MATLAB & Simulink Techforum &
2018 EXPO

5G、無人科技、工業物聯、智慧超未來

Automatic C Code Generation from MATLAB

- Generating readable and portable C code easily

Phoebe Li, Terasoft Inc.

TeraSoft 鈦思科技 www.terasoft.com.tw

MathWorks

Agenda

- Motivation
  - Why translate MATLAB to C?
  - Challenges of manual translation
- Using MATLAB Coder
  - Three-step workflow for generating code
- Use cases
  - Integrate algorithms using source code/libraries
  - Accelerate through MEX
  - Prototype by generating EXE
- Conclusion
  - Working with Fixed Point Designer, Simulink and Embedded Coder
  - Verification with SystemVerilog DPI-C Component
  - Other deployment solutions

Why Engineers Translate MATLAB to C Today

.c

Implement C code on processors or hand off to software engineers

.lib
.dll

Integrate MATLAB algorithms with existing C environment using source code and static/dynamic libraries

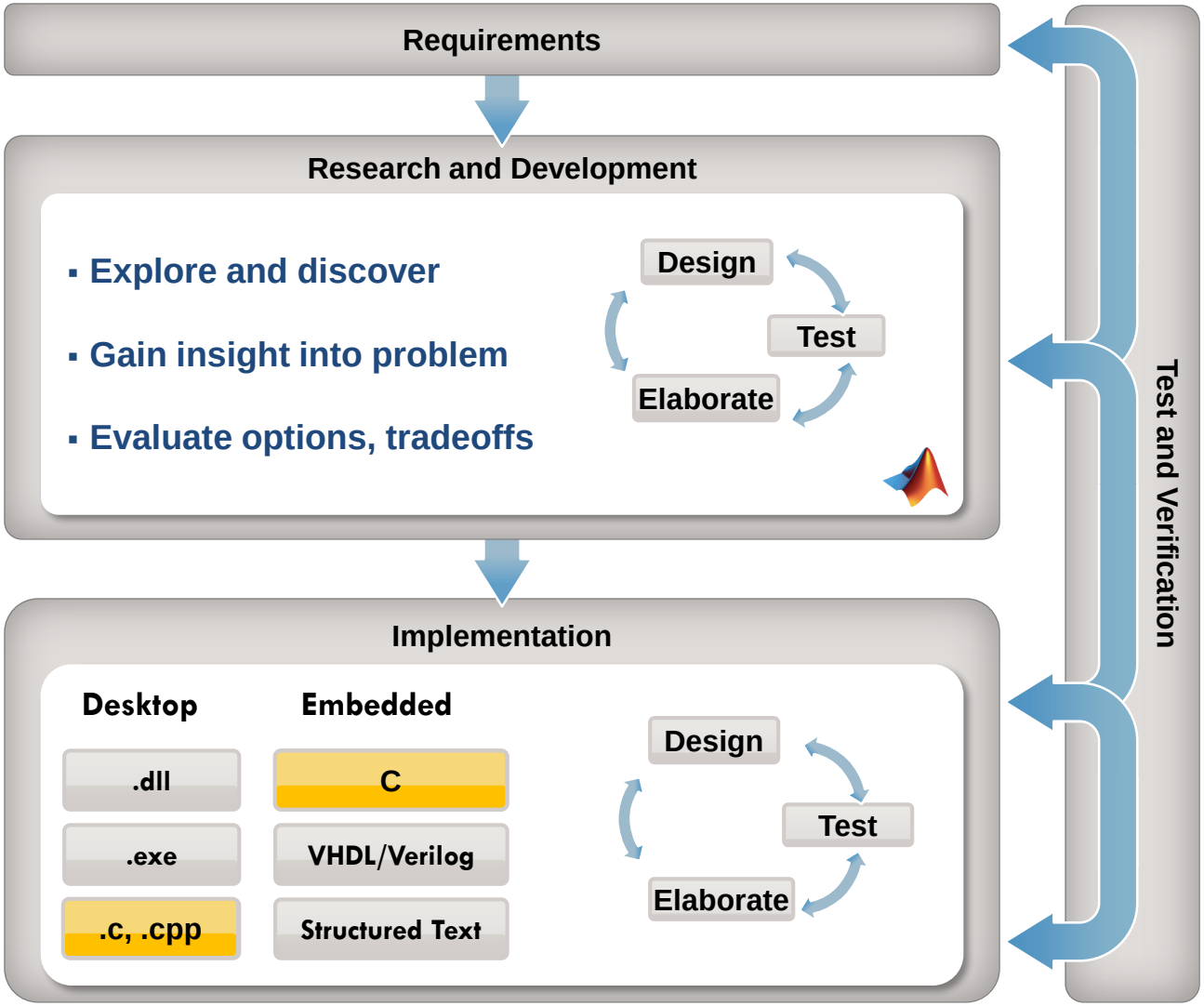
.exe

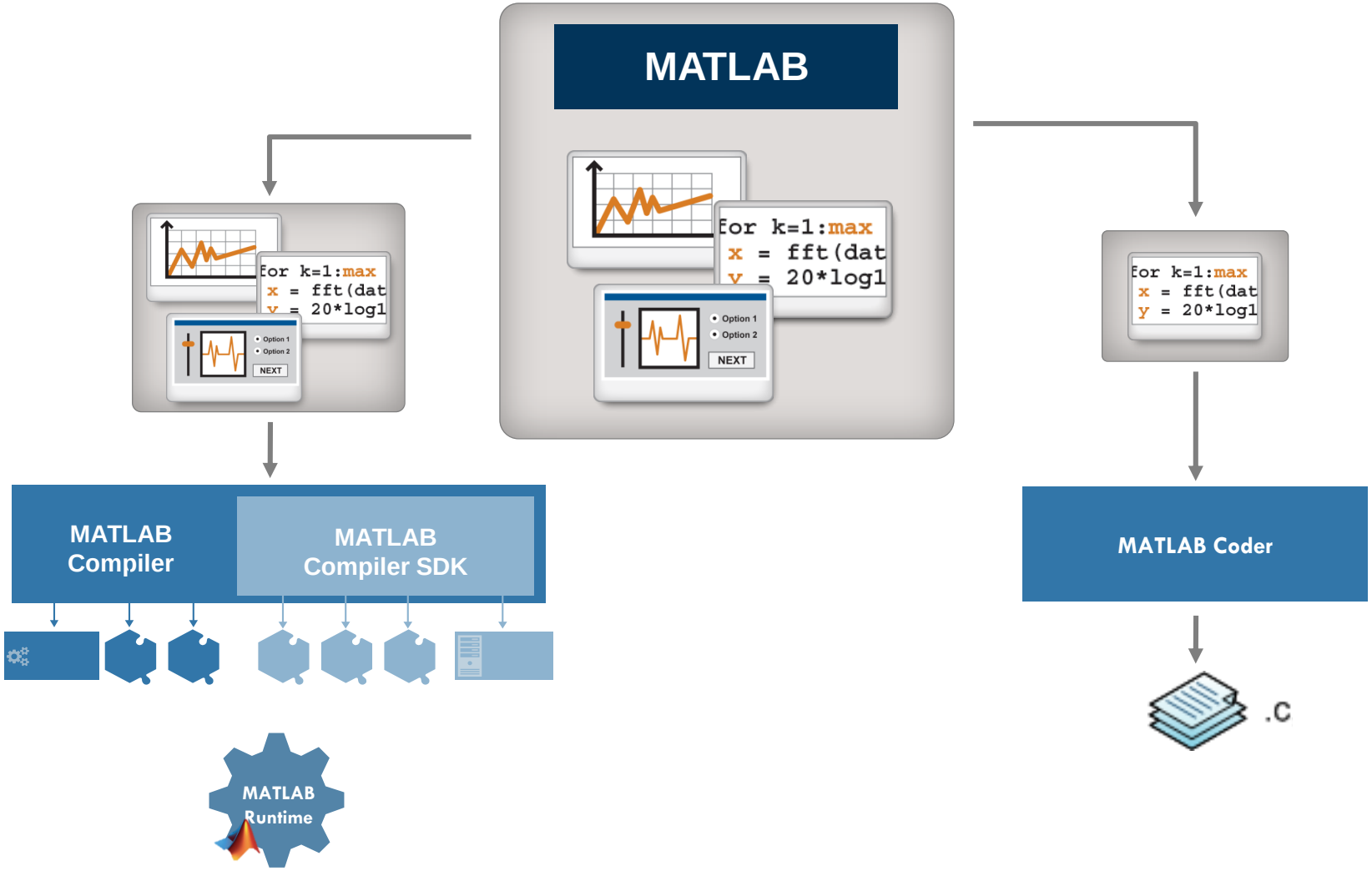
Prototype MATLAB algorithms on desktops as standalone executables

MEX

Accelerate user-written MATLAB algorithms

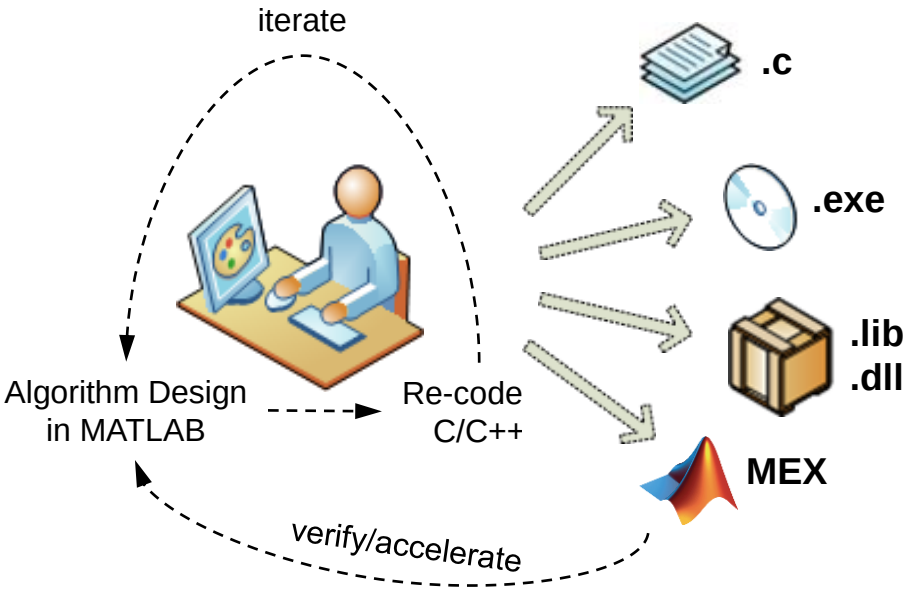
Algorithm Development Process




MATLAB
for k=1:max
x = fft(dat
y = 20*log1
Option 1
Option 2
NEXT
MATLAB
Compiler
MATLAB
Compiler SDK
MATLAB
Runtime
MATLAB Coder
.C

Challenges with Manual Translation

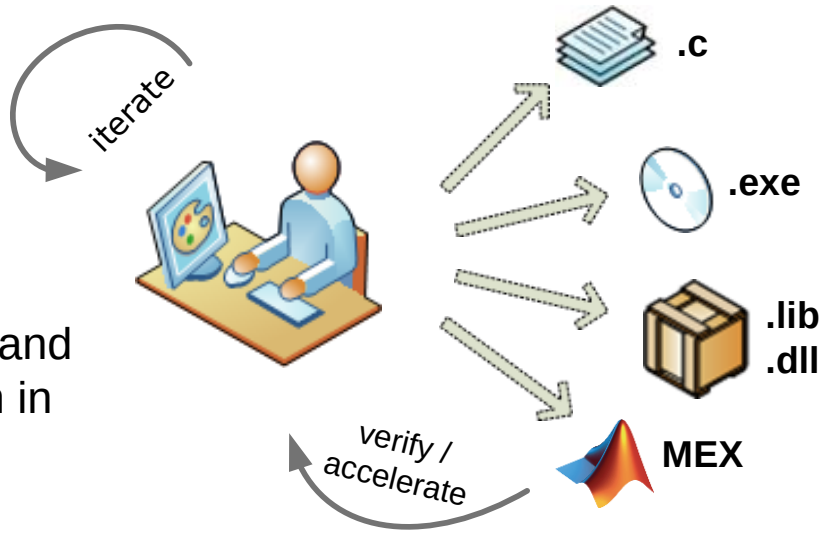
from MATLAB to C



- Separate functional and implementation specification
  - Leads to multiple implementations that are inconsistent
  - Hard to modify requirements during development
  - Difficult to keep reference MATLAB code and C code in sync
- Manual coding errors
- Time-consuming and expensive process

Automatic Translation of MATLAB to C

Algorithm Design and Code Generation in MATLAB



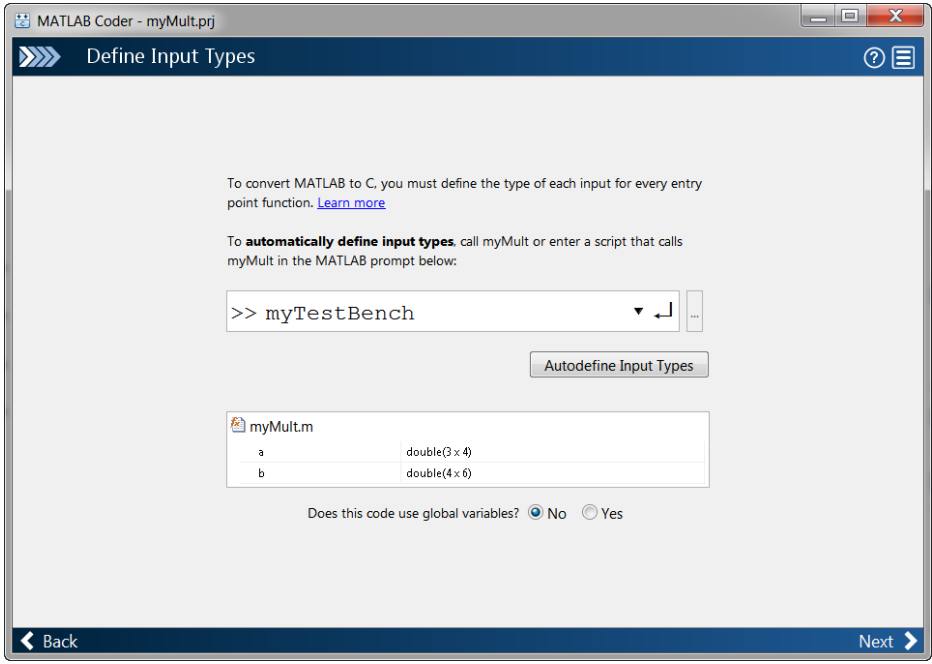
With MATLAB Coder, design engineers can:

- Maintain one design in MATLAB
- Design faster and get to C quickly
- Test more systematically and frequently
- Spend more time improving algorithms in MATLAB

Simple Demo

```
c = a*b
```

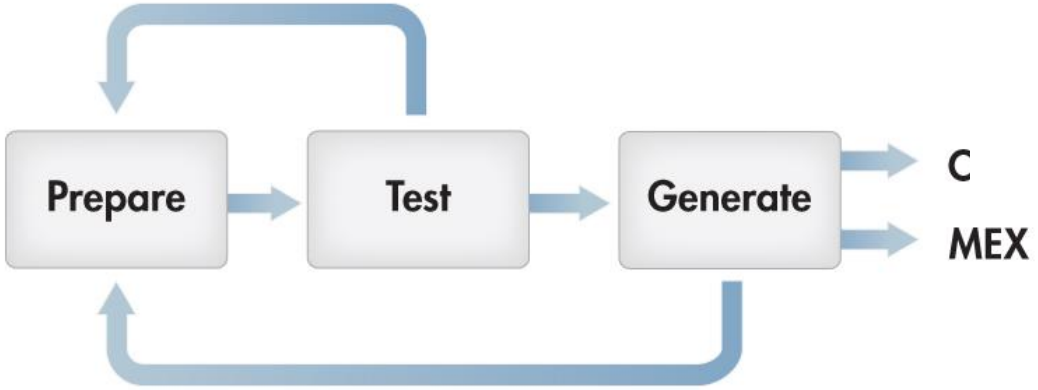
- MATLAB Coder app
- Autodefine input type
- Check for Run-Time issues
- Code generation report



Agenda

- Motivation
  - Why translate MATLAB to C?
  - Challenges of manual translation
- Using MATLAB Coder
  - Three-step workflow for generating code
- Use cases
  - Integrate algorithms using source code/libraries
  - Accelerate through MEX
  - Prototype by generating EXE
- Conclusion
  - Working with Fixed Point Designer, Simulink and Embedded Coder
  - Verification with SystemVerilog DPI-C Component
  - Other deployment solutions

Using MATLAB Coder: Three-Step Workflo



Prepare your MATLAB algorithm for code generation

- Make implementation choices
- Use supported language features

Test if your MATLAB code is ready for code generation

- Validate that MATLAB program generates code
- Accelerate execution of user-written algorithm

Generate source code or MEX for final use

- Iterate your MATLAB code to optimize
- Implement as source, executable, or library

Implementation Considerations

```
function a= foo(b,c)
a = b * c;
```

- Element by element multiply
- Dot product
- Matrix multiply

logical
integer
real
complex
…

C

```
double foo(double b, double c)
{
   return b*c;
}
```

```
void foo(const double b[15],
         const double c[30], double a[18])
{
  int i0, i1, i2;
  for (i0 = 0; i0 < 3; i0++) {
    for (i1 = 0; i1 < 6; i1++) {
      a[i0 + 3 * i1] = 0.0;
      for (i2 = 0; i2 < 5; i2++) {
        a[i0 + 3 * i1] += b[i0 + 3 * i2] * c[i2 + 5 * i1];
      }
    }
  }
}
```

Implementation Considerations

- Polymorphism
- Memory allocation
- Processing matrices and arrays
- Fixed-point data types

7	Lines of MATLAB
105	**Lines of C**

```
function [x_est, p_est] = kalman_estimate(R,H,x_prd,p_prd,z)
  S = H * p_prd' * H' + R;
  B = H * p_prd';
  klm_gain =  (S \ B)';
  x_est = x_prd + klm_gain * (z - H * x_prd);
  p_est = p_prd - klm_gain * H * p_prd;
```

```
#include "kalman_estimate.h
void kalman_estimate(const
                     const
                     double
{
  double klm_gain[12];
  int r1;
  int r2;
  int k;
  double S[4];
  double a21;
  double B[12];
  double a22;
  double Y[12];
  double b_z[2];
  double b_klm_gain[36];
  for (r1 = 0; r1 < 2; r1++) {
    for (r2 = 0; r2 < 6; r2++) {
      klm_gain[r1 + (r2 << 1)] = 0.0;
```

```
    for (k = 0; k < 6; k++) {
      klm_gain[r1 + (r2 << 1)] +=
    }
  }
}

for (r1 = 0; r1 < 2;
  for (r2 = 0; r2 <
    a21 = 0.0;
    for (k = 0; k <
      a21 += klm_gai
    }

    S[r1 + (r2 << 1)
  }
}

for (r1 = 0; r1 < 2;
  for (r2 = 0; r2 <
```

```
if (fabs(S[1
  r1 = 1;
  r2 = 0;
} else {
  r1 = 0;
  r2 = 1;
}

a21 = S[r2] / S[r1];
a22 = S[2 + r2] - a21 * S[2 + r1];
for (k = 0; k < 6; k++) {
  Y[1 + (k << 1)] = (B[r2 + (k << 1)] - B[r1 + (k << 1)] * a21) / a22;
  Y[k << 1] = (B[r1 + (k << 1)] - Y[1 + (k << 1)] * S[2 + r1]) / S[r1];
}

for (r1 = 0; r1 < 2; r1++) {
  for (r2 = 0; r2 < 6; r2++) {
    klm_gain[r2 + 6 * r1] = Y[r1 + (r2 << 1)];
  }
}
```

```
for (r1 = 0; r1 < 6; r1++) {
  for (r2 = 0; r2 < 6; r2++) {
    b_klm_gain[r1 + 6 * r2] = 0.0;
    for (k = 0; k < 2; k++) {
      b_klm_gain[r1 + 6 * r2] += klm_gain[r1 + 6 * k] * H[k + (r2 << 1)];
    }
  }
}

for (r1 = 0; r1 < 6; r1++) {
  for (r2 = 0; r2 < 6; r2++) {
    a21 = 0.0;
    for (k = 0; k < 6; k++) {
      a21 += b_klm_gain[r1 + 6 * k] * p_prd[k + 6 * r2];
    }

    p_est[r1 + 6 * r2] = p_prd[r1 + 6 * r2] - a21;
  }
}
```

Example: Newton/Raphson Algorithm

Preparing your MATLAB code

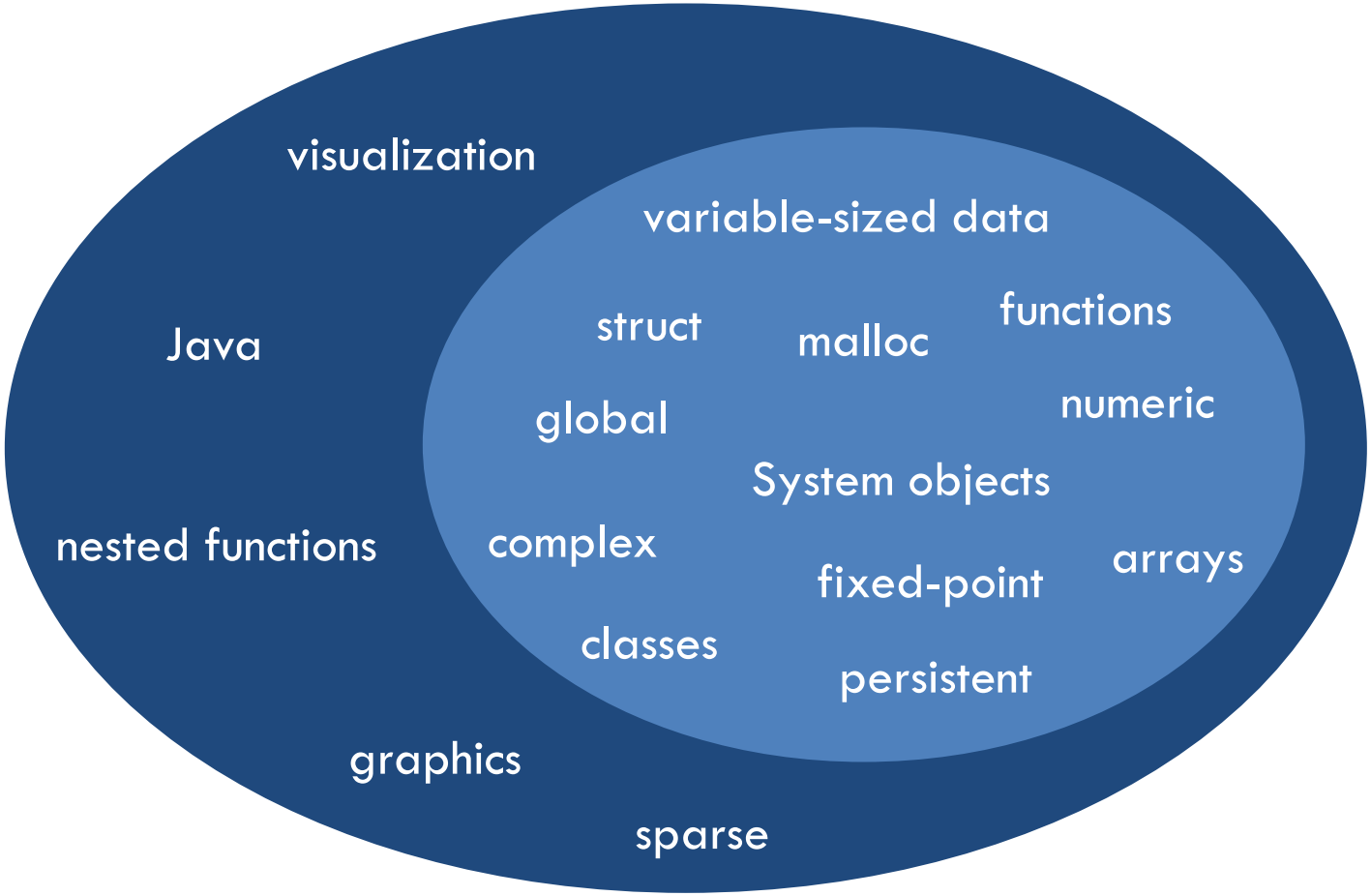
$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

- Pre-allocate
- Identify more efficient constructs
- Select code generation options

```
function [x,h] = newtonSearchAlgorithm(b,n,tol)
% Given, "a", this function finds the nth root of a
% number by finding where: x^n-a=0.

    notDone = 1;
    aNew    = 0; %Refined Guess Initialization
    a       = 1; %Initial Guess
    cnt     = 0;
    h(1)=a;
    while notDone
        cnt = cnt+1;
        [curVal,slope] = f_and_df(a,b,n); %square
        yint = curVal-slope*a;
        aNew = -yint/slope; %The new guess
        h(cnt)=aNew;
        if (abs(aNew-a) < tol) %Break if it's converged
            notDone = 0;
        elseif cnt>49 %after 50 iterations, stop
            notDone = 0;
            aNew = 0;
```

MATLAB Language Support for Code Generation

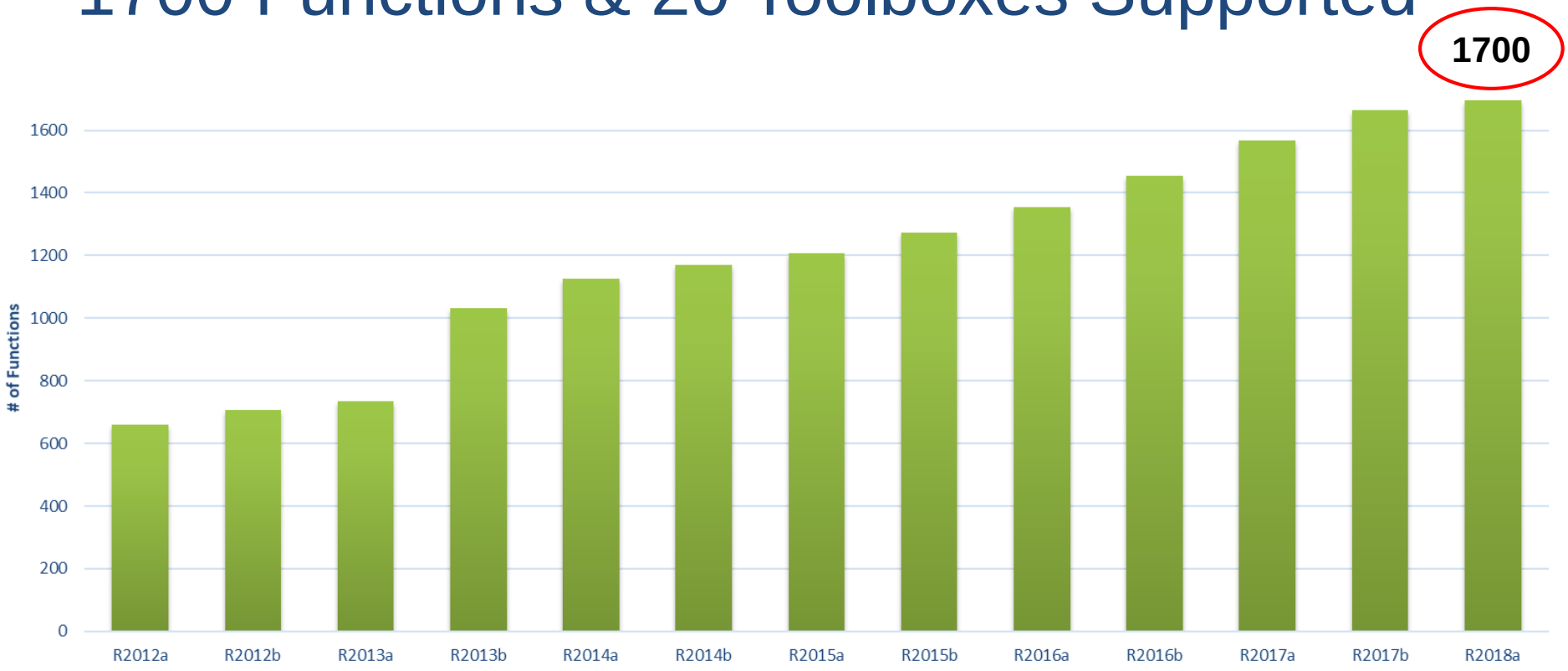


Supported MATLAB Language Features and Functions

Broad set of language features and functions supported for code generation

Matrices and Arrays	Data Types	Programming Constructs	Functions
• Matrix operations • N-dimensional arrays • Subscripting • Frames • Persistent variables • Global variables	• Complex numbers • Integer math • Double/single-precision • Fixed-point arithmetic • Characters • Structures • Cell arrays • Numeric class • Variable-sized data • MATLAB Class • System objects	• Arithmetic, relational, and logical operators • Program control (`if`, `for`, `while`, `switch`)	• MATLAB functions and subfunctions • Variable-length argument lists • Anonymous functions • Nested functions • Function handles

1700 Functions & 20 Toolboxes Supported



- Aerospace Toolbox
- Audio System Toolbox
- Automated Driving System Toolbox
- Communications System Toolbox
- Computer Vision System Toolbox
- Control System Toolbox
- DSP System Toolbox
- Fixed-Point Designer
- Image Acquisition Toolbox
- Image Processing Toolbox
- Model Predictive Control Toolbox
- Neural Networks Toolbox
- Optimization Toolbox
- Phased Array System Toolbox
- Robotics System Toolbox
- Signal Processing Toolbox
- Stats & Machine Learning Toolbox
- System Identification Toolbox
- Wavelet Toolbox
- WLAN System Toolbox

Agenda

- Motivation
  - Why translate MATLAB to C?
  - Challenges of manual translation
- Using MATLAB Coder
  - Three-step workflow for generating code
- Use cases
  - Integrate algorithms using source code/libraries
  - Accelerate through MEX
  - Prototype by generating EXE
- Conclusion
  - Working with Fixed Point Designer, Simulink and Embedded Coder
  - Verification with SystemVerilog DPI-C Component
  - Other deployment solutions

MATLAB Coder Use Cases

.lib
.dll

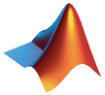
.exe

Integrate
algorithms with custom software

Prototype
algorithms on PCs

Accelerate
algorithm execution

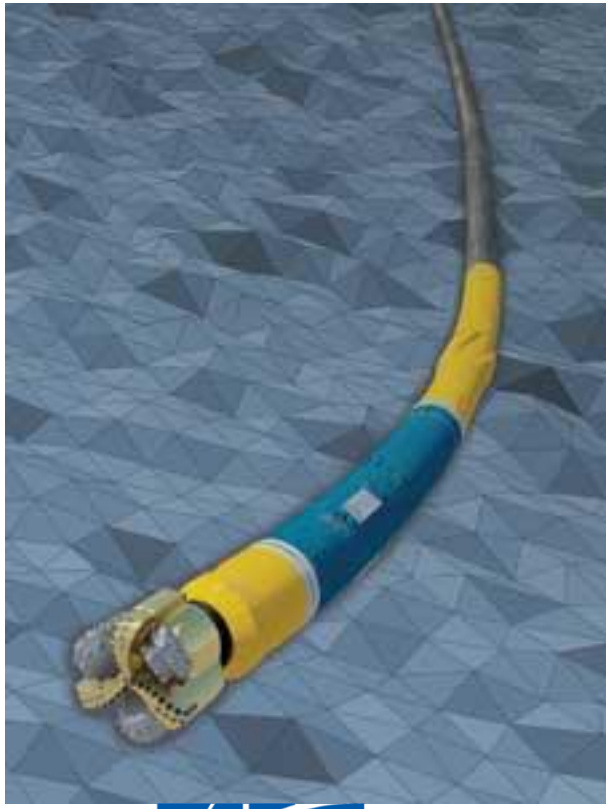
Implement
algorithms on embedded processors



MEX

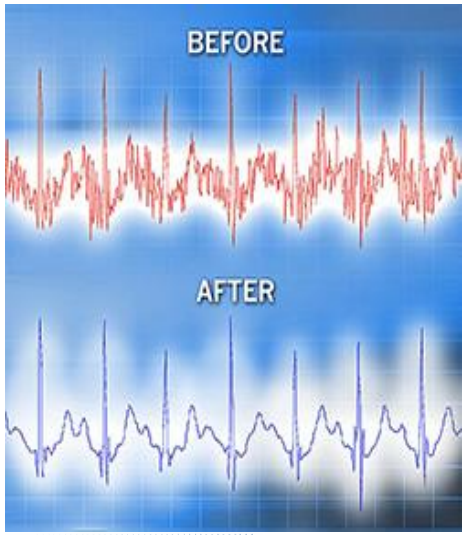
.c

Examples of MATLAB Coder Usage



BAKER HUGHES

DELPHI



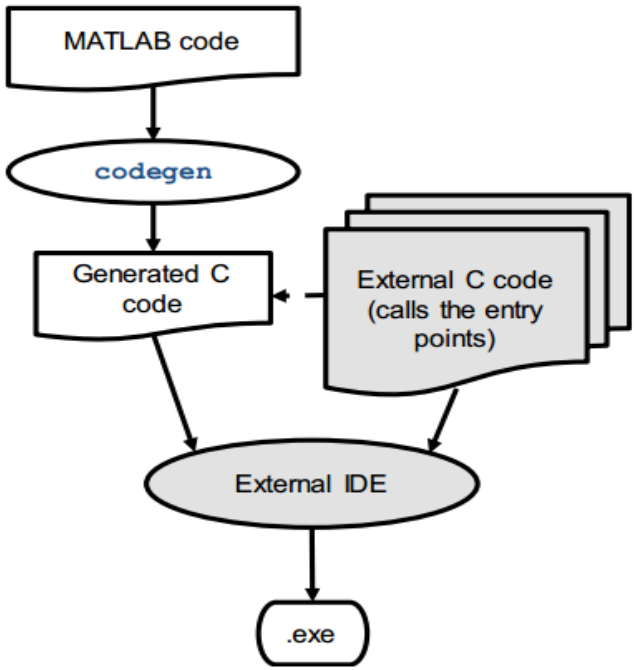
VivaQuant™
Quantifying vital signs. Accurately.

iSonea
Sound Asthma Management

dorsaVi
Inspiring the world to move

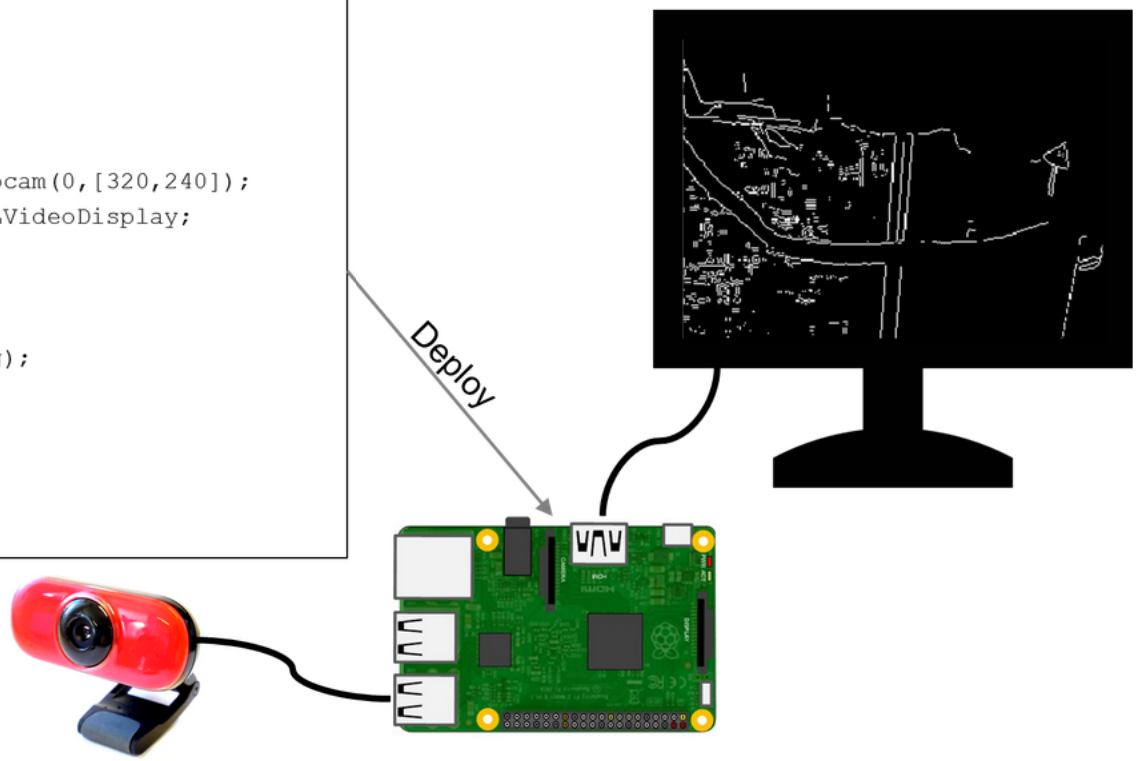
Integrating External C Code Using an External IDE

- Generate C code from your MATLAB function.
- Call entry points in your external code.
- Add appropriate **#include** directives in your code.
- Add generated code and all its dependencies to your IDE project.
- Compile the project to generate a standalone executable.
- Run the executable and verify its results.



Example: Integrating Generated Code into Raspberry Pi

```
function myApplication()
   %#codegen
   persistent w
   persistent d
   if isempty(w)
       w = matlab.raspi.webcam(0,[320,240]);
       d = matlab.raspi.SDLVideoDisplay;
   end
   for k = 1:1000
       img = snapshot(w);
       img = edgeDetect(img);
       displayImage(d,img);
   end
   release(w);
   release(d);
end
```



Acceleration Strategies

- Better algorithms

  Matrix inversion vs. QR or SVD
  - Different approaches to solving the same problem

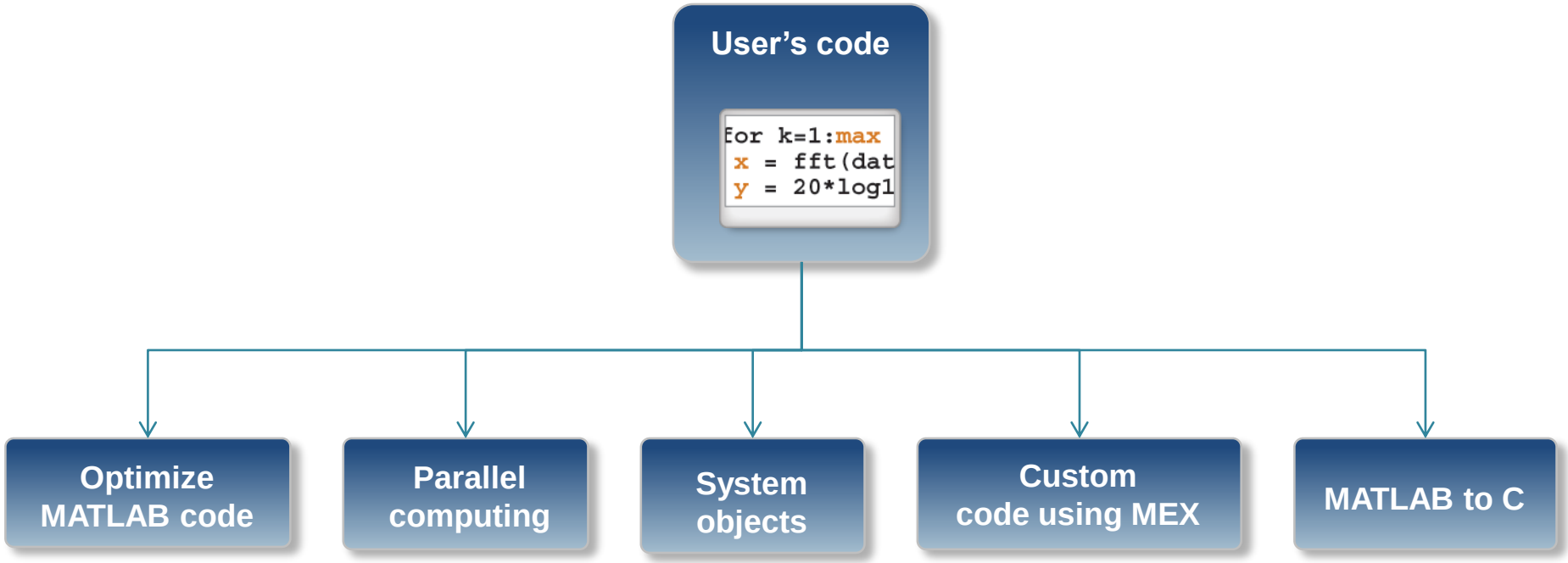
- More efficient implementation

  Hand-coded vs. optimized library (BLAS and LAPACK)
  - Different optimization of the same algorithm

- More computational resources

  Single-threaded vs. multithreaded (multithreaded BLAS)
  - Leveraging additional processors, cores, GPUs, FPGAs, etc.

Accelerating Algorithm Execution



Acceleration Using MEX

- Speed-up factor will vary

- When you **may** see a speedup:
  - Often for communications and signal processing
  - Always for fixed point
  - Likely for loops with states or when vectorization isn't possible

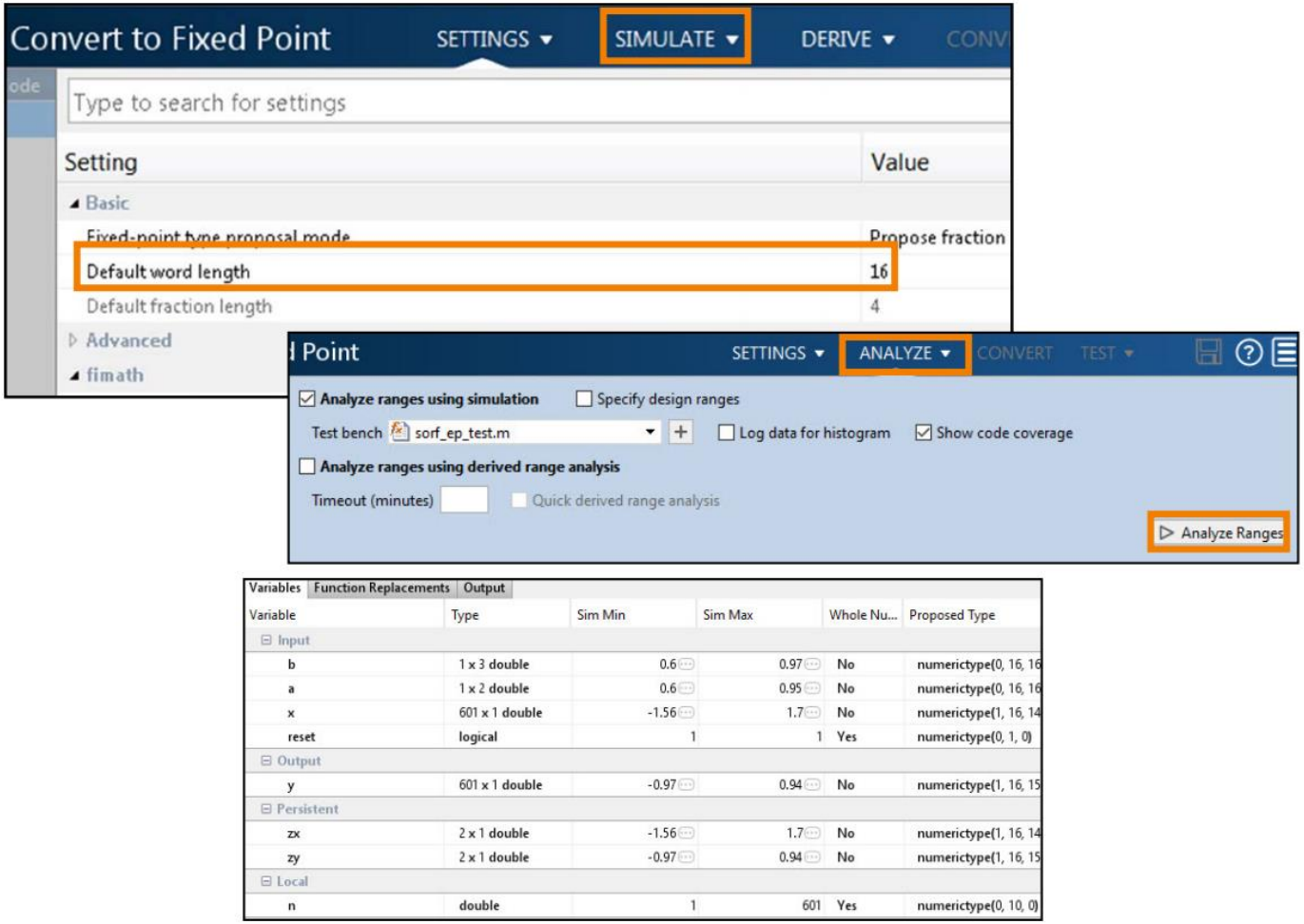
- When you **may not** see a speedup:
  - MATLAB implicitly multithreads computation.
  - Built-functions call IPP or BLAS libraries.

Agenda

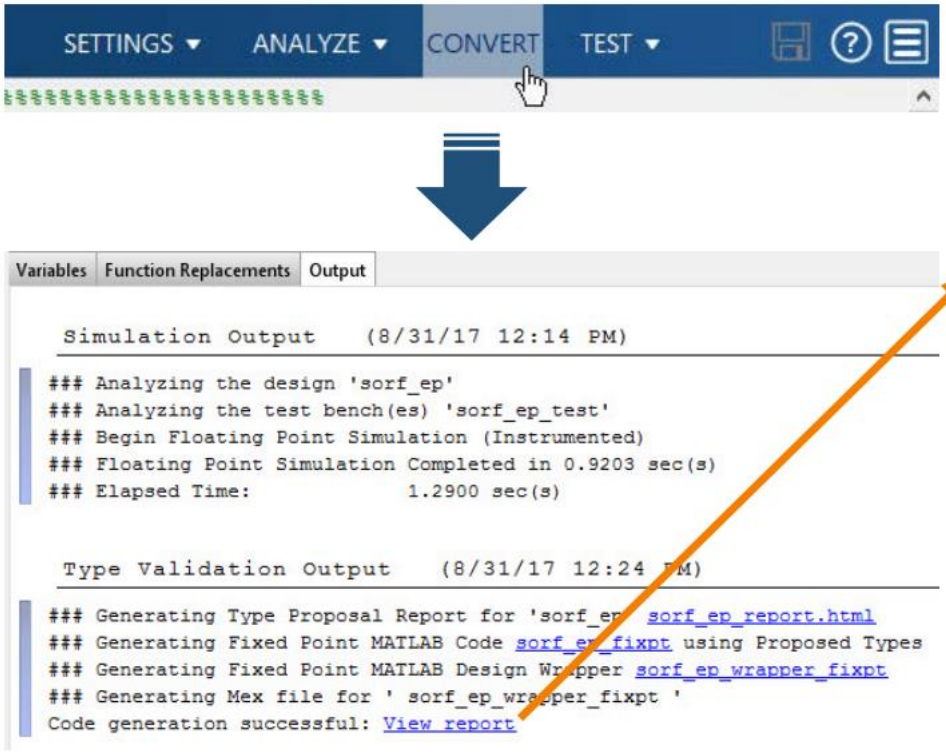
- Motivation
  - Why translate MATLAB to C?
  - Challenges of manual translation
- Using MATLAB Coder
  - Three-step workflow for generating code
- Use cases
  - Integrate algorithms using source code/libraries
  - Accelerate through MEX
  - Prototype by generating EXE
- Conclusion
  - Working with Fixed Point Designer, Simulink and Embedded Coder
  - Verification with SystemVerilog DPI-C Component
  - Other deployment solutions

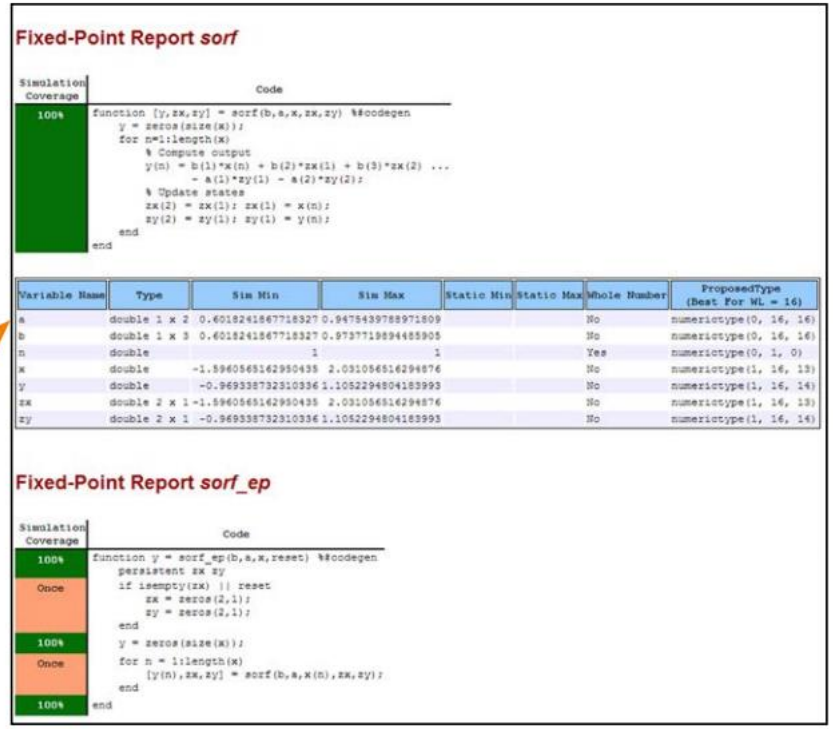
Fixed Point C Code Generation

- Computing Simulation Ranges



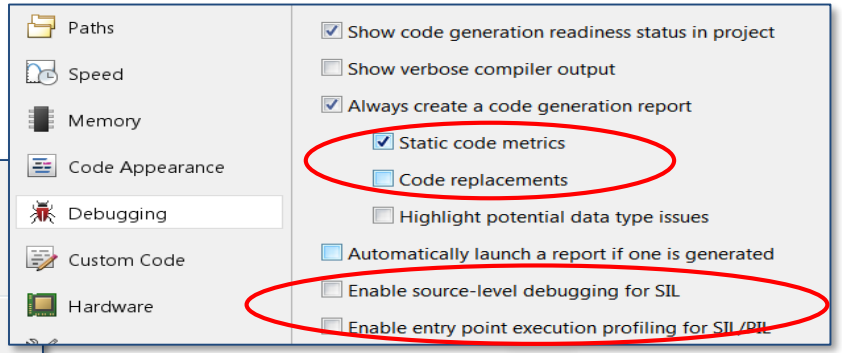
Fixed Point C Code Generation - Validating Proposed Data Types

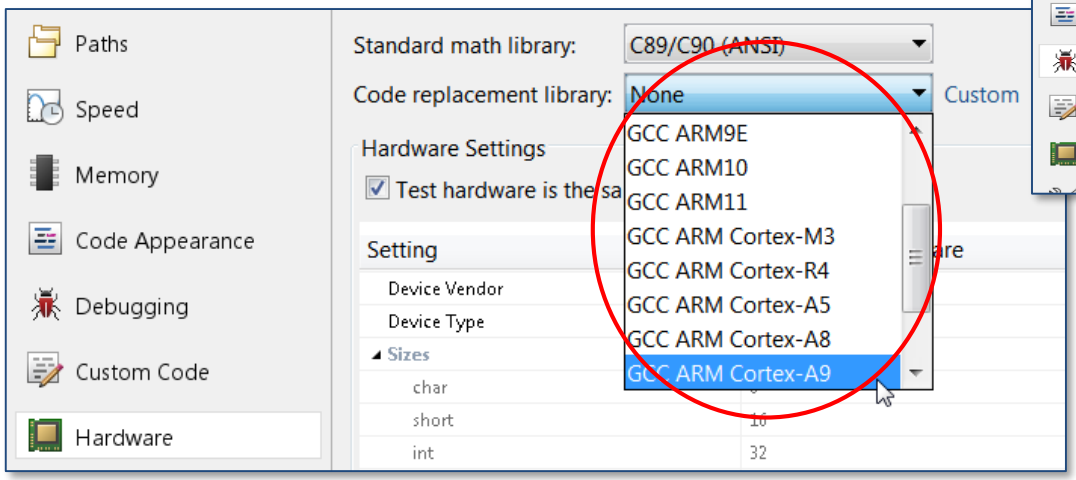


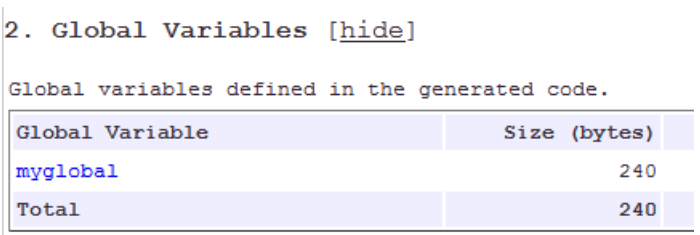


Working with Embedded Coder

- Advanced support for MATLAB Coder, including:
  - Speed & Memory
  - Code appearance
  - Hardware-specific optimization
  - Software/Processor-in-the-loop verification
  - Execution profiling

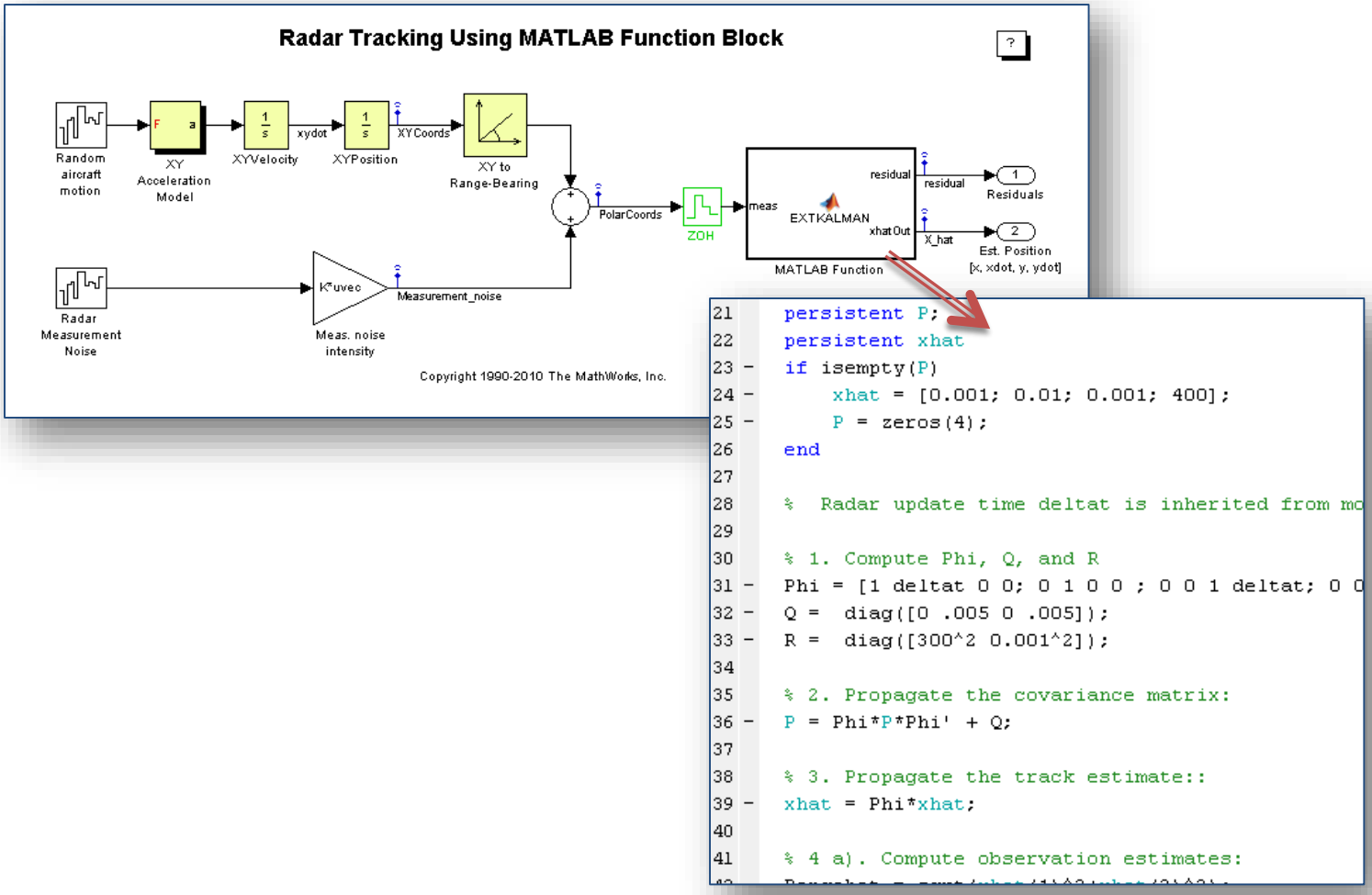






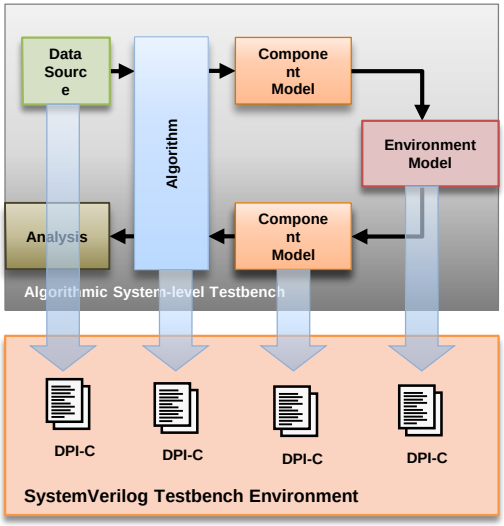
Working with Simulink and Embedded Coder

MATLAB Function block in Simulink



SystemVerilog DPI-C Component Generation

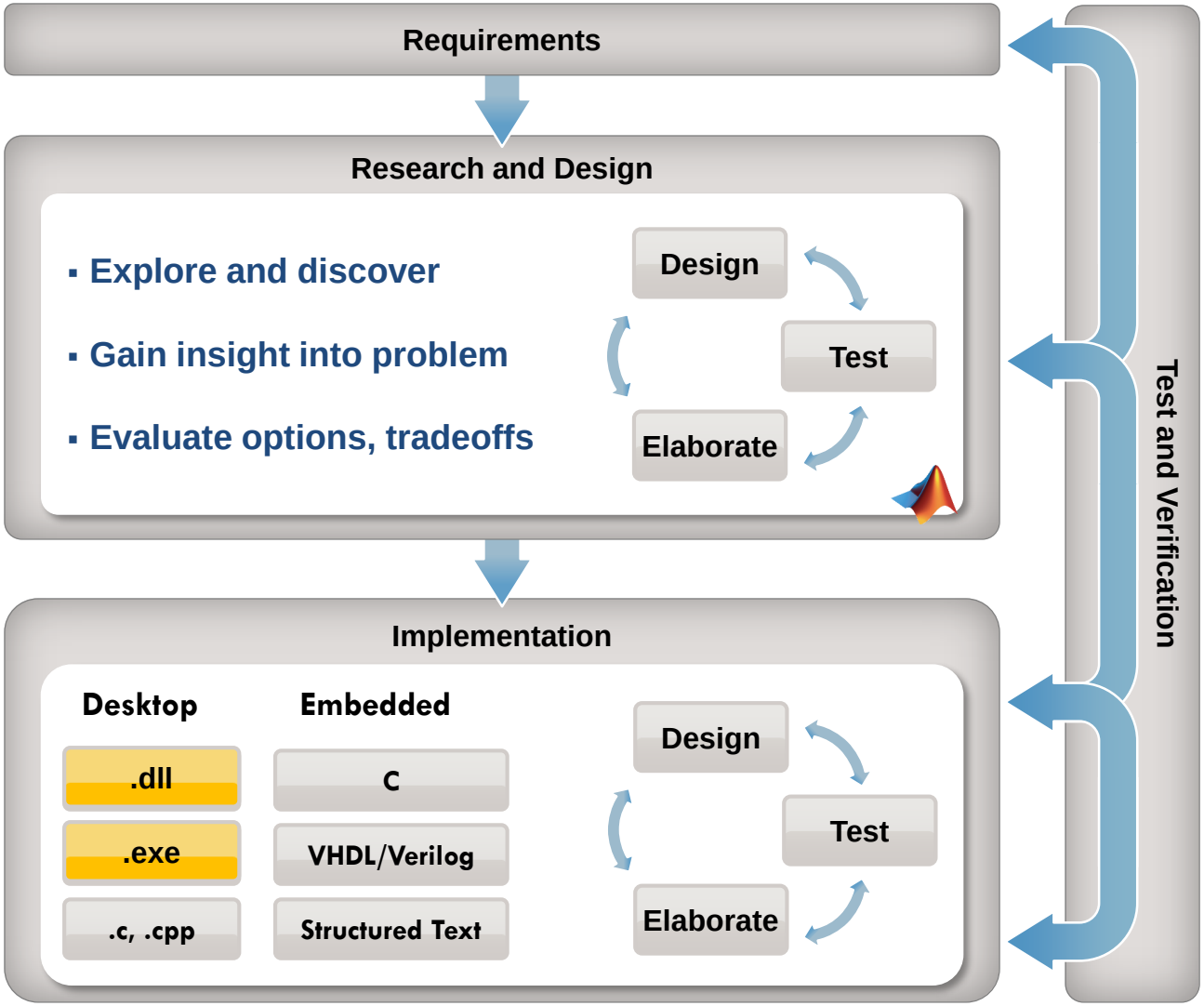
-Reuse of models in SystemVerilog Testbench



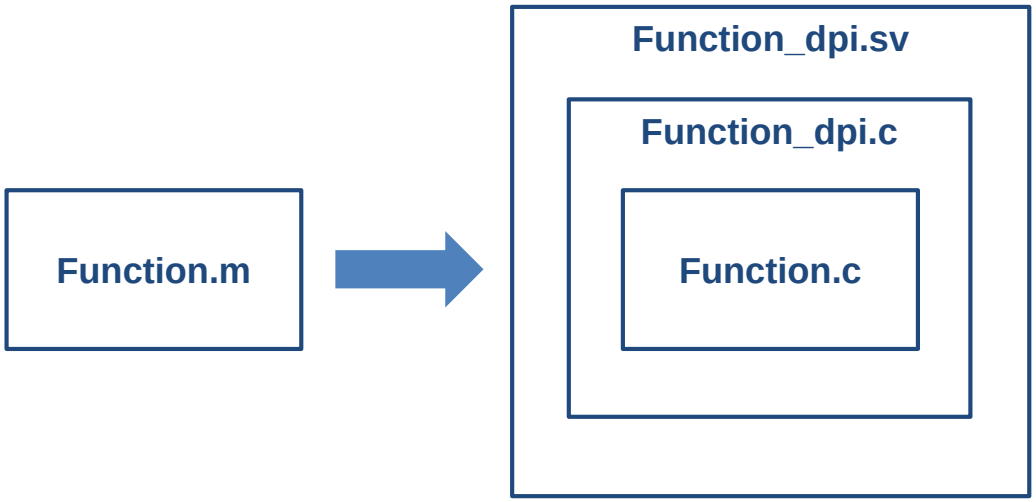
HDL Verifier
MATLAB Coder

- Develop
  - System components (IP and test benches) in Simulink and MATLAB
  - Model, Simulate, and Verify
- Export
  - Components as C code with SystemVerilog wrappers
- Integrate
  - Components with components in the HDL Simulator or IC design tool
- Verify
  - Verification of the complete system design!

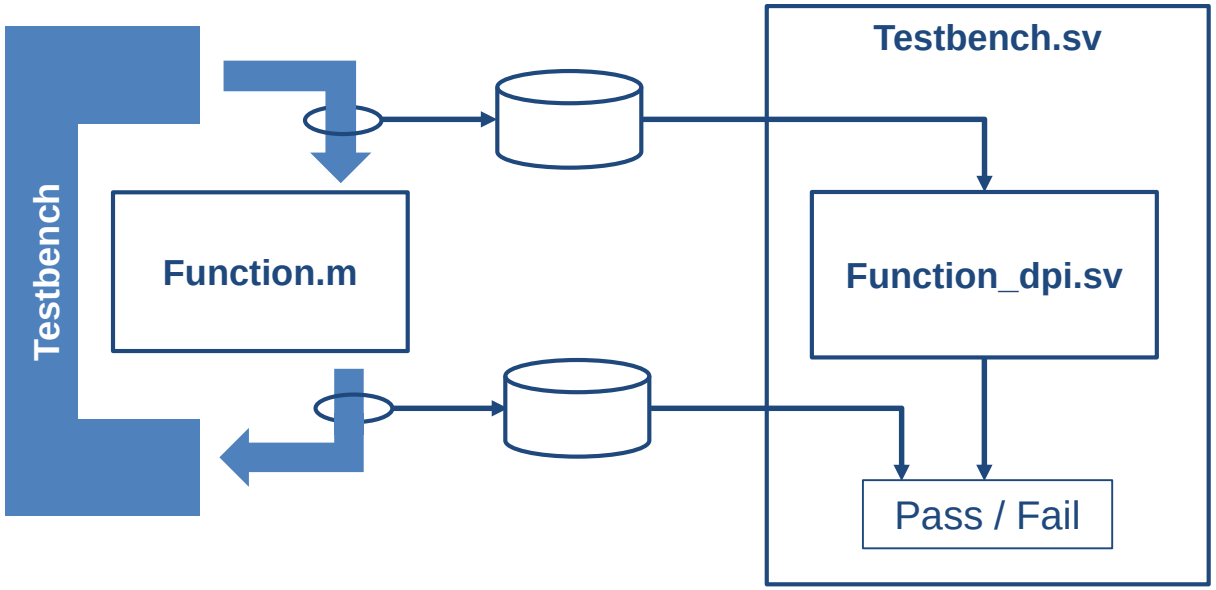
Other Desktop Deployment Options



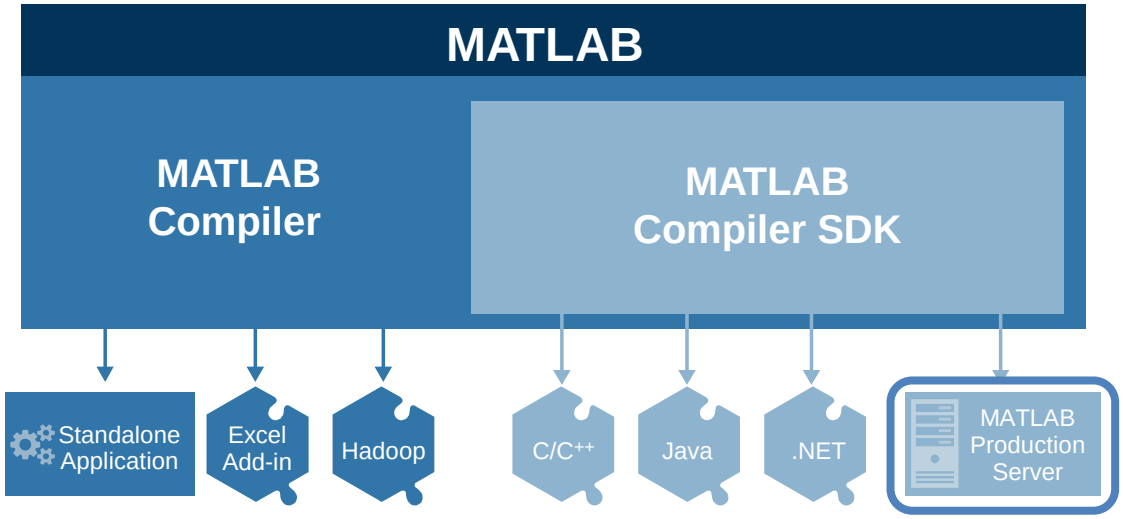
SystemVerilog Component Generation from MATLAB



SystemVerilog Component Verification from MATLAB



Other Deployment Options



MATLAB Compiler for sharing MATLAB programs without integration programming

MATLAB Compiler SDK provides implementation and platform flexibility for software developers

MATLAB Production Server provides the most efficient development path for secure and scalable web and enterprise applications

Choosing the Right Deployment Solution

MATLAB Coder

MATLAB Compiler
MATLAB Compiler SDK

Output	Portable and readable C source code	Software components
MATLAB support	Subset of language Some toolboxes	Full language Most toolboxes Graphics
Additional libraries	None	MATLAB Runtime
License model	Royalty-free	Royalty-free
Extensions	Embedded Coder	MATLAB Production Server

Q&A

MATLAB & Simulink Techforum &

2018 EXPO