# Design Experiments for the FPGA Vision Remote Lab

Prof. Dr. Marco Winzker, Bonn-Rhein-Sieg University, 2018

**Competencies and Learning Objectives**

With the lecture you can experience product development from algorithm to circuit implementation. Investigate different tasks in this process with design experiments.

Information about the experiments is also available as a video lecture:
https://youtu.be/KMoczJ_p7Gc

**Content of these Slides**

- Overview
- Access to the lecture
- Ideas for experiments

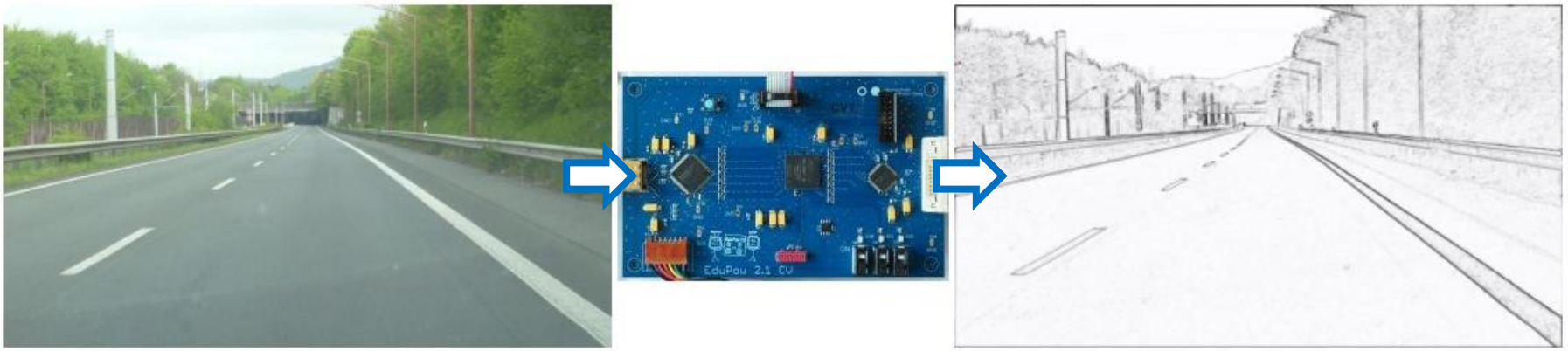
Feel free to use this material for your needs

Version: 23. August 2018

# Overview

The lecture covers the design of a lane detection algorithm for cars and its implementation on an FPGA.

**input image** **FPGA board** **output image**



**Choose your learning goals**

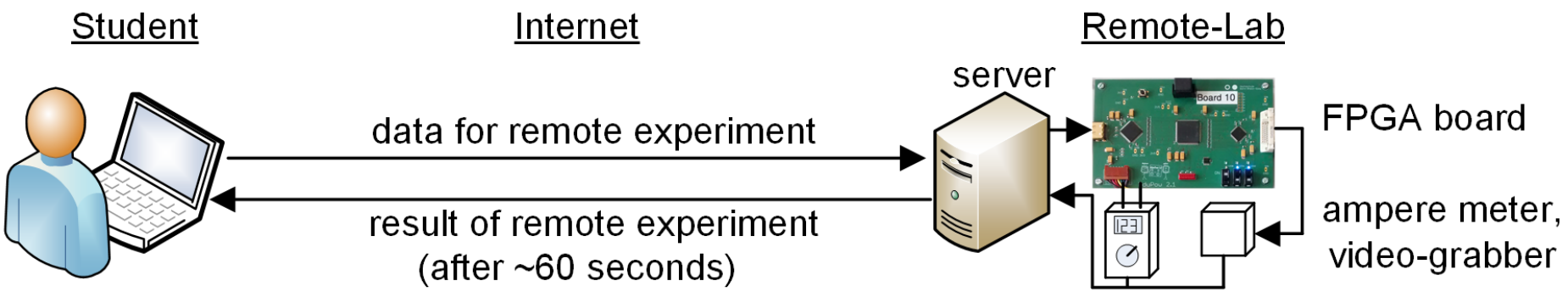
- Signal processing: Understand and optimize the image processing algorithm
- Digital design: Modify the FPGA design and reduce FPGA resources and power consumption
- Microelectronics: Compare power consumption for different CMOS technologies

# Access to the Lecture

All material can be accessed from the project page
http://www.h-brs.de/fpga-vision-lab

The lecture provides:

- Seven lecture videos
- Access to remote-lab with FPGA
  - Altera/Intel Cyclone V
  - Altera/Intel Cyclone IV for comparison of CMOS technologies
- Source code: C and VHDL



# Lecture Videos

**Introduction** (2:24 minutes)

scope and overview of the lectures

**Technical Background**

**Lane Detection** (9:58 minutes)

algorithm and verification
implementation as C program

**Low-Power Design** (15:48 minutes)

CMOS power consumption
digital design for low-power

**Digital Design**

**Circuit Design** (14:18 minutes)

architecture of lane detector
fixed-point implementation

**VHDL Simulation** (10:13 minutes)

verification concept
design flow for simulation

**Experiment on Remote Lab**

**FPGA Remote Lab** (8:44 minutes)

FPGA design flow
access to remote lab

**Next Steps** (2:39 minutes)

suggestions for further experiments

- Subtitles in English, Arabic, Spanish
- Select which videos are of interest for you

# Set-Up of Design-Flow

**Learning goal:**

- Understand the design-flow of FPGA design and remote-lab usage
- This experiment should be done as a preparation for all other experiments

**Level of difficulty:** Easy

**Experiment:**

a) Install the Intel/Altera Quartus design software and device files for Cyclone V on your computer

b) Download the source files for FPGA design and perform FPGA synthesis

c) Upload the bitfile (filetype: sof) to the remote-lab and perform the experiment

The remote-lab should detect the edges of the input image as shown in the video lectures.

# Optimize Position of RGB-to-Y Conversion (1/2)

**Learning goal:**

- Working with the VHDL files
- Understanding the structure of the circuit design

**Level of difficulty:** Moderate

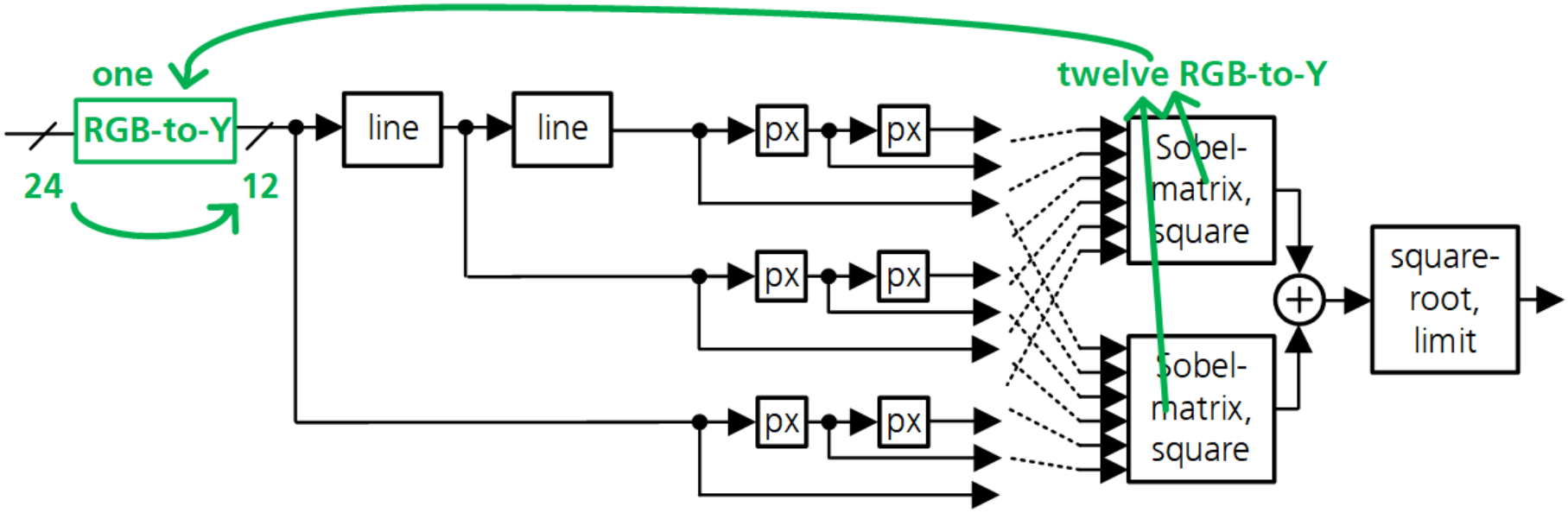
**Experiment:**

The RGB (red, green, blue) input pixel need 24 bit and are stored in two line memories for vertical edge detection. For processing they are converted to luminance (Y) with 12 bit resolution.

It is better to first convert RGB to Y and only store 12 bit to save memory. Also this reduces resources for processing, as only one conversion is required and not several instances.

# Optimize Position of RGB-to-Y Conversion (2/2)

a) Change the VHDL code to perform RGB-to-Y before the line memories
b) Simulate the new VHDL code to verify the changes
   - The output of the modified code should be identical to the original design
c) Perform FPGA synthesis and compare resource usage with original design
d) Upload the bitfile to the remote-lab, check the functionality and compare power dissipation with original design



# Reduce Word Width of Luminance Values (1/3)

**Learning goal:**

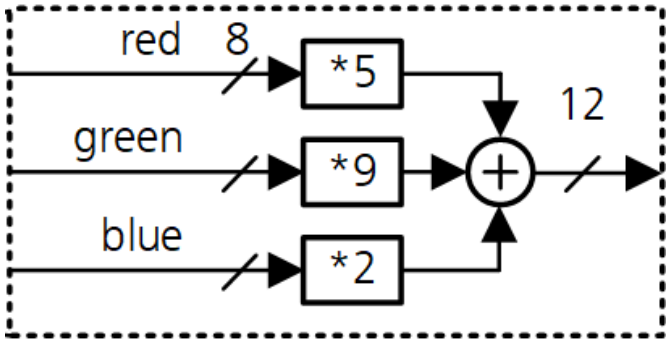
- Optimization of the signal processing algorithm
- Design verification with C-program and simulation

**Level of difficulty:** Advanced

**Experiment:**

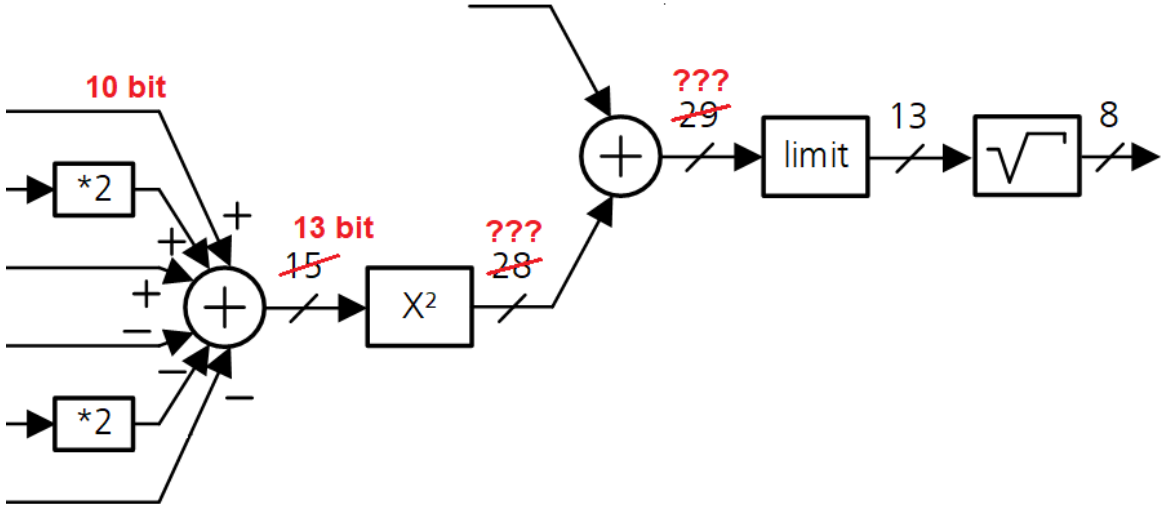
An input pixel has three components (red, green, blue), each with 8 bit. They are converted to 12 bit luminance values. Reduce the word width of luminance values to save resources.

Reducing word width can be done by discarding the lower bits of the 12 bit luminance value after adding red, green and blues components



# Reduce Word Width of Luminance Values (2/3)

Please note that reducing the word width of luminance values also reduces the word width of the consecutive processing steps (see diagram for an example).




a) Reduce the word width of luminance in the C code to 10 bit, 8 bit or other word width and check if the output image still has acceptable quality

b) Change the VHDL code accordingly and verify the changes by simulation
   - NOTE: The output of the design is not identical to the original design. Use the modified C code to generate new expected values for the testbench.

c) Perform FPGA synthesis and compare resource usage to original design

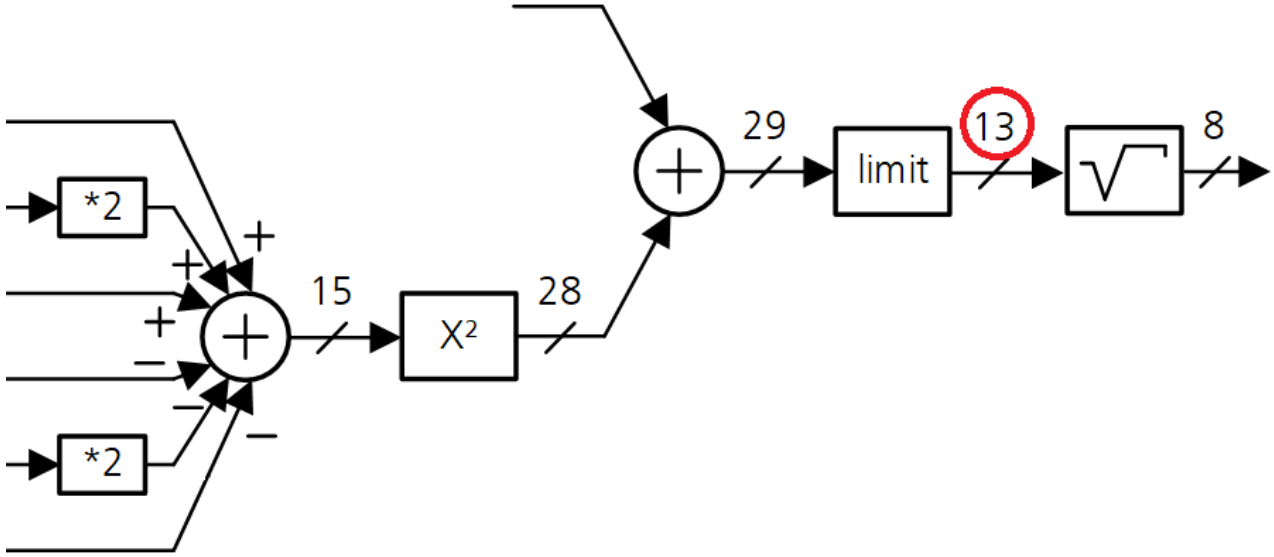
d) Use the remote-lab to check functionality and compare power dissipation

# Reduce Word Width of Luminance Values (3/3)

Also the input of the square-root function can be changed from 13 to 10 bit or 8 bit.

e) Change C code and VHDL code

f) Verify the design and compare it with other versions of the design




These optimization steps can be done on their own or combined with each other.

They can also be combined with the experiment “interchanging the order of RGB-to-Y conversion and line memories”.

# Power Consumption of different FPGAs

**Learning goal:**

- Understand the influence of CMOS technology for power consumption

**Level of difficulty:** Moderate

**Experiment:**

Implement a design on two different FPGAs and compare resource usage and power consumption.

a) Check the data sheet of Altera/Intel Cyclone IV and Cyclone V FPGAs. Which CMOS technology do they use?

b) Implement a design for both FPGAs. Different constraints files for pin locations are provided.

c) Check resource usage and compare the power consumption

d) Compare other circuit designs, e.g. a filter or shift register (see next section)

See video lecture “Comparing CMOS Technologies with FPGA Experiments”

https://youtu.be/7hSjqMc742A

# Reduce Power Consumption by “Sleep Mode” (1/2)

**Learning goal:**

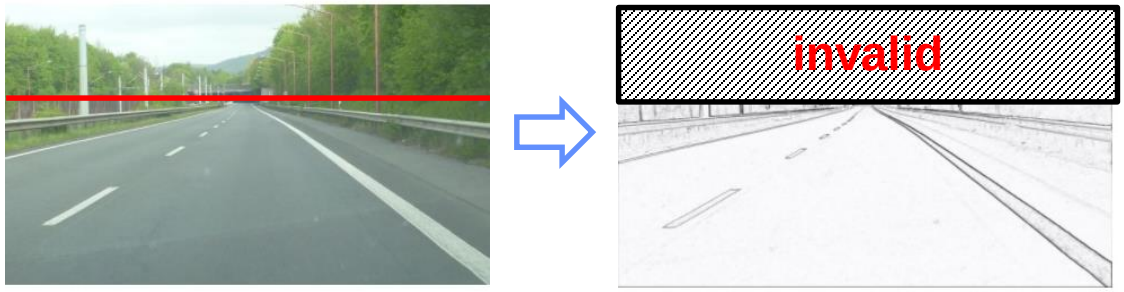
- Optimization of power consumption
- Understand sleep modes which disabling a circuit for certain times

**Level of difficulty:** Advanced

**Experiment:**

Lane detection is active for the complete image. However, lanes are not present in the top region of the image (see diagram).

Therefore processing can be switched off for the top of the image. Consequently, there is no valid output for this region.



# Reduce Power Consumption by “Sleep Mode” (2/2)

Look at test images to determine, how many lines of the image you want to skip. The image has 720 lines and, as an example, processing can be disabled for one third of the image, which is 240 lines.

Different techniques for such a “sleep mode” are possible:

- Set input pixel to a fixed value. This will reduce switching activity.
- Disable all flip-flops in the signal processing by a “sleep-signal”.
- Switch of the clock for the signal processing circuit. Please note:
  - You need two clock signals: One for generating the sync signals and counting lines. The other is for signal processing and can be disabled.
  - The FPGA has predefined routing for clock signal, so you will drive two clock networks. Find out if you save power or if the overhead increases power consumption.

a) Modify the design and simulate it. Modify the testbench to ignore undefined output lines.
b) Perform FPGA synthesis and compare resource usage to original design
c) Use the remote-lab to check functionality and compare power dissipation

# General Image Processing Experiments

**You can also use the remote lab for general experiment about image processing.**

**The following experiments are not specific for lane detection.**

# Inverting the Input Image

**Learning goal:**

- Understand input and output signals of the VHDL code
- Understand design-flow of FPGA design and remote-lab usage

**Level of difficulty:** Easy

**Experiment:**

Invert the color of every pixel.

a) Check the existing VHDL code to understand the input and output signals

b) Invert the values of RGB (red, green, blue) for all pixel

Optional:

c) Use the input switches to select which color component is inverted
   - The switches connect to the input port `enable_in(2 downto 0)`

d) Convert the image to black and white by calculating luminance and output this value on the red, green and blue channel

# Image Enhancement with Sharpening Filter

**Learning goal:**

- Signal processing with an FPGA

**Level of difficulty:** Advanced

**Experiment:**

Improve the perceived image quality by sharpening the image with an FIR filter. Research literature about sharpening filter.

a) Develop an algorithm using C code

b) Implement the algorithm with VHDL, simulate it and check functionality on the remote-lab
   - To see the output image in full resolution click on the image

Different filter functions are possible. For example you can implement the filter function h = [1;0;-9;48;-9;0;1]·1/32 in vertical and horizontal direction.

Develop a filter that has good quality with low effort.

Make sure the output values stay in the range of 0 to 255. Don’t forget limiting.

# Power Consumption of Shift Register (1/2)

**Learning goal:**

- Understand power consumption in digital circuits
- Compare static and dynamic power consumption

**Level of difficulty:** Moderate

**Experiment:**

Implement a shift register with different numbers of flip-flops. Delay red, green, blue pixel of the input image but do not delay sync signals.

The image content is shifted by the number of delay stages.

a) Implement a shift register for the image signal. You need to delay 24 bit for RGB data. Check resource usage.

b) Compare the power consumption for 100, 200 and 300 clock cycles delay. How much delay can you implement?

c) To measure static power consumption, disable the clock signal with a switch.
   - The switches connect to the input port `enable_in(2 downto 0)`

# Power Consumption of Shift Register (2/2)

The FPGA synthesis will try to implement a shift register with block-RAMs. To observe power consumption of flip-flops, specify an array with the code in the box.

```
[...]
type delay_array is array (0 to 100)
                 of std_logic_vector(23 downto 0);
signal delay  : delay_array;
attribute ramstyle : string;
attribute ramstyle of delay : signal is "logic";
[...]
```

This experiment is also explained in the video lecture “Comparing CMOS Technologies with FPGA Experiments”

https://youtu.be/7hSjqMc742A

Further option for this experiment:

d) Remove the “attribute ramstyle” to implement the circuit with block-RAMs. Compare resource usage and power consumption.

# Brightness Adjustment

**Learning goal:**

- Signal processing with an FPGA

**Level of difficulty:** Complex

**Experiment:**

When you have images that are dark or bright, it can be useful to adjust the brightness. Also the contrast can be enhanced. Research literature on this topic.

The FPGA board does not provide a frame memory, so you can analyze one image, for example with a histogram of signal values. The result can then be used to optimize the next image.

a) Develop an algorithm using C code

b) Implement the algorithm with VHDL, simulate it and check functionality on the remote-lab

# Complex Image Processing Experiment

**And you can use the remote lab for a very complex image processing experiment.**

**This experiment can take several weeks or a complete semester.**

# Further Steps in Lane Detection (1/2)

**Learning goal:**

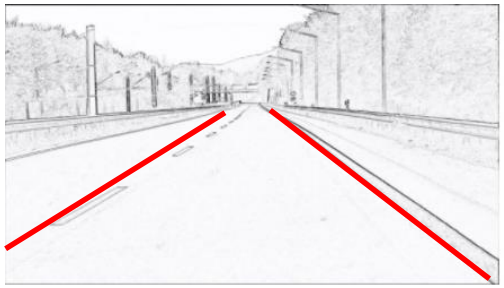
- Development of image processing algorithms

**Level of difficulty:** Very complex

**Experiment:**

The current design detects edges in the input image. Next step in a lane detection system are identification of lane boundaries from the edges, tracking of lanes and warning when a vehicle crosses lanes.

In the output image you can superimpose the lane boundaries on the original image or the edge detection image.



# Further Steps in Lane Detection (2/2)

a) Research literature about lane detection and lane departure warning
   - Good starting points are http://scholar.google.de/ and http://ieeexplore.ieee.org/
   - Often a Hough transform is proposed
b) Develop an algorithm using the C code
c) Implement the algorithm with VHDL, simulate it and check functionality on the remote-lab

The FPGA board does not provide a frame memory, so you can calculate lanes for one image and superimpose the result on the next image.

You can also generate a warning for leaving a lane and indicate it by a red box in the image.