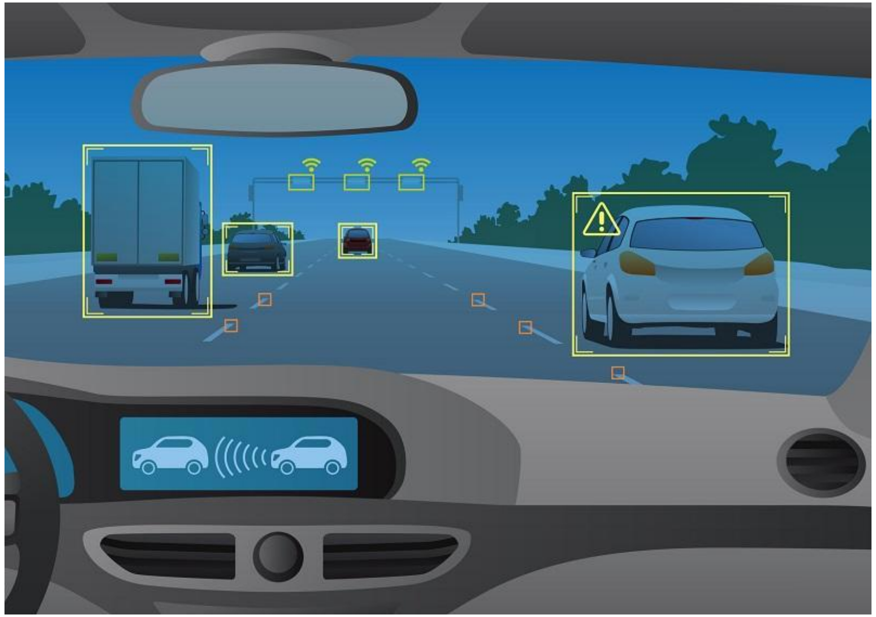
Implementation of Vision algorithms on BRAVE FPGA's developed in MATLAB and VHDL environment

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

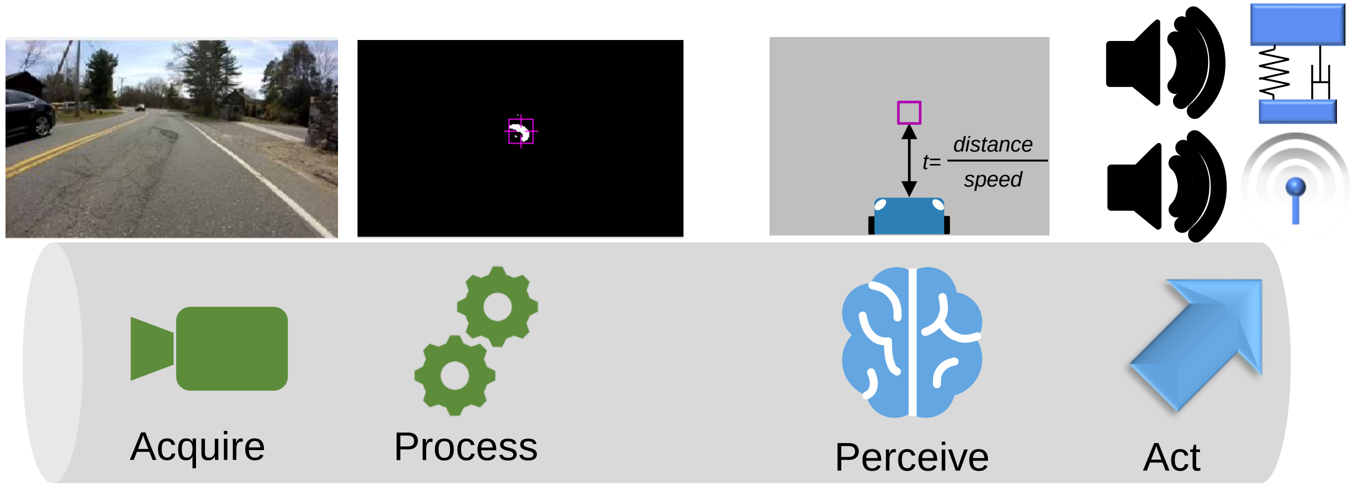
"Be able to stop the robot with millimeter accuracy in less than 0.5 seconds without causing damage to the robot"

"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

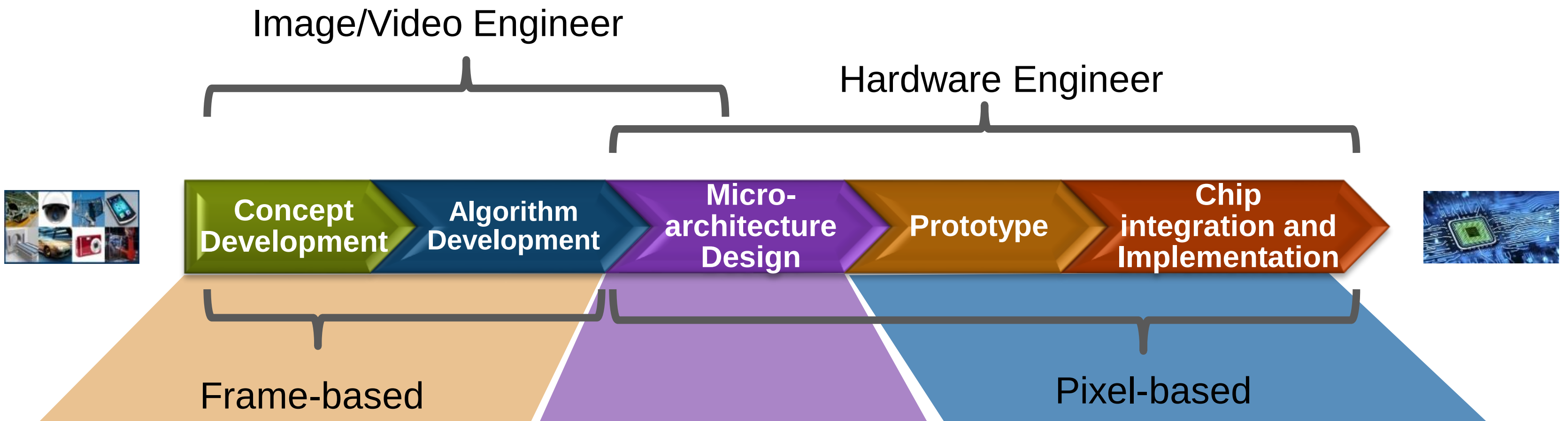
FPGA Hardware

Few coords

Complex, more flexible
- Small data calculations
- Executive control

Embedded Software

Typical Vision Workflow



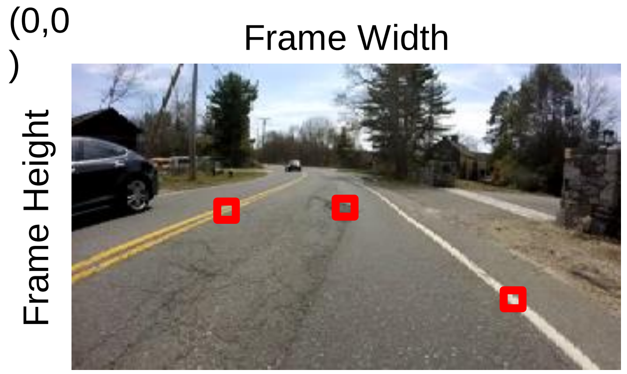
Concept & Algorithm
- Develop system-level algorithms
- Simulate, analyze, modify
- Partition hardware vs software implementation

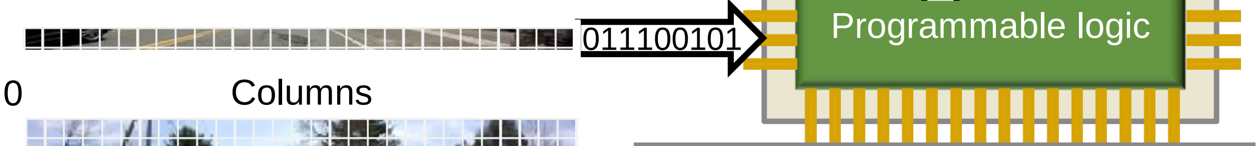
Algorithm to Micro-architecture
- Convert to streaming algorithms
- Add hardware micro-architecture
- Convert data types to fixed-point

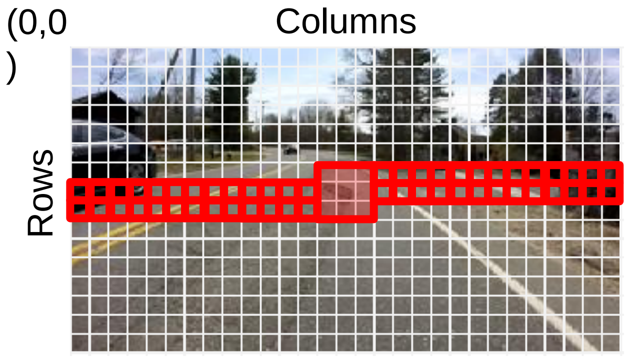
Micro-architecture to implementation
- HDL code creation
- Speed and area optimization
- Verification
- FPGA/ASIC/SoC implementation

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

Algorithm (Golden Reference)

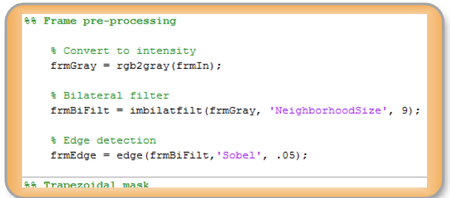
Algorithm w/ HW Implementation

Fixed-Point, Optimized Implementation

FPGA/ASIC Implementation

MATLAB

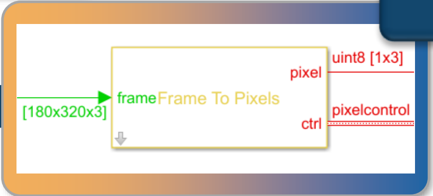
Simulink

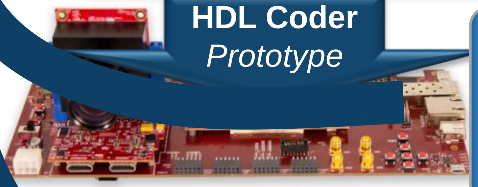


```
%% Frame pre-processing

    % Convert to intensity
    frmGray = rgb2gray(frmIn);

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

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

%% Trapezoidal mask
```

Verify

HDL-ready IP blocks

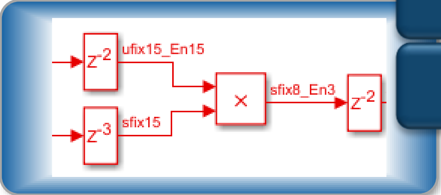




HDL Coder
Prototype

Fixed Point Designer

HDL Coder



HDL Coder
Production

Core interface

VHDL / Verilog

Constraints

Introduction

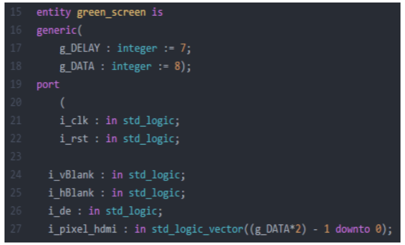
- Objectives of project
- Description of deployed applications on FPGA (Green Screen and Edge Detection)
- Methodology of simulations and implementations
- Debugging and Verifying RTL designs on Xilinx and BRAVE FPGA with MATLAB/Simulink workflow and on conventional way
- BRAVE Implementation and lessons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

Objectives

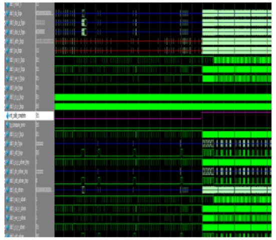
- Modeling, Simulating and Implementing FPGA design with help of high-level MATLAB/Simulink workflow
- Modeling, Simulating and Implementing FPGA design with conventional RTL HDL workflow
- Compare between MATLAB/Simulink and conventional RTL workflows (time for modulating, simulating, verifying/debugging || 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

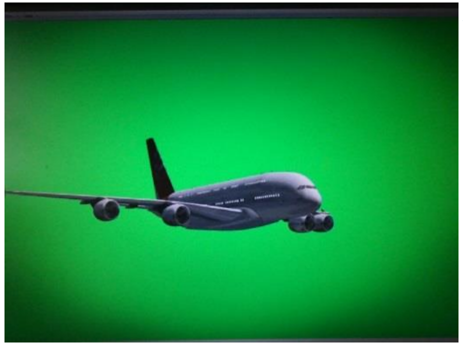
Green Screen

Conventional VHDL modeling

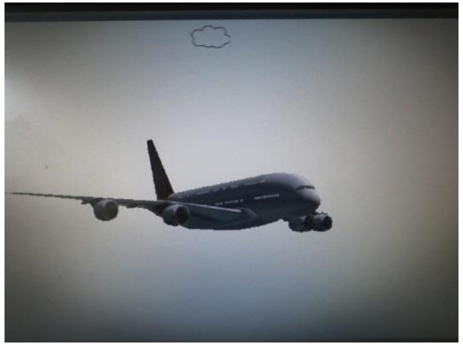


```
entity green_screen is
generic(
    g_DELAY : integer := 7;
    g_DATA : integer := 8);
port
    (
    i_clk : in std_logic;
    i_rst : in std_logic;

  i_vBlank : in std_logic;
  i_hBlank : in std_logic;
  i_de : in std_logic;
  i_pixel_hdmi : in std_logic_vector((g_DATA*2) - 1 downto 0);
```

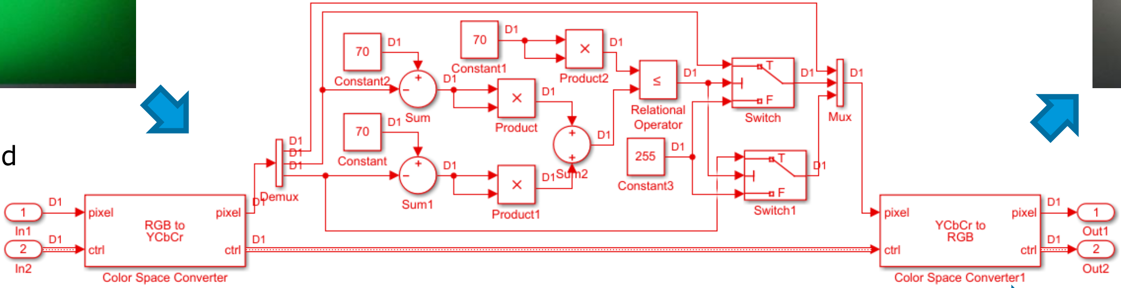




Input image:
Green background



Output image:
Green background is replaced with white background

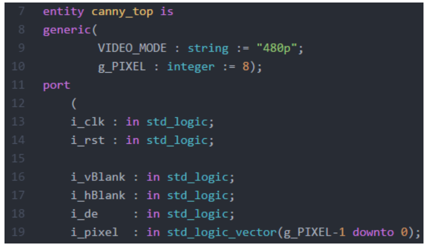


MATLAB/Simulink modeling

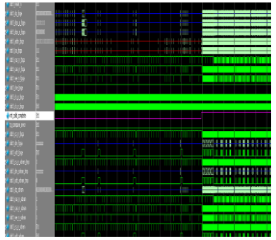
Edge Detection

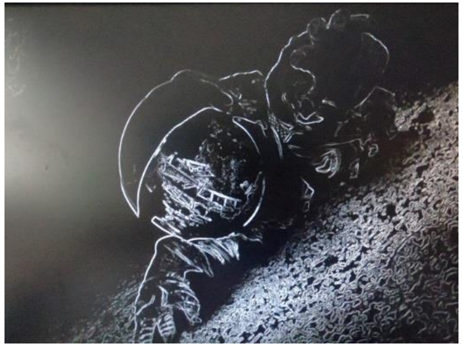
Conventional VHDL modeling

Input image

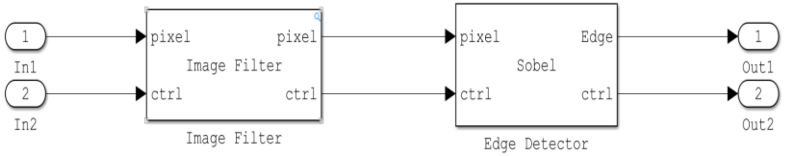


```
entity canny_top is
generic(
        VIDEO_MODE : string := "480p";
        g_PIXEL : integer := 8);
port
    (
    i_clk : in std_logic;
    i_rst : in std_logic;

    i_vBlank : in std_logic;
    i_hBlank : in std_logic;
    i_de     : in std_logic;
    i_pixel  : in std_logic_vector(g_PIXEL-1 downto 0);
```



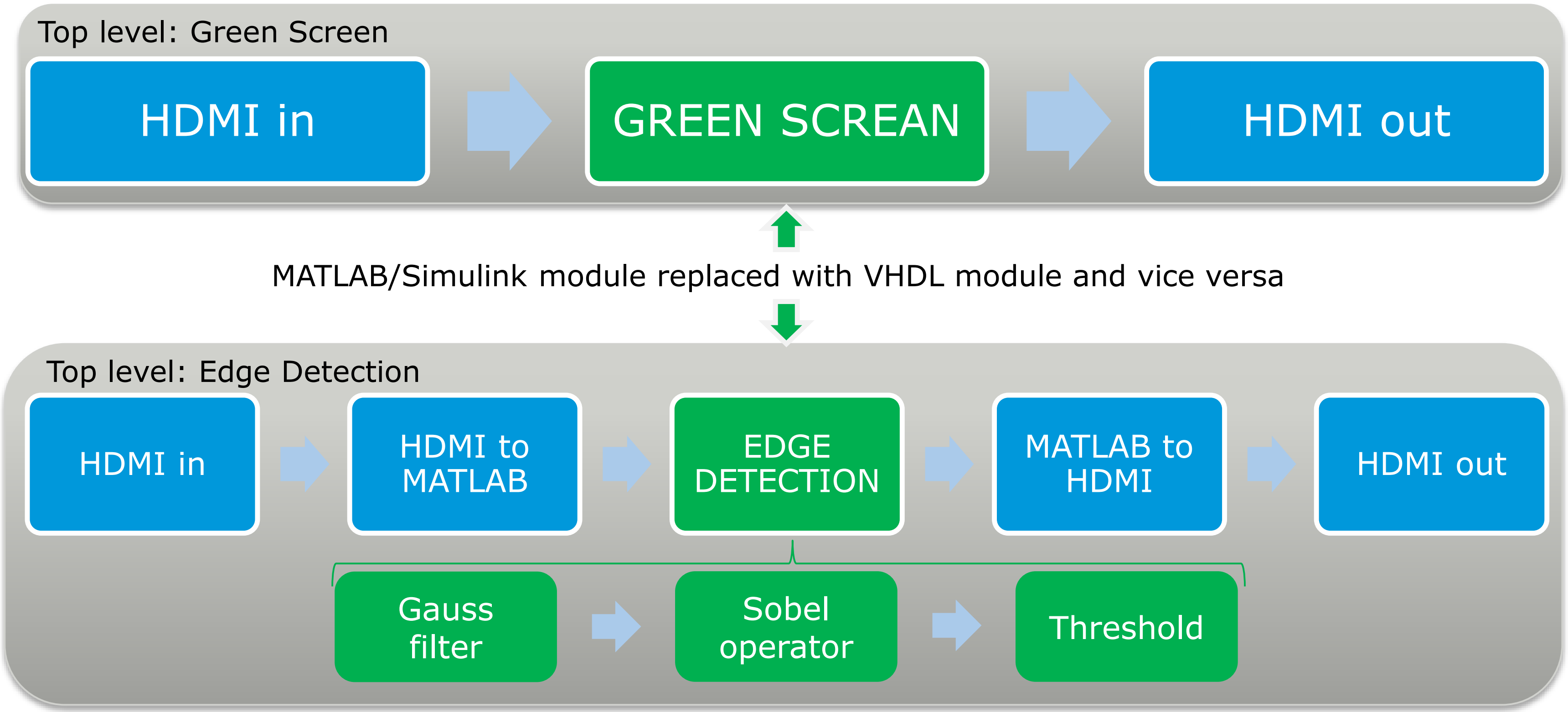


Output image: Detection of edges

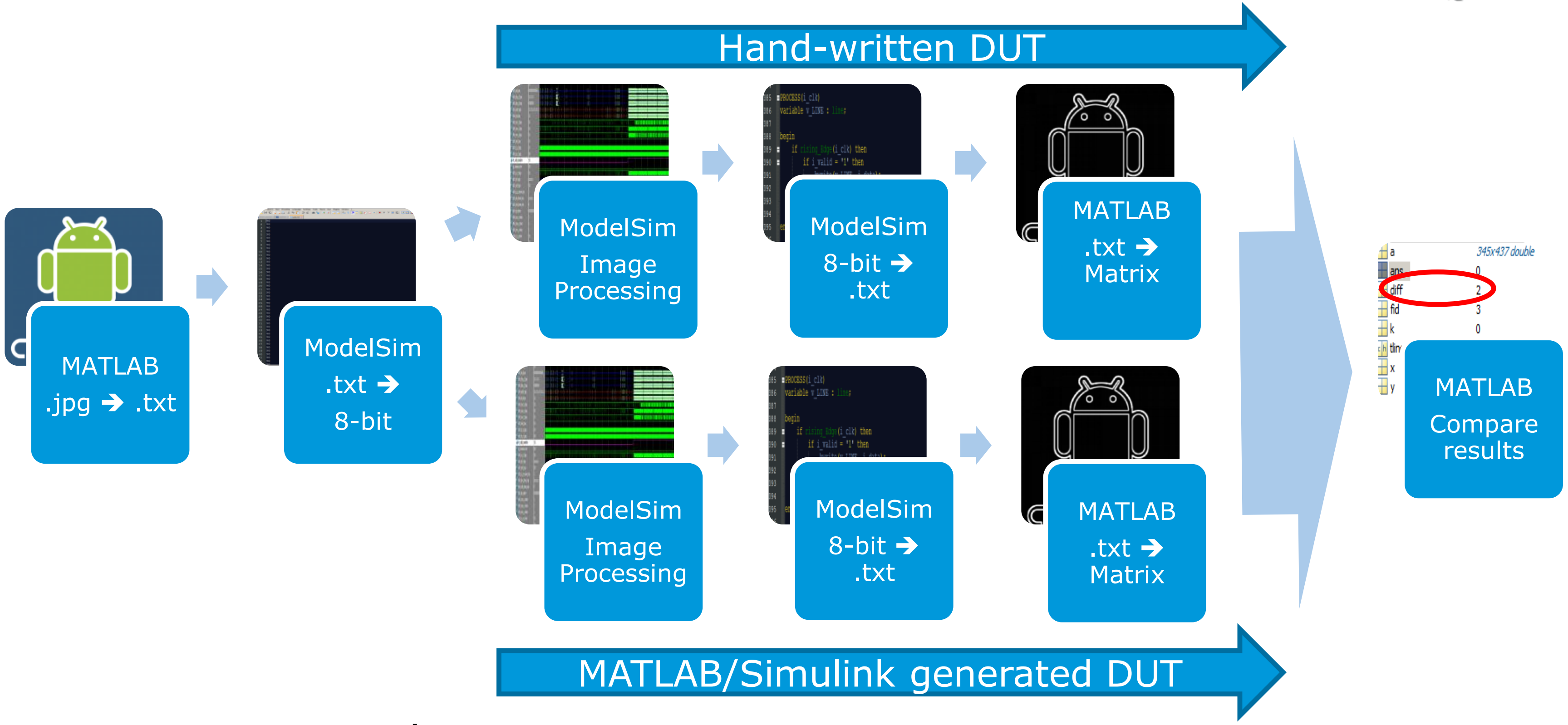


MATLAB/Simulink modeling

Abstract model for testing functionality of FPGA models



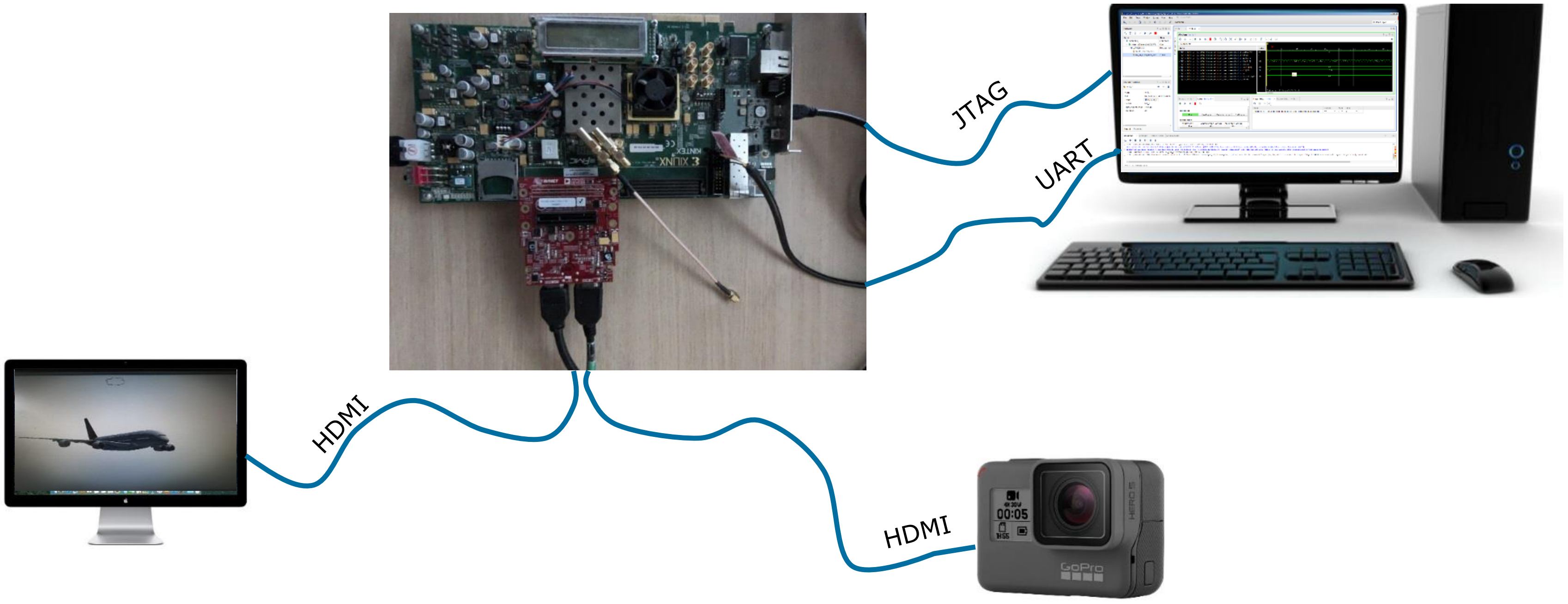
Simulation process



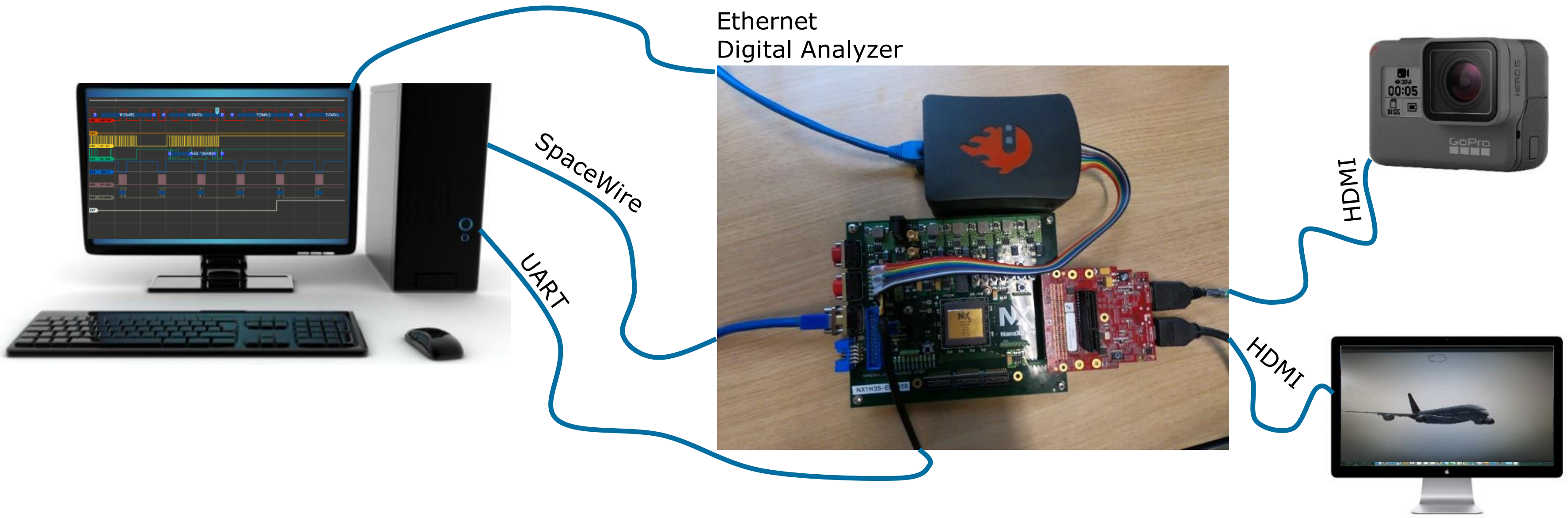
DUT ➔ Design under test

Validating of MATLAB/Simulink model with help of HDL Verifier on Xilinx Kintex 7

Validation on Breadbord Xilinx Kintex 7



Validation of designs on BRAVE FPGA



BRAVE used resources

Green Screen

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

Edge Detection

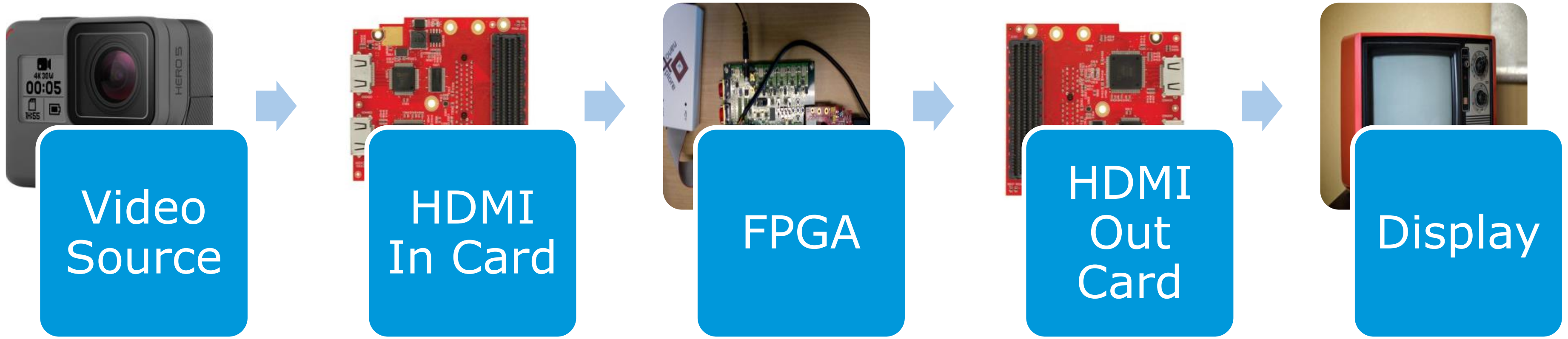
	4-LUT	DFF	XLUT	4-bits carry	Register file block	Cross domain clock	Clock switch	Digital signal processor	Memory block	WFG	PLL	Clock Frequency	TDI
Hand-written RTL	2871/32256 (9%)	2120/32256 (7%)	13/2016 (1%)	176/2016 (9%)	0/168 (0%)	0/168 (0%)	0/336 (0%)	27/112 (25%)	4/56 (8%)	1/32 (4%)	1/4 (25%)	112.322MHz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

TDI ➔ Time for Development and Implementation, Verification on Kintex7

Lesson learned

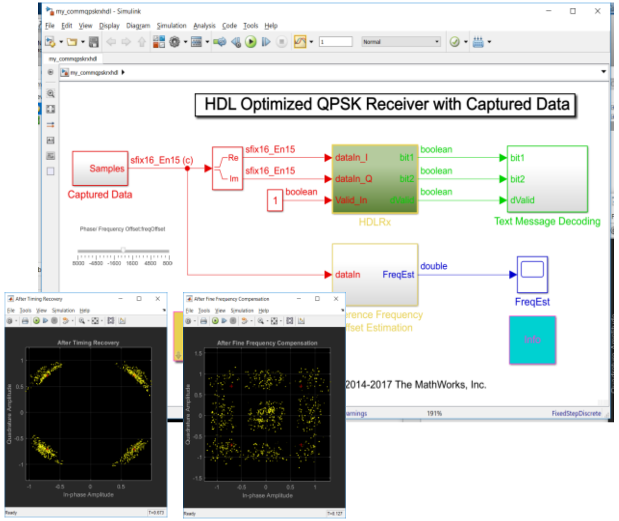
- MATLAB/Simulink accelerates development of a model and functional tests of a model
- HDL Coder helps user to generate VHDL code from developed model in Simulink
- HDL verifier:
  - Cosimulation (Simulink and ModelSim) ➔ user can simulate Simulink model directly in ModelSim environment
  - FPGA-In-The-Loop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

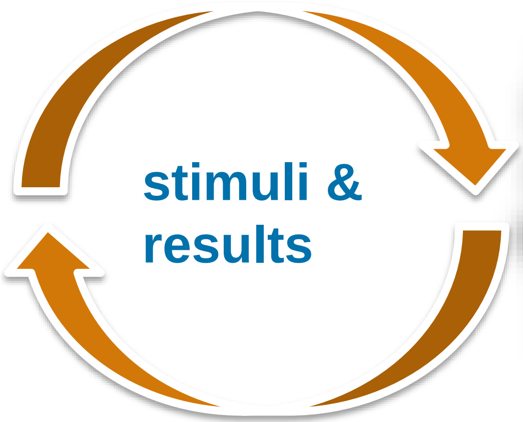
BRAVE FPGA Hardware Test Environment

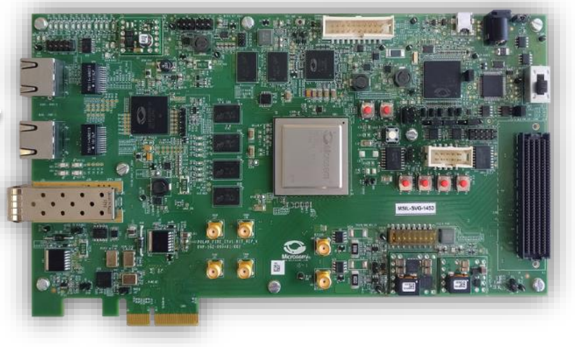


DEMO: Software Defined Radio on FPGA/SoC

- Verify using FPGA-in-the-Loop techniques on Microsemi FPGA boards







Questions?