CSEE 4840: FInal Project

Fatima Dantsoho
Michael Lanzano

Outline

- Overview
- Hardware Implementation
  - OV7670 camera configuration
  - Sliding Window & Sobel Filter
  - Dataflow Design
- Software Implementation
- Challenges & Lessons Learned
- Conclusion

Overview

A simple digital camera system in order to illustrate some of the main concepts related to digital design with **Verilog** and **FPGAs**, video formats, CMOS cameras, basic image processing algorithms (edge detection) embedded programming of microcontrollers.

Design Goal

Perform real-time edge detection and display on VGA monitor

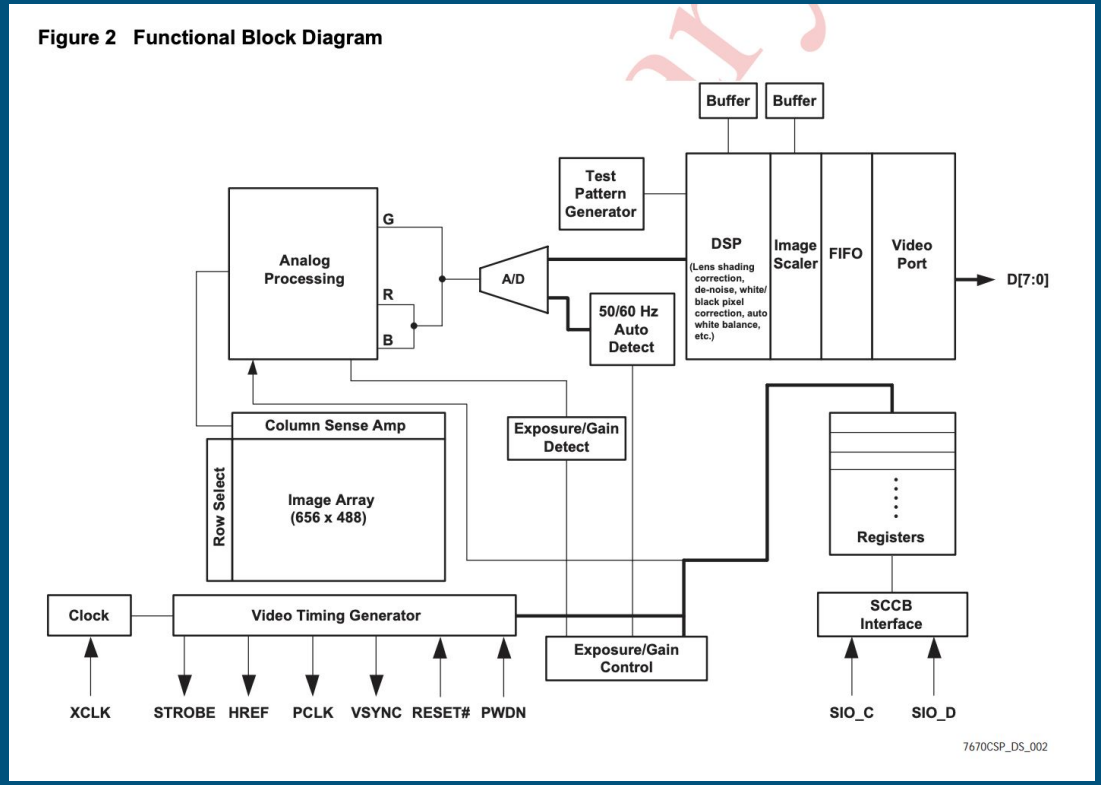
Background

- Edge detection with X and Y gradient Filters
  - kernel based convolution that processes an array of pixels from an input image to generate a single pixel in an output image
  - The Sobel kernel is designed to extract the gradient in an image in both X and Y direction

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \text{ and } G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$

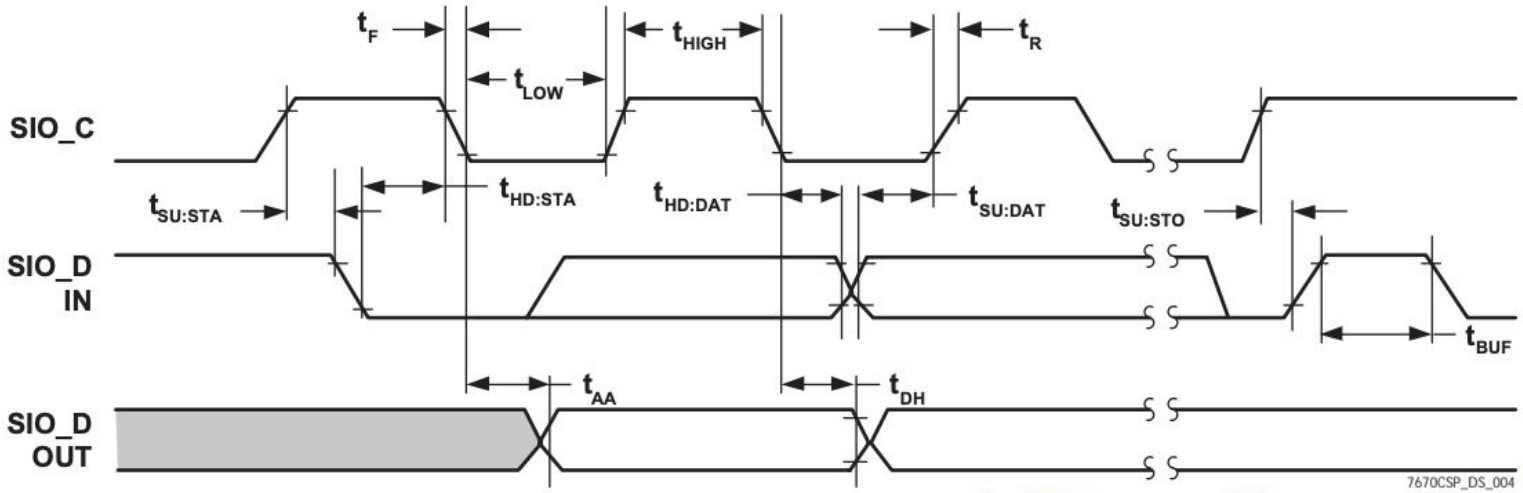
Equation (1)

OV7670 CMOS camera configuration

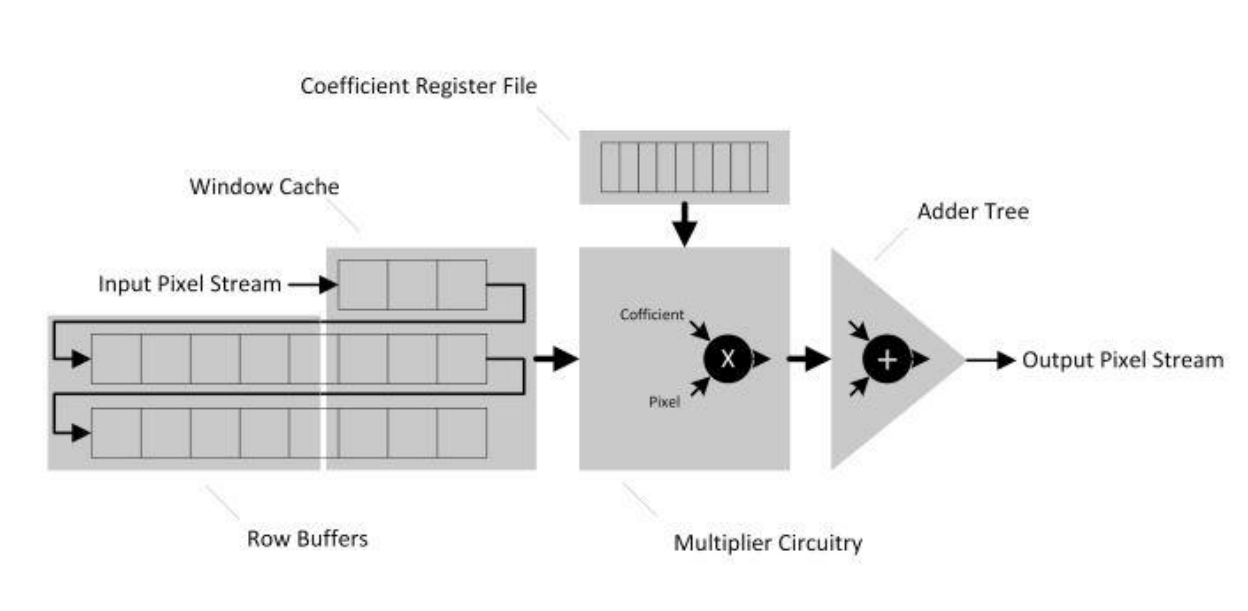


OV7670 CMOS camera configuration

Figure 4 SCCB Timing Diagram



Sliding Window & Sobel Filter



Sliding Window & Sobel Filter

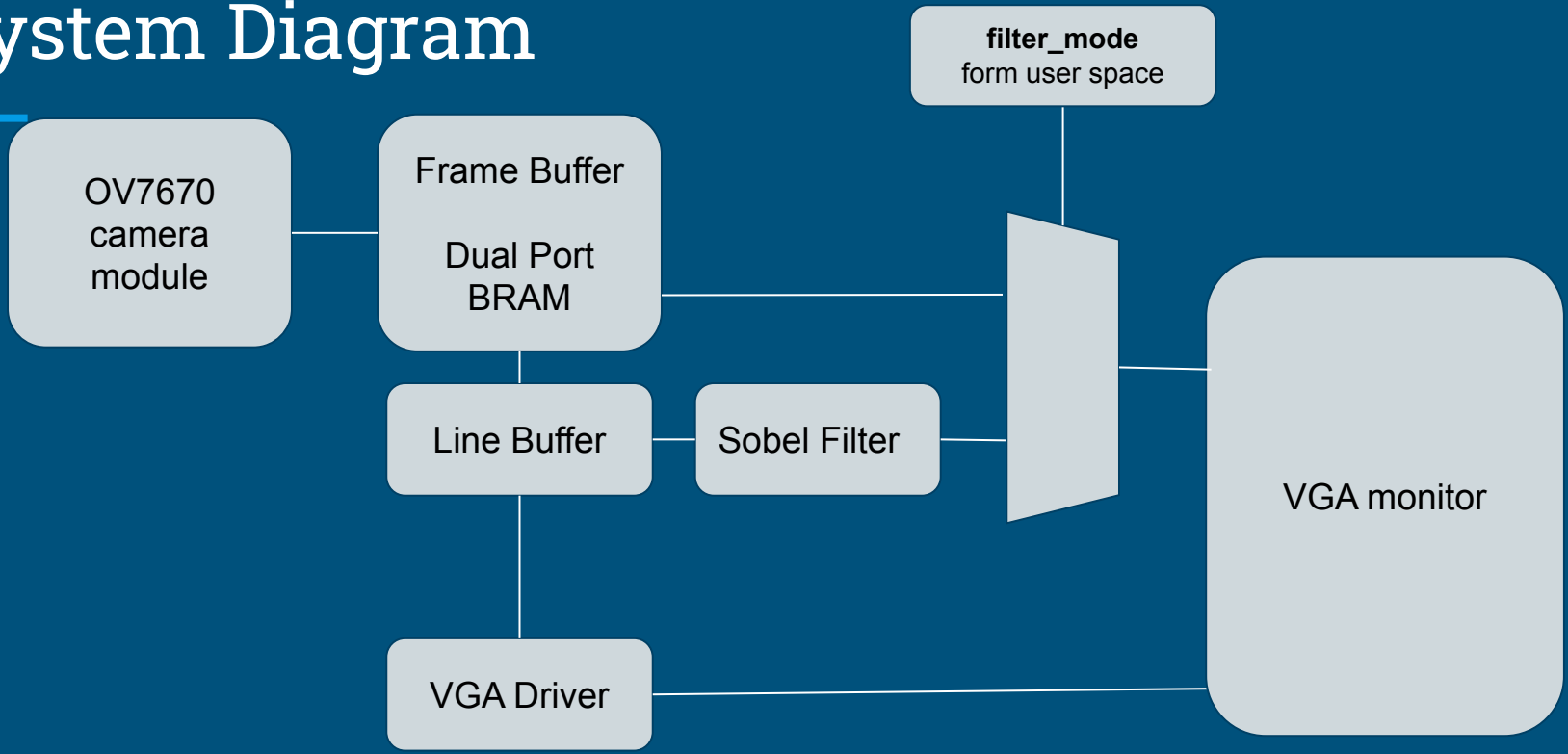
```
always_ff @(posedge clock)
  if(reset) begin
    ptr <=0;
    sliding[0][0] <= 0;
    sliding[0][1] <= 0;
    sliding[0][2] <= 0;
    sliding[1][0] <= 0;
    sliding[1][1] <= 0;
    sliding[1][2] <= 0;
    sliding[2][0] <= 0;
    sliding[2][1] <= 0;
    sliding[2][2] <= 0;
  end
  else begin
    sliding[0][0] <= inputPixel;
    sliding[1][0] <= sliding[0][0];
    sliding[1][1] <= sliding[0][1];
    sliding[1][2] <= sliding[0][2];
    sliding[2][0] <= sliding[1][0];
    sliding[2][1] <= sliding[1][1];
    sliding[2][2] <= sliding[1][2];

    buffer[ptr] <= sliding[BUFFER_SIZE-1][BUFFER_SIZE-2:0];
    sliding[0][BUFFER_SIZE-1:1] <= buffer[ptr];
    if(ptr < ROW_SIZE-BUFFER_SIZE) ptr <= ptr + 1;
        else ptr <= 0;
  end
```

```
 always_ff @(posedge clock)
   if (reset) begin
      gx1 <= 0;
      gx2 <= 0;
      gy1 <= 0;
      gy2 <= 0;
   end
   else begin
     gx1 <= sliding[0][0] + sliding[2][0] + (sliding[1][0]<<1);
     gx2 <= sliding[0][2] + sliding[2][2] + (sliding[1][2]<<1);
     gy1 <= sliding[0][0] + sliding[0][2] + (sliding[2][1]<<1);
     gy2 <= sliding[2][0] + sliding[2][2] + (sliding[0][1]<<1);
   end


logic [WORD_SIZE+1:0] gx, gy;
 always_comb begin
   if (gx1 > gx2) gx <= gx1-gx2;
    else gx <= gx2 - gx1;
   if (gy1 > gy2) gy <= gy1-gy2;
    else gy <= gy2-gy1;
 end
```

System Diagram



Challenges & Lessons Learned

- Timing synchronization
- Interfacing with SCCB
- Physical Wiring and Interference
- Generalized vs Specific Convolution

Q & A