Integration of an Imaging System and A/D Converter System

by

Jasen C. Li

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

January 1, 1997
[June 1997]

Author______
Department of Electrical Engineering and Computer Science
January 1, 1997

Certified by__________
James K. Roberge
Thesis Supervisor

Accepted by
F.R. Morgenthaler
Chairman, Department Committee on Graduate Theses

Integration of an Imaging System and A/D Converter System

by

Jasen C. Li

Submitted to the
Department of Electrical Engineering and Computer Science

January 1, 1997

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

One of NASA's most famous centers, the Jet Propulsion Lab has a joint venture with MIT. The project is a camera demonstration system that has specific output modules. A high-speed amplifier interface will drive both a display and an analog-to-digital converter (ADC) board. The display interface is of higher priority because it is necessary to validate the camera system. The system consists of a joystick to center the field-of-view, an array controlling the timing, an image sensor, and a video monitor. The second interface consists of an MIT built ADC board, part of a wireless sensor system, and the image sensor. The demonstration of the systems will hopefully verify an application for JPL and also show that the MIT sensor system is capable of being a platform for a variety of sensors. Another objective is to help both groups identify enhancements that will ease interfacing to full systems in the future.

Thesis Supervisor: James K. Roberge [illegible]
Title: Professor of Electrical Engineering

Table of Contents

Acknowledgments

I would like to thank Dr. Eric Fossum at JPL for hiring me and allowing me to work in an exciting group and environment. I also want to thank Prof. Roberge for being my undergraduate advisor for four years and finishing it off by supervising my Master's thesis.

0. Introduction

The motivation is to simultaneously satisfy current objectives of the Jet Propulsion Laboratory (JPL) and the Microsystems Technology Laboratory. (MTL) Located in Pasadena, CA, JPL would like to see how compatible its sensors are in camera systems or other types of imaging systems. On the other hand, MTL based at the Massachusetts Institute of Technology in Cambridge, MA, is determining whether its wireless sensor system can be a platform for an array of different sensors. A serious interest in low-power applications and design further ties the two research groups together.

1. Jet Propulsion Laboratory

One of the main goals of the Advanced Imager Technology group at JPL is to design, test, and fabricate image sensors using mainstream CMOS technology. The group has developed an active pixel technology that has several advantages over charge-coupled devices. (CCDs) These active pixel sensors (APS) retain nearly all the performance of a CCD but use about 100 times less power while using a standard 5 volt (Or 3.3 volt) supply. The possibility of providing functionality on-chip such as analog-to-digital conversion, clocking, power conditioning, etc. also allows miniaturization. Other reasons to choose APS include fast, digital readout and also cost, since dedicated CCD fabrication lines are not needed. Applications include robotics, miniaturized imaging instruments, and night vision.

The first objective at JPL is to build a small camera system that displays a low resolution image. A system description includes a joystick interface, an image sensor, a TV monitor, and an analog-to-digital conversion (ADC) board. (Fig. 1) The joystick will

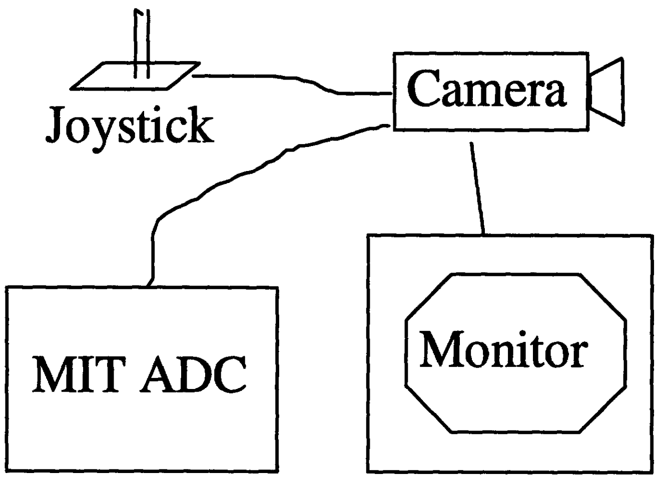


Figure 1: Camera System Block Diagram

center the field of view and A/D conversion of the joystick's output will produce the correct starting address for a field programmable gate array. (FPGA) In addition to being given the window starting address, the FPGA will require a 16 MHz clock input in addition to standard TV monitor signals. (FPGA paper in appendix) It will then provide the correct row and column addresses along with the timing signals for the image sensor. A 1K x 1K APS image sensor will be used and its output will be sent to two modules. The first interface will involve a black-and-white display. Standard NTSC timing will be used, namely 30 frames/sec. Another interface will be between the image sensor and the ADC board. Analog and digital circuitry will be designed, built, and implemented to correctly drive both outputs. The responsibility within this camera demo system (Fig. 2) will be these interfaces along with part of the FPGA design.

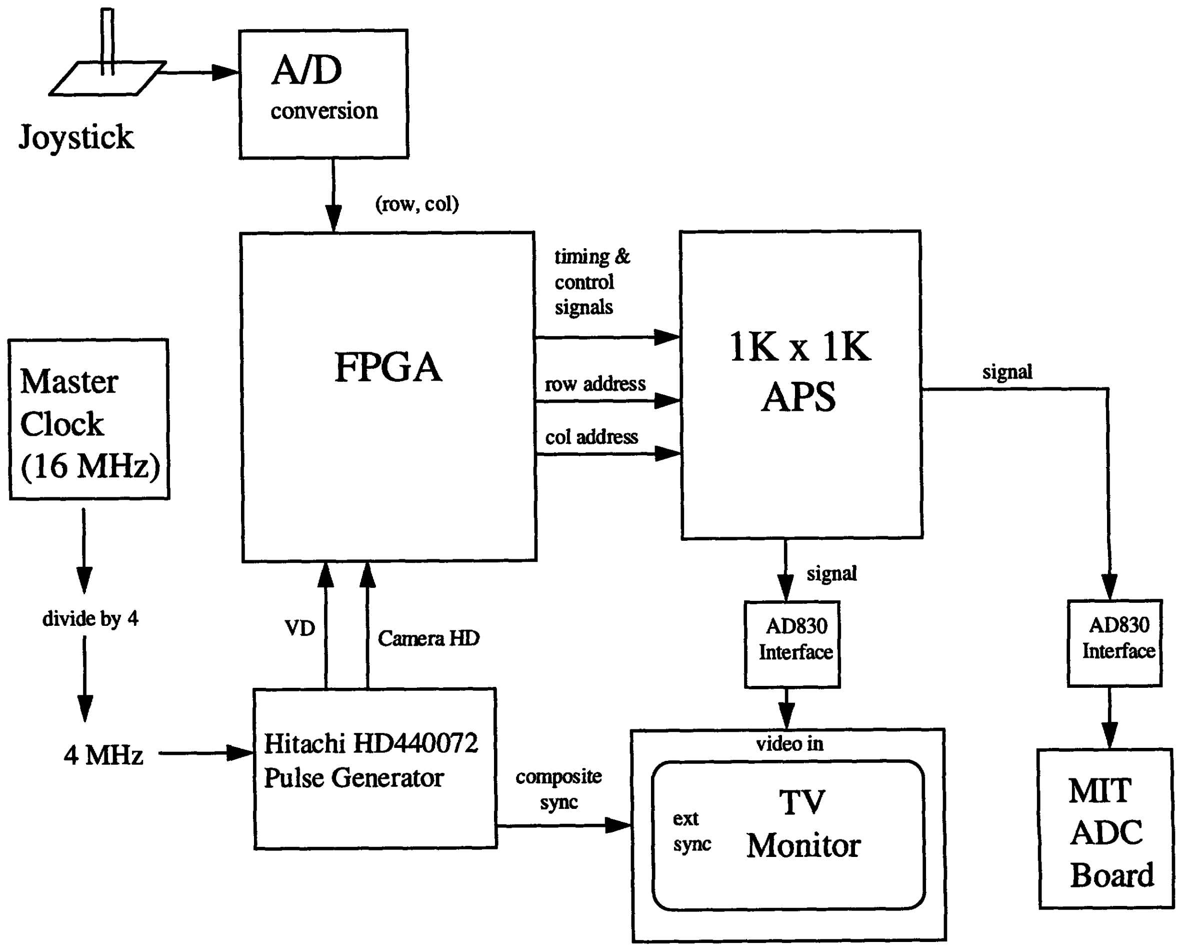


Figure 2: Camera Demo System

2. Microsystems Technology Laboratories

The other system scenario is the collaboration between JPL and MTL. MTL has a broad variety of interests but one important task is a wireless sensor system. (Fig. 3) The main consideration is low power design not just for a single module but in terms of an entire system. One desire is to design, fabricate, and characterize an image sensor system that can wirelessly transmit at data rates of 1 bit/sec to 1 Mbit/sec over output power levels of 10 uW to 10 mW.

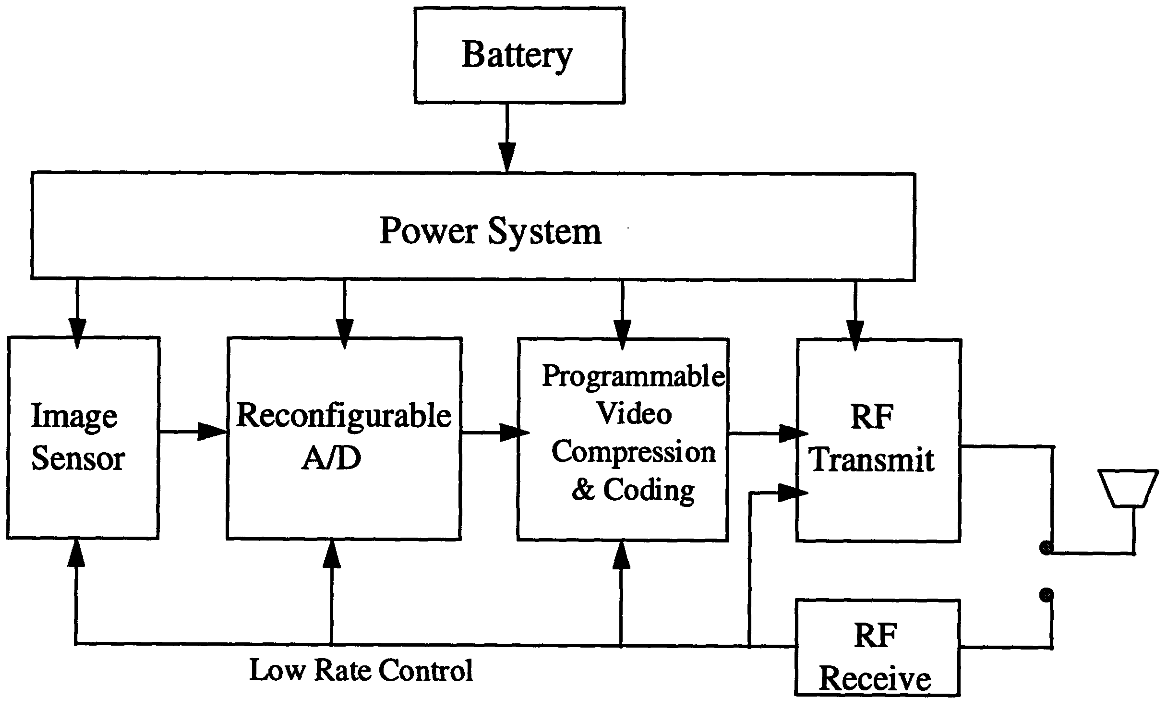


Figure 3: Wireless Sensor System

A team of experts from different areas has been assembled to design specific modules. The block diagram includes several circuits, but the specific area that has been designed is the interface between the image sensor and the reconfigurable A/D. An image sensor was chosen because it has a variable data rate and because APS has extremely low power dissipation. A CCD sensor has dissipation due to high powered clock drivers because of the large gate capacitance. It is hoped that the use of the current imager will reduce the need for much energy recovery circuitry although those techniques will be investigated. To be compatible with different resolutions and speeds, a reconfigurable A/D converter will be placed in the system. Currently, the system uses a pipeline architecture optimized for op-amp use to save power.

Another module will perform the video compression of the digitized imager data. Dynamic algorithms and variable power supplies will also make efficient use of power. In addition, the RF circuits are designed with low power dissipation in transmitting digital data for a wide range of data rates. Applications for this wireless sensor system include security systems or portable camcorders.

Most importantly, the MTL group has designed a low power/low voltage A/D converter. One feature of the converter is digital error correction using redundant encoding of the decision bits. It also uses an S-C (Switched-Capacitor) dynamic common-mode feedback method to make the op-amp common-mode voltage stable. Recent measurements show good performance but the A/D converter needs more characterization.

Essential blocks to test the ADC board include an image sensor, an acquisition board, and a computer to store the final data and display information. (Fig. 4)

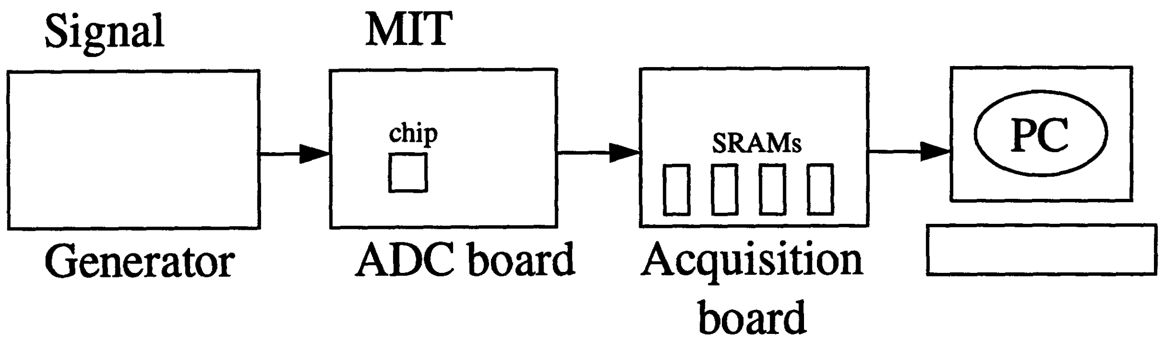


Figure 4: MTL test setup

The test setup uses a signal generator in place of the image sensor to provide a clean sinusoidal signal. This signal is then digitized for linearity and histogram tests. However, the goal is to replace the generator with the image sensor's differential analog output by correctly interfacing the two components. The custom ADC board has a socket for the chip, provides the various analog and digital control signals, and prepares the digital output. Further processing is done by the acquisition board that contains digital hardware to store the digital codes provided by the ADC board. When the acquisition is finished, the computer takes control of the acquisition card and obtains the digital codes. A routine written in C processes these codes to achieve the equivalent binary codes. MATLAB can then perform linearity tests, sorting the data and displaying the appropriate plots.

3. Summary of Goals

Common to both projects is the image sensor and ADC board interface. Characterizing the A/D converter, understanding its requirements, and determining a comprehensive set of interface specifications is one task. A test board holding the A/D converter will then be completed.

Hardware providing the appropriate analog signals and clocks necessary to interface the imager to the appropriate output modules will be the main design objective. Other requirements include level shifting and gain to transform the sensor's output voltage range to the correct input voltage range for the A/D converter and the monitor. Additional considerations include speed, termination to limit signal reflection, and decoupling of power supplies to reduce noise. The main demonstration will display the image sensor's output on the black-and-white screen. The second system demonstration will include the interface between the JPL imager and the MTL designed A/D converter, with the bitstream acquired by a workstation. That workstation will demonstrate digital functions, namely compression and encryption in software, and display the results. However, the responsibility of this project is the interface. The task specification includes neither the code density tests for the A/D converter nor the computational work of the PC. MTL is responsible for the latter tasks.

The demonstration of the two systems will hopefully verify an application for the JPL group and show that the MIT wireless sensor is capable of being a platform for a variety of sensors. Another objective is to help both groups identify any enhancements that will ease interfacing to full systems in the future. An important constraint imposed is that the collaboration should not duplicate any current efforts nor require any substantial budget.

4. Thesis Work

4.1 APS Imager

The Advanced Imager group designed a 1K x 1K CMOS active pixel sensor (APS) imager that has been fabricated in the HP 1.2 micron CMOS process. The imager had a differential analog signal output that could be interfaced to a generic data converter. At first, the CC256C was chosen for its lower read-noise. Then the CC256D was selected, a 256 x 256 photo-diode sensor that had several advantages:

(1) greater full well capacity

(2) better blue response, and QE (quantum efficiency) in general

(3) lower dark current after radiation

(4) higher fill factor

In the end, the newer 1K x 1K sensor was picked to see how well it would fit into a camera-type application.

Initially, research was done to characterize the imager's output. The output was differential (Sout and Rout) with a common-mode level of 1.5 volts. Measuring the signals, one went from 1.5 V to 1.8 V and the other from 1.2 V to 1.5 V resulting in an absolute difference range of 0 to 600 or 700 mV. Also, the data rate was about one Msample/sec. It was then discovered that the output was only valid for the high part of the clock cycle. Thus research was done testing some sample-and-hold designs to provide the output continuously. The main requirement was that the interface should drive the signal cleanly at video rates.

4.2 A/D Converter Board

The following table lists the important specifications of the A/D converter:

MIT Analog-to-Digital Converter Specifications	
Process	HP 0.8 micron CMOS, 1 poly, 3 metal
Chip Size	3.25 mm x 3.65 mm
Minimum power supply voltage	Single 1.8 V
Power consumption (at 1.8 V)	5 mW
Miscellaneous	2 Megasamples/sec, 10 bit resolution

The work done included reading and comprehending all the background material including Vincent Peiris' paper on his A/D converter, the schematics, and testing procedures. (Silk-screen and code in appendix) In the paper, Peiris describes his converter, the reasons for his design configuration, and the specifications. After reading through the cryptic schematics and figuring out all the correct parts and values, the board slowly came together. A few hundred analog and digital components were soldered onto this massive board. Then the board was tested using the MTL setup and verified for functionality. A program written in C (vp_acq1.c) was used to acquire the converter data, which was put into a file. Then David Martin wrote another C program (code.c) to analyze the data and split the data into two data sets. These sets were divided for differential non-linearity and integral non-linearity plots. Furthermore, a MATLAB script (data.m) was used to analyze the data sets and display the plots. More work was done to modify both the C code and MATLAB code to fit the necessary tests.

Input characterization was as follows. With AVDD set to +1.5 V and AVSS to -1.5 V, the input voltage allowed was -0.9 to +0.9 volts and a data rate of approximately 2 Msamples/sec.

4.3 Display

A v19/v20 ElectroHome monitor was used in the demonstration. In preliminary testing, an analog signal was fed into the video A input and an external sync was provided by a Hitachi HD440072 integrated circuit. This 28-pin IC required a 4 MHz master clock and ran off of a 7 volt power supply. In turn, it provided a composite sync that was used as the sync for the monitor. Two other standard NTSC signals were also generated. One was VD (Vertical drive), a field clock, and the other was camera HD (Horizontal drive), a row clock required by the gate array in charge of the system timing. Here, system design really paid off. The initial design had extra components to synchronize the system. However, doing some research and choosing the Hitachi chip solved everything by providing signals to synchronize the display, FPGA, and image sensor.

Initially, a one-volt peak-to-peak (Allowed input range was 0.5 Vpp to 2.0 Vpp) square wave was used as input. According to NTSC standards, the time for a row was 63.558 microseconds, resulting in a frequency of about 15.7 kHz. Since the blanking interval occurred during 10 of these microseconds, the expected result was a screen that was slightly more black than white or vice versa. This result was achieved.

4.4 Interface

The key to the interface was determining the input and output specifications. Having done that, some sort of differential amplifier was needed that could provide the correct gain and level-shifting. Other details included noise problems, speed, and termination. The basic interface was a high-speed buffer, a video-rate sample-and-hold circuit, and then an amplifier to drive the monitor.

4.4.1 First AD8011 Amplifier

A knowledgeable circuit designer might ask why some of the following designs were used. First of all, a video-rate op-amp was needed. Second, the AD8011, although possessing excellent speed, had different input impedances. Testing some of the standard differential designs resulted in some disappointing results with the AD8011. This fact forced some non-standard designs that made testing less tedious.

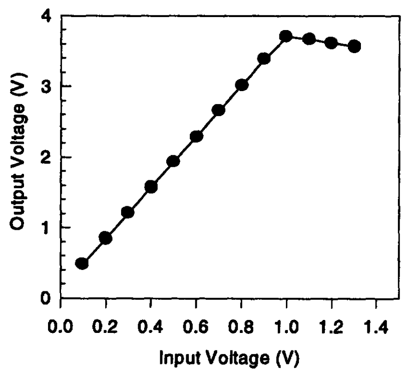
The first AD8011 based amplifier was designed with a gain of 3.7. (Appendix) The front two stages of the design were unity-gain buffers based on these op-amps, which had good noise and gain characteristics, (According to Analog Devices) out to 10 MHz. At first, the two unity-gain buffers did not have any resistors in the feedback loop, but the noise was intolerable and the addition of 10k resistors in the feedback solved the problem. Solving the circuit, the output voltage was calculated according to this equation.

$$Vout = 3.7 * (Signal - Reset) + Vref$$

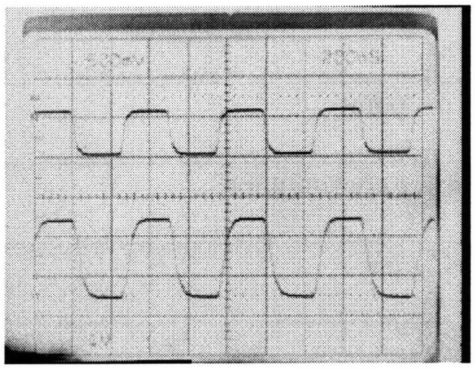
The following table shows some important statistics:

Parameter	Measured Result
3 dB point	approximately 8 MHz
power consumption	75.86 mW
DC gain	3.7 V/V
noise	414 uV
dynamic range	79.0 dB

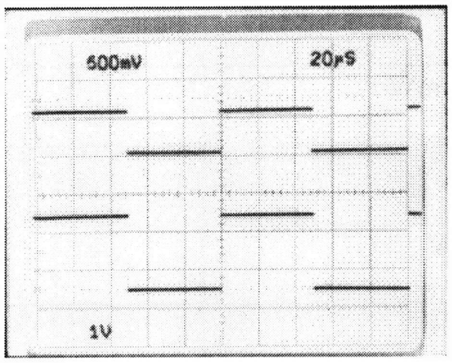
In the testing procedure, these characteristics were quantified: DC gain, AC gain, power consumption, amplifier noise, dynamic range, offset, and how Vref affected the amplifier. First, DC gain was calculated by varying the DC input from 0.1 V to 1.3 V and measuring the output voltage with an oscilloscope. The output versus input graph shows



the increase in output voltage up to 3.7 volts, where the op-amp saturated. In the AC gain measurements, an input of 580 mV peak-to-peak (0 V offset) was used, and this time the frequency was varied. Several scope photos were taken to qualitatively measure the waveform fidelity. Waveform reproduction started to degrade at about 2 MHz. (See photos A, B).

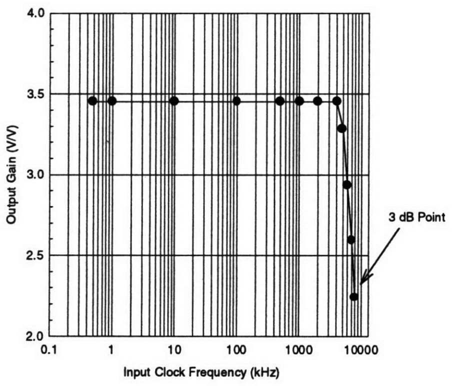


2 MHz



10 kHz

The graph below shows the AC gain versus frequency. The 3dB point for this data was between 7 and 8 MHz. For power consumption, the V+ (5.47 V, V- side was at -5.67 V) side current was measured to be 6.81 mA, resulting in a power of 75.86 mW. In these calculations, both the reset signal and Vref were set to ground.



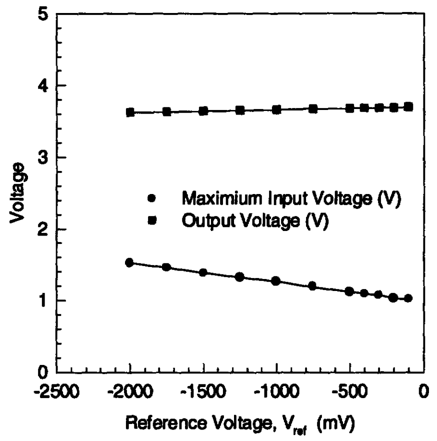
Input Clock:
Square wave
580 mV_{p-p}
Offset = 0.0 V

The noise measurements were more complicated since pulse instrument clocks were used. Using a data acquisition program, a 3x3 array and 500 frames per datum were taken and a noise level of 414 μV was measured. The system noise was measured to be approximately 34 μV, a negligible amount in calculating rms noise. The dynamic range of the amplifier was calculated to be

$$-20*\log[\text{amplifier noise/max output voltage}]$$

$$= -20*\log[0.414\ \text{mV}/3700\ \text{mV}] = 79.0\ \text{dB}$$

How the offset changed as a function of frequency was also observed and found to increase with frequency. In addition, Vref was varied and the maximum input and output voltage for saturation were measured. As Vref was made more and more negative, the maximum input voltage increased. This was useful because the desired input range centered around 1.5 V. (See graph) Finally, testing waveform fidelity, Vref was set



to -2 V and waveform fidelity did not degrade compared to when Vref was set to ground.

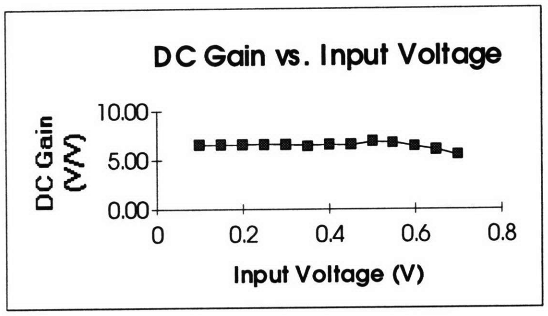
In conclusion, the amplifier input-output characteristics followed the equation closely. Power consumption and noise were fairly low but the input voltage range was tight unless Vref was added.

4.4.2 Second AD8011 Amplifier

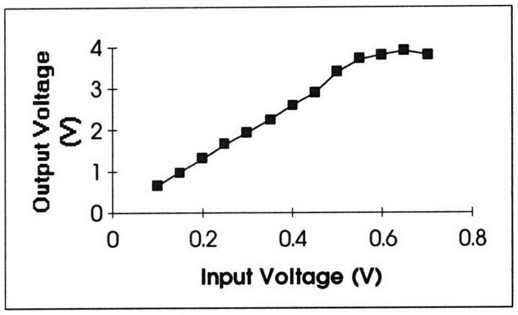
Advantages of the second AD8011 based differential amplifier (Appendix) were two-fold. First, it used only two op-amps, and second, by changing one resistor value, the gain could be set. Although the design was unconventional, it simplified testing and performed well in other applications for the Advanced Imager group. In any case, it was not used in the final interface.

The gain for this circuit was 2 * (R5 + R) / R5. During the testing phase, R was set to 10K and R5 to 6.8K to hopefully keep power consumption low. However, when using a square wave input, an overshooting problem was noticeable. This phenomenon was especially obvious at frequencies over 1 MHz. An attempt to correct the overshooting was to reduce R to 1.5K and R5 to 618 ohms. This variation halved the settling time from 50 ns to 25 ns but did not decrease the ringing. Finally, a 50 ohm resistor on the input and a voltage divider at the output drastically diminished the problem. Both of the modifications were made with the knowledge of simple circuit theory. One side effect was that the 50 ohm resistor caused the amplifier to see half of the original input signal. After these modifications, there was only slight ringing caused by the system and loading effects. This was easily solved by using normal termination, i.e. 50 ohms when connected to the ADC board.

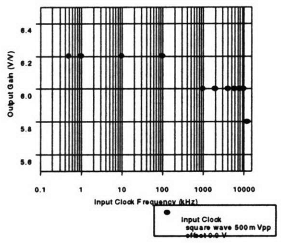
The optimal gain for this amplifier with the given resistor values should have been around 6.85. As shown on the left, the maximum gain was close to 6.8 at 500 mV, but the



range of inputs was limited to about 0.7 V before saturation. The output versus input graph also shows good slope characteristics. As for the AC gain, a 500 mV peak-to-peak



sine wave was used and output versus frequency was measured. The graph below shows that there was little change in amplitude for frequencies up to 12 MHz.



Noise and dynamic range measurements were made in the same manner as the testing of the first amplifier. For this circuit, total noise was measured to be in the range of

160 uV and the system noise to be 30 uV, again negligible. The dynamic range result was 89.5 dB, a bit higher than the first circuit.

4.4.3 AD830 Amplifier

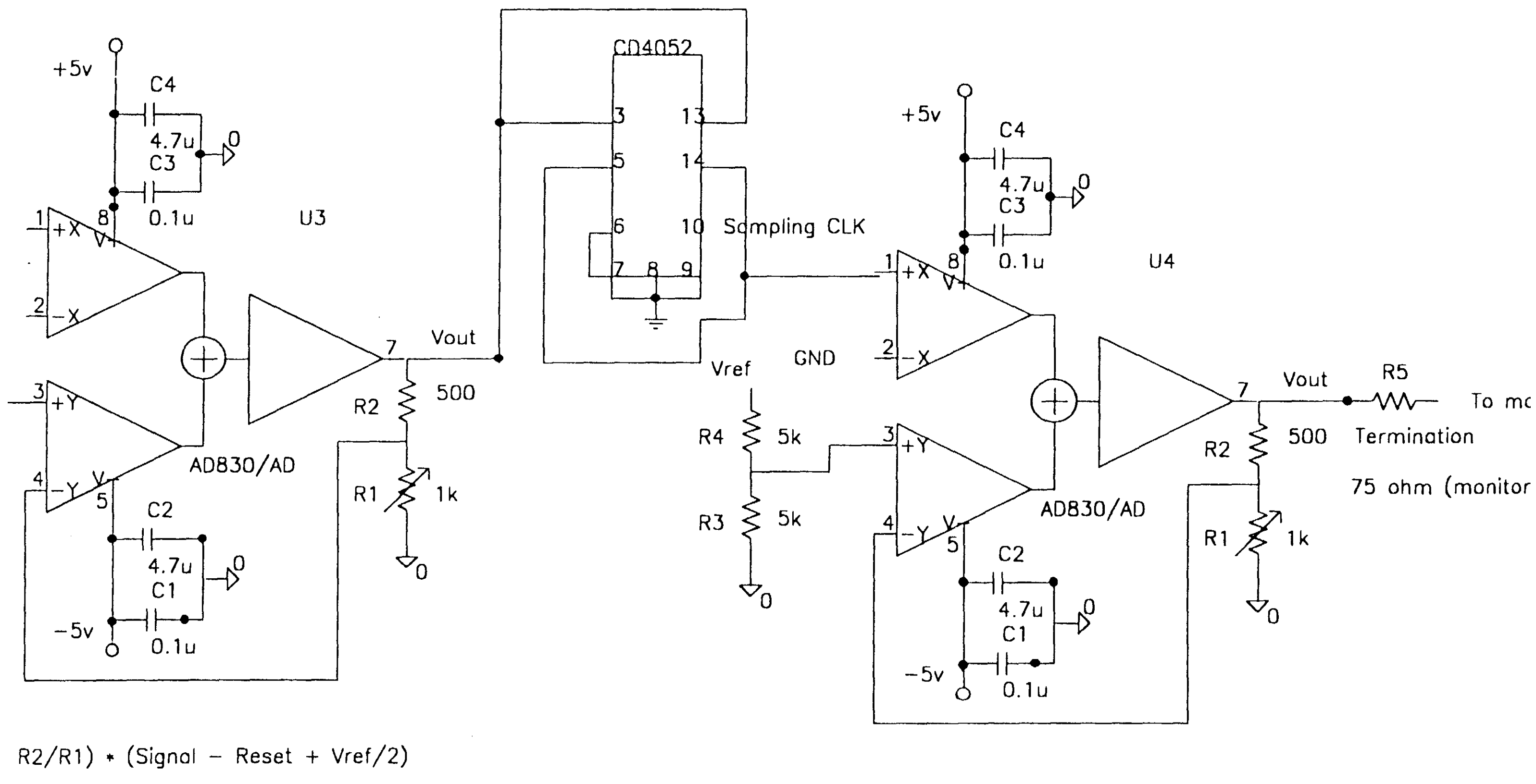
In the end, the best design used an AD830 based circuit because this high-speed amplifier was designed for video applications. Not only was the AD830 designed to drive high-speed signals, it could also difference two signals. (Interface schematic on the following page) Signal, the higher voltage input, and reset, were fed into the first differencing amp. Bypass capacitors were placed on the power supplies (V- for pin 5, V+ for pin 8) to reduce noise. (Again, general circuit theory) The general equation that described the op-amp

$$Vout = A * (V1 - V2 + V3 - V4)$$

was easy to verify. After plugging in the correct values, Vout = (1 + R2 / R1) * (Signal - Reset + Vref/2) was justified both mathematically and physically. During testing, both Vref and the pot were varied to confirm that the equation held. Since the characteristics were excellent, the identical AD830 based circuit was used for the buffer and amplifier.

Inter-connections for the interface were as follows. A differential input went into pins one and two of the AD830 buffer. The output of the buffer was connected to the input of the sample-and-hold (S/H) circuit. Then the S/H circuit's output went into the positive input of the AD830 amplifier while the second input was grounded. Finally, a 75 ohm termination resistor was used when driving the monitor.

AD830 Interface with Level-Shifting and Sample-and-Hold



R2/R1) * (Signal - Reset + Vref/2)

Vout = (1 + R2/R1) * (Pin1 - 0 + Vref/2)

With a basic understanding of how the AD830 worked, a summary of results was obtained. Overall, the amplifier generated a clean signal with good waveform fidelity past 2 MHz, and the overshoot was less than 2% with an input of 100 mV and output of 200 mV. (Resistor ratio of 3 since the termination cut the gain in half) The specification for settling time was 35 ns and during testing, a 25 - 30 ns settling time was observed for most combinations of gain and offset.

4.4.4 Sample-and-Hold

Like the driver circuit, many sample-and-hold circuits were examined and tested. However, to discuss each design would provide extreme boredom. Many of the circuits had problems in the bias current, capacitance relationship resulting in drastic droop. ($I = C * dV/dt$) Another problem was that the RC time constant was inadequate. The final solution used a 4052 integrated circuit containing CMOS transmission gates.

The 4052 was a multiplexor with two inputs and four outputs for each. S0 and S1 chose which of the four lines was chosen so S0 was grounded and the sampling clock connected to S1. NotG was also grounded to keep the multiplexor on at all times. At the output, a 100 pF capacitor was used as the sampling capacitor. At first this seemed adequate. However, the CMOS gates could not drive a signal at the desired 2-3 MHz. Therefore, to reduce resistance by one-half, the two inputs were tied together and the output corresponding to the S0, S1 selection was tied to the previous output thereby connecting two “resistors” in parallel. The results were surprisingly good with the buffer-

S/H circuit-amplifier combination driving a 2 MHz signal. There was no significant droop at that frequency and adequate rise and fall times.

5. Test Results

The main interest was whether or not the camera demo system could produce an analog signal. If so, the signal could be used by a TV monitor and by the A/D setup designed by MTL. The hypothesis was that if there was good quality on the monitor, the signal would be good enough for the MTL setup. To have good image quality on the TV, the analog signal had to meet NTSC timing standards, i.e. 30 frames per second and 63.56 microseconds for each row on the monitor. (10 of the 63 microseconds had to be blank) The FPGA was designed to control the image sensor such that data would be output for 50 microseconds per row and then stop, producing an image that almost covered the width of the screen with a small vertical strip of black on the right side.

In the final design, 500 ns per pixel of TV output was allocated, translating into a 2 MHz clock. The output of the image sensor would not be able to drive the monitor or A/D setup so an interface had to be designed and tested. An AD830 integrated circuit was the centerpiece of the interface, which had a settling time of 30 ns or so, insignificant with 500 ns per pixel. Voltage errors were around 3-5%, again insignificant with a black-and-white monitor. The sample-and-hold circuit added some delay and distortion, but because pixels usually do not alternate black-and-white, but rather every two or more pixels are black, then two or more are white, etc., the signal has more time to settle to its correct value and will not be affected as much by the sample-and-hold. In any case, the final system will be modified so that the sample-and-hold circuit can be bypassed thereby reducing noise. This will be done by taking the output of the image sensor from different nodes which are actually continuous signals. Currently, the output is taken from nodes which have the correct signal value during the high period of the clock and are zero during the low part of the clock.

6. Final Notes

The current setup has two boards. A joystick board with digital components and an FPGA and interface board. Later, all parts will be placed on a single board. Integration and testing began by testing the individual circuits in a modular fashion. After wiring all necessary components, the FPGA outputs, sensor's socket, clocks, and interface circuits were thoroughly checked for correct signals. Several incorrect connections were discovered and corrected. Finally, the sensor was put in place and an image was displayed on the monitor.

The final image was obtained after many iterations of testing and debugging. Much digital work was done in modifying the design of the FPGA. (the control part of the system)

In terms of the MTL demonstration, the image sensor-AD830 interface connection has the correct characteristics and the AD830 interface-ADC board connection also works well. To best interface the image sensor and the A/D converter, both modules should be housed on the same board along with the interface. Currently, there is no way to do that because of the lack of resources. If the sensor, interface, and A/D converter are all on one board, no termination will be necessary and the following connections can be made. The differential analog output of the imager can be connected to the differential inputs of the interface. Since the imager's output range is 0 - 600 mV, a gain of 2.5 and offset of -750 mV will center the interface's output about 0 V and have an absolute value signal of 750 mV. This is within the input range of the A/D converter. (-0.9 V to +0.9 V) If a fuller range is desired, the gain and offset can be adjusted accordingly.

After this process, the MTL setup can acquire all the data and perform the desired functions in a digital manner. With the correct specifications, the imager from JPL and the MTL A/D converter system can be interfaced successfully.

The following tasks were researched or completed:

- Characterizing/Researching the A/D board, building it, testing it with the MTL setup
- Modifying C code and MATLAB scripts to perform the ADC testing
- Characterizing the image sensor
- Testing the display and providing the correct NTSC and video signals
- Designing, testing, and searching for an adequate video-speed differential amplifier
- Designing an appropriate sample-and-hold circuit in between the buffer and amplifier
- Digital design modifications of the FPGA
- Integrating the joystick, FPGA, and display (Camera demo) using the AD830 interface
- Integrating the camera demo system with the ADC board using the AD830 interface

References

1. Moreno, Lourdes C., "Interface between a joystick, ADC and camera APS chip using an FPGA," JPL, 1996.
2. Fossum, Eric, "CMOS Image Sensors: Electronic Camera On A Chip," pp.-7, IEEE International Electron Devices Meeting Technical Digest, 1995.
3. Peiris, Vincent, "Low Voltage Low Power CMOS Pipeline A/D Converter," internal MTL document/report for Swiss National Fund, 1995.
4. Sodini, Charles, "Ultra Low Power Wireless Sensors," internal MTL document, 1995.
5. Fossum, Eric, private communication, JPL, 1996.
6. Li, Jasen, "AD8011 Based Amplifier," internal JPL document, 1995.
7. Li, Jasen, "2nd Differential Amplifier," internal JPL document, 1995.

A1. FPGA Paper

Lourdes Moreno
University of Pennsylvania
Jet Propulsion Lab (7/96)

<u>Interface between a joystick, ADC and camera APS chip using an FPGA</u>

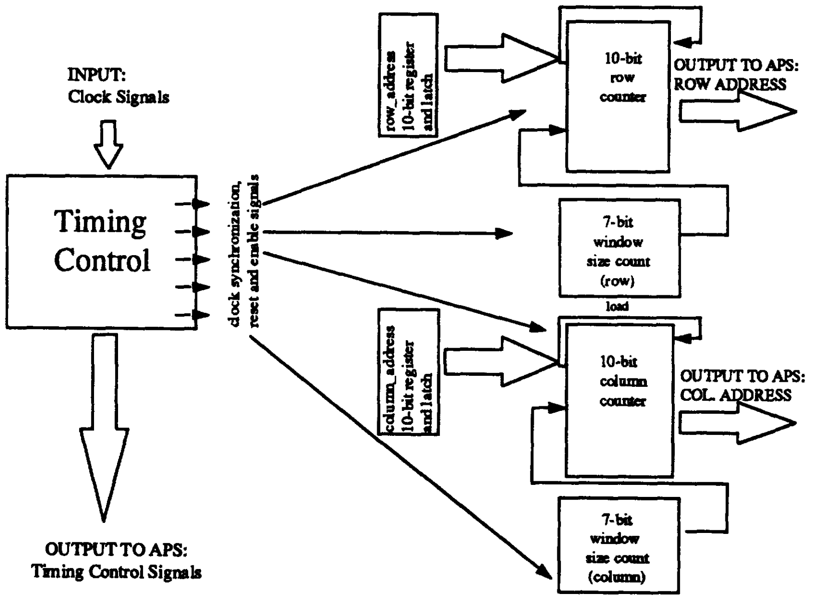
Inside the FPGA

The main objective of the FPGA is to provide the 1024 X 1024 APS with all the required timing and control signals for it to read an image and display it according to specific parameters using the NTSC format readout.

FPGA I/O Structure

The inputs for the FPGA consist of a 16MHz master clock and the digital starting address coming from the joystick interface. The row address as well as the column address are latched and stored in the corresponding 10-bit registers every 33 msecs. For each frame, these values are used as the initial address for readout. The frame rate is 30 frames/second.

The FPGA provides as its output the correct row and column address based on a specific window size.


INPUT:
Clock Signals
Timing Control
OUTPUT TO APS:
Timing Control Signals
clock synchronization, reset and enable signals
row_address 10-bit register and latch
10-bit row counter
OUTPUT TO APS: ROW ADDRESS
7-bit window size count (row)
load
column_address 10-bit register and latch
10-bit column counter
OUTPUT TO APS: COL. ADDRESS
7-bit window size count (column)

It is also designed to do a divide-by-four on the 16MHz master clock, providing the Hitachi EIA Pulse Generator with the required 4MHz NTSC clock. The FPGA also produces as its output all the timing control signals needed for the 1K x 1K image sensor:

- RESET
- SHR (Sample-and-Hold: reset value)
- PG (Photogate enable)
- SHS (Sample-and-Hold: signal value)
- ROW_EN (Row-count enable)
- COL_EN (Column-count enable)
- DDS(Double Delta Sampling)
- CLAMP

The FPGA timing signals are designed to operate the 1K x 1K APS chip in the photogate mode. A different FPGA design of the readout timing control logic has to be made in order to achieve photodiode operation. The future goal is to design the digital control circuits that would enable the user to choose between these two modes of operation for the APS.

Image Readout

"The architecture of the CMOS APS is typically designed to read out a row at a time, as selected by a decoder, with the reset and signal levels held on sampling capacitors at the bottom of the column. The column capacitors are selected by a decoder for buffered readout. Thus, the sensor can be read out in a sequential or nearly random access subsampled fashion, or by selecting a small window for readout to enable electronic zoom." (E.Fossum, "CMOS Image Sensors: Electronic Camera On A Chip", p.5)

Four counters are used in order to achieve the desired window size reading starting from the initial address specified through the joystick interface. These counters are designed to synchronously output the correct rows and columns to be read. They are all clocked by the 16MHz master clock input and their enable pin is controlled by the digital logic designed for the APS timing signals.

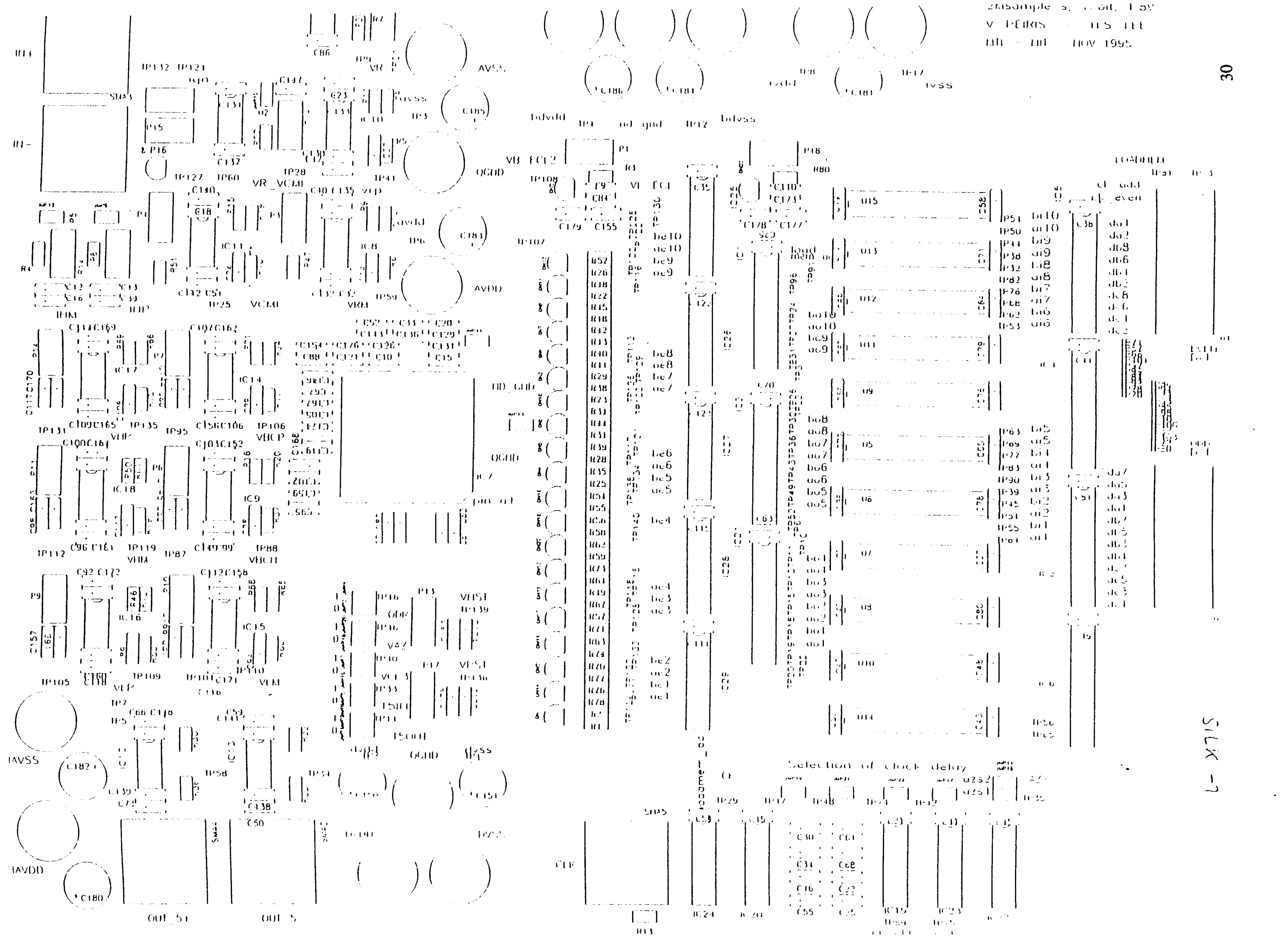
The following window size counters make it possible to translate a 100 x 100 window in the imager to a 200 x 100 display window in the TV monitor:

- A 7-bit column counter counting up to 100
- A 7-bit row counter double counting up to 100

Two 10-bit counters are designed to be loaded with the corresponding row and column addresses coming from the joystick through buses, and to count up until the next loading signal is received. These counters provide the APS with the (row,col) address it expects for reading the image. The 10-bit row counter is designed to double count in order for it to be in sync with the 7-bit row counter.

References

[1] E.Fossum, "CMOS Image Sensors: Electronic Camera On A Chip", pp.1-7.


2Msample/s, [illegible]bit, 1.5V
V. PEIRIS - ITS TEE
[illegible] - [illegible] NOV 1995
SILK-1
IN+
IN-
SMA5
AVSS
AVDD
QGND
VB_ECL2
VI_ECL
VR_VCMI
VCMI
VRM
IC7
HD_GND
QGND
bdvdd
bdvss
hd_gnd
bdvss
load
LOADHIT
clk_odd
clk_even
VHP
VHM
VBCP
VBCN
VLP
VLM
VHST
VPST
VAZ
IAVSS
BAVDD
OUT_51
OUT_5
Selection of clock delay
CLK
dvdd
dvss
GGND
IC24
IC20
IC19
IC23
R13

A2. Data Acquisition Code (vp_acq1.c acquires the raw digital codes)

```
/* Program for Pipeline ADC Data Acquisition System:         */
/* Basic acquisition of a block of data                    */
/*                                               */
/* Vincent Peiris                              */
/* (inspired from Andy's code)                     */
/* Microsystems Technology Laboratory                  */
/* Nov. 27th, 1995                              */

#include <stdio.h>
#include <conio.h>
#include <malloc.h>
/* #include <dos.h>  */

/* base address for PXB-721 is 0x300                  */
/* 8255A_3 is presently used                       */
/* PORT_D3 is control register                      */
#define PORT_A3 0x308
#define PORT_B3 0x309
#define PORT_C3 0x30A
#define PORT_D3 0x30B

/* mode 0x82 configures PORT_A3 as output, PORT_B3 as input   */
/* and PORT_C3 as output                           */
#define MODE 0x82

/* set all instruction bits to 0 for full reset             */
#define RESET 0x00

/* instructions to read SRAM DATA_C, DATA_B, DATA_A          */
#define DATA_C 0x42
#define DATA_B 0x22
#define DATA_A 0x02

/* data acquisition system instruction format
1   LOADMEM
2   REQREAD
3   REQWRITE
4   REQCOUNTMAR
5   REQCLRMAR
6   X0
7   X1
8   NC
*/
/* data length read from SRAM (size of a block)          */
/* 2^10 for a 10bit resolution. Can go upto 2^17 samples    */
/* which fills up the SRAM                         */
#define DATA_MAX 1100

/* margin to flush out a given number of initial samples    */
#define MARGIN_START 20
```

```
/* margin required for time align to prevent array over-run */
/* (concerns only last samples in an entire block          */
#define MARGIN_END 10

/***  MAIN ROUTINE                              ***/
/* Consisting in 4 parts                        */
/* Written as a monoblock, sorry about that!          */

main()
{
FILE *stream;
long i, j, k;
int *a, *b, *c, *d;
int a1, a2, a3, a4, a5, a6, a7, a8, a9, a10;
int b1, b2, b3, b4, b5, b6, b7, b8, b9, b10;
int a1b1, a2b2, a3b3, a4b4, a5b5, a6b6, a7b7, a8b8, a9b9, a10b10;
int temp1, temp2, temp3, temp4, temp5;
a = calloc(DATA_MAX, sizeof(int));
b = calloc(DATA_MAX, sizeof(int));
c = calloc(DATA_MAX, sizeof(int));
d = calloc(DATA_MAX, sizeof(int));

/*** PART 1: STARTING ACQUISITION AND CONVERSION PROCESS ***/
/* When ADC card receives signal REQCOUNTMAR, the          */
/* conversion process starts and data is stored in         */
/* SRAM's on the Acquisition card                    */

/* set up PXB-721 mode */
outp(PORT_D3, MODE);

/* acquire data */

outp(PORT_A3, 0x10);    /* clear MAR */
outp(PORT_A3, RESET);   /* reset */
outp(PORT_A3, 0x01);    /* set LOADMEM high */
outp(PORT_A3, 0x09);    /* acquire data with LOADMEM */
                        /* and REQCOUNTMAR high      */

for(i = 0; i <= 250000; i++)
;
/* wait a while till the A/D has finished converting  */
/* a block. Might want to experiment ant increase or  */
/* decrease this depending on the time it effectively */
/* takes for the ADC to convert an entire block       */

/*** PART 2: LOADING CONTENTS OF SRAMS INTO ARRAYS A,B,C ***/
/* Once the conversion process is finished, an entire      */
/* block of data is download from the SRAM's into the      */
/* computers for further processing                  */

outp(PORT_A3, 0x10);      /* clear MAR */

for(j = 0; j <= (MARGIN_START+DATA_MAX+MARGIN_END-1); j++)
/* reading sufficient data to have a DATA_MAX size block,  */
/* once the unwanted MARGIN_START and MARGIN_END data will */
/* have been flushed away                        */
{

outp(PORT_A3, DATA_C); /* read out DATA_C from current MAR address */
c[j] = inp(PORT_B3);  /* get DATA_C */
outp(PORT_A3, DATA_B); /* read out DATA_B from current MAR address */
b[j] = inp(PORT_B3);  /* get DATA_B */
outp(PORT_A3, DATA_A); /* read out DATA_A from current MAR address */
a[j] = inp(PORT_B3);  /* get DATA_A */
outp(PORT_A3, RESET);  /* reset */
outp(PORT_A3, 0x08);  /* increment MAR */
}

outp(PORT_A3, RESET);  /* reset */

/*** PART 3: COMPUTATION OF EFFECTIVE BINARY CODE       ***/
/* The brute data must be rearranged in order to get      */
/* The effective output word in standard binary code.     */
/* This implies time-aligning data (bits are issued       */
/* sequentially in a pipeline A/D), and then error        */
/* correction (convert from 2 bit redundant notation      */
/* to standard 1 bit binary notation)                */

for(j = MARGIN_START; j <= (MARGIN_START+DATA_MAX-1); j++)
/* j is the index for arrays a, b, c                */
/* k is the index for array d                      */
/* k is j - MARGIN_START, thus starts from 0, and only    */
/* DATA_MAX codes are processed into d                 */
{
k = j - MARGIN_START;

/* Time alignement in prevision of further error correction*/
/* In the following comments are represented:             */
/* - the position of ai or bi in either SRAM a,b, or c.   */
/* - the final disposition in the 11bit word before       */
/*   error correction.                        */
/* Note that character . stands for 0. The indexes        */
/* i in ai,bi are also ommited for simplicity          */

/* The idea is to isolate the proper bit using masking,   */
/* then to shift the bit to its desired location. in      */
/* prevision of the error correction algorithm.          */
/* Note that the bits are inverted due to inversion in    */
/* the pseudo ECL stage                         */

/* from SRAM c */
a1 = (c[j] & 0x01)   << 10;   /* .... ...a  ->  a.. .... .... */
b1 = (c[j] & 0x04)   << 7;    /* .... .b..  ->  .b. .... .... */
a2 = (c[j] & 0x02)   << 8;    /* .... ..a.  ->  .a. .... .... */
b2 = (c[j] & 0x08)   << 5;    /* .... b...  ->  ..b .... .... */
a3 = (c[j+1] & 0x10) << 4;    /* ...a ....  ->  ..a .... .... */
b3 = (c[j+1] & 0x40) << 1;    /* .b.. ....  ->  ... b... .... */
a4 = (c[j+1] & 0x20) << 2;    /* ..a. ....  ->  ... a... .... */
b4 = (c[j+1] & 0x80) >> 1;   /* b... ....  ->  ... .b.. .... */
/* from SRAM b */
a5 = (b[j+2] & 0x01) << 6;   /* .... ...a  ->  ... .a.. .... */
b5 = (b[j+2] & 0x04) << 3;   /* .... .b..  ->  ... ..b. .... */
a6 = (b[j+2] & 0x02) << 4;   /* .... ..a.  ->  ... ..a. .... */
b6 = (b[j+2] & 0x08) << 1;   /* .... b...  ->  ... ...b .... */
a7 = (b[j+3] & 0x10)    ;   /* ...a ....  ->  ... ...a .... */
b7 = (b[j+3] & 0x40) >> 3;   /* .b.. ....  ->  ... .... b... */
a8 = (b[j+3] & 0x20) >> 2;   /* ..a. ....  ->  ... .... a... */
b8 = (b[j+3] & 0x80) >> 5;   /* b... ....  ->  ... .... .b.. */
/* from SRAM a */
a9 = (a[j+4] & 0x01) << 2;   /* .... ...a  ->  ... .... .a.. */
b9 = (a[j+4] & 0x04) >> 1;   /* .... .b..  ->  ... .... ..b. */
a10= (a[j+4] & 0x02)    ;   /* .... ..a.  ->  ... .... ..a. */
b10= (a[j+4] & 0x08) >> 3;   /* .... b...  ->  ... .... ...b */
/* generation of a word containing both bits ai,bi */
/* at the proper location. This is to prepare the  */
/* for digital error correction                   */
a1b1 = a1 | b1;          /*         ->  ab. .... .... */
a2b2 = a2 | b2;          /*         ->  .ab .... .... */
a3b3 = a3 | b3;          /*         ->  ..a b... .... */
a4b4 = a4 | b4;          /*         ->  ... ab.. .... */
a5b5 = a5 | b5;          /*         ->  ... .ab. .... */
a6b6 = a6 | b6;          /*         ->  ... ..ab .... */
a7b7 = a7 | b7;          /*         ->  ... ...a b... */
a8b8 = a8 | b8;          /*         ->  ... .... ab.. */
a9b9 = a9 | b9;          /*         ->  ... .... .ab. */
a10b10 = a10 | b10;        /*         ->  ... .... ..ab */

/* error correction: sum of all ai,bi words, then   */
/* elimination of LSB (the 11th bit) in order to    */
/* obtain the effective 10bit binary code           */

temp1 = a1b1+a2b2+a3b3+a4b4+a5b5+a6b6+a7b7+a8b8+a9b9+a10b10;
temp2 = temp1 >> 1;
d[k] = temp2;
/* d[k] = (a1b1+a2b2+a3b3+a4b4+a5b5+a6b6+a7b7+a8b8+a9b9+a10b10) >> 1; */
} /* end for(j = ...) */

/*** PART 4: WRITING THE OUTPUT FILE                ***/
/* An data output file is written for further processing     */
/* with other softwares (MATLAB,...)                */

/* write data out to file */
stream = fopen("DATA.ADC", "w+");

for(j = 0; j <= (DATA_MAX - 1); j++)
{
fprintf(stream, "%d\n", d[j]);
}
/* close file */
fclose(stream);
return 0;
} /*** END OF MAIN ROUTINE ***/
```

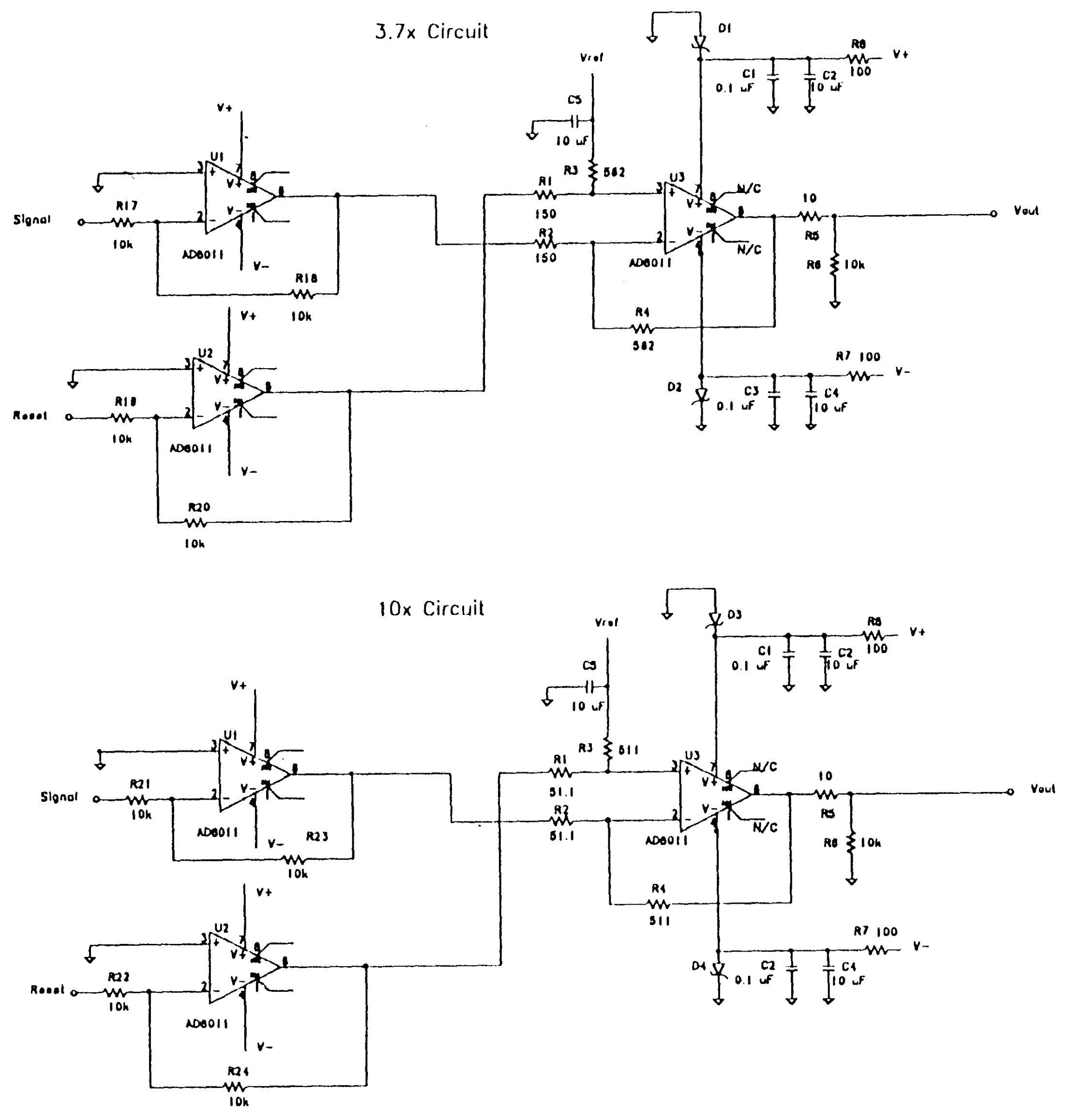
A3. Data Analysis Code (code.c takes the digital codes and transforms the data into differential non-linearity and integral non-linearity data)

```
/* Program to make DNL/INL                                    */
/* Written by Jungwook Yang. modified by David Martin      */
/* Microsystems Technology Laboratory                  */
/* 02-22-96                                  */
/* modified by Jasen Li 03-21-96                   */

#include <stdio.h>
#include <malloc.h>
#include <math.h>

#define MAXCODE  1024                        /* maximum # of codes, 2^10 */
#define PI 3.1415926

main(argc, argv)
int argc;
char *argv[];
{
 long *histogram;
 long *cumulative_histogram;
 double *voltage;
 double *dnl, *inl;

 FILE *file;
 int i;
 long j,tmp;
 double temp, mean, maximum_dnl;
 int maximum, minimum;
 char filename[256];

 /* Get filename */

 strcpy (filename, "data");
 strcat(filename, argv[1]);
 strcat(filename, ".dat");

 /* Get data from data.dat and store it in histogram */

 printf("Reading data file.\n");
 histogram = (long *) calloc(MAXCODE, sizeof(long));
 for (i=0; i < MAXCODE; ++i) {
  histogram[i] =0;
 }
 file = (FILE *) fopen(filename, "r");
 j =0;
 while (fscanf(file, "%d", &tmp) != EOF) {
  j++;
  tmp = tmp >> 1;
  if (tmp < 0) {
   printf("Data less than zero.\n");
  } else if (tmp > MAXCODE -1) {
      printf("Data too big.\n");
    } else {
      histogram[tmp] = histogram[tmp] + 1;
    }
  }
  fclose(file);

  for (i = 0; i < MAXCODE; ++i) {
    printf("%d %d\n", i, histogram[i]);
  }
  /* Get maximum and minimum */

  printf("Getting maximum and minimum.\n");
  for (i=MAXCODE -1; i >=0; --i) {
    if (histogram[i] != 0) {
      minimum = i;
    }
  }
  for (i =0; i < MAXCODE; ++i) {
    if (histogram[i] != 0) {
      maximum = i;
    }
  }
  /*  maximum = maximum -1; */

  /* Make cumulative histogram */

  printf("Creating Cumulative Histogram.\n");
  tmp =0;
  cumulative_histogram = (long *) calloc(MAXCODE, sizeof(long));
  for (i=minimum; i <= maximum; ++i) {
    tmp = tmp + histogram[i];
    cumulative_histogram[i] = tmp;
  }

  /* Get Voltage Transitions */

  printf("Calculating Voltage Transitions.\n");
  voltage = (double *) calloc(MAXCODE, sizeof(double));
  for (i=minimum; i <= maximum; ++i) {
    voltage[i]=-1.0 * cos(PI * (double)cumulative_histogram[i] / (double) j);
  }
  /* Calculating DNL */

  printf("Calculating DNL.\n");
  temp =0;
  dnl = (double *) calloc(MAXCODE, sizeof(double));
  for (i = minimum; i < maximum; ++i) {
    dnl[i]=voltage[i+1]-voltage[i]; /* check this */
    temp = temp + dnl[i];
    }
  mean=temp/ (double)(maximum-minimum);
  for (i=minimum; i < maximum; ++i) {
    dnl[i]=dnl[i]/mean - 1.0;
```

```
}
maximum_dnl =0.0;
for (i = minimum; i < maximum; ++i) {
 if ((dnl[i] * dnl[i]) > maximum_dnl) {
   maximum_dnl = dnl[i] * dnl[i];
  }
}

/* Calculating INL */

printf("Calculating INL.\n");

inl = (double *) calloc(MAXCODE, sizeof(double));
inl[minimum] = dnl[minimum];
for (i = minimum +1; i < maximum; ++i) {
        inl[i] = inl[i-1] - dnl[i];
}

/* Storing DNL to a file */

printf("Storing DNL.\n");
file = (FILE *) fopen("addnl.dat", "w");
for (i=minimum; i < maximum; ++i) {
  fprintf(file, "%d %f\n", i, dnl[i]);
}
fclose(file);

/* Storing INL to a file */

printf("Storing INL.\n");
file = (FILE *) fopen("adinl.dat", "w");
for (i=minimum; i < maximum; ++i) {
  fprintf(file, "%d %f\n", i, inl[i]);
}
fclose(file);

/* Give final results */

printf("Maximum code was %d:\n", maximum);
printf("Minimum code was %d:\n", minimum);
printf("Maximum DNL was %f:\n", sqrt(maximum_dnl));

}
```

A4. Data Display Code (data.m takes the DNL and INL data sets and plots them using MATLAB software)

```
ACTION='pause';
%ACTION='print adidnl.ps -a';

echo on
%%
%% David A. Martin, 2/23/96
%%
%% Matlab script file for data plotting
%%
%% DNL Data
%%


hold off
load addnl.dat

orient tall;
subplot(211);
plot(addnl(:,1), addnl(:,2))
hold on

axis([0 1024 -.65 .25]);    % Scale axis
axis('square');
%title('A/D DNL Data, Chip #C1, 5.5MHz'); % Label plot
xlabel('OUTPUT CODE');                           % Label x-axis
ylabel('DNL LSB');                       % Label y-axis
pause

hold off
load adinl.dat

subplot(212);
plot(adinl(:,1), adinl(:,2))
hold on

axis([0 1024 -.6 1.25]);    % Scale axis
axis('square');
%title('A/D INL Data, Chip #C1, 5.5MHZ'); % Label plot
xlabel('OUTPUT CODE');                           % Label x-axis
ylabel('INL LSB');                       % Label y-axis
```

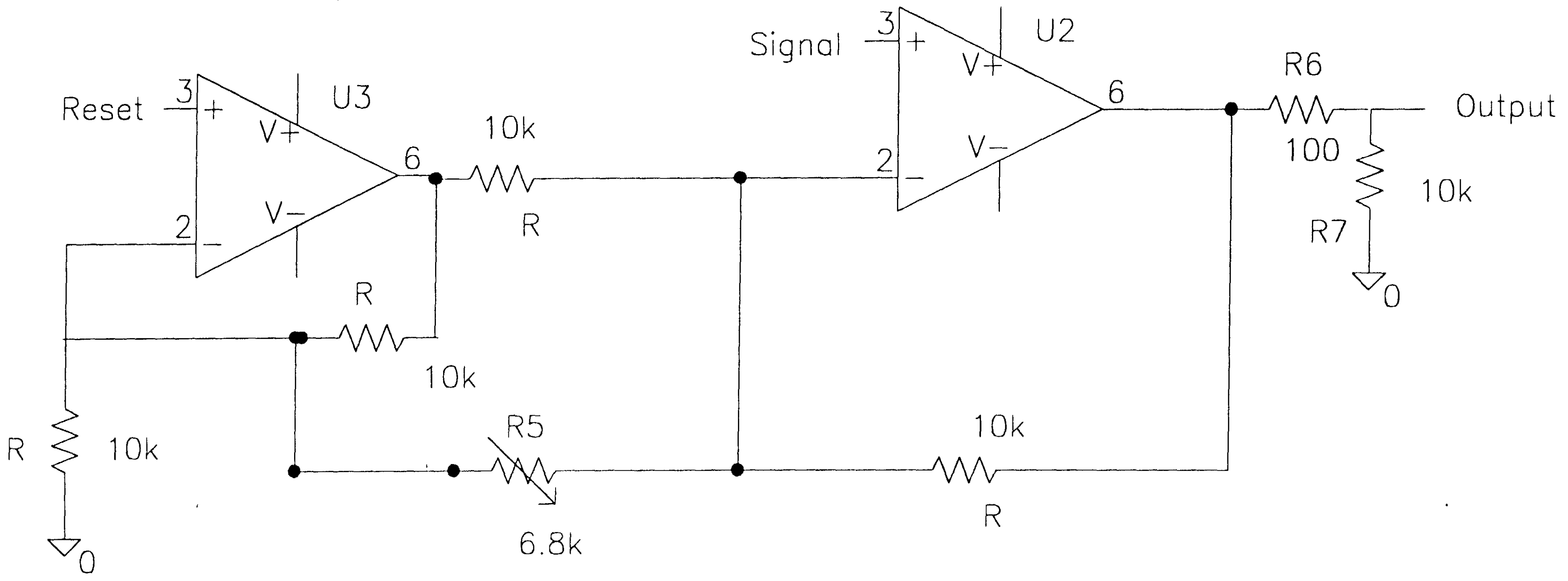


Notes:

All resistors are 1/8 W 5% precision
Opamps are AD8011
C2 and C4 are electrolytic
C5 is not to be electrolytic
D1 thru D4 are 5.6 V

Jet Propulsion Laboratory 4800 Oak Grove Dr Pasadena, CA 91109 Craig Staller	Page Size: B
CC258 Opamp	
Revision: C	5 June 1995 Page 1 of 1

AD8011 Amplifier Circuit



$$V_{out} = 2 * (1 + R / R5) * (Signal - Reset)$$