

Официальный партнер Xilinx в России

Компания МАКРО ГРУПП

Владимир Викулин,
Инженер по
применению Xilinx

Нейронные сети и искусственный интеллект на платформе Xilinx



27,28.03.2019 г.



План вебинара

- И.И. Задачи для решения методами И.И.
- Что такое нейронная сеть, как она устроена и в каких режимах функционирует. Типы нейронных сетей. Что такое машинное обучение.
- Программное обеспечение для работы с нейронными сетями и облачные сервисы
- Методы повышения производительности нейронных сетей.
- Технология работы с серверными платформами Xilinx.
- Технология работы с оконечными устройствами Xilinx
- Перспективы развития систем ИИ на платформе Xilinx
- Пример работы нейронной сети распознавания дорожных знаков на плате ZCU102.



1. И.И. (1) Задачи, решаемые с помощью ИИ

- Искусственный интеллект – направление информатики, изучающее и разрабатывающее методы решения задач интеллектуального (творческого) характера с использованием различных технических средств - компьютеров, программируемой логики и др.
- Типовые задачи:
 - Распознавание образов
 - Перевод текстов
 - Анализ данных, поиск закономерностей, экспертные заключения
 - Самообучение
 - Замена человека в контуре управления
 - ...



1. И.И. (2) Области применения

- Финансы
 - Анализ рисков
 - Высоочастотный трейдинг
- Компьютерные технологии
 - Машинный перевод
 - Распознавание речи, Понимание речи, общение с человеком на естественном языке
 - Распознавание образов, анализ изображений
- Управление сложными устройствами и системами
 - ADAS для автомобилей
 - Управление роботами
- ...



1. И.И. (3) Классификация

- Конвекционные
 - *Экспертные системы*: программы, которые, действуя по определенным правилам, обрабатывают большое количество информации, и в результате выдают заключение или рекомендацию на ее основе
 - *Рассуждение по аналогии* (Case-based reasoning).
 - *Байесовские сети доверия*: вероятностные модели, представляющие собой систему из множества переменных и их вероятностных зависимостей
 - *Поведенческий подход*: модульный метод построения систем ИИ, при котором система разбивается на несколько сравнительно автономных программ поведения, которые запускаются в зависимости от изменений внешней среды



1. И.И. (4) Классификация

- Вычислительные
 - **Нейронные сети** – модель вычисления на графах (появилась в результате математического моделирования структур мозга)
 - *Нечеткие системы*: методики для рассуждения в условиях неопределенности
 - *Эволюционные вычисления*: модели, использующие понятие естественного отбора, обеспечивающего отсеивание наименее оптимальных согласно заданному критерию решений.



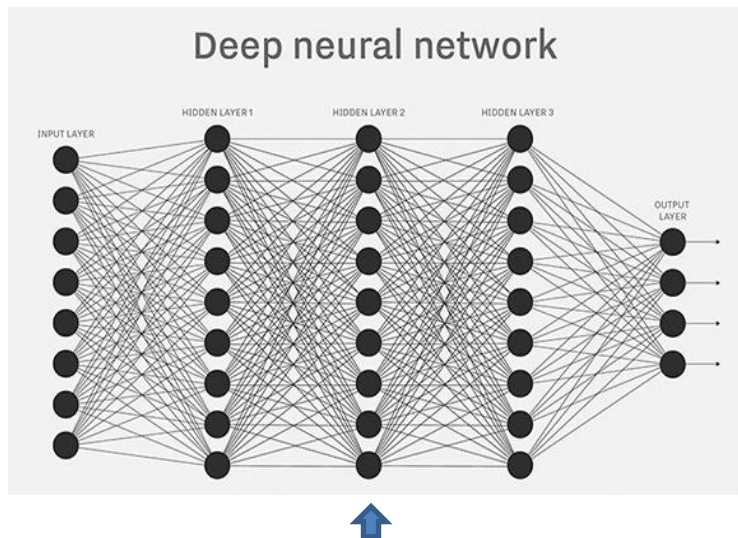
2. Нейронные сети (1)

- **Нейронные сети** – наиболее интенсивно развивающийся в данный момент метод построения систем ИИ
- Появились в результате попытки моделирования структур головного мозга, но далеко ушли от предмета изучения
- Способны к обучению и самообучению
- После обучения способны осуществлять классификацию для конечного числа вариантов. На данный момент в ряде областей по точности превосходят человека
- Мотивы предпочтения одного варианта другому не поддаются выяснению

2. Нейронные сети (2)

Общая структура Н.С.

Входной слой
(напр.
двумерное
изображение)



Выходной слой
(каждый кружок –
возможный
вариант)

Напр:

- Кошечка
- Собачка
- Человек
- Неведома зверюшка
- Все остальное



Внутренние слои и связи

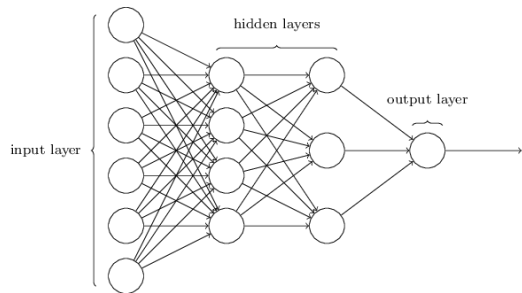
(Благодаря отсутствию обратной связи можно обеспечить
детерминированное время отклика)

2. Нейронные сети (3)

Разновидности НС

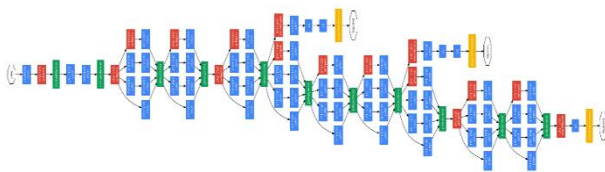
Multi-Layer Perceptron

- Classification
- Universal Function Approximator
- Autoencoder



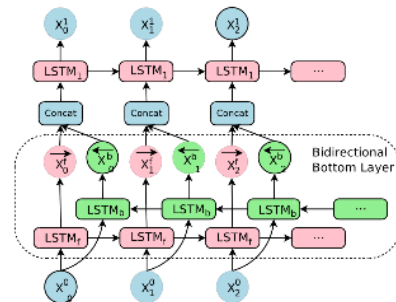
Convolutional Neural Network

- Feature Extraction
- Object Detection
- Image Segmentation



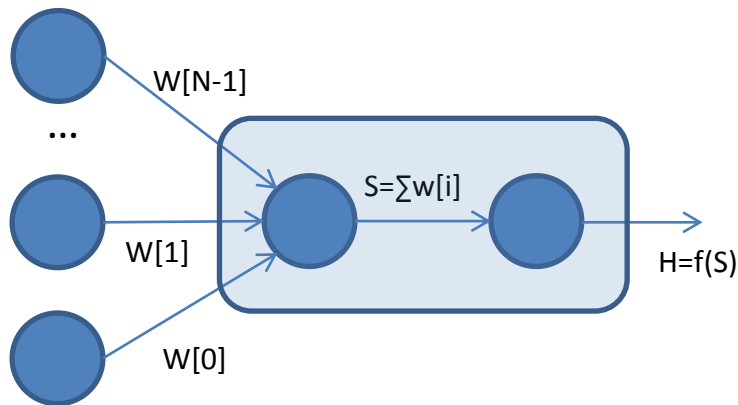
Recurrent Neural Network

- Sequence and Temporal Data
- Speech to Text
- Language Translation



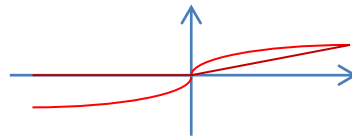
2. Нейронные сети (4)

Основной структурный элемент Н.С. - перцептрон (аналог нейрона)



Выход H

1. $f(S)$ – нелинейная функция
2. Выход H нормирован в диапазоне $[-1..+1]$



2. Нейронные сети (5)

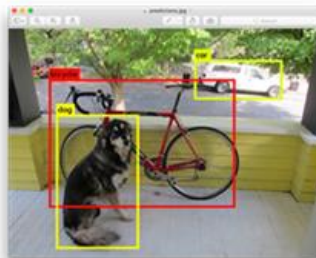
Типовые задачи для (сверточных) НС

Классификация



“Собака”

Обнаружение



Сегментация



2. Нейронные сети (6)

Этапы работы с сетью

	Этап	Действия	Прим.
1	Создание	Структура, кол-во слоев и т.д.	Нет четких правил
2	Обучение	Настройка параметров (послойно) на размеченной фиксированной входной последовательности	Чрезвычайно ресурсоемкий, требуется float point
3	Тестирование	Проверка на последовательностях, неизвестных сети при обучении	
4	Оптимизация	Необязательный (но важный)	
5	Применение	Обработка реальных данных	



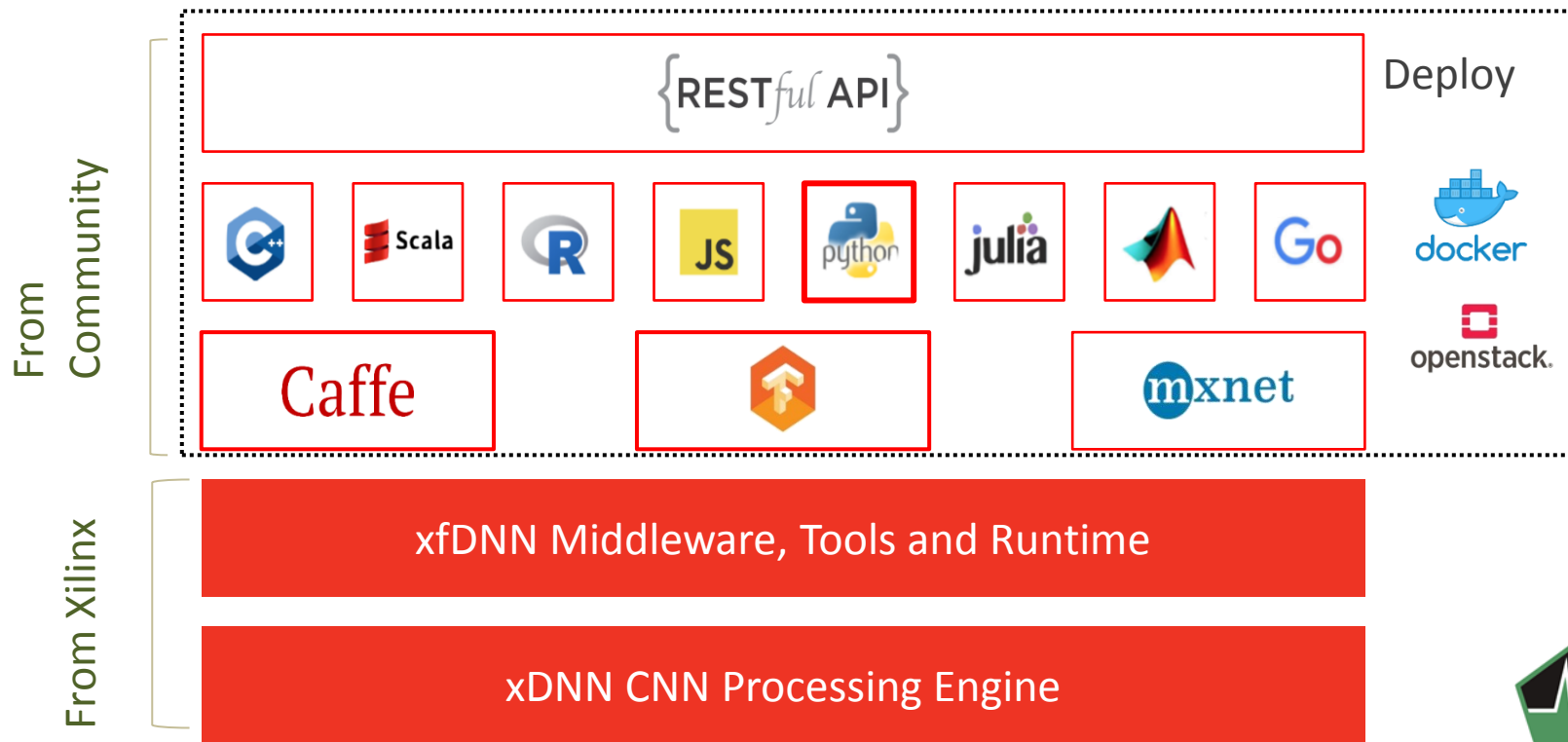
2. Нейронные сети (7)

Для тех, кто хочет занырнуть поглубже:



3. Средства для работы с НС

Стек технологий Xilinx



3. Средства для работы с НС (2)

Инфраструктура работы с НС

Язык разработки	Python
Библиотеки	TensorFlow, Keras ...
Модели	Конфигурации сетей
Датасеты	Cifar ...
Фреймворки	Cafe, Jupiter Notebook ...
Аппаратные ускорители	Nvidia GPUs

Подробнее [здесь](#)



4. оптимизация НС (1)

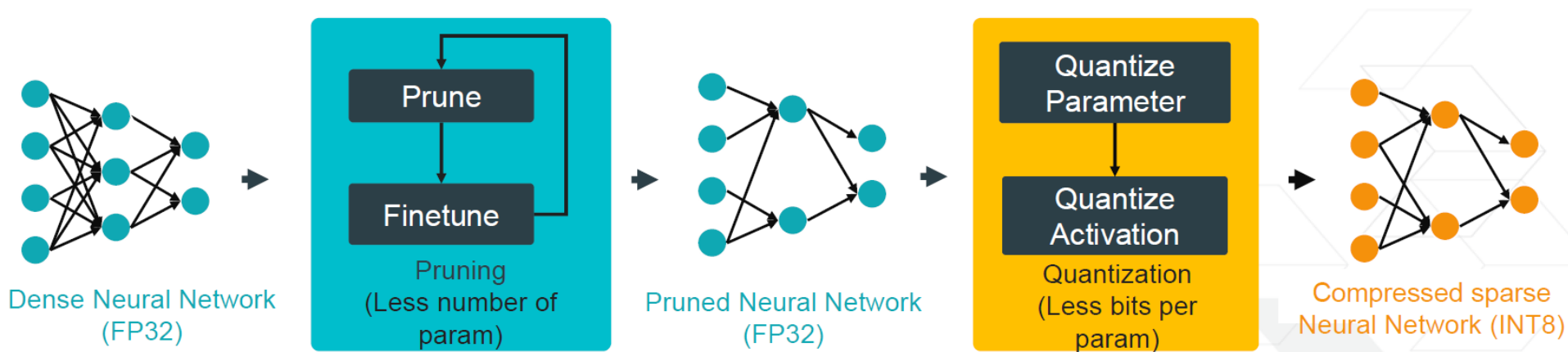
Режимы - обучение и работа (inference)

Критерии	Обучение	Работа
Характеристики	Точность	- Скорость работы - Скорость реакции - Реальное время
Вычислительные ресурсы	Огромные	Средние
Вычисления с FP	Обязательны	Нет



4. оптимизация НС (2)

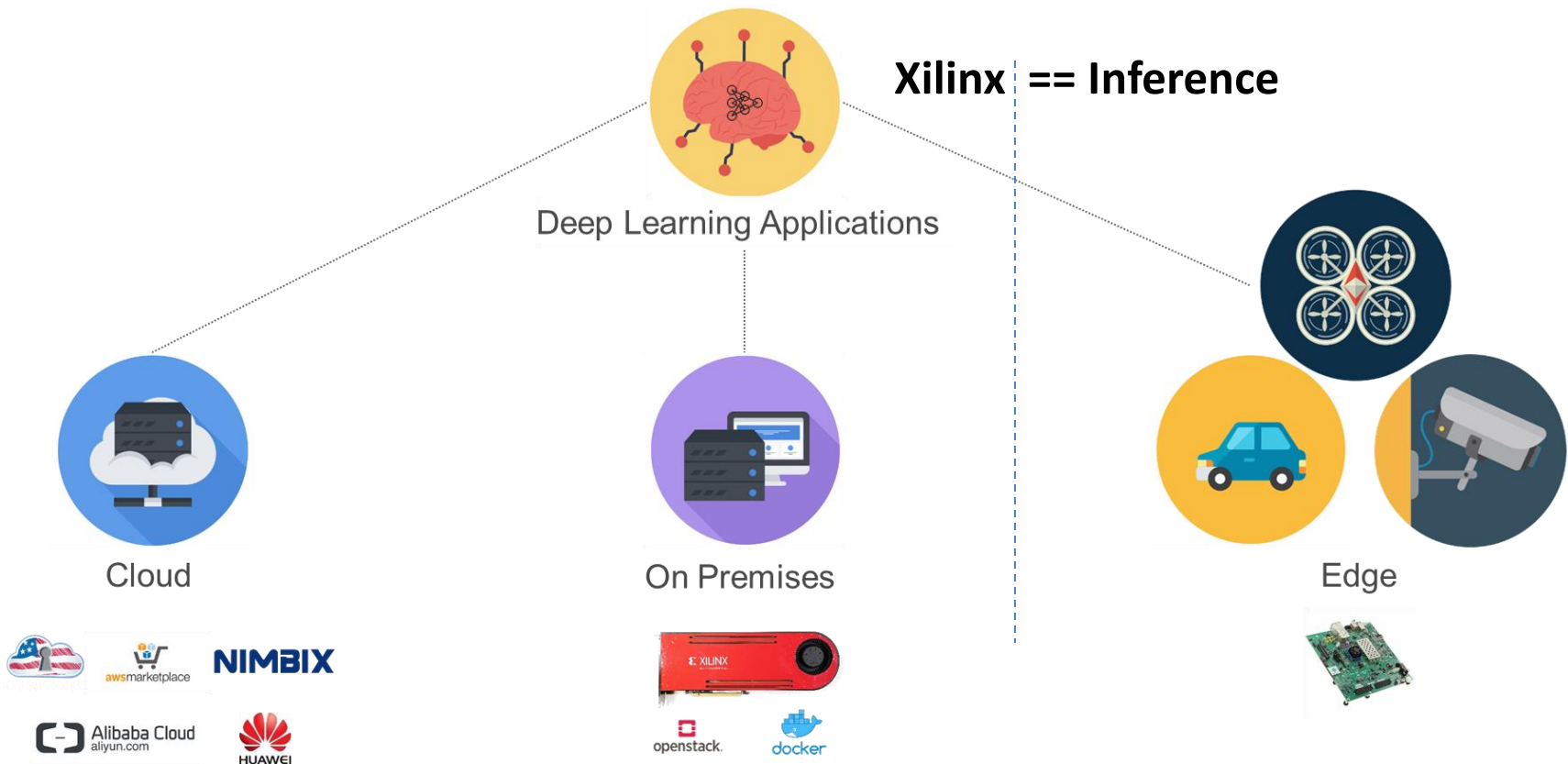
2 шага последовательных шага оптимизации – прореживание (Pruning) и квантизация (Quantization)



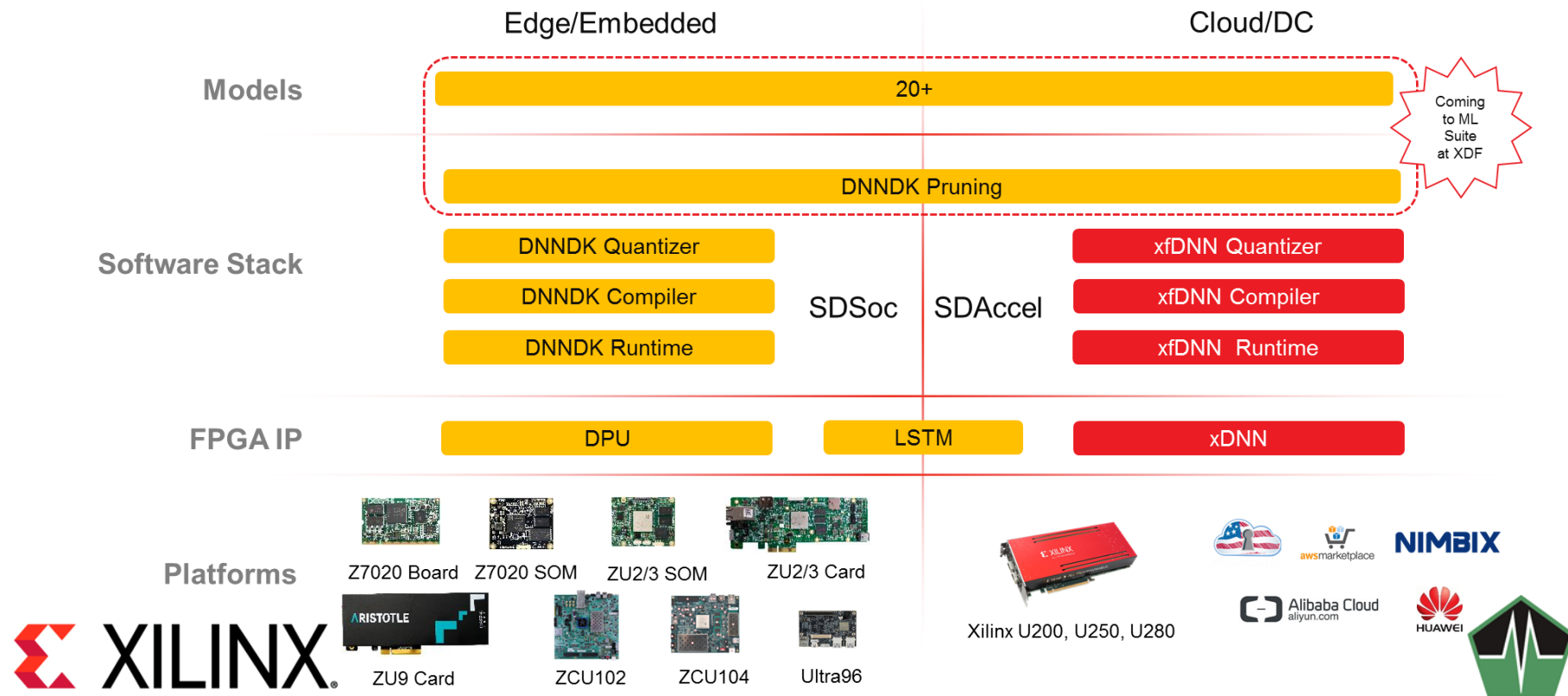
- Consists of two separate tools
 - Pruning Tool - decent_p
 - Quantization Tool – decent_q

- Effects
 - Compress model size 5x ~ 100x
 - Compress running time 1.5x – 10x

5. Решения Xilinx для ИС (1)



5. Решения Xilinx для HC (2)



6. Серверные/Облачные решения Xilinx (1)

Облака для работы

[AWS](#)

[Nimbix](#)



6. Серверные/Облачные решения Xilinx(2)

Xilinx ML-SUITE

Open Source Software / Apps
And Models

xfDNN Middleware

xfDNN Middleware, Tools and
Runtime

xfDNN Compiler and Optimizer

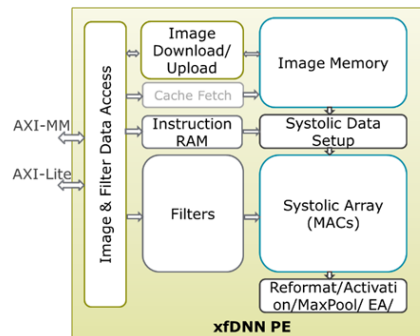
Xilinx xfDNN Runtime

Xilinx Platforms

Xilinx Reconfigurable
Acceleration Platform

Boards / Platforms

xDNN PE on Xilinx FPGAs



- Xilinx xDNNv2 IP
- 500Mhz
- Support for 16 and 8 bit

Features		Description	
Supported Operations	Convolution	Kernel Sizes	1x1, 3x3, 5x5, 7x7
		Strides	1-8
		Padding	<= Kernel Size
		Dilation	Factor: 1,2,4
		Activation	ReLU
		Bias	Per Channel
		Scaling	Scale & Shift Value Per Channel
		Kernel Sizes	2x2, 3x3, 4x4, 5x5, 6x6, 7x7
	Max Pooling	Strides	1-7
		Padding	<= Kernel Size
	Avg Pooling	Kernel Sizes	2x2, 3x3, 4x4, 5x5, 6x6, 7x7
		Strides	1-7
	Element-wise Add	Padding	<= Kernel Size
		Width & Height must match; Depth can mismatch.	
Miscellaneous	DDR caching	Supported; See Documentation	
	Expanded set of image sizes	Square, Rectangular	
	Upsampling	Strides	Factor: 2,4,8,16
	Convolution Transpose	see Convolution	
	Deconvolution	see Convolution	
	Data width	16-bit or 8-bit	
	Target Clock Speed	XDNN core: <= 300MHz; DSP Systolic Array: <= 500MHz;	
	Networks Supported	Classification, Detection, Segmentation	



6. Серверные/Облачные решения Xilinx (3)

- ML-SUITE – серверный/облачный продукт, 4 варианта: U200, U250, VCU1525, Cloud
- Ссылки: <https://www.xilinx.com/products/design-tools/ai-inference/ai-developer-hub.html>
- Скачивать отсюда: <https://www.xilinx.com/products/acceleration-solutions/xilinx-machine-learning-suite.html#overview>
- Ресурсы на github: <https://github.com/Xilinx/ml-suite>



7. Решения Xilinx для автономных систем (1)

Модели



Face detection

Pose estimation

Video analytics

Lane detection

Object detection

Segmentation

Фреймворки

Caffe

Darknet



Средства разработки



Compression

Pruning

Quantization

Compilation

Compiler

Assembler

Runtime

Core API
Driver

Loader
Profiler

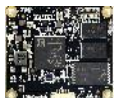
IP ядра

DPU Core

Платформы



Z7020 Board



Z7020 SOM



ZU2/3 SOM



ZU2/3 Card



ZU9 Card



ZCU102



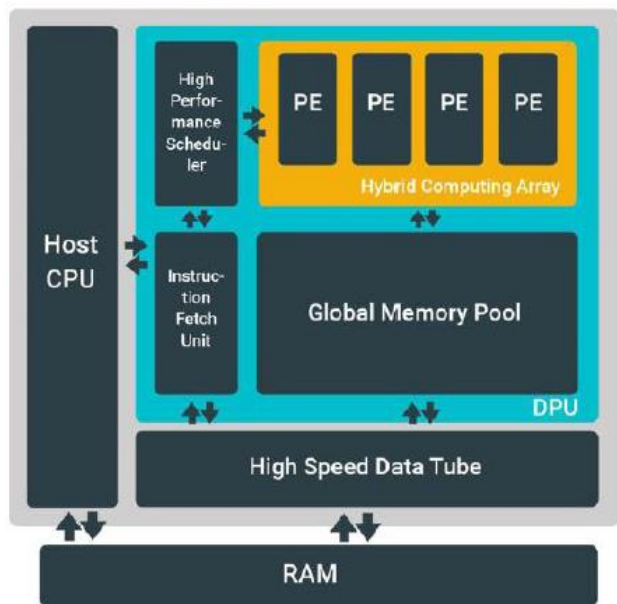
ZCU104



Ultra96

7. Решения Xilinx для автономных систем (2)

DPU

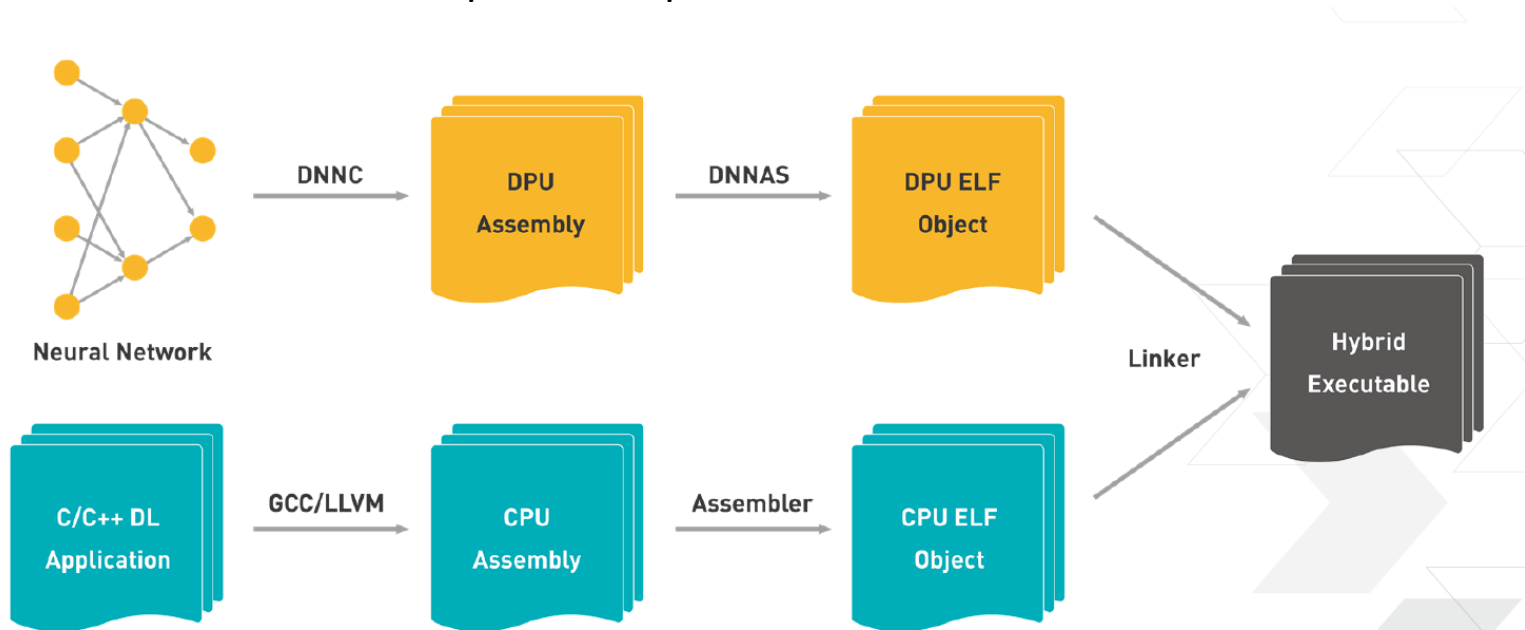


- Сопроцессор для CPU
- Soft IP-ядро, оптимизированное для операций с HC
- Собственная система команд
- Несколько разновидностей, различающихся по производительности: B1152 ...B4096 ...
- Устанавливается в платформу и далее не меняется
- Аппаратный минимум – Zynq7020



7. Решения Xilinx для автономных систем (3)

Разработка приложений



7. Решения Xilinx для автономных систем (4)

Структура DNNDK

```
Sdnndk_pkg
|-- common
|-- COPYRIGHT
|-- host_x86
|   |-- install.sh
|   |-- models
|   |-- pkgs
|   |-- ubuntu14.04
|   |-- ubuntu16.04
|-- board_name
|   |-- install.sh
|   |-- pkgs
|   |-- bin
|   |-- driver
|   |-- include
|   |-- lib
|   |-- samples
|   |-- common
|   |-- inception_v1
|   |-- inception_v1_mt
|   |-- resnet50
|   |-- resnet50_mt
|   |-- mobilenet
|   |-- mobilenet_mt
|   |-- face_detection
|   |-- pose_detection
|   |-- video_analysis
|   |-- adas_detection
|   |-- segmentation
```

Пример программы

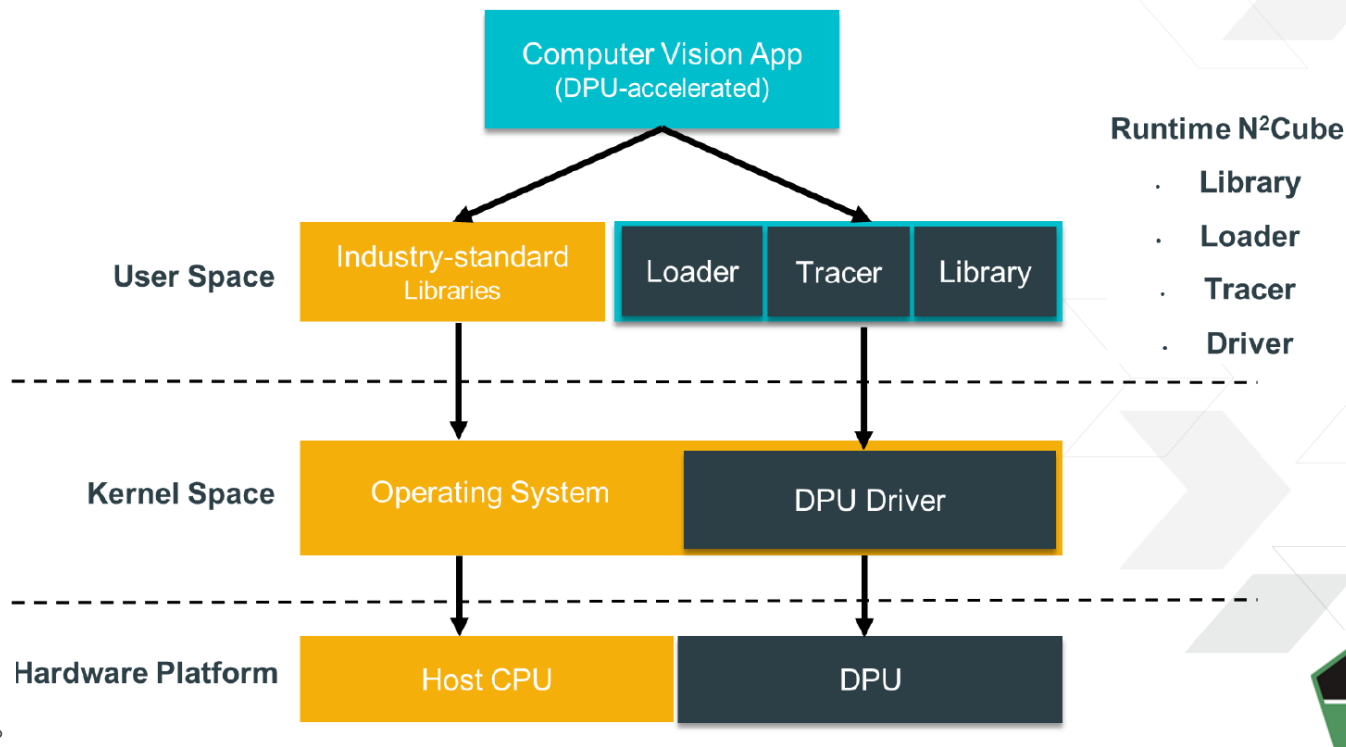
```
1 int main(int argc, char *argv[])
2 {
3     DPUKernel *kernel_conv;
4     DPUKernel *kernel_fc;
5     DPUTask *task_conv;
6     DPUTask *task_fc;
7     char *input_addr;
8     char *output_addr;
9
10    /* DNNDK API to attach to DPU driver */
11    dpuInit();
12
13    /* DNNDK API to create DPU kernels for CONV & FC networks */
14    kernel_conv = dpuLoadKernel("resnet50_conv", 224, 224);
15    kernel_fc = dpuLoadKernel("resnet50_fc", 1, 1);
16
17    /* Create tasks from CONV & FC kernels */
18    task_conv = dpuCreateTask(kernel_conv);
19    task_fc = dpuCreateTask(kernel_fc);
20
21    /* Set input tensor for CONV task and run */
22    input_addr = dpuGetTensorAddress(dpuGetTaskInputTensor(task_conv));
23    setInputImage(Mat &image, input_addr);
24    dpuRunTask(task_conv);
25    output_addr = dpuGetTensorAddress(dpuGetTaskOutputTensor(task_conv));
26
27    /* Run average pooling layer on CPU */
28    run_average_pooling(output_addr);
29
30    /* Set input tensor for FC task and run */
31    input_addr = dpuGetTensorAddress(dpuGetTaskInputTensor(task_fc));
32    setFCInputData(task_fc, input_addr);
33    dpuRunTask(task_fc);
34    output_addr = dpuGetTensorAddress(dpuGetTaskOutputTensor(task_fc));
35
36    /* Display the Classification result from FC task */
37    displayClassificationResult(output_addr);
38
39    /* DNNDK API to destroy DPU tasks/kernels */
40    dpuDestroyTask(task_conv);
41    dpuDestroyTask(task_fc);
42
43    dpuDestroyKernel(kernel_conv);
44    dpuDestroyKernel(kernel_fc);
45
46    /* DNNDK API to detach from DPU driver and free DPU resources */
47    dpuFini();
48
49    return 0;
50 }
```

Просто вызываем ф-ции API



7. Решения Xilinx для автономных систем (5)

Конфигурация target-системы



7. Решения Xilinx для автономных систем (6)

DeePhi [DNNDK](#) – быстрый старт

1. Загрузить соответствующий вашему оборудованию пакет
2. Установить на HOST-систему под linux (либо сразу взять сконфигурированную VM)
3. Сконфигурировать средства разработки для хоста
4. Сконфигурировать целевую платформу
5. Взять описание готовой обученной сети и провести оптимизацию
6. Скомпилировать оптимизированную сеть
7. Интегрировать сеть в собственное приложение (программу)
8. Выполнить программу на целевой платформе.



7. Решения Xilinx для автономных систем (7)

Преимущества технологии аппаратного ускорения Xilinx в системах реального времени

Driving Event: Car Stopped Ahead

Response:
Nvidia Engages Brakes
(Batch-size = 8 to 64)

Nvidia



49-320ms



Xilinx



2.7ms



Notes:

- Xilinx: ZU9 running GoogLeNet @ batch = 1
- Nvidia TX1: 256 Cores; running GoogLeNet @ batch = 8
- Assuming the car was driving at 65 mph



7. Решения Xilinx для автономных систем (8)

Ссылки

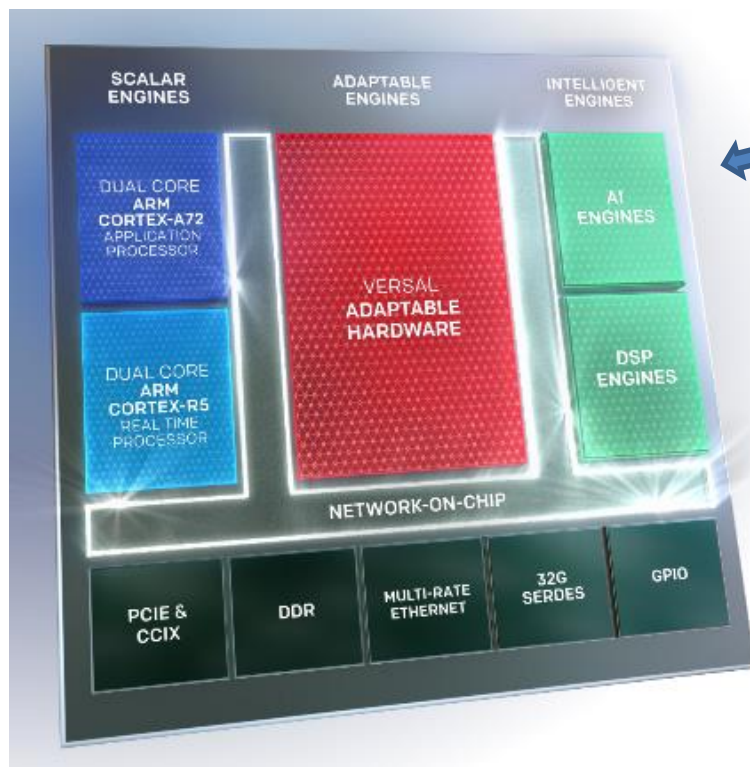
- Стартовая страница: <https://www.xilinx.com/products/design-tools/ai-inference/ai-developer-hub.html#edge>
- DNNDK User Guide
https://www.xilinx.com/support/documentation/user_guides/ug1327-dnndk-user-guide.pdf
- Wiki <https://xilinx-wiki.atlassian.net/wiki/spaces/A/overview>
- Forum <https://forums.xilinx.com/t5/DeepPhi-DNNDK/bd-p/DeepPhi>
- Github <https://github.com/Xilinx/Edge-AI-Platform-Tutorials>



7. Решения Xilinx для автономных систем (8)

Перспективы развития

ACAP [Versal](#) 7nm
Серия AI



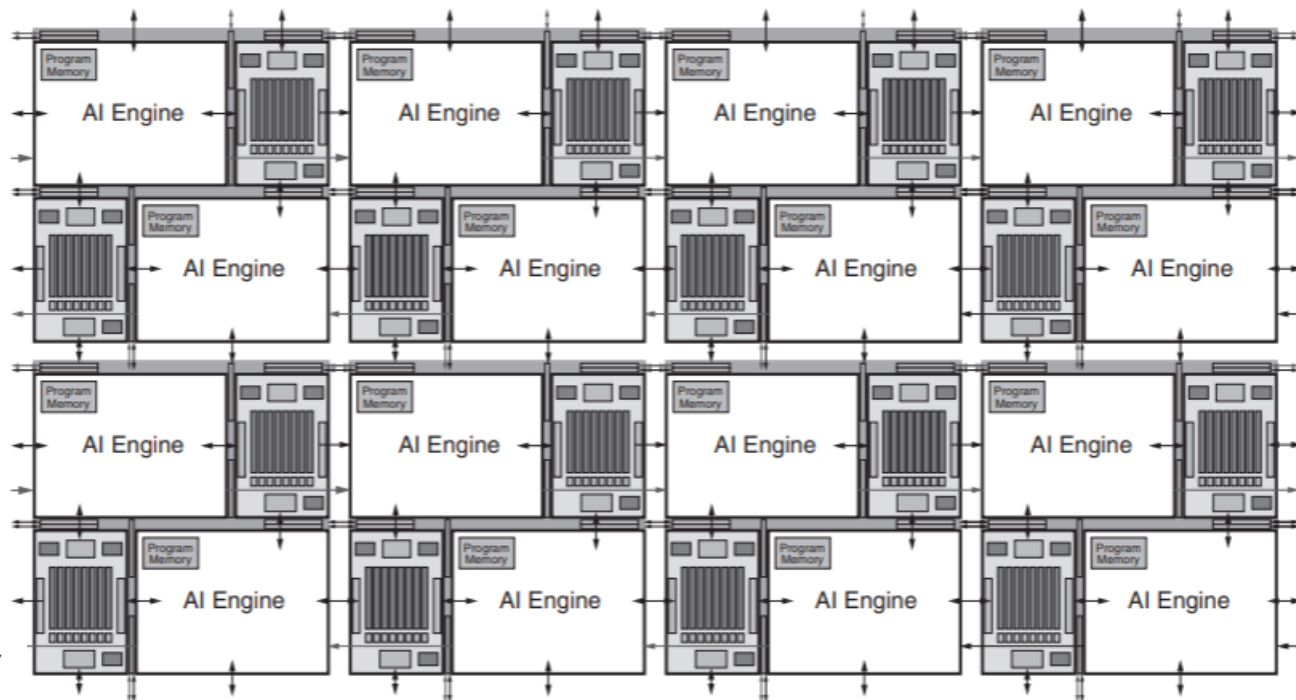
Аппаратные
Процессоры AI



7. Решения Xilinx для автономных систем (9)

Перспективы развития

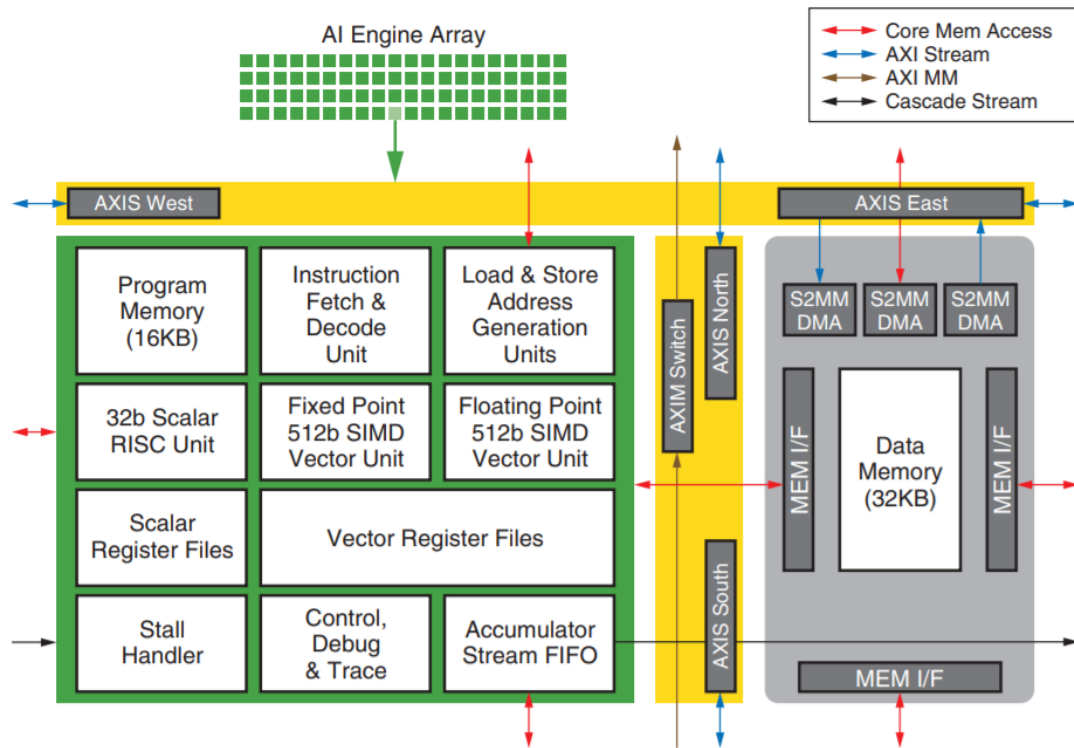
Процесоры AI
Составляют
массив



7. Решения Xilinx для автономных систем (10)

Перспективы развития

Структура ячейки AI



8. Практическая демонстрация



Platform	Device	Revision
ZCU102	ZU9EG	Rev D2, Rev 1.x
Ultra96	ZU3EG	Rev D and newer

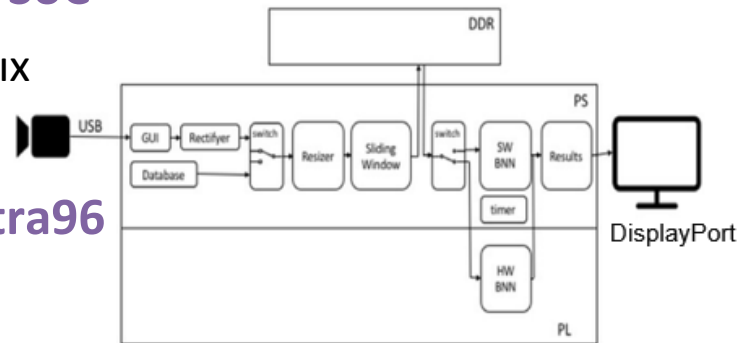
➤ Демонстрация на ZU+ MPSoC

1. Распознавание дорожных знаков с веб-камеры

➤ Платформы: ZCU102 и Ultra96

➤ Название:

AIC Design Example,
[TechTip](#) on Xilinx Wiki



ZCU102:
Tiles per second: ~19500
Images per second: ~90
Acceleration: ~8500

Ultra96:
Tiles per second: ~14500
Images per second: ~66
Acceleration: ~6200

Спасибо за внимание!

Компания Макро Групп:

- ✓ **Официальный партнер Xilinx**
- ✓ **Комплексная поставка электронных компонентов**
- ✓ **Техническая поддержка по всем вопросам применения продукции и ПО Xilinx**
- ✓ **Контрактное производство**

Обращайтесь:

- ✓ Vladimir.Vikulin@macrogroup.ru
- ✓ Dmitry.Khorkov@macrogroup.ru
- ✓ fpga@macrogroup.ru

