CMOS Single Photon Avalanche Photodiodes and Digital Control Interface for Lab-on-Chip DNA Analysis Systems

by

Andrew Michael Hakman

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of

Master of Science

in

Integrated Circuits and Systems

Department of Electrical and Computer Engineering

University of Alberta

Abstract

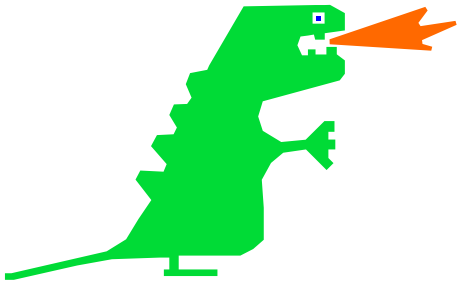
BioMEMS focuses on microelectromechanical systems for biological, or biomedical applications. Combining BioMEMS and CMOS allows highly integrated, complex analysis devices to be realized. One application of particular interest, is medical diagnostic testing. The overall goal of the multi-disciplinary, multi-institutional project to which this research contributes, is a single-use, small, portable, low power, Lab-on-Chip genetic analysis device. Combining all of the electronic structures, including instrumentation, communication, high voltage generation, and optoelectronics onto one CMOS die, and vertically integrating the BioMEMS structures, including heaters, chambers, separation channels, electrostatic valves, and magnetic separation, allows for wafer manufacturing of both CMOS and BioMEMS structures at the same time. Such a device could revolutionize healthcare, by providing inexpensive, fast, point-of-care diagnostics, even in remote regions, without any of the infrastructure currently required to perform such testing.

This work focuses on CMOS Single Photon Avalanche Photodiodes (SPAPDs) for optical detection of fluorescently labeled molecules, such as DNA, suitable for integration in a CMOS - BioMEMS Lab on Chip. The advantages of SPAPDs versus conventional photodiode detectors are higher speed, greater sensitivity, and direct digital output. The digital pulses produced by SPAPDs can eliminate the need for an analog to digital converter for optical detection. The the combination of higher speed and greater sensitivity should allow fluorescence lifetime detection to be achieved, eliminating the need for problematic optical filters.

In addition to the development of SPAPDs, a new SPI-based digital interface was developed for the Lab-on-Chip system. A new modular, addressable register-

based interface was developed, allowing easy changes or additions to any on-chip subsystems.

To my prodigious parents



Acknowledgements

The largetst acknowledgement and thanks has to go to my parents. Your gracious and unwaivering support, not only during this project, but in essentially everything I have ever done, is simply amazing. I can't thank you enough for all your support, and everything you have done for me. It has been greatly appreciated.

I would like to thank my supervisor, Dr. Duncan Elliott, for his support, guidance, and instruction throughout the project.

I would also like to thank all of the other researchers I worked with on this project. It was a pleasure working with all of you: Mohammad Behnam, Maziyar Khorasani, Wesam Al-Haddad, Sunny Ho, Sheng Choi, Benjamin Martin, David Sloan, Saul Caverhill-Godkewitsch, and Matthew Reynolds. I also appreciate the friendships of others, from the Integrated Circuits and Systems lab, and the former High Capacity Digital Communication lab, who were working on other projects: Tyler Brandon, Caitlin Brandon, Leendert van den Berg, Saeed Fouladifard, Brendan Crowley, Philip Marshall, Eric Son, Anthony Ho, Russell Dodd, and last but certainly not least John Koob.

A big thank you to all of my other Edmonton friends. Sometimes it is absolutely essential to spend some time with good friends, doing completely research-unrelated things: crashing model airplanes into trees, dancing, golfing, playing music together, having a picnic in the park, movie nights, or even just chatting about hobbies we have in common! What amazing friends you all are!

I would also like to thank NSERC, Teledyne Dalsa Semiconductor Inc., and the Canadian Microelectronics Corporation for their support of this research, through funding, the fabrication of all of the devices presented in this thesis, and the access to fabrication facilities. In particular, I would like to thank Stephane Martel of Teledyne DALSA Semiconductor for technical insights.

Table of Contents

List of Tables

List of Figures

Nomenclature

List of Acronyms

ADC	Analog to Digital Converter, page 10
APD	Avalanche Photodiode, page 2
CE	Capillary Electrophoresis, page 4
CMOS	Complimentary Metal Oxide Semiconductor, page 1
$\overline{\text{CS}}$	SPI Chip Select, page 65
DIP	Dual In-line Package, page 32
DNA	Deoxyribonucleic acid, page 4
DRC	Design Rule Check, page 17
DUT	Device Under Test, page 32
FPGA	Field Programmable Gate Array, page 62
GAPD	Geiger mode Avalanche Photodiode, page 2
HV	High-voltage, page 9
ITP	Isotachophoresis, page 6
LoC	Lab on Chip, page 2
LoD	Limit of Detection, page 10
LVS	Layout Versus Schematic, page 17
MEMS	Microelectromechanical systems, page 1
MISO	SPI Master In Slave Out, page 65
MOSI	SPI Master Out Slave In, page 65
PCR	Polymerase Chain Reaction, page 4

PMT	Photomultiplier tube, page 7
qPCR	Quantitative Polymerase Chain Reaction, page 5
$\overline{\text{RST}}$	SPI Reset, page 65
SCK	SPI Clock, page 65
SCR	Silicon Controlled Rectifier, page 12
SMU	Source Measurement Unit, page 37
SPAPD	Single Photon Avalanche Photodiode, page 2
TDSI	Teledyne DALSA Semiconductor Inc., page 10

Chapter 1

Introduction

Over the past several decades, CMOS technology has achieved great advances in miniaturization, large scale production, and cost reduction, which in turn has led to electronic devices becoming ubiquitous in modern society. Utilizing these advances, and levering CMOS with other technologies, such as MEMS and Microfluidics, opens the door to a potential vast array of new technological applications. These include drug identification and detecting explosives residues, drugs and endogenous small molecule and oligonucleotide detection, protein and peptide detection, and analysis of nucleic acids and oligonucleotides [1] [2]. One application of particular interest, is in biomedical and healthcare related devices. Making advanced medical diagnoses as ubiquitous, and inexpensive as pure electronic technology is today, would have large positive impact. This is especially true where the infrastructure, or skilled technicians, which are currently required to perform such diagnoses, are unavailable.

The overall goal of this multi-disciplinary, multi-institutional project, is to develop a genetic analysis device, which is miniaturized, fully integrated, and thus the complete instrument can be wafer fabricated, one-time-use, and inexpensive. Further goals are for the instrument to be low mass, and low power consumption, allowing for portable, and or battery operation. Current genetic analyzers, such as the Agilent 2100 bioanalyzer [3], which has only a subset of the functionality, in that it performs capillary electrophoresis, and optical detection only, are large bench-top sized machines, requiring skilled operators, and have substantial price tags. Many

other research efforts, such as that of Chabinyc et al. [4] have integrated the fluidics and optical detectors, but not the remainder of the electronics required for the full system. Previous efforts by Rech et al. [5] have integrated APD arrays for optical detection, but still require large optics, and have not integrated the rest of the controlling electronics, resulting in a much larger, and heavier device. By integrating the electronics and microfluidics on the same wafer, the goal is to have an inexpensive, USB flash-drive-sized device, which does not require a skilled operator. The single-use nature of the device eliminates any potential for cross-contamination between tests, and simplifies operation. A single-use microfluidic system is a virtual requirement for medical approval (e.g. FDA), since in a mult-use system, it is difficult to prove that a subsequent test result will not be contaminated by a previous sample.

The primary focus of this research is to investigate the use of Geiger mode Avalanche Photodiodes, which are also called Single Photon Avalanche Photodiodes, or SPAPDs, for optical detection in a Lab on Chip (LoC) application. To meet the goal of integrating as much of the system as possible onto a single wafer, and to be a viable option for optical detection, the Avalanche Photodiode (APD) must be manufacturable on the same CMOS die, and in the same process as the rest of the system. SPAPDs also offer advantages, in that detected photons are counted, resulting in a direct digital readout, without the need for an ADC. In a LoC device, simple Geiger pulse counting should provide an adequate indication of the amount of light which is incident on the detector. Alternatively, some SPAPD systems use an ADC and digital signal processing to increase sensitivity in counting pulses. SPAPDs are also fast enough to investigate the use of fluorescence lifetime detection in the future, which would eliminate the need for an optical filter to block the excitation light from the detector.

A secondary objective of this research was to improve the digital control interface, between the LoC system, and the external microcontroller.

In Chapter 2, a general overview of genetic analysis is presented, providing context for the remainder of the work, and how it relates to the overall project. In

Chapter 3, the primary focus of this work, which revolves around the optical detector required for the LoC system, is presented. A background of photo detectors, as well as the motivation for investigating the use of Geiger mode Avalanche Photodiodes, will be discussed. The problems encountered in achieving functional Geiger mode devices is also presented, so similar issues can be avoided in the future. The designs of the functional devices will be presented, along withe the results of the characterization performed on all of the devices built. The custom test apparatus, which was assembled to characterize all of the optical devices, is also presented. A great deal of effort was invested in the test apparatus, and while it is not optimal in its current form, it is presented so future work may improve on the current infrastructure.

In Chapter 4, the new salable and modular digital control interface, which was developed to allow easier integration of all of the subsystems required to assemble a functional LoC system, is presented.

In Chapter 5, all of the chips which were fabricated during this work are presented. System chips combine the most current progress in all subsystems into one device, to evaluate the performance of the overall system. A general overview of the design flow, and the rationale of the layout of the system chip, ICKAALC8 is discussed.

Finally, in Chapter 6, the contributions of this research are summarized, and suggested directions for future improvements are presented.

Chapter 2

Background

A background of general LoC techniques will be presented, to give context of how the work of this thesis fits into the overall application. The LoC system under development, depends on a few key techniques, Polymerase Chain Reaction (PCR) for DNA amplification, Capillary Electrophoresis (CE) for the separation of the DNA fragments, and fluorescence detection for the measurement of the DNA fragments separated by CE.

2.1 PCR

PCR is used to amplify the incoming DNA sample to be analyzed. PCR is a commonly used method of DNA amplification [6]. During PCR, the DNA of interest is thermally cycled in the presence of primers, short DNA sequences of interest which are unique to the organism to be detected, and DNA polymerase. During the first stage, denaturation, the PCR mixture is heated, which breaks the bonds of the double helix, producing two single strands of DNA. In the second step, annealing, the mixture is cooled, and the primers bind to complementary sequences of the single strands. In the third stage, elongation, the DNA polymerase binds to the primers, and the remaining unbound portions of the single stranded DNA, and completes the complimentary copy. Each stage of the reaction must be carried out at a specific temperature, which is dependent on the primers and polymerase used. Temperatures as high as 94 °C to 96 °C are required for the denaturation stage of PCR.

Every time the PCR cycle is run, the number of copies of DNA is ideally doubled. This process can be repeated as many times as required, to generate as many copies of the DNA as required for further analysis steps, up to practical limits, including initial concentrations of primers and neuculitides. The system under development is designed to perform PCR in a microfluidic chamber, using a patterned metal heater for temperature control.

A variant of PCR, called quantitative PCR (qPCR), can be used to detect the presence and quantity of a target sequence, as PCR is carried out [7]. Quantitative PCR uses intercolator fluorophore which fluoresces more brightly when attached to double-stranded DNA, such as that produced by PCR. Fluorescent tagged PCR products are monitored using a photo detector, and the presence of fluorescence light is used to determine whether the target sequence of interest exists in the sample. The intensity of the fluorescence light versus the number of PCR cycles performed is used to determine the quantity of copies of that sequence, which have been produced by PCR.

Both of these variants of PCR are useful in LoC applications, and it is desirable that the LoC system be capable of both PCR and qPCR. Both variants also have impacts on the optical detection system. Because PCR involves thermally cycling the contents of the PCR chamber, and because in a fully integrated system the microfluidic PCR chamber is bonded to the CMOS die, performing PCR will undoubtedly cause temperature variations of the entire die, including the photo detector. As will be discussed in more detail in the APD section, the reverse bias threshold of SPAPDs are very sensitive to temperature fluctuation. This will be an important factor in implementing qPCR with SPAPDs, as the photo detector needs to be directly under the PCR chamber, and operating while PCR is being performed.

2.2 Capillary Electrophoresis

Capillary Electrophoresis (CE) is used to separate the various sized DNA fragments, resulting from the PCR amplification. A two-channel, four-well design, is the basis of all the microfluidics implemented thus far in the LoC project at University of

Alberta. This standard design structure can be seen in many other microfluidic devices designed for CE, such as Wooley and Mathies [8] in 1994 and Effenhauser et al. [9] in 1997, among many others. The basic two-channel, four-well design is shown in Figure 2.1.

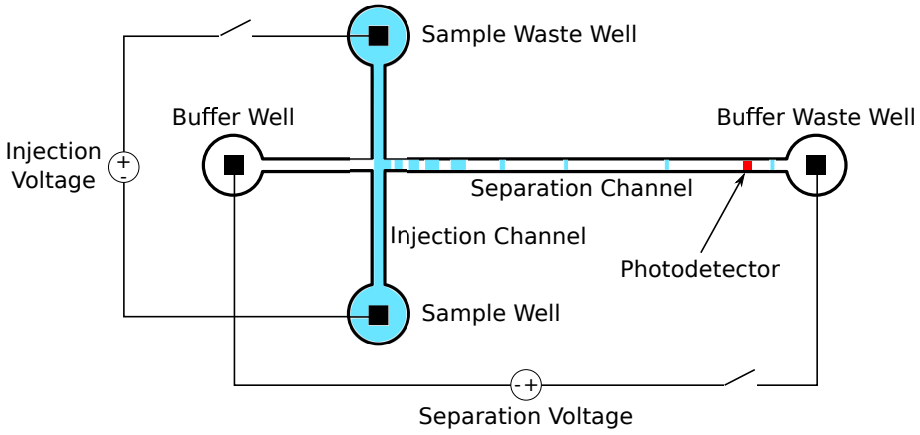


Figure 2.1: Standard two-channel, four-well CE design

The DNA sample of interest is loaded into the sample well. Due to electrophoretic motion of charged molecules resulting from the potential applied between the sample, and sample waste wells, the DNA migrates towards the sample waste well. This capillary is called the injection channel, and is filled with conductive buffer. The injection potential is removed, and a known quantity of analyte is now in the intersection of the injection and separation channels. A separation potential is applied between the buffer, and buffer waste wells, and the injected DNA is pulled through the separation channel towards the buffer waste well. The separation channel is filled with a sieving polymer, which imparts different mobilities to different length DNA fragments. The smaller fragments move though the sieving polymer more easily, and thus more quickly, than the larger fragments. Because of the size-dependent mobility, the timing of the various sized fragments passing over the detector, is how the fragments are identified. The resulting output of the photo detector intensity, versus time since the separation process was started, is the primary measurement obtained by the system, and is called an electropherogram.

A process known as Isotachophoresis (ITP), which is quite similar to CE, and

can use similar microfluidic channels, can be used to detect non-DNA molecules [10].

2.3 Fluorescence Detection

To be able to detect the DNA fragments as they pass by the optical detector, they are first tagged with a fluorescent fluorophore. End-label fluorophores are attached to the PCR primers, resulting in each PCR product having a fluorescent tag attached to it. In the area of the photo detector, light of suitable wavelength for exciting the fluorophore is present. The excitation light source in the LoC project is likely to be an LED, external to the CMOS - BioMEMS wafer. In microfluidic systems, previously the detector of choice was a photomultiplier tube (PMT), which is quite bulky, and requires additional optics to focus the fluorescence light on the PMT [11]. In addition, traditionally, an optical filter attenuates the excitation wavelength, and allows the fluorescent light from the fluorophore to reach the detector. In a fully integrated system, with microfluidic channels directly on top of the CMOS die, integrating such an optical filter can pose significant challenges. An alternative approach is to pulse the excitation light, and measure the fluorescence decay after the excitation light has been removed. This is known as fluorescence lifetime detection [12]. Because the measurement is taken after the excitation light has been removed, there is no need to filter it optically from the detector. This does require a very fast, and very sensitive detector, which is one of the motivations for investigating the use of SPAPDs as the photo detector in LoC devices. SPAPDs meet both the requirements of being fast, and extremely sensitive.

2.4 Summary

The main subsystems of a LoC device, with relevance to the investigation of the use of an integrated APD detector, were presented. A general explanation of the target application, and the overall system operation was given. As well, explanations were given for the motivation for investigating APD-based detection, namely

the possibility of using fluorescence lifetime detection, thus eliminating the need for the optical excitation light filter. The potential effect the microfluidic side of the design could have on integrated APD photo detectors, due to varying die temperature resulting from performing PCR, was also mentioned.

Chapter 3

Avalanche Photodiode Detection for LoC Applications

The optical detector is one of the most critical parts of a LoC system. The performance of the optical system largely determines the performance of the entire system. The high voltage process that is required to generate and switch the high voltages for CE, afford the opportunity to integrate a different kind of optical detector than can easily be built into a regular low-voltage CMOS process. The additional doping levels available in a high-voltage (HV) CMOS process make it possible to implement APDs. In this chapter, a brief overview of photo diode theory will be covered, as well as the motivations for investigating the use of APDs for LoC systems. The special challenges faced when designing and implementing APDs, rather than much simpler standard photo diodes, will also be discussed. Special considerations relating to the test and characterization that was performed on the devices will be discussed, and the characterization and evaluation of the performance of all of the devices that were implemented will be presented. The focus of the results will be on the designs that successfully operate in the desired single-photon-counting Geiger mode. There will also be a brief discussion of more advanced APD quench circuits, including one active quench circuit that was designed and implemented, but not fully characterized.

3.1 Motivation for APD based detection

As mentioned previously, the motivation of the entire project is to build a low-cost, portable, single-use DNA sequencing system, which can be used by relatively inexperienced operators. To meet the goals of low cost and compact size, as much of the system as possible should be implemented on the fewest number of dies, or external devices. Integrating the optical detection system on the same CMOS die as the rest of the system is an essential part of reaching that goal. The optical detection subsystem in a LoC device is one of the more challenging subsystems, as it is the interface between the fluorophore tagged DNA molecules of interest, and the rest of the system. The Limit of Detection (LoD), or the minimum quantity of tagged DNA that can be detected, determines the sensitivity of the entire system [13]. The sensitivity of the photodetector and associated circuits, is a determining factor in the LoD of the system. Improving the performance of this subsystem, while keeping it compatible with the CMOS process used to fabricate the rest of the system, is a high priority in developing a successful and competitive LoC product.

There are some key reasons to specifically investigate the integration and use of APDs as the optical sensor in a LoC system. Geiger mode APDs do not require an analog to digital converter (ADC), as they output digital pulses which are counted with a counter, rather than an analog current or voltage as a conventional photo diode produces. The system has contained an on-chip successive approximation ADC for many generations, implemented by Khorasani [14], and more recently a delta-sigma ADC implemented by Martin [15]. Both have proven to be challenging to implement in the Teledyne TDSI Semiconductor Incorporated (TDSI) CMOSP8G 300V CMOS process with the desired performance. Successful implementation of an APD detection system would eliminate the need for the on-chip ADC in the optical detection path, thus eliminating the problems associated with implementing one.

Another motivating factor in the favor of APD based detection is the possibility of fluorescence lifetime detection [16]. Currently, a continuous excitation light

source is used to illuminate the tagged DNA molecules in the microfluidic channel. An optical filter is required between the microfluidic channel and the optical detector, to filter out the excitation light, only allowing the fluorescence light to reach the detector. Without the filter, the excitation light saturates, or severely reduces the dynamic range of the detector, making it harder to measure the signal of interest, the fluorescence light [15, chapt. 3].

The filter also presents many challenges, in both the design, and the manufacturability of the entire system. There are two types of filters which could be used to attenuate the excitation light, interference filters which offer superior attenuation, and absorption filters which are popular in RGB digital camera sensors. Both types of filters pose challenges when implemented as the excitation light filter in a LoC system. The higher performance interference filters require parallel light to be most effective. This is possible with a longer optical path, but does not occur when the filter is 10 µm above the photodetector, under a 100 µm wide channel. In this configuration, the excitation light emanates from the channel in a more isotropic manner, rather than parallel. Absorption filters have less attenuation in the wavelengths they are designed to block. Increasing the thickness of the filter does not result in a significant increase of attenuation. Some excitation light also results in autofluorescence, and this longer wavelength light can pass through a thicker filter. If the autofluorescent light is emitted isotropically, the further the filter is from the detector, the more attenuation there will be, but this is at odds with CMOS vertical integration. Both types of filters can also be combined, and this has been investigated in work that is directly related to the overall LoC project [17].

Because of the speed and sensitivity of APDs, fluorescence lifetime measurements may be an option, eliminating the requirement for the excitation light filter all together. In fluorescence lifetime measurement, rather than illuminating the sample continuously, the excitation light is pulsed. When the excitation light is removed, the fluorophore will continue to fluoresce for some given amount of time, dependent on the fluorescence lifetime of the fluorophore in use. Cy3, Cy3B, and Cy5 have fluorescence lifetimes of 0.3 ns, 2.8 ns, and 1.0 ns respectively [18]. Some

fluorophores have much longer fluorescence lifetimes, such as Pyrene (>100 ns), $Ru(bpy)_2(dcpby)[PF_6]_2$ (375 ns), and $Ru(bpy)_2[PF_6]_2$ (600 ns) [18]. Given the speed and sensitivity of an APD, it should be possible to measure the fluorescence decay, and because the excitation light is not present at the measurement instant, no filter is required to filter it optically.

Along with these potential advantages, the APDs also present some unique challenges. Because of the very nature of the device, in that it must be run in an unstable region of operation to get the desired functionality, special design considerations are required at the layout level. APDs typically require some additional doping layers, that are deeper, and more lightly doped, than the typical *n+/p+* and one well layer available in a standard CMOS process. These are required to construct guard rings, to precisely control the electric field strength, and thus the region where the avalanche breakdown will occur [19]. The guard rings required in APDs are considerably different than typical CMOS guard rings, which are to prevent SCR latchup. Some methods have been developed to overcome the lack of these layers in standard CMOS processes, however, as in the original work in which the method was developed, likely a full process simulation would be required to achieve the desired result [20, pg. 95]. It is questionable if most semiconductor fabs would be be willing to sufficiently divulge the process recipe, in order to be able to simulate constructing a guard ring using this technique. Thus, implementation of that method might still be quite difficult. Fortunately, this project is already being implemented in the TDSI CMOSP8G HV BiCMOS process, which already contains the required deeper and less heavily doped layers, so additional process steps or special methods need to be employed to construct the required guard rings.

Because this project is being implemented in an existing commercial HV CMOS process, there is no opportunity to adjust any of the process parameters from those specified by TDSI. Therefore the goal of this work is to determine if functional Geiger mode APDs can be designed, and successfully operated in the existing process, so they can be integrated on-chip with the rest of the system, rather than building the best APD possible, which would require process customization.

Even though the extra doping layers, required for easier implementation of APDs, are available in the CMOSP8G process, successfully designing functional Geiger mode APDs proved quite difficult. Many designs, 85 in total, were implemented over 6 tapeouts before functional Geiger Mode APDs were achieved. When implementing a custom device like an APD in a process for the first time, and having no prior experience with functional APDs in any technology, there are not a lot of easy debugging options available to determine why correct operation could not be achieved. Initially, there was also uncertainty whether the problem was with the device, or the test apparatus, which further complicated debugging. A key focus of this thesis is to document the problems encountered, as well as any useful observations that may be helpful to future work in implementing APDs in this process, or for others implementing APDs in any process without prior experience.

3.2 Photodiode and APD Background

Any P-N junction diode can be used in some capacity as a light detector. Because the goal of the overall project is an integrated LoC system on a single die, the discussion of photo detectors will be limited to devices manufacturable in Si VLSI processes. A standard 3 region diode IV curve is shown in Figure 3.1 for reference.

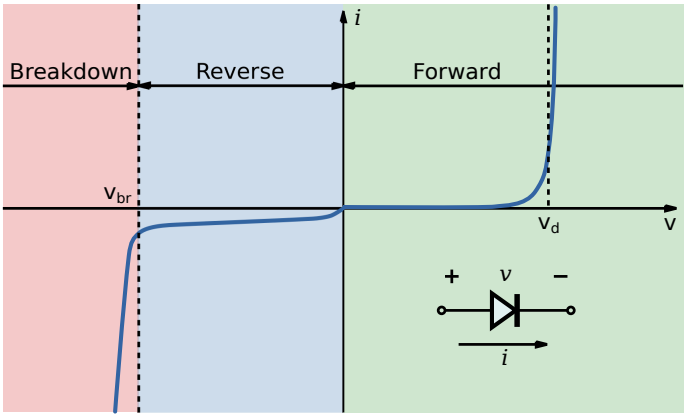


Figure 3.1: Standard diode I-V characteristic. For illustration only. Compressed scale results in 'kink' at origin. (Taken from Wikipedia [21])

A photo diode can be operated in several modes. The primary modes of op-

eration of basic photo diodes are photovoltaic, and photoconductive. Avalanche operation will be explained later in this section.

Photovoltaic mode operates the device with no external bias voltage across it, and the photovoltaic effect produces a photocurrent dependent on the light hitting the device. The advantage of photovoltaic mode is a very low dark current [22], the amount of current generated when no light is striking the device. The photo diodes previously implemented in this project, first by Khorasani [14], and with an improved sense amplifier by Martin [15], operate in this mode.

The other common mode of operation of a basic photo diode, is photoconductive mode. In this mode, the device is externally reverse biased. The reverse bias condition increases the width of the depletion region, which lowers the device capacitance, and increases the speed of the device, and results in slightly higher quantum efficiency [22]. The downside however, is increased dark current, and noise in the sense amplifier resulting from the increased dark current. The amount of reverse bias that can be applied is limited by the reverse bias breakdown threshold of the device, V_{br}, and the acceptable noise level in the system. Because the signal of interest, generated by the fluorophore tagged DNA traveling by the optical sensor, is extremely slow, typically less than 10 Hz, this mode of operation is generally uninteresting for the intended LoC detection application.

Avalanche Photo Diodes

Looking at the standard diode IV curve, there are not many other options for the operating region of a photo detector. Operating in forward bias, but below the threshold voltage, would provide the opposite effects as photoconductive mode. The depletion region would be smaller, resulting in higher device capacitance, and less sensitivity. Operating in forward bias, but above the threshold voltage, leaves no depletion region at all, resulting in no place for photons to be absorbed and turned into carriers. This only leaves the region with reverse bias below the reverse bias threshold. At first glance, it seems counter intuitive that this region can be useful for detection. It is not merely the reverse bias voltage being below the threshold

that causes current to flow though. Some external perturbance is required to start the flow of current [23]. There are three common sources of such an external force which will initiate current flow: an absorbed photon, w energy higher than the band gap energy of the semiconductor material of the detector, resulting in an electron-hole pair being generated; a thermally generated electron-hole pair; or a released trapped carrier [24]. All three of these will result in an ionized carrier pair, which will get swept out of the depletion region by the strong electric field, due to the high reverse bias voltage. Above the reverse bias threshold voltage, the electric field is strong enough to give the carrier enough energy, that when it collides with another lattice atom, leading to the generation of another carrier pair [19]. This is known as impact ionization. The generated carrier pair will also be accelerated by the electric field, and obtain enough energy to cause further impact ionization upon colliding with other lattice atoms. This chain reaction gives rise to the name avalanche photo diode, in that, once one carrier pair is generated, the effect is quickly amplified into a high current from the resulting avalanche of resulting impact ionizations. If not externally controlled, the current would continue to rise until the device is physically damaged. This leads to the obvious need for some external circuitry to quench the current flow once it starts. The device design required to operate in this region, as well as methods of quenching the avalanche to both detect and count events, and to prevent the device from damage from over-current, and overheating, will be discussed in the following sections.

In the past, operating at a bias voltage just below reverse bias breakdown, in the ”near-Geiger” mode was also performed [25]. This is generally not practiced currently, as advancements in active quenching, or utilizing on chip passive quenching, have negated the benefits of near-Geiger operation.

3.3 Special Design Considerations of APDs

As mentioned above, to operate in the Geiger region above the reverse-bias breakdown threshold, very high electric fields are present in the device. Without any special design considerations, the highest electric field strength will occur at the triple

point of the edge of the active region and the field oxide [20]. This peripheral top junction will have a lower breakdown voltage than the bottom plane junction, which is the desired junction for successful operation. The complexity of implementing APDs lies in properly controlling the electric field strengths in the device, in order to have the strongest field strengths at the bottom plane junction of the device. By carefully designing the physical layout of the device, the electric field strength can be lowered at the periphery of the device, leaving the region of highest field strength constrained to the bottom of the device [20]. This is most easily achieved if there are doping layers which are deeper, and more lightly doped, than the layers used for the primary *p-n* junction. Such layers are available in the TDSI CMOSP8G HV process required in this project. The high-voltage process is required for the high voltages necessary for CE. The deeper, lower doping concentration regions, are used to form guard rings around the periphery of the device, lowering the electric field strength around the edge, eliminating the undesired edge breakdown. A general cross-section diagram of an APD is shown in Figure 3.2.

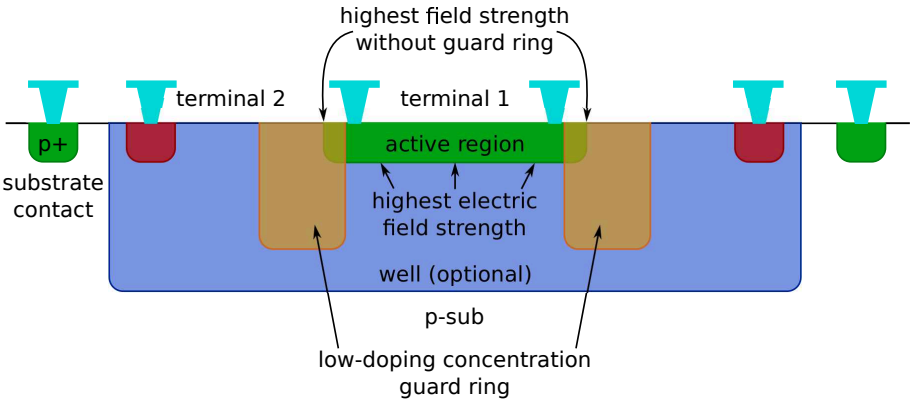


Figure 3.2: Generalized APD cross-section. Guard ring must be of same type as active region, but lower doping concentration. Well must be of opposite type as active region, or if no well is present, active region must be *n-type*.

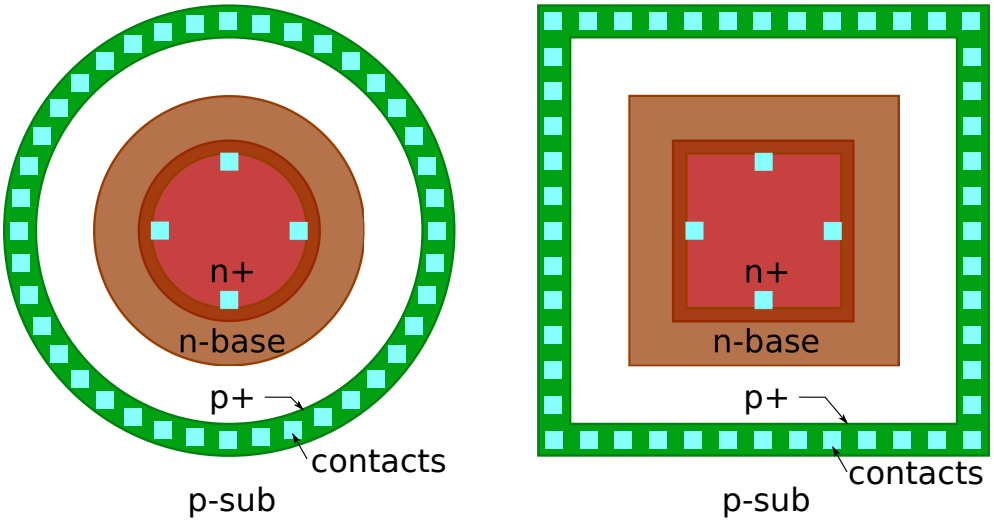
Khorasani [14] designed some initial APD devices, which upon testing were not found to operate in the desirable single-photon-counting Geiger mode. Initially it was thought this lack of Geiger mode operation was due to the device area being very large, and thus the junction capacitance was too high. The devices were sub-

sequently shrunk, and fabricated and tested again. The resulting devices still did not operate in Geiger mode. It was later discovered that there was confusion about the *active* layer, and which doping layers required an overlay of *active* region to actually be present in the physical design, and which did not. In the CMOSP8G process, all layers, except *HV_Def*, *hvnwell*, and *pwell*, require a region of *active* over them to actually be physically present. Because of this confusion, many designs had doping regions, that while appearing correct in the CAD layout, were not being fabricated as intended. Another related problem that was discovered, was using a visualization for the *active* layer, which only showed an outline of the region, rather than one with some shading to clearly indicate whether an area was covered with active or not. The edges of many layers tended to be co-incident in the layouts, easily hiding the edge of the active layer, and leading to confusion as to whether a region was covered with active or not. After these layout problems were discovered, it was found that related problems affected all of the previous designs. This was a major contributing factor, along with several others, that were considered to potentially be be the reason, or reasons Geiger mode operation could not be achieved. Especially in a more complicated CMOS process, such as a high voltage / BiCMOS process that has additional doping layers beyond the usual *p+/n+*, and an *n-type* or *p-type* well or two, it is imperative that one fully understand the intricacies of the CAD layout, versus what is actually being fabricated. Typically DRC (design rule check) or LVS (layout versus schematic) would tend to catch such errors. However, because the DRC rules are generally designed for transistors, or other commonly implemented devices, typically capacitors, normal diodes, or resistors, and the extractor only knows how to interpret the layouts of the same set of devices, they are of no assistance when designing fully-custom non-standard devices such as APDs. It is far too easy to draw what looks like an APD, with correct guard rings, in the CAD program, but to get something completely different, incorrect, and unintended, back in silicon. Without extremely advanced equipment, none of which is typically found in a university level lab, there really is no effective way to verify the fabricated design in silicon. The only method of debugging, is to discover

the devices will not function in Geiger mode when tested, and attempt to determine intuitively what the cause of the problem might be.

Another layout problem that was discovered, was incorrectly assuming the lower doping concentration, and more deeply driven-in layers, would laterally diffuse more than the higher concentration shallow layers used in the primary junction. Several previous designs, both intentionally, and unintentionally, had co-incident edges of the highly doped middle region, and lower doped guard ring regions, due to the layout issues mentioned above. The higher doped region was present across the entire width of the guard ring. It was previously thought that the more deeply driven-in guard ring would laterally diffuse more, and thus form the protective guard ring. As a result of testing the devices that were designed in this way, Geiger operation could not be achieved, and it was surmised to be due to absence of guard ring formation. Additionally, DRC rules should not be used to determine the minimum width of the guard ring. As mentioned previously, the DRC rules are intended for different types of devices and designs entirely, and thus should not be assumed correct when using the layers in a non-standard way, to build non-standard custom devices. If it is imperative to make the guard rings of minimal size in the future, a set of devices with varying guard ring widths should be implemented and tested to determine the minimum guard ring width. This could also be accomplished with process simulation, but because the process recipe is proprietary to TDSI, it is unknown whether they would be comfortable divulging enough information to perform an accurate simulation, and thus process simulation most likely is not an option. If making the devices as small as possible isn't necessary, it is best to be liberal with the guard ring width. As will be discussed further in the results section, some devices were designed to have fairly liberal guard rings, in an attempt to eliminate another possible cause of being unable to achieve Geiger operation, as occurred in previous designs. Not enough variants of the same design, with different width guard rings were built, to actually find the minimum size. All the devices, with all sizes of guard rings built in the final generation, successfully operate in Geiger mode.

In all of the literature consulted during this project, all implemented APDs are generally round, or some approximation thereof, depending on the process used for implementation, minimum grid sizes, and other DRC rules pertaining to allowed angles. In contrast, all of the early devices implemented in this work, which did not function in Geiger mode, were square. The rationale behind the round structures seems to be to avoid sharp corners, and thus higher electric field strengths at those corners. But is this actually an issue when the periphery of the device is already covered with a guard ring, and the highest electric field strengths should be limited to the bottom plane junction? It was postulated that this could be another reason the earlier devices weren't operating correctly, so in the final generation of designs, both round and square versions of all designs were implemented. An example of the same design implemented in both round and square versions is shown in Figure 3.3.



(a) Round (b) Square

Figure 3.3: Plan view showing round vs. square *n+ p-substrate* APD design. Square variant created by replacing round p-cells with square ones, resulting in all key linear dimensions being the same as the round version.

All of the early generation devices had manually drawn layouts, and with the number of layers in each design being upwards of 12, coupled with the sheer number of different designs, each implemented at multiple different sizes, it was very

difficult to systematically change parameters of the designs. A lot of time, and manual labour was required to generate different versions of the designs to try to isolate why they were not functioning in Geiger mode. These problems, coupled with the desire to build round devices, which are prohibitively time consuming to draw correctly by hand, led to the development of a custom pcell for laying out APD devices. While the layout CAD program does have tools for drawing arcs, the vertices used to make up the arc will not fall on the design grid, and thus is not manufacturable, without considerable manual adjustment. All structures in an APD layout, on all layers, are either solid regions, or ring-type regions with the center removed. Both round and square regions and rings are required, depending on whether a round or square device is being constructed. In Figure 3.3, it is clear to see the central *n*+ region was a solid circle (or square), the *n-base* guard ring was a doughnut, as was the p+ contact ring. All of the remaining layers, which were not shown in Figure 3.3 for clarity, such as the metal contact rings and the various required active regions, were also similarly either solid regions, or doughnuts or rings. The pcell developed, started from a pcell found on the Cadence community forums, that was able to draw on grid circles [26]. It was modified to be able to chop a circular hole, whose vertices are also all on-grid, out of the center. A modified version was created which could generate square regions, and rings, in the same manner. While by no means does this automate the entire layout of an APD, being able to easily draw circles by specifying a diameter, and doughnuts by specifying the inner and outer diameter, makes an APD layout much simpler, quicker, less prone to error, and very easy to modify. Rather than going in layer by layer, and resizing a number of edges manually, it is much faster to simply adjust the diameter of the circle or ring at each layer. The only part of the APD layout that remains manual, is placing the contacts and vias to or between metal layers. This can be quickly achieved manually once the rest of the device is designed, but could be automated with another pcell if desired in the future. The round and square pcell code can be found in Listings A.1 and A.2 in Appendix A.

All of the special physical design requirements to control the electric field

strengths in the device, to ensure the breakdown occurs in the intended location, were discussed in this section. Because layout, or layout related issues, were the most significant problems encountered in ultimately achieving Geiger mode operation, especially considering the inability of being able to easily verify what was actually fabricated, versus what was drawn in the CAD package, the focus of the discussion was on the layout issues discovered in the previous designs. Some layout aids in the form of pcells were developed, making APD layout easier, and less prone to error. In addition, the pcells allowed the layouts to be easily modified, and thus facilitated making variants of the devices to investigate the causes of the previous absence of Geiger mode operation. It is hoped that the observations made regarding the layout issues, will eliminate similar issues in future work on this project, and perhaps eliminate issues that may be encountered by anyone implementing APDs in any similar process technology.

3.4 APD Designs

In this section, the final generation of APD designs, which correctly function in Geiger mode, will be discussed. Non-functionality in previous generations of devices was determined to be due to three possible design flaws. Because all previous designs were found to have layout issues, an entirely new set of APD layouts was created, using the pcells presented in the previous section. Rather than making many different designs in many different sizes, resulting in a large number of devices to test, and potentially much unfruitful testing being conducted if the devices failed to work again, a smaller subset of designs was implemented than had been in the past. Three basic layer configurations were chosen. Two of the configurations were implemented in two different sizes, rather than implementing all the different designs in four different sizes as had been in the past. One configuration was also implemented with a differing amount of overlap between the primary implant layer, and the guard ring. This was thought to be one of the other potential problems causing the lack of Geiger mode operation in the previous devices. Three variants were produced to test that theory. One with a 'regular' amount of overlap, based on

looking at existing design rules, in the 1 µm to 2 µm range, one with more generous overlap, in the 3 µm to 3.5 µm range, and one with no overlap at all, just the guard ring doping layer abutted to the edge of the primary implant region.

Overall there were 7 unique layer topologies, which were all implemented in both round and square versions, with the same key design dimensions in both versions, resulting in 14 total devices. The round and square versions were not directly comparable, as the area of a round versus square region of the same dimension is not the same. However, this was done to eliminate the possibility that the overall geometry of the device was the cause of non-functional devices in the past. With this smaller, but more wisely chosen subset of devices, it was theorized that, between the layout problems that were fixed, the amount of overlap between the guard ring and the primary junction region, and the general device geometry, the cause of the previous failures could be isolated. As will be shown in the results section, all of the devices in the final generation of devices, functioned in Geiger mode. This indicated that previous non-functionality was due to layout issues, and not due to the amount of overlap between the guard rings and the primary active regions, or the overall device shape.

The first design implemented was an *n+* primary active region, protected by an *n-base* guard ring, sitting directly on *p-substrate*, which was contacted with a *p+* contact ring. A plan view can be seen in Figure 3.3 in the previous section. The plan view shows how the design was mapped from the round version to the square version, but otherwise is not very illustrative. For the remainder of the devices, the plan view will be omitted, but in a similar fashion, the circular pcells used to construct the round device, were directly replaced with the square variant, keeping all key dimensions the same between the two geometries. In Figure 3.4, the cross-section view of the device is presented. This more clearly shows the interesting parts of the design, and is equally applicable to the round, and square implementations. This design was implemented in one size, where the central *n+* region is 18 µm. The dimension indicates the size of the active region not overlapped with guard rings, and not taking into account some of that area that might be covered by

contact metal above. More complete dimensions of all of the devices is given in Table C.7 in Appendix C. The round version of this design is designated as device *OP6-1*, and the square version *OP6-8*.

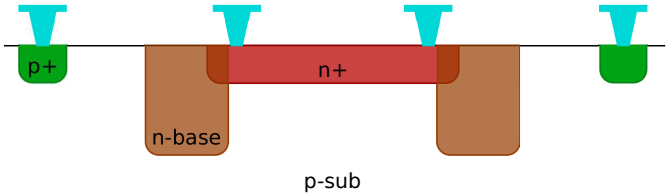


Figure 3.4: *n+ p-sub* APD cross-section. Implemented in *OP6-1* (round) & *OP6-8* (square)

The second design is essentially the opposite of the first. A *p+* primary active region is protected by a *p-base* guard ring, which is sitting in an *hvnwell* tub. The *hvnwell* is contacted with a *n+* contact ring, and the substrate around the *hvnwell* is contacted with a *p+* contact ring. The cross-section view of this design can be seen in Figure 3.5. This design was implemented in two different sizes of the central *p+* active region. The smaller 8 µm design is designated *OP6-2* and *OP6-9*, round and square respectively, and the larger 18 µm version is designated *OP6-3* and *OP6-10* round and square respectively.

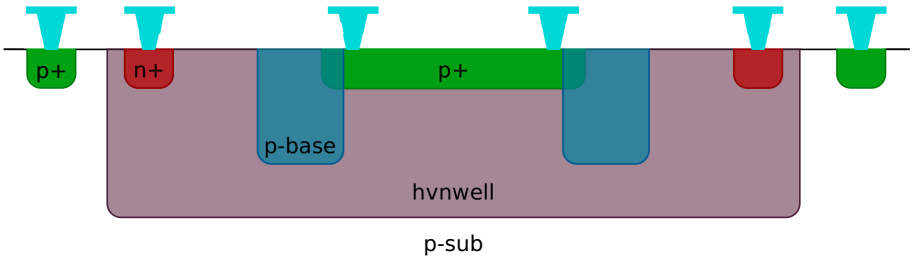


Figure 3.5: *p+ hvnwell* APD cross-section. Implemented in *OP6-2* (8 µm) / *OP6-3* (18 µm) [round] & *OP6-9* (8 µm) / *OP6-10* (18 µm) [square]

Two variants of the previous design were built to determine if the amount of overlap between the active region and the guard ring was causing the previous failures. The first variant is the abutted version, where there is zero overlap between the

p+ and *p-base* regions, as drawn in the CAD design. Some lateral diffusion might cause overlap to exist in the manufactured device, but the equipment necessary to verify the manufactured devices is not available, nor is an accurate estimation of the lateral diffusion coefficient from TDSI. The cross-sectional view of this design is illustrated in Figure 3.6. Only one size, 18 µm active region, of this device was built. In this design, if the overlap, or lack thereof, of the guard ring was the cause of the issue, it would be unlikely to be functional. This design has the designation *OP6-4* and *OP6-11*, round and square respectively.

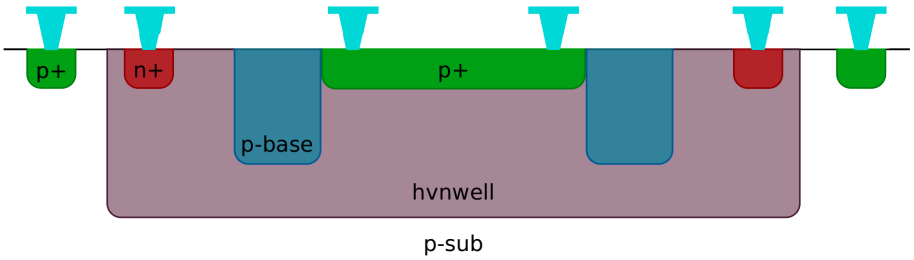


Figure 3.6: *p+ hvnwell* APD cross-section. Abutted *p+ p-base* regions. Implemented in *OP6-4* (round) & *OP6-11* (square)

The second variant has an extra-wide overlap region between the primary *p+* active area and the *p-base* guard ring. As well, the contacts to the *p* side of the device were placed in the *p+ p-base* overlap region, eliminating the need to cover any of the *p+* only region with metal or contacts. The cross-section of this design can be seen in Figure 3.7. Because this design had the highest chance of working if the overlap amount was the cause of the previous issues, two sizes were built. The smaller, 8 µm active region device is designated as *OP6-5* and *OP6-12*, and the larger 18 µm active region device as *OP6-6* and *OP6-13*, round and square respectively in both cases.

The final design is similar to the previous one, with a *p+* primary active region, but rather than *p-base* guard rings, has *p-well* guard rings, and instead of sitting in an *hvnwell* tub, sits in an *lvnwell* tub. This device has a substantially lower reverse bias breakdown voltage than the others, due to the difference of the characteristics of the *lvnwell* compared to the *hvnwell*. This also shows that *p-well*, which is typically

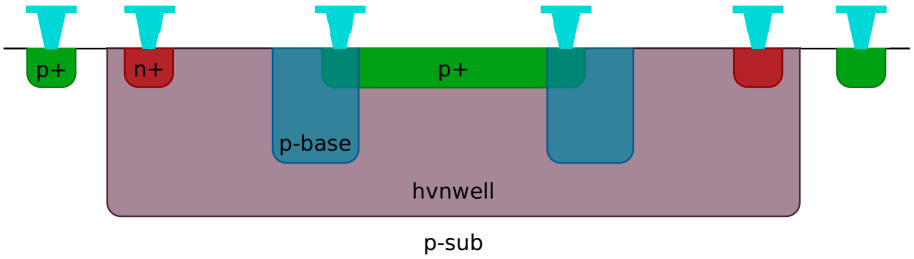


Figure 3.7: *p+ hvnwell* APD cross-section. Large overlap between *p+* and *p-base* regions. Contact for *p* region in overlap area. Implemented in *OP6-5* (8 µm) / *OP6-6* (18 µm) [round] & *OP6-12* (8 µm) / *OP6-13* (18 µm) [square]

the well used for building low voltage NMOS transistors in the TDSI CMOSP8G process, can be used effectively as an APD guard ring. Looking at the doping profiles obtained from TDSI, the *p-well* is even more lightly doped, and about 3 times deeper than *p-base*. The cross-sectional view of this design can be seen in Figure 3.8. Only one size of this device was built, with a *p+* active region that is 18 µm, and is designated *OP6-7* and *OP6-14*, round and square respectively.

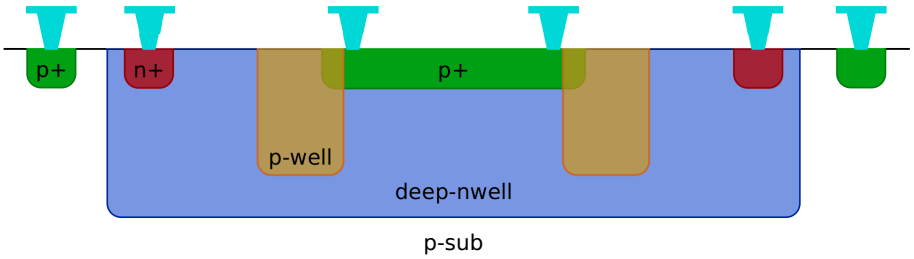


Figure 3.8: *p+ deep-nwell* APD cross-section. Implemented in *OP6-7* [round] & *OP6-14* [square]

In this section, the seven different APD designs that were implemented were presented. The motivations behind choosing each variant that was implemented, and which potential problems were isolated with each variant, were given. It was theorized that with this smaller, but more thoughtfully chosen subset of devices, the issues plaguing previous generations of devices could finally be isolated. A summary of all of the APD designs is presented in Table 3.1.

Junction	Guard Ring	Size	Comments	Round	Square
n+ - p-sub	*n-base*	18 µm		OP6-1	OP6-8
p+ - hvnwell	*p-base*	8 µm		OP6-2	OP6-9
p+ - hvnwell	*p-base*	18 µm		OP6-3	OP6-10
p+ - hvnwell	*p-base*	18 µm	Abutted GR	OP6-4	OP6-11
p+ - hvnwell	*p-base*	8 µm	3.5 µm Overlap GR	OP6-5	OP6-12
p+ - hvnwell	*p-base*	18 µm	3 µm Overlap GR	OP6-6	OP6-13
p+ - deepnwell	*p-well*	18 µm	Low-voltage region	OP6-7	OP6-14

Table 3.1: Summary of SPAPD designs

3.5 Quench circuits

There are two methods used for controlling the avalanche current, and preventing the device from damage due to heating: passive and active quenching. Both methods of quenching must stop the avalanche current, and prepare the APD for the next detection event.

3.5.1 Passive Quench

In passive quenching, a resistor is placed in series with the APD device. A schematic showing passive quench configuration is given in Figure 3.9.

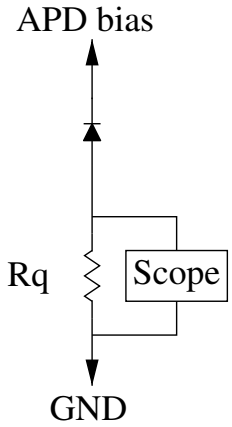


Figure 3.9: Passive quench configuration.

Before an avalanche event happens, no appreciable current is flowing, so the voltage across the resistor is 0 V, and all of the reverse bias voltage appears across

the APD. Once an event is started, the current quickly builds up to a level sufficient to lower the voltage across the APD to exactly V_{br}, and the current fluctuates around the mean value to maintain V_{br} across the device. As the voltage approaches V_{br}, the intensity of $I_d(t)$ falls, and the number of carriers crossing the junction decreases [27]. Because the avalanche effect is statistical, once the number of carriers crossing the junction is reduced, the probability that no carrier impact ionizes any further carriers increases, and the current quenches. The time required for this to happen is called the self-quenching time, and is dependent on the junction capacitance and internal resistance of the APD, as well as the external quench resistor value. The voltage across the quench resistor can be monitored, for instance by a fast comparator, and when the voltage is above a threshold, an event is detected, and counted with a digital counter. This is why an APD operated above V_{br} is considered a 'single photon' counting device - a current pulse is generated from one photon being absorbed in the depletion region of the device. Other names for this operating mode of an APD include Single Photon APD (SPAPD), and Geiger mode APD (GAPD), because the behaviour resembles a Geiger tube emitting a pulse when ionizing radiation is detected, or less commonly some combination of those terms.

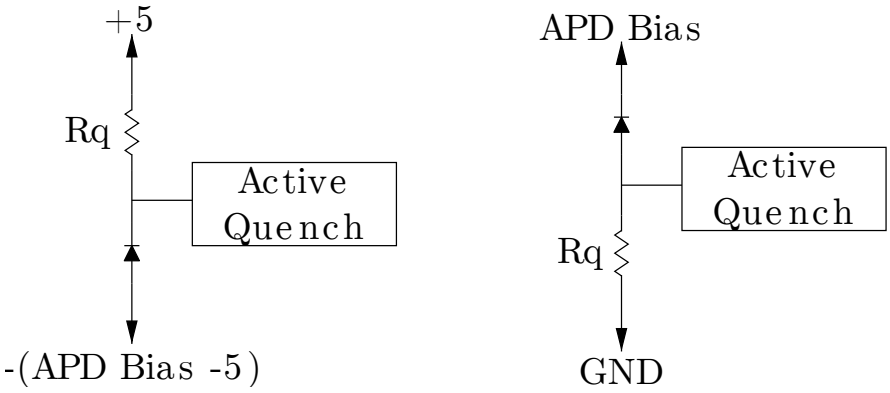
Passive quenching is very easy to implement, in that only a resistor is required, but it does have some drawbacks. The main drawback is *dead-time*, or the time after an event has started, and before the device is ready to detect the next event. Because the quenching process is statistical, and dependent on the device properties as well as the quench resistor value, the time for the device to self quench is variable, and can be rather slow. While the current is still flowing before the device has quenched, subsequent photons which are absorbed in the depletion region, can not be detected. Only once the current has completely quenched, can more photons be detected. The solution to this issue is to use active quenching.

3.5.2 Active Quench

The goal of active quenching is to use active elements in the quench circuit to speed up the detection, quenching, and reset phases of operation of the APD, and thus reduce the dead time [28]. Typically an active quench circuit still uses a quench resistor, but rather than waiting for the current to self-quench, as in passive quenching, the voltage change across the quench resistor is detected, and the voltage across the APD is forced below the breakdown threshold to quench the current. This is called active quench. The APD is held in the quenched state, with a bias below the breakdown threshold, for a fixed amount of time, called the hold-off time, to allow any trapped carriers to escape, reducing the likelihood of after-pulsing resulting from the carriers escaping once the device is biased above the breakdown threshold again. After the device remains in the quenched condition for the desired interval, the bias voltage is forced above the threshold again, resetting the device for the next event. This phase of operation is called active reset. Active quenching also has another advantage. Because the avalanche event is detected and quenched sooner than with passive quenching, less current flows for a shorter duration, lessening the chance of trapped carriers [29].

An active quench circuit, based on the active quench circuit in [30] was designed, and tested in simulation. The quench circuit can be seen in Figure 3.11. The quench circuit was designed for the APD and quench resistor topology shown in Figure 3.10a.

As current begins to flow in the APD as a result of an avalanche breakdown, the voltage at the cathode of the APD begins to fall. The active quench circuit detects this, and forces that node to 0 V, thus lowering the bias across the APD by 5 V, and quenching the avalanche event. The device remains actively quenched for a fixed amount of time, set by a bias voltage on a monostable circuit, shown in Figure 3.12. After the quench hold time is reached, the active quench transistor is turned off, the active reset transistor is turned on, and the voltage at the cathode of the APD is raised back up to 5 V, resetting the device for the next avalanche event. The device is held in the reset stage for a fixed amount of time, set by the bias voltage



(a) Current APD and quench resistor arrangement

(b) Preferable alternate APD and quench resistor arrangement

Figure 3.10: The current APD quench arrangement, undesirable as it requires a negative APD bias, and an alternate arrangement more suitable for integration with the rest of the system, as only a positive APD bias is required.

to a second monostable. The active quench circuit was designed to have the delays set by external bias voltages, so that the delays can be adjusted to achieve optimal operation.

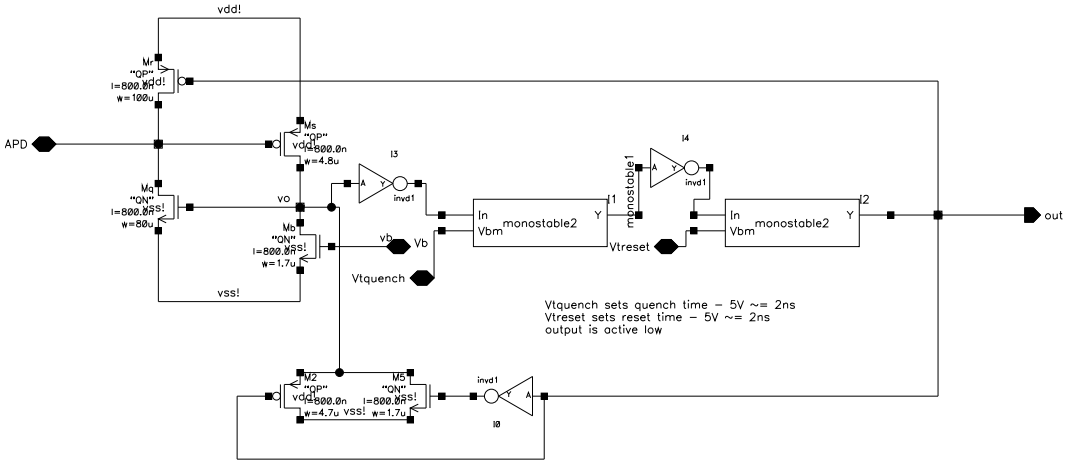


Figure 3.11: Schematic of active quench circuit. Length of quench and reset phases controlled by bias voltages Vtreset and Vtquench.

A simulation of the active quench circuit in operation is shown in Figure 3.13. The APD has been replaced by a pulse generator with a 'slow' rise and fall time of 2 ns, connected to the APD node of the active quench circuit through a 100 kΩ

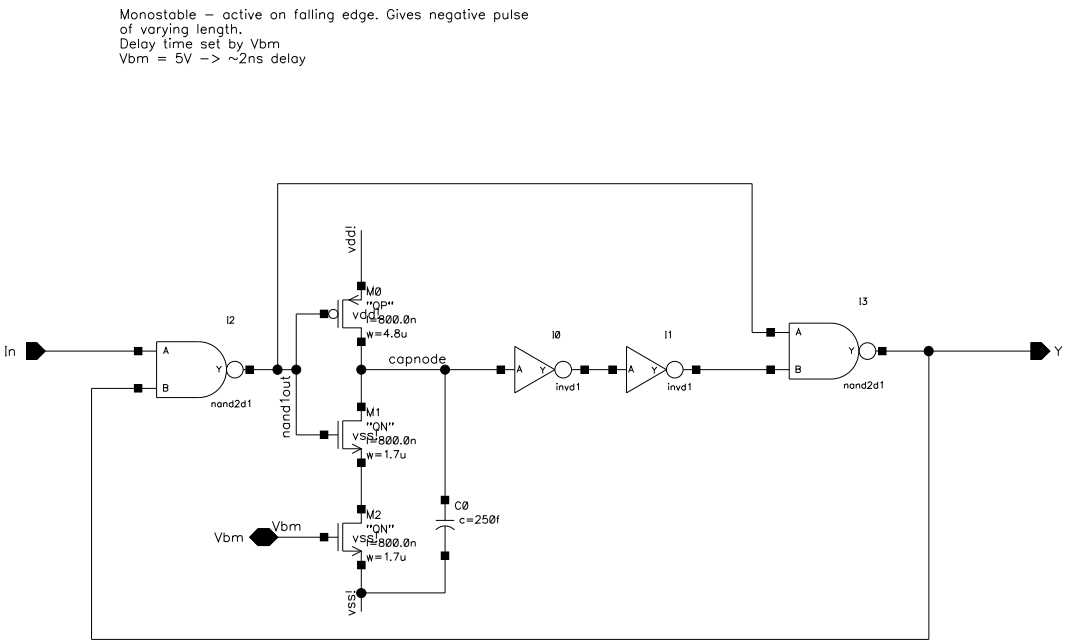


Figure 3.12: Schematic of monostable used for timing in the quench circuit.

resistor. As the voltage at the APD node begins to fall in region *a*, simulating an avalanche event, the quench circuit detects this, and actively forces the APD node to 0 V in region *b*. The device remains quenched for roughly 5 ns, at which point the active reset transistor is turned on, and the voltage at the APD node is returned to 5 V, returning the bias across the APD above the reverse bias threshold, and readying the device for the next avalanche event. With the biasing chosen in the test bench, the device remains actively reset for roughly 20 ns, as shown in region *c*. The reset transistor is then shut off, and the device is ready for the next avalanche event, which occurs at the end of region *d*.

Due to the unexpectedly large amount of time required to characterize all of the APDs in Geiger mode, and measure the non-functional APDs in photoconductive mode, the active quench circuit has not been characterized in silicon yet. In addition, the topology of quench resistor and APD that the quench circuit was designed for, does not lead to an easy integration with the rest of the system, due to the requirement of a negative APD bias voltage. The quench circuit should be adjusted for the topology shown in Figure 3.10b before being implemented again. This ar-

rangement does not require a negative APD bias voltage, and a high positive voltage is relatively easy to obtain in the system, by tapping various stages of the capacitive charge pump generating the high voltage for injection and separation during CE. The active quench circuit would need to be adjusted to detect the rising voltage at the APD anode during an avalanche event, quench it by raising the anode voltage to 5 V, and reset by forcing that node back to 0 V.

It is important that regardless of how the quench circuit is implemented, the quenching and resetting of the APD is done using voltages the low voltage transistors in the CMOSP8G process are capable of switching. While high voltage transistors are available, and it could be possible to lower the cathode voltage rather than increasing the anode voltage, the high voltage transistors are very slow compared to the low voltage transistors. Using high voltage transistors in an active quench circuit would negate the entire benefit of active quenching, faster device operation, and reduced dead time. Another important factor that should be considered for any future quench circuit development, is to simulate the quench circuit in all process corners. It is especially important to simulate at elevated temperatures, which are expected due to PCR thermal cycling.

3.6 Test Apparatus and Calibration

Because of the sensitivity of APDs, especially when operated in single photon counting Geiger mode, and the very small currents involved in characterizing any photo detectors in photoconductive mode, special care must be given to the test apparatus and the calibration of the testing environment to ensure accurate, meaningful results are being collected. In this section, the special attention that was given to the testing environment, as well as the calibration methods and results are presented.

While testing the first versions of the APD designs, a number of different test arrangements were used, all with several disadvantages. Many of the parameters between tests, and even between various runs of the same test, could not be kept consistent due to inadequacies of the test setup. Different light sources were used.

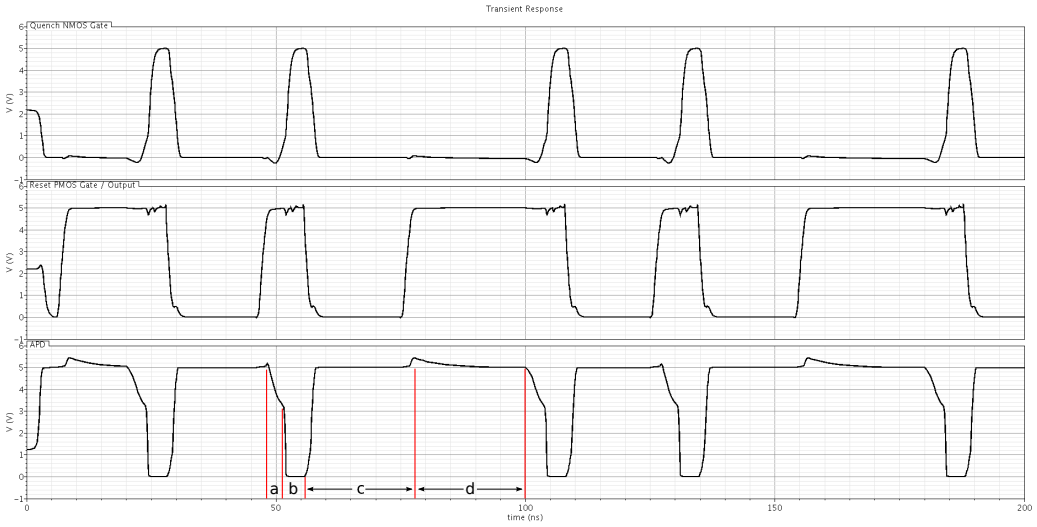


Figure 3.13: Simulation of active quench circuit operation. Geiger pulse starts and is detected in period *a*, quench transistor turned on during *b* forcing APD bias below reverse breakdown, device held above reverse bias breakdown in reset during *c*, and ready for next pulse in *d*.

Some devices were packaged in DIP packages, while some were bare dies, which affected the distance between the light source and the device under test (DUT). The distance between the optical power meter and light source could not be properly measured, and therefore accurate calibration of the light source was unachievable. The probing infrastructure available for bare dies was quite primitive and lacked any shielding, either optical or EM. In addition, there was no automated way to sweep the intensity of the light source to test the optical response of the devices. Initial attempted methods of obtaining a dark test environment, included making the whole room dark by sealing up windows and around doors to block external light, and using a 'dark enclosure' made out of black fabric previously used for other optical experiments on the LoC project. In the case of the dark room, even after sealing up the door and window to block as much external light as possible, it was impossible to block all internal light sources in the room. All test equipment has illuminated indicators or displays which can not be fully turned off, and are difficult to completely block, while still leaving the equipment functional. After spending enough time inside the second fabric 'dark enclosure', it was noticed that

even with 2 layers of black cloth, the room lights from outside the enclosure were still visible through the fabric, and again, the test equipment and operator were also inside the dark area. As the test operator, spending hours inside a completely dark area is less than ideal. A home made 'dark box', made out of ABS plastic sheet, was also used to enclose the probe station. The box had many holes in the ABS from previous items being mounted to the box, and the door also leaked a lot of light. The biggest problem was the box was mechanically unstable, and opening or closing the cover disturbed the probes from the probe pads on the chip. The probes themselves also had some issues. The probes had rather long, unshielded leads. Both of the spaces these initial test environments were in became unavailable, and the whole apparatus had to be moved twice. In hindsight though, this was actually a fortuitous event for the test environment.

The second time the test environment was moved, a more permanent location was chosen, and a proper dark enclosure was purchased for the probe station. Frustratingly, even a commercial 'dark enclosure', specifically designed for a wafer probe station, was found to have significant openings where light could leak in. Modifications were made to seal up all potential sources of outside light leaking into the enclosure. Once these modifications were complete, the enclosure provided a fully light tight, EMI shielded environment, for testing the extremely sensitive Geiger mode APDs. The box was also mechanically stable enough to be able to open and close the lid without probes moving from the probe pads, and contained only the probe station itself in the dark enclosure, eliminating the test equipment, and the operator, from the dark environment. Along with the enclosure for the probe station, the probes themselves were upgraded to more modern probes, with shielded cables and BNC connectors, so that the pass-throughs in the enclosure could be used. The new probes also provided a more reliable, and better shielded connection to the test equipment.

In order to obtain consistent results so that all devices could be properly compared, a consistent, automated test environment was required for characterizing all of the devices that were designed and fabricated. All devices were tested as bare

dies in the probe station. The light source was applied through the illumination port of the microscope on the probe station. This eliminated the uncertainty of the distance between the light source and the DUT, and hence the optical intensity of the light source on the DUT. The microscope in the probe station was optically focused on the surface of the DUT through the eyepieces, ensuring consistent spacing between the objective lens and the DUT, and thus also providing consistent optical intensity. Regardless of which device is used, either fabricated bare die, optical power meter sensor, or other device required as the DUT, and regardless of the physical size of that device, because the surface is in focus, the distance remains constant.

The light source chosen was a red LED. Red LEDs have a reasonably narrow wavelength profile, and do not jump between various modes of emission, as was found to be the case with laser sources that were previously used for other experiments in the LoC project. Red was chosen due to the expected emissions of the fluorophores. A common fluorophore used in CE is Cy5, which has an optimal excitation wavelength of 649 nm, and an emission wavelength of 670 nm [31]. Initially, a regular 3 mm T-1 diffused red LED was mounted coaxially with the existing white LED illumination source on the probe station. Because of the diffused case, as well as very small output power, the maximum amount of light available at the DUT, after passing through the optics of the microscope, was extremely small, on the order of $20\,\mathrm{nW\,cm^{-2}}$. Because a light source with a greater dynamic range was desirable for testing the manufactured devices, a much brighter LED source was required. An approximately 10 W RGB LED, mounted to a large heat sink was used to replace the existing white LED illumination source on the probe station. Using just the red channel of the 10 W LED means roughly 3 W (electrical input) of red light is available, which produces more than enough light through the optics of the microscope to the DUT, resulting in a sufficient available dynamic range of excitation light for testing the fabricated devices. The light at the DUT position at maximum current is roughly $38\,\mu\mathrm{W\,cm^{-2}}$. The LED is driven by a programmable power supply, and programmable current mode DAC, that was available on a test

board built by David Sloan, for testing other LoC systems. With the programmable power supply and programmable current mode DAC, the current flowing through the LED can be swept from 0 to 225 mA in 65536 steps, or with a resolution of roughly 3.4 µA, under software control, and by doing so, the optical power incident on the DUT is also swept accordingly.

The optical spectrum of the light from the red LED at the DUT position was measured using an Ocean Optics USB2000 spectrometer. From the spectral plot of the LED light source in Figure 3.14, it can be seen that the LED produces a narrow wavelength range, in an approximately 54 nm wide region centered around 623 nm, with no significant spurious emission that is not around the main peak. While only slightly covering the 670 nm wavelength that is expected from the Cy5 emission, 623 nm of the LED provides a reasonable approximation of that wavelength. LEDs that emit in deeper red, as the fluorophore does, are harder to obtain, especially with relatively high output power. The peak wavelength of 623 nm is used to set the wavelength being measured in the optical power meter, which is used for the remainder of the calibration, as well as anywhere else a wavelength value of the light source is required in any calculations.

The illumination at the DUT position was characterized by using a Newport 1930C optical power meter, with a Newport 918-UV optical power measurement head, in place of the DUT. The LED current was swept, and the resulting optical power measured. The illumination spot size produced by the lowest power objective available on the probe station microscope, is slightly larger than the area of the optical power meter sensor. Therefore, the optical power measured is over the area of the detector, which is 1 cm^2. The current flowing through the LED was measured using an Agilent U1252A DMM with remote USB interface. As would be expected, the light at the DUT position is linearly related to the current flowing through the LED, as seen in Figure 3.15. The linear regression of that relationship, as expressed in Equation 3.1, can be used to map between the measured LED current and the light output. When measuring other devices for optical characterization, the LED current is always measured, and Equation 3.1, along with the area of the device

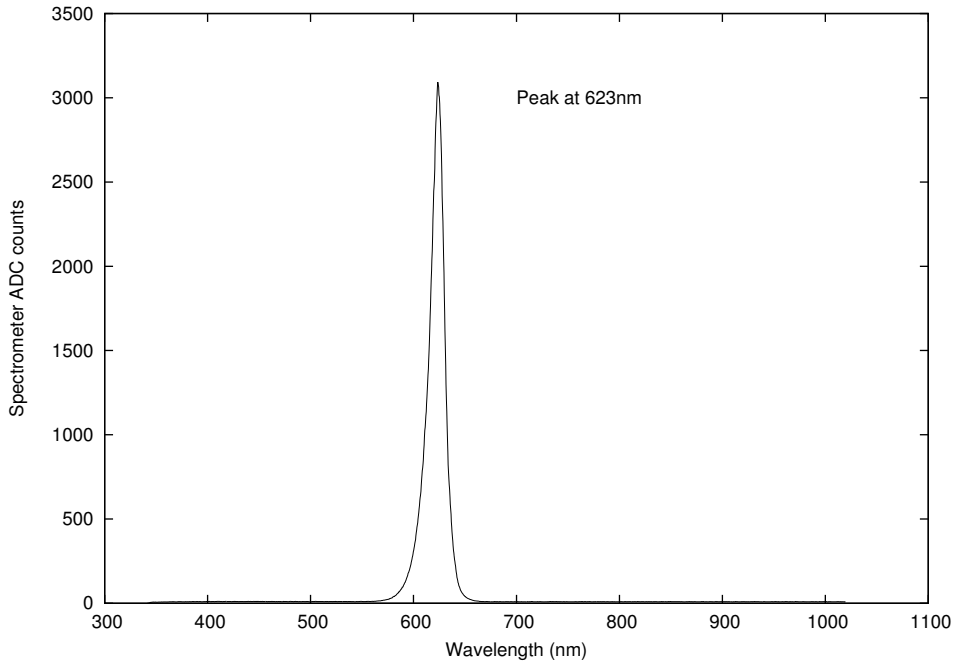


Figure 3.14: Optical spectral output of red LED light source. Peak wavelength 623 nm

being measured, is used to determine how much light is incident on the DUT.

$$P_{OI} = 1.733\,41\times10^{-4}\,\mathrm{W\,cm^{-2}\,A^{-1}} \times I_{led}[\mathrm{A}] + 1.110\,62\times10^{-7}\,\mathrm{W\,cm^{-2}} \qquad (3.1)$$

If the illumination was consistent across the entire field, the value reported by the power meter could simply be divided by the area of the detector to give the optical intensity anywhere in the light field. Unfortunately, the light intensity was measured in several spots horizontally across the center of the light field, as indicated in Figure 3.16a, and the intensity was not consistent across the field. In Figure 3.16b, the light measured at each point across the field is compared with the average of all the measurements. In the region that is within the FOV of the microscope, where all of the devices were measured, because it was necessary to be able to see them to place the probes on the probe pads to connect to the device, the light intensity tends to be greater than 5% above the average of all the readings across the entire field, but varies greatly with position. This variation leads to error

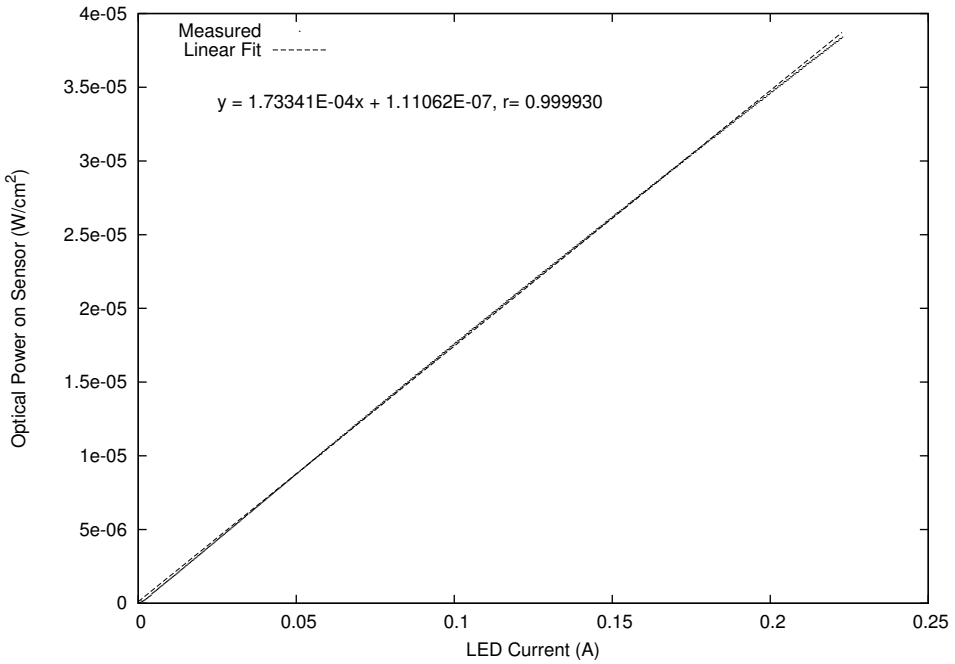
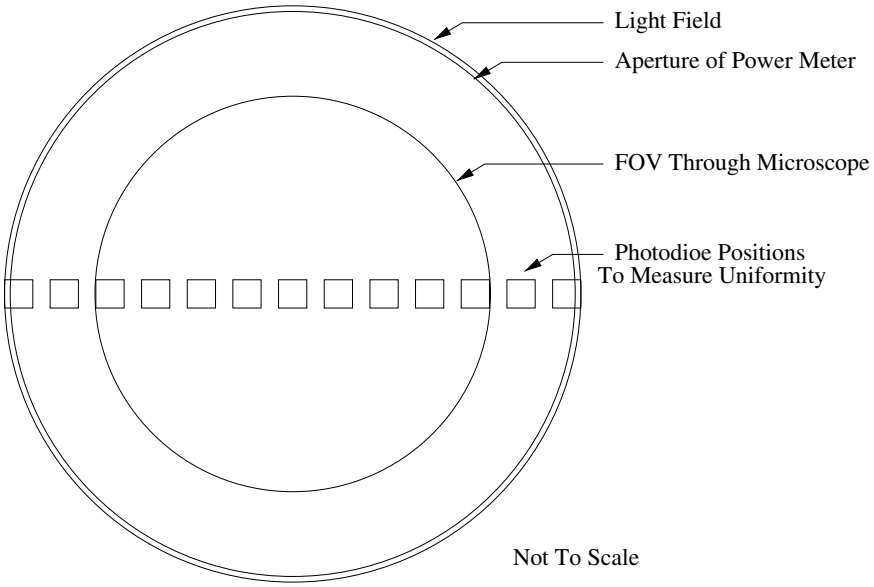


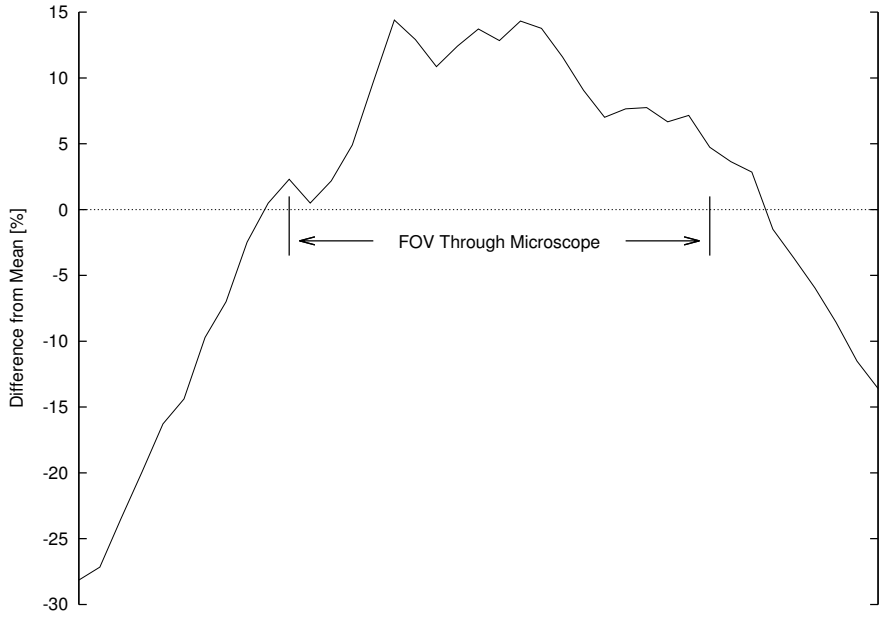
Figure 3.15: Optical output calibration of measurement light source.

when determining how much light is actually striking the device during any given test, as not only is the light intensity dependent on the current flowing through the LED, it's also dependent on the physical location of the DUT in the light field. Most devices tested were in different locations in the field, as it was easier to move the probes from device to device, rather than reposition the entire chip when moving from device to device. Despite best efforts to assemble a repeatable and uniform test environment, this is a major flaw which limits the accuracy, and comparability of all of the optical measurements.

Dark, reverse bias I-V curves were measured for all devices. The primary information obtained from the dark I-V curve, is the reverse bias threshold for each device, which was necessary for knowing what reverse bias voltage to use when testing for Geiger mode operation. For performing dark I-V curves, a Keithly 4200-SCS semiconductor characterization system was used. The available 4200-SCS contains 4 SMUs, and thus no other additional test equipment was required to perform the dark I-V curve testing. A sweep voltage range was set, and the current



(a) Light field uniformity setup



(b) Light field uniformity horizontally across center of field

flowing through the device was measured by the SMU. A current compliance value of not more than 100 µA was used for all tests on the 4200-SCS, to ensure the devices would not be damaged by excessive current, or be affected by heating, when the voltage sweep was above the reverse bias breakdown voltage.

Any test or calibration that required multiple instruments to be coordinated, or programmed, was done with custom written Python scripts running on a Linux computer. The equipment was connected to the test computer by RS-232 serial, USB, or GPIB depending on what was available on the equipment. The GPIB bus was interfaced to the PC using a Prologix GPIB to USB interface, which appears as a virtual serial port to the operating system. This allows a GPIB connection without installing the typical NI GPIB libraries, which, despite many attempts in different Linux varieties, could not be made to work correctly. Having the equipment entirely controllable through scripts, greatly increases the flexibility of the entire test system, and allows fairly complicated, or very long-running tests to be carried out relatively easily.

All devices were tested in photovoltaic mode, by sweeping the intensity of the LED light source, and the resulting photocurrent measured with a Keithly 6487 picoammeter. Internal averaging in the picoammeter was used to obtain more reliable results. Especially at very low current levels, individual readings can be quite noisy, and not an accurate measurement of the current flowing out of the device. All of the devices that were fabricated, could be used as normal photo diodes, even if they did not function in Geiger mode. This test allows all of the devices responsivities to be compared.

For the devices which do operate in Geiger mode, the devices were tested in a passive quench arrangement. An Agilent 54845A 1.5 GHz 8GSa/sec oscilloscope was used as an ADC and data acquisition device to capture the fast Geiger pulses. The scope was set up in free-run (no trigger) mode, with a time base of 20 ms, and 10 individual captures were performed at each measurement. While the time base was set to 20 ms, the scope actually captured a slightly larger period of time, culminating in the unusual 26.21 ms time period shown in all of the results. The raw

trace data was transferred to the PC, and the Geiger pulses detected and counted by software. A software pulse detector allowed for more flexibility during device characterization. Ultimately, the pulse detection should be implemented with a high-pass filter, and a pulse lengthening circuit integrated with the active quench circuit, producing an output suitable to drive a digital counter. The average number of pulses in the 10 captures was reported, and gave an average pulse count per 26.21 ms window. In addition, the raw trace data from every capture of the oscilloscope was saved, allowing further processing, or verification with the raw data, should any further analysis have been required, without having to physically run all of the tests again. Programmable Agilent E3647A power supplies were used for any bias voltages that needed to be controlled during testing. The actual bias voltage across the APD was measured with an HP 34401A bench DMM, as it was found the Agilent E3647A power supplies were not as accurate as a DVM, and did not produce exactly the voltage they were set to. The test script outputted all of the raw data for each measurement, along with other key data, such as the quench resistor value used for the test, so that the output data was complete. With numerous test data from numerous different devices, and testing under different conditions, this eliminated any chance of later confusion of the results. A sample of the data produced by the test script is shown in Table C.14.

A few different approaches were tried for the software peak detection, including implementing a software Schmidt trigger. With fixed thresholds for the Schmidt trigger, the detector was extremely hard to tune, as the overall nature of the signal changed as the light level was increased, and the thresholds would have had to have been dynamically adjusted. Rather than implementing dynamic thresholds, a much simpler approach was used. The 3 main characteristics of a Geiger pulse were used to build a more suitable detector. From looking at the raw data captured by the oscilloscope, a Geiger pulse always had a relatively fast rise time, with a minimum amplitude, then a slower recovery time as the current quenches. The software pulse detection filter looked for the biggest voltage delta between the current point, and a small window of previous points. If that delta was greater than the threshold, a peak

was detected, and the detection moved ahead by the number of points in the forward window. The minimum delta ensured that random noise is not detected as peaks, the small backward window size ensures the pulse was fast enough to be a Geiger pulse. The backward window was set to 6 data points, or 2.4 µs, the forward window, or hold-off, was set to 19 data points, or 7.6 µs, and the voltage threshold was set to 3 mV. The actual peak detection code can be see in the *peakdetect* function in Listing B.3, in Appendix B. In the future an analog circuit will be required to perform this detection, independent of baseline. It will probably have to look for positive spikes in the first derivative, using a high pass filter. A small sample of oscilloscope data, showing the peaks, as detected by the software peak detector, is presented in Figure 3.17.

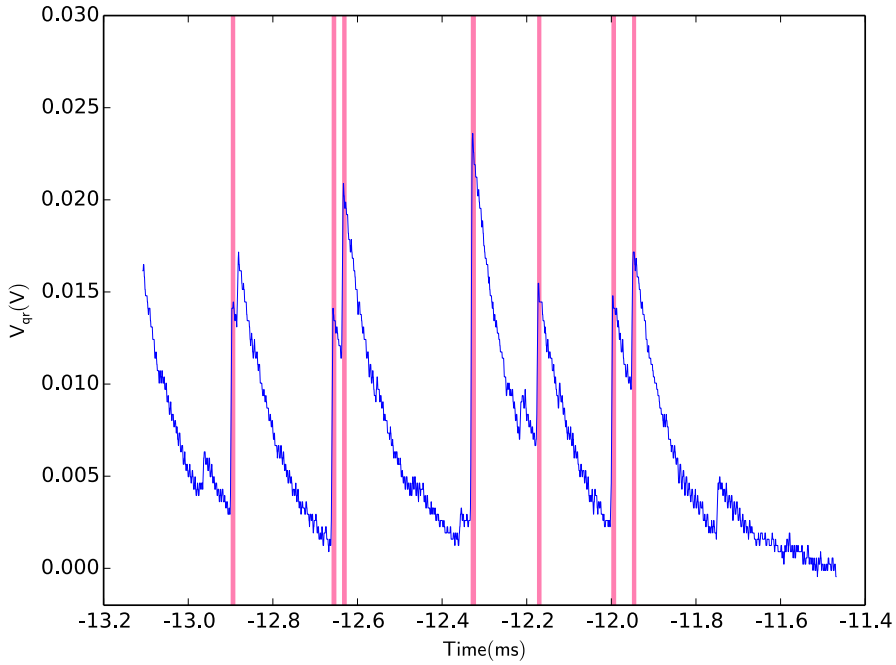


Figure 3.17: Raw oscilloscope data showing voltage across quench resistor, Geiger pulses, and software pulse detection (shaded regions).

In this section, a great deal of time and effort went into the test apparatus, to ensure consistent results were presented. Testing devices that were so sensitive, or

in some cases produced such small currents, or such fast pulses, demanded a very advanced test setup to acquire the most accurate results possible. Almost every aspect of the test apparatus was significantly improved, either through researching and acquiring new equipment, or upgrading, or modifying existing equipment that was unsuitable.

3.7 Test Results of Fabricated APDs

In this chapter, because of the very large number of devices fabricated during this work, interesting or key results will be presented directly, and the remainder of the results will be summarized in tables in Appendix C. All devices will be referred to in *[chip name]-[device number]* form. Tables C.1 though C.7 have a more complete description of each device, the doping layers used, and the important device sizes. To complete such a myriad of tests, almost all operations from data collection and analysis, to graph production were automated. All of the scripts written to automate the data collection and analysis are presented in Appendix B. For the early generation devices which did not correctly function in Geiger mode, dark I-V curves were measured to find the reverse breakdown voltage, to know what voltage to operate the devices at to look for Geiger operation. All of the devices were characterized in photovoltaic mode, measuring incident light versus current out, or responsivity. This allows all of the devices to be compared, for use as regular photovoltaic photo diodes, even though they did not correctly function in Geiger mode. For the later generation devices that did function in Geiger mode, in addition to these tests, all devices were tested with a passive quench arrangement, with a swept intensity light source. Results, in the form of Geiger counts per time period, versus light intensity, will be presented. This is the cardinal result showing correct Geiger operation. Some additional tests were carried out on a selected device, to show the effects of increasing the bias voltage beyond V_{br}.

3.7.1 Dark I-V and Photovoltaic Mode

The first test performed on all of the devices was a reverse bias dark I-V curve. The objective of this test is to measure each device's reverse breakdown voltage, to know roughly what bias voltage should be applied to investigate Geiger operation in further tests. The reverse breakdown voltage is dependent on what doping layers are used to construct the primary junction of the device. As was mentioned previously, a current compliance was always used when running I-V curves to prevent damage to the device from excess current, or device heating, once the reverse bias threshold was reached. In addition to the reverse bias threshold, the dark current at any reverse bias voltage between 0 V, photovoltaic mode, and reverse bias breakdown, that is the entire photoconductive region of operation, can be determined from the data collected. This is not an important result for this work, but should it be desirable to operate any devices in photoconductive mode in the future, this data would be useful to compare the devices built.

Most of the I-V curves are what would normally be expected from measuring the reverse bias I-V curve of a *p-n* junction. An example of one of the curves from a device that does not operate in Geiger mode, is shown in Figure 3.18. This curve is from OP2-14, a *p+ - hvnwell* diode with incorrectly formed *p-base* guard ring, leaving it essentially just a *p+ - hvnwell* diode. From the I-V curve, which shows the breakdown voltage around 18V, it was easy to tell something completely unintended was happening. The breakdown voltage of *P+ - hvnwell* should be much higher, in excess of 40 V, indicating an unintended junction is breaking down instead, and the presence of a layout issue with the device.

An interesting phenomenon was seen in the I-V curves of *some* of the devices which do actually function in Geiger mode. Rather than a clean transition from the cutoff region to conduction, a very noisy curve is observed instead. This was likely caused by the device experiencing avalanche breakdowns, the compliance current being reached, and the 4200-SCS lowering the bias voltage in response to maintain the current compliance, in effect acting as an active quenching mechanism. One such I-V curve from OP6-3 is shown in Figure 3.19. This behaviour is not present

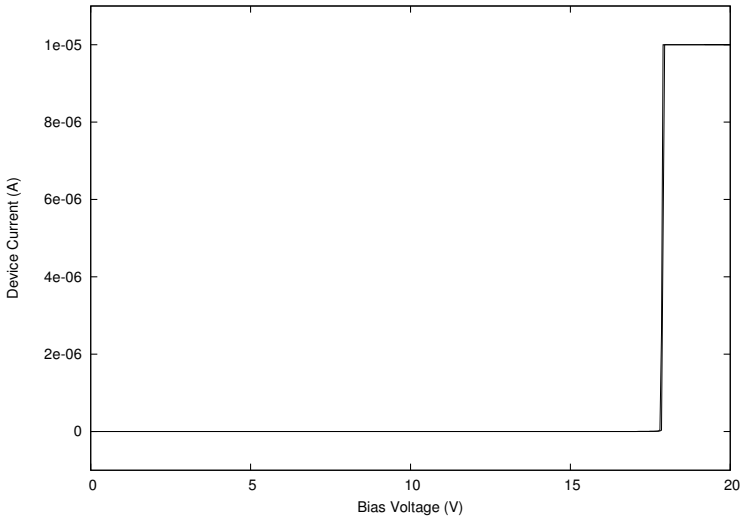


Figure 3.18: Dark I-V curve of OP2-14, showing unexpectedly low breakdown voltage, but overall expected characteristic of a reverse biased *p-n* junction.

in all devices which function in Geiger mode, so while a noisy I-V curve was likely a good indicator of Geiger mode function, the lack of the characteristically noisy I-V curve does not mean the device will not be functional in Geiger mode. The I-V curve of OP6-7, which as will be shown later, was one of the best performing Geiger mode devices, showed no such characteristics in it's I-V curve, which is shown in Figure 3.20.

All of the devices built, were characterized in photovoltaic mode. The photo diode being tested was connected to a Keithly 6487 picoammeter, the LED light source intensity was swept, and the LED current was measured. The resulting photocurrent was measured, giving a incident light versus photocurrent response curve. Using Equation 3.1 above, and the device active area, the LED current measurement values are translated into incident optical power figures. The active area used for this measurement is the total active area, including guard rings. An example of the response curve as collected from OP6-2 is shown in Figure 3.21. The slope of this response curve has the units $\mathrm{A\,W^{-1}}$, which is the photo diode's responsivity.

The result of this responsivity calculation for all OP6 devices is presented here in Table 3.2, and for all of the other devices tested in Tables C.8 through C.13

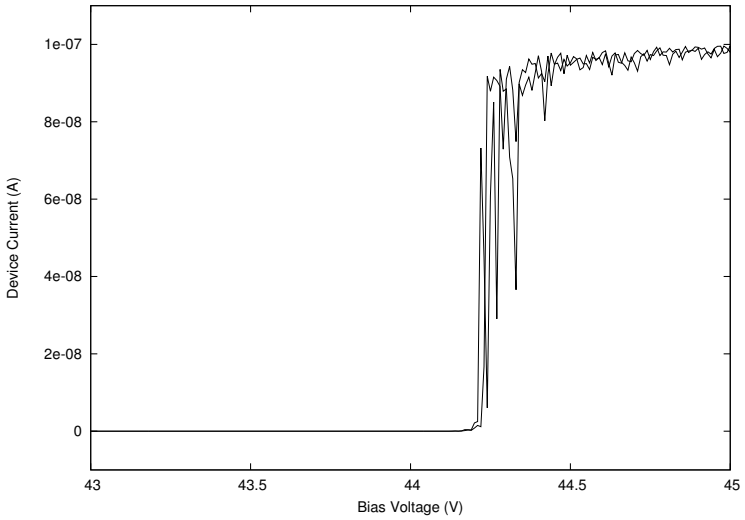


Figure 3.19: Dark I-V curve of OP6-3, showing the 'noisy' characteristic of the curve, a good indicator of Geiger operation.

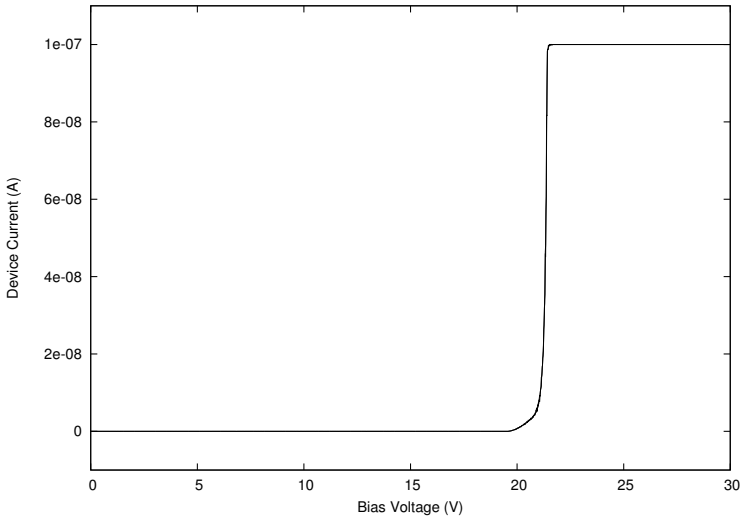


Figure 3.20: Dark I-V curve of OP6-7, which operates in Geiger mode, but does not show the 'noisy' characteristic seen in other APD devices.

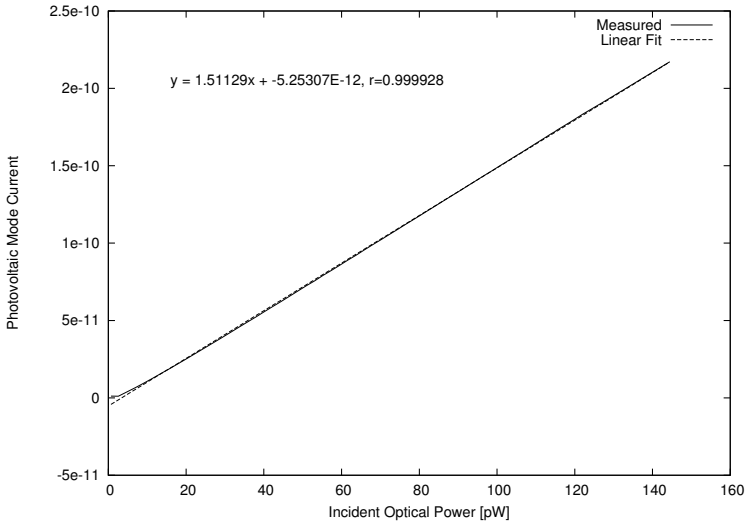


Figure 3.21: Photovoltaic response of OP6-2. Slope of the curve gives the device's responsivity [$\mathrm{A\,W^{-1}}$]

in Appendix C. In addition to the responsivity, the correlation coefficient of the linearization of the response curves is presented, showing that the response curves are indeed extremely linear.

From even a cursory examination of the results, there was clearly a significant problem in the test or measurement environment. The photocurrent generated in a photo diode is given by,

$$I_{ph} = P_{PD}\left(\frac{q\lambda}{hc}\right) \tag{3.2}$$

where I_{ph} is the photocurrent, P_{PD} is the optical power incident on the photodiode, q is the elementary charge constant, λ is the wavelength of the incident light, h is Plank's constant, and c is the speed of light.

Because the responsivity is the ratio of the photo current to optical power input, it follows that:

$$R_{ideal} = \frac{I_{ph}}{P_{PD}} = \frac{q\lambda}{hc} \tag{3.3}$$

For $\lambda = 632\,\mathrm{nm}$, the maximum possible $R_{ideal} = 0.502\,\mathrm{A\,W^{-1}}$. Typically, the actual responsivity is even lower, as this ideal maximum value does not take into

Device	Responsivity ($\mathrm{A\,W^{-1}}$)	r (correlation coeff.)
OP6-1 (round)	9.740	0.9997
OP6-8 (square)	9.002	0.9998
OP6-2 (round)	1.511	0.9999
OP6-9 (square)	1.501	0.9999
OP6-3 (round)	1.099	0.9999
OP6-10 (square)	1.099	0.9999
OP6-4 (round)	1.078	0.9999
OP6-11 (square)	1.074	0.9999
OP6-5 (round)	1.396	0.9999
OP6-12 (square)	1.396	0.9999
OP6-6 (round)	1.114	0.9999
OP6-13 (square)	1.122	0.9999
OP6-7 (round)	3.303	0.9998
OP6-14 (square)	3.272	0.9998

Table 3.2: Measured OP6 photo diode responsivities in photovoltaic mode

account reflective losses at the air-semiconductor boundary. Yet in Table 3.2, responsivity values as high as $9.740\,\mathrm{A\,W^{-1}}$ are observed, and in the Tables in Appendix C, responsivities as high as $1103.148\,\mathrm{A\,W^{-1}}$ are reported. Further test runs were performed on a selection of devices, and again the results were consistent with the results obtained on the first run, which are presented here. Also, looking at Table 3.2, there was good correlation of the responsivities obtained for the round and square versions of the same designs, OP6-1 and OP6-8, OP6-2 and OP6-9, etc. Because the results are consistent between multiple runs, and between devices of the same design but different geometry, there is some value in comparing the results against each other, but not in the absolute value. Because the responsivity measurements are extremely high, either the current value measured is erroneously higher than the actual current flowing in the device, the optical power measurement erroneously lower than the amount of light actually incident on the device, or some combination thereof. With measurements in the range of 100 pA, and no redundant instruments to verify readings, measurement error may be responsible for the unexpected responsivities. Another possible cause is the non-uniformity of the light field in the probe station, as discussed in Section 3.6, though from the measurement of

Device	Responsivity [A/W]
TC7-1	1103.148
PN1-7	83.559
PN1-1	82.022
TC7-2	69.967
OP4-8	56.762
OP4-6	55.895
OP4-25	47.817
TC7-4	30.705
PN1-10b	25.760
PN1-15	22.134

Table 3.3: Top 10 devices with the highest responsivity in photovoltaic mode

the error presented in that section, that effect is not large enough to produce results that do not agree with the theoretical maximum by several orders of magnitude. The 10 highest, as-measured, responsivity devices are presented in Table 3.3.

It is interesting to note that none of these highest responsivity devices includes the photo diodes used currently in the LoC system.

3.7.2 Geiger Mode Characterization

In this section, the results of the 7 APD designs presented previously, are shown and discussed.

All of the devices that functioned in Geiger mode were characterized using the test apparatus mentioned previously. All devices were characterized in a passive quench arrangement, with a quench resistor value of 200 kΩ. The device was manually biased above its reverse bias threshold, until Geiger pulses were observed on the oscilloscope, and ensuring the response at maximum incident light was not clipping on the scope. The test script was unable to automatically detect clipping of the waveform. Then the test script shown in Listing B.3 in Appendix B was used to sweep the light source, from zero to the maximum, and back down to zero again, and count the Geiger pulses produced during a fixed time window. The data has been corrected for the amount of light actually hitting the device, using Equation 3.1, and the active area of each device not covered by metal, which can be found

in Table C.7 in Appendix C. This gives a result of Geiger pulses per time period, versus incident light.

In all of the graphs in this section, there is a slight irregularity towards the low-light side of the graph. The current mode DAC, used to control the LED current, has a low current threshold, below which it was unable to limit the current further. To obtain currents less than that threshold, the LED bias voltage was also controlled, to obtain lower LED current, and thus lower light levels. Right at the boundary between the 2 methods of control, there were a small number of tests performed at roughly the same current level, sometimes with the overlapping tests being in non-monotonic order, leading to to the slight irregularities seen in the graphs. Rather than 'clean-up' the data, the irregularities are left in, as they do not affect the overall results.

OP6-1 and OP6-8

The results obtained from OP6-1, and OP6-8, the round and square versions of the first APD design, are shown in Figure 3.22. A significant hysteresis can be seen in the results. As the light level is increased, the number of pulses per period quickly rises, and then remains relatively constant, even with increasing light levels. Even when the light level is brought back down to zero, the pulse count remains the same, or even increases. It was initially thought that this behaviour could be attributed to self-heating of the device during operation, and a corresponding shift in the threshold voltage. While there was no readily available means of measuring the device temperature in the test environment, temperature was ruled out as the cause of this effect. The relationship between temperature and reverse bias threshold voltage in APDs, has been found to be positive [32] [27]. If the device was self-heating during operation, this would actually increase the reverse bias threshold, and because the applied bias voltage remained constant during the test, this would have led to either the device being biased less above the threshold, or even under the threshold if the threshold shift was great enough. Either of these conditions would lead to a decreased response, which is the opposite of the increased response observed. The

increased response observed, is then most likely due to carrier trapping in the depletion region. Once photons are absorbed, and carriers generated, some of them become trapped, rather than swept out of the region. When these trapped carriers are later released, they too cause further avalanches to happen, and the process builds up until an equilibrium is reached, and the device continues to trigger itself on released trapped carriers, regardless of incident light. This effect makes this design quite unsuitable as a photodetector, as once it reaches a steady state, its output is no longer related to the incident light. If the voltage is lowered below the reverse bias threshold, and the trapped carriers given time to escape, the same characteristic can be seen in subsequent tests.

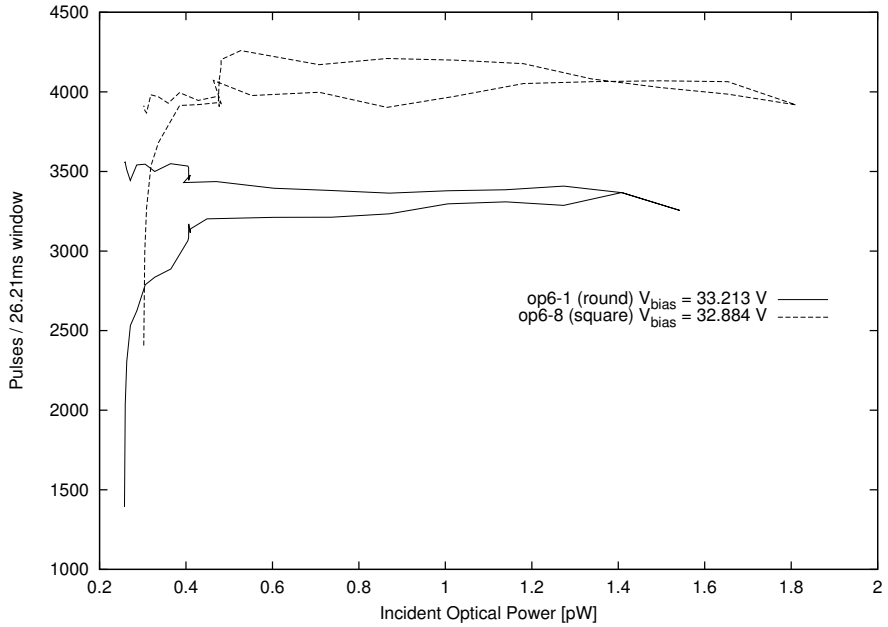


Figure 3.22: Geiger mode characterization of *n+ - p-sub* APDs with *n-base* guard rings. Both OP6-1 (round) and OP6-8 (square) have 18 µm active regions. Neither respond linearly to light to be effective photodetectors.

OP6-2 and OP6-9

The results obtained from OP6-2, and OP6-9, the round and square versions of the second APD design, are shown in Figure 3.23. There is once again a slight hysteresis between the upward and downward sweeps of the light intensity, but the

number of Geiger pulses per period does track the light intensity, and returns to the initial value in the square version, and relatively close to the initial value for the round version. The hysteresis effect is most likely once again caused by carrier trapping, though the effect is much less severe than in the first design. This design does appear to be a viable photodetector.

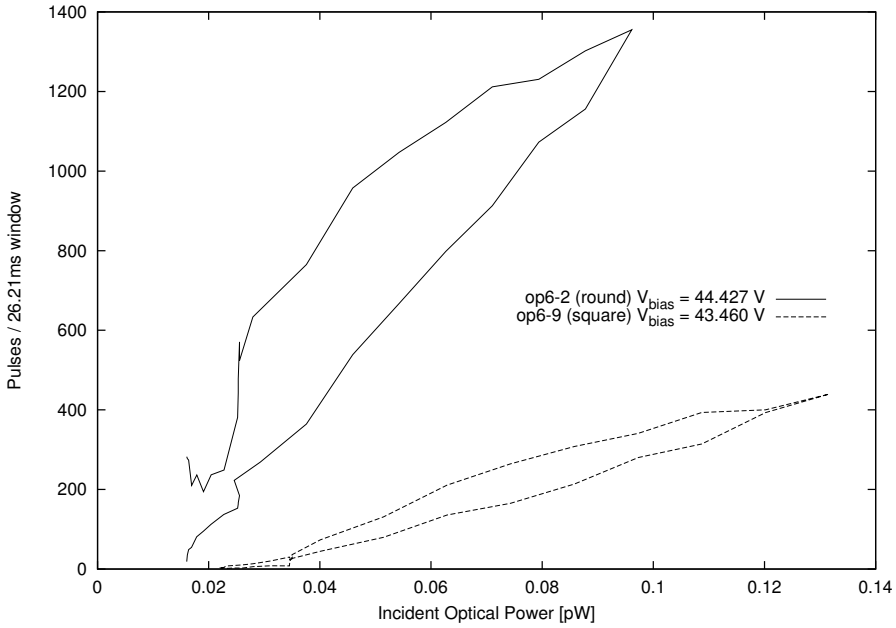


Figure 3.23: Geiger mode characterization of *p+ - hvnwell* APDs with *p-base* guard rings. Both OP6-2 (round) and OP6-9 (square) have 8 µm active regions. Both show a response which tracks the incident light level, making them viable photodetectors.

OP6-3 and OP6-10

The third design is the larger, 18 µm active-area-region version of the previous design, which was 8 µm. The Geiger response curves can be seen in Figure 3.24. The round version shows the same characteristic as the first design, with an uncontrollable hysteresis. The square device, which was biased 2V below the round device, shows quite acceptable behaviour, with little hysteresis effects, and a linear response. The round version was biased at a higher voltage, because no Geiger pulses were observed at any lower voltage.

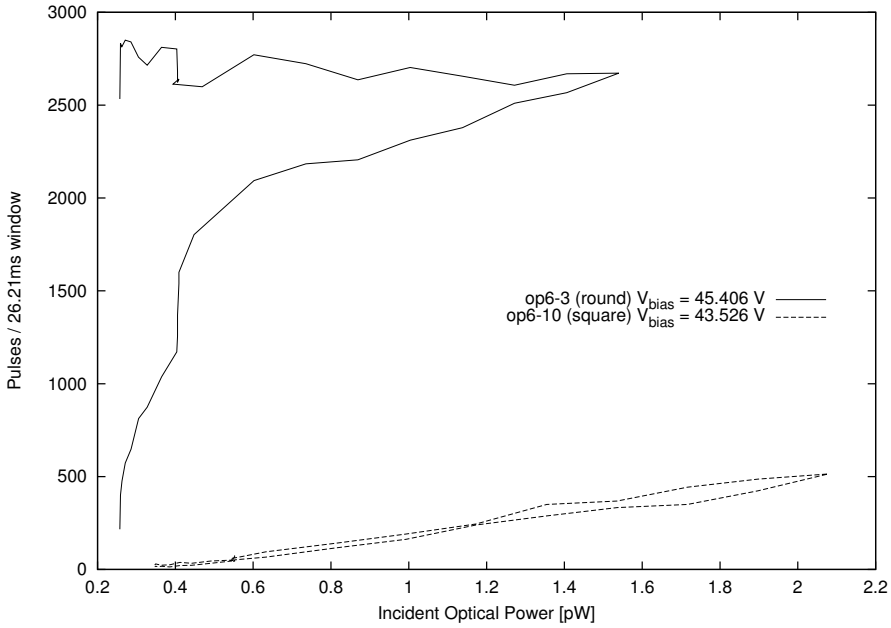


Figure 3.24: Geiger mode characterization of *p+ - hvnwell* APDs with *p-base* guard rings. Both OP6-3 (round) and OP6-10 (square) devices have 18 µm active regions. The square OP6-10 device's response tracks incident light, while the round OP6-3 does not.

A comparison of the 8 µm OP6-9 and the 18 µm OP6-10 is shown in figure 3.25. Despite the difference in area, the OP6-9 and OP6-10 devices respond very similarly. In fact, graphing both results on the same graph, without correcting for the device area, yields almost identical results. This can be seen in figure 3.26.

It is quite notable, and unexpected, that the two devices with differing active areas would show such similar responses.

OP6-4 and OP6-11

This variation of the design has no overlap between the primary active area, and the guard ring, as drawn in the CAD layout. The Geiger response curves can be seen in Figure 3.27. The round version of the device required a higher bias voltage to obtain detectable pulses, and thus shows a much steeper, yet still generally linear response with light intensity. The square version, operating at a lower bias voltage, shows an even more linear response, with fewer hysteresis effects. As both devices

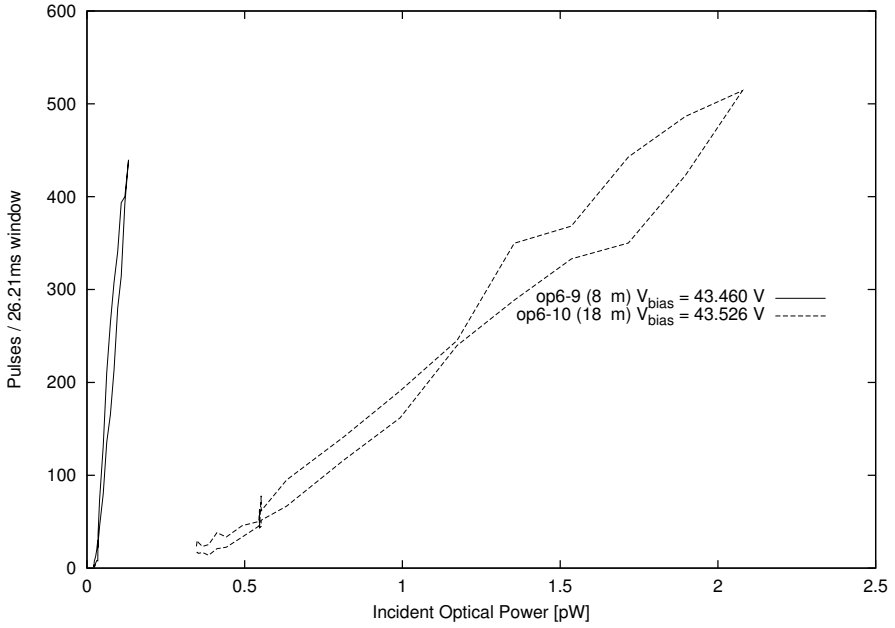


Figure 3.25: Comparing 8 µm and 18 µm *p+ - hvnwell* APDs, corrected for differing device area. The larger OP6-10 shows a slightly higher maximum pulse rate than the smaller OP6-9 does.

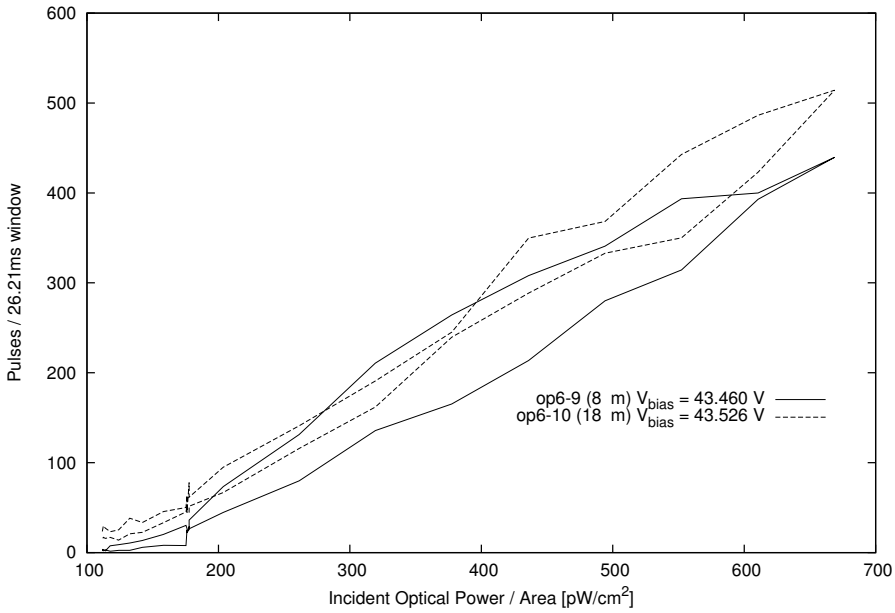


Figure 3.26: Comparing 8 µm and 18 µm *p+ - hvnwell* APDs, without correcting for device area. The response of both devices is unexpectedly similar, though the larger device does show a slightly higher maximum pulse rate.

successfully operate in Geiger mode, and produce detectable pulses, it seems there is enough lateral diffusion of the layers during fabrication, that the guard rings are still effective at preventing the undesired surface breakdown. This answers one of the main questions posed previously, as to why the previous generations of devices failed to work. Clearly the amount of overlap, or even lack of any overlap, as drawn in CAD, is not one of the factors contributing to non-functional devices in the past.

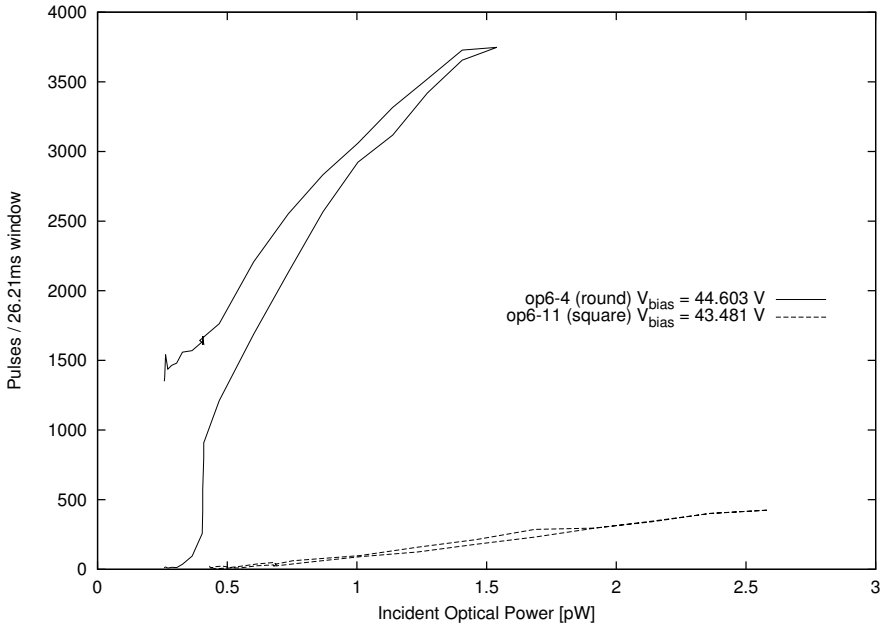


Figure 3.27: Geiger mode characterization of *p+ - hvnwell* APDs with abutted *p-base* guard rings. Both OP6-4 (round) and OP6-11 (square) have 18 µm active regions. Both show a linear response to incident light intensity, making them viable photodetectors, eliminating guard ring overlap as a cause of previous non-functional devices.

OP6-5 and OP6-12

The next variant of the design was to have a more generous overlap. The active area of these designs is 8 µm. The Geiger response curves can be seen in Figure 3.28. Similarly to OP6-3 and OP6-10, the round device requires a higher bias voltage to produce Geiger pulses, and is generally non-responsive to the incident light, but instead is rather uncontrolled. The square device, operating at a slightly lower bias

voltage, produces a more of a linear response, with little hysteresis effects.

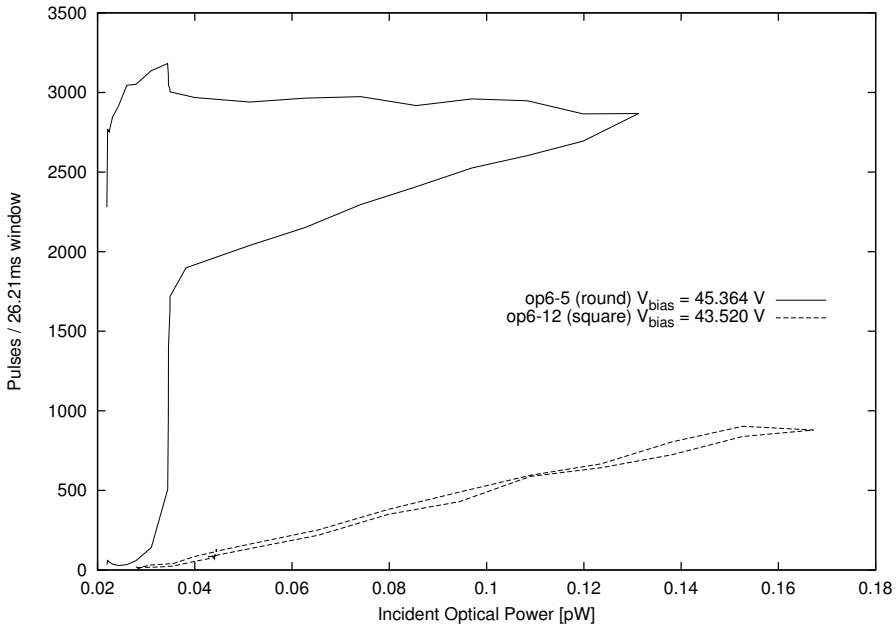


Figure 3.28: Geiger mode characterization of *p+ - hvnwell* APDs with large-overlap *p-base* guard rings. Both OP6-5 (round) and OP6-12 (square) have 8 µm active regions. The round OP6-5 does not show a linear response to incident light, whereas the square OP6-12 does.

OP6-6 and OP6-13

This design is the 18 µm version of the previous design. The Geiger response curves can be seen in Figure 3.29. The round device again requires a higher bias voltage to produce Geiger pulses than the square device. This size of the round version does however have a linear response with incident light, though with some hysteresis effects visible. Again, the square device, operating at the lower bias voltage, produces a more linear response, with a minimal amount of hysteresis. As OP6-12 and OP6-13 are different sizes of the same design, and both have reasonably well behaved response curves, it is interesting to compare their responses without correcting for device area, as was done for OP6-9 and OP6-10. The comparison can be seen in Figure 3.30.

In this instance, the smaller device, operating at a slightly higher bias voltage,

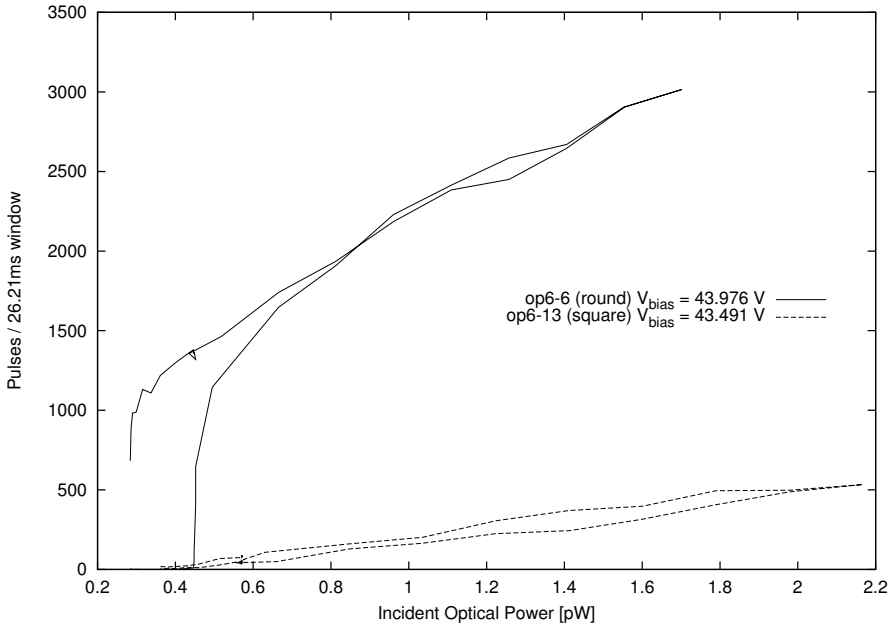


Figure 3.29: Geiger mode characterization of *p+ - hvnwell* APDs with large-overlap *p-base* guard rings. Both OP6-6 (round) and OP6-13 (square) have 18 µm active regions. Both show a linear response to incident light.

produces more Geiger pulses than the larger device operating at a slightly lower bias.

OP6-7 and OP6-14

The last designs, OP6-7 and OP6-14 are substantially different than the others. Rather than being built in a HV region of the CMOSP8G process, they are built in a low voltage region, and also use more traditional low voltage doping layers. The reverse bias threshold is much lower, around 20V, which is substantially lower than the other devices, at around 32V and 43V. The Geiger response curves can be seen in Figure 3.31. This design shows extremely good relation between the light intensity, and the pulse count per unit time. The hysteresis effect, between the increasing and decreasing sweeps of the light intensity, is the smallest of all the devices. Unlike the other devices, which require fastidious setting of the bias voltage, to get correct and repeatable operation, OP6-7 and OP6-14 are quite forgiving of bias voltage setting. Because of this, OP-14 was selected to show the effect of sweeping the bias

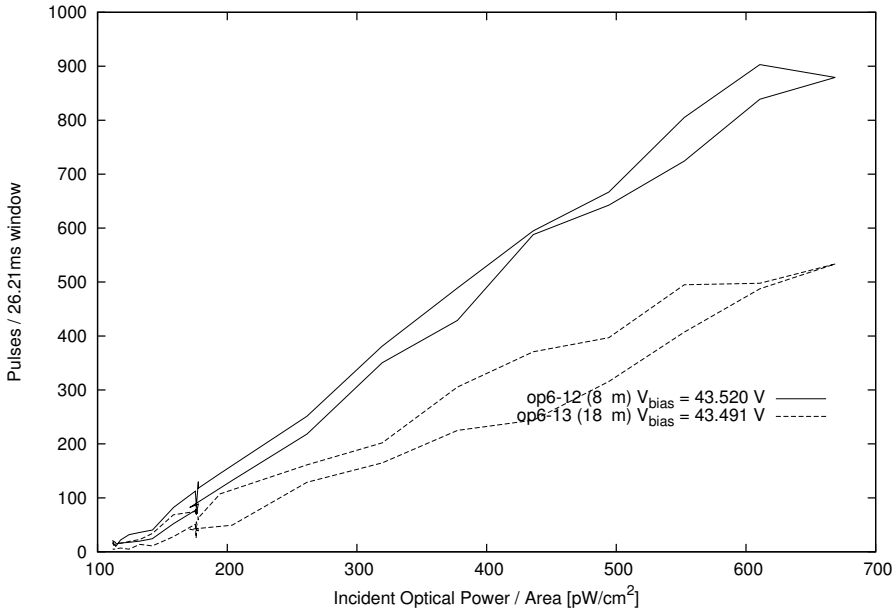


Figure 3.30: Comparison of two square *p+ - hvnwell* APDs with large-overlap *p-base* guard rings, OP6-12 (8 µm) and OP6-13 (18 µm) without correcting for device area. The smaller OP6-12, at a slightly higher bias voltage, shows a larger response than the larger OP6-13, at a slightly lower bias voltage.

voltage. As the bias voltage is increased, the amplitude of the pulses increases as well, so the oscilloscope voltage range had to be increased to avoid clipping at the highest bias voltage settings. As a result, the software pulse detector also had to be slightly adjusted for this test. The voltage delta threshold for detecting a pulse needed to be increased, as the original threshold of 3 mV led to noise being incorrectly detected as pulses. The threshold was increased to 20 mV. The family of response curves resulting from sweeping the bias voltage, is illustrated in Figure 3.32. Because of the adjustment of the pulse detector, the bias sweep curves should not be directly compared with the other graphs. The higher pulse detection voltage threshold may also result in smaller pulses being missed This may occur following a large pulse, but before the voltage across the quench resistor has returned to 0 V.

Both the round and square versions of the *p+ - deep-nwell* APD were also characterized in a pulses-per-period versus bias voltage configuration. The bias voltage was swept up and back down to identify any hysteresis effects in the response. Two

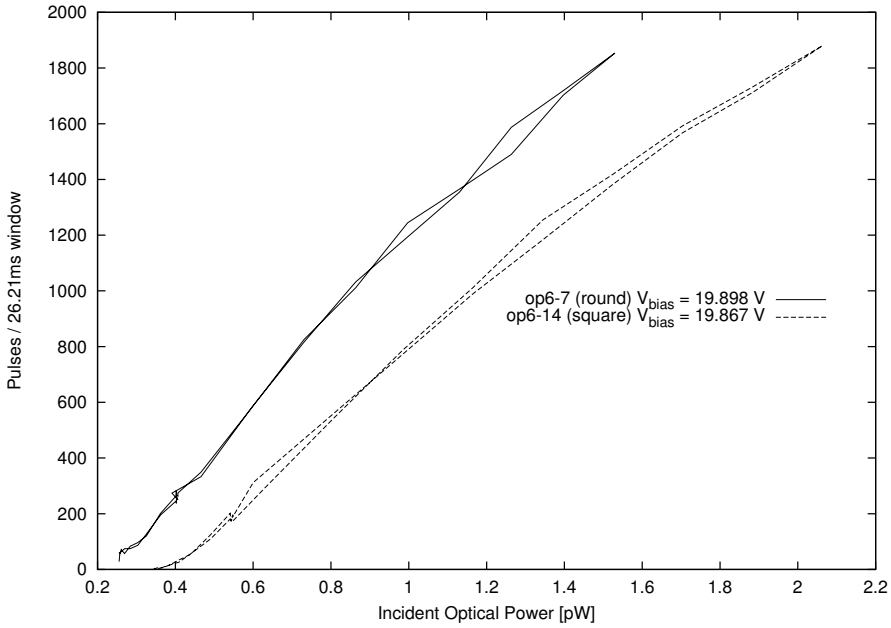


Figure 3.31: Geiger mode characterization of *p+ - deep-nwell* APDs with *p-well* guard rings. Both OP6-7 (round) and OP6-14 (square) have 18 µm active regions. Both devices show very linear response to incident light, with minimal hysteresis effects between increasing and decreasing incident light.

incident light levels were used for both devices; dark, and 667 $\mathrm{nW\,cm^{-2}}$. The results of this characterization can be seen in Figure 3.33. No significant hysteresis effect is seen with respect to the bias voltage. It is interesting to note that the round device shows a higher dark pulse rate, and lower pulse rate when exposed to light compared to the square device, at all bias voltages tested. The square device has a lower dark pulse rate, and higher pulse rate in response to light, both desirable characteristics for SPAPD devices. An effort was made to test both the round and square devices in the very center of the field of view of the probe station microscope. Even so, the effect of the non-uniform illumination light field may be a contributing factor in the results of this characterization.

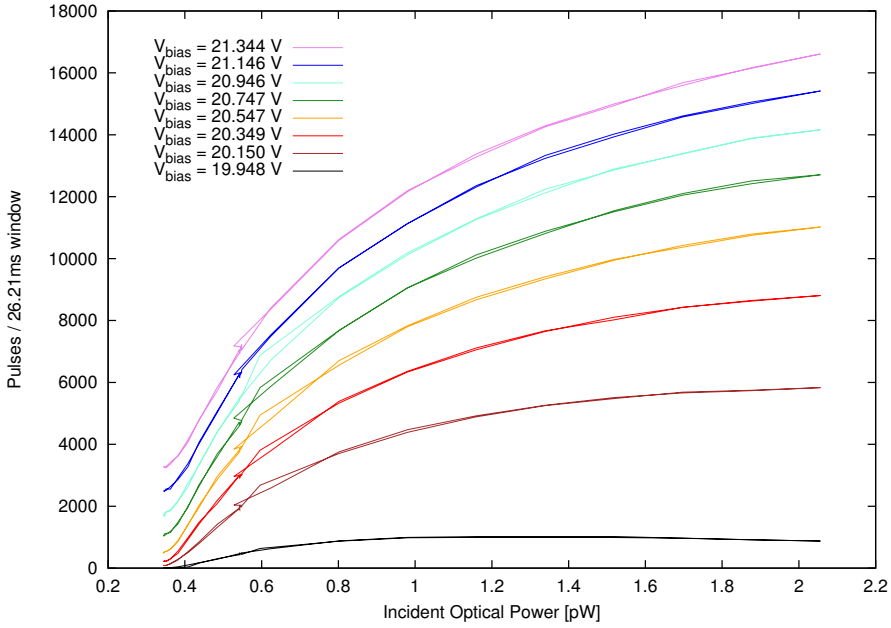


Figure 3.32: Square *p+ - deep-nwell* APD bias voltage sweep. As bias voltage increases, so do both the dark pulse rate, and the pulse rate at any given incident light level.

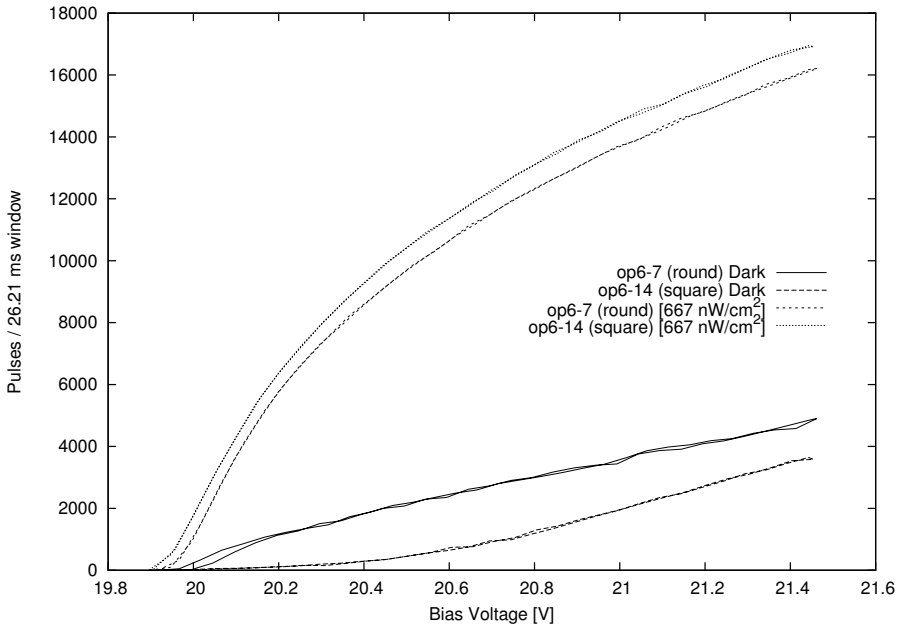


Figure 3.33: *p+ - deep-nwell* APD pulse rate vs. bias voltage characterization. No hysteresis effect with respect to bias voltage is seen. Square device shows lower dark pulse rate, and higher light-exposed pulse rate at any given bias.

3.8 Summary

From the numerous tests and characterizations that were performed on all the devices, some conclusions can be drawn.

Considering the photovoltaic testing, despite the obvious systematic error affecting the absolute value of all of the results, the results obtained still appear useful for comparing the devices against each other. Comparing the results of all the devices measured, there was an extremely broad range of responsivities observed, $0.000\,986\,\mathrm{A\,W^{-1}}$ to $1103.1478\,\mathrm{A\,W^{-1}}$. None of the devices showing higher responsivities are currently used as the photovoltaic mode photodetectors in the LoC system.

All of the devices designed and fabricated in the last generation, OP6, functioned in some manner in Geiger mode. The *n+ - p-sub* devices, OP6-1 and OP6-8 both appear to have suffered from severe trapping effects, and in their current state, are not suitable for use as photodetectors. The response of those devices did not track the incident light input very well. OP6-3 and OP6-5, both round devices, showed similar issues in their response, while their square counterparts did not. Once exposed to the maximum light intensity used during the test, the number of pulses per time period remained relatively constant, and subsequently did not decrease with decreasing incident light. The remainder of the devices showed reasonably linear response to incident light, and appeared to be good candidates to investigate further, for use as Geiger mode photodetectors in the LoC project. The lower voltage devices, OP6-7 and OP6-14, showed the best response, with the least amount of hysteresis between the upward and downward sweeps of the incident light.

It was very unexpected to find both sets of devices that were fabricated in two different sizes, OP6-9 / OP6-10, and OP6-12 / OP6-13, to have such a similar Geiger response. With OP6-12, the smaller of the two similar designs, but operated at a higher bias voltage, which actually showed a greater response than the larger OP6-13, it seems to indicate that the bias voltage was a greater factor affect-

ing device response, than physical size was.

From the results obtained and presented in this chapter, it is clear that Geiger mode APDs can be successfully implemented in the TDSI CMOSP8G process, and thus integrated with the rest of the LoC system under development. Much further study, and refinement of the devices, and associated quenching circuits, will be required before an optimal detection system is achieved. This work is a monumental step towards that goal, as functional Geiger mode devices have not been achieved previously in this project.

Chapter 4

Register-Based SPI Interface

For a LoC system to be useful, there must be the ability to control all functions in the system, as well as extract the measurement data from it. It is not feasible to implement all processing and logic required directly in the LoC system, due to the large minimum feature size of the 0.8 µm TDSI CMOSP8G HV process required for the high voltages necessary for CE, and available silicon area, and thus cost. In the past, Philip Marshall investigated synthesizing an open source USB core from opencores.org and found that implementing a USB controller directly in the TDSI CMOSP8G process would require more than 8 mm^2 of chip area [33]. Because this area is very large compared to the rest of the area required, only the required custom hardware, as well as a much simpler SPI slave control interface to that hardware, are implemented on-chip. The intended external controller is a microprocessor, or FPGA, due to the hard real time requirements of some functions, such as heater control for PCR.

The interface to the LoC system is an SPI slave controller implemented on-chip. SPI is a good choice of interface bus, as hardware-based SPI peripherals are very common on a wide range of external microprocessors, which is the intended external controller in this system. There are also a very large selection of other commercial chips available with SPI interfaces that may be required externally during development of a functioning system. Having all such peripherals, including the custom LoC system, on one bus, simplifies the overall system design.

In this chapter, the existing control interface from the previous generation of

the system will be briefly discussed, noting its shortcomings, and desired improvements. A replacement control interface is presented, which was designed to eliminate the shortcomings and implement the desired improvements over the previous design. Test results, as well as important performance metrics, will be given.

4.1 Previous Designs

Previous generations of the LoC system had fewer functions on-chip to control, and each controllable function had a unique op-code. The SPI command was only one byte, with three bits used for the opcode, and the remaining 5 for data to control the functions. Each opcode had a unique bit mapping for the data field. When more than 8 commands were required, major changes were required to both the existing external firmware on the microprocessor as well as the VHDL of the SPI controller itself. Previously, I had made these changes in the VHDL of the SPI controller when more than 8 functions were required, but the code was very difficult to modify, and difficult to maintain in general. The basic command was extended to 16 bits, and the op-code field extended to 4 bits, but if more than 16 op-codes were required in the future, this would again require all of the existing external software to be modified, and the entire SPI peripheral reworked. The decoding of each of the unique commands took place directly in the SPI controller, so even small changes to one subsystem would require major changes, and potentially introduce new bugs to the entire SPI peripheral.

Another drawback of the previous design was all of the commands, with the exception of the read ADC command, were write only. Even the ADC result was not stored internally, as the SAR ADC was clocked by the SPI clock, and the output of the comparator was connected directly to MISO output. This increases the complexity of the external control software as no current system state can be read back. If the state of the system is lost for any reason on the external controller, there is no ability to be able to read the current system state back to know how to proceed. There was also no way to read the last ADC result back without taking a new measurement.

In the previous design, ICKAALC7, five physical digital I/O pins were used to set the hardware state of the chip, for example to select between internal and external clocks for the ADC, PWM controller, or selecting between other internal functions. As the design of the LoC system was in the past pad-limited, using many digital I/O pins for selectors that are typically set to constant values externally is rather expensive and wasteful of chip area. This also affects the microfluidic design bonded to the surface of the silicon die, which can cover non-essential pads, re-using that area, but can not occupy space above pads which must be connected to for correct operation. Fewer essential pads increase the flexibility and versatility of the microfluidic design.

While the choice of SPI as the protocol for the controller was good, the existing implementation had some deficiencies that needed to be improved. The main deficiencies were the lack of a common command format for all commands, the ability to add new functions without disturbing the implementation of existing functions, both in the controller itself and in the external control software, the inability to read state back from the system, and the use of physical I/O pins to set system hardware state.

4.2 Register-based Design

To address the shortcomings of the previous design, a new design was implemented. Because the architecture of the new controller is significantly different than the previous one, the new controller has been entirely re-written. The new design is based on an addressable 'register' file architecture. The standard command length is now 16 bits, rather than 8. The high order byte is used for the op-code and address, and the low byte is used as a generic data field. All registers are one byte wide. The digital logic was designed so that all of the registers in this design are actually implemented physically as latches, as the latches available in the TDSI digital I/O cell library are 49% of the area of the D flip-flops. The details of the implementation will be discussed below, but for the remainder of this chapter, 'register' means register functionality implemented with latches.

Bits	Op-Code
00	noop
01	read
10	write
11	invalid

Table 4.1: Op-Code Definition

Rather than having a unique op-code for each on board function, having standardized registers for all functions reduces the op-code count to 3, namely read, write, and noop. These op-codes are sufficient to cover all operations required, regardless of how many actual functions are implemented on the chip, and any additions, changes, or deletions to those functions in future versions. Instead of using the opcode to specify which function is being controlled, an address is used to specify which register, and thus which internal function is being accessed. This allows both the design of the digital logic in the chip, as well as the external software to be very modular. This modularity allows for new on-chip hardware to be introduced, existing functionality of any subsystem to be changed, or removed all together if not required in the future, and the corresponding change done to the firmware that controls that function, without affecting any other subsystem or its associated firmware. The 3 op-codes are represented with 2 bits, and the mapping between the bits and the opcodes can be seen in Table 4.1. The remaining 6 bits in the first byte in the command sent via SPI are available for address use, giving 2^6 or 64 possible addresses. Nowhere near 64 addresses are required for the currently implemented on-chip subsystems, so the address decoder implemented is only 5 bits wide, with the highest order bit reserved for future expansion. Currently this 'extra' address bit has been associated with the op-code decoding, so if a command is sent with a presently invalid address with the MSB set, there will not be aliasing to a valid address, to aid in software debugging.

The controller is designed around a standard SPI slave model, with the typical 4 wire interface: chip select ($\overline{\text{CS}}$), master out slave in (MOSI), a tri-state master in slave out (MISO), and clock (SCK) pins. Also an external chip reset ($\overline{\text{RST}}$)is

provided.

The controller is designed to read bits on the MOSI pin on the rising edge, and transmits data on the MISO pin on the falling edge of SCK, with the clock idling low. In common SPI parlance, as described by Freescale's SPI Block Guide [34], this is defined as CPOL=0, CPHA=0. When $\overline{CS}$ is asserted, the internal bit counters are reset, and the controller is ready to begin receiving data. The basic SPI command structure can be seen in Figure 4.1. The first 8 bits (most significant byte) is shifted into one shift register to decode the op-code and address. If the op-code specifies a read, the address is decoded, the MISO pin is enabled, and the contents of the register addressed, is shifted out during the next 8 clock cycles. If a write was specified, the next 8 bits are shifted into a second 7-bit wide shift register, with the least significant bit coming directly from the MOSI pin. This allows the controller to complete the write with the last cycle in the transfer in which the clock is high, without requiring any extra clock edges that would not normally be available. Both the rising edge when the LSB is valid, and the falling edge returning the clock to the idle low state are used in the design.

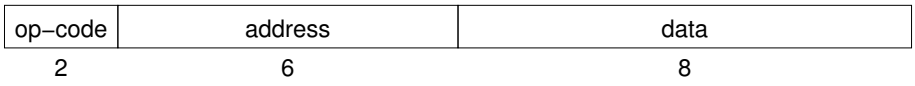


Figure 4.1: Format of SPI commands for new register based SPI interface

As mentioned previously, all of the 'registers' in the register file are actually implemented with latches, rather than more traditional flip-flop based registers. There were 2 reasons to implement the register file in this fashion. The flip flops available in the TDSI DL08G digital logic cell library are considerably larger than the latches. Looking at the Verilog output of Synopsys Design Compiler, which was used to synthesize the VHDL RTL design using the DL08G standard cell library, the registers were implemented using cell 'dlad1', a 1 bit D-type latch. This latch has an area of $1140.0\,\mu m^2$, whereas the equivalent d-type flip flop, 'dffd1', which has one data input, and complimentary outputs, same as the 'dlad1' actually used, is $2316.0\,\mu m^2$. The latch being 49% the size of the D flip-flop, and there being up to 256 of them given the 5-bit address space (32 registers * 8 bits), the slight in-

crease in complexity of the controlling logic to use latches rather than flip-flops, is more than made up for in total area savings. The other reason latches are preferable over registers in this situation, is using latches rather than flip-flops eliminates the need for a clock tree, as well as the dynamic power that would be consumed if the SPI clock was distributed to potentially 256 flip flops if the 5-bit address space is fully populated. With this design, each output of the address decoder only drives the 8 latch inputs of one register, and only one latch enable signal toggles per write transfer. This also leads to a savings in core area.

These registers are combined together into the 'reg_file' entity, as seen in Listing D.4, which has both a parametric data width of each register, and number of registers in this group. Even with a 5 bit wide address field, the address space is not fully populated in the current design, and logically related groups of function registers are grouped together with gaps in addresses between the groups, to allow for expansion. Implementing the 'reg_file' entity, where the number of registers is specified as a parameter, makes it easy to populate the address space with groups of registers with gaps in the addressing, and without having to instantiate each register individually. This also makes adding or removing registers in the future extremely easy. As can be seen in the VHDL source for the 'spi_reg_interface' in Listing D.1, there are only 2 lines of code for each group of registers. To add or remove registers in the future, only at most 5 literal values need to be changed. The address map, as implemented in this version of the controller, can be seen in Table 4.2.

All of the addresses listed as 'Reserved' are not implemented currently, and while some speculation about future usage is made, those addresses could actually be used for any necessary function in future designs.

The register groups, along with an address decoder, various shift registers, and glue logic to generate the read and write signals used internally, is combined to form the entire SPI controller for the system. An overall schematic of the system can be seen in Figure 4.2.

The unused address decoding logic that would connect to the unimplemented registers, or other registers where those signals are not used, such as read only

Address	Function
0x00	Serial Number / Version (ro)
0x01	Function Mask (ro)
0x02	Config Register 1
0x03	Config Register 2
0x04 - 0x05	Reserved - future config
0x06	vctrl - high voltage regulation control
0x07	HV bank 1 - HV channel 0-3 controls
0x08	HV bank 2 - HV channel 4-7 controls
0x09 - 0x0C	Reserved - additional HV channels
0x0d - 0x11	Reserved
0x12	Heater PWM
0x13	Vsubtract
0x14 - 0x17	Reserved - future optical control
0x18	Optical intensity control
0x19	GPIO
0x1A - 0x1C	Reserved for future GPIO
0x1D	ADC source select
0x1E - 0x1F	ADC result (read only from SPI)

Table 4.2: Register Function Layout in 5-Bit Address Space

registers, or the write signals for the ADC registers, is optimized out in the syntheses of the VHDL, and is not actually implemented in the final design.

A new feature implemented in this design is the concept of read-only registers, which are used for chip version numbers at address 0x00, and chip capability bit flags, at address 0x01. This function allows the external firmware to determine what version of the chip is connected, and what features the connected chip actually implements, allowing the same firmware to control different versions of the chip with different subsystems absent or present in the design.

The two configuration registers at address 0x02 and 0x03, replace the five physical I/O pins in the previous design. As well, as many new digital selectors were added, that are used for setting basic system state. A list of the function of all of the bit locations in these 2 configuration registers can be seen in Tables 4.3 and 4.4.

As mentioned previously, it is desirable to eliminate as many physical I/O pins as possible. These software accessible configuration registers eliminate physical I/O pins, as well as allow easier programmatic control of the chip state. With the

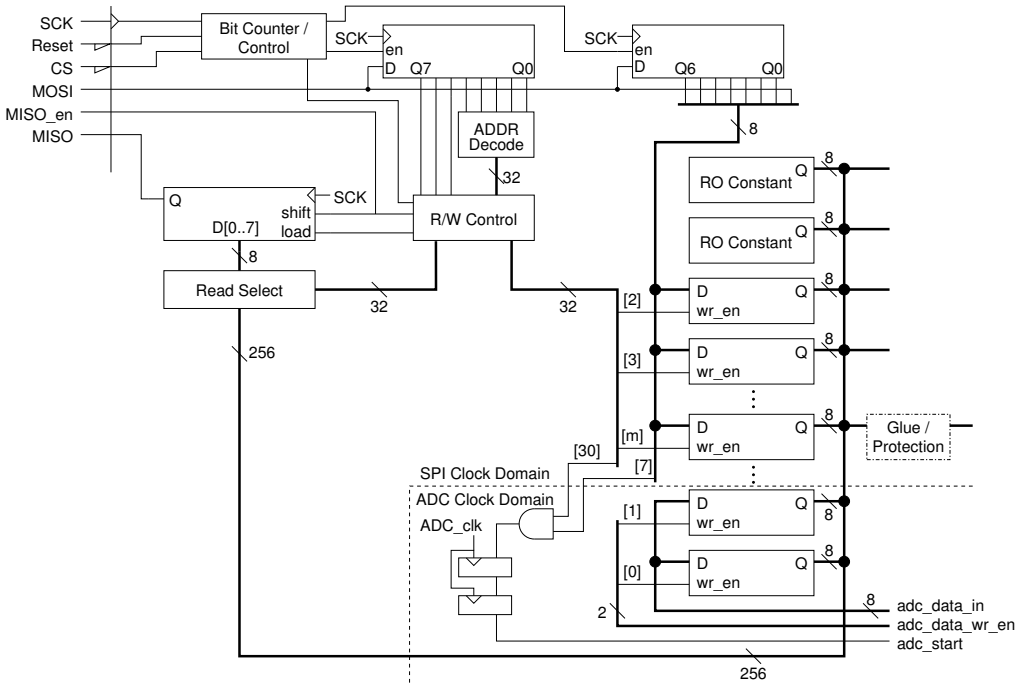


Figure 4.2: Schematic showing design of new digital control system

Bit	Function	Description
0	vco_enable_b	Enables or disables internal VCO
1	vco_mode	
2	vco_override	Selects between internal VCO and external pin
3	vmon_select	Selects the source of the HV monitor cct.
4	boost_pwm_select	Selects clock source for boost converter
5	Reserved	Unused
6	Reserved	Unused
7	Reserved	Unused

Table 4.3: Bit names and descriptions of configuration register 1 at 0x02

software configuration registers, the previous 5 I/O pads required to set the chip state are eliminated, and new configurable chip state options added. When considering the microfluidic design, 5 fewer pads are now essential, and must remain exposed. The area above them can be used for microfluidic designs instead. This allows for more flexibility in both the chip state control, and the microfluidic design.

Another feature implemented in the new design, is an SPI command initiated synchronous reset. A synchronous reset can be performed by writing to 'read only' register 1, which will reset all chip state to power on reset state. This allows the

bit	Function	Description
0	Vreference_mode	controls the voltage bias reference
1	Ireference_mode0	controls the current bias reference
2	Ireference_mode1	controls the current bias reference
3	boost_en	Enables the boost converter controller
4	cp1_en	Enables charge pump 1 (300V experimental)
5	cp2_en	Enables charge pump 2 (40V)
6	hv_free_run	Disables HV regulation control
7	synchronous reset	Programmatic chip reset

Table 4.4: Bit names and descriptions of configuration register 2 at 0x03

controlling software the ability to reset the entire controller to a known state, at any given time, without having to physically interrupt power to the chip, or use the asynchronous hardware reset pin. This potentially eliminates another essential pad that must remain accessible through the microfluidic layers.

Some register outputs are fed through glue logic, or protection logic as shown on the right side of Figure 4.2, before being connected to the internal logic. This ensures invalid inputs cannot be set by accident through the SPI interface. This is particularly important for the HV control registers, as each channel is controlled with two bits, one to drive the channel to HV, one to drive the channel to GND, or neither connected to let the channel float. Protection XOR gates are inserted so that the channel cannot be driven to HV and GND at the same time, as this would short the HV power supply. Similar logic is used to ensure the outputs of the configuration registers drive the chip to a 'safe' state on power up. This ensures that if some built in functions are not working, or if the configuration registers could not be written to, the chip would still be as testable as possible with the power-on defaults.

Most of the internal functions controlled through the interface are essentially asynchronous, though technically in the SPI clock domain. A special case that is not in the SPI clock domain is the on-chip ADC converter. The ADC runs on the VCO clock, which is selectable between internal and external source through the configuration registers. In most instances, the VCO clock will not be synchronous with the SPI clock. Because of this multiple clock domain design, signals crossing

the clock domain boundary between the SPI and VCO domains must be synchronized to the destination clock domain. The ADC start signal crossing into the ADC clock domain is double-flopped to synchronize it to the VCO clock, and the ADC result is protected with flag bits to indicate when the data is valid. The ADC result registers are read-only from the SPI interface, though 'writing' to the MSB result register is used to initiate an ADC conversion. The data inputs and write signals to the two ADC result registers are connected to the ADC controller, rather than the SPI controller. To start an ADC conversion, a write is performed to the most significant byte ADC result register, address 0x1E, with the most significant bit of the data field set to one. This signal is synchronized to the ADC clock domain using a double flip flop synchronizer, and starts an ADC conversion. The successive approximation ADC performs the conversion, and the result is written into the two ADC result registers based on the VCO clock. The output of the ADC result registers is readable through SPI during the conversion, but if the conversion is not yet complete, the data might not yet be valid. Because the ADC is 12 bits, the highest order 4 bits of the sixteen-bit result are not used for valid data, and are instead used as flag bits to indicate if the conversion is complete, and the data valid. As soon as a conversion is started, the highest order 4 bits are set, indicating the data is invalid, and the ADC performs the conversion. The least significant byte is written into the LSB result register, then on the next VCO clock, the highest order 4 bits of the result are written to the MSB result register, without disturbing the flag bits. On the following clock, the 4 flag bits are cleared, indicating the conversion is complete, and the result data is valid. By changing only the flag bits in the final write, the rest of the data is guaranteed to be valid after the flags are cleared, as all the data bits were written in the previous VCO clock cycle.

Functional Verification and Test Bench

To thoroughly test the design before it was implemented, a test bench was implemented, based on the test bench of the previous design, but with some newly added features to further test corner cases that couldn't be simulated with the previous test

bench. The test bench reads a series of SPI commands from an external file, and simulates an external SPI master controller, sending those SPI commands to the designed SPI slave interface. Having the SPI commands read in from an external file, allows quick changes to the test conditions without having to recompile the test bench itself. Mentor Graphics ModelSim was used to verify both the VHDL behavioural code, and the post-syntheses Verilog code, to ensure correct operation. New features added are the ability to send arbitrarily many 8-bit SPI transactions back to back, without de-asserting chip select. Being able to send two or more SPI commands back to back, without de-asserting chip select, increases the effective transfer rate, this being able to ensure the controller handles this condition properly was a requirement of the test bench. The test bench also now allows 8-bit incomplete commands to be sent, then de-asserts chip select, and then continues sending valid commands, making sure the SPI slave controller does not take any action based on the incomplete command. With these two new abilities, the test bench is able to cover many more scenarios than the previous version, giving a much higher confidence that the design will function in more use cases. Many combinations of commands were tested using the test bench, such as two consecutive reads, two consecutive writes, read following a write, write following a read, and combinations of those, with incomplete commands inserted as well. Because of this added test functionality, some bugs were found and corrected in various command combinations, leading to a much more solid final design.

4.3 Results

The new digital controller was tested in silicon by David Sloan, who was developing new RTOS based external control software for the LoC system chip. No bugs or other errata were found in the new SPI interface. All implemented registers can be written and read back. The ADC functions which require the clock domain crossing were also tested, and found to function as designed.

The chip area required to implement only the SPI controller is not available, as all the digital logic for the entire system chip is synthesized, placed, and routed as

Metric	LC7 SPI Interface (non-modular)	LC8 SPI Interface (modular)
Dimensions [µm]	1340 x 1300	1340 x 1380
Area [mm^2]	1.742	1.849
Clock tree required	no	no
Subsystems / registers	10	14
SPI command length [bits]	16	16
State readable	no, only ADC result	yes, all
Additional physical state I/O pins	5	0
Modular / scalable / maintainable	no	fully

Table 4.5: Comparison of the new SPI interface design vs. the previous version

one block. The additional logic, beyond the SPI interface itself, remained largely the same between the previous design and the current one, thus it is valid to compare the overall properties of the digital logic of both designs.

As can be seen in Table 4.5, the functionality has been improved, solving all of the issues with the previous design, as well as adding 4 new registers. The area required to implement the new design is only marginally larger than the previous implementation. In fact, only one additional row of digital standard cell area was added to the digital place and route region, to successfully route the new design.

4.4 Summary

In this chapter, the previous design of the digital control interface of the LoC system chip was discussed. The main drawback was that the previous controller had custom op-codes and command layout for each op-code implemented, which lead to an extremely difficult to maintain design, that required changes to the existing functionality in order to add new features. Other important shortcomings in the previous design were the inability to read back any of the state of the system, and external IO pins required to configure the state of the system. External pins were previously used to set the state of, or to enable or disable certain subsystems, such as clock source selects, or charge pumps.

Taking these shortcomings into account, a new design with a standardized com-

mand structure, and more modular architecture was developed. The new design was implemented in VHDL, and simulated with a custom test bench in Mentor Graphics ModelSim. The design was then synthesized with Synopsys Design Compiler to the TDSI DL08G standard cell library, and placed and routed in Cadence First Encounter. The area required to implement the new design, which addresses the problems of the previous design, as well as implements new features, is minimally greater than that required for the previous design. The design was implemented in TDSI Semiconductor CMOSP8G technology, and verified in silicon. No problems were found during testing of the manufactured ASIC. The design and implementation of the new digital controller was a success, and provides the LoC system a flexible and easily maintainable design for future implementations.

Chapter 5

System Chip ICKAALC8, and Photodiode Test Chips

ICKAALC8, the eighth generation system chip implemented in the LoC project, will be briefly discussed. The main features of the chip will be mentioned, along with the design considerations used for determining the final layout of all the subsystems in the chip. In addition, the photodiode test chips implemented during this work will also be briefly discussed. Micrographs of all the chips will be presented.

5.1 ICKAALC8

To test the overall LoC system, as well as the integration of all of the subsystems that comprise the complete LoC system, 'system chips' are produced periodically. The goal of the system chips is to actually perform PCR and CE, and evaluate the performance of the entire system. After a system chip has been fabricated, all electrical subsystems are tested, and if everything appears to be functional enough to benefit from running further experiments, a microfluidic design is created. The microfluidic design is then fabricated on top of wafers, which were held at DALSA when the original multi-project-wafer containing the system chip was fabricated. The initial design has to be mindful of the microfluidic design, which could later be added, as there is no opportunity to make any changes to the CMOS design if the fluidic design proceeds. Because the tapeout time frame, from submitting a design, to receiving silicon back, is in excess of 6 months, the newest features and designs

are usually included in the system chip. If the new subsystem works, then an entire 6 month tapeout cycle is saved, versus first verifying it in a separate test chip.

ICKAALC8 is the eighth generation system chip produced in the LoC project. The size of the chip was 5 mm by 5 mm. The main features of LC8 are:

- The new register-based SPI interface presented in Chapter 4
- The new ADC interface that was also presented in Chapter 4, to interface the ADC, which runs on a different clock, to the SPI interface, which runs on the SPI clock
- An on-chip VCO, to clock the the ADC, and some other on-chip subsystems which depend on a continuous clock, such as the capacitive charge pumps, and boost controller
- An experimental 300 V capacitive charge pump designed by Al-Hadad [35]
- A 40 V cap active charge pump designed by Marshall
- An on-chip boost converter controller, to generate the 300 V required for CE, should the charge pump not function correctly
- An HV monitoring and regulation system
- A current monitor on the HV rail, which can aid in debugging fluidic connections, and monitoring injection and separation phases of CE
- 5 tri-state HV switches, able to switch the electrodes which connect to the microfluidic wells to either HV (300 V), ground, or left floating [35], [14]
- 3 bi-state HV switches, able to switch CE electrodes to either ground, or floating [35], [14]
- 4 large metal contacts, intended to contact the bottoms of the fluidic wells, to enable injection and separation in CE

- Multiple photovoltaic mode photodiodes (Geiger operation had not yet been achieved when LC8 was designed) for differential detection by Martin [15]

- A large heater switch transistor, and associated control capable of switching large currents, for a wire trace heater to be placed under a large fluidic well, to perform PCR on-chip, by Ho [36].

The VHDL developed for the new SPI interface, along with existing VHDL of other subsystems implemented previously, was compiled with Synopsys Design Compiler. The resulting verilog netlist was imported into Cadence as a schematic, and included in the top level schematic of the entire chip, to facilitate LVS verification. The verilog netlist was also imported into Cadence First Encounter, to place and route all of the digital components into one area of the chip. Timing analysis was carried out in First Encounter on the digital subsystems, to ensure there were no timing violations post place and route. No clock tree was added to the design, as was discussed in Chapter 4, because the new SPI interface's registers were implemented as latches, and the clock connects to relatively few devices, a clock tree was not required to meet timing requirements. The digital block can be seen in callout 1 in the micrograph presented in Figure 5.1. Similarly, a netlist of the majority of the analog components was exported from Cadence, and First Encounter was used to place and route a majority of the analog subsystems. The analog components placed and routed using First Encounter are in the block seen in the lower left side of the chip.

A preliminary microfluidic layout was developed, and the on-chip devices, which must align with the microfluidics were placed first. These devices include the large metal contacts to the microfluidic wells at callouts 4, the heater switch and contacts at callout 10, and the photodiodes at callout 11, which must be under the end of the separation channel. The pad frame was then developed, attempting to keep all the pads, which must be accessible for the operation of the chip, away from the microfluidics. Pads that are only required for electrical testing, before the fluidic

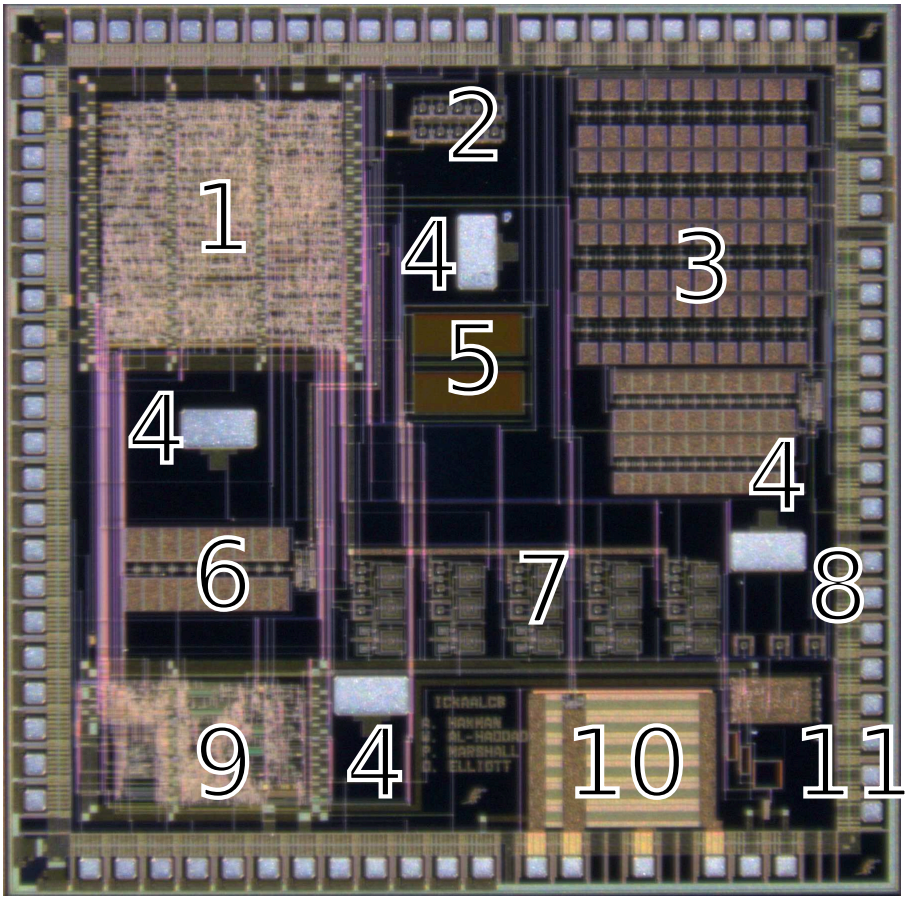


Figure 5.1: Micrograph showing ICKAALC8, the 8th generation LoC system chip

design is added on top, are grouped together and placed such that the microfluidic design could cover them if necessary. Generally, the pads on the right side of the chip are for characterizing the charge pump, and can be covered with the fluidic design. The essential pads, including the digital and analog power buses, as well as the SPI interface pads, are grouped together on the left side of the chip, and the leftmost few pads of the top and bottom.

Once the pad frame was determined, and the essential devices for interfacing with the microfluidics were placed, the remaining large devices, along with the digital and analog blocks, were placed manually in First Encounter. The 300 V charge pump can be seen in callout 3, and the 40 V charge pump between the digital and analog blocks at callout 6. The boost converter controller is located at callout 2, and the large HV monitoring divider resistor at callout 5. The heater switch, and

associated controller, are at callout 10 in the bottom right corner, the tri-state HV switches for the electrode contacts are at callout 7, and the bi-state HV switches are at callout 8.

Once all of the devices were placed, and routed in Encounter, the design was imported back into Cadence, and DRC and LVS were run, to verify no design rules were violated, and all of the devices were correctly connected together. Many manual fixes were required to get LVS to pass, and for the design to be DRC error free.

Microfluidics were never fabricated for this design, as the SAR-ADC was found to perform very poorly. The new register-based SPI interface was thoroughly tested while all chip functionality was evaluated, by David Sloan. The new SPI interface was found to function as designed, and no issues were found.

While assembling the entire system chip, for the most part single handedly involved a great deal of work, the experience gained with all of the CAD tools was very valuable, and generally quite enjoyable. Taking a large, mixed signal design from HDL code and schematic, right through to final chip layout was very educational and satisfying. The effort by those in the past, especially Phillip Marshall, who enabled automated place and route using the CMOSP8G kit, and who scripted large parts of the design flow, was greatly appreciated.

5.2 PN1

PN1 was the first photodiode test chip taped-out. It contained all of the devices that were designed by previous researchers on the project, simply connected to probe pads, so the devices could be characterized without any other circuitry connected to them. Some of the devices were not intended to be photodiodes, but were simply devices that needed to be placed stand-alone on a test chip, so they could be characterized. The devices are numbered from left to right. One device between 10 and 11 was missing a number, so that one was designated PN1-10b. One device number, PN1-17 was placed on the chip, and there was no actual device present at that location.

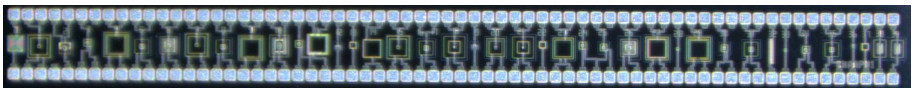


Figure 5.2: Micrograph showing ICKAAPN1, the first photodiode test chip

5.3 TC7

TC7, or Test Chip 7, was the second chip taped-out containing photodiodes. The devices that were previously designed to be APDs were shrunken, and taped-out again. Because this chip was not exclusively for photodiodes, regular bond pads were used rather than probe pads. The bond pads used for the photodiode devices were HV pads, which did not contain any ESD protection structures. It was found in the past that the ESD structures in the pads were photosensitive as well, and would have lead to erroneous results when exposed to light, when the photodiodes were being characterized. The photodiode devices are numbered 1 through 4, starting with the device in the top left corner, and proceeding down the left side of the chip.

5.4 OP1

Because the APD devices in TC7 could not be operated in Geiger mode, some adjustments to the designs were implemented, which appeared on OP1. The devices are numbered 1 through 7, starting in the upper left corner of the chip, and proceeding counter-clockwise. Two devices, which are connected to experimental sense amps, are un-numbered between TC7-4 and TC7-5. The micrograph of TC7 can be seen in Figure 5.4.

5.5 OP2

Once again, modified APD layouts were attempted on OP2. OP2 was taped-out in the same tapeout cycle as LC8, the system chip discussed in this chapter. Bare photodiode devices with probe pads were provided to be able to characterize the bare devices. Many devices, attached to experimental quench circuits, and associated digital counters were included around the bare devices. The 40V charge pump was

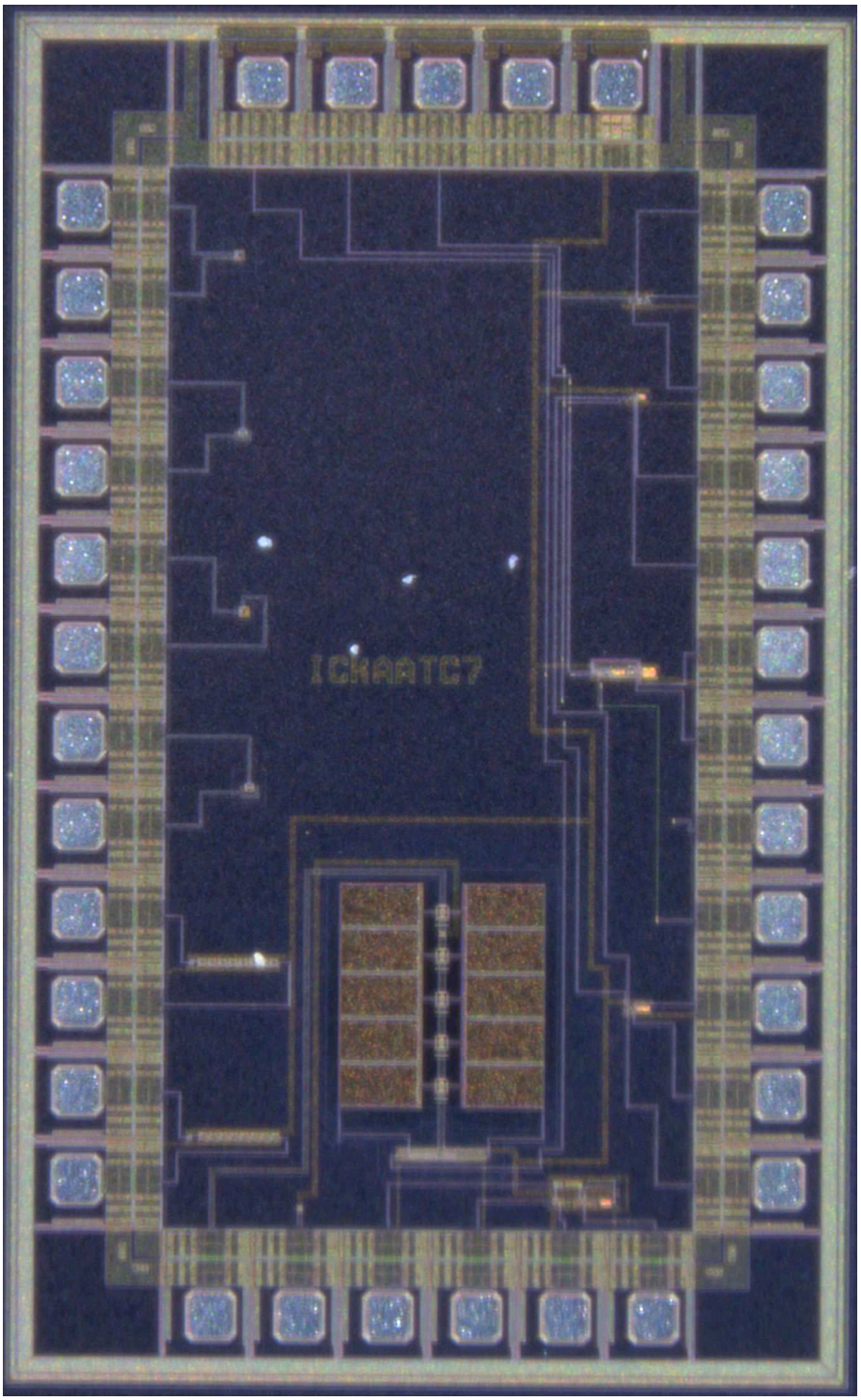


Figure 5.3: Micrograph showing ICKAATC7, the second photodiode test chip. Devices are numbered from 1 through 4 from top to bottom of left side. This chip also includes many devices which are not photodiodes.

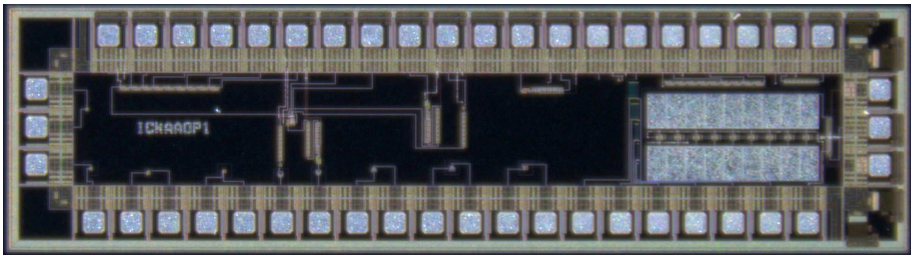


Figure 5.4: Micrograph showing ICKAAOP1, the third photodiode test chip.

also included, to provide on-chip bias generation. Because the raw devices were found to not operate in Geiger mode, none of the quench circuit connected devices were tested. The devices are numbered 1 through 20, from left to right, in the 'bare device with probe pads' region of the chip. The micrograph can be seen in Figure 5.5.

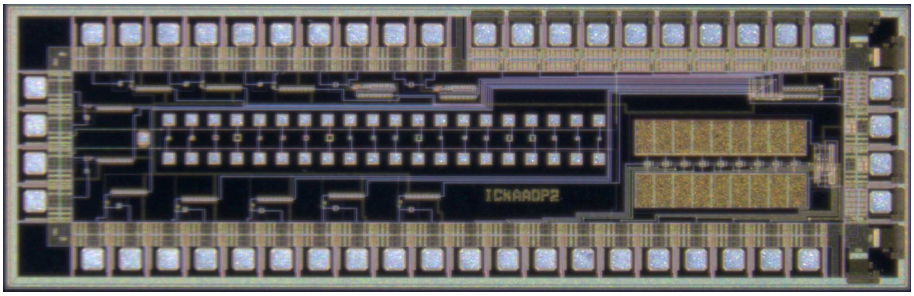


Figure 5.5: Micrograph showing ICKAAOP2, the fourth photodiode test chip.

The LC8 / OP2 tapeout also marked the first sighting of the Nano Dragon, or *Dragonus nanus*, as seen in Figure 5.6. The nano dragon's existence was first foretold in the random mumblings of bleary-eyed, sleep deprived VLSI researchers of tapeouts past, though was thought to only exist in LoC lore. The nano dragon's typical habitat includes the unused space in corner pads, and around chip identification logos. An amazing property of nano dragons, is that they are completely DRC error free! The nano dragon has been spotted on many chips since LC8, though in recent years, nano dragon numbers seem to be in decline. Through habitat rehabilitation, as well as other ongoing conservation efforts, it is hoped that nano dragon populations can recover, and the nano dragon will make a comeback in the future. Though no scientific analysis was performed, nano dragon presence seems to be correlated

with overall chip success!

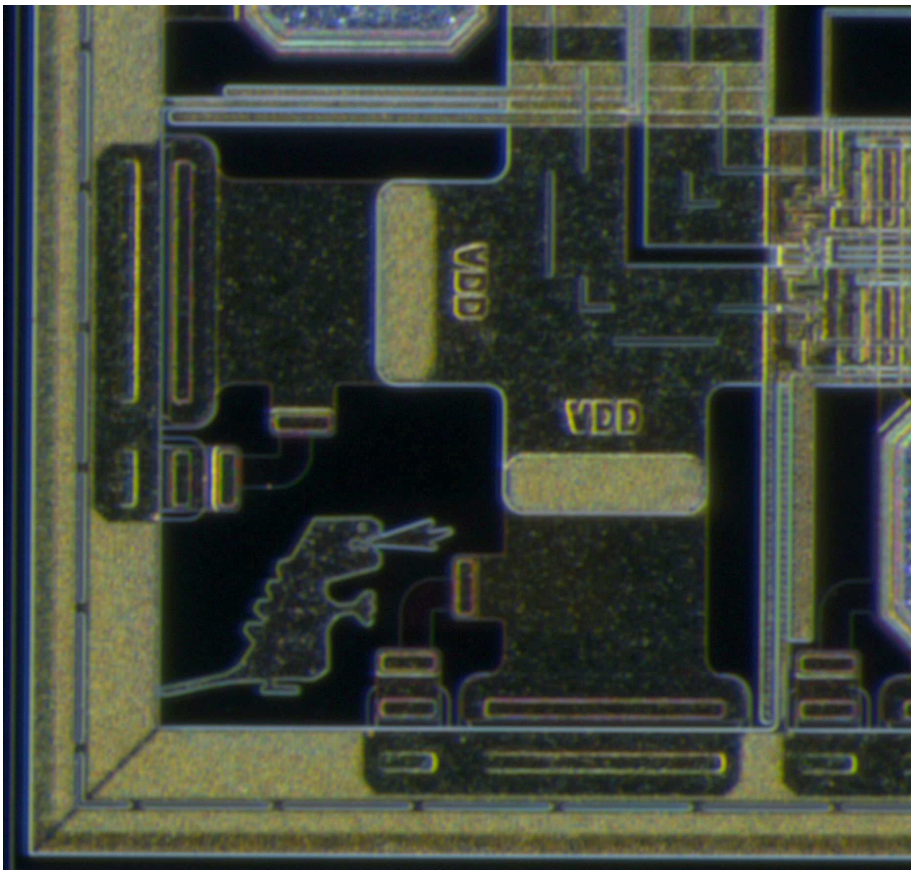


Figure 5.6: The elusive Nano Dragon, *Dragonus nanus*, seen here in one of its typical habitats, the corner pad.

5.6 OP4

The devices on OP4 are primarily a selection of *p*-*n* junctions from many combinations of the various *p* and *n* type layers available in the CMOSP8G process. These were intended to provide physical references of the breakdown voltages of various layer combinations, to evaluate device level simulation that was being attempted at the time. The simulated breakdown voltages did not match the breakdown voltages observed in previous devices that had been tested at the time. These devices also provided an interesting mix of devices, that when characterized in photovoltaic mode, showed how the responsivities of the various layer combinations compared.

Some previous APD designs were also included, with on-chip quench resistors. It was thought that perhaps stray capacitance in the probes, or long probe leads was leading to Geiger mode being unachievable. This turned out to be false, as the devices with the on-chip quench resistors still would not operate in Geiger mode. The micrograph of OP4 is shown in 5.7.

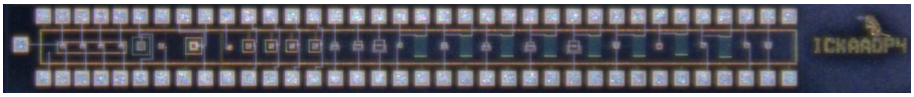


Figure 5.7: Micrograph showing ICKAAOP4, the bare-device portion of the fifth generation photodiode test chips.

Along with OP4, several devices were paired with experimental active quench circuits, should any of the devices have been found to operate in Geiger mode. During this design cycle, a separate chip, OP5 contained these structures. The micrograph of OP5 can be seen in Figure 5.8.

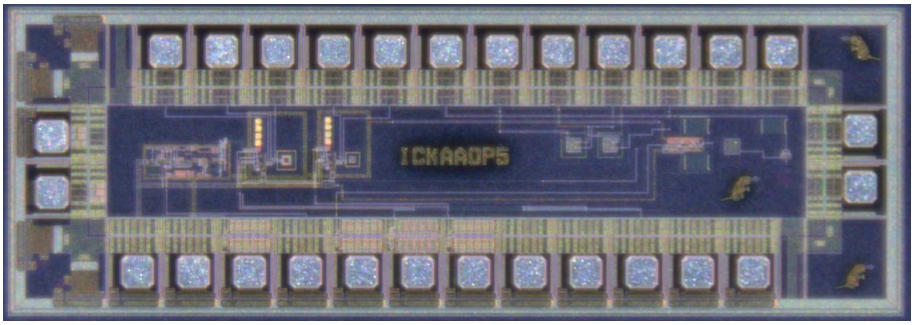


Figure 5.8: Micrograph showing ICKAAOP5, part of the fifth generation of photodiode test chips.

5.7 OP6

OP6 contained the new, pcell based APD layouts. These were the first layouts which actually functioned in Geiger mode. The bare APD layouts were arranged in a row, with probe pad connections, and are simply numbered 1 through 14 from left to right. Other various test structures were also on this chip, connected to bond pads. The micrograph of OP6 is shown in Figure 5.9.

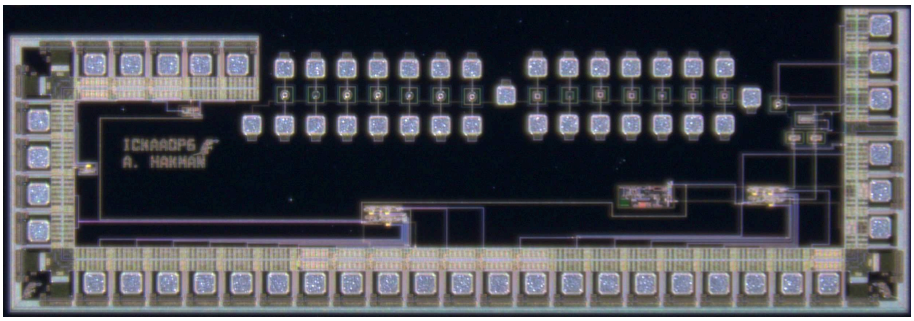


Figure 5.9: Micrograph showing ICKAAOP6, the sixth photodiode test chip, and the first one to contain devices which were observed to function in Geiger mode.

5.8 OP7

OP7 contained the same APD layouts as on OP6, but connected to the active quench circuit mentioned in Chapter 3.5.2. As mentioned in that section, the active quench circuit has not been tested or characterized. Various values of quench resistor were used, as the APDs on OP6 had not been tested at the time that the OP7 design had to be submitted. Another notable structure on this chip, is the fast counter, intended to be used with the quench circuit to count pulses. It is included in the top right corner. Probe pads, designed to be compatible with RF probes available in another lab, are intended to allow the counter to be tested for maximum frequency of operation. The micrograph is shown in Figure 5.10.

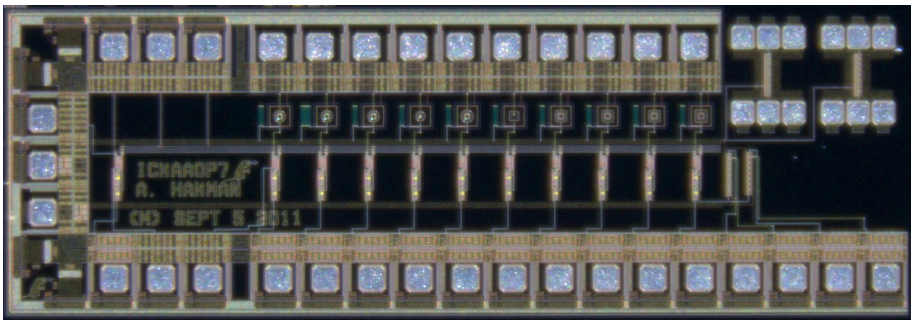


Figure 5.10: Micrograph showing ICKAAOP7, the seventh photodiode test chip. Contains same devices as OP6, but connected to active quench circuit.

5.9 Summary

All of the chips produced for this research were discussed, and micrographs presented. The device numbering used on all the photodiode test chips was mentioned, to facilitate any further testing or characterization of those devices in the future. The design flow used for producing ICKAALC8 was discussed, and all of the important subsystems listed, and physically called out in the micrograph.

Chapter 6

Contributions and Future Work

In this chapter, the contributions of this work will be presented. Guidance for the direction of future work, to continue the efforts begun, will also be discussed.

6.1 Contributions

The contributions in this thesis are easily divided into three categories, the optical detection system, and the digital control interface, and overall LoC project. Both the contributions, and future work of each will be discussed separately.

6.1.1 Overall LoC Project

ICKAALC8, a 'system chip' presented in Chapter 5, combining all of the LoC subsystems into an integrated system, including the new digital interface presented in Chapter 4 was assembled. All aspects of taking the design from individual subsystems, to a finished ASIC were executed. These included floor planning, digital synthesis, automated analog and digital place and route, manual place and route, and LVS and DRC verification of the overall design.

6.1.2 Digital Interface

As the LoC system evolved, and became more complex with more on-chip subsystems, the previous SPI controller became a hindrance. The custom opcodes used for each subsystem, with custom command format for each, became very difficult

to maintain, both in the SPI controller HDL, and the external firmware. A new controller, with a generic addressable register architecture, was designed, verified in simulation, and physically verified in silicon. Rather than a custom opcode for each subsystem, a 2-bit opcode, 6-bit address, and 8-bit data formatted command is now standard for all subsystems. Registers can be placed at any address, or group of consecutive addresses in the address space, so similar functions can be grouped together. The ability to read back the value of any register is also available now, whereas previously, the only value that could be read from the chip was the result of the ADC.

6.1.3 Optical Detection System

The primary focus of this work was the optical detection system for LoC applications. Specifically, the goal was to investigate the feasibility of integrating Geiger mode APDs with the existing system. After errors and misconceptions were identified, 3 families of functional SPAPDs were created. Two designs were produced in multiple device sizes to evaluate the effect of device area on performance. The cause of the failures of the previous devices was tracked down to issues in the layout design of the earlier devices. The other potential issues, which were presumed could have also contributed to the previous failures, were evaluated, and ruled out. These included the amount of overlap between the guard ring and the primary junction layers, the overall geometry of the device, and whether round device geometry was necessary for correct operation. Of the functional designs in the last generation, the device which used low voltage region doping layers, was found to perform the best. The low voltage device showed minimal hysteresis effects between increasing, and subsequently decreasing the incident light intensity. The hysteresis effects seen in the high voltage region devices, are suspected to be due to increased carrier trapping in those devices. Temperature effects, due to device heating during operation, were ruled out as the cause of the hysteresis seen in the results. The positive correlation between device temperature, and reverse bias breakdown threshold voltage in APDs, would lead to a decreased response as the device temperature increases,

which is the opposite of the effect observed.

In addition to the eventual success of the APD devices, other important contributions were also completed. A more suitable test apparatus, for testing and characterizing optical devices was assembled. Automated test scripts were developed, to facilitate easier device testing, and data acquisition. The scripts developed should be easily extended to adapt to any new tests that wish to be performed in the future, to further characterize the devices. Pcells were developed to significantly simplify APD layout, and to easily generate both round and square devices with little effort. Pcell-based layout of future APDs should also help to avoid hard to detect layout errors, and reduce the amount of effort required to make variants of the same design, for instance to produce the same device, but with guard rings of a different doping layer, to compare the performance of both devices.

Preliminary work was carried out to implement an active quenching circuit for the APDs, but has not yet been characterized. In addition, all of the devices built, whether Geiger functional or not, were characterized in photovoltaic mode.

6.2 Future Work

6.2.1 Optical Detection System

While it has taken a great deal of work to get to the point of functional Geiger mode devices, there is much more work that needs to be done before an optimal APD based optical detection system is ready to integrate with the rest of the LoC system.

Some devices in photovoltaic mode exhibited responsivities much greater than the theoretical maximum. This strongly points to a systematic problem with the test apparatus, the cause of which is unknown. The optical power meter, and picoammeter, should be verified to be operating correctly, at the very least by comparing measurements with an independent optical power meter and picoammeter. It would also be ideal to improve the test apparatus further, especially with respect to the non-uniform illumination observed across the field of view.

It would be interesting to determine why in 2 instances, the round devices, OP6-

3 and OP6-5, showed uncontrolled responses, once exposed to more intense light, yet the equivalent square devices, OP6-10 and OP6-12, did not. Perhaps the same effect is the cause of OP6-1 and OP6-8 showing similar response characteristics.

More devices should be built, with different combinations of doping layers. For example, because OP6-7 and OP6-14 show a response with minimal hysteresis, will the other designs show the same response simply by replacing the *hvnwell* with a *lvnwell*, or does the *pbase* versus *pwell* guard ring play a factor as well? How does the inverse device, with *n+* central region, *n-base* guard rings, sitting in *pwell* perform? How will that device compare with OP6-1 and OP6-8, which are directly on *p-substrate*?

All of the existing APD devices should be characterized on multiple wafers, and the wafer-to-wafer variation investigated. Specifically, the reverse bias threshold voltage of the same devices on multiple wafers should be evaluated, to determine how much of a factor process variation will have on the threshold voltages. It should also be possible to model the change in depletion region volume with the changing threshold voltages between wafers.

Now that the devices function properly in Geiger mode, more size variants of the devices which show good response curves should be built. A larger selection of sizes should also be built. Currently, the two devices that were built in two sizes both show remarkably similar response. This should be investigated further.

The active quench circuit should be adjusted, as recommended in Section 3.5.2, so that negative bias voltages are not required. The existing quench circuit should be characterized on OP7, using external bias circuitry. All future designs of quench circuits should be simulated in all process corners, especially at the elevated operating temperatures expected due to PCR.

A minimal APD test chip should be fabricated, which can have replaceable microfluidic channels mounted above it, as from past experience, the channels are hard to clean and re-use, to attempt fluorescence lifetime detection. This would be a very desirable goal to reach, as this would eliminate the necessity of the optical filter, to filter the excitation light from the photodetector.

6.2.2 Digital Interface

Because the new register-based SPI interface was designed to be easily modifiable and maintainable, and has been verified to function as intended, no future work should be required, other than to add or remove registers for on-chip subsystems, as the need arises.

Bibliography

[1] Elisabeth Verpoorte, "Microfluidic chips for clinical and forensic analysis," *Electrophoresis*, vol. 23, pp. 677..712, 2002.

[2] A. Manz and G. H. W.Sanders, "Chip based microsystems for genomic and proteomic analysis," *Trends Anal. Chem.*, vol. 19, pp. 364..378, 2000.

[3] Agilent Technologies, *2100 Bioanalyzer System*, November 2013, `http://www.chem.agilent.com/library/Brochures/5991-3323EN.pdf` Last accessed Oct. 26, 2014.

[4] Michael L. Chabinyc, Daniel T. Chiu, J. Cooper McDonald, Abraham D. Stroock, James F. Christian, Arieh M. Karger, and George M. Whitesides, "An Integrated Fluorescence Detection System in Poly(dimethylsiloxane) for Microfluidic Applications," *Analytical Chemistry*, vol. 73, no. 18, pp. 4491–4498, 2001.

[5] Ivan Rech, Stefano Marangoni, Angelo Gulinatti, Massimo Ghioni, and Sergio Cova, "Toward single-molecule detection with very compact DNA sequencer based on single-photon avalanche diode array.," in *Single Molecule Spectroscopy and Imaging*, Jorg Enderlein, Ed., 2008, vol. 6862.

[6] K. Mullis, F. Faloona, S. Scharf, R. Saiki, G. Horn, and H. Erlich, "Specific Enzymatic Amplification of DNA In Vitro: The Polymerase Chain Reaction," *Cold Spring Harb Symp Quant Biol*, vol. 51, pp. 263–273, 1986.

[7] F. Ponchel, C. Toomes, K. Bransfield, F.T. Leong, S.H. Douglas, S.L. Field, S.M. Bell, V. Combaret, A. Puisieux, and A.J. Mighell, "Real-time PCR based on SYBR-Green I fluorescence: An alternative to the TaqMan assay for a relative quantification of gene rearrangements, gene amplifications and micro gene deletions," *BMC Biotechnology*, vol. 3, pp. 18, 2003.

[8] A. T. Woolley and R. A. Mathies, "Ultra-high-speed DNA fragment separations using microfabricated capillary array electrophoresis chips," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 91, no. 24, pp. 11348–11352, November 1994.

[9] C.S. Effenhauser, G.J. M. Bruin, A. Paulus, and M. Ehrat, "Integrated capillary electrophoresis on flexible silicone microdevices: analysis of DNA restriction fragments and detection of single DNA molecules on microchips," *Analytical Chemistry*, vol. 69, pp. 3451..3457, 1997.

[10] B. Jung, Y. Zhu, and J.G. Santiago, "Detection of 100 attomolar Fluorophores Using a High Sensitivity On-Chip CE System and Transient Isotachophoresis," *Analytical Chemistry*, vol. 79, pp. 345..349, 2007.

[11] K. Seiler, D.J. Harrison, and A. Manz, "Planar glass chips for capillary electrophoresis: repetitive sample injection, quantitation, and separation efficiency," *Analytical Chemistry*, vol. 65, pp. 1481..1488, 1993.

[12] J. C. Jackson, D. Phelan, A. P. Morrison, R. M. Redfern, and A. Mathewson, "Characterization of Geiger Mode Avalanche Photodiodes for Fluorescence Decay Measurements," *SPIE Proceedings*, 2002.

[13] D. MacDougall and W. B. Crummett, "Guidelines for data acquisition and data quality evaluation in environmental chemistry," *Analytical Chemistry*, vol. 52, no. 14, pp. 2242–2249, 1980.

[14] Maziyar Khorasani, "Mixed Signal Design of a High Voltage (HV) CMOS Integrated Circuit for Microfluidic Applications," M.S. thesis, University of Alberta, 2008.

[15] Benjamin Russell Martin, "CMOS Instrumentation for Genetic Analysis Lab-on-a-Chip," M.S. thesis, University of Alberta, 2012.

[16] Daniel Mosconi, David Stoppa, Mattia Malfatti, Matteo Perenzoni, Mauro Scandiuzzo, and Lorenzo Gonzo, "A CMOS Sensor based on Single Photon Avalanche Diode for Fluorescence Lifetime Measurements," in *Instrumentation and Measurement Technology Conference*, April 2006, pp. 416–419.

[17] Charles Richard, Alan Renaudin, Vincent Aimez, and Paul G. Charette, "An

integrated hybrid interference and absorption filter for fluorescence detection in lab-on-a-chip devices," *Lab on Chip, The Royal Society of Chemistry*, vol. 9, pp. 1371–1376, 2009.

[18] ISS Data Tables, "Lifetime Data of Selected Fluorphores," `http://www.iss.com/resources/reference/data_tables/LifetimeDataFluorophores.html`, Last accessed Dec. 3, 2014.

[19] Marc Dandin, Akin Akturk, Babak Nouri, Neil Goldsman, and Pamela Abshire, "Characterization of Single-Photon Avalanche Diodes in a 0.5 μm Standard CMOS ProcessPart 1: Perimeter Breakdown Suppression," *IEEE Sensors Journal*, vol. 10, no. 11, pp. 1682–1690, November 2010.

[20] Alexis Rochas, *Single Photon Avalanche Diodes in CMOS Technology*, Ph.D. thesis, Universit Jean monnet, 2003.

[21] H1voltage, "Diode-IV-Curve," `http://commons.wikimedia.org/wiki/File:Diode-IV-Curve.svg`, January 2009, CC-BY-SA-3.0.

[22] "Photodiode Characteristics and Applications," `http://www.osioptoelectronics.com/application-notes/AN-Photodiode-Parameters-Characteristics.pdf`, Last accessed Dec. 4, 2014.

[23] William G. Lawrencea, Christopher Stapelsa, Frank L. Augustineb, and James F. Christiana, "Single Photon Detection using Geiger Mode CMOS Avalanche Photodiodes," in *Optoelectronic Devices: Physics, Fabrication, and Application II*, Joachim Piprek, Ed., 2005, vol. 6013.

[24] Franco Zappa, Massimo Ghioni, Sergio Cova, Carlo Samori, and Andrea Carlo Giudice, "An Integrated Active-Quenching Circuit for Single-Photon Avalanche Diodes," *IEEE Transaction on Instrumentation and Measurement*, vol. 49, no. 6, pp. 1167–1175, 2000.

[25] Hirofumi Kawazumi, Joon Myong Song, Takanori Inoue, and Teiichiro Ogawa, "Application of an avalanche photodiode in a near-Geiger operation as a fluorescence detector for capillary electrophoresis," *Journal of Chromatography A*, vol. 744, pp. 31–36, 1996.

[26] Roger Light, "Creating circle in Virtuoso," http://www.cadence.com/community/forums/p/11890/15683.aspx, March 2009, Last accessed Sept. 15, 2014.

[27] S. Cova, M. Ghioni, A. Lacaita, C. Samori, and F. Zappa, "Avalanche photodiodes and quenching circuits for single-photon detection," *Applied Optics*, vol. 35, no. 12, pp. 1956–1976, 1996.

[28] W.J. Kindt and K.-J. de Langen, "Integrated readout electronics for Geiger mode avalanche photodiodes," in *Solid-State Circuits Conference, 1998. ESSCIRC '98. Proceedings of the 24th European*, Sept 1998, pp. 216–219.

[29] Mingguo Liu, Chong Hu, Joe C. Campbell, Zhong Pan, and Mark M. Tashima, "Reduce Afterpulsing of Single Photon Avalanche Diodes Using Passive Quenching With Active Reset," *IEEE Journal of Quantum Electronics*, vol. 44, no. 5, pp. 430–434, May 2008.

[30] Naser Faramarzpour, Jamal Deen, Shahram Shirani, and Qiyin Fang, "Fully Integrated Single Photon Avalanche Diode Detector in Standard CMOS 0.18-m Technology," *IEEE Transactions On Electron Devices*, vol. 55, no. 3, pp. 760..767, Mar. 2008.

[31] GE Healthcare, https://www.gelifesciences.com/gehcls_images/GELS/Related%20Content/Files/1314787424814/litdoc28954683AD_20110831133505.pdf, *Amersham Cy5 Maleimide Mono-Reactive Dye 5-pack - Product Specification Sheet*, Last accessed Nov. 18, 2014.

[32] M. Ghioni, S. Cova, F. Zappa, and C. Samori, "Compact active quenching circuit for fast photon counting with avalanche photodiodes," *Review of Scientific Instruments*, vol. 67, no. 10, pp. 3440, October 1996.

[33] Philip Marshall, "Personal Communication between P. Marshall and A. Hakman," September 2014.

[34] Freescale Semiconductor Inc., *SPI Block Guide V04.01*, July 2004.

[35] Wesam Ahmed Al-Haddad, "High Voltage CMOS Devices and Systems for Lab-on-a-Chip Applications," M.S. thesis, University of Alberta, 2010.

[36] Sunny Ho, "VLSI design and system integration for a USB genetic amplification platform," M.S. thesis, University of Alberta, 2011.

Appendix A

Pcells for APD Layouts

A.1 On-Grid Circle and Circular Ring Pcell

Listing A.1: circlePcell.il

```
/*
circlePcell.il Roger Light 2008/01/26
originally found at http://atchoo.org/tools/cadence/circlePcell.il
via Cadence Community Forums: http://www.cadence.com/community/forums/p/11890/15683.aspx
modified by Andrew Hakman in 2012 to be able to create on-grid doughnut shapes as well as circles

Version 1.1

This file contains functions for a layout PCELL that draws a circle shape out
of polygons. This differs from the conics circle in that all points remain on
the grid specified and only lines with angles a multiple of 45 degrees are
used.

The circle is created by starting at position 0,Radius with a direction of 0
(using the direction numbers as shown below). To keep the angles smooth the
next direction can only be one of 7, 0, 1. To choose which of these to take,
the equation of a circle x^2 + y^2 = r^2 is used to determine which of the
three possible directions is closest to the actual radius.

If the number of points in the polygon is going to exceed 2048 then it is split
into multiple polygons, resulting in an appearance of a round cheese chopped
into pieces.

Direction

      5 6 7
      \ | /
    4 - O - 0
      / | \
      3 2 1

--- Using it ---

You'll need to modify the call to pcDefinePCell() below to specify your
preferred library/cell names and possibly the default radius, grid and layer.
Once you've done that just use 'load("/path/to/circlePcell.il")' in the CIW or
the libInit.il file for your library.

*/
(defun RALgetNextPoint (direction X Y grid radius)
  (let (X0 Y0 R0 D0 X1 Y1 R1 D1 X2 Y2 R2 D2 X3 Y3 R3 D3 X4 Y4 R4 D4
    X5 Y5 R5 D5 X6 Y6 R6 D6 X7 Y7 R7 D7 ret)

    (if direction == 0 || direction == 1 || direction == 7 then
      (X0 = X + grid)
      (Y0 = Y)
      (R0 = (X0**2 + Y0**2)**0.5)
      (D0 = (abs R0 - radius)))

    (if direction == 1 || direction == 2 || direction == 0 then
      (X1 = X + grid)
      (Y1 = Y - grid)
      (R1 = (X1**2 + Y1**2)**0.5)
      (D1 = (abs R1 - radius)))
```

```
(if direction == 2 || direction == 3 || direction == 1 then
  (X2 = X)
  (Y2 = Y - grid)
  (R2 = (X2**2 + Y2**2)**0.5)
  (D2 = (abs R2 - radius)))

(if direction == 3 || direction == 4 || direction == 2 then
  (X3 = X - grid)
  (Y3 = Y - grid)
  (R3 = (X3**2 + Y3**2)**0.5)
  (D3 = (abs R3 - radius)))

(if direction == 4 || direction == 5 || direction == 3 then
  (X4 = X - grid)
  (Y4 = Y)
  (R4 = (X4**2 + Y4**2)**0.5)
  (D4 = (abs R4 - radius)))

(if direction == 5 || direction == 6 || direction == 4 then
  (X5 = X - grid)
  (Y5 = Y + grid)
  (R5 = (X5**2 + Y5**2)**0.5)
  (D5 = (abs R5 - radius)))

(if direction == 6 || direction == 7 || direction == 5 then
  (X6 = X)
  (Y6 = Y + grid)
  (R6 = (X6**2 + Y6**2)**0.5)
  (D6 = (abs R6 - radius)))

(if direction == 7 || direction == 0 || direction == 6 then
  (X7 = X + grid)
  (Y7 = Y + grid)
  (R7 = (X7**2 + Y7**2)**0.5)
  (D7 = (abs R7 - radius)))

(caseq direction
       (0
        /* Possible 7 0 1 */
        (if D0 < D1 && D0 < D7 then
          /* Go D0 */
          (ret = (list X0 Y0 0))
          else
          (if D1 < D0 && D1 < D7 then
            /* Go D1 */
            (ret = (list X1 Y1 1))
            else
            /* Go D7 */
            (ret = (list X7 Y7 7))
            )
          )
        )

       (1
        /* Possible 0 1 2 */
        (if D1 < D2 && D1 < D0 then
          /* Go D1 */
          (ret = (list X1 Y1 1))
          else
          (if D2 < D1 && D2 < D0 then
            /* Go D2 */
            (ret = (list X2 Y2 2))
            else
            /* Go D0 */
            (ret = (list X0 Y0 0))
            )
          )
        )

       (2
        /* Possible 1 2 3 */
        (if D2 < D3 && D2 < D1 then
          /* Go D2 */
          (ret = (list X2 Y2 2))
          else
          (if D3 < D2 && D3 < D1 then
            /* Go D3 */
            (ret = (list X3 Y3 3))
            else
            /* Go D1 */
            (ret = (list X1 Y1 1))
            )
          )
        )

       (3
        /* Possible 2 3 4 */
        (if D3 < D4 && D3 < D2 then
          /* Go D3 */
```

```
            (ret = (list X3 Y3 3))
            else
            (if D4 < D3 && D4 < D2 then
              /* Go D4 */
              (ret = (list X4 Y4 4))
              else
              /* Go D2 */
              (ret = (list X2 Y2 2))
              )
            )
          )

          (4
           /* Possible 3 4 5 */
           (if D4 < D5 && D4 < D3 then
             /* Go D4 */
             (ret = (list X4 Y4 4))
             else
             (if D5 < D4 && D5 < D3 then
               /* Go D5 */
               (ret = (list X5 Y5 5))
               else
               /* Go D3 */
               (ret = (list X3 Y3 3))
               )
             )
           )

          (5
           /* Possible 4 5 6 */
           (if D5 < D6 && D5 < D4 then
             /* Go D5 */
             (ret = (list X5 Y5 5))
             else
             (if D6 < D5 && D6 < D4 then
               /* Go D6 */
               (ret = (list X6 Y6 6))
               else
               /* Go D4 */
               (ret = (list X4 Y4 4))
               )
             )
           )

          (6
           /* Possible 5 6 7 */
           (if D6 < D7 && D6 < D5 then
             /* Go D6 */
             (ret = (list X6 Y6 6))
             else
             (if D7 < D6 && D7 < D5 then
               /* Go D7 */
               (ret = (list X7 Y7 7))
               else
               /* Go D5 */
               (ret = (list X5 Y5 5))
               )
             )
           )

          (7
           /* Possible 6 7 0 */
           (if D7 < D0 && D7 < D6 then
             /* Go D7 */
             (ret = (list X7 Y7 7))
             else
             (if D0 < D7 && D0 < D6 then
               /* Go D0 */
               (ret = (list X0 Y0 0))
               else
               /* Go D6 */
               (ret = (list X6 Y6 6))
               )
             )
           )

        )
  ret
  ))

pcDefinePCell(list(ddGetObj("apds") "circle" "layout")
    ((Radius float 2.0) (Grid float 0.10) (Layer string "pplus1") (Doughnut boolean "FALSE") (InnerRadius →
        ↪ float 1.0))
    let(( pcParameters pcParamProp pcLayer pcPurpose polyList polyListInner
        X Y direction ret cheese points polygonidoutter polygonidinner)

    (pcParameters = ((pcCellView~>parameters)~>value))

    (pcParamProp = car(exists(prop pcParameters ((prop~>name) == "Radius"))))
```

```
(Radius = (pcParamProp~>value))

(pcParamProp = car(exists(prop pcParameters ((prop~>name) == "Grid"))))
(Grid = (pcParamProp~>value))

(pcParamProp = car(exists(prop pcParameters ((prop~>name) == "Layer"))))
(Layer = (pcParamProp~>value))

(pcParamProp = car(exists(prop pcParameters ((prop~>name) == "Doughnut"))))
(Doughnut = (pcParamProp~>value))

(dbReplaceProp pcCellView "viewSubType" "string" "maskLayoutParamCell")

(polyList = nil)

(X = 0.0)
(Y = Radius)
(direction = 0)

(pcLayer = Layer)
(pcPurpose = "drawing")

(polyList = (tconc polyList (list X Y)))
(ret = (RALgetNextPoint direction X Y Grid Radius))
(X = (car ret))
(Y = (cadr ret))
(direction = (caddr ret))
(polyList = (tconc polyList (list X Y)))
(points = 1)
(cheese = 0)
(while !( (abs X) < Grid && (abs Radius-Y) < Grid)
       (points = points + 1)
       (ret = (RALgetNextPoint direction X Y Grid Radius))
       (X = (car ret))
       (Y = (cadr ret))
       (direction = (caddr ret))
       (polyList = (tconc polyList (list X Y)))

       /* Split into multiple polygons if the number of points will exceed
        * the maximum. If this happens, the final polygon must contain an
        * additional point at 0,0 so that it closes properly. This is
        * controlled with the "cheese" variable because the output shape
        * resembles a round cheese chopped into pieces. */
       (if points == 2047 then
         (polyList = (tconc polyList (list 0.0 0.0)))
         (dbCreatePolygon pcCellView
                          (list pcLayer pcPurpose)
                          (car polyList))
         (points = 0)
         (cheese = 1)
         (polyList = (tconc nil (list X Y)))
         )
       )

(if (onep cheese) then
  (polyList = (tconc polyList (list 0.0 0.0)))
  )

polygonidoutter = (dbCreatePolygon pcCellView
                (list pcLayer pcPurpose)
                (car polyList))

(if (Doughnut == "TRUE") then
;generate list of co-ordinates for inner hole circle

    (polyListInner = nil)

        (X = 0.0)
        (Y = InnerRadius)
        (direction = 0)

        (polyListInner = (tconc polyListInner (list X Y)))
        (ret = (RALgetNextPoint direction X Y Grid InnerRadius))
        (X = (car ret))
        (Y = (cadr ret))
        (direction = (caddr ret))
        (polyListInner = (tconc polyListInner (list X Y)))
        (points = 1)
        (cheese = 0)
        (while !( (abs X) < Grid && (abs InnerRadius-Y) < Grid)
        (points = points + 1)
                (ret = (RALgetNextPoint direction X Y Grid InnerRadius))
                (X = (car ret))
                (Y = (cadr ret))
                (direction = (caddr ret))
                (polyListInner = (tconc polyListInner (list X Y)))
               )
    polygonidinner = (dbCreatePolygon pcCellView (list pcLayer pcPurpose) (car polyListInner))
```

```
        dbLayerAndNot(pcCellView (list pcLayer pcPurpose) (list polygonidoutter) (list polygonidinner) 32)
                dbDeleteObject(polygonidoutter)
        dbDeleteObject(polygonidinner)
        ;leChopShape( polygonid (car polyList) t t)
        ;(dbCreatePolygon pcCellView (list pcLayer pcPurpose) (car polyList))
    )
    t
    )
    )
```

A.2 Square and Square Ring Pcell

Listing A.2: squarePcell.il

```
/*
squarePcell.il Andrew Hakman 2012
based on circlePcell.il Roger Light 2008/01/26
originally found at http://atchoo.org/tools/cadence/circlePcell.il
via Cadence Community Forums: http://www.cadence.com/community/forums/p/11890/15683.aspx

Modified to make squares and square rings
Version 1.1

--- Using it ---
You'll need to modify the call to pcDefinePCell() below to specify your
preferred library/cell names and possibly the default radius, grid and layer.
Once you've done that just use 'load("/path/to/circlePcell.il")' in the CIW or
the libInit.il file for your library.
*/

pcDefinePCell(list(ddGetObj("apds") "square" "layout")
    ((Radius float 2.0) (Grid float 0.10) (Layer string "pplus1") (Doughnut boolean "FALSE") (InnerRadius →
        ↪ float 1.0))
    let(( pcParameters pcParamProp pcLayer pcPurpose polyList polyListInner
        X Y direction ret cheese points polygonidoutter polygonidinner)

    (pcParameters = ((pcCellView~>parameters)~>value))

    (pcParamProp = car(exists(prop pcParameters ((prop~>name) == "Radius"))))
    (Radius = (pcParamProp~>value))

    (pcParamProp = car(exists(prop pcParameters ((prop~>name) == "Grid"))))
    (Grid = (pcParamProp~>value))

    (pcParamProp = car(exists(prop pcParameters ((prop~>name) == "Layer"))))
    (Layer = (pcParamProp~>value))

    (pcParamProp = car(exists(prop pcParameters ((prop~>name) == "Doughnut"))))
    (Doughnut = (pcParamProp~>value))

    (dbReplaceProp pcCellView "viewSubType" "string" "maskLayoutParamCell")

    (X = 0.0)
    (Y = Radius)
    (direction = 0)

    (pcLayer = Layer)
    (pcPurpose = "drawing")

    polygonidoutter = (dbCreatePolygon pcCellView (list pcLayer pcPurpose) list(-Radius:-Radius -Radius:→
        ↪ Radius Radius:Radius Radius:-Radius))

    (if (Doughnut == "TRUE") then
        polygonidinner = (dbCreatePolygon pcCellView (list pcLayer pcPurpose) list(-InnerRadius:-InnerRadius →
            ↪ -InnerRadius:InnerRadius InnerRadius:InnerRadius InnerRadius:-InnerRadius) )

        dbLayerAndNot(pcCellView (list pcLayer pcPurpose) (list polygonidoutter) (list polygonidinner) 32)
                dbDeleteObject(polygonidoutter)
        dbDeleteObject(polygonidinner)
        ;leChopShape( polygonid (car polyList) t t)
        ;(dbCreatePolygon pcCellView (list pcLayer pcPurpose) (car polyList))
    )
    t
    )
    )
```

Appendix B

Automated Test Scripts

B.1 LED Calibration Script

Listing B.1: led_cal.py

```
#!/usr/bin/python

import serial
import io
import re
import matplotlib
matplotlib.use('pdf')
import matplotlib.pyplot as plt
from time import sleep
from time import clock
from struct import *
from numpy import *
from collections import namedtuple
import argparse
import signal
import sys

scope_metadata = namedtuple('scope_metadata', 'data_len, xinc, xorg, xref, yinc, yorg, yref, clipped, →
    ↪ peakcount')

def signal_handler(signal, frame):
    dac_glb.write('d0\n') #set current DAC
    setps(gpib_glb, 3, 2, 0.0, 0.5)
    initdmm(gpib_glb, 12)
    initdmm(gpib_glb, 12)
    print("reset bias and turned off led")
    sys.exit(0)

signal.signal(signal.SIGINT, signal_handler)

def num (s):
    try:
        return int(s)
    except ValueError:
        try:
            return float(s)
        except:
            return float('NaN')

def initopm(port):
  port.write('LAMBDA_a 623\n') #set wavelength to peak of LED
  port.write('UNITS_a W\n') #set units to watts
  port.write('MODE_a DCCONT\n') #continuous acuisition mode
  port.write('FILTER_a 1\n') #analog filtering
  port.write('FILTER_a 3\n') #analog and digital filtering
  sleep(0.1)
  port.write('DSBUF_a 0\n') #fixed buffer mode
  sleep(0.1)
  port.write('DSINT_a 1\n') #1ms interval for storing data to buffer
  sleep(0.1)
  port.write('DSSIZE_a 100\n') #100 data points stored

def readopm(port):
  port.write('DSE_a 1\n') #take reading
  sleep(1)
```

```
  port.write('STMEAN:a?\n') #get mean value
  result=port.readline()
  result=num(result)
  return result

def setps (gpib, addr, outp, voltage, current):
    gpib.write('++addr '+str(addr)+'\n')
    gpib.write('INST:NSEL '+str(outp)+'\n')
    gpib.write('VOLT '+str(voltage)+'\n')
    gpib.write('CURR '+str(current)+'\n')

def readhhdmm (port):
    port.write('READ?\n')
    result=port.readline()
    result=num(result)
    return result



############################################################
## MAIN ####################################################
############################################################
dac_ser = serial.Serial('/dev/ttyUSBdac', 115200, timeout=1)
opm_ser = serial.Serial('/dev/ttyS0', 38400, timeout=1)
#pam_ser = serial.Serial('/dev/ttyS0', 57600, timeout=1)
dmm_ser = serial.Serial('/dev/ttyUSBdmm', 19200, timeout=1)
gpib_ser = serial.Serial('/dev/ttyUSBgpib', 115200, timeout=1)

#for ctrl-c cleanup only!#####
dac_glb = dac_ser           ##
gpib_glb = gpib_ser         ##
##############################

initopm(opm_ser)

#turn off lightsource voltage for dark measurement
setps(gpib_ser, 3, 2, 0.0, 0.5)

print ('dac, led_current, power meter, led bias')

dac_ser.write('d0\n')
setps(gpib_ser, 3, 2, 0.0, 0.5)
sleep(0.25)
for i in range(0,10):
    led_current=readhhdmm(dmm_ser)
    opower=readopm(opm_ser)
    print('0, '+str(led_current)+', '+str(opower)+', 0.0')

for i in range(50, 120):
  setps(gpib_ser, 3, 2, i/10.0, 0.5)
  sleep(0.25)
  opower=readopm(opm_ser)
  led_current=readhhdmm(dmm_ser)
  print('0, '+str(led_current)+', '+str(opower)+', '+str(i/10.0))

for i in range (0, 2001, 100):
   dac_ser.write('d'+str(i)+'\n')
   sleep(0.25)
   if (i==1300):
    raw_input("Switch DMM to mA range")
   opower=readopm(opm_ser)
   led_current=readhhdmm(dmm_ser)
   print(str(i)+', '+str(led_current)+', '+str(opower)+', 12.0')

dac_ser.write('d0\n') #set current DAC
setps(gpib_ser, 3, 2, 0.0, 0.5)

gpib_ser.close()
dac_ser.close()
dmm_ser.close()
opm_ser.close()
```

B.2 Photovoltaic Mode Characterization Script

Listing B.2: pvmode_test.py

```
#!/usr/bin/python

import serial
import io
import re
from time import sleep

dac_ser = serial.Serial('/dev/ttyUSBdac', 115200, timeout=1)
pam_ser = serial.Serial('/dev/ttyS0', 57600, timeout=1)
```

```
dmm_ser = serial.Serial('/dev/ttyUSBdmm', 19200, timeout=1)

ser_pam = io.TextIOWrapper(io.BufferedRWPair(pam_ser, pam_ser, 1), newline = '\n', line_buffering = True)

dac_ser.write('d0\n') #turn off the current DAC
pam_ser.write('*RST\n') #reset the picoampmeter
pam_ser.write('*RST\n')
pam_ser.write('SYST:ZCH ON\n') #turn on zero check
pam_ser.write('RANG 2E-9\n') #2na range for zero check
pam_ser.write('INIT\n') #trigger reading to be used as zero correction
sleep(0.7)
pam_ser.write('SYST:ZCOR:ACQ\n') #use last reading taken as zero correct value
pam_ser.write('SYST:ZCOR ON\n') #perform zero correction
pam_ser.write('RANG:AUTO ON\n') #set range to auto
pam_ser.write('NPLC 1\n') #set integration time to one PLC
pam_ser.write('AVER ON\n') #turn on the averaging filter
pam_ser.write('AVER:TCON REP\n') #not moving average
pam_ser.write('AVER:COUN 5\n') #set the readings to average to 50
pam_ser.write('SYST:ZCH OFF\n') #turn off zerocheck to start readings

    #print data header
print 'dac, led_current, picoampmeter'
i=0;
for i in range(0, 65535, 1000):
    dac_ser.write('d{}\n'.format(i))
    sleep(0.2)
    pam_ser.write('READ?\n') #take reading
    pam_result=ser_pam.readline() #read result
    pam_result=re.findall(r"[+-]? *(?:\d+(?:\.\d*)?|\.\d+)(?:[eE][+-]?\d+)?", pam_result)
    pam_result=pam_result[0]
    if (i==16000):
        sleep(0.5) # let DMM get to new range
    dmm_ser.write('READ?\n')
    dmm_result=dmm_ser.readline()
    print 'd{}, {}, {}'.format(i, float(dmm_result), float(pam_result))
dac_ser.write('d65535\n')
pam_ser.write('SYST:LOC\n')

dac_ser.close()
pam_ser.close()
dmm_ser.close()
```

B.3 Geiger Mode Characterization Script

Listing B.3: geiger_test.py

```
#!/usr/bin/python

import serial
import io
import re
import matplotlib
matplotlib.use('pdf')
import matplotlib.pyplot as plt
from time import sleep
from time import clock
from struct import *
from numpy import *
from collections import namedtuple
import argparse
import signal
import sys

scope_metadata = namedtuple('scope_metadata', 'data_len, xinc, xorg, xref, yinc, yorg, yref, clipped, →
    ↪ peakcount')

def signal_handler(signal, frame):
    #set apd bias power supply bias back below threshold after tests
    setps(gpib_glb, 4, 2, 18.0, 0.001)
    #set dac to 0 and LED bias to 0
    dac_glb.write('d0\n') #set current DAC
    setps(gpib_glb, 3, 2, 0.0, 0.5)
    initdmm(gpib_glb, 12)
    initdmm(gpib_glb, 12)
    print("reset bias and turned off led")
    sys.exit(0)

signal.signal(signal.SIGINT, signal_handler)

def num (s):
    try:
        return int(s)
    except ValueError:
        try:
            return float(s)
```

```
        except:
            return float('NaN')

def setps (gpib, addr, outp, voltage, current):
    gpib.write('++addr '+str(addr)+'\n')
    gpib.write('INST:NSEL '+str(outp)+'\n')
    gpib.write('VOLT '+str(voltage)+'\n')
    gpib.write('CURR '+str(current)+'\n')

def initdmm (gpib, addr):
    gpib.write('++addr '+str(addr)+'\n')
    gpib.write('*RST\n')
    gpib.write('*CLS\n')
    gpib.write('*CLS\n')

def readdmm (gpib, addr):
    gpib.write('++addr '+str(addr)+'\n')
    gpib.flush()
    gpib.write('MEAS:VOLT:DC?\n')
    gpib.flush()
    gpib.flushInput()
    gpib.write('++read eoi\n')
    reading=gpib.readline()
    reading=num(reading)
    gpib.flushInput()
    while isnan(reading):
        gpib.write('MEAS:VOLT:DC?\n')
        sleep(0.25)
        gpib.write('++read eoi\n')
        reading=gpib.readline()
        reading=num(reading)

    return reading

def readhhdmm (port):
    port.write('READ?\n')
    result=port.readline()
    result=num(result)
    return result

def initscope (gpib, addr):
    gpib.write('++addr '+str(addr)+'\n') #talk to scope
    gpib.write(':SYST:HEAD OFF\n') #turn off response headers
    gpib.write(':TIM:REF CENT;RANG 2.0E-2 ;POS 0.0e0\n') #set the timebase
    gpib.write(':CHAN1:RANG 8.0e-2;OFFS 3.58e-2;INP DC;SCAL 1e-2\n') #setup channel 1
    gpib.write(':ACQ:AVER 0;BWL 1;CONF TCH;MODE RTIM;POIN 65536;SRAT 2.5e+6\n') #setup the aqcuisition
        ↪ settings
    gpib.write(':TRIG:SWE AUTO\n') #setup the trigger for auto
    gpib.write(':WAV:FORM BYTE\n') #set the data format to byte
    gpib.write(':WAV:SOUR CHAN1\n') #select data from channel 1

def scopecapture (gpib, addr):
    gpib.write('++addr '+str(addr)+'\n') #talk to scope
    gpib.write('*CLS\n') #clear status
    gpib.write(':DIG CHAN1\n') #capture data on channel 1
    gpib.write('*OPC?\n')
    gpib.write('++read eoi\n')
    complete=gpib.readline()
    complete=num(complete)
    while not complete:
        gpib.write('*OPC?\n')
        gpib.write('++read eoi\n')
        complete=gpib.readline()
        complete=num(complete)

    gpib.write(':CHAN1:DISP ON\n')

    gpib.write(':WAV:XINC?\n') #get the xincrement
    gpib.write('++read eoi\n')
    xinc=gpib.readline()
    xinc=num(xinc)
    gpib.write(':WAV:XOR?\n') #get the xorigin
    gpib.write('++read eoi\n')
    xorg=gpib.readline()
    xorg=num(xorg)
    gpib.write(':WAV:XREF?\n') #get the xreference
    gpib.write('++read eoi\n')
    xref=gpib.readline()
    xref=num(xref)
    gpib.write(':WAV:YINC?\n') #get the yincrement
    gpib.write('++read eoi\n')
    yinc=gpib.readline()
    yinc=num(yinc)
    gpib.write(':WAV:YOR?\n') #get the yorigin
    gpib.write('++read eoi\n')
    yorg=gpib.readline()
    yorg=num(yorg)
    gpib.write(':WAV:YREF?\n') #get the yreference
    gpib.write('++read eoi\n')
```

```
    yref=gpib.readline()
    yref=num(yref)
    gpib.write(':WAV:DATA?\n')
    gpib.write('++read eoi\n')
    byte = gpib.read(1)
    while byte != '#':
        byte=gpib.read(1)
    header_len = int(gpib.read(1))
    data_len = int(gpib.read(header_len))

    rawdata = fromstring(gpib.read(data_len), dtype=int8)

    timedata = arange( 0,data_len,1 )
    timedata = ((timedata - xref) * xinc) + xorg
    voltdata = (rawdata * yinc) + yorg
    gpib.read(1) #read the terminating character
    gpib.write(':WAV:CLIP?\n')
    gpib.write('++read eoi\n')
    clipped=gpib.read(1)
    clipped=num(clipped)
    gpib.flushInput()
    md = scope_metadata(data_len, xinc, xorg, xref, yinc, yorg, yref, clipped, 0)
    return md, rawdata, timedata, voltdata

def peakdetect (md, voltdata):
    peaks = zeros(md.data_len)
    peakcount=0
    deltathres=0.003 #what delta do we consider a peak
    backwindow=6 #how many points back (how slow can the peak be) to find the delta above
    fwdwindow=19 #how many points will we wait before looking for the next peak?
    for k in range(backwindow, md.data_len):
        if (voltdata[k]-voltdata[k-backwindow:k-1].min() > deltathres):
            peakcount=peakcount+1
            peaks[k:k+fwdwindow]=1
            k = k+fwdwindow
    return peakcount, peaks

def savedata(md, timedata, voltdata, rawdata, peaks, filename):
    savez(filename, md=md, timedata=timedata, voltdata=voltdata, rawdata=rawdata, peaks=peaks)

def loaddata(filename):
    npzfile = load(filename)
    return npzfile['md'], npzfile['timedata'], npzfile['voltdata'], npzfile['rawdata'], npzfile['peaks']


def makegraph(timedata, voltdata, peaks, peakcount, filename):
    fig, ax1 = plt.subplots()
    plt.ylim([-0.003, 0.045])
    plt.ticklabel_format(style='sci', axis='x', scilimits=(-2,2))
    ax1.set_autoscaley_on(False)
    ax1.set_autoscalex_on(False)
    ax1.plot(timedata[:4096], voltdata[:4096], 'b-', linewidth=0.3)
    ax1.axhline(y=mean, color='g')
    plt.ylabel('VQR [v]')
    ax2 = ax1.twinx()
    ax2.set_autoscaley_on(False)
    plt.ylim([0,1])
    ax2.fill_between(timedata[:4096], 0, peaks[:4096], color='r', alpha=0.2, linewidth=0.0)
    ax2.axes.get_yaxis().set_visible(False)
    plt.xlabel('Time [s]')
    plt.title('APD [device] Vbias='+str(i/100.0)+' Events='+str(peakcount))
    plt.savefig(filename, dpi=200)
    plt.close(fig)

def domeasurement(runs, led_bias, apd_bias, dac, dmm_ser, gpib_ser, device, quench):
    results=zeros(10)
    vmeas=zeros(10)
    lcurr=zeros(10)
    for j in range(runs):
        led_current=readhhdmm(dmm_ser)
        setps(gpib_ser, 4, 2, apd_bias, 0.001)
        sleep(0.25)
        vbiasmeas=readdmm(gpib_ser, 12)
        md, rawdata, timedata, voltdata = scopecapture(gpib_ser, 7)
        peakcount, peaks = peakdetect(md, voltdata)
        newmd = scope_metadata(md.data_len, md.xinc, md.xorg, md.xref, md.yinc, md.yorg, md.yref, md.clipped →
            ↪, peakcount)
        md=newmd
        results[j]=md.peakcount
        vmeas[j]=vbiasmeas
        lcurr[j]=led_current
        print(quench+', '+str(led_bias)+', '+str(apd_bias)+', '+str(vbiasmeas)+', '+str(dac)+', '+str(→
            ↪ led_current)+', '+str(j)+', '+str(md.peakcount)+', '+str(md.clipped)+', , , , ')
        savedata(md, timedata, voltdata, rawdata, peaks, args.device+"_qr"+quench+"_lb"+str(led_bias)+"_b"+→
            ↪ str(apd_bias)+"_d"+str(dac)+"_r"+str(j))
    print(quench+', '+str(led_bias)+', '+str(apd_bias)+', '+str(vmeas.mean())+', '+str(dac)+', '+str(lcurr.→
        ↪ mean())+', , , , '+str(results.mean())+', '+str(results.std())+', '+str(results[0:runs/2].mean()→
        ↪ )+', '+str(results[runs/2:runs].mean()))
```

```
############################################################
## MAIN ####################################################
############################################################

parser = argparse.ArgumentParser()
parser.add_argument("device", help="Device name")
parser.add_argument("quench", help="Quench Resistor Value")
parser.add_argument("--startb", help="Start bias voltage", type=float)
parser.add_argument("--stopb", help="Stop bias voltage", type=float)
args=parser.parse_args()

if args.startb:
    startv=int(args.startb*100)
else:
    startv=4450
if args.stopb:
    stopv=int(args.stopb*100)+1
else:
    stopv=startv+10


dac_ser = serial.Serial('/dev/ttyUSBdac', 115200, timeout=1)
#pam_ser = serial.Serial('/dev/ttyS0', 57600, timeout=1)
dmm_ser = serial.Serial('/dev/ttyUSBdmm', 19200, timeout=1)
gpib_ser = serial.Serial('/dev/ttyUSBgpib', 115200, timeout=1)

#for ctrl-c cleanup only!#####
dac_glb = dac_ser           ##
gpib_glb = gpib_ser         ##
##############################

initscope(gpib_ser, 7)
initdmm(gpib_ser, 12)

#turn off lightsource voltage for dark measurement
setps(gpib_ser, 3, 2, 0.0, 0.5)

print("Quench_r, led_bias, Vbias_set, Vbias_meas, dac, led_curr, run, peaks found, clipping, mean, std.dev, →
    ↪ mean_firsthalf, mean_secondhalf")
for apd_bias in range(startv, stopv, 20):
    runs=10
    dac=0
    dac_ser.write('d0\n')
    setps(gpib_ser, 3, 2, 0.0, 0.5) #set LED bias to 0v for dark measurement
    sleep(0.5)
    domeasurement(runs, 0, apd_bias/100.0, 0, dmm_ser, gpib_ser, args.device, args.quench)

    for ledbias in range (50, 69, 2):
        setps(gpib_ser, 3, 2, ledbias/10.0, 0.5) #set LED bias
        sleep(0.5)
        domeasurement(runs, ledbias/10.0, apd_bias/100.0, dac, dmm_ser, gpib_ser, args.device, args.quench)

    setps(gpib_ser, 3, 2, 12.0, 0.5) #set LED bias

    for dac in range(0, 1001, 100):
        dac_ser.write('d'+str(dac)+'\n') #set current DAC
        sleep(0.5)
        domeasurement(runs, 12.0, apd_bias/100.0, dac, dmm_ser, gpib_ser, args.device, args.quench)

    for dac in range(900, -1, -100):
        dac_ser.write('d'+str(dac)+'\n') #set current DAC
        sleep(0.5)
        domeasurement(runs, 12.0, apd_bias/100.0, dac, dmm_ser, gpib_ser, args.device, args.quench)

    for ledbias in range (68, 49, -2):
        setps(gpib_ser, 3, 2, ledbias/10.0, 0.5) #set LED bias
        sleep(0.5)
        domeasurement(runs, ledbias/10.0, apd_bias/100.0, dac, dmm_ser, gpib_ser, args.device, args.quench)
    dac=0
    dac_ser.write('d0\n')
    setps(gpib_ser, 3, 2, 0.0, 0.5) #set LED bias to 0v for dark measurement
    sleep(0.5)
    domeasurement(runs, 0, apd_bias/100.0, 0, dmm_ser, gpib_ser, args.device, args.quench)

#set apd bias power supply bias back below threshold after tests
setps(gpib_ser, 4, 2, 18.0, 0.001)
#set dac to 0 and LED bias to 0
dac_ser.write('d0\n') #set current DAC
setps(gpib_ser, 3, 2, 0.0, 0.5)

gpib_ser.close()
dac_ser.close()
dmm_ser.close()
```

Appendix C

APD Cell Information and Results

Device	Cell Name	P Layer	N Layer	Active Area (μm^2)
PN1-0	pd_nplus_ringsp-epi	P-epi square	n+ rings	5062.8
PN1-1	hvnwell-pbase	pbase and p+ ring	hvnwell	167.04
PN1-2	ngd20fd	not a photodiode		
PN1-3	HV-AVA-PN	pbase and p+ ring	hvnwell	126.04
PN1-4	hvnwell-p+-n+	p+ square	hvnwell	24025
PN1-5	diode2 (rectifier)	*not a photodiode*		
PN1-6	peg95ea	*not a photodiode*		
PN1-7	hvnwell-p+n+bac	p+ ring	hvnwell	167.04
PN1-8	diode2 (rectifier)	*not a photodiode*		
PN1-9	hvnwell-p+-n+pbase	p+ pbase ring	hvnwell	1103.04
PN1-10	NUG50FA	*not a photodiode*		
PN1-10b	HV-AVA-PN	p+ pbase ring	hvnwell	126.04
PN1-11	NBPP_m	*not a photodiode*		
PN1-12	new_apd2	p+ square pbase gr	hvnwell	100
PN1-13	photodiode_p	p+ square	lvnwell	2361.96
PN1-14	photodiode_p_mode1	p+ square	lvnwell	22081.96
PN1-15	HVnwell-pbse2	pbase rectangle	hvnwell	625
PN1-16	diode2 (rectifier)	*not a photodiode*		
PN1-17	*no device*			
PN1-18	diode (rectifier)	*not a photodiode*		
PN1-19	new_apd2	p+ square pbase gr	hvnwell	100
PN1-20	photo_transistor	p+	n+	
PN1-21	diode (rectifier)	*not a photodiode*		
PN1-22	photodiode_n	pwell	n+	2361.96
PN1-23	hvnwell-pepi	P-epi	hvnwell	23716
PN1-24	HV-AVA-PN	pbase and p+ ring	hvnwell	126.04
PN1-25	LV-AVA-PN	P-epi with pwell gr	deep nwell	88.36
PN1-26	peg95ea	*not a photodiode*		
PN1-27	LV-AVA-PN-f	p+ and pwell ring	deep nwell	1414.04
PN1-28	APD_2_10	p+ ring	hvnwell	93.16
PN1-29	PD_pplus_hvnwell	p+ ring	hvnwell	1103.04
PN1-30	LV-AVA-PN	P-epi with pwell gr	deep nwell	88.36
PN1-31	diode2 (rectifier)	*not a photodiode*		
PN1-32	diode_high_current	*not a photodiode*		
PN1-33	APD_2_10	p+ ring	hvnwell	93.16
PN1-34	HV-AVA-PN	pbase and p+ ring	hvnwell	126.04
PN1-35	diode2 (rectifier)	*not a photodiode*		
PN1-36	APD_1_10	P-epi	n+ ring	81
PN1-37	photodiode_n	pwell	n+	2361.96
PN1-38	hv_floating_Nbase_zener	*not a photodiode*		
PN1-39	mystery_diode	*not a photodiode*		

Table C.1: PN1 Device Parameters

Device	Cell Name	P Layer	N Layer	Active Area (μm^2)	Comment
TC7-1	APD_2_7u	p+ ring	Hvnwell	72.76	Active ring instead of square
TC7-2	new_apd1_7u	p+ pwell guard ring	Lvnwell	49	pwell covers entire p+ region
TC7-3	APD_1_7u	P-epi	n+ ring	63	Active ring instead of square
TC7-4	new_apd2_7u	p+ pbase guard ring	hvnwell	49	No surround of pbase on p+ – incorrect active

Table C.2: TC7 Device Parameters

Device	Cell Name	P Layer	N Layer	Active Area (μm^2)	Comment
OP1-1	new_apd1_4u	p+ pwell gr	lvnwell	81	pwell under entire area – no area with only p+
OP1-2	new_apd2_4u	p+ pbase gr	hvnwell	16	no overlap between outer edge of pbase and p+
OP1-3	APD_1_4u	P-epi	n+ ring	45	incorrect active – pbase drawn but not manufactured
OP1-4	APD_2_4u	p+ ring	hvnwell	52.36	incorrect active – nbase drawn but not manufactured
OP1-5	new_apd1_7u	p+ pwell gr	lvnwell	144	pwell under entire area – no area with only p+
OP1-6	new_apd2_7u	p+ pbase gr	hvnwell	49	no overlap between outer edge of pbase and p+
OP1-7	APD_2_7u	p+ ring	hvnwell	72.76	incorrect active – nbase drawn but not manufactured

Table C.3: OP1 Device Parameters

Device	Cell Name	P Layer	N Layer	Active Area (μm^2)	Comment
OP2-1	APD_1_4u	p-epi	n+ pbase middle	81	pbase mid is 4x4
OP2-2	APD_1_7u	p-epi	n+ pbase middle	144	pbase mid is 7x7
OP2-3	APD_1_10	p-epi	n+ pbase middle	225	pbase mid is 10x10
OP2-4	APD_1_20	p-epi	n+ pbase middle	625	pbase mid is 20x20
OP2-5	APD_2_4u	p+ nbase middle	hvnwell	88.36	nbase mid is 4x4
OP2-6	APD_2_7u	p+ nbase middle	hvnwell	153.76	nbase mid is 7x7
OP2-7	APD_2_10	p+ nbase middle	hvnwell	237.16	nbase mid is 10x10
OP2-8	APD_2_20	p+ nbase middle	hvnwell	645.16	nbase mid is 20x20
OP2-9	new_apd1_4u	p+ pwell gr	lvnwell	88.36	P+ only is 1.4x1.4
OP2-10	new_apd1_7u	p+ pwell gr	lvnwell	144	P+ only is 4x4
OP2-11	new_apd1	p+ pwell gr	lvnwell	225	P+ only is 7x7
OP2-12	new_apd1_20u	p+ pwell gr	lvnwell	625	P+ only is 7x7
OP2-13	new_apd2_4u	p+ pbase gr	hvnwell	16	gr not extended past p+
OP2-14	new_apd2_7u	p+ pbase gr	hvnwell	49	gr not extended past p+
OP2-15	new_apd2	p+ pbase gr	hvnwell	100	gr not extended past p+
OP2-16	new_apd2_20u	p+ pbase gr	hvnwell	400	gr not extended past p+
OP2-17	new_apd3_20u	p-epi	n+ nbase gr	400	gr not extended past n+
OP2-18	new_apd3	p-epi	n+ nbase gr	100	gr not extended past n+
OP2-19	new_apd3_7u	p-epi	n+ nbase gr	49	gr not extended past n+
OP2-20	new_apd3_4u	p-epi	n+ nbase gr	16	gr not extended past n+

Table C.4: OP2 Device Parameters

Device	Cell Name	P Layer	N Layer	P Active Area (μm^2)	N Active Area (μm^2)
OP4-1	lateral_pplus_nplus_diode_pwell	p+	n+	27	27
OP4-2	lateral_pplus_nplus_diode_pwell_abutted	p+	n+	24	24
OP4-3	lateral_pplus_nplus_diode_lvnwell	p+	n+	27	27
OP4-4	lateral_pplus_nplus_diode_lvnwell_abutted	p+	n+	24	24
OP4-5	Hvnwell-sub	P-epi	hvnwell		4422.25
OP4-6	nbase_psub_test	P-epi	N-base		100
OP4-7	nplus_pwell_lvnwell_sub	pwell	n+ ring		137.56
OP4-8	nplus_psub_test	P-epi	n+ square		57.76
OP4-9	pbase_hvnwell_test	P-base	hvnwell	100	
OP4-10	pplus_hvnwell_test	p+	hvnwell	100	
OP4-11	pplus_lvnwell_test	p+	lvnwell	100	
OP4-12	pwell_lvnwell_test	pwell	lvnwell	129.96	
OP4-13	new_apd2_10_no_drl_bigger_pbase	p+ pbase	hvnwell	190.44	
OP4-14	new_apd2_10_no_drl_bigger_pbase_4um_ptop	p+ pbase	hvnwell	475.24	
OP4-15	new_apd2_10_no_drl_bigger_pbase_10um_ptop	p+ pbase	hvnwell	1142.44	
OP4-16	new_apd1_7u	p+ pwell	lvnwell	144	
OP4-17	new_apd2	p+ pbase gr	hvnwell	100	
OP4-18	new_apd2_10_no_drl_bigger_pbase	p+ pbase gr	hvnwell	190.44	
OP4-19	new_apd2_10_no_drl_bigger_pbase_4um_ptop	p+ pbase ptop	hvnwell	475.24	
OP4-20	new_apd2_10_no_drl_bigger_pbase_10um_ptop	p+ pbase ptop	hvnwell	1142.44	
OP4-21	new_apd3_no_drl	P-epi	n+ nbase		100
OP4-22	APD_1_10	P-epi	n+ pbase		225
OP4-23	APD_2_7u_no_drl	p+ nbase middle	hvnwell	153.76	645.16
OP4-24	APD_2_7u_no_drl	p+ nbase middle	hvnwell	153.76	645.16
OP4-25	new_apd3_no_drl	P-epi	n+ nbase gr		100

Table C.5: OP4 Device Parameters Part 1

Device	Active no gr (μm^2)	Metal covered active (μm^2)	Comment
OP4-1		13.52	lateral p+ n+ junction inside pwell inside lvnwell, p+ n+ overalp 0.8
OP4-2		13.52	same as above, but p+ n+ don't overlap separated by 0.35
OP4-3		13.52	lateral p+ n+ junction in lvnwell, p+ n+ overlap 0.8
OP4-4		13.52	same as above, but p+ n+ don't overlap separated by 0.35
OP4-5		1086.24	hvwell covered with pfield + ptop except middle, surrounded with pbase
OP4-6		40.47	nbase with n+ contact ring
OP4-7		137.56	only n+ ring, completely covered with M1
OP4-8		37.81	
OP4-9		40.185	pbase with p+ contact ring
OP4-10		40.185	
OP4-11		40.185	
OP4-12		41.515	
OP4-13	77.44	46.9	pbase p+ overlap 1u
OP4-14	77.44	65.3	pbase p+ overlap 1u, 4u ptop around pbase
OP4-15	77.44	92.9	pbase p+ overlap 1u, 10u ptop around pbase
OP4-16	16	19.13	P+ only is 4x4, p+ total is 7x7
OP4-17	49	38.16	P+ only is 7x7, pbase gr not extended around p+
OP4-18	77.44	46.9	pbase p+ overlap 1u, pbase extends 1.5u beyond p+
OP4-19	77.44	65.3	pbase p+ ovrlp 1u, pbase extends 2.3u beyond p+, 4um ptop around pbase
OP4-20	77.44	92.9	pbase p+ overlap 1u, 10u ptop around pbase
OP4-21	49	38.16	no guard ring in effect broken active
OP4-22		90.44	pbase square is 10x10 but not contacted
OP4-23		76.32	nbase square is 7x7 but not contacted, nwell is sitting in p-epi
OP4-24		76.32	nbase square is 7x7 but not contacted, nwell is sitting in p-epi
OP4-25		38.16	N+ only (no nbase) in middle is 7x7, nbase gr not extended around n+

Table C.6: OP4 Device Parameters Part 2

Device	Cell Name	P Layer	N Layer	Active Area	Act. no gr	Comment
OP6-1	apd_n+_psub_18u_rnd	P-epi p+ cont.	n+ nbase gr	706.858	254.469	n+ nbase overlap 1u
OP6-2	apd_p+_hvnwell_8u_rnd	p+ pbase gr	hvnwell	380.132	19.634	p+ pbase overlap 0.5u
OP6-3	apd_p+_hvnwell_18u_rnd	p+ pbase gr	hvnwell	706.858	254.469	P+ pbase overlap 1u
OP6-4	apd_p+_hvnwell_abut_18u_rnd	p+ pbase gr	hvnwell	706.858	314.159	p+ pbase abutted
OP6-5	apd_p+_hvnwell_overlap_8u_rnd	p+ large pbase gr	hvnwell	380.132	19.634	p+ pbase overlap 3.5u
OP6-6	apd_p+_hvnwell_overlap_18u_rnd	p+ large pbase gr	hvnwell	907.92	254.469	p+ pbase overlap 3u
OP6-7	apd_p+_lvnwell_18u_rnd	p+ pwell gr	lvnwell	804.247	254.469	p+ pwell overlap 1u
OP6-8	apd_n+_psub_18u_sqr	P-epi p+ cont.	n+ nbase gr	900	324	n+ nbase overlap 1u
OP6-9	apd_p+_hvnwell_8u_sqr	p+ pbase gr	hvnwell	484	25	p+ pbase overlap 0.5u
OP6-10	apd_p+_hvnwell_18u_sqr	p+ pbase gr	hvnwell	900	324	p+ pbase overlap 1u
OP6-11	apd_p+_hvnwell_abut_18u_sqr	p+ pbase gr	hvnwell	900	400	p+ pbase abutted,
OP6-12	apd_p+_hvnwell_overlap_8u_sqr	p+ large pbase gr	hvnwell	484	25	p+ pbase overlap 3.5u
OP6-13	apd_p+_hvnwell_overlap_18u_sqr	p+ large pbase gr	hvnwell	1156	324	p+ pbase overlap 3u
OP6-14	apd_p+_lvnwell_18u_sqr	p+ pwell gr	lvnwell	1024	324	p+ pwell overlap 1u

Table C.7: OP6 Device Parameters

Device	Responsivity ($A W^{-1}$)	r (correlation coeff.)
pn1-0	0.674490658 5	0.994613926 7
pn1-1	82.022409336 1	0.994610150 2
pn1-3	6.820433392 2	0.994597073 2
pn1-4	0.713092175 5	0.999872981 9
pn1-7	83.559313660 8	0.994521474 9
pn1-9	21.578520564 2	0.994547621 2
pn1-10b	25.759968511 3	0.994500752 2
pn1-12	0.736566064 6	0.994588340 3
pn1-13	0.208789527 7	0.994530771 3
pn1-14	0.234133497 4	0.994662062
pn1-15	22.133626870 1	0.994515821 1
pn1-19	0.715078855 7	0.999877602 7
pn1-22	0.299912916 3	0.994603428 1
pn1-23	0.710021776 8	0.999935535 3
pn1-24	7.174645334 8	0.999925665 6
pn1-25	1.895153468 2	0.994578221 7
pn1-27	3.021801513 5	0.994569728 5
pn1-28	0.141080510 3	0.994404580 5
pn1-29	16.995214262 7	0.994473995 4
pn1-30	2.107654739 2	0.999921691 8
pn1-33	0.135576808 2	0.999860483 9
pn1-34	5.844517361 1	0.999952621
pn1-36	0.000986011 2	0.999571226 4
pn1-37	0.299266845 5	0.999924992 1

Table C.8: PN1 device responsivities, photovoltaic mode

Device	Responsivity ($A W^{-1}$)	r (correlation coeff.)
tc7-1	1103.147 840 945 6	0.999 988 625 5
tc7-2	69.966 843 298 2	0.999 993 351 5
tc7-3	16.661 569 709	0.998 759 745 3
tc7-4	30.705 392 321 2	0.999 987 771 6

Table C.9: TC7 device responsivities, photovoltaic mode

Device	Responsivity ($A W^{-1}$)	r (correlation coeff.)
op1-1	1.044 873 236 4	0.999 869 571 7
op1-2	3.521 446 285 9	0.999 936 731 8
op1-3	0.403 171 911 1	0.999 412 582 7
op1-4	15.040 311 372 3	0.999 935 664 4
op1-5	0.719 844 021 3	0.999 898 792 4
op1-6	1.498 432 675 6	0.999 886 497
op1-7	11.101 584 500 2	0.999 946 009

Table C.10: OP1 device responsivities, photovoltaic mode

Device	Responsivity ($A W^{-1}$)	r (correlation coeff.)
op2-1	2.682 465 197 9	0.999 948 338 7
op2-2	2.905 604 680 9	0.999 952 259 5
op2-3	2.401 513 462 7	0.999 947 803 5
op2-4	1.424 877 201 4	0.999 939 934 2
op2-5	9.990 016 938 2	0.999 660 856 5
op2-6	6.863 185 833 1	0.999 643 609 6
op2-7	0.358 479 817 8	0.999 930 078 9
op2-8	0.256 001 788	0.999 868 053 1
op2-9	2.569 321 115 2	0.999 943 089 5
op2-10	2.190 207 125 6	0.999 948 485
op2-11	1.734 608 853 6	0.999 952 728 4
op2-12	1.165 255 019 7	0.999 947 575 1
op2-13	5.959 021 52	0.999 915 156 2
op2-14	2.713 992 684 8	0.999 936 374 3
op2-15	1.801 875 265	0.999 937 385 1
op2-16	0.666 960 375 7	0.999 929 501 9
op2-17	3.162 311 624 1	0.999 945 577 7
op2-18	5.667 363 585 8	0.999 949 554 4
op2-19	7.440 693 382 3	0.999 946 052 9
op2-20	8.040 548 762 2	0.999 916 366 2

Table C.11: OP2 device responsivities, photovoltaic mode

Device	Responsivity ($A W^{-1}$)	r (correlation coeff.)
op4-1	lateral device	
op4-2	lateral device	
op4-3	lateral device	
op4-4	lateral device	
op4-5	9.780952185 2	0.999963574 9
op4-6	55.894564447 5	0.999963398
op4-8	56.762185098	0.999968682 8
op4-9	4.420384459	0.999963625
op4-10	2.316444229 4	0.999961158 1
op4-11	0.543063957	0.999855012 2
op4-12	16.586723886 5	0.999963343 9
op4-13	3.554617894 9	0.999964122 5
op4-14	2.818142815 5	0.999961021 4
op4-15	2.342370538 2	0.999963253 4
op4-24	0.430412141 5	0.999960685
op4-25	47.817284833 8	0.999967495 9

Table C.12: OP4 device responsivities, photovoltaic mode

Device	Responsivity ($A W^{-1}$)	r (correlation coeff.)
op6-1	9.740198782 3	0.999741904 5
op6-2	1.511294274 3	0.999928469 6
op6-3	1.099185597 5	0.999882068 5
op6-4	1.078081008 7	0.999882064 2
op6-5	1.396216288	0.999900224 5
op6-6	1.113858651 2	0.999884070 5
op6-7	3.302839059	0.999818571 8
op6-8	9.001787878 4	0.999789464 2
op6-9	1.501100202 6	0.999939477 9
op6-10	1.098733992 6	0.999892672 3
op6-11	1.073679288 8	0.999893333 9
op6-12	1.395848034 6	0.999911388
op6-13	1.122437922 9	0.999895800 5
op6-14	3.271356933 5	0.999846582 2

Table C.13: OP6 device responsivities, photovoltaic mode

Quench_r	led_bias	Vbias_set	Vbias_meas	dac	led_curr	run	peaks found	clipping	mean	std.dev
					⋮					
200k	12.0	19.92	19.87094	400	0.0011987	0	779	0		
200k	12.0	19.92	19.870754	400	0.0011986	1	728	0		
200k	12.0	19.92	19.870816	400	0.0011988	2	779	0		
200k	12.0	19.92	19.870667	400	0.001199	3	776	0		
200k	12.0	19.92	19.870791	400	0.0011989	4	712	0		
200k	12.0	19.92	19.870841	400	0.0011989	5	784	0		
200k	12.0	19.92	19.870853	400	0.001199	6	788	0		
200k	12.0	19.92	19.870803	400	0.0011989	7	769	0		
200k	12.0	19.92	19.870865	400	0.0011985	8	734	0		
200k	12.0	19.92	19.870853	400	0.0011991	9	886	0		
200k	12.0	19.92	19.8708183	400	0.00119884				773.5	45.251
200k	12.0	19.92	19.870853	500	0.0015337	0	1003	0		
200k	12.0	19.92	19.870791	500	0.0015338	1	1009	0		
					⋮					

Table C.14: Sample data produced by Geiger mode test script

Appendix D

VHDL for Register-Based SPI Interface

D.1 spi_reg_interface.vhd

This is the top level of the register based modular SPI interface. Multiple register files of various lengths are inserted into the address space.

Listing D.1: spi_reg_interface.vhd

```
----------------------------------------------------------------------------------
-- SPI <=> Register File Top Level
-- October 2009
-- Andrew Hakman
----------------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity spi_reg_interface is
    generic (
        data_width : integer := 8;
        addr_width : integer := 5);

    port (
-- spi ports
        spi_clk      : in  std_logic; -- spi clock received from master
        spi_mosi     : in  std_logic; -- master out slave in serial bitstream -MSB 1st
        spi_miso     : out std_logic; -- master in slave out -MSB 1st
        spi_miso_en  : out std_logic; -- enable signal for MISO pad
        spi_cs_b     : in  std_logic; -- chip select active low
        spi_reset_b  : in  std_logic; -- chip hardware reset active low
-- register file ports
        reg_out : out std_logic_vector ((32 * data_width)-1 downto 0);
-- adc control ports
        adc_start : out std_logic;
        adc_data_in : in std_logic_vector (data_width-1 downto 0);
        adc_data_wr_en : in std_logic_vector (1 downto 0);
-- synchronous reset
        synch_reset_b : out std_logic);
end spi_reg_interface;

architecture behave of spi_reg_interface is
  constant opc_noop  : std_logic_vector := "000";
  constant opc_read  : std_logic_vector := "010";
  constant opc_write : std_logic_vector := "100";

  signal s_spi_miso : std_logic;
  signal s_spi_miso_en : std_logic;

  signal s_adc_data_in : std_logic_vector (data_width-1 downto 0);
  signal s_adc_start : std_logic;
  signal s_adc_data_wr_en : std_logic_vector (1 downto 0);
```

```
signal rx_bit_counter : natural range 0 to 15;

signal shift_b_opc_addr : std_logic;
signal shift_b_data : std_logic;
signal shift_b_tx : std_logic;

signal rx_opc_addr_shift_en_b : std_logic;
signal rx_data_shift_en_b : std_logic;
signal tx_shift_en_b : std_logic;
signal tx_load_en_b : std_logic;

signal opc_addr_bus : std_logic_vector (data_width-1 downto 0);
signal spi_data_in_bus : std_logic_vector (data_width-1 downto 0);
signal spi_data_out_bus : std_logic_vector (data_width-1 downto 0);

signal addr_bus : std_logic_vector (4 downto 0);
signal opcode : std_logic_vector (2 downto 0);
signal reg_addr_decoded : std_logic_vector (31 downto 0);
signal reg_addr_latched : std_logic_vector (31 downto 0);
signal reg_addr_wr_select : std_logic_vector (31 downto 0);

signal decode_latch : std_logic;
signal reg_wr_en : std_logic;
signal decode_latch_with_reset : std_logic;
signal reg_wr_en_with_reset : std_logic;

type reg_out_array is array (31 downto 0) of std_logic_vector(data_width-1 downto 0);

signal reg_data_d_bus : std_logic_vector (data_width-1 downto 0);
signal reg_data_q_bus : std_logic_vector ((32 * data_width)-1 downto 0);
signal reg_data_q_read : reg_out_array;

component reg_file is
  generic (
    data_width : integer;
    reg_count : integer);

  port(
    d_in : in std_logic_vector (data_width-1 downto 0);
    direct_out : out std_logic_vector ((reg_count * data_width)-1 downto 0);
    wr_en : in std_logic_vector (reg_count-1 downto 0));
end component;

component piso_shift_reg is
  generic (data_width : integer);
  port (
  clk   : in std_logic;
  ser_out   : out std_logic;
  shift_b   : in std_logic;
  load_b    : in std_logic;
  pd_in     : in std_logic_vector (data_width-1 downto 0));
end component;

component sipo_shift_reg is
  generic (data_width : integer);
  port (
  clk   : in std_logic;
  ser_in    : in std_logic;
  shift_b   : in std_logic;
  reset_b : in std_logic;
  pd_out    : out std_logic_vector (data_width-1 downto 0));
end component;

component addr_decoder is
  port(
    addr        : in std_logic_vector (4 downto 0);
    reset_b     : in std_logic;
    decode_out : out std_logic_vector (31 downto 0)
  );
end component;



begin

  spi_miso <= s_spi_miso;
  spi_miso_en <= s_spi_miso_en;
  shift_b_opc_addr <= rx_opc_addr_shift_en_b or spi_cs_b;
  shift_b_data <= rx_data_shift_en_b or spi_cs_b;
  shift_b_tx <= tx_shift_en_b or spi_cs_b;
  addr_bus <= opc_addr_bus(4 downto 0);
  opcode <= opc_addr_bus(7 downto 5);
  reg_out <= reg_data_q_bus;
  s_adc_data_in <= adc_data_in;
  s_adc_data_wr_en <= adc_data_wr_en;
  adc_start <= s_adc_start;

  GEN_REG_DATA_Q_READ : for i in 0 to 31 generate
    reg_data_q_read(i) <= reg_data_q_bus(((i+1)*data_width)-1 downto (i*data_width));
```

```
    end generate GEN_REG_DATA_Q_READ;

    --instantiate components
    --read only "registers"
    --register 0 is "read only" - serial number
    reg_data_q_bus(7 downto 0) <= "00000001";
    --register 1 is "read only" - function mask
    reg_data_q_bus(15 downto 8) <= "11111111";

    -- 2 r/w latches, address 2 to 3 - config registers
    reg_file_config: reg_file
      generic map (data_width => data_width, reg_count => 2)
      port map (d_in => reg_data_d_bus, direct_out => reg_data_q_bus((4*data_width)-1 downto (2*data_width)) →
            ↪, wr_en => reg_addr_wr_select(3 downto 2));

    -- leave space for 2 more config registers for future use - address 4 and 5

    -- 3 r/w latches, address 6 - 8 - 6: vctrl - hv regulation control, 7 and 8: HV channel control
    reg_file_hv: reg_file
      generic map (data_width => data_width, reg_count => 3)
      port map (d_in => reg_data_d_bus, direct_out => reg_data_q_bus((9*data_width)-1 downto (6*data_width)) →
            ↪, wr_en => reg_addr_wr_select(8 downto 6));

    -- leave space for 3 more HV latches for future use - address 9 - 12

    -- 13-17 currently unallocated

    -- 2 PCR control - address 18: Heater PWM, address 19: vsubtract
    reg_file_pcr: reg_file
      generic map (data_width => data_width, reg_count => 2)
      port map (d_in => reg_data_d_bus, direct_out => reg_data_q_bus((20*data_width)-1 downto (18*data_width →
            ↪)), wr_en => reg_addr_wr_select(19 downto 18));

    -- 4 reserved - future optical - address 20 - 23

    -- 2 optical, digital IO - 24 optical, 25 digital IO
    reg_file_optics_dio: reg_file
      generic map (data_width => data_width, reg_count => 2)
      port map (d_in => reg_data_d_bus, direct_out => reg_data_q_bus((26*data_width)-1 downto (24*data_width →
            ↪)), wr_en => reg_addr_wr_select(25 downto 24));

    -- 3 reserved - future digital - address 26-28

    -- 1 latch for ADC channel select - address 29
    reg_file_adc_chan: reg_file
      generic map (data_width => data_width, reg_count => 1)
      port map (d_in => reg_data_d_bus, direct_out => reg_data_q_bus((30*data_width)-1 downto (29*data_width →
            ↪)), wr_en => reg_addr_wr_select(29 downto 29));

    -- 2 ADC result latches at end of address space on address 30 to 31 - THESE ARE CONNECTED DIFFERENTLY →
          ↪THAN THE REST
    reg_file_adc : reg_file
      generic map (data_width => data_width, reg_count => 2)
      port map (d_in => s_adc_data_in, direct_out => reg_data_q_bus((32*data_width)-1 downto (30*data_width) →
            ↪), wr_en => s_adc_data_wr_en);


    rx_opc_addr_sr: sipo_shift_reg
      generic map (data_width => data_width-1)
      port map (clk => spi_clk, ser_in => spi_mosi, shift_b => shift_b_opc_addr, pd_out => opc_addr_bus(→
            ↪data_width-1 downto 1), reset_b => spi_reset_b);

    --connect mosi to lsb of opc_addr_bus
    opc_addr_bus(0) <= spi_mosi;

    rx_data_sr: sipo_shift_reg
      generic map (data_width => data_width)
      port map (clk => spi_clk, ser_in => spi_mosi, shift_b => shift_b_data, pd_out => spi_data_in_bus, →
            ↪reset_b => spi_reset_b);

    tx_data_sr: piso_shift_reg
      generic map (data_width => data_width)
      port map (clk => spi_clk, ser_out => s_spi_miso, shift_b => shift_b_tx, load_b => tx_load_en_b, pd_in →
            ↪=> spi_data_out_bus);

    addr_decode: addr_decoder
      port map (addr => addr_bus, reset_b => spi_reset_b, decode_out => reg_addr_decoded);


  --all latches active high (as those are the parts available in the digital components library)

  --combinational logic
  reg_addr_wr_select <= reg_addr_latched when reg_wr_en_with_reset = '1' else (others => '0');
  decode_latch_with_reset <= decode_latch when spi_reset_b = '1' else '1';
  reg_wr_en_with_reset <= reg_wr_en when spi_reset_b = '1' else '1';
  reg_data_d_bus <= spi_data_in_bus;
  s_adc_start <= spi_data_in_bus(data_width-1) and reg_addr_wr_select(30);

  falling_edge_bit_counter: process (spi_clk, spi_cs_b, spi_reset_b)
```

```
  begin
    if (spi_reset_b = '0') then
      rx_bit_counter <= 0;
      rx_opc_addr_shift_en_b <= '0';
      rx_data_shift_en_b <= '1';
      s_spi_miso_en <= '0';
      decode_latch <= '0';
      reg_wr_en <= '0';
      synch_reset_b <= '1';
    else --not reset
      if (spi_cs_b = '0') then
    if falling_edge(spi_clk) then --transmit data and count clocks
      if (rx_bit_counter = 0) then
        reg_wr_en <= '0';
      end if;
      if (rx_bit_counter = 15) then -- next clock cylces will be receiving opcode and address - turn on that→
          ↪ shift register
        rx_opc_addr_shift_en_b <= '0';
        rx_data_shift_en_b <= '1';
        s_spi_miso_en <= '0';
        if (opcode = opc_write) then
          reg_wr_en <= '1';
        end if;
      end if;
      if (rx_bit_counter = 6) then --will finish receiving opcode and address on next rising edge - turn on →
          ↪decode latch
        decode_latch <= '1';
        rx_opc_addr_shift_en_b <= '1';
      end if;
      if (rx_bit_counter = 7) then -- about to start receiving and transmitting data byte
        decode_latch <= '0'; --turn off decode latch
        rx_data_shift_en_b <= '0'; -- start receiving data
        s_spi_miso_en <= '1'; -- enable miso pin
        if (reg_addr_latched(1)='1' and opcode = opc_write) then --synch reset
          synch_reset_b <= '0';
        end if;
      end if;

      --update counter
      if (rx_bit_counter < 15) then
        rx_bit_counter <= rx_bit_counter + 1;
      else
        rx_bit_counter <= 0;
      end if;
        end if; --falling edge
      else -- not reset, chip enable de-asserted
        rx_bit_counter <= 0;
        rx_opc_addr_shift_en_b <= '0';
    rx_data_shift_en_b <= '1';
    decode_latch <= '0';
    reg_wr_en <= '0';
      end if; --enabled
    end if; --reset
  end process falling_edge_bit_counter;

  rising_edge_counter: process (spi_clk, spi_cs_b, spi_reset_b, reg_addr_decoded)
  begin
    if (spi_reset_b = '0') then
      tx_shift_en_b <= '1';
      tx_load_en_b <= '1';
      reg_addr_latched <= reg_addr_decoded;
    else
      if rising_edge(spi_clk) then
    --register address decoding register
    if (decode_latch_with_reset = '1') then
      reg_addr_latched <= reg_addr_decoded;
    end if;
    if (rx_bit_counter = 7) then -- ready to setup for a read or a write
      if (opcode = opc_read) then --read
          tx_load_en_b <= '0';
          spi_data_out_bus <= reg_data_q_read(conv_integer(addr_bus));
      elsif (opcode = opc_noop) then --noop
      end if;
    end if;
    if (rx_bit_counter = 8) then-- turn off the piso shift reg load signal if we're doing a read
      tx_load_en_b <= '1';
      if (opcode = opc_read) then --read
        tx_shift_en_b <= '0'; --enable the output shift register to shift
      end if;
    end if;
    if (rx_bit_counter = 15) then -- turn off the output shift
      tx_shift_en_b <= '1';
    end if;
      end if; --rising edge
    end if; --not reset
  end process;
end architecture;
```

D.2 spi_reg_interface_tb.vhd

This is the enhanced testbench used for testing the functionality of the new register based SPI interface.

Listing D.2: spi_reg_interface_tb.vhd

```
--------------------------------------------------------------------
-- Digital Components test bench
-- This testbench was originally created and designed by Wesam Al-Haddad
-- Modified and extended by Andrew Hakman November 2009
-- Added ability to send multiple commands back to back without
-- de-asserting chip select, and the ability to interrupt commands
-- in the middle of sending.
--
-- The testbench is configured for either RTL or post-synthesis simulaiton (look at the end of the file). → For post-
-- synthesis simulation the verilog.v file (which defines the basic components such as flip-flops) followed → by the
-- gate_netlist file (either in verilog or vhdl) and this testbench are compiled and then simulated by → typing
-- vsim work.post_sim. To switch between configurations one must recompile beforehand
--
--------------------------------------------------------------------

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE STD.TEXTIO.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL; --use this or use ieee.numeric_std, cannot use both at the same time
USE IEEE.MATH_REAL.ALL;

entity spi_reg_interface_tb is
  generic (
      data_width : integer := 8;
      addr_width : integer := 5
      );

end spi_reg_interface_tb;

ARCHITECTURE Behave of spi_reg_interface_tb is

component spi_reg_interface is
    generic (
        data_width : integer;
        addr_width : integer);

    port (
-- spi ports
        spi_clk      : in  std_logic; -- spi clock received from master
        spi_mosi     : in  std_logic; -- master out slave in serial bitstream -MSB 1st
        spi_miso     : out std_logic; -- master in slave out -MSB 1st
        spi_miso_en  : out std_logic; -- enable signal for MISO pad
        spi_cs_b     : in  std_logic; -- chip select active low
        spi_reset_b  : in  std_logic; -- chip hardware reset active low
-- register file ports
        reg_out : out std_logic_vector ((32 * data_width)-1 downto 0);
-- adc control ports
        adc_start : out std_logic;
        adc_data_in : in std_logic_vector (data_width-1 downto 0);
        adc_data_wr_en : in std_logic_vector (1 downto 0));
end component;

component adc_control is
port(
    adc_clk : in std_logic;
    adc_sample : out std_logic;
    adc_eval : out std_logic;
    adc_reset_b : out std_logic;
    dac_on : out std_logic;
    reset_b : in std_logic;

    start_conversion : in std_logic;
    end_conversion : out std_logic;

    adc_sar_in : in std_logic_vector (11 downto 0);
    adc_result : out std_logic_vector (7 downto 0);
    adc_result_wr_en : out std_logic_vector (1 downto 0)
);
end component;


constant Clk_high   : time := 100 ns;
```

```
constant Clk_low     : time := 100 ns;
constant Clk_period  : time := Clk_high + Clk_low;  -- 5 MHz
constant cs_time     : time := 50 ns;

constant adc_clk_high : time := 90 ns;
constant adc_clk_low : time := 90 ns;
constant adc_clk_period : time := adc_clk_high + adc_clk_low;

signal reg_out : std_logic_vector ((32 * data_width)-1 downto 0);
signal clk : std_logic := '0';
signal mosi : std_logic := '0';
signal miso, miso_en : std_logic;
signal cs_b : std_logic := '1';
signal reset_b : std_logic := '1';

signal adc_clk : std_logic := '0';
signal s_adc_sar_in : std_logic_vector (11 downto 0) := "110011001100";
signal s_adc_result : std_logic_vector (data_width-1 downto 0);
signal s_adc_result_wr_en : std_logic_vector (1 downto 0);
signal s_adc_start_conversion : std_logic;
signal s_adc_end_conversion : std_logic;
signal s_adc_start : std_logic;
signal s_adc_sample, s_adc_eval, s_adc_reset_b, s_adc_dac_on : std_logic;

begin

SPI : spi_reg_interface
  generic map (data_width => data_width, addr_width => addr_width)
  port map ( spi_clk => clk, spi_mosi => mosi, spi_miso => miso, spi_miso_en => miso_en, spi_cs_b => cs_b, →
      ↪ spi_reset_b => reset_b, reg_out => reg_out, adc_start => s_adc_start,
            adc_data_in => s_adc_result, adc_data_wr_en => s_adc_result_wr_en);

adc_ctrl : adc_control
  port map (adc_clk => adc_clk, adc_sample => s_adc_sample, adc_eval => s_adc_eval, adc_reset_b => →
      ↪ s_adc_reset_b, dac_on => s_adc_dac_on, start_conversion => s_adc_start_conversion,
       end_conversion => s_adc_end_conversion, adc_sar_in => s_adc_sar_in, adc_result => s_adc_result, →
            ↪ adc_result_wr_en => s_adc_result_wr_en, reset_b => reset_b);

Reset_generator : process
begin
  reset_b <= '1';
  wait for 2*clk_period;
  reset_b <= '0';
   wait for 2*clk_period;
   reset_b <= '1';
   wait;
end process Reset_generator;

adc_clk_generator : process
begin
  wait for adc_clk_low;
  adc_clk <= '1';
  wait for adc_clk_high;
  adc_clk <= '0';
end process adc_clk_generator;

--tiny bit of glue logic to tie adc start signal from reg_interface to adc_controller
adc_start_sr_latch : process (s_adc_start, s_adc_end_conversion)
  begin
    if (s_adc_start = '1' and s_adc_end_conversion = '0') then
      s_adc_start_conversion <= '1';
    elsif (s_adc_start = '0' and s_adc_end_conversion = '1') then
      s_adc_start_conversion <= '0';
    end if;
end process;

-- this generates SPI MOSI data, clock, and chip select to emulate a real SPI master
-- data is read in and processed 8 bits at a time
-- if you place a 'c' after the data on a line in the opcode file, the next command will be run continuously →
    ↪ with the current command
-- i.e. no de-assertion of chip select, and continuous clock

spi_mosi_generator : process
    file infile : text;

    variable fstatus : file_open_status;
    variable bit_count : natural := 0;
    variable cur_bit : bit;
    variable continuation : character;
    variable continue : boolean := false;
    variable good_bit : boolean := false;
    variable opcode : line;
begin
    wait for 6*clk_period; -- don't start until after initial reset
    file_open(fstatus, infile, "opcodes.txt", read_mode);
    while not endfile(infile) loop
    readline(infile, opcode);
    cs_b <= '0';
    if (not continue) then
```

```
      wait for cs_time;
    end if;
    for bit_count in 0 to 7 loop
      read (opcode, cur_bit, good_bit);
      if (good_bit) then
        mosi <= To_StdULogic(cur_bit);
        wait for clk_low;
        clk <= '1';
        wait for clk_high;
        clk <= '0';
      end if;
    end loop;
    --see if we have a 'c' character now
    read(opcode, continuation, continue);
    if (continue and continuation = 'c') then
      next;
    else
      continue := false;
      wait for cs_time;
      cs_b <= '1';
      mosi <= not mosi;
      wait for 2*clk_period;
    end if;
    end loop;   -- end read file loop
    file_close(infile);
end process spi_mosi_generator;
end Behave;
```

Sample opcodes used for verification

These are some sample opcode inputs to the testbench to test various scenarios.

Listing D.3: opcodes.txt

```
01000000c -- read read only register 0
00000000
10000001c -- write register 1
00000001
10000010c -- write register 2
00000010c
10000011c -- continuously write register 3 after writing 2
00000011
01000010c -- read register 2
00000000c
01000011c -- continuously read register 3
00000000
10000100c --write register 4
00000100c
01000001c -- continuous read register 1 after write
00000000
10000101c --write register 5
00000101c
10000110  --broken write register 6 - only send opcode/addr part of word 00000110
10000111c --write register 7
00000111
10001000c
00001000
10001001c
00001001
10001010c
00001010
10001011c
00001011
10001100c
00001100
10001101c
00001101
10001110c
10101010
10001111c
01010101
00000000c
00000000
01001110c
00000000
```

D.3 reg_file.vhd

The register file block which is used to create blocks of registers. Both the number of registers in the block, as well as the width of each register is parameterized.

Listing D.4: reg_file.vhd

```
--------------------------------------------------------------------------------
-- Register File Top Level
-- October 2009
-- Andrew Hakman
--------------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;

entity reg_file is
generic (
   data_width : integer;
   reg_count : integer);

port(
    d_in : in std_logic_vector (data_width-1 downto 0);
    direct_out : out std_logic_vector ((reg_count * data_width)-1 downto 0);
    wr_en : in std_logic_vector (reg_count-1 downto 0));

end reg_file;

architecture behave of reg_file is
  component latch_n is
  generic (data_width : integer);
  port(
    din     : in std_logic_vector (data_width-1 downto 0);
    dout    : out std_logic_vector (data_width-1 downto 0);
    cs      : in std_logic
  );
  end component;

type reg_out_array is array (reg_count-1 downto 0) of std_logic_vector(data_width-1 downto 0);

signal din : std_logic_vector(data_width-1 downto 0);
signal reg_outs : reg_out_array;
signal s_wr_en : std_logic_vector (reg_count-1 downto 0);

begin
-- connect signals
    din <= d_in;
    s_wr_en <= wr_en;

    GEN_DIRECT_OUT : for i in 0 to reg_count-1 generate
        direct_out(((i+1)*data_width)-1 downto (i*data_width)) <= reg_outs(i);
    end generate GEN_DIRECT_OUT;

-- generate the registers themselves
    GEN_REGISTERS : for i in 0 to reg_count-1 generate
        regX : latch_n
            generic map (data_width => data_width)
            port map (din => din, dout => reg_outs(i), cs => wr_en(i));
    end generate GEN_REGISTERS;

end;
```

D.4 reg_addr_decode.vhd

The address decoder for the register file.

Listing D.5: reg_addr_deocde.vhd

```
--------------------------------------------------------------------------------
-- Address decoder for register file
-- October 6 2009
-- Andrew Hakman
--------------------------------------------------------------------------------

library IEEE;
```

```
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity addr_decoder is
port(
    addr       : in std_logic_vector (4 downto 0);
    reset_b    : in std_logic;
    decode_out : out std_logic_vector (31 downto 0)
);
end addr_decoder;

architecture behave of addr_decoder is
-- signals
signal s_decode_out : std_logic_vector (31 downto 0);

begin
  process (reset_b, addr)
    begin
      if (reset_b = '0') then
    s_decode_out <= (others => '1');
      else
    case addr is
      when "00000" => s_decode_out <= x"00000001";
      when "00001" => s_decode_out <= x"00000002";
      when "00010" => s_decode_out <= x"00000004";
      when "00011" => s_decode_out <= x"00000008";
      when "00100" => s_decode_out <= x"00000010";
      when "00101" => s_decode_out <= x"00000020";
      when "00110" => s_decode_out <= x"00000040";
      when "00111" => s_decode_out <= x"00000080";
      when "01000" => s_decode_out <= x"00000100";
      when "01001" => s_decode_out <= x"00000200";
      when "01010" => s_decode_out <= x"00000400";
      when "01011" => s_decode_out <= x"00000800";
      when "01100" => s_decode_out <= x"00001000";
      when "01101" => s_decode_out <= x"00002000";
      when "01110" => s_decode_out <= x"00004000";
      when "01111" => s_decode_out <= x"00008000";
      when "10000" => s_decode_out <= x"00010000";
      when "10001" => s_decode_out <= x"00020000";
      when "10010" => s_decode_out <= x"00040000";
      when "10011" => s_decode_out <= x"00080000";
      when "10100" => s_decode_out <= x"00100000";
      when "10101" => s_decode_out <= x"00200000";
      when "10110" => s_decode_out <= x"00400000";
      when "10111" => s_decode_out <= x"00800000";
      when "11000" => s_decode_out <= x"01000000";
      when "11001" => s_decode_out <= x"02000000";
      when "11010" => s_decode_out <= x"04000000";
      when "11011" => s_decode_out <= x"08000000";
      when "11100" => s_decode_out <= x"10000000";
      when "11101" => s_decode_out <= x"20000000";
      when "11110" => s_decode_out <= x"40000000";
      when "11111" => s_decode_out <= x"80000000";
      when others  => s_decode_out <= x"00000000";
    end case;
      end if;
    end process;
  decode_out <= s_decode_out;
end behave;
```

D.5 adc_control.vhd

The logic that controlls the ADC, which is in a separate clock domain from the register file.

Listing D.6: adc_control.vhd

```
--------------------------------------------------------------------------------
-- ADC Controller
-- November 15 2009
-- Andrew Hakman
--------------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;

entity adc_control is
port(
```

```
    adc_clk : in std_logic;
    adc_sample : out std_logic;
    adc_eval : out std_logic;
    adc_reset_b : out std_logic;
    dac_on : out std_logic;
    reset_b : in std_logic;

    start_conversion : in std_logic;
    end_conversion : out std_logic;

    adc_sar_in : in std_logic_vector (11 downto 0);
    adc_result : out std_logic_vector (7 downto 0);
    adc_result_wr_en : out std_logic_vector (1 downto 0)
);
end adc_control;

architecture behave of adc_control is
-- signals
signal start_f1, start_f2 : std_logic;
signal conversion_count : natural range 0 to 22;
signal s_adc_result : std_logic_vector (7 downto 0);
signal s_adc_sample, s_adc_eval, s_adc_reset_b, s_dac_on, s_end_conversion : std_logic;
signal s_adc_result_wr_en : std_logic_vector (1 downto 0);

begin
  adc_result <= s_adc_result;
  adc_sample <= s_adc_sample;
  adc_eval <= s_adc_eval;
  adc_reset_b <= s_adc_reset_b;
  dac_on <= s_dac_on;
  end_conversion <= s_end_conversion;
  adc_result_wr_en <= s_adc_result_wr_en;

  process(reset_b, adc_clk)
    begin
      if (reset_b = '0') then
    s_end_conversion <= '1';
    s_adc_reset_b <= '0';
    conversion_count <= 0;
    s_adc_eval <= '0';
    s_adc_sample <= '0';
    s_dac_on <= '0';
    s_adc_result_wr_en <= "11";
    s_adc_result <= "00000000";
    start_f1 <= '0';
    start_f2 <= '0';
      else
    if (falling_edge(adc_clk)) then
      start_f1 <= start_conversion;
      start_f2 <= start_f1;
      if (start_f2 = '1') then
        s_adc_reset_b <= '1';
        s_dac_on <= '1';
        if (conversion_count = 0) then
          s_adc_sample <= '1';
          s_adc_result <= "11110000";
          s_adc_result_wr_en <= "10";
        end if;
        if (conversion_count = 1) then
          s_adc_sample <= '0';
          s_adc_result_wr_en <= "00";
        end if;
        if (conversion_count = 2) then
          s_adc_eval <= '1';
        end if;
        if (conversion_count = 14) then
          s_adc_eval <= '0';
          s_adc_result <= adc_sar_in(7 downto 0);
          s_adc_result_wr_en <= "01";
        end if;
        if (conversion_count = 15) then
          s_adc_result_wr_en <= "00";
        end if;
        if (conversion_count = 16) then
          s_adc_result <= "1111" & adc_sar_in(11 downto 8);
          s_adc_result_wr_en <= "10";
        end if;
        if (conversion_count = 17) then
          s_adc_result <= "0000" & adc_sar_in(11 downto 8);
        end if;
        if (conversion_count = 18) then
          s_adc_result_wr_en <="00";
          s_end_conversion <= '1';
        end if;

        if (conversion_count = 22) then -- should never get here
          conversion_count <= 0;
        else
          conversion_count <= conversion_count + 1;
```

```
        end if;
      else --start_f2 = '0' - not converting
        s_end_conversion <= '0';
        s_adc_reset_b <= '0';
        conversion_count <= 0;
        s_adc_eval <= '0';
        s_adc_sample <= '0';
        s_dac_on <= '0';
        s_adc_result_wr_en <= "00";
        s_adc_result <= "00000000";
      end if;
    end if; --falling edge
      end if; --reset
  end process;
end behave;
```

D.6 latch_n.vhd

Latch with paramaterized width used to build the registers in the register file.

Listing D.7: latch_n.vhd

```
-- Nbit latch
-- October 27 2009
-- Andrew Hakman

library IEEE;
use IEEE.std_logic_1164.all;

entity latch_n is
generic (data_width : integer);
port(
    din     : in std_logic_vector (data_width-1 downto 0);
    dout    : out std_logic_vector (data_width-1 downto 0);
    cs      : in std_logic
);
end latch_n;

architecture behave of latch_n is
-- signals
signal data : std_logic_vector (data_width-1 downto 0);
begin
  process(cs, din)
    begin
        if(cs='1') then
            data <= din;
        end if;
  end process;

dout <= data;
end behave;
```

D.7 sipo_shift_register.vhd

Serial in parallel out shift register used for receiving SPI data.

Listing D.8: sipo_shift_register.vhd

```
-- clocked n-bit shift register, serial in, parallel out
-- active on rising clock edge for SPI receive
-- October 2009
-- Andrew Hakman

library IEEE;
use IEEE.std_logic_1164.all;

entity sipo_shift_reg is
    generic (data_width : integer);
```

```
    port (
        clk   : in std_logic;
        ser_in    : in std_logic;
                shift_b   : in std_logic;
        reset_b   : in std_logic;
        pd_out    : out std_logic_vector (data_width-1 downto 0));

end sipo_shift_reg;

architecture behave of sipo_shift_reg is
signal s_pd_out : std_logic_vector (data_width-1 downto 0);

begin
  pd_out <= s_pd_out;
  process (clk, reset_b)
  begin
    if (reset_b = '0') then
      s_pd_out <= (others => '0');
    else
      if (rising_edge(clk)) then
    if (shift_b = '0') then
      s_pd_out <= s_pd_out(data_width-2 downto 0) & ser_in;
    end if;
      end if;
    end if;
end process;
end architecture;
```

D.8 piso_shift_register.vhd

Parallel in serial out shift register used for transmitting SPI data.

Listing D.9: piso_shift_register.vhd

```
-------------------------------------------------------------------------------
-- clocked n-bit shift register, parallel in, serial out,
-- sensitive on falling edge of clk for SPI transmission
-- October 2009
-- Andrew Hakman
-------------------------------------------------------------------------------

library IEEE;
use IEEE.std_logic_1164.all;

entity piso_shift_reg is
    generic (data_width : integer);

    port (
        clk   : in std_logic;
        ser_out   : out std_logic;
                shift_b   : in std_logic;
        load_b    : in std_logic;
        pd_in     : in std_logic_vector (data_width-1 downto 0));
end piso_shift_reg;

architecture behave of piso_shift_reg is
signal int_data : std_logic_vector (data_width-1 downto 0);
begin
ser_out <= int_data(data_width-1);

process (clk)
    begin
        if (falling_edge(clk)) then
            if (load_b = '0') then
                int_data <= pd_in;
            end if;
            if (shift_b = '0') then
                int_data <= int_data(data_width-2 downto 0)&'0';
            end if;
        end if;
    end process;
end architecture;
```

D.9 digital_components.vhd

Very top level of the digital controller. Shows the register file connected to all other subsystems, and glue or protection logic required for some subsystems.

Listing D.10: digital_components.vhd

```
-------------------------------------------------------
-- This file describes the toplevel of the
-- digital components.
-- digitalComponents.vhd
-- HVChip Project
-- Authors: Russell Dodd, Wesam Al-Haddad, Andrew Hakman
-- last modified on November 22 2009
-------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity digital_components is
    generic (data_width : integer := 8; addr_width : integer := 5);
    port(
        spi_clk : in std_logic;      -- CLK from master
        spi_mosi : in std_logic;        -- opcode from master to CCI
        spi_miso : out std_logic;               -- opcode to master from CCI
        spi_miso_en : out std_logic;            -- signal for tristate pad from CCI
        spi_cs_b : in std_logic;        -- chip select to CCI
        spi_reset_b : in std_logic;     -- universal reset to CCI

       --clock overrides and related
        PWM_in : in std_logic;      -- for boost control, supplied PWM
        external_clk : in std_logic;    -- the external clock
            charge_pump_clk : out std_logic;                -- Charge Pump Clock
        charge_pump_clk2 : out std_logic;       -- Second charge pump clock

        current_in : inout std_logic;   -- for vco black box

        heater_pwm_b : out std_logic;           -- heater PWM, from heater control
        heater_sample : out std_logic;              -- take in the heater measure current ??
        heater_measure : out std_logic;
        heater_measure_b : out std_logic;

        boost_voltage : out std_logic_vector(7 downto 0);   -- for boost control from CCI
        boost_comparator_clk : out std_logic;       -- from boost control
        boost_pwm : out std_logic;          -- from boost control
        vmon_select : out std_logic;

        pd_reset : out std_logic;
        led_out : out std_logic_vector(2 downto 0);
        led_laser : out std_logic;

        adc_sar_compare : in std_logic;     -- input from the comparator
        adc_bin_dac    : out std_logic_vector(8 downto 0); -- goes directly to ADC_DAC
        adc_thermo_dac    : out std_logic_vector(6 downto 0); -- goes to DAC
        adc_dac_on      : out std_logic;
        adc_clk     : out std_logic;    -- need this for the analog clocked comparator
        adc_chan_sel : out std_logic_vector(3 downto 0);    -- Sample/Hold logic for analog mux from CCI
        adc_sample : out std_logic;             -- the sample and hold switch for the ADC

        hv_channels : out std_logic_vector(17 downto 0);    -- from hv control combined

                digitalout : out std_logic_vector(7 downto 0); -- Digital outputs

        vsubtract : out std_logic_vector(7 downto 0); -- DAC for amplifier

        Ireference_mode : out std_logic_vector (2 downto 0);
        Vreference_mode : out std_logic;

        hv_regul_ctrl : in std_logic
        );
end digital_components;

architecture structural of digital_components is

-- component instantiations

component thermoLogic
    port (  bits_in  : in std_logic_vector (2 downto 0);
        bits_out : out std_logic_vector (5 downto 0)
         );
end component;
```

```
component SAR
    port(   clk     : in std_logic; -- comes from adc_clk from CCI
        adc_eval : in std_logic; -- comes from ADC_on from CCI
        reset   : in std_logic; -- comes from ADC_reset
        comparator_in : in std_logic;
        sar_output : out std_logic_vector(11 downto 0)); -- goes to thermo
end component;

component vco  -- black box, analog component in digital domain
    port(   current : inout std_logic;
        vco_mode : in std_logic;
        vco_enable_b : in std_logic;
        clk_override : in std_logic;
        vco_clk : out std_logic);
end component;

component boostControl  -- done
    port(   clk : in std_logic;
        rst_b : in std_logic;
        boost_en : in std_logic;
        pwm_ext_mux : in std_logic;
        pwm_in_mux  : in std_logic;
        pwm : out std_logic;
        clk_out : out std_logic);
end component;

component heaterSwitchControl is
    port(   clk : in std_logic;
        counter : in std_logic_vector(7 downto 0);
        duty_cycle : in std_logic_vector(7 downto 0);
        pwm_b : out std_logic;
        measure : out std_logic;
        measure_b : out std_logic;
        sample : out std_logic);
end component;

component clockSystem -- done
    port(   mode : in std_logic; -- comes from config_register
        external_clk : in std_logic;
        vco_clk : in std_logic;
        clk : out std_logic_vector(14 downto 0));
end component;

component hv_control_static_combined is
    port (  hv_reset_b : in std_logic;
            hv_outputs_in : in std_logic_vector(9 downto 0);
            channels_out : out std_logic_vector(14 downto 0)
    );
end component;

component adc_control is
port(
    adc_clk : in std_logic;
    adc_sample : out std_logic;
    adc_eval : out std_logic;
    adc_reset_b : out std_logic;
    dac_on : out std_logic;
    reset_b : in std_logic;

    start_conversion : in std_logic;
    end_conversion : out std_logic;

    adc_sar_in : in std_logic_vector (11 downto 0);
    adc_result : out std_logic_vector (7 downto 0);
    adc_result_wr_en : out std_logic_vector (1 downto 0)
);
end component;

component spi_reg_interface is
    generic (
        data_width : integer := 8;
        addr_width : integer := 5);

    port (
-- spi ports
        spi_clk      : in  std_logic; -- spi clock received from master
        spi_mosi     : in  std_logic; -- master out slave in serial bitstream -MSB 1st
        spi_miso     : out std_logic; -- master in slave out -MSB 1st
        spi_miso_en  : out std_logic; -- enable signal for MISO pad
        spi_cs_b     : in  std_logic; -- chip select active low
        spi_reset_b  : in  std_logic; -- chip hardware reset active low
-- register file ports
        reg_out : out std_logic_vector ((32 * data_width)-1 downto 0);
-- adc control ports
        adc_start : out std_logic;
        adc_data_in : in std_logic_vector (data_width-1 downto 0);
        adc_data_wr_en : in std_logic_vector (1 downto 0);
-- synchronous reset
        synch_reset_b : out std_logic);
```

```
end component;

-- signal declarations

signal slv_15_clockSystem_clk : std_logic_vector(14 downto 0);

-- single wires
signal sl_clockSystem_external_clk : std_logic;
signal sl_vco_current, sl_vco_clk : std_logic;
signal sl_vco_mode, sl_clk_override : std_logic;
signal sl_hv_regul_ctrl, sl_boost_pwm_out, sl_boost_comparator_clk : std_logic;
signal sl_heater_pwm_b, sl_heater_measure, sl_heater_measure_b, sl_heater_sample : std_logic;
signal sl_sar_compare :std_logic;
signal sl_pwm_external, sl_pwm_in, sl_reset_hv : std_logic;
signal sl_adc_dac_on : std_logic;
signal sl_adc_start_conversion : std_logic;
signal sl_adc_end_conversion : std_logic;
signal sl_spi_adc_start : std_logic;
signal sl_spi_reset_b, sl_spi_cs_b, sl_spi_mosi, sl_spi_clk, sl_spi_miso, sl_spi_miso_en : std_logic;
signal sl_adc_clk, sl_adc_eval, sl_adc_reset_b : std_logic;
signal sl_reset_boost : std_logic;
signal sl_adc_sample : std_logic;
signal sl_synch_reset_b1 : std_logic;
signal sl_synch_reset_b2 : std_logic;
signal sl_system_reset_b : std_logic;
signal sl_hv_regul_with_freerun : std_logic;
signal sl_boost_en : std_logic;
signal sl_cp_clk, sl_cp_clk2 : std_logic;

-- buses
signal slv_16_hv_output : std_logic_vector(15 downto 0); -- protected version of register output
signal slv_7_thermo_dac : std_logic_vector(6 downto 0);
signal slv_6_thermo_dac : std_logic_vector(5 downto 0);
signal slv_12_sar_output : std_logic_vector (11 downto 0); -- output from SAR
signal hv_xored : std_logic_vector (7 downto 0);
signal slv_15_channels1to5 : std_logic_vector(14 downto 0);
signal slv_8_adc_result : std_logic_vector(7 downto 0);
signal slv_2_adc_result_wr_en : std_logic_vector(1 downto 0);
signal slv_256_reg_outs : std_logic_vector((8*32)-1 downto 0);
signal slv_3_ireferencemode_out : std_logic_vector (2 downto 0);

type reg_out_array is array (31 downto 0) of std_logic_vector(data_width-1 downto 0);
signal slv_reg_outs_by_reg : reg_out_array;

begin
--separate out the register bus by register - makes it easier to connect it to things
  GEN_SLV_REG_OUTS_BY_REG : for i in 0 to 31 generate
    slv_reg_outs_by_reg(i) <= slv_256_reg_outs(((i+1)*data_width)-1 downto (i*data_width));
  end generate GEN_SLV_REG_OUTS_BY_REG;


-- no straight to pin connections, everything goes through signals!!
-- inputs (there are not 14 of them)
sl_spi_clk <= spi_clk;--in
sl_spi_mosi <= spi_mosi;--in
sl_spi_cs_b <= spi_cs_b;--in
sl_spi_reset_b <= spi_reset_b; --in
sl_pwm_external <= slv_reg_outs_by_reg(2)(4);
sl_pwm_in <= PWM_in;--in
sl_vco_current <= current_in;--inout
sl_clk_override <= not (slv_reg_outs_by_reg(2)(2));
sl_vco_mode <= slv_reg_outs_by_reg(2)(1);
sl_clockSystem_external_clk <= external_clk;--in
sl_hv_regul_ctrl <= hv_regul_ctrl;--in
sl_sar_compare <= adc_sar_compare;


sl_reset_boost <= sl_system_reset_b;
sl_reset_hv <= sl_system_reset_b;

-- outputs, single wires
spi_miso <= sl_spi_miso; --out
spi_miso_en <= sl_spi_miso_en; -- out
heater_pwm_b <= sl_heater_pwm_b;--out
heater_measure <= sl_heater_measure;--out
heater_measure_b <= sl_heater_measure_b;
heater_sample <= sl_heater_sample;
boost_comparator_clk <= sl_boost_comparator_clk;
pd_reset <= slv_reg_outs_by_reg(24)(4);
boost_pwm <= sl_boost_pwm_out;  --out
adc_dac_on <= sl_adc_dac_on;
adc_clk <= sl_adc_clk;
sl_adc_clk <= slv_15_clockSystem_clk(1);
led_laser <= slv_reg_outs_by_reg(24)(3);
charge_pump_clk <= sl_cp_clk;
charge_pump_clk2 <= sl_cp_clk2;
adc_sample <= sl_adc_sample;
Vreference_mode <= not (slv_reg_outs_by_reg(3)(0));
Ireference_mode <= slv_3_ireferencemode_out;
```

```
vmon_select <= slv_reg_outs_by_reg(2)(3);

--outputs, buses
boost_voltage <= slv_reg_outs_by_reg(6);--out
adc_chan_sel <= slv_reg_outs_by_reg(29)(3 downto 0);--out
adc_bin_dac <= slv_12_sar_output(8 downto 0);
adc_thermo_dac <= slv_7_thermo_dac(6 downto 0);
led_out <= slv_reg_outs_by_reg(24)(2 downto 0);
digitalout <= slv_reg_outs_by_reg(25);

vsubtract <= slv_reg_outs_by_reg(19);

-- outputs from the hv_control_combined channels
hv_channels(14 downto 0) <=  slv_15_channels1to5;
-- the single channels
hv_channels(15) <= slv_16_hv_output(10);
hv_channels(16) <= slv_16_hv_output(12);
hv_channels(17) <= slv_16_hv_output(14);

-- channels 11, 13, 15 are not connected

vosc : vco
port map(    current => sl_vco_current,
        vco_mode => sl_vco_mode,
        clk_override => sl_clk_override,
        vco_clk => sl_vco_clk,
        vco_enable_b => slv_reg_outs_by_reg(2)(0));

bc : boostControl
port map(    clk => slv_15_clockSystem_clk(10),
        rst_b => sl_reset_boost,
        boost_en => sl_boost_en,
        pwm_ext_mux => sl_pwm_external,
        pwm_in_mux  => sl_pwm_in,
        pwm => sl_boost_pwm_out,
        clk_out => sl_boost_comparator_clk);

thl : thermoLogic
port map (   bits_in => slv_12_sar_output(11 downto 9),
        bits_out => slv_6_thermo_dac
     );
-- new thermoLogic mapping to please synopsys
slv_7_thermo_dac <= slv_6_thermo_dac(5 downto 3) & slv_12_sar_output(11) & slv_6_thermo_dac(2 downto 0);

sarb : SAR
port map (   clk => sl_adc_clk,
        adc_eval => sl_adc_eval,
        reset => sl_adc_reset_b,
        comparator_in => sl_sar_compare,
        sar_output => slv_12_sar_output -- output 9 bits goes to binary_dac and 3 goes to thermologic
     );

hsc : heaterSwitchControl
port map(    clk => slv_15_clockSystem_clk(0),
        counter => slv_15_clockSystem_clk(13 downto 6),
        duty_cycle => slv_reg_outs_by_reg(18),
        pwm_b => sl_heater_pwm_b,
        measure => sl_heater_measure,
        measure_b => sl_heater_measure_b,
        sample => sl_heater_sample);

cs : clockSystem
port map(    mode => sl_clk_override,
        external_clk => sl_clockSystem_external_clk,
        vco_clk => sl_vco_clk,
        clk => slv_15_clockSystem_clk);

hv : hv_control_static_combined
port map(    hv_reset_b => sl_reset_hv,
        hv_outputs_in => slv_16_hv_output(9 downto 0),
    --  clk_in => slv_15_clockSystem_clk,
        channels_out => slv_15_channels1to5
    );

adc_ctrl : adc_control
port map( adc_clk => sl_adc_clk, --5Mhz clock from VCO divider
      adc_sample => sl_adc_sample,
      adc_eval => sl_adc_eval,
      adc_reset_b => sl_adc_reset_b,
          dac_on  => sl_adc_dac_on,
      reset_b => sl_system_reset_b,
      start_conversion => sl_adc_start_conversion,
      end_conversion => sl_adc_end_conversion,
      adc_sar_in => slv_12_sar_output,
      adc_result => slv_8_adc_result,
      adc_result_wr_en => slv_2_adc_result_wr_en
    );

spi_reg_int : spi_reg_interface
```

```
generic map ( data_width => data_width, addr_width => addr_width)
port map (
-- spi ports
        spi_clk => sl_spi_clk,
        spi_mosi => sl_spi_mosi,
        spi_miso => sl_spi_miso,
        spi_miso_en => sl_spi_miso_en,
        spi_cs_b => sl_spi_cs_b,
        spi_reset_b => sl_system_reset_b,
-- register file ports
        reg_out => slv_256_reg_outs,
-- adc control ports
        adc_start => sl_spi_adc_start,
        adc_data_in => slv_8_adc_result,
        adc_data_wr_en => slv_2_adc_result_wr_en,
        synch_reset_b => sl_synch_reset_b1
      );

-- glue logic to tie adc start signal from reg_interface to adc_controller
adc_start_sr_latch : process (sl_spi_adc_start, sl_adc_end_conversion)
  begin
    if (sl_spi_adc_start = '1' and sl_adc_end_conversion = '0') then
      sl_adc_start_conversion <= '1';
    elsif (sl_spi_adc_start = '0' and sl_adc_end_conversion = '1') then
      sl_adc_start_conversion <= '0';
    end if;
end process;

-- glue logic to decode ireferenceout
with slv_reg_outs_by_reg(3)(2 downto 1) select slv_3_ireferencemode_out <=
"001" when "00",
"010" when "01",
"100" when "10",
"000" when others;

--glue logic to protect HV channels
--raw register out on registers 7 and 8
--protected version goes to slv_16_hv_output
--each hv channel has 2 bits of control
--xor each 2 bits together (to make sure they are both not 1)
--then and the appropriate xored bit with each bit
--this way both control bits of any channel can never both be 1 at the same time
GEN_HV_XORs_1 : for i in 0 to 3 generate
    hv_xored(i) <= slv_reg_outs_by_reg(7)(2*i) xor slv_reg_outs_by_reg(7)((2*i)+1);
    slv_16_hv_output(2*i) <= slv_reg_outs_by_reg(7)(2*i) and hv_xored(i);
    slv_16_hv_output((2*i)+1) <= slv_reg_outs_by_reg(7)((2*i)+1) and hv_xored(i);
end generate GEN_HV_XORs_1;
GEN_HV_XORs_2 : for i in 4 to 7 generate
    hv_xored(i) <= slv_reg_outs_by_reg(8)((2*i)-8) xor slv_reg_outs_by_reg(8)(((2*i)+1)-8);
    slv_16_hv_output(2*i) <= slv_reg_outs_by_reg(8)((2*i)-8) and hv_xored(i);
    slv_16_hv_output((2*i)+1) <= slv_reg_outs_by_reg(8)(((2*i)+1)-8) and hv_xored(i);
end generate GEN_HV_XORs_2;

--synchronous reset register
synch_reset_register : process (sl_synch_reset_b1, spi_clk, spi_reset_b)
  begin
    if (spi_reset_b = '0') then
      sl_synch_reset_b2 <= '1';
    elsif (rising_edge(spi_clk)) then
      sl_synch_reset_b2 <= sl_synch_reset_b1;
    end if;
end process;
sl_system_reset_b <= sl_spi_reset_b and sl_synch_reset_b2;

--glue logic for charge pump clocks and boost enable
sl_hv_regul_with_freerun <= sl_hv_regul_ctrl or slv_reg_outs_by_reg(3)(6);
sl_boost_en <= sl_hv_regul_with_freerun and slv_reg_outs_by_reg(3)(3);
sl_cp_clk <= sl_hv_regul_with_freerun and slv_reg_outs_by_reg(3)(4) and slv_15_clockSystem_clk(0);
sl_cp_clk2 <= sl_hv_regul_with_freerun and slv_reg_outs_by_reg(3)(5) and slv_15_clockSystem_clk(0);

end structural;
```