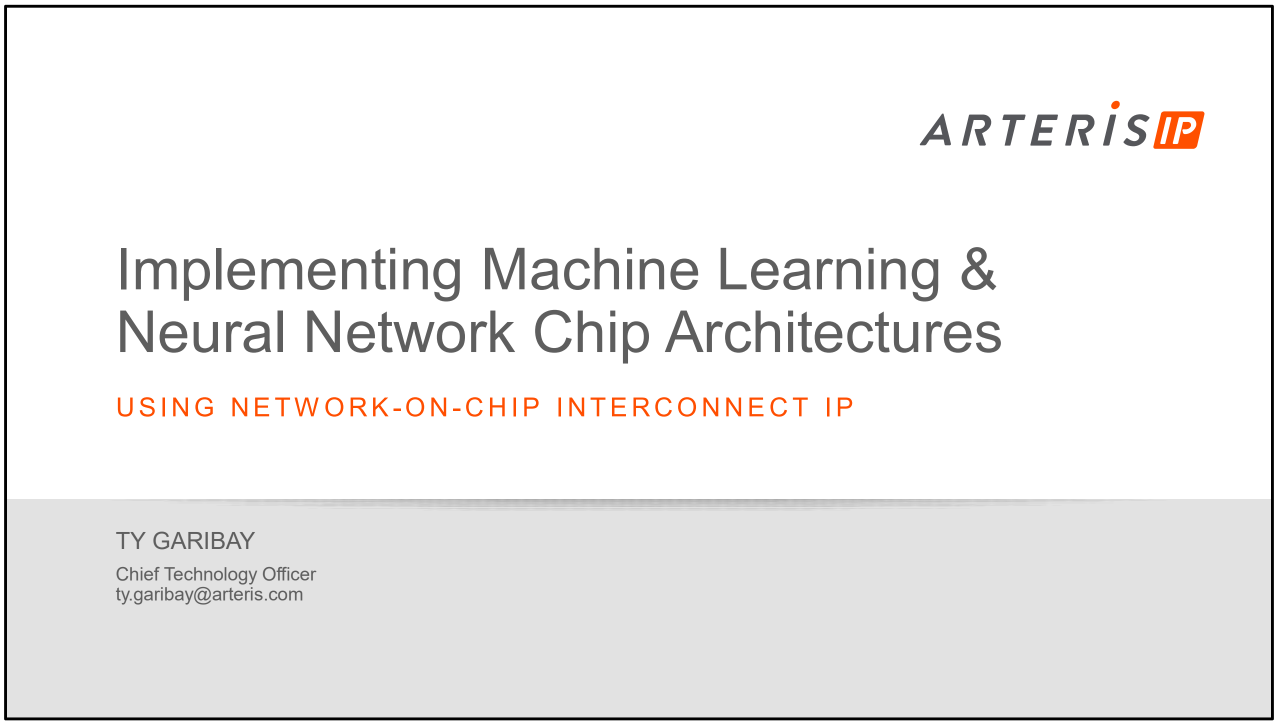


Implementing Machine Learning & Neural Network Chip Architectures

USING NETWORK-ON-CHIP INTERCONNECT IP

TY GARIBAY

Chief Technology Officer
ty.garibay@arteris.com

Title:
Implementing Machine Learning & Neural Network Chip Architectures Using Network-on-Chip Interconnect IP

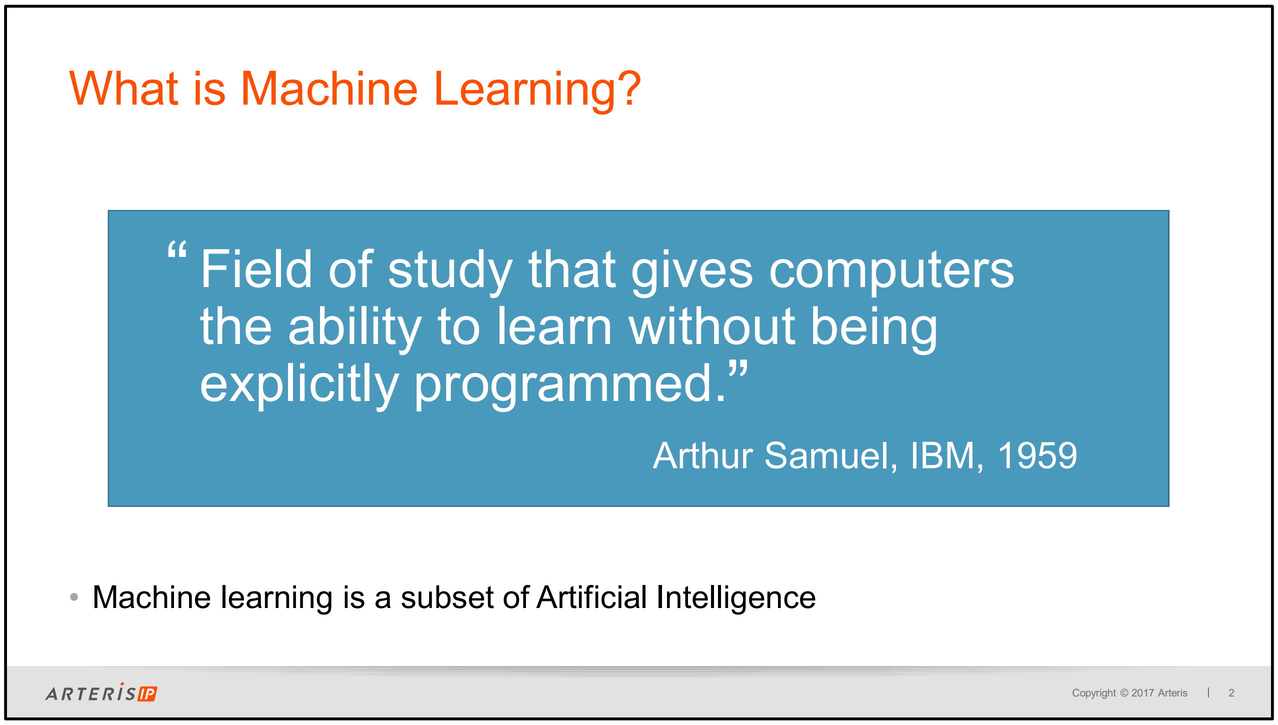
Primary author:
Ty Garibay, Chief Technology Officer, ArterisIP, ty.garibay@arteris.com

Secondary author:
Kurt Shuler, Vice President, Marketing, ArterisIP, kurt.shuler@arteris.com

Abstract:
A short tutorial and overview of machine learning and neural network chip architectures, with emphasis on how network-on-chip interconnect IP implements these architectures.

Time to read:
10 minutes

Time to present:
30 minutes



What is Machine Learning?

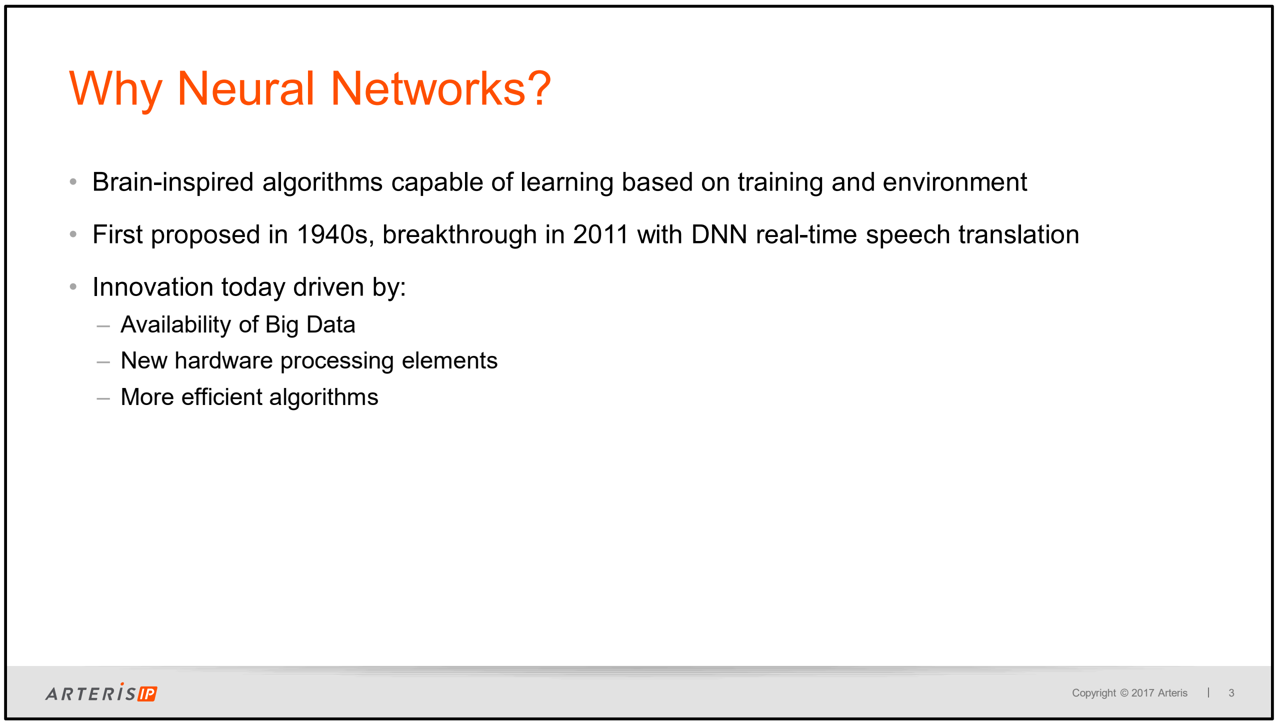
“Field of study that gives computers the ability to learn without being explicitly programmed.”

Arthur Samuel, IBM, 1959

- Machine learning is a subset of Artificial Intelligence

Copyright © 2017 Arteris | 2

- Machine learning is a subset of artificial intelligence, and explicitly relies upon experiential learning rather than programming to make decisions.
- The advantage to machine learning for tasks like automated driving or speech translation is that complex tasks like these are nearly impossible to explicitly program using rule-based if…then…else statements because of the large solution space.
- However, the “answers” given by a machine learning algorithm have a probability associated with them and can be non-deterministic (meaning you can get different “answers” given the same inputs during different runs)
- Neural networks have become the most common way to implement machine learning. Let’s learn more to understand why…



Why Neural Networks?

- Brain-inspired algorithms capable of learning based on training and environment
- First proposed in 1940s, breakthrough in 2011 with DNN real-time speech translation
- Innovation today driven by:
  - Availability of Big Data
  - New hardware processing elements
  - More efficient algorithms

Copyright © 2017 Arteris | 3

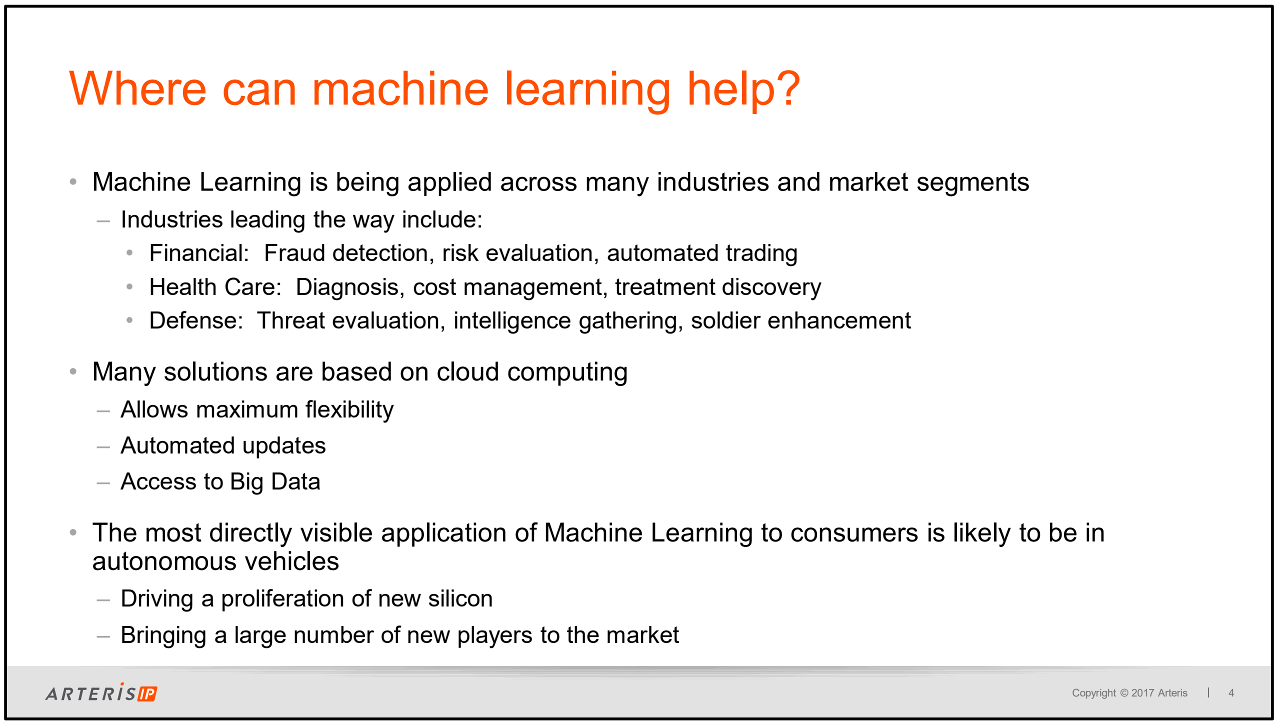
The idea of machine learning has been around since WWII, but hasn't become practical until recently. Innovations are:

- **Big Data** - Just like training the brain of a young child, training a neural network requires lots of data. The Internet and its data center backbones now provide a means to capture and store massive amounts of data from which neural networks can be trained and information can be inferred.

- **Powerful hardware** – There's been innovation not just in the scalability of x86 systems in data centers to train neural nets (NN), but also in the processing power and specialization of on-chip hardware elements that can more efficiently process certain NN algorithms. Chips using NN hardware accelerators are used in both the back-end data center NN training process, as well as the embedded, near real-time classification process.

- **New algorithms** – Computer scientists have created a plethora of new NN algorithms within the last 5 years that are targeted for various use cases. SoC architects tailor the use of these algorithms for whatever problems they are trying to solve, and tailor the SoC hardware architecture, including hardware accelerator choices and interconnect configuration, to solve the chosen problems most efficiently.

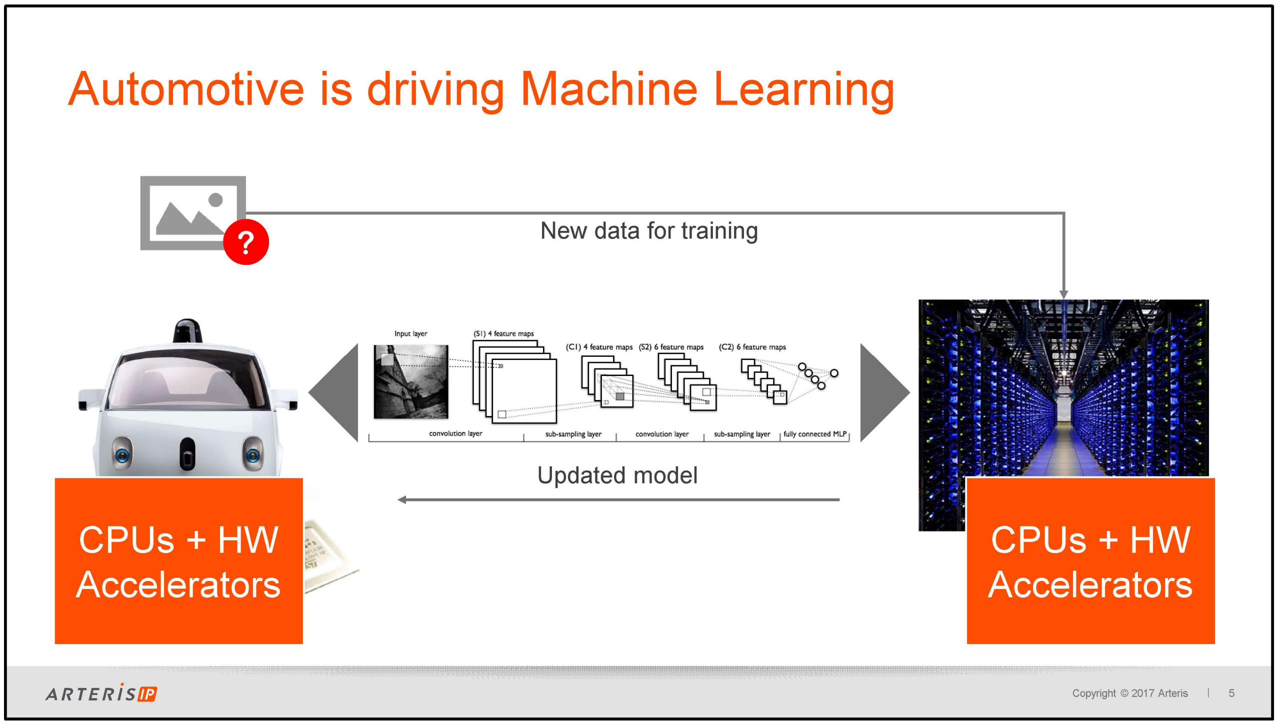
Before diving into the details of a neural network, let's look at an example of one being used in the real-world.



Where can machine learning help?

- Machine Learning is being applied across many industries and market segments
  - Industries leading the way include:
    - Financial: Fraud detection, risk evaluation, automated trading
    - Health Care: Diagnosis, cost management, treatment discovery
    - Defense: Threat evaluation, intelligence gathering, soldier enhancement
- Many solutions are based on cloud computing
  - Allows maximum flexibility
  - Automated updates
  - Access to Big Data
- The most directly visible application of Machine Learning to consumers is likely to be in autonomous vehicles
  - Driving a proliferation of new silicon
  - Bringing a large number of new players to the market

Copyright © 2017 Arteris | 4



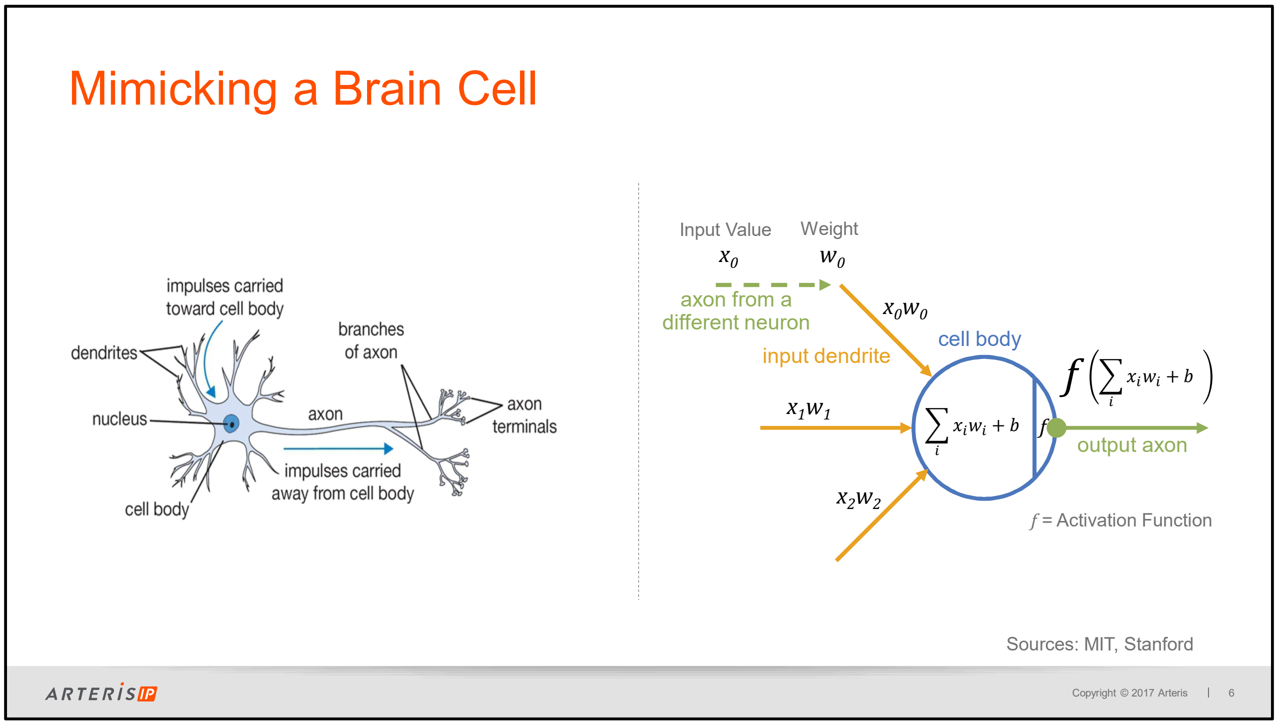
Automotive is driving Machine Learning

In this example, neural networks are being used in the datacenter to train neural networks running on the car. There is specialized accelerator hardware in the data center to train the network, and specialized hardware in the device that uses the trained neural net to classify objects.

When the car experiences something that doesn't make sense or is new, it sends that data to the data center so that it can be learned. The data center then updates the "weights" in the car's neural network.

I'll explain what "weights" are soon.

Now let's dive into the smallest element of this system, the neuron.



Mimicking a Brain Cell

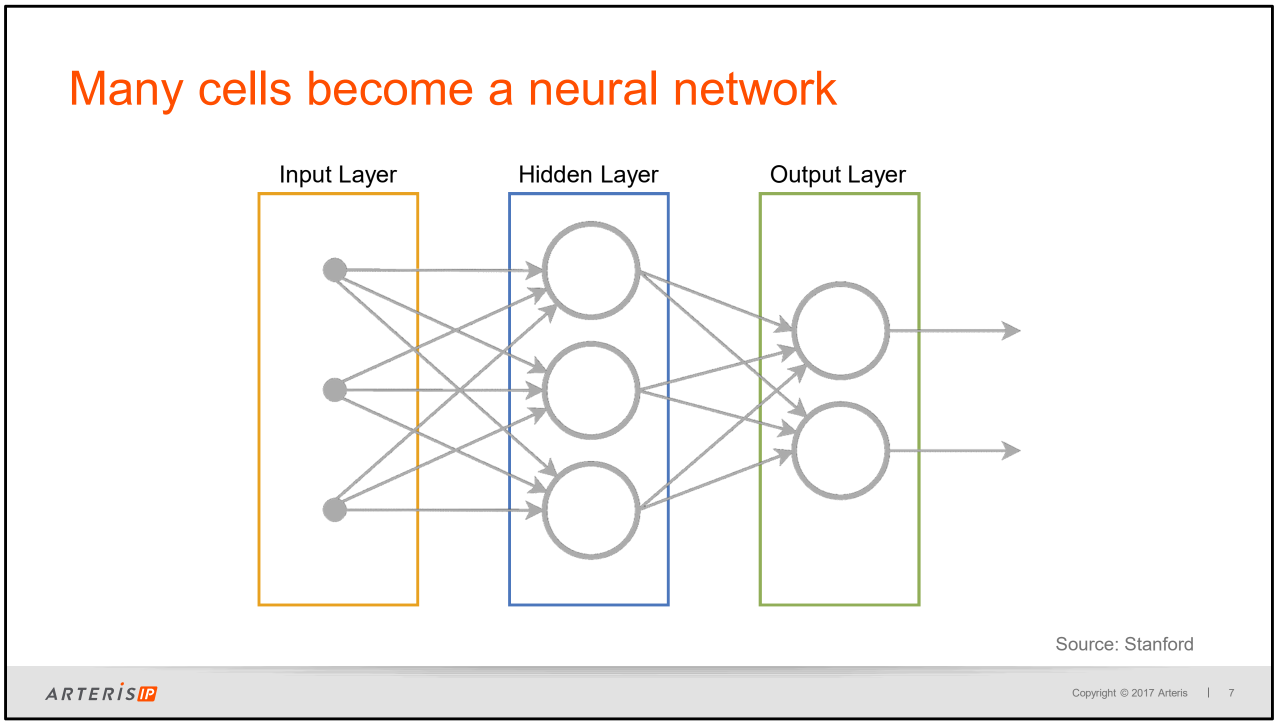
Sources: MIT, Stanford

On the left you see a brain neuron and on the right is its mathematical equivalent.

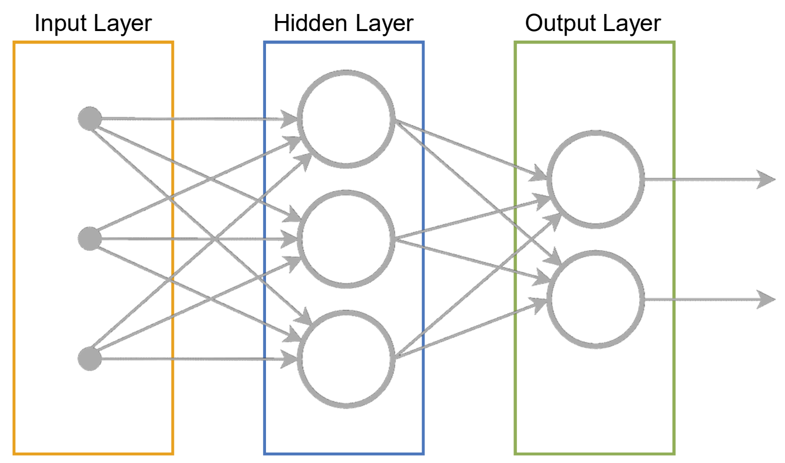
Neurons receive input signal from **dendrites** and produce output signal along the **axon**, which interacts with the dendrites of other neurons via synaptic weights (w_x), represented by the variable "w" on the right.

- The weights are what are learned. They control the amount of influence that the input dendrite has on the neuron's ability to "fire".
- The ability of the neuron to fire is determined by the **activation function**, which is related to the sum of the input values times weights for each input dendrite. A value above or below a certain threshold cause an output signal on the **output axon**.
- The output axon then connects to a dendrite of different neuron.

Now let's scale this up…



Many cells become a neural network

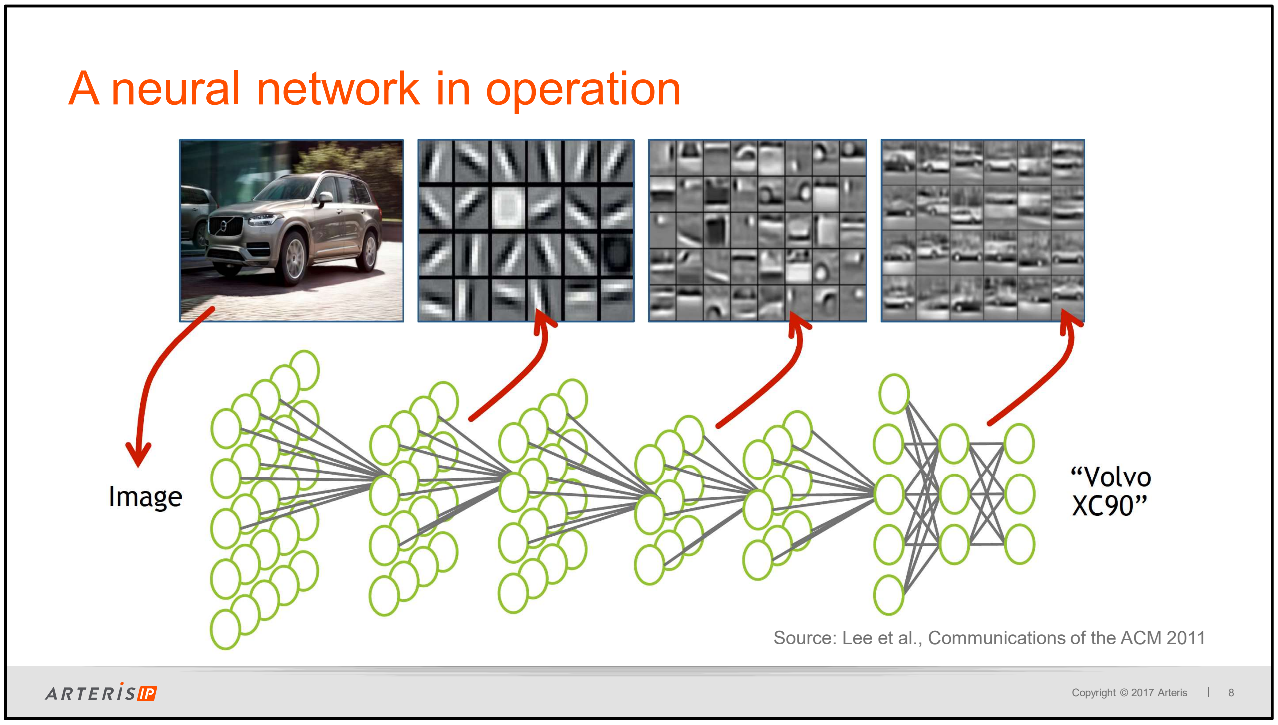


Source: Stanford

This picture is an example of simple feed-forward neural network. Each circle is a neuron and an arrow represents connections between neurons.

The weighted sum calculations we described before occur from left to right, until the neural network provides its outputs.

In traditional feed-forward neural networks, a hidden layer neuron is a neuron whose output is connected to the inputs of other neurons and is therefor not visible as a network output.



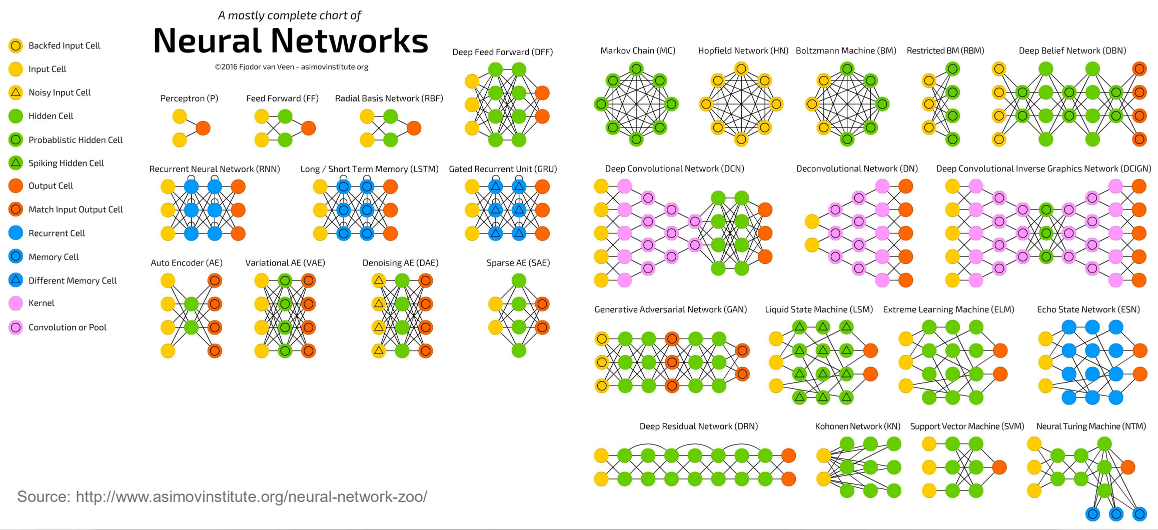
A neural network in operation

Source: Lee et al., Communications of the ACM 2011

When scaled up and trained, neural networks are capable of recognizing objects in a picture, and many other things.

The training process determines the weights for each neuron input. A neural network with a set of learned weights then performs "classification" on its inputs.

Many types with specialized processing requirements

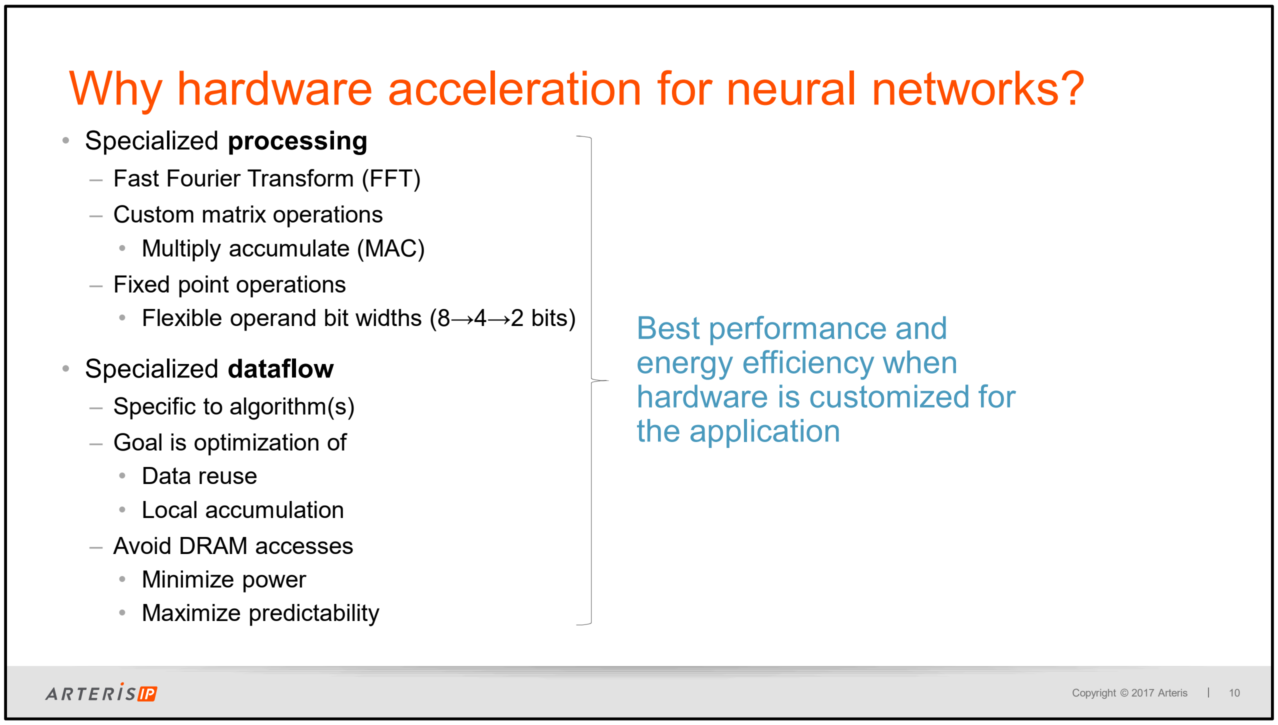

A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Source: http://www.asimovinstitute.org/neural-network-zoo/

ARTERIS IP Copyright © 2017 Arteris | 9

There are an infinite number of neural net possibilities, but this chart shows some classes of neural networks based on their neural functions and data flows (or topologies),

These functions and topologies are implemented in software and hardware using algorithms.



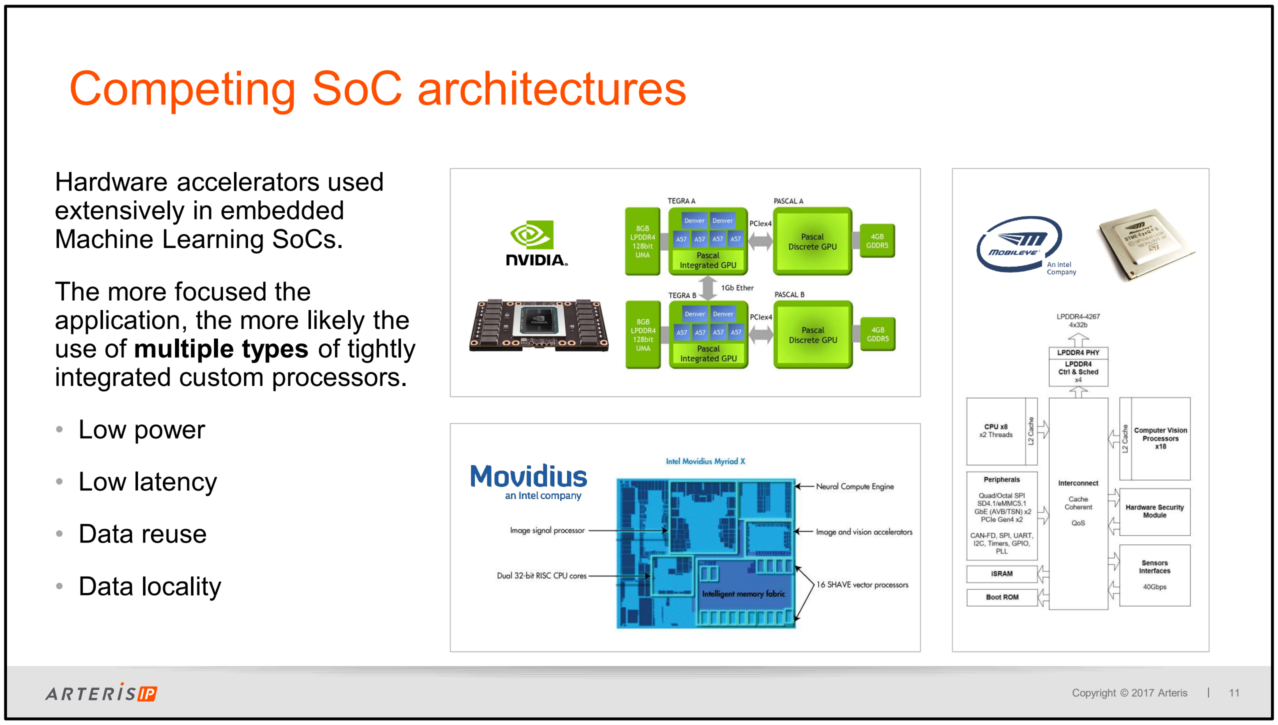
Why hardware acceleration for neural networks?

- Specialized **processing**
  - Fast Fourier Transform (FFT)
  - Custom matrix operations
    - Multiply accumulate (MAC)
  - Fixed point operations
    - Flexible operand bit widths (8→4→2 bits)
- Specialized **dataflow**
  - Specific to algorithm(s)
  - Goal is optimization of
    - Data reuse
    - Local accumulation
  - Avoid DRAM accesses
    - Minimize power
    - Maximize predictability

Best performance and energy efficiency when hardware is customized for the application

To put a neural network in a car that must meet real-time requirements, or in a data center that must use as little electricity and cooling as possible, it is necessary to hardware accelerate the algorithms.

The neural network implements specialized processing functions as well as dataflow functions. Chip designers compete by choosing what to accelerate, how to accelerate it, and how to interconnect that functionality with the rest of the neural net.

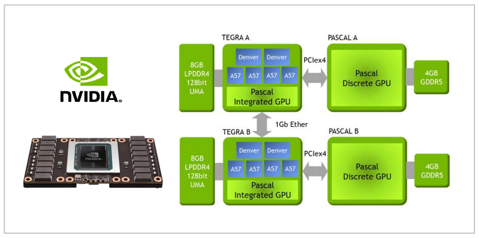


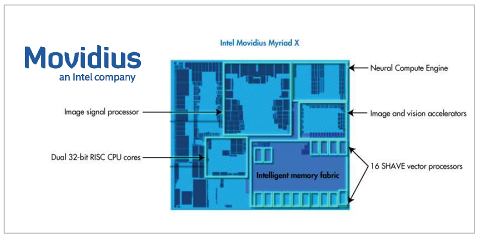
Competing SoC architectures

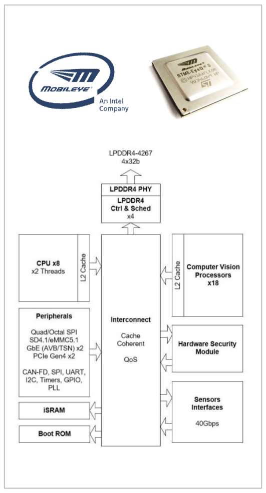
Hardware accelerators used extensively in embedded Machine Learning SoCs.

The more focused the application, the more likely the use of **multiple types** of tightly integrated custom processors.

- Low power
- Low latency
- Data reuse
- Data locality




Movidius
an Intel company
Intel Movidius Myriad X
Neural Compute Engine
Image signal processor
Image and vision accelerators
Dual 32-bit RISC CPU cores
16 SHAVE vector processors
Intelligent memory fabric

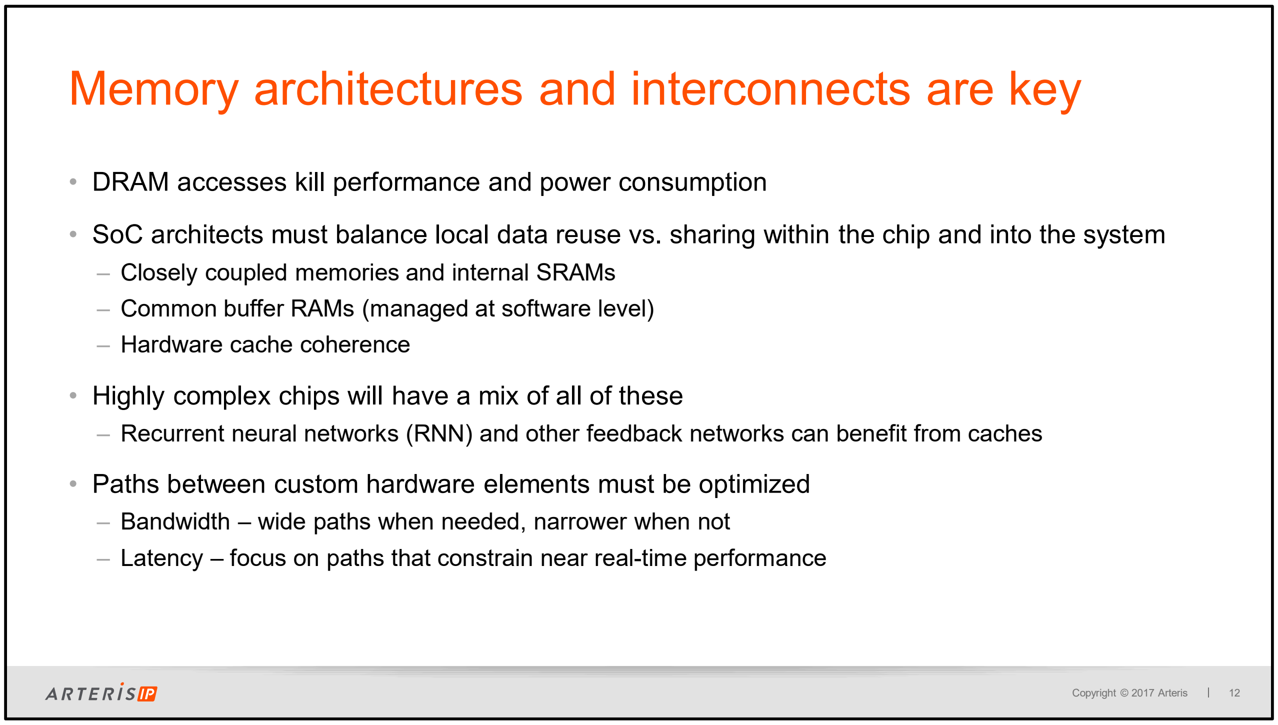


ARTERIS IP

Copyright © 2017 Arteris | 11

What has evolved are SoC architectures where more and more types and numbers of hardware accelerators are being added with each new version of a chip. In essence, the architects are slicing the algorithms finer and finer by adding more and more accelerators created specifically to increase the efficiency of the neural net.

Of course, as the types and numbers of processing elements increases, the interconnect and memory architecture connecting these elements becomes the critical path to success…



Memory architectures and interconnects are key

- DRAM accesses kill performance and power consumption
- SoC architects must balance local data reuse vs. sharing within the chip and into the system
  - Closely coupled memories and internal SRAMs
  - Common buffer RAMs (managed at software level)
  - Hardware cache coherence
- Highly complex chips will have a mix of all of these
  - Recurrent neural networks (RNN) and other feedback networks can benefit from caches
- Paths between custom hardware elements must be optimized
  - Bandwidth – wide paths when needed, narrower when not
  - Latency – focus on paths that constrain near real-time performance

Copyright © 2017 Arteris | 12

Architects of these systems must do everything they can to avoid slow and power hungry off-chip DRAM accesses. As a result, they implement various memory techniques that balance keeping data close to where it will be used, while also allowing the data to be reused if necessary.

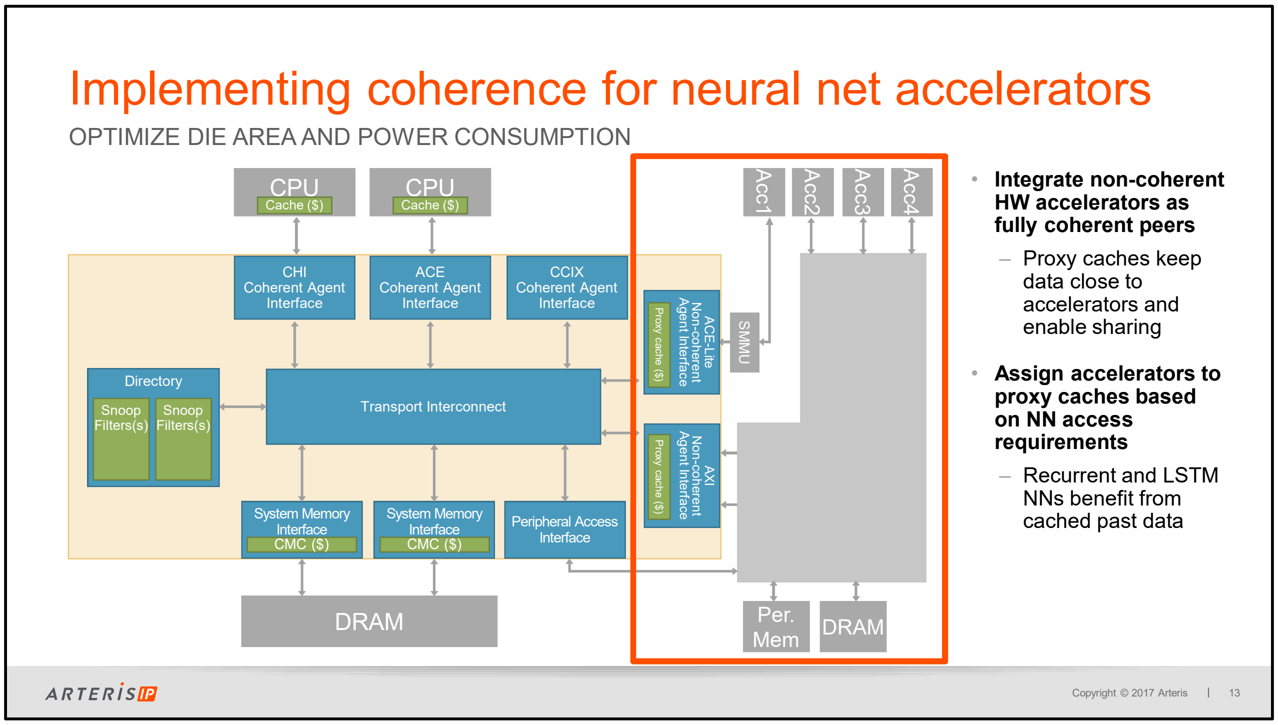
Three techniques that are often used, sometime on the same chip, include:

1. Memories that are closely coupled to a single processing element. These are most often implemented as internal SRAMs and are usually "transparent" to the running software.

2. Another approach is to provide RAM buffers that can be shared by multiple processing elements. Access to these are usually managed at the software level, which can lead to software complexity as the system scales up.

3. As systems become larger, it is often useful to implement hardware cache coherency. This allows processing elements to share data without the overhead of direct software management. Feedback neural networks that keep and use past state data, like RNNs, can benefit from the use of cache coherence.

No matter what the memory architectures used, the overall data flow must be optimized for QoS, meaning that bandwidth and latency requirements must be met. Architects ensure this by properly configuring their on-chip interconnect, whether non-coherent or cache coherent.



Implementing coherence for neural net accelerators

OPTIMIZE DIE AREA AND POWER CONSUMPTION

- **Integrate non-coherent HW accelerators as fully coherent peers**
  - Proxy caches keep data close to accelerators and enable sharing
- **Assign accelerators to proxy caches based on NN access requirements**
  - Recurrent and LSTM NNs benefit from cached past data

CPU
Cache ($)
CPU
Cache ($)
CHI Coherent Agent Interface
ACE Coherent Agent Interface
CCIX Coherent Agent Interface
Directory
Snoop Filters(s)
Snoop Filters(s)
Transport Interconnect
System Memory Interface CMC ($)
System Memory Interface CMC ($)
Peripheral Access Interface
DRAM
ACE-Lite Non-coherent Agent Interface
Proxy cache ($)
AXI Non-coherent Agent Interface
Proxy cache ($)
SMMU
Acc1
Acc2
Acc3
Acc4
Per. Mem
DRAM

ARTERIS IP   Copyright © 2017 Arteris | 13

Cache coherence is being implemented in neural network SoCs today. A novel approach is to integrate existing non-coherent processing elements into a cache coherent system using configurable proxy caches. The association, connectivity and behavior of each proxy cache can be optimized for use cases. In this diagram, the four (4) hardware accelerators act as coherent peers to the two ARM CPU clusters. In addition, processing elements connected to a proxy cache can efficiently share data with each other using that cache.

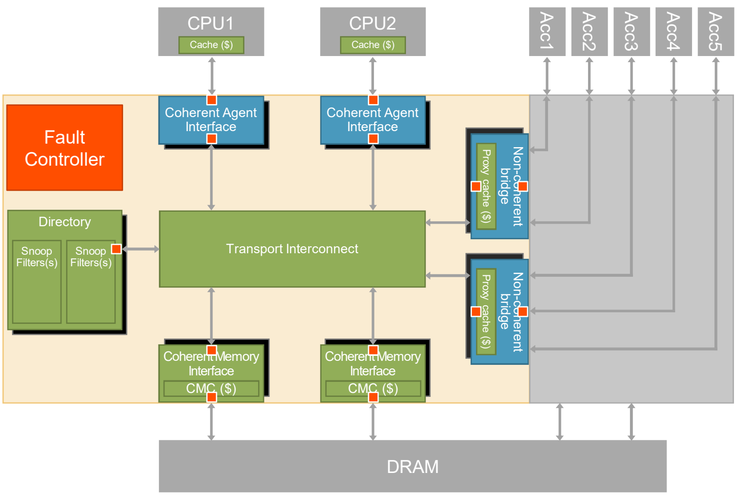
There are two (2) use cases where we are seeing use of proxy caches to allow non-coherent IP to participate as fully-coherent peers in the coherent domain:

1. Networking and storage – Lots of existing IP to be made coherent with the latest ARM processor clusters

2. Machine learning – Hardware accelerators for use within the datacenter for learning as well as the automobile for classification

Data flow efficiency is not the only challenge facing architects. What about data protection to ensure the safety of the system when it is part of an automobile or industrial robot?

Data flow protection for functional safety



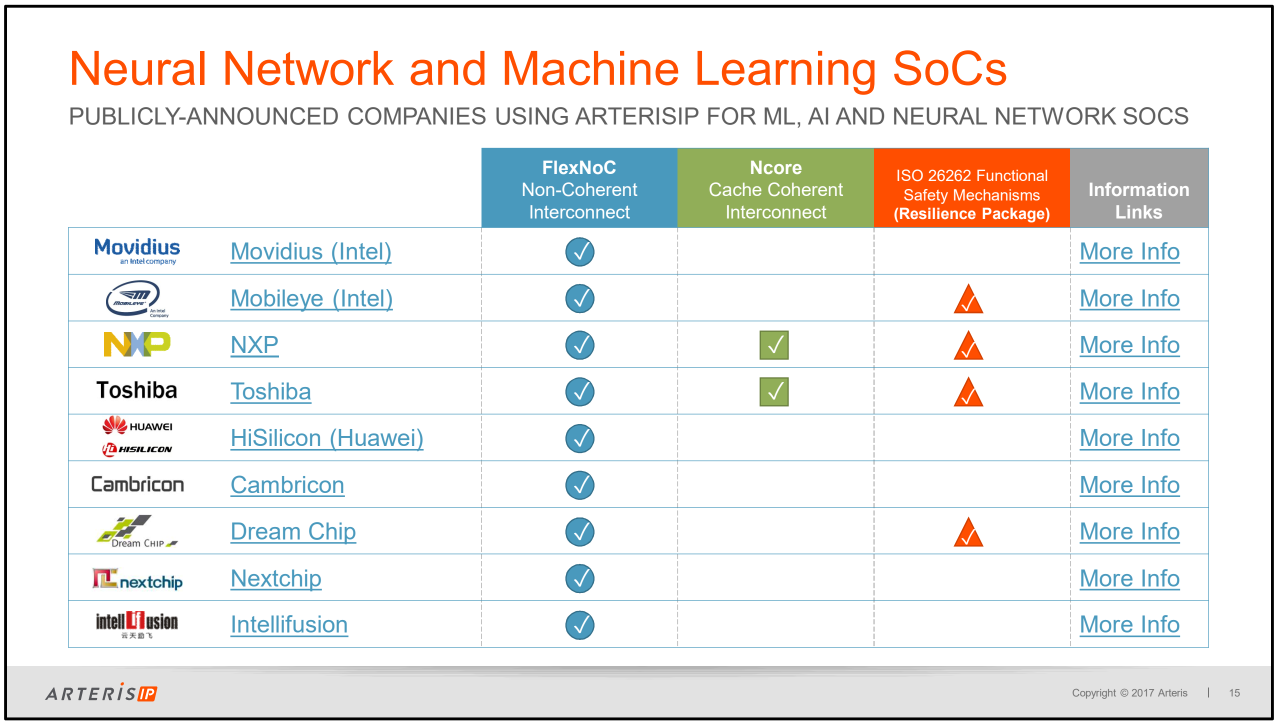
- **Data protection (at rest & transit)**
  - Parity for data path protection
  - ECC memory protection
- **Intelligent Ncore hardware unit duplication**
  - Don't duplicate protected memories or links
  - Do duplicate HW that affects packets
  - Integrate checkers, ECC/parity generators & buffers
- **Fault controller with BIST**

The SoC interconnect plays a big role in system functional safety because it "sees" all the data transferred on a chip. Ensuring the interconnect can find and in some cases fix errors improves the safety of the chip and improves diagnostic coverage to meet requirements for ISO 26262 automotive functional safety and other safety and reliability standards. To do this, we implement two sets of techniques:

1. **ECC and parity protection** for all data portions of the chip having safety or reliability requirements. This protection is configurable.

2. **Intelligent hardware duplication** of the parts of the interconnect that affect the content of a packet.

Both of these techniques do not impact performance, and they are both integrated with automatically generated ECC generators and data checkers. In addition, these safety mechanism are automatically integrated with a **configurable on-chip fault controller**, which manages interconnect-wide BIST functions and communicates error status to runtime software.



Neural Network and Machine Learning SoCs

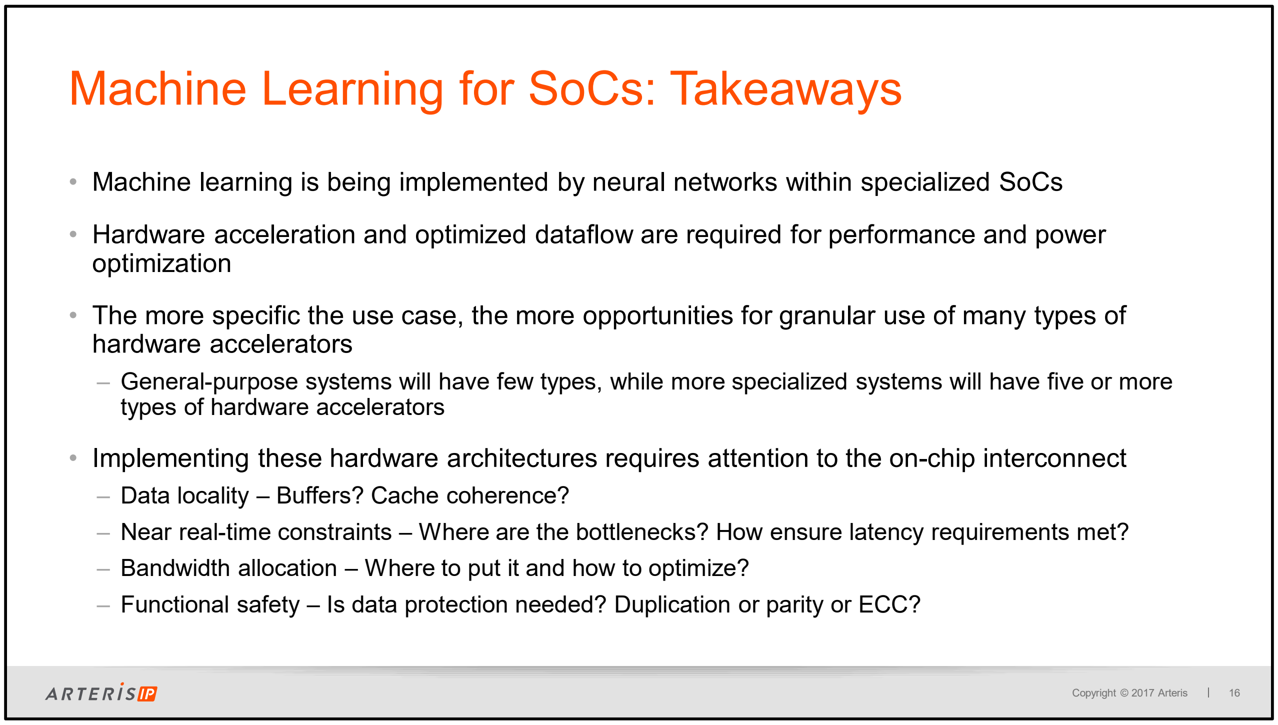
PUBLICLY-ANNOUNCED COMPANIES USING ARTERISIP FOR ML, AI AND NEURAL NETWORK SOCS

		FlexNoC Non-Coherent Interconnect	**Ncore** Cache Coherent Interconnect	ISO 26262 Functional Safety Mechanisms **(Resilience Package)**	**Information Links**
Movidius an Intel company	Movidius (Intel)	✓			More Info
Mobileye An Intel Company	Mobileye (Intel)	✓		✓	More Info
NXP	NXP	✓	✓	✓	More Info
Toshiba	Toshiba	✓	✓	✓	More Info
HUAWEI HISILICON	HiSilicon (Huawei)	✓			More Info
Cambricon	Cambricon	✓			More Info
Dream CHIP	Dream Chip	✓		✓	More Info
nextchip	Nextchip	✓			More Info
intellifusion	Intellifusion	✓			More Info

ARTERIS IP Copyright © 2017 Arteris | 15

ArterisIP is the most experienced interconnect IP company working with machine learning and neural network SoC design teams to create all kinds of architectures: Non-coherent, cache coherent, and functionally safe.

Here are some public examples of design teams who work with us to create these chips.



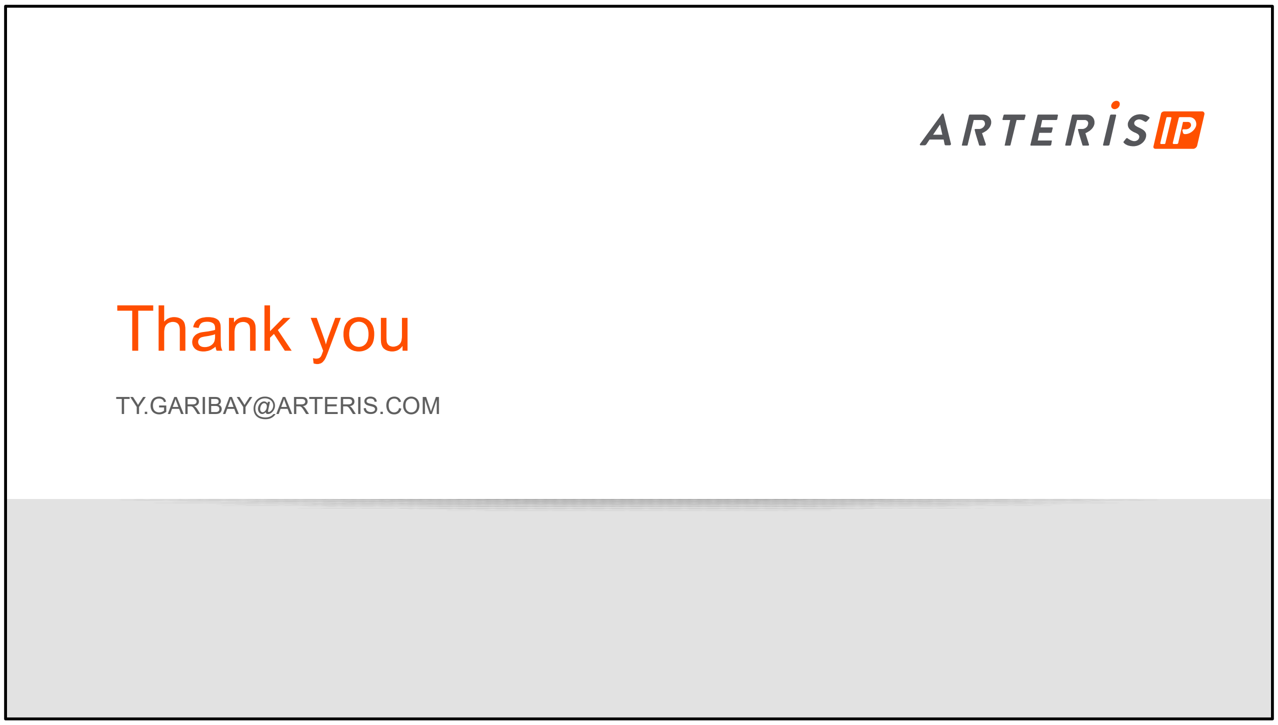
Machine Learning for SoCs: Takeaways

- Machine learning is being implemented by neural networks within specialized SoCs
- Hardware acceleration and optimized dataflow are required for performance and power optimization
- The more specific the use case, the more opportunities for granular use of many types of hardware accelerators
  - General-purpose systems will have few types, while more specialized systems will have five or more types of hardware accelerators
- Implementing these hardware architectures requires attention to the on-chip interconnect
  - Data locality – Buffers? Cache coherence?
  - Near real-time constraints – Where are the bottlenecks? How ensure latency requirements met?
  - Bandwidth allocation – Where to put it and how to optimize?
  - Functional safety – Is data protection needed? Duplication or parity or ECC?

ARTERIS IP Copyright © 2017 Arteris | 16

Machine learning requires hardware customization to just for algorithm acceleration but also for dataflow optimization.

Network-on-chip interconnect IP provides the means to meet not only functional, power and cost requirements, but also ISO 26262 functional safety and data reliability requirements, when required.



ARTERIS IP

Thank you

TY.GARIBAY@ARTERIS.COM

Please contact me at ty.garibay@arteris.com with any of your questions or comments!