# Hardware-Software Implementation With Model-Based Design

**Sudhir Sharma**
Product Manager,
HDL Code Generation And Verification
The MathWorks

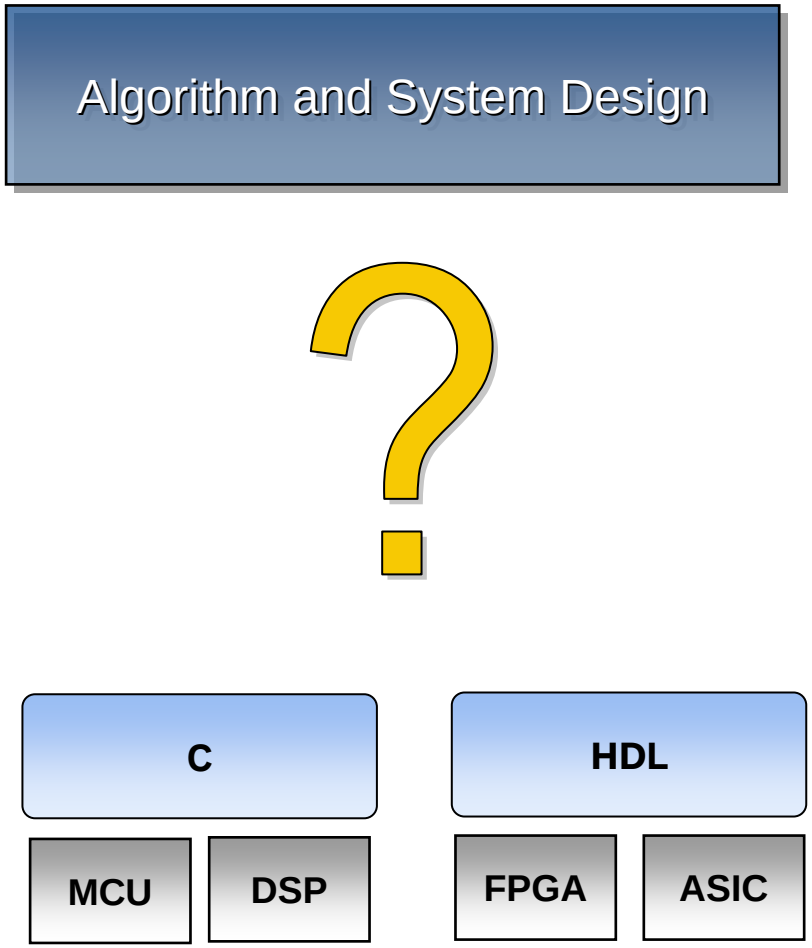


MathWorks
Aerospace and Defense Conference '07

# Agenda

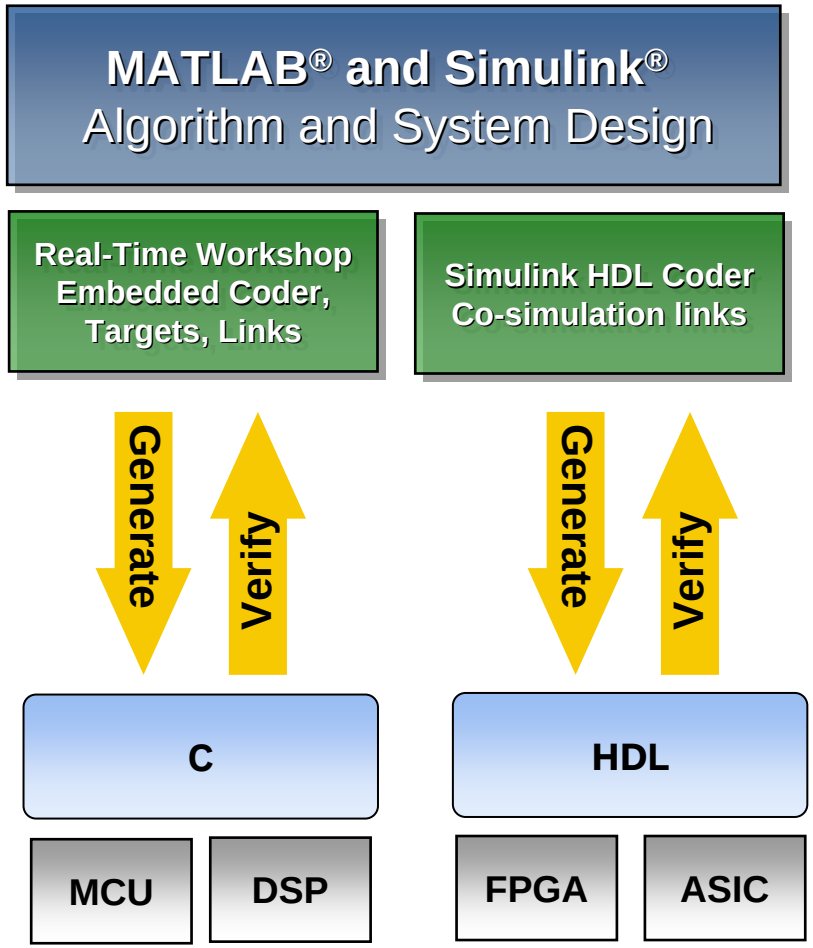
- What is the System Design Challenge
- Solutions for Embedded Software Development
  - Automatic Code Generation
  - Verification
- Solutions for Hardware Development
  - Automatic Code Generation
  - Verification

# System design to implementation gap

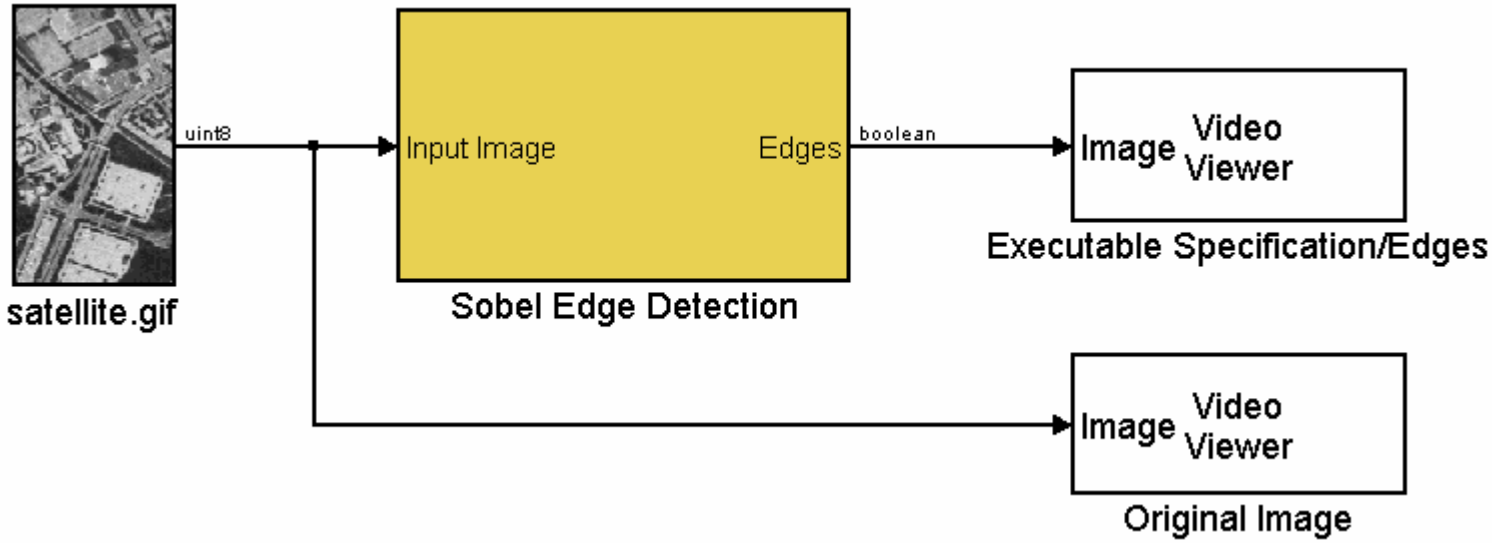


# Integrated Design Flow for Embedded Software and Hardware

- Design, simulate, and validate system models and algorithms in MATLAB and Simulink
- Automatically generate production software for embedded processors
- Verify the software implementation against the system model
- Verify the hardware implementation against the system model

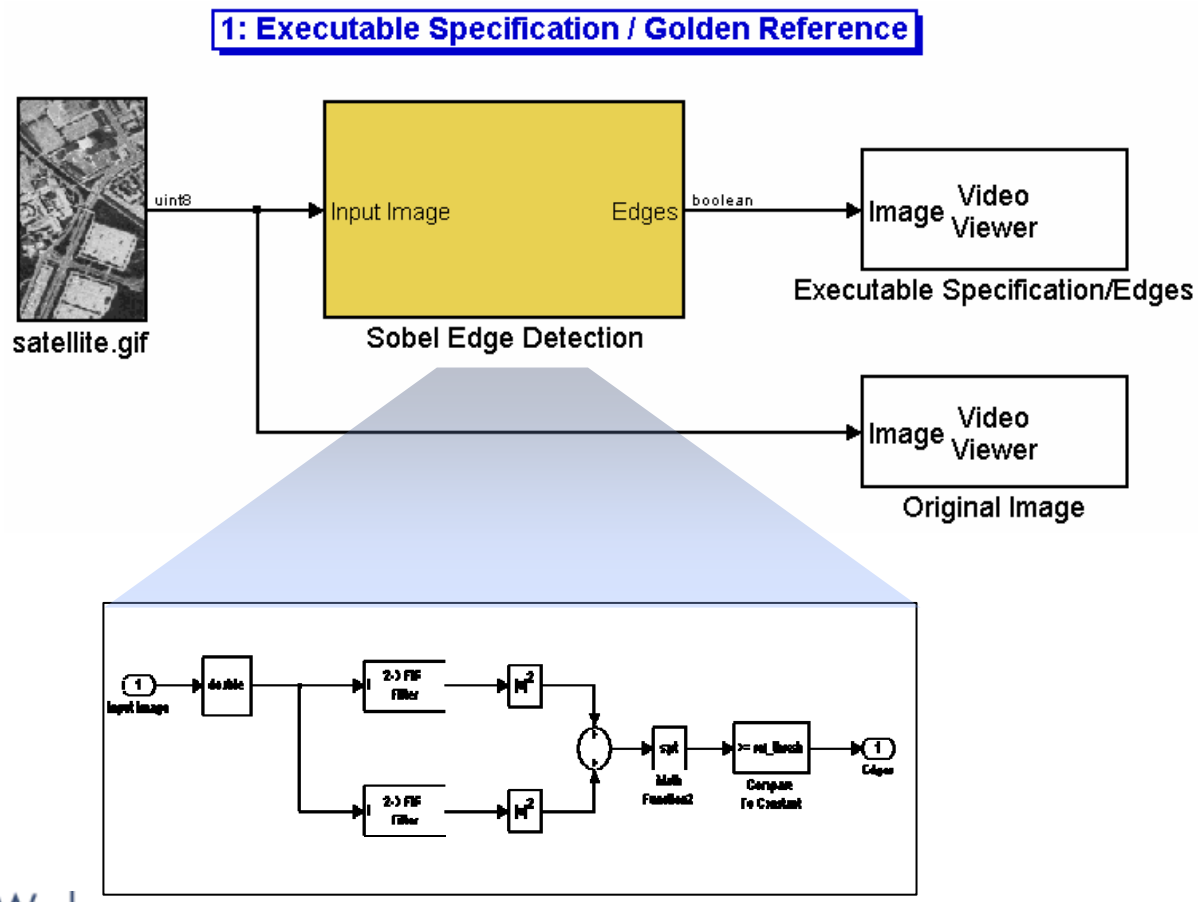


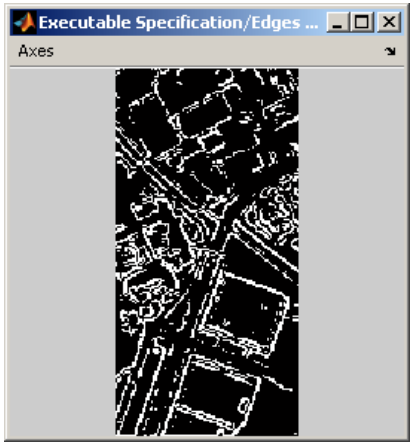
# Case Study: Sobel Edge Detection Algorithm



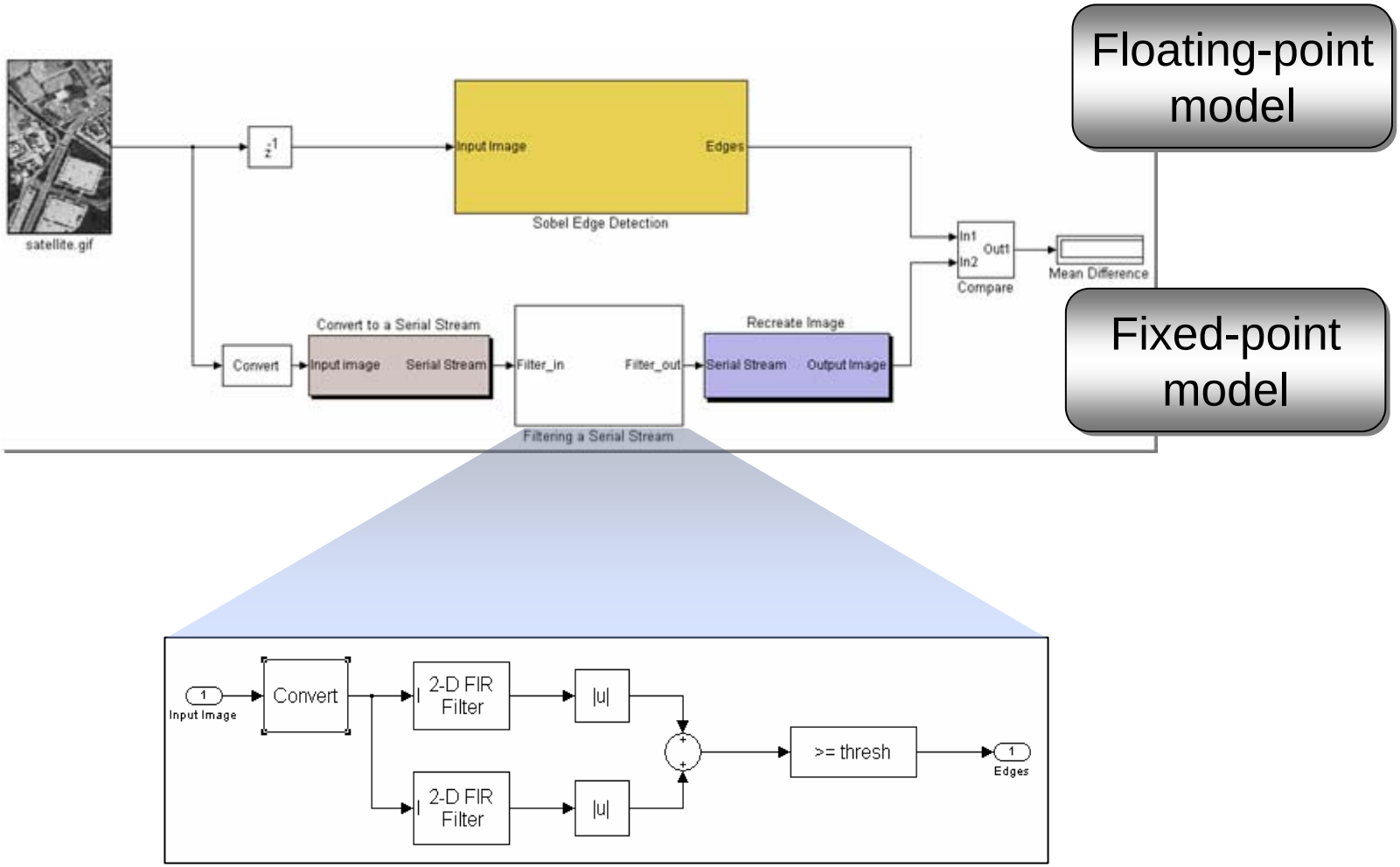
# Floating-Point System Specification

- Start by developing a golden specification

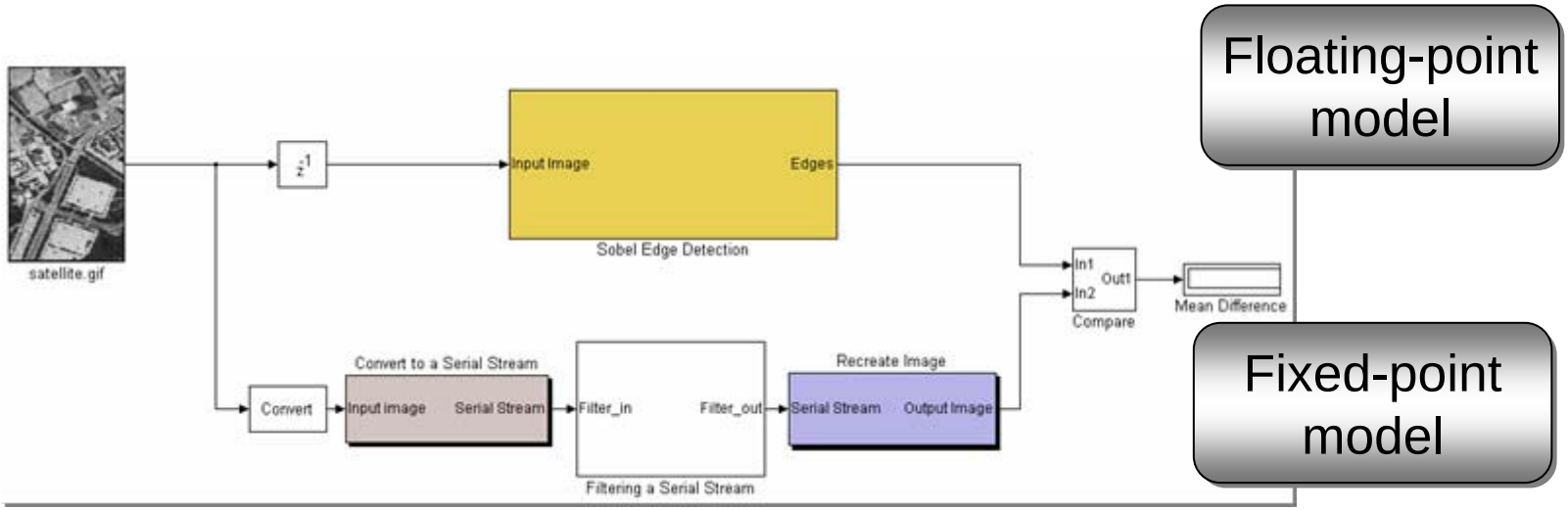


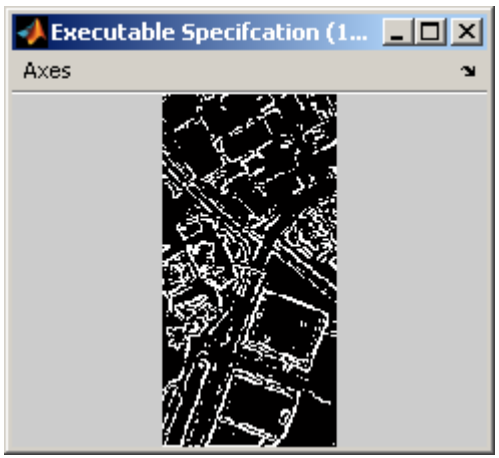



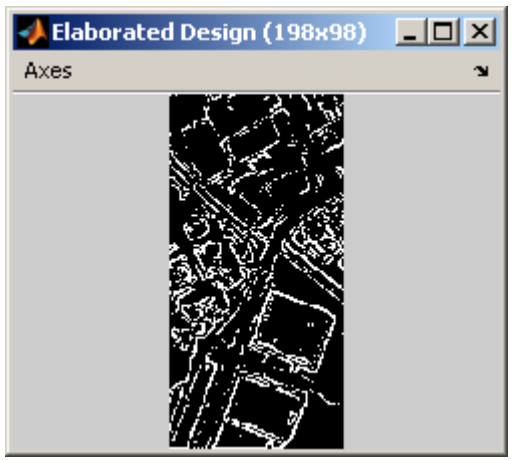

# Fixed-Point Modeling

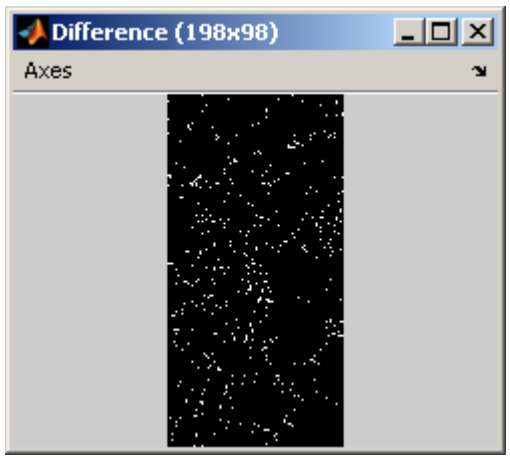


# Fixed-Point Modeling

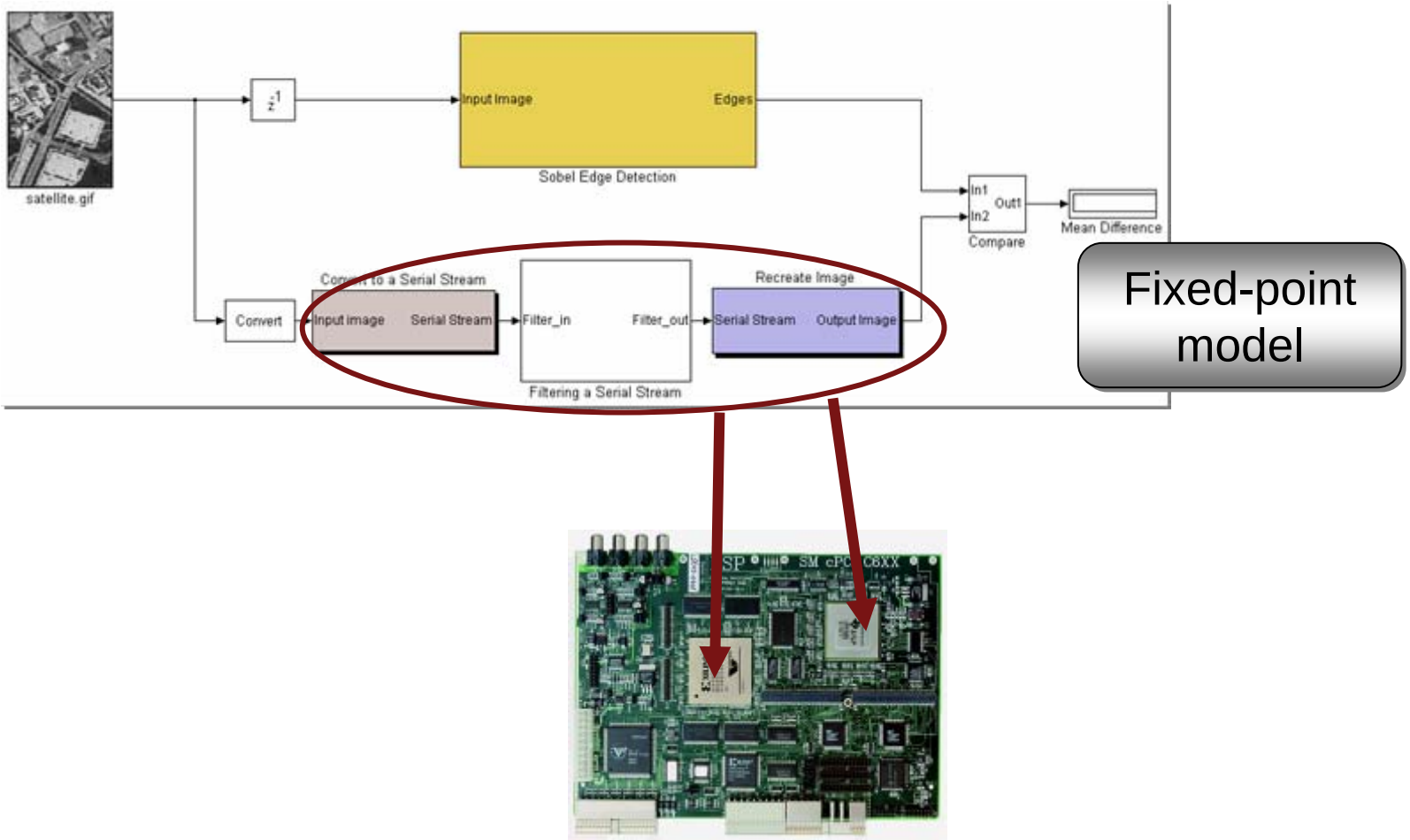


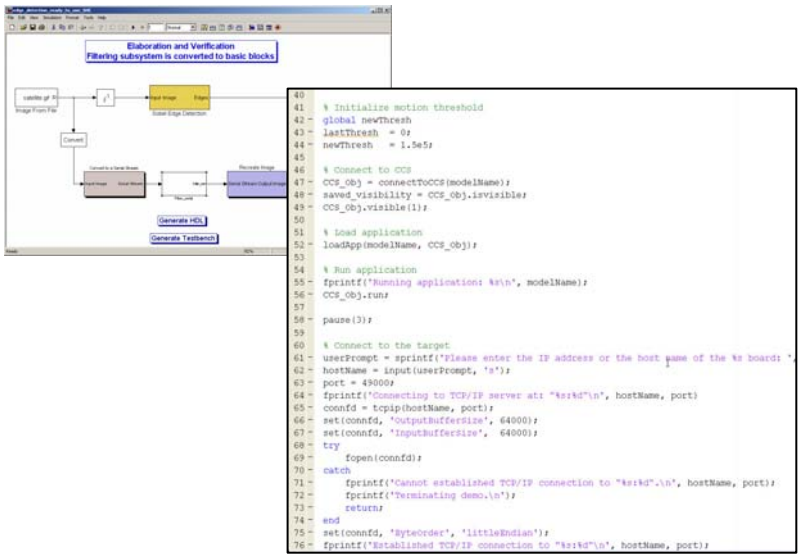



# Implementation on DSP, GPP, or an FPGA?

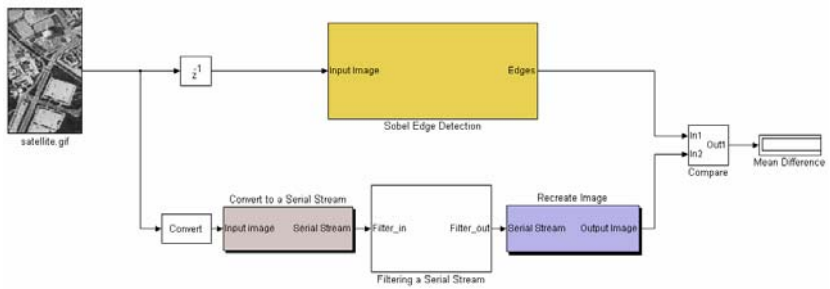


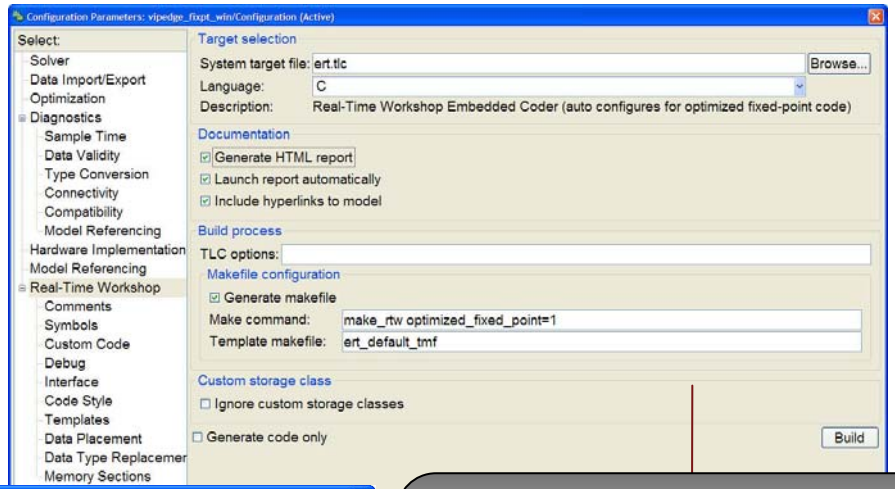
# Agenda

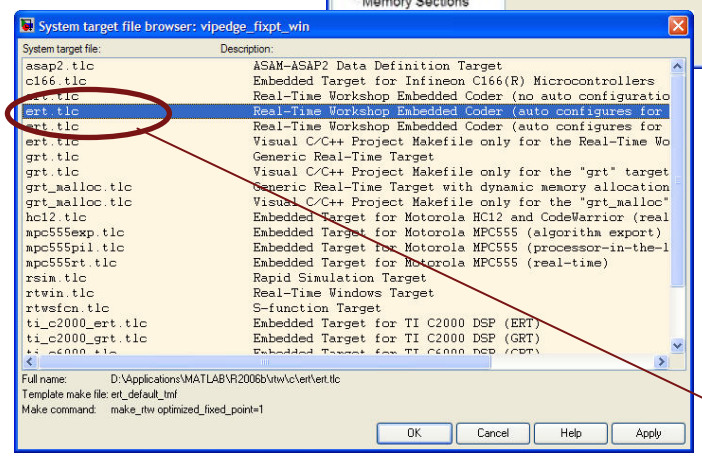
- What is the System Design Challenge
- Solutions for Embedded Software Development
  - Automatic Code Generation
  - Verification
- Solutions for Hardware Development
  - Automatic Code Generation
  - Verification



# Implementation on DSP and GPP



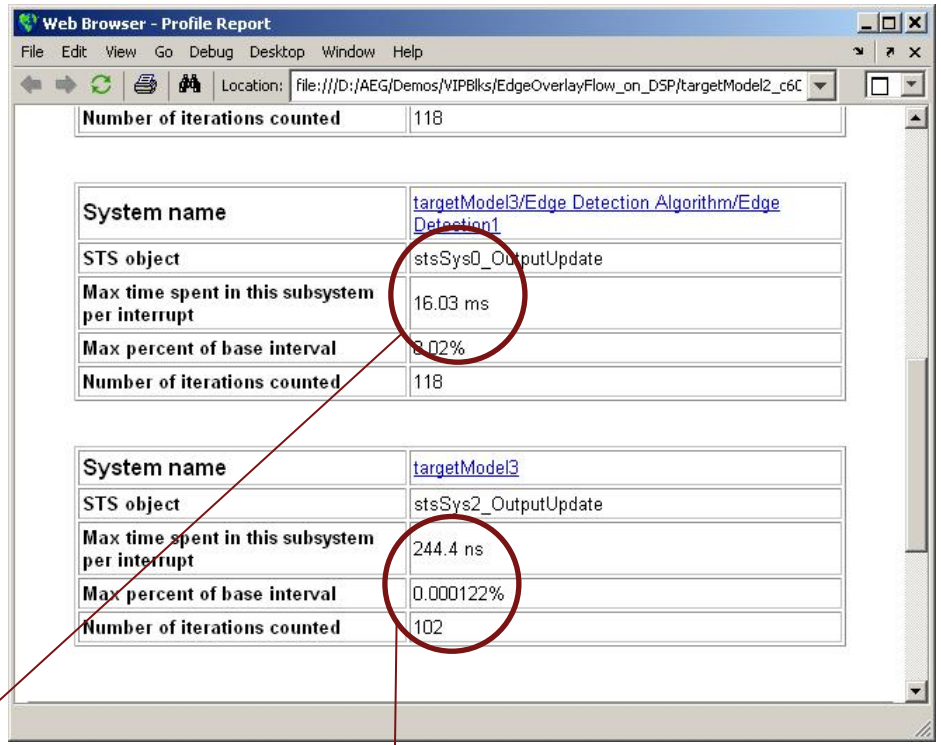

Fixed-point model

Code generation options and preferences

Select target or flavor of generated code

# Code Execution on Target and Profiling

- Build and execute
  - Auto-generate ‘C’ and ASM
  - Integrate RTOS and scheduler
  - Create full CCS project
  - Invoke compiler, linker, and download code
  - Run on target
- Profile code performance




System profiling includes entire DSP application code

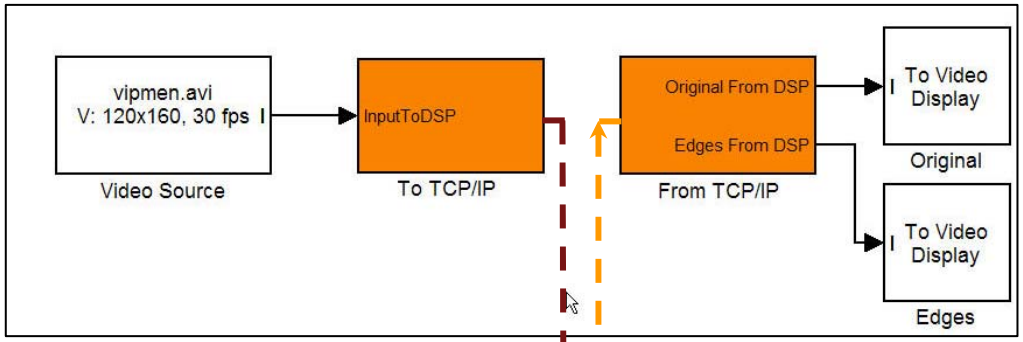
16.03 ms
8.02%
118

244.4 ns
0.000122%
102

Subsystem profiling

# Design Verification and Visualization:

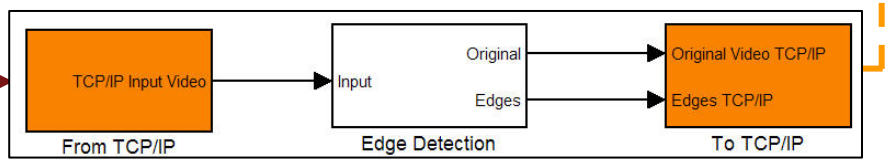
## Simulink as verification test bench




Processor and hardware-in-the-loop testing, simulation, visualization, and verification of embedded software with Simulink

Simulink test bench

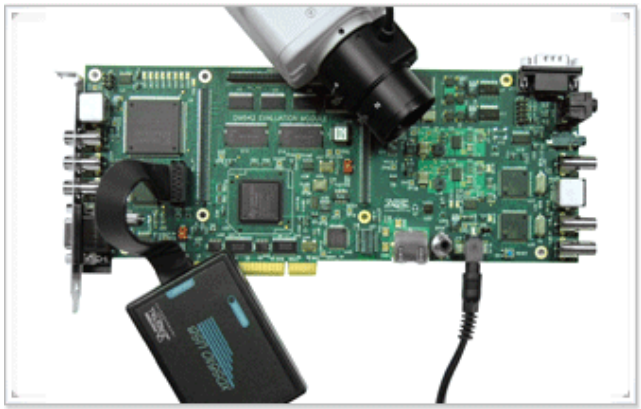
Device or design under test (DUT)




Simulink system design embedded on DSP

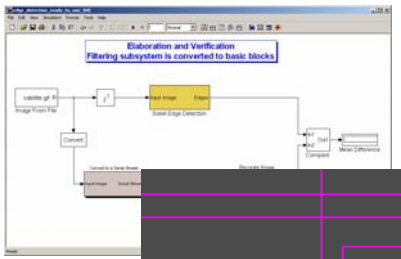
# Review: Code Generation for Embedded Software

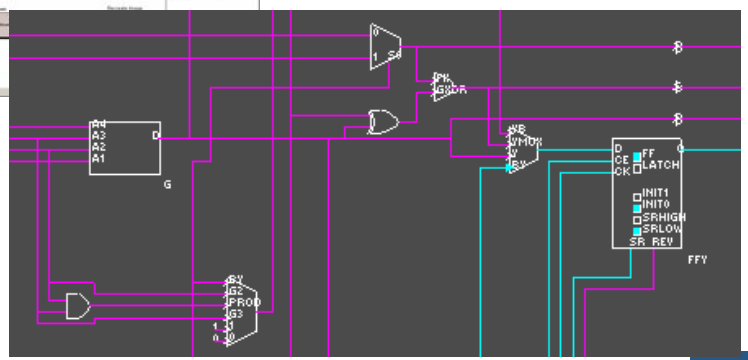
- Code Generation
  - Real Time Workshop – ANSI/ISO C code for rapid prototyping, acceleration
  - Real Time Workshop Embedded Coder – Embedded deployment
- Links
  - Link for Altium TASKING
  - Link for Analog Devices VisualDSP++ New!
  - Link for TI Code Composer Studio
- Targets
  - Target for TI C6000 DSP
  - Target for TI C2000 DSP
  - Target for Infineon C166 Microcontrollers
  - Target for Freescale MPC5xx Microcontrollers



# Agenda

- What is the System Design Challenge
- Solutions for Embedded Software Development
  - Automatic Code Generation
  - Verification
- Solutions for Hardware Development
  - Automatic Code Generation
  - Verification

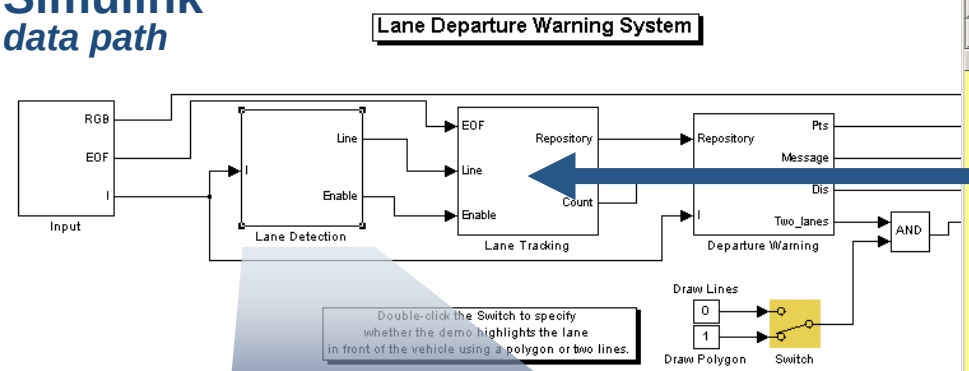


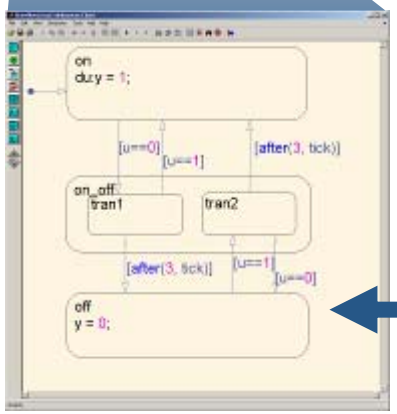
# Code Generation for Hardware

**Simulink HDL Coder**
***Correct-by-construction VHDL and Verilog code***

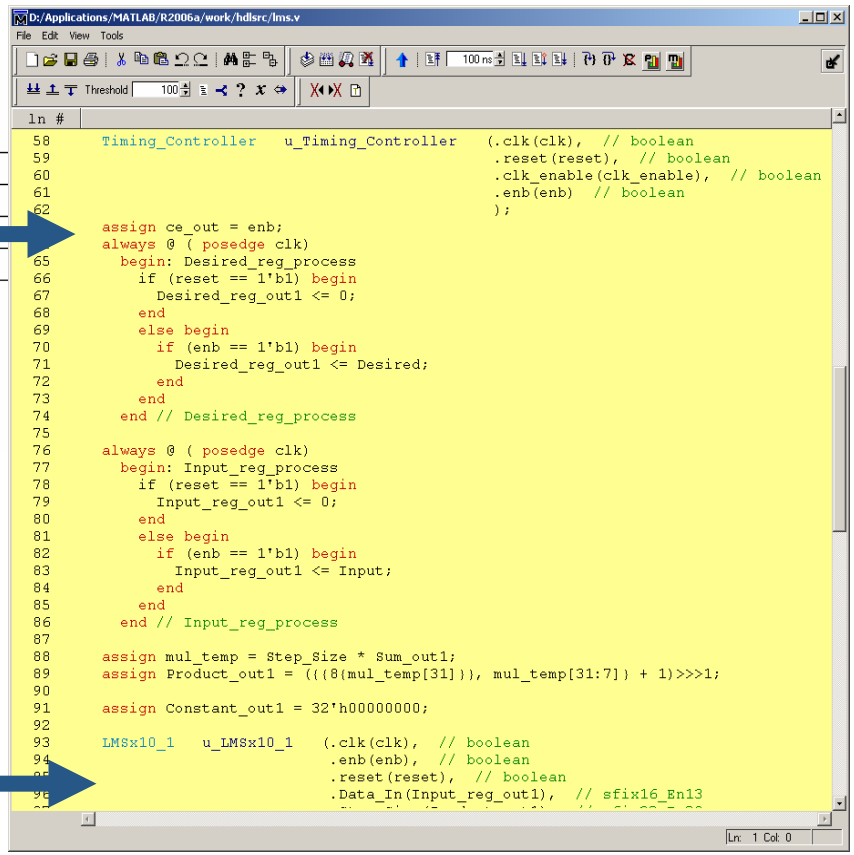
## Simulink
*data path*




## Stateflow
*control logic*

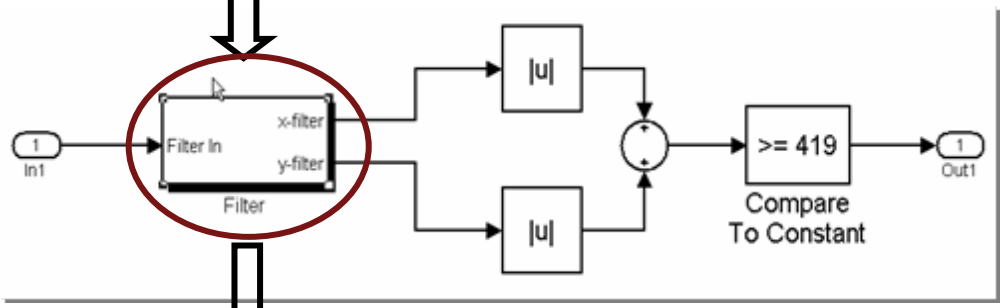



## Generated Verilog code

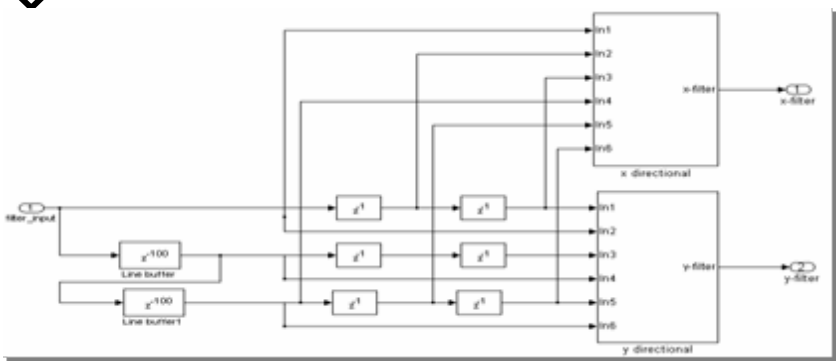


# Fixed-Point Implementation on an FPGA


Floating-point model



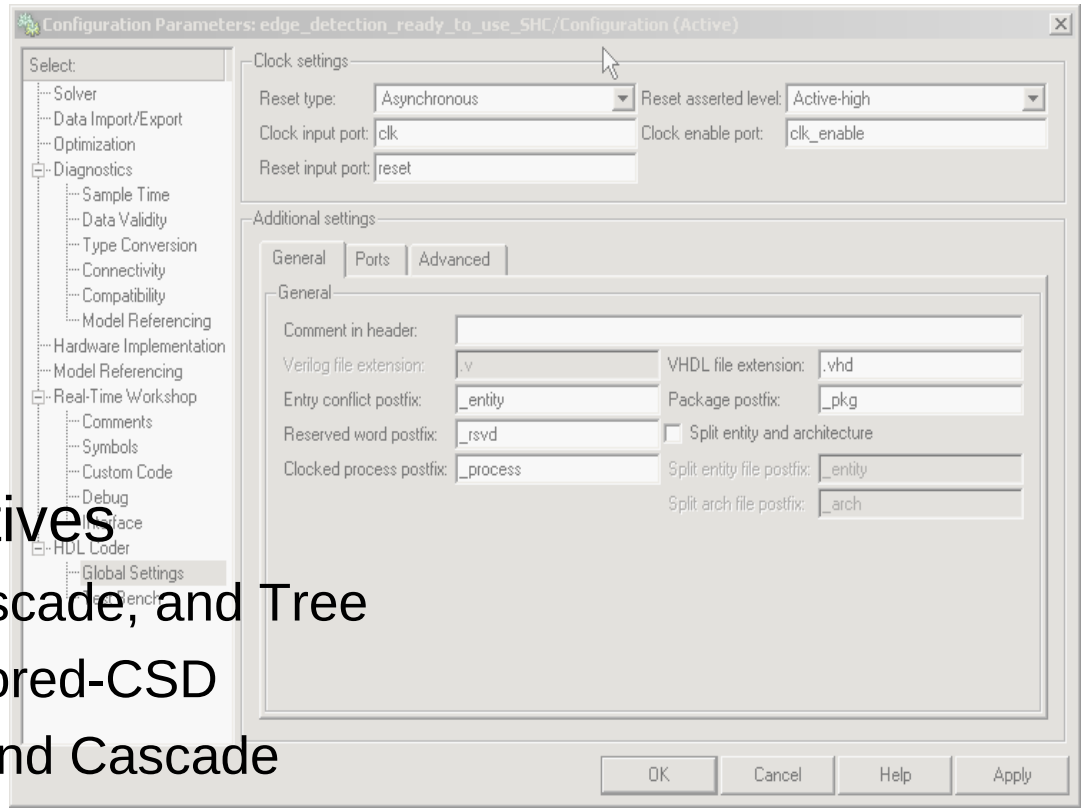

Fixed-point model



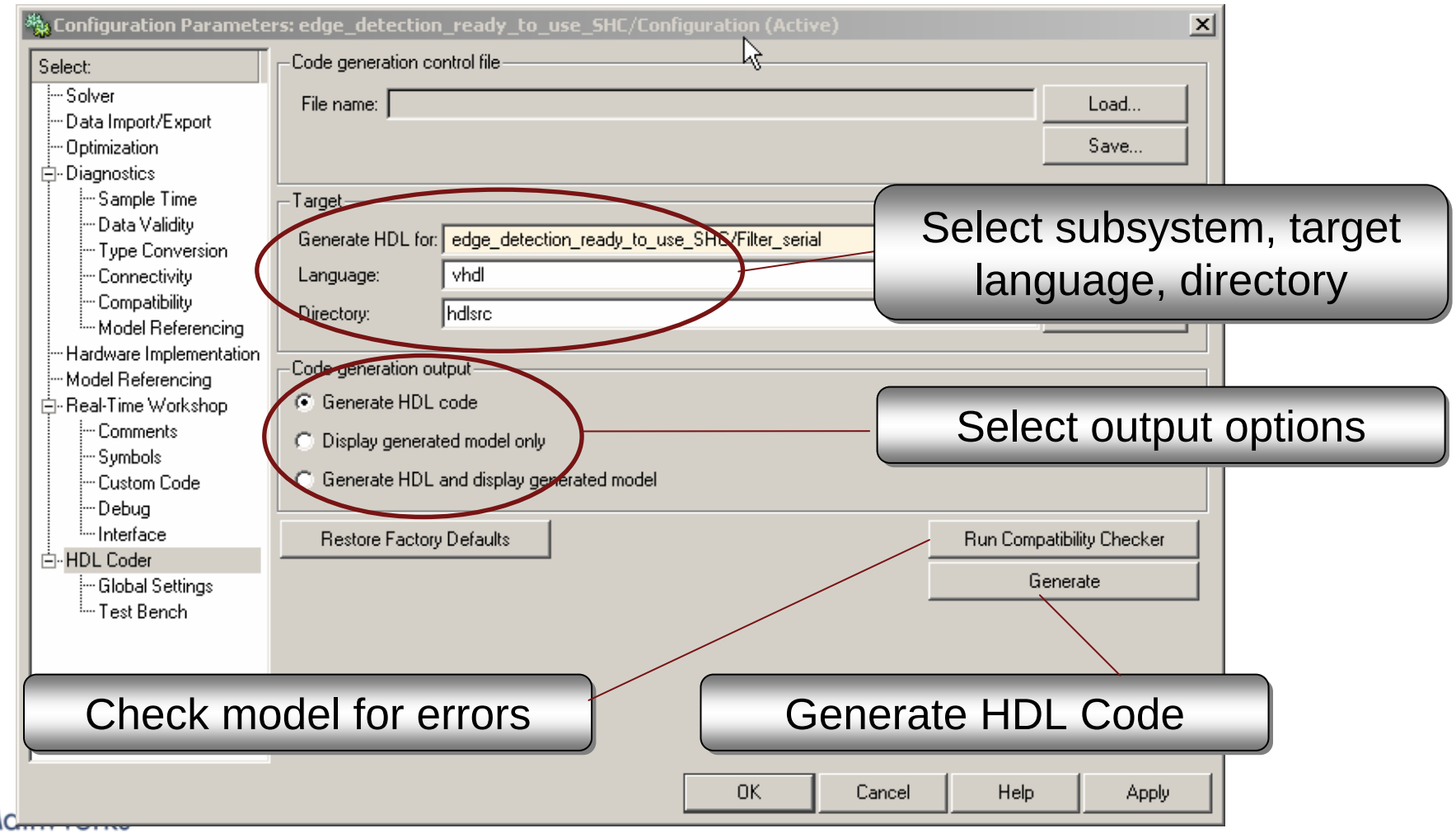

Hardware specific model

# Design Space Exploration

- Speed
  - How fast can this design run?
- Area
  - Can I use a smaller chip?
- Power
  - Can I target a mobile device?
- Implementation Alternatives
  - Sum & Product: Linear, Cascade, and Tree
  - Gain: Multiplier, CSD, Factored-CSD
  - Minimum/Maximum: Tree and Cascade
  - Lookup Table: Inline or hierarchical



# Code Generation Options



# More Code Generation Options

Configuration Parameters: edge_detection_ready_to_use_SHC/Configuration (Active)

Select:
- Solver
- Data Import/Export
- Optimization
- Diagnostics
  - Sample Time
  - Data Validity
  - Type Conversion
  - Connectivity
  - Compatibility
  - Model Referencing
- Hardware Implementation
- Model Referencing
- Real-Time Workshop
  - Comments
  - Symbols
  - Custom Code
  - Debug
  - Interface
- HDL Coder
  - Global Settings
  - Test Bench

Clock settings

Reset type: Asynchronous
Reset asserted level: Active-high
Clock input port: clk
Clock enable port: clk_enable
Reset input port: reset

Additional settings

General | Ports | Advanced

General

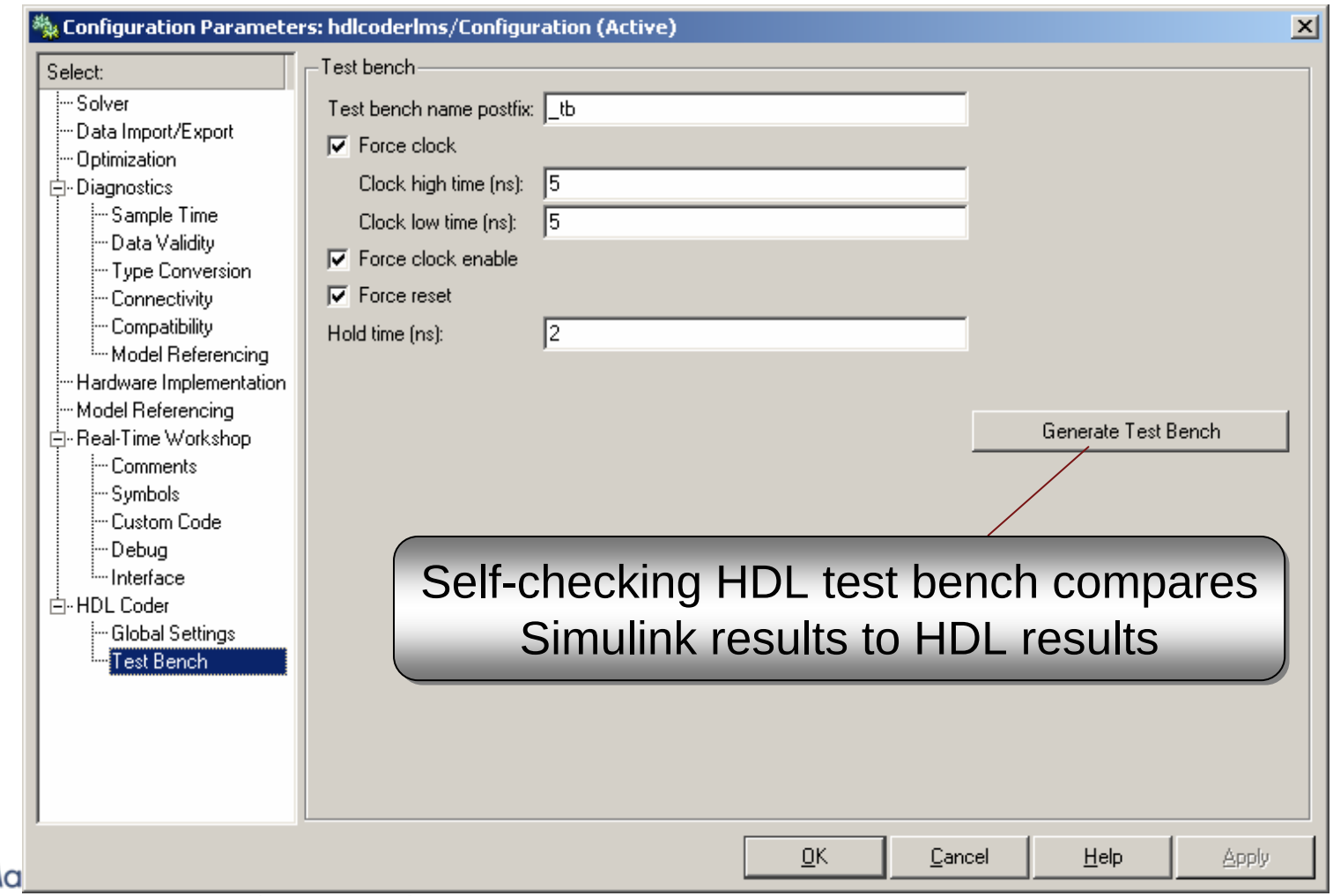
Comment in header:
Verilog file extension: .v
VHDL file extension: .vhd
Entry conflict postfix: _entity
Package postfix: _pkg
Reserved word postfix: _rsvd
Split entity and architecture
Clocked process postfix: _process
Split entity file postfix: _entity
Split arch file postfix: _arch

OK | Cancel | Help | Apply

Select reset and clock options

Set language-specific options: input/output datatypes, timescale directives, …

# Generate HDL Test Bench

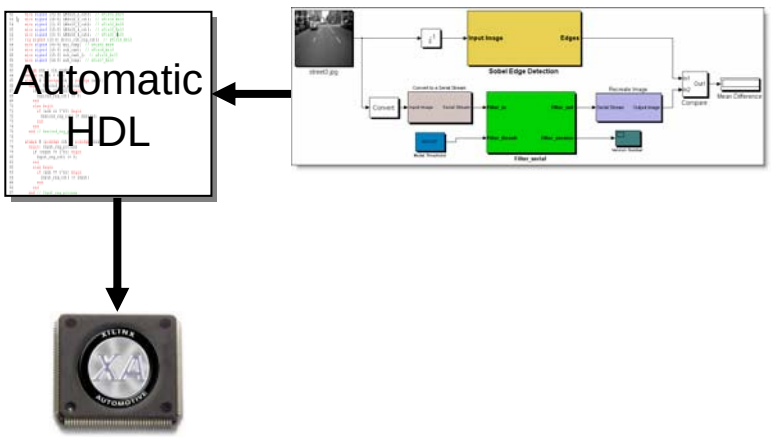



Self-checking HDL test bench compares Simulink results to HDL results

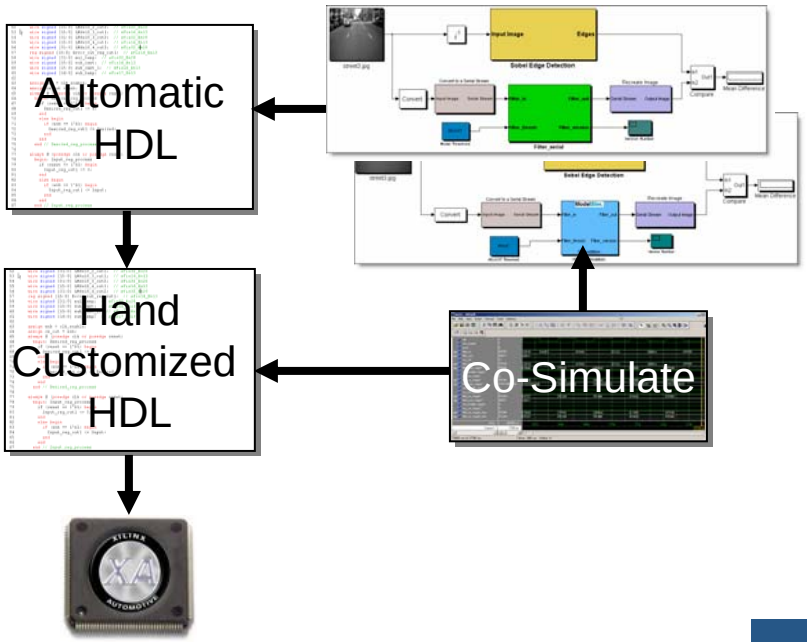
# Automatic HDL Code Generation

- ‘Correct-by-construction’
  - Matches Fixed-Point System Model
  - Faster design implementation
  - Reduces verification burden
- Benefits Include:

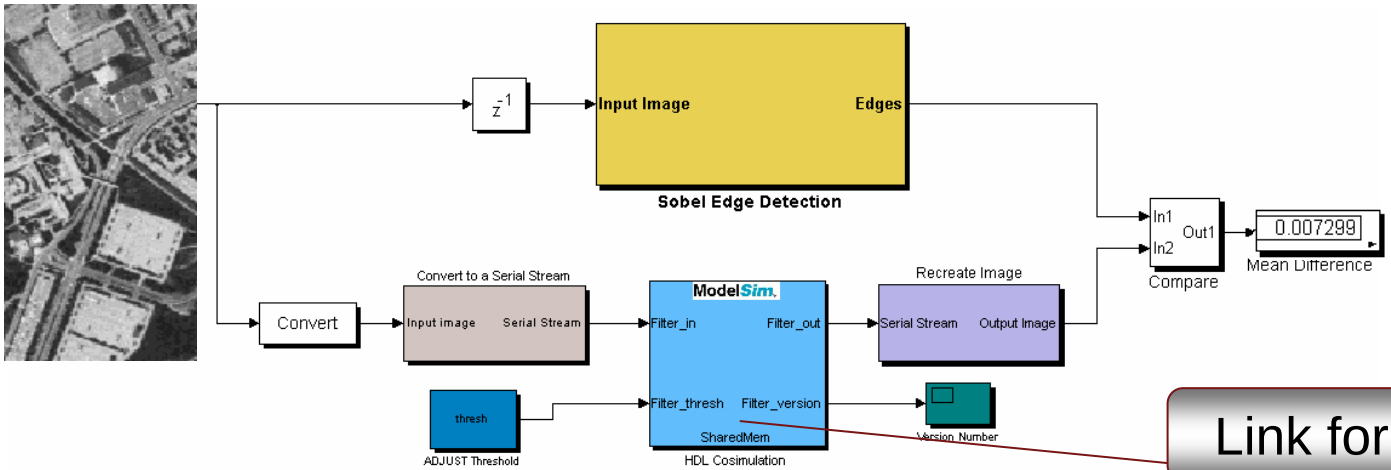
1 Rapid FPGA implementation




2 Reference code for HDL engineers

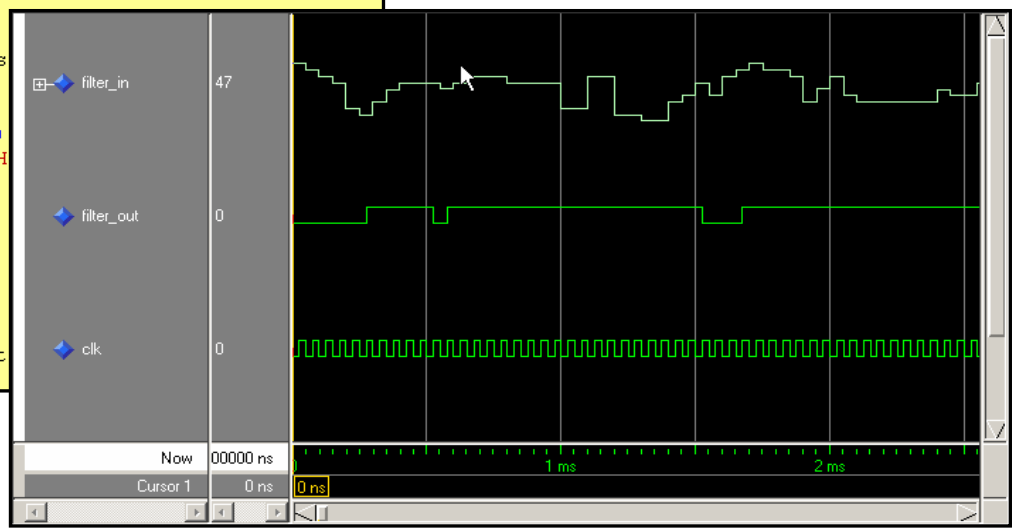


# Verification with system specification



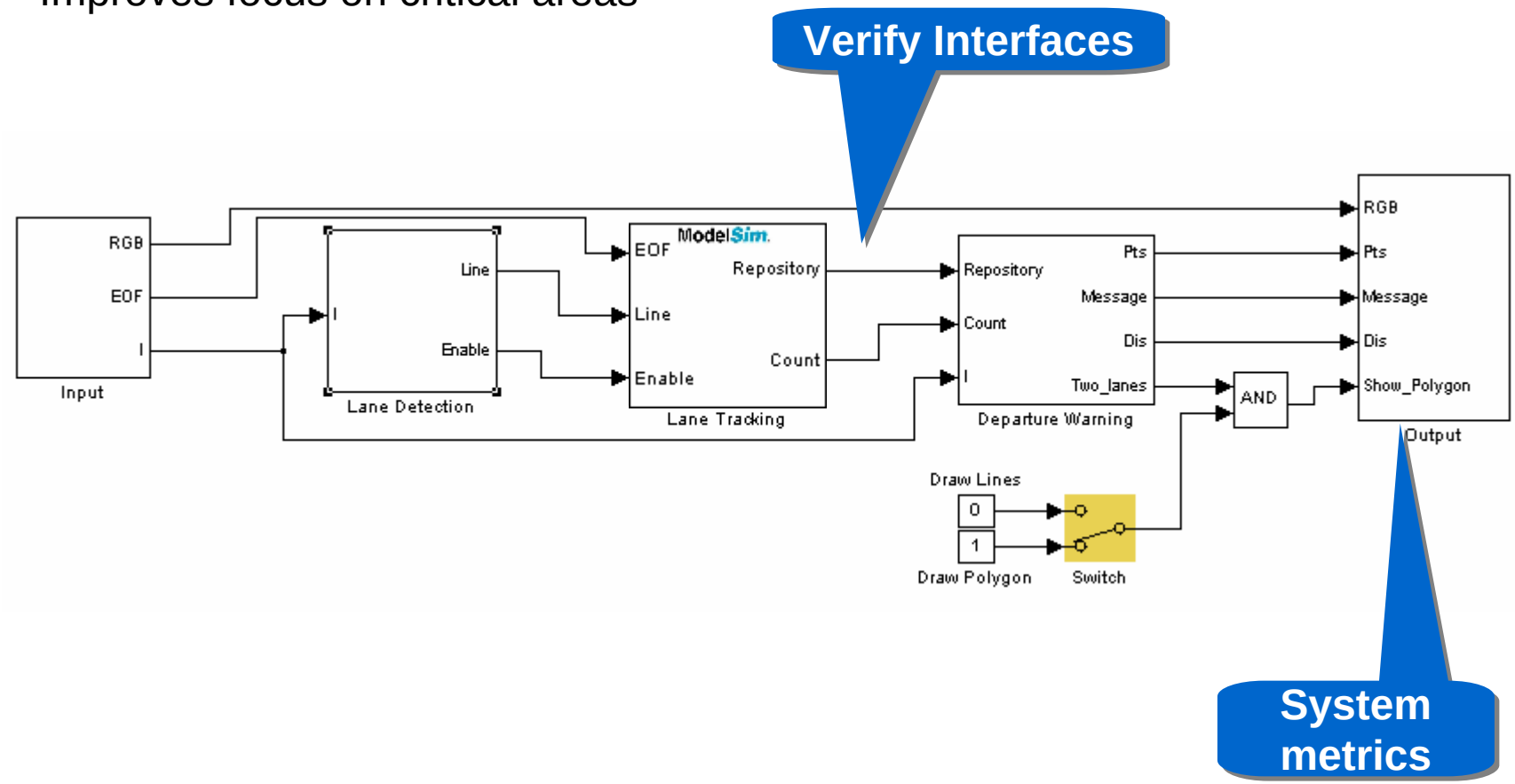

Link for ModelSim

```
117          y_filter => y_directional_out1  -- ufix11
118          );
119      s <= signed(filter_input);
120
121      Delay2_process : PROCESS (clk, res
122      BEGIN
123        IF reset = '1' THEN
124          Delay2_out1 <= (OTHERS => '0')
125        ELSIF clk'event AND clk = '1' TH
126          IF enb = '1' THEN
127            Delay2_out1 <= s;
128          END IF;
129        END IF;
130      END PROCESS Delay2_process;
131
132
133      s_1 <= std_logic_vector(Delay2_out
134
```



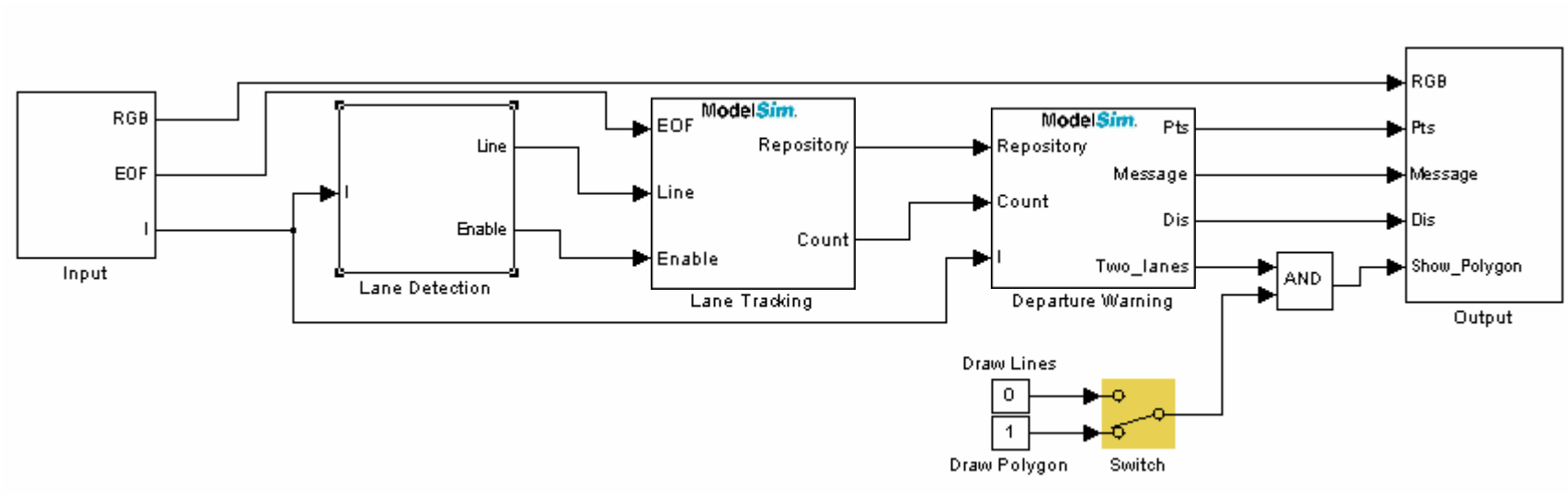
# Making full use of the system model

- Promotes parallelism in design and verification tasks
- Improves focus on critical areas



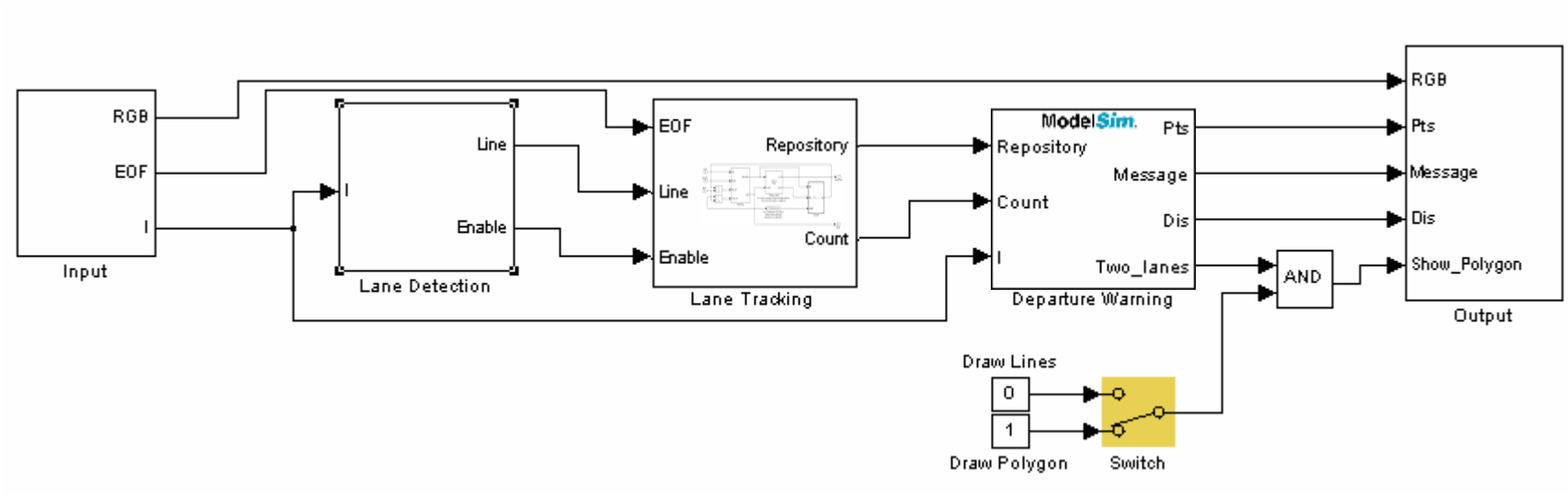
# Making full use of the system model

- Promotes parallelism in design and verification tasks
- Improves focus on critical areas
- Accelerates verification at all levels



# Making full use of the system model

- Promotes parallelism in design and verification tasks
- Improves focus on critical areas
- Accelerates verification at all levels
- Supports re-use and “what-if” scenarios



# Implementation on an FPGA

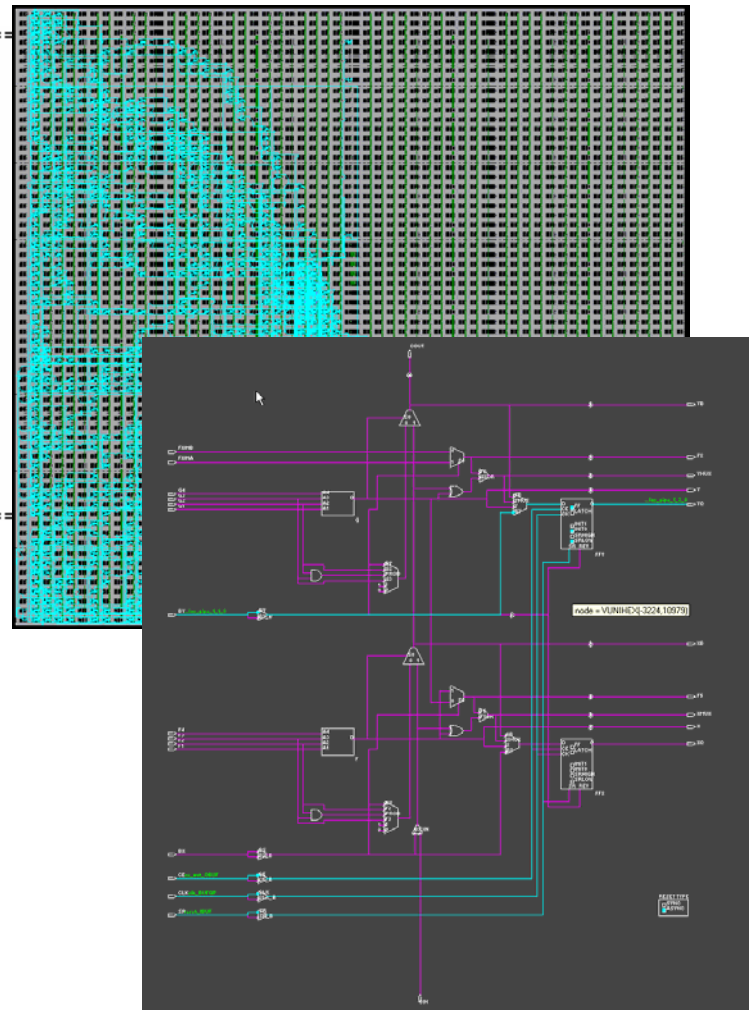
```
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 4vsx25ff668-12

 Number of Slices:                    1105  out of  10240     10%
 Number of Slice Flip Flops:          1903  out of  20480      9%
 Number of 4 input LUTs:               156  out of  20480      0%
 Number of IOs:                         14
 Number of bonded IOBs:                 14  out of    320      4%
 Number of GCLKs:                        1  out of     32      3%


Design statistics:
   Minimum period:   4.106ns   (Maximum frequency: 243.546MHz)
   Minimum input required time before clock:   3.883ns
   Maximum output delay after clock:   7.323ns
```



# Review: Code Generation for Hardware

- Code Generation
  - Simulink® HDL Coder – FPGA and ASIC deployment using VHDL and Verilog New!
  - Filter Design HDL Coder – Filter implementation from MATLAB
- Links
  - Link for Mentor ModelSim
  - Link for Cadence® Incisive® New!

# Summary

- Design and verify <u>software and hardware</u> from MATLAB and Simulink
- Accelerate product development using Model-Based Design