Signal Processing with MATLAB: System Simulation and Real-Time Implementation

Jonas Rutström
Senior Application Engineer
Why are we here today?

- Learn more about algorithm and system design in MATLAB and Simulink
  - The why, how, and what…

- Hopefully get some new ideas that makes you work easier
  - There is always something new to learn…

- Give you the possibility to talk with MathWorks representatives
  - Share your thoughts, give us feedback – We are here for you!
Algorithm Development

Hardware Implementation
DEMO
(Kinect)
From MATLAB Script to Real-Time C Code
1. Experiment with algorithm in MATLAB
2. Customize the MATLAB code
3. Generate C/C++ code
4. Verify/validate generated code
5. Optimize generated code
1. Experiment with algorithm in MATLAB
2. Customize the MATLAB code
3. Generate C/C++ code
4. Verify/validate generated code
5. Optimize generated code
A closer look at the algorithm…
ALGORITHM

Sound source

\[ \theta \rightarrow d \]

\[ \text{delay} \]

\[ d \]
How to calculate the delay?
Cross-correlation

Interpolation

Delay
1. Experiment with algorithm in MATLAB

- **Simplify:**
  - Get a static source
  - Take a picture
  - Record the stereo audio input

- **Understand your setup**
  - Distance between microphones?
  - Resonable delay?

- **Implement according to the masterplan**
  - verify the result
DEMO

(MATLAB Script)
1. Experiment with algorithm in MATLAB

```matlab
%% Script for Algorithm Design
imageFile = 'stillMiddle5.jpg';
audioFile = 'stillMiddle5.wav';

%% Read static data from first audio file
% Each data vector here has 4 channels
[data, Fs] = wavread(audioFile);
frameStatic = imread(imageFile);
figure(1), imshw(frameStatic, 'InitialMagnification', 50);

gure(2);
subplot(411), plot(data(:,1)), ylbe('Mic #1');
subplot(412), plot(data(:,2),), ylbe('Mic #2');
subplot(413), plot(data(:,3),), ylbe('Mic #3');
subplot(414), plot(data(:,4),), ylbe('Mic #4');

%% Algorithm
% Select only a buffer at some point in the data (channel 1 and 4)
L = 512;
y = data(10000 + (0:L-1), [1 4]);
% Cross-correlation between inputs at two microphones
[xc, lages] = xcorr(y(:,1), y(:,2));

Command Window
>> algorithmDesign
The measured delay of arrival is -3.24932 samples @ 16 kHz
Equivalent to -0.000203082 seconds
and to an angle of about 20 degrees
>>
```
What is the next step?
1. Experiment with algorithm in MATLAB
2. Customize the MATLAB code
3. Generate C/C++ code
4. Verify/validate generated code
5. Optimize generated code
How do you intend to use the algorithm?
Function + Test bench
2. Customize the MATLAB code

- Testability
- Re-usability
- C code generation
2. Customize the MATLAB code

- Testability
- Re-usability
- C code generation
2. Customize the MATLAB code

- Testability
- Re-usability
- C code generation
DEMO

(Algorithm + Test Bench + Code Generation)
2. Customize the MATLAB Code
Performance?
2. Customize the MATLAB Code

How long did it take?

- tic/toc

```matlab
% start timer
tic

% execute code
out = myFunction(in);

% stop timer (and store elapsed time)
et = toc;
```

Where are the bottlenecks?

- profile

```matlab
% turn on profiler
profile on

% execute code
out = myFunction(in);

% turn off profiler
profile off
% open html report
profile report
```
2. Architect/review/optimize MATLAB code

Let’s profile two versions of the interpolate function!
DEMO

(Profile)
2. Architect/review/optimize MATLAB code

- Alternative 1

```matlab
function odata = interpSimple(idata,r)
    l = 4;
    cutoff = .5;
    b = computeFilter(l,r,cutoff);
    odata = useFilter(b, idata, l, r);
end
```

- Alternative 2

```matlab
function odata = interpSeparated(idata,r)
    l = 4;
    cutoff = .5;
    persistent b
    if(isempty(b))
        b = computeFilter(l,r,cutoff);
    end
    odata = useFilter(b, idata, l, r);
end
```

14 differences found. Use the toolstrip buttons to navigate to them.
2. Architect/review/optimize MATLAB code

- **Alternative 1**

<table>
<thead>
<tr>
<th>Time</th>
<th>Calls</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.48</td>
<td>7650</td>
<td>% Cross-correlate pair channels to get&lt;br&gt;( x_c = \text{corr}(\text{in}(i,:), \text{in}(:,2)); )</td>
</tr>
<tr>
<td>1.04</td>
<td>7650</td>
<td>% Interpolate to increase spatial resolution&lt;br&gt;( x_{\text{Int}}(:, i, j) = \text{interpSimple}(x_c, \text{InterpFactor}); )</td>
</tr>
<tr>
<td>0.05</td>
<td>7650</td>
<td>[-,Index] = max(xcInt);</td>
</tr>
<tr>
<td>0.92</td>
<td>7650</td>
<td>delayVector(n) = Index + InterpFactor</td>
</tr>
<tr>
<td>0.01</td>
<td>153</td>
<td>&lt; 0.01 delay = median(delayVector);</td>
</tr>
</tbody>
</table>

- **Alternative 2**

<table>
<thead>
<tr>
<th>Time</th>
<th>Calls</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.42</td>
<td>7650</td>
<td>% Cross-correlate pair channels to get&lt;br&gt;( x_c = \text{corr}(\text{ul}(i,:), \text{ul}(i,:)); )</td>
</tr>
<tr>
<td>1.54</td>
<td>7650</td>
<td>% Interpolate to increase spatial resolution&lt;br&gt;( x_{\text{Int}} = \text{interpSeparated}(x_c, \text{InterpFactor}); )</td>
</tr>
<tr>
<td>0.04</td>
<td>7650</td>
<td>[-,Index] = max(xcInt);</td>
</tr>
<tr>
<td>0.03</td>
<td>7650</td>
<td>delayVector(n) = Index + InterpFactor -1</td>
</tr>
<tr>
<td>0.01</td>
<td>153</td>
<td>&lt; 0.01 delay = median(delayVector);</td>
</tr>
</tbody>
</table>

Interpolation: 5x faster  
Algorithm: > 2x faster
2. Architect/review/optimize MATLAB code

- Lesson learned from profiling?
  - Separate **initialization and setup** from **recurring execution**
Can we make it execute faster?
1. Experiment with algorithm in MATLAB
2. Customize the MATLAB code
3. Generate C/C++ code
4. Verify/validate generated code
5. Optimize generated code
DEMO

(Matlab to C MEX)
3. Generate C/C++ Code

~8x faster

<table>
<thead>
<tr>
<th>Versions of the Algorithm</th>
<th>Elapsed Time (sec)</th>
<th>Acceleration Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Traditional use of MATLAB functions</td>
<td>6.5367</td>
<td>1.0000</td>
</tr>
<tr>
<td>2. Separating setup and execution</td>
<td>2.1904</td>
<td>2.8842</td>
</tr>
<tr>
<td>3. MATLAB Code - MEX version - MATLAB functions</td>
<td>0.7807</td>
<td>8.3730</td>
</tr>
</tbody>
</table>
Are there other implementation options?
System Objects
2. Architect/review/optimize MATLAB code

init/setup

process
2. Architect/review/optimize MATLAB code

<table>
<thead>
<tr>
<th>Versions of the Algorithm</th>
<th>Elapsed Time (sec)</th>
<th>Acceleration Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Traditional use of MATLAB functions</td>
<td>6.6733</td>
<td>1.0000</td>
</tr>
<tr>
<td>2. Separating setup and execution</td>
<td>2.1771</td>
<td>3.0652</td>
</tr>
<tr>
<td>3. System objects – simulation mode</td>
<td>1.0823</td>
<td>6.1659</td>
</tr>
</tbody>
</table>

~ 2X faster in MATLAB
3. Generate C/C++ Code

~3x faster in C

<table>
<thead>
<tr>
<th>Versions of the Algorithm</th>
<th>Elapsed Time (sec)</th>
<th>Acceleration Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Traditional use of MATLAB functions</td>
<td>6.6733</td>
<td>1.0000</td>
</tr>
<tr>
<td>2. Separating setup and execution</td>
<td>2.1771</td>
<td>3.0652</td>
</tr>
<tr>
<td>3. System objects - simulation mode</td>
<td>1.0823</td>
<td>6.1659</td>
</tr>
<tr>
<td>4. MATLAB Code: MEX version - MATLAB functions</td>
<td>0.7382</td>
<td>9.0404</td>
</tr>
<tr>
<td>5. MATLAB Code: MEX version - System objects</td>
<td>0.2329</td>
<td>28.6588</td>
</tr>
</tbody>
</table>
3. Generate C/C++ Code

**DSP System Toolbox**
Design and simulate signal processing systems

**Communications System Toolbox**
Design and simulate the physical layer of communication systems

**Computer Vision System Toolbox**
Design and simulate computer vision and video processing systems

**Phased Array System Toolbox**
Design and simulate phased array signal processing systems
3. Generate C/C++ Code

### System Objects in DSP System Toolbox

#### DSP System Design Elements

<table>
<thead>
<tr>
<th>Signal</th>
<th>System Objects in Communications System Toolbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coi</td>
<td>System Objects in Computer Vision System Toolbox</td>
</tr>
<tr>
<td></td>
<td>System Objects in Phased Array System Toolbox</td>
</tr>
</tbody>
</table>

#### Phased Arrays

- **Antenna and Microphone Elements**
  - phased.CircularAntennaElement
  - phased.CrossedDipoleAntennaElement
  - phased.CustomAntennaElement
  - phased.CustomMicrophoneElement
  - phased.IsootropicAntennaElement
  - phased.OmniDirectionalMicrophoneElement
  - phased.ShortDipoleAntennaElement

- **Array Geometries and Analyses**
  - phased.VLA
  - phased.URA
  - phased.ConformalArray
  - phased.PartitionedArray
  - phased.ReplicatedSubarray
  - phased.HeterogeneousConformalArray
  - phased.HeterogeneousVLA
  - phased.HeterogeneousURA

---

43
1. Experiment with algorithm in MATLAB
2. Customize the MATLAB code
3. Generate C/C++ code
4. Verify/validate generated code
5. Optimize generated code
4. Verify/Validate Generated Code

Software-in-the-Loop Verification

Verify numerical behavior of generated source code through software-in-the-loop execution.

Test file: testbenchForCodegen.m

Redirect entry-point calls to static library
DEMO

(Software-in-the-Loop)
1. Experiment with algorithm in MATLAB
2. Customize the MATLAB code
3. Generate C/C++ code
4. Verify/validate generated code
5. Optimize generated code
4. Optimize Generated Code

- Operator level (e.g. fixed-point arithmetic)
  
  ```
  c = a + b
  *c = _sadd(a, b)
  ```

- Routine level
  
  ```
  out = matlabFcn(in)
  optimFcn(*in, *out)
  ```

Solution: Code Replacement Libraries (CRL) (requires Embedded Coder)
4. Optimize Generated Code

- Calling legacy code
- Non-algorithmic code (e.g. peripherals, profiling)
- Interacting with larger C/C++ application
- Verifying C/C++ using existing MATLAB testbench

```matlab
if (~isempty(coder.target))
    % Call the following C function
    % void trackUsingPairs(const real_T inpBuffs[
    %     real_T delays[3], real_T xc[3048])
    coder.ceval('trackUsingPairs', ...)
    coder.rref(inpBuffs), ...
    coder.wref(angleOfArrival), ...
    coder.wref(delays), ...
    coder.wref(xc));
else
    coder.ceval('cfun_name', arg1, arg2, ...)
```
What if software is not fast enough?
HW/SW Co-Design

System = Hardware + Software
HW/SW Co-Design

The Zynq Platform - Example
Generating HDL Code
DEMO

(MATLAB $\rightarrow$ HDL Code)
Verifying Handwritten HDL Code

MATLAB → Manual Translation → HDL → Manual Verification

HDL
DEMO
(Verification of Handwritten HDL Code)
Summary | Algorithm Development

1. Experiment with algorithm in MATLAB
2. Customize the MATLAB code
3. Generate C/C++ code
4. Verify/validate generated code
5. Optimize generated code
Summary | Hardware Implementation

System = Hardware + Software

“Integrated Workflow for both SW and HW System Design”
Wrap Up and Q & A

More information...

- Contact us for more information:

  www.mathworks.com