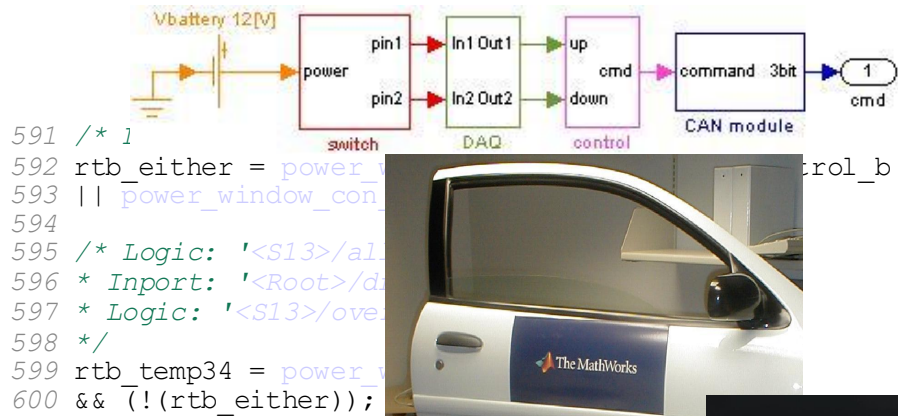
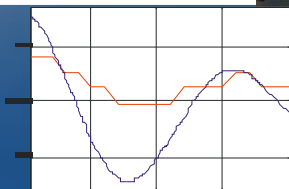


# MATLAB<sup>®</sup> and Simulink<sup>®</sup> for Embedded System Design



Pieter J. Mosterman  
 pieter.mosterman@mathworks.com

Senior Research Scientist  
 The MathWorks, Inc.



# Introduction

- **Increasing complexity of embedded systems**
- **Complexity**
  - **Intricacy**
  - **Size**
- **Raising the Level of Abstraction**
- **Compilers to handle the complexity because of size**

# Agenda

- **The System Design Challenge**
- **Software Design Flow**
- **Hardware Design Flow**
- **Summary**

# The System Design Challenge

- We design, simulate, and validate system models and algorithms in MATLAB® and/or Simulink®
- How can we implement and verify designs on DSPs and GPPs?
- How can we implement and verify designs on FPGAs and ASICs?

**MATLAB® and Simulink®**  
Algorithm and System Design



**C**

**MCU**

**DSP**

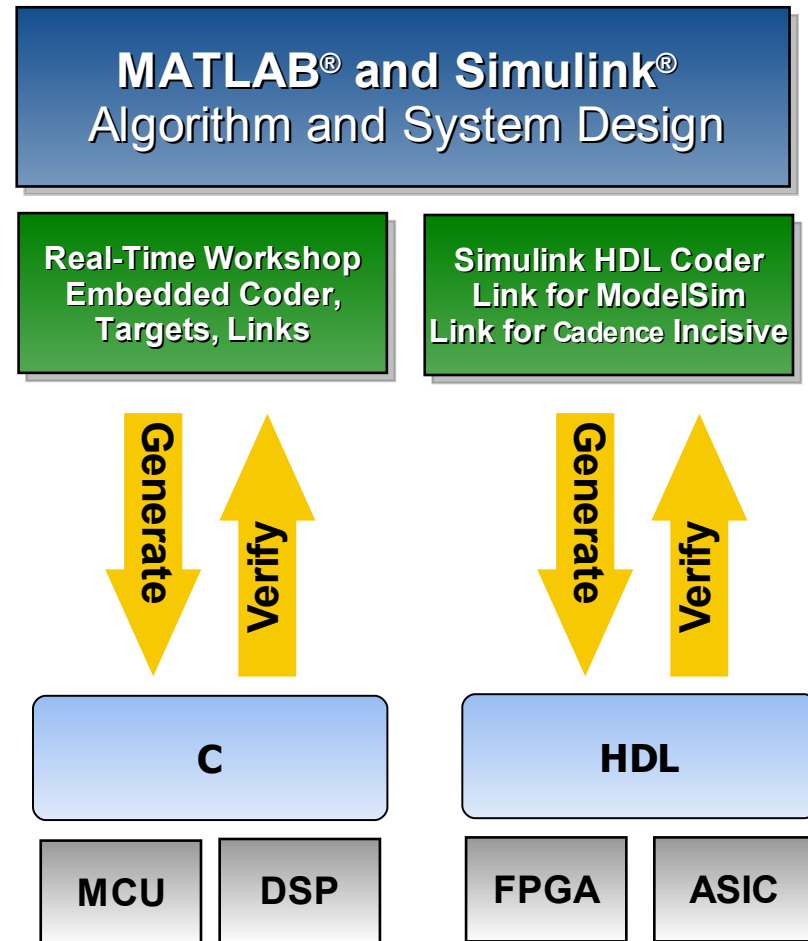
**HDL**

**FPGA**

**ASIC**

# Integrated Design Flow for Embedded Software and Hardware

- Design, simulate, and validate system models and algorithms in MATLAB and Simulink
- Automatically generate C and HDL
- Verify hardware and software implementations against the system and algorithm models

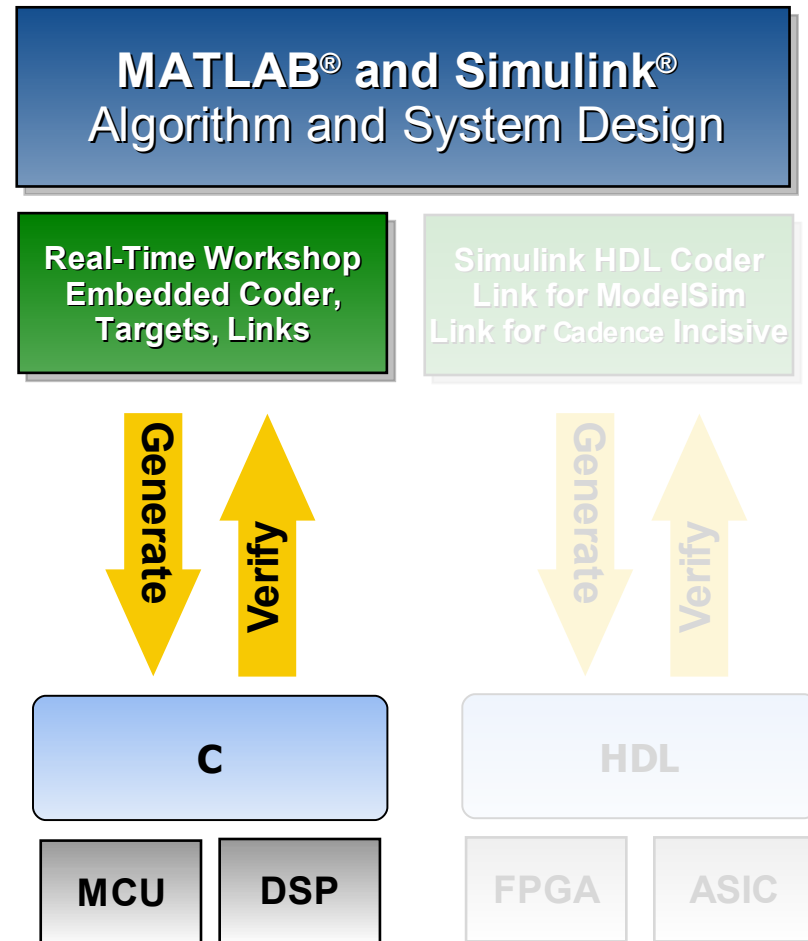


# Agenda

- **The System Design Challenge**
- **Software Design Flow**
- **Hardware Design Flow**
- **Summary**

# Integrated Design Flow for Embedded Software

- Implementation with automatic C code generation
- Implementation
  - Floating- and fixed-point code
  - Integration with downstream IDEs and tools
  - Links to verification
  - Device drivers
  - Optimization options



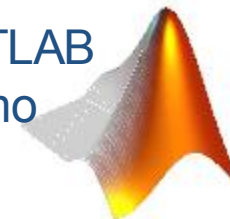
# Representative Application – Video

- Video system design and implementation
  - Encapsulates challenges: complexity, convergence, time-to-market
  - Sophisticated algorithms
  - Floating- and fixed-point issues
  - DSP or FPGA/ASIC implementations
  
- Design flows and steps shown directly applicable to other signal processing applications

Embedded Software  
Case Study:  
Video Edge Detection  
on a DSP



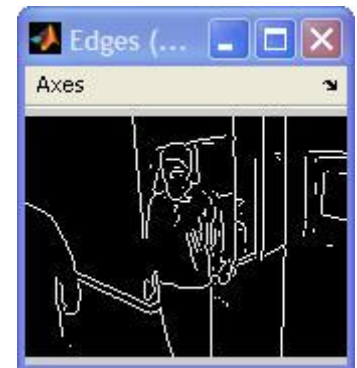
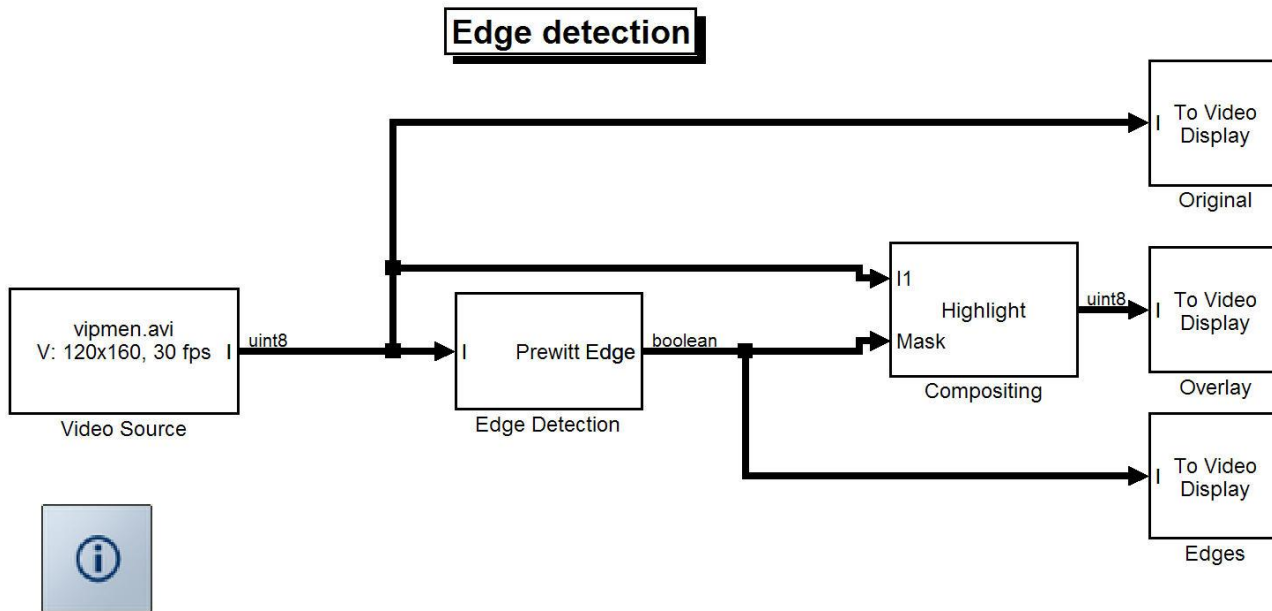
Live  
MATLAB  
Demo





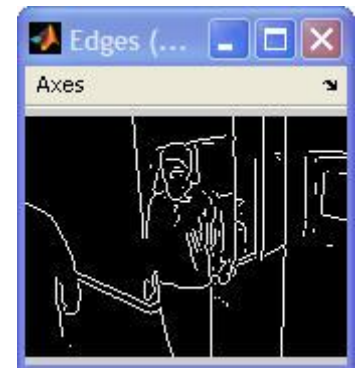
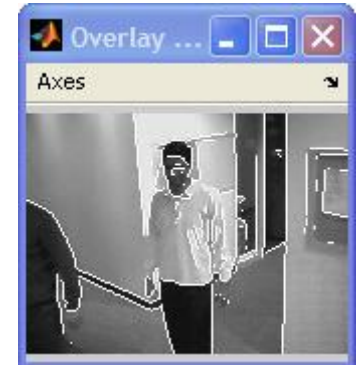
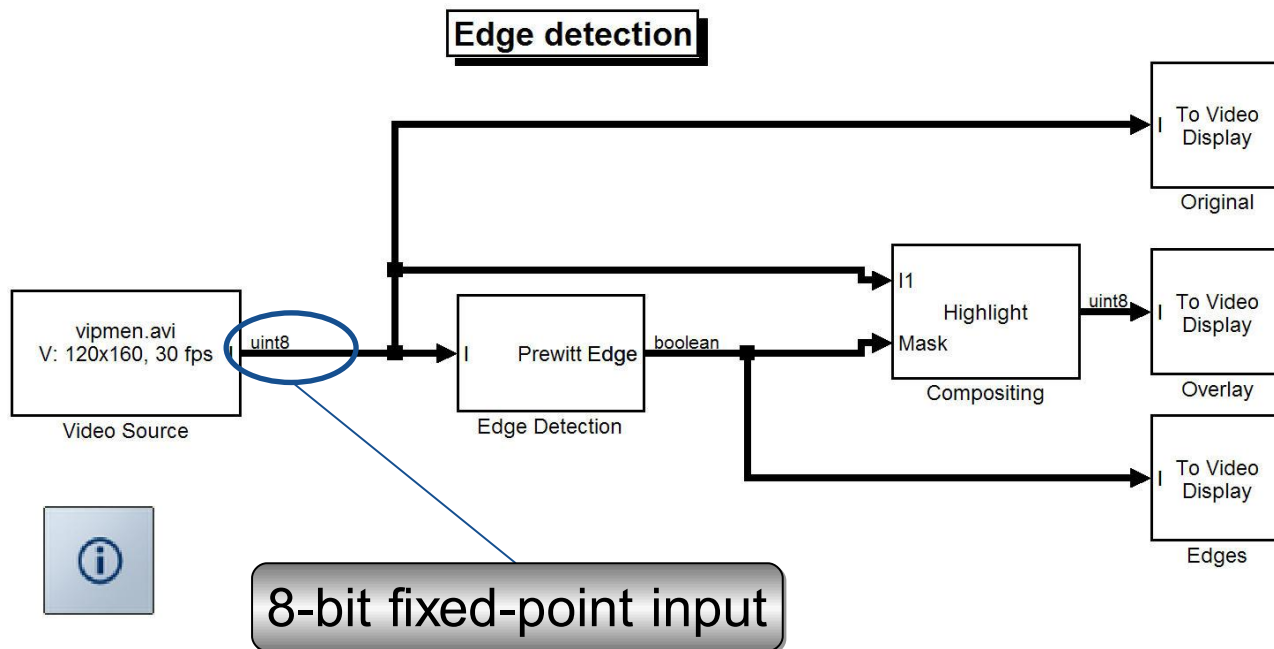
# Example: Video Edge Detection

- Floating point video edge detection system based on Prewitt algorithm
- Compositing original image with detected edges
- Utilizes blocks from Video and Image Processing Blockset



# Converting to Fixed-Point

- Polymorphic blocks capable of floating- and fixed-point operation
- 8-bit input datatype – blocks inherit fixed-point data
- Simulink Accelerator provides fast fixed-point simulation



# Automatic Code Generation for Implementation on GPPs and DSPs

The screenshot displays the MATLAB/Simulink environment. On the left, a Simulink model titled 'Edge Detection' is shown. It includes a 'Video Source' block (vipmen.avi, 120x160, 30 fps), an 'Edge Detection' block (Prewitt Edge), and a 'Highlight Mask' block (Compositing). The output of the 'Edge Detection' block is connected to three 'To Video Display' blocks, labeled 'Original', 'Overlay', and 'Edges'. On the right, the 'Configuration Parameters' dialog box is open, showing various settings for code generation. The 'Target selection' section is active, showing the system target file as 'ert.tlc' and the language as 'C'. The 'Real-Time Workshop' section is also visible, with options for generating HTML reports and launch reports.

Fixed-point model

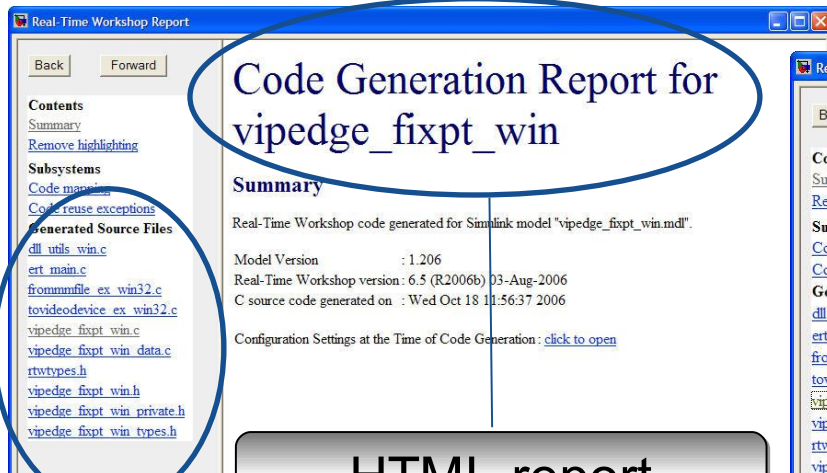
The screenshot shows the 'System target file browser' dialog box. It lists various system target files (SLC files) and their descriptions. The 'ert.tlc' file is highlighted, indicating it is the selected target. The dialog box also shows the full name, template make file, and make command for the selected target.

System target file:	Description:
asap2.tlc	ASAM-ASAP2 Data Definition Target
c166.tlc	Embedded Target for Infineon C166(R) Microcontrollers
ert.tlc	Real-Time Workshop Embedded Coder (no auto configuration)
ert.tlc	Real-Time Workshop Embedded Coder (auto configures for)
ert.tlc	Real-Time Workshop Embedded Coder (auto configures for)
ert.tlc	Generic Real-Time Target
grt.tlc	Visual C/C++ Project Makefile only for the Real-Time Wo
grt_malloc.tlc	Generic Real-Time Target with dynamic memory allocation
grt_malloc.tlc	Visual C/C++ Project Makefile only for the "grt_malloc"
hc12.tlc	Embedded Target for Motorola HC12 and CodeWarrior (real
mpc555exp.tlc	Embedded Target for Motorola MPC555 (algorithm export)
mpc555pil.tlc	Embedded Target for Motorola MPC555 (processor-in-the-l
mpc555rt.tlc	Embedded Target for Motorola MPC555 (real-time)
rsim.tlc	Rapid Simulation Target
rtwin.tlc	Real-Time Windows Target
rtwfcn.tlc	S-function Target
ti_c2000_ext.tlc	Embedded Target for TI C2000 DSP (ERT)
ti_c2000_grt.tlc	Embedded Target for TI C2000 DSP (GRT)
ti_c2000_tlc	Embedded Target for TI C2000 DSP (GRT)

Code generation options and preferences

Select target or flavor of generated code

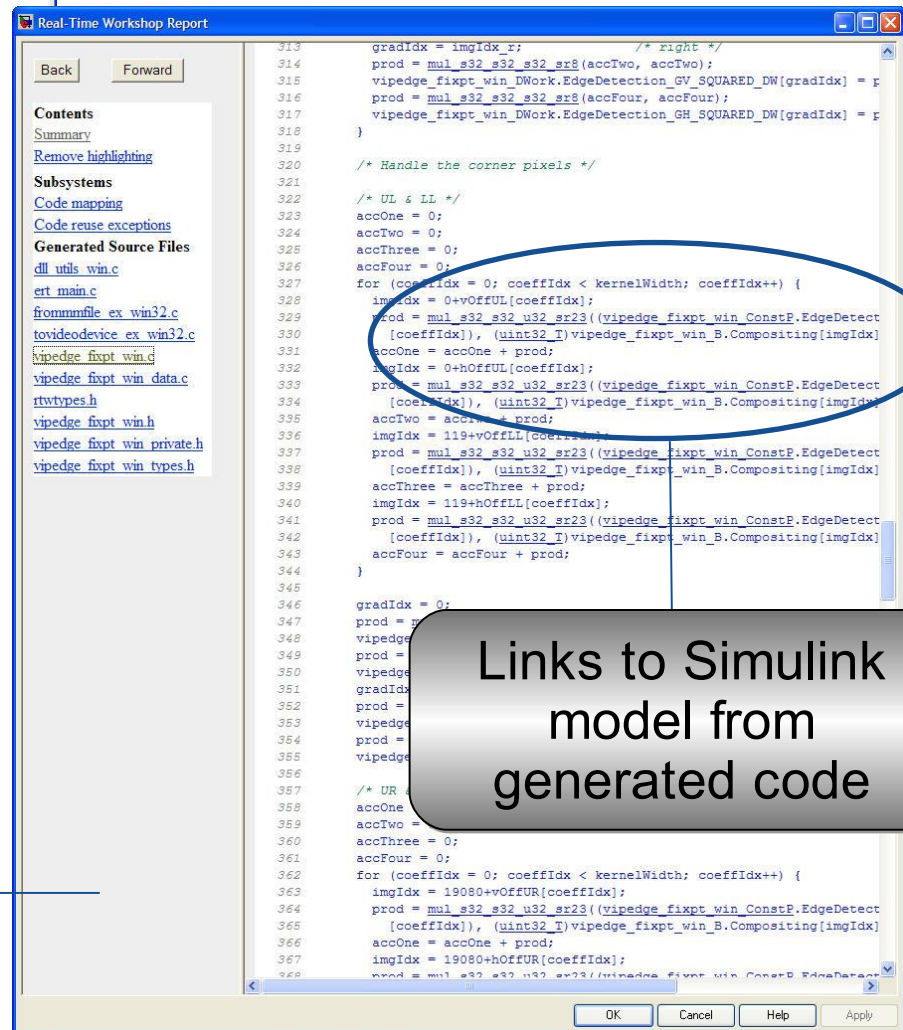
# Code Generation Report



HTML report

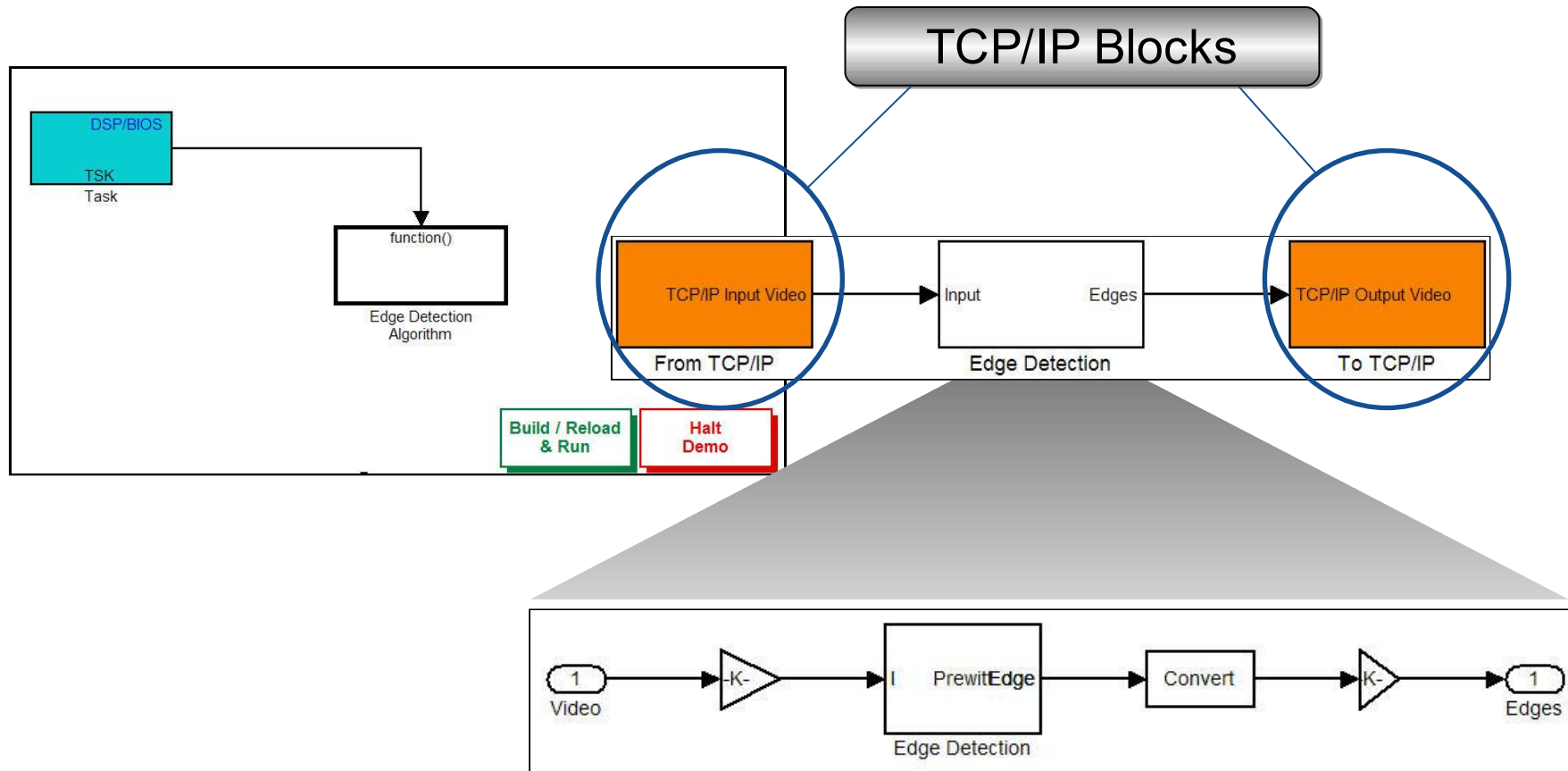
Links to code files from HTML report

Readable, commented code



Links to Simulink model from generated code

# Video Edge Detection Embedded Software System on TI 6000™



# Target Code Generation Options

The image displays two overlapping screenshots of the 'Configuration Parameters: c6416dsksurveil\_hsrtdx/Configuration' dialog box. The top screenshot shows the 'Code generation target type' set to 'C6416DSK'. Under the 'Diagnostics' section, the 'Enable High-Speed RTDX' checkbox is checked and circled in blue. Below it, the 'Export CCS handle to MATLAB base workspace' checkbox is also checked. The bottom screenshot shows the 'Optimization' section with 'Incorporate DSP/BIOS' checked and circled in blue. Below that, 'Inline run-time library functions' is checked and circled in blue. At the bottom of the 'Optimization' section, 'Use target-specific optimization for speed (allow LSB differences)' is checked. Three callout boxes with blue lines pointing to the circled options contain the following text: 'High-Speed RTDX', 'Incorporate DSP/BIOS', and 'Inline Signal Processing Blockset functions'. The dialog box also features a 'Select:' tree on the left and 'OK', 'Cancel', 'Help', and 'Apply' buttons at the bottom.

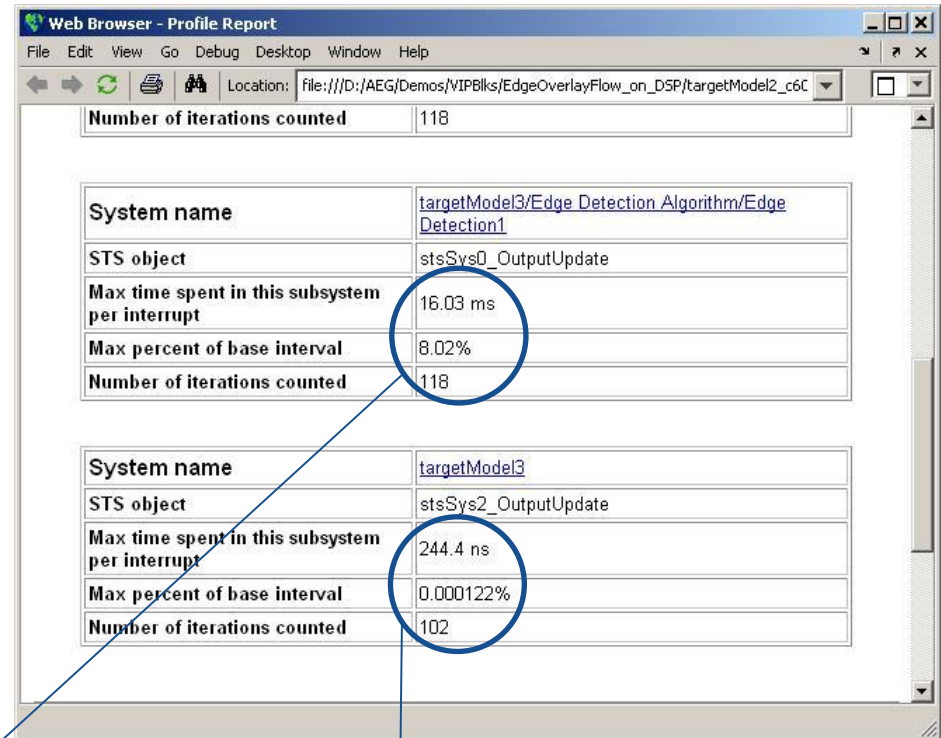
High-Speed RTDX

Incorporate DSP/BIOS

Inline Signal Processing Blockset functions

# 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

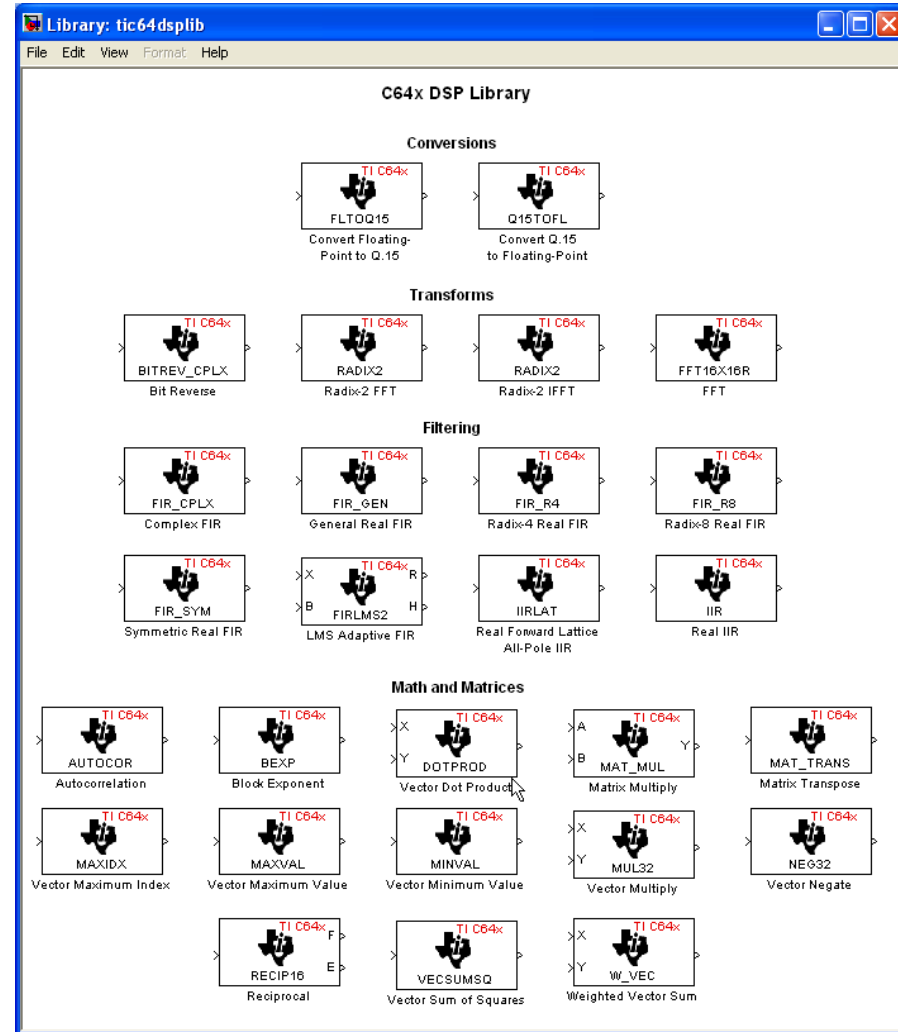
16.03 ms
8.02%
118

244.4 ns
0.000122%
102

**Subsystem** profiling

# Code Optimization Options

- Utilize target-specific blocks
  - C-callable assembler libraries
  - Simulate bit-true in Simulink
  - Generate calls to hand-optimized assembler libraries
  - Highly optimized implementation of core functionality
  - C62x and C64x fixed-point DSPs
  
- Manual optimization by user



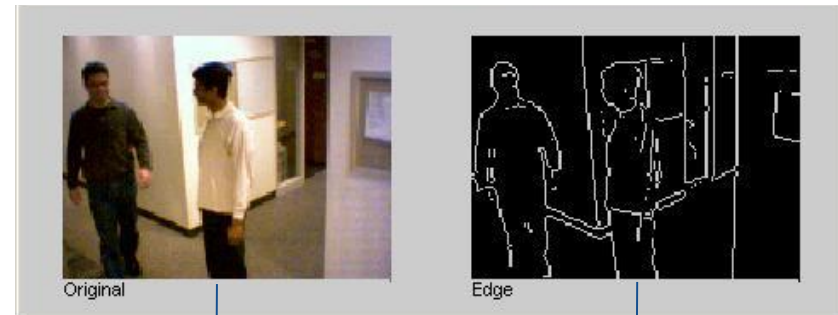


# Design Verification and Visualization: MATLAB as software test bench

```

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 = 49000;
64 fprintf('Connecting to TCP/IP server at: "%s:%d"\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:%d".\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:%d"\n', hostName, port);
    
```

Visualize and debug embedded software with MATLAB

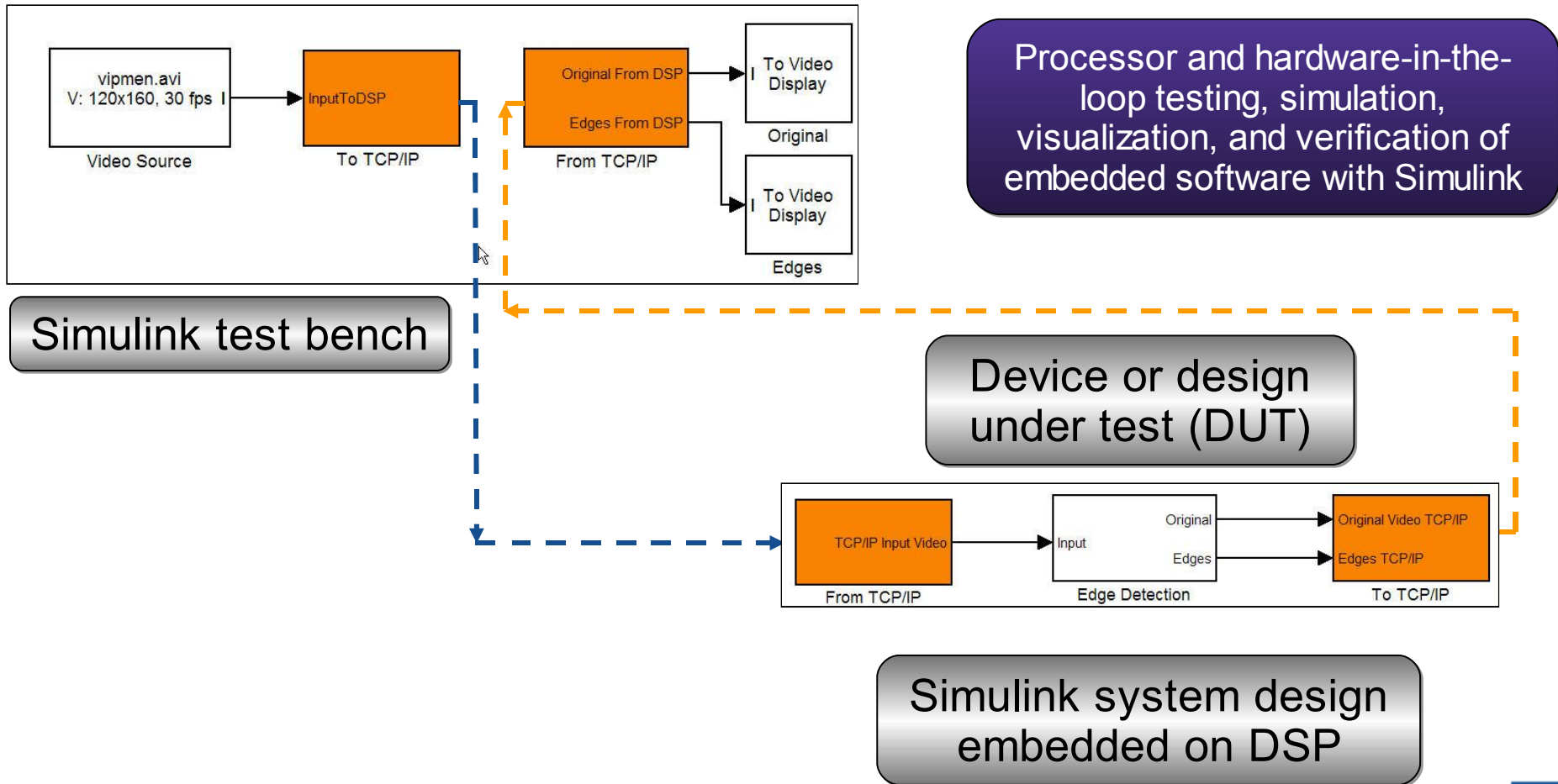


Input video

Captured video

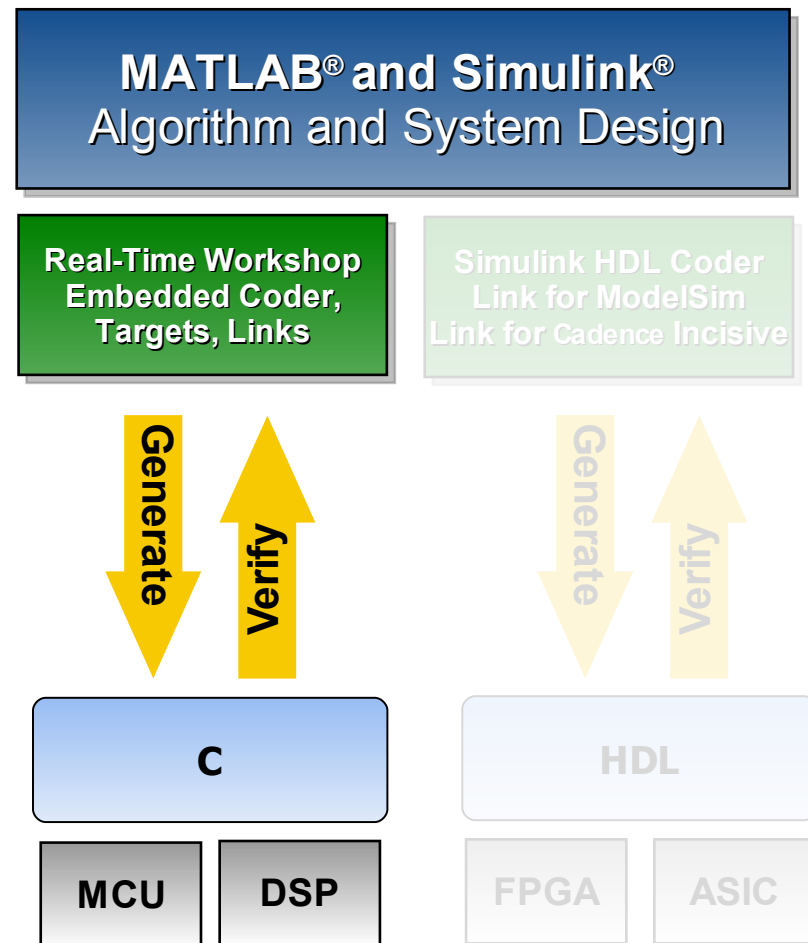
MATLAB script  
(test bench)

# Design Verification and Visualization: Simulink as software test bench



# Review: Integrated Design Flow for Embedded Software

- Drive system development with an executable specification
- Quickly create complete working code base
- Use code profiles to identify and optimize bottlenecks
- Verify code with Links to IDEs and processors

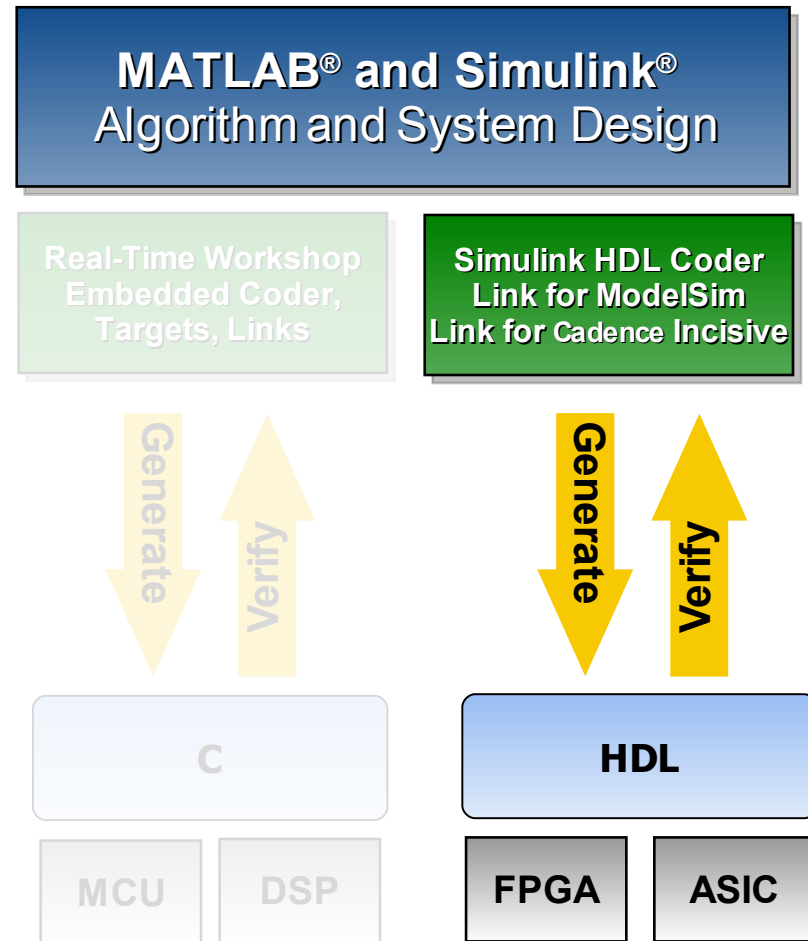


# Agenda

- **The System Design Challenge**
- **Software Design Flow**
- **Hardware Design Flow**
- **Summary**

# Integrated Design Flow for Hardware (FPGA and ASIC)

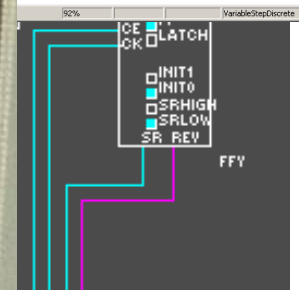
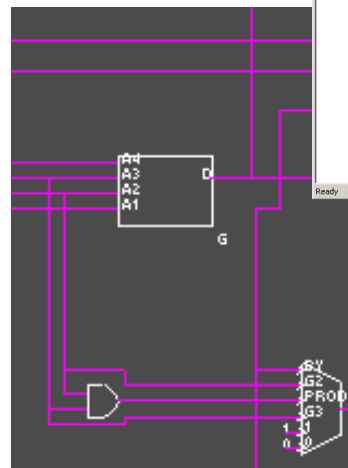
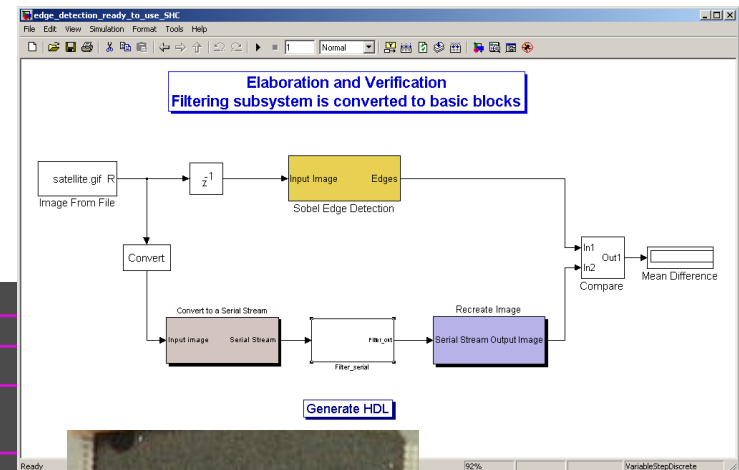
- Design elaboration
  
- Implementation
  - HDL code generation (VHDL and Verilog)
  - test bench generation
  - Links to verification
  - Integration with synthesis tools



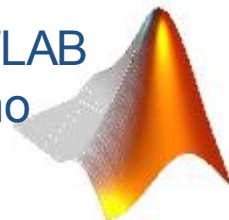
# What We Will Show In This Case Study

- Behavioral modeling and simulation
- Fixed-point modeling and simulation
- Design elaboration
- HDL generation
- Co-simulation using Link for ModelSim

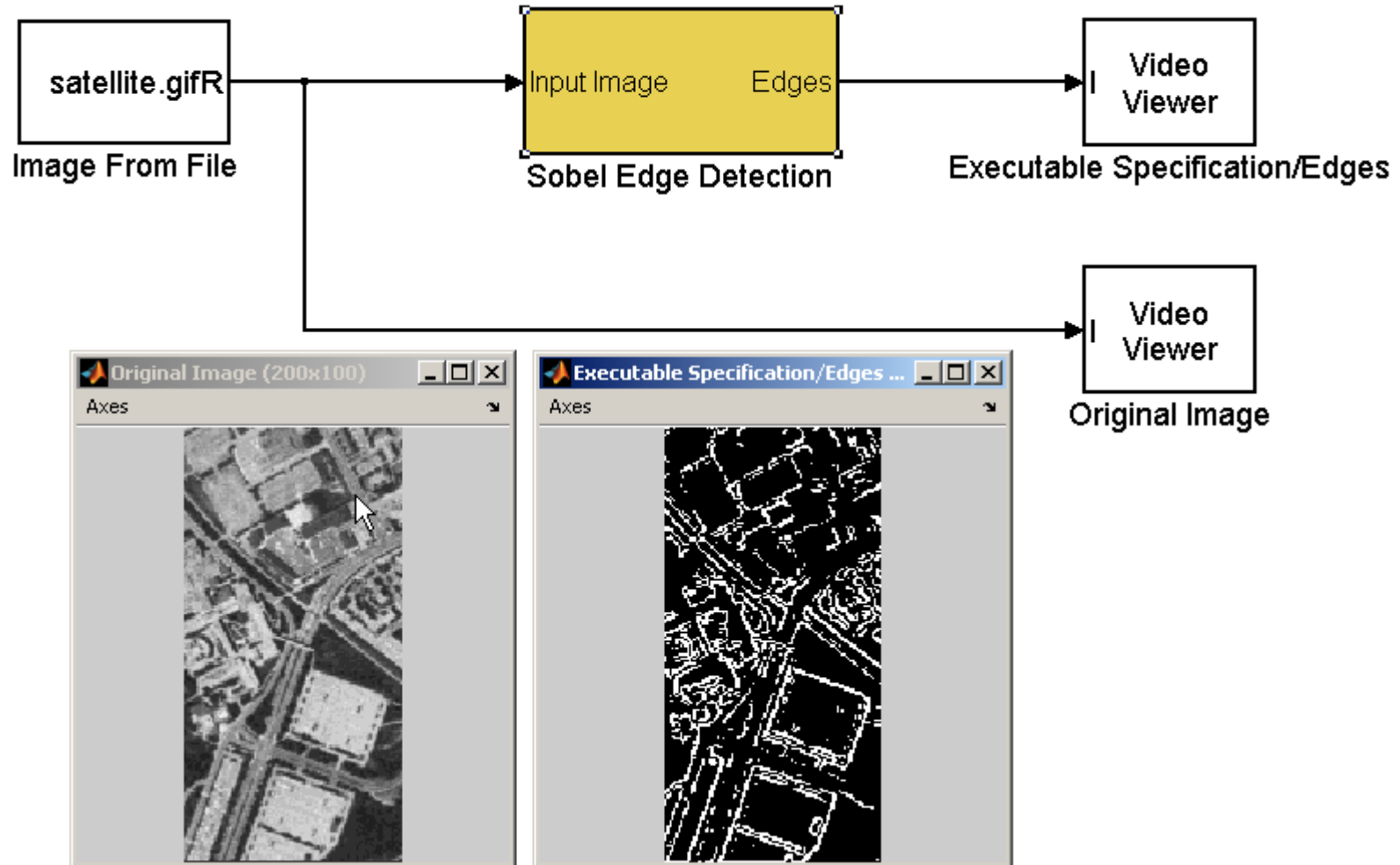
Hardware Case Study:  
Video Edge Detection  
on an FPGA



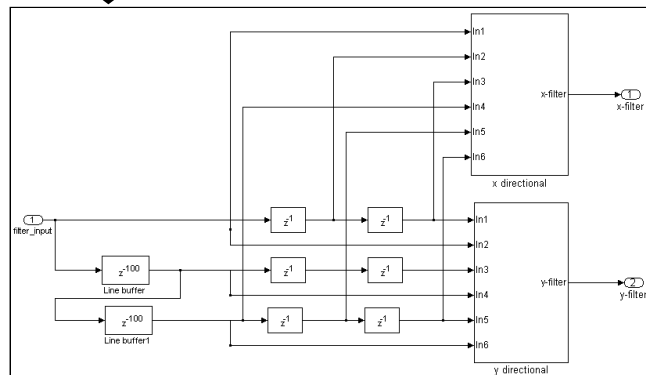
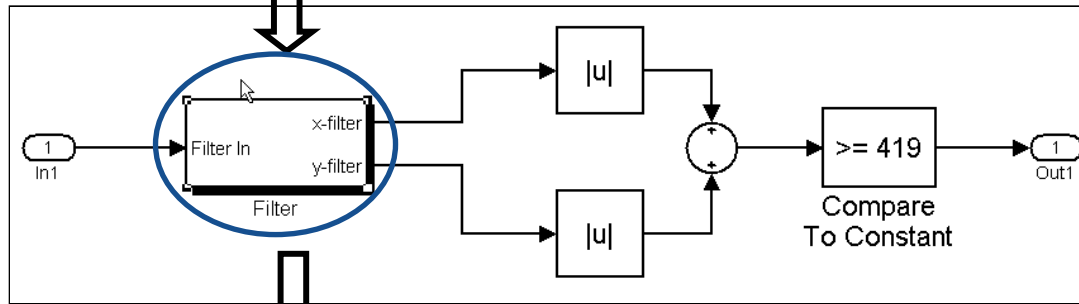
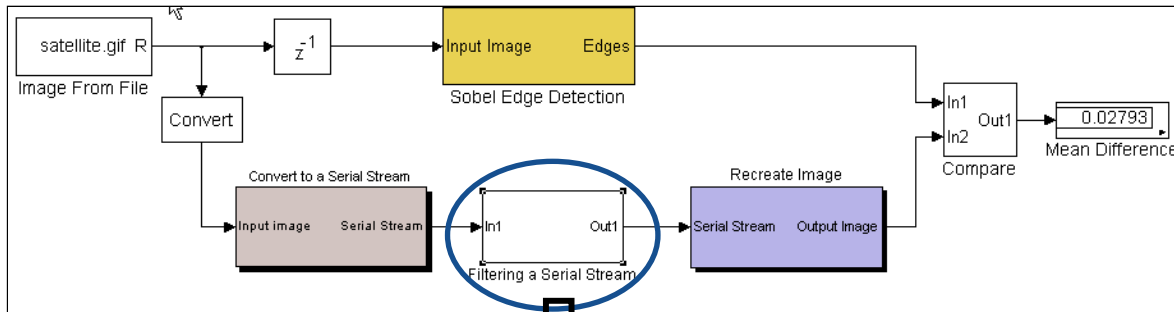
Live  
MATLAB  
Demo



# Executable Specification



# Design Elaboration



Automatically generate Verilog or VHDL code from elaborated models



# HDL Code Generation Using GUI

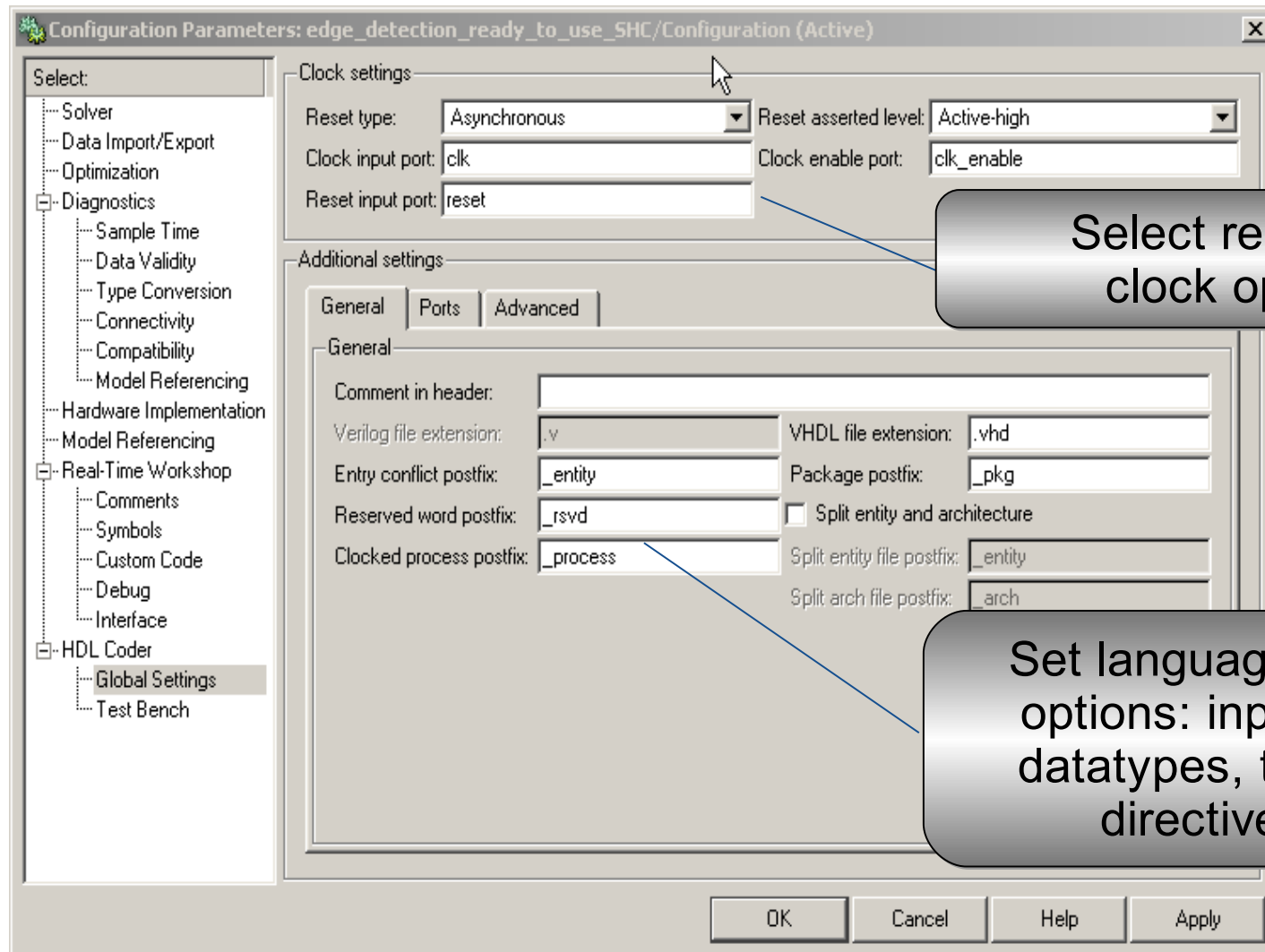
The screenshot shows the 'Configuration Parameters' dialog box for HDL code generation. The 'HDL Coder' section is expanded in the left sidebar. The main area is divided into several sections:

- 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 buttons: '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.

Callouts with arrows point to the following elements:

- Select subsystem, target language, directory:** Points to the 'Generate HDL for:', 'Language:', and 'Directory:' fields.
- Select output options:** Points to the 'Generate HDL code' radio button.
- Check model for errors:** Points to the 'Run Compatibility Checker' button.
- Generate HDL Code:** Points to the 'Generate' button.

# More Code Generation Options



Select reset and clock options

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

# Automatically Generate Test Bench

**Configuration Parameters: hdlcoder/ms/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):

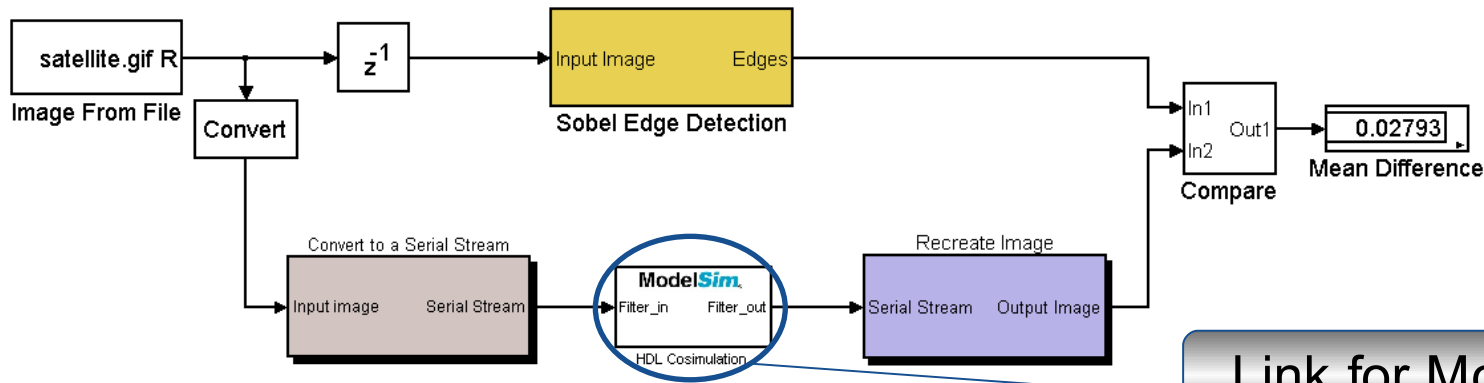
Force clock enable

Force reset

Hold time (ns):

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

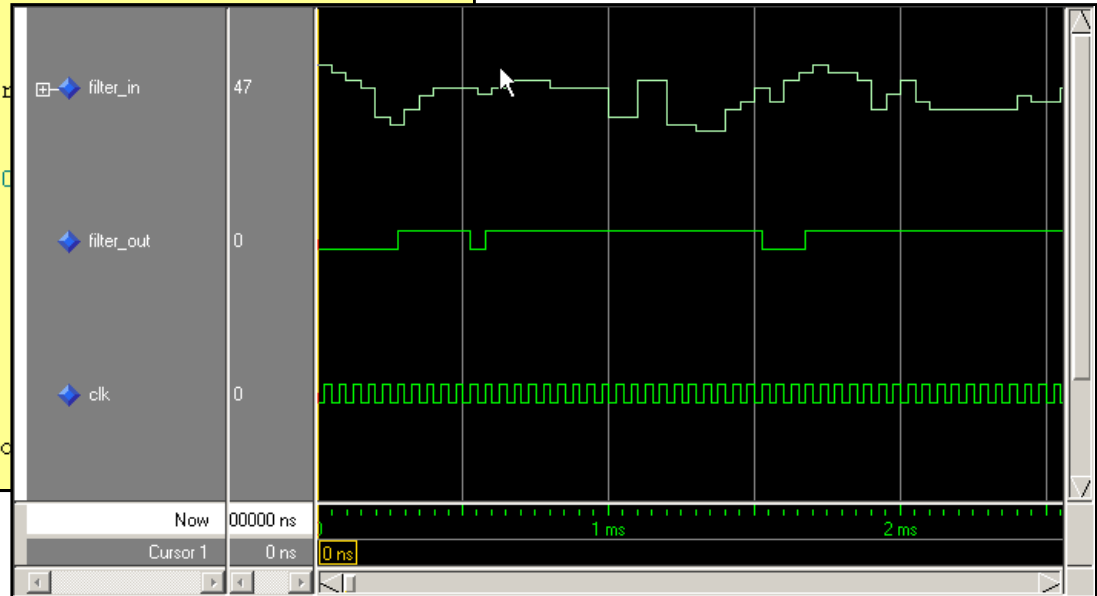
# Co-simulate Generated HDL



```

117     y_filter => y_directional_out1 -- ufix11
118   );
119   s <= signed(filter_input);
120
121   Delay2_process : PROCESS (clk,
122 BEGIN
123   IF reset = '1' THEN
124     Delay2_out1 <= (OTHERS => '0');
125   ELSIF clk'event AND clk = '1' THEN
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_o
134

```

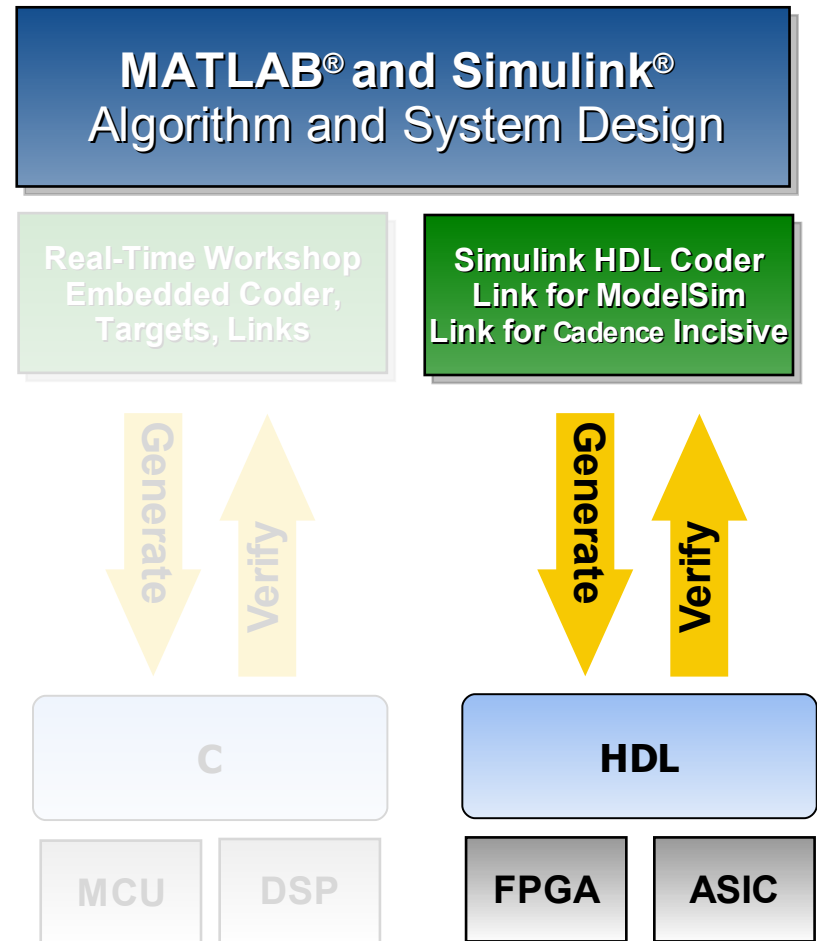


HDL code executing on ModelSim simulator

Link for ModelSim

# Review: Integrated Design Flow for Hardware

- Drive system development with an executable specification
- Quickly create complete working HDL code base and test benches
- Verify code with Links to RTL simulators and synthesis tools



# Agenda

- **The System Design Challenge**
- **Software Design Flow**
- **Hardware Design Flow**
- **Summary**

# Summary

- Accelerate development using Model-Based Design
  - Generate
    - Real-Time Workshop
    - Simulink HDL Coder
  - Verify
    - Link for Cadence Incisive
  
- Design and verify software and hardware from MATLAB and Simulink

