ECE 337

ASIC IMAGE PROCESSOR

"Preliminary Proposal Draft"

Ryan Beasley, Sidharth Mudgal, Suppatach Sabpisal,

Hao Xiong, Akanksha Sharma

Wednesday 11:30-2:20

T.A. Utpal Mahanta

1 Executive Summary

We are designing a low energy application-specific integrated circuit that will operates as an auxiliary unit as part of a system on chip inside any camera - including security camera, or digital camera. The goal of this design is to give the ability of real-time still image manipulation to optical devices (computer monitors, surveillance camera, digital cameras, computer vision applications, etc.).

The **ASIC Image Processor** has wide range of possible commercial and industrial applications. It can be used to interface over UART with any digital devices that has imaging functionality. It can also be used as part of a system-on-chip, to provide additional processing power. It can also be used to operate an image processing operations that requires processing of countless images repeatedly over time – *for instance, if you want to process 2 images taken every 5 seconds for the next 5 years and feed that data into a hard drive (may require additional hardware*. This is a critical point because if only one functionality is needed to be performed, ASIC architecture can provide order-of-magnitude reduction in power consumption compared to traditional methods. In addition, hardware implementation of image processing is done in more “real-time” fashion that it’s (generally) slower software counterpart. In industrial environment, it can be used as part of a system that inspect manufactured parts in the manufacturing line, in real-time.

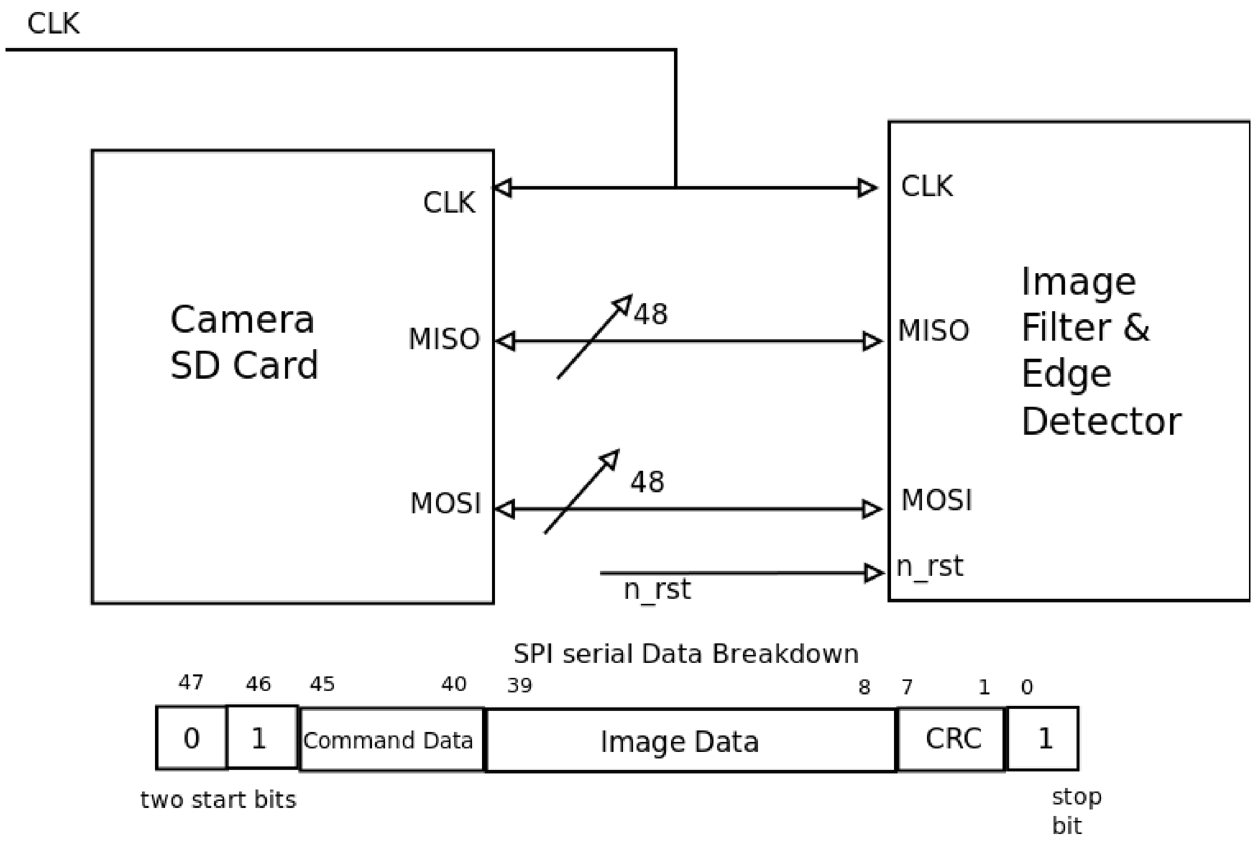
Our design of the ASIC Image Processor will be optimized for speed, because size doesn’t matter to us. It is trying to provide a “real-time” processing capability and supports wide range of image processing instruction sets and can be interfaced with easily over UART and can be applied to any system that support mentioned protocol.

This design is appropriate for ASIC because a microcontroller is too expensive for industrial applications, and consume too much power for commercial applications. In addition, when developing computer vision or image processing applications, a microcontrollers are too slow compared to ASIC because the system require almost instantaneous response.

To perform this task it will require UART interface, and an optional SD card interface and expertise in image processing. The implementation must be very robust and flexible for future change.

The rest of this proposal contains extensive detail on the implementation of each block and their requirements, design specifications and image processing techniques.

2 System Usage Diagram



2.1 Operational Characteristics

This ASIC design is to be used in a camera to work as multiple filters and also an edge detector. For this the ASIC design would have to be connected to the camera's SD card. Once the picture is taken the ASIC design will take over and run the data through it along with an enable signal that will tell the ASIC design whether to run the photo through the edge detection process or the filtering process or both. If the photo sent to the filters the photo could be ran through one or multiple filters and saved back onto the SD card.

The over all design of this ASIC needs to have a clk, MISO, MOSI, and n_rst, These signals all have a dedicated function. The clk will run to ensure that the functions are not being performed on incomplete data and will not corrupt the data. As for the MISO and MOSI these signals are sent to and from the ASIC to an optional external memory source. These signals control the flow of data so the ASIC can handle many different sizes of images. Finally there is the n_rst signal. This signal is a fail safe to ensure that the process can be terminated if the image is not the right signal or is to large for our design.

Signal Name	Signal Type	Bits	Description
Clk	In	1	This is the clock of the system
MISO	In	48	This collects data from the external memory source
MOSI	Out	48	This sends the final image back to the external memory source
n_rst	In	1	This is the reset signal

To be memory and cost effective the image processor will process rows of image data at a time instead of waiting to read the entire image onto memory. Since Gaussian blur and edge detection algorithms require access to the surrounding 2-3 pixels (depending on mask size) there would be 5 rows that are currently required for processing and an extra 'n' rows (depending on cycles required for processing 1 pixel) that act as a buffer to allow reading data while the previously read image data is being processed. This way image receiving and processing occurs in parallel.

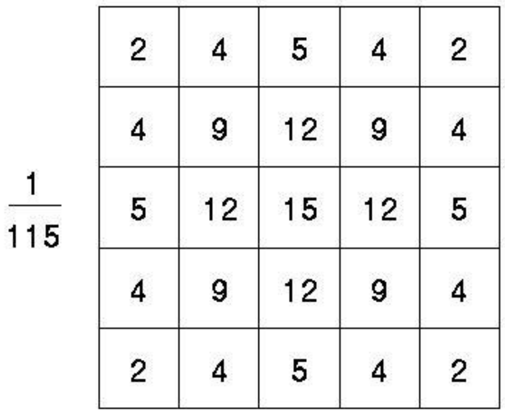
Block Specifications

1. ***UART transceiver***

   This blocks is responsible for receiving image data in bitmap format and also for transmitting the processed image stream. It interfaces with the controller to begin streaming processed image data.

2. ***Controller***
   The controller is responsible for the overall coordination of the image processor. It determines which pixels need to be processed in what sequence and when they are ready to be streamed. It also enables the necessary filter blocks so as to correctly apply the filter requested.
3. ***Gaussian Blur***

   This module applies a Gaussian blur on a pixel by looking at its surrounding pixels. Effectively it applies a "mask" that computes the pixel by averaging its neighboring pixels according to their "weights". An example of a mask is shown below:



$\frac{1}{115}$

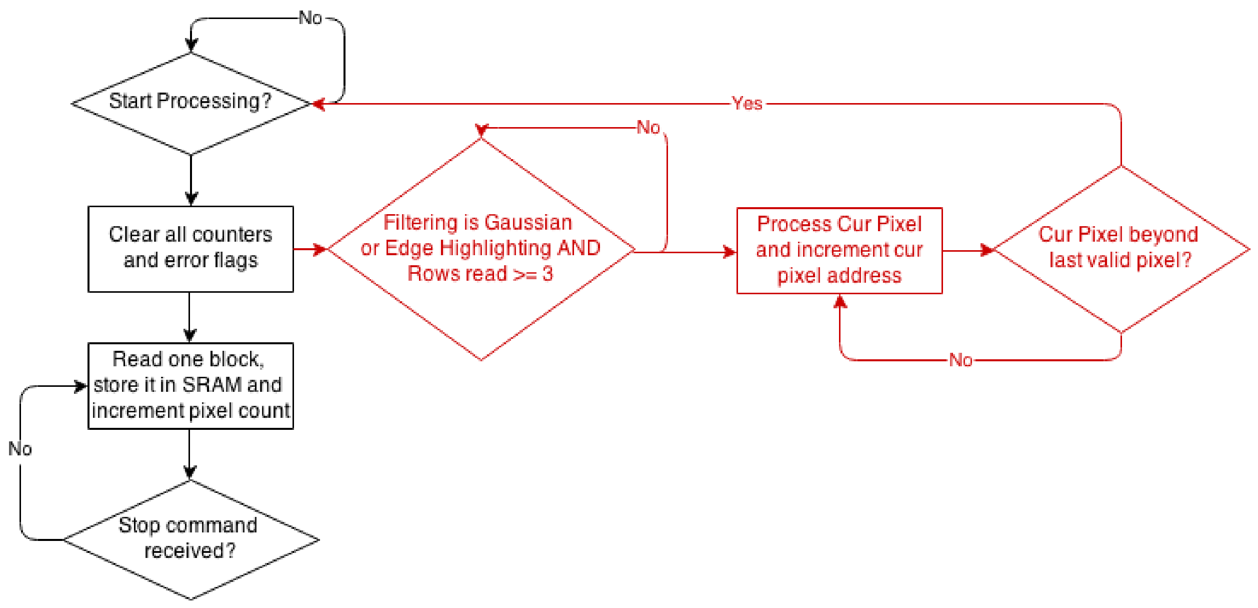
2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Figure 3 Discrete approximation to Gaussian function with σ=1.4

Source: http://dasl.mem.drexel.edu/alumni/bGreen/www.pages.drexel.edu/_weg22/can_tut.html

4. ***Inversion Filter***
   Inverts the image. Inversion of each pixel value is done by flipping the bits
5. ***Edge highlighting***
   This block detects edges in the image and overlays it over the original image to highlight the edges. Canny Edge detection algorithm is used for this purpose. Since the first step in any edge detection algorithm is to smoothen it, Gaussian blur is first applied before edge detection is performed.
6. ***Tinting Filter***
   Tints the image by manipulating the RGB values to highlight certain tones in the image.

Sequence of operations


Start Processing?
No
Clear all counters and error flags
Read one block, store it in SRAM and increment pixel count
Stop command received?
No
Filtering is Gaussian or Edge Highlighting AND Rows read >= 3
No
Process Cur Pixel and increment cur pixel address
Cur Pixel beyond last valid pixel?
Yes
No

Processing does not begin until at least 3 rows of image data has been read so as to be able to apply Gaussian blur with noticeable effect. Processed pixels are transmitted right away.

Gaussian blur computation for corner pixel (highlighted)

CurPixel	CurPixel	CurPixel	CurPixel+1	CurPixel+2
CurPixel	CurPixel	CurPixel	CurPixel+1	CurPixel+2
CurPixel	CurPixel	CurPixel	CurPixel+1	CurPixel+2
CurPixel+Width	CurPixel+Width	CurPixel+Width	CurPixel+Width+1	CurPixel+Width+2
CurPixel+2*Width	CurPixel+2*Width	CurPixel+2*Width	…	…

CurPixel is the address of the current pixel (here pixel 0)

For corner and edge cases in Gaussian blur and Edge highlighting, the image is padded with pixels from the edges to avoid a white glow near the edges.

2.2 Requirements

The main intended application of our ASIC Image Processor is primarily to provide a fast real-time image processing capability to optical devices, which have many industrial and commercial applications such as computer vision, manufacturing, digital or surveillance cameras and more. Hence, speed is the number one priority in our design. Secondary priority would be power consumption. Since the chip has a possibility of being utilized in factory condition, it must be able to withstand high temperature swing.

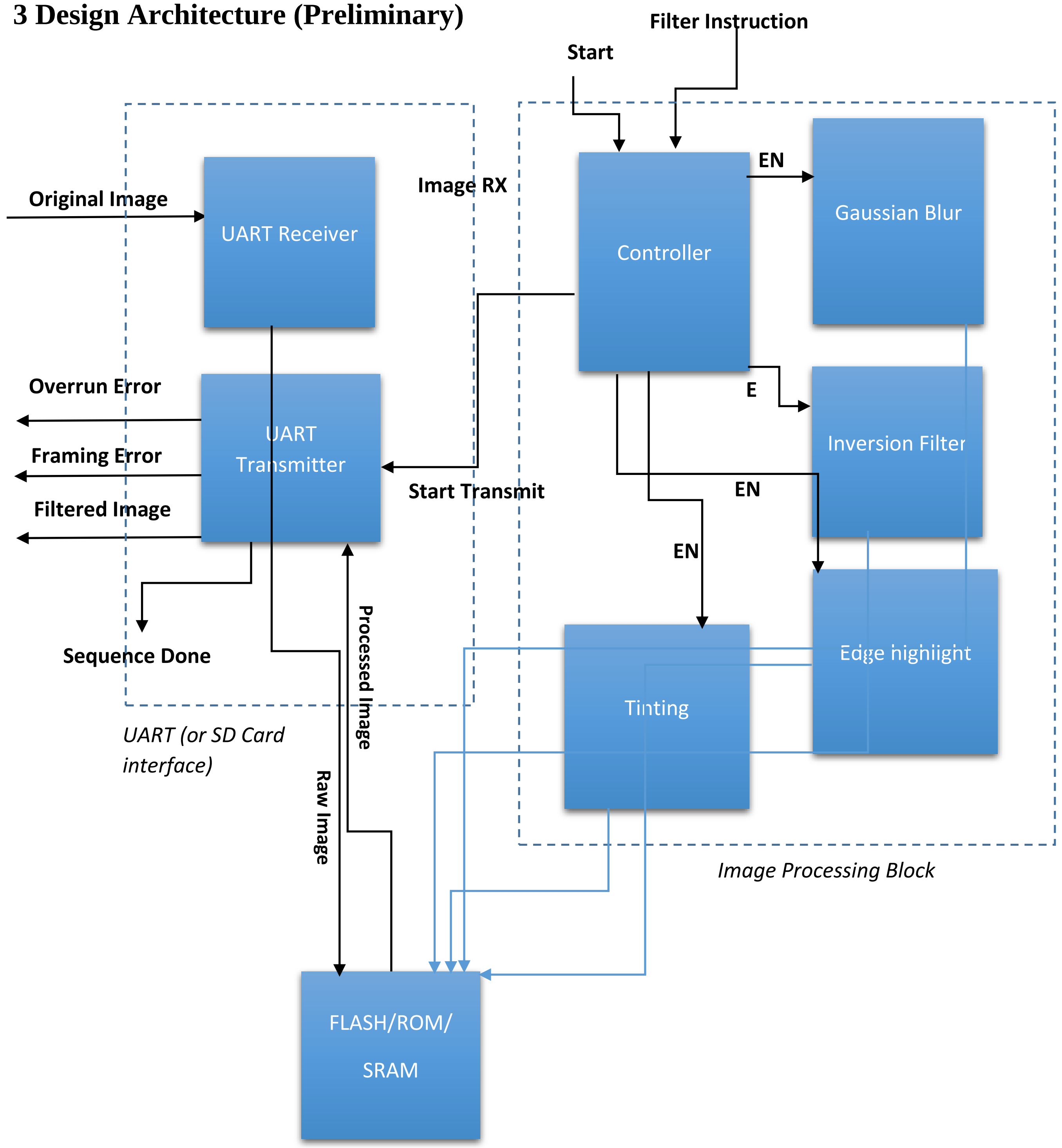
Area target: 1.5mm x 1.5mm

Pin count: 6

Max throughput: 230,400 bytes per second

Since the application of this chip in commercial and industrial environment may require handling of realistic images, we estimate the input and output images to be 8 bit deep (number of bits used to represent color in a single pixel), containing 2^8 (256) colors per pixel. So if we have a 720p image (1280x720), the image will contain 921,600 pixels and it would need 7,372,800 bits (921,600 bytes) the represent the entire image. Say we operate on 400 MHz clock and we transfer 1 bit/cycle (serial). We should be able to process 1/4 portion of the image every second. So approximately, a 720p bitmap image of depth 8 will be able to be processed at around 4 to 5 seconds.

3 Design Architecture (Preliminary)


Filter Instruction
Start
Original Image
UART Receiver
Image RX
Controller
EN
Gaussian Blur
Overrun Error
Framing Error
Filtered Image
UART
Transmitter
Start Transmit
E
Inversion Filter
EN
EN
Sequence Done
Processed Image
Edge highlight
Tinting
UART (or SD Card interface)
Raw Image
Image Processing Block
FLASH/ROM/
SRAM