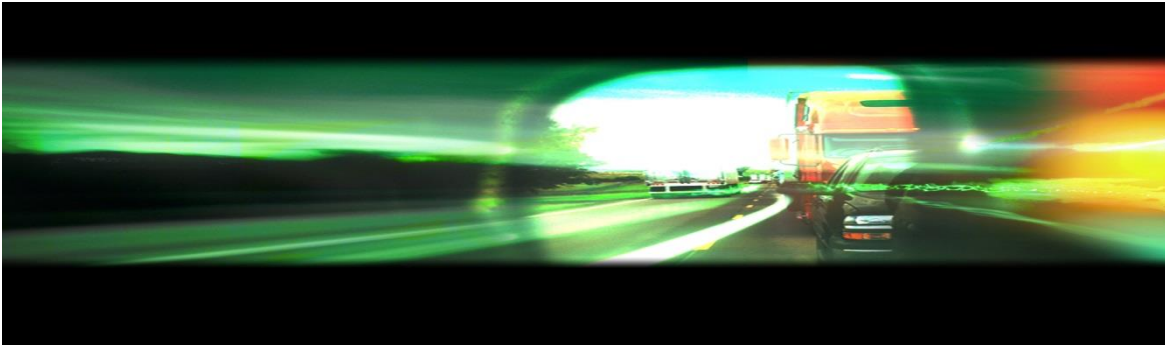


**A HIGH-SPEED TRAPEZOID IMAGE SENSOR
DESIGN FOR CONTINUOUS TRAFFIC MONITORING
AT SIGNALIZED INTERSECTION APPROACHES**

Final Report



TranLIVE

Suat U. Ay

October 2014

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle A High-Speed Trapezoid image sensor design for continuous traffic monitoring at signalized intersection approaches		5. Report Date October 2014	
		6. Source Organization Code KLK910	
7. Author(s) Ay, Suat U.		8. Source Organization Report No. N14-17	
9. Performing Organization Name and Address TranLIVE & NIATT University of Idaho 875 Perimeter Dr. MS0901 Moscow, ID 83844-0901		10. Work Unit No. (TRAIS)	
		11. Contract or Grant No. DTRT12GUTC17	
12. Sponsoring Agency Name and Address US Department of Transportation Research and Special Programs Administration 400 7th Street SW Washington, DC 20509-0001		13. Type of Report and Period Covered Final Report: 08/01/2013 – 09/30/2014	
		14. Sponsoring Agency Code USDOT/RSPA/DIR-1	
15. Supplementary Notes:			
16. Abstract The goal of this project is to monitor traffic flow continuously with an innovative camera system composed of a custom designed image sensor integrated circuit (IC) containing trapezoid pixel array and camera system that is capable of intelligent future extractions. The new trapezoid CMOS image sensor IC was designed, fabricated, and tested. New feature extraction algorithms for moving object monitoring was developed and implemented in field programmable gate array (FPGA) platform. A camera system composing of FPGA platform to run the algorithms and control the trapezoid imager was developed.			
17. Key Words Image sensor, feature extraction, traffic monitoring, trapezoid image sensor, video detection		18. Distribution Statement Unrestricted; Document is available to the public through the National Technical Information Service; Springfield, VT.	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 53	22. Price ...

TABLE OF CONTENTS

FIGURES ii

TABLES iii

EXECUTIVE SUMMARY 1

DESCRIPTION OF PROBLEM..... 2

APPROACH AND METHODOLOGY 3

 Geometric Design of TZOID Pixel Array 4

 TZOID Imager Integrated Circuit (IC) Design..... 10

 Pixel Circuits..... 11

 Column Readout 11

 Analog-to-Digital Converter (ADC) and Memory Blocks 12

 Reference Generator and DACs..... 13

 Other Blocks 13

 Integration 13

 Algorithm Design and Development 14

 Platform Selection..... 14

 Algorithm Selection 15

 Algorithm Design..... 16

 Prototype Development 18

 OpenCV Development..... 18

 Hardware Design 19

 Verification Process 21

 Testbed Development 21

 Graphical User Interface Development 22

 Advantages of Proposed Approach..... 23

FINDINGS; CONCLUSIONS; RECOMMENDATIONS 26

 Conclusions..... 26

 Observations 26

 Future Directions 27

REFERENCES 28

PROJECT DEMO LINKS 28

APPENDIX..... 29

Low-Level Module Block Diagrams for Hardware Implementation	29
Command Interpreter	29
Image Processor	29
Speed Detection Module.....	30
Write Operational Register Module.....	31
Write Tripwire Registers Module	32
Frame Reader and FTDI Driver	32
Request Current Frame Module.....	34
Generalized Request Module.....	34
Circuit Diagrams of the TZOID Imager IC	37
TZOID Imager and Camera Pictures	48
Student Algorithm Design Poster (UI Engineering EXPO, May 2014)	53

FIGURES

Figure 1: a) Two lane incoming road to an intersection, b) trapezoid pixel array.....	3
Figure 2: Critical measures and geometry on xy-direction for TZOID imager design.	4
Figure 3: Pixel height (dx_{xyi}) and distance resolution versus observation distance.	7
Figure 4: Relative distance measurement error.	7
Figure 5: Absolute distance measurement error.	8
Figure 6: Viewing perspective of the camera sensor on xz plane.....	9
Figure 7: The TZOID imager dimensions.	9
Figure 8: The TZOID imager pixel width (dx_{zi}) and height (dx_{yi}) versus row numbers.	10
Figure 9: Circuit block diagram of the TZOID imager.....	10
Figure 10: Pixel photodiode capacitances of the TZOID imager versus row address.....	11
Figure 11: a) Raspberry Pi single board computer, b) FPGA development board.	14
Figure 12: a) The current frame, b) the foreground frame after processing [4].....	16
Figure 13: Delta cap thresholding method (DCTM) for background generation.	17
Figure 14: Tripwire method for data reduction and speed detection.	18
Figure 15: OpenCV testing of the DCTM and Tripwire algorithms on a live image:.....	19
Figure 16: Complete system block diagram for hardware implementation.....	20
Figure 17: Testbed block diagram and final picture of the testbed.....	21
Figure 18: Developed GUI screenshots.	22

TABLES

Table 1: Gain Settings of the Programmable Column Charge Amplifier..... 12

EXECUTIVE SUMMARY

Most video detection systems currently used in traffic signal operations cannot provide the continuous traffic detection and monitoring needed for several advanced traffic control applications. They use virtual zone or line approach to extract and to identify existence of vehicles in the field of view (FOV). The goal of this project was to monitor traffic flow continuously with an innovative camera system composed of a custom designed image sensor integrated circuit (IC) containing trapezoid pixel array and intelligent feature extraction engine. Continuous monitoring is referred to as the detection of vehicle speed, location, and acceleration in FOV with a camera continuously. It is desirable that this is done on image sensor level intelligently without requiring complicated digital post processing of video streams and high communication bandwidth associated with it. The developed trapezoid image sensor IC aims to reduce the overall cost and complexity of the monitoring system while providing intelligent and continuous incoming traffic information while opening doors for several dynamic traffic signal control applications, reducing vehicle emissions and fuel consumption [1][2]. More specific objectives are to develop technology that consumes lower power, requires lower communication bandwidth and local memory while processing captured images with higher frame rates ($>100\text{FPS}$) by using less powerful processing units (microprocessors, CPUs, or FPGAs).

The new trapezoid (TZOID) image sensor developed during this project addresses the set objectives with drastically new technology. It composes of a 40×152 array of complementary metal-oxide-semiconductor (CMOS) active pixel sensors (APS) with scaled pixel sizes between $3.4\mu\text{m}$ and $48.7\mu\text{m}$. The TZOID imager was designed assuming the camera system is mounted on a traffic light pole 28.5ft above ground while monitoring traffic flow up to 800ft away. Image feature extraction algorithms “Delta Cap Thresholding Method (DCTM)” and “Tripwire Method” were developed and implemented using hardware definition language (HDL) of Verilog on a field programmable gate array (FPGA) development platform for future implementation in application specific integrated circuit (ASIC) that would contain a new TZOID image sensor.

DESCRIPTION OF PROBLEM

Control of traffic signals mostly depends on monitoring vehicles on signalized intersection approaches. Correct detection methods used in signalized intersection approaches (whether loop or video detectors) are based on binary detection at specific points on the approach. These methods do not provide dynamic properties of approaching vehicles such as location, speed, acceleration/deceleration, and expected time of arrival at the stop bar location. If the traffic controller has access to this information, it would optimize traffic flow achieving better efficiency by minimizing delay and elimination of unnecessary stops. Continuous monitoring of traffic can be done by using several methods. One method is to use complicated radar systems that might be prohibitively expensive to be utilized in intersections. Another method is to use camera systems mounted on the traffic light poles. Current state of the art image sensor integrated circuits (ICs) used in these relatively cost effective camera systems is not optimized for traffic flow monitoring. They only generate video streams and backend high-performance digital signal processors (DSPs) or microprocessors take care of extracting intelligent information such as location and speed of the vehicles from the high bandwidth video streams. These camera sensors have fixed pixel and array sizes and frame rates are not always optimum for fast and intelligent information extraction. An intelligent image sensor that is designed specifically for extracting “only” the dynamic properties of incoming traffic and the vehicles with high resolution, high speed, and embedded intelligence on sensor level is needed.

APPROACH AND METHODOLOGY

The proposed new camera system will be mounted on a traffic light pole (typically 29ft high) on the opposite side of the stop line of the intersection (typically 24-48ft away) looking directly toward two (or four) lanes (24-48ft wide) of incoming traffic. A typical camera system monitoring at such an intersection is composed of an image sensor with constant pixel size (p_i). If it was to provide intelligent information about the vehicles at the far end of the FOV (vehicle C in Figure 1a), it would require very high resolution pixels which are not needed for vehicles in the near field (vehicle A in Figure 1a). A pixel array with variable pixel sizes (Figure 1b), covering trapezoid FOV (Figure 1a) will obtain the speed, location and acceleration of an incoming vehicle linearly and accurately. We called the image sensor with changed 2D pixel size a trapezoid image sensor or TZOID for short. For the TZOID image sensor design, it is necessary to scale pixel sizes at the near end of the focal plane to be small enough to detect vehicle license plates (6" x 18"). In the camera system, it is also necessary to capture only the road in the FOV which is in trapezoid shape. In the proposed image sensor, reducing the pixel sizes towards the far end of the FOV provides higher resolution for those parts of the FOV.

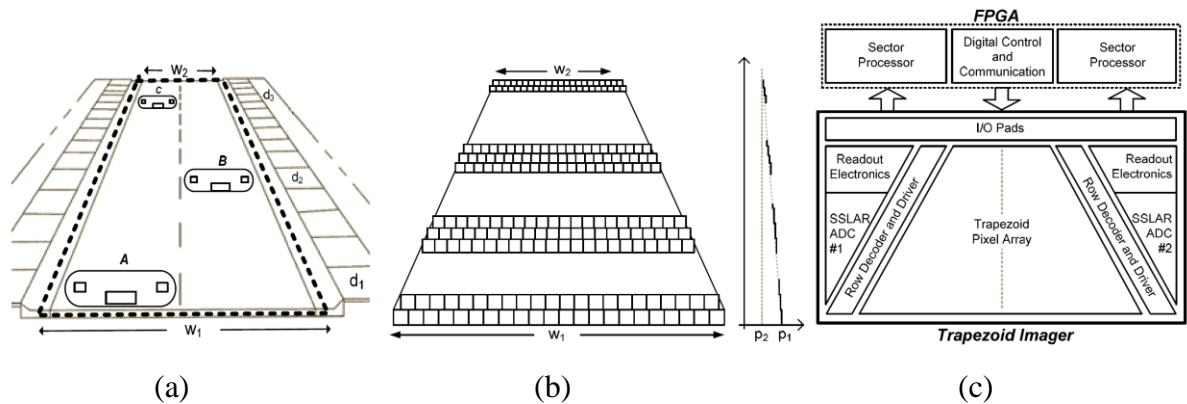


Figure 1: a) Two lane incoming road to an intersection, b) trapezoid pixel array with scaled pixel sizes from large (p_1) at the near end to small (p_2) at the far end, c) block diagram of the proposed trapezoid image sensor for continuous traffic monitoring.

The new TZOID image sensor composes of: a) standard 3 transistor (3T) CMOS APS pixels, b) trapezoid pixel array with scaled size pixels, c) high speed readout electronics including single-slope ramp type analog-to-digital converters (ADCs) with programmable bit resolution (6-bit or 8-bit). A general purpose FPGA board with USB connectivity to PC is

used for implementing intelligent future extraction algorithms and low bandwidth communication. The TZOID imager was fabricated in a standard $0.18\mu\text{m}$ CMOS process with trapezoid pixel array (40×152) running up to 100 frames per second allowing accurate extracting traffic and vehicle information every 10ms.

Geometric Design of TZOID Pixel Array

This section provides details about the geometry and equations for designing trapezoid image sensor array. Figure 2 shows the geometry for xy-direction.

A lens with effective focal length (EFL) of R_f is placed at $(0, h_p)$ coordinates with viewing angle of β . Sensor is located at the EFL of the camera lens. Camera is focused at the mid distance (d_m) between the stopping lane (d_0) and the far observation point on the road (d_1). The h_p is height of the traffic pole where the camera system is mounted. Pixel size depends on focal length of the lens system. Based on the distance resolutions at stopping lane (dx_0) and the distance resolutions on far end (dx_1), minimum (dx_{xyp1}) and maximum pixel sizes (dx_{xyp0}) on xy-directions are calculated with respect to EFL (R_f) which can be taken from lens datasheets. Assuming $dx_i = dx_0$, we can find pixel size dx_{xypi} using the following equations:

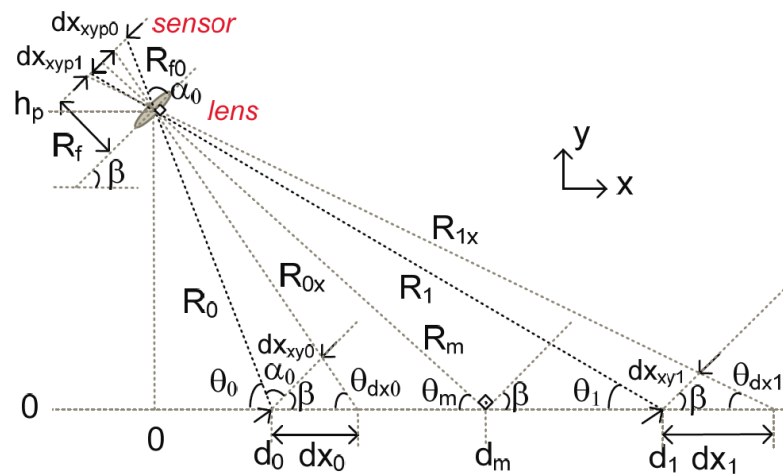


Figure 2: Critical measures and geometry on xy-direction for TZOID imager design.

$$\frac{R_{f0}}{R_0} = \frac{dx_{xyp0}}{dx_{xy0}} \quad (1)$$

$$\frac{R_{fi}}{R_i} = \frac{dx_{xyfi}}{dx_{xyi}} \quad (2)$$

$$R_{f0} = \frac{R_f}{\sin(\alpha_0)} \quad (3)$$

$$R_{fi} = \frac{R_f}{\sin(\alpha_i)} \quad (4)$$

$$\alpha_i = \pi - \beta - \theta_i \quad (5)$$

$$\beta = \frac{\pi}{2} - \theta_m \quad (6)$$

$$\theta_m = \text{atan}\left(\frac{h_p}{d_m}\right) = \text{atan}\left(\frac{2 \cdot h_p}{d_0 + d_1}\right) \quad (7)$$

$$\theta_m = \text{atan}\left(\frac{2 \cdot h_p}{d_0 + d_1}\right) \quad (8)$$

$$\beta = \frac{\pi}{2} - \theta_m \quad (9)$$

$$\beta = \frac{\pi}{2} - \text{atan}\left(\frac{2 \cdot h_p}{d_0 + d_1}\right) \quad (10)$$

$$\theta_i = \text{atan}\left(\frac{h_p}{d_i}\right) \quad (11)$$

$$\theta_{dxi} = \text{atan}\left(\frac{h_p}{d_i + dx_0}\right) \quad (12)$$

$$\alpha_i = \frac{\pi}{2} + \theta_m - \theta_i \quad (13)$$

$$\alpha_i = \frac{\pi}{2} + \text{atan}\left(\frac{2 \cdot h_p}{d_0 + d_1}\right) - \text{atan}\left(\frac{h_p}{d_i}\right) \quad (14)$$

$$dx_{xyi} = \frac{\sin(\theta_{dxi}) \cdot dx_0}{\sin(\pi - \beta - \theta_{dxi})} = \frac{\sin\left(\operatorname{atan}\left(\frac{h_p}{d_i + dx_0}\right)\right) \cdot dx_0}{\sin\left(\frac{\pi}{2} + \operatorname{atan}\left(\frac{2 \cdot h_p}{d_0 + d_1}\right) - \operatorname{atan}\left(\frac{h_p}{d_i + dx_0}\right)\right)} \quad (15)$$

$$dx_{xypi} = \frac{R_f \cdot dx_{xyi}}{R_i} = \frac{R_f \cdot dx_{xyi}}{R_i \cdot \sin(\alpha_i)} \quad (16)$$

Thus,

$$dx_{xypi} = \frac{R_f \cdot \sin\left(\operatorname{atan}\left(\frac{h_p}{d_i + dx_0}\right)\right) \cdot dx_0}{\sqrt{d_i^2 + h_p^2} \cdot \sin\left(\frac{\pi}{2} + \operatorname{atan}\left(\frac{2 \cdot h_p}{d_0 + d_1}\right) - \operatorname{atan}\left(\frac{h_p}{d_i}\right)\right) \cdot \sin\left(\frac{\pi}{2} + \operatorname{atan}\left(\frac{2 \cdot h_p}{d_0 + d_1}\right) - \operatorname{atan}\left(\frac{h_p}{d_i + dx_0}\right)\right)} \quad (17)$$

$$dx_{xypi} = \frac{R_f \cdot \sin\left(\operatorname{atan}\left(\frac{h_p}{d_i + dx_0}\right)\right) \cdot dx_0}{\sqrt{d_i^2 + h_p^2} \cdot \cos\left(\operatorname{atan}\left(\frac{2 \cdot h_p}{d_0 + d_1}\right) - \operatorname{atan}\left(\frac{h_p}{d_i}\right)\right) \cdot \cos\left(\operatorname{atan}\left(\frac{2 \cdot h_p}{d_0 + d_1}\right) - \operatorname{atan}\left(\frac{h_p}{d_i + dx_0}\right)\right)} \quad (18)$$

For a given pixel height (dx_{xypi}), it is necessary to calculate distance resolution (dx_i). Distance resolution (dx_i) can be found by using (18) and by iteration.

Figure 3 shows one design assuming a focal length of camera optics is 4.2mm, four lanes of city road, and observation distance of 265m from stopping lane while stopping lane is 14.6m away from the sensor. Effective viewing distance from the camera location is about 280m (279.6m to be exact). Design has six sections with 0.25m, 0.5m, 1m, 2m, 5m, and 10m per pixel distance resolutions. Pixel sizes (width and/or height) vary between 48.7 μ m and 3.4 μ m to accommodate the section distance resolutions assuming focal length of the camera optics is 4.2mm.

Pixel heights (on xy-direction) are adjusted to fit into the layout design grid of the fabrication process. For example, design equation (18) results in a pixel height of 3.413 μ m but process grid allows only 3.40 μ m or 3.45 μ m pixel size causing measurement error due to rounding.

After a manufacturable pixel size design/adjustment, distance measurement errors are calculated as shown in Figure 4 and Figure 5.

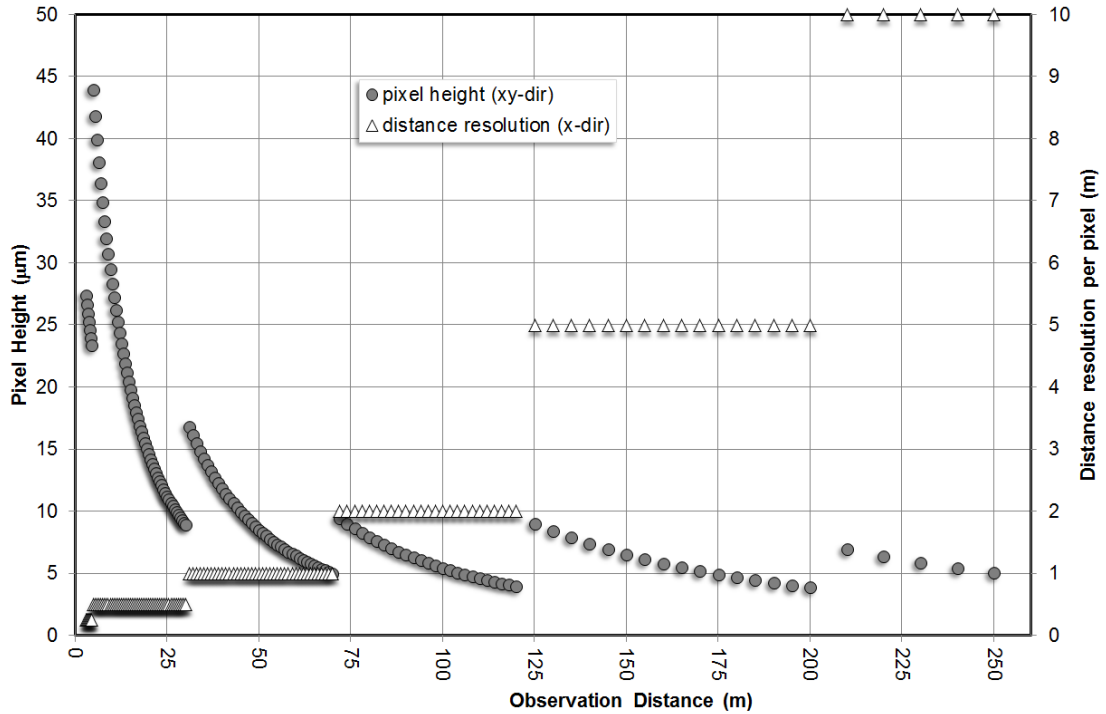


Figure 3: Pixel height (dx_{xyi}) and distance resolution versus observation distance on xy -direction for the proposed TZOID imager.

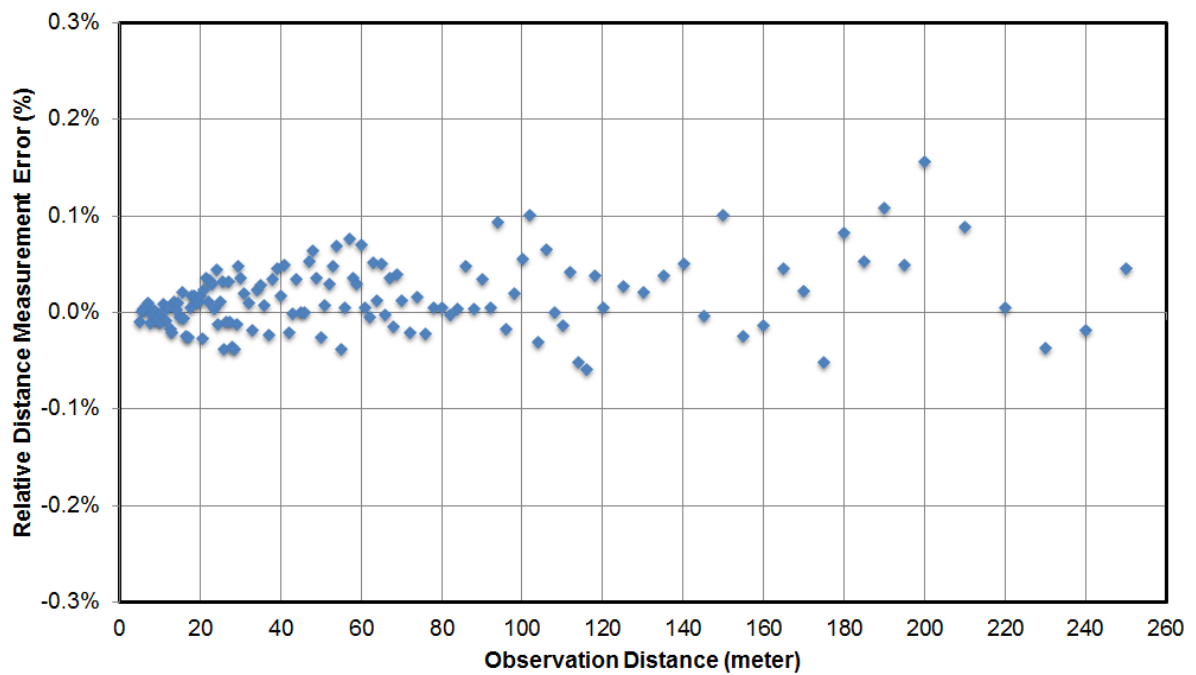


Figure 4: Relative distance measurement error.

Expected measurement error is less than $\pm 1.0\%$ of the absolute distance resolution on the road. The design results in measurement error (on x-direction) of $\pm 1\text{cm}$ for far end of the observation field where per pixel measurement distance is 10m. At closer end of the observation field (the stopping lane) this is 1mm with 0.25m resolution.

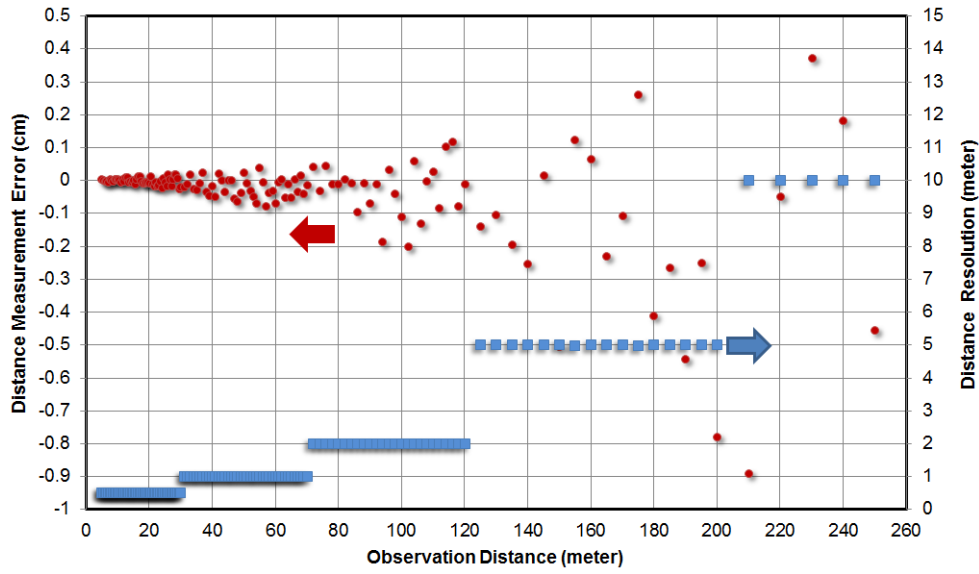


Figure 5: Absolute distance measurement error.

Pixel size on z-direction can be different than the pixel size on xy-direction that would result in rectangular pixels. Assuming the camera is placed at the middle of the road, and focal length (R_f) of the optics is known, than the minimum and maximum pixel sizes can be determined for the trapezoid imager. Figure 6 show the viewing perspective of the imager on z-direction. Pixel width on the focal plane is drawn and given by equation (21).

$$R_{fmi} = R_i \sin(\alpha_i) = \sqrt{h_p^2 + d_i^2} \cdot \sin\left(\frac{\pi}{2} + \text{atan}\left(\frac{2 \cdot h_p}{d_0 + d_1}\right) - \text{atan}\left(\frac{h_p}{d_i}\right)\right) \quad (19)$$

$$\frac{R_f}{R_{fmi}} = \frac{dx_{zpi}}{dx_z} \quad (20)$$

$$dx_{zpi} = \frac{R_f \cdot dx_z}{R_{fmi}} = \frac{R_f \cdot dx_z}{\sqrt{h_p^2 + d_i^2} \cdot \sin\left(\frac{\pi}{2} + \text{atan}\left(\frac{2 \cdot h_p}{d_0 + d_1}\right) - \text{atan}\left(\frac{h_p}{d_i}\right)\right)} \quad (21)$$

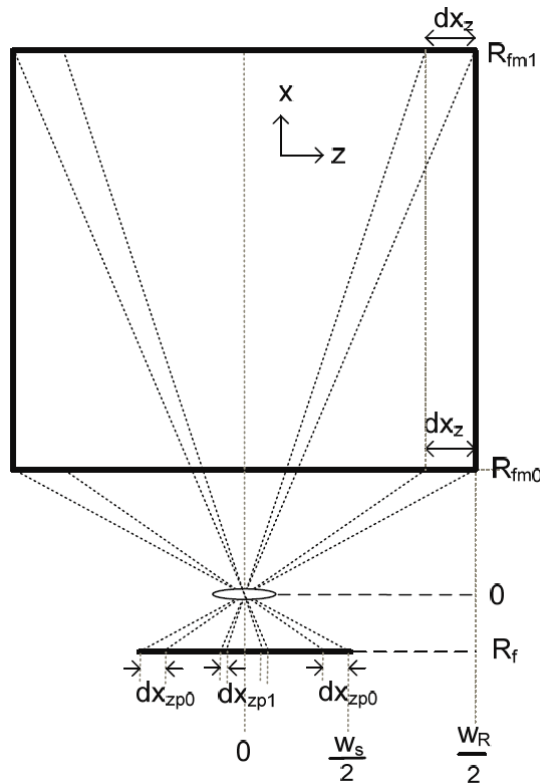


Figure 6: Viewing perspective of the camera sensor on xz plane.

The observation resolution per pixel on z -direction is set 0.21m for the design. This resulted in a maximum pixel width (dx_{zpi}) of $48.7\mu\text{m}$. This limits the number of columns for the imager. Based on the maximum pixel width, and assuming 10 columns per lane and maximum four lanes to be monitored, the number of imaging pixel columns is set to 40 ($N=40$ in Figure 7). Number of rows on the TZOID imager is set to 152 ($M=152$ in Figure 7). Of these pixels the first five (0-4) and last three rows (149-151) are covered for dark level measurement and correction. Resulting pixel sizes are shown in Figure 8.

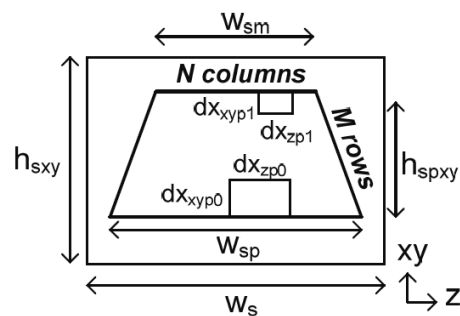


Figure 7: The TZOID imager dimensions.

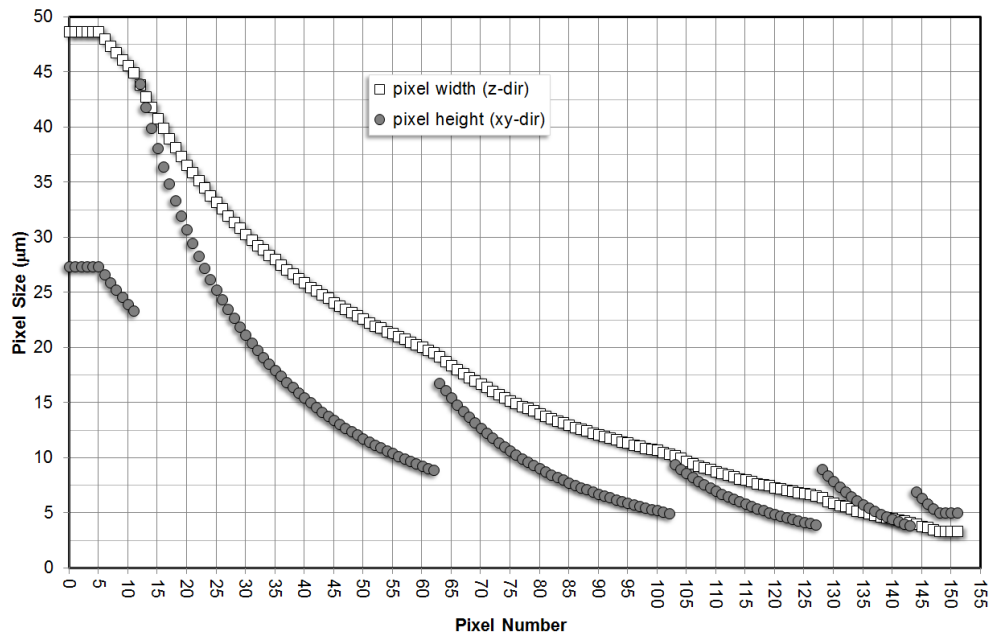


Figure 8: The TZOID imager pixel width (dx_{zi}) and height (dx_{yi}) versus row numbers.

TZOID Imager Integrated Circuit (IC) Design

Circuit block diagram of the TZOID imager is shown in Figure 9 including the details of the APS pixel and column readout circuits. Details of the critical building blocks are given in this section while circuit diagrams are in the Appendix.

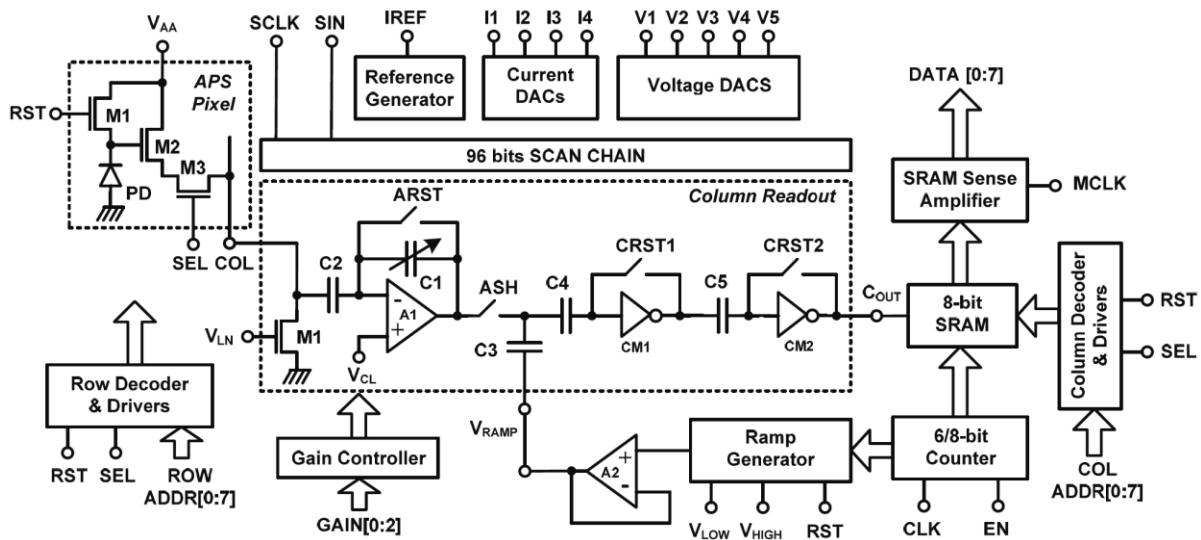


Figure 9: Circuit block diagram of the TZOID imager.

Pixel Circuits

Standard CMOS APS pixel containing three metal-oxide-semiconductor field effect transistors (MOSFET) (M1-M3) in each pixel was used in the design. Pixel size varies in the imager as shown in Figure 8; however, photosensitive element (photodiode-PD) sizes are kept constant in three regions. Associated photodiode capacitances for different pixel array rows are given in Figure 10. PD capacitance is 75fF for rows 0 to 62, 25fF for rows 63 to 135, and 12.5fF for rows 136 to 151. They are binary weighted for easy application of gain during readout. As a result of scaled PD capacitance, higher sensitivity is achieved by pixels that monitor vehicles far away from the stop light while vehicles close by are monitored with less sensitivity.

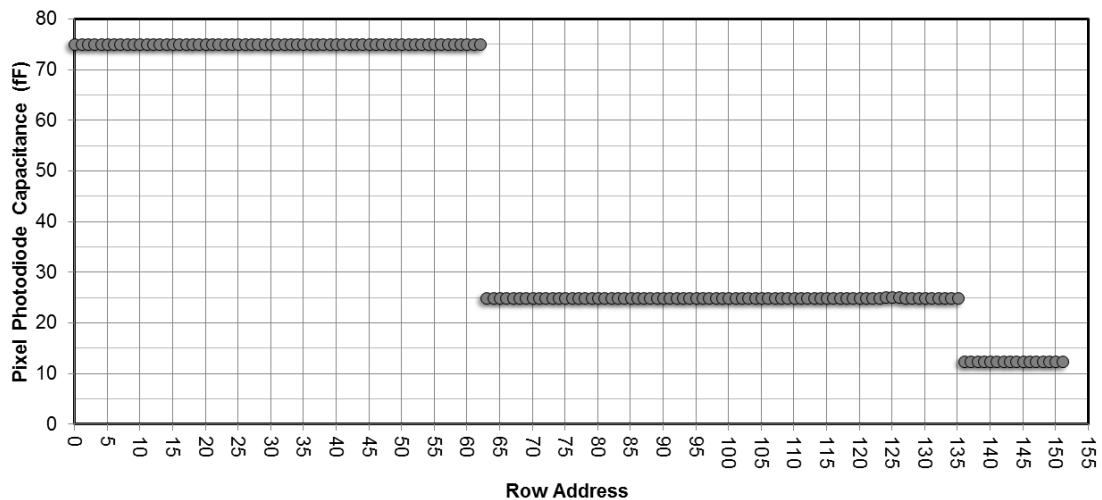


Figure 10: Pixel photodiode capacitances of the TZOID imager versus row address.

Column Readout

Column readout block composes of pixel source follower load transistor (M1), programmable gain (C2/C1) charge amplifier (A1), and auto-zero column comparators (CM1,CM2). Pixel photodiode signals are read through the pixel source follower with a gain of 0.75 V/V. The pixel signal and reset levels are subtracted performing correlated double sampling (CDS) and amplified by the charge amplifier. Gain of the charge amplifier is set with 3-bit binary control. Sixteen possible gain settings are listed in Table 1. Charge amplifier gain could be set between 0.67V/V and 32 V/V achieving pixel photodiode to chip

output overall gain of between 0.5 V/V and 24 V/V. Charge amplifier output is sampled on a sample-and-hold capacitor (C3). Bottom plate of this capacitor is connected to global ramp generator output for column level ADC operation. Auto-zero two-stage comparator was used for the design. Offset voltages of the comparators are stored on two capacitors (C4, C5) and subtracted/cancelled during analog-to-digital conversion operation. Column readout circuit generates a latch signal (Cout) to 8-bit transparent SRAM memories also located on each column. Circuit diagrams of the column readout block are shown in the Appendix between Figure A.9 and Figure A.11.

Table 1: Gain Settings of the Programmable Column Charge Amplifier

Digital Setting	Column Gain	
	ASC	PD-to-output
0	0.67	0.5
1	1.33	1.0
2	2.00	1.5
3	2.67	2.0
4	3.33	2.5
5	4.00	3.0
6	5.33	4.0
7	6.00	4.5
8	6.67	5.0
9	8.00	6.0
10	10.00	7.5
11	12.00	9.0
12	16.00	12.0
13	20.00	15.0
14	24.00	18.0
15	32.00	24.0

Analog-to-Digital Converter (ADC) and Memory Blocks

A single slope ramp signal is applied during digitization of the pixel signals sampled on each column. Triggering occurs when the ramp signal is equal to the sampled pixel signal allowing digital quantization of them. Global programmable (6-bit/8-bit) synchronous counter, binary weighted charge redistribution digital to analog converter (DAC) with output buffer is used in the ADC block. Output of the synchronous counter also drives transparent SRAM cells placed on each column. A global sense amplifier was designed to digitize the column SRAM signals and drive to the digital pads of the imager IC. Circuit diagrams of the ADC and memory blocks are shown in the Appendix between Figure A.12 and Figure A.17.

Reference Generator and DACs

A supply independent, programmable current reference circuit was designed and integrated in the TZOID imager. Reference currents could set to 5 μA , 10 μA or 20 μA . using 2-bit control. Default current setting is 5 μA for the design. Current DACs use reference current by binary scale mirroring and have 6-bit programming control through scan chain. Similarly voltage DACs were also designed that could be controlled with 6-bit resolution. Four current and five voltage DACS were designed and integrated in the design. Circuit diagrams of them can be found in Figure A.18 to Figure A.21.

Other Blocks

Standard logic cells were used for addressing rows and columns of the TZOID imager during readout. Detailed circuit diagram of the row and column decoder and driver circuits are shown in Figure A.22 and Figure A.23.

Integration

The TZOID imager blocks were simulated and their functionalities were verified before physical design and integration. Designed physical layout of the imager blocks were checked against the schematic and integrated into IC as shown in Figure A.24. A 0.18 μm CMOS process was used for the design and fabrication. Core supply voltage of the TZOID imager was set to 1.8V. Physical size of the imager die is 3.23mm x 2.9mm with 68 pads. A 68 pin Ceramic Leadless Chip Carrier (CLCC) package were chosen to package the fabricated imager die. Packaged TZOID imager is shown in Figure A.25.

The packaged TZOID imager was soldered on custom designed camera board that holds FPGA development board, and various voltage regulators and digital buffers as shown in Figure A.26. The FPGA board on the camera board handles controlling communication with the PC, controlling the TZOID imager, and reading frames of images from the imager. Custom software that a user could set TZOID control registers and communicates with FPGA board was also written. User interface of the program is shown in Figure A.27. It shows the TZOID video stream live while performing different image processing algorithms. A sequence of images from the TZOID imager is shown in Figure A.28.

Algorithm Design and Development

Platform Selection

A platform and language is necessary to be determined before the algorithm design and development starts. Thus, the first task was to investigate available, expandable, and low power platforms for the algorithm development and implementation. The identified options were: Raspberry Pi portable single board computer, and FPGA development platform as shown in Figure 11.

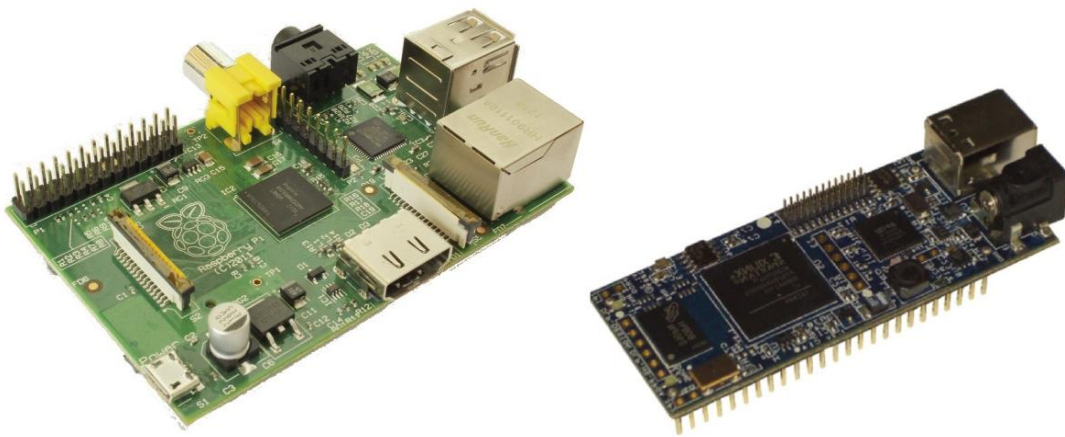


Figure 11: a) Raspberry Pi single board computer, b) FPGA development board.

- Raspberry Pi Computer: The Raspberry Pi is a low cost, credit-card sized (85.6mm x 56mm x 21mm) ARM based single board computer that plugs into a computer monitor or TV, and provides a number of standard IO options as shown in Figure 11a. It is a capable device that enables people of all ages to explore computing hardware at a very low cost (<\$50). It's capable of doing everything one would expect from a desktop computer, from browsing the internet and playing high-definition videos, to making spreadsheets, word-processing, and playing games [2]. While a Raspberry Pi has enough processing power to run almost any algorithm one could throw at it, the major downside is the code used to implement developed algorithms could not be synthesized into an Application Specific Integrated Circuit (ASIC) that could be integrated with the TZOID image sensors.

- FPGA Development Boards: A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing to implement application specific functions and algorithms, [3]. Figure 11b shows a selected FPGA development board containing Xilinx XC3S400A-4FTG256 FPGA. The downside to the FPGA development platform is that it does not have the level of processing capabilities and expansion ports that the Raspberry Pi provides. However, any algorithm or code designed for the FPGA could easily be integrated into an ASIC with image sensor of choice. For these reasons, we decided to use the FPGA development board instead of the Raspberry Pi or some other micro-processor based development systems.

Algorithm Selection

Several image processing and feature extraction algorithms are used in modern camera based traffic control systems. Brief descriptions of them are reviewed here:

Gaussian Evidence Gathering Algorithms

The cleanest forms of evidence gathering in image-based motion detection systems are to use Gaussian based algorithms. While they do an excellent job at gathering evidence for the presence of a moving object in the field of view, they require demanding signal processing capabilities of a microprocessor board that cannot be done in real-time. Because of this, we decided not to use Gaussian-based evidence gathering algorithms in our project.

Background Subtraction Based Algorithms

Background subtraction algorithms reduce the amount of data by subtracting unmoving background from video streams while identifying only moving objects in the field of view. This method requires three frames to be generated to detect the presence of a moving object:

- **Background Frame:** The background frame is a moving average of all the frames the image sensors captures. We developed algorithms to aid in the generation of the background frame.
- **Current Frame:** The current frame is simply the most recent frame to be read from the image sensor.

- **Foreground Frame:** The foreground frame is the difference between the current frame and the background frame, hence facilitating the “Background Subtraction.” When the background frame is subtracted from the current frame, the only objects left behind are objects that are moving in the current frame. Thus, the foreground frame plays an important role in detecting whether or not a vehicle or an object is moving in the field of view. An illustration is shown in Figure 12 [4].

Other algorithms such as edge detection and blobbing and centroid detection are not considered in this application due to extensive need for processing power while implementing of each of these algorithms.



Figure 12: a) The current frame, b) the foreground frame after processing [4].

Algorithm Design

As mentioned before, there are three major problems with the current image-based motion detection systems of today. First, there are privacy issues. This algorithm requires acquiring and transmitting images of people and vehicles driving on the road. Second, there are data rate and communication issues. Here whole frames of image data must be captured, processed, and sent to the traffic controller, and current traffic controllers are unable to handle this high data rate. Third, high speed and powerful processors are required to perform the calculation intensive image processing operations. Thus, this project’s goal was to invent algorithms that will eliminate these three issues and make image-based traffic detection a more viable option in the future.

Delta Cap Thresholding Method (DCTM)

One of the problems encountered when generating the background, current, and foreground frames is having objects incorporate too quickly into either the background or foreground. When this happens, it is impossible to tell if an object is moving. To get around this, current systems use complicated algorithms that gather evidence for an object being part of either the background or foreground frames [5]. The developed algorithm does this by a method it is called “*thresholding*.” In general, when the current frame is subtracted from the background frame, it would let certain pixels change from the current frame to the next. These pixels are called “*deltas*.” The new algorithm allows small deltas to be incorporated into the background immediately. Large deltas however are only allowed to change the background by a certain value, it is called “*delta cap*” (Figure 13). The proposed background generation method (Delta Cap Thresholding Method – DCTM) is simple and elegant and avoids all the complicated algorithms that other systems use. This also allows implementation of the design on an FPGA and solving one of the major problems today with image-based motion detection. The DCTM is used for obtaining foreground pixels, as well as determining whether or not a tripwire has been tripped as discussed below.

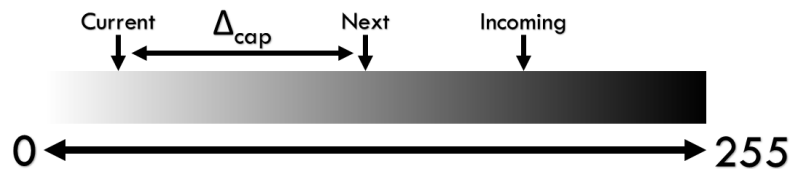


Figure 13: Delta cap thresholding method (DCTM) for background generation.

Tripwire Method

As mentioned before, two other issues with image-based motion detection today are high data rates and privacy issues. The “*tripwire*” method used in the proposed algorithm is the answer to both of these issues. In short, the method calls for only monitoring certain rows of pixels on the captured scene image for “*deltas*.” When a certain number of deltas are obtained in a specific image row, it is assumed that an object is at that location on the road.

This method allows keeping data rates extremely low by monitoring only 32 rows of pixels on the scene image, rather than all rows. Additionally, there is not enough resolution and hence information in one row of pixels to determine any private information about a vehicle or a person eliminating privacy issues. It also greatly facilitates proposed algorithms for speed and position detection. If the distances between tripwires are pre-determined and known, then the speed of the object could be found by simply dividing that distance by the time between tripwires' "tripping." Figure 14 below graphically describes the proposed method.

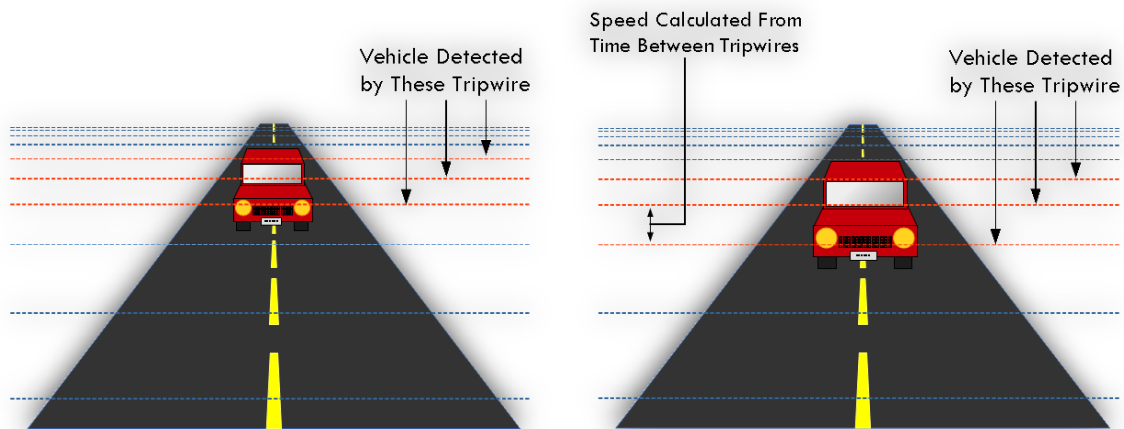


Figure 14: Tripwire method for data reduction and speed detection.

Prototype Development

OpenCV Development

An open source video library called OpenCV was used for testing developed algorithms on software before implementing them on FPGA development hardware. OpenCV allowed tweaking the algorithms to figure out critical design parameters while displaying live video from an image sensor. Figure 15 below shows all four frames that are generated for our OpenCV prototype software running DCTM and tripwire algorithms. Notice that as the hand is moved, it shows up in the current frame, but does not show up in the background frame while only the arm and hand (moving objects) shows up on the foreground frame. Also

notice that only those tripwires that have enough evidence for a moving object are tripped in the tripwire frame.

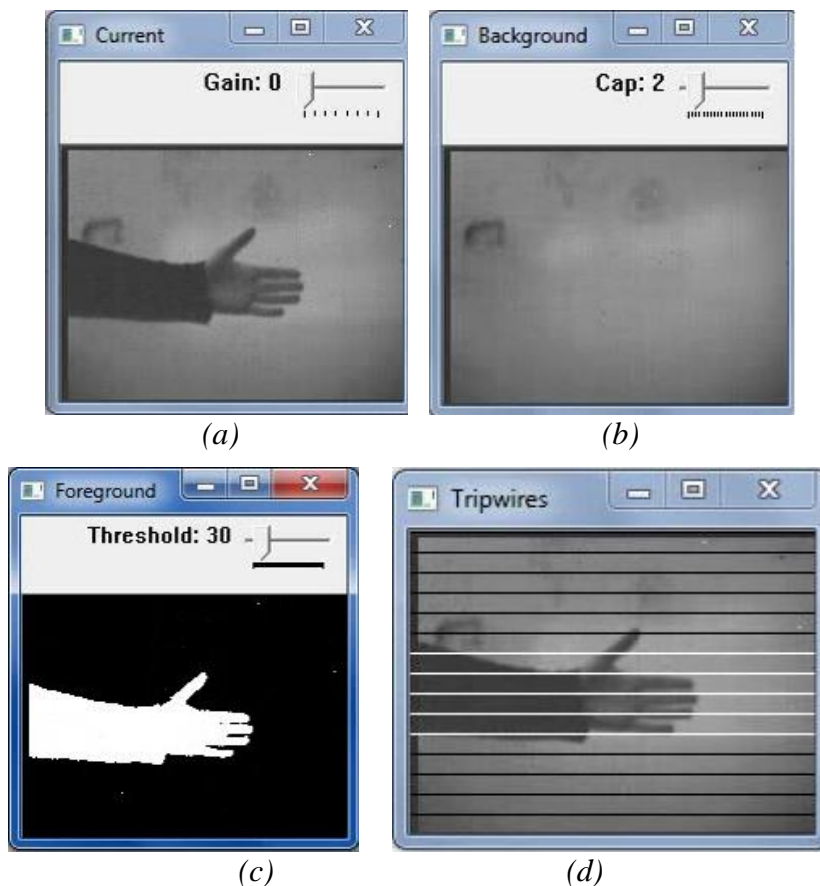


Figure 15: OpenCV testing of the DCTM and Tripwire algorithms on a live image: a) current frame, b) the background image extracted by DCTM, c) the foreground image showing moving objects, and d) tripwires frame [6][7].

Before all of our modules were implemented in hardware, a hardware simulation program was developed that fed an image data file to the image processor running developed algorithms and steams out the background, foreground, current, and tripwire frames.

Hardware Design

Once the algorithms were designed and simulated in OpenCV environment, coding for hardware implementation was done. A hardware description language (HDL) called Verilog was used in this project. Modular design approach was chosen to allow expendability and

easy verification and integration. Each block was tested individually and verified. A complete system block diagram for hardware implementation of the algorithms is shown in Figure 16.

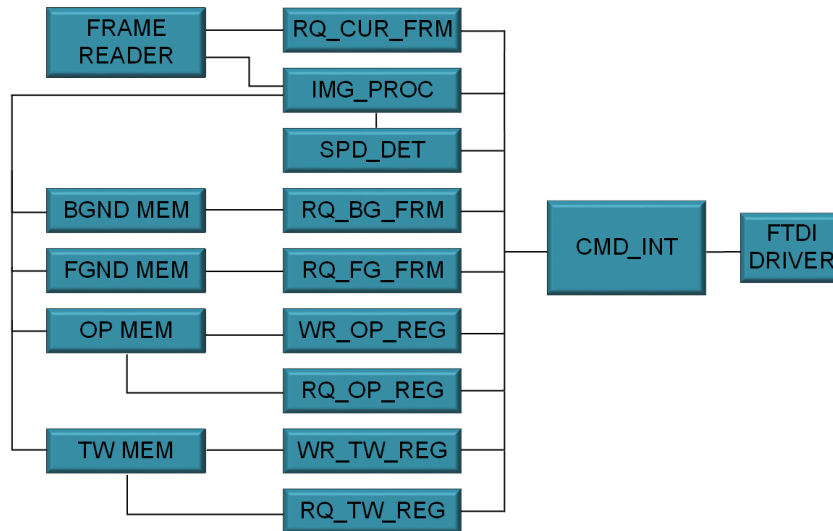


Figure 16: Complete system block diagram for hardware implementation.

The FTDI driver of the FPGA-II board takes care of USB2 communication protocol. The command interpreter handles all data and control signals between the FTDI driver and the rest of the system. The request current frame module pulls current image data from the image sensor and passes it to the command interpreter. The image processor implements all developed image processing algorithms and stores the background and current frames in memory. The speed detection module gleans speed and position information from tripwire data. The request background and foreground frame modules pull the background and foreground frames from memory where they were stored by the image processor. The write and request operational register modules store or read camera and analog to digital converter (ADC) setting, and other control values for image sensor operation. The request and write tripwire register modules take in the rows that are programmed as tripwires as well as the distance between them and store them or read them from memory. Finally, there are several memories that we use for storing background and foreground frames, as well as tripwire and image sensor operation settings while the frame reader takes care of all handshaking with the

image sensor, allowing users to easily pull data off of the FPGA platform. Low level module details of the hardware implemented on FPGA platform is given in the Appendix of this report.

Verification Process

The designed modules were debugged and simulated by using standard iSim simulator that comes with the Xilinx Integrated Software Environment (ISE) that allows synthesis, analysis, and simulated HDL designs, enabling the developer to compile their codes, perform timing analysis, examine RTL diagrams, and configure the target device with the programmer bit stream.

Testbed Development

For future speed detection verification, a testbed prototype was developed. The system outlined in Figure 17 is intended to provide an accurate tripwire facsimile through the use of photo gates similar to those used for safety in automatic garage doors. The testbed is scaled down to a 1/64 representation of 200 meters of road making the laboratory test environment very controllable and practical.

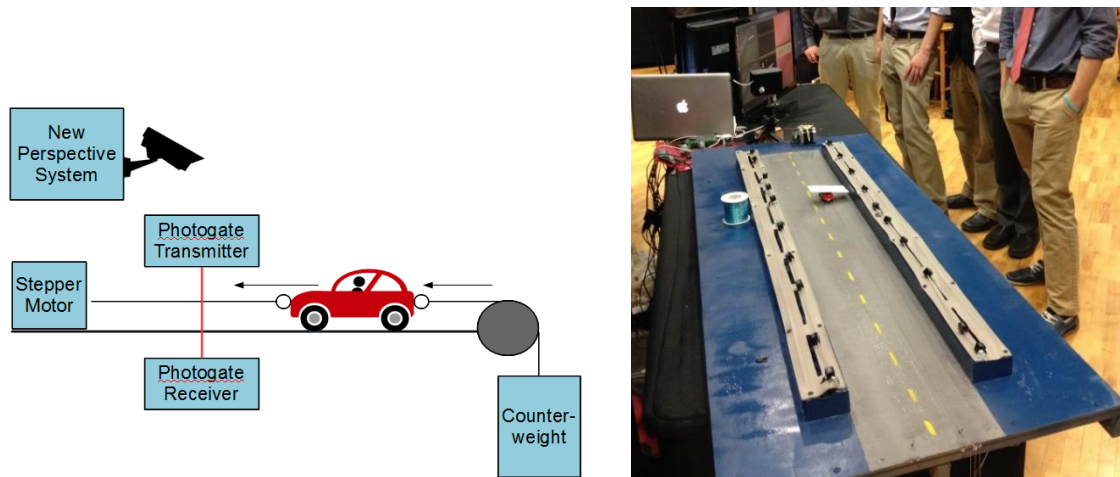


Figure 17: Testbed block diagram and final picture of the testbed.

As shown in Figure 17, a stepper motor controlled by a microcontroller pulls a car along the road guided by two opposing lines. The lines are held taut by a counterweight. Photo gates are mounted on either side of the road at the same positions of the virtual tripwires projected

by the camera. These photo gates report the exact speed of the car which then can be compared to the values reported by the camera. Though the testbed wasn't completed in time for full system verification, coarse testing and speed verification was possible regardless. The existing testbed would still be relevant for future research to extract high fidelity data about the performance of the system.

Graphical User Interface Development

For demonstration and testing a graphical user interface (GUI) was developed using the Qt software framework as shown on Figure 18. The GUI was able to simultaneously display all three frames of data being created by the image processor as well as the speed and position data the system was reporting. Communication to the GUI was achieved through a simple packet based system.

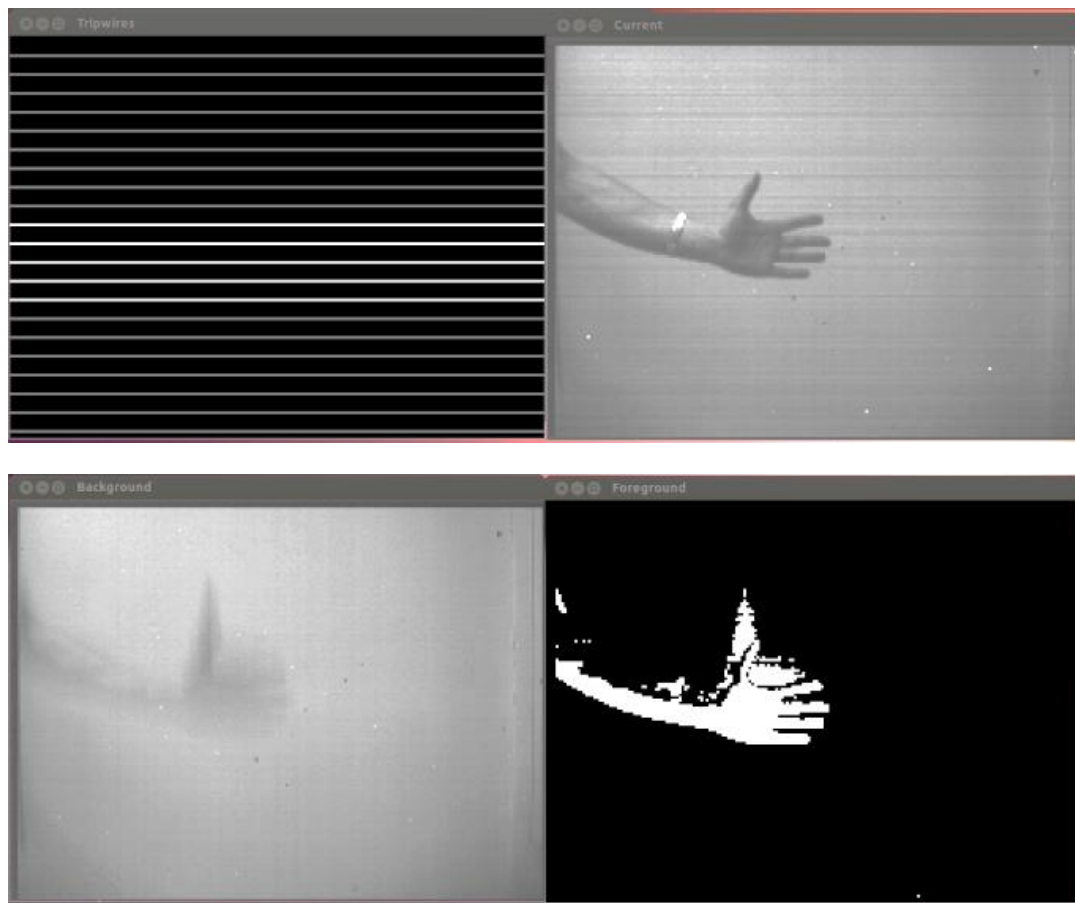


Figure 18: Developed GUI screenshots.

Advantages of Proposed Approach

Other approaches exist for traffic monitoring and vehicle detection [6-7]. They are currently widely used in the video based traffic monitoring industry. They use a standard, rectangular, high-resolution image sensor IC for gathering scene image and tossing out pixels or combining pixels to accomplish certain image processing functions. The stringent requirements of the processing and image readouts are the main reasons why a custom image sensor with trapezoid pixel array with varied pixel sizes were designed in this project. Explaining these requirements and why it is not a feasible solution to use existing image sensor IC would make our design approach more appreciated.

First of all, it is true that by combining smaller pixels it is possible to form trapezoid like pixel array characteristics. To form a $30\mu\text{m} \times 30\mu\text{m}$ pixel size, 10×10 of small pixels with $3\mu\text{m} \times 3\mu\text{m}$ pixel size has to be combined to get the average pixel value. However, no mainstream image sensor IC could do this merging/combining on-chip. Modern image sensors have “binning” capability however they can combine 2×2 or 4×4 but no more than these number of pixels on-chip. Thus, full image sensor array has to be readout before combining pixel signals that has to be done on software. This means that high-bandwidth communication has to be used during frame readout. Second, an auxiliary memory array has to be used to map pixel sizes and which row and columns has to be combined to form which pixel on the trapezoid pixel array. If this mapping is done, the amount of memory required to store this information would be larger than the frame memory that needed to be used to process the captured video images. Third, combining pixels of a rectangular image sensor with fixed pixel size would also need high processing capacity DSP or CPU based computing systems with frame memories. What we are proposing is to simplify the camera system achieving not only doing image processing faster and using low-power but also doing it without requiring large frame memories, high-speed communication channels and powerful digital processors.

The proposed trapezoid imager achieves 55 times more improvement on several fronts including reduction in power consumption, in communication bandwidth, and in processing element speed. How?

1. We build an image sensor that has trapezoid array of pixel totaling 152 (Height) x 40 (Width) = 6080 total pixels. Horizontally the minimum pixel size is $3.4\mu\text{m}$ while vertically it is $3.87\mu\text{m}$. Thus, if a rectangular array image sensor is chosen, pixel size has to be $3.4\mu\text{m} \times 3.4\mu\text{m}$ to accommodate minimum pixel size in the trapezoid image sensor.
2. The total pixel array height of 152 pixels with varied height is $1988\mu\text{m}$ while the maximum width of the trapezoid array is $1948\mu\text{m}$ (or 40 columns of pixels multiplied by the largest pixel size in a horizontal direction which is $48.7\mu\text{m}$). If the rectangular pixel array is used, the number of pixels in a horizontal direction has to be $1988\mu\text{m}/3.4\mu\text{m} = 573(\text{W})$ and vertical direction $1988\mu\text{m}/3.4\mu\text{m} = 585(\text{H})$. So, a rectangular image sensor with $585(\text{H}) \times 573(\text{W}) = 335,205$ pixel array size is required to function as the designed trapezoid sensor with $152 \times 40 = 6080$ pixels which is 55 times larger compared to the trapezoid imager.
3. To form a 152×40 trapezoid pixel array using 585×573 rectangular pixel at least 152×4 bytes (largest pixel size in the vertical direction, $43.9\mu\text{m}/3.4\mu\text{m} = 13 \sim 4\text{-bit}$) = 608 bytes of a array mapping memory is required just to map 585×573 rectangular pixel array to 152×40 trapezoid pixels considering each row of trapezoid pixels have the same pixel height and width.
4. By designing the trapezoid array, we achieved $335,205/6080 = 55$ times data reduction and speed up in terms of frame capture. This also translates to lower camera system power consumption. Our sensor running at 120 frames per second (FPS) consumes less than 10mW of power. Thus it achieves $10\text{mW}/(120 \times 6080) = 13.7 \text{ nJ/pixel}$ (nano Joules per pixel) energy consumption per pixel read. If the same energy is used in rectangular pixels (585×573) to maintain 120 FPS, total power required by the rectangular imager is 551mW ($335,205 \times 13.7 \times 120 \times 10^{-9}$) or half a watt of power which is again 55 times higher than the custom made trapezoid design.

5. Rectangular imager with 585x573 array size running 120 FPS generates 40.225Mbytes/sec video stream that has to be processed by a very fast digital processor (DSP, or CPU, or uP/uC system). If this has to be sent through a serial cable to the central system, it requires a reliable connection with 40.225Mbyte/sec x 8bits= 321.8Mbit/sec speed. In trapezoid, required bandwidth is $152 \times 40 \times 120 = 0.7296 \text{Mbyte/sec} \times 8 \text{bit} = 5.837 \text{Mbit/sec}$ which is also 55 times lower communication bandwidth. Also a simple digital processing unit or uP or uC based system could be used.

FINDINGS; CONCLUSIONS; RECOMMENDATIONS

Conclusions

A new concept of scaled pixel array type image sensor, called trapezoid or TZOID, was designed, fabricated, and tested in an expandable camera platform. This unique image sensor IC was fabricated in a 0.18 μ m CMOS process and mounted on a 68-pin open-lid CLCC package. A low resolution pixel array of 40x152 was used in this version of the imager monitoring up to 800ft of incoming traffic. Initial tests showed that the image sensor works properly, and generating reasonable scene videos up to 100 frames per second. Detailed testing of the TZOID imager is still on-going.

Two image processing algorithms were developed (Delta Cap Thresholding Method – DCTM and Tripwiring Method) for detecting incoming vehicles to an intersection while extracting their speed and location information. The algorithms were implemented in the hardware definition language (HDL) of Verilog and synthesized and tested on a field programmable gate array (FPGA) development platform. Intent of the use of Verilog HDL language is to allow integration of the image processing engine running the developed algorithms and the TZOID imager on a single integrated circuit (IC) chip in the future.

It is shown that it is possible to extract speed and location features without requiring high processing and communication bandwidth on camera systems. Complexity of image processing could move hardware down to image sensor pixels while achieving compact, low power, and high speed operation without requiring high bandwidth communications. It is also shown that the trapezoid image sensor provides 55 times better performance in terms of communication bandwidth, power consumption, memory and post processing requirements than that of mainstream video based approaches that uses rectangular, fixed-size pixel array image sensor ICs.

Observations

The goals of the project were to come up with simple feature (speed, location) extraction algorithms that are suitable for application specific integrated circuit (ASIC) design and to develop a unique image sensor for traffic monitoring. Both goals were achieved; however,

they are not tested together on the same camera platform due to the time lag between the TZOID IC design and fabrication (fabricated chips arrived in May 2014) and algorithm development (carried out between September 2013 and May 2014). Algorithms were tested using a rectangular image sensor IC with 200x150 pixel array that was designed by Dr. Ay and his research team in the past (watch [8]-[11]). However both are working properly, issues such as sensitivity and noise for both algorithms and TZOID imager were observed. Some of the issues are believed to be emanating from “affordable” optics used in the camera boards during testing (watch [12]). Precise tuning and improvement of both image sensor pixel and readout electronics as well as the algorithm parameters are needed if they are integrated together on an single ASIC.

Future Directions

Thus, three major future works are identified: 1) test the algorithms with video stream coming from the TZOID imager mounted on a real traffic pole under different lighting conditions (day, night, etc.) , 2) improve the TZOID image quality by increasing the number of pixels while addressing sensitivity and noise issues on pixel and readout electronics, 3) synthesize the feature extraction engine code into logic and integrated with the next generation TZOID imager to design single chip TZOID imager ASIC for traffic monitoring.

REFERENCES

- [1] Federal Highway Administration, "The National Intersection Safety Problem," 2007.
URL: http://safety.fhwa.dot.gov/intersection/resources/fhwasa10005/brief_2.cfm.
- [2] T. Vanderbilt, Traffic, Knopf, 2008.
- [3] Raspberry Pi Foundation, "What is a Raspberry Pi?,"
URL: <http://www.raspberrypi.org/help/what-is-a-raspberry-pi/>. [April 2014].
- [4] Wikipedia, "Field-programmable gate array," April 2014.
URL: http://en.wikipedia.org/wiki/Field-programmable_gate_array. [April 2014].
- [5] J. K. V. S. D. C. M.J. McHugh, "Foreground-Adaptive Background Subtraction,"
Boston University, Electrical & Computer Engineering, 2007-2008.
URL: <http://vip.bu.edu/projects/vsns/background-subtraction/fa/>. [30 April 2014].
- [6] L. A. Klein, M. K. Mills, D. R.P. Gibson, "Traffic Detector Handbook: Third Edition – Volume I", Pub. No. FHWA-HRT-06-108, U.S. DOT-FHA, October 2006
URL: <http://www.fhwa.dot.gov/publications/research/operations/its/06108/06108.pdf>
- [7] L. A. Klein, M. K. Mills, D. R.P. Gibson, "Traffic Detector Handbook: Third Edition – Volume II", Pub. No. FHWA-HRT-06-139, U.S. DOT-FHA, October 2006
URL: <http://www.fhwa.dot.gov/publications/research/operations/its/06139/06139.pdf>

PROJECT DEMO LINKS

- [8] <http://www.ece.uidaho.edu/ee/analog/suatay/data/demos/demo1/demo1.html>
- [9] <http://www.ece.uidaho.edu/ee/analog/suatay/data/demos/demo2/demo2.html>
- [10] <http://www.ece.uidaho.edu/ee/analog/suatay/data/demos/demo3/demo3.html>
- [11] http://www.ece.uidaho.edu/ee/analog/suatay/data/demos/live_demo1/live_demo1.html
- [12] http://www.ece.uidaho.edu/ee/analog/suatay/data/demos/tzoid_demo/tzoid_demo.html

APPENDIX

Low-Level Module Block Diagrams for Hardware Implementation

After sub-partitioning the system, modules were designed and tested individually.

Command Interpreter

The command interpreter (Figure A.1) is the interface between the system and the outside world. It takes care of all data and control signals to or from our system, as well as controlling which modules are “on” and “off” at any given time.

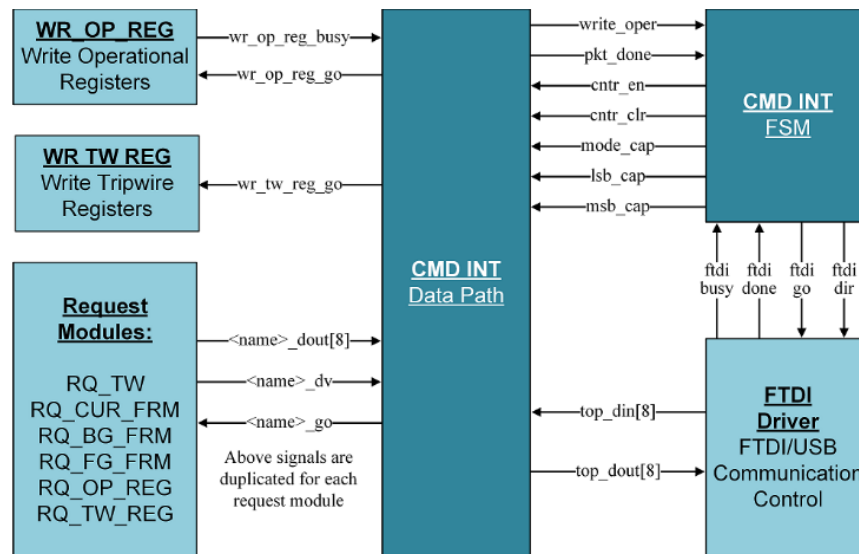


Figure A.1: Command interpreter module.

Image Processor

The image processing block was the piece responsible for implementing both the tripwire and delta cap algorithms (Figure A.2). Due to the nature of these algorithms, each pixel could be processed without regard to any surrounding pixels. This approach allowed flexibility with respect to the time and order of pixels as they are processed. The image sensor used in hardware development presented pixels serially in order from the upper left corner of the captured image to the lower right corner. With each pixel being processed and stored in the time it took for another pixel to arrive. The background and foreground frame data are

presented to the system in real time, with each frame representing the latest data, in contrast to the tripwire detector, which instead represents the status of the previous frames. This introduces a pseudo pipelining reducing the length of the critical path by more than a factor of 2 while allowing the system to use high clock rates. The module fills two full frame memories (the background and foreground), and alerts the system when those frames are ready and valid. Both the foreground and background modules consist entirely of combinational logic, which allowed the hardware to do calculations very quickly. The state machine that controlled the system was also very simple, consisting of only three states.

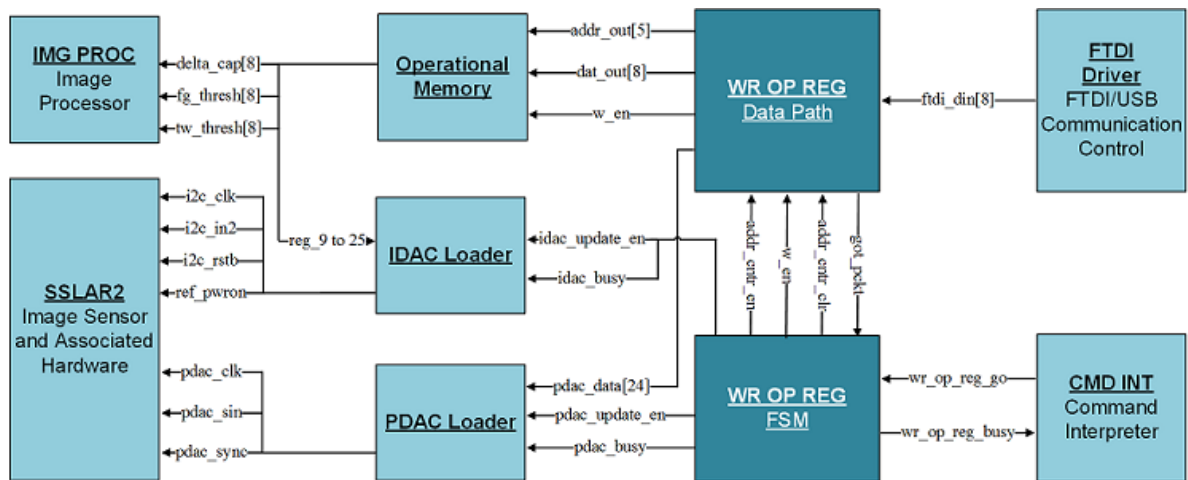


Figure A.2: Image processor block diagram.

Speed Detection Module

The speed detection module (Figure A.3) implements a simple leading edge detection algorithm that monitors the tripwire status and determines the most likely position of the front of the vehicle or moving object. The algorithm waits for the road to become clear at the top of the frame before starting to monitor the tripwires. It then looks for trips at the farthest distance tripwires, moving down the street towards the lights as it detects trips. These trips are reported in a packet that includes the last tripwire tripped and the time difference between that tripwire and the previous one.

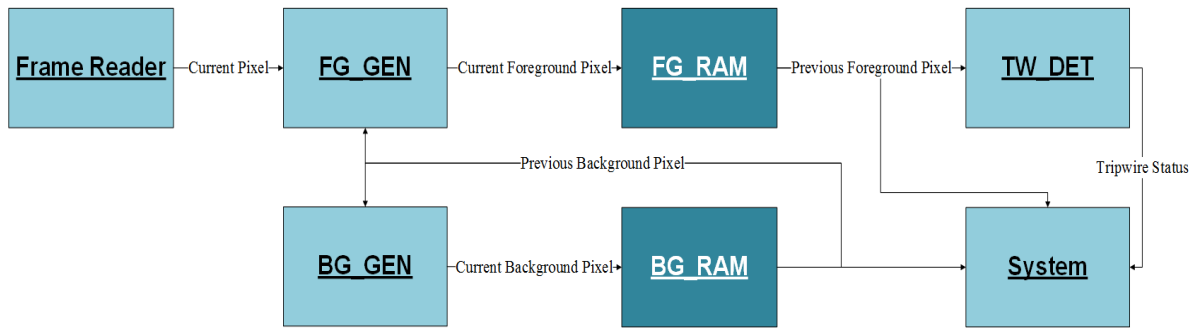


Figure A.3: Speed detection module.

Write Operational Register Module

The operational register module holds and distributes settings that dictate how the system works (Figure A.4). These settings are largely parameters involved with the digital to analog converters providing inputs to the image sensor. These parameters were output through a modified series-parallel interface (SPI) port to their respective devices. Other settings involved the different thresholds governing our image processing characteristics, namely the Delta Cap, Tripwire threshold, and foreground threshold.

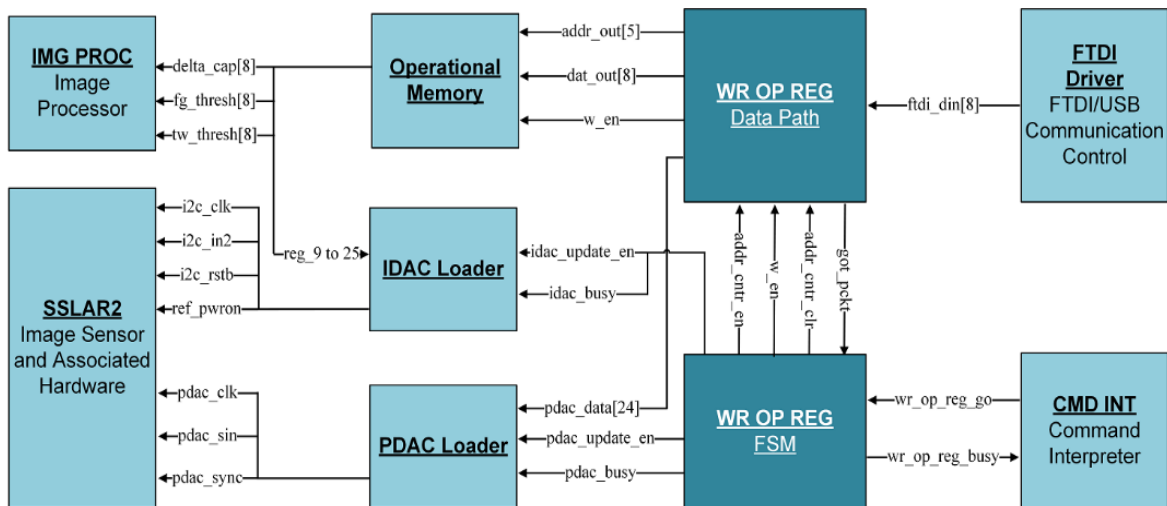


Figure A.4: Write operational register module.

Write Tripwire Registers Module

The least calculation intensive method for determining speed from the system requires the rows that are used as tripwires in the image sensor to be equidistant from each other as projected onto the road. For example, tripwires may be set exactly 10 meters apart from each other on the road – meaning that a car would “trip” a tripwire every ten meters. For a square image sensor such as the image sensor used (SSLAR2) looking at a trapezoidal road, this might translate into rows 0, 25, 40, 52, 62, 70, etc. being chosen as tripwires. Since the distance between tripwires is not equidistant from the perspective of the image sensor, users must have some way of telling the image processor and speed detection modules which rows to monitor. To do this, simple geometric calculations given in previous sections were used prior to installing the system that tells us which rows to choose as tripwires, and then store these positions in memory on the FPGA. These were not hard-coded, because they would be changed based on the way the system is installed. Thus, the write tripwire registers module (Figure A.5) simply takes the calculated values and stores them in memory to be used later.

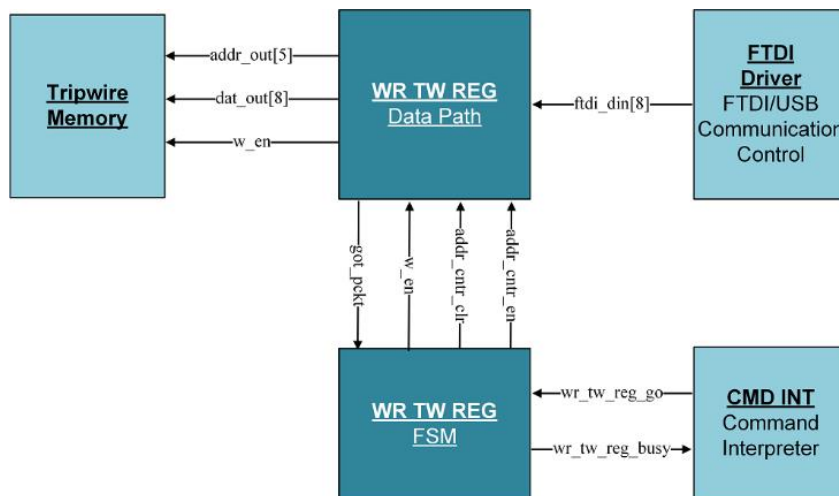


Figure. A.5: Write tripwire registers module.

Frame Reader and FTDI Driver

Both the frame reader and FTDI driver, composes of data path and finite state machine (FSM) diagrams for these two blocks, are designed for reliable sending of the collected

information through USB2 port of the FPGA-II development port to a PC. The implemented USB communication finite state machine is shown in Figure A.6.

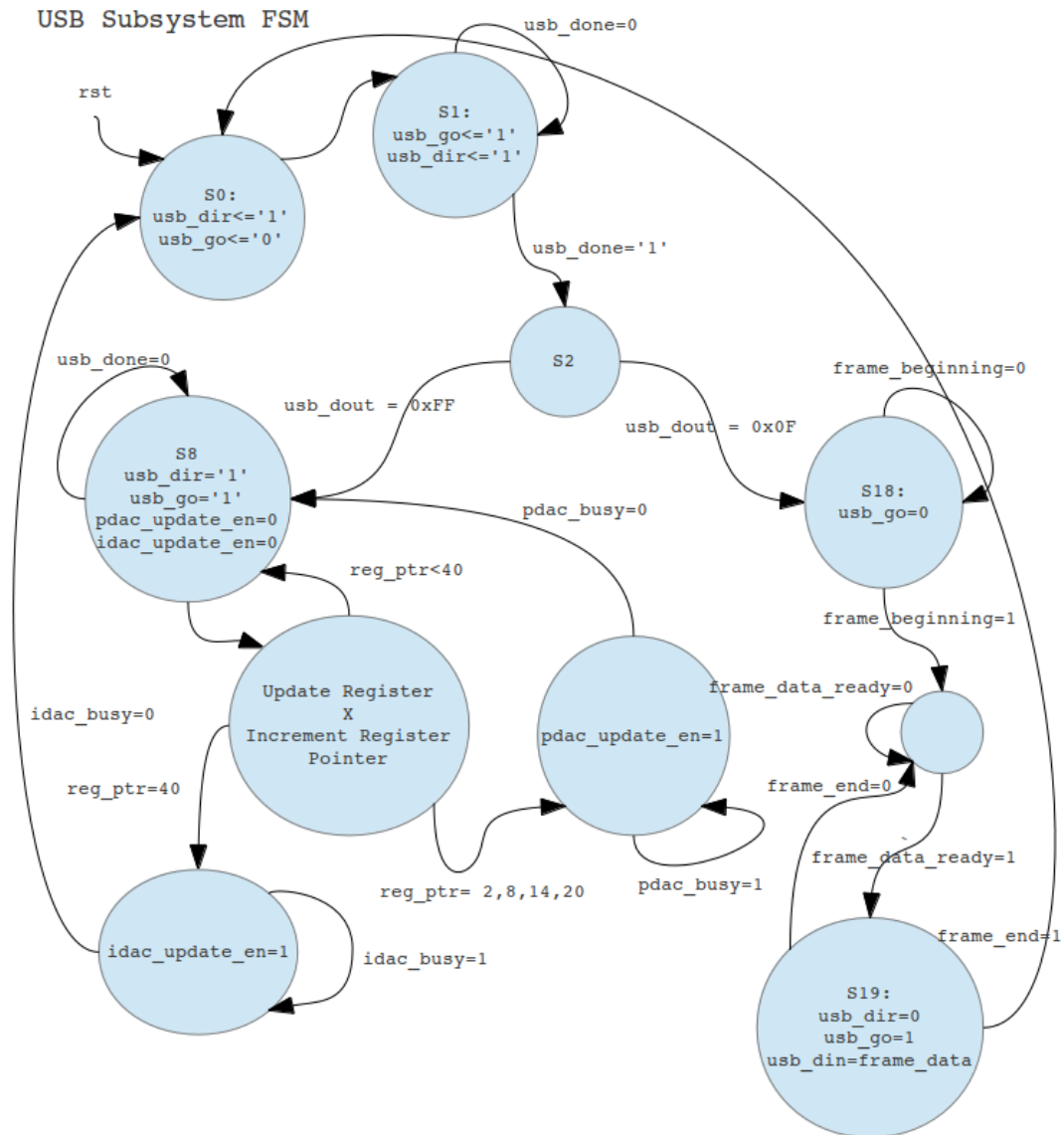


Figure. A.6: USB Communication finite state machine.

Request Current Frame Module

The request current frame module is shown in Figure A.7. Its job is to get data from the frame reader (which pulls data off the SSLAR2 image sensor) and pipe it out to the command interpreter.

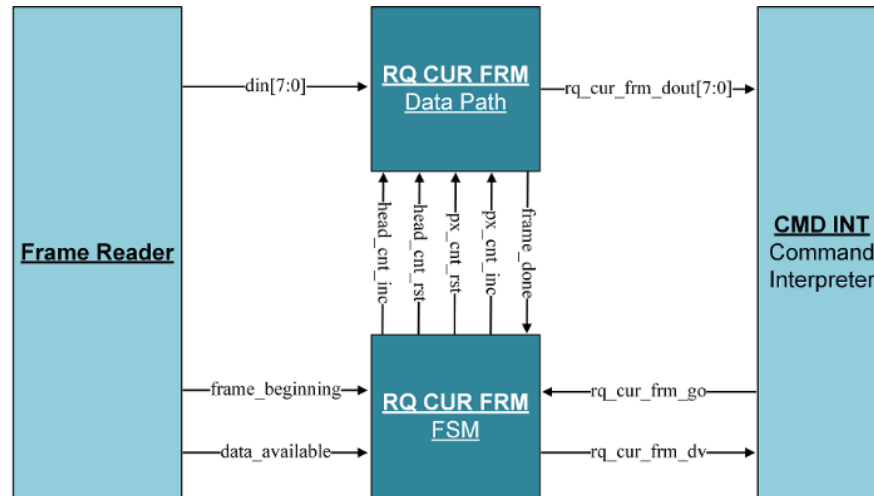


Figure A.7: Request current frame module.

Generalized Request Module

Four of the modules in the system are very similar – they simply access data stored in memory and send it to the command interpreter. These modules are the request tripwire registers module, the request background frame module, the request foreground frame module, and the request operational registers module. Figure A.8 details the basic data path for each of these modules. Dissimilarities among them are discussed below.

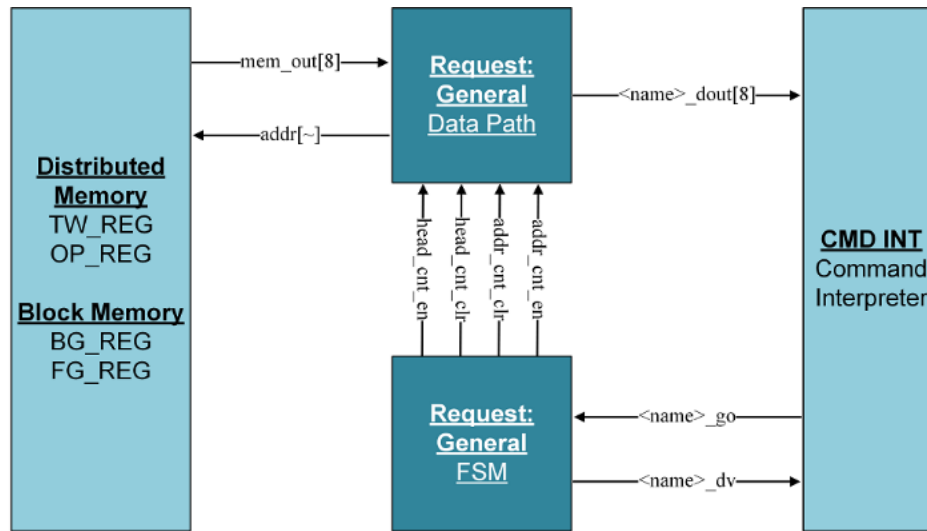


Figure A.8: Generalized request module.

The dissimilarities among these four modules occur mainly in the type of data they request from memory, and the type of memory that data is stored in. As shown in Figure A.8, the background and foreground frames are stored in block memory. This memory has two busses (write and data), so can be accessed by two different modules to initiate a write sequence. It has only one address and data bus for a read sequence, so it can only be accessed by one module for a read sequence. The tripwire and operational registers use distributed memory, which can be read to and written from by multiple modules. The biggest difference between these modules however is the data they request. The background module requests 30,000 bytes of data from memory. The foreground module requests 30,000 bits of data from memory. The tripwire module requests up to 257 bytes ($32 \times \text{number of tripwires} + 1$) but this number obviously varies depending on the number of tripwires. Finally the operational module requests 28 bytes. Table A.1 below summarizes these data requirements.

Table A.1: Request Module Summary

Module	Data Accessed	Memory Accessed
Request Tripwire Registers	Up to 257 Bytes	Distributed
Request Operational Registers	28 Bytes	Distributed
Request Background Registers	30,000 Bytes	Block
Request Foreground Registers	30,000 Bits	Block

Circuit Diagrams of the TZOID Imager IC

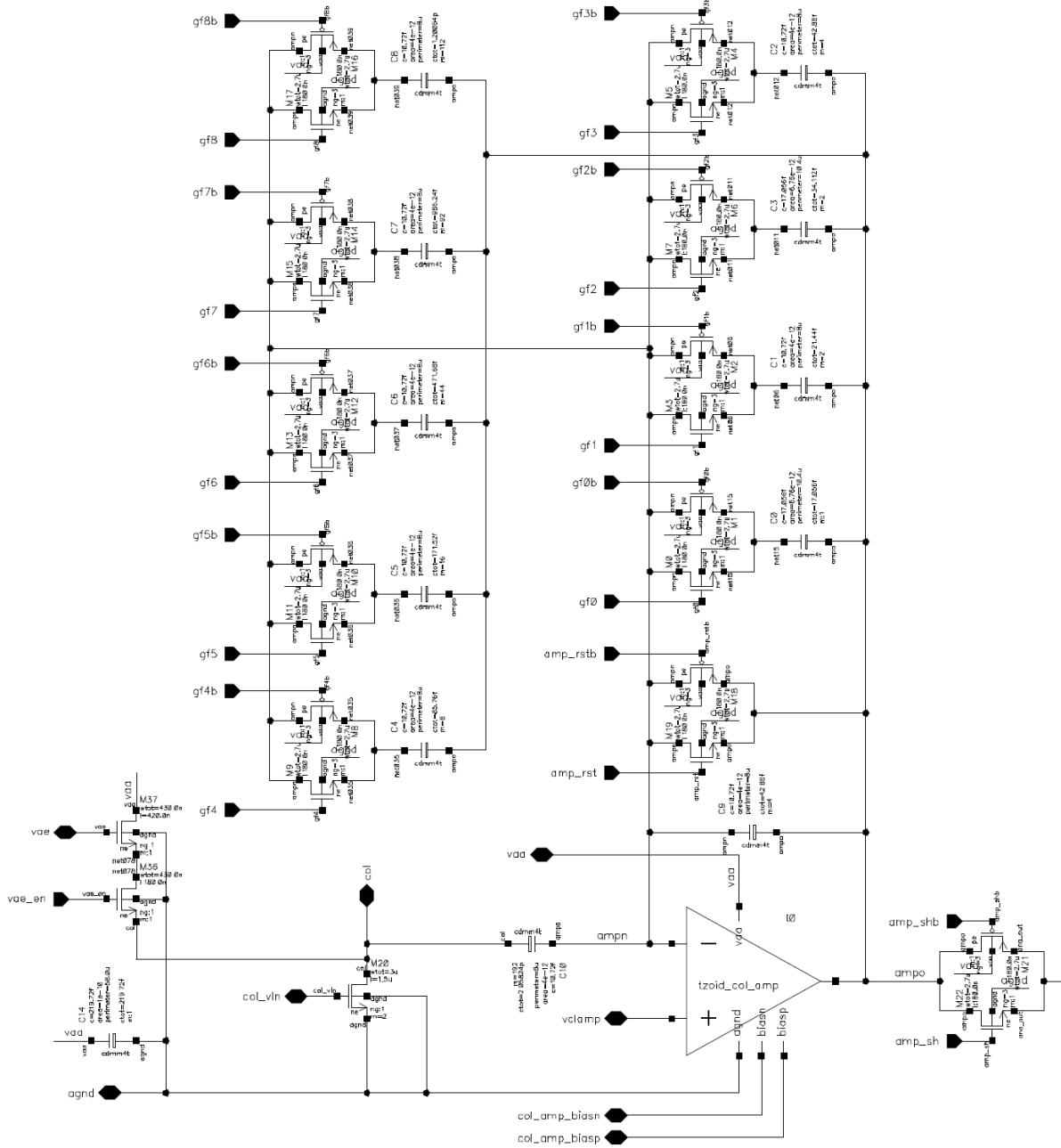


Figure A.9: Circuit diagram of the column charge amplifier with programmable gains.

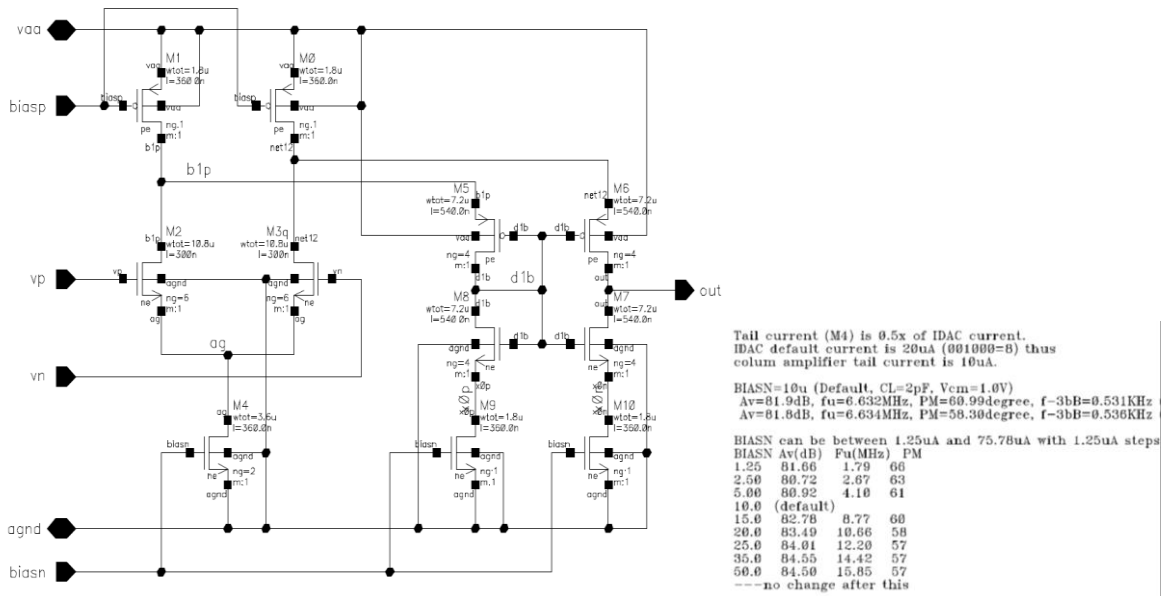


Figure A.10: Circuit diagram of the column amplifier (A1).

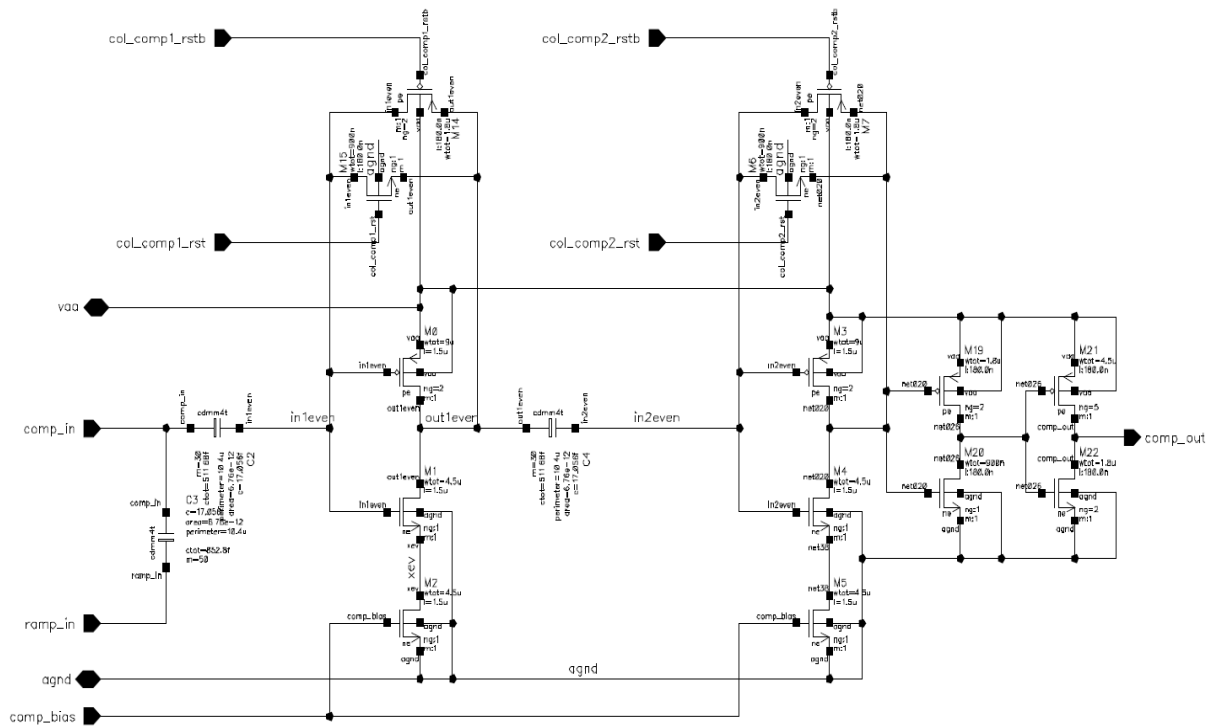


Figure A.11: Circuit diagram of column comparators.

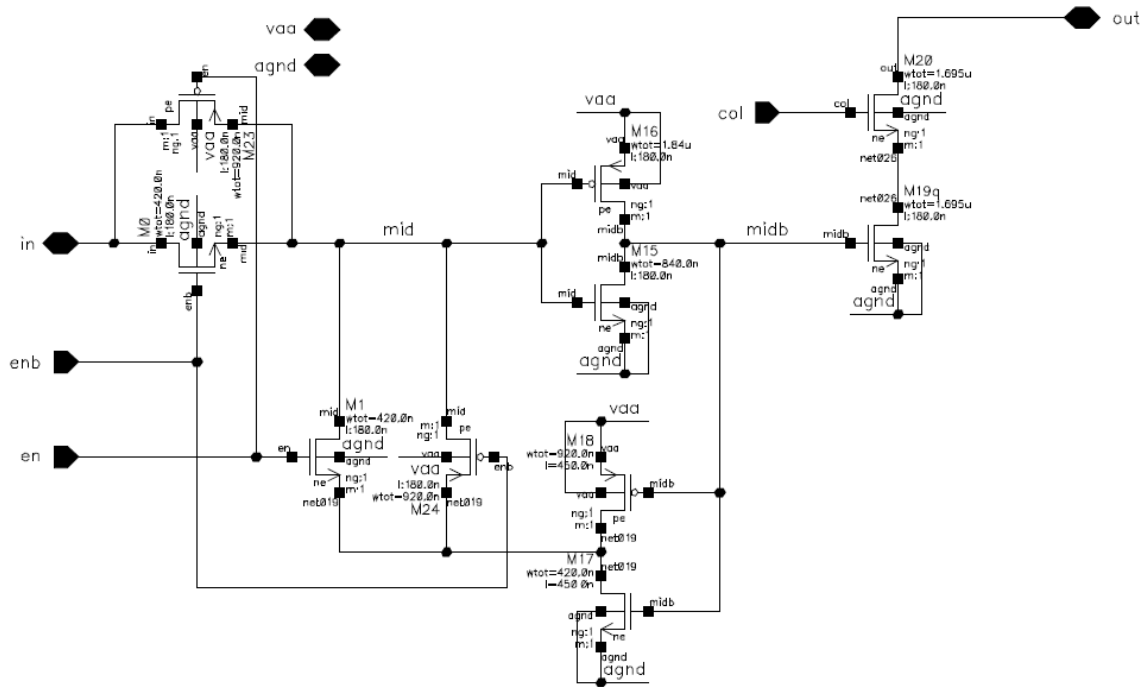


Figure A.12: Circuit diagram of single bit column SRAM cell.

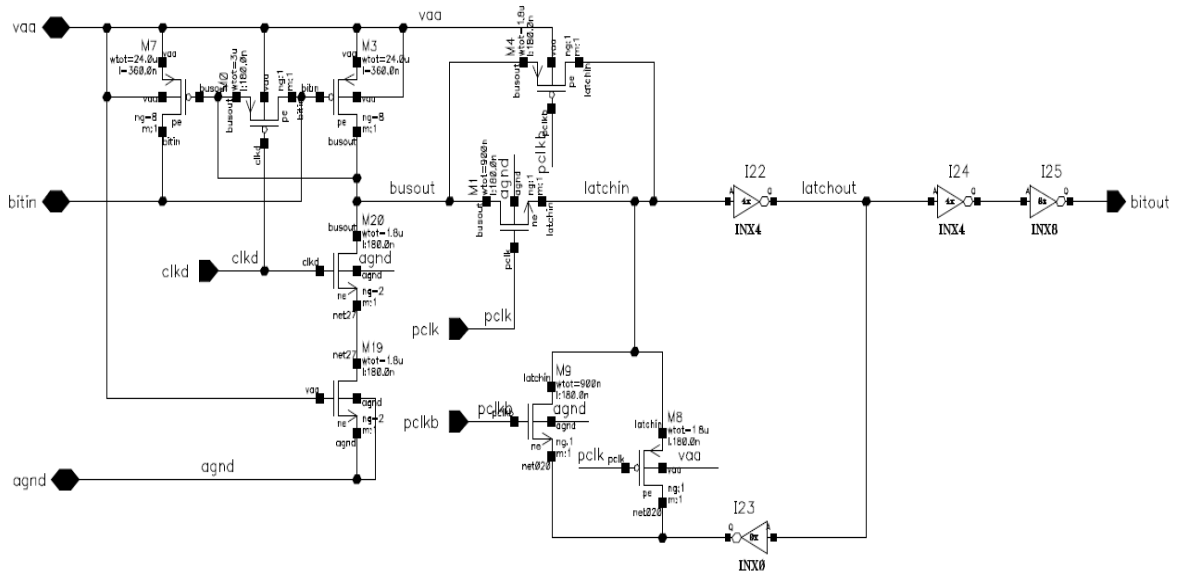


Figure A.13: Circuit diagram of global SRAM sense amplifier.

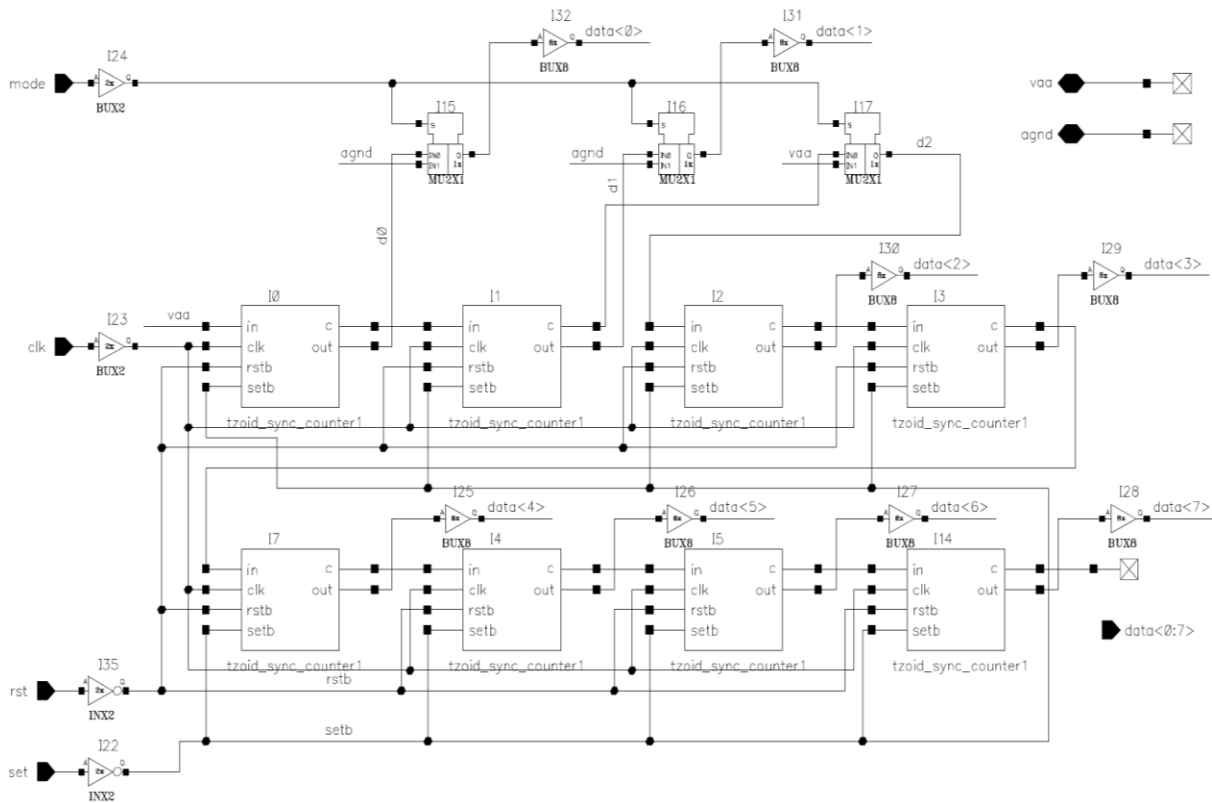


Figure A.14: Circuit diagram of the programmable (6-bit/8-bit) synchronous counter.

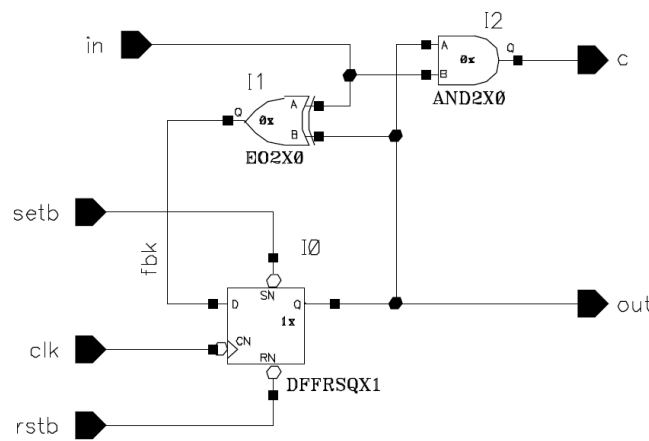


Figure A.15: Circuit diagram of the synchronous counter slice.

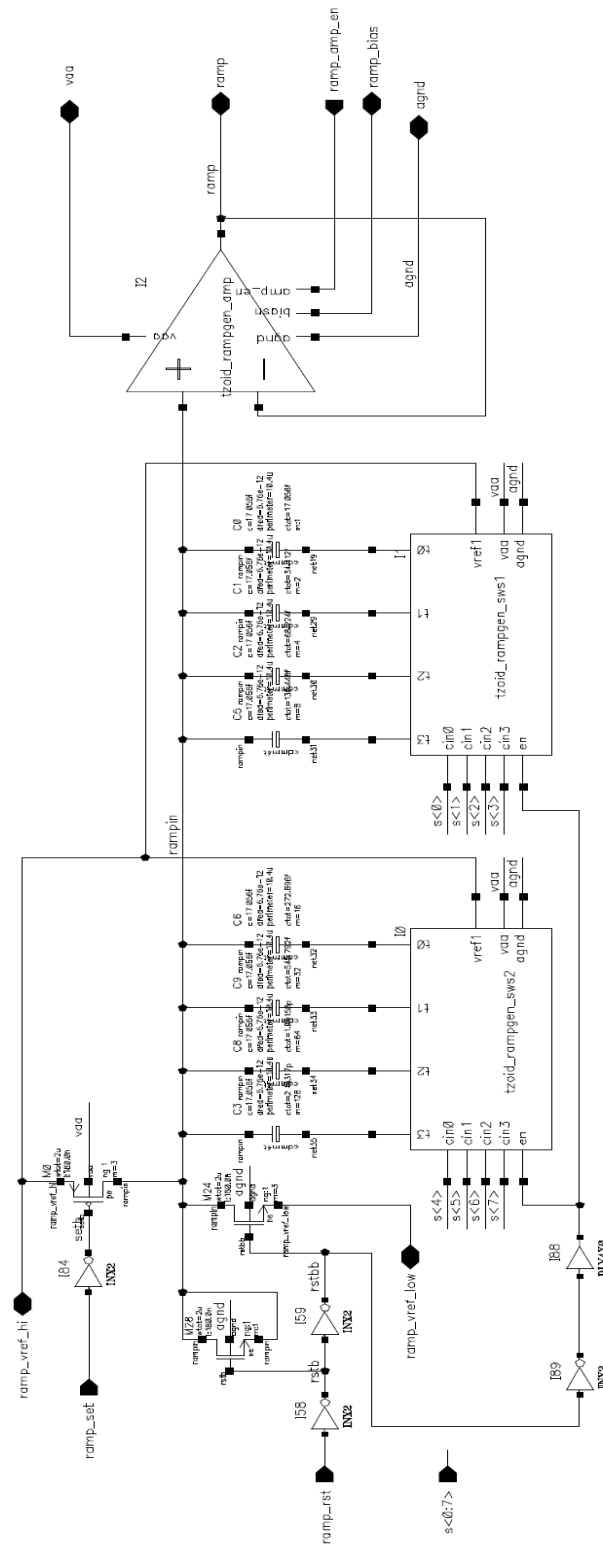


Figure A.16: Circuit diagram of the ramp generator.

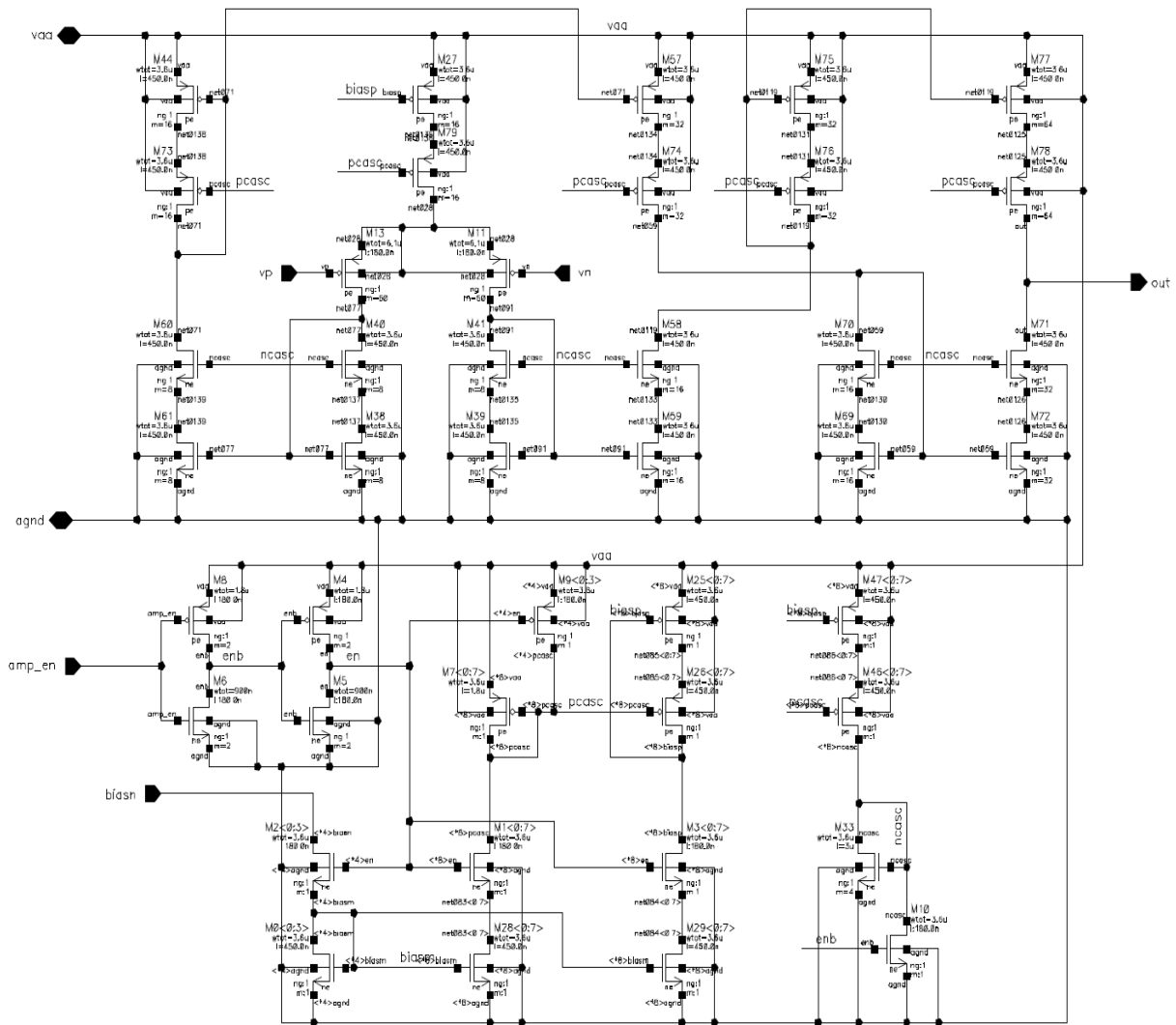


Figure A.17: Circuit diagram of the operational amplifier (A2) for ramp generator.

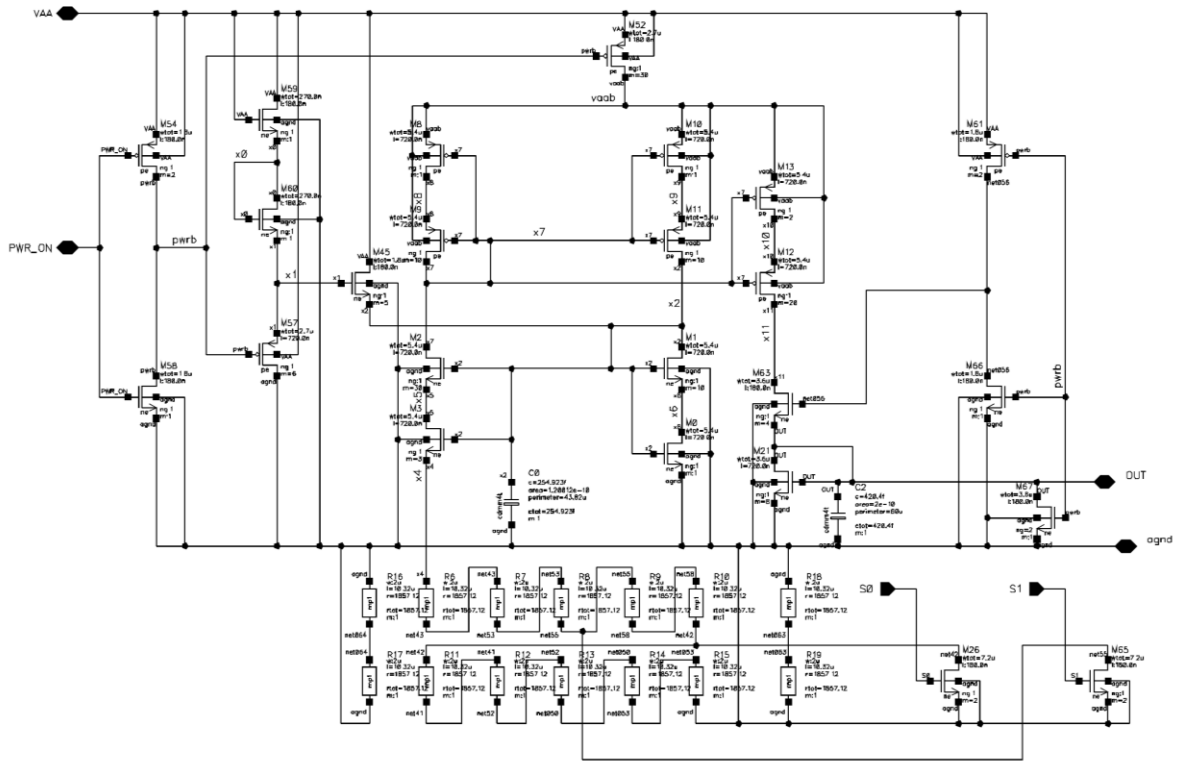


Figure A.18: Circuit diagram of the programmable reference current generator.

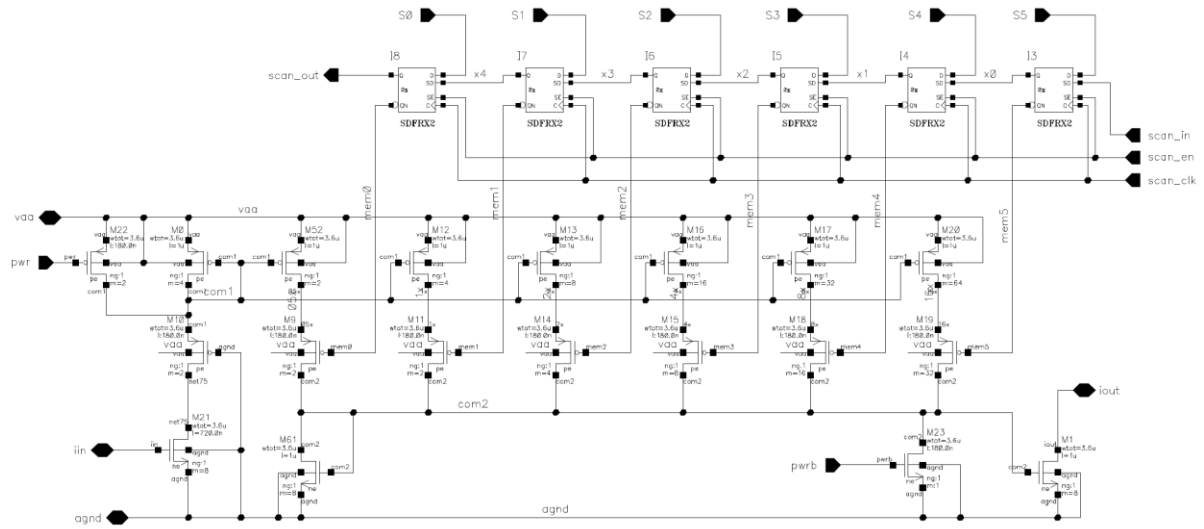


Figure A.19: Circuit diagram of 6-bit programmable current DAC.

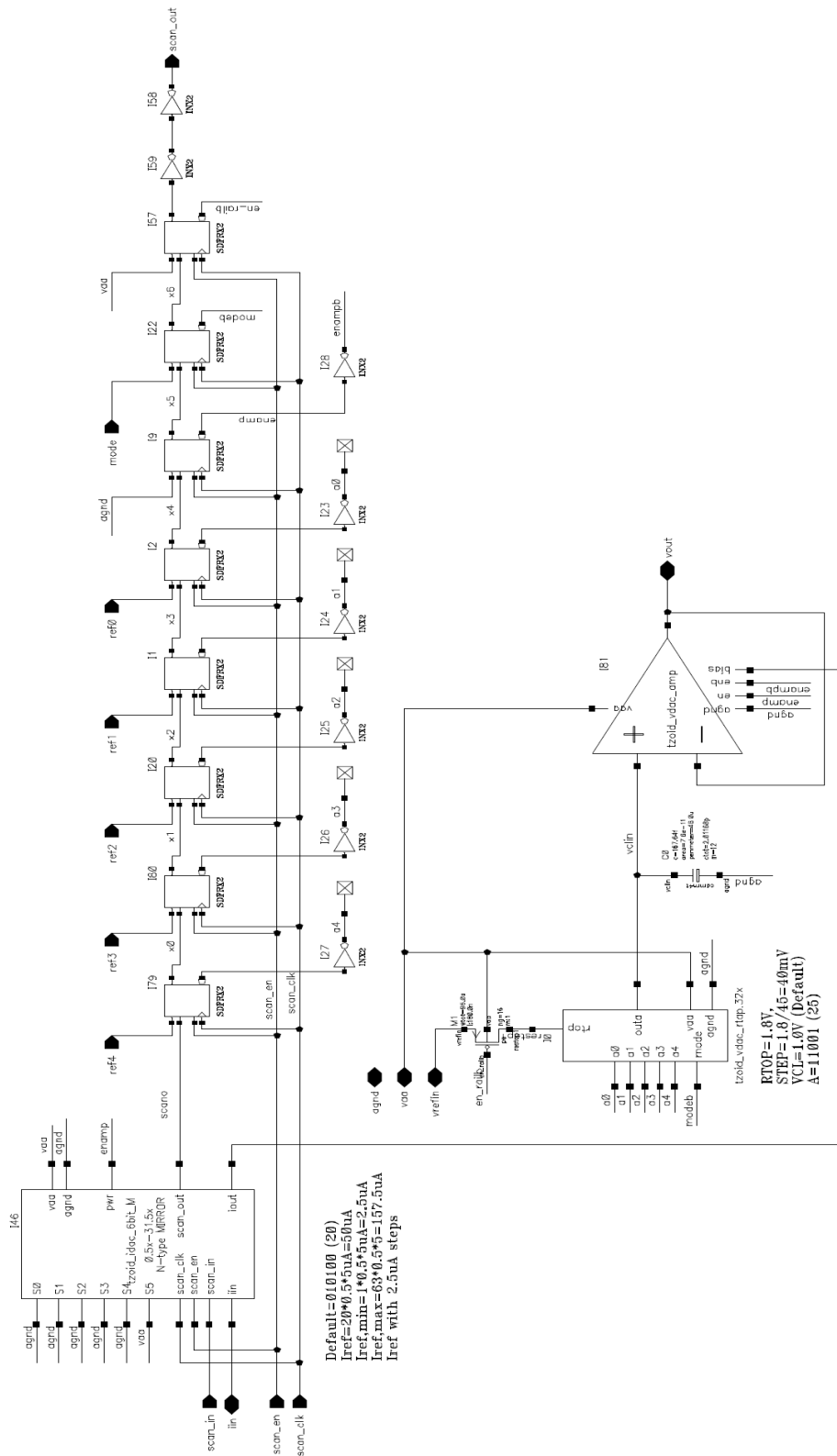
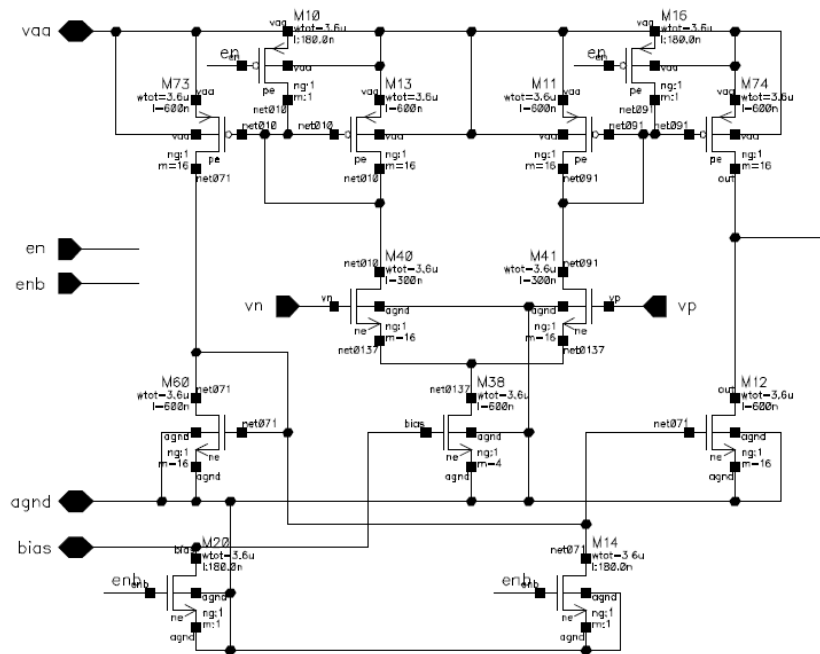


Figure A.20: Circuit diagram of the 6-bit voltage DAC.



$CL=1\text{pF}$, $I_{\text{cure}}=10\text{--}150\mu\text{A}$
 $V_{\text{cm}}=1.25$, $A_v=52.10\text{dB}$, $PM=35\text{--}36\text{degree}$,
 $V_{\text{cm}}=1.00$, $A_v=53.50\text{dB}$, $PM=38\text{--}39\text{degree}$,
 $V_{\text{cm}}=0.70$, $A_v=54.25\text{dB}$, $PM=49\text{--}50\text{degree}$,
 $CL=2\text{pF}$, $I_{\text{cure}}=10\text{--}150\mu\text{A}$
 $V_{\text{cm}}=1.25$, $A_v=52.60\text{dB}$, $PM=50\text{degree}$,
 $V_{\text{cm}}=1.00$, $A_v=53.50\text{dB}$, $PM=52\text{--}53\text{degree}$,
 $V_{\text{cm}}=0.70$, $A_v=54.25\text{dB}$, $PM=62\text{--}63\text{degree}$,

Figure A.21: Circuit diagram of the voltage DAC OPAMP.

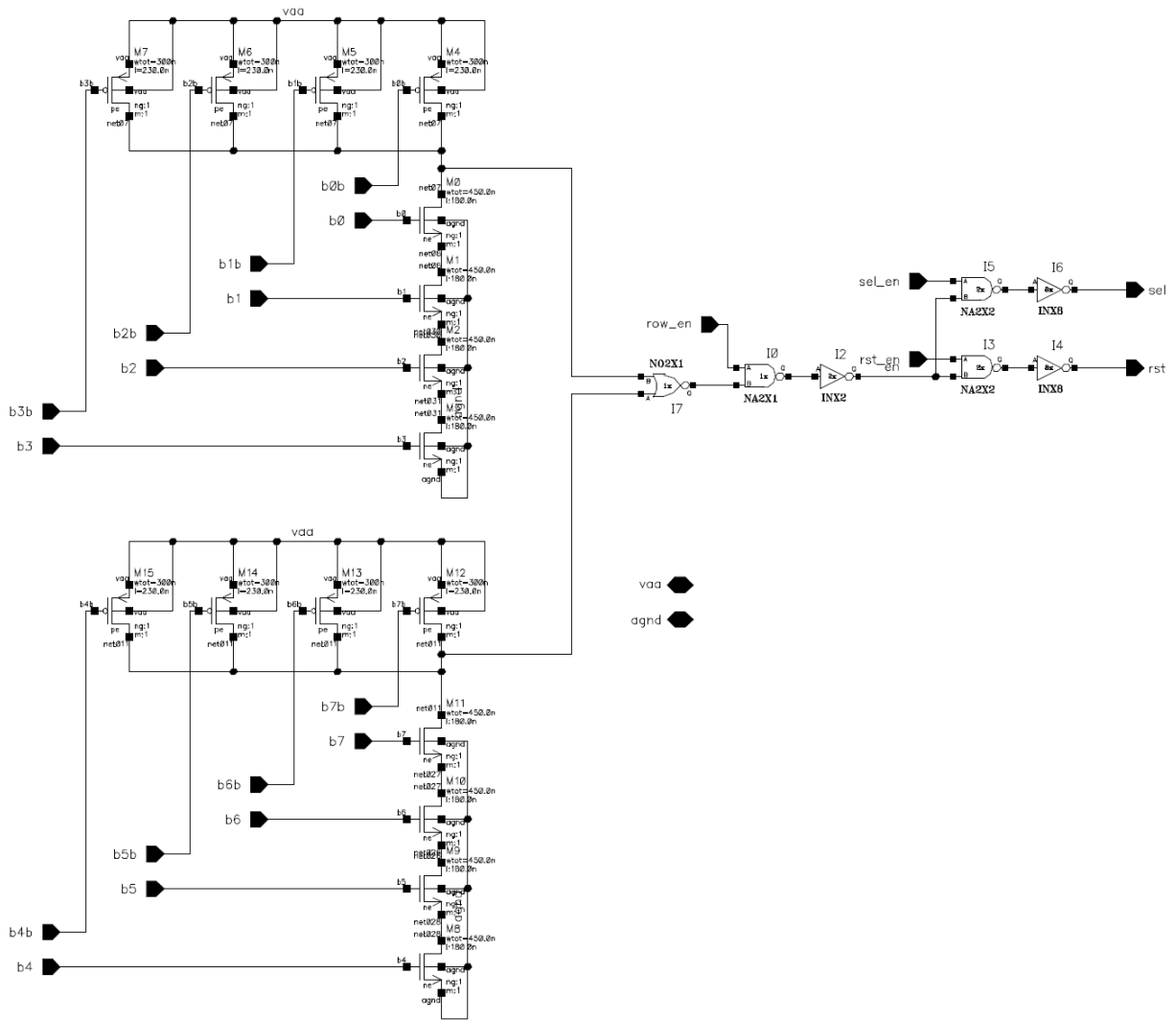


Figure A.22: Circuit diagram of the row decoder and driver slice.

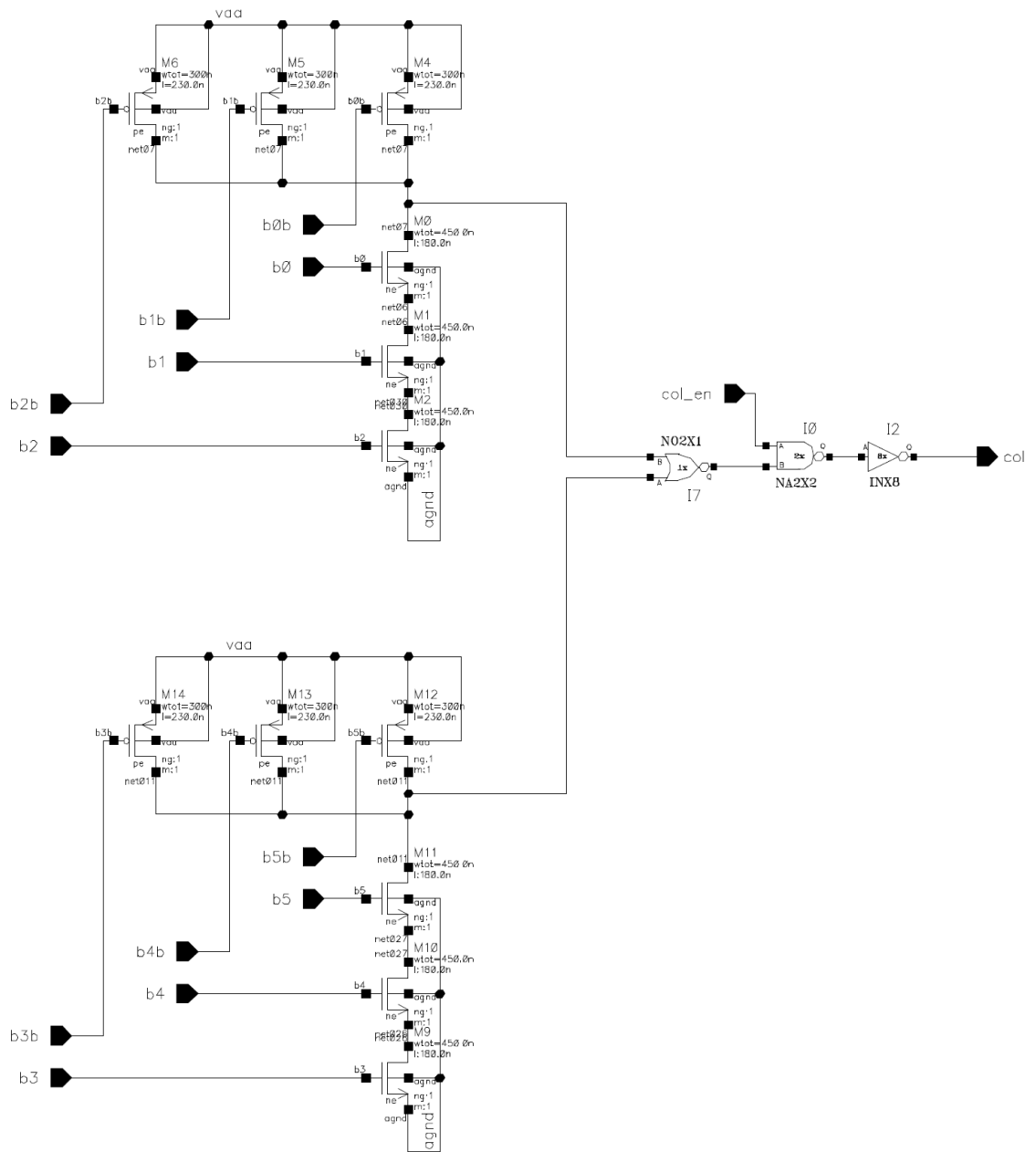


Figure A.23: Circuit diagram of the column decoder and driver slice.

TZOID Imager and Camera Pictures

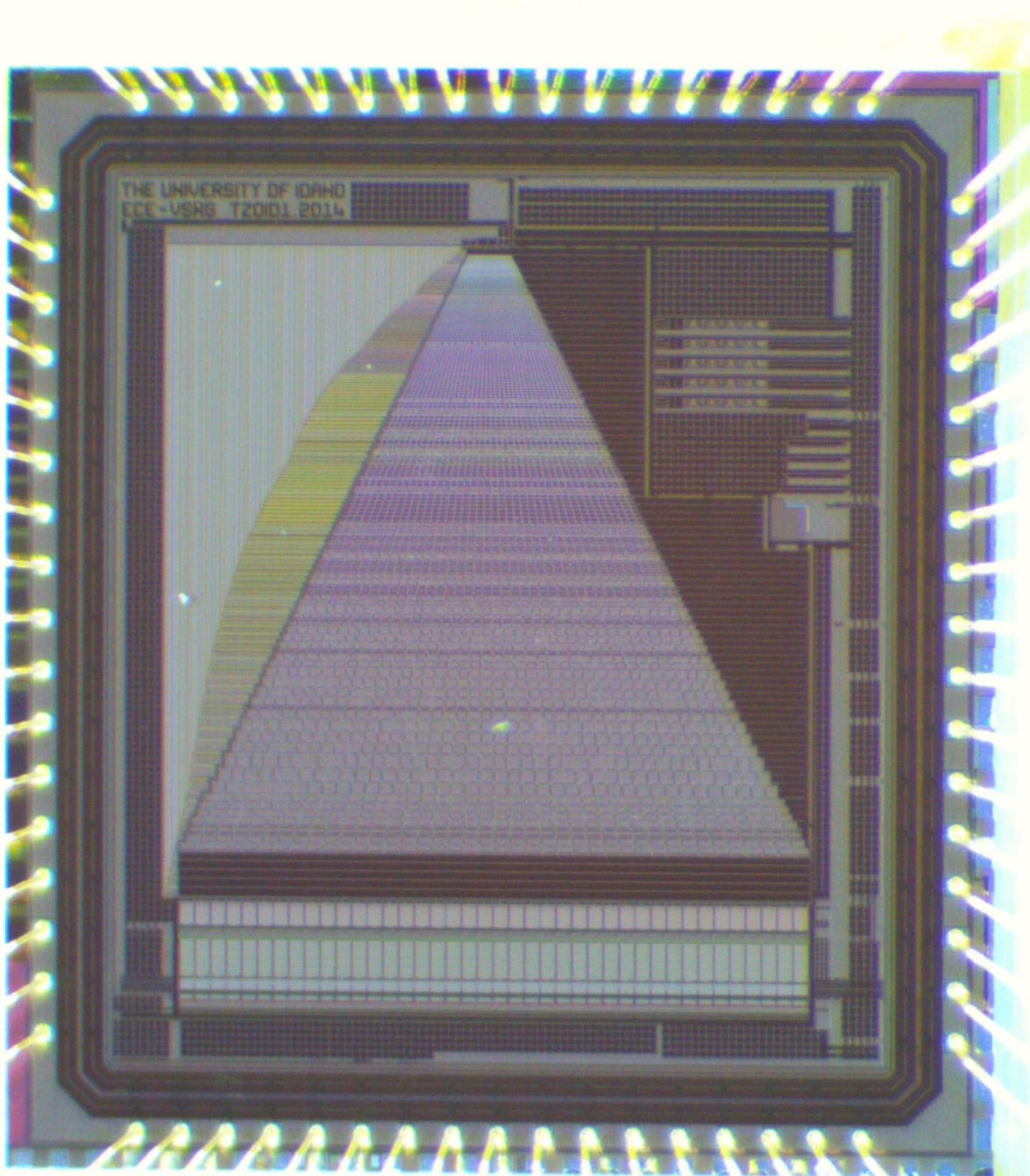


Figure A.24: Micrograph of the fabricated and packaged TZOID imager.

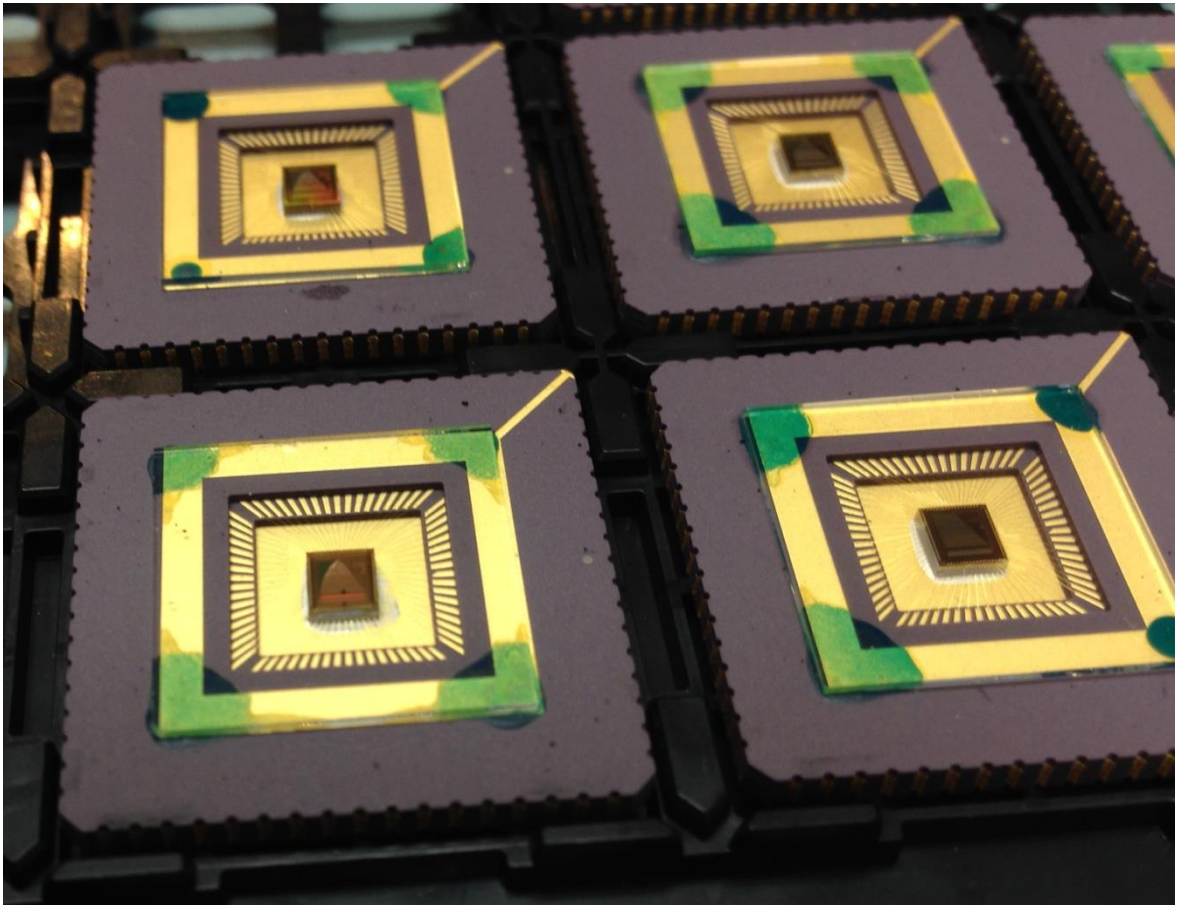


Figure A.25: Pictures of the fabricated and packaged TZOID imager ICs.

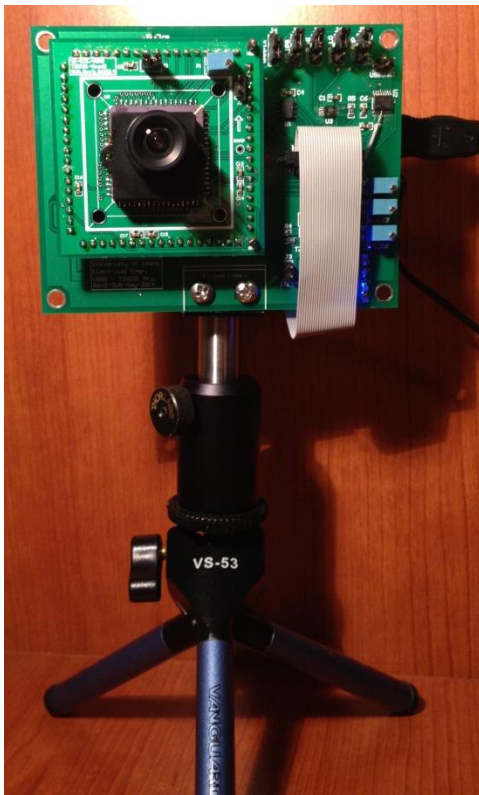


Figure A.26: Picture of the TZOID camera system.

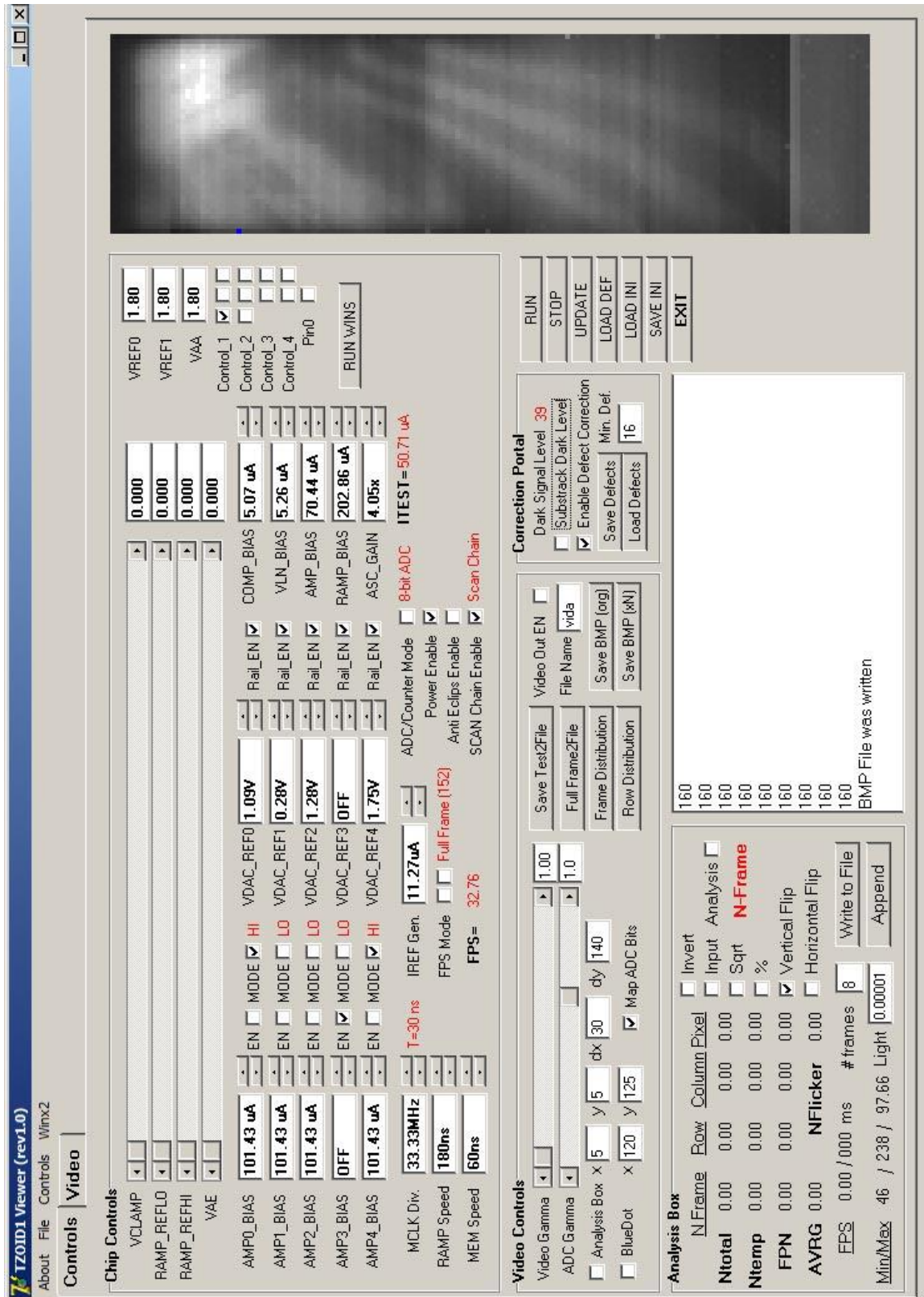


Figure A.27: Picture of the TZOID camera user interface program.

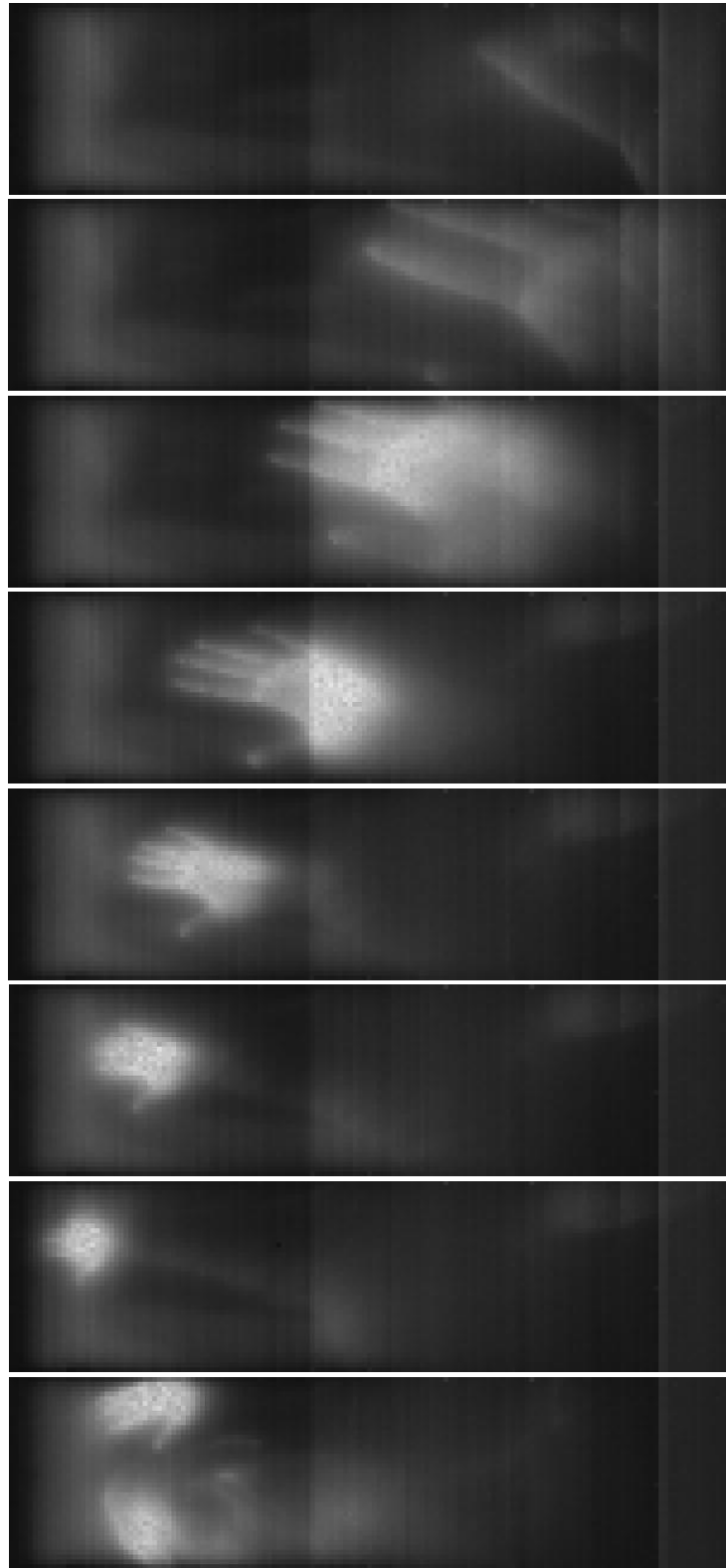


Figure A.28: Sequences of images taken by TZOID camera at 32FPS.

Student Algorithm Design Poster (UI Engineering EXPO, May 2014)

CREATE. INTEGRATE. ADVANCE.
CREATE. INTEGRATE. ADVANCE.





Image-Based Real-Time Traffic Detection

Team New Perspective



Project Sponsors

- Dr. Suat Utku Ay
- NIATT (National Institute for Advanced Transportation Technology)
- VSRG (VLSI Sensors Research Group)

Project Mentors

- Kyle Swenson – Computer Engineering
- Ismail Cevik – Electrical Engineering

Project Sponsors

- Dr. Suat Utku Ay
- NIATT (National Institute for Advanced Transportation Technology)
- VSRG (VLSI Sensors Research Group)

Faculty Advisor

- Dr. Touraj Assefi

Vehicle Detection Method Comparison


RADAR/LIDAR	Image-Based
<ul style="list-style-type: none"> • Accurate • Easy to Install 	<ul style="list-style-type: none"> • Cheap • Easy to Install
<ul style="list-style-type: none"> • Expensive • Complicated • Difficult to Install • Difficult to Replace • Binary Data 	<ul style="list-style-type: none"> • Calculation Intensive • Privacy Concerns • High Data Rates
<ul style="list-style-type: none"> • Current Standard • Reliable • Simple 	<ul style="list-style-type: none"> • Unreliable • Expensive • Privacy Concerns • Unreliable
<ul style="list-style-type: none"> • Inductive Loops 	<ul style="list-style-type: none"> • Accurate • Scalable
<ul style="list-style-type: none"> • GPS 	<ul style="list-style-type: none"> • Accurate • Scalable • Expensive • Privacy Concerns • Unreliable

Project Summary

In response to the growing concern over intersection safety issues, traffic related pollution, and rising gas prices, we have developed a unique, real-time, image-based vehicle speed detection system. Our design outperforms its competition by delivering real-time data without any of the privacy issues, high data rates, or complicated algorithms that plague real-time image-based traffic detection today. To reduce future costs and improve responsiveness, our design is implemented on a Field Programmable Gate Array (FPGA) that will later be combined with an image sensor, providing a single-chip solution for real-time image-based traffic control.

Design Process

As an early proof of concept, we tested our Tripwire and Image Processing algorithms using a software library called OpenCV. The figure to the right shows tripwires tripping in response to a moving input.



Our image processing control system sits directly between the camera and the outside world (left). We send only minimum data because all processing is done on-chip, greatly reducing controller processing requirements.

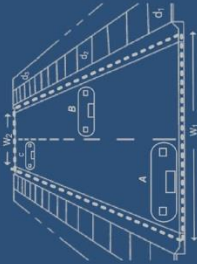
After extensive hardware testing, we configured the FPGA with our algorithms to process real-time image data captured from the SSLAR2, a revolutionary CMOS image sensor developed by Dr. Suat Ay. This camera system is shown on the right.

Future Project Goals

Future project goals include:


- Improving algorithms in low contrast situations such as snowy or low-light conditions
- Improve algorithms for noise filtering
- Integrate image processing chip and image sensor into an Application-Specific Integrated Chip (ASIC)
- Use of Trapezoidal Image Sensor (below)

Image-based motion detection algorithms often use Cartesian coordinates (left). These algorithms are useful when objects are not moving in one axis, as they are in a traffic situation. However, they also work well for removing image noise and in low contrast situations. We would eventually like to implement some form of these algorithms into our system.



Eventually, we would like to integrate a trapezoidal image sensor (right) into our system. This would greatly simplify our control algorithms as it mirrors the shape of the road, meaning no geometric pre-processing of the image would be necessary. Pixel projection would also be uniform no matter where the pixel appeared on the road, so the distance between tripwires would stay constant. This would relieve our system of even more calculations.

Inventions and Innovations

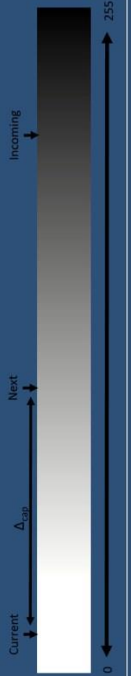



Tripwire Method


To deal with the data rate issues encountered in image-based motion detection, we decided to only monitor certain rows in our image sensor. We have termed these rows "tripwires". The highly efficient and innovative nature of the tripwire design makes speed detection trivial; we simply measure the time it takes for a vehicle to travel between tripwires and use the distance between them to determine vehicular speed. Furthermore, the tripwires eliminate privacy concerns because there is insufficient resolution in one row of pixels to determine anything beyond the presence or absence of a vehicle.


Threshold Method

In image-based motion detection, it is necessary to generate several frames (background, foreground, and current) that are used to describe moving objects. Commonly used algorithms for generating these frames are computationally expensive, requiring high performance processors to satisfy real-time requirements. Our technique eliminates the need for high speed processors by implementing a modified moving average filter where the maximum amount of change in the average caused by a new pixel is limited by a maximum threshold value. This has the effect of allowing small changes to be absorbed into the background (average) quickly, while delaying the incorporation of significant changes into the background.









CREATE. INTEGRATE. ADVANCE.
CREATE. INTEGRATE. ADVANCE.

University of Idaho
College of Engineering

EXPO 2014

A High-Speed Trapezoid Image Sensor Design for Continuous Traffic Monitoring . . .

53