

# 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

Algorithm and System Design



C

MCU

DSP

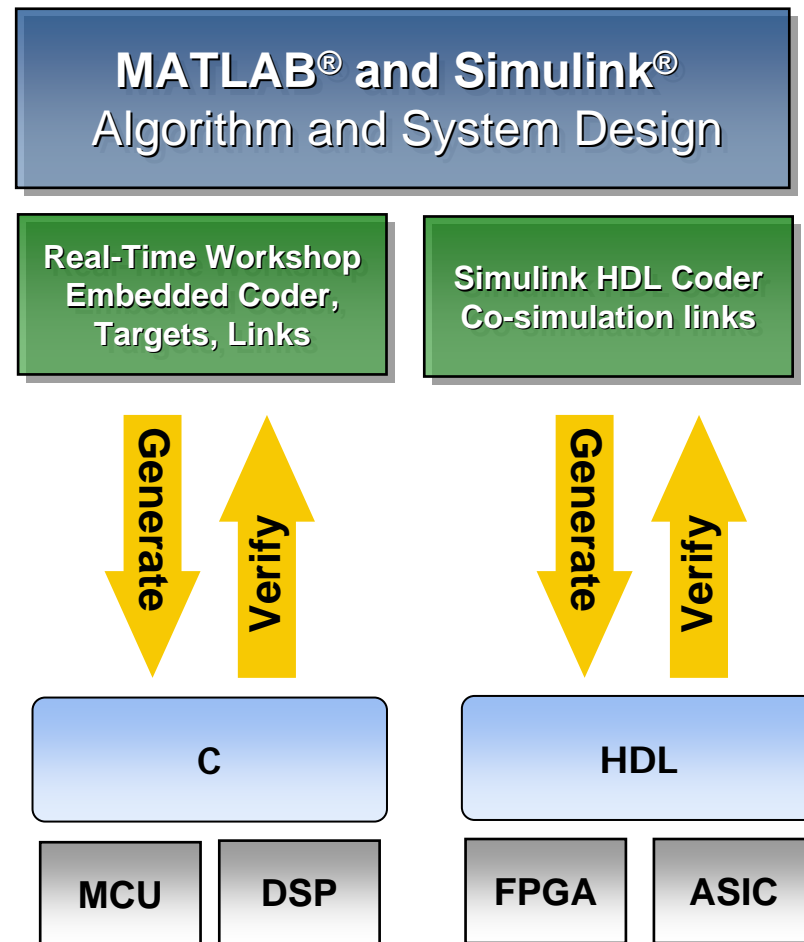
HDL

FPGA

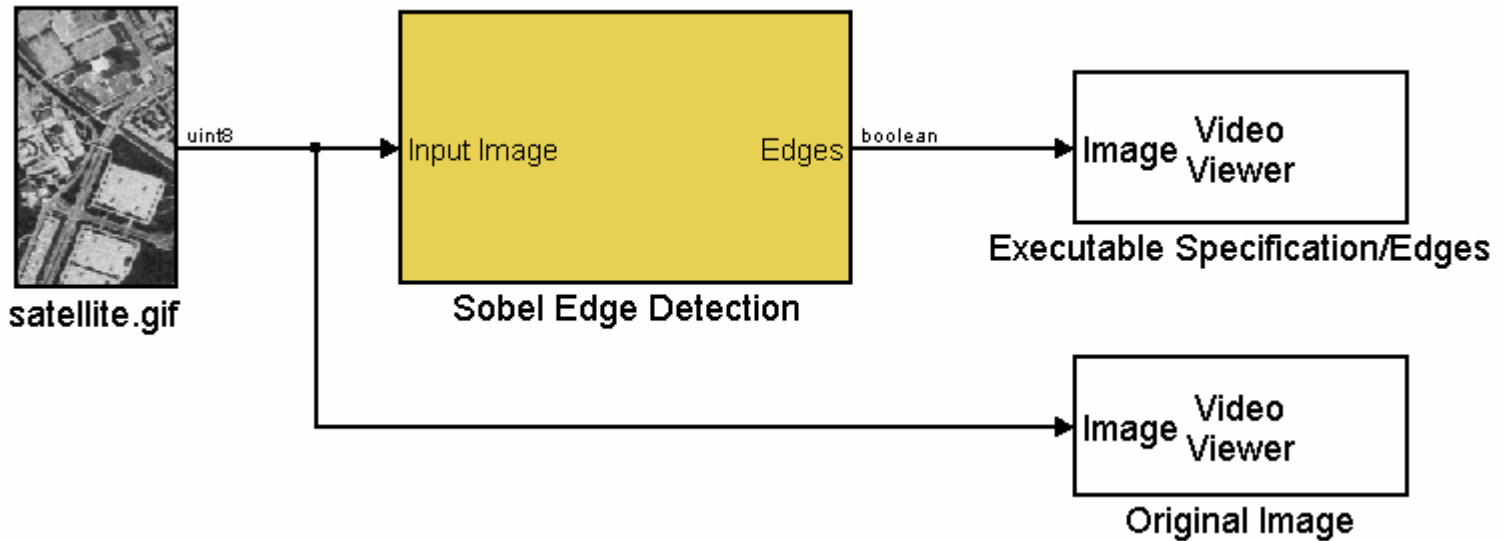
ASIC

# 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



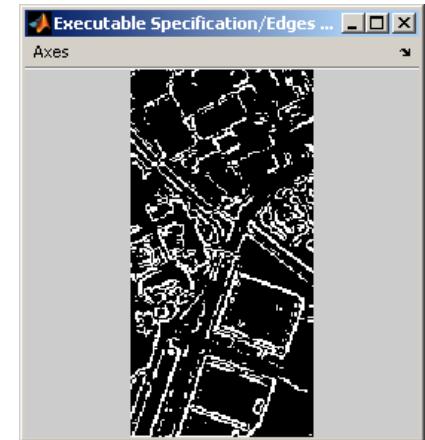
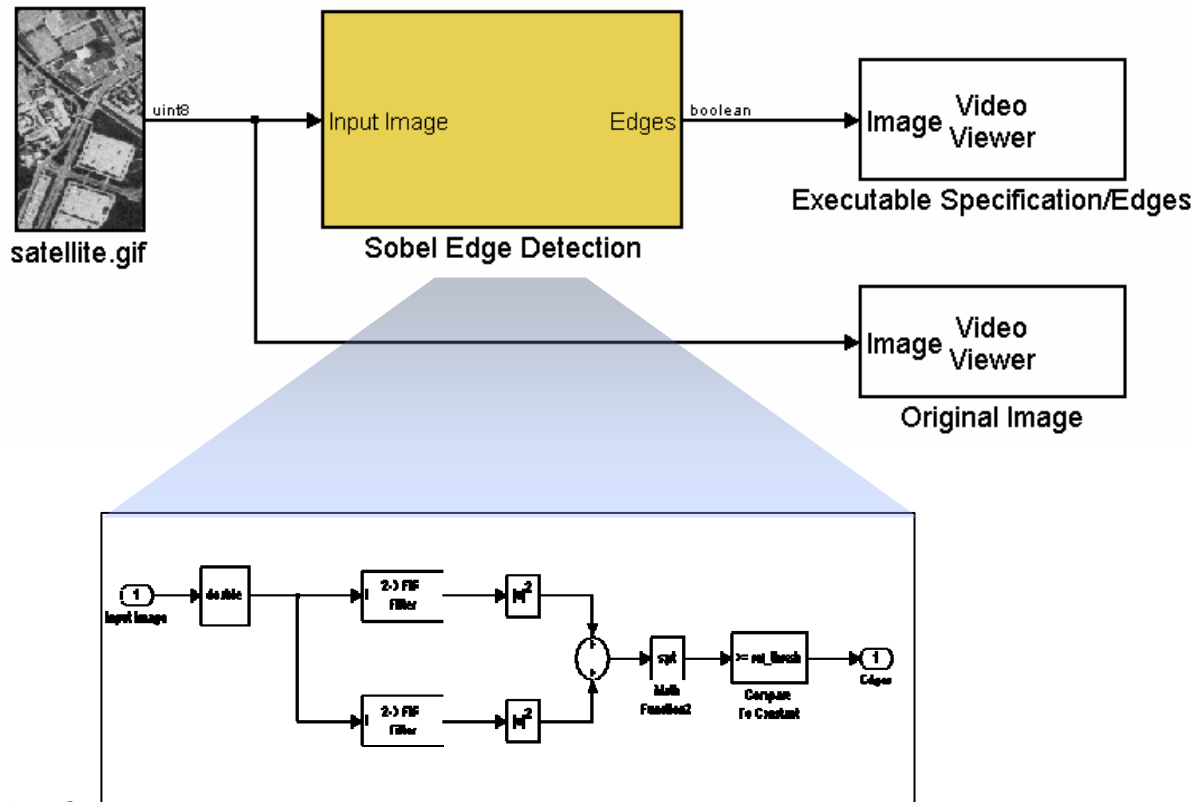
# Case Study: Sobel Edge Detection Algorithm



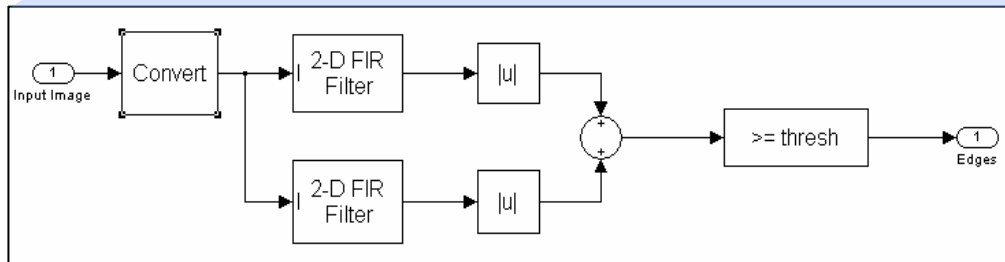
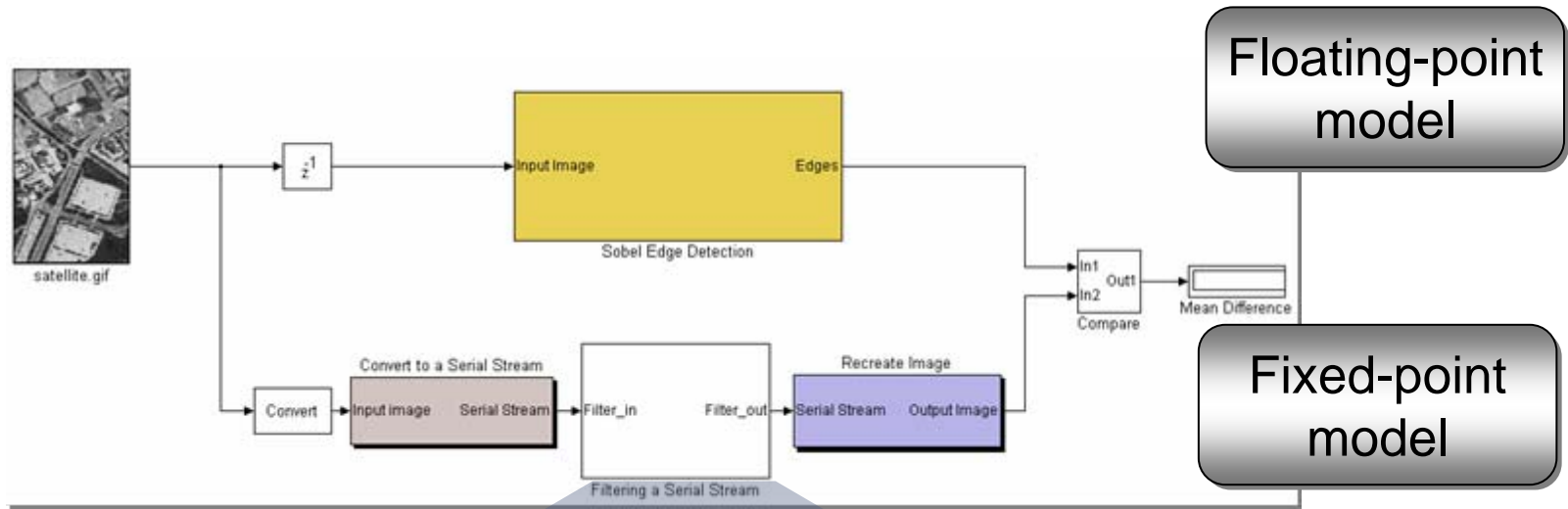
# Floating-Point System Specification

- Start by developing a golden specification

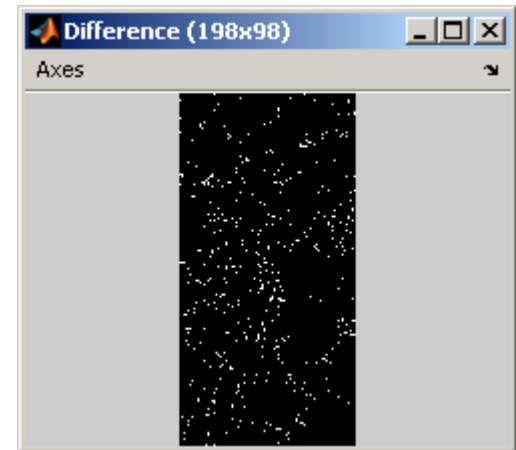
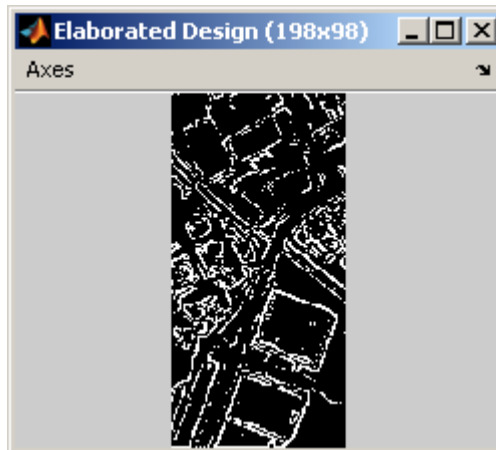
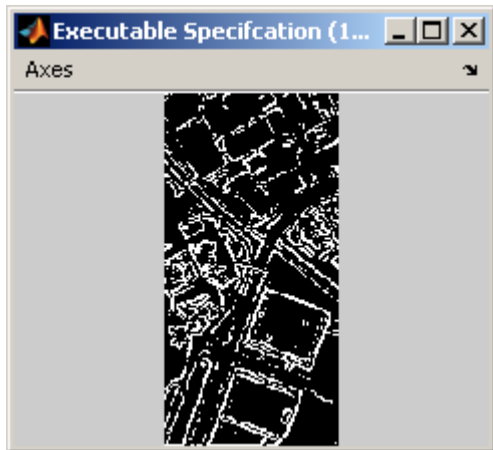
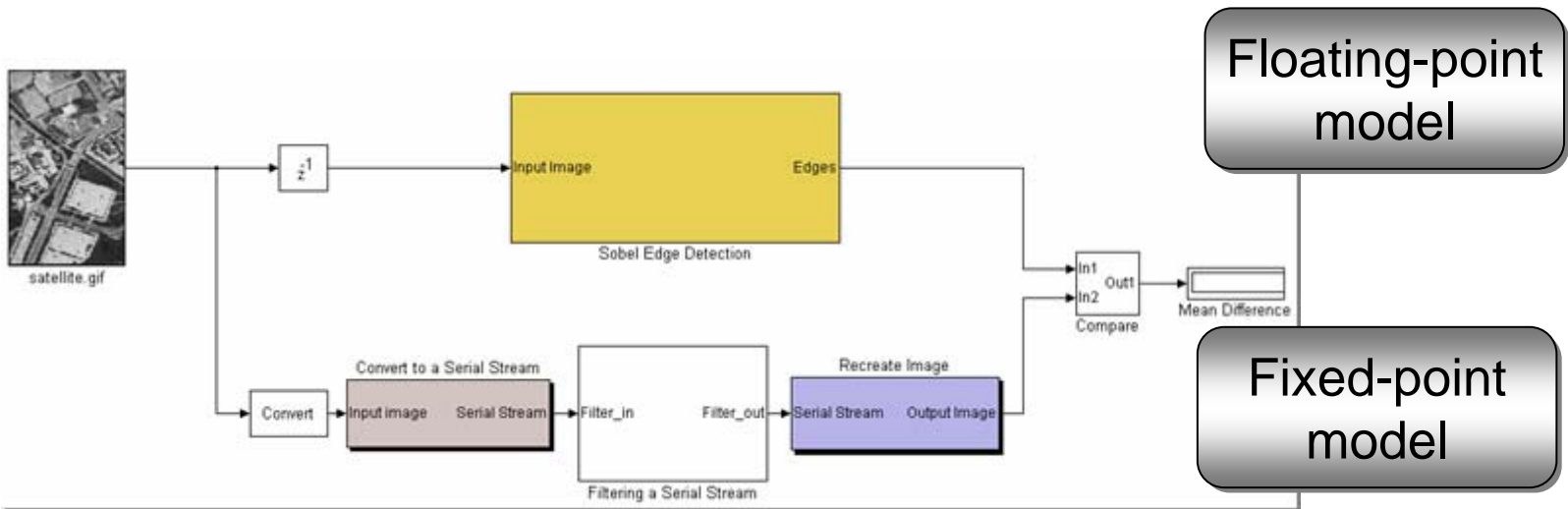
**1: Executable Specification / Golden Reference**



# Fixed-Point Modeling

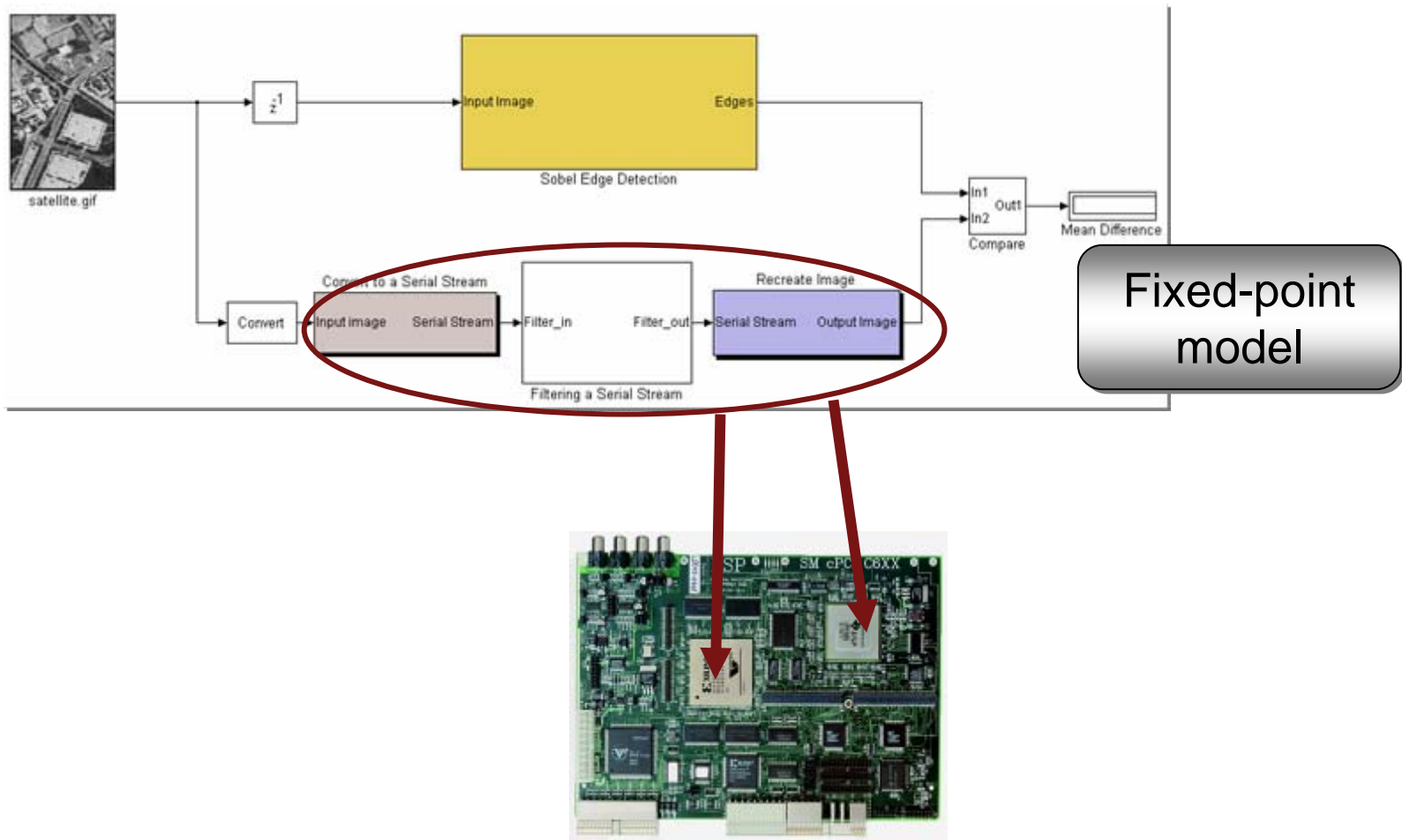


# Fixed-Point Modeling



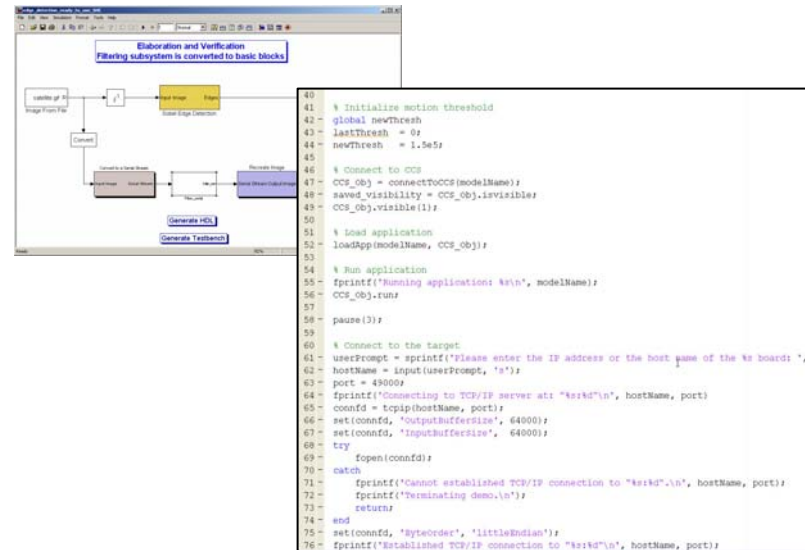


# Implementation on DSP, GPP, or an FPGA?



# Agenda

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

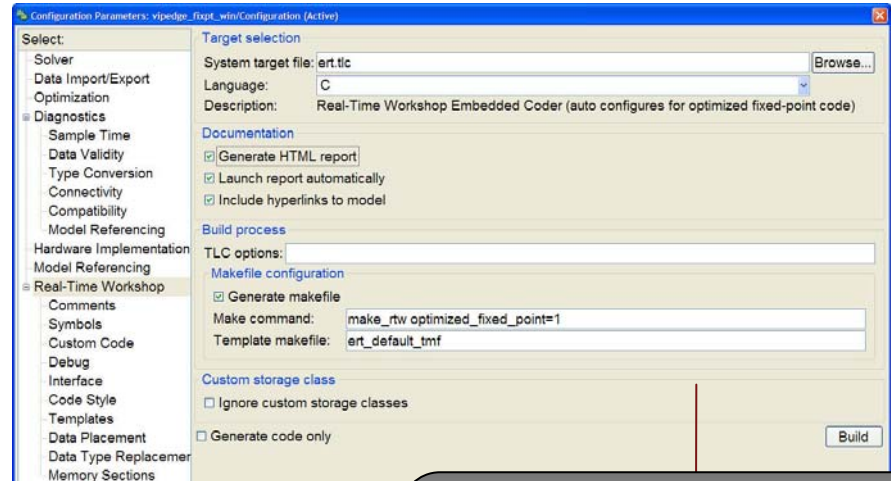
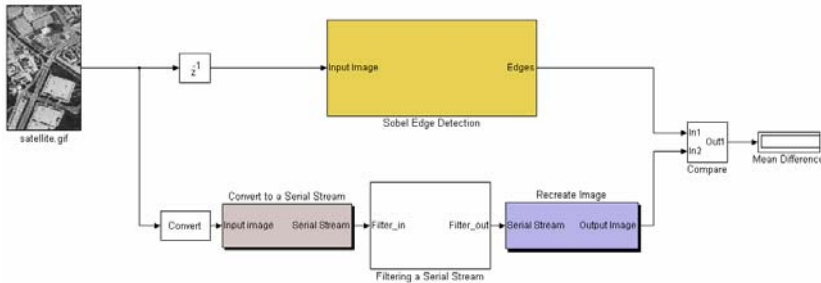


The screenshot displays a Simulink model window titled "Elaboration and Verification" with a subtitle "Filtering subsystem is converted to basic blocks". The model includes blocks for "Image Filter", "Edge Edge Detection", "Image Filter", "Image Filter", "Image Filter", and "Image Filter". Below the model are buttons for "Generate C/C++" and "Generate Testbench".

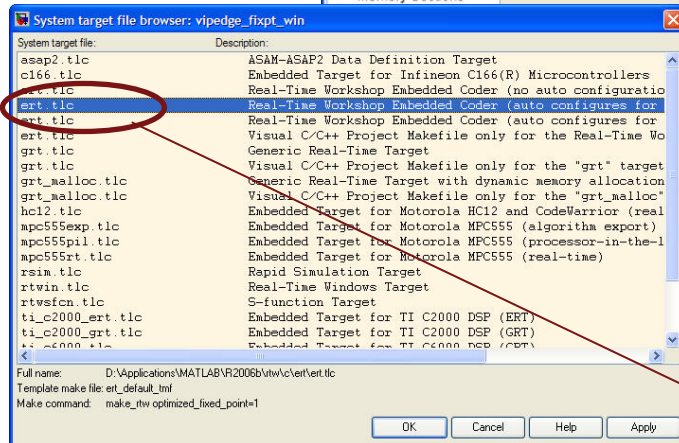
```

40
41 % Initialize motion threshold
42 global newThresh
43 lastThresh = 0;
44 newThresh = 1.5e5;
45
46 % Connect to CCS
47 CCS_obj = connectToCCS(modelName);
48 saved_visibility = CCS_obj.isvisible;
49 CCS_obj.visible(1);
50
51 % Load application
52 loadApp(modelName, CCS_obj);
53
54 % Run application
55 fprintf('Running application: %s\n', modelName);
56 CCS_obj.run;
57
58 pause(3);
59
60 % Connect to the target
61 userPrompt = sprintf('Please enter the IP address or the host name of the %s board: ');
62 hostName = input(userPrompt, 's');
63 port = 43000;
64 fprintf('Connecting to TCP/IP server at: %s\n', hostName, port);
65 connfd = tcpip(hostName, port);
66 set(connfd, 'OutputBufferSize', 64000);
67 set(connfd, 'InputBufferSize', 64000);
68 try
69     fopen(connfd);
70 catch
71     fprintf('Cannot established TCP/IP connection to %s\n', hostName, port);
72     fprintf('Terminating demo.\n');
73     return;
74 end
75 set(connfd, 'ByteOrder', 'littleEndian');
76 fprintf('Established TCP/IP connection to %s\n', hostName, port);
  
```

# Implementation on DSP and GPP



Fixed-point model

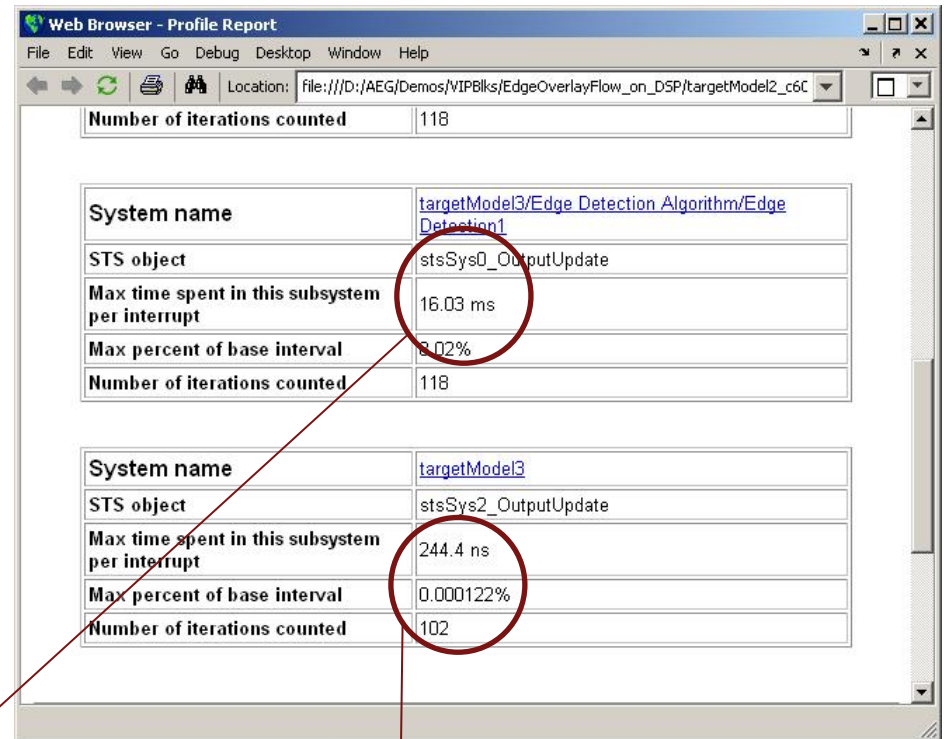


Code generation options and preferences

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



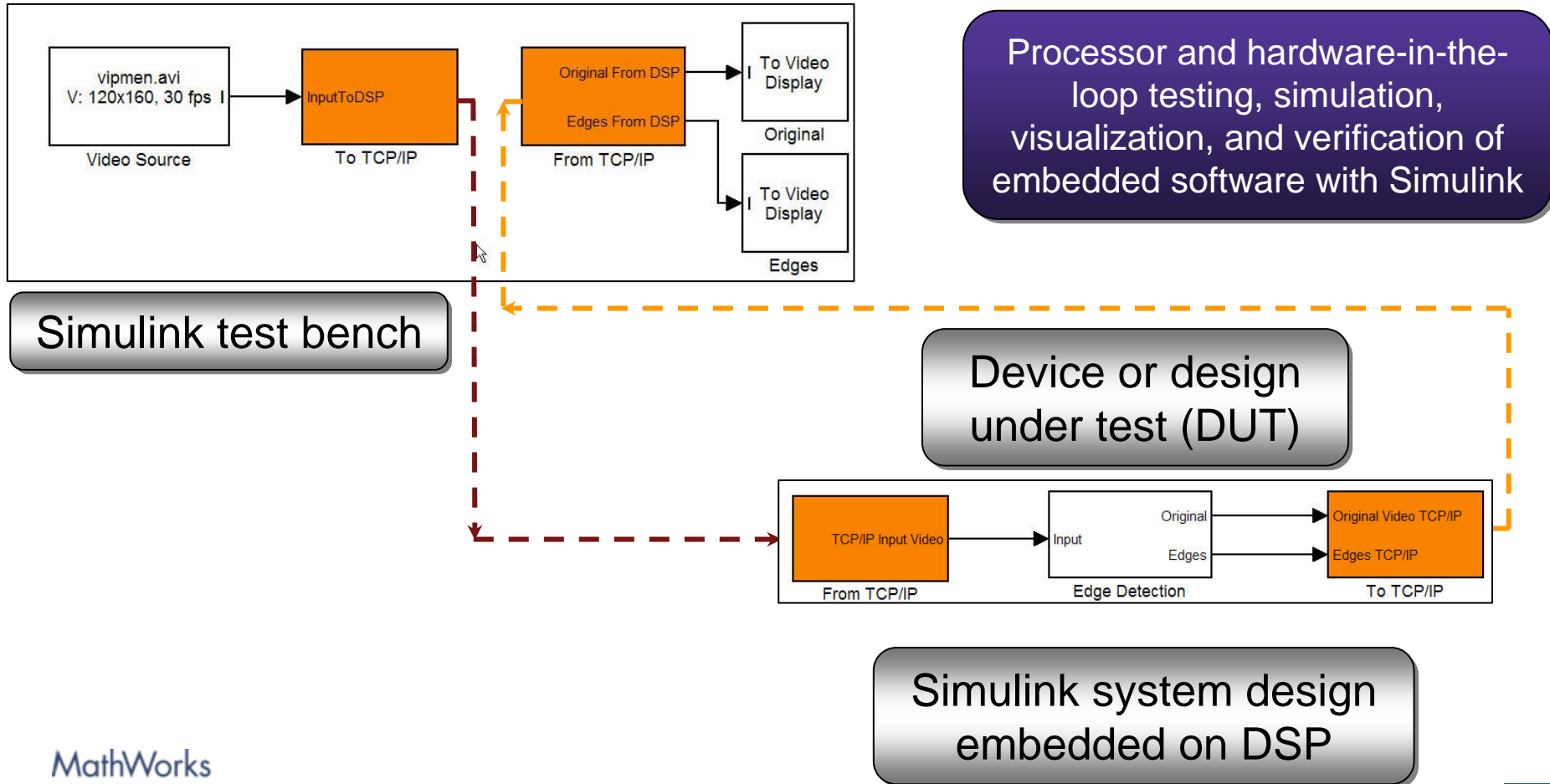
**System** profiling includes entire DSP application code

16.03 ms
8.02%
118

244.4 ns
0.000122%
102

**Subsystem** profiling

# Design Verification and Visualization: Simulink as verification test bench



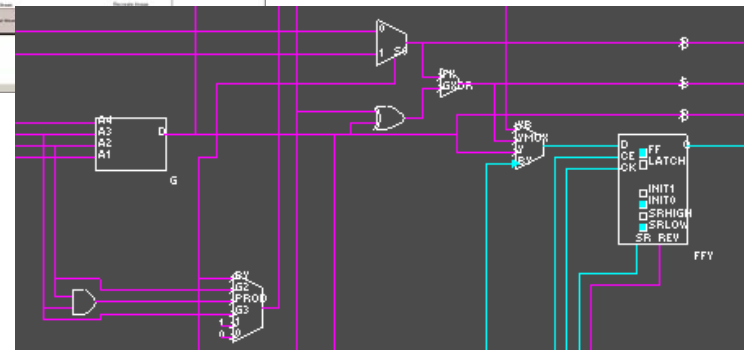
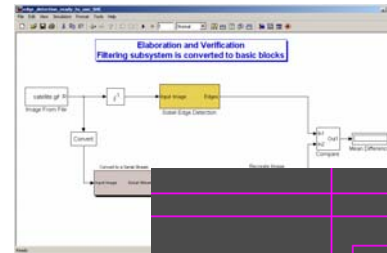
# Review: Code Generation for Embedded Software

- 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





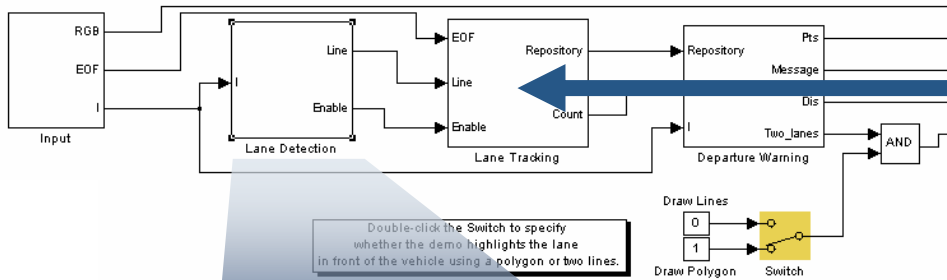
# Code Generation for Hardware

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

## Generated Verilog code

### Simulink data path

**Lane Departure Warning System**

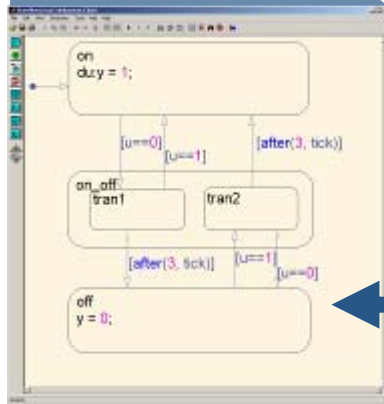


```

58 Timing_Controller u_Timing_Controller (.clk(clk), // boolean
59 .reset(reset), // boolean
60 .clk_enable(clk_enable), // boolean
61 .enb(enb) // boolean
62 );
63
64 assign ce_out = enb;
65 always @ ( posedge clk)
66 begin: Desired_reg_process
67 if (reset == 1'b1) begin
68 Desired_reg_out1 <= 0;
69 end
70 else begin
71 if (enb == 1'b1) begin
72 Desired_reg_out1 <= Desired;
73 end
74 end // Desired_reg_process
75
76 always @ ( posedge clk)
77 begin: Input_reg_process
78 if (reset == 1'b1) begin
79 Input_reg_out1 <= 0;
80 end
81 else begin
82 if (enb == 1'b1) begin
83 Input_reg_out1 <= Input;
84 end
85 end
86 end // Input_reg_process
87
88 assign mul_temp = Step_Size * Sum_out1;
89 assign Product_out1 = (((mul_temp[31]), mul_temp[31:7]) + 1)>>1;
90
91 assign Constant_out1 = 32'h00000000;
92
93 LMSx10_1 u_LMSx10_1 (.clk(clk), // boolean
94 .enb(enb), // boolean
95 .reset(reset), // boolean
96 .Data_In(Input_reg_out1), // sfix16_En13

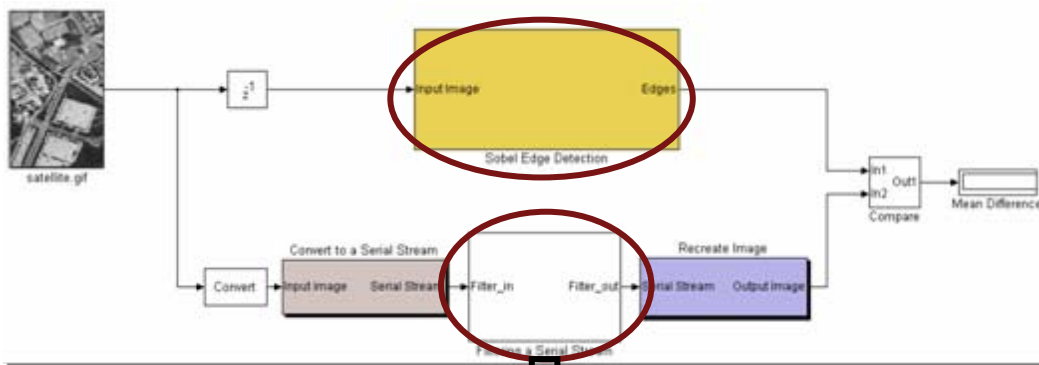
```

### Stateflow control logic

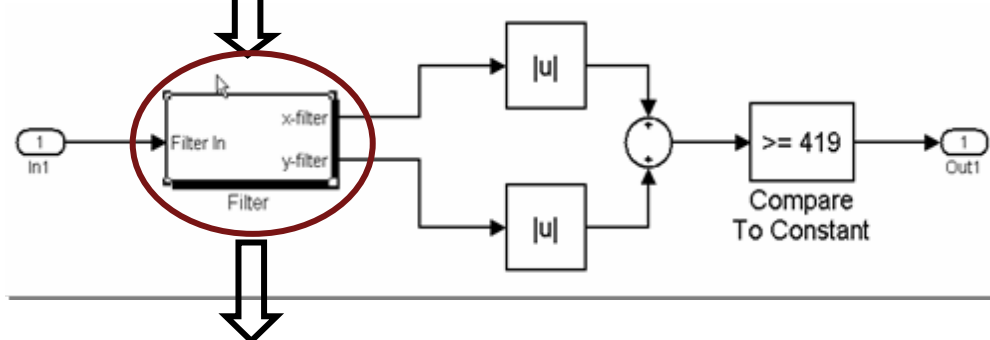




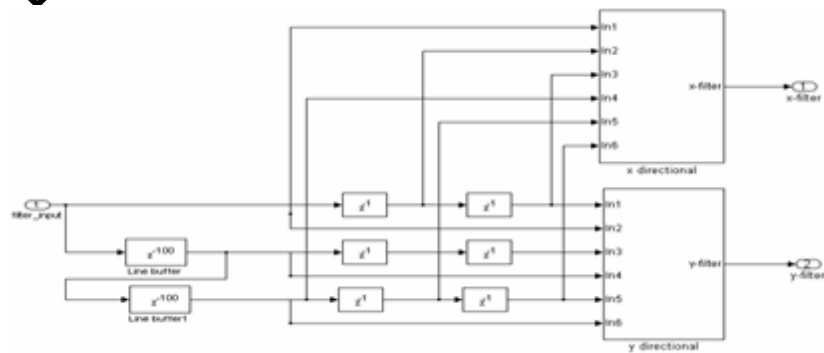
# Fixed-Point Implementation on an FPGA



Floating-point model



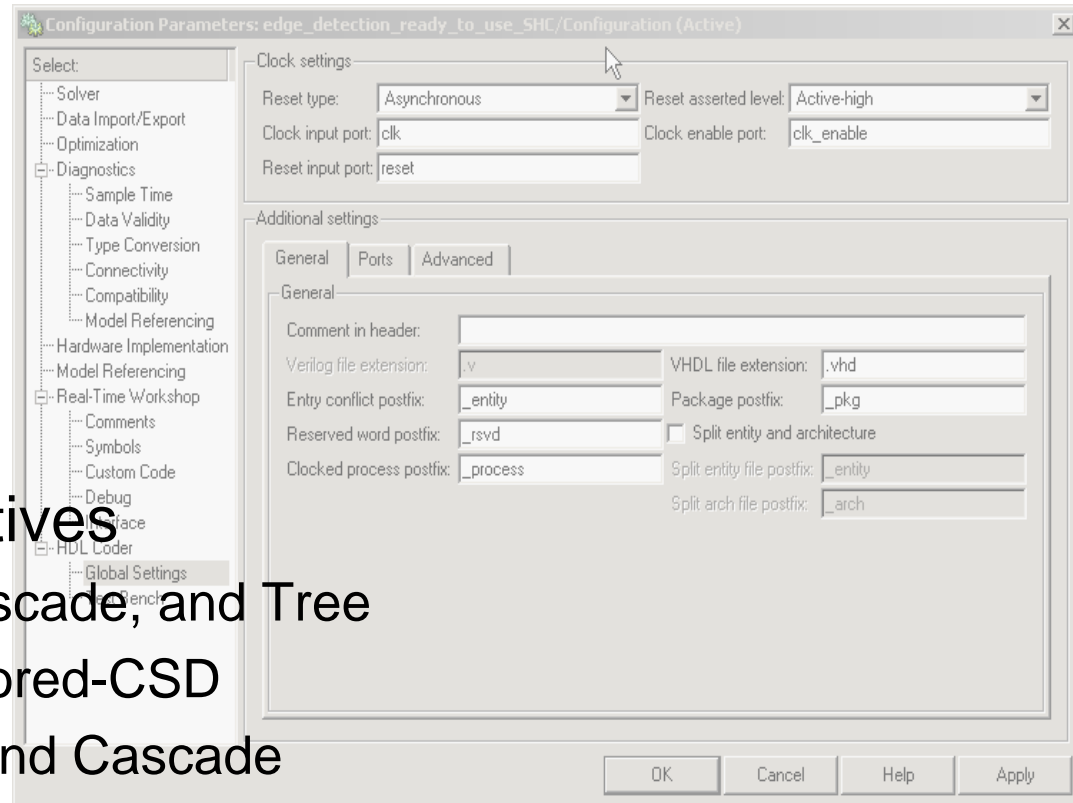
Fixed-point model



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

The screenshot shows the 'Configuration Parameters: edge\_detection\_ready\_to\_use\_SHC/Configuration (Active)' dialog box. The left sidebar lists various configuration categories, with 'HDL Coder' selected. The main area is divided into several sections:

- Code generation control file:** Includes 'File name:' with 'Load...' and 'Save...' buttons.
- Target:** Includes 'Generate HDL for:' (set to 'edge\_detection\_ready\_to\_use\_SHC/Filter\_serial'), 'Language:' (set to 'vhdl'), and 'Directory:' (set to 'hdlsrc').
- Code generation output:** Includes three radio button options: 'Generate HDL code' (selected), 'Display generated model only', and 'Generate HDL and display generated model'.
- Buttons:** 'Restore Factory Defaults', 'Run Compatibility Checker', and 'Generate'.
- Footer:** 'OK', 'Cancel', 'Help', and 'Apply' buttons.

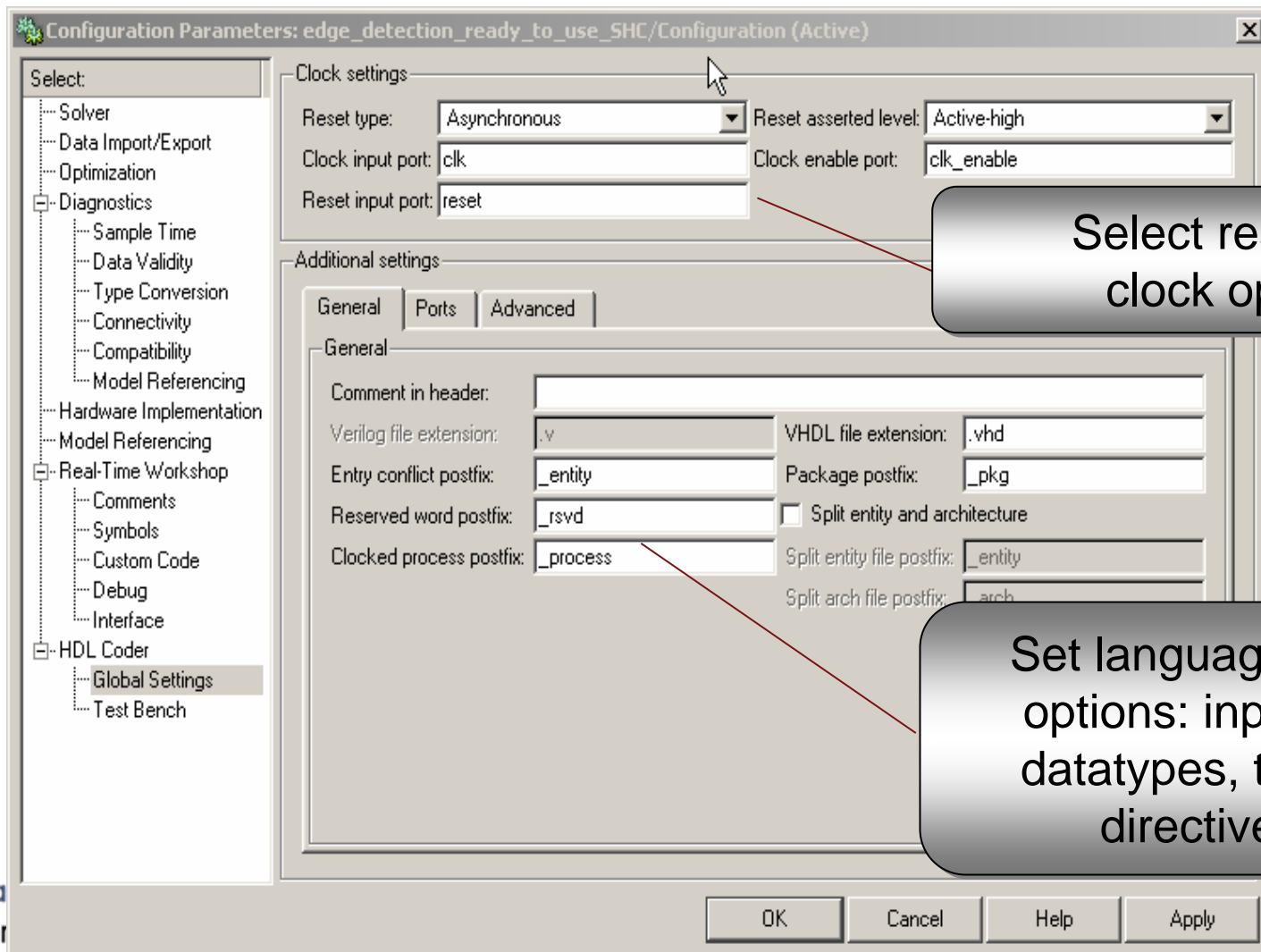
Select subsystem, target language, directory

Select output options

Check model for errors

Generate HDL Code

# More Code Generation Options



Select reset and clock options

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

# Generate HDL Test Bench

**Configuration Parameters: hdlcoderlms/Configuration (Active)**

Select:

- Solver
- Data Import/Export
- Optimization
- Diagnosics
  - 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**

Test bench

Test bench name postfix:

Force clock

Clock high time (ns):

Clock low time (ns):

Force clock enable

Force reset

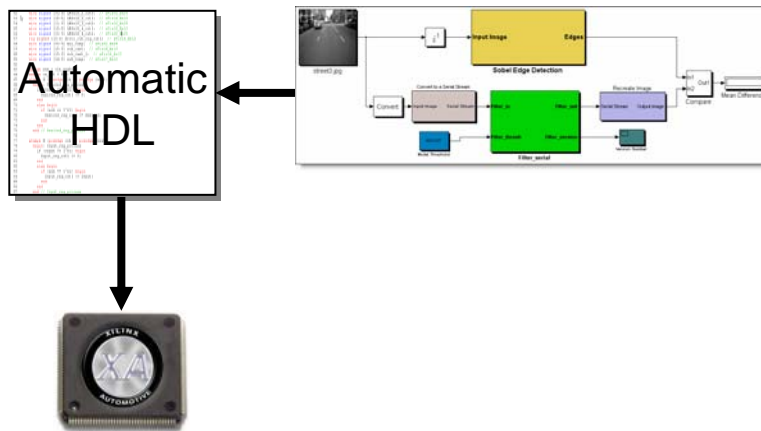
Hold time (ns):

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

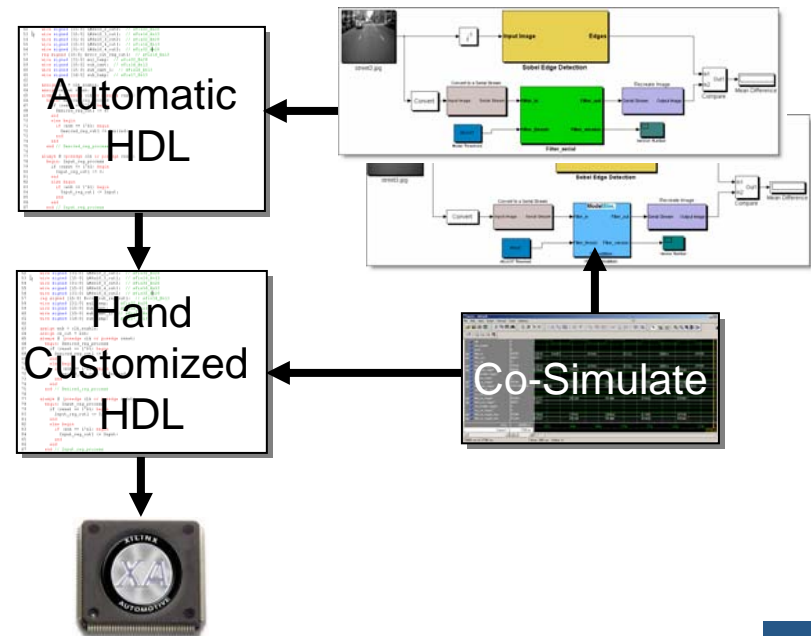
# Automatic HDL Code Generation

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

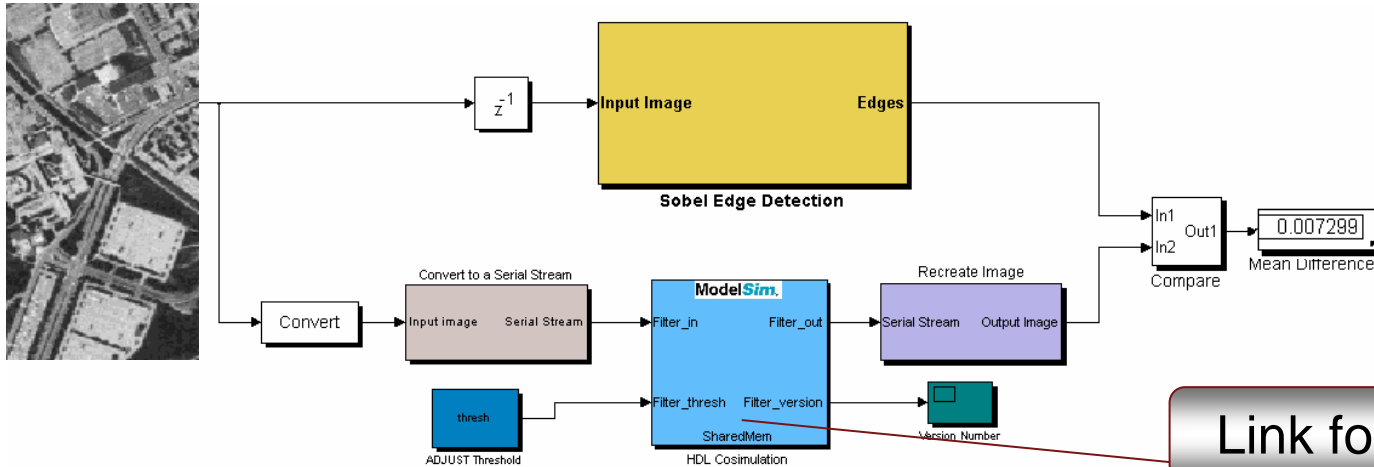
## 1 Rapid FPGA implementation



## 2 Reference code for HDL engineers



# Verification with system specification

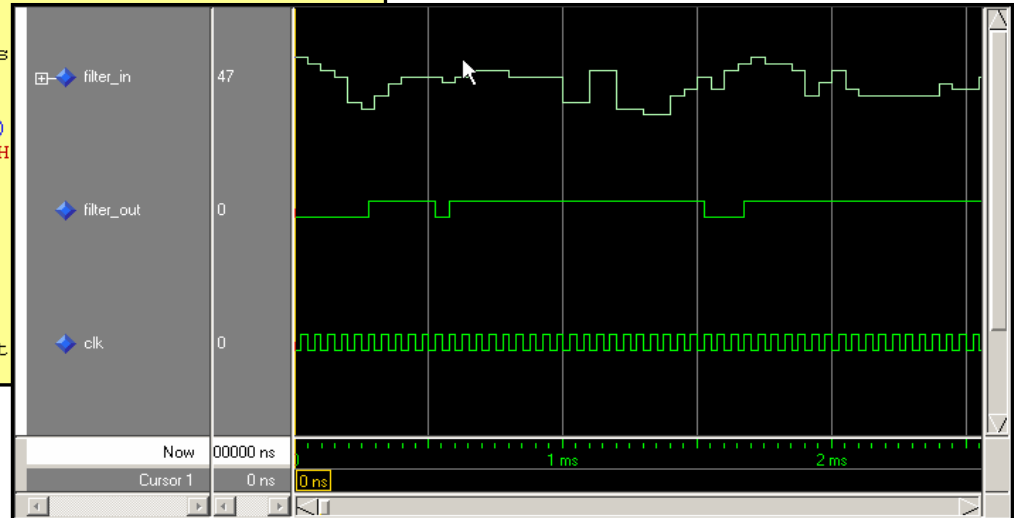


Link for ModelSim

```

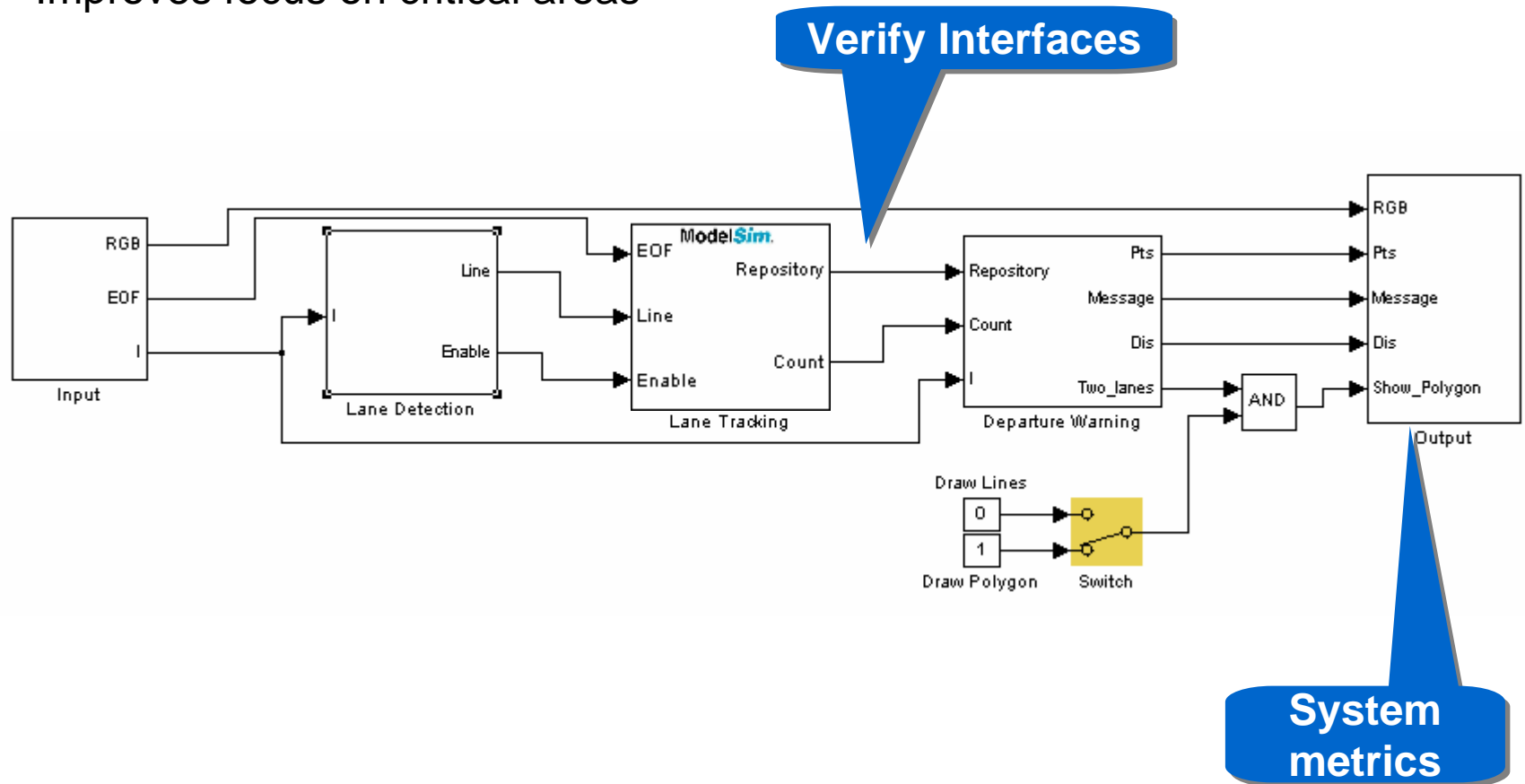
117     y_filter => y_directional_out1 -- ufix11
118   );
119   s <= signed(filter_input);
120
121   Delay2_process : PROCESS (clk, res
122 BEGIN
123     IF reset = '1' THEN
124       Delay2_out1 <= (OTHERS => '0');
125     ELSIF clk'event AND clk = '1' TH
126       IF enb = '1' THEN
127         Delay2_out1 <= 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

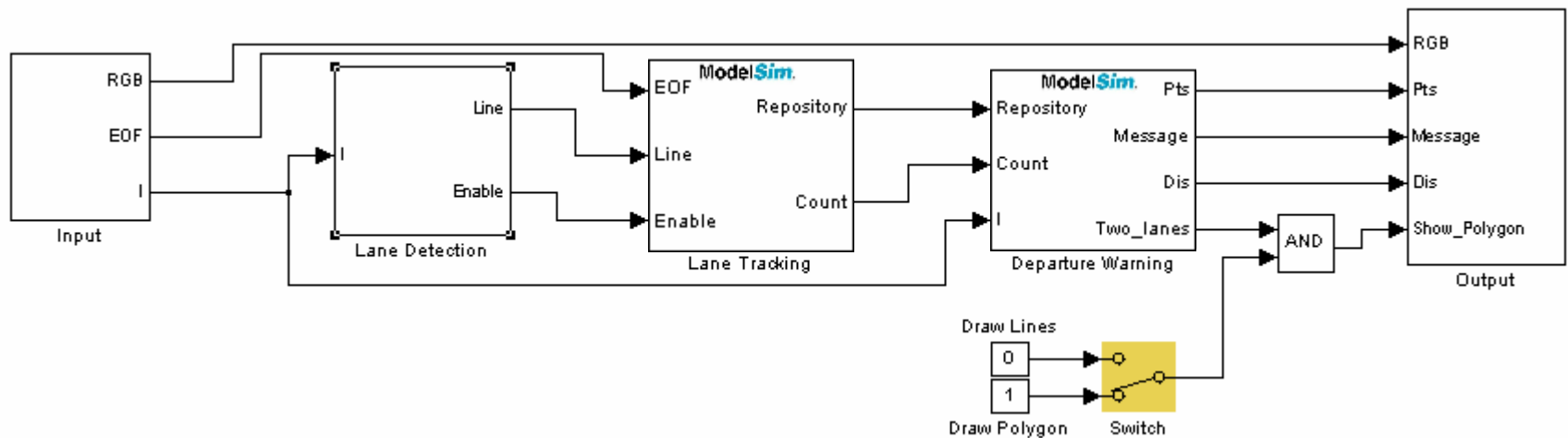
- Promotes parallelism in design and verification tasks
- Improves focus on critical areas





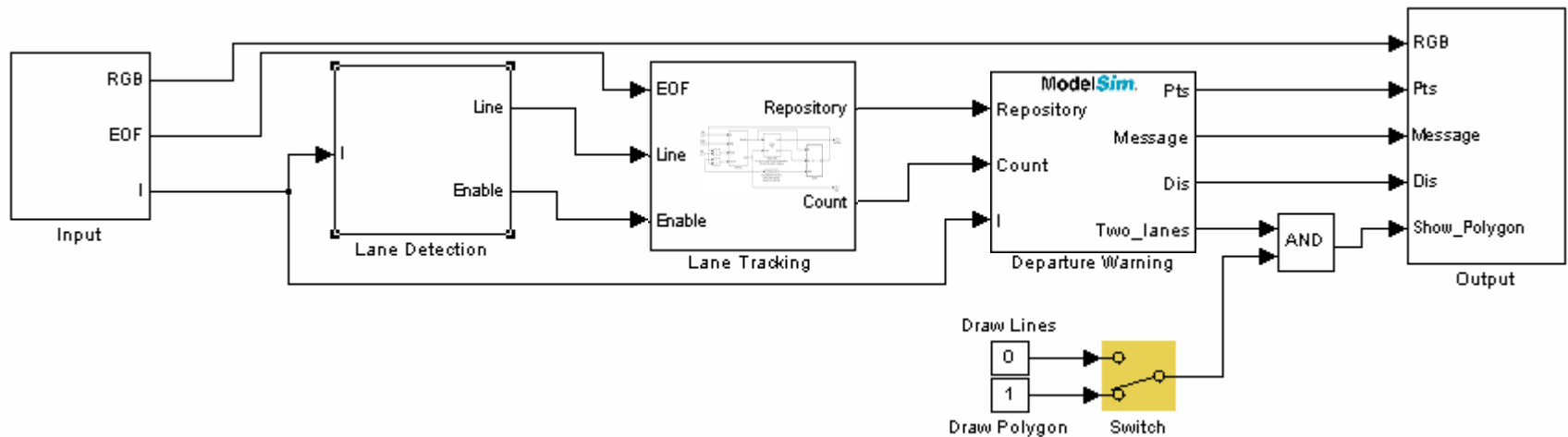
# Making full use of the system model

- 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

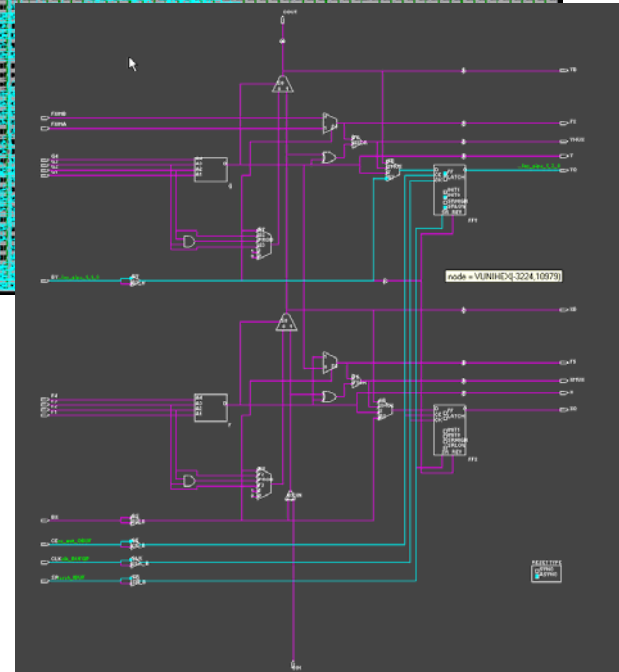
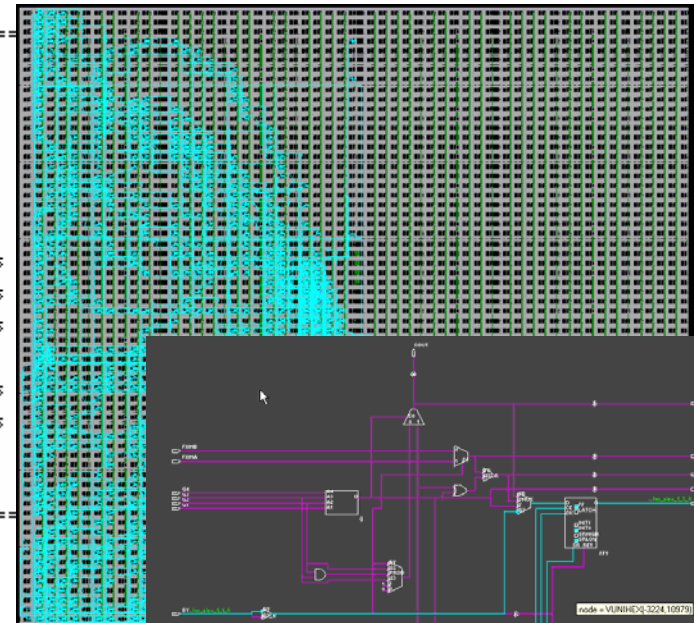
-----  
 Device utilization summary:  
 -----

➔ 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 software and hardware from MATLAB and Simulink
- Accelerate product development using Model-Based Design

