High Performance DSP for Vision, Imaging and Neural Networks

Greg Efland, Sandip Parikh, Himanshu Sanghavi, Aamir Farooqui
2016 Hot Chips
Flint Center for the Performing Arts, Cupertino, California
August 21-23, 2016
Vision Processors
Emerging Applications

Mobile
HDR
Video Stabilizer
Face Detection

Automotive
Traffic Sign Recognition
Gesture Control

Drone
3D Vision

Security
People Detection

Wearables and IoT
Vision Processors
Opportunity

• Ever increasing demand for vision and image processing
  – Cameras everywhere
  – Dramatic complexity expansion of vision in cars, IoT, mobile, consumer
  – Emergence of compute-hungry neural networks

• Vision Processors are designed for the complex algorithms in computer vision and imaging
  – Programmable architectures for rapidly evolving and diverse applications
  – Memory architectures designed for vision and imaging application needs
  – Power-efficient solutions for vision and imaging applications
Cadence Tensilica Vision Processors

Evolution

• Architecture evolved over 4 generations
  – Member of large family of Cadence DSPs

• Key characteristics
  – Wide SIMD processing: 512-bit width, 64-/32-/16-way processing
  – Local memory based: multiple banks, multiple 512-bit load/store
  – FLIX™ instructions (VLIW): 5 slots, multiple formats, scalar / vector mix

• Delivered as soft IP
  – Customer extensible with new operations and interfaces
  – Customer configured according to requirements
Real Time Vision with Cadence

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Automotive, mobile, surveillance, IOT, CNN</td>
<td>Virtual Platforms, FPGA development systems, at-speed silicon</td>
<td>OpenCV, OpenVx, CNN Framework</td>
<td>One GUI for configuration, compilation, debug, modeling, RTOS</td>
<td>Fastest DSP for vision – classic and CNN</td>
<td>1B cores in imaging/video/vision</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Xtensa™ Foundation, Xtensa™ Processor Generator</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Hundreds of customers, thousands of designs, 5 billion cores</td>
<td></td>
</tr>
</tbody>
</table>

© 2016 Cadence Design Systems, Inc. All rights reserved.
Cadence Tensilica Vision Processors
Evolution

• Vision P6
  – Significantly improved performance on a wide range of imaging and vision applications compared to predecessors
  – Performance improvements achieved through incremental increases in area and power while maintaining efficiency and software compatibility

• Key areas of improvement
  – Data type diversity
  – Enhanced memory parallelism and data movement
  – CNN enhancements
Tensilica Vision P6 DSP

- Enhanced Memory Parallelism
  - SuperGather™

- Enhanced Data Movement
  - Aligning vector load/store
  - Integrated 2-D DMA

- CNN Enhancements
  - 256 MACs

- Diverse Data Type Support
  - 8/-16/-32-bit fixed-point
  - Single-/half-precision FP
Cadence Tensilica Vision Processors
Evolution

• Vision P6 status
  – Sampled to early engagement customers
  – General release October 2016
Data Type Diversity
Data Type Diversity

- Data type requirements vary by application
  - A wide range of supported data types enables a wide range of applications
  - Configurable data type support enables efficient targeted solutions

- Vision P6 adds vector floating-point support
  - Leverages existing resources – vector register files and load/store operations
  - Incremental cost – floating-point data paths
  - Optional – no cost for applications that don’t require floating-point

- Vision P6 computation data types
  - Fixed-point: 8-/16-/32-bit
  - Floating-point: single-/half-precision
Data Type Diversity
Fixed-Point Data Types

• Fixed-point
  – Primary data type for many applications
  – 8-bit, 16-bit most common computation types
  – 8-bit increasingly important – e.g. CNN network inference
  – 32-bit for precision – e.g. perspective warp mapping functions
  – Multiple 8x8/8x16/16x16 MACs per SIMD way

<table>
<thead>
<tr>
<th>Data Type</th>
<th>SIMD</th>
<th>MACs per Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit fixed-point</td>
<td>64-way</td>
<td>256 (8x8) / 128 (8x16)</td>
</tr>
<tr>
<td>16-bit fixed-point</td>
<td>32-way</td>
<td>64 (16x16)</td>
</tr>
<tr>
<td>32-bit fixed-point</td>
<td>16-way</td>
<td>16 (16x32) / 8 (32x32)</td>
</tr>
</tbody>
</table>
Data Type Diversity
Floating-Point Data Types

• Floating-point
  – For dynamic range – e.g. RANSAC
  – Ease porting of applications from other platforms
  – Support both scalar and N-way vector processing
  – Single-precision and half-precision independently configurable

<table>
<thead>
<tr>
<th>Data Type</th>
<th>SIMD</th>
<th>MADDs per Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bit single precision</td>
<td>16-way</td>
<td>16</td>
</tr>
<tr>
<td>16-bit half precision</td>
<td>32-way</td>
<td>32</td>
</tr>
</tbody>
</table>
Enhanced Memory Parallelism and Data Movement
Gather-Scatter
Opportunity

• A number of kernels access data in disparate memory locations

  – Not simple fixed-stride accesses: data in arbitrary rows and/or columns
  – Hard to efficiently vectorize using wide loads/stores since arbitrary movement of data across vector registers, particularly across rows, can be expensive
  – If data is sparse, wide loads/stores also waste memory bandwidth and energy
Gather-Scatter

Opportunity

- **Vision P6** supports multiple banks per local data RAM
  - Up to 4 banks: multiple load/store operations and DMA transfers per cycle
  - Banks are 512 bits wide, interleaved on low-order address bits
  - Typically implemented using multiple narrow memories: up to 32 sub-banks

- Enable software to exploit independently addressable sub-banks
  - Incremental hardware increase leverages existing memory structure
  - Significant performance increase for a number of applications
SuperGather
Hardware Operations

• Gathers: vector of addr $\rightarrow$ vector register
  – Non-blocking operations – stall on data use
  – Up to 4 outstanding gather operations in flight

• Scatters: vector register $\rightarrow$ vector of addrs
  – Posted operations
  – Up to 2 outstanding scatter operations in flight

• Up to 32 8/16-bit elements, 16 32-bit elements read / written per cycle
  – Reads and writes to different sub-banks performed in parallel
  – Multiple reads or writes to same sub-bank address combined into single access
  – Reads overlapped across different gathers; writes overlapped across different scatters
Gather-Scatter Programming Model

• Gather and scatter operations are invoked via C intrinsics
  – C compiler schedules all operations including gather and scatter
  – Loop unrolling, software pipelining handles gather and scatter
  – C type qualifier ‘restrict’ supported for gather and scatter base pointers

• Example: update N independent histograms in parallel (HoG)
  – Placement of each histogram in a single sub-bank bounds conflicts

```
LOOP for vBins, vWeights in vInput:
    vOffsets = vBins * pitch + vBase
    vCounts = GATHER(base, vOffsets)
    SCATTER(vCounts + vWeights, base, vOffsets)
```
Gather-Scatter Performance

- Overall benchmark speedup relative to no gather-scatter
  – 2X to 10X over various kernels

- Gather cycle count reduction due to overlap
  – 0.5X to 1X image warp cases
Vector Load/Store, Shuffle Operations and DMA

• Efficient data reorganization to support computation
  – Aligning load/store operations access vectors of arbitrary length and/or alignment in memory
  – Shuffle operations perform arbitrary shuffles/routing of vector register data across SIMD lanes

• Integrated DMA
  – 1-D and 2-D transfers between local data RAM and external memory or within local data RAM
  – Arbitrary alignment of source, destination
  – Independent source and destination pitch
  – Up to 64 outstanding external memory requests to deal with large system memory latencies
Enhanced Memory Parallelism and Data Movement

Summary

• Memory parallelism and flexible data movement key to high performance across a wide range of application kernels
• Gather-scatter provides significant performance increase for modest incremental hardware increase by exploiting existing memory structure
• Memory bandwidth balanced with compute requirements

<table>
<thead>
<tr>
<th>Type</th>
<th>Peak Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aligned / Unaligned Loads</td>
<td>2 x 64 bytes/cycle</td>
</tr>
<tr>
<td>Aligned / Unaligned Stores</td>
<td>1 x 64 bytes/cycle</td>
</tr>
<tr>
<td>Gathers</td>
<td>64 bytes/cycle</td>
</tr>
<tr>
<td>Scatters</td>
<td>64 bytes/cycle</td>
</tr>
<tr>
<td>DMA</td>
<td>64 bytes/cycle</td>
</tr>
</tbody>
</table>
CNN Enhancements
Neural Network Applications

- Neural networks need lots of compute – especially multiply-add
- Two key compute engine metrics:
  - Scaling to high total compute
  - High multiply-add per watt
- Vision DSPs target emerging deployment opportunities
  - High total compute at good efficiency
  - Flexibility and programmability for related processing (e.g. ROI extraction) and evolving network structures
3D-Filter Interpretation of CNN Layer

Convolutional Layer

\[ y_i = \sum_{j=1}^{D} F_{ij} \circ x_j \]

One 3D Filter per Layer “i” Output

Each filter has \( L = H \cdot W \cdot D \) coefficients

There are \( N \) such filters
CNN Enhancements

Opportunity

- 3-D convolution: significant data reuse possible
  - Load bandwidth supports > 1 MAC per element
- Fixed-point types appropriate for computation
  - Moving to 8-bit types for inference
- Additional MACs enable significantly improved CNN kernel performance
  - Incremental hardware increase for additional MACs
  - 2X – 4X peak MAC performance of Vision P5
CNN Enhancements

Operations

- Paired 16x16, 8x16 and 8x8 MAC operations
  
  *Scalar x vector:* \(\text{acc}[i] += c_0 \times v_0[i] + c_1 \times v_1[i]\)
  
  *Vector x vector:* \(\text{acc}[i] += v_0[i] \times v_1[i] + v_2[i] \times v_3[i]\)

- Quad 8x8 MAC operations
  
  *Scalar x vector:* \(\text{acc}[i] += c_0 \times v_0[i] + c_1 \times v_1[i] + c_2 \times v_2[i] + c_3 \times v_3[i]\)

- Variants support vectorizing in different dimensions
  - Different networks and layers within a given network often need different approaches for best performance
CNN Performance

• 3-D convolution kernel
  – Input: 14x14x64 (8-bit); conv: 5x5x64, stride 1; output: 10x10x64 (8-bit)

<table>
<thead>
<tr>
<th></th>
<th>Multiplier Utilization</th>
<th>Relative Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vision P5</td>
<td>95%</td>
<td>1.0</td>
</tr>
<tr>
<td>Vision P6</td>
<td>80%</td>
<td>3.3</td>
</tr>
</tbody>
</table>

• Alexnet layer 1 convolution
  – Input: 227x227x3 (8-bit); conv: 11x11x3, stride 4; output: 55x55x96 (8-bit)

<table>
<thead>
<tr>
<th></th>
<th>Multiplier Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vision P6</td>
<td>57%</td>
</tr>
</tbody>
</table>
Vision P6 Demo
Traffic Sign Recognition: CNN

Tensilica® Vision DSP Running Convolutional Neural Network (CNN) Prediction Algorithm
Compare the input signs with 43 trained signs
People Detection and Face Detection

People detection: An example of computer vision histogram of oriented gradient algorithm

- 64x128 detection window, 1.2 scale factor, L1 Histogram normalization
- **Market**: Automotive, drone, and robot
  Applications: Collision avoidance, object detection, passenger detection
- **Market**: Surveillance
  Applications: People counting, people detection/tracking

Face detection: An example of computer vision OpenCV, Haar cascade algorithm

- Full Detection every frame
- 22 levels
- **Market**: Mobile, automotive, wearable, tablets
  Applications: Selective auto exposure, white balance, focus, face tracking, face authentication