An Activity Detection System for Frequency-Encoded Pixels

by

Victor C. Lum

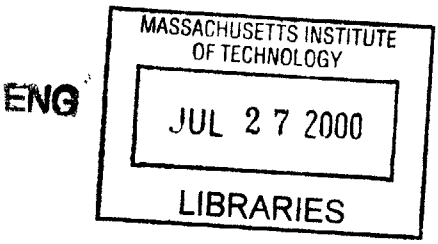
Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 5, 2000



ENG

Author .....
Department of Electrical Engineering and Computer Science
May 5, 2000

Certified by .....
Lisa McIlrath
Visiting Professor, MIT AI Laboratory
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

An Activity Detection System for Frequency-Encoded Pixels

by

Victor C. Lum

Submitted to the Department of Electrical Engineering and Computer Science
on May 5, 2000, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Electrical Engineering and Computer Science

Abstract

It is possible, using a first-order sigma-delta modulation scheme, to represent an imager's pixel value as an oversampled bit stream. The goal of the project was to demonstrate the feasibility of computer vision algorithms on such sigma-delta encoded bit streams. The computational modules used were constrained to fit within the footprint of a single pixel. This ensures that fully parallel operation could be achieved using a 3D architecture. 3D silicon technology allows multiple wafers to be stacked together and interconnects to be made between adjacent layers. Bonding wafers together in this manner would allow computational modules to be placed directly beneath its associated pixel. This process increases the potential for high parallelism and low power computation. The activity detection algorithm which was developed incorporates a frequency-locking analog storage mechanism with a lowpass filter/monitor to simulate a multimodal adaptive background activity detector. This system has been simulated in software and has proven itself capable of detecting activity with a fixed field of view. The simulation was developed for the TI TMS320C40 DSP and later extended to an x86 architecture for faster performance and real time evaluation. The GoDSP^{TM} Code Composer IDE was used to develop the DSP software while running it on a White Mountain DSP Slalom-40 board. The x86 implementation was written using MVC++-4.0 running on a 450 Mhz Pentium II based Gateway GP6-450.

Thesis Supervisor: Lisa McIlrath
Title: Visiting Professor, MIT AI Laboratory

Acknowledgments

I would like to thank Lisa McIlrath for her guidance; Chris Stauffer for taking the time to patiently explain his models; Sam Lefian for his hardware advice; Akin Aida for distracting me and occasionally pointing out problems; Chuck Sawabini for his support on the test system; and for the funding provided by Defence Advanced Research Projects Agency under contract DAAK60-96-K-0204, "3D Computational Image Sensors for Advanced Low Power Visual Processing."

Contents

List of Figures

Chapter 1

Introduction

Sigma-delta modulators, also known as oversampling A/D converters, generate a low resolution—typically 1-bit—output by sampling the analog input at many times the Nyquist rate. A high resolution N-bit output is obtained by filtering and down-sampling the bit stream to the Nyquist rate. A sigma-delta modulation scheme has two primary benefits. First, it reduces the number of wires required to extract the data. A single wire for each pixel is sufficient, as opposed to 8 or more for a direct A/D conversion. Second, the dynamic range and resolution of the sigma-delta conversion have the theoretical potential to be orders of magnitude greater than non-oversampling ADCs.

A new multilayer silicon process has been developed in a collaborative effort between MIT Lincoln Labs and Northeastern University. Their process enables several wafers, fabricated using existing technologies, to be bonded together and electrical contacts to be made between adjacent layers. The addition of a third dimension to silicon design allows for the introduction of new types of computational architectures.

The two technologies, frequency-encoded pixels and 3D architecture, have been combined in a CMOS image sensor designed by McIlrath at the MIT Artificial Intelligence Laboratory[1]. The imager is composed of a surface layer of densely packed photodiodes. Directly beneath each of these diodes is circuitry which encodes the current outputs into digital bit streams. The purpose of this project was to explore potential 3D microarchitectures which could be used to manipulate these bit streams.

The constraints of the problem require that computational modules are either small to fit within the footprint of a pixel, or modular to be divided amongst several layers. Basic modules were sought which could perform elementary functions, such as addition, subtraction, and memorization. Once these modules were designed, they were combined to emulate the behavior of an activity detection system designed by Stauffer[4]. This adaptive algorithm is modular enough to be broken up and placed on several layers, parallel enough for an identical computational module to be assigned to every pixel, and sufficiently complicated to use several of the basic computing modules.

For the simulations, an existing 64x64 imager was connected to a PC-based simulation environment. The imager consisted of an array of interleaved photodiodes and oversampling modulators which generated bit streams for each of the pixels. These bit streams were then transmitted to the simulation environment using an existing test system. The activity detection simulation was implemented in a C-based package running first on a DSP and later on an 0x86 microprocessor.

Chapter Two will discuss the first basic area of research being demonstrated, 3D silicon systems and sigma-delta modulation of pixel values. It describes the 3D silicon technology for which these algorithms have been targeted. The new process method allows for much more freedom than conventional two dimensional integrated circuits. Some of the major benefits and potential applications are discussed.

Chapter Three introduces the existing sigma-delta modulator and some of the basic properties of its output. The modulator uses an asynchronous sampling method rather than a more standard synchronous one. The outputs of these two approaches are identical, but the asynchronous implementation was found to be smaller. A brief explanation of potential methods for demodulating the signal are described, and small modules capable of adding, subtracting, and storing the bit streams are discussed.

Chapter Four describes Stauffer's research into building multimodal activity detection systems. The model he uses is adapted to an architecture which can be implemented efficiently using the 3D silicon process. This adaptive model is discussed in detail, focusing on its theoretical performance potential. Some of the tradeoffs which

were made to meet the constraints of the design objectives are discussed.

Chapter Five contains the results of the simulations. The test system was used to monitor activity over the course of several hours and store the results. These results are analyzed and evaluated for accuracy and efficiency. The paper concludes with a discussion of future work for developing these technologies.

Appendix A contains information on the software models used to simulate the behavior of the physical elements.

Chapter 2

Architectures for 3D Computational Imagers

As designers reach the limits of modern process technology, they must search for the next step in the process evolution. One potential technology to fill this role is three dimensional integrated circuit fabrication. In this process, fully fabricated SOI wafers are bonded together to form a multilayer silicon "sandwich". Electrical connections are then made between adjacent layers.

The immediately apparent benefits of 3D technology is the increase in available bandwidth between circuit blocks which reduces the necessity for signals to be routed off-chip. A modern processor has at most hundreds of pins to communicate with the rest of the computer system. In a multilayer design, it is possible to eliminate this restriction by moving large sections of the system onto other layers of the same 3D chip. Figure 2-1 illustrates the potential savings which can be achieved by building a complete system in a single multilayer IC.

Process yields in modern systems are inversely proportional to the size of the target circuit. As the size of 2D arrays increase, so does the probability of a flaw appearing in the circuit . The increased size is necessary in the 2D approach simply to fit all of the design hcomponents onto a single die. By segmenting that design and placing each section on a separate layer, the footprint for each section is consequently smaller, reducing the individual chances for a process flaw to occur. By designing

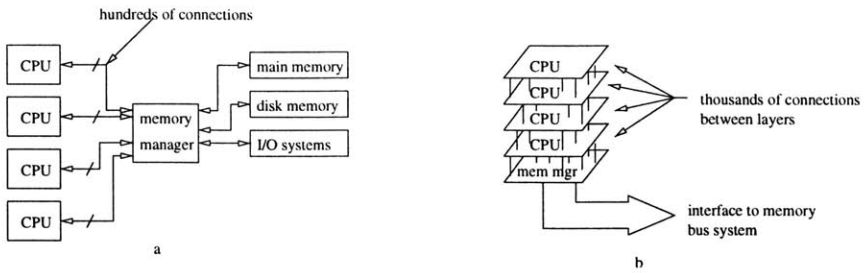


Figure 2-1: Comparison of off chip routing for modern multiprocessor system (a) and for an equivalent multiprocessor, multilayer design(b).

an intelligent parallel architecture rather than conventional serial approaches, it may even be possible to adapt to such flaws by bypassing non-functional sections of the circuit.

The high degree of parallelism afforded by this multilayer technology results in lower power consumption for equal computation. Since power is directly proportional to the clock frequency ($P = CV^2f$), there is a one-to-one power benefit for any reduction in frequency. A highly parallel system would be able to compute the same number of operations as a contemporary serial processor, but at a significantly lower operational frequency. By replacing high-capacitance inter-chip wires with low-capacitance interlayer vias, the overall power savings can be dramatic as both frequency and capacitance may be reduced.

The efficiency of highly parallel 3D systems has been well documented in the field of biology. Organic brains are massive computational systems capable of performing billions of operations every second. Not only can our brains outperform any modern computer, it is also orders of magnitude more power efficient. A human brain can perform 3×10^{13}operations per Joule - far more than any current microprocessor. The DEC Alpha21164, which is typical of current microprocessors has an average efficiency of 6.25×10^6 operations per Joule[3]. With this technology it becomes possible to more closely mimic, physically, the architecture of an organic brain. Neuromorphic engineering is just one of the possible areas of study which can benefit from the multilayer structure. Computational node-based algorithms can be more

easily and compactly implemented with 3D silicon. Kilobyte-word microprocessors, pixel-parallel computational imagers, high density memories, and systems-on-a-chip technologies are just a few of the many areas which stand to benefit from this advance in technology.

2.1 Conventional Silicon

Integrated circuits are currently made by placing and interconnecting transistors in a planar array. To layout the various regions for doping, the locations of polysilicon, and the traces of interconnecting metal, the process uses a set of masks for each of the layers. Here *layers* refers to the process layers such as the polysilicon, metal 1, metal 2, etc, which are the silicon equivalents of signal planes in a PCB. Through a series of etching and growing steps, a 2D wafer is constructed using these masks to control the growth. Recent advances in process technology have increased the realizable size of a single die, while simultaneously reducing the minimum width of a single transistor.

Both of these improvements have involved a significant cost for the developer. New fabrication facilities with more sophisticated machinery has been required for every improvement in transistor size. Increasing the available die size was only made possible by creating even cleaner environments—reducing the chances for process-based contamination. The 3D process promises to improve device performance without the overhead cost of new facilities and equipment. It accomplishes this by using existing technologies to fabricate the pre-bonded wafers.

Even with larger die sizes and smaller transistors, many heavily parallel systems could not be easily implemented on a single chip, or even on multiple chips. The main issue in such systems is the complexity and quantity of the interconnects required to assure communication between computational modules. Consider a four-dimensional hypercube arrangement of nodes. While it is possible to project the cube onto a two dimensional surface, the interconnect complexity increases the size considerably over the equivalent three dimensional projection. Both can be see in figure 2-2.

If a sufficiently large numbers of nodes is required, the 2D design would exceed the

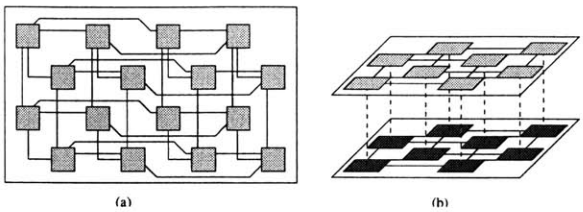


Figure 2-2: Layout of a 2x2x2x2 hypercube in (a) two dimensions and (b) three dimensions.

bounds of the size of a single die. The 3D design could be expanded to virtually an unlimited number of nodes. Since each of the layers is identical, the mask sets can be reused, reducing both development time and costs. Other geometric patterns, such as simple cubic, face-centered cubic, and hexagonal close-packed would be much easier to realize in 3D than in 2D. In all of these formations, more nodes can be placed in in the same amount of silicon area because the interconnect cost is much lower.

2.2 Creating 3D Silicon

A process, originally developed in collaboration between Northeastern University and MIT Lincoln Labs, has been developed to bond two wafers together and connect them electrically. Circuits are fabricated on SOI wafers, using a standard process. The wafers are bonded together and the handle substrate of one is removed. Vias are patterned through the back side of the remaining thin film circuit layer and filled with metal to make the electrical connection. These vias can be placed in any arrangement on the circuit. A cross section of possible configurations of a three layer image sensor can be seen in figure 2-3.

This bonding process can be repeated several times, allowing several circuit layers to be connected in complex systems. There is no theoretical limit to the number of layers which can be added. Furthermore, since the process uses standard fabrication technologies, current manufacturing facilities can still be used to produce the individual wafers. The only new equipment that is required are the tools for the final bonding step of the process.

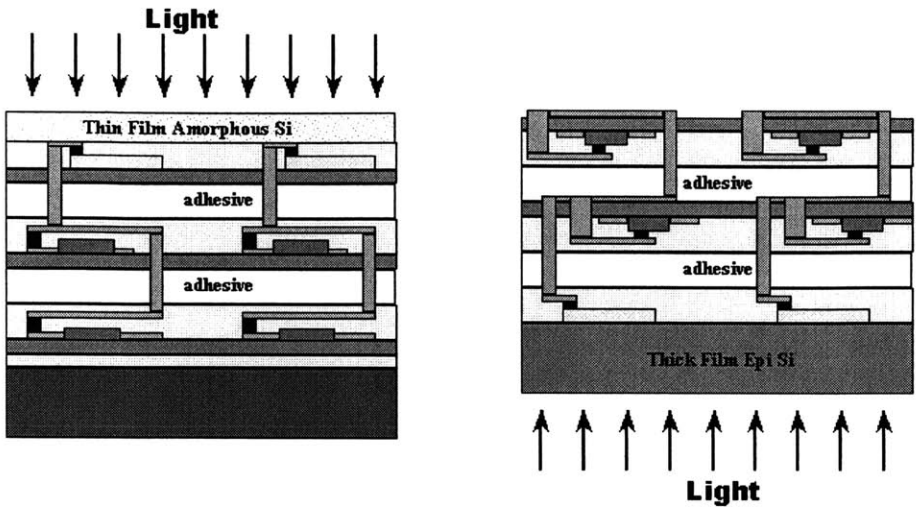


Figure 2-3: Cross section of a 3D wafer.

2.3 Imagers in 3D Silicon

Smart imaging technology can benefit from 3D processing. Contemporary imagers can be subdivided into three categories; serial output, pixel-parallel output, and column-parallel output systems. Each of these systems has its own benefits, but also a set of performance issues which are inherent in their design.

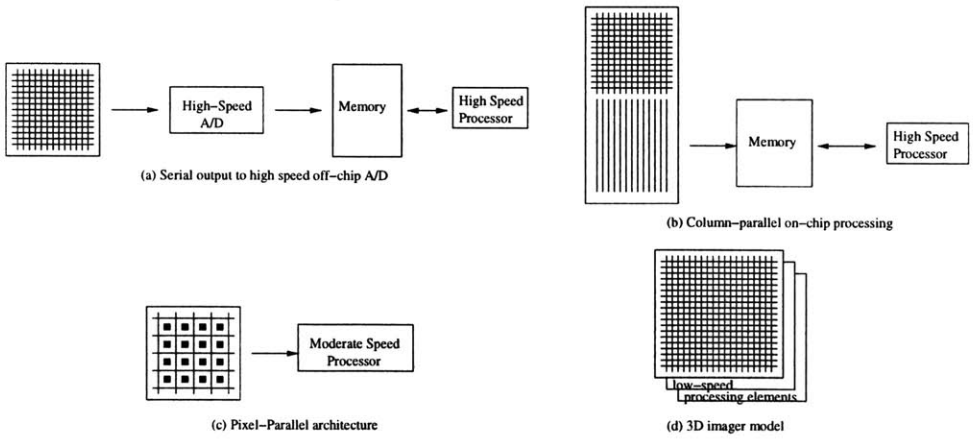


Figure 2-4: Comparison of the three standard imagers and the proposed 3D imager.

A serial output imager generates pixel values in a high-speed raster-based format.

These values are then digitized using an analog/digital converter. Though the interface to this type of imager is fairly simple, it suffers from a severe data bottleneck. As imager sizes increase, the system must work proportionally faster to meet the same frame rate requirements - resulting in higher power consumption. The total size of the imager is then limited by the speed of the A/D converter and desired frame rate. The limiting relationship specifies that the [number of pixels] = [maximum conversion rate]/[required frame rate].

A pixel-parallel design places a single processing element adjacent to every photodiode. These small elements are responsible for interpreting the values of the photodiode and responding accordingly. There is no limit to the size of such a pixel array, and the speed is limited only by the speed of the slowest processing element. However, a pixel-parallel system suffers from a low fill factor - i.e. the relative area in the array devoted to light sensing by pixels is small due to the area required for the processing elements adjacent to each pixel. Many image processing algorithms assume that that the pixels are close to one another. As the distance between physical pixels increases, the validity of this assumption, and the models based on it, break down. Conversely, large processing elements are typically required to perform most useful operations.

A column-parallel architecture is a combination of these two designs. It associates all of the pixels in a single column with a dedicated A/D converter. Using this approach, the fill factor is higher than the pixel-parallel system, and the processing speed is lower than for the serial output system. The column-parallel approach has become the defacto standard for high speed high-resolution imagers because it provides such a good combination of speed, performance, and power.

A 3D architecture can improve performance and increase parallelism. By moving all of the wiring and processing elements to lower levels of the design, a 3D pixel array would have a very high fill factor. Because of the small pixel footprint in high resolution images the savings in space is dramatic. In the 3D system, the surface layer would be entirely devoted to light sensing.

The challenge in designing 3D architectures is in developing area-efficient computational systems. All of the modules which are simulated must fit within the footprint

of a single pixel. If this is not the case, a fully parallel system would not be possible. The small size of the pixel forces designers to look beyond conventional digital systems towards bit stream encoding technologies and analog-digital hybrid circuits. One of the luxuries afforded by the heavy parallelism is the relatively low frequency at which each module can operate. Using sigma-delta modulation it is possible to sacrifice a slight increase in frequency for a reduction in implementation size. This is only one of many methods for designing area-efficient circuits. The realm of 3D architectures remains largely unexplored and certainly other clever designs are yet to be discovered.

Chapter 3

Sigma-Delta Modulation

Sigma-delta modulation is widely used in the DSP field to digitize analog signals. It is an oversampling method which trades sampling speed for system complexity. The small circuits needed for low-order sigma-delta systems are ideally suited to the space constraints of the target 3D architecture. Sigma-delta modulation of the photodiode current accomplishes digitization of the signal, but without the expense of multiple interconnects. While a typical bit-parallel digitization converts the analog value to 8 or more bits of data, the output of the sigma-delta modulator can be a series of single bits. The resulting benefit of the modulation scheme over the analog sampling is the reduction in the number of wires required for transmitting the output to the next processing layer. The desire for high resolution imagers dictates the need for smaller pixel sizes. A 1 cm^2 die is considered a large size using modern processes. Reduced yields make larger designs uneconomical. If a 256x256 array is desired, each pixel will have a footprint less than 40μm on a side. In a larger 1024x1024 configuration, pixels are limited to 10μm per side.

The pixel design upon which the proposed 3D architecture is based is an asynchronous sampled oscillator whose frequency is proportional to the incident light intensity. It can be shown that the output of a sampled oscillator is equivalent to that of a first-order sigma-delta modulator, and hence the input signal can be determined by analyzing the output bit patterns.

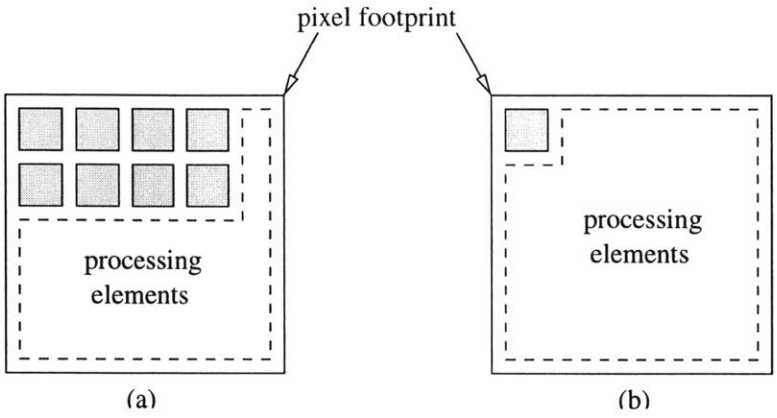


Figure 3-1: Comparison of processing space for (a) an 8-bit A/D system and (b) an oversampled modulator.

3.1 Synchronous Modulation

The minimum frequency at which a signal can be sampled without loss of information is known as the Nyquist sampling frequency. Typically, this sampling rate is twice the highest frequency component of the continuous signal. At the Nyquist rate, there is neither any aliasing of high-band signals, nor is any frequency band unused. The diagram in figure 3-2 depicts a signal with a maximum frequency component of f_0 sampled at, below, and above the Nyquist frequency.

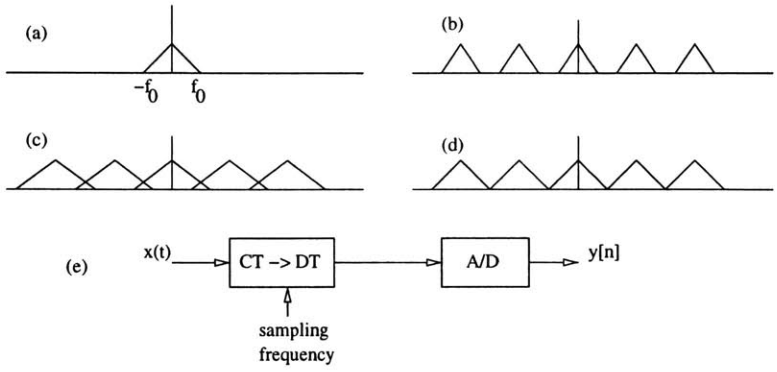


Figure 3-2: The signal, (a), sampled below the Nyquist rate (b), above the Nyquist rate(c), and at the Nyquist rate (d). (e) diagram for the sampling circuitry.

A signal which is undersampled is sampled below the Nyquist frequency (c). An

undersampled signal encounters aliasing problems where the higher frequencies of the input overlap with themselves. A signal which is sampled above the Nyquist frequency is called oversampled. As the degree of oversampling increases, measured as the ratio of the sampling frequency to the Nyquist frequency, the proportion of the total bandwidth used by the input signal decreases. Oversampling by itself only offers a modest increase in the signal to noise ratio. The added benefit of using a sigma-delta modulation is achieved by the noise shaping characteristic of the feedback loop.

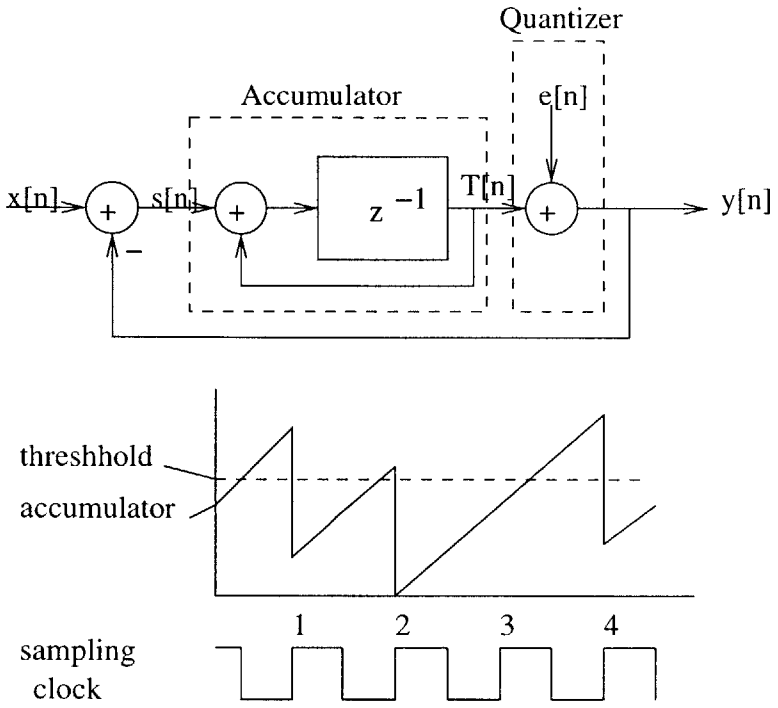


Figure 3-3: Discrete time implementation of a first order $\Sigma - \Delta$ modulator.

A diagram for a first-order sigma-delta loop is shown in figure 3-3. The system is composed of an accumulator and a one-bit quantizer. The quantizer is a threshold function which returns one if the output of the accumulator is greater than some threshold, and zero otherwise. If the accumulator were implemented as a digital unary adder the quantizer would return the value of the most significant bit of the sum. In figure 3-3, this quantizer is modeled as the addition of an error signal to

the output of the accumulator. This error is the difference between the accumulator value and the quantizer output. For an n-bit digital accumulator, the error signal would be the lowest (n-1) bits of the sum.

Consider a constant input into the system, $x[n] = c$. Assume that the value of the accumulator is initially below the threshold of the quantizer, such that $s[n]$ follows $x[n]$. At the rising edge (1) of the sampling clock, the accumulator is above the threshold of the quantizer and sets $y[n] = 1$. This value is subtracted from the accumulator output such that $T[n+1] = T[n] + x[n] - 1$.

The relationship between the input, output and error signals for the modulator in figure 3-3 can be shown to be

$$y[i] = x[i-1] + e[i] - e[i-1] \tag{3.1}$$

The transfer function of the input to the output is a simple unit delay with no magnitude shaping, $|H_x(e^{j\omega})| = 1$. The frequency components of the input is precisely copied in the spectrum of the output. The transfer function between the error and the output has the magnitude response, $|H_e(e^{j\omega})| = 2 \cdot \sin(\frac{\omega}{2})$. A plot of this function can be see in figure 3-4. This shaping serves to reduce the impact of noise around the baseband, but increases it at higher frequencies. The noise from an 8-bit A/D converter is at most $\frac{1}{256}$ of the total input range. Therefore, the noise error for such a system is uniform across all frequency bands.

The key to the success of sigma-delta modulation is the high noise rejection ratio at the baseband. At these lower frequencies, where the noise is at its lowest, the signal to noise ratio is maximal, recalling that the magnitude of the signal transfer function is identically 1 everywhere. By sufficiently oversampling the input, it is possible to ensure that all of the input's spectrum falls within this low-noise baseband. The overall SNR performance of the system would be very poor if the noise in the upper band remained in the output signal. Hence, when demodulating the data, a high quality low-pass filter is typically used to attenuate the noise in the upper frequencies and maintain the high SNR achieved by oversampling.

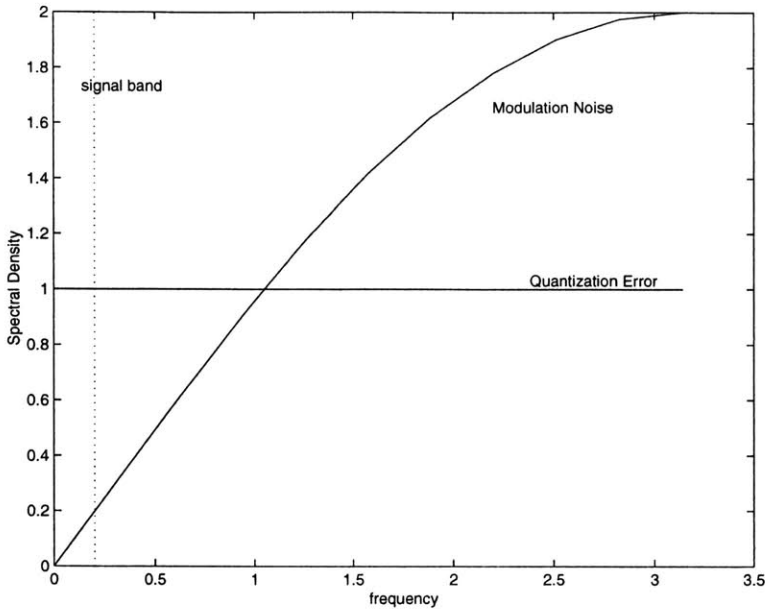


Figure 3-4: Magnitude response for quantization noise in first order system.

3.2 Asynchronous Modulation

Rather than using the synchronous model described above, an asynchronous modulator was used for implementation reasons. Functionally, the asynchronous and the synchronous modular are identical, but the asynchronous circuit is much smaller than its synchronous counterpart. The target design would not use a digital accumulator for the sigma-delta system since the input signal is a continuous time (CT) analog signal rather than a discrete time (DT) digital signal.

When using an analog based integrator, the synchronous modulator must contain a high precision analog subtraction circuit to perform the subtraction in the feedback path. Though it is possible to build an analog subtractor, it is a large circuit which requires several well matched components to function properly. An asynchronous implementation eliminates the need for the complex subtractor, using instead a one-bit resetable memory. A block diagram of this asynchronous modulator is shown in figure 3-5.

Consider a constant input $x(t) = c$ for the diagram in figure 3-5. The output of

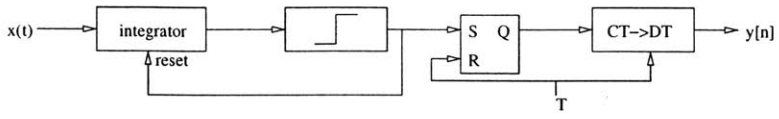


Figure 3-5: Block diagram for an asynchronous modulator

the integrator would slowly rise from the reset value, A, to the quantizer's threshold value, B. Once the threshold is reached, the quantizer output value is asserted, causing the S-R latch to be set and return the integrator to its reset value of A. The latter effect will force the quantizer output back to zero, allowing the integrator to restart and continue. The first effect will cause the system to remember that a reset has occurred. Once every sampling period, the value of the S-R latch is polled, and then reset. If the integrator exceeded the threshold quantity in the period since the last sampling, the output is a 1; otherwise, it is zero.

Figure 3-6 illustrates the behavior of both modulators. The diagram assumes that the input signal is a positive constant, $x[i] = c > 0$, and Z is the output of the system integrators. The solid line represents the behavior of the synchronous version of the system and the dashed line represents the asynchronous response. In both models, the integrator value is reduced by some constant value whenever its output exceeds some upper bound. In the asynchronous model, this reduction of the accumulator value occurs immediately after the integrator value exceeds the bound. For synchronous systems, the integrator resets at the first sampling pulse after the integrator exceeds the bound.

When the synchronous modulator is reset, the integrator value is changed to the same value as the asynchronous waveform. (If the synchronous modulator is reset at $t = t_0$ then $Z_{sync}(t_0) = Z_{async}(t_0)$). This relationship implies that the output for the two modulators are equivalent if A and B are the same value for both models. From the diagram it is evident that there is an upper bound for the input value of the integrator. $x_{max} = \frac{A-B}{\tau}$. Once an input exceeds this limit, the slope of the accumulator output will be large enough that the accumulator will be reset at every input sampling. All values greater than x_{max} will also result in the same bit stream

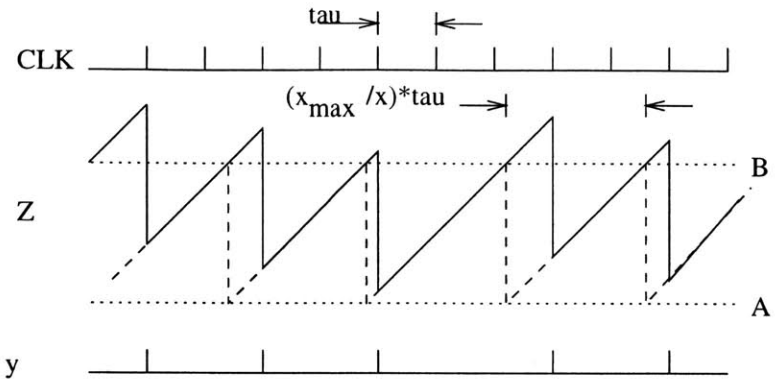


Figure 3-6: Waveform of a clocked synchronous (solid) and asynchronous modulator (dashed).

pattern, and thus be indistinguishable from one another.

3.3 Data Recovery

Once the oversampled bit stream is created, the original input signal is almost completely recoverable. This is done using the system seen in figure 3-7.

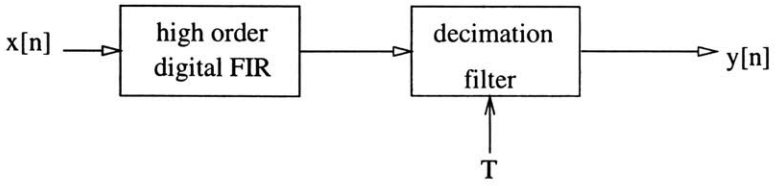


Figure 3-7: Diagram of a typical FIR tap filter.

The high band noise from figure 3-4 can be removed by applying a digital lowpass filter to the target bit stream. To extract higher resolution values for the input the serial stream must then be decimated down to a lower frequency - gaining back in resolution what was lost in bandwidth. Typically, the LPF is implemented as an FIR tap filter. A tap filter consists of a series of latches, multipliers, and adders. Figure 3-8 contains the schematic for a 4 element tap filter.

The implementation of a single stage in the tap filter is very complex. The multi-

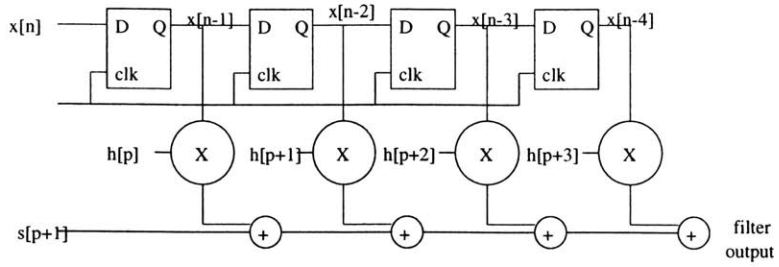


Figure 3-8: System used to extract data from the oversampled bit stream

plier circuit alone would require more area than can fit within the constraint of a single pixel. Since this implementation would be prohibitively expensive for a pixel-parallel processor, other methods for extracting the data were also explored. A recursive decoding algorithm based on the bit stream structure is presented.

3.3.1 Standard Filtering Techniques

The SNR of the LPF is related to the frequency response of the digital filter. A high order LPF would have a better noise rejection characteristic than a lower order filter. Figure 3-9 shows a few of the many possible filters which can be used.

The ideal lowpass filter is one in which the baseband frequency is unaffected and the higher frequencies are completely attenuated - as seen in figure 3-9(a). Using an ideal lowpass filter, the signal band noise is found to be

$$n_0 = e_{rms}\frac{\pi}{\sqrt{3}}(2f_0T)^{\frac{3}{2}} \tag{3.2}$$

Where n_0 is the rms value for the total noise power in the signal band and $2f_0$ is the Nyquist sampling rate for the signal. Since $2f_0T^{-1}$ is the oversampling ratio, equation 3.2 states that the maximum achievable resolution is 1.5 bits (9 dB) for every doubling of the oversampling ratio.

The averaging filter, seen in figure 3-9(c), has significantly poorer performance than the ideal filter and the near-optimal triangle filter. The SNR for the averaging filter increases its resolution by only 1-bit for every doubling of the oversampling ratio.

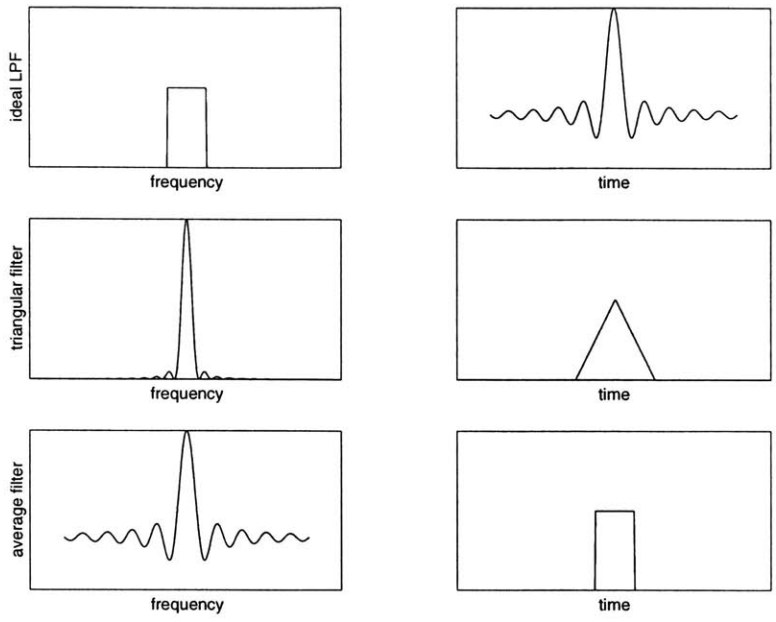


Figure 3-9: Frequency and temporal response for (a) an ideal low pass filter, (b) a triangular filter, and (c) a time averaging filter.

Figure 3-10 graphically demonstrates the better noise performance of the triangle filter over the averaging filter.

It can be shown that the triangular filter seen in figure 3-9(b) has an SNR which increases by 9dB for every doubling of the oversampling ratio. It is near optimal since its SNR curve, seen in figure 3-10, has the same slope as the SNR curve for the ideal filter. However, the SNR curve for the triangle filter is a small offset below the optimal curve which makes its performance slightly worse than the ideal filter's.

3.3.2 Recursive Decoding

Implementing the lowpass filters as FIR systems is not a viable option for the 3D design. Tap filters require large complex circuits to implement and thus are not area-efficient. The many bits of output which would have to be passed between layers also consumes area which could be dedicated to processing circuitry. Consequently, another approach had to be devised for reconstructing the original inputs from the bit streams. This was done by taking advantage of the structure of first-order sigma-delta

modulator outputs.

It has been shown that these bit streams have a unique structure[2]. They are composed either of a sequences of ones separated by single zeros, or sequences of zeros separated by single ones. The length of these sequence is determined by the constant input to the first-order modulator and can have only two values, differing at most by 1. This sequences of sequences also has a significant structure. If every instance of the longer sequence is coded with a '1' and shorter sequences are coded with a '0', another bit stream can be derived which can be proven to be indistinguishable from the output of a first-order sigma-delta modulator. By recursively applying this substitution, all finite length bit streams can be reduced to a base case. From this base case it is possible to recurse back through the levels of substitution to find the value of the original modulator input[2].

The resulting recursive decoder model is a near optimal method for approximating the sigma-delta modulator's input signal. The SNR increases by 9dB per octave, 1.5-bits for every doubling of the oversampling ratio, and the constant offset is smaller than that of the triangular filter. Figure 3-10 compares the SNR vs. number of samples for the three methods.

From the plot, it can be seen that the recursive decoder has a better signal-to-noise ratio than either of the other two methods. To extract an 8 bit value requires 45 samples for the recursive decoder, 64 for the triangular filter, and 256 samples with the averaging filter. By eliminating the recursive step of the algorithm and simply averaging the lengths of the sequences, it is possible to generate a value for the input which is as accurate as that of the triangular FIR. The implementation of such a design would be smaller than either a triangular tap filter or a recursive decoder, and still generate near optimal outputs.

3.4 Bit Stream Computation

One goal of the project was to identify interesting methods of computation using the bit stream encoded values of the pixels. While it would be easiest to demodulate

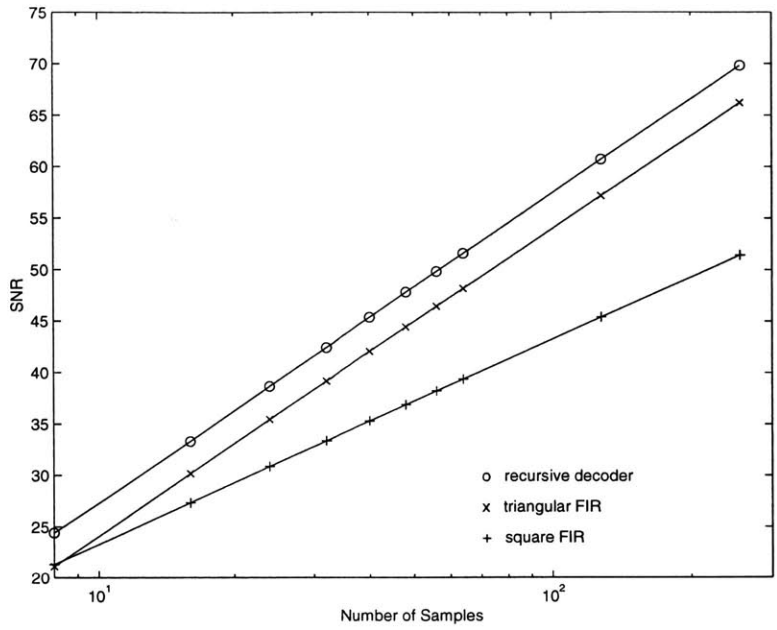


Figure 3-10: SNR for the recursive filter, a triangular filter, and the averaging filter

the signals and perform algorithms on the outputs, it would not be as efficient. The demodulation hardware was found to be complex and so, not in keeping with the design philosophy. Even using the small recursive decoder, a high resolution digital output would be generated. To perform computations using these high resolution values would require conventional digital systems which are typically large and involve many interconnects.

It would not be feasible to use standard digital models for computation in a high density parallel architecture like the proposed imager. A better model to follow is one in which all processing is performed on the bit streams prior to their demodulation. A series of components were designed which are able to perform functions on these 1-bit data streams and generate 1-bit outputs. There are components which can perform addition, subtraction, and storage.

3.4.1 Bit Stream Subtraction

Using the knowledge of the bit stream structure described in 3.3.2, it is possible to build small modules which are capable of performing simple functions on the pixel bit streams. Modules which can perform rudimentary addition and subtraction of bit streams have been explored. Though their outputs do not permit simple reconstruction of the input value, these modules can still be used to perform basic tasks.

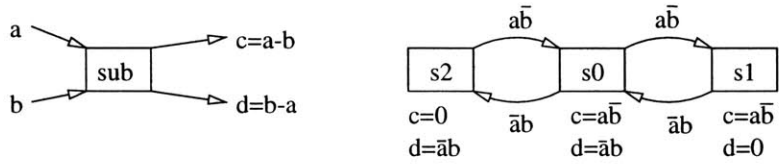


Figure 3-11: Block diagram and state machine for a bit stream subtractor

Consider the system in figure 3-11. This finite state machine is able to compute a value for **a** - **b** and **b** - **a**, where **a** and **b** are sigma-delta encoded bit streams. The system is essentially composed of two one-bit half-subtractors which have been joined such that they share a common state, s0. The FSM works by detecting consecutive unmatched 1's in the bit stream. Consider the two signals, a and b as follows.

$$\text{a} = 1\,1\,1\,0\,1\,1\,1\,0\,1\,1\,1\,0\,1\,1\,1\,0\,1\,1\,1\,0$$
$$\text{b} = 1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0$$

The value for a is approximately $\frac{3}{4}x_{max}$ while the value for b is approximately $\frac{1}{5}x_{max}$. These values are exact to within the quantization error afforded by 20 bits of data. Given these inputs, the estimate of the outputs would be $c = a - b = \frac{11}{20}x_{max}$ and $d = b - a = -\frac{11}{20}x_{max}$. If the subtraction circuit was initially in the neutral, s0 state, the outputs would be

$$\text{c} = 0\,0\,1\,0\,1\,0\,1\,0\,1\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1\,0$$
$$\text{d} = 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0$$

The first difference, $c = a - b$, has a time average of $\frac{1}{2}x_{max}$ and the second difference, $d = b - a$ has one of zero. The first differs from the estimated value

by $\frac{1}{20}x_{max}$ and the second is outside the range of the subtractor, as the module is unable to represent negative results. The information is not completely lost, since the magnitude of the difference is always encoded in the non-zero output of the subtractor. Consequently, by OR-ing the outputs together, it is possible to generate the absolute value of the difference, $|a - b|$. The error in **c** is caused by the choice of initial state. Had the the initial state been s1, the output would have had no perceivable error between **c** and the estimated value. When the signals terminated, the final state of the system was s1, so if the input signals, **a** and **b** repeated the same sequence, the time average of **c** would approach the estimated value of the difference.

The structure **c** cannot be modeled as the output signal of a first-order sigma-delta modulator. Consequently, it can not be used as the input to a subtraction unit unless the other input has the structure of first-order sigma-delta bit stream. This feature allows for differences to be thresholded against a reference value. Consider the arrangement in figure 3-12.

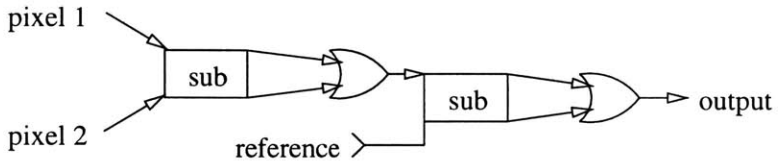


Figure 3-12: Localized edge detection unit.

The circuit above behaves like a local edge detector. By finding the difference between adjacent pixels, an estimation of an image's intensity gradient field can be determined. An edge can be roughly modeled as those locations in an image where the gradient field is above some threshold value. Consequently, by subtracting the reference—a first-order sigma-delta modulation of the threshold—from the magnitude of the gradient, edges between pixels can be found where the output from the circuit is non-zero. Once edges have been identified, it is possible to perform higher level functions such as connectivity and segmentation. Subtraction circuits can also be used in several other applications. Template matching, for instance, is a prime example of an algorithm which requires many subtraction operations to generate error signals.

3.4.2 Bit Stream Addition

Template matching requires not only the use of subtractors, but the use of adders as well. Figure 3-13 illustrates one potential design for a bit stream adder. The module is a half-adder which combines two first-order sigma-delta encoded bit streams and generates a bit stream whose time average is the sum of the inputs. Like the subtractor, the adder's output is not characteristic of a sigma-delta modulation.

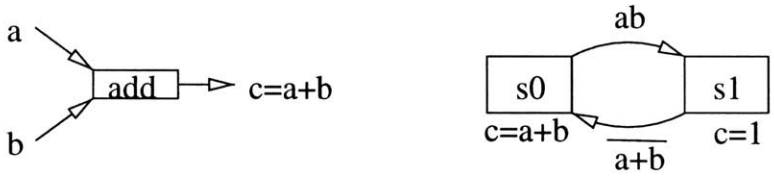


Figure 3-13: Block Diagram and State Machine for a bit stream adder

Consider the two inputs, **a** and **b**, such that

$$\text{a} = 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0$$
$$\text{b} = 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0$$

It is apparent that the value of **a** is $\frac{1}{3}x_{max}$ and that **b** had a value of $\frac{1}{4}x_{max}$. Again, these are only accurate to within the precision which 12 data samples allow. The expectation for the sum is $\frac{7}{12}x_{max}$. The actual output is

$$\text{c} = 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0$$

which has a time averaged value of $\frac{7}{12}x_{max}$. Similar to the subtractor circuit, the outputs from two adder circuits can not be fed into a third adder; however, it is possible to build a more complex module which is capable of adding two such inputs. Its finite state machine is drawn in figure 3-14.

Both the adder and the subtractor can be implemented in a relatively small area. A subtractor requires only two bits of state and the adder requires one. Both systems also use a small amount of control logic to determine the next state and the outputs. One implementation of a template matcher would require one layer of subtractors

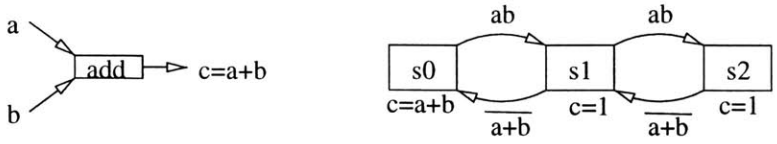


Figure 3-14: Block Diagram and State Machine for a bit stream adder capable of adding subtractor outputs

to generate the differences between the pixels and the template, and several layers of configurable adders to calculate the degree of correspondence by adding up the generated differences.

3.4.3 Bit Stream Storage

Algorithms more complex than template matching often require a storage mechanism to store a reference value which can be used for later computation or comparison. Conventional thought dictates that, to store the bit stream, it would first have to be converted to a digital value and stored using some type of register. The size of the demodulation hardware has already been found to be complex add to this the complexity of a single 8-bit register and the resulting circuit would be much larger than an acceptable pixel footprint. Consequently, another method was devised for storing a unique value corresponding to a bit stream which required neither demodulating the stream or implementing large registers.

Storage of the bit stream frequency is accomplished by matching the frequency of a voltage controlled oscillator (VCO) to the input. Once the two signals are matched, the reference voltage for the VCO is a representation for that frequency. Storage of this value can be maintained by breaking the feedback loop. Without this connection, changes to the input cannot affect the stored analog voltage.

Matching the VCO and pixel input values is accomplished using a frequency detector circuit. This is a modification of a phase-frequency detector (PFD) in that it only attempts to lock the two oscillators in frequency and not in phase. A frequency detector can be implemented as the simple finite state machine seen in figure 3-16.

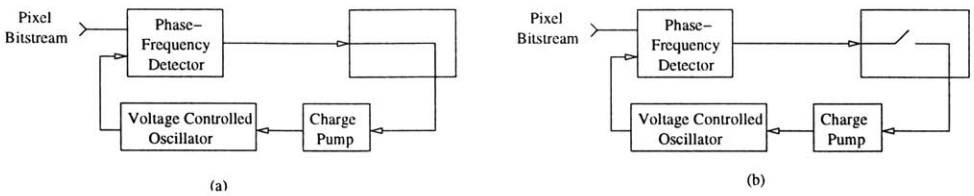


Figure 3-15: The frequency value can be matched by closing the loop (a), and stored by opening it (b).

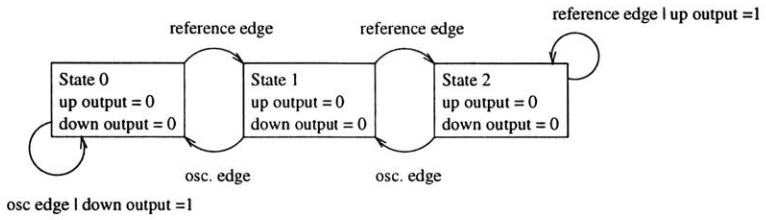


Figure 3-16: State transition digram for a frequency detector

The design is very similar to the bit stream subtractor circuit. The PFD, like the subtractor, generates a difference signal. Instead of generating the difference between the bit stream values, it generates the difference between the number of rising edges of the bit stream input.

The outputs of the frequency detector indicate whether the VCO frequency is higher, lower, or equal to the input signal. These signals can be used to govern the behavior of the charge pump. If the frequency detector finds an unmatched edge on either input, it generates a correction pulse. An *unmatched edge* is a rising edge on either of the inputs which does not have a corresponding edge on the other signal. The correction pulse is sent to the charge pump to initiate a change in frequency. Figure 3-17 shows the generation of a corrective pulse resulting from an unmatched edge in the second frequency signal.

The correction signals generated by the frequency detector are interpreted by the charge pump as control signals. The charge pump alters the reference voltage, which results in a change in the VCO output signal. The new signal should more closely match the input frequency. A bit stream is never precisely stored in the design;

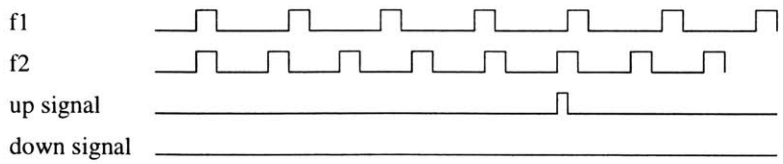


Figure 3-17: The response of the frequency detector to unequal frequencies

however, the stored frequency can be brought arbitrarily close to the target frequency within the precision allowed by the circuits by leaving the feedback loop closed. So long as the input value remains constant, the loop will continue to approach the correct value - an approximation to the input can be made which is sufficient for most computational requirements.

The speed at which a bit stream can be stored is a combination of two primary factors. First, frequencies are matched much faster if the reference voltage of the storage mechanism prior to the start of the target input is already close to the voltage corresponding to the target input. A storage mechanism with a VCO operating at 100Hz would lock into a 90Hz signal faster than a storage system with a VCO operating at 10kHz. The other governing factor on the speed of storage, is the size of voltage increments generated by the charge pump. For optimal performance these ΔV's must be monotonically related to the frequency difference . If there is a large difference between the stored frequency and the input frequency, the charge pump should make large increments in the reference voltage to more quickly approach the input frequency. Once the VCO frequency is within some neighborhood of the input, the ΔV changes to the charge pump should decrease in size, as the charge pump performs more fine-tuning of the reference voltage. As ΔV approaches zero, the error in the stored value will also.

Using these three modules, it is possible to implement a variety of simple pixel-parallel algorithms. An algorithm had to be chosen which exhibited several key properties. First, it should have a repeating structure such that every pixel performs the same computations. Second, each of these computational modules must be easily dividable into several layers with relatively few interconnects between them for effi-

cient 3D implementation. Finally, the algorithm should be one which could use the computational modules described above.

Chapter 4

Activity Detection

An activity detection system is similar to a motion tracking system, in that they both identify areas of motion in a sequence of images. While a motion tracker will assign a velocity to every pixel in the image, an activity detector assigns a scalar value to every pixel. The magnitude of this value is a measure of how "active" the system believes that pixel to be. By thresholding these values, it is possible to identify areas of the image which contain active sites. Activity detection has many potential applications. As a security device, an activity sensor would be able to monitor important areas and alert security guards when something has been detected. Activity detection also has the potential to reduce transmission bandwidth and storage space. By only transmitting sections of the image where activity is occurring, less data would have to be sent, freeing bandwidth for additional cameras. Or, by only saving images where activity is occurring, less storage space would be required.

A good activity detection system should have several properties. First, it should be able to differentiate between the portion of the image that is background and those that are not. The sections which are not background correspond to the active region of the image. An activity detection system should also be able to adapt to slow variations in the background. As the sun moves during the course of the day, shadows shift and the incident light on the background changes. These slow variations should not trigger alerts. The ideal system should also be able to compensate for oscillating pixels. Pixels whose value oscillate between two or three values - a tree blowing in

the wind, a fluttering flag, and a traffic light are but a few examples that result in "oscillating pixels." These signals should also be part of the background and not affect the activity of the system.

One system which meets all of these functional criteria is the multimode adaptive sensors developed by Stauffer[4]. The model is presented in 4.1. Using this basic approach for finding activity, a similar system which could be more easily implemented in small silicon areas, has been developed. The potential circuit has been simulated using an existing sigma-delta based imager.

4.1 Multimode Activity Detection Model

Current background subtraction techniques are done by learning the background at some initialization time. These methods suffer if the background changes over time. A shadow moving during the course of a day would be sufficient to cause a false positive. This problem could be avoided if an adaptive system were built which could dynamically adjust to these changes and learn a new background model. A static representation of the background - even if updated frequently would not be able to adapt to oscillations in the image. The canonical example of this behavior is a tree blowing in the wind. In such a case, pixels would oscillate between the value of the leaf and the object behind it. Stauffer's work attempts to address both of these issues[4]. The model learns multiple modes for each pixel and is capable of dynamically learning a background, even if there are local oscillations in pixel values.

The design incorporated in this model is an activity detection system which has components that can be implemented in conventional silicon. The model was chosen because it contains elements which can be parallelized on a per pixel basis, lending itself towards implementation in multilayer silicon design. Some modifications were made to the design to simplify the final implementation.

4.1.1 Observed Pixel Behavior

Through observations of pixel values of a fixed imager, it was found that the distribution typically could be modeled as a sum of Gaussian distributions, or a set of Gaussian distributions in an intensity-probability plane. Such plots can be seen in figure 4-1. The mean of these distributions represents the actual value of the pixel and the small variations are caused by small changes in the light, in the imager, or in a variety of other factors. An oscillating pixel value, such as described in the previous subsection, manifests itself as a bimodal distribution - which can be modeled as the sum of two Gaussians. For a leaf on a tree blowing in the wind, one of these Gaussians would represent the leaf, and the other, the object behind the leaf.

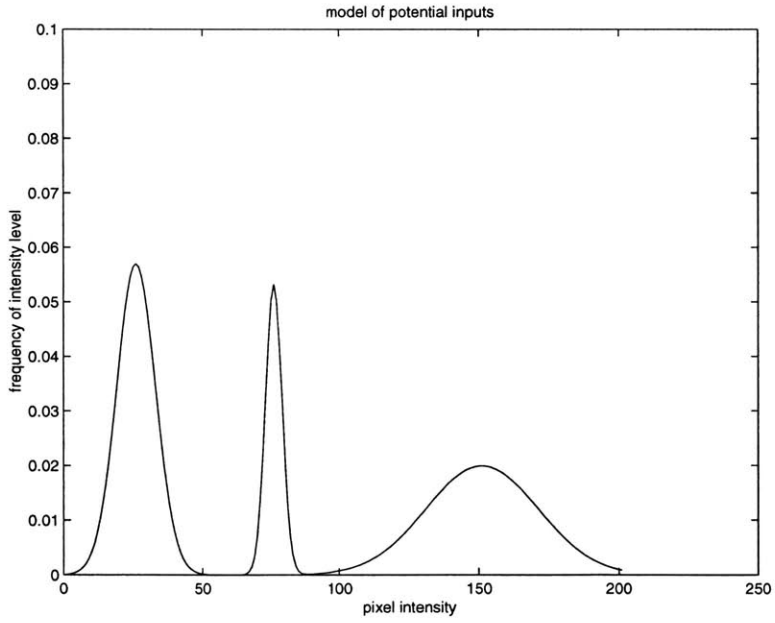


Figure 4-1: The variation in a pixel value has a distribution which can be modeled as a sum of Gaussians.

A Gaussian can be specified using a mean and a standard deviation. Each of the pixels in the image can have several Gaussian models associated with it. Let us consider a pixel which has changed its input value. One of the Gaussian models associated with that pixel begins to learn this new input by establishing a mean and standard deviation. While the input value remains within some small deviation of the

mean, the Gaussian model continues to be built, improving the standard deviation as the constant pixel value is maintained. This adaptive learning behavior is seen in figure 4-2.

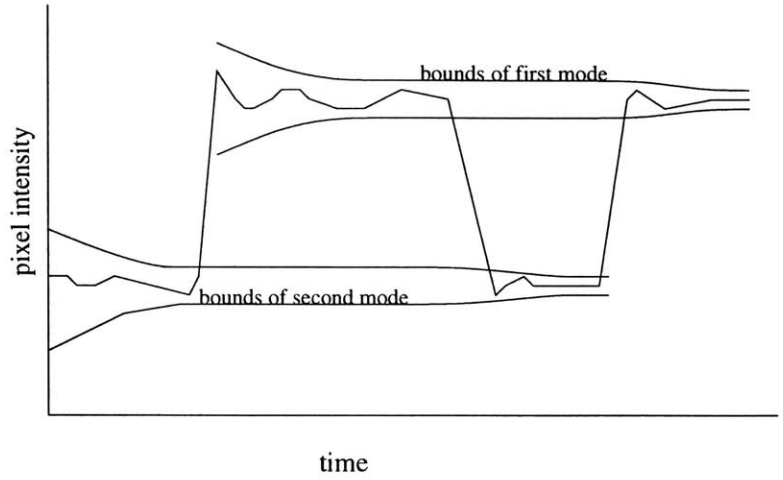


Figure 4-2: As a signal oscillates between two values, two tracking modes tighten their bounds on each of the separate signals.

After some amount of time, let the input value change to small oscillations about a different constant value. To adjust to this new signal, another of the Gaussian models associated with the target pixel is assigned to learn the new value. Again, it begins with a wide standard deviation which is refined as more samples are added. If the input signal then returns to the original value after some time, the first model will identify that the new level is sufficiently close to its value to be considered a match. It will then continue to improve its model for the input.

In the above example, the pixel is only active twice. The first instance occurs when the first value appears initially. The second active session occurs when the input value changes to the other value. During subsequent input changes, the new input signal already satisfies one of the pixel's associated Gaussian models. Using a set of Gaussian models for each pixel input permits the system to be capable of learning the background Since the models are constantly updating, a slow change would not cause the system to signal an alert.

4.1.2 Identifying the Background

One of the key issues associated with this multiple mode approach to background detection is determining which of the modes corresponds to the background value. Upon closer examination of the problem, it becomes no longer a question of which mode represents the background, but which modes occur sufficiently frequently to be considered regular values for the input. It becomes necessary to create a metric which can be used to determine whether a mode is a regular value or a transient one.

The properties of the metric are threefold. First, the metric for a mode should be large if has been present for a long time. This would be indicative of the case where the pixel input has been constant with little or no variation in the past. Second, the metric should be greater for those modes with small standard deviations. Such signals are the ones which not only have been present for sufficient time for a model to be built, but which also exhibit very little fluctuation in their value. Finally, as the time since the last appearance of a value that is part of the mode increases, the metric should decrease. The following subsection proposes a ranking metric which satisfies all of these properties.

Ranking the Modes

In the example of the oscillating input, no attention was given to the time between input changes. It was assumed that the switching was sufficiently fast to ensure that neither of the two Gaussian models had become expired - or in other words, neither of the modes were so long unused that they were no longer considered valid values for the pixel. The determination of the age of a particular mode is done by taking the ratio of a prior probability (π) and the model's standard deviation ($rank_m = \frac{\pi_m}{\sigma_m}$). The prior probability is a weighting assigned to each mode based on its values and their frequency of appearance over some amount of time. One formula which exhibits the desired properties is $\pi_m[n] = \alpha f(n) + (1 - \alpha) \cdot \pi_m[n-1]$ where $f(n)$ is a boolean function which returns 1 if the pixel value at time n fits within the bounds of mode m and 0 otherwise. α is an experimentally determined constant which represents the

learning rate of the system.

This ranking system exhibits all of the properties desired for a ranking metric. The metric for older signals decays over time through the prior probability function. It also enables older patterns to have equal or greater rank than a new signal which has not yet been well learned. Background values, which would be expected to have smaller standard deviations and larger priors, would be given the highest ranks. Transient signals such as a car driving along a street, would be given smaller ranks by virtue of having both a small prior and a larger standard deviation.

Replacement Strategy

For a model with n modes, a replacement strategy must be implemented once the n+1st input value is encountered. The strategy is responsible for deciding which of the older modes to replace with this new input. Using the ranking system described in subsection 4.1.2, the choice is made by choosing the mode with the lowest rank. This guarantees that neither the more recent signals nor those which are present for long periods of time are replaced. The ranking system ensures that in typical applications, the background information will not be lost due to input value changes.

4.2 Activity Detection Algorithm

A methodology for implementing a system similar to the one developed by Stauffer is presented in this section. While designing the system, the key points of concern are to maintain a small physical footprint, and to maximize modularity. A modular design would enable the system to be more economically built in 3D silicon, by reusing the same masksets for different layers.

4.2.1 Overview

The multimode tracking system can be segmented into two components, the Gaussian models and the control logic. To simulate the Gaussian models, there are a set of memory modules as seen in figure 4-3. These modules are responsible for learning and storing pixel values. Controlling these modules is the second component, the control system. This system is responsible for deciding which of the modules should be tasked to tracking the current input as well as deciding when the pixel input is a new value.

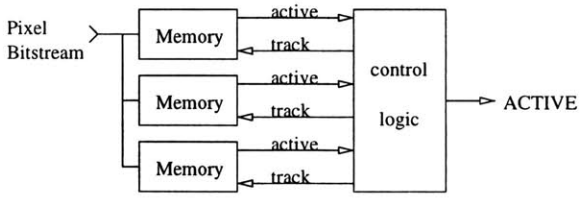


Figure 4-3: Block Diagram of basic multimode background system

The division of the system in this manner allows for maximum benefit from the 3D process. The control system could reside on one layer of the design, lying just below a pixel. Each of the identical memory modules could then be contained on layers stacked below that one. By implementing the design in such a fashion, the complexity could be reduced if all of the memory layers were identical, as seen in figure 4-4.

There are two major differences between the proposed physical system and the

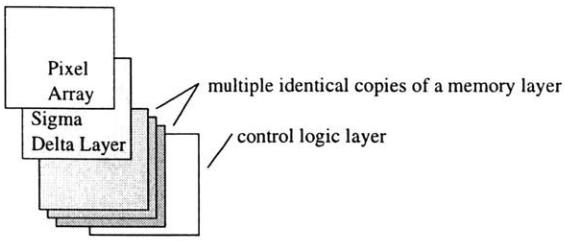


Figure 4-4: The multimode system has a high degree of repetition allowing for many copies of the same layer

software-based system in section 4.1 . The Gaussians model from the software system are able to adaptively change their standard deviations, and thus reduce the spread, Δf, of frequencies which are covered by a particular mode. The proposed memory modules are not capable of dynamically changing their bounds, but rather used a fixed Δf to define acceptable values. In the software model, a complex ranking system was generated to identify the most probable value for a pixel. The robustness of that design was traded for a simpler replacement strategy. Both a least recently used (LRU) and a sequential ordering strategy were examined.

4.2.2 The Control Logic

The main purpose of the control logic is to execute a replacement strategy for the memory modules. Specifically, during input frequency changes, it must decide which, if any, of the memory elements should be assigned to track the new signal. Optimally, the element would be one which represents a transient signal, since storing a new value would necessitate deleting the older one. The replaced module should never be the one containing the stored background value. There are many ways to decide which module should be replaced, and two such possibilities were explored.

In the previous work, the selection was accomplished by assigning a ranking to every mode based on the ratio of its prior probability and its standard deviation. The pixel which had the lowest ranking was selected as the target for replacement. Such a strategy behaved very well, however, it could not be feasibly integrated into the control system since the required circuitry would be too large.

Two alternate strategies were examined. The first was a standard least recently used (LRU) system; and the second, a sequential approach. In the LRU method, the oldest pixel is chosen to track in to the new signal. This model presumes that signals which have not been encountered for some time were transient signals and are not likely to reappear; or if they do, they represent activity of some sort. The LRU strategy thus presumes that the background will always be one of the most recently encountered signals. In sequential ordering each of the memory elements are ordered so that whenever a new signal is detected, the next element in line is assigned to track it.

The LRU strategy consumes less area than the prior probability calculator, but remains fairly complex. It requires that each element be able to store its age relative to the other elements and must contain logic to decide which element is oldest. The smallest possible method for monitoring age would be to use a small analog counter. Even with this tiny device, two dedicated signals from the control system would have to be connected to every mode. This implies that the control logic would have to have prior knowledge of the total number of memory modes. While that may not be a large problem, routing all of these connections would be. As is stands, the LRU system would perform well, but would have a high cost both in terms of area and design inflexibility.

The sequential ordering approach has the benefit of a small footprint implementation. It could be implemented as a circular shift register with a single latch on every layer. Of the many latches, no more than one would contain the active signal. To activate the next element, the control logic need only shift the registers. The current memory mode would then receive an inactive signal. The memory module on the subsequent layer would then begin to track the input. This design limits the per layer implementation to a single latch and the interconnect requirement to three signals - the shift value, the shift enable signal, and the return signal from the bottom layer to the top. A sequentially ordered replacement strategy would allow for the same control logic to control an arbitrary number of memory modules. The design complexity is greatly reduced using this model, but only at the cost of performance.

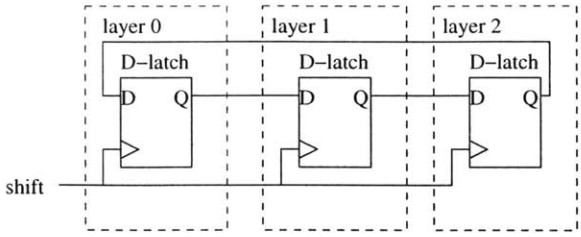


Figure 4-5: A multilayer circular shift register

Of the two proposed methods, the LRU system has a slightly better performance overall. Performance of a replacement strategy is a measure of the rate at which the background is not chosen to be replaced. The LRU system would only replace the background value during periods of heightened activity. The sequential system would replace the background value once every **n** changes in pixel value, where **n** is the number of modes in the system. For applications where high activity is expected, or when false positives from background values are acceptable, the sequential strategy would be the better choice. LRU should only be chosen when low activity, high-efficiency is needed.

4.2.3 The Memory Module

The memory module is the core element of the design. It is this system which is responsible for learning and storing a representation of the current pixel value. The module also incorporates a monitoring system to determine whether the new input signal is different from the stored value. The design of the module incorporates several analog components in place of conventional digital elements because of space constraints. The tight integration of digital and analog designs results in a compact system which meets the design criteria of both performance and size.

The diagram in figure 4-6 depicts the basic memory module. The system is composed of a frequency locking loop and an activity monitoring/filtering portion. The frequency locking mechanism is identical to the one described in 3.4.3 with the addition of a monitoring system into the feedback loop. The monitor system tracks

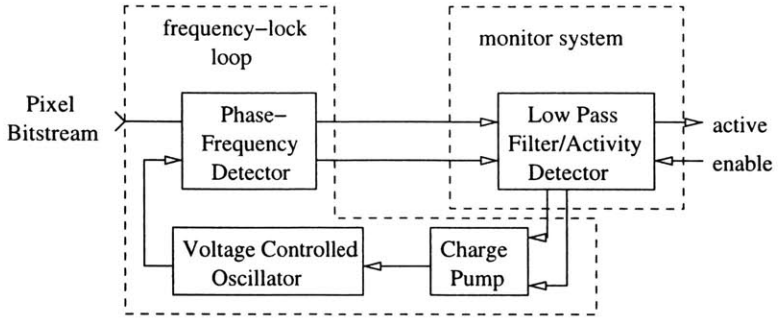


Figure 4-6: Block Diagram of basic multimode background system

the error signals from the PFD and indicates when the input value is different from the stored value. This information is conveyed to the control logic. If this module is assigned to track the input, it is the monitor system which generates the correction pulses to the storage system's charge pump.

Monitoring the Frequency Loop

The purpose of the monitor component is to report changes in the pixel input and decide if the charge pump should alter the reference frequency. Rather than simply watch the charge pump correction signal and decide whether the input is changing, the monitor is placed within the loop to be able to prevent a new signal from altering the reference voltage too quickly. As a loop filter, the monitor behaves much like a low-pass system. If the pixel value does not match the VCO value, the monitor signals the control logic. The correction signal for tracking the new pixel value is only allowed to pass through if the module's enable signal is activated by the control logic. If the pixel input is within the specified Δf of the VCO frequency, the correction signals are issued to the charge pump to better match the VCO and the input frequency. This allows the input signal to slowly drift during the course of a day without triggering an alert. Such behavior would include the motion of the sun, or a candle burning in the night.

The monitor is composed of three edge counters, two 2-bit counters, one 5-bit counter and a simple compare logic unit. Each of the two 2-bit counters are connected

to the outputs of the frequency detector. The third counter is used to measure either the number of pixel edges or edges of a fixed reference oscillator. The implications for this connection choice are discussed later in this subsection. Figure 4-7 illustrates this circuitry.

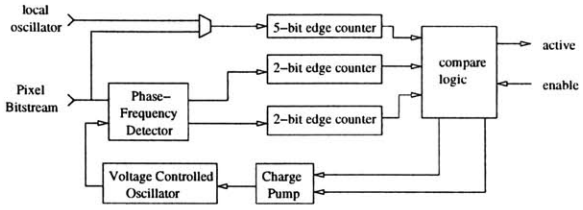


Figure 4-7: Diagram of a memory module's monitoring subsystem

The implementation of the edge counters would be as analog counters. The use of analog counters is possible only because their outputs are used for comparisons and not computation. As illustrated in Figure 4-8, an analog counter represents an immense savings in size over its digital counterpart. The footprint of the analog counter is small, compared to the size of the many gates and latches needed to implement the digital counterpart.

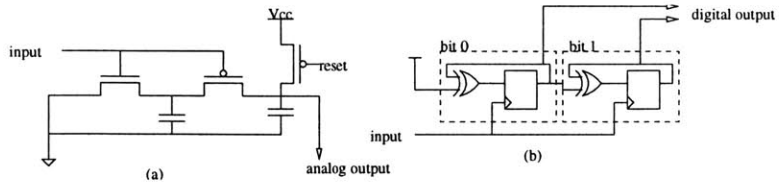


Figure 4-8: Implementation of an analog counter (a) and a digital counter (b).

The monitor is able to determine activity by counting the edges on the frequency detector outputs as well as on the pixel input bit stream or fixed oscillator. Whenever there are two or more unmatched pulses the compare logic checks the distance between them. If this distance is less than some threshold (δ_1) - that is to say, if unmatched pulses are too close together, the new input is considered to be active. Alternatively, if there have been more than some threshold (δ_2) number of input edges, then the

VCO is considered to be sufficiently close to the input frequency. Once either of these criteria have been satisfied, the counters are reset and the process begins again. The use of two different threshold values creates a buffer region at the edge of activity and inactivity which behaves much like the hysteresis of a Schmitt-Trigger.

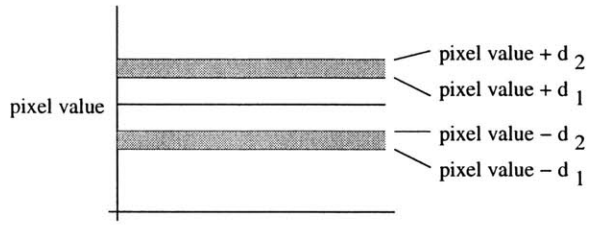


Figure 4-9: By using the correct thresholds, a Schmidt-trigger style hysteresis can be obtained.

The diagram in figure 4-9 illustrates the boundaries defined by the two thresholds, δ_1 and δ_2. For the VCO signal to be considered adequately matched to the input frequency, it must be within $\frac{1}{\delta_2}$ of the target frequency. The VCO will remain locked to the input until its frequency difference is greater than $\frac{1}{\delta_1}$. A useful hysteresis is only accomplished if $\delta_2 > \delta_1$.

The third edge counter is used to establish the measurement for the time between unmatched pulses. There are two possibilities for measuring the time. The first is to use a fixed global oscillator. This method requires an added clock distribution network across the layers; however, it provides a uniform measure of time for all of the modes. An alternative is to use the input bit stream edges as the basis for the counter. Using this input method, the number of interconnecting vias is reduced, but it imposes a non-uniform frequency matching constraint for the frequency-lock loop.

Measuring the distance between unmatched pulses using the pixel wavelengths produces a measure of the frequency difference relative to the input frequency. This measure is not as accurate as an absolute temporal measurement. Higher frequency signals have larger acceptance bandwidths than for lower frequency signals using the wavelength measurement while a temporal measurement has a uniform width acceptance band; however, to enable the time measurement, a global clock would

have to be available. Though only one such oscillator would be needed for the entire chip, it would require additional interconnect between all of the layers to distribute the clock signal. By altering the frequency of the clock signal, it is possible to reduce the size of the 5-bit counter to one or two bits of resolution.

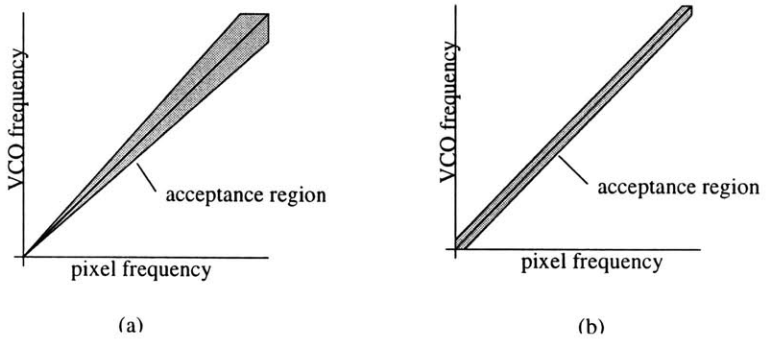


Figure 4-10: A proportional δf system (a) would have less accuracy at higher frequencies than a fixed δf system (b).

The distance between unmatched pulses is directly related to the frequency difference of two oscillators. Consider the two signals shown in figure 4-11. The first signal has a slightly higher frequency than the second signal. The frequency of the unmatched pulses, $f3$, would be the difference of the two input frequencies, $f3 = f2 - f1$. The larger the difference between the target frequencies, the smaller the distance between unmatched pulses. A measure of this distance is actually the wavelength of the $f3$ signal.

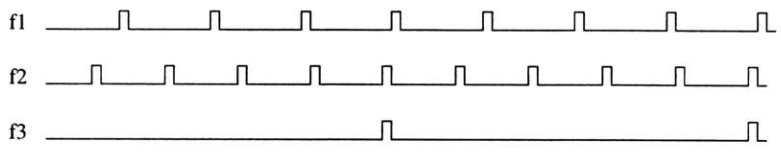


Figure 4-11: Two unmatched frequencies (f1 and f2) and the resulting sequence of unmatched edge detections (f3).

The distance between unmatched edges can also be used to generate a better guess for charge pump behavior than a simple increase/decrease option. When the distance between the unmatched pulses is low, the difference between the two signals

is relatively high; if the distance is high, then the difference is small. Using this knowledge, is it possible to build a faster tracking frequency loop. The improvement is accomplished by using a graded charge pump. A graded charge pump is a system which allows the output voltage be incremented by one more than one δv. If the pixel bit stream is known to vary greatly from the VCO frequency, the reference voltage can be incremented by a large step. And when the distance between unmatched pulses is large, a smaller step size can be used to adjust the reference voltage.

Chapter 5

Results and Analysis

The activity detection system which has been described in chapter 4 was simulated in software. The system was designed to run on two separate processors. Initial development was targeted for a dual 40Mhz DSP system which was used in a development environment to simulate and test candidate 3D architectures. The DSPs used were two TI TMS320C40 processors mounted on a White Mountain Slalom-40^{TM} development card. A faster processing system was required to simulate the full array of pixels in real time. Consequently, the simulation was rewritten for a faster Pentium-based PC.

Bit stream values were created using a small test imager. A MOSIS test imager was fabricated containing an array of photodiodes and sigma-delta modulators. The imager was housed in a camera system which controlled the sampling speed for the modulators, and the output. Once a predefined buffer is filled, the data is uploaded to the processing platform. There, the parallel architecture is simulated on a serial processor while simultaneously decimating the bit streams to generate a displayable image.

5.1 Experimental Setup

To test the functionality of the physical model, a simple two dimensional version of the basic imager was implemented in a 0.5 μm MOSIS process. This camera contains

an array of photodiodes that are each coupled to a current controlled oscillator. The output from the chip is the bit stream data for every pixel in the array. Test chips have been fabricated with 48x48 and 64x64 pixel arrays.

The test chip is controlled by a camera system which is responsible for maintaining the bias voltages, monitoring the power consumption of the device, and controlling the sampling speed of the pixel oscillators. Data gathered by the test system is sent to the simulation testbed in one of two methods. The first is a high-speed 32 bit interface that provides data to the Slalom-40 DSP card housed within a PC. The second method for uploading data is a standard IEEE 1284 parallel port which is used to interface to a PC without the DSP card.

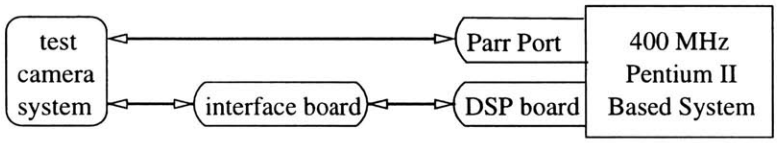


Figure 5-1: Experimental data was gathered using the test system and then collected in the PC.

A DSP-based development card was used for initial system simulations. It served as an initial testbed for the simuation. Once the algorithms were working reliably on that system, they were ported to a C++ Microsoft Windows98 based application which was independent of the DSP card. With the test system and a portable computer, the simulation could be moved to observe different surroundings and monitor the behavior of the camera under various lighting conditions and traffic patterns.

To capture an image, the test system first samples each pixel at a uniform rate to generate the sigma-delta bit stream. The data is temporarily stored in local memory until a predetermined number of samples has been reached. Once the sampling is complete, the data is then transmitted to the simulation system. A high-speed custom parallel port interface was used to provide high bandwidth communication to the DSP system. This 32-bit wide interface could transfer data at a maximum rate of 20 Mbytes / second. To transfer data to the PC, the test system used the standard IEEE 1184 ECP parallel port interface. The theoretical maximum throughput of this

interface is 2 MBytes / second. Due to the interface format, the performance was slightly less than this maximum.

5.2 Simulation Results

Figure 5-2 contains the reaction of a single pixel to an oscillating input. The pixel has two memory modules which are shown in the graph. The top row of graphs shows the input value to the pixel. It oscillates between two constant input values of -7 and -41. Generation of the test patterns is accomplished by integrating this input using an 8-bit integrator and returning the overflow bit. These constants correspond to sigma-delta modulator inputs of $\frac{7}{256}x_{max}$ and $\frac{41}{256}x_{max}$, respectively. The second row of graphs contains the reference voltages of the VCO in each of the two separate modes. The third row of graphs are the tracking signals from the control logic to the target modes. A value of '1' indicates that the mode should be tracking the signal, a value of '0' indicates that the memory value should be held constant.

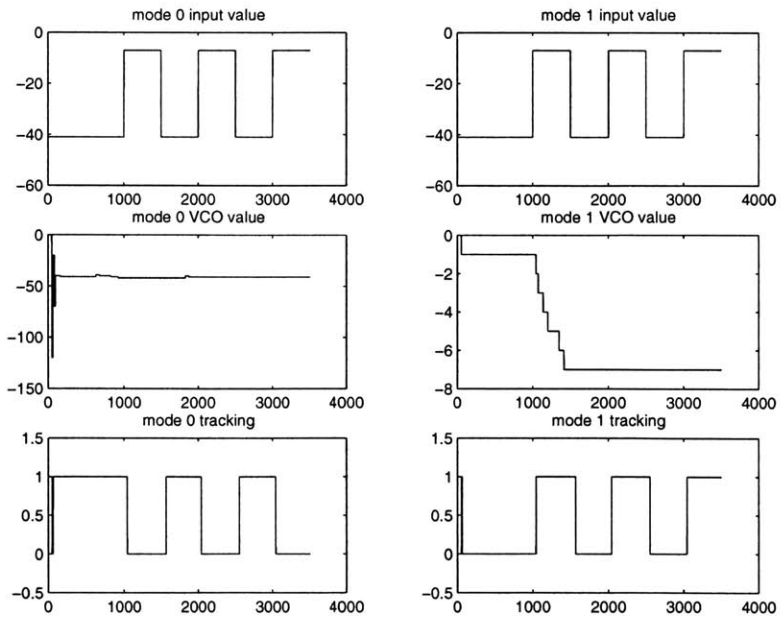


Figure 5-2: Behavior of a single pixel's multimode system in response to a bimodal oscillating input.

In the figure, there is a clear relationship between the changes in input and the

changes in the control logic outputs. Mode 0 has "learned" the frequency of the -41 signal and Mode 1 has "learned" the input of the -7 signal. As the input oscillates between these two values, the control logic is able to identify the mode whose input is closest to the input value. The control logic is also able to identify when the target pixel is active. Figure 5-3 displays the initial outputs from the control logic. The top signal is the active output from the logic, the second and fourth are the VCO reference values, and the third and fifth are the tracking signals associated with the two modes.

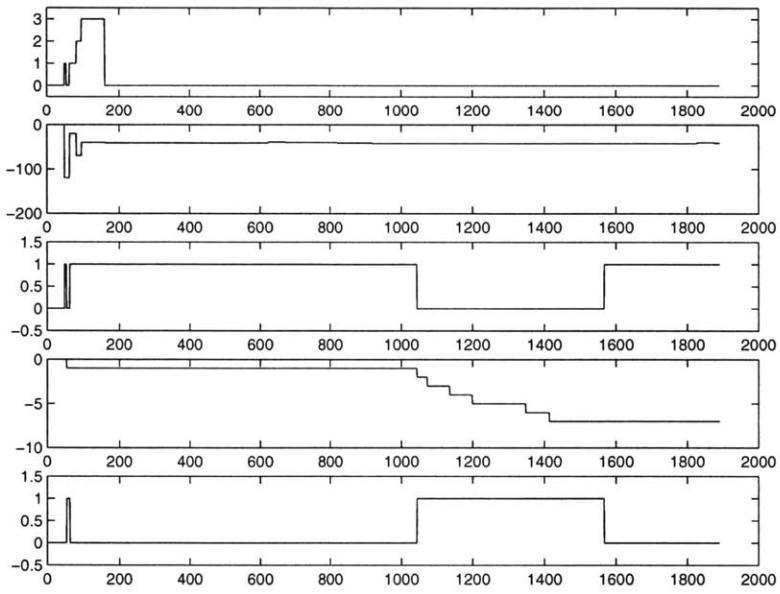


Figure 5-3: A closer look at the initial stages of the behavior of a single pixel's multimode system in response to a bimodal oscillating input.

The plot shows a period of activity only when the simulation was started. A non-zero value for the activity indicates that the pixel is active. The appearance of the second input value did not trigger the activity detection system. The difference between the second mode and the second frequency was sufficiently low for the two signals to be considered well-matched. As the fourth plot shows, during the duration of the second input, the value of the VCO was slowly improving its estimate of the input frequency. This is characteristic of the system's ability to continue tracking a signal even after it has locked into the initial frequency. This ability would allow

the system to track, without triggering activity signals, slow varying signals, such as those caused by the lighting changes during the day.

The behavior of the tracking algorithm after initialization of the first input signal shows a rapid change in the stored value of the zeroth mode. The change is a result of the graded charge pump discussed in section 4.2.3. The time required to reach a locked frequency was much faster than if a non-graded charge pump had been used. In the latter case, over 500 sample points would have been required to achieve lock. The graded charge pump was able to lock in 176 samples.

By replicating several copies of the pixel modules simulated above, it was possible to simulate the behavior of a full system of processing elements. The images in figure 5-4 taken with the 64x64 imager show the result of such a simulation. The image on the left represents the systems interpretation of the background. This is what the simulation expects to be seen by the imager. The central image is the current view from the imager. The highlighted regions designate the areas of activity detected by the simulation. Some local grouping has been done in an attempt to identify regions of activity rather than single pixels. The grouping was done using a 2-step dilation process followed by a 3-step contraction process. The dilation and contraction were done using a standard '+' template design. Consequently, the areas of activity exhibit diagonal edges rather than straight or jagged ones. The third image is the difference between the intensities of the first two images. The difference has been thresholded and the output displayed. The regions of white are those areas where the foreground and the background differ by more than the threshold and the black sections represent areas where the two intensities are relatively close.

A human figure in the image has been identified as different from the background. It is a clearly discernible form in the central picture. Once the human entered the frame, the system detected the change and initiated a storage of the file. The identification began when fewer than 24 samples from the bit stream had been processed. Due to the nature of the detection mechanism, activity is not uniformly detected. The dark portion of the human's jacket was identified first. The frequency representing that section of the outfit differed greatly from the background. Subsequently, the

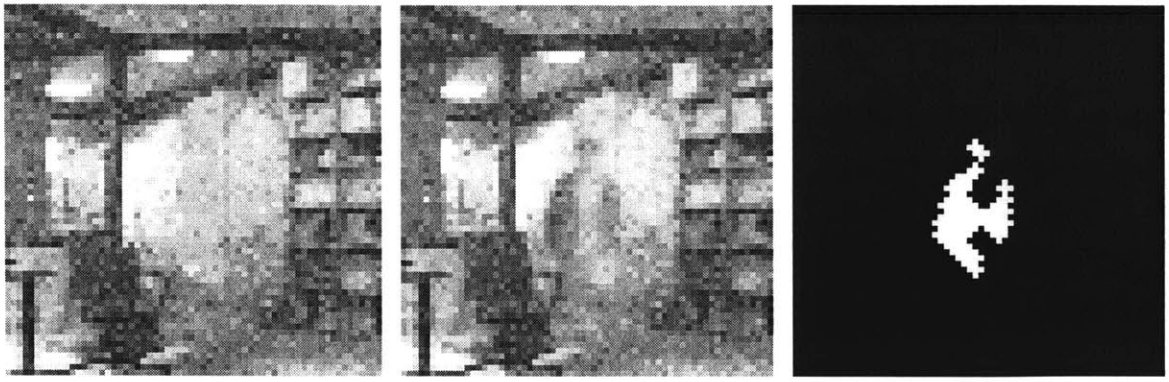


Figure 5-4: The activity detection system at work. (a) the background of the system. (b) the highlighted areas of activity, and (c) the thresholded difference between the background and the active regions.

distance between unmatched edges was very small and detection occurred rapidly. The light color of the target's shirt was not as distinguishable as the jacket. Compared to the light background, the white shirt did not represent a very large frequency difference. The result was a slower identification of activity in that area.

The activity detection system was also tested over the course of several hours. From 6pm one evening to 10am the next morning, the simulation ran with the imager overlooking a section of the lab. During those 16 hours, the simulation saved images whenever the number of active pixels exceeded a set number. Figure 5-5 contains the 64 images which were taken and the times at which they were taken. If the system had been configured to automatically store all of the images during the test - much like a VCR-based security camera, the data set would have contained 7,680 images. This larger data set would have required 120 times more storage space. The activity detection system has distilled the data to only those images when there was significant action.

Images **1-A** through **1-E** are caused by the appearance of a person in the center or the image. His presence was sufficient to trigger an activity alert. The number of pixels remained relatively high after the intruder left because the system had not gained a very good model for the background before the person appeared. Consequently, a few more frames of data were needed for the simulated pixels to tighten

Figure 5-5: A series of images taken over a 15 hour interval. These images were recorded when the system detected a high level of activity in the system.

their frequency loops and designate all of the pixels as inactive.

Images **1-G** through **2-H** highlight the movement of a member of the lab's evening cleaning staff. She, like several others in the images, is very blurry. She can be seen in image **2-B**. The blurriness is due to the speed of the test camera system. The imager gathers images by integrating the input over time - in this case, about 2 seconds. An object will appear clearly only if it is stationary for the full integration time.

Images **3-A** through **3-F** contain the faint trace of a person walking in front of the camera. Unfortunately, they were walking so quickly that the imager was not able to get a clear picture of them. At 7:58 pm, a person can be seen looking into the office. At 8:06 another figure appeared in front of the office door and remained there for two minutes. No significant activity was logged until 10:06pm. At that time, the far door was opened - note the black area in image **4-E**. In the next image, **4-F**, the light in that office is turned on. This caused a significant lighting change in the image. Consequently, the next several images were stored while the system adapted to this new background. No notable activity was detected for several hours after that. The images in row 7 show the return of the occupant of the office opening the door and leaving it ajar. Row 8 contains the blurs of people walking in front of the camera.

The majority of the images capture periods of easily recognizable activity. The remainder either contained blurred images of moving persons, or were caused by subtle, but sufficiently large, lighting changes. There is no way to determine how much activity was missed because no log was taken on a constant basis. Notably absent are any images which correspond to dawn. The slow increase in light from the rising sun did not cause the system to detect any activity.

5.3 Future Work

This work remains a theoretical possibility until it is actually implemented in silicon and its functionality tested. An obvious application of this activity detector would be in the field of security monitoring. A sensor which could identify movements and alert a security guard would result in more of the guard's attention being focused

on active regions. Additionally, an activity detector could greatly reduce the storage cost for recording a scene - needing only to capture frames when there is motion. This same concept could be beneficial in a situation where bandwidth is limited. A wireless camera, for instance, may only be able to transmit 1 Mbyte per second on its wireless connection. The maximum imager size which can transmit 24-bit colors at 30fps along this connection without compression would be 105x105 pixels. For many applications, this resolution is not sufficient. A much higher resolution imager could be used if the data transmitted were filtered to contain only sections of important, active, regions.

Extending the work beyond the base level of identifying active pixels, requires higher level systems which have a more globalized sense of the overall image. The software program may be able to identify neighboring active pixels as portions of the same object and group them together. Similarly, it may be able to reject global changes in intensity caused by a cloud passing in front of the sun, or a dimming of lights, etc.

Further research could attempt to simulate higher level features as opposed to just finding differences relative to the background. One such application would attempt to learn the standard traffic flow in a scene. Envision a security camera which monitors a park adjacent to a busy intersection. Over time, a database could be created which tabulates average activity levels in a region of the image. Using the software to group together pixels, an understanding of typical sizes of moving objects in any section can be constructed, as well as an ability to link such movements to times of day, and of the week. A good database would learn that large objects tend to move along the street areas in straight lines, human sized objects can be seen moving randomly in the park, or perpendicularly across the street. By linking these movements to times, the system could learn that during rush hour there is more traffic in the streets, and more people in the park on afternoons and weekends and none at night. Such a system would then be able to not only select out those images when there is activity, but only images when there is abnormal behavior - a car driving through the park, or several people in the park at night. This would have to be a larger processor-based operation by virtue

of its basic requirements for storage and global awareness. The exploration of various 3D architectures will continue to expand and unveil new methods for computation and new approaches for building circuits.

Appendix A

Implementation Notes for the Activity Detection System

The multimode simulation was written using a combination of C and assembly level code. The software was compiled and run on a TMS320C40 located on a White-Mountain^{TM} DSP Slalom-40 development card. Compilation was done using TI's v5.1 compiler/linker/assembler for the C40. GoDSP's Code Composer package was used for development and debug. The main loop of the program has 5 basic steps. The first step collects a single value of the input bitstream for every pixel. In the second step, these bitstream values are used to update the frequency detector and their associated output counters. The VCO's are then polled in the third step to establish their outputs. A rising edge on the VCO output would cause a state change in the frequency detector. In the fourth step, the counter for the input edges is updated. Finally, in the fifth step, the image is extracted from the various modules.

```
1>  while(TRUE) {
2>      updateCamera();
3>      for(i=0;i<CAMERA_SAMPLES;i++) {
4>          get_one_frame_of_data();
5>          update_frequency_detectors();
6>          for(j=0;j<VCO_RESOLUTION;j++) {
```

```
7>            update_VCO_values();
8>            update_frequency_detectors();
9>         }
10>        update_pixel_edge_count();
11>        display_image();
12>    }
13> }
```

Encoding the mode states. For this example, consider a 4x4 imager. Each pixel will have one memory mode associated with it. Each of these 16 modes requires 2 bits of frequency detector state, 4 bits for the two detector output counters, 2 bits for the activity flag, 2 bits for a VCO output flag, 2 bits to remember the previous input value, 2 bits for the current input and two words to represent the VCO state. To efficiently use the memory of the processor, the states of all 16 modes are packed into a single 32 bit word, `state`. Similarly, the `up` output counters are all packed into the word, `up_sum` and the `down` counter into `down_sum`. The activity flags are packed into `activity`; the VCO flags into `VCOflags`; the new input vlaues into `newInputs`; and the old input values into `oldInputs`. Thus, to extract the n^{th} modes status, one would simply multiply each of these words by $0x3 >> 2n$.

Updating the input data The only important event which comes from the input bitstream is the creation of a rising edge. If such an edge is detected, the frequency detector state should be updated and, possibly, the up output counter should be incremented. Detecting these edges is a matter of running this code:

```
risingEdges = (oldInputs ^ newInputs) & newInputs;
```

the 32 bit word risingEdges can then be used to update the value for `up_sum` with the expression:

```
up_sum = up_sum & (state & 0xaaaaaaaa) >> 1 & risingEdges  |
         up_sum & 0xaaaaaaaa |
```

```
        up_sum ^ (state & 0xaaaaaaaa) >> 1 & risingEdges  |
((up_sum & 0xaaaaaaaa) >> 1 )& up_sum;
```

This line of code increments a modes's up counter if both the frequency detector is in the 11 state and there was a rising edge on the pixel bitstream. The two bits representing the up counter sum of any of a mode will, if incremented, move from 00 $\rightarrow$ 01 $\rightarrow$ 10 $\rightarrow$ 11. Once it reaches 11, it will remain there, and will not change until the pixel is reset.

The frequency detector state can be updated by

```
state = risingEdges & 0x55555555 |
        risingEdges & state & 0x55555555) << 1 |
        state
```

A rising edge on the input will cause the state to move from 00 $\rightarrow$ 01 $\rightarrow$ 11. A rising edge on the pixel input can only cause these state changes. Once the state reaches 11, further rising edges on the pixel input will not change the state.

Voltage Controlled Oscillator Each mode requires a counter to implement the VCO. The current implementation uses a 32-bit int, but its full range is never used. Each mode also has an adjustable reference value. The call to the function update_VCO_values() on line 7 increments the counter by one for each mode in the model. If the counter value exceeds the reference value, the counter is reset and the associated `VCOflag` is toggled. If the flag is to set for the n^{th} pixel, it would be done using this code:

```
if(currMode->VCOcounter ++ > currMode->VCOreferece) {
   VCOflags |= 0x3 << 2*n;
}
```

Once all of the VCOflags have been updated, the frequency detector has its state updated and, if neccessary, down_sum integrator values are incremented. Both of these functions are accomplished using this code:

```
down_sum = down_sum & ~state & VCOflags & 0x55555555 << 1 |
           down_sum & 0xaaaaaaaa |
           (~state & VCOflags & 0x55555555) ^ down_sum  |
           ((down_sum & 0xaaaaaaaa) >> 1) & down_sum;

state = ~VCOflags & state |
        VCOflags & ((state & 0xaaaaaaaa) >> 1);

VCOflags = 0; /* reset the flags */
```

Any two bits in down_sum will, if incremented, move from $00 \rightarrow 01 \rightarrow 10 \rightarrow 11$. Once the sum reaches 11, it will not increase or decrease until the next pixel reset. The two bits of state for any mode in the system will move transition from $11 \rightarrow 01 \rightarrow 00$ if the VCOflag was set.

The number of times that the VCO is updated for every bit of the input bitstream is determined by the variable, VCO_RESOLUTION, which is defined by the user. The larger VCO_RESOLUTION is , the better the fidelity of the frequency control available to the VCO. Currently, this variable is set to 10.

updating the pixel edge count Finally, update_pixel_edge_count is called. This function will count the number of rising edges on the pixel bitstream. Only one counter is required for each pixel. If the count exceeds a reference value, a series of pixel updates are called. These updates will update the activity flag for each mode, determine which mode should be tracking the input, update the reference value of the tracking mode, and then reset the pixel's counters.

```
for(i=0;i<NUM_PIXELS;i++)  {
   if(currPixel->active)
      pixMax = ACTIVE_THRESH
   else
      pixMax = INACTIVE_THRESH
```

```
    if(currPixel->inputEdgeCounter++ > pixMax) {
        update_active(currPixel);
        update_tracking(currPixel);
        update_reference_voltages(currPixel);
        reset_pixel(currPixel);
    }
    currPixel++;
}
```

Note first that there are two pixelCount threshholds. By having a larger threshhold for active pixels, it guarantees that once a pixel is marked inactive, the frequency is sufficiently close to the input that the pixel will not be incorrectly become active again. The current values for these are 16 and 8.

update_active() updates the active flag for every mode of every pixel. Again, the flags for the 16 modes are stored in a 32 bit word, `active`. Activity is determined by this code:

```
active &= ~(currPixel->mask);      /* clear the old flag */
active |=  (  (currPixel->mask) &
              (  (~active &
                  ((down_sum | up_sum) & ALL_5s) &
                  (((down_sum  | up_sum) & ALL_As) >> 1))|
                 ( active &
                  ((((down_sum | up_sum ) & ALL_As) >> 1) |
                 ((down_sum | up_sum) & ALL_5s) )  )));
```

The above code follows the diagram in figure A-1.

updateTracking follows the diagram in figure A-2.

update_reference_voltages() will, for the tracking mode of every pixel, change the VCO reference value according to the up_sum or down_sum value. The larger the value of up_sum or down_sum the greater the magnitude of the adjustment. These magnitudes are chosen by the user. The current values can be seen in table A.

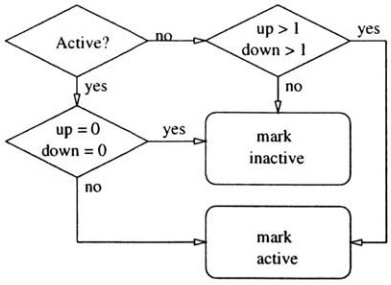


Figure A-1: Flow chart for activity detection

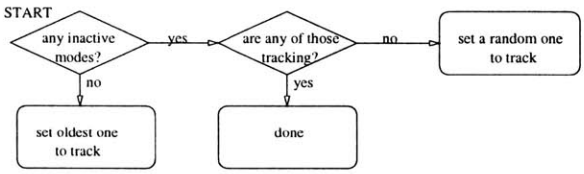


Figure A-2: Flow chart to update tracking information

up value	down value	offset magnitude
3	3	2 * VCO_RESOLUTION
2	2	VCO_RESOLUTION
1	1	1
0	0	0

Table A: reference value adjustment values

reset_pixel() sets the values of all the integrators of a pixel (the input edge counter, and the up and down counters for every mode) to zero.

Bibliography

[1] Lisa McIlrath. A low-power, low-noise, ultra-wide dynamic range CMOS imager with pixel-parallel A/D conversion. *2000 VLSI Symposium on VLSI Symposium, June 2000, Honolulu HI.*

[2] Lisa McIlrath. A robust exact method for optimal decoding of first-order $\Sigma - \Delta$ sequences. *Submitted to IEEE Transactions on Signal Processing*, 1999.

[3] Rahul Sarpeshkar. Analog versus digital: Extrapolating from electronics to neurobiology. *Neural Computation*, 10(7):1601–1638, 1998.

[4] Christopher Stauffer. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition*, 2:650–658, 1999.