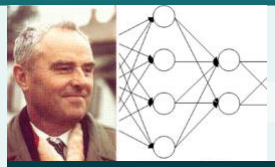
The Flex Logix InferX X1: Pairing Software and Hardware to Enable Edge Machine Learning

Randy Allen
Vice President of Software
Flex Logix Technologies

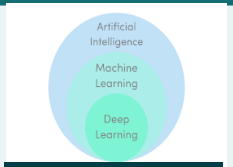
History of ML/AI

1952: ML coined



1965 Neural Networks

1997: Deep Blue



2006: Deep Learning

AI

Winter — Boom — Winter

Birth — Symbolic AI — Tedious engineering — Deep learning

ML

ML Winter

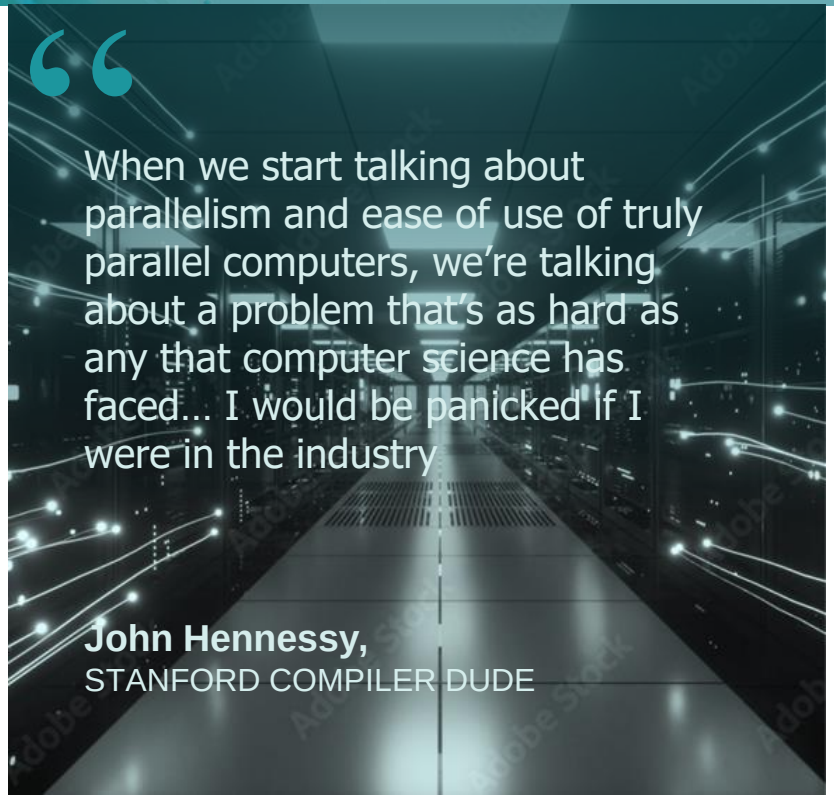
Legitimized Minis — Killer Micros — Ubiquitous GPUs

1950 1960 1970 1980 1990 2000 2010 2020

KNOWLEDGE DRIVEN — DATA DRIVEN

flexlogix AI + eFPGA

"It's not possible." "No, it's necessary."



Not a new challenge

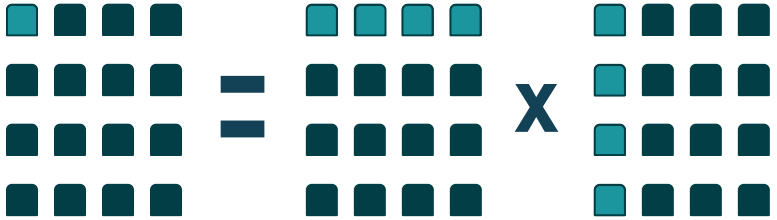
"Dead Parallel Computer Society"

Convex, Alliant, Multiflow, Encore, FPS, Inmos, Kendall Square, MasPar, nCUBE, Sequent, SGI, Thinking Machines ….

Matrix multiplication: a simple example

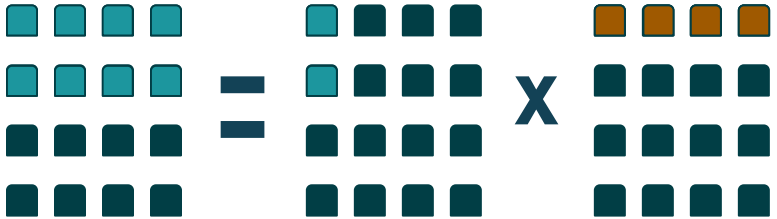
Scalar

```
for(i=0; i<n; i++)
    for(j=0; j<n; j++)
        for(k=0; k<n; k++)
            c[i][j] += a[i][k] * b[k][j];
```



Vector

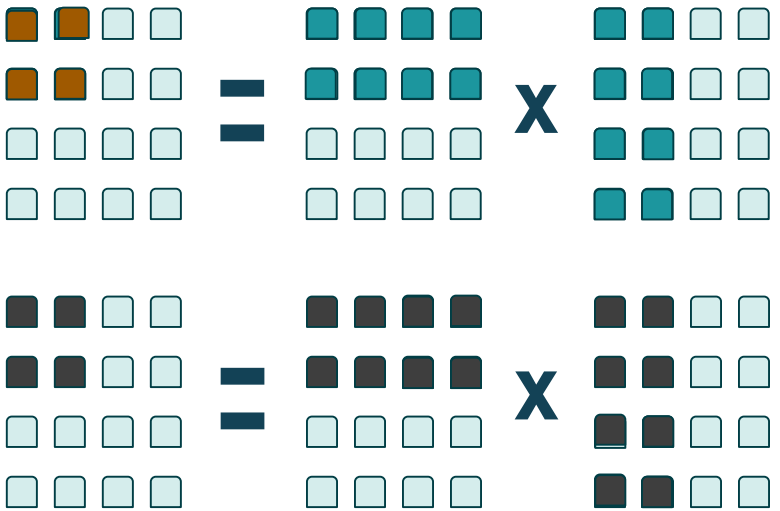
```
for(i=0; i<n; i++)
  for(js=0; js<n; js+=s)
    for(k=0; k<n; k++)
      for(j=js; j<js+s; j++) //vector
          c[i][j] += a[i][k] * b[k][j];
```



Matrix multiplication in parallel

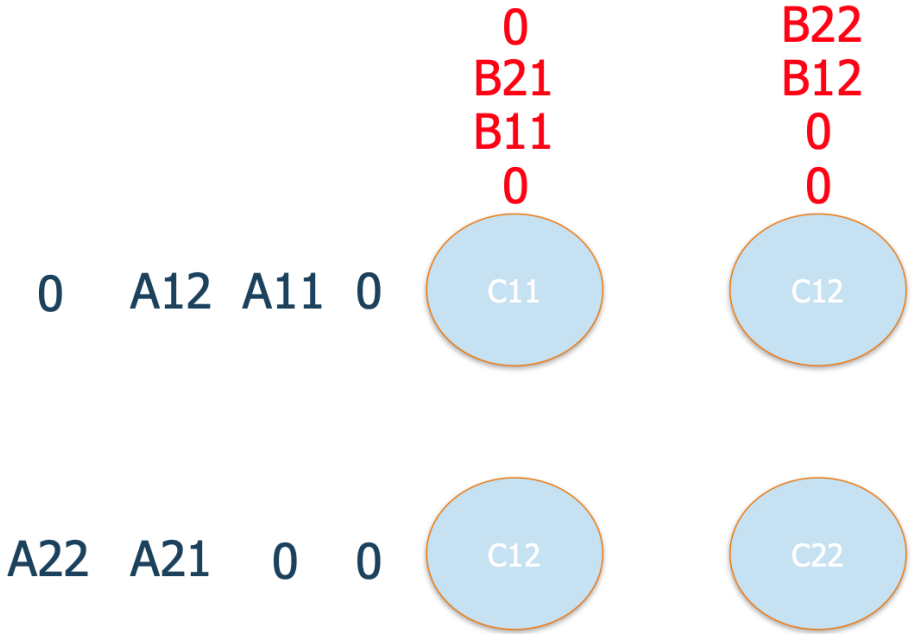
Scalar: Cache optimized

```
for(ib=0; ib<n; ib+=bi)
  for(jb=0; jb<n; jb+=bj)
    for(k=0; k<n; k++)
      for(i=ib; i<ib+bi; i++)
        for(j=jb; j < jb+bj; j++)
            c[i][j] += a[i][k] * b[k][j];
```

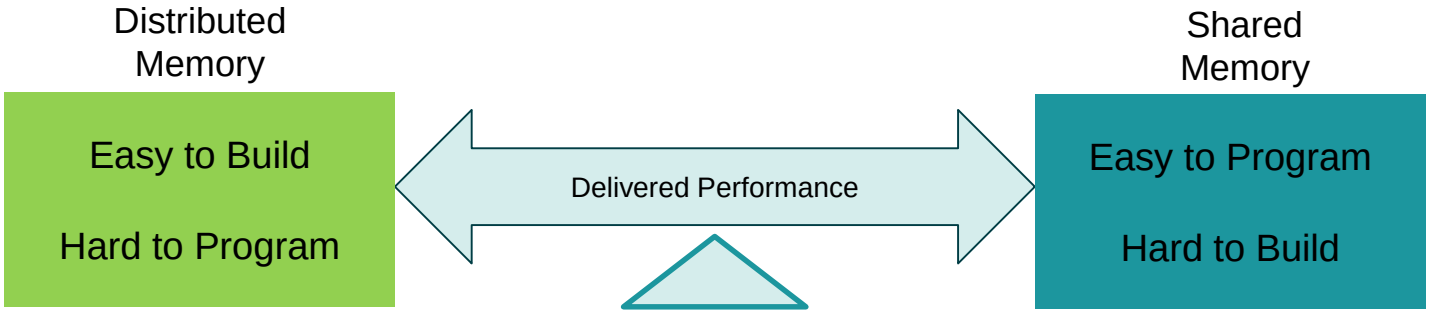


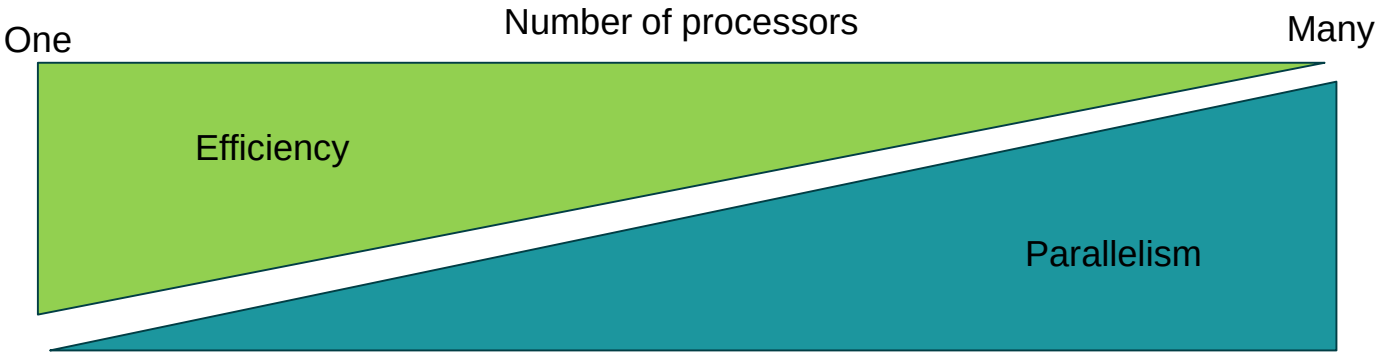
Systolic array

```
for(i=0; i<n; i++)
  for (j=i; i>0; i-){
     for (k=i; i>1; k-) {
          ta(j,k) = ta(j,k-1);
          tb(k,j) = tb(k-1,j);
     }
  }
  it = i;
  for (j=0; j<i; j++){
      ta(j,i) = a(j,t);
      tb(1,j) = b(t,j);
      it = it - 1;
  }
  for (j=0; j<i; j++)
     for k=0; k<i; k++)
        c(j,k) += ta(j,k) * tb(j,k);
```



Achieving balance





The real challenge of AI is

NOT

- Packing more processors onto a die than anyone else in the world
- Creating the fastest processor synchronization in the world
- Designing a large ({distributed, fetch-and-phi, shared,}) memory system

BUT INSTEAD

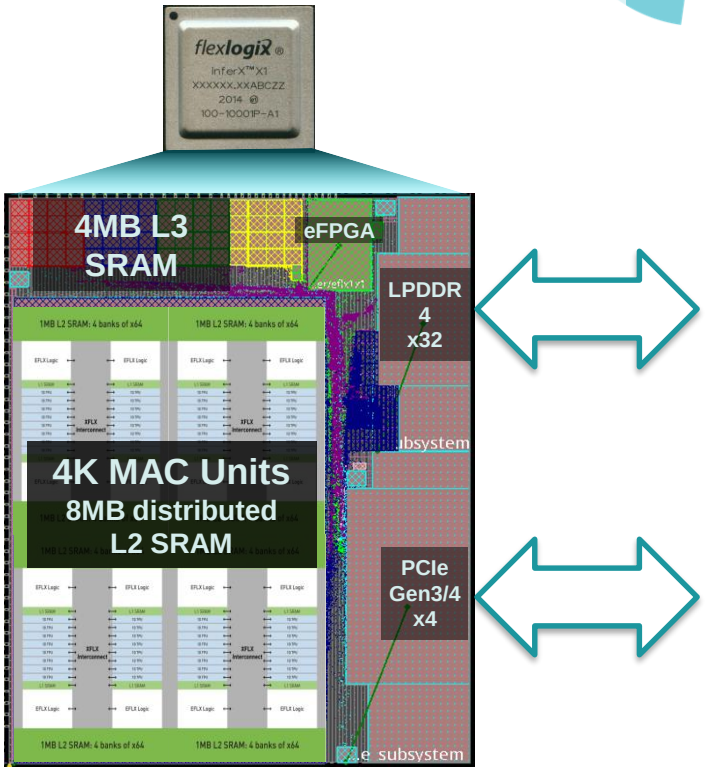
- Designing a multiprocessor system with a balance between processors and memory
- With the right set of specialized instructions for acceleration
- ***In conjunction*** with software that can effectively utilize the system

BECAUSE

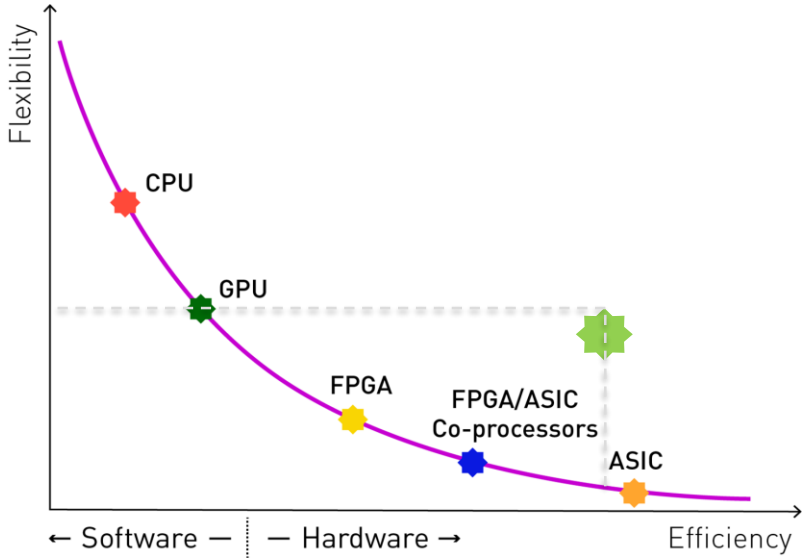
- More processors are useful only if the software can compile the net to use them
- Eliminating the need for synchronization is far more effective than faster synchronization
- The key to fast processing is reusing data, not fetching it quickly

The Flex Logix InferX™ X1

- Dynamic TPU Array
- Accelerator/Co-processor for host processor
- ASIC performance but dynamic to new models
- Low power/High performance
- Designed for edge (B=1) applications



ASIC efficiency with CPU/GPU flexibility



2019: Xin Feng, Computer vision algorithms and hardware implementations: A survey InferX position added by Flex Logix

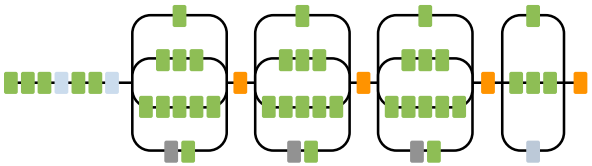
InferX Technology

- Dynamic Tensor Processing
- Highly Silicon Efficient
- Programmed via standard AI Model Paradigms (TensorFlow, PyTorch, ONNX)
- Layer by Layer reconfiguration in microseconds

System level view of software

NN Model Framework

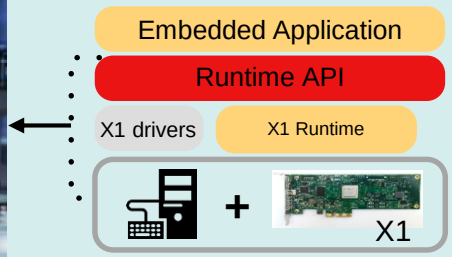
Conv
AvgPool
MaxPool
Concat



TensorFlow

Neural Net

Embedded App at The Edge



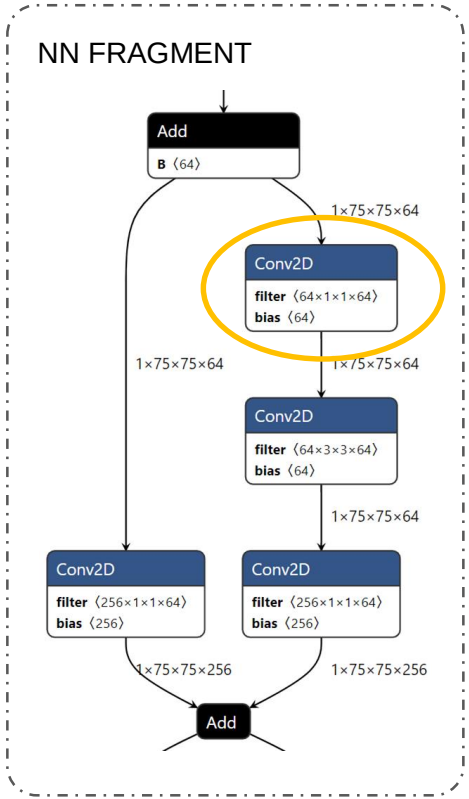
.ncf

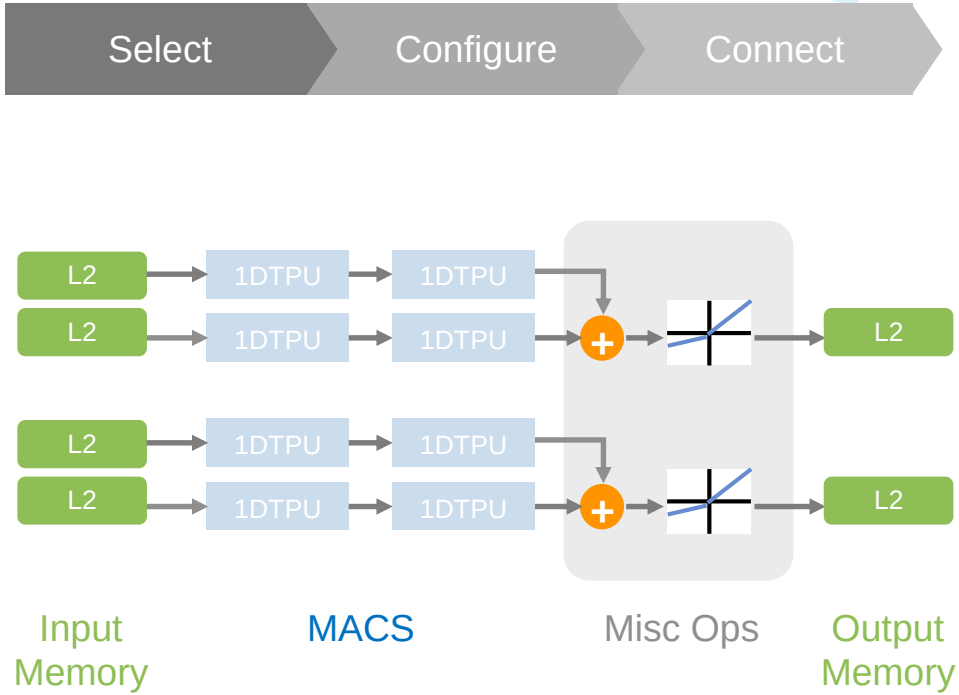
InferXDK Software Development Toolkit

InferX compiler: unleashes the power of X1

SELECTION OF OPERATORS

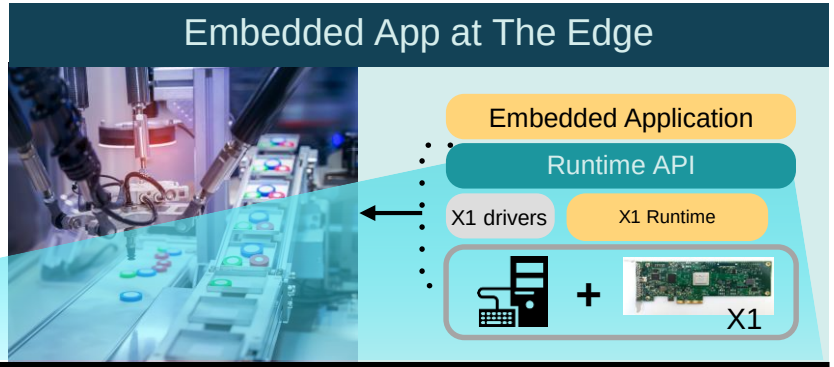
Add
Convolution
AvgPool
MaxPool
Affine
Concatenate
….





Runtime software API

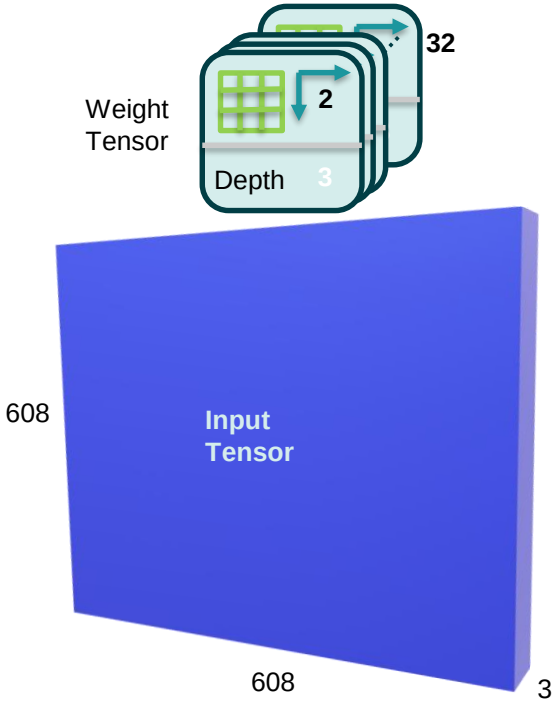
Simple API used to couple host application to the inference processing that is offloaded to the InferX X1 co-processor



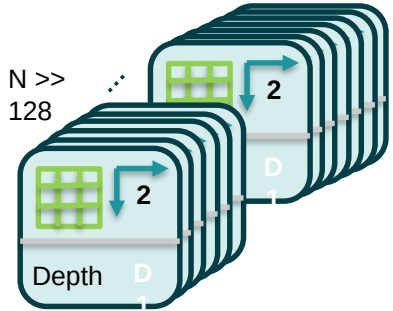
INFERX RUNTIME C++ API

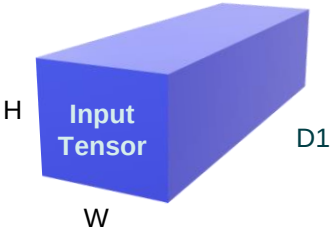
FUNCTIONS	DESCRIPTION	ARGS
StartInference	Executes single inference	Data_handle
InferMulti	Executes multiple inferences	times
WaitForInferenceReady	Waits for inference to finish	{ }
GetInferOutputData	Retrieves inference results	data_out
SetModel	Loads model into memory	filename
SetData	Established Data	data
Initialize	Initialize X1 HW	{ }

Assuring high quality performance

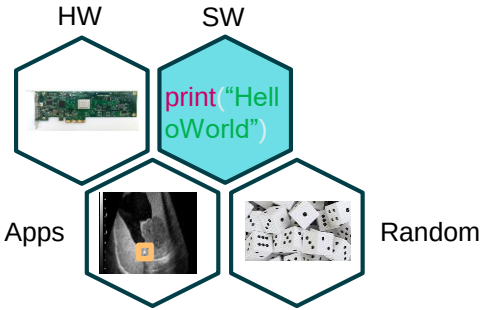


HW configuration depends on Conv2D shape





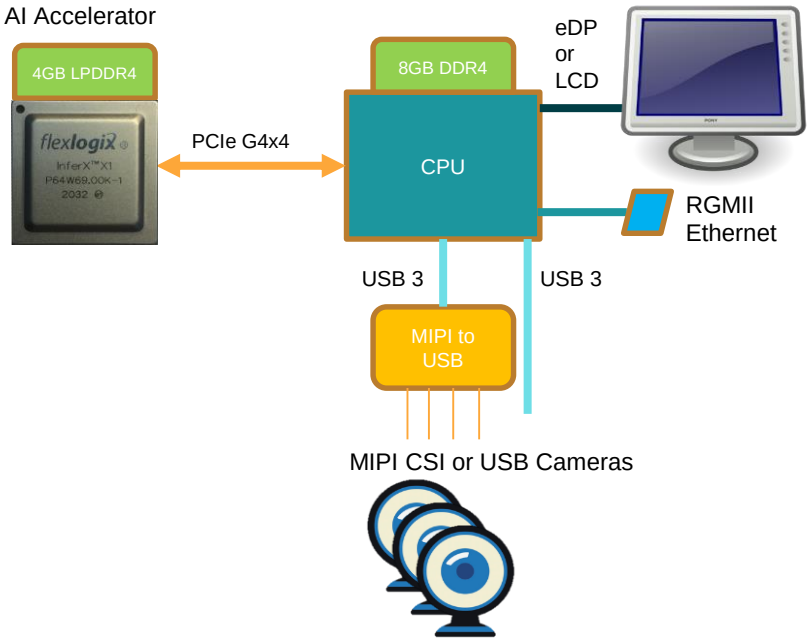
Cross Functional Exploration



Capturing All Attributes

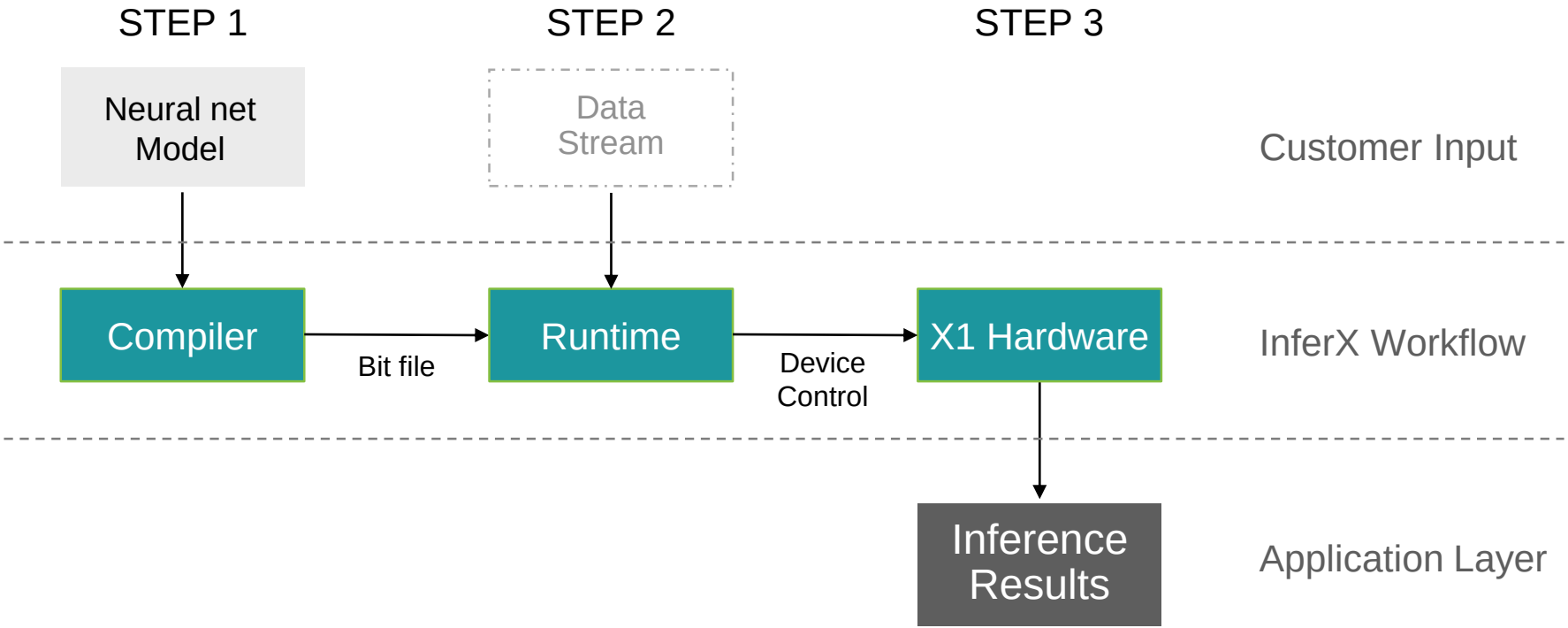
- Input/Output Quantization
- Activation Functions
- Padding

Accelerating AI for embedded processing



- Retains customer programming model
  - Does not force customer to move to Arm or Linux with SoC approach
  - Minimize time to market and development cost
  - Only need Add-in PCIe or M.2 card plus drivers
- Reduces dependence on rapid interface technology evolution
  - Accelerator uses standard PCIe, DRAM
  - Leave Camera and Display interface selection/evolution to Host
- Enables R&D and silicon focus on acceleration processing
  - Dedicate more silicon to high leverage acceleration logic
  - Reduced engineering complexity

Minimal effort required to go from model to inference



Putting it all together

Can help you gain a competitive advantage

Product Impact	Hardware	Compiler SW	Runtime SW
• Optimized solutions around your use cases	• Future proofed • GPU flexibility with ASIC performance • Low power	• Robust • Maximum performance • Minimal user intervention	• The InferX Runtime API is streamlined for ease-of-use

Contact Us

Inferx @flex-logix.com

Visit us on the expo floor for more information

Thank you