Accelerate the Design and Prototyping of Signal Processing Algorithms

Daniel Aronsson
A system model
System level design
Algorithm design
A toy example
Need to listen for a specific tone (FFT?)
• Probably need some cleaning up (filtering?)
• Need to output a frame of data adequate for the decoder

Use a buffer for the output. Fill it with the input (filtered).
• Take the FFT of a snippet of the buffer to listen for that tone.

One FFT, one filter, one buffer
• Constants (buffer lengths a.s.o)
• Initially, reserve buffer memory, set up filter

Building an algorithm
Data processing is performed on large "batches" of data.

There’s no separation between algorithm and surrounding test environment ("testbench").
A traditional way of working

Loading lots of data into memory is inefficient

Manual indexing is error prone

Batch-processing code is hard to convert to a streaming data algorithm!

Continuous plotting using `drawnow` is slow
Algorithm/testbench separation

- File Access
- Low-Cost Hardware
- High-End Instruments
- ...

- Function
- System Object

- File Access
- Plots
- Scopes
- ...

- Consider architecture
- Consider efficiency
- No plotting
System Objects

• Designed specifically for implementing and simulating dynamic systems with inputs that change over time
• Use internal states to store past behavior, which is used in the next computational step
• Optimized for iterative computations that process large streams of data, such as signal processing and audio systems

Many System objects support:
✓ Fixed-point arithmetic
✓ C code generation
✓ HDL code generation
✓ Executable files or shared libraries generation
Access System Object code templates from the menu

- Call `step(<obj_name>, <input>)` to process data
- No need to call `setup()` – it is called automatically the first time you call `step()`.
- Public properties are exposed in the Simulink block dialog – access `Preview Block Dialog` from the MATLAB menu
The System Object template

```matlab
classdef Untitled < matlab.System

properties
end

% Pre-computed constants
properties(Access = private)
end

methods(Access = protected)
function setupImpl(obj)
    % Perform one-time calculations, such as computing constants
end

function y = stepImpl(obj,u)
    % Implement algorithm. Calculate y as a function of input u and
    % discrete states.
    y = u;
end

end
end
```

- One FFT, one filter, one buffer
- Constants (buffer lengths a.s.o)
- Initially, reserve buffer memory, set up filter
- Use a buffer for the output. Fill it with the input (filtered).
- Take the FFT of a snippet of the buffer to listen for that tone.
MATLAB

```matlab
>> sa = dsp.SpectrumAnalyzer(...
    'SampleRate',10e3);
>> ...
>> step(sa, x);
```
Low-Cost Hardware support
Summary

• Separate algorithm from testbench!
• System objects provide a framework for efficiently working with streaming data
• System objects provide a seamless way of integrating MATLAB components into Simulink
• Low-Cost Hardware support provides cheap real-world data access