Embedded AI for Vehicle Motion Control

Qian Weizhe, Infineon Tech.

2024 MathWorks
中国汽车年会

Application Benefits of Embedded AI

Embedded AI

– Enables more **accurate** control approaches which leads to **higher energy efficiency** especially in the Electric Drive Train

– Enables **higher dependability** by the use of virtual sensors

– Reduces the **bill of material** for the entire system

– Reduces the system and software **development effort**

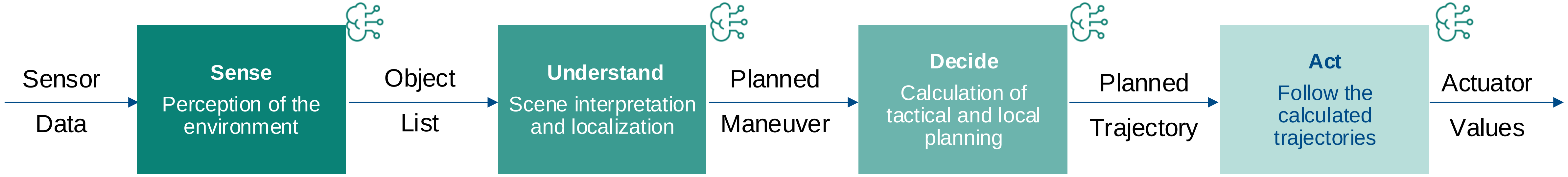
AI Enhanced vehicle motion with AURIX^{TM} TC4x is driving a test vehicle

Sensor Data → **Sense** Perception of the environment → Object List → **Understand** Scene interpretation and localization → Planned Maneuver → **Decide** Calculation of tactical and local planning → Planned Trajectory → **Act** Follow the calculated trajectories → Actuator Values

	Sense	Understand	Decide	Act
Data Volume	› Very high	› High	› Medium	› Low
Compute Effort	› Very high	› High	› Medium to l	
Realtime Criticality	› Medium	› Medium		› High
Compute Unit	› HPC or specialized HW	› HPC	› HPC or MCU	› MCU

- IFX has the focus to support real-time processing for **safety critical** and **dependable applications**
- IFX want to explore **resource aware algorithms** based on AI approaches

AI Enhanced vehicle motion with AURIX^{TM} TC4x is driving a test vehicle

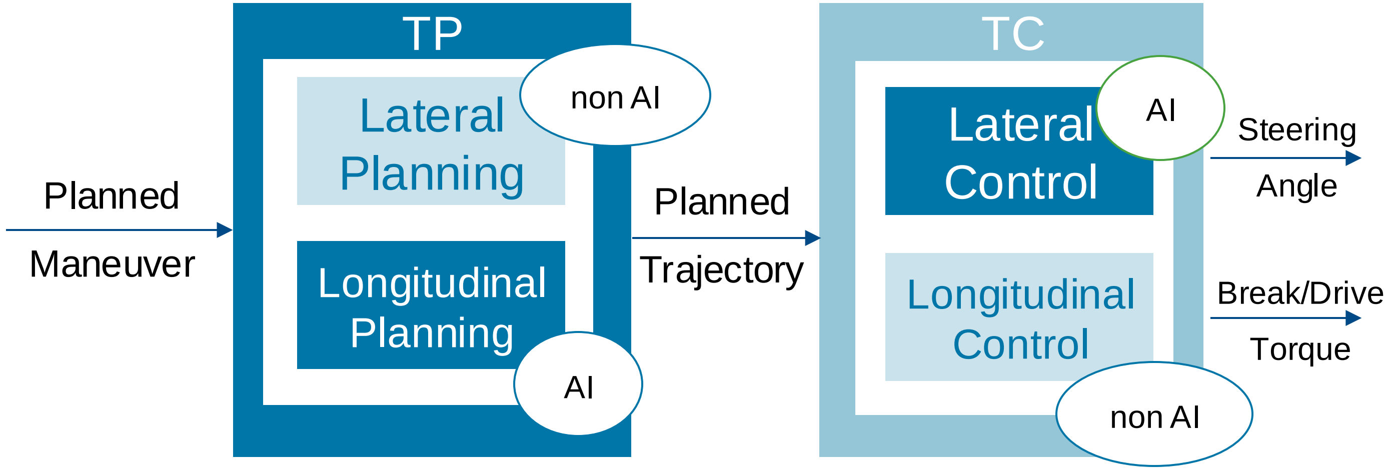


AI in perception and scene understanding already common!

AI to complement and enhance Trajectory Planning and Control is the step to improve driving comfort and energy efficiency

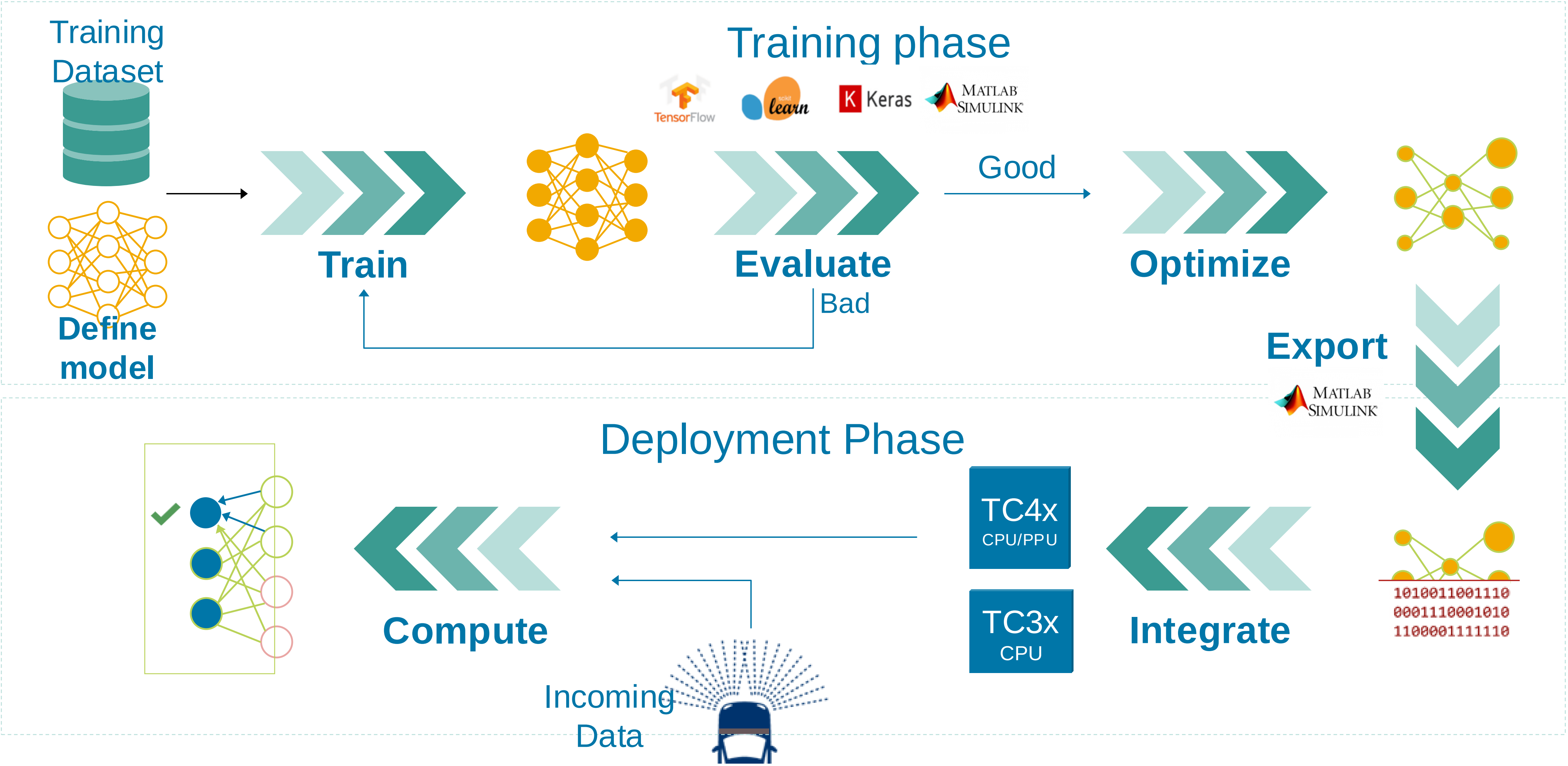
AI enhanced Trajectory Control (TC) increased the tracking accuracy by 50%

Trajectory Planning (TP) with AI enhanced Model Predictive Control (MPC) increases the energy efficiency for an ACC by up to 10%



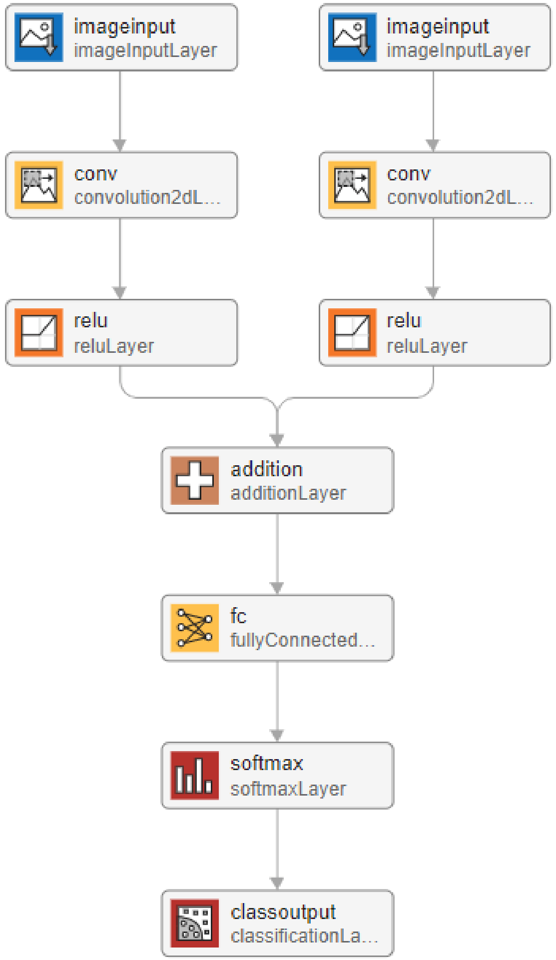
How to integrate AI workflow for AURIX^{TM} TC4x software development?

AI Development flow using different frameworks



Deep Learning Toolbox – Model design and training

Build-in Layers & Custom Layers




```
classdef preluLayer < nnet.layer.Layer ...
        & nnet.layer.Acceleratable
    % Example custom PReLU layer.

    properties (Learnable)
        % Layer learnable parameters

        % Scaling coefficient
        Alpha
    end

    methods
        function layer = preluLayer(args)
            % layer = preluLayer creates a PReLU layer.
            %
            % layer = preluLayer(Name=name) also specifies the
            % layer name.

            arguments
                args.Name = "";
            end

            % Set layer name.
            layer.Name = args.Name;

            % Set layer description.
            layer.Description = "PReLU";
        end

        function layer = initialize(layer,layout)
            % layer = initialize(layer,layout) initializes the layer
            % learnable parameters using the specified input layout.

            % Skip initialization of nonempty parameters.
            if ~isempty(layer.Alpha)
                return
            end

            % Input data size.
            sz = layout.Size;
            ndims = numel(sz);

            % Find number of channels.
            idx = finddim(layout,"C");
            numChannels = sz(idx);

            % Initialize Alpha.
            szAlpha = ones(1,ndims);
            szAlpha(idx) = numChannels;
            layer.Alpha = rand(szAlpha);
        end

        function Z = predict(layer, X)
            % Z = predict(layer, X) forwards the input data X through the
            % layer and outputs the result Z.

            Z = max(X,0) + layer.Alpha .* min(0,X);
        end
    end
end
```

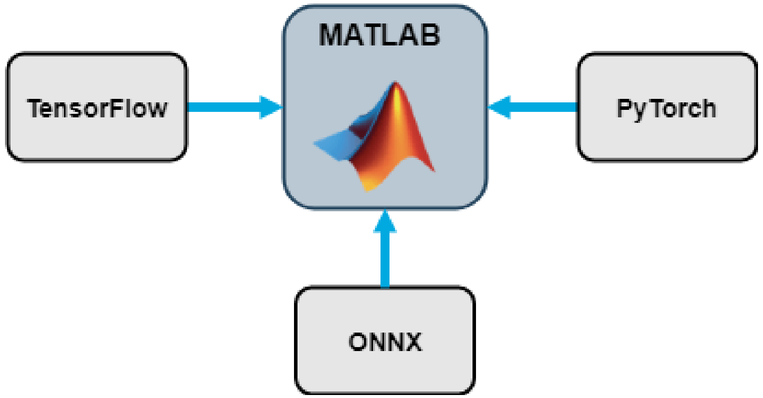
Custom Function

```
function [Y1,Y2,state] = model(parameters,X,doTraining,state)

% Initial operations
% Convolution - conv1
weights = parameters.conv1.Weights;
bias = parameters.conv1.Bias;
Y = dlconv(X,weights,bias,Padding="same");

% Batch normalization, ReLU - batchnorm1, relu1
offset = parameters.batchnorm1.Offset;
scale = parameters.batchnorm1.Scale;
trainedMean = state.batchnorm1.TrainedMean;
trainedVariance = state.batchnorm1.TrainedVariance;

if doTraining
    [Y,trainedMean,trainedVariance] = batchnorm(Y,offset,scale,trainedMean,trainedVariance);

    % Update state
    state.batchnorm1.TrainedMean = trainedMean;
    state.batchnorm1.TrainedVariance = trainedVariance;
else
    Y = batchnorm(Y,offset,scale,trainedMean,trainedVariance);
end

Y = relu(Y);


% Main branch operations
% Convolution - conv2
weights = parameters.conv2.Weights;
bias = parameters.conv2.Bias;
YnoSkip = dlconv(Y,weights,bias,Padding="same",Stride=2);

% Batch normalization, ReLU - batchnorm2, relu2
offset = parameters.batchnorm2.Offset;
scale = parameters.batchnorm2.Scale;
trainedMean = state.batchnorm2.TrainedMean;
trainedVariance = state.batchnorm2.TrainedVariance;
```

Ref: https://ww2.mathworks.cn/help/deeplearning/build-deep-neural-networks.html

Import external models into MATLAB workspace

Functions That Import Deep Learning Networks



External Deep Learning Platforms and Import Functions

This table describes the external deep learning platforms and model formats that the Deep Learning Toolbox functions can import.

External Deep Learning Platform	Model Format	Import Model as Network
TensorFlow 2 or TensorFlow-Keras	`SavedModel` format	`importNetworkFromTensorFlow`
PyTorch	Traced model file with the `.pt` extension	`importNetworkFromPyTorch`
ONNX	ONNX model format	`importNetworkFromONNX`

Reference: `Interoperability Between Deep Learning Toolbox, TensorFlow, PyTorch, and ONNX', https://ww2.mathworks.cn/help/deeplearning/ug/interoperability-between-deep-learning-toolbox-tensorflow-pytorch-and-onnx.html

Using Model Based Development to develop and build whole application for AURIX^{TM} TC4x

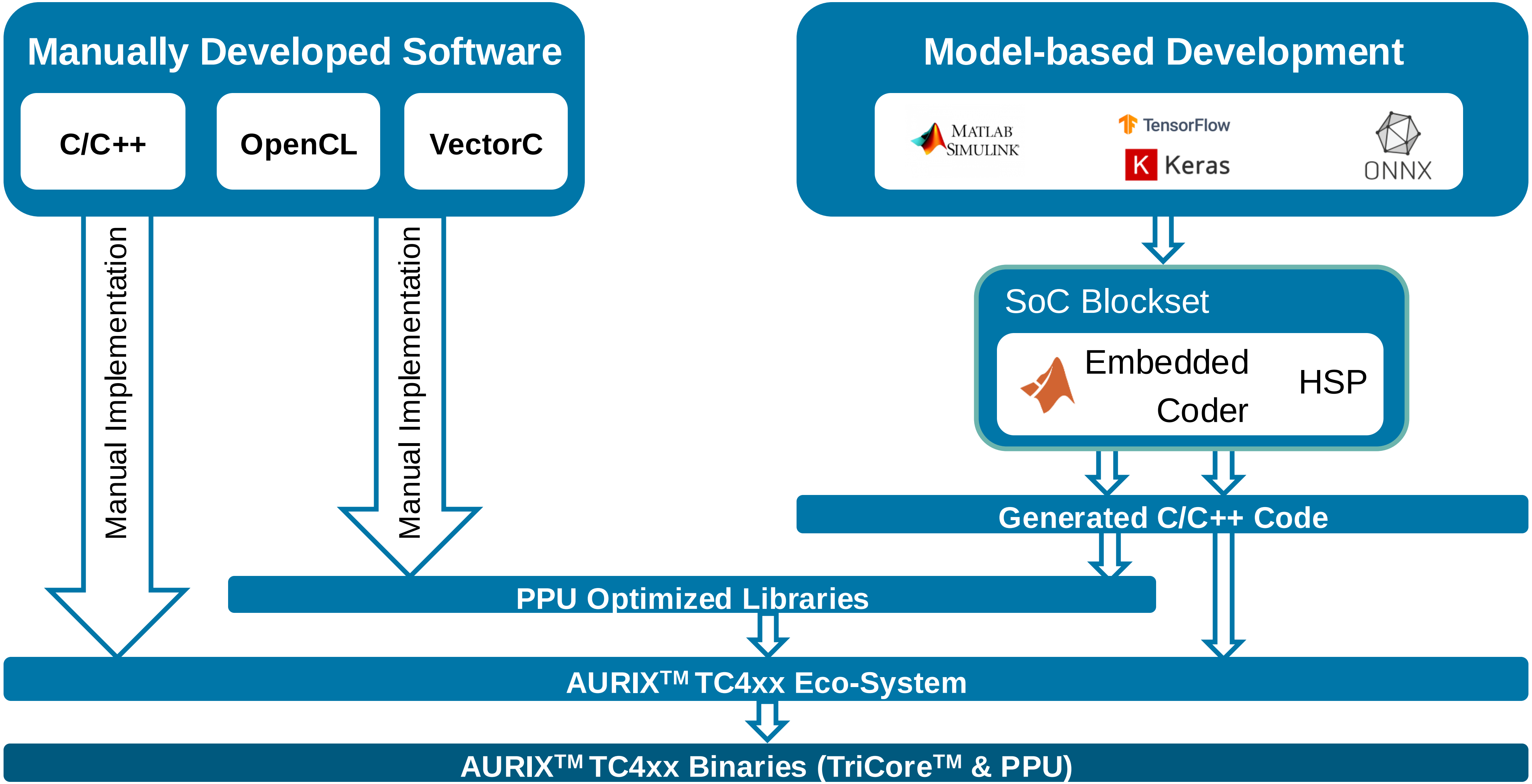
Why should we provide an ecosystem for model driven development?

Embedded AI

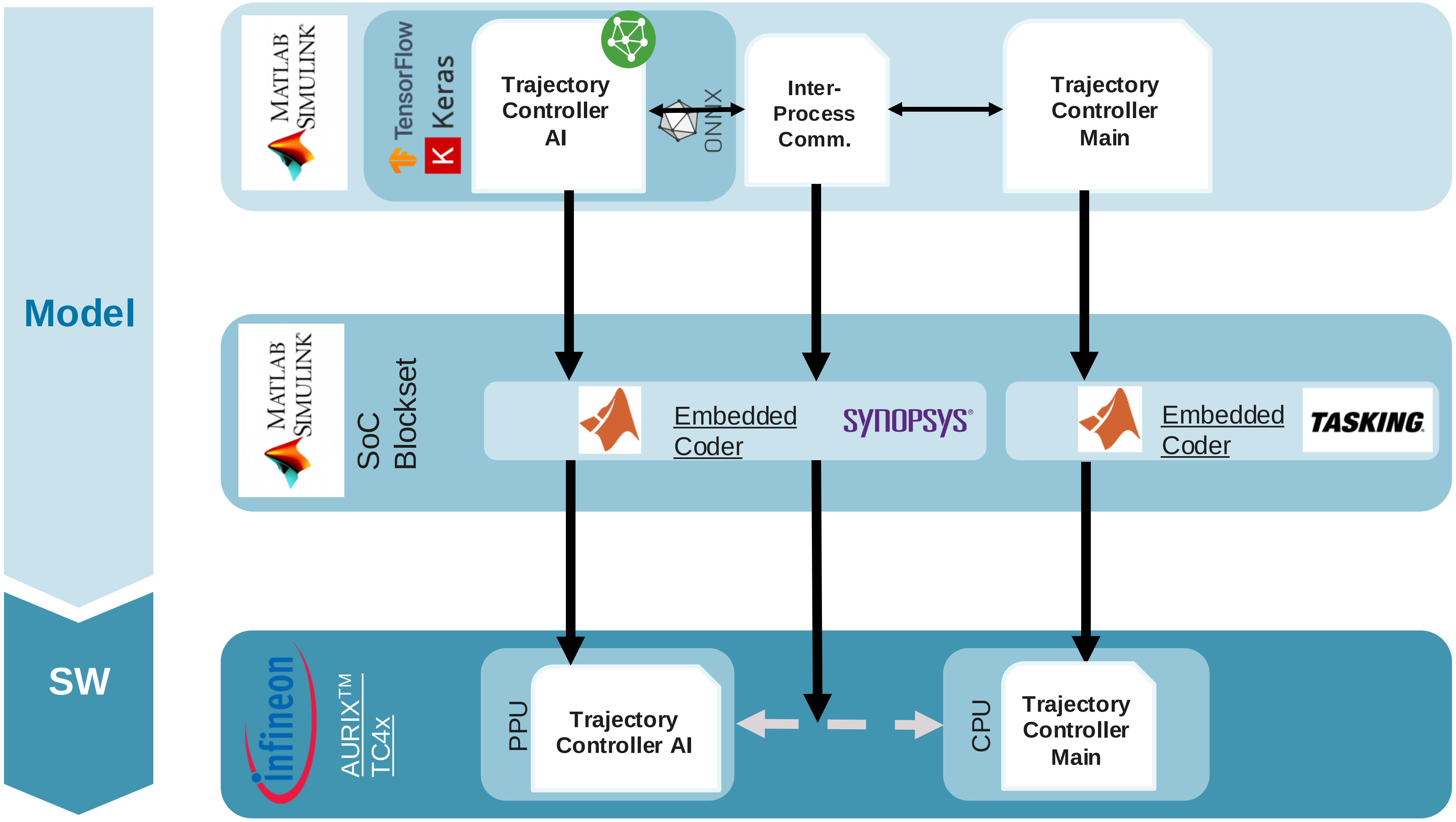
- Customers can use MATLAB/Simulink and include **IFX models for simulation and Autocode generation**
- **Reduction of engineering effort** and time to market advantage and provides further **abstraction** of HW
- **Reduction of bug rate** coming from manual coding
- **Maximization of re-use** from previous SW development projects

Mathworks provides Hardware Support Package for AURIX^{TM} TC4X since MATLAB 2022b

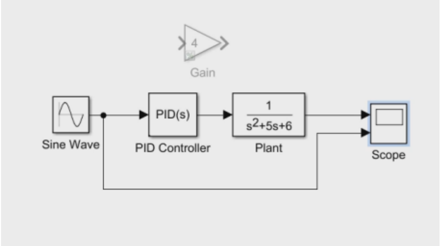
Embedded Software Development Landscape for AURIX^{TM} TC4x



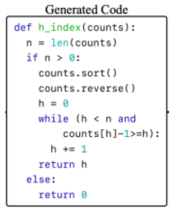
Partitioning of the Application using Mathworks Embedded Coder and SoC Blockset for AURIX^{TM} TC4x



How to use MATLAB Extensions to develop Software for AURIX™ TC4x ?

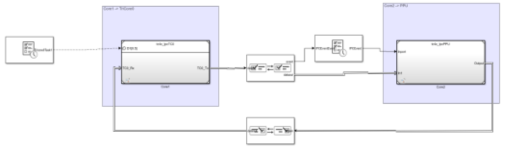


Embedded Coder ®



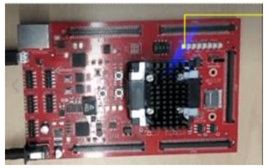
What is Embedded Coder ®?

- efficient C/C++ code
- AUTOSAR, MISRA C ™
- **code is portable** and can be compiled and executed **on any processor**



SoC Blockset™

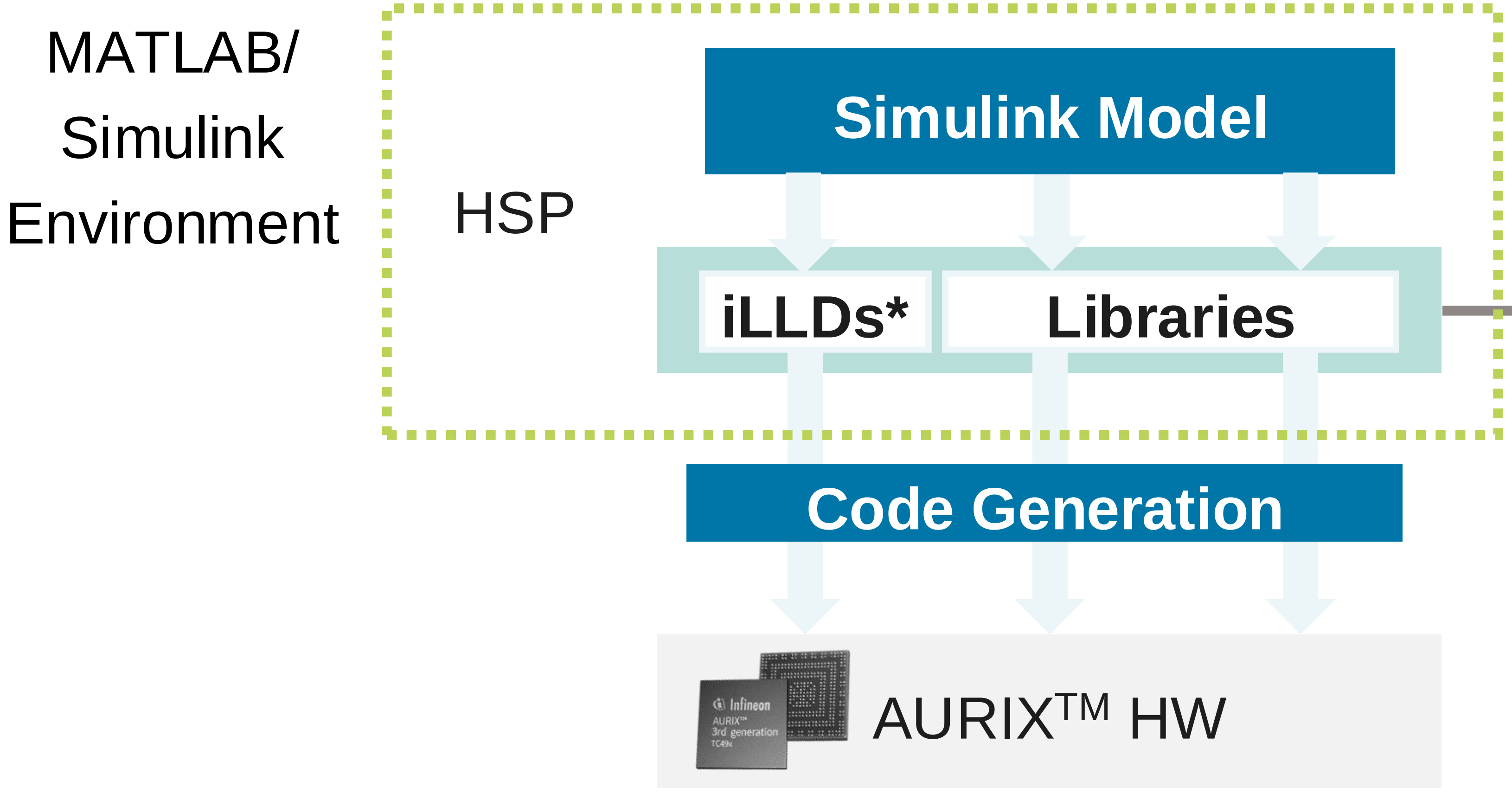
infineon AURIX™ HSP



What is SoC Blockset ?

- enables simulation and analysis of the performance of **algorithms on multicore SoC**
- **assists** the **code generation** for the target SoC

What is TC4x Hardware Support Package (HSP)?



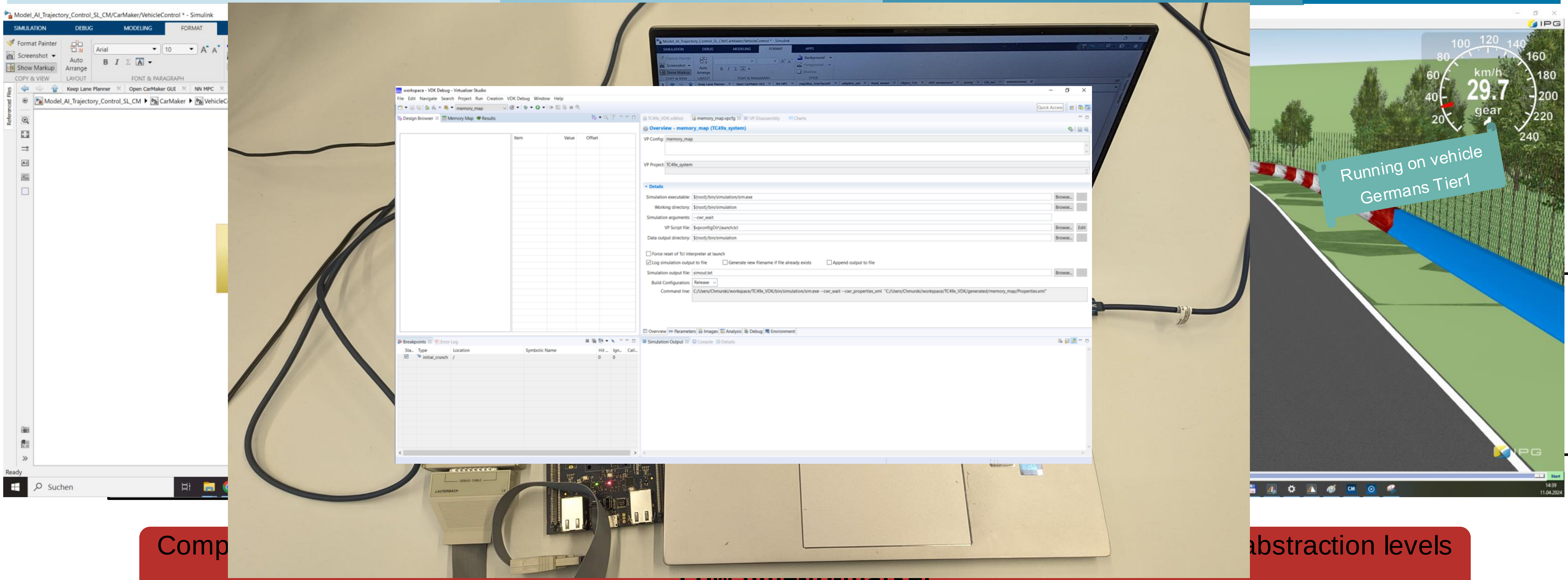
AURIX™ HSP

- Plugin for Embedded Coder and SoC Blockset
- Translates Simulink models into executable code
- Generated Code Optimized for AURIX™ TC4x
- Support for new peripherals added successively in new releases

**iLLD – Infineon Low Level Drivers*

Model-driven development is key for customer enablement

Software-in-the-Loop	Virtual Hardware In-the-Loop	Hardware In-the-Loop AURIX™ TC4	Vehicle Integration



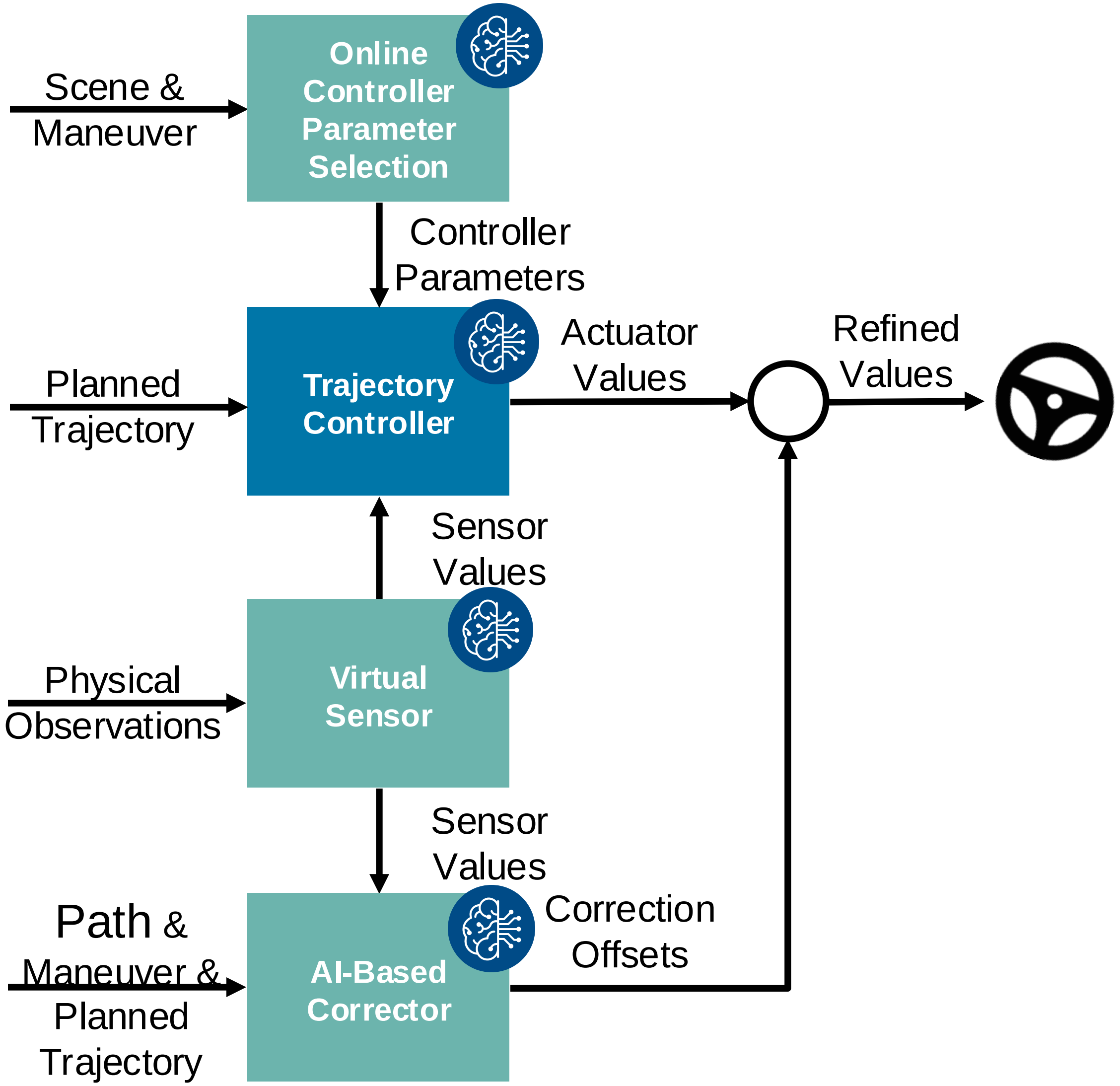
Comp… …abstraction levels

Low time-to-market

Example solutions for an AI enhanced Trajectory Controller

AI-Enhanced Trajectory Controller

For now, AI-Enhancements are considered for Lateral Control and not for Longitudinal Control



Online Controller Parameter Selection

- Use of classic controller
- Adapt controller parameterization during runtime
- Improved trajectory tracking

Correction of Actuator Values

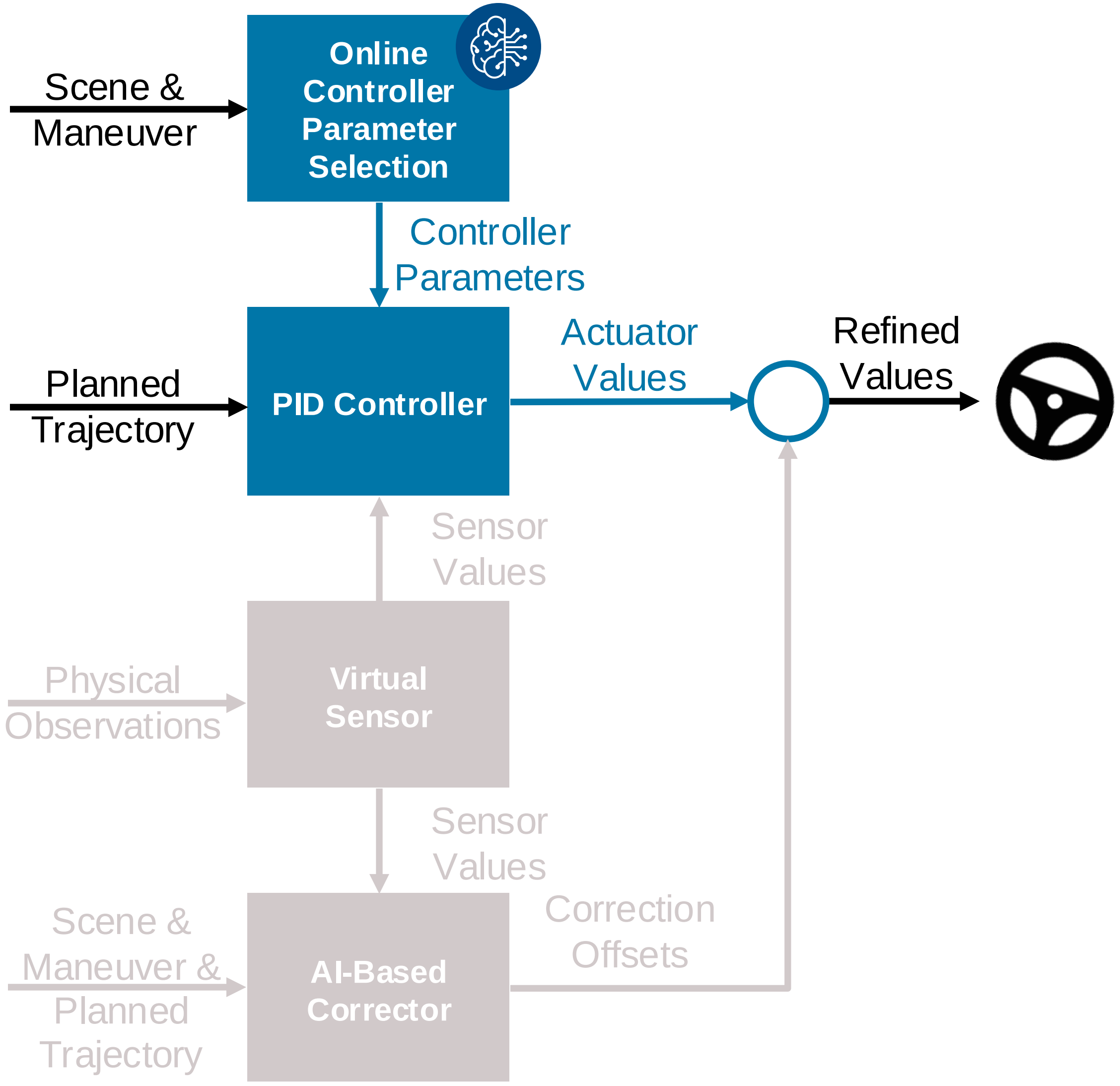
- Use of classic controller
- Let AI choose slight corrections
- Improved trajectory tracking

Virtual Sensor

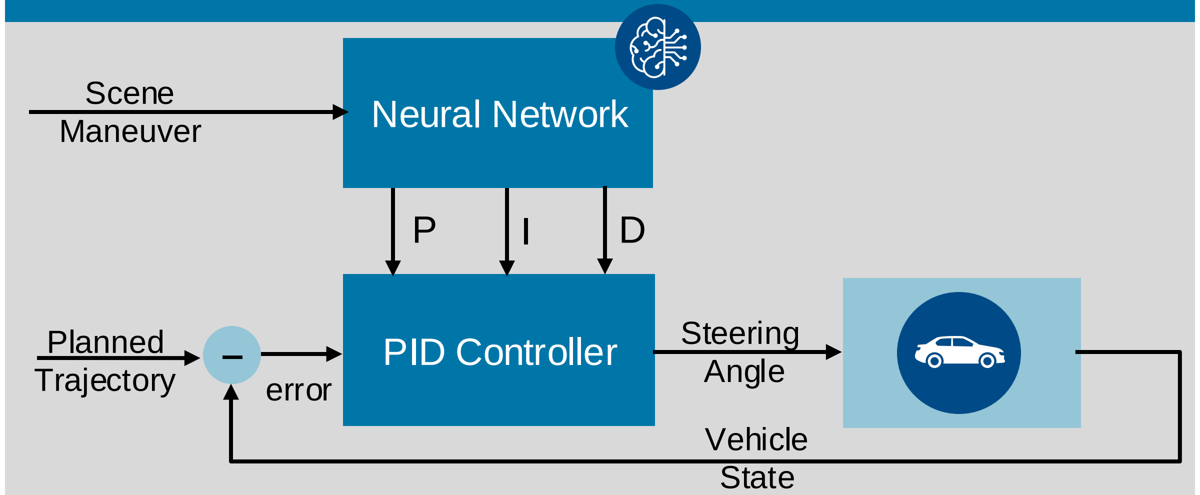
- Use of available sensor measurements
- Infer additional information from measurements
- Improved trajectory tracking

AI Purpose and Benefit

Classical Controller With AI-based Online Controller Parameter Selection



AI-Enhanced Trajectory Controller



<u>Learning</u>

- Using **MLP** to learn mapping between **scene maneuver** and **optimal controller parameters (PID)** for current driving scenario contained inside ODD

<u>Input</u>

- **Polynomial interpolation** of the **planned trajectory ahead, vehicle dynamics**

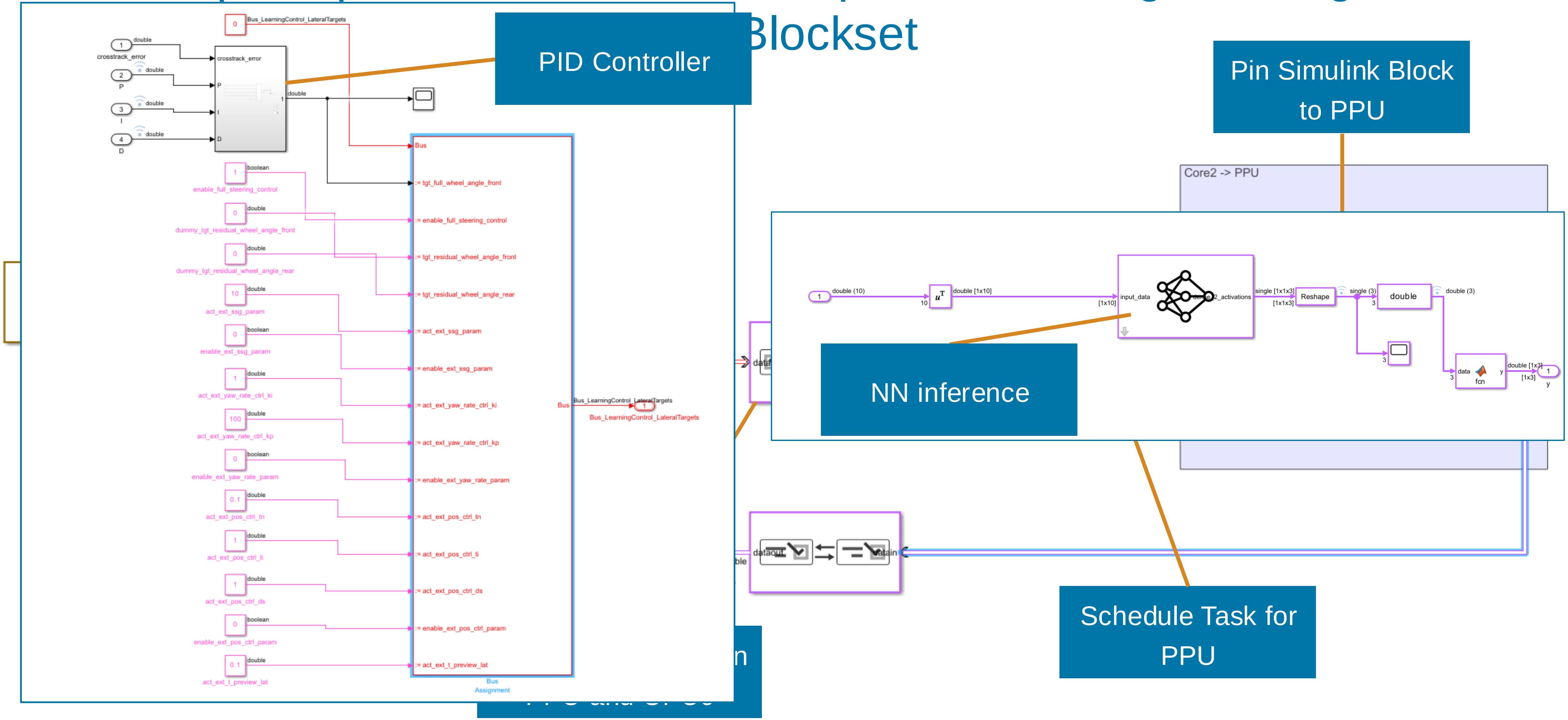
<u>Output</u>

- **PID gains** for P, I and D

<u>Structure</u>

- Actor NN consists of **339 parameters & 3 layers**

One-stop-shop solution for sw development, building and target ... Blockset



AI-Enhanced PID Controller outperforms Baseline Controller

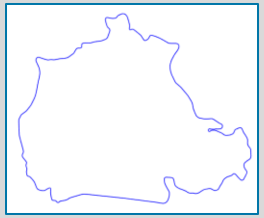
Results Tracking Accuracy

Training Setup

- Closed loop simulation CarMaker & MATLAB/Simulink
- Lane change scenario – right & left

Test Setup

- Nürburgring track – 70 km/h



Test Results

- AI-Enhanced Controller tracking acc higher by 47%

KPI	Conventional PID	AI-Enhanced PID
Accumulated lateral deviation (CTE)	68313.0 m	**35907.0 m**

Performance Measures

PPU running Neural Network

- memory footprint **~37 KB**
- Execution time **1403 cycles ~3.5 µs**

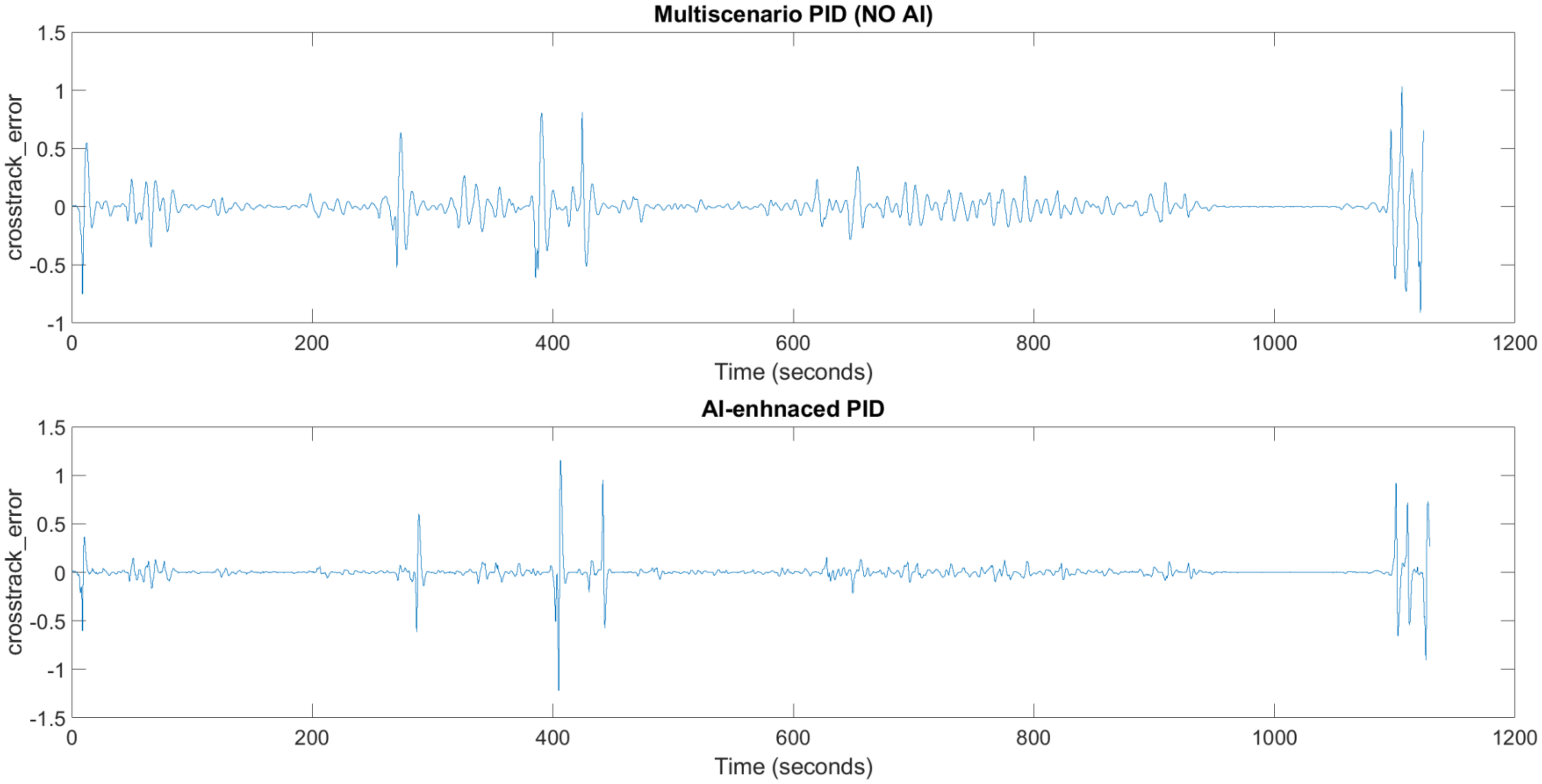
TriCore CPU running PID controller

- Execution time **250 cycles ~0.63 µs**

Summary

- Classical controller suffers in generalization
- Linearized controller can be improved introducing ML **covering non-linear behavior or enable adaptation**
- Implies **higher energy effiency**

AI-Enhanced PID Controller outperforms Baseline Controller



KPI	Conventional PID	AI-Enhanced PID
Accumulated lateral deviation (CTE)	68313.0 m	**35907.0 m**

Comparison of Classical Controller with AI-based Online Parameter Selection

Classical Trajectory Controller

AI-Enhanced Trajectory Controller

Summary

- Embedded AI **enables** the innovation for the **next generation** of Automated Driving and Electric Vehicle.
- **AURIX™ TC4x** with its ASIL-D related **AI accelerator (PPU)** provides the backbone to use Embedded AI in safety critical applications.

- MATHWORKS together with Infineon (AURIX™ HSP) offer complete ecosystem for **model driven development** and close-loop validation on different abstraction levels.
- User-friendly and flexible **AI model design and deployment** provided by MATHWORKS speeds up algorithm design and development.

- Embedded AI in **trajectory control and planning** can increase **energy efficiency** & **dependability.**
- Infineon has developed an AI enhanced **trajectory control** consisting out of **resource aware AI models** (Neural Network) and ODD definition for training and test dataset acquisition

2024 MathWorks
中国汽车年会

Thank you

MathWorks