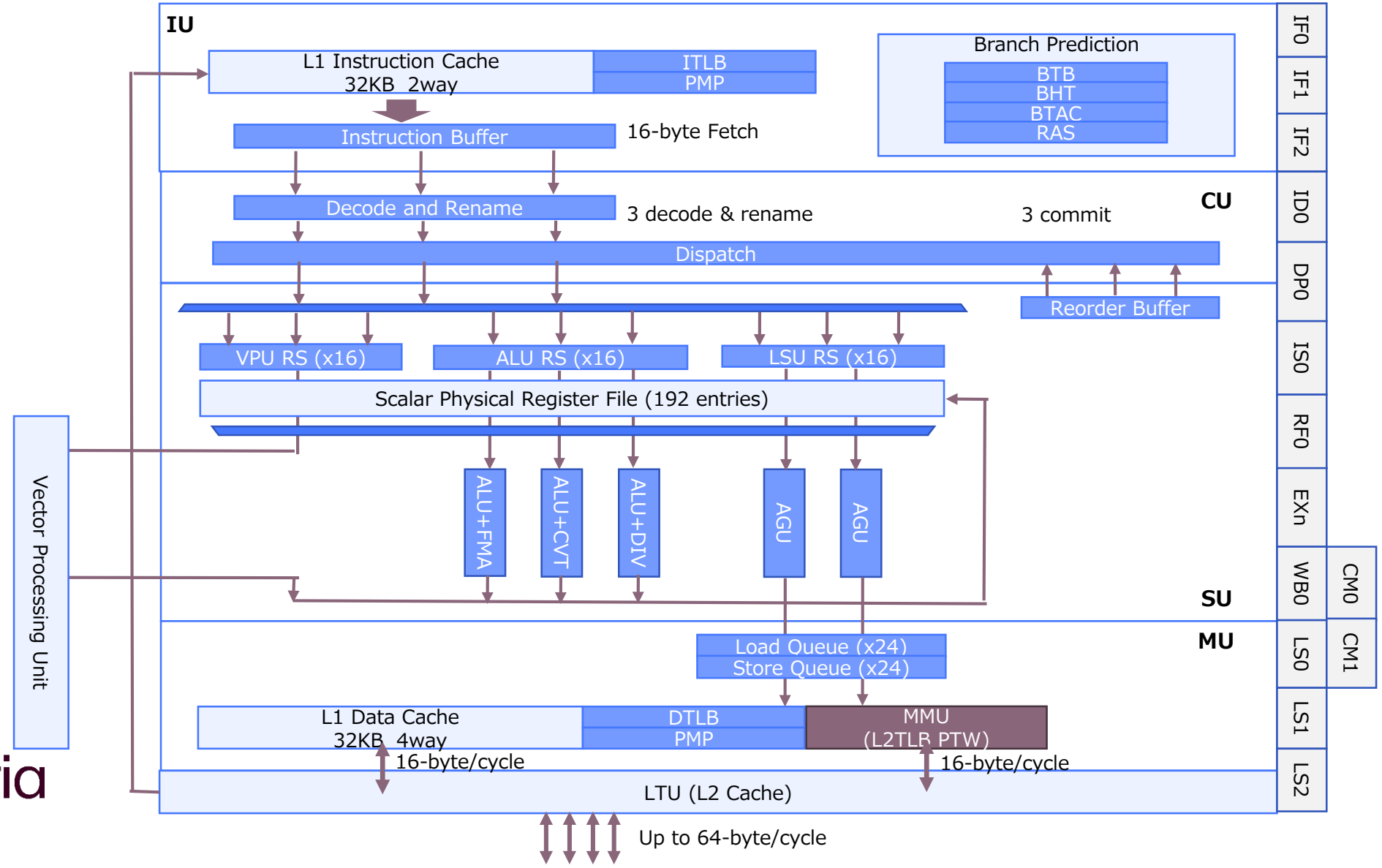


- Computing Core for Data Plane
  - RV64GCV+BtiManip (zba_zbb_zbs)
  - Standard Interrupt Peri. (ACLINT, PLIC)
  - Up to 4 Cores / Cluster
- Efficient 64-bit Scalar units
  - 10+ stages 3-way Out-of-Order
  - Hardware multithreading (up to 2 harts)
- Full-scale 512-bit Vector units
  - 2 arithmetic pipelines/unit
  - 8192 bits width vector register
  - 0.4 TOPS@Int8(@1.6GHz)
- L2 Cache
  - Coherent-enabled
  - 128KB～2 MB w/ECC

※*Cluster and core parameters are configurable*

Akaria

Pipeline Overview

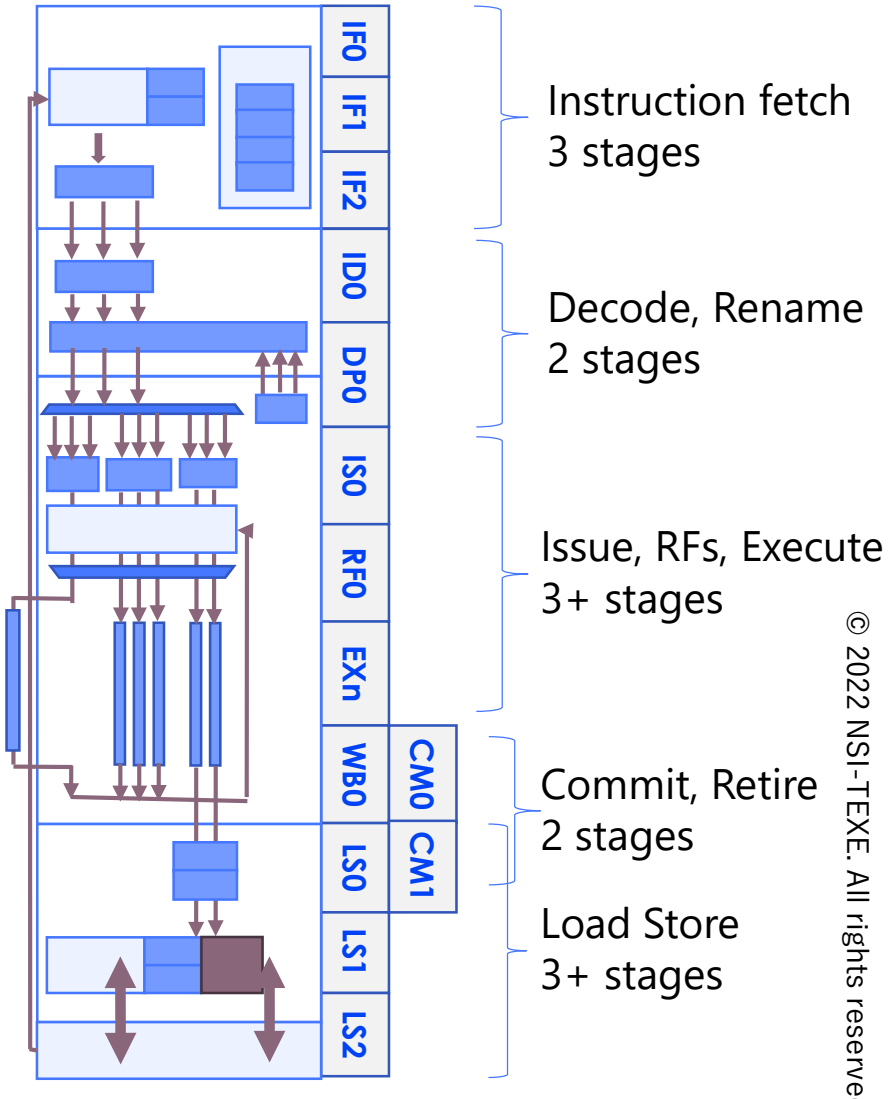


Akaria

Microarchitecture

- Microarchitecture Overview
  - ISA: RV64GCV+BitManip(zba_zbb_zbs)
  - 10+ stages 3-way Out-of-Order
  - Hardware Multithreading Support (up to 2 Harts)

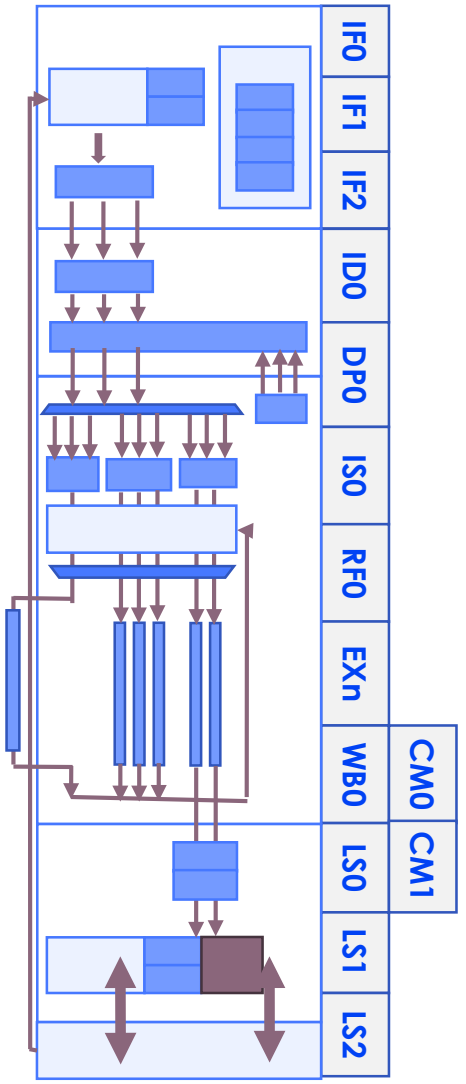
- Instruction fetch Unit (IU):
  - 32 kB L1 I-Cache (2way)
  - 16byte instruction fetch (4-8 instructions)
  - Branch Prediction :
    - BTB (Branch Target Buffer)
    - BHT (Branch History Table, gShare)
    - BTAC (Branch Target Address Cache)
    - RAS (Return Address Stack)



Akaria

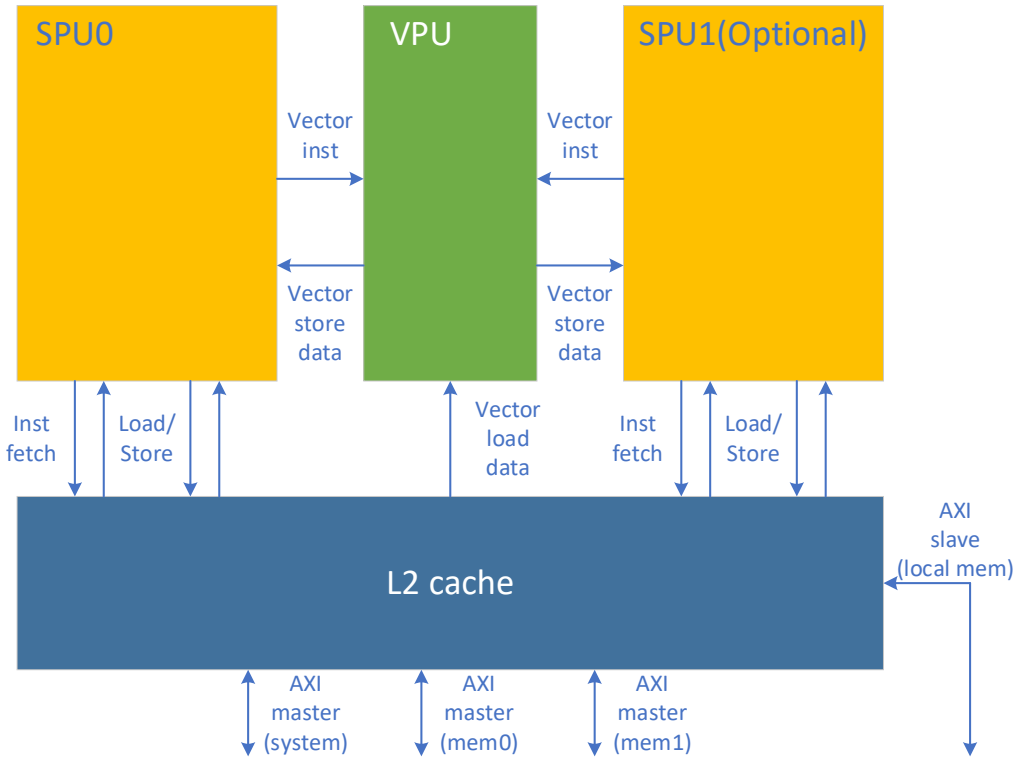
Microarchitecture

- Instruction Control Unit (CU):
  - x3 Decode & x3 Rename & x3 Dispatch
  - x3 Commit & Reorder Buffer 64 entries
  - CSR/Debug Support
- Scalar execution Unit (SU) :
  - Reservation Station x16 (ALU/LSU)
  - x3 ALU, x1 FMA, x1 MUL/DIV, x1 CVT
  - Scalar Physical Register File (192 entries@2harts)
- Memory Unit (MU) :
  - x2 AGU, x2 Load/Store issue
  - Load/Store Queue x24
  - 32 kB L1 Data Cache (4 way)



Akaria

Vector Processing Unit



- NSI's 2nd generation VPU architecture
- RISC-V Vector Extension v1.0 support
- SPU-VPU tightly coupled design
  - Minimize SPU-VPU synchronization wait
  - Speculative execution support
- Up to four threads support
  - Shared from two SPUs with two harts each
- Long vector register (Up to 8192 VLEN)
- Wide vector pipeline (512 bit x 4)
- High memory bandwidth (512bit / cycle)

Akaria

VPU Key Features

Long vector length design

- Long vector lengths can reduce the load on the instruction control section of SPUs and VPUs and improve execution efficiency

Stall-less pipeline design

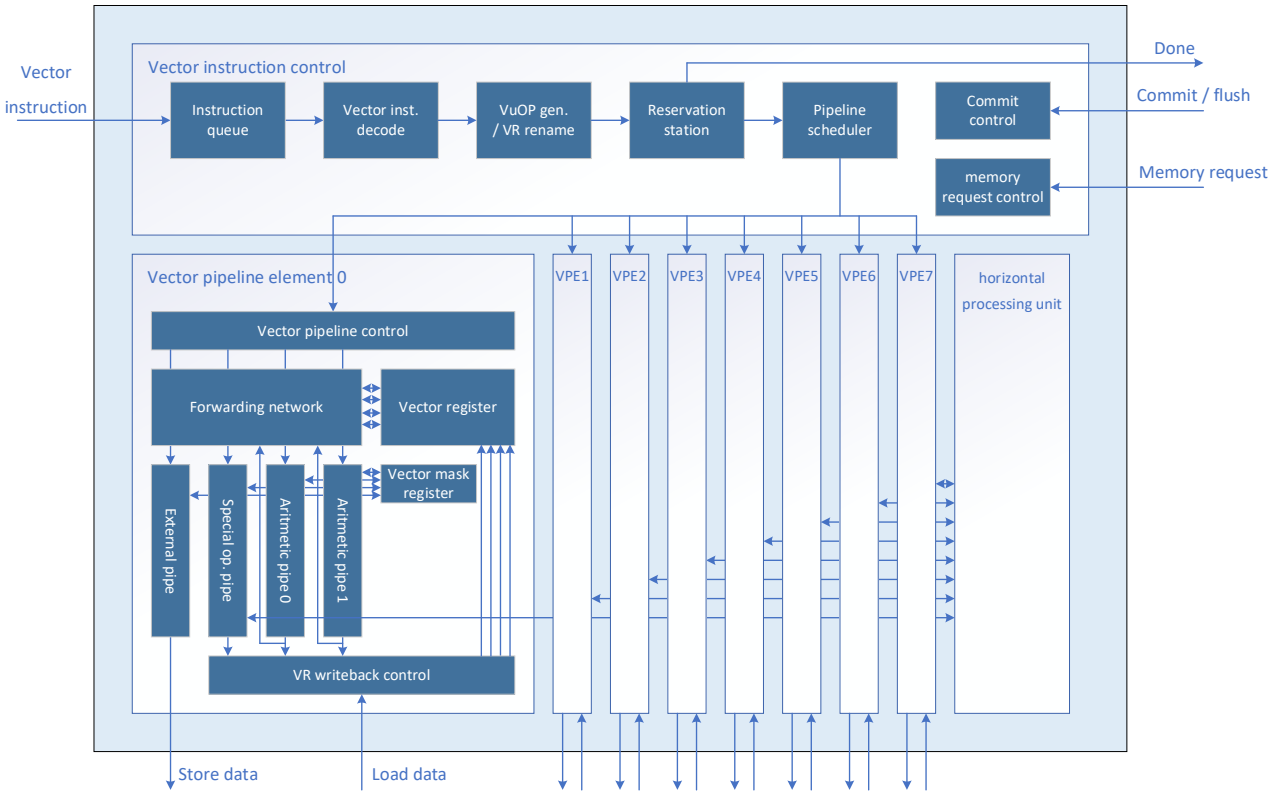
- Eliminate resources for pipeline stall tolerance by designing pipeline scheduler to prevent any conflicts.
- Deterministic pipeline behavior also enables vector chaining

Multi-threading support

- Hide SPU-VPU synchronization bottleneck by multi-thread execution

Akaria

VPU Structure Overview

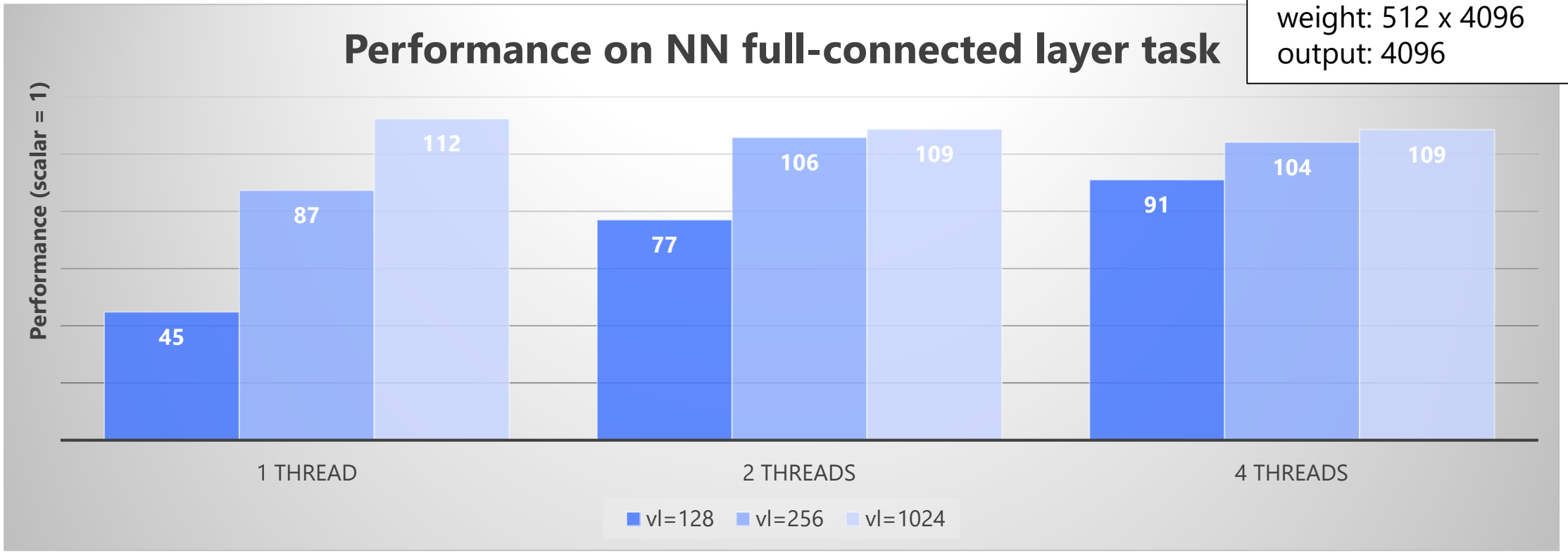


- 256KB physical vector register
- VR/VMR renaming
- Out-of-order execution
- Speculative execution
- Flexible VR access scheduling
- 8 pipeline elements and horizontal processing unit
- Four 512-bit wide execution pipes
  - Arithmetic pipe x 2
  - Special operation pipe x 1
  - External pipe x 1
- Vector chaining support

Akaria

VPU Performance

Data type: int8
Data size:
input: 512
weight: 512 x 4096
output: 4096



* Measurements are under development. The scalar code is highly optimized (unrolled loop). L2 cache status assumes a cold start.

- Up to x112 speed up from original scalar code
- Multi-threading boosts performance of short vector length tasks

Akaria

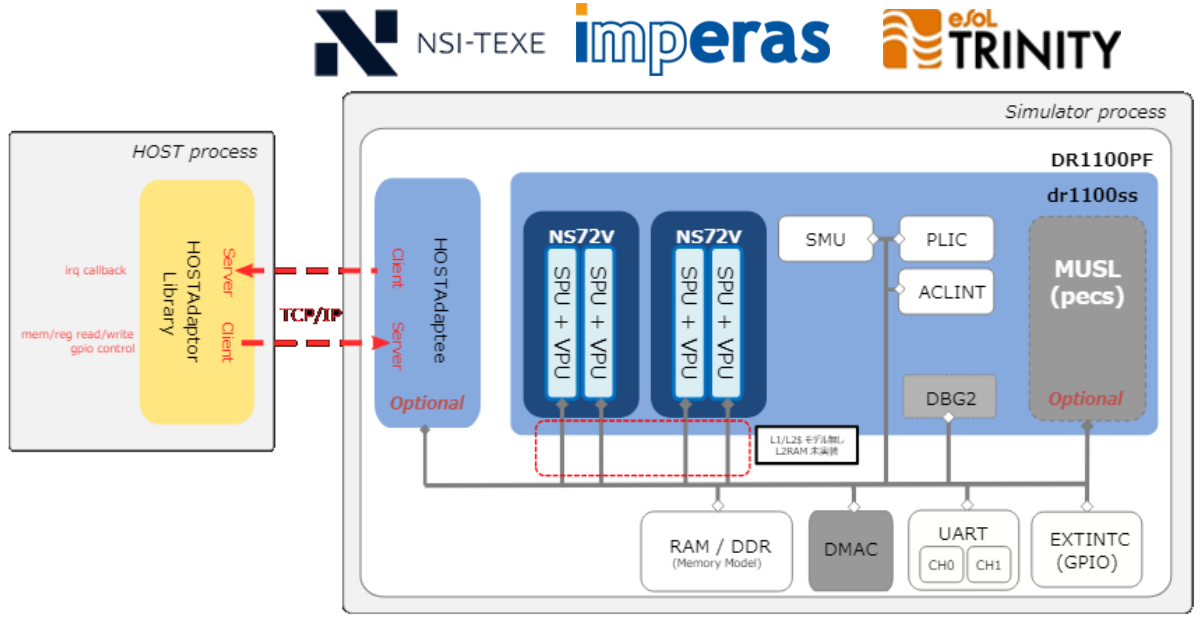
Akaria

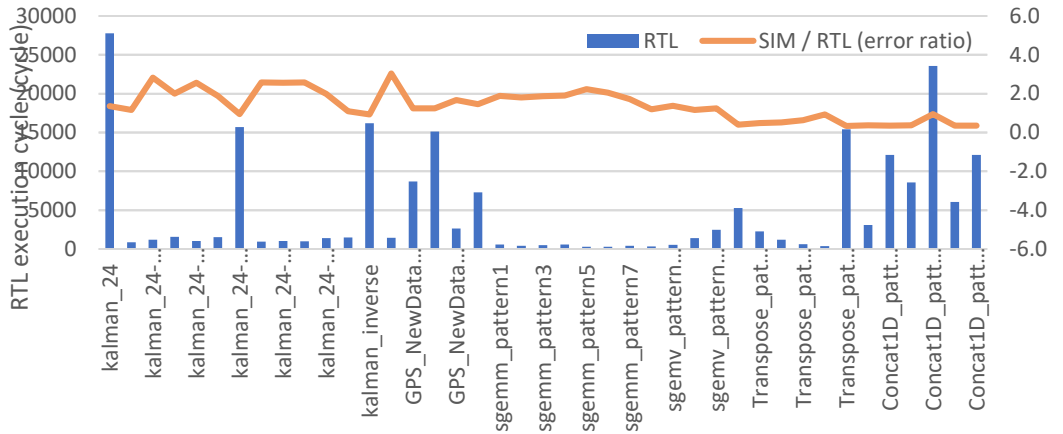
NS72 Demo

- Performance Estimator
- Host/Device Programming Env.

NS72 Performance Estimator

- Estimate rough performance gains from vector units
- Based on Imperas cpuDEV timing estimation capabilities
- Simulating pipeline, bus throughput and latency in a simplified manner
- Users can quickly estimate execution performance with reasonable accuracy without FPGAs, emulators, etc.



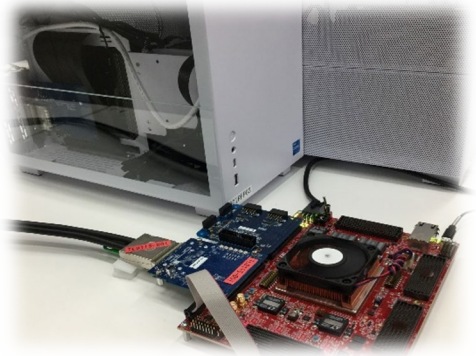


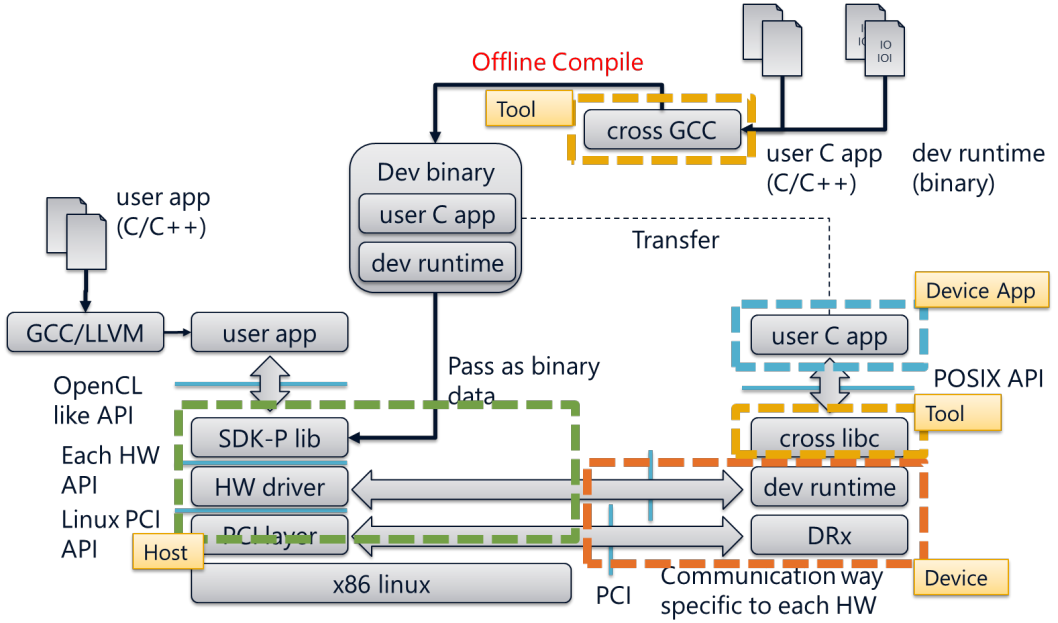
Akaria

Akaria Host/Device Programming

- Provides an easy to experience workload offload model
- C and OpenCL interfaces are available

- FPGA-integrated PC kits are available
- Can be used transparently with System Simulator





Host side (x86_64/ARM/RISC-V)

```
clGetPlatformIDs(...);
clGetDeviceIDs(...);
clCreateContext(...);
clCreateCommandQueueWithProperties(...);

clBuildProgram(elf_binary_data);
clCreateKernel(...);

a = clCreateBuffer(CL_MEM_READ_ONLY);
b = clCreateBuffer(CL_MEM_READ_ONLY);
c = clCreateBuffer(CL_MEM_WRITE_ONLY);
n = 1000;
bytes = n * sizeof(float);

clSetKernelArg(0, &a);
clSetKernelArg(1, &b);
clSetKernelArg(2, &c);
clSetKernelArg(3, &n);

float *h_a = (float *)malloc(bytes);
float *h_b = (float *)malloc(bytes);
float *h_c = (float *)malloc(bytes);

clEnqueueWriteBuffer(a);
clEnqueueWriteBuffer(b);

clEnqueueNDRangeKernel(...);
clFinish(...);

clEnqueueReadBuffer(c);
```

a, b for input
c for output

4 arguments
(argc = 4 + 1)

Device side (NS72V)

```
argv[0]: string of kernel name
argv[1]: buffer a
argv[2]: buffer b
argv[3]: buffer c
argv[4]: variable n

int main(int argc, char *argv[])
{
  const float *a = (const float *)argv[1];
  const float *b = (const float *)argv[2];
  float *c = (float *)argv[3];
  const int *n = (const int *)argv[4];

  printf("hello %s¥n", argv[0]);

  for (int i = 0; i < *n; i++) {
    c[i] = a[i] + b[i];
  }

  return 0;
}
```

Akaria

Thank you!

- Akaria NS72 as a compute core for data plane
- More efficient and mature 2nd gen. VPU for vector operation
- Domain Specific Accelerator based on NS72 will be released

- Akaria NS family will grow
  - Data plane : more performance and power efficiency
  - Control plane : more functionality

Akaria