Person Detector System using ORCA (RISC-V) with SVE (Vectors) and CNN Accelerator...

in about 5,000 4-input LUTs and 5mW power

Guy Lemieux
CEO VectorBlox & Prof. Univ. British Columbia

7th RISC-V Workshop, Nov 28-29, 2017
• Note:

  – This talk uses “VectorBlox Vector Instructions”

  – NOT the same as “RISC-V Vectors” discussed earlier
Smarter Sleep
Smarter TVs

(stay on)

(turn off)
Smarter Doors
1st Prototype

Lattice FPGA iCE40 UltraPlus
About 2.5mm x 2.5mm
5,280 LUTs
1Mb SRAM

Camera
640 x 480 native
Only 32 x 32 needed
Person Detector Demo
Deep Learning Database

People

Not People
Binarized Convolutional Neural Network

- Inspired by BinaryConnect
  Database: CIFAR10 (10 categories)
  1b weights (-1/+1), float32 activations, 92% accurate

- Changes
  Database: Custom Person (1 category)
  1b weights (-1/+1), int8 activations, 98% accurate
  Reduced network size (40x smaller)
Person Detector System

VectorBlox ORCA

- VHDL (BSD License)
- Portable (5 FPGA vendors)
- Fast (200MHz fully pipelined)
- Small (< 2000 LUT4s)
- Configurable
  + Streaming Vector Extensions
  + Binary CNN Accelerator
Streaming Vectors

- Vector instructions **operate only on Stream Memory**
  - Address generator hardware
  - Streams data through RISC-V ALU
- C to alloc vector of 8 words
  
  ```
  vbx_word_t *vsrc1 = vbx_sm_malloc( 32 );
  ```
# Streaming Vector Extensions

<table>
<thead>
<tr>
<th>DMA</th>
<th>SVE-DMA</th>
<th>v.dma2mem, v.dma2vec, v.dma2d</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMD</td>
<td>SVE-P</td>
<td>byte &amp; halfword versions of M / I / base</td>
</tr>
<tr>
<td>SVE-base</td>
<td>v.mov, v.cmv_z, v.cmv_nz, v.setgt/u, v.custom0 ... v.custom7, vlen, vstride, vquery</td>
<td></td>
</tr>
</tbody>
</table>
Extended SVE

• Base SVE
  – 32b encoding, custom-0 opcode slot
  – 1D / 2D vectors/matrices
  – Packed SIMD
  – ISA supports CPU scaling: 1 ALU ... 128+ ALUs

• Extended SVE
  – 64b encoding
    • Richer data types like float32, float16
    • Type conversions
  – Adds 3D vectors/matrices/volumes
  – Masked execution to bypass “nop slots” (density time masking)
FIR filter (12x speedup)

- **RV32IM**
  
  00000030 <scalar_fir(long*, long*, long*, int, int)>:
  30: 40e686b3 sub a3,a3,a4
  34: 06d05263 blez a3,98 <.L6>
  38: 00269693 slli a3,a3,0x2
  3c: 00271e13 slli t3,a4,0x2
  40: 00d50eb3 add t4,a0,a3
  44: 01c60e33 add t3,a2,t3
  48: 00100f13 li t5,1
  
  0000004c <.L10>:
  4c: 005a683 lw a3,0(a1)
  50: 0062803 lw a6,0(a2)
  54: 00460793 addi a5,a2,4
  58: 0058893 mv a7,a1
  5c: 03068833 mul a6,a3,a6
  60: 01052023 sw a6,0(a0)
  64: 02ef5263 ble a4,t5,88 <.L11>
  
  00000068 <.L13>:
  68: 0048a683 lw a3,4(a7)
  6c: 0007a303 lw t1,0(a5)
  70: 00478793 addi a5,a5,4
  74: 00488893 addi a7,a7,4
  78: 026686b3 mul a3,a3,t1
  7c: 00d80833 add a6,a6,a3
  80: 01052023 sw a6,0(a0)
  84: fefe12e3 bne t3,a5,68 <.L11>
  
  00000088 <.L11>:
  88: 00450513 addi a0,a0,4
  8c: 00458593 addi a1,a1,4
  90: faae9ee3 bne t4,a0,4c <.L10>
  94: 0008067 ret

- **RV32IM + SVE**
  
  00000000 <vector_fir(long*, long*, long*, int, int)>:
  0: 00007b7 lui a5,0x0
  4: 00e7a023 sw a4,0(a5)
  8: 40e686b3 sub a3,a3,a4
  c: 02d05063 blez a3,2c <.L1>
  10: 00269693 slli a3,a3,0x2
  14: 00d586b3 add a3,a1,a3
  18: 08c5fe2b vlen a4,a6,a6
  
  0000001c <.L3>:
  1c: a6e50cab v.mul.vvw.acc a0,a1,a2
  20: 00458593 addi a1,a1,4
  24: 00450513 addi a0,a0,4
  28: fed598e3 bne a1,a3,1c <.L3>
  
  0000002c <.L1>:
  2c: 0008067 ret

Vectors:
loop1: 1 instruction + no stalls
loop2: 16 bytes code

No vectors
loop1: 8 instructions + stalls
loop2: 72 bytes code
Custom Streaming Accelerators

Data Memory

Different addresses

Memory Bus

SVE

Stream Memory

ORCA Processor

Domain-specific Streaming Pipeline
Streaming Convolution Accelerator
Streaming Convolution Accelerator
Streaming Convolution Accelerator
Streaming Convolution Accelerator
Streaming Convolution Accelerator

~20 ops
~20 ops
~20 ops
Person Detector Performance

- 24 MHz, 1 cycle per instruction (4 stages)

  - Overall 71x speedup
  - Effectively a 1.7 GHz RISC-V processor
  - Vectors 8x speedup
  - Binary CNN 73x speedup

- Power ~5mW
Area Breakdown

4,852 4-input LUTs (92% of FPGA)

- ORCA: 44%
- Vectors: 18%
- CNN: 18%
- Camera: 6%
- Other: 14%
Person

Confidence scores for 10 categories:
- float 32
- int 8
Airplane
Dog
Person
Instruction Sets Want to be Free!

Releasing our Vector ISA as an open spec
Recently joined RISC-V Vector Working Group
Will discuss as possible alternative
SVE vs “RISC-V Vector Extension”

- SVE is a “memory-to-memory” architecture
  - Challenges “conventional wisdom” RISC (reg-to-reg/load-store)
- Advantages
  - No named vector registers in ISA
    - Uses C pointers into Streaming Memory (scalar registers)
    - No register allocation, no compiler changes needed
  - Performance
    - Free loop unrolling
    - No saving/restoring vector data on accelerated function calls (call stack)
    - Up to 10,000x speedup (N-body problem with custom pipeline)
  - No storage wasted with Streaming Memory
    - Any Number of Vectors, Any Length Vector
    - Vectors & subword SIMD elements fully packed (no internal fragmentation)
    - Free software scratchpad (eg, if vectors not used)
- Easier/simpler hardware
  - Double-buffered DMA instead of prefetching + vector register renaming
So Long, and Thanks for All the Fish!!