Masterclass

MATLAB EXPO 2019

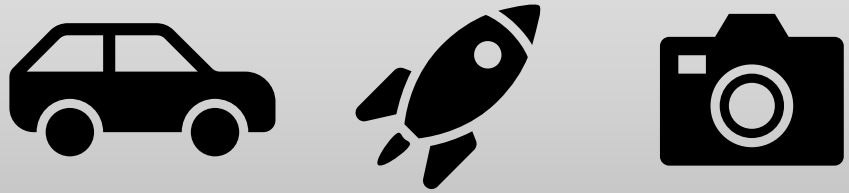
Adopting Model-Based Design for FPGA, ASIC, and SoC Development

Fahd Morchid

Agenda

- Why Model-Based Design for FPGA, ASIC, or SoC?
- Case Study – Pulse Detector
- HW/SW Co-Design
- Customer results

Just an example, the workflow is the same for...



Agenda

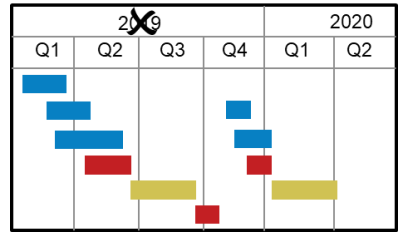
- Why Model-Based Design for FPGA, ASIC, or SoC?
- Case Study – Pulse Detector
- HW/SW Co-Design
- Customer results

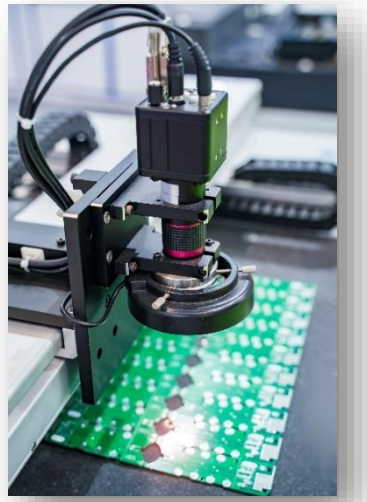
FPGA, ASIC, and SoC Development Projects

67% of ASIC/FPGA projects are **behind schedule**

Over **50% of project time** is spent on **verification**

75% of ASIC projects require a **silicon re-spin**

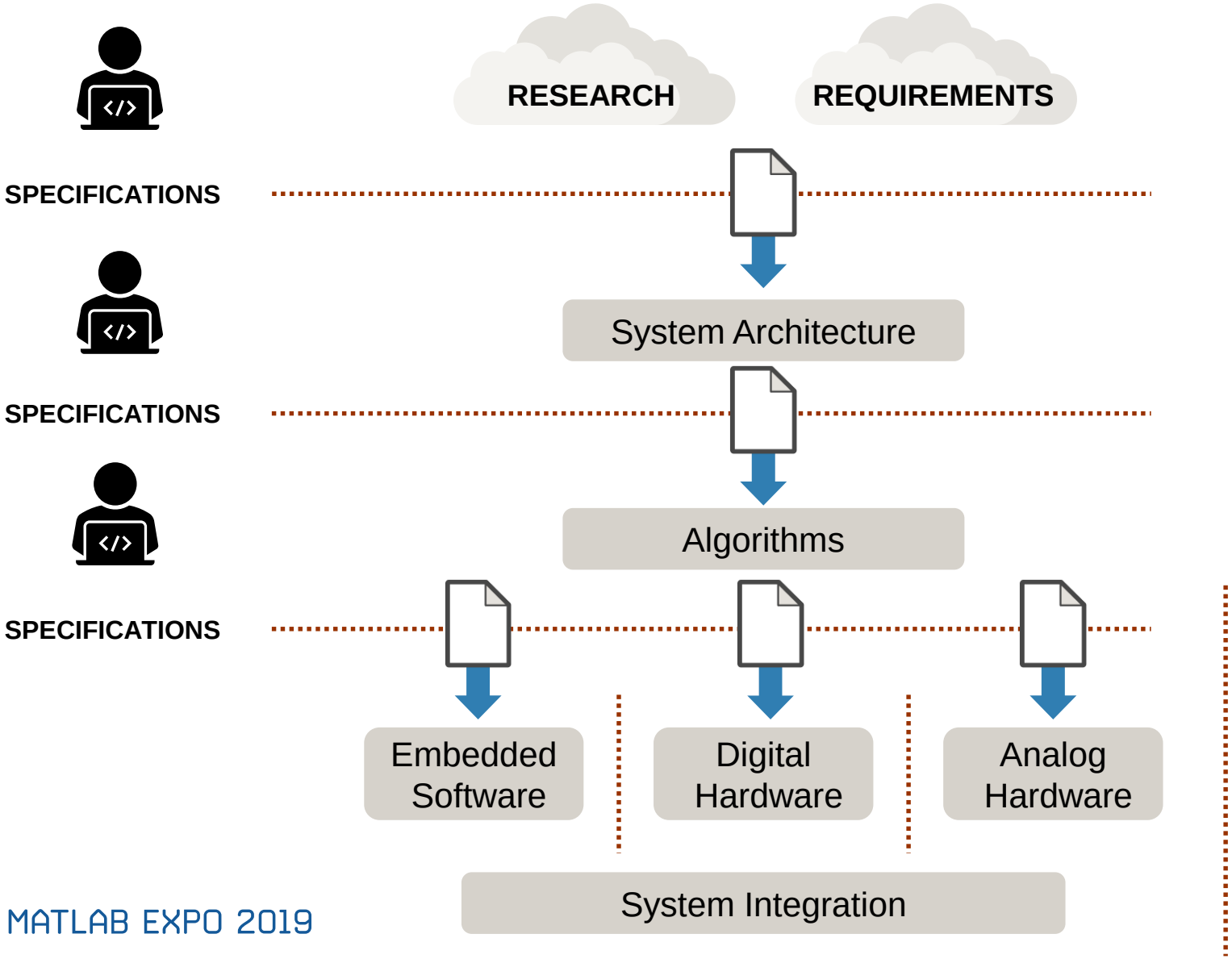




84% of FPGA projects have non-trivial **bugs escape into production**

Statistics from 2018 Mentor Graphics / Wilson Research survey, averaged over FPGA/ASIC

Many Different Skill Sets Need to Collaborate

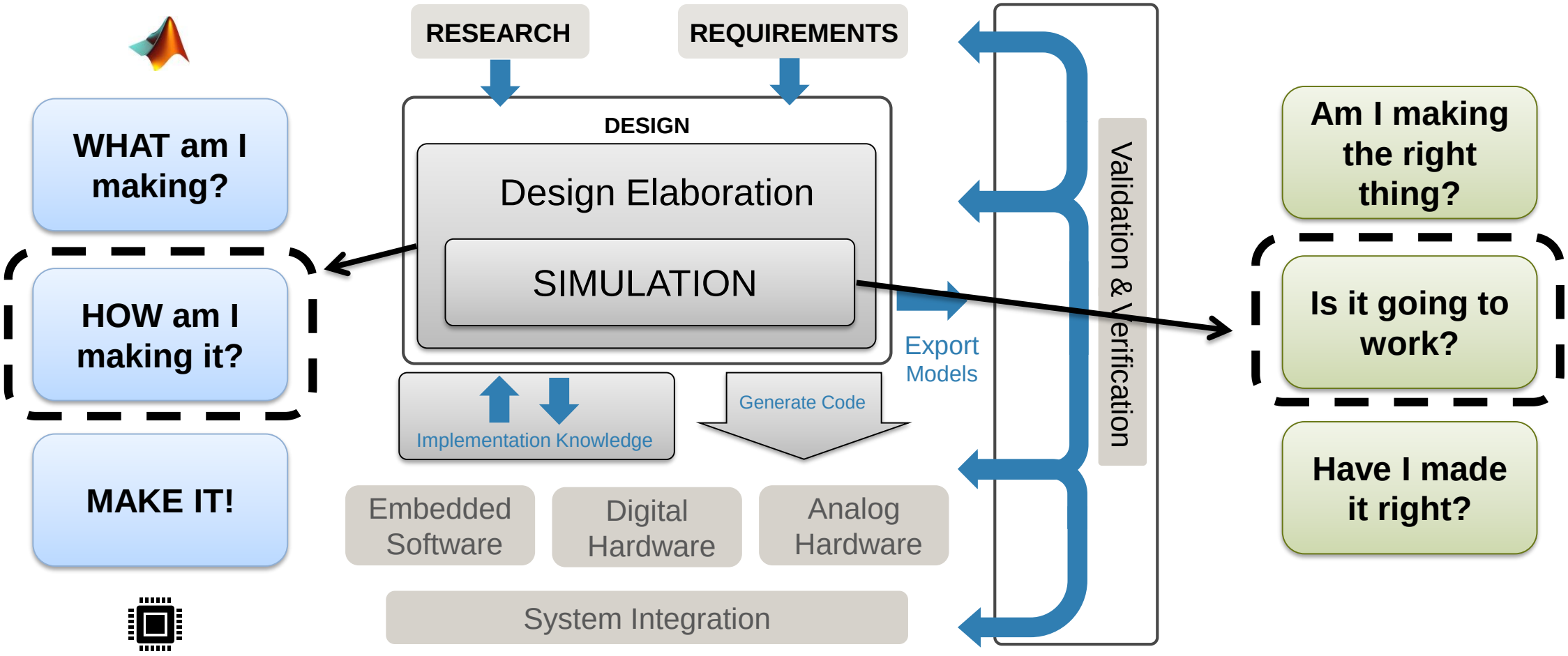


- Poor communication across teams
- Key decisions made in silos
- System-level issues found in late stages
- Hard to adapt to changing requirements

"Rapid innovation under a rapid timeline – that's when this flow falls apart."

Jamie Haas
Allegro Microsystems

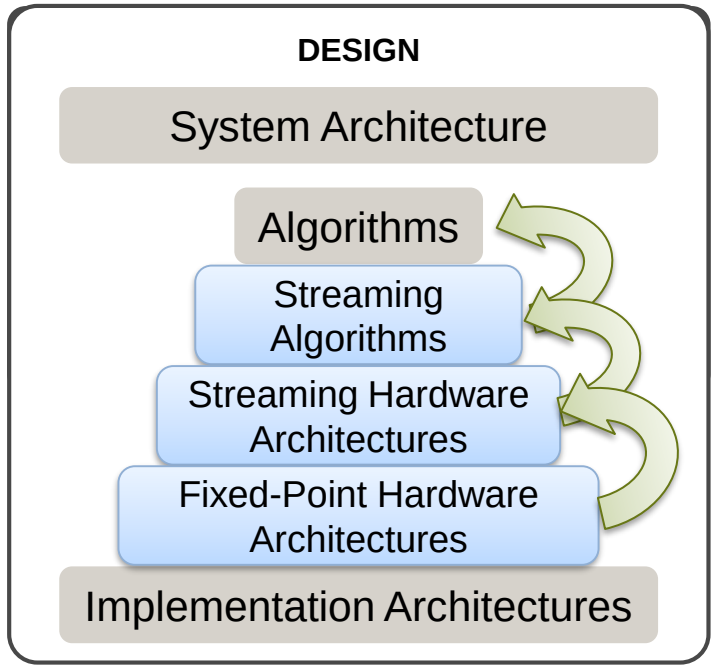
SoC Collaboration with Model-Based Design



General Approach: Use the Strengths of MATLAB and Simulink

MATLAB

- ✓ Large data sets
- ✓ Explore mathematics
- ✓ Control logic
- ✓ Data visualization



Simulink

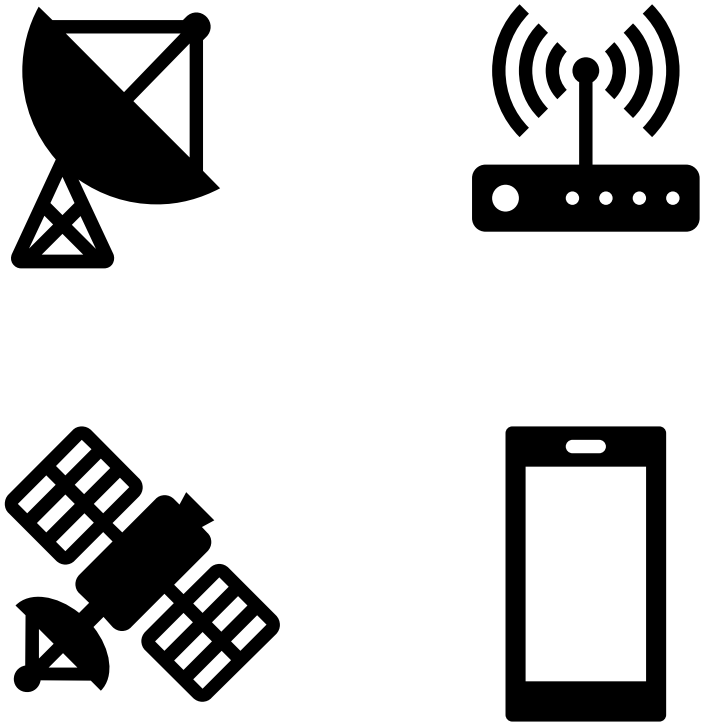
- ✓ Parallel architectures
- ✓ Timing
- ✓ Data type propagation
- ✓ Mixed-signal modeling

Agenda

- Why Model-Based Design for FPGA, ASIC, or SoC?
- Case Study – Pulse Detector
- HW/SW Co-Design
- Customer results

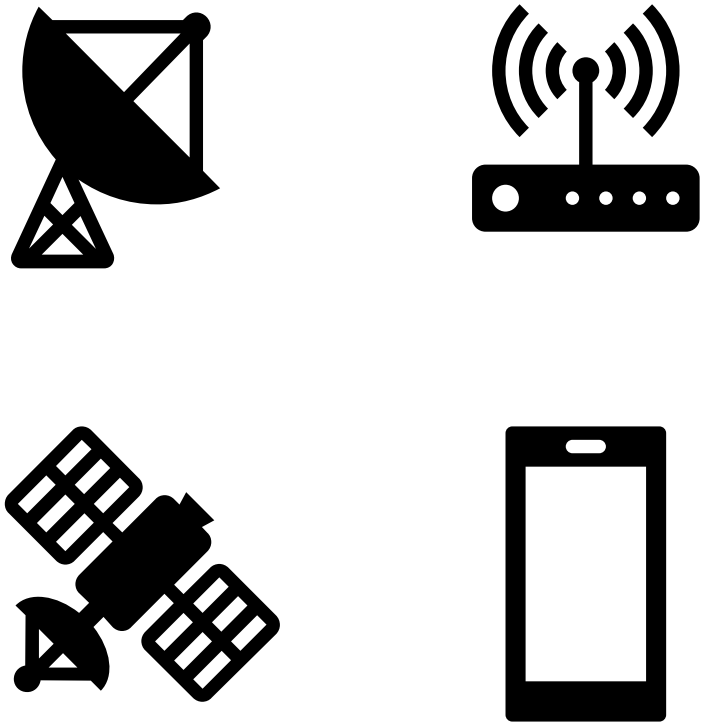
Case Study | *Pulse Detector*

1. Example Overview
2. Reference Pulse Detector
3. Pulse Detector Design
4. Prepare for Hardware Implementation
5. Fixed-point Conversion
6. HDL code generation, synthesis and verification



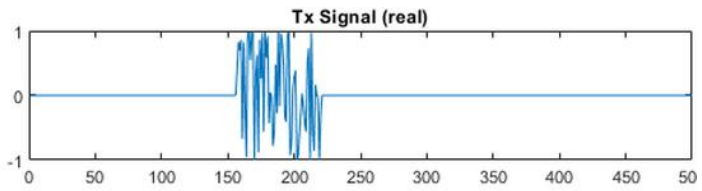
Case Study | *Pulse Detector*

1. Example Overview
2. Reference Pulse Detector
3. Pulse Detector Design
4. Prepare for Hardware Implementation
5. Fixed-point Conversion
6. HDL code generation, synthesis and verification

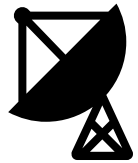


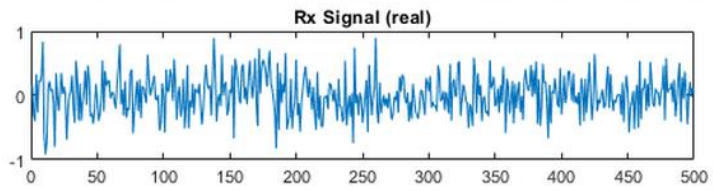
Pulse Detector | *Overview*

Send

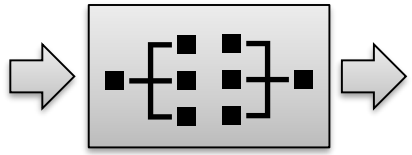


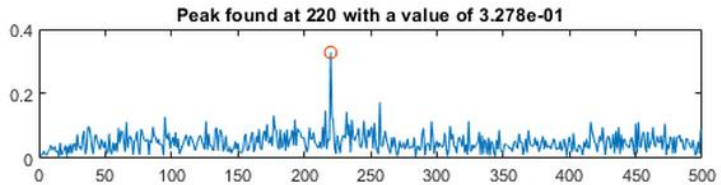
Receive





Detect

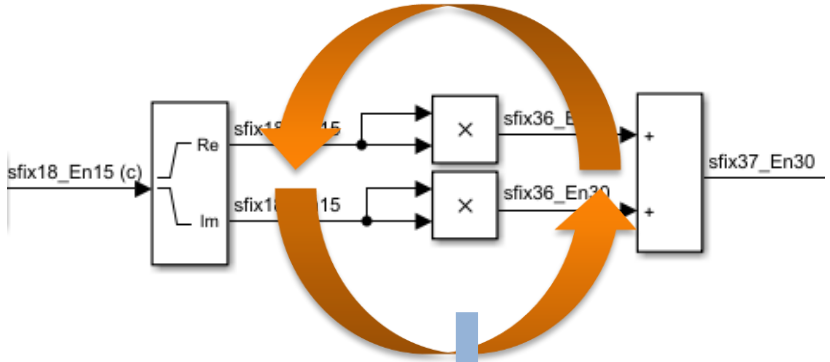




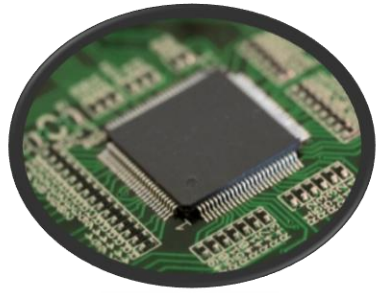
Reference Design (MATLAB)

```
% Create matched filter coefficients
CorrFilter = conj(flip(pulse))/PulseLen;

% Correlate Rx signal against matched filter
FilterOut = filter(CorrFilter,1,RxSignal);

% Find peak magnitude & location
[peak, location] = max(abs(FilterOut));

% Print results
figure(1)
subplot(311); plot(real(TxSignal)); title('Tx Signal (real)');
subplot(312); plot(real(RxSignal)); title('Rx Signal (real)');

t = 1:length(FilterOut);
str = sprintf('Peak found at %d with a value of %.3d',location,peak);
subplot(313); plot(t,abs(FilterOut),location,peak,'o'); title(str);
```

Detector Design (Simulink)

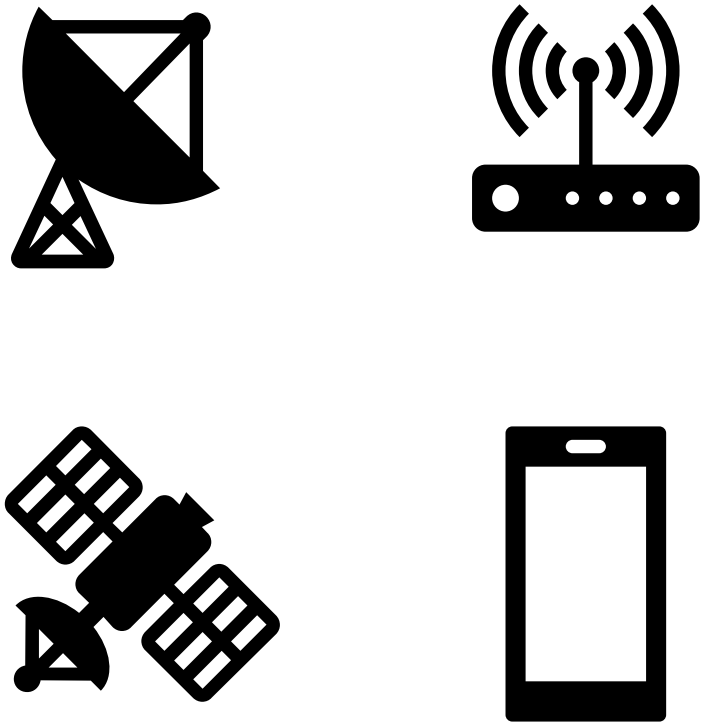


Hardware Implementation (HDL)



Case Study | *Pulse Detector*

1. Example Overview
2. Reference Pulse Detector
3. Pulse Detector Design
4. Prepare for Hardware Implementation
5. Fixed-point Conversion
6. HDL code generation, synthesis and verification



Pulse Detector | *Reference Design (MATLAB)*

Algorithm Stimulus

Reference Algorithm

Software Algorithm

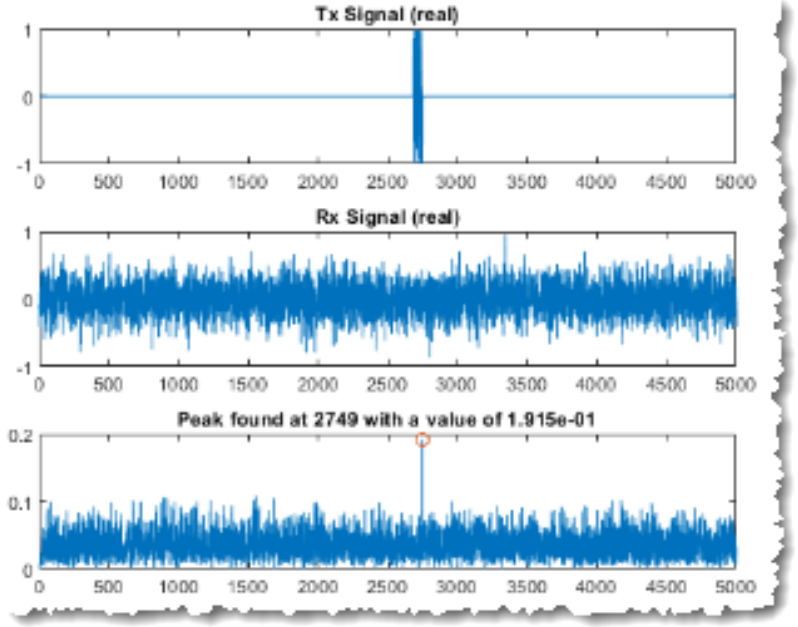
Analysis

Create input stimulus

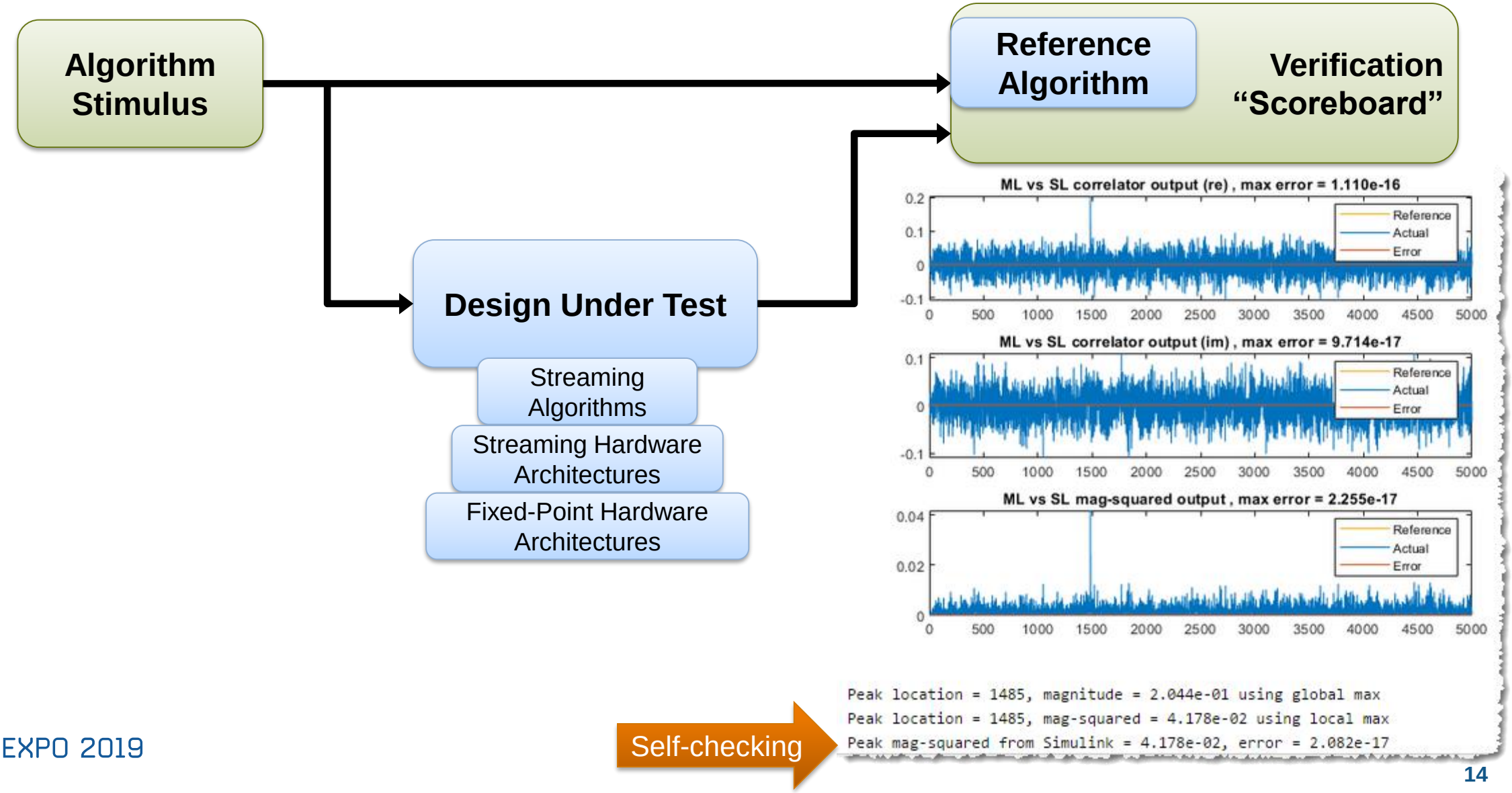
```
function [ CorrFilter, RxSignal, RxFxPt ] = pulse_detector_stim

% Create pulse to detect
rng('default');
PulseLen = 64;
theta = rand(PulseLen,1);
pulse = exp(1i*2*pi*theta);

% Insert pulse to Tx signal
rng('shuffle');
TxLen = 5000;
PulseLoc = randi(TxLen-PulseLen*2);

TxSignal = complex(zeros(TxLen,1));
TxSignal(PulseLoc:PulseLoc+PulseLen-1) = pulse;

% Create Rx signal by adding noise
Noise = complex(randn(TxLen,1),randn(TxLen,1));
RxSignal = TxSignal + Noise;

% Scale Rx signal to +/- one
scale1 = max([abs(real(RxSignal)); abs(imag(RxSignal))]);
```

MATLAB golden reference

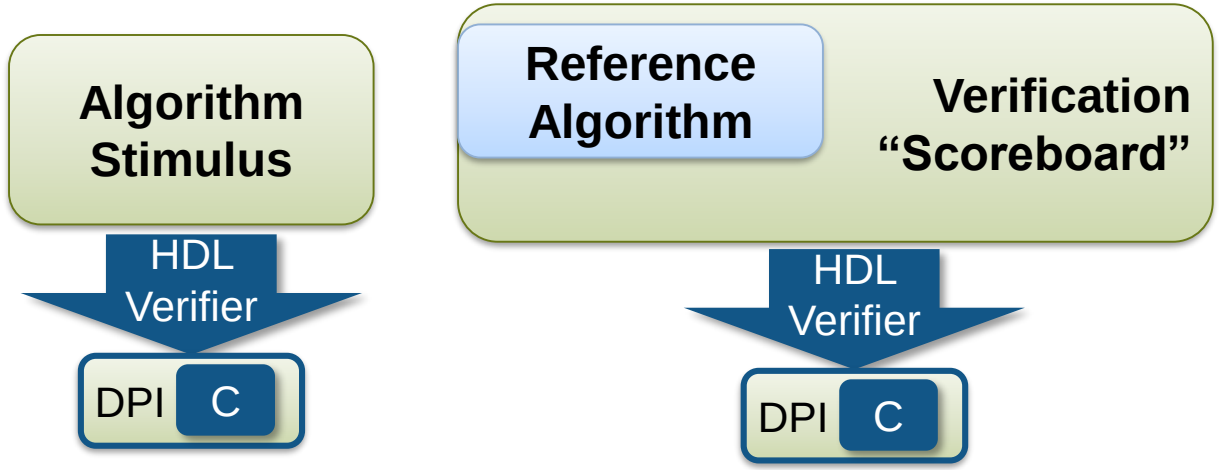
```
% Create matched filter coefficients
CorrFilter = conj(flip(pulse))/PulseLen;

% Correlate Rx signal against matched filter
FilterOut = filter(CorrFilter,1,RxSignal);

% Find peak magnitude & location
[peak, location] = max(abs(FilterOut));
```



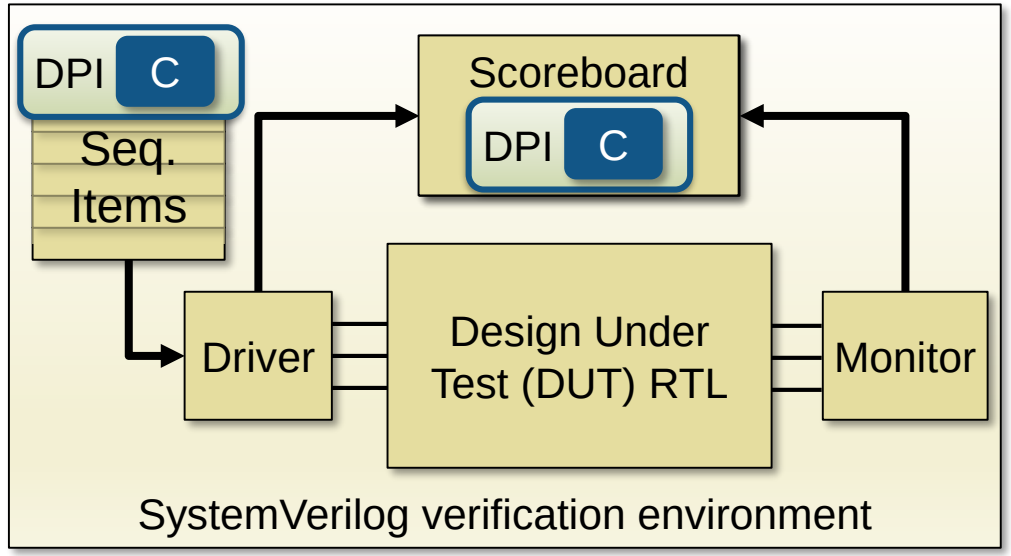
Pulse Detector | *Reference Design (MATLAB)*



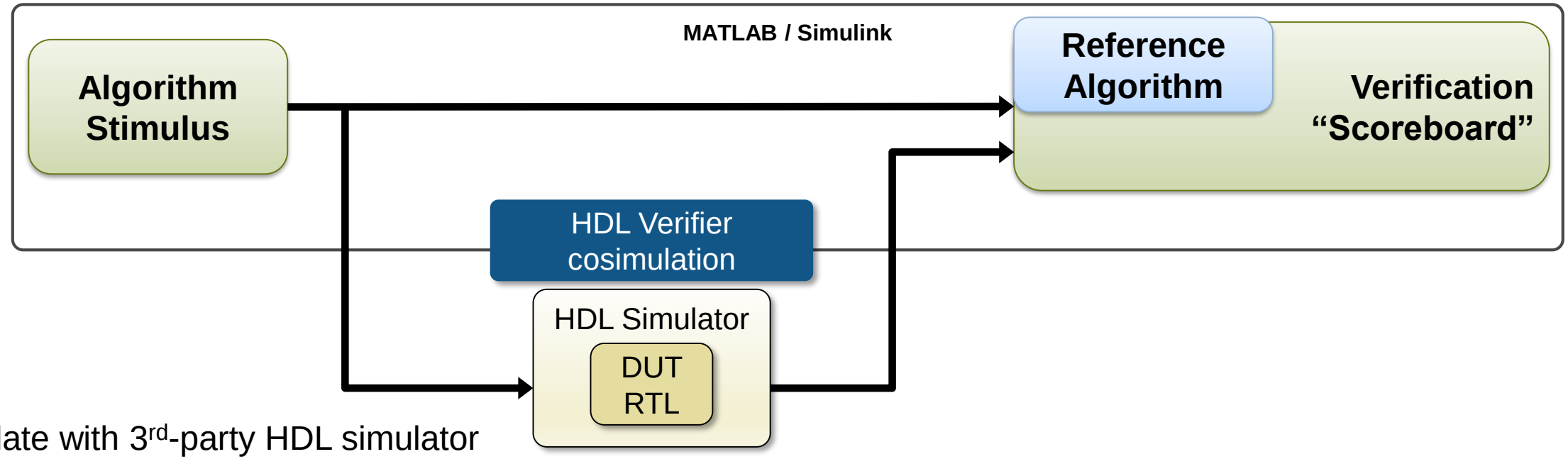
Pulse Detector | *Reference Design (MATLAB)*



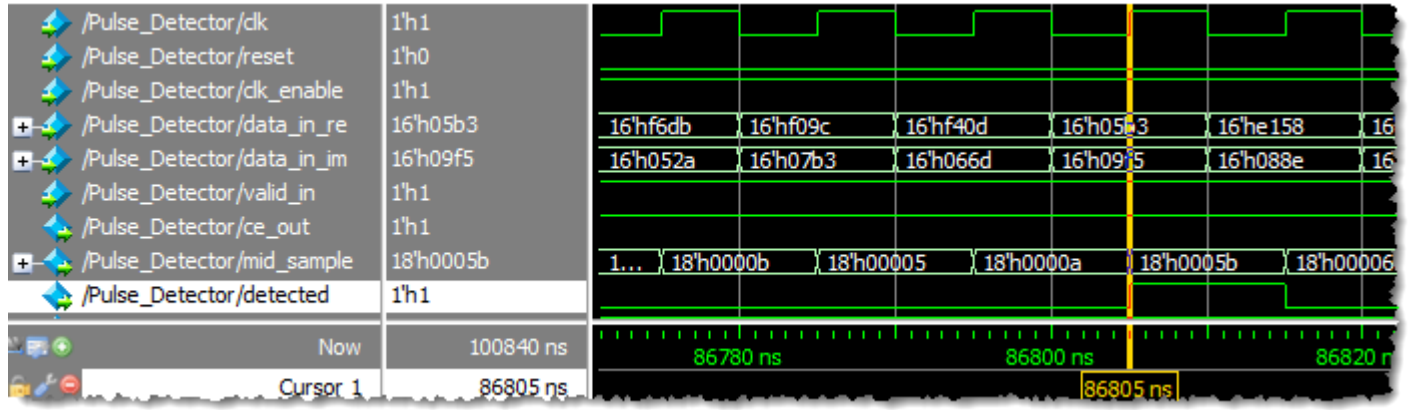
- Reuse MATLAB/Simulink models in verification
  - Scoreboard, stimulus, or models external to the RTL
  - Runs natively in SystemVerilog simulator
  - Eliminate re-work and miscommunication
  - Save testbench development time
  - Easy to update when requirements change



Pulse Detector | *Reference Design (MATLAB)*



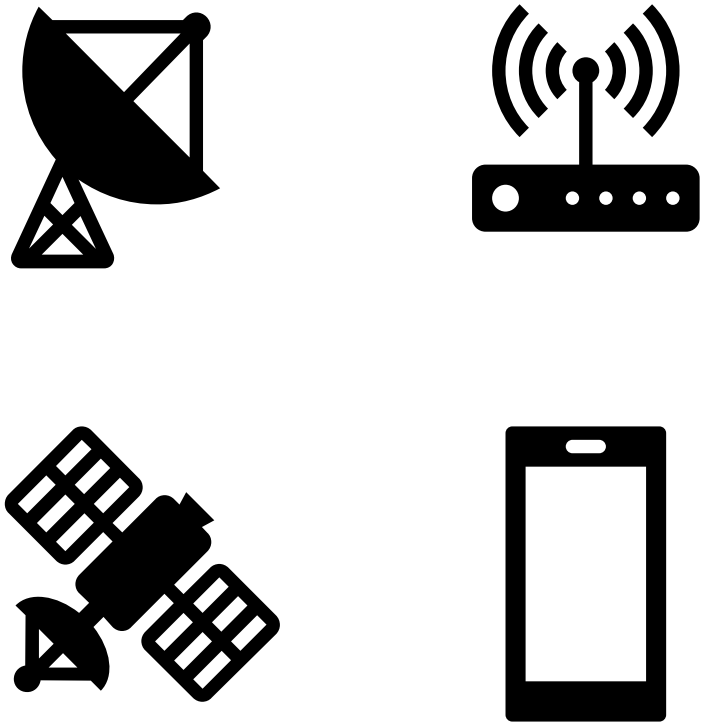
- Co-simulate with 3rd-party HDL simulator
  - Reuse MATLAB/Simulink test environment
  - Run HDL design in a supported simulator*
  - Generate co-simulation infrastructure and handshaking
  - Analyze both the design and test environment



* Mentor Graphics® ModelSim® or Questa®
Cadence® Incisive® or Xcelium™

Case Study | *Pulse Detector*

1. Example Overview
2. Reference Pulse Detector
3. Pulse Detector Design
4. Prepare for Hardware Implementation
5. Fixed-point Conversion
6. HDL code generation, synthesis and verification



Pulse Detector | *Design in Simulink*

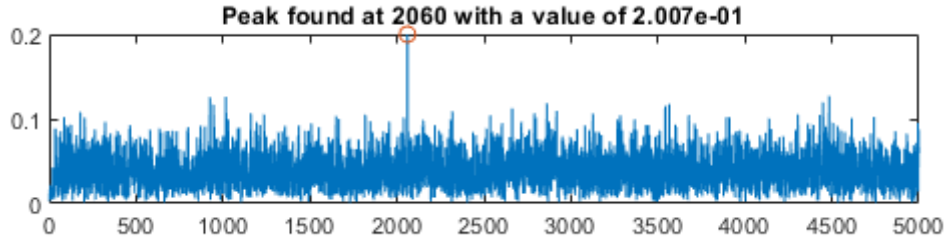
Streaming Architecture

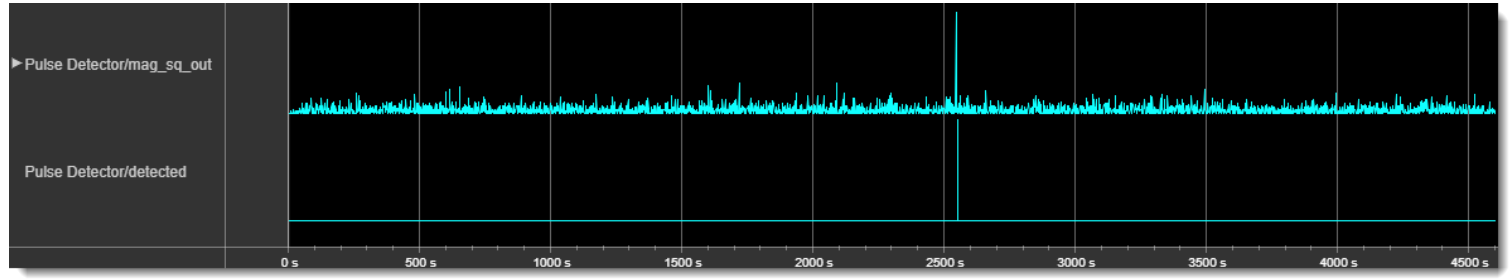
Hardware friendly implementation of peak finder

Instead of calculating the maximum value of the entire frame, we look for a local peak within a sliding window of the last 11 samples using the following criteria:

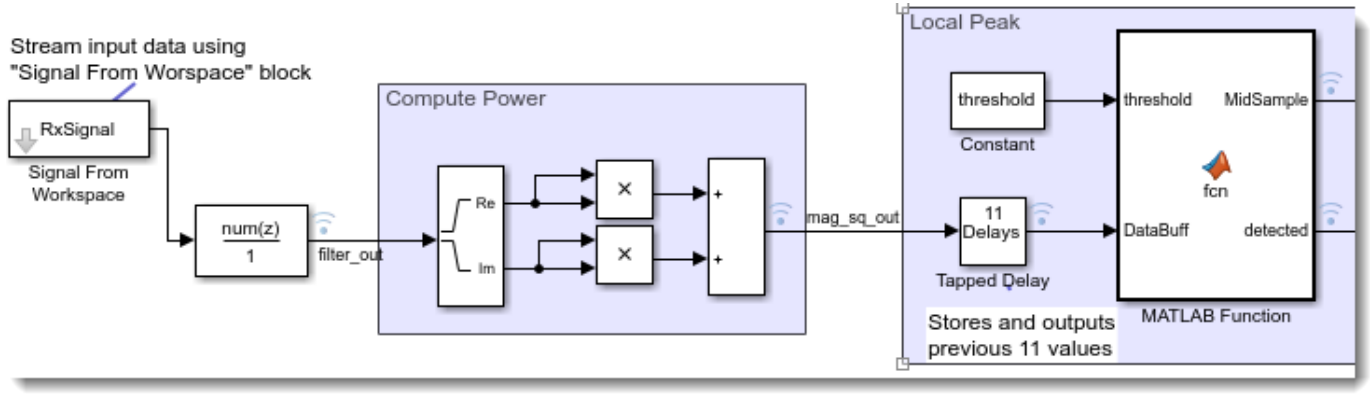
- The middle sample is the largest
- The middle sample is greater than a pre-defined threshold

```
WindowLen = 11;
MidIdx = ceil(WindowLen/2);
threshold = 0.03;

% Compute magnitude squared to avoid sqrt operation
MagSqOut = abs(FilterOut).^2;

% Sliding window operation
for n = 1:length(FilterOut)-WindowLen

    % Compare each value in the window to the middle sample via s
    DataBuff = MagSqOut(n:n+WindowLen-1);
    MidSample = DataBuff(MidIdx);
    CompareOut = DataBuff - MidSample; % this is a vector

    % if all values in the result are negative and the middle sam
    % greater than a threshold, it is a local max
    if all(CompareOut <= 0) && (MidSample > threshold)
        peak_2 = MidSample;
        location_2 = n + (MidIdx-1);
    end
end
```






```
% Simulate model
sim('pulse_detector_v1')

% Correlation filter output
FilterOutSL = squeeze(logsout.getE
compareData(real(FilterOut),real(F
compareData(imag(FilterOut),imag(F
```



_dev_AG (snapshot_centos_20190510) [En fonction] - Oracle VM VirtualBox : 1

Machine Écran Entrée Périphériques Aide

pulse_detector_streaming_arch * - Simulink

it View Display Diagram Simulation Analysis Code Tools Help

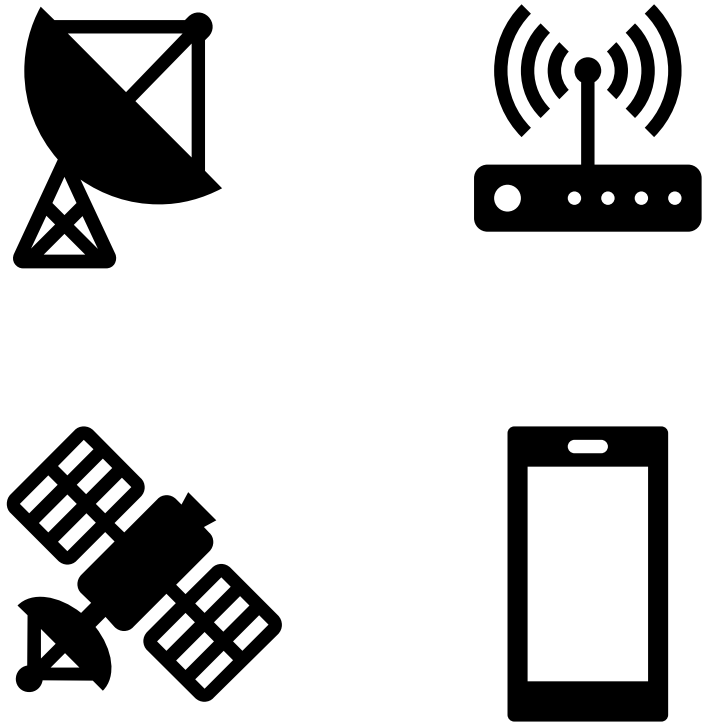
simTime Normal

se_detector_streaming_arch

pulse_detector_streaming_arch

Case Study | *Pulse Detector*

1. Example Overview
2. Reference Pulse Detector
3. Pulse Detector Design
4. Prepare for Hardware Implementation
5. Fixed-point Conversion
6. HDL code generation, synthesis and verification

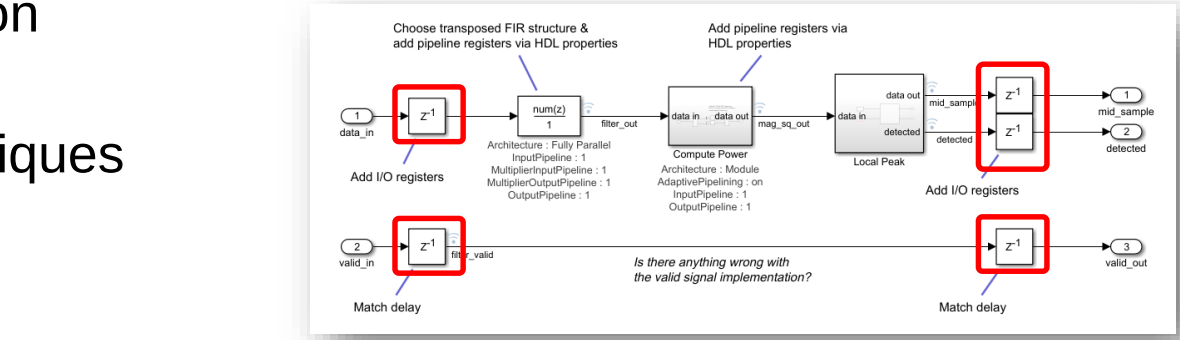


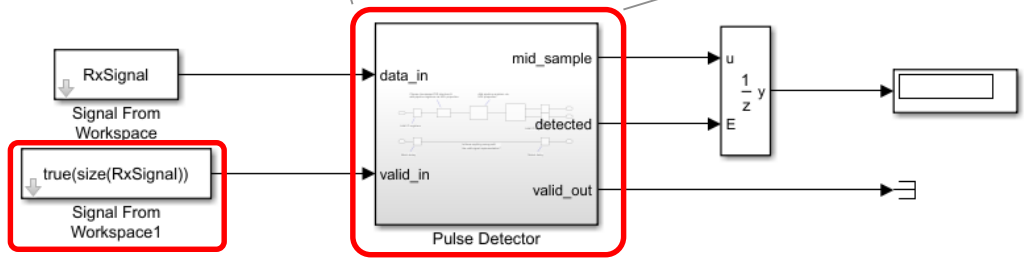
Pulse Detector | *Prepare for Hardware Design*

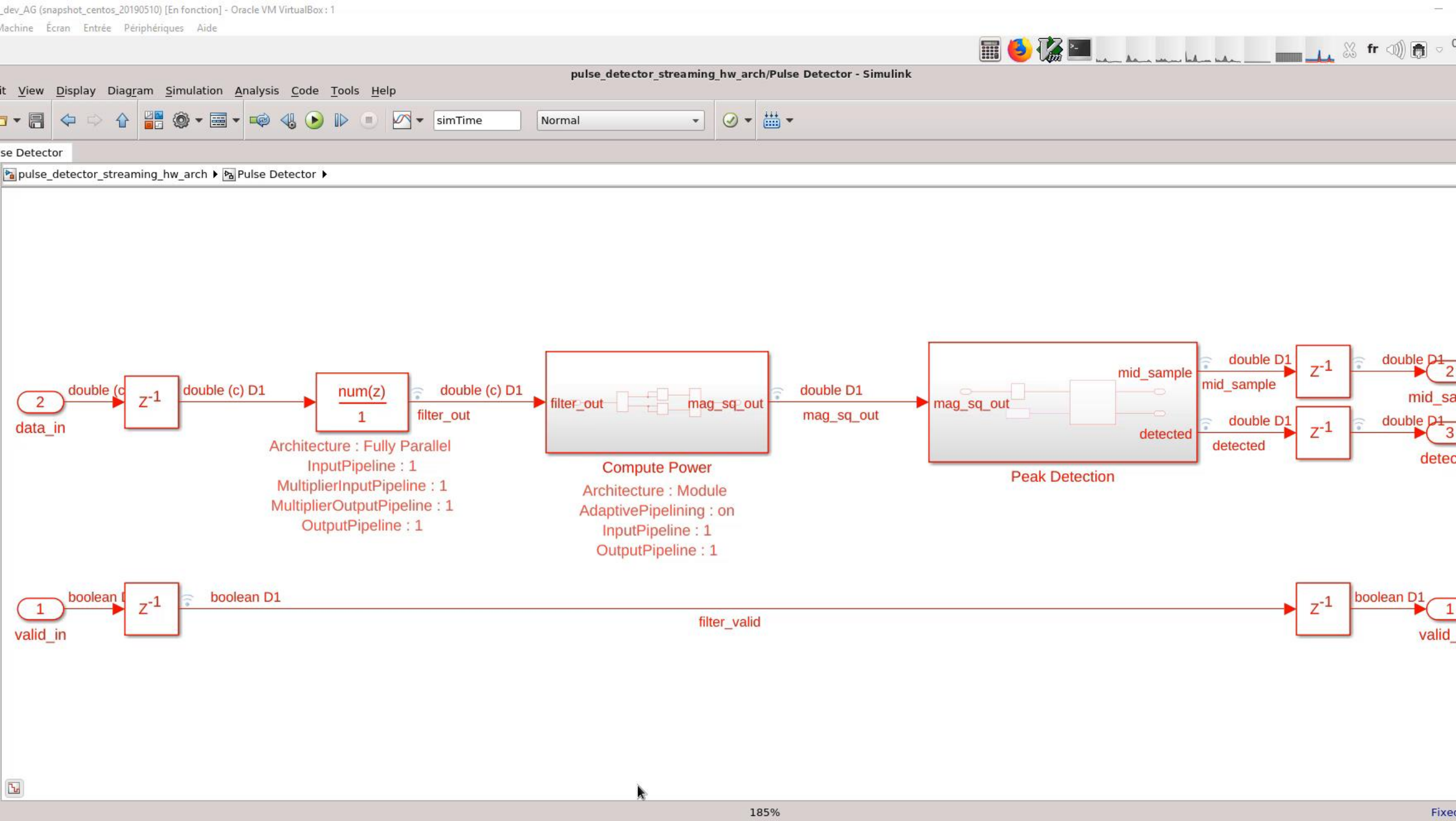
Micro Architecture

In this step, we:

- prepare the model for HDL code generation
- pipeline the data path using various techniques
- add data valid control signal
- verify against MATLAB golden reference

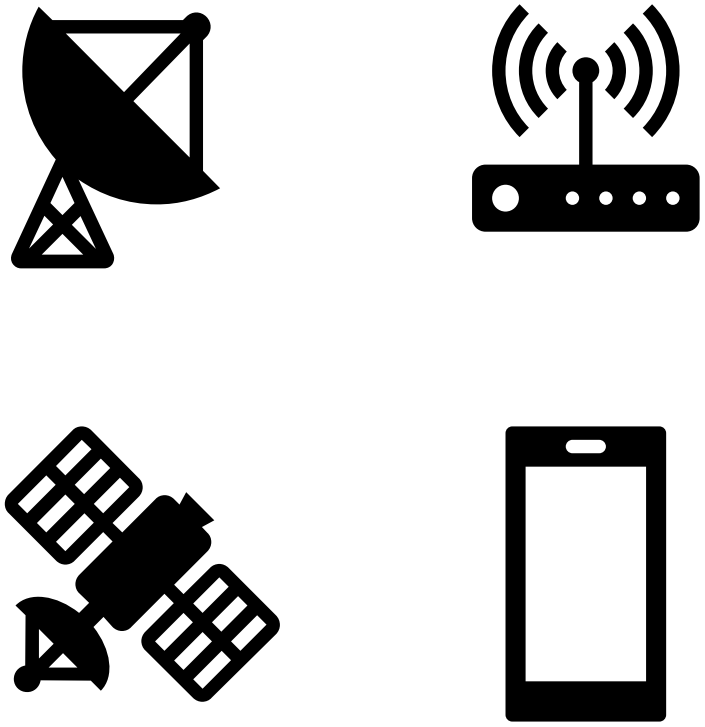





_dev_AG (snapshot_centos_20190510) [En fonction] - Oracle VM VirtualBox : 1
Machine Écran Entrée Périphériques Aide
pulse_detector_streaming_hw_arch/Pulse Detector - Simulink
it View Display Diagram Simulation Analysis Code Tools Help
simTime
Normal
se Detector
pulse_detector_streaming_hw_arch
Pulse Detector
2
data_in
double (c
z-1
double (c) D1
num(z)
1
double (c) D1
filter_out
Architecture : Fully Parallel
InputPipeline : 1
MultiplierInputPipeline : 1
MultiplierOutputPipeline : 1
OutputPipeline : 1
filter_out
mag_sq_out
Compute Power
Architecture : Module
AdaptivePipelining : on
InputPipeline : 1
OutputPipeline : 1
double D1
mag_sq_out
mag_sq_out
mid_sample
detected
Peak Detection
double D1
mid_sample
double D1
detected
z-1
z-1
double D1
2
mid_sa
double D1
3
detec
1
valid_in
boolean
z-1
boolean D1
filter_valid
z-1
boolean D1
1
valid_
185%
Fixe

Case Study | *Pulse Detector*

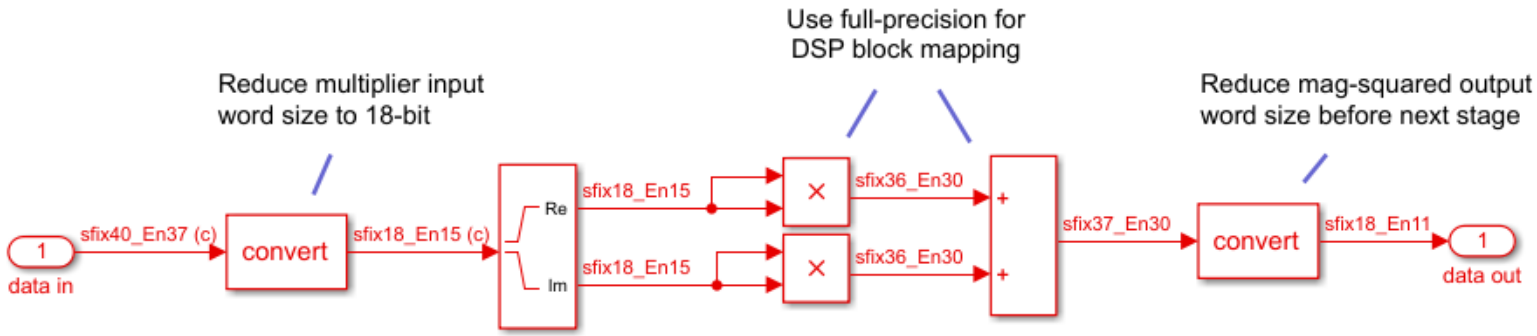
1. Example Overview
2. Reference Pulse Detector
3. Pulse Detector Design
4. Prepare for Hardware Implementation
5. Fixed-point Conversion
6. HDL code generation, synthesis and verification



Pulse Detector | *Fixed-Point Conversion*

In this step, we:

- convert the model to fixed-point



- compare the Simulink fixed-point model to the MATLAB golden reference

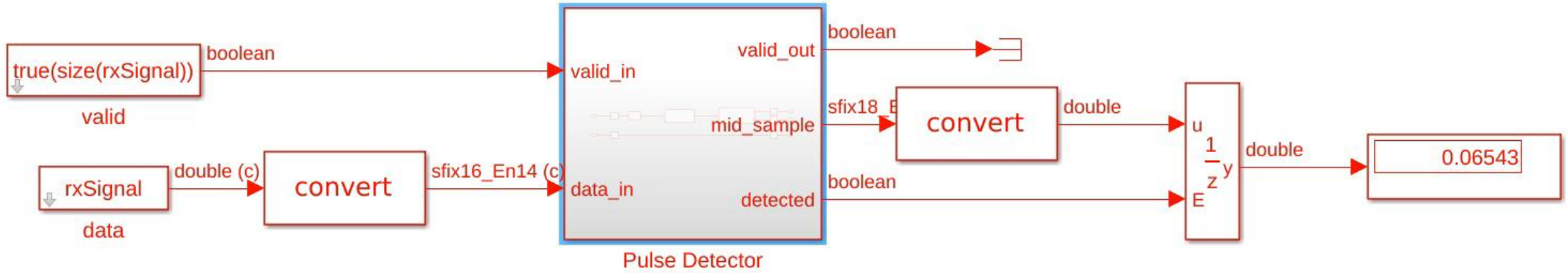
pulse_detector_streaming_hw_fp_arch - Simulink

it View Display Diagram Simulation Analysis Code Tools Help

simTime Normal

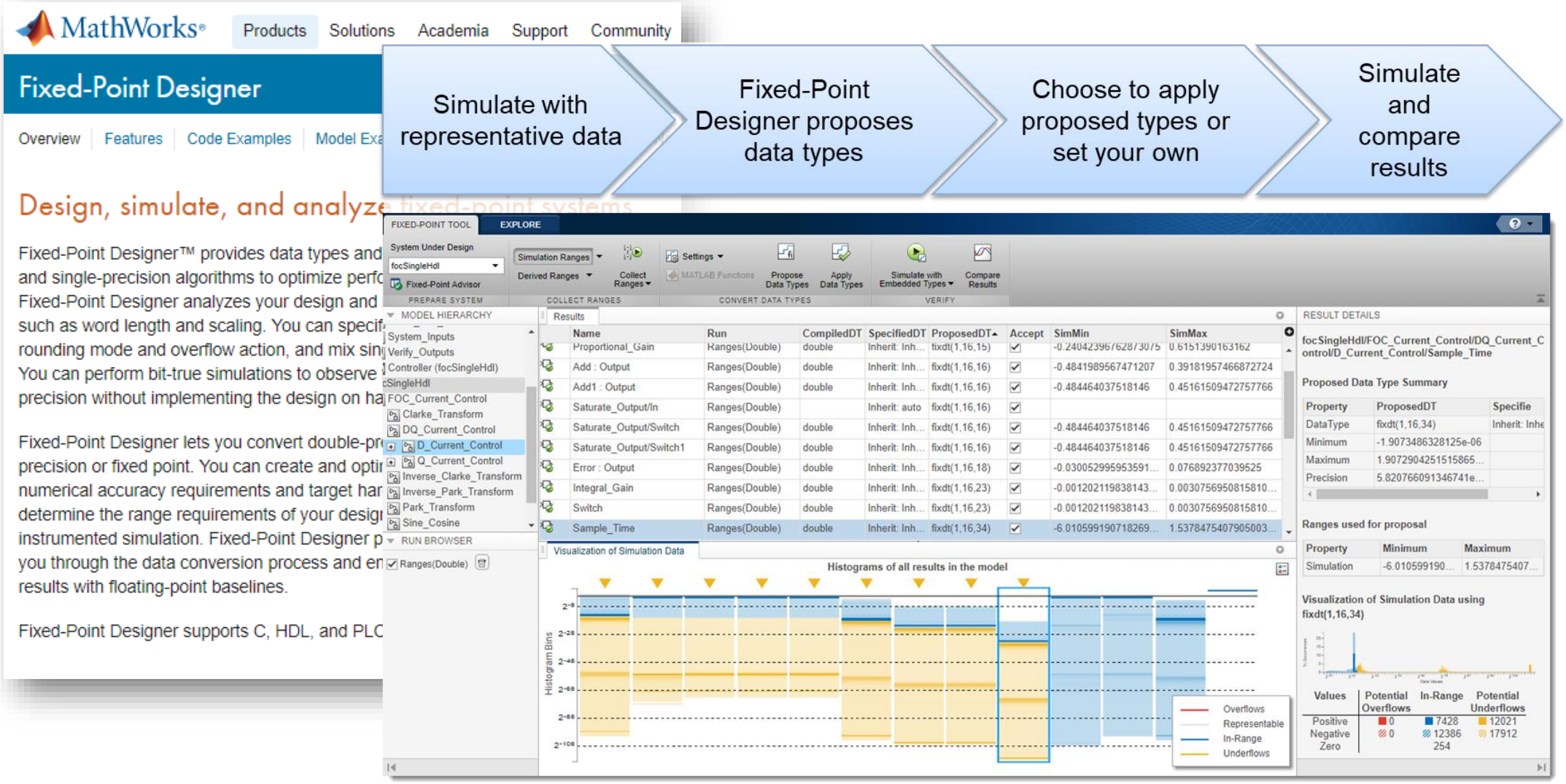
se_detector_streaming_hw_fp_arch

pulse_detector_streaming_hw_fp_arch



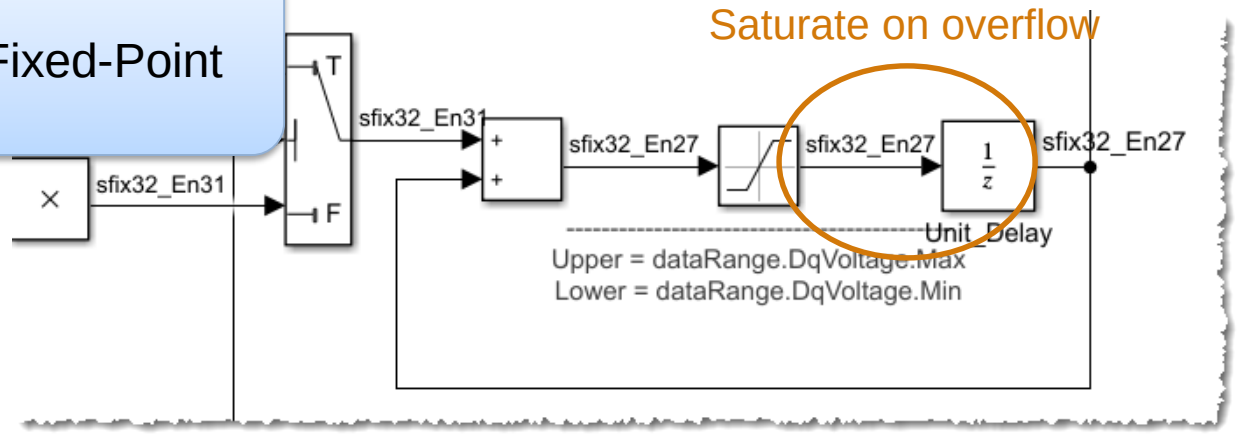
Some words about Fixed-Point conversion...

Fixed-Point Conversion | *Automated Approach*

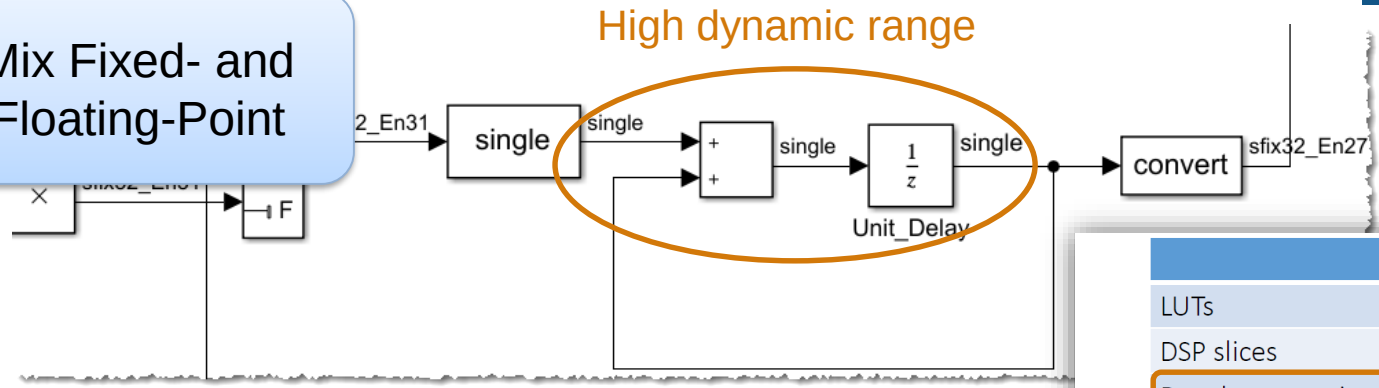


Fixed-Point Conversion | *Native Floating-Point*

Fixed-Point



Mix Fixed- and Floating-Point



embeddedaward2017

HDL Coder Native Floating Point

- Extensive math and trigonometric operator support
- Optimal implementations without sacrificing numerical accuracy
- Mix floating- and fixed-point operations
- Generate target-independent HDL

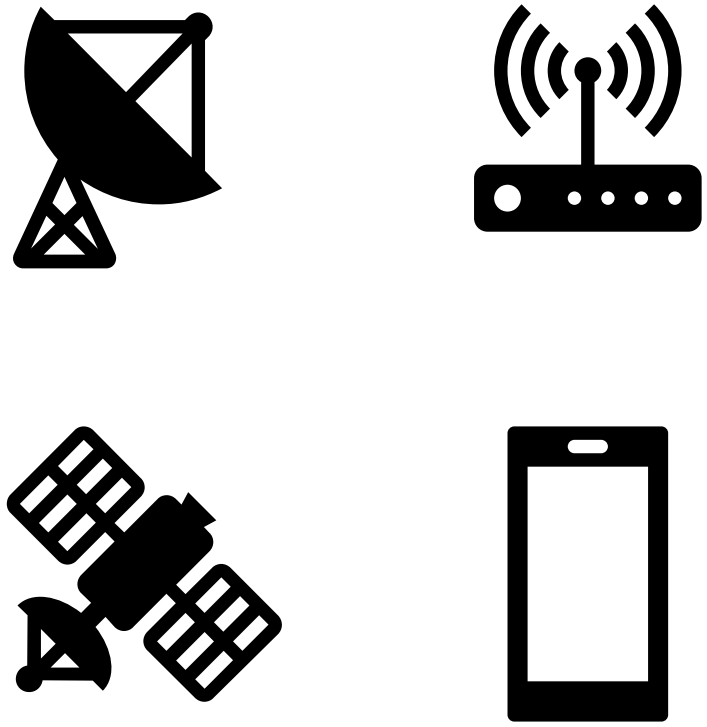
	Fixed point	Floating point
LUTs	10k	25k
DSP slices	50	100
Development time	~1 week	~1 day

~2x more resources
~5x less development effort

DEMCON

Case Study | *Pulse Detector*

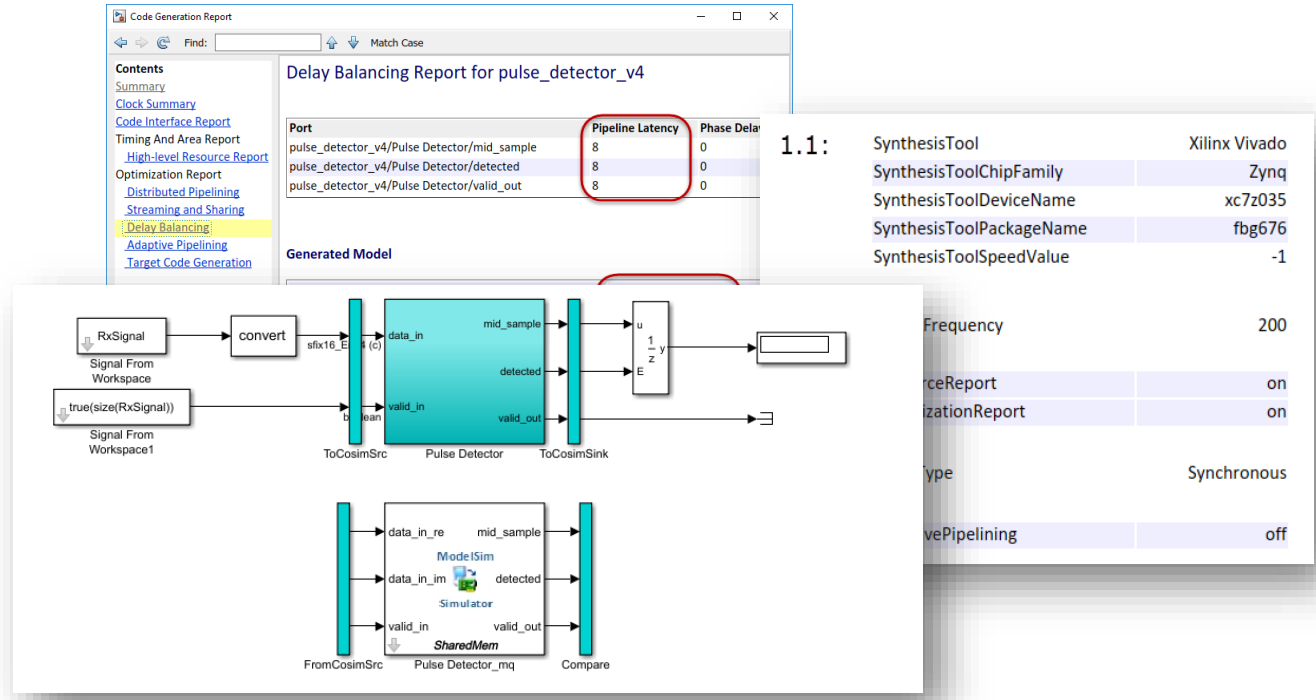
1. Example Overview
2. Reference Pulse Detector
3. Pulse Detector Design
4. Prepare for Hardware Implementation
5. Fixed-point Conversion
6. HDL code generation, synthesis and verification



Pulse Detector | *HDL Code Generation and Verification*

In this step, we:

- generate HDL code and reports
- synthesize the design using Xilinx Vivado
- verify the design



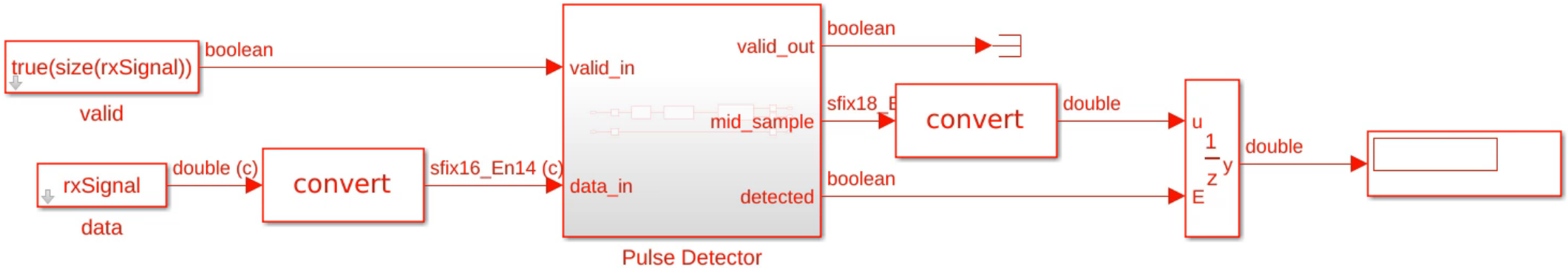
pulse_detector_streaming_hw_fp_arch_code_gen - Simulink

Edit View Display Diagram Simulation Analysis Code Tools Help

simTime Normal

pulse_detector_streaming_hw_fp_arch_code_gen | Peak Detection | Pulse Detector | Filter | Compute Power

pulse_detector_streaming_hw_fp_arch_code_gen



HDL Workflow Advisor - pulse_detector_streaming_hw_fp_arch_code_gen/Pulse Detector

File Edit Run Help

Find:

- HDL Workflow Advisor
  - 1. Set Target
    - ^1.1. Set Target Device and Synthe
    - 1.2. Set Target Frequency
  - 2. Prepare Model For HDL Code Genera
    - 2.1. Check Global Settings
    - ^2.2. Check Algebraic Loops
    - ^2.3. Check Block Compatibility
    - ^2.4. Check Sample Times
  - 3. HDL Code Generation
    - 3.1. Set Code Generation Options
      - 3.1.1. Set Basic Options
      - 3.1.2. Set Report Options
      - 3.1.3. Set Advanced Options
      - 3.1.4. Set Optimization Options
      - 3.1.5. Set Testbench Options
    - ^3.2. Generate RTL Code and Testb
    - 3.3. Verify with HDL Cosimulation
  - 4. FPGA Synthesis and Analysis
    - 4.1. Create Project
    - 4.2. Perform Synthesis and P/R
      - 4.2.1. Run Synthesis
      - 4.2.2. Run Implementation
    - 4.3. Annotate Model with Synthesis

3.1.5. Set Testbench Options

Test Bench Generation Output

- [ ] HDL test bench
- [ ] Cosimulation model
- [ ] SystemVerilog DPI test bench

Simulation tool: Mentor Graphics Modelsim [ ] HDL code coverage

Configuration

Test bench name postfix:	_tb
[x] Force clock	
Clock high time (ns):	5
Clock low time (ns):	5
Hold time (ns):	2
Setup time (ns):	8
[x] Force clock enable	
Clock enable delay (in clock cycles):	1
[x] Force reset	
Reset length (in clock cycles):	2
[x] Hold input data between samples	
[ ] Initialize test bench inputs	
[ ] Multi-file test bench	
Test bench data file name postfix:	_data
Test bench reference postfix:	_ref

Help Apply

simTime Normal

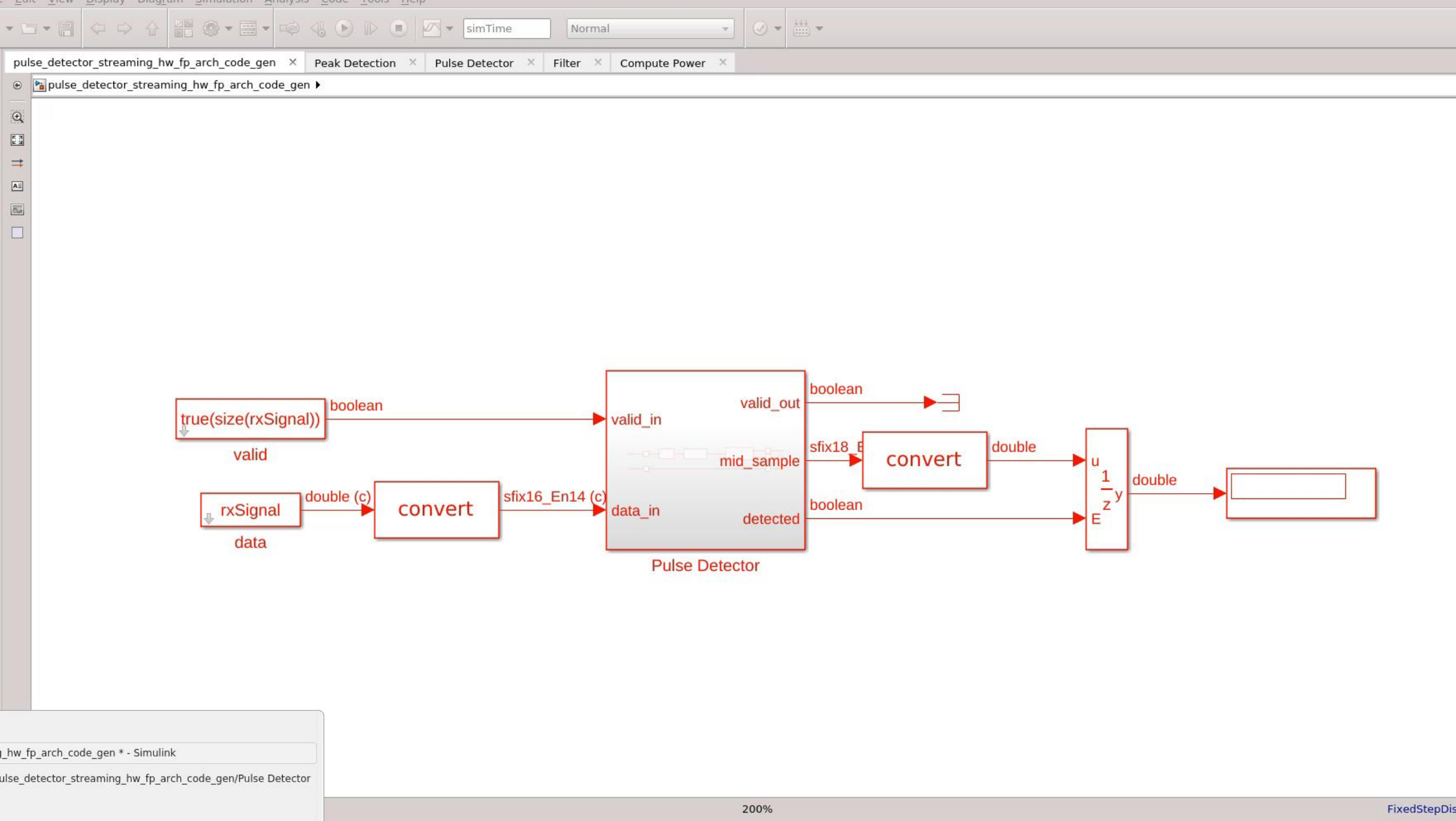
pulse_detector_streaming_hw_f

pulse_detector_streaming

239%

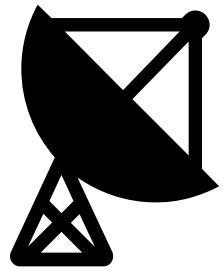
FixedStepDis

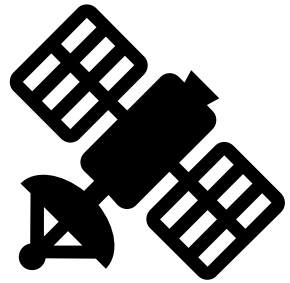
Is there more?

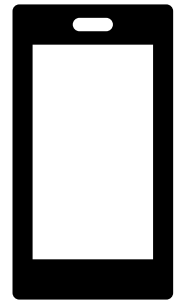

simTime
Normal
pulse_detector_streaming_hw_fp_arch_code_gen
Peak Detection
Pulse Detector
Filter
Compute Power
pulse_detector_streaming_hw_fp_arch_code_gen
true(size(rxSignal))
valid
boolean
rxSignal
data
double (c)
convert
sfix16_En14 (c)
valid_in
data_in
valid_out
mid_sample
detected
Pulse Detector
boolean
sfix18_E
convert
double
boolean
u
1
z
y
E
double
_hw_fp_arch_code_gen * - Simulink
ulse_detector_streaming_hw_fp_arch_code_gen/Pulse Detector
200%
FixedStepDis

Case Study | *Pulse Detector*

1. Example Overview
2. Reference Pulse Detector
3. Pulse Detector Design
4. Prepare for Hardware Impleme
5. Fixed-point Conversion
6. HDL code generation, synthesis and verification







Case Study | Workflow Summary

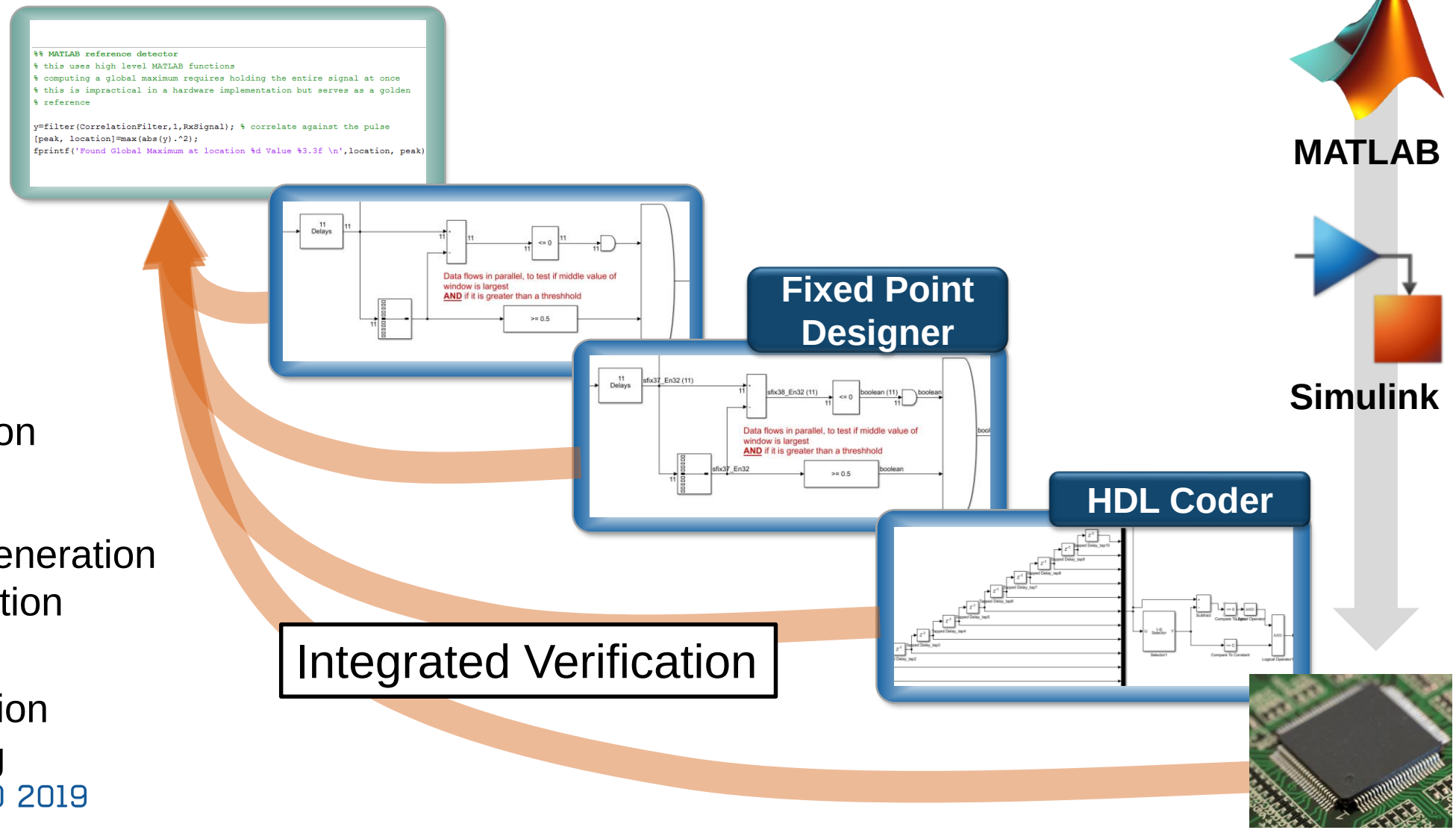
Golden Reference

Hardware Architecture

Fixed-point Implementation

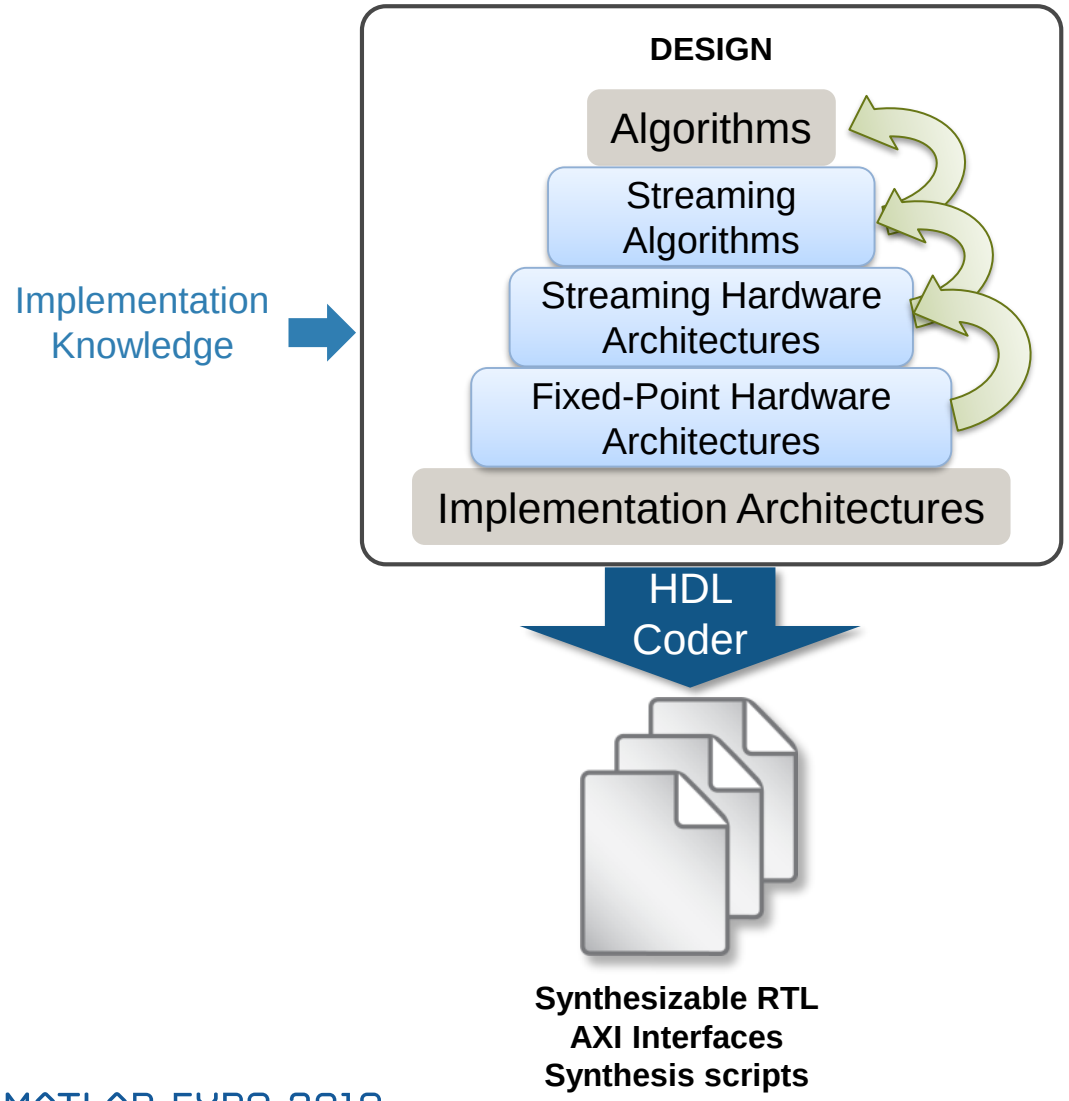
HDL Code Generation and Optimization

HDL Verification and Targeting



A few more words about code generation ...

Automatically Generate Production RTL

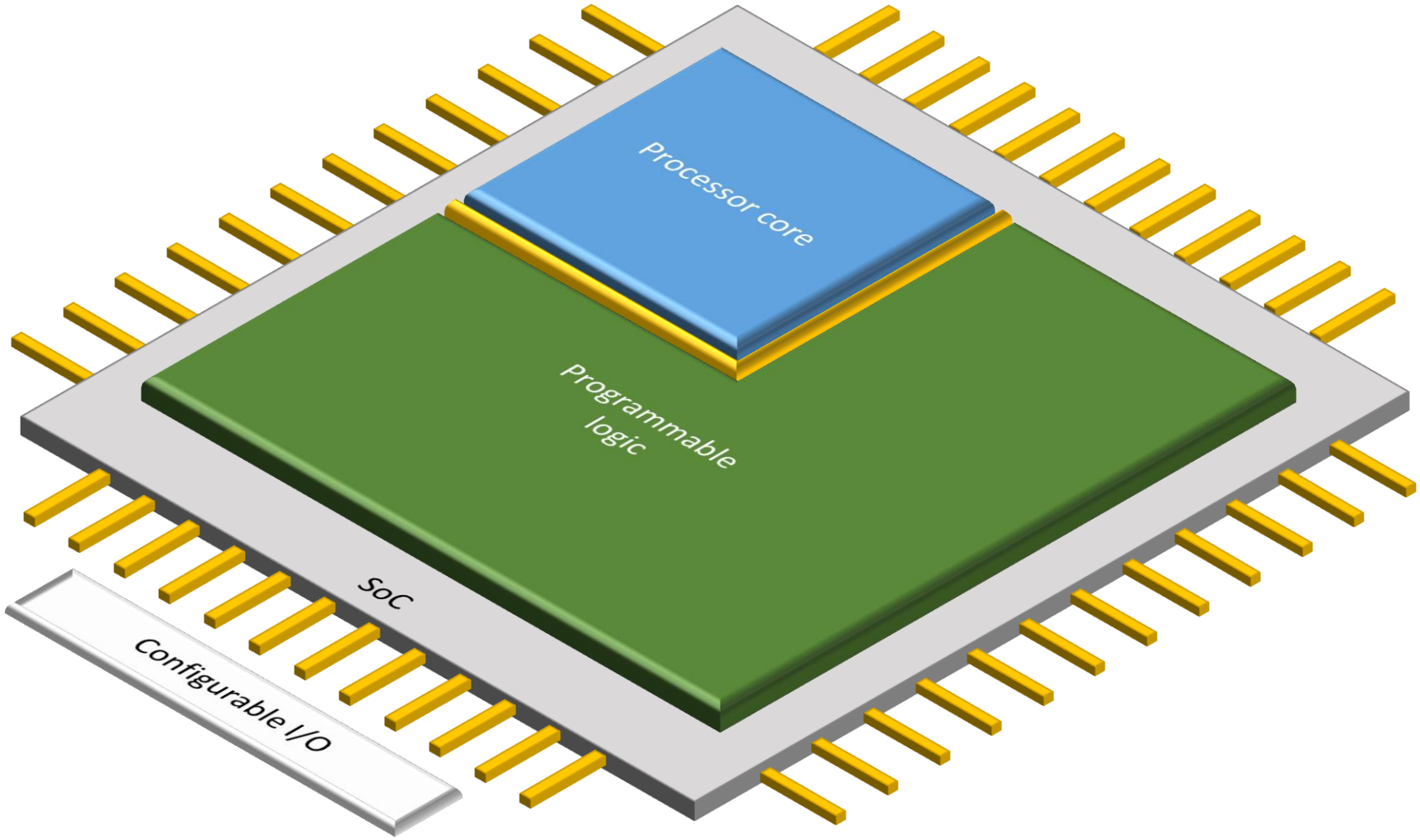


- Choose from over 300 supported blocks
  - Including MATLAB functions and Stateflow charts
- Quickly explore implementation options
- Generate readable, traceable Verilog/VHDL
  - Optionally generate AXI interfaces with IP core
- Production-proven across a variety of applications and FPGA, ASIC, and SoC targets

Agenda

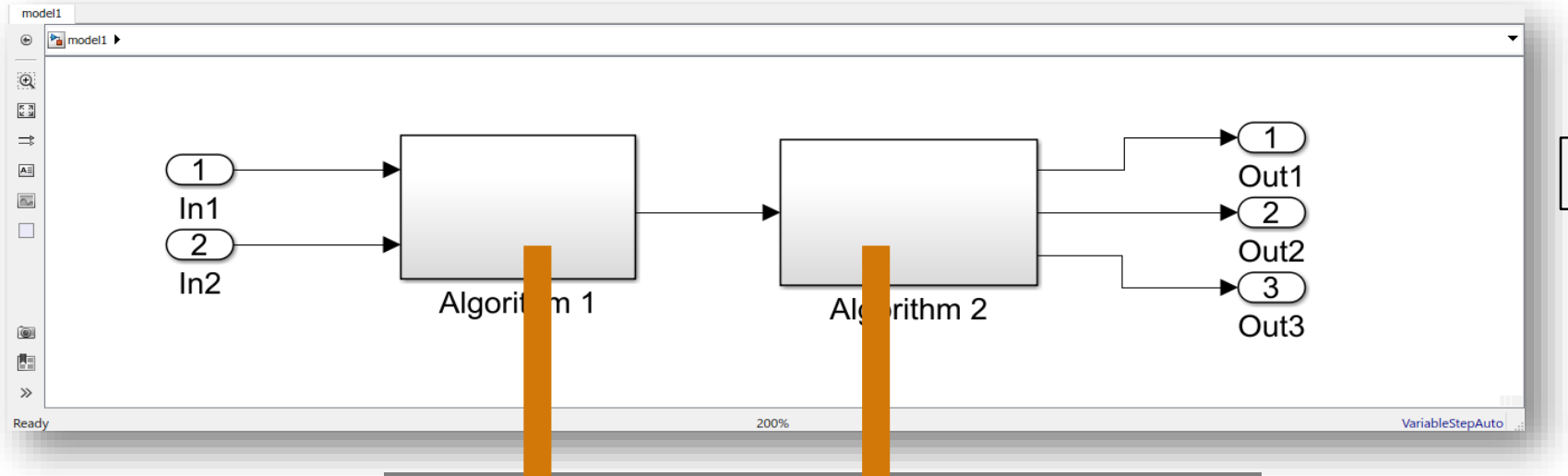
- Why Model-Based Design for FPGA, ASIC, or SoC?
- Case Study – Pulse Detector
- HW/SW Co-Design
- Customer results

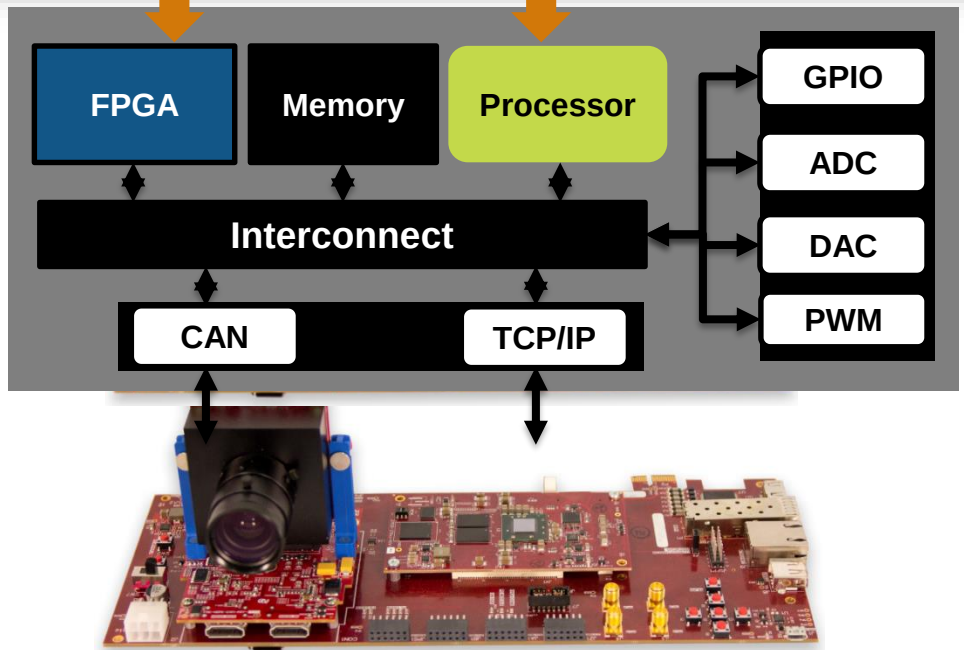
HW/SW Design



Model Based Design Workflow for SoC

Deploy to Hardware with Coders and HW Support Package





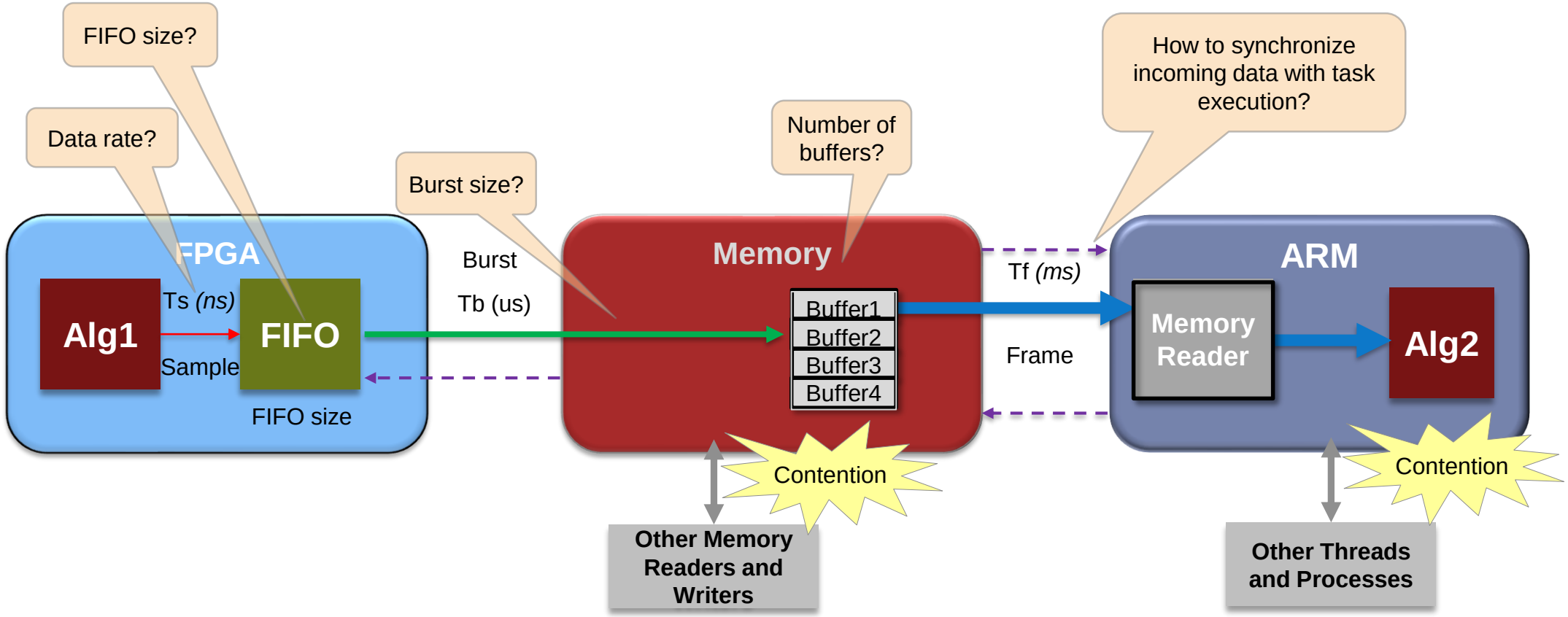
Algorithmic Model

Algorithmic Code

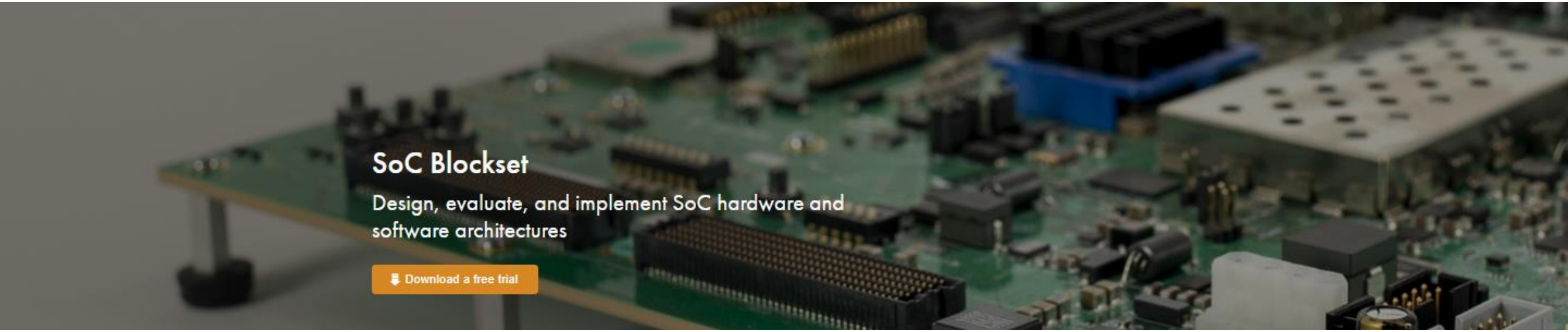
HW Support Package (Reference Design)

Hardware Platform

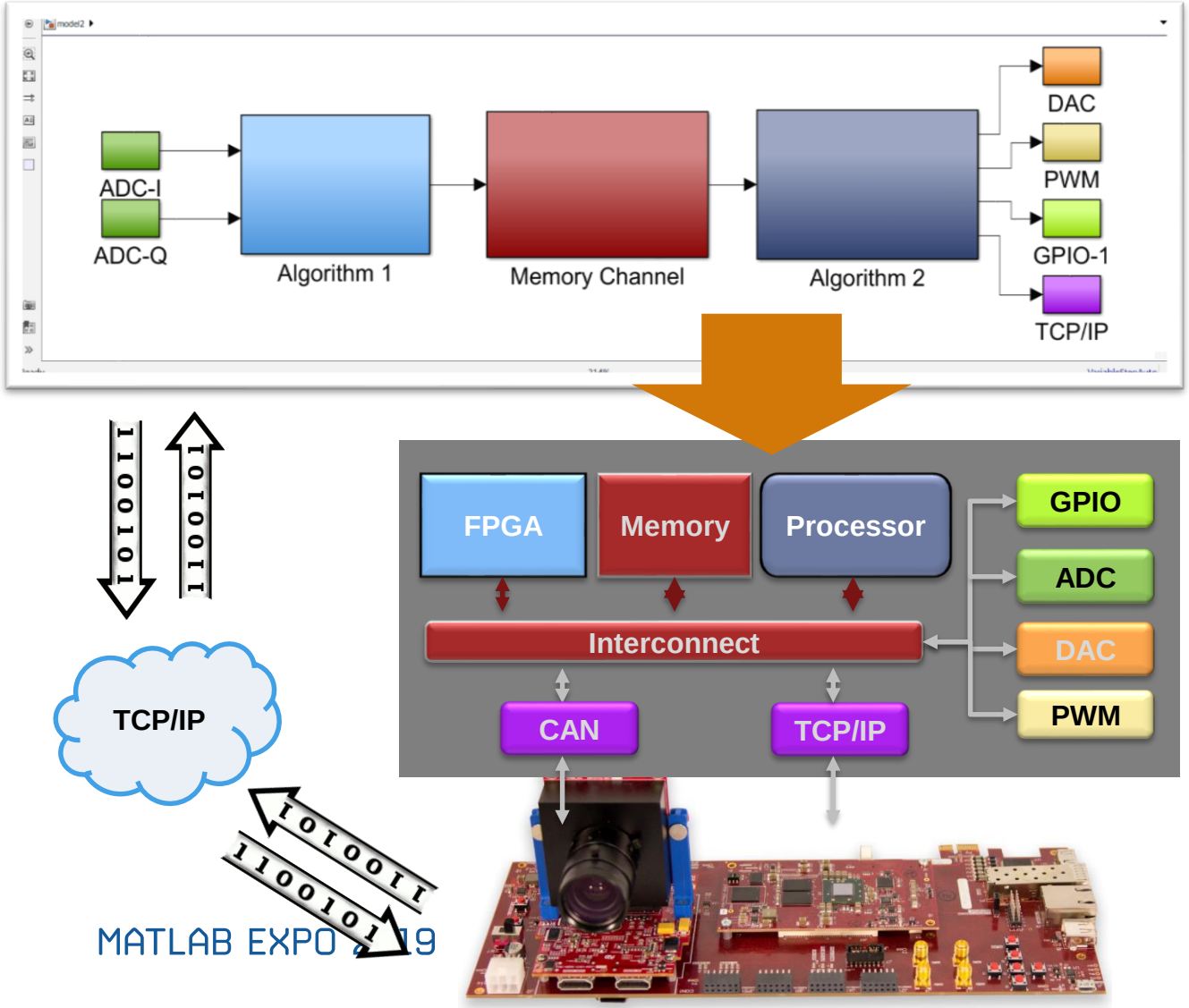
Actual Data Exchange Between FPGA and Processor



SoC Blockset | *Model and Simulate SoC Architecture*



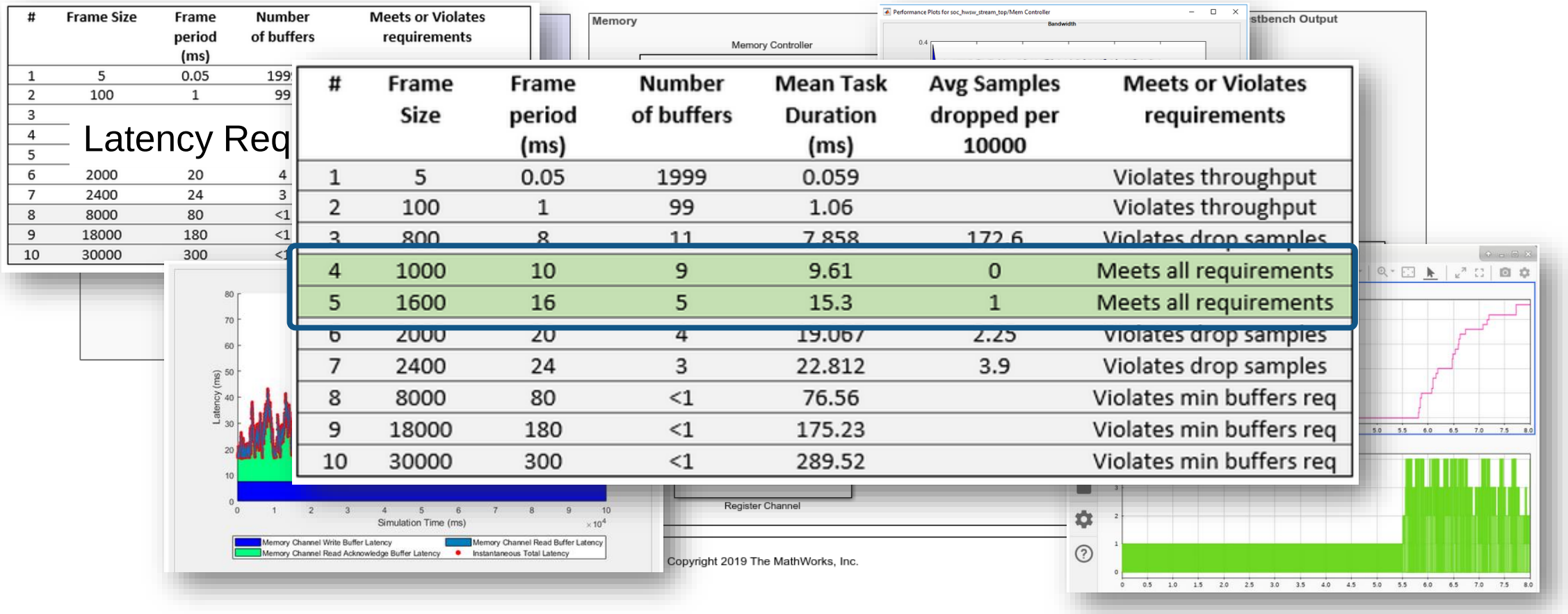
SoC Blockset | *Model and Simulate SoC Architecture*



- Simulate algorithms as well as hardware/software architecture
  - Memory
  - Internal/external connectivity
  - I/O
  - Task scheduling
- Deploy on support hardware
- Profile performance using external mode

SoC Blockset | *Example*

Streaming Data from Hardware to Software



#	Frame Size	Frame period (ms)	Number of buffers	Mean Task Duration (ms)	Avg Samples dropped per 10000	Meets or Violates requirements
1	5	0.05	1999	0.059		Violates throughput
2	100	1	99	1.06		Violates throughput
3	800	8	11	7.858	172.6	Violates drop samples
4	1000	10	9	9.61	0	Meets all requirements
5	1600	16	5	15.3	1	Meets all requirements
6	2000	20	4	19.067	2.25	Violates drop samples
7	2400	24	3	22.812	3.9	Violates drop samples
8	8000	80	<1	76.56		Violates min buffers req
9	18000	180	<1	175.23		Violates min buffers req
10	30000	300	<1	289.52		Violates min buffers req

SoC Blockset | *Workflow Summary*

Simulate SoC Architectures

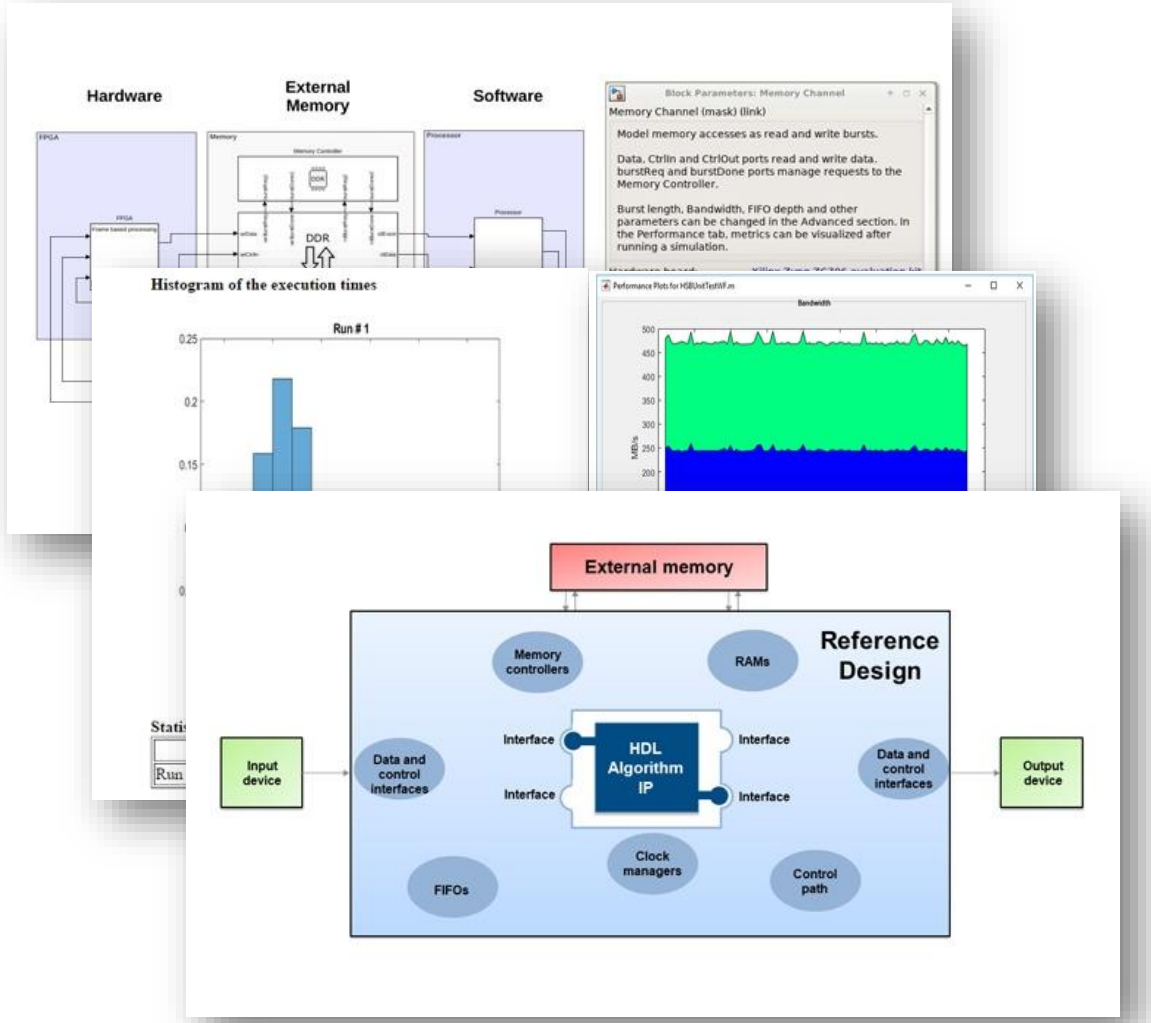
Develop and combine software algorithms, hardware logic, memory systems, and I/O devices into your SoC application. Evaluate architecture alternatives before deploying to hardware.

Analyze System Performance

Evaluate memory performance and task execution through simulation and perform on-device profiling.

Deploy to SoC and FPGA Devices

Generate reference designs and RTL code for programmable logic. Generate C/C++ code for processor tasks.



Agenda

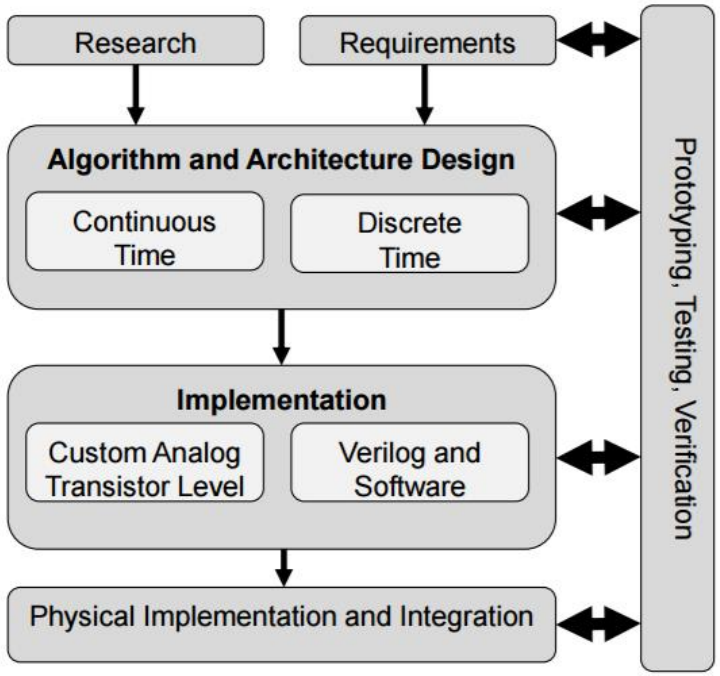
- Why Model-Based Design for FPGA, ASIC, or SoC?
- Case Study – Pulse Detector
- HW/SW Co-Design
- Customer results

Results at Allegro Microsystems

Allegro MicroSystems, LLC

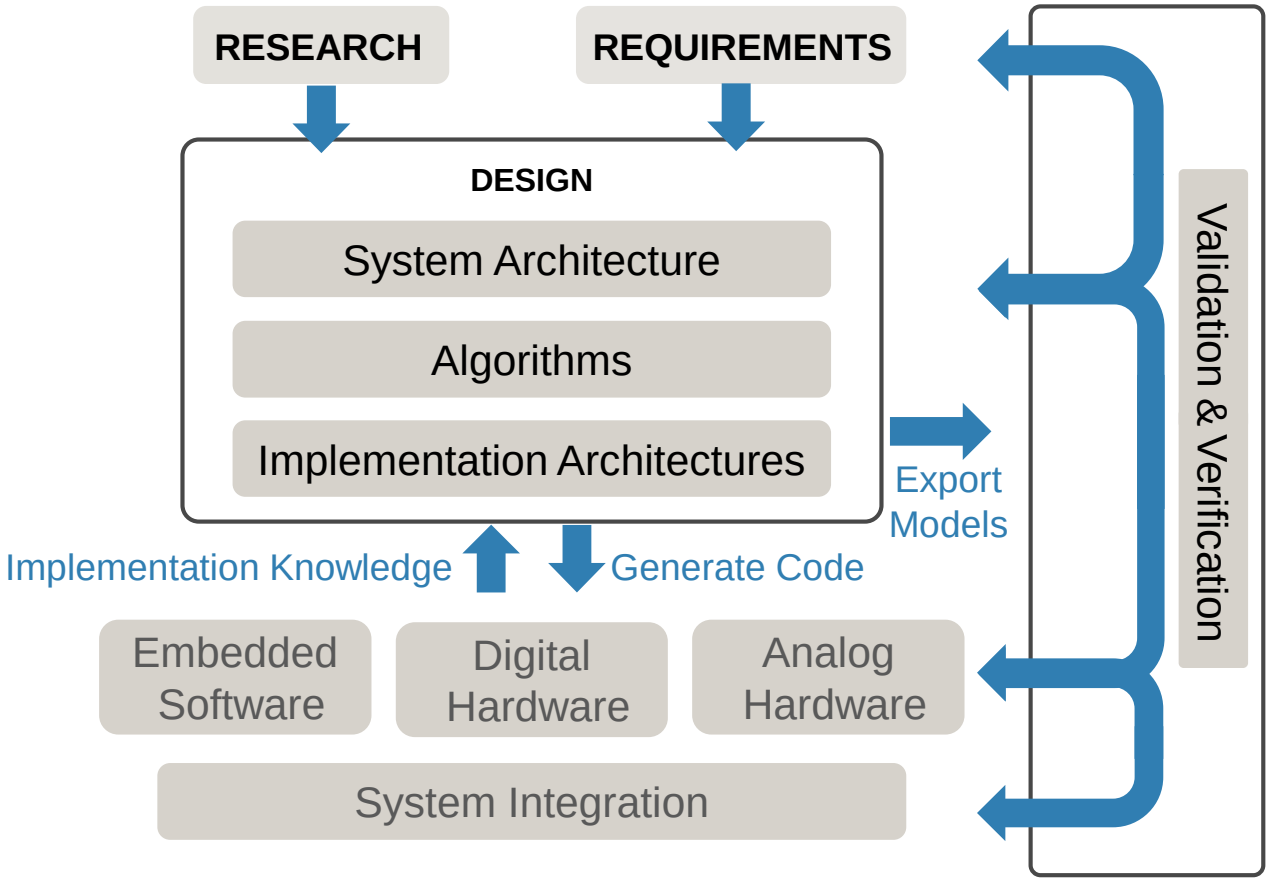
Allegro Confidential Information

The Enlightenment: Model Based Design



- **Architecture and Algorithm Design Evolve into Executable Specifications**
- **Front load testing and verification**
- **Development is "parallelized"**
- **Continuous Equivalency Testing is utilized**
- **.... And of course auto-generated production code**

Getting Started Collaborating with Model-Based Design



- ❑ Refine algorithm toward implementation
- ❑ Verify refinements versus previous versions
- ❑ Generate verification models
- ❑ Add hardware implementation detail and generate optimized RTL
- ❑ Simulate System-on-Chip architecture

- ➢ Eliminate communication gaps
- ➢ Key decisions made via cross-skill collaboration
- ➢ Identify and address system-level issues before implementing subsystems
- ➢ Adapt to changing requirements with agility

Learn More

- **Visit FPGA & SoC booth!**

- Next steps to get started with:
  - Verification: Improve RTL Verification by Connecting to MATLAB webinar
  - Fixed-point quantization: Fixed-Point Made Easy webinar
  - Incremental refinement, HDL code generation: HDL self-guided tutorial
  - SoC Blockset: Getting Started with SoC Blockset