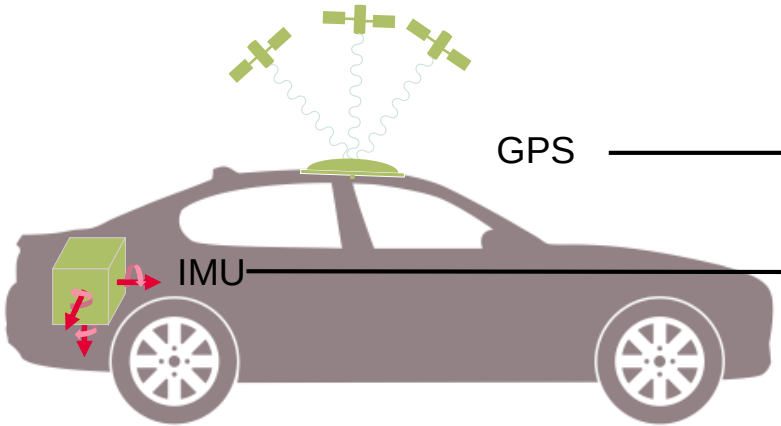
Model-based software design on next generation AURIX™

Embedded Solutions Conference 2021

Problem statement

Data Acquistion | **Strapdown Navigation** | **Sensor Fusion**



GPS → Position, Speed → $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$

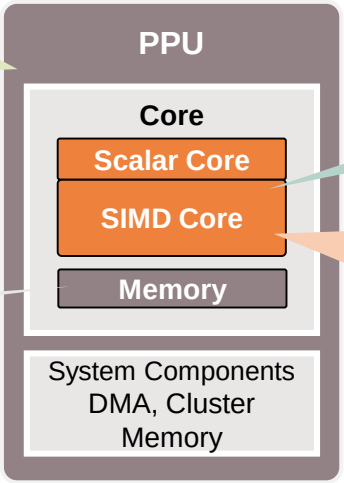
Applying Kalman Filter

IMU → Acceleration, Turn Rate → $\int$ → Position, Speed, Orientation → $P_{k|k} = (I - K_k H_k)\, P_{k|k-1}$

- Enhanced ADAS functions rely on sensor fusion
- Sensor fusion algorithms have a high demand of matrix and vector operations
- Keeping the time budget within a hard real-time system is challenging with big matrix operations
- Algorithm developer like to work HW independent

The next AURIX™ generation provides breakthrough performance with SIMD vector DSP for vector operations

PPU is a complete compute cluster flexible configurable for many use cases



Scalar core includes double precision floating point unit

VLIW architecture for parallel processing with a mixture of concurrent scalar and SIMD instruction issues

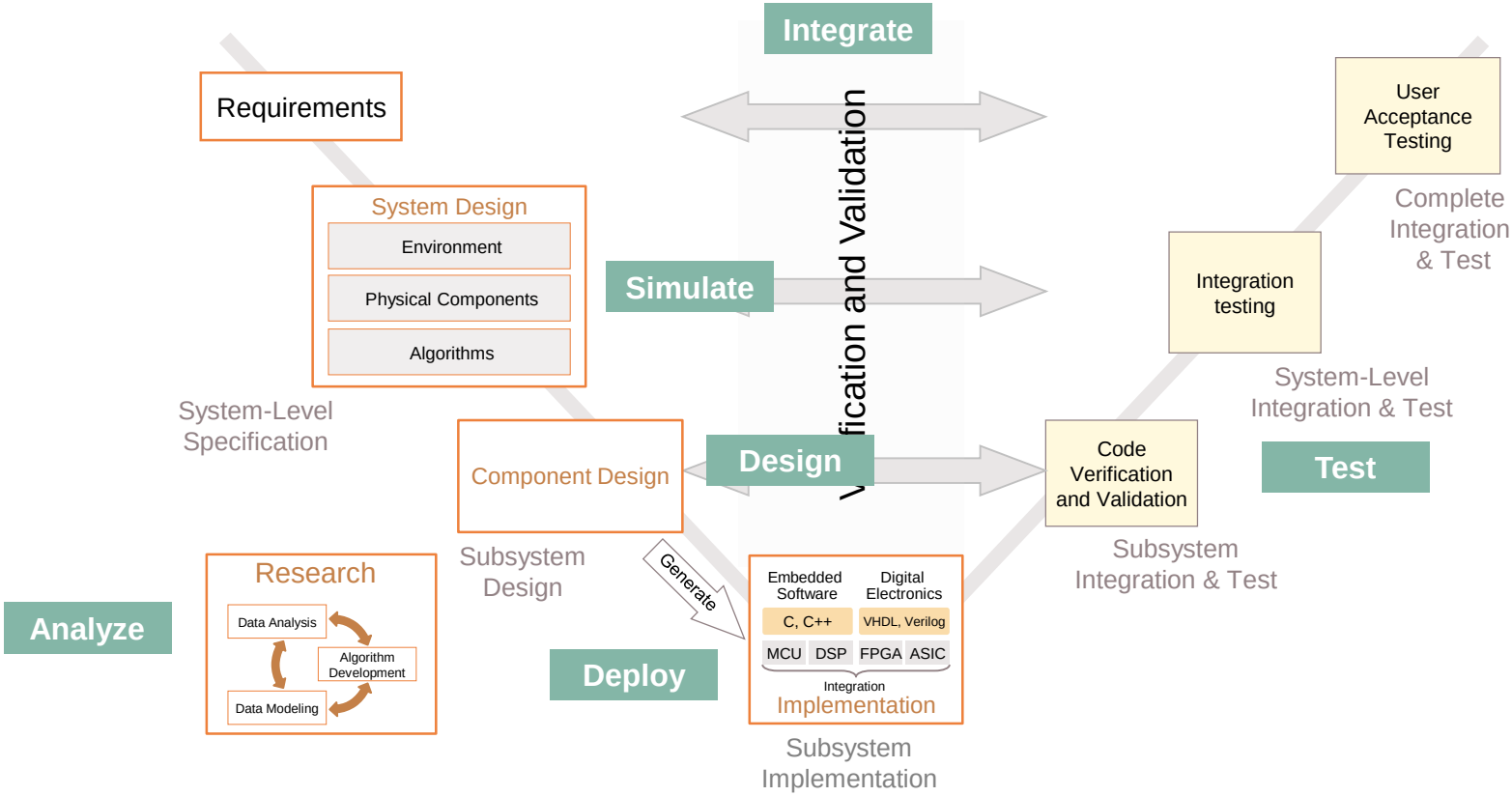
Can be programmed using model based design approaches and code generation

Core local memory enables fast vector data processing

The new SIMD vector DSP co-processor Parallel Processing Unit (PPU) in the new generation AURIX™ offers a breakthrough performance for

- Vector / Matrix operation acceleration & data processing
- Deep Learning Algorithm
- High speed control implementations

Model-based design



Develop Automated Driving Applications with MATLAB® and Simulink®

Automated Driving Systems

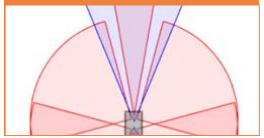
Environment

Scenes

Scenarios

Vehicles

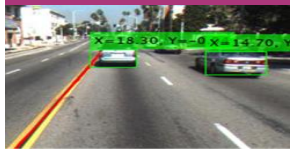
Sensors



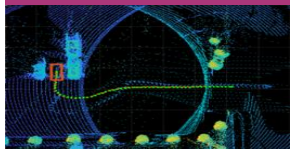
Dynamics

Algorithms

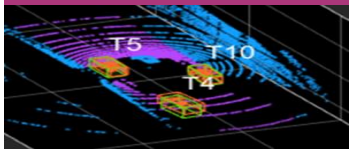
Detection



Localization

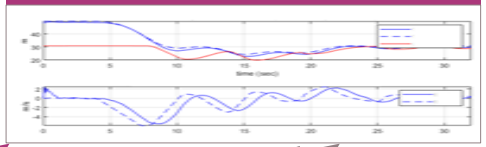


Tracking & Fusion



Planning

Decision & Controls



Development Platform

Analyze | Simulate | Design | Deploy | Integrate | Test

Develop Automated Driving Applications with MATLAB® and Simulink®

Automated Driving Systems

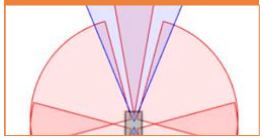
Environment

Scenes

Scenarios

Vehicles

Sensors



Dynamics

Algorithms

Detection

Localization

Tracking & Fusion

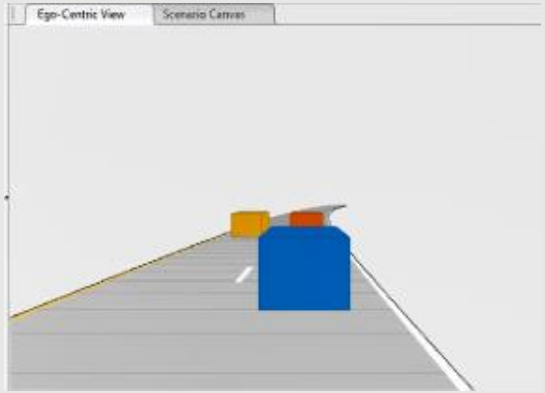
Planning

Decision & Controls

Development Platform

Analyze | **Simulate** | Design | Deploy | Integrate | Test

Synthesize scenarios to test algorithms and systems

Scenes	**Cuboid**	**Unreal Engine® from Epic Games®**
		
Testing	Controls, sensor fusion, planning	Controls, sensor fusion, planning, perception
Sensing	Probabilistic vision (detection list) Probabilistic lane (detection list) Probabilistic RADAR (detection list) LiDAR (point cloud)	Monocular camera (image, labels, depth) Fisheye camera (image) Probabilistic RADAR (detection list) LiDAR (point cloud)

Graphically author scenarios with Driving Scenario Designer

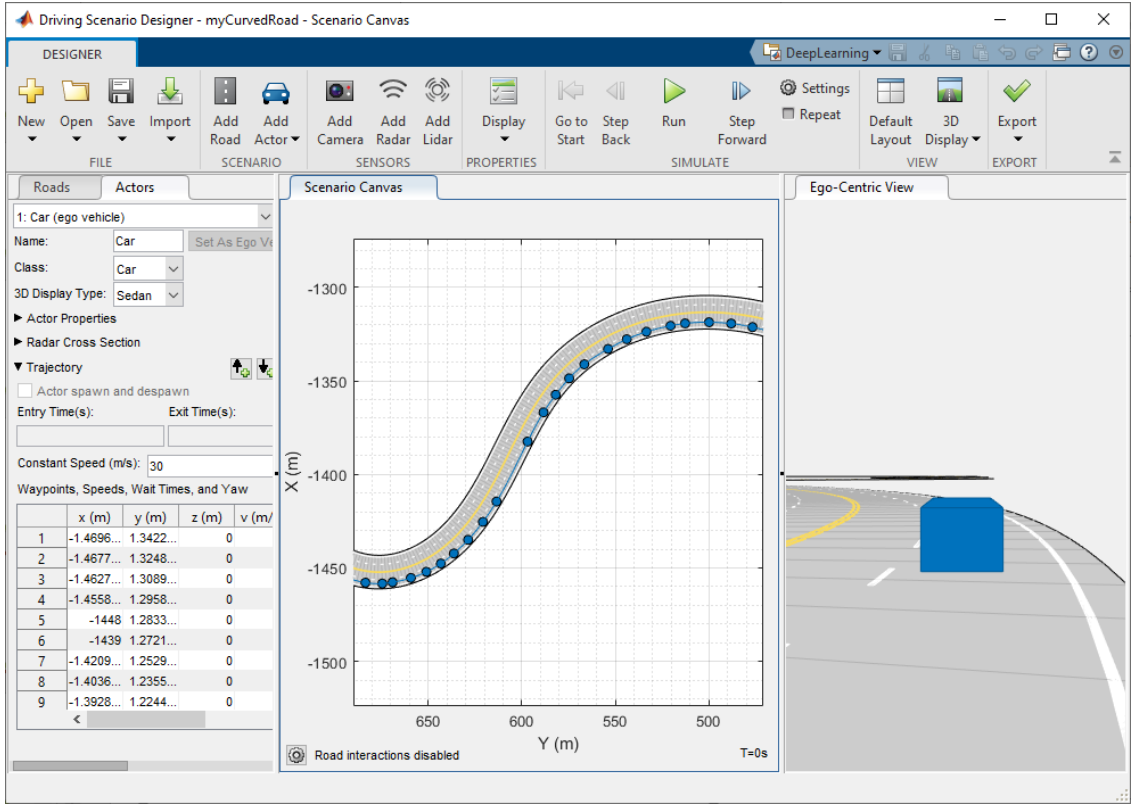
Design scenes
- Roads, Lane markings
- Pre-built scenes (Euro NCAP)

Import roads
- OpenDRIVE, HERE HD Live Map

Add actors
- Size, Radar cross-section (RCS)
- Trajectories

Model sensors
- Vision object detections
- Vision lane detections
- Radar detections
- Lidar point cloud

Export scenarios
- MATLAB code
- Simulink model



Generate stimulus and model sensors (IMU, GPS)

Scenario definition (road, actors, waypoints) → **Trajectory calculation** → **Sensor modeling (IMU, GPS)**

▼ Trajectory

Actor spawn and despawn

Entry Time(s): Exit Time(s):

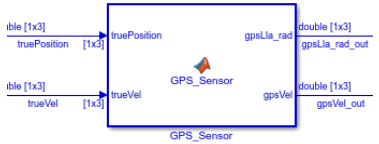
Constant Speed (m/s): 30

Waypoints, Speeds, Wait Times, and Yaw

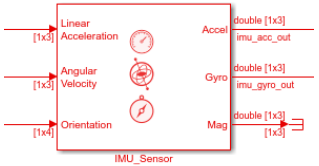
	x (m)	y (m)	z (m)	v (m/s)	wait (s)	yaw (deg)
1	-1.4696...	1.3422...	0	30	0	-86.1052
2	-1.4677...	1.3248...	0	30	0	-79.0934
3	-1.4627...	1.3089...	0	30	0	-66.1304
4	-1.4558...	1.2958...	0	30	0	-59.9762
5	-1448	1.2833...	0	30	0	-54.7399
6	-1439	1.2721...	0	30	0	-48.7037
7	-1.4209...	1.2529...	0	30	0	-45.4141
8	-1.4036...	1.2355...	0	30	0	-45.1651
9	-1.3928...	1.2244...	0	30	0	-46.8670
10	-1.3837...	1.2142...	0	30	0	-49.6887
11	-1.3776...	1.2067...	0	30	0	-52.2234
12	-1.3699...	1.1957...	0	30	0	-58.4887
13	-1.3645...	1.1856...	0	30	0	-65.0658
14	-1.3609...	1.1767...	0	30	0	-71.2725
15	-1.3577...	1.1641...	0	30	0	-79.7196
16	-1.3566...	1.1535...	0	30	0	-89.9488

```
100     % Generate trajectory.
101 -   groundTruth = waypointTrajectory('SampleRate', imuFs, ...
102         'Waypoints', position, ...
103         'TimeOfArrival', t, ...
104         'Orientation', orientation);
105
```

- Position
- Orientation
- Velocity
- Acceleration
- Angular velocity




```
1   function [gpsLla_rad,gpsVel]  =  GPS_Sensor(truePosition,trueVel, ...
2                                               localOrigin, gpsFs)
3
4 -  persistent gps;
5
6 -  if isempty(gps)
7        % Instatiation of GPS Sensor System Object
8 -      gps = gpsSensor('UpdateRate', gpsFs, 'ReferenceFrame', 'NED');
9        % Parameter defintion of GPS Sensor instance
10 -     gps.ReferenceLocation = localOrigin;
11 -     gps.DecayFactor = 0.5;                % Random walk noise parameter
12 -     gps.HorizontalPositionAccuracy = 1.0;
13 -     gps.VerticalPositionAccuracy =  1.0;
14 -     gps.VelocityAccuracy = 0.1;
15   end
16
17 -  [gpsLla_deg,gpsVel] = gps(truePosition,trueVel);
18 -  gpsLla_rad = [deg2rad(gpsLla_deg(1)),deg2rad(gpsLla_deg(2)),gpsLla_deg(3)];
19
```



Develop Automated Driving Applications with MATLAB® and Simulink®

Automated Driving Systems

Environment

Scenes

Scenarios

Vehicles

Sensors

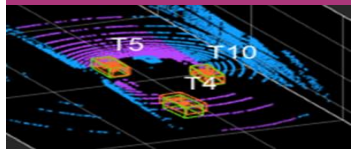
Dynamics

Algorithms

Detection

Localization

Tracking & Fusion



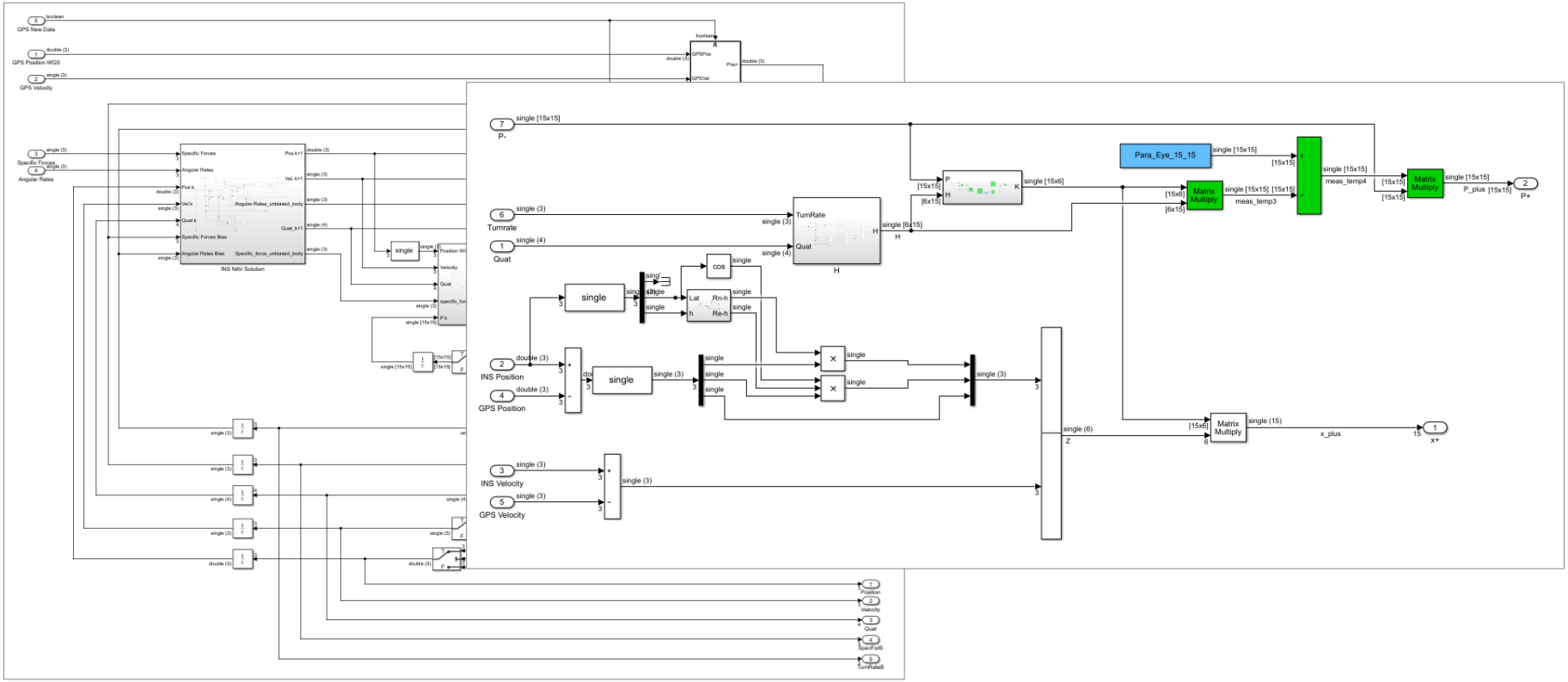
Planning

Decision & Controls

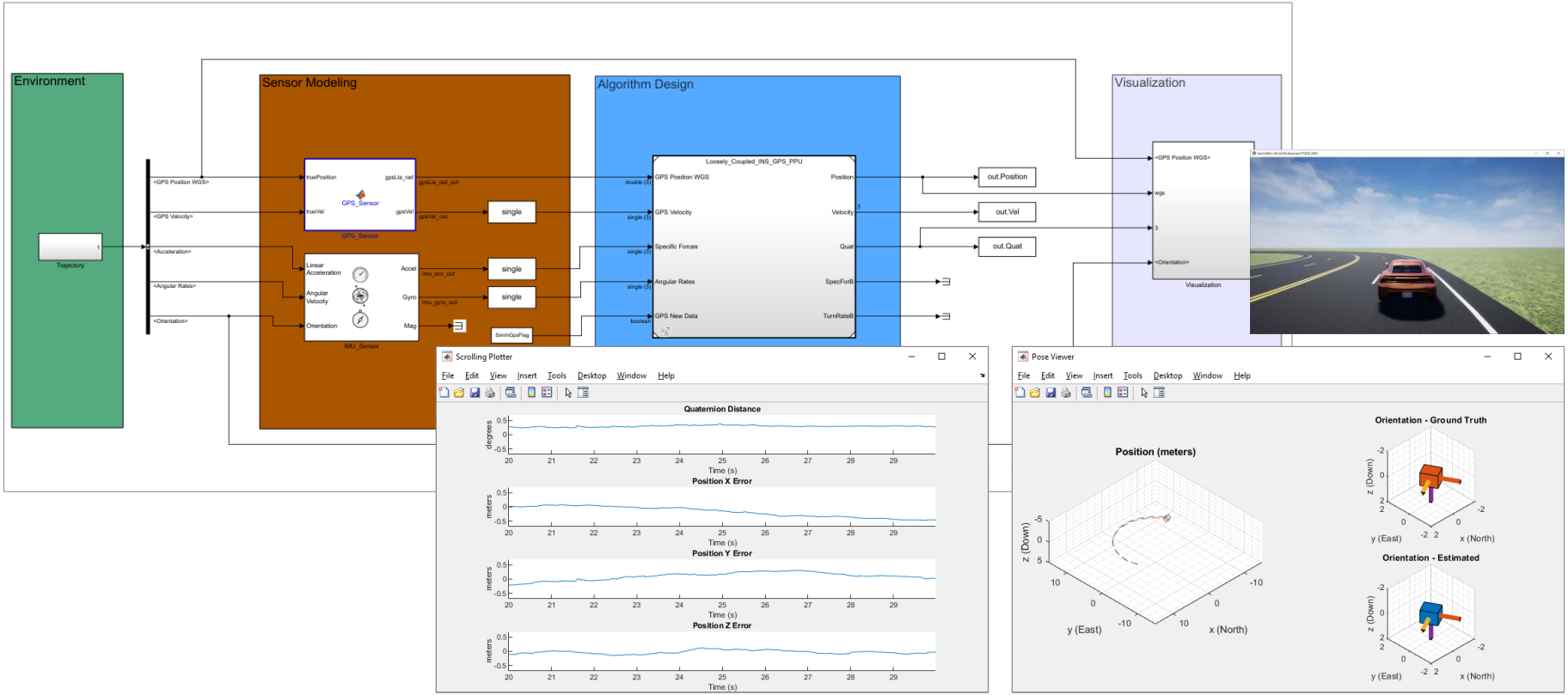
Development Platform

Analyze | Simulate | **Design** | Deploy | Integrate | Test

Design and verify the fusion algorithm



Design and verify the fusion algorithm



Optimal utilization of PPU requires data type selection following step-wise approach for best implementation

Algorithm design

.. using double precision numbers

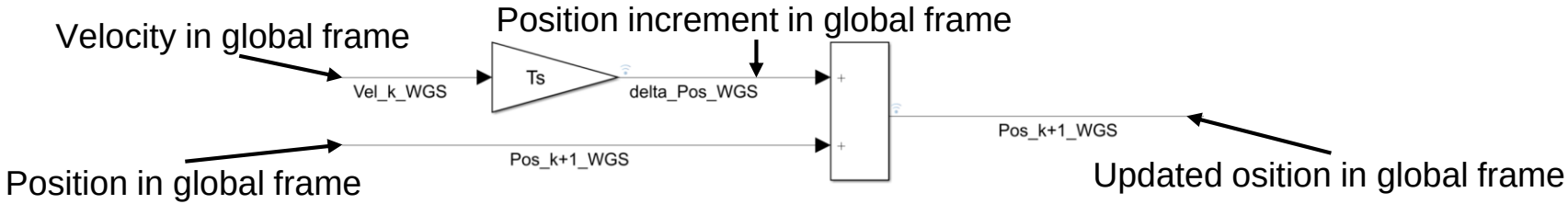
.. all single precision

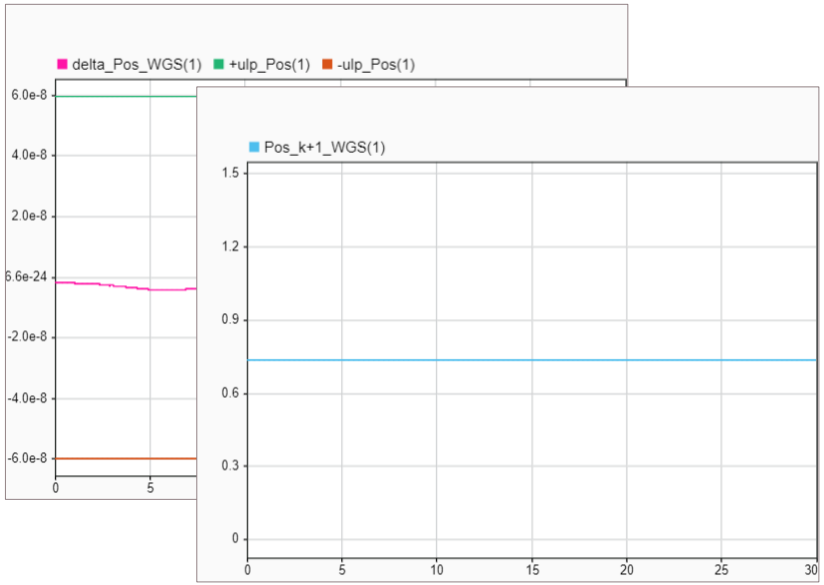
Verification loops

Motivation

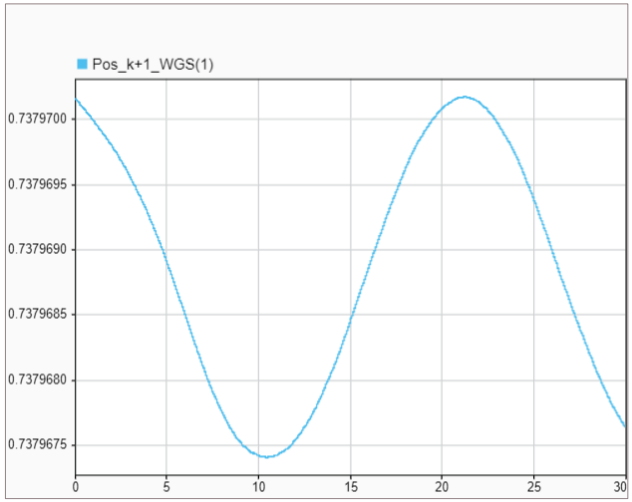
Conversation to reduce memory footprint and enable utilization of SIMD unit

Deeper look into the model





Conversion of sum to double



Optimal utilization of PPU requires data type selection following step-wise approach for best implementation

Algorithm design

.. using double precision numbers

.. all single precision

.. execute position update and correction in double precision

Verification loops

Motivation

Conversation to reduce memory footprint and enable utilization of SIMD unit

Reduced number of double precision calculation in the scalar part of the algorithm

Develop Automated Driving Applications with MATLAB® and Simulink®

Automated Driving Systems

Environment

Scenes

Scenarios

Vehicles

Sensors

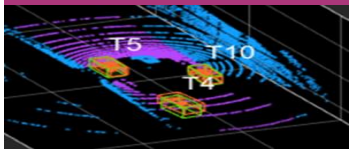
Dynamics

Algorithms

Detection

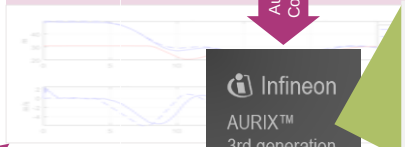
Localization

Tracking & Fusion



Planning

Decision & Controls

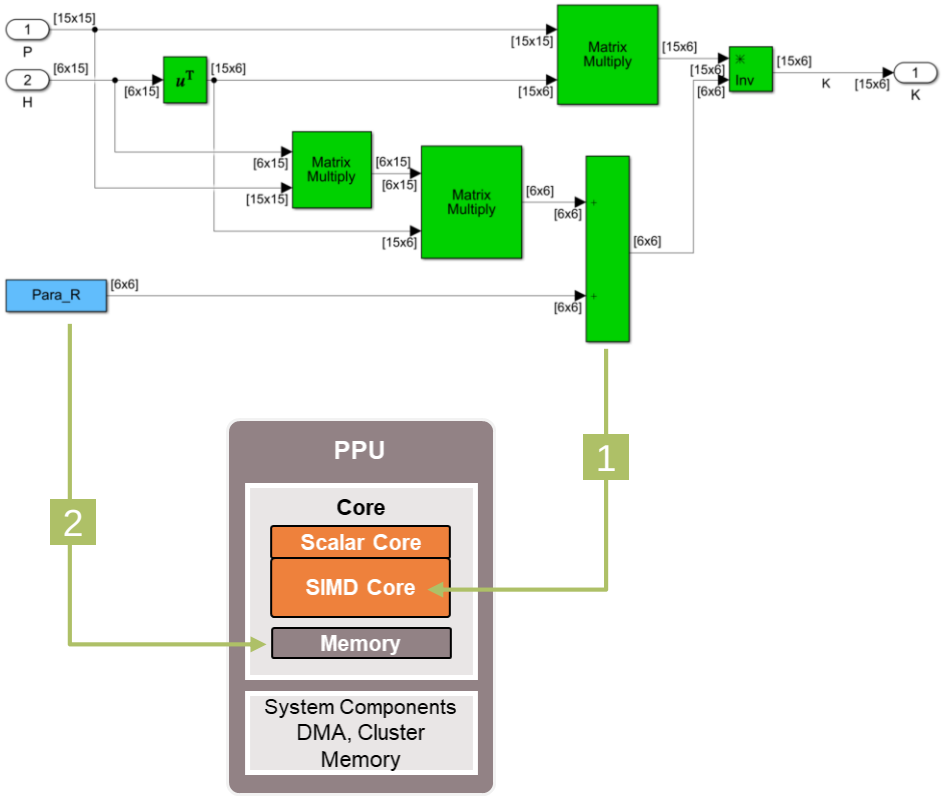


Automatic Code Gen.

Development Platform

Analyze | Simulate | Design | **Deploy** | Integrate | Test

Dedicated tool flow enables implementation from model-level to target device code



1 Code Replacement Library

Name	Implementation	NumIn	In1Type	In2Type	OutType
RTW_OP_ADD	ifx_mm_add_f32	2	single[5 5; Inf Inf]	single[5 5; Inf Inf]	single[5 5; Inf Inf]
RTW_OP_ELEM_MUL	ifx_ms_scale_f32	2	single[5 5; Inf Inf]	single	single[5 5; Inf Inf]
RTW_OP_ELEM_MUL	ifx_ms_scale_f32	2	single	single[5 5; Inf Inf]	single[5 5; Inf Inf]
RTW_OP_MINUS	ifx_mm_minus_f32	2	single[5 5; Inf Inf]	single[5 5; Inf Inf]	single[5 5; Inf Inf]
RTW_OP_MUL	ifx_mm_mul_f32	2	single[5 5; Inf Inf]	single[5 5; Inf Inf]	single[5 5; Inf Inf]
RTW_OP_RDIV	ifx_mm_rdiv_f32	2	single[5 5; Inf Inf]	single[5 5; 16 16]	single[5 5; Inf Inf]
RTW_OP_TRANS	ifx_m_transpose_f32	1	single[5 5; Inf Inf]		single[5 5; Inf Inf]

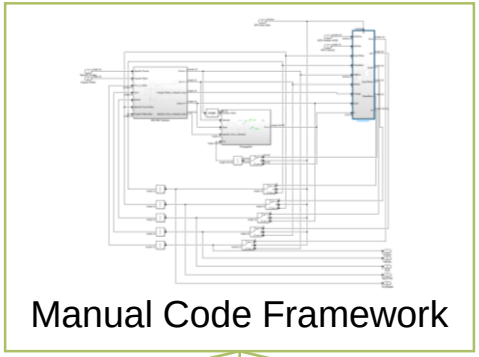
2 Custom Storage Class

Definition:

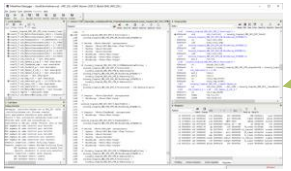
```
__vccm
/* CSC definition comment generated by default */
const DATATYPE DATANAME[DIMENSION] [= {...}];
```

There are different options for deploying the code on a target

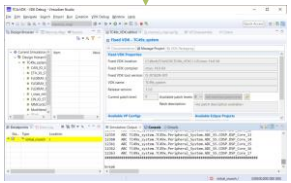
- Code is packed into a manual generated framework
- It can be deployed to several platforms



Manual Code Framework



PPU only simulator

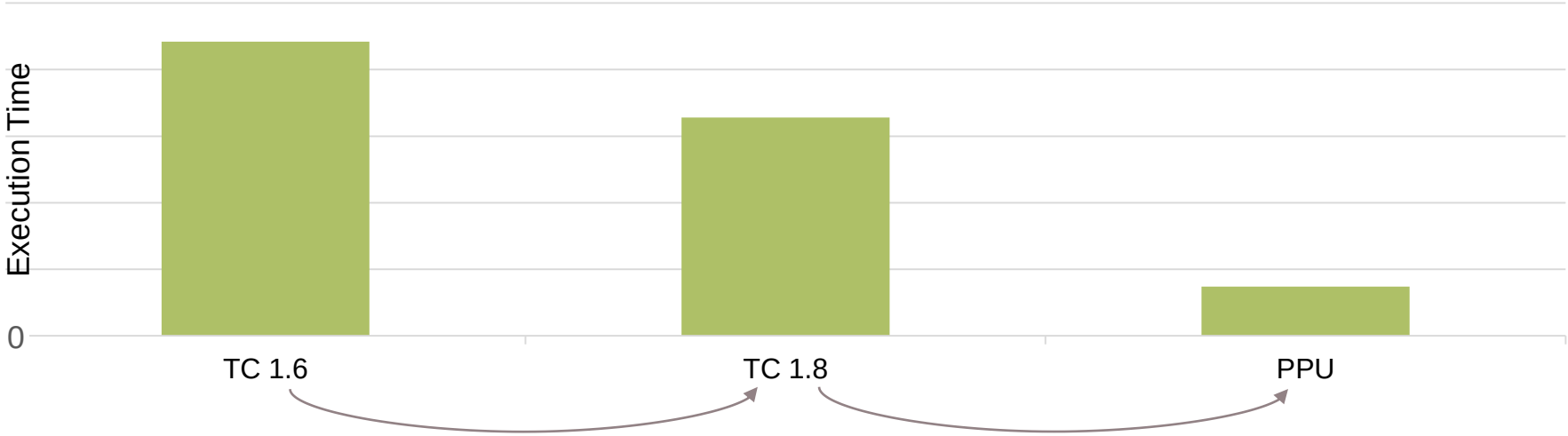


Virtual prototype

Real silicon

as soon as available

Execution timing improves significantly in new AURIX™ generation; especially when implementing on PPU



- Frequency scaling
- Double precision floating point support

- Utilizing the SIMD capabilities

Summary

- Next AURIX™ generation will bring a huge step forward in the vector processing domain
- Model based software development support of the PPU will start in the summer using block replacement based code replacement libraries
- Already first version of model based code generation support package will bring a significant enhancement in performance
- In the future we will focus on further improvements especially with respect to the whole SoC integration

Part of your life. Part of tomorrow.