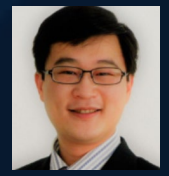
Valeo

SMART TECHNOLOGY
FOR SMARTER CARS

Developing The Next Generation LiDAR with Model-Based Design

Chenji TU, Valeo Detection Systems GmbH



MathWorks
AUTOMOTIVE
CONFERENCE 2024

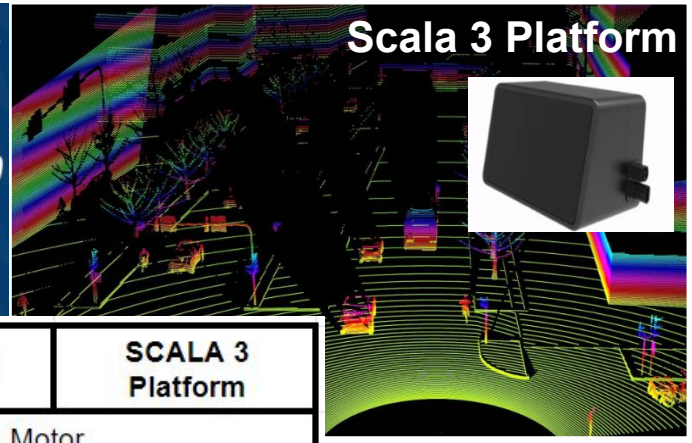
Agenda

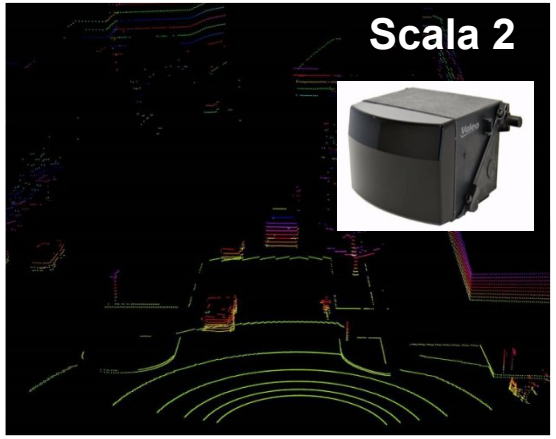
- Overview of Valeo's ADAS Portfolio and LiDAR (Scala™) Evolution
- Motivation to adopt Model-Based Design (MBD)
- LiDAR Development with MBD
- Sensor's Azimuth Accuracy, Precision, Missing Shot Improvement
- FPGA Area, Speed Optimization and Verification with MBD
- Key Takeaways
- Gratitude to MathWorks Professionals and Consultants

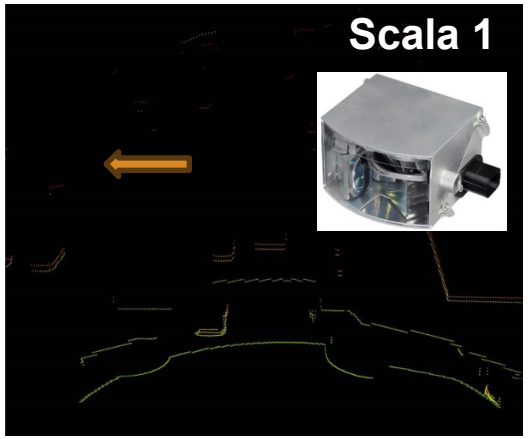
Overview of Valeo's ADAS Portfolio and LiDAR (Scala™) Evolution

Valeo, a mobility leader, adopts **Model-Based Design (MBD)** for advanced **LiDAR** in **automotive ADAS**.

LiDAR
Automated Driving

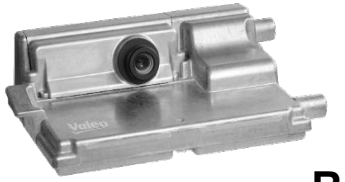






Valeo LiDAR Projects	SCALA 1	SCALA 2	SCALA 3 Platform
Tx	Infrared Laser, Motor		
Rx (Vertical Pixel #)	APD (3-Pixel)	APD (16-Pixe)	SPAD (360-Pixel)
H. FoV	145° (+/-72.5°)	133° (+/-66.5°)	120° (+/-60°)
V. FoV	3.2°	10°	0.07° * 360 = 25.2°
H. Res.	0.25°	0.125° / 0.25°	0.2° / 0.1°
V. Res.	0.8°	0.6°	0.07°
Motor RPM	750	750	150 - 600
FPS	25	25	20
Points per Scan	5k	31k	650k
Points per second	131k	783k	12.9M

Smart Front Camera
Automated Driving



Mid-Range Radar
Automated Driving

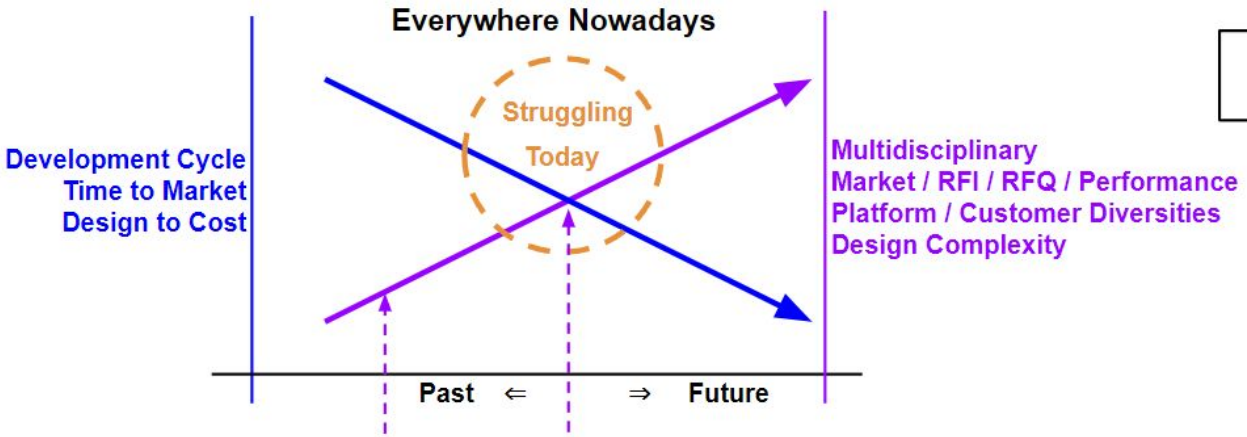
Satellite Camera
Parking Assistance

Rain Sensor
ADAS

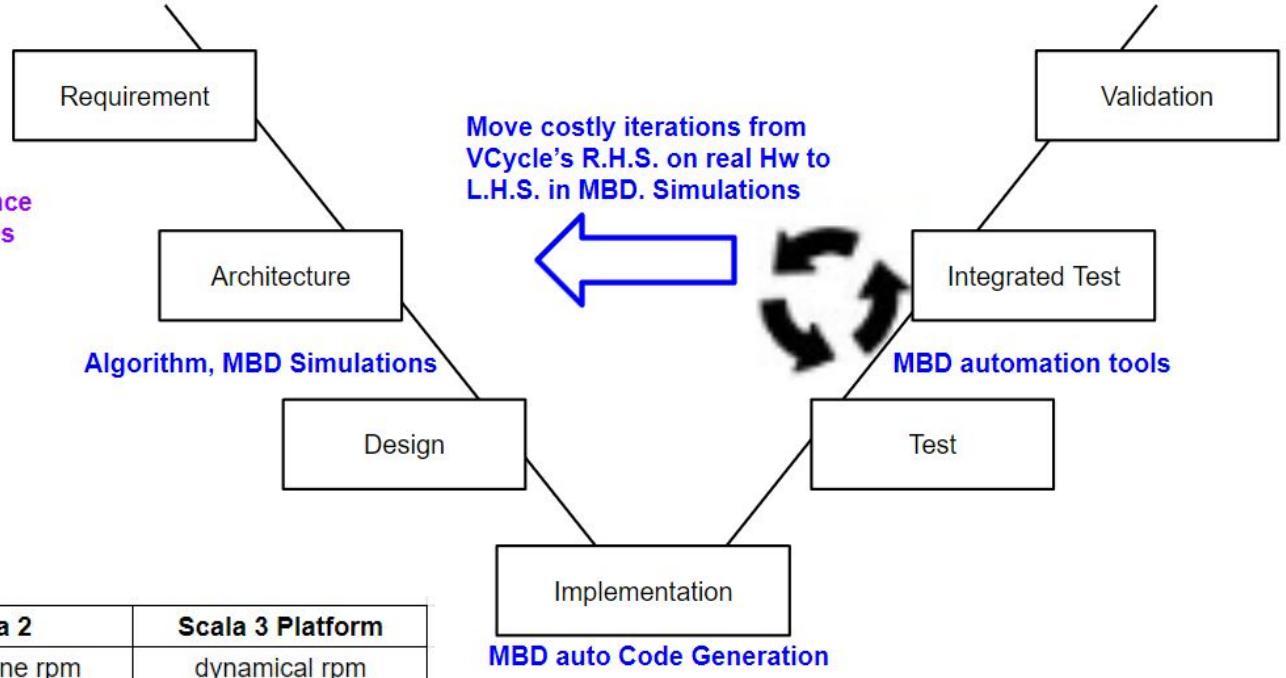
Ultrasonic Sensor
Parking Assistance

Motivation to adopt Model-Based Design (MBD)

MBD reduces time to market and design cost in a multidisciplinary, exponentially complex safety critical development process.



Scala 2 AASC ⇒ Scala 3:
Simulation Complexity: ✕ **9.3**
Accuracy / Precision Req.: ✕ **2**
Time to Market: ÷ **2**
Com. Perf Drop (Jitter) : ÷ **1.5**

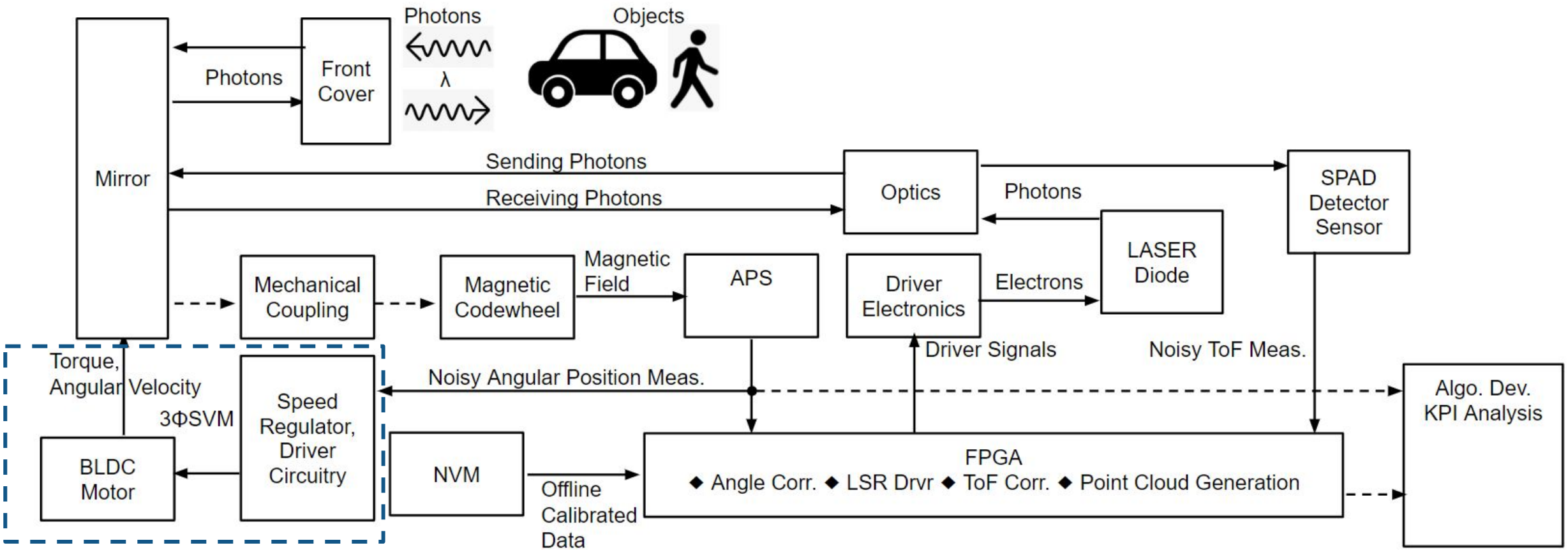


Projects	Scala 2	Scala 3 Platform
Motor Speed	fixed to one rpm	dynamical rpm
Accuracy, Precision	lower, fixed to one rpm	higher, dynamical rpm
RPM Tracking	No	Yes
Raw Resolution	lower	higher
Config. Extraploated Res.	No	Yes
Config. Scan Patterns	No	Yes
Time to Market	5 yr	2.5 yr

MBD is the key to efficiency

LiDAR Development with MBD – Lidar System Overview

Overview of LiDAR System Elements and Signal Chain.



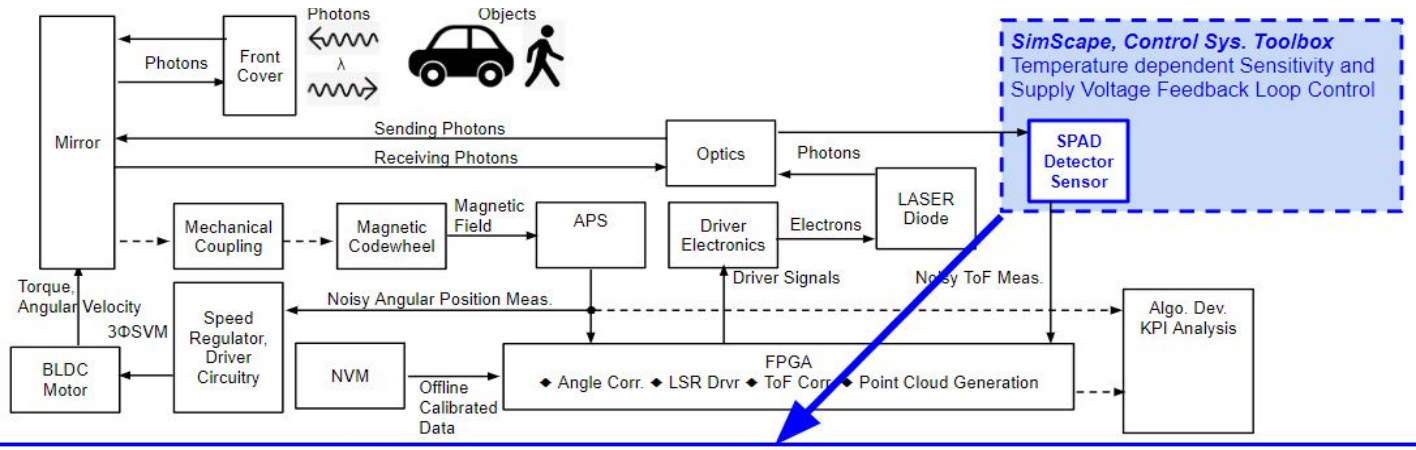
LiDAR point cloud begins with motor driven laser shots

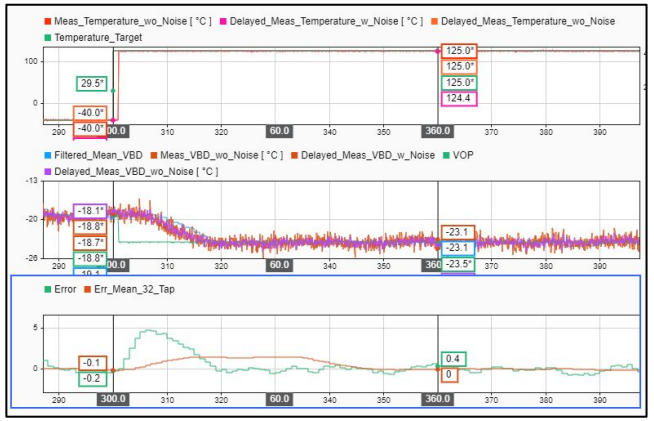
LiDAR Development with MBD – SPAD

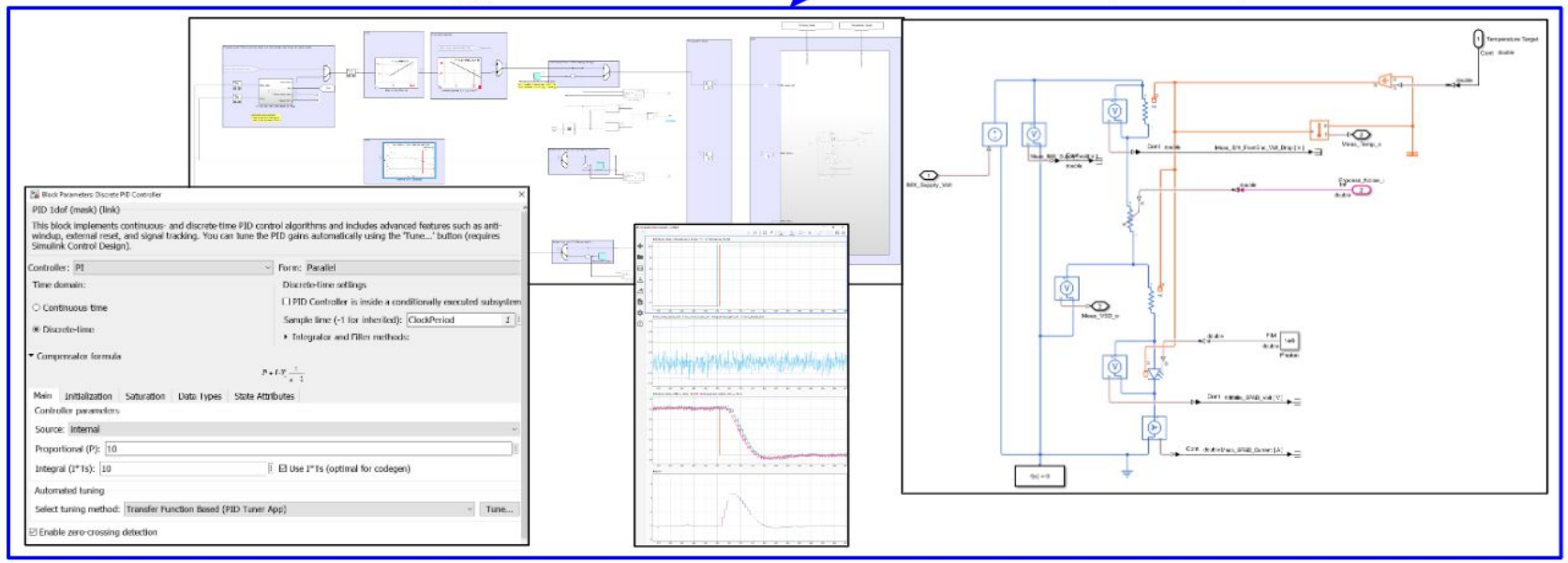
Temperature Dependent Detector Sensitivity Feedback Loop Control Development is based on **Simscape** (Thermal Port), **DSP** (Noise Floor), **Control System Toolbox** (PI).

Quick system behavioral modeling, simulation, and auto code generation for controller

- Diode, Power Supply, DAC,...
- Temperature-dependent plant and setpoints
- Controllability scenarios
- Delay, noise floor, SNR,...
- PI controller, LPF targeting ARM





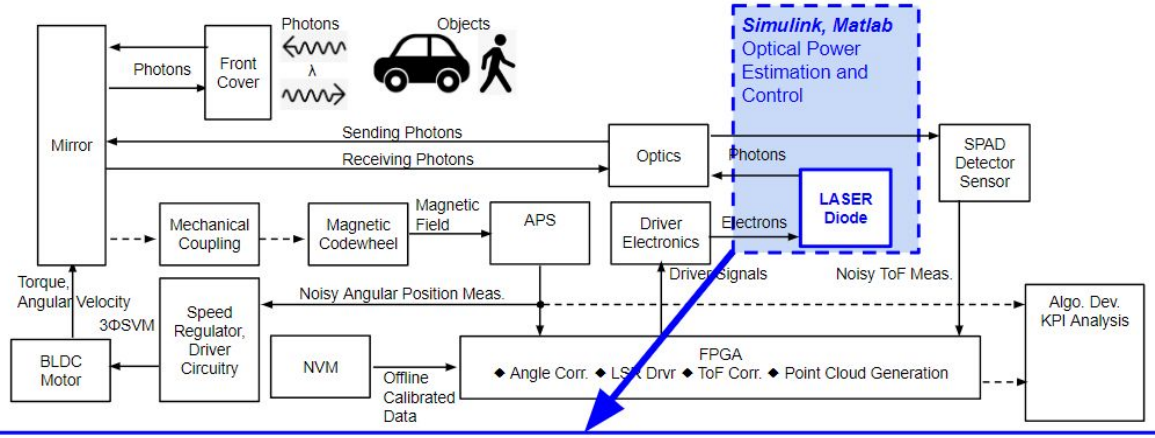


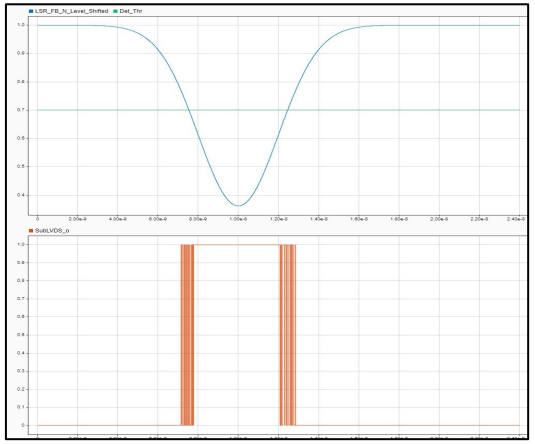
LiDAR Development with MBD – Laser Power Management

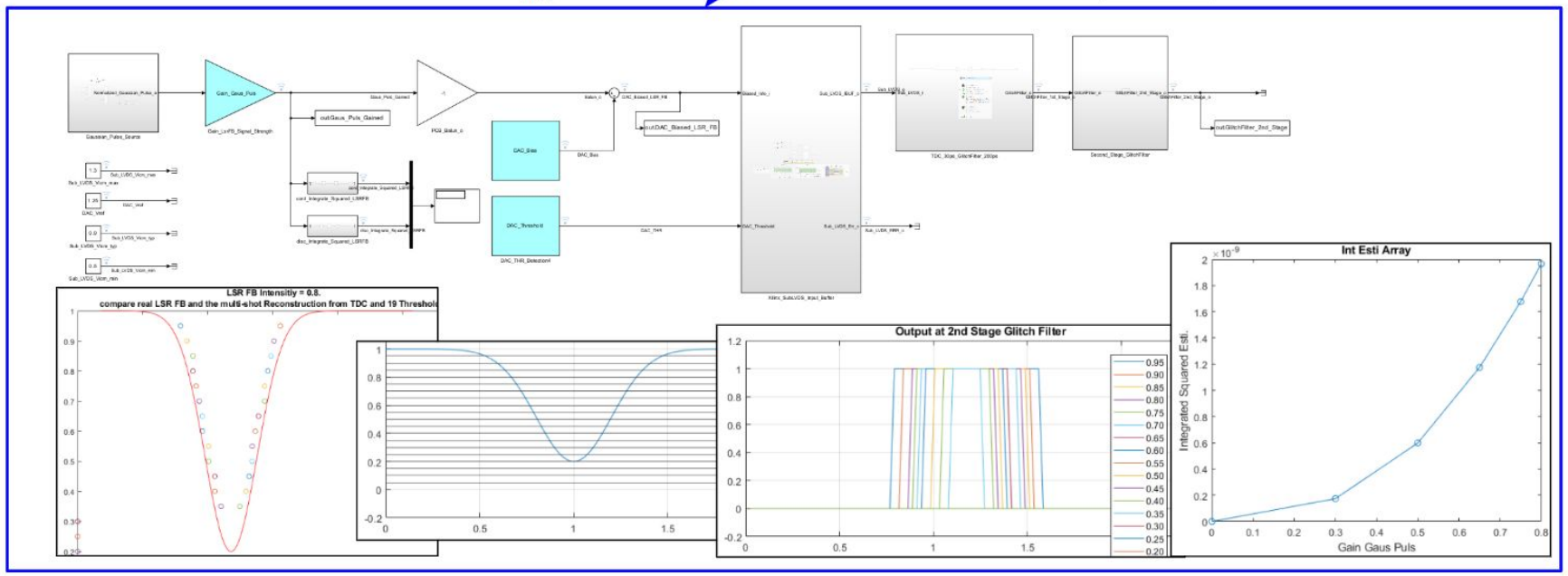
Laser Power Estimation Development (Range, Eye Safety) is based on **Simulink** (Signal Profile, Detection, High-Speed Sampling) and **MATLAB, Statistics Toolbox** (Correlation, Algorithm).

Quick System Behavioral Modeling, Simulation and Algorithm Evaluation:

- Optical Gaussian pulse
- Xilinx differential input buffer
- DAC threshold swiping
- Time-to-digital conversion
- Pulse profile reconstruction
- Correlation, power estimation





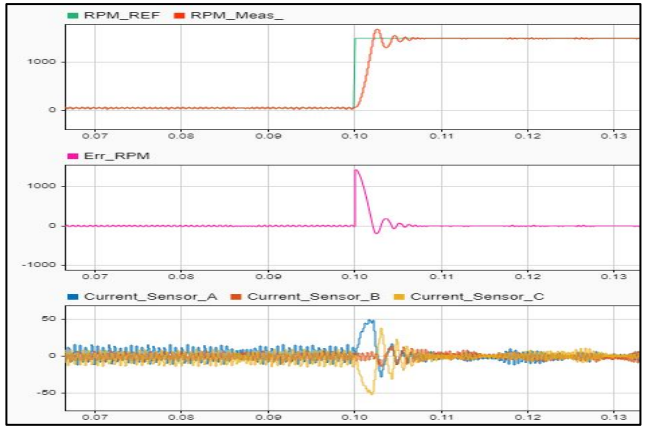


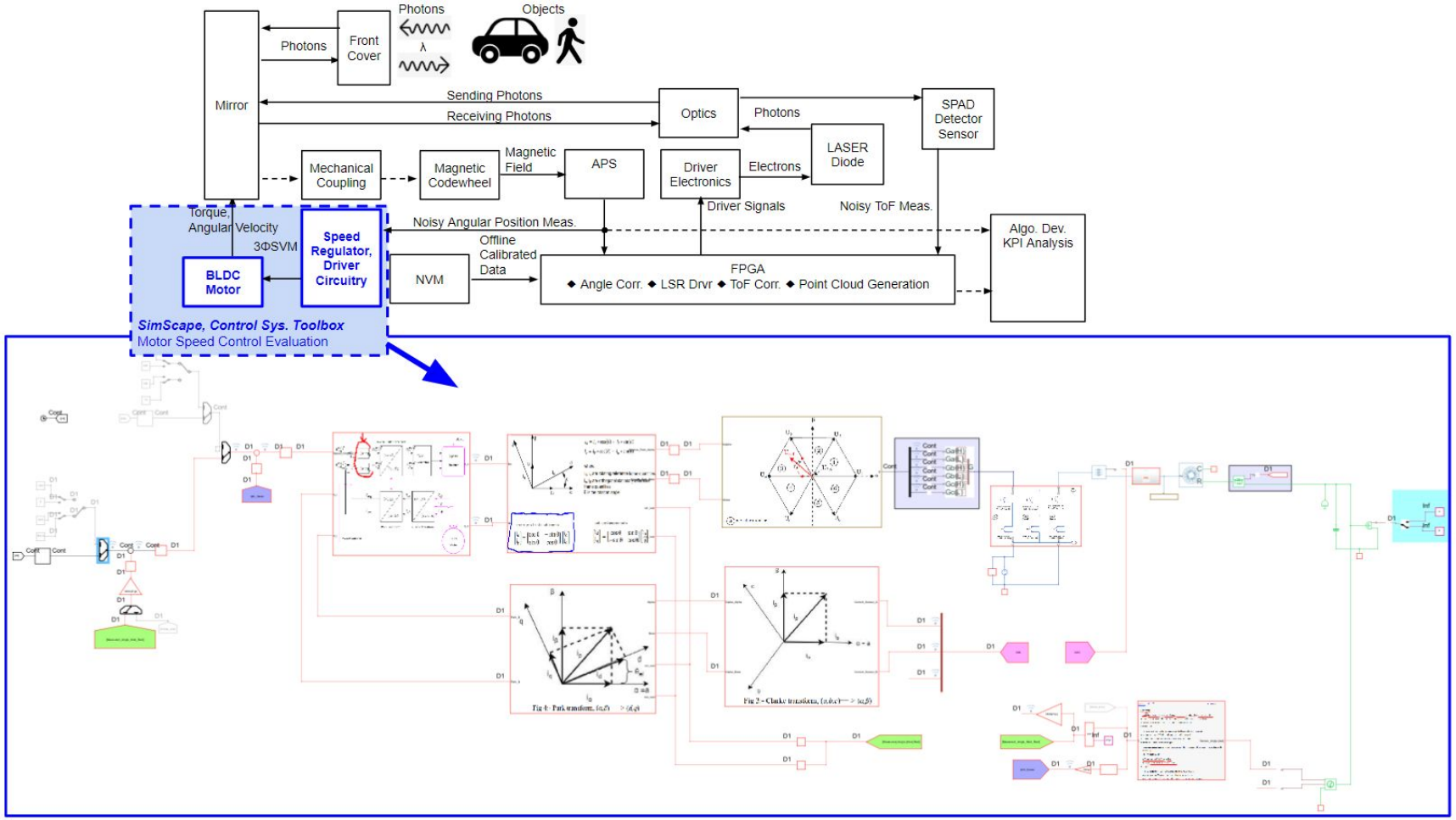
LiDAR Development with MBD – Motor Control

Motor Control improvement investigation is based on **SimScape** (BLDC, SVPWM, Gate Driver) and **Control System Toolbox** (Control Algorithms).

Quick system behavioral modeling, simulation, and Algorithm Evaluation:

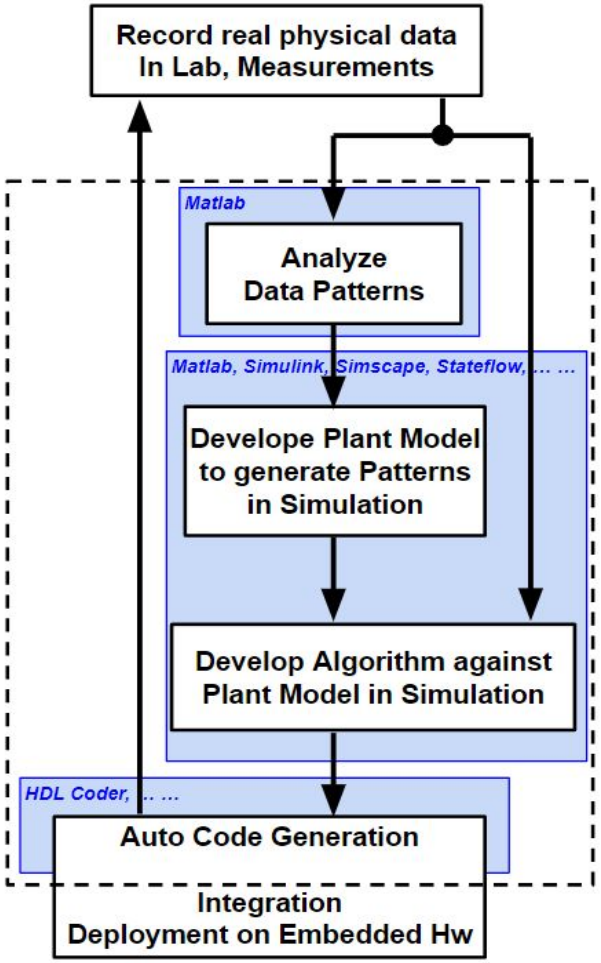
- Investigate potential improvements in existing motor system design for speed regulation stability, position control, noisy APS, etc.
- Collaborating with MathWorks consulting services

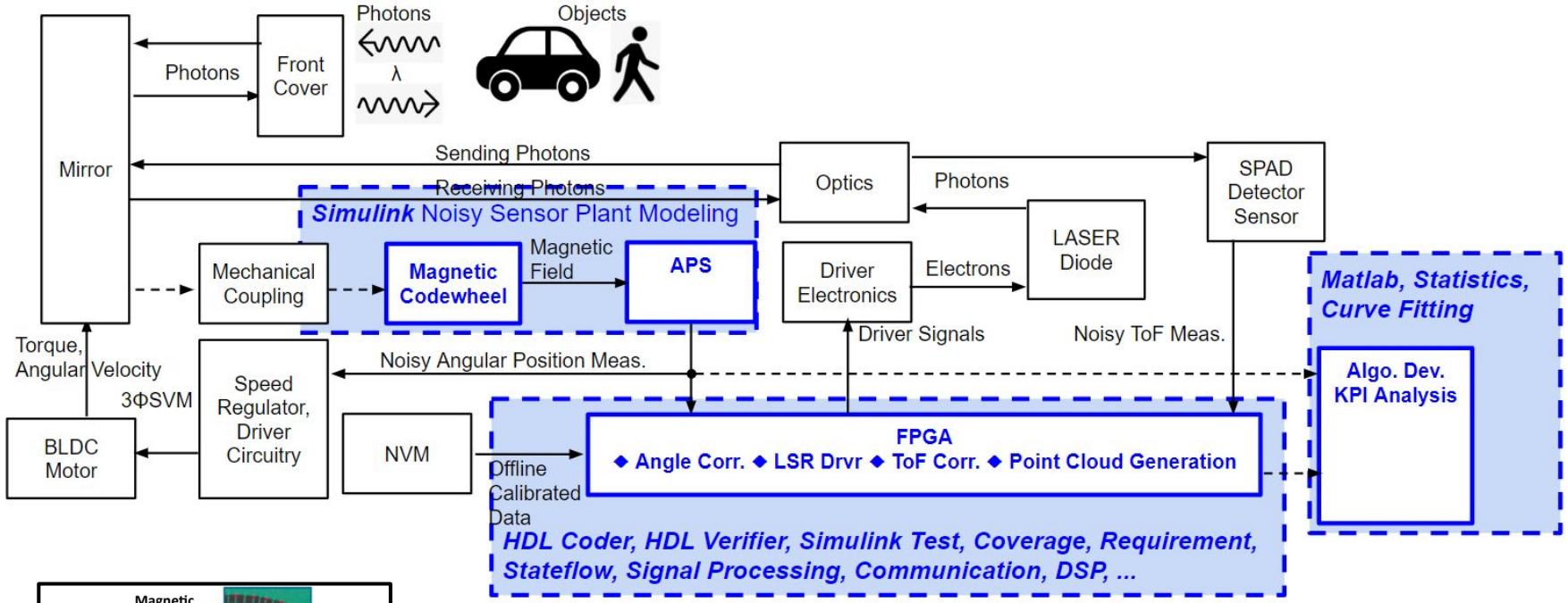


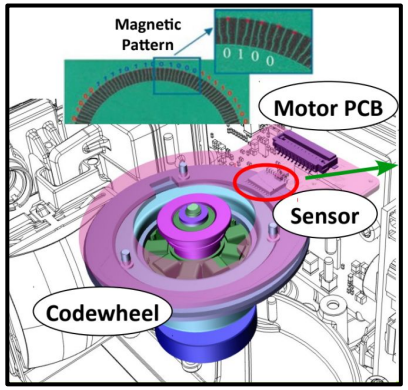


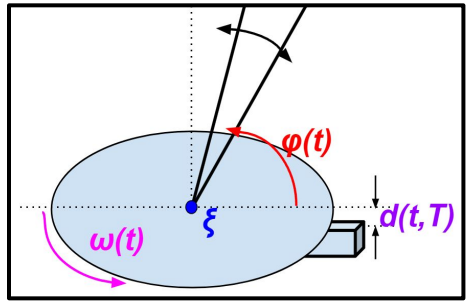
LiDAR Development with MBD – Motor Angle Decoder & Laser Driver

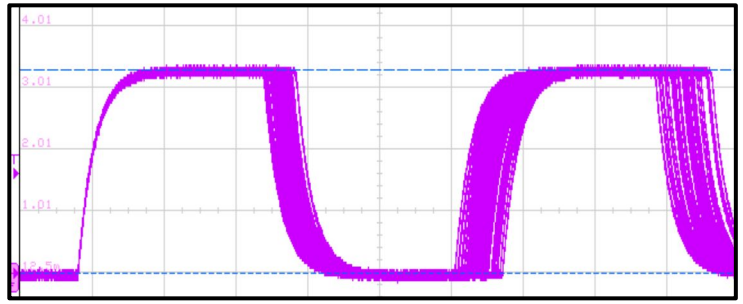
Motor Angle Decoder and Laser Driver development is based on **HDL Coder, Stateflow, Simulink,...**





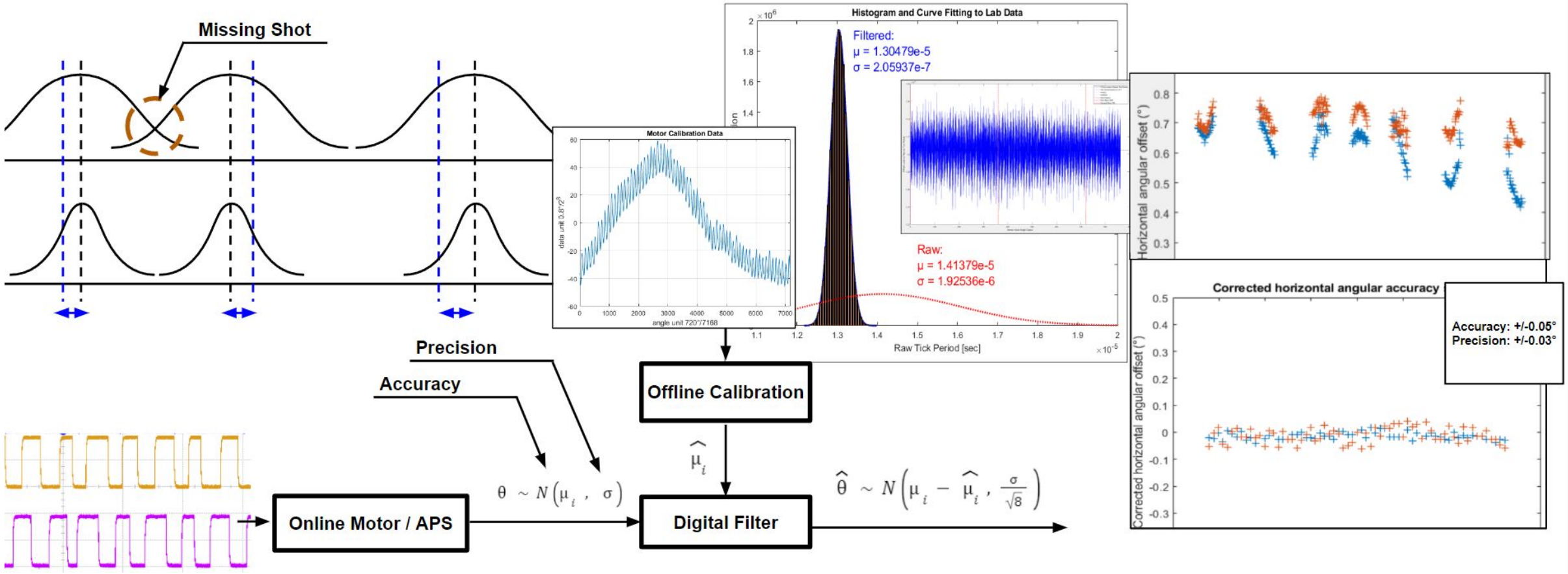






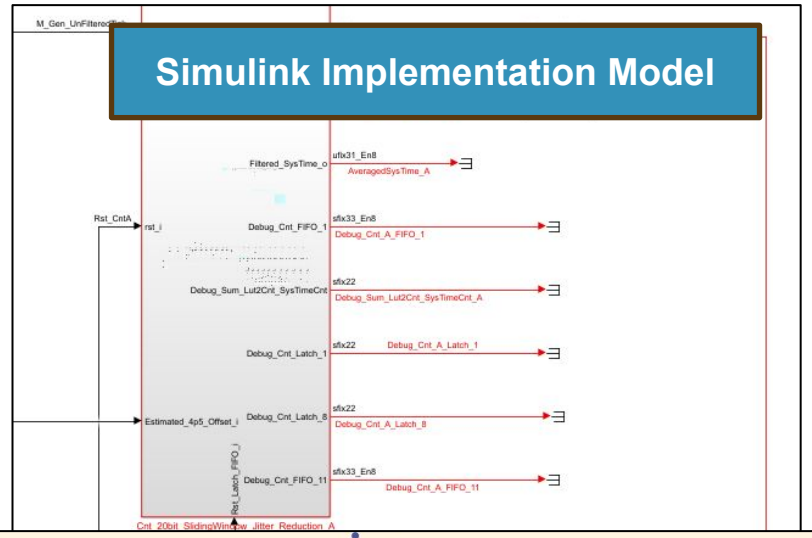
Sensor's Azimuth Accuracy, Precision, Missing Shot Improvement I

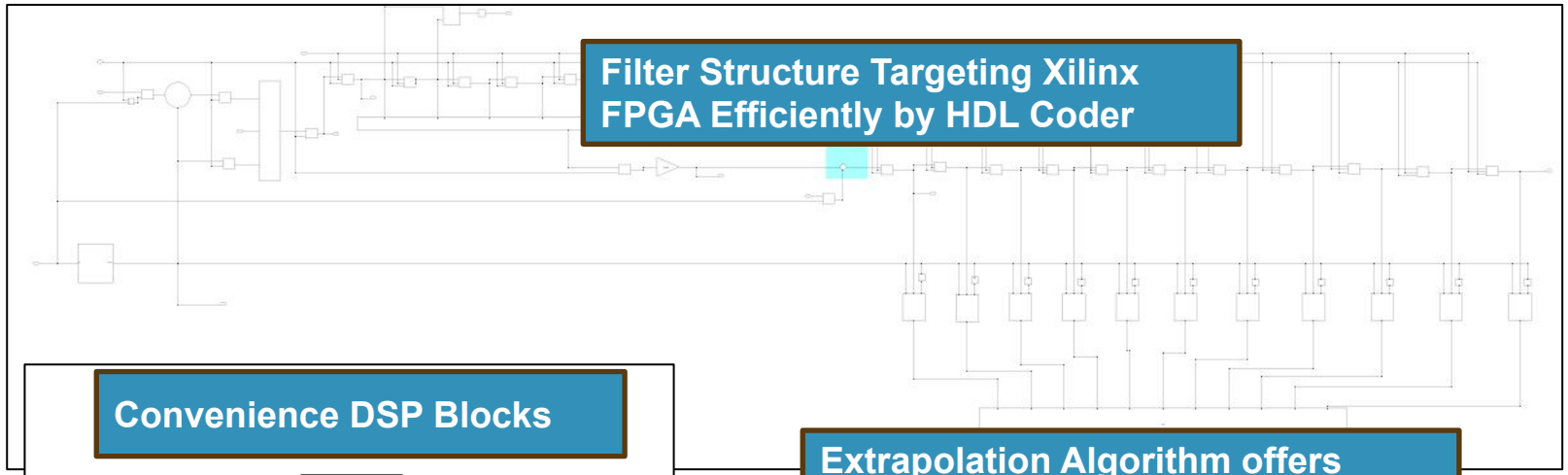
MBD is the cornerstone in development, from ideas to algorithms with automatic production code generation, ISO / ASPICE process compliance till KPI / requirements fulfillment of the product.

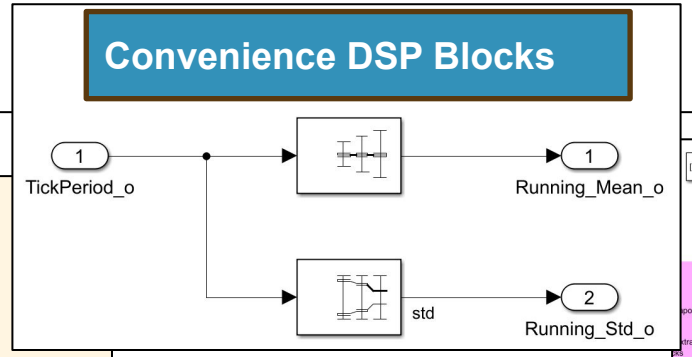


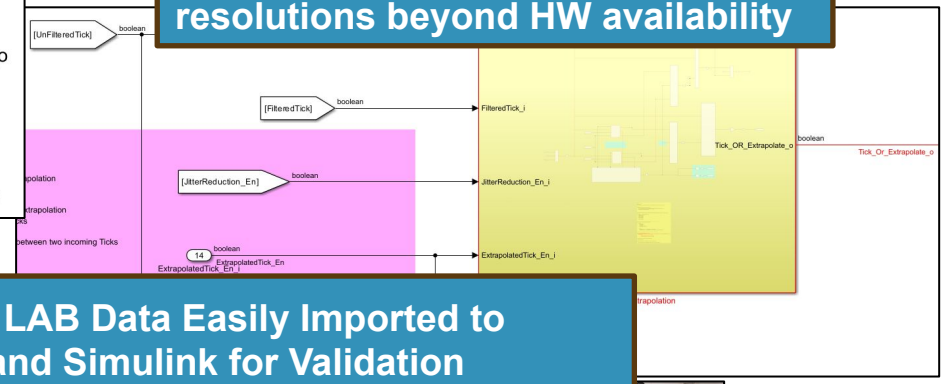
Sensor's Azimuth Accuracy, Precision, Missing Shot Improvement II

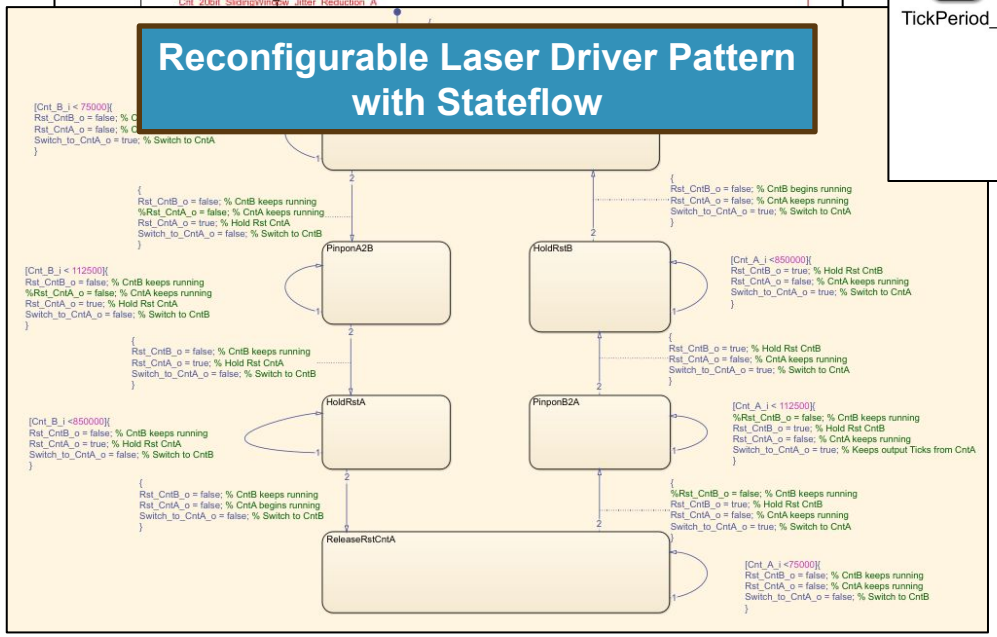
Easy to use tools and Apps in an integrated, consistent development environment with **MBD**

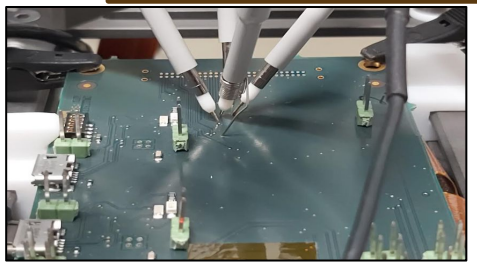


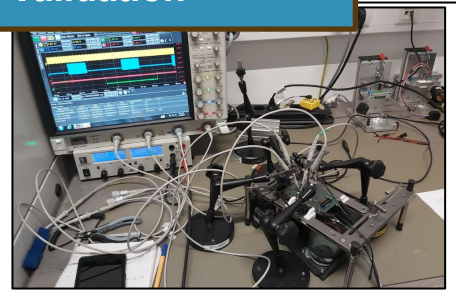








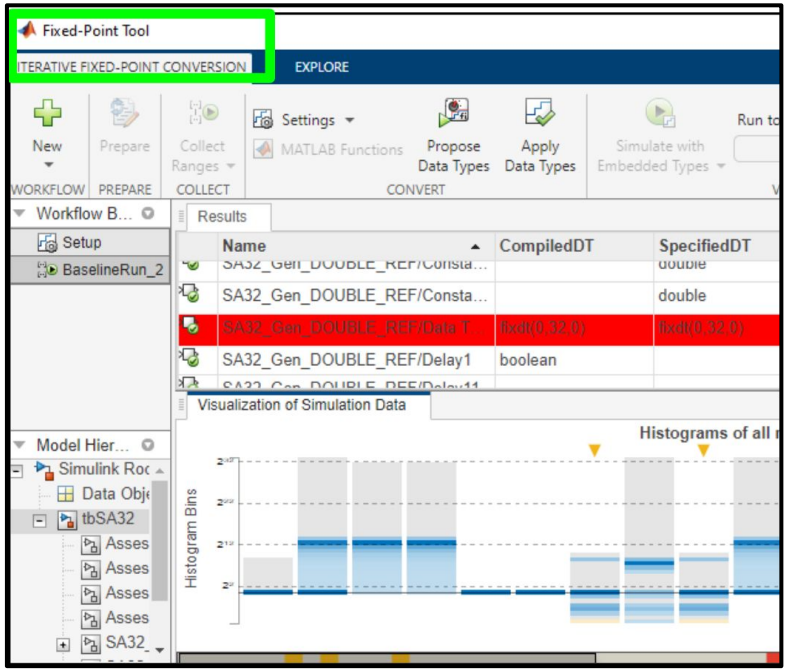


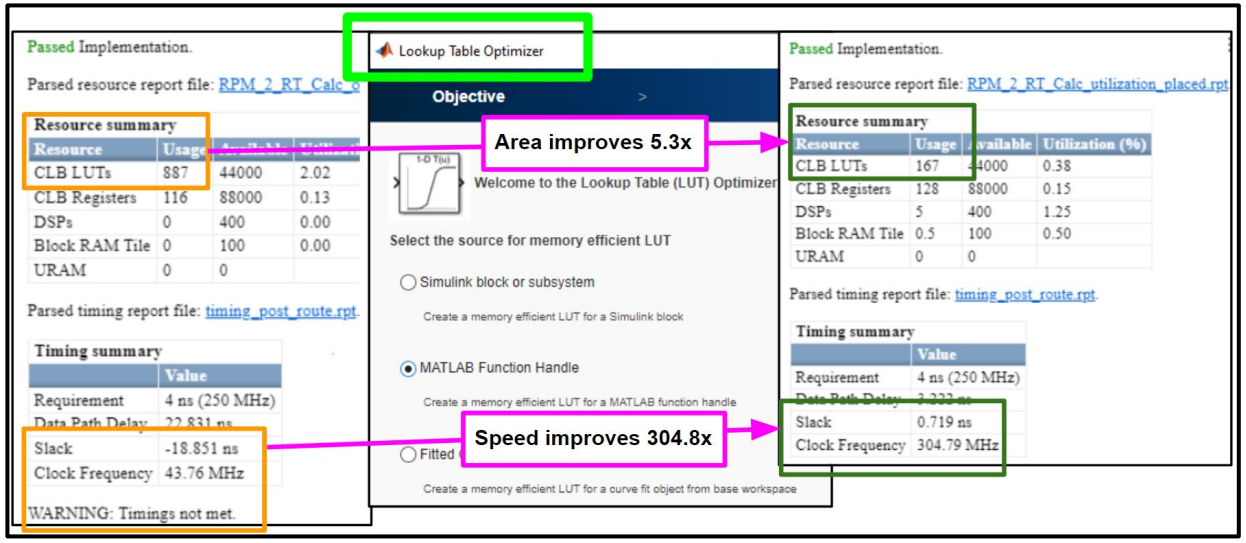


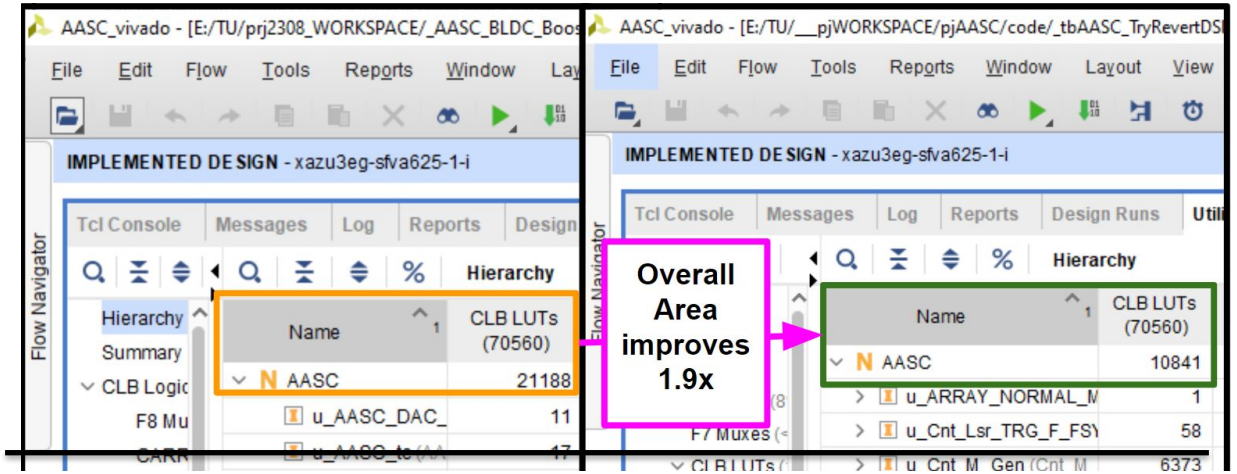
Recorded LAB Data Easily Imported to MATLAB and Simulink for Validation

FPGA Area and Speed Optimization with MBD

- **Fixed-Point Designer** detects overflows and potential errors, proposes HW-efficient data types
- **Lookup Table Optimizer** maps arithmetics to RAM, helps with trade-off exploration and design optimization



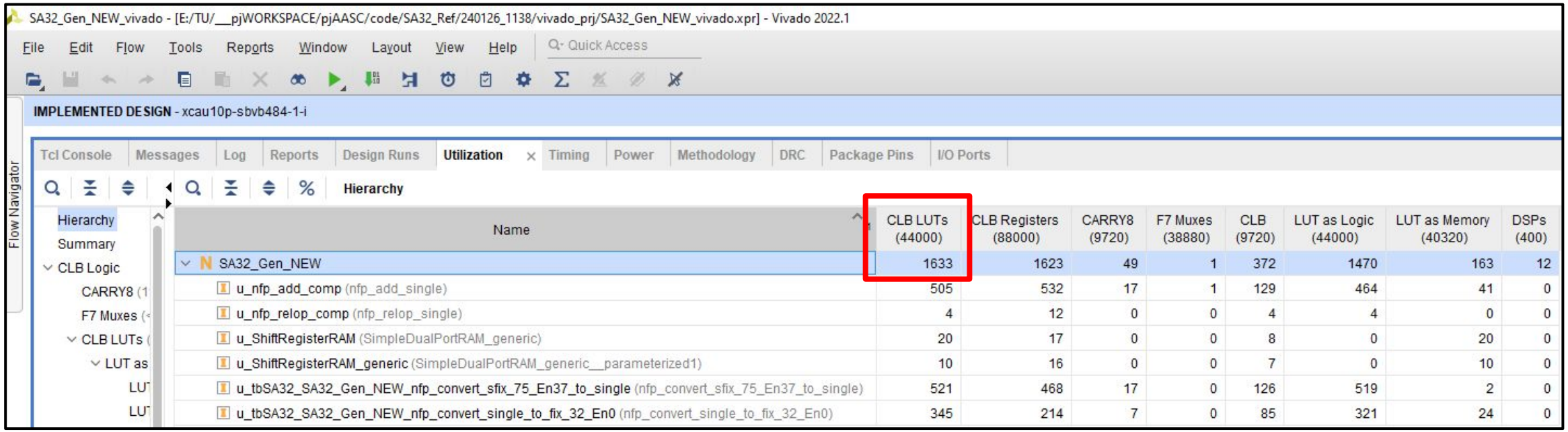




FPGA Area and Speed Optimization with MBD - Example

Example: The original design of Scan Angle for Point Cloud Data Structure has overflow and consumes many FPGA Resources.

- Overflow detected in design
- Consumes many CLB LUTs

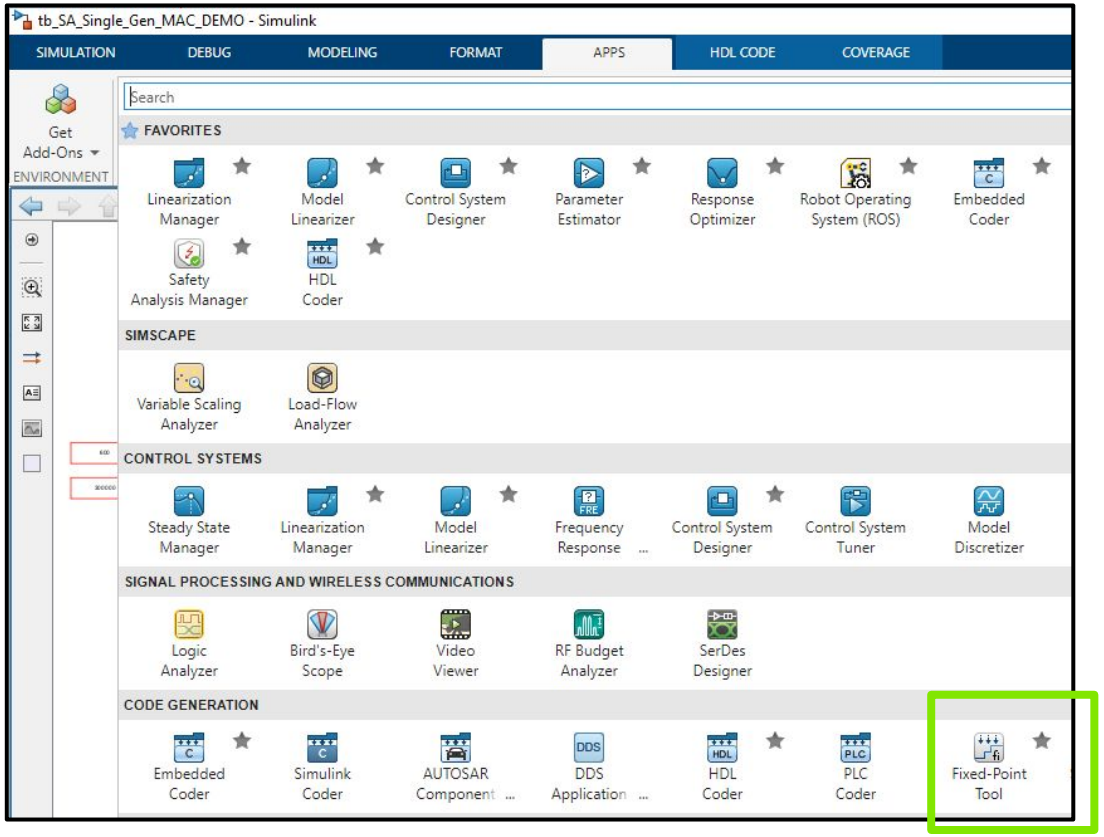


Name	CLB LUTs (44000)	CLB Registers (88000)	CARRY8 (9720)	F7 Muxes (38880)	CLB (9720)	LUT as Logic (44000)	LUT as Memory (40320)	DSPs (400)
SA32_Gen_NEW	1633	1623	49	1	372	1470	163	12
u_nfp_add_comp (nfp_add_single)	505	532	17	1	129	464	41	0
u_nfp_relop_comp (nfp_relop_single)	4	12	0	0	4	4	0	0
u_ShiftRegisterRAM (SimpleDualPortRAM_generic)	20	17	0	0	8	0	20	0
u_ShiftRegisterRAM_generic (SimpleDualPortRAM_generic__parameterized1)	10	16	0	0	7	0	10	0
u_tbSA32_SA32_Gen_NEW_nfp_convert_sfix_75_En37_to_single (nfp_convert_sfix_75_En37_to_single)	521	468	17	0	126	519	2	0
u_tbSA32_SA32_Gen_NEW_nfp_convert_single_to_fix_32_En0 (nfp_convert_single_to_fix_32_En0)	345	214	7	0	85	321	24	0

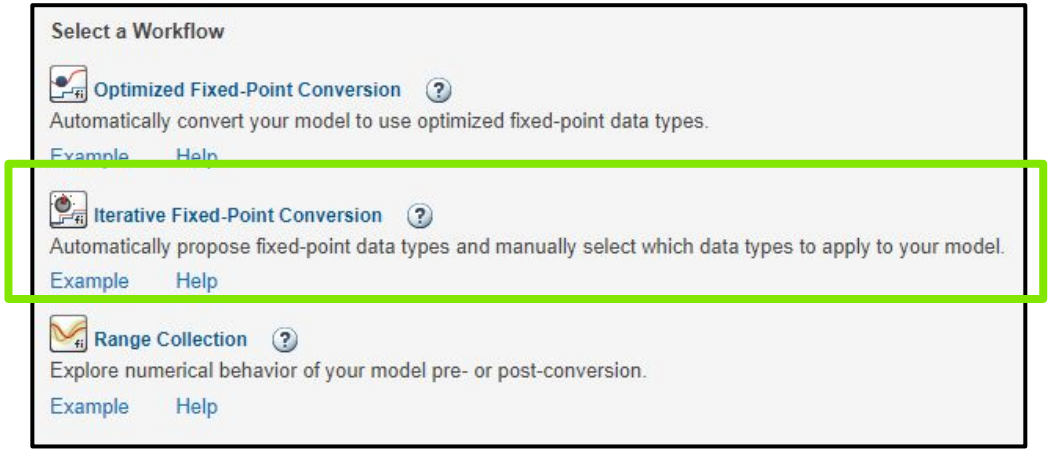
FPGA Area and Speed Optimization with MBD – Fixed-Point Tool I

Use the App "**Fixed-Point Tool**" to convert the data types used in the design under optimization to more efficient ones, i.e., given a defined tolerance, to analyze over/underflow in the signal chains, to automatically propose improved data types, leading to reduced Area.

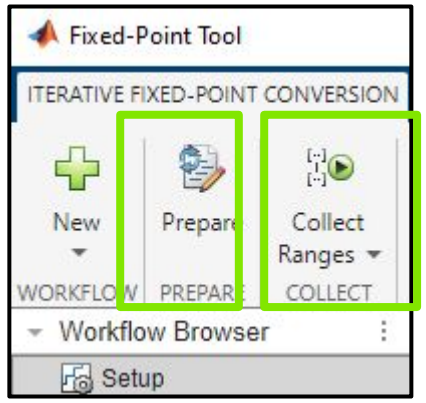
(1) Open Fixed Point Tool App



(2) Select Iterative Fixed-Point Conversion



(3) Prepare to create restore point

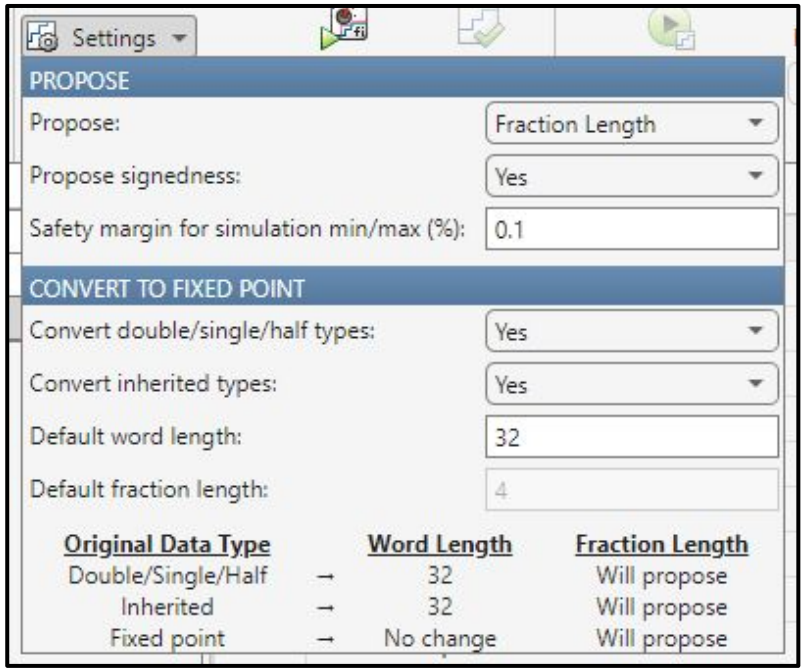


(4) To automatically collect signals' data ranges driven by a full range test vector

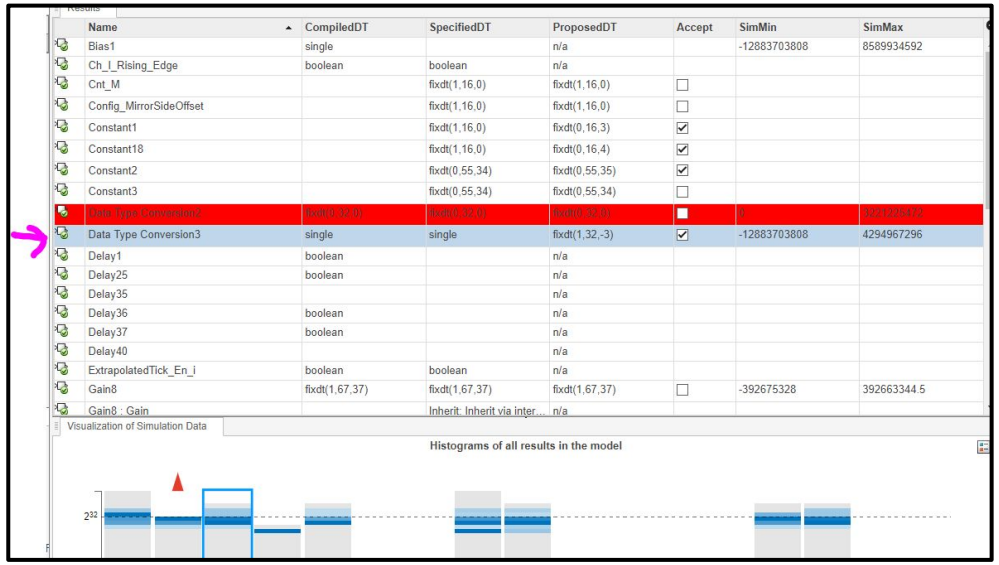
FPGA Area and Speed Optimization with MBD – Fixed-Point Tool II

Step-by-Step follow the intuitive flow in the App

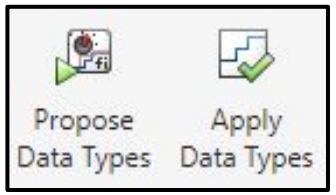
(5) User's inputs directive for Optimization Tool's searching process

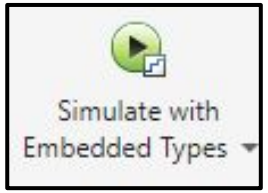


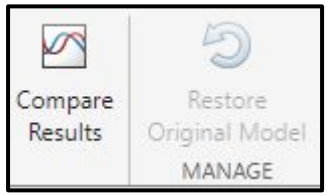
(6) Tool automatically detects critical Overflow and Underflow. Evaluate. Go back to (5) and change directive, or analyze then change simulink design to solve critical issues.

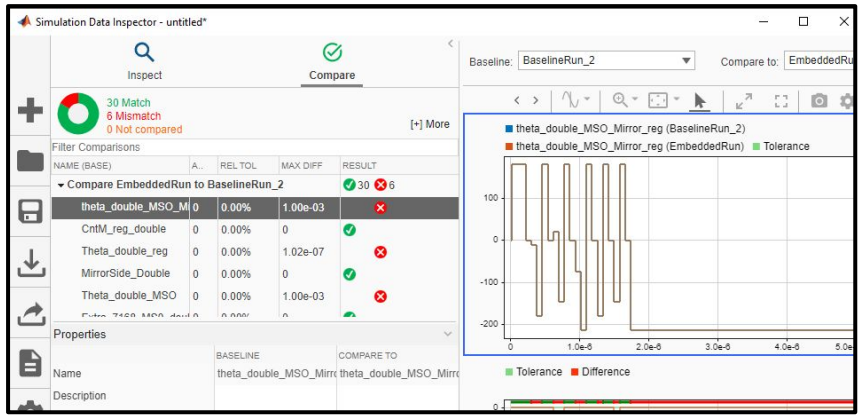


(7) Let App propose new fixed point data types in the design, then change the design with new types, fix some minor type mismatch errors, then Simulate, compare the differences, analyze,...









FPGA Area and Speed Optimization with MBD – Fixed-Point Tool III

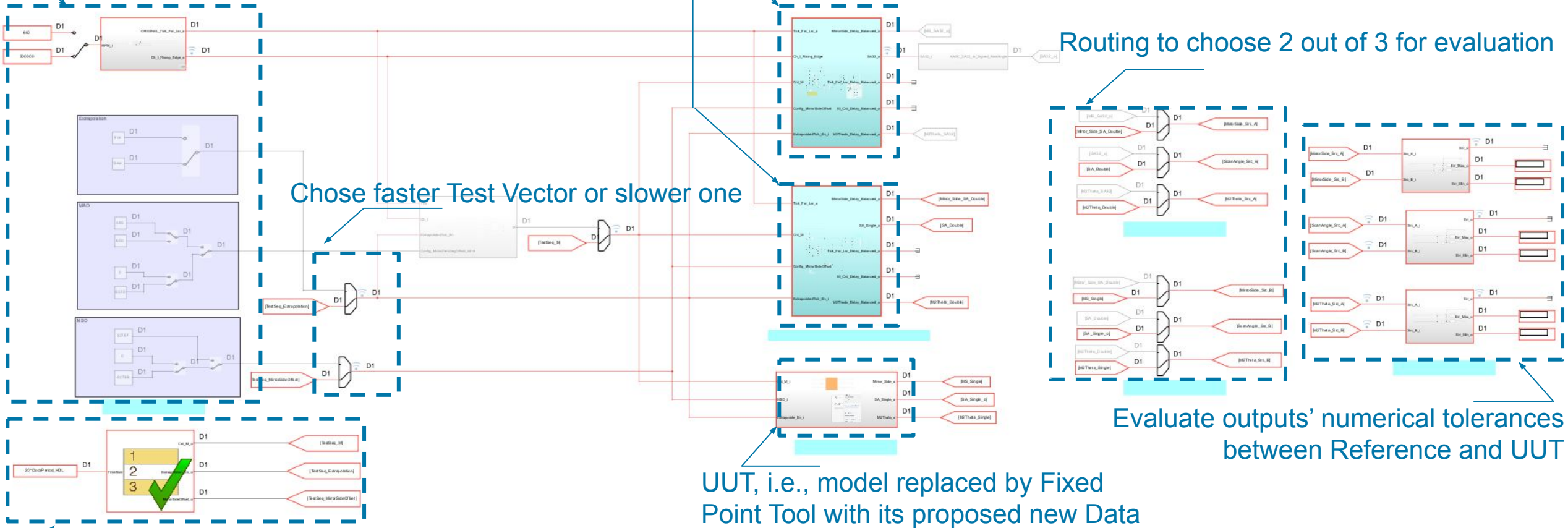
(8) Iteratively develop testbench to cover full range inputs and to evaluate performance toleration

Original model in floating point data type “double”, i.e., highest numerical accuracy

Realistic testbench. Simulation time much longer.

Original model to be improved

Routing to choose 2 out of 3 for evaluation



Chose faster Test Vector or slower one

Evaluate outputs’ numerical tolerances between Reference and UUT

UUT, i.e., model replaced by Fixed Point Tool with its proposed new Data Type leading to improved Area

Deduced Test Sequence covering full input range. Simulation time much shorter.

FPGA Area and Speed Optimization with MBD – Fixed-Point Tool IV

Compare Utilization before and after "**Fixed-Point Tool**" workflow:

HDL Workflow Advisor - tbSA32/SA32_Gen_NEW

File Edit Run Help

Find:

- HDL Workflow Advisor
  - 1. Set Target
    - ^1.1. Set Target Device and Synthesis Tool
    - 1.2. Set Target Frequency
  - 2. Prepare Model For HDL Code Generation
    - 2.1. Check Model Settings
  - 3. HDL Code Generation
    - 3.1. Set HDL Options
    - ^3.2. Generate RTL Code and Testbench
  - 4. FPGA Synthesis and Analysis
    - 4.1. Create Project
    - 4.2. Perform Synthesis and P/R
      - 4.2.1. Run Synthesis
      - 4.2.2. Run Implementation
    - 4.3. Annotate Model with Synthesis Result

4.2.2. Run Implementation

Analysis

Run place and route for specified FPGA device

Input Parameters

☐ Skip this task

☐ Ignore place and route errors

Run This Task

Result: Passed

Passed Implementation.

Parsed resource report file: SA32_Gen_NEW_utilization_placed.rpt.

Resource summary

Resource	Usage	Available	Utilization (%)
CLB LUTs	1633	44000	3.71
CLB Registers	1623	88000	1.84
DSPs	12	400	3.00
Block RAM Tile	0	100	0.00
URAM	0	0	

Parsed timing report file: timing_post_route.rpt.

Timing summary

	Value
Requirement	4 ns (250 MHz)
Data Path Delay	3.54 ns
Slack	0.403 ns
Clock Frequency	278.01 MHz

```
Generated Post Route Timing Report timing_post_route.rpt.
Task "Run Implementation" successful.
Generated logfile: E:\TU\__pjWORKSPACE\pjAASC\code\SA32_Ref\240126
```

LUTs 54% reduced

HDL Workflow Advisor - tb_SA_Single_Gen/SA_Single_Gen

File Edit Run Help

Find:

- HDL Workflow Advisor
  - 1. Set Target
    - ^1.1. Set Target Device and Synthesis Tool
    - 1.2. Set Target Frequency
  - 2. Prepare Model For HDL Code Generation
    - 2.1. Check Model Settings
  - 3. HDL Code Generation
    - 3.1. Set HDL Options
    - ^3.2. Generate RTL Code and Testbench
  - 4. FPGA Synthesis and Analysis
    - 4.1. Create Project
    - 4.2. Perform Synthesis and P/R
      - 4.2.1. Run Synthesis
      - 4.2.2. Run Implementation
    - 4.3. Annotate Model with Synthesis Result

4.2.2. Run Implementation

Analysis

Run place and route for specified FPGA device

Input Parameters

☐ Skip this task

☐ Ignore place and route errors

Run This Task

Result: Passed

Passed Implementation.

Parsed resource report file: SA_Single_Gen_utilization_placed.rp

Resource summary

Resource	Usage	Available	Utilization (%)
CLB LUTs	735	44000	1.67
CLB Registers	636	88000	0.72
DSPs	9	400	2.25
Block RAM Tile	0	100	0.00
URAM	0	0	

Parsed timing report file: timing_post_route.rpt.

Timing summary

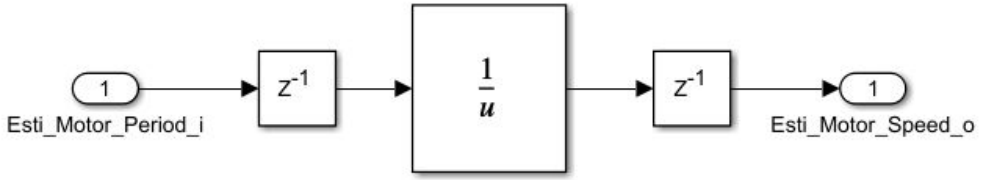
	Value
Requirement	4 ns (250 MHz)
Data Path Delay	3.863 ns
Slack	0.118 ns
Clock Frequency	257.60 MHz

```
Generated Post Route Timing Report timing_post_route.rpt.
Task "Run Implementation" successful.
Generated logfile: E:\TU\__pjWORKSPACE\__R23b\_AASC\code\
****** Vivado v2022.1 (64-bit)
```

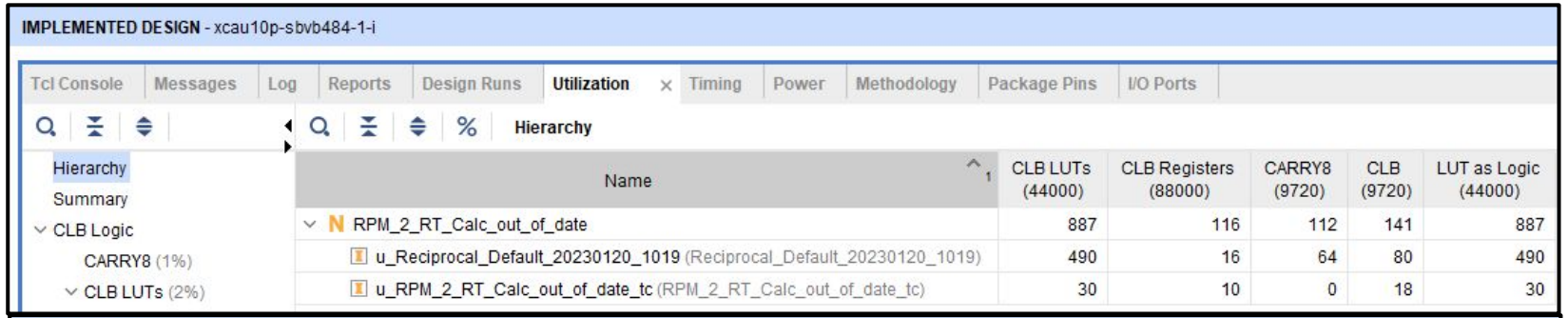
FPGA Area and Speed Optimization with MBD - Example

Example: Motor Speed Calculation requires the arithmetic operation **"Reciprocal"** which consumes lots of FPGA LUT resources, also challenging to meet timing constraints.

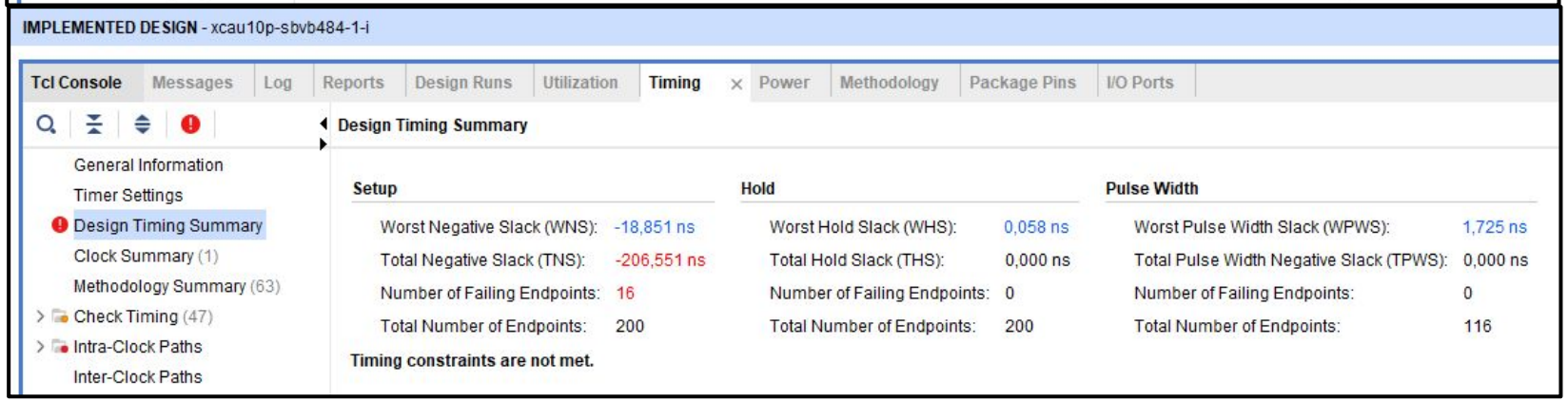
$$\widehat{RPM} = \frac{60}{\widehat{MotorPeriod}} = \frac{60}{7168 \times \widehat{TickPeriod}} = \frac{60/7168}{\widehat{TickPeriod}}$$



- Consumes many CLB LUTs
- Timing failure



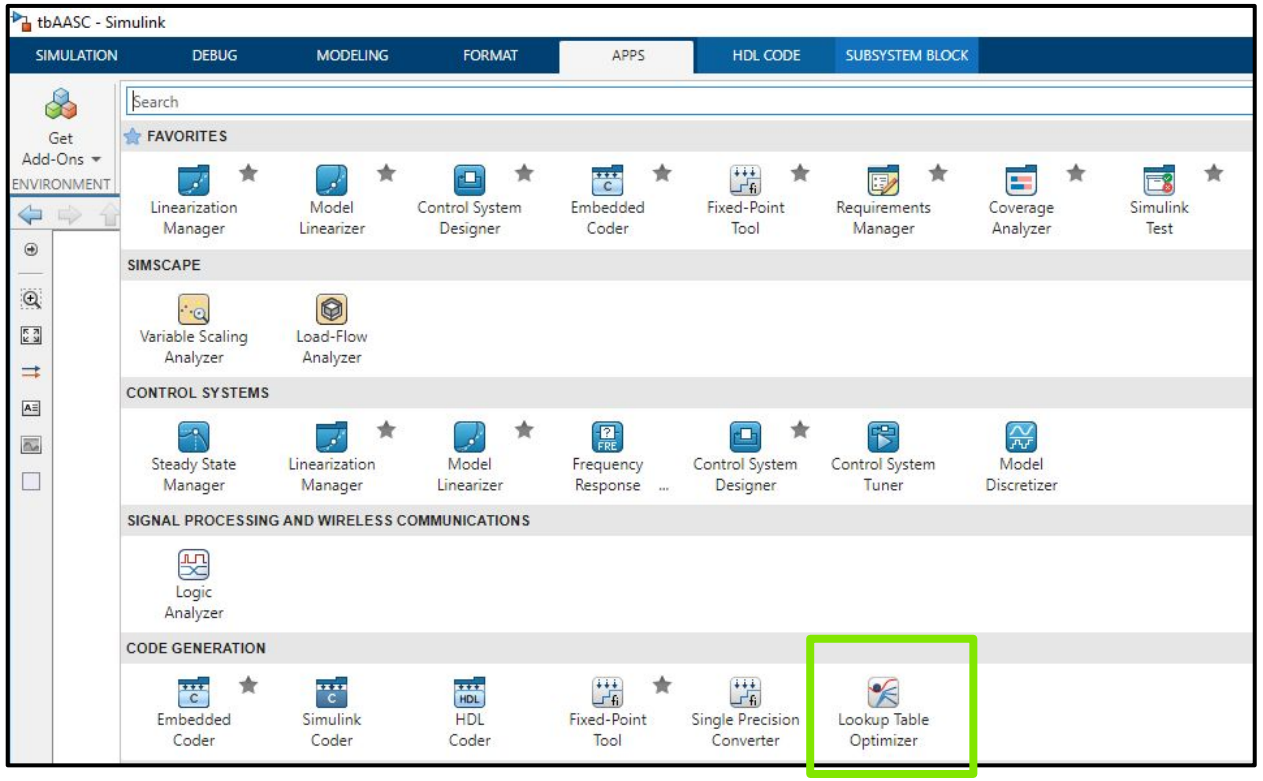
Name	CLB LUTs (44000)	CLB Registers (88000)	CARRY8 (9720)	CLB (9720)	LUT as Logic (44000)
RPM_2_RT_Calc_out_of_date	887	116	112	141	887
u_Reciprocal_Default_20230120_1019 (Reciprocal_Default_20230120_1019)	490	16	64	80	490
u_RPM_2_RT_Calc_out_of_date_tc (RPM_2_RT_Calc_out_of_date_tc)	30	10	0	18	30



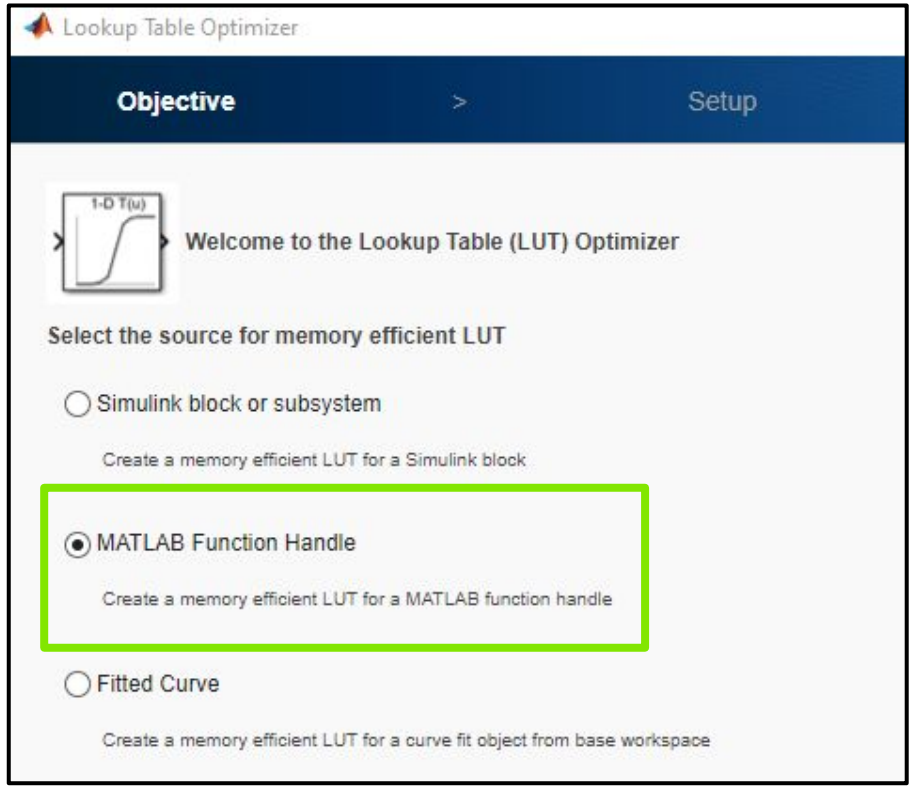
FPGA Area and Speed Optimization with MBD – LUT Optimization I

Use the App “Lookup Table Optimizer” to re-target the design from FPGA LUT into RAM Blocks, reducing the use of LUT resources and improving timing performance.

(1) Open the lookup table optimizer app

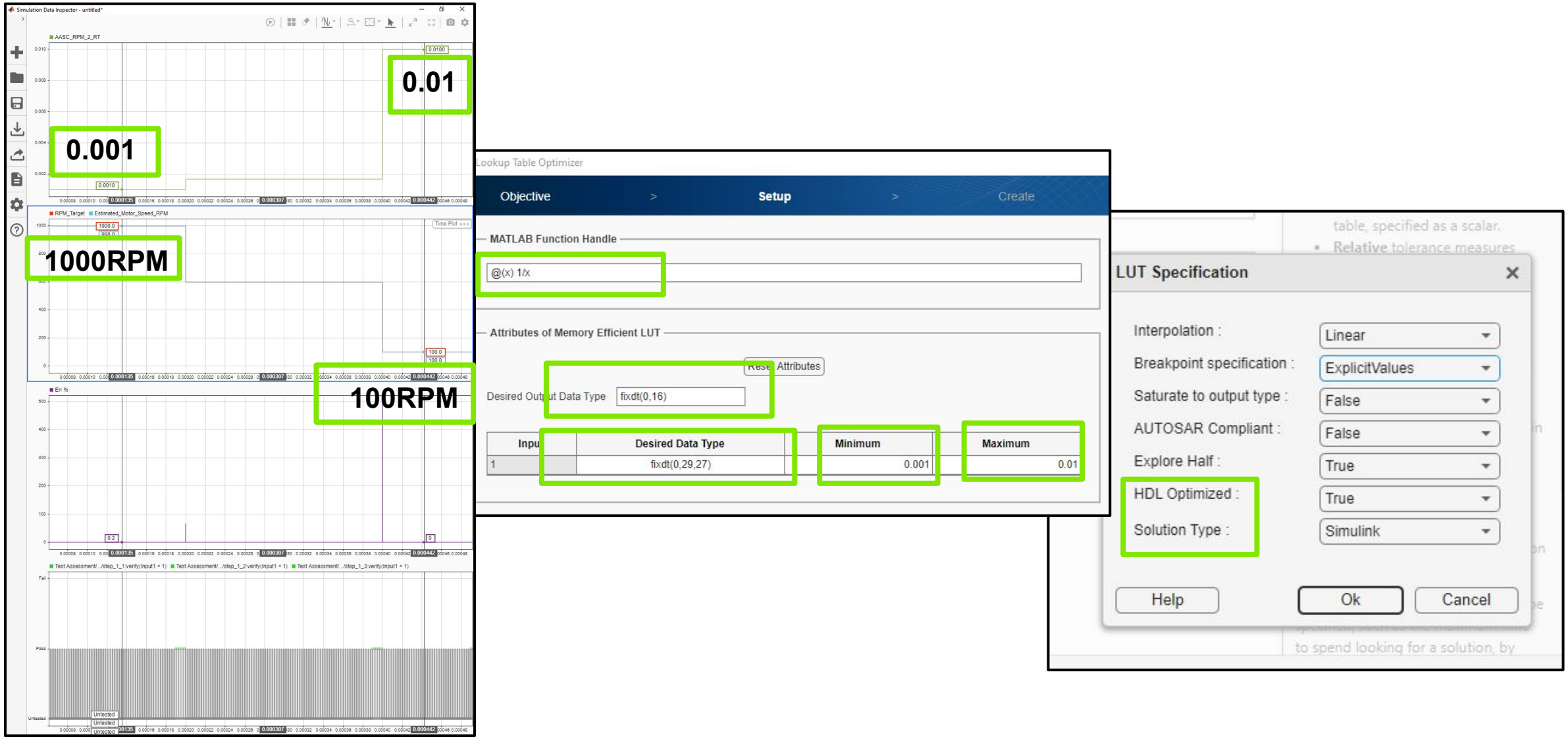


(2) Choose MATLAB function handle



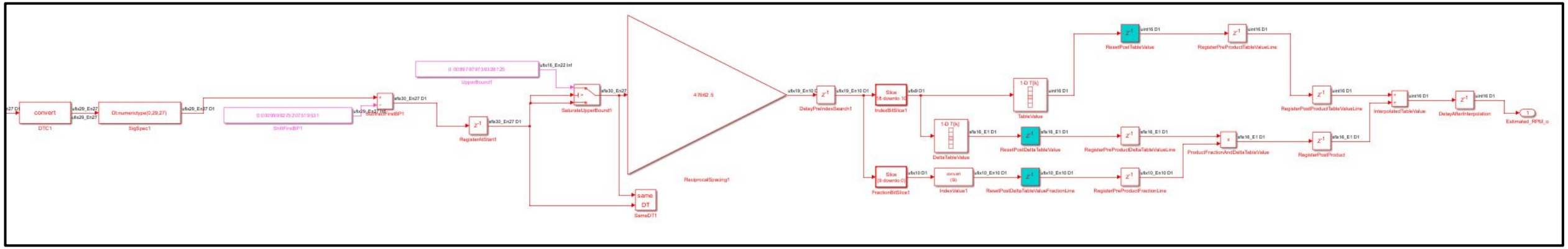
FPGA Area and Speed Optimization with MBD – LUT Optimization II

Specify function, enter the targeted input data type and input range, set to HDL Optimized



FPGA Area and Speed Optimization with MBD – LUT Optimization III

Tool generates new model targeting HDL efficiently



- CLB LUTs reduced from 887 to 167
- Achieved Timing closure!

IMPLEMENTED DESIGN - xcau10p-sbvb484-1-i

Tcl Console | Messages | Log | Reports | Design Runs | **Utilization** | Timing | Power | Methodology | DRC | Package Pins | I/O Ports

Hierarchy
- Hierarchy
- Summary
- CLB Logic

Name	CLB LUTs (44000)	CLB Registers (88000)	CARRY8 (9720)	F7 Muxes (38880)	F8 Muxes (19440)	CLB (9720)	LUT as Logic (44000)	Block RAM Tile (100)	DSPs (400)
RPM_2_RT_Calc	167	128	14	19	4	43	167	0.5	5

IMPLEMENTED DESIGN - xcau10p-sbvb484-1-i

Tcl Console | Messages | Log | Reports | Design Runs | Utilization | **Timing** | Power | Methodology | DRC | Package Pins | I/O Ports

- General Information
- Timer Settings
- Design Timing Summary
- Clock Summary (1)
- Methodology Summary (52)
- Check Timing (47)
- Intra-Clock Paths

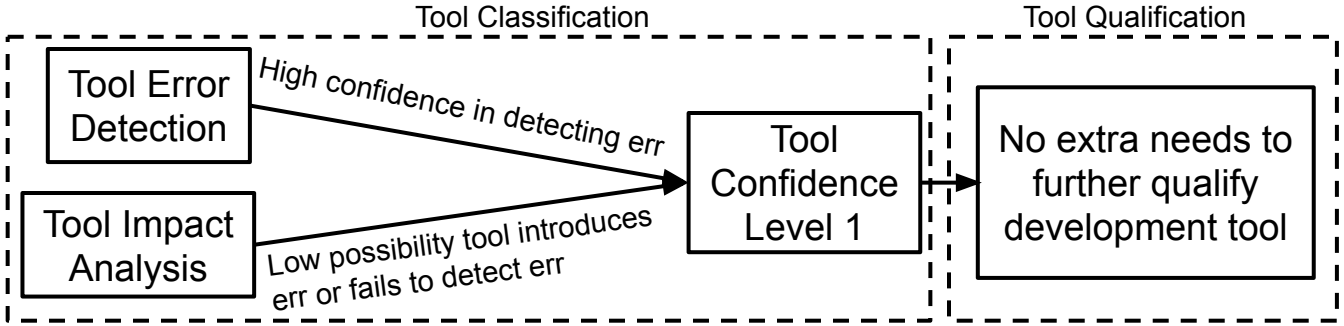
Design Timing Summary

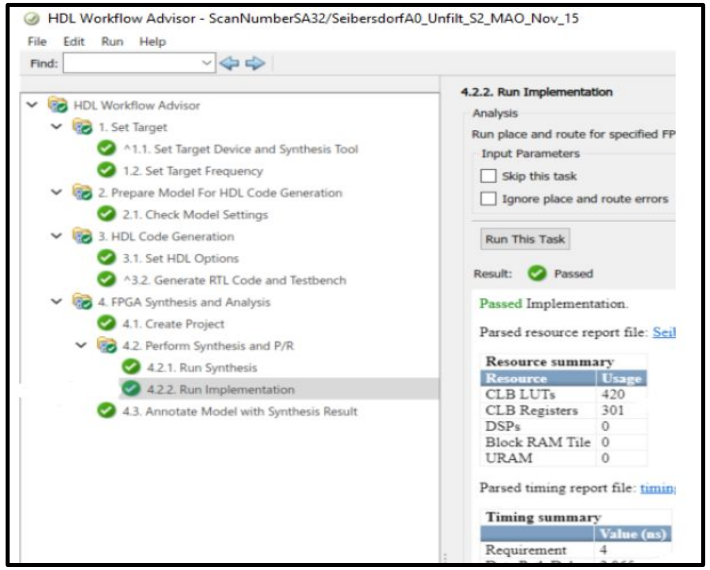
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0,719 ns	Worst Hold Slack (WHS):	0,056 ns	Worst Pulse Width Slack (WPWS):	1,457 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	316	Total Number of Endpoints:	316	Total Number of Endpoints:	133

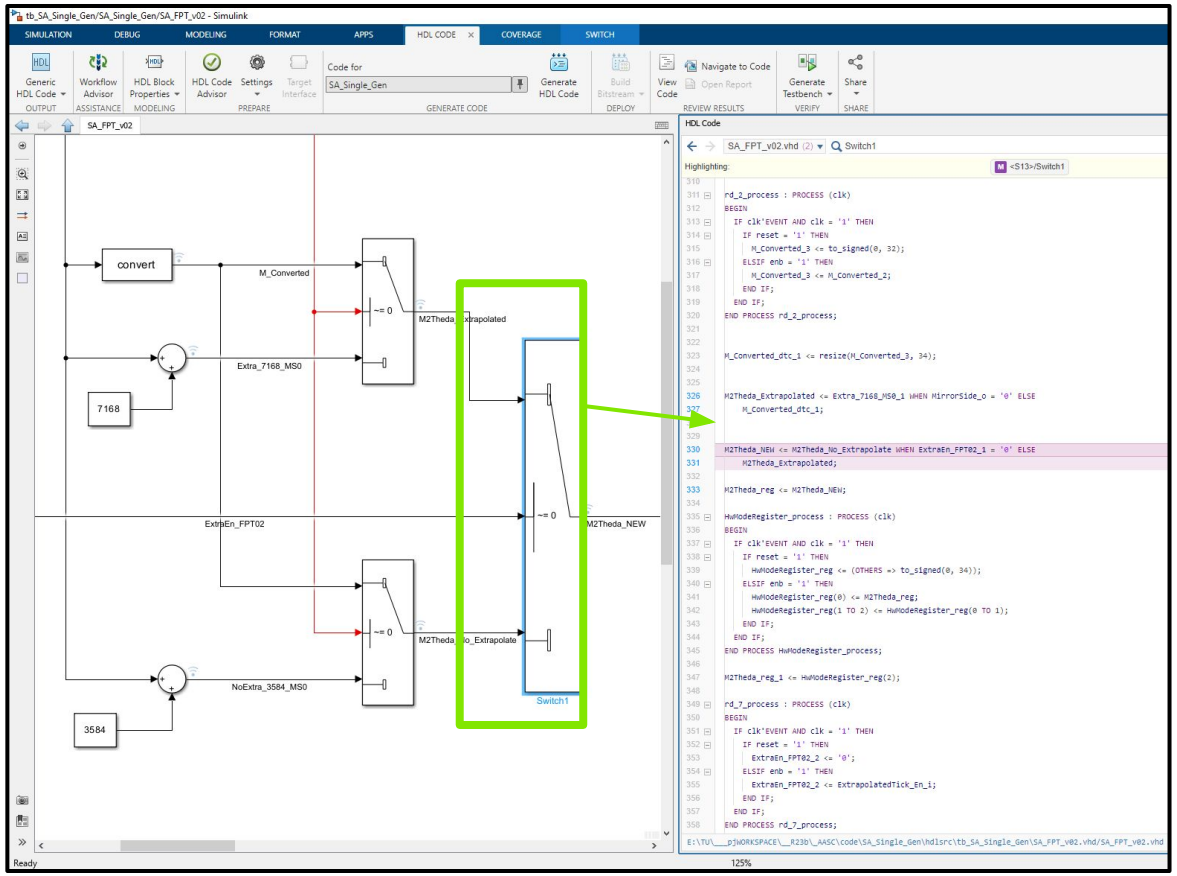
All user specified timing constraints are met.

FPGA Area and Speed Optimization with MBD

- **Simulink Test** provides non intrusive test harnesses for model tests and efficient integrated workflows.
- **Simulink Coverage** provides Model Coverage
- **HDL Verifier** proves the equivalence between the model and generated code.
- **HDL Coder** is also certified by TÜV SÜD to be suitable for use in developing ISO 26262 products for all ASILs when taking V&V measures

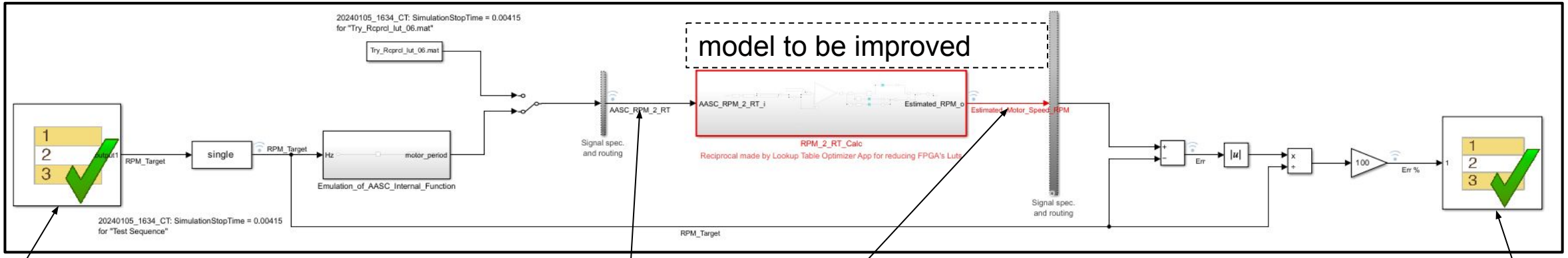






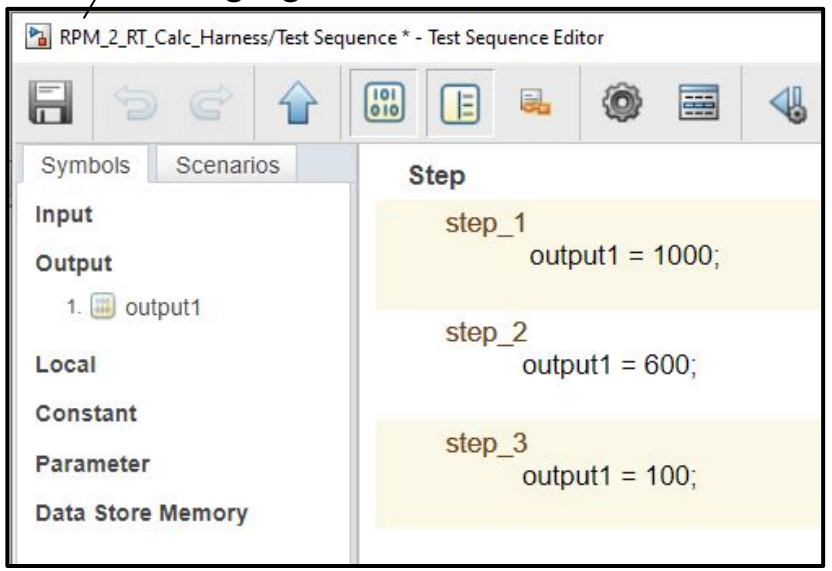
FPGA Verification with MBD – Model Verification

Use **Simulink Test** to create a test harness, to evaluate input / output data range, data types, testbench to define expected reference behaviors.

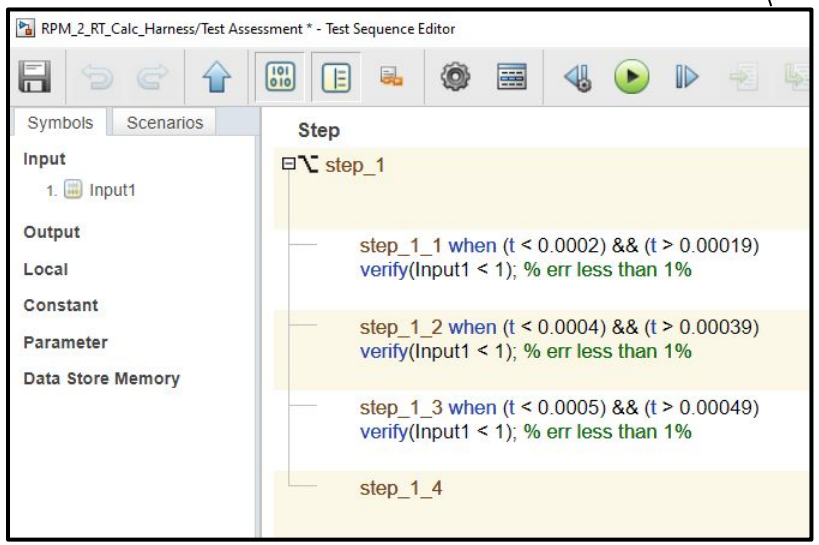


Test Sequence specifying test cases, ranging between 100RPM and 1000RPM

Test Assessment to compare results and judge correctness for the test cases ranging between 100RPM and 1000RPM



Investigate input / output data types, minimal, maximal ranges



FPGA Verification with MBD – Measuring Test Coverage

Achieve 100% Coverage with the same Testbench against the new, efficient model (MiL).

Test Harness: RPM_2_RT_Calc_Harness/RPM_2_RT_Calc * - Simulink

SIMULATION | DEBUG | MODELING | FORMAT | APPS | COVERAGE | HARNESS

Coverage ON | Cumulative Collection | Settings | Stop Time | Simulatio | Normal | Fast Restart | Step Back | Analyze Coverage | Step Forward | Stop | Coverage Highlighti... | Coverage Details | Results Explorer | Generate Report

STATUS | PREPARE | ANALYZE | REVIEW RESULTS

RPM_2_RT_Calc

Coverage Details

Details

1. SubSystem block "RPM_2_RT_Calc"

Child Systems: FractionBitSlice1, IndexBitSlice1

Metric	Coverage (this object)	Coverage (inc. descendants)
Execution	NA	100% (30/30) objective outcomes

Switch block "SaturateUpperBound1"

Justify or Exclude

Parent: RPM_2_RT_Calc_Harness/RPM_2_RT_Calc

Metric	Coverage
Execution	100% (1/1) objective outcomes

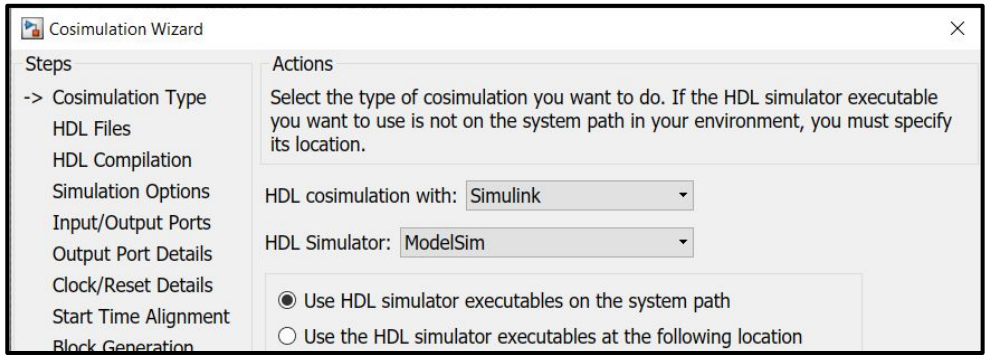
Execution analyzed

Block executed	100%
	875001/875001

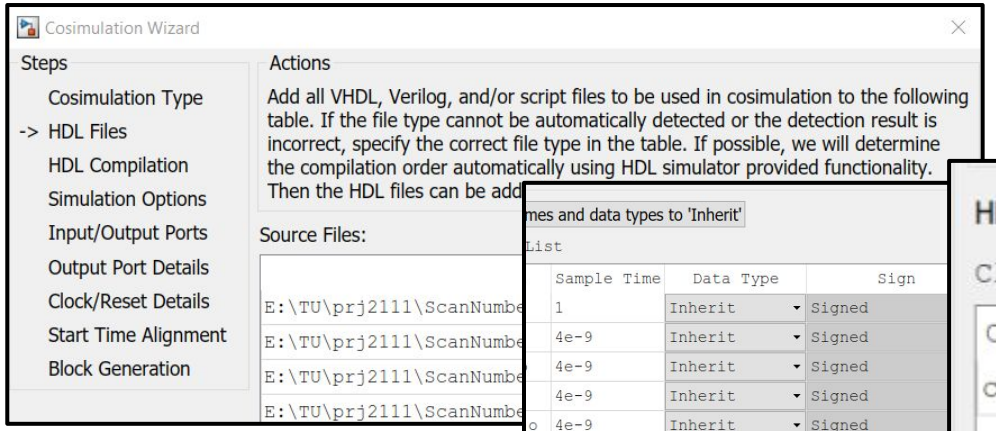
FPGA Verification with MBD – Back-to-Back Verification

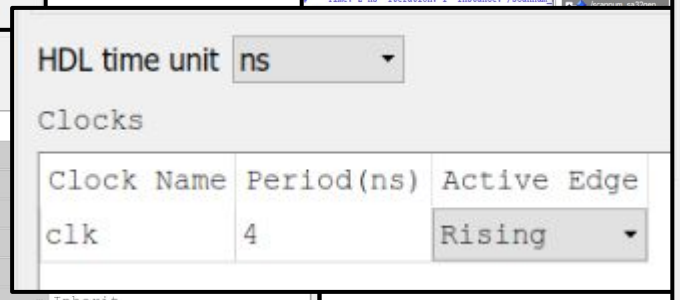
(7) **HDL Verifier** validates that the model and generated HDL code have the same behavior as in HDL simulator (co-simulation) and on FPGA Hardware (FPGA-in-the-Loop). It can also incorporate legacy hand code into Simulink models for simulation.

- Open cosimWizard

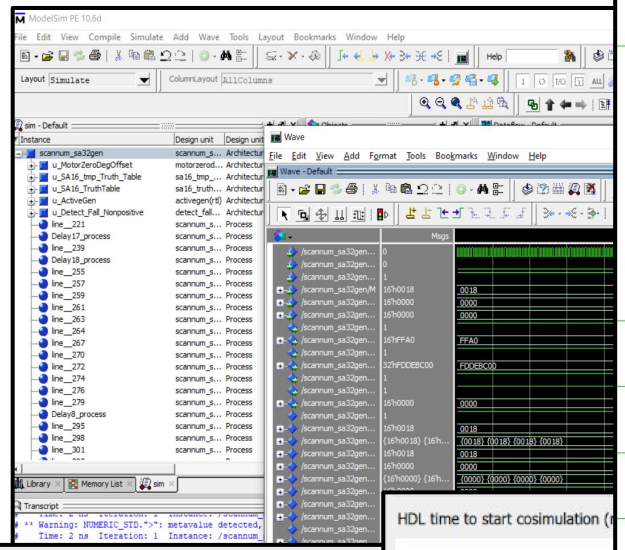


- Specify source code, sample time, data types, reset timing...

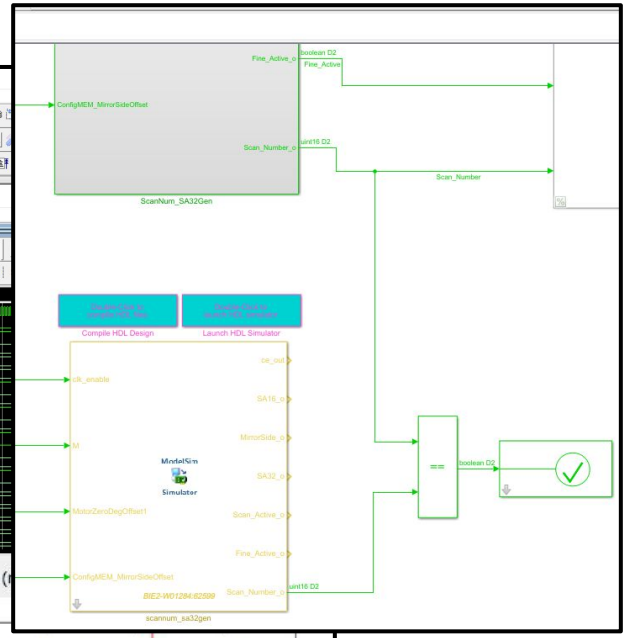


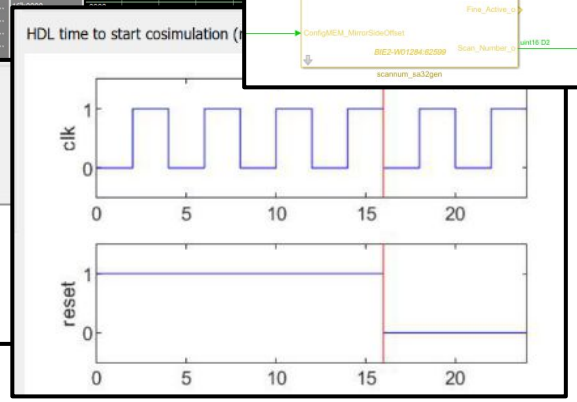


- HDL simulator



- Compare model output against HDL simulator output





Key Takeaways

Aspects	**Project: Scala 2** Bottom-up, Manual Approach	**Project: Scala 3** MBD Approach
Executable specification	Requirements and design implementations are separate files. High efforts in understanding the design and maintaining consistency and traceability.	Model serves as both simulatable requirements and design implementation. Easy to understand, high confidence level in consistency throughout V-Cycle. Automatic traceability.
System-level simulation	Surprises can occur late in the design process. Can be challenging to fix. Different teams use diverse tools. Domain knowledge not as easy to share.	Cross-domain teams collaborate in the same Simulink environment to investigate ideas, possibilities, limits, and identify risk early.
What-if analysis	Limited options explored as HW implementation. Not easy, high cost to perform generic thorough investigation.	Many design options, scenarios and trade study can be simulated and evaluated early on HW
Automatic production code generation	Hand coding is prone to manual errors, experience and time consuming. Requirement fulfillment can be difficult to understand.	Systematic, automatic approach. Higher efficiency.
Knowledge management	High effort, difficulties in communication, and in convincing different teams and stakeholders.	Teams / stakeholders communication easier

Gratitude to MathWorks Professionals and Consultants

“MathWorks Experts collaborated with Valeo throughout project development, offering expertise in Model-Based Design, guidance, issue investigation, technical evaluation and design optimization - from the early design phase to project completion.”

Thank you for supporting Valeo’s project success !!

MathWorks
AUTOMOTIVE CONFERENCE 2024
Europe

Thank you

Valeo
SMART TECHNOLOGY FOR SMARTER CARS