RISC-V Vector Processors Generator by Chisel

Liwei Ma
Liwei.Ma@Sifive-China.com

Outline

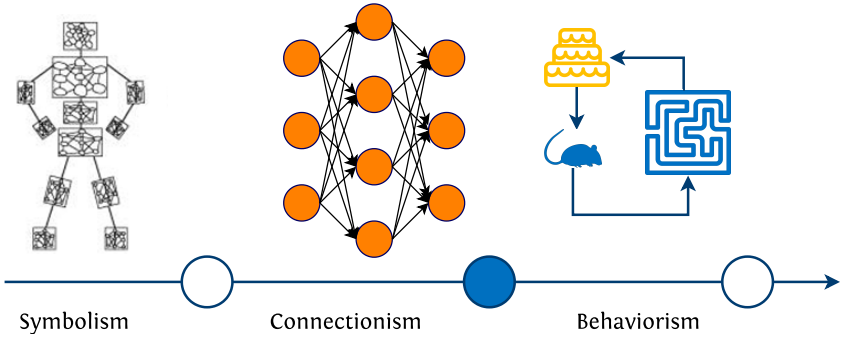
1. IC Industry Environment Changed
2. RISC-V Vector Processors
3. Hardware Agile Design by Chisel

SiFive

IC Industry Environment Changed

Change I: Computation Paradigm Shift

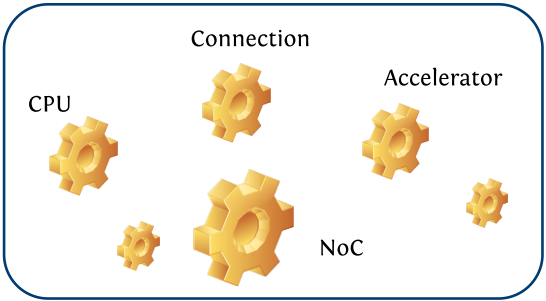
- Logic-based computation ⇒ raw-data-based computation



SiFive

Changes II: Productivity Shift

- Company-wide local workspace ⇒ cross-company cloud workspace

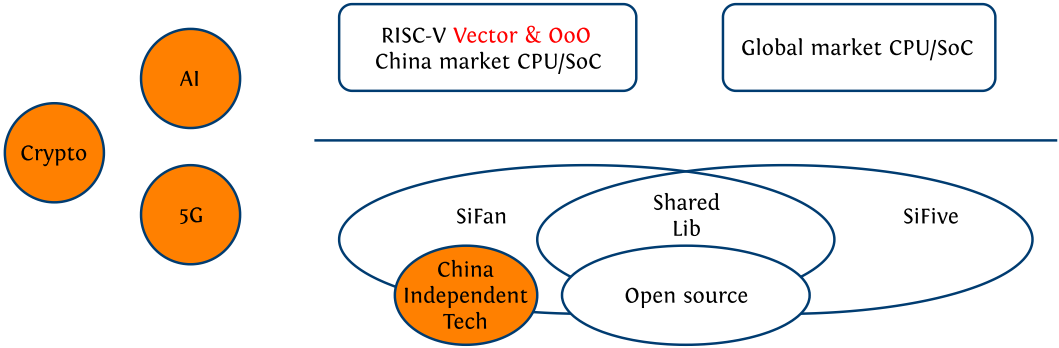


Cloud workspace with Chisel

SiFive

SiFive China CPU Development Roadmap

- Focused on RISC-V vector processors for AI, crypto and 5G.



RISC-V Vector Processors

Why RISC-V Vector ISA Matters

X86	ARM	RISC-V
		
PC Scalar	Mobile Scalar	AI-IoT Vector
Symbolism	Symbolism	Connectionism

Comparision of RISC-V Vector and GPU and ASIC

GPU

- It is huge, and hungry.

ASIC

- It is slow and fragmented.

RISC-V Vector

- It is unified programmable.

SiFive

RISC-V Vector ISA Introduction

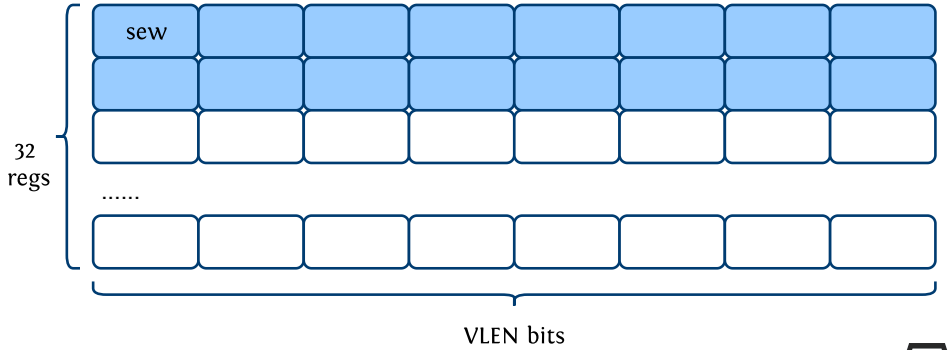
- 367 vector instructions VS 296 scalar instructions

Config	`setvl/setvli` vtype (vsew, vlmul) and vl (length)
Load/store	unit stride, stride, indexed and first fault
Atomic OPs	atomic memory read-modify-write
Integer	logic, bitwise, and integer arithmetic
Fixed-point	fixed-point arithmetic
Floating-point	floating-point arithmetic
Reduction	reduce a vector to a scalar
Mask	mask helper instructions
Permutation	move element around in a vector

SiFive

RISC-V Vector Register File

- Variable standard element width (vsew)
- Vector register grouping (vlmul)

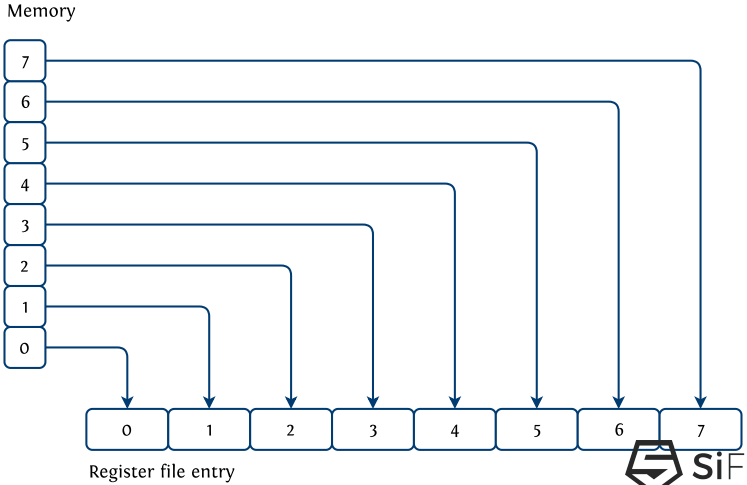


SiFive

RISC-V Vector Load Store Instructions

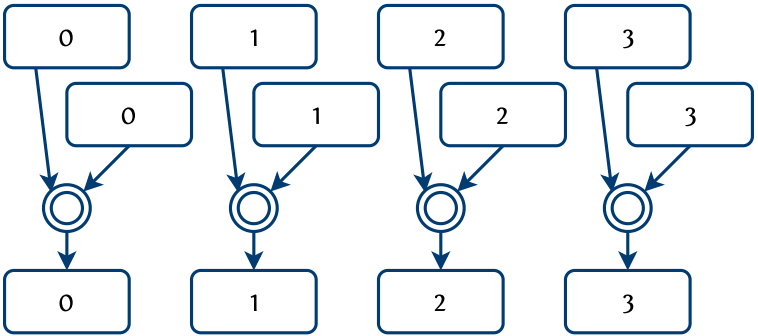
An example of v1h, vsew = 32, vlmul = 2, VLEN = 256 bits

- align different sew
- sign or zero extended
- unit stride, stide, indexed

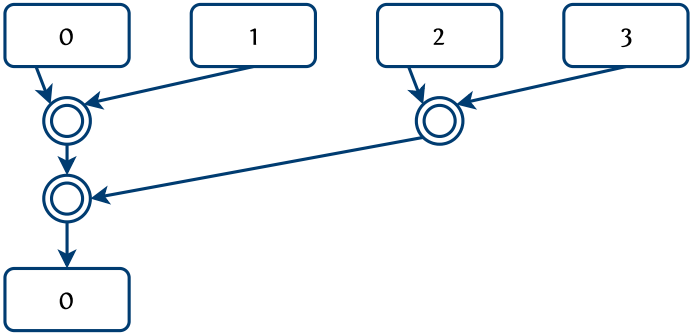


SiFive

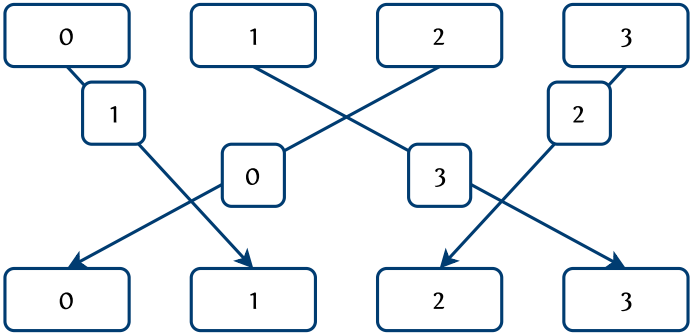
Element-wise Vector Computation Instructions



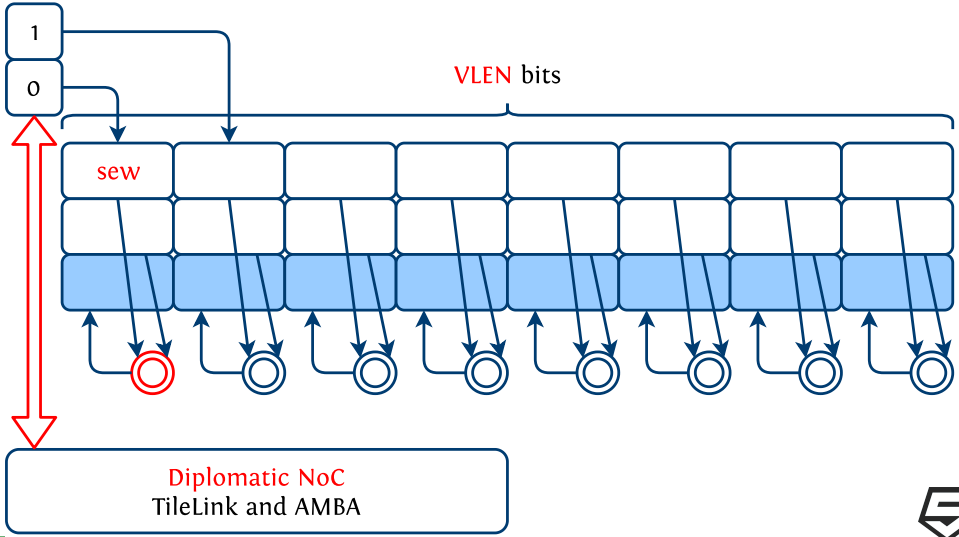
Reduction Vector Computation Instructions



Permutation Vector Computation Instructions



RISC-V Vector Processors Generator



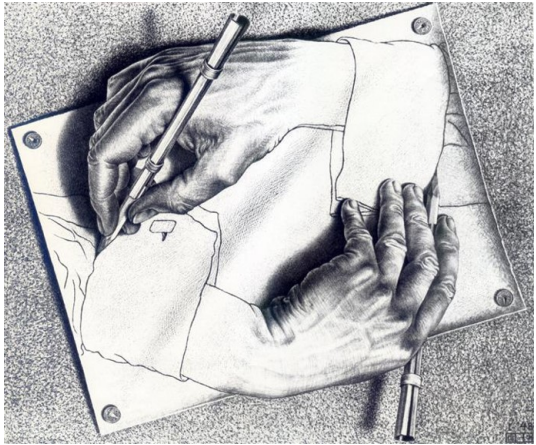
SiFive

Hardware Agile Design by Chisel

Chisel is The Interface For AI Self-Evolution

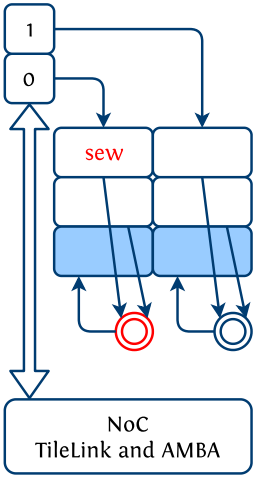
Rebuild IC Industry with AI

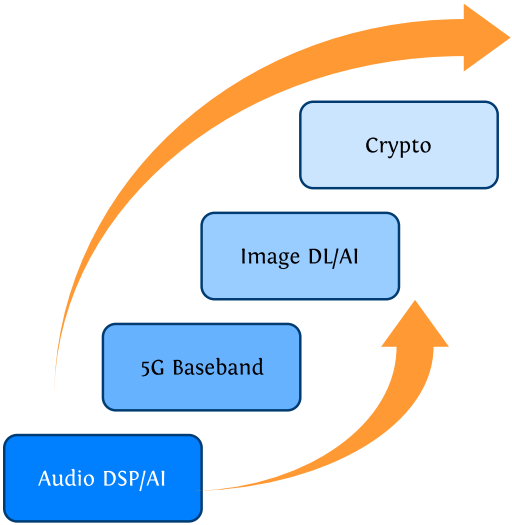
- AI explore IC design space
- Chisel is the interface for AI to access IC



SiFive

SiFive China Vector Processor Roadmap





Sand: Chisel-based RISC-V Vector Formal SPEC

- Inspired by RISC-V Scalar SPEC

Forvis	BlueSpec	Haskell
Grift	Galois	Haskell
Sail	Cambridge	Sail
Riscv-plv	MIT	Haskell
Kami	SiFive	Coq

- Why not a SPEC in the design language?

Sand	SiFan	Chisel

- Easy to read, check and reference
- Synthesizable verification

SiFive

Instruction Implementationn Example

```
1  object VREDSUM {
2    val pattern = BitPat("b000000??????????010????1010111")
3    def execute(x: UInt, y: UInt) = x + y}
4
5  val group = List(VREDSUM, VREDMAXU, ...)
6
7  val line2 = vsrc2.asVecOfUInt(lineElems)
8  val line1 = vsrc1.asVecOfUInt(lineElems)
9  val result = line2.fold (line1(0)) { case(x,y) =>
10   MuxLookup(funct6, 0.U,
11     group map {op.FUNCT6 -> op.execute(x,y)})}
```

SiFive

Instruction Execution Example

vredsum program snippet

```
auipc   t0,0xa0000
addi    t0,t0,-140 # 20000100 <vdata2>
vle.v   v0,(t0)
auipc   t0,0xa0000
addi    t0,t0,-408 # 20000000 <begin_vector_rom>
vle.v   v8,(t0)
auipc   t0,0xa0000
addi    t0,t0,60 # 200001e0 <vinitdata>
vle.v   v16,(t0)
vredsum.vs      v16,v8,v0
vslidedown.vi   v24,v16,1
vmv.x.s s8,v24
li      t4,0
li      gp,2
bne     s8,t4,800003b4 <fail>
vmv.x.s s8,v16
lui     t4,0xe
addi    t4,t4,-620 # dd94 <begin_vector_rom-0x1fff226c>
li      gp,2
bne     s8,t4,800003b4 <fail>
```

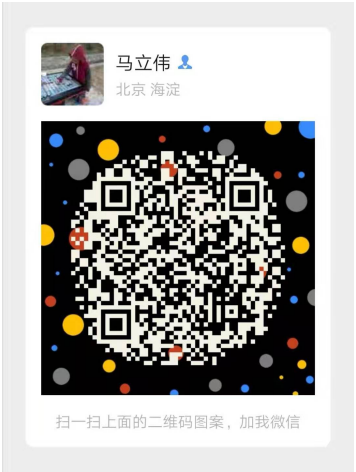
execution result

```
PRINT @80000144 (v0 )    = [ 3b 3e e1 41 16 36 fb c2 ]
PRINT @80000148 (v8 )    = [ 32 43 f6 a8 88 5a 30 8d ]
PRINT @8000014c (v16)    = [ 61 bf ed f7 24 29 b0 23 ]
PRINT @80000154 (v16)    = [ 0 0 0 0 0 0 0 74 ]
PRINT @80000160 (v24)    = [ 0 0 0 0 0 0 0 0 ]
PRINT @80000174 (v24)    = [ 0 0 0 0 0 0 0 0 ]
PRINT @800001b0 (v0 )    = [ 3b3e e141 1636 fbc2 ]
PRINT @800001b4 (v8 )    = [ 3243 f6a8 885a 308d ]
PRINT @800001b8 (v16)    = [ 61bf edf7 2429 b023 ]
PRINT @800001c0 (v16)    = [ 0 0 0 dd94 ]
PRINT @800001cc (v24)    = [ 0 0 0 0 ]
PRINT @800001e0 (v24)    = [ 0 0 0 0 ]
PRINT @80000220 (v0 )    = [ 3b3ee141 1636fbc2 ]
PRINT @80000224 (v8 )    = [ 3243f6a8 885a308d ]
PRINT @80000228 (v16)    = [ 61bfedf7 2429b023 ]
PRINT @80000230 (v16)    = [ 0 d0d522f7 ]
PRINT @8000023c (v24)    = [ 0 0 ]
PRINT @80000250 (v24)    = [ 0 0 ]
```

SiFive

How We Collaborate on RISC-V Vector

How We Collaborate on RISC-V Vector



How We Collaborate on RISC-V Vector

Program in RISC-V-V

- Try Spike
- Try GCC assembler

Execute on Rock

- Sand, introduced
- Rock, on the way

Explore new market

- AI
- 5G
- Crypto

SiFive