CS 152 Computer Architecture and Engineering
CS252 Graduate Computer Architecture

Lecture 23  Domain-Specific Architectures

Krste Asanovic
Electrical Engineering and Computer Sciences
University of California at Berkeley

http://www.eecs.berkeley.edu/~krste
http://inst.eecs.berkeley.edu/~cs152
Last Time in Lecture 22

- Multiple types of virtual machine
  - user level
  - system level
- User virtual machine is ISA plus environment
- System virtual machine provides environment for an OS
- ISA translation
  - running one ISA on another
  - implementing portions of an ISA in software
- System virtual machine monitors or hypervisors
  - manage multiple OS on same machine
Moore’s Law

“Cramming more components onto integrated circuits”, Gordon E. Moore, Electronics, 1965
Figure 7.1 Time before new Intel semiconductor process technology measured in nm. The y-axis is log scale. Note that the time stretched previously from about 24 months per new process step to about 30 months since 2010.
End of Dennard (Voltage) Scaling

Why did we hit a power/cooling wall?

Dennard Scaling

Post-Dennard Scaling

Data courtesy S. Borkar/Intel 2011
Single-Thread Processor Performance

40 years of Processor Performance

[ Hennessy & Patterson, 2017 ]
Domain-Specific Architectures

In early 2000s, microprocessors moved to manycore architecture (multiple general-purpose cores per die) to improve energy efficiency for parallel workloads.

Now, great interest in more specialized architectures to further improve energy efficiency on certain workloads

Four examples:

Google TPU
Microsoft Catapult
Intel Crest (now Nervana)
Google Pixel Visual Core
Guidelines for DSAs

- Use dedicated memories to minimize data movement
- Invest resources into more arithmetic units or bigger memories
- Use the easiest form of parallelism that matches the domain
- Reduce data size and type to the simplest needed for the domain
- Use a domain-specific programming language
# Guidelines for DSAs

<table>
<thead>
<tr>
<th>Guideline</th>
<th>TPU</th>
<th>Catapult</th>
<th>Crest</th>
<th>Pixel Visual Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design target</td>
<td>Data center ASIC</td>
<td>Data center FPGA</td>
<td>Data center ASIC</td>
<td>PMD ASIC/SOC IP</td>
</tr>
<tr>
<td>1. Dedicated memories</td>
<td>24 MiB Unified Buffer, 4 MiB Accumulators</td>
<td>Varies</td>
<td>N.A.</td>
<td>Per core: 128 KiB line buffer, 64 KiB P.E. memory</td>
</tr>
<tr>
<td>2. Larger arithmetic unit</td>
<td>65,536 Multiply-accumulators</td>
<td>Varies</td>
<td>N.A.</td>
<td>Per core: 256 Multiply-accumulators (512 ALUs)</td>
</tr>
<tr>
<td>3. Easy parallelism</td>
<td>Single-threaded, SIMD, in-order</td>
<td>SIMD, MISD</td>
<td>N.A.</td>
<td>MPMD, SIMD, VLIW</td>
</tr>
<tr>
<td>4. Smaller data size</td>
<td>8-Bit, 16-bit integer</td>
<td>8-Bit, 16-bit integer</td>
<td>21-bit Fl. Pt.</td>
<td>8-bit, 16-bit, 32-bit integer</td>
</tr>
</tbody>
</table>
Example: Deep Neural Networks

- Inspired by neuron of the brain
- Computes non-linear “activation” function of the weighted sum of input values
- Neurons arranged in layers

<table>
<thead>
<tr>
<th>Name</th>
<th>DNN layers</th>
<th>Weights</th>
<th>Operations/Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP0</td>
<td>5</td>
<td>20M</td>
<td>200</td>
</tr>
<tr>
<td>MLP1</td>
<td>4</td>
<td>5M</td>
<td>168</td>
</tr>
<tr>
<td>LSTM0</td>
<td>58</td>
<td>52M</td>
<td>64</td>
</tr>
<tr>
<td>LSTM1</td>
<td>56</td>
<td>34M</td>
<td>96</td>
</tr>
<tr>
<td>CNN0</td>
<td>16</td>
<td>8M</td>
<td>2888</td>
</tr>
<tr>
<td>CNN1</td>
<td>89</td>
<td>100M</td>
<td>1750</td>
</tr>
</tbody>
</table>
Example: Deep Neural Networks

- Most practitioners will use existing design
  - Topology
  - Data type

- Training (learning):
  - Calculate weights using backpropagation algorithm
  - Supervised learning: stochastic gradient descent

<table>
<thead>
<tr>
<th>Type of data</th>
<th>Problem area</th>
<th>Size of benchmark’s training set</th>
<th>DNN architecture</th>
<th>Hardware</th>
<th>Training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>text [1]</td>
<td>Word prediction (word2vec)</td>
<td>100 billion words (Wikipedia)</td>
<td>2-layer skip gram</td>
<td>1 NVIDIA Titan X GPU</td>
<td>6.2 hours</td>
</tr>
<tr>
<td>audio [2]</td>
<td>Speech recognition</td>
<td>2000 hours (Fisher Corpus)</td>
<td>11-layer RNN</td>
<td>1 NVIDIA K1200 GPU</td>
<td>3.5 days</td>
</tr>
<tr>
<td>images [3]</td>
<td>Image classification</td>
<td>1 million images (ImageNet)</td>
<td>22-layer CNN</td>
<td>1 NVIDIA K20 GPU</td>
<td>3 weeks</td>
</tr>
<tr>
<td>video [4]</td>
<td>activity recognition</td>
<td>1 million videos (Sports-1M)</td>
<td>8-layer CNN</td>
<td>10 NVIDIA GPUs</td>
<td>1 month</td>
</tr>
</tbody>
</table>

- Inference: use neural network for classification
Multi-Layer Perceptrons

Parameters:
- $\text{Dim}[i]$: number of neurons
- $\text{Dim}[i-1]$: dimension of input vector
- Number of weights: $\text{Dim}[i-1] \times \text{Dim}[i]$
- Operations: $2 \times \text{Dim}[i-1] \times \text{Dim}[i]$
- Operations/weight: 2

Example: Deep Neural Networks
Convolutional Neural Network

- Computer vision
- Each layer raises the level of abstraction
  - First layer recognizes horizontal and vertical lines
  - Second layer recognizes corners
  - Third layer recognizes shapes
  - Fourth layer recognizes features, such as ears of a dog
  - Higher layers recognizes different breeds of dogs
Convolutional Neural Network

- Parameters:
  - DimFM[i-1]: Dimension of the (square) input Feature Map
  - DimFM[i]: Dimension of the (square) output Feature Map
  - DimSten[i]: Dimension of the (square) stencil
  - NumFM[i-1]: Number of input Feature Maps
  - NumFM[i]: Number of output Feature Maps
  - Number of neurons: NumFM[i] x DimFM[i]^2
  - Number of weights per output Feature Map: NumFM[i-1] x DimSten[i]^2
  - Total number of weights per layer: NumFM[i] x Number of weights per output Feature Map
  - Number of operations per output Feature Map: 2 x DimFM[i]^2 x Number of weights per output Feature Map
  - Total number of operations per layer: NumFM[i] x Number of operations per output Feature Map = 2 x DimFM[i]^2 x NumFM[i] x Number of weights per output Feature Map = 2 x DimFM[i]^2 x Total number of weights per layer
  - Operations/Weight: 2 x DimFM[i]^2

Example: Deep Neural Networks
### Recurrent Neural Network

- Speech recognition and language translation
- Long short-term memory (LSTM) network

---

Example: Deep Neural Networks

---

Copyright © 2019, Elsevier Inc. All rights Reserved
Recurrent Neural Network

- Parameters:
  - Number of weights per cell: 
    \[3 \times (3 \times \text{Dim} \times \text{Dim}) + (2 \times \text{Dim} \times \text{Dim}) + (1 \times \text{Dim} \times \text{Dim}) = 12 \times \text{Dim}^2\]
  - Number of operations for the 5 vector-matrix multiplies per cell: 2 x Number of weights per cell = 24 x Dim^2
  - Number of operations for the 3 element-wise multiplies and 1 addition (vectors are all the size of the output): 4 x Dim
  - Total number of operations per cell (5 vector-matrix multiplies and the 4 element-wise operations): 24 x Dim^2 + 4 x Dim
  - Operations/Weight: \(~2\)
Convolutional Neural Network

- **Batches:**
  - Reuse weights once fetched from memory across multiple inputs
  - Increases operational intensity

- **Quantization**
  - Use 8- or 16-bit fixed point

- **Summary:**
  - Need the following kernels:
    - Matrix-vector multiply
    - Matrix-matrix multiply
    - Stencil
    - ReLU (Rectified Linear Unit = max(0,x))
    - Sigmoid
    - Hyperbolic tangent
Tensor Processing Unit

- Google’s DNN ASIC
- 256 x 256 8-bit matrix-multiply unit
- Large software-managed scratchpad
- Coprocessor on the PCIe bus
Tensor Processing Unit
TPU ISA

- **Read_Host_Memory**
  - Reads memory from the CPU memory into the unified buffer

- **Read_Weights**
  - Reads weights from the Weight Memory into the Weight FIFO as input to the Matrix Unit

- **MatrixMatrixMultiply/Convolve**
  - Perform a matrix-matrix multiply, a vector-matrix multiply, an element-wise matrix multiply, an element-wise vector multiply, or a convolution from the Unified Buffer into the accumulators
  - takes a variable-sized B*256 input, multiplies it by a 256x256 constant input, and produces a B*256 output, taking B pipelined cycles to complete

- **Activate**
  - Computes activation function

- **Write_Host_Memory**
  - Writes data from unified buffer into host memory

Copyright © 2019, Elsevier Inc. All rights Reserved
## TPU Implementation

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Unified Buffer for activations (96Kx256x8b = 24 MiB)</td>
<td>29% of chip</td>
</tr>
<tr>
<td>Matrix multiply unit (256x256x8b = 64K MAC)</td>
<td>24%</td>
</tr>
<tr>
<td>Host Interf.</td>
<td>2%</td>
</tr>
<tr>
<td>Accumulators (4Kx256x32b = 4 MiB)</td>
<td>6%</td>
</tr>
<tr>
<td>Control</td>
<td>2%</td>
</tr>
<tr>
<td>Activation pipeline</td>
<td>6%</td>
</tr>
<tr>
<td>PCIe Interface</td>
<td>3%</td>
</tr>
<tr>
<td>Misc. I/O</td>
<td>1%</td>
</tr>
</tbody>
</table>

![TPU diagram]

Copyright © 2019, Elsevier Inc. All rights Reserved
TPU Operation
The TPU and the Guidelines

- Use dedicated memories
  - 24 MiB dedicated buffer, 4 MiB accumulator buffers

- Invest resources in arithmetic units and dedicated memories
  - 60% of the memory and 250X the arithmetic units of a server-class CPU

- Use the easiest form of parallelism that matches the domain
  - Exploits 2D SIMD parallelism

- Reduce the data size and type needed for the domain
  - Primarily uses 8-bit integers

- Use a domain-specific programming language
  - Uses TensorFlow
**Microsoft Catapult**

- Needed to be general-purpose and power-efficient
  - Uses FPGA PCIe board with dedicated 20 Gbps network in 6 x 8 torus
  - Each of the 48 servers in half the rack has a Catapult board
  - Limited to 25 watts
  - 32 MiB Flash memory
  - Two banks of DDR3-1600 (11 GB/s) and 8 GiB DRAM
  - FPGA (unconfigured) has 3962 18-bit ALUs and 5 MiB of on-chip memory
  - Programmed in Verilog RTL
  - Shell is 23% of the FPGA
Microsoft Catapult: CNN

- CNN accelerator, mapped across multiple FPGAs
Microsoft Catapult: CNN
Microsoft Catapult: Search Ranking

- Feature extraction (1 FPGA)
  - Extracts 4500 features for every document-query pair, e.g. frequency in which the query appears in the page
  - Systolic array of FSMs
- Free-form expressions (2 FPGAs)
  - Calculates feature combinations
- Machine-learned Scoring (1 FPGA for compression, 3 FPGAs calculate score)
  - Uses results of previous two stages to calculate floating-point score
- One FPGA allocated as a hot-spare
Microsoft Catapult: Search Ranking

- Version 2 of Catapult
  - Placed the FPGA between the CPU and NIC
  - Increased network from 10 Gb/s to 40 Gb/s
  - Also performs network acceleration
  - Shell now consumes 44% of the FPGA
  - Now FPGA performs only feature extraction
Catapult and the Guidelines

- Use dedicated memories
  - 5 MiB dedicated memory
- Invest resources in arithmetic units and dedicated memories
  - 3926 ALUs
- Use the easiest form of parallelism that matches the domain
  - 2D SIMD for CNN, MISD parallelism for search scoring
- Reduce the data size and type needed for the domain
  - Uses mixture of 8-bit integers and 64-bit floating-point
- Use a domain-specific programming language
  - Uses Verilog RTL; Microsoft did not follow this guideline
CS152 Administrivia

- Wednesday April 25, lecture time will be review session
- Friday April 27, discussion will cover caches and review
- Final exam, Tuesday May 8, 306 Soda, 11:30am-2:30pm
CS252 Administrivia

- Monday April 23rd, last Project Checkpoint, 4-5pm, 405 Soda
  - Prepare 10-minute presentation on current status
- Project presentations, May 2nd, 2:30pm-5pm, 511 Soda
Intel Crest (now “Nervana”)  
- DNN training  
- 16-bit fixed point  
- Operates on blocks of 32x32 matrices  
- SRAM + HBM2
Google Pixel Visual Core

- Pixel Visual Core
  - Image Processing Unit
  - Performs stencil operations
  - Descended from Image Signal processor
Pixel Visual Core

- Software written in Halide, a DSL
  - Compiled to virtual ISA
  - vISA is lowered to physical ISA using application-specific parameters
  - pISA is VLSI

- Optimized for energy
  - Power Budget is 6 to 8 W for bursts of 10-20 seconds, dropping to tens of milliwatts when not in use
  - 8-bit DRAM access equivalent energy as 12,500 8-bit integer operations or 7 to 100 8-bit SRAM accesses
  - IEEE-754 floating-point operations require 22X to 150X of the cost of 8-bit integer operations

- Optimized for 2D access
  - 2D SIMD unit
  - On-chip SRAM structured using a square geometry
Figure 7.38 Floor plan of the 8-core Pixel Visual Core chip. A53 is an ARMv7 core. LPDDR4 is a DRAM controller. PCIE and MIPI are I/O buses.

© 2019 Elsevier Inc. All rights reserved.
Pixel Core VLIW Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Scalar</th>
<th>Math</th>
<th>Memory</th>
<th>Imm</th>
<th>MemImm</th>
</tr>
</thead>
<tbody>
<tr>
<td># Bits</td>
<td>43</td>
<td>38</td>
<td>12</td>
<td>16</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 7.35 VLIW format of the 119-bit pISA instruction.
Pixel Visual Core
Pixel Visual Core

Copyright © 2019, Elsevier Inc. All rights Reserved
Pixel Visual Core
Visual Core and the Guidelines

- Use dedicated memories
  - 128 + 64 MiB dedicated memory per core
- Invest resources in arithmetic units and dedicated memories
  - 16x16 2D array of processing elements per core and 2D shifting network per core
- Use the easiest form of parallelism that matches the domain
  - 2D SIMD and VLIW
- Reduce the data size and type needed for the domain
  - Uses mixture of 8-bit and 16-bit integers
- Use a domain-specific programming language
  - Halide for image processing and TensorFlow for CNNs
Computer Architecture in 2018

- Explosion of interest in custom architectures due to end of transistor scaling
  - Full employment for computer architects!
- But need to learn about application domains
  - Cannot just work with precompiled binaries anymore!

- Get involved in research projects,
  - ADEPT – microprocessor architecture and chip design
  - RISE – machine learning, datacenter software, and security
- Undergrad research experience is the most important part of application to top grad schools!
End of CS152/CS252

- Thanks for taking part in first offering of redesigned course structure!
- Welcome feedback on course eval, or via email