FPGA Design & Prototyping Using MATLAB and Simulink

Curie Chung
MathWorks Application Engineering

Agenda

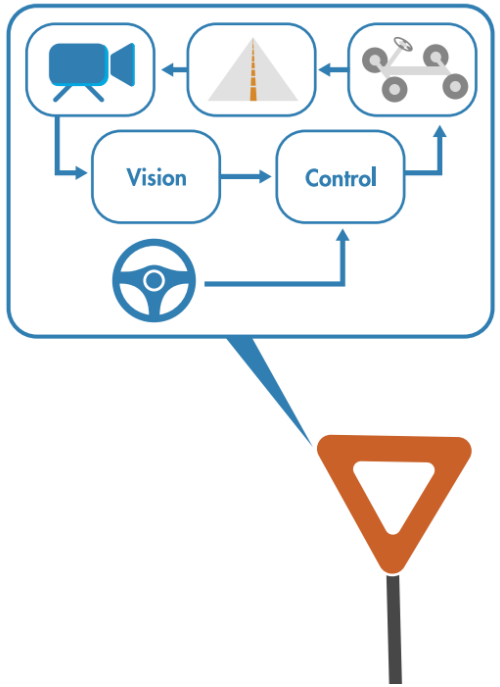
From Algorithm to HDL and Hardware Prototyping with Model Based Design

In this seminar, you will learn about the workflow for implementing a MATLAB algorithm on an FPGA using HDL Coder and Simulink. The demonstration will include:

- Defining a MATLAB pulse detection algorithm commonly used in digital communications and radar applications
- Implementing a streaming, fixed-point architecture in Simulink
- Simulink modeling best practices for creating efficient hardware
- Generating and synthesizing HDL for an FPGA device

You will also learn about real-time FPGA prototyping and debugging solutions in HDL Coder and HDL Verifier.

New Applications Require High-Speed Prototype Systems



ADAS

"New types of LiDAR sensors require more complex algorithms to process more points and extract features"

Industrial Automation

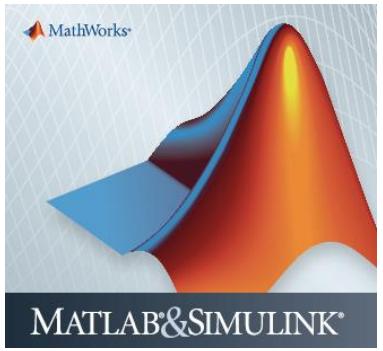
"The controller required numerous input and output channels and had to run at frequencies higher than 1 MHz"

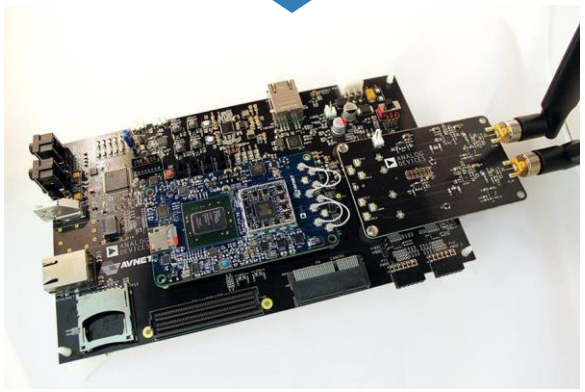
High-Bandwidth Communication

"We needed to validate system architecture for a 1 GHz bandwidth multi-user communication system"

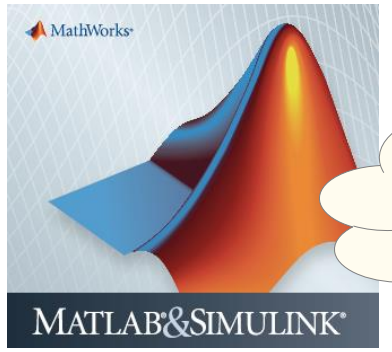
From Algorithm to Prototype

Ideal:





Reality:

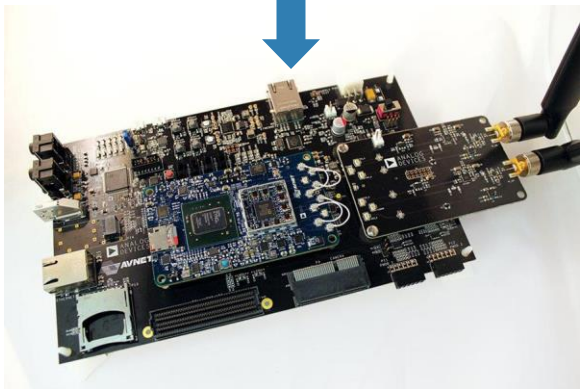


"We don't know much about FPGA design and our hardware team is busy"

SPECIFICATIONS & MEETINGS

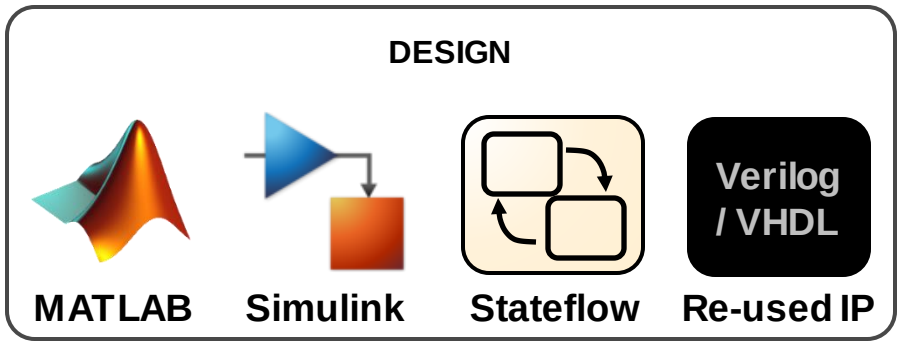
HARDWARE TEAM

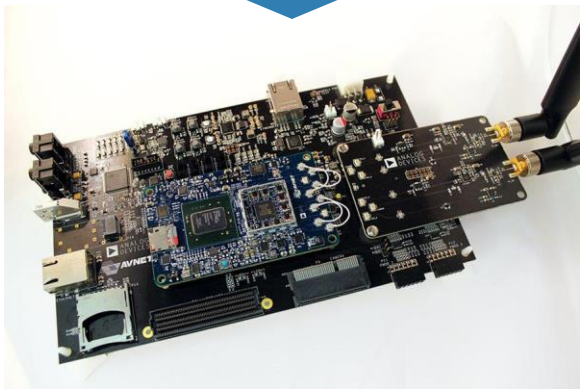
- Hardware Architecture
- Fixed-Point Quantization
- Manual HDL Coding
- FPGA implementation



HDL Coder

Connect algorithm and system design to FPGA prototype hardware



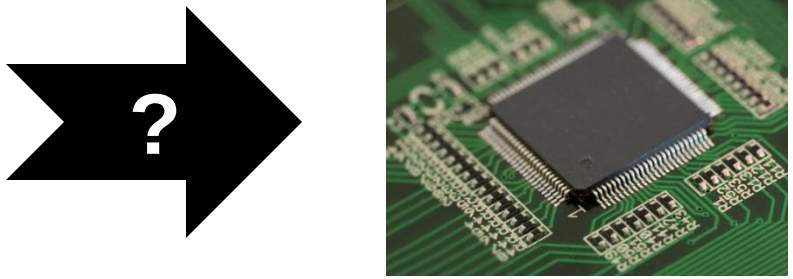


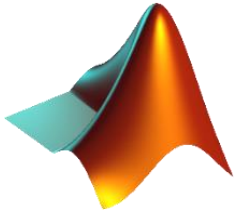
- Take your algorithm to deployment in one environment
  - Add hardware architecture to algorithm and verify
  - Automatically convert to fixed-point or use native floating point
  - Generate code, run implementation, deploy to FPGA board
- Get started with hardware-ready blocks and examples
  - Over 250 Simulink blocks
  - Customize with MATLAB Function blocks and Stateflow charts
  - Example designs for signal processing, wireless comms, motor control, image/video processing
- Prototype connected to MATLAB/Simulink or standalone
- Easily re-target for FPGA or ASIC production
  - Optimized, readable, traceable, and rule-compliant Verilog/VHDL

How to Go from MATLAB Algorithm to HDL Implementation?

```
TxSignal = 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))]);
RxSignal = RxSignal/scale1;
```

```
%% 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));
```



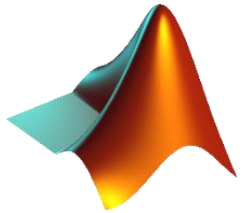


MATLAB

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

Simulink

- ✓ Timing
- ✓ Evaluate architectures
- ✓ Data type analysis

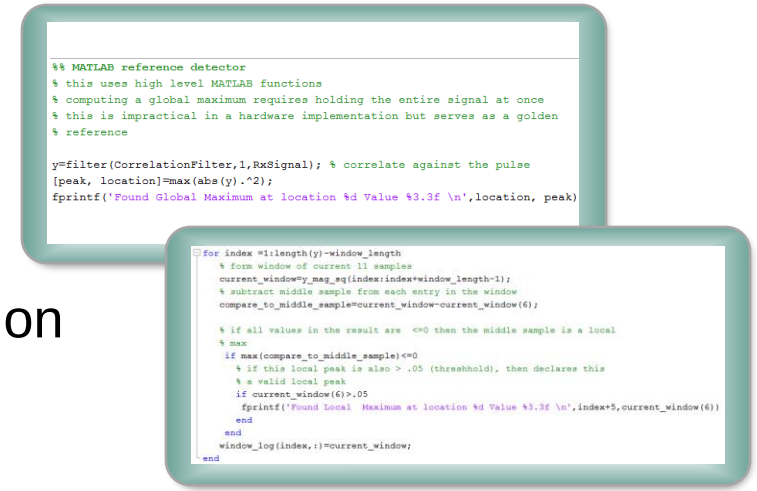


MATLAB

Simulink

Golden Reference

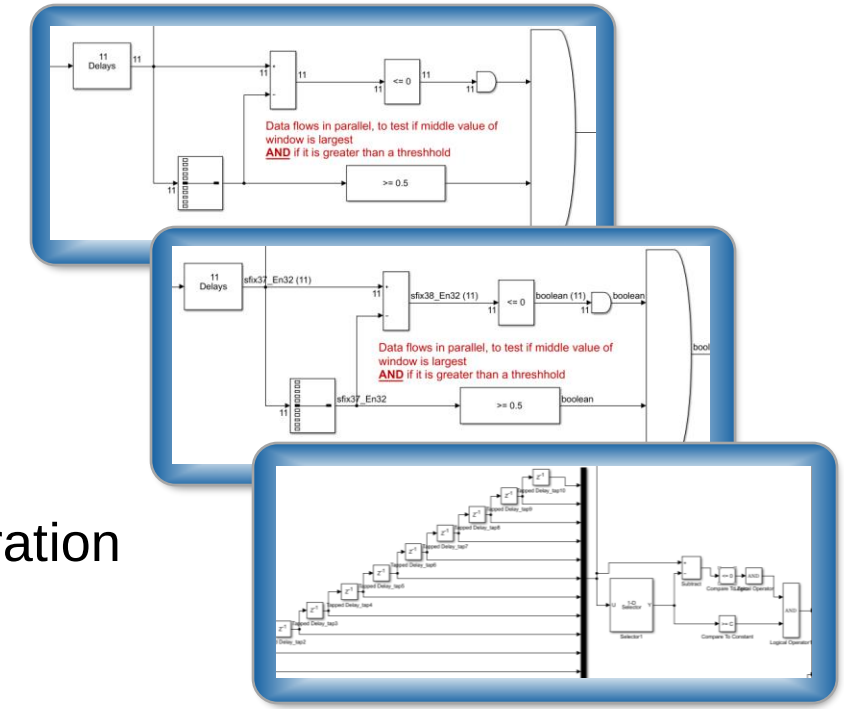
Implementation Algorithm



Hardware Architecture

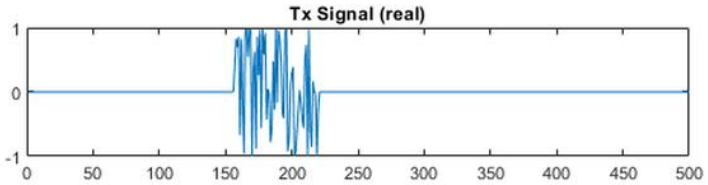
Fixed-point Implementation

HDL Code Generation and Optimization

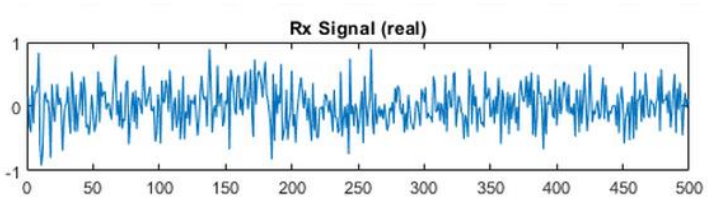


Example: Pulse Detector

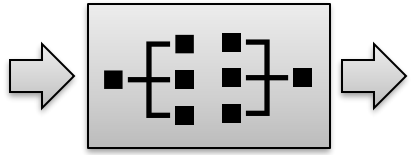
Send

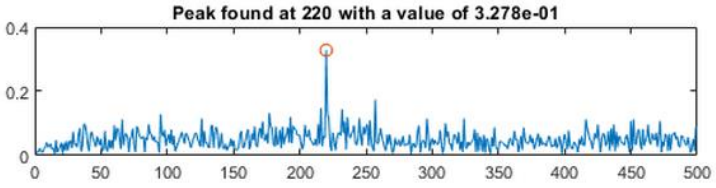


Receive

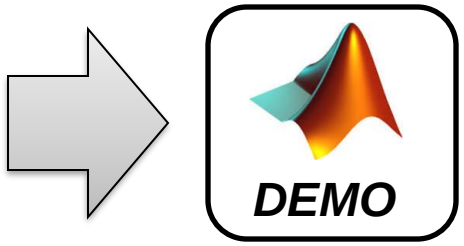


Detect





1. Frame-Based Algorithm to Sample-Based Model



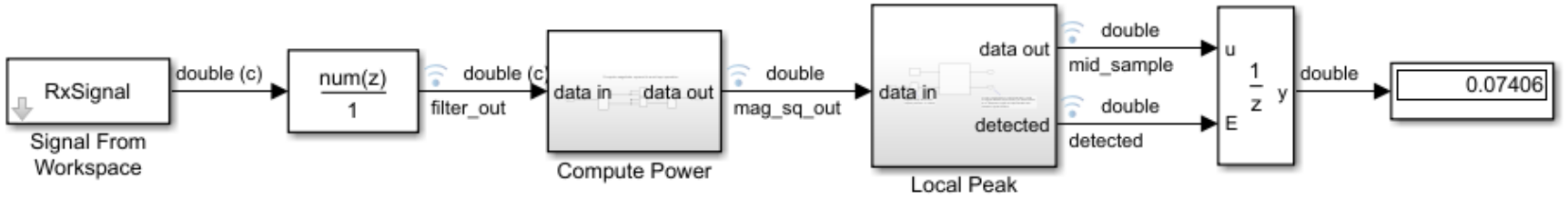
Vector MATLAB reference

MATLAB golden reference

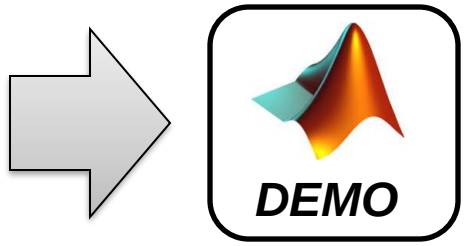
```
% Create matched filter coefficients
CorrFilter = conj(flip(pulse))/PulseLen;

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

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

Verify ✓

Streaming Simulink model



2. Hardware Micro Architecture

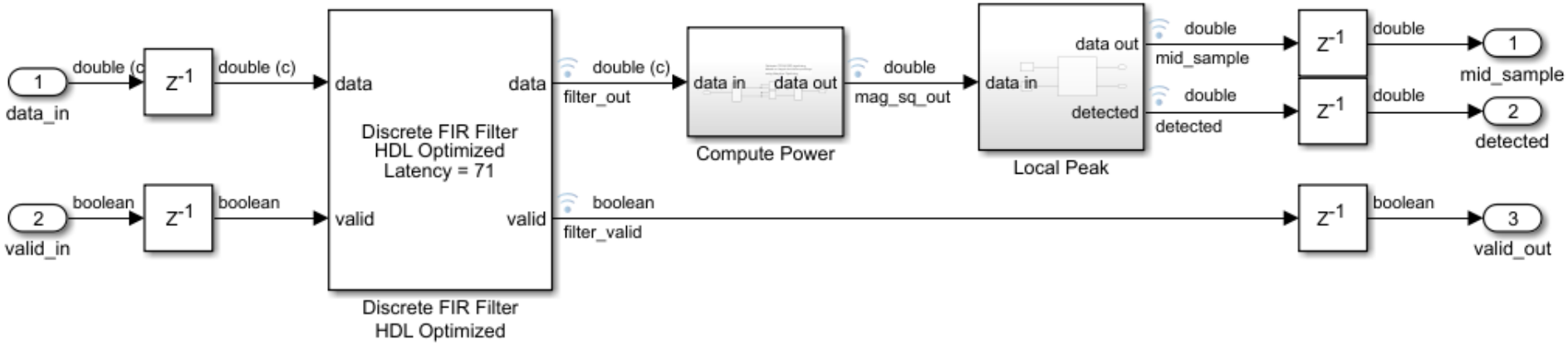


- Prepare the model for HDL code generation
- Use hardware-efficient blocks & architectures
- Add control signal and pipelining
- Verify architecture & control logic without quantization obscuring design problems!

MATLAB golden reference

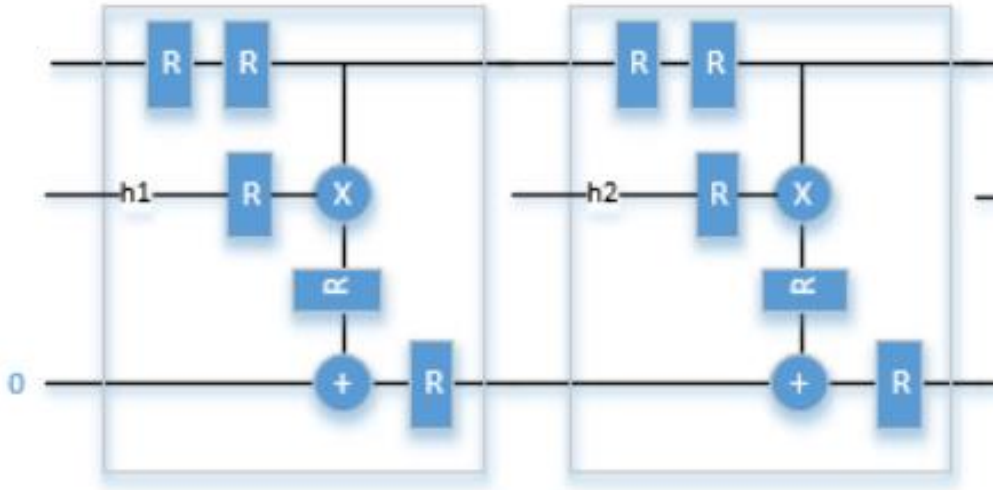
```
% Create matched filter coefficients
CorrFilter = conj(flip(pulse))/PulseLen;

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

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

Verify



Mapping FIR Filter to FPGA DSP blocks

Systolic FIR taps



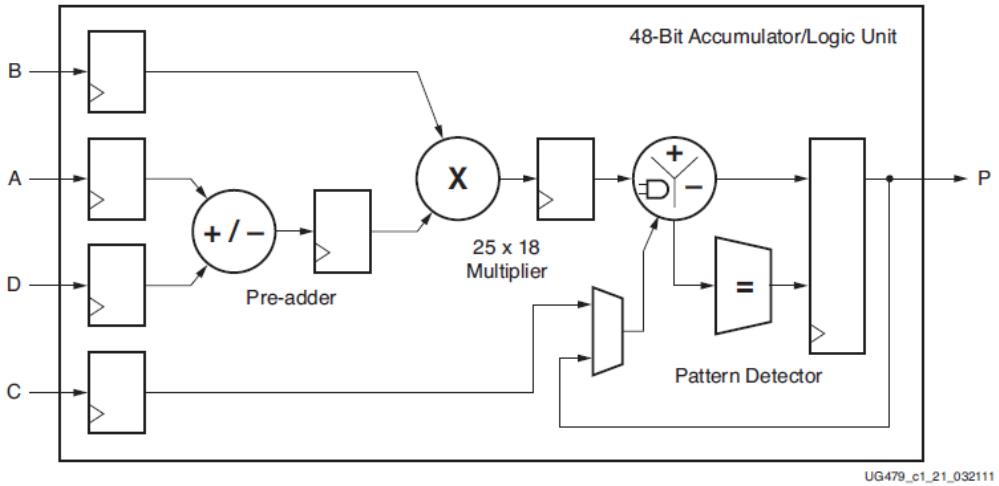
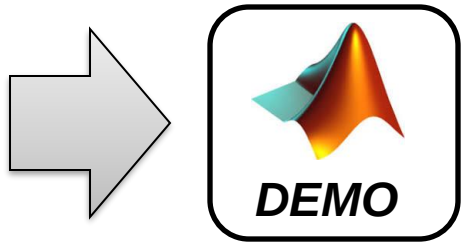


Figure 1-1: **Basic DSP48E1 Slice Functionality**

3. Fixed-Point Implementation and Analysis

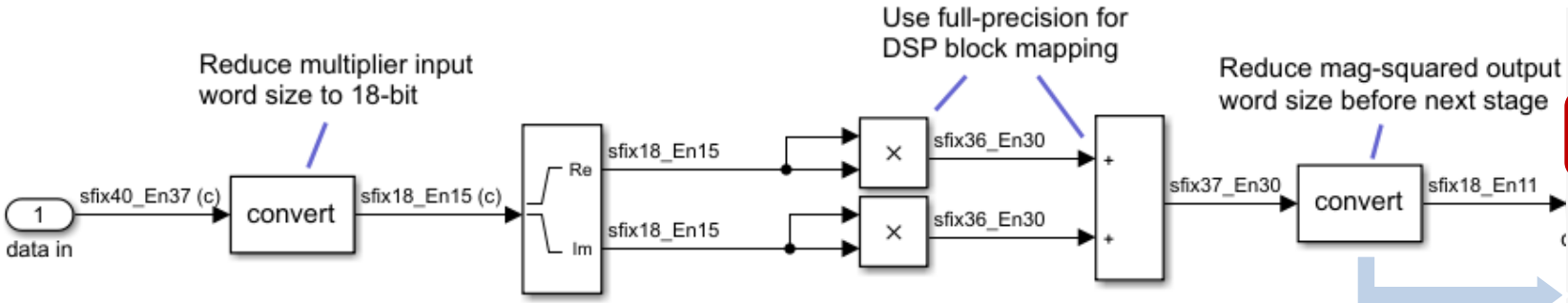


- Convert model to fixed-point data types
  - Choose appropriate word sizes (e.g. 18x18 multiply)
  - Automatic full-precision arithmetic
  - Manage bit growth
- Analyze range & precision using Fixed-Point Tool
  - Automatic min/max & overflow/underflow logging
  - Maximize precision using simulation data

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));
```

Verify



Property	Specified Data Type
DataType	fixdt(1,18,11)
Minimum	-64
Maximum	63.99951171875
Precision	0.00048828125

Range Information

Property	Minimum	Maximum
Simulation	0	0.0801240143...

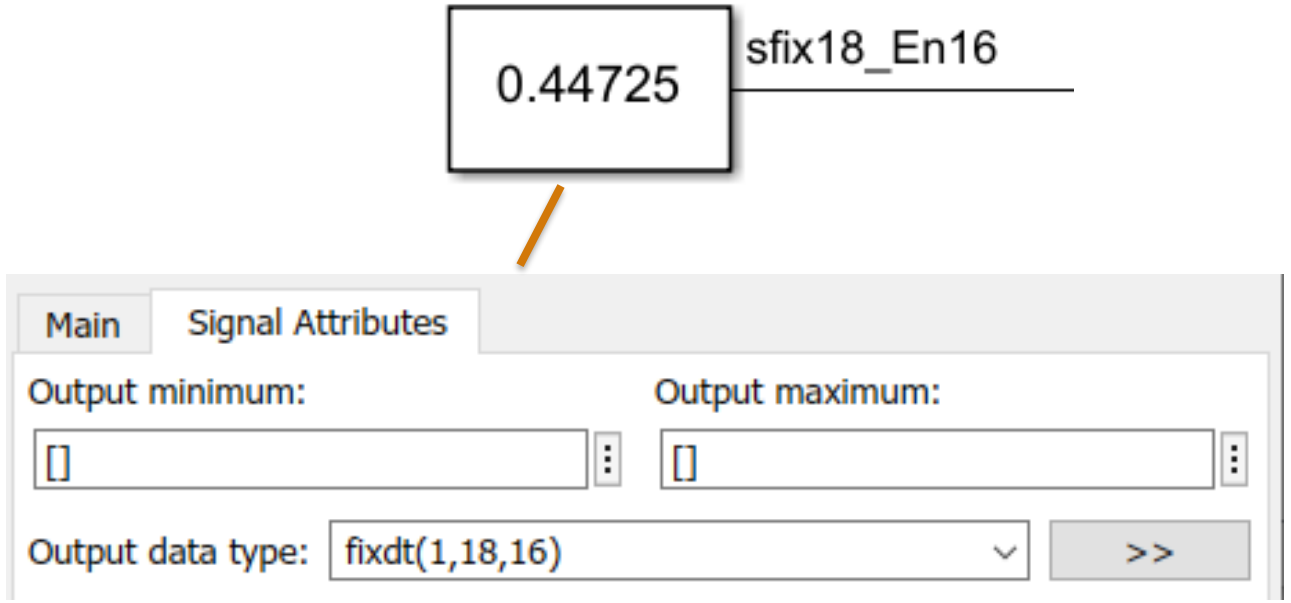
Fixed-Point Representation

```
>> x = fi(0.44725,1,18,16)

      DataTypeMode: Fixed-point: binary point scaling
        Signedness: Signed
        WordLength: 18
    FractionLength: 16
```

```
>> x.range

  [-2.0000    2.0000)
```

```
>> x.eps (resolution)

   1.5259e-05
```

```
>> x.bin

    '0001110010011111111'
```



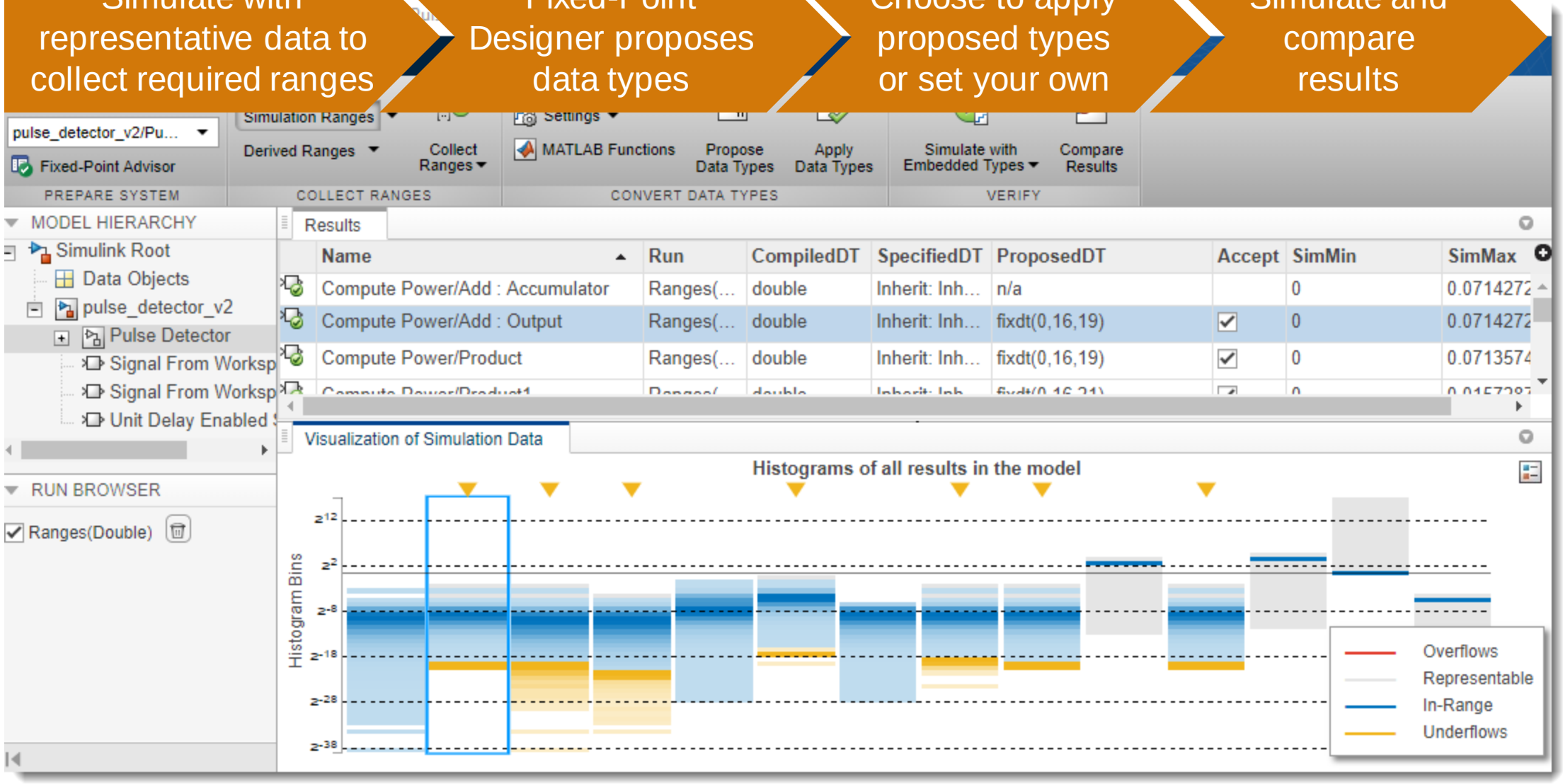
Guided and Automated Fixed-Point Quantization

Simulate with representative data to collect required ranges

Fixed-Point Designer proposes data types

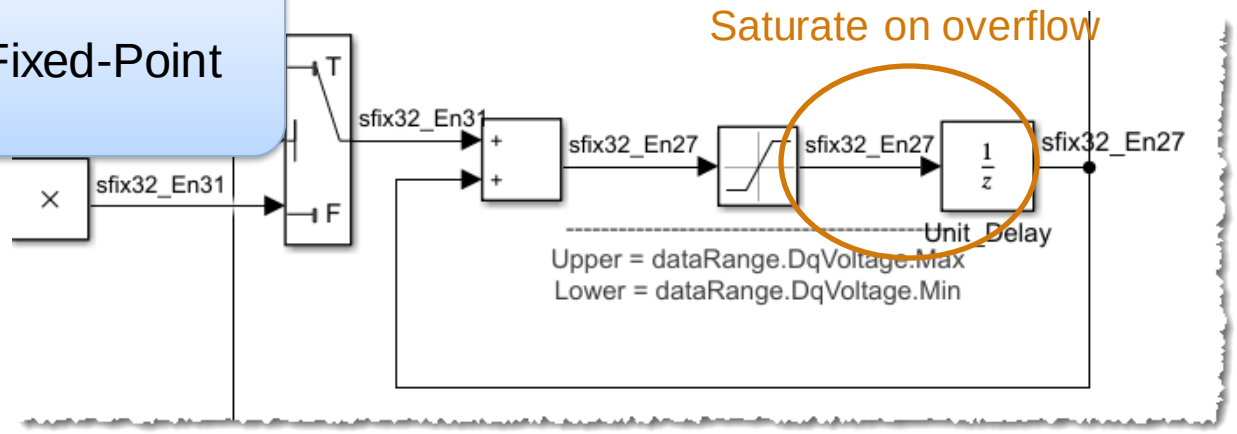
Choose to apply proposed types or set your own

Simulate and compare results

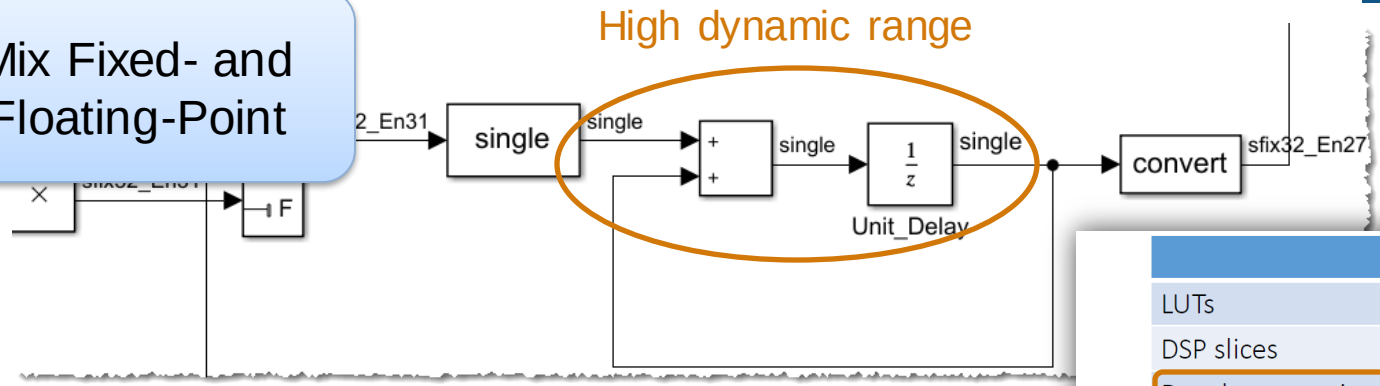


Native Floating Point Support

Fixed-Point



Mix Fixed- and Floating-Point



embeddedaward2017

HDL Coder Native Floating Point

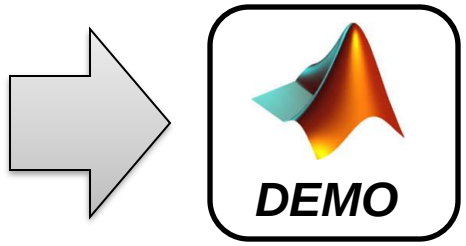
- Extensive math and trigonometric operator support
- Optimal implementations without sacrificing numerical accuracy
- Mix floating- and fixed-point operations
- Generate target-independent HDL

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

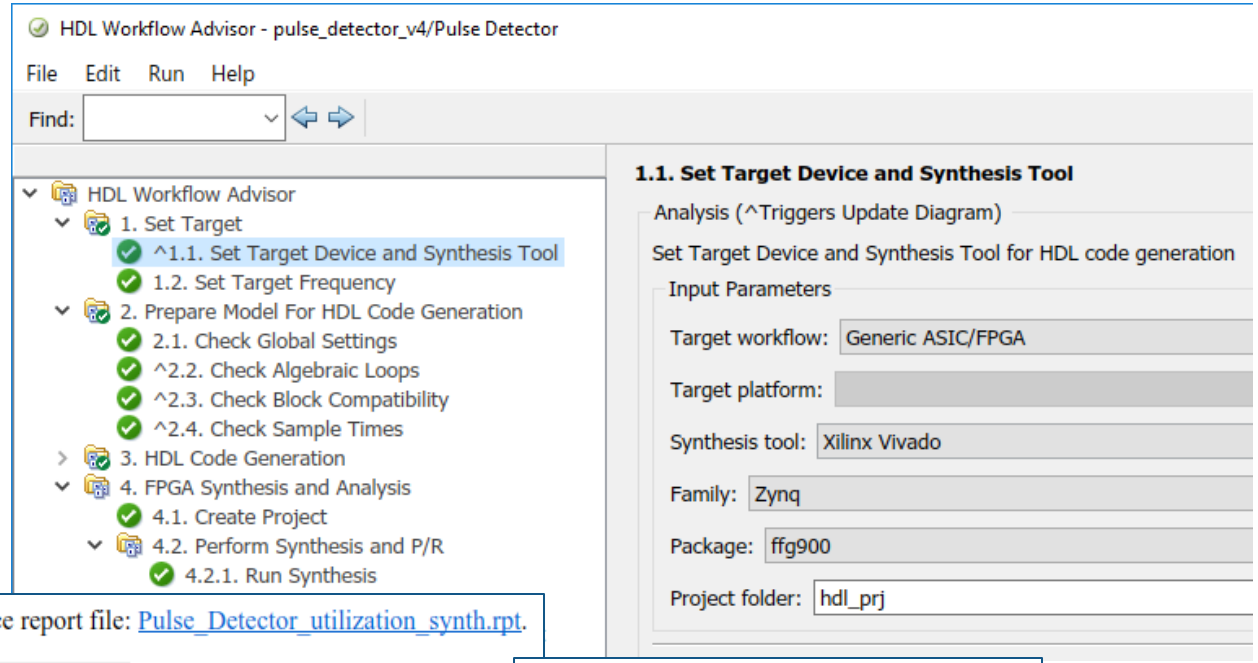
~2x more resources
~5x less development effort

DEMCON

4. Check, Generate and Synthesize HDL



- Check model for HDL compatibility
- Generate HDL code and design summary
- Trace between HDL code and model
- Run synthesis and review results



Parsed resource report file: Pulse_Detector_utilization_synth.rpt.

Resource summary

Resource	Usage
Slice LUTs	5293
Slice Registers	5310
DSPs	258
Block RAM Tile	0

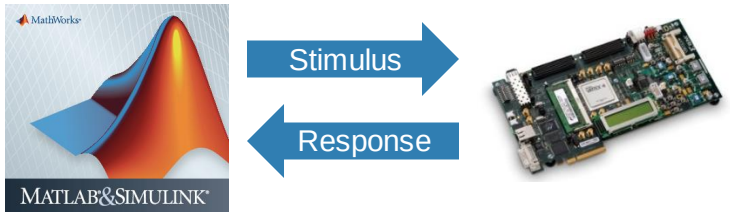
Parsed timing report file: timing_post_map.rpt.

Timing summary

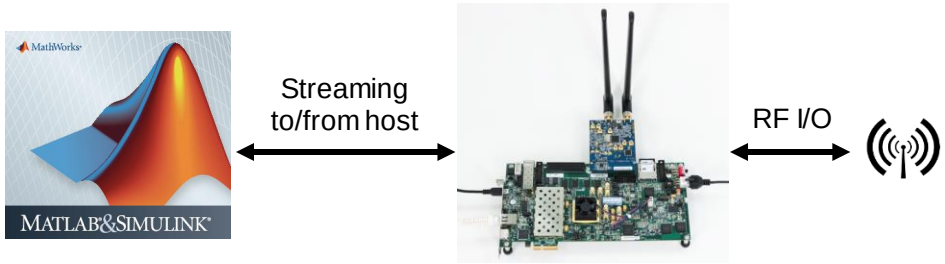
	Value (ns)
Requirement	5
Data Path Delay	3.051
Slack	0.892

FPGA / SoC Prototyping and Debugging

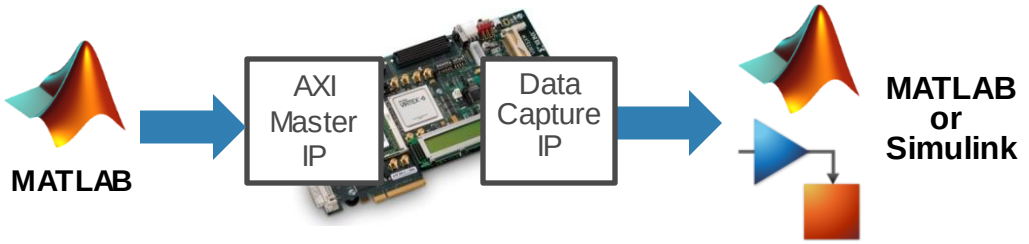
- Hardware cosimulation using FPGA-in-the-Loop



- Deploying designs to prototyping hardware
  - Targeting SDR & Vision evaluation platforms
  - Integrating custom hardware platforms

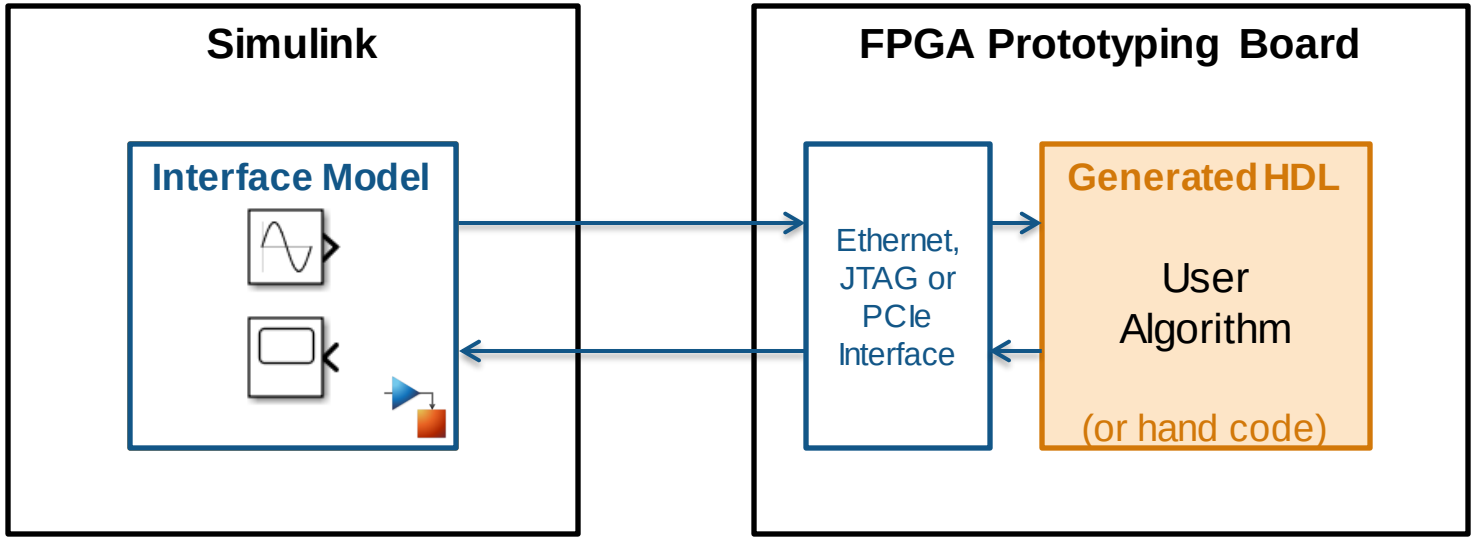


- Debugging FPGA designs in MATLAB
  - Capturing data from free-running FPGA design
  - Interacting with registers & DDR memories



Hardware Cosimulation Using FPGA-in-the-Loop

- An automated workflow to verify design in target hardware
- User algorithm I/O connected to Simulink via Ethernet, PCIe or JTAG
- FPGA is synchronized to Simulink model execution (non real-time)
- Accelerate compute-intensive algorithms using vector mode



HDL Verifier Support Package for Xilinx, Intel & Microsemi Boards

Device Family	Board	Ethernet (FIL)	JTAG (FIL, AXI Master, Data Capture)	PCI Express (FIL)[a]
Xilinx Artix®-7	Digilent Nexys™4 Artix-7	x	x	
	Digilent Arty Board		x	
Xilinx Kintex-7	Kintex-7 KC705	x	x	x
Xilinx Kintex UltraScale™	Kintex UltraScale FPGA KCU105 Evaluation Kit	x	x	
Xilinx Kintex UltraScale+	Kintex UltraScale+ FPGA KCU116 Evaluation Kit		x	
Xilinx Spartan-6	Spartan-6 SP605	x		
	Spartan-6 SP601	x		
	XUP Atlys Spartan-6	x		
Xilinx Spartan-7	Digilent Arty S7-25	x	x	
Xilinx Virtex UltraScale	Virtex UltraScale FPGA VCU108 Evaluation Kit	x	x	
Xilinx Virtex UltraScale+	Virtex UltraScale+ FPGA VCU118 Evaluation Kit		x	x
Xilinx Virtex-7	Virtex-7 VC707	x	x	x
	Virtex-7 VC709		x	x
Xilinx Virtex-6	Virtex-6 ML605	x		
Xilinx Virtex-5	Virtex ML505	x		
	Virtex ML506	x		
	Virtex ML507	x		
	Virtex XUPV5–LX110T	x		
XilinxVirtex-4	Virtex ML401	x		
	Virtex ML402	x		
	Virtex ML403	x		
Xilinx Zynq	Zynq-7000 ZC702		x	
	Zynq-7000 ZC706		x	
	ZedBoard		x	
	ZYBO™ Zynq-7000 Development Board		x	
	PicoZed™ SDR Development Kit		x	
	MiniZed™		x	
Xilinx Zynq UltraScale+	Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit		x	
	Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit		x	
	Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit		x	
	Zynq UltraScale+ RFSoC ZCU111 Evaluation Kit		x	

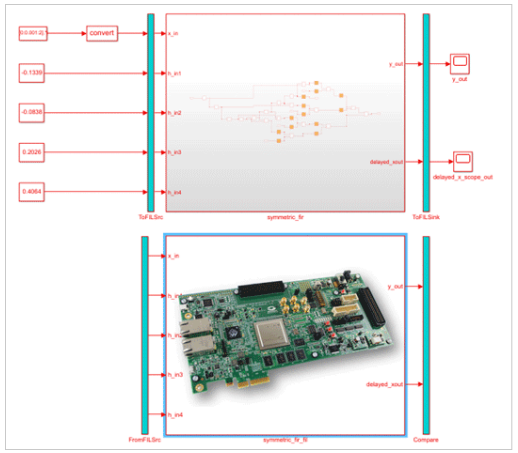
Intel Arria II	Arria II GX FPGA Development Kit	x	x	
Intel Arria V	Arria V SoC Development Kit		x	
	Arria V Starter Kit	x	x	
Intel Arria 10	Arria 10 SoC Development Kit	x	x	
	Arria 10 GX	x	x	x
Intel Cyclone IV	Cyclone IV GX FPGA Development Kit	x	x	
	DE2-115 Development and Education Board	x	x	
	BeMicro SDK	x	x	
Intel Cyclone III	Cyclone III FPGA Starter Kit		x	
	Cyclone III FPGA Development Kit	x	x	
	Altera Nios II Embedded Evaluation Kit, Cyclone III Edition	x	x	
Intel Cyclone V	Cyclone V GX FPGA Development Kit	x	x	
	Cyclone V SoC Development Kit		x	
	Cyclone V GT Development Kit	x	x	x
	Terasic Atlas-SoC Kit / DE0-Nano SoC Kit		x	
	Arrow SoCKit Development Kit		x	
Intel Cyclone 10 LP	Altera Cyclone 10 LP Evaluation Kit		x	
Intel Cyclone 10 GX	Altera Cyclone 10 GX FPGA Evaluation Kit		x	
Intel MAX 10	Arrow MAX 10 DECA	x	x	
Intel Stratix® IV	Stratix IV GX FPGA Development Kit	x	x	
Intel Stratix V	DSP Development Kit, Stratix V Edition	x	x	x
Microsemi SmartFusion®2	Microsemi SmartFusion2 SoC FPGA Advanced Development Kit	x		
Microsemi Polarfire®	Microsemi Polarfire Evaluation Kit	x		
Microsemi RTG4®	RTG4-DEV-KIT	x		

Requires:

- HDL Verifier

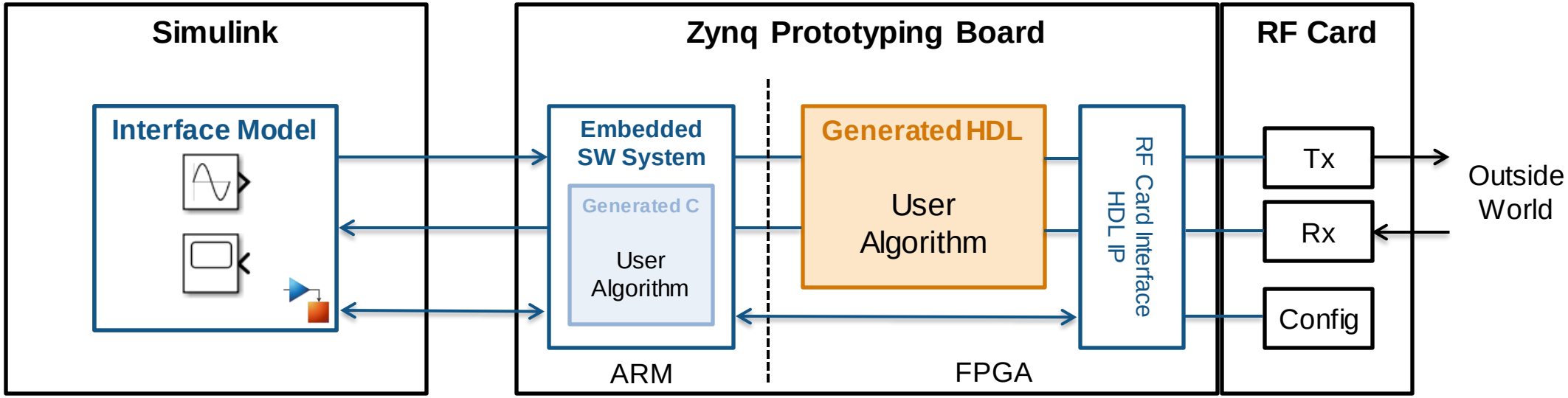
Optional:

- HDL Coder



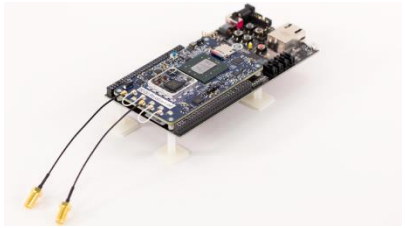
Deploying Designs to SDR or Vision Evaluation Platforms

- Deploy algorithm to Zynq-based radio / vision platforms for real-time proc.
- Integrate with HDL reference design & embedded system automatically
- Deploy low-bandwidth SW algorithm to ARM
- Control RF / video hardware, tune parameters & visualize output in Simulink



Zynq-Based Radio and Vision Hardware Support Packages

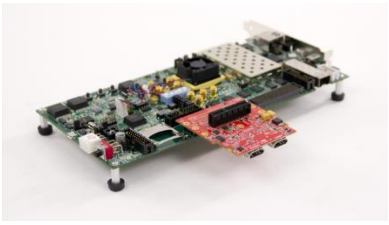
- Zynq Radio HSP and USRP Embedded HSP
  - ADI RF SOM
  - ZCU102 / ZC706 / ZedBoard + ADI FMCOMMS2, 3 or 4
  - ZC706 + ADI FMCOMMS5
  - ZCU111 RFSoC*
  - USRP E310



Requires:
- Comms Toolbox

HW targeting requires:
- HDL Coder
- Embedded Coder

- Zynq Vision HSP
  - ZC706 / ZC702 / ZedBoard + FMC-HDMI-CAM
  - Avnet PizoZed Embedded Vision Kit

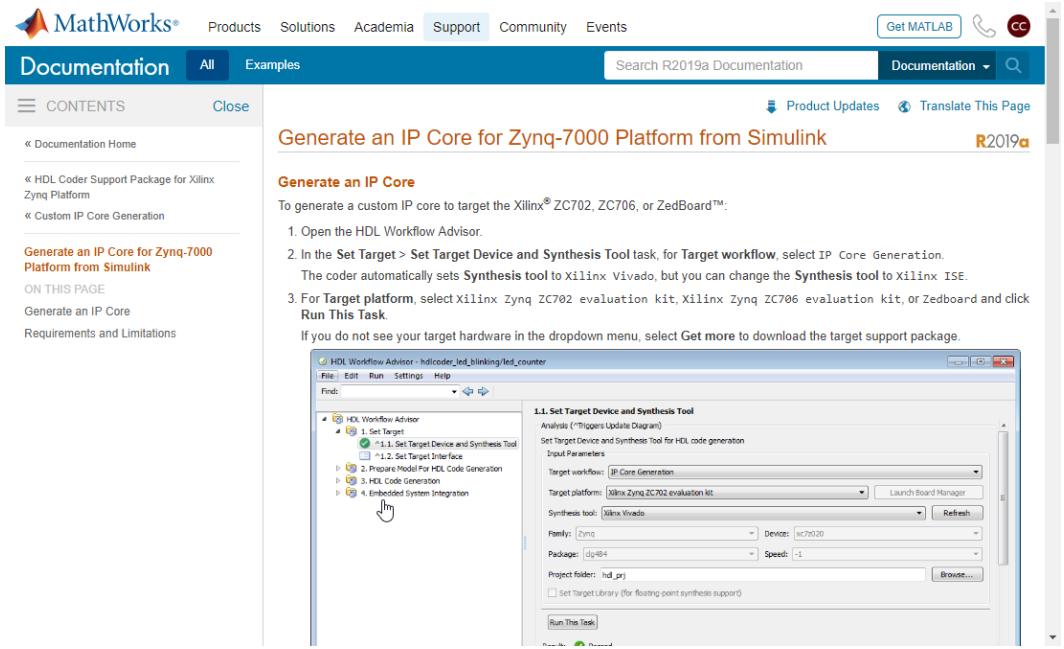


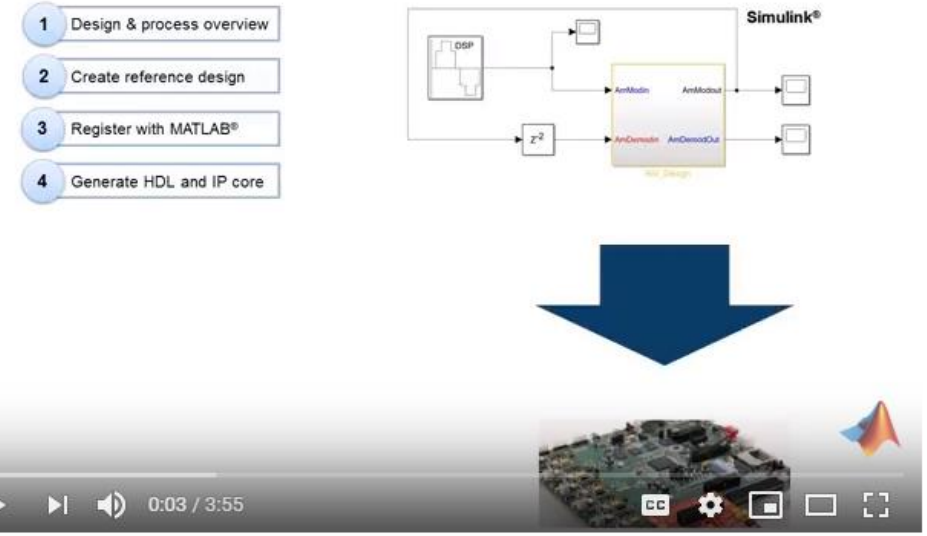
Requires:
- Computer Vision Toolbox

HW target requires:
- HDL Coder
- Embedded Coder

Deploying Designs to Custom Hardware Platforms

- Generate IP Core with AXI interfaces
- Register custom HDL reference design with MATLAB using provided API





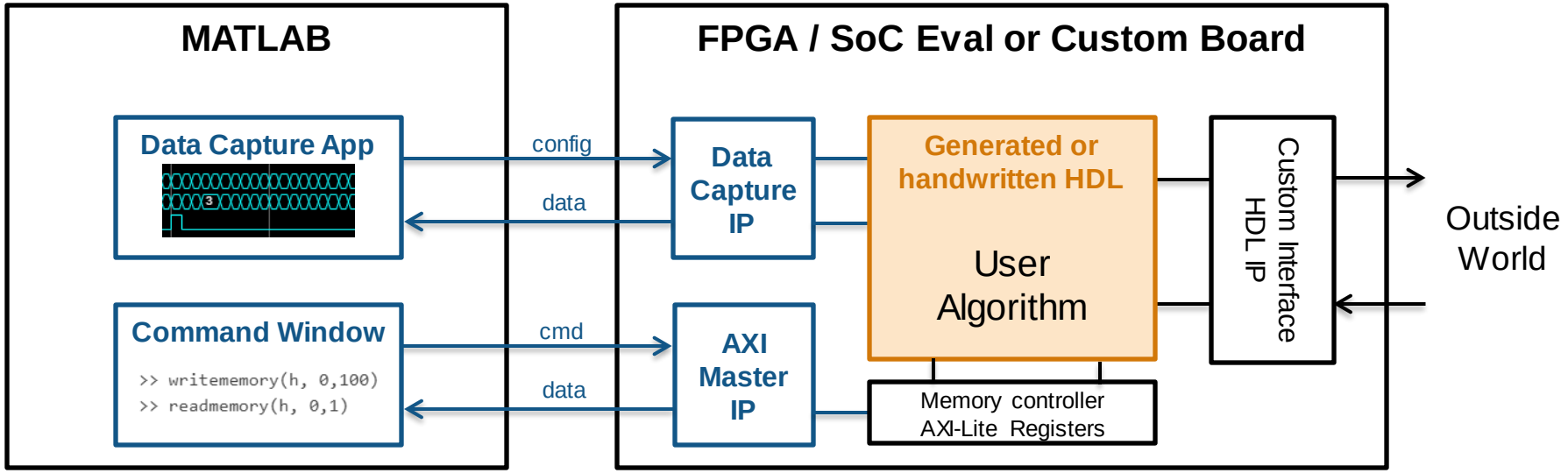
Custom Reference Design, Part 1: Introduction to the Workflow

https://www.mathworks.com/help/hdlcoder/custom-reference-design.html

https://www.youtube.com/watch?v=ZD8Tj9A6MHg

Debugging FPGA designs in MATLAB

- Access to FPGA signals & memories in real-time operations
- Integrate Data Capture IP to your design; trigger & capture buffer of data
- Integrate AXI Master IP with your FPGA memories; read/write parameters and DDR memory with simple MATLAB commands

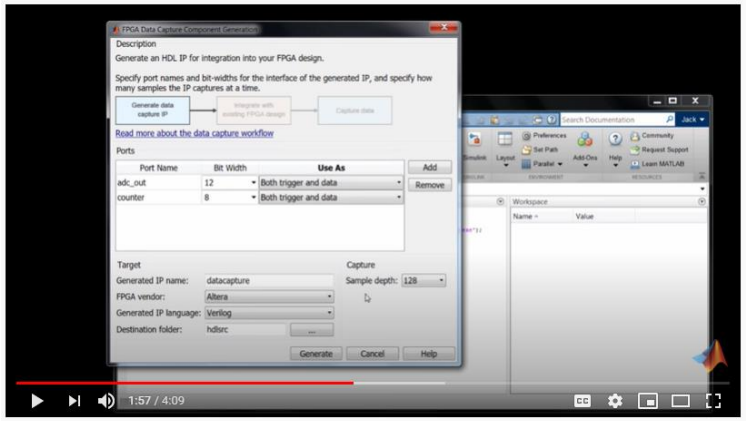


HDL Verifier Support Package for Xilinx, Intel & Microsemi Boards

- Data Capture & AXI Master over JTAG supports:
  - Same hardware as FPGA-in-the-Loop
- AXI Master over Ethernet supports:
  - Xilinx ZC706, ZedBoard, KC705, Intel Arrow MAX10 DECA
- AXI Master over PCIe supports:
  - Xilinx KCU116, Intel Arria 10 GX

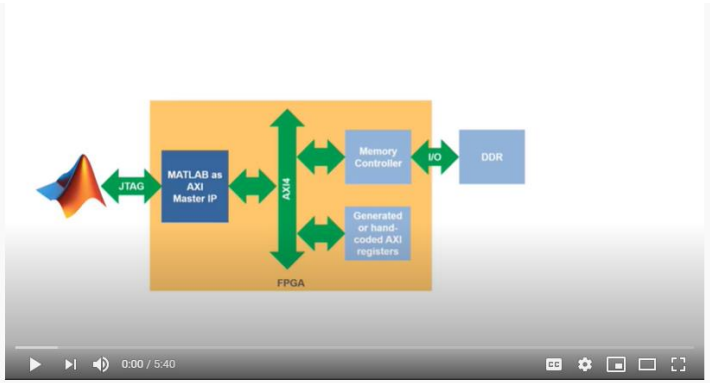
Requires:
- HDL Verifier

Optional:
- HDL Coder



HDL Verifier: FPGA Data Capture

https://www.youtube.com/watch?v=eHebToNdLpU

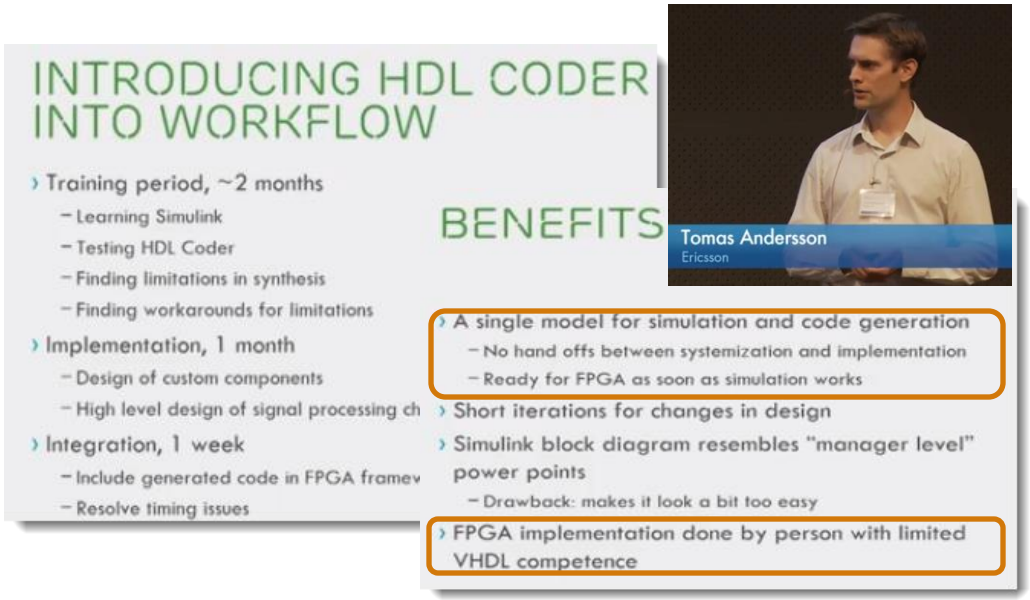


MATLAB as AXI Master with Xilinx FPGA and Zynq SoC Boards

https://www.youtube.com/watch?v=U-RaLUFkodI

Resources to Get Started and Speed Adoption

- Getting started:
  - MATLAB Onramp
  - Simulink Onramp
  - HDL pulse detector self-guided tutorial and videos



- Proof-of-concept guided evaluations
  - Free support via weekly WebEx meetings using custom sample designs

- Training & consulting services
  - HDL code generation, FPGA signal processing & Zynq programming training courses
  - Consulting service on deep technical coaching, custom design / hardware and more