CSEP 548: Computer Systems Architecture

Dark Silicon, Specialization, Systems for ML
Luis Ceze, Spring 2017

(based on slides lifted from Me, Hadi Esmaeilzadeh, Michael Taylor, Carlo Del Mundo, Liang Luo and the interwebs at large)
What is the catch with Moore’s law?
Dennard scaling:
Doubling the transistors; scale their power down

Transistor: 2D Voltage-Controlled Switch

- Dimensions
- Voltage
- Doping Concentrations

Area: $0.5 \times \downarrow$
Capacitance: $0.7 \times \downarrow$
Frequency: $1.4 \times \uparrow$
Power = Capacitance $\times$ Frequency $\times$ Voltage$^2$
Power: $0.5 \times \downarrow$
Dark silicon
What if you can’t power them anymore?

Area $\times 0.5$ →
Power $\times 0.5$

$\downarrow$

Can’t turn all transistor on at the same time. Part of the chip gets “dark”.

Dark Silicon
Looking back
Evolution of processors

Single-core Era

1971
740 KHz

2003
3.4 GHz

2004

Multicore Era

2013
3.5 GHz

Is parallelism long-term solution?
What now?

Need at least 18%-40% per generation from architecture alone without additional power
Possible paths forward

- Do Nothing
- Technology Breakthrough
- Software Bloat Reduction
- Specialization and Co-design
- Biological Computing
- Quantum Computing
- Approximate Computing
Specialization and efficiency

Source: Bob Broderson, Berkeley Wireless group

Why?
The Value of a Bitcoin

USD/BTC Exchange rate (in USD)

Price ($)
BTC Mining Computing Evolution

- CPU
- GPU
  - Portable OpenCL Imp
  - Completely unrolled double SHA256 hash
  - AMD >> Nvidia
    - instruction set match
    - microarch (VLIW) match
    - higher ALU density
    - memory BW not used
- FPGA
  - verilog
  - “gateway drug to ASIC”: boards, protocols, thermals, verilog
- ASIC
Energy Costs and USD/BTC
Say when to unplug/plug HW

- daily $ per Gh/s falls as technology advances and more machines deployed
- daily $/GH/s rises if USD/BTC rises.
- Today, CPUs, GPUs, and even FPGAs do not recoup energy costs
- Rising USD/BTC: old machines get fired up.
- Steady state: cheap energy wins (Iceland?)
HW design in one slide

- Declare compute components, memory elements, interconnection
- “Place and route” distributes those in space
  - And checks is timing works --- i.e., all signals can be stable for a target clock frequency
  - Assess HW resource utilization, power consumption, etc.

```vhdl
-- Design Name : parity_using_assign
-- File Name   : parity_using_assign.v
-- Function    : Parity using assign
-- Coder       : Deepak Kumar Tala
--------------------------------------------------------------------
module parity_using_assign (  
data_in , // 8 bit data in  
parity_out // 1 bit parity out
);
output parity_out ;
input [7:0] data_in ;
wire parity_out ;

assign parity_out = (data_in[0] ^ data_in[1]) ^  
(data_in[2] ^ data_in[3]) ^  
(data_in[4] ^ data_in[5]) ^  
(data_in[6] ^ data_in[7]) ;
endmodule
```
Neural networks

neural network

computing a single layer

\[
\begin{bmatrix}
    x_7 \\
    x_8 \\
    x_9
\end{bmatrix}
= f \left( \begin{bmatrix}
    W_{67} & W_{57} & W_{47} \\
    W_{68} & W_{58} & W_{48} \\
    W_{69} & W_{59} & W_{49}
\end{bmatrix} \begin{bmatrix}
    x_6 \\
    x_5 \\
    x_4
\end{bmatrix} \right)
\]
Systolic Arrays

computing a single layer

\[
\begin{bmatrix}
X_7 \\
X_8 \\
X_9
\end{bmatrix}
= f \left( \begin{bmatrix}
W_{67} & W_{57} & W_{47} \\
W_{68} & W_{58} & W_{48} \\
W_{69} & W_{59} & W_{49}
\end{bmatrix}\begin{bmatrix}
X_6 \\
X_5 \\
X_4
\end{bmatrix} \right)
\]
Making it fast in HW

systolic array

processing unit

1 - processing elements in hardwired logic

2 - local storage for synaptic weights

3 - sigmoid unit implements non-linear activation functions

4 - vertically micro-coded sequencer
Scaling it up

PU control

PE

PE

PE

PE

Storage

f

PU control

PE

PE

PE

PE

Storage

f

PU control

PE

PE

PE

PE

Storage

f

PU control

PE

PE

PE

PE

Storage

f

PU control

PE

PE

PE

PE

Storage

f

bus

scheduler
Google’s Tensor Processing Unit (TPU)

- **30-80x** TOPS/watt vs. 2015 CPUs and GPUs.
- 8 GiB DRAM.
- 8-bit fixed point.
- 256x256 MAC unit.
- Support for data reordering, matrix multiply, activation, pooling, and normalization.

*Figure 3. TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.*
**Figure 1.** TPU Block Diagram. The main computation part is the yellow Matrix Multiply unit in the upper right hand corner. Its inputs are the blue Weight FIFO and the blue Unified Buffer (UB) and its output is the blue Accumulators (Acc). The yellow Activation Unit performs the nonlinear functions on the Acc, which go to the UB.

**Figure 2.** Floor Plan of TPU die. The shading follows Figure 1. The light (blue) data buffers are 37% of the die, the light (yellow) compute is 30%, the medium (green) I/O is 10%, and the dark (red) control is just 2%. Control is much larger (and much more difficult to design) in a CPU or GPU.
## Experimental Testbed

<table>
<thead>
<tr>
<th>Model</th>
<th>Die</th>
<th>Benchmarked Servers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mm²</td>
<td>nm</td>
</tr>
<tr>
<td></td>
<td>Idle</td>
<td>Busy</td>
</tr>
<tr>
<td>Haswell E5-2699 v3</td>
<td>662</td>
<td>22</td>
</tr>
<tr>
<td>NVIDIA K80 (2 dies/card)</td>
<td>561</td>
<td>28</td>
</tr>
<tr>
<td>TPU</td>
<td>NA*</td>
<td>28</td>
</tr>
</tbody>
</table>

Table 2. Benchmarked servers use Haswell CPUs, K80 GPUs, and TPUs. Haswell has 18 cores, and the K80 has 13 SMX processors. Figure 10 has measured power. The low-power TPU allows for better rack-level density than the high-power GPU. The 8 GiB DRAM per TPU is Weight Memory. GPU Boost mode is not used (Sec. 8). SECDEC and no Boost mode reduce K80 bandwidth from 240 to 160. No Boost mode and single die vs. dual die performance reduces K80 peak TOPS from 8.7 to 2.8. (*The TPU die is ≤ half the Haswell die size.)*

---

8x K80 GPUs

Figure 3. TPU Printed Circuit Board. It can be inserted in the slot for an SATA disk in a server, but the card uses PCIe Gen3 x16.
The Roofline Model

Performance [GFLOPS]

Operational Intensity [FLOPS/byte]
Stars are for the TPU, triangles are for the K80, and circles are for Haswell. All TPU stars are at or above the other 2 rooflines.
### App breakdown by Performance Counters

<table>
<thead>
<tr>
<th>Application</th>
<th>MLP0</th>
<th>MLP1</th>
<th>LSTM0</th>
<th>LSTM1</th>
<th>CNN0</th>
<th>CNN1</th>
<th>Mean</th>
<th>Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array active cycles</td>
<td>12.7%</td>
<td>10.6%</td>
<td>8.2%</td>
<td>10.5%</td>
<td>78.2%</td>
<td>46.2%</td>
<td>28%</td>
<td>1</td>
</tr>
<tr>
<td>Useful MACs in 64K matrix (% peak)</td>
<td>12.5%</td>
<td>9.4%</td>
<td>8.2%</td>
<td>6.3%</td>
<td>78.2%</td>
<td>22.5%</td>
<td>23%</td>
<td>2</td>
</tr>
<tr>
<td>Unused MACs</td>
<td>0.3%</td>
<td>1.2%</td>
<td>0.0%</td>
<td>4.2%</td>
<td>0.0%</td>
<td>23.7%</td>
<td>5%</td>
<td>3</td>
</tr>
<tr>
<td>Weight stall cycles</td>
<td>53.9%</td>
<td>44.2%</td>
<td>58.1%</td>
<td>62.1%</td>
<td>0.0%</td>
<td>28.1%</td>
<td>43%</td>
<td>4</td>
</tr>
<tr>
<td>Weight shift cycles</td>
<td>15.9%</td>
<td>13.4%</td>
<td>15.8%</td>
<td>17.1%</td>
<td>0.0%</td>
<td>7.0%</td>
<td>12%</td>
<td>5</td>
</tr>
<tr>
<td>Non-matrix cycles</td>
<td>17.5%</td>
<td>31.9%</td>
<td>17.9%</td>
<td>10.3%</td>
<td>21.8%</td>
<td>18.7%</td>
<td>20%</td>
<td>6</td>
</tr>
<tr>
<td>RAW stalls</td>
<td>3.3%</td>
<td>8.4%</td>
<td>14.6%</td>
<td>10.6%</td>
<td>3.5%</td>
<td>22.8%</td>
<td>11%</td>
<td>7</td>
</tr>
<tr>
<td>Input data stalls</td>
<td>6.1%</td>
<td>8.8%</td>
<td>5.1%</td>
<td>2.4%</td>
<td>3.4%</td>
<td>0.6%</td>
<td>4%</td>
<td>8</td>
</tr>
<tr>
<td>TeraOps/sec</td>
<td>12.3</td>
<td>9.7</td>
<td>3.7</td>
<td>2.8</td>
<td>86.0</td>
<td>14.1</td>
<td>21.4</td>
<td>9</td>
</tr>
</tbody>
</table>

**Table 3.** Factors limiting TPU performance of the NN workload based on hardware performance counters. Rows 1, 4, 5, and 6 total 100% and are based on measurements of activity of the matrix unit. Rows 2 and 3 further break down the fraction of 64K weights in the matrix unit that hold useful weights on active cycles. Our counters cannot exactly explain the time when the matrix unit is idle in row 6; rows 7 and 8 show counters for two possible reasons, including RAW pipeline hazards and PCIe input stalls. Row 9 (TOPS) is based on measurements of production code while the other rows are based on performance-counter measurements, so they are not perfectly consistent. Host server overhead is excluded here. The MLPs and LSTMs are memory-bandwidth limited but CNNs are not. CNNF results are explained in the text.
Latency Results (99%ile)

<table>
<thead>
<tr>
<th>Type</th>
<th>Batch</th>
<th>99th% Response</th>
<th>Inf/s (IPS)</th>
<th>% Max IPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>16</td>
<td>7.2 ms</td>
<td>5,482</td>
<td>42%</td>
</tr>
<tr>
<td>CPU</td>
<td>64</td>
<td>21.3 ms</td>
<td>13,194</td>
<td>100%</td>
</tr>
<tr>
<td>GPU</td>
<td>16</td>
<td>6.7 ms</td>
<td>13,461</td>
<td>37%</td>
</tr>
<tr>
<td>GPU</td>
<td>64</td>
<td>8.3 ms</td>
<td>36,465</td>
<td>100%</td>
</tr>
<tr>
<td>TPU</td>
<td>200</td>
<td>7.0 ms</td>
<td>225,000</td>
<td>80%</td>
</tr>
<tr>
<td>TPU</td>
<td>250</td>
<td>10.0 ms</td>
<td>280,000</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 4. 99-th% response time and per die throughput (IPS) for MLP0 as batch size varies for MLP0. The longest allowable latency is 7 ms. For the GPU and TPU, the maximum MLP0 throughput is limited by the host server overhead. Larger batch sizes increase throughput, but as the text explains, their longer response times exceed the limit, so CPUs and GPUs must use less-efficient, smaller batch sizes (16 vs. 200).
Programming the TPU

TensorFlow graph

TPU bitstream

TPU host instructions

Programming FPGAs

Description in high-level language

RTL

FPGA vendor RTL tools

FPGA bitstream
# NVIDIA’s Rebuttal to the TPU

<table>
<thead>
<tr>
<th></th>
<th>K80 2012</th>
<th>TPU 2015</th>
<th>P40 2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inferences/Sec &lt;10ms latency</td>
<td>1(\frac{1}{13})X</td>
<td>1X</td>
<td>2X</td>
</tr>
<tr>
<td>Training TOPS</td>
<td>6 FP32</td>
<td>NA</td>
<td>12 FP32</td>
</tr>
<tr>
<td>Inference TOPS</td>
<td>6 FP32</td>
<td>90 INT8</td>
<td>48 INT8</td>
</tr>
<tr>
<td>On-chip Memory</td>
<td>16 MB</td>
<td>24 MB</td>
<td>11 MB</td>
</tr>
<tr>
<td>Power</td>
<td>300W</td>
<td>75W</td>
<td>250W</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>320 GB/S</td>
<td>34 GB/S</td>
<td>350 GB/S</td>
</tr>
</tbody>
</table>

“CNNs constitute only about 5% of the representative NN workload for Google. More attention should be paid to MLPs and LSTMs. Repeating history, it’s similar to when many architects concentrated on floating-point performance when most mainstream workloads turned out to be dominated by integer operations.”
Neural acceleration

[Esmaeilzadeh et al.]

Find an approximate program component

Compile the program and train a neural network
Neural acceleration

Find an approximate program component

Compile the program and train a neural network

Execute on a fast Neural Processing Unit (NPU)

[Esmaeilzadeh et al.]
Summary of NPU results

<table>
<thead>
<tr>
<th>application</th>
<th>domain</th>
<th>error metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>blackscholes</td>
<td>option pricing</td>
<td>MSE</td>
</tr>
<tr>
<td>fft</td>
<td>DSP</td>
<td>MSE</td>
</tr>
<tr>
<td>inversek2j</td>
<td>robotics</td>
<td>MSE</td>
</tr>
<tr>
<td>jmeint</td>
<td>3D-modeling</td>
<td>miss rate</td>
</tr>
<tr>
<td>jpeg</td>
<td>compression</td>
<td>image diff</td>
</tr>
<tr>
<td>kmeans</td>
<td>ML</td>
<td>image diff</td>
</tr>
<tr>
<td>sobel</td>
<td>vision</td>
<td>image diff</td>
</tr>
</tbody>
</table>

0.8x - 11.1x (3x mean) speedup
1.1x - 21x (3x mean) energy red.

0.9x - 24x (3.7x mean) speedup
1.5x - 51x (6.8x mean) energy red.

1.3x - 38x (3.8x mean) speedup
0.9x - 28x (2.8x mean) energy red.
Batches, forward and backward propagation

1. A batch of samples are loaded into GPU.
2. The batch of samples does forward propagation and prediction error is derived.
3. The batch of samples undergoes backward propagation.
4. The model is updated and used for subsequent training.

Examples:

- **AlexNet**
  - 2012
  - 6 days
  - 2 GTX 580

- **ZFNet**
  - 2013
  - 12 days
  - GTX 580

- **VGGNet**
  - 2014
  - 21 days
  - 4 Titan Black

- **RESNET**
  - 2015
  - 21 days
  - 8 GPUs
Data Parallelism

1. Each device sees different parts of the data set. Devices work independently of each other.

2. Local gradient is calculated per device, and are communicated with parameter server during each batch.
Distributed DNN Training (MXNET, TENSORFLOW...)

Data Parallelism
1. Each device sees different parts of the data set. Devices work independently of each other.
2. Local gradient is calculated per device, and are communicated with parameter server during each batch.
3. The parameter aggregates all updates and apply changes to the next model.

Where is the bottleneck? How do we improve it?