

# RISC-V CON

ONLINE WEBINAR

## 解決大數據物聯網時代的 RISC-V Vector處理器

王勝雯

客製運算暨技術服務處 處長

晶心科技

2020/08/13

# Agenda

- Andes Overview
- RISC-V & RVV (RISC-V Vector Extension) Overview
- Andes VPU (Vector Processing Unit) Microarchitecture
- Andes NX27V Vector Processor
- Development Tools
- Summary

# Andes Corporate Overview



## Silicon Valley Tie

- Core R&D from AMD, DEC, Intel, MIPS, nVidia, and Sun

## 15-Year CPU IP Company

- IPO in 2017; HQ in Taiwan
- AndeStar™ V1-V3, V5 (RISC-V)

## 1+ Bn Annual Run Rate of Andes-Embedded SoC

- 5+ Bn total shipment
- ~300 customers Worldwide

## RISC-V International Founding Premier Member and Major Contributor

- Chairing Task Groups
- Contributing to GNU, LLVM, uBoot, glibc, Linux, etc.



# V5 Adoptions: From MCU to Datacenters

## ■ Edge to Cloud:

- ADAS
  - AIoT
  - Blockchain
  - FPGA
  - MCU
  - Multimedia
  - Security
  - Wireless (BT/WiFi)
- Datacenter AI accelerators
  - SSD: enterprise (& consumer)
  - 5G macro/small cells

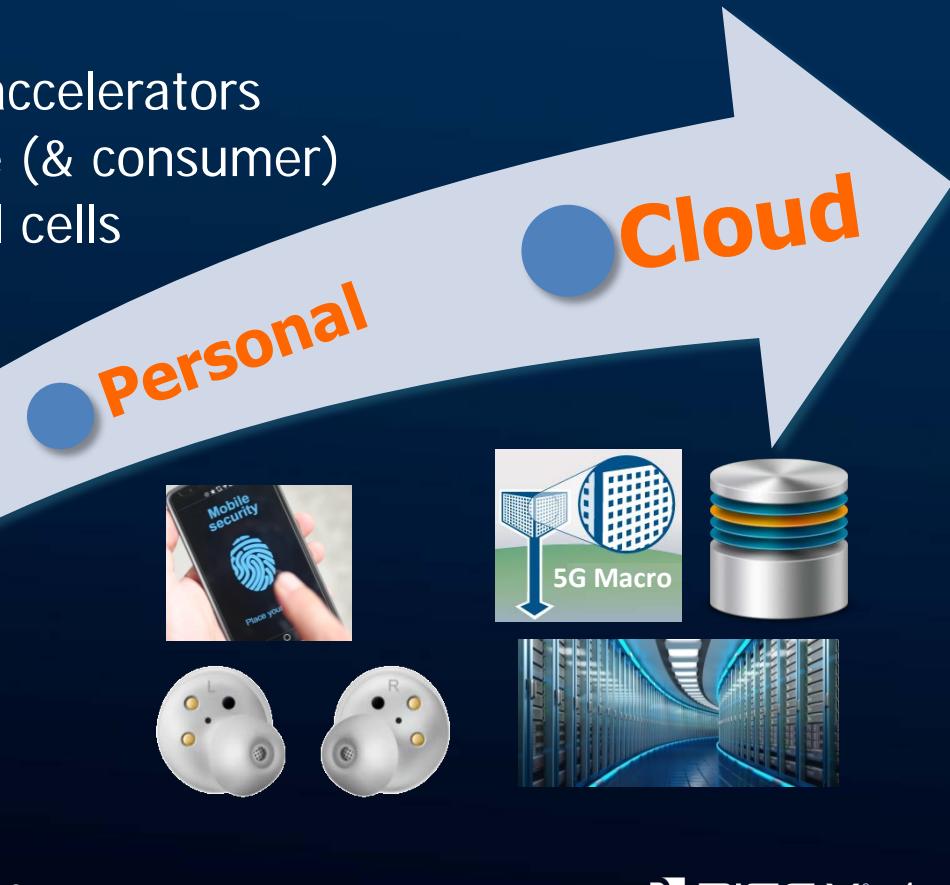
■ 1 to 1000+ core

■ 40nm to 7nm

■ Many in AI



Taking RISC-V® Mainstream





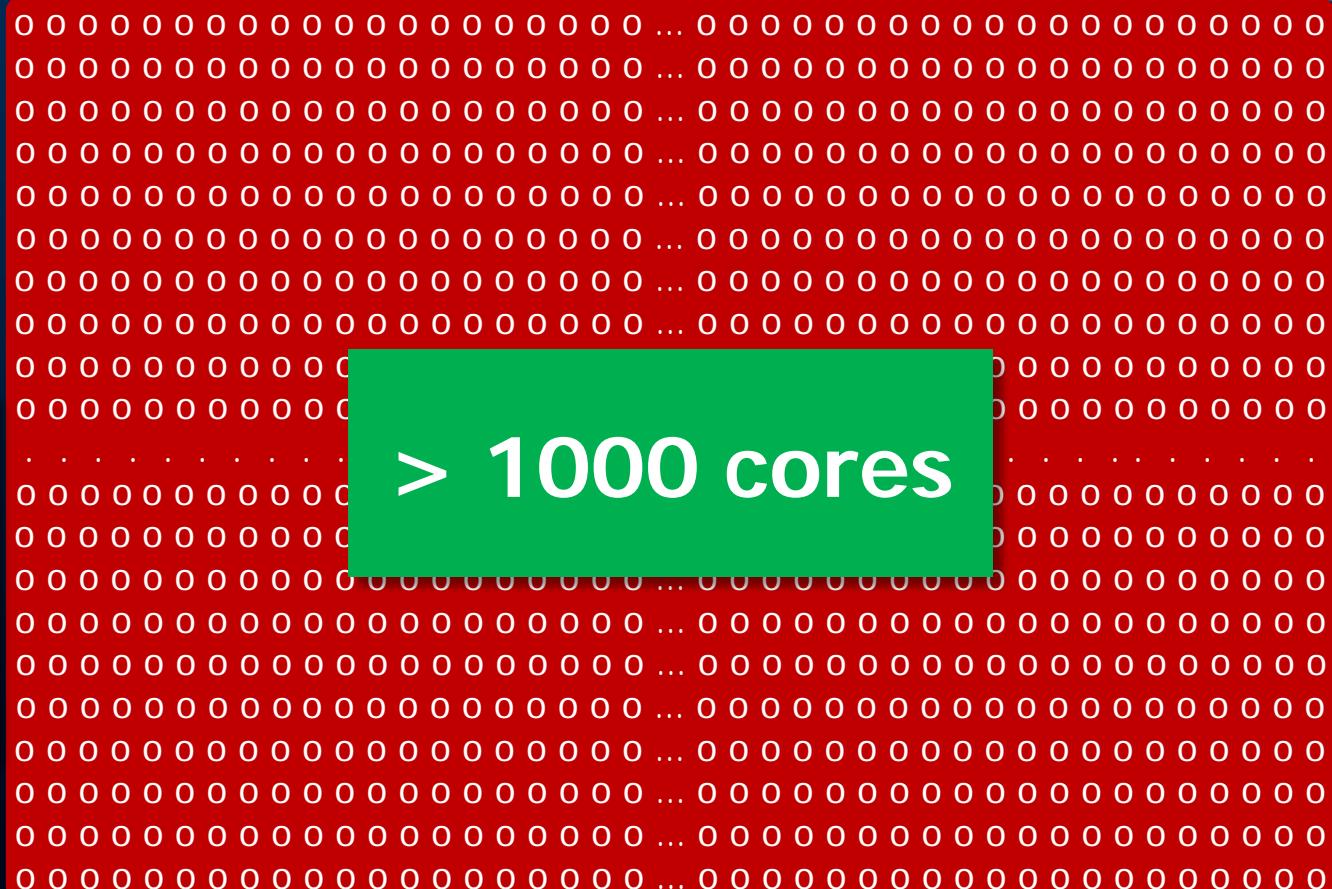
# Andes RISC-V V5 in SoC

0 single core

0000 2-8 cores  
0000

00000000  
0 > 30 cores 0  
00000000

0000000000000000  
0000000000000000  
00 > 100 cores 00  
00  
0000000000000000  
0000000000000000  
0000000000000000



Taking RISC-V® Mainstream



# RISC-V & RVV Overview

# RISC-V & RVV Background

■ An open processor architecture started by UC Berkeley

- Compact, modular, extensible

■ **RISC-V International** (formerly RISC-V Foundation):

- Formed in 2015 to govern its growth
- 500+ members including industry and research institute/university
- 2019 Dec. RISC-V Summit has over 1000+ attendees

■ **RISC-V Vector Extension**

- Defined in a RISC-V International Task Group
- Vector instruction set with scalable vector registers
- 2x and 4x data expansion arithmetic
- Over 300+ vector instructions, covering load/store, integer, fixed-point/floating-point operations



# ANDES RISC-V Vector Processing vs. SIMD

## ■ Assume the same computation width in hardware

- SIMD\_add: do 4 "8+8" in 1 cycle
- SIMD\_mul: do 4 "8\*8" in 1 cycle
- VEC\_add( $N$ ): do  $N$  SIMD\_add, 1 per cycle
- VEC\_mul( $N$ ): do  $N$  SIMD\_mul, 1 per cycle

## ■ Advantages of Vector Processing

- Save instruction issue bandwidth by issuing multiple SIMD operations at once
- Start subsequent independent (scalar or vector) instructions sooner

→ Con: setup overhead (vector length/type), HW cost (vector registers, control logic, Q's), sophisticated to program

## ■ When to use Vector Processing

- Applications with large arrays of data

Cycles:	1	2	3	4	5	6	7	8	9	10
SIMD_add 1	F	D	E	M	...	...	...	...	...	...
SIMD_add 2		F	D	E	M	...	...	...	...	...
SIMD_add 3			F	D	E	M	...	...	...	...
SIMD_add 4				F	D	E	M	...	...	...
SIMD_mul 1					F	D	E	M	...	...
SIMD_mul 2						F	D	E	M	...
SIMD_mul 3							F	D	E	M
SIMD_mul 4								F	D	E
I1									F	D
I2										F D

Cycles:	1	2	3	4	5	6	7	8	9	10
VEC_add(4)	F	D	E	M	...	...	...	...	...	...
VEC_mul(4)		F	D	E	M	...	...	...	...	...
I1			F	D	E	M	W			
I2				F	D	E	M	W		

# RVV: Scalable Vector Registers

## ◆ 32 Vector Registers (VR)

- ◆ Each with VLEN bits, depending on HW config.

## ◆ Data formats:

- ◆ SEW (standard element width): 8 to 1024
- ◆ Such as int8, int16, int32, int64, fp16, fp32, fp64.

## ◆ For **VLEN=512**, each VR has

- ◆ 16 elements when SEW=32 (int32/fp32)
- ◆ 32 elements when SEW=16 (int16/fp16)
- ◆ 64 elements when SEW=8 (int8)

## ◆ LMUL (Length Multiplier): VR combining

- ◆ Can be set to 1, 2, 4, or 8 at runtime by SW

## ◆ Example: For VLEN=512 and LMUL=8

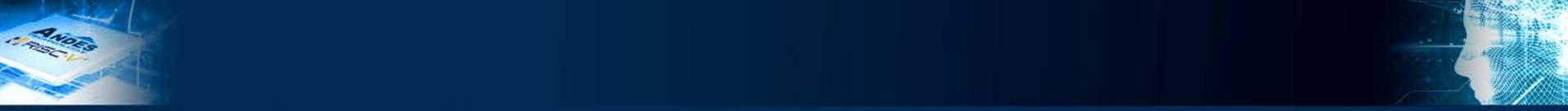
- ◆ v0 represents v0~v7, or effectively a 4096-bit register with 128 fp32 data.

V0	V0	V1	V0	V1	V2	V3
V1						
V2	V2	V3				
V3						
V4	V4	V5	V4	V5	V6	V7
V5						
V6	V6	V7				
V7						
...						
V24	V24	V25	V24	V25	V26	V27
V25						
V26	V26	V27				
V27						
V28	V28	V29	V28	V29	V30	V31
V29						
V30	V30	V31				
V31						

LMUL = 1      2      4

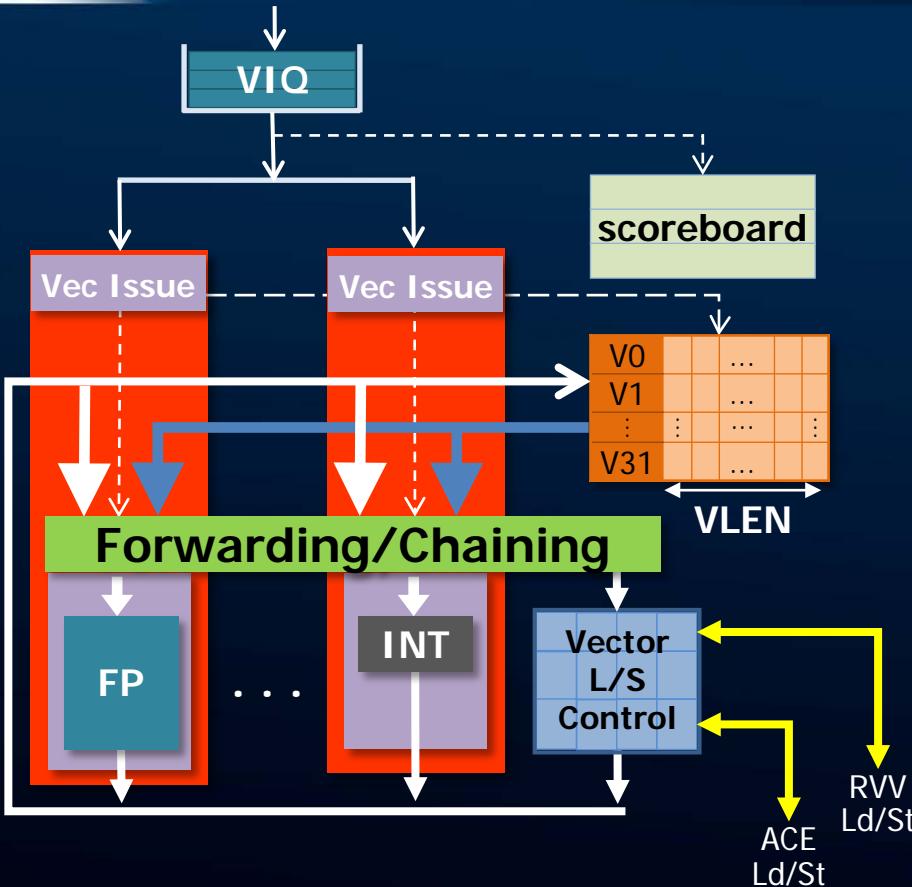
# Applications of Vector Processors

- Computation Kernels (Matrix Multiply, FFT, Sort)
- Cryptography (RSA, DES/IDEA, SHA/MD5)
- Machine/Deep Learning, AR/VR
- Multimedia Processing (Graphics, Image, Video)
- Networking (memcpy, memset)
- Scientific Computing (Modeling and Analysis)



# Andes VPU Microarchitecture

# Andes VPU Microarchitecture



- Supporting the latest RVV spec
- Data formats:
  - Standard: int8~int64, fp16~fp64
  - Andes-extended: bfloat16 and int4
- VLEN & SIMD width: 128, 256, 512
- Vector compute instructions:
  - Start execution after retired
  - Chainable, and most fully pipelined
  - Multiple Functional Units operating independently (OOO)
- Independent memory access paths thru RVV load/store and ACE load/store

# A Chaining Example with LMUL=8

LMUL	Cycles →	1	2	3	4	5	6	7	8	9	10
1	<b>vfcvt.f.x.v</b> v8, v16	X1	<u>X2</u>								
1	<b>vfadd.vv</b> v24, v8, v0			X1	X2	X3					

- LMUL=1 → 512 bits
- LMUL=8 → 4096 bits
- V0~V7 group 1
- V8~V15 group 2
- V16~V23 group 3
- V24~V31 group 4

Xn: execution stage

LMUL →

Chaining →

# A Chaining Example with LMUL=8

LMUL	Cycles →	1	2	3	4	5	6	7	8	9	10
1 2	vfcvt.f.x.v v8, v16 v9, v17	X1	X2								
1 2	vfadd.vv v24, v8, v0 v25, v9, v1		X1 X2	X1 X2	X1 X2 X3	X1 X2 X3					

- LMUL=1 → 512 bits
- LMUL=8 → 4096 bits
- V0~V7 group 1
- V8~V15 group 2
- V16~V23 group 3
- V24~V31 group 4

Xn: execution stage

LMUL →

Chaining →

# ANDES RISC-V

# A Chaining Example with LMUL=8

LMUL		Cycles →		1	2	3	4	5	6	7	8	9	10
1	2	vfcvt.f.x.v	v8, v16	X1	<u>X2</u>								
	8		v9, v17	X1	<u>X2</u>								
			...			...	...	...	...	...	...	...	
			v15, v23									X1	<u>X2</u>
1	2	vfadd.vv	v24, v8, v0			X1	<u>X2</u>	X3					
	8		v25, v9, v1			X1	<u>X2</u>	X3					
			...			...	...	...	...	...	...	...	
			v31, v15, v7										X1
				Xn: execution stage				LMUL		Chaining			

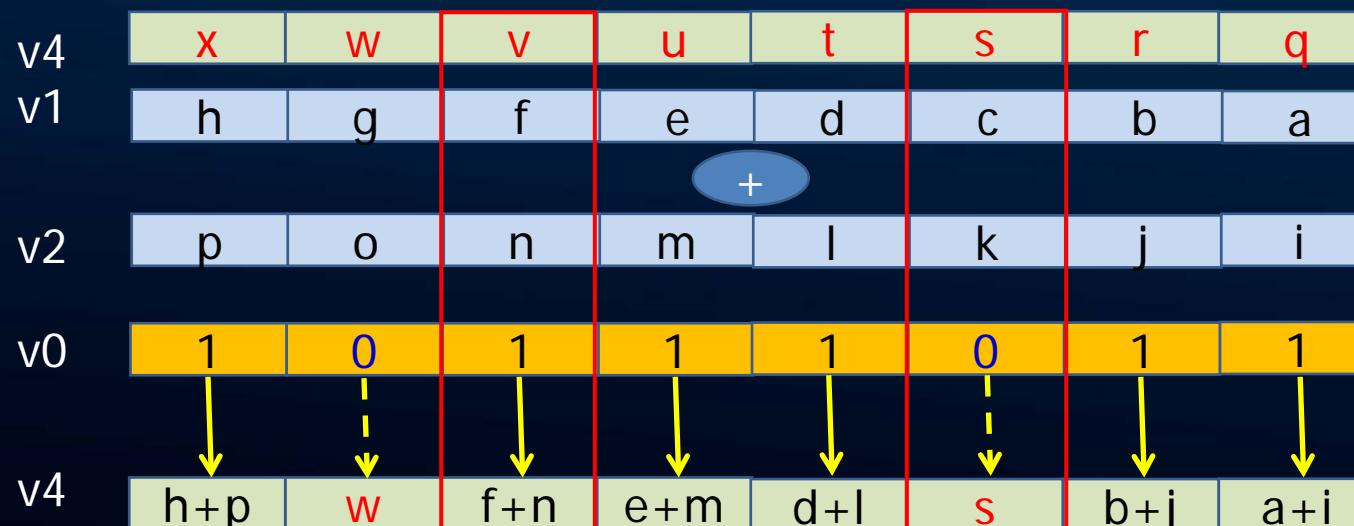
- LMUL=1 → 512 bits
- LMUL=8 → 4096 bits
- V0~V7 group 1
- V8~V15 group 2
- V16~V23 group 3
- V24~V31 group 4

# Vector Mask Example

- Only the least-significant bit of each element of the mask vector v0 is used to control execution

vadd v4, v2, v1, v0.t

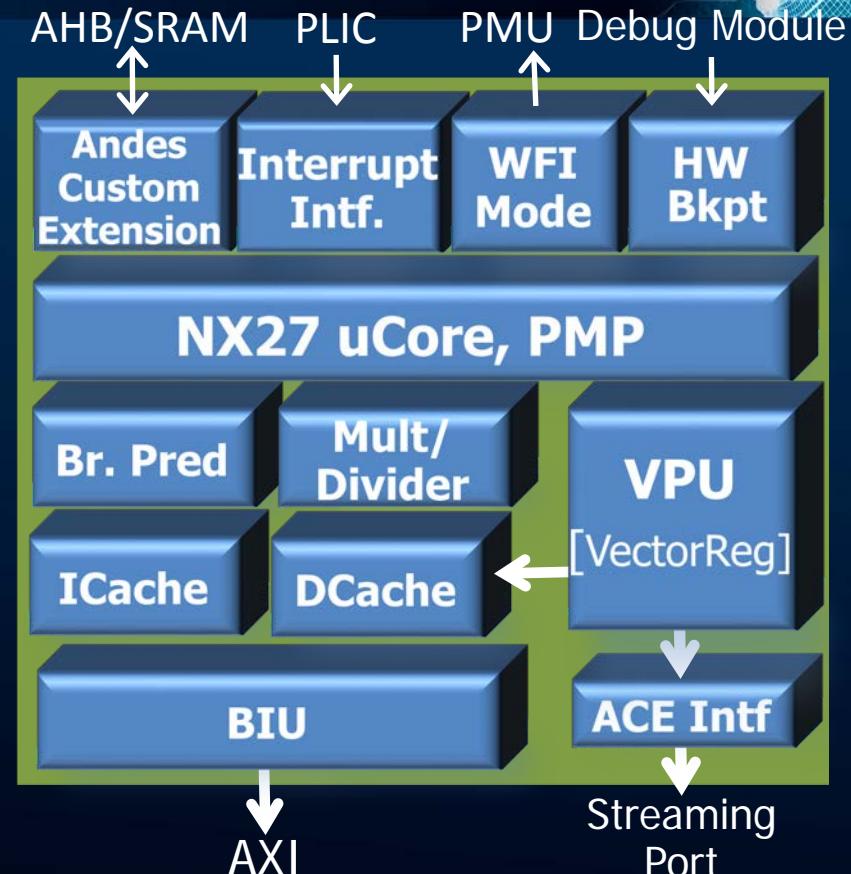
v0.t : enable the update if v0[i].LSB=1



# Andes NX27V Vector Processor

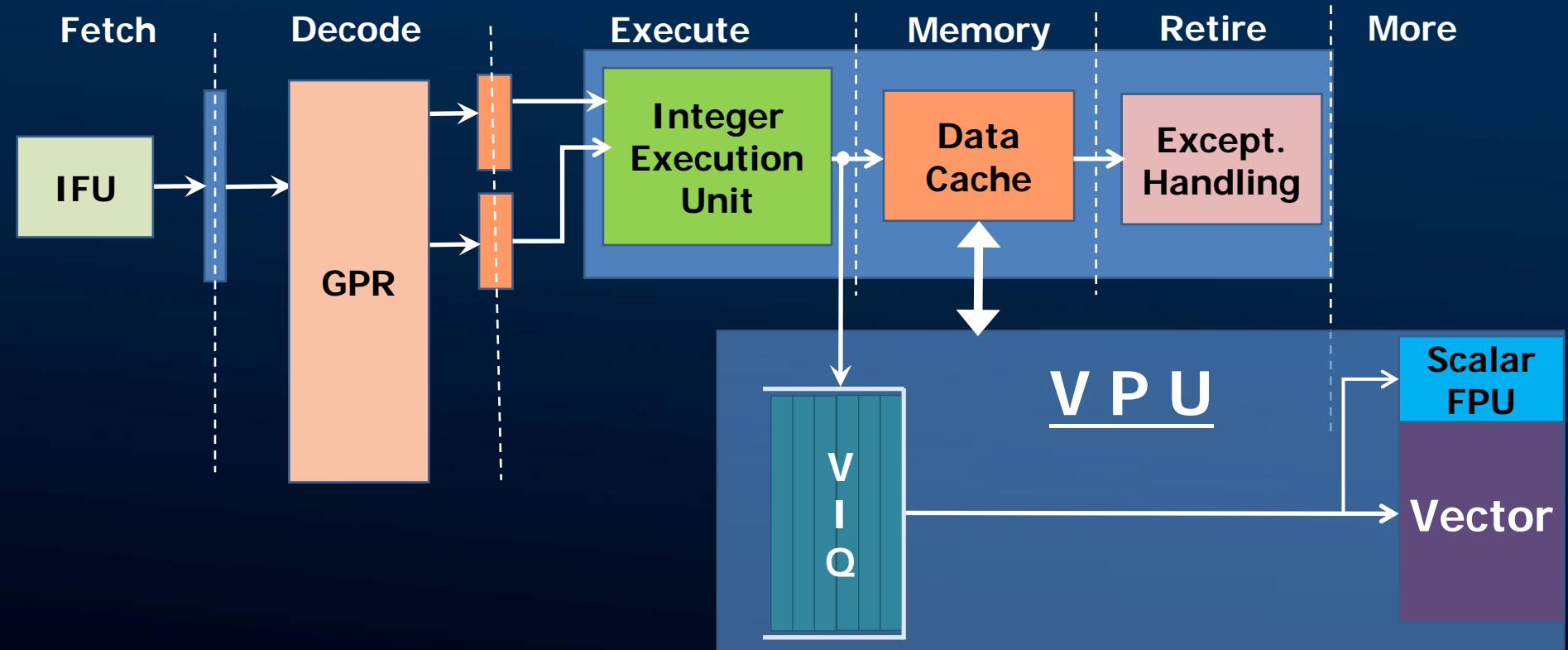
# Overview of NX27V

- AndeStar V5 architecture:
  - RV64GCNP + Andes V5 Extensions
  - RVV extension
- 5-stage pipeline, single-issue, in-order
- Optional branch prediction
- I/D caches
  - Caches: 8KB to 64KB
  - HW unaligned load/store accesses
  - Optional parity or ECC protection
- I\$/D\$ prefetch
- 16 outstanding data accesses
- Wide data paths to feed VPU:
  - Cached and uncached RVV load/stores
  - Streaming Port for ACE loads/stores





# NX27V Pipeline

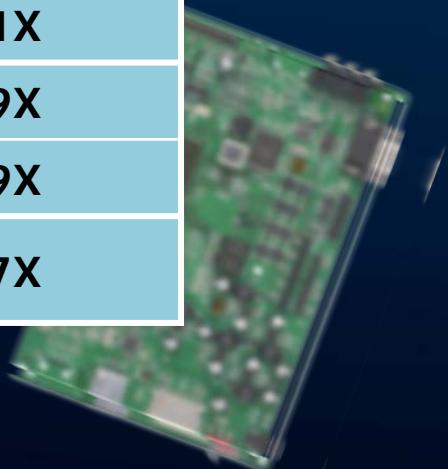




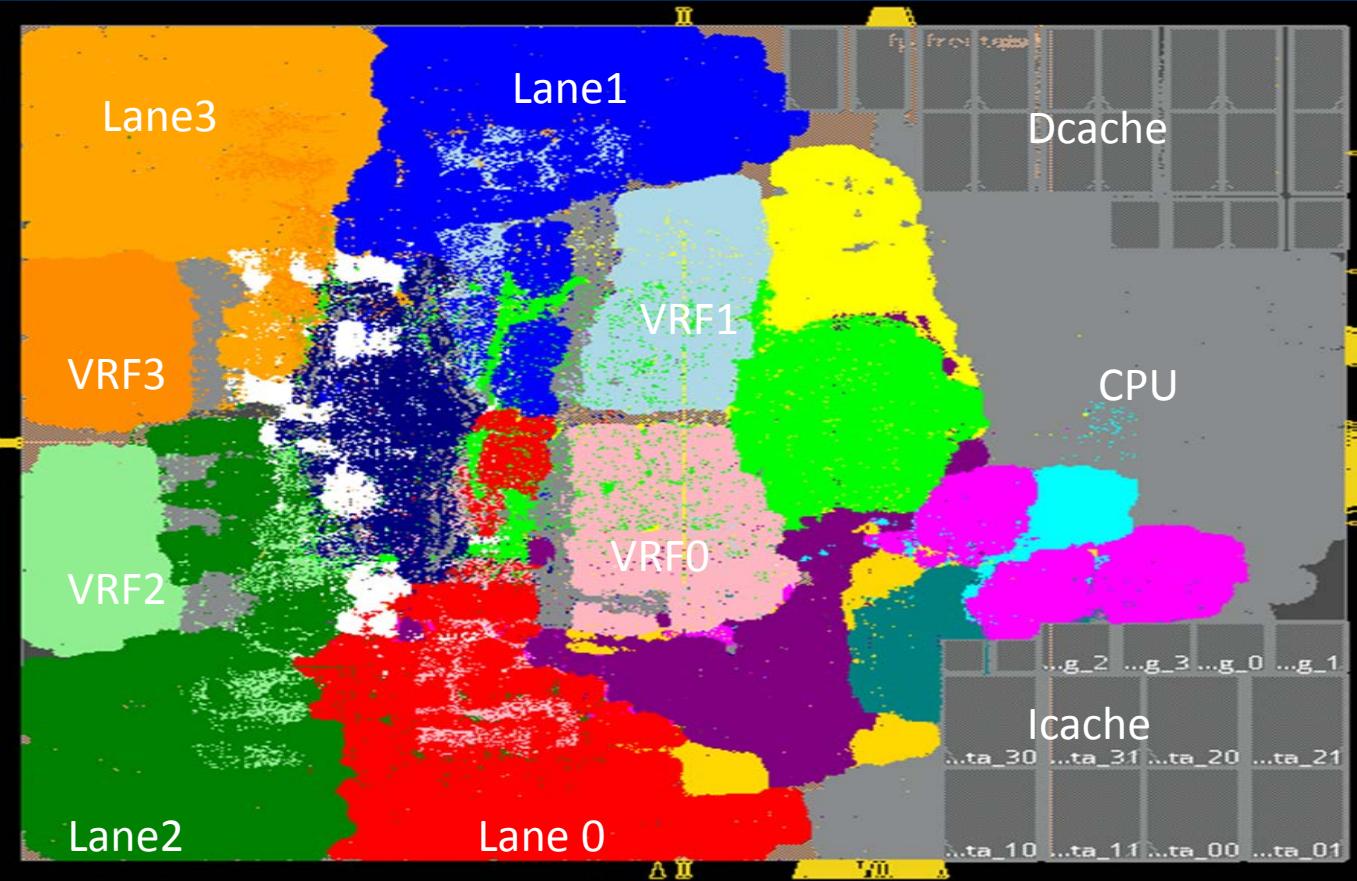
# NX27V Speedups

Functions	Speedup <sup>1</sup>
F32 basic mathematical functions	19X
RGB CNN functions	18X
Depthwise CNN functions	18X
Pointwise CNN functions	21X
F32 filtering functions	19X
Q7 filtering functions	39X
F32 32x32x32 matrix multiplication	57X

Note 1: Compared to pure C scalar code compiled with high optimization;  
both vector and scalar code ran on the NX27V FPGA with 512-bit VLEN, 256-bit bus.



# NX27V Floorplan



- ◆ 7nm
- ◆ 1GHz  
(worst case)
- ◆ ~0.3mm<sup>2</sup>

Taking RISC-V® Mainstream

# Andes NX27V vs. Competitions

	Andes NX27V	CAxxx	CMxx
Architecture	RVV/Andes VPU	Popular SIMD	Hxxx
Vector register number	32	32	8
Vector Length	Up to 512b 	128b	128b
SIMD width	Up to 512b/cycle 	128b/cycle	64b/cycle
LMUL	Yes 	None	None
Chaining	Yes 	Not applicable	Yes

# Development Tools

# Development Tools

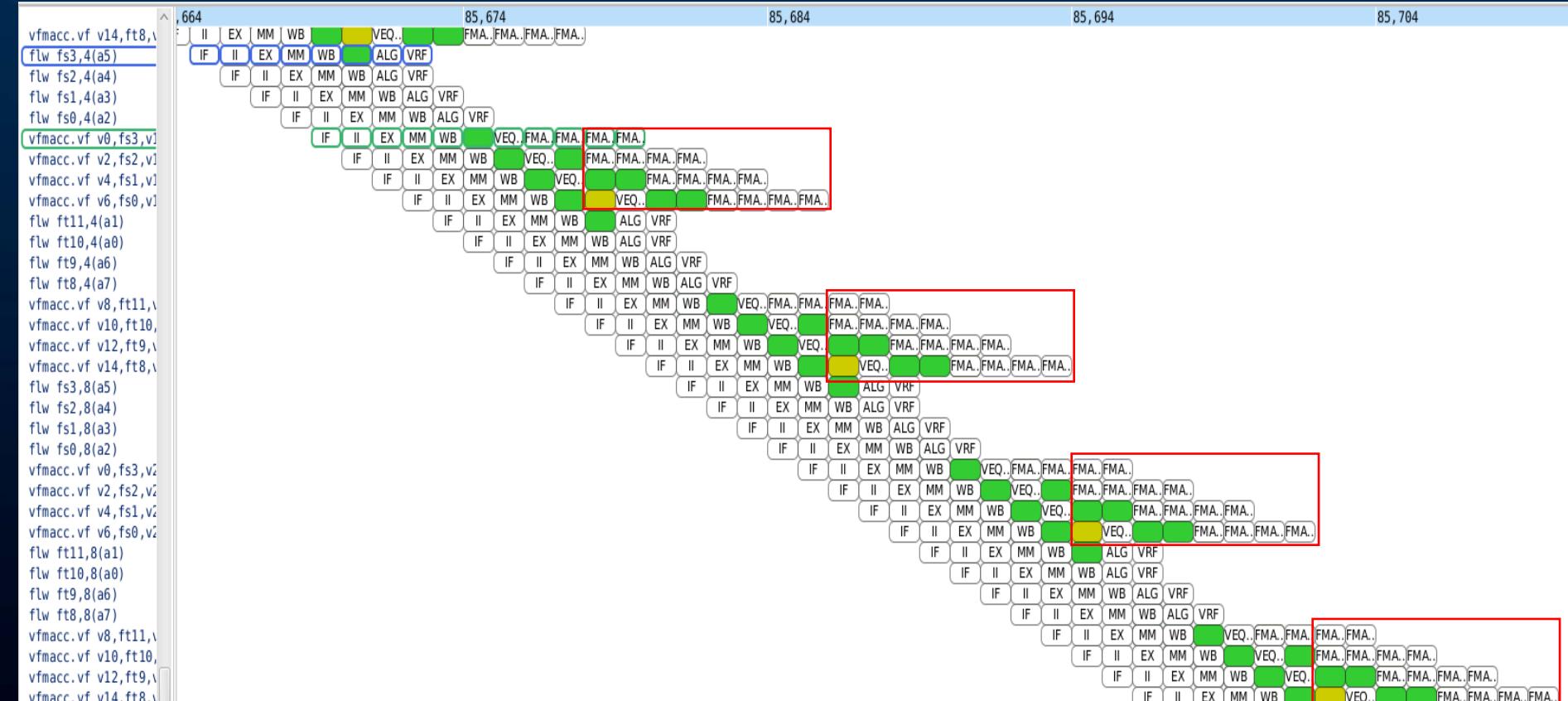
## ■ Standard tools in AndeSight™ IDE:

- AndeSim: Near-cycle accurate simulator
- Compiler (intrinsic functions and inline assembly)
- Assembler
- Debugger and ICE
- Computation library

## ■ Advanced tools:

- AI compiler support thru LLVM
- AndesClarity pipeline visualizer and analyzer
  - ◆ Pipeline view of instructions and functional units
  - ◆ Resource view corresponding to instruction usage
  - ◆ Stall bubbles with different colors for data dependency

# Clarity: Pipeline View



# Summary

# Summary

- RVV opens the doors for RISC-V to further exceed other ISA's
- Andes NX27V vector processor
  - The **world first** commercial RISC-V vector processor
  - Setting the standard for high performance RISC-V with innovative design
  - Flexible VPU configurations to enable a wide range of applications
  - AndesClarity for performance optimization
- Andes Technology will continue to expand vector product families based on Andes VPU microarchitecture

# HEART Design Created by Andes CCBU



## Application-Focused

Andes allows partners to focus on application development and assists with secure and effective IP development

## Experts of CPU

Andes as an experienced CPU expert delivers cutting-edge and innovative CPU implementation



## Human Resource Saving

Andes helps reduce long-term manpower and expenditures with project team support

E

A

R

H

T

Create Your Own  
Next-generation  
IP



## Reliable Dedicated Technical Services

Our technical service team is organized to provide our customers in-depth knowledge, technical support and consulting service with web-based online interface.

## Time to Market

Andes delivers performance-efficient IP with full-featured integrated development environment and solutions to help customers innovate their product in a shorter time frame.

# Successful Stories



**AI/IoT/5G  
Implements**

**Vector  
Processor**

**Interface  
Customization**

**Advanced  
Tools**

**Custom  
AndeSight**

Implementation and  
Verification Service

AndesCore Vector  
and ACE Extension

AndesCore Interface  
Customization

Compiler and  
Debugging Support

High Performance  
and Cycle Accurate  
Analysis Tool



Thank you!