The SiFive Vector Processor

Mark Thronson

SiFive Senior Director, RISC-V Core IP Prod Mgmt

DEC. 11, 2019
Leading the charge on vector development

- Inventors of RISC-V were driven by desire for an open architecture extensible to support computer architecture research
  - Krste Asanovic, Yunsup Lee, Andrew Waterman

- SiFive are key contributors to RISC-V vector ecosystem:
  - RISC-V Vector working group
    - Chair - Krste Asanovic
  - RISC-V assembler – developed vector support → contributed to open source
    - Maintainer – Jim Wilson, with key contributors Kito Cheng, Nelson Chu
  - RISC-V SPIKE ISS – developed vector support → contributed to open source
    - Maintainer – Andrew Waterman, with key contributors Dave Wen, Chih-Min Chao

... And building on this expertise to offer SiFive Intelligence Processors
SiFive Intelligence Processors

A portfolio of RISC-V Core IP with vector intelligence (VI)

- Support for RISC-V ISA Vector (RVV) Extensions
  - ISA defined to support a wide range of implementation and use cases
  - Scalability with binary compatibility
  - Single development environment - shared ISA for scalar & vector operations

- 2020 portfolio extending from microcontroller to robust application processor IP cores

- Scalable vector implementations in software and hardware
  - Configurable vector operation size in software via vector ISA
  - Configurable hardware microarchitecture capabilities delivered in SiFive VI cores
SiFive Vector Intelligence (VI) Processor Roadmap

2020 and beyond

Vi2

Performance

Vi7

DSP

2020 and beyond

Voice/Audio

Vision Processing

Time Series

5G

SmartVoice/AI

IoT

Edge

Datacenter

FAD

ADAS

Datacenter

Vi8
SiFive VI2 Series Core IP Highlights

For Smart voice/audio, general DSP applications, and beyond

- Support for element data types common to DSP, AI, and general compute applications
  - Integer/Fixed/Floating Point types across 8-32 bits

- SiFive microarchitecture builds on RVV ISA scalability
  - Separate scalar and vector datapaths
  - Initial vector unit default config = 128b datapath, 512b VLEN
    - Extends to 4096b vectors via LMUL
  - Datapath width, vector register length (VLEN), data types, & other parameters are configurable (↑ or ↓)

- Leverages existing SiFive core IP debug and trace hardware solutions
  - Can easily work with 3rd party tool offerings
  - Freedom Studio will support formatting and display of the vector registers and control registers
SiFive VI2 DSP kernel performance

> 9x speed-up on 32b Floating Point across a range of DSP library functions

- SiFive VI2 core configured @ 128b datapath width (VLEN 512b)
- Vector vs Scalar performance = vectorized functions in assembly vs C-source library compiled to scalar

**RVV F32 vs Scalar F32**

<table>
<thead>
<tr>
<th>Category</th>
<th>RVV vs Scalar GEOMEAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BasicMathFunctions</td>
<td>15.0x</td>
</tr>
<tr>
<td>ComplexMathFunctions</td>
<td>11.7x</td>
</tr>
<tr>
<td>FilteringFunctions</td>
<td>5.5x</td>
</tr>
<tr>
<td>MatrixFunctions</td>
<td>5.3x</td>
</tr>
<tr>
<td>SupportFunctions</td>
<td>13.0x</td>
</tr>
<tr>
<td>OVERALL GEOMEAN</td>
<td>9.3x</td>
</tr>
</tbody>
</table>

- input size: 32
- input size: 16
- input size: 32
- input size: [4x4] x [4x4]
- input size: 32
DSP kernel: mat_mul_f32

22.4x speed up – RVV vs Scalar (unrolled)

Speed-up for [16x16] x [16x16] elements

Scalar (unrolled C)

/* Loop unrolling: Compute 4 MACs at a time. */
colCnt = numColsA >> 2U;
/* matrix multiplication */
while (colCnt > 0U) {
/* c(m,n) = a(1,1) * b(1,1) + a(1,2) * b(2,1) + .... + a(m,p)*b(p,n) */
/* Perform the multiply-accumulates */
sum += *pIn1++ * *pIn2;
pIn2 += numColsB;
sum += *pIn1++ * *pIn2;
pIn2 += numColsB;
/* k_loop:
vfmacc.vf v0, ft0, v16
add bkp, bkp, bstride
flw ft4, (amp)
add amp, amp, astride
vmacc.vf v1, ft1, v16
addi kt, kt, -1 # Decrement k counter
flw ft5, (amp)
add amp, amp, astride
vmacc.vf v2, ft2, v16
flw ft6, (amp)
add amp, amp, astride
flw ft7, (amp)
vmacc.vf v3, ft3, v16
add amp, amp, astride
flw ft8, (amp)
add amp, amp, astride
vmacc.vf v4, ft4, v16
flw ft9, (amp)
add amp, amp, astride
vmacc.vf v5, ft5, v16
flw ft10, (amp)
add amp, amp, astride
...
DSP kernel: mat_mul_f32 performance example

31.2x speed up – RVV vs Scalar (without unrolling)

Speed-up for [16x16] x [16x16] elements

Scalar (C without unrolling)

```c
... while (colCnt > 0U) {
    /* c(m,n) = a(1,1) * b(1,1) + a(1,2) * b(2,1) + .... + a(m,p) * b(p,n) */
    /* Perform the multiply-accumulates */
    sum += *pIn1++ * *pIn2;
    pln2 += numColsB;
    /* Decrement loop counter */
    colCnt--;
}
...
```

Vector (assembly)

```assembly
... k_loop:
    vfmacc.vf v0, ft0, v16
    add bkp, bkp, bstride
    flw ft4, (amp)
    add amp, amp, astride
    vfmacc.vf v1, ft1, v16
    addi kt, kt, -1 # Decrement k counter
    flw ft5, (amp)
    add amp, amp, astride
    vfmacc.vf v2, ft2, v16
    flw ft6, (amp)
    add amp, amp, astride
    flw ft7, (amp)
    vfmacc.vf v3, ft3, v16
    add amp, amp, astride
    flw ft8, (amp)
    add amp, amp, astride
    vfmacc.vf v4, ft4, v16
    flw ft9, (amp)
    add amp, amp, astride
    vfmacc.vf v5, ft5, v16
    flw ft10, (amp)
    add amp, amp, astride
    ...
...
SiFive Vector Evaluation Platform (VEP)

- A complete evaluation and development environment for SiFive’s VI2 Core IP
  - Algorithmic evaluation and porting platform for customers and partners

- Initial VEP support includes:
  - Toolchain
    - RISC-V assembler with vector extensions support
    - RISC-V gcc compiler with SiFive vector intrinsics
  - Optimized libraries for RISC-V vector extensions
    - DSP functions library (written for compatibility with CMSIS DSP APIs)
    - Plus highly optimized kernels for FFT, matrix math, etc.
  - Execution models
    - Performance model for SiFive VI2 core
    - SPIKE instruction set simulator (ISS)

Available now – customers and partners contact SiFive

```c
#include <rvv_vector.h>
#include <stddef.h>

void vec_add_rvv(int *a, int *b, int *c, size_t n) {
    size_t vl;
    rvv_int32m2_t va, vb, vc;
    for (vl = rvv_setvl_32m2 (n);
         (vl = rvv_setvl_32m2 (n));
    va = rvv_add_rv_v32m2 (b, vl); }
```
Vector intrinsics programming example: Vector add

Intrinsic code example

```c
#include <rvv_vector.h>
#include <stddef.h>

void vec_add_rvv(int *a, int *b, int *c, size_t n) {
    size_t vl;
    rvv_int32m2_t va, vb, vc;
    for (;vl = rvv_setvl_32m2 (n);) {
        vb = rvv_le_int32m2 (b);
        vc = rvv_le_int32m2 (c);
        va = rvv_add_vv_int32m2 (vb, vc);
        rvv_se_int32m2 (a, va);
        a += vl;
        b += vl;
        c += vl;
        n -= vl;
    }
}
```

Generated assembly

```
.Lloop_entry:
    vsetvli a5,a3,e32,m4 # Loop condition
    beq a5,zero, .Lexit
    vle.v v2,0(a1) # Load from b
    vle.v v4,0(a2) # Load from c
    vadd.vv v2,v2,v4 # Addition
    vse.v v2,0(a0) # Store to a
    sub a3,a3,a5 # Decrement count
    slli a4,a5,2 # Multiply length by four bytes/element
    add a1,a1,a4 \ #
    add a2,a2,a4 | Bump pointer
    add a0,a0,a4 /
    j .Lloop_entry

.Lexit:
    ret
```
Takeaways

- 2020 is the year for vectors in RISC-V – SiFive has big plans to make that a reality

- Roadmap of SiFive Intelligence Processors covering a wide range of implementation points and use cases

- SiFive investment in hardware AND software to enable solutions around RISC-V vector extensions

- SiFive Vector Evaluation Platform (VEP) is here to enable evaluation and development of hardware and software

S/W partners/algorithm developers – contact SiFive to vectorize your solutions for RISC-V!
Thank You

facebook.com/SiFive  @SiFive  linkedin.com/company/sifive  sifive.com

©2019 SiFive, Inc. All rights reserved. All trademarks referenced herein belong to their respective companies. This presentation is intended for informational purposes only and does not form any type of warranty.

Certain information in this presentation may outline SiFive’s general product direction. The presentation shall not serve to amend or affect the rights or obligations of SiFive or its licensees under any license or service agreement or documentation relating to any SiFive product. The development, release, and timing of any products, features, and functionality remains at SiFive’s sole discretion.

FOOTNOTES: