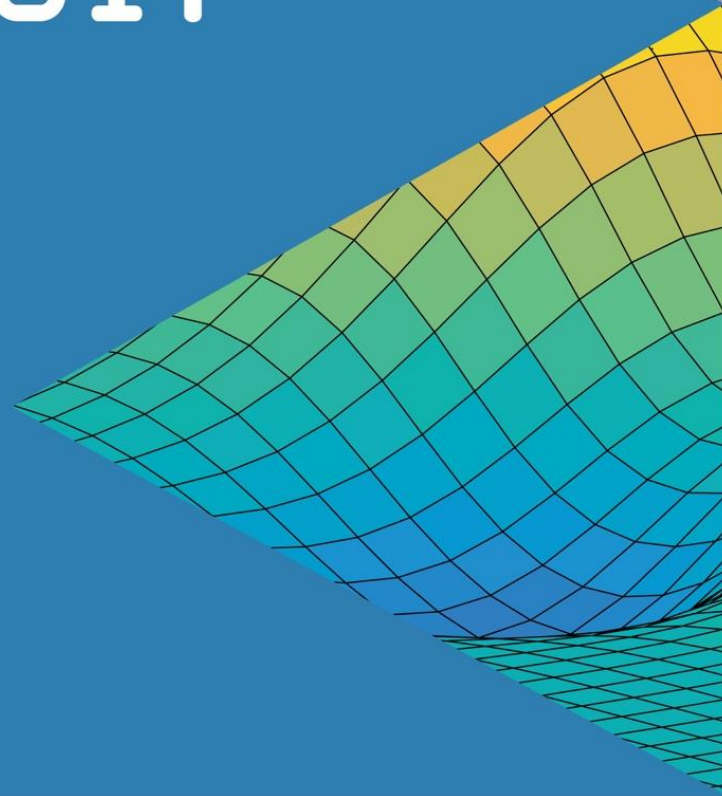


Designing and Implementing Real-Time Signal Processing Systems

MATLAB EXPO 2017

Vidya Viswanathan



Smart Systems

- What makes a system “smart”?
 - Sensors
 - Analysis based on available data
 - Networking capabilities
 - Standalone devices



Signal Processing is everywhere!

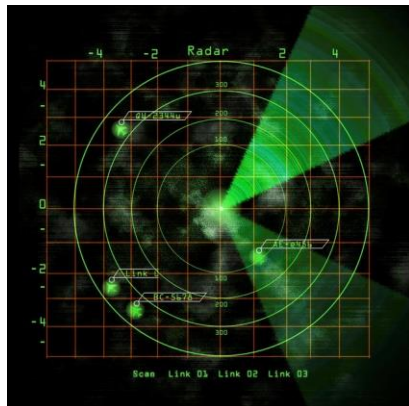
Computer Vision and
Image Processing



Audio Systems and
Consumer
Electronics



Communications



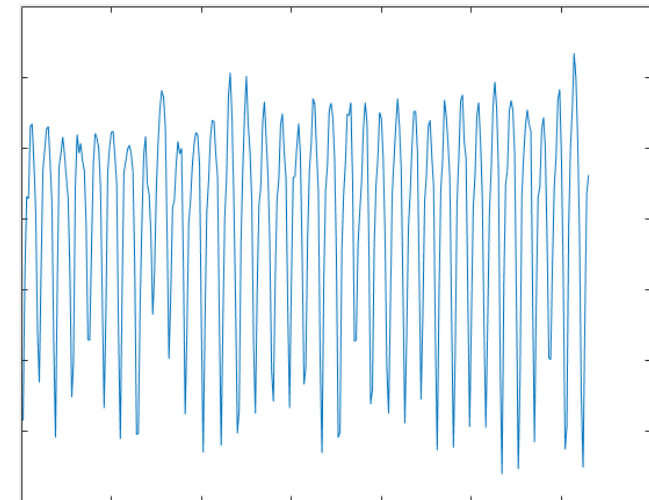
Medical Devices

Problem Statement

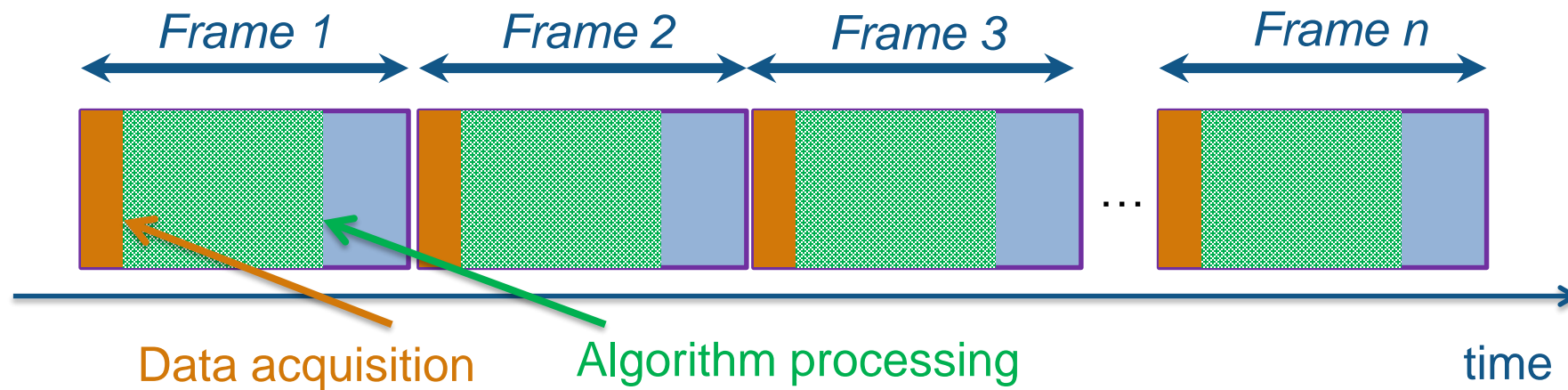
- Photoplethysmography (PPG)
 - Measuring the volumetric change during each cardiac cycle due to pumping of blood
 - Illuminate the skin and measure the variation in intensity
- What are we going to build?
 - An android app that estimates the heart rate based on optical sensors



Pulse Oximeter



Real-Time Stream Processing



As long as

$$\text{Data acquisition} + \text{Algorithm processing} \leq \text{Frame time}$$

We have

Real-time signal processing

Challenges in a Real-Time Signal Processing System Design

Framework for real-time simulations

*"I have to **process large data** and test my simulations with **streaming** signals. I need a simulation testbench that can **keep up with real-time** data."*

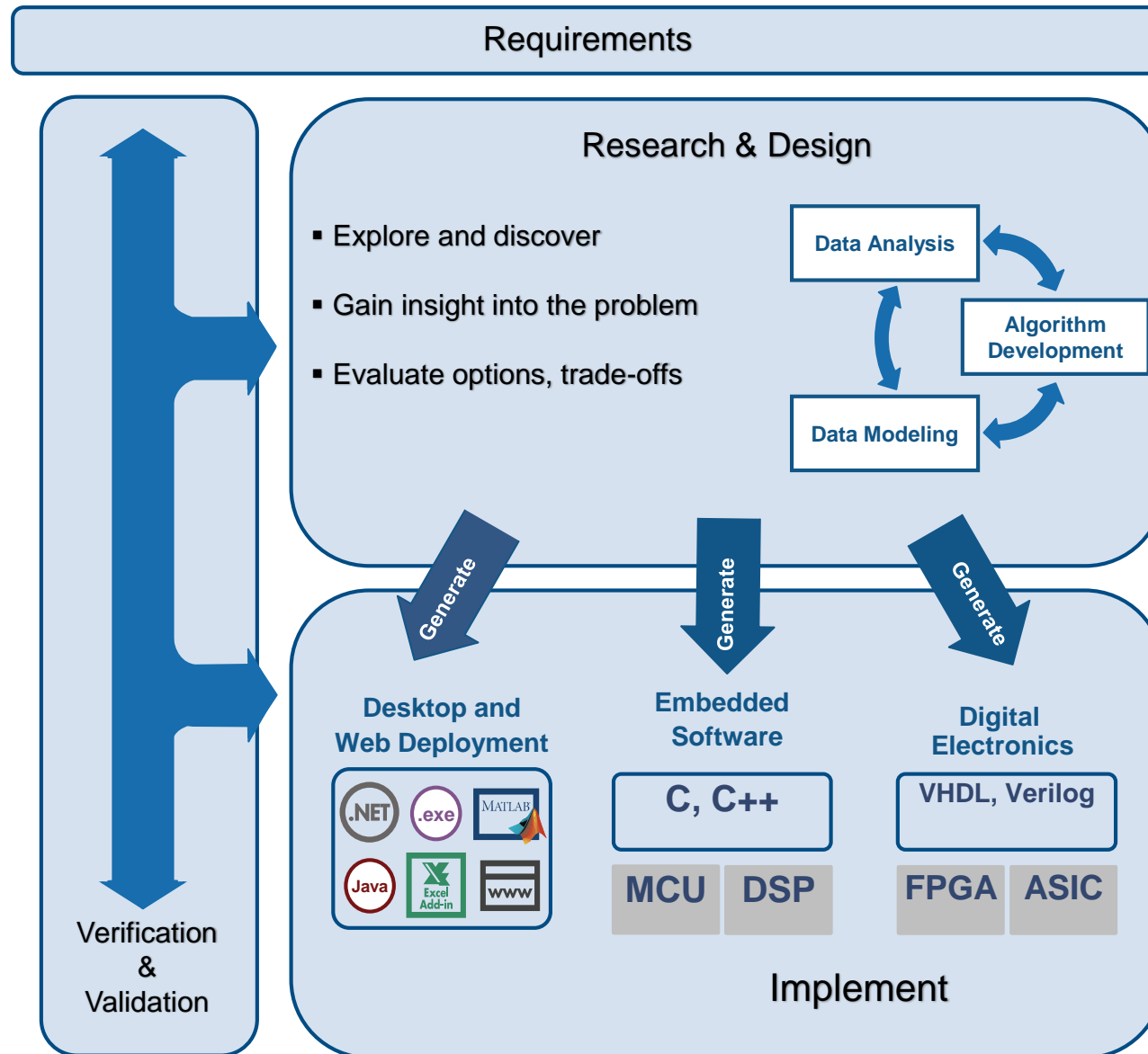
Rapid innovation

*"I need to **find innovative algorithms** and create and model a working system very quickly."*

Simulation acceleration & implementation

*"I need to **optimize my high-level MATLAB algorithm** for **speed**. I then need to verify that the optimized code works the same way as the original MATLAB code."*

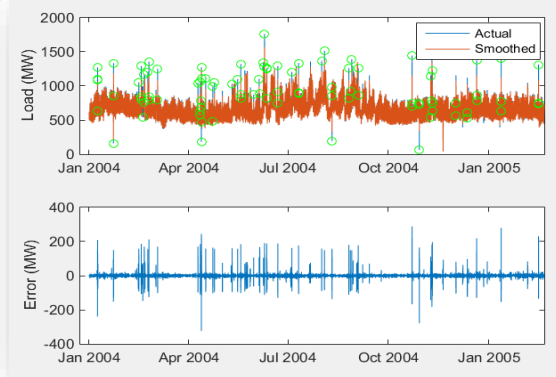
Model Based Design Workflow



Agenda



Connect and Acquire



Signal
Processing Algorithm



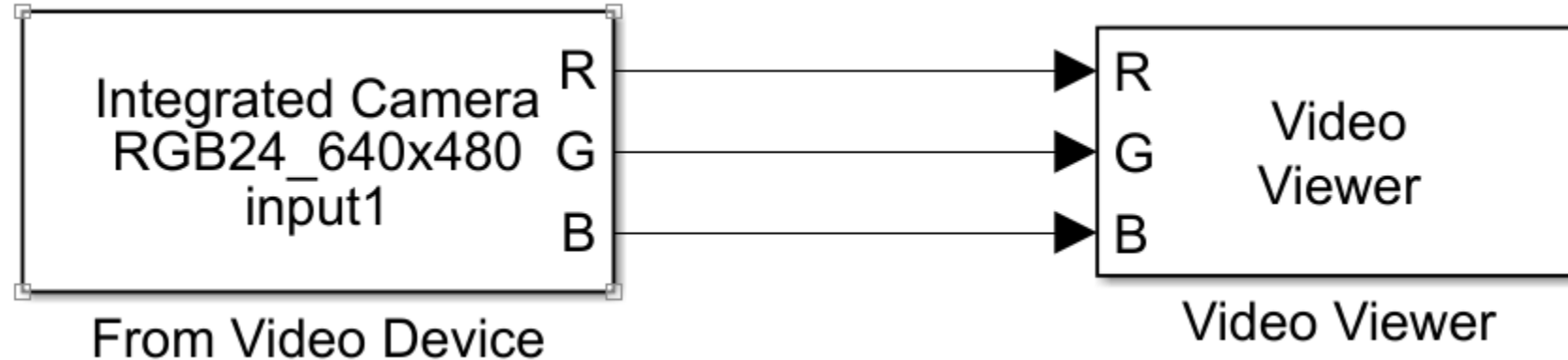
Embedded Implementation

Automate

Data Acquisition

- Image Acquisition Toolbox enables you to acquire live video streams directly into MATLAB and Simulink

[Image Acquisition Toolbox Support Package for OS Generic Video Interface](#)



Data Acquisition

- Signal of interest : Average intensity of each frame
- How do you choose the frame rate?
 - Heart Rate Range : 40 to 230 beats per minute
i.e 0.667 to 3.8333 beats per second (Hz)
 - Minimum frame rate required: 8fps
- How do you choose the resolution?
 - High resolution images are not required in this case
 - Resolution chosen : 640 x 480

MATLAB Connects to Your Hardware Devices

Instrument Control Toolbox

Instruments and RS-232 serial devices



Data Acquisition Toolbox

Plug-in data acquisition devices and sound cards



Image Acquisition Toolbox

Image capture devices



Communications System Toolbox

USRP, RTL-SDR, FPGA-SDR

MATLAB

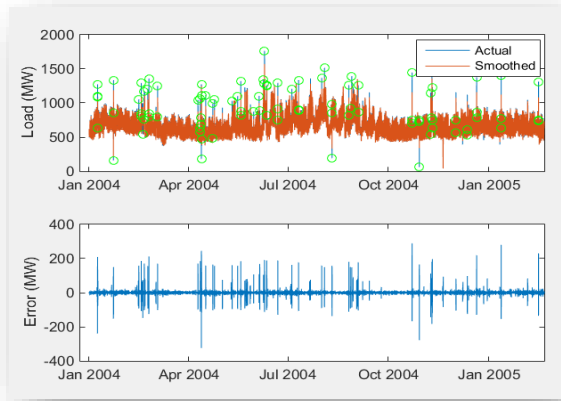
Interfaces for communicating with everything



Agenda



Connect and Acquire



Signal
Processing Algorithm

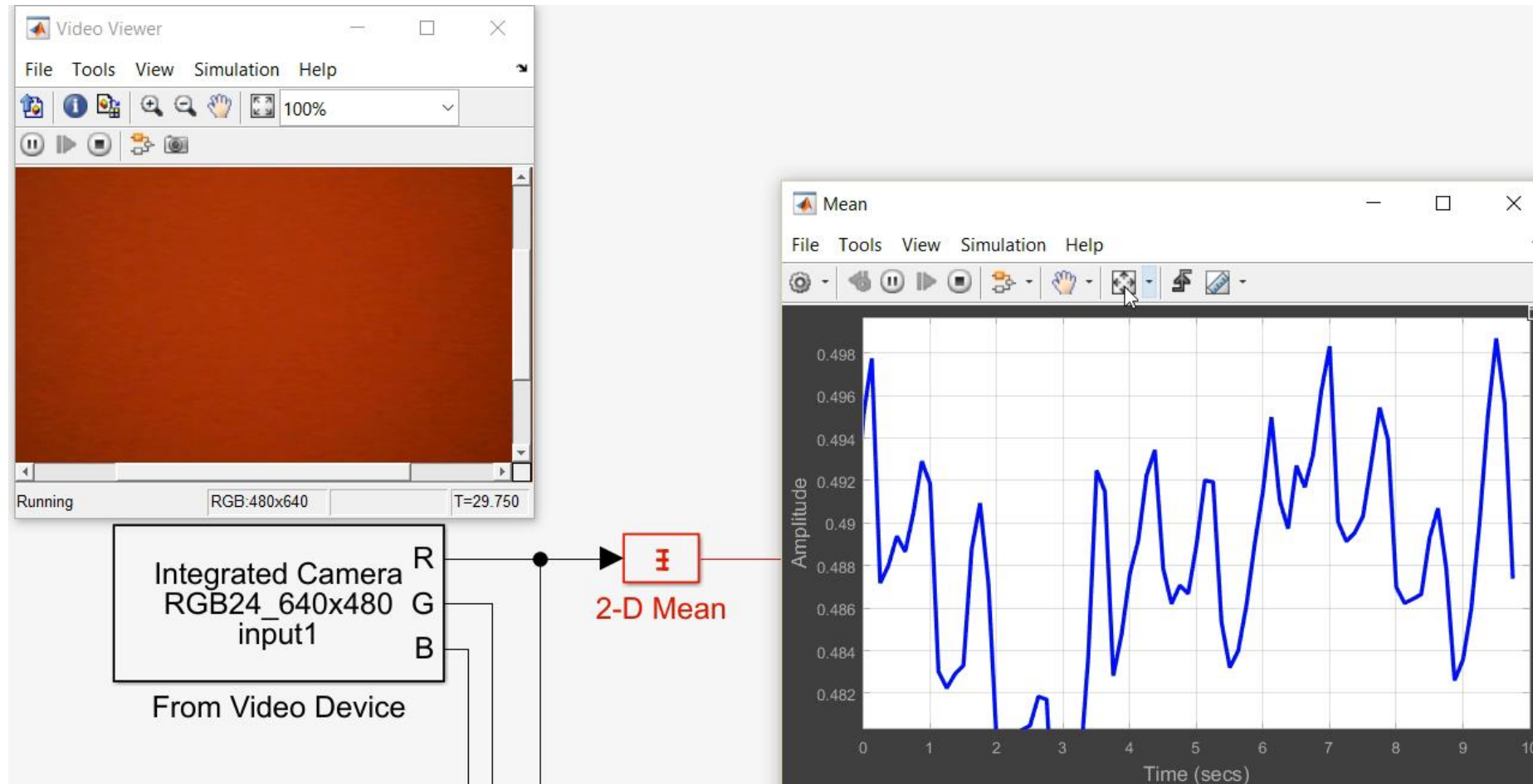


Embedded Implementation

Algorithm Development

Step 1: Computing the average intensity of each frame

- 2D Mean Estimation

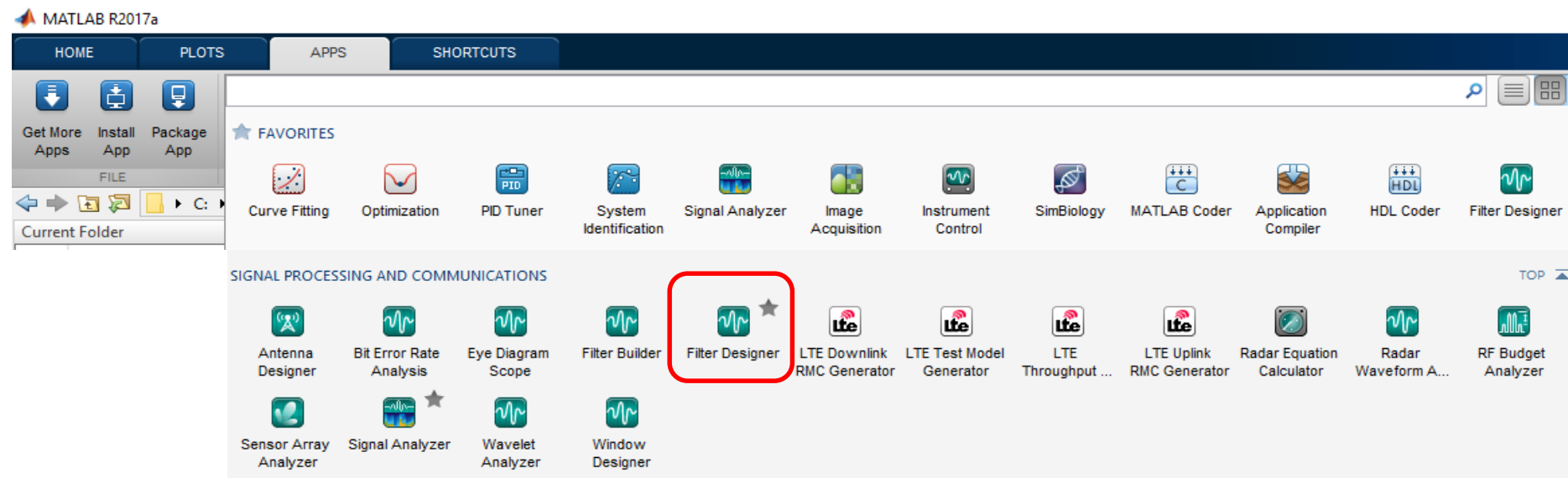


Algorithm Development

Step 2: Extracting the band of interest (Filter specification)

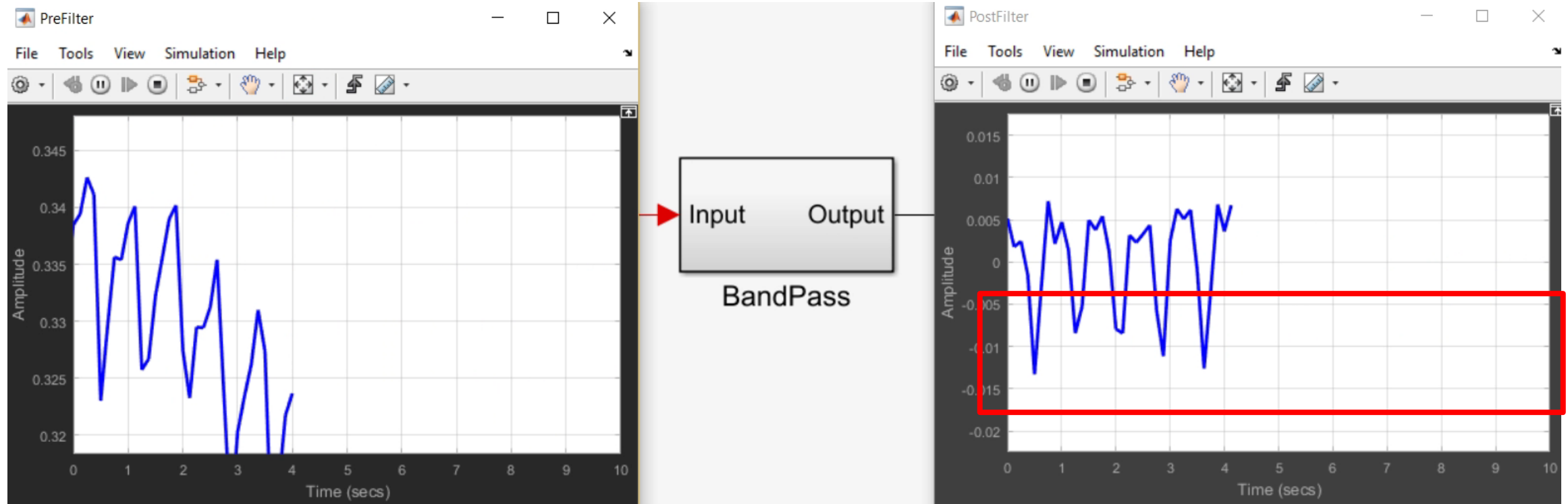
- **Heart Rate Range :**
40 to 230 beats per
minute (0.667 to 3.8333 Hz)

- Choice of Filter?
 - Bandpass filter
- FIR or IIR?
- Order of Filter?



Algorithm Development

Step 2: Extracting the band of interest (Filtered output)



Algorithm Development

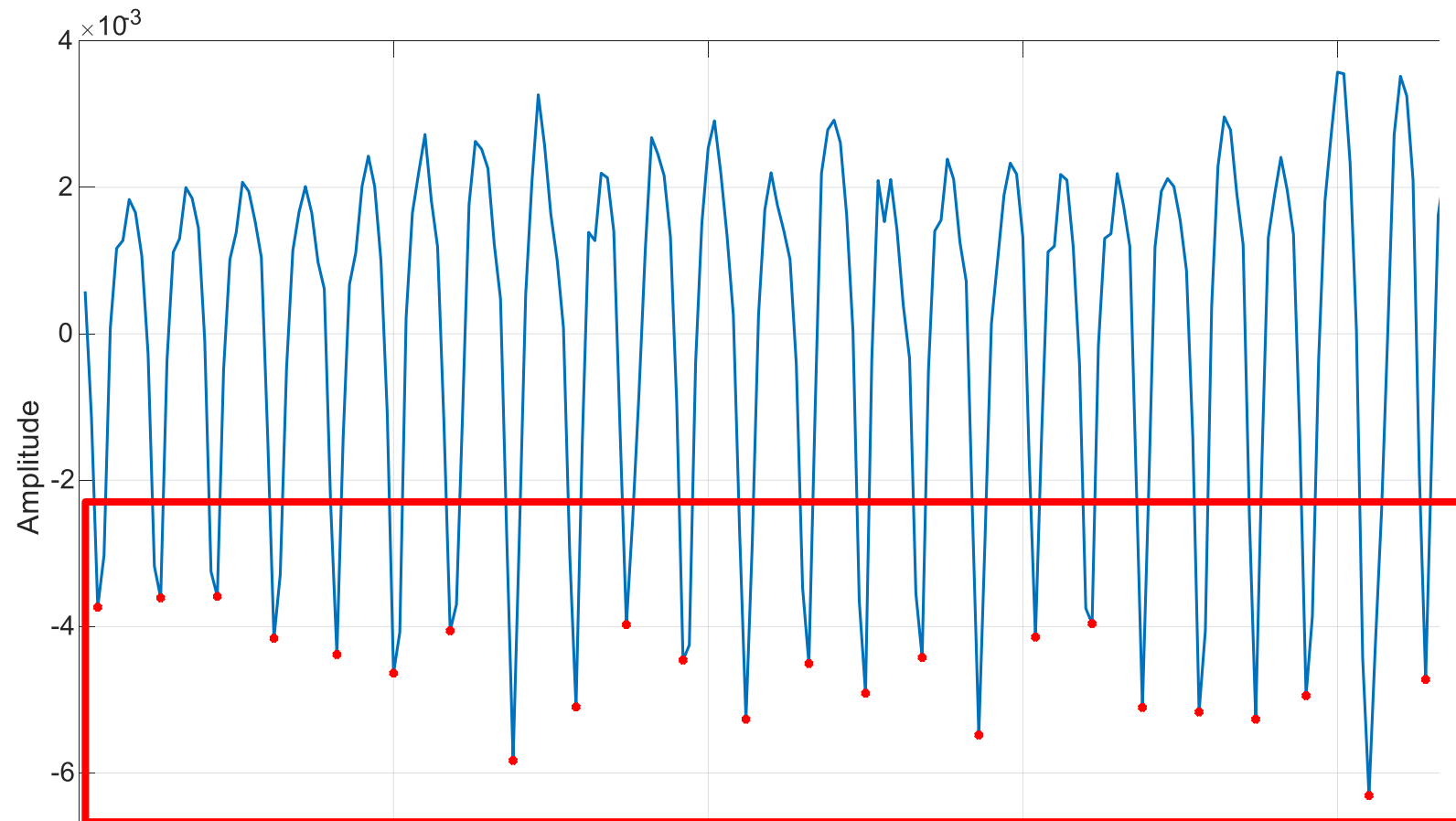
Step 3: Heart Rate Estimation

- Techniques to estimate number of troughs (peaks) in one second
 - Count of number of troughs in a span of “ N ” seconds
 - Estimate the average time between adjacent peaks over “ N ” seconds
- Spectral Analysis
 - Non-parametric : Periodogram, Welch
 - Parametric : Yule-Walker, Burg
 - Subspace methods : Eigen Vector, MUSIC
- Wavelet Analysis

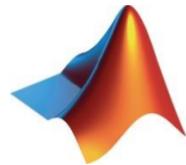
“findpeaks”

Algorithm Development

Step 3: Heart Rate Estimation



```
[pks_val,locs] = findpeaks(-u, 'MinPeakDistance',5)
```

 **MathWorks®** | *Training Services*

Signal Processing with MATLAB

This two-day course shows how to analyze signals and design signal processing systems using MATLAB®, Signal Processing Toolbox™, and DSP System Toolbox™.

Topics include:

- Creating and analyzing signals
- Performing spectral analysis
- Designing and analyzing filters
- Designing multirate filters
- Designing adaptive filters

Signal Processing with Simulink

This three-day course, targeted toward new users of Simulink®, uses basic modeling techniques and tools to demonstrate how to develop Simulink block diagrams for signal processing applications.

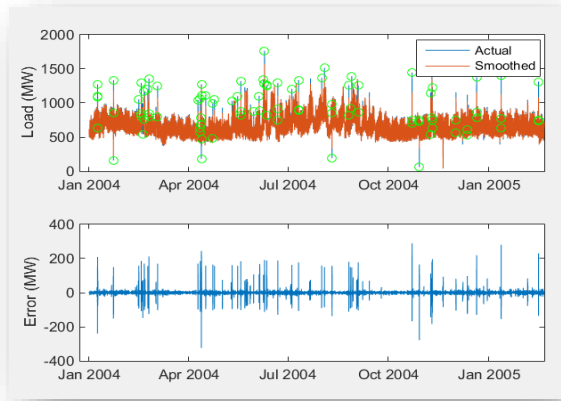
Topics include:

- Using the Simulink interface
- Modeling single-channel and multi-channel discrete dynamic systems
- Implementing sample-based and frame-based processing
- Modeling mixed-signal (hybrid) systems
- Developing custom blocks and libraries
- Modeling condition-based systems
- Performing spectral analysis with Simulink
- Integrating filter designs into Simulink
- Modeling multirate systems

Agenda



Connect and Acquire

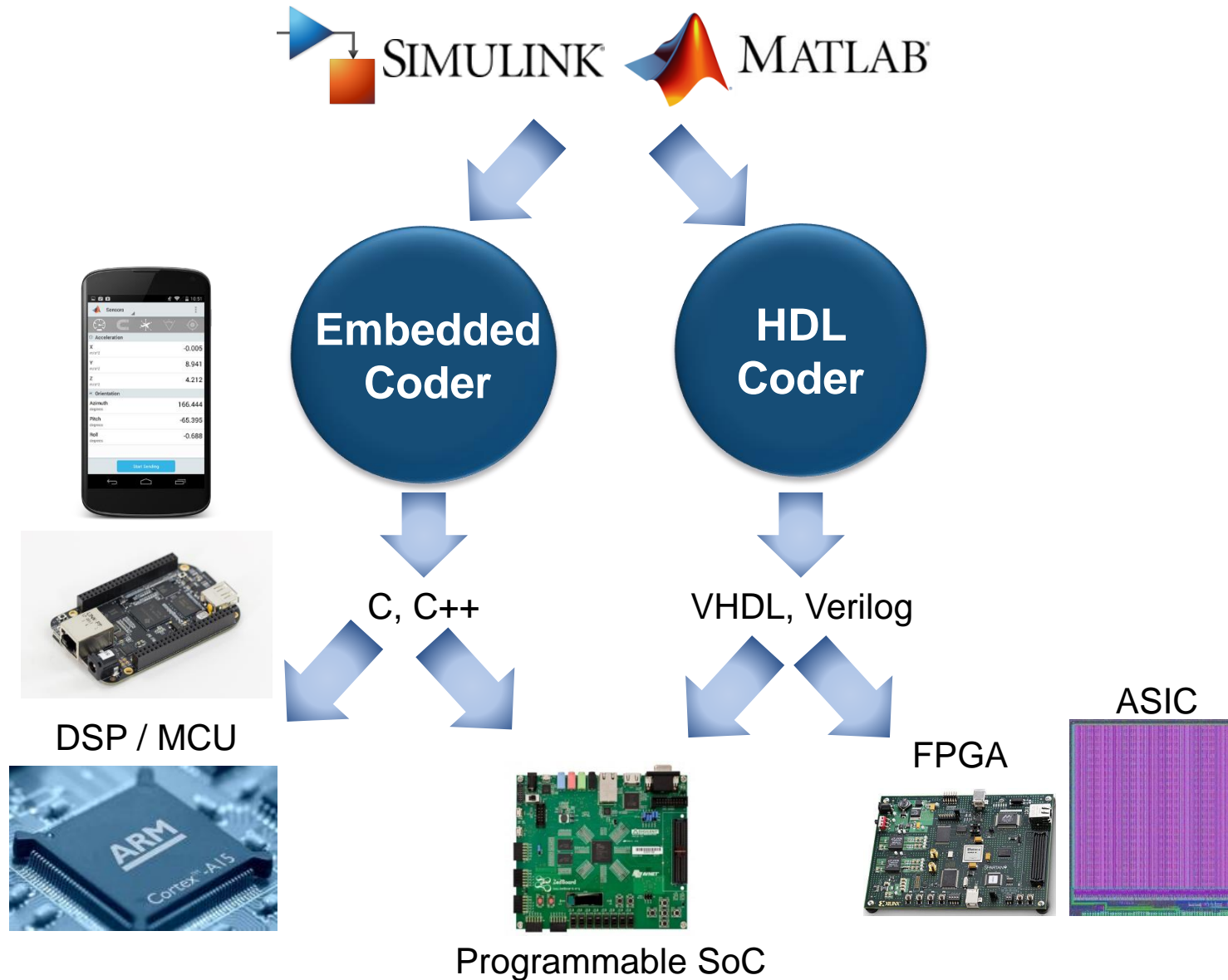


Signal
Processing Algorithm

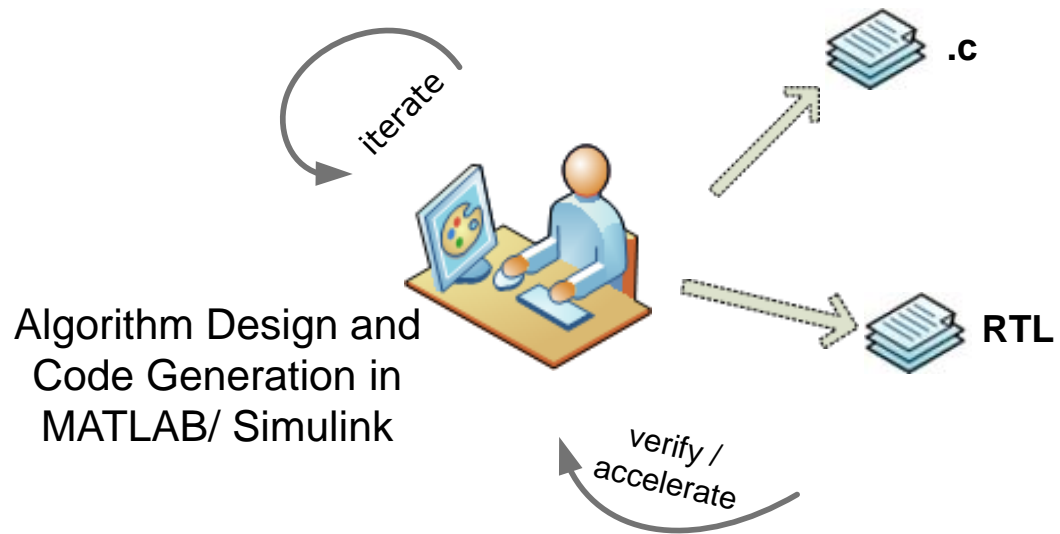


Embedded Implementation

Implement and Prototype Algorithms in Hardware



Why Automatic Code Generation?



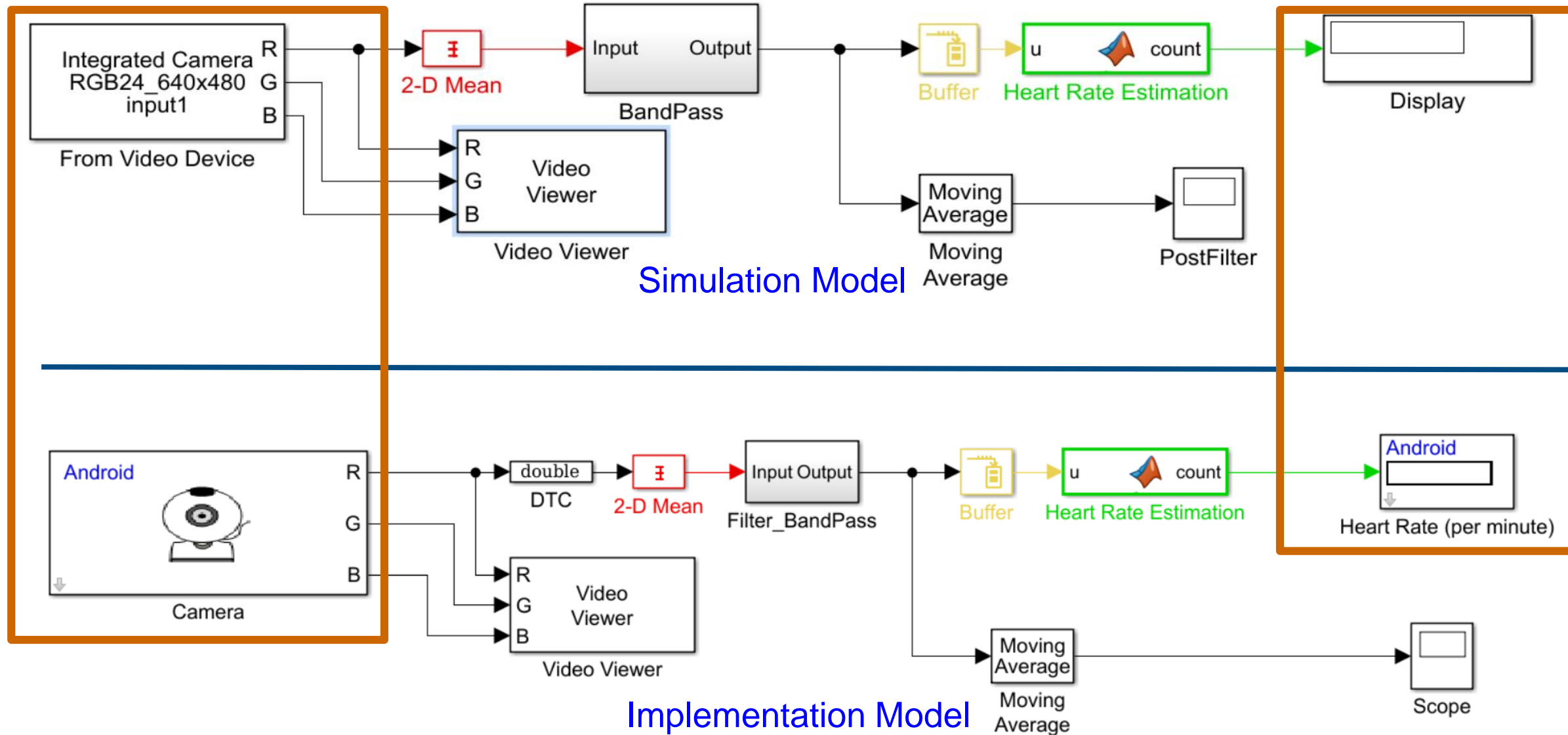
With automatic code generation, design engineers can:

- Maintain one design across simulation and implementation
- Design faster and get to C or RTL quickly
- Test more systematically and frequently
- Spend more time improving algorithms

Challenges with manual coding:

- Separate functional and implementation specification
 - Leads to multiple implementations that are inconsistent
 - Hard to modify requirements during development
- Manual coding errors
- Time-consuming and expensive process

From Simulation model to Implementation model (Android device)



One-Click Implementation and Execution

Configuration Parameters: HeartRateEstimation_External/Run on Hardware Configuration (Active)

★ Commonly Used Parameters ≡ All Parameters

Select:

- Solver
- Data Import/Export
- > Optimization
- > Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
- ▼ Code Generation
 - Report
 - Comments
 - Symbols
 - Custom Code
 - Interface
 - Code Style
 - Verification
 - Templates
 - Code Placement
 - Data Type Replacement
 - Memory Sections
 - > Coverage
 - > HDL Code Generation

Hardware board: Android Device

Code Generation system target file: [ert.tlc](#)

Device vendor: ARM Compatible Device type: ARM Cortex

► Device details

Hardware board settings

Operating system options

Base rate task priority: 40

Target Hardware Resources

Groups
Build options
Device options
App options
ThingSpeak properties
External mode

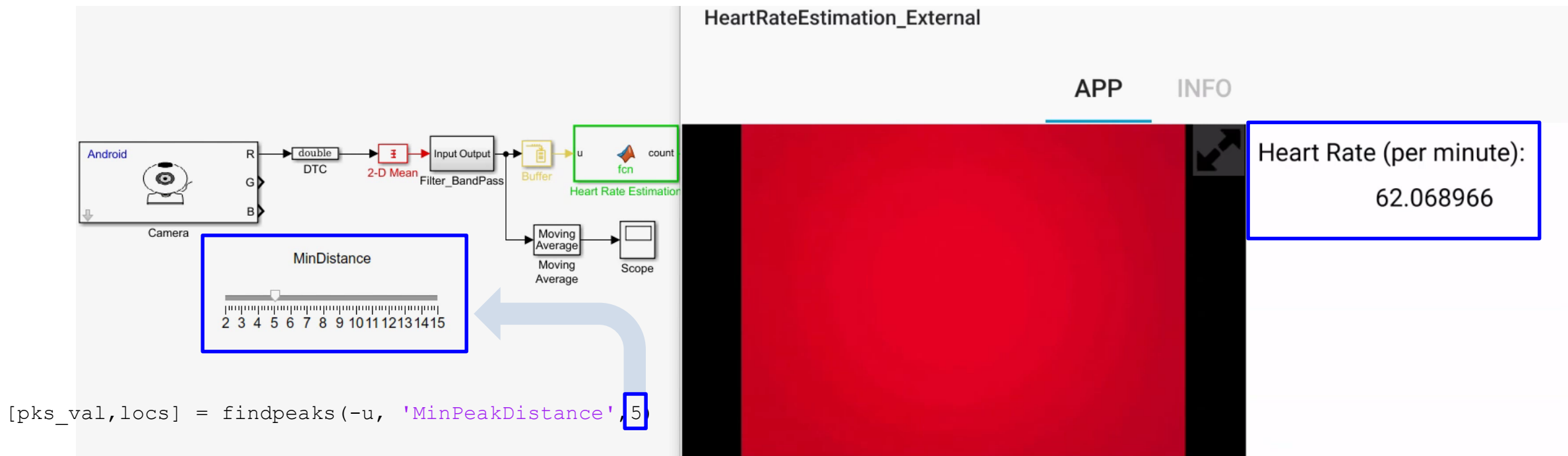
Build action: Build, load and run

Build

Build, load and run

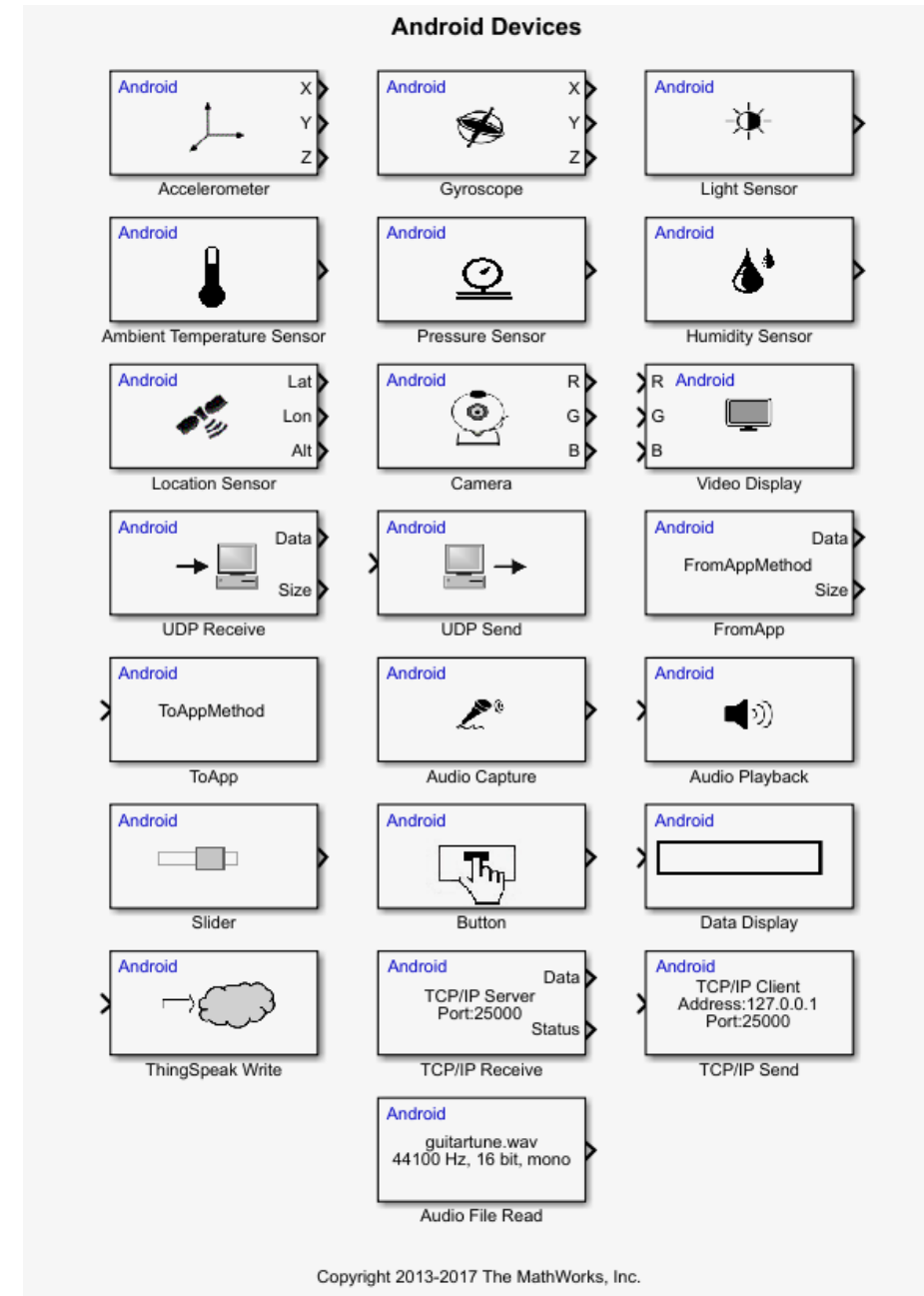
Parameter Tuning using External Mode

- Find the optimal values for performance
- Accelerate parameter tuning
- Develop and validate your model using the actual data and hardware for which it is designed

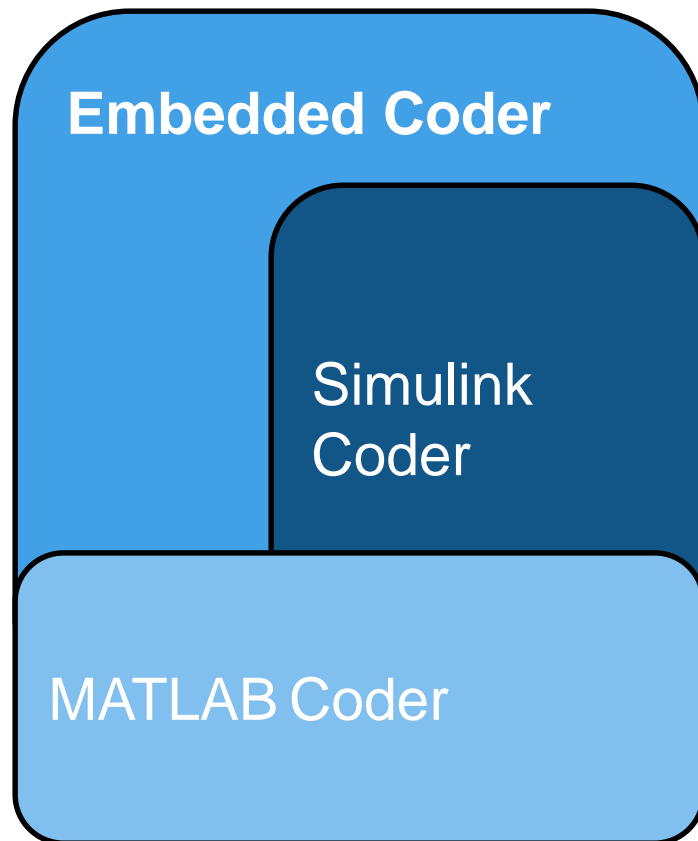


Simulink Support Package for Samsung GALAXY Android Devices

- Interactive parameter tuning and signal monitoring
- Model deployment for standalone operation
- Simple UI using sliders and buttons
- Generation of Android Studio compatible projects



Code Generation Products for C/C++



Embedded Coder™
Automatically generate C and C++
optimized for embedded systems

Simulink® Coder™
Automatically generate C and C++ from
Simulink models and **Stateflow** charts

MATLAB® Coder™
Automatically generate C and C++ from
MATLAB code

Embedded Coder Support for ARM Processors

ARM Cortex A Support from Embedded Coder

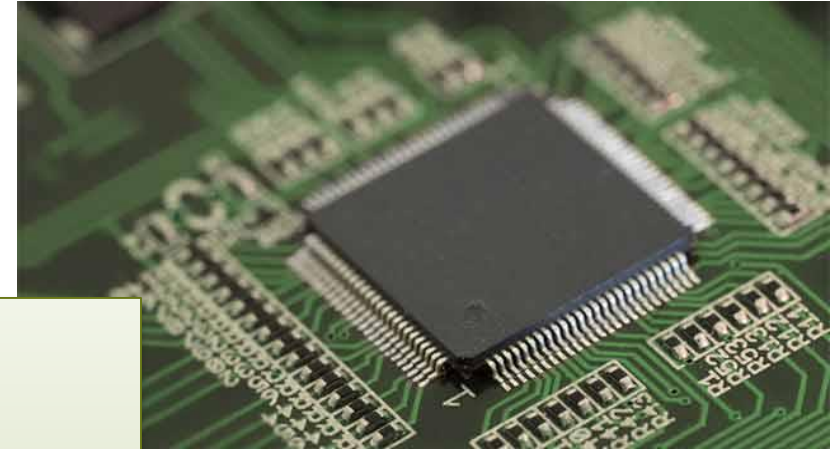
ARM Cortex-M Support from Embedded Coder

ARM Cortex-M CMSIS Library Support from DSP System Toolbox

ARM Cortex A Ne10 Library Support from DSP System Toolbox

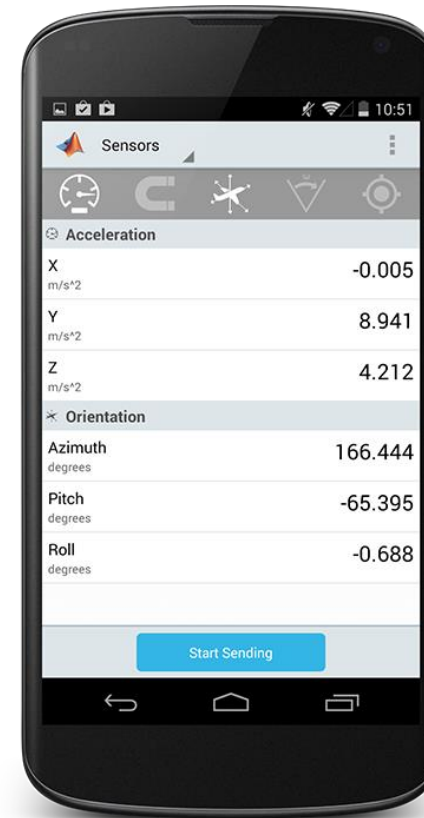
Optimized functions for signal processing algorithms

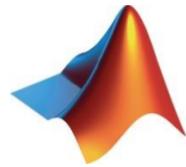
- FFT/IFFT
- Multirate FIR decimation/interpolation
- FIR filter, FIR lattice filter
- Digital Down Converter and Up Converter
- CIC Compensator Decimator and Interpolator



Connecting MATLAB and Simulink to Hardware

- Android and iOS
- Arduino® Uno, Mega 2560
- Raspberry Pi Model B
- BeagleBone Black
- Texas Instruments processors
- Analog Devices processors
- ...



 **MathWorks®** | *Training Services*

MATLAB to C with MATLAB Coder

This two-day course focuses on generating C code from MATLAB® code using MATLAB Coder™. The focus is on developing MATLAB code that is ready for code generation, generating C code that meets optimization requirements, and integrating generated code into parent projects and external modules. This course is intended for intermediate to advanced MATLAB users.

Embedded Coder for Production Code Generation

This hands-on, three-day course focuses on developing models in the Simulink® environment to deploy on embedded systems. The course is designed for Simulink users who intend to generate, validate, and deploy embedded code using Embedded Coder®.

Topics include:

- Generated code structure and execution
- Code generation options and optimizations
- Integrating generated code with external code
- Generating code for multirate systems
- Customizing generated code
- Customizing data
- Deploying code

VivaQuant Accelerates Development and Validation of Embedded Device for Ambulatory ECG Sensing

Challenge

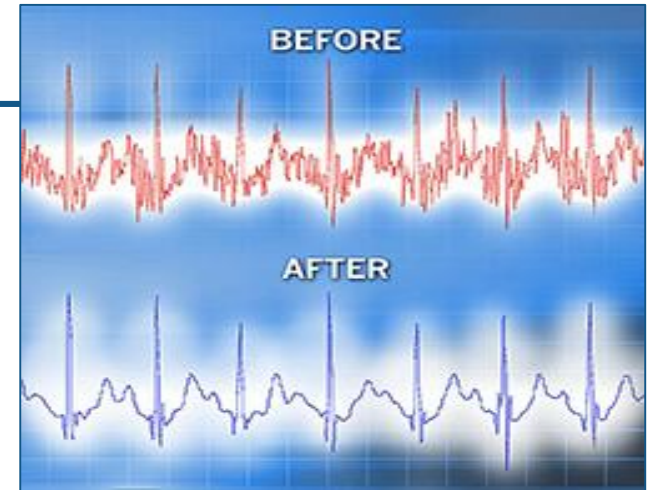
Design and implement an embedded system for extracting accurate information from noisy electrocardiogram signals

Solution

Use MATLAB to develop an algorithm for **removing in-band noise**, and use **Fixed-Point Designer** and MATLAB Coder to implement it on an ARM Cortex-M series processor

Results

- Development accelerated by 300%
- Power and memory consumption minimized
- Rigorous testing enabled



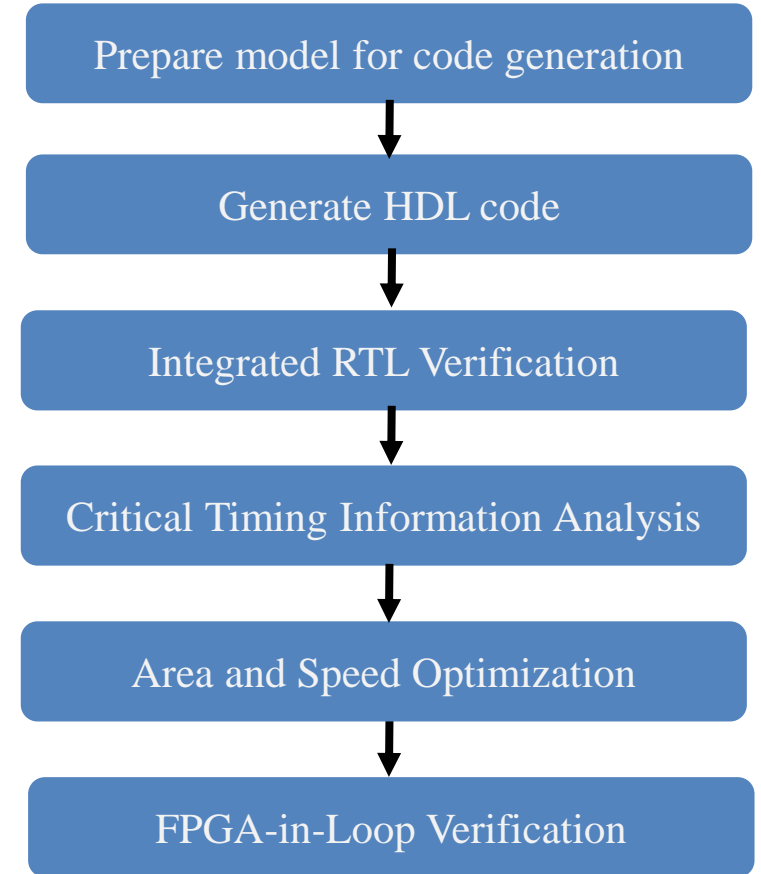
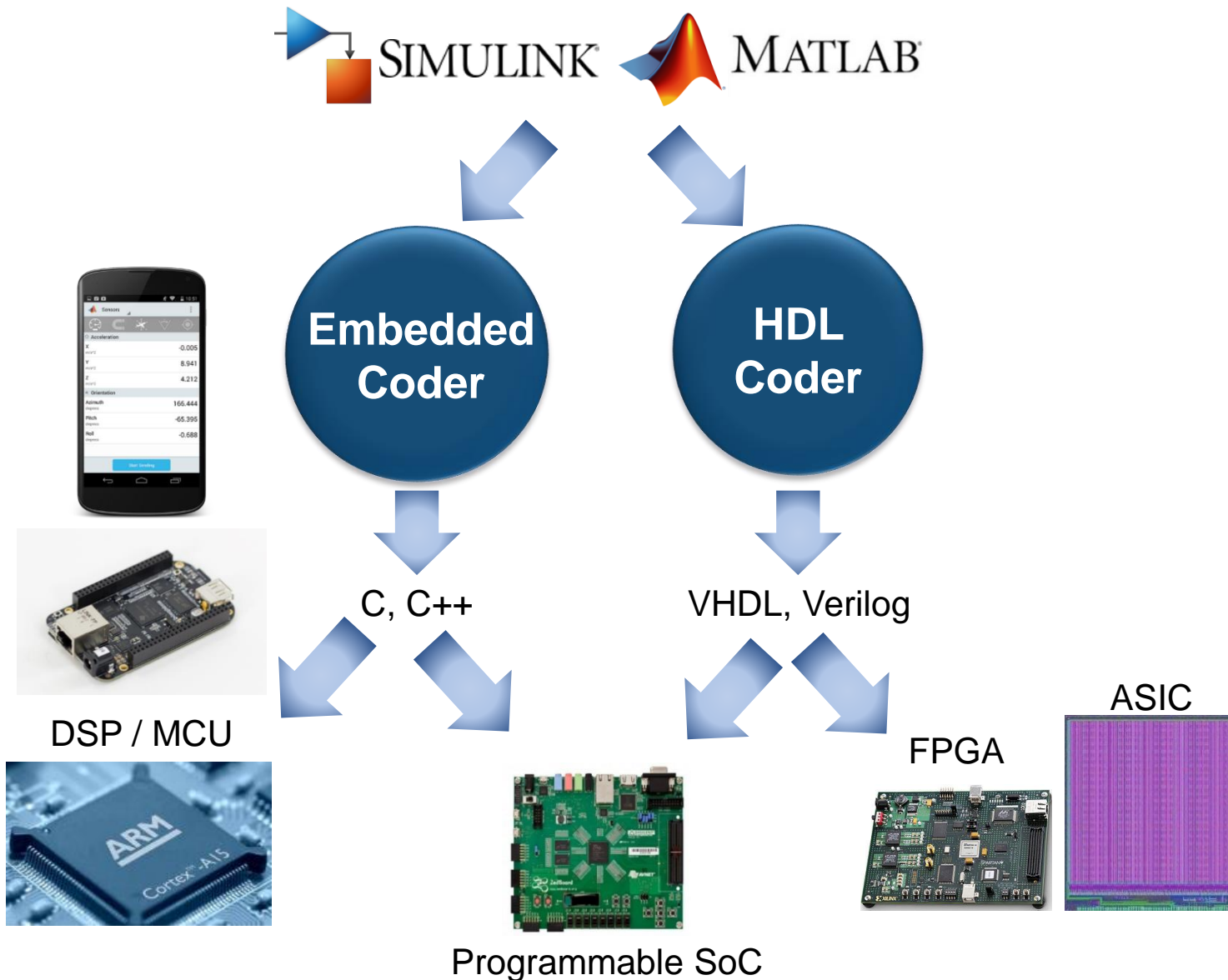
ECG snippet before and after processing with VivaQuant's embedded in-band noise removal algorithm.

“MATLAB, MATLAB Coder, and Fixed-Point Designer enabled our small team to develop a complex real-time signal processing algorithm, optimize it to reduce power and memory requirements, accelerate embedded code implementation, and perform the rigorous testing required for medical device validation.”

Marina Brockway
VivaQuant

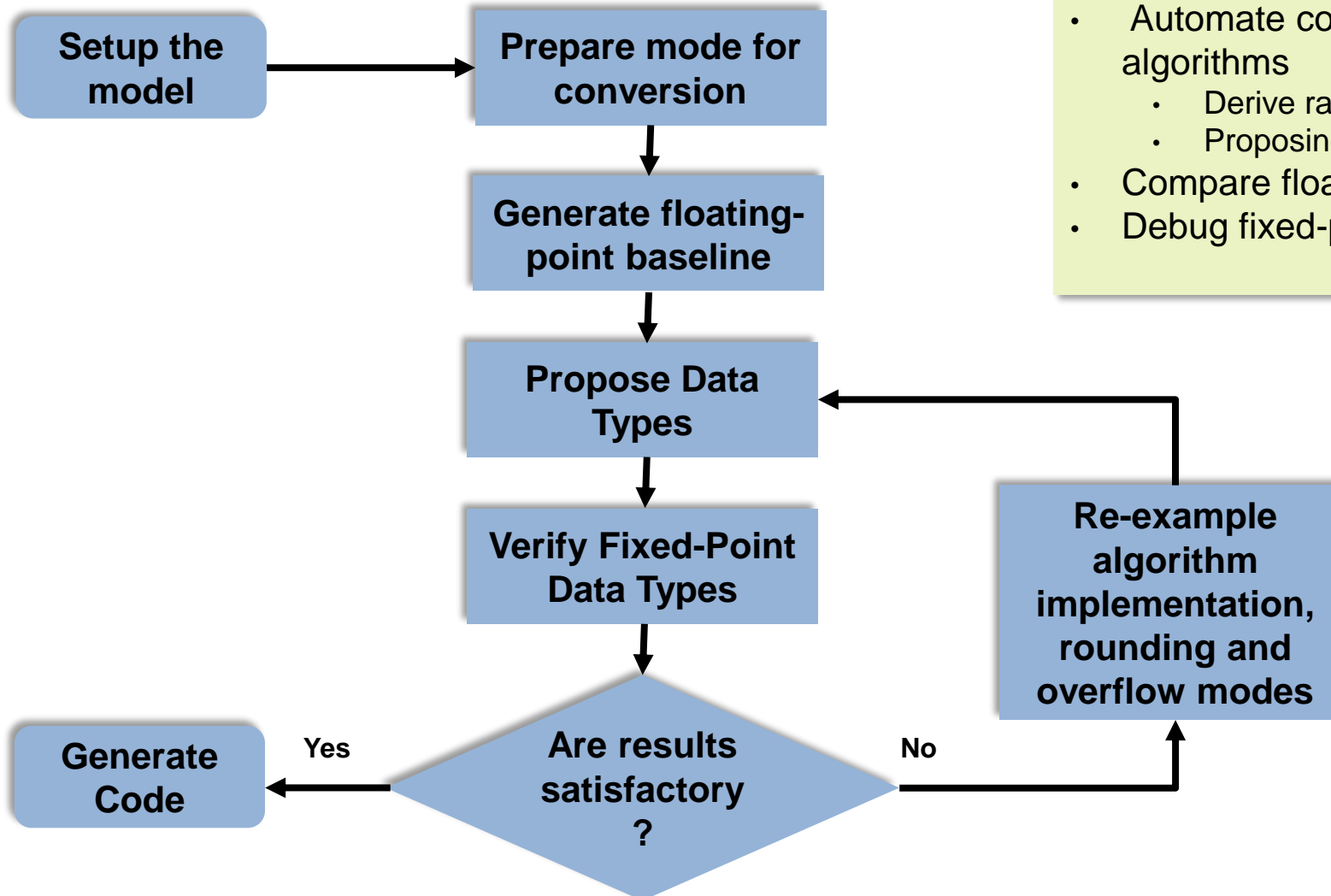
[Link to user story](#)

Implement and Prototype Algorithms in Hardware



Preparation of an Implementation Model

Fixed-Point Designer

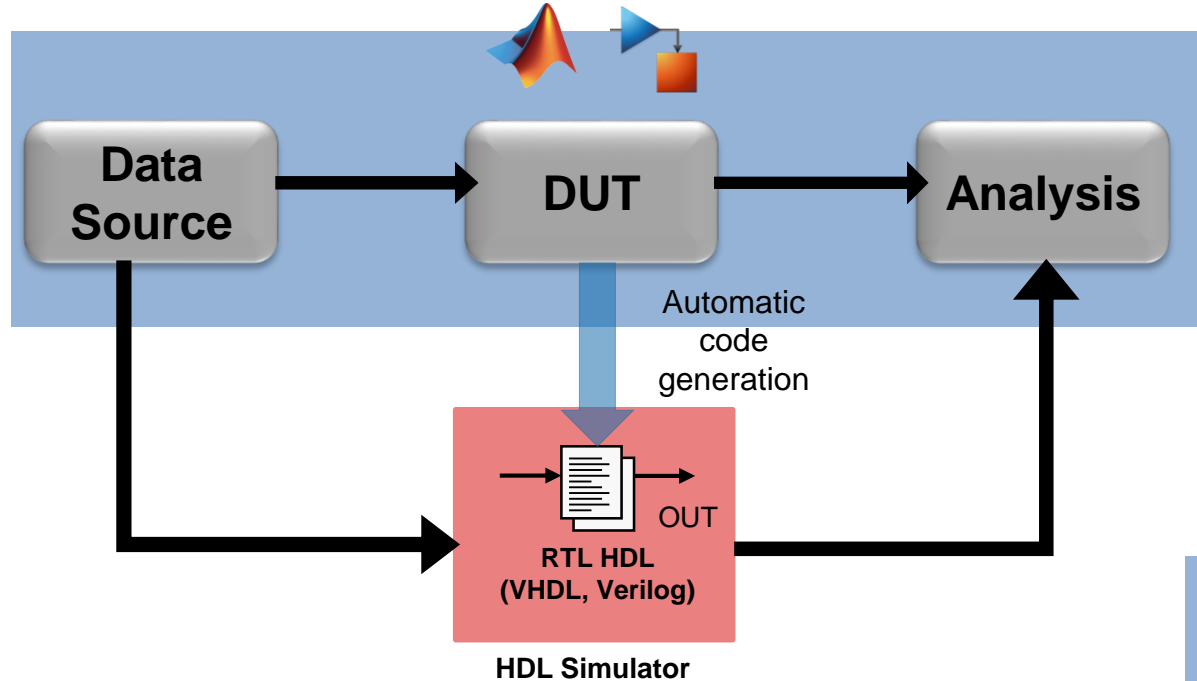


Fixed-Point Designer enables you to

- Automate conversion from floating-point to fixed-point algorithms
 - Derive range information
 - Proposing data types based on the ranges
- Compare floating-point and fixed-point results
- Debug fixed-point models

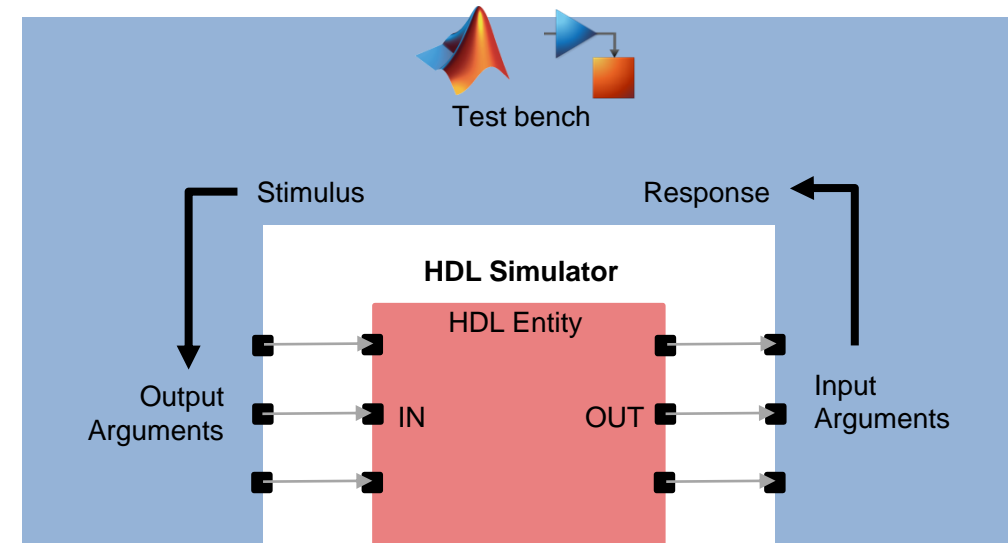
Verification of Implemented Design

HDL Verifier



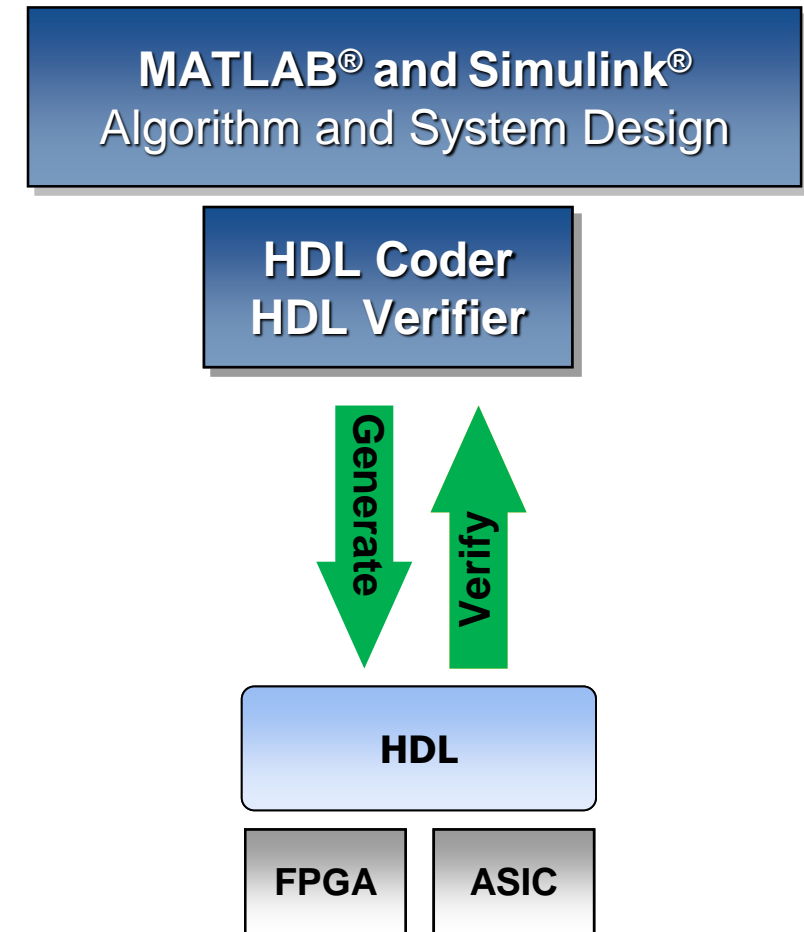
Verify your generated RTL against MATLAB/ Simulink results

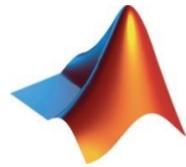
MATLAB/ Simulink as a test bench



What HDL Coder and HDL Verifier offers?

- **Code Generation**
 - Target-independent HDL Code
 - VHDL-1993 (IEEE® 1076-1993) or later
 - Verilog-2001 (IEEE 1364-2001) or later
- **Verification**
 - Generate HDL test-bench
 - Co-simulate with ModelSim and Incisive
 - FPGA-in-the-loop verification
- **Design automation**
 - Synthesize using integrated Xilinx and Altera synthesis tool interface
 - Optimize for area-speed
 - Program Xilinx and Altera boards



 **MathWorks®** | *Training Services*

Generating HDL Code from Simulink

Two-day course shows how to generate and verify HDL code from a Simulink® model using HDL Coder™ and HDL Verifier™

Topics include:

- Preparing Simulink models for HDL code generation
- Generating HDL code and testbench for a compatible Simulink model
- Performing speed and area optimizations
- Integrating handwritten code and existing IP
- Verifying generated HDL code using testbench and cosimulation

DSP for FPGAs

This three-day course will review DSP fundamentals from the perspective of implementation within the FPGA fabric. Particular emphasis will be given to highlighting the cost, with respect to both resources and performance, associated with the implementation of various DSP techniques and algorithms.

Topics include:

- Introduction to FPGA hardware and technology for DSP applications
- DSP fixed-point arithmetic
- Signal flow graph techniques
- HDL code generation for FPGAs
- Fast Fourier Transform (FFT) Implementation
- Design and implementation of FIR, IIR and CIC filters
- CORDIC algorithm
- Design and implementation of adaptive algorithms such as LMS and QR algorithm
- Techniques for synchronisation and digital communications timing recovery

FLIR Accelerates Development of Thermal Imaging FPGA

Challenge

Accelerate the implementation of advanced thermal imaging filters and algorithms on FPGA hardware

Solution

Use MATLAB to develop, simulate, and evaluate algorithms, and use HDL Coder to implement the best algorithms on FPGAs

Results

- Time from concept to field-testable prototype reduced by 60%
- Enhancements completed in hours, not weeks
- Code reuse increased from zero to 30%

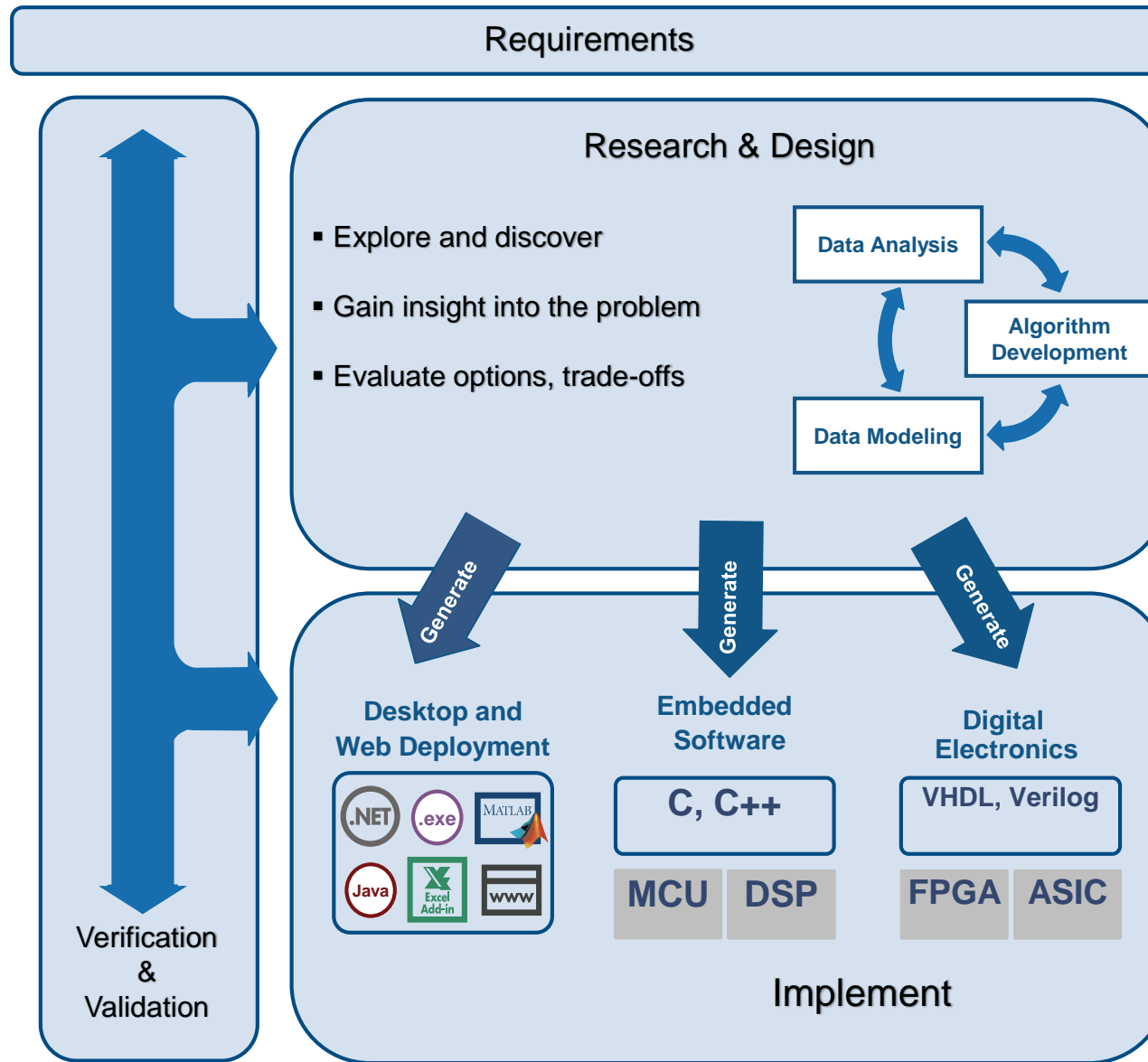


Raw image (left) and image after applying filter developed with HDL Coder (right).

“With MATLAB and HDL Coder we are much more responsive to marketplace needs. We now embrace change, because we can take a new idea to a real-time-capable hardware prototype in just a few weeks. There is more joy in engineering, so we’ve increased job satisfaction as well as customer satisfaction.”

Nicholas Hogasten
FLIR Systems

Model Based Design Workflow



Executable Specification

Design with Simulation

Automatic Code Generation

Continuous Test and Verification

Real-Time Signal Processing System Design

with MATLAB and Simulink

Framework for real-time simulations

Stream processing techniques and hardware peripheral access that speed up simulation and reduce memory footprint

Rapid Innovation

Pre-defined algorithms as functions and blocks for quick prototyping

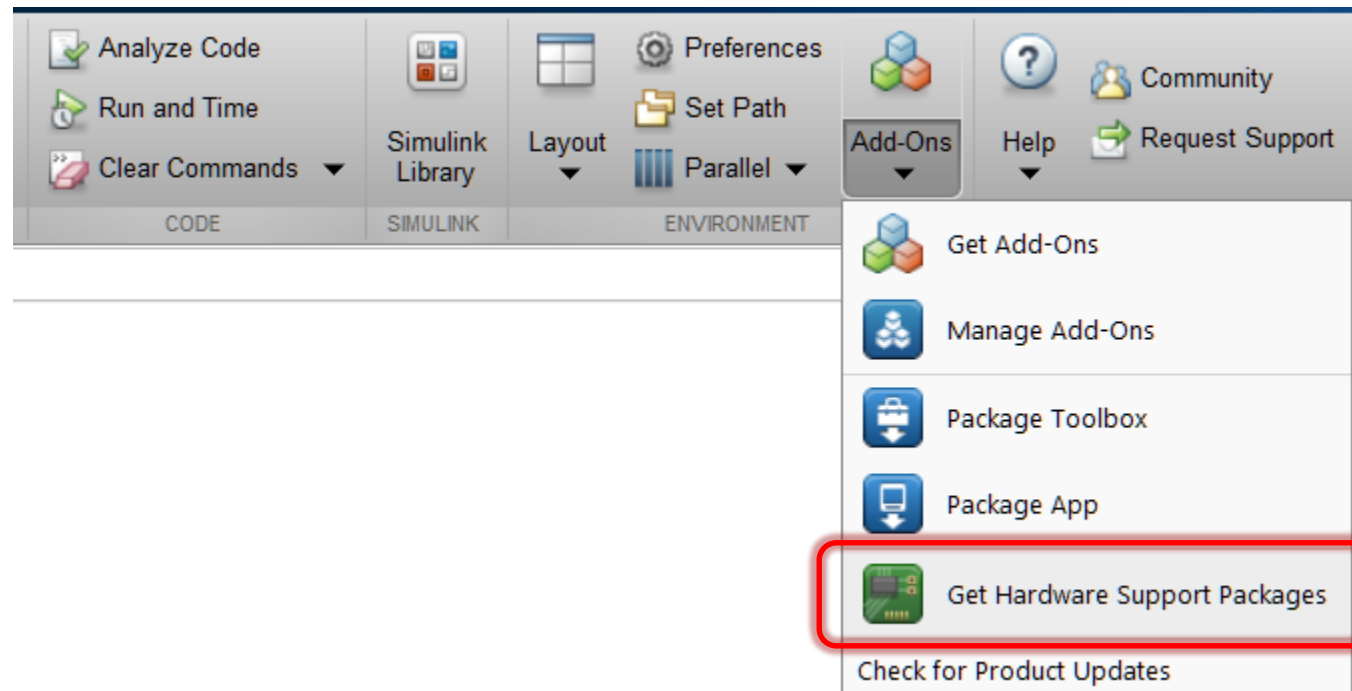
Simulation acceleration & Implementation

Support for C/C++ code and HDL code generation that enables design continuity and faster simulation

Call to Action

Get Hardware Support Packages

From the MATLAB Toolstrip:
Add-Ons → Get Hardware Support Packages



From the MATLAB Command Line:
>> supportPackageInstaller

Call to Action

Videos and Webinars

- [Real-time Audio Processing for Algorithm Prototyping and Custom Measurements](#)
 - Process low-latency streaming audio directly in MATLAB, including via ASIO or CoreAudio
 - Tune parameters in real-time through user interfaces or external MIDI controls

- [Signal Processing and Machine Learning Techniques for Sensor Data Analytics](#)
 - Introduction to common signal processing methods
 - Explore and test different classification algorithms
 - Embedded sensor analytics workflow

- [Understanding Wavelets](#)

Call to Action

Developing and Deploying Analytics for IoT Systems

15:45–16:30

The combination of smart connected devices with data analytics and machine learning is enabling a wide range of applications, from home-grown traffic monitors to sophisticated predictive maintenance systems and futuristic consumer products. While the potential of the Internet of Things (IoT) is virtually limitless, designing IoT systems can seem daunting, requiring a complex web infrastructure and multinomial expertise.

Developing and Prototyping Next-Generation Communications Systems

15:45–16:30

Wireless communication has seen a proliferation of standards addressing many traditional applications, such as mobile telephone and wireless broadband internet access, and emerging areas, such as Internet of Things and vehicle-to-vehicle communication. Developing radios for next-generation communications systems requires expertise in antenna and RF design, DSP and digital logic implementation, embedded software development, and system architecture modeling and simulation. MATLAB® and Simulink® provide a platform that encompasses algorithm design, system simulation, over-the-air testing, prototyping, and implementation.



Amit Doshi, Senior Application Engineer, MathWorks India



Dr. Amod Anandkumar, Team Lead – Signal Processing and Communications, Application Engineering Group, MathWorks India

Call to Action

Parallel Computing with MATLAB and Simulink

16:45–17:30

Large-scale simulations and data processing tasks take an unreasonably long time to complete or require a lot of computer memory. Users can expedite these tasks by taking advantage of high-performance computing resources, such as multicore computers, GPUs, computer clusters, and cloud computing services.

In this session, Alka discusses how to boost the execution speed of computationally and data-intensive problems using MATLAB® and parallel computing products. Alka demonstrates several high-level programming constructs that allow you to easily create parallel MATLAB applications without low-level programming.



Alka Nair, Application Engineer, MathWorks India

Simplifying Image Processing and Computer Vision Application Development

16:45–17:30

Image processing and computer vision is an enabling technology that is driving the development of several of the smart systems today including self-driving cars, augmented reality, hyperspectral imaging, and medical imaging. Developers of modern image processing and computer vision applications face many challenges regarding handling large data sets and working with new computing paradigms, such GPU computing. You can use MATLAB® to simplify your image processing and computer vision application development workflow.



Elza John, Training Engineer, MathWorks India

Call to Action

Demo Stations and Workshops

- Machine Learning Made Easy Using MATLAB
 - Data import and feature selection
 - Choosing algorithms
- Simulink for Maker Hardware
 - Develop standalone applications for Arduino, LEGO MINDSTORMS EV3, and Raspberry Pi
 - Create Android and iOS apps using Simulink®
- Hardware-Software Codesign and Prototyping on SoC FPGAs
 - HDL and C code generation
 - Partitioning an algorithm between ARM® core and programmable logic
- Hands-on workshop on HDL code generation



Speaker Details

Email: Vidya.Viswanathan@mathworks.in

LinkedIn:

<https://www.linkedin.com/in/vidyaviswanathan>

Contact MathWorks India

Products/Training Enquiry Booth

Call: 080-6632-6000

Email: info@mathworks.in

Your feedback is valued.

Please complete the feedback form provided to you.