Design Considerations in Systems Employing Multiple Charge Integration for the Detection of Ionizing Radiation

by Michael J. Hall, Bachelor of Science

A Thesis Submitted in Partial
Fulfillment of the Requirements
for the Master of Science Degree

Department of Electrical and Computer Engineering
in the Graduate School
Southern Illinois University Edwardsville
Edwardsville, Illinois

December 13, 2007

ABSTRACT

DESIGN CONSIDERATIONS IN SYSTEMS EMPLOYING MULTIPLE CHARGE INTEGRATION FOR THE DETECTION OF IONIZING RADIATION

by

Michael Hall

Advisor: Dr. George L. Engel

Often in nuclear physics experiments the type of incident radiation must be classified, the energy of the particle must be determined, and the position of interaction within the detector must be estimated. This thesis presents design considerations for several systems that use gated integrators to extract the above information from the pulse. Since the performance of such systems depends upon the signal-to-noise ratio (SNR) of the integrators, an analysis of the SNR characteristics of a gated integrator is presented.

A particle identification (PID) system employing pulse-shape discrimination (PSD) is highlighted. The proposed system makes use of a newly developed multi-channel integrated circuit (PSD8C). We demonstrate that the PID system will work well with both fast and slow detectors, such as organic liquids (e.g. BC 501) and CsI(Tl) scintillation detectors, respectively. For liquid scintillation detectors with a full scale energy range of 10 MeVee, simulation shows a discrimination threshold (1% error of misclassification) of 1.44 MeVee (dynamic range of 17 dB). For CsI(Tl) scintillation detectors with a full scale energy range of 100 MeV, simulation shows a discrimination threshold of 1.55 MeV (dynamic range of 36 dB).

Pulse data from an experiment using a prototype CsI(Na) detector was also analyzed and used to generate an energy spectrum. The energy spectrums for a "noise-free" and a

"noisy" (additive noise consistent with the PSD8C chip) system were compared. As these pulse data were collected with ^{137}Cs (662 keV) and ^{60}Co (1173 and 1332 keV) gamma-ray sources, the generated spectrum contained three Gaussian peaks corresponding to the photopeaks. The effect of the chip's noise is not significant. The width of the first Gaussian increased from 5.68% to 6.34%. The second increased in width from 4.37% to 4.58% while the third increased in width from 4.20% to 4.24%.

This work was initiated by the heavy-ion nuclear chemistry and physics group at Washington University in Saint Louis and is funded by NSF grant #06118996.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. George Engel for his support on this project without which I could not have progressed so far. I would also like to thank Dr. Lee Sobotka, Dr. Robert Charity, and Mr. Jon Elson, Department of Chemistry, Washington University, Saint Louis, for their help during the various stages of this project. I would like to thank my fellow researchers, especially Justin Proctor, who has worked with me and supported me in my research.

I would like to thank Dr. Scott Smith and Dr. Bob Leander, ECE faculty members at SIUE, who actively encouraged me. With their support, I was able to explore new areas. I would also like to thank Dr. Brad Noble and the rest of the ECE faculty who has supported me throughout my education at SIUE, and finally I would like to extend my appreciation to my parents, brothers, and friends whom have supported me all my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

BACKGROUND AND SIGNIFICANCE

Background

As part of a joint research effort, Washington University and Southern Illinois University Edwardsville are developing systems employing multiple charge integration for the detection of ionizing radiation. The current objective of the research is to produce a microchip capable of particle identification that will complement an existing (shaped and peak-sensing) analog chip called HINP16C (Engel et al., Nucl. Instru. Methods A573, 418-426 (2007)). Much of the work presented in this thesis was used to guide the design of this new chip, christened PSD8C (Pulse-Shape Discrimination – 8 Channels).

A detailed description of the PSD8C chip can be found in a companion thesis written by another graduate student, Justin Proctor. A brief description of the PSD8C chip is provided here. Each of the eight channels is composed of a time-to-voltage converter (TVC) with two time ranges (0.5 μsec, 2 μsec) and three sub-channels. Each of the sub-channels consists of a gated integrator with 8 programmable charging rates and a pair of externally programmable gate generators that define the start (with 4 time ranges) and width (with 4 time ranges) of the gate relative to an external discriminator signal. The chip supports 3 triggering modes.

PSD8C produces four sparsified analog pulse trains (3 integrator outputs and 1 TVC output) with synchronized addresses for off-chip digitization with a pipelined ADC. The PSD8C chip with two biasing modes occupies an area of approximately 2.8 mm x 5.7 mm and has an estimated power dissipation of 135 mW in the high-bias mode. The chip is to be fabricated in the AMIS 0.5-micron NWELL process (C5N) in early 2008. While the chip was designed to perform particle identification using pulse-shape discrimination (PSD), it

may also be used to obtain total pulse-height information. It will be able to work with many different types of detectors.

In a radiation detection system, detectors are used to sense radiation. When struck by a particle, a detector will produce a pulse. Encoded in the pulse-shape is information about the incident radiation including the pulse-height, particle type, and/or position of interaction within the detector. While the parameters may be extracted in many different ways, this thesis analyzes analog-based systems that employ multiple charge integration to extract the information. This method is not only effective but produces systems that are small (and relatively speaking, inexpensive). Alternative DSP-based approaches can claim neither of these advantages.

In the remainder of this chapter, three examples of systems that employ multiple charge integration will be described. The three systems are intended for use in three different types of applications:

1. Applications requiring total pulse-height information
2. Applications requiring particle identification using pulse-shape discrimination
3. Applications requiring information regarding position within detector where a particle deposited its energy

The first two of these systems will be analyzed, and the performance, when implemented using the PSD8C IC, will be presented. The third, since it has not been fully developed will only be discussed briefly here in Chapter 1. Before discussing these three systems, some additional background material explaining how detectors and light sensors work will be provided. It is hoped that this information will make it easier for the reader to understand the models presented in Chapter 2 of this thesis.

Detectors

Scintillation detectors are commonly used in the detection of ionizing radiation. They work by converting a fraction of the deposited energy into visible light. The detector is coupled to either a photomultiplier tube or a photodiode. Most of the energy deposited in a scintillator ends up being degraded into vibrational phonons in the crystal, *i.e.* the bulk of the energy is converted to heat. However, a fraction (~ 10% in bright scintillators) is converted into visible photons. The total number of these visible photons (total time integral) is proportional to the energy deposition in the scintillator. In many crystals, the time dependence of the pulse shape depends on the type of ionizing radiation and thus a pulse-shape analysis yields the particle type (gamma-ray, electron, proton, alpha-particle, *etc.*).

There are two different types of scintillating material: organic and inorganic. Organic scintillators are either liquid or solid (either crystal or amorphous plastic for the latter) and can take on many different shapes and sizes. One of the main advantages of organic scintillators is that they have fast decay times. This means that the time of interaction can be determined with better resolution. Inorganic scintillators are always crystals. They can have a very high light yield, and long, short, or a mixture of decay times. Three of the most common inorganic scintillators are NaI(Tl), CsI(Tl), and CsI(Na) (Knoll 221-222, 235).

The light output of many scintillators is nonlinear in the amount of light they produce per unit energy. As Birks first suggested, this is primarily due to quenching which degrades the light output (Knoll 227). It is therefore convenient to measure the light output (or pulse-height) in the equivalent pulse-height that electrons would produce, the electron equivalent energy (*i.e.* MeVee).

Light sensors

The light from a detector is converted to an electric charge using a light sensor. Two light sensors that are typically used are a photomultiplier tube and a photodiode. A photomultiplier tube (PMT) consists of a photocathode followed by a cascade of dynodes and terminated by an anode. When a photon of light enters the photocathode, it may interact to produce an electron. The probability of converting the visible photon into an electron at the photocathode is known as the quantum efficiency.

Photomultiplier tubes typically have low quantum efficiency, in the range of 20%. This low quantum efficiency degrades the energy resolution which is ultimately determined by counting (Poisson) statistics of the primary photoelectrons generated at the cathode. These primary electrons then go through the cascade of dynodes which will multiply them. When an electron hits a dynode, several more electrons are produced. Since there is a cascade of dynodes, the multiplication occurs many times giving several orders of magnitude of gain. This will occur in a highly linear fashion (if the dynode string is "stiff", *i.e.* the transient charge is small compared to the stored charge on the capacitors attached to each dynode). The resulting electrons are then collected at the anode terminal. The photomultiplier tube is an exceedingly low-noise device which is one of its main advantages (Knoll 265-273).

A photodiode is a semiconducting device with a p-n junction that is sensitive to light. When photons strike the p-n junction, it will then generate current (Streetman and Banerjee 398). The classic photodiode has high quantum efficiency but no electron multiplication. A separate charge-sensitive amplifier must be used to generate a detector signal. The main advantage of photodiodes is that they are smaller than PMT's, an advantage which is some-

times paramount. As CSA's are generally far noisier than a PMT, photodiodes generally do not perform as well as a PMT (Knoll 287-288).

Example Systems That Employ Multiple Charge Integration

Systems employing multiple charge integration can be used for many different applications. Some applications may need to get total pulse-height information whereas others may require particle identification (PID) or information regarding the position within the detector where the energy was deposited. Three example systems will be discussed.

The first system used multiple charge integration to get total pulse-height information and to tag pile-up. The second system used pulse-shape discrimination (PSD) to identify the type of particle as well as the deposited energy. The third system will use (what we like to refer to as) an analog-assisted DSP (AA-DSP) technique to gather more detailed pulse-shape information. In the later case, when the data originates from a large volume solid-state Ge detector, the shape analysis will yield the position of interaction in the detector as well as deposited energy and timing information. Most of the attention in this thesis will be directed at the first two systems, since at this point in time the third system has not yet been fully developed.

Total pulse-height information using the high resolution scintillation array (HiRSA)

Multiple charge integration can be used to get total pulse-height information and to detect and tag pile-up. A single integrator can be used to extract energy information by integrating over the entire (or a large percentage) of the pulse. If a second integrator is used to integrate over a different region, then the ratio of the two integrators can be used as a means to detect pile-up. Simulations were done on real digitized pulse waveforms from a

CsI(Na) detector which is the prototype for a new detector project at the National Superconducting Cyclotron Laboratory (NSCL) at Michigan State[1]. In this section, I describe the reason for this new detector system, the working elements, and provide a few words about alternative electronics for this application.

Much of present day nuclear physics concerns the structure of nuclei removed from beta-stability. The study of the properties of such nuclei requires reactions to be preformed with them. The NSCL is a facility devoted to creating beams of these unstable nuclei. What this really means is all experiments are secondary beam experiments. A primary stable beam slammed into a stable target. One or more of unstable products of this primary reaction are selected to make a secondary beam which is used, together with a secondary target, to initiate a second reaction. While the nuclei in the secondary beam are unstable, they have lifetimes longer than 1 ms. This is plenty of time for electrostatic means of generating a beam (by say selection on magnetic rigidity and velocity or specific energy loss) which works on times scales of less than 1 μs.

The simplest secondary reaction would be a "Coulomb excitation" reaction. In this process the unstable nuclei are excited by the time varying Coulomb Field as it flies closely by a nucleus of an atom in the secondary target. Once excited, it will decay by emitting gamma-rays, the energies of which tell you about the excited quantum states of the unstable nucleus. To make this experiment work, one needs a gamma-ray detector with high efficiency (as the secondary beams are generally very low in intensity) and with as high resolution as possible. The first such array at the NSCL was provided by Washington University. This was a set of long cylindrical NaI(Tl) detectors salvaged from an decommissioned PET. The second, was a larger array of NaI(Tl) from Argonne National Laboratory. The third was

[1] This project is being lead by Dr. Dirk Weisshaar and Prof. Alenandra Gade at the NSCL. We thank them for the files containing the source data in the form of digitized wave forms.

a large array of solid state Ge detectors. The Ge detectors have outstanding energy resolution but very poor efficiency. In an effort to complement the Ge array, the staff at the NSCL proposed and were approved to build a 192 element CsI(Na) scintillation array. The (tentative) name of this device is: High Resolution Scintillation array (HiRSA).

CsI(Na) is a scintillator with a brightness (~ 40 visible photons per keV deposited) very similar to NaI(Tl) but with a better match to peak quantum efficiency of photocathodes used in PMT's. The channel count (~ 200) is sufficiently large to warrant careful consideration of cost of the electronics per channel. However, one must not sacrifice energy resolution and should maintain the ability to identify pile-up events.

The detectors in the array are either square (3"x3"x3") or rectangular (2"x2"x4") with either the 3"x3" or 2"x2" face attached to the PMT. By mid 2007, the NSCL had a prototype and had conducted source tests with both digital and conventional electronics. For testing the DSP, they collected (and stored) digitized wave forms (10 ns intervals) from one of the prototypes. The source tests used weak ^{137}Cs (662 keV) and ^{60}Co (1173 and 1332 keV) gamma-ray sources. Weak sources were used to minimize pile-up for these first tests. Due to the fact that weak sources were used, natural background gammas from ^{40}K (1460 keV) and ^{226}Ra (609 keV) were detected.

While the NSCL was funded (by the NSF) for the physical device, they were not given any funds for the electronics. The initial cost estimate for a fully digital system was close to 200 k$ (~ 900 $/ch). On top of this, FPGA development would need to be done. The full cost of implementing our PDS system (post development) was estimated to be only 30 k$. This factor of 7 reduction (from DSP to chip analog – with some signal shape analysis) is similar to what we have found in other applications and a major reason why this chip development was undertaken. (Keep in mind that the development of the DSP is very large

and also largely borne by the federal government in the form of grants to universities, national laboratories and private companies.)

Particle identification using pulse-shape discrimination (PSD)

Particle identification of nuclear radiation is typically done using pulse-shape discrimination (PSD). There are several ways in which this can be accomplished:

1. "Sensing differences in the decay times of pulses"
2. "Integrating pulse charge over different time intervals"
3. "Digital capture and shape analysis of pulses"

The first method works by filtering the pulse through a shaper to produce a "bipolar pulse". The shape of this bipolar pulse and its zero-crossing depends on the pulse shape and decay time. Since different particles will have different pulse shapes and decay times, the time difference between the zero-cross and the start of the pulse will change depending on the type of particle that struck the detector (Bryan et al. 1).

The second method works by integrating the pulse over various time intervals. The results of the integrations depend on both the deposited energy and the particle type. To discriminate between particles, the integration needs to be normalized by looking at the ratio between two integrators. There is usually a fast and slow component in the pulse waveform that depends on both the deposited energy and the particle type. Therefore, if one were to capture these two components and look at their ratio (which is actually a function of the deposited energy), then the particle type can be determined (Bryan et al. 1).

The third method works by using high-speed analog-to-digital flash converters to digitize each pulse and then do a complete "shape analysis". The shape analysis is done "off-

line [using a computer rather than in an analog fashion] to determine particle type, energy, and timing information" (Bryan et al. 1-2).

Although the main purpose of pulse-shape discrimination (PSD) is for particle identification, we will also show that this pulse-shape information can be used to tag and perhaps correct pile-up events. Typically PSD is done using a detector, a constant-fraction discriminator, and multiple gated integrators. Crucial to PSD is the photodetector's electronics noise. A good detector should have a high light yield and very little electronic noise. These conditions minimize the detection threshold for discriminating between particles. One way to determine the effectiveness of a PSD system is to calculate a figure-of-merit (Marisaldi et al. 1917-1919):

$$D = \frac{\Delta r}{\sigma_{tot}} \tag{1.1}$$

In equation 1.1, r is the ratio between two integrators, Δr is the difference in the ratios for two particles and σ_{tot} is the square root of the sum of the noise variances of the r distributions for the two particles. The figure-of-merit allows us to judge the PSD system. A high figure-of-merit will allow better particle discrimination. There are two ways to increase the figure-of-merit. One is to make Δr larger and the other is to decrease the noise is both r distributions. This can be done by picking good integration regions and integration time constants. Noise can also be improved by picking a good detector with a high light yield, and by reducing the electronics noise of the system.

Positional information using analog-assisted DSP (AA-DSP)

A future system is being planned that will get the position of interaction in the detector using an analog-assisted DSP approach. Analog-assisted DSP is a mix between charge

integration and traditional digital capture methods. It utilizes multiple analog "round-robin" charge integrators to integrate under each region of a pulse. Then using digital signal processing, the position of interaction in the detector, deposited energy, and timing information can be determined. In order to characterize a pulse, there will be, for example, 12 fast charge integrators and 12 slow charge integrators. The fast charge integrators are used to capture the fast rising portion of the pulse whereas the slow charge integrators are used to capture the slow decaying portion of the pulse. An illustration showing the 12 slow charge integrators can be seen in figure 1.1.

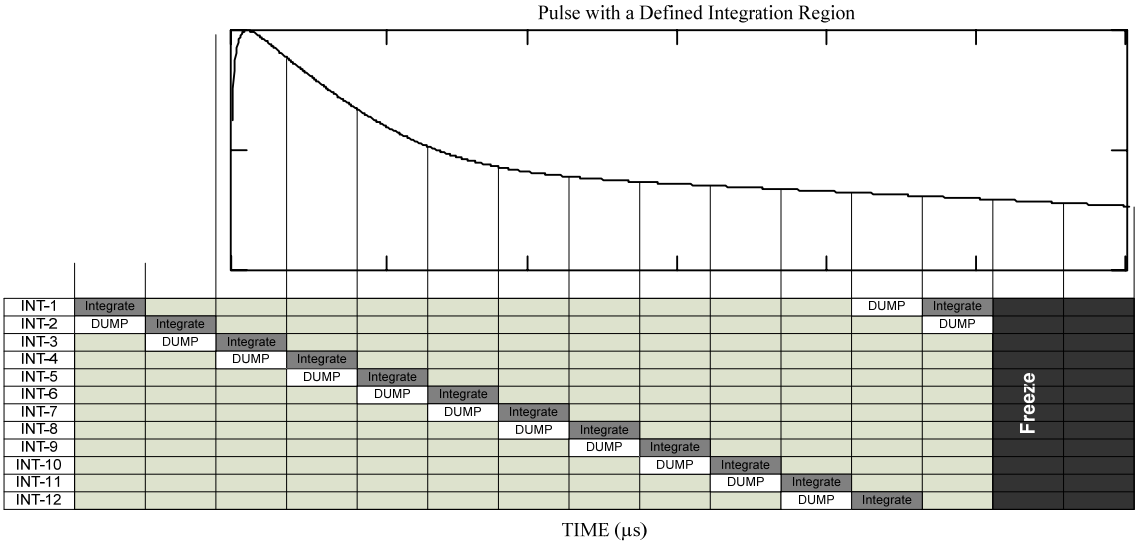

Figure 1.1 Round-robin charge integrators used in AA-DSP

Since this is a round-robin scheme, the 12 fast and slow charge integrators are continuously running. However, when a new pulse arrives, a counter is started which will stop the round-robin integrators after 10 integrations have occurred past the arrival of the pulse. The values of these integrators are then held until retrieved for off-line analysis.

Scope of Thesis

The objective of this thesis is to analyze the performance of several systems that employ multiple charge integration in the detection and analysis of ionizing radiation. Predicting the performance of such systems requires a careful characterization of the integrator and its associated noise sources. Performance depends on the number of gated integrators employed, the integration regions selected, and the SNRs of the integrators.

There are 5 chapters in this thesis. Chapter 2 will characterize an analog integrator and will derive noise equations that predict the effects of noise at the output. Chapter 3 will discuss the performance of pulse-shape discrimination using the probability of error (probability of misclassifying the particle) and the figure-of-merit (FOM) as criteria for judging overall performance. Chapter 4 will present simulation results using pulse models for CsI(Tl) and liquid scintillation detectors. It will also show simulations on real digitized pulse waveforms from a CsI(Na) detector and give the proposed implementation of the PSD chip for this application. Finally, Chapter 5 will give the conclusion and discuss the future direction of this research work.

CHAPTER 2

DERIVATION OF INTEGRATOR OUTPUT NOISE

Modeling the System

In this chapter, the system up to the integrator output will be characterized. The model shown in figure 2.1 illustrates each component of the system. Once modeled, noise sources can be identified and the contribution of each noise source at the output predicted.

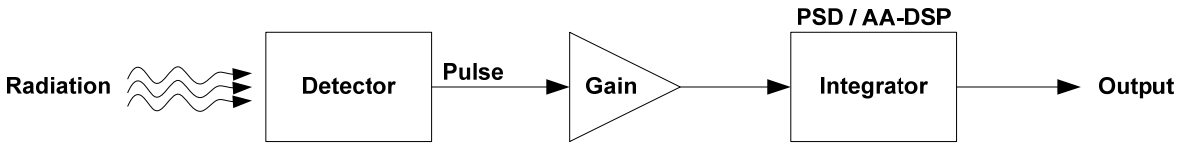

Figure 2.1 System model

In figure 2.1, radiation strikes a detector which produces a pulse. For modeling purposes, we define this pulse in terms of electron volts per unit time. Then through a series of energy conversions in the detector, the pulse is converted into equivalent electrons. The pulse then goes through a series of gain stages in which it is converted from a current at the input to a voltage at the output and amplified in magnitude. This overall gain is characterized as a transresistive gain. Finally, the pulse is integrated over a specified region and a voltage is presented at the output of the integrator.

Detector

In the detector, a particle undergoes a number of energy conversions that incur some kind of loss. When a particle strikes the detector, it produces visible photons and incurs a conversion efficiency loss. Then it undergoes visible light collection, and photocathode quantum efficiency losses. The photocathode quantum efficiency is the probability of an electron being ejected when the photocathode interacts with a photon. This in turn gives the

number of electrons proportional to the energy of the incident particle. This energy conversion, described by k_{det}, is given in equation 2.1. The detector parameters are summarized in table 2.1.

E_{rad}	Energy of incident radiation
E_{vis}	Energy of visible photon radiation
ε_{con}	Conversion efficiency
ε_{coll}	Visible light collection efficiency
ε_{q}	Photocathode quantum efficiency

Table 2.1 Detector parameter descriptions

$$k_{\det} = \frac{\varepsilon_{con} \cdot \varepsilon_{coll} \cdot \varepsilon_{q}}{E_{vis}} \tag{2.1}$$

The energy of incident radiation that was deposited in the detector can be related to the equivalent number of electrons that were produced by multiplying k_{det} by the energy, E_{rad}, and then converted to charge by multiplying by q (the charge of an electron) where $q = 1.602 \cdot 10^{-19}$ C.

$$Q = E_{rad} \cdot k_{\det} \cdot q \tag{2.2}$$

Pulse

A pulse is produced by the detector when a particle collides with the detector. Depending on particle type, a different pulse shape is observed as explained in Chapter 1. The resulting pulse is best modeled by using a sum of decaying exponentials. Many pulses can be modeled using just two exponential terms, but for completeness, the model is defined for k exponentials with both rising and falling time constants. The resulting pulse is described by

equation 2.3. It is defined such that the integration of $p(E,t)$ from time t=0 to t=∞ is the energy E.

$$p(E,t) = E \cdot \frac{1}{\sum_{i=1}^{k} A_i} \cdot \sum_{i=1}^{k} \left(\frac{A_1}{\tau_{F,i} - \tau_{R,i}} \cdot \left(e^{-\frac{t}{\tau_{F,i}}} - e^{-\frac{t}{\tau_{R,i}}} \right) \right) \quad \text{for } k \text{ exponentials} \tag{2.3}$$

The pulse is characterized by the rising and falling time constants (the τ_{Fi} and τ_{Ri} terms) and the weight (the A_i terms) on each exponential. An example of a two-component exponential pulse is shown in figure 2.2 with a log scale on the y-axis. The exponential components can be seen as the two falling slopes on the pulse which illustrates a fast and slow component.

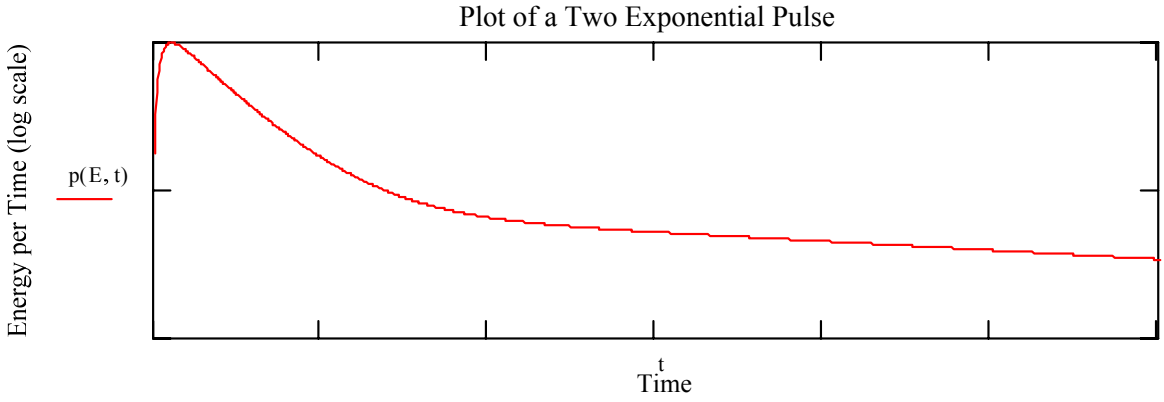

Figure 2.2 Two exponential pulse plot using the defined model

The advantage of using this pulse model is the fact that it is able to represent not only the falling decay of the pulse but also the rise time. Since it uses sums of exponentials to represent a pulse, one can exploit the principle of linear superposition when analyzing the system. Later in this chapter, the area under the pulse described by equation 2.3 over a specified region will be needed. This integral will be important in several of the noise analyses which will be presented later. Integrating the pulse $p(E,t)$ from t_1 to t_2 gives the following expression:

$$P(E,t_1,t_2) = E \cdot \frac{1}{\sum_{i=1}^{k} A_i} \cdot \sum_{i=1}^{k} \left(\frac{A_i}{\tau_{F,i} - \tau_{R,i}} \cdot \left(\tau_{F,i} \cdot \left(e^{-\frac{t_1}{\tau_{F,i}}} - e^{-\frac{t_2}{\tau_{F,i}}} \right) - \tau_{R,i} \cdot \left(e^{-\frac{t_1}{\tau_{R,i}}} - e^{-\frac{t_2}{\tau_{R,i}}} \right) \right) \right) \quad (2.4)$$

Gain stage

In order to count the number of electrons produced by the detector (since the number is generally quite small), the number of electrons must be amplified. Also, the resulting charge packet must be converted to a voltage. One way to do this is to use a photomultiplier tube (PMT) which will multiply the number of electrons produced by the detector by several orders of magnitude. Passing these electrons through a terminating resistance, *RT*, (typically 50 Ω), will produce a voltage which may be further amplified using a voltage amplifier with gain A_v. All of these gain stages can be combined into a single transresistive gain parameter, Ar_{gain}. This scenario is illustrated in figure 2.3.

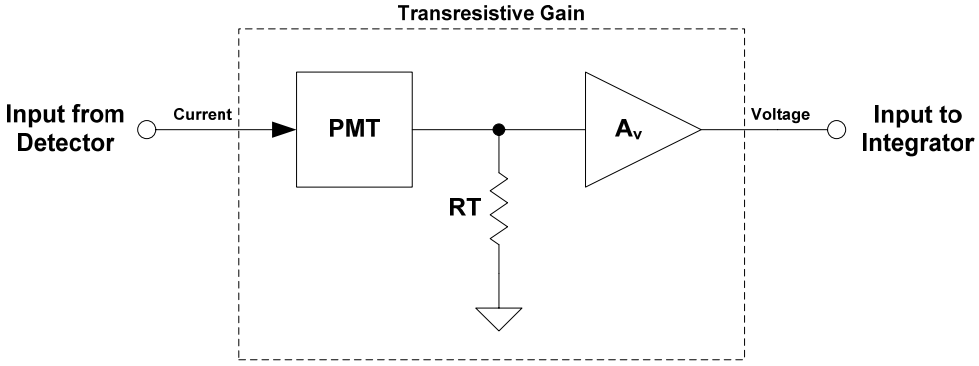

Figure 2.3 Gain stage after the detector

The transresistive gain can be calculated by multiplying the gain of each stage. This results in the following equation:

$$Ar_{GAIN} = PMT_{GAIN} \cdot RT \cdot A_V \quad (2.5)$$

The transresistive gain sets the range of the system. Therefore, it is important to be able to calculate this parameter. In order to do this, the energy of the incident particle must be related to the voltage that is seen at the input of the integrator:

$$V_{IN}(t) = p(E_{rad}, t) \cdot k_{\det} \cdot q \cdot Ar_{GAIN} \tag{2.6}$$

By applying a max function to this equation and solving for the transresistive gain, it will allow us to select an appropriate gain such that the maximum energy of incident radiation will produce a pulse with a peak voltage within the specification of the integrator circuit. The result is shown below:

$$Ar_{GAIN} = \frac{V_{IN,\max}}{\max(p(E_{\max}, t)) \cdot k_{\det} \cdot q} \tag{2.7}$$

This equation works well for one particle, but if there are several different particles in an experiment, then it must be slightly modified. In the case of multiple particles, each with a different pulse shape, one must select the maximum peak between all of the pulses for the maximum energy of incident radiation. The revised equation is shown below:

$$Ar_{GAIN} = \frac{V_{IN,\max}}{\max(\max(p_1(E_{\max}, t)), \max(p_2(E_{\max}, t)), \ldots) \cdot k_{\det} \cdot q} \tag{2.8}$$

Once the transresistive gain has been determined using equation 2.8, then the appropriate photomultiplier tube, terminating resistance, and voltage gain in equation 2.5 can be determined.

Integrator

Each channel on PSD8C consists of three gated integrators. The integrator can be implemented in different ways, but the design that has been chosen in the PSD8C integrated

circuit is presented in figure 2.4. In this design, the integrator consists of an operational amplifier, a resistor *R*, and a capacitor *C*.

In order to create a gated integrator, the switch, INT-x, was added to control the integration. On-chip gate generation circuits (not shown) are used to control when the integration starts and finishes. Before the particle arrives, the DUMP switch is asserted to remove charge from the integrator capacitor. Depending on the detector type, and the integration region selected, the signal may be very large or small so the charging rate must be programmable. Therefore, the time constant of the integrator, τ_{int}, is controlled by selecting a different resistor in the array. Lastly, a DAC was added to account for offset in the op amp and in any earlier electronics.

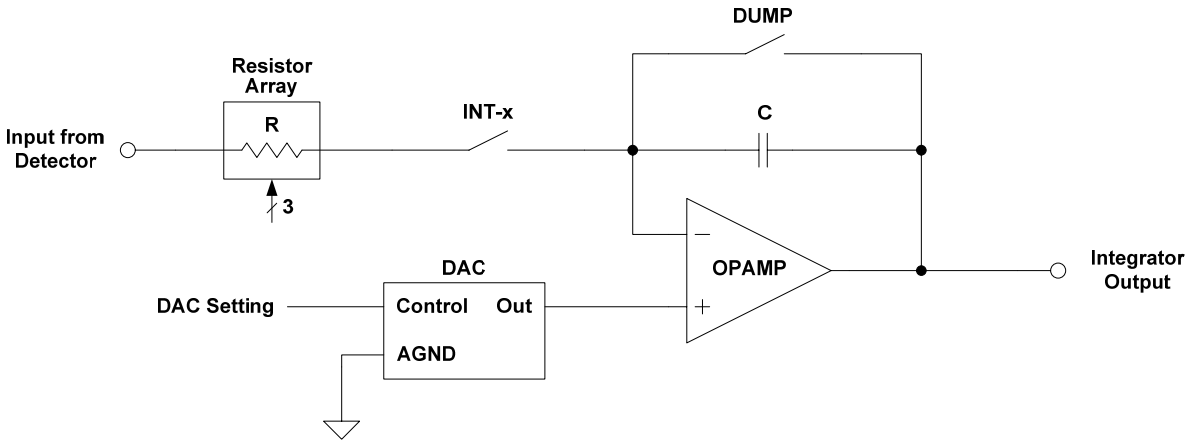

Figure 2.4 Integrator used in PSD8C design

In order to more easily analyze the circuit, a simpler model which consists of the resistor, capacitor, and the op amp will be used. In order to completely characterize the integrator, the positive input to the op amp will be left as a variable. There are two reasons for this. First, the thermal noise of the op amp can be modeled as a voltage source at the positive terminal of the op amp. Second, the offset in the op amp can also be modeled at the positive terminal. The simple integrator model is shown in figure 2.5.

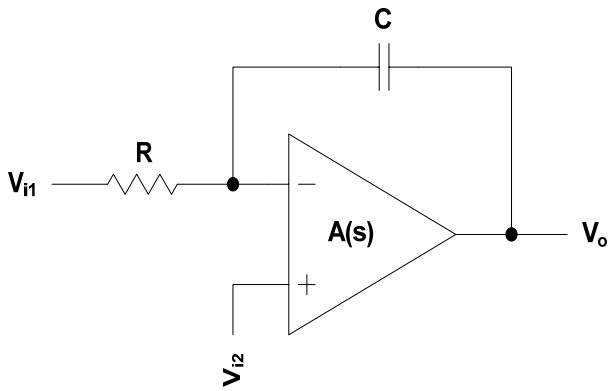

Figure 2.5 Simple integrator model

A general expression describing the integrator transfer function is given in equation 2.9. This equation shows the individual contributions of V_{i1} and V_{i2} at the output node and incorporates the transfer function of the op amp, $A(s)$.

$$V_o(s) = -\frac{\frac{1}{\tau_{\text{int}} \cdot s}}{1 + \frac{1}{A(s)} \cdot \left(1 + \frac{1}{\tau_{\text{int}} \cdot s}\right)} \cdot V_{i1} + \frac{1 + \frac{1}{\tau_{\text{int}} \cdot s}}{1 + \frac{1}{A(s)} \cdot \left(1 + \frac{1}{\tau_{\text{int}} \cdot s}\right)} \cdot V_{i2} \tag{2.9}$$

If the op amp is ideal, then $A(s)$ would go to infinity leaving a one in the denominator and giving the resulting equation:

$$V_o(s) = -\frac{1}{\tau_{\text{int}} \cdot s} \cdot V_{i1} + \left(1 + \frac{1}{\tau_{\text{int}} \cdot s}\right) \cdot V_{i2} \tag{2.10}$$

However, since the op amp is non-ideal, a better model is needed. A real op amp is characterized by an open-loop gain, a dominate pole, and several parasitic poles. It is sufficient to model the open-loop gain, A_0, and the dominate pole, ω_d, of the op amp as shown in equation 2.11. The dominate pole is related to the amplifier's gain-bandwidth product (GBW) where $\omega_d = 2\pi \cdot GBW / A_0$.

$$A(s) = \frac{A_0}{1 + \frac{s}{\omega_d}} \tag{2.11}$$

For the integrator circuit, modeling the bandwidth of the op amp is important since insufficient bandwidth will result in attenuation of the signal at the output. The effects of finite gain in the op amp, however, can be neglected since there is practically zero contribution to the output (this will be proved later through simulation). This allows us to approximate *A(s)* with a simpler model shown in equation 2.12. In this equation, ω_u is the unity gain angular frequency and is equal to $2\pi \cdot GBW$.

$$A(s) \approx \frac{\omega_u}{s} \tag{2.12}$$

Substituting this op amp model into equation 2.9 gives the result seen in equation 2.13 where $\tau_o = \tau_{int} \,||\, \tau_u$, τ_{int} is the time constant of the integrator, and τ_u is $1/\omega_u$:

$$V_o(s) = -\frac{1}{\tau_{\text{int}} \cdot s} \frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u} \frac{1}{1 + s \cdot \tau_o} V_{i1} + \left(1 + \frac{1}{\tau_{\text{int}} \cdot s}\right) \frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u} \frac{1}{1 + s \cdot \tau_o} V_{i2} \tag{2.13}$$

This equation shows that passing an input through a non-ideal integrator is equivalent to low-pass filtering the input, applying a gain factor, and then passing it through an ideal integrator. This equivalent model is illustrated in figure 2.6.

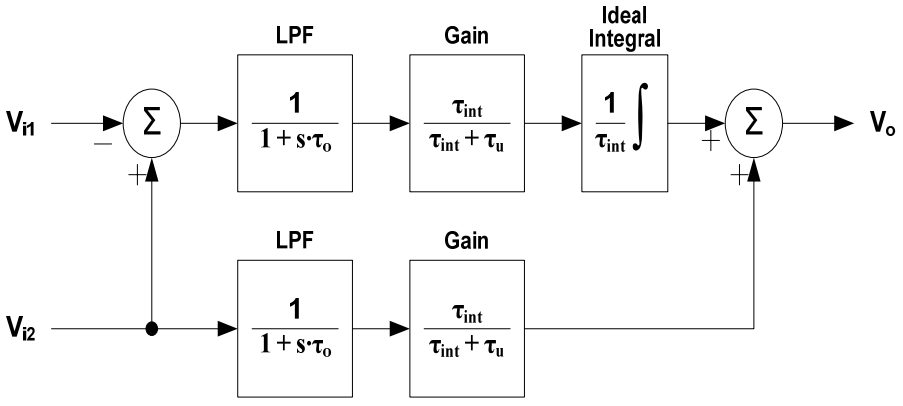

Figure 2.6 Block diagram illustrating bandwidth-limited integrator model

For a good approximation, only the gain term and the ideal integrator are needed. This allows us to simplify the model. An expression for the output voltage is as follows:

$$V_o(t_1,t_2) = -\frac{1}{\tau_{\text{int}} + \tau_u}\int_{t_1}^{t_2} V_{i1}(t)dt + \frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u}\left(\frac{1}{\tau_{\text{int}}}\int_{t_1}^{t_2} V_{i2}(t)dt + V_{i2}(t)\right) \tag{2.14}$$

Furthermore, the $V_{i1}(t)$ term can be replaced with the input voltage, $V_{IN}(t)$ from equation 2.6, and the integration of the pulse, *p(E,t)*, can be replaced with $P(E,t_1,t_2)$.

$$V_o(t_1,t_2) = -\frac{P(E_{rad},t_1,t_2)\cdot k_{\text{det}}\cdot q\cdot Ar_{GAIN}}{\tau_{\text{int}} + \tau_u} + \frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u}\left(\frac{1}{\tau_{\text{int}}}\int_{t_1}^{t_2} V_{i2}(t)dt + V_{i2}(t)\right) \tag{2.15}$$

This equation is very important for characterizing the integrator circuit and will be used in further calculations to predict noise in the system, the signal-to-noise ratio (SNR), and ultimately, the performance of the entire system.

Noise Sources

Inherent in any system is noise which may come from many places. It is important to understand what type of noise exists in the system, where it is coming from, and how much noise is present. This knowledge will allow a designer to make intelligent trade-offs when attempting to reduce noise and improve performance. The effort to reduce the noise comes at the expense of area, cost, and other factors.

For the gated integrator of figure 2.4, we have identified several noise sources as shown in figure 2.7. First, there is the Poisson noise created by the random arrival of electrons proportional to the energy of incident radiation that was deposited in the detector. Second, there is electronics noise, and third, there is the quantization noise of the analog-to-digital converter (ADC) which is used to digitize the analog voltage at the output of the gated integrator.

The electronics noise can be classified as follows: jitter-induced noise, thermal noise of the integrating resistor, thermal and 1/f noise of the op amp sampled on to the integrating

capacitor, and the continuous input-referred thermal and 1/f noise of the op amp. Jitter can be separated into two contributing sources. One is the jitter-induced noise associated with the external constant-fraction discriminator (CFD) which provides a signal to tell us that a pulse is on its way, and the other is the jitter-induced noise associated with PSD8C's on-chip gate generators that set the integration region.

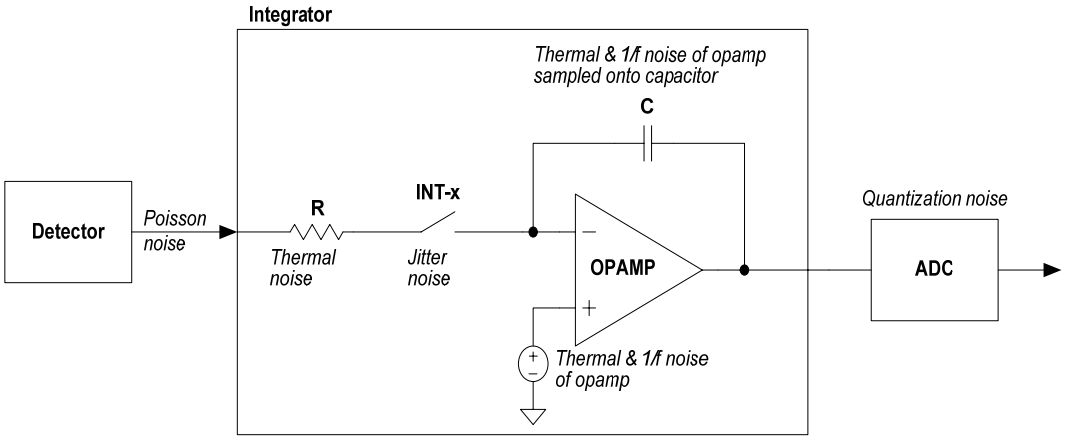

Figure 2.7 Noise sources identified in the system

Poisson noise

The first type of noise considered is Poisson noise. This noise is a result of the random arrival of electrons proportional to the energy of incident radiation that struck the detector. If the arrival of electrons is discretized, it can be observed that there are an average number of electrons that arrive over a time T_s. The smaller T_s becomes, the fewer number of electrons will arrive. Integrating from t_1 to t_2 will give the total number of electrons, N_e, over that range.

Since the variance of a Poisson random variable is equal to its expected value (Yates and Goodman 80), the variance of the discrete electrons is also N_e. If a factor, K, is defined

as the parameter that relates the number of electrons to the equivalent voltage at the output of the integrator, then the variance at the output can be determined as follows:

$$\begin{aligned} V_{OUT} &= K \cdot Ne \\ Var[V_{OUT}] &= Var[K \cdot Ne] \\ &= K^2 \cdot Ne \\ &= K^2 \cdot \frac{|V_{OUT}|}{K} \\ Var[V_{OUT}] &= K \cdot |V_{OUT}| \\ \sigma_p{}^2 &= K \cdot |V_{OUT}| \end{aligned} \tag{2.16}$$

Using the equations from the system model presented earlier, the factor, K, can be determined as follows:

$$K = \frac{q \cdot Ar_{GAIN}}{\tau_{\text{int}} + \tau_u} \tag{2.17}$$

This gives the following complete expression for the Poisson noise at the output:

$$\sigma_p{}^2 = \frac{q \cdot Ar_{GAIN}}{\tau_{INT} + \tau_u} \cdot |V_{OUT}| \tag{2.18}$$

Jitter

Another form of noise that affects the system is jitter-induced. This type of noise is caused by an uncertainty in when a pulse is integrated (see figure 2.8). The integration region of a pulse is defined by two parameters: the delay or initial time (T_i), and the width or integration period (T). By integrating sooner or later than expected, the voltage at output will be larger or smaller. To determine the effect of jitter at the output, the simple case of one exponential with both rising and falling time constants must first be analyzed:

$$f_S(t) = \frac{A}{\tau_F - \tau_R} \cdot \left(e^{-\frac{t}{\tau_F}} - e^{-\frac{t}{\tau_R}} \right) \tag{2.19}$$

The pulse of equation 2.19 must be integrated from a start time, T_i, for a period, T, to get the equation at the output of an integrator:

$$\begin{aligned} f_{OUT,S} &= \int_{Ti}^{Ti+T} f_S(t)dt \\ &= \frac{A}{\tau_F - \tau_R} \cdot \left[\int_{Ti}^{Ti+T} e^{-\frac{t}{\tau_F}} dt - \int_{Ti}^{Ti+T} e^{-\frac{t}{\tau_R}} dt \right] \\ &= \frac{A}{\tau_F - \tau_R} \cdot \left[\tau_F \cdot e^{-\frac{Ti}{\tau_F}} \cdot \left(1 - e^{-\frac{T}{\tau_F}} \right) - \tau_R \cdot e^{-\frac{Ti}{\tau_R}} \cdot \left(1 - e^{-\frac{T}{\tau_R}} \right) \right] \end{aligned} \tag{2.20}$$

Pulse with a Defined Integration Region

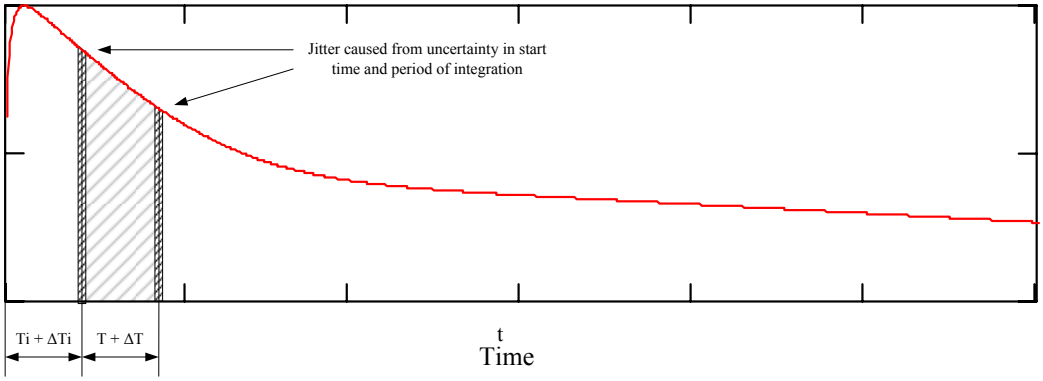

Figure 2.8 Pulse showing the jitter noise in the defined integration region

There are two contributions of jitter to the output: the uncertainty in T_i, and the uncertainty in T. To analyze the effects of jitter, these two uncertainties in time must be analyzed separately. We begin by assuming that the integration start time, T_i, has some uncertainty:

$$\text{Let } Ti = Ti' + \Delta Ti$$

$$f_{OUT,S} = \frac{A}{\tau_F - \tau_R} \cdot \left[\tau_F \cdot e^{-\frac{Ti'+\Delta Ti}{\tau_F}} \cdot \left(1 - e^{-\frac{T}{\tau_F}} \right) - \tau_R \cdot e^{-\frac{Ti'+\Delta Ti}{\tau_R}} \cdot \left(1 - e^{-\frac{T}{\tau_R}} \right) \right] \tag{2.21}$$

Using a Taylor expansion, the ΔT_i term can be simplified into a linear expression. It can also be assumed that ΔT_i is centered about 0:

$$e^{-\frac{t}{\tau}} \approx 1-\frac{t}{\tau} \quad \text{at t = 0} \quad \text{(taylor expansion)} \tag{2.22}$$

$$f_{OUT,S} = \frac{A}{\tau_F - \tau_R} \cdot \left[\tau_F \cdot e^{-\frac{Ti'}{\tau_F}} \cdot \left(1-\frac{\Delta Ti}{\tau_F}\right) \cdot \left(1-e^{-\frac{T}{\tau_F}}\right) - \tau_R \cdot e^{-\frac{Ti'}{\tau_R}} \cdot \left(1-\frac{\Delta Ti}{\tau_R}\right) \cdot \left(1-e^{-\frac{T}{\tau_R}}\right) \right]$$

To further simplify this expression, all of the constant terms that will give a 0 variance can be dropped (since we are only interested in the uncertainty at the output).

$$f_{OUT,S} = \frac{A}{\tau_F - \tau_R} \cdot \left[-\tau_F \cdot e^{-\frac{Ti'}{\tau_F}} \cdot \frac{\Delta Ti}{\tau_F} \cdot \left(1-e^{-\frac{T}{\tau_F}}\right) + \tau_R \cdot e^{-\frac{Ti'}{\tau_R}} \cdot \frac{\Delta Ti}{\tau_R} \cdot \left(1-e^{-\frac{T}{\tau_R}}\right) \right] \tag{2.23}$$

The expression for the output in equation 2.20 can be broken up into two contributions: the rising exponential and the falling exponential.

$$\begin{aligned} f_{OF} &= -\frac{A}{\tau_F - \tau_R} \cdot \tau_F \cdot e^{-\frac{Ti}{\tau_F}} \cdot \left(1-e^{-\frac{T}{\tau_F}}\right) \\ f_{OR} &= \frac{A}{\tau_F - \tau_R} \cdot \tau_R \cdot e^{-\frac{Ti}{\tau_R}} \cdot \left(1-e^{-\frac{T}{\tau_R}}\right) \end{aligned} \tag{2.24}$$

The functions $f_{OUT,S}$, f_{OF}, and f_{OR} can be related to the output voltage by multiplying them by the energy of incident radiation and some multiplicative factor, K.

$$\begin{aligned} V_{OUT,S} &= E_{rad} \cdot K \cdot f_{OUT,S} \\ V_{OF} &= E_{rad} \cdot K \cdot f_{OF} \\ V_{OR} &= E_{rad} \cdot K \cdot f_{OR} \end{aligned} \quad \text{where} \quad K = \frac{k_{\text{det}} \cdot q \cdot Ar_{GAIN}}{\tau_{\text{int}} + \tau_u} \tag{2.25}$$

Common terms in equation 2.23 can be substituted with the output functions in equation 2.24. Then using equation 2.25, the output can be written in terms of a voltage. Next, the variance of $V_{OUT,S}$ can be written as a constant times a random variable. The constant term can be pulled out of the variance operator and squared (Yates and Goodman 79):

$$V_{OUT,S} = \left(\frac{V_{OF}}{\tau_F} + \frac{V_{OR}}{\tau_R} \right) \cdot \Delta Ti$$
$$Var\left[V_{OUT,S,Ti}\right] = \left(\frac{V_{OF}}{\tau_F} + \frac{V_{OR}}{\tau_R} \right)^2 \cdot {\sigma_{Ti}}^2 \tag{2.26}$$

This equation related the uncertainty in the output voltage for a single exponential to the uncertainty in time associated with the start of the integration. Repeating this process, the uncertainty in the integration period, *T*, can be derived as well. This results in a different equation for the variance and is given in equation 2.27.

$$Var\left[V_{OUT,S,T}\right] = \left(\frac{V_{OF}}{\tau_F} \cdot \frac{e^{-\frac{T'}{\tau_F}}}{1 - e^{-\frac{T'}{\tau_F}}} + \frac{V_{OR}}{\tau_R} \cdot \frac{e^{-\frac{T'}{\tau_R}}}{1 - e^{-\frac{T'}{\tau_R}}} \right)^2 \cdot {\sigma_T}^2 \tag{2.27}$$

Now that the simple case of a single exponential has been solved, this expression can be expanded for *n* exponentials. First, let's define constants for the above variance equations:

$$c_{i,Ti} = \frac{V_{OF,i}}{\tau_{F,i}} + \frac{V_{OR,i}}{\tau_{R,i}}$$
$$c_{i,T} = -\frac{V_{OF,i}}{\tau_{F,i}} \cdot \frac{e^{-\frac{T'}{\tau_{F,i}}}}{1 - e^{-\frac{T'}{\tau_{F,i}}}} - \frac{V_{OR,i}}{\tau_{R,i}} \cdot \frac{e^{-\frac{T'}{\tau_{R,i}}}}{1 - e^{-\frac{T'}{\tau_{R,i}}}} \tag{2.28}$$

For the uncertainty in the integration start time, each exponential is correlated. The same thing is true for the jitter-induced noise in the integration period. Thus, the variance at the output can be written as follows:

$$\begin{aligned} Var\left[V_{OUT,Ti}\right] &= Var\left[V_{OUT,Ti,1} + V_{OUT,Ti,2} + \ldots + V_{OUT,Ti,n}\right] \\ &= Var\left[\left(c_{1,Ti} + c_{2,Ti} + \ldots + c_{n,Ti}\right) \cdot \Delta Ti\right] \\ &= \left(\sum_{i=1}^{n} c_{i,Ti} \right)^2 \cdot {\sigma_{Ti}}^2 \end{aligned} \tag{2.29}$$

$$\begin{aligned} Var\left[V_{OUT,T}\right] &= Var\left[V_{OUT,T,1} + V_{OUT,T,2} + \ldots + V_{OUT,T,n}\right] \\ &= Var\left[\left(c_{1,T} + c_{2,T} + \ldots + c_{n,T}\right)\cdot \Delta T\right] \\ &= \left(\sum_{i=1}^{n} c_{i,T}\right)^2 \cdot {\sigma_T}^2 \end{aligned} \tag{2.30}$$

The total variance in the output voltage is simply the sum of the individual variances. This is true because the uncertainty in the integration start time is independent of the uncertainty in the width of the integration period.

$$\begin{aligned} Var\left[V_{OUT}\right] &= Var\left[V_{OUT,Ti} + V_{OUT,T}\right] \\ &= Var\left[V_{OUT,Ti}\right] + Var\left[V_{OUT,T}\right] \\ {\sigma_j}^2 &= \left(\sum_{i=1}^{n} c_{i,Ti}\right)^2 \cdot {\sigma_{Ti}}^2 + \left(\sum_{i=1}^{n} c_{i,T}\right)^2 \cdot {\sigma_T}^2 \end{aligned} \tag{2.31}$$

In the proposed PSD system used for PID, there are two identified sources of jitter. One is from the external constant fraction discriminator (CFD) and the other is from the on-chip gate generators. Since the jitter from the CFD is the same over all integrators, this means that the voltage outputs of these integrators are also correlated. Correlation in the PSD plot actually helps to improve the signal-to-noise ratio (SNR), however, not significantly. The jitter associated with the gate generators, however, is completely uncorrelated at the output because there are separate gate generator circuits for each integrator.

Integrating resistor noise

There are several sources of electronic noise associated with the integrator circuit. One of them is the integrating resistor, R_{INT}. See figure 2.9. This resistor exhibits thermal noise which has a flat power spectral density given in equation 2.32 (Razavi 209). In this equation, k_B is Boltzmann's constant (equal to $1.38065 \cdot 10^{-23}$ J/K), and T_J is the junction temperature (typically around 300 K).

$$S_v(f) = 4k_B T_J \cdot R_{INT} \quad \text{where } f \geq 0 \tag{2.32}$$

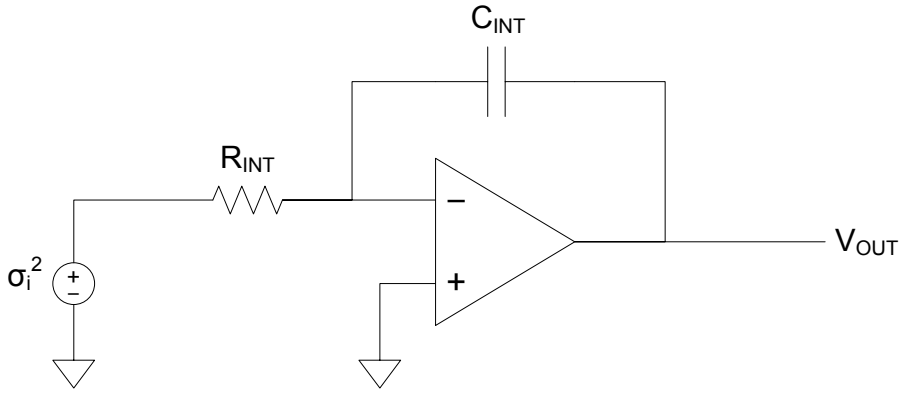

Figure 2.9 Integrator circuit with integrating resistor, R_{INT}, noise source

In order to calculate the effect of this noise source at the output, the input-output transfer function must be derived. This can be done by referring to figure 2.6. One observes that

$$V_{OUT} = \frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u} \cdot \frac{1}{\tau_{\text{int}}} \cdot \frac{1}{s} \cdot V_{IN} \tag{2.33}$$

By applying the variance operator, one may write

$$Var[V_{OUT}] = K^2 \cdot Var\left[\frac{1}{s} \cdot V_{IN}\right] \quad \text{where } K = \frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u} \cdot \frac{1}{\tau_{\text{int}}} \tag{2.34}$$

Since the noise at the input is thermal noise and is independent at any other point in time, then the noise power must be integrated instead of the noise spectrum.

$$\begin{aligned} Var[V_{OUT}] &= K^2 \cdot \int_0^T S_v(f) dt \\ &= K^2 \cdot T \cdot S_v(f) \end{aligned} \tag{2.35}$$

Substituting K and $S_v(f)$ back into equation 2.35 and reorganizing terms, an expression for the variance of the noise at the output can be written:

$$\sigma_{RI,t}{}^2 = 4\frac{k_B T_J}{C_{\text{int}}} \cdot \frac{T}{\tau_{\text{int}}} \cdot \left(\frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u} \right)^2 \tag{2.36}$$

OTA thermal noise

The thermal noise of the operational transconductance amplifier (OTA) can be modeled as a voltage source at the positive terminal of the op amp as illustrated in figure 2.10. It can therefore contribute noise to the output in two ways. First, the noise is sampled onto the capacitor while the circuit is integrating. Second, since there is a voltage at the positive terminal, it is also in series with the capacitor voltage. Therefore, a continuous-time noise contribution can be seen at the output as well.

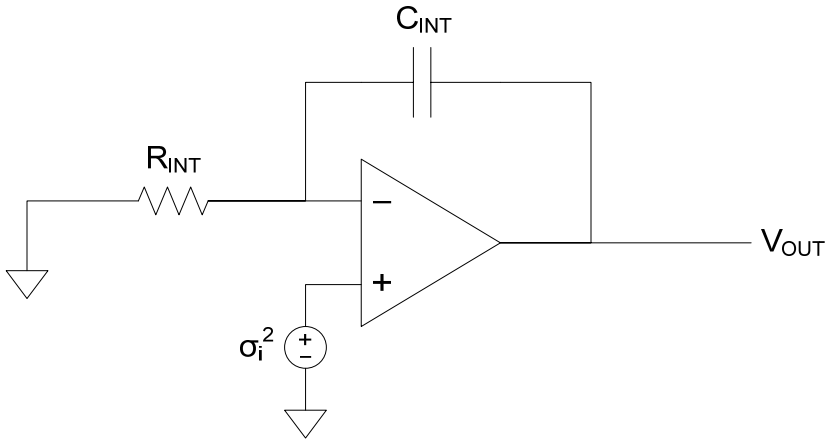

Figure 2.10 Integrator circuit with OTA noise source

The thermal noise of the op amp can be treated in the same way as the integrating resistor by finding an equivalent noise resistance, *RN*, of the op amp. This will give the following spectral density:

$$S_v(f) = 4k_B T_J \cdot RN \quad \text{where } f \geq 0 \tag{2.37}$$

For the sampled case, the relationship between input and output remains the same as that of the integrating resistor. Therefore, using equation 2.35 and substituting into it the spectral density, $S_v(f)$, for the op amp, the following expression for the noise at the output can be written:

$$\sigma_{OTA,t}^{\ 2} = 4\frac{k_B T_J}{C_{\text{int}}} \cdot \frac{T}{\tau_{\text{int}}} \cdot \left(\frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u} \right)^2 \cdot \frac{RN}{R_{\text{int}}} \tag{2.38}$$

In the continuous case, the noise at the positive terminal of the op amp is not integrated, but rather is in series with the output voltage. The output noise must be calculated by taking into consideration the spectral density of the input thermal noise. Thermal noise itself has a flat spectral density. Figure 2.6 models the integrator as a low-pass filter followed by a gain factor. The effects of finite bandwidth can be determined by integrating the power spectral density of a low-pass filter from 0 to infinity:

$$\begin{aligned} H(s) &= \frac{1}{1+\frac{s}{\omega_o}} \\ \int_0^\infty \left| H(j2\pi f) \right|^2 df &= f_o \cdot \frac{\pi}{2} \end{aligned} \tag{2.39}$$

This demonstrates that that the effective noise bandwidth can be computed by taking the 3 dB corner frequency f_o and multiplying by $\pi/2$. Therefore, the total integrated noise observed at the output of the low pass filter is:

$$\sigma_v^{\ 2} = 4k_B T_J \cdot RN \cdot f_o \cdot \frac{\pi}{2} \tag{2.40}$$

The gain factor from figure 2.6 must be incorporated into the expression. This is done by relating the input to the output.

$$V_{OUT} = \frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u} \cdot V_{IN,P} \tag{2.41}$$

Then, applying the variance operator, an expression can be derived for the output noise.

$$Var\left[V_{OUT}\right] = \left(\frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u}\right)^2 \cdot Var\left[V_{IN,P}\right] = \left(\frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u}\right)^2 \cdot \sigma_v{}^2 \tag{2.42}$$

This expression relates the output variance to the input variance. Next, σ_v^2 can be substituted into equation 2.42. The corner frequency, f_o, is simply $1/(2\pi\ \tau_o)$ where $\tau_o = \tau_{int} \,||\, \tau_u$. A simple expression for the output noise results in the following:

$$\sigma_{OTA+,t}{}^2 = k_B T_J \cdot RN \cdot \frac{1}{\tau_o} \cdot \left(\frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u}\right)^2 \tag{2.43}$$

OTA 1/f noise

Flicker noise, also known as 1/f noise, is a random noise source inherent in silicon based electronics. This type of noise is largely dependent on the process in which chips are made and can be tied to how clean the facilities are kept. The 1/f noise power spectral density of a CMOS device can be modeled as follows (Razavi 215):

$$S_v(f) = \frac{k_f}{f} \quad \text{where } k_f = \frac{K_F}{C_{ox} \cdot W \cdot L} \text{ and } f \geq 0 \tag{2.44}$$

The 1/f noise of the op amp comes from the contribution of each transistor in the circuit. All of these noise sources can be modeled by referring them back to the input (the positive terminal) and determining an equivalent k_f value for the combined noise sources.

For the integrator circuit, the 1/f noise of the op amp contributes a continuous and sampled noise to the output in the same way as the thermal noise. To determine this contribution, the power spectral density from equation 2.44 must first be integrated:

$$\int_{f_L}^{f_H} \frac{k_f}{f} df = k_f \cdot \ln\left(\frac{f_H}{f_L}\right) \tag{2.45}$$

Equation 2.45 states that the noise is band-limited. From figure 2.6, it is known that the bandwidth at the output is limited by a corner $1/\tau_o$, where $\tau_o = \tau_{int} \,||\, \tau_u$, and that the noise is multiplied by some constant gain factor. If the system is calibrated every t_{cal} time, then a lower frequency corner of $1/t_{cal}$ can be assumed. Therefore, the total integrated 1/f noise at the input is as follows:

$$\sigma_v^{\,2} = k_f \cdot \ln\left(\frac{t_{cal}}{\tau_o}\right) \tag{2.46}$$

In order to compute the total integrated 1/f noise at the output, the result from equation 2.42 which relates the input variance to the output variance can be used. Then, making substitutions for K and σ_v^2, the following expression can be written:

$$\sigma_{OTA+,f}^{\,2} = k_f \cdot \ln\left(\frac{t_{cal}}{\tau_o}\right) \cdot \left(\frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u}\right)^2 \tag{2.47}$$

For the sampled case, the input must be related to the output. Then applying the variance operator, one may show:

$$\begin{aligned} V_{OUT} &= \frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u} \cdot \frac{1}{\tau_{\text{int}}} \cdot \frac{1}{s} \cdot V_{IN} \\ Var\left[V_{OUT}\right] &= \left(\frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u}\right)^2 \cdot Var\left[\frac{1}{\tau_{\text{int}} \cdot s} \cdot V_{IN}\right] \end{aligned} \tag{2.48}$$

Flicker noise is inherently correlated to itself since it has a noise slope of -10 dB per decade. Therefore, the 1/f noise voltage can be assumed constant during the integration. This will give the worst case 1/f noise voltage at the output since it assumes perfect correlation. Later in this chapter, time-domain simulations will show that the actual noise at the output of the integrator is somewhat smaller than the value calculated using equations 2.48 and 2.49.

$$Var[V_{OUT}] = \left(\frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u}\right)^2 \cdot Var\left[\frac{1}{\tau_{\text{int}}} \cdot \int_0^T \sigma_v dt\right]$$
$$= \left(\frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u} \cdot \frac{T}{\tau_{\text{int}}}\right)^2 \cdot {\sigma_v}^2 \tag{2.49}$$

Using equation 2.47, an expression for the sampled 1/f noise at the output can be written as a function of the continuous-time 1/f noise:

$${\sigma_{OTA,f}}^2 = {\sigma_{OTA+,f}}^2 \cdot \left(\frac{T}{\tau_{\text{int}}}\right)^2 \tag{2.50}$$

This equation demonstrated that the sampled 1/f noise contribution depends on the ratio of the total integration time, T, to the integrator time constant, τ_{int}.

Quantization noise

The final noise source in the system to discuss is the quantization noise of an analog-to-digital (ADC) converter. The analog voltage at the output of the integrator is assumed to be digitized with an N-bit converter. Quantization noise can be modeled using a uniform random variable. The variance is known to be (Yates and Goodman 114):

$${\sigma_{ADC}}^2 = \frac{{Q_{bin}}^2}{12} \tag{2.51}$$

The quantization bin size, Q_{bin}, depends on the range and resolution of the ADC. It can be calculated by dividing the range by 2^N:

$$Q_{bin} = \frac{V_{range}}{2^N} \tag{2.52}$$

Total noise at integrator output

The total noise at the output of a single integrator can be calculated as the sum of the variances of each individual noise source.

$$\sigma_{VOUT}^{\ 2} = \sigma_{p}^{\ 2} + \sigma_{j}^{\ 2} + \sigma_{RI,t}^{\ 2} + \sigma_{OTA+,t}^{\ 2} + \sigma_{OTA,t}^{\ 2} + \sigma_{OTA+,f}^{\ 2} + \sigma_{OTA,f}^{\ 2} + \sigma_{ADC}^{\ 2} \qquad (2.53)$$

The signal-to-noise ratio can also be calculated by dividing the output of the integrator by the total noise at the output.

$$SNR = \frac{V_{OUT}}{\sigma_{VOUT}} \qquad (2.54)$$

Validation of Analytical Derivations

The noise at the output of an integrator was simulated using MATLAB by generating time-domain noise at the input. This noise can then be integrated over many different realizations. This allows the variance in the output voltage to be measured and compared against our derivations. In this simulation, the input pulses, the integrator, and each noise source was modeled in the time-domain. Although one is able to derive the effects of 1/f noise at the output, it is difficult to simulate 1/f noise in the time-domain at the input. Therefore, separate MATLAB simulations were done in order to see the effects of 1/f noise. For validating the remainder of the noise sources, a MATLAB simulation was performed using the parameters shown in table 2.2.

	Detectors	
	Liquid scintillator	CsI(Tl) scintillator
Number of noise samples generated	5,000	1,000
Pulse model	Gamma	Proton
Integration region	44 - 130 ns	1,500 - 3,000 ns
Integrator time constant, τ_{INT}	5 ns	200 ns
Bias mode of op amp	High	Low
Open loop gain of op amp	60 dB	60 dB
Gain-bandwidth product of op amp	50 MHz	50 MHz
Transresistive gain	4,359,072 Ω	4,598,239 Ω
Energy of incident radiation deposited	10 MeVee	100 MeV
Jitter in the start of integration	1.0 ns	7.0 ns
Jitter in the period of integration	0.5 ns	0.5 ns

Table 2.2 Setup for validating the analytical derivations of the system noise for liquid and CsI(Tl) scintillators.

The equations predicting the amount of noise at the output of the integrator were tested using two different types of detectors: liquid scintillator and CsI(Tl) scintillator. Liquid scintillator is a fast detector which requires a high gain-bandwidth product and therefore will be able to test the finite bandwidth which has been incorporated into the noise equations. CsI(Tl) scintillator is a slow detector which does not require as a much gain and will give results closer to what would be expected in an ideal system.

The model used for the integrator op amp is that of the one used in the PSD8C chip. As such, the typical gain-bandwidth product is 50 MHz with an open-loop gain above 60 dB (the typical open-loop gain is actually around 70 dB). It has been found that it is not important to model the finite gain of the op amp in the noise equations since its contribution is negligible. In order to prove this, the input noise was simulated using 60 dB open-loop gain which is considered the low end for the PSD8C op amp.

In the MATLAB simulations, the input pulses were modeled using a Poisson random variable with a lambda set by the average number of electrons arriving at a given point in time. Jitter-induced noise was incorporated by introducing an uncertainty in the start time and uncertainty in the width of integration period using a Gaussian random variable. The

thermal noise of the integrating resistor was modeled as a Gaussian random variable at the input to the integrator circuit. Similarly, the thermal noise of the op amp was also modeled as a Gaussian random variable, however, at the positive terminal of the op amp. Finally, the quantization noise of the analog-to-digital converter (ADC) was modeled as a uniform random variable at the output of the integrator. The results of these MATLAB simulations are shown in table 2.3.

	Detectors			
	Liquid scintillator		CsI(Tl) scintillator	
	Predicted noise (mV)	Simulated noise (mV)	Predicted noise (mV)	Simulated noise (mV)
Poisson noise	2.905	2.875	1.399	1.410
Jitter	5.517	5.505	2.475	2.579
Integrating resistor noise (thermal)	0.103	0.101	0.110	0.109
OTA noise sampled onto capacitor (thermal)	0.397	0.392	0.123	0.124
OTA continuous noise (thermal)	0.077	0.075	0.179	0.172
OTA noise sampled onto capacitor (1/f)	1.648	N/A	1.669	N/A
OTA continuous noise (1/f)	0.096	N/A	0.223	N/A
Quantization noise	0.070	0.071	0.070	0.070

Table 2.3 Validation of the analytical derivations of the system noise for liquid and CsI(Tl) scintillators.

In the above table, the noise levels were predicted using the derivations for the output noise and compared against the standard deviation of the time-domain noise waveform observed at the output of the integrator when driven by the appropriate noise source. The simulations demonstrate that the analytically predicted output noise for each noise source (except 1/f noise) is correct.

The effects of 1/f noise can be assessed by creating a time-domain noise simulation using a method described by Stephen C. Terry, et al. The main characteristic of 1/f noise is

that it has a noise slope of -10 dB/decade in the frequency-domain. Terry describes that this can be modeled by passing white noise through a noise-shaping filter. This filter consists of a bank of single-pole low-pass sub-filters that are summed at the output. The corner frequency of each sub-filter is one decade apart so that the complete filter produces the desired noise slope. Since it would require too much memory and computational time to simulate 1/f noise for a long period, Terry instead simulates only the higher frequency components and simply adds a constant Gaussian random variable for the lower frequency components.

Using this time-domain simulation of 1/f noise, we can now see how it compares to our prediction of the 1/f noise at the integrator output. In the derivation for the 1/f noise, it was assumed that the noise was constant while integrating. Because of this assumption, it represents the worst case at the output. However, since 1/f noise is not completely constant over the integration period, the actual noise at the output will be less.

The effects of 1/f noise can be further reduced by using a technique called correlated-double-sampling (CDS). This technique works by first integrating the signal of interest (with the additive effect of the 1/f noise). *Immediately* afterwards (waiting some period of time before performing the second integration would be less beneficial), the noise is integrated again but this time without the signal. The reason that the technique works is that the noise contribution to the second integration will be similar to the noise contribution to the first integration. This is due to the fact that 1/f noise is correlated in time. Thus, subtracting the two integrator outputs can remove a significant portion of the 1/f noise contribution. The noise "predicted" using the equations derived in this chapter, the "actual" noise observed in simulation, as well as the results of simulations where the CDS technique described above is employed is presented in figure 2.11.

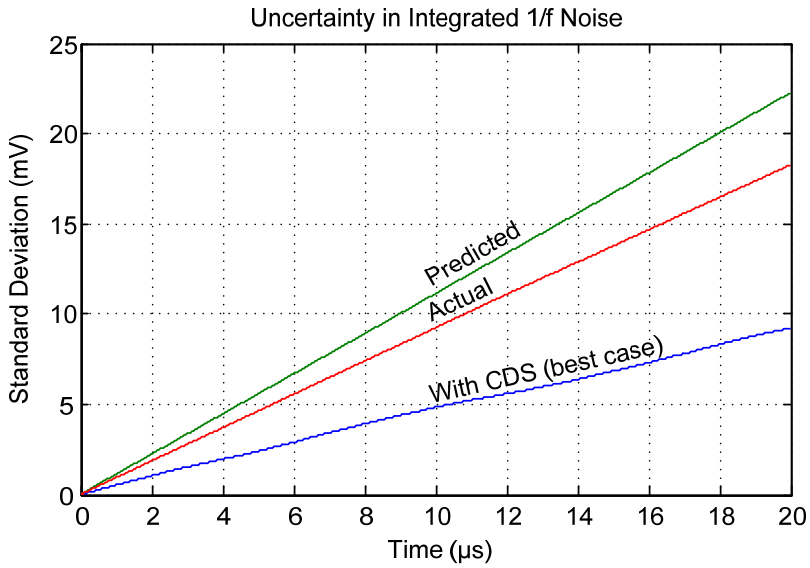

Figure 2.11 Uncertainty in the time-domain integrated 1/f noise

The above plot was created by taking the standard deviation at the integrator output over 2,000 realizations at different points in time. This plot clearly demonstrates that there is a close relationship between the "predicted" noise (using our derived equation) and the "actual" noise (obtained through simulation). In fact, one could define a "fudge" factor to produce an equation that models the "actual" noise observed when performing time-domain simulations. It was determined that that there is a -1.7 dB difference between the "predicted" (analytically derived equation) noise curve and the "actual" (time-domain simulation) noise curve. Since the difference between the relative curves was small, the theoretically predicted 1/f noise performance will be used in the remainder of the thesis. This will only slightly over-estimate the 1/f noise contribution to the output noise voltage.

If correlated-double-sampling (CDS) is used, then the best case improvement in the noise would be -7.6 dB from the predicted noise. The best case for correlated-double-sampling is when the second integration takes place immediately after the first integration. It is clear from this plot that 1/f noise can be improved using correlated-double-sampling.

CHAPTER 3

PREDICTING PERFORMANCE

In Chapter 3, we will use the noise characteristics of the gated integrators, derived in Chapter 2, to predict the performance of systems employing multiple charge integration. In one proposed system, only total pulse-height information is needed but in another, particle identification is desired. In both cases, it will be the signal-to-noise ratio (SNR) of the gated integrators that set overall performance. While tedious and long (many minutes to hours) time-domain simulations were performed to verify the correctness of our analytically derived output noise equations, we will use the equations in subsequent chapters to quickly and accurately predict the performance of systems employing gated integrators.

Discriminating Between Particles

In the previous chapter, the gated integrator was fully characterized. However, in order to discriminate between particles striking a detector, at least two different regions under the pulse-shape must be integrated. Depending on which regions are chosen, it will be easier or harder to discriminate between particles.

While it is theoretically possible to discriminate between many different particles, for illustrative purposes, we assume that there are only two particles that must be classified. Moreover, while more than two gated integrators may be used, we analyze a system in which only two gated integrators per channel are used. Systems of this sort may be implemented using the PSD8C IC since each channel on the chip contains three gated integrators.

In this chapter, particle identification using pulse-shape discrimination will be discussed in detail. Due to noise in the system, there is an inherent uncertainty in determining which particle struck the detector. Using hypothesis testing, the probability of a misclassifi-

cation can be computed. Finally, the overall performance of the system can be improved by choosing appropriate integration regions. This can be done by defining a figure-of-merit (FOM), which is a metric in which to judge system performance.

Pulse-shape discrimination is done by looking at the output of two integrators. Usually, one of these integrators covers an early region and the other a late region of the pulse. If the energy of incident radiation is scaled and the integrator outputs are plotted on the *x*- and *y*-axis, then one will get the pulse-shape discrimination plot shown in figure 3.1.

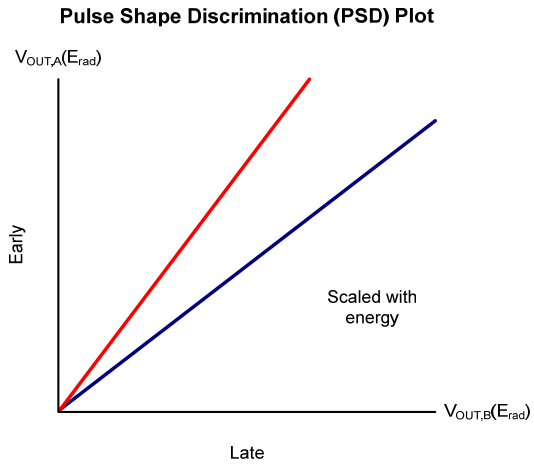

Figure 3.1 Pulse-shape discrimination plot

In this plot, there is a linear relationship between the two integrators. As the energy scales, the output of each integrator scales linearly. Therefore, particle discrimination can be done by looking at the ratio of these integrators. In particular, we consider the angle formed by each line to the *x*-axis in figure 3.1. The angle can be computed as the arc tangent of *A* over *B* where *A* is the output of the A integrator (early) and *B* is the output of the B integrator (late):

$$\theta = \tan^{-1}\left(\frac{A}{B}\right) \tag{3.1}$$

For two particles, equation 3.1 would yield the angles θ_1, and θ_2. By defining a threshold between these angles, one can discriminate between the particles. Ideally, these

angles should be spread apart in order to get the best discrimination. However, discrimination may not improve if the system noise increases as well.

Angle uncertainty

Since each integrator has noise at its output, this noise will factor in as an uncertainty in the angle, θ. Therefore, it becomes important to determine this uncertainty. The first step is to apply a Taylor expansion to equation 3.1.

$$\theta \approx \theta_0 + \frac{\partial \theta}{\partial A}\Big|_{A_0,B_0}(A - A_0) + \frac{\partial \theta}{\partial B}\Big|_{A_0,B_0}(B - B_0) \tag{3.2}$$

The variance operator is then applied to both sides of equation 3.2. Assuming that the covariance between A and B is zero (valid since the noise associated with the two integrator outputs is uncorrelated, except perhaps the jitter-induced noise), then the following is obtained:

$$Var(\theta) \approx \left(\frac{\partial \theta}{\partial A}\Big|_{A_0,B_0}\right)^2 \cdot Var(A) + \left(\frac{\partial \theta}{\partial B}\Big|_{A_0,B_0}\right)^2 \cdot Var(B) \tag{3.3}$$

The partial derivatives of θ with respect to A and B are then computed:

$$\frac{\partial \theta}{\partial A} = \frac{1}{1+\left(\frac{A}{B}\right)^2} \cdot \frac{1}{B} = \frac{1}{1+\tan^2(\theta)} \cdot \frac{1}{B} = \frac{\cos^2(\theta)}{B} \tag{3.4}$$

$$\frac{\partial \theta}{\partial B} = \frac{1}{1+\left(\frac{A}{B}\right)^2} \cdot \frac{-A}{B^2} = \frac{1}{1+\tan^2(\theta)} \cdot \frac{-\tan(\theta)}{B} = -\frac{\sin(2\theta)}{2B} \tag{3.5}$$

Substituting equations 3.4 and 3.5 into equation 3.3 gives the following:

$$Var(\theta) = \left[\frac{\cos^2(\theta)}{B}\right]^2 Var(A) + \left[\frac{\sin(2\theta)}{2B}\right]^2 Var(B) \tag{3.6}$$

The first term in this expression can be written in terms of the A integrator output by making the substitution $B = A / \tan(\theta)$. Since the signal-to-noise ratio (SNR) can be com-

puted as A / σ_A for the A integrator and B / σ_B for the B integrator, then the variance of theta can be written in terms of the signal-to-noise ratios of each integrator.

$$Var(\theta) = \left[\frac{\sin(2\theta)}{2A}\right]^2 Var(A) + \left[\frac{\sin(2\theta)}{2B}\right]^2 Var(B) \tag{3.7}$$

$$\sigma_\theta{}^2 = \frac{1}{4}\left\{\left[\frac{\sin(2\theta)}{SNR_A}\right]^2 + \left[\frac{\sin(2\theta)}{SNR_B}\right]^2\right\} \tag{3.8}$$

This equation tells us that the variance of the angle is dependent on the angle, θ, and on the signal-to-noise ratio of each individual integrator. It is also important to note that this equation gives us an approximation of the variance of θ which is good for small uncertainties in θ.

Hypothesis testing

Now that equations have been written that describe both the angle between the integrator outputs and the uncertainty in that angle, we must now answer the question of how well can one discriminate between particles. To do this, binary hypothesis testing can be used. According to Yates and Goodman, "there are two hypothetical probability models, H_0 and H_1, and two possible conclusions: *accept* H_0 as the true model, and *accept* H_1" (302). The probability models, H_0 and H_1, are referred to as "a priori probabilities" and "reflect the state of knowledge about the probability model before an outcome is observed" (302).

For the purposes of particle discrimination, we will assume that $P[H_0] = P[H_1] = 0.5$. There are two ways in which a particle can be misclassified. First, we can accept H_0 when H_1 is true, and second, we can accept H_1 when H_0 is true. The probability of these two events is written as $P[A_0 \mid H_1]$ and $P[A_1 \mid H_0]$, respectfully. Knowing this, the total probability of error can be calculated using the equation from Yates and Goodman (305):

$$P_{ERR} = P[A_1 \mid H_0] \cdot P[H_0] + P[A_0 \mid H_1] \cdot P[H_1] \tag{3.9}$$

The decision to accept or reject a hypothesis is determined by applying a threshold to the two pulses in the pulse-shape discrimination (PSD) plot. If one looks at the angular histogram between the two integrator outputs as illustrated in figure 3.2, then two Gaussian curves will be seen which overlap at some point.

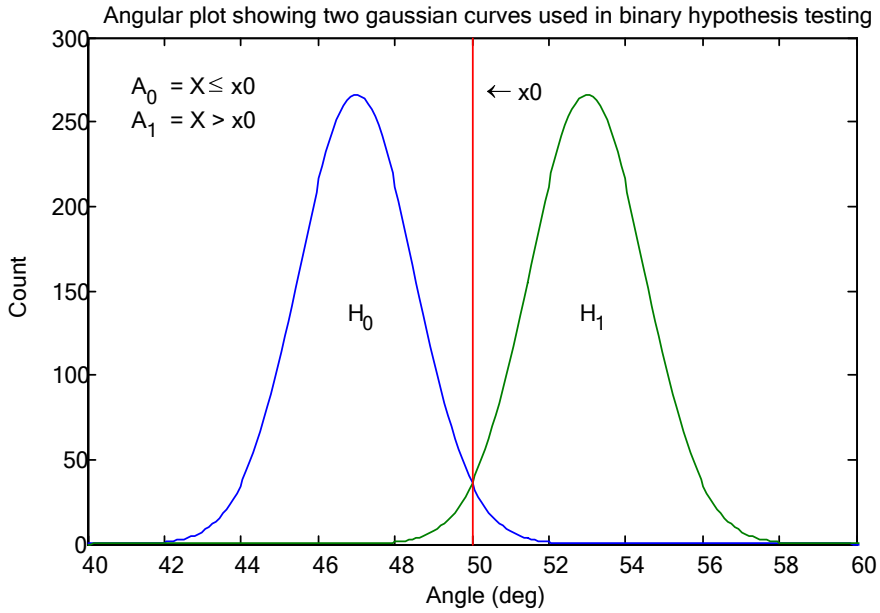

Figure 3.2 Angular histogram of pulse-shape discrimination plot

In order to discriminate, we have to make a decision. We will accept H_0 if $X \leq x0$ or accept H_1 if $X > x0$. Knowing the mean and standard deviation of each Gaussian curve, then the standard normal cumulative distribution function (CDF), $\Phi(z)$, can be used to write the individual probability of error of each particle. The standard normal CDF is defined as follows (Yates and Goodman 120):

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{u^2}{2}} du \tag{3.10}$$

For a Gaussian random variable, X, with mean, μ, and standard deviation, σ, the CDF of X can be written as (Yates and Goodman 120):

$$F_X(x) = \Phi\left(\frac{x-\mu}{\sigma}\right) \tag{3.11}$$

Then using equation 3.11, the probability of error of each particle can be written.

$$\begin{aligned} P[A_1 \mid H_0] &= P[X > x0 \mid H_0] = 1 - \Phi\left(\frac{x0-\theta_0}{\sigma_{\theta_0}}\right) \\ P[A_0 \mid H_1] &= P[X \le x0 \mid H_1] = \Phi\left(\frac{x0-\theta_1}{\sigma_{\theta_1}}\right) \end{aligned} \tag{3.12}$$

Substituting this into equation 3.9 gives the total probability of error or misclassification of a particle:

$$P_{ERR} = \left[1 - \Phi\left(\frac{x0-\theta_0}{\sigma_{\theta_0}}\right)\right] \cdot P[H_0] + \left[\Phi\left(\frac{x0-\theta_1}{\sigma_{\theta_1}}\right)\right] \cdot P[H_1] \tag{3.13}$$

Optimization of integration regions

Optimal regions of integration can be found in order to improve overall performance. The ability to discriminate between particles depends on the integration regions and value of the integrating resistor. Therefore, system performance can be improved by choosing good values. In the previous section, hypothesis testing was discussed as a way to determine statistically how well two particles can be discriminated between.

This method, however, does not work well for optimization. Instead, a figure-of-merit (FOM) should be defined as discussed in Chapter 1. System performance can be improved by maximizing the figure-of-merit. The figure-of-merit tries to spread apart the two angles, θ_0 and θ_1, while minimizing noise ($\sigma_{\theta 0}$ and $\sigma_{\theta 1}$). The FOM is defined as follows:

$$FOM = \frac{|\theta_1 - \theta_0|}{\sqrt{\sigma_{\theta_1}{}^2 + \sigma_{\theta_0}{}^2}} \tag{3.14}$$

Validation of Angular Uncertainty

The derivation of the angular uncertainty in equation 3.8 was tested using a MATLAB simulation. In Chapter 2, the output of an integrator was characterized using equation 2.15 along with its associated noise variance in equation 2.53. The simulation was setup to predict the output of two integrators (by getting the signal voltage and the noise variance) using these characteristic equations (with $V_{i2} = 0$ in equation 2.15). With this information, noise at the output of each integrator could be generated using a Gaussian random variable with the correct mean (the signal voltage) and standard deviation (the square root of the noise variance). The angular uncertainty could also be predicted by using the signal-to-noise ratio of each integrator output and the angle between them. This allows the variance in the angle between the "noisy" integrator outputs to be measured and compared against the derivation. The parameters for this simulation are shown in table 3.1.

	Detectors	
	Liquid scintillator	CsI(Tl) scintillator
Number of noise samples generated	5,000	5,000
Pulse model	Gamma	Proton
Integrator A region	0 - 200 ns	0 - 400 ns
Integrator B region	44 - 130 ns	1,500 - 3,000 ns
Integrator A time constant, $\tau_{INT,A}$	20 ns	500 ns
Integrator B time constant, $\tau_{INT,B}$	5 ns	200 ns
Bias mode of op amp	High	Low
Open loop gain of op amp	60 dB	60 dB
Gain-bandwidth product of op amp	50 MHz	50 MHz
Transresistive gain	4,359,072 Ω	4,598,239 Ω
Energy of incident radiation	10 MeVee	100 MeV
Jitter in the start of integration	1.0 ns	7.0 ns
Jitter in the period of integration	0.5 ns	0.5 ns

Table 3.1 Setup for validating the analytical derivation of the angular uncertainty for liquid and CsI(Tl) scintillators.

The equation predicting the angular uncertainty was tested using two different detectors over a range of energies. In the derivation of this equation, an approximation of the arc

tangent was made using a Taylor expansion in order to get a linear expression for the variance. In this approximation, it is assumed that the variance at the output of each integrator is small compared to the signal voltage. In cases where this is not true, such as for low energy particles, one would expect the angular uncertainty to become larger in a non-linear way. However, since discrimination between particles depends on having good signal-to-noise ratios, then this approximation will be a good approximation.

	Detectors			
	Liquid scintillator		CsI(Tl) scintillator	
Energy of incident radiation	Predicted angular uncertainty (°)	Simulated angular uncertainty (°)	Predicted angular uncertainty (°)	Simulated angular uncertainty (°)
100 keV	10.878	10.882	89.793	82.032
1 MeV	1.268	1.254	9.027	10.108
10 MeV	0.406	0.410	1.024	1.033
100 MeV	N/A	N/A	0.425	0.427

Table 3.2 Validation of the analytical derivation of the angular uncertainty for liquid and CsI(Tl) scintillators.

In the above table, the angular uncertainty was *predicted* using the derivation in equation 3.8 and compared against the standard deviation of the angle between two "noisy" integrator outputs that were *simulated*. The results demonstrate that the analytically predicted uncertainty is in good agreement with the simulated uncertainty. The results also show that the agreement between the predicted and simulated uncertainty is not as close for low energy particles due to the fact that the angular uncertainty is high (the assumption that the uncertainty scales linearly is no longer true).

CHAPTER 4

RESULTS

Introduction

Since the performance of a particle identification (PID) system that uses pulse-shape discrimination (PSD) techniques has been characterized in Chapter 3, it is possible to predict the performance of these types of systems. In particular, this chapter investigates the performance of PID systems which use different types of detectors. Once again we assume that the PSD8C IC is used to implement the PID system.

In addition to using the PSD8C IC for particle identification, it can also be used to obtain total pulse-height (*i.e.* energy) information. Theoretically, total pulse-height information is obtained by integrating under the entire pulse using a single gated integrator. However, integrating for a long period of time causes an increase in the amount of noise at the output. What is really important is not the output noise; it is the signal-to-noise ratio (SNR) of the integrators (as was the case for PID systems). Clearly, in systems seeking to obtain energy information, performance is directly related to SNR at the integrator output.

Therefore, we also look at optimizing the SNR of an integrator with respect to total integration time in order to improve the performance of systems seeking energy information. Moreover, since each channel of the PSD8C circuit contains three gated integrators (and only one is needed to obtain the total energy of the particle), we demonstrate in this chapter how the other two gated integrators can be used to implement a pile-up detector. Pile-up occurs when a second pulse arrives at the integrator output before it has finished integrating a pulse that had arrived earlier in time. In some applications, pile-up severely degrades overall system performance and must be detected. Appropriate action may then be taken to deal effectively with the pile-up events.

Op Amp Noise Results

The op amp used in the PSD8C IC is a two-stage design. The chip can run in either a low- or high-power consumption mode. Depending on the mode, the noise of the op amp will be different. *Spectre* (SPICE-like simulator distributed by Cadence) simulations were performed in order to determine the equivalent noise resistance, RN, of the op amp and the k_f parameter for 1/f noise of the op amp. These noise parameters are summarized in table 4.1.

Power Consumption	RN_t (Ω)	k_f (V^2/Hz)
Low	25,000	1.93E-09
High	7,400	9.12E-10

Table 4.1 Simulated op amp noise for low and high power consumption; (RN_t is the equivalent resistance for the thermal noise of the op amp; k_f is the constant multiplicative factor of the 1/f noise).

We will find, in the simulation results, that sampled 1/f noise is the dominate noise source. In Chapter 2, the derivation for the 1/f noise contribution to the output assumes that the 1/f noise voltage is constant while being integrated. In this case, we are assuming that the 1/f noise is perfectly correlated. In order to provide bounds on the performance of the PID system under investigation, we present cases *with* (worst-case) and *without* (unrealizable, best-case scenario) 1/f noise included as the actual performance of the system will lie somewhere in-between. Recall, however, that without using CDS, the performance of the system will be very close to the case where the 1/f noise *is* included. At present PSD8C is incapable of implementing the CDS technique discussed in Chapter 2 of this thesis. As we demonstrated in Chapter 2, even when CDS is used, not all of the 1/f noise can be removed.

Simulation Results for a CsI(Tl) Scintillation Detector

The CsI(Tl) scintillation detector produces pulses with long time constants and therefore does not require as high of a GBW. Therefore, this detector can be run in low-power consumption mode which is characterized by a thermal noise equivalent resistance of 25,000 Ω and a 1/f noise multiplicative factor of $1.93 \cdot 10^9$ V^2/Hz for the op amp.

In Chapter 2, the energy conversions that take place in the detector were described in equation 2.1 using several parameters (dependent on the properties of the physical devices). The parameters used for the CsI(Tl) scintillation detector are given in table 4.2.

E_{vis}	3 eV
ε_{con}	0.17
ε_{coll}	0.8
ε_q	0.25

Table 4.2 Detector parameters used for the CsI(Tl) scintillation detector.

Sample pulses

For the CsI(Tl) scintillation detector, we were provided with pulse models for two particles that must be classified: an alpha, and a proton particle. These pulses can be constructed using the general model in equation 2.3 which expresses a pulse as a sum of exponentials. The parameters for these pulses are listed in table 4.3.

	A_1	τ_{F1} (ns)	τ_{R1} (ns)	A_2	τ_{F2} (ns)	τ_{R2} (ns)
Alpha	0.5	200	2	0.5	7,000	70
Proton	0.5	700	7	0.5	7,000	70

Table 4.3 Pulse models for a CsI(Tl) scintillation detector.

Using MATLAB, one can simulate these pulses with their associated Poisson noise. Poisson noise is the noise associated with the random arrival of electrons. These "noisy" waveforms are illustrated in figure 4.1.

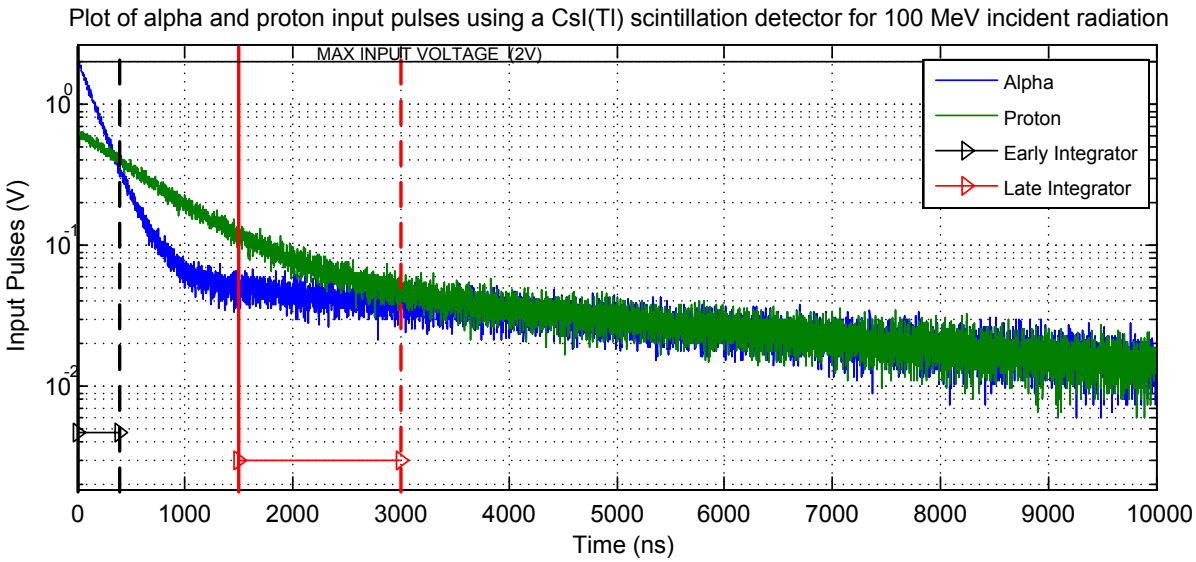

Figure 4.1 Alpha and proton input pulses using a CsI(Tl) scintillation detector

Integration regions

In order to do PID using PSD, one must choose appropriate integration regions which allow one to discriminate easily between particles. This is done by designating one integrator as the "early" integrator and the other one as the "late" integrator. In the case of CsI(Tl), the integration regions shown in table 4.4 were selected.

	Delay (ns)	Width (ns)
Early Integrator	0	400
Late Integrator	1,500	1,500

Table 4.4 Integration regions chosen for a CsI(Tl) scintillation detector.

Noise breakdown

In the process of designing PSD8C, we analyzed the effect of each noise source and created a breakdown describing its contribution to the output noise of the integrator. Using the integration regions in table 4.4, we created a noise breakdown for the early integrator (figure 4.2) and for the late integrator (figure 4.3). This was done both *with* and *without* 1/f

noise included in order to get an upper and lower bound for the expected performance of the system.

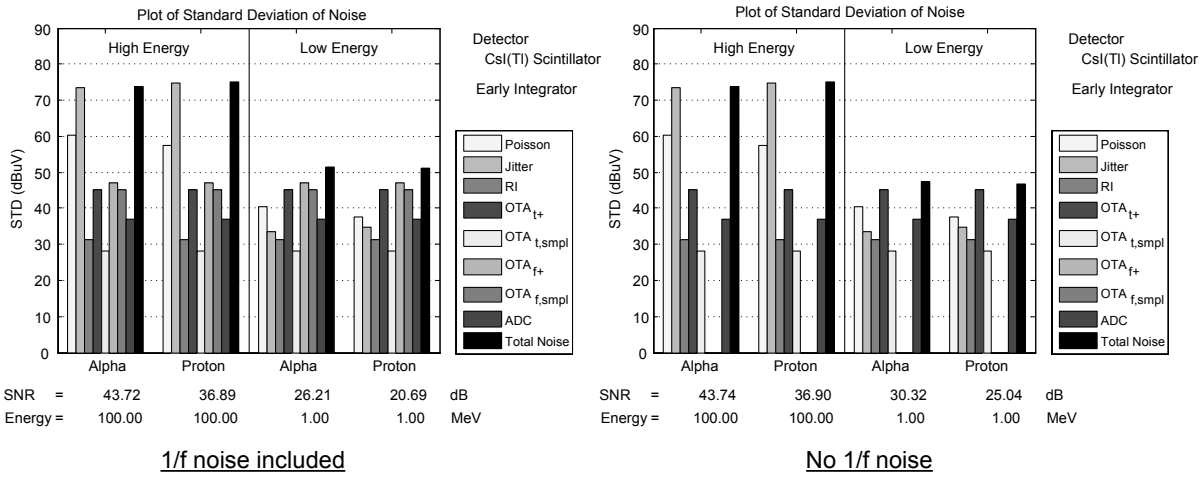

Figure 4.2 Noise break down of early integrator for a CsI(Tl) scintillation detector; (left) 1/f noise included; (right) no 1/f noise

For a CsI(Tl) scintillation detector, the worst case time jitter associated with the externally generated discriminator signal is approximately 7 ns (standard deviation). For the early integrator, the integration will begin before the arrival of the pulse which effectively eliminates the jitter-induced noise contribution associated with the start of the integration period.

However, some jitter-induced noise is still present at the A integrator output because the time jitter associated with the discriminator will also affect the time at which the integration ends. Since the width for the early integrator is short, one can still observe a strong jitter-induced noise contribution at the output in figure 4.2. This makes the jitter-induced noise the dominant noise source at high energies. At low energies, however, electronics noise from the integrating op amp is the dominate noise source.

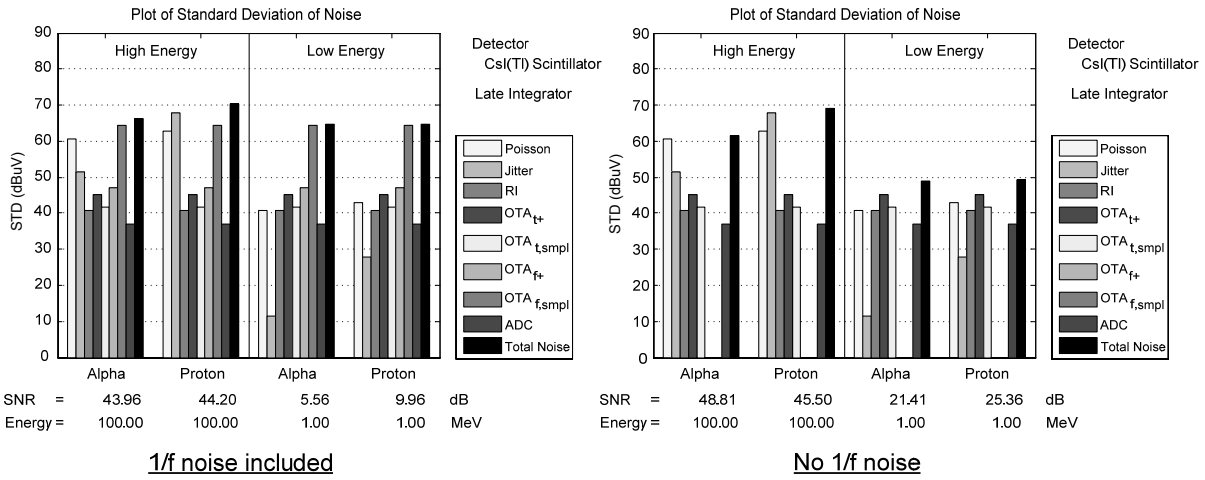

Figure 4.3 Noise break down of late integrator for a CsI(Tl) scintillation detector; (left) 1/f noise included; (right) no 1/f noise

For the late integrator shown in figure 4.3, jitter-induced noise is not as important since the B (late) integrator starts integrating in the tail of the pulse where the signal is small. For the late integrator, the sampled 1/f noise is the dominate noise source which greatly affects the signal-to-noise ratio (SNR) at low energies. The effect of the 1/f noise is to reduce the overall dynamic range of the system.

Pulse-shape discrimination

In this section we consider the performance of a PID system using the CsI(Tl) detectors. Plotting the late integrator output on the *x*-axis and the early integrator output on the *y*-axis will produce a PSD plot that can be used for particle identification. This was done *with* and *without* 1/f noise included as shown in figure 4.4.

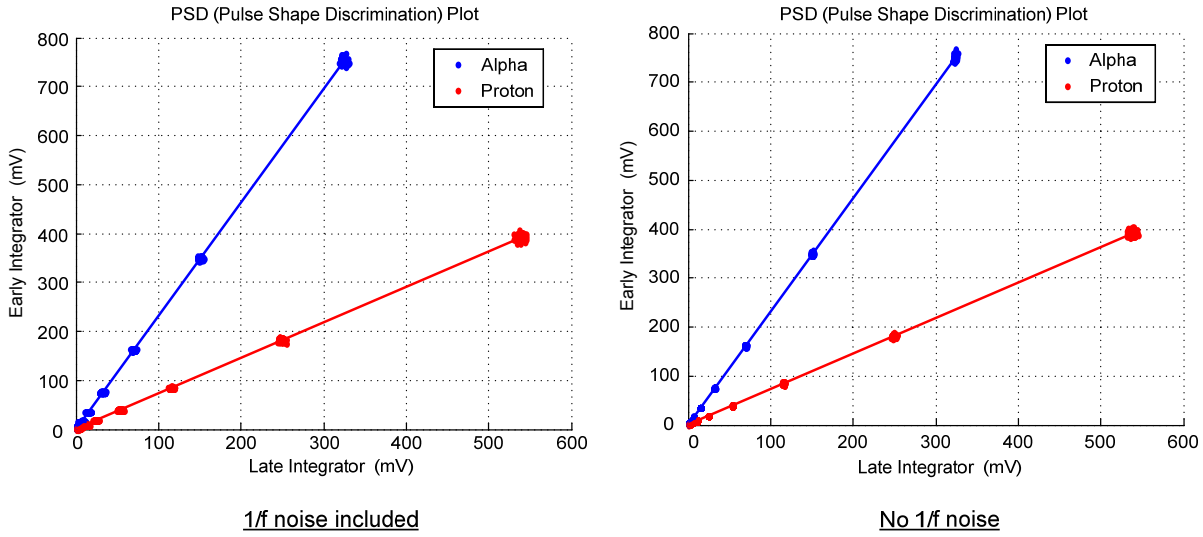

Figure 4.4 Pulse-shape discrimination (PSD) plot for a CsI(Tl) scintillation detector scaled from 0.1 to 100 MeV; (left) 1/f noise included; (right) no 1/f noise

We conclude that PID using PSD works very well for high energy particles using a CsI(Tl) scintillation detector. However, at low energies, discriminating between the two particles becomes difficult. To quantify the point at which the system performance becomes questionable, one can determine the levels at which the probability of error or misclassification is about 1%.

Figure 4.5 shows the angular histogram plots for the CsI(Tl) scintillation detector with 1/f noise included on the left plot but absent on the right.

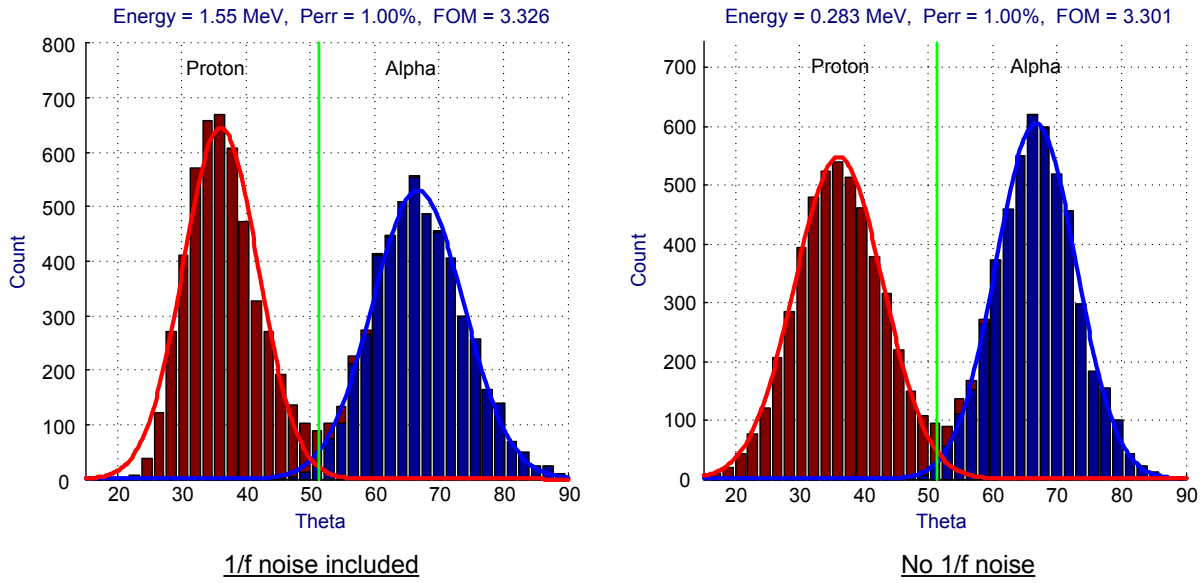

Figure 4.5 Angular histogram plot for a CsI(Tl) scintillation detector; (left) 1/f noise included; (right) no 1/f noise

Energy

The pulse-shape discrimination (PSD) system can also be used to obtain total pulse-height (energy) information by integrating under the entire pulse. There are a couple of considerations when choosing the integration time. First, there is the percentage of the area under the pulse. It is usually desirable to integrate for at least 90% of the pulse. The second consideration is the SNR of the integrator. A long integration time will cause thermal and 1/f noise to be integrated onto the capacitor, thereby lowering the SNR. The simulation results for a proton particle using a CsI(Tl) scintillation detector are presented in figure 4.6.

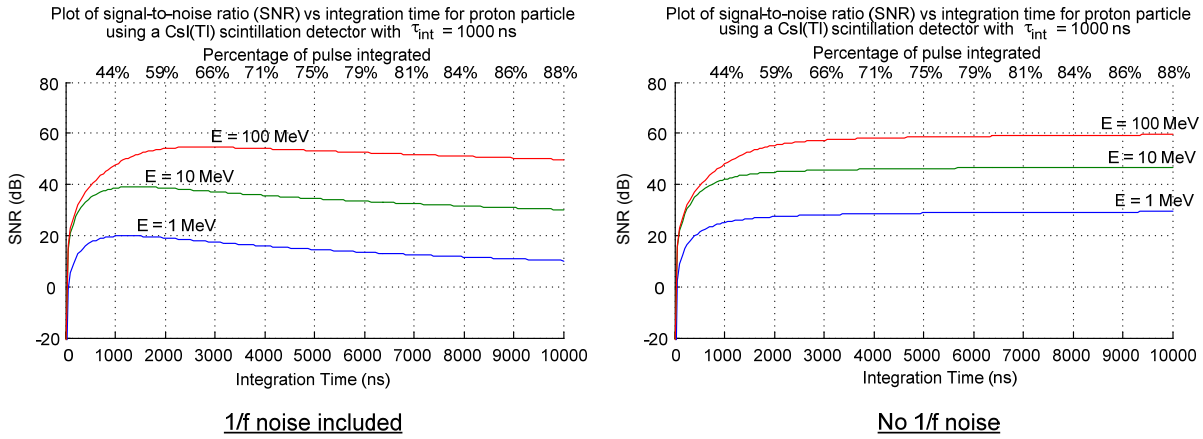

Figure 4.6 Signal-to-noise ratio vs integration time plot for a CsI(Tl) scintillation detector; (left) 1/f noise included; (right) no 1/f noise

In this plot, the effects of 1/f noise are clearly important. Choosing the optimal integration of about 3 μs (maximizes SNR for 100 MeV inputs) may not be desirable because only 66% of the pulse is shown to be integrated at that point. If the pulse were integrated over the full 10 μs, then 88% of the pulse would be integrated at a sacrifice of about 5 dB of SNR. For a low energy particle such as 1 MeV, the loss is closer to 7 dB.

Simulation Results for a Liquid Scintillation Detector

The liquid scintillation detector produces pulses with much shorter time constants than the CsI(Tl) scintillation detector and therefore the op amp must possess a large GBW. This large GBW will also help improve noise performance at the expense of power. In this high power consumption mode, the thermal noise equivalent resistance of the op amp is 7,400 Ω with a 1/f noise multiplicative factor of $9.12 \cdot 10^{-10}$ V^2/Hz.

In Chapter 2, the energy conversions that take place in the detector were described in equation 2.1 using several parameters (dependent on the properties of the physical devices). The parameters used for the liquid scintillation detector are given in table 4.5.

E_{vis}	3 eV
ε_{con}	0.045
ε_{coll}	0.8
ε_{q}	0.25

Table 4.5 Detector parameters used for the liquid scintillation detector.

Sample pulses

For this detector, we have two sample pulse models: a gamma-ray, and a neutron particle. These pulses were constructed using the general model in equation 2.3 which expresses a pulse as a sum of exponentials. The parameters for these pulses are listed in table 4.6.

	A_1	τ_{F1} (ns)	τ_{R1} (ns)	A_2	τ_{F2} (ns)	τ_{R2} (ns)
Gamma	1.000	10	0.1			
Neutron	0.833	10	0.1	0.167	25	0.25

Table 4.6 Pulse models for a liquid scintillation detector.

Using MATLAB, one can simulate these pulses with their associated Poisson noise. These "noisy" pulses are illustrated in figure 4.7.

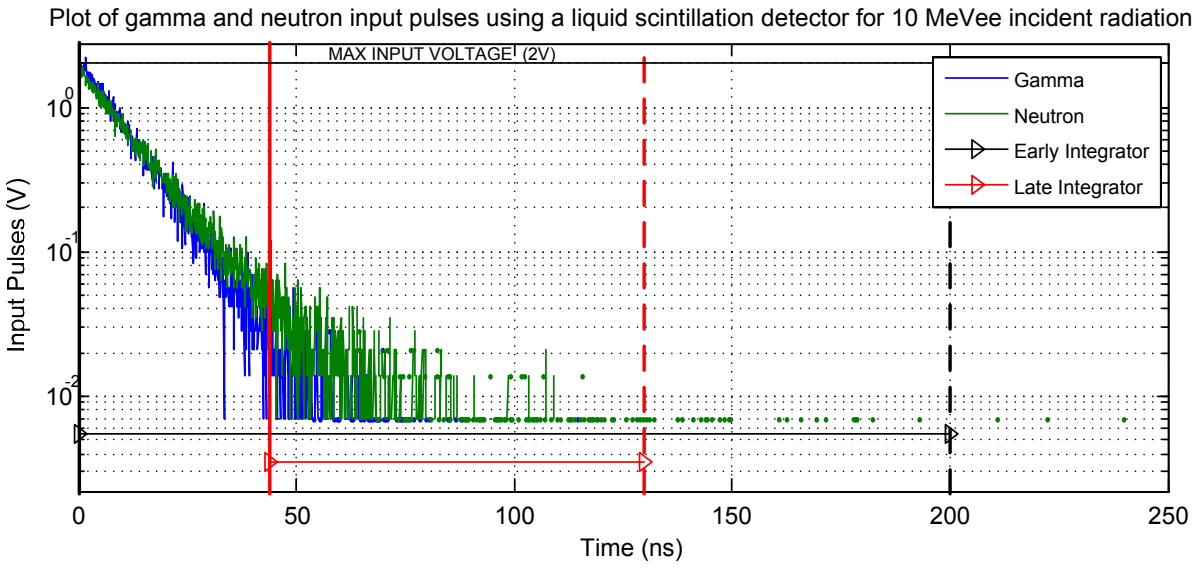

Figure 4.7 Gamma and neutron input pulses using a liquid scintillation detector

Integration regions

For the liquid scintillation detector, one needs to choose integration regions that will allow one to discriminate easily between particles. Since this detector produces very fast pulses, the integration regions need to be shorter. Also, the early integrator is set to integrate over the entire pulse region in order to get a large signal at the integrator output. Since the late integrator must integrate in the tail, one will not see as much signal at the output. The integration regions used are shown in table 4.7.

	Delay (ns)	**Width (ns)**
Early Integrator	0	200
Late Integrator	44	86

Table 4.7 Integration regions chosen for a liquid scintillation detector.

Noise breakdown

The noise at the output of the integrator for the liquid scintillation detector can be broken down into individual contributions. This was done *with* and *without* 1/f noise included as shown in figure 4.8.

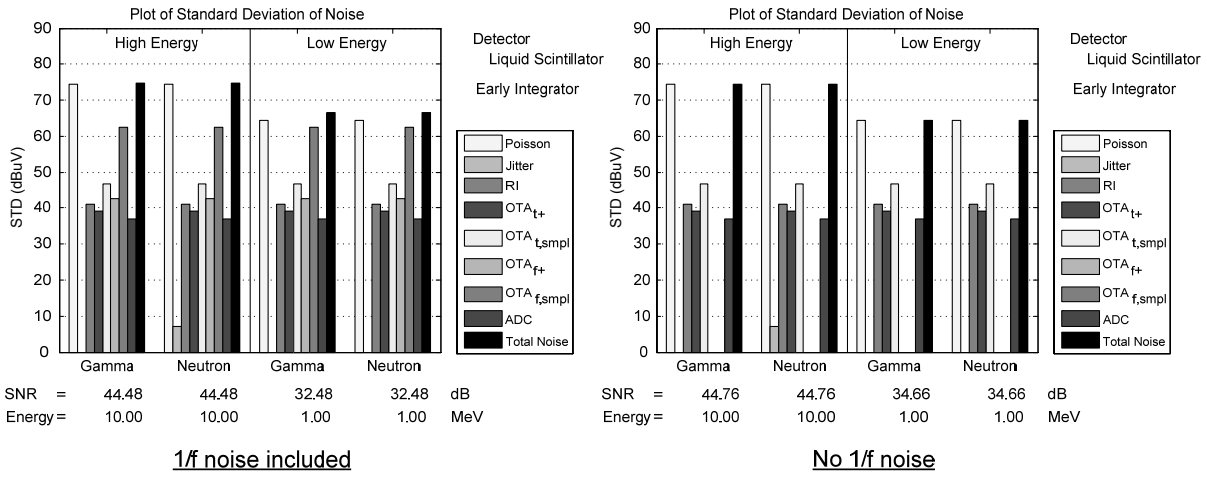

Figure 4.8 Noise break down of early integrator for a liquid scintillation detector; (left) 1/f noise included; (right) no 1/f noise

For a liquid scintillation detector, the worst case time jitter associated with the discriminator signal (heralds the onset of the pulse) is about 1 ns (standard deviation). For the early integrator, the integration will begin before the start of the pulse which effectively eliminates the contribution of jitter-induced noise at the output. Since the integration region ends at a point in the pulse where there is little signal, one does not observe any significant contribution from jitter-induced noise in figure 4.8. In both cases where 1/f noise is included and suppressed, Poisson noise remains the dominant noise source.

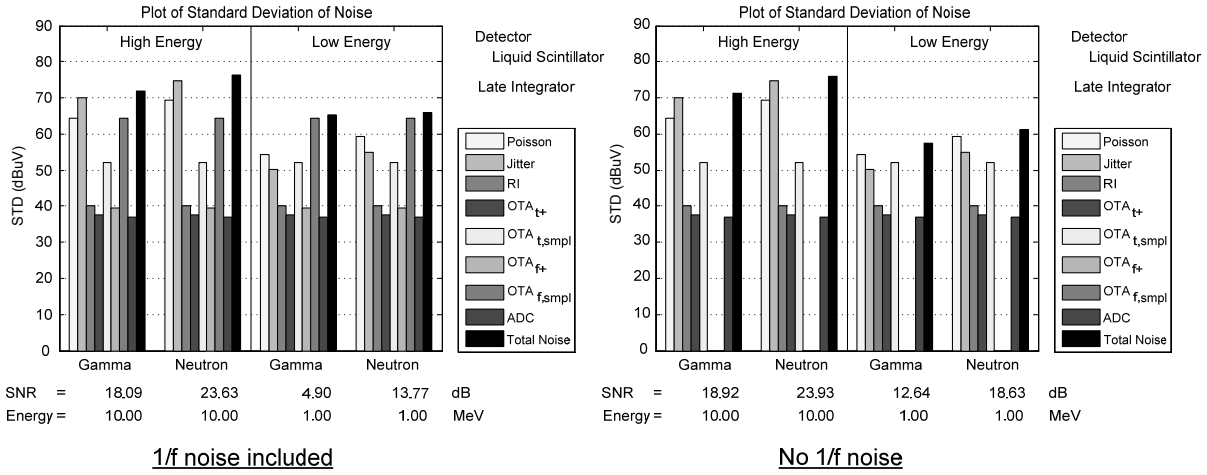

Figure 4.9 Noise break down of late integrator for a liquid scintillation detector; (left) 1/f noise included; (right) no 1/f noise

For the late integrator (results shown in figure 4.9), jitter-induce noise becomes more significant at high energies. However, at low energy *without* 1/f noise included, Poisson noise is the dominant noise source. If we include the 1/f noise source, then there is a loss of about 7.7 dB of SNR for gamma-rays and 4.83 dB for neutrons.

Pulse-shape discrimination

After simulating these pulses, pulse-shape discrimination is performed by plotting the late integrator on the *x*-axis and the early integrator on the *y*-axis. This yields a plot with a linear upward growing trend. This was done both *with* and *without* 1/f noise included as shown in figure 4.10.

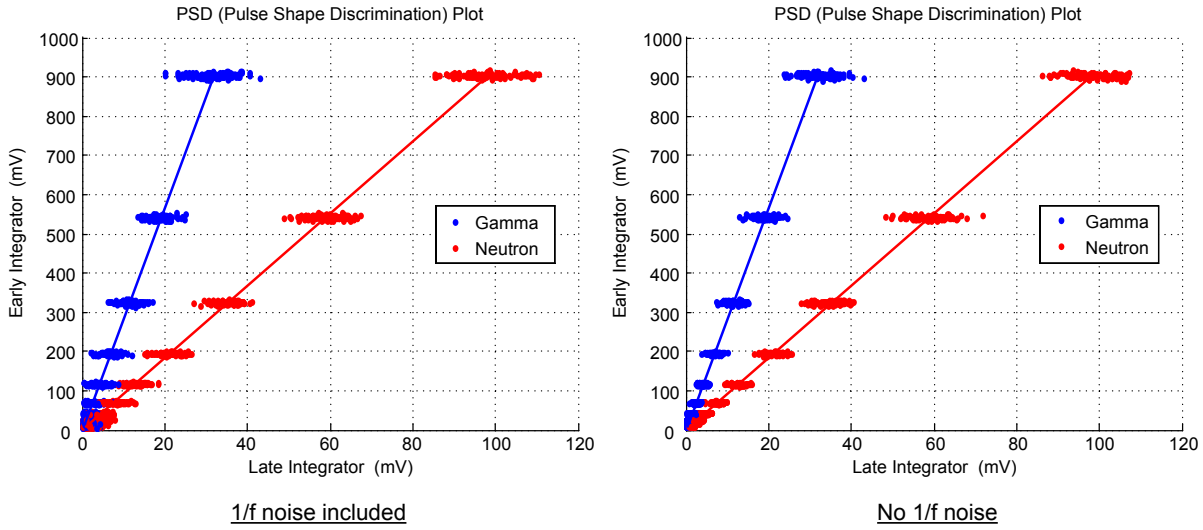

Figure 4.10 Pulse-shape discrimination (PSD) plot for a liquid scintillation detector scaled from 0.1 to 10 MeVee; (left) 1/f noise included; (right) no 1/f noise

Since the late integrator integrates in tail of the pulse, there is much less signal which lowers the signal-to-noise ratio (SNR) and produces the elliptical noise shape seen in figure 4.10. The energy range shown in the above PSD plot is from 100 keVee to 10 MeVee. This plot shows that we can discriminate between particles very well at high energies. At low energies, however, the data points begin to overlap.

The angular histogram plot in figure 4.11 shows the angular noise distribution of each particle *with* and *without* 1/f noise included. The performance (1% of particles misclassified) will be 1.44 MeVee.

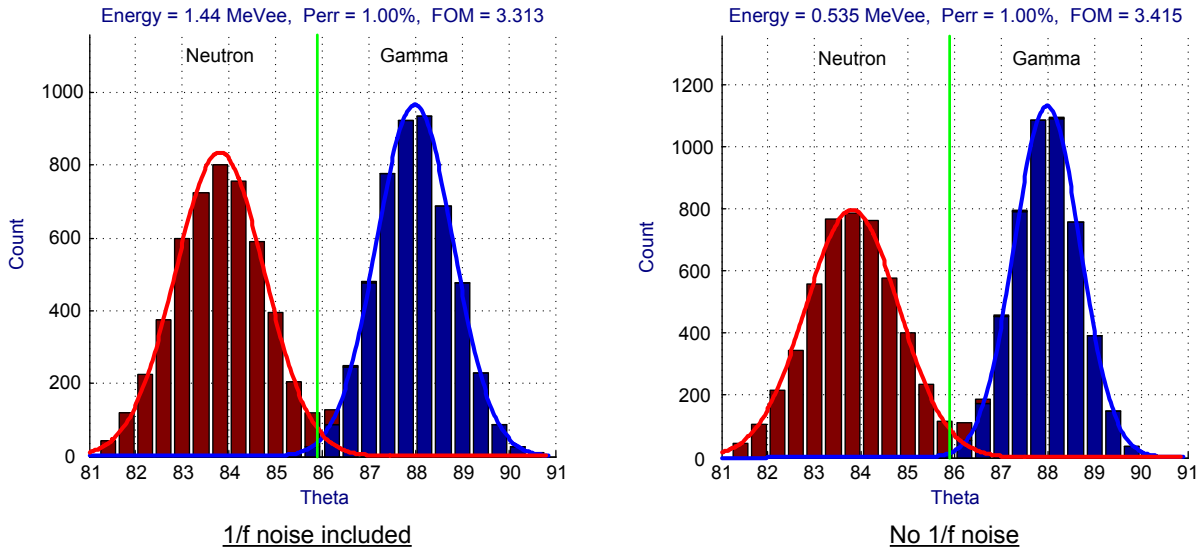

Figure 4.11 Angular histogram plot for a liquid scintillation detector; (left) 1/f noise included; (right) no 1/f noise

Energy

Using the PSD8C chip, total pulse-height information can be obtained using a liquid scintillation detector. Since the liquid scintillation detector produces pulses with short time constants, the chip is capable of integrating nearly 100% of the pulse. The simulation results for a neutron particle using a liquid scintillation detector are shown in figure 4.12.

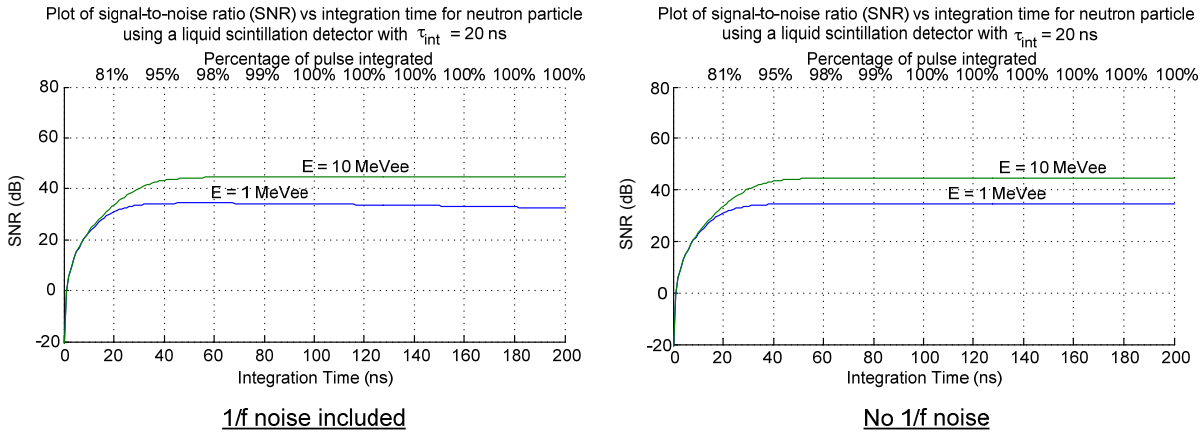

Figure 4.12 Signal-to-noise ratio vs integration time plot for a liquid scintillation detector; (left) 1/f noise included; (right) no 1/f noise

The effects of 1/f noise are not as apparent with the liquid scintillator model simulations as they were with CsI(Tl) scintillator simulations. In the former, the entire pulse can be integrated in about 100 ns with only a minimal loss of the SNR from the optimum value.

Simulations for the New High Resolution Scintillation Array (HiRSA)

Pulse data was obtained that was captured in an experiment using a prototype CsI(Na) detector. The original data file contained 20,000 pulses, however, a few pulses were corrupted, some saturated the ADC, and some contained pile-up. All of the "bad" pulses were removed from the input data file. The pile-up events were the more difficult to detect and remove. An algorithm, based on the ratio of two integration periods and the total integrated energy, to detect pile-up was used.

Before working on the pile-up detection routine, however, an energy spectrum was generated. As these data were collected with ^{137}Cs (662 keV) and ^{60}Co (1173 and 1332 keV) gamma-ray sources, the generated spectrum had three Gaussian peaks corresponding to the photopeaks. These peaks result when the entire energy of the gamma ray is transferred to an e^- and all the energy of the e^- is converted into heat [~90%] and visible photons [~10%] in the crystal. Using the centroid of the photopeaks, it is possible to calibrate the energy of the pulses. The pulses in the file were offset to start at 1 μs. Since the PSD8C chip will be used in this application, the following integration regions were selected:

(A) 0.9 – 2.0 μs

(B) 0.9 – 4.0 μs

(C) 0.9 – 10.0 μs

All three of the integrators were assigned the same integrator time constant, τ_{int}, of 500 ns. The energy of the pulse was calibrated using the C integrator.

Since pile-up detection is important, an algorithm based on the ratio between the A and C integrators and the energy in the pulse was designed. If the ratio versus the pulse energy is plotted as shown in figure 4.13, a band of data containing the real, single-hit spectrum is obtained. Pile-up events are below the single-hit events as the former contains contributions from additional pulses after the initial one that triggered the CFD. This additional light increases the integration for the wide gate more than it does for the narrow gate. (The C integrator increases more than the A integrator.) Pulses with contributions from other pulses before the CFD triggers would also not fall on the single hit locus. Normally, this locus of pulses will have the same ratio regardless of the energy in the pulse. However, at lower energy, the pulses tend to have a smaller ratio. This is due to a low energy garbage distribution.

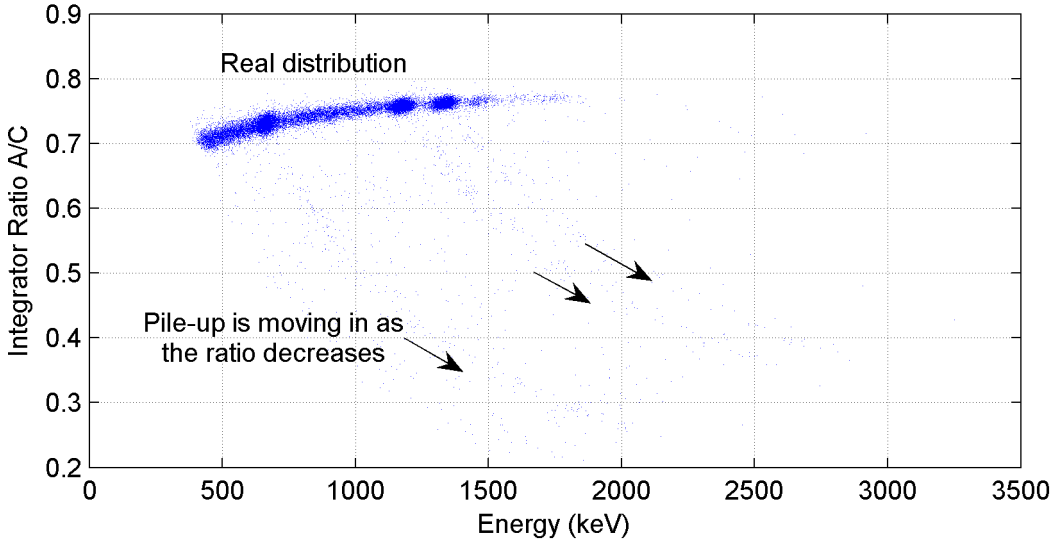

Figure 4.13 Pile-up detection plot of weak ^{137}Cs (662 keV) and ^{60}Co (1173 and 1332 keV) gamma-ray sources using a CsI(Na) scintillator

To detect and remove these pile-up events, a two-dimensional threshold must be applied to keep the real distribution, but to remove the rest. This was done by fitting a best fit

line through the locus of pulses and defining an upper and lower bound in terms of a percentage. Anything outside of those bounds is then tagged as pile-up.

The energy spectrum of the data can now be generated with pile-up removed. It is important to fit the energy spectrum so that we can calculate the full-width half-maximum (FWHM) and determine the effect of the chip (noise) on the ultimate performance. The standard figure-of-merit used by experimenters is the FWHM, expressed in percent, a measure that is reported in the figures.

MATLAB was instructed to fit the histogram data to a distribution which is the sum of three Gaussian distributions and a Beta distribution. The Gaussian distributions are each parameterized by a standard deviation and a mean. The Beta distribution has two parameters and is a simple way to describe the pulse-heights from Compton scattering. It is important to appreciate that this "background" results from a physical process and not from a detector or electronics malfunction.

We "suggest" values for all parameters and then the optimizer within MATLAB refines these initial guesses. MATLAB returns a standard deviation and a mean for each Gaussian. We take the standard deviation and multiply by a factor of 2.35 to get the FWHM. Finally, we take the FWHM and divide by the mean. This value, expressed as a percentage, is annotated on the plots. This procedure produced the best fit curves present in figure 4.14.

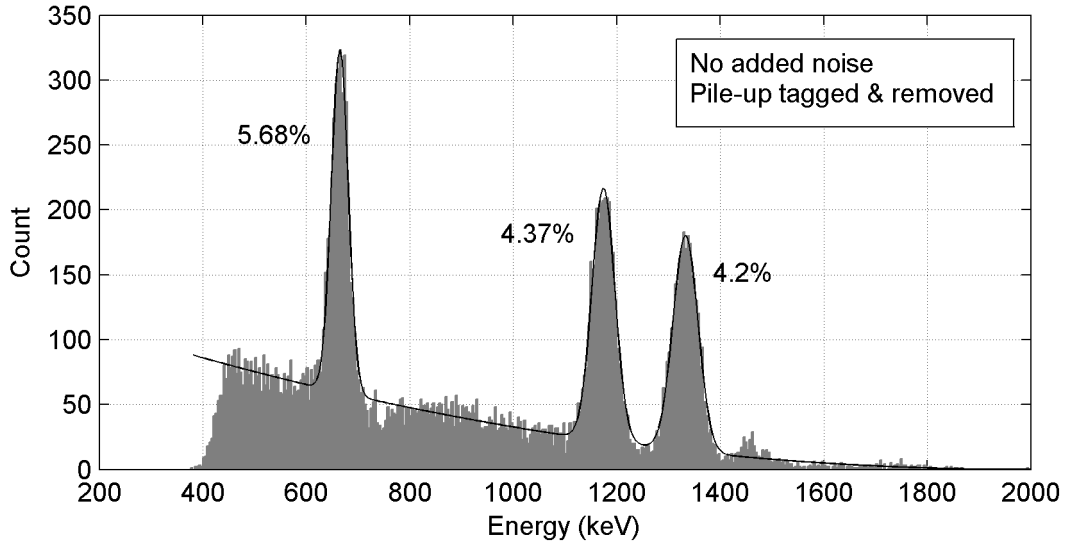

Figure 4.14 Energy spectrum of weak ^{137}Cs (662 keV) and ^{60}Co (1173 and 1332 keV) gamma-ray sources using a CsI(Na) scintillator (ideal, noiseless system)

The energy spectrum in figure 4.14 contains no additional noise, and pile-up events have been removed. The noise associated with the PSD8C IC was added to the integrator outputs. The energy spectrum with noise included is shown in figure 4.15.

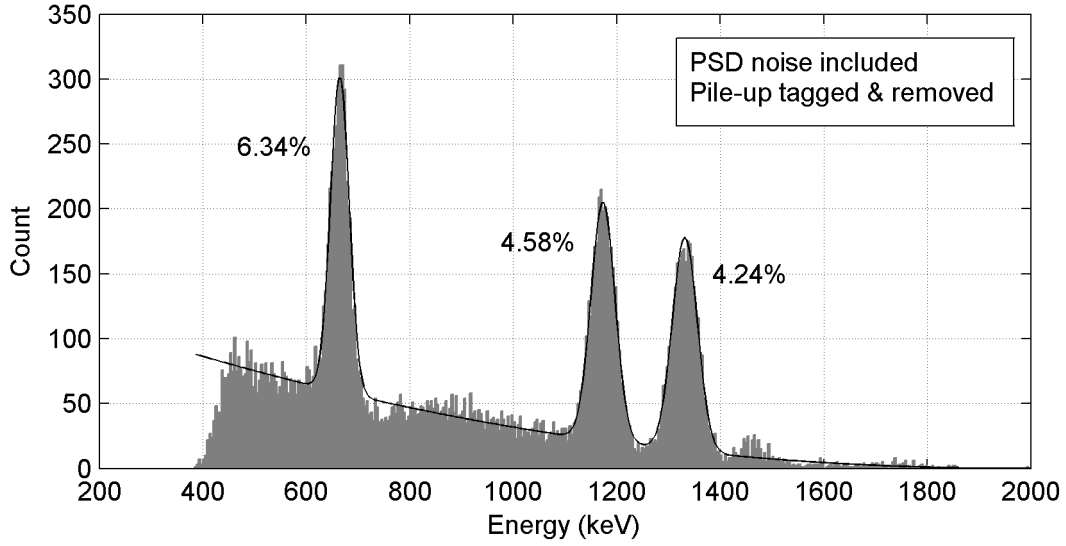

Figure 4.15 Energy spectrum of weak ^{137}Cs (662 keV) and ^{60}Co (1173 and 1332 keV) gamma-ray sources using a CsI(Na) scintillator (PSD8C chip noise added)

As seen in these two figures, the effect of the chip's noise is not significant. The width of the first Gaussian increased from 5.68% to 6.34%. The second increased in width from 4.37% to 4.58%. The third increased in width from 4.20% to 4.24%.

A summary of the pulse characteristics is given in table 4.8. In this table, there are three categories of pulses: saturated pulses, pile-up events, and "good" pulses. The saturated pulses are pulses which have exceeded the range of the analog-to-digital converter (ADC) and thus have samples equal to the maximum ADC value. Pile-up events are those pulses tagged by the pile-up detection algorithm which was designed to use the ratio between two integrators and the pulse energy. "Good" pulses are the pulses which have not saturated the ADC and do not contain pile-up.

	Count	**Percentage**
Saturated pulses	392 / 19998	1.96%
Pile-up events	1076 / 19998	5.38%
Good pulses	18530 / 19998	92.66%

Table 4.8 Breakdown of pulse data (excluding corrupted pulses).

CHAPTER 5

SUMMARY AND FUTURE WORK

Summary

In nuclear physics experiments, the type of incident radiation must be classified, the energy of the particle striking the radiation detector must be determined, and the position within the detector where the energy was deposited must be estimated. When a radiation detector is exposed to incident radiation, the output of the detector is a charge packet where the magnitude of the charge packet is proportional to the energy of the particle striking the detector. After amplification and conversion to a voltage, the resulting pulse (modeled as the sum of several exponentials) may be analyzed to extract the aforementioned parameters that are of interest to nuclear physicists. While the parameters may be extracted in many different ways, this work studied analog-intensive systems that employ multiple charge integration to extract the information. This method is not only effective but produces systems that are small (and relatively speaking, inexpensive). Alternative DSP-based approaches can claim neither of these advantages.

This thesis presented design considerations for several proposed systems employing multiple integrators. The performance of such systems depends upon the number of gated integrators, as well as the signal-to-noise ratio (SNR) of the integrators. A detailed analysis of the SNR characteristics of a gated integrator was presented in Chapter 2. All noise sources were identified and modeled. The noise sources which were identified and carefully analyzed include:

- Poisson noise,
- quantization noise of the ADC,

- noise induced by the time jitter associated with both the discriminator signal (that heralds the onset of the pulse) and by the gate generators,
- thermal noise of the integrating resistor,
- thermal and 1/f noise of the op amp sampled on to the integrating capacitor,
- and the continuous input-referred thermal and 1/f noise of the op amp.

A model for a non-ideal integrator was presented. It was determined that a non-ideal integrator (constructed from an op amp possessing finite bandwidth) can be modeled as the cascade of a gain block, followed by a low-pass filter, followed by an ideal integrator. It was also found that finite gain effects were not important provided the finite gain of the op amp exceeded 40 dB.

Given the input noise spectral characteristics and the transfer function of the integrator, accurate (yet compact) equations predicting the noise at the output of the gated integrator were derived. These "noise equations" were verified against the results of time-domain simulations. The results were in *very close* agreement. The derivation of a compact set of equations that can be used to predict the output noise of a gated integrator is significant because time-domain simulations are tedious and slow. It was determined that the SNR of the gated integrator is a strong function of the input signal shape, the region of integration, and the noise properties of the integrator op amp.

A study of a particle identification (PID) system employing pulse-shape discrimination (PSD) using two integrators was highlighted in Chapter 3. The proposed system makes use of a newly developed multi-channel integrated circuit (PSD8C). In fact, much of the work presented in this thesis was used to guide the design of the chip. Notably, the 1/f noise performance of the integrator op amp used while of little importance for systems using liquid scintillation detectors (fast), played a large role in determining the performance of systems

using CsI(Tl) scintillation detectors (slow). Poisson noise and not electronics noise determined the performance of CsI-based systems.

Particle identification is performed by considering an angle, θ. The angle can be computed as the arc tangent of A over B where A is the output of the A integrator (early) and B is the output of the B integrator (late):

$$\theta = \tan^{-1}\left(\frac{A}{B}\right)$$

For two particles, the above equation would yield the angles θ_1, and θ_2. By defining a threshold between these angles, we can discriminate between the particles.

Since the SNR for the gated integrators could be computed efficiently by using a compact set of analytically derived equations rather than through a series of long computationally intensive time-domain simulations, it was possible to determine optimum integration regions. Before finding the optimal regions of integration it was necessary to determine the variance of the angle, θ, and to define a figure-of-merit (FOM). In Chapter 3, it was demonstrated that the variance of θ was dependent on both the angle, θ, and on the SNR of the individual integrators.

In Chapter 4, we conclude that the proposed PID system will work well with both fast and slow detectors, such as organic liquids (e.g. BC 501) and CsI(Tl) scintillation detectors, respectively. For liquid scintillation detectors with a full-scale energy range of 10 MeVee, simulation shows a discrimination threshold (1% error of misclassification) of 1.44 MeVee resulting in a dynamic range of 17 dB. For CsI(Tl) scintillation detectors with a full-scale energy range of 100 MeV, simulation shows a discrimination threshold of 1.55 MeV and a dynamic range of 36 dB.

In addition to using the PSD8C IC for particle identification, it is possible to use the chip for experiments where total pulse-height information is needed. The PSD8C chip

consists of 8 channels with each channel composed of 3 gated integrators. The thesis demonstrates that one of the integrators can be used to integrate the pulse to obtain energy while the other two integrators can be used implement a pile-up detector.

Pulse data from an experiment using a prototype CsI(Na) detector was analyzed and used to generate an energy spectrum. The energy spectrums for a "noise-free" and a "noisy" (additive noise consistent with PSD8C chip) system were compared. As these pulse data were collected with weak ^{137}Cs (662 keV) and ^{60}Co (1173 and 1332 keV) gamma-ray sources, the generated spectrum contained three Gaussian peaks corresponding to the photopeaks. As presented in Chapter 4, the effect of the chip's noise is not significant. The width of the first Gaussian increased from 5.68% to 6.34%. The second increased in width from 4.37% to 4.58% while the third increased in width from 4.20% to 4.24%.

Finally, while not analyzed in any detail because the design is incomplete, a system using a technique that we developed called Analog-Assisted Digital Signal Processing (AA-DSP) was described. In this system each channel would comprise of approximately two dozen gated integrators, organized in a round-robin integration scheme. The system will be capable of extracting detailed pulse-shape information. This information will be used to produce higher quality energy estimates, superior particle identification capability, and information regarding where within a detector the particle deposited its energy.

Future Work

While equations predicting the noise expected at the output of the gated integrator are presented and verified (using simulation), the analytical predictions should be validated using real-world data. Off-the-shelf gated integrators using op amps with well-characterized noise properties should be designed and tested. Measurements made on these circuits should be compared to the results presented in Chapter 2.

Moreover, when the PSD8C IC returns from fabrication, tests must be performed on the IC. We plan to build a test system where the HiRSA pulse waveforms used in Chapter 4 to generate the energy spectrum can be "played" into the IC. The integrator outputs will be digitized and captured. Energy spectrums will be re-generated and compared to those presented in this thesis. Other tests will be performed on the PSD8C IC to confirm the predictions made in this thesis.

Finally, we would like to build time-domain simulation models for both time-domain white noise and 1/f noise sources that we can use when we perform electrical simulations (using the Cadence *Spectre* simulator) of the transistor-level circuits that are used in the PSD8C and future IC designs. These models will be written using the VerilogA modeling language. At present, *Spectre* only provides noise levels in the frequency-domain.

REFERENCES

Bryan, William L., Charles L. Britton, John T. Mihalczo, John S. Neal, Sara A. Pozzi, and Raymond W. Tucker. "Fast neutron - gamma pulse shape discrimination of liquid scintillation signals for time correlated measurements." *2003 IEEE Nuclear Science Symposium Conference Record*. vol. 2. 2003, 1192-1195. *IEEE Xplore*. IEEE. 4 Dec. 2007.

Engel, George L. "Annual Report (2007) to NSF reporting progress." 2007. <http://www.ee.siue.edu/~gengel/PSDproject/NSFReport2007.pdf>.

Engel, George L., Muthukumar Sadasivam, Mythreyi Nethi, Jon M. Elson, Lee G. Sobotka, Robert J. Charity. "A multi-channel integrated circuit for use in low- and intermediate-energy nuclear physics—HINP16C." *Nuclear Instruments and Methods in Physics Research* 573.3 (2007): 418-426.

Engel, George L., Lee G. Sobotka, and Michael Famiano. "NSF-MRI proposal." Jan. 2006. For the development of a PSD chip. <http://www.ee.siue.edu/~gengel/PSDproject/NSFproposalPSD.pdf>.

Knoll, Glenn F. *Radiation Detection and Measurement*. John Wiley & Sons, Inc., 2000, 3rd ed.

Marisaldi, Martino, Claudio Labanti, and Heike Soltau. "A pulse shape discrimination gamma-ray detector based on a silicon drift chamber coupled to a CsI(Tl) scintillator: prospects for a 1 keV – 1 MeV monolithic detector." *IEEE Transactions on Nuclear Science* 51.4 (2004): 1916-1922. *IEEE Xplore*. IEEE. 7 Dec. 2006.

Nayak, B K., E T. Mirgule, and R K. Choudhury. "Application of pulse shape discrimination in Si detector for fission fragment angular distribution measurements." *Pramana – Journal of Physics* 65.6 (2005): 1053-1059. 14 Dec. 2006 <http://www.iisc.ernet.in/pramana/v65/p1053/fulltext.pdf>.

Razavi, Behzad. *Design of Analog CMOS Integrated Circuits*. New York, NY: McGraw-Hill Higher Education, 2001.

Sobotka, Lee G. Personal communication with. Professor, Chemistry Department, Washington University, Saint Louis, MO.

Streetman, Ben G., and Sanjay Kumar Banerjee. *Solid State Electronic Devices*. Saddle River, NJ: Prentice Hall, 2006, 6th ed.

Terry, Stephen C., Benjamin J. Blalock, James M. Rochelle, M. Nance Ericson, and Sam D. Caylor. "Time-domain noise analysis of linear time-invariant and linear time-variant systems using MATLAB and HSPICE." *IEEE Transactions on Nuclear Science* 52.3 (2005): 805-812. *IEEE Xplore*. IEEE. 3 Dec. 2007.

Yates, Roy D., and David J. Goodman. *Probability and Stochastic Processes*. Hoboken, NJ: John Wiley & Sons, Inc., 2005, 2nd ed.

APPENDIX A

SUMMARY OF NOISE DERIVATION EQUATIONS

Poisson: $$\sigma_p^{\,2} = \frac{q \cdot Ar_{GAIN}}{\tau_{INT} + \tau_u} \cdot \left| V_{OUT} \right|$$

Jitter: $$V_{OF,i} = E_{rad} \cdot \frac{k_{\det} \cdot q \cdot Ar_{GAIN}}{\tau_{\text{int}} + \tau_u} \cdot \frac{A}{\tau_{Fi} - \tau_{Ri}} \cdot \tau_{Fi} \cdot e^{-\frac{Ti}{\tau_{Fi}}} \cdot \left(1 - e^{-\frac{T}{\tau_F}}\right) \qquad c_{i,Ti} = -\left(\frac{V_{OF,i}}{\tau_{Fi}} + \frac{V_{OR,i}}{\tau_{Ri}}\right)$$

$$V_{OR,i} = -E_{rad} \cdot \frac{k_{\det} \cdot q \cdot Ar_{GAIN}}{\tau_{\text{int}} + \tau_u} \cdot \frac{A}{\tau_{Fi} - \tau_{Ri}} \cdot \tau_{Ri} \cdot e^{-\frac{Ti}{\tau_{Ri}}} \cdot \left(1 - e^{-\frac{T}{\tau_{Ri}}}\right) \qquad c_{i,T} = \frac{V_{OF,i}}{\tau_{Fi}} \frac{e^{-T/\tau_{Fi}}}{1 - e^{-T/\tau_{Fi}}} + \frac{V_{OR,i}}{\tau_{Ri}} \frac{e^{-T/\tau_{Ri}}}{1 - e^{-T/\tau_{Ri}}}$$

$$\sigma_j^{\,2} = \left(\sum_{i=1}^{n} c_{i,Ti}\right)^2 \sigma_{Ti}^{\,2} + \left(\sum_{i=1}^{n} c_{i,T}\right)^2 \sigma_T^{\,2}$$

where i = 1, 2, ..., n
for n exponentials

Integrating resistor: $$\sigma_{RI,t}^{\,2} = 4 \frac{k_B T_J}{C_{\text{int}}} \cdot \frac{T}{\tau_{\text{int}}} \cdot \left(\frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u}\right)^2$$

OTA: $$\sigma_{OTA+,t}^{\,2} = k_B T_J \cdot RN \cdot \frac{1}{\tau_o} \cdot \left(\frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u}\right)^2 \qquad \sigma_{OTA+,f}^{\,2} = k_f \cdot \ln\left(\frac{t_{cal}}{\tau_o}\right) \cdot \left(\frac{\tau_{\text{int}}}{\tau_{\text{int}} + \tau_u}\right)^2$$

$$\sigma_{OTA,t}^{\,2} = \sigma_{RI,t}^{\,2} \frac{RN}{R_{\text{int}}} \qquad \sigma_{OTA,f}^{\,2} = \sigma_{OTA+,f}^{\,2} \cdot \left(\frac{T}{\tau_{\text{int}}}\right)^2$$

ADC: $$\sigma_{ADC}^{\,2} = \frac{1}{12}\left(\frac{V_{range}}{2^{bits}}\right)^2$$

Total: $$\sigma_{VOUT}^{\,2} = \sigma_p^{\,2} + \sigma_j^{\,2} + \sigma_{RI,t}^{\,2} + \sigma_{OTA+,t}^{\,2} + \sigma_{OTA,t}^{\,2} + \sigma_{OTA+,f}^{\,2} + \sigma_{OTA,f}^{\,2} + \sigma_{ADC}^{\,2}$$

APPENDIX B

MATLAB CODE USED FOR OPTIMIZATION

LoadPSD.m

```
% Load PSD Definitions
PSD.Constants.q = 1.602e-19;
PSD.Constants.k = 1.381E-23;

PSD.Detectors(1).Name = 'CsI(Tl)';
PSD.Detectors(1).epi_con = 0.17;
PSD.Detectors(1).Emax = 100e6;
PSD.Detectors(1).EnergyUnits = 'MeV';
PSD.Detectors(1).T = 10000e-9;
PSD.Detectors(1).Fs = 1e9;
PSD.Detectors(1).Integrators(1).Name = 'A';
PSD.Detectors(1).Integrators(1).TauIndex = 7;
PSD.Detectors(1).Integrators(2).Name = 'B';
PSD.Detectors(1).Integrators(2).TauIndex = 6;
PSD.Detectors(1).Axis.XMin = 0;
PSD.Detectors(1).Axis.XMax = 10000;
PSD.Detectors(1).Constraints.DAmin = 0;
PSD.Detectors(1).Constraints.DAmax = 0;
PSD.Detectors(1).Constraints.WAmin = 10e-9;
PSD.Detectors(1).Constraints.WAmax = 1000e-9;
PSD.Detectors(1).Constraints.DBmin = 10e-9;
PSD.Detectors(1).Constraints.DBmax = 2000e-9;
PSD.Detectors(1).Constraints.WBmin = 10e-9;
PSD.Detectors(1).Constraints.WBmax = 2000e-9;
PSD.Detectors(1).BiasMode = 1;

PSD.Detectors(2).Name = 'Liquid Scintillator';
PSD.Detectors(2).epi_con = 0.045;
PSD.Detectors(2).Emax = 10e6;
PSD.Detectors(2).EnergyUnits = 'MeVee';
PSD.Detectors(2).T = 250e-9;
PSD.Detectors(2).Fs = 10e9;
PSD.Detectors(2).Integrators(1).Name = 'A';
PSD.Detectors(2).Integrators(1).TauIndex = 3;
PSD.Detectors(2).Integrators(2).Name = 'B';
PSD.Detectors(2).Integrators(2).TauIndex = 1;
PSD.Detectors(2).Axis.XMin = 0;
PSD.Detectors(2).Axis.XMax = 250;
PSD.Detectors(2).Constraints.DAmin = 0;
PSD.Detectors(2).Constraints.DAmax = 0;
PSD.Detectors(2).Constraints.WAmin = 1e-9;
PSD.Detectors(2).Constraints.WAmax = 200e-9;
PSD.Detectors(2).Constraints.DBmin = 0;
PSD.Detectors(2).Constraints.DBmax = 200e-9;
PSD.Detectors(2).Constraints.WBmin = 1e-9;
PSD.Detectors(2).Constraints.WBmax = 200e-9;
PSD.Detectors(2).BiasMode = 2;

PSD.Detectors(1).Pulses(1).Name = 'Alpha';
PSD.Detectors(1).Pulses(1).vA = [10 10]*1e-9;
PSD.Detectors(1).Pulses(1).vTauF = [200 7000]*1e-9;
PSD.Detectors(1).Pulses(1).vTauR = 0.01*PSD.Detectors(1).Pulses(1).vTauF;
PSD.Detectors(1).Pulses(1).NormMax = CalcNormPulseMax(PSD.Detectors(1).Pulses(1),
   0, 0);

PSD.Detectors(1).Pulses(2).Name = 'Proton';
PSD.Detectors(1).Pulses(2).vA = [10 10];
```

```
PSD.Detectors(1).Pulses(2).vTauF = [700 7000]*1e-9;
PSD.Detectors(1).Pulses(2).vTauR = 0.01*PSD.Detectors(1).Pulses(2).vTauF;
PSD.Detectors(1).Pulses(2).NormMax = CalcNormPulseMax(PSD.Detectors(1).Pulses(2),
    0, 0);

PSD.Detectors(2).Pulses(1).Name = 'Gamma';
PSD.Detectors(2).Pulses(1).vA = [1]*1e-9;
PSD.Detectors(2).Pulses(1).vTauF = [10]*1e-9;
PSD.Detectors(2).Pulses(1).vTauR = 0.01*PSD.Detectors(2).Pulses(1).vTauF;
PSD.Detectors(2).Pulses(1).NormMax = CalcNormPulseMax(PSD.Detectors(2).Pulses(1),
    0, 0);

PSD.Detectors(2).Pulses(2).Name = 'Neutron';
PSD.Detectors(2).Pulses(2).vA = [1 0.2]*1e-9;
PSD.Detectors(2).Pulses(2).vTauF = [10 25]*1e-9;
PSD.Detectors(2).Pulses(2).vTauR = 0.01*PSD.Detectors(2).Pulses(2).vTauF;
PSD.Detectors(2).Pulses(2).NormMax = CalcNormPulseMax(PSD.Detectors(2).Pulses(2),
    0, 0);

PSD.LTI_Integrators.Ideal.Name = 'Ideal';
PSD.LTI_Integrators.Ideal.Type = 0;

PSD.LTI_Integrators.NonIdeal.WorstGBW.Name = 'NonIdeal WorstGBW';
PSD.LTI_Integrators.NonIdeal.WorstGBW.Type = 1;
PSD.LTI_Integrators.NonIdeal.WorstGBW.A0  = -4954;
PSD.LTI_Integrators.NonIdeal.WorstGBW.GBW = 34.1e6;
PSD.LTI_Integrators.NonIdeal.WorstGBW.FP  = 65e6;
PSD.LTI_Integrators.NonIdeal.WorstGBW.wd  =
    2*pi*PSD.LTI_Integrators.NonIdeal.WorstGBW.GBW/PSD.LTI_Integrators.NonIdeal.Wor
    stGBW.A0;
PSD.LTI_Integrators.NonIdeal.WorstGBW.wp  =
    2*pi*PSD.LTI_Integrators.NonIdeal.WorstGBW.FP;

PSD.LTI_Integrators.NonIdeal.WorstPM.Name = 'NonIdeal WorstPM';
PSD.LTI_Integrators.NonIdeal.WorstPM.Type = 1;
PSD.LTI_Integrators.NonIdeal.WorstPM.A0  = -4954;
PSD.LTI_Integrators.NonIdeal.WorstPM.GBW = 41.2e6;
PSD.LTI_Integrators.NonIdeal.WorstPM.FP  = 57e6;
PSD.LTI_Integrators.NonIdeal.WorstPM.wd  =
    2*pi*PSD.LTI_Integrators.NonIdeal.WorstPM.GBW/PSD.LTI_Integrators.NonIdeal.Wors
    tPM.A0;
PSD.LTI_Integrators.NonIdeal.WorstPM.wp  =
    2*pi*PSD.LTI_Integrators.NonIdeal.WorstPM.FP;
PSD.LTI_Integrators.V = {PSD.LTI_Integrators.Ideal,
    PSD.LTI_Integrators.NonIdeal.WorstGBW, PSD.LTI_Integrators.NonIdeal.WorstPM};

PSD.OTA.NoiseModel(1).Name = 'Low Bias';
PSD.OTA.NoiseModel(1).RN = 25000;
PSD.OTA.NoiseModel(1).KF_flicker = 1.93e-9;
PSD.OTA.NoiseModel(2).Name = 'High Bias';
PSD.OTA.NoiseModel(2).RN = 7400;
PSD.OTA.NoiseModel(2).KF_flicker = 9.12e-10;
PSD.OTA.GBW = 50e6;

PSD.Chip.vTau = [5 10 20 50 100 200 500 1000]*1e-9;
PSD.Chip.C = 10e-12;
PSD.Chip.Vi_max = 2;
PSD.Chip.Vo_max = 1;
PSD.Chip.T = 300;
PSD.Chip.kT = PSD.Constants.k*PSD.Chip.T;

PSD.EnergyConv.Evis = 3;                    % Energy of visible photon (eV)
PSD.EnergyConv.epi_coll = 0.8;              % Visible light collection efficiency
PSD.EnergyConv.epi_q = 0.25;                % Photocathode Quantum efficiency
```

```
PSD.ADC.Bits = 12;
```

CalcKdet.m

```
function Kdet = CalcKdet(Common)
% Calculate Kdet

Kdet = Common.D.epi_con .* Common.EC.epi_coll .* Common.EC.epi_q ./ ...
       Common.EC.Evis;
```

ChooseARgain.m

```
function ARgain = ChooseARgain(Common, Pulses)
% Choose the appropriate transresistive gain for the given pulses

% Setup Variables
Vi_max = Common.Chip.Vi_max;
Emax = Common.D.Emax;
q = Common.Constants.q;
Kdet = Common.D.Kdet;

vARgain = zeros(1,size(Pulses,2));
for i = 1:size(Pulses,2)
    vARgain(i) = Vi_max/(Emax*Kdet*q*Pulses(i).NormMax);
end

ARgain = min(vARgain);
```

CalcNormPulse.m

```
function [Z] = CalcNormPulse(Model, t, IsoExp, IsoRiseFall)
% Calculate the Normalized Pulse

vA = Model.vA(:);
vTauF = Model.vTauF(:);
vTauR = Model.vTauR(:);

sumA = sum(vA);

if sumA == 0
    sumA = 1;
end

if ~exist('IsoExp')
    IsoExp = 1:size(vA,1);
elseif IsoExp == 0
    IsoExp = 1:size(vA,1);
else
    IsoExp = intersect(IsoExp, 1:size(vA,1));
end

KF = 0; KR = 0;
if ~exist('IsoRiseFall')
    KF = 1; KR = 1;
elseif IsoRiseFall == 0
    KF = 1; KR = 1;
elseif IsoRiseFall == 1
    KF = 1;
elseif IsoRiseFall == 2
    KR = 1;
end
```

```
[mA, mT] = ndgrid(vA(IsoExp), t);
[mTauF, mT] = ndgrid(vTauF(IsoExp), t);
[mTauR, mT] = ndgrid(vTauR(IsoExp), t);

Z = sum(1./sumA .* mA ./ (mTauF - mTauR) .* (KF*exp(-mT./mTauF) - KR*exp(-
    mT./mTauR)) .* (mT >= 0), 1);
```

CalcNormPulseMax.m

```
function Value = CalcNormPulseMax(Model, IsoExp, IsoRiseFall)
% Calculate the Normalized Pulse Max

if ~exist('IsoExp')
    IsoExp = 0;
end
if ~exist('IsoRiseFall')
    IsoRiseFall = 0;
end

Value = CalcNormPulse(Model, ...
    fminsearch(@(t) -CalcNormPulse(Model, t, IsoExp, IsoRiseFall), 1e-12),
...
    IsoExp, IsoRiseFall);
```

CalcIntgNormPulse.m

```
function [Z] = CalcIntgNormPulse(Model, t1, t2, IsoExp, IsoRiseFall)
% Calculate the integral of the normalized pulse.

vA = Model.vA(:);
vTauF = Model.vTauF(:);
vTauR = Model.vTauR(:);

sumA = sum(vA);

if sumA == 0
    sumA = 1;
end

if ~exist('IsoExp')
    IsoExp = 1:size(vA,1);
elseif IsoExp == 0
    IsoExp = 1:size(vA,1);
else
    IsoExp = intersect(IsoExp, 1:size(vA,1));
end
if size(IsoExp,2) == 0
    vA = 0;
    IsoExp = 1;
end

KF = 0; KR = 0;
if ~exist('IsoRiseFall')
    KF = 1; KR = 1;
elseif IsoRiseFall == 0
    KF = 1; KR = 1;
elseif IsoRiseFall == 1
    KF = 1;
elseif IsoRiseFall == 2
    KR = 1;
end
```

```
if t2 < t1
    t2 = t1;
end

mA = vA(IsoExp);
mTauF = vTauF(IsoExp);
mTauR = vTauR(IsoExp);

Z = sum(1./sumA .* mA ./ (mTauF - mTauR) .* (...
    KF*mTauF .* (exp(-t1./mTauF) - exp(-t2./mTauF)) - ...
    KR*mTauR .* (exp(-t1./mTauR) - exp(-t2./mTauR))), 1);
```

CalcNormVOFs.m

```
function NormVOFs = CalcNormVOFs(Model, t1, t2)
% Calculate the output voltage for each falling exponential.

NormVOFs = zeros(1,size(Model.vA,2));
for i = 1:size(NormVOFs,2)
    NormVOFs(i) = CalcIntgNormPulse(Model, t1, t2, i, 1);
end
```

CalcNormVORs.m

```
function NormVORs = CalcNormVORs(Model, t1, t2)
% Calculate the output voltage for each rising exponential.

NormVORs = zeros(1,size(Model.vA,2));
for i = 1:size(NormVORs,2)
    NormVORs(i) = CalcIntgNormPulse(Model, t1, t2, i, 2);
end
```

CalcMaxOutputVoltage.m

```
function [VOUTmax] = CalcMaxOutputVoltage(Common, Pulse, Intg, D, W)
% Calculate the maximum output voltage.

% Setup Variables
q = Common.Constants.q;
Kdet = Common.D.Kdet;
ARgain = Common.D.ARgain;
Emax = Common.D.Emax;
GBW = Common.OTA.GBW;

% Setup Equations
TauInt = Common.Chip.vTau(Intg.TauIndex);
TauU = 1/(2*pi*GBW);
GainFactor = TauInt/(TauInt+TauU);
Kout = q*ARgain/TauInt*GainFactor;

% Output Voltage
VOUTmax = Emax*Kdet*Kout*CalcIntgNormPulse(Pulse, D, D+W, 0, 0);
```

CalcJitterNoiseT_V.m

```
function SigmaVOUT = CalcJitterNoiseT_V(Pulse, VOFs, VORs, T, SigmaJitterT)
% Calculates the jitter noise at the output due to the integration period.
```

```
vA = Pulse.vA(:);
vTauF = Pulse.vTauF(:);
vTauR = Pulse.vTauR(:);

VOFs = VOFs(:);
VORs = VORs(:);

SigmaVOUT = abs(sum(VOFs ./ vTauF .* exp(-T ./ vTauF) ./ (1 - exp(-T ./ vTauF)) +
    ...
                    VORs ./ vTauR .* exp(-T ./ vTauR) ./ (1 - exp(-T ./ vTauR)), 1)
    .* SigmaJitterT);
```

CalcJitterNoiseTi_V.m

```
function SigmaVOUT = CalcJitterNoiseTi_V(Pulse, VOFs, VORs, SigmaJitterTi)
% Calculates the jitter noise at the output due to the starting integration.

vA = Pulse.vA(:);
vTauF = Pulse.vTauF(:);
vTauR = Pulse.vTauR(:);

VOFs = VOFs(:);
VORs = VORs(:);

SigmaVOUT = abs(sum(VOFs ./ vTauF + VORs ./ vTauR, 1) .* SigmaJitterTi);
```

CalcJitterNoiseV.m

```
function SigmaVOUT = CalcJitterNoiseV(Pulse, VOFs, VORs, T, SigmaTi, SigmaT)
% Calculates the jitter noise at the output.

SigmaVOUT = sqrt(CalcJitterNoiseTi_V(Pulse, VOFs, VORs, SigmaTi).^2 + ...
                 CalcJitterNoiseT_V(Pulse, VOFs, VORs, T, SigmaT).^2);
```

CalcIntg.m

```
function [Vout, SNR, Extra] = CalcIntg(Common, Pulse, Intg, E, D, W)
% Calculate the output voltage and signal-to-noise ratio for the given
% integrator.

% Setup Variables
q = Common.Constants.q;
kT = Common.Chip.kT;
Kdet = Common.D.Kdet;
ARgain = Common.D.ARgain;
Cint = Common.Chip.C;
GBW = Common.OTA.GBW;

% Setup Equations
TauInt = Common.Chip.vTau(Intg.TauIndex);
Rint = TauInt/Cint;
TauU = 1/(2*pi*GBW);
TauO = 1/(1/TauInt+1/TauU);
GainFactor = TauInt/(TauInt+TauU);
Kout = q*ARgain/TauInt*GainFactor;

% Output Voltages
Vout = E*Kdet*Kout*CalcIntgNormPulse(Pulse, D, D+W, 0, 0);
VOFs = E*Kdet*Kout*CalcNormVOFs(Pulse, D, D+W);
VORs = E*Kdet*Kout*CalcNormVORs(Pulse, D, D+W);

% Poisson Noise
```

```
VarP = Kout*abs(Vout);
SigP = sqrt(VarP);

% Jitter Noise
SigmaTi = Common.Noise.SigmaTi;
SigmaT = Common.Noise.SigmaT;
if D == 0
    SigmaT = sqrt(SigmaTi^2 + SigmaT^2);
    SigmaTi = 0;
end
SigJ = CalcJitterNoiseV(Pulse, VOFs, VORs, W, SigmaTi, SigmaT);
VarJ = SigJ^2;

% Integrating Resistor Thermal Noise
VarRI = 4*kT/Cint*W/TauInt*GainFactor^2;
SigRI = sqrt(VarRI);

% OTA Thermal Noise (Continuous)
RN = Common.OTA.NoiseModel(Common.D.BiasMode).RN;
VarOTAtc = kT*RN*(1/TauO)*GainFactor^2;
SigOTAtc = sqrt(VarOTAtc);

% OTA Thermal Noise (Sampled)
VarOTAts = VarRI*RN/Rint;
SigOTAts = sqrt(VarOTAts);

% OTA 1/f Noise (Continuous)
Kf = Common.OTA.NoiseModel(Common.D.BiasMode).KF_flicker;
Tcal = Common.Noise.Tcal;
VarOTAfc = Kf*log(Tcal/TauO)*GainFactor^2;
SigOTAfc = sqrt(VarOTAfc);

% OTA 1/f Noise (Sampled)
VarOTAfs = VarOTAfc * (W/TauInt)^2;
SigOTAfs = sqrt(VarOTAfs);

% ADC Quantization Noise
ADC_bits = Common.ADC.Bits;
Vo_max = Common.Chip.Vo_max;
Qbin = Vo_max/(2^ADC_bits);
VarADC = Qbin^2/12;
SigADC = sqrt(VarADC);

% Selectively turn off noise sources
if isfield(Common.Noise, 'SuppressPoisson') && Common.Noise.SuppressPoisson == 1
    VarP = 0; SigP = 0;
end
if isfield(Common.Noise, 'SuppressJitter') && Common.Noise.SuppressJitter == 1
    VarJ = 0; SigJ = 0;
end
if isfield(Common.Noise, 'SuppressRI') && Common.Noise.SuppressRI == 1
    VarRI = 0; SigRI = 0;
end
if isfield(Common.Noise, 'SuppressOTAtc') && Common.Noise.SuppressOTAtc == 1
    VarOTAtc = 0; SigOTAtc = 0;
end
if isfield(Common.Noise, 'SuppressOTAts') && Common.Noise.SuppressOTAts == 1
    VarOTAts = 0; SigOTAts = 0;
end
if isfield(Common.Noise, 'SuppressOTAfc') && Common.Noise.SuppressOTAfc == 1
    VarOTAfc = 0; SigOTAfc = 0;
end
if isfield(Common.Noise, 'SuppressOTAfs') && Common.Noise.SuppressOTAfs == 1
    VarOTAfs = 0; SigOTAfs = 0;
end
```

```
if isfield(Common.Noise, 'SuppressADC') && Common.Noise.SuppressADC == 1
    VarADC = 0; SigADC = 0;
end

% Total noise
VarTotal = VarP + VarJ + VarRI + VarOTAtc + VarOTAts + VarOTAfc + VarOTAfs +
   VarADC;
SigTotal = sqrt(VarTotal);

% Signal-to-noise ratio
SNR = Vout/SigTotal;

% Save extra variables
Extra.Vars.GainFactor = GainFactor;
Extra.Vars.TauInt = TauInt;
Extra.Vars.TauU = TauU;
Extra.Vars.TauO = TauO;
Extra.NoiseBreakdown.SigP = SigP;
Extra.NoiseBreakdown.SigJ = SigJ;
Extra.NoiseBreakdown.SigRI = SigRI;
Extra.NoiseBreakdown.SigOTAtc = SigOTAtc;
Extra.NoiseBreakdown.SigOTAts = SigOTAts;
Extra.NoiseBreakdown.SigOTAfc = SigOTAfc;
Extra.NoiseBreakdown.SigOTAfs = SigOTAfs;
Extra.NoiseBreakdown.SigADC = SigADC;
Extra.Output.Vout = Vout;
Extra.Output.SNR = SNR;
Extra.Output.SigTotal = SigTotal;
```

CalcPSD.m

```
function [Theta, varTheta, Extra] = CalcPSD(Common, Pulse, Intgs, E, DA, WA, DB,
   WB)
% Calculate theta and variance of theta used in Pulse-Shape Discrimination.

[Va,SNRa,INTa] = CalcIntg(Common, Pulse, Intgs(1), E, DA, WA);
[Vb,SNRb,INTb] = CalcIntg(Common, Pulse, Intgs(2), E, DB, WB);

if SNRa == 0
    SNRa = 1e10;
end
if SNRb == 0
    SNRb = 1e10;
end

% Calculate the angle and variance in the angle
Theta = angle(Va+j*Vb);
varTheta = sin(2*Theta)^2/4*(1/(SNRa^2)+1/(SNRb^2));
stdTheta = sqrt(varTheta);

% Save extra variables
Extra.Theta = Theta;
Extra.stdTheta = stdTheta;
Extra.ThetaDeg = Theta/pi*180;
Extra.stdThetaDeg = stdTheta/pi*180;
Extra.INT(1) = INTa;
Extra.INT(2) = INTb;
```

ComputeFOM.m

```
function [FOM, Extra] = ComputeFOM(SimParams, E, DA, WA, DB, WB)
% Calculate the Figure-Of-Merit (FOM).  The larger the FOM, the better the
% Pulse-Shape Discrimination will be.
```

```
Common = SimParams.Common;
Pulses = SimParams.Pulses;
Intgs = SimParams.Integrators;

[Theta0, varTheta0, PSD0] = CalcPSD(Common, Pulses(1), Intgs, E, DA, WA, DB, WB);
[Theta1, varTheta1, PSD1] = CalcPSD(Common, Pulses(2), Intgs, E, DA, WA, DB, WB);

% FOM -> Figure-of-Merit
FOM = abs(Theta1-Theta0)/sqrt(varTheta1+varTheta0);

% Perr -> Probability of Error
ThetaThres = (Theta0 + Theta1) / 2;
P01 = normcdf(ThetaThres, Theta1, sqrt(varTheta1));
P10 = 1 - normcdf(ThetaThres, Theta0, sqrt(varTheta0));
if Theta0 > Theta1
    P01 = 1 - P01;
    P10 = 1 - P10;
end
H0 = 0.5; H1 = 0.5;
Perr = P10 * H0 + P01 * H1;

% Save extra variables
Extra.FOM = FOM;
Extra.Perr = Perr;
Extra.PSD(1) = PSD0;
Extra.PSD(2) = PSD1;
```

OptimizeFOM.m

```
function [OptPoint, Saved] = OptimizeFOM(SimParams, OptConstraints, E, N)
% Optimizes the FOM function.

DAmin = OptConstraints.DAmin;
DAmax = OptConstraints.DAmax;
WAmin = OptConstraints.WAmin;
WAmax = OptConstraints.WAmax;

DBmin = OptConstraints.DBmin;
DBmax = OptConstraints.DBmax;
WBmin = OptConstraints.WBmin;
WBmax = OptConstraints.WBmax;

fFOM = @(x) ComputeFOM(SimParams, E, x(1), x(2), x(3), x(4));

maxFOM = -1;
FOM = zeros(N,1);
Regions = zeros(N,4);
for i = 1:N
    DA = unifrnd(DAmin, DAmax);
    WA = unifrnd(WAmin, WAmax);
    DB = unifrnd(DBmin, DBmax);
    WB = unifrnd(WBmin, WBmax);

    Regions(i,:) = abs(fminsearch(@(x) -fFOM(abs(x)), [DA, WA, DB, WB]));
    FOM(i) = fFOM(Regions(i,:));

    if FOM > maxFOM
        maxFOM = FOM;
        OptPoint.DA = Regions(i,1);
        OptPoint.WA = Regions(i,2);
        OptPoint.DB = Regions(i,3);
        OptPoint.WB = Regions(i,4);
    end
```

```
end

Saved.FOM = FOM;
Saved.Regions = Regions;
```

SimOpt.m

```
% Main Simulation (SimOpt.m)

% Clear the workspace and variables.
clear all;
clc;

% Start timer
t0 = clock;

% Load PSD
LoadPSD;

% Setup simulation parameters
SimParams.Common.EC = PSD.EnergyConv;
SimParams.Common.OTA = PSD.OTA;
SimParams.Common.Chip = PSD.Chip;
SimParams.Common.ADC = PSD.ADC;
SimParams.Common.D = PSD.Detectors(1);
SimParams.Common.D.Kdet = CalcKdet(SimParams.Common);
SimParams.Common.Constants = PSD.Constants;
SimParams.Pulses = SimParams.Common.D.Pulses(1:2);
SimParams.Integrators = SimParams.Common.D.Integrators(1:2);
%SimParams.Common.OTA.GBW = 50e6;

SimParams.Common.D.ARgain = ChooseARgain(SimParams.Common, SimParams.Pulses);
switch SimParams.Common.D.Name
    case 'CsI(Tl)'
        SimParams.Common.Noise.SigmaTi = 7e-9;
        SimParams.Common.Noise.SigmaT = 0.5e-9;
        StartingX = [400e-9, 1500e-9, 1500e-9];
    case 'Liquid Scintillator'
        SimParams.Common.Noise.SigmaTi = 1e-9;
        SimParams.Common.Noise.SigmaT = 0.5e-9;
        StartingX = [50e-9, 30e-9, 50e-9];
end
SimParams.Common.Noise.Tcal = 1000;
% SimParams.Common.Noise.SuppressPoisson = 0;
% SimParams.Common.Noise.SuppressJitter  = 0;
% SimParams.Common.Noise.SuppressRI      = 0;
% SimParams.Common.Noise.SuppressOTAtc   = 0;
% SimParams.Common.Noise.SuppressOTAts   = 0;
% SimParams.Common.Noise.SuppressOTAfc   = 0;
SimParams.Common.Noise.SuppressOTAfs   = 1;
% SimParams.Common.Noise.SuppressADC     = 0;


% Setup optimization parameters
E = 2.1e6; E = 0.9e6; E = 1.55e6; E = 0.393e6;
N = 20;
OptConstraints = SimParams.Common.D.Constraints;

%[Vout, SNR] = CalcIntg(SimParams.Common, SimParams.Pulses(1), SimPar-
    ams.Integrators(2), E, 1500e-9, 1500e-9)

% Perform Optimization
% [OptPoint, Saved] = OptimizeFOM(SimParams, OptConstraints, E, N);
% OptPoint.DA = 0;
```

```
% OptPoint.WA = 200e-9;
% OptPoint.DB = 30e-9;
% OptPoint.WB = 172e-9;

fFOM = @(x) ComputeFOM(SimParams, E, 0, x(1), x(2), x(3));

x = abs(fminsearch(@(x) -fFOM(abs(x)), StartingX));
OptPoint.DA = 0;    OptPoint.WA = x(1);
OptPoint.DB = x(2); OptPoint.WB = x(3);

switch SimParams.Common.D.Name
    case 'CsI(Tl)'
        OptPoint.DA = 0;
        OptPoint.WA = 400e-9;
        OptPoint.DB = 1500e-9;
        OptPoint.WB = 1500e-9;
    case 'Liquid Scintillator'
        OptPoint.DA = 0;
        OptPoint.WA = 50e-9;
        OptPoint.DB = 30e-9;
        OptPoint.WB = 50e-9;
end

[FOM, Extra] = ComputeFOM(SimParams, E, OptPoint.DA, OptPoint.WA, OptPoint.DB,
   OptPoint.WB);

VA1 = CalcMaxOutputVoltage(SimParams.Common, SimParams.Pulses(1), SimPar-
   ams.Integrators(1), OptPoint.DA, OptPoint.WA);
VB1 = CalcMaxOutputVoltage(SimParams.Common, SimParams.Pulses(1), SimPar-
   ams.Integrators(2), OptPoint.DB, OptPoint.WB);
VA2 = CalcMaxOutputVoltage(SimParams.Common, SimParams.Pulses(2), SimPar-
   ams.Integrators(1), OptPoint.DA, OptPoint.WA);
VB2 = CalcMaxOutputVoltage(SimParams.Common, SimParams.Pulses(2), SimPar-
   ams.Integrators(2), OptPoint.DB, OptPoint.WB);

format short g
disp([SimParams.Common.D.Name, ' Detector, ', num2str(E*1e-6, 3), ' MeV']);
disp(['A - ', SimParams.Pulses(1).Name, ', B - ', SimParams.Pulses(2).Name]);
disp([' ']);
disp(['DA:  ', num2str(OptPoint.DA*1e9, 5), ' ns']);
disp(['WA:  ', num2str(OptPoint.WA*1e9, 5), ' ns']);
disp(['DB:  ', num2str(OptPoint.DB*1e9, 5), ' ns']);
disp(['WB:  ', num2str(OptPoint.WB*1e9, 5), ' ns']);
disp([' ']);
disp(['FOM:  ', num2str(FOM, 5)]);
disp(['Perr: ', num2str(Extra.Perr*100, 3), '%']);
disp([' ']);
disp(['Theta1: ', num2str(Extra.PSD(1).ThetaDeg, 4), ' deg']);
disp(['Sigma1: ', num2str(Extra.PSD(1).stdThetaDeg, 4), ' deg']);
disp([' ']);
disp(['Theta2: ', num2str(Extra.PSD(2).ThetaDeg, 4), ' deg']);
disp(['Sigma2: ', num2str(Extra.PSD(2).stdThetaDeg, 4), ' deg']);
disp([' ']);
disp(['VA1: ', num2str(VA1, 5), ' V']);
disp(['VB1: ', num2str(VB1, 5), ' V']);
disp(['VA2: ', num2str(VA2, 5), ' V']);
disp(['VB2: ', num2str(VB2, 5), ' V']);
disp([' ']);
disp(['SNRa1: ', sprintf('%6.2f dB', 20*log10(Extra.PSD(1).INT(1).Output.SNR))]);
disp(['SNRb1: ', sprintf('%6.2f dB', 20*log10(Extra.PSD(1).INT(2).Output.SNR))]);
disp(['SNRa2: ', sprintf('%6.2f dB', 20*log10(Extra.PSD(2).INT(1).Output.SNR))]);
disp(['SNRb2: ', sprintf('%6.2f dB', 20*log10(Extra.PSD(2).INT(2).Output.SNR))]);
disp([' ']);
```

```
disp(['TauA: ',
    num2str(SimParams.Common.Chip.vTau(SimParams.Integrators(1).TauIndex)*1e9), '
    ns']);
disp(['TauB: ',
    num2str(SimParams.Common.Chip.vTau(SimParams.Integrators(2).TauIndex)*1e9), '
    ns']);

%[ComputeFOM(SimParams, 5e6, 0, OptPoint.WA, OptPoint.DB, OptPoint.WB);
% ComputeFOM(SimParams, 5e6, 0, 239e-9, 400e-9, 857e-9);
% ComputeFOM(SimParams, 5e6, 0, StartingX(1), StartingX(2), StartingX(3))]

% Stop timer
disp([' ']);
disp(['Finished entire script in ', num2str(etime(clock, t0)), ' seconds!!!'])
```