Implementation of Vision algorithms on BRAVE FPGA's developed in MATLAB and VHDL environment

Mr. Klemen Bravhar (ESA)
Mr Stephan van Beek (MathWorks)

09/04/2018
Why Deploying Algorithms to FPGA/ASIC/SoC 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”
Modern Applications Often Require Custom Hardware

*Autonomous System Application Example*

1M+ pixels per frame

- **High-speed, well-defined**
  - Repetitive processing
  - Large amount of data

- **Few coords**

- **Complex, more flexible**
  - Small data calculations
  - Executive control

---

FPGA Hardware

Embedded Software
Typical Vision Workflow

**Image/Video Engineer**
- Concept Development
- Algorithm Development
- Micro-architecture Design
- Prototype
- Chip integration and Implementation

**Hardware Engineer**
- Frame-based
- Pixel-based

**Concept & Algorithm**
- Develop system-level algorithms
- Simulate, analyze, modify
- Partition hardware vs software implementation

**Algorithm to Micro-architecture**
- Convert to streaming algorithms
- Add hardware micro-architecture
- Convert data types to fixed-point

**Micro-architecture to implementation**
- HDL code creation
- Speed and area optimization
- Verification
- FPGA/ASIC/SoC implementation
Frame-Based vs Pixel-Streaming Algorithms

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

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

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

Algorithm (Golden Reference)

Algorithm w/ HW Implementation

Fixed-Point, Optimized Implementation

FPGA/ASIC Implementation

Simulink

HDL Coder

HDL-redy IP blocks

Verify

HDL Coder Prototype

HDL Coder Production

Fixed Point Designer

Core interface

VHDL / Verilog

Constraints
Introduction

• Objectives of project
• Description of deployed applications on FPGA (Green Screen and Edge Detection)
• Methodology of simulations and implementations
• Debugging and Verifying RTL designs on Xilinx and BRAVE FPGA with MATLAB/Simulink workflow and on conventional way
• BRAVE Implementation and lessons learned
Objectives

- Modeling, Simulating and Implementing FPGA design with help of high-level MATLAB/Simulink workflow
- Modeling, Simulating and Implementing FPGA design with conventional RTL HDL workflow
- Compare between MATLAB/Simulink and conventional RTL workflows (time for modulating, simulating, verifying/debugging || used resources)
- FPGA(RTL) design has to include following components: DSP, Memory, Stream of data, LUTs, etc.
- Synthesis, Map and Place&Route of the source codes on Xilinx Kintex-7
- Synthesis, Map and Place&Route of the source codes on BRAVE FPGA
Green Screen

**Conventional VHDL modeling**

```
entity green_screen is
  generic(
    g_RELAY : Integer := 7;
    g_DATA : Integer := 8);
  port(
    i_clk : in std_logic;
    i_reset : in std_logic;
    i_wBlank : in std_logic;
    i_bBlank : in std_logic;
    i_din : in std_logic_vector;
    i_pixel_hi : in std_logic_vector(g_DATA-1 downto 0);
  );
end green_screen;
```

**Input image:**
Green background

**Output image:**
Green background is replaced with white background

**MATLAB/Simulink modeling**

**Input image:**
Green background

**Output image:**
Green background is replaced with white background
Edge Detection

Conventional VHDL modeling

MATLAB/Simulink modeling

Input image

Output image: Detection of edges
Abstract model for testing functionality of FPGA models

Top level: Green Screen

HDMI in ➔ GREEN SCREEN ➔ HDMI out

MATLAB/Simulink module replaced with VHDL module and vice versa

Top level: Edge Detection

HDMI in ➔ HDMI to MATLAB ➔ EDGE DETECTION ➔ MATLAB to HDMI ➔ HDMI out

Gauss filter ➔ Sobel operator ➔ Threshold

Sobel operator

Threshold
Simulation process

Hand-written DUT

MATLAB/Simulink generated DUT

DUT ➔ Design under test
Validating of MATLAB/Simulink model with help of HDL Verifier on Xilinx Kintex 7
Validation on Breadboard Xilinx Kintex 7
Validation of designs on BRAVE FPGA
# BRAVE used resources

## Green Screen

<table>
<thead>
<tr>
<th></th>
<th>4-LUT</th>
<th>DFF</th>
<th>XLUT</th>
<th>4-bits carry</th>
<th>Register file block</th>
<th>Cross domain clock</th>
<th>Clock switch</th>
<th>Digital signal processor</th>
<th>Memory block</th>
<th>WFG</th>
<th>PLL</th>
<th>Clock Frequency</th>
<th>TDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand-written RTL</td>
<td>536/32256 (2%)</td>
<td>446/32256 (2%)</td>
<td>2/2016 (1%)</td>
<td>13/2016 (1%)</td>
<td>0/168 (0%)</td>
<td>0/168 (0%)</td>
<td>0/336 (0%)</td>
<td>2/112 (2%)</td>
<td>0/56 (0%)</td>
<td>0/32 (0%)</td>
<td>0/4 (0%)</td>
<td>114.929MHz</td>
<td>32 hours</td>
</tr>
<tr>
<td>MATLAB/Simulink generated RTL</td>
<td>557/32256 (2%)</td>
<td>451/32256 (2%)</td>
<td>3/2016 (1%)</td>
<td>15/2016 (1%)</td>
<td>0/168 (0%)</td>
<td>0/168 (0%)</td>
<td>0/336 (0%)</td>
<td>2/112 (2%)</td>
<td>0/56 (0%)</td>
<td>0/32 (0%)</td>
<td>0/4 (0%)</td>
<td>116.729MHz</td>
<td>24 hours</td>
</tr>
</tbody>
</table>

## Edge Detection

<table>
<thead>
<tr>
<th></th>
<th>4-LUT</th>
<th>DFF</th>
<th>XLUT</th>
<th>4-bits carry</th>
<th>Register file block</th>
<th>Cross domain clock</th>
<th>Clock switch</th>
<th>Digital signal processor</th>
<th>Memory block</th>
<th>WFG</th>
<th>PLL</th>
<th>Clock Frequency</th>
<th>TDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand-written RTL</td>
<td>2871/32256 (9%)</td>
<td>2120/32256 (7%)</td>
<td>13/2016 (1%)</td>
<td>176/2016 (9%)</td>
<td>0/168 (0%)</td>
<td>0/168 (0%)</td>
<td>0/336 (0%)</td>
<td>27/112 (25%)</td>
<td>4/56 (8%)</td>
<td>1/32 (4%)</td>
<td>1/4 (25%)</td>
<td>112.322MHz</td>
<td>108 hours</td>
</tr>
<tr>
<td>MATLAB/Simulink generated RTL</td>
<td>3137/32256 (10%)</td>
<td>2047/32256 (7%)</td>
<td>32/2016 (2%)</td>
<td>177/2016 (9%)</td>
<td>0/168 (0%)</td>
<td>0/168 (0%)</td>
<td>0/336 (0%)</td>
<td>2/112 (2%)</td>
<td>7/56 (13%)</td>
<td>1/32 (4%)</td>
<td>1/4 (25%)</td>
<td>66.234MHz</td>
<td>24 hours</td>
</tr>
</tbody>
</table>

**TDI ➔ Time for Development and Implementation, Verification on Kintex7**

---

*ESA UNCLASSIFIED - For Official Use*  
Mr. Klemen Bravhar (ESA) Mr Stephan van Beek (MathWorks) | 09/04/2018 | Slide 16
Lesson learned

- MATLAB/Simulink accelerates development of a model and functional tests of a model
- HDL Coder helps user to generate VHDL code from developed model in Simulink
- HDL verifier:
  - Cosimulation (Simulink and ModelSim) ➞ user can simulate Simulink model directly in ModelSim environment
  - FPGA-In-The-Loop ➞ Debugging and verifying of RTL design directly on FPGA
- Decreases time for prototyping, verifying and deploying RTL design on FPGA
BRAVE FPGA Hardware Test Environment

Video Source -> HDMI In Card -> FPGA -> HDMI Out Card -> Display
DEMO: Software Defined Radio on FPGA/SoC

• Verify using FPGA-in-the-Loop techniques on Microsemi FPGA boards
Questions?