

MATLAB EXPO 2018

[Sub Track 1-3]

FPGA/ASIC을 타겟으로 한 알고리즘의 효율적인
생성 방법 및 신기능 소개


정승혁 과장

Senior Application Engineer

MathWorks Korea



Outline

- 
- When FPGA, ASIC, or System-on-Chip (SoC) hardware is needed
 - Hardware implementation considerations
 - Workflow from system/algorithm to FPGA/ASIC hardware
 - Demonstration: Vision processing algorithm deployed to FPGA
 - Conclusion

Why Are Our Customers Deploying to FPGA/ASIC Hardware?



Speed

“Real-time image processing for an aircraft head’s up display”

“Evaluate the algorithm in field testing to analyze system performance”

“Optimal performance @ Piezo resonance frequency”

Power

“11 year device with a 1 A*hr battery”

Latency

“Be able to stop the robot with millimeter accuracy in less than 0.5 seconds without causing damage to the robot”

“Audio transducer prototypes must run in real time with low latencies”

“Motor control latency < 1us”

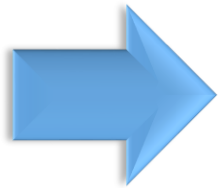
We need to get to market quickly, but we have no experience designing FPGAs!

Modern Applications Often Require Custom Hardware

ADAS Application Example



1M+ pixels per frame

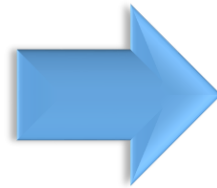


High-speed, well-defined

- Repetitive processing
- Large amount of data

FPGA Hardware

Few coords



Complex, more flexible

- Small data calculations
- Executive control

Embedded Software

Outline

- When FPGA, ASIC, or System-on-Chip (SoC) hardware is needed
- » ▪ Hardware implementation considerations
- Workflow from system/algorithm to FPGA/ASIC hardware
- Demonstration: Vision processing algorithm deployed to FPGA
- Conclusion

Frame-Based vs Streaming Algorithms

Frame-Based

- Whole frame at a time
- Random access to any pixel via [x,y] coordinate

Streaming

- Sample-by-sample, row-by-row
- Region of interest (ROI) stored in a multi-line buffer

(0,0)

Frame Width

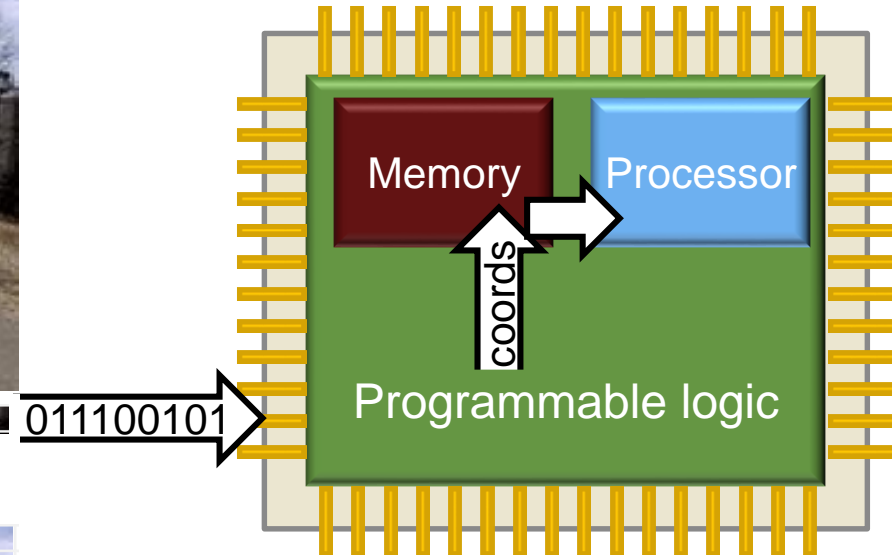
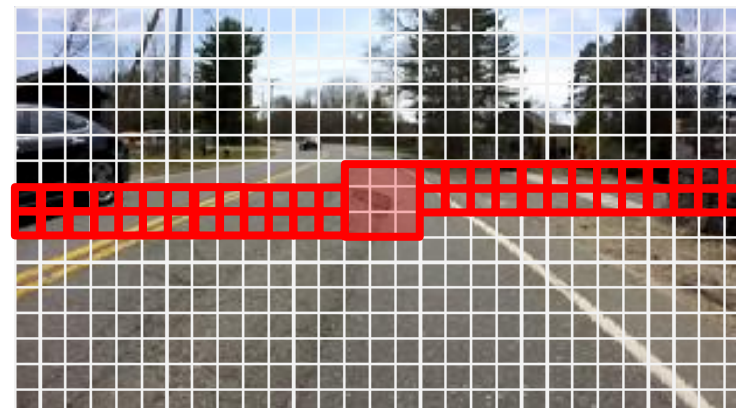
Frame Height



(0,0)

Columns

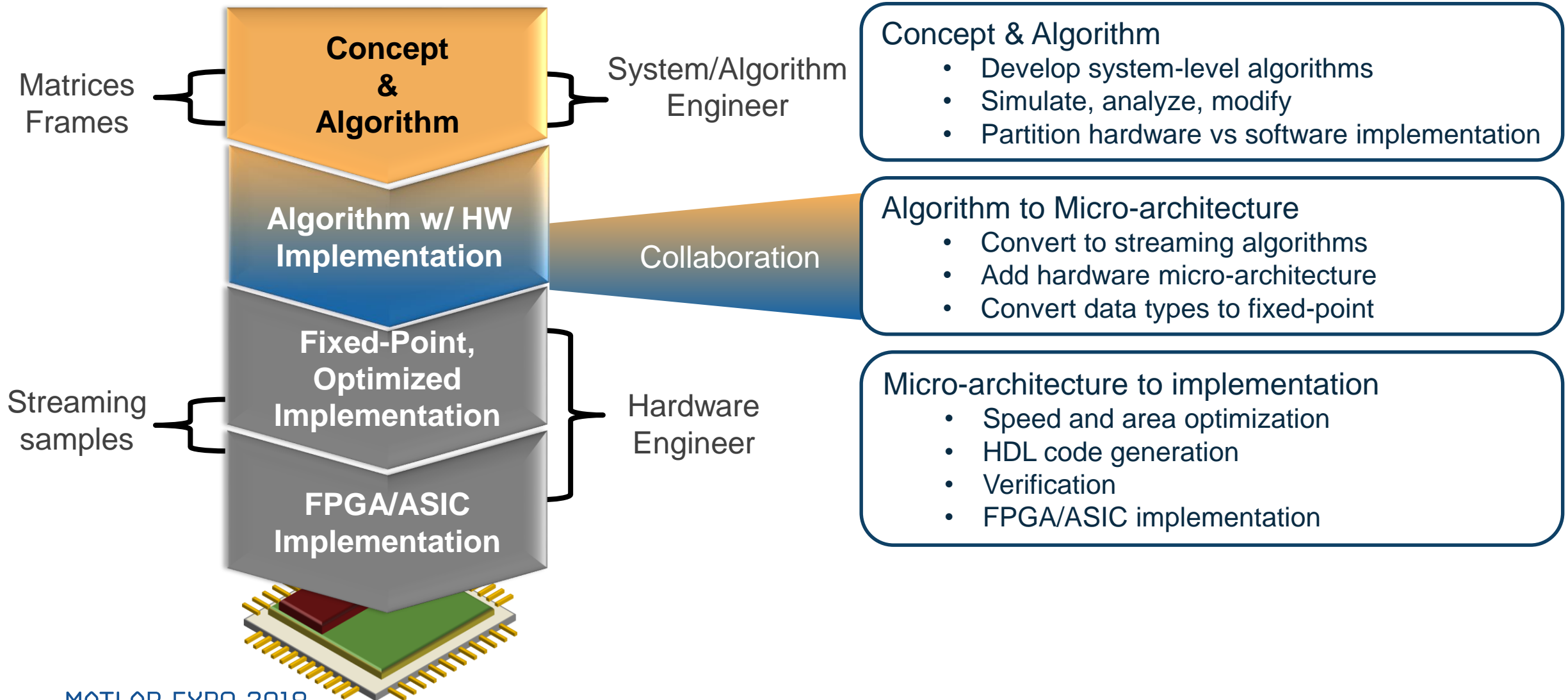
Rows



Hardware

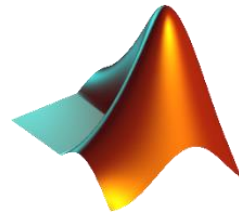
- Bit-by-bit, but parallel computation
- Fixed and finite resources
- Buffers require memory storage
- Communications with software go through dedicated memory

Bridging the Gap from Algorithm to Implementation

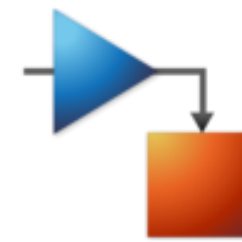


Outline

- When FPGA, ASIC, or System-on-Chip (SoC) hardware is needed
- Hardware implementation considerations
- » ▪ Workflow from system/algorithm to FPGA/ASIC hardware
- Demonstration: Vision processing algorithm deployed to FPGA
- Conclusion



MATLAB



Simulink

Algorithm
(Golden Reference)

Algorithm w/ HW
Implementation

Fixed-Point,
Optimized
Implementation

FPGA/ASIC
Implementation

```

%% Frame pre-processing
% Convert to intensity
frmGray = rgb2gray(frmIn);

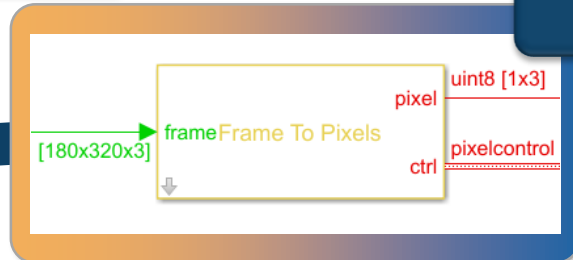
% Bilateral filter
frmBiFilt = imbilatfilt(frmGray, 'NeighborhoodSize', 9);

% Edge detection
frmEdge = edge(frmBiFilt, 'Sobel', .05);

%% Trapezoidal mask

```

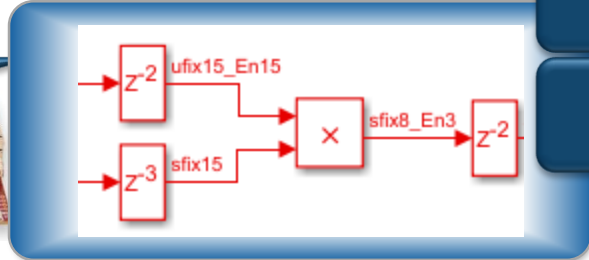
Verify



HDL-ready
IP blocks



HDL Coder
Prototype



Fixed Point
Designer

HDL Coder

HDL Coder
Production

Core
interface

VHDL /
Verilog

Constraints

Outline

- When FPGA, ASIC, or System-on-Chip (SoC) hardware is needed
- Hardware implementation considerations
- Workflow from system/algorithm to FPGA/ASIC hardware
- » ▪ Demonstration: Vision processing algorithm deployed to FPGA
- Conclusion

Demo : Pothole Detection using Traditional Image Processing

- Bilateral filter followed by Sobel edge detection as in 17b
- New trapezoidal mask plus Morphological Closing
- New Centroid 31x31 and New Maximum Area Detection
- New Graphic Marker and Text (character) overlay

R2018a



Algorithm Overview

MATLAB R2018a

HOME PLOTS APPS EDITOR PUBLISH VIEW

FILE NAVIGATE EDIT BREAKPOINTS RUN

Editor - C:\MATLAB\MLExp2018\PotHoleDetect_alg.m

```
49 - end
50 - end
51 -
52 - % morphological closing
53 - frmClose = imclose(frmMasked, [0 1 1 1 0;1 1 1 1 1;1 1 1 1 1;1 1 1 1 1;0 1 1 1 0]);
54 -
55 - %% Detect centroids
56 - % Apply a combination of morphological dilation and image arithmetic
57 - % operations to remove uneven illumination and to emphasize the
58 - % boundaries between the cells.
59 - hBlob = vision.BlobAnalysis( ...
60 -     'MinimumBlobArea', areaThreshold, ...
61 -     'BoundingBoxOutputPort', false, ...
62 -     'OutputDataType', 'single');
63 - % Return Centroids and Area
64 - [Area,Centroids] = hBlob(frmClose);
65 -
66 - %% Overlay markers
67 -
68 - frmOverlay = frmIn*0.7 + frmMasked*0.3;
69 - if isempty(Centroids)
```

Workspace

Command Window

fx >>

PotHoleDetect_alg Ln 59 Col 37

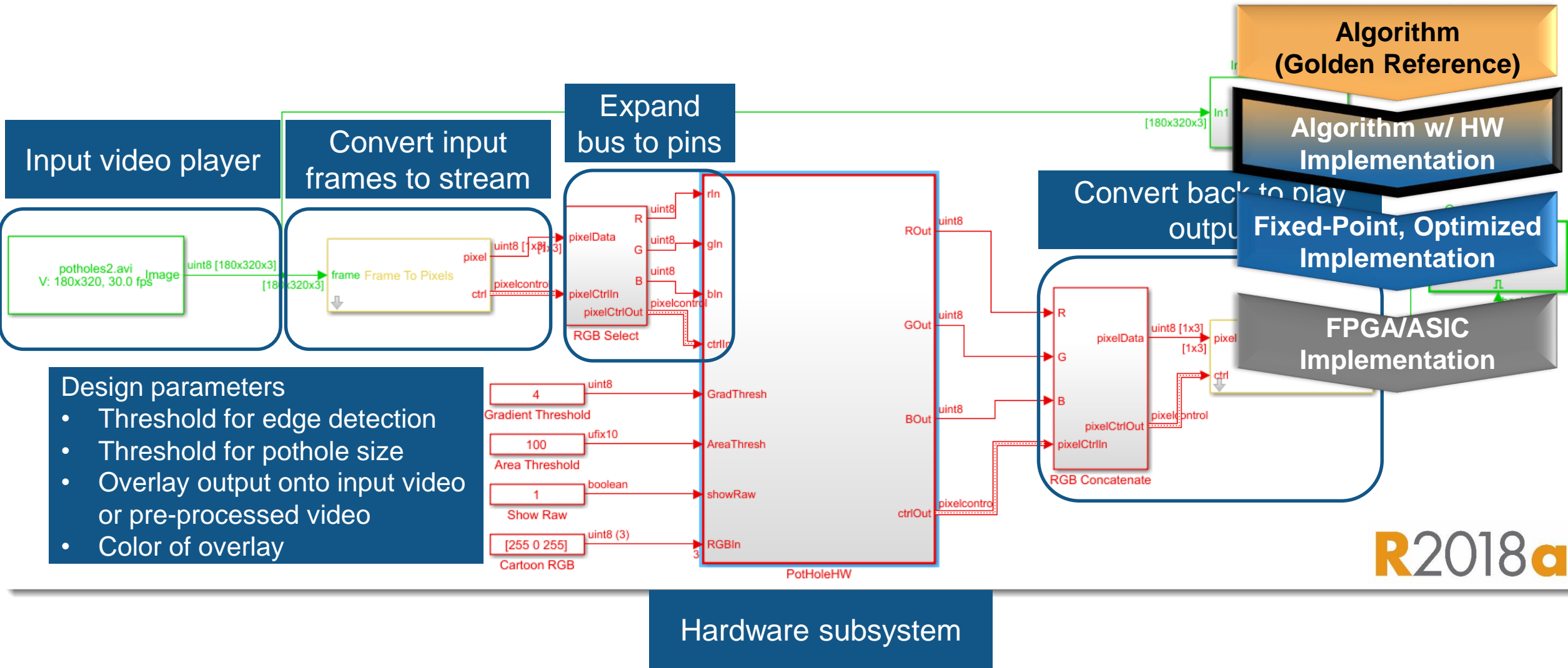
Algorithm
(Golden Reference)

Algorithm w/ HW
Implementation

Fixed-Point, Optimized
Implementation

FPGA/ASIC
Implementation

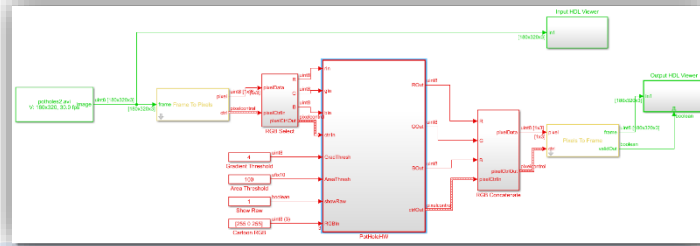
Algorithm with Hardware Implementation: Top-Level



R2018a

Hardware subsystem

Algorithm with Hardware Implementation: Hardware Subsystem



Align timing of centroids with input or pre-processed image

Overlay centroids

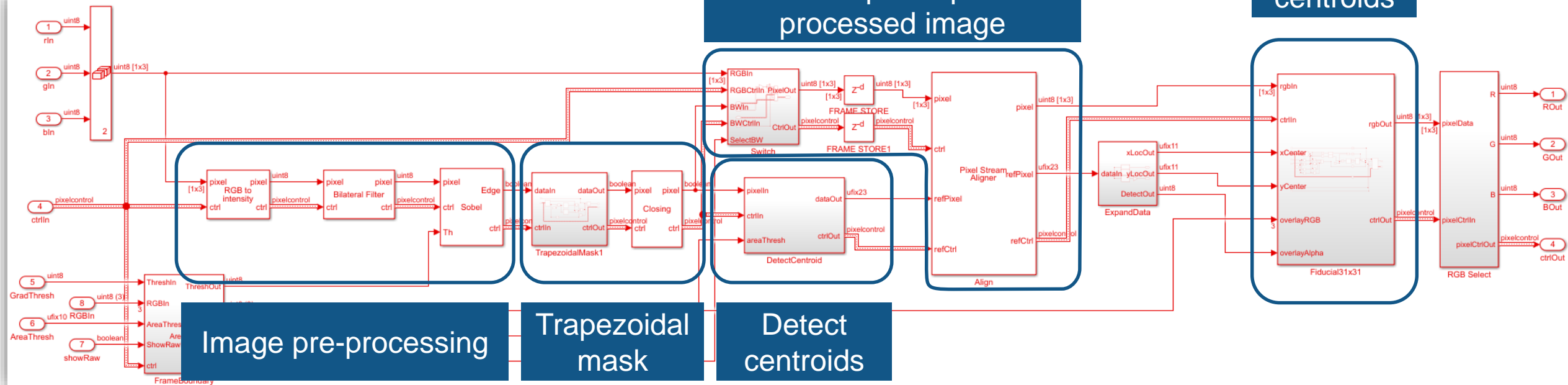


Image pre-processing

Trapezoidal mask

Detect centroids

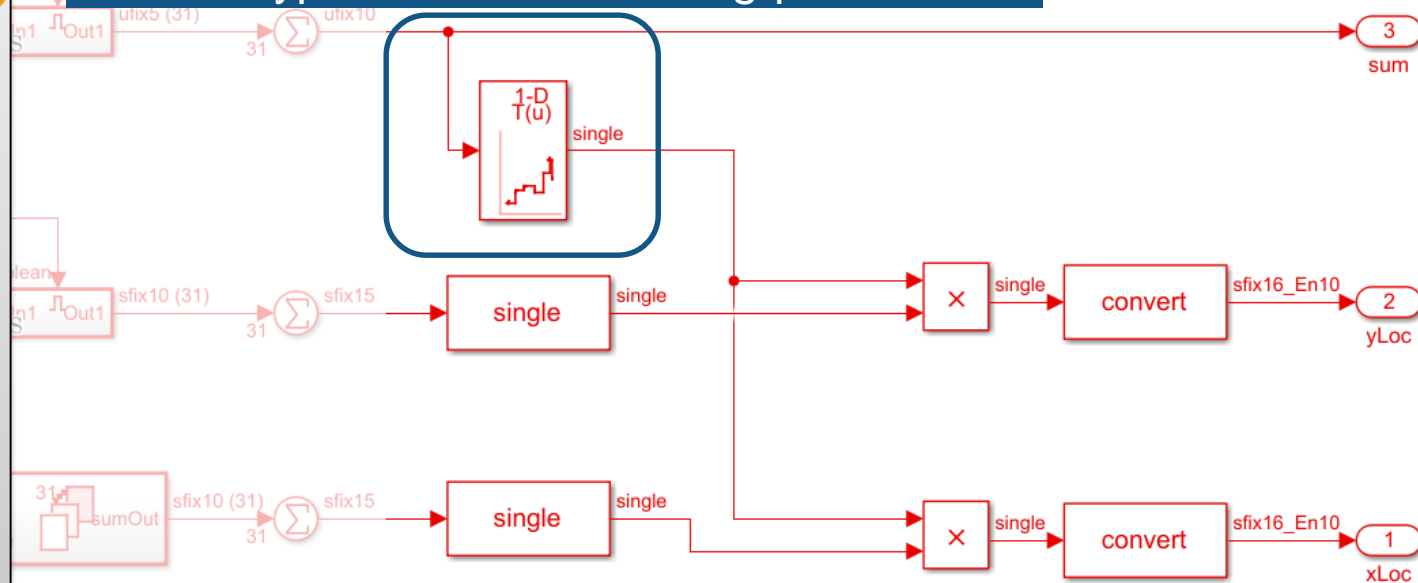
Streaming Math with Native Floating-Point for Prototyping

Centroid Kernel

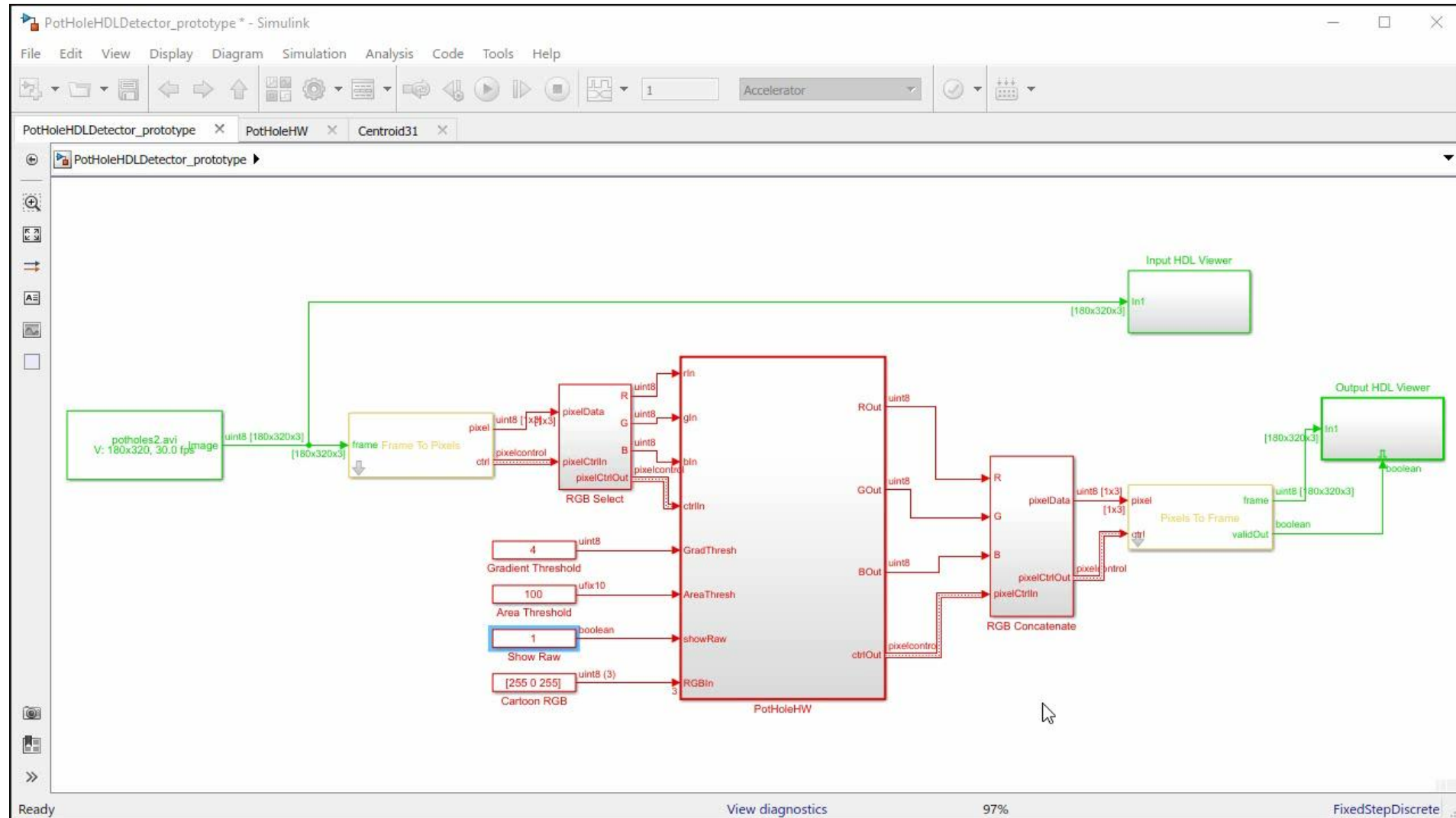
HDL Coder Native Floating Point **R2016b**

- **IEEE-754** Single precision support
- Extensive math and trigonometric operator support
- **Highly optimal** implementations without sacrificing **numerical accuracy**
- **Mix** floating and fixed point operations in the same design
- Generate **target-independent** synthesizable VHDL or Verilog

- ROM stores weights: $1./[1, 1:1023]$
- Don't know level of precision required
- Prototype with native floating-point!



Prototype Target



Fixed-Point, Optimized Implementation: General Approach

1 Know your hardware

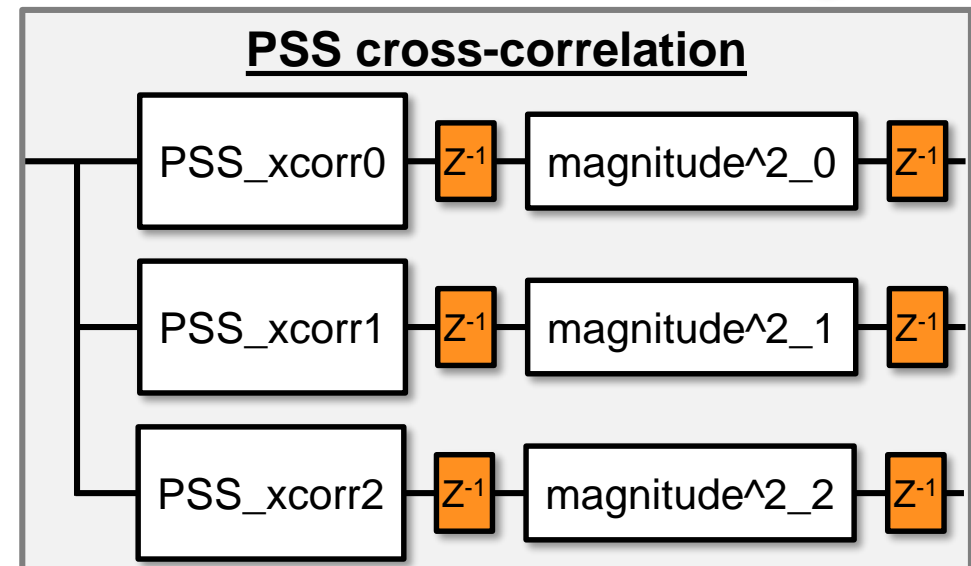
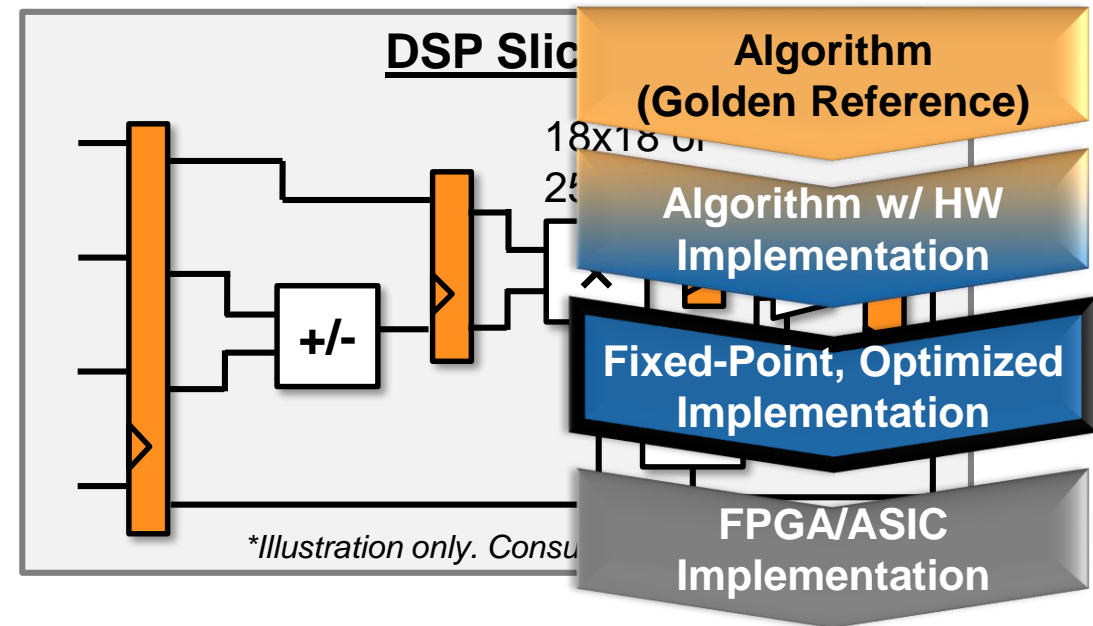
- How much on-chip RAM?
- Typical achievable frequency
- Available I/O
- FPGA: How many DSP slices?

2 Know your performance requirements

- Control system latency
- Comms system throughput
- Video frame size & rate

3 Simple steps, then address bottlenecks

- Fixed-point quantization – esp. multipliers!
- Minimize/avoid use of off-chip RAM
- Parallelize operations for speed
- Use HDL Coder optimizations & reports



Automate Fixed-Point Quantization with Fixed-Point Designer

Simulate with representative data to collect required ranges

Fixed-Point Designer proposes data types

Choose to apply proposed types or set your own

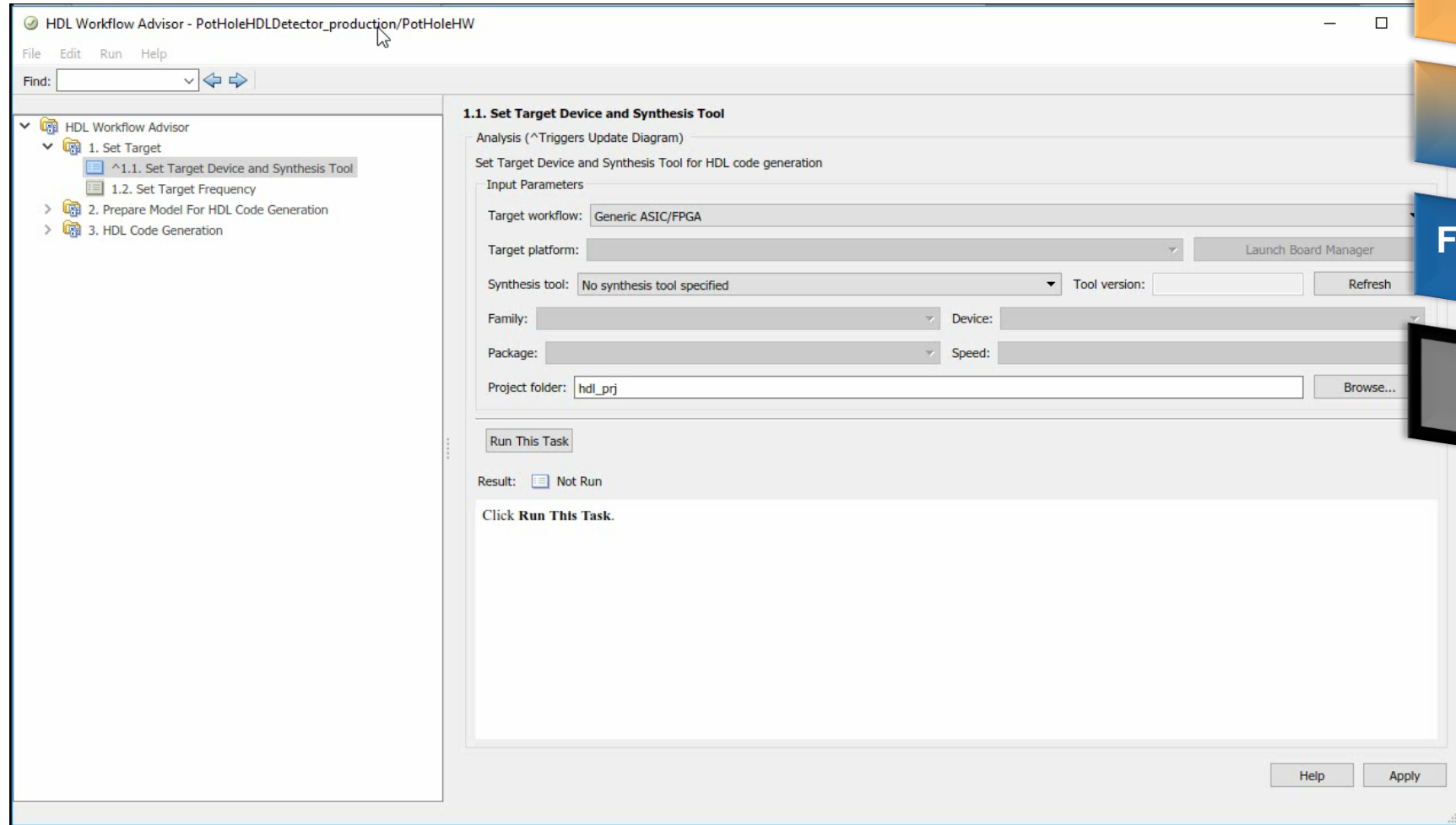
Simulate and compare results

The screenshot displays the Fixed-Point Designer tool interface. The top toolbar includes options for Simulation Ranges, Derived Ranges, Collect Ranges, MATLAB Functions, Propose Data Types, Apply Data Types, Simulate with Embedded Types, and Compare Results. The main workspace is divided into several panes:

- MODEL HIERARCHY:** Shows a tree view of the system under design, including components like PotHoleHDLDetector, DetectCentroid, and Centroid31.
- Results Table:** A table listing simulation results for various components. The columns include Name, Run, CompiledDT, SpecifiedDT, ProposedDT, Accept, SimMin, and SimMax.
- RUN BROWSER:** Shows the current simulation run (Run 1) is selected.
- Visualization of Simulation Data:** A histogram titled "Histograms of all results in the model" showing the distribution of simulation data. The y-axis is labeled "Histogram Bins" and ranges from 2^{-13} to 2^{12} . The x-axis represents the range of values. A legend indicates:
 - Overflows (Red)
 - Representable (Grey)
 - In-Range (Blue)
 - Underflows (Yellow)

Name	Run	CompiledDT	SpecifiedDT	ProposedDT	Accept	SimMin	SimMax
AreaThresh	Run 1		Inherit: auto	fixdt(0, 10, 3)	<input checked="" type="checkbox"/>		
BOut	Run 1		Inherit: auto	n/a			
DetectCentroid/Centroid31/CentroidKernel/1-D Lookup Table1 : Br...	Run 1		Inherit: Same ...	n/a			
DetectCentroid/Centroid31/CentroidKernel/1-D Lookup Table1 : Int...	Run 1		Inherit: Same ...	n/a			
DetectCentroid/Centroid31/CentroidKernel/1-D Lookup Table1 : Ou...	Run 1	single	single	fixdt(0, 16, 15)	<input checked="" type="checkbox"/>	0.0010405827...	1
DetectCentroid/Centroid31/CentroidKernel/1-D Lookup Table1 : Table	Run 1		single	fixdt(0, 16, 15)	<input checked="" type="checkbox"/>		
DetectCentroid/Centroid31/CentroidKernel/Data Type Conversion4	Run 1	fixdt(1, 16, 10)	fixdt(1, 16, 10)	fixdt(1, 16, 11)	<input checked="" type="checkbox"/>	-15	15
DetectCentroid/Centroid31/CentroidKernel/Data Type Conversion5	Run 1	fixdt(1, 16, 10)	fixdt(1, 16, 10)	fixdt(1, 16, 11)	<input checked="" type="checkbox"/>	-15	15
DetectCentroid/Centroid31/CentroidKernel/Data Type Conversion6	Run 1	single	single	fixdt(1, 16, 3)	<input checked="" type="checkbox"/>	-3608	3720
DetectCentroid/Centroid31/CentroidKernel/Data Type Conversion7	Run 1	single	single	fixdt(1, 16, 3)	<input checked="" type="checkbox"/>	-2201	3720
DetectCentroid/Centroid31/CentroidKernel/EnTapDelay23/In1	Run 1		Inherit: auto	fixdt(1, 16, 8)	<input checked="" type="checkbox"/>		

Production Target – IP Core Gen

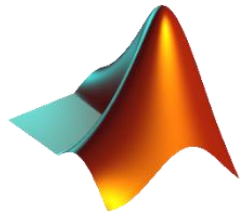


**Algorithm
(Golden Reference)**

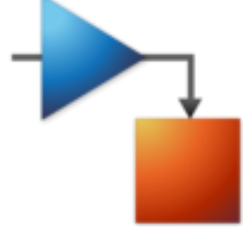
**Algorithm w/ HW
Implementation**

**Fixed-Point, Optimized
Implementation**

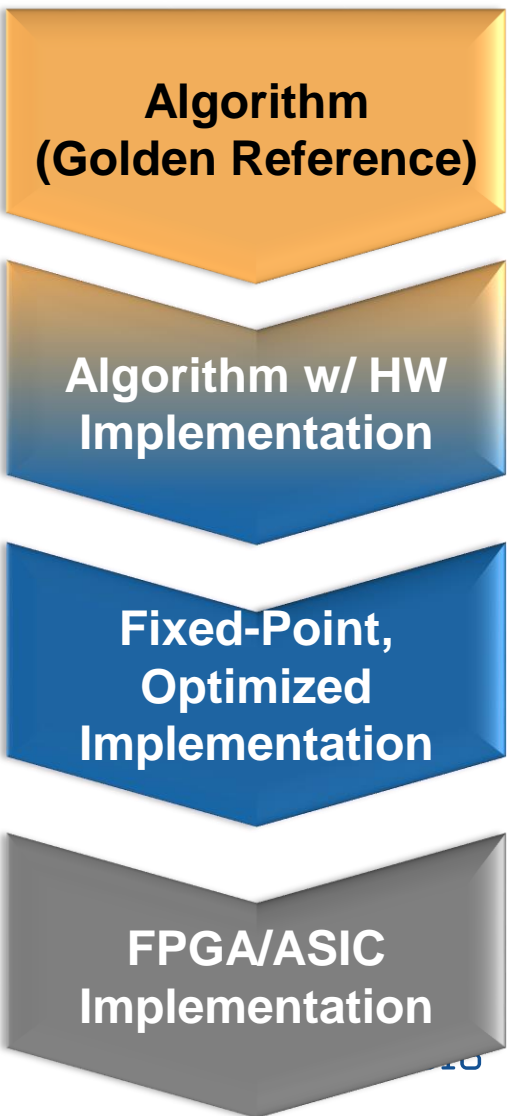
**FPGA/ASIC
Implementation**



MATLAB



Simulink



```

%% Frame pre-processing
% Convert to intensity
frmGray = rgb2gray(frmIn);

% Bilateral filter
frmBiFilt = imbilatfilt(frmGray, 'NeighborhoodSize', 9);

% Edge detection
frmEdge = edge(frmBiFilt, 'Sobel', .05);

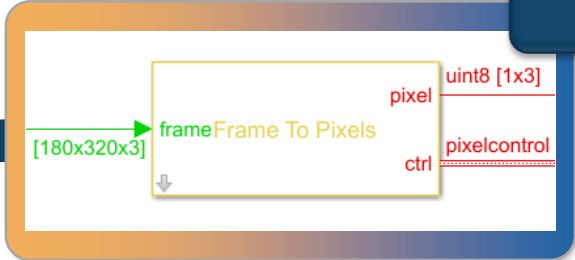
%% Trapezoidal mask

```

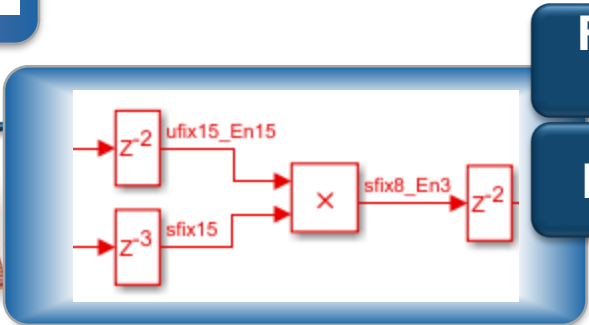
Floating Point

HDL-ready IP blocks

Verify



HDL Coder Prototype



Fixed Point Designer

HDL Coder

HDL Coder Production

- ✓ Cosimulation
- ✓ Export DPI-C models

HDL Verifier

Core interface

VHDL / Verilog

Constraints

18a Key Features

R2018a

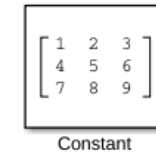
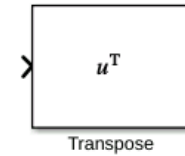
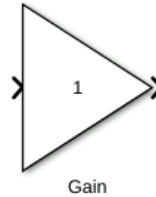
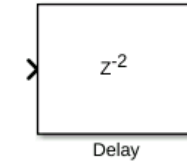
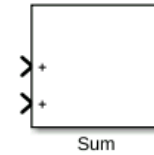
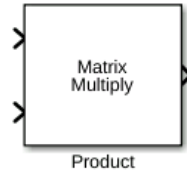
Theme	Feature
Simulink modeling	<u>Matrix support in HDL Coder</u>
	<u>Model Advisor Integration</u>
	<u>Line level traceability</u>
Optimizations	<u>Critical Path Estimation for Floating point algorithms</u>
	Constant folding and strength reduction for math operations
	Floating point algorithmic improvements
Workflow	<u>Microsemi workflow</u>
	<u>Test points support in IP Core workflow</u>
	<u>Synthesis reporting</u>

Supported HDL Blocks with Matrix Types

R2018a

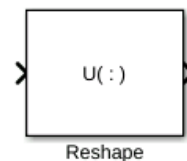
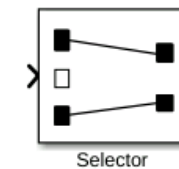
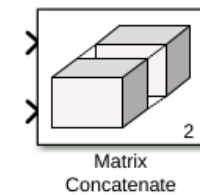
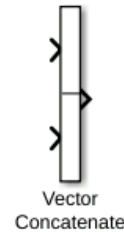
- Functional blocks

- Product
- Gain
- Sum
- Transpose
- Delay
- Constant

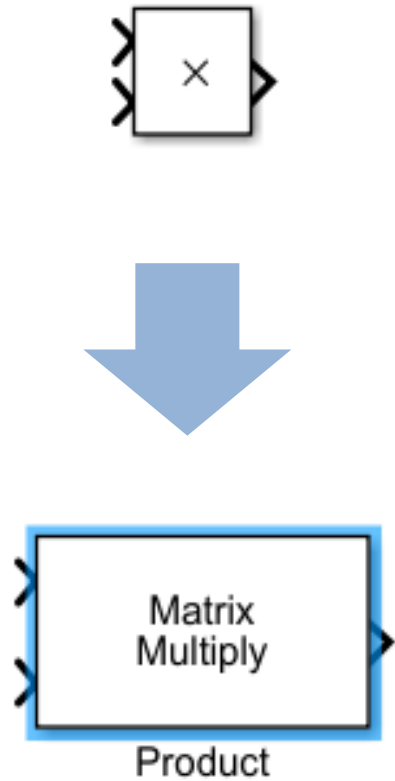


- Wiring blocks

- Vector/Matrix Concatenate
- Selector
- Reshape
- No HDL Implementation



Matrix Product



Block Parameters: Product

Product

Multiply or divide inputs. Choose element-wise or matrix product and specify one of the following:

a) * or / for each input port. For example, **/* performs the operation 'u1*u2/u3*u4'.

b) scalar specifies the number of input ports to be multiplied.

If there is only one input port and the Multiplication parameter is set to Element-wise(*), a single * or / collapses the input signal using the specified operation. However, if the Multiplication parameter is set to Matrix(*), a single * causes the block to output the matrix unchanged, and a single / causes the block to output the matrix inverse.

Main Signal Attributes

Number of inputs: 2

Multiplication: Matrix(*)

OK Cancel Help Apply

Configuration of product block in matrix mode

HDL Properties: mmul

General Native Floating Point

Implementation Architecture: Matrix Multiply

Implementation Parameters

ConstrainedOutputPipeline: 0

DotProductStrategy: Fully Parallel

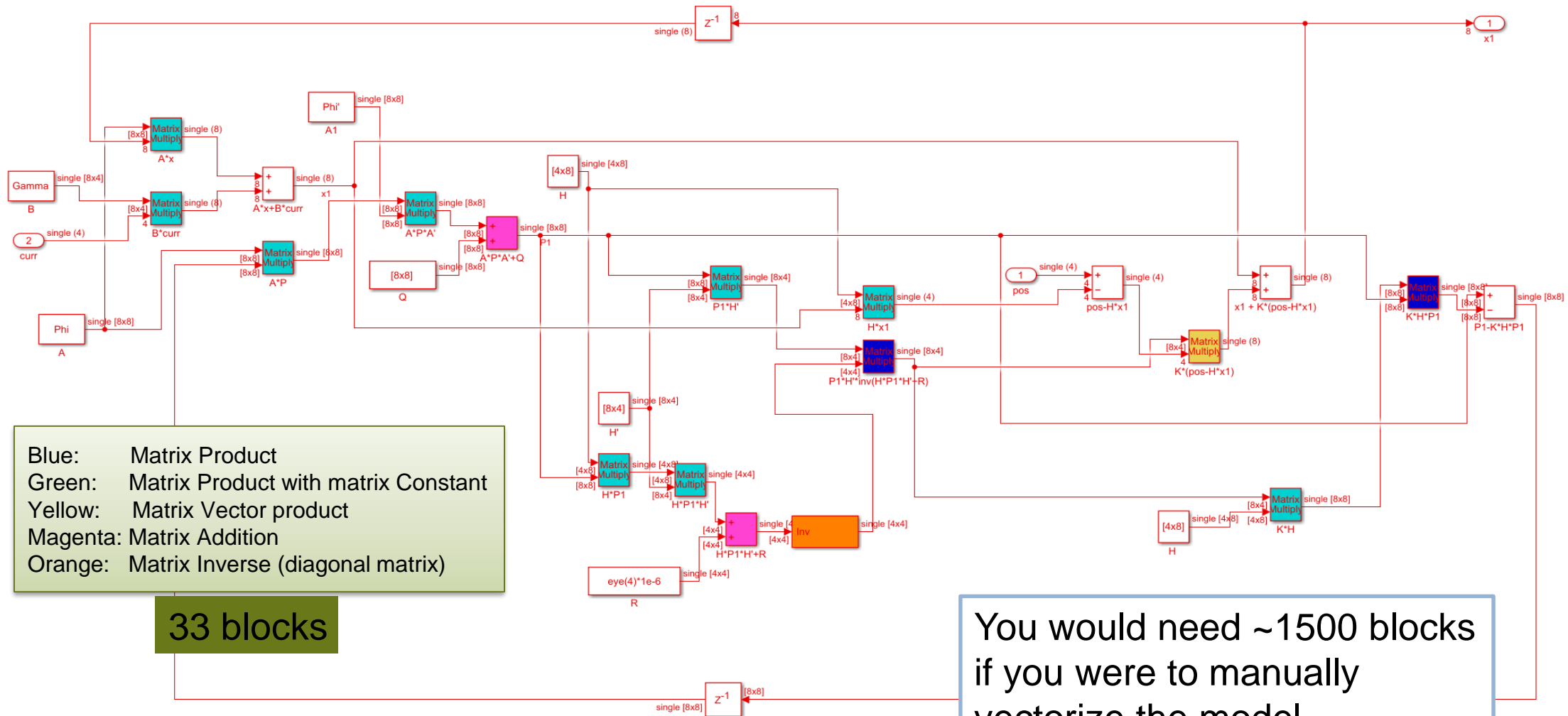
DSPStyle: Serial Multiply-Accumulate

InputPipeline: Parallel Multiply-Accumulate

OutputPipeline: 0

OK Cancel Help Apply

Matrix Support Use Case



Blue: Matrix Product
Green: Matrix Product with matrix Constant
Yellow: Matrix Vector product
Magenta: Matrix Addition
Orange: Matrix Inverse (diagonal matrix)

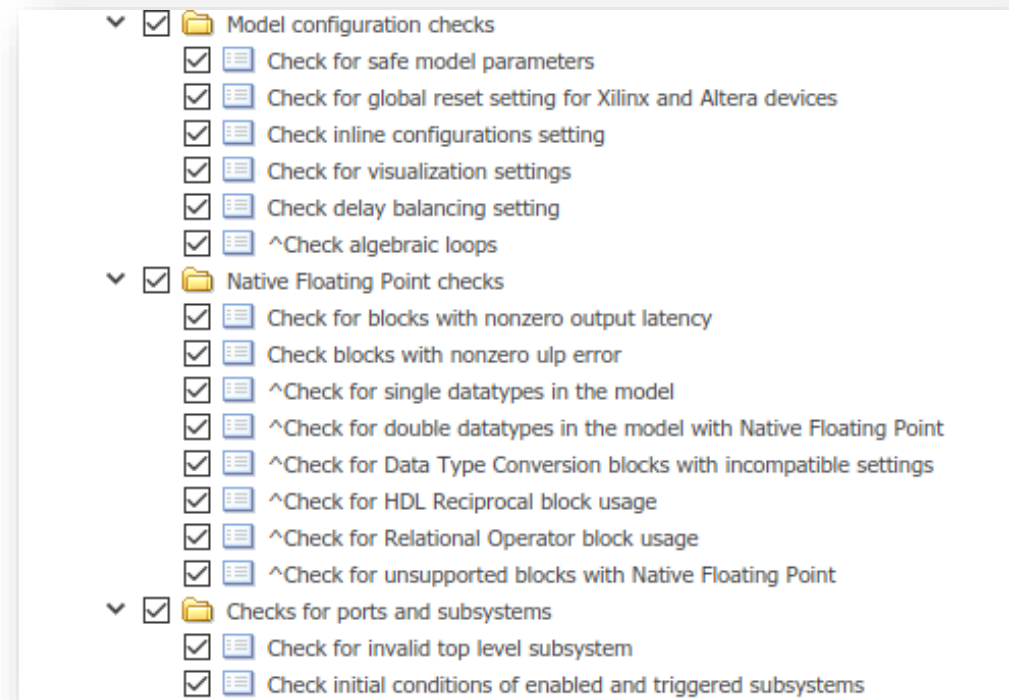
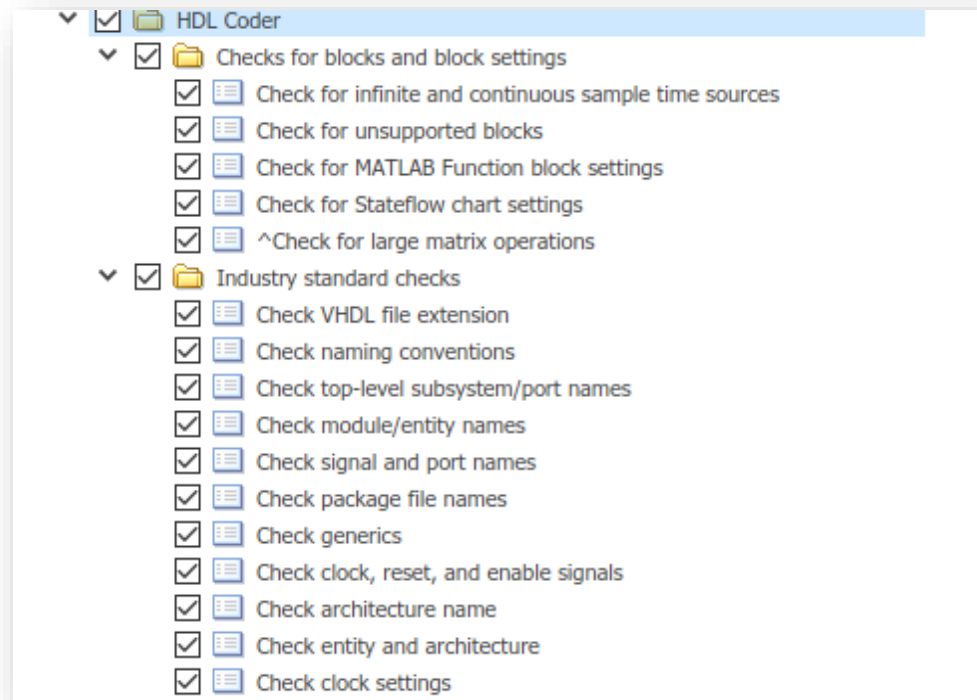
33 blocks

You would need ~1500 blocks if you were to manually vectorize the model

Model Advisor - Workflow

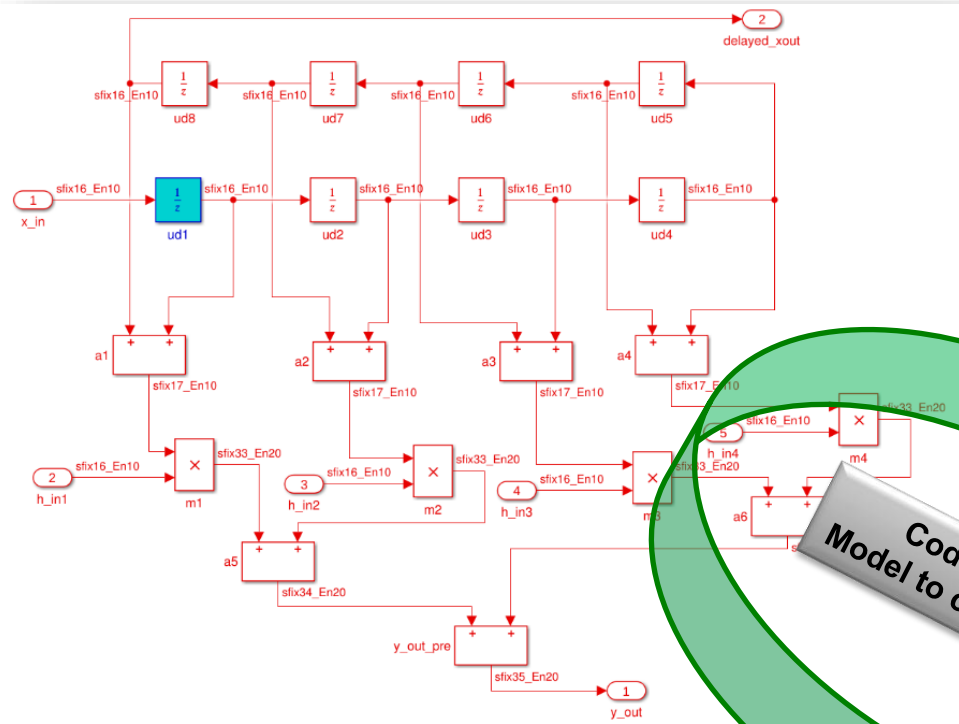
R2018a

- hdlmodelchecker
- Checks and fixes, Standalone infrastructure
- HDL Coder Model Advisor Integration



New Line level traceability

Customers: Safety-Critical Application(Aero/Def, Auto, Medical)
Ex) DO 254 customers



R2018a

HDL Code Generation Report

Find: Match Case

Contents

- Summary
- Clock Summary
- Code Interface Report
- Traceability Report

Generated Source Files

- symmetric_fir.vhd (10)

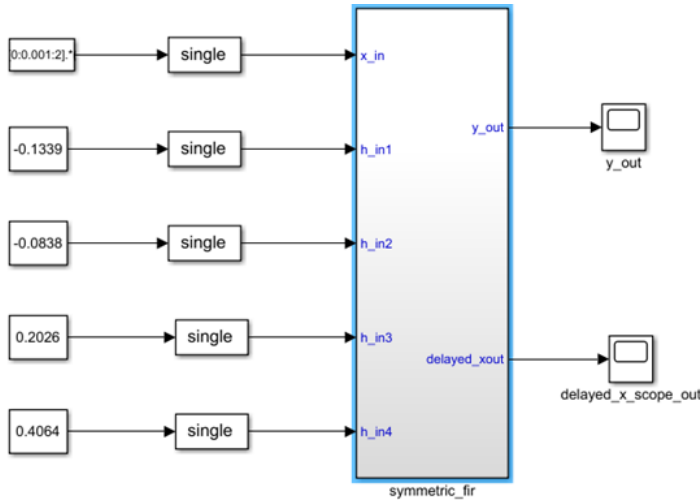
Referenced Models

```

103 SIGNAL y_out_pre_add_cast : signed(34 DOWNTC
104 SIGNAL y_out_pre_add_cast_1 : signed(34 DOWNTC
105 SIGNAL y_out_pre_out1 : signed(34 DOWNTC
106
107 BEGIN
108 x_in_signed <= signed(x_in);
109
110 enb <= clk_enable;
111
112
113 ud1_process : PROCESS (clk, reset)
114 BEGIN
115 IF reset = '1' THEN
116 ud1_out1 <= to_signed(16#0000#, 16);
117 ELSIF clk'EVENT AND clk = '1' THEN
118 IF enb = '1' THEN
119 ud1_out1 <= x_in_signed;
120 END IF;
121 END IF;
122 END PROCESS ud1_process;
123
124 ud2_process : PROCESS (clk, reset)
125 BEGIN
126 IF reset = '1' THEN
127 ud2_out1 <= to_signed(16#0000#, 16);
128 ELSIF clk'EVENT AND clk = '1' THEN
129 IF enb = '1' THEN
130 ud2_out1 <= ud1_out1;
131 END IF;
132 END IF;
133 END PROCESS ud2_process;
    
```

Code to model & Model to code traceability

Synthesis Timing and Area Summary Reporting



HDL Workflow Advisor

- > 1. Set Target
- > 2. Prepare Model For HDL Code Generation
- > 3. HDL Code Generation
- > 4. FPGA Synthesis and Analysis
 - ✓ 4.1. Create Project
 - ✓ 4.2. Perform Synthesis and P/R
 - ✓ 4.2.1. Perform Logic Synthesis
 - ✓ 4.2.2. Perform Mapping
 - ✓ 4.2.3. Perform Place and Route
 - 4.3. Annotate Model with Synthesis Result

4.2.3. Perform Place and Route

Analysis

Run place and route for specified FPGA device

Input Parameters

Skip this task

Ignore place and route errors

Run This Task

Result: ✓ Passed

Passed Place and Route.

Parsed resource summary file: [symmetric_fir_quartus_fit.rpt](#)

Resource summary	
Resource	Usage
LUTs	4234
Registers	4487
Memory	19
DSPs	16

Parsed timing summary file: [symmetric_fir_postroute.tqr](#)

Timing summary	
	Value (ns)
Data delay	2.885
Slack	-0.909
Timing constraints	not met

Task "Perform Place and Route" successful.
Generated logfile: [hdl_pri/hdlsrc/sfir_single_blockflow_task_PerformPlaceAndRoute_](#)
[0;32mInfo: *****

R2018a

Resource summary file

```

-----+-----+-----+
; Resource                               ; Usage                               ;
-----+-----+-----+
; ALUTs Used                             ; 4,298 / 58,080 ( 7 %)             ;
;   -- Combinational ALUTs               ; 4,234 / 58,080 ( 7 %)             ;
;   -- Memory ALUTs                      ; 64 / 29,040 ( < 1 %)             ;
;   -- LUT_REGS                           ; 0 / 58,080 ( 0 %)                 ;
; Dedicated logic registers               ; 4,487 / 58,080 ( 8 %)             ;
; M9K blocks                              ; 19 / 462 ( 4 %)                   ;
; M144K blocks                            ; 0 / 16 ( 0 %)                     ;
; Total MLAB memory bits                  ; 528                                ;
; Total block memory bits                 ; 4,601 / 6,617,088 ( < 1 %)       ;
; Total block memory implementation bits   ; 175,104 / 6,617,088 ( 3 %)       ;
; DSP block 18-bit elements                ; 16 / 384 ( 4 %)                   ;
-----+-----+-----+

```

Timing summary file

```

; Slack                                   ; -0.909 (VIOLATED)
-----+-----+-----+
; Statistics
-----+-----+-----+
; Property                                ; Value ; Count ;
-----+-----+-----+
; Setup Relationship                       ; 2.000 ;      ;
; Clock Skew                              ; 0.053 ;      ;
; Data Delay                               ; 2.885 ;      ;
-----+-----+-----+

```

Support Microsemi Libero

- Microsemi Libero[®] SoC 11.8 SP2
- Microsemi Libero SoC Polarfire[®] 2.0

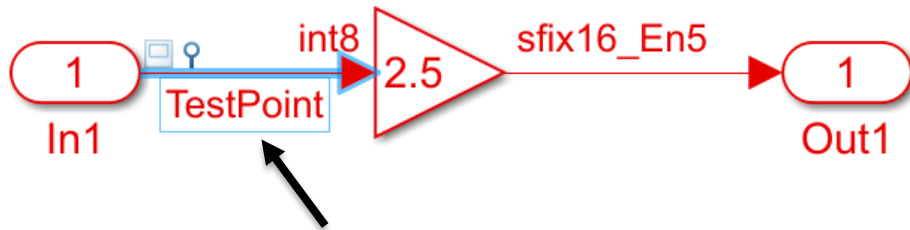
The screenshot displays the HDL Workflow Advisor window for a project named 'sfir_fixed/symmetric_fir'. The interface is divided into a left-hand navigation pane and a main configuration area. The navigation pane shows a tree view of tasks, with '1.1. Set Target Device and Synthesis Tool' selected and highlighted in green. The main area is titled '1.1. Set Target Device and Synthesis Tool' and contains the following configuration options:

- Analysis (^Triggers Update Diagram)**: Set Target Device and Synthesis Tool for HDL code generation.
- Input Parameters**:
 - Target workflow: Generic ASIC/FPGA
 - Target platform: [Dropdown menu]
 - Synthesis tool: Microsemi Libero SOC (highlighted with a red box)
 - Tool version: 11.8
 - Family: SmartFusion2
 - Device: M2S150TS
 - Package: 1152 FC
 - Speed: -1
 - Project folder: hdl_prj

At the bottom of the configuration area, there is a 'Run This Task' button and a 'Result' section showing a green checkmark and the text 'Passed Set Target Device and Synthesis Tool.' The bottom right corner of the window has 'Help' and 'Apply' buttons.

Testpoints – IP Core Support

R2018a



1. Define test point signals in Simulink

```

ENTITY HDL_DUT_ip_src_HDL_DUT IS
  PORT(
    clk          : IN  std_logic;
    reset       : IN  std_logic;
    clk_enable   : IN  std_logic;
    in1         : IN  std_logic_vector(7 DOWNTO 0);
    ce_out      : OUT std_logic;
    out_rsvd    : OUT std_logic_vector(15 DOWNTO 0);
    tp_TestPoint : OUT std_logic_vector(7 DOWNTO 0);
  );
END HDL_DUT_ip_src_HDL_DUT;

```

4. Debug or Test with Generated Port

Processor/FPGA synchronization: Free running

Target platform interface table

Port Name	Port Type	Data Type	Target Platform Interfaces	Bit Range / Address / FPGA Pin
In1	Inport	int8	AXI4-Lite	x"100"
Out1	Outport	sfix16_En5	AXI4-Lite	x"110"
TestPoint	Test point	int8	AXI4-Lite	x"120"

2. Assign interface mapping in HDL workflow advisor



3. Generate HDL interface code with test point signals

Critical Path Estimation with NFP

R2018a

Find: Match Case

Contents

- Summary
- Clock Summary
- Code Interface Report
- Timing And Area Report
 - High-level Resource Report
 - Native Floating-Point Resource Report
 - Critical Path Estimation**
- Optimization Report
 - Distributed Pipelining
 - Streaming and Sharing
 - Delay Balancing
 - Adaptive Pipelining
- Traceability Report

Generated Source Files

- FOC_Current_Control_pkg.vhd
- nfp_relop_comp.vhd
- nfp_relop_comp_block.vhd
- nfp_sincos_comp.vhd
- nfp_add_comp.vhd
- nfp_mul_comp.vhd
- nfp_relop_comp_block1.vhd
- nfp_relop_comp_block2.vhd
- nfp_gain_pow2_comp.vhd
- nfp_sub_comp.vhd
- nfp_umihus_comp.vhd
- nfp_add2_comp.vhd
- nfp_relop_comp_block3.vhd
- FOC_Current_Control.vhd

Referenced Models

Critical Path Report for hdlcoderFocCurrentFloatHdl/FOC_Current_Control

Summary Section

Critical Path Delay : 186.884 ns
 Critical Path Begin : [Delay_Register3](#)
 Critical Path End : [rd_1](#)
 Highlight: Critical Path: [hdl_prj2/hdlsrc/hdlcoderFocCurrentFloatHdl/criticalPathEstimated.m](#)

Critical Path Details

Id	Propagation (ns)	Delay (ns)	Block Path
1	0.2980	0.2980	Delay_Register3
2	46.9460	46.6480	Sine_Cosine
3	46.9460	0.0000	Subsystem_out1
4	59.8530	12.9070	Product3t
5	75.7260	15.8730	Add22
6	91.5990	15.8730	Error
7	91.5990	0.0000	Error_outbuff
8	104.5060	12.9070	Proportional_Gain1
9	120.2910	15.7850	Add14
10	120.2910	0.0000	Add1_outbuff
11	122.6680	2.3770	Relational_Operator2
12	122.6680	0.0000	Relational_Operator_outbuff1
13	123.6350	0.9670	Switch2
14	126.0120	2.3770	Relational_Operator11
15	126.0120	0.0000	Relational_Operator1_outbuff1
16	126.9790	0.9670	Switch11
17	126.9790	0.0000	Voltage1
18	126.9790	0.0000	qVoltage
19	138.8570	11.8780	Product41
20	154.7300	15.8730	Add21
21	167.6370	12.9070	Gain11
22	167.6370	0.0000	Gain1_outbuff
23	183.5100	15.8730	Add13
24	183.5100	0.0000	mux
25	183.5100	0.0000	abcVoltage
26	183.5100	0.0000	t1
27	185.8870	2.3770	Max_stage1_compare
28	186.8540	0.9670	Max_stage1_switch
29	186.8840	0.0300	rd_1

OK Help

Path timing estimated within 10%


Max Delay Paths

```

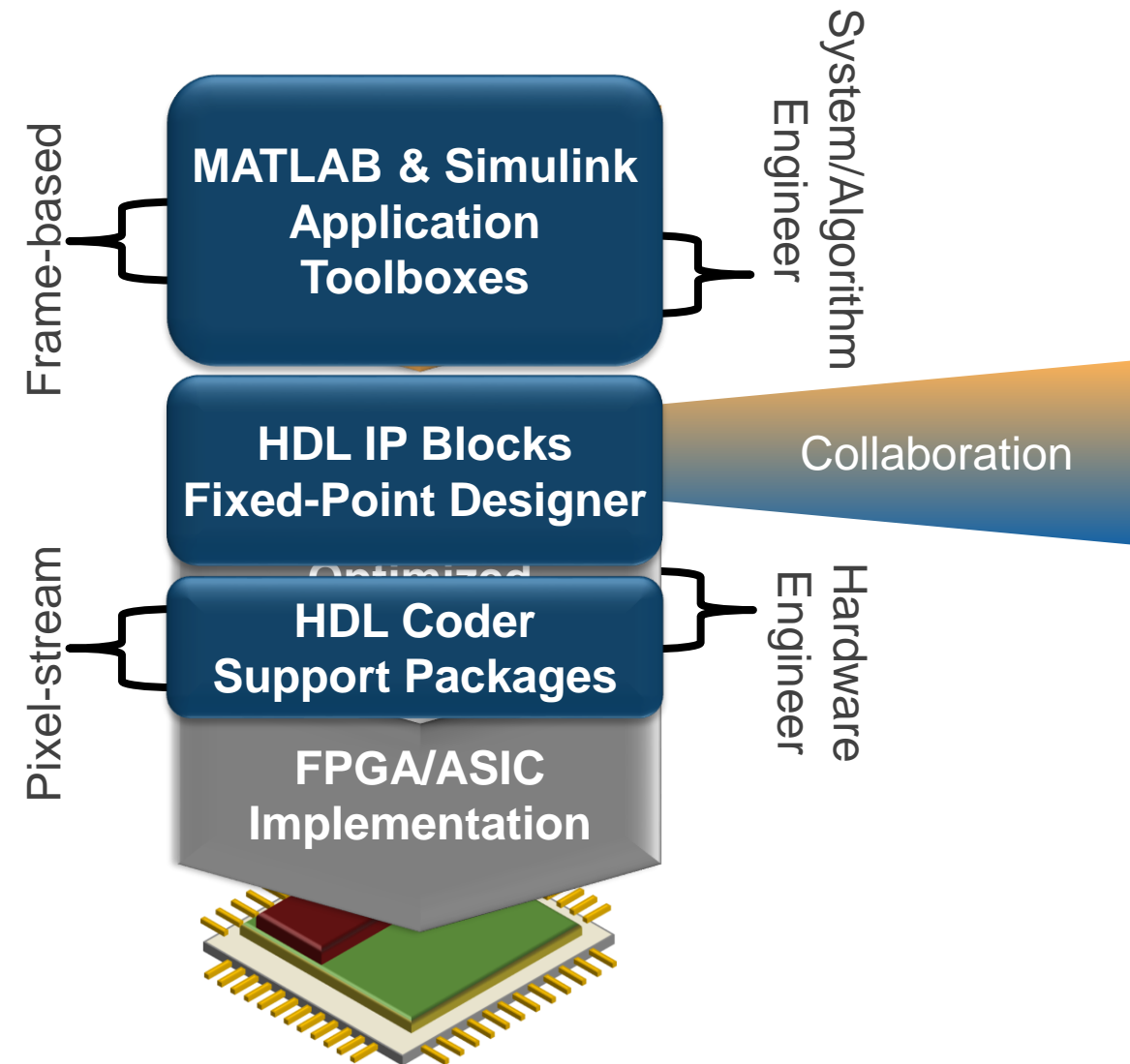
slack (VIOLATED) :      -4.693ns (required time - arrival time)
Source:                Delay_Register3_out1_reg[26]/C
                       (rising edge-triggered cell FDRE clocked by MWCLK {r
Destination:           Max_stage1_val_1_reg[21]/D
                       (rising edge-triggered cell FDRE clocked by MWCLK {r
Path Group:            MWCLK
Path Type:             Setup (Max at Slow Process Corner)
Requirement:          200.000ns (MWCLK rise@200.000ns - MWCLK rise@0.000ns)
Data Path Delay:      204.660ns (logic 73.555ns (35.940%) route 131.104ns (
Logic Levels:         404 (CARRY4=185 DSP48E1=7 LUT1=1 LUT2=27 LUT3=24 LUT4=
Clock Path Skew:      -0.034ns (DCD - SCD + CPR)
Destination Clock Delay (DCD): 0.638ns = ( 200.638 - 200.000 )
Source Clock Delay (SCD): 0.672ns
Clock Pessimism Removal (CPR): 0.000ns
Clock Uncertainty:    0.035ns ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
Total System Jitter (TSJ): 0.071ns
Total Input Jitter (TIJ): 0.000ns
Discrete Jitter (DJ): 0.000ns
Phase Error (PE): 0.000ns
    
```

Xilinx Implementation Timing Report

Outline

- When FPGA, ASIC, or System-on-Chip (SoC) hardware is needed
- Hardware implementation considerations
- Workflow from system/algorithm to FPGA/ASIC hardware
- Demonstration: Vision processing algorithm deployed to FPGA
-  ▪ Conclusion

Workflow From Frames to Pixels to Hardware



- New application innovation happens at the system-level
 - Implemented across software and hardware
- Successful implementation requires collaboration
- Connected workflow to FPGA/ASIC/SoC hardware delivers:
 - Broader micro-architecture exploration
 - Agility to make changes, simulate, generate code
 - Continuous verification