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

- Concept Development
- Algorithm Development
- Prototyping
- Architecture design
- Prototyping
- Chip design

Frame based
- Image/Video Engineer

Pixel based
- HW Engineer
Challenges in Design and Prototyping for Video and Image

Modeling video image processing systems
- Pixel-streaming behavior
- Code generation ready model
- Prototyping and concept proofing
- Technology independent code

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

---

**Edge Detection and Image Overlay**

![Diagram](image.png)

Copyright 2014 The MathWorks, Inc.
A Complete Solution for Embedded Vision

Concept Development
Algorithm Development
Prototyping
Architecture design
Prototyping
Chip design

Frame based

Pixel based

Computer Vision System Toolbox
Image Processing Toolbox
Vision HDL Toolbox
HDL Coder
MATLAB Coder
Fixed Pt Designer
HDL Verifier

MATLAB
# A Complete Solution for Embedded Vision

<table>
<thead>
<tr>
<th>Product</th>
<th>Capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vision HDL Toolbox</td>
<td>Design and simulate image processing, video, and computer vision systems for FPGAs and ASICs</td>
</tr>
<tr>
<td>HDL Coder</td>
<td>Provide RTL code and testbench generation capability for the functions and blocks in Vision HDL Toolbox</td>
</tr>
<tr>
<td>HDL Verifier</td>
<td>Provide FPGA-in-the-loop capability for Vision HDL Toolbox</td>
</tr>
<tr>
<td>Computer Vision System Toolbox</td>
<td>Provide frame based computer vision functions and blocks as well as image and video I/O capability</td>
</tr>
<tr>
<td>Image Processing Toolbox</td>
<td>Provide image processing and analysis functions</td>
</tr>
</tbody>
</table>
Model-Based Design For Embedded Vision
From Concept to Production

- Automate regression testing
- Detect design errors
- Support certification and standards
- Generate efficient code
- Explore and optimize implementation tradeoffs
- Build behavioral model for fast simulation and testing
- Convert to prototype model for targeting hardware
- Automate regression testing
- Detect design errors
- Support certification and standards

- RESEARCH
- REQUIREMENTS
- DESIGN
- IMPLEMENTATION
- INTEGRATION

- Testbench & Algorithms
- Behavioral Model
- Prototype Model

- C, C++
- VHDL, Verilog
- ARM
- FPGA
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

Test bench

Design
Enhanced Edge Detection
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 $\rightarrow$ pixel-stream processing $\rightarrow$ 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
  - 480p
  - 480pH
  - 720p
  - 768p
  - 1024p
  - 1080p
  - 1200p
  - 2K
  - 4K
  - 8K

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

**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
HDL Coder

Provides VHDL and Verilog code generation for MATLAB and Simulink

MATLAB Code

10X more concise

60% reduction in time to produce a working prototype

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

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
Distributed Pipelining
Speed Optimization

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

(Analog) Plant Model
Embedded Processor
Digital Controller
FPGA

Control
Measurement

100 KHz
(funcional rate)

100 MHz
(FPGA Clock rate)

Oversampling = 1000
Functional Delay

Latency Budget = 1000 cycles (Z^{-1000})

Clock-Rate Pipelining Exposes latency budget
Critical Path Estimation

```matlab
hdlsim_param(model, 'CriticalPathEstimation', {'on' | 'off'})
```

## Supported Targets

- Xilinx
  - Artix
  - Kintex
  - Virtex
  - Zynq
- Altera
  - Cyclone V
  - Stratix V
- Vivado
Meeting Resource Constraint

Area Optimization

Clk = 10MHz

MUX

DEMUX

SCHEDULING

Clk = 60MHz
Hardware Design Solution: Resource Sharing and Streaming
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](image)
Traceability Between Model and Code

```matlab
// <S11>/Desired_reg
always @(posedge clk or posedge reset)
    begin : Desired_reg_process
        if (reset)
            Desired_reg_out1 <= 0;
        else
            if (enb)
                Desired_reg_out1 <= Desired;
    end
```
Resource Utilization Estimation

Resource Utilization Report for sfir_fixed_demo

Summary

<table>
<thead>
<tr>
<th>Resource</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multipliers</td>
<td>4</td>
</tr>
<tr>
<td>Adders/Subtractors</td>
<td>7</td>
</tr>
<tr>
<td>Registers</td>
<td>8</td>
</tr>
<tr>
<td>RAMs</td>
<td></td>
</tr>
<tr>
<td>Multiplexers</td>
<td></td>
</tr>
</tbody>
</table>

Detailed Report

Report for Subsystem: symmetricfir

Multipliers (4)

[-] 1 × 16-bit Multiply : 4

- Product
- Product1
- Product2
- Product3

Adders/Subtractors (7)
Integrating Legacy HDL Code

HDL Supported Blocks

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
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
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
Backup
## CVST and VHT, complimentary products

*Product details*

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