Accelerators-I

Tushar Krishna
Associate Professor @ Georgia Tech
Visiting Professor @ MIT EECS and CSAIL
Outline

• Why do we need accelerators?
• Why now?
• How to design accelerators
Outline

• Why do we need accelerators?
• Why now?
• How to design accelerators
Power Constraints in Modern Computers

<< 1W  ~ 1W  ~ 15W  ~ 50W  ~ 100W  ~ 100W
Energy and Power Consumption

• Energy Consumption = $\alpha \times C \times V^2$

- Switching activity factor (between 0 to 1)
- Capacitance
- Voltage

• Power Consumption = $\alpha \times C \times V^2 \times f$

- Frequency
# CMOS Scaling (idealized)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Gen X</th>
<th>Gen X+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate Width (Moore’s Law)</td>
<td>1.0</td>
<td>0.7</td>
</tr>
<tr>
<td>Device Area/Capacitance</td>
<td>1.0</td>
<td>0.5</td>
</tr>
<tr>
<td>Voltage (Dennard’s)</td>
<td>1.0</td>
<td>0.7</td>
</tr>
<tr>
<td>Energy</td>
<td>$\sim 1.0 \times 1.0^2 = 1.0$</td>
<td>$\sim 2 \times 0.7 \times 0.7^2 = 0.65$</td>
</tr>
<tr>
<td>Delay</td>
<td>1.0</td>
<td>0.7</td>
</tr>
<tr>
<td>Frequency</td>
<td>$1/1.0 = 1.0$</td>
<td>$1/0.7 = 1.4$</td>
</tr>
<tr>
<td>Power</td>
<td>$\sim 1.0 \times 1.0^2 \times 1.0 = 1.0$</td>
<td>$\sim 2 \times 0.7 \times 0.7^2 \times 1.4 = 1.0$</td>
</tr>
</tbody>
</table>

[ Dennard et al., "Design of ion-implanted MOSFET's with very small physical dimensions“, JSSC 1974 ]
CMOS Scaling (idealized)

- Moore’s Law (transistors) + Dennard’s Scaling (voltage)
  - 2.8X in chip capability per generation at constant power
  - ~5000X performance improvement in 20 years

Source: “Advancing Computer Systems Without Technology Progress, ISAT Outbrief, 2012
“Power Wall”

- Dennard’s Scaling has stopped
  - Why? Already operating close to $V_{\text{threshold}}$
  - Cannot increase operating frequency
    \[ P = \frac{1}{2} CV^2 \]

Source: “Advancing Computer Systems Without Technology Progress, ISAT Outbrief, 2012"
Technology Trends

First inflection: Multicore

48 Years of Microprocessor Trend Data

- Transistors (thousands)
- Single-Thread Performance (SpecINT x 10^3)
- Frequency (MHz)
- Typical Power (Watts)
- Number of Logical Cores

Year

Utilization Wall ("Dark Silicon")

Source: M. Taylor, "Is Dark Silicon Useful? Harnessing the Four Horsemen of the Coming Dark Silicon Apocalypse", DAC 2012
Power Management options?

- **Tackle power across all levels of the computing stack**
  - **Technology**
    - Cost of switching
  - **Circuits**
    - High-speed vs low-power implementation
    - Clock gating and power gating support
    - DVFS
  - **Microarchitecture and ISA**
    - Simplify design
    - Parallelism
    - Heterogeneity and Specialization
  - **Compiler**
    - Instruction footprint and Cache behavior
  - **OS**
    - Tune DVFS and power states
  - **Algorithm:**
    - Switching activity

<table>
<thead>
<tr>
<th>Application / Algorithm</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compiler</td>
</tr>
<tr>
<td></td>
<td>ISA</td>
</tr>
<tr>
<td></td>
<td>Microarchitecture</td>
</tr>
<tr>
<td></td>
<td>Circuits</td>
</tr>
<tr>
<td></td>
<td>Technology (Devices)</td>
</tr>
</tbody>
</table>
Power Management options?

- **Technology**
  - Cost of switching

- **Circuits**
  - High-speed vs low-power implementation
  - Clock gating and power gating support
  - DVFS

- **Microarchitecture and ISA**
  - Simplify design
  - Parallelism
  - Heterogeneity and Specialization

- **Compiler**
  - Instruction footprint and Cache behavior

- **OS**
  - Tune DVFS and power states

- **Algorithm:**
  - Switching activity

---

<table>
<thead>
<tr>
<th>Application / Algorithm</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compiler</td>
</tr>
<tr>
<td></td>
<td>ISA</td>
</tr>
<tr>
<td></td>
<td>Microarchitecture</td>
</tr>
<tr>
<td></td>
<td>Circuits</td>
</tr>
<tr>
<td></td>
<td>Technology (Devices)</td>
</tr>
</tbody>
</table>
Heterogeneity and Specialization

**Chip type:**
- Microprocessor
- Microprocessor + GPU
- General purpose DSP
- Dedicated design

**Graph:**
- Energy Efficiency (MOPS/mW)
- Chip types: CPUs, CPUs+GPU, GP DSPs, Dedicated design

**Improvements:**
- CPUs: ~1000x

**Why?**

*Improve Energy Efficiency via Customization!*
A modern CPU

Instruction Fetch, Decode, Scheduling and Speculation for “programmability”

Actual Computation

Implicit data management via caches

AMD Zen (2016)
Quantifying this overhead

Embedded Processor Energy Breakdown

- Arithmetic: 70%
- Clock and control: 24%
- Data supply: 28%
- Instruction supply: 42%

Source: Dally et al. Efficient Embedded Computing, IEEE’08
# Performance/Area Benefits

<table>
<thead>
<tr>
<th></th>
<th>GFLOP/s</th>
<th>(GFLOP/s)/mm²</th>
<th>GFLOP/J</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FFT-210</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Core i7 (45nm)</td>
<td>67</td>
<td>0.35</td>
<td>0.71</td>
</tr>
<tr>
<td>Nvidia GTX285 (55nm)</td>
<td>250</td>
<td>1.41</td>
<td>4.2</td>
</tr>
<tr>
<td>Nvidia GTX480 (40nm)</td>
<td>453</td>
<td>1.08</td>
<td>4.3</td>
</tr>
<tr>
<td>ATI R5870 (40nm)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Xilinx V6-LX760 (40nm)</td>
<td>380</td>
<td>0.99</td>
<td>6.5</td>
</tr>
<tr>
<td>same RTL std cell (65nm)</td>
<td>952</td>
<td>239</td>
<td>90</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Mopts/s</th>
<th>(Mopts/s)/mm²</th>
<th>Mopts/J</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Black-Scholes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Core i7 (45nm)</td>
<td>487</td>
<td>2.52</td>
<td>4.88</td>
</tr>
<tr>
<td>Nvidia GTX285 (55nm)</td>
<td>10756</td>
<td>60.72</td>
<td>189</td>
</tr>
<tr>
<td>Nvidia GTX480 (40nm)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ATI R5870 (40nm)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Xilinx V6-LX760 (40nm)</td>
<td>7800</td>
<td>20.26</td>
<td>138</td>
</tr>
<tr>
<td>same RTL std cell (65nm)</td>
<td>25532</td>
<td>1719</td>
<td>642.5</td>
</tr>
</tbody>
</table>

Source: Chung et al., "Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs?", MICRO 2010
Outline

• Why do we need accelerators?
• Why now?
• How to design accelerators
Domain-specific Accelerators in SoCs
"AI is the new electricity" – Andrew Ng

Object Detection

Image Segmentation

Medical Imaging

Speech Recognition

Text to Speech

Recommendations

Games
AI Compute Demands Growing Exponentially

AlexNet to AlphaGo Zero: A 300,000x Increase in Compute

Deep and steep
Computing power used in training AI systems
Days spent calculating at one petaflop per second*, log scale

By fundamentals
- Language
- Speech
- Vision
- Games
- Other

---

Source: OpenAI

The Economist

Google’s dedicated TensorFlow processor, or TPU, crushes Intel, Nvidia in inference workloads

By Joel Hruska on April 6, 2017 at 9:48 am | 25 Comments

How Amazon is racing to catch Microsoft and Google in generative A.I. with custom AWS chips

Published Sat, Aug 12 2023 · 9:00 AM EDT | Updated Mon, Aug 21 2023 · 7:40 PM EDT

Meta announces AI training and inference chip project

By Katie Paul and Stephen Nellis
May 18, 2023 · 4:34 PM EDT · Updated 7 months ago

Microsoft announces custom AI chip that could compete with Nvidia

Published Wed, Nov 15 2023 · 11:00 AM EST | Updated Wed, Nov 15 2023 · 3:05 PM EST
HW Beyond Cloud Computing

**WIRED**

Musk Says Tesla Is Building Its Own Chip for Autopilot

**ars TECHNICA**

Surprise! The Pixel 2 is hiding a custom Google SoC for image processing

Google's 8-core Image Processing Unit will be enabled with Android 8.1.

---

Elon Musk disclosed plans for Tesla to design its own chip to power its self-driving function.

Elon Musk

[Also Nvidia, Intel, Qualcomm...]

December 4, 2023
AI Chips ecosystem

AI Chip Landscape

Tech Giants/System
- Google
- Microsoft
- Facebook
- AWS
- Apple
- IBM
- Alibaba Group
- Huawei
- Samsung
- Intel
- NVIDIA
- AMD
- Xilinx
- ARM
- Unisoc
- MediaTek
- Rockchip
- Broadcom
- Samsung
- Western Digital
- NOKIA
- LG

IC Vender/Fabless
- Global Foundries
- TSMC
- UMC
- SMIC
- AMS
- Triquint
- Xilinx
- Marvell
- Broadcom
- NVIDIA
- Intel
- AMD
- ARM
- Samsung
- TSMC
- UMC
- SMIC
- AMS
- Triquint
- Xilinx
- Marvell
- Broadcom
- NVIDIA
- Intel
- AMD
- ARM
- Samsung

Startup in China
- Cambricon
- Graphcore
- SambaNova
- Nanoflask
- Hailo
- AIx
- Tinki
- KALRAY
- Kogas
- Tachyum
- Esperanto
- PEZY Computing
- Ela Compute
- Greenwaves

Startup Worldwide
- FPGA
- Achiron
- fableg
- Processing in Memory
- MYTHIC
- Syntiant
- gTMR
- Lightmatter
- OpenPitaya
- Neuromorphic
- Hailo
- Brainchip

IP/Design Service
- ARM
- Synopsys
- Iagination
- CEVA
- Cadenza
- SiFive
- ArterisIP
- Root electronics
- Broadcom
- GUC
- Synplicity
- PARADAY
- Silicon

Automated Driving
- Mobileye
- ZF
- Bosch
- NVIDIA
- NVIDIA

Smart Voice
- NXP
- Rockchip
- Baidu
- Huawei

Automated Driving
- Tesla
- NVIDIA
- NVIDIA

Compilers
- TensorFlow
- Glow
- LLVM
- NVIDIA TensorRT

benchmarks
- MLPerf
- AI-Benchmark
- AI Matrix

Grants
- Amazon Alexa
- Baidu
- DALL-E
- DAWNbench
Opportunities

From EE Times – September 27, 2016

“Today the job of training machine learning models is limited by compute, if we had faster processors we’d run bigger models... in practice we train on a reasonable subset of data that can finish in a matter of months. We could use improvements of several orders of magnitude – 100x or greater.”

– Greg Diamos, Senior Researcher, SVAIL, Baidu

ACM’s Celebration of 50 Years of the ACM Turing Award (June 2017)

“Compute has been the oxygen of deep learning”

– Ilya Sutskever, Research Director of Open AI
Demand for Computer Architects

Meta is seeking an ASIC Engineer, Architecture to join our infrastructure organization. Our servers and data centers are the foundation upon which our rapidly scaling infrastructure efficiently operates and upon which our innovative services are delivered. By holding this role, you will be an integral member of an ASIC team to build accelerators for some of our top workloads enabling our data centers to scale efficiently. You will have an opportunity to work with AI/Machine Learning (ML) and video codec experts in the company, help architect state-of-the-art machine learning accelerators, and video transcoders and contribute to modeling these accelerators. Come work and learn alongside our expert engineers to build “Green” data center accelerators.

ASIC Engineer, Architecture
Sunnyvale, CA | +3 more

ASID Design Engineer, Machine Learning Accelerator Cores
Google
Sunnyvale, CA

Full-time

Job highlights
Identified by Google from the original job post

Qualifications
• Bachelor’s degree in Electrical Engineering, Computer Science, or equivalent practical experience
• 2 years of industrial experience
• Experience in logic design and functional and Power Performance Area (PPA) closure
• Experience applying engineering practices (e.g., code review, testing, refactoring)

Responsibilities
• As an ASIC Design Engineer, you will be a part of a team developing ways to accelerate computation in data centers
• You’ll have dynamic, multi-faceted responsibilities in areas such as project definition, design, and implementation
• You will participate in the architecture, documentation, and implementation of data center accelerators
• Define architecture and micro-architecture specifications

Embedded Software Development Engineer, Machine Learning Accelerators
Amazon
Boston, MA

Full-time

Job highlights
Identified by Google from the original job post

Qualifications
• Note that prior Machine Learning knowledge or experience is required for this role
• B.S. in Computer Science, Electrical Engineering, or related technical field
• C or C++ project experience, with strong object-oriented programming background
• Experience with computer architecture
• Experience with firmware development

Responsibilities
• Software / Hardware architecture and co-design
• Embedded software development, testing, and debug
• Test suite and infrastructure development
• Developing software which can be maintained, improved upon, documented, tested, and reused
• Close collaboration with RTL designers, design verification engineers, and other software teams

HW Development Manager, FPGA and ASIC IP design – CSI / Azure – Cloud Server Infrastructure
Microsoft
Bellevue, WA, US

Microsoft is seeking a highly motivated, FPGA and ASIC IP design engineering manager to build innovative FPGA-based computing systems within a large design team. The group will ... careers.microsoft.com

13 connections work here
1 month ago

Physical Design Engineer
Microsoft
Redmond, WA, US

1-2 years of experience in ASIC physical design flows and methodologies. Job responsibilities will entail taking RTL logic through a full ASIC design flow. Worked with toolsets ... careers.microsoft.com

13 connections work here
2 weeks ago
Why do we need DNN accelerators?

- **Millions of Parameters (i.e., weights)**
  - Billions of computations ➔ **Need lots of parallel compute**
  
<table>
<thead>
<tr>
<th>DNN Topology</th>
<th>Number of Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet (2012)</td>
<td>3.98M</td>
</tr>
<tr>
<td>VGGnet-16 (2014)</td>
<td>28.25M</td>
</tr>
<tr>
<td>GoogleNet (2015)</td>
<td>6.77M</td>
</tr>
<tr>
<td>Resnet-50 (2016)</td>
<td>23M</td>
</tr>
<tr>
<td>DLRM (2019)</td>
<td>540M</td>
</tr>
<tr>
<td>Megatron (2019)</td>
<td>8.3B</td>
</tr>
</tbody>
</table>

- Heavy data movement ➔ **Need to reduce energy**

  ![Diagram of data movement and energy cost]

  - This makes CPUs inefficient
  - This makes GPUs inefficient
Outline

- Why do we need accelerators?
- Why now?
- How to design accelerators
HW-SW Co-Design

- Target Domain (e.g., AI)
- Key Computation Kernels
- Compute and Memory Behavior
- Hardware Structures
- Design-space Exploration

Constraints (e.g., Area and Power)

Accelerator
Understanding Deep Neural Networks

Low Level Features

High Level Features

Input: Image

Modified Image Source: [Lee, CACM 2011]

Output: “Volvo XC90”
Convolution Operation

Input Image

Filter

Output Image

R

H

E
Convolution Operation

The convolution operation involves an input image, a filter, and an output image. The filter is applied to each region of the input image to produce a partial sum, which is then accumulated to form the output image. The operations are illustrated with dot product and partial sum accumulation.
Convolution Operation

Many Input Channels

Input Image

Output Image
Convolution Operation

Many Filters

Input Image

Output Image

Many Output Channels
Convolution Operation

Many

Input Images

Many

Output Images

Filters

Many

Input Images

Many

Output Images

Many

Output Images

Many

Output Images

Many

Output Images

Many

Output Images

Many

Output Images
Representation: Tensors

- Rank-0 - Scalar
- Rank-1 - Vector
- Rank-2 - Matrix
- Rank-3 - Cube
Example: Input Act/Fmap Tensor

The compiler “lowers” high-rank tensors into appropriate HW structures

I[C][H][W]
What to accelerate?

Transformer (Language Understanding)

GNMT (Machine Translation)

Runtime breakdown on V100 GPU

Matrix multiplications (GEMMs) consume around 70% of the total runtime on modern deep learning workloads.

Prime candidate for acceleration

Possible Pitfall?
Beware of Amdahl’s Law
GEMMs in Deep Learning

**Forward Pass**

1. Input Activation
2. Model Weight
3. Output Activation

   (Input Activation of Layer \( i + 1 \))

**Backward Pass**

1. Input Activation (Transpose)
2. Input Gradient of Layer \( i + 1 \)
3. Weight Gradient of Layer \( i \)

1. Input Gradient of Layer \( i + 1 \) (Transpose)
2. Model Weight
3. Input Gradient of Layer \( i \)
Spatial (or Dataflow) Accelerators

• **Millions of Parameters (i.e., weights)**
  - Billions of computations

- Spread computations across thousands of PEs (i.e., **parallelism**)
- **Heavy data movement**

**Features**
- Thousands of Processing Elements (PEs)
- Custom Memory Hierarchy (typically scratchpads, no caches)
- Custom NoCs

**Reuse** data within the array via dataflow.
What is Reuse?

Example Operations

\[
\begin{align*}
\text{C1} &= A1 \times B1 A2 \times B5 A3 \times B9 A4 \times B13 \\
\text{C5} &= A5 \times B1 A6 \times B5 A7 \times B9 A8 \times B13
\end{align*}
\]

\[
\begin{align*}
\text{C1} &= A1 \times B1 A2 \times B5 A3 \times B9 A4 \times B13 \\
\text{C11} &= A9 \times B3 A10 \times B7 A11 \times B11 A12 \times B15
\end{align*}
\]
Examples of Data Reuse in DNN

**Convolutional Reuse**

CONV layers only (sliding window)

Reuse:
- Activations
- Filter weights
Examples of Data Reuse in DNN

**Convolutional Reuse**

CONV layers only (sliding window)

- **Reuse:**
  - Activations
  - Filter weights

**Fmap Reuse**

CONV and FC layers

- **Reuse:**
  - Activations

Diagram:
- Filters
- Input Fmap
- Reuse arrows connecting different layers.
Examples of Data Reuse in DNN

### Convolutional Reuse

CONV layers only (sliding window)

- **Filter Reuse**
  - Activations
  - Filter weights

- **Fmap Reuse**
  - CONV and FC layers

- **Filter Reuse**
  - CONV and FC layers (batch size > 1)

---

December 4, 2023

MIT 6.5900 (ne 6.823) Fall 2023

L24-43
Why does reuse help?

Suppose
- 100 PEs operating at 1 GHz => 100 GFLOPs/sec
- Each PE needs 2 bytes of read and 1 byte of write.
- DRAM BW ~25 GBps

Simple Accelerator:
BW requirement: 3 bytes/cycle => 300 GBps
--> Mem Bound!

Suppose we have data reuse
- weight reused completely
- input reused for 10 cycles
- psums reused for 10 cycles

BW requirement: 20 GBps
→ Compute Bound

Realistic?
How to exploit Reuse?

GEMM

Weights reused across multiple input activations (called “weight stationary” mapping)

What HW structures would you need?
Building a DNN Accelerator

<table>
<thead>
<tr>
<th>Memory Read</th>
<th>MAC*</th>
<th>Memory Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter</td>
<td>ALU</td>
<td>updated partial sum</td>
</tr>
<tr>
<td>weight</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fmap activation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>partial sum</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* multiply-and-accumulate
Building a DNN Accelerator

- Memory Read
- MAC*: multiply-and-accumulate
- Memory Write

Worst Case: all memory R/W are DRAM accesses

- Example: AlexNet [NeurIPS 2012] has 724M MACs
  → 2896M DRAM accesses required
Building a DNN Accelerator

Extra levels of local memory hierarchy
Building a DNN Accelerator

Opportunities: **data reuse**

Extra levels of local memory hierarchy

Diagram showing Memory Read, MAC, and Memory Write processes with data flows and the concept of data reuse.
Building a DNN Accelerator

**Opportunities:**

1. **data reuse**

   - Can reduce DRAM reads of `filter/fmap` by up to **500×**

   **-- AlexNet CONV layers**
Building a DNN Accelerator

Opportunities:
1. **data reuse**
2. **local accumulation**

1. Can reduce DRAM reads of filter/fmap by up to **500×**
2. Partial sum accumulation does **NOT** have to access DRAM
Building a DNN Accelerator

Opportunities:

1. **data reuse**
   - Can reduce DRAM reads of filter/fmap by up to $500 \times$

2. **local accumulation**
   - Partial sum accumulation does **NOT** have to access DRAM

- Example: DRAM access in AlexNet can be reduced from $2896\text{M}$ to $61\text{M}$ (best case)
Building a DNN Accelerator

Leverage Parallelism for Throughput!
Building a DNN Accelerator

Leverage Parallelism for *spatial* data reuse!
Spatial DNN Accelerator

Local Memory Hierarchy
- Global Buffer
- Direct inter-PE network
- PE-local memory (RF)

Processing Element (PE)
- Reg File: 0.5 – 1.0 kB
- Control
Hardware structures to exploit reuse

Temporal Reuse

Spatial Reuse

Spatio-Temporal Reuse

Memory Hierarchy / Staging Buffers

Multicasting-support NoCs

Neighbor-to-Neighbor Connections

The availability of the hardware structure limits the “mapping-space”

E.g., Custom memory hierarchies in accelerators.
E.g., Hierarchical Bus in Eyeriss (ISCA 2016), Tree in MAERI (ASPLOS 2018)
E.g., TPU (ISCA 2017), local network in Eyeriss (ISCA 2016)
Design-space of a DNN Accelerator

**HW Design-Space**

**Mapping Design-Space aka Map Space**

**Dataflow**

**Ordering (aka “stationary”)**

**Parallelism Dimension**

**DNN Dimensions**

**Mapping**

**HW Resources (PE, Buffers)**

**Dataflow Flexibility**

**Number of PEs**

**Buffer sizes (global/local)**

**NoCs**

**Data / Computation Tile Sizing**

**Number: Tile IDs**

**Filter Tiles**

**Input Tiles**

**Output Tiles**

**Tiling**

**Mapping on entire accelerator at time = 1**

**Tile Scheduling**

**Spatial Partitioning**

**Number: Tile IDs**

**Number of PEs**

**Buffer sizes (global/local)**

**NoCs**

**HW Design-Space**

**Mapping Design-Space aka Map Space**

**HW or SW depending on flexibility**
CPU Compute Model

Program → Compiler → Binary Program → Architecture → Processor → Processed Data

Input Data → Processor → Behavioral Statistics → Processor

μArchitecture → Compiler
DNN Compute Model

Model (Shape) \[\rightarrow\] Mapper \[\rightarrow\] DNN Accelerator

Dataflow

Configuration

Input Activations

Behavioral Statistics

Output Activations

Compiled Program

Processed Data

Behavioral Statistics
Putting it all together: Case Study of Google TPUv1 (2016)

“Systolic Array”
Systolic Array Structure

Schematic of MAC PE

MAC Processing Element (PE)

Systolic Array
Difference from CPU Architecture

• Registers distributed across PEs
  – One per operand per PE
    • Can me more than one as well

• Operands “forwarded” from one register to the other
  – More energy-efficient than reading large register file

• Stage data through array in deterministic manner
  – No need for hazard checks etc
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Walkthrough Example
Performance/watt

~200X incremental perf/W of Haswell CPU
~70X incremental perf/W of K80 GPU
NVIDIA Response: Tensor Cores

Google TPU – large tensor engine vs NVIDIA: multiple tiny tensor engines. Trade-off?
Summary

- High Throughput requirements and Energy costs of Data Movement are key drivers towards the trend towards custom accelerators.

- Heavy HW-SW Co-Design is used in practice to design accelerators.

- Open questions: how to avoid “over”-specialization.
Thank you!

Next Lecture: Accelerators-II