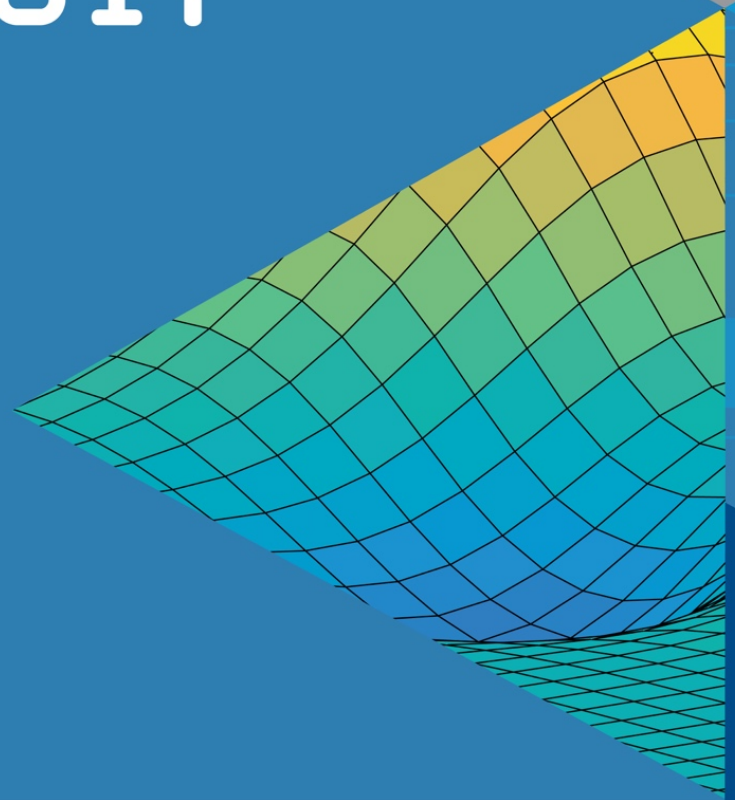
Better than Hand – Generating Highly Optimized Code using Simulink and Embedded Coder

MATLAB EXPO 2017

Lars Krause
Application Engineering



Challenges

Limited time and resources are common constraints for development projects

- Fit advanced algorithms into **low-cost production hardware**
  - Limited ROM, RAM, stack, and speed
- **Embedded device** often not known during design
  - Need optimal implementation
- **Hand coding is process bottleneck**
  - Adds bugs, delays, iterations

*"The **advantages of Model-Based Design** over hand-coding in C can't be overestimated."*
Kazuhiro Ichikawa, Ono Sokki

Ono Sokki Reduces Development Time for **Precision Automotive Speed** Measurement Device

Solutions

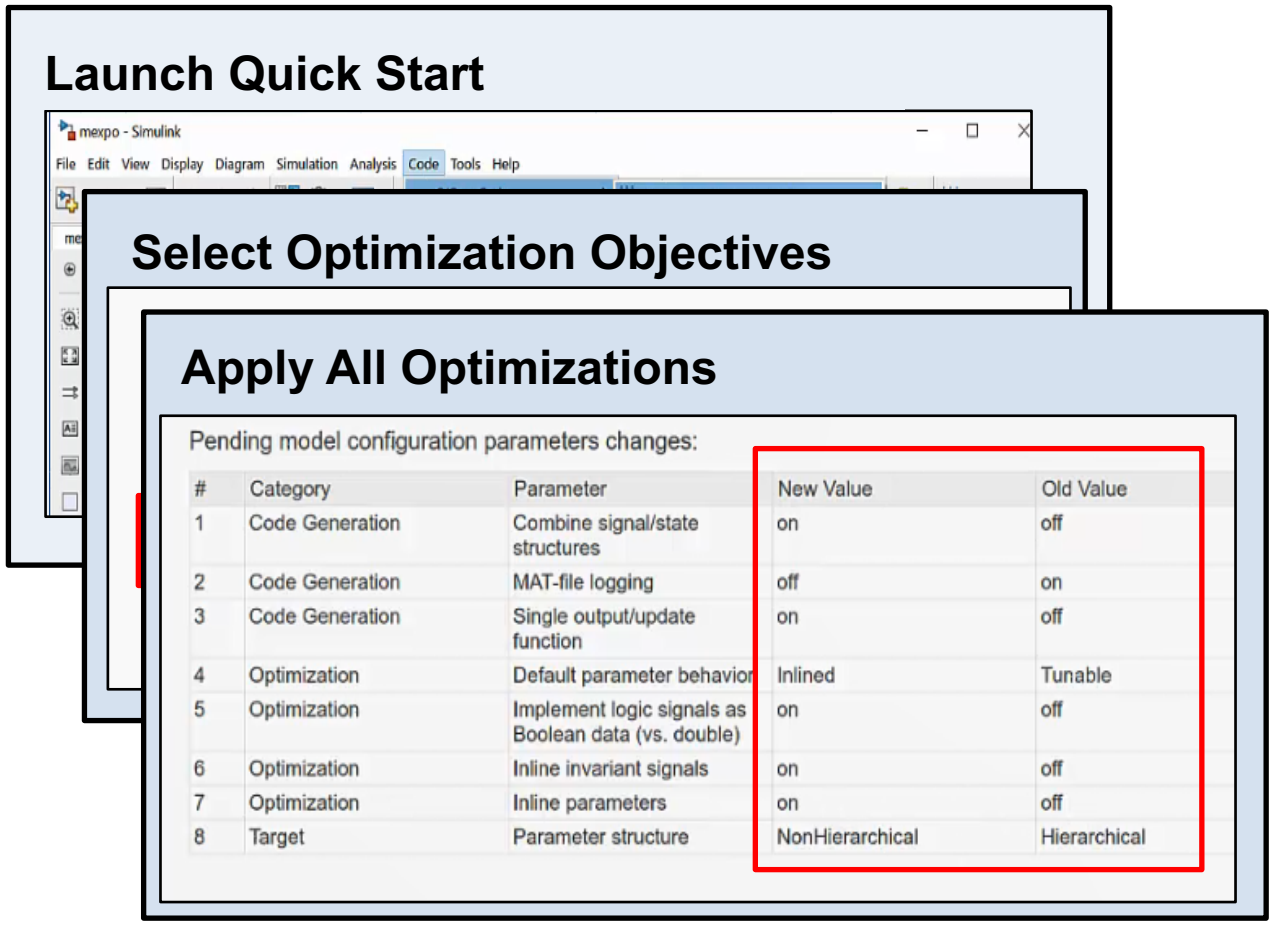
Techniques for accelerating the development process

Optimization Techniques

1. Use optimal settings
2. Optimize data types
3. Target vector engines
4. Use hardware support packages
5. Reuse components
6. Reduce variables
7. Reduce logic

1. Use optimal settings

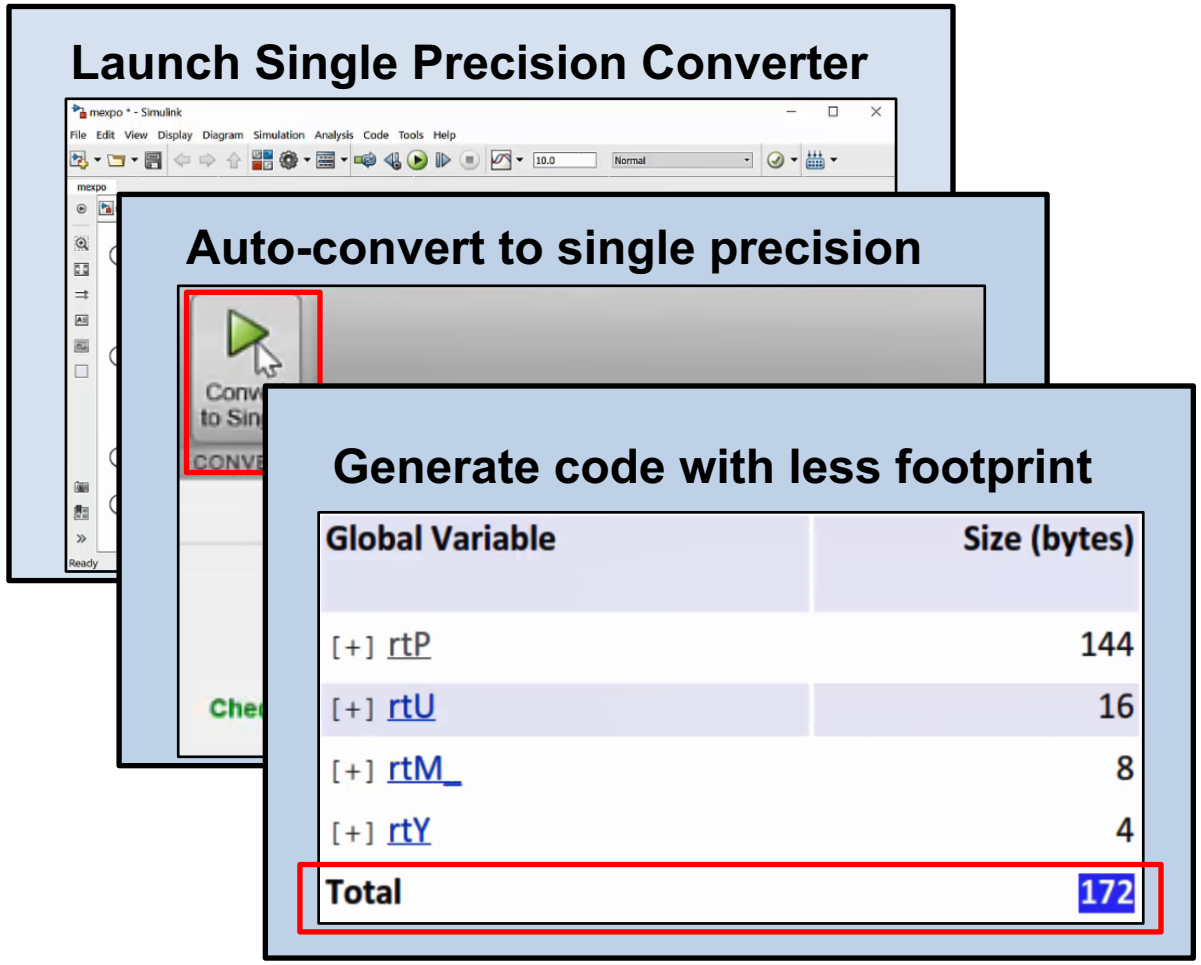
Embedded Coder Quick Start



- Prepare your model for production code generation
- Optimize generated code, independently of target
- Find optimal settings with the Quick Start Tool

2. Optimize data types

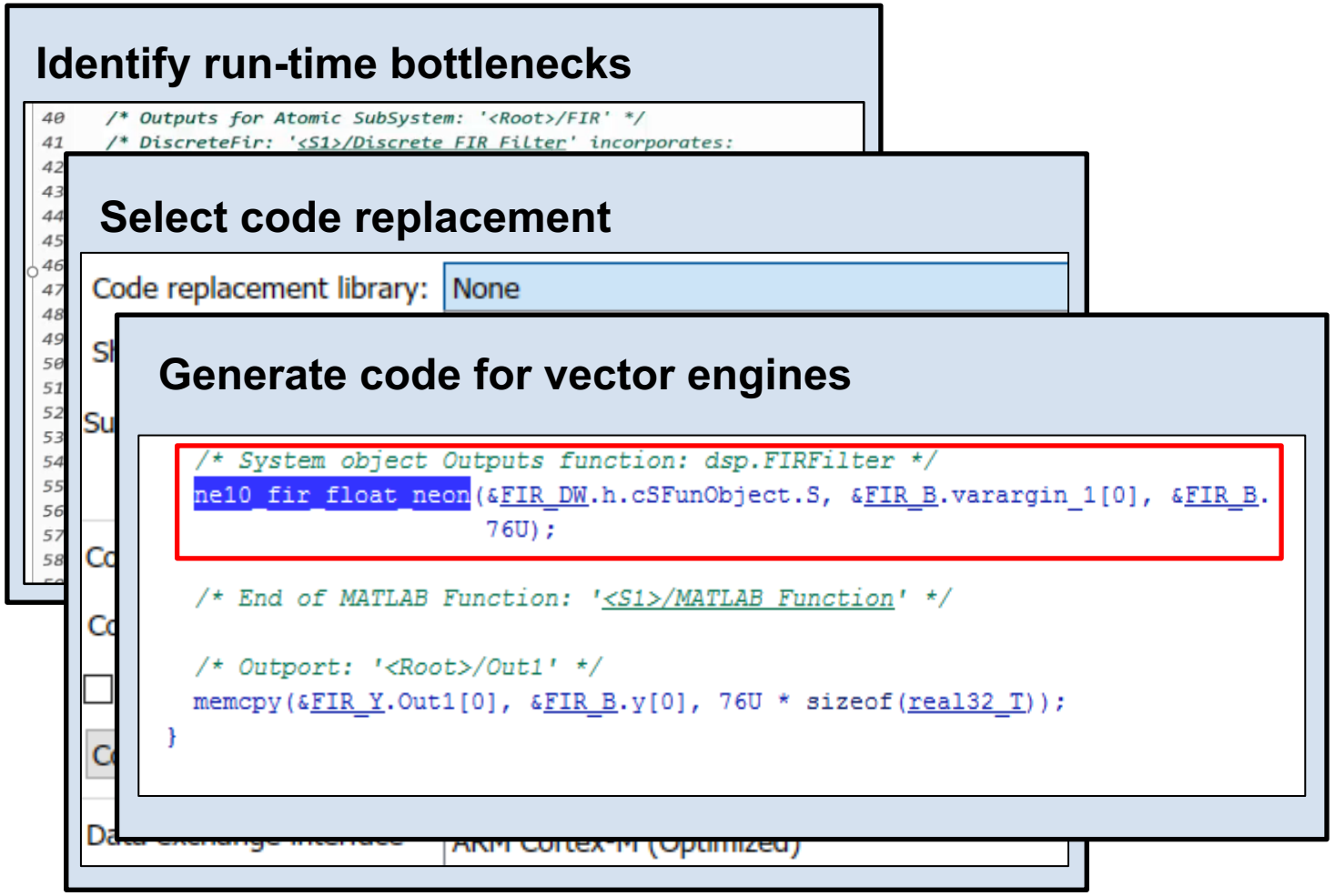
Single Precision Converter



- Bring new algorithms from simulation to production
- Convert double-precision systems to single precision
- Save resources
  - Less memory footprint
  - Double precision not optimally supported in many targets

3. Target vector engines

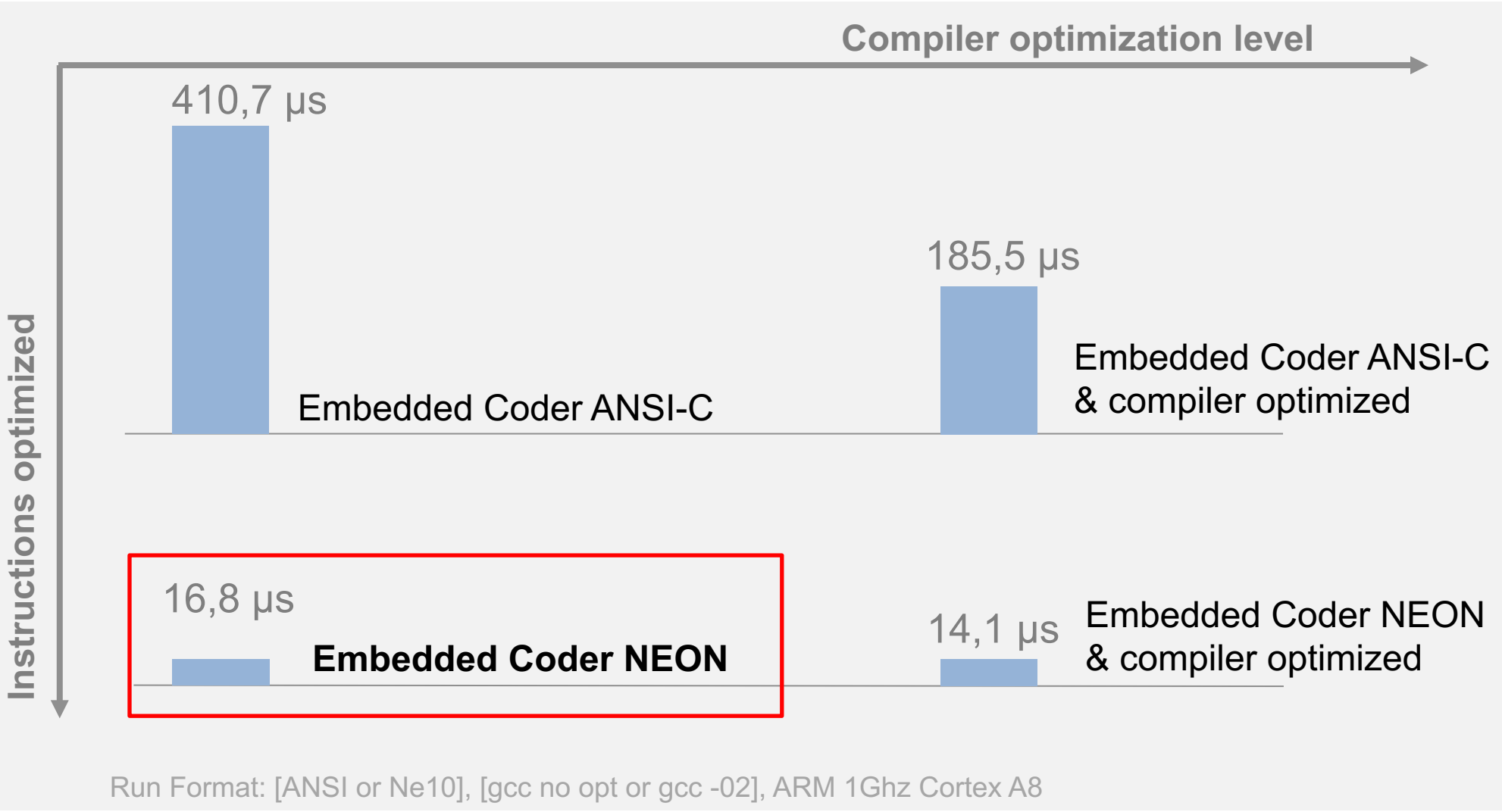
Replace time-consuming code with vector instructions



- Optimize code for your target
- Generate highly optimized code using vector instruction sets
- Increase real-time execution efficiency

4. Target vector engines

Execution times of a FIR filter - PIL benchmark results, ARM Cortex-A



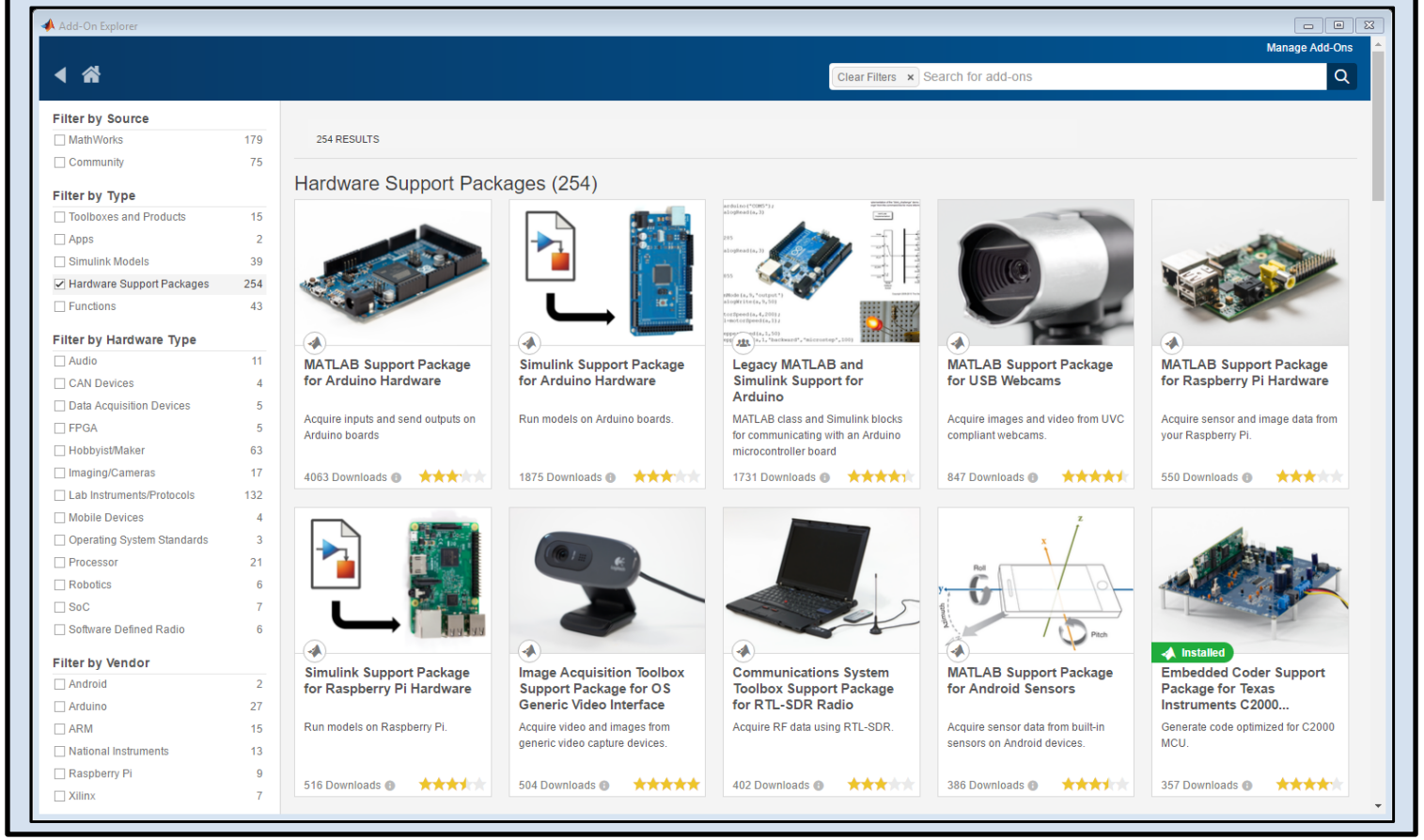
Run Format: [ANSI or Ne10], [gcc no opt or gcc -02], ARM 1Ghz Cortex A8

- Vector instructions have a significant impact on execution time
- This impact can exceed the impact of compiler optimization

4. Use hardware support packages

Download hardware support packages with the Add-On Explorer

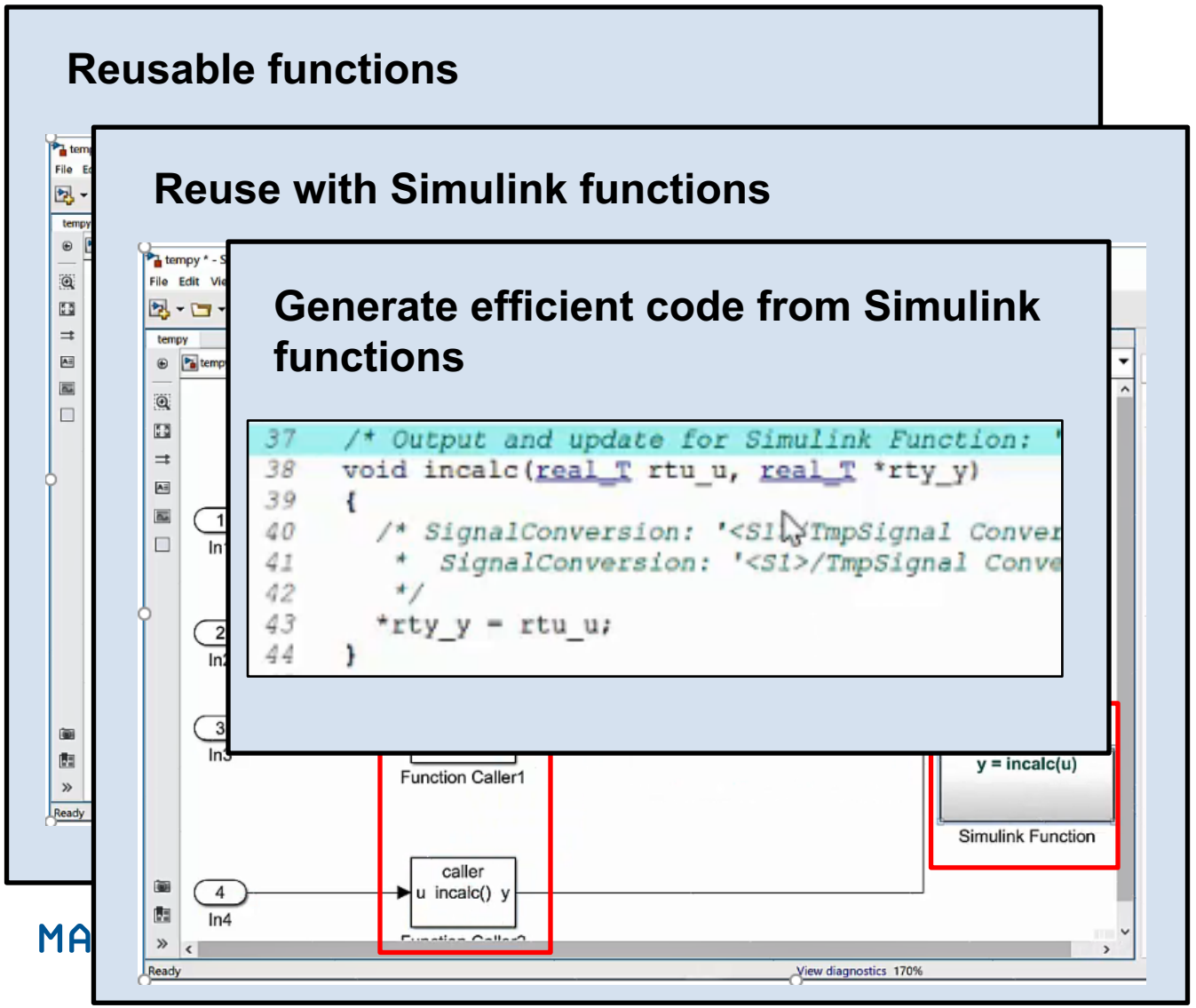
The MATLAB Add-On Explorer



- Fast code adaption to many targets
  - Model is the golden reference for code generation
  - Generated code is optimized for specific targets
- MathWorks package support:
  - ARM, ..., Zynq
- Additional packages:
  - NXP, TI, Infineon, STMicroelectronics,…

5. Reuse components

Reuse with Simulink functions



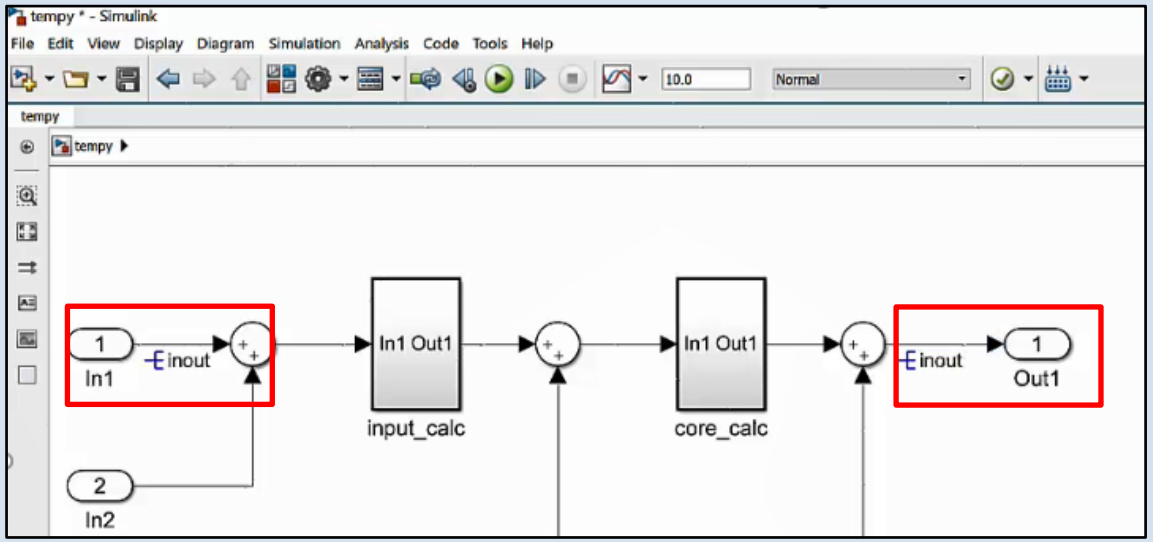
- Clear reusability structure
- Generate compact and efficient code

6. Reduce variables

New options for global RAM optimization

1) Pass scalar output as individual argument

2) Reuse input signals for output

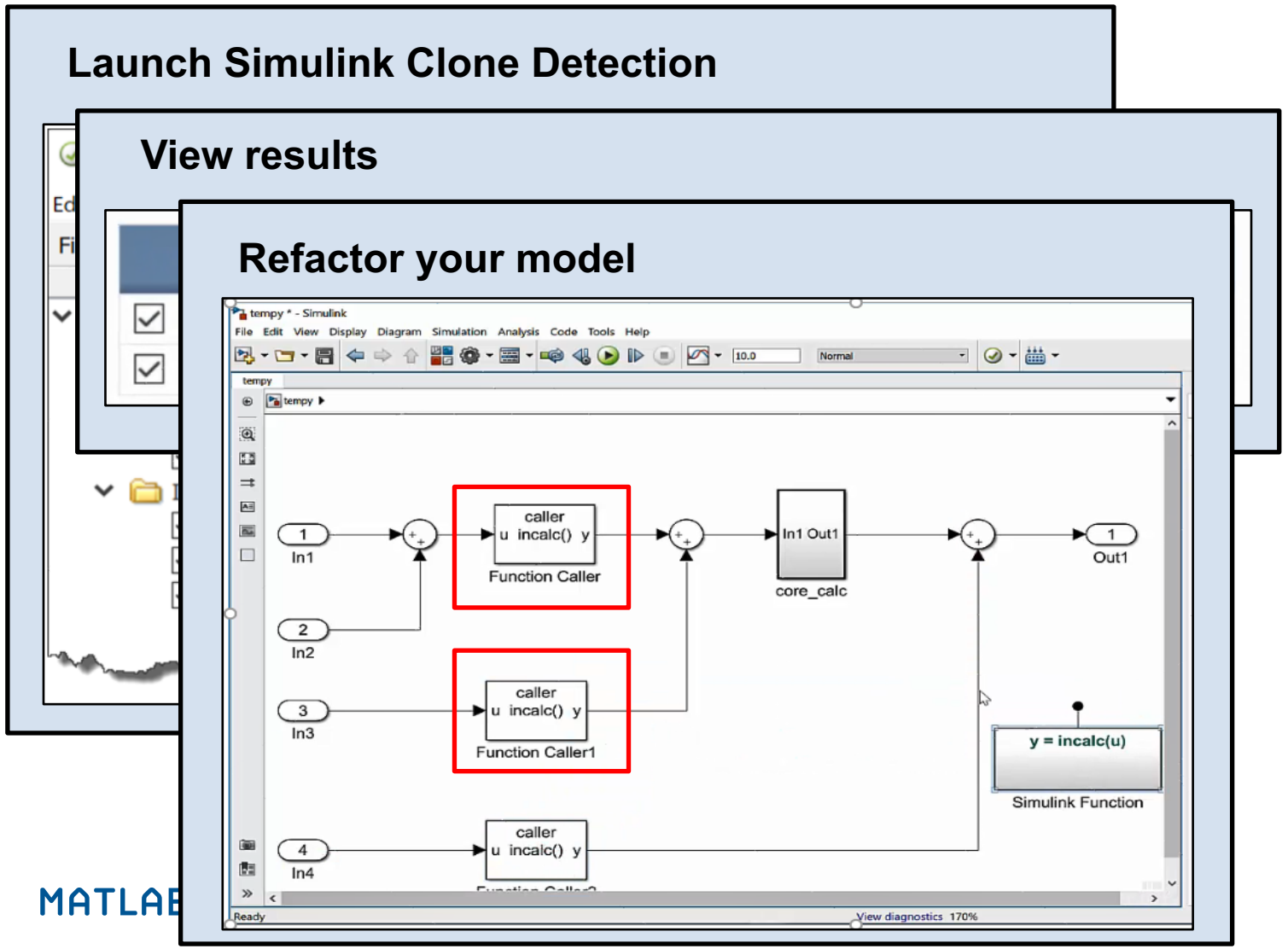



```
44  real_T input_calc(real_T rtu_In1)
45  {
46    /* Gain: '<S2>/Gain' */
47    return 2.0 * rtu_In1;
48  }
```

- Reduced RAM usage
  - No additional variables needed for intermediate results in both cases

7. Reduce logic

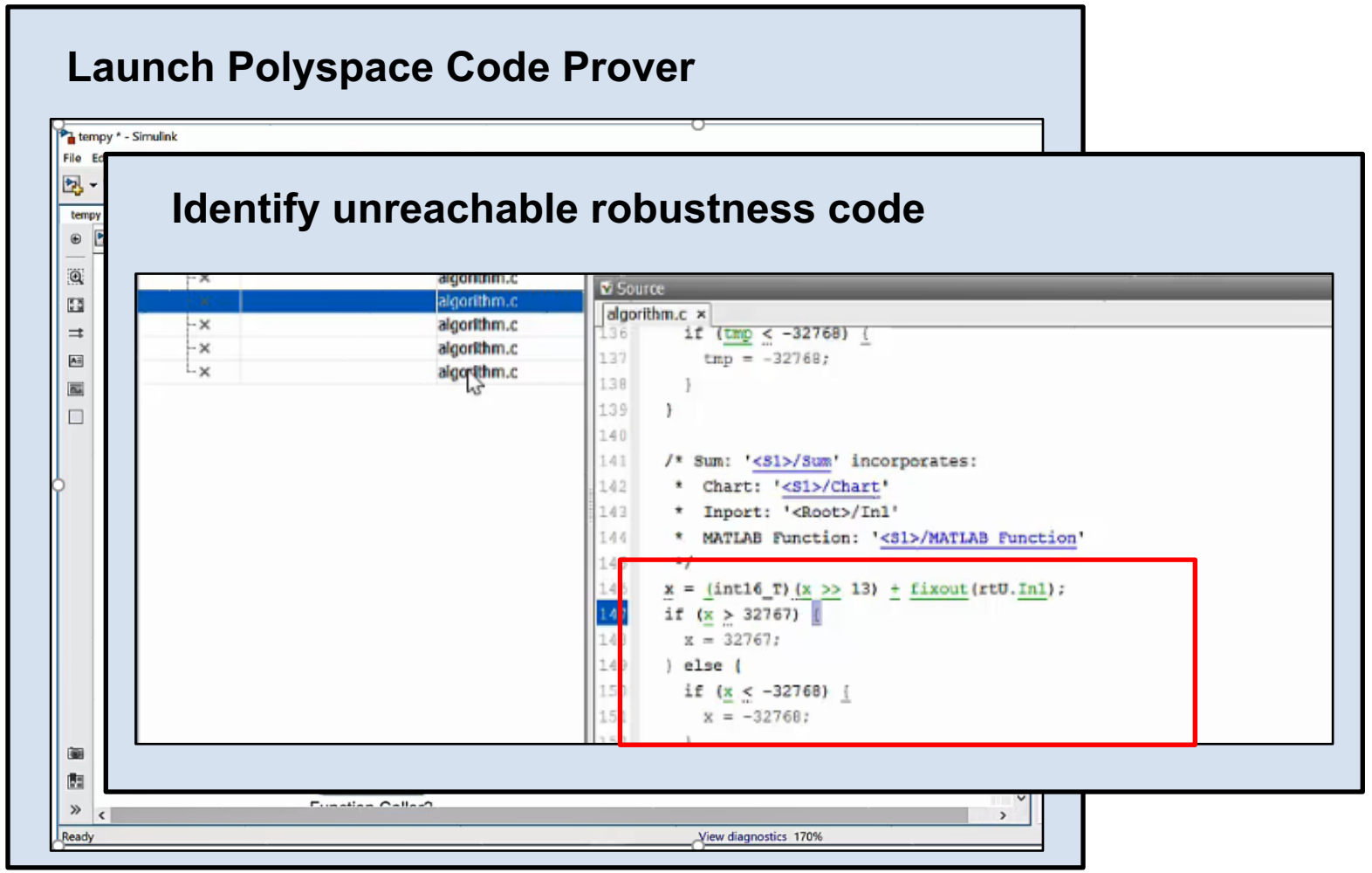
Enable component reuse with *Simulink Clone Detection*



- Identify modeling clones
- Improve model componentization
- Enable Reuse:
  Replace clones with
  - Simulink functions or
  - library blocks

7. Reduce logic

Polyspace Code Prover



- Remove unnecessary robustness code
- Analyze generated and hand written C/C++ source code without program execution
- Prove absence of run-time errors, e.g. overflow

Solution Summary

Accelerating the software development process with automatic code generation

Optimization Techniques

1. Use optimal settings
2. Optimize data types
3. Target vector engines
4. Use hardware support packages
5. Reuse components
6. Reduce variables
7. Reduce logic

*The code generated with Embedded Coder required about **16% less RAM** than the handwritten code used on a previous version of the ECU; the code met all project requirements for efficiency and structure.*
Mario Wünsche, Daimler

Daimler Designs Cruise Controller for Mercedes-Benz Trucks