NTNU
Norwegian University of Science and Technology

TFE4540 – Specialization Project

Noise-shaping SAR ADCs: A systematic study, investigation of energy-efficiency, and behavioral simulation

---

Author:
Harald Garvik

December 19, 2014

NTNU – Trondheim
Norwegian University of
Science and Technology

Abstract

A noise-shaping SAR (NS-SAR) is a successive approximation ADC (SAR ADC) where the undigitized residue voltage from the end of each conversion is fed into an analog filter called the loop filter. By means of this filter and oversampling, the quantization noise as well of the noise of the SAR comparator is noise-shaped, like in a delta-sigma modulator. The ADC accuracy is thus increased at the cost of increased sample rate, and extra circuitry to implement the filter. Since the concept of noise-shaping means that not all parts of the circuitry need to be as accurate as the ADC itself, NS-SARs are especially promising for the realization of energy-efficient ADCs in medium/high accuracy regions where other converters suffer from thermal noise.

As the concept of noise-shaping SARs is relatively new, there exists a lot of questions regarding their potential that still remain to be answered. The main aim of this thesis is therefore to do a systematic study of NS-SARs at the architectural level, and especially investigate what to expect from such converters when it comes to energy efficiency. As a basis for this study, a general linear model for NS-SARs is developed and used, as this has not been found anywhere in the literature.

The main tool used to study the NS-SARs is a custom-developed behavioral simulation framework, which computes the ADC accuracy for a given set of design variables by doing a behavioral time domain simulation followed by a FFT. Moreover, the framework has the ability to do such simulations systematically for a large set of design variable combinations, and in this way "map" the NS-SAR design space. The data from such simulations are used together with developed power consumption models to predict the energy efficiency of the converter as a function of design variables. Different kinds of loop filters have been used in the simulations, and all are of relatively low order. Also, the noise transfer functions (NTFs) of the loop filters are automatically synthesized prior to each simulation.

One of the most important results that is found using the simulation framework, is that NS-SARs can operate energy-efficiently at medium/high accuracies, where the energy efficiency of standard SARs are seriously compromised by thermal noise. Of the loop filters that is used in the simulations, the most simple one, having one zero and two poles, turns out to yield the most energy-efficient operation in most practical cases. When it comes to the NTFs of the loop filters, it is generally found that their poles can be used to significantly increase the ADC accuracy without giving rise to unstable operation. Loop filters having more poles than zeros are therefore found to be promising for NS-SAR use. Poles can be realized passively without significant increase of power consumption.

When a target ADC accuracy is given as a design specification, it is not trivial how to choose the architectural level design point (i.e. the combination of bit count in the actual SAR, amount of oversampling, and possibly also loop filter complexity) such that this accuracy is achieved in the most energy efficient manner. The simulation framework can solve this problem for a given target accuracy by means of a search through a set of simulation data. The locations of the found design points is also further investigated in this thesis through the development of a simplified, analytical mathematical model for the energy efficiency. From this model, it is found that an optimal bit count for the SAR exists as a function of the loop filter complexity and the power division between the actual SAR and the loop filter. The results from this model is found accurate enough to function as an approximation formula having practical value during design work.

Contents

Chapter 1

Introduction

In today's compact, high-performing, and often battery powered electronic systems, *energy efficiency* is of primary concern. This is also true for Analog-to-digital converters (ADCs), which are present in almost all of today's electronic devices, working as bridges between the analog world and the digital systems.

In studies of the power consumption in ADCs, thermal noise is found to be the main element setting a lower bound for power consumption in medium to high resolution converters [1–3]. To better understand how this impacts the power consumption of a given ADC, we may first consider the minimum energy E_{min} needed to sample a signal with a given accuracy in the presence of thermal noise. In [1], this minimum energy is given to be $E_{min} = 8kT \cdot \text{SNR}$, where SNR is the wanted ratio between the signal power and the thermal noise power. If we then state the SNR in this expression as *effective number of bits*[1] (ENOB) using the well-known equation $\text{ENOB} = \frac{\text{SNR}_{\text{dB}} - 1.76}{6.02}$, we can obtain

$$E_{min} \propto 2^{2 \cdot \text{ENOB}} \tag{1.1}$$

This important relation shows us that if one want to double the accuracy by increasing ENOB by one, the minimum sampling energy E_{min} quadruples.

To see how these considerations relates to already designed ADCs, we consider a plot from [1], given in figure 1.1. Here, the conversion energy per sample, normalized to E_{min}, is plotted versus accuracy for ADCs reported at the International Solid-State Circuits Conference (ISSCC) and the VLSI Circuit Symposium (VLSI) in the period 1998-2013. Lower bounds are indicated in the figure, and shows us that we have two distinct regimes when it comes to energy efficiency.

In the medium/high accuracy region, the lower bound takes the form of a flat line and is thus proportional to E_{min}. This means that accuracy and energy trades according to equation 1.1 for ADCs operating in this region; That is, the energy quadruples per bit. This clearly suggests that these ADCs have their accuracy limited by thermal noise. Also note that the energy bound turns out to be at $100 \cdot E_{min}$, and not E_{min}. This is because an ADC also uses energy for many more tasks than only the sampling itself, for which the energy bound was derived.

For lower accuracies however, the lower bound is not proportional to E_{min} any more, and the accuracy-energy trade-off is more close to a doubled energy per bit relationship. This is because the energy efficiency of ADCs in this region tends to be limited by the minimum feature sizes of the CMOS process rather than thermal noise [1]. Also, a connected fact is that the energy dissipation

[1] Effective number of bits (ENOB) is a measurement unit for accuracy. If an accuracy is given as a signal-to-noise ratio (SNR), the corresponding ENOB number states how many bits an ideal Nyquist ADC of the same accuracy would have. The relation between number of bits and accuracy in an ideal Nyquist ADC can for instance be found in [8, chapter 15].

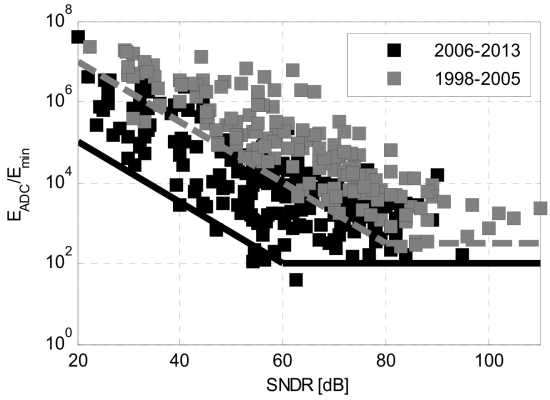


Figure 1.1: Conversion energy per sample, normalized to E_{min}, versus accuracy for ADCs reported at the ISSCC and VLSI [1].

in this regime is often dominated by the digital part of the ADC [2] (whose energy efficiency also is limited by minimum feature sizes).

Interestingly, figure 1.1 also shows that the "knee" between these two regimes tends to move towards lower accuracies over time, meaning that more and more ADCs will operate in the "thermal regime". This trend is easy to understand when we know about the continuously ongoing scaling of CMOS processes being used for electronic devices.

While it is a good thing that scaling makes the non-thermally limited ADCs more and more energy efficient, it also means that more and more ADCs will operate in the thermal regime, where the energy quadruples per bit. This in turn means that for more and more ADCs, energy and accuracy do not trade equally anymore, and energy-efficient systems might thus be forced to use lower-accuracy ADCs. This trend makes it a hot topic to investigate how it is possible to make ADCs that trade energy and accuracy equally in a larger accuracy span. That is, ADCs were accuracy can be significantly decoupled from thermal noise constraints.

1.1 Noise-shaping SAR

In recent years, successive approximation register ADCs (SAR ADCs) have achieved the best energy efficiencies [4, 5]. This makes it interesting to investigate if further SAR architecture improvements can increase the energy efficiency even more, possibly by enabling the above mentioned decoupling from the thermal regime.

The *Noise-shaping* SAR (NS-SAR) is a modified SAR architecture, which to the author's best knowledge was first demonstrated in a taped-out circuit in [6]. Here, the concepts of oversampling and noise-shaping used in delta-sigma ADCs are incorporated into the SAR architecture through the use of a loop filter, resulting in a converter accuracy greater than the number of bits used in the SAR.

The use of oversampling and noise-shaping means that the extra accuracy is achieved by means of increased sampling speed and the use of signal processing techniques, rather than by direct improvement of circuit block accuracies. As thermal noise limitations is related just to the accuracy of circuit blocks, noise-shaping SARs may be promising candidates for more thermal noise immune operation.

1.2 Goals and main contributions

As the concept of noise-shaping SARs is relatively new, there are many open questions regarding its potential. Among others, no systematic study of design trade-offs at the top architectural level has yet been done. Central design parameters at this abstraction level is mainly noise-shaping complexity (for example in terms of modulator order), amount of oversampling, and the number of bits used in the actual SAR. To do an optimal choice of these parameters in terms of maximum energy efficiency is non-trivial.

The goal of this work has therefore been to do a comprehensive exploration of the NS-SAR design space at the architectural level. This is necessary to make the most energy-efficient NS-SARs, and to investigate the limits of the concept itself. More specifically, the fundamental problems that have been addressed are

- How does accuracy and energy trade through the accuracy span? Will the NS-SARs also start to exhibit quadrupled energy per bit behavior at a certain accuracy, or it is possible prevent this?
- Will the NS-SAR be more energy efficient than a normal SAR? If yes, under which circumstances?
- What is the optimal way to design the noise shaping (i.e. the loop filter) to maximize energy efficiency for a given accuracy? Moreover, is one loop filter type always the best choice, or is this a function of e.g. accuracy?
- Given a design spec. (i.e. wanted accuracy and bandwidth), and possibly also a already chosen loop filter, what is then the optimum choice of oversampling rate and number of bits in the SAR itself?

Additionally, an ADC specification of 11.33 bit ENOB (70 dB SNR), 2 MHz bandwidth, and at least 32 MHz sample rate has been of special interest through the work of this thesis. This is because an energy efficient ADC having these specifications will probably be made as a master thesis, based on this work.

Both a theoretical and simulation-based approach has been used to answer questions stated above. Existing theory has been adapted to and extended for the NS-SAR, and an extensive behavioral simulation framework capable of systematic exploration of the design space has been developed. Specifically, the main contributions of this thesis are

- The literature seems to lack a generalized treatment of NS-SARs, independent of concrete topologies. In this thesis, the NS-SAR concept has therefore been generalized, and a general linear model has been derived. This is done in section 3.2.
- The "Kenney-Carley" modulator stability criterion presented in [7] has been re-derived specifically for NS-SARs to verify its validity also in this context. Additionally, the criterion is extended to include the effects of thermal noise in the SAR comparator. These derivations are found in section 4.2.
- Four specific loop filter topologies are considered in the thesis. Three of them are known topologies, whereas one has not been found presented anywhere else. This loop filter has been called "2. order with resonator and extra pole", and is presented in section 4.4.3.
- A comprehensive behavioral simulation framework has been developed in Matlab. This framework represents the specific modulators in a flexible manner, which allows them to easily adapt to the test environment. Specifically, an optimal noise transfer function (NTF)

is synthesized for each loop filter every time a design variable changes. Time domain simulations are run on the different NS-SARs under various conditions, and by sweeping appropriate design variables this results in an "ENOB-map", which gives the ADC accuracy across the design space.

Simple power consumption models are also developed for the NS-SAR, and the combination of these models and the ENOB-map are used to systematically evaluate the energy efficiency as a function of design variables. Moreover, a simple search algorithm called OPT-MOD is developed. This uses the ENOB-map and the power consumption models to find the most energy-efficient design configuration when a design specification is given. All this is described in chapter 6.

- The performance of the chosen NS-SARs/loop filters are evaluated using the simulation framework, and it is among others found that it is of key importance to choose optimal positions for the poles in the noise transfer functions for NS-SARs. These results are given in section 6.1, and also to a certain extent in chapter 4.
- Also by using the simulation framework, energy efficiency is evaluated systematically for all considered NS-SARs, and this is compared to the energy efficiency of a normal SAR. It is found that NS-SARs can yield significantly higher accuracies than normal SARs without entering the thermal regime. Also, normal SARs seems to be best in the low-accuracy region. The results can be found in section 6.2.
- The location of the most energy efficient design points has been explored mathematically through the development of a simplified, analytical energy efficiency model. From this model, a formula for optimum numbers of SAR bits is derived. This is a function of loop filter complexity, and the relative power consumption division between the actual SAR and the loop filter.

  The formula is verified by comparisons to behavioral simulations and its accuracy is found good enough to serve as a simple, practical approximation which can be used during ADC design. The formula, derivation, and verification are found in section 6.2.1.

1.3 Thesis outline

The rest of this thesis is organized as follows:

Chapter 2 - Background Theory Here, the most relevant background theory regarding SAR ADCs, oversampling, and noise shaping is presented.

Chapter 3 - Noise-shaping sar overview A coverage of the NS-SAR architecture in general. The description given is based upon [6], but is also further developed into a generalized NS-SAR representation.

Chapter 4 - Loop filters for noise-shaping SARs First, the noise transfer functions are considered in general, and it is investigated how optimal NTFs should be chosen. Second, four different loop filter architectures are picked and analyzed.

Chapter 5 - The behavioral simulation framework An extensive description of the developed simulation framework.

Chapter 6 - Behavioral simulation results and discussions Results aiming to answer the questions stated in this introduction are presented and discussed.

Chapter 7 - General discussions The work presented in this thesis is discussed in general.

Chapter 8 - Conclusion The thesis is concluded.

Chapter 2

Background Theory

To understand the rest of the thesis, basic knowledge about SAR ADCs, in addition to oversampling and noise-shaping ADCs is needed. An overview of these topics is therefore given in this chapter. All the theory presented should be regarded as well-known and established in the data converter community, and is covered in a large amount of books and papers. Among them are [8, 9].

It is assumed that the reader has basic knowledge about analog circuits and the general principles of analog-to-digital conversion; This is also covered in the above mentioned references.

2.1 Successive approximation ADCs (SAR ADCs)

A successive approximation (SAR) ADC performs a binary search through possible digital values to find the best matching digital word D_{out} for an analog input signal V_{in}. The search is done serially, and controlled by a clock signal. Since a binary search halves the search space per iteration, this means that we are able to decide the value of one more bit in the digital word per clock cycle. Thus, a B bit converter uses B clock cycles for the binary search part of the conversion.

Figure 2.1 shows a block schematic of a conceptual SAR ADC. A digital SAR logic block controls the conversion, an internal D/A converter converts the current state of the binary search to an analog value, and an analog comparator performs the comparisons needed during the search. A sample and hold circuit is also needed to sample V_{in} and hold that value during the the entire conversion.

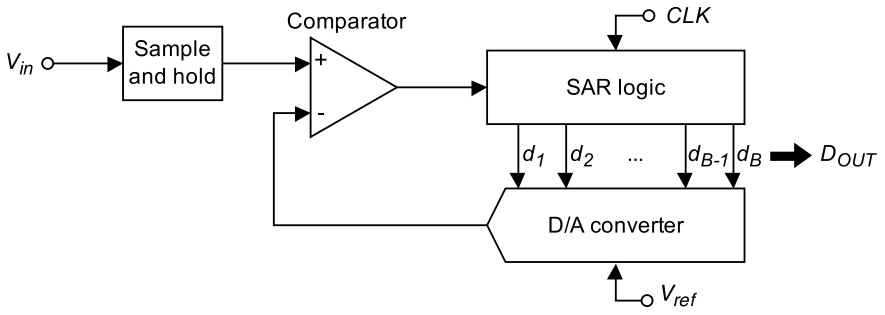


Figure 2.1: SAR conceptual block schematic.

Exactly how this SAR performs the binary search is best explained by the use of an example. Let us define the legal input range as $V_{in} \in [0, V_{ref}]$ (single-ended converter) and consider an input value $V_{in} = 0.7V_{ref}$. The conversion will then proceed as follows:

- In the start of the conversion V_{in} is sampled and output to the positive comparator input. The SAR sets its MSB (most significant bit) d_1 to 1, and leaves the rest at 0. The internal DAC, which also operates in the range $[0, V_{ref}]$ will thus output $0.5V_{ref}$ to the negative comparator input.
- The first comparison is then performed with the result $0.7V_{ref} > 0.5V_{ref} \Rightarrow 1$. The SAR logic then knows that the input is larger than $0.5V_{ref}$, and the d_1 bit is left at 1.
- For the next comparison, d_2 is also switched to one and the DAC now outputs $0.5V_{ref} + 0.25V_{ref} = 0.75V_{vef}$. The comparison now becomes $0.7V_{ref} < 0.75V_{ref} \Rightarrow 0$. The result is that d_2 is switched back to 0.
- When d_3 then is switched to 1, the DAC output is $0.5V_{ref} + 0.125V_{ref} = 0.625V_{ref}$ since d_2 was switched back. The resulting comparison will leave d_3 as one.
- The procedure is repeated until all B bits are determined.

In this example, the ADC was assumed single-ended. A differential SAR will operate after the same principles, and the first conversion will then be used to determine the sign of the signal.

2.1.1 Charge-redistribution SAR

Actual SAR ADCs are often of the charge-redistribution type, shown in a single-ended 4 bit version in figure 2.2. The general D/A converter in the conceptual ADC is now replaced by a charge-redistribution switched capacitor DAC, which also doubles as sample and hold circuit. Moreover, the difference between V_{in} and the DAC voltage is now taken within the DAC itself, and not explicitly by the comparator. This way to reuse the DAC for many different operations means that charge-redistribution SARs are quite simplistic circuits with a minimal of circuitry.

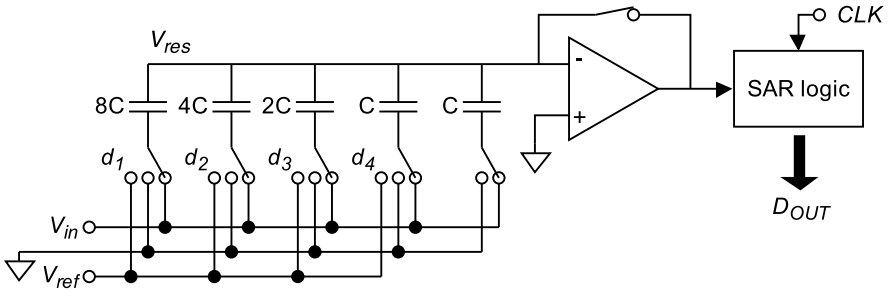


Figure 2.2: A 4-bit charge-redistribution SAR.

A conversion cycle in the charge-redistribution SAR proceeds as follows:

1. **Sample phase** Figure 2.2 shows the ADC in sample mode. The input V_{in} is sampled onto the DAC capacitor bottom plates, while the comparator is held in reset. The reset action ensures that $V_{res} = 0$ V if we assume infinite gain and no offset in the comparator.

2. **Hold phase** The comparator is now taken out of reset (the negative feedback switch is opened) such that the top plate node becomes high impedance. All of the capacitor bottom plates are then switched to ground. The top plate voltage V_{res} is then given by charge conservation

$$-16CV_{in} = 16CV_{res} \Rightarrow V_{res} = -V_{in} \tag{2.1}$$

3. **Bit cycling phase** The binary search may now take place. This is performed by using the bits $d_1 \ldots d_4$ set by the SAR logic to switch individual DAC capacitor bottom plates to V_{ref}. To see how this affects V_{res}, we consider the switching of the MSB capacitor of value 8C to V_{ref}. The charge is still conserved, and charge balance equation now gives

$$-16CV_{in} = 8CV_{res} + 8C(V_{res} - V_{ref}) \Rightarrow V_{res} = -V_{in} + \frac{1}{2}V_{ref} \tag{2.2}$$

We realize that this means that V_{res} is always the difference between the current DAC voltage and V_{in}. This in turn, means that we have $V_{res} < 0$ and positive comparator outputs when the DAC voltage is less than V_{in}, and $V_{res} > 0$ and negative comparator outputs when the DAC voltage is greater than V_{in}. Using this, the necessary comparisons can be carried out.

Note that the rightmost capacitor of value C in the DAC is never switched to V_{ref}, but have to be present such that the MSB capacitor can be half of the total capacitance, the next one a forth, and so on.

Another point which gets important when noise-shaping SARs are introduced in chapter 3, is the actual value of V_{res} after a conversion has completed. The state of the DAC is then set by the final digital output D_{out}, which means that V_{res} holds the difference between the analog version of D_{out} and V_{in}. Mathematically, this is more convenient to state if we regard D_{out} as a V_{ref} normalized value in the interval $[0, V_{ref}]$. That is

$$D_{out} = (2^{-1}d_1 + 2^{-2}d_2 + \ldots + 2^{-B}d_B)V_{ref} \tag{2.3}$$

The final V_{res} can then just be written as

$$V_{res} = D_{out} - V_{in} \tag{2.4}$$

We see that this is the quantization error that has arisen during the A/D conversion.[1]

2.2 Oversampling

In many ADCs, speed can be traded for accuracy by the use of oversampling. To understand how this is possible, it is necessary to examine the quantization error in more detail. Formally, this is the difference between the digital output D_{out}, and an imagined ideal output where no quantization is present (i.e. as in an ADC with infinite resolution and accuracy). This is shown in figure 2.3a, where the black line is the actual D_{out}, and the gray line is the ideal output. For every value of V_{in}, we see that the quantization error is the difference between the black and the gray line.

The digital value is easiest to treat mathematically if we, as discussed in the previous section, regard it as a normalized value directly comparable to V_{in}. That is, D_{out} has also the unit of

[1] Note that it is not necessarily the case that the top plates hold the quantization error after a complete conversion. This is because the last comparison, regarding the LSB, often can be done without switching any capacitor. This is dependent on the switching scheme and DAC setup, which may differ heavily from the basic one presented here. Although the LSB capacitor must be switched in the SAR presented in this section, one must still ensure that it is switched back if LSB equals zero to get the quantization error on the top plates.

volts, and the slope of the gray line in figure 2.3a is equal to 1 (D_{out} will be treated like this in the rest of the thesis unless the opposite is obvious). The quantization error V_Q can then be written, equally as for the SAR top plates, as

$$V_Q = D_{out} - V_{in} \tag{2.5}$$

By examining figure 2.3a, we see that V_Q is bounded by $[-\frac{\Delta}{2}, \frac{\Delta}{2}]$ as long as the input not moves out of range and leaves the output saturated at $D_{out,max}$ (or an equivalent $D_{out,min}$). This condition is called overloading and is always unwanted.

The parameter Δ is simply the distance between two adjacent output levels and can for a B bit ADC be defined as

$$\Delta = \frac{\text{Full-scale input range (FSR)}}{2^B} = \frac{V_{ref}}{2^B} \tag{2.6}$$

where the last equality is only valid if we assume that the input range is $[0, V_{ref}]$.

From equation (2.5), we see that V_Q is a deterministic signal dependent on V_{in}. If we, however, assume that V_{in} varies quite rapidly, then V_Q turns out to approximately take the form of uniformly distributed additive white noise. We therefore usually choose to model the quantization error as an additive white noise source E. As we know that the quantization noise is bounded by $[-\frac{\Delta}{2}, \frac{\Delta}{2}]$ (if we assume no overload), we can sketch the probability density function for E. This is done in figure 2.3b. The noise power is then given by

$$P_E = \int_{-\infty}^{\infty} e^2 f_E(e)\, de = \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} e^2\, de = \frac{\Delta^2}{12} \tag{2.7}$$

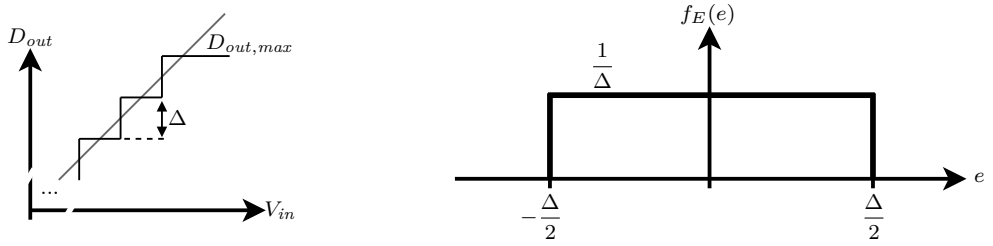
We also want to know the noise power spectral density $S_E^2(f)$, which is related to the noise power through the relation $\int_{-\infty}^{\infty} S_E^2(f)\, df = P_E$. As we assume that the noise is white, we know that the power spectral density is flat. Moreover, no energy can lie outside $[-f_s/2, f_s/2]$ since we operate in discrete time. The resulting spectrum is shown in figure 2.4a, and the value for $S_E^2(f)$ have to be

$$S_E^2(f) = \frac{\Delta^2}{12 f_s} \tag{2.8}$$

to satisfy

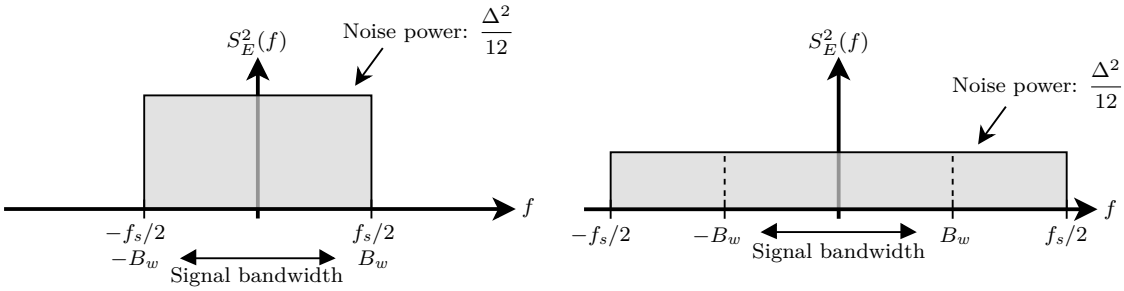
$$P_E = \int_{-f_s/2}^{f_s/2} S_E^2(f)\, df = \frac{\Delta^2}{12} \tag{2.9}$$

It is now possible to explain oversampling through the use of the noise power and noise spectral density. First consider figure 2.4a, which shows $S_E^2(f)$. Also indicated in the figure is the bandwidth of the signal to be digitized, ranging from $-B_w$ to B_w. We see that this coincides with the



(a) Top of ADC transfer characteristic. **(b)** Quantization noise probability density function.

Figure 2.3: Definition of the parameter Δ in an ADC transfer characteristic, and the associated probability density function for the quantization noise.



(a) OSR = 1 **(b)** OSR = 2

Figure 2.4: Noise spectral densities with and without oversampling.

range $[-\frac{f_s}{2}, \frac{f_s}{2}]$. The converter is thus operating at the Nyquist rate[2] $f_s = 2B_w$. The noise power which falls in the signal band is thus seen to be $\frac{\Delta^2}{12}$.

In figure 2.4b, the sample rate is doubled while the signal bandwidth is retained. The total noise power is the same, because it is not dependent on f_s. The spectral density which is inversely proportional to f_s is halved, however. It is then evident from the figure that the noise power inside the signal band is also halved. This means that if we use a digital post-filter to take away the quantization noise that lies outside the signal band, the signal to noise ratio is increased compared to the nyquist-rate case. This is the advantage of oversampling.

To quantitatively establish the advantage of oversampling, we integrate the noise spectral density in the signal band. We then get the in-band noise power, which will be the total noise power in the final output if we assume that we can take away all the out-of-band noise. We get

$$P_E = \int_{-B_w}^{B_w} S_E^2(f)\, df = \frac{\Delta^2}{12 f_s} 2B_w = \frac{\Delta^2}{12\,\mathrm{OSR}} \tag{2.10}$$

where we have also introduced the oversampling rate OSR, which tells how many times faster than the nyquist rate the converter operates. That is

$$\mathrm{OSR} = \frac{f_s}{2B_w} \tag{2.11}$$

Now it is also interesting to establish a signal to noise ratio as a function of OSR. We do that by using the already calculated P_E and the power of a full range sinusoid. The full input range of the converter is generally given as $2^B\Delta$ (see equation 2.6) and the sinusoid amplitude is thus the half of this. This yields a signal power of

$$P_S = \left(\frac{2^B\Delta}{2}\frac{1}{\sqrt{2}}\right)^2 = \frac{\Delta^2 2^{2B}}{8} \tag{2.12}$$

We can then calculate the signal to quantization noise ratio (which we call it since only quantization noise is considered) as

$$\mathrm{SQNR} = 10\log\left(\frac{P_S}{P_E}\right) = 10\log\left(\frac{3}{2}\mathrm{OSR}\,2^{2B}\right) = 6.02B + 1.76 + 10\log(\mathrm{OSR}) \tag{2.13}$$

This shows that the SQNR is increased by 3 dB every time OSR is doubled. It is even more instructive to express the SQNR in terms of ENOB (effective number of bits) using the relation

[2]Note that no practical AD converter can operate exactly at the Nyquist rate, because this would require an infinitely sharp anti-aliasing filter to band-limit the input at exactly B_w. It is however easier to discuss oversampling if we allow this in the discussions.

$\text{SNR}_{\text{dB}} = 6.02\,\text{ENOB} + 1.76$. Inserting for SQNR in equation 2.13 and converting to base 2 logarithm, we get

$$\text{ENOB} = B + \frac{1}{2}\log_2(\text{OSR}) \tag{2.14}$$

This equation shows that we get 0.5 extra ENOB of accuracy for every doubling of the OSR, or for every OSR octave. The advantage from oversampling alone is thus quite small. That is, to increase the ENOB significantly, it is probably necessary to use an unreasonable high OSR.

2.3 Noise-shaping ADCs (Delta-sigma modulators)

For oversampling to be attractive, we need a way to increase the amount of extra ENOB acquired per OSR octave. This can be done by introducing a loop filter $H(z)$ in front of the ADC, and a negative feedback loop around the whole system. This results in a system called a delta-sigma modulator[3], and is shown in figure 2.5. Note that an extra D/A converter is also needed since the feedback goes from the digital to the analog domain.

To understand how this system can improve the oversampling concept, assume that the loop filter has very high gain in the signal band. The structure then gets quite analogous to an op-amp in unity gain configuration, and it can be understood that the input will pass quite unaffected through to the output. The quantization noise which is introduced in the actual ADC will, however, be attenuated by the feedback structure, in the same way as noise in an op-amp output stage is also attenuated by feedback.

To define this effect more formally, it is easiest to use a linear model of the delta-sigma modulator. This can be obtained if we assume that the quantization error is white noise, as we can then just swap the ADC block with a noise source $e(n)$. This is done in the modulator in figure 2.6, which is now an entirely linear model. The system has now two independent inputs (the signal and the quantization noise) and can thus be described by transfer functions from each input to the output. The output can be written down in the z-plane as

$$D_{out}(z) = \frac{1}{1+H(z)}E(z) + \frac{H(z)}{1+H(z)}U(z) \tag{2.15}$$

From this, we define the *signal transfer function* (STF) and the *noise transfer function* (NTF) as

$$\text{STF}(z) = \frac{D_{out}(z)}{U(z)} = \frac{H(z)}{1+H(z)} \tag{2.16}$$

[3]All of the terms ADC, *modulator* and *loop filter* are used extisively throughout this thesis. Noise-shaping ADCs, and thus NS-SARs can be called modulators, and the words ADC and *modulator* are therefore used interchangeably when refering to NS-SARs. The term *loop filter* does generally refer to the loop filter alone, and not the whole ADC.

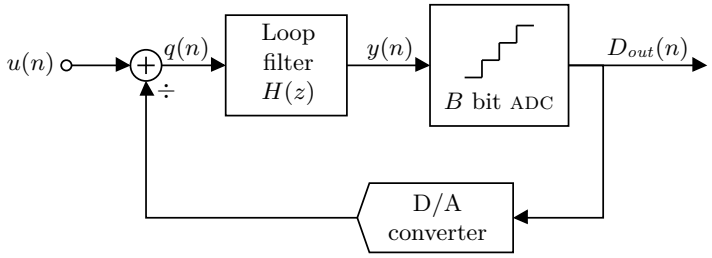


Figure 2.5: Delta sigma modulator

and

$$\mathrm{NTF}(z) = \frac{D_{out}(z)}{E(z)} = \frac{1}{1 + H(z)} \tag{2.17}$$

This shows that the quantization noise gets attenuated and the signal passes through if $H(z)$ is large in the signal band.

The loop filter needs to contain at least one sample of delay [10, chapter 4], and the obvious choice of $H(z)$ as a pure gain block is therefore not possible. This means that we have to select $H(z)$ as a low pass filter with high gain in the signal band to get a desirable NTF and STF. Outside the signal band $H(z)$ will start to roll off and the noise attenuation will disappear, or even change to noise amplification. This means that the noise inside the signal band is sort of moved to the outside of the signal band. The noise spectrum will thus not be flat anymore, and we therefore refer to the whole concept as *noise shaping*.

To find the new value for the in-band noise power, the noise spectral density is integrated in the signal band while taking the filtering into account. This yields

$$P_E = \int_{-B_w}^{B_w} S_E^2(f)|\mathrm{NTF}(f)|^2\,df = \frac{\Delta^2}{12 f_s}\int_{-B_w}^{B_w} |\mathrm{NTF}(f)|^2\,df \tag{2.18}$$

The task left is to choose a sensible $H(z)$, which gives good in-band noise attenuation while not being to complex to implement as a circuit. A common choice is to choose $H(z)$ to be a delaying discrete time integrator. That is

$$H(z) = \frac{z^{-1}}{1 - z^{-1}} \tag{2.19}$$

Integrators always have a pole at DC, which means that the NTF will get a zero at DC. This is because further inspection of equation 2.17 reveals that $H(z)$ poles always yield NTF zeros. Inserting for $H(z)$ in (2.17) we specifically get

$$\mathrm{NTF}(z) = \frac{1}{1 + \frac{z^{-1}}{1-z^{-1}}} = 1 - z^{-1} \tag{2.20}$$

We recognize this as a discrete time differentiator, which is a high pass filter. The in-band noise, residing in the lower end of the frequency range is thus attenuated. The corresponding frequency response magnitude can be found as

$$|\mathrm{NTF}(f)| = 2\sin(\pi f) \tag{2.21}$$

where f is normalized frequency. This frequency response has its minimum (which is zero) at DC, and its maximum value at $f = 0.5$. This is as expected from a high pass filter.

We can also obtain the STF by inserting into equation 2.16. This yields

$$\mathrm{STF}(z) = z^{-1} \tag{2.22}$$

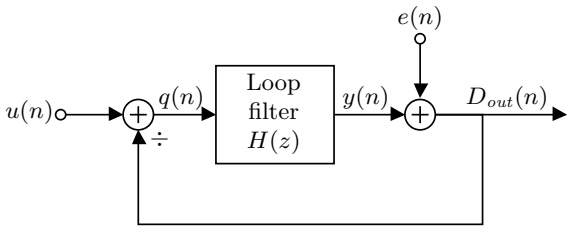


Figure 2.6: Delta sigma modulator linear model.

which is a unit delay that not destroys the signal integrity.

To improve the noise shaping even more, an approach is to choose $H(z)$ such that higher order differentiation is achieved. Explained simply, this corresponds to cascading of more integrators in the circuitry. A more general differentiator NTF can then then given as

$$NTF(z) = (1 - z^{-1})^L \tag{2.23}$$

where L is the filter order. The frequency response will be L times that in equation (2.21). It also turns out that L integrators are needed to realize an L-th order filter. We can then evaluate 2.18 to find the in-band noise power of a modulator using a differentiator NTF of order L. An approximate solution when OSR $>>$ 1 is given in [9] as

$$P_E = \frac{\Delta^2}{12 f_s} \int_{-B_w}^{B_w} |\mathrm{NTF}(f)|^2 \, df \approx \frac{\Delta^2}{12} \frac{\pi^{2L}}{(2L+1)\mathrm{OSR}^{2L+1}} \tag{2.24}$$

which shows that P_E decreases by a factor 2^{2L+1} for every OSR octave. As for the pure oversampling case, we can use this result to obtain an expression for the SQNR stated as ENOB when a full scale sinusoid is used as input. This can be found as

$$\mathrm{ENOB} = B + \frac{1}{2}\log_2\left(\frac{2L+1}{\pi^{2L}}\right) + \left(L + \frac{1}{2}\right)\log_2(\mathrm{OSR}) \tag{2.25}$$

Here, the last term which is dependent on OSR is the most important. That shows that we now get $L + 0.5$ bit per OSR octave, which can give rise to a substantial oversampling advantage.

Although practical loop filter design is most often not as easy as to just pick a L-th order differentiator NTF, these simple noise transfer functions are still extensively used when discussing delta-sigma modulators. This is because they are quite simple, and show coarsely what performance you can expect from a delta-sigma modulator of order L.

One last thing to mention is that a more general delta-sigma modulator model does not have the structure of 2.6, but rather that of figure 2.7 [10, chapter 4]. This is because a general loop filter $H(z)$ does not only take $q(n)$ (the difference between the input and the output) as input. It will rather have arbitrary entry points for $u(n)$ and $D_{out}(n)$ and therefore needs to be represented by a general filter block having two transfer functions; L_0 from $u(n)$ to $y(n)$ and L_1 from $D_{out}(n)$ to $y(n)$. From this, general expressions for NTF and STF can be obtained as function of L_0 and L_1.

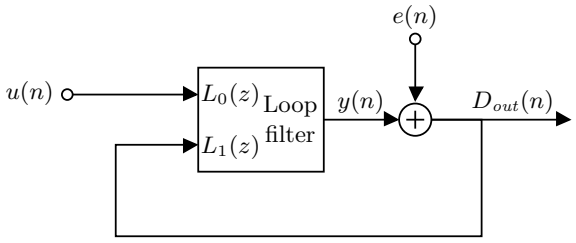


Figure 2.7: Delta sigma modulator with general loop filter.

Chapter 3

Noise-shaping SAR overview

The actual ADC performing all the quantizations in a delta-sigma modulator can in principle be of any type, including SAR. The most straight-forward way to use a SAR to do quantizations in a delta-sigma modulator would be to just swap the ADC block in the general delta-sigma structure with a SAR, as shown in figure 3.1. Considering the figure, we realize that when doing this, we still need a separate DAC outside the SAR to implement the feedback from the digital to the analog domain. Taking this into account, we see that the overall amount of circuitry in the modulator will probably be substantially larger than for the SAR alone. This makes energy efficient design more demanding.

It is, however, possible to use a SAR in a delta-sigma modulator in a much more efficient way, by realizing that the signal $q(n)$ that enters the loop filter in figure 3.1 is actually the negative of the final SAR residue voltage V_{res}, which can be obtained from the SAR DAC top plates after a completed conversion. If we utilize this fact, it is not necessary to have an extra DAC in the feedback loop, and the system complexity immediately comes closer to that of the SAR alone. In simple terms, we can say that this approach is more like adding delta-sigma techniques to the inside of the SAR structure, rather than to just include a SAR in a delta-sigma modulator. It is thus natural to call delta-sigma modulators of this form for noise-shaping SARs (NS-SAR).

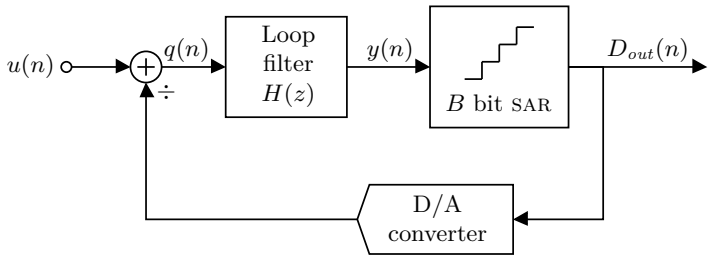


Figure 3.1: Straight-forward way to use a SAR in a delta-sigma modulator.

To the best of the author's knowledge, the first and only NS-SAR designed and fabricated to this date is a 10.0 ENOB one presented by Fredenburg and Flynn in [6]. Before this, the general idea of exploiting the top plate residue voltage of the SAR was presented by Kim, Kim and Cho in [11].

In [6], the most simple way to do noise-shaping in a SAR is presented initially, and this is also done in this overview as an instructive example. After this, we will generalize the NS-SAR structure and establish a general linear model in addition to general expressions for the NTF and STF. This will serve as the base for the NS-SAR loop filter discussions in chapter 4. This generalization is not done in [6], because this paper focuses more on a concrete NS-SAR implementation rather than on

general discussions. Lastly, we will look into what happens to the thermal noise in the different blocks of the SAR when noise shaping is introduced. This topic is also extensively treated in [6].

3.1 Simple noise shaping in a SAR

In figure 3.2, simple noise shaping is added to the 4-bit charge-redistribution SAR discussed in section 2.1.1. Specifically, the negative of the final value of the residue voltage $-V_{res}$ is sampled onto a capacitor after a completed conversion $n-1$, and held there during the consecutive conversion n. The capacitor is connected to the comparator positive terminal, meaning that comparisons during the binary search are now on the form

$$-V_{in}(n) + V_{D/A} < -V_{res}(n-1) \tag{3.1}$$

where $V_{D/A}$ is just the current DAC voltage at some point of the binary search. If we rearrange (3.1) into

$$-[V_{in}(n) - V_{res}(n-1)] + V_{D/A} < 0 \tag{3.2}$$

we see that the value $-V_{res}(n-1)$ on the capacitor is added to the input, and therefore also to the output. The output can then be written down as

$$D_{out}(n) = V_{in}(n) - V_{res}(n-1) + V_Q(n) \tag{3.3}$$

where $V_Q(n)$ is the quantization error of the current sample n. If we then use the expression found in section 2.1.1 for the residue voltage

$$V_{res}(n) = D_{out}(n) - V_{in}(n) \tag{3.4}$$

and put this into equation (3.3), we get

$$D_{out}(n) = V_{in}(n) - D_{out}(n-1) + V_{in}(n-1) + V_Q(n) \tag{3.5}$$

Finally, we take take the z-transform and rearrange to get

$$D_{out}(z) = V_{in}(z) + \frac{1}{1+z^{-1}} V_Q(z) \tag{3.6}$$

From this we see that the input still passes straight through to the output, while the quantization error is shaped by the term $\frac{1}{1+z^{-1}}$, which turns out to be a simple high pass filter. Noise shaping is thus achieved by modifying the SAR in this way. The maximum noise attenuation in the signal band is in this case only -6 dB, and this quite poor performance will not lead to attractive bandwidth-accuracy trade-offs when it comes to energy efficiency [6].

3.2 Generalization of the noise-shaping SAR

If the $-V_{res}$ sample capacitor (see figure 3.2) in the simple NS-SAR is exchanged for some kind of loop filter, more effective noise shaping will be achieved. We can formalize this idea for an arbitrary loop filter $H(z)$ by deriving a general NS-SAR linear model, and a general NS-SAR NTF and STF. When having this at hand, further exploration of possible filters $H(z)$ is possible.

To obtain the model, NTF and STF, we first consider figure 3.3a, which shows a linear model for the simple noise-shaping SAR. The model is quite simple; The sum of $-V_{res}(n-1)$ and $V_{in}(n)$ is digitized into the signal $D_{out}(n)$, and a white noise source $e(n)$ is used to represent the quantization errors. This is in accordance to what was derived for the NS-SAR circuit in section 3.1.

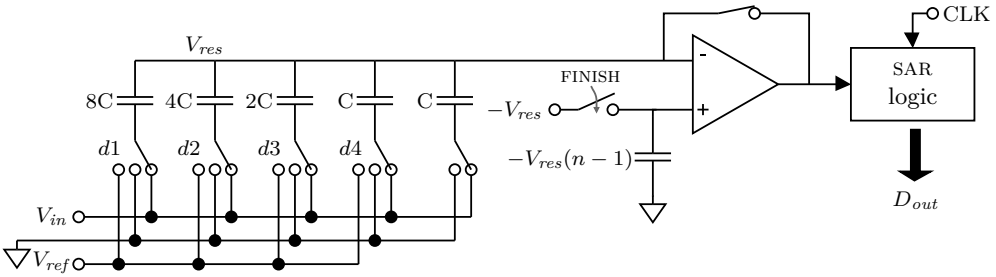
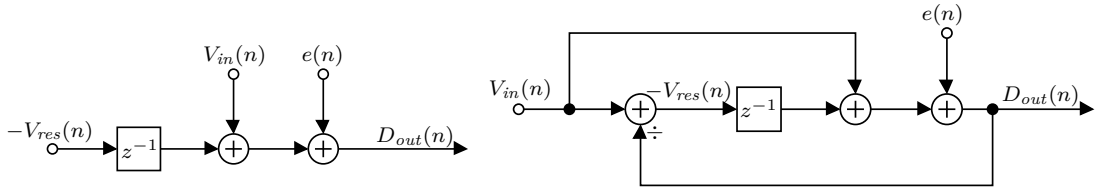


Figure 3.2: A simple noise shaping SAR adding the previous residue to the current output.



(a) Shown as consistent as possible with respect to the physical circuit.

(b) Logical signal flow paths for V_{res} added.

Figure 3.3: Linear models for the simple NS-SAR.

If we then express $-V_{res}(n)$ in this model in terms of $V_{in}(n)$ and $D_{out}(n)$ by drawing some extra arrows, we arrive at the model in figure 3.3b. This model starts to resemble that of a delta-sigma modulator (was given in figure 2.6), but with a specific loop filter z^{-1} rather than a general loop filter $H(z)$, and an extra feed forward from $V_{in}(n)$ to the quantizer input. We now realize that a generalization of the model can be made by just changing the loop filter function z^{-1}, realized by the $-V_{res}$ sampling capacitor, with a general loop filter $H(z)$. This gives the model in figure 3.4, which is the general NS-SAR linear model used in the rest of this work. Here, the signal names are also changed to those used for the modulators in chapter 2.

We can now obtain the general NS-SAR NTF and STF. By using the model, the output can be written down in the z-plane as

$$D_{out}(z) = U(z) + \frac{1}{1+H(z)}E(z) \tag{3.7}$$

which gives

$$\mathrm{STF}(z) = \frac{D_{out}(z)}{U(z)} = 1 \tag{3.8}$$

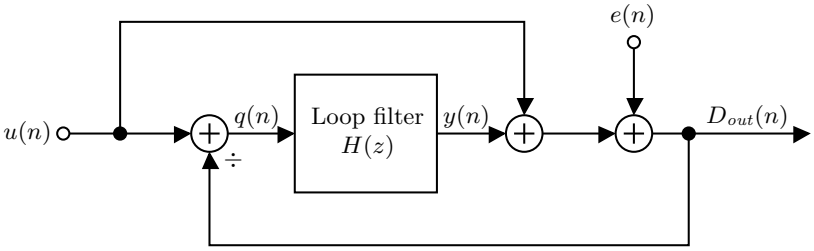


Figure 3.4: General NS-SAR linear model.

and

$$\mathrm{NTF}(z) = \frac{D_{out}(z)}{E(z)} = \frac{1}{1 + H(z)} \tag{3.9}$$

This means that the NTF of the NS-SAR structure is equal to the general delta-sigma NTF, while the STF is always unity because of the feed-forward in the model. This is a very desirable property, because the loop filter input and output then takes the form

$$Q(z) = -\frac{1}{1 + H(z)} E(z) \tag{3.10}$$

$$Y(z) = -\frac{H(z)}{1 + H(z)} E(z) \tag{3.11}$$

which is not dependent on the modulator input $U(z)$. This means that the signal swing inside $H(z)$ can be kept small if $E(z)$ is made small by the choice of a reasonable high bit count B in the SAR. The circuitry complexity of $H(z)$ can thus be held simpler. This advantage of an unity NTF was probably first pointed out in [12], and is also commonly utilized in standard delta-sigma modulators by the introduction of the same feed-forward path as in the NS-SAR model.

One more thing to mention about the NS-SAR model is that the feedback from D_{out} is not a physical signal inside the NS-SAR, because the residue signal $q(n)$ is generated directly by the internal SAR DAC. This means that if further generalizations of the model should be done, then $D_{out}(z)$ should probably not be allowed to enter into nodes of the loop filter on its own, as this will imply that an analog version of $D_{out}(n)$ has to be created explicitly.

3.3 Thermal noise in the noise-shaping SAR

As stated in the introduction, one of the main aims of this work is to decouple the ADC accuracy from thermal noise constraints. How thermal noise in the NS-SAR affects the accuracy is hence of primary concern. This topic is treated extensively in [6], which is thus the basis for the discussions in this section.

The first and most important thing to point out is that the NS-SAR also noise-shapes the comparator noise. This can be seen by realizing that the comparator noise will enter into the linear model in figure 3.4 through the same terminal as the quantization noise $e(n)$. The comparator noise will thus be treated similarly as the quantization noise by the modulator, and thus noise-shaped. This means that the comparator noise does not need to be lowered when noise-shaping is introduced to a SAR.

This noise-shaping of the comparator noise is more or less pointed out as the main advantage of the NS-SAR in [6], and this is probably indeed the case since the comparator often dominate the power consumption at medium to high ADC resolutions [2]. This means that if the comparator had to be made to the accuracy of the whole NS-SAR, then any energy-efficiency increase would probably not have been possible compared to a pure SAR.

Noise in the DAC is also present in a SAR, and its noise power is given by the common formula $\frac{kT}{C}$, where C is the total DAC capacitance (k is Boltzmann's constant and T is absolute temperature). This noise enters into the NS-SAR model at the same terminal as the input, and is thus not noise-shaped. The oversampling will, however, attenuate the DAC noise, but only by 3 dB per OSR octave.[1] This means that it might be necessary to increase the DAC capacitance to achieve the wanted accuracy, and this will thus lead to higher power consumption. Whether this is a deal-breaker for energy efficiency improvements or not, depends on if the DAC dominates the overall SAR

[1] Equivalent to the quantization noise in a oversampling converter, discussed in section 2.2.

power consumption. If we turn to [2], this is at least not the case in the SAR power consumption model presented there.

Chapter 4

Loop filters for noise-shaping SARs

The main aim of this chapter is to describe the NTFs and loop filter structures chosen for test in this work, and to give the reasoning behind the choices. First the NTF is accounted for on its own, and important restrictions on its form in addition to its effect on modulator stability are discussed. This consecutively leads to a section where poles and zeros for the the NS-SAR NTFs are considered.

After this, four specific loop filter structures chosen for the NS-SAR simulations will be presented. Structure-specific NTFs will be given in terms of the filter coefficients, and mappings between the NTF poles and zeroes and the coefficients will be derived.

4.1 NTF restrictions

In section 2.3, it was stated that the loop filter needs to contain at least one sample of delay. This must also true for the NS-SAR, because the opposite would mean that a current output $D_{out}(n)$ could depend on the current $V_{res}(n)$, which is not available before $D_{out}(n)$ is available.

While this is clear, it is not equally easy to see from an arbitrary NTF transfer function if there will be at least one sample of delay through all the branches of the loop filter. We therefore use the delay requirement to derive an equivalent restriction for the NTF, in the same way as it is done in [10, chapter 4].

Consider the NS-SAR linear model, repeated in figure 4.1. Assume that all signals are initially zero, and that $e(n)$ is set equal to 1 at sample $n = 0$. If the loop filter fulfills the delay requirement, $D_{out}(0) = e(0) + y(0) + u(0) = e(0) = 1$ must hold because $y(n)$ was initially zero, and cannot instantaneously change because of the delay inside $H(z)$ ($u(n)$ is just held at zero all the time in this derivation). This in turn, dictates that the NTF impulse response $ntf(n)$ have to equal 1 at $n = 0$. Else, there is no way $D_{out}(0) = e(0)$ can hold. So summed up, the requirement $ntf(0) = 1$ must be satisfied in the time domain.

We can obtain an equivalent requirement in the z-domain if we write the NTF in terms of its impulse response. That is

$$\mathrm{NTF}(z) = ntf(0) + ntf(1)z^{-1} + ntf(2)z^{-2} + ntf(3)z^{-3} + ... \tag{4.1}$$

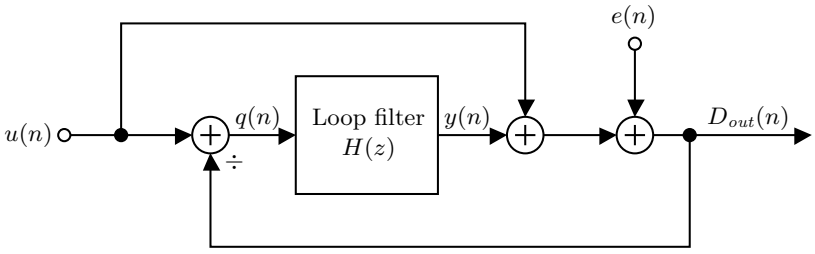


Figure 4.1: The general NS-SAR linear model from chapter 3.

From this we realize that $ntf(0) = 1$ translate into the requirement $\mathrm{NTF}(\infty) = 1$.

Then we consider a general rational NTF on the form

$$\mathrm{NTF}(z) = \frac{b_m z^m + b_{m-1} z^{m-1} + ... + b_0}{a_n z^n + a_{n-1} z^{n-1} + ... + a_0} \tag{4.2}$$

For this to evaluate into something other than 0 or ∞ when $z \to \infty$, the polynomials have to be of the same degree. That is $m = n$. If we then invoke the $\mathrm{NTF}(\infty)$ requirement, we get

$$\mathrm{NTF}(\infty) = \frac{b_n}{a_n} = 1 \tag{4.3}$$

which must be fulfilled for all realizable NTFs.

We often write the NTF on "delay-form" where we only use negative exponents of z. It is thus useful to restate the results from [10, chapter 4] for this situation. The equivalent requirement is then that the highest order term in both numerator and denominator is the z-independent one, and it have to be equal in both numerator and denominator. That is

$$\mathrm{NTF}(z) = \frac{\alpha + b_{n-1} z^{-1} + b_{n-2} z^{-2} + ...}{\alpha + a_{n-1} z^{-1} + a_{n-2} z^{-2} + ...} \tag{4.4}$$

where $\alpha = b_n = a_n$.

This restriction means that is out of the question to put some constant factor in front of the rational NTF. This in turn, means that it is not possible to normalize it at some appropriate point in the frequency response. This is a serious disadvantage because NTF optimizations performed to get low signal band gain might lead to high out-of-band gain, which can not easily be prevented without some way to normalize. While this high noise gain outside the signal band do not compromise the signal itself, it nevertheless leads to modulator stability problems. This is discussed in the next section.

4.2 Modulator stability

Stability theory of linear systems is well established, and one should hence believe that it is an easy task to ensure modulator stability. However, it is important to not forget that despite the use of linear models, noise-shaping modulators are in fact not linear systems. The quantizer is inherently non-linear, and the white noise assumption used to linearize it is an approximation. Moreover, the circuitry that constitutes the loop filter will not act linearly under all conditions.

The linear system assumption always breaks down when some component in the feedback loop starts to saturate. Most often, this will be the quantizer that enters overload [10, page 98].

When this happens, the white noise assumption breaks down and the quantizer will only output a constant value regardless of the input. This condition can then lead to subsequent saturations in the loop filter integrators, and the whole system may enter a overall saturation state which might not be possible to exit without a system reset.

Knowing this, a very safe criterion to set to prevent instability is that quantizer overload should never happen. In [7], Kenney and Carley used this criterion as a basis to derive an equivalent criterion for the NTF, or specifically the L_1 norm $\sum |ntf(n)|$ of the impulse response. This criterion is relatively simple, and is widely used for multi-bit delta-sigma modulators.[1] It has also been chosen to use this criterion for the NTFs made in this thesis, and it will thus be derived in detail in the following. As opposed to [7] which considers a standard delta-sigma modulator, this derivation is based upon the NS-SAR structure. It is thus proven that the criterion is indeed valid for NS-SARs.

The task at hand is to ensure that the quantizer never overloads. Looking at the module in figure 4.1, we realize that this means that the sum $u(n) + y(n)$ have to be bounded to the valid quantizer input range, called FSR. Although the input $u(n)$ is directly under our control, the filter output $y(n)$ will always be some sort of accumulation of previous residue signals $q(n)$. Its value is therefore not easily determined.

It is still possible to establish a bound for $y(n)$. We first rewrite the expression given for $Y(z)$ in equation (3.11) in terms of the NTF. That is

$$Y(z) = -\frac{H(z)}{1 + H(z)} E(z) = (\mathrm{NTF}(z) - 1)\, E(z) \tag{4.5}$$

In the time domain, this is equivalent to the convolution

$$y(n) = ntf(n) * e(n) - e(n) = \sum_{i=0}^{\infty} ntf(i)\, e(n-i) - e(n) \tag{4.6}$$

If we then also use the fact $ntf(0) = 1$, derived in section 4.1, equation (4.6) can be simplified to

$$y(n) = \sum_{i=1}^{\infty} ntf(i)\, e(n-i) \tag{4.7}$$

We know from section 2.2 that $e(n)$ is bounded by $[-\frac{\Delta}{2}, \frac{\Delta}{2}]$ if we assume no quantizer overload. The largest possible value for the $y(n)$ convolution is therefore obtained when

$$\begin{aligned} e(n-i) &= \frac{\Delta}{2} \ \text{ for } ntf(i) > 0 \\ e(n-i) &= -\frac{\Delta}{2} \ \text{ for } ntf(i) < 0 \end{aligned} \tag{4.8}$$

Similarly, the equivalent smallest possible value is obtained when the inequality signs of equation (4.8) are flipped. This means that the magnitude of $y(n)$ is always bounded by the relation

$$|y(n)| \leq \frac{\Delta}{2} \sum_{i=1}^{\infty} |ntf(i)| = \frac{\Delta}{2} \left(\sum_{i=0}^{\infty} |ntf(i)| - 1 \right) \tag{4.9}$$

as long as no overload is present. The last rewriting in equation (4.9) was done because it is convenient to state the criterion in terms of the entire impulse response.

[1] The criterion is not applicable for modulators having only one bit in the quantizer. This is because overload cannot be explicitly defined when only one transition exists in the ADC transfer characteristic.

It is now possible to use worst case peak-to-peak values for both $y(n)$ and $u(n)$ to write a conservative criterion which ensures no quantizer overload. That is

$$U_{\text{p-p},max} + \underbrace{\Delta \left(\sum_{i=0}^{\infty} |ntf(i)| - 1 \right)}_{Y_{\text{p-p},max}} \leq \text{FSR} \tag{4.10}$$

where $U_{\text{p-p},max}$ is the maximum input peak-to-peak value and FSR is the quantizer full-scale range. Also note that the worst-case peak-to-peak value for $y(n)$ was set to twice that of equation (4.9), which only gives the worst-case magnitude.

This criterion shows that there exists a trade-off between the chosen maximum input value and the loop filter design, which will set the value of $\sum |ntf(n)|$. Additionally, Δ can also be changed by changing the number of bits B in the quantizer. These trade-offs will be discussed in more detail in section 4.3.2.

An interesting observation is that if we set $U_{\text{p-p},max}$ equal to FSR, then $\sum |ntf(n)|$ must be equal to $ntf(0) = 1$ to not violate the criterion. An impulse response like this implies that no noise shaping takes place at all, and a smaller value must thus be chosen for $U_{\text{p-p},max}$ to get a sensible system. This seems logical because the output $y(n)$ from the loop filter must also be "allowed to take some of the place" in the quantizer input range.

Equation (4.10) only takes quantization noise into account, likewise as in [7]. In this work, the criterion is also extended to include the comparator noise of the NS-SAR. To do this, we utilize that the comparator noise enters into the linear model (figure 4.1) at the same terminal as the quantization noise $e(n)$. The $e(n)$ terms in equation (4.6) and (4.7) can thus be replaced by a signal

$$e'(n) = e(n) + v_{comp}(n) \tag{4.11}$$

The new largest possible value for $y(n)$ will occur when

$$\begin{aligned} e'(n-i) &= \frac{\Delta}{2} + V_{comp,max} \quad \text{for } ntf(i) > 0 \\ e'(n-i) &= -\frac{\Delta}{2} - V_{comp,max} \quad \text{for } ntf(i) < 0 \end{aligned} \tag{4.12}$$

and the other way around for the smallest possible value. $V_{comp,max}$ is the maximum comparator noise amplitude. By continuing the derivation similarly as above, we arrive at the new criterion

$$U_{\text{p-p},max} + (\Delta + V_{comp,\text{p-p}}) \left(\sum_{i=0}^{\infty} |ntf(i)| - 1 \right) \leq \text{FSR} \tag{4.13}$$

where we see that $v_{comp}(n)$ also contribute to the "accumulation term". This is expected since the comparator noise is shaped likewise as the quantization noise.

If we assume that the comparator noise is Gaussian white noise, there exists no definite value for $V_{comp,\text{p-p}}$. We must therefore assume that V_{comp} falls within $\pm m$ standard deviations of its expectation value, which is zero. We thus get $V_{comp,\text{p-p}} = 2m\sigma$. Then, by utilizing that the noise power V_{comp}^2 equals the variance σ^2, we finally write

$$U_{\text{p-p},max} + \left(\Delta + 2m\sqrt{V_{comp}^2} \right) \left(\sum_{i=0}^{\infty} |ntf(i)| - 1 \right) \leq \text{FSR} \tag{4.14}$$

It is not directly trivial to select an appropriate value for m, but if we set $V_{comp}^2 = \frac{\Delta^2}{12}$ (i.e. equal to the quantization noise), the first parenthesis in equation (4.14) evaluates to 1.57Δ if $m = 1$,

and 2.15Δ if $m = 2$. This demonstrates that the impact of the comparator noise on the modulator stability is comparable to that of the quantization noise, if the noise power is the same for both contributions.

4.3 Choice of poles and zeros for the NS-SAR NTF

After the establishment of NTF restrictions as well as stability criteria in section 4.1 and 4.2, it is now possible to discuss the actual loop filter design in form of its NTF. In section 4.1, it was shown that no constant scaling factors can be used in the NTF. A consequence of this is that its properties are solely determined by its poles and zeros. In the following, we will therefore discuss the NTF design in terms of its pole and zero locations in the z-plane.

4.3.1 Zeros

The simple differentiator noise transfer functions $\mathrm{NTF}(z) = (1 - z^{-1})^L$ (section 2.3) have all their zeros at DC. Although this works well, it is possible to achieve a lower in-band noise power P_E if zeros are spread out in the in-band region of the z-plane unit circle.

When it comes to implementation, this is possible to do by the use of resonators. A resonator is essentially a cascade of two integrators having a negative feedback path around themselves. This feedback path moves the two integrator poles from DC to a non-zero location on the unit circle, as a complex conjugated pair. As poles in $H(z)$ give rise to zeros in the NTF, this results in corresponding NTF zeros.

It is possible to find the optimum zero locations by minimizing the in-band noise gain

$$P_N = \int_{-B_w}^{B_w} |\mathrm{NTF}(f)|^2 \, df \tag{4.15}$$

with respect to the NTF zeros. An approximate analytical solution for this problem is given in [13], and the optimum zero locations for the lowest modulator orders are given in table 4.1. We see that the SNR improvements that arise increase with order, but is still significant already from $L = 2$. These improvements are achieved by only adding some extra feedback paths, which should not alter the modulator power consumption significantly. Optimized zeros are thus chosen when possible for the modulators in this thesis.

The calculation of the values in table 4.1 is done in [13] by means of an approximate NTF frequency

Table 4.1: Optimal zero locations relative to the signal band edge, along with corresponding SNR improvements. Tabulated data are obtained from [13].

Order L	Zero locations normalized to B_w	SNR improvement
1	0	0 dB
2	$\pm\frac{1}{\sqrt{3}}$	3.5 dB
3	$0, \pm\sqrt{\frac{3}{5}}$	8 dB
4	$\pm\sqrt{\frac{3}{7} \pm \sqrt{\left(\frac{3}{7}\right)^2 - \frac{3}{35}}}$	13 dB

response

$$|NTF(\omega)|^2 \approx k \prod_{i=1}^{L} (\omega^2 - \omega_i^2)^2 \tag{4.16}$$

and is stated to be valid when the oversampling ratio is high, and when the magnitude of the NTF denominator (just denoted as k) is approximately constant in the signal band.

The magnitude of the NTF denominator is determined by the poles, and will be reasonably constant in the signal-band if the poles are located at some distance from it. When we think of the NTF as a high pass filter, this is roughly equivalent to having a high cut-off frequency, which is a desirable property. The assumption is therefore probably valid for most well-designed modulators, and has also been considered as that in the modulators simulated in this work.

The factor $(\omega^2 - \omega_i^2)$ is an approximation of $|\cos(\omega) - \cos(\omega_z)|^2$. This approximation is not very good when the OSR is low (as typical in NS-SARs), because the integral in equation (4.16) then includes regions far from ω_i. The purpose of this equation is nevertheless not to give an accurate value for $|\mathrm{NTF}(\omega)|^2$, but rather to estimate its minimum with respect to the zero positions. This is a less strict requirement, and is probably more dependent on how the approximation resembles the shape of the true $|\mathrm{NTF}(\omega)|^2$. This shape resemblance has been looked at qualitatively and found to be reasonable. Some zero sweeps have also been conducted in the simulator framework, and optimum zero locations has been found to be consistent with table 4.1.

4.3.2 Poles

Although the poles of the NTF in principle also should be chosen such that the in-band NTF gain is minimized, there are also stability considerations that must be taken into account. This is because the poles more or less set the value of $\sum_{i=0}^{\infty} |ntf(i)|$, which is a part of the stability criterion in equation (4.13). In practice, the poles are therefore primarily a tool to adjust stability. We will, however, see in the following that extra NTF gain attenuation can also be achieved due to the poles in some cases.

To the best of the author's knowledge, there are no universal, analytical solution for optimal NTF pole locations. To choose pole locations is therefore more like an optimization problem, where both modulator stability and in-band gain must be taken into account. One common way to choose reasonable pole locations is to use a filter synthesis tool, and let the NTF denominator be that of a common filter type, e.g. Butterworth. The filter cutoff frequency can then be tuned until stability and performance are acceptable.[2] This approach could maybe be called design by hand, since trial and error and manual intervention might be needed to find the optimal solution.

In this thesis, modulator simulations are performed in lots of points in the design space, and a more automatic way to find optimum poles for each of these points is therefore needed. The approach chosen is therefore to treat the pole selection task as a non-linear optimization problem, which is solved numerically. The optimization objective is to minimize the in-band gain P_N given by equation (4.15), and the most important constraint is the stability criterion in equation (4.13). This optimization problem is solved in [7], and their solution, which is called CLANS, also happens to be included in the widely used Delta Sigma Toolbox for Matlab, written by Richard Schreier [14]. CLANS is therefore used to compute poles in this work, and more details about the implementation will be given in chapter 5.

One challenge regarding the pole optimization, is that the maximum peak-to-peak input $U_{\text{p-p},max}$ constitutes an extra degree of freedom in the stability criterion in (4.13). This value is regarded as a constant in CLANS, and must therefore be chosen prior to optimization. This choice is non-trivial however, as a decrease in $U_{\text{p-p},max}$ may sometimes result in a better SNR, because better pole

[2] See e.g. [10, chapter 4] for more information on this approach.

choices are then available for the optimizer. Under different conditions, a decrease in $U_{\text{p-p},max}$ may nevertheless result in a worse SNR merely because the signal power was decreased. A search for the best $U_{\text{p-p},max}$ is therefore needed, and this is done by invoking CLANS for a range of acceptable $U_{\text{p-p},max}$.

In the lack of analytical solutions, it is instructive to have a look at some generated NTFs to get some insight into the purpose of the poles under different design conditions. We therefore consider figure 4.2, where frequency responses and pole-zero locations are given for a second order modulator having the numbers of bits B varied. The input signal uses 90 % of the quantizer input range, and the OSR is 8. Moreover, all modulators have a double zero at DC as this makes the plot simpler than if the optimal non-DC zero locations are used. A pure second order differentiator modulator is also plotted for comparison.

When B equals five bit, we see that the performance is worse than that of a pure differentiator (i.e. larger in-band gain). This is because Δ in the stability criterion is reasonably large[3], and the NTF impulse response must therefore be quite constrained to not render the modulator unstable. When B is increased to 8 bit, there is however "room" for a much more aggressive modulator because Δ diminishes drastically. This results in better performance than the differentiator.

One could say that the poles have a stabilizing function for $B = 5$, because the differentiator have to be "softened up" to yield a stable modulator. The poles are then typically moved into the right half of the z-plane. On the contrary, the poles could be said to have a performance boosting function when $B = 8$, because they are used to make a modulator which is better than the differentiator. The poles are then typically moved into the left half of the z-plane.

An interesting observation is that if we consider traditional delta-sigma modulators, the quantizer is typically of the flash type [9]. This means that the maximum value for B is quite constrained, and it is difficult to get performance boosting poles in the modulator. In a SAR however, it is perfectly acceptable to have e.g. around 10 bits, and the poles can therefore be easily used to boost the performance of the modulator. This seems to be a significant advantage of NS-SAR modulators.

4.4 Specific loop filter topologies chosen for test

Three different loop filter topologies have been analyzed and used in simulations in this work. They are of first, second, and third order and consequently use different amounts of power, which is assumed to be proportional to the numbers of integrators in the filter. In the following, each modulator is presented and their main features and advantages are highlighted.

4.4.1 1. order with extra pole (MOD1-EP)

The 1. order with extra pole filter is depicted in figure 4.3, and is the same as that used in [6]. Its loop filter transfer function is given as

$$H(z) = \frac{c_1 a_1 z^{-1} + c_2 a_1 z^{-2}}{1 - z^{-1}} \tag{4.17}$$

and the corresponding NTF as

$$\text{NTF}(z) = \frac{1 - z^{-1}}{1 + c_1 a_1 z^{-1} + c_2 a_1 z^{-2}} \tag{4.18}$$

[3]The comparator noise power were chosen equal to the quantization noise power in this example. This is an acceptable design choice for the comparator. For low values of B, minimum feature sizes might however be the constraining factor.

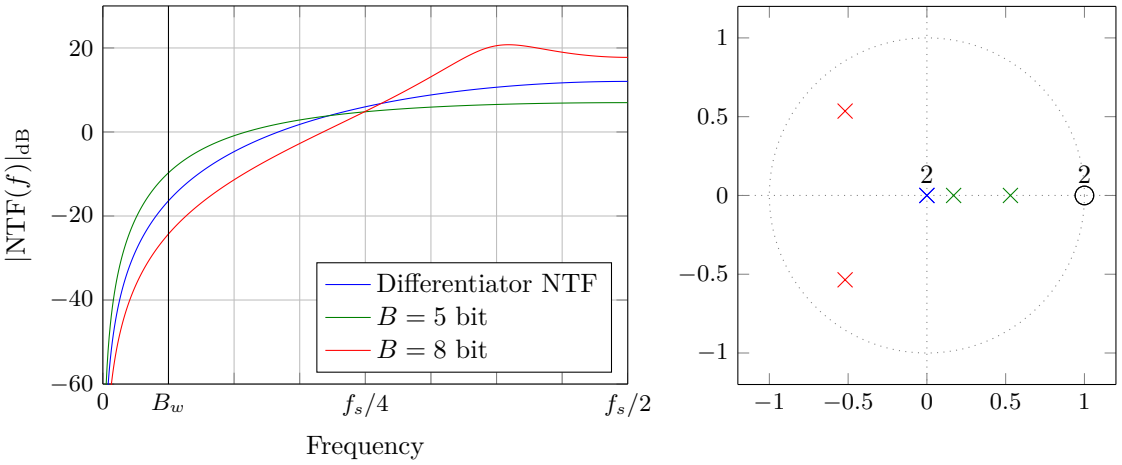


Figure 4.2: NTF frequency response and pole-zero plot for 2. order modulators having different number of bits in the quantizer. Compared to a pure 2. order differentiator NTF. All modulators have a double zero at DC. OSR = 8.

We see that the NTF has a first order differentiator numerator, and it has thus a zero at DC. This is due to the integrator in the right half of the filter topology. The denominator is however of second degree, and a pole pair can be chosen freely by the coefficients c_1 and c_2. This is due to the FIR filter inserted in front of the integrator. As a first order differentiator modulator is normally stable on its own, the second order poles are added only because of the potential advantage of accuracy enhancing poles in NS-SARs.

Due to second order poles, one could argue that this filter is of second order. The FIR filter part is, however, possible to realize passively by storing old input values on capacitors [6]. This means that the power consumption is more comparable to that of a pure first order filter, only made up of a single integrator. The filter is therefore called first order in this thesis.

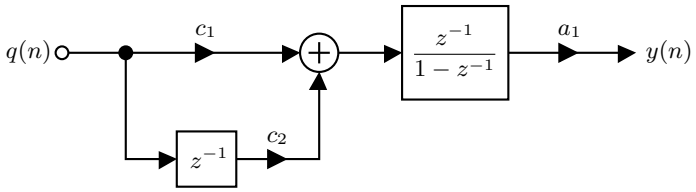


Figure 4.3: 1. order modulator with extra pole.

4.4.2 2. order with resonator (MOD2-RES)

This filter is shown in figure 4.4, and has a loop filter function

$$H(z) = \frac{a_1 z^{-1} + (a_2 - a_1) z^{-2}}{1 + (g_1 - 2) z^{-1} + z^{-2}} \tag{4.19}$$

which yields the NTF

$$\mathrm{NTF}(z) = \frac{1 + (g_1 - 2) z^{-1} + z^{-2}}{1 + (g_1 + a_1 - 2) z^{-1} + (a_2 - a_1 + 1) z^{-2}} \tag{4.20}$$

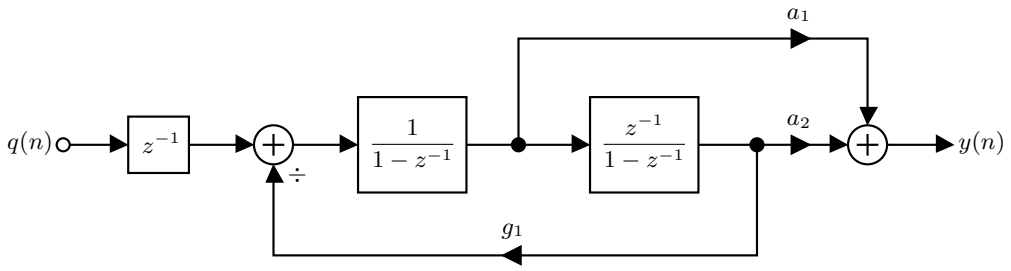


Figure 4.4: 2. order modulator with resonator.

The two integrator blocks and the feedback path constitute a resonator, which gives rise to a non-DC pole pair. This can be seen from the NTF numerator, which is not on differentiator form any more. We can find the magnitude and phase of these complex conjugated zeros $Z = re^{\pm j\theta}$ by considering a general polynomial having these zeros as roots. That is $(1 - Zz^{-1})(1 - Z^*z^{-1}) = 1 - 2r\cos(\omega)z^{-1} + r^2z^{-2}$. By equating this to the NTF numerator, we get

$$\begin{aligned} r &= 1 \\ g_1 &= 2 - 2\cos(\omega) \end{aligned} \tag{4.21}$$

This shows that the zero pair lies on the unit circle, and at an freely chosen angle/frequency set by g_1. This is the wanted behavior.

The NTF also has a second order denominator, where the feed forward coefficients a_1 and a_2 can be adjusted to select either two real poles, or one complex conjugated pole pair. The extra delay block at the filter input does not alter the poles or zeros, but is added solely to satisfy the NTF delay requirement, which was discussed in section 4.1. Without this extra delay, there exists a delay-free path from the input to the output through the a_1 feed forward branch.

The general structure of loop filters like this, which combines resonators and feed-forwarding is given in for instance [10, chapter 4]. It is there called a modified CIFF (Cascaded Integrators Feed Forward) structure.

4.4.3 2. order with resonator and extra pole (MOD2-RES-EP)

Due to the good possibilities to exploit accuracy enhancing poles in NS-SARs, a second order modulator with extra pole has been designed in this work. This topology is shown in figure 4.5 and is based upon the MOD2-RES filter, but with an extra feed forward path added in the output section of the filter. This gives rise to a third order pole polynomial in the NTF. This comes without substantial power increase as the feed-forward can be realized passively by the use of two capacitors, where each of them are connected either to the resonator output or the comparator input every second sample. This modification of the second order modulator might therefore be quite beneficial when it comes to energy efficiency.

The transfer function is

$$H(z) = \frac{a_1z^{-1} + (a_2 - a_1)z^{-2} + a_3z^{-3}}{1 + (g_1 - 2)z^{-1} + z^{-2}} \tag{4.22}$$

and the NTF is thus

$$\mathrm{NTF}(z) = \frac{1 + (g_1 - 2)z^{-1} + z^{-2}}{1 + (g_1 + a_1 - 2)z^{-1} + (a_2 - a_1 + 1)z^{-2} + a_3z^{-3}} \tag{4.23}$$

This NTF is similar to that given for MOD2-RES except for the z^{-3} term, which gives rise to the extra pole.

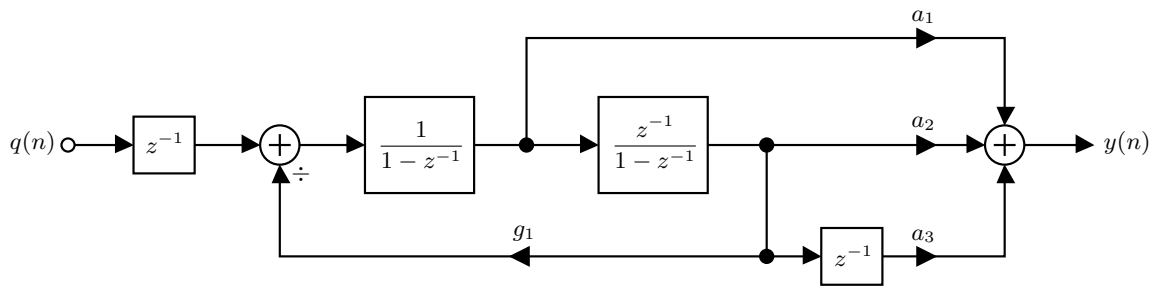


Figure 4.5: 2. order modulator with resonator and extra pole.

4.4.4 3. order with resonator (MOD3-RES)

The 3. order modulator with resonator is also of the modified CIFF structure, and has one more integrator and one more feed-forward path compared to MOD2-RES. This yields three zeros, and three poles. The filter is depicted in figure 4.6 and has transfer function

$$H(z) = \frac{(a_1 + a_2)z^{-1} + ((g_1 - 2)a_1 - a_2 + a_3)z^{-2} + a_1 z^{-3}}{1 + (g_1 - 3)z^{-1} + (3 - g_1)z^{-2} - z^{-3}} \tag{4.24}$$

and NTF

$$\mathrm{NTF}(z) = \frac{1 + (g_1 - 3)z^{-1} + (3 - g_1)z^{-2} - z^{-3}}{1 + (a_1 + a_2 + g_1 - 3)z^{-1} + ((g_1 - 2)a_1 - a_2 + a_3 - g_1 + 3)z^{-2} + (a_1 - 1)z^{-3}} \tag{4.25}$$

By setting $z = 1$, it is found that there is always a zero at DC. The rest of the zeros can be found by equating the numerator to $(1 - 2r\cos(\omega)z^{-1} + r^2 z^{-2})(1 - z^{-1})$, and this shows that the other zero pair also obeys equation 4.21. This makes it possible to choose the optimum zero locations given in table 4.1.

From the NTF denominator, we can see that it is possible to select the three poles freely by selecting appropriate values for the feed-forward coefficients.

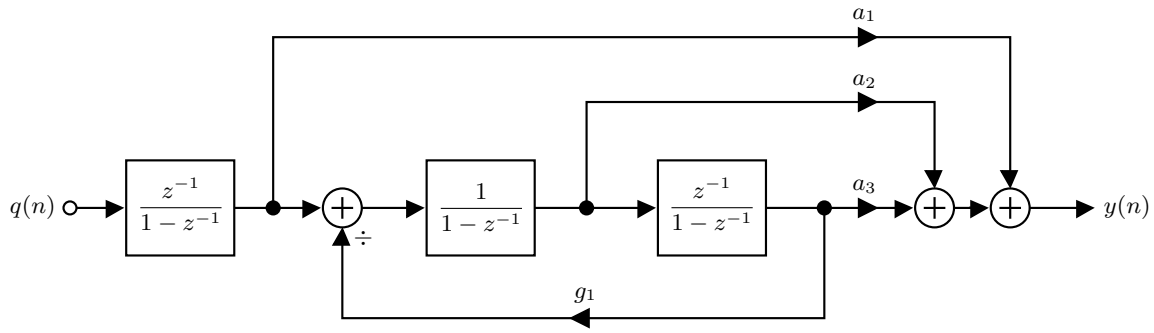


Figure 4.6: 3. order modulator with resonator.

Chapter 5

The behavioral simulation framework

The behavioral simulations in this work have been carried out using a custom-made simulation framework written in Matlab [15]. This framework will be described in this chapter, with an emphasis on the overall structure, modeling techniques and algorithms. All important assumptions made in the various parts of the framework will specifically be mentioned.

The organization of this chapter will to a large extent follow figure 5.1, which shows a conceptual overview over the software. Four main parts can be identified:

NS-SAR time domain simulator In short, this simulator is given a time domain[1] input signal and a loop filter description, and then computes the corresponding ADC output signal. SNDR and ENOB can then be obtained using FFT.

Loop filter representation and synthesis Each loop filter is represented by a Matlab class which inherits from a generic loop filter class. Objects of these classes holds all relevant information about a specific filter, and can synthesize its NTF in different ways. The filter descriptions needed by the simulator are for instance generated and returned from these objects.

Power consumption models Power consumption models are needed to estimate energy efficiency. The models used in this work need some input information about power consumption in a specific design point, and then output power consumption estimates for other relevant parts of the design space according to sensible assumptions.

Test-benches Various tests are formed using the aforementioned parts of the software. This can be simple test-benches to test and experiment with a modulator, or it can be one- or multi-dimensional design variable sweeps which maps large parts of the design space. The data from such sweeps can often be subject to post-processing to extract relevant information.

In the following, each of these main parts will be elaborately described and discussed. The core parts of the Matlab code can be found in appendix A.

[1] In the following discussions, it is implicitly understood that *time domain* means the discrete time domain denoted by the sample variable $n = \frac{t}{T}$, where T is the sample period.

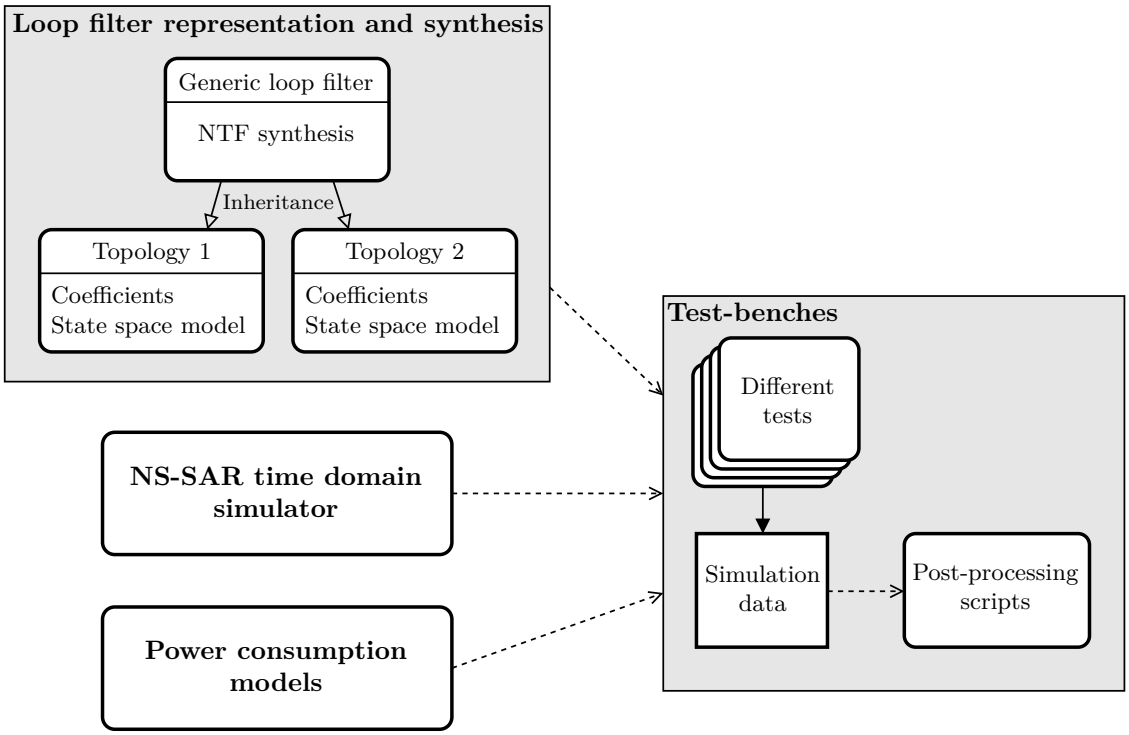


Figure 5.1: Overview over the behavioral simulation framework.

5.1 NS-SAR time domain simulator

As already mentioned, the NS-SAR time domain simulator computes the ADC time domain output when given an input signal and a loop filter description. This is done by the use of a simple behavioral model which for every discrete time step calculates the output and the next modulator state recursively.

The NS-SAR accepts inputs in the interval $[-V_{ref}, V_{ref}]$, where V_{ref} is an selectable input parameter to the simulator. The number of bits is also set freely by a parameter B. Having this defined, equation (2.6) gives the spacing Δ between two adjacent output levels as

$$\Delta = \frac{2V_{ref}}{2^B} \tag{5.1}$$

The mentioned loop filter description is given as a state-space model on the form[2]

$$\boldsymbol{x}(n+1) = \mathbf{A}\boldsymbol{x}(n) + \mathbf{B}q(n) \tag{5.2a}$$
$$y(n) = \mathbf{C}\boldsymbol{x}(n) + \mathbf{D}q(n) \tag{5.2b}$$

where the $K \times 1$ vector $\boldsymbol{x}(n)$ holds all the K filter state variables at time n. $q(n)$ and $y(n)$ are the filter input and output, as in rest of the thesis (see e.g. figure 4.1). The filter states are essentially all the K delay element outputs in the filter.

The first equation is the state equation, which gives the next filter states based on the current states and the current input. This relationship is controlled by the $K \times K$ matrix $\mathbf{A}$, which

[2]In the implementation of the simulation framework, the modulator input $u(n)$ is also an input to the state-space model. Although this should yield realizable modulator, the functionality has so far never been used and is neither consistent with the general NS-SAR linear model used in this work. The $u(n)$-terms are thus omitted in this presentation.

specifies how the next states depend on the current ones, and the $K \times 1$ matrix $\mathbf{B}$, which specifies the input dependencies. The second equation is the output equation, and specifies the output based on the current modulator states using the $1 \times K$ matrix $\mathbf{C}$. The last term which gives the dependency between the current input and the current output are required to be zero due to the NTF constraints discussed in section 4.1, but is written in the equation for formality. The 1×1 matrix $\mathbf{D}$ is thus zero by definition. The state-space matrices are passed to the simulator as a lumped matrix $\left[\begin{smallmatrix}\mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D}\end{smallmatrix}\right]$ which from now on will be called the ABCD-matrix.

The loop filter state-space description is inspired by the aforementioned Delta-Sigma Toolbox [14], which uses state-space models to specify loop filters for the general delta-sigma converter structure, presented in the end of the background theory chapter (figure 2.7). Although this structure/model type can be used to represent NS-SARs, it has been chosen to make a simplified state-space model type which is more consistent with the general NS-SAR linear model presented in this work. The use of state-space models in the Delta-Sigma Toolbox is also covered in [10], and includes a description on how to write the state equations for a loop filter.

Having defined the state-space model, it is now possible to describe how the simulator operates. This is best done by dividing the actions carried out at each time step into three distinct operations:

1. First, the current loop filter output is computed by using equation 5.2b.
2. The binary search is then conducted more or less as described in section 2.1, but modified to accommodate both positive and negative input values. The input to the comparator part of the simulator is $u(n) + y(n)$ (ADC input and modulator output). After this, the current output $D_{out}(n)$ is available.
3. As the binary search is completed, the residue $q(n)$ is also available. The next filter states can thus be calculated using equation (5.2a).

The comparator noise is also modeled in the NS-SAR simulator. This is done by adding a Gaussian distributed stochastic variable to the binary search comparisons. The variance of this variable is set to $\frac{\Delta^2}{12}$, which means that the comparator noise power equals the quantization noise power. When the comparator sizing is not constrained by minumum transistor dimensions (for small bit counts), this is a sensible assumption which for instance is used in [2]. It has nevertheless been chosen to keep the comparator noise power equal to the quantization noise power also at low bit counts, as this results in a convenient constant spacing between the bit count and the ENOB of the pure SAR in the simulations, without altering any results significantly.

The output signal from the time domain simulator is usually fed into a function [16] that performs an FFT, and extracts SNDR and ENOB from this. Plotting of the output spectrum is also supported by this function.

5.2 Loop filter representation and synthesis

The different loop filters are represented using Matlab's object oriented programming (OOP) capabilities. That is, different loop filters are defined as classes that both hold design data, and are able to perform different synthesis tasks and other relevant operations. When the filters are used in the tests, objects of the different classes are instantiated and provided input data about the relevant design point and operating conditions. The objects then dynamically generate and provide loop filter information that matches the given operating conditions. Say that one for instance want to simulate a modulator having a specific OSR and a specific number of bits; This information is then provided to the loop filter object, which among others generates a tailor-made NTF and a corresponding ABCD-matrix that can be provided to the time domain simulator. This

OOP approach enables the loop filters to efficiently adapt to the test environment when different design variables are swept, and this in turn yields clean and simple test-benches.

Although loop filter structures can be very different, a lot of the operations regarding the NTF synthesis are very similar. Also, a large set of data fields can be identified as common and mandatory for all loop filters. This includes for instance the NTF and ABCD-matrix, which have to be defined by all valid loop filter representations. Because of this large degree of resemblance, a generic loop filter class has been made to gather common code and to define an interface/template for the specific loop filter descriptions. When new loop filters are added to the framework, they inherit from the generic class and only need to specify topology specific information. It is thus quite easy to add new filters.

In the rest of this section, we will first have a look at how the NTF is synthesized in the generic modulator class, followed by a short description of how the specific loop filter structures are represented. This will give the reader a good idea of how the loop filter representation system works.

5.2.1 NTF synthesis

The NTF synthesis algorithm computes the NTF poles and zeros, and subsequently calculates the numerator and denominator coefficients for the NTF. How the zeros are determined depends on a setting called `optimizedZeros`, which tells if the optimal zero locations introduced in section 4.3.1 should be used. If this setting is not set, all the zeros are placed at DC ($z = 1$).

The pole computation is controlled by a setting `poleMode`, which either equals `none`, `butterworth` or `clans`. The first mentioned places all poles at $z = 0$, whereas `butterworth` sets the poles to those of a Butterworth high-pass filter with selectable cut-off frequency. The mode used when calculating all the results presented in this thesis is however `clans`, which invokes the CLANS algorithm described in section 4.3.2.

The Delta Sigma Toolbox implementation of CLANS is used, and the core of this implementation exploits the Matlab Optimization Toolbox to compute the poles. Some minor changes were still done to the code to accommodate different numbers of poles and zeros, and to correct some reliability issues.

The most important input argument to CLANS is the wanted value of $\sum |ntf(n)| - 1$, which is obtained from equation (4.14) as

$$\sum_{i=0}^{\infty} |ntf(i)| - 1 = \frac{\mathrm{FSR} - U_{\mathrm{p\text{-}p},max}}{\Delta + 2m\sqrt{V_{comp}^2}} = \frac{2V_{ref} - 2U_{\mathrm{a},max}}{\frac{2V_{ref}}{2^B} + 2m\frac{2V_{ref}}{2^B}\frac{1}{\sqrt{12}}} = \frac{V_{ref} - U_{\mathrm{a},max}}{\frac{V_{ref}}{2^B}\left(1 + \frac{m}{\sqrt{3}}\right)} \tag{5.3}$$

where $U_{\mathrm{a},max}$ is the wanted maximum input amplitude and m is the number of standard deviations taken into account for the Gaussian comparator noise. Further, the comparator noise power was set equal to the quantization noise power, and equation (5.1) was used for Δ in the above development. V_{ref} is the same V_{ref} as for the time domain simulator. $m = 1$ has been used for all simulations presented in this thesis, and never lead to unstable NTFs.

$U_{\mathrm{a},max}$ is in principle an unwanted degree of freedom in the above equation, since it affects the modulator ENOB in an nontrivial way. This was discussed in section 4.3.2. Because of this, an extra search algorithm which finds the optimal $U_{\mathrm{a},max}$ has been written on top of the CLANS implementation. This algorithm is given an acceptable range for $U_{\mathrm{a},max}$ as input, and then conducts a linear sweep in this range, where CLANS is invoked in every sweep point. After this, the $U_{\mathrm{a},max}$ that gave the best estimated performance is selected as the optimal one. The performance is

evaluated by computing

$$\gamma_i = (U_{\mathrm{a},max})_{i,\mathrm{dB}} - (P_n)_{i,\mathrm{dB}} \tag{5.4}$$

where i is the current point of the input amplitude sweep, and $(P_n)_i$ is the in-band gain of the NTF computed by CLANS for that sweep point. The resemblance between the optimal $U_{\mathrm{a},max}$-values found by this estimation method, and those observed in exhaustive time domain sweeps has been found satisfying.

All-in-all, these synthesis techniques yields tailor-made NTFs optimized for the current design point. For the NTF to be optimal, the test-bench must pull the computed optimal $U_{\mathrm{a},max}$ value from the loop filter object, and use that as the amplitude for the time domain simulator input signal. Since the CLANS algorithm is invoked in every point of the $U_{\mathrm{a},max}$ sweep, the NTF synthesis can take a bit time if the sweep is selected to be too fine-grained. The number of points in this sweep can therefore be tuned from the test-bench until acceptable performance is achieved.

5.2.2 Representation of specific filter structures

The main task of the specific loop filter classes is to define and compute topology specific coefficients, and to define the ABCD-matrix. The first mentioned is obtained by coefficient comparison between the synthesized NTF, and the topology specific NTFs presented in section 4.4. So far, the symbolic comparisons has been done by hand, and obtained formulas for the coefficients has been put into the loop filter classes. Numbers are then computed every time the NTF changes.

The ABCD-matrices are also found symbolically by hand, and are expressed terms of the topology coefficients. New numeric matrices are thus calculated every time the numeric values of the coefficients change. The loop filter function $H(z)$ as well as the topology specific NTF are also computed for every new set of coefficients, and the resemblance between the synthesized NTF and the one computed back from the coefficients can be examined to spot implementation errors. Also, it is possible to include topology specific parasitics in the coefficient-defined NTF.

The topology specific loop filter classes also sets up standard settings and restrictions for the NTF synthesis performed by the generic class. If the filter does for instance not contain a resonator, optimized zeros are deselected, and can neither be turned on manually from the test-bench.

5.3 Estimation of power consumption

Reliable estimates of the ADC power consumption are crucial when investigating energy efficiency, and power consumption models for both the loop filter and for the SAR ADC (without loop filter) have thus been designed. The models do not aim to calculate absolute power consumption stand-alone, but rely on an assumption of the power consumption in one specific operating point (in terms of frequency and accuracy), and scales this absolute value across the design space. The accuracy of the absolute power estimates are thus a function of the input assumption quality, whereas the relative accuracy when the power is scaled does only depend on the model quality.

Both models can also be called black-box models, as they only consider how the power scales globally for the whole SAR or loop filter when frequency or accuracy are changed; The mechanisms that governs the power consumption in the specific circuit blocks are not a concern of the models. The models are thus simple, but model the most important global scaling mechanisms in a satisfiable way.

5.3.1 Loop filter

The loop filter model is given as

$$P_{\mathrm{F}} = P_{\mathrm{F},0}\, L\, \frac{f_s}{f_{s,0}} \tag{5.5}$$

where $P_{\mathrm{F},0}$ is the assumed power of a first order modulator operating at frequency $f_{s,0}$.

We see from the formula that the power is assumed to scale linearly with the filter order L. This is a sensible assumption as the integrators dominate the loop filter power consumption, and we need one integrator per order/NTF zero. It is assumed that the extra poles in some of the filters presented in section 4.4 are realized passively without significant power consumption increase.

It is also assumed that the power consumption scales linearly with operating frequency. This is not true when the ADC operates at high frequencies, where the transistor transit frequency f_t needs to to be increased as the speed is scaled, with the consequence of a square relationship between frequency and power [1]. We thus have a linear region, and a squared region for the power-frequency relationship, and this can also be seen from the empirical data in [1], where a regime change can be spotted at 100 MHz. As this work mainly focuses on the relationship between power and accuracy, and not on the power-speed relationship, it is assumed that we stay in the linear region all the time. The validity of this assumption is strengthened by the fact that the OSR in NS-SARs are usually quite low (will come clear in chapter 6), and this means that B_w and f_s should be of the same order of magnitude in most situations.

5.3.2 SAR

As stated in chapter 1, the energy per conversion quadruples per bit for ADCs limited by thermal noise. For lower-accuracy ADCs where thermal noise is normally not the main constraint, the energy correspondingly doubles per bit. This means that the SAR power consumption is proportional to $2^{\beta\,\mathrm{ENOB}}$, where β equals 1 for the lowest ENOB-values or bit counts, and 2 for the highest ones.

In the model, it has been chosen to model β as shown in figure 5.2a. Here, a linear transition region connects the value of 1 for low bit counts and 2 for high bit counts. The transition region is defined by a parameter B_{th}, which defines the assumed position of the power regime change, and a width parameter δ which defines the smoothness of the transition.

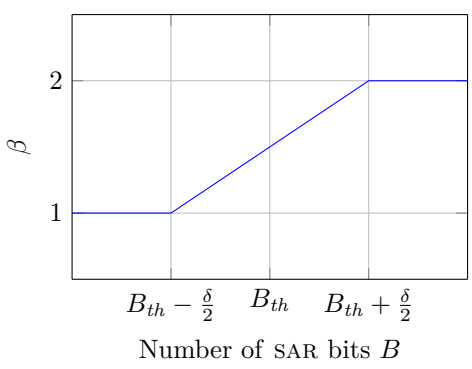
By using β, the power consumption of a SAR having B bits and operating at the frequency f_s can then be modeled as

$$P_{\mathrm{SAR}} = P_{\mathrm{SAR},0}\, \frac{f_s}{f_{s,0}} 2^{\left(\int_{B_0}^{B} \beta\, dB\right)} \tag{5.6}$$

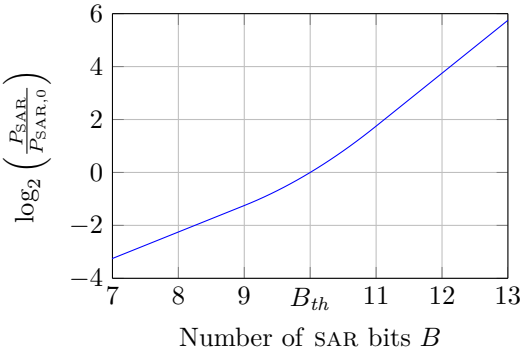
where $P_{\mathrm{SAR},0}$ is the assumed power consumption when the SAR has B_0 bits and operates at $f_{s,0}$. The power vs. frequency scaling mechanisms and assumptions are identical as for the loop filter.

The output of this model normalized to $P_{\mathrm{SAR},0}$ is plotted logarithmically for some bit values in figure 5.2b. One can identify three distinct regimes along the x-axis: The doubled power per bit regime to the left, the transition region in the middle, and the quadrupled power per bit to the right. The value of B_0 is equal to B_{th} in the plot.

It has been chosen to use the number of bits B as input to the model rather than ENOB, because it is the number of bits B which is set explicitly as a design variable during the tests. The use of SAR ENOB instead would have lead to separate simulations of the SAR alone, to compute its ENOB without the loop filter. The use of B as a substitution for SAR ENOB is valid as long as one assumes that every bit increase in the SAR results in the same SAR ENOB increase. This should be



(a) The β parameter.



(b) The SAR power estimate relative to the reference power when $B_{th} = 10$ and $\delta = 2$. The reference frequency $f_{s,0}$ is used, and $B_0 = B_{th}$.

Figure 5.2: Plots related to the SAR power model

close to the truth, because it is typically wanted to re-size/re-scale the SAR circuits after the bit increase to get a comparable ENOB increase.

5.4 Design space mapping

One can utilize the already mentioned parts of the simulation framework for a lot of different tests, and the task/experiment at hand determines how the test-bench is designed. An important test-bench called the ENOB-mapper is however used to generate all the simulation raw data used to obtain the results in chapter 6, and thus needs some extra explanation.

The ENOB-mapper performs a two-dimensional sweep of OSR and B for all the modulators of choice. For every OSR-B combination, NTF synthesis with CLANS and the optimal amplitude algorithm is first invoked in the filter object. This yields an optimal loop filter which is used to conduct a NS-SAR time domain simulation using the optimal input amplitude. FFT is finally taken of the time domain output signal to compute ENOB.

As both NTF-synthesis, time domain simulation and FFT is performed in each point in the 2D-sweep, and also for every modulator of interest, the ENOB-mapping usually take hours. Matlab Parallel Processing Toolbox is however utilized to speed up the simulation.

As shown in figure 5.3, the data generated by the sweep can be visualized as a three-dimensional ENOB-map for each modulator. Although one not gets very much information by visually inspecting the 3D-plots, the data sets together with the power consumption models can be used as basis for a wide range of analyses. In practice, these analyses are written as post-processing scripts that import an already computed ENOB-map. This means that no further time domain simulations are needed, and all the post-processing scripts thus complete in seconds when run.

5.4.1 Optimal modulator algorithm (OPT-MOD)

Most of the post-processing scripts are used to generate the plots shown in chapter 6, and their function will be explained there if needed. An important energy efficiency search algorithm named OPT-MOD is however described here, as it is quite vital in all the energy-efficiency analyses carried out.

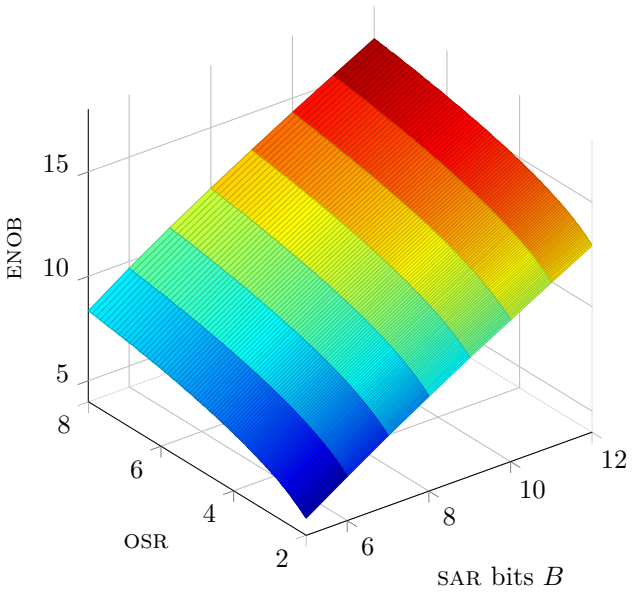


Figure 5.3: An example of an ENOB map for the MOD1-EP NS-SAR

The task of the OPT-MOD is to take in an ADC design specification in form of B_w and ENOB, and find the OSR-B point where this specification can be satisfied with the best energy efficiency. Which loop filter to use is also given as an input to the algorithm.

The metric chosen to quantitatively represent the energy efficiency is the commonly used ADC figure of merit (FOM), given as

$$\text{FOM} = \frac{P}{2^{\text{ENOB}} \cdot 2B_w}\ ,\ \left[{}^{\text{J}}/_{\text{conv.-step}}\right] \tag{5.7}$$

where low FOM means good energy efficiency. We observe that the FOM value stays constant if the power doubles when eiher ENOB is increased by one, or if the bandwidth is doubled. This metric thus "allows" power and accuracy, or power and bandwidth to trade equally.

Having defined FOM, the OPT-MOD algorithm can be described in a series of steps:

1. The accuracy specification is given to the algorithm in form of a ENOB range $[\text{E}_{min}, \text{E}_{max}]$. Having this, the ENOB-map (which is also an input) is searched for points that matches the valid ENOB range. The matching points are candidates for the optimal design, and are stored in a table.

2. To use the power consumption models, the sample rate f_s is needed. This is found by multiplying the OSR-values corresponding to the points found in step 1 with the wanted bandwidth.

   Note that the models used in the framework yields the same results independently of the absolute sample rate. This means that the sample rates/bandwidths used when the ENOB-map was generated has no effect on the outcome of this algorithm. That is, it is nothing in the models that makes fast ADCs perform worse than slow ones, or vice verca.

3. The FOM is then calculated for all the candidate designs. The power is set to $P_{\text{F}} + P_{\text{SAR}}$, and the presented power consumption models are used.

4. Now, the design with the lowest FOM is picked as the optimal one, and returned from the algorithm.

Chapter 6

Behavioral simulation results and discussions

In this chapter, results obtained from the behavioral simulation framework will be presented. The first part of the chapter aims to analyze the performance of the different loop filters presented in section 4.4, compared to pure differentiator loop filters. ENOB of the modulators is used as the main performance metric, and energy efficiency/FOM is not considered.

The next part of the chapter focuses on energy efficiency. The OPT-MOD algorithm presented in section 5.4.1 is used to pick optimum design points for different specified ENOB values, and the resulting energy efficiency curves for the different modulators and a pure SAR are analyzed. An analytical formula that aims to predict which OSR-B values the OPT-MOD algorithm picks in different situations is also developed.

It should be noted that some of the ENOB values in the plots presented in this chapter are very high, as it is nothing that stops the models from outputting high ENOB when OSR and SAR bits are increased sufficiently. In reality, various circuit phenomenons that is not modeled will however counteract this high performance, and the presented results can therefore not be interpreted as valid after the ENOB has crossed some threshold. It can nevertheless in some situations be interesting to study theoretical performance and developments also in this high-ENOB region, and it is thus often shown in the plots.

It is expected that the SAR DAC linearity is the main constraining factor which is not modeled. This DAC is equivalent to the feedback DAC in a normal delta-sigma modulator, and thus have to be as linear as the modulator output [10, chapter 6]. In [17], a 12-bit ADC using a highly energy efficient DAC having maximum INL (integral non-linearity) of 0.5 LSB is presented. This means that the DAC in a 12 ENOB NS-SAR should probably be both realizable and energy efficient. ENOB equal to 12 can thus be regarded as a rule-of-thumb upper threshold for validity in the results to be presented.

6.1 Loop filter performance

Before discussing energy efficiency, it is instructive to analyze how the loop filters presented in this thesis perform in different situations, both relative to each other, and relative to pure differentiator loop filters. This is not directly trivial, as there are no analytical formulas to consider when the poles are picked numerically. Also, both OSR and the number of SAR bits impact the loop filter

performance, which means that the performance of the loop filters has to be analyzed in a two-dimensional design space.

The approach taken to this analysis is to consider 2D-projections of the numerical ENOB data generated by the ENOB-mapper. That is, we analyze the performance with respect to the number of SAR bits, and with respect to OSR isolated, and then try to draw an overall conclusion.

In the plots that follows, the performance of 1.-3. order pure differentiator modulators having NTFs of $(1 - z^{-1})^L$ are plotted for reference. The ENOB of these modulators are predicted with help from the in-band noise power equation (2.18), rather than by time domain simulations.[1] This facilitates the use of these modulators as a theoretical reference also in design points where stability problems would have corrupted real simulations. Also, a full range input signal is always selected in the ENOB computations for simplicity; This is neither possible in simulations.

As mentioned in section 5.1, comparator noise with power equal to the quantization noise power is modeled in the time domain simulator. Due to this fact, the differentiator ENOB values are adjusted to also reflect a noisy comparator. Since a pure, ideal SAR loses approximately 0.5 ENOB when comparator noise equal to the quantization noise is introduced, a compensation for the differentiator is done by using a modified quantization step value $\Delta' = \frac{\text{FSR}}{2^{B-0.5}}$ in the formulas.

6.1.1 ENOB versus SAR bits

Figure 6.1 shows the ENOB versus SAR bits for all modulators at OSR 2 and 8. The performance of differentiator modulators are also plotted for reference, where the lowermost curve is for first order, and the uppermost for third order. Note that at in figure 6.1a, the points at $B = 6$ for MOD2-RES-EP and MOD3-RES deviates a lot from the pattern. This is believed to be due to bad performance of the optimizer in the CLANS algorithm, and the points should probably not be regarded as valid.

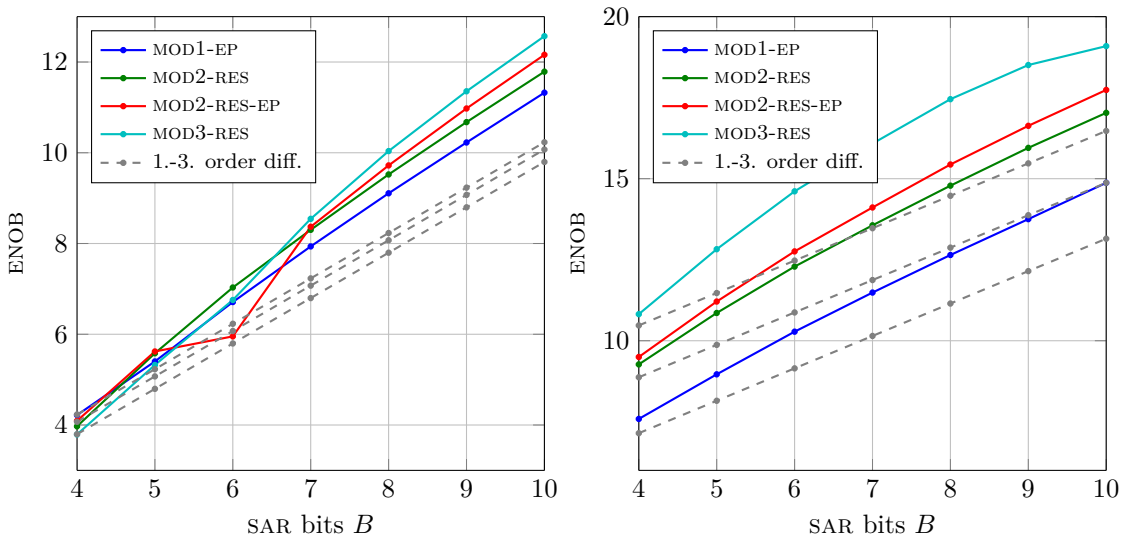
The first thing to note is that the differentiator curves have a slope of exactly 1 ENOB per SAR bit, while all the other modulators have greater slopes. The reason for this is that the poles is altered by CLANS to yield a more aggressive NTF every time the bit count is increased. It is thus possible to gain more than one ENOB per SAR bit.

In figure 6.1a it is also possible to see that the ENOB curves for all the simulated modulators crosses those of the differentiators in the 4-5 bit region. This is thus where the poles change functionality from stabilizing to accuracy enhancing, as discussed in 4.3.2. In figure 6.1a, when the OSR equals 8, the corresponding crossings seems to occur at around three bits, which is off the plot. This is probably due to the increased freedom to select smart pole locations when the signal-band is smaller.

MOD1-EP and MOD2-RES both have two poles, and we can see from the plots that this results in identical slopes. MOD2-RES-EP and MOD3-RES get slightly steeper slopes due to an extra pole, although the difference is marginal.

Note that a slope of less than one ENOB per SAR bit that can be seen at the end of the MOD3-RES-curve in figure 6.1b. There is probably no good reasons for this, so it is reasonable to assume that CLANS picked sub-optimal pole positions for this point.

[1] An analytical expression for ENOB in pure differentiator modulator was in fact given in equation (2.25). This is however an approximation only valid for reasonably large OSR values, and numerical calculation of the in-band noise with equation (2.18), thus yields more valid results in all situations.



(a) OSR = 2 **(b)** OSR = 8

Figure 6.1: ENOB vs. bits in the SAR for all modulators.

6.1.2 ENOB versus OSR

Figure 6.2 shows how the ENOB develops with the OSR for all modulators. Differentiator curves are again plotted, where the lowermost one in the right end of the plots is for first order, and so forth. Note that we again have some CLANS problems in figure 6.2a for low OSR. It is thus assumed that the shape of the true curves should be more like those of figure 6.2b.

The most important thing to note from these plots is that the shape of the curves is almost identical in both figure 6.2a and 6.2b. This means that although optimal pole positions greatly enhances absolute ENOB at high bit counts, the ENOB per OSR performance seems to be quite similar for different values of B.

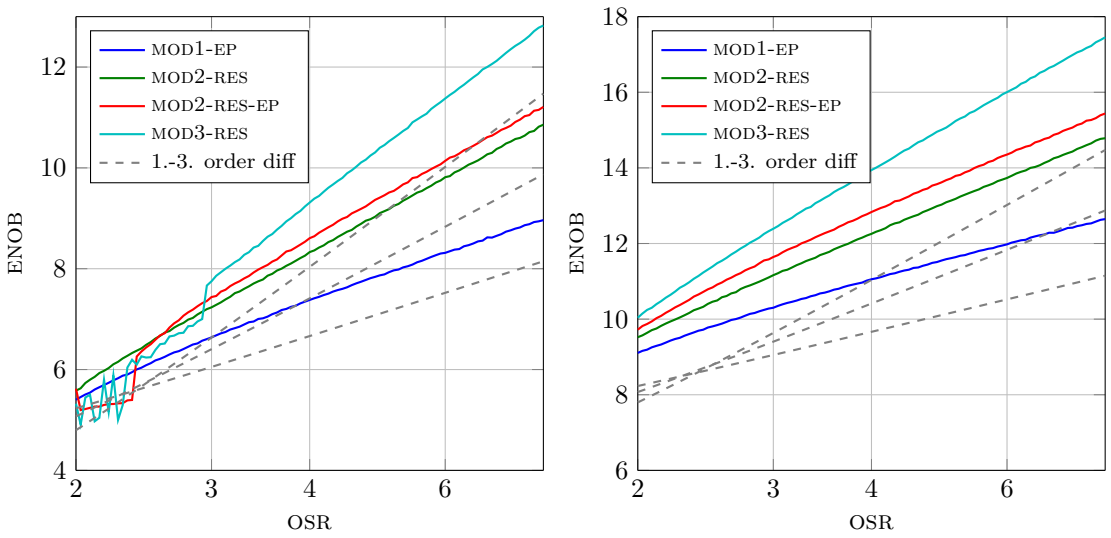
To further study the relationships, it is more instructive to look at the derivatives of the curves in figure 6.2, or the ENOB per OSR octave plot. This is given for 8 SAR bits in figure 6.3. If one not consider the CLANS problems, the plot for 5 SAR bits is quite similar due to the discussed resemblance between the curves in figure 6.2.

The derivatives is approximated numerically from the simulation data, which does not change by the same rate between different data points (can be seen as small wobbling in figure 6.2). The numerical differentiation thus yielded a quite noisy signal, and a moving average filter of approximately $^1/_3$ octave length has therefore been used on the data to obtain figure 6.3.

The first thing to note is that all the simulated curves approach the ones of the pure differentiators as OSR is increased. This means that when OSR $\to \infty$, all the simulated modulators perform equal to their differentiator counterparts.

The curves do however deviate from the differentiators when OSR is decreased. This means that the effect of optimally chosen poles boost the ENOB per octave relationship at low OSR. Also, a performance difference between between MOD2-RES and MOD2-RES-EP can readily be seen at low OSR due to the extra pole in MOD2-RES-EP.

The fact that this plot looks quite similarly also for lower SAR bit counts, can interpreted as that the relative "help" from the poles as OSR is increased is similar for different bit counts. The



(a) $B = 5$ **(b)** $B = 8$

Figure 6.2: ENOB vs. OSR for all modulators.

absolute performance advantage does however increase with number of SAR bits, as shown in figure 6.1.

6.1.3 Discussions

As the modulator performance varies in a non-regular manner in a two-dimensional design space, it is not straight-forward to reveal all the underlying mechanisms.

The most important conclusion to draw, is probably that optimally chosen poles are of the highest importance when OSR is low. The outcome of this is better ENOB per OSR curves both for high and low SAR bit counts, as well as relatively large absolute performance gains at high bit counts.

A significant advantage of having three poles in the NTF rather than two can however not easily be seen from the presented data. The extra pole added is however a real one because of the odd pole count. An interesting question is thus if four poles can yield a better performance gain, since this will give the possibility for two complex-conjugated pole pairs.

The effect of optimal zero positions is not evident from any of the plots. This means that the zero placement only yields the same absolute performance gain under all conditions, i.e. offset in the ENOB curves. This is also in accordance to table 4.1, where constant improvements were in fact presented. Optimal zero locations is thus most important at low OSR, where these offsets can be a relatively large part of the total performance gain compared to a pure SAR.

6.2 Energy efficiency

One of the main tasks of this thesis, is to find the optimum design points in terms of energy efficiency for NS-SAR ADCs. This has been done by means of the OPT-MOD algorithm, which picks the best possible OSR-B point for the modulator of choice when given a design specification. Specified ENOB has thus been swept for all modulators, and the resulting optimum design points

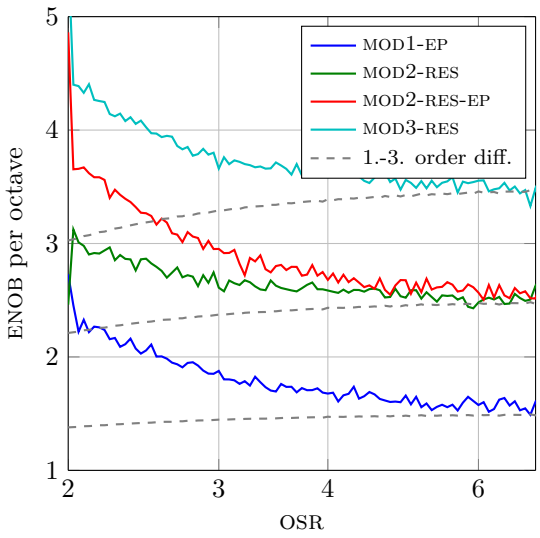


Figure 6.3: ENOB per octave for all modulators. $B = 8$.

and corresponding FOM-values returned from OPT-MOD has been plotted and analyzed. As also stated in section 5.4.1, the FOM is given as

$$\text{FOM} = \frac{P}{2^{\text{ENOB}} \cdot 2B_w} \ , \ \left[{}^{\text{J}}/_{\text{conv.-step}}\right] \tag{6.1}$$

Note that OPT-MOD take the wanted bandwidth as an input argument in addition to the ENOB. When evaluating energy efficiency, it is nevertheless not necessary to sweep this variable as long as it is assumed that we stay in the region where power scales linearly with frequency. This can be proven by showing that FOM is not dependent on absolute values of f_s and B_w in this case:

$$\text{FOM} \propto \frac{f_s}{B_w} = \frac{B_w \cdot \text{OSR}}{B_w} = \text{OSR} \tag{6.2}$$

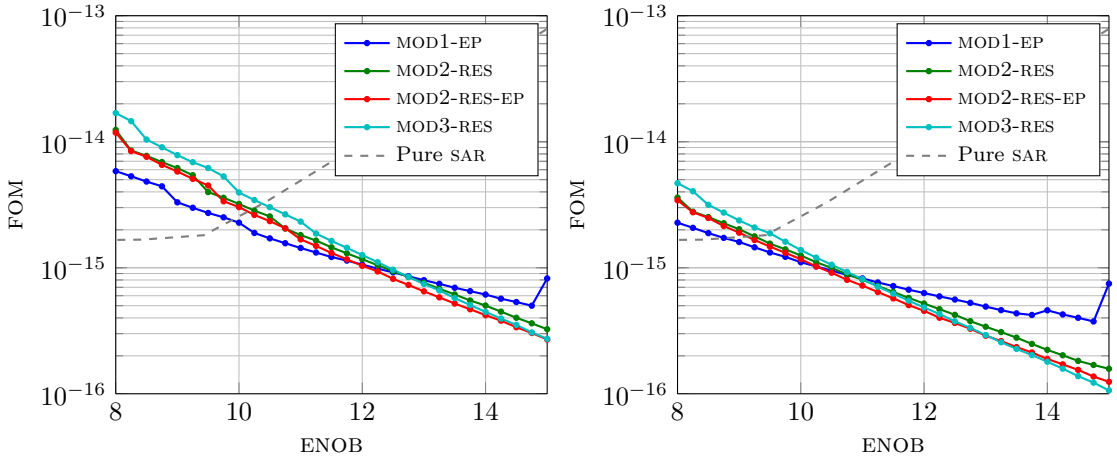
B_w is thus not a consideration as long as the resulting $f_s = 2B_w\text{OSR}$ is a value which is assumed to be in the linear region, and which is otherwise realizable in the practical system. This is assumed fulfilled in the following discussions.

The power consumption models given in section 5.3 need a handful of input parameters, and the ones used to generate the results to be discussed are given in table 6.1. The SAR and loop filter power assumptions originate from ongoing work at NTNU regarding energy efficient SARs and integrators in 28 nm FDSOI technology [16, 18]. The value of $B_{th} = 10$ bit seems like a reasonable assumption both when considering [1], and the introduction chapter of [4]. Also note that OPT-MOD need a specified ENOB range as specification, and not just a distinct value. This is solved by using a step size of 0.25 ENOB during the sweeps, and setting the width of the OPT-MOD ENOB range to the same value.

A plot of FOM versus specified ENOB is given in figure 6.4a for all simulated modulators, and also a pure SAR. We see that all modulators have a higher FOM (worse performance) than the pure SAR at low ENOB values, and that this relationship changes for higher ENOB. Particularly interesting is the fact that when the SAR FOM starts to increase at the regime change, the modulator FOMs are unaffected. This is because the modulators can increase ENOB by adjusting OSR, and thus not need to use energy-inefficient SARs having $B > B_{th}$. This result clearly shows the capability of the NS-SARs to be energy efficient in regions where normal SARs have entered the thermal regime.

Table 6.1: Input parameters to the power consumption models.

Parameter	Value	Description
$f_{s,0}$	32 MHz	Frequency of SAR and filter power assumptions.
B_0	10 bit	Bit count of SAR power assumption.
$P_{\mathrm{F},0}$	20 µW	Assumed power of a first order loop filter @ $f_{s,0}$.
$P_{\mathrm{SAR},0}$	40 µW	Assumed SAR power @ B_0, $f_{s,0}$.
B_{th}	10 bit	Transition between SAR energy regimes.
δ	1 bit	Width of SAR regime change transition.



(a) $P_{\mathrm{SAR},0} = 40\,\mu W$ and $P_{\mathrm{F},0} = 20\,\mu W$.

(b) $P_{\mathrm{SAR},0} = 40\,\mu W$ and $P_{\mathrm{F},0} = 5\,\mu W$.

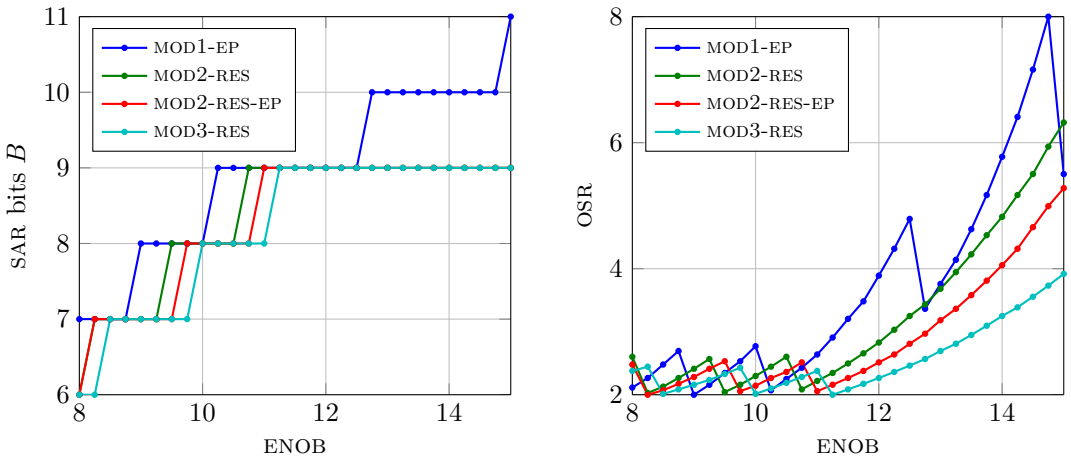
Figure 6.4: FOM versus specified ENOB for optimal modulators.

That the modulator FOMs actually improve when ENOB is increased might actually seem a bit illogical. The validity of this phenomenon is however understood by realizing that when increasing OSR, the 2^{ENOB} term in the denominator of the FOM formula increases faster than the power consumption in the numerator when the modulators yields more than 1 ENOB per octave.

When comparing the modulators, we see that the FOM improvement slope gets better with modulator order, and this can be linked to the fact that ENOB per octave performance increase with order. At low ENOB, low order nevertheless seems to be best. This phenomenon can be interpreted as it is a bigger penalty to put a energy-consuming high-order loop filter into a SAR, than e.g. a first order filter which uses less energy. The high-order modulator then needs a quite long ENOB-increase span to counteract this initial penalty. If we then consider figure 6.4b, where the loop filter energy consumption has been decreased by a fourfold, the "insertion penalties" of the loop filters are considerably smaller. The crossing point of the FOM slopes of the modulators is thus moved to the left.

To learn more about the locations of the FOM optimums in the design space, it is instructive to observe which OSR-B values that are picked by OPT-MOD during the ENOB sweep. This is shown in figure 6.5 for the situation corresponding to the FOM in figure 6.4a. It is quite evident that the development here can be divided into two distinct phases: First, the number of SAR bits is increased quite actively, while the OSR values is attracted toward the minimum value included in the ENOB-map. Then, the SAR-bits settle at locations that appears to be optimums, and the OSR is increased to reach higher ENOB.[2]

[2]Eventually, the MOD1-EP modulator reaches the upper OSR limit included in the ENOB-map, and is thus forced to use a SAR in the thermal regime to reach the specified ENOB.



(a) Number of SAR bits.

(b) OSR

Figure 6.5: OSR and SAR bits values picked by the OPT-MOD algorithm for $P_{\mathrm{SAR},0} = 40\,\mu W$ and $P_{\mathrm{F},0} = 20\,\mu W$.

If we consider the ADC ENOB specification of 11.33 ENOB given in the introduction, we see from the preceding plots that this converter can be realized energy-efficiently by using the MOD1-EP converter, 9 SAR bits, and an OSR of approximately 3.

To get more insight into the mechanisms governing the apparent SAR bit optimums, a mathematical approach has been taken. This will now be described in the following.

6.2.1 Mathematical approach to the optimum design space points

Assume that we have a loop filter which uses a power of P_F, and a pure SAR that operates in the doubled energy per bit regime. We are allowed to add or remove bits from this SAR by redesigning, and we assume that the consequence is that the power P_S doubles per bit, and that the ENOB E_S is increased by one per bit. Let then the special bit count case where $P_S = P_F$ be called the reference case, where we call the SAR power $P_{S,ref}$ and the corresponding ENOB $E_{S,ref}$. That is, $P_{S,ref} = P_F$ is always true. The FOM of the pure SAR in this reference case is

$$\mathrm{FOM}_{ref} = \frac{P_{S,ref}}{2^{E_{S,ref}} \cdot 2B_w} \tag{6.3}$$

Assume that the bandwidth B_w never changes in the following.

We then insert the loop filter into the SAR, to make a NS-SAR where the OSR is adjusted freely. Assume that it is now also allowed to adjust the number of bits in the SAR. The total power consumption relative to $P_{S,ref}$ will then be scaled by a factor

$$P_{scale} = (1 + 2^{E_S - E_{S,ref}})\mathrm{OSR} \tag{6.4}$$

where the first term in the parenthesis is the loop filter, which never changes its power. The second term, which corresponds to the SAR, is however different from one when the bits are adjusted away from the reference case. Also, the overall power consumption scales with OSR if we assume that both the loop filter and SAR power consumption scales linearly with frequency.

We also get a change in accuracy. The 2^{ENOB} factor in the FOM thus exhibit a change which can be given relative to $E_{S,ref}$ as

$$E_{scale} = 2^{E_S + \alpha \log_2(\mathrm{OSR}) - E_{S,ref}} = \mathrm{OSR}^{\alpha}\, 2^{E_S - E_{S,ref}} \tag{6.5}$$

where the E_S-term is the extra accuracy we get due to a new bit count in the pure SAR. α is the ENOB per OSR octave parameter which is assumed to be a constant in this derivation.

The new FOM can now be given as

$$\text{FOM} = \text{FOM}_{ref}\,\frac{P_{scale}}{E_{scale}} = \text{FOM}_{ref}\,\frac{(1+2^{E_S-E_{S,ref}})\,\text{OSR}}{\text{OSR}^{\alpha}\,2^{E_S-E_{S,ref}}} = \text{FOM}_{ref}\,(1+2^{E_{S,ref}-E_S})\,\text{OSR}^{1-\alpha} \quad (6.6)$$

We now want to eliminate OSR from this equation. This can be done by realizing that the ENOB E of the whole NS-SAR is given as

$$E = E_S + \alpha\log_2(\text{OSR}) \;\Rightarrow\; \text{OSR} = 2^{(E-E_S)/\alpha} \quad (6.7)$$

By substituting this into equation (6.6) and reorganizing, we finally obtain

$$\text{FOM} = \text{FOM}_{ref}\,2^{(\frac{1}{\alpha}-1)E}\,(2^{E_S}+2^{E_{S,ref}})\,2^{-\frac{1}{\alpha}E_S} \quad (6.8)$$

We now have an analytical expression for FOM in terms of various ENOB values, a reference FOM and ENOB per octave, which is assumed to be a constant parameter. Now consider that we have a given design specification, loop filter and power consumption estimate. This implies that both E, $E_{S,ref}$ and α are known and constant, and the only varying variable left is E_S. This means that we can solve for the minimal FOM in terms of an optimal E_S value by equating $\frac{d\text{FOM}}{dE_S}$ to zero and solve for $E_{S,opt}$. This yields

$$E_{S,opt} = E_{S,ref} + \log_2(\frac{1}{\alpha-1}) \quad (6.9)$$

We have now solved analytically for the optimal ENOB in the SAR under the simplifications that has been given. The corresponding OSR needed to achieve the specified NS-SAR ENOB can be found from equation (6.7).

We can also use this result to make an equation which gives the optimum power in the SAR relative to $P_{S,ref} = P_F$. To do this, we just need to recall that it is assumed that the SAR power is doubled per bit. We can then write

$$P_{S,opt} = P_F\,2^{E_{S,opt}-E_{S,ref}} = P_F\,\frac{1}{\alpha-1} \quad (6.10)$$

where equation (6.9) was inserted to obtain the final result.

To gain more insight into these equations, we consider the values tabulated in table 6.2 for various loop filters. It can be seen that a value of $\alpha = 2$ gives us the situation where the SAR and loop filter power should be equal to each other. A decrease in α-value from here yields more power to the SAR, while an increase yields more power to the filter. This seems logical if we think that "better" filters should intuitively be allowed to use more power compared to the SAR.

Table 6.2: Optimal SAR ENOB and SAR power values obtained using the simplified, analytical model.

α – ENOB per octave	Modulator order	E_{opt} – opt. SAR ENOB	P_{opt} – opt. SAR power
1.5	1. order	$E_{S,ref}+1$	$2\,P_F$
2	–	$E_{S,ref}$	P_F
2.5	2. order	$E_{S,ref}-0.58$	$0.67\,P_F$
3.5	3. order	$E_{S,ref}-1.3$	$0.4\,P_F$
4.5	4. order	$E_{S,ref}-1.8$	$0.29\,P_F$

These results are interesting themself, because they reveal that an optimum SAR bit count actually exists, and shed some light over the mechanisms governing the placement of the optimum design

points. The relations, and perhaps especially the power consumption relation, can however be useful in practice if proven accurate enough. Consider for instance a close-to-finished modulator design at the transistor level, where one knows the SAR and loop filter power from simulations. The numbers can then be plugged into the formula, and it is possible to find out if it could be benefitial to try to add or remove a bit from the SAR.

Verification

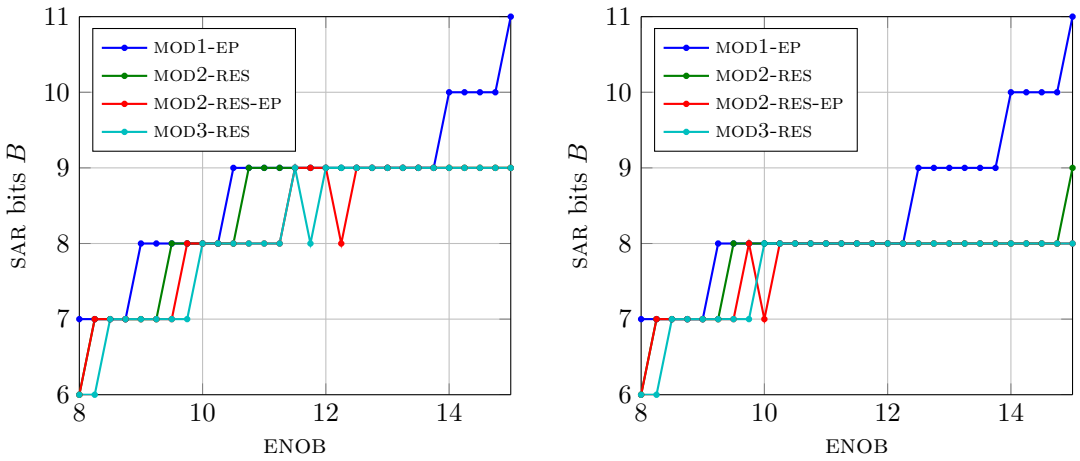
It is necessary to test the developed formulas on some behavioral simulations to verify their accuracy. Consider the simulation discussed earlier in this section, where a first order loop filter uses 20 µW and a 10 bit SAR uses 40 µW. $E_{S,ref}$ is thus at 9 bit for this situation, since the SAR power is then scaled to 20 µW. It is acceptable to consider bit counts rather than E_S values since we assume a constant offset between these quantities. Similarly, a second order modulator which uses 40 µW will get its reference position at 10 bit, and a third order using 60 µW at 10.58 bit.

We can then plug these values into the appropriate rows of table 6.2 to obtain the optimal SAR bit counts. This yields 10 bit for 1. order, 9.42 bit for 2. order, and 9.28 bit for 3. order. When comparing this to figure 6.5a, we see that the SAR bit count indeed stabilizes at 9 bit when this does not imply corresponding OSR values located off the ENOB map. Also, the first order modulator stabilizes at 10 bit a little later. This must be considered as good correspondance to the predictions. The reason for the delay in the MOD1-EP stabilization can probably be explained by the fact that it exhibits higher ENOB per octave values when the OSR is low (figure 6.3). Altough this is also true for the other modulators, this phenomenon probably only draws their optimum values closer to 9.0 bit.

Figure 6.6 shows two other situations where $P_{F,0}$ has been decreased by a twofold and a fourfold respectively. By using the same approach as for the first situation, we obtain optimums of 9 bit for 1. order, 8.42 bit for 2. order, and 8.28 bit for 3. order for the twofold decrease, and correspondingly 8 bit, 7.42 bit, and 7.28 bit for the fourfould decrease. This should just imply a shift downwards of the patterns observed in figure 6.5a. What happens instead, is that all the modulators stabilize at 9 and 8 bit respectively, i.e. the optimums for MOD1-EP. The effect is also more pronounced for the fourfold decrease (the climbs that happen in the right part of the plots is just because the modulators start to hit the "OSR-roof" of the current ENOB-map).

This can perhaps be understood by realizing that although we actually get one extra ENOB per bit increase for a pure SAR, this is not necessarily true for a NS-SAR. This was shown in figure 6.1, where all the simulated modulators exhibit more-than-one-ENOB per bit behavior. Moreover, the effect is largest at low bit counts, and this can probably bias the predicted optimums a bit up when operating in these regions.

The conclusion is that the simplified analytical model can predict the optimums reasonably well in regions where neither the OSR nor the the bit counts are too low. When this is not true, there is a bias towards higher bit counts than predicted because of more-than-one-ENOB per bit behavior, or towards lower bit counts because of large ENOB per octave values. Nevertheless a rule of thumb saying that *for 1.-3. order modulators, the optimum is where the 1. order filter and the* SAR *uses equal amounts of power, or where the 2. order filter and the* SAR *uses equal amounts of power*, is valid for all the considered situations. Also, when the bit counts are low, the second statement is most likely the right one.



(a) $P_{\mathrm{SAR},0} = 40\,\mu W$ and $P_{\mathrm{F},0} = 10\,\mu W$. **(b)** $P_{\mathrm{SAR},0} = 40\,\mu W$ and $P_{\mathrm{F},0} = 5\,\mu W$.

Figure 6.6: SAR bits values picked by the OPT-MOD algorithm for various power shares.

6.2.2 Discussions

In this section, it has been confirmed that NS-SARs have the ability to be energy-efficient at medium/high accuracies. This is in contrast to pure SARs, which suffer from thermal noise in the same regions.

It was found that the FOM of NS-SARs decreases as the specified ENOB is increased, and that high order modulators experience a steeper slope of decrease. Low order modulators do, however, get a smaller "insertion penalty" and are thus best at relatively low ENOB values. Because of this, it is not possible to pick a modulator that is the optimal one under all conditions.

In the FOM curves in figure 6.4a however, where a resonably realistic power assumption was used, MOD1-EP is best until an ENOB of 12 is reached. This suggest that this modulator probably is the optimal one in most realistic situations. When the FOM equals 12, there is nevertheless a draw between MOD1-EP and MOD2-RES-EP. It could thus be interesting to implement both modulators at the transistor level to get a better estimate of the FOM, not possible to achieve using approximate behavioral simulations. It seems however reasonable to believe that a transistor level comparison will be to the advantage of MOD1-EP because of its simplicity. It will also have the smallest footprint.

To better understand the locations of optimal design points, a simplified mathematical model was developed. This revaled that there exists an optimum bit count for the SAR, and also an equivalent optimal power division between the SAR and the loop filter. These optimums are always given relative to the design point where SAR and filter uses the same amount of energy, and are also a function of the ENOB per octave parameter.

The model was found to function quite good at high bit counts and high OSR, because these conditions lead towards a scenario where the assumptions about constant ENOB per octave, and one ENOB per bit count come closer to the truth. At lower bit counts and OSR, errors were observed, but the model was nevertheless able to predict a two-bit region where the optimum should be. The accuracy is considered to be good enough to give useful help at the transistor level, where it may be too impractical and time-consuming to do an exhaustive search through different combinations of bit counts and OSR. Before transistor level design starts, behavioral ENOB-mapping followed by invocation of the OPT-MOD algorithm does still provide the best estimate.

It is probably feasible to extend the model to also predict the optimal value for the ENOB per octave parameter, and this can shed some more light on the optimal modulator choice, which is so far only studied in the plots. The prediction formulas that are most useful in practice are nevertheless believed to be the ones already developed for optimal bit count and power division.

Chapter 7

General discussions

As the results found in this thesis was discussed in chapter 6, the aim of this chapter is to discuss the work carried out in general. First, some strengths and weaknesses in both the modeling and the NTF synthesis will be pointed out and discussed. After this, the specific loop filter topologies considered in this thesis are considered, and it is discussed if there exists a large need to test others.

7.1 NS-SAR modeling and power consumption estimation

There exists a lot of parasitics in a practical ADC that are not modeled in the SAR (in the time domain simulator) or the loop filters in this thesis. In fact, the only parasitic effect that is modeled is the thermal noise in the comparator, which is included due to its potentially large impact on the ADC performance and energy efficiency.

The most important effect which is not modeled is probably the linearity of the SAR DAC. It was pointed out in the beginning of chapter 6 that this have to be as accurate as the whole NS-SAR, and it can therefore be expected to quickly become the main constraint as the target ENOB is increased. In spite of this, the DAC linearity is reasonably easy to consider when simulation results are interpreted, if one assume that an energy efficient, linear DAC can be realized up to a certain accuracy. This was done during the discussions in chapter 6, and it was there also found to be a reasonable assumption. Linearity constraints have thus been accounted for even if it is not explicitly implemented in the models. From chapter 6, it should also be evident that it can actually be an advantage to not implement DAC unlinearity in the models, because it is then possible to study theoretical developments also across the high-accuracy region.

The $\frac{kT}{C}$-noise in the DAC is neither modeled in the NS-SAR simulator. It was stated in section 3.3 that this noise is not noise-shaped (but nevertheless oversampled), and this means that the NS-SAR is really not immune to it, compared to the comparator noise. The noise could have been added to the simulations, but this would not reveal very much information as one have to assume that the DAC always is scaled such that its noise not compromises the overall ADC accuracy. It is more interesting to be able to model the increased power consumption that this DAC scaling due to noise gives rise to. This would call for a more advanced power consumption model, that takes into account the power consumption in the individual circuit blocks. Instead of doing this, it has just been assumed that the DAC power consumption never dominates, as this is often seen to be the truth in the literature [4, 17]. However, the overall validity of this assumption will be a very interesting question in future transistor level NS-SAR implementations.

Another important family of circuit parasitics are the ones belonging to the loop filter itself, for instance noise, linearity, settling time errors, finite amplifier gain, and so forth. In general, these parasitics has not been considered in this thesis, and there could thus possibly be some important effects to discover during transistor level design. A power consumption from a transistor level integrator implementation has, however, been used as input to the power consumption model, and some parasitic effects are thus already baked into this number. Also, the absolute values of the FOM has not been of primary concern in this thesis. More important questions are therefore if the assumption of linear power scaling with respect to the integrator count is correct, and if things like the addition of extra poles or the realization of a resonator do really not affect the power consumption.

When it comes to finite amplifier gain in the integrators, this was in fact modeled during this work, but the effect was found to be negligible in the presented loop filters due to low order and low OSR (this was valid down to really low gain values such as 30 dB and so on). The models were therefore not used during the final ENOB-mapping (i.e. the gain was set really high to just turn them off), and it was chosen to not treat the topic of finite amplifier gain extensively in this thesis.

All in all, the needed modeling accuracy really depends on the intended use. The goal of this work was to explore the whole architectural design space for NS-SARs, and investigate how this ADC type could handle usual constraints for energy efficiency, such as comparator noise. For this, the designed models do probably have sufficient accuracy. The next logical step is probably to do a transistor level implementation of an NS-SAR, and rather come back and improve the models after this.

7.2 NTF synthesis

The CLANS algorithm has been central in the NTF synthesis, and this yields automatically generated NTFs with close-to-optimum chosen poles. However, one must remember that the Kenney-Carley stability criterion that serves as a basis for this, is very conservative and yields filters that should be theoretically stable in *all* cases. It is therefore in most practical cases possible to drive the modulator at larger input signals than that used as input to CLANS, and still not experience any instability. That is, the CLANS algorithm underestimates the stability of the modulator in most practical cases. This is nevertheless not a critical problem in this work if the algorithm "treats" all design points equally, such as a fair comparison can take place. This is assumption is probably correct because of the simplicity of the stability criterion, and also seems plausible from the smoothness of the ENOB-per-bit and ENOB-per-OSR curves that was presented in section 6.1.

When a single design point has been chosen for an ADC however, there is probably always a potential to optimize the NTF even more, or increase the maximum input amplitude from that predicted by the NTF synthesis. The last mentioned is pretty easy, because one can just do a lot of time domain simulations in the chosen design point with various input amplitudes, and pick the amplitude which yielded the highest ENOB. Although the modulator is stable for a sinusoid of a certain amplitude, it can still be unstable when presented for other kinds of waveforms of the same amplitude. This is also a consideration that must be investigated during practical ADC design.

When it comes to the above mentioned curves, there was also some questionable data points which was mentioned when they were introduced in chapter 6. It was believed that this was due to some problems during the actual optimizer run in CLANS, and the current implementation of the algorithm can therefore not be said to be reliable enough from design point to design point. That is, it is not acceptable to just query OPT-MOD for the optimal design point for an ADC specification; one also have to do some sweeps to check the validity of the ENOB map in the

vicinity of the design point. When doing systematic exploration of the design space, the CLANS problems are less critical because it is pretty evident from plots if some single data points are outliers and thus not trustworthy.

7.3 Loop filter topologies

Although it was not easy to draw a definite conclusion, the first order modulator with extra pole was found to be the best choice for most practical situations in chapter 6. This means that it is probably not of primary importance to test modulators of higher order for NS-SAR use. However, because of the found importance of poles, it could still be interesting to try to have four poles in a filter as this can result in two complex-conjugated pairs. It could also be interesting to include simulations without the resonator parts of the loop filters enabled (i.e. all zeros at DC). Especially for second order modulators, this should not yield a much worse FOM and ENOB, and it is a good question if the theoretical 3.5 dB extra accuracy (see section 4.3.1) really is sufficient to justify the increased implementation complexity.

As also discussed in chapter 6, all the loop filters delivers quite comparable performance, and the numerical simulations is thus probably not accurate enough to really tell which is best. This calls for transistor level implementation of more than one modulator, and perhaps also an extension of the mathematical model developed in 6.2.1 to also predict the optimum modulator order.

Chapter 8

Conclusion

In this work, a comprehensive study of the noise-shaping SAR architecture has been performed at the architectural level. Questions of primary concern during this study has been the energy efficiency of NS-SARs, their immunity to thermal noise compared to normal SARs, and the optimum choice of architectural level design variables such as SAR bit count and OSR. As the concept of NS-SARs is relatively new, there is still not written very much about it in the literature. Among others, no generalized treatment that is independent of concrete loop filter topologies was found. Because of this, it was necessary to generalize the NS-SAR concept, and develop a general linear model in this thesis.

The main aid in the exploration of NS-SARs has been a custom-developed behavioral simulation framework, which has the ability to map the ADC accuracy as a function of design variables. The accuracy in each design point is stated in the form of ENOB, and is found by doing a behavioral time domain simulation followed by an FFT. These results are used together with power consumption models for the loop filter and the SAR to predict the energy efficiency. Four different loop filters has been chosen and used during the simulations, and one of them has not been seen used anywhere else. The noise transfer functions of the loop filters are synthesized automatically by the simulation framework every time design variables are changed, and it is thus easy to do a sweep across many design points.

Using the simulation framework, it was found that NS-SARs can operate energy-efficiently at medium/high accuracies, where the energy efficiency of standard SARs are seriously compromised by thermal noise. Of the loop filters that was tested, one of first order (i.e. having one zero), but with two poles turned out to yield the most energy-efficient operation in most practical cases. All the loop filter performed quite comparable in the simulations however, so more investigation might be needed to definitely conclude on how complex the loop filter should be. The importance of using the poles in the NTF to enhance the performance was evident in all the loop filters, and configurations having more poles than zeros was therefore found to be promising. Poles can be realized passively without significant increase of power consumption.

The location of the most energy efficient design points was also investigated using the simulation framework (i.e. which SAR bit count and OSR to choose given a target ENOB). This was done by means of a search algorithm called OPT-MOD, which searches already done simulations for the most energy efficient design point when given an ADC specification. The locations found by this algorithm was further investigated through the development of a simplified, analytical mathematical model for the energy efficiency. From this, it was found that an optimal bit count for the SAR exists as a function of the power division between the actual SAR and the loop filter, as well as the loop filter order. The results from this model was found accurate enough to function

as an approximation having practical value during design work.

8.1 Further work

This thesis has provided a lot of useful knowledge about the NS-SAR performance and trade-offs at the architectural level. The next logical step is to design a NS-SAR at the transistor level to test the validity of the results found and the assumptions done in this work. It can then be interesting to implement more than one of the loop filters since the behavioral simulations carried out was a bit inconclusive regarding the relative performance between them. A transistor level implementation of an NS-SAR having the specification stated in the introduction of this thesis will probably be done as a master thesis based on this work.

After a transistor level implementation, it will probably be possible to improve the behavioral models based on new knowledge. They can thus serve even better as a practical design tool together with the behavioral simulation framework.

Bibliography

[1] B. Murmann, "Energy limits in A/D converters," in *2013 IEEE Faible Tension Faible Consommation (FTFC)*, Jun. 2013, pp. 1–4. DOI: 10.1109/FTFC.2013.6577781.

[2] D. Zhang, C. Svensson, and A. Alvandpour, "Power consumption bounds for SAR ADCs," in *2011 20th European Conference on Circuit Theory and Design (ECCTD)*, Aug. 2011, pp. 556–559. DOI: 10.1109/ECCTD.2011.6043594.

[3] T. Sundstrom, B. Murmann, and C. Svensson, "Power dissipation bounds for High-Speed nyquist Analog-to-Digital converters," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 3, pp. 509–518, Mar. 2009, ISSN: 1549-8328. DOI: 10.1109/TCSI.2008.2002548.

[4] P. Harpe, E. Cantatore, and A. van Roermund, "A 10b/12b 40 kS/s SAR ADC with Data-Driven noise reduction achieving up to 10.1b ENOB at 2.2 fJ/Conversion-Step," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 12, pp. 3011–3018, Dec. 2013, ISSN: 0018-9200. DOI: 10.1109/JSSC.2013.2278471.

[5] H. Tai, Y. Hu, H. Chen, and H. Chen, "11.2 a 0.85fJ/conversion-step 10b 200kS/s subranging SAR ADC in 40nm CMOS," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, Feb. 2014, pp. 196–197. DOI: 10.1109/ISSCC.2014.6757397.

[6] J. Fredenburg and M. Flynn, "A 90-MS/s 11-MHz-Bandwidth 62-dB SNDR Noise-Shaping SAR ADC," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 12, pp. 2898–2904, Dec. 2012, ISSN: 0018-9200. DOI: 10.1109/JSSC.2012.2217874.

[7] J. G. Kenney and L. R. Carley, "Design of multibit noise-shaping data converters," *Analog Integrated Circuits and Signal Processing*, vol. 3, no. 3, pp. 259–272, May 1993, ISSN: 0925-1030, 1573-1979. DOI: 10.1007/BF01239365. [Online]. Available: http://link.springer.com/article/10.1007/BF01239365.

[8] T. C. Carusone, D. Johns, and K. W. Martin, *Analog Integrated Circuit Design*. Wiley, 2012, ISBN: 9781118092330.

[9] J. de la Rosa, "Sigma-Delta modulators: tutorial overview, design guide, and State-of-the-Art survey," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 1, pp. 1–21, Jan. 2011, ISSN: 1549-8328. DOI: 10.1109/TCSI.2010.2097652.

[10] R. Schreier and G. C. Temes, *Understanding Delta-Sigma Data Converters*. Wiley, Nov. 2004, ISBN: 9780471465850.

[11] K. Kim, J. Kim, and S. Cho, "Nth-order multi-bit sigma-delta ADC using SAR quantiser," *Electronics Letters*, vol. 46, no. 19, pp. 1315–1316, Sep. 2010, ISSN: 0013-5194. DOI: 10.1049/el.2010.1554.

[12] J. Silva, U. Moon, J. Steensgaard, and G. Temes, "Wideband low-distortion delta-sigma ADC topology," *Electronics Letters*, vol. 37, no. 12, pp. 737–738, Jun. 2001, ISSN: 0013-5194. DOI: 10.1049/el:20010542.

[13] R. Schreier, "An empirical study of high-order single-bit delta-sigma modulators," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 8, pp. 461–466, Aug. 1993, ISSN: 1057-7130. DOI: 10.1109/82.242348.

[14] R. Schreier, *Delta sigma toolbox - file exchange - MATLAB central*. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/file_infos/19-delta-sigma-toolbox.

[15] I. The MathWorks, *MATLAB - the language of technical computing*. [Online]. Available: http://www.mathworks.com/products/matlab/?s_tid=hp_fp_ml.

[16] C. Wulff, private communication, 2014.

[17] Y. Chung, M. Wu, and H. Li, "A 12-bit 8.47-fJ/Conversion-Step Capacitor-Swapping SAR ADC in 110-nm CMOS," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. PP, no. 99, pp. 1–9, 2014, ISSN: 1549-8328. DOI: 10.1109/TCSI.2014.2340583.

[18] E. Larsen, private communication, 2014.

List of Figures

List of Tables

Appendix A

Simulation framework source code

In this appendix, selected portions of the simulation framework Matlab code are given. All core functions are covered, but much of the less important code is left out.

A.1 Generic loop filter class

```
classdef nssar_modulator < handle
    %NSSAR_MODULATOR Generic loop filter class.

    properties(Constant)
        N = 100000 %Numbers of points in frequency response.
    end

    %% General modulator properties, with standard values. Can be changed by user
        through set methods.
    properties (SetAccess = private)
        A_dB = 80
        OSR = 4
        bits = 10
 targetOverloadAmplitude = 0.9
 poleMode = 'none'
 optimizedZeros = 0
 compNoiseAmplitude = 0
        fButter = 0.1
        clansRMax = 0.95
        vref = 1
    end


    %% Properties calculated in class.
    properties (SetAccess = private)
        NTF_pre
        clansConstraintDeviation
        Aopt = NaN
    end


    %% Dependent properties belonging to this class.
    properties (Dependent = true, SetAccess = private)
        beta
        gamma
        P_inband
 clansNoiseLevels
    end
```

```
38
39      %% Modulator properties set from child class.
40      properties (Abstract, SetAccess = private)
41          poleOrder
42  zeroOrder
43  clansAllowed
44  butterAllowed
45  optimizedZerosAllowed
46      end
47
48      %%Topology dependent properties. Must be redefined in topology specific child
            classes.
49      properties (Abstract, Dependent = true, SetAccess = private)
50          NTF_post
51          H
52          ABCD
53      end
54
55      methods
56          %%Constructor
57          function o = nssar_modulator(OSR,bits,poleMode,optimizedZeros)
58              o.OSR = OSR;
59              o.poleMode = poleMode;
60       o.optimizedZeros = optimizedZeros;
61              o.bits = bits;
62
63              o.updateNTF_pre;
64          end
65
66
67          %%Function to compute the NTF based on different modulator properties.
68  function updateNTF_pre(o)
69    %Switch on pole mode:
70                switch o.poleMode
71                  case 'none'
72                    p = zeros(o.poleOrder,1);
73                    if(o.optimizedZeros == 1)
74                        z = exp(1i*pi/o.OSR*ds_optzeros(o.zeroOrder,1));
75                    else
76                        z = ones(o.zeroOrder,1);
77                    end
78                  case 'butter'
79                    [~,p,~] = butter(o.poleOrder,2*o.fButter,'high');
80                    if(o.optimizedZeros == 1)
81                        z = exp(1i*pi/o.OSR*ds_optzeros(o.zeroOrder,1));
82                    else
83                        z = ones(o.zeroOrder,1);
84                    end
85                  case 'clans'
86                    clans_ntf = clans_varorder(o.zeroOrder,o.poleOrder,o.OSR, ...
87                                               o.clansNoiseLevels,o.clansRMax,o.
                                                    optimizedZeros);
88                    p = clans_ntf.p{1,1};
89                    z = clans_ntf.z{1,1};
90                  otherwise
91                    disp(['Illegal pole mode chosen. Poles set to 0, ' ...
92                          'and zeros to 1'])
93                    p = zeros(o.poleOrder,1);
94                    z = ones(o.zeroOrder,1);
95                end
96
97                %Make numerator and denominator polynominals.
98                [b,a] = zp2tf(z,p,1);
99                b = [b(o.poleOrder-o.zeroOrder+1:end) zeros(1,o.poleOrder-o.zeroOrder
                      )];
100
101               % Compute how well Clans did its job.
```

```
102             if(strcmp(o.poleMode,'clans'))
103                 h = sum(abs(impz(b,a,100)))-1;
104                 o.clansConstraintDeviation = (h-o.clansNoiseLevels)/o.
                        clansNoiseLevels*100;
105             end
106
107             %Give a warning if clans has returned something unexpected.
108             if (strcmp(o.poleMode,'clans'))
109                 if(a(1) ~= 1)
110                     disp(['First coefficient in pole polynominal ' ...
111                           'originating from clans is not 1!']);
112                 end
113             end
114
115             %Normalize denominator as this is not necessarily done.
116             a = a./a(1);
117
118             %Return the NTF as a cell array
119             o.NTF_pre = cell(2,1);
120             o.NTF_pre{1} = b;
121             o.NTF_pre{2} = a;
122         end
123
124
125     %%Noise levels to use in Clans
126 function clansNoiseLevels = get.clansNoiseLevels(o)
127         clansNoiseLevels = (o.vref-o.targetOverloadAmplitude)/(o.
                compNoiseAmplitude+o.vref/2^o.bits);
128 end
129
130     %%Compute inband gain/attenuation
131     function P_inband = get.P_inband(o)
132
133         h = o.magFreqz(o.NTF_post{1},o.NTF_post{2});
134         n = length(h);
135         df = 0.5/(n-1);
136         i_bw = floor(n/o.OSR);
137
138         h_sq = 2*(abs(h).^2);
139         P_inband = 10*log10(df*trapz(h_sq(1:i_bw)));
140     end
141
142     %%Function to compute the optimum overload level when using poleMode ==
            clans.
143     function computeAndSetAopt(o,AMin,AMax,devMin,devMax,points)
144         print = 0; %Debug flag.
145
146         %Amplitudes to evaluate.
147         A = linspace(AMax,AMin,points);
148         %Inband gains of returned NTFs.
149         P_INBAND = zeros(1,length(A));
150         %Clans deviation (degree of success).
151         DEVIATION = zeros(1,length(A));
152         %SNR estimator.
153         SNR_REL = zeros(1,length(A));
154
155         for i=1:length(A)
156             %Compute noiseLevels for this case:
157             noiseLevels = (o.vref-A(i))/(o.compNoiseAmplitude+o.vref/2^o.bits);
158
159             %Run Clans:
160             clans_ntf = clans_varorder(o.zeroOrder,o.poleOrder,o.OSR,
                    noiseLevels,o.clansRMax,o.optimizedZeros);
161
162             p = clans_ntf.p{1,1};
163             z = clans_ntf.z{1,1};
164             [b,a] = zp2tf(z,p,1);
```

```
165
166             %Compute constraint deviation:
167             h = sum(abs(impz(b,a,100)))-1;
168             DEVIATION(i) = (h-noiseLevels)/noiseLevels*100;
169
170             %Compute in-band gain:
171             f = o.magFreqz(b,a);
172             n = length(f);
173             df = 0.5/(n-1);
174             i_bw = floor(n/o.OSR);
175
176             f_sq = 2*(abs(f).^2);
177             P_INBAND(i) = 10*log10(df*trapz(f_sq(1:i_bw)));
178
179             %Find a "relative SNR" to base the optimum choise
180             %upon
181             SNR_REL(i) = 20*log10(A(i))-P_INBAND(i);
182
183         end
184
185         %Print vectors if debugging.
186         if(print)
187             A %#ok
188             P_INBAND %#ok
189             DEVIATION %#ok
190             SNR_REL %#ok
191         end
192
193         optFound = 0;
194         %Find the best amplitude. Discard amplitudes which
195         %violates constraints.
196         for i=1:length(A)
197
198             %Find the current best amplitude.
199             [~,iMax] = max(SNR_REL);
200
201             %Constraints violated?
202             if(DEVIATION(iMax) >= devMax || DEVIATION(iMax) <= ...
203                devMin)
204                 %Set the point to NaN if yes.
205                 SNR_REL(iMax) = NaN;
206                 %disp(['Warning: Best clans amplitude discarded ' ...
207                 %      'because of constraints violation.']);
208             else
209                 %Break if the current best is a legal point.
210                 optFound = 1;
211                 break;
212             end
213         end
214
215         %Return the optimum amplitude, if any. Also update
216         %modulator on success.
217         if(optFound == 0)
218             disp('Warning: No optimal amplitude found')
219             o.Aopt = NaN;
220         else
221             o.Aopt = A(iMax);
222             o.targetOverloadAmplitude = A(iMax);
223             o.updateNTF_pre;
224         end
225     end
226
227
228     %%Function to make a plot showing frequency response and poles/zeros of NTF
          and H.
229     function nicePlot(o,mode)
230         if(strcmp(mode,'pre'))
```

```
231                 NTF = o.NTF_pre;
232             else
233                 NTF = o.NTF_post;
234             end
235 
236             %Reference NTF (differentiator).
237             one_matrix = ones(o.zeroOrder);
238             [REF_b,REF_a] = zp2tf(one_matrix(:,1),0,1);
239 
240             [NTF_h_dB,f] = o.dBFreqz(NTF{1},NTF{2});
241             REF_h_dB = o.dBFreqz(REF_b,REF_a);
242             H_h_dB = o.dBFreqz(o.H{1},o.H{2});
243 
244             %Plot frequency responses and poles and zeroes.
245             subplot(2,2,1)
246             plot(f,NTF_h_dB,f,REF_h_dB,0.5/o.OSR*[ 1  1],[-150 100])
247             title('NTF frequency response')
248             legend('Optimized zeroes','Pure differentiator NTF','Location','
                    SouthEast')
249             xlabel('Normalized frequency')
250             ylabel('|NTF(f)| in dB')
251             grid
252 
253             subplot(2,2,2)
254             zplane(NTF{1},NTF{2})
255             title('NTF poles and zeroes')
256 
257             subplot(2,2,3)
258             plot(f,H_h_dB)
259             title('Modulator filter frequency response')
260             xlabel('Normalized frequency')
261             ylabel('|H(f)| in dB')
262             grid
263 
264             subplot(2,2,4)
265             zplane(o.H{1},o.H{2})
266             title('Modulator filter poles and zeroes')
267 
268             set(gcf,'color','w')
269             set(gcf, 'Position', [500 300 1000 800])
270         end
271 
272         %% Compute a frequency response in dB.
273         function [H_dB,f] = dBFreqz(o,b,a)
274             f = linspace(0,0.5,o.N);
275             H = freqz(b,a,o.N);
276             H_dB = 20*log10(abs(H));
277         end
278 
279         %% Compute a frequency response magnitude.
280         function [H_mag,f] = magFreqz(o,b,a)
281             %N = 10000;  %Points in FR.
282             f = linspace(0,0.5,o.N);
283             H = freqz(b,a,o.N);
284             H_mag = abs(H);
285         end
286 
287         %%Set methods for properties that in turn affect the NTF
288         function changeA_dB(o,A_dB)
289             o.A_dB = A_dB;
290         end
291 
292         function changeOSR(o,OSR)
293             o.OSR = OSR;
294             o.updateNTF_pre;
295         end
296 
```

```
297         function changeBits(o,bits)
298             o.bits = bits;
299             o.updateNTF_pre;
300         end
301 
302         function changeTargetOverloadAmplitude(o,targetOverloadAmplitude)
303             o.targetOverloadAmplitude = targetOverloadAmplitude;
304             o.updateNTF_pre;
305         end
306 
307         function changePoleMode(o,poleMode)
308             if(o.clansAllowed == 0 && strcmp(poleMode,'clans'))
309                 disp('Clans not supported in this modulator');
310             elseif (o.butterAllowed == 0 && strcmp(poleMode, ...
311                                                    'butter'))
312                     disp(['Butterworth poles not supported in this ' ...
313                           'modulator']);
314             end
315             o.poleMode = poleMode;
316             o.updateNTF_pre;
317         end
318 
319         function changeOptimizedZeros(o,optimizedZeros)
320             if(o.optimizedZerosAllowed == 0 && optimizedZeros ~= 0)
321                 disp(['Optimized zeros not supported in this ' ...
322                       'modulator']);
323             else
324                 o.optimizedZeros = optimizedZeros;
325                 o.updateNTF_pre;
326             end
327         end
328 
329 
330         function changeCompNoiseAmplitude(o,compNoiseAmplitude)
331             o.compNoiseAmplitude = compNoiseAmplitude;
332             o.updateNTF_pre;
333         end
334 
335         function changeFButter(o,fButter)
336             o.fButter = fButter;
337             o.updateNTF_pre;
338         end
339 
340         function changeClansRMax(o,clansRMax)
341             o.clansRMax = clansRMax;
342             o.updateNTF_pre;
343         end
344 
345         function changeVref(o,vref)
346             o.vref = vref;
347             o.updateNTF_pre;
348         end
349 
350         %%Get methods for finite OTA gain coefficients.
351         function beta = get.beta(o)
352             alpha = 1;  %C1/C2.
353             A = 10^(o.A_dB/20);
354             beta = (alpha*A)/(A + alpha + 1);
355         end
356 
357         function gamma = get.gamma(o)
358             alpha = 1;  %C1/C2.
359             A = 10^(o.A_dB/20);
360             gamma = (A + 1)/(A + alpha + 1);
361         end
362     end
363 end
```

A.2 Example of specific loop filter class

```
%Second order resonator loop filter with extra pole using feed forward structure.
classdef MOD2_RES_FIF < nssar_modulator

    %%Misc. settings.
    properties (Constant)
        DEF_POLE_MODE = 'clans';
        DEF_OPT_ZEROS = 1
    end

    %%Definition of abstract superclass properties. Essentially sets constraints
        defined by topology.
    properties (SetAccess = private)
        poleOrder = 3
 zeroOrder = 2
 clansAllowed = 1
 butterAllowed = 1
 optimizedZerosAllowed = 1
    end

    %%Definition of abstract superclass properties. Topology dependent quantities.
    properties (Dependent = true, SetAccess = private)
        NTF_post
        H
        ABCD
    end

    %%Topology related quantities, i.e. coefficients.
    properties (Dependent = true, SetAccess = private)
        g1
        a1
        a2
        a3
    end

    methods
        %%Constructor
        function o = MOD2_RES_FIF(OSR,bits,compNoiseAmplitude)
            %Initializing superclass
            o@nssar_modulator(OSR,bits,MOD2_RES_FIF.DEF_POLE_MODE,MOD2_RES_FIF.
                DEF_OPT_ZEROS);
            o.changeCompNoiseAmplitude(compNoiseAmplitude);
        end

         %%Get methods for topology parameters
        function g1 = get.g1(o)
            g1 = o.NTF_pre{1}(2) + 2;
        end

        function a1 = get.a1(o)
            a1 = o.NTF_pre{2}(2)+2-o.g1;
        end

        function a2 = get.a2(o)
            a2 = o.NTF_pre{2}(3)+o.a1-1;
        end

        function a3 = get.a3(o)
            a3 = o.NTF_pre{2}(4);
        end

        %%Get methods for dependent properties
        %NTF derived from coefficients
        function NTF_post = get.NTF_post(o)
            NTF_post = cell(2,1);
            NTF_post{1} = [1 (o.g1-2) 1];
```

```
            NTF_post{2} = [1 o.g1+o.a1-2 o.a2-o.a1+1 o.a3];
        end

        %Loop filter function H
        function H = get.H(o)
            H = cell(2,1);
            H{1} = [o.a1 o.a2-o.a1 o.a3];
            H{2} = [1 (o.g1-2) 1 0];
        end

        %State space matrix ABCD
        function ABCD = get.ABCD(o)
             ABCD = [0    0    0                0    0 1;
                     1    1    -o.g1            0    0 0;
                     1    1    (1-o.g1)         0    0 0;
                     0    0    1                0    0 0;
                     o.a1 o.a1 (o.a2-o.a1*o.g1) o.a3 0 0];
        end
    end
end
```

A.3 Time domain simulator

```
function [dout,vres,vmod,X] = simulateNSSAR(vin,ABCD,bits,vref)
%SIMULATENSSAR ns_sar time domain simulator.

%Generation of DAC voltages.
lsb = vref/2^bits;
coeffs = zeros(1,bits);
for i=1:bits
    coeffs(i) = 2^(bits-i);
end

%Generate comparator noise.
N = length(vin);
comp_noise = randn(bits,N)*sqrt((2*vref/2^bits)^2/12);

n_vin = 1;  %We have one input.
n_vres = 1; %We have one quantizer/output.

%Derive "order" (i.e. number of states) from the size of the ABCD-matrix.
order = size(ABCD,1) - n_vres;

%Initialize output vectors.
X = zeros(order,N);
dout = zeros(n_vres,N);
vres = zeros(n_vres,N);
vmod = zeros(n_vres,N);

%Split ABCD matrix.
A = ABCD(1:order,1:order);
B = ABCD(1:order,order+1:order+n_vin+n_vres);
C = ABCD(order+1:order+n_vres, 1:order);
%Ignore part of D matrix that tries to map quantizer residues to modulator
%outputs. This is non-causal behavior.
D1 = ABCD(order+1:order+n_vres,order+1:order+n_vin);

%The actual simulation.
for n=1:N
    %1. Compute modulator output for current timestep.
    vmod(:,n) = C*X(:,n) + D1*vin(:,n);

    %2. Run binary search to compute dout and vres for current timestep.
    vres(1,n) = vin(n);
    for z=1:bits
        if(vres(1,n) + vmod(1,n) > 0 + comp_noise(z,n))
```

```
            dout(1,n) = dout(1,n) + coeffs(z);
            vres(1,n) = vres(1,n) - coeffs(z)*lsb;
        else
            vres(1,n) = vres(1,n) + coeffs(z)*lsb;
        end
    end

    %3. Calculate states for the next timestep.
    if(n<N)
        X(:,n+1) = A*X(:,n) + B*[vin(:,n);vres(:,n)];
    end
end
end
```

A.4 OPT-MOD algorithm

```
function [bits, osr, fom, fs] = optimal_mod(modNum,enobMin,enobMax,bw,fsMin,fsMax,
    P_nomMod,P_nomSar,ENOB,BITS,OSR)


%%1. Find values in ENOB(:,:,modNum) that lies in the valid enob range. Save the
    indices and the values.
ENOB = ENOB(:,:,modNum);
[iEnob(:,1),iEnob(:,2)] = find(ENOB(:,:)>=enobMin & ENOB(:,:)<=enobMax);
enob = ENOB((ENOB(:,:)>=enobMin & ENOB(:,:)<=enobMax));


%%2. For all these data points, compute FOM. Enob value from ENOB, bw is input,
    needed fs from 2*bw*osr (osr value picked up from OSR).

%Compute the needed fs for every valid modulator:
FS = 2*bw*OSR(iEnob(:,2))';

%Just instantiate the actual modulator with some dummy values to be able to read
    out some properties.
[~,zeroOrder,~] = mod_info(modNum);

%Compute FOM.
FOM = nssarFOM(enob,bw,FS,BITS(iEnob(:,1))',zeroOrder,P_nomMod,P_nomSar);

%%3. Search for the lowest FOM value, and return the corresponding quantizer bits,
    osr and FOM. Discard modulators where fs is not in the valid range, and give a
    warning every time the best modulator is discarded.
[~,iMin] = min(FOM);

bits=BITS(iEnob(iMin,1));
osr=OSR(iEnob(iMin,2));
fom=FOM(iMin);
fs=FS(iMin);

%Note: The rejection of design points with invalid fs is not implemented yet.
```

A.5 ENOB-mapper

```
tic

clear all
close all

poolobj = parpool(16);

rng(12345)

%Declaration of variables/settings.
```

```
OSR_init = 4;
A_dB = 80;          %Integrator OTA gain.
vref = 1;           %ADC reference voltage.
amp = 0.9*vref;     %Initial input signal amplitude
fsig_target = 1e6;  %Wanted input signal frequency.
fs = 32e6;          %Modulator sample rate.
N = 2^18;           %Number of points in input signal and FFT.
use_hann = 1;       %Switch for Hanning window in FFT.
plot_en = 0;        %Switch for FFT plot.
bits_init = 8;      %Use a value for bits when the modulators are instantiated.
compNoiseAmplitude_init = sqrt((2*vref/2^bits_init)^2/12); %Initial Amplitude of
    comparator noise.

%FOM settings
P_0_sar = 40.6e-6;
P_0_mod = 10e-6;
c = 1;
f_s_0 = 32e6;   %Reference sample frequency.
B_0 = 10;       %Reference number of bits.
Bth = 10;       %"Pain" threshold SAR power model
BthWidth = 1;   %"Knee width" SAR power model.

%Aopt settings
Amin = 0.5;
Amax = 0.95;
devMin = -20;
devMax = 5;
points = 20;

%Values to sweep.
BITS = 4:12;                %Bits to sweep
OSR = 2.^linspace(1,3,201); %OSR octaves to sweep
[~,mod_num] =  mod_inst(1,OSR_init,bits_init,compNoiseAmplitude_init); %Loop filter
     count.

%Values to compute.
ENOB = zeros(length(BITS),length(OSR),mod_num);

%Loop filter array.
mods = cell(length(BITS),1);

for i=1:mod_num
    fprintf('Starting simulation on modulator %i...\n',i)
    parfor j=1:length(BITS)
        compNoiseAmplitude = sqrt((2*vref/2^BITS(j))^2/12);
        mods{j} = mod_inst(i,OSR_init,bits_init,compNoiseAmplitude_init);
        mods{j}.changeA_dB(A_dB);
        mods{j}.changeVref(vref);
        mods{j}.changeBits(BITS(j));
        mods{j}.changeCompNoiseAmplitude(compNoiseAmplitude);
        ENOB_TEMP = zeros(1,length(OSR));
        for k=1:length(OSR)
            mods{j}.changeOSR(OSR(k));
            mods{j}.computeAndSetAopt(Amin,Amax,devMin,devMax,points);
            if(isnan(mods{j}.Aopt))
                ENOB_TEMP(k) = NaN;
            else
                [vin,~,cycles] = input_sine(mods{j}.Aopt,fsig_target,fs,N);
                dout = simulateNSSAR(vin,mods{j}.ABCD,BITS(j),vref);
                dout_amp = 2^BITS(j)/2;
                [~,~,ENOB_TEMP(k)] = dofftsd(dout,dout_amp,use_hann,plot_en,fs,'',
                    OSR(k),cycles);
            end
        end
        ENOB(j,:,i) = ENOB_TEMP;
        fprintf('Finished working on %i bits.\n',BITS(j))
    end
```

```
end

save('ENOB_map')

delete(gcp)

toc
```

A.6 FOM vs ENOB post-processing script

```
clear all
close all

%Set to one to plot everything.
everything=1;

%Load ENOB data.
load ENOB_map ENOB BITS OSR;
load sar_map;

%Nominal power consumptions:
P_nomSar = 40e-6;
P_nomMod = 20e-6;

%ADC spec:
bw = 2e6;
fsMin = 32e6;
fsMax = 128e6;

%Enob to sweep:
deltaEnob = 0.25;
enobStart = 8;
enobEnd = 15;
enob = [enobStart:deltaEnob:enobEnd];

[~,~,modTot] = mod_info(1);

%Compute FOM for SAR alone:
sarFom = sarFOM(SARMAP_BITS,SARMAP_ENOB,2*bw,P_nomSar);

for i=1:modTot
    for j=1:length(enob)
        %Compute a tight enob range to look in:
        enobMin = enob(j) - deltaEnob/2;
        enobMax = enob(j) + deltaEnob/2;
        %Invoke opt-mod.
        [bits(i,j),osr(i,j),fom(i,j),fs(i,j)] = optimal_mod(i,enobMin,enobMax,bw,
            fsMin,fsMax,P_nomMod,P_nomSar,ENOB,BITS,OSR);
    end
end

if(everything==1)
    subplot(2,2,1)
end

for i=1:modTot
    semilogy(enob,fom(i,:),'.-')
    hold all
end
semilogy(SARMAP_ENOB,sarFom,'--','color',[0.5 0.5 0.5]);
xlabel('ENOB')
ylabel('FOM')
title('FOM vs. ENOB for optimized modulators')
grid
legend('1. order FIF','2. order RES','2. order RES EP','3. order RES','Pure SAR')
xlim([enobStart enobEnd])
```

```
56
57 if(everything==1)
58 subplot(2,2,2)
59 hold all
60 for i=1:modTot
61     plot(enob,bits(i,:),'.-')
62 end
63 xlabel('ENOB')
64 ylabel('Quantizer bits')
65 title('Quantizer bits vs. ENOB for optimized modulators')
66 grid
67 legend('1. order FIF','2. order RES','2. order RES EP','3. order RES','location','
       NorthWest')
68
69 subplot(2,2,3)
70 hold all
71 for i=1:modTot
72     plot(enob,osr(i,:),'.-')
73 end
74 xlabel('ENOB')
75 ylabel('OSR')
76 title('OSR vs. ENOB for optimized modulators')
77 grid
78 legend('1. order FIF','2. order RES','2. order RES EP','3. order RES','location','
       NorthWest')
79
80 subplot(2,2,4)
81 hold all
82 for i=1:modTot
83     plot(enob,fs(i,:),'.-')
84 end
85 xlabel('ENOB')
86 ylabel('Sampling frequency')
87 title('Sampling frequency vs. ENOB for optimized modulators')
88 grid
89 legend('1. order FIF','2. order RES','2. order RES EP','3. order RES')
90 end
91
92 set(gcf,'color','w')
93 set(gcf, 'Position', [500 300 1.3*1000 1.3*800])
```