A Deep Dive into AI Chip Arithmetic Engines

SERGIO MARCHESE, OneSpin Solutions, Munich, Germany

Tesla's autopilot chip executes 72-trillion additions and multiplications per second: It better get the math right

ARTIFICIAL INTELLIGENCE (AI) IS steadily progressing toward advanced, high-value applications that will have a profound impact on our society. Automobiles that can drive themselves are perhaps the most talked about, imminent technological revolution, but there are many more applications of AI.

AI software, such as a neural network (NN) implementing a machine learning (ML) or deep learning (DL) algorithm, requires high-performance "artificial brains," or hardware, to run on. Computer vision is fundamental to many complex, safety-critical decision-making processes.

Since AlexNet won the ImageNet competition in 2012, convolutional neural networks (CNNs) have become the method of choice to perform accurate image classification and object recognition. Hardware platforms targeting computer vision and other NN-based applications can speed up execution and reduce power consumption of AI, making real-world, real-time applications possible. AI chips and hardware accelerators that power ML and DL algorithms include large arrays of specialized resources that can be directly mapped to –– and parallelize the execution of –– the required computational steps. Xilinx's Versal, a notable example of a heterogeneous computing platform, includes AI engines optimized for ML inference tasks, in addition to traditional central processing unit (CPU) and graphics processing unit (GPU) resources, and programmable logic.

Autonomous vehicles (AVs) are one of the most promising, disruptive innovations expected to become reality within a few years. Tesla, along with many other market players, is investing heavily in the AI technology that underpins AVs. This commitment is so strong that, despite not having previous hardware development capabilities, it decided to develop its own, highly specialized chip. (See Figure 1.)

Tesla's full self-driving (FSD) chip, presented in April 2019, includes two neural network accelerators (NNAs) developed in house. It also integrates third-party intellectual property (IP), including a GPU and an Arm-based processor subsystem. Tesla claims that the FSD computer, already deployed in production and based on a board that includes two FSD chips, is 21X faster than its previous, NVIDIA-based solution, while also being 20% less expensive (excluding development cost). According to Tesla, the FSD computer will be able to support autonomous driving once software catches up.

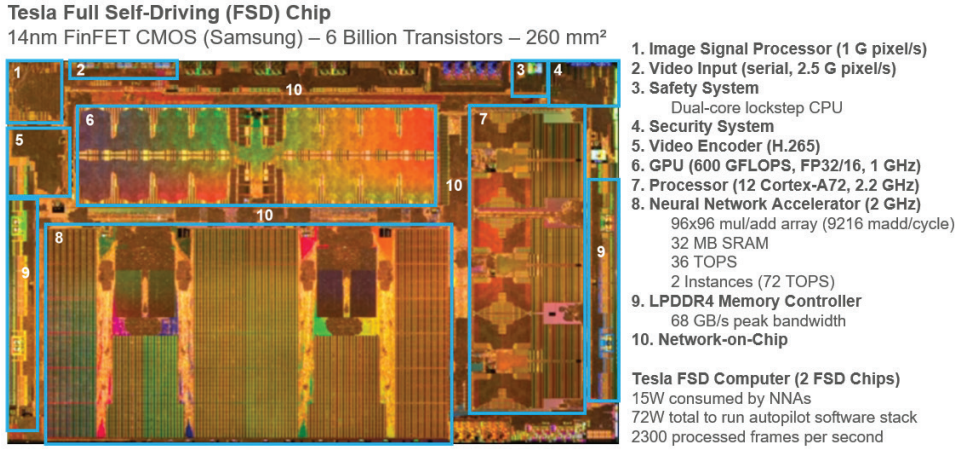

Figure 1. Tesla's FSD chip, presented in April 2019. The chip integrates third-party IP, including a GPU and processor and two instances of a neural network accelerator developed in house. *Source: Tesla (die picture) and OneSpin (annotations).*

The art of approximation

The Tesla CNN performs a few different types of data processing operations, but not in equal numbers. The operation that gets used most frequently by a huge margin is convolution (see Figure 2 below). The computational workload of a convolution layer may involve deeply nested

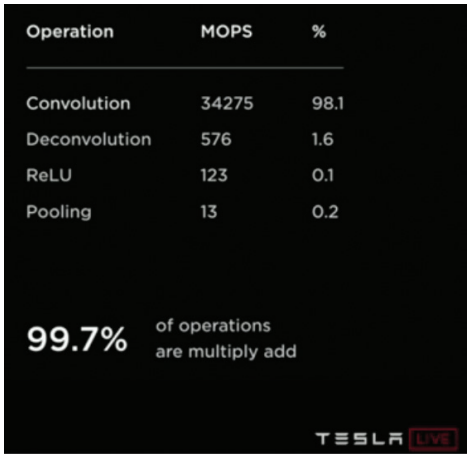

Figure 2. Tesla's neural network performs four main types of data processing operations, with convolution being the most frequently used. These operations are largely based on arithmetic multiplication and addition. *Source: Tesla.*

loops (see Figure 3 below).

Convolution and deconvolution are based on arithmetic multiplication and addition. While CNNs are the realm of experts, more than 99% of the basic operations they perform (99.7% in the case of Tesla) are good old multiplication and addition. It is obvious that AI chips must support these operations efficiently.

A crucial question is how much precision is required by the CNN. More precision requires more complex hardware, which ultimately leads to higher cost and power consumption. Tesla uses integer arithmetic, 32 bit for addition and 8 bit for multiplication. Each FSD chip NNA includes an array of 96x96 multiplication and addition hardware units and can perform 72-trillion operations per second (TOPS) without draining the car's battery.

However, CNNs often require floating-point (FP) arithmetic. FP representations of real numbers (see Figure 4) have significant advantages over fixed-point. For example, given a fixed bit width for binary encoding, FP formats cover a much wider range of values without losing precision. Half precision (16 bits) is typically sufficient for AI platforms. Lower precisions such as bfloat16, 12 bits or 8 bits, have also been demonstrated to adequately support certain applications. Implementing CNNs on embedded devices poses even tougher requirements on storage area and power consumption. While using low-precision, fixed-point representations of CNN weights and activations may be a viable option, this papers argue that using FP numbers may result in significantly more efficient hardware implementations. Fused multiply-add (FMA) operations, where rounding is computed only on the final result, provide additional performance improvements.

FP hardware units are much harder to design compared to fixed-point. The IEEE 754 standard defines many corner-case scenarios and non-ordinary values, such as +0, -0, signed infinity, and NaN (not a number). Moreover, there are four possible rounding modes (roundTowardZero, roundTiesToEven, roundTowardPositive, and roundTowardNegative), as well as five exception flags (invalid operation, division by zero, inexact result, underflow and overflow). How can engineers be absolutely sure that the hardware will always compute the correct result?

Getting the math right

To make the right decisions, tremendously smart AI systems must first get the basic math right. Even an apparently minor bug, for example, causing a small rounding mistake, can have a huge impact as errors may accumulate over many operations. Rigorous, pre-silicon verification of hardware designs must detect errors before integrated circuits (ICs) are manufactured or programmed and shipped.

Simulation-based verification relies on input vectors to stimulate the design. Certain corner cases defined by the IEEE standard could be missed. Moreover, even for half-precision FP hardware, there is a huge number of combinations to verify. Two input operands of 16 bits each generate 2^32 different combinations. Exhaustive testing using simulation is not feasible.

Unlike simulation, formal methods use mathematical proof techniques that can exhaustively verify control and data processing logic. The 1994 Intel Pentium FP division bug had an estimated cost of $475 million. Since then, a number of academic institutions and semiconductor heavyweights, including Intel, AMD, and IBM, have carried out a considerable amount of research in the application of formal methods to FP hardware verification. Although these methods have been successful in the verification of complex industrial designs, they suffer from drawbacks that have hindered widespread adoption. Some methods use non-commercial, proprietary tools, while others rely on theorem provers, thus requiring highly specialized skills and considerable engineering effort. Further, results are hard to reuse across different designs.

Over the past 15 years, formal

```
for(r=0; r<R; r++)              //output feature map
  for(q=0; q<Q; q++)            //input feature map
    for(m=0; m<M; m++)          //row in feature map
      for(n=0; n<N; n++)        //column in feature map
        for(k=0; k<K; k++)      //row in convolution kerenel
          for(l=0; l<L; l++) //column in convolution kernel
            Y[r][m][n]+=W[r][q][k][l]*X[q][m+k][n+l];
```

Figure 3. An example of a loop-nest representing the computation in a convolution layer of a CNN. Source: J. Cong and B. Xiao, Minimizing Computation in Convolutional Neural Networks.

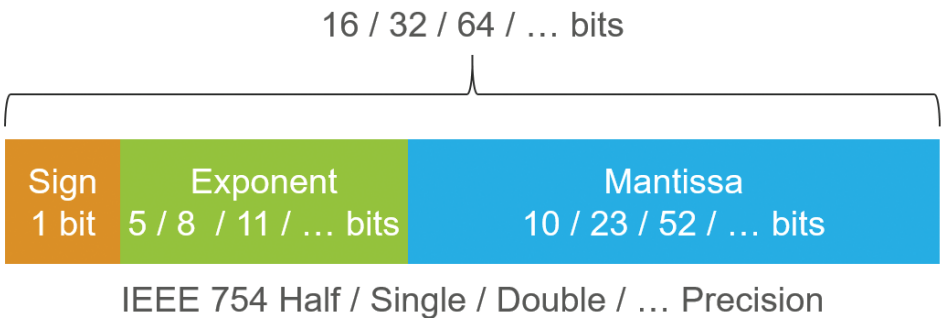

Figure 4. The binary representation of IEEE 754 floating-point numbers has three components: sign bit, exponent, and mantissa. The total number of bits available determines the precision of the floating-point number. *Source: OneSpin.*

verification tools have matured significantly. Formal applications (apps) that automate recurrent verification tasks have been crucial in driving widespread adoption in the industry. Nowadays, there are also formal apps targeting FP hardware verification (see Figure 5 below). They can automatically recognize and tackle complex FP arithmetic proofs, including for multiplication, once known to be a no-go zone for formal. Additionally, certain tools include a validated, IEEE-754 compliant model of FP arithmetic. The FP formal app can be applied to a variety of hardware implementations by engineers with limited expertise in formal and FP arithmetic.

At DVCon US 2018, Xilinx presented a paper on how its engineers verified FP hardware using an automated formal solution, and how the formal flow compared to simulation-based verification. Within days of effort, Xilinx engineers were able to find corner-case bugs and achieve exhaustive verification of 32-bit and 16-bit FP multiplication, type conversion functions between FP numbers of different precision, and other operations.

High-integrity AI chips

FP hardware is crucial to many modern AI chips and heterogeneous compute platforms. Digital electronics engineers are well aware of how hard it is to design and verify FP hardware. Chips for high-integrity applications, where safety and security are also key factors, have additional, and ever stricter, assurance requirements.

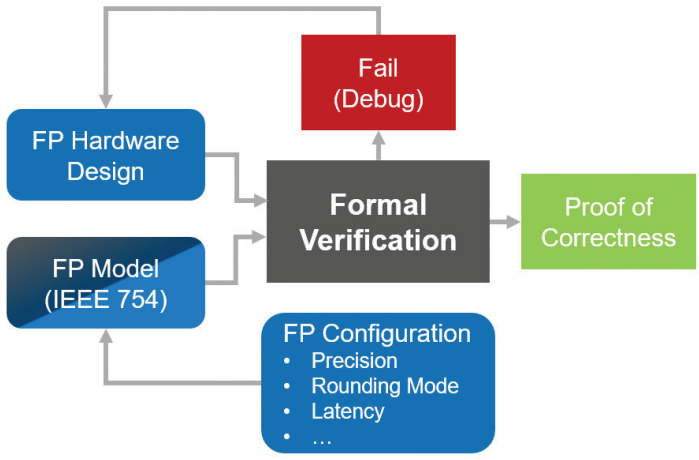

Figure 5. Modern formal verification tools can overcome complexity issues and mathematically prove that the floating-point (FP) hardware design always computes correct results. In addition, certain tools include a validated, IEEE 754-compliant model of floating-point arithmetic, thus minimizing the engineering effort required to achieve high-integrity AI chips. *Source: OneSpin*

In the past, big semiconductor companies have invested big bucks in formal verification. They recognized that formal methods were the only way to ensure that arithmetic hardware units would always compute correct results. In-house development of technology and methodology by numerous experts were necessary to overcome complexity issues.

Nowadays, with dozens of start-ups involved in the development of AI accelerators, it is crucial to provide easy-to-use formal verification solutions that can enable rigorous, exhaustive verification while also reducing development effort. Formal solutions that include an executable specification of IEEE 754 operations, paired with innovative proof technology, may overcome the traditional complexity issues of formal verification for FP arithmetic while also dramatically reducing engineering effort. For more information about formal verification of FP hardware, visit onespin.com/fpu

Editor's Note: OneSpin will feature its full complement of certified IC integrity verification solutions in Booth #2245 during ES Design West co-located at SEMICON West July 9-11 at San Francisco's Moscone Center.

About the author

Sergio Marchese is technical marketing manager at OneSpin Solutions. He has 20 years of experience in electronic chip design and deployment of advanced hardware development solutions across Europe, North America, and Asia. His expertise covers functional verification, safety standards, including ISO 26262 and DO-254, and detection of hardware Trojans and security vulnerabilities. He is passionate about enabling the next generation of high-integrity chips that underpin the Internet of Things, 5G, artificial intelligence, and autonomous vehicles.