Pete Bannon, Ganesh Venkataramanan, Dejit Das Sarma, Emil Talpes, Bill McGee & Team
PLATFORM GOALS

Autopilot hardware Features & Performance to support FSD
Retrofit existing HW2.x vehicles
Sub 100W
Lower part costs to enable redundancy architectures

Focus exclusively on Tesla requirements
Safety & Security
Reduce Software migration effort
FULL SELF DRIVING COMPUTER

Dual Redundant SOCs
Redundant Power supplies
Backward compatible connectors and form factor
Overlapping camera field with redundant paths
DRIVING THE CAR

Perceive & Plan

Sensors
Ultrasonic – Wheel Ticks – Steering Angle

Compare

Validate

Act

Actuator ECUs
FSD CHIP GOALS

> 50 TOPS of neural network performance

High utilization (~80%)
- Optimized for batchsize of one

Sub 40W/Chip
- Best in class power efficiency for Inference
- Latencies and design style of CPUs

GPU & CPUs for post processing & general purpose needs

Security & Safety needs

Modular to enable various platform redundancy uses
FSD CHIP

14nm FinFET CMOS
260 mm², 6 billions transistors
AEC Q100
37.5 x 37.5mm FCBGA
Already in production
FSD CHIP

Compute dominant chip

Tesla designed NN accelerator

Proven industry IPs for standard functions:

- CPUs
- GPU
- ISP
- H.265 video encoder
- Memory controller
- PHYs
- On chip interconnect
- Peripherals
NN ACCELERATOR

- 2 Independent instances
- 2Ghz+ Design
- 96x96 MACs (36.8 TOPS/NNA)
- Hardware SIMD, ReLU & Pool units
- 32MB SRAM/instance
  - Bandwidth Optimized
- Programs resident in SRAMs
- Simple programming model
DEVELOPMENT CHALLENGES

Short dev cycle
- 14 months from Arch to Tape out

Simplified implementation
- Choosing proven IPs
- Simpler clock and power distribution

Memory density & speeds

Balanced programmability, flexibility within fast paced development

Simulation challenges (Esp. NNA)
NNA DESIGN MOTIVATION

A single convolution is a 7 deep nested for loop:
1. For each Image
2. For each Output Channel
3. For each Output X position
4. For each Output Y position
5. For each Input Channel
6. For each Input Y within kernelY
7. For each Input X within kernelX

99.7% of operations are multiply accumulates

Speeding up just MACs, by orders of magnitude, makes Quantization/Pooling more performance sensitive

Dedicated Quantization and Pooling HW to speed things all around

<table>
<thead>
<tr>
<th>Operation</th>
<th>MOPS</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>34275</td>
<td>98.1</td>
</tr>
<tr>
<td>Deconvolution</td>
<td>576</td>
<td>1.6</td>
</tr>
<tr>
<td>ReLU</td>
<td>123</td>
<td>0.1</td>
</tr>
<tr>
<td>Pooling</td>
<td>13</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Example convolutional neural network

Inception like CNNs
CONVOLUTION REFACTORED FLOW

- Merge Output X & Output Y to create larger input to process
- Process OutputX.Y and Output Channel 96 at a time.

1. For each Image
2. For (Output X * Output Y), step 96
3. For each Output Channel, step 96
4. For each Input Channel
5. For each Input Y within KernelY
6. For each Input X within KernelX
OUR COMPUTE SCHEME

- Maximum Data sharing (Reduced SRAM and DRAM activity)
- Minimized Data shifting power
- Further Power reduction with smarter shifting
- Increased compute Bandwidth Utilization

Acc \((30b) = 8b \times 8b + \text{Acc}\)
**DESIGN PHILOSOPHY**

Flexible state machine based control logic to reduce control power overheads

- Special complex Ops for fusing commonly used sequences like RELU-Shift-Sat
- Loop constructs built into state machines

Eliminate DRAM reads/writes

Minimize SRAM reads

Optimized MAC switching power
- In place Data Reuse vs result movement

Single clock domain

DVFS enabled power & clock distribution

<table>
<thead>
<tr>
<th>Icache</th>
<th>Register File</th>
<th>Control</th>
<th>ADD</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.0pJ</td>
<td>6.0pJ</td>
<td>39.0pJ</td>
<td>0.1pJ</td>
</tr>
</tbody>
</table>

* Mark Horowitz “Computing’s Energy Problem (and what we can do about it)”, ISSCC 2014
Compact instruction set yet Powerful and Flexible
Limited Out of order (DMA Rd, DMA Wr and Compute can be OOO & Simultaneous)

32B to 256B

Opcodes | Parameters | Dependencies | Extensions
---|---|---|---
DMA Read | Kernels Params | Flags | SIMD Sequences
DMA Write | DMA params | Errors | Complex Operations
Convolution | Compression/Decompression | Synch |
Deconvolution | Pre-calculated hints | | 
Inner-product | Loops | |
Scale | |
Eltwise | |
Stop | | |
NNA MICROARCHITECTURE

DMA Rd
DMA Rd
Conv
SIMD
DMA Wr
DMA Rd
Conv
...

Network Programs

DMA Rd
Conv
SIMD
Conv
...

Network Sequencer

Address Sequencer

Data Address Cache

Req Count Estimator

Base BLK
Addr

Weight Address

+96

Weight Address

Data Miss Addr

Rd Req

Rd Req

Wt Addr

Addr0

Addr1

Addrn

Data Address Sequencer

Rd Req

Rd Req

Nxt
Addr

Data Seqencer

SRAM BANKS

32MB

256B

128B

Weight buffers

Mul Acc Array 96x96

Block Cache

Data Aligners

SIMD Lanes

128B

32MB

Deconv

Pooling

wr Coalescing Buf

Bk End DP

Command Sequencer

Cmd Q

Control & Status

DMA Rd

DMA Rd

Conv

SIMD

DMA Wr

DMA Rd

Conv

...

Commands
Programmable SIMD unit

Rich Instruction set
- Signed/Unsigned INT & FP32 arithmetic
- Predication support for all instructions

Pipelined implementation of Quantization
- Fuses ReLu, Scale and Normalization layers

Full SIMD Program support
- Argmax, Exponential, Sigmoid, Tanh and other functions
POOLING HARDWARE

Average and Max pooling support
Built for most common small pooling sizes
Rearranges output pixels to implement faster pooling
Average pooling implemented with scaled reciprocals to avoid slow divide operation
Larger pooling sizes are processed in MAC datapath

Input – 96B

96B x96B Pooling Array with Byte level Controls

96 Poolers Max or Average Kx.Ky

96 Effective Divides 1/(Kernel size)
Using Scale Reciprocal

Output 96B
RESULTS

Power

HW 2.5: 57W
FSD: 72W
NNA: 15W

1.25 x
RESULTS

Cost

0.8 x
SUMMARY

- Completely optimized SOC from scratch
- Outstanding Perf/W for Tesla’s networks with NNA
- Enables full redundancy at optimal cost
- You can own one today

FSD Computer will help enable new safety and autonomy levels of the future