

TI 技術研討會
嵌入式處理器解決方案

Implementing Video Tracking Systems on DM6437 EVM using MATLAB and Simulink

Esther Wu
Application Engineer
Terasoft

Agenda

- **Motivation**

 - Rapid Design and Implementation of Embedded Signal/Video Processing Algorithms and Systems on DM6437 EVM

- **Model Based Design and Simulation**

 - Case Study: *Video Tracking & Lane Detection*

 - Floating Point Reference Model

 - Fixed-Point Conversion

 - Embedded MATLAB – Integrating MATLAB Code

- **Implementation on DM6437 EVM**

 - Rapid Prototyping with Fixed-Point Code Generation

 - Implementation Decisions

 - Memory Mapping

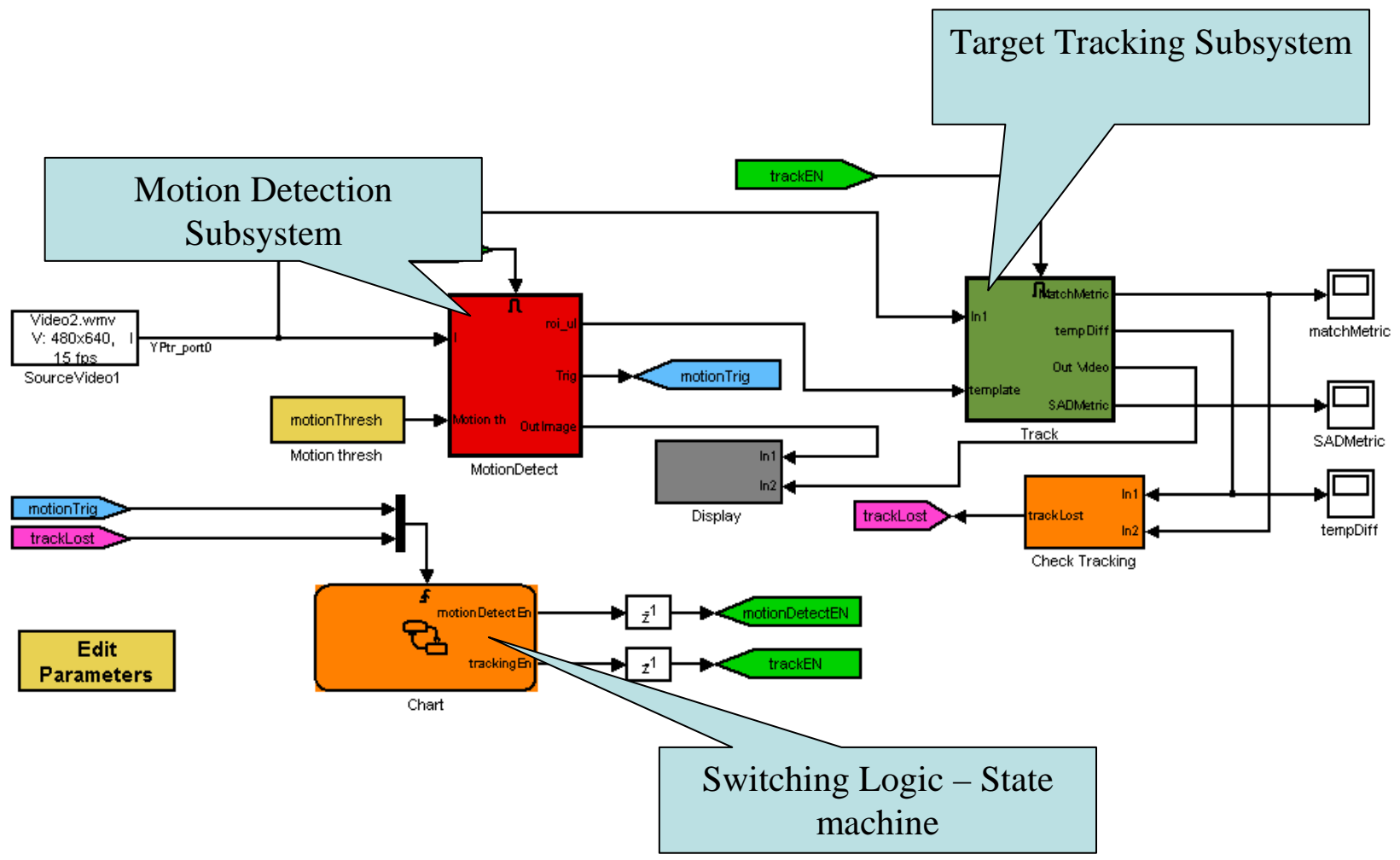
 - Code profiling on DSP to identify bottlenecks

 - Code and memory optimizations

 - Peripheral Support

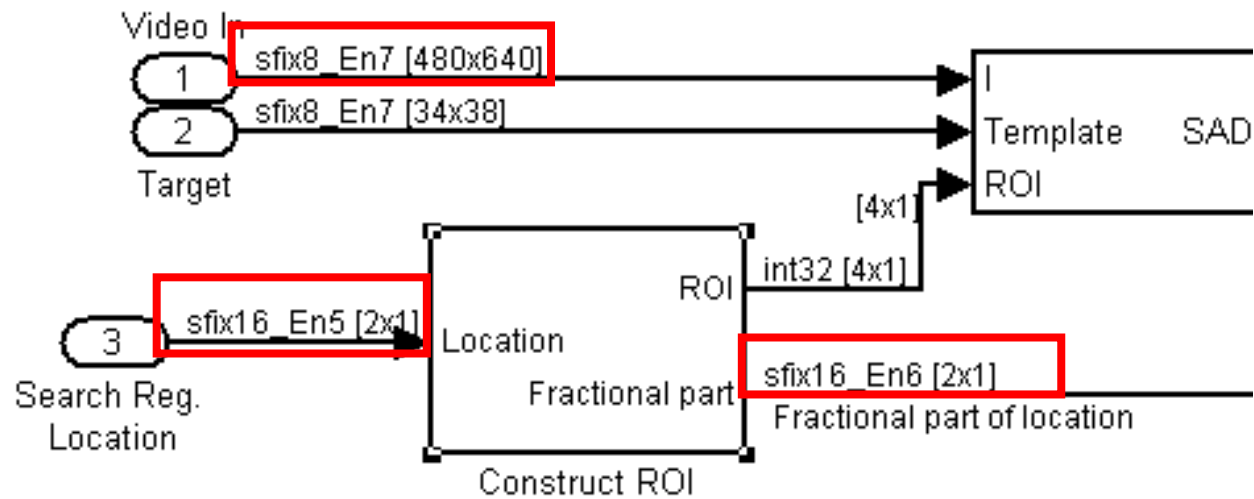
 - Integrating custom code and 64x+ ASM Library (IMGLIB)

Video Tracking Model



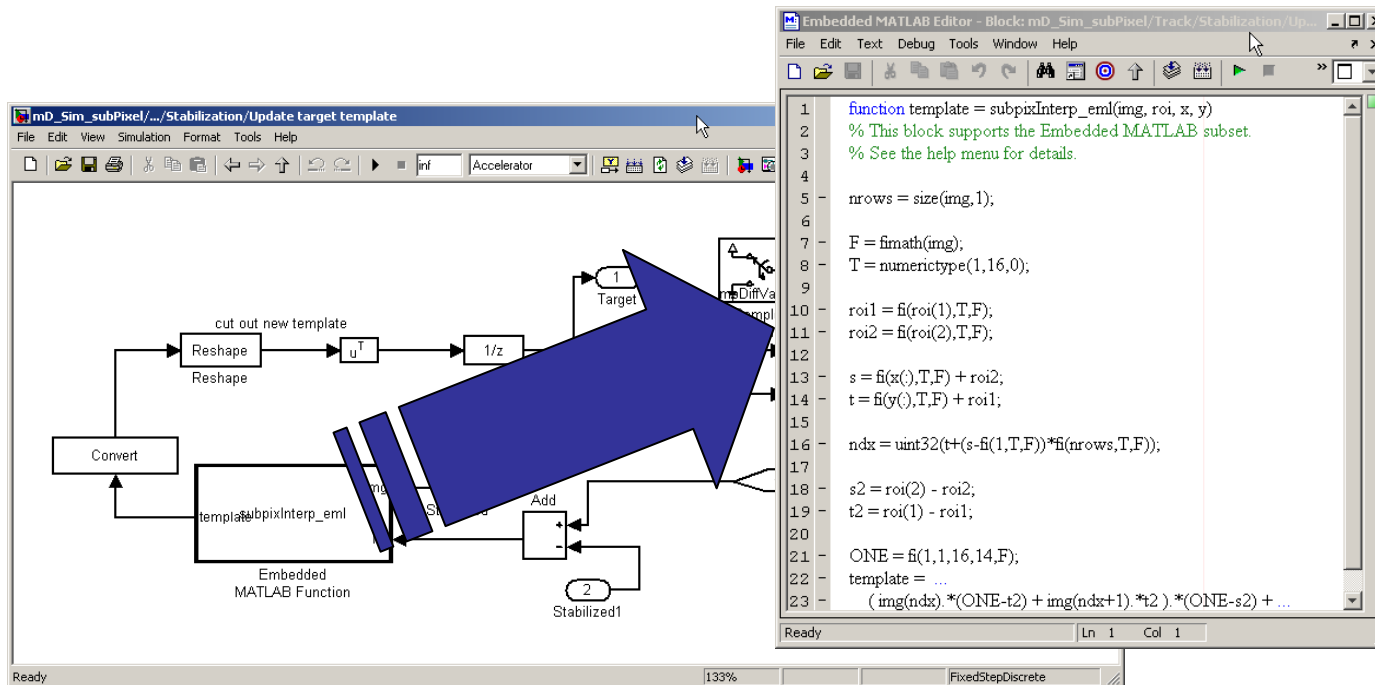
Fixed-Point Conversion

- Modeling Fixed-point data types for bit-true simulation
- Manual or Autoscaling to determine the optimum fractional settings (scaling) for DSP word lengths



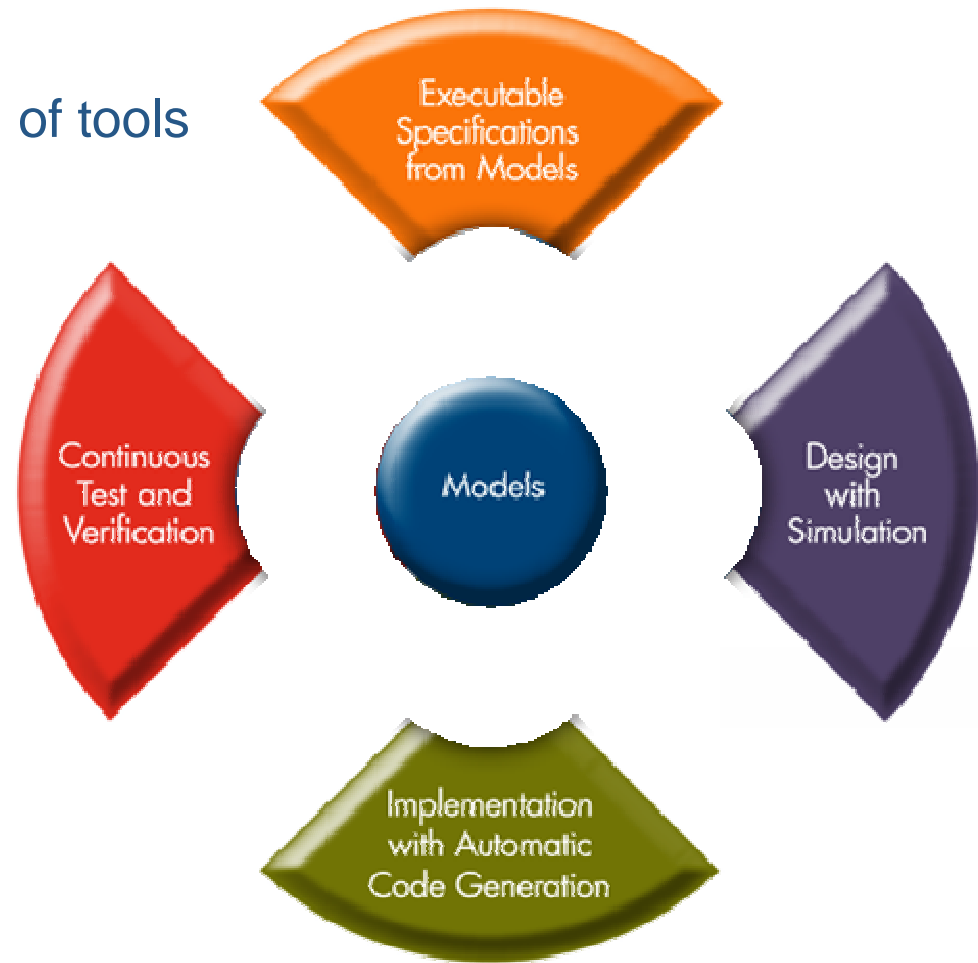
Using Embedded MATLAB

- Integrate M-code in Simulink for simulation and C-code generation to implement on TI DSP
- Sub-pixel Estimation for more robust tracking performance



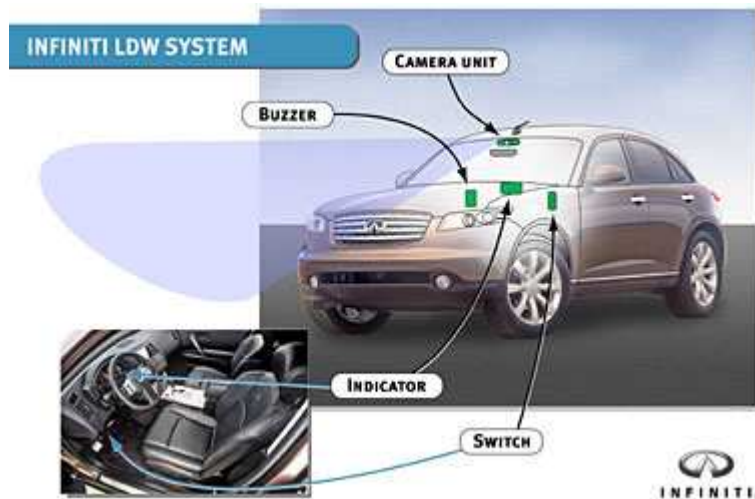
Model Based Design Concept

- Focus on the complete algorithm-design lifecycle
 - MBD demands tight integration of tools
 - Maintains model abstraction (defers implementation)
 - Simplifies:
 - Understanding
 - Verification
 - Maintenance
 - Enhancement
 - Re-targetability

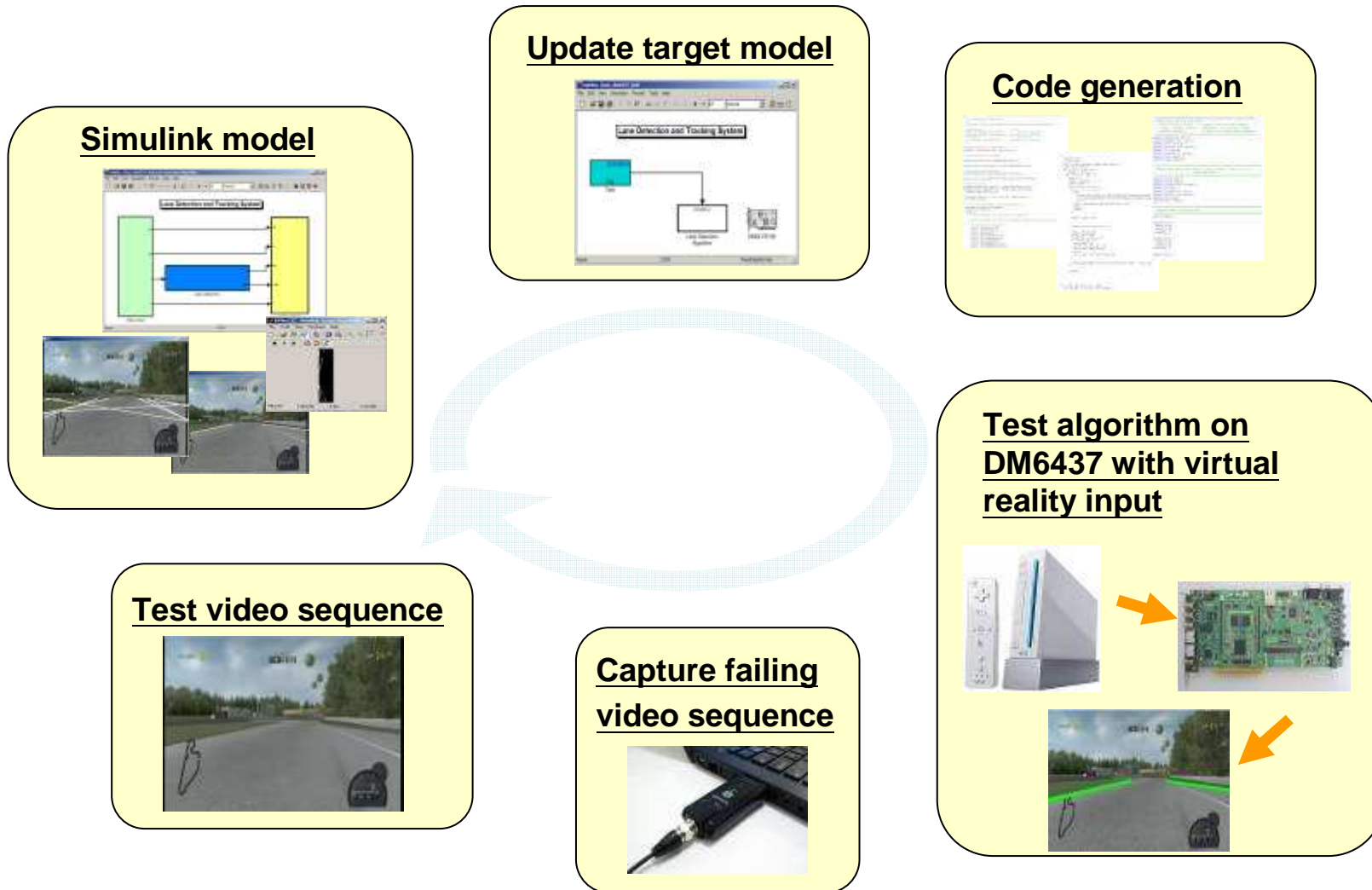


Automotive Design Example: Lane Departure Warning

Results from Simulink Model of
Lane Departure Warning System

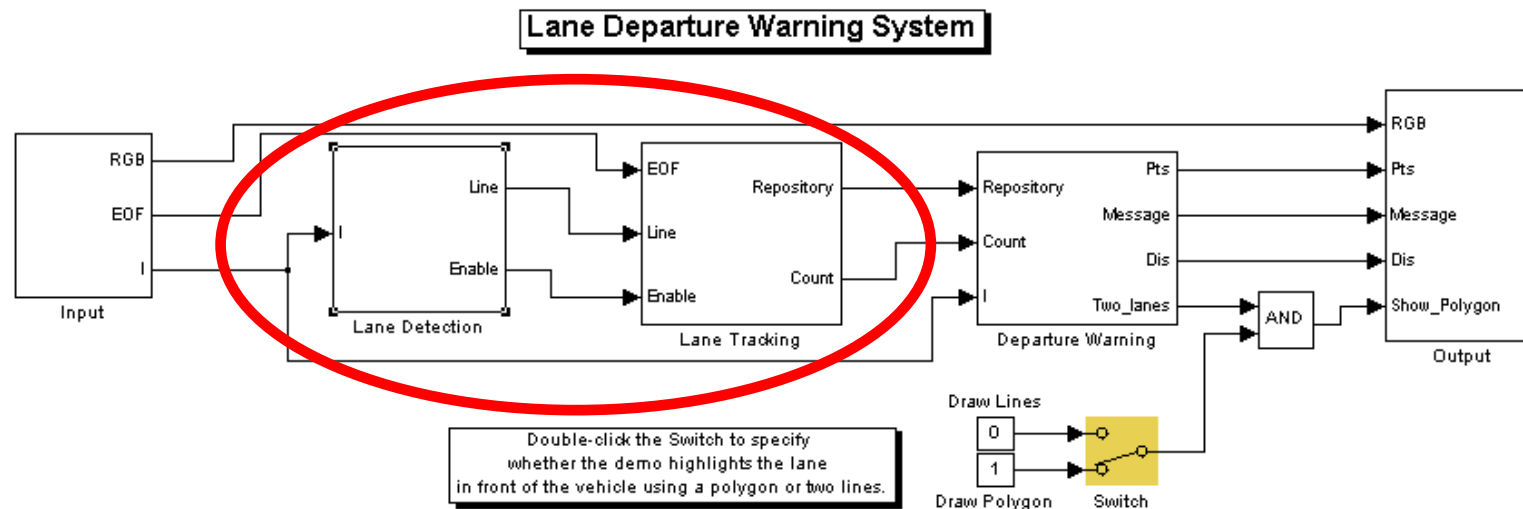


Why did We Choose Model Based Design? Iterative Algorithm Design Workflow



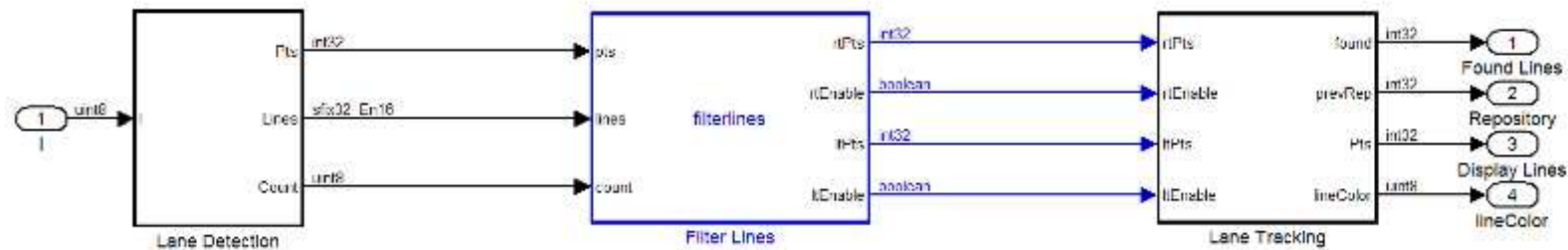
Executable Specification

System Model: Implement lane detection and tracking algorithms as part of lane departure warning system



Behavioral Model

Start by developing a golden reference specification of the algorithm

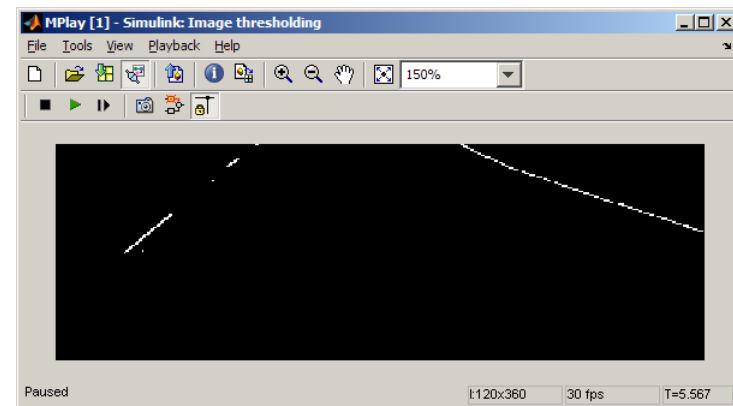


High-level blocks: Hough Transform, Autothreshold

- Video and Image Processing Blockset

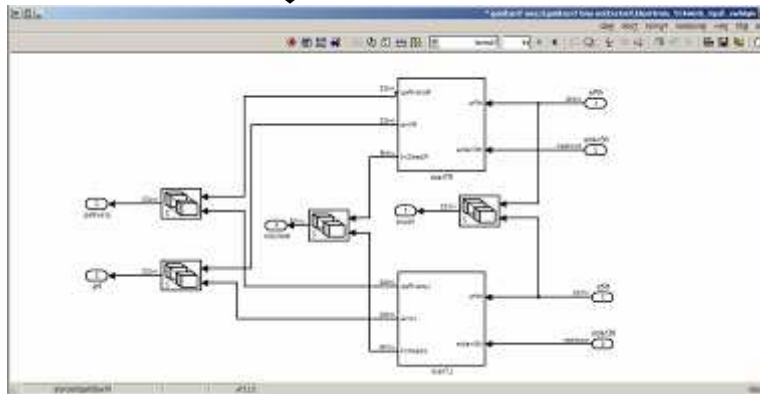
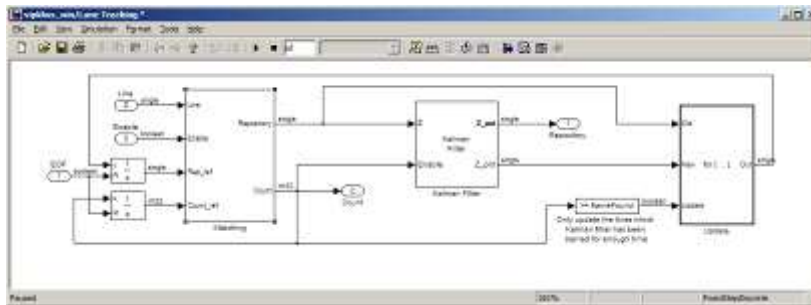
Lower-level blocks: Abs, 2-D Filter, For-Iterator

- Simulink, Signal Processing Blockset



Design with Simulation

- Implement the design using verified iterations
- Make decisions appropriate to deployment
 - Patch processing (*Efficient Memory Use*)
 - Fixed-point operation (*float-to-fixed conversion*)



Behavioral

Floating-point model
Total Lane Tracking

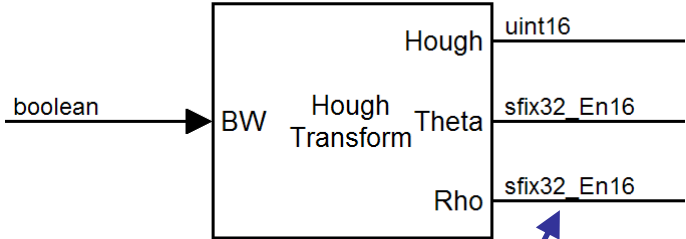
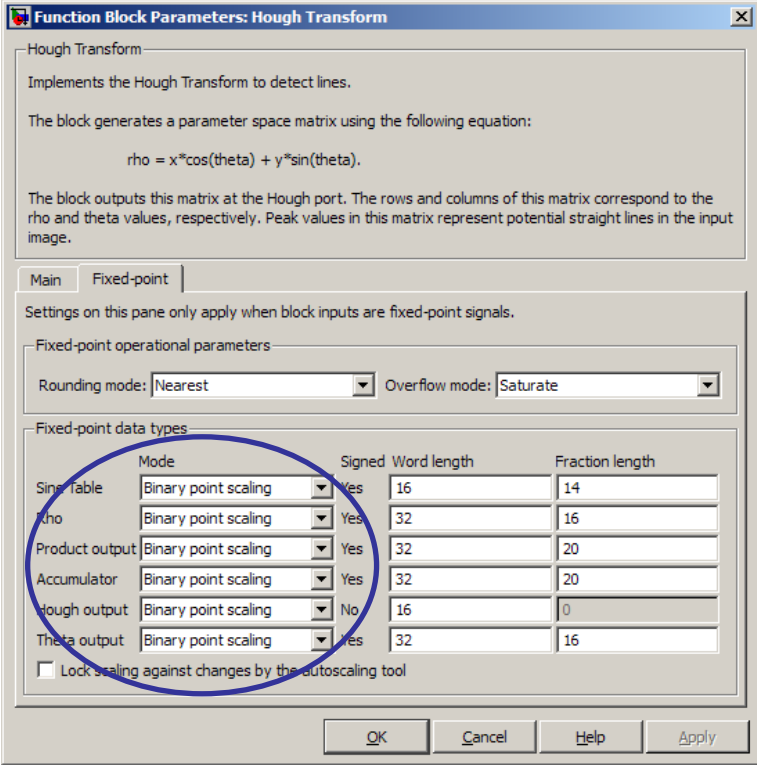


Implementation

Fixed-point model
Left/Right Lane Tracking

Migration to Fixed-point

- Fixed-point data types – full manual control



“sfix32_En16”
signed fixed-point number
32 bit word, 16 bit fraction

Migration to Fixed-point

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Sine Table	Binary point scaling	Yes	16	14
Rho	Binary point scaling	Yes	32	16
Product output	Binary point scaling	Yes	32	20
Accumulator	Binary point scaling	Yes	32	20
Hough output	Binary point scaling	No	16	0
Theta output	Binary point scaling	Yes	32	16

Lock scaling against changes by the autoscaling tool

Fixed-point operational parameters

Rounding mode: Nearest Overflow mode: Saturate

Fixed-point data types

	Mode	Signed	Word length	Fraction length
Sine Table	Binary point scaling	Yes	16	14
Rho	Binary point scaling	Yes	32	16
Product output	Binary point scaling	Yes	32	20
Accumulator	Binary point scaling	Yes	32	20
Hough output	Binary point scaling	No	16	0
Theta output	Binary point scaling	Yes	32	16

Lock scaling against changes by the autoscaling tool

boolean → BW Hough Transform

Hough → uint16

Theta → sfix32_En16

Rho → sfix32_En16

“sfix32_En16”
signed fixed-point number
32 bit word, 16 bit fraction

Migration to Fixed-point

Autoscaling Tool: Advisor to help set fixed-point scale factors

Fixed-Point Tool

File Simulation View Tools Results Autoscaling Help

Model Hierarchy

Contents of: dspanc_fixpt_win32* (mmo)

Name	Run	SimMin	SimMax	SimDT	ProposedFL	Acc
Noisy Signal/To Workspace	Reference	---	---	---	---	<input type="checkbox"/>
Manual Switch3/SwitchControl	Reference	0	0	double	---	<input type="checkbox"/>
Manual Switch2/SwitchControl	Reference	0	0	double	---	<input type="checkbox"/>
Manual Switch1/SwitchControl	Reference	1	1	double	16	<input type="checkbox"/>
LMS Filter : Product output W'u	Reference	0.04	0.2575	double	40	<input checked="" type="checkbox"/>
LMS Filter : Product output u'u	Reference	0.001267	0.8463	double	31	<input checked="" type="checkbox"/>
LMS Filter : Product output Q'u	Reference	0	11.89	double	27	<input checked="" type="checkbox"/>
LMS Filter : Product output mu'e	Reference	-0.06382	0.04816	double	32	<input checked="" type="checkbox"/>
LMS Filter : Output Signal	Reference	-1.879	1.768	double	34	<input checked="" type="checkbox"/>
LMS Filter : Error Signal	Reference	-1.96	1.899	double	---	<input type="checkbox"/>
LMS Filter : Accumulator W'u	Reference	-1.9	1.768	double	30	<input checked="" type="checkbox"/>
LMS Filter : Accumulator u'u	Reference	0	80.93	double	24	<input checked="" type="checkbox"/>
Data Type Conversion	Reference	-1.596	1.204	double	---	<input type="checkbox"/>
Acoustic Environment/Sum1/Switch	Reference	-0.04431	0.2426	double	---	<input type="checkbox"/>
Acoustic Environment/Sum2	Reference	-1.968	1.899	double	---	<input type="checkbox"/>
Acoustic Environment/Digital Filter	Reference	---	---	---	---	<input type="checkbox"/>
Acoustic Environment/Digital Filter	Reference	-0.8053	0.8368	double	31	<input checked="" type="checkbox"/>
Acoustic Environment/Digital Filter	Reference	1.74	1.74	double	---	<input type="checkbox"/>
Acoustic Environment/Digital Filter	Reference	1.75	1.75	double	30	<input checked="" type="checkbox"/>
Acoustic Environment/Conversion	Reference	-3.319	3.449	double	13	<input checked="" type="checkbox"/>
Noisy Signal/To Workspace	Active	---	---	---	---	<input type="checkbox"/>
Manual Switch3/SwitchControl	Active	0	0	boolean	---	<input type="checkbox"/>

Current System: dspanc_fixpt_win32

Results settings

Store results as: Active

Overwrite or merge results: Overwrite

Simulation settings

Logging mode: Minimums, maximums and overflows

Data type override: Use local settings

Autoscale fixed-point blocks

Autoscale using: Reference

Percent safety margin (e.g. 10 for 10%): 0

Propose fraction lengths

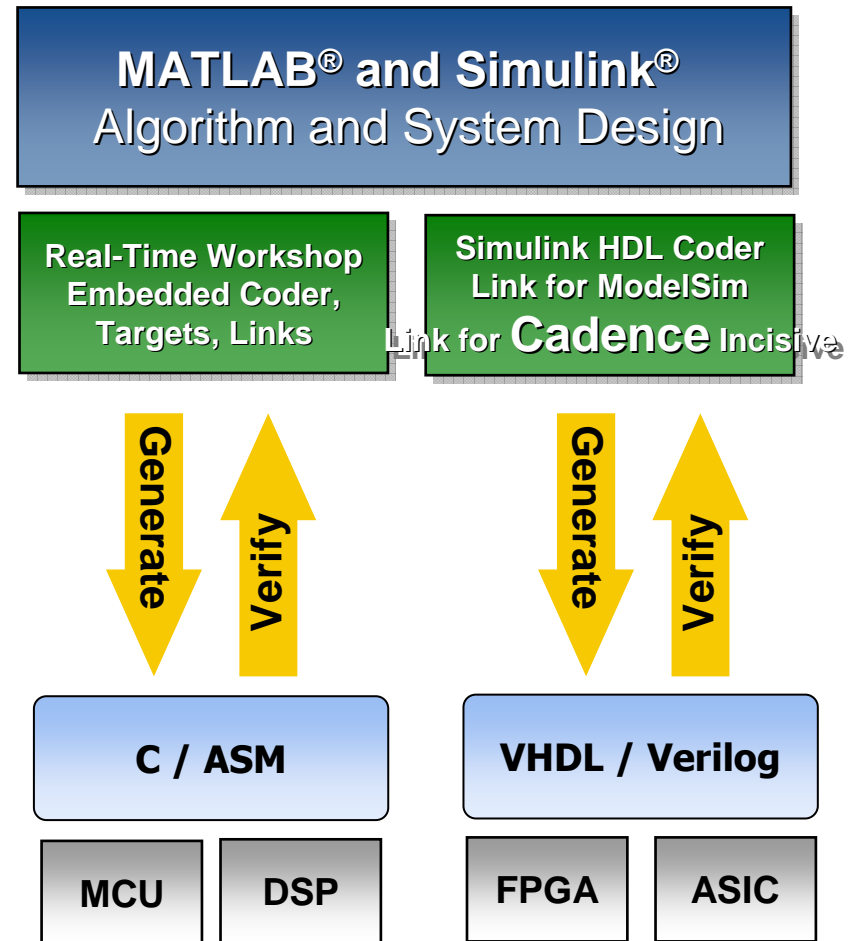
Apply accepted fraction lengths

Revert Help Apply

Code Generation

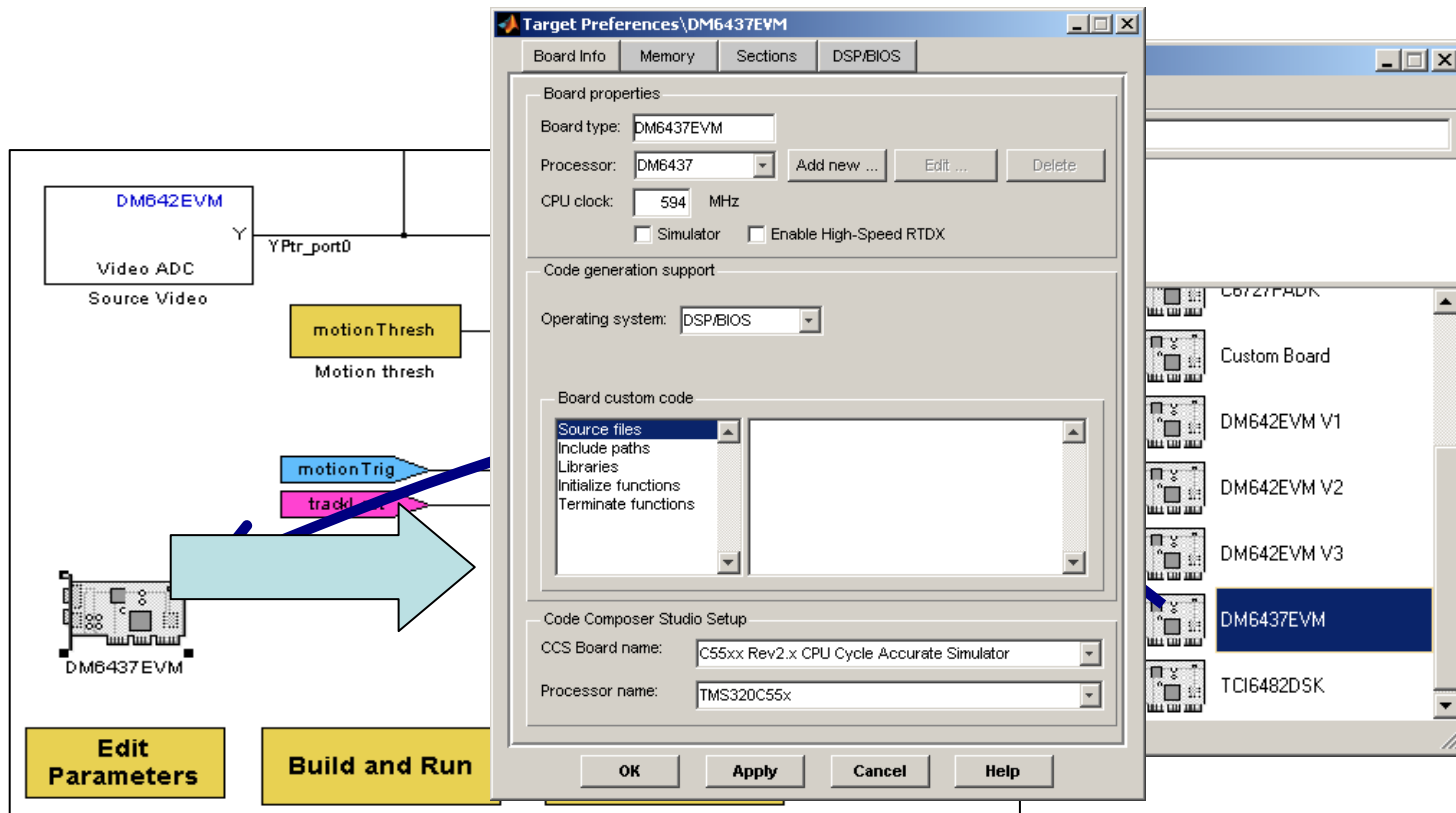
Model-Based Design supports both Software and Hardware systems

- Coders
 - Code generation from models
 - Language options
 - Code interfacing, optimization
- Links
 - Verification tool integration
 - Project generation, build, download
 - Co-simulation, SIL/PIL/HIL
- Targets
 - Processor & memory specific optimization
 - Device drivers, board support
 - Schedulers, RTOS integration



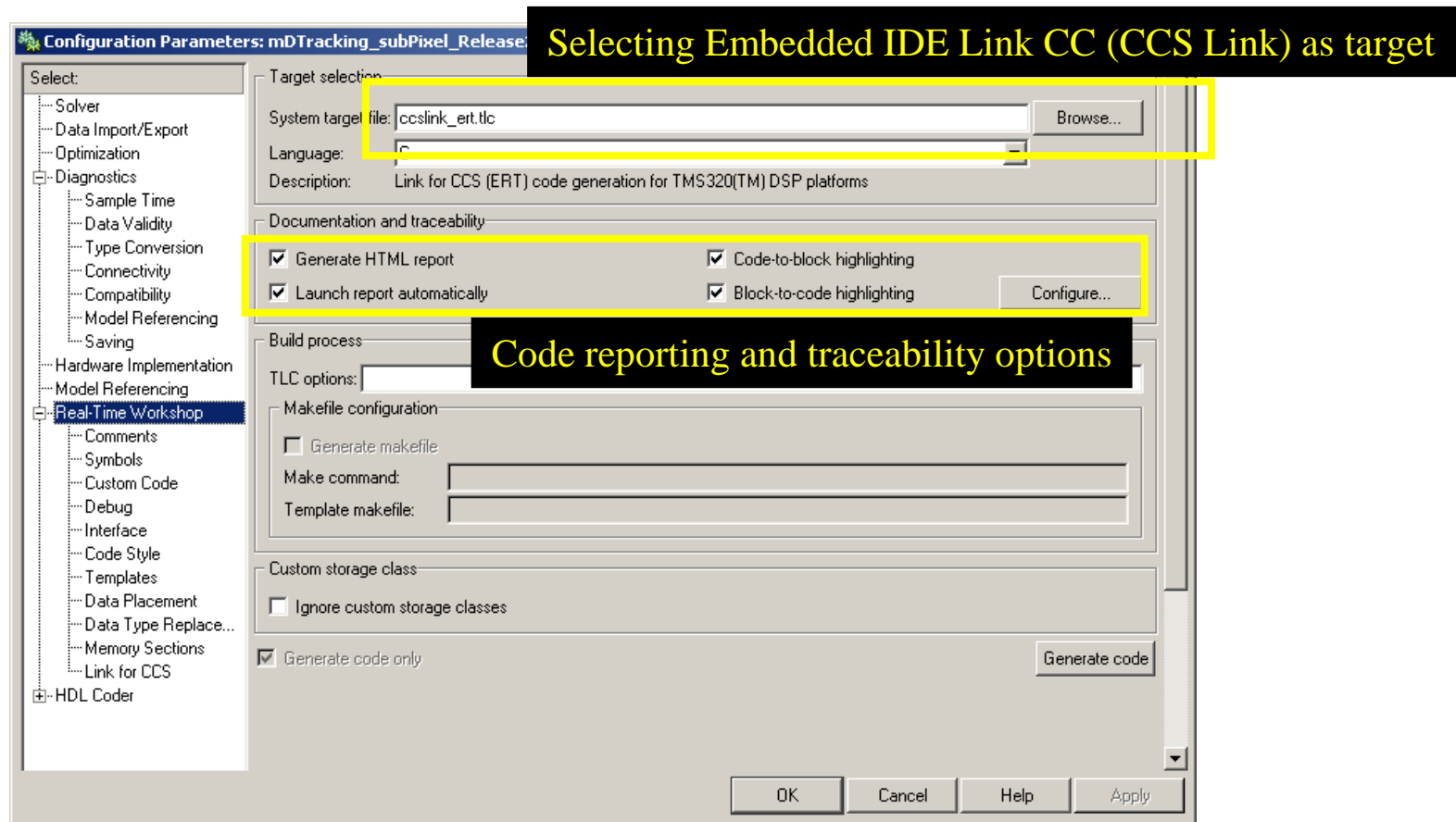
DM6437 Implementation

- Using *Target Preference* blocks to set RTW parameters



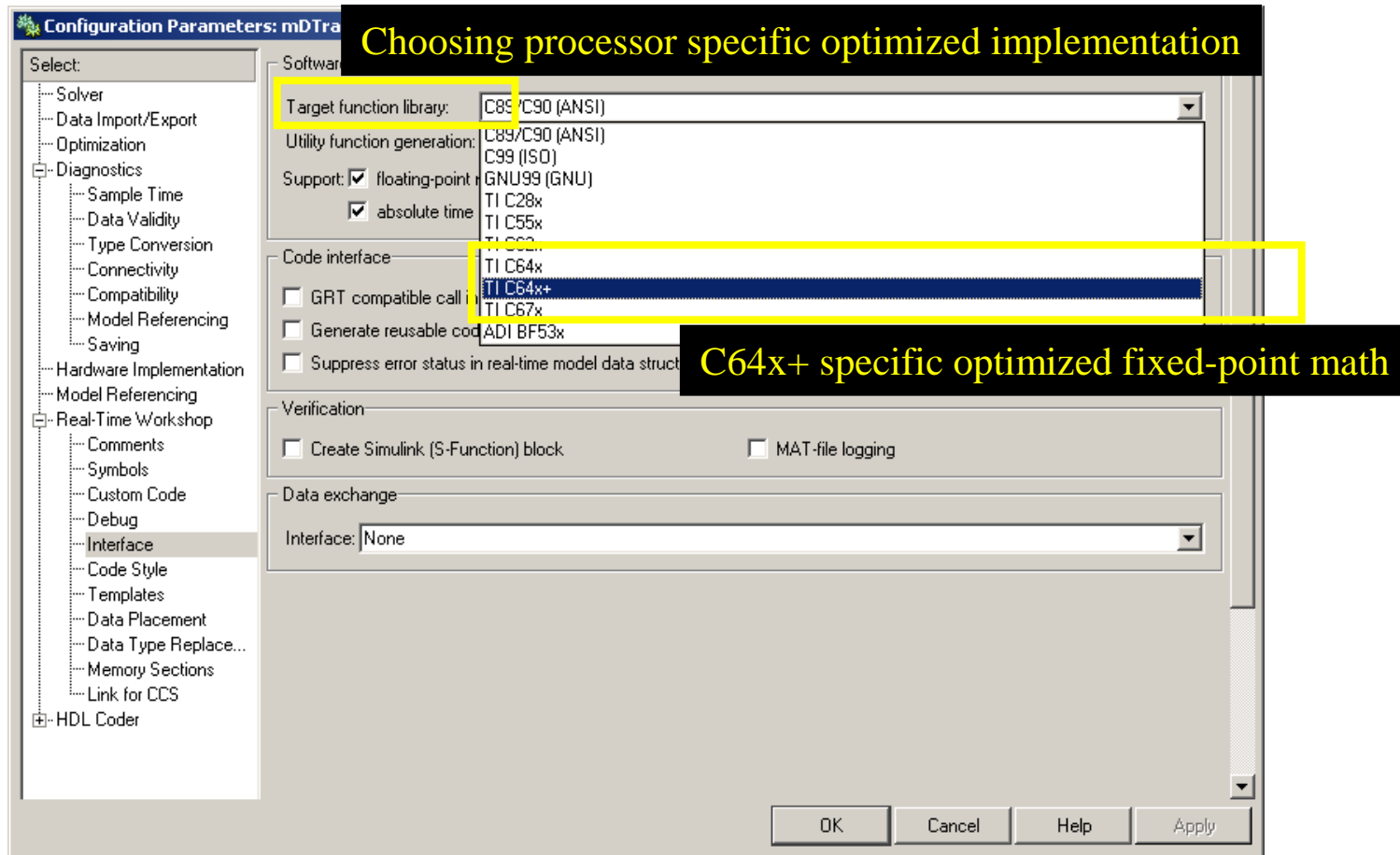
Real-Time Workshop Embedded Coder

- Configuration of Code generation options



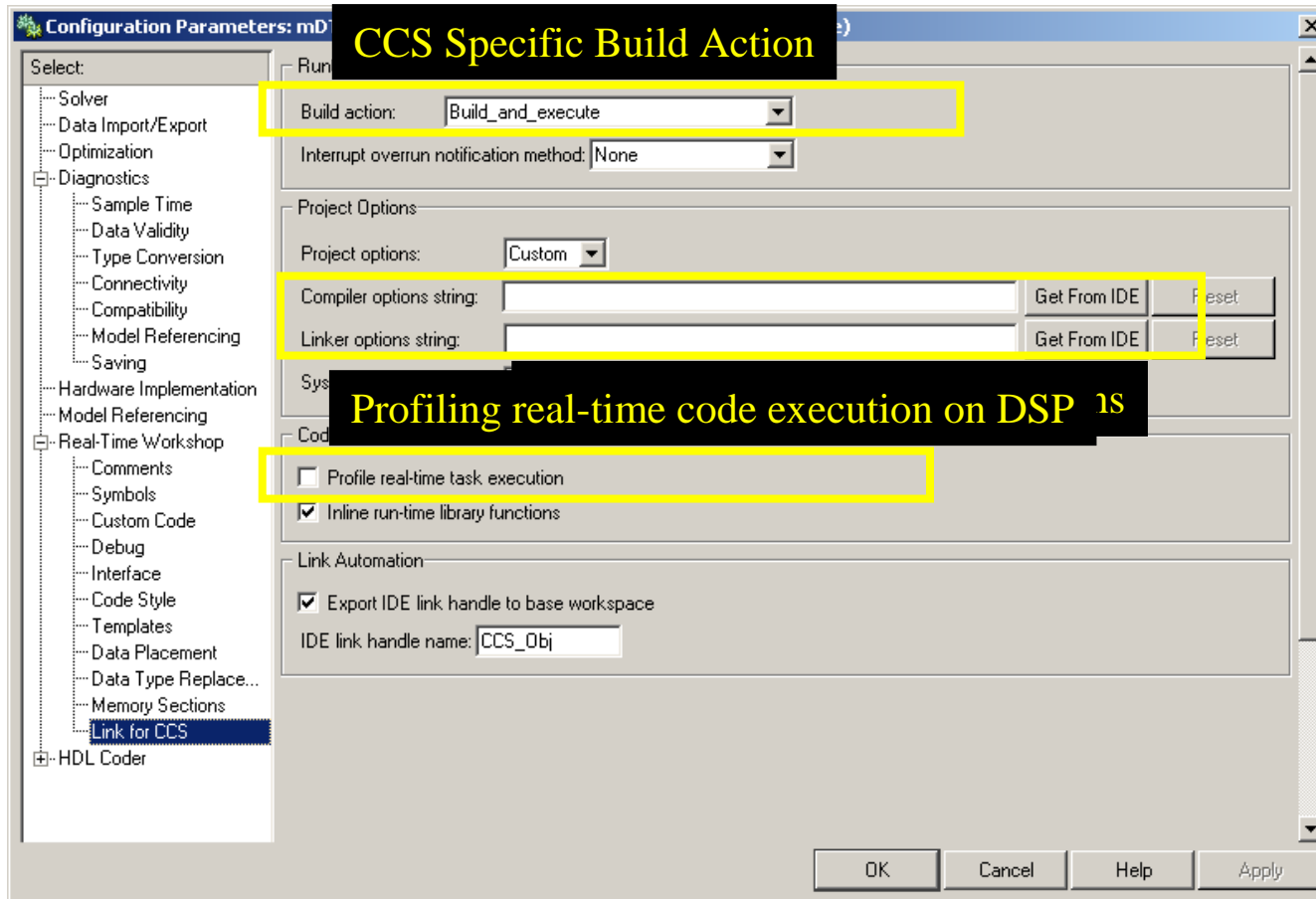
Real-Time Workshop Embedded Coder

- Configuration of Code generation options



Real-Time Workshop Embedded Coder

- Configuration of Code generation options



Integrating Custom Code

C64x+ IMGLIB Assembly Library

- MathWorks and TI have worked together to wrap C64x+ IMGLIB image processing libraries as Simulink blocks
- Blocks Generate function calls to the optimized libraries
- Simulated using the functional equivalent C code on the host PC
- *Download from:*

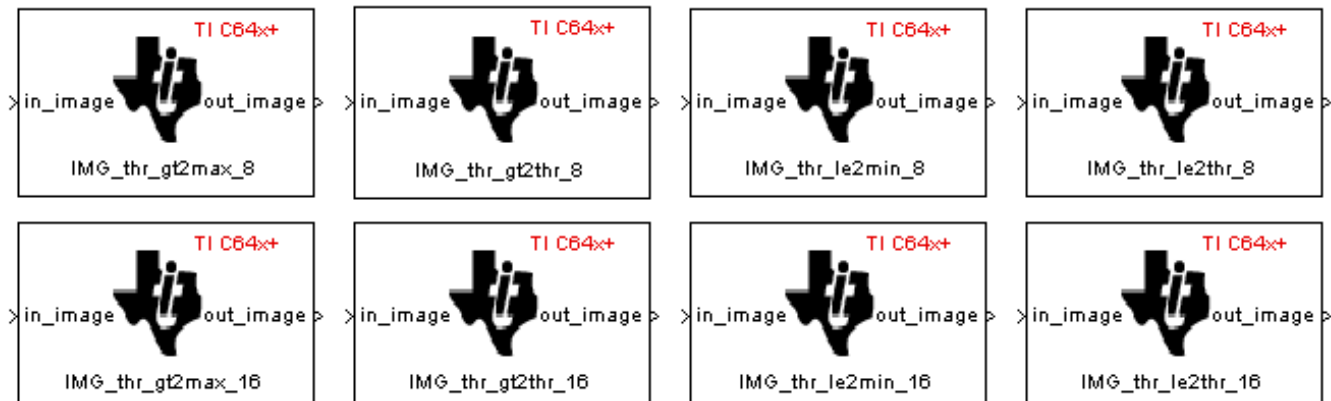
<http://focus.ti.com/docs/toolsw/folders/print/sprc589.html>

C64x+ IMLIB in Simulink

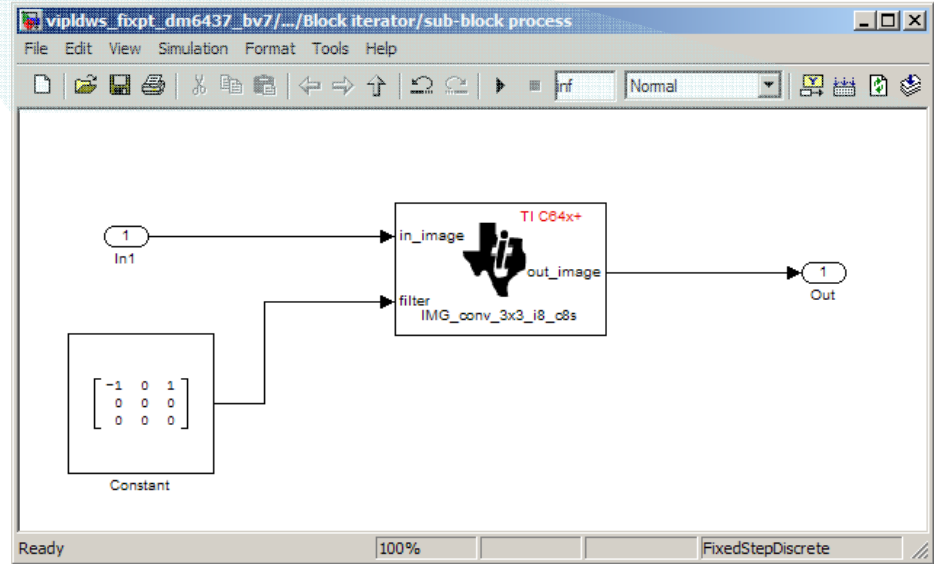
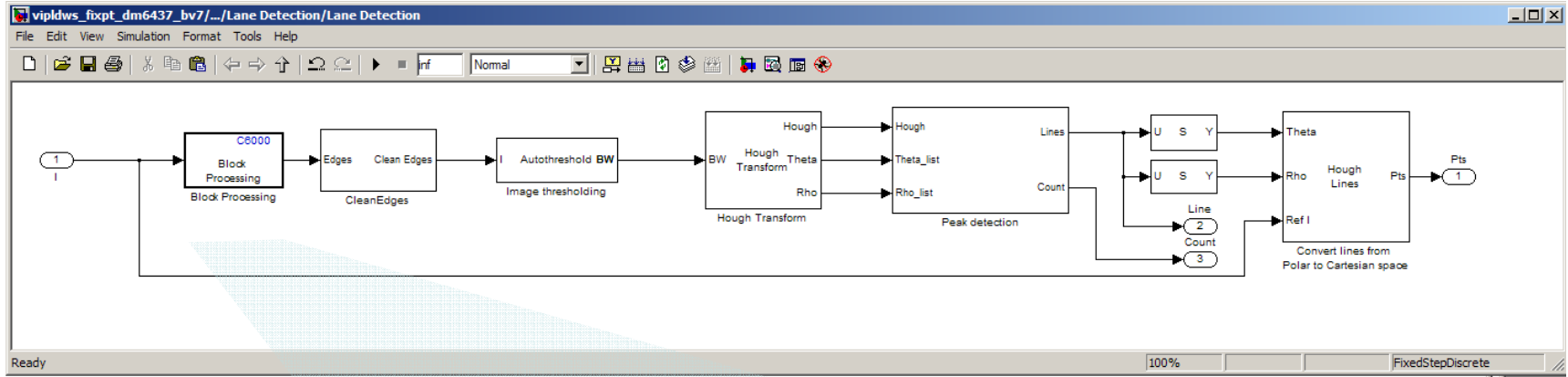
Edge Detection Functions



Image Threshold Functions

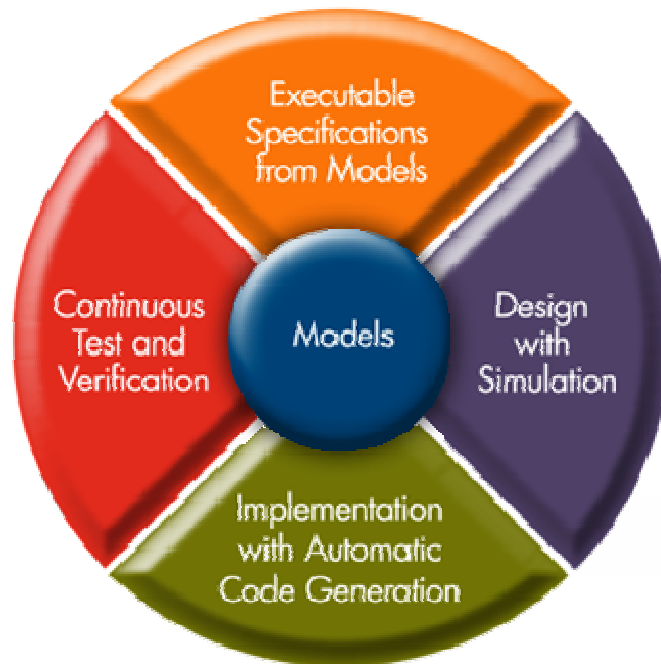


C64x+ IMGLIB Vertical Filter w/ DMA



MBD Implementation Decisions

- ✓ Automatic Code Generation
- ✓ Language Specialization (C, HDL)
- ✓ Design Space Exploration
- ✓ System Integration



Doheny Eye Institute Develops Next-Generation of Retinal Prosthesis with MathWorks Tools

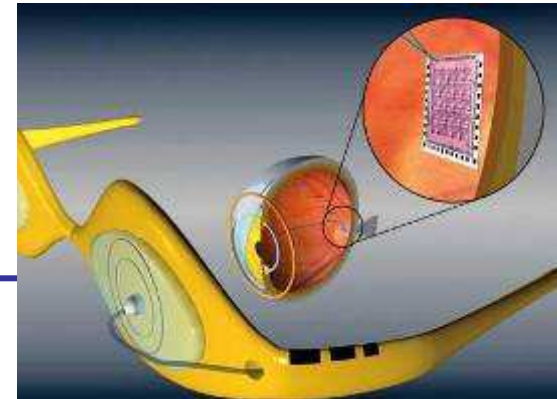


Illustration of a retinal prosthesis prototype .

The Challenge

To develop next-generation, higher-resolution retinal prostheses

The Solution

Use MathWorks tools to develop, simulate, and automatically generate code for real-time image processing algorithms

The Results

- Development time reduced from months to weeks
- DSP deployment streamlined
- Patient testing improved

“With Video and Image Processing Blockset and Embedded Target for TI C6000, we rapidly prototype our image and video processing algorithms on the DM642 board. This can save me days or weeks of time.”

Neha Parikh,
Doheny Eye Institute

Next Steps

- Visit us at the Table to see other demos
- Contact MathWorks representative to get a trial and more information

Thank You!