# EEL 4930 - System-on-Chip Design

# Spring 2023

## Homework 1

Submitted By

Mohammad Bin Monjil (50782093)

Shuvagata Saha (58898380)

Hasan Al Shaikh (34715621)

Date: April 16, 2023

## Task 1:

We used the Pentalinux tool by Xilinx to build an OS image which can be run on the Zedboard SoC. We boot the built image on the QEMU emulator. We run the "Hello World" application which was named myapp. We modified the image to load this application and then we boot the image on SoC to run the application.

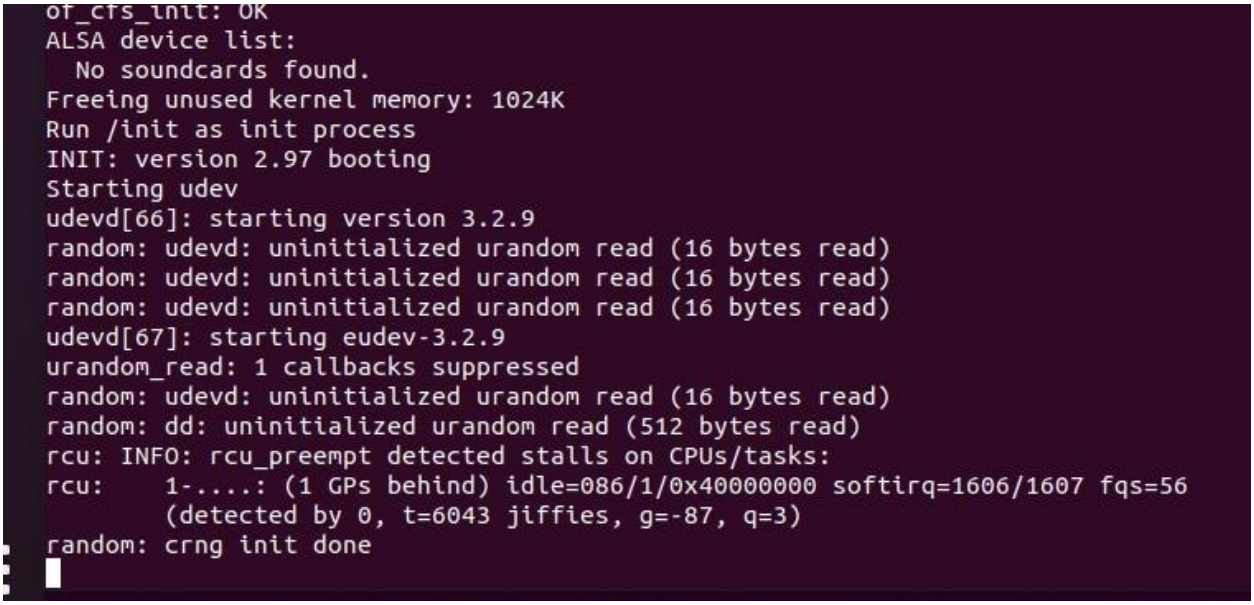


Figure 1: Challenge in booting the linux system in Qemu.

We faced a challenge while booting the image. The synchronization mechanism rcu detected stalls on CPU when the VM was using more than 1 processor. We solved this problem by assigning only a single processor to the VM.

## Task 2:

We completed Task 2 following the provided high-level block diagram with a input module to read image and send to processing module through a 24 bit int channel, processing module to perform median filtering/edge detection and send the processed image through another 24bit int channel to the output module. The output module is used to retrieve the image.

In our implementation, the processing module first collects all the pixels from the channel, converts the pixel values into a cv:: Mat object, performs median/sobel on the Mat object using opencv functions. The processed image is then converted back into integer values and sent through the channel to output module. The output module then converts the ineteger values back into the Mat array object. There are several signals connected between the module (other than the channels) to synchronize the data transfer between them. Figure 1 and 2 shows the resulting processed image on a sample image.

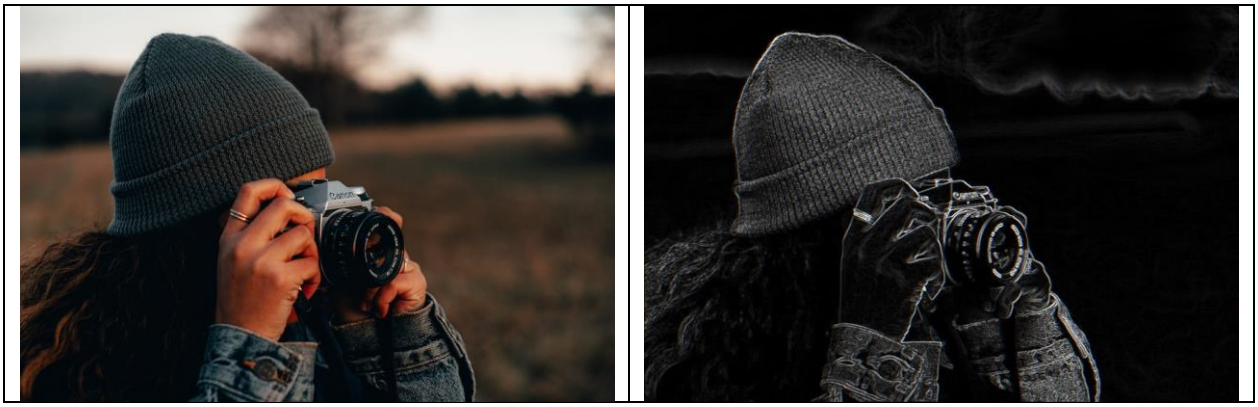

Figure 2: Sobel edge detection

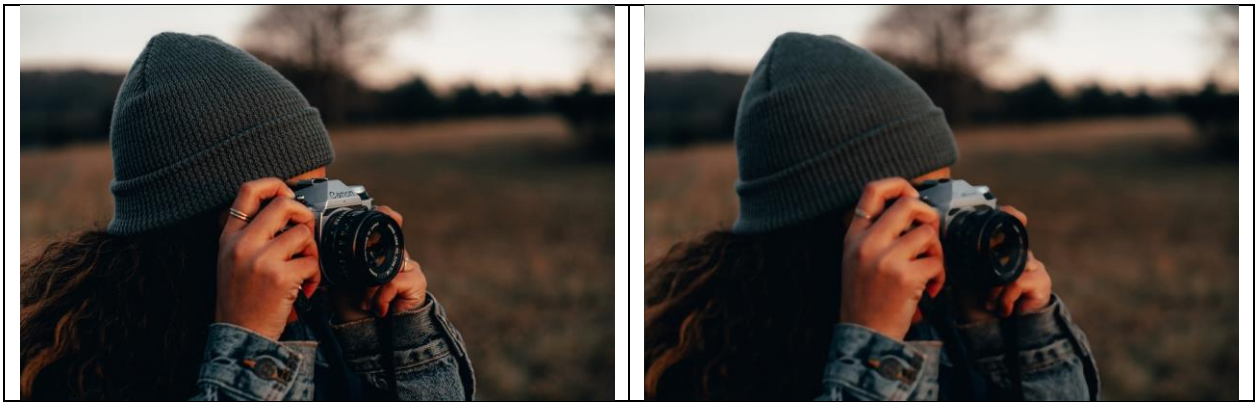

Figure 3: Median Filtering using a kernel size of 7

## Challenges:

For the sake of simplicity we initially implemented the above version. The execution time of this implementation is on the higher side. Later, we attempted to implement the project using FIFO concepts to reduce the runtime. The main change was in the processing module. Our pseudocode for the thread of the processing module is given in appendix. Unfortunately, we could not finish debugging the code.

## Submissions:

Task 1: Bitstream and .xsa file, linux images and Petalinux project files, video.
Task 2: C++ project files, binaries, readme.txt

## Contribution of members:

Mohammad Bin Monjil – Processing Module, Top-level
Shuvagata Saha – Task 1
Hasan Al Shaikh – Input & output module, report

# Appendix

Pseudocode of the thread of processing module with FIFO :