READING ELECTIVE

FINAL UPDATE

IMPLEMENTING CENTROID ALGORITHM
ON FPGA

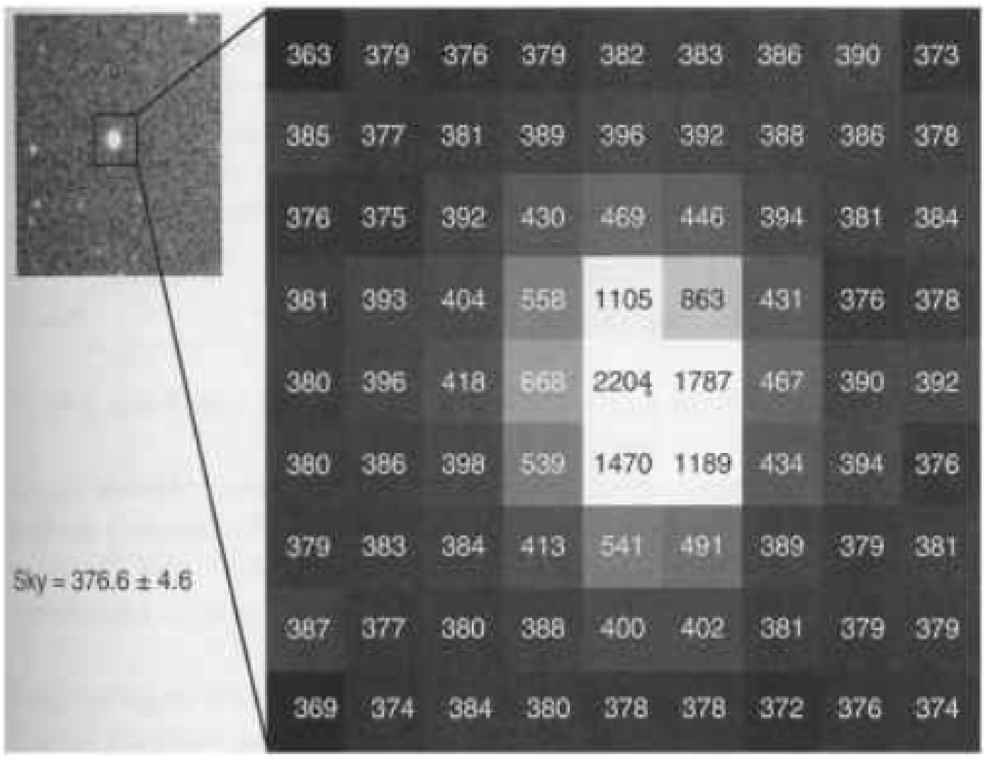
Pratyush Nandi – IMT2017518

Contents

1 Centroid Algorithm

The centroid of a physical object is its barycenter, or center of gravity. It is the point at which the object—suspended from a thread or teetering on a fulcrum-balances. In an image, the centroid is the point at which the values of all surrounding pixels are "balanced." The distribution of light among them enables us to recover the exact center of the star image within a small fraction of a pixel. The process is called determining a centroid.

The value of a star's centroid lies not in knowing exactly where on the CCD the light from a particular star happened to fall—which depends on factors that we cannot reproduce—but of determining the relative positions of the stars in the image, since the star images were formed at the same time. Even though every star image has been enlarged by diffraction, spread by atmospheric turbulence, and smeared by guiding errors, a precise record of their relative positions is stored in the image.

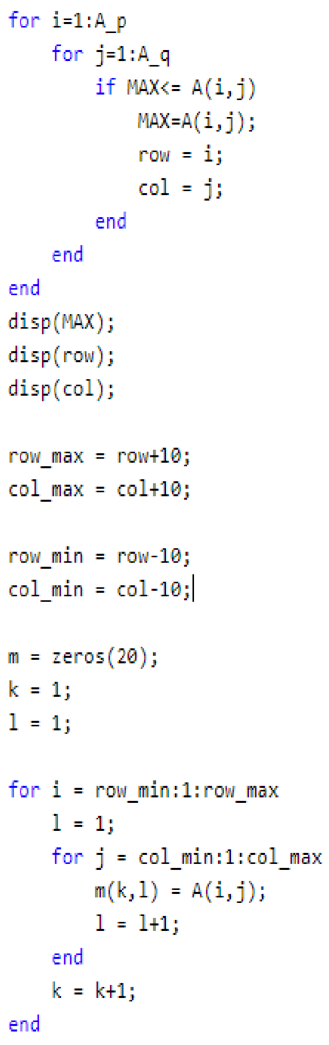


363	379	376	379	382	383	386	390	373
385	377	381	389	396	392	388	386	378
376	375	392	430	469	446	394	381	384
381	393	404	558	1105	863	431	376	378
380	396	418	668	2204	1787	467	390	392
380	386	398	539	1470	1189	434	394	376
379	383	384	413	541	491	389	379	381
387	377	380	388	400	402	381	379	379
369	374	384	380	378	378	372	376	374

Figure 1: Centroid of Image

2 MATLAB Implementation

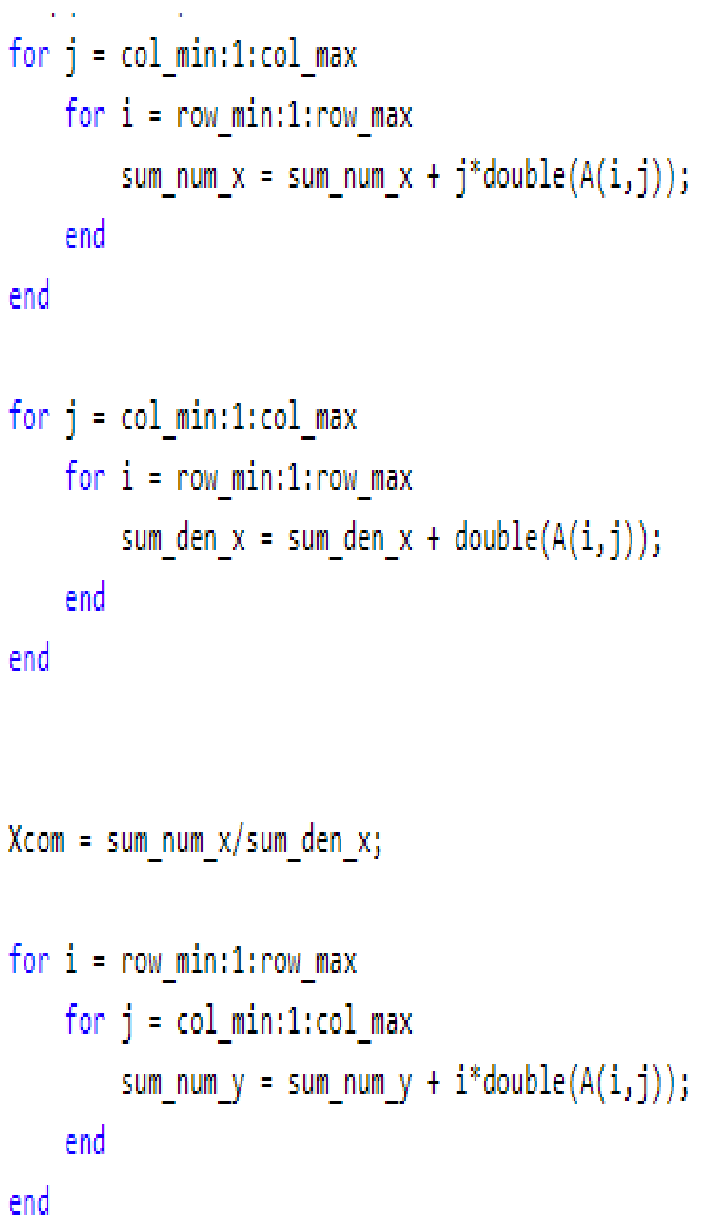
I used *MATLAB* as golden reference and verification purposes. The code has several parts. The first part is to calculate ROI(Region of Interest). To calculate ROI I found the maximum pixel present in the picture assuming it to be center of the star. Then once I find the maximum pixel the I extract a submatrix from the overall image and center of the submatrix is the pixel with highest intensity. The snippet of the code for calculating submatrix is attached in figure 2.



```
for i=1:A_p
    for j=1:A_q
        if MAX<= A(i,j)
            MAX=A(i,j);
            row = i;
            col = j;
        end
    end
end
disp(MAX);
disp(row);
disp(col);

row_max = row+10;
col_max = col+10;

row_min = row-10;
col_min = col-10;

m = zeros(20);
k = 1;
l = 1;

for i = row_min:1:row_max
    l = 1;
    for j = col_min:1:col_max
        m(k,l) = A(i,j);
        l = l+1;
    end
    k = k+1;
end
```

Figure 2: MATLAB code snippet for calculating ROI

Now to calculate centroid of image I use this submatrix. The algorithm is extracted from a paper mentioned in the reference. The snippet of the code is attached in figure 3.



```
for j = col_min:1:col_max
    for i = row_min:1:row_max
        sum_num_x = sum_num_x + j*double(A(i,j));
    end
end

for j = col_min:1:col_max
    for i = row_min:1:row_max
        sum_den_x = sum_den_x + double(A(i,j));
    end
end



Xcom = sum_num_x/sum_den_x;

for i = row_min:1:row_max
    for j = col_min:1:col_max
        sum_num_y = sum_num_y + i*double(A(i,j));
    end
end
```

Figure 3: MATLAB code snippet for calculating Centroid

3 Verilog Implementation

The MATLAB code was taken as foundation code. The hardware implementation was done on Xillinx platform. The synthesizable code was burnt on to a **basys3** fpga board.

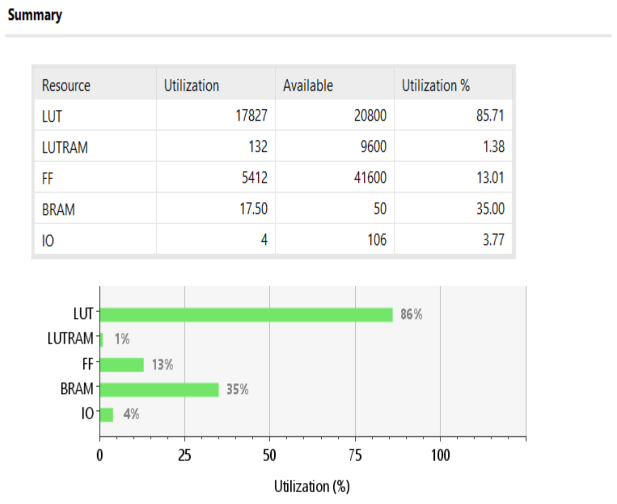
3.1 Functional Description & Hierarchy

The code is divided into several modules for making it more understandable and feasible. The functional description of the modules are as following:

- *topcontroller*: This module is the central module of our code. It calls other modules such as Uart and several IPs. This code initializes registers, matrices and blockrams. The crux of the code lies in the FSM that we have designed. The 6 state FSM first receives data from pc through the UART and fills it in the image blockram. Then next state uses a sophisticated algorithm to extract a sub-matrix from a single dimension blockram. The next state uses a trigger flag to calculate centroid.
- *Uart*: This module is mainly for transferring and receiving data from pc to fpga and viceversa.

3.2 Summary of Resource Utilization

In figure 4 we can see the summary of resource utilization post synthesis.

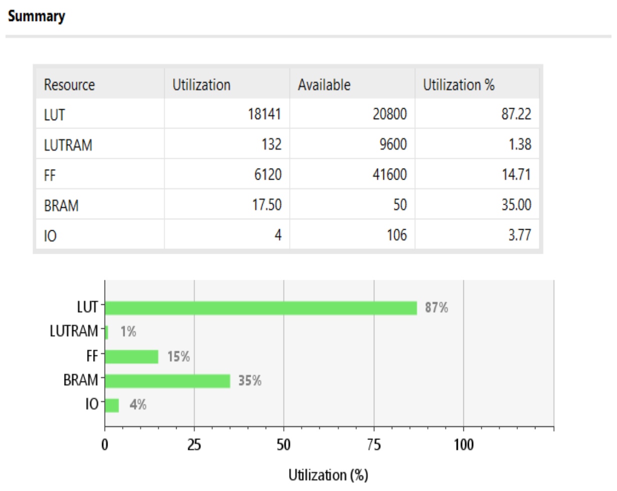


Summary

Resource	Utilization	Available	Utilization %
LUT	17827	20800	85.71
LUTRAM	132	9600	1.38
FF	5412	41600	13.01
BRAM	17.50	50	35.00
IO	4	106	3.77

Figure 4: Resource Utilization Post Synthesis

In figure 5 we can see the summary of resource utilization post implementation.



Summary

Resource	Utilization	Available	Utilization %
LUT	18141	20800	87.22
LUTRAM	132	9600	1.38
FF	6120	41600	14.71
BRAM	17.50	50	35.00
IO	4	106	3.77

Figure 5: Resource Utilization Post Implementation

In figure 6 we can see device mapping done on Basys3 Board.

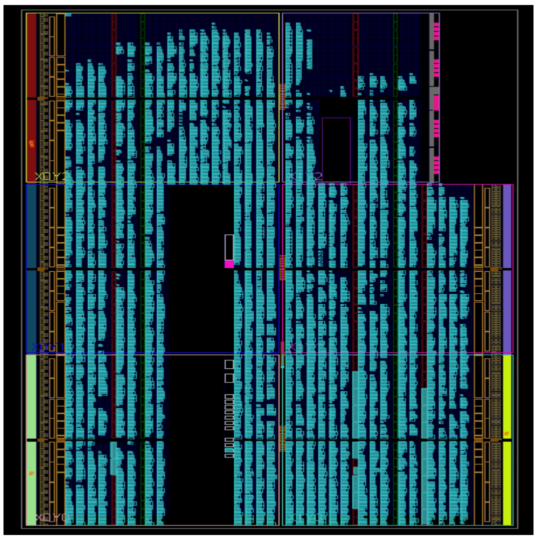


Figure 6: Device Mapping

3.3 Test-Bench

I used a test-bench to verify our working of the code. I sent data through a loop to the uart and received data from uart. Main purpose the implementation was to verify working of my code and as mentioned I used MATLAB as my golden reference.

4 Results

The final output of my project didn't work. Xcom and Ycom gave me FF in hex. This means there are some XX values while transfering the data. Simulation part worked perfectly. Current status: **Incomplete**

5 References

1. https://www.astronomyclub.xyz/image-processing/determining-a-centroid.html
2. Papers given by Nanditha Ma'am
3. https://github.com/pratyush48/Project$_Centroid$