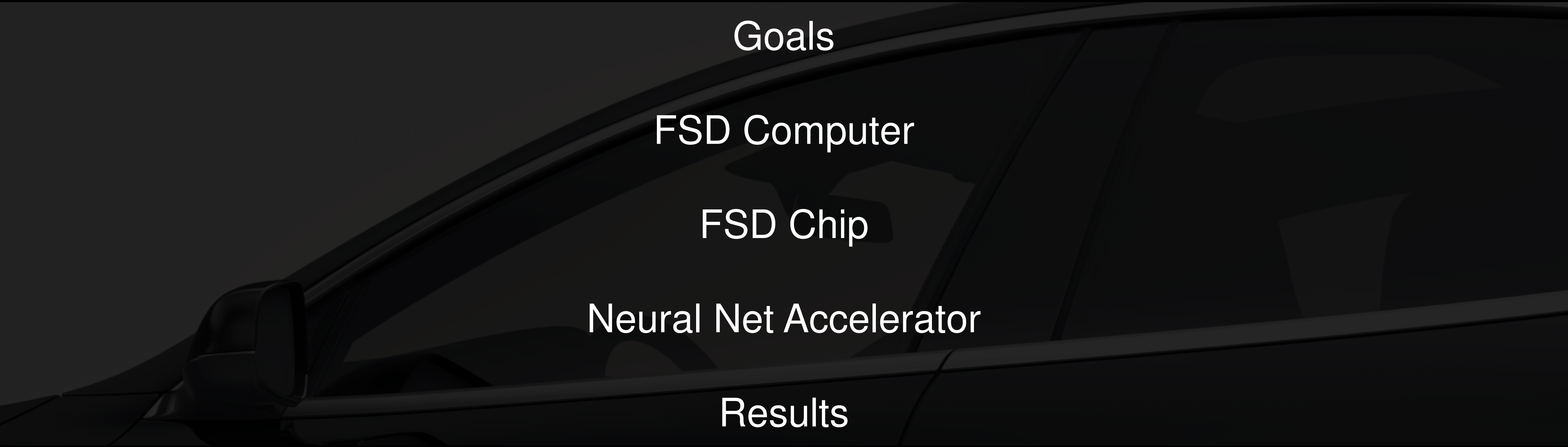
COMPUTE AND REDUNDANCY SOLUTION FOR THE

FULL SELF-DRIVING COMPUTER

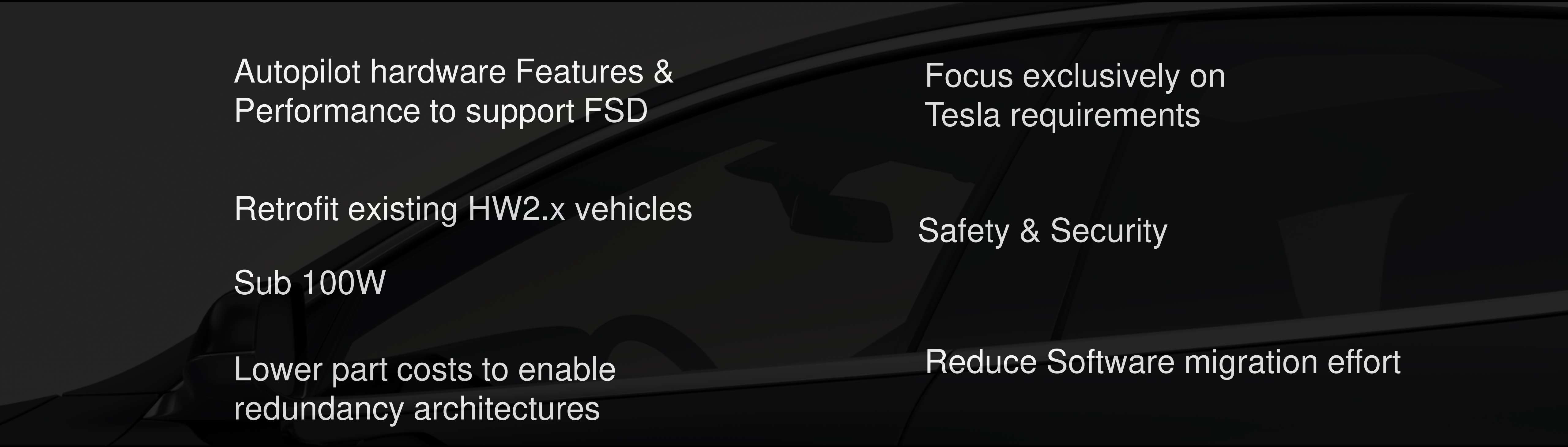
Pete Bannon, **Ganesh Venkataramanan**, **Debjit Das Sarma**, Emil Talpes, Bill McGee & Team

OUTLINE



TESLA

PLATFORM GOALS



Autopilot hardware Features & Performance to support FSD

Retrofit existing HW2.x vehicles

Sub 100W

Lower part costs to enable redundancy architectures

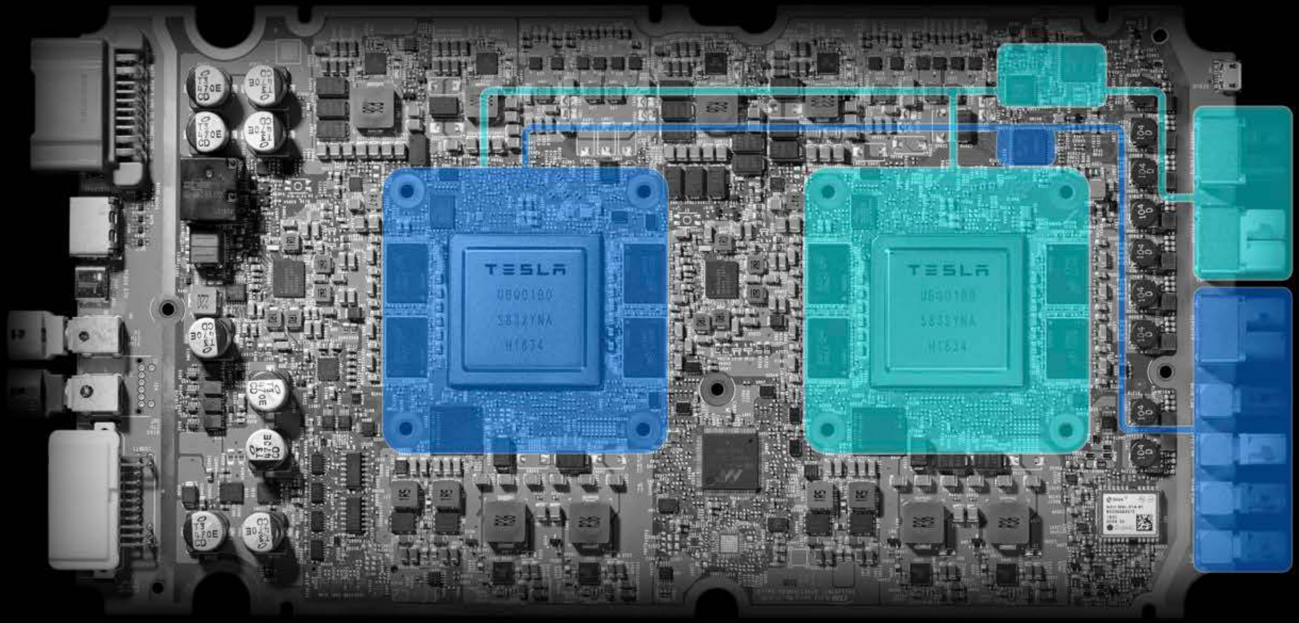
Focus exclusively on Tesla requirements

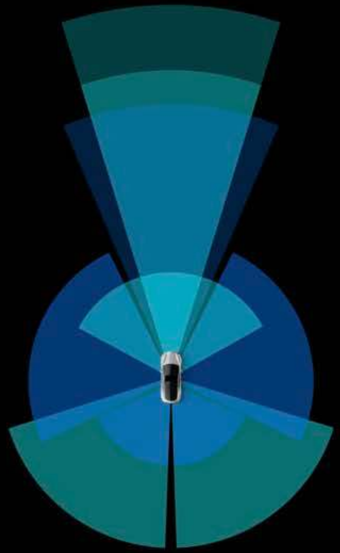
Safety & Security

Reduce Software migration effort

TESLA

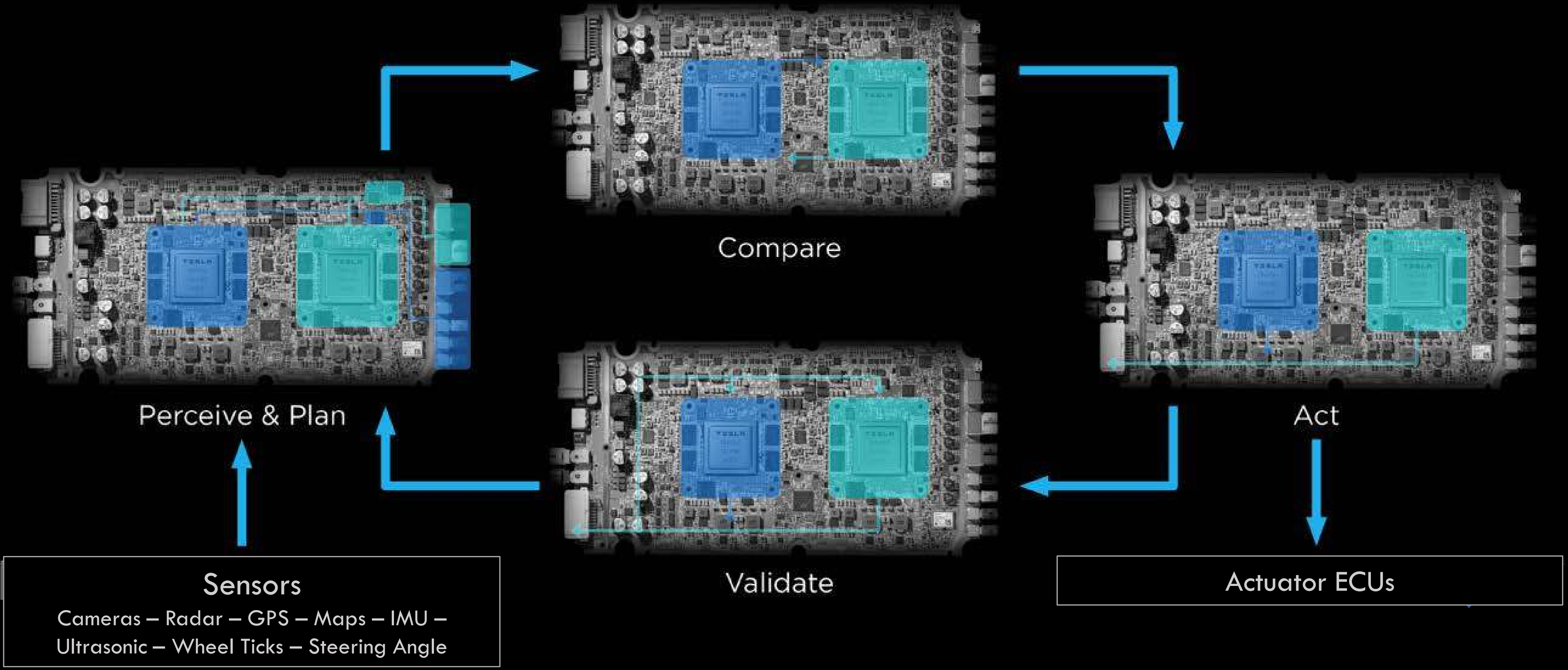
FULL SELF DRIVING COMPUTER





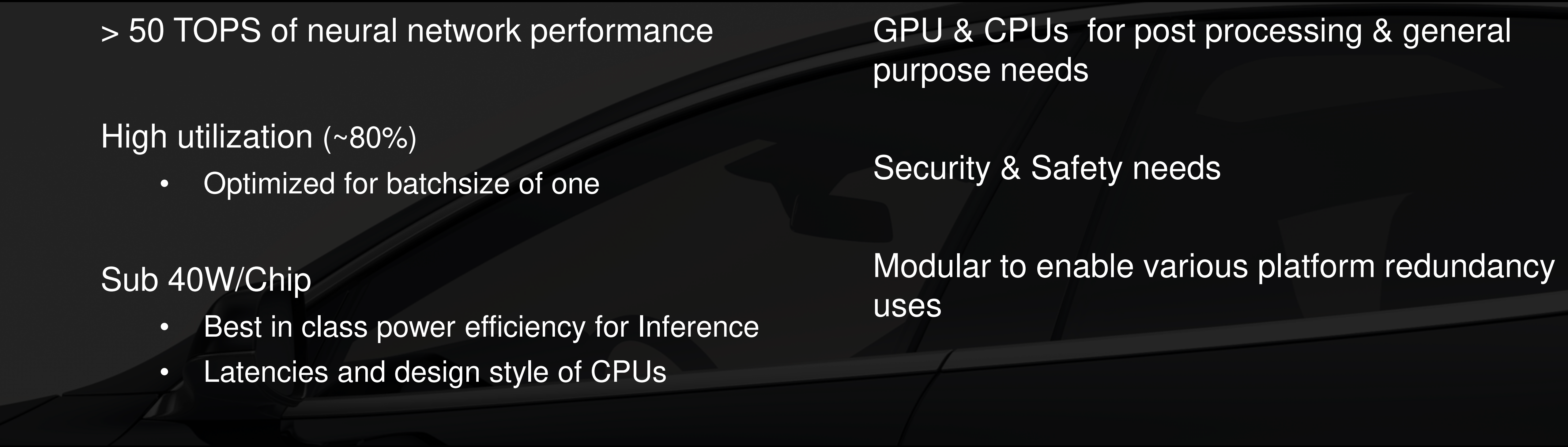
Dual Redundant SOCs
Redundant Power supplies
Backward compatible connectors and form factor
Overlapping camera field with redundant paths

DRIVING THE CAR



Compare
Perceive & Plan
Act
Validate
Sensors
Cameras – Radar – GPS – Maps – IMU – Ultrasonic – Wheel Ticks – Steering Angle
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

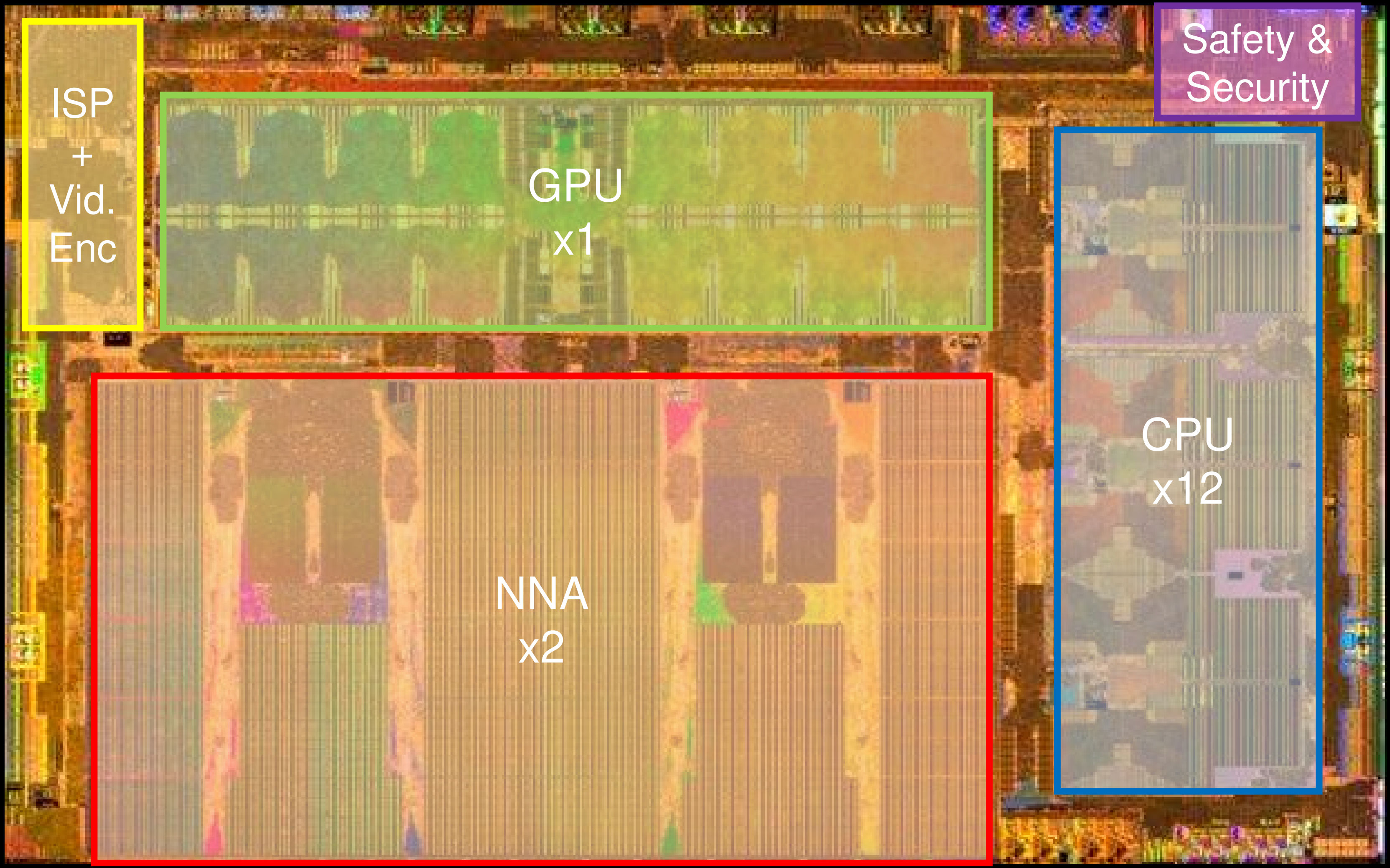
Security & Safety needs

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



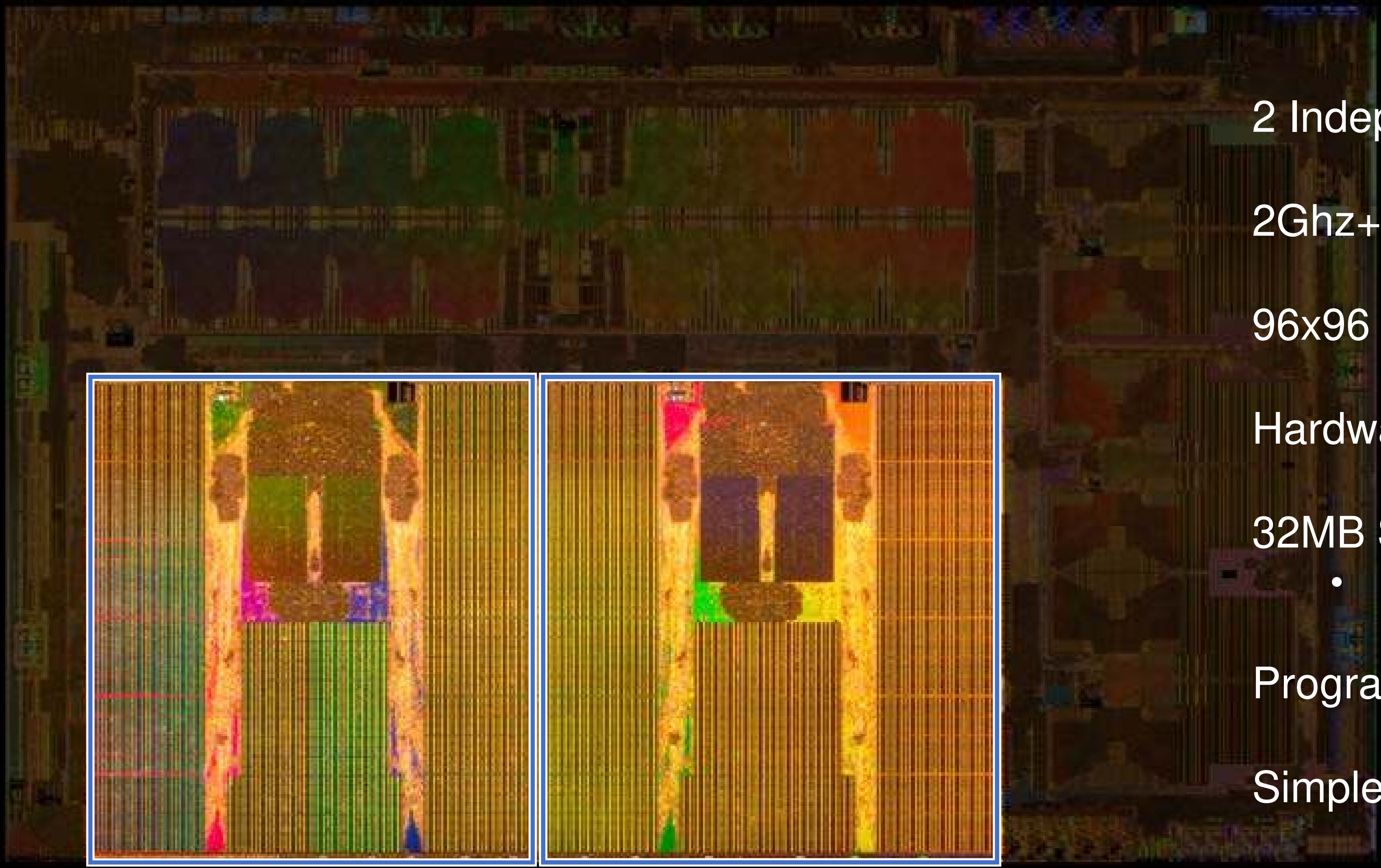
Compute dominant chip

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

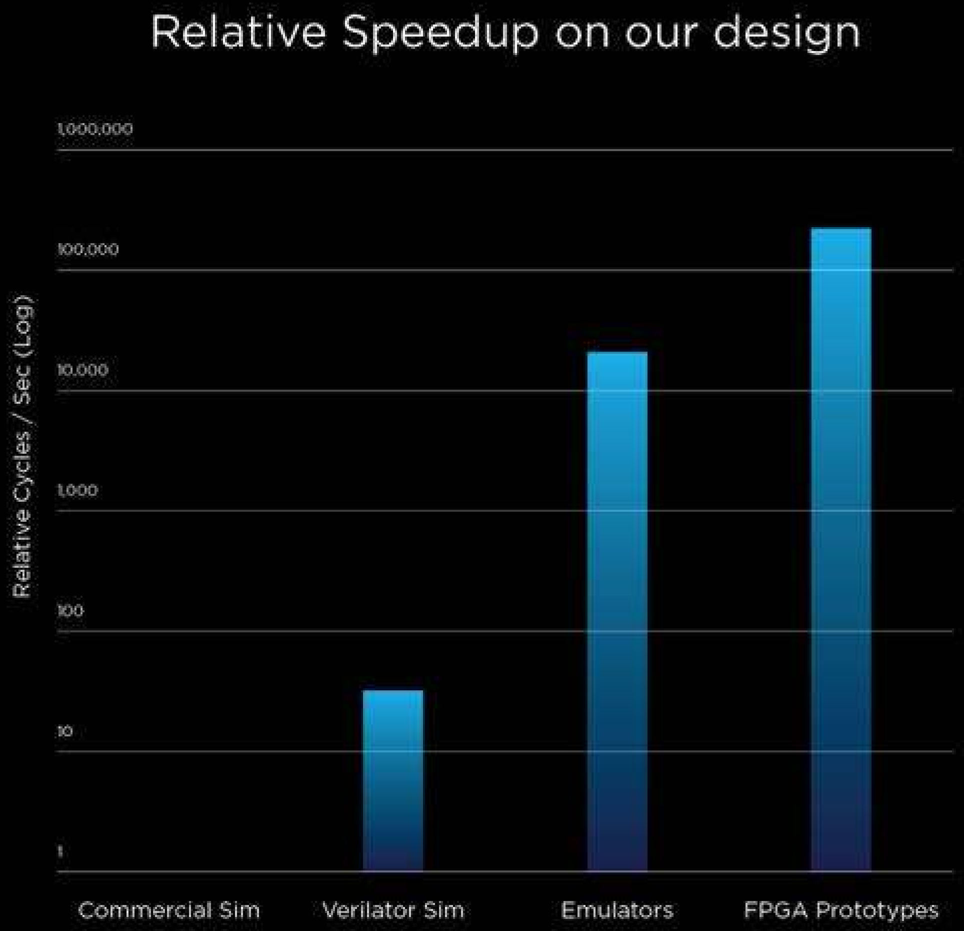
Simplified implementation

- Choosing proven IPs
- Simpler clock and power distribution

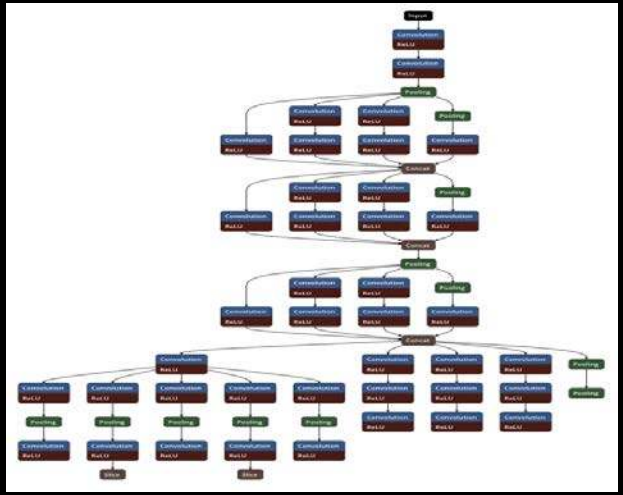
Memory density & speeds

Balanced programmability, flexibility within fast paced development

Simulation challenges (Esp. NNA)



NNA DESIGN MOTIVATION


Example convolutional neural network

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
```

Operation	MOPS	%
Convolution	34275	98.1
Deconvolution	576	1.6
ReLU	123	0.1
Pooling	13	0.2

Inception like CNNs

99.7% of operations are multiply accumulates

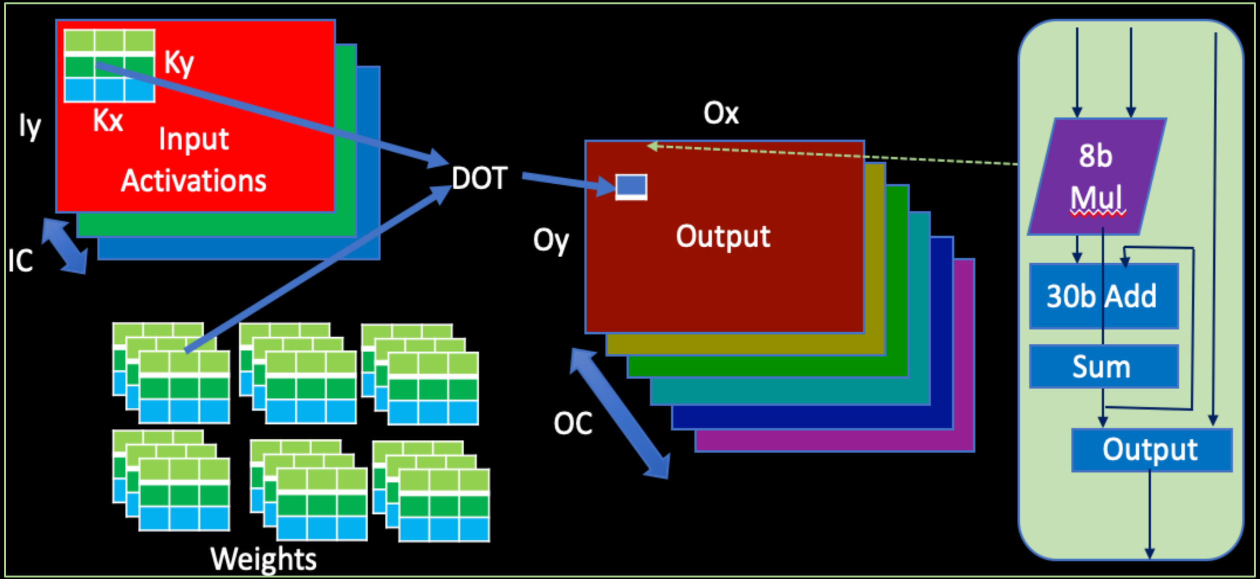
Speeding up just MACs, by orders of magnitude, makes Quantization/Pooling more performance sensitive

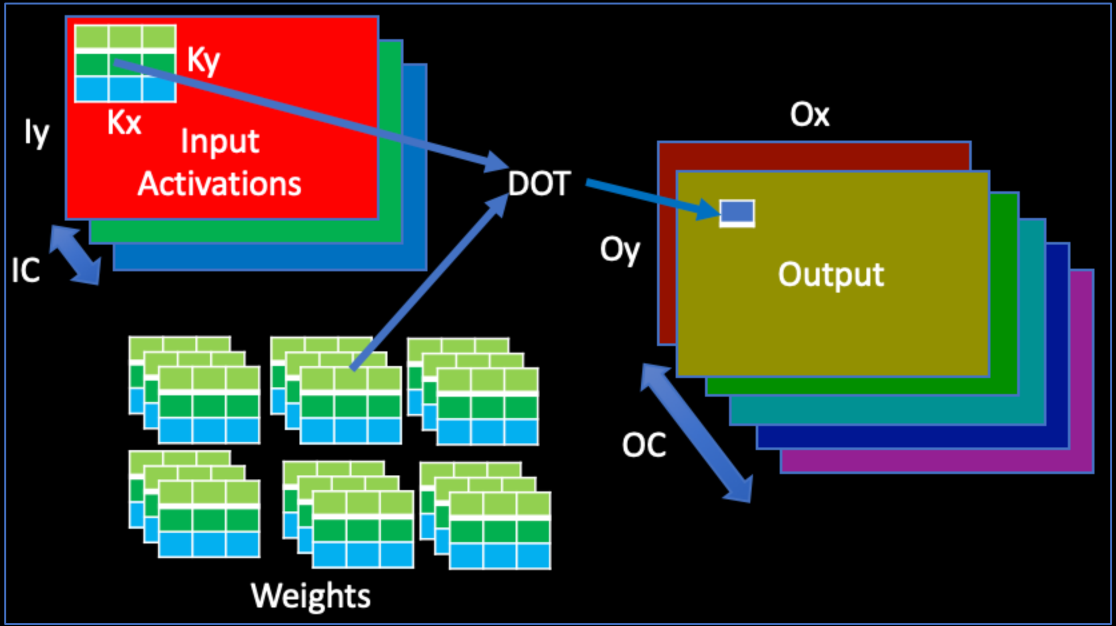
Dedicated Quantization and Pooling HW to speed things all around

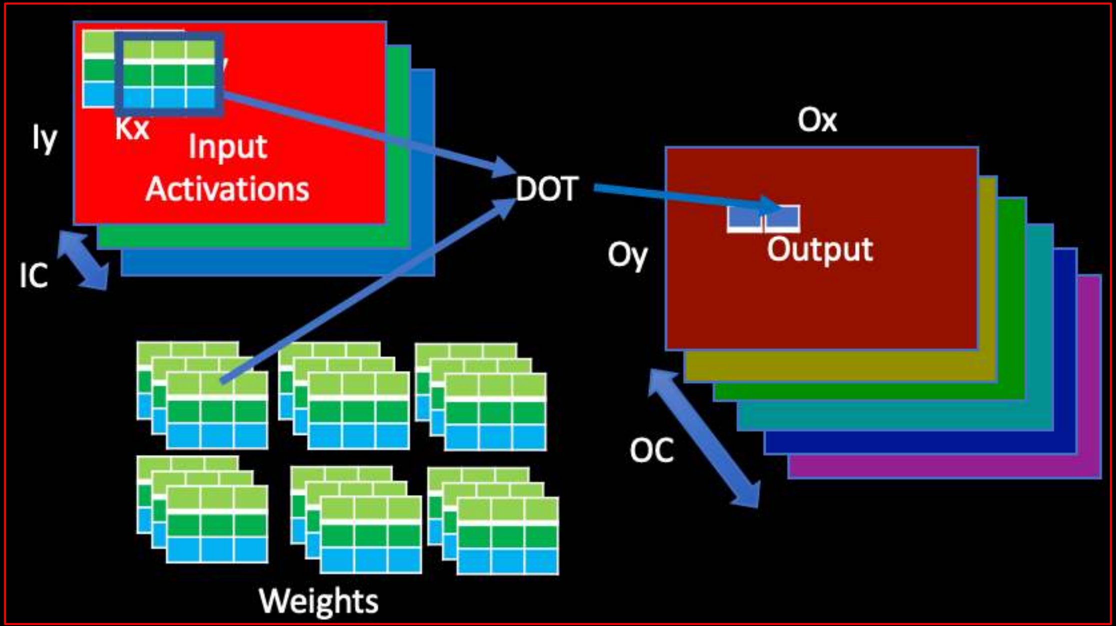
CONVOLUTION REFACTORED FLOW

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

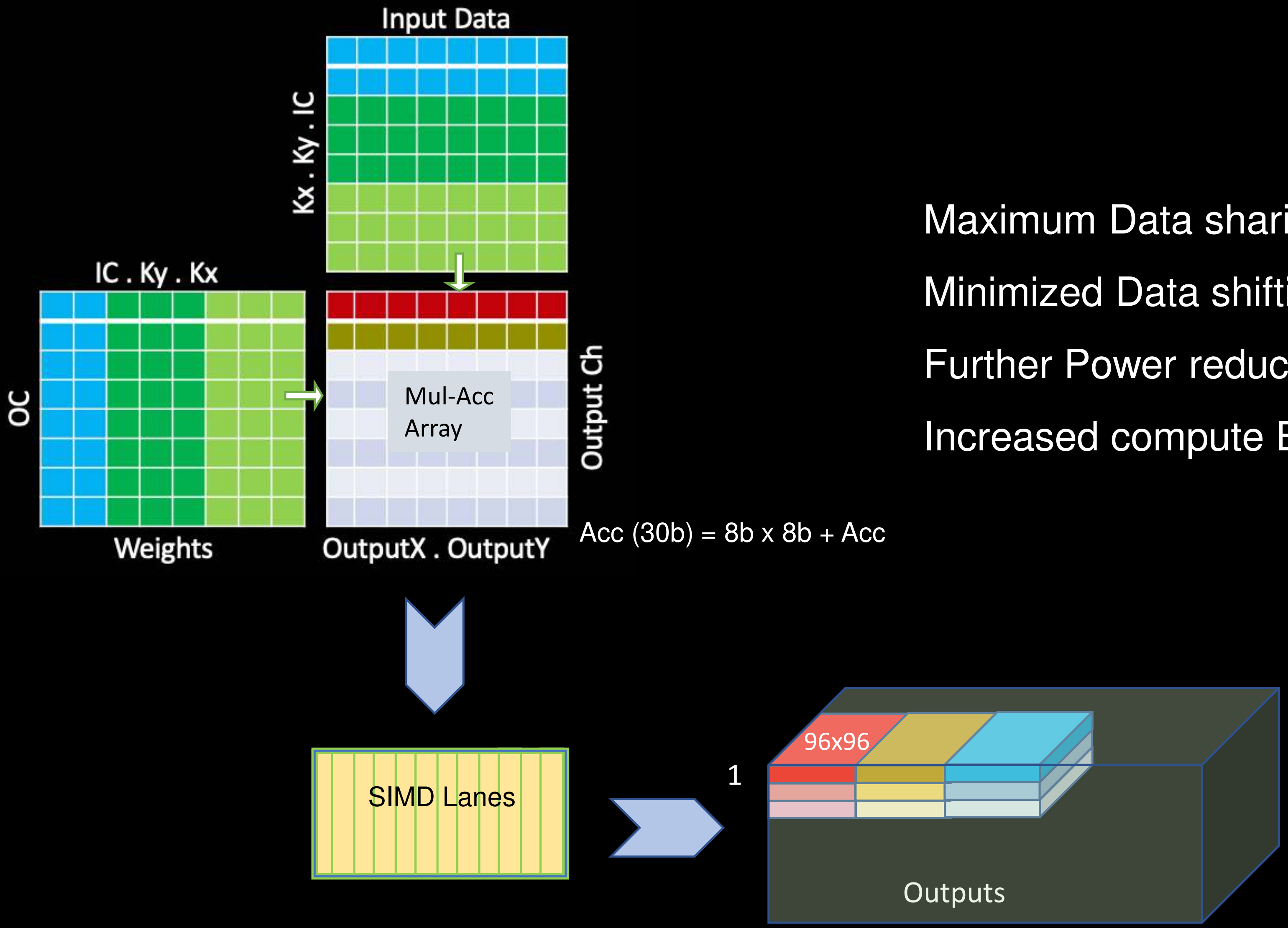
```
1.  For each Image
2.      For (Output X * Output Y), step 96
3.          For each Output Channel, step 96
4.              For each Input Channel
5.                  For each Input Y within KernelY
6.                      For each Input X within KernelX
```







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

DESIGN PHILOSPHY

Icache	Register File	Control	ADD
~~25.0pJ~~	~~6.0pJ~~	~~39.0pJ~~	0.1pJ

* Mark Horowitz "Computing's Energy Problem (and what we can do about it)", ISSCC 2014

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

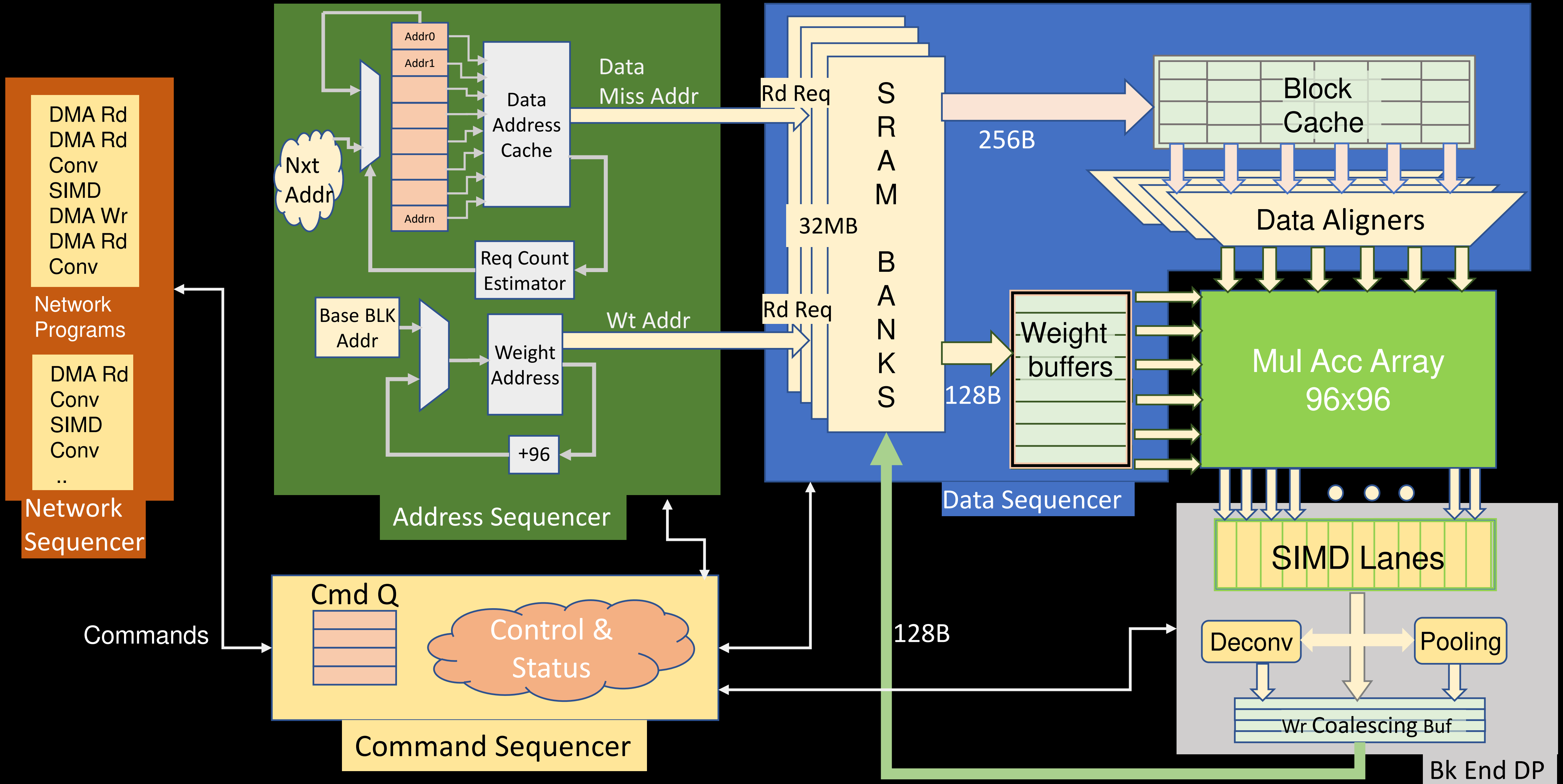
INSTRUCTION SET

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

NNA MICROARCHITECTURE



TESLA

SIMD DATAPATH

Programmable SIMD unit

Rich Instruction set

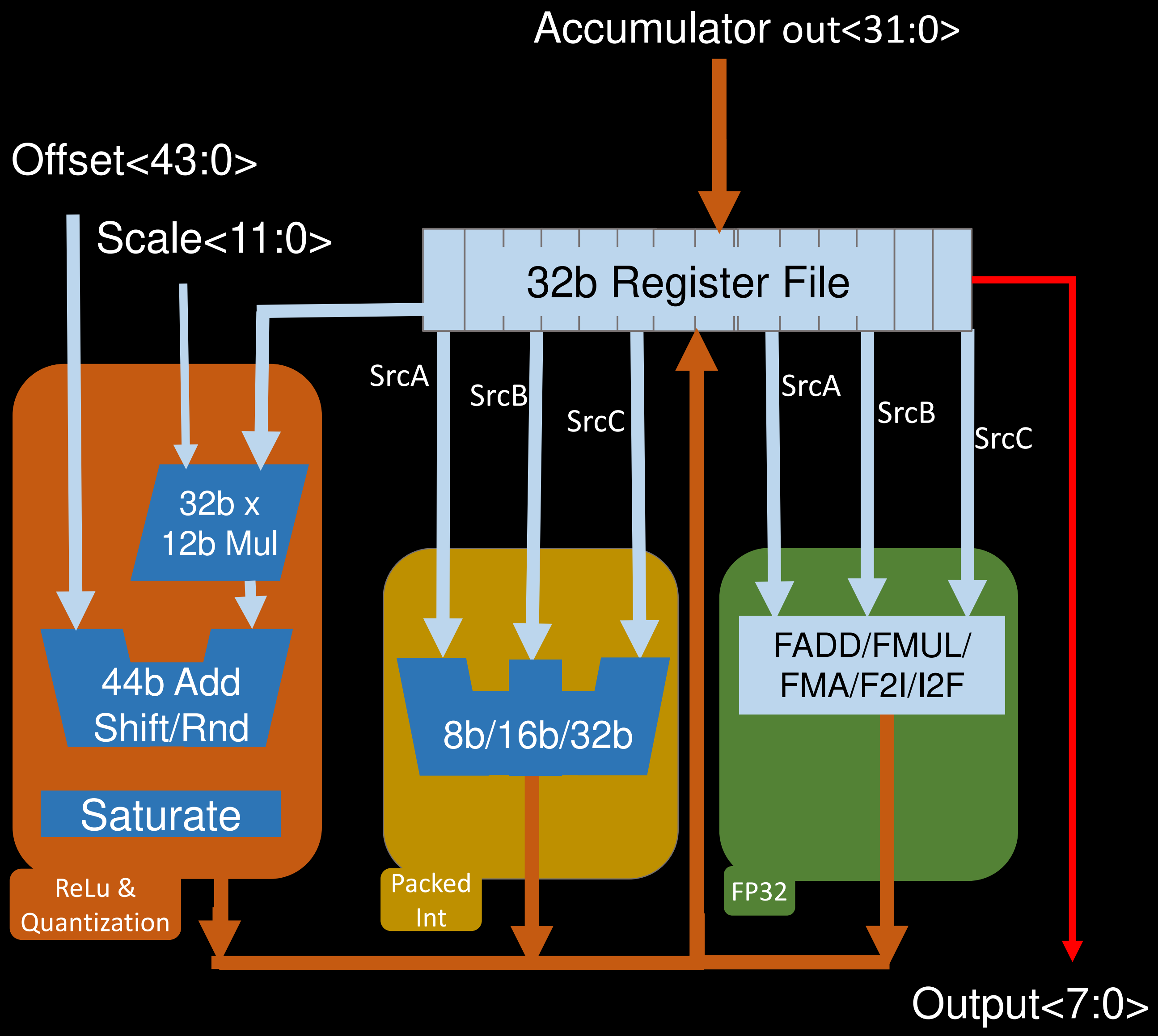
- Signed/Unsigned INT & FP32 arithmetic
- Predication support for all instructions

Pipelined implementation of Quantization

- Fuses ReLu, Scale and Normalization layers

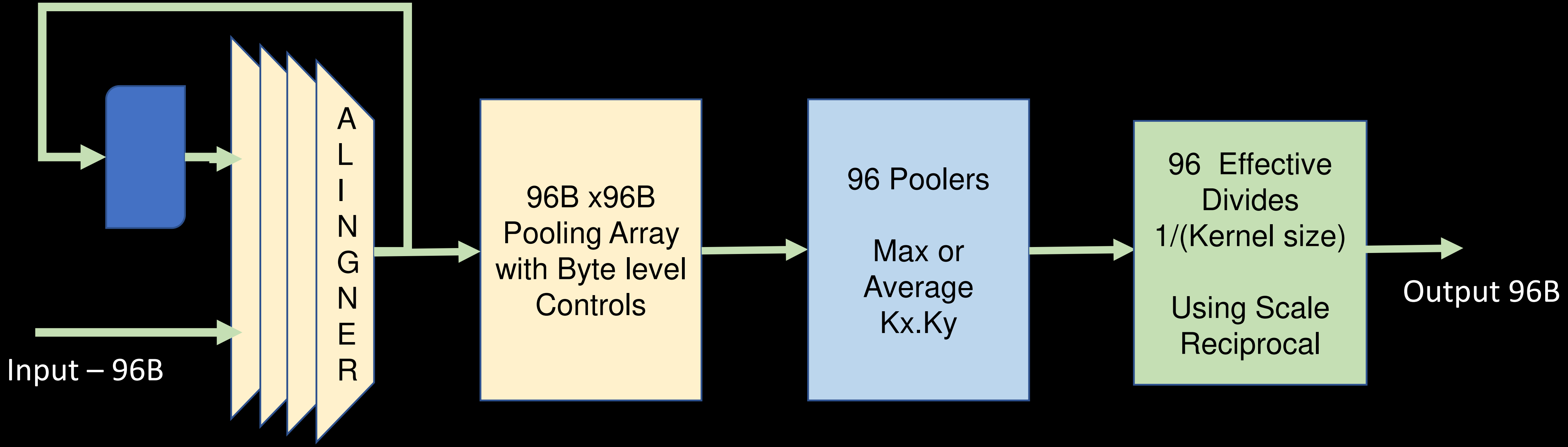
Full SIMD Program support

- Argmax, Exponential, Sigmoid, Tanh and other functions



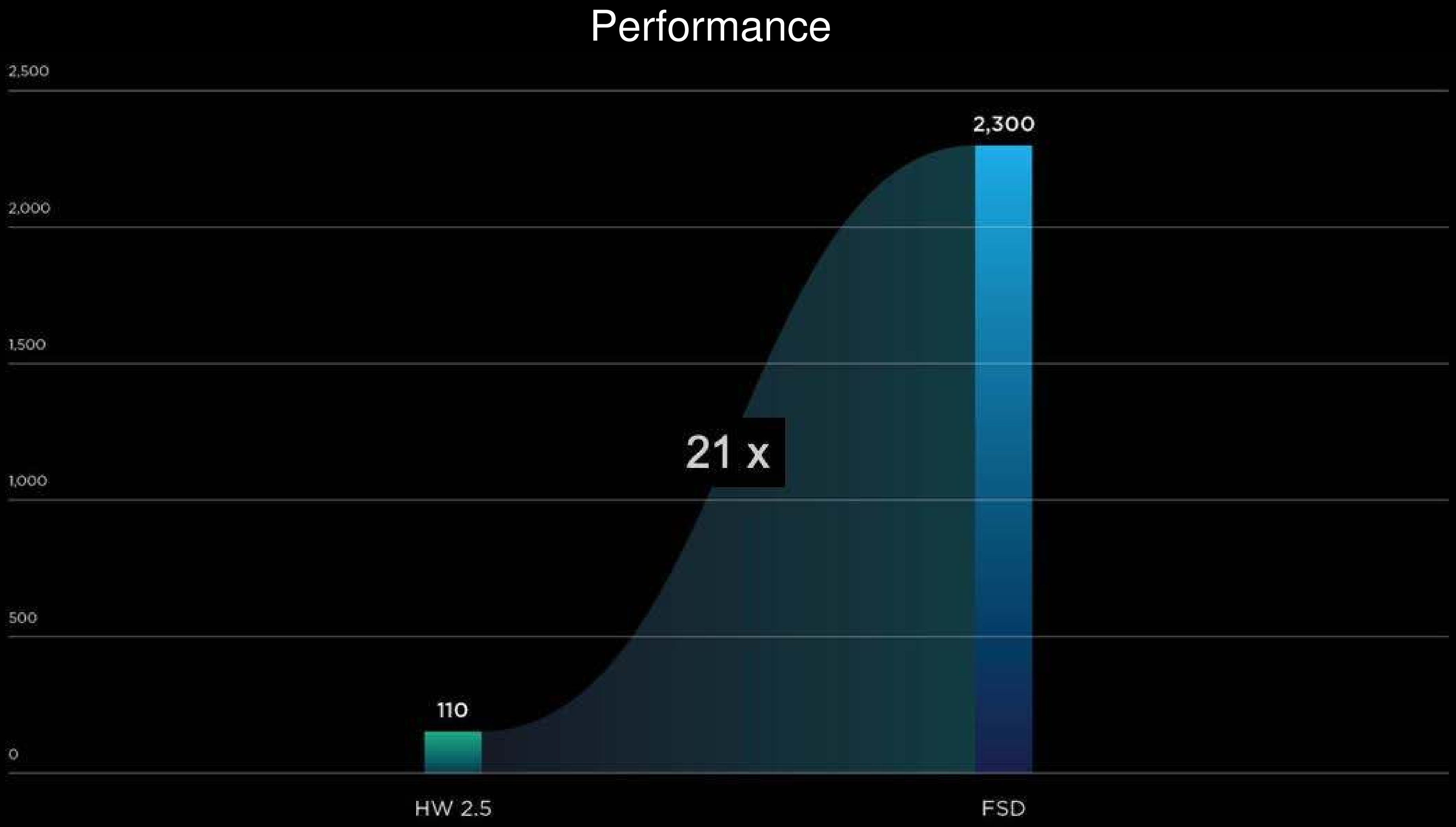
POOLING HARDWARE

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

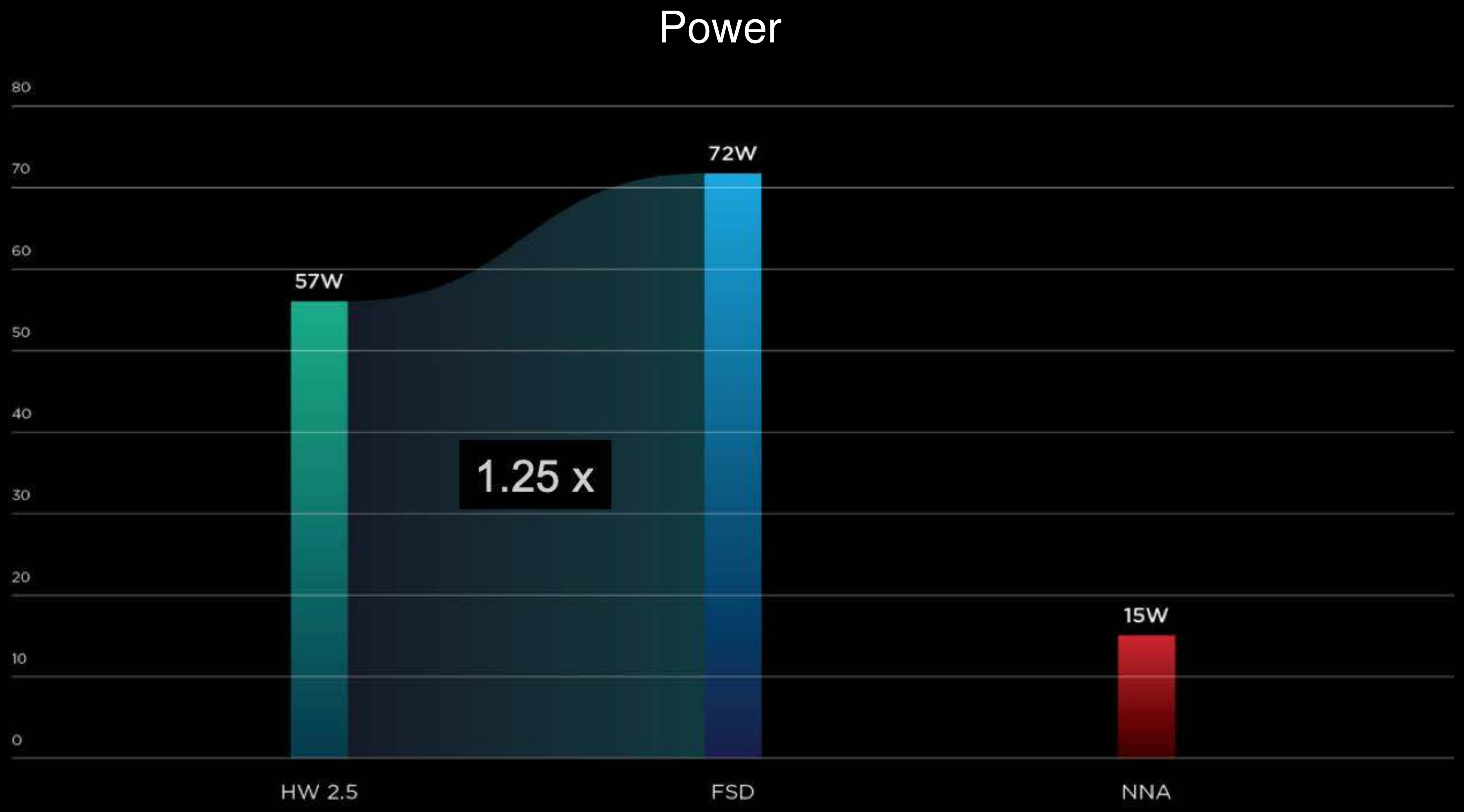


ALIGNER
96B x96B Pooling Array with Byte level Controls
96 Poolers
Max or Average Kx.Ky
96 Effective Divides 1/(Kernel size)
Using Scale Reciprocal
Input – 96B
Output 96B

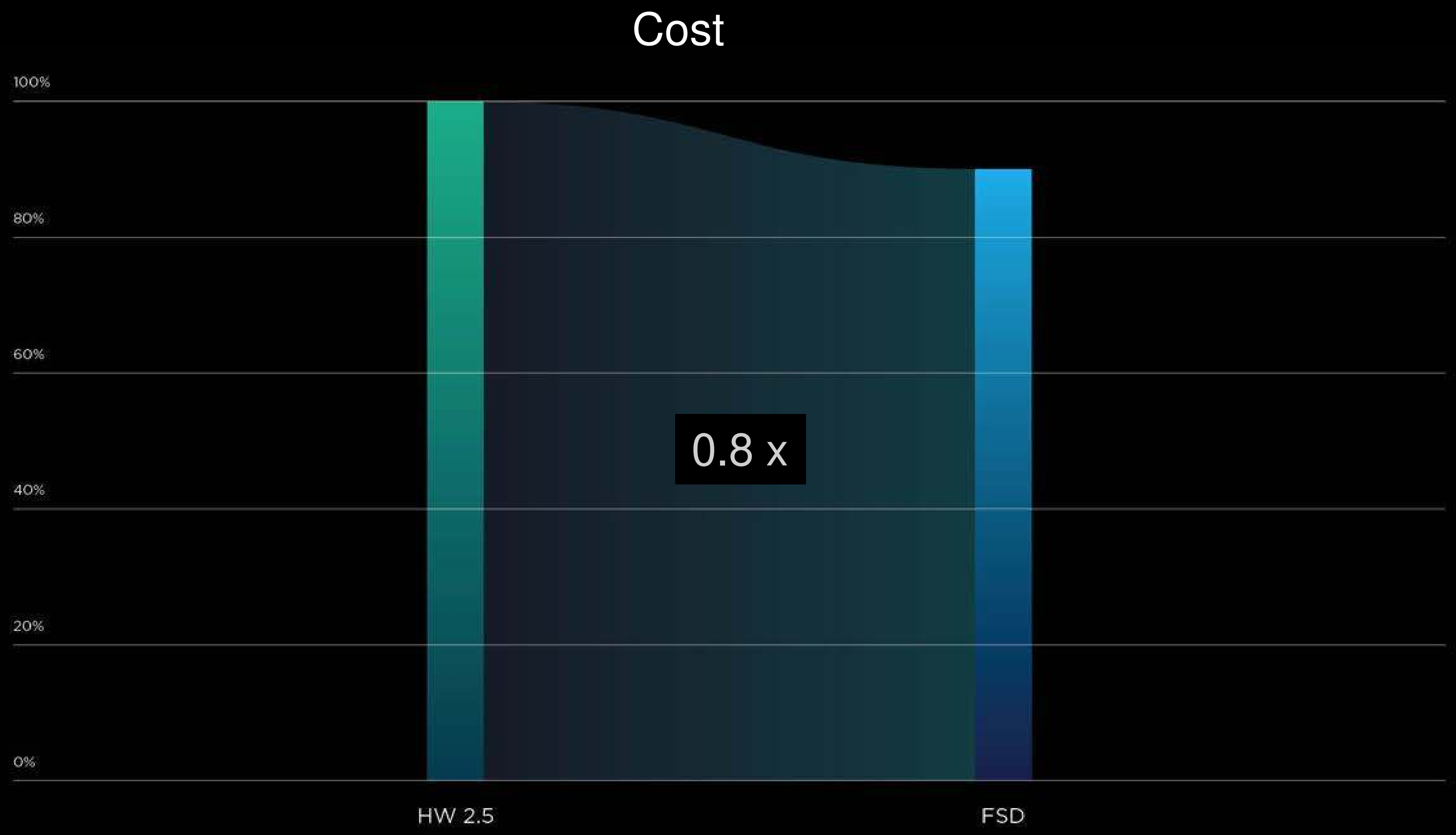
RESULTS



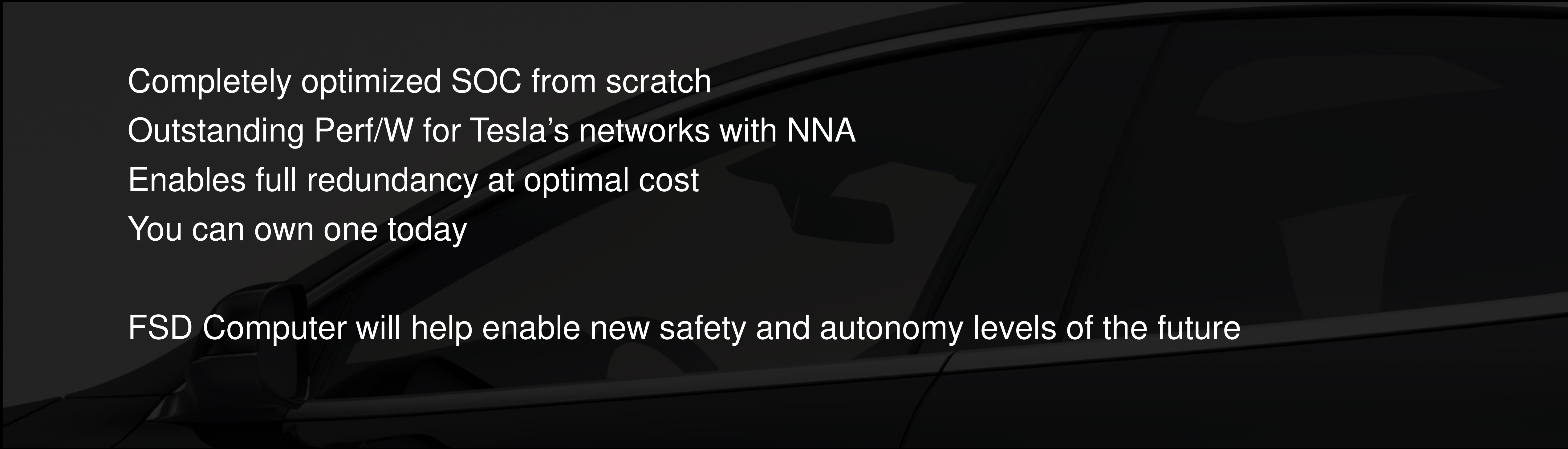
RESULTS



RESULTS



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

S I M P L E - P O W E R F U L - E F F I C I E N T