UNIVERSITY OF OSLO
Department of Informatics

UWB impulse radar in 90 nm CMOS

Master thesis

Håkon A. Hjortland

August 7, 2006

Abstract

Recently, the FCC has released a very wide unlicensed spectrum allocation from 3.1 to 10.6 GHz. The allowed emission in this spectrum is very low, so the use of this spectrum allocation limits itself to relatively short range applications. The fact that CMOS technology now reaches speeds of tens of GHz, opens up a whole new area of interesting possibilities to create cheap and wide band radio technology. One application here, is short-range radar. Thanks to the wide spectrum allocation the radar is able to send impulses rather than bursts of a carrier wave. This makes processing of the received signal much easier than the matched filters which are required in the carrier wave burst case.

In this master thesis we present two related sampling techniques for radar applications which use mostly digital circuitry and which can achieve high sampling rates. We have called these circuits, which are partially based on the Suprathreshold Stochastic Resonance (SSR) principle, swept threshold and stochastic resonance samplers. Although the samplers are mostly digital, which makes them perfect for CMOS, they are not clocked. We discover that for the case where the input signal contains much noise, typical for radars, the crudeness of these simple samplers does actually not have a very detrimental effect on the signal processing. A radar implementation in 90 nm CMOS using these samplers is presented, which is shown to reach sampling rates of about 23 GHz.

Contents

List of Figures

Acronyms

AD	Analog to Digital
ADC	Analog to Digital Converter
AWGN	Additive White Gaussian Noise
CDF	Cumulative Distribution Function
CMOS	Complementary Metal-Oxide Semiconductor
CPU	Central Processing Unit
CTQA	Continuous-Time Quantized Amplitude (term introduced in this thesis)
DAC	Digital to Analog Converter
EIRP	Effective Isotropically Radiated Power
FCC	Federal Communications Commission
FFT	Fast Fourier Transform
LNA	Low Noise Amplifier
MES	Microelectronic Systems
MIR	Micropower Impulse Radar
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
NMOS	N-channel MOSFET
PDF	Probability Density Function
PMOS	P-channel MOSFET
PPM	Pulse-Position Modulation
PWM	Pulse-Width Modulation
radar	RAdio Detection And Ranging

RFID	Radio Frequency Identification
RMS	Root Mean Square
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface Bus
SSR	Suprathreshold Stochastic Resonance
UWB-IR	Ultra-WideBand Impulse Radio
UWB	Ultra-WideBand

Acknowledgements

First of all I want to thank my supervisor Tor Sverre Lande for accepting me as his student and for the very interesting and invaluable discussions we have had. I also want to thank him for his great ability to motivate and inspire and for his wide insight into the field. Great thanks also goes to my co-supervisor, Dag T. Wisland.

I also want to thank Kjetil Meisal and Claus Limbodal for great discussions and for being the first master students at our research group to work with UWB, thus paving the way for the rest of us and quickly getting us up to speed with the UWB basics. Big thanks for their contributions to the radar implementation, namely the LNA, made by Kjetil, the thresholder, which Claus gave me invaluable help me with, and the pre-thresholder level shifter, by Kjetil and Claus. A huge thank you goes to Novelda for sponsoring this work.

I also want to give big thanks to Håvard Moen for being a great friend throughout our studies at the university, and for being a good discussion partner. I especially want to thank him for his help with various pieces of software, like teaching me the ruby programming language, showing me how to use the gschem and pcb software for PCB design, guiding me through the LaTeX jungle and much more. Big thanks also for his pulse generator without which there would be no radar.

Thanks also to Håvard Kolle Riis for practical help and especially the good help with the chip tape-out.

I also want to thank the entire Microelectronic Systems (MES) research group for being such a great environment for both social and academic activity. I want to thank everybody who have helped me through interesting discussions.

As a last note of appreciation, I want to thank the free open source software community for providing great, free (as in freedom, not gratis), software. To mention a few, both ruby, octave, gschem, pcb, LaTeX and xfig have been invaluable in my work with this project.

1. Introduction

The Federal Communications Commission (FCC) has just released a wide spectrum allocation for unlicensed use. The spectrum stretches from 3.1–10.6 GHz, giving an incredible bandwidth of 7.5 GHz. Thanks to advances in CMOS technology, we are now able to create cheap solution is this technology that are able to reach the high frequencies of the spectrum allocation band.

Possibilities for many new applications thus opens up. Communication, tracking, Radio Frequency Identification (RFID) and radar are perhaps the main fields of interest. In this master thesis we will present a UWB radar in CMOS at frequencies approximating the 3.1–10.6 GHz band. The radar sampler is implemented using an unconvential technique, where most of the signal processing is done using digital circuit elements, although unclocked. This sampling technique relates to Suprathreshold Stochastic Resonance (SSR), which is a relatively newly identified phenomenon which is currently being explored, sparked off by [Stoc 00].

1.1. Medical application for UWB impulse radar

One of the many possible applications for Ultra-WideBand Impulse Radio (UWB-IR) radar is medical imaging. With pulse durations of about 150 ps, the length of the pulse when propagating through vacuum is about 4.5 cm. When propagating through tissue, the speed and thus the length of the pulse decreases, and when also taking into account the fact that the pulse has to travel both to the object and back again, we end up with a resolution which is probably adequate for many medical imaging purposes.

1.2. Outline of thesis

In chapter 2 we will describe the UWB spectrum allocation and possible pulses which can be used within the constraints of the spectrum mask.
In chapter 3 we will describe some general radar background.
In chapter 4 we coin the term Continuous-Time Quantized Amplitude (CTQA), which will be used to describe such signals and signal processing. We try to explore the properties of this signal processing domain.
In chapter 5 we describe a few different radar samplers and present simulations of the sampling methods used in the radar which this master thesis revolves around.

In chapter 6 we describe the radar circuit that has been implemented and present measurements made on the circuit.
In chapter 7 we conclude our observations.

2. UWB

The FCC has recently allocated a very wide unlicensed spectrum in the 3.1–10.6 GHz range. Allowed power transmission in this range is -41.3 dBm/MHz EIRP. Effective Isotropically Radiated Power (EIRP) means that the antenna must transmit at a power level so that an observer some distance away which assumes the antenna is isotropic, will think the antenna is transmitting at that given power. This specification means that if you increase your antenna gain by for instance a factor 100, i.e. focus your transmitted energy, you will also have to reduce the power by a factor 100. Otherwise, observers located in your beam will think you just increased your transmission power 100-fold. dBm is a logarithmic unit where 0 dBm $=$ 1 mW.

The regulation also allows for some weaker signals to be transmitted above and below the main spectrum mask. This makes it easier to create compliant signals, since the cutoffs of the signals do not need to be so steep then. Different conditions apply for handheld equipment and equipment to be used indoors.

One can use multiple wide band carrier based transmissions in order to fill the spectrum, but another way is to send pulses which fill the whole spectrum by them selves. A pulse shape that works very well here is the Gaussian pulse [Siwi 04]. The wider the pulse, the more narrow the bandwidth it uses. And the shorter the pulse, the wider the bandwidth will be. Since the Gaussian pulse has a center frequency of 0 Hz, we will have to multiply it with the desired center frequency to shift it up in the spectrum. The center frequency in the 3.1–10.6 GHz range is 6.85 GHz. Several such pulses are shown in figure 2.1 and 2.2. They are plotted together with the FCC emissions masks. When multiplying the Gaussian pulse with the center frequency, a phase offset has to be chosen. In the time domain plots, the phase is chosen so that the pulses are antisymmetric, and in the frequency domain plots, curves from a range of different phases are plotted. We see that for the short pulses, the emission levels in the lower frequencies is very dependent on the phase.

We see from the plots that in order to meet the emission mask constraints, the pulse has to consist of a few cycles. The monocycle pulse simply has a too wide spectrum.

When sending consecutive pulses from for instance a radar, we have to make sure that this is not done in a periodic fashion. If, for instance, pulses are sent periodically at a rate of 10 MHz, the resulting spectrum will consist of spikes at 10 MHz, 20 MHz, and so on, with no energy in between. This means that the spikes will have quite high power and they will thus potentially be able to cause interference to narrow band equipment. By instead sending out the pulses at random intervals, we avoid the spikes

in the spectrum.

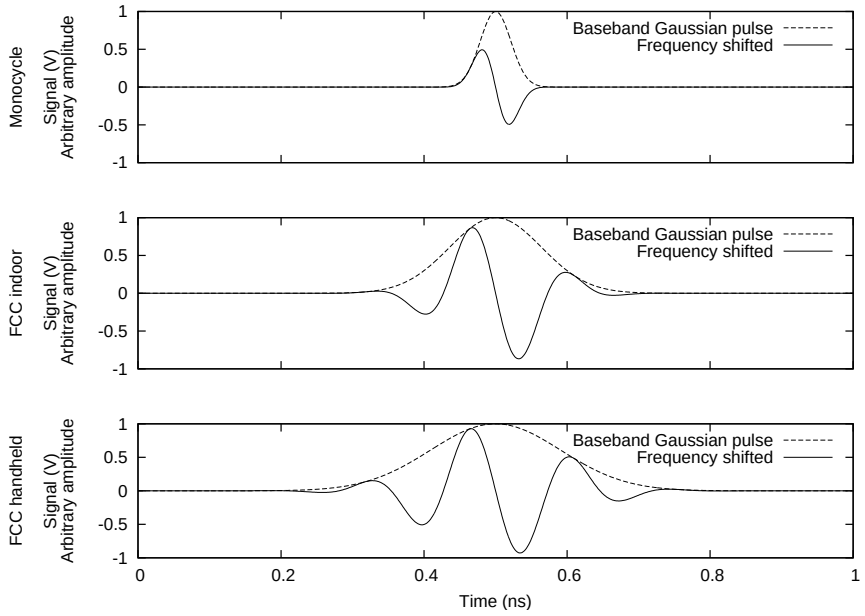

Figure 2.1.: Time-domain plot of different Gaussian UWB pulses

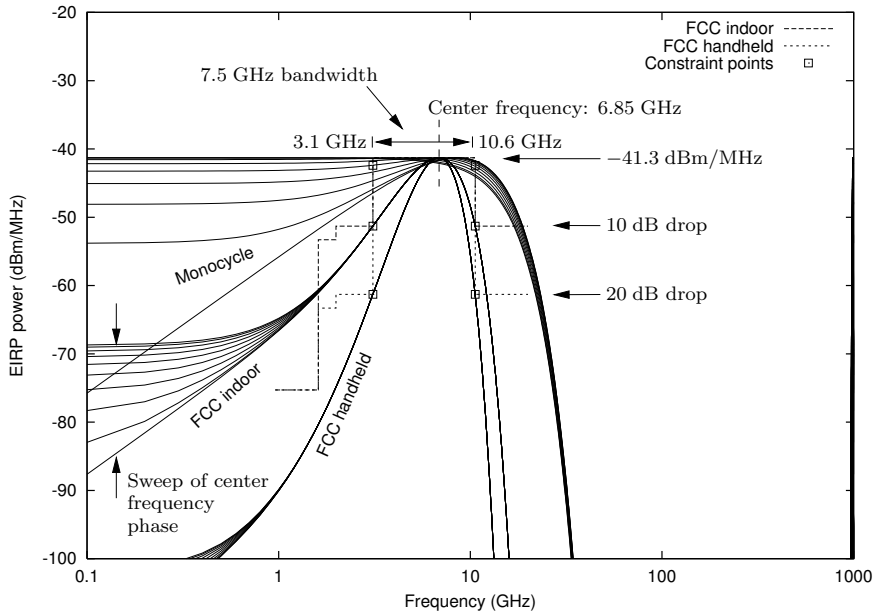

Figure 2.2.: FCC emission mask and spectrum of different Gaussian UWB pulses

3. Radar

3.1. History of radar

The history of radar starts about one hundred years ago. In 1887, Heinrich Hertz started experimenting with radio waves [Hist], and already in 1904, the German inventor Christian Hülsmeyer patented a simple form of radar which he called the *telemobiloscope*. He demonstrated a range of 3km, but no one seemed to be interested in his device. It wasn't until around 1930 that radars were again being developed, this time by at least eight countries who needed it as a warning system for aircraft attacks [Thum]. During World War II, the British radar system *Chain Home* helped the Royal Air Force fight the Luftwaffe [Hist], and since then, radar has become a crucial sensor for both warfare, shipping, aviation, weather forecast, geological surveying and more.

3.2. UWB-IR radar

Conventional radar sends out pulses of a carrier wave and waits for reflections to return. When they do, the radar has to match the received signal with the one sent out. By sending specific pulse patterns, for example barker code, the matching can be done quite precisely, and the round trip time of the signal can therefore be measured with good accuracy despite the pulse pattern being long. The advantage of doing this, is that more energy can be sent out, resulting in a better Signal to Noise Ratio (SNR). Either way, conventional radar systems have a more or less long template to match the received signal with. The cost of this is that the systems can become complex.

A different approach which takes a step in the other direction, is UWB-IR radar systems. By sending out baseband pulses instead of pulses with a carrier frequency, the task of recognizing the return signal becomes much easier. The ideal pulse would be a unit impulse, but since that's not possible, very short pulses will have to do. Still, both transmission and reception of the signal becomes very simple, there's no carrier wave involved.

Since UWB-IR is a baseband signal, it will interfere with a wide spectrum of frequencies. This puts some constraints on the transmission power one can use in the radar systems without causing trouble for other devices. As a consequence, UWB-IR radars become low-power and therefore short-range. In return however, it is possible to create such radars that are high resolution.

4. Continuous-time quantized amplitude signal processing

The increasingly fine-pitched CMOS technology is usually not a good candidate for analog designs [Anne 99, Merc 04]. It is, however, excellent for digital circuits, now reaching speeds of several gigahertz in commodity CPU's. Its widespread use helps keep prices down and push for even further development, making it a very interesting technology. But although the digital circuits are very high performance, there is still some processing power, even of a somewhat analog quality, left to be harnessed from the technology. This is not widely done and we will therefore try to describe the technique in this chapter.

4.1. Signal representation

The quality of the technology which traditional digital design does not exploit, is the timing of a digital signals propagating through logical gates. Take for instance the basic circuit in CMOS technology, the inverter. It can be used as an analog amplifier, a logical inverter of a digital signal, a buffer to handle fan-out and it can also be used for other creative uses such as for instance multilevel logic. But when we are using it to process digital signals, i.e. quantized amplitude, which is what it does best, we can let the timing of the rising and falling edges of our signals represent analog data. That is, we let the very time at which a raising edge occurs, the exact width of the pulse, or some measurement like that, be our quantity of interest (figure 4.1). Even with quantized amplitude, this way of representing information theoretically allows for infinite analog precision. This is actually exactly how information is coded with for instance Pulse-Width Modulation (PWM) and Pulse-Position Modulation (PPM), and

		Time	
		Quantized	Continuous
Amplitude	Quantized	Digital	CTQA
	Continuous	Analog sampled-data (switch-cap...)	Analog

Table 4.1.: Signal processing domains

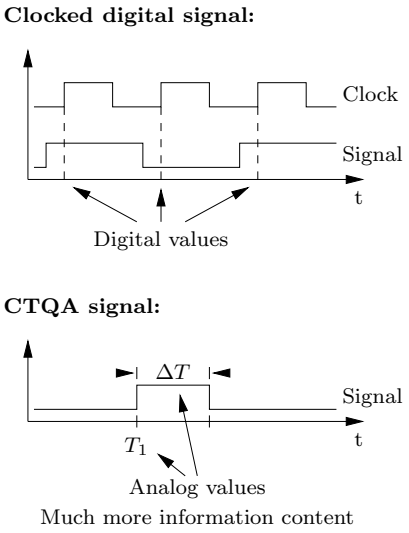

Figure 4.1.: Clocked digital / CTQA signal representation

our signal representation thus bears great resemblance to these modulation schemes, but our signals do no not need to be as periodic and systematic as PWM and PPM signals usually are.

As a simple example of this kind of signal representation, consider a pulse with a width of 1 to 2 ns. It could for instance hold an analog value represented by the width of the pulse. Without phase noise in the system, this analog value would have infinite precision. When sending such a signal through an inverter, the inverter will only add a constant, tiny delay to the time of the transition events, and thus, the information represented by the timing of these events is preserved, only a bit skewed in time. There will of course be introduced noise to the transition timings when the signal goes through the inverter, and in this signal processing domain, our source of noise is therefore the phase noise of the inverter. We will call this signal representation and signal processing domain Continuous-Time Quantized Amplitude (CTQA). Table 4.1 shows how it relates to other signal processing domains in terms of quantized/continuous time and amplitude. An interesting observation to make is that CTQA also relates to the coding of signals in the nervous system. Here, the signals are transmitted as nerve impulses, which is a coding scheme very similar to CTQA with constant pulse width.

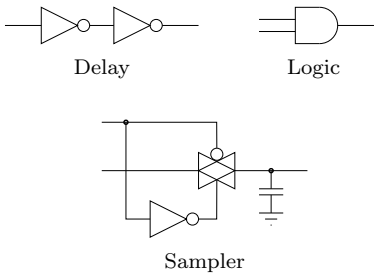

Figure 4.2.: Examples of basic CTQA circuits

4.2. Signal processing operations

The signal processing operations that are available for this kind of signal representation will probably be limited compared to what is available in amplitude-based domains. We will now present some basic operations and circuits, though.

Delay element (Figure 4.2.) A simple cascade of two inverters will delay the signal by two gate delays. If only the relative timing between signals A and B is interesting, delaying B will have the same effect as advancing A, so we have both positive and negative delays available. Also, by delaying A by the minimum buffer delay T, and B by a slightly slower buffer with delay $1.1T$, we get a relative delay of $0.1T$, much smaller than the minimum buffer delay, thus enabling the use of very small delays. This must not be confused with the minimum available pulse width, though, which is only determined by the speed of the technology.

Sampler (Figure 4.2.) A transmission gate can act as basic digital signal sampler, but more elaborate designs may be required for real applications. The sampler can for instance be made to sample an input signal on the rising edge of a trigger signal. The output of the sampler is of lower frequency than the original input signal, and can now be processed by slower, conventional digital circuitry.

Logic (Figure 4.2.) Inverting, and-ing, or-ing, and so forth should be quite simple. Alone, this would correspond to simple combinational logic, where the output is updated as soon as the input is changed. Combined with delays, however, a circuit can for instance detect patterns in time across several input lines. See the monocycle pulse detector example below.

Pulse shaping (Figure 4.3.) Stretching pulses, shortening pulses, turning edges into pulses with constant widths, constraining pulses to maximum and minimum widths, and so forth.

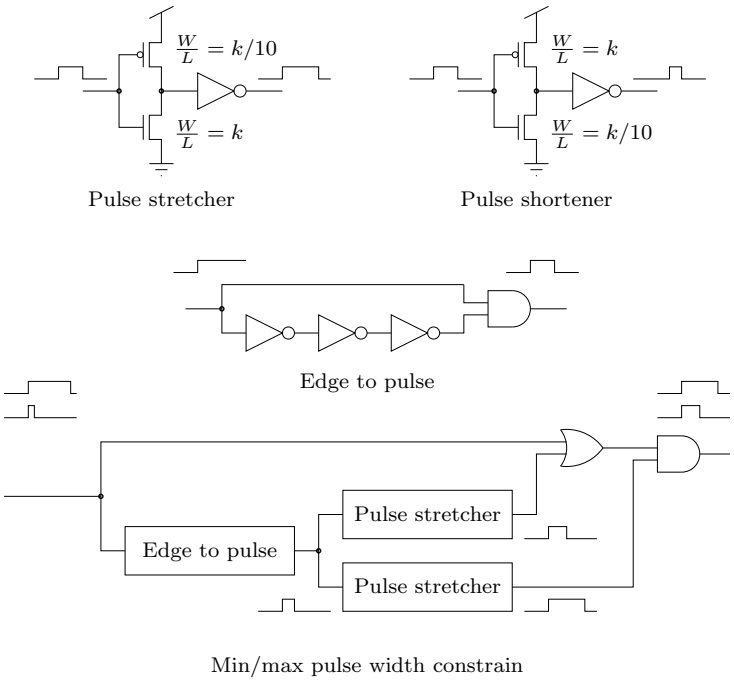

Figure 4.3.: Examples of CTQA pulse shapers

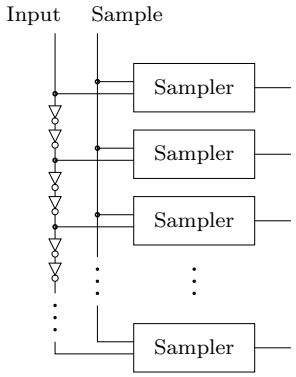

Figure 4.4.: Example circuit using CTQA: Sampled delay line

Example: Sampled delay line (Figure 4.4.) A basic but useful circuit for this signal processing domain is the delay line, consisting of a cascade of inverters. When sending a rising or falling edge into this structure, it propagates down the line at the rate of the gate delay. After some time, the opposite edge can be sent in, even if the first edge is still propagating. This means that the delay line holds a history of edges and it thus forms a sort of memory. A way of getting an actual practical use of this processing domain, is to sample the delay line at one or more points in its spatial structure. By sampling more than one point, we can extract interesting information by looking at the samples relative to each other, and by also sampling at a very specific time, each sample contains information about the value of the input signal at a specific time in the history of the signal before the sampling was triggered. This example has been taken from the RAKE receiver in [Limb 05]. A similar, but slightly different, topology has been used in the circuit implementation described later in this thesis.

Example: Monocycle pulse detector (Figure 4.5.) This circuit uses a delay element and an AND gate to detect a pattern consisting of a pulse on one input line followed by a pulse on another input line a constant period of time later. With its thresholder front end, this circuit is able to detect the analog monocycle pulse shape. This example has been taken from the UWB impulse radio receiver front end in [Meis 05].

What we have now is a set of extremely simple circuits in a very fast technology that is able to transfer, process and to some extent sample a sort of analog values, thus giving the previously purely digital logical gates in the technology analog capabilities while still only using quantized amplitude. We are also turning the gate delay, which is the performance-limiting factor in traditional digital designs, into an analog memory.

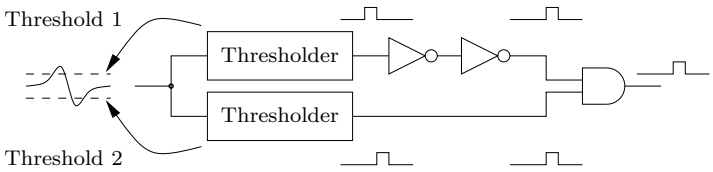

Figure 4.5.: Example circuit using CTQA: Monocycle pulse detector

Operations such as analog addition, multiplication and so forth on the analog values represented by the transition timings of the signals, for instance addition of two pulse widths, is probably much more difficult. And when it comes to processing of multi-bit digital values, CTQA could possibly be used to create for instance some sort of serial adders or similar circuits. But whether that is feasible for, say, 32 bit, and without using clocks, like in traditional serial adders, is another question. So the CTQA signal processing domain may or may not be able to replace parts of or the entire traditional analog and digital signal processing domains. But as we have described it in this chapter, CTQA is a domain of its own, with its own particular properties and its own set of possibilities for signal processing.

4.3. Advantages

Simple As shown in figure 4.2 and 4.3, basic CTQA circuits are very simple and based on digital design. A consequence of the simple, digital circuit architecture, is that they consume very little power, only a little bit caused by leakage currents, when there are no signal transitions. This is in contrast to analog designs which often have some sort of bias currents or other leakage currents constantly running. So in a CTQA system where there are few transitions in the signals, the power consumption could be extremely low, even though the system is capable of high speed processing of analog values, represented by the transition timings of the signal, as soon as any transition is made.

No clock Clock distribution consumes a huge amount of power in high speed digital circuits, and when synchronous global clocks are used, the peak current drawn by the circuit can become larger ([Gonz 05, table 1]). For instance, in [Maho 05], the 90 nm Itanium® processor code-named Montecito is said to use "<25 W" on clock distribution. In a simple digital design, this huge amount of power is even consumed when there is no actual processing being done. The clock is continuously distributed to the entire chip. Needless to say, all this is an enormous waste of energy on distributing a signal that does not really carry any useful information. CTQA circuits, however, do not require a clock. They might have some sort

of a clock, but many solutions probably will not require any. And in those circuits that have one, the clock does not necessarily need to be at the speed at which the signal processing is done. The delay line architecture in [Limb 05], for instance, is used to sample a quantized-amplitude signal at a 1 GHz sampling rate using a 20 MHz clock. This reduction or absence of clock requirements gives CTQA a much nicer power profile than traditional digital circuits. Peak currents are reduced, there is no huge, constant, power consumption even when the circuit is idle, and energy is not wasted on distributing a signal that does not contain any real information.

High speed Since basic CTQA circuits might consist of only short delays and combinatorial logic, they are very fast and yield their results immediately, just like combinatorial logic and analog circuits does. This is in contrast to clocked digital circuits, where the result is not yielded until a given period of time has elapsed to allow all of the circuits in the system to settle first. And since interesting operations can be done with very simple CTQA circuits, we may be able to avoid complex and slow traditional circuit topologies.

Low power As mentioned above, the fact that CTQA circuits are simple, based on digital circuits and not clocked, makes them low power. And, as mentioned, when there is no transitions, the circuits do not use any current at all, except of course some small leakage currents.

Compact memory cells The delayline is essentially a memory element which can hold the analog values represented by the transition timings of the signal, and which has very compact memory cells. Writing to and reading from the memory is of course very simple, but the memory behaves more like a continuously shifting shift register than a traditional long-term memory. This does however fit very well with the CTQA signal processing domain.

Perfect for CMOS As mentioned in the beginning of the chapter, CTQA goes very well together with CMOS. This is because CMOS primarily is a digital technology and not very well suited for analog circuits. Since CTQA is based on digital circuits and not analog ones, but still is able to do some sort of signal processing on the analog values represented by the transition timings of the signal, this is a good match of technology and circuit topology. Also, since CMOS is cheap and quite fast, CMOS and CTQA becomes a very interesting combination.

Unique capabilities As shown in the examples in section 4.2, CTQA is for instance able to detect patterns across time and to sample signals at intervals corresponding to only two inverter gate delays. Both of these tasks can be done with clocked digital circuitry too, but the circuits would probably be much bigger and also

considerably slower. This shows that CTQA has a unique set of capabilities that neither clocked digital nor analog circuits possess.

4.4. Challenges

Limited functionality CTQA is maybe not a general signal processing domain which can be a complete replacement for either analog nor digital signal processing. If this is so, it means it is only good for solving a limited set of problems.

Phase noise In analog designs, the SNR is given by

$$SNR = 20\log_{10}\left(V_{\text{RMS headroom}}/V_{\text{RMS noise}}\right),$$

where we define

$$V_{\text{RMS headroom}} = A_{\text{Headroom}}/\sqrt{2} = V_{\text{Headroom}}/2\sqrt{2},$$

where V_{Headroom} is the voltage span available for signal swing. For CTQA with for instance a PWM-based signal coding, the SNR will be given by

$$SNR = 20\log_{10}\left(\Delta T_{\text{RMS modulation}}/\Delta T_{\text{RMS phase noise}}\right),$$

where we define

$$\Delta T_{\text{RMS modulation}} = \Delta T_{\text{Modulation}}/2\sqrt{2},$$

where $\Delta T_{\text{Modulation}}$ is the difference in width between the shortest and longest possible pulse in the coding. $\Delta T_{\text{RMS phase noise}}$ is the standard deviation of the phase noise introduced by the circuits which the signal travels through. In this case, with PWM coding, it is

$$\Delta T_{\text{RMS phase noise}} = \sqrt{2}\Delta T_{\text{RMS phase noise inverter}},$$

since the phase noise added to the pulse width gets contributions from both the rising and falling edge of the pulse.

So by choosing a big $\Delta T_{\text{Modulation}}$, the SNR increases, but since the pulse width then also increases, the rate at which pulses can be sent into the system will then of course be reduced, causing lower throughput, which corresponds to a reduction in bandwidth. This means that the fundamental limitation to the SNR in a CTQA system will be the phase noise introduced by the circuits.

Mismatch Like phase noise deteriorates the signal, so will device mismatch. One device might be faster than another when they were expected to have the same speed, causing a relative skew between two signal. Depending on the circuit, that could affect the signal processing. It might, however, be possible to compensate this with some sort of tuning of the circuits. But if that is not possible, the device mismatch will add a constant distortion to the signal, effectively lowering the SNR of the system.

Minimum pulse width When sending transitions of a signal into a circuit with increasing speed, i.e. with shorter and shorter delay between one edge and the next, one might start to experience some interaction between the edges. If, for instance, a rising edge through a buffer is almost done pulling the output signal up, and is at, say, 95% when the falling edge comes, then the falling edge will be transmitted faster through the buffer than normal because it is now a shorter way to go down to 50%. So in this way, signal transitions that are close enough may start interfere with each other. And if a short pulse is sent through for instance a delay line, the edges might travel towards each other because of this kind of pulse interaction, causing the pulse to become shorter and shorter until it finally disappears. A second problem that could contribute to this, is mismatch between the travel-speed of a raising and a falling edge through the delay line. If the first NMOS, the second PMOS, third NMOS and so forth, which are the transistors which transmits a rising edge, are either faster or slower than the opposite set of transistors, there will be a difference in travel-speed between rising and falling edges, which could potentially cause already short pulses to be swallowed by the delay line. Because of all this, one should never use pulses below a given minimum width if one wants to be guaranteed that the pulse will travel through the circuit.

4.5. Summary

In this chapter we have tried to explore a signal processing domain which has traditionally not been used very much. The signals in this domain have quantized amplitude, like traditional digital logic, but are not clocked, i.e. they are continuous in time. We have called this signal processing domain Continuous-Time Quantized Amplitude (CTQA).

CTQA circuits have a unique set of properties and capabilities, and are maybe not capable of being general replacements for either analog nor digital circuits. For the tasks they are able to perform, however, they can be much faster, use less power and be much smaller than traditional digital or analog circuits. And because basic CTQA circuits are based on simple digital logic, CTQA is very well suited for cheap and fast modern fine-pitch CMOS. In fact, CTQA should be able to get more processing power

out of CMOS than clocked digital circuits do, but again, the tasks it is able to perform are probably somewhat limited.

5. Sampling methods

The task of a sampler

When the impulse radar sends out a pulse at time $t = 0$, the energy is emitted by the antenna, then propagates through space, hits one or more targets in sequence, then scatters back to the receiver antenna. This means that the signal we receive is a sequence of echoes of the transmitted pulse with varying amplitudes. A range of different effects could complicate the matter, but this remains the essence of the radar. When the antenna has received the backscattered energy and converted it to a voltage signal on a transmission line, the task of a general radar receiver circuit is to sample this signal at a sufficiently high sampling rate.

Sampling rate

Since we operate at baseband throughout the system in our impulse radar, i.e. we do not use a carrier frequency, the sampling frequency has to be at least twice that of the frequency of our pulse in order to get a correct, full readout of the signal. That being said, it might however be that we are not really interested in the full readout. Maybe we just want one or a few samples at given points of time, or maybe we create a front end in front of the sampler which somehow stretches a received pulse in length so that we can sample at lower frequencies. For the remainder of this chapter, however, we will assume that a full readout is required.

In order to get a full readout of our signal which has frequency content in the 3.1–10.6 GHz range with center frequency of 6.85 GHz, we have to sample with about 21 GHz. This is quite a lot to demand from and AD converter, and presents our first problem. In this chapter we will show how the strobed sampler can help us out here.

Noise

Our second problem is noise. The farther away the target is, the weaker the returned echo will be. Since the noise received at the antenna and created by the receiver system itself will be the same, this means our signal to noise ratio will get lower as the target moves away from the radar. In order to compensate this, we have to integrate the energy from several received pulses. In this chapter, we will present several topologies for doing that.

Note that we do not consider radar clutter to be noise in this context. One can not integrate away that noise. We will call the signal without any noise, but potentially with unwanted clutter the "ideal signal" or "ideal V_{in}". The noise will be assumed to be ad-

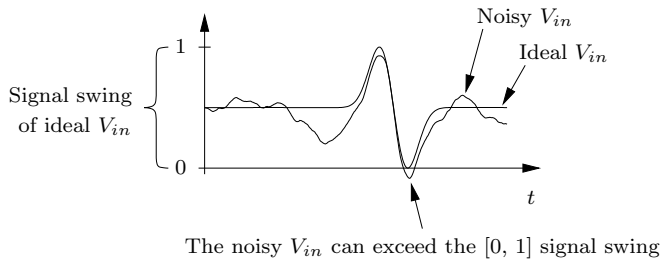

Figure 5.1.: Signal swing used in analysis

ditive white Gaussian noise, according to the Additive White Gaussian Noise (AWGN) channel model. This means that the actual received signal will be ideal $V_{in}(t) + N(t)$, where $N(t)$ is the noise signal, which has a Gaussian, i.e. normally distributed, Probability Density Function (PDF) with mean $\mu = 0$ and standard deviation σ_N. $N(t)$ must be considered to be a "stochastic function", to put a name on it. Each time the radar receives its sequence of echoes, said to start at $t = 0$ each time, the ideal $V_{in}(t)$ will be the same, but $N(t)$ will be different each time, which results in that $V_{in}(t)$, the actually received signal, will also be different each time, and also a "stochastic function". As a convention for further analysis, we will assume that the ideal V_{in} has a range of 0–1 V. The appearance of the signal is illustrated in figure 5.1.

Chapter overview

In the remainder of this chapter we will present the principle of the strobed sampler and a few samplers using this principle. We will perform system-level simulations on three of the samplers and compare them to each other. And in addition to the simulations, we will also try to find analytical expressions for the performance of the samplers.

5.1. Strobed sampler

The idea of the strobed sampler is to sample at only a single point in time, τ (figure 5.2). This may not sound very interesting, but by repeating the process while sweeping τ or by using parallel structures with different τ, one can effectively get a sampling frequency as high as one wants. The bandwidth of the part of system before the sampler switch will of course be a limiting factor to how high frequencies can be sampled. Also, the sampler switch will not be infinitely fast, so some averaging over a time span of the signal will occur when the sampling is made. This again reduces how high frequencies can be sampled. Furthermore, jitter in the τ delay can also reduce the maximum feasible sampling frequency.

So in a strobed sampler, the only part of the input signal we are interested in, is $V_{in}(\tau)$.

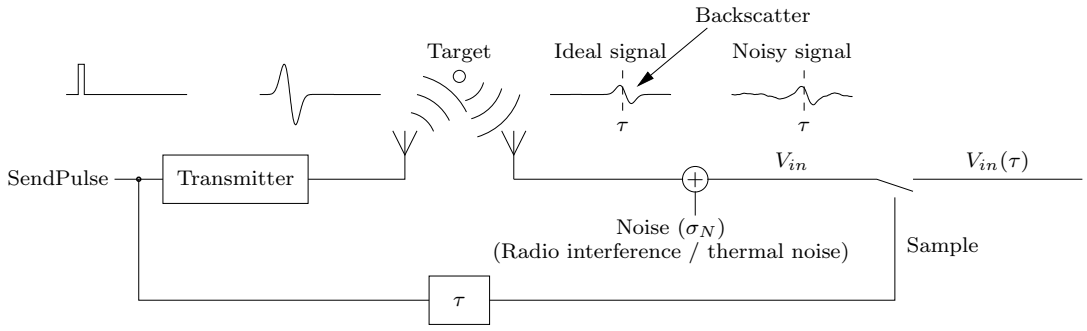

Figure 5.2.: Strobed sampler

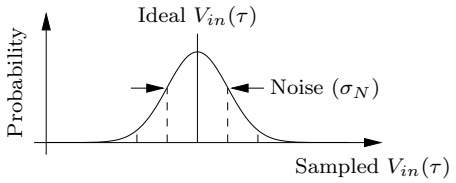

Figure 5.3.: PDF of sampled signal

Because of noise, this becomes a stochastic variable, i.e. it varies from one sampling to the next. The sampled signal will therefore have a Gaussian PDF with a given σ_N (figure 5.3).

Averaging sampler

If the noise, σ_N, is too great for a given application, it can be reduced by averaging multiple samples. This means that we need to receive the same ideal V_{in} multiple times so that we can sample $V_{in}(\tau)$ multiple times. The actual V_{in} and $V_{in}(\tau)$ will of course vary because of noise, but that is exactly the point. By somehow averaging several $V_{in}(\tau)$ samples, the noise will be reduced. The more samples are averaged, the weaker the noise becomes. Since this is an attempt to recover the ideal $V_{in}(\tau)$, we will call the output from the averaging sampler $V_{in\ recovered}(\tau)$. The remaining noise in $V_{in\ recovered}(\tau)$, we will call $\sigma_{N\ recovered}$. So the the essence of the averaging sampler is to average multiple samples to reduce the noise to an acceptable level (figure 5.4). The other samplers in this chapter are strobed samplers with different averaging strategies.

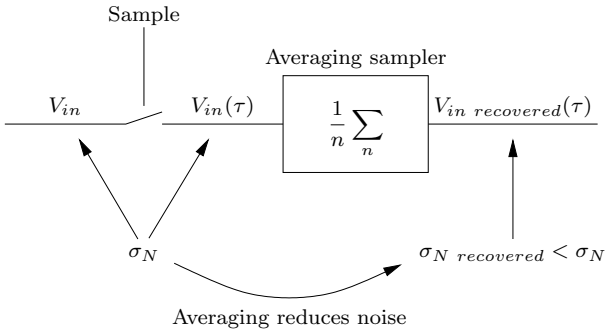

Figure 5.4.: Averaging sampler

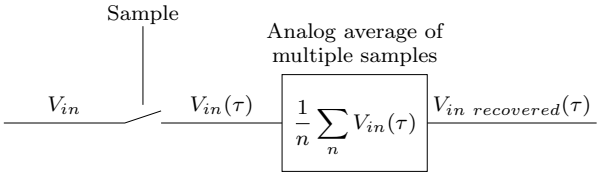

Figure 5.5.: Theoretical analog average sampling method

5.2. Theoretical analog average sampler

As a reference point for the other averaging samplers, we will use a theoretical analog average sampler. It might be possible to implement this sampler in a circuit, but here we will only concern ourselves with its mathematical representation.

The analog average sampler works by simply taking an arithmetic average of several samples (figure 5.5):

$$V_{in\ recovered}(\tau) = \frac{1}{n}\sum_n V_{in}(\tau), \tag{5.1}$$

where $V_{in}(\tau)$ is a stochastic variable evaluating to different values at each of the n samplings. Since the noise of $V_{in}(\tau)$ is normally distributed, it is reduced by the square root of the number of samplings, n:

$$\sigma_{N\ recovered} = \frac{\sigma_N}{\sqrt{n}} \tag{5.2}$$

This means we can get arbitrarily low $\sigma_{N\ recovered}$ by choosing n large enough. By for instance averaging over $n = 100$ samples, the noise is reduced to $\sigma_{N\ recovered} = \sigma_N/10$.

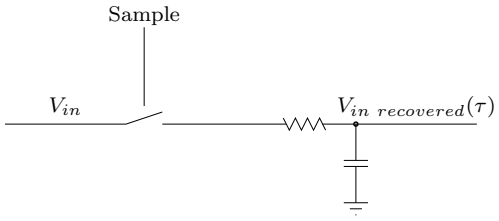

Figure 5.6.: Approximate functional equivalent of MIR

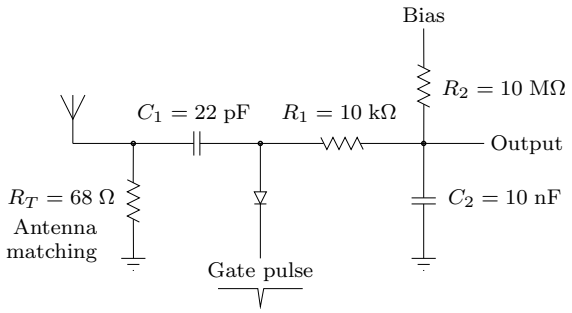

Figure 5.7.: Simplified MIR circuit

The analog average sampler is possibly the best averaging sampler available, but there might perhaps be room for some improvements like for instance rejecting extreme samples.

5.3. MIR

The Micropower Impulse Radar (MIR) from [McEw 94] has a strobed sampler circuit which can be implemented using discrete components. An approximate functional equivalent of the sampler is show in figure 5.6. As we can see, it is quite similar to the theoretical analog average sampler in that it averages the analog values of several samples. The difference, though, is that the averaging is weighted, i.e. the most recent samples are more heavily weighted in the average than the older ones. Also, when the ideal $V_{in}(\tau)$ changes rapidly by a large value, there will be a given settle time before the output has stabilized to the new value.

A simplified schematic of the actual MIR circuit is shown in figure 5.7. It works by sampling the input from the antenna using a diode which gets pulled down at $t = \tau$. To look at the details of its operation, consider first R_1. This resistor creates together with C_1 a high pass filter or AC-coupling with a cut-off below 1 MHz, meaning the

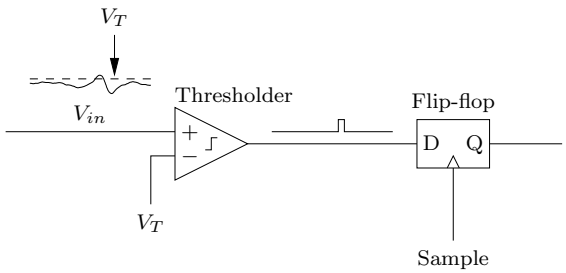

Figure 5.8.: Thresholded sampler

antenna signal goes straight through C_1, unaffected by R_1. So for the high frequency input signals from the antenna in the GHz range, the right part of the circuit can be considered to be non-existent. When $t = \tau$, a gate pulse pulls the diode down, causing it to start to conduct. This will charge the capacitor C_1 for a short while, i.e. the duration of the gate pulse. Eventually, C_1 will hold the voltage $V_{in}(\tau)$, the antenna voltage at the time the gate pulse is triggered, minus the voltage above the diode, which will be the lowest voltage of the gate pulse plus the diode threshold voltage. So, clearly, the capacitor C_1 will become charged with a voltage which is proportional to $V_{in}(\tau)$. But since the diode can only pull the charge one direction, R_1 and R_2 are used to pull the charge of C_1 the other way. This way, the C_1 voltage will always follow $V_{in}(\tau)$ as it goes up and down over time. Note that R_T, the antenna or transmission line impedance and the diode will be the the R in the RC time constant which determines the charging of C_1.

If we look at frequencies in the kHz range, the gate pulse and diode becomes merely ignorable glitches and can be ignored. The antenna signal will hopefully be about 0 at these frequencies, so the antenna can also be ignored. Thus we end up with only C_1 in series with R_T and R_1 connected to the output. R_1 and C_2 forms a low pass filter of about 1.5 kHz, and so, only frequencies below this cut-off will appear on the output. The signal on the output will then simply be the low pass filtered voltage of C_1, which in turn is proportional to $V_{in}(\tau)$, and we thus have an averaging strobed sampler. One thing to note here, is that the diode used might introduce non-linearities because of its exponential characteristic.

5.4. Thresholded sampler

A basic thresholded sampler is shown in figure 5.8. The input signal V_{in} is thresholded with a threshold V_T, resulting in a Continuous-Time Quantized Amplitude (CTQA) signal. CTQA signal processing is described in chapter 4. Next, the quantized signal is

sampled at time $t = \tau$ by a simple D-flip-flop. Much information is of course thrown away here when we threshold the signal, but much of the circuit can now use simple digital logic, so the tradeoff could be reasonable in some cases.

A problem with the thresholded sampler, is that quantized pulses shorter than a given minimum width might not survive through the buffers and logic of the system. This represents a non-linearity in the sampler system, since the sampled bit might not actually correctly reflect whether $V_{in}(\tau)$ was above or below the threshold. The surrounding signal values, before and after $t = \tau$, will then be the deciding factors. We will ignore this effect in our further analysis of thresholded samplers in this chapter since the effect is highly non-linear and could complicate analysis considerably.

A practical circuit might need a slightly more sophisticated quantized sampler than a simple D-flip-flop, since the digital value could be in the middle of a transition at the exact time it gets sampled. A regular D-flip-flop might not handle that very well.

5.4.1. Swept threshold sampler

We have called this sampler the swept threshold sampler. It is much like ramp-compare or flash Analog to Digital Converters (ADCs), but besides its operating regions which resemble this ADCs, we will also explore its performance in cases of large amounts of noise, which constitute additional regions of operation.

The ramp-compare ADC works by comparing the input signal with a ramp. When the two signals cross, a comparator triggers, and if we know the value of the ramp at this exact time, we also know the value of the input signal. By using a ramp with a known slope we can get the amplitude using a clock. A second approach is to create the ramp using a DAC. In both cases, we know the ramp amplitude and thus the input signal amplitude when the comparator triggers.

A flash ADC works by using for instance 255 different comparators to compare the input signal to 255 different threshold levels. The result of this is a thermometer coding output, i.e. one end of the string of the 255 output bits will be a sequence of ones, and the other end of the string will be zeros. This can then be decoded to an 8-bit digital value.

When there is no noise in the system, the swept threshold sampler works like a serialized flash ADC. Instead of using a massively parallel structure, we instead sweep the threshold over time and perform the comparison with the input signal multiple times. Each time a comparison is made, we increment a counter if the comparator output was 1. This should be functionally equivalent to the flash ADC, but it means we have to repeat the sampling for instance 255 times. This is the price we pay for a smaller circuit.

By combining this serialized flash ADC architecture with strobed sampling, we get the system shown in figure 5.9.

When there is noise in the system, though, some erroneous thresholdings might occur with the thresholds which are near to the input signal amplitude. Some thresholdings

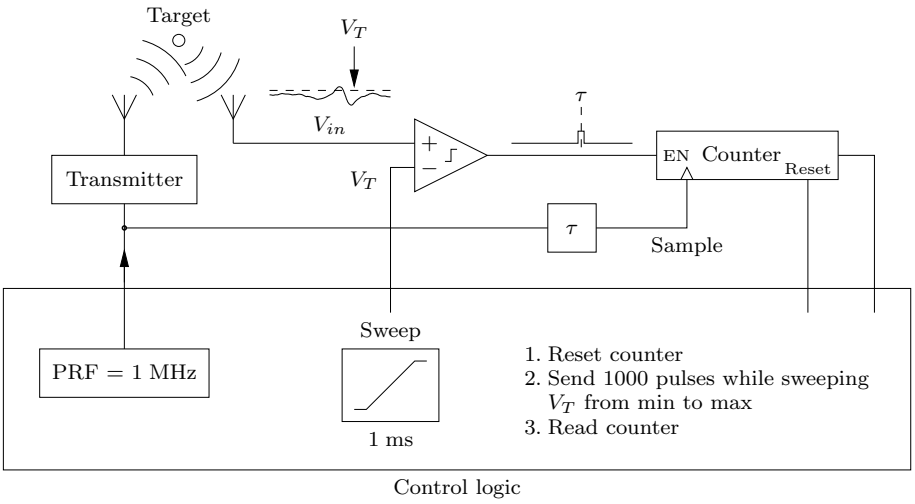

Figure 5.9.: Swept threshold sampler

that should have been 0, could become 1, and opposite. This will somewhat even each other out, but the total counter value will still contain noise because of this. Also when the noise is strong, even larger than the signal, we will be able to recover the input signal. The output signal will of course be very noisy, but by simply averaging multiple of these output signals, we can reduce the noise in the final recovered signal. This, of course, is the same as just increasing the number of steps in the first place. So the more steps, the more the noise gets reduced. The process of sampling a noisy signal is shown in figure 5.10.

Some aspects of the behavior of a noisy multiple-threshold system like this is described in [Stoc 00], and there, the term Suprathreshold Stochastic Resonance (SSR) is coined. Stochastic resonance is a term usually used to describe how a weak signal can be helped through a non-linear system, for instance a thresholder, by adding noise. The purpose of the noise here is to occasionally push the input signal over the threshold, which it could not cross by itself. The term Suprathreshold Stochastic Resonance is meant to describe the case where the signal is not weak anymore, but where stochastic resonance phenomena are still observed.

Non-linearity compensation

When the input signal is at rail, i.e. 0 or 1 using our convention, the expected number of counts is not 0 or n, the number of steps, as expected. For instance when the signal is 0, we could get erroneous ones from the comparator, but we could not get erroneous zeros, since all comparator outputs are supposed to be zero anyway. This means that

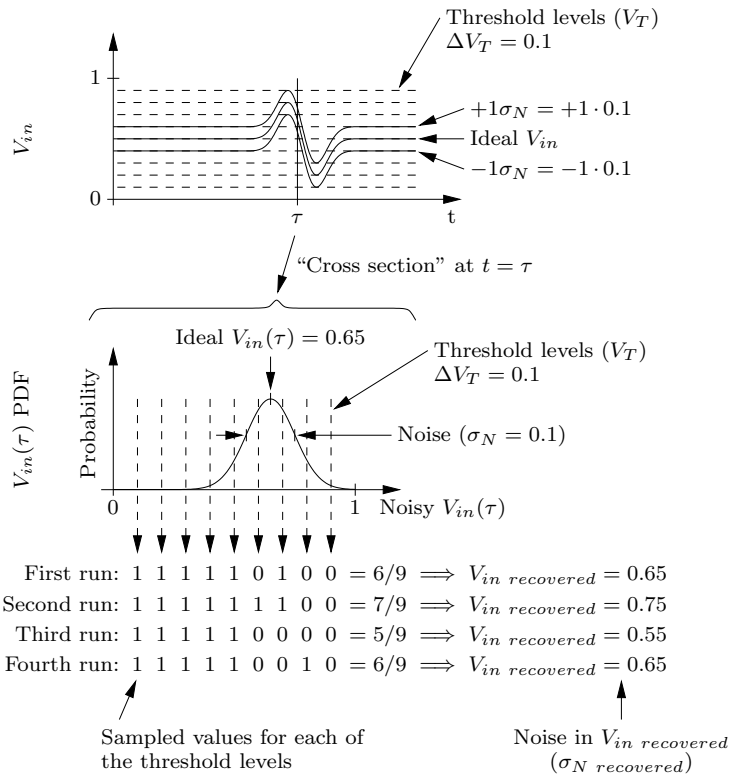

Figure 5.10.: Swept threshold sampler principle of operation

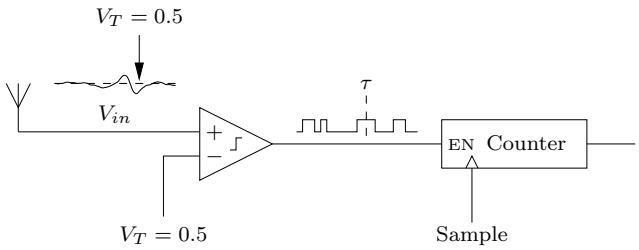

Figure 5.11.: Stochastic resonance sampler

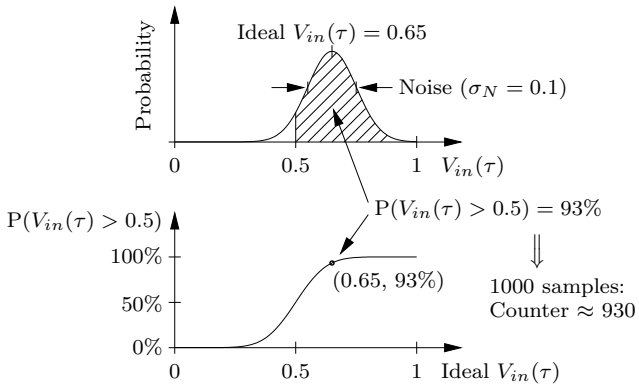

Figure 5.12.: Stochastic resonance sampler principle of operation

expected counter value will have a tendency towards $n/2$ when the input signal approaches rail. By creating a translation map from counter value to recovered signal value, this is easily compensated. This does however require that we know σ_N.

The same applies for extremely noisy signals, where none of the thresholds will have a zero probability of producing an erroneous comparison. This means that we will seldom get a counter value of either 0 or n. The counter values will rather be concentrated around $n/2$. Again, a translation map made using knowledge about σ_N fixes the problem.

5.4.2. Stochastic resonance sampler

In what we have called the stochastic resonance sampler, the threshold is permanently set to the DC of the signal, which is 0.5 with our convention (figure 5.11). This is does at first seem like a simple 1-bit sampler, but if there is a lot of noise in the signal, on the order of the signal itself, this sampler starts to display some very interesting characteristics. If the signal, $V_{in}(\tau)$, is for instance 1.5 standard deviations, σ_N, over DC, i.e. 0.5,

and $\sigma_N = 0.1$, the signal will be $0.5 + 1.5 \cdot 0.1 = 0.65$, and the probability of the thresholder giving a 1, will be 93%. This percentage is given by the area of the PDF which is above 0.5 (figure 5.12). If we perform for instance 1000 samplings, we should expect to get counter value of about 930.

By sampling this way and mapping the counter value to standard deviations, we get information about how many standard deviations the signal is above or below 0.5. If we also know σ_N, we can calculate the absolute value of the input signal. There will of course be noise in this recovered signal.

This kind of sampler is analyzed in [Stoc 00] in terms of information transmission through the system. We will, however, perform simulations later in this chapter where we look at the noise levels in the input signal and the recovered signal, rather than information transmission. This is probably more relevant for radar applications.

5.4.3. Hybrid

If there is noise in a swept threshold sampler which is greater than the threshold steps ΔV_T, and a reduction in the noise of the recovered signal is desired, one would usually just increase the number of threshold steps and thus reduce also ΔV_T. This will require a more advanced DAC to set the threshold levels, so one might want to avoid this. By instead increasing the number of samplings per threshold level, one will essentially be using stochastic resonance sampling at each of the threshold levels, and should thus be able to reduce the noise of the recovered signal. A different way to look at it, is to say that we simply perform a normal set of samplings to get $V_{in\ recovered}(\tau)$, but that we then repeat this process several times and then average the result. This will reduce the noise in the recovered signal, $\sigma_{N\ recovered}$.

This is a bit similar to dither, a technique used in normal ADCs, where noise is added to the input signal. By repeated sampling of the same signal, we can then recover amplitudes lower than the original quantization error of the ADC.

5.5. Simulations

Simulations have been performed on the analog average, swept threshold and stochastic resonance samplers in order to characterize them and to compare their performance. Our value of interest is the noise in the recovered signal, $\sigma_{N\ recovered}$. The lower it is, the better the signal to noise ratio is. Two parameters determine its strength. The first is obviously the noise σ_N in the input signal $V_{in}(\tau)$. The second is n, the number of samples we average over. The higher n is, the lower $\sigma_{N\ recovered}$ becomes. However, increasing n means sending out more pulses from the radar, and that costs energy, fills up our spectrum allocation and takes time, so we want to get a good $\sigma_{N\ recovered}$ with as

few samplings as possible. The sampling method determines how high n must be to achieve a given $\sigma_{N\ recovered}$.

The conditions of the simulation are as follows:

- We want to sample $V_{in}(\tau)$ for a given τ, which is kept constant.
- The ideal $V_{in}(\tau)$ lies between 0 and 1.
- $V_{in}(\tau)$ has Gaussian noise with standard deviation σ_N.
- n samplings are performed. For the swept threshold sampler this also means n threshold levels.
- A value $V_{in\ recovered}(\tau)$ is the result of the samplings.
- When the sampling process is repeated, $V_{in\ recovered}(\tau)$ will attain different values each time. The standard deviation of this noise in $V_{in\ recovered}(\tau)$ will be called $\sigma_{N\ recovered}$, and is our value of interest.
- $V_{in\ recovered}(\tau)$ may not always be normally distributed.
- $\sigma_{N\ recovered}$ is calculated as the RMS error of the $V_{in\ recovered}(\tau)$ PDF. If this value varies with the ideal $V_{in}(\tau)$, the largest value is used.
- We will sweep both σ_N and n to properly characterize the samplers.

5.5.1. Sampler PDFs

When we are sampling a signal $V_{in}(\tau)$ we need to find the PDF of $V_{in\ recovered}(\tau)$ in order to analyze the performance of the sampler. To find this PDF, we first choose a value for n, σ_N and the ideal $V_{in}(\tau)$. We then calculate the PDF of $V_{in\ recovered}(\tau)$. This will be different from one sampling method to another. The ideal PDF would be a single spike at the value of the ideal $V_{in}(\tau)$. But because of the noise in the input signal, the PDF will be blurred out. And in the swept threshold and stochastic resonance samplers, quantization effects will create scattered, separate spikes rather than a continuous PDF.

By sweeping the ideal $V_{in}(\tau)$, we get a series of PDFs. To display all these simultaneously, we can map the probabilities to gray scales and display the PDFs as a 2D picture (figure 5.13), where each vertical column represents a single PDF. Here, white means a zero probability of a given value of $V_{in\ recovered}(\tau)$ occurring, and black means a 100% probability. As expected, when the ideal $V_{in}(\tau)$ is near 0, the PDF will be concentrated near $V_{in\ recovered}(\tau) = 0$, and when $V_{in}(\tau)$ is near 1, the PDF will be concentrated near $V_{in\ recovered}(\tau) = 1$. If there is much noise in $V_{in\ recovered}(\tau)$, this will be visible as a vertical blur in each column, which of course will just look like a general blur of the line which goes from (0, 0) to (1, 1).

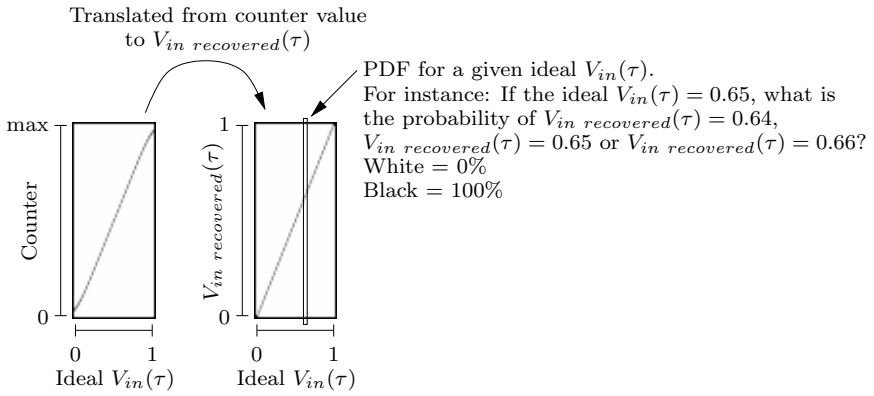

Figure 5.13.: How to read sampler PDFs

Also shown in figure 5.13, is the counter PDF. This is essentially the same as the $V_{in\ recovered}(\tau)$ PDF, but before the compensation for the sometimes non-linear response of the counter value to the ideal $V_{in}(\tau)$ value.

Figure 5.14 shows PDFs for a range of both n and σ_N for both the swept threshold and the stochastic resonance samplers, both the counter PDF and the $V_{in\ recovered}(\tau)$ PDF. Figure 5.15 and 5.16 group these same PDFs by sampler, then counter / $V_{in\ recovered}(\tau)$.

In figure 5.15 (swept threshold) we see how increased σ_N causes more blurred PDFs and how increased n reduces this blurring again, thus reducing $\sigma_{N\ recovered}$. For $\sigma_N = 0.32$ we see how we get non-linearities near 0 and 1, as described in section 5.4.1. We also see how this is compensated in the $V_{in\ recovered}(\tau)$ PDF. For $\sigma_N = 1$ and 3.2 we see the second problem described in section 5.4.1, where the counter hardly ever reaches 0 or max. Again, this gets compensated. For $n = 10$ we can clearly see the quantization effect.

In figure 5.16 (stochastic resonance) the PDFs are quite different. For low noise cases, the usable $V_{in}(\tau)$ signal swing becomes very small, and if one goes beyond this, the counter goes to 0 or max, which translates to 0 or 1 for $V_{in\ recovered}(\tau)$. This is quite logical, as there simply is not enough noise to make $V_{in}(\tau)$ cross the threshold when it is far away from the threshold $V_T = 0.5$. So for certain values of the ideal $V_{in}(\tau)$, the stochastic resonance sampler performs very poorly when there is little noise. When $\sigma_N = 0.1$ or 0.32, we see that the counter PDFs look like the cumulative normal distribution function. Actually, this is true for all values of σ_N, but it is only at this "zoom level" that we can see the whole shape clearly. At $\sigma_N = 0.32$ and above we see that the noise is strong enough to allow for a full signal swing of the ideal $V_{in}(\tau)$ from 0 to 1. The translation from counter value to $V_{in\ recovered}(\tau)$ value is necessary to compensate

the non-linear response of the counter value to the ideal $V_{in}(\tau)$ value. We can also observe, as expected, that an increase in n reduces the spread of the PDF, but also that it expands the usable signal swing region in low noise cases.

In figure 5.14 we can observe that when there is very much noise, with $\sigma_N = 1$ or 3.2, the two samplers have very similar PDFs. This suggests that the swept threshold sampler, which has its threshold levels quite close together compared to the noise PDF when there is much noise, might start to behave like the stochastic resonance sampler, which has its threshold levels completely squeezed together, when there is much noise. We will therefore refer to the high noise case as the stochastic resonance region of the swept threshold sampler.

5.5.2. Finding the RMS error

To calculate the RMS error we take the square root of the sum of the squares of the possible $V_{in\ recovered}(\tau)$ amplitude deviations from the ideal response weighted by their respective probabilities of occurring:

$$RMSerror = \sqrt{\sum_{V=0 \text{ to } 1 \text{ (small steps)}} \underbrace{\mathrm{P}(V_{in\ recovered}(\tau) = V)}_{\text{Probability of deviation}} \cdot (\underbrace{V - V_{in\ ideal}(\tau)}_{\text{Deviation}})^2} \tag{5.3}$$

This essentially calculates the standard deviation of the PDF, but using the ideal $V_{in}(\tau)$ instead of the actual mean of the PDF when calculating the errors or deviations. This ensures that PDFs with incorrect means get a high RMS error. If $V_{in\ recovered}(\tau)$ has an offset, it does not help if the regular standard deviation of the PDF is low, so this should be a good measure for the noise in the recovered signal, or maybe more correctly, noise + distortion.

We have now calculated the RMS error for a given PDF, but there are of course multiple PDFs for a given combination of n and σ_N, so all these should be calculated. This is shown in figure 5.17 and 5.18. Only the shape of the variations of the RMS error is shown, the graphs are not to scale, but they all go from 0 up to some RMS value. The dashed line shows a minimum RMS error calculated from the quantization levels. This is necessary because the RMS error is only sampled for a few discrete values of the ideal $V_{in}(\tau)$, which could mean we miss the maximum error. The calculated dashed line alleviates this problem to a certain extent, but does not fix it completely.

In figure 5.17 (swept threshold) we can see for the low noise $n = 10$ cases that the RMS error goes up and down as the ideal $V_{in}(\tau)$ goes in and out of sync with the threshold levels. In the other cases, the RMS error is relatively uniform across the ideal $V_{in}(\tau)$ range.

In figure 5.18 (stochastic resonance) we can observe for the low noise cases how the RMS error skyrockets as we go out of the usable operating region. This indicates, as expected, that the low noise cases have very bad worst-case RMS errors.

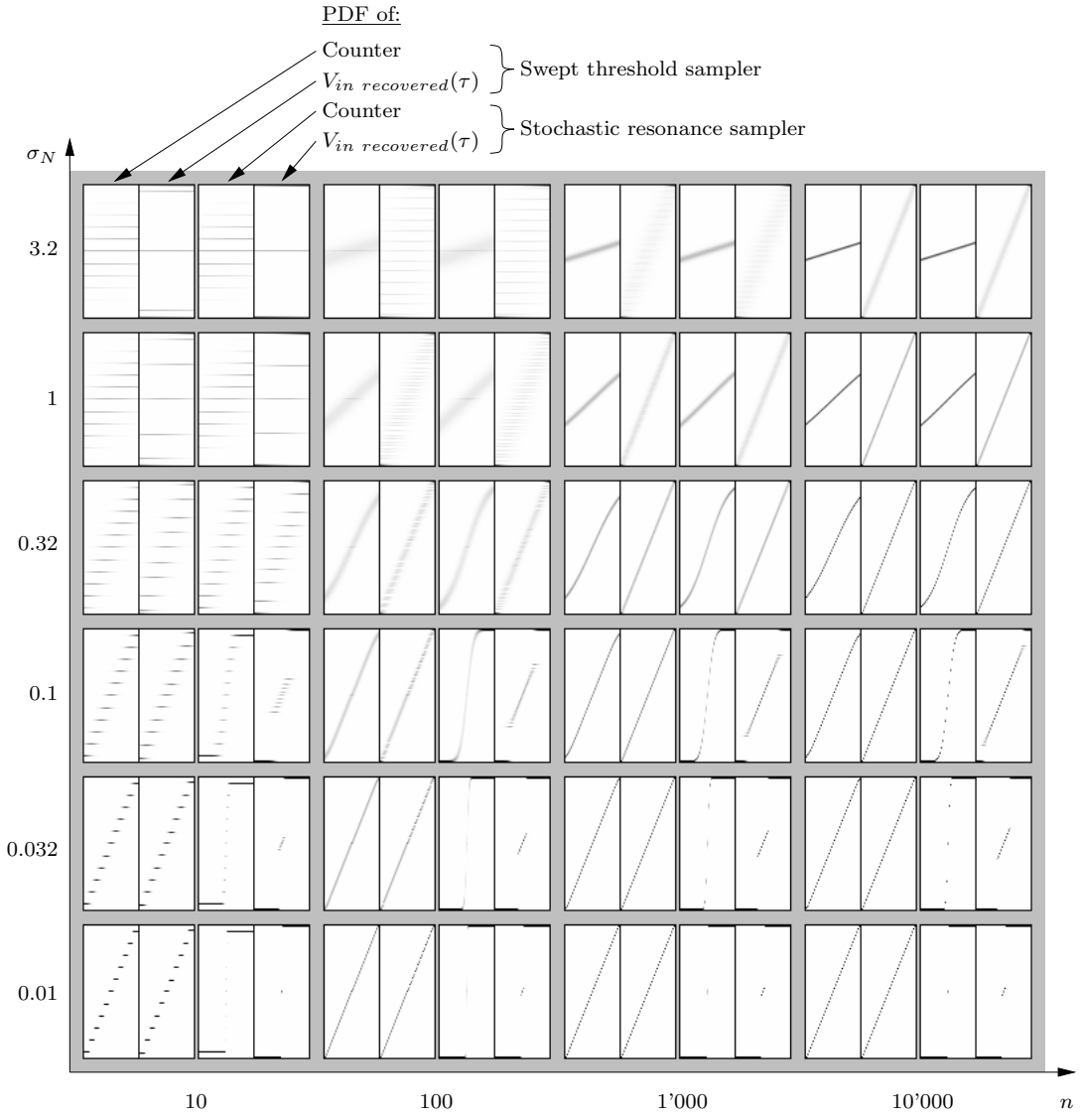

Figure 5.14.: PDFs of samplers

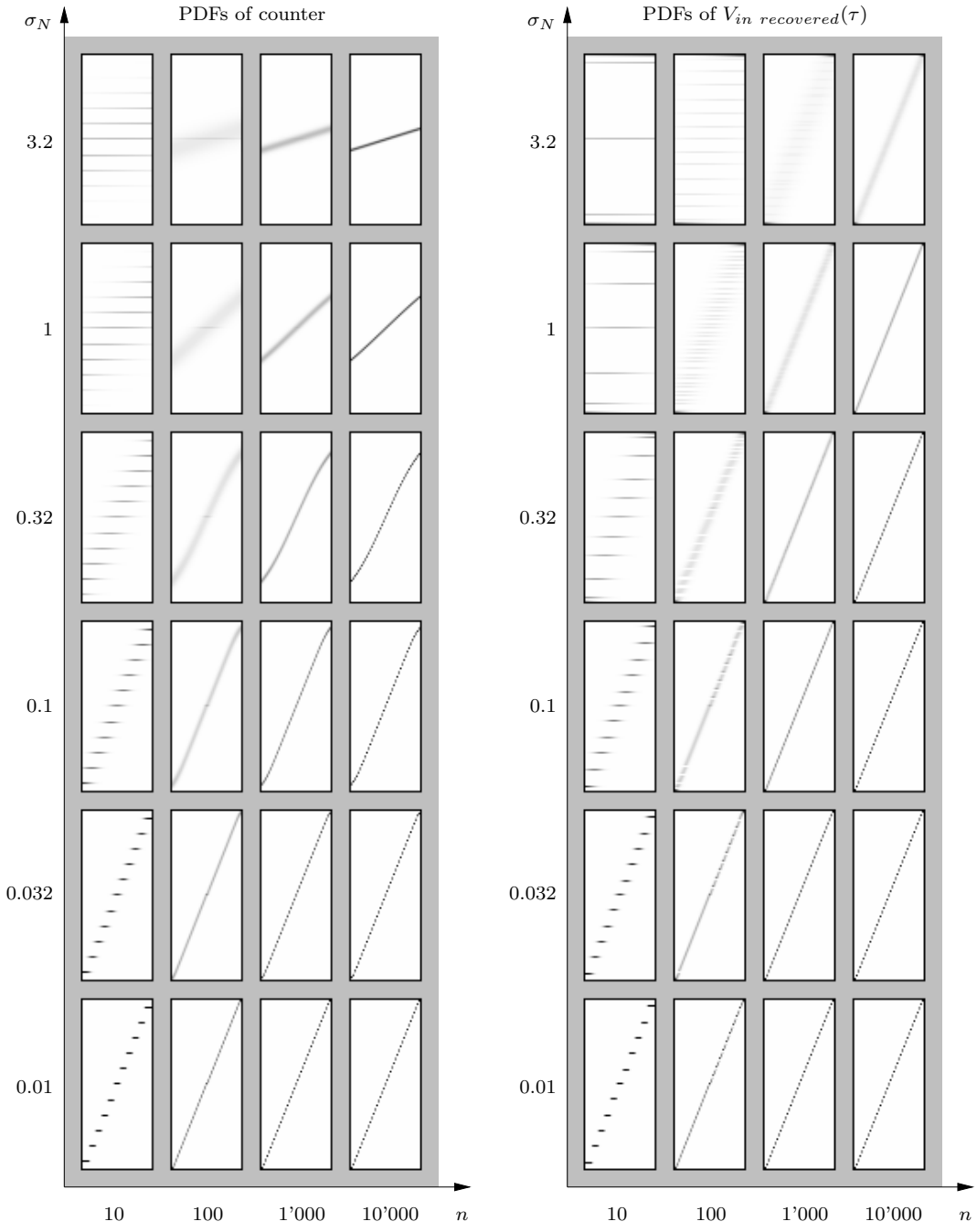

Figure 5.15.: Swept threshold sampler PDFs

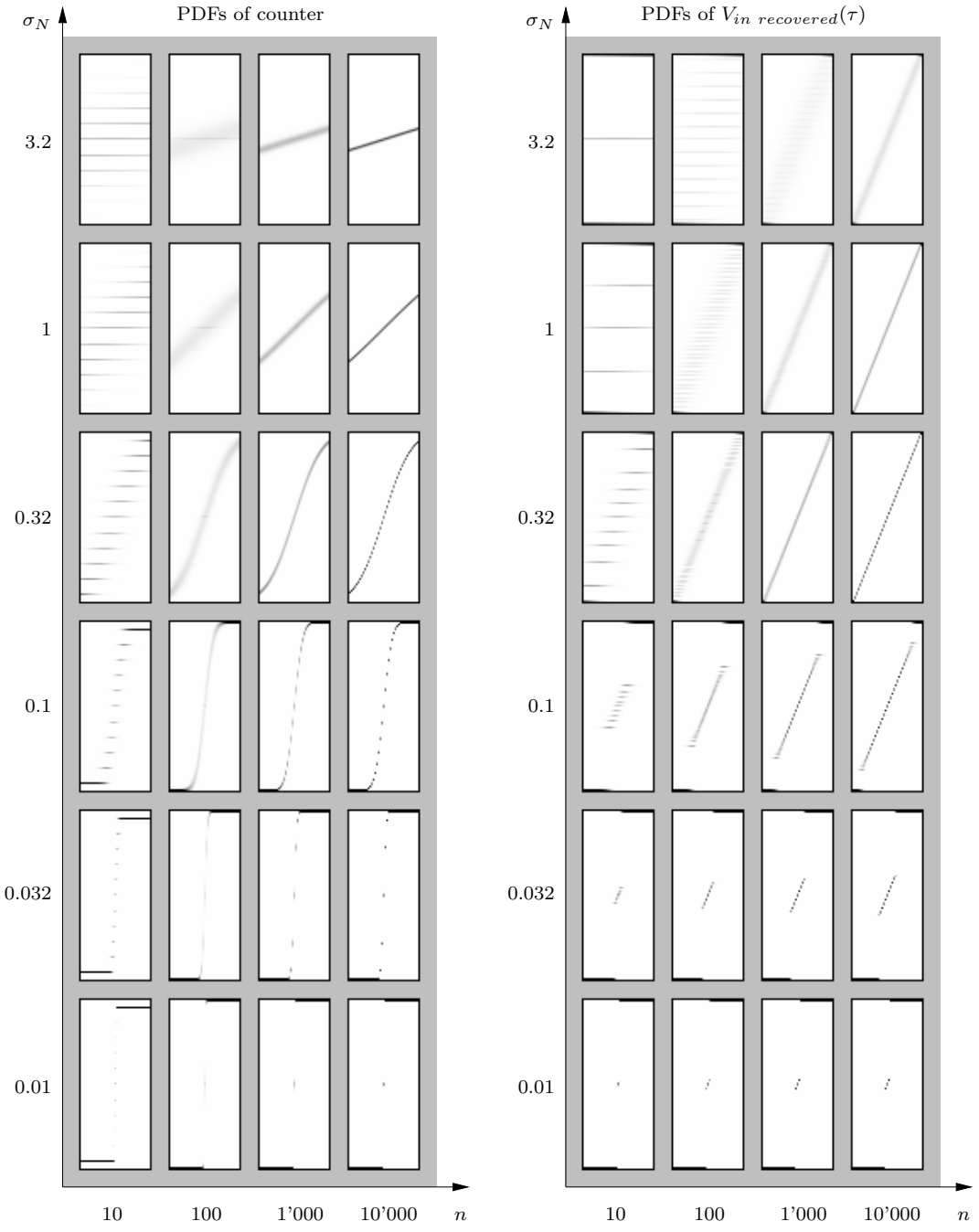

Figure 5.16.: Stochastic resonance sampler PDFs

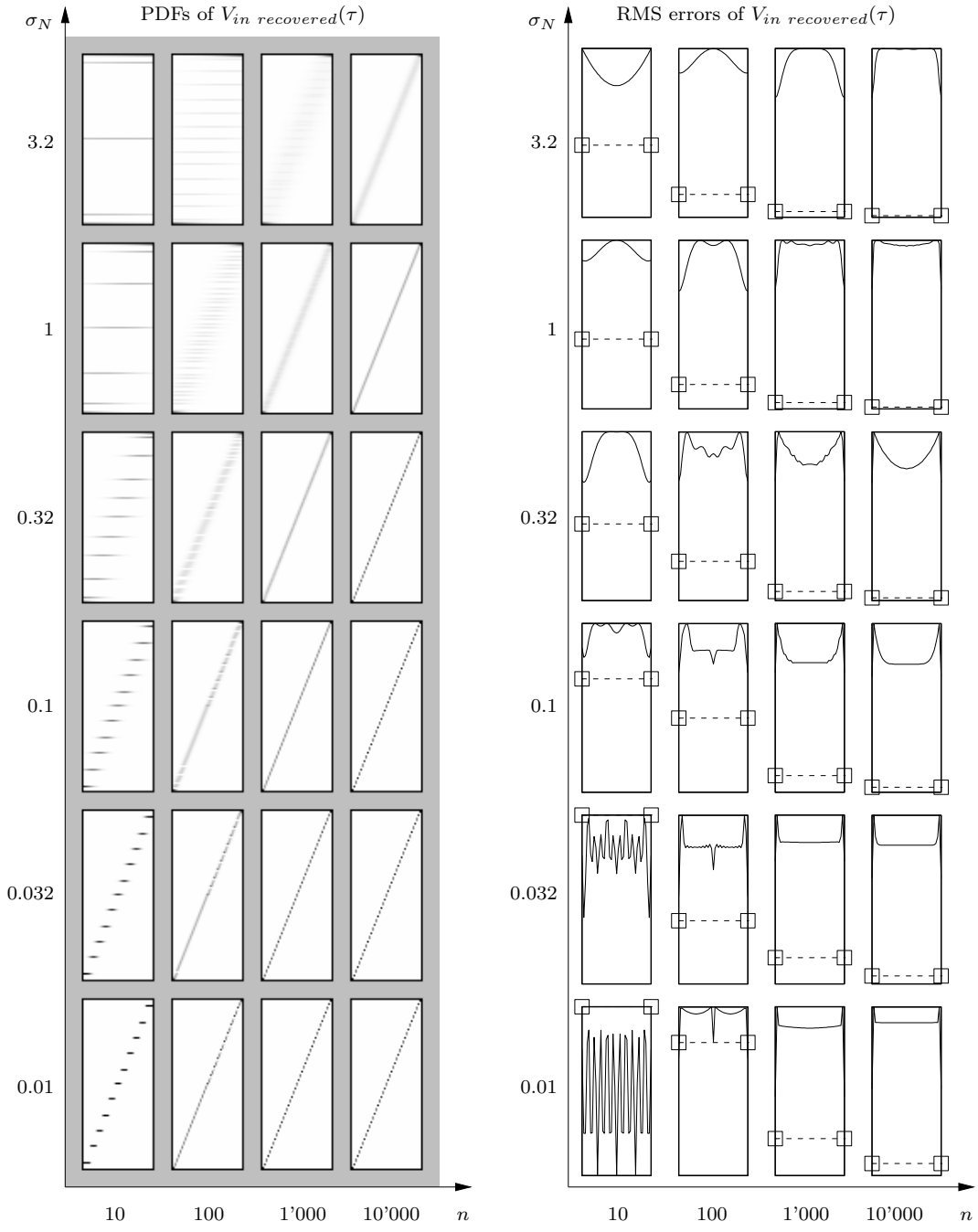

Figure 5.17.: Swept threshold sampler RMS errors

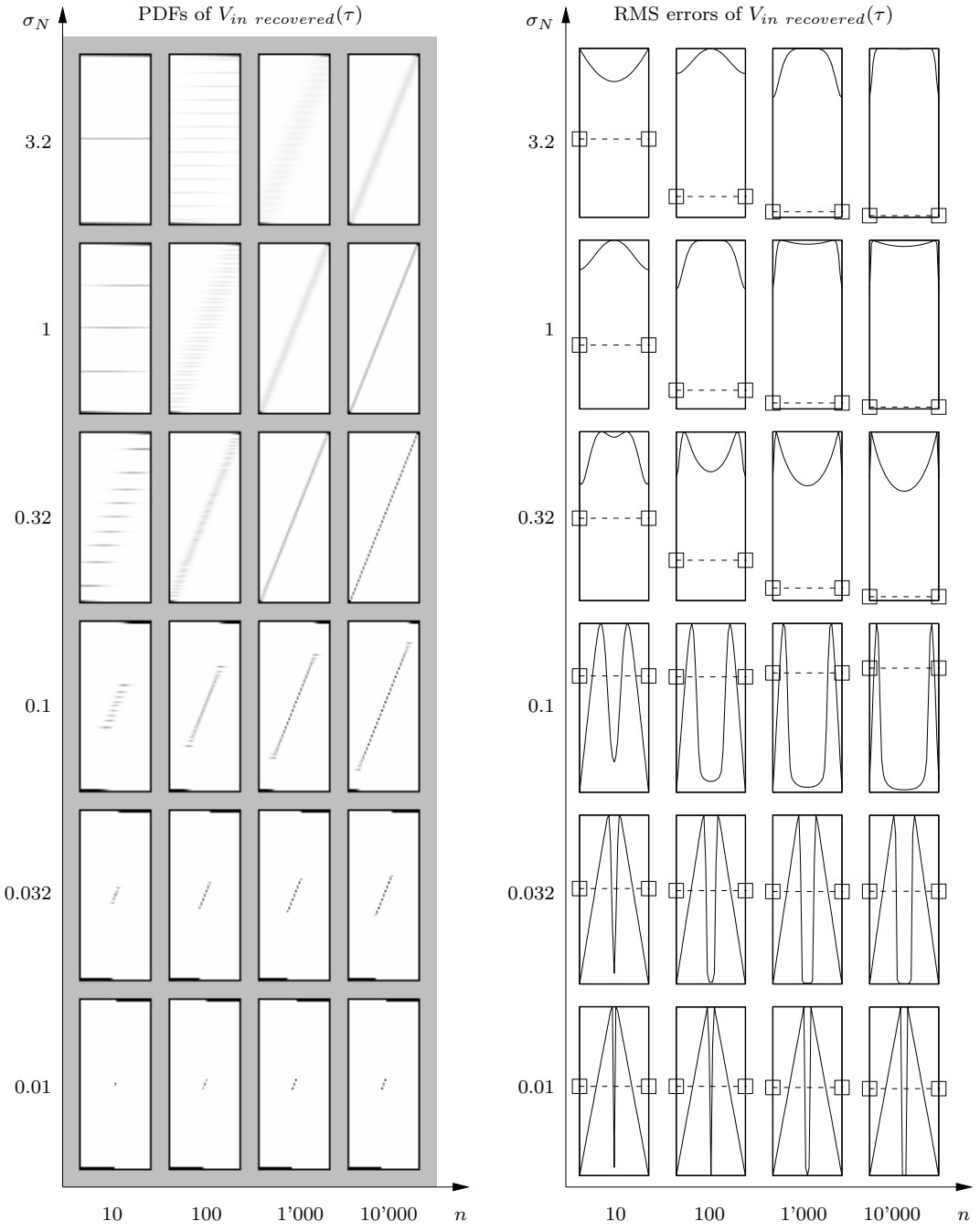

Figure 5.18.: Stochastic resonance sampler RMS errors

5.5.3. Noise in the recovered signal

We now have the RMS error of the recovered signal $V_{in\ recovered}(\tau)$ as a function of n, σ_N and the ideal $V_{in}(\tau)$ (figure 5.17 and 5.18). To evaluate the performance of sampler, however, we are not really interested in the variation of the RMS error over the range of the ideal $V_{in}(\tau)$. A good measurement for the performance of the sampler would simply be the maximum of RMS error over this range. This is what we will now use to evaluate the different samplers, and we will call this value $\sigma_{N\ recovered}$, since it is the worst case standard deviation of $V_{in\ recovered}(\tau)$. As mentioned earlier, it is not exactly the standard deviation, but it is close enough, so we will call it that.

3D plots

$\sigma_{N\ recovered}$ is plotted against n and σ_N in figure 5.19–5.22. The four figures show the three samplers first plotted individually and then together. The axes of all the plots are the same. In addition to the simulated data, which ranges from $n = 1$ to $n = 10'000$, extrapolated data points are also shown for $n = 10^8$. These data points have been made by extrapolating from the data points for $n = 3'000$ and $n = 10'000$. In figure 5.19 (swept threshold), one can see that there are several distinct regions of the surface. Unfortunately, the data point for $n = 10^8$ and $\sigma_N = 10^{-4}$ is extrapolated from two data points near the interface between two regions. Because of this, the data point is probably a bit erroneous.

The reason for the high density of data points for σ_N between about 0.1 and 0.3, is that this is required in order to reveal the details of the stochastic resonance plot in this particular region.

Observations that can be made from the swept threshold plot (figure 5.19) is first of all that $\sigma_{N\ recovered}$ increases when σ_N increases, and that an increase in n causes a decrease in $\sigma_{N\ recovered}$. We can also identify at least three regions of operation. First, in the low n, low σ_N area, we are only limited by the quantization error caused by the low n. As n or σ_N increases, the quantization error catches up with the regular noise, and other effects thus take over as the noise source. The second operating region, is the rightmost part of the plot, is the main operating region of the swept threshold sampler, where the noise is greater than the quantization error and gets averaged with increasing n. In the upper left corner we see a limitation effect on $\sigma_{N\ recovered}$. The noise can not get any worse than 1, which means that for instance an ideal $V_{in}(\tau) = 0$ gets read out as a 1 with a 100% probability. It can not get any worse than this. The limitation effect probably occurs a little lower than this, though.

In the stochastic resonance plot (figure 5.20), we see, as we already know, that we require a minimum level of noise before the sampler behaves properly. Beyond a given noise level, however, the performance starts to deteriorate again. Also in this plot we see a limitation effect on $\sigma_{N\ recovered}$ in the upper left corner.

In the analog average plot (figure 5.21), we simply see the plain surface created by

the noise formula $\sigma_{N\ recovered} = \sigma_N / \sqrt{n}$.

In the plot with all samplers (figure 5.22), we make a remarkable discovery. The swept threshold and stochastic resonance samplers seem to perform almost as well as the analog average sampler when the noise is about $\sigma_N = 1$ and stronger. That means that these relatively simple circuits are almost as good as the mathematically perhaps ideal sampler when there is much noise. A truly surprising result!

Plots of σ_N-slices

Figure 5.23–5.26 shows plots of slices from the 3D plots. Several values of σ_N have been chosen and the 3D plots have been sliced at these values. The resulting 2D plots have then been superimposed.

Figure 5.23 (swept threshold) shows how the swept threshold sampler follows the quantization error limit (also plotted in the graph) when n and σ_N are small. The formula for the quantization error is:

$$QE = \frac{1}{2(n+1)} \tag{5.4}$$

Figure 5.24 (stochastic resonance) is a bit hard to read and does not really tell us much new information.

Figure 5.25 (analog average) is not so interesting either.

Figure 5.26 (all three samplers) shows for $\sigma_N = 1$ how all the three samplers have the same response to an increase in n, and that the swept threshold and stochastic resonance samplers only perform slightly worse than the analog average sampler at this noise level. We also see how the analog average sampler is not error-limited like the other samplers. This is simply because $V_{in\ recovered}(\tau)$ in this sampler does not get constrained to [0, 1].

Plots of n-slices

Figure 5.27–5.30 shows n-slices from the 3D plots.

Figure 5.27 (swept threshold) shows clearly the quantization error limited operating region in the upper left area of the graph. The leftmost data points on the extrapolated graph are, as mentioned earlier, erroneous, and it can clearly be seen here that they are indeed irregular.

Figure 5.28 (stochastic resonance) shows how the stochastic resonance sampler performs best when the noise is about $\sigma_N = 0.3$.

Figure 5.29 (analog average) is again not very interesting.

Figure 5.30 (all three samplers) shows again how the three samplers behave very similarly for $\sigma_N = 1$ or higher. Error limiting and lack thereof for the analog average sampler is also again seen. A new observation is the change in operating region in the swept threshold sampler when $\sigma_N \geq 1$. This is as earlier described the stochastic resonance operating region of the swept threshold sampler, where the threshold levels become so close to each other compared to the noise PDF that they behave like the

stochastic resonance sampler, where the threshold levels are completely squeezed together.

Minimum required n

An interesting perspective on the data from the simulation, is to ask how high n must be in order to achieve a given $\sigma_{N\ recovered}$. This information can be extracted by essentially finding the crossing line between a chosen $\sigma_{N\ recovered}$-plane in the 3D plot and each of the sampler surfaces. The result is shown in figure 5.31. Again we see the good performance of the swept threshold and stochastic resonance samplers for $\sigma_N = 1$ or higher. Figure 5.32 shows a comparison between the samplers. A ratio-line of 1.56 is plotted in. This is the convergence limit, which will be shown later. In high noise cases, it seems swept threshold and stochastic resonance samplers only need somewhere between 1.5 and 2 times the number of samplings which the analog average sampler would need in order to achieve the same $\sigma_{N\ recovered}$.

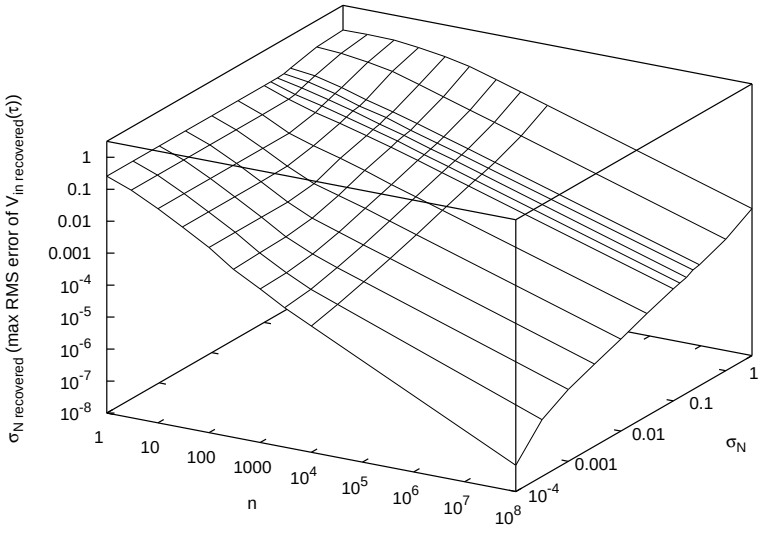

Figure 5.19.: 3D plot of $\sigma_{N\ recovered}$ for the swept threshold sampler

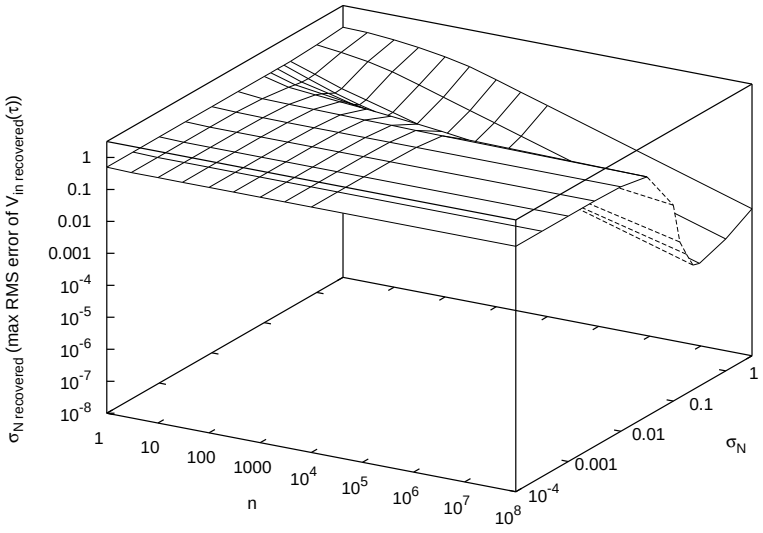

Figure 5.20.: 3D plot of $\sigma_{N\ recovered}$ for the stochastic resonance sampler

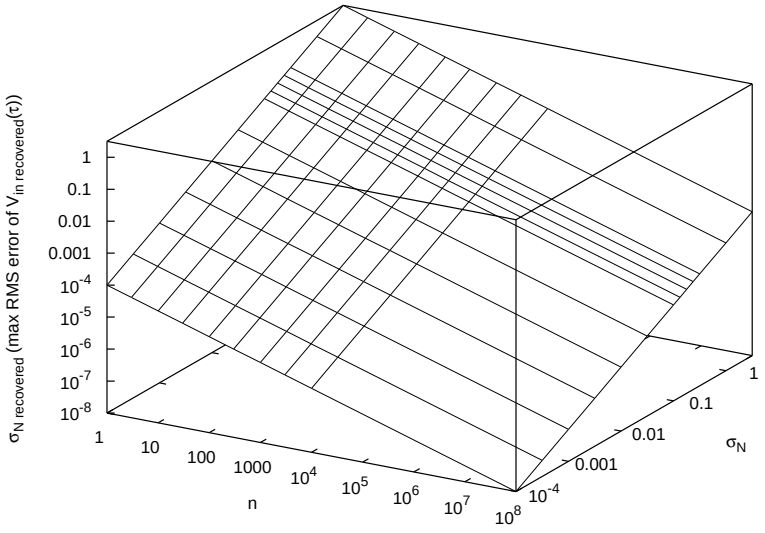

Figure 5.21.: 3D plot of $\sigma_{N\ recovered}$ for the analog average sampler

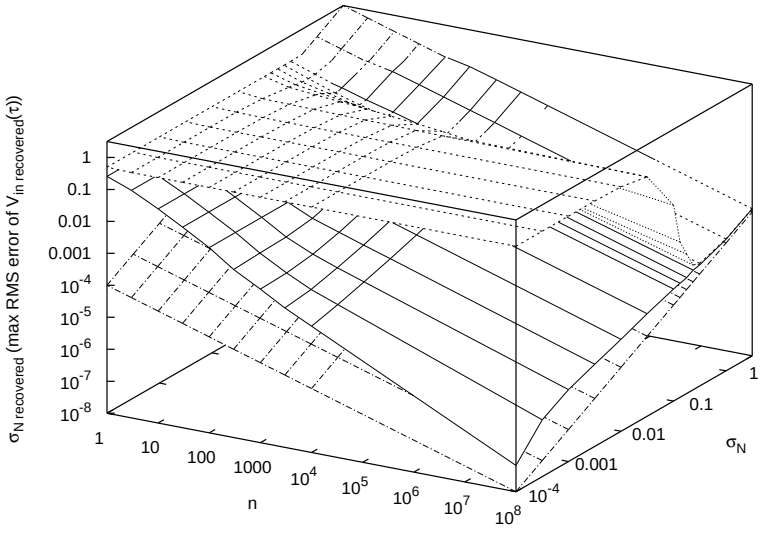

Figure 5.22.: 3D plot of $\sigma_{N\ recovered}$ with all three samplers

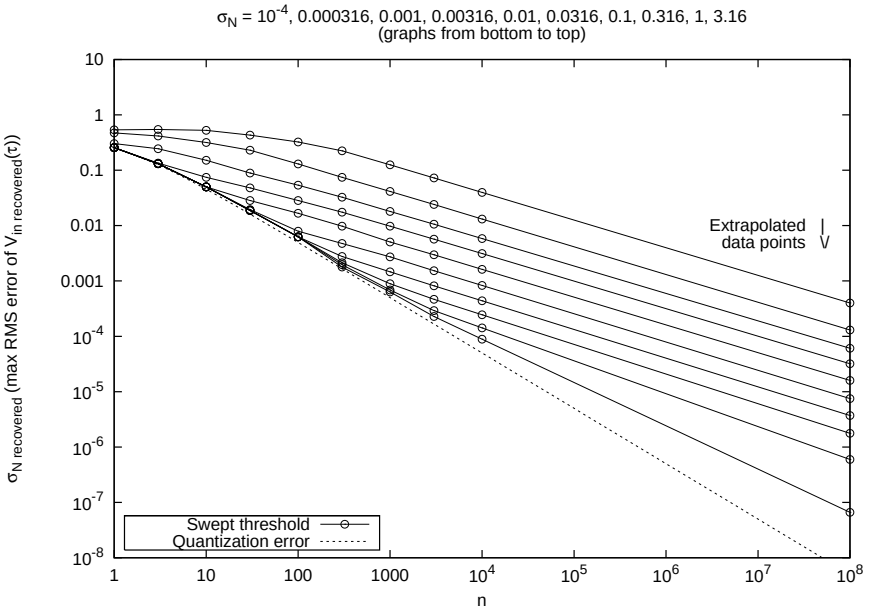

Figure 5.23.: Cross sections of 3D plot (σ_N-slices) for the swept threshold sampler

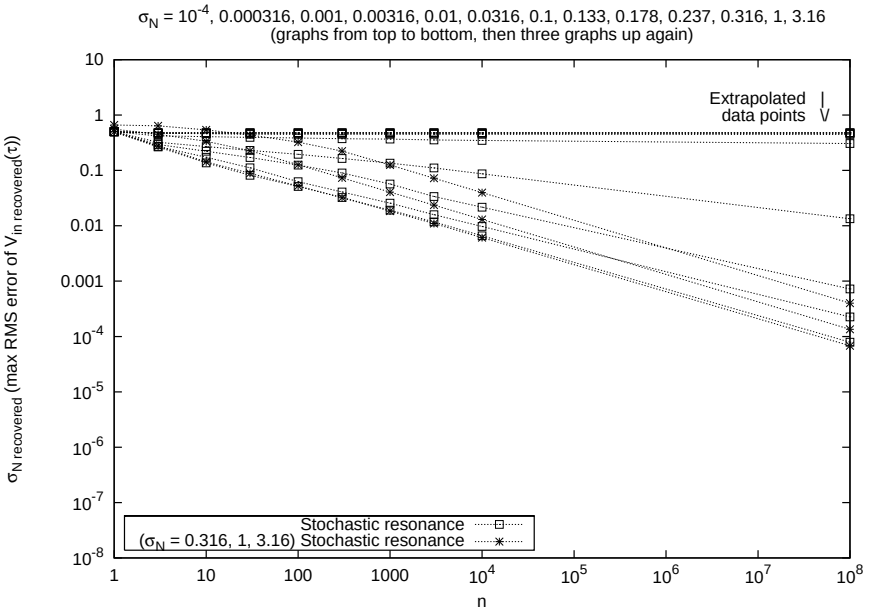

Figure 5.24.: Cross sections of 3D plot (σ_N-slices) for the stochastic resonance sampler

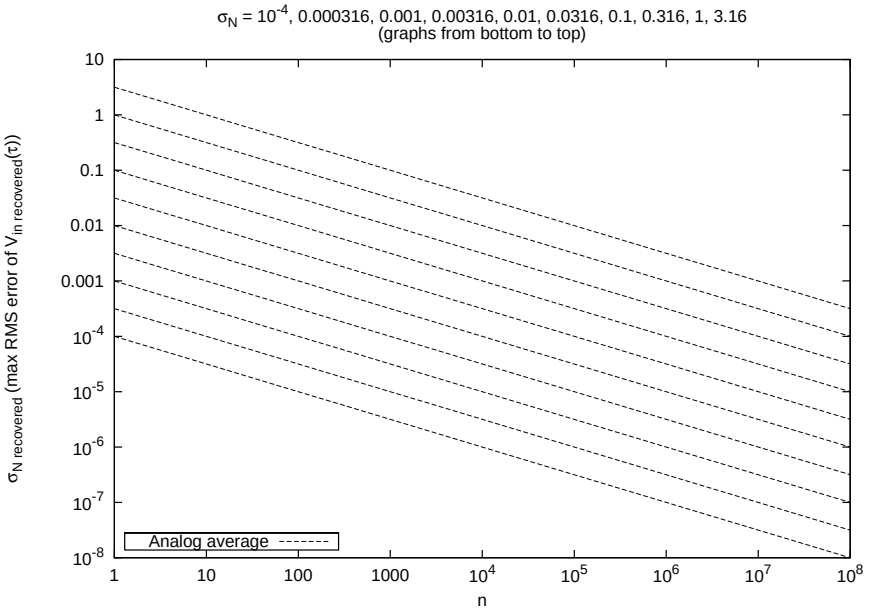

Figure 5.25.: Cross sections of 3D plot (σ_N-slices) for the analog average sampler

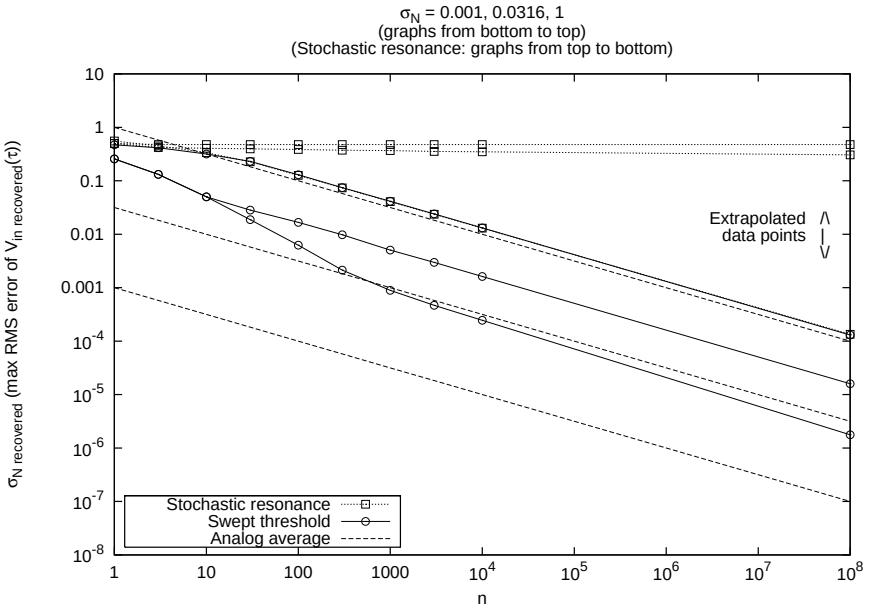

Figure 5.26.: Cross sections of 3D plot (σ_N-slices) with all three samplers

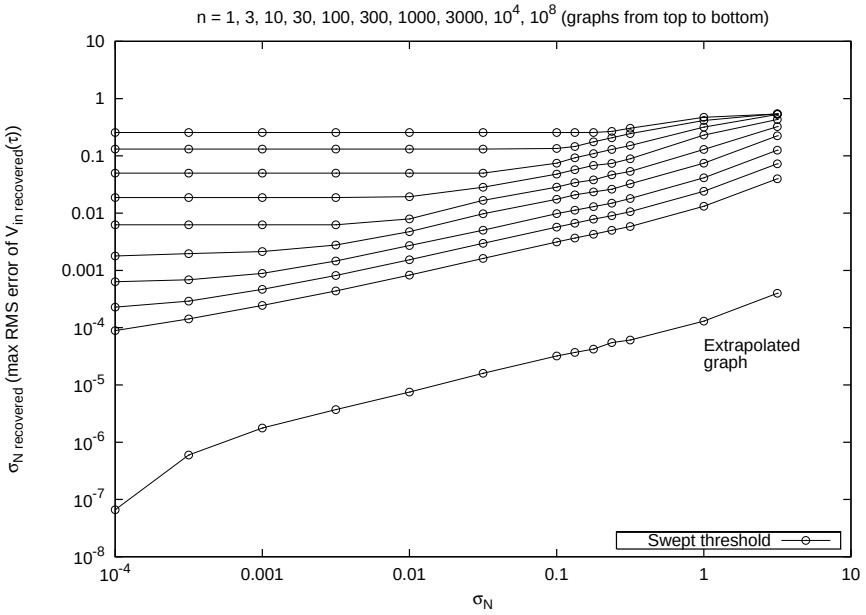

Figure 5.27.: Cross sections of 3D plot (n-slices) for the swept threshold sampler

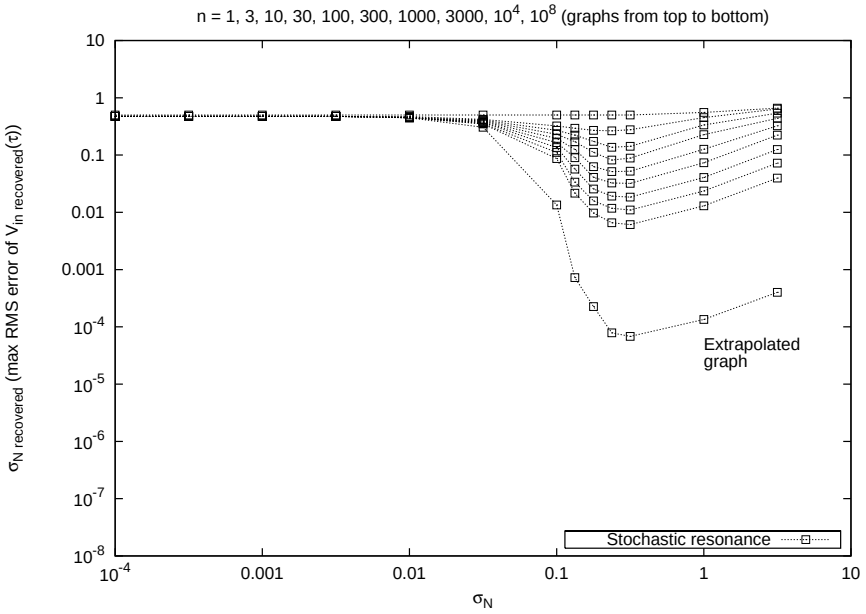

Figure 5.28.: Cross sections of 3D plot (n-slices) for the stochastic resonance sampler

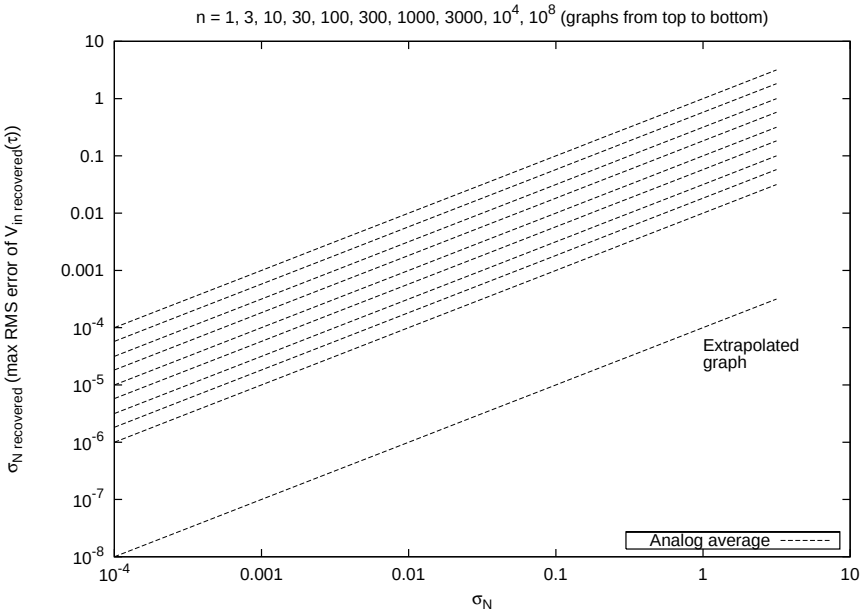

Figure 5.29.: Cross sections of 3D plot (*n*-slices) for the analog average sampler

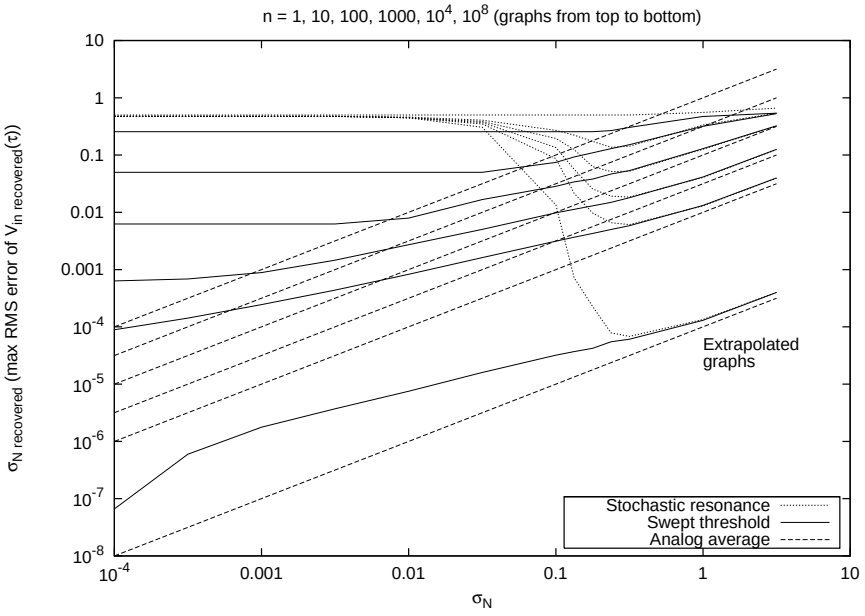

Figure 5.30.: Cross sections of 3D plot (*n*-slices) with all three samplers

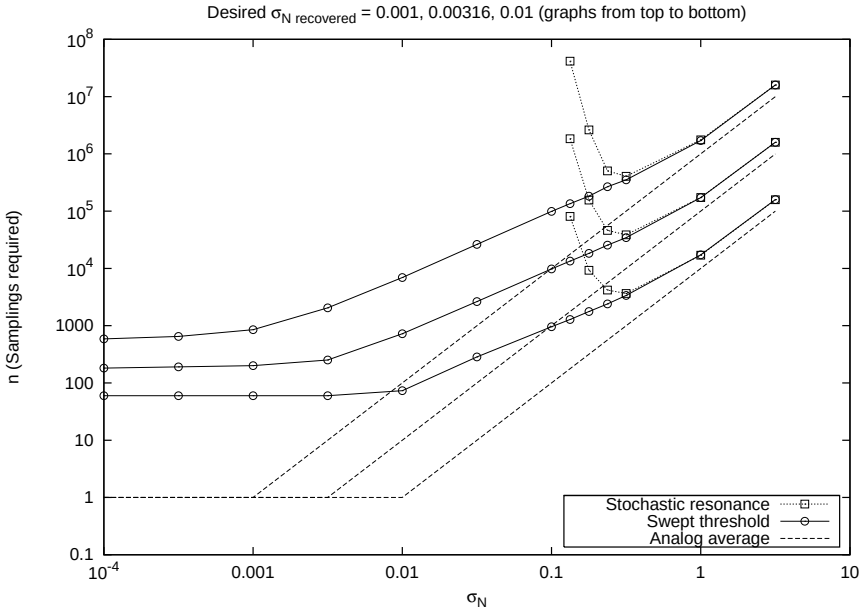

Figure 5.31.: Samplings (n) required to achieve a given $\sigma_{N\ recovered}$

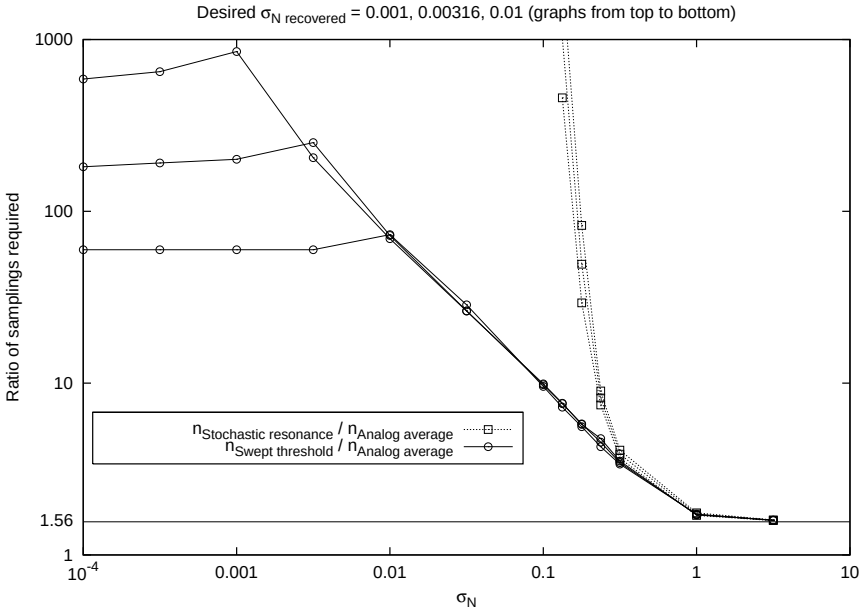

Figure 5.32.: Samplings (n) required: Comparison between the samplers

5.6. Analytic results

The following Octave functions produce the results shown in figure 5.33 to 5.39. Due to time constraints, further elaboration are not included.

```
function ANrms = analytic_ST(n, sigma_N)
comp = analytic_ST_components(n, sigma_N);
ANrms = max(max(comp.QE, comp.ST), comp.SR);
ANrms = min(ANrms, comp.QEmax);

function comp = analytic_ST_components(n, sigma_N)
k = 0.1 / (normal_cdf(0.1) - 0.5);
comp.QE = ones(length(sigma_N), 1) * (0.5 ./ (n+1));
comp.ST = sqrt( 1.5*0.5*sigma_N' * (1 ./ (n+1)) );
comp.SR = k * sigma_N' * (1 ./ (2 * sqrt(n)));
comp.QEmax = ones(length(sigma_N), length(n)) * 0.5;
```

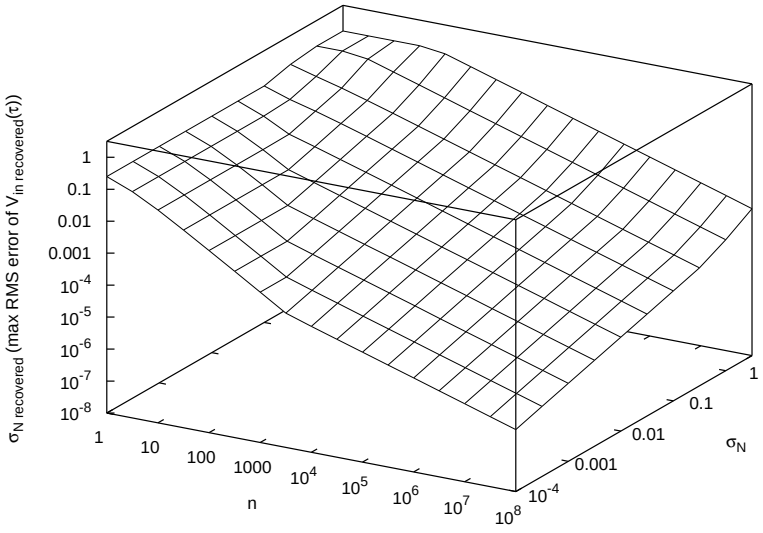

Figure 5.33.: 3D plot of estimated $\sigma_{N\ recovered}$ for the swept threshold sampler

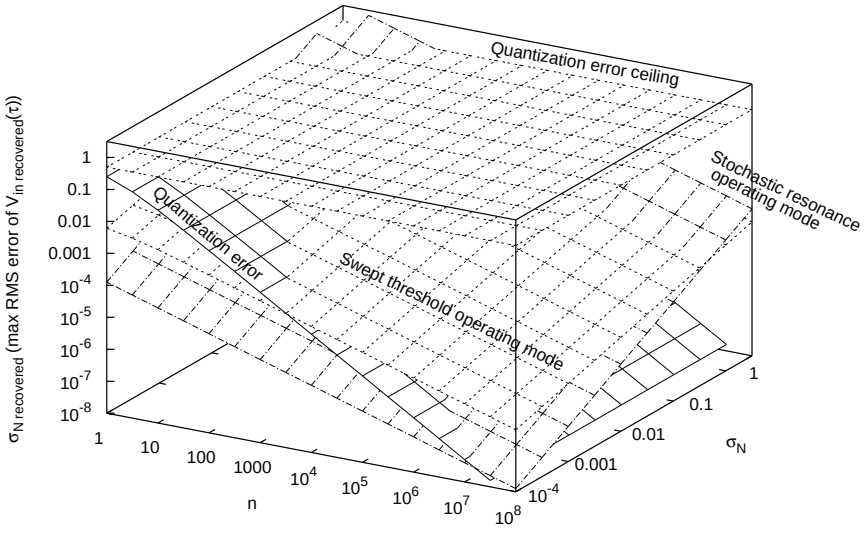

Figure 5.34.: 3D plot of the components of the $\sigma_{N\ recovered}$ estimation formula

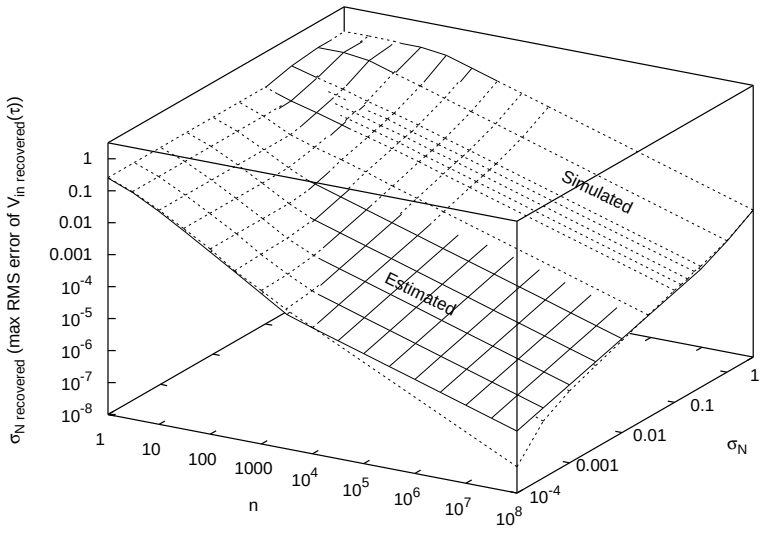

Figure 5.35.: 3D plot of simulated and estimated $\sigma_{N\ recovered}$ for the swept threshold sampler

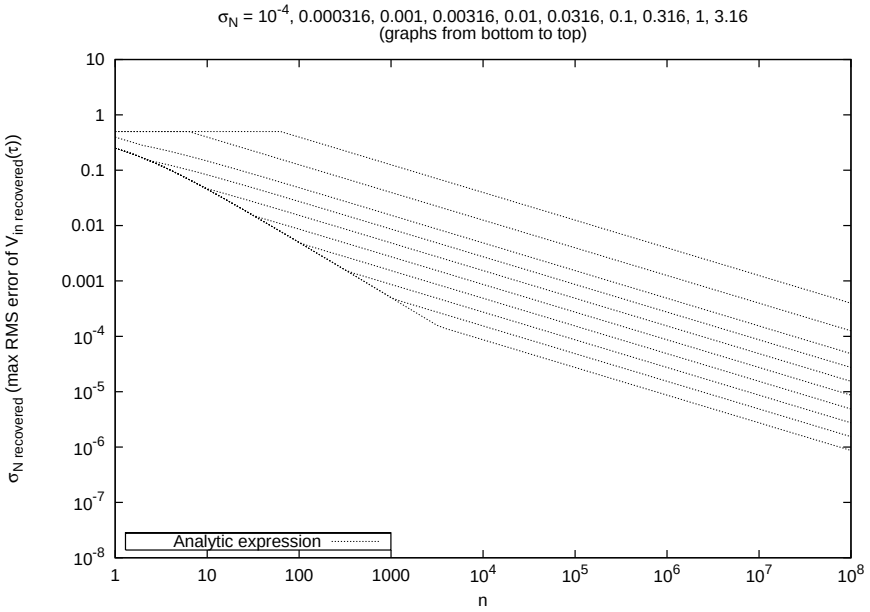

Figure 5.36.: Cross sections of estimated $\sigma_{N\ recovered}$ 3D plot (σ_N-slices)

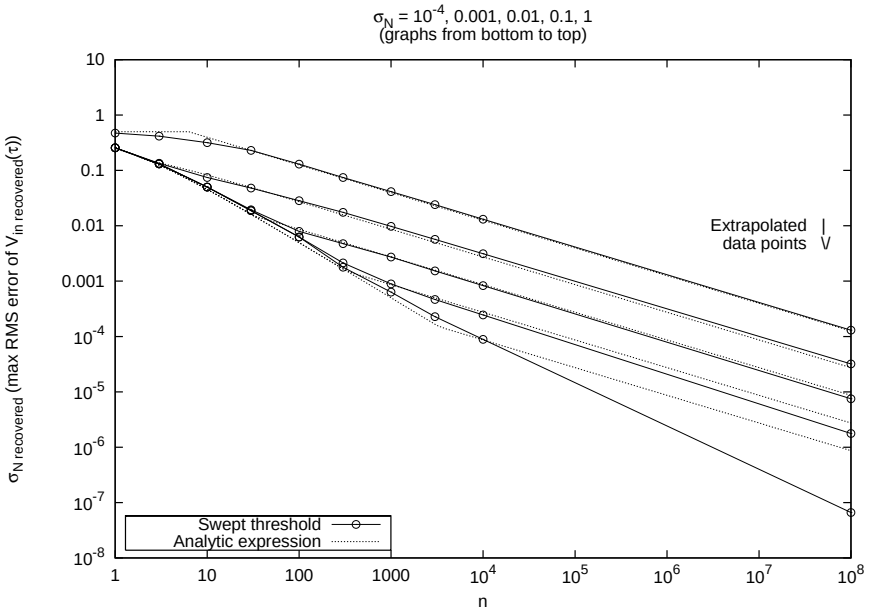

Figure 5.37.: Comparison between estimated and simulated $\sigma_{N\ recovered}$ (σ_N-slices)

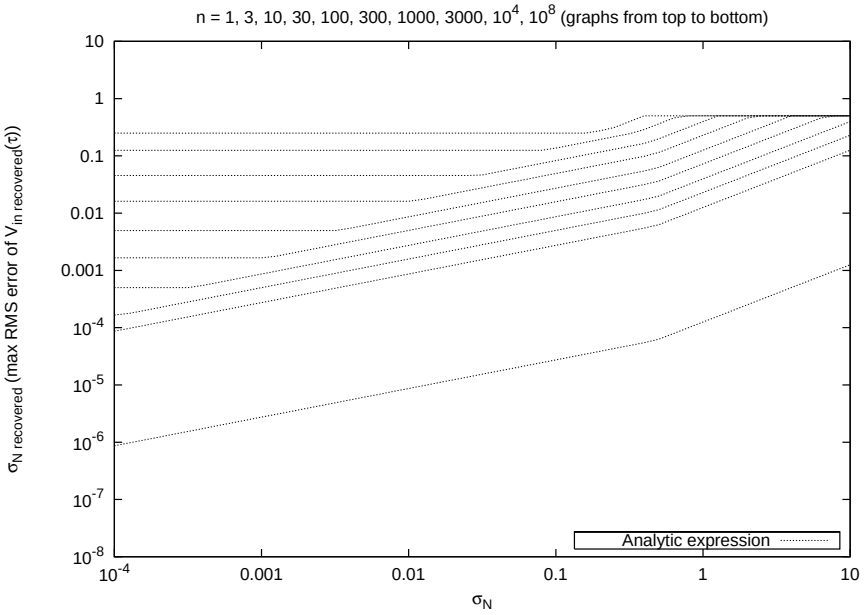

Figure 5.38.: Cross sections of estimated $\sigma_{N\ recovered}$ 3D plot (n-slices)

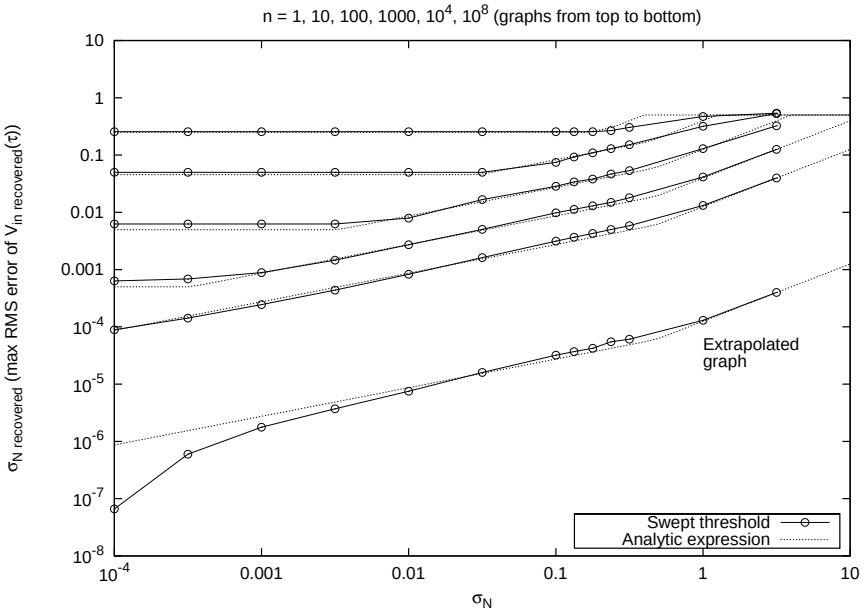

Figure 5.39.: Comparison between estimated and simulated $\sigma_{N\ recovered}$ (n-slices)

6. Implementation

A radar using the swept threshold / stochastic resonance sampler described in section 5.4 has been implemented in 90 nm CMOS. In this chapter we will describe the implemented circuit and measurements made of the circuit.

6.1. Circuit

A block diagram of the circuit is shown in figure 6.1. As the basic swept threshold / stochastic resonance sampler, this circuit also sends a radar pulse, then waits a given time τ, and then samples the thresholded input signal. The most significant additions, is that we are now using an LNA to amplify the signal before the thresholder and that we are taking 64 samples per sent pulse rather than just one. By using an LNA, the demands on the thresholder in terms of noise factor is reduced, and the thresholder can be designed without that being a difficult constraint to be dealt with. And by taking 64 samples per transmitted pulse, we get much more information from each pulse, which saves us both time and energy compared to acquiring the 64 samples one by one. The extra circuitry required to do this is not very big, we essentially just need a digital sampler and a counter for each new sample to be taken.

There are also other features of interest in the circuit. The first is the pulse generator, which is tunable, that is, it is capable of transmitting four slightly different pulses. Secondly, the initial delay, τ, is digitally tunable with 12 configuration bits. Also, the digital buffer which sends the thresholded signal to the 64$\times$sampler has a override functionality for debugging. Furthermore, this buffer is only enabled when a signal is about to be sampled, thus conserving energy. Most digital interfacing is done through a serial digital interface, based on Serial Peripheral Interface Bus (SPI). This conserves the pin count drastically.

The LNA was made by Kjetil Meisal, and the work was sponsored by Novelda, so its design will not be covered in this master thesis. The thresholder was created together with Claus Limbodal, also this work sponsored by Novelda, and will be discussed in this chapter. The pulse generator was created by Håvard Moen, and is described in [Moen 06].

The total radar sampler circuit consists of 30 schematic cells, so it is a bit complex, but luckily most of the cells are purely digital, thus reducing the complexity somewhat. An overview of the schematic cells is shown in figure 6.2. This does not include the pulse

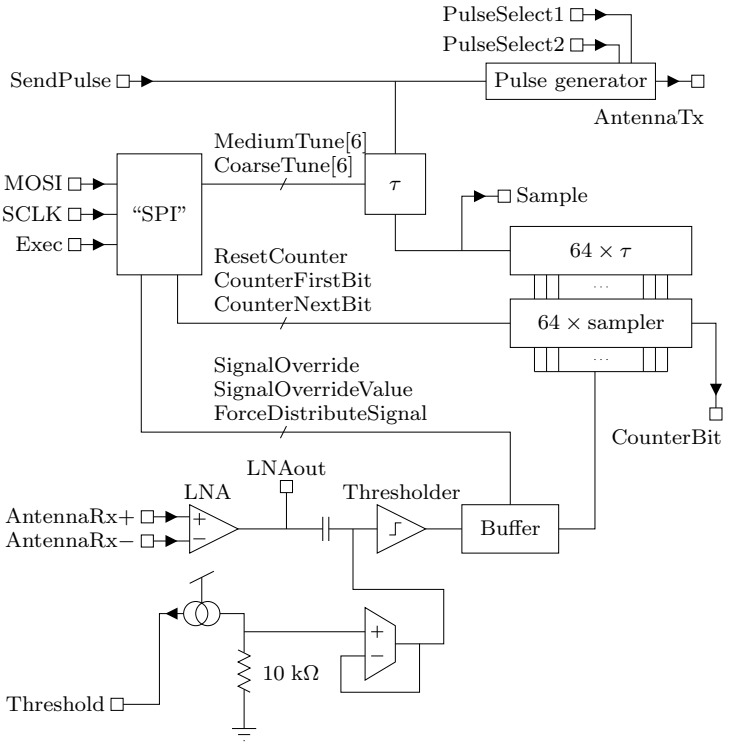

Figure 6.1.: Block diagram of circuit

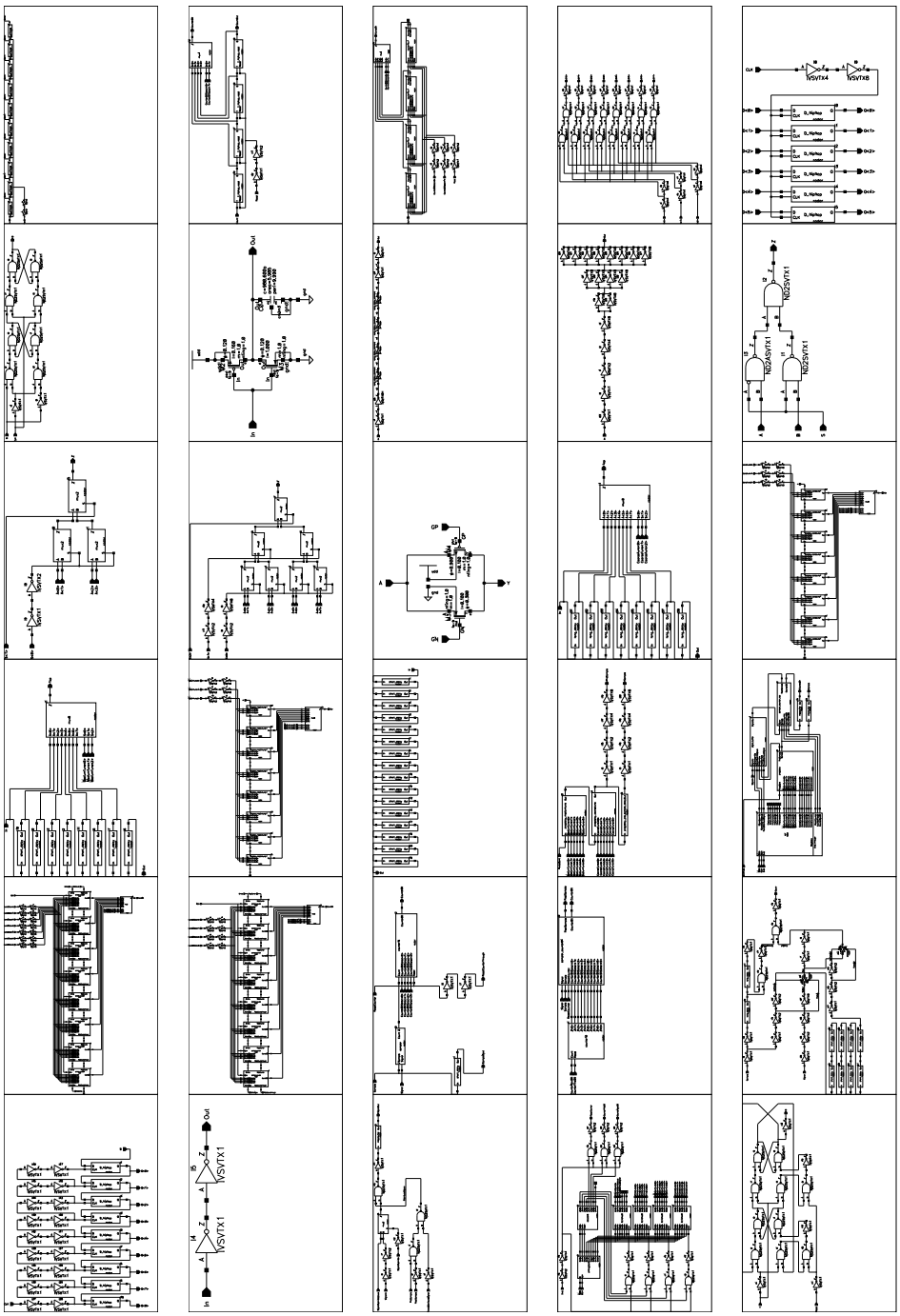

Figure 6.2.: The radar sampler circuit consists of 30 schematic cells, mostly digital

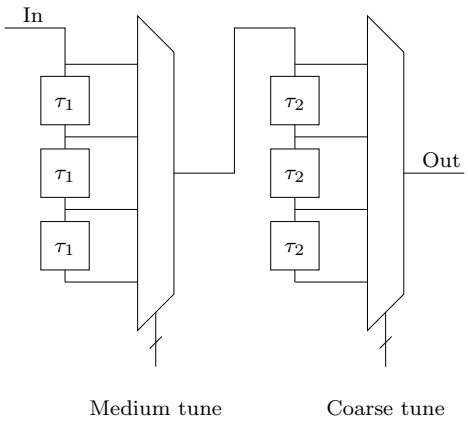

Figure 6.3.: Programmable delay

generator, thresholder or LNA.

6.1.1. Serial digital interface

The serial digital interface is described in the data sheet in appendix A. Its purpose is to provide registers and signaling wires to control the rest of the radar sampler circuit.

6.1.2. Pulse generator

The pulse generator has two configuration bit inputs which allows selection of different pulses to be transmitted. This is covered in [Moen 06]. The input bits are pins on the chip, i.e. they are not controlled by the serial digital interface. The output is single ended and can be connected to for instance a 50 Ω coaxial cable.

6.1.3. Programmable initial delay

The programmable initial delay, τ, is controlled by two 6-bit registers in the serial digital interface. Each register controls a multiplexer which taps a delay line, thus allowing us to choose how many unit delays of that delay line we want. We can cascade two such tapped delay lines, one with short delays and one with longer delays, where the long delays are equal to about half the maximum delay of the first delay line (figure 6.3). This way, we are able to create a programmable delay with high precision and wide range. We effectively get 11 bits of freedom, allowing a selectable delay from 0 to about 2000 unit delays.

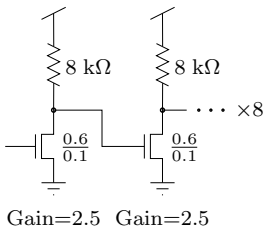

Figure 6.4.: Common source amplifier

6.1.4. LNA

The LNA has differential input and single ended output. Input impedance is 50 Ω in the 3.1–10.6 GHz range. Created by Kjetil Meisal, so not described in detail here.

6.1.5. Thresholder

The thresholder is implemented as a high gain amplifier with a constant operating point at about Vdd/2 and with a level shifter in front, as in [Meis 05]. The threshold level is set by the level shifter. In this implementation, we have chosen to use a common source amplifier instead of an inverter based solution as in the paper, though.

The level shifter works by pulling the right side of the capacitor in figure 6.1 to a given voltage. If, for instance, the amplifier operating point is 0.5 V and its input is pulled to 0.4 V by the transconductance amplifier, the input signal has to have a spike of at least 0.1 V in order to cross the threshold. Thus, the threshold is in this case 0.1 V. The input to the transconductance amplifier is set with a current being pulled from a pin. This is in order to improve the signal quality. The level shifter also works as a high pass filter with a cut-off frequency of about 1 GHz.

A typical configuration with a 5 kΩ resistor between the pin and a voltage source V_{DAC} has an approximate transfer function $V_{res} = 1.118 - 1.427V_{DAC}$ between the input voltage and the voltage over the internal 10 kΩ resistor. The transfer function is relatively linear for $V_{DAC} \in [250, 650]$ mV. The V_{res} range will then be $[190, 761]$ mV. The actual threshold voltage is the difference between the resistor voltage and the operating point of the thresholding element, $V_T = V_{op} - V_{res}$. With $V_{op} \approx 0.5$ V:

$$V_T = -0.618 + 1.427V_{DAC}$$

The common source amplifier used in this design is shown in figure 6.4. It consists of eight cascaded common source amplifiers, each with a gain of about 2.5. This results in a total gain of $2.5^8 \approx 1500$. The reason for choosing a cascade of several low-gain amplifiers instead of a few high-gain amplifiers, is that we want a flat gain all the way from

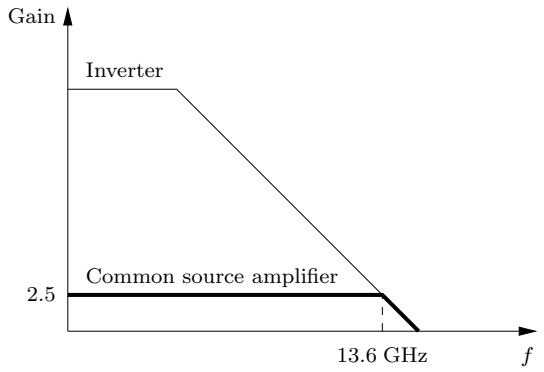

Figure 6.5.: Frequency response of inverter and common source amplifier

DC to our highest frequency of interest. Since the gain bandwidth product is constant, we can increase the bandwidth of an amplifier by reducing the gain (figure 6.5).

The reason for the flat gain being so important in an amplifier which is going to clip anyway, is that lower frequency signals, including DC, could mask out interesting high frequency signals. Assume for instance that we have an inverter with an operating point of 500 mV, a low bandwidth, but a high gain. If the DC level of the input signal is 499 mV, the output could be for instance $-1\text{ mV} \cdot 5000 = -5\text{ V}$, which would of course clip. If a high frequency spike with an amplitude of 2 mV then occurs, which of course crosses the threshold theoretically, it might be amplified to for instance $2\text{ mV} \cdot 1000 = 2\text{ V}$. The total output from the amplifier, the sum of the DC and the spike, will then be $-5\text{ V} + 2\text{ V} = -3\text{ V}$, which still clips to ground. What this means, is that the DC signal was so strongly amplified that it masked out the high frequency signal which was less amplified. So to compensate this, we have to reduce the high gain of the low frequency components, i.e. create a flat gain. This technique for increasing the bandwidth of an amplifier is not new, but lesson to learn is that when creating a clipping amplifier, it is easy to think that a little extra gain at any frequency could not hurt, since we are clipping anyway. This is, however, as shown, wrong. Extra gain could cause masking.

With the first resistors in the cascade having a voltage drop of about Vdd/2 and the last resistors having voltage drops of alternatingly 0 and Vdd, the average voltage drop is Vdd/2. With Vdd$=$ 1 V, this means that the common source amplifier cascade consumes $8 \cdot 0.5\text{ V}/8\text{ k}\Omega = 0.5\text{ mA}$ statically, and probably nearly the same when switching fast too. 0.5 mA might not be much in some applications, but for other, low power applications, it could be a considerable power drain. This is, however, the price we pay for using a class A amplifier. Note that an inverter amplifier would not be very much better, since the first ones in the cascade here too would conduct a short circuit current, i.e. the first inverters in the cascade would behave as class A amplifiers.

The RMS voltage of the thermal noise in the first resistor in the common source amplifier cascade will be:

$$\begin{aligned} v_n &= \sqrt{4k_BTR\Delta f} \\ v_n &= \sqrt{4 \cdot 1.38 \cdot 10^{-23} \cdot 290 \cdot 8\ \text{k} \cdot 10\ \text{G}} \\ &= 1.13\ \text{mV} \end{aligned}$$

To make the noise input referred, we have to divide with the gain: $v_n' = v_n/2.5 = 1.13\ \text{mV}/2.5 = 0.45\ \text{mV}$ This is just a rough estimate of the noise, since the resistor is of course connected in a circuit, which changes the conditions, but it gives us an idea of the order of magnitude of the noise. And in addition to the thermal noise, there might also be other forms of noise that contribute.

Since the thresholder is just a support element for the radar sampler which we intended to test in this master thesis, we have not put very much work into it. There might very well be better solutions available in terms of both power consumption and noise.

6.1.6. Buffer

The buffer between the thresholder and 64× sampler is necessary in order to drive the 64 parallel samplers which are all connected to the same line. To allow for short pulses to propagate through the circuits, the buffer, and the rest of the system too, tries to use only a factor two fan-out. This means the buffer consists of several steps, doubling the total transistor width with each step.

The buffer has a signal override feature which can be used to override the input from the thresholder. This can be used for debugging purposes.

The buffer is only enabled when a signal is about to be sampled, starting a short while before the sampling sequence starts, and ending right after it is done. This is in order to try to save power. In case of failure of this system, the buffer can be forced to be enabled at all times.

6.1.7. 64× sampler

The 64× sampler consists of a delay line, with 64 taps, and 64 samplers, each connected to a 16 bit counter. All the 64 samplers have the same input, namely the output from the buffer after the thresholder. When a radar pulse is sent out, the initial delay element τ, is triggered, and after a while, its output activates. This in turn triggers the 64-tap delay line in the 64× sampler. When each of the taps on the delay line triggers, the corresponding sampler samples the digital signal from the buffer. This means the sample rate is determined by the unit delay in the delay line, i.e. how long it takes from one tap triggers until the next triggers.

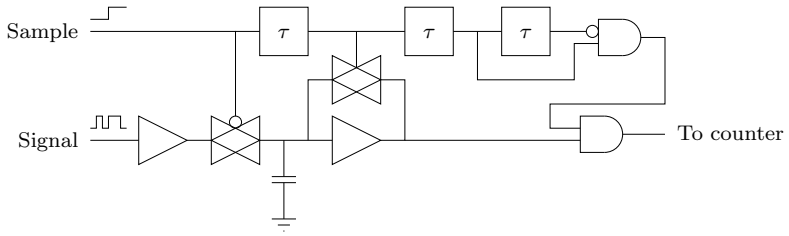

Figure 6.6.: Sampler

The 64 samplers are principally just simple digital samplers and could be for instance D-flip-flops. But since the input signal can be so rapidly changing, it might be that it is in the middle of a transition when the sampler is supposed to acquire its value. Because of this, we use a slightly more elaborate design (figure 6.6). When the sample signal goes high, the first transmission gate stops conducting, and the voltage of the input signal is thus stored on the capacitor. After a short delay, a feedback transmission gate starts to conduct, which ensures that the charge on the capacitor settles on either 0 or Vdd. So far, we essentially have a sample-threshold-and-hold circuit. It should be much less susceptible than a regular D-flip-flop to problems caused by samplings being made in the middle of input signal transitions. After the settle-feedback step, the signal should be purely digital, without any metastability which could propagate further into the system. A short time after this, a one shot pulse is generated and AND-ed with the sampled digital signal. If a "1" was sampled, a one-shot pulses is thus sent to the counter, incrementing it by one. If a "0" was sampled, no pulse is sent to the counter, and so, in that case, it does not get incremented.

The counters can be reset and the counter values can be read out with the serial digital interface. The $64 \cdot 16$ counter bits can be read out in sequence through a mux which is controlled by a counter which in turn can be stepped through by the serial digital interface.

6.1.8. Layout

Since the circuit is mostly digital, it is not as susceptible to interference as analog designs, but basic precautions have still been taken in the form of guard rings around the different sub-circuits. In particular, delay lines are enclosed in their own guard rings, since these have a somewhat analog nature. That is, interference could cause jitter in the delays.

The layout of the entire system is shown in figure 6.7, and a photomicrograph of the die is shown in figure 6.8. The total die size is about 1.2×1.2 mm.

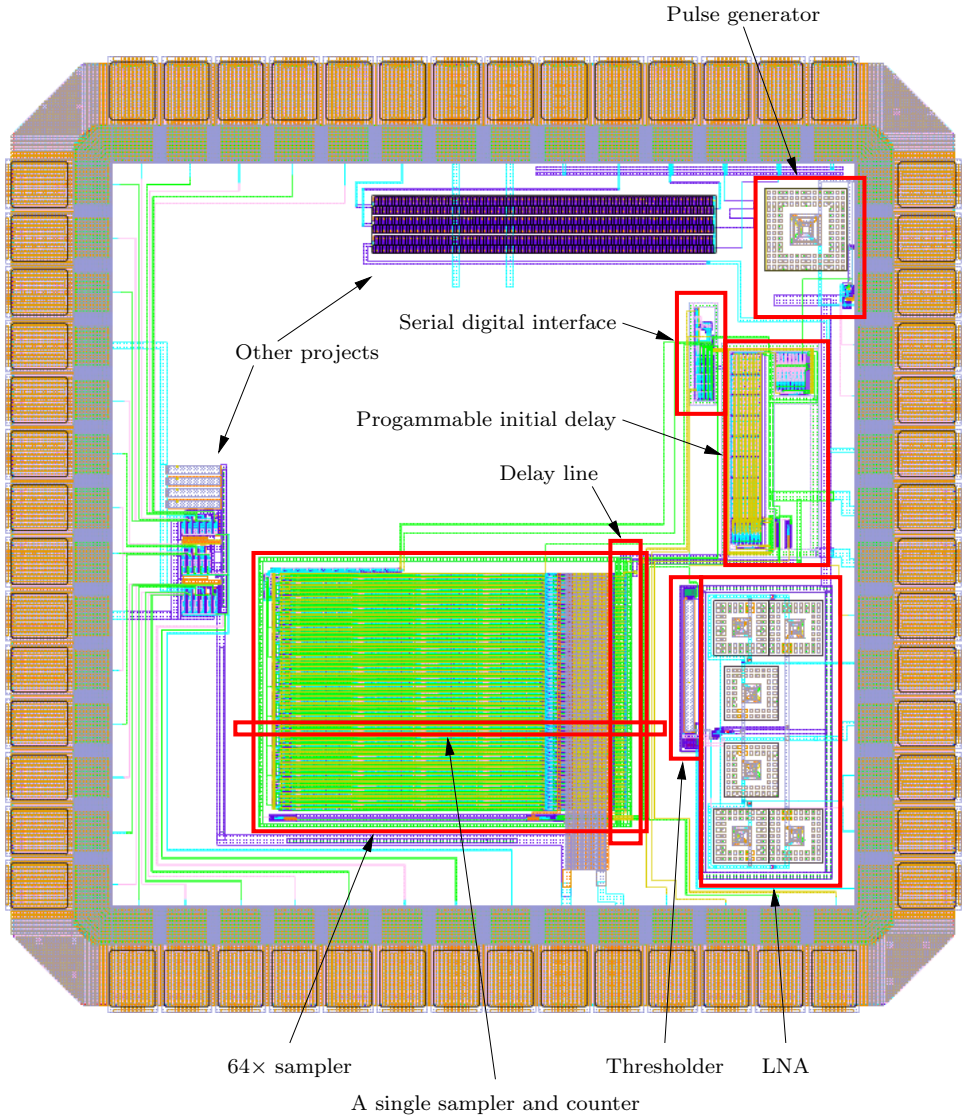

Figure 6.7.: Layout

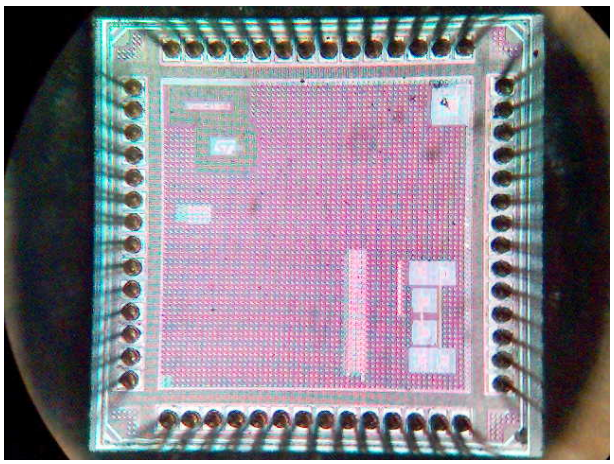

Figure 6.8.: Photomicrograph of chip die

6.2. Measurements

The measurement setup is shown in figure 6.9. Note that the antennas are differential, but are being fed with single ended coaxial cables. This is not a correct combination, but has been sufficient for our measurements.

The measurements were done using a USB to digital IO converter connected to a computer. The IO pins on this converter were used to interface to the serial digital interface of the radar chip. The radar was powered by the USB bus through the converter module.

The measurements made with swept threshold and stochastic resonance sampling are very simple, and do not map the counter values back to $V_{in\ recovered}$ with a transfer function.

6.2.1. LNA

The LNA turned out to be oscillating, so it could not be used. Instead, the LNAout pin, originally intended for measurement of the LNA, was used as input instead. To make the LNA quiet, one of its inputs, which were externally AC-coupled, were tied to a rail. It could not simply be powered down because it was connected to the same power supply pin as the thresholder.

A theory of why the LNA was oscillating, is that the LNAout pin, which is very close to both the LNA input pins, could be coupling into these, thus causing a positive feedback. Especially the bond wire is a potential culprit here. A big package with long bond wire stretches and thin, close, bond wires could cause strong coupling between

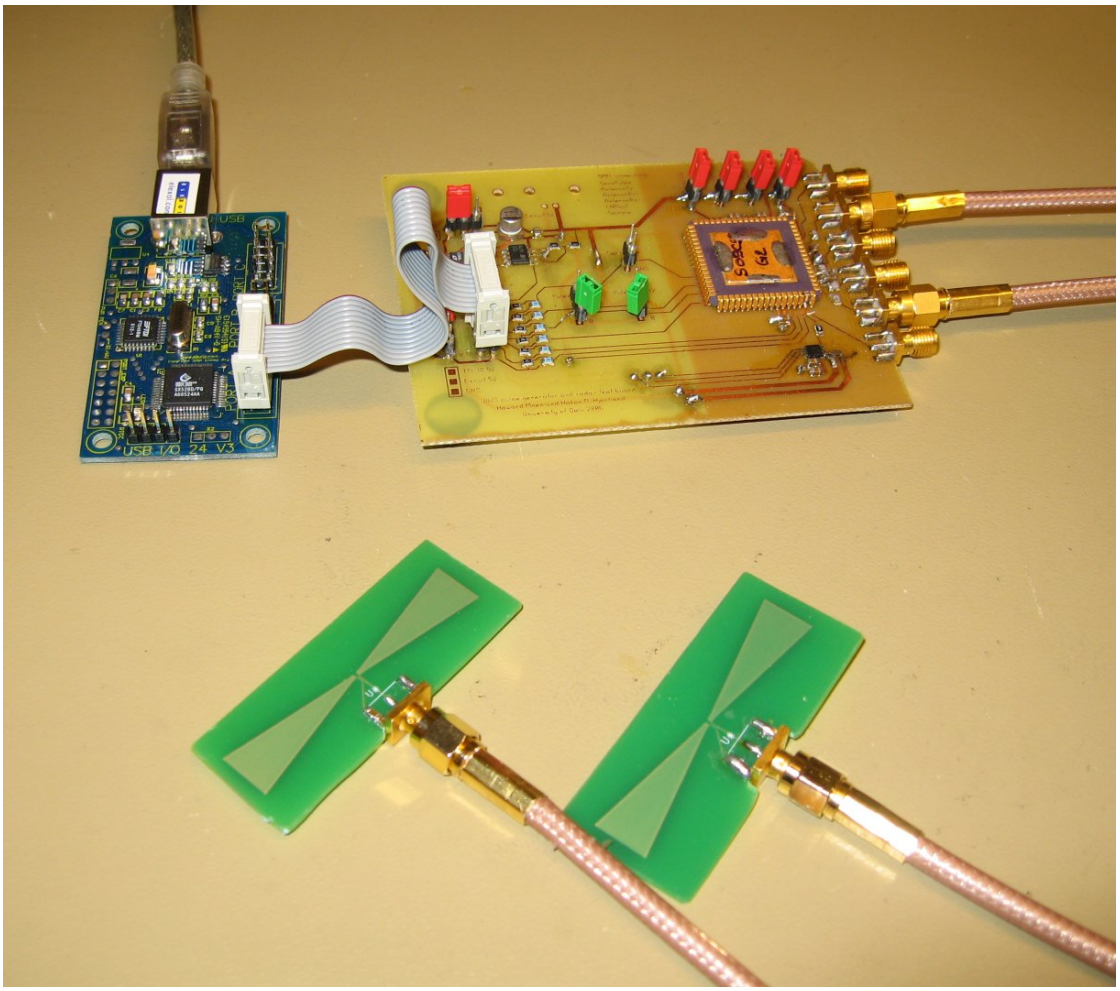

Figure 6.9.: Measurement setup

adjacent bond wires at high frequencies.

A possible fix for this problem would be to remove the LNAout bonding wire. That can be done because it is not needed for normal operation of the radar. This delicate operation has not yet been attempted due to limitation of time.

6.2.2. Radar readout with different pulses

Figure 6.10 shows measurements with the radar sampler in swept threshold mode with no pulse and the four different pulses which the pulse generator can produce. A loop cable connects the pulse generator and the LNAout pin. Each gray scale image consists of 256 horizontal rows. Each of these rows is a full readout from the sampler, namely the values from the 64 counters, i.e. the 64 samplers. Each row in one of the gray scale images corresponds to a given initial delay, τ. The top row is for $\tau = 0$, and then each following row is an increment of one unit delay. Dark shades of gray represent a low $V_{in\ recovered}$, and light shades of gray represents high values.

What we see in the gray scale images with pulses, are black lines at about 45°. This is the pulse that has gone through the cable and then has been sampled. The 45° slope simply means that when we increase the initial delay, the pulse appears earlier in the 64-sample sequence. Since the slope is quite close to exactly 45°, it means that the sampling interval and the τ unit delays are almost exactly equal. Any difference between the two delays would cause the slope to get a different angle.

Since the initial delay circuit is made up of two cascaded tapped delay lines, one with short unit delays and one with long unit delays, we have to find out how many short delays corresponds to one long delay in order to be able to sweep τ smoothly. This has been found by trial and error, and the factor between the two is 32. That is, 32 short delays equal one long delay.

When there is no pulse, we can still see some patterns in the gray scale image. This measurement is actually done by simply removing the loop cable since the pulse generator can not be turned off. The pin that triggers the pulse generator is the same that triggers the initial delay element which in turn triggers the sampling. The patterns we see in the upper left corner is therefore probably direct coupling between the pulse generator and the input pin, probably either between the bonding wires, or between wires on the PCB. We also see vertical stripes. This is interference noise that is correlated to the start of sampling sequence, not the transmission of the pulse. A possible explanation for this, is the "sample" output pin which was meant to be a measuring point for characterizing the initial delay circuit. It raises when the initial delay output raises, and is thus correlated to the start of the sampling sequence. It was discovered that ripping off the pin helped alleviate the problem somewhat. All measurements in this chapter were made with with the pint off. A further improvement would probably be to remove the bonding wire too, but this is a delicate operation, and has not been attempted yet.

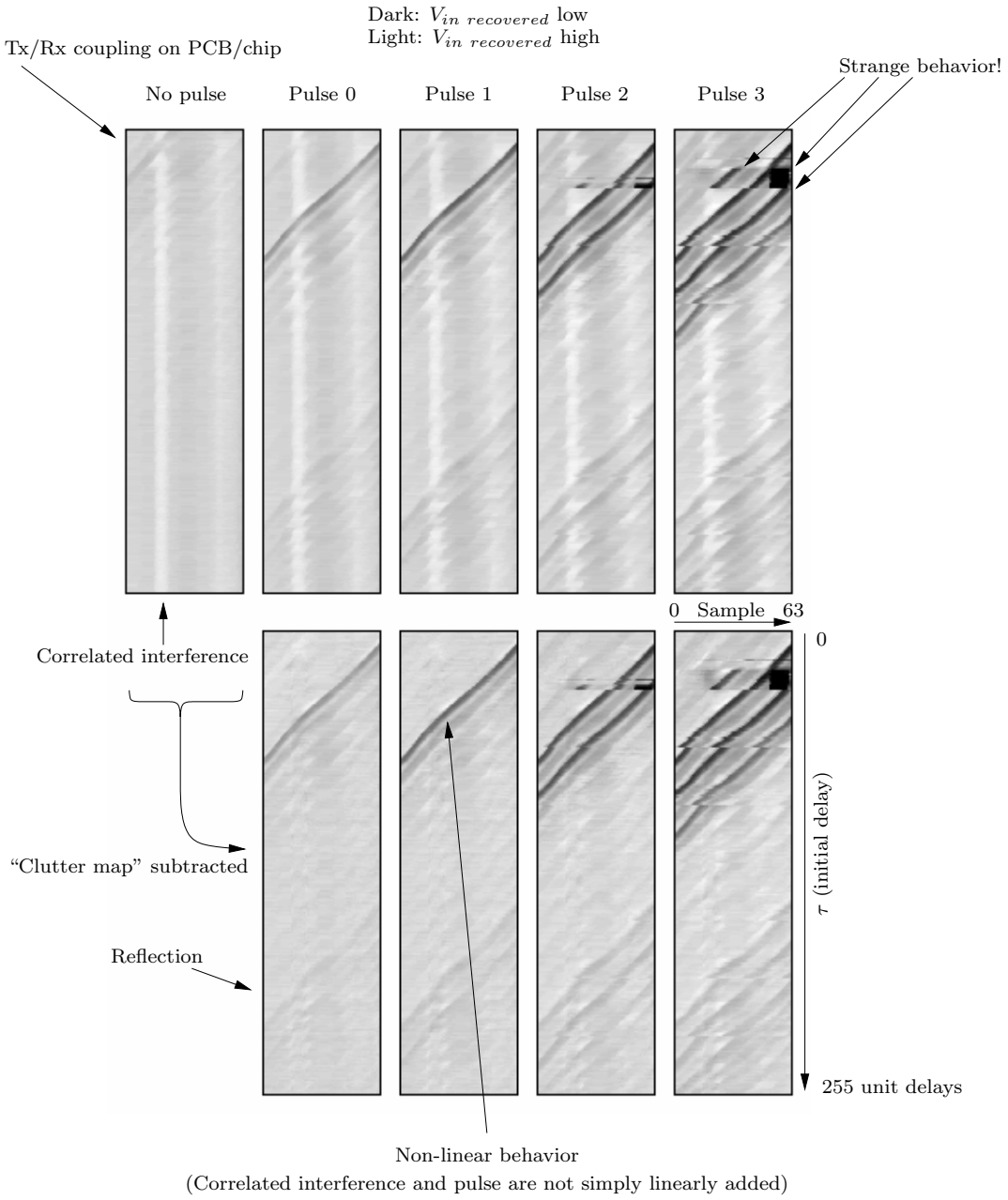

Figure 6.10.: Readout from the radar with different pulses

By subtracting the "clutter map" we get when there is no pulse, or when there is no loop cable, to be precise, we get better readouts of the cases with pulses.

After subtracting the clutter map we can see that some remains from the correlated interference from the "sample" pin still exist in the crossing between the pulse and the interference. This suggests that there are some non-linearities involved in addition of the two signals, i.e. the interference and the pulse. This could mean that our input pathway from package pin to thresholder input is not a linear system.

An interesting observation to make is the reflections appearing a little later than the main pulses. This is probably due to the coax not being properly terminated since since it is connected to the LNAout pin instead of the LNA input which has 50 Ω matching.

A second observation is that pulse 2 and pulse 3 cause some very strange behavior in the sampler. The 45° lines suddenly start and stop in "mid-air", a dark spot appears for only some τ, and the factor between short and long unit delays in the initial delay circuit seems to have changed, thus the horizontal tears in the gray scale images. What causes all this is not known.

Pulse 1 looks very good, so it will be used in all the subsequent measurements.

6.2.3. Radar readout with different cable lengths

Figure 6.11 shows readouts from the radar sampler for loop cables of different lengths. Again we see reflections. An interesting thing to note here, is that the pulse which has gone through the longest cable is a little weaker than the others, but not very much blurred. If there was much jitter in the initial delay circuit, that would have caused blurring, so apparently there is not very much jitter.

6.2.4. Measuring the sample rate

Figure 6.12 shows a plot of the radar readouts for two different cable lengths at a given τ. The pulses are measured to be separated by 30.0 samples. This difference is due to the difference in travel time in through each of the cable lengths. The cable, "Suhner Switzerland RG 400/U", has a signal propagation velocity of 69% of c. Knowing that the difference in cable length is 27.5 cm, we can calculate the sampling period and

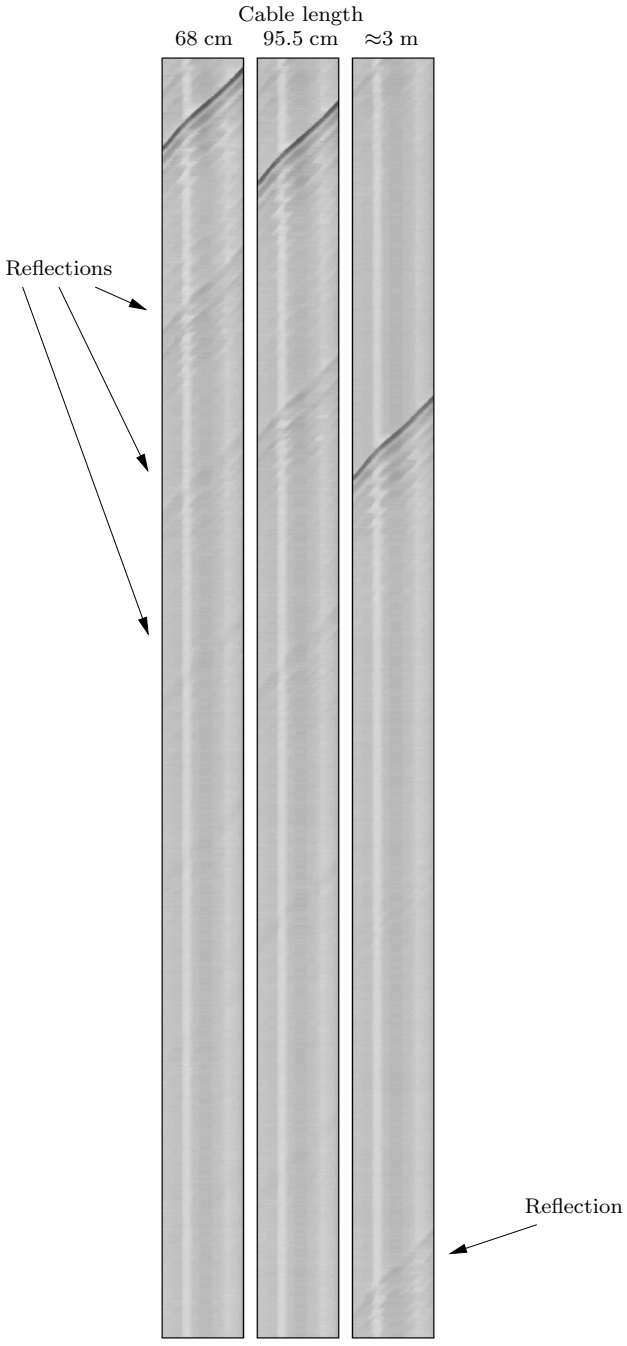

Figure 6.11.: Readout from the radar with different cable lengths

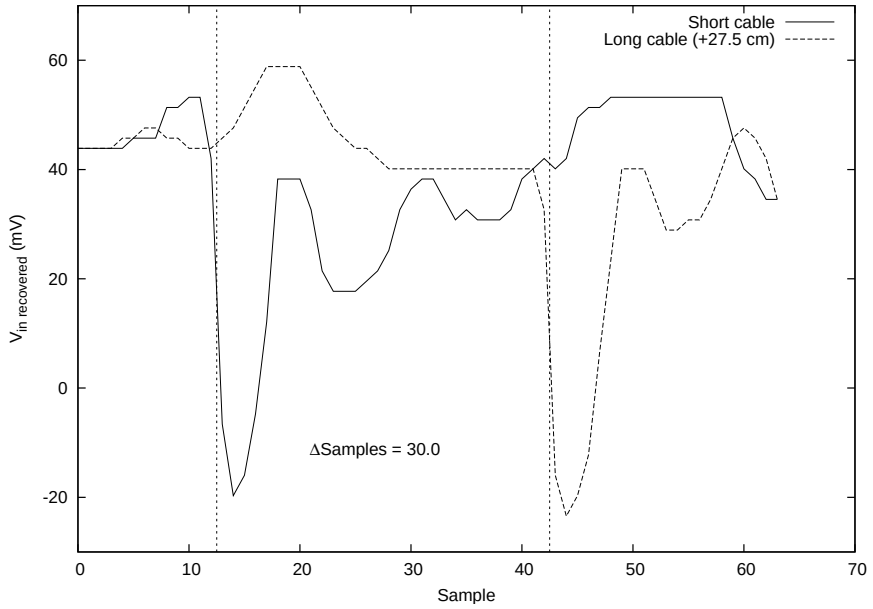

Figure 6.12.: Measuring the sample rate

sample rate:

$$
\begin{aligned}
v &= 0.69c \\
\Delta s &= 27.5 \text{ cm} \\
\Delta Samples &= 30.0 \\
\Delta t &= \Delta s / v \\
&= 1.33 \text{ ns} \\
T_s &= \Delta t / \Delta Samples \\
&= 44 \text{ ps} \\
f_s &= 1 / T_s \\
&= 23 \text{ GHz}
\end{aligned}
$$

So the radar sampler has a sampling frequency of 23 GHz!

6.2.5. Detail view of swept threshold sampling

Figure 6.13 shows a detailed view of the swept threshold sampling method. Each row in the images is a readout of the 64 counters, and for row from bottom to top the threshold V_T is increased. Only one sampling has been done at each threshold level. Black

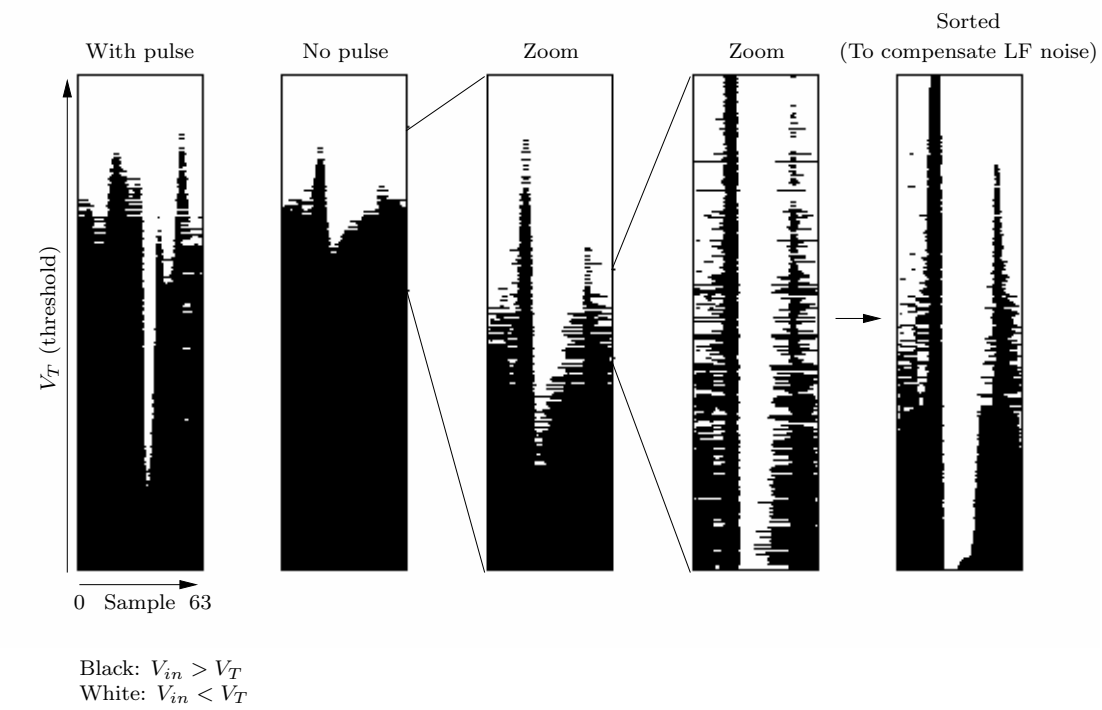

Figure 6.13.: Detailed view of swept threshold sampling

represents a counter value of "1", and white represents a counter value of "0". Measurements both with and without a pulse is shown, and a zoom-in of the no pulse case is shown. The zoom-in simply means stepping through a short range of V_T with smaller steps.

A normal swept threshold readout would be made just like this measurement, but instead of reading and resetting the counter at each threshold level like here, the counters would simply be allowed to accumulate throughout the sweep, and not read before the entire range of V_T had been swept through. That means that summing each vertical column of these measurements would produce a normal swept threshold readout.

The second zoom level looks quite untidy. This could be due to low frequency noise which essentially shifts the threshold level up and down randomly. To compensate this, we sort the rows according to how many "1"-s there are in them. The result of this operation looks much better than the unsorted one. For normal swept threshold readout, though, the order of the rows do not matter, since they are summed anyway.

An interesting observation is that we see sequences of consecutive "1"-s or "0"-s down to one or two counters in length. This means that the system is able to handle also very short pulses.

6.2.6. Comparison to oscilloscope measurement

In figure 6.14, we have plotted the readout from the radar together with a measurement of the pulse using a regular oscilloscope. We see there is a big difference in both shape, phase and amplitude of the pulse. What causes all this is not known, but the inductance of the bonding wire, the broken LNA and parasitic capacitances are all possible culprits here.

6.2.7. Radar measurement of moving hand

Figure 6.15 shows a measurement with antennas connected to both the pulse generator and the input to the radar sampler. The antennas are placed close together, and a hand is being moved back and forth once in front of them.

Since we have no LNA, there is much noise in the system, and we have to average a lot of samples. Here, we average 120'000 samples. Because of this high number and because the readout circuitry used is a bit slow, the whole session took about 350 seconds to record.

The leftmost gray scale image shows the original swept threshold readout. Here, we can see some horizontal lines. This is some sort of low frequency noise which can be removed by subtracting the DC-value from each row. This has been done in the second image. The "clutter map" has also been subtracted from the second image. This removes the vertical pattern. The third image is simply an amplification of the second image.

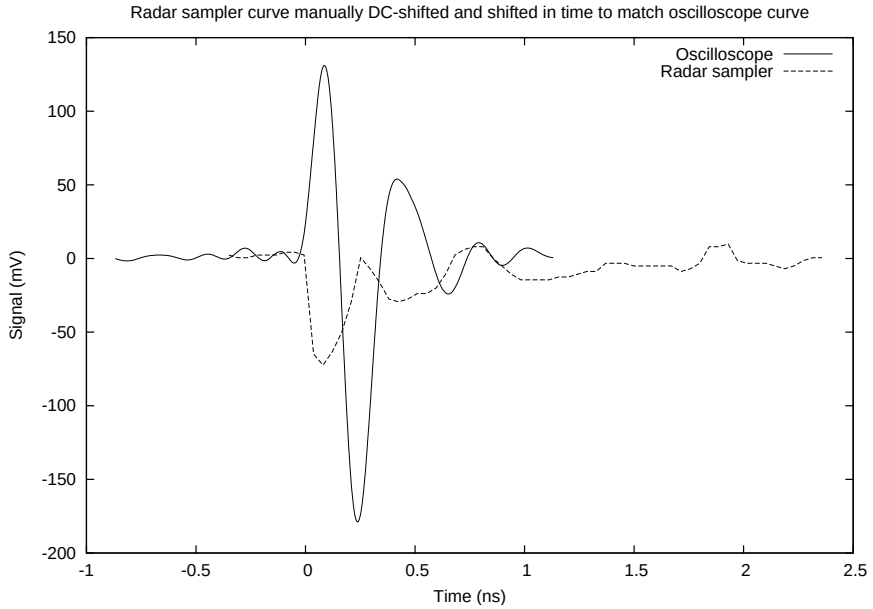

Figure 6.14.: Comparison between readout from radar and oscilloscope

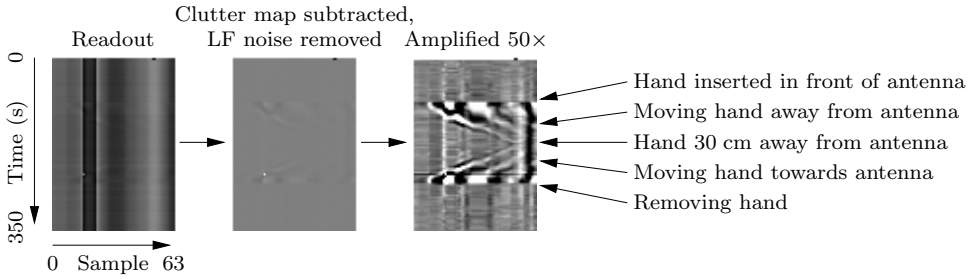

Figure 6.15.: Radar measurement of moving hand

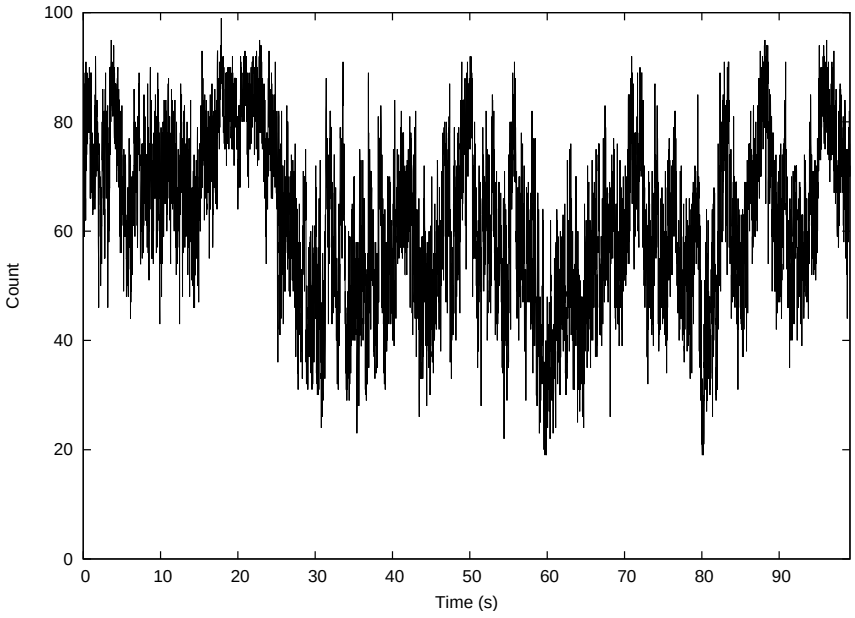

Figure 6.16.: Low frequency noise in the radar

We can see that we are able to detect a hand at a distance of at least 30 cm from the antennas. The closer the hand is, the stronger the radar echo is. We also note that the resolution is very high, with tens of pixels covering just 30 cm. The resolution is more precisely:

$$\begin{aligned} \Delta R &= cT_s/2 \\ &= 6.6 \text{ mm} \end{aligned}$$

The radar has a resolution of 6.6 mm! On the other hand, it does not have a very long range, though. These properties are quite different compared to other radar systems like for instance air traffic radar.

6.2.8. Low frequency noise

To measure the noise characteristics of the system, we have used stochastic resonance sampling. We set the threshold to a value near the DC point of the signal, then sent 100 pulses and read back one of the counters. We then repeated this in rapid succession to get a readout of the low frequency characteristics of the noise. We are here using noise to measure noise itself, so the results might not have a very high quality, but we are indeed getting some interesting readout anyway. In figure 6.16, we can clearly see how

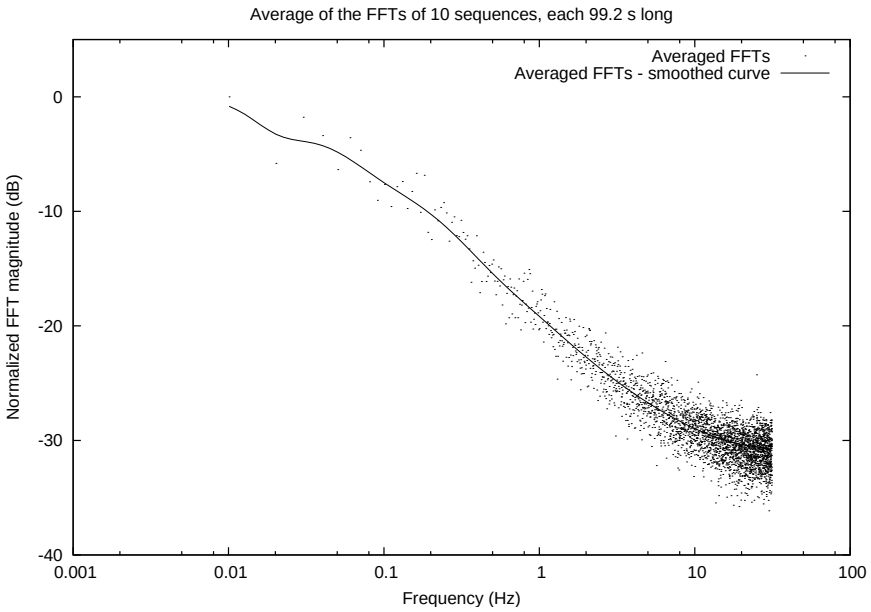

Figure 6.17.: Low frequency noise in the radar: FFT

the readout varies with time at very low frequencies. The Fast Fourier Transform (FFT) in figure 6.17 confirms this, and shows that the noise exhibits a $1/f$ characteristic. The source of the noise could be the first NMOS transistor in the common source amplifier cascade. The transistors in the common source amplifier have a very small gate area, namely $0.6 \cdot 0.1\mu m^2$, so they probably generate some flicker noise.

The measurement setup has used USB powering of the circuit, so this could be a noise. Battery supply has been tried, however, and no improvement could be seen.

It might seem strange at first that low frequency noise has anything to say for us, since we operate at such high frequencies. But considering that this noise might be modeled as a noisy threshold, we quickly understand how this definitely affects us. The tricky thing about low frequency noise, is that it can not be averaged away within a short period of time. But since all the 64 samples in our sampler gets the same offset problem due to this noise, the low frequency noise can simply be alleviated by removing the DC component from the sequence of the 64 samples. This might sound perfect at first, but because the 64 samples have different values of ideal V_{in}, they will each interact differently with a given pattern of low noise fluctuations. Thus, some of the problem from the low noise will leak through the DC-removal step.

Ways to fix this problem is to perhaps use greater gate area or perhaps using an amplifier with some sort of feedback structure, so as to dynamically correct for the low

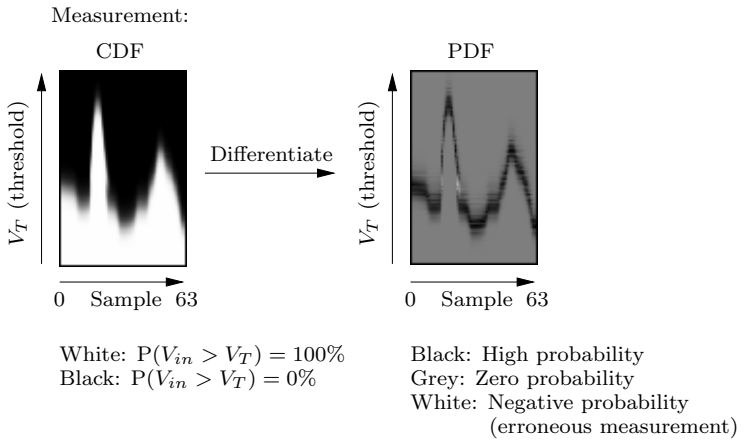

Figure 6.18.: Measuring σ_N: CDF and PDF gray scale plots

frequency noise.

6.2.9. Measuring the noise standard deviation

To measure σ_N, the noise in the input signal, we can sweep the threshold through a given range at some given step size. By resetting the counters, taking many samples and then reading out the counters at each step, we get a set of data which describes the probability of the input signal being over the threshold at each of the given threshold levels. Such a data set is shown in the left gray scale image in figure 6.18. Each vertical column in this image is a Cumulative Distribution Function (CDF). By differentiating this data with respect to V_T, we get PDFs in the columns, shown in the right gray scale image in the figure. The CDFs are also plotted in figure 6.19, and the PDFs are plotted in figure 6.20. Both these last graphs have the curves shifted so that they overlap in order to compensate the fact that the CDFs and PDFs have different centers. This, of course, is because the ideal V_{in} is different for each of the 64 samples.

We can see that the plot of the PDFs roughly resembles a normal distribution function. The standard deviation is calculated for each of the 64 samples and plotted in figure 6.21. We see that $\sigma_N \approx 1.4$ mV. Where there is sharp slopes in the input signal, the σ_N rises. This is probably due to jitter in the initial τ or other places in the system. If a sharply rising curve is shifting slightly back and forth, the amplitude at a given, fixed point will fluctuate with a very high amplitude. That is the reason jitter noise in combination with sharp slopes gives high σ_N.

Corresponding measurements for when a pulse is being received is shown in figure 6.22 and figure 6.23. Again, sharp edges cause an increase in σ_N.

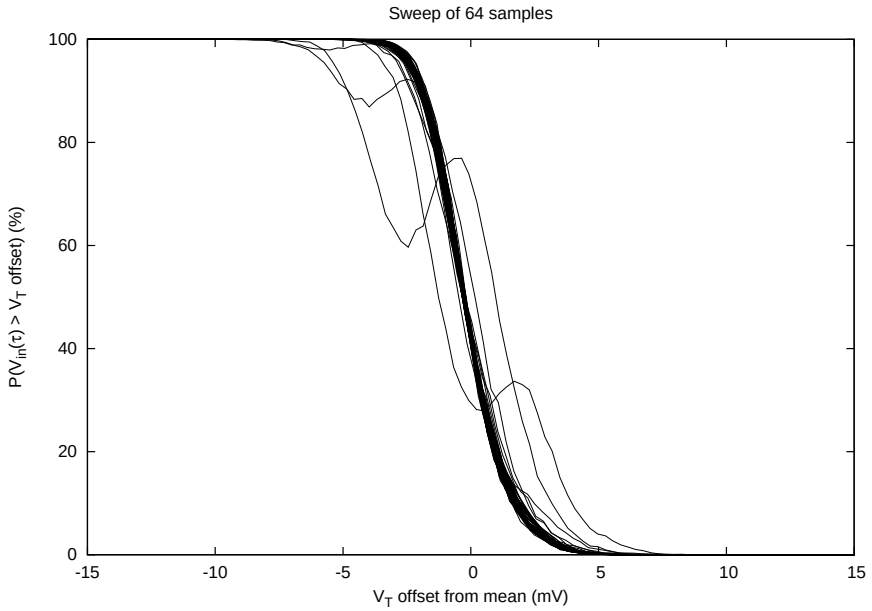

Figure 6.19.: Measuring σ_N: CDF

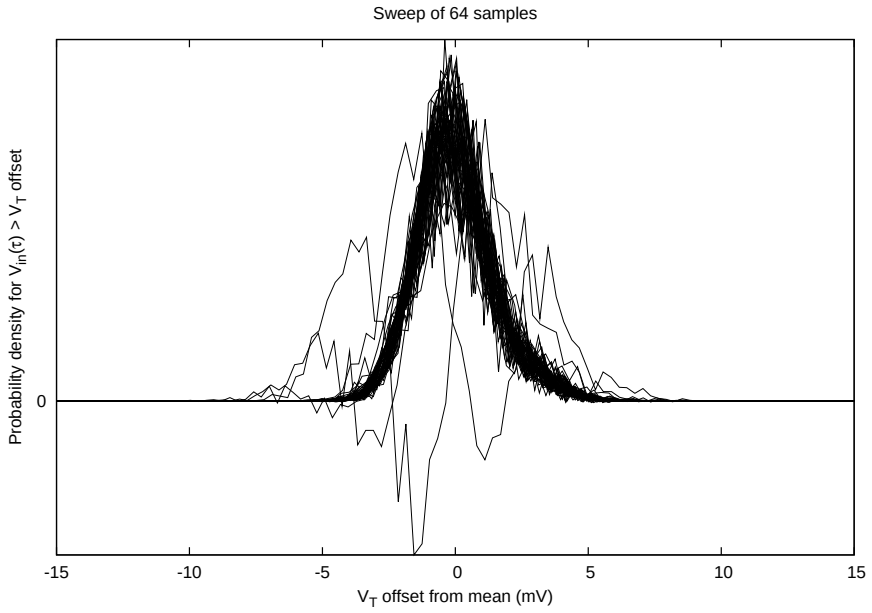

Figure 6.20.: Measuring σ_N: PDF

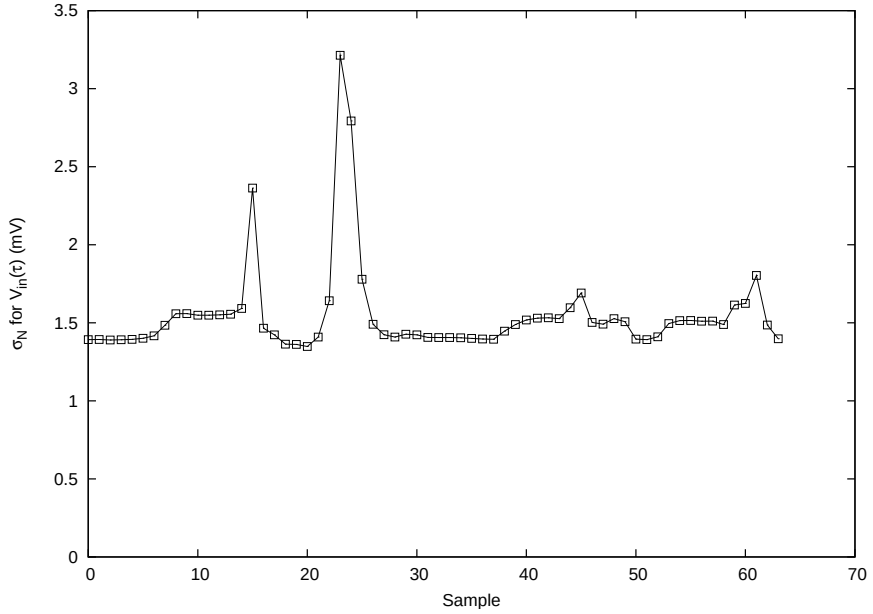

Figure 6.21.: Measurement of σ_N

Note that the CDFs presented here are actually inverted compared to the normal CDFs, which go from low to high, not high to low.

6.2.10. Measuring the noise in the recovered signal

To measure $\sigma_{N\ recovered}$, we repeat a complete sampling 100 times, i.e. we retrieve the complete sample $V_{in\ recovered}$ 100 times. Each time we retrieve it, it will be slightly different. This is what we call the $\sigma_{N\ recovered}$ noise. In figure 6.24, we have made these 100 complete samplings for both swept threshold and stochastic resonance, and done so also for different values of n, i.e. the number of steps for swept threshold and the number of repeated samplings for stochastic resonance. To reduce the low frequency noise problem, we have tried to remove these noise components, which appear as horizontal lines in the uncorrected gray scale images.

Next, we choose sample 5 and measure $\sigma_{N\ recovered}$ for it, i.e. we calculate the standard deviation of all the 100 readouts of sample 5. Sample 5 is chosen because the stochastic resonance sampling method only has a very narrow good region consisting of the first few samples. The stochastic resonance data is LF-noise compensated by removing the DC component of sample 0, i.e. making sample 0 equal throughout the 100 repeated samplings. This means that the closest samples, sample 1, sample 2 and so on, are quite

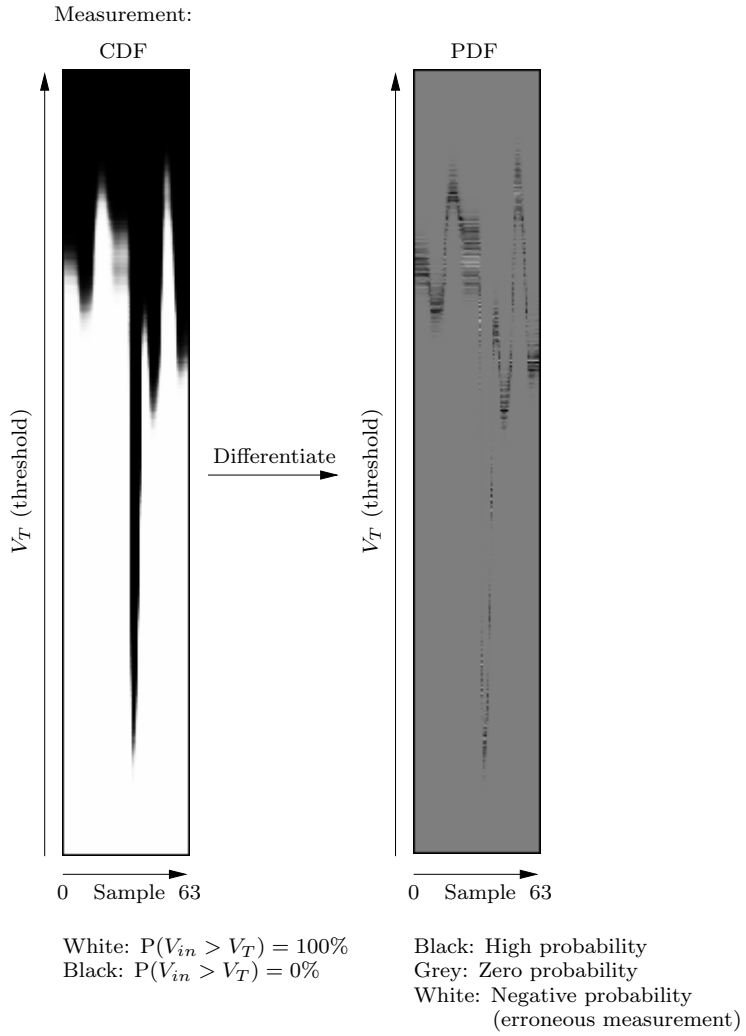

Figure 6.22.: Measuring σ_N of received pulse: CDF and PDF gray scale plots

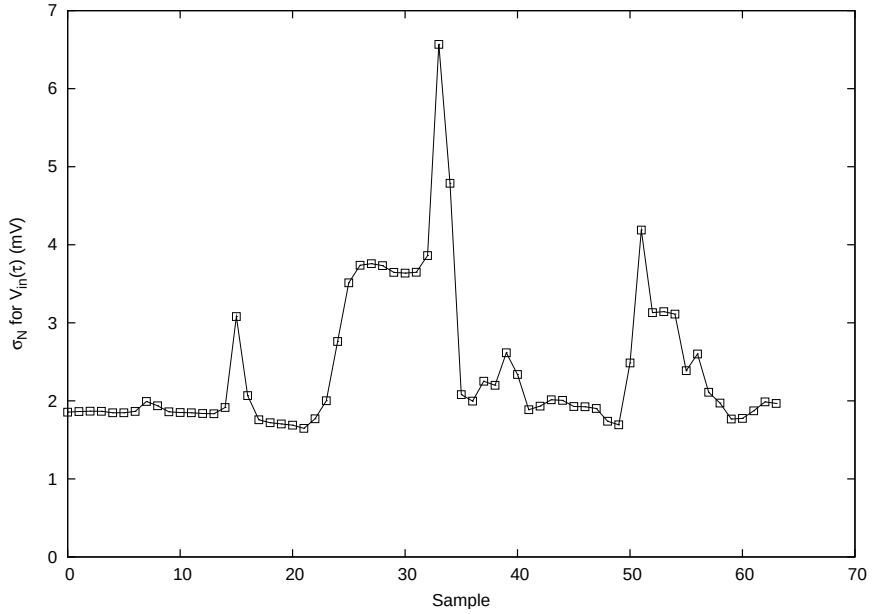

Figure 6.23.: Measurement of σ_N of received pulse

tied to this and therefore exhibit less noise than they should. Sample 5 is hopefully sufficiently far away. Higher samples start to go out of the stochastic resonance good region, so using a higher sample is not a good idea either.

The measured $\sigma_{N\ recovered}$ is plotted in figure 6.25 and figure 6.26. We see that the measured data fits well with the theoretical model.

σ_N for swept threshold is 0.0321, higher than the measured 0.0014. This is because of a compensation that has been done. We are sweeping over a shorter range than from 0 to Vdd=1 V, so we have to increase σ_N accordingly in order for the model to match. In other words, if we look at a smaller signal, the input noise gets relatively larger.

For stochastic resonance, we have a choice of σ_N because there really is not any signal range in this measurement. We choose $\sigma_N = 0.32$, since this is a good operating region for the stochastic resonance sampler, and because the counter to $V_{in\ recovered}$ mapping function is almost simply proportional at this noise level (figure 5.16). This means that little error is introduced due to the lack of a mapping function.

6.2.11. Power consumption

Digital Vdd: 0.34 mA

When setting the threshold to the DC point of the input signal, so that it constantly

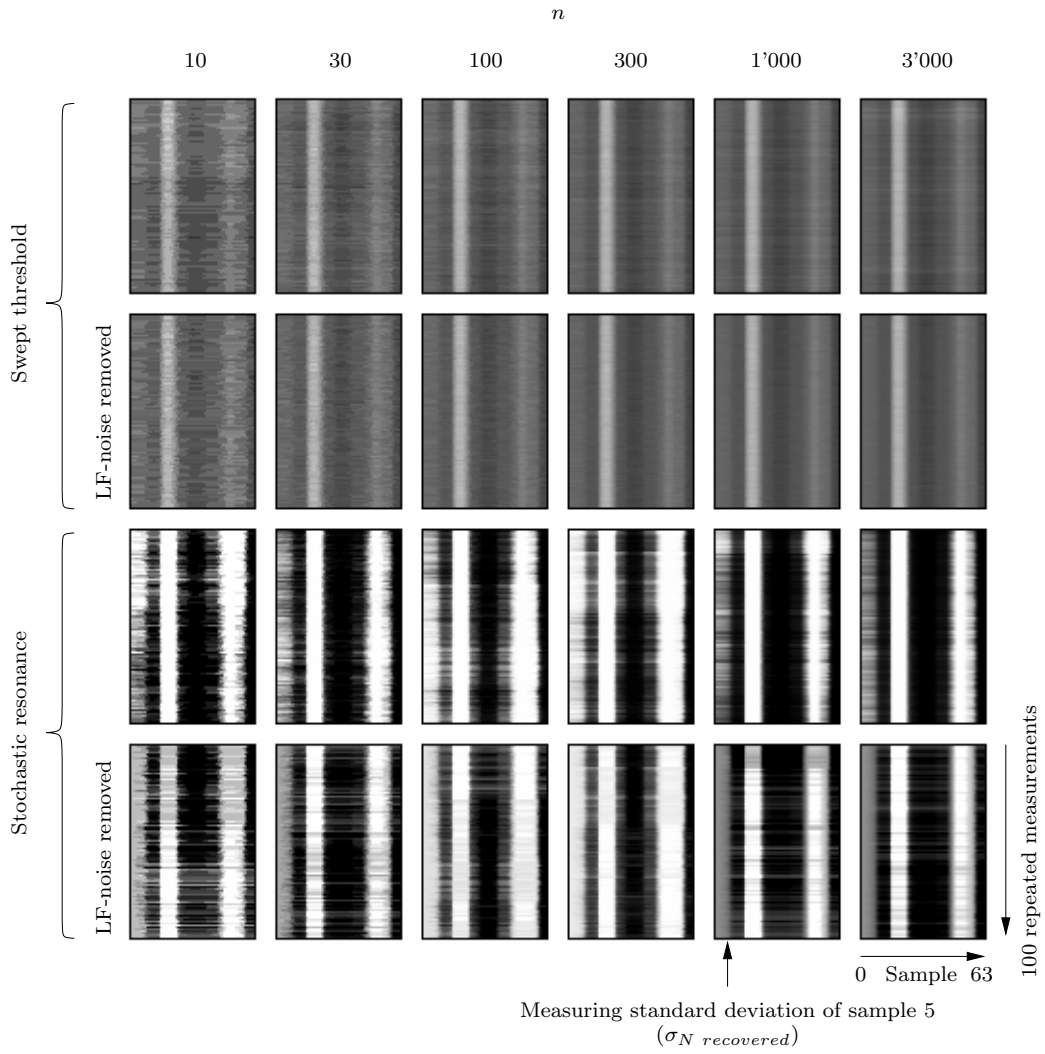

Figure 6.24.: Measuring $\sigma_{N\ recovered}$ in repeated samplings with both sampling methods

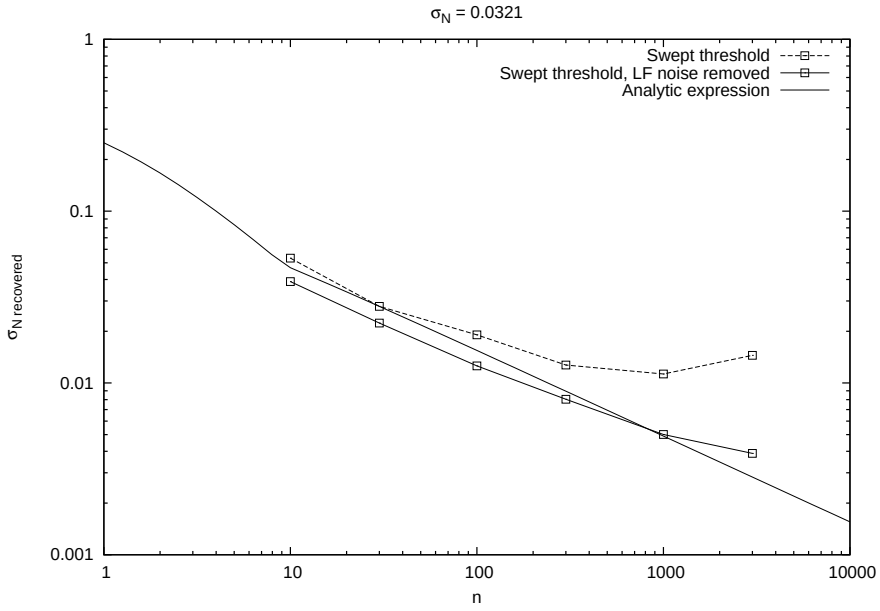

Figure 6.25.: Measured $\sigma_{N\ recovered}$ for swept threshold

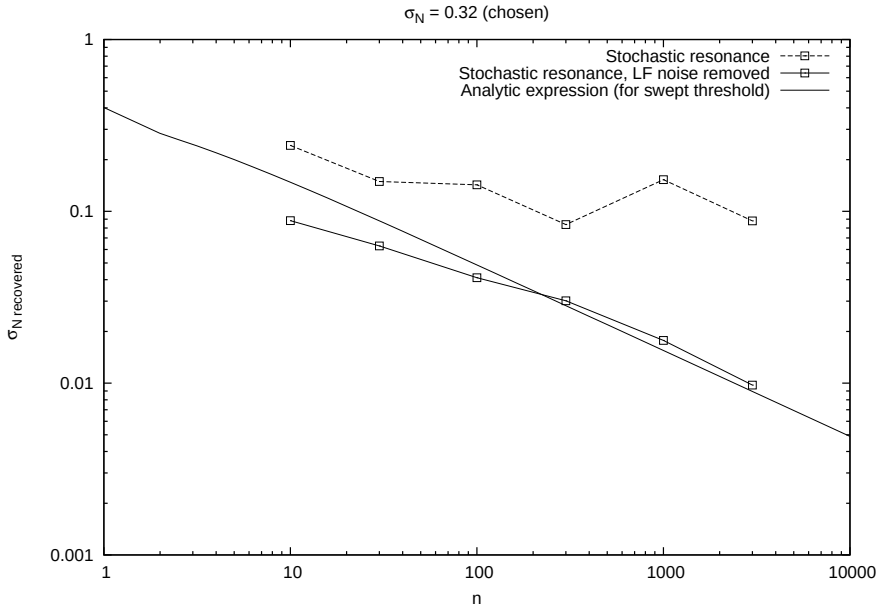

Figure 6.26.: Measured $\sigma_{N\ recovered}$ for stochastic resonance

crosses the threshold, and setting the ForceDistributeSignal register bit for the buffer, i.e. force the buffer to distribute the signal also when a sample is not just about to be made, in this cases, this power supply line draws 1.3 mA. This shows how limitation of signal distribution can save power. The 0.34 mA is still quite high, and might be because of leakage currents through the transistors. High threshold transistors should probably have been used at least in the less frequency demanding areas.

Analog Vdd1 (Pulse gen.): 8.45 mA
A design flaw caused the pulse generator to have a constant current flowing through an NMOS and an inductor.

Analog Vdd2 (Prog. delay, SPI, sampler delay line): 0.02 mA
Not affected by transmission of pulses, but we only reach about 39 kHz with our measurement setup.

Analog Vdd3 (LNA, thresh.): 4.7 mA
Does not change when setting the threshold to the DC-point of the input signal, thus causing rapid threshold crossings. The LNA is probably what is drawing the most power here.

Entire circuit when in use: 57 mA
Also includes DAC and level-shifter circuitry.

6.3. Summary

A working short-range high resolution UWB impulse radar has been designed, implemented in 90 nm CMOS and measurements have been made. The radar has a sampling rate of 23 GHz and thus a resolution of 6.6 mm. Some strange problems have been encountered however. The sampler does not seem to be able to read the same pulse shape as a regular oscilloscope reads. Where this problem lies, is hard to say. There is also a lot of noise in the system because the LNA did not work and thus had to be bypassed.

6.3.1. Suggested improvement

The implemented circuit requires a DAC to set the threshold level. It might be possible to create a circuit which does not require this, which simply adjusts itself to its operating point, allowing for stochastic resonance readout. This self-adjustment will perhaps also remove the problem with the low-frequency noise.

7. Conclusion

In this master thesis we have presented an unconventional radar sampler technique and an implementation of it which was shown to work. The sampler in the implementation operates at 23 GHz, thus giving the radar a resolution of 6.6 mm. We have also tried to go in depth into the CTQA signal processing domain which the radar sampler is based on. System level simulations have been done on the sampling technique, and measurements have been made that agree with these results.

The most important result in this master thesis is that the introduced swept threshold sampling technique is actually very close to a perfect sampler in cases where the input signal is buried in strong noise. This is a bit surprising since the circuit is quite simple and coarse as it is mostly based on digital unclocked circuitry.

It is hoped that this result can be a contribution to the ongoing exploration of the recently discovered Suprathreshold Stochastic Resonance (SSR) phenomenon, introduced by [Stoc 00].

7.1. Further research

The analog parts of the circuit implementation needs to be worked through. The signals that are sampled with the radar sampler implementation are heavily distorted and plagued with noise, especially a difficult low frequency noise.

An idea for a circuit is that if you have noise of $\sigma_N = 0.1$ and then take your one input amplifier, for instance an LNA, and "split" it into 100 smaller amplifiers. Each would have 100 times as high input impedance and thus 10 times as high noise amplitude. They would also each perhaps consume only one hundredth of the power. In total, presenting the same input impedance and power consumption as the original LNA. Now, however, the noise is $\sigma_N = 1$, which is perfect for a swept threshold or stochastic resonance sampler. In addition, one sampling yields 100 samples at once. Thus reducing the noise again. This way, it might be possible to "shape" the noise distribution in order to put the sampler into the right working conditions. This idea needs more research though.

A. Chip data sheet

Data sheet: S09C58G2: Experimental short-range UWB impulse radar with swept-threshold sampler

Håkon A. Hjortland

July 29, 2006

1 Description

- Experimental design!
- 3.1–10.6 GHz target frequency range
- Approx. 23 GHz swept-threshold strobed sampler
- Sampler uses quantized, continuous-time, signals
- Tunable pulse transmitter
- Single-ended antenna output
- Double-ended antenna input
- Pulse transmission externally triggered
- SPI-inspired interface
- 1.0 V supply voltage
- Implemented in STMicroelectronics 90 nm CMOS technology
- JLCC-68 package
- Created by Håkon A. Hjortland, Tor Sverre Lande, Håvard Moen, Kjetil Meisal and Claus Limbodal

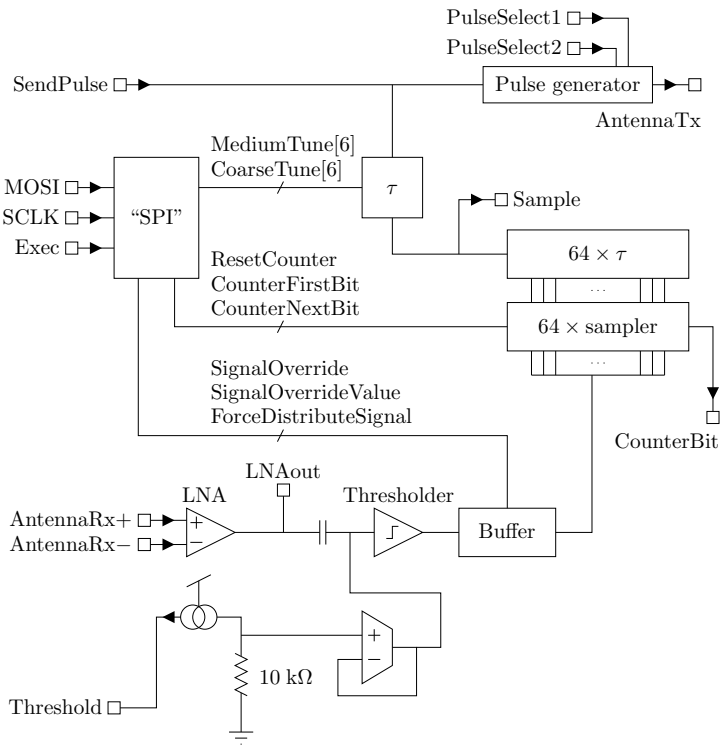

Figure 1: Block diagram for chip

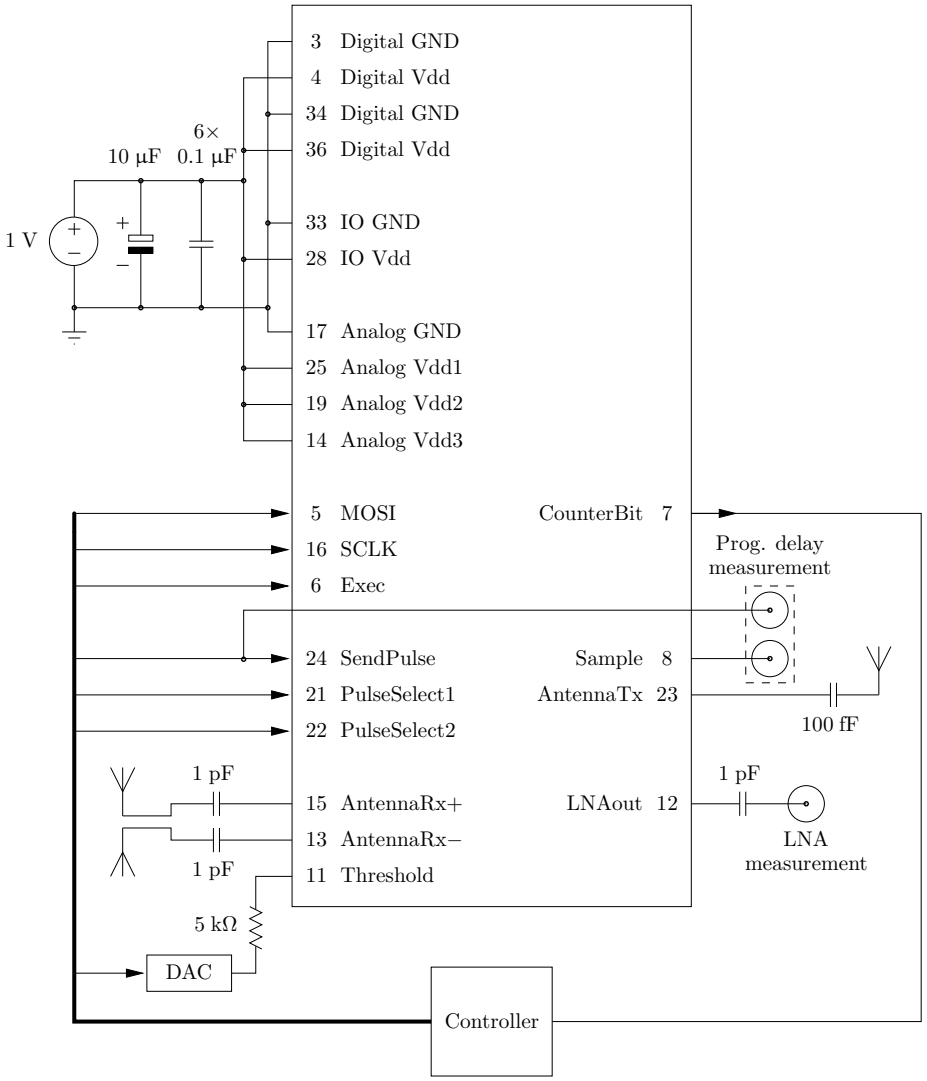

Figure 2: Typical circuit

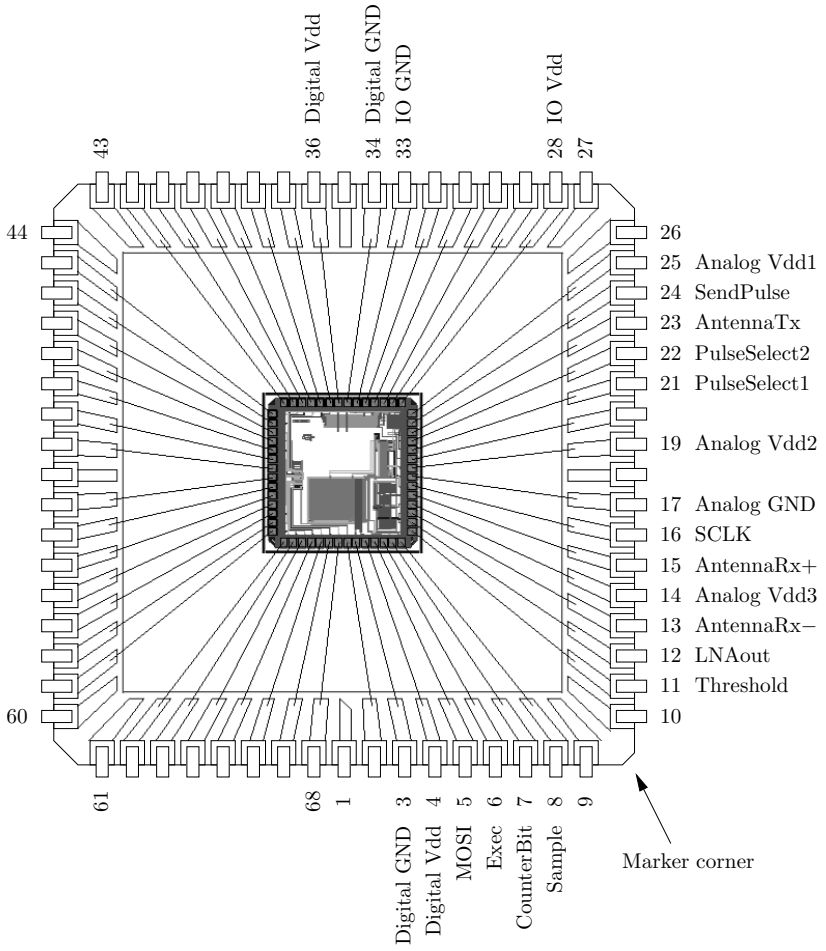

Figure 3: Wire bond diagram, JLCC68 package overview and pin configuration. Not to scale.

Pin	Description
Digital GND	Padframe digital IO GND (core-side).
	Core supply (digital circuits).
	Substrate connection.
Digital Vdd	Padframe digital IO Vdd (core-side).
	Core supply (digital circuits).
IO GND	Padframe digital IO GND. ESD reference for pads.
IO Vdd	Padframe digital IO Vdd. ESD reference for pads.
Analog GND	Common GND for analog circuits.
	Substrate connection.
Analog Vdd1	Pulse generator, SendPulse buffer.
Analog Vdd2	Prog. delay, "SPI", sampler delayline.
Analog Vdd3	LNA, thresholder.

Table 1: Power supply pins

2 System overview

The SendPulse pin triggers a pulse transmission. After a programmable time τ, 64 parallel samplers are triggered in sequence, sampling at an interval of about 43 ps (23 GHz). The antenna input is thresholded with a given threshold. It is this quantized signal that is sampled. If one of the 64 samplers sample a '1', a corresponding 16-bit counter is incremented. After up to 65535 pulse transmissions / samplings have been made, the 1024 bits of the counters can be read out through a serial interface. The counters can then be reset.

3 Recommended operating conditions

Vdd	1.0 V
Digital V_L	0.0 V
Digital V_H	1.0 V
Threshold	0.0–0.2 mA

4 Pin description

4.1 Power supply pins

See table 1.

4.2 MOSI, SCLK, Exec, CounterBit

"SPI" interface (section 5 on page 7).

4.3 SendPulse

Trigger a pulse transmission/sampling.

4.4 AntennaTx

Transmit antenna. Single ended. Requires AC coupling of about 100 fF. Can be connected to a transmission line of e.g. 50–75 Ω.

4.5 Sample

Measuring point: Output from programmable delay. Goes high when the sampling sequence begins.

4.6 PulseSelect1, PulseSelect2

PulseSelect<2,1> is a 2-bit number which selects 3, 5, 7 or 9 delay elements for the pulse generator. Shorter delay means higher frequency.

4.7 AntennaRx+, AntennaRx−

Receiver antenna. Differential input. Requies a 1 pF AC-coupling capacitor on each line. Internally biased to about Vdd/2. Input impedance is 50 Ω in the 3.1–10.6 GHz range.

4.8 LNAout

Measuring point: Single-ended LNA output.

4.9 Threshold

Threshold selector. Current drawn from this pin determines the threshold. See figure 1 on page 2 for details. A typical configuration with a 5 kΩ resistor between the pin and a voltage source V_{in} (figure 2 on page 3) has an approximate transfer function $V_{res} = 1.118 - 1.427V_{in}$ between the input voltage and the voltage over the internal 10 kΩ resistor. The transfer function is relatively linear for $V_{in} \in [250, 650]$ mV. The V_{res} range will then be $[190, 761]$ mV. The actual threshold voltage is the difference between the resistor voltage and the operating point of the thresholding element, $V_{threshold} = V_{op} - V_{res}$. With $V_{op} \approx 0.5$ V:

$$V_{threshold} = -0.618 + 1.427V_{in}$$

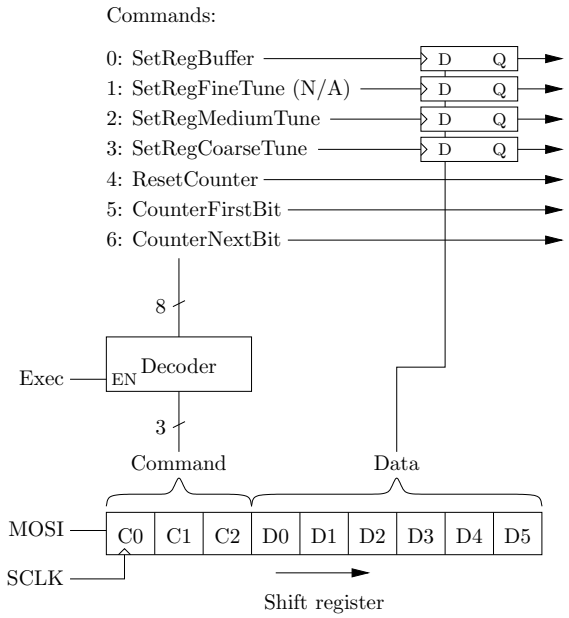

Figure 4: "SPI" interface

5 "SPI" interface

SPI-inspired serial interface. Bits from the MOSI pin is clocked into a shift register on the rising edge of SCLK (figure 4). When Exec goes high, the given command is executed. The command can be either "Set a register" or "Send a signal".

5.1 Register "Buffer"

Bit	Signal	Description
D0	SignalOverride	'1' means override input signal
D1	SignalOverrideValue	Which value to override with
D2	ForceDistributeSignal	The input signal is usually only distributed a short time after a pulse has been sent. '1' here enables distribution at all times.
D3–D5	(Unused)	

5.2 Register "FineTune"

Not implemented.

5.3 Register "MediumTune"

6-bit value. 0–63 × approx. 30 ps programmable delay.

5.4 Register "CoarseTune"

6-bit value. 0–63 × approx. 1 ns programmable delay.

5.5 ResetCounter, CounterFirstBit, CounterNextBit

Signal	Function
ResetCounter	Reset the counters in the sampler
CounterFirstBit	MUX the first counter bit to CounterBit
CounterNextBit	Step to the next bit on each Exec rising edge

Typical sample/readout procedure:

1. Send ResetCounter signal
2. Transmit n ($\leq$65535) pulses (triggered with SendPulse)
3. Send CounterFirstBit
4. Read CounterBit
5. Send CounterNextBit
6. Read CounterBit
7. Repeat until 1024 bits have been read

Bit order: <one spurious bit>, sample 63, sample 62, ..., sample 1, sample 0.
Each sample: Counter bit 15, 14, ..., 1, 0.
Counter goes 0, 65535, 65534, ...
Count c from readout r: $c = (65536 - r) \bmod 65536$.

6 Problems

- The LNA seems to oscillate. LNAout can be used as input instead. Silence the LNA by pulling one of its inputs up or down.
- The Sample output pin raises at a time correlated to the start of the sampling sequence of the 64× sampler. If this pin somehow couples into the sampler input, this could be a problem since such correlated interference noise can not be filtered out through averaging. Ripping the pin off of the chip carrier or at least not connecting the pin to a wire seems to reduce the problem.

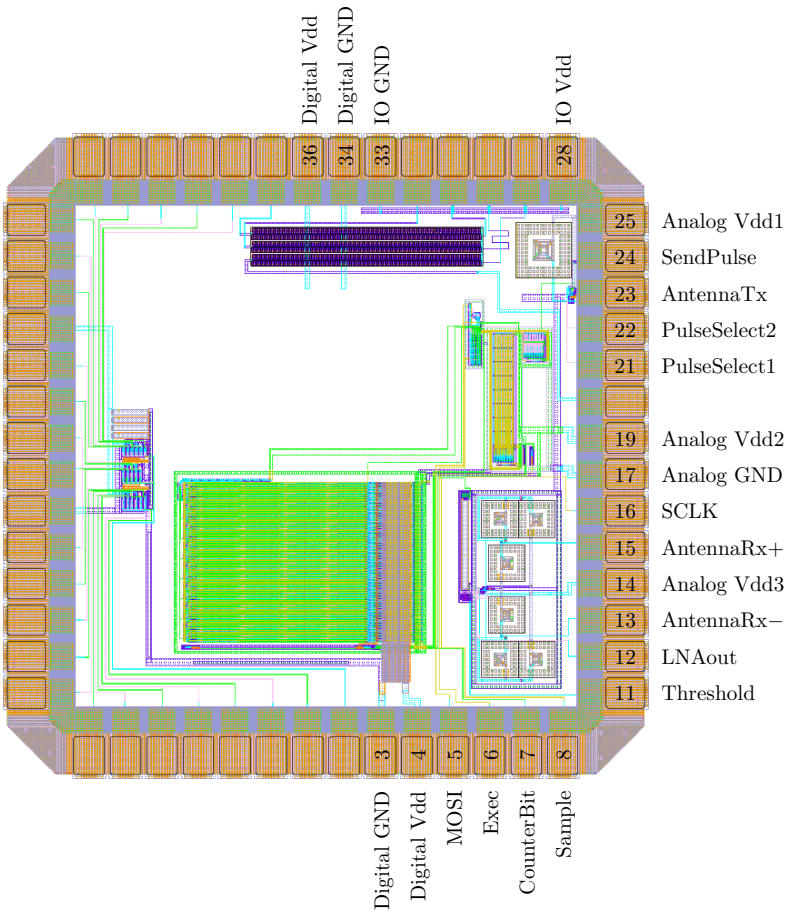

Figure 5: Padframe pinout diagram

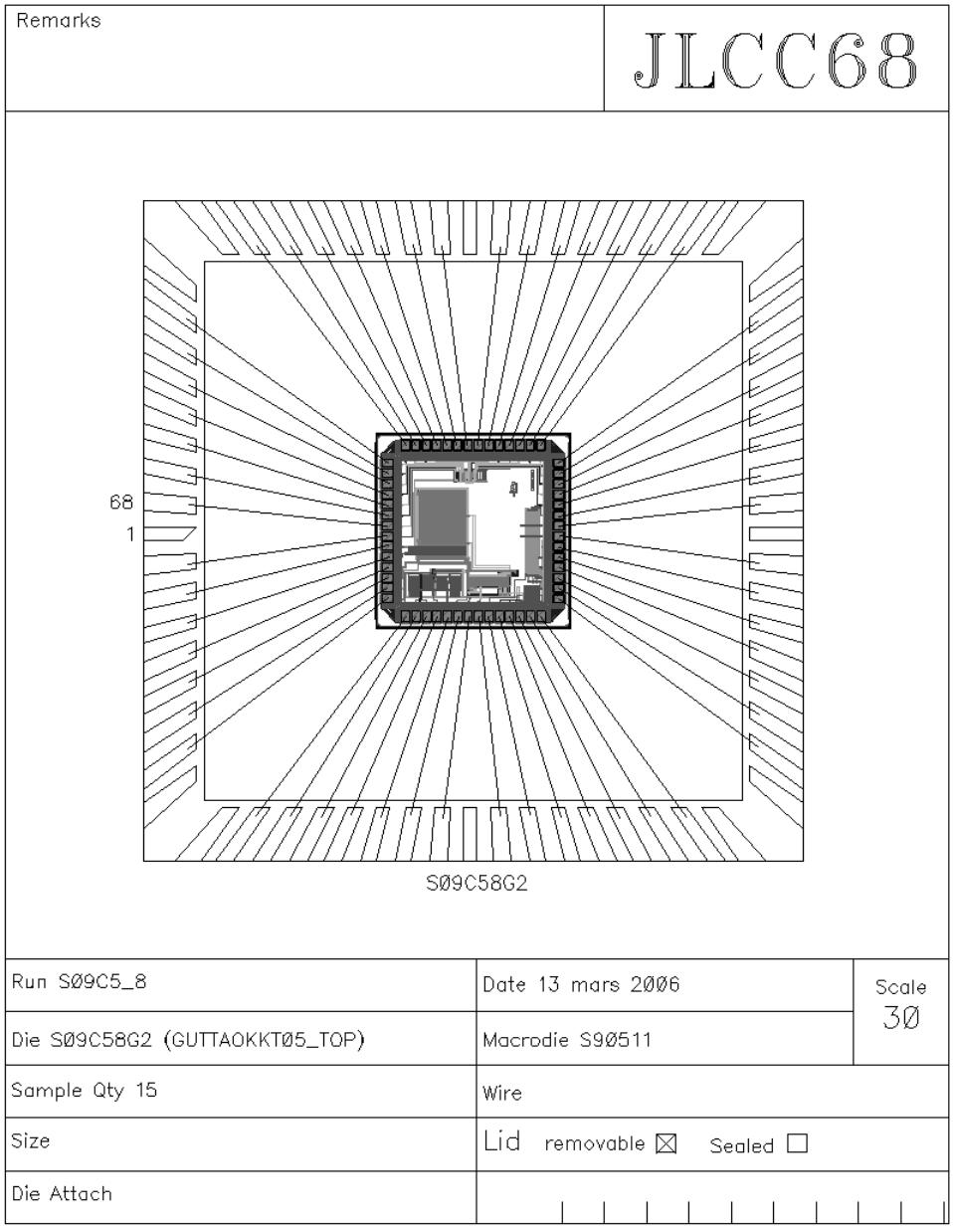

Figure 6: Wire bond diagram

B. Paper

The following paper has been submitted to the "IEEE Transactions on Circuits and Systems II, Express Brief" journal. Acceptance status is as of this writing not yet known.

CMOS Impulse Radar

Håkon A. Hjortland, Dag T. Wisland and Tor Sverre Lande
Microelectronic Systems, Dept. of Informatics
University of Oslo, Norway
Email: haakoh@ifi.uio.no

Claus Limbodal and Kjetil Meisal
Novelda A.S, Forskningsveien 2A
0373 Oslo, Norway

***Abstract*— A novel CMOS Impulse Radar (CIR) is proposed exploring the concept of swept-threshold sampling. Time-domain signal processing with counter-based integration in parallel structures is used. With continuous time, delayline based parallel sampling topologies we achieve a sampling rate in excess of 20GHz. A functional CMOS Impulse radar is implemented in silicon with measured system performance.**

I. Introduction

The conventional radar was introduced in the pioneering days of World War II and the original radar technology was based on impulse emission. Later, signatures or bursts were used for improved quality. The high frequency pulses or bursts of microwaves are demanding dedicated technology with high frequency devices (SAW filters, bipolars). In the late eighties [1] the impulse radar technique was revived for ground penetrating radar (GPR) for short-range detection of mines buried in the ground. During the nineties McEwan at Lawrence Livermore National Laboratory developed the micropower impulse radar (MIR) for a number of short-range applications.

In 2002 FCC released the largest unlicensed frequency band ever, known as Ultra Wide Band (3.1–10.6GHz). Combined with short gate delay of deep submicron (nanoelectronics) technology, exciting new application are feasible in standard CMOS.

In this paper we are exploring the possibility of short-range radar implementation in standard CMOS technology. Novel simplified processing techniques sweeping threshold in continuous time in combination with delay lines is taking advantage of technology scaling and is shown to work in silicon.

A. Background

Conventional radar technology is exploring signatures often made up by sinusoidal bursts (chirps). The duration of these bursts are limiting the depth resolution of the radar. A shorter signature will increase the resolution. The shortest signature is a single pulse and the shorter pulse length, the better resolution. However, backscattered energy from short pulses are hard to recover and significant signal processing is required.

McEwan [2] proposed the concept of micropower impulse radar (MIR). Randomized pulses are emitted and the backscattered energy sampled at a fixed, but short time interval (strobed) after pulse emission. The integration of backscattered energy is done with an analog integrator and sampling diode(s), preferably fast switching diodes like schottky diodes or step recovery diodes (SRD). The impulse radar may have significant more mileage if available in cheap CMOS technology. However, novel processing techniques and creative use of simple MOS transistors must be utilized to achieve this goal.

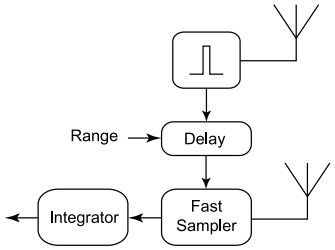

Fig. 1. McEwan micropower impulse radar principle

As indicated in figure 1, the pulses are emitted repeatedly at randomized intervals and the backscattered energy is sampled after an accurate, fixed delay from pulse emission. A typical pulse duration is $<$1ns, maybe close to 100ps, giving wide bandwidth ranging towards 10GHz. The width of the sampling window should be shorter, possibly a fraction of the emitted pulse. The range is set by an accurate delayed strobing after pulse emission. The backscattered signal is weak and often buried in noise. Improved signal-to-noise ratio is achieved by significant integration using fast samplers and analog integrators. An interesting variant of this radar is the motion detecting radar [3] having Doppler-like behavior. Even low frequency movement like heart beat or breathing can be detected as exemplified by Staderini [4] in his demonstrator of a medical radar.

Aiming at a standard CMOS implementation, the high speed signal processing, often analog, is a major challenge. The low power supply voltage virtually prohibiting any kind of high quality analog processing.

The advantage of deep submicron technology is the high speed with gate delay towards 10ps. In this paper we propose novel processing solutions where high quality analog signal processing is simplified to crude thresholding of backscattered energy and continuous time signal processing is explored for target detection.

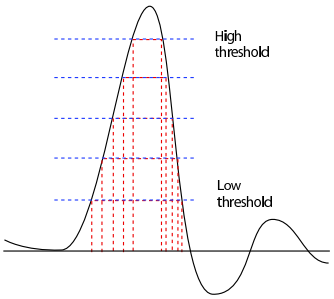

Fig. 2. By sweeping the thresholding level, the incoming signal may be recovered. To some extend the quantized pulse width reflects the strength of the signal.

II. System overview

The fundamental simplification of using quantized or digital values is well known. In $\Delta - \Sigma$ analog-to-digital converters the analog input signal is converted to a sequence of bits clocked at high frequency known as bitstreams [5]. The pulse density coding (or PWM coding) of bitstreams is accurately representing the continuous analog input value. The coarse thresholding of the continuous input to just '0' and '1' is compensated by increased clock speed. Trading reduced number of quantization levels with higher clock speed is quite appropriate for fine pitch technology and is explaining the gaining popularity of $\Delta - \Sigma$ analog-to-digital converters.

In the CMOS Impulse Radar (CIR) we are exploring this technique further by processing two-level quantized (digital) values in continuous time.

A. Swept-threshold coding

As explained above, the pulses are emitted repeatedly and the backscattered energy accurately strobed. In order to improve signal processing quality we are changing the threshold level "mapping out" the analog input signal as shown in figure 2.

The backscattered signal is close to buried in noise. For a given threshold white noise will appear as variations in pulse widths and sometimes even disappearing pulses. By sweeping the threshold we are able to recover quite a lot of the pulse energy and even take advantage of added noise. Pulses below the thresholding level will normally pass undetected but with white noise added, the pulses will sometimes exceed the threshold and the weak signal may be sensed. This behavior is often called *stochastic resonance* [6]. The concept of swept-threshold coding enables quite accurate signal processing even with only two quantization levels.

B. Integration

A crucial function of weak signal reception is significant integration. The purpose of integration is to get rid of added white noise. With repeated pulse emission and thresholding the white noise will introduce uncertainty in pulse detection. The probability of detecting a pulse, $P('1') = P_{emission} + P_{noise}$, will depend on both the signal strength and the noise strength.

In swept threshold coding the noise by itself give a sequence of bits containing an equal number of '1's and '0's (assuming threshold level is in the white noise range), while the emitted imposed signal will contribute with more '1's for positive signals or more '0's for negative signals. The important observation is that the integration can be realized in swept-threshold coding simply by counting '1's and the emitted signal value is the difference from the average value. The integration time is simply the number of pulses emitted and can be controlled. Normally a longer integration time will increase quality, but take longer.

The quality of this style of integration is quite sensitive to the threshold level and behavior is different when the threshold is above the white noise level. With stronger signal levels well above the white noise floor signal detection is simpler. The important sweeping of the threshold level enables iterations to an appropriate threshold level adapting to the received energy of the received pulse.

Analysis of swept-threshold sampling compared to pure analog integration as a function of integration level and noise has been performed (figure 3).

Our estimations reduce the standard deviation of the white noise term for the analog average from σ_N to $\sigma_N/\sqrt{n}$ by averaging over n samples. With little noise in the input signal, the swept threshold sampling is not doing too well. The surprisingly good performance, however, with increasing noise levels ($\sigma_N = 0.1$ or 1, 1 being the signal swing), indicates the swept threshold method needs only 10 or even just twice the number of samples of the full analog integrator! These estimates makes the swept-threshold solution quite promising since most real backscattered signal is quite noisy. An interesting observation is that stochastic resonance behavior, only available for large noise components, has the same performance as the swept-threshold sampling.

C. High speed sampling

The simple radar shown in figure 1 is only processing reflecting energy for one strobing time reflecting a fixed distance. Different distances is measured by sweeping the delay and sequentially measure backscattered energy.

With the swept-threshold coding we are able to explore continuous time structures and achieve sampling rates exceeding $20GHz$ exploring delaylines. The samplers are similar to D-latches with all the data inputs (D) connected to the incoming quantized signal. By clocking (or strobing) the latches in a fast sequence, we are able to accumulate reflected energy at several distances for each emitted pulse.

In figure 4 the principle is shown. We are latching the quantized input pulse with a clock input or strobe pulse delayed by small delays (τ) consisting of two inverters. In the current implementation 64 parallel samplers are used and the inverter delay is approximately 21ps with load measured. In this way we are sampling the incoming pulse with a sampling distance of about 43ps which is equivalent to a sampling rate of

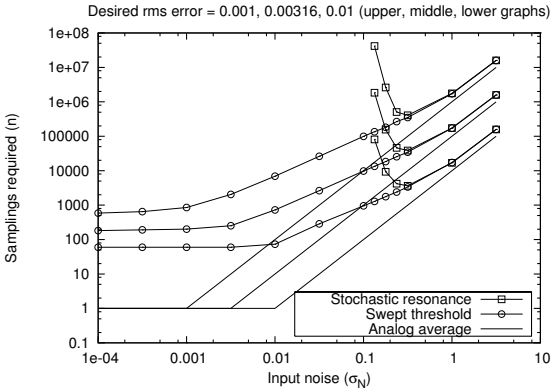

Fig. 3. Comparison of pure analog integration with swept threshhold integration indicates a fairly strong integration to perform adequately. For high noise levels, the swept-threshold technique is doing almost as good as analog integration

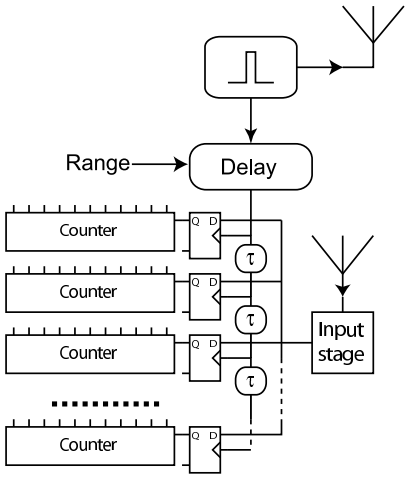

Fig. 4. Delayline sampler and integrators

$\approx 23GHz$ confirmed by measurements. The depth resolution achieved should be less than 9mm assuming speed of light electromagnetic propagation (in most cases the propagation will be slower, increasing spatial resolution).

With the parallel sampler we are sampling at least 41cm depth. In order to adjust further we have added an initial delay (similar to the McEwan radar) before trigging the sampling sequence. This delay is adjustable, again using delaylines. We have one coarse delayline of $\approx 64 \times 1.4ns$ delay elements and a fine tuning delayline of $64 \times 43ps$ delays. By selecting a coarse and a fine tap using multiplexers we may vary the initial delay from 0 to 90ns with 43ps resolution. With this tuning ability, we are theoretically able to detect targets at a range of up to $> 13m$.

As will be shown by measurements all these novel processing techniques are explored to implement a working short range radar in CMOS.

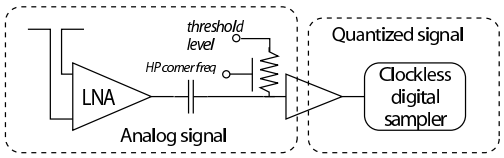

Fig. 5. Input stage of CMOS Impulse Radar consists of an LNA, thresholding element with a buffer amplifier. A simple first order HP filter is included on the input of the thresholding element using a resistive load to the threshold voltage

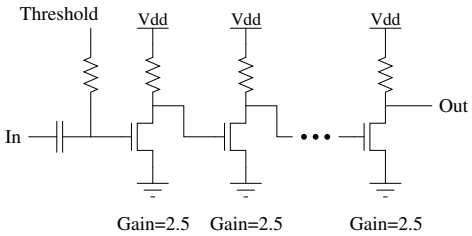

Fig. 6. Common source buffer amplifier consists of eight cascaded common-source passive load amplifiers with highpass tunable filter on the input.

III. Implementation

Based on the principles above we have implemented a prototype of the swept-threshold radar in ST Microelectronics $90nm$ technology. The process was quite new and the design kit lacked some features for doing high quality analog structures.

The input stage (figure 5) of the proposed CMOS radar consists of a low-noise amplifier (LNA) designed for a signal bandwidth of 3.1–10.6GHz. Several LNA topologies in CMOS with required bandwidth are published [7] [8]. Typically 10dB gain is achievable with careful design.

The threshold element is AC coupled to the LNA by a blocking capacitor and by driving the input to the buffer amplifier through an equivalent resistance, both thresholding and high pass filtering is achieved. The resistive element is just a weak transconductance amplifier with tunable tail current setting the corner frequency of the HP filter. Some integration due to parasitic capacitance is unavoidable on the input node of the buffer amplifier.

The buffer amplifier consists of eight cascaded common source amplifiers with a total gain of $2.5^8 = 1526$ (figure 6). We are avoiding the high-gain single stage structures like inverters, since they are hard to linearize for the required frequency band. By using several low-gain common-source amplifiers, we are able to maintain an overall linear gain as indicated in figure 7. A total of eight cascaded common-source amplifiers were used.

The drawback of this approach is significant power consumption, but with emphasis on novel system aspects, the cascaded solution was chosen.

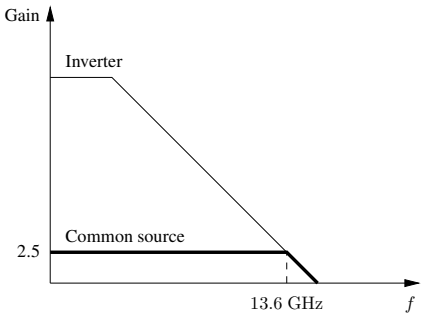

Fig. 7. Buffer amplifiers bode plot indicates the simple idea of controllable flat gain for high frequencies explored in the current prototype.

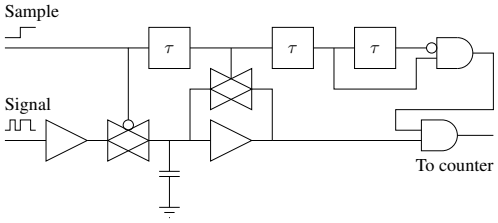

Fig. 8. The sampler consists of several latching elements as the one shown here.

Leaving the pure analog domain the next stage is sampling of the quantized signal. Since the signal may change rapidly, a specialized latching element has been designed (figure 8). The input signal is driven through an open transmission gate and the input voltage is latched when the transmission gate is strobed. The latched voltage is then shaped by turning on positive feedback with another transmission gate over an amplifier driving the value to '0' or '1'. If the latched value is '1', the pulse duration is shaped to $\approx 1ns$. Suitable timing is achieved inserting inverters. The result of the sampler is a $1ns$ counting pulse for the counter (integrator) when a '1' is detected.

As shown in figure 4 the latching elements are all reading the same input, but the strobing signal is generated by a delayline with two inverter ($\approx 23ps$) between samples.

In order to program the sensing range, a digitally controllable initial delay element is included as explained above

The CMOS impulse radar has been implemented in STMicroelectronics 90nm CMOS technology sharing silicon with other projects (figure 9). The actual chip photo is really not interesting due to the large number of covering metal. The first silicon is a feasibility study and no efforts were taken for compact layout. Included in the chip is a pulse transmitter designed for UWB signal generation.

IV. Measured results

The package chip was mounted on a suitable PCB (figure 10) and interfaced through a simple parallel-to-USB card (Elexol USB I/0 24 V3). Software for display on a PC was

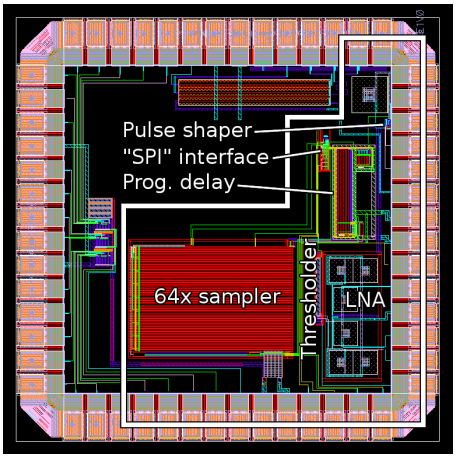

Fig. 9. Layout of the CMOS impulse radar enclosed in the area framed with a white line.

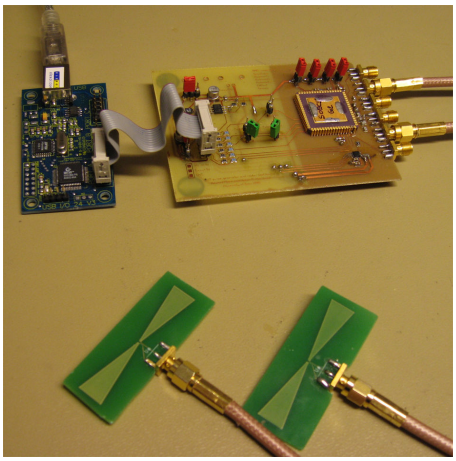

Fig. 10. The measurement setup with the packaged chip mounted on a custom made PCB and interfaced to a PC using a parallel-to-USB interface.

written and quite remarkably we are able to clearly trace simple movements.

In figure 11 the integrator count is indicated by whiter color for high counter values, while black is indicating a low counter values. A hand close to the antennas shows the wave pattern backscattered energy from the hand. When moving the hand away from the radar, the same pattern is shifted rightward as expected. Then again the tracing of moving the hand back again establishing a similar reflection pattern. We are still in an early testing phase, but the first results are already convincing. In our initial experiments we even bypassed the LNA and still get accountable results.

Through these simple experiments all our assumptions are verified and the implemented CMOS impulse radar is a functional system.

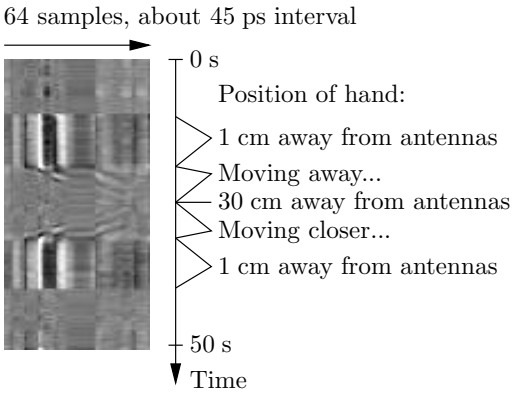

Fig. 11. Measured chip performance moving a hand in front of the radar.

TABLE I
CMOS IMPULSE RADAR SPECIFICATIONS

Measured sampling frequency	23GHz (6.5mm)
Measured initial delay	0–90ns (0–13.5m)
Parallel sampler	64 samples (41cm)
Sampling method	Swept threshold
Signal processing	Quantized continuous time-domain

V. CONCLUSION

A working CMOS impulse radar is designed and implemented in CMOS exploring several novel signal processing techniques.

The swept-threshold coding scheme is making implementation in CMOS feasible exploring time-domain processing. Thresholding of input signals simplifies the integration to digital counters. The high speed, parallel sampling structure using delaylines is giving increased depth resolution over a wide programmable range for every emitted pulse. The estimated system performance is summarized in table I and the proposed CMOS impulse radar is verified in silicon.

REFERENCES

[1] D. L. Black, "An overview of impulse radar phenomenon," in *Aerospace and Electronics Conference, 1992. NAECON 1992., Proceedings of the IEEE 1992 National*, vol. 1, Dayton, OH, May 18–22, 1992, pp. 320–326.

[2] T. E. McEwan, "Ultra-wideband radar motion sensor," U.S. Patent 5 361 070, Nov. 1, 1994.

[3] ——, "Modulated pulse doppler sensor," U.S. Patent 6 426 716, July 30, 2002.

[4] E. M. Staderini, "Uwb radars in medicine," *IEEE Aerosp. Electron. Syst. Mag*, vol. 17, no. 1, pp. 13–18, Jan. 2002.

[5] S. R. Norsworthy, R. Schreier, and G. C. Temes, *Delta-Sigma Data Converters : Theory, Design, and Simulation.* Wiley-IEEE Press, 1997, no. ISBN: 0780310454.

[6] D. G. Luchinsky, R. Mannella, P. V. E. McClintock, and N. G. Stocks, "Stochastic resonance in electrical circuits. ii. nonconventional stochastic resonance," *IEEE Trans. Circuits Syst. II [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, vol. 46, no. 9, pp. 1215–1224, Sept. 1999.

[7] A. Bevilacqua and A. M. Niknejad, "An ultra-wideband cmos lna for 3.1 to 10.6 ghz wireless receivers," in *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, vol. 1, Feb. 15–19, 2004, pp. 382–533.

[8] K. Meisal, C. Limbodal, T. S. Lande, and D. Wisland, "Cmos impulse radio receiver front-end," in *NORCHIP Conference, 2005. 23rd*, Nov. 21–22, 2005, pp. 133–136.

Bibliography

[Anne 99] A.-J. Annema. "Analog circuit performance and process scaling". *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, 1999.

[Gonz 05] D. Gonzalez, L. Parrilla, A. Garcia, E. Castillo, and A. Lloris. "Efficient clock distribution scheme for VLSI RNS-enabled controllers". In: *Integrated Circuit and System Design. 15th International Workshop, PATMOS 2005. Proceedings Lecture Notes in Computer Science Vol. 3728*, 2005.

[Hist] "History of radar". `<http://en.wikipedia.org/wiki/History_of_radar>`. As of 2005-Aug-1.

[Limb 05] C. Limbodal, K. Meisal, T. S. Lande, and D. Wisland. "A spatial RAKE-receiver for real-time UWB-IR applications". In: *2005 IEEE International Conference on Ultra Wideband IEEE Cat. No.05EX1170C*, 2005.

[Maho 05] P. Mahoney, E. Fetzer, B. Doyle, and S. Naffziger. "Clock distribution on a dual-core, multi-threaded Itanium(R)-family processor". In: *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, 2005.

[McEw 94] T. E. McEwan. "Ultra-wideband radar motion sensor". US Patent 5'361'070, 1994.

[Meis 05] K. Meisal, C. Limbodal, T. S. Lande, and D. Wisland. "CMOS impulse radio receiver front-end". In: *Proceedings. Norchip IEEE Cat. No. 05EX1266*, 2005.

[Merc 04] A. Mercha, W. Jeamsaksiri, J. Ramos, S. Jenei, S. Decoutere, D. Linten, and P. Wambacq. "Impact of scaling on analog/RF CMOS performance". In: *Solid-State and Integrated Circuits Technology, 2004. Proceedings. 7th International Conference on*, 2004.

[Moen 06] H. Moen. *UWB Impulse Radio for RFID*. Master's thesis, University of Oslo, Department of Informatics, 2006.

[Siwi 04] K. Siwiak and D. McKeown. *Ultra-Wideband Radio Technology*. John Wiley & Sons Ltd, first Ed., 2004.

[Stoc 00] N. G. Stocks. "Suprathreshold Stochastic Resonance in Multilevel Threshold Systems". *Phys. Rev. Lett.*, Vol. 84, No. 11, pp. 2310–2313, Mar 2000.

[Thum] M. Thumm. "Historical german contributions to physics and applications of electromagnetic oscillations and waves". <http://www.radarworld.org/history.pdf>.