TI TECH DAYS

# Model-Based Design of Video Applications for TI DSPs

**Dick Benson**
Consultant Application Engineer
dick.benson@mathworks.com

# Agenda

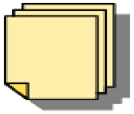
- Introduction to Model-Based Design *(15 min)*
- Basics of Simulink® *(15 min)*
- Design and Implementation of Video Applications *(35 min)*
  - *Edge detection example*
- Advanced Video Applications *(15 min)*
  - *Video stabilization example*
- Next Steps and Discussion *(10 min)*

# System Design Challenges

- Increasing system complexity and computation demands
- Embedded system resource constraints
  - Real-time requirements
- Designing for a target processor
  - Micro-controller, GPP, DSP, FPGA
- End-product price, power, size
  - Roadmap for adding features, performance
- Testing and validating results

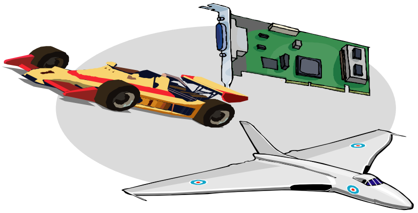
# Problems with Traditional Development

### Requirements and Specifications



**Text-based**
- Prevents rapid iteration

### Design



**Physical prototypes**
- Incomplete and expensive

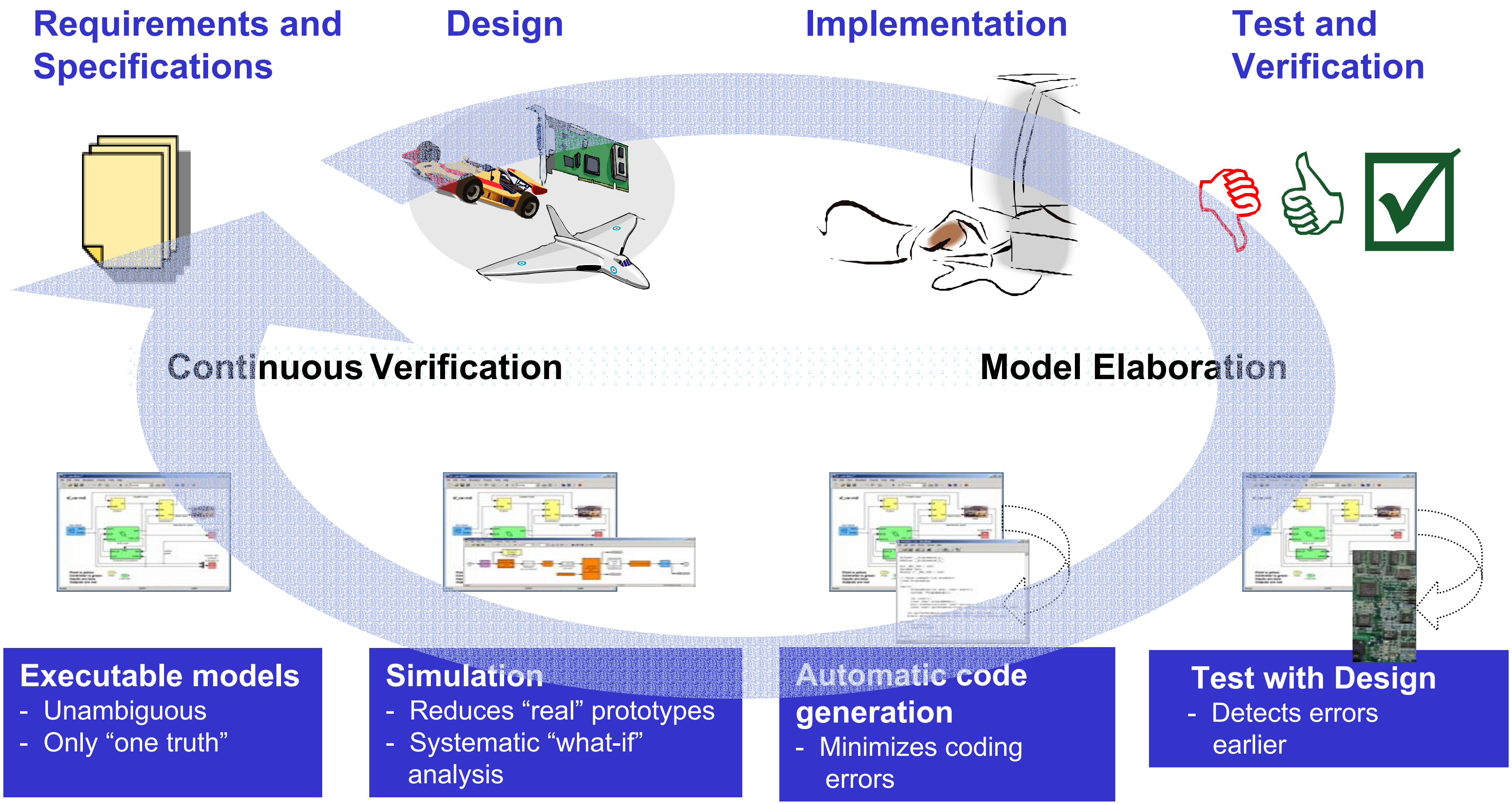
### Implementation

**Manual coding**
- Introduces human error

### Test and Verification

**Traditional testing**
- Errors found too late in the process

# Advantages of Model-Based Design



# Real Results Across Industries

TOYOTA — Standard for Powertrain Controls Production Code Development

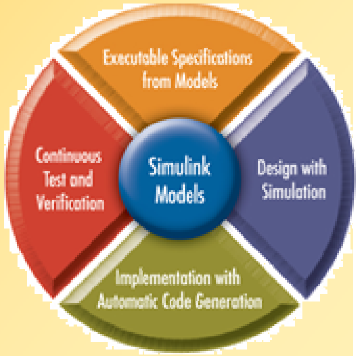
LOCKHEED MARTIN — JSF Flight Control System

zyray WIRELESS — W-CDMA Baseband Processors

TEXAS INSTRUMENTS — Specialty Chipsets for DSP Customers

# The Value of Model-Based Design

Model-Based Design
- **Executable specification**
- **Design with simulation**
- **Implementation through code generation**
- **Continuous test and verification**




Innovation
- **Rapid design iterations**
- **“What-if” studies**
- **Unique features and differentiators**

Quality
- **Reduce design errors**
- **Minimize hand coding errors**
- **Unambiguous communication internally and externally**

Cost
- **Reduce expensive physical prototypes**
- **Reduce re-work**
- **Reduce testing**

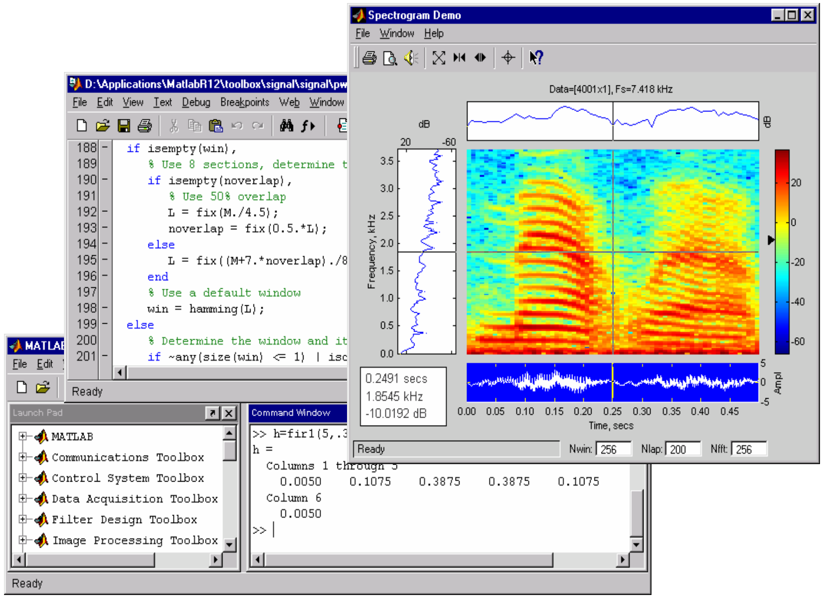
Time-to-market
- **Get it right the first time**

# Technical Computing

## MATLAB

### The leading environment for technical computing

- The *de facto* industry-standard, high-level programming language for algorithm development
- Toolboxes for control system design, signal and image processing, statistics, optimization, symbolic math, and other areas
- Foundation of the MathWorks product family

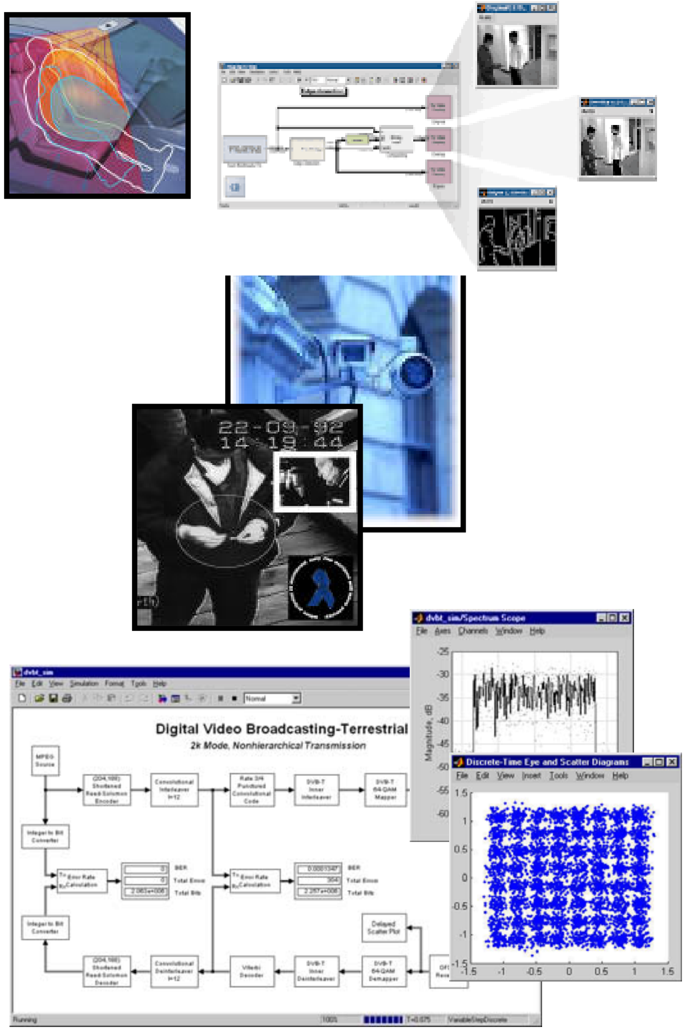


# Model-Based Design

SIMULINK®

The leading environment for modeling, simulating, and implementing dynamic and embedded systems

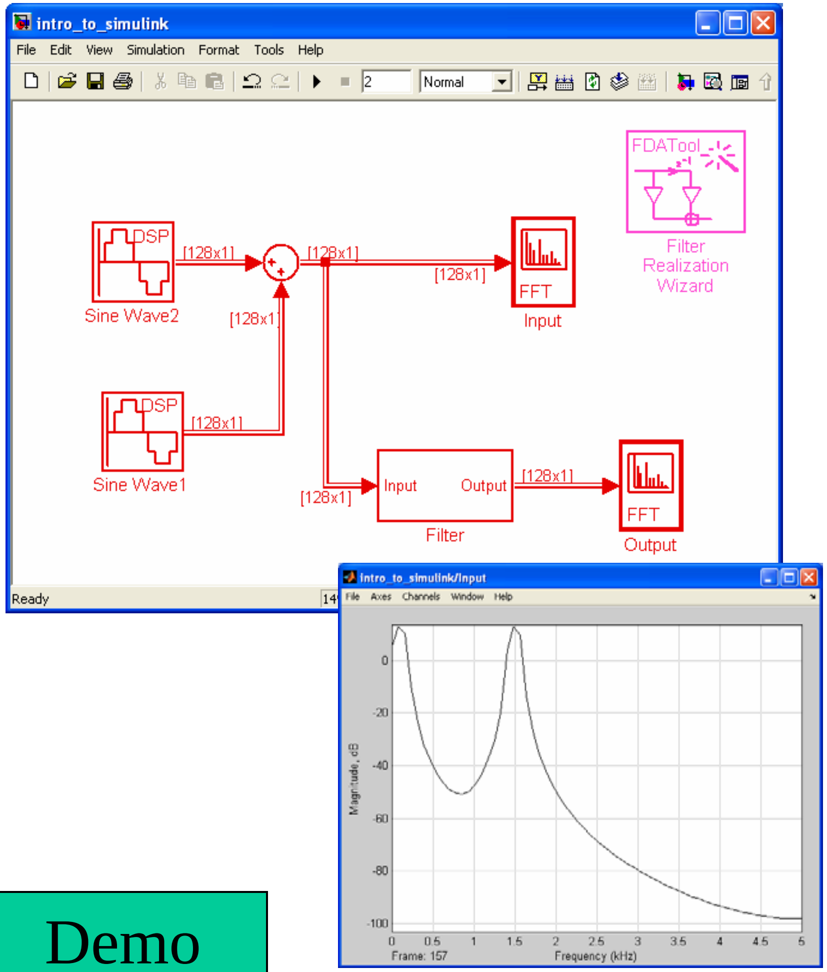
- Foundation for model-based design, including physical-domain modeling, automatic code generation, and verification and validation
- Open architecture for integrating models from other tools
- Applications in controls, signal processing, communications, and other system engineering areas



# Basics of Simulink

# Basics of Simulink

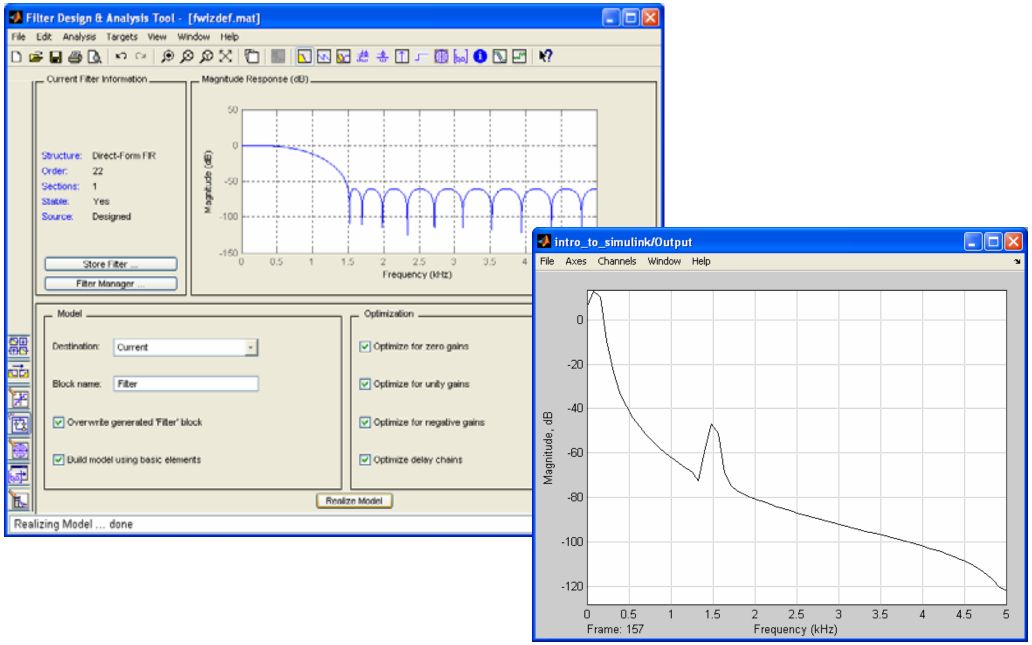
- Simple signal processing model
  - Signal Processing Blockset
    - SP Sources: Sine Wave
    - SP Sinks: Spectrum Scope
    - Filtering: Filter Designs: Filter Realization Wizard
  - Simulink
    - Math Operations: Sum




Demo

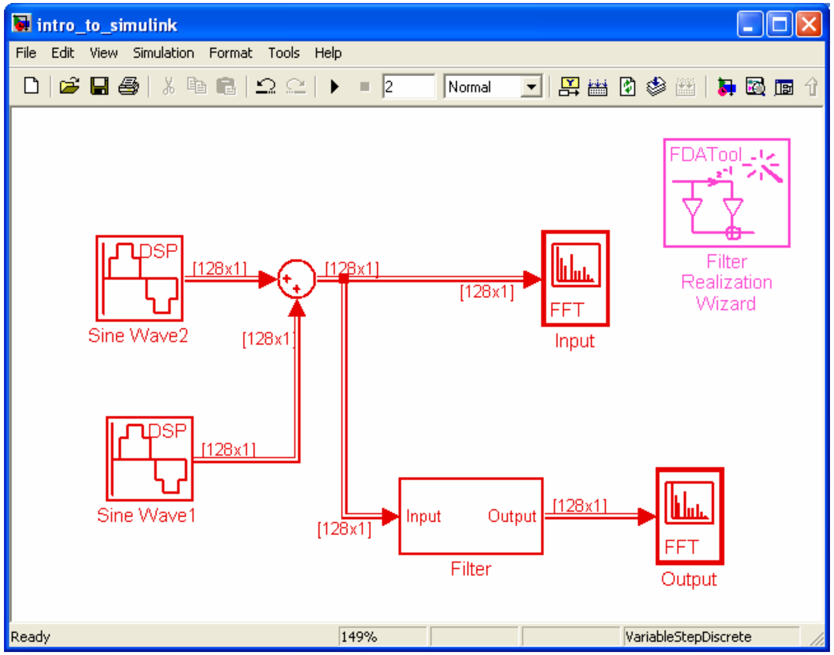
# Basics of Simulink (continued)

- Build low-pass filter
- Filter out the high-frequency tone

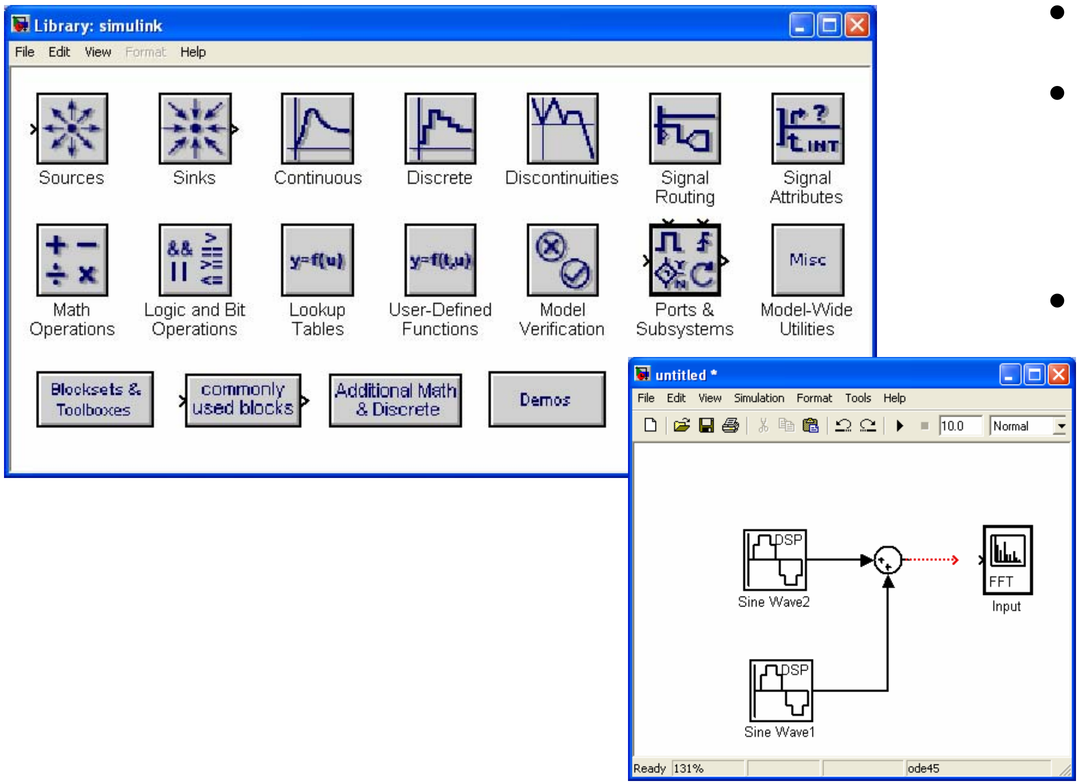


# Basics of Simulink (continued)

- Design, simulate, test, and visualize with Simulink
- Frame-based processing
- Use M-code and filters designed in MATLAB

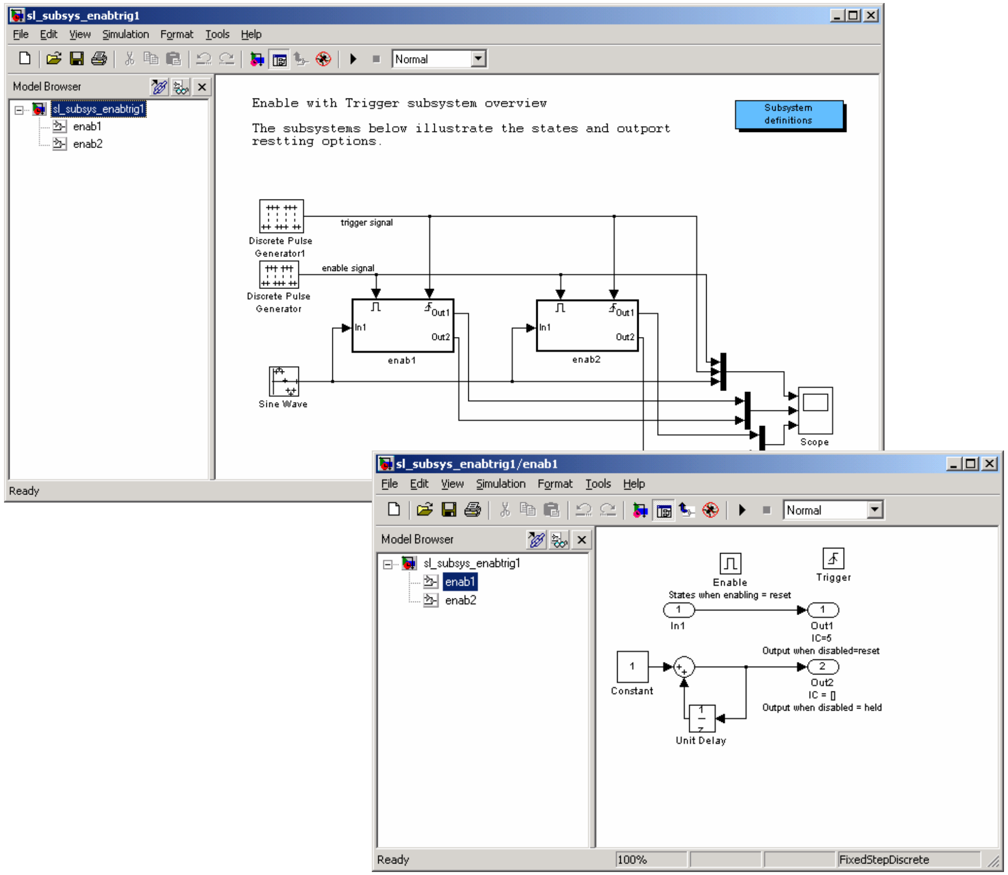


# Model Construction



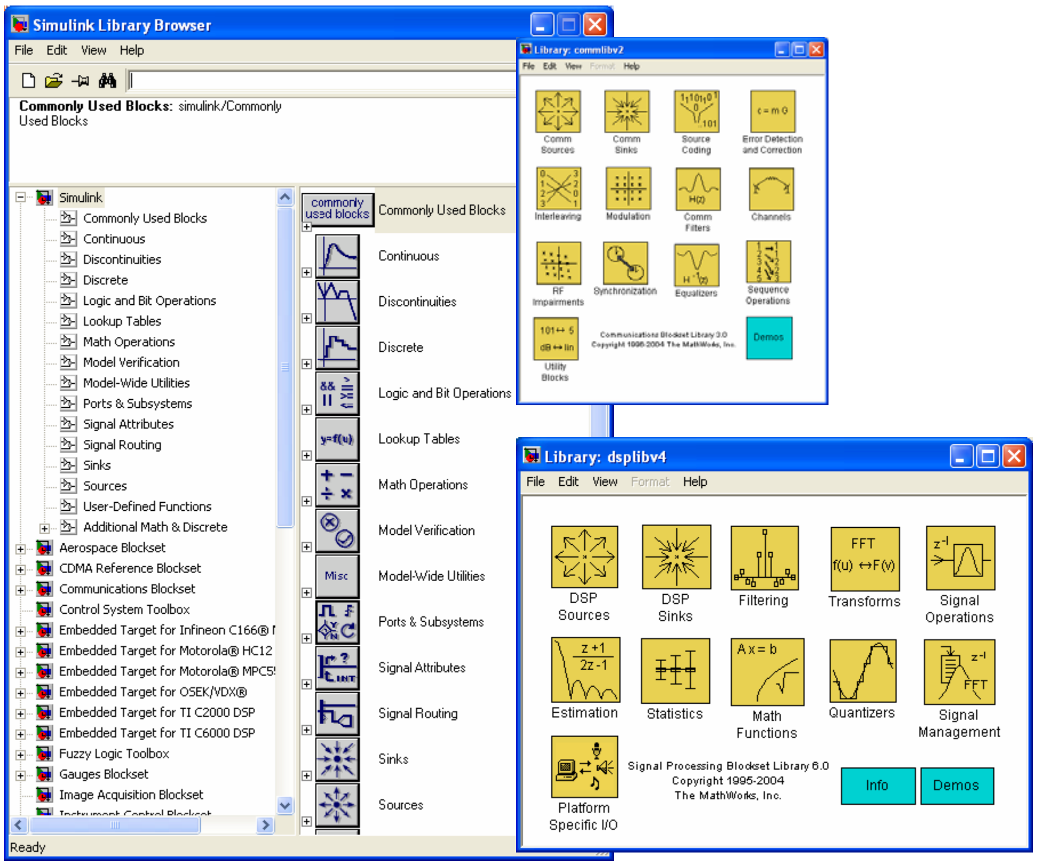

- Drag and drop
- Connect
- Digital
  - **Fast frame-based simulation**
- Analog
  - **Variable-step numerical integration solvers**
  - **Zero-crossing detection**

# Sub-Systems and Hierarchy




- Group multiple blocks into subsystem to any level
- Model browser
- Conditionally executed subsystem
  - **Enabled and triggered**
  - **If, while, for, switch**
- Configurable subsystem
  - **Swap model components easily**

# The Block Libraries



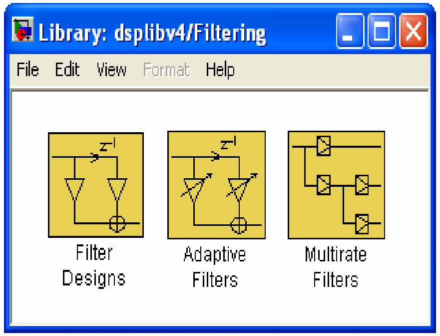

- Simulink
  - **Sources**
  - **Sinks**
  - **Continuous**
  - **Discrete**
  - **Nonlinear**
  - **Math**
- Simulink Fixed Point

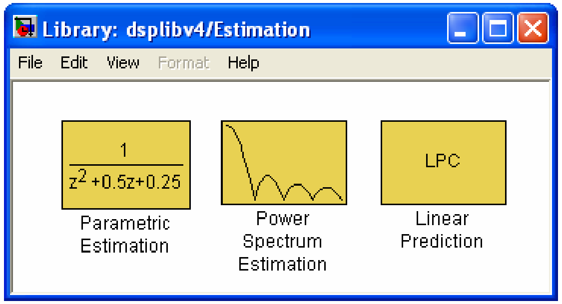
- **Signal Processing Blockset**
- **Video and Image Processing Blockset**

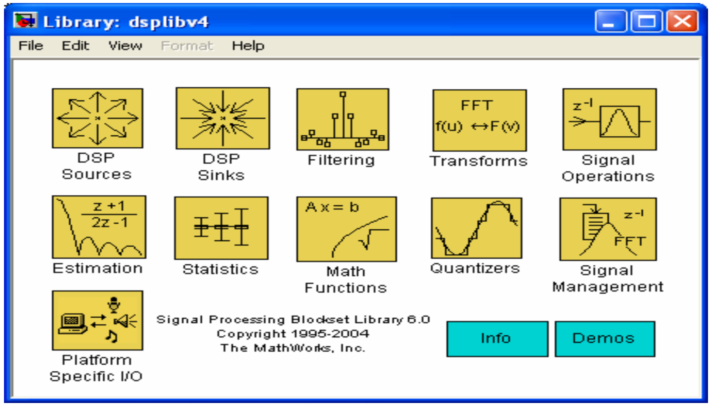
- Communications Blockset
- RF Blockset
- Others

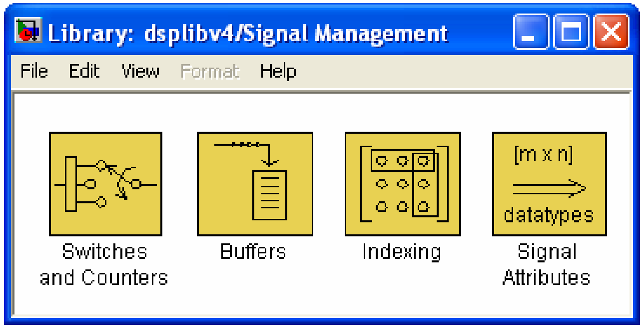
# Signal Processing Blockset

- Streaming data
- Multi-rate systems
- Transforms, filters, estimators
- Enables frames in Simulink
- Fixed- and floating-point support

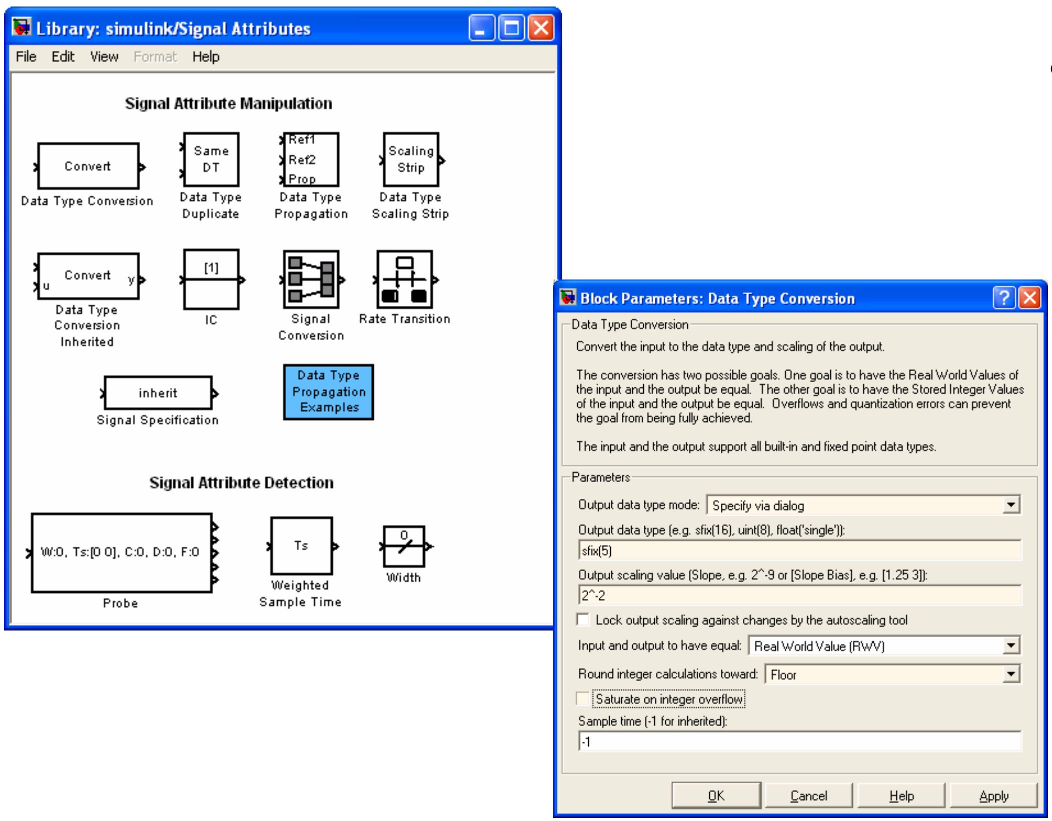


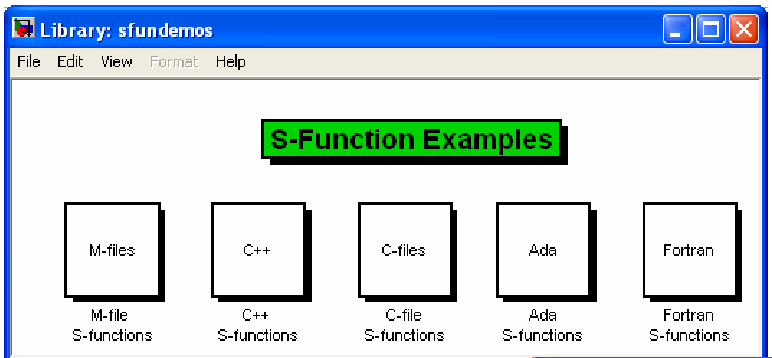



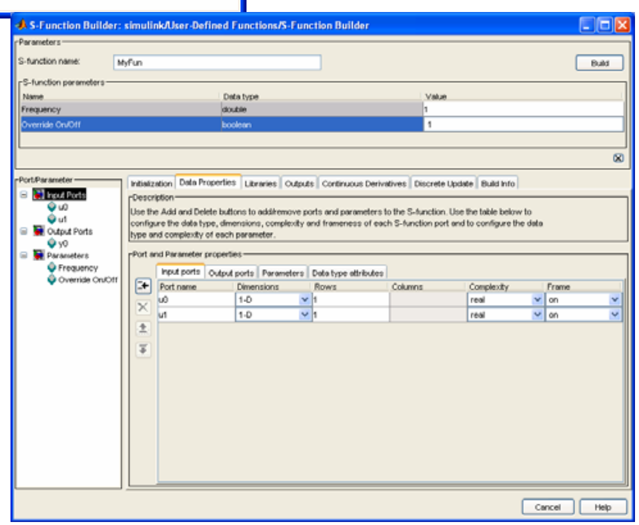
# Data Types




- Default double
- C data types in Simulink
- Simulink Fixed Point
  - **Specify word length**
  - **Integer, fixed, fractional, and custom float types**
  - **Trap overflow and saturation**
  - **Auto-scaling**
  - **Round-off options**
  - **Include own bit-true code**

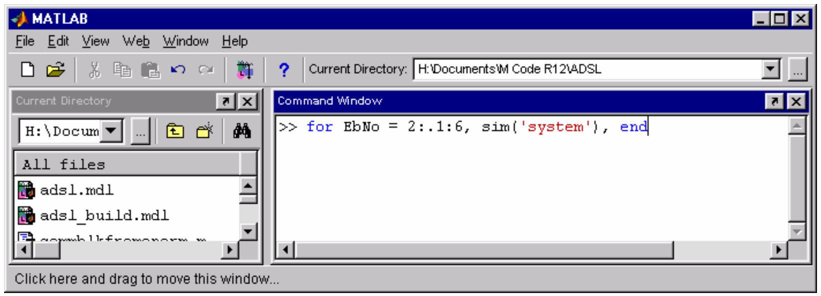
# User Defined Blocks and Libraries

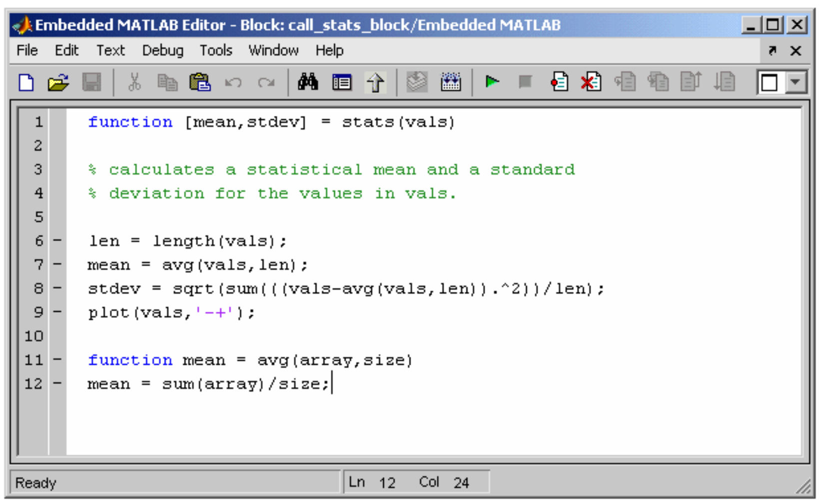



- Created from
  - **Other blocks**
  - **C Code**
  - **MATLAB Code**
- User-defined:
  - **Parameter GUIs "masks"**
  - **Icons**
  - **Libraries**

# Co-Develop with MATLAB

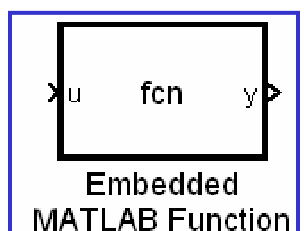


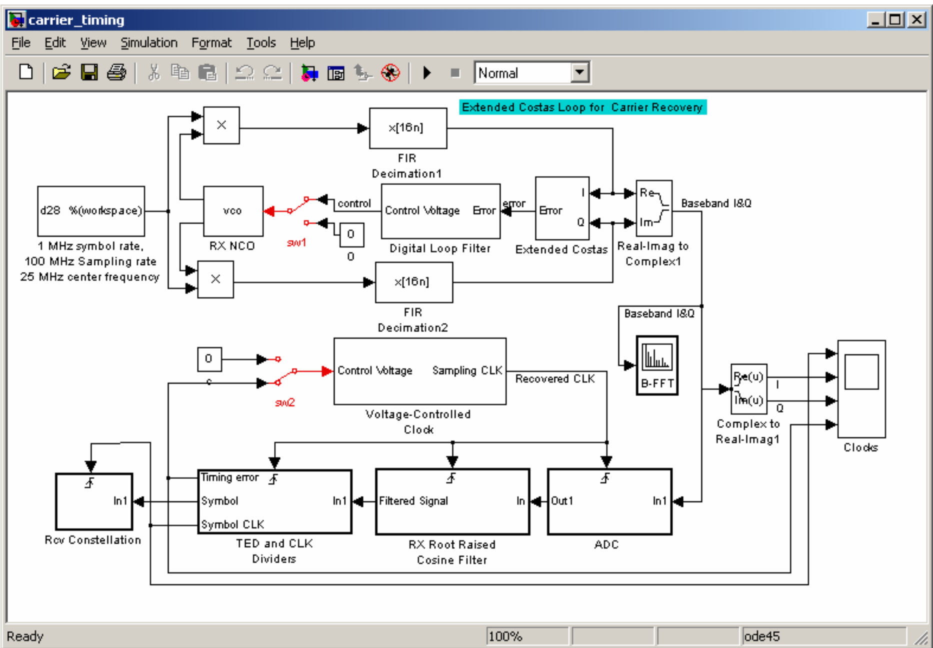

- Change parameters and run Simulink simulations from MATLAB

  **>> for EbNo = 2:.1:6, sim('system'), end**

- MATLAB S-functions
- Embedded MATLAB Function
  - **Integration of Embedded MATLAB Functions in Simulink**

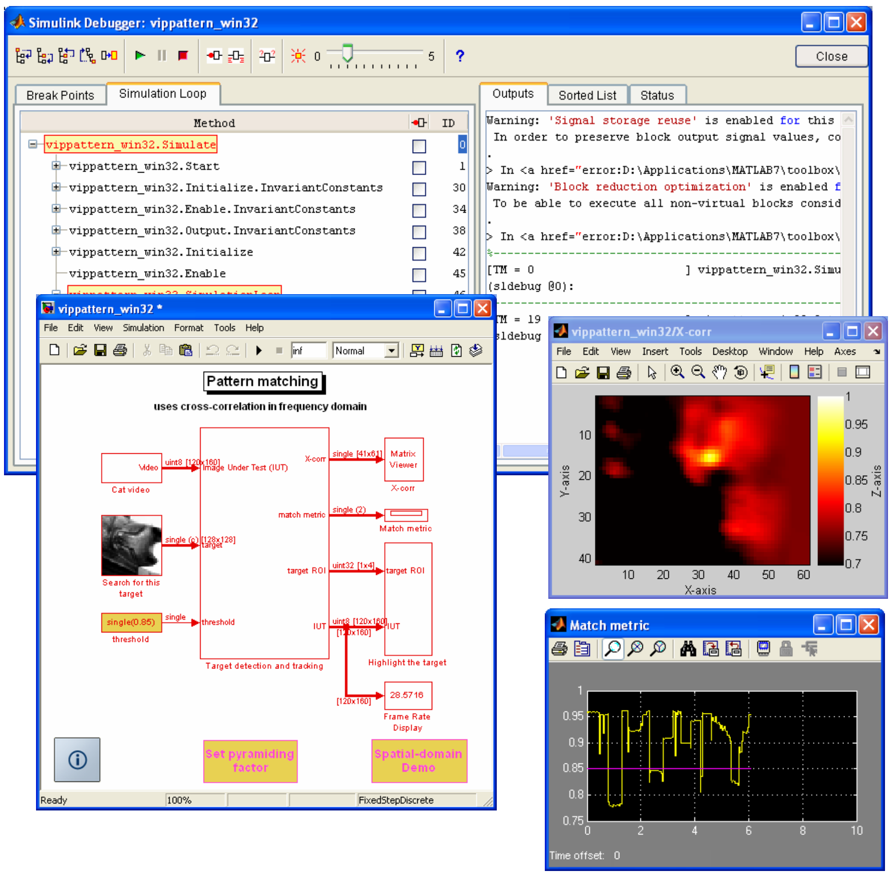


# Complex Timing and Concurrency




- Complex timing
  - **Feedback**
  - **Asynchronous edge triggered blocks**
  - **Multirate digital with arbitrary sample rates**
- Concurrency
  - **True expression of parallelism**
  - **Important for whole system or hardware subsystem design**
  - **Not possible with programming language such as C**

# Debug, Profile, and Accelerate Models




- Debug
  - **Single step blocks and look at inputs, state, and outputs**
  - **Stop on block or at specific time**
- Profile
  - **Generate report**
  - **Show elapsed time on every block**
  - **Optimize model simulation time**
- Accelerate
  - **Compile to C Code and run on host**

TI TECH DAYS

# Design and Implementation of Video Applications

# Modeling Video Applications

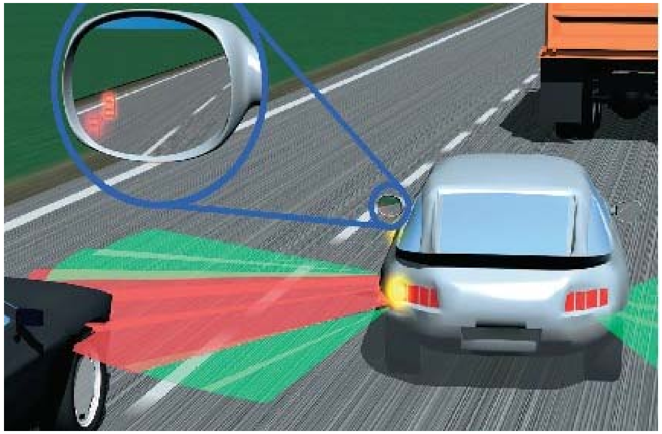
– **Video and Image Processing Blockset**

– Provides over 50 components and 100's of algorithms **focused on implementation of embedded systems**

| | | | |
|---|---|---|---|
| Streaming video in/out | Detection, Thresholding | Tracking, Counting | Background Estimation |

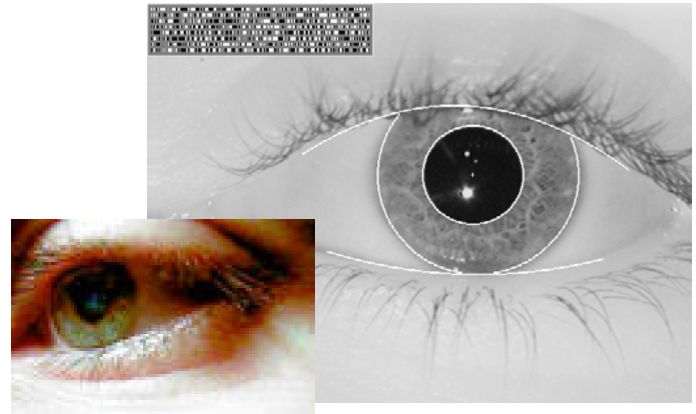
# Video and Image Processing Blockset Libraries

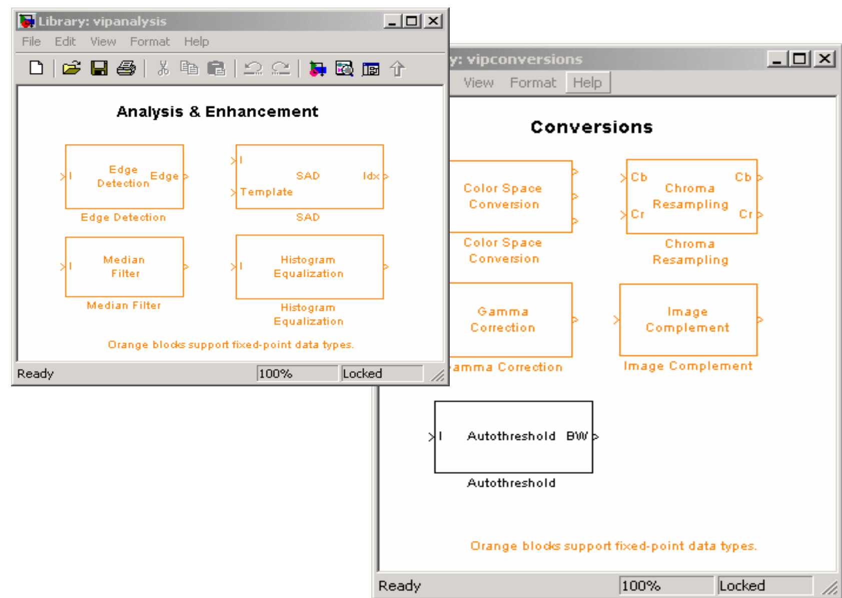
- Basic primitives
  - **Padding, correlation, statistics, thresholding**
  - **Block processing**
  - **2-D filtering, 2-D transforms**
- Geometric transformations
  - **Rotation, translation, resize, shear**
  - **Interpolation: nearest neighbor, bi-linear, bi-cubic**
- Edge detection
  - **Sobel, Prewitt, Roberts**
- Morphological operations
  - **Erode, dilate, open, close**
  - **Labeling of connected-components**



# Libraries (continued)

- Analysis and Enhancement
  - **Edge detection, median filtering, motion vector estimation (SAD)**
- Superimposing images and graphics
  - **On-screen text overlays, Picture-in-picture**
- Conversions
  - **Color-space conversions (RGB, YCbCr, etc)**
  - **Chrominance re-sampling (4:2:2, 4:2:0, etc)**





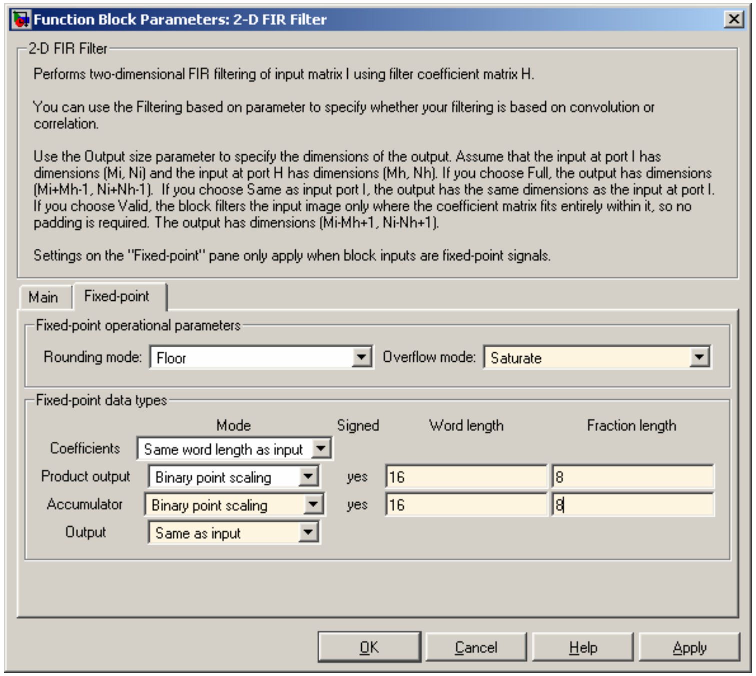
# Simulating Video Applications

- Simulink, Signal Processing Blockset, Video and Image Processing Blockset*
- Fixed-Point considerations

- Avoid inaccurate results due to finite word effects

- Built in tools for scaling and modeling finite word effects

- Easy to change parameters to simulate impact of rounding, overflow, etc.

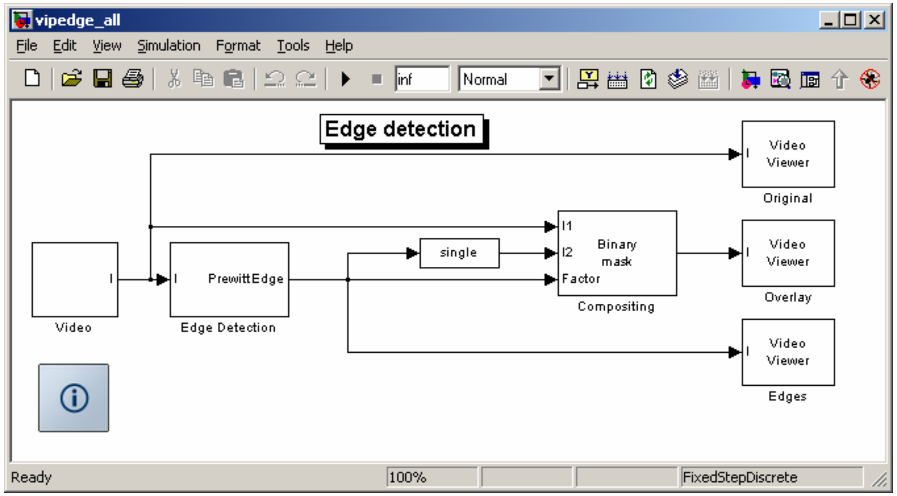



*Requires Simulink Fixed-Point for integer and fixed point data types

# Design an Edge Detection System

# Edge Detection

- **Fundamental component of many applications**
  - Object tracking and recognition
  - Biomedical signal processing
  - Unmanned vehicle technology
  - Segmentation for video compression



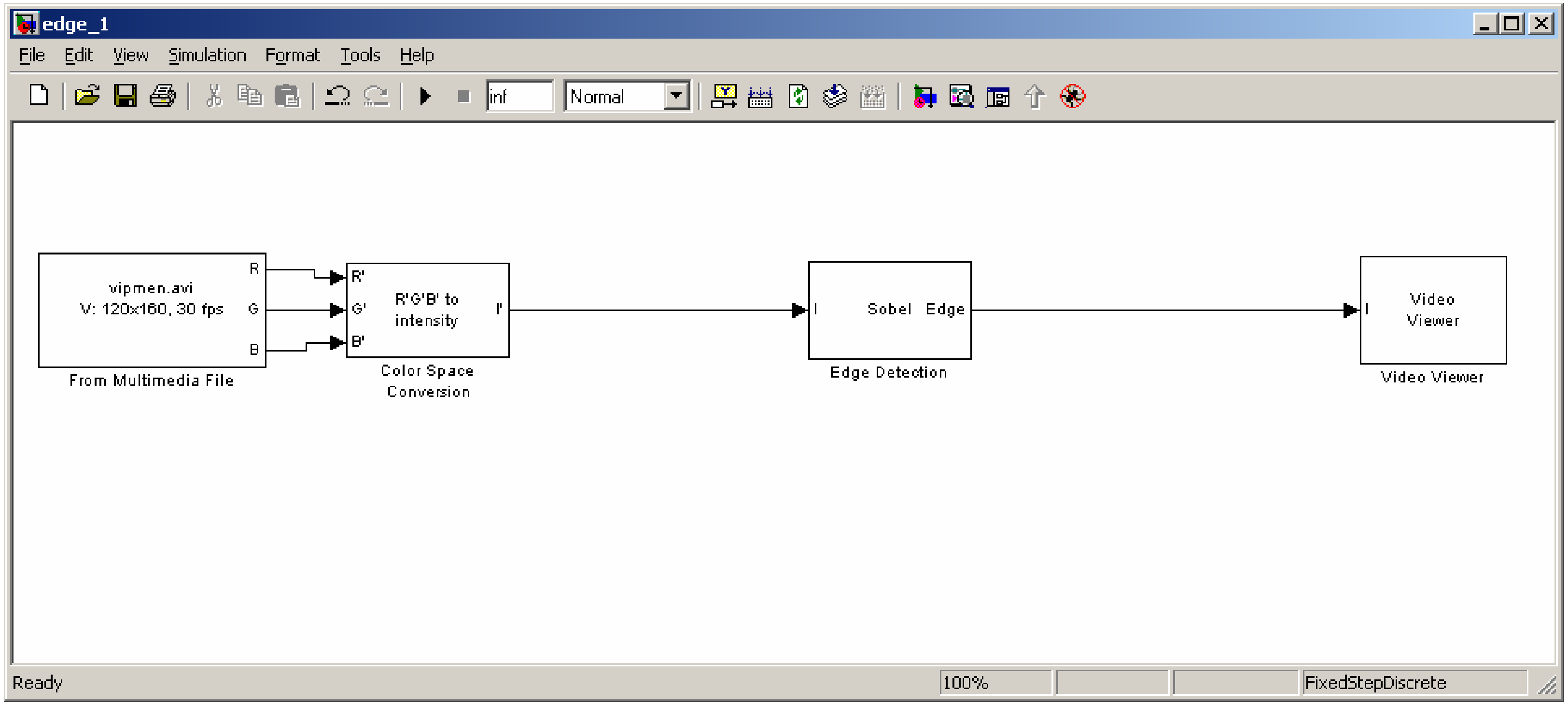

Demo

# Building Edge Detection Model

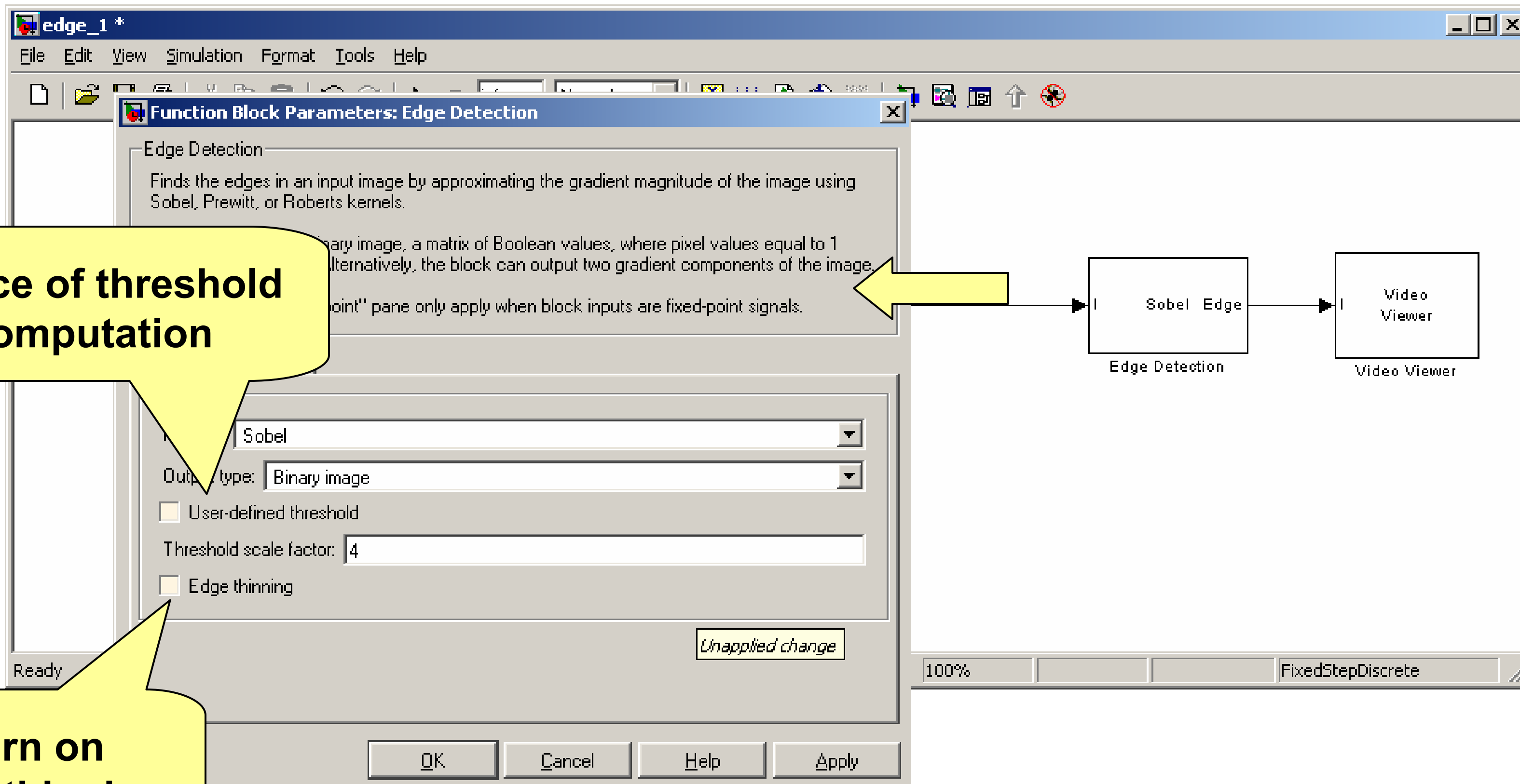
- **What you will do…**
  - Find edges
  - Overlay it onto original input
  - Convert the model to fixed-point

- **What you will learn about …**
  - Video sources and sinks
  - Data type and interpretation of color and intensity
  - Integer processing as a special case of fixed-point
  - Accelerator mode and fixed-point models

# Edge Detection and Video Compositing



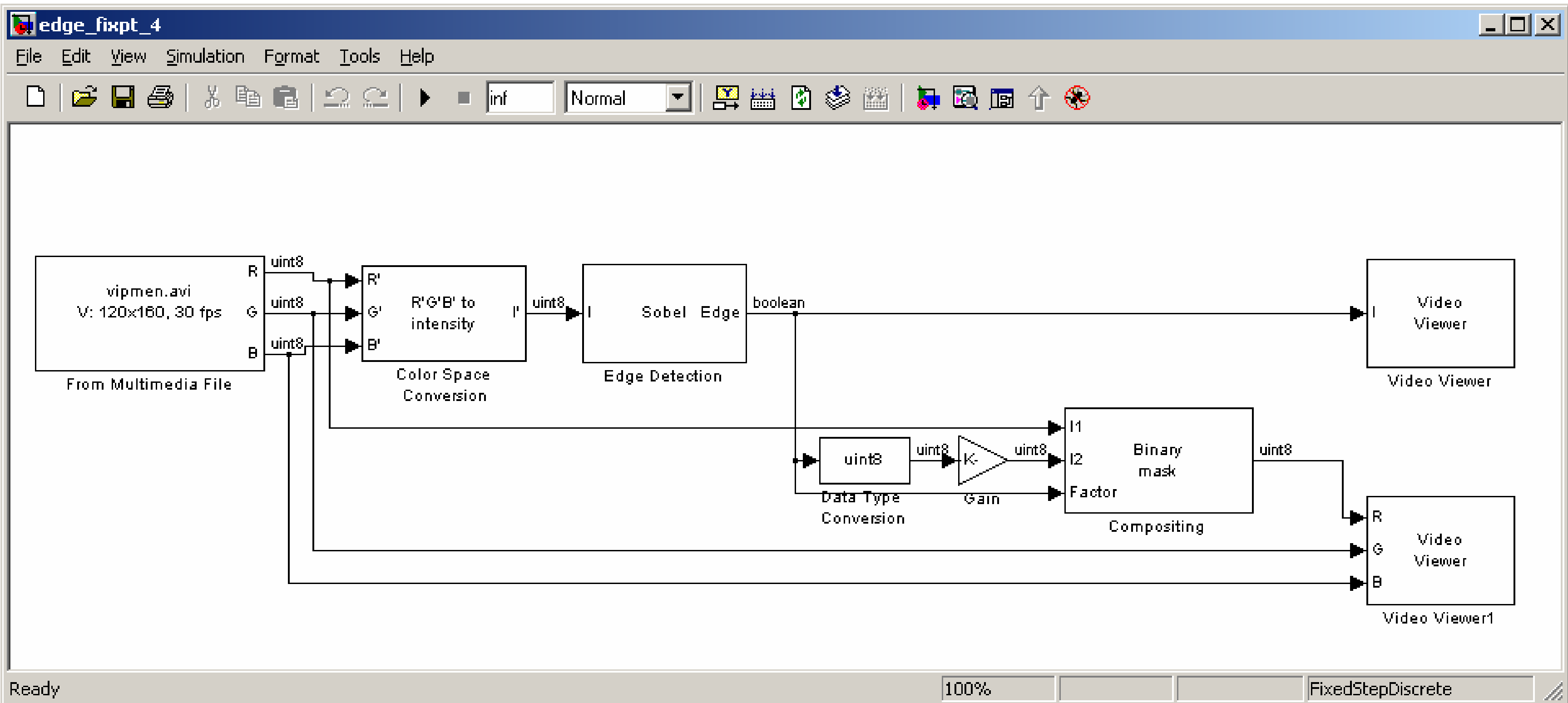
# Configuring Parameters



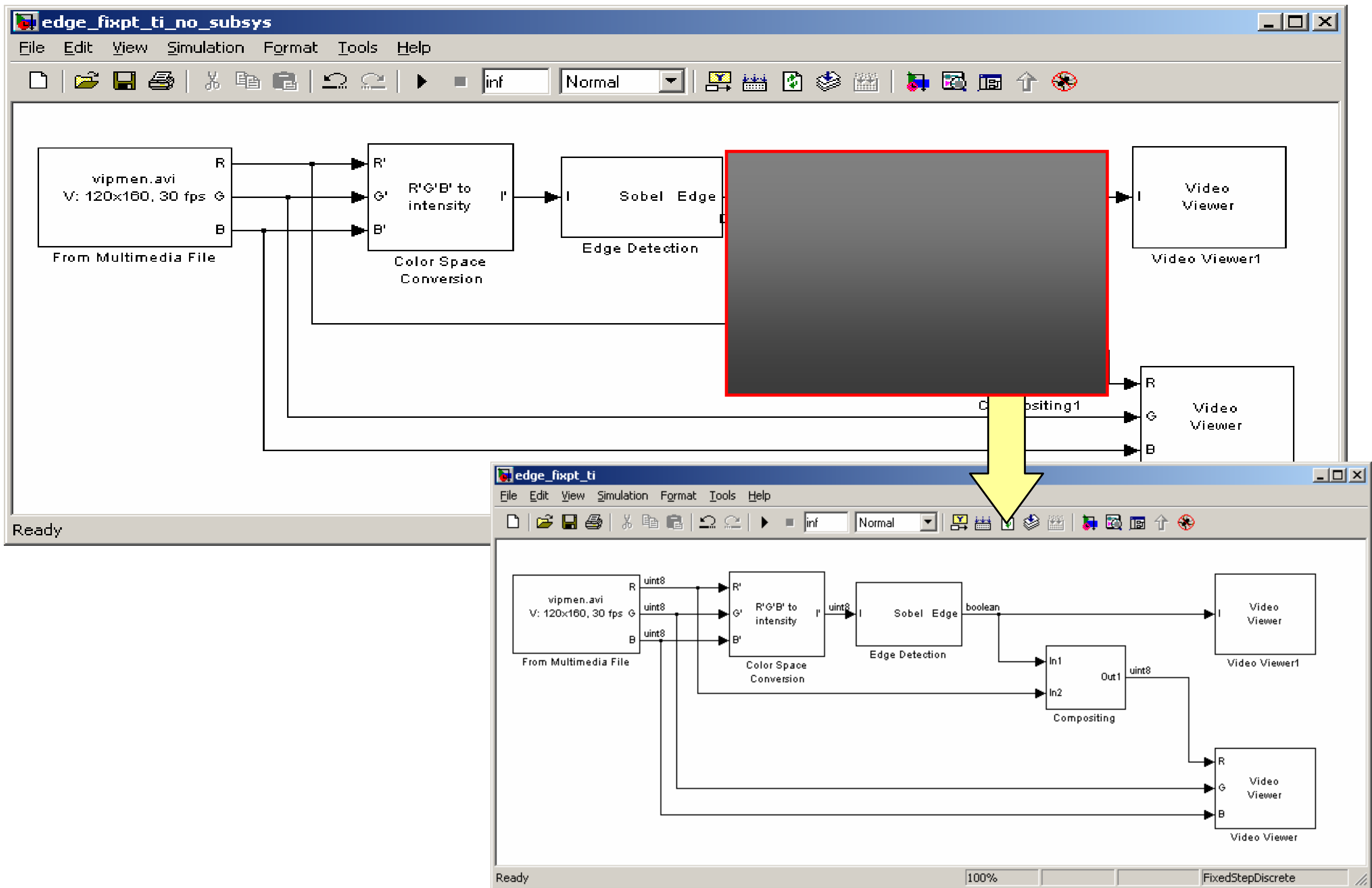

Choice of threshold computation

Turn on Edge thinning

# Overlay Original and Detected Edge



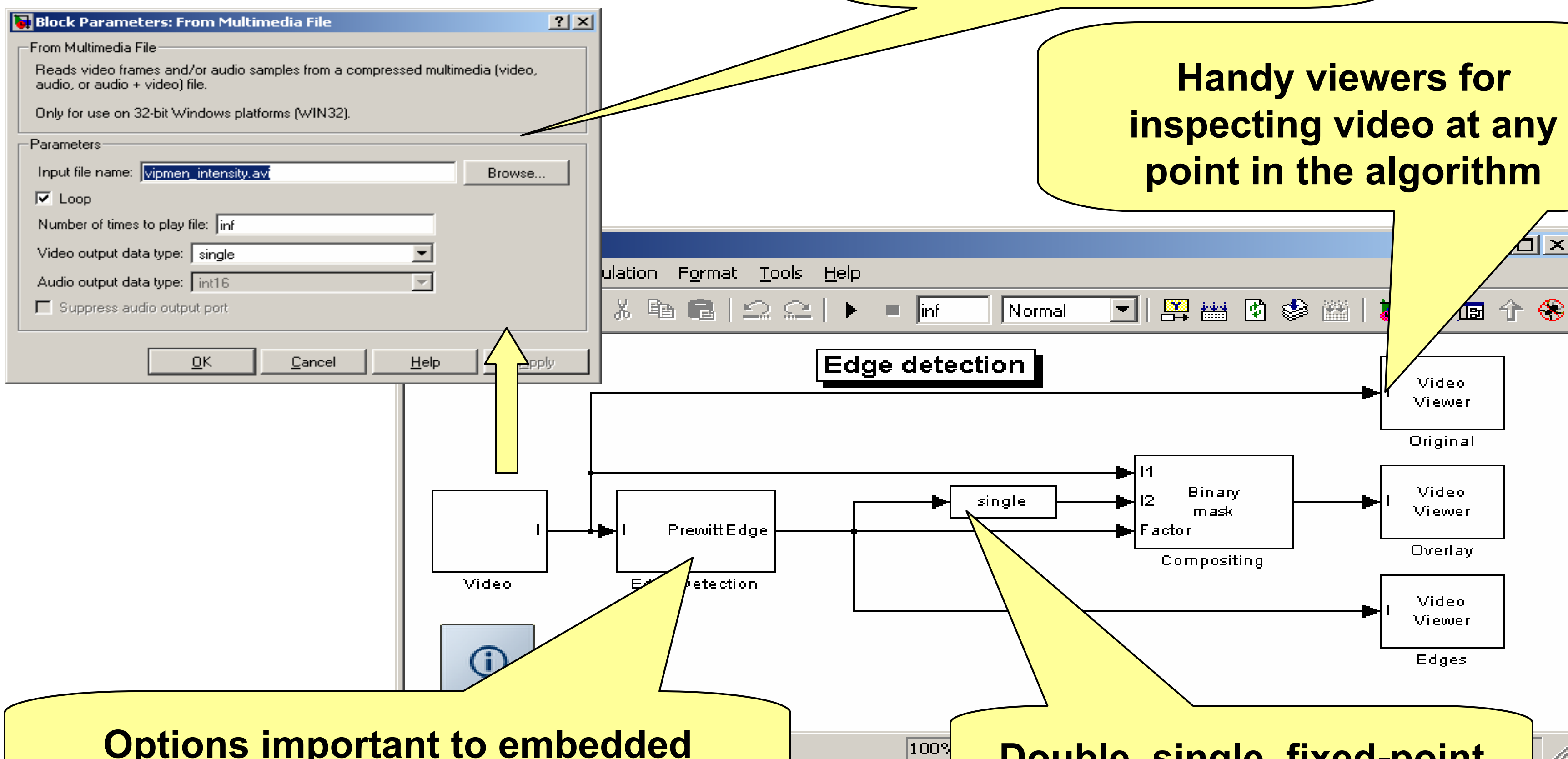
# Integrating the Final System



# Model Summary

Easy to import streaming video into the simulation

Handy viewers for inspecting video at any point in the algorithm

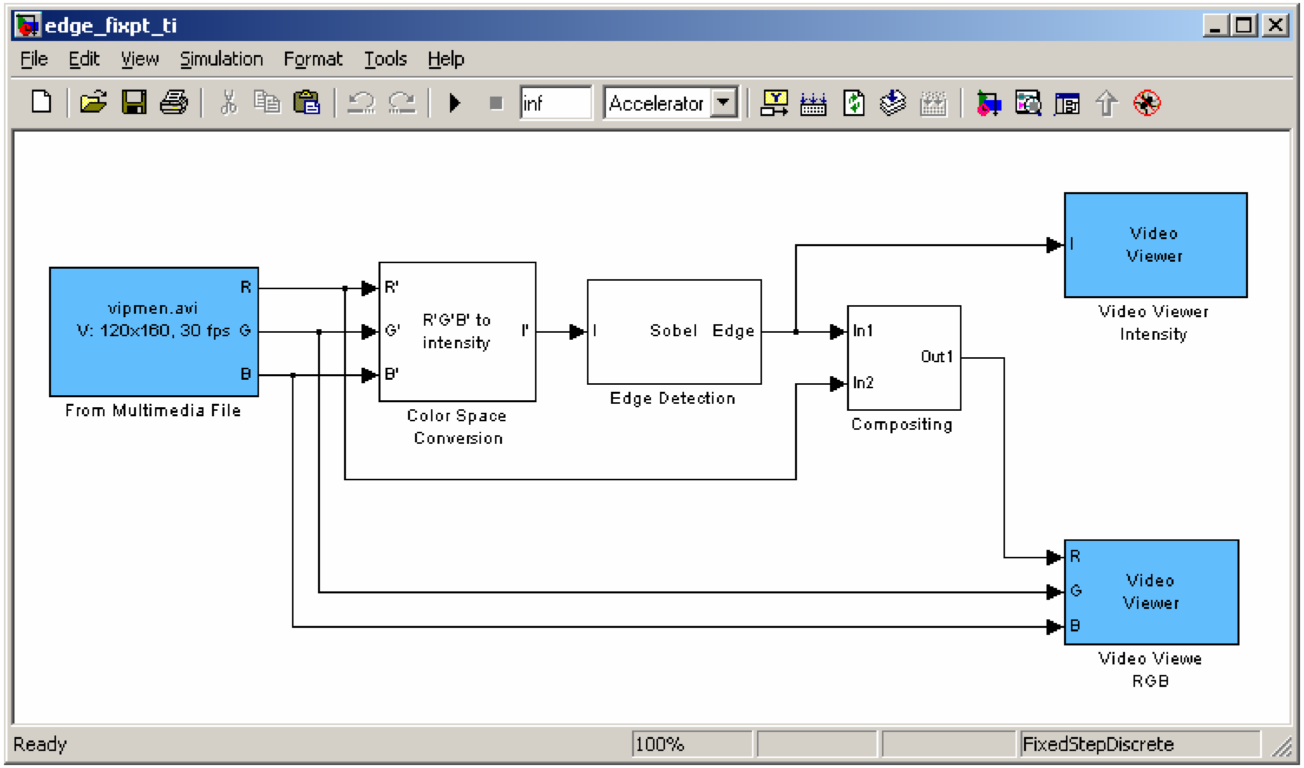



Options important to embedded system designers

Double, single, fixed-point

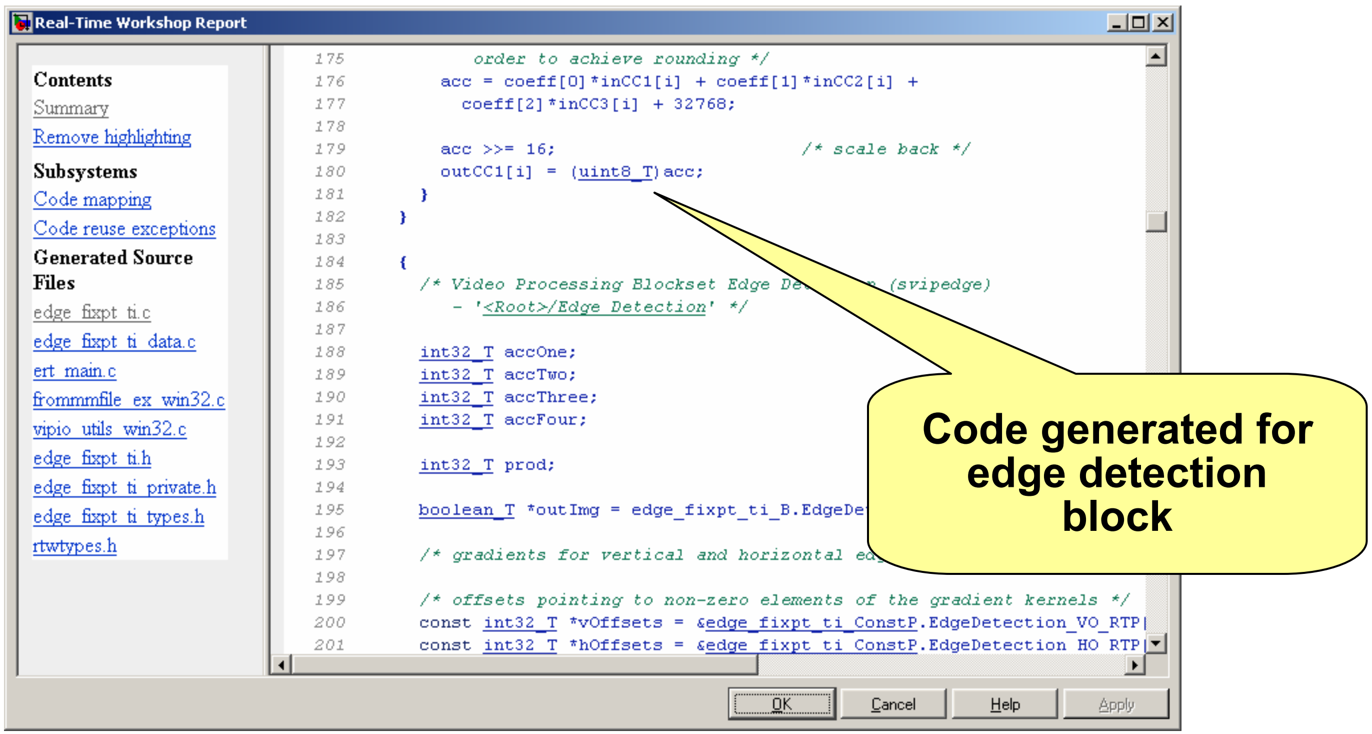
# Implementation of the
# Edge Detection System on TI DSP

# Generating Target-Independent Code

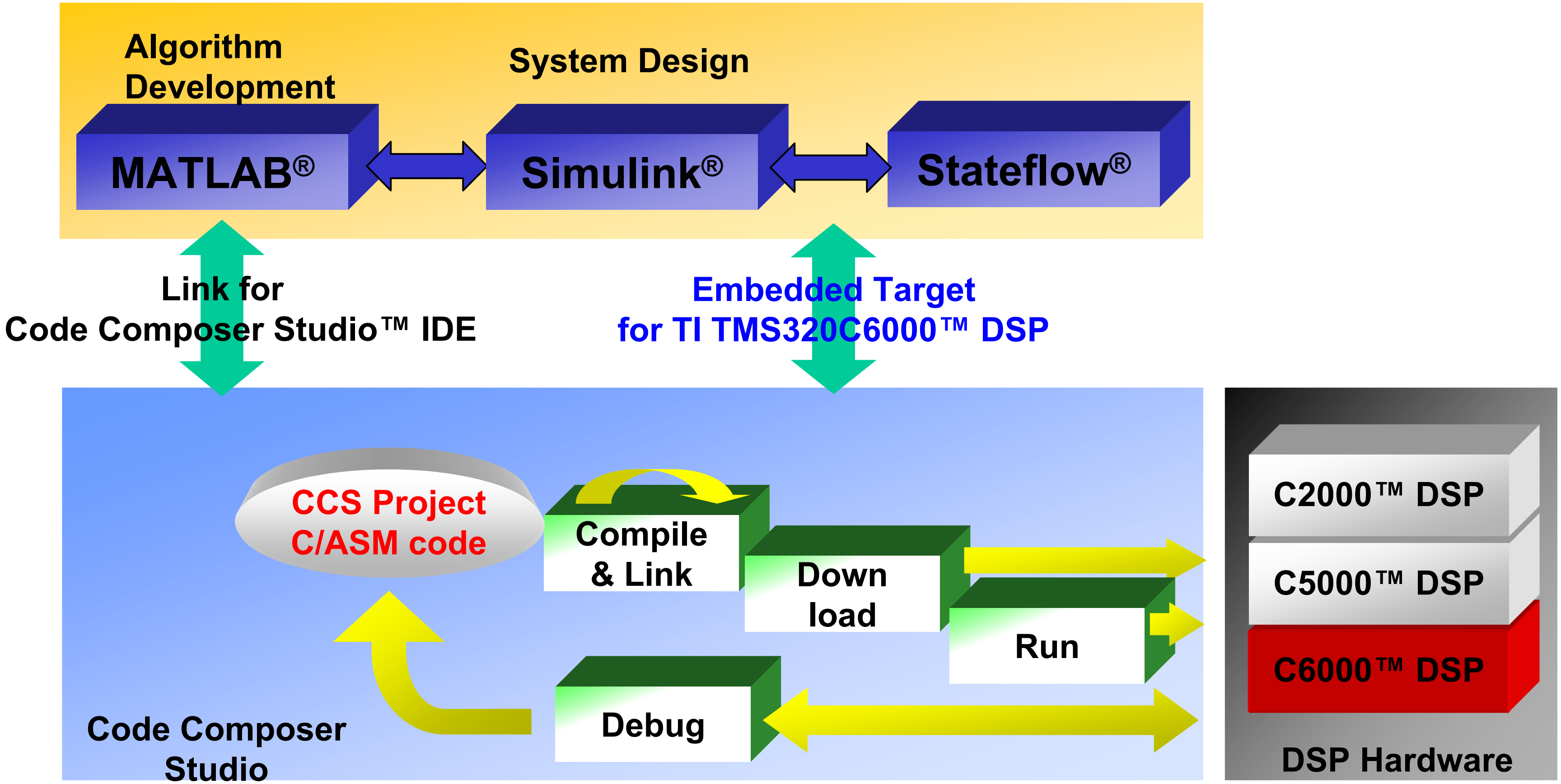



Demo

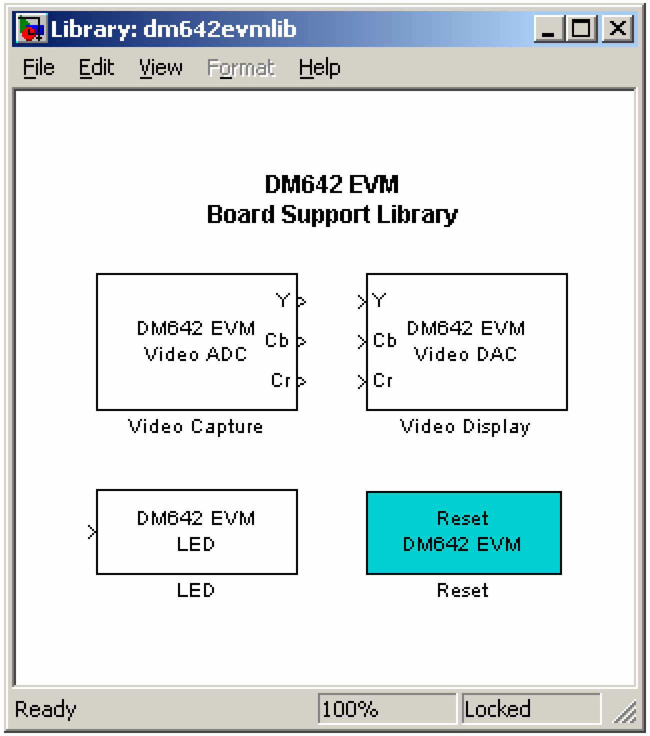
# Target-Independent Code (continued)

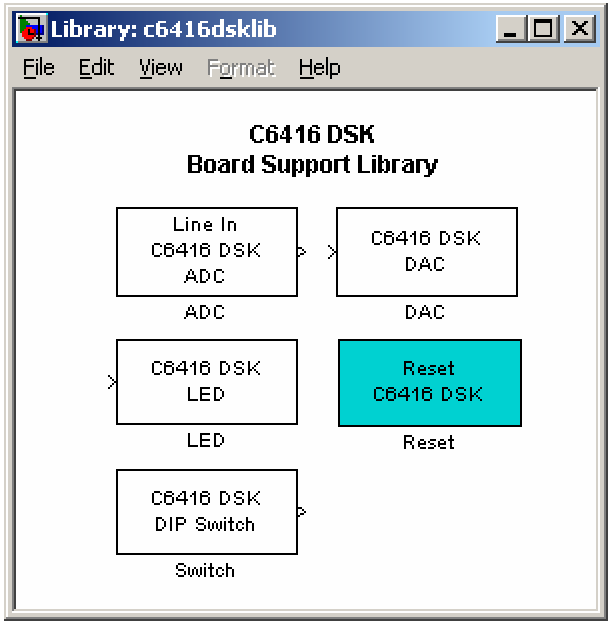


# Embedded Target for TI C6000

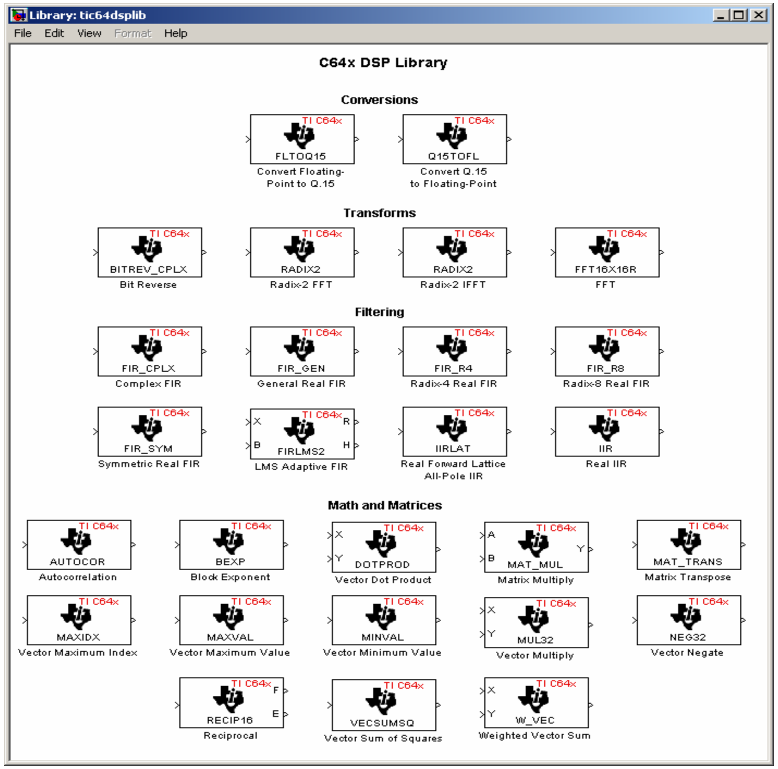


# Tour of Device-Driver Libraries

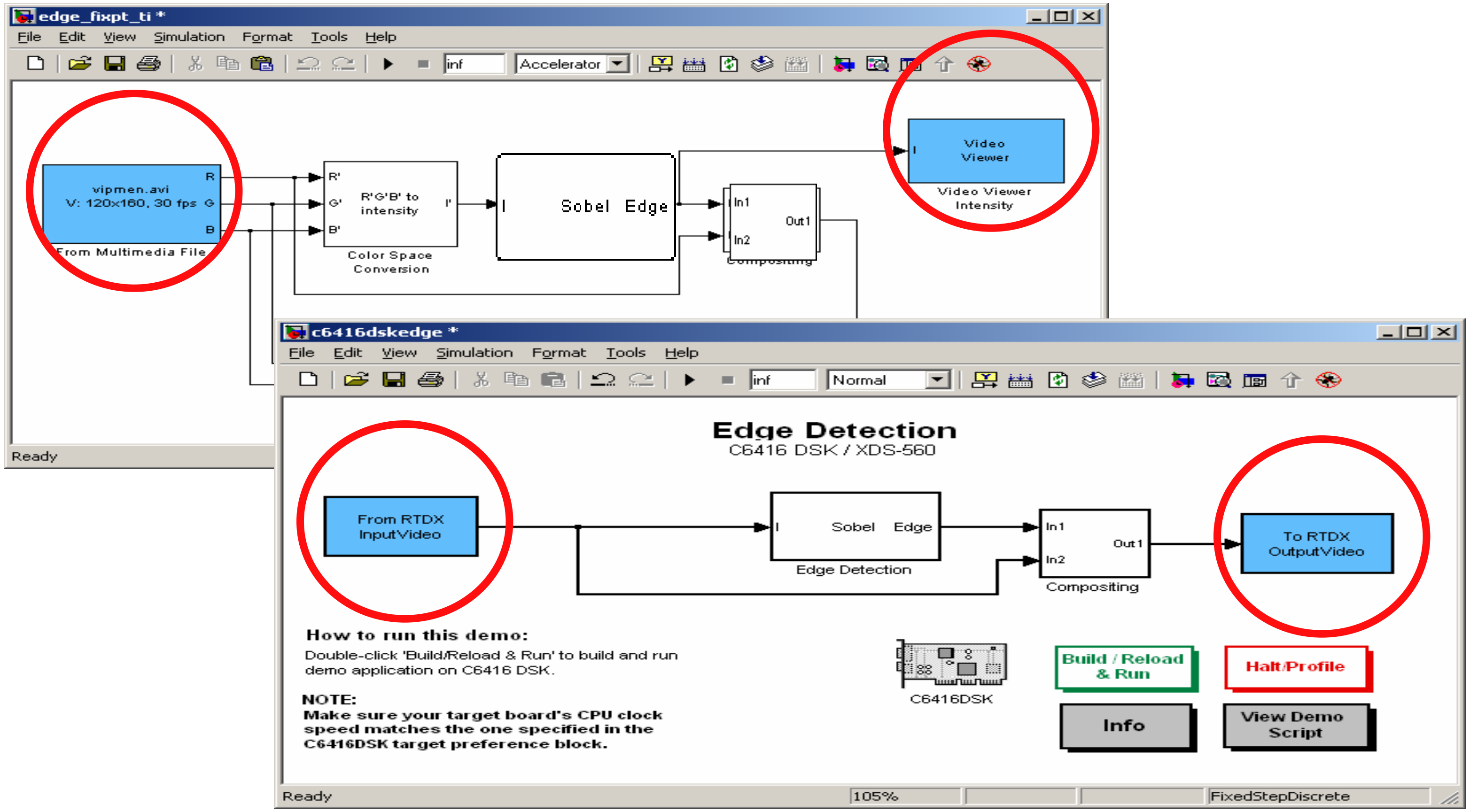


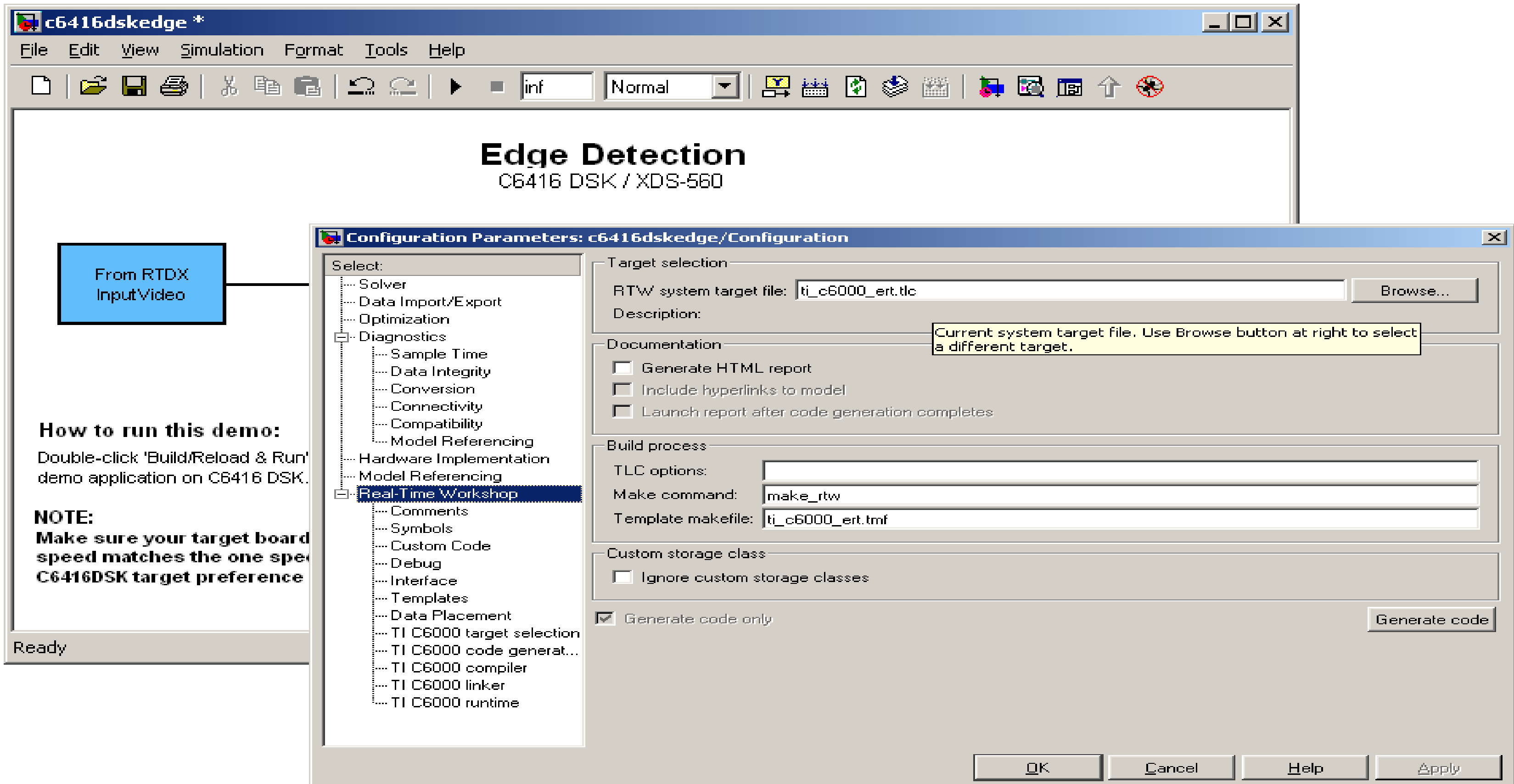
# Optimized Block Libraries



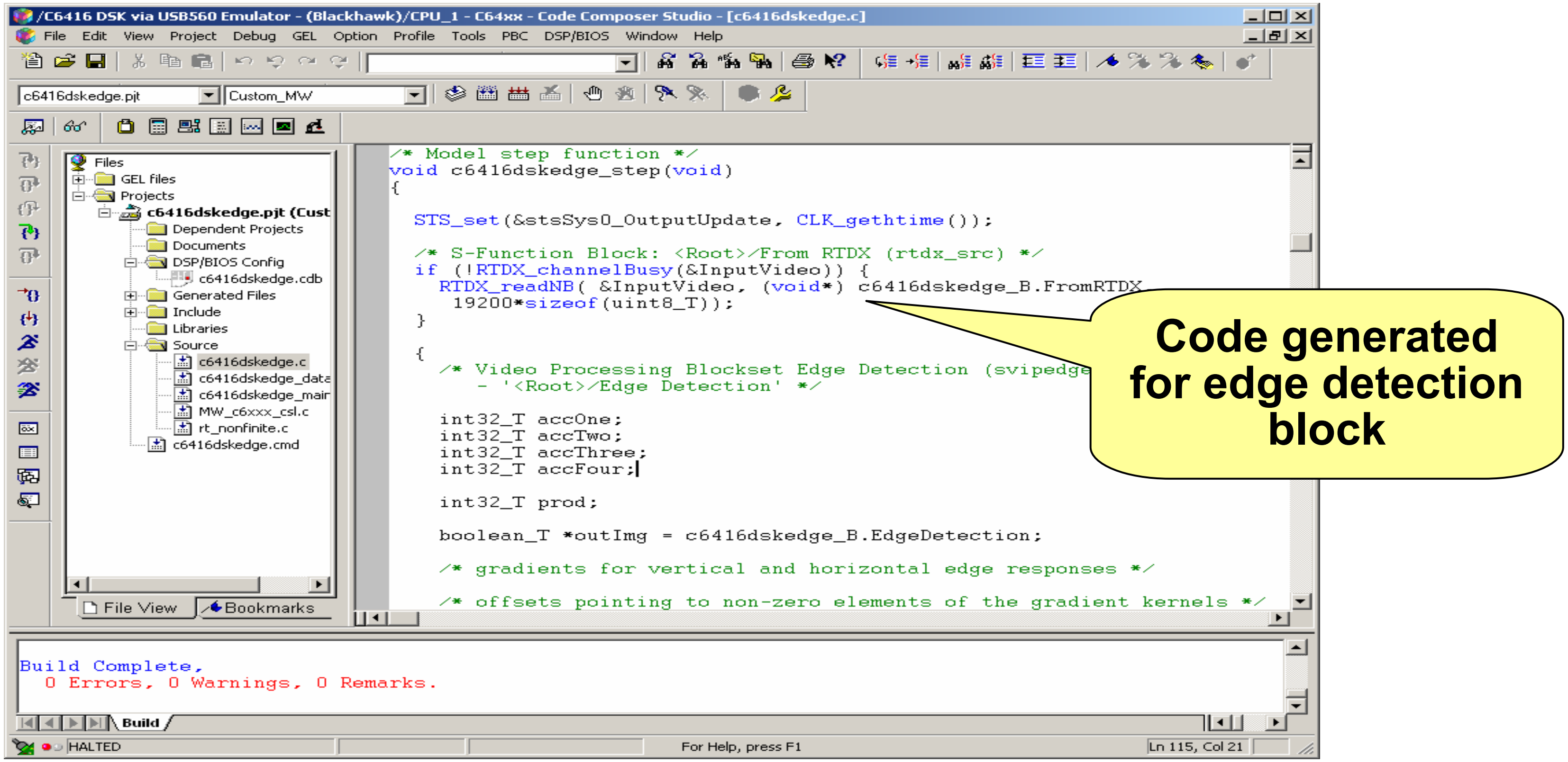
# Creating Target-Specific Model



# Generating Code for Target



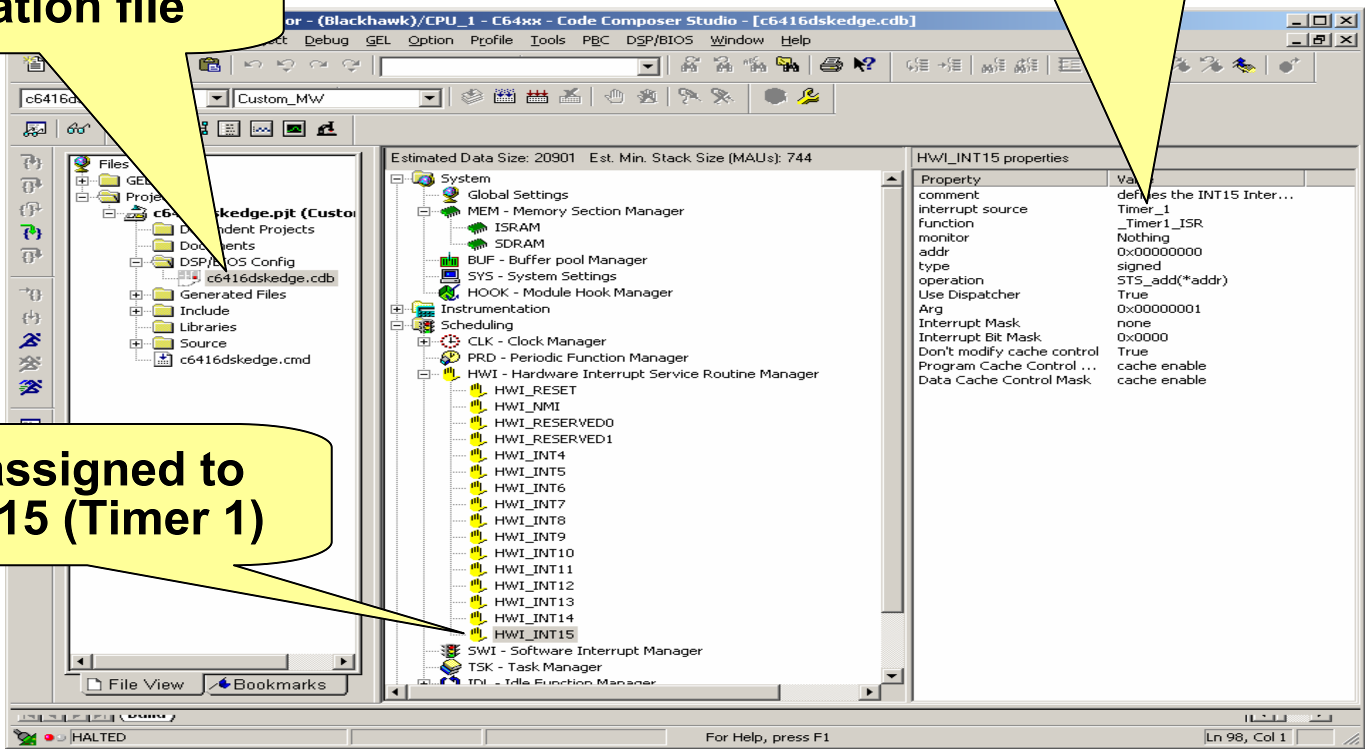
# Analyzing Generated Code



# Taking a Closer Look

Model base rate is tied to Timer 1 INT

Generated DSP/BIOS configuration file

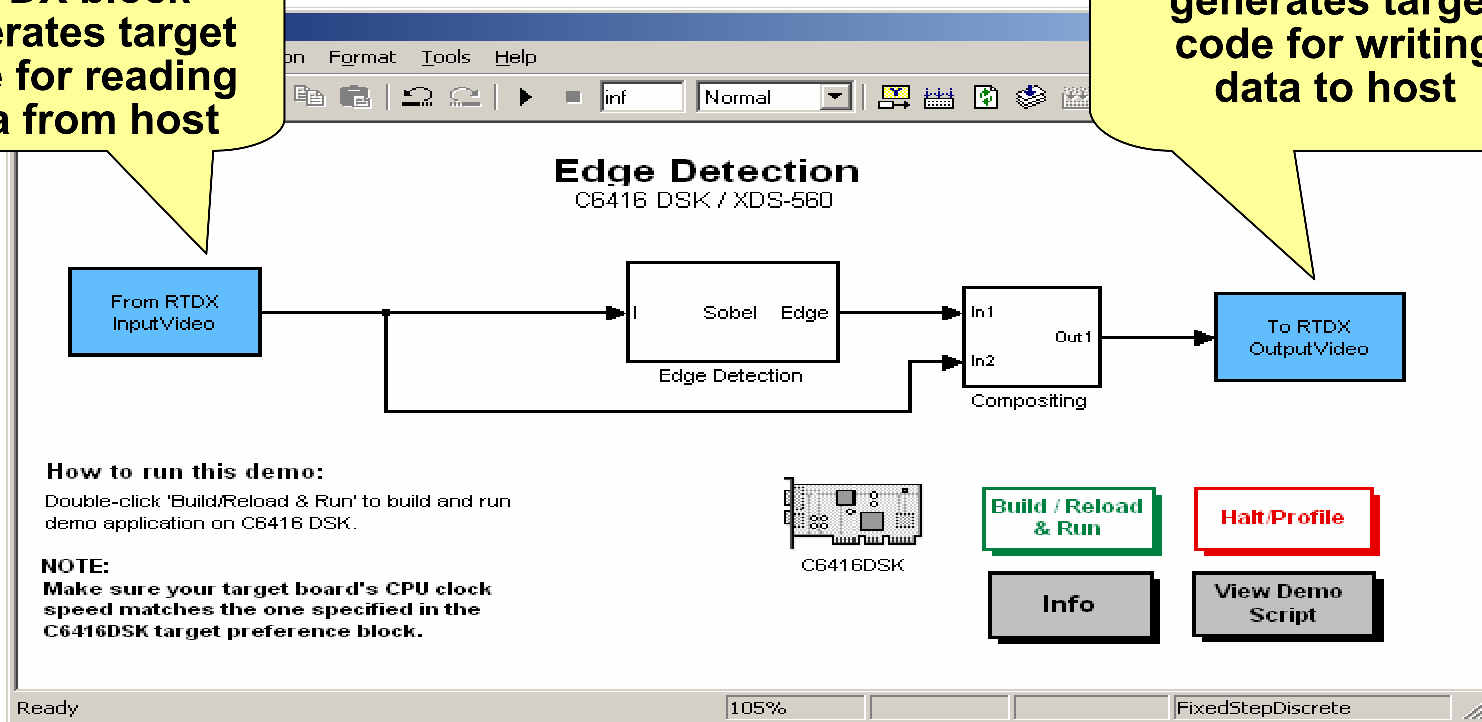



ISR is assigned to HW_INT15 (Timer 1)

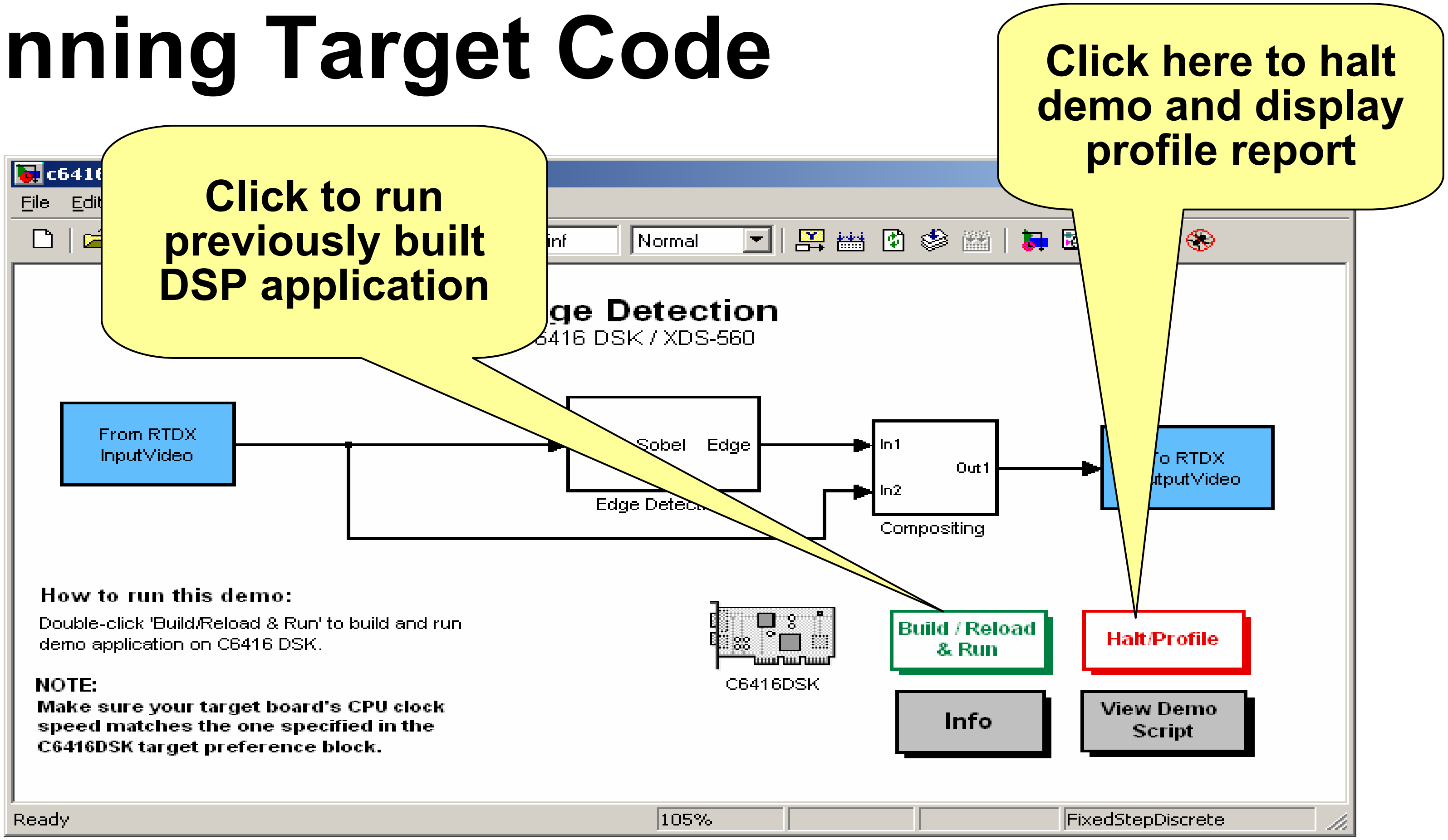
# Verifying Target Code

RTDX block generates target code for reading data from host

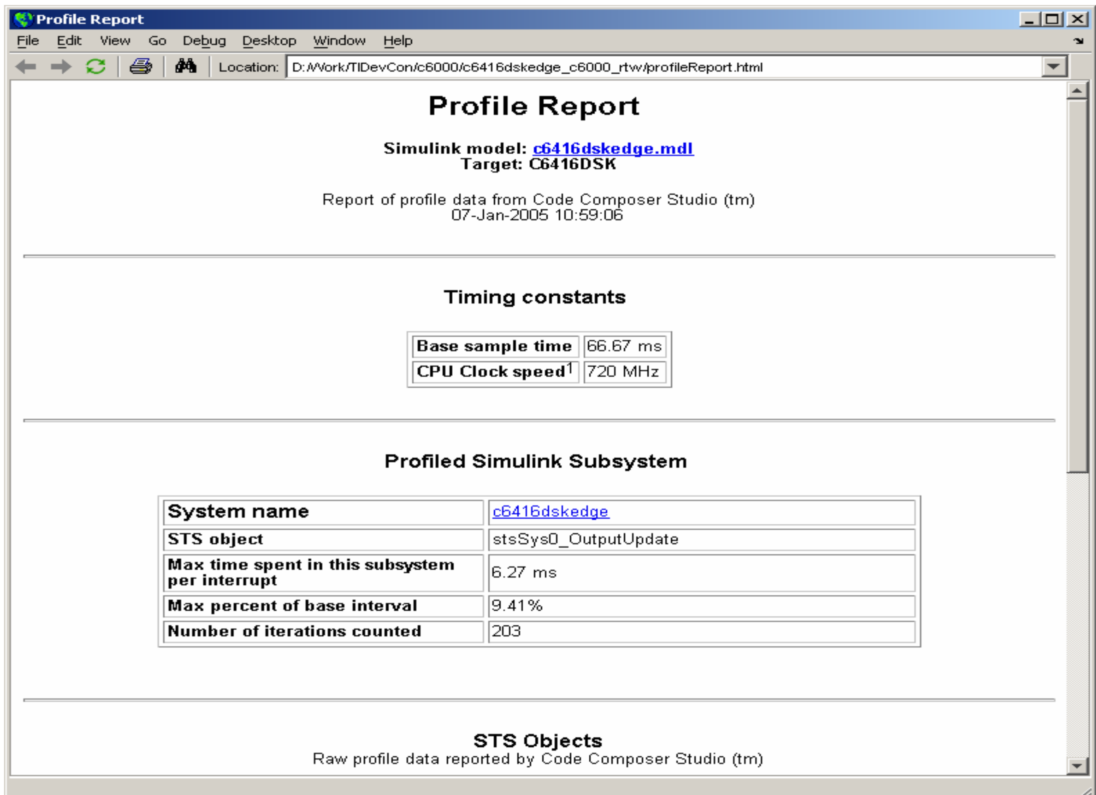
RTDX block generates target code for writing data to host



# Running Target Code



# Profiling Real-time Execution


Profile Report

File Edit View Go Debug Desktop Window Help

Location: D:/Work/TIDevCon/c6000/c6416dskedge_c6000_rtw/profileReport.html

## Profile Report

**Simulink model: c6416dskedge.mdl**
**Target: C6416DSK**

Report of profile data from Code Composer Studio (tm)
07-Jan-2005 10:59:06

### Timing constants

| | |
|---|---|
| **Base sample time** | 66.67 ms |
| **CPU Clock speed**[1] | 720 MHz |

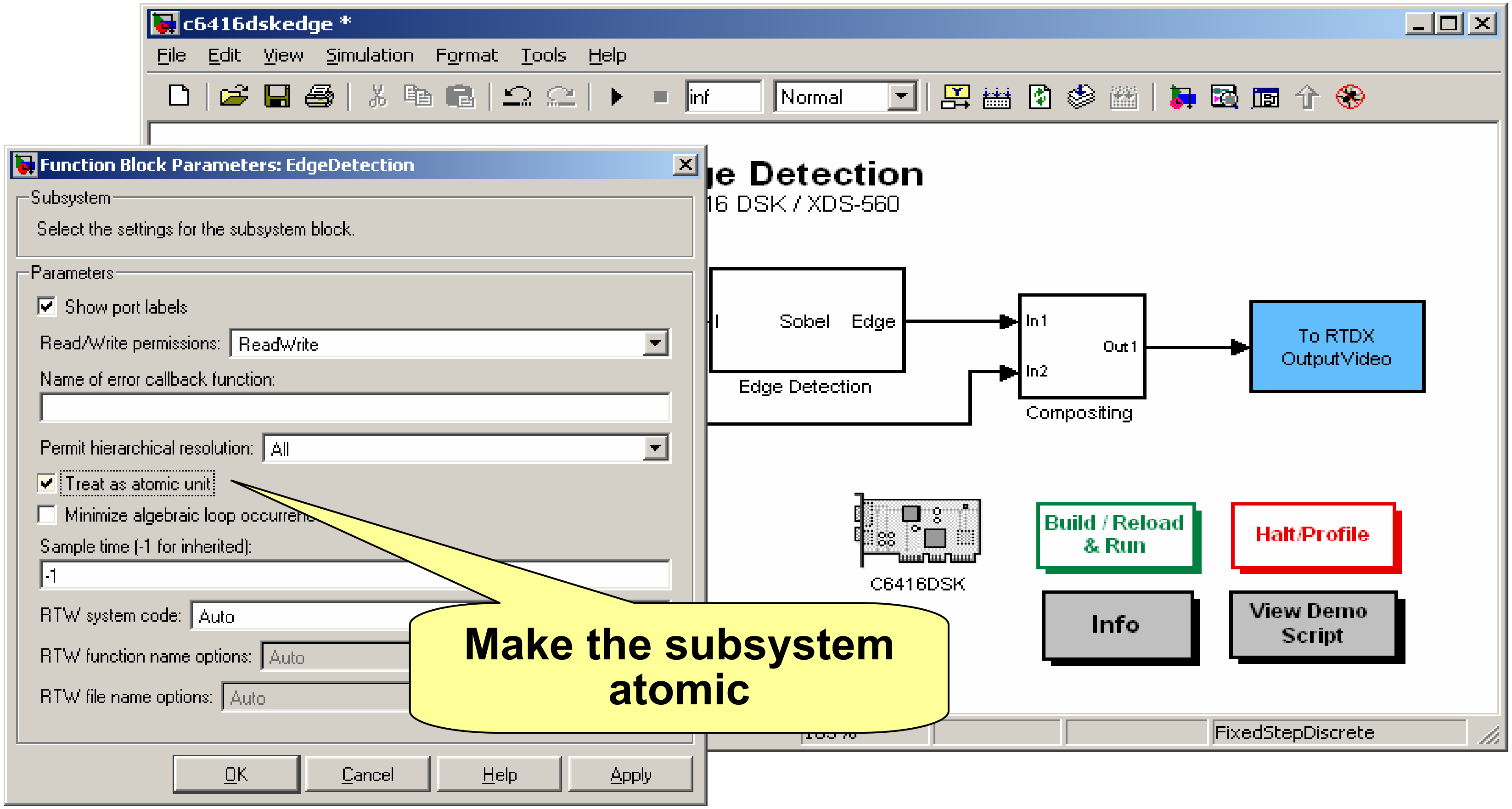
### Profiled Simulink Subsystem

| | |
|---|---|
| **System name** | c6416dskedge |
| **STS object** | stsSys0_OutputUpdate |
| **Max time spent in this subsystem per interrupt** | 6.27 ms |
| **Max percent of base interval** | 9.41% |
| **Number of iterations counted** | 203 |

### STS Objects

Raw profile data reported by Code Composer Studio (tm)

# Getting Further Insight

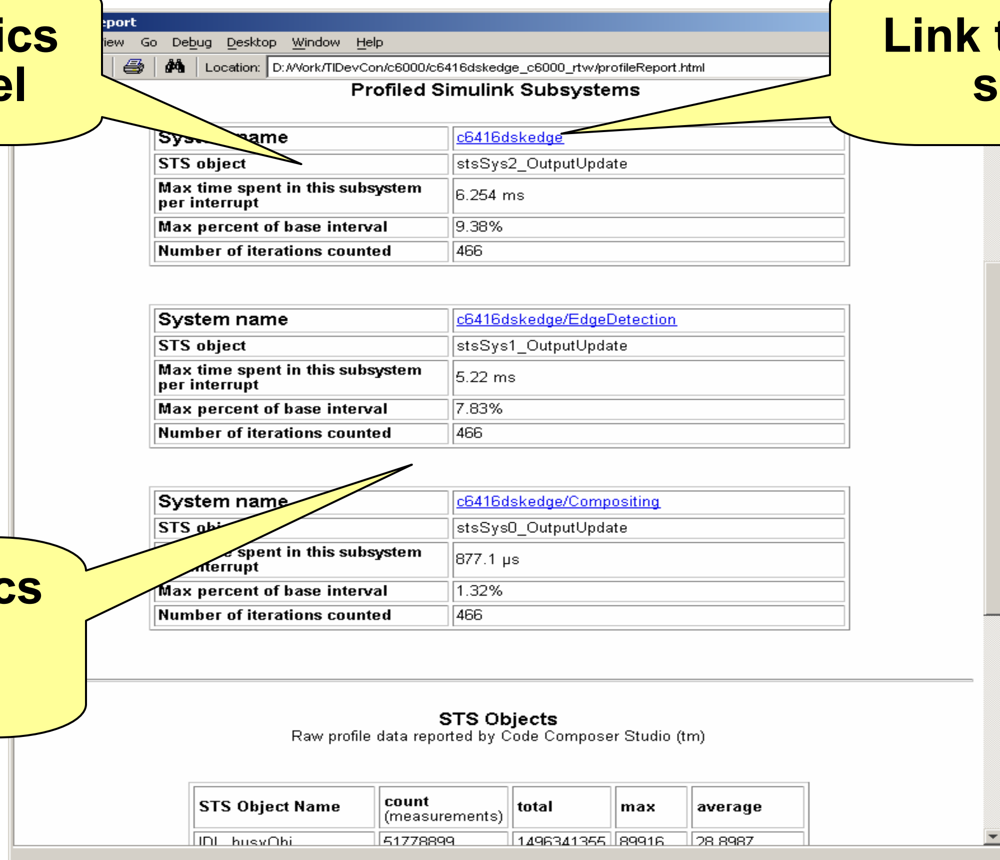


# Examining Profile Report

Execution statistics for overall model

Link to the profiled subsystem

Execution statistics for atomic subsystems



D:/Work/TIDevCon/c6000/c6416dskedge_c6000_rtw/profileReport.html

**Profiled Simulink Subsystems**

| | |
|---|---|
| System name | c6416dskedge |
| STS object | stsSys2_OutputUpdate |
| Max time spent in this subsystem per interrupt | 6.254 ms |
| Max percent of base interval | 9.38% |
| Number of iterations counted | 466 |

| | |
|---|---|
| System name | c6416dskedge/EdgeDetection |
| STS object | stsSys1_OutputUpdate |
| Max time spent in this subsystem per interrupt | 5.22 ms |
| Max percent of base interval | 7.83% |
| Number of iterations counted | 466 |

| | |
|---|---|
| System name | c6416dskedge/Compositing |
| STS object | stsSys0_OutputUpdate |
| Max time spent in this subsystem per interrupt | 877.1 μs |
| Max percent of base interval | 1.32% |
| Number of iterations counted | 466 |

**STS Objects**

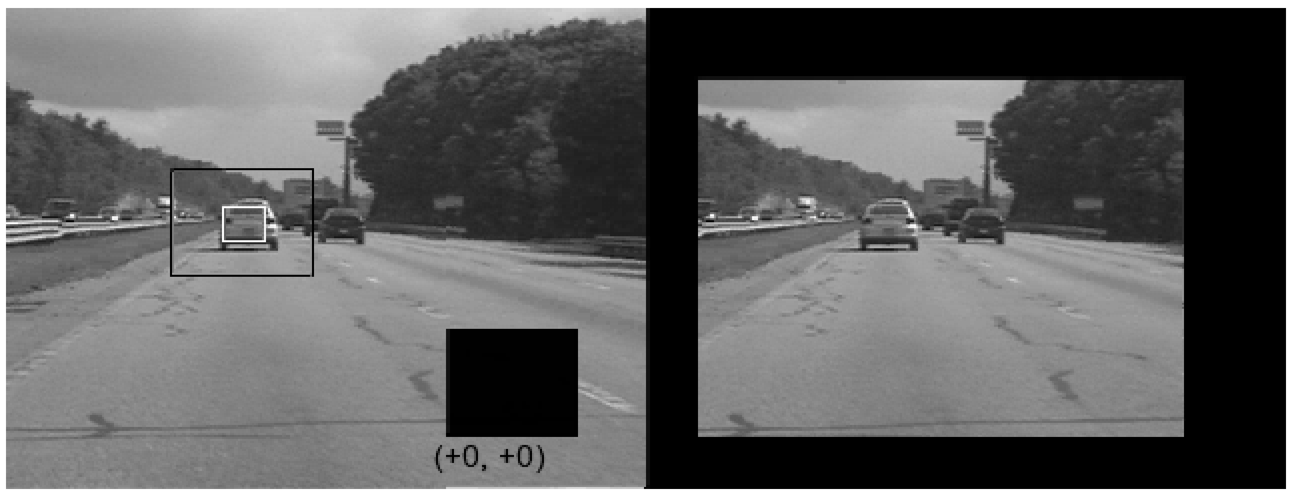
Raw profile data reported by Code Composer Studio (tm)

| STS Object Name | count (measurements) | total | max | average |
|---|---|---|---|---|
| IDL_busyObj | 51778899 | 1496341355 | 89916 | 28.8987 |

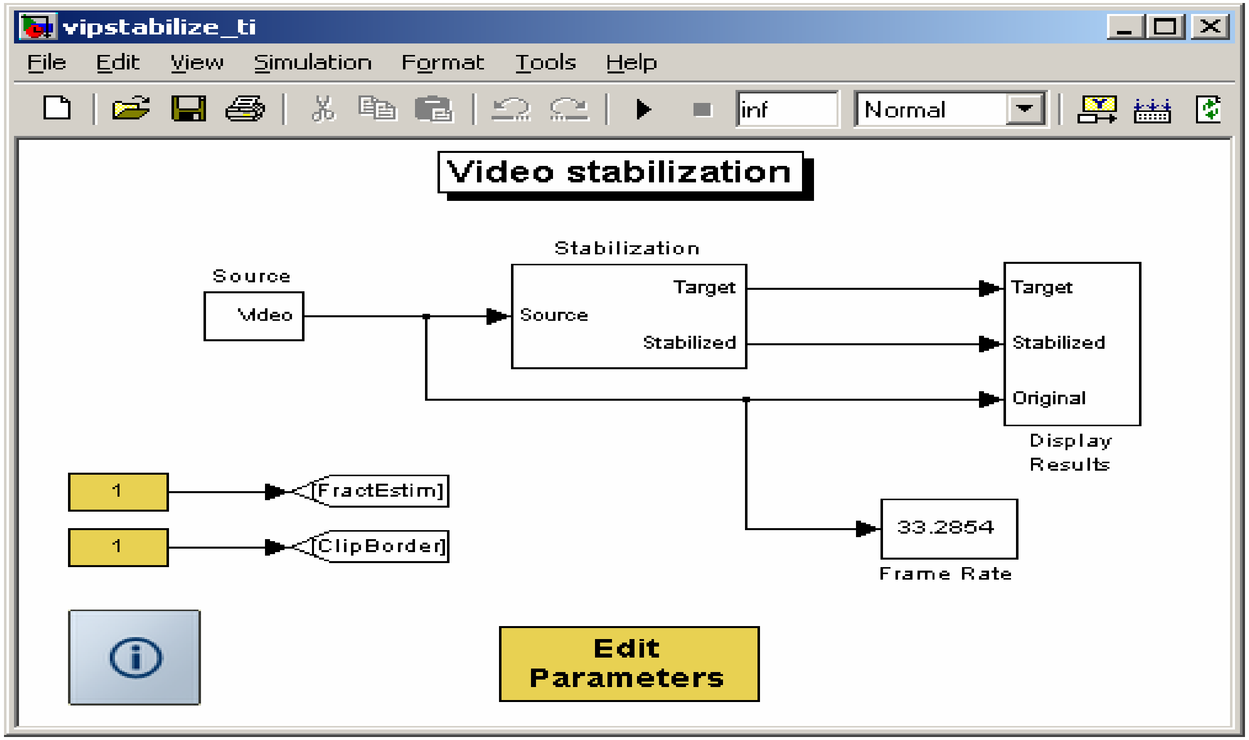
# Advanced Video Applications

# Video Stabilization

Track **and** remove **motion in a video sequence**

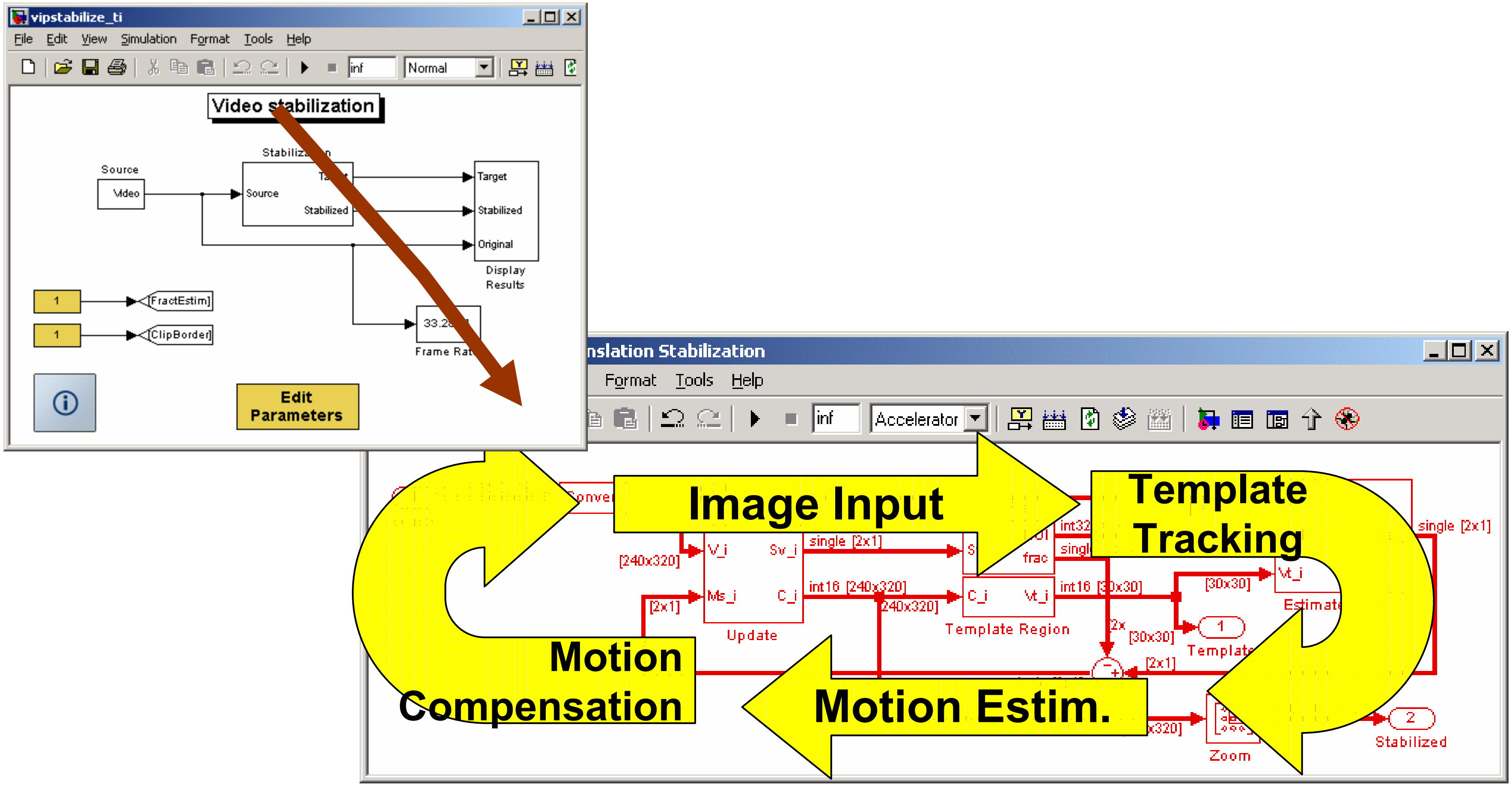


# Video Stabilization (continued)




Demo

# Model Overview

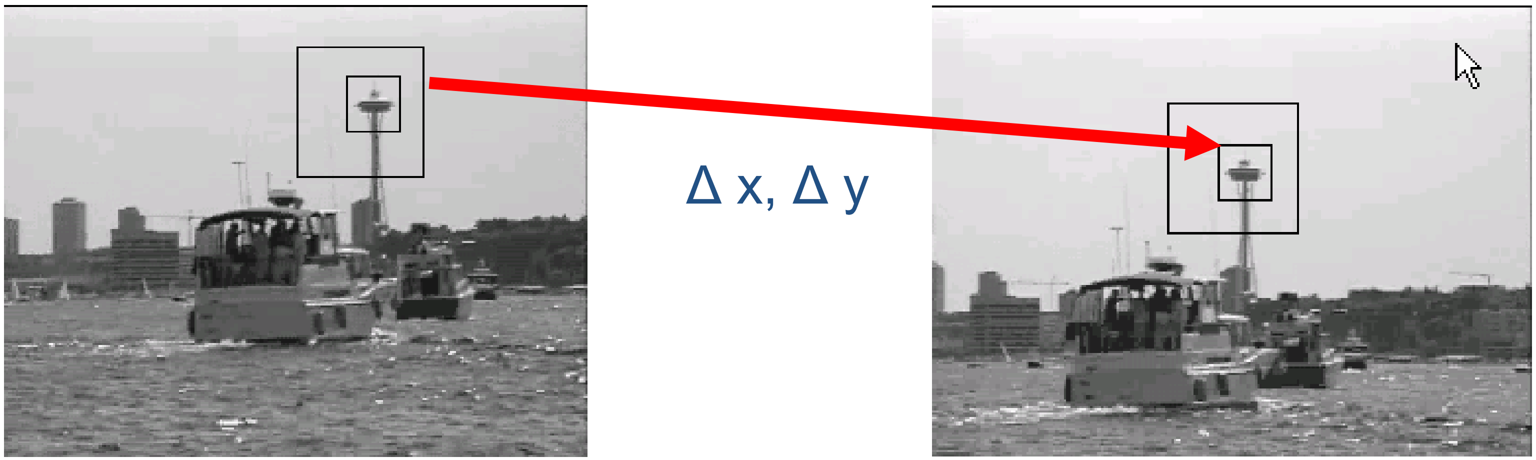


# Algorithm Overview

- **Steps to Stabilize Motion**
  - Estimate target position from template
  - Compute inter-frame motion
  - Compensate motion
  - Update matching template

Frame (n-1):
Origin=(100,100)

Frame n:
Origin=(80,80)



# Estimate Target Position

**(Computationally expensive)**

$$E = \min_{S} \sum_{T} |V(x,y) - T(x - x_o, y - y_o)|$$

Motion estimate $(x_o, y_o)$

Video Image, V

**Computational Cost**

= 2 * $N^2$ (over T: NxN)

* $L^2$ (over S: LxL)

* fps

Ex:

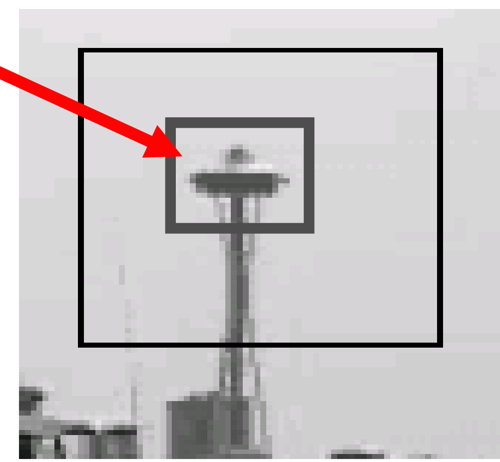
32x32 template

64x64 search

30 frames/sec

>250 million adds

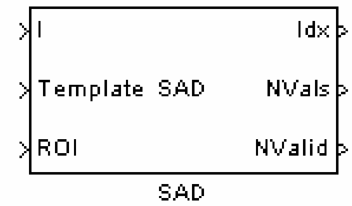
***per second***



# Search for Target Position

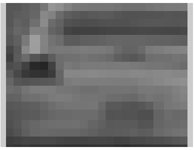
## Sum of Absolute Differences (SAD)

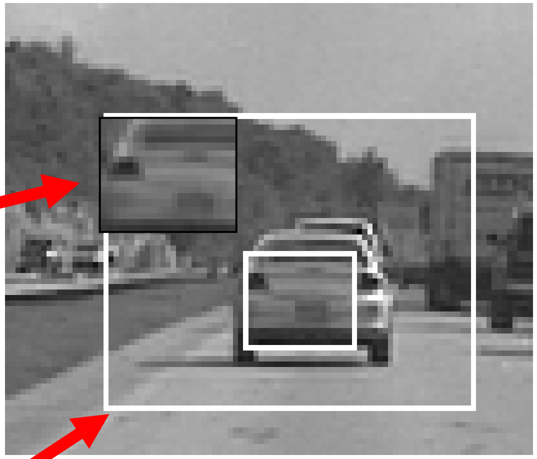
$$E = \min_{S} \sum_{T} \left| V(x,y) - T(x - x_o, y - y_o) \right|$$




Video Image, V

Template Image, T





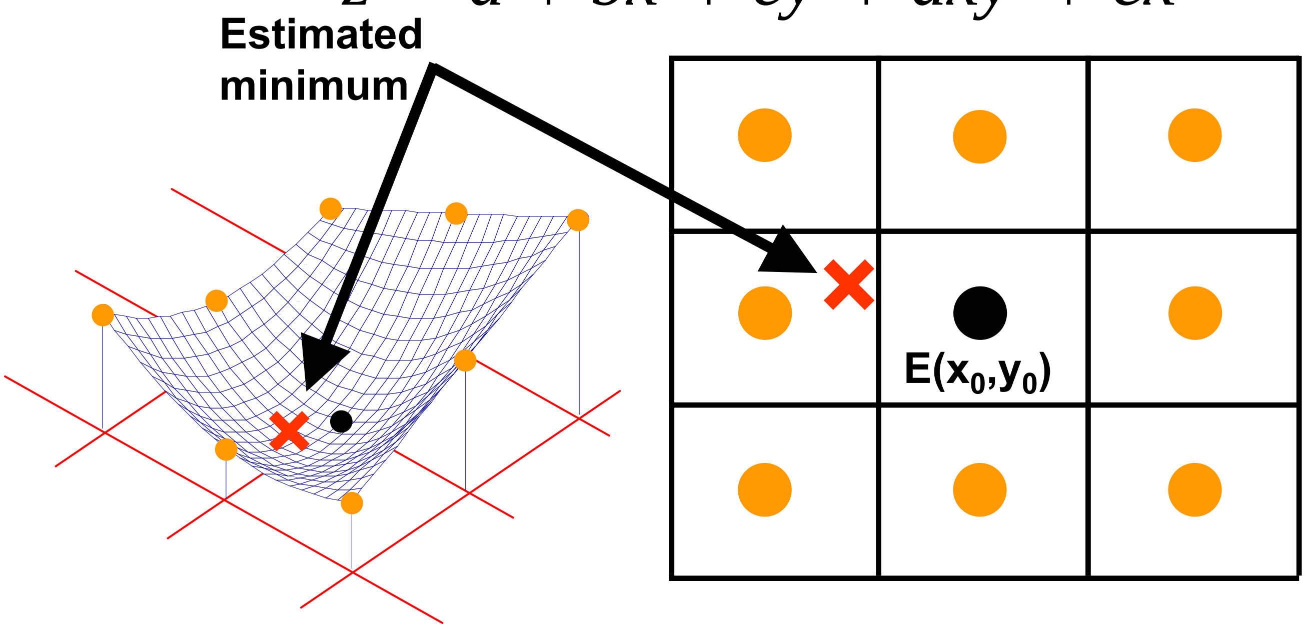
Search for best fit (minimum SAD)

# Sub-Pixel Estimation of Target Motion

**Refine coarse motion estimates**

- **Find minimum of a** quadratic surface **over 3x3 neighborhood**

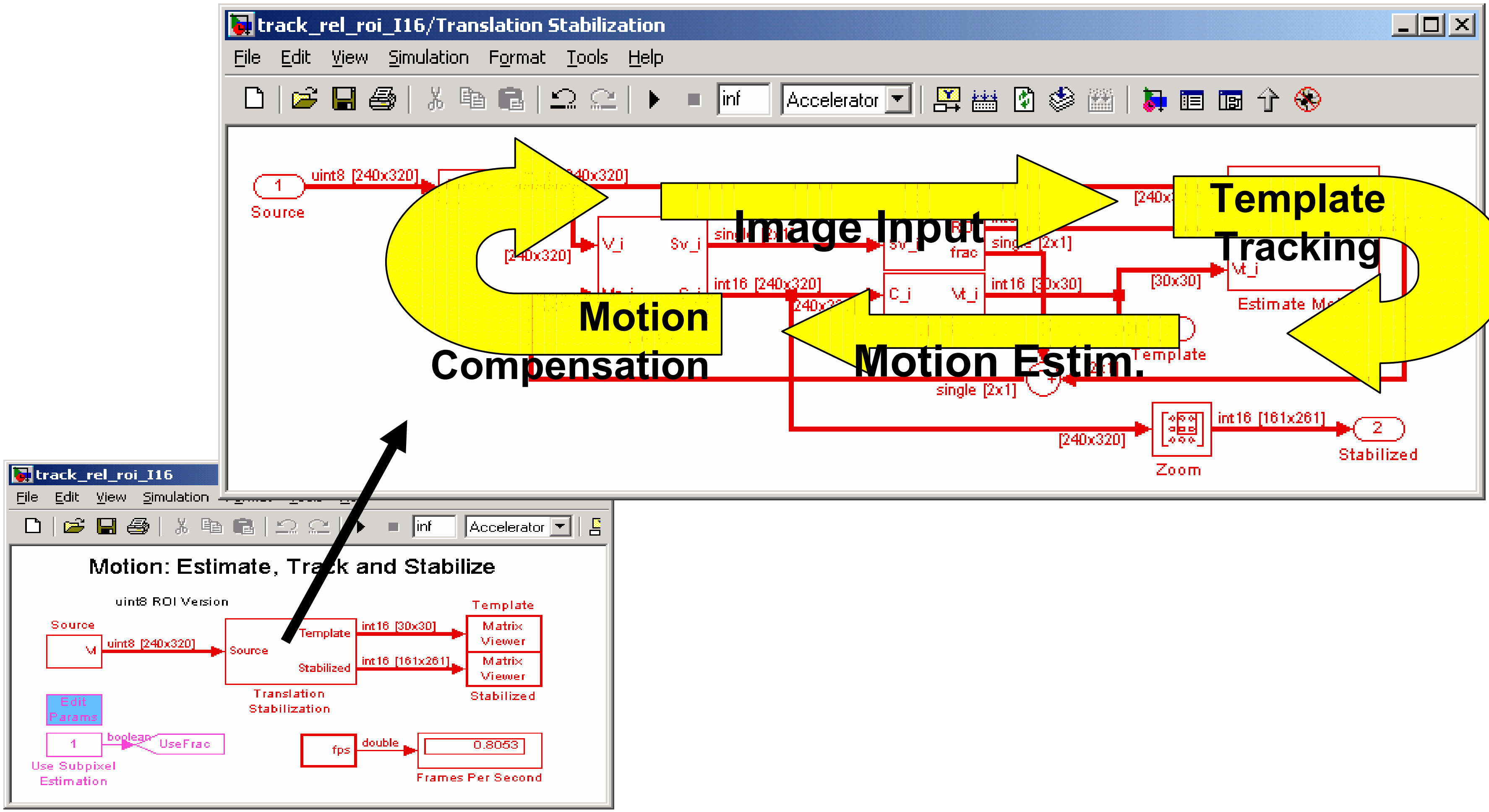
$$z = a + bx + cy + dxy + ex^2 - fy^2$$




→ **Solve Ax=B for minimum**

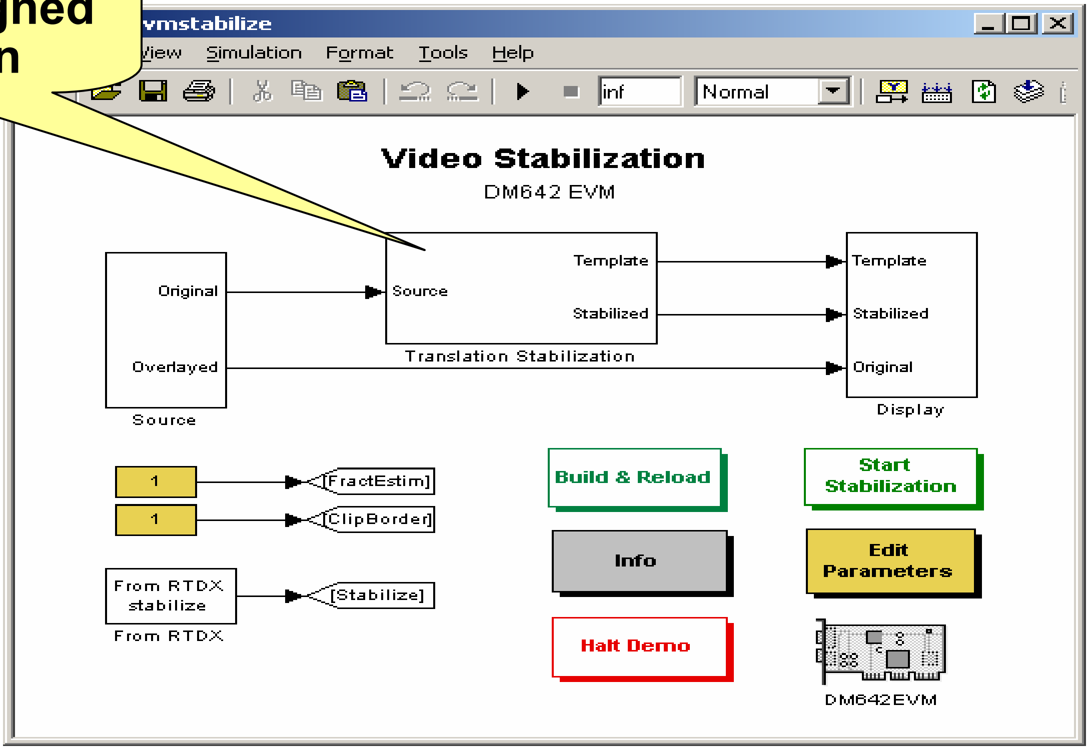
→ **Sub-pixel estimate**

# Integrating Video Stabilization System



# Video Stabilization on TI DM642 EVM

Stabilization algorithm designed in simulation



# Next Steps

# More Information

- Products
  - www.mathworks.com/dsp
- Example models available for download on MATLAB Central
  - www.mathworks.com/matlabcentral/
  - Click on “File Exchange”
- Upcoming seminars, Webinars, and more…
  - www.mathworks.com/dsp_events
- View a recorded Webinar (more than 50 available)
  - www.mathworks.com/ webinar

# Next Steps

- **Arrange for a live** online demo **and** discussion
- **Arrange for an** onsite visit **by MathWorks Applications Engineer**
- **Request an** evaluation license **and try it out**
- **Attend a MathWorks** training course
- **Contact Rob Segal, your account manager:**
  - **508-647-7615**
  - **Robert.Segal@mathworks.com**

– Thank you for your interest!