
# RISC-V Vector Extension Webinar I

July 13th, 2021

Thang Tran, Ph.D.
Principal Engineer

# Andes Technology Corporation

Pure-play
CPU IP Vendor

RISC-V Founding
Premier Member

Major Open-Source
Contributor/Maintainer

16-year-old
Public Company

RISC-V Ambassador
Running Task Groups
TSC Vice Chair
Director of the Board

**Quick Facts**

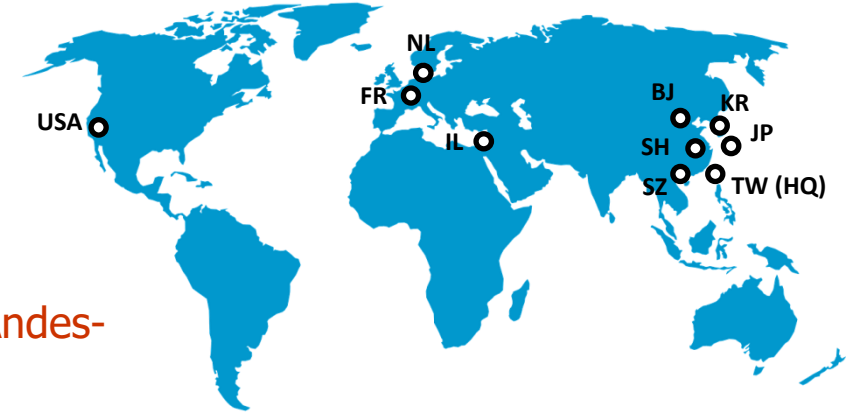
**100+ years**
CPU Experience in Silicon Valley

**80%**
R&D

**200+**
Licensees

**20K+**
AndeSight IDE installations

**7B+**
Total shipment of Andes-Embedded™ SoC



# Andes Technology Corporation Overview

## Andes Highlights

- Founded in March 2005 in Hsinchu Science Park, Taiwan, ROC.
- World class 32/64-bit RISC-V CPU IP public company
- Over 200 people; 80% are engineers; R&D team consisting of Silicon Valley veterans
- **TSMC OIP Award** "Partner of the Year" for New IP (2015)
- A **Premier founding member** of RISC-V Foundation
- 2018 MCU Innovation Award by China Electronic News: AndesCore™ N25F/NX25F
- ASPENCORE WEAA 2020 Outstanding Product Performance of the Year: **AndesCore™ NX27V**
- 2020 HsinChu Science Park Innovation Award: **AndesCore™ NX27V**

## Andes Mission

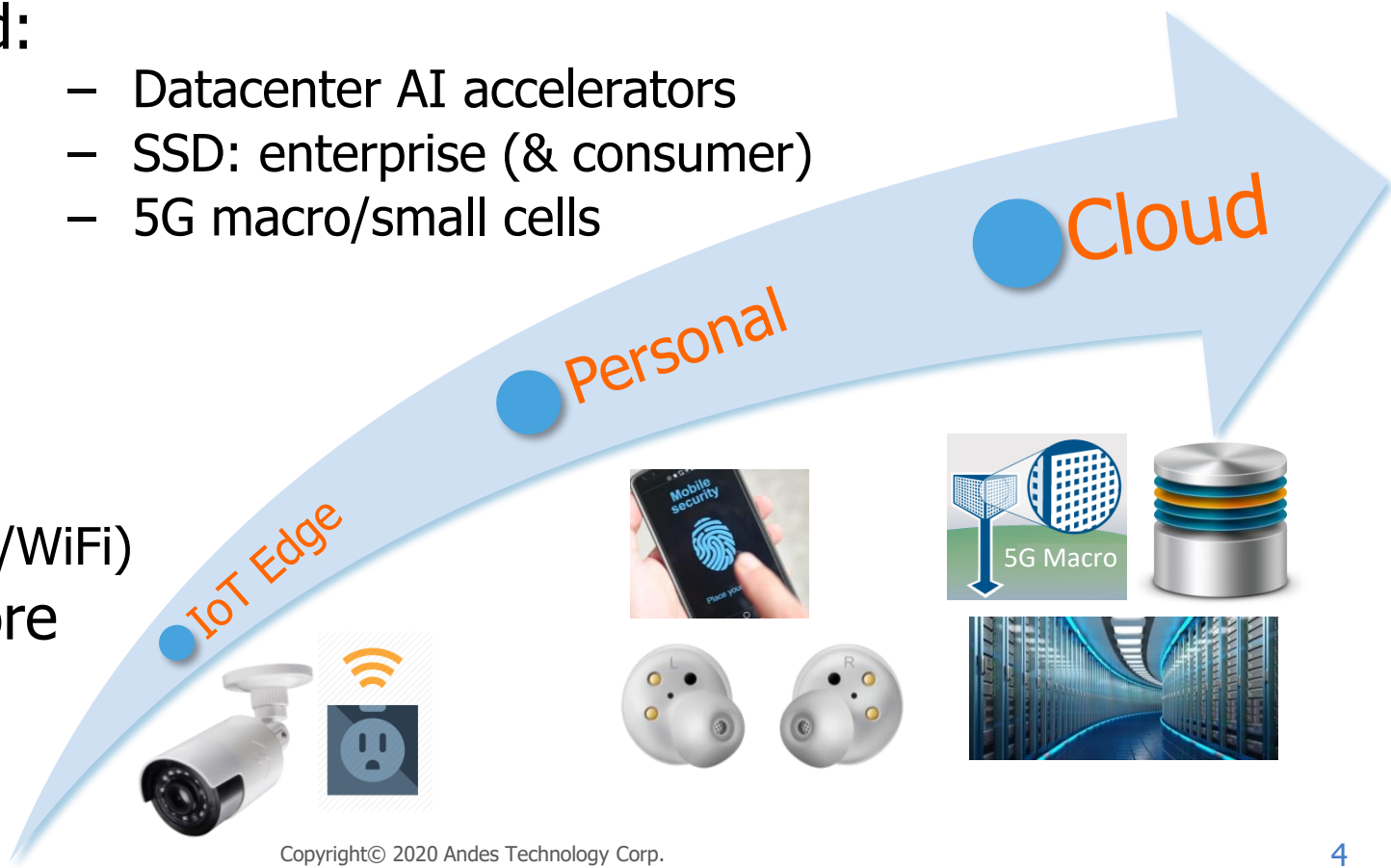
- **Trusted Computing Expert** and World No.1 **RISC-V** IP Provider

## Emerging Opportunities

- **AIoT, 5G/Networking, Storage and Cloud computing**

# 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 5nm
- Many in AI



# Webinar I - Agenda

- **Andes overview**
- **Vector technology background**
  - **SIMD/vector concept**
  - **Vector processor basic**
- RISC-V V extension ISA
  - Basic
  - CSR
  - Memory operations
  - Compute instructions
- Sample codes
  - Matrix multiplication
  - Loads with RVV versions 0.8 and 1.0
- AndesCore™ NX27V introduction
- Summary

# Terminology

- ISA: Instruction Set Architecture
- GOPS: Giga Operations Per Second
- GFLOPS: Giga Floating-Point OPS
- **XRF**: Integer register file
- FRF: Floating-point register file
- **VRF**: Vector register file
- SIMD: Single Instruction Multiple Data
- MMX: Multi Media Extension
- SSE: Streaming SIMD Extension
- AVX: Advanced Vector Extension
- ACE: Andes Custom Extension
- CSR: Control and Status Register
- **SEW**: Element Width (8-64)
- ELEN: Largest Element Width (32 or 64)
- **XLEN**: Scalar register length in bits (64)
- FLEN: FP register length in bits (16-64)
- **VLEN**: Vector register length in bits (128-512)
- **LMUL**: Register grouping multiple (1/8-8)
- EMUL: Effective LMUL
- **VLMAX**/MVL: Vector Length Max
- AVL/**VL**: Application Vector Length
- **Configurable**: parameters are fixed at built time, i.e. cache size
- **Extensible**: added instructions to ISA includes custom instructions to be added by customer
- **Standard extension**: the reserved codes in the ISA for special purposes, i.e. FP, DSP, …
- **Programmable**: parameters can be dynamically changed in the program

# RISC-V & RVV Background

- **An open processor architecture started by UC Berkeley**
  - Compact, modular, extensible
- **RISC-V International (formerly RISC-V Foundation)**
  - RISC-V Foundation formed in 2015 to govern its growth
  - 750+ members including industry and research institute/university
- **RISC-V Vector Extension**
  - Defined in a Foundation Task Group
  - Vector instruction set with scalable vector
  - Scalable data sizes with 2x data expansion arithmetic
  - Over 300+ vector instructions, including load/store, integer, fixed-point/floating-point operations

# RISC-V V Spec Status

- Version 1.0-rc1 was published in June 2021
- Note from the spec:

  *"When finally approved and the release candidate tag is removed, version 1.0 is intended to be sent out for public review as part of the RISC-V International ratification process. Version 1.0 is also considered stable enough to begin developing toolchains, functional simulators, and initial implementations, including in upstream software projects, and is not expected to have major functionality changes except if serious issues are discovered during ratification. Once ratified, the spec will be given version 2.0".*
- Version 1.0 is targeted for public ratification by July 2021

# Vector technology background

# Flynn's Classification

- Instruction Stream
  - Sequence of Instructions read from memory
- Data Stream
  - Operations performed on the data in the processor
- SIMD is the most interesting which is the basic for:
  - MMX, SSE, SSE2-5, AVX
  - Altivec
  - Neon

Microprocessors

DSP
Graphic
Vector

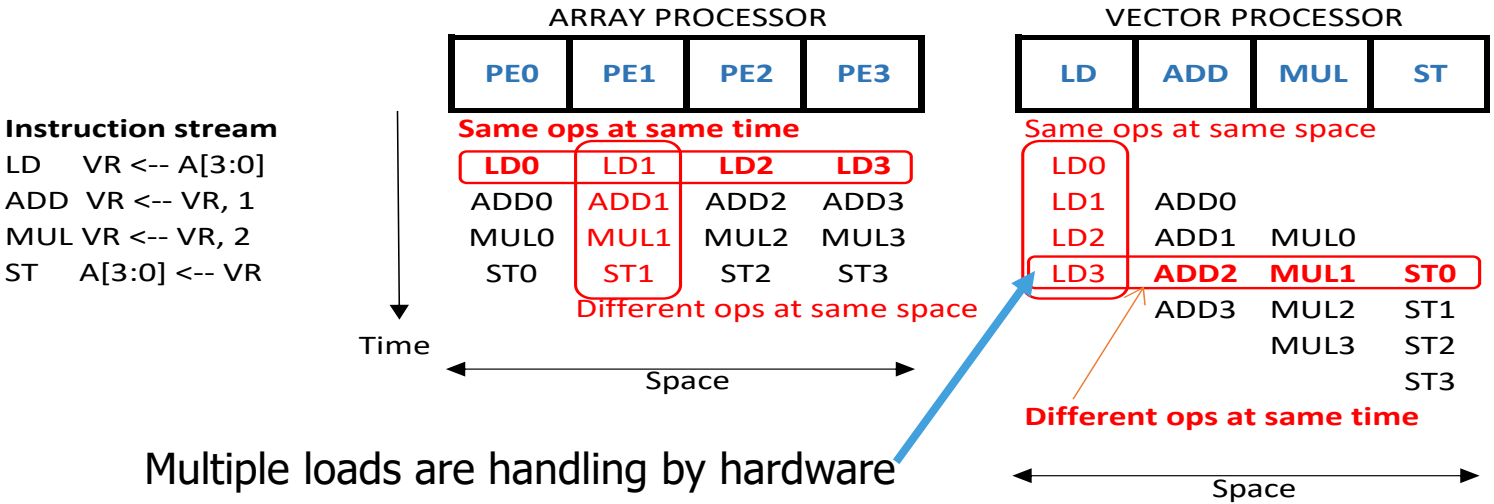
| | | Number of Data Streams | |
|---|---|---|---|
| | | Single | Multiple |
| Number of Instruction Streams | Single | SISD | SIMD |
| | Multiple | MISD | MIMD |

# Why SIMD?

- Single 32-bit Microprocessor
  - One 32-bit element per instruction
- 512-bit SIMD processor
  - **16** 32-bit elements per instruction
  - **32** 16-bit elements per instruction
  - **64** 8-bit elements per instruction. If the SIMD processor clock frequency is at 1 GHz, then the performance is 64 GOPS
- Increasing the SIMD size, the higher the performance
- SIMD is the basic for Vector Processor
  - N operations per instruction where the elements in the operations are independent from each other

# Parallel Processing

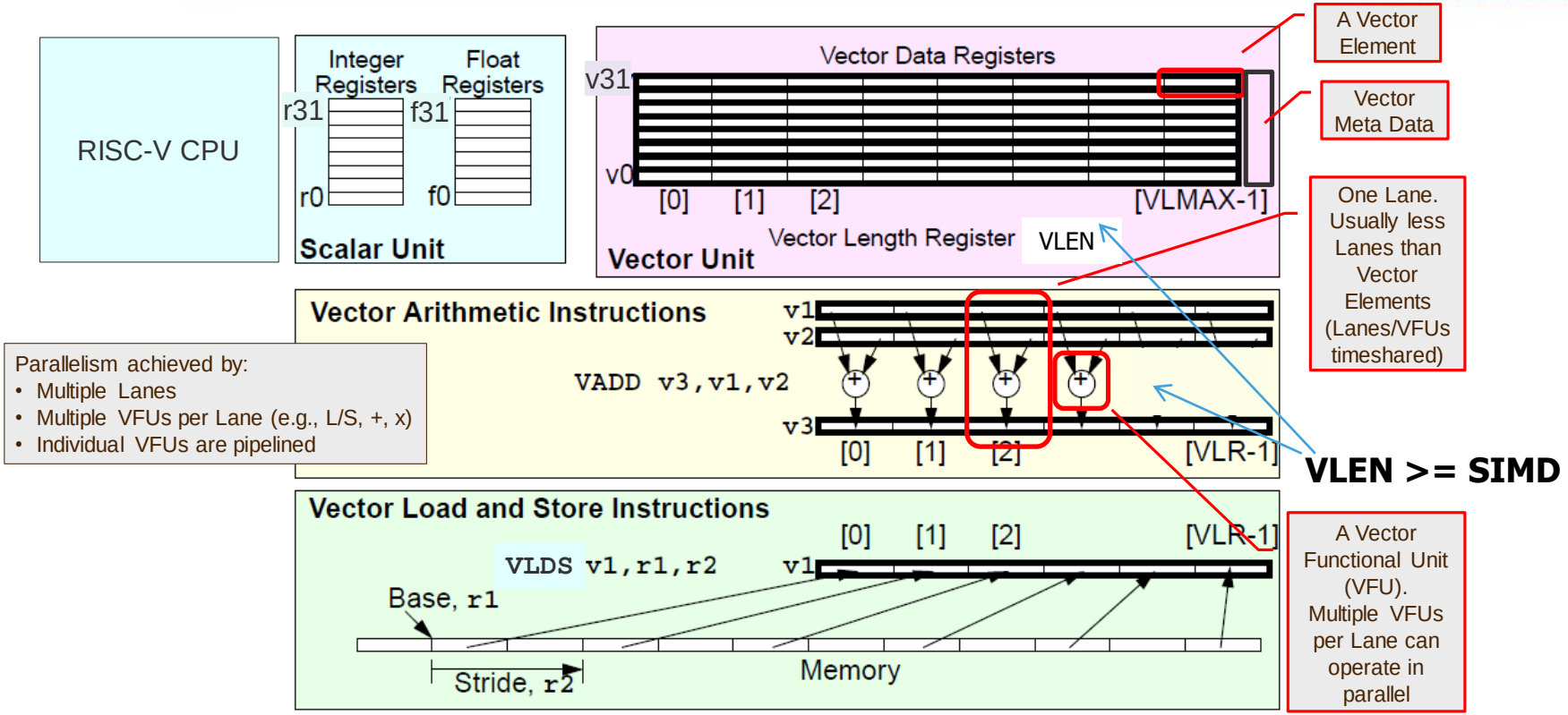
- Time-space duality:
  - Array processor: Instruction operates on multiple data elements at the same time – multi-thread, multi-core
  - Vector processor: Instruction operates on multiple data elements in consecutive time steps – pipeline of SIMD execution



# Intel MMX, SSE, SSE2, AVX

- **MMX** (1997) adds 57 new **integer** instructions, **64-bit** registers
  - 8 of 8-bit, 4 of 16-bit, 2 of 32-bit, or 1 of 64-bit (reuse 8FP registers – FP and MMX cannot mix)
  - Number of elements is defined in the opcode, only consecutive memory load/store is allowed
  - Move, Add, Subtract, Shift, Logical, Mult, MACC, Compare, Pack/Unpack
- **SSE** (1999) adds 70 new instructions for single precision **FP**, 2 of 32-bit
- **SSE2** (2001) adds 144 new instructions for FP, 4 of 32-bit, **128-bit** registers
- **SSE3** (2004), SSE4 (2006), SSE5 (2007) add more new instructions
- **AVX** (2008) adds new instructions for **256-bit** registers
- **AVX2** (2013) adds more new instructions
- **AVX-512** (2015) adds new instructions for **512-bit** registers
- Issue:
  - Each register length has a new set of instructions
  - Many instructions for the 512-bit SIMD

# Vector Extension Processor Overview

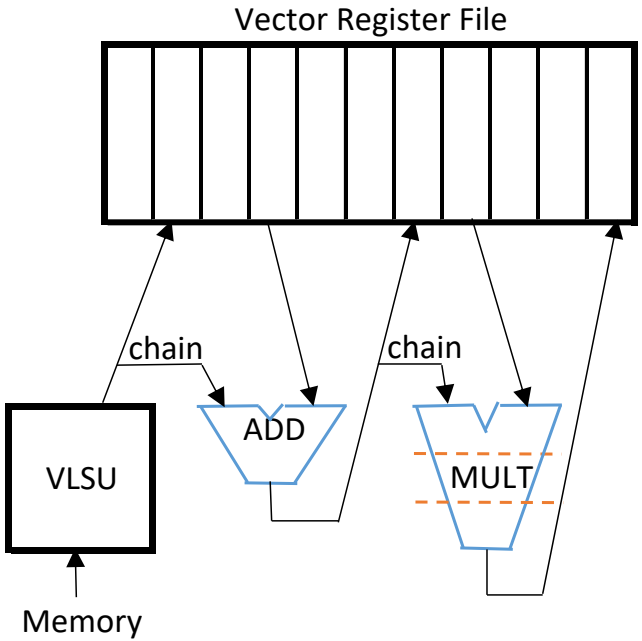



*Newell, R., "RISC-V Vector Extension Proposal Snapshot," Microsemi, Microchip Technology Inc., 2018*

# Vector Instruction Chaining

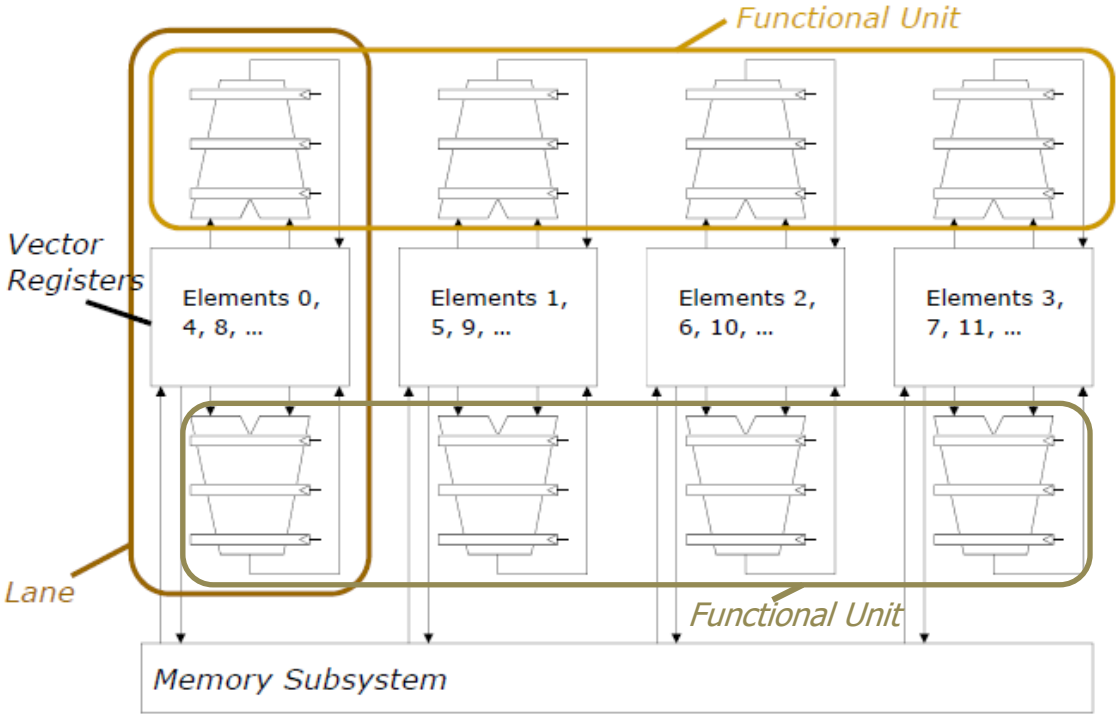
- Independent functional units are needed
  - Partial data of an instruction can be forwarded and operations are scheduled to be executed in parallel
  - **Data chaining can happen only if the instructions operate on multiple SIMD widths**

- VLEN=SIMD width, single instruction/cycle
  - The operation is on multiple elements but it is basically pipelining, not chaining
  - The instructions must be set for a vector length that is greater then the SIMD width in order for chaining to happen



# Chaining – Vector Unit Structure

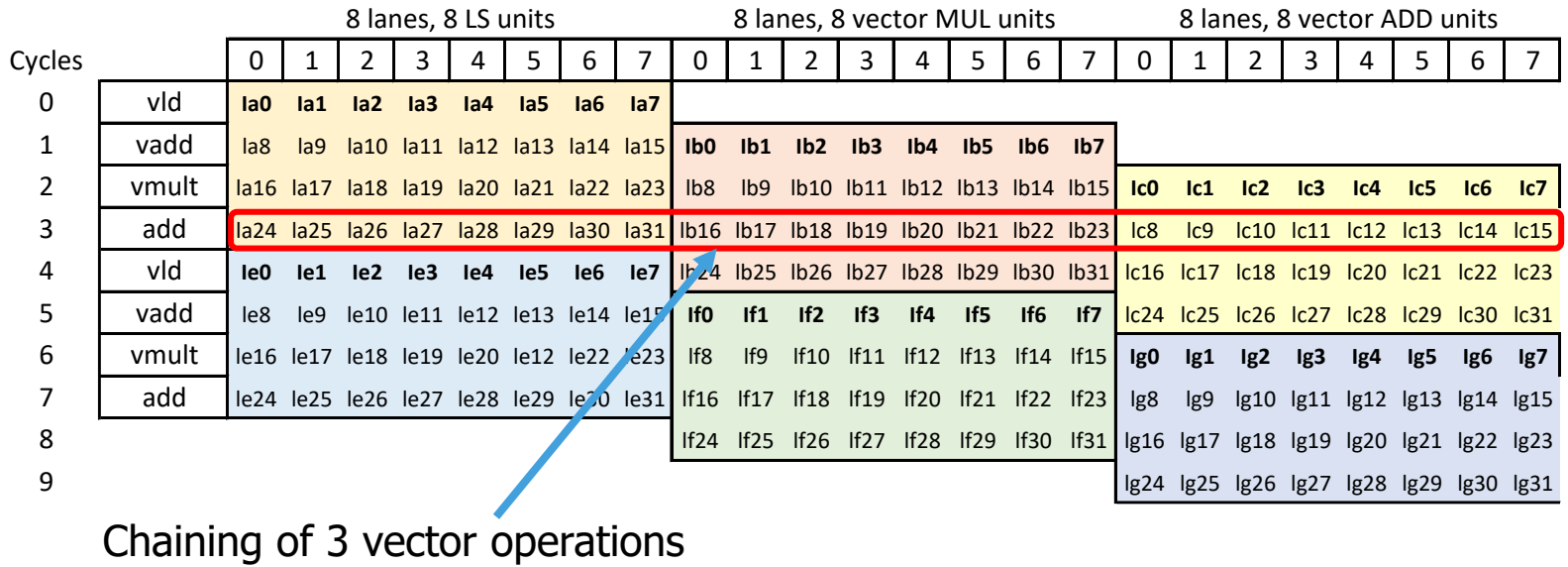
- Four lanes, 4 functional units to fetch/execute independent data




16 elements, 4 lanes:
- Elements 0-3
- Elements 4-7
- Elements 8-11
- Elements 12-15

# Chaining

- 32 elements, 8 lanes = 8 units to fetch/execute 8 independent elements
  - vld: fetch data 32 elements in 4 clock cycles, 8 elements per cycle
  - Vector add/mult: each vector instruction passes through the functional units 4 times



# Vectorize Loop Iterations

- X and Y are vectors, a is scalar, n is a large number
- For N=64, dynamic instructions
  - Processor = 64x9 + 2 = 578:
  - Vector = 6 (5 vectors and 1 scalar)

C Code

```
for (i=0; i <n;  i=i+1)
    Y[i] = a * X[i] + Y[i];
```

Processor/SIMD Code

```
        L.D       F0,a
        DADDIU    R4,Rx,#512
Loop:   L.D       F2,0(Rx)
        MUL.D     F2,F2,F0
        L.D       F4,0(Ry)
        ADD.D     F4,F4,F2
        S.D       F4,9(Ry)
        DADDIU    Rx,Rx,#8
        DADDIU    Ry,Ry,#8
        DSUBU     R20,R4,Rx
        BNEZ      R20,Loop
```

Vector Code

```
L.D       F0,a      → scalar
LV        V1,Rx
MULVS.D   V2,V1,F0
LV        V3,Ry     } vectors
ADDVV.D   V4,V2,V3
SV        V4,Ry
```

RAW Dependencies

Operate on 64 elements

*From: Patterson, D., CS252, UCB*

# Advantages of Vector Architecture

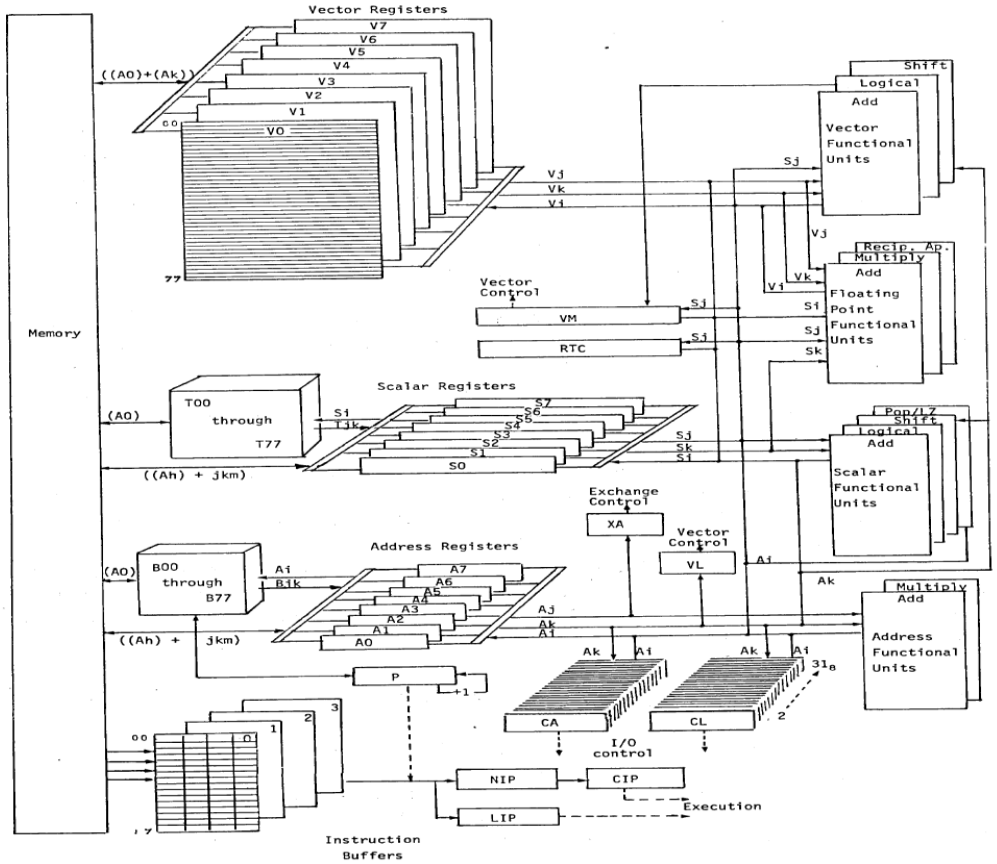
- From Spec92fp*
  - Static Instructions: scalar/vector = 4.5X
  - Dynamic Instructions: scalar/vector = 23.8X
  - Operations: scalar/vector = 1.4X (the extra operations are loop overheads)
- Most efficient way for data parallel processing
  - High computation throughput (operate on multiple elements at the same time)
  - Require high memory bandwidth, not low latency, data operation is chaining. The data memory is interleaving multiple banks for higher memory bandwidth
- Orthogonal to microprocessor architecture
  - Scalar or superscalar microprocessor with vector unit, e.g. Cray X1, Alpha Tarantula, Andes NX27V
  - Parallel vector processors, e.g. NEC Earth Simulator, Broadcom Calisto

**Quintana, F., et.al., "An ISA comparison between Superscalar and Vector Processors," U. of Las Palmas de Gran Canaria*

# Example: Cray 1

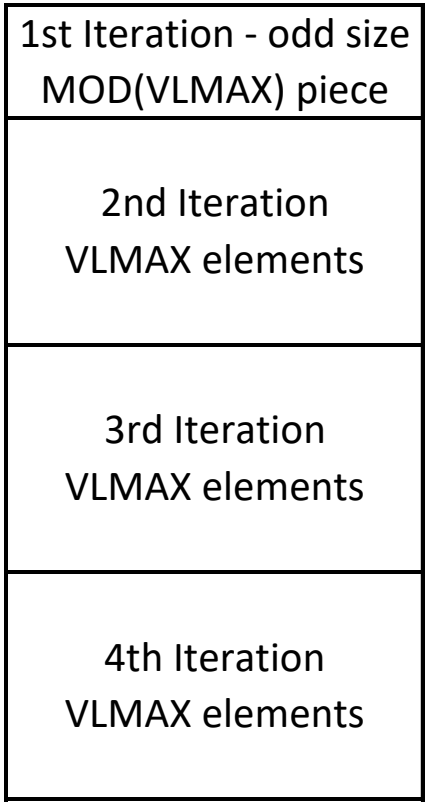
- Scalar and vector mode
- 8 64-element vector registers
  - SEW = 64-bit
  - VLEN = 64x64 = 4096-bit
- 8 64-bit scalar registers
- 8 24-bit address registers
- 16 memory banks
  - Bank latency = 11 cycles
  - Sustain 16 parallel accesses to different banks
- 77 x 104 x 57 inches, 5.5 tons, 115 kWatt (Wikipedia)

*"Cray-1 Computer System Hardware Reference Manual,"1977, Cray Research, Inc.*



# Strip Mining

- When the size of vector > VLMAX
  - size MOD(VLMAX) = the number of elements in the first iteration
  - n/VLMAX = number of iterations for each VLMAX time
- Example of application VL = 200 elements
  - 200 MOD(64) = 8 elements in first iteration
  - 200/64 = 3 iterations of 64 elements
- Strip Mining is assumed to be done in software similar to loop operation for VLMAX
  - vl or vector mask register is used for the first iteration



# Technical Challenges of Vector Architecture

- Complexity of vector register file (VRF) with VLEN=512-bit
  - Complexity of handling read and write ports with multiple lanes are much more expensive in VRF in comparison to scalar or FP register files
  - Performance is related to the number of read and write ports
  - The number of functional units are also limited by the number of read/write ports. Stalling of writing back to VRF means additional 512-bit buffers and back pressure to previous pipeline stages
- Microprocessor methods of handling out-of-order execution and retirement become expensive for vector processor
  - Register renaming and/or re-order buffer added several vector registers of 512-bit
- Expensive to implement precise exception for external data errors and long interrupt latency
  - In AI or embedded market, the precise exception is less important
- Vector loads/stores are expensive, especially for stride and index
- *Andes NX27V* ***vector processor simplifies many of the above issues***

# Applications of Vector Processing

- Scientific computations (astrophysics, atmospheric, ocean modeling, bioinformatics, physics, biomolecular simulation, chemistry, fluid dynamics, material sciences, quantum chemistry)
- Engineering computations (computer vision and image, data mining and data intensive computing, CAD/CAM engineering analysis, military applications, vlsi design)
- Multimedia Processing (compress., graphics, audio synth, image proc.)
- Standard benchmark kernels (Matrix Multiply, FFT, Convolution, Sort)
- Artificial Intelligent
- Lossy Compression (JPEG, MPEG video and audio)
- Lossless Compression (Zero removal, RLE, Differencing, LZW)
- Cryptography (RSA, DES/IDEA, SHA/MD5)
- Speech and handwriting recognition
- Operating systems/Networking (memcpy, memset, parity, checksum)
- Databases (hash/join, data mining, image/video serving)
- Language run-time support (stdlib, garbage collection)

*Source: Rochester Institute of Technology, "Introduction to Vector Processing," Shaaban, EECC722*

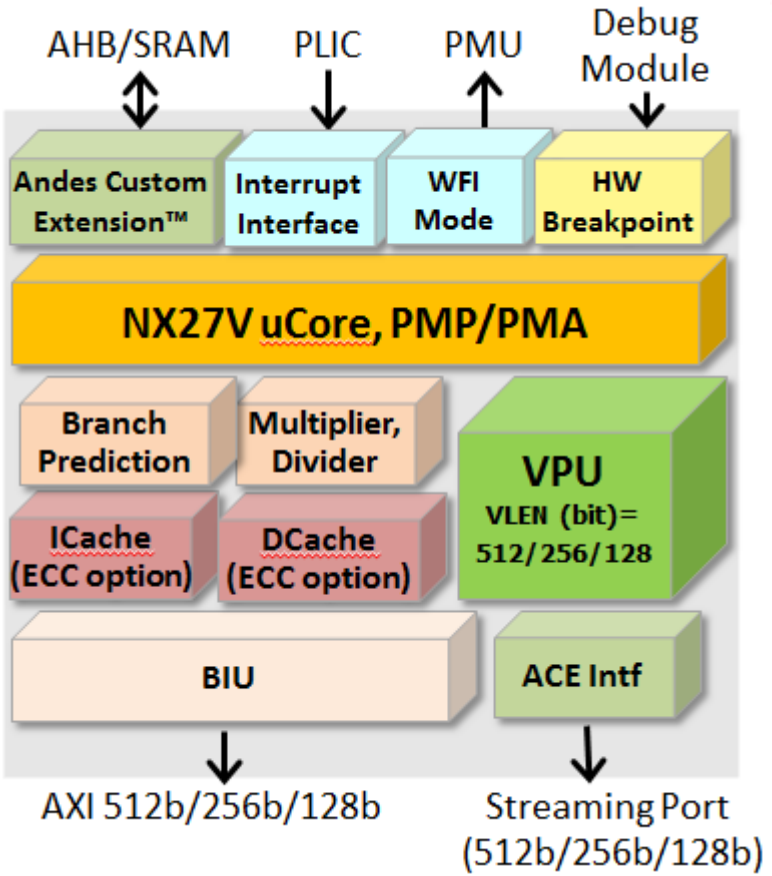
# Vectorizing Media Processing

- Kernel
  - Matrix transpose/multiply
  - DCT (video, communication)
  - FFT (audio)
  - Motion estimation (video)
  - Gamma correction (video)
  - Haar transform (media mining)
  - Median filter (image processing)
  - Separable convolution (img. proc.)
- Vector length
  - # vertices at once
  - image width
  - 256-1024
  - image width, iw/16
  - image width
  - image width
  - image width
  - image width

*(http://www.research.ibm.com/people/p/pradeep/tutor.html)*

# Overview of NX27V

- **AndeStar V5 architecture:**
  - RV64GCN+ Andes V5 Extensions
  - RV Vector extension (RVV)
- **5-stage pipeline, single-issue**
  - Optional branch prediction
- **I/D caches**
  - Caches: 8KB to 64KB
  - I$/D$ prefetch
  - HW unaligned load/store accesses
  - 16 non-blocking outstanding data accesses
- **Wide data paths to feed VPU:**
  - Cached and uncached RVV load/stores
  - Streaming Ports for ACE loads/stores



# Follow Andes, Find Latest Trends


# Thank you!

ANDES
TECHNOLOGY