

Optimization and Implementation of Embedded Signal Processing Algorithms

Jonas Rutström
Senior Application Engineer

Two important questions in embedded design...

1. What's your algorithm?

Two important questions in embedded design...

2. What's your target?

Targets are very different...

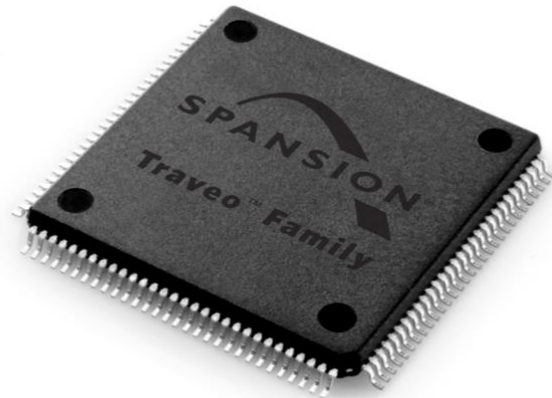


The embedded hardware might be very different...

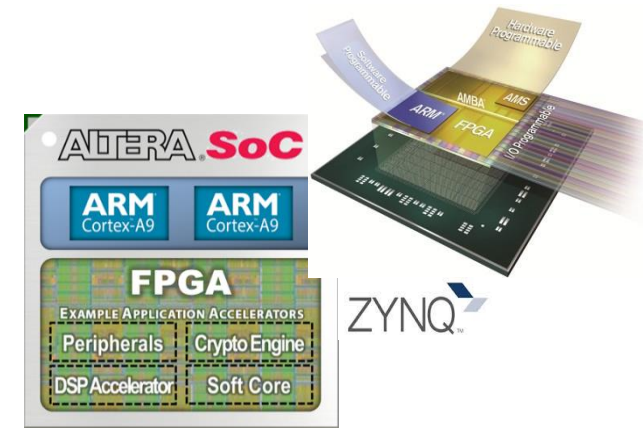
DSP



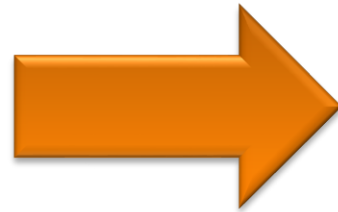
MCU



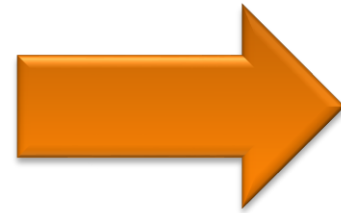
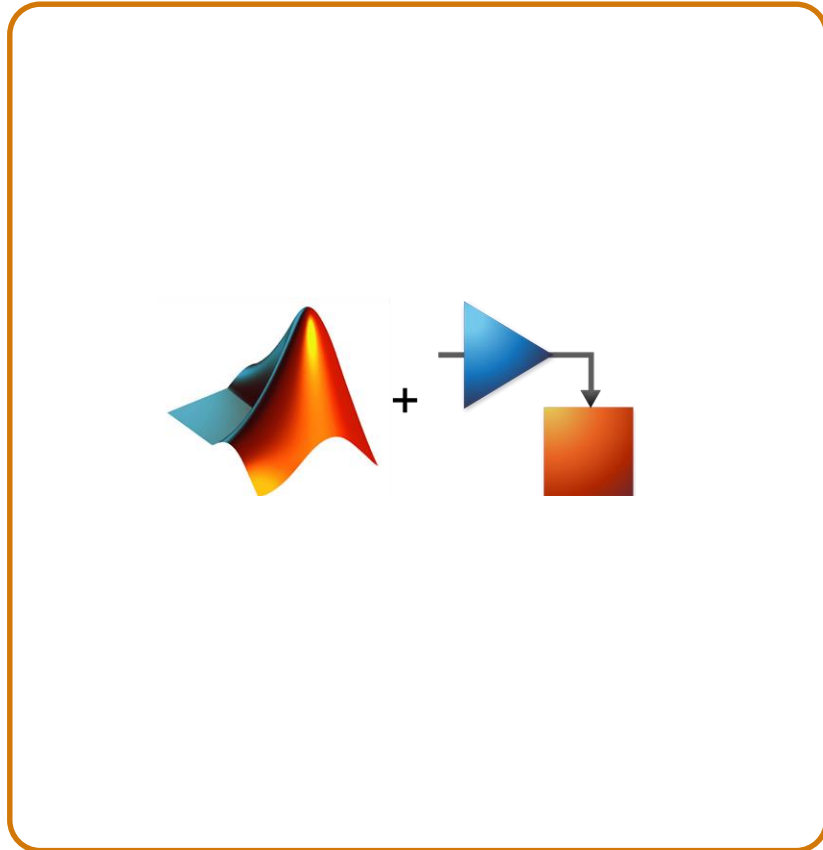
SoC/FPGA



Workflow | Idea to implementation



Workflow | Idea to implementation



Prototyping

Workflow | Idea to implementation



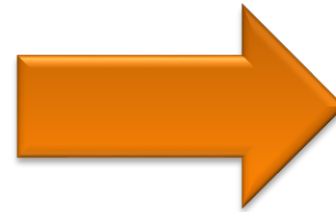
Accelerate the Design and Prototyping of Signal Processing Algorithms
15:00-15:30

In this session we show how you can use MATLAB® for designing and prototyping an algorithm that operates on streaming data. This way of working makes simulations fast and memory efficient. We also show how to test algorithms against data coming from low-cost hardware such as a smartphone or Raspberry Pi™. In the session Optimization and Implementation of Embedded Signal Processing Algorithms, we integrate the algorithm in a larger system that is targeted for a specific hardware platform.

Daniel Aronsson, MathWorks

You will see how to:

- Work with streaming data
- Accelerate simulations by keeping the memory footprint low
- Test algorithms against low-cost hardware



Prototyping

Workflow | Idea to implementation

Accelerate the Design and Prototyping of Signal Processing Algorithms
15:00-15:30

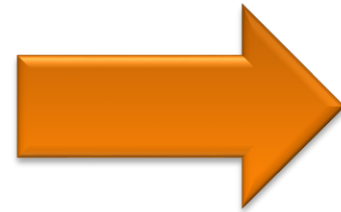
In this session we show how you can use MATLAB® for designing and prototyping an algorithm that operates on streaming data. This way of working makes simulations fast and memory efficient. We also show how to test algorithms against data coming from low-cost hardware such as a smartphone or Raspberry Pi™. In the session Optimization and Implementation of Embedded Signal Processing Algorithms, we integrate the algorithm in a larger system that is targeted for a specific hardware platform.

Daniel Aronsson, MathWorks

You will see how to:

- Work with streaming data
- Accelerate simulations by keeping the memory footprint low
- Test algorithms against low-cost hardware

Prototyping



ALTEIRA SoC
ARM Cortex-A9 ARM Cortex-M3
FPGA
Peripherals: SPI, I2C, UART, CAN, Ethernet
DSP Accelerator Soft-Cores

ZYNQ

TMS320C5506 DSP
Texas Instruments

Production

Workflow | Idea to implementation

Accelerate the Design and Prototyping of Signal Processing Algorithms
15:00-15:30

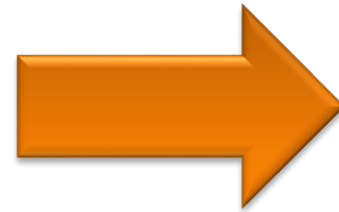
In this session we show how you can use MATLAB® for designing and prototyping an algorithm that operates on streaming data. This way of working makes simulations fast and memory efficient. We also show how to test algorithms against data coming from low-cost hardware such as a smartphone or Raspberry Pi™. In the session Optimization and Implementation of Embedded Signal Processing Algorithms, we integrate the algorithm in a larger system that is targeted for a specific hardware platform.

 Daniel Aronsson, MathWorks

You will see how to:

- Work with streaming data
- Accelerate simulations by keeping the memory footprint low
- Test algorithms against low-cost hardware

Prototyping



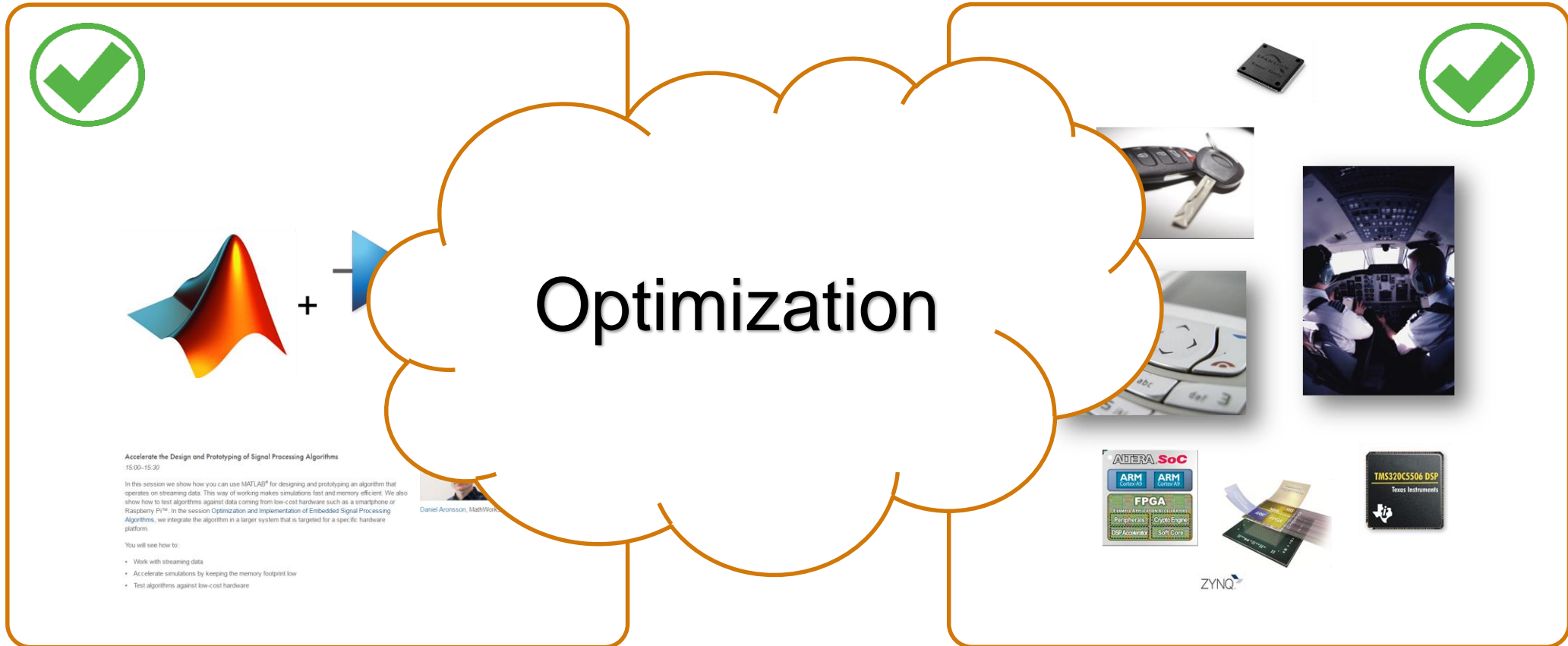
ALTELA SoC
ARM Cortex-A9 ARM Cortex-M3
FPGA
Peripherals: CryptEngine, DSP Accelerator, Soft-Cores

ZYNQ

TMS320C5506 DSP
Texas Instruments

Production

Workflow | Idea to implementation



Accelerate the Design and Prototyping of Signal Processing Algorithms
15:00-15:30

In this session we show how you can use MATLAB® for designing and prototyping an algorithm that operates on streaming data. This way of working makes simulations fast and memory efficient. We also show how to test algorithms against data coming from low-cost hardware such as a smartphone or Raspberry Pi™. In the session Optimization and Implementation of Embedded Signal Processing Algorithms, we integrate the algorithm in a larger system that is targeted for a specific hardware platform.

Daniel Aronsson, MathWorks

You will see how to:

- Work with streaming data
- Accelerate simulations by keeping the memory footprint low
- Test algorithms against low-cost hardware

Prototyping

Production

Let's take a step back...

What do we mean with optimization?

Speed



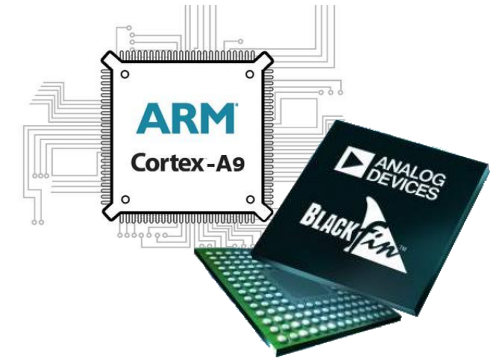
Memory



Readability



Target



Speed



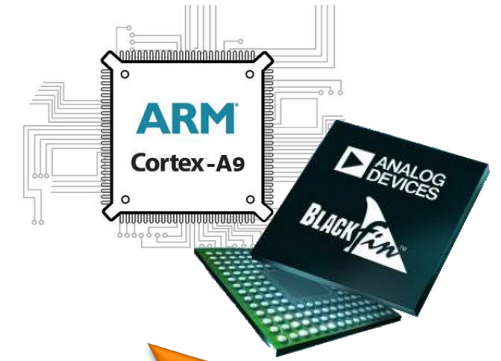
Memory



Readability



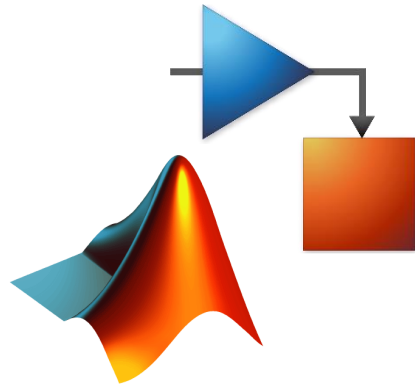
Target



Fixed point

Optimization at different phases in the development...

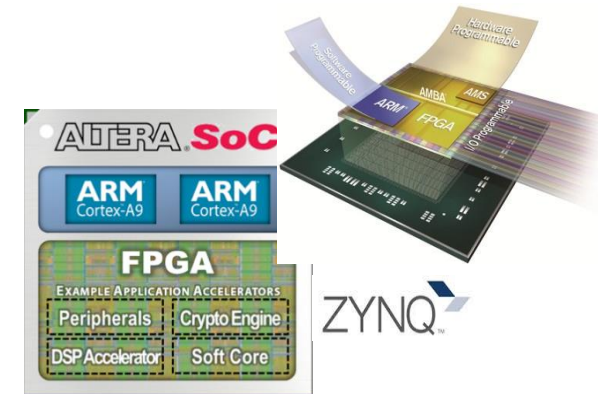
Prototyping



C

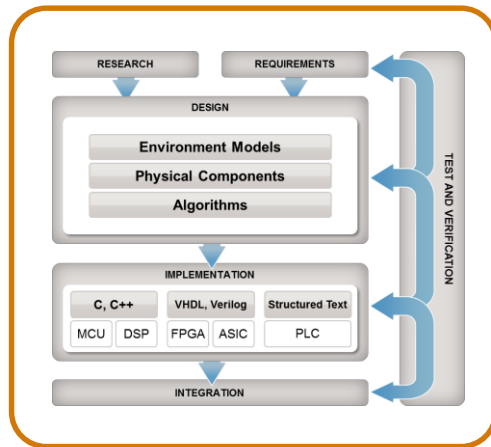


SoC/FPGA



Optimization at different phases in the development...

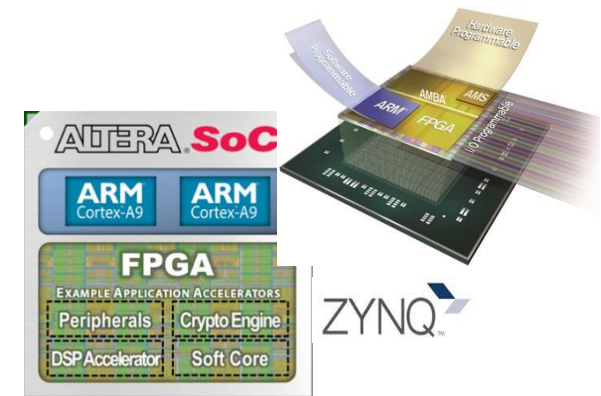
Prototyping



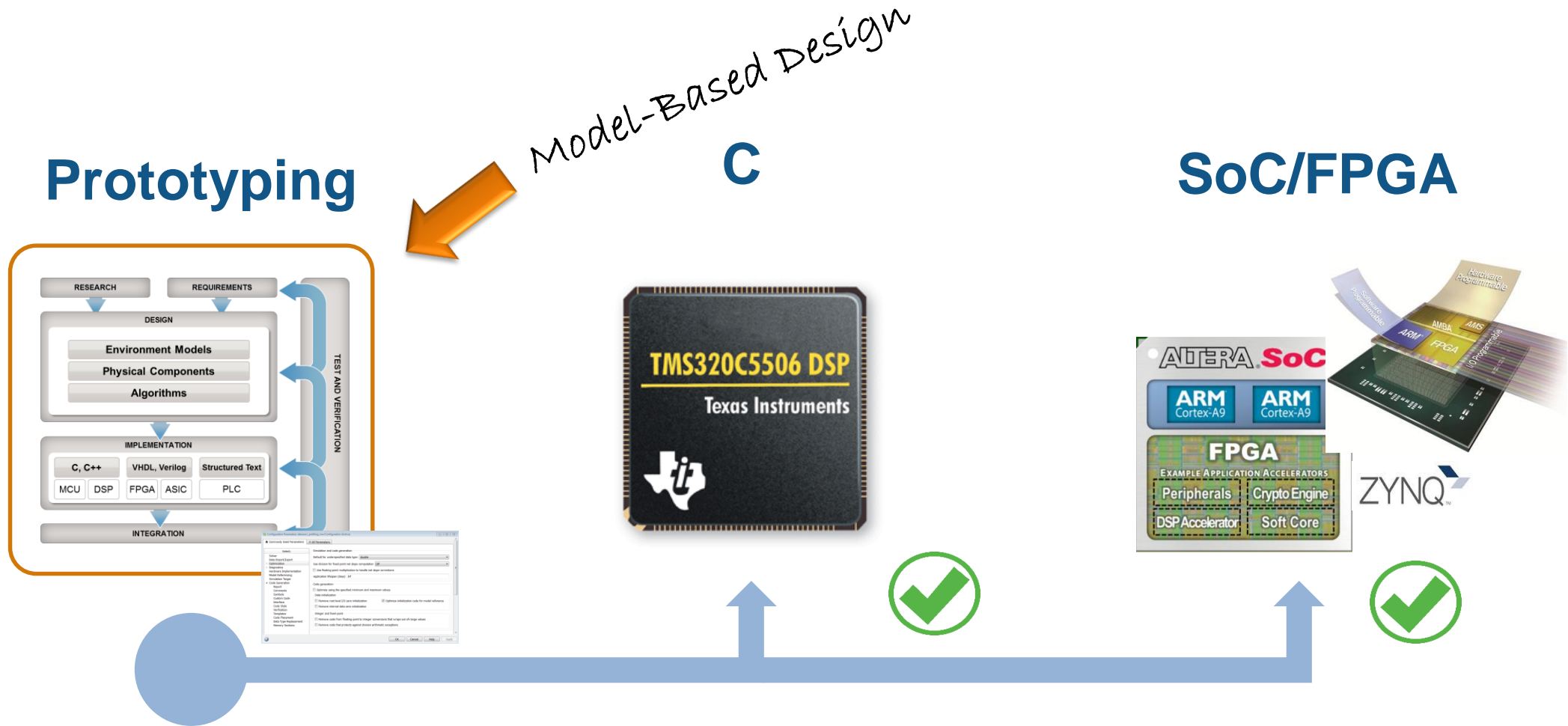
Model-Based Design
C



SoC/FPGA

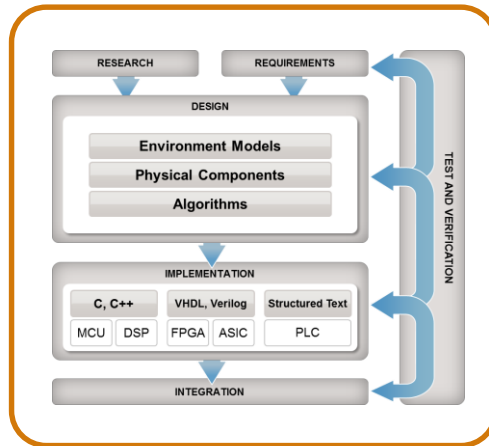


Optimization at different phases in the development...



Optimization at different phases in the development...

Prototyping



How can I analyze performance and optimize code in MATLAB/Simulink?

Profiling!

Profiling in MATLAB

How long did it take?

- tic/toc

```
% start timer
tic

% execute code
out = myFunction(in);

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

Where are the bottlenecks?

- profile

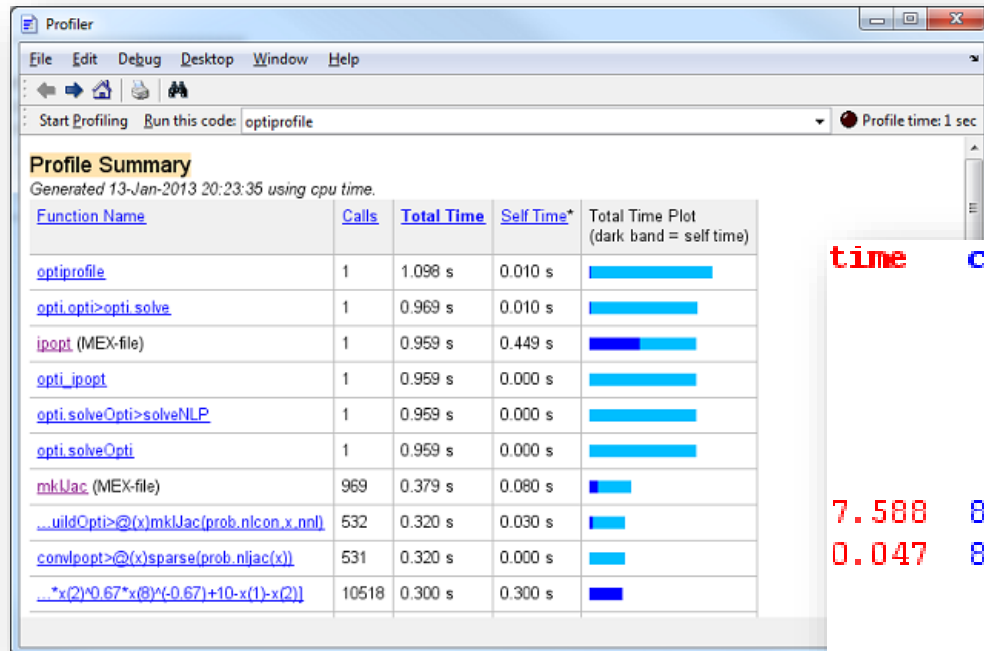
```
% turn on profiler
profile on

% execute code
out = myFunction(in);

% turn off profiler
profile off

% open html report
profile report
```

Profiling in MATLAB



```

time   calls  line
      1     3  function perfTest
      1     4     start = now;
      1     5     for row = 1 : 1000
      1     6         if now-start > 1/(24*60*60)
      1     7             for col = 1 : 100
      1     8                 data = magic(4);
      1     9             end
      1    10         else
      1    11             for col = 1 : 50
      1    12                 data = magic(3);
      1    13             end
      1    14         end
      1    15     end
      1    16 end
  
```

Profiling in Simulink

Web Browser - Simulink Profiler Report

Simulink Profiler Report

[Summary](#) | [Function Details](#) | [Simulink Profiler Help](#) | [Clear Highlighted Blocks](#)

Simulink Profile Report: Summary

Report generated 05-Apr-2016 11:07:29

Total recorded time: 6.49 s
 Number of Block Methods: 32
 Number of Internal Methods: 5
 Number of Model Methods: 11
 Clock precision: 0.00000004 s
 Clock Speed: 2601 MHz

To write this data as modelProfileData in the base workspace [click here](#)

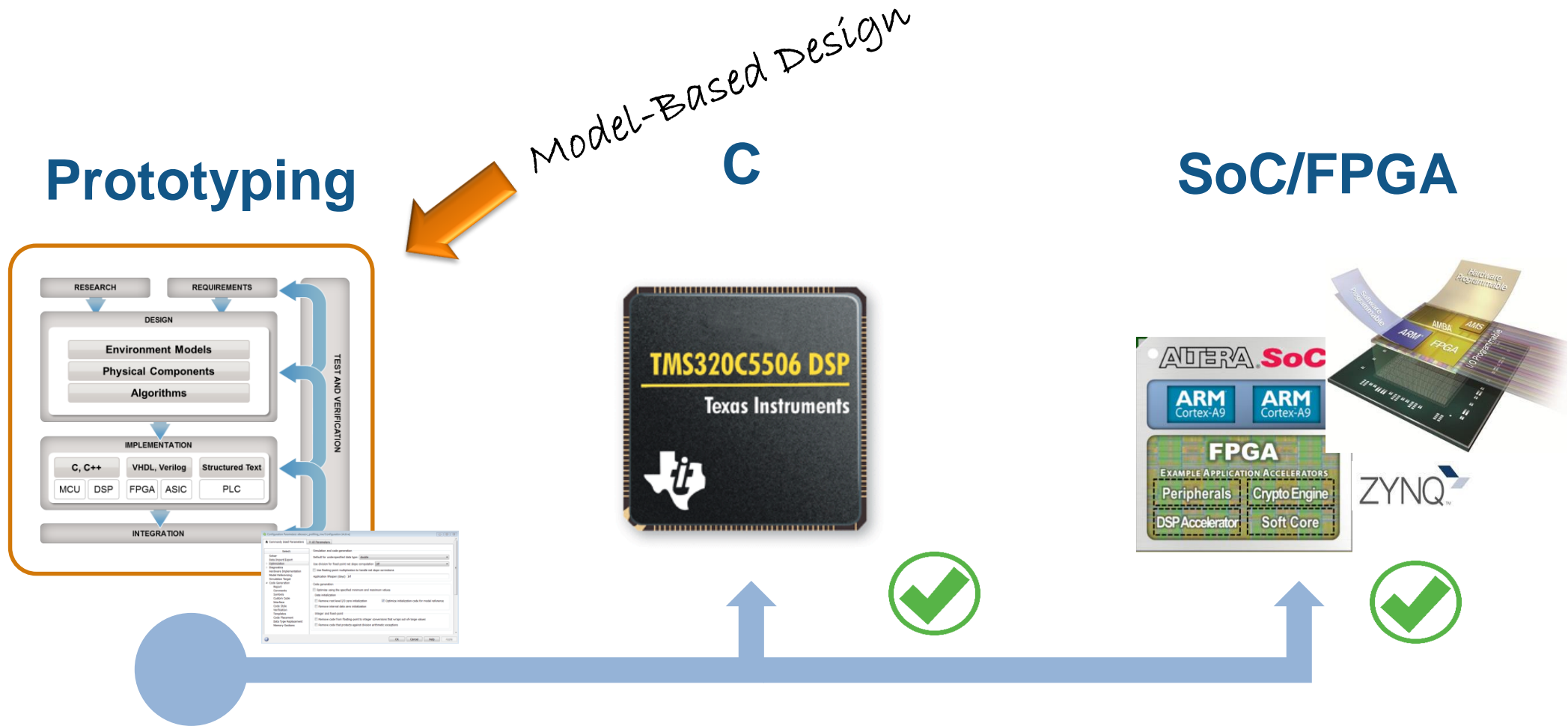
Function List

Name	Time	Time %	Calls	Time/call	Self time	Self time %	Location (must use MATLAB Web Browser to view)
simulate(model1)	6.48964160	100.0%	1	6.4896416000	0.00000000	0.0%	model1
initializationPhase	2.41801550	37.3%	1	2.4180155000	1.17000750	18.0%	model1
simulationPhase	2.29321470	35.3%	1	2.2932147000	0.46800300	7.2%	model1
model1.Start	1.24800800	19.2%	1	1.2480080000	0.00000000	0.0%	model1
compileAndLinkPhase	1.17000750	18.0%	1	1.1700075000	1.17000750	18.0%	model1
model1.Outputs.Major	0.95160610	14.7%	39	0.0244001564	0.00000000	0.0%	model1
model1.Update	0.87360560	13.5%	39	0.0224001436	0.00000000	0.0%	model1
model1/Spectrum Analyzer1 (SpectrumAnalyzer.Start)	0.63960410	9.9%	1	0.6396041000	0.63960410	9.9%	model1/Spectrum Analyzer1
terminationPhase	0.60840390	9.4%	1	0.6084039000	0.51480330	7.9%	model1
model1/Spectrum Analyzer1 (SpectrumAnalyzer.Update)	0.56160360	8.7%	39	0.0144000923	0.56160360	8.7%	model1/Spectrum Analyzer1
model1/Spectrum Analyzer (SpectrumAnalyzer.Start)	0.42120270	6.5%	1	0.4212027000	0.42120270	6.5%	model1/Spectrum Analyzer
model1/Time Scope (Scope.Outputs.Major)	0.42120270	6.5%	39	0.0108000692	0.42120270	6.5%	model1/Time Scope
model1/Detector/MATLAB System (MATLABSystem.Outputs.Major)	0.39000250	6.0%	39	0.0100000641	0.39000250	6.0%	model1/Detector/MATLAB System

Example

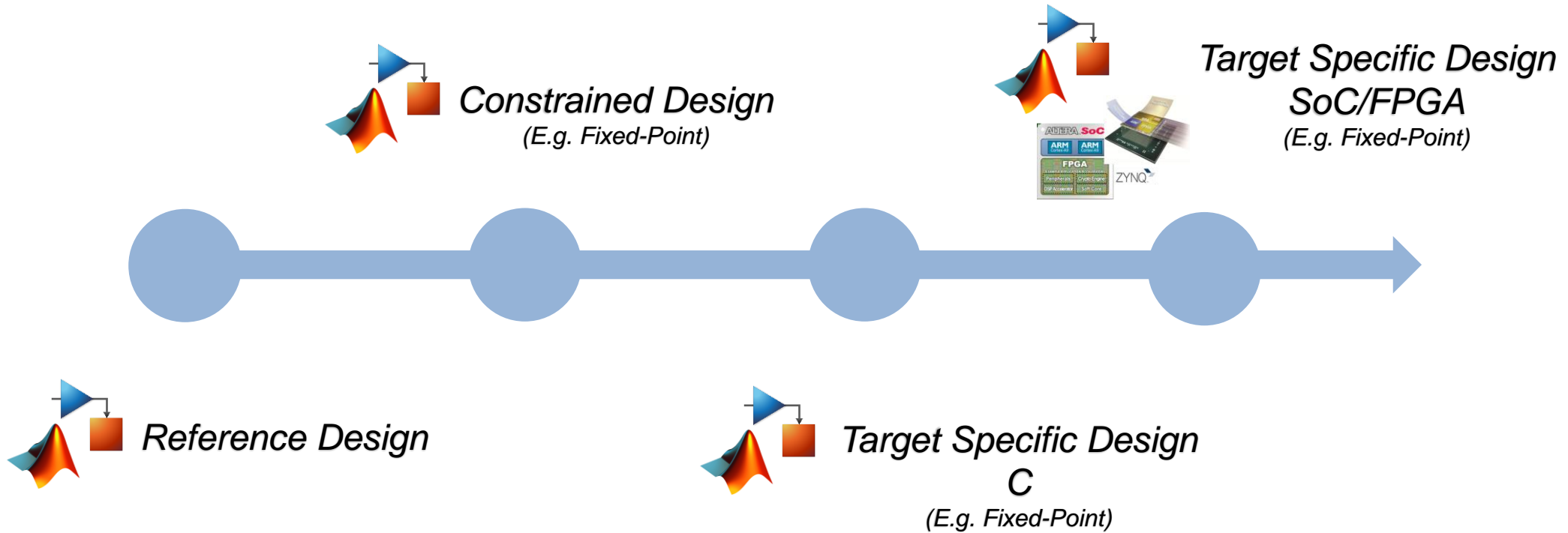
(Profiling when prototyping in MATLAB)

Optimization at different phases in the development...

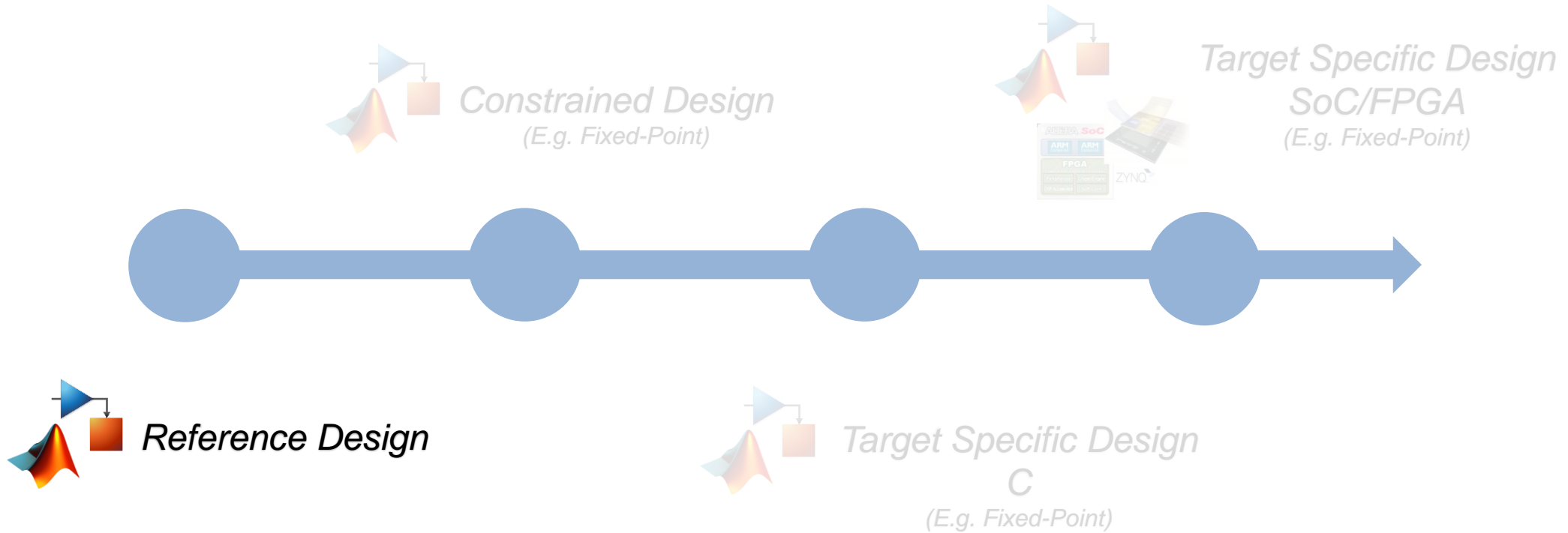


**Now, lets take a closer look of the
workflow in embedded design!**

Example: Workflow for embedded design



Example: Workflow for embedded design



Key Detection Algorithm

Accelerate the Design and Prototyping of Signal Processing Algorithms

15:00–15:30

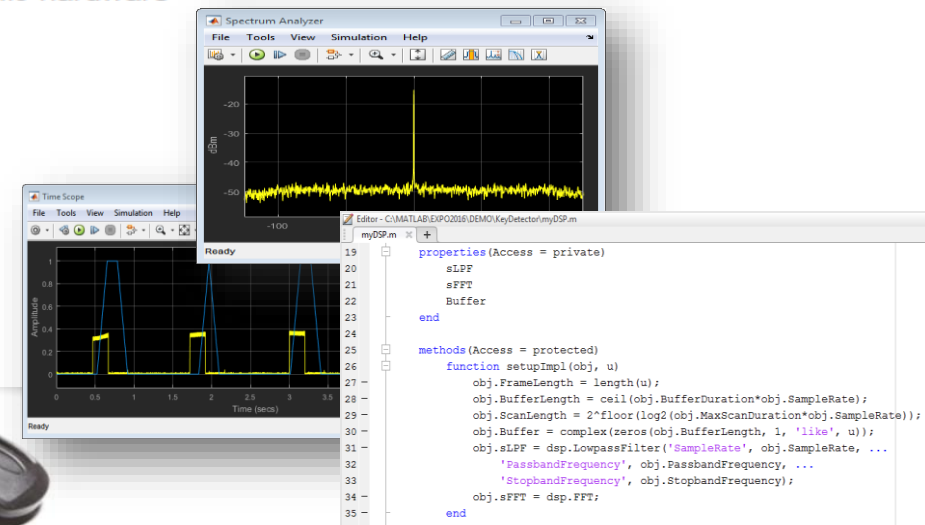
In this session we show how you can use MATLAB® for designing and prototyping an algorithm that operates on streaming data. This way of working makes simulations fast and memory efficient. We also show how to test algorithms against data coming from low-cost hardware such as a smartphone or Raspberry Pi™. In the session [Optimization and Implementation of Embedded Signal Processing Algorithms](#), we integrate the algorithm in a larger system that is targeted for a specific hardware platform.



Daniel Aronsson, MathWorks

You will see how to:

- Work with streaming data
- Accelerate simulations by keeping the memory footprint low
- Test algorithms against low-cost hardware



Component Integration

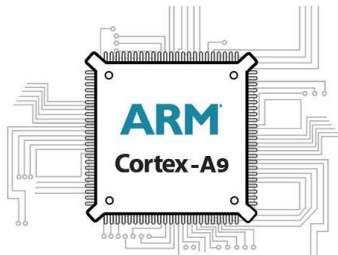
System Design



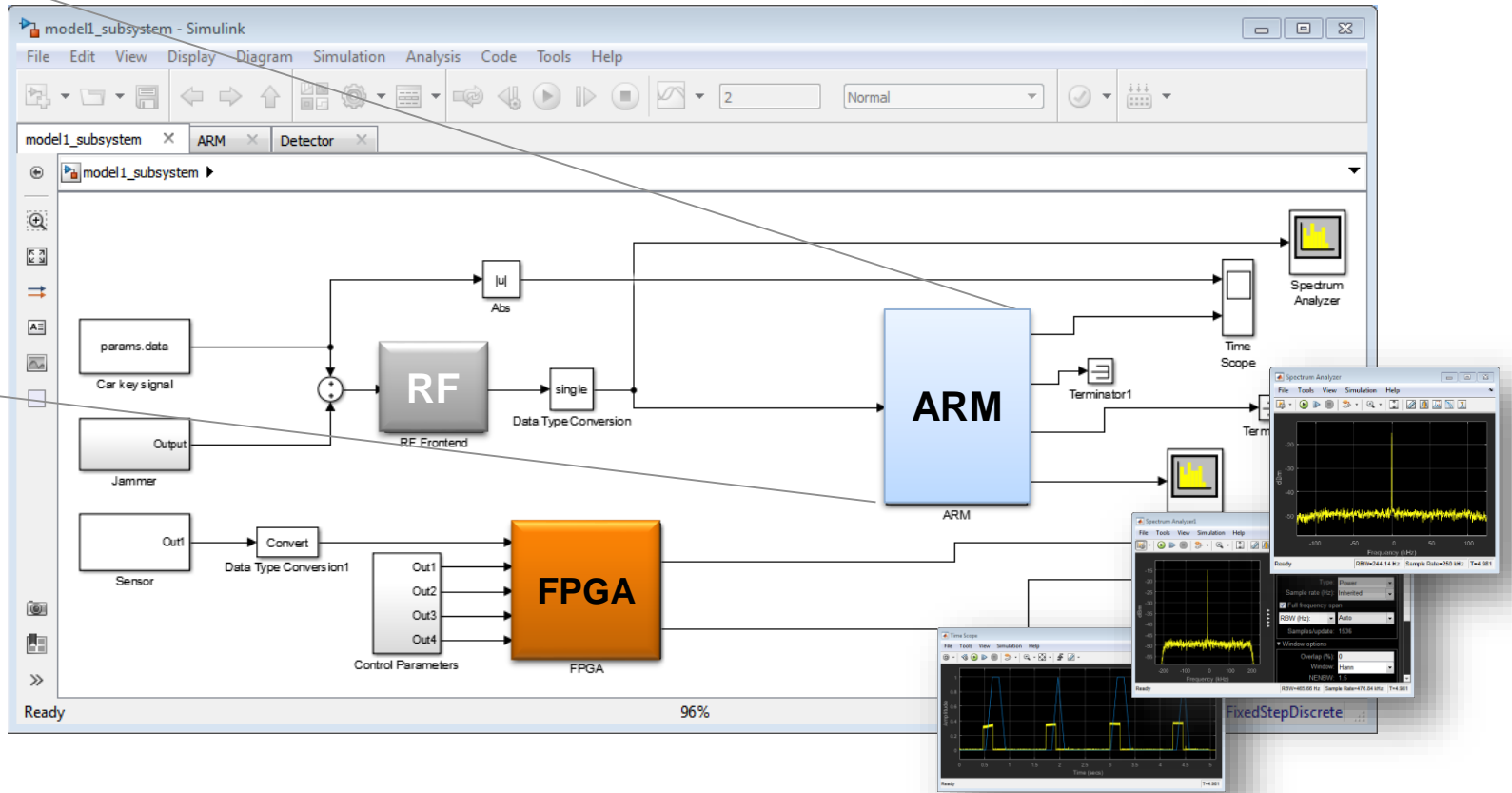
System Object
(Reference Design)

```

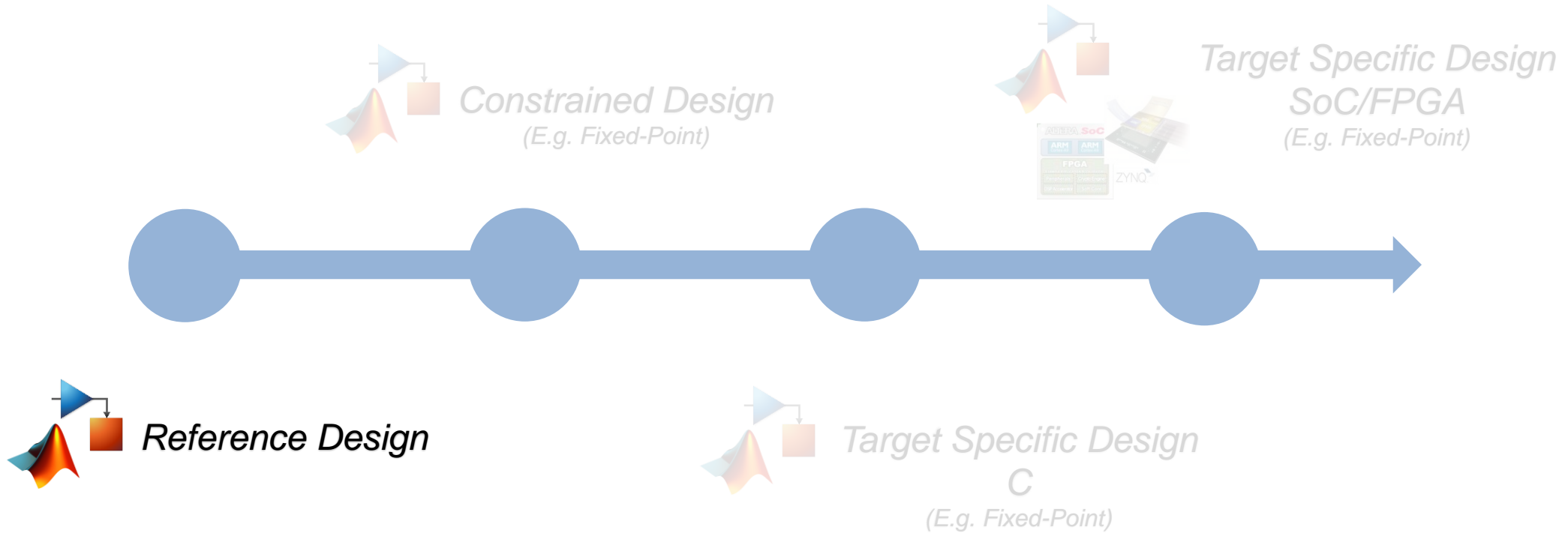
myObj = obj
properties(Access = private)
    sLFF
    sFFT
    Buffer
end
methods(Access = protected)
function setupObj(obj, u)
    obj.FrameLength = length(u);
    obj.BufferLength = ceil(obj.BufferDuration*obj.SampleRate);
    obj.ScanLength = 2^floor(log2(obj.MaxScanDuration*obj.SampleRate));
    obj.Buffer = complex(zeros(obj.BufferLength, 1, 'like', u));
    obj.sLFF = dsp.LowpassFilter('SampleRate', obj.SampleRate, ...
        'PassbandFrequency', obj.PassbandFrequency, ...
        'StopbandFrequency', obj.StopbandFrequency);
    obj.sFFT = dsp.FFT;
end
    
```



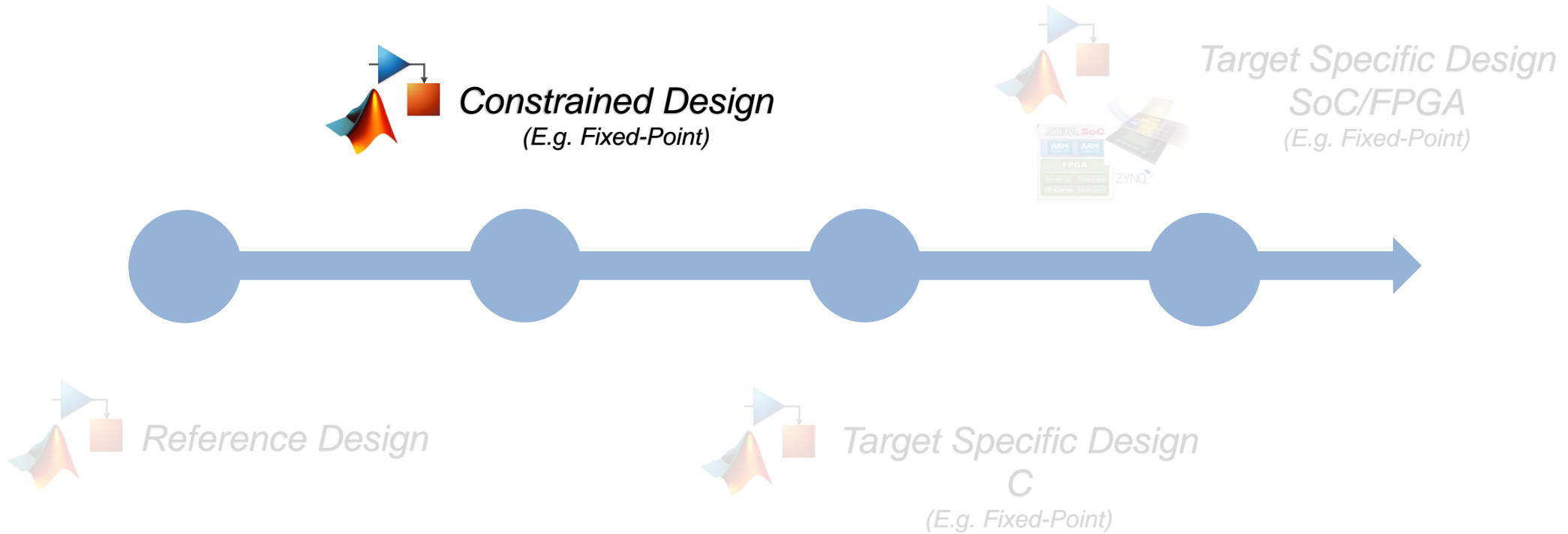
Optimize



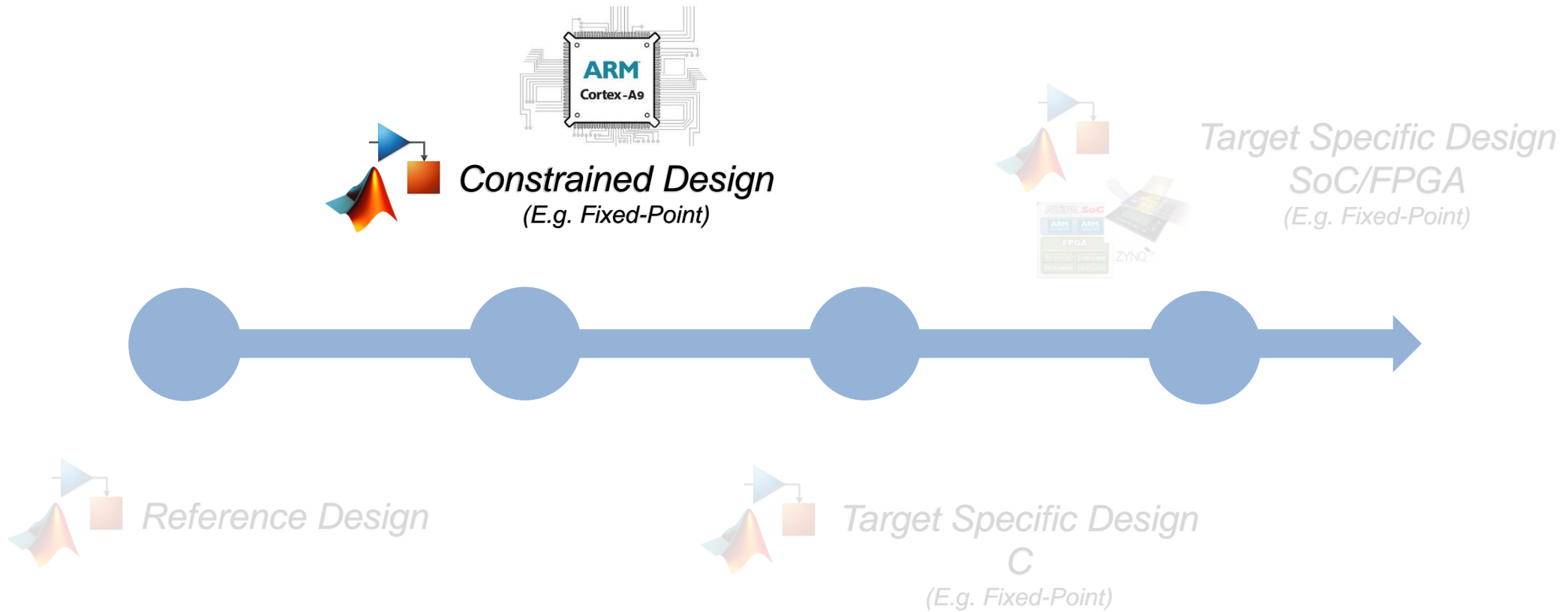
Example: Workflow for embedded design



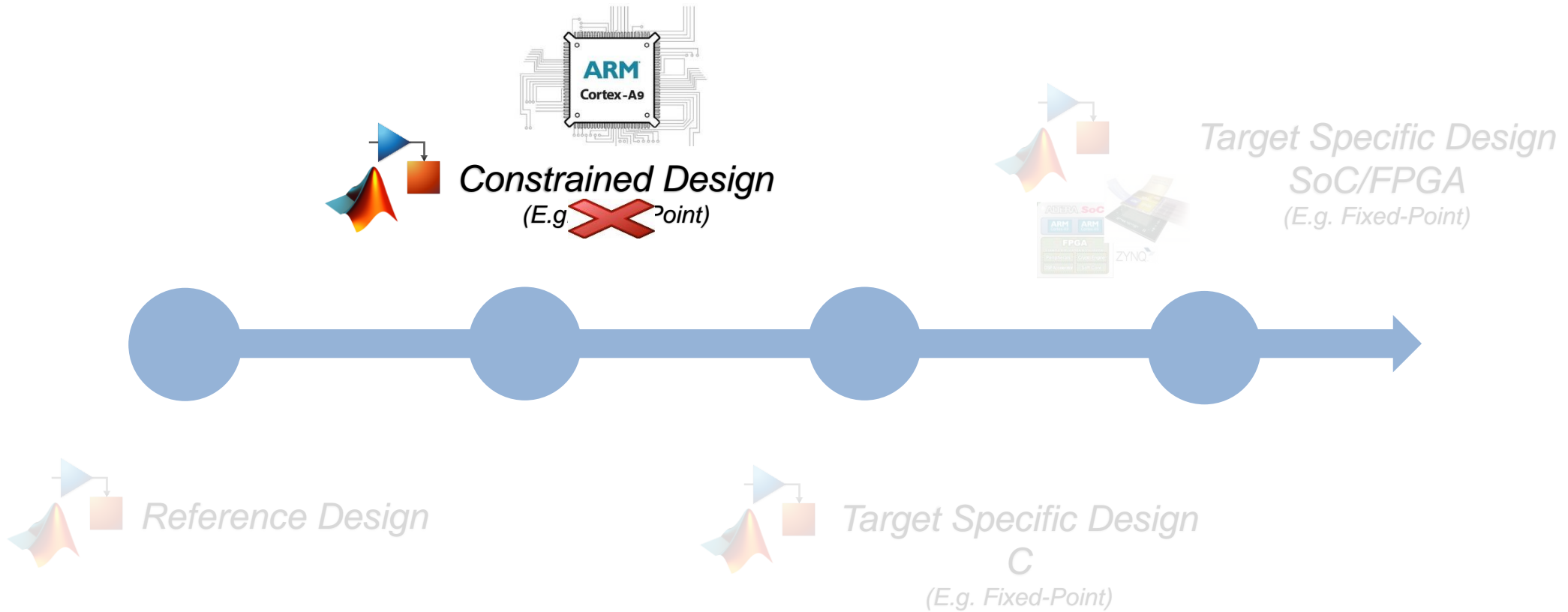
Example: Workflow for embedded design



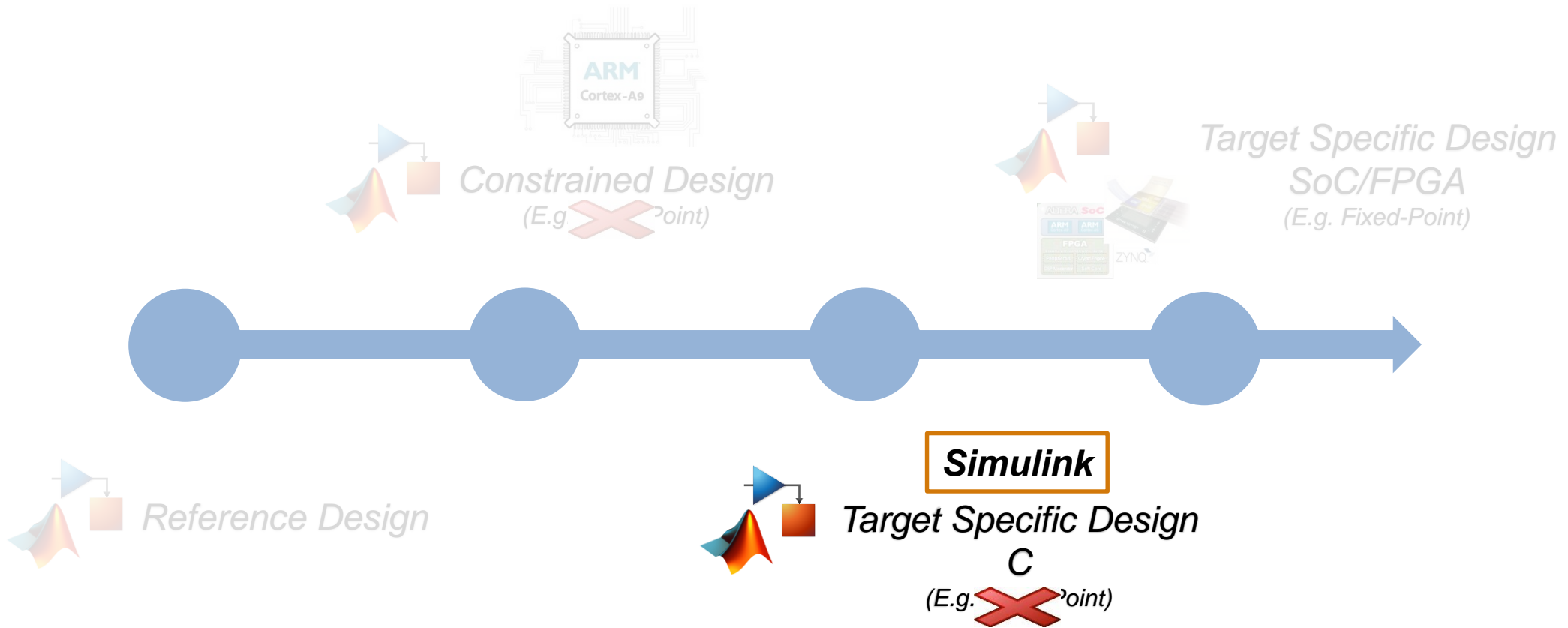
Example: Workflow for embedded design



Example: Workflow for embedded design



Example: Workflow for embedded design



DEMO

*(System Integration and Generating Code for ARM Cortex A9 from Simulink)
(Including verification with PIL)*

Component Integration

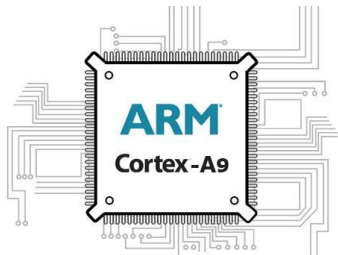
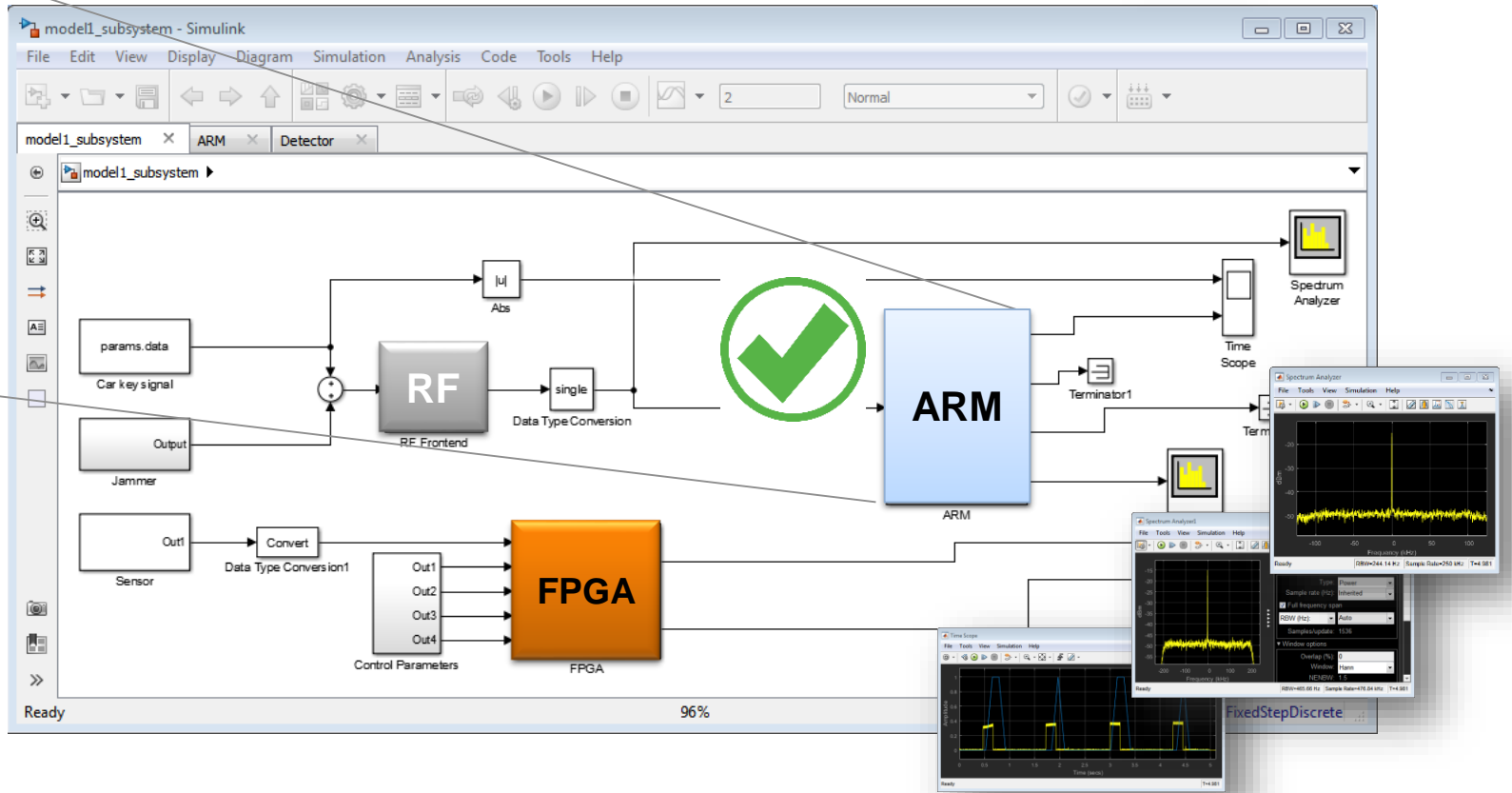
System Design



System Object
(Reference Design)

```

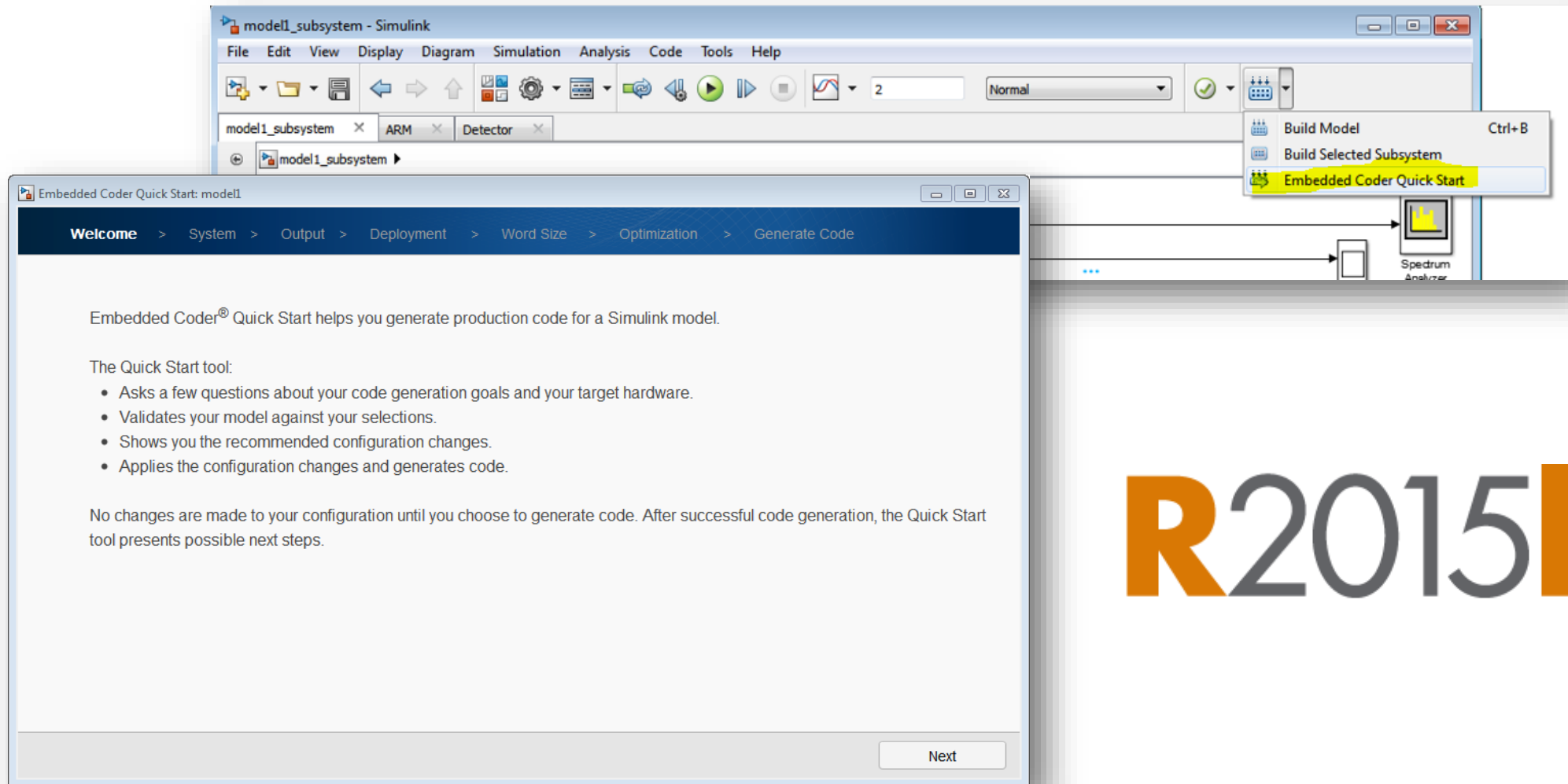
myDSP = classdef
    properties(Access = private)
        sLFF
        sFFT
        Buffer
    end
    methods(Access = protected)
        function setup2mp1(obj, u)
            obj.FrameLength = length(u);
            obj.BufferLength = ceil(obj.BufferDuration*obj.SampleRate);
            obj.SpanLength = 2^floor(log2(obj.MaxSpanDuration*obj.SampleRate));
            obj.Buffer = complex(zeros(obj.BufferLength, 1, 'like', u));
            obj.sLFF = dsp.LowpassFilter('SampleRate', obj.SampleRate, ...
                'PassbandFrequency', obj.PassbandFrequency, ...
                'StopbandFrequency', obj.StopbandFrequency);
            obj.sFFT = dsp.FFT;
        end
    end
end
    
```



Optimize

A few words about Embedded Coder...

Embedded Coder Quick Start



The image shows a Simulink window titled "modell_subsystem - Simulink" with a menu bar (File, Edit, View, Display, Diagram, Simulation, Analysis, Code, Tools, Help) and a toolbar. A context menu is open over the "Spectrum Analyzer" block, listing "Build Model" (Ctrl+B), "Build Selected Subsystem", and "Embedded Coder Quick Start" (highlighted in yellow). In the foreground, the "Embedded Coder Quick Start: model1" window is open, displaying a "Welcome" screen with navigation tabs: Welcome > System > Output > Deployment > Word Size > Optimization > Generate Code. The main text reads: "Embedded Coder® Quick Start helps you generate production code for a Simulink model. The Quick Start tool:

- Asks a few questions about your code generation goals and your target hardware.
- Validates your model against your selections.
- Shows you the recommended configuration changes.
- Applies the configuration changes and generates code.

 No changes are made to your configuration until you choose to generate code. After successful code generation, the Quick Start tool presents possible next steps." A "Next" button is located at the bottom right of the window.

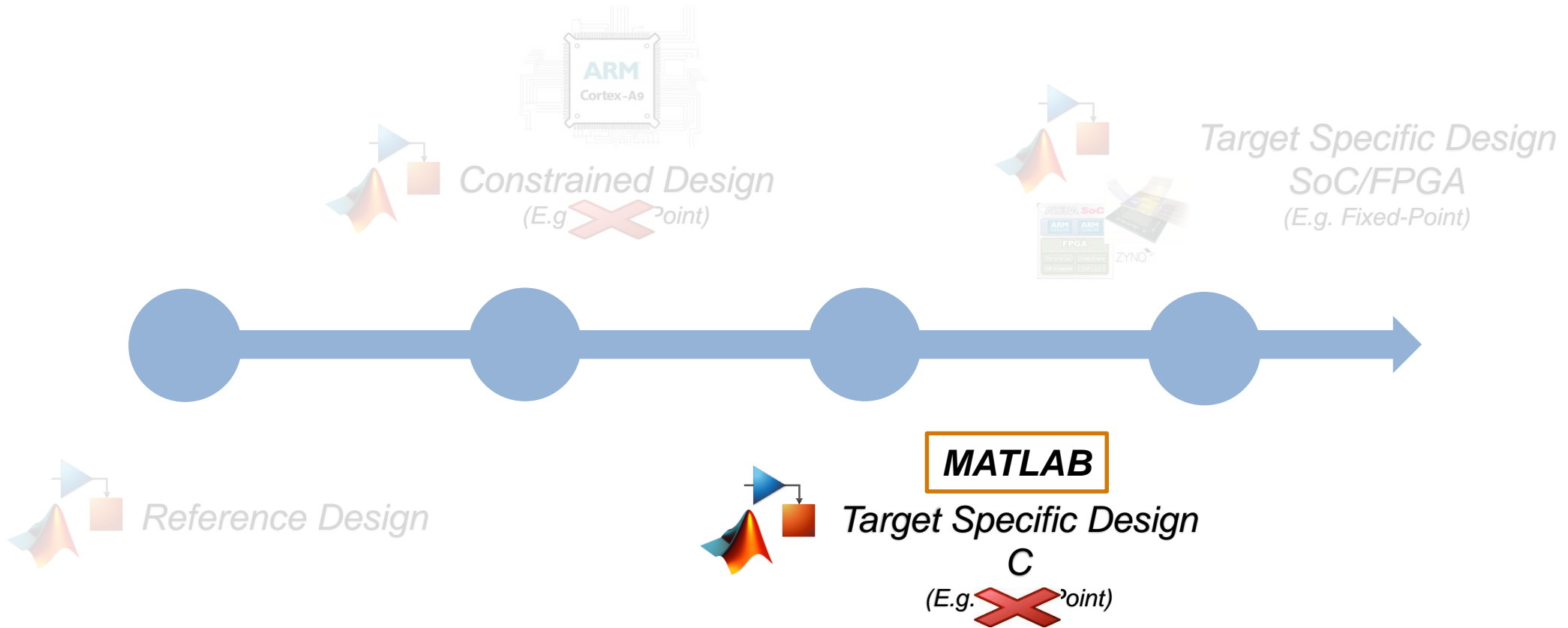
R2015b

DEMO

(Embedded Coder Quick Start)

Another use case...

Example: Workflow for embedded design



Example

(Generating Code ARM Cortex A9 from MATLAB)

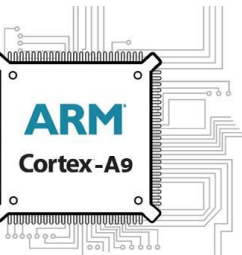
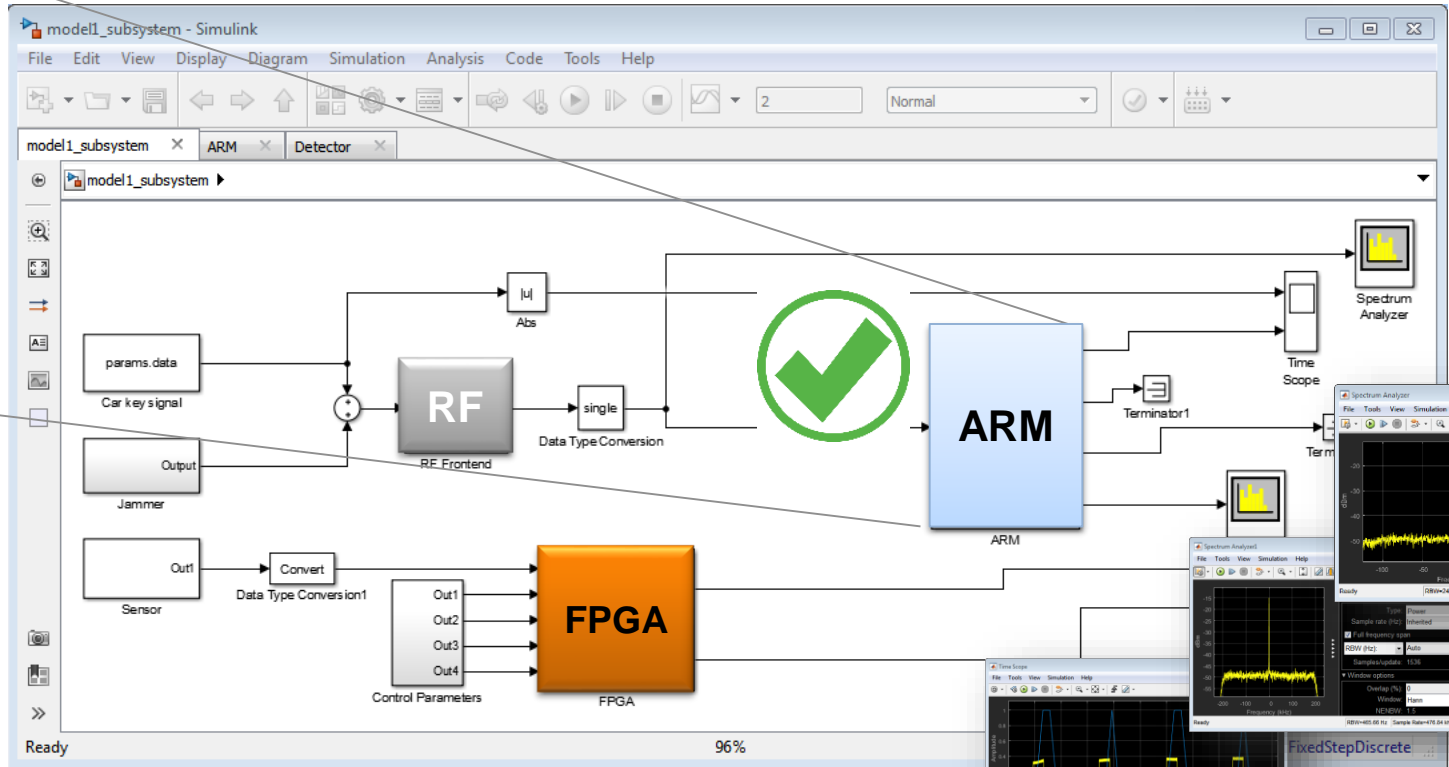
Component Integration

System Design



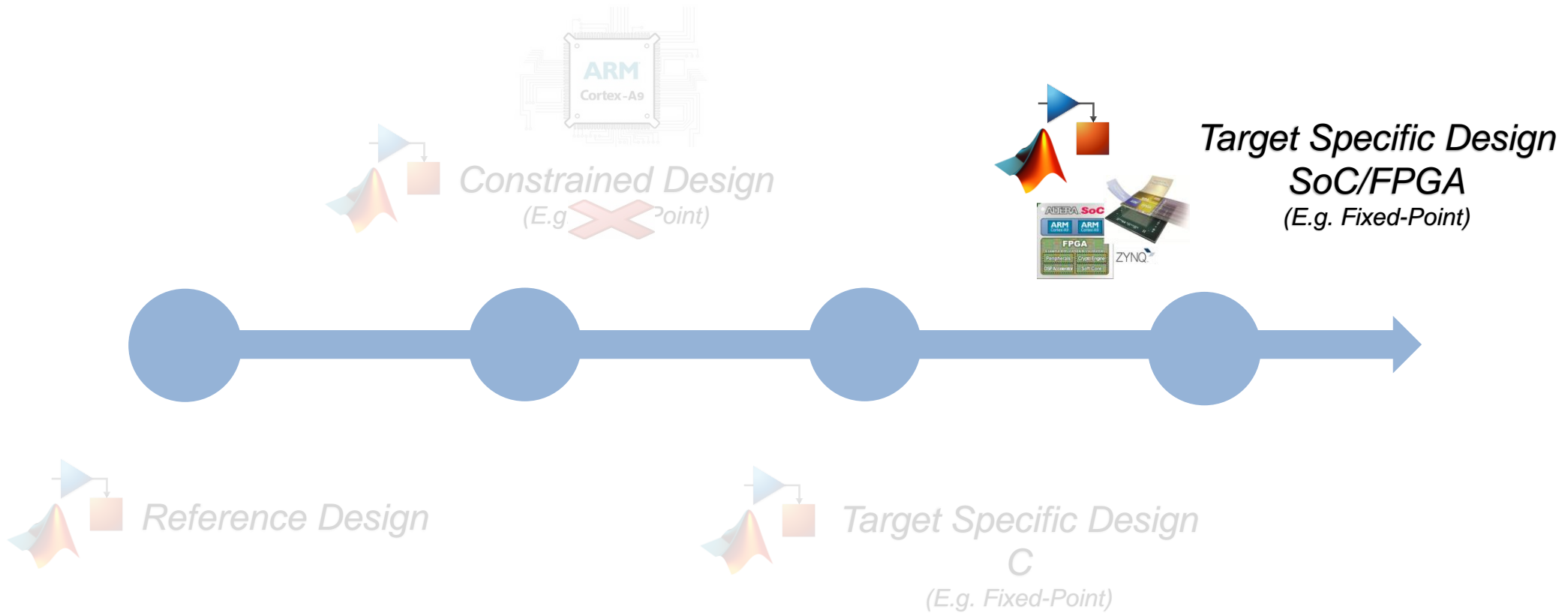
System Object (Reference Design)

```
myDSP = classdef('myDSP')
    properties(Access = private)
        sLFF
        sFFT
        Buffer
    end
    methods(Access = protected)
        function setup2mp1(obj, u)
            obj.FrameLength = length(u);
            obj.BufferLength = ceil(obj.BufferDuration*obj.SampleRate);
            obj.SpanLength = 2^floor(log2(obj.MaxSpanDuration*obj.SampleRate));
            obj.Buffer = complex(zeros(obj.BufferLength, 1, 'like', u));
            obj.sLFF = dsp.LowpassFilter('SampleRate', obj.SampleRate, ...
                'PassbandFrequency', obj.PassbandFrequency, ...
                'StopbandFrequency', obj.StopbandFrequency);
            obj.sFFT = dsp.FFT;
        end
    end
end
```

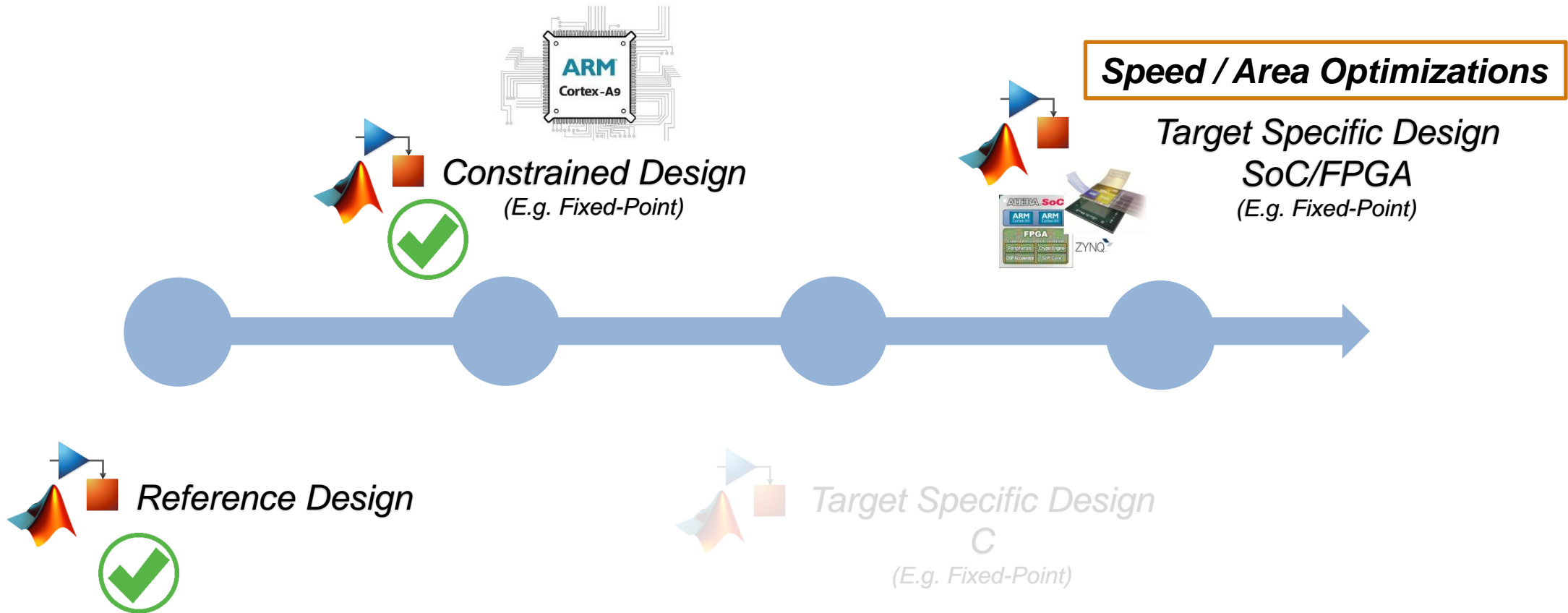


Optimize

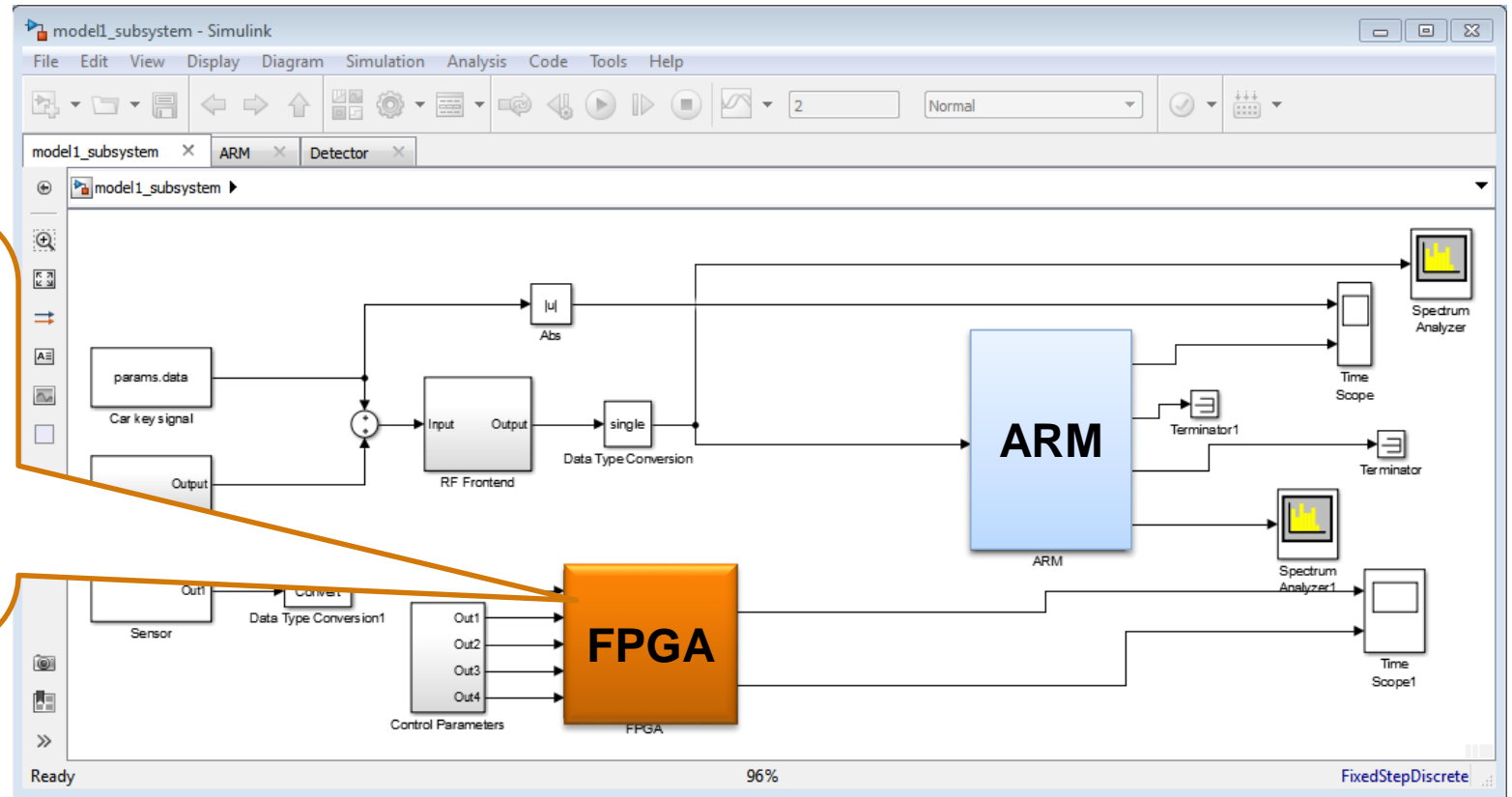
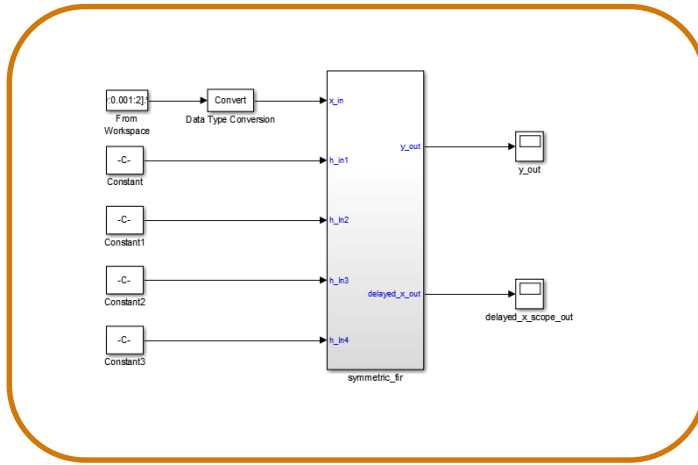
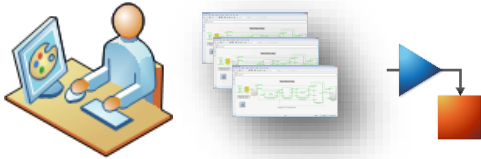
Example: Workflow for embedded design



Example: Workflow for embedded design



Area Optimization for FPGA implementation



Example

(Area Optimization for FPGA Implementation)

Component Integration

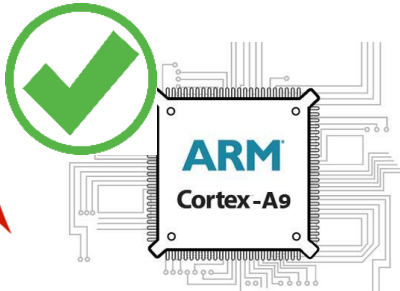
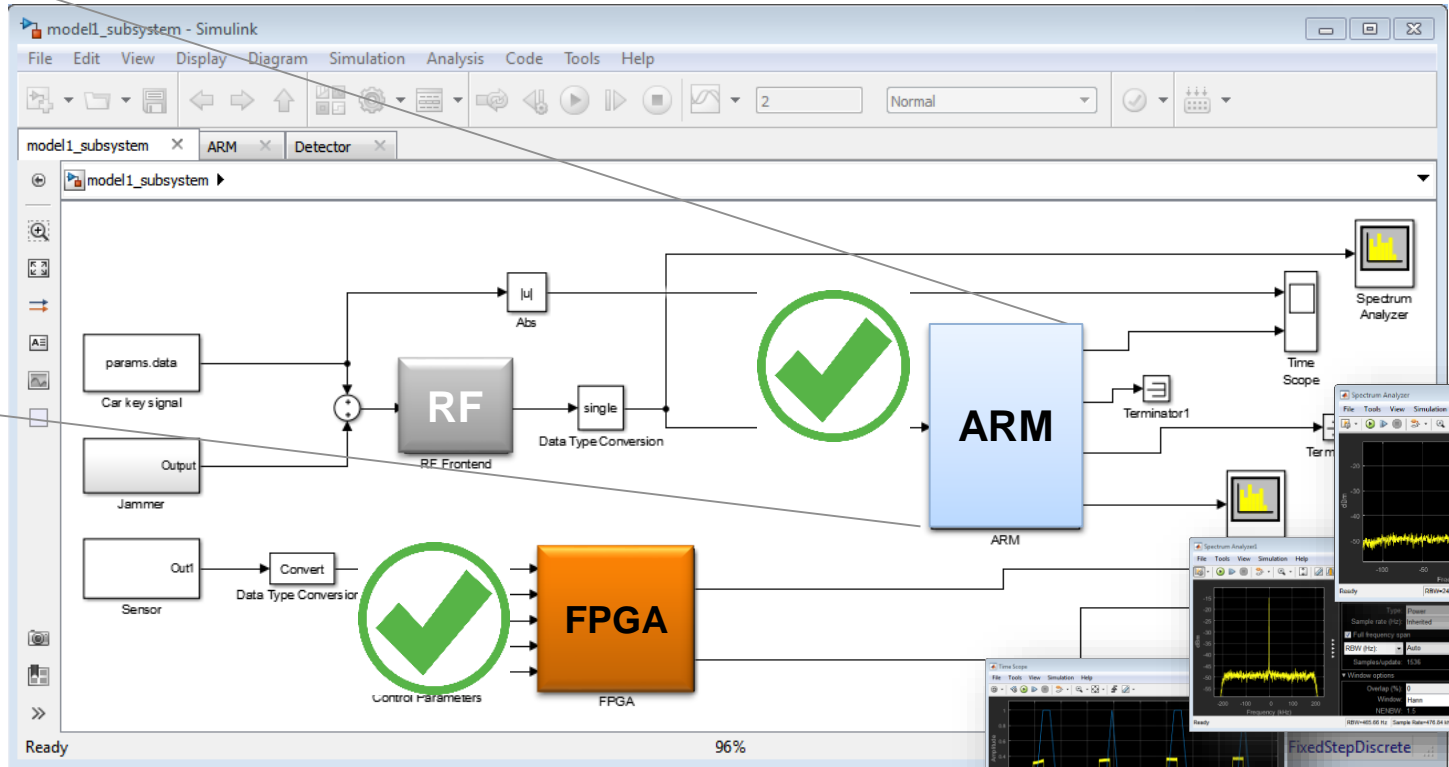
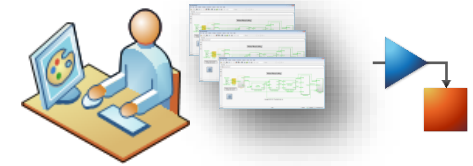
System Design



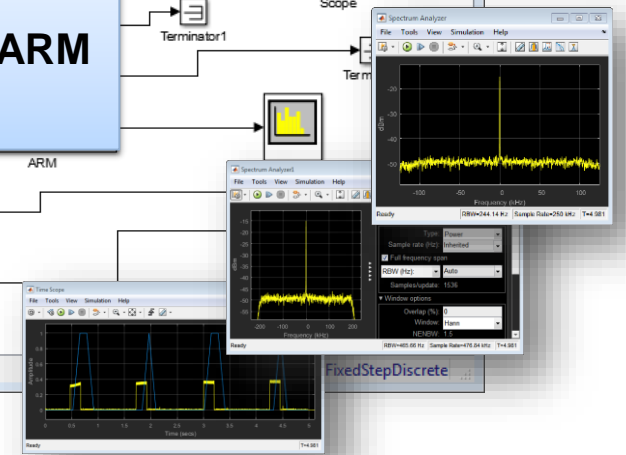
System Object
(Reference Design)

```

myObj = obj
properties(Access = private)
    sLFF
    sFFT
    Buffer
end
methods(Access = protected)
function setupObj(obj, u)
    obj.FrameLength = length(u);
    obj.BufferLength = ceil(obj.BufferDuration*obj.SampleRate);
    obj.ScanLength = 2^floor(log2(obj.MaxScanDuration*obj.SampleRate));
    obj.Buffer = complex(zeros(obj.BufferLength, 1, 'like', u));
    obj.sLFF = dsp.LowpassFilter('SampleRate', obj.SampleRate, ...
        'PassbandFrequency', obj.PassbandFrequency, ...
        'StopbandFrequency', obj.StopbandFrequency);
    obj.sFFT = dsp.FFT;
end
    
```

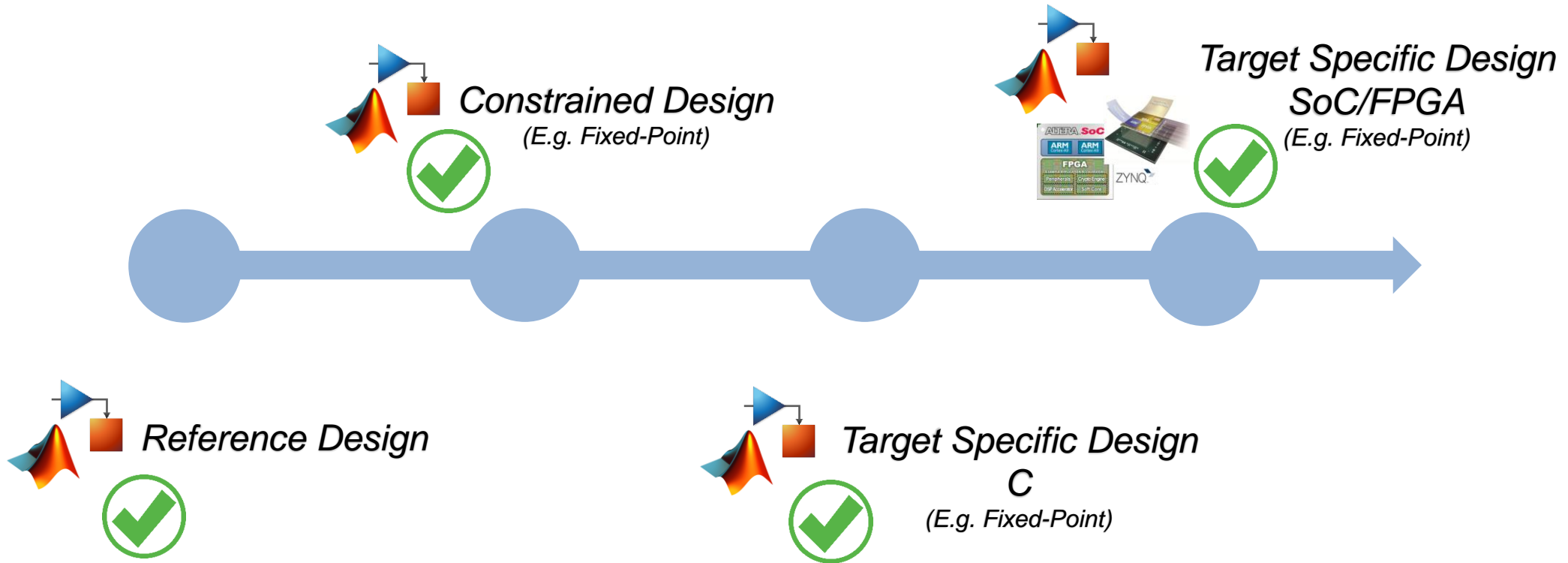


Optimize



Summary

Optimization and Implementation of Embedded Signal Processing Algorithms



Questions?