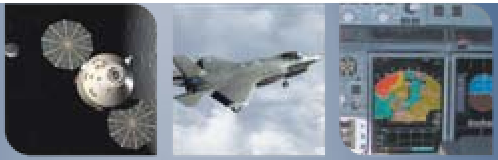
The MathWorks MATLAB® & SIMULINK®

Moving MATLAB® Algorithms into Complete Designs with Fixed-Point Simulation and Code Generation

Houman Zarrinkoub, PhD.
Product Manager
Signal Processing Toolboxes
The MathWorks Inc.



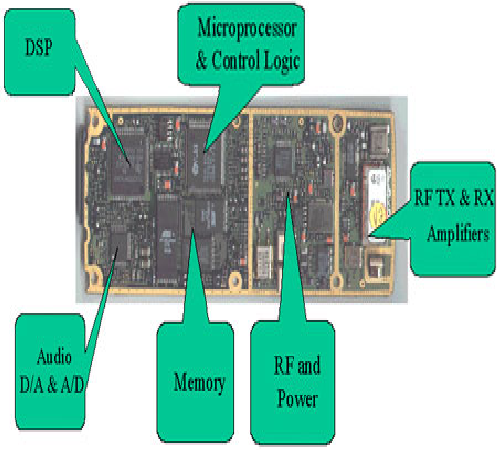
MathWorks
Aerospace and Defense Conference '07

Outline

- Challenges in fixed-point signal processing
- Traditional design workflow
- Advantages of Model-Based Design workflow
  - Streamlined Float-to-fixed conversion
  - Acceleration of fixed-point simulation
  - Automatic C code generation
- Demo
- Summary

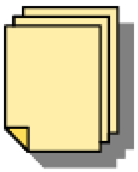
Fixed-point signal processing applications

- Tasks
  - Design and analyze fixed-point algorithms
  - Verify fixed point implementations
- Hardware targets
  - FPGA or ASIC
  - Fixed-point DSP chip



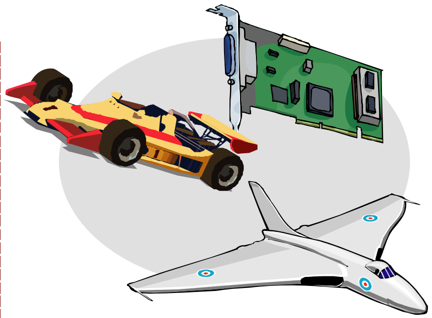
Traditional Development flow

Requirements and Specifications



Text-based
- Prevents rapid iteration

Design



Physical prototypes
- Incomplete and expensive

Implementation

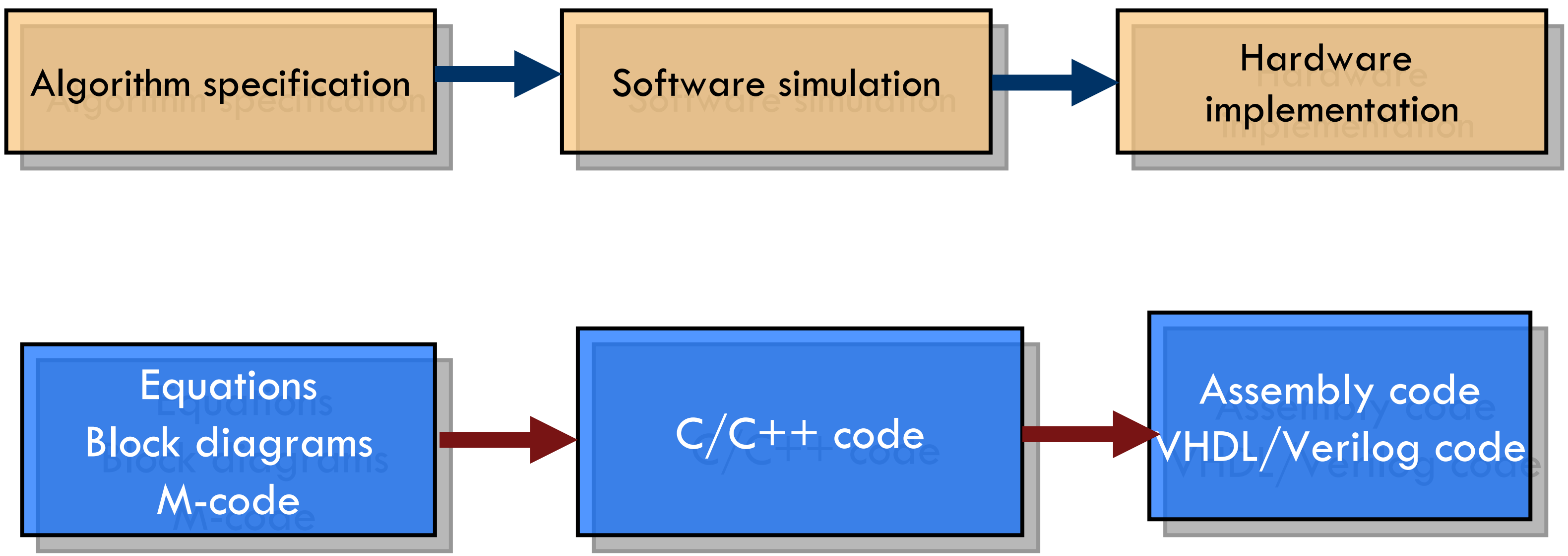
Manual coding
- Introduces human error

Test and Verification

Traditional testing
- Errors found too late in the process

Multiple truths in traditional workflows

Re-implement as you go down the level of abstraction



Examine a fixed-point algorithm design

Traditional workflow

MATLAB

1. Set-up simulation flow
2. Express your floating-point algorithm
   - Focus on algorithmic integrity, proof of concept
3. Simulate (floating-point)
   - Iterate on algorithm trade-offs
   - Validate against requirements

C/C++

4. Convert design to fixed-point
   - Focus of design viability based on implementation constraints
5. Simulate (fixed-point)
   - Iterate on implementation trade-offs
   - Validate against original requirements

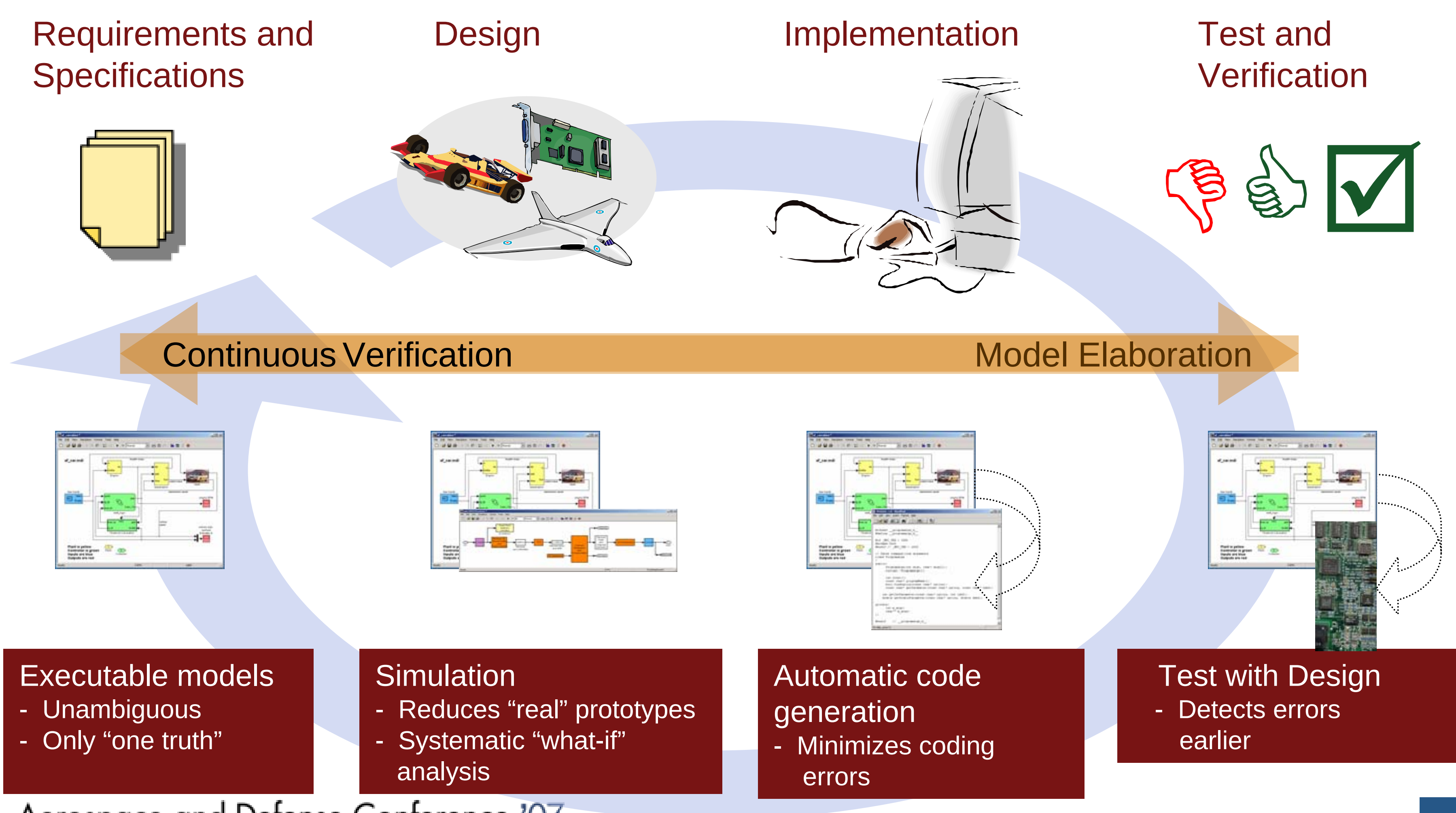
Assembly or HDL

6. Generate code for implementation
7. Validate and verify design after deployment

Problems with traditional workflow

- Multiple truths (Copies of same algorithm)
  - Floating-point M code
  - Floating-point C code
  - Fixed-point C code
  - Assembly code
  - Verilog/VHDL code
- Error-prone process
  - Using different tools
  - Exchange data across tools
  - Multiple update/test of code

Model-Based Design Workflow



Examine a fixed-point algorithm design

Model-based Design workflow

1. Set-up simulation flow
2. Express your floating-point algorithm
   - Focus on algorithmic integrity, proof of concept
3. Simulate (floating-point)
   - Iterate on algorithm trade-offs
   - Validate against requirements
4. Convert design to fixed-point
   - Focus on design viability based on implementation constraints
5. Simulate (fixed-point)
   - Iterate on implementation trade-offs
   - Validate against original requirements
6. Generate code for implementation
7. Validate and verify design after deployment

MATLAB & Simulink

Advantages of Model-Based Design workflow

- Maintain One Truth
- One integrated design environment
- MATLAB benefit:
  - Integrated visualization, analysis & design
- No sacrifice of simulation speed
- Automatic path to implementation

How does Model-Based Design makes fixed-point design faster and easier?

- Streamline process of converting your MATLAB algorithms to fixed-point
- Simulate fixed-point algorithms with large data sets at compiled-C-code speed
- Integrate with system-level design in Simulink
- Generate embeddable C code for implementation with Real-Time Workshop®

Streamlined floating-to-fixed conversion: introducing Data-type override

- Turn on the logging mode
- Set data type override parameters
- Observe dynamic range of variables in your M-code
- Set the best fixed-point attributes to avoid overflow/underflow & large quantization errors

MATLAB 7.4.0 (R2007a)

File Edit Debug Desktop Window Help

To get started, select MATLAB Help or Demos from the Help menu.

```
>> reset(fipref);
>> fipref('LoggingMode','on');
>> fipref('DataTypeOverride','ScaledDoubles');
>> logreport(b,a,x,y,acc)
                  minlog         maxlog     lowerbound     upperbound     noverflows    nunderflows
         b     0.2929077      0.5857849             -1      0.9999695              0              0
         a             0              1             -2       1.999939              0              0
         x    -0.9999507      0.9999695             -1      0.9999695              0              7
         y    -0.9555664      0.9640198             -1      0.9999695              0              0
       acc            -1      0.9999695             -1      0.9999695             17              1
>> 
```

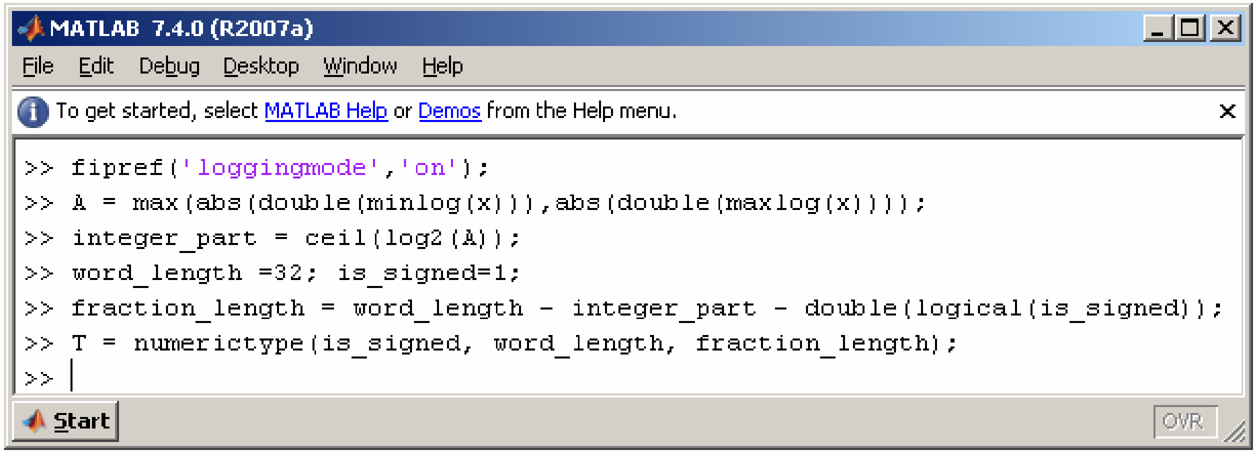
Start   OVR

Tools for scaling a fixed-point variable

Data logging

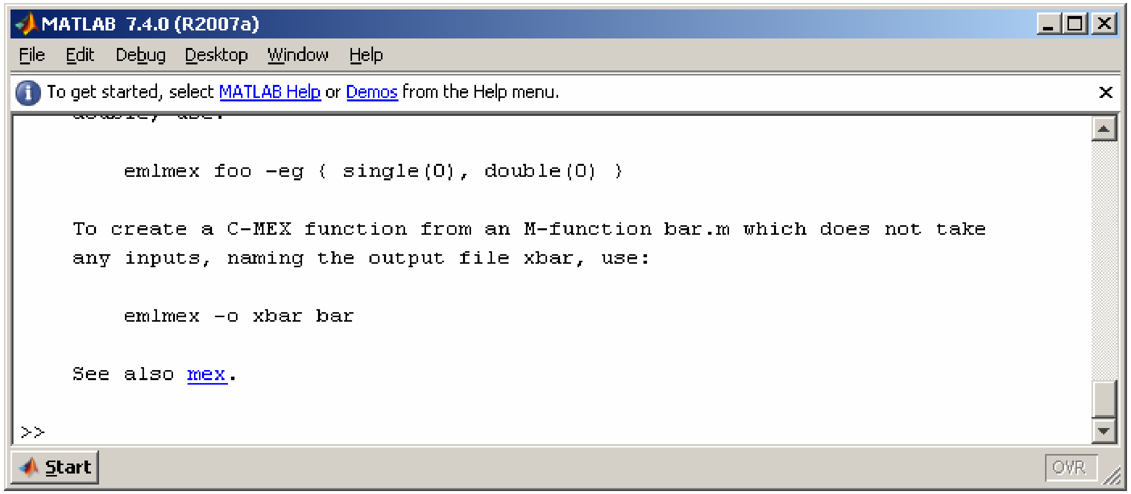
Steps involved with dynamic range analysis to convert a design into fixed-point

1. Compute the range based on the min/max logs
2. Compute the integer part to fit variable within range
3. Compute the fraction length as the rest of bit budget
4. Construct the fixed-point numeric type object



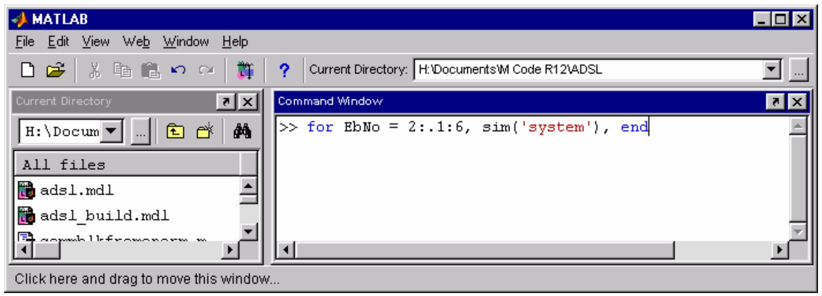
Fixed-point acceleration: introducing new emlmex function

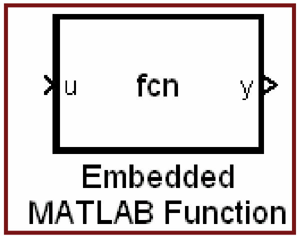
- Fast simulation through code generation
- Automatic generation of C-MEX function from M-function
- M-code confined to embedded MATLAB language subset
- Compile C-code execution speed (beyond 100x acceleration in MATLAB)

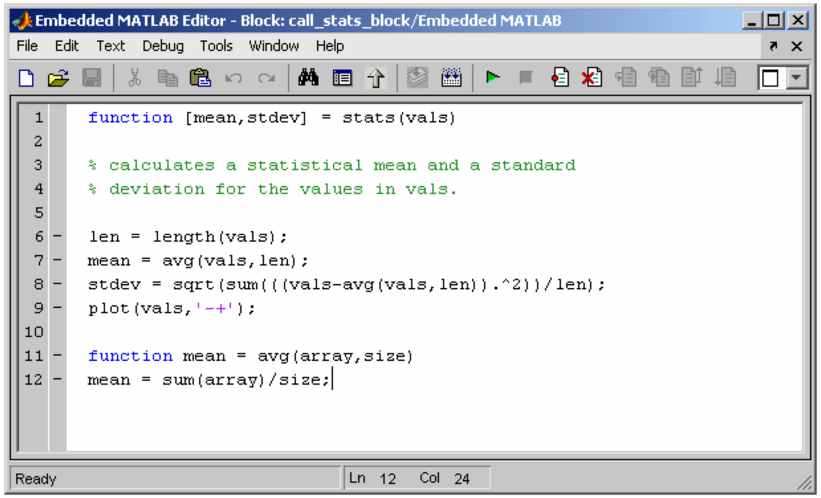


Integrate MATLAB design with Simulink

Embedded MATLAB Function block







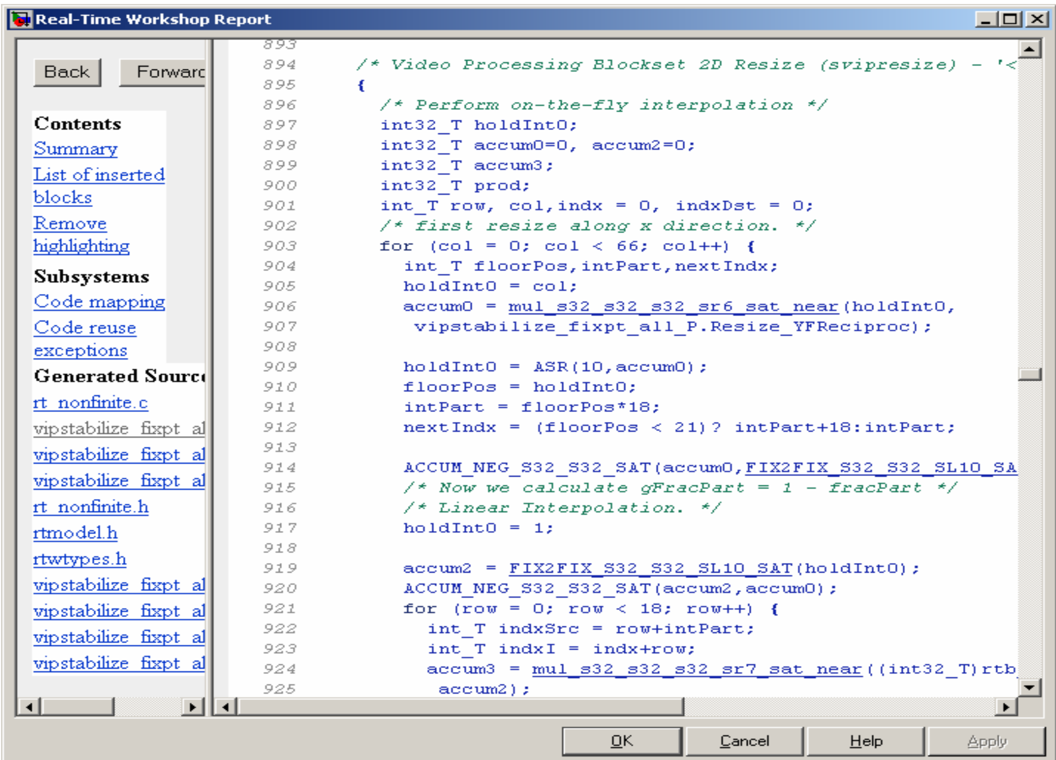
Change parameters and run Simulink® simulations from MATLAB

Embedded MATLAB Function

Integration of Embedded MATLAB Functions in Simulink

Automatic fixed-point C code generation

Real-Time Workshop



- Enabled via Simulink Fixed-point
  - Real-Time Workshop®
  - Real-Time Workshop® Embedded Coder
- Supports up to 32-bit fixed-point numbers
- Uses only native C integer data types

Hands-on Demonstration

1. Implement the algorithm with floating-point data types in M.
2. Convert to fixed-point data types in M and run with default settings; observe scaling issues!
3. Log the full numerical range of variables (data logging and data type override)
4. Use the logged minimum and maximum values to set the fixed-point scaling.
5. Validate the fixed-point solution interactively.
6. Convert M to MEX using EMLMEX function for fast simulation and large test-set verification.
7. Convert M to C in Simulink Embedded MATLAB Function block with Real-Time Workshop for embedded implementation.

For more information

- Fixed-point signal processing webinars
  - Fixed-Point Programming in MATLAB
    - http://www.mathworks.com/cmspro/req11440.html?eventid=32477
  - Fixed-Point Signal Processing with MATLAB and Simulink
    - http://www.mathworks.com/cmspro/req12157.html?eventid=35522

- About MATLAB and Simulink signal processing products

  http://www.mathworks.com/products/product_listing/index.html
  - Relevant product demos
    - http://www.mathworks.com/products/demos/index.html
  - User-contributed examples in MATLAB Central
    - http://www.mathworks.com/matlabcentral

Summary

Model-based design

- Single-truth, integrated design environment for development of a design from idea all the way to realizable implementation

Benefits

- Integrated modeling, simulation and prototyping for signal processing systems
- Easy conversion to fixed-point data types and trade-off analyses
- Automatic generation of C-code for DSPs
- Construct test harnesses for real-time hardware verification