Implementing Video Image Processing Algorithms on FPGA

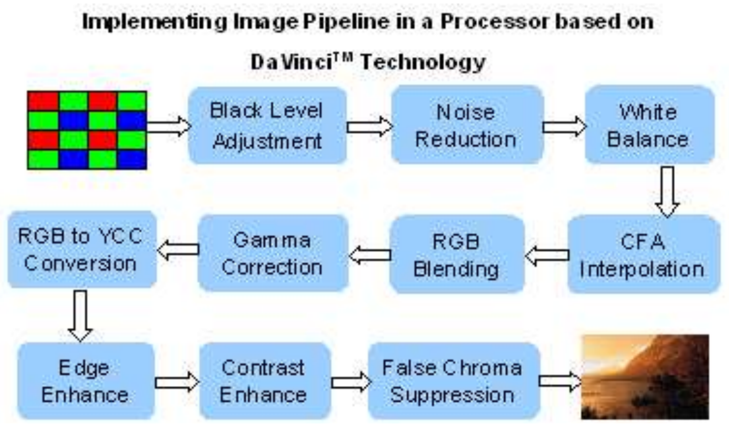
Video Image Processing and Computer Vision

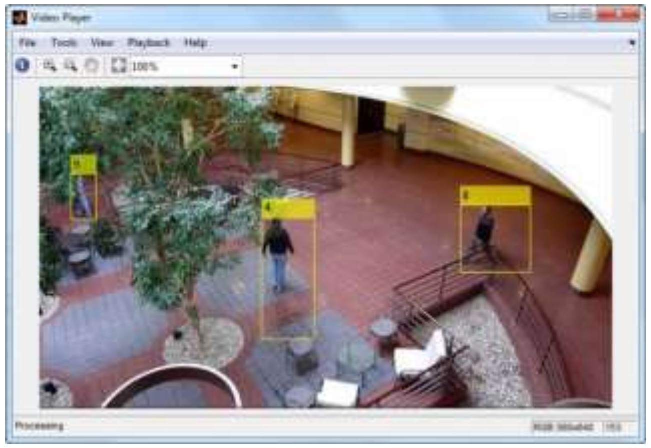
Video Image Processing

- Video in and out
- Gamma correction
- Color balancing
- Noise removal
- Image sharpening

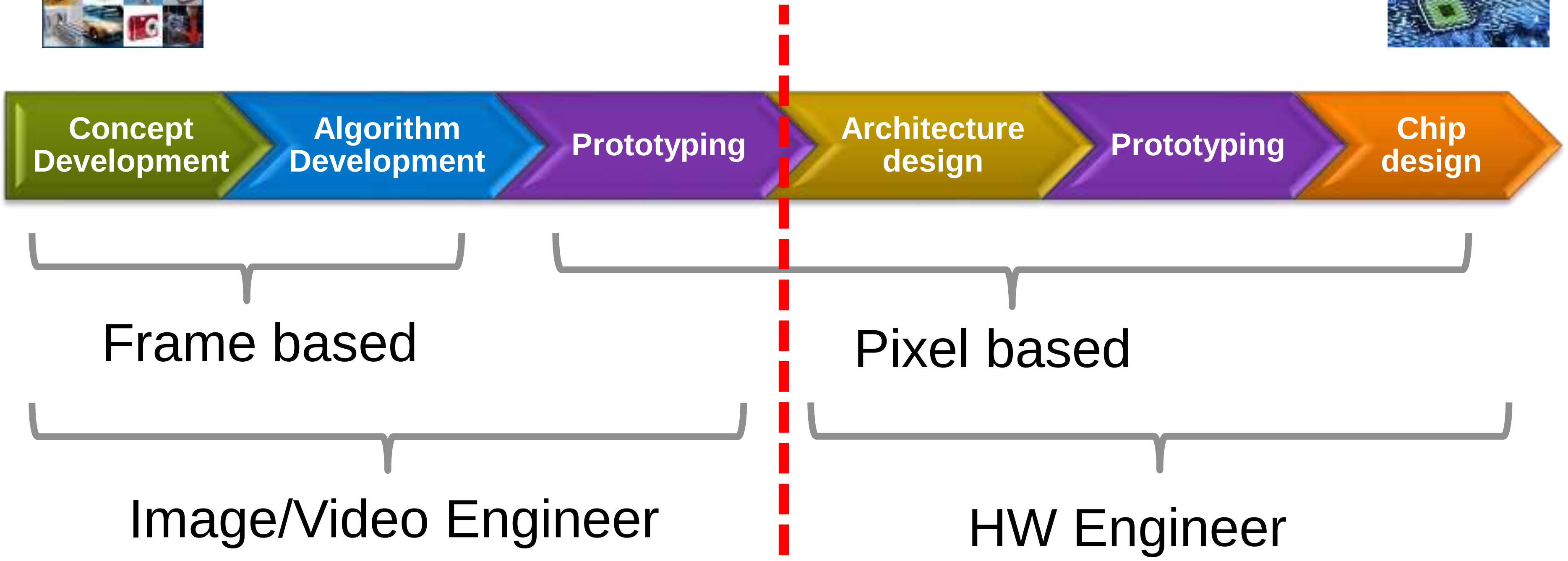
Computer Vision

- Feature matching, and extraction
- Object detection and recognition
- Object Tracking and motion estimation
- Dynamic resolution scaling
- Focus assessment





Workflow for Video Image Processing



Challenges in Design and Prototyping for Video and Image

Algorithm Designer

Modeling video image processing systems

- ✓ Pixel-streaming behavior
- ✓ Code generation ready model
- ✓ Prototyping and concept proofing
- ✓ Technology independent code

Hardware Engineer

Prototyping and Designing FPGA and ASIC for video and image processing algorithms

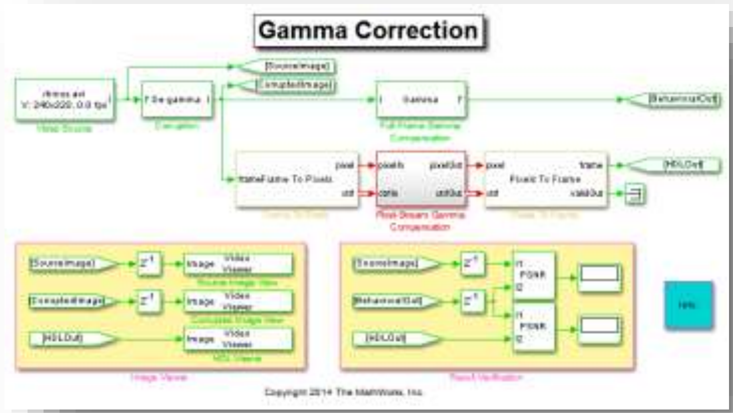
- ✓ Portable, readable and efficient IP Cores
- ✓ Flexible architecture and controllable latency
- ✓ FPGA-in-the-loop testing using ML and SL as frame based test bench

Vision HDL Toolbox

Design and prototype video image processing systems

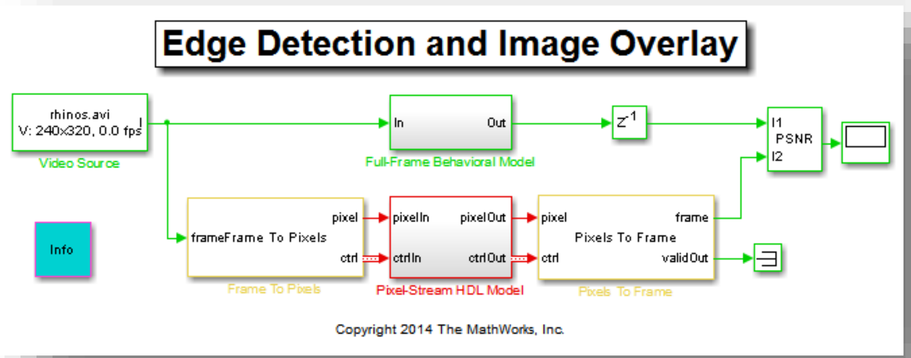
- Modeling hardware behavior of the algorithms
  - Pixel-based functions and blocks
  - Conversion between frames and pixels
  - Standard and custom frame sizes

- Prototyping algorithms on hardware
  - (**With HDL Coder**) Efficient and readable HDL code
  - (**With HDL Verifier**) FPGA-in-the-loop testing and acceleration

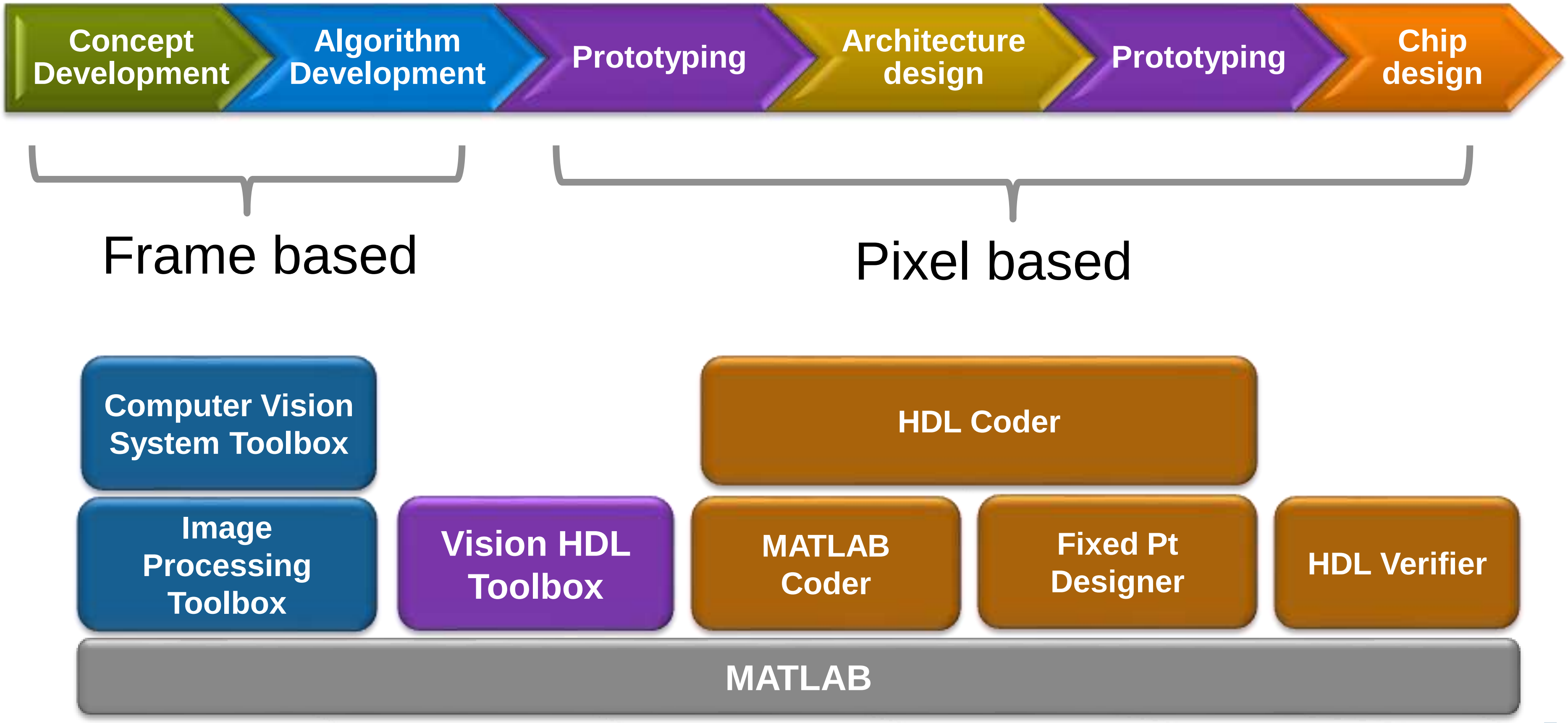


Pixel Based Video Image Algorithms

- **Analysis & Enhancement**
  - Edge Detection, Median Filter
- **Conversions**
  - Chroma Resampling, Color-Space Converter
  - Demosaic Interpolator, Gamma Corrector, Look-up Table
- **Filters**
  - Image Filter, Median Filter
- **Morphological Operations**
  - Dilation, Erosion,
  - Opening, Closing
- **Statistics**
  - Histogram
  - Image Statistics
- **I/O Interfaces**
  - Frame to Pixels, Pixels to Frame, FIL versions
- **Utilities**
  - Pixel Control Bus Creator
  - Pixel Control Bus Selector



A Complete Solution for Embedded Vision

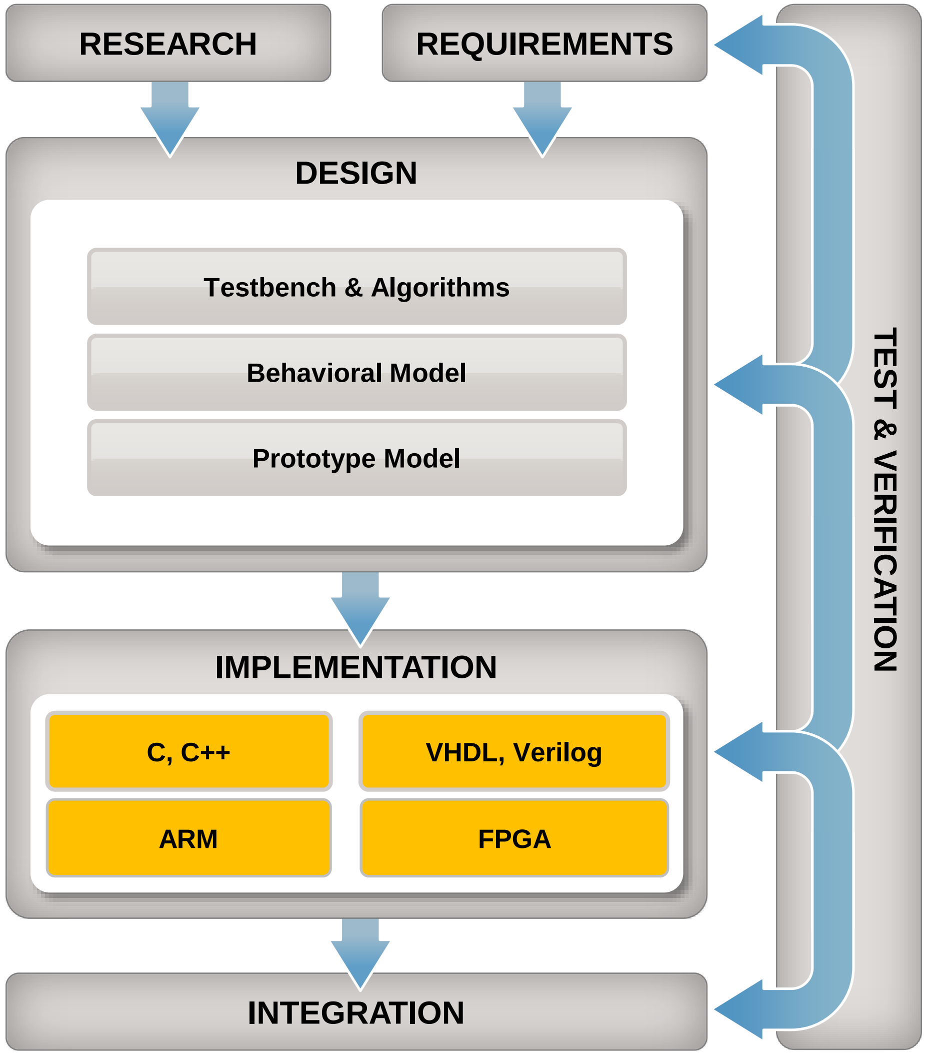


A Complete Solution for Embedded Vision

Product	Capabilities
Vision HDL Toolbox	Design and simulate image processing, video, and computer vision systems for FPGAs and ASICs
HDL Coder	Provide RTL code and testbench generation capability for the functions and blocks in Vision HDL Toolbox
HDL Verifier	Provide FPGA-in-the-loop capability for Vision HDL Toolbox
Computer Vision System Toolbox	Provide frame based computer vision functions and blocks as well as image and video I/O capability
Image Processing Toolbox	Provide image processing and analysis functions

Model-Based Design For Embedded Vision

From Concept to Production



- Build behavioral model for fast simulation and testing
- Convert to prototype model for targeting hardware

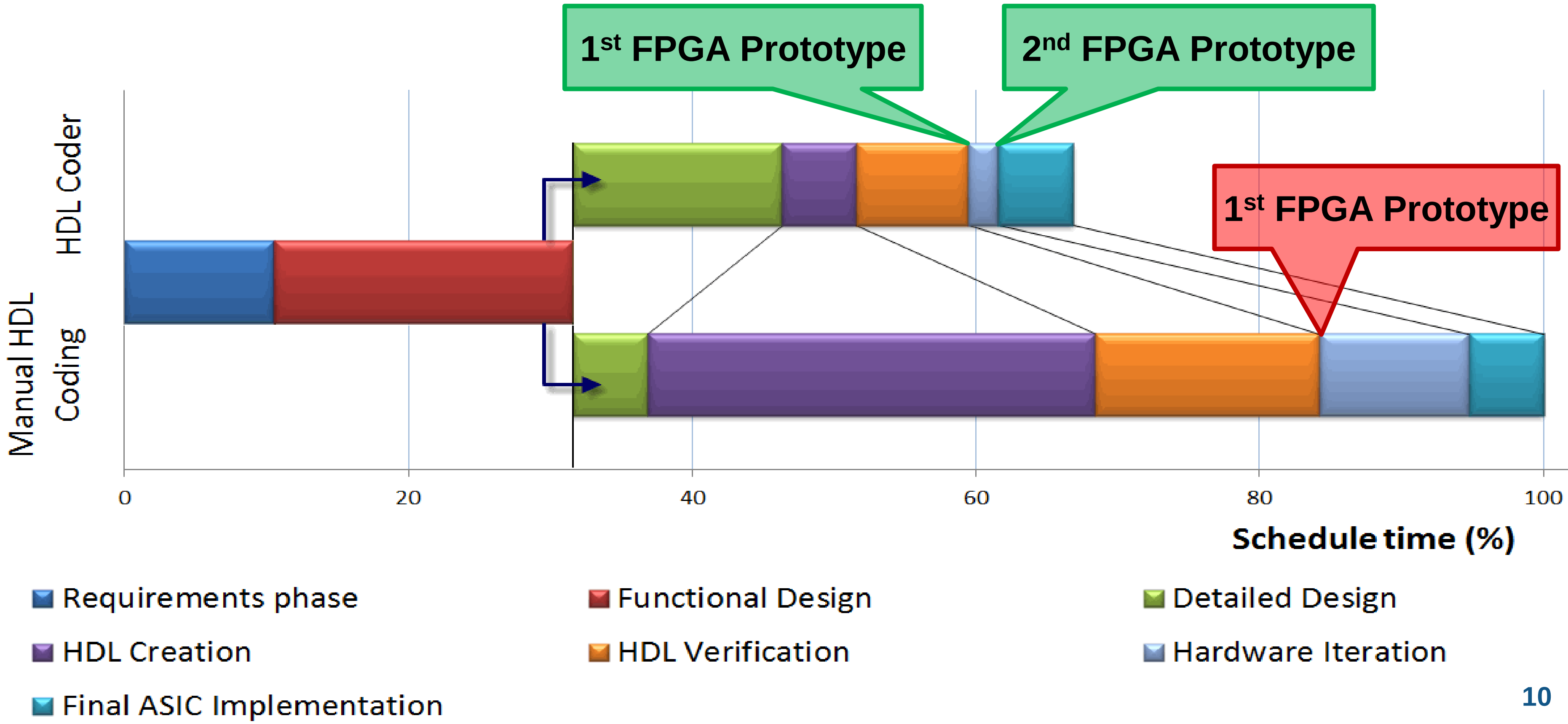
- Generate efficient code
- Explore and optimize implementation tradeoffs

- Automate regression testing
- Detect design errors
- Support certification and standards

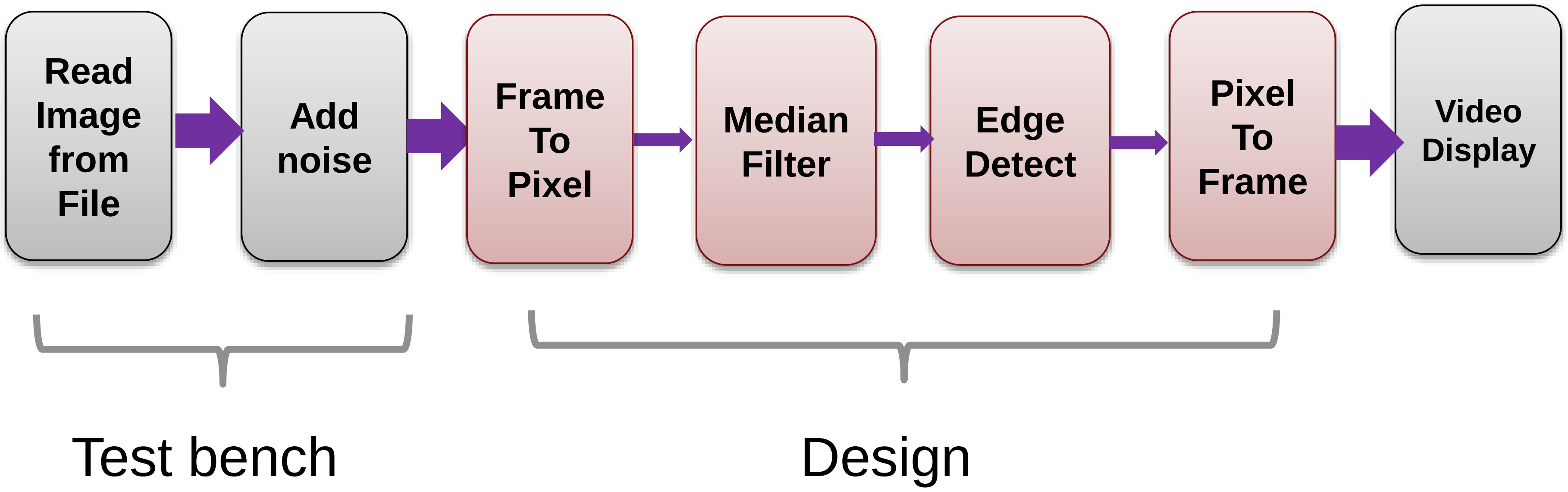
ROI: Customer Adoption Of Model-Based Design

Time spent on FPGA implementation

- Shorter implementation time by 48% (total project 33%)
- Reduced FPGA prototype development schedule by 47%
- Shorter design iteration cycle by 80%

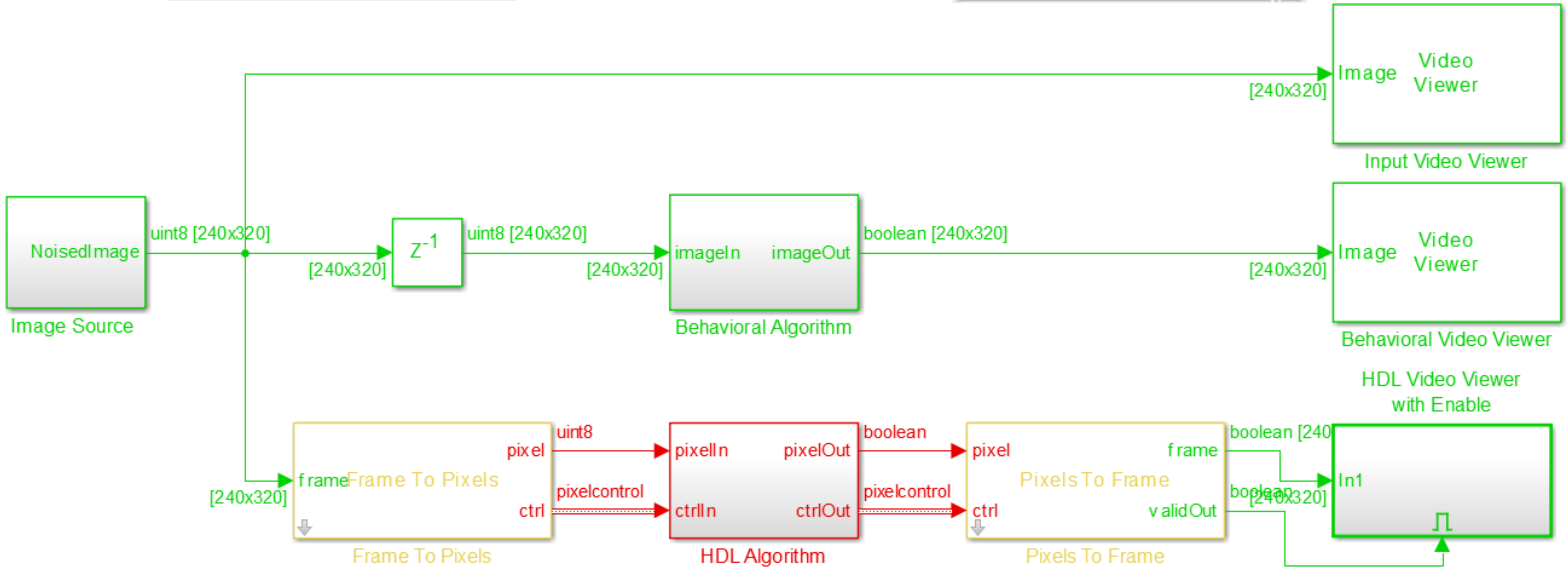


Demo: Enhanced Edge Detection



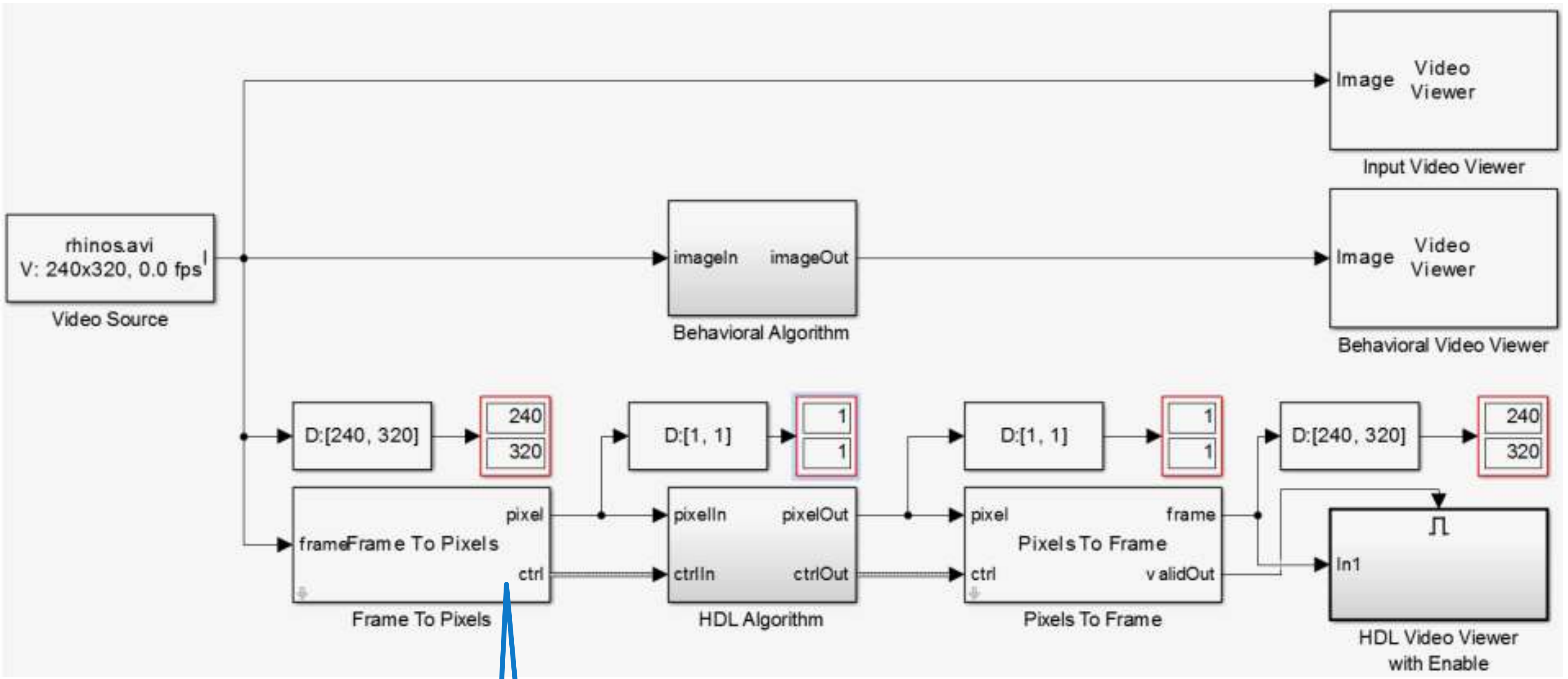
Enhanced Edge Detection

Demo



Streaming Pixel Interface

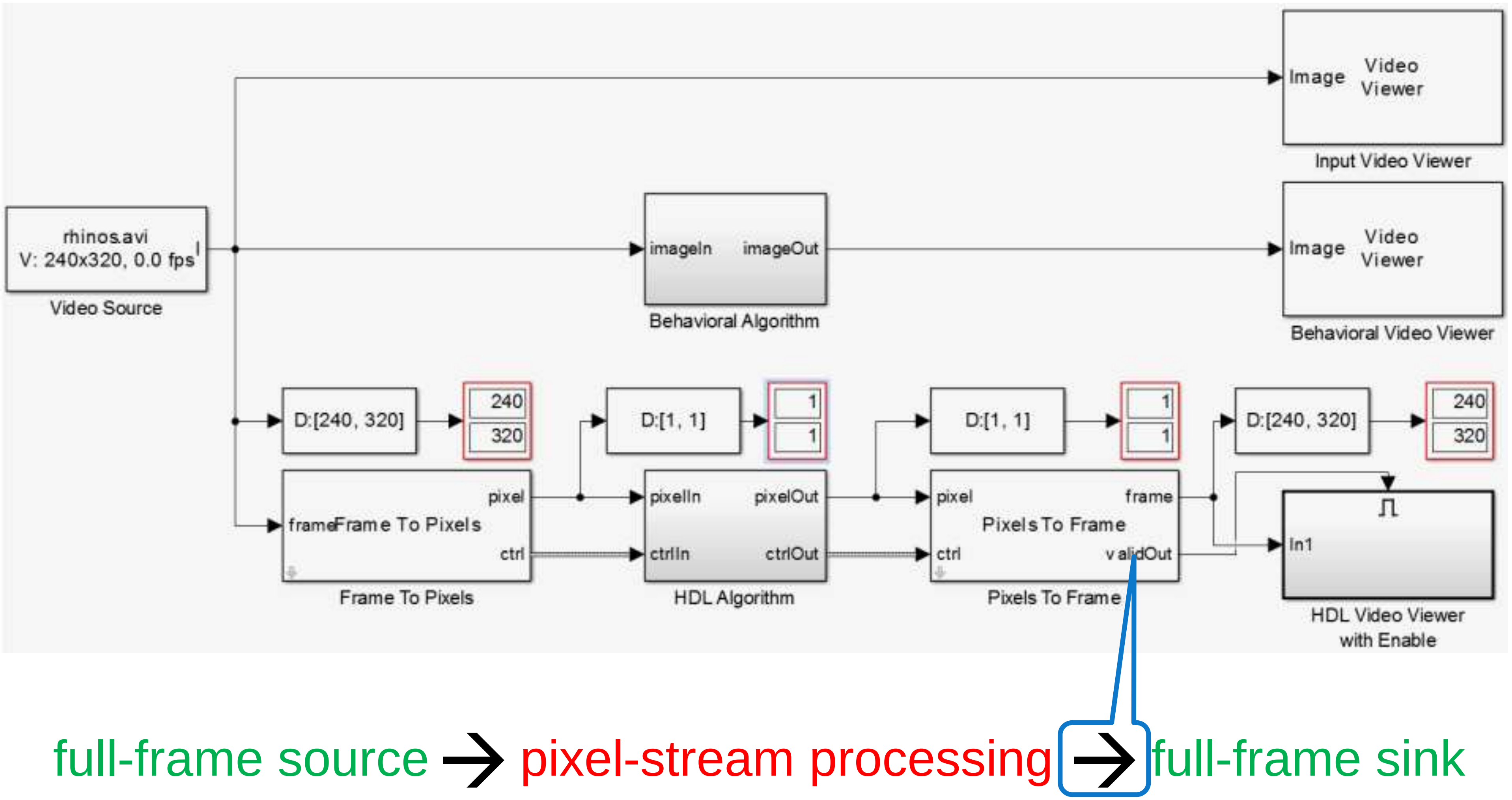
Full Frame vs Pixel Stream



full-frame source → pixel-stream processing → full-frame sink

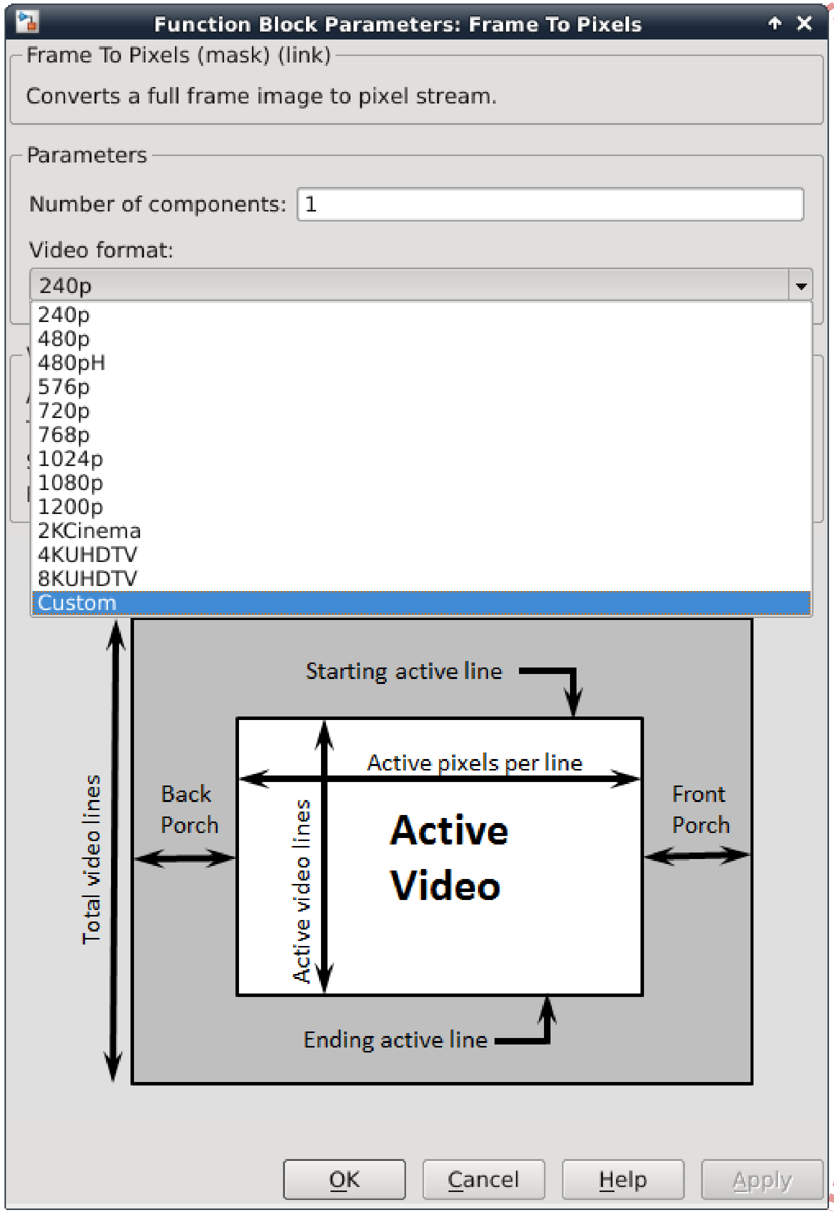
Streaming Pixel Interface

Full Frame vs Pixel Stream



full-frame source → pixel-stream processing → full-frame sink

Frame To Pixels and Pixels To Frame



Function Block Parameters: Frame To Pixels

Frame To Pixels (mask) (link)

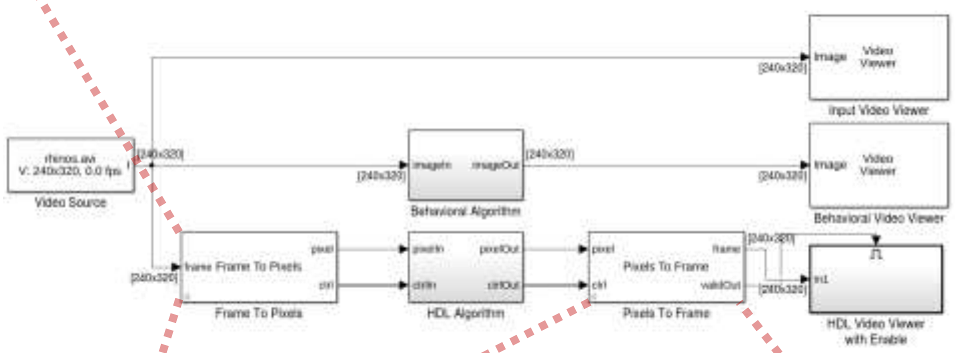
Converts a full frame image to pixel stream.

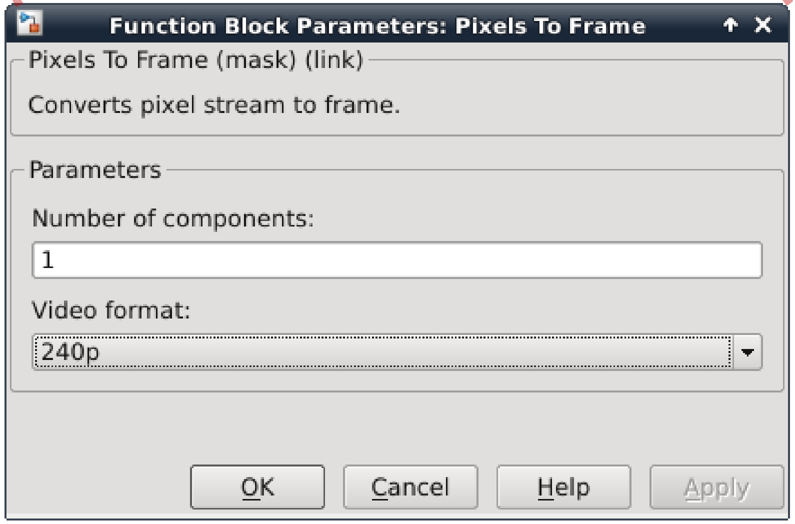
Parameters

Number of components: 1

Video format: 240p

240p
480p
480pH
576p
720p
768p
1024p
1080p
1200p
2KCinema
4KUHDTV
8KUHDTV
Custom





Function Block Parameters: Pixels To Frame

Pixels To Frame (mask) (link)

Converts pixel stream to frame.

Parameters

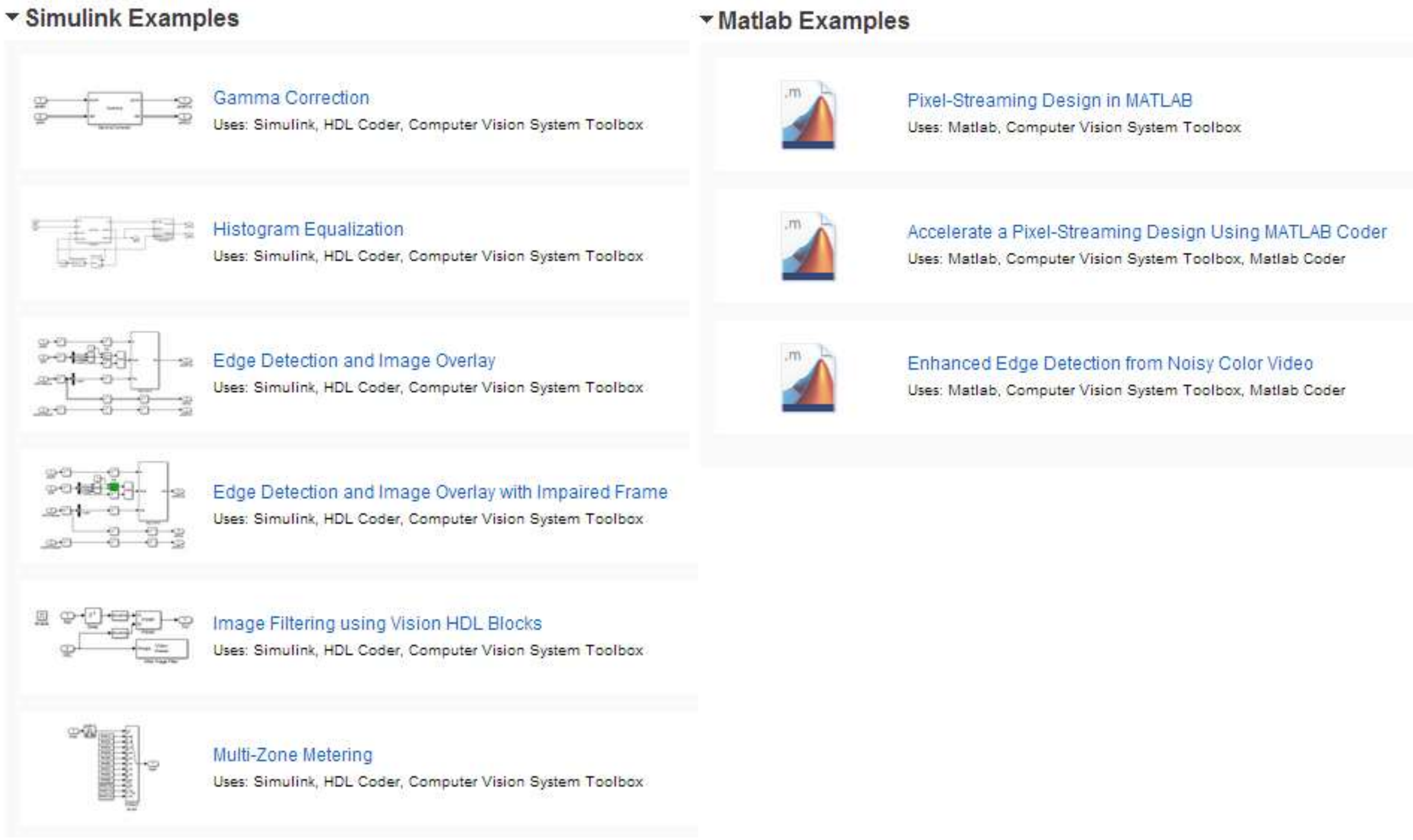
Number of components:

1

Video format:

240p

Examples: Starting Points for Your Models



Simulink Examples

Gamma Correction
Uses: Simulink, HDL Coder, Computer Vision System Toolbox

Histogram Equalization
Uses: Simulink, HDL Coder, Computer Vision System Toolbox

Edge Detection and Image Overlay
Uses: Simulink, HDL Coder, Computer Vision System Toolbox

Edge Detection and Image Overlay with Impaired Frame
Uses: Simulink, HDL Coder, Computer Vision System Toolbox

Image Filtering using Vision HDL Blocks
Uses: Simulink, HDL Coder, Computer Vision System Toolbox

Multi-Zone Metering
Uses: Simulink, HDL Coder, Computer Vision System Toolbox

Matlab Examples

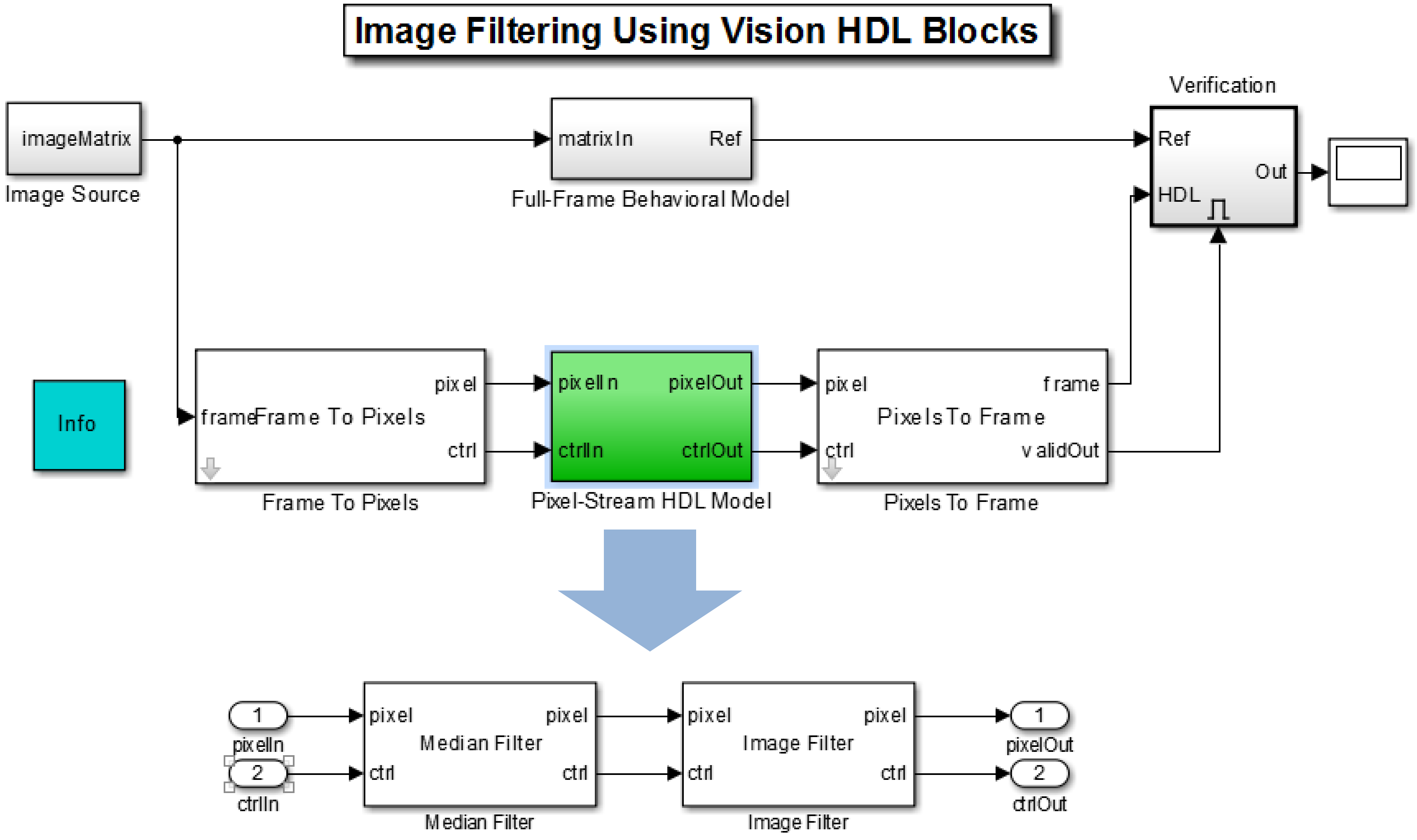
Pixel-Streaming Design in MATLAB
Uses: Matlab, Computer Vision System Toolbox

Accelerate a Pixel-Streaming Design Using MATLAB Coder
Uses: Matlab, Computer Vision System Toolbox, Matlab Coder

Enhanced Edge Detection from Noisy Color Video
Uses: Matlab, Computer Vision System Toolbox, Matlab Coder

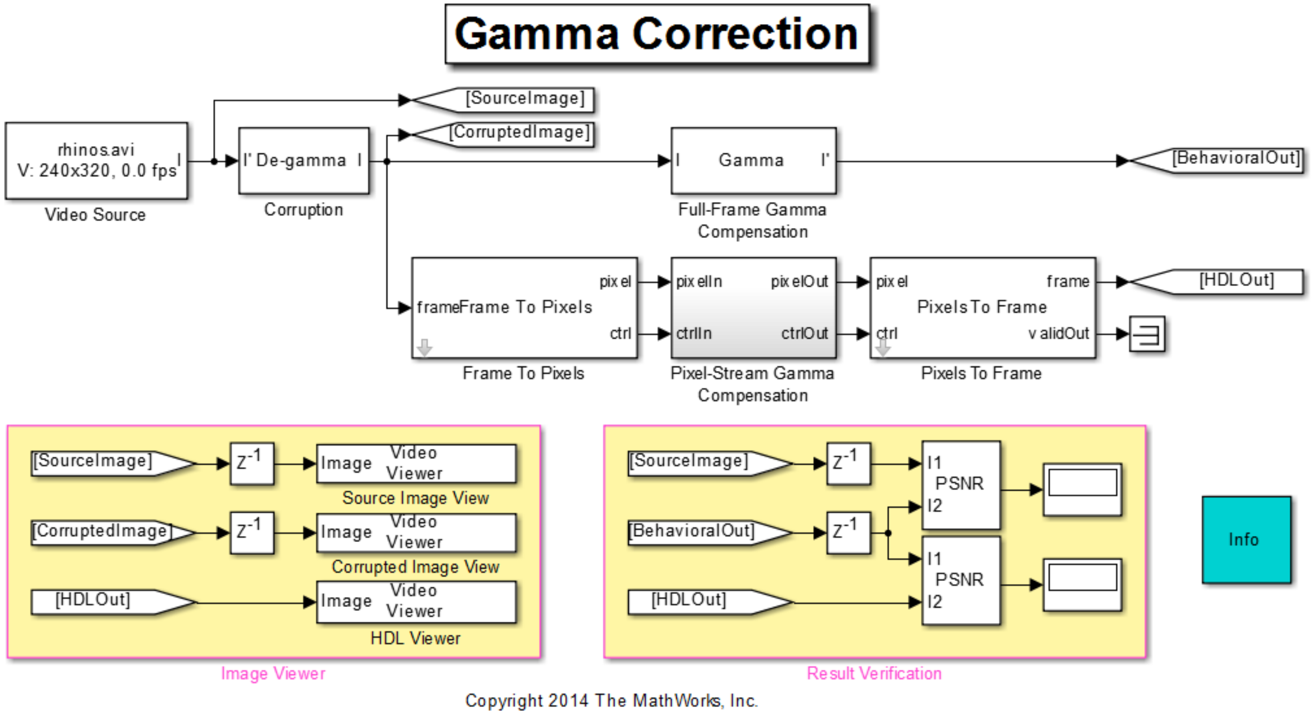
Image Filtering

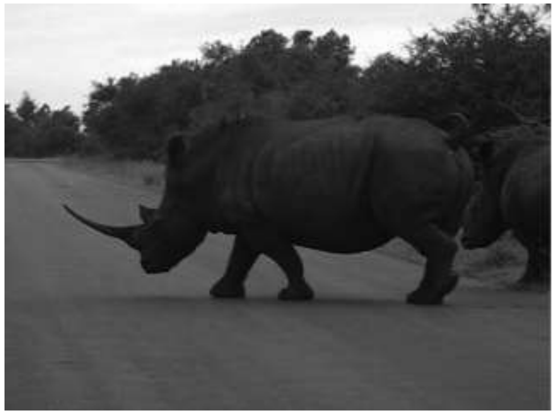
Keywords: Median Filter, Image Filter, PSNR

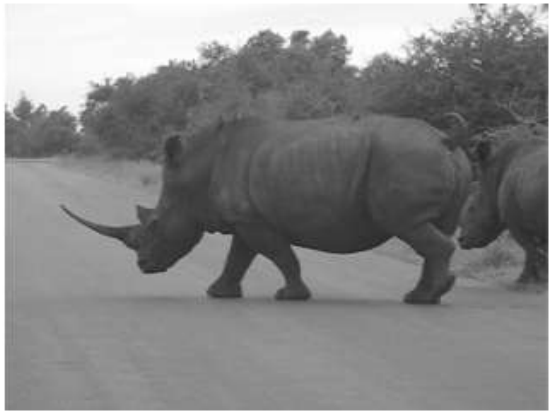


Gamma Correction Example

Keywords: Gamma, PSNR

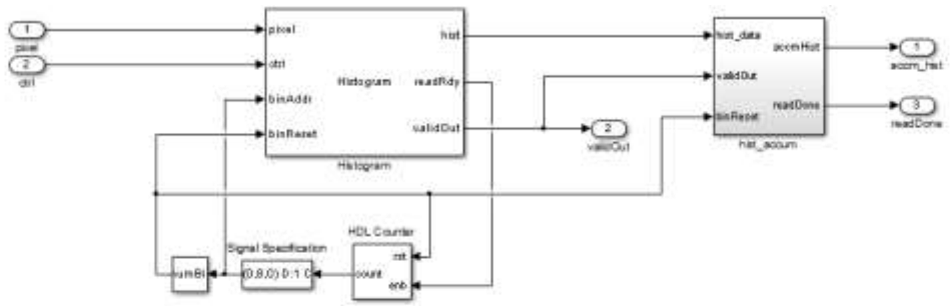


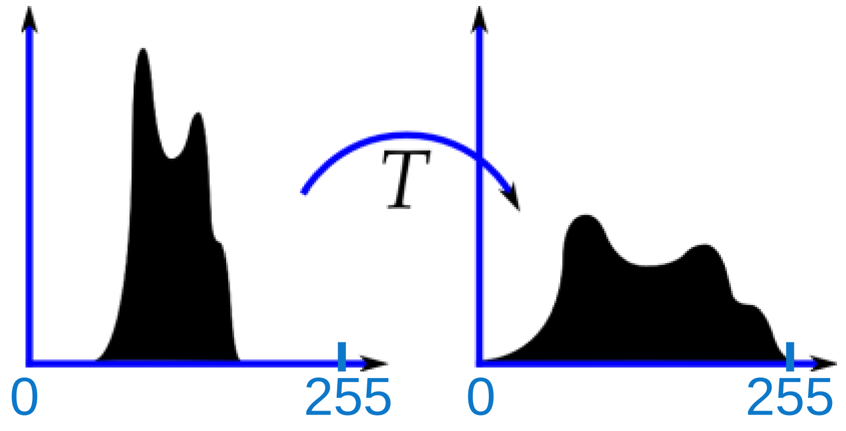


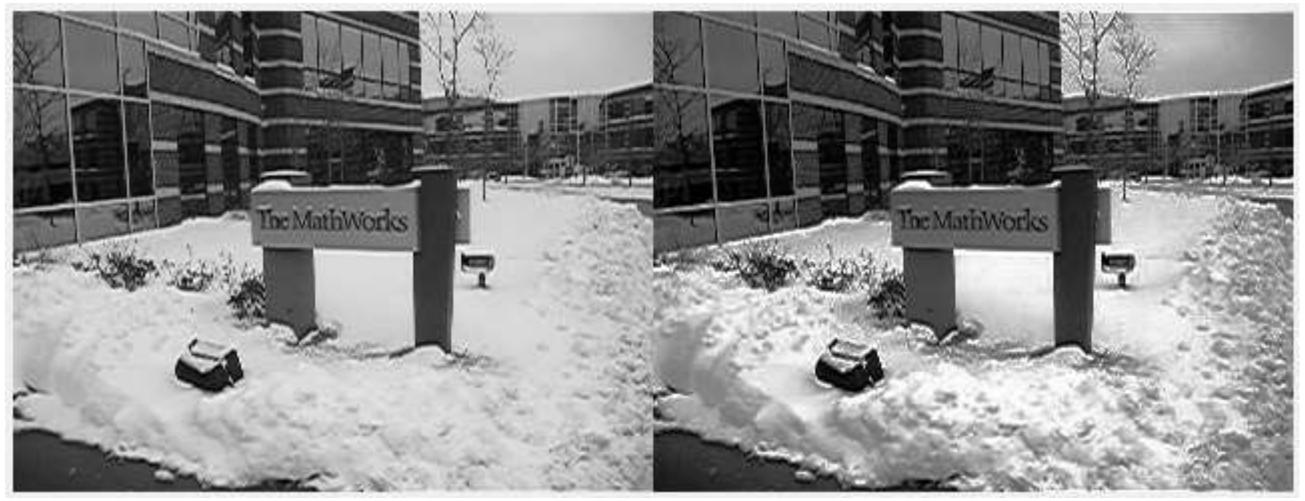


Histogram Equalization

Keywords: Histogram, Linear Equalization, External Frame Delay

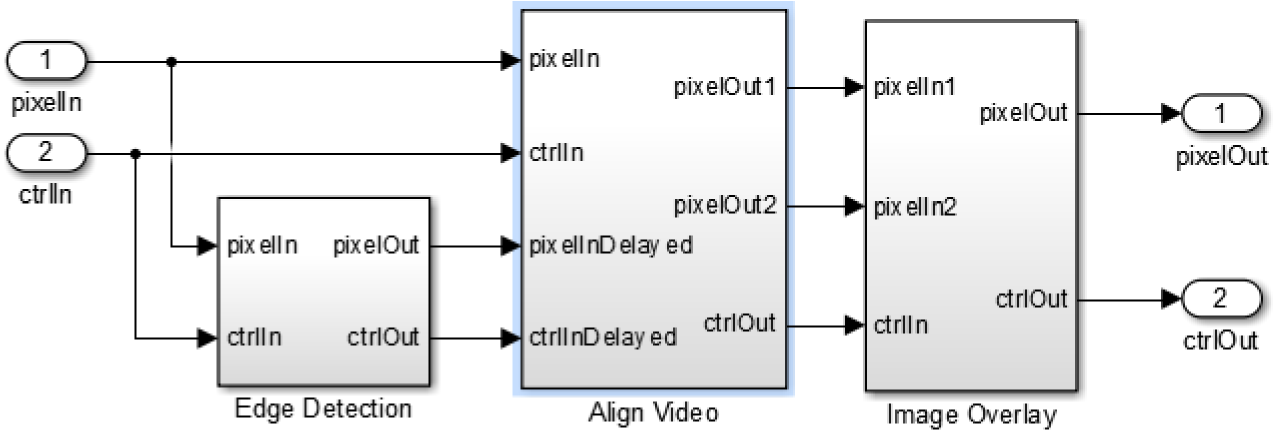


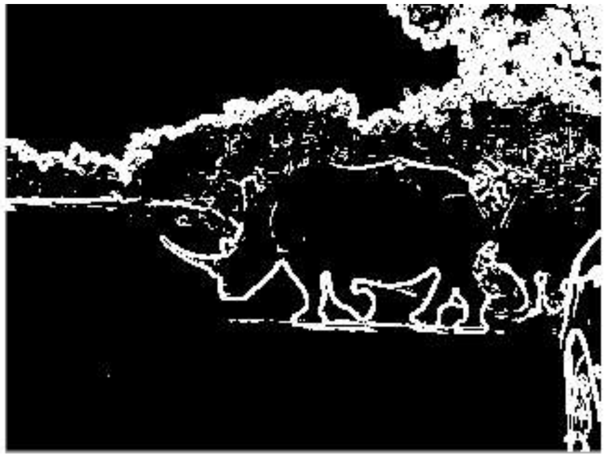


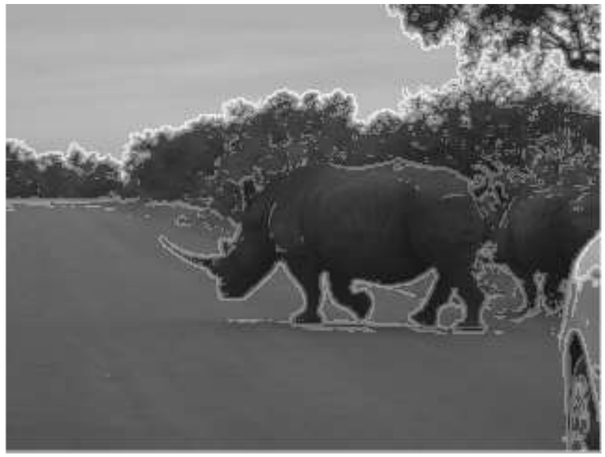


Edge Detection and Image Overlay

Keywords: Sobel, Align Video, Alpha Mix, PSNR

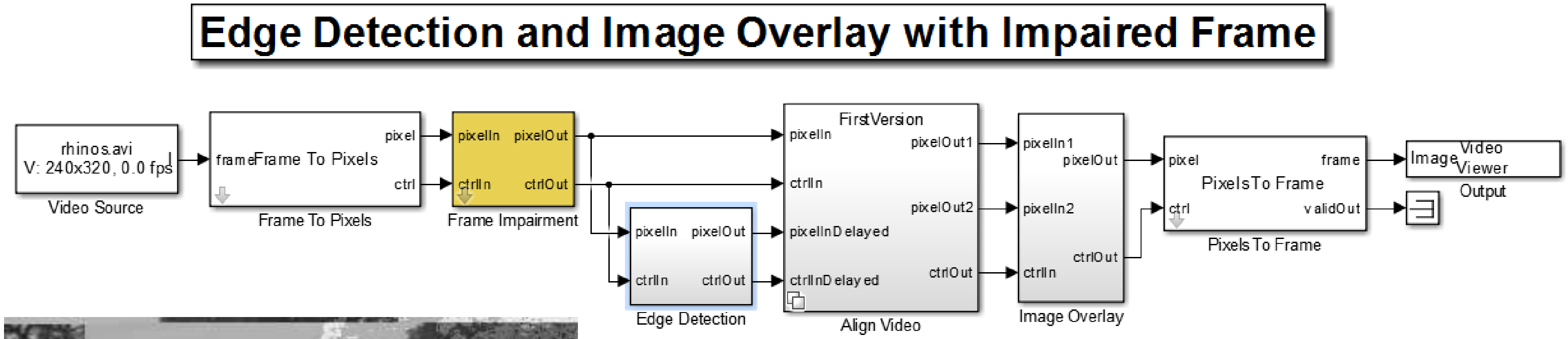




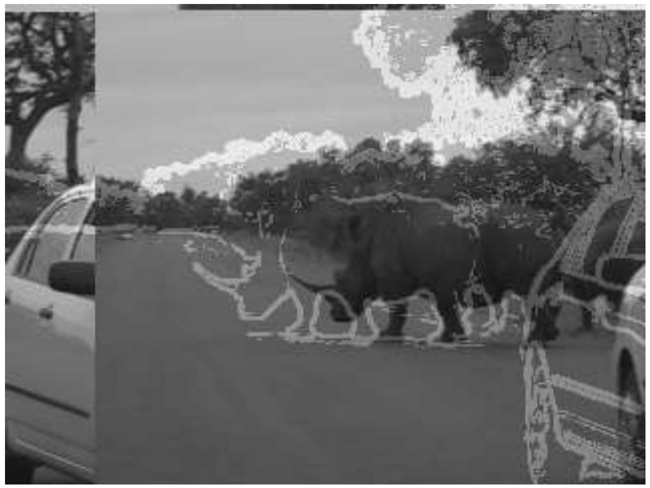


Edge Detection with Impaired Frame

Keywords: Sobel, Align Video, Alpha Mix, PSNR

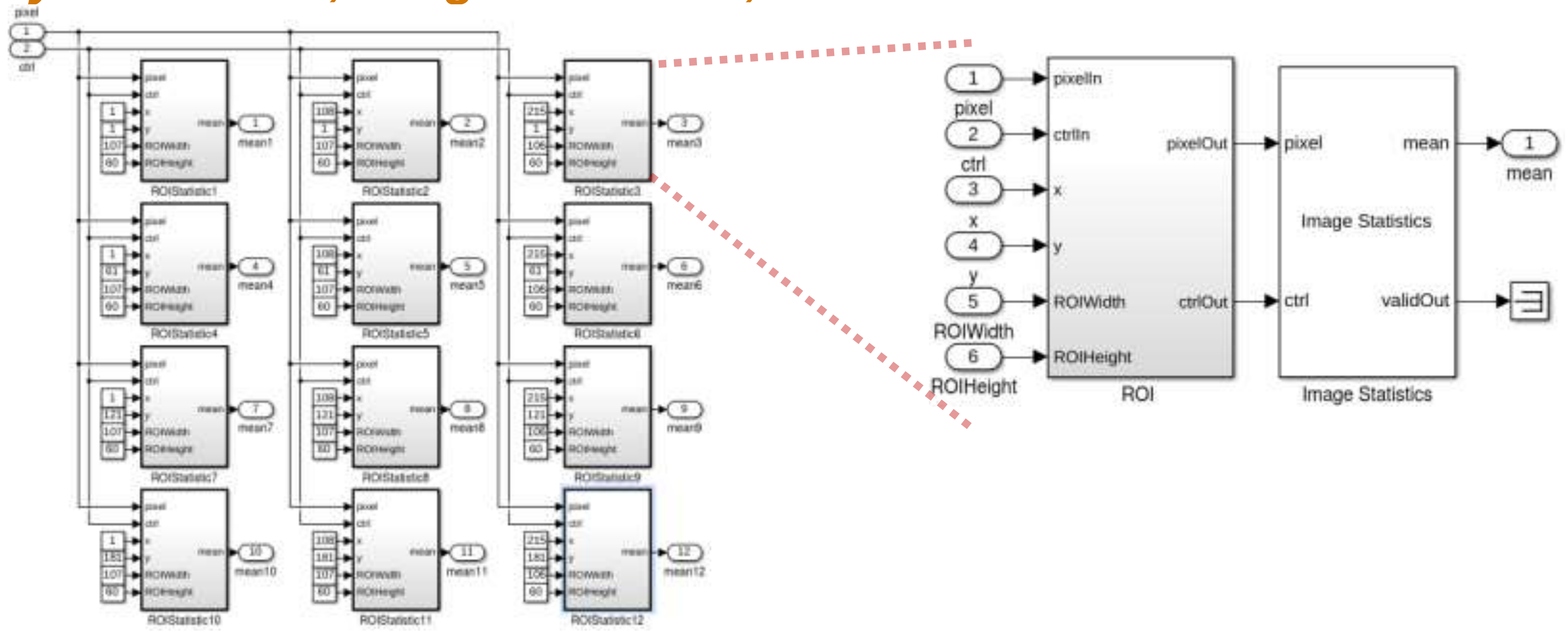


Align Video is implemented as a variant subsystem.
To simulate using the **FirstVersion** variant, type **VERSION=1** at MATLAB prompt.
To simulate using the **SecondVersion** variant, type **VERSION=2** at MATLAB prompt.



Multi-Zone Metering

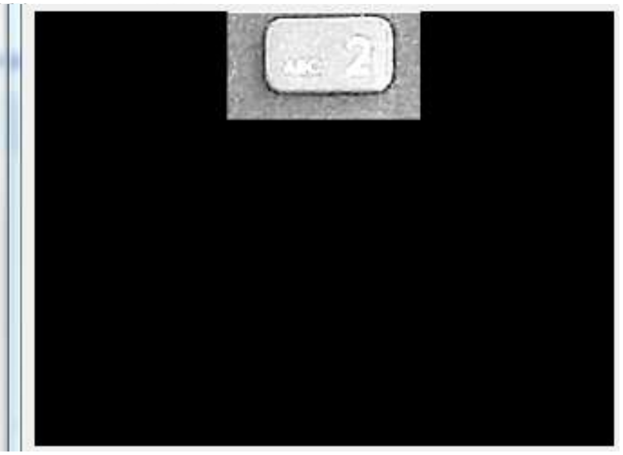
Keywords: ROI, Image Statistics, Mean



Source

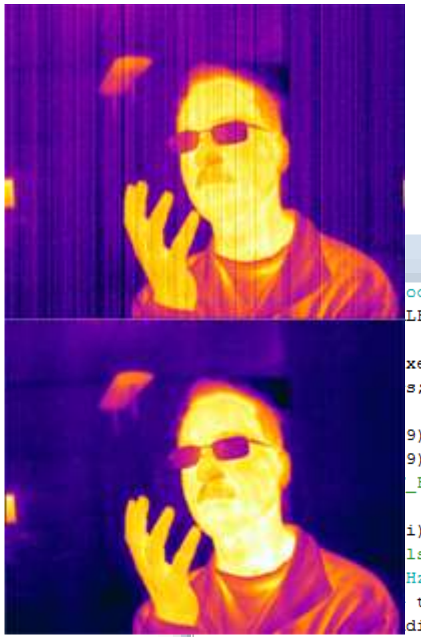
Mask

Selected ROI

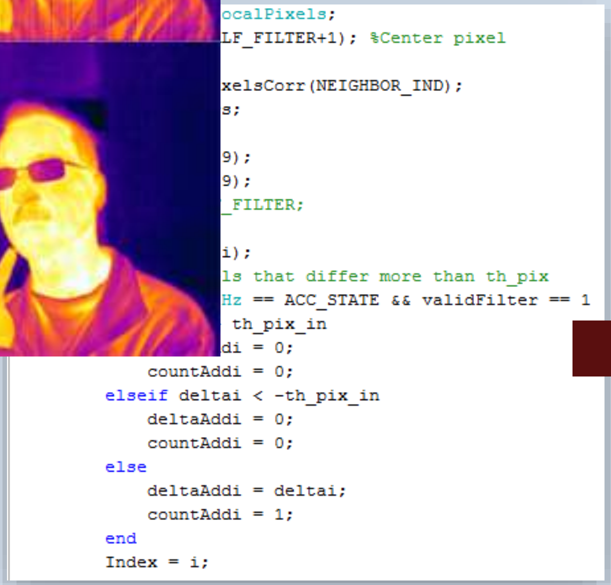


HDL Coder

Provides VHDL and Verilog code generation for MATLAB and Simulink



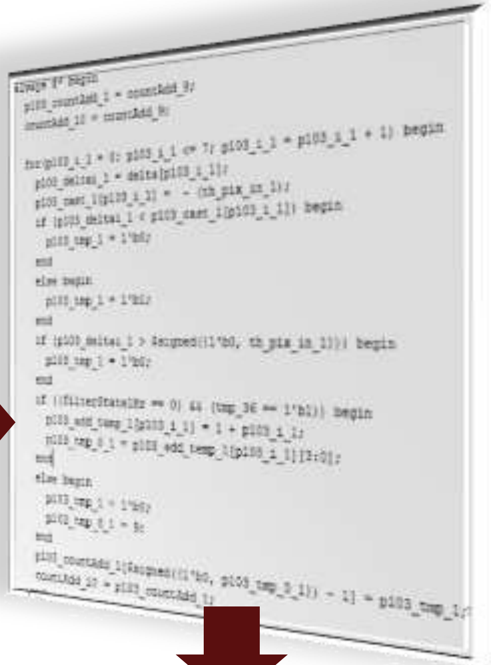
FLIR Systems



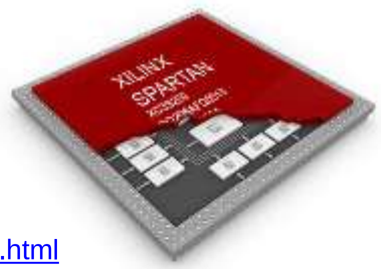
```
ocalPixels;
LF_FILTER+1); %Center pixel

xelsCorr(NEIGHBOR_IND);
s;

9);
9);
_FILTER;

i);
ls that differ more than th_pix
Hz == ACC_STATE && validFilter == 1
 th_pix_in
di = 0;
        countAddi = 0;
    elseif deltai < -th_pix_in
        deltaAddi = 0;
        countAddi = 0;
    else
        deltaAddi = deltai;
        countAddi = 1;
    end
    Index = i;
```

MATLAB Code
10X more concise

HDL Coder



```
always @* begin
 p103_countAdd_1 = countAdd_9;
 countAdd_10 = countAdd_9;

 for(p103_i_1 = 0; p103_i_1 <= 7; p103_i_1 = p103_i_1 + 1) begin
  p103_deltai_1 = delta[p103_i_1];
  p103_cast_1[p103_i_1] = - (th_pix_in_1);
  if (p103_deltai_1 < p103_cast_1[p103_i_1]) begin
   p103_tmp_1 = 1'b0;
  end
  else begin
   p103_tmp_1 = 1'b1;
  end
  if (p103_deltai_1 > $signed({1'b0, th_pix_in_1})) begin
   p103_tmp_1 = 1'b0;
  end
  if ((filterStateHz == 0) && (tmp_36 == 1'b1)) begin
   p103_add_temp_1[p103_i_1] = 1 + p103_i_1;
   p103_tmp_0_1 = p103_add_temp_1[p103_i_1][3:0];
  end
  else begin
   p103_tmp_1 = 1'b0;
   p103_tmp_0_1 = 0;
  end
  p103_countAdd_1[$signed({1'b0, p103_tmp_0_1}) - 1] = p103_tmp_1;
  countAdd_10 = p103_countAdd_1;
```

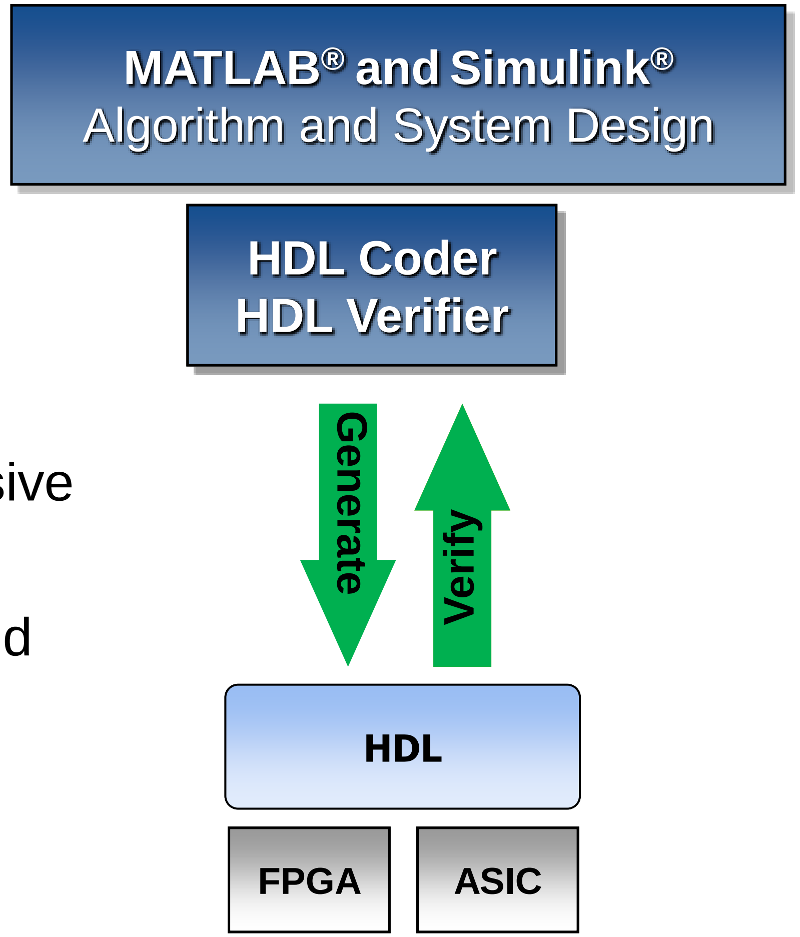


60% reduction in time to produce a working prototype

Reference: mathworks.com/company/user_stories/FLIR-Accelerates-Development-of-Thermal-Imaging-FPGA.html

HDL Products Key Features

- Code Generation
  - Target-independent HDL Code
  - IEEE 1376 compliant VHDL®
  - IEEE 1364-2001 compliant Verilog®
- Verification
  - Generate HDL test-bench
  - Co-simulate with ModelSim and Incisive
- Design automation
  - Synthesize using integrated Xilinx and Altera synthesis tool interface
  - Optimize for area-speed
  - Program Xilinx and Altera boards



HDL Coder : Key Features and Options

Area Optimizations

- Simulink
  - Streaming
  - Sharing
  - Line buffers as RAMs
  - RAM Fusion
  - Architecture Flattening
- MATLAB
  - RAM Mapping
  - Loop Streaming
  - Resource Sharing
  - CSD/FCSD

Speed Optimizations

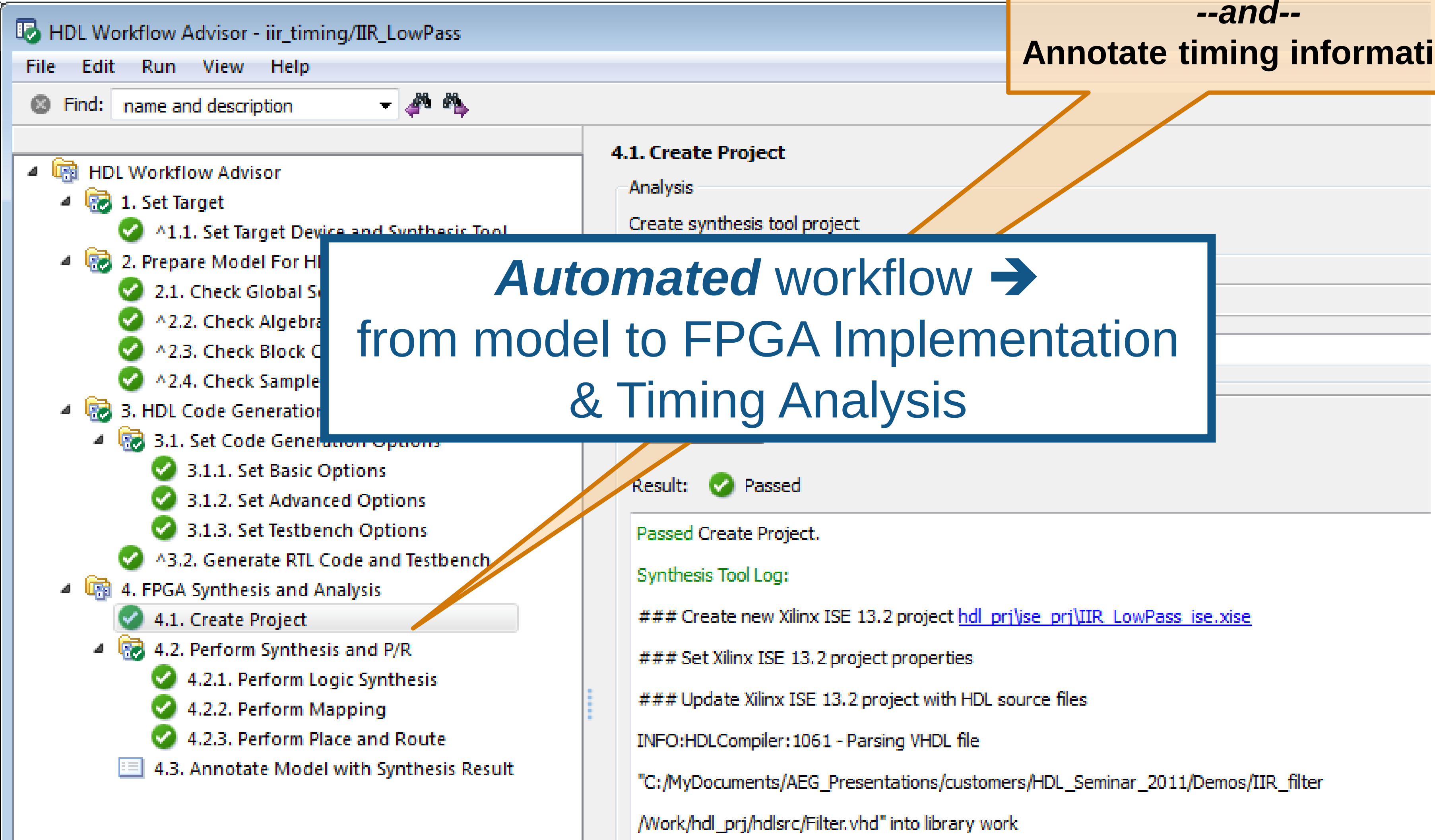
- Simulink
  - Input/Output pipelining
  - Distributed Pipelining
  - Hierarchical Dist. Pipelining
  - Constrained Pipelining
  - Back-Annotation
- MATLAB
  - Input/Output pipelining
  - Distributed pipelining
  - Loop Unrolling

Validation and Verification

- Automatic Delay Balancing
- Validation model generation

Hardware Design Solution:
HDL Workflow Advisor

Create FPGA project
Run P&R
--and--
Annotate timing information



HDL Workflow Advisor - iir_timing/IIR_LowPass

File Edit Run View Help

Find: name and description

- HDL Workflow Advisor
  - 1. Set Target
    - ^1.1. Set Target Device and Synthesis Tool
  - 2. Prepare Model For HI
    - 2.1. Check Global S
    - ^2.2. Check Algebra
    - ^2.3. Check Block C
    - ^2.4. Check Sample
  - 3. HDL Code Generation
    - 3.1. Set Code Generation Options
      - 3.1.1. Set Basic Options
      - 3.1.2. Set Advanced Options
      - 3.1.3. Set Testbench Options
    - ^3.2. Generate RTL Code and Testbench
  - 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

4.1. Create Project

Analysis

Create synthesis tool project

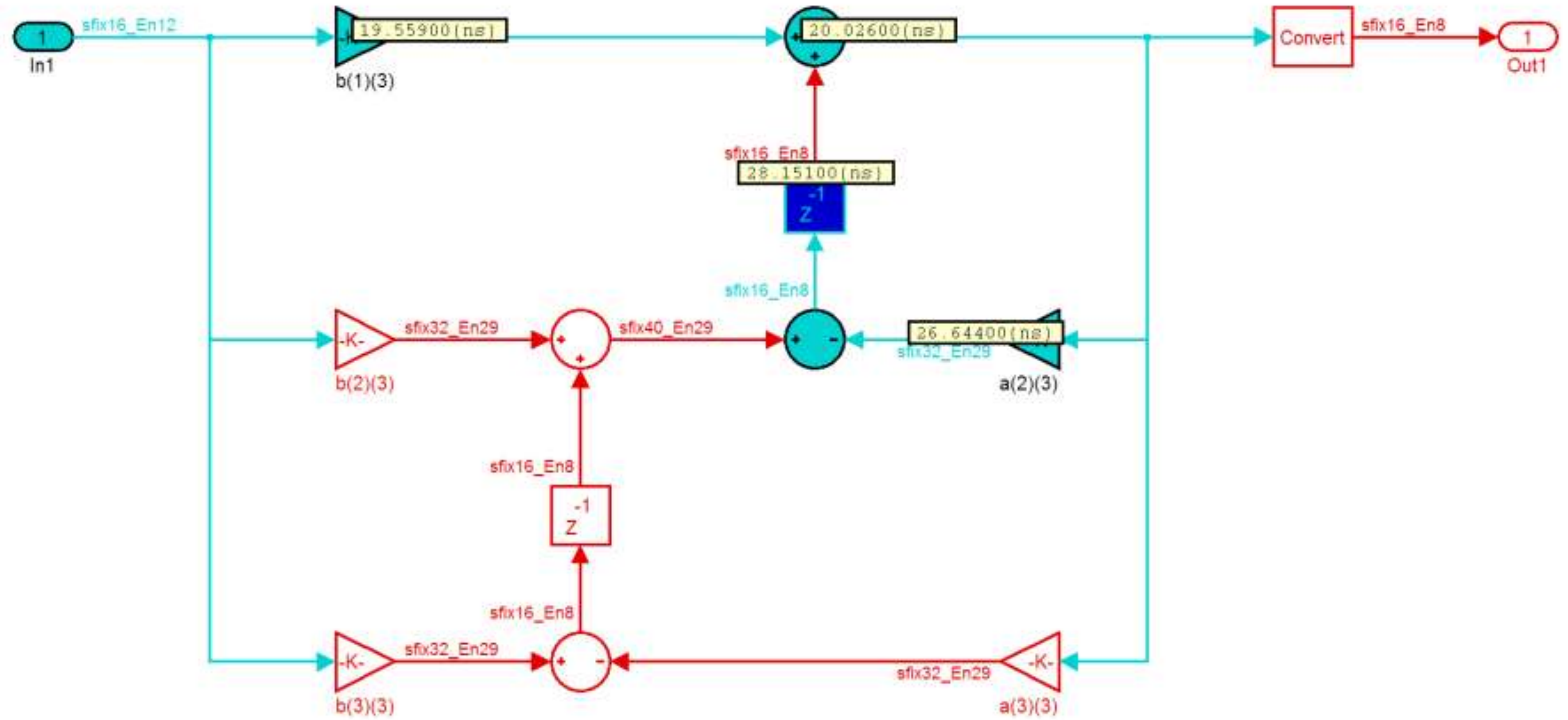
Result: ✓ Passed

```
Passed Create Project.

Synthesis Tool Log:

### Create new Xilinx ISE 13.2 project hdl_prj\ise_prj\IIR_LowPass_ise.xise

### Set Xilinx ISE 13.2 project properties

### Update Xilinx ISE 13.2 project with HDL source files

INFO:HDLCompiler:1061 - Parsing VHDL file

"C:/MyDocuments/AEG_Presentations/customers/HDL_Seminar_2011/Demos/IIR_filter

/Work/hdl_prj/hdlsrc/Filter.vhd" into library work
```

Automated **workflow ➔**
from model to FPGA Implementation
& Timing Analysis

Identifying the critical path

Integrating with P&R Timing Analysis

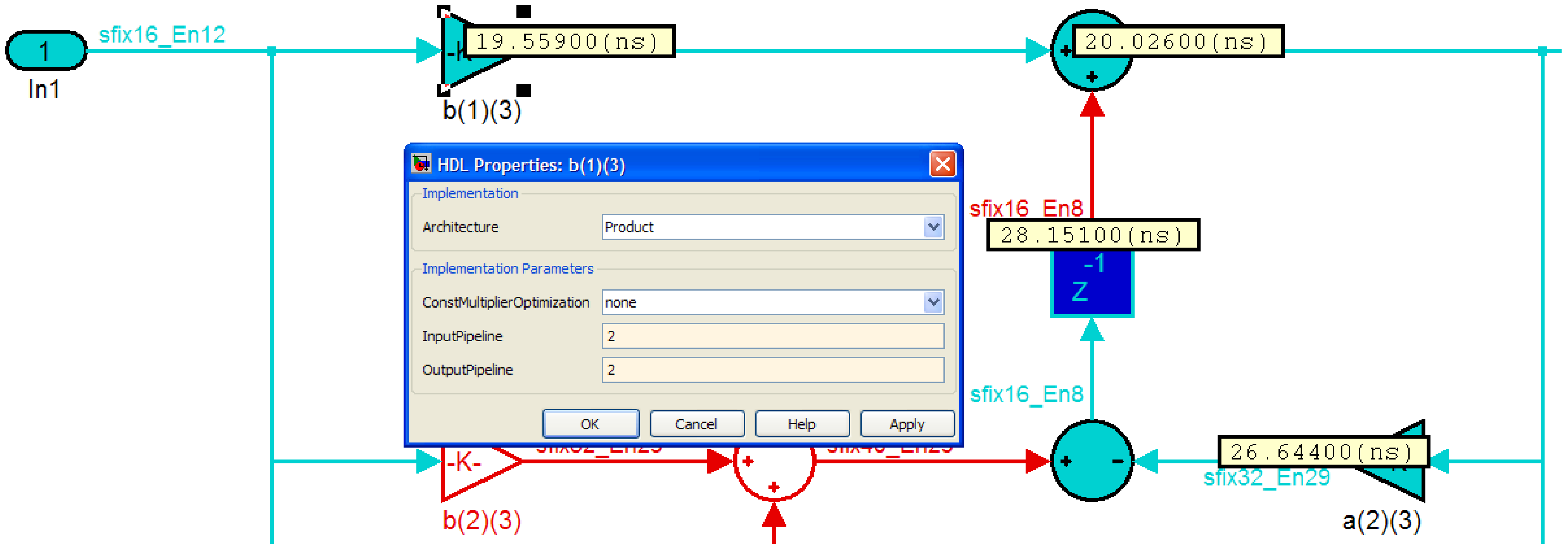


Critical Path highlighting:

- Visual representation of critical path in your model
- Easier to identify bottlenecks of your model

Meeting Timing Constraint

Distributed Pipelining



HDL Properties: b(1)(3)

Implementation

Architecture: Product

Implementation Parameters

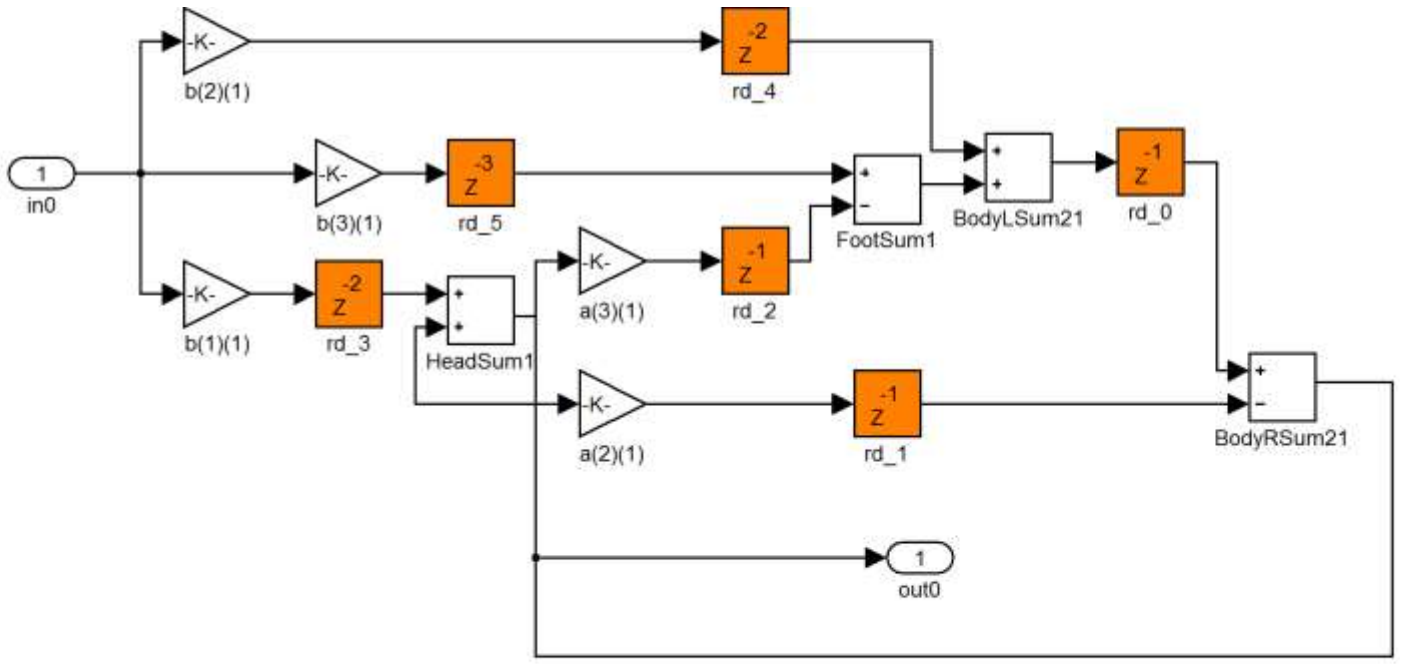
ConstMultiplierOptimization: none

InputPipeline: 2

OutputPipeline: 2

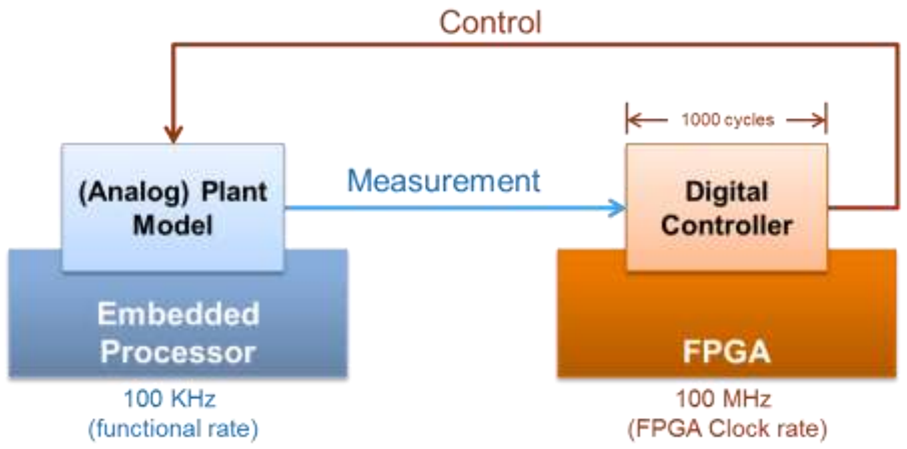
Distributed Pipelining

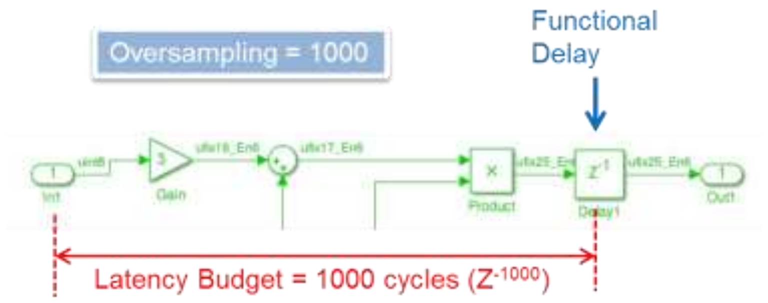
Speed Optimization



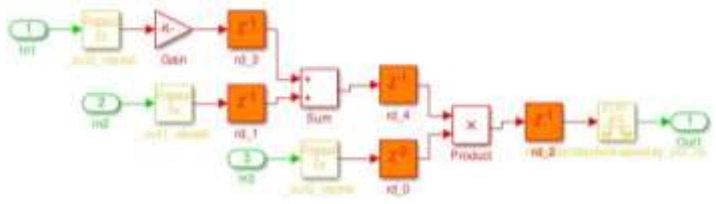
- Distributed pipelining (model retiming)
- Automatic delay compensation where needed
- Constrained retiming gives you more influence

Clock-Rate Pipelining

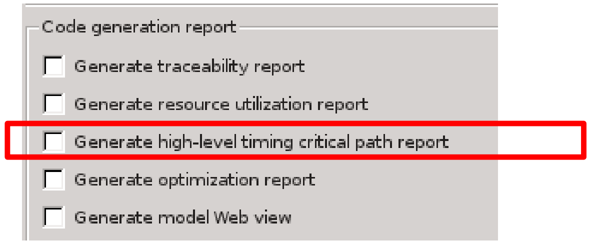




Clock-Rate Pipelining
Exposes latency budget



Critical Path Estimation



Code generation report

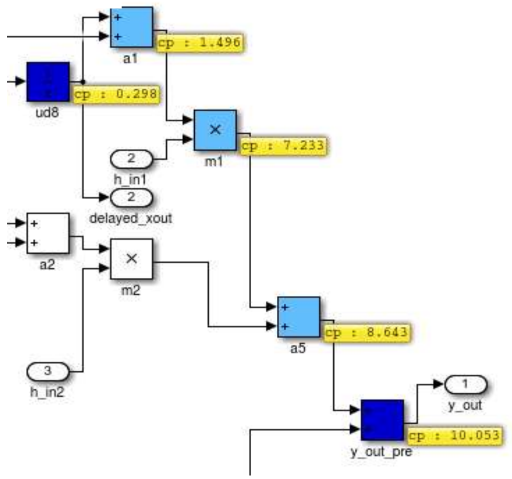
- [ ] Generate traceability report
- [ ] Generate resource utilization report
- [ ] Generate high-level timing critical path report
- [ ] Generate optimization report
- [ ] Generate model Web view

Generate HDL code

hdlset_param(model, 'CriticalPathEstimation', {'on'|'off'})

```
### Estimated critical path for design: hdl prj/hdlsrc/mulfloat/criticalPathEstimated.m
```

Critical path highlighted on generated model



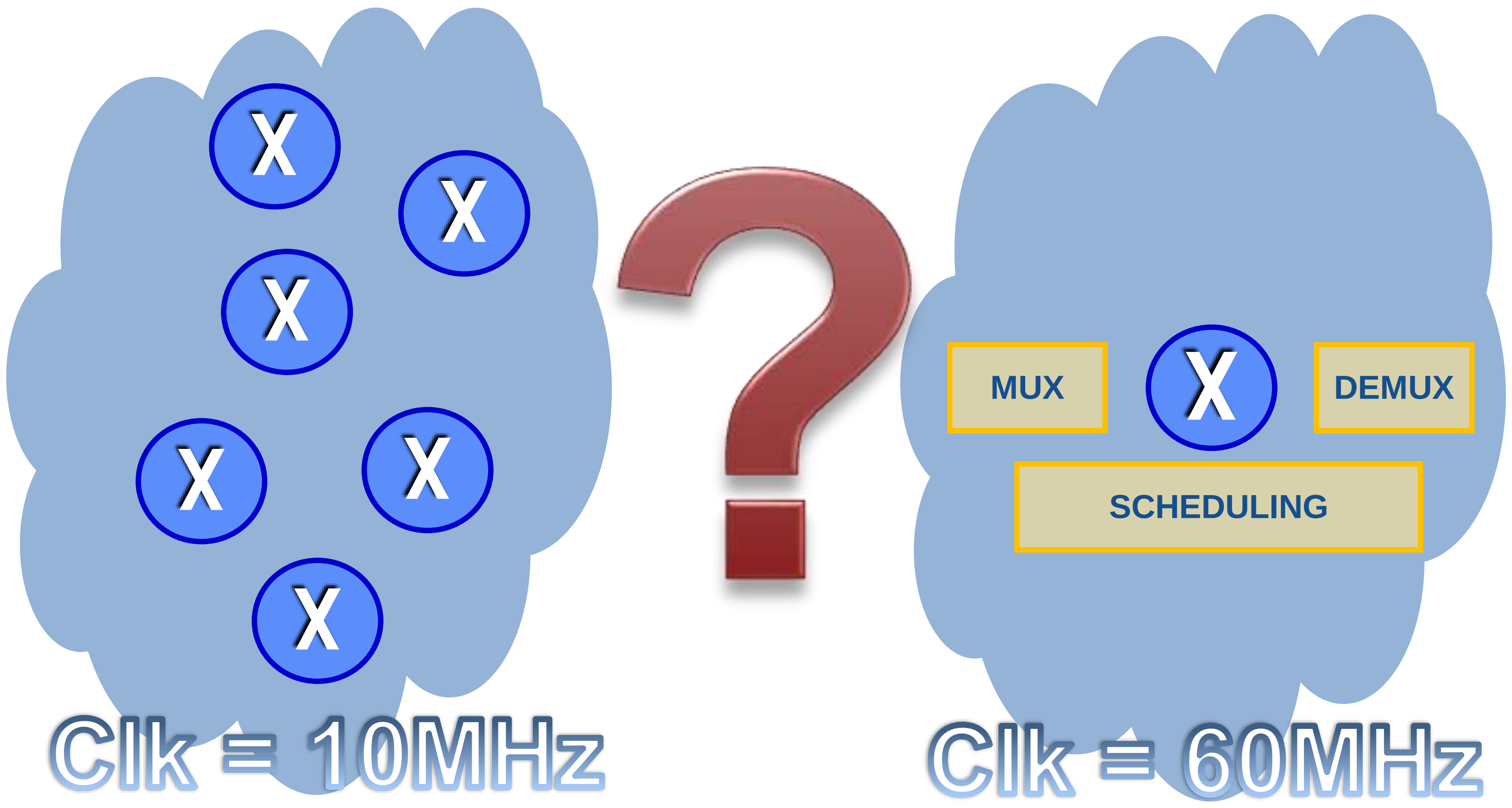
Supported Targets

XILINX
ARTIX KINTEX VIRTEX ZYNQ
VIVADO

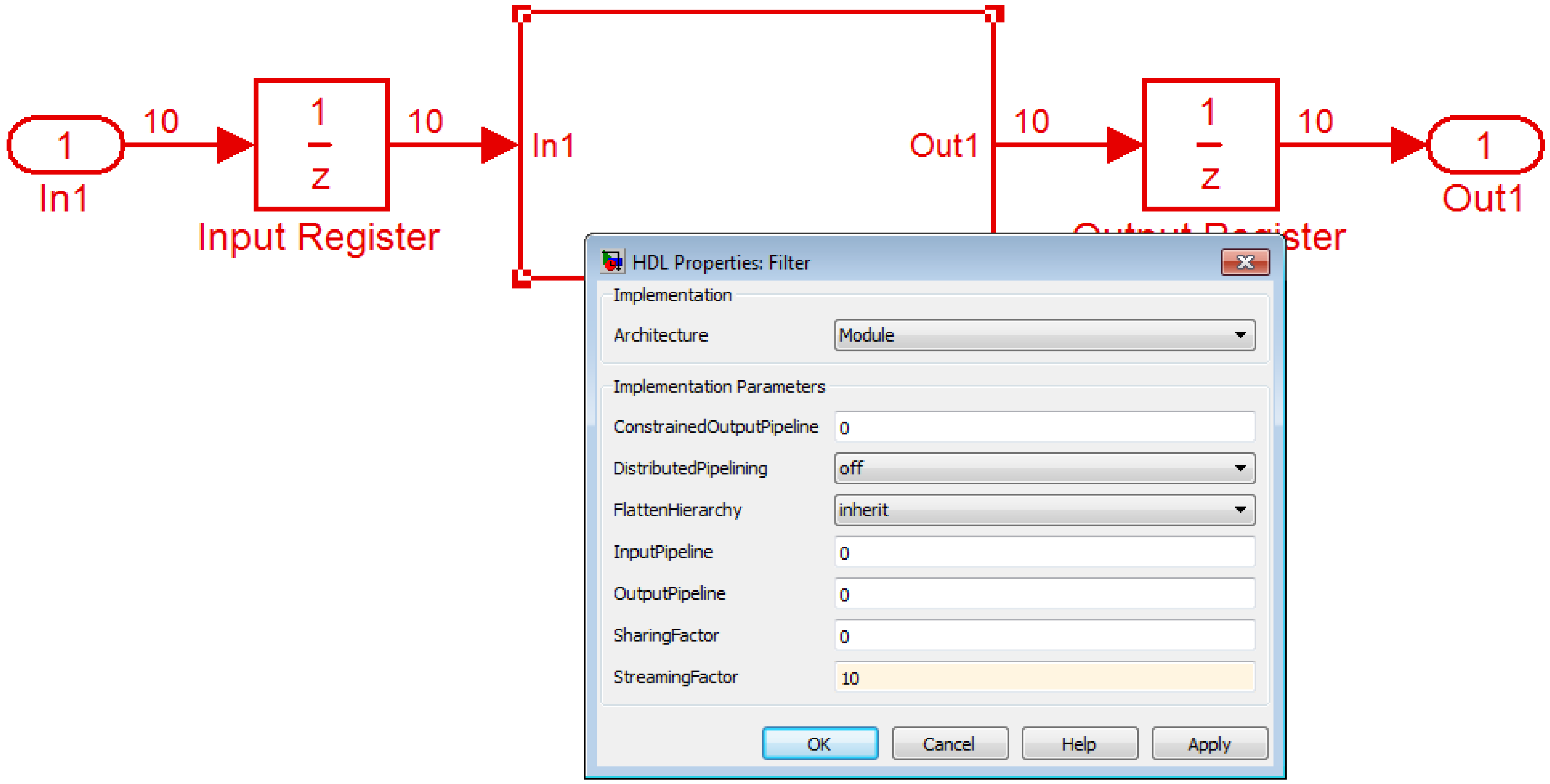
ALTERA
Cyclone V

Meeting Resource Constraint

Area Optimization



Hardware Design Solution:
Resource Sharing and Streaming



HDL Properties: Filter

Implementation

Architecture	Module

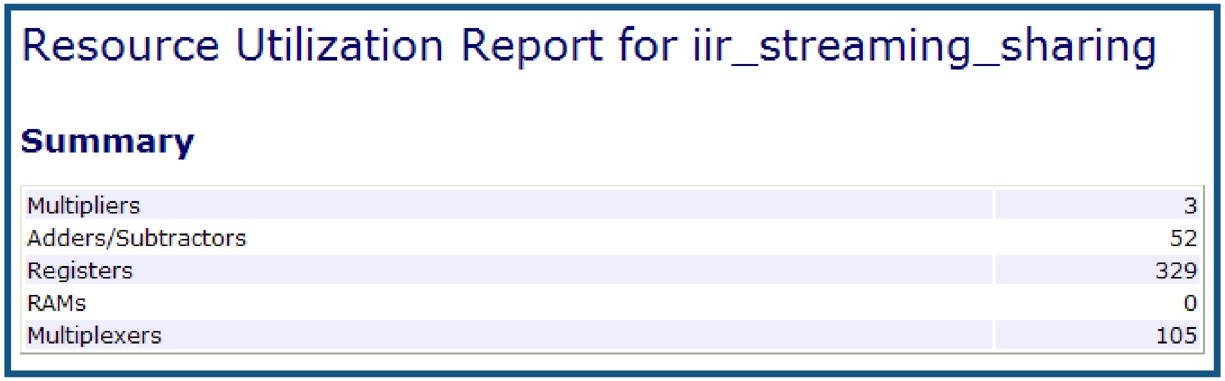
Implementation Parameters

ConstrainedOutputPipeline	0
DistributedPipelining	off
FlattenHierarchy	inherit
InputPipeline	0
OutputPipeline	0
SharingFactor	0
StreamingFactor	10

Resource Sharing and Streaming

Area Optimization

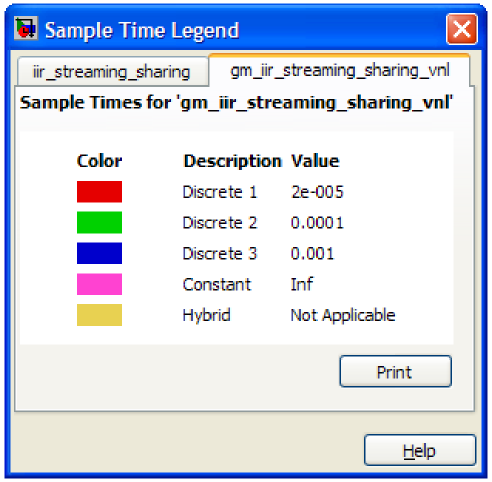
- Easily share **multipliers** and **identical subsystems**
- Direct feedback through resource utilization report
- Prove correctness through validation models



Resource Utilization Report for iir_streaming_sharing

Summary

Multipliers	3
Adders/Subtractors	52
Registers	329
RAMs	0
Multiplexers	105

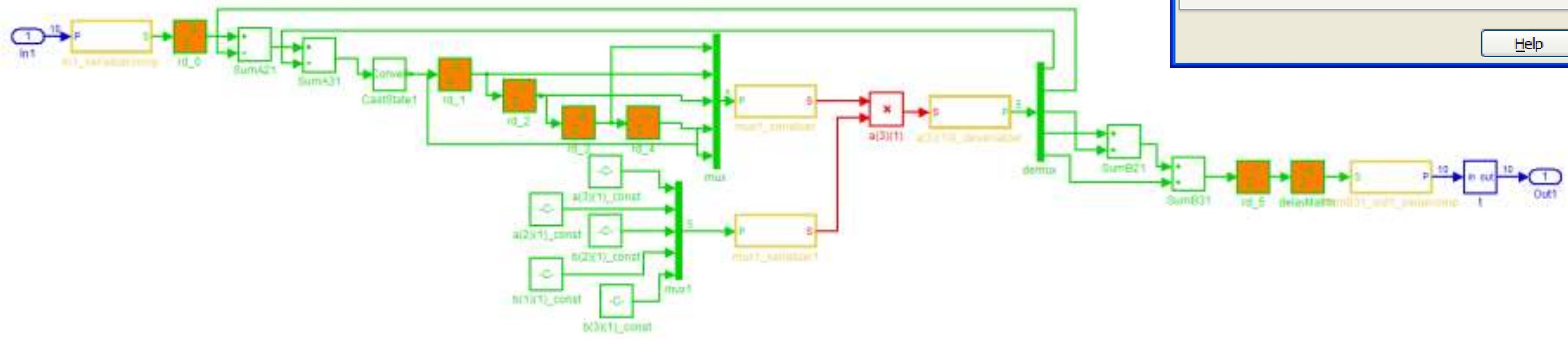


Sample Time Legend

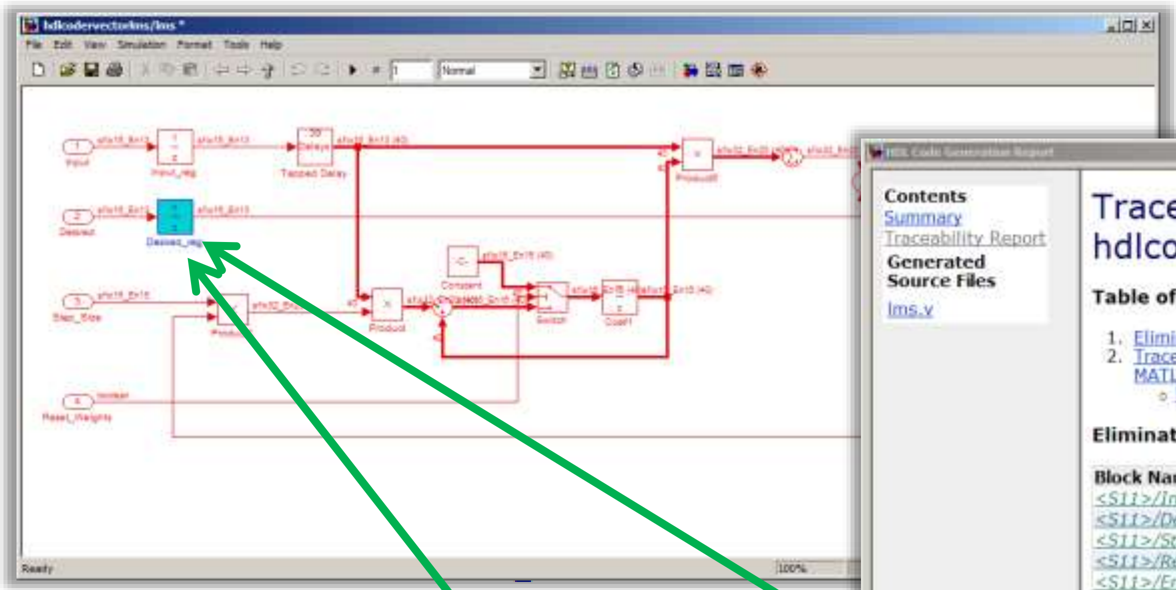
iir_streaming_sharing | gm_iir_streaming_sharing_vnl

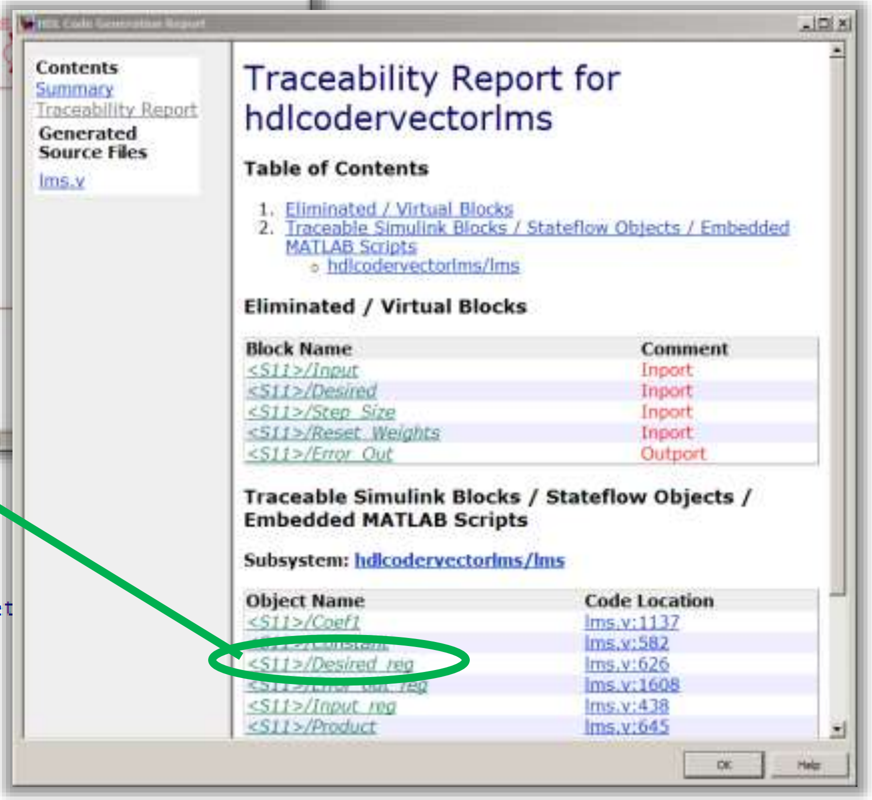
Sample Times for 'gm_iir_streaming_sharing_vnl'

Color	Description	Value
🟥	Discrete 1	2e-005
🟩	Discrete 2	0.0001
🟦	Discrete 3	0.001
🟪	Constant	Inf
🟨	Hybrid	Not Applicable



Traceability Between Model and Code





Contents
Summary
Traceability Report
Generated Source Files
lms.v

Traceability Report for hdlcodervectorlms

Table of Contents

1. Eliminated / Virtual Blocks
2. Traceable Simulink Blocks / Stateflow Objects / Embedded MATLAB Scripts
   - hdlcodervectorlms/lms

Eliminated / Virtual Blocks

Block Name	Comment
<S11>/Input	Inport
<S11>/Desired	Inport
<S11>/Step_Size	Inport
<S11>/Reset_Weights	Inport
<S11>/Error_Out	Outport

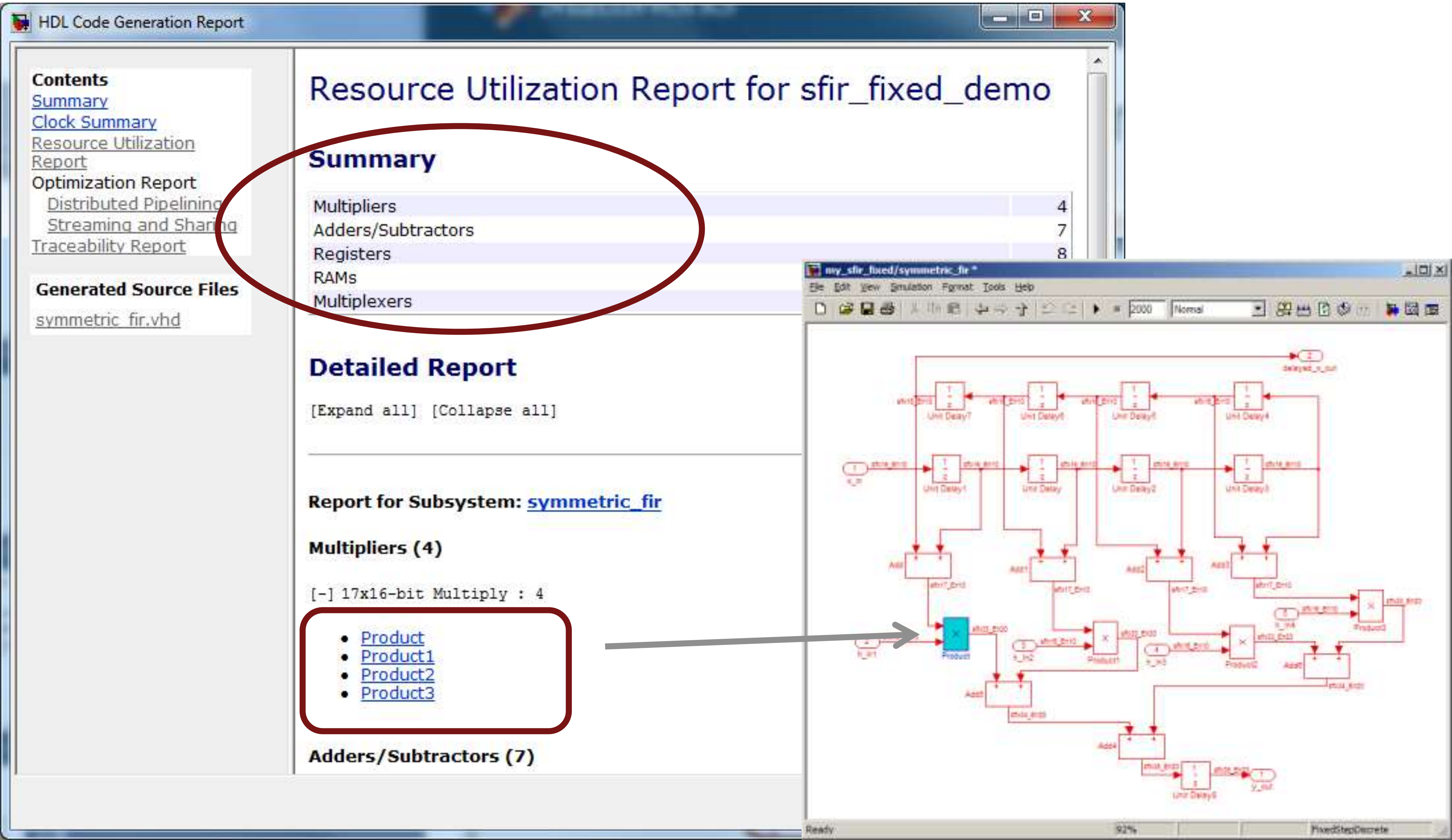
Traceable Simulink Blocks / Stateflow Objects / Embedded MATLAB Scripts

Subsystem: hdlcodervectorlms/lms

Object Name	Code Location
<S11>/Coef1	lms.v:1137
<S11>/Constant	lms.v:582
<S11>/Desired_reg	lms.v:626
<S11>/Error_Out_reg	lms.v:1608
<S11>/Input_reg	lms.v:438
<S11>/Product	lms.v:645

```
623
624
625
626     // <S11>/Desired_reg
627     always @(posedge clk or posedge reset
628       begin : Desired_reg_process
629         if (reset)
630           Desired_reg_out1 <= 0;
631         else
632           if (enb)
633             Desired_reg_out1 <= Desired;
634
635       end
```

Resource Utilization Estimation



HDL Code Generation Report

Generated Source Files

symmetric_fir.vhd

Resource Utilization Report for sfir_fixed_demo

Summary

Multipliers	4
Adders/Subtractors	7
Registers	8
RAMs	
Multiplexers	

Detailed Report

[Expand all] [Collapse all]

Report for Subsystem: symmetric_fir

Multipliers (4)

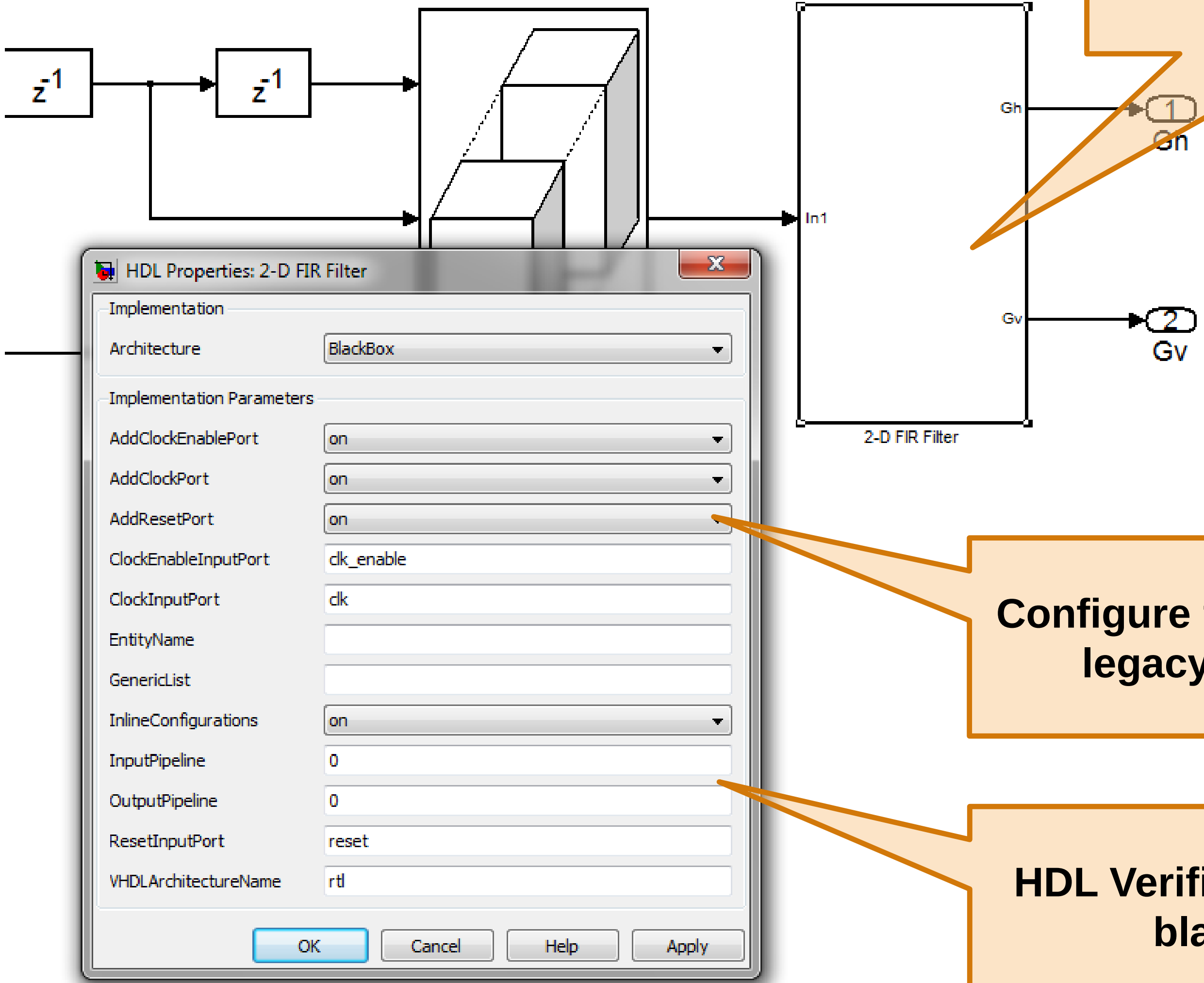
```
[-] 17x16-bit Multiply : 4
```

- Product
- Product1
- Product2
- Product3

Adders/Subtractors (7)

Integrating Legacy HDL Code

HDL Supported Blocks



HDL Properties: 2-D FIR Filter

Implementation

Architecture	BlackBox

Implementation Parameters

AddClockEnablePort	on
AddClockPort	on
AddResetPort	on
ClockEnableInputPort	clk_enable
ClockInputPort	clk
EntityName	
GenericList	
InlineConfigurations	on
InputPipeline	0
OutputPipeline	0
ResetInputPort	reset
VHDLArchitectureName	rtl

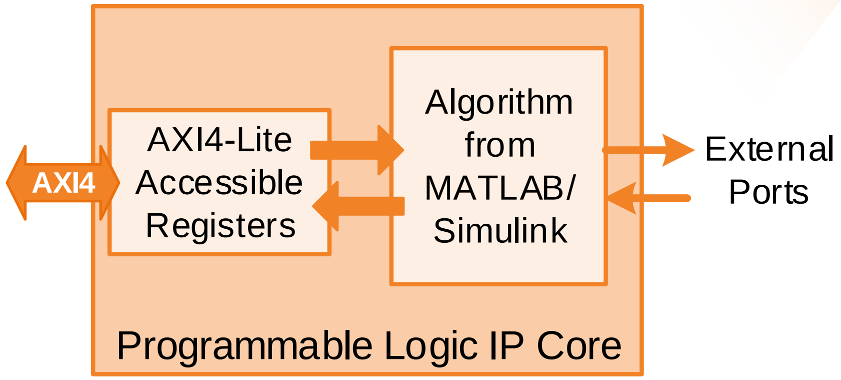
Integrate legacy HDL code in Simulink using black boxes

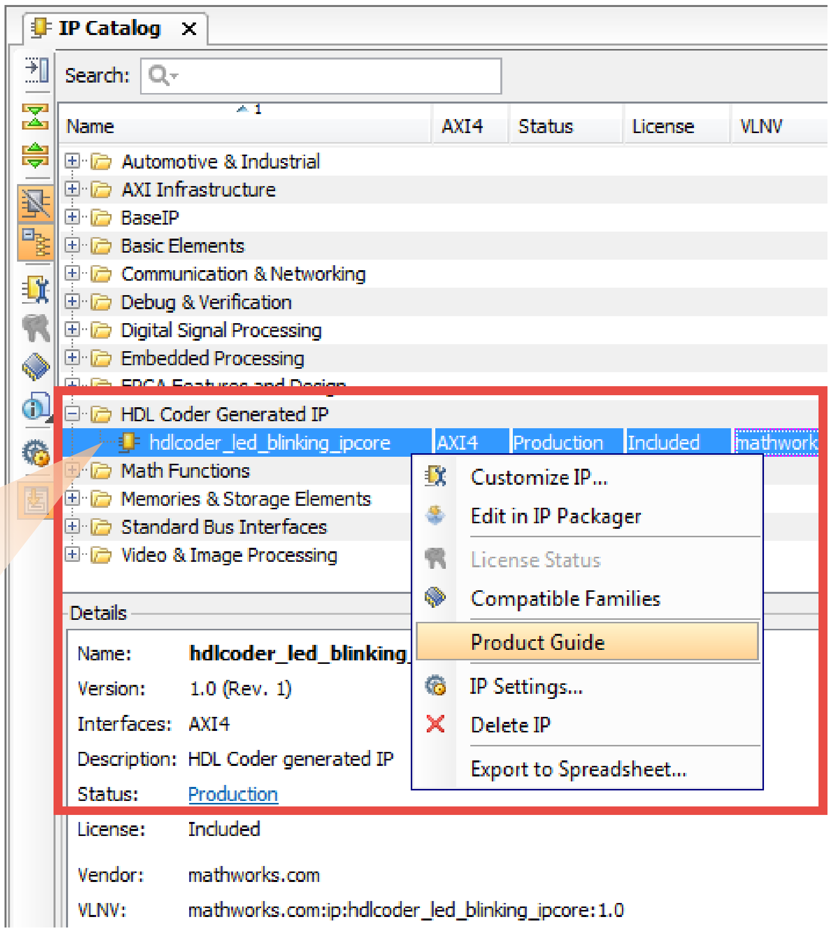
Configure the interface to legacy HDL code

HDL Verifier is a special black box

Vivado IP Core Generation

- Generate sharable and re-usable Vivado IP core from MATLAB/ Simulink HDL Workflow Advisor
- Support AXI4 interfaces to connect FPGA IP core to Zynq ARM processor
- Generate IP core report as IP Product Guide


AXI4
AXI4-Lite Accessible Registers
Algorithm from MATLAB/ Simulink
External Ports
Programmable Logic IP Core

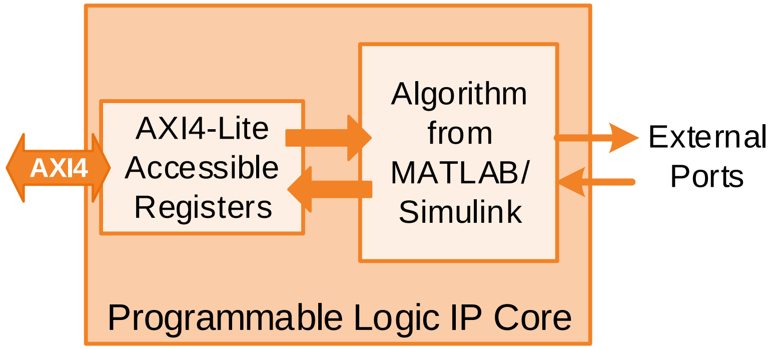


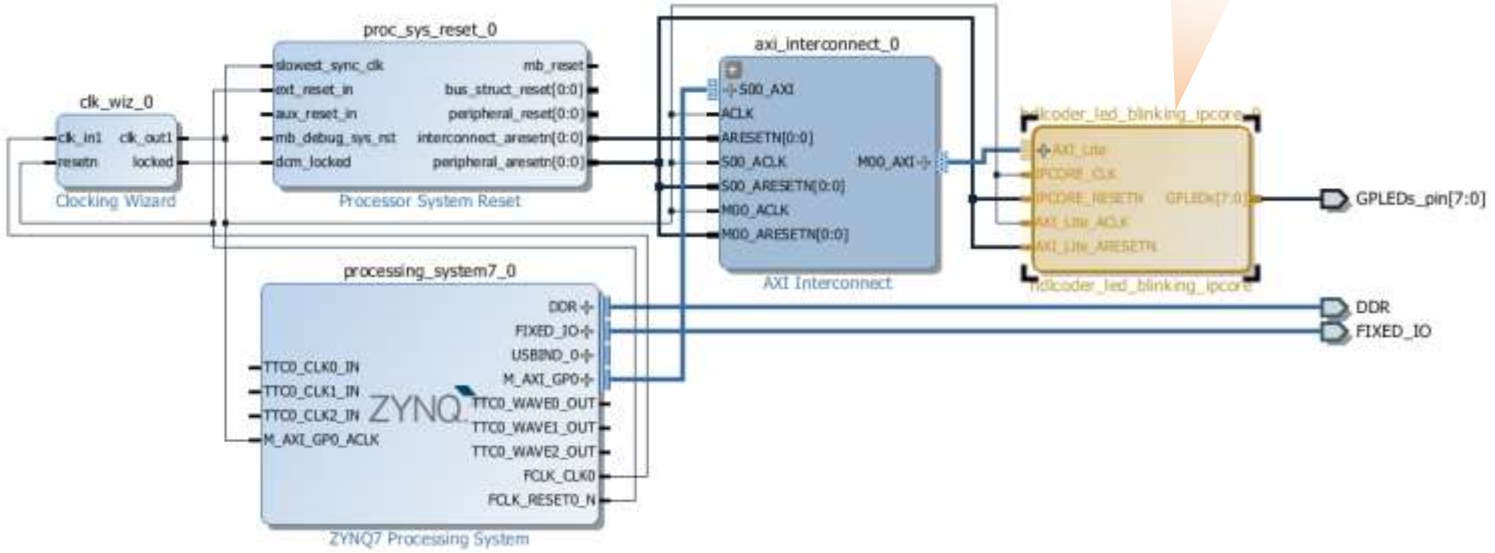
Name: hdlcoder_led_blinking
Version: 1.0 (Rev. 1)
Interfaces: AXI4
Description: HDL Coder generated IP
Status: Production
License: Included
Vendor: mathworks.com
VLNV: mathworks.com:ip:hdlcoder_led_blinking_ipcore:1.0

MathWorks
R2014b

Vivado IP Integrator Support for Zynq

- Integrate Xilinx Vivado IP Integrator tool flow into HDL Workflow Advisor
- Insert the generated IP core into Vivado Zynq system design
- Build and Program Zynq board

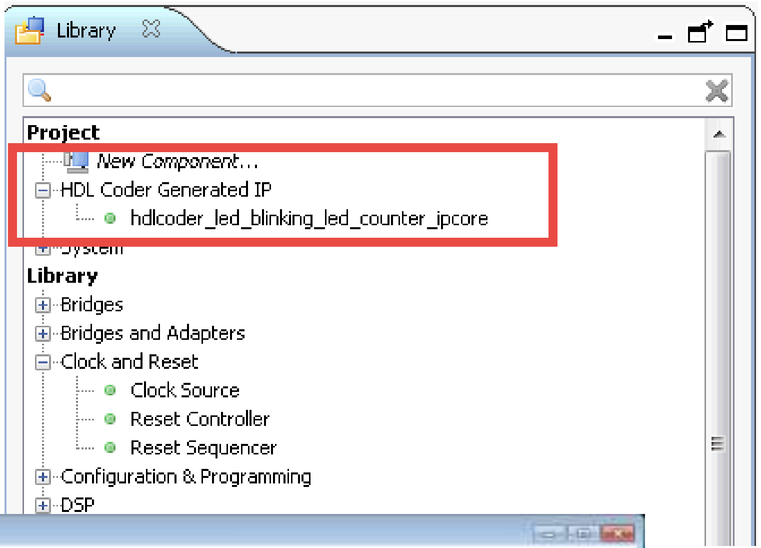

AXI4
AXI4-Lite Accessible Registers
Algorithm from MATLAB/ Simulink
External Ports
Programmable Logic IP Core



R2014b

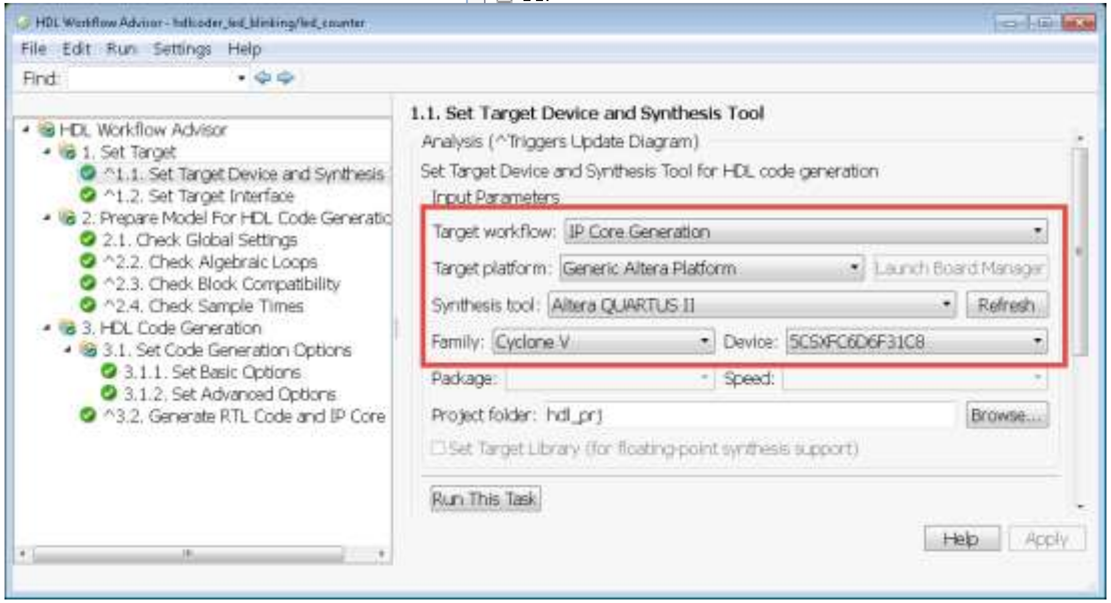
Altera IP Core Generation + QSys Integration

- Generate sharable and re-usable Altera IP from MATLAB/ Simulink HDL WFA
- Support AXI4 interfaces to connect FPGA IP to Altera SoC ARM processor
- Generate IP core report as IP Data Sheet
- QSys integration for rapid prototyping


Library

Project
- New Component…
- HDL Coder Generated IP
  - hdlcoder_led_blinking_led_counter_ipcore
- System

Library
- Bridges
- Bridges and Adapters
- Clock and Reset
  - Clock Source
  - Reset Controller
  - Reset Sequencer
- Configuration & Programming
- DSP


HDL Workflow Advisor - hdlcoder_led_blinking/led_counter

File Edit Run Settings Help

Find:

- HDL Workflow Advisor
  - 1. Set Target
    - ^1.1. Set Target Device and Synthesis
    - ^1.2. Set Target Interface
  - 2. Prepare Model For HDL Code Generatic
    - 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 Advanced Options
    - ^3.2. Generate RTL Code and IP Core

1.1. Set Target Device and Synthesis Tool

Analysis (^Triggers Update Diagram)

Set Target Device and Synthesis Tool for HDL code generation

Input Parameters

Target workflow: IP Core Generation

Target platform: Generic Altera Platform — Launch Board Manager

Synthesis tool: Altera QUARTUS II — Refresh

Family: Cyclone V — Device: 5CSXFC6D6F31C8

Package: — Speed:

Project folder: hdl_prj — Browse...

☐ Set Target Library (for floating-point synthesis support)

Run This Task

Help Apply

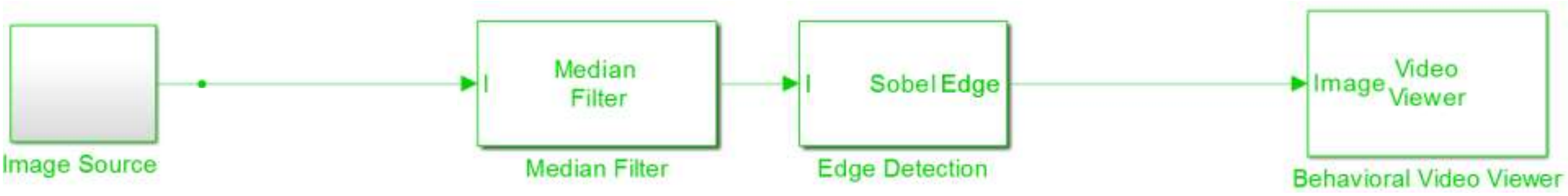
Backup

CVST and VHT, complimentary products

Product details

	VHT	CVST
Image Processing	Pixel based	Frame based
Workflow	System prototyping	Algorithm design
Algorithms	Image filtering Color space conversion Edge detection Statistics & histogram Morphological operations	(Most of the VHT algorithms, plus) Object detection & tracking Feature extraction and matching Stereo vision Camera calibration Image registration
I/O	Frame-to-pixel Pixel-to-frame FIL	File I/O Video display
Code gen	HDL (with HDL Coder)	C (with ML Coder or SL Coder)

MBD Workflow for Embedded Vision (video)



MBD workflow

1. Build behavior model with CVST blocks and simulate
2. Build prototyping model with VHT blocks and simulate
3. Generate HDL code using HDL Coder
4. Run HDL code on FPGA and run testbench in SL (FIL)