Design of a high-speed, high-resolution pipelined AD converter

Harpe, P.J.A.

Award date:
2004

TU/e technische universiteit eindhoven

Design of a High-Speed, High-Resolution Pipelined AD Converter

P.J.A. Harpe

January 22, 2004

Master of Science thesis

Project period: February 2003 - January 2004

Department of Electrical Engineering
Capacity group: Information and Communication Systems
Chair: Mixed-signal Microelectronics

Supervisors:
A. Zanikopoulos M.Sc.
Dr. Ir. J.A. Hegt
Prof. Dr. Ir. A.H.M. van Roermund

Abstract

This thesis presents the design of a 12-bit 100 MS/s analog-to-digital converter (ADC). The converter is designed for the UMC 0.25 μm CMOS process.

Goal of this project is to design a 12-bit ADC composed of simple, identical 'basic blocks'. Because of that, design time is minimized and flexibility is maximized. It was decided to create a pipelined converter, based on stages with 1.5-bit resolution each. With the help of two correction techniques (1.5-bit redundancy and digital post-correction), the requirements for the accuracy of the analog cells are reduced significantly. Digital processing is used to correct for impairments in the analog domain. By making use of these correction techniques, it becomes possible to combine high-resolution, high-speed, simplicity and robustness in a single design. Moreover, the combination of correction techniques and a low resolution per stage reduces the analog design complexity considerably.

This thesis contains two parts. The first part starts with an introduction to high-speed analog-to-digital conversion and an explanation of the used correction techniques. A system-level error-model of the ADC is introduced and the influence of several errors on the converter's overall performance is investigated. Finally, this model results in a set of design constraints to be fulfilled by the analog design.

In the second part of this thesis, the theory of the first part is used to derive a set of design constraints for a specific target: a 12-bit, 100 MS/s ADC. The most critical analog components are designed at transistor-level. Simulations are performed to show that the presented design meets all constraints. Also, the implementation of the digital post-correction algorithm in a FPGA is discussed.

Contents

Chapter 1

Introduction

In this chapter, an introduction to the research project is given, as well as some background information and several design issues of specific importance to this project. Furthermore, an overview of the chapters of this thesis is given.

1.1 Project background

The target of this research project is the design of an analog-to-digital converter (ADC), to be used in combination with a FPGA[1]. FPGA's are used in a large variety of applications, for example telecommunications, video processing, computer systems, etc. For quite a lot of applications, ADC's are used to convert external analog information to a digital representation that can be processed by the FPGA. Each application has its own specific requirements for the ADC. Especially the resolution, speed and power consumption of the converter are important selection criteria. Nowadays, it is necessary to design different ADC's for each different application.

1.2 Design goal

This master thesis project is part of a larger project, aimed at the design of a flexible ADC which can be used in various configurations, serving a broad range of applications. The basic idea is to design a chip in CMOS technology, consisting of simple, identical 'basic blocks'. By making different combinations of these blocks, it is possible to adapt parameters like power consumption, resolution and sample frequency at any moment.

The target of this master thesis project is the design of a 12-bit ADC, with a sample frequency as high as possible, using a pipelined architecture. Although this target is restricted to a pipelined converter, and lacks most of the flexibility, the analog

[1]Field Programmable Gate Array

parts of this converter will be designed such that the possibility of combining these parts in a different topology still remains. Important keywords for this design are flexibility and simplicity of the analog basic cells.

1.3 Chapter overview

Chapter 2 gives a general introduction to ADC's, with pipelined architectures in particular. Two accuracy-improving correction techniques are also discussed.

In chapter 3, the link between error sources and overall performance is analyzed. Moreover, the influence of two possible correction methods on the performance is investigated.

Chapter 4 deals with the transistor level design of the 'basic block' of which the pipelined ADC is composed. Simulation results are given as well.

Chapter 5 discusses the design, implementation and verification of the digital post-correction algorithm. This algorithm is used to improve the accuracy of the converter.

After these chapters, conclusions (chapter 6) are drawn and recommendations for future research (chapter 7) are given.

Chapter 2

High-speed, high-resolution analog-to-digital conversion

In this chapter, an introduction to high-speed analog-to-digital converters is given, with great emphasis on pipelined structures. Two methods (1.5-bit redundancy and digital post-correction) that improve the converter's accuracy are discussed.

2.1 Analog-to-digital conversion

Analog to digital conversion is the process where analog signals are mapped onto digital code representations. Although the analog input signal can be represented in several domains (for example voltage, current, charge), we will assume that the input is in the voltage domain in this thesis. Also, we will assume that the digital codes are described in the binary domain.

The analog input signal $V_{in}(t)$ is defined for each moment in time, and we suppose that it can take any value within a certain range:

$$\forall_t : -A_{max} \leq V_{in}(t) \leq A_{max} \tag{2.1}$$

To represent this signal in a limited amount of digital data, the input signal is sampled on fixed time intervals and quantized in the amplitude domain. An example is given in figure 2.1. The sample frequency f_s of the converter determines the time interval T_s between two consecutive sample moments, according to:

$$T_s = \frac{1}{f_s} \tag{2.2}$$

Sample moment k takes place at time:

$$t = kT_s \tag{2.3}$$

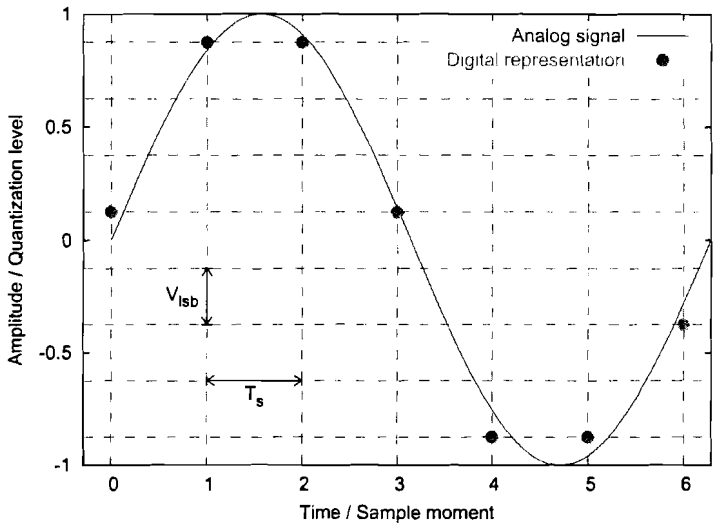


Figure 2.1: *Example of the conversion of an analog signal (line) to a $N = 3$ bit digital representation (dots). The dashed grid indicates the sampling in time and quantization in amplitude.*

Suppose that the digital output code at each sample moment consists of N bits, and the codes are equally distributed over the allowed input range. Then, the distance between each pair of successive codes is given by:

$$V_{lsb} = \frac{A_{max}}{2^{N-1}} \tag{2.4}$$

When the input voltage is rounded to the nearest code, the quantization error $q[k]$ is limited to the range:

$$\begin{aligned} -\frac{1}{2}V_{lsb} \leq q[k] \leq \frac{1}{2}V_{lsb} \\ -\frac{A_{max}}{2^N} \leq q[k] \leq \frac{A_{max}}{2^N} \end{aligned} \tag{2.5}$$

The consequence of sampling in the time domain is that we can only represent input signals with frequencies below $\frac{1}{2}f_s$, as stated by the Nyquist theorem [1]. The quantization of the amplitude results in loss of accuracy. The larger the number of effective bits produced by the converter, the higher the accuracy is.

2.1.1 ADC architectures

In practice, a lot of architectures for ADC's exist. Some architectures are aimed at high speed, where other architectures are aimed at a higher accuracy or lower power consumption. Figure 2.2 displays the speed and accuracy of some high-speed designs that were presented in the last four years. For a fair comparison, only

full-CMOS designs were taken into account. Based on this picture, it is concluded that the pipelined architecture seems to be the best solution for our design goal: a 12-bit converter with a sample frequency as high as possible. To be in line with the performance of the converters from figure 2.2, a sample frequency of at least 75 MHz should be achieved.

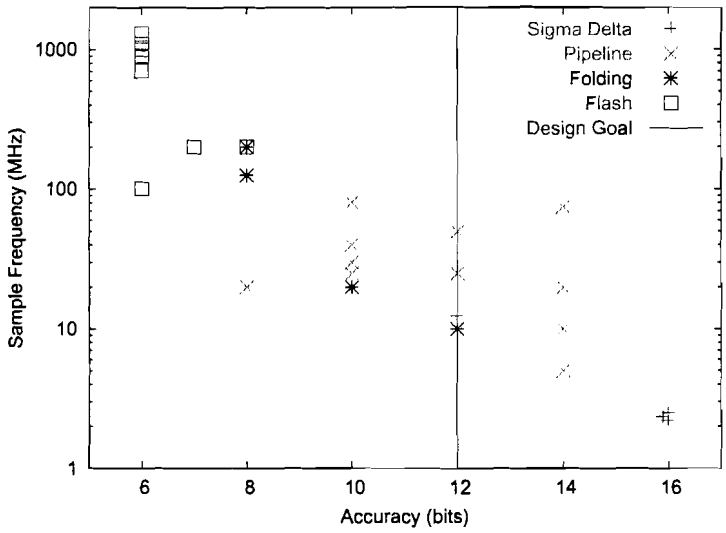


Figure 2.2: *Comparison of state-of-the-art performance in speed and accuracy for Sigma Delta, pipelined, folding and flash architectures. Also indicated is the specific design goal of this project.*

In the following sections, the flash and pipelined architecture will be explained. It will become clear that the flash architecture can hardly combine high speed with high resolution. Thereafter, it will be shown that the pipelined architecture has the potential to achieve both high speed and high resolution at the same time.

Flash architecture

In a flash AD converter, the conversion from the analog to the digital domain is made in a single step. Figure 2.3 shows an example of a $N = 2$ bit flash converter.

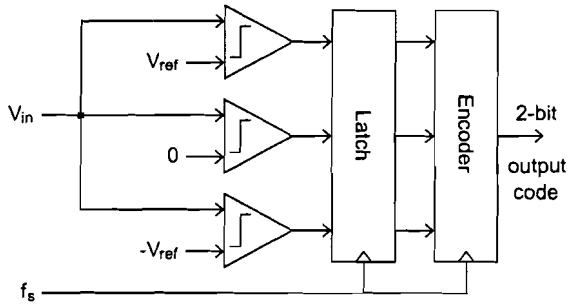


Figure 2.3: *Example of a 2-bit flash AD converter.*

The input signal is applied to a set of $2^N - 1$ comparators, with different reference voltages. The comparator outputs are latched at the sample frequency f_s. An encoder is used to translate the output values of the comparators in a N-bit output code.

The input node of this circuit is the bottleneck, with respect to the speed-accuracy trade-off. The maximum sample frequency of a flash converter is determined by the pole frequency of the input node. The larger the capacitive load at this node, the lower the maximum frequency will be:

$$f_{s,max} \propto \frac{1}{C_{in}} \tag{2.6}$$

The total input capacitance is equal to the number of comparators times the input capacitance of a single comparator (C_{comp}):

$$C_{in} = (2^N - 1)C_{comp} \approx 2^N C_{comp} \tag{2.7}$$

Assuming that the comparators are based on CMOS differential pairs, the input capacitance of a comparator is determined by the size of the input transistors (WL is the transistor's width times its length):

$$C_{comp} \propto WL \tag{2.8}$$

The (random) offset voltage of a comparator determines its accuracy, and is a function of the random deviation of the threshold voltage of the transistors. Using the transistor matching model described in [2] and equation 2.4 gives:

$$\left.\begin{array}{l} \sigma_{V_{off}} \propto \sigma_{V_{th}} \propto \frac{1}{\sqrt{WL}} \\ V_{lsb} \propto 2^{-N} \end{array}\right\} \Rightarrow WL \propto 2^{2N} \tag{2.9}$$

Combining equations 2.6, 2.7, 2.8 and 2.9 yields:

$$f_{s,max} \propto \frac{1}{2^{3N}} \tag{2.10}$$

From this result, it is obvious that increasing the converter's accuracy has to be paid for by a reduced sample frequency. This conclusion is in accordance with picture 2.2, where it is was shown that flash converters are used only for low accuracies ($N \leq 8$ bits).

It should be noted, that the trend estimation from equation 2.10 is rather pessimistic. Techniques like folding and multi-step flash reduce the input capacitive load greatly, and obtain higher speed for moderate accuracies. Roughly, these systems are useful for $N \leq 10$. In figure 2.2 some folding architectures were inserted. It can be

seen that these converters perform better than expected based on equation 2.10. A detailed discussion of all architectures, derived from the flash principle, is beyond the scope of thesis. More information about these architectures can be found in [3], [4] and [5].

Pipelined architecture

Another method of achieving high accuracy without putting an excessive capacitive load on the input node, is the use of a pipelined converter, based on flash sub-converters. The general structure of a pipelined converter is given in figure 2.4. Each block in this picture has an analog input and both an analog and a digital output. The digital output gives a coarse quantization of the input voltage. The analog output represents the quantization error, and will be quantized in the consecutive stages. Digital logic is used to combine the separate digital outputs of each cell, and combine them to provide a valid N-bit output code. In front of the first block of the pipeline a dedicated sample-and-hold stage is placed, sampling the analog input at the sample frequency f_s.

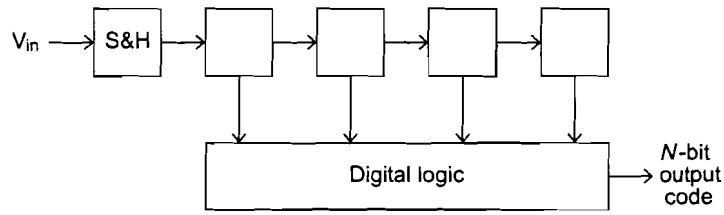


Figure 2.4: *Example of a pipelined AD converter with four stages and an additional sample-and-hold stage in front of the converter. Figure 2.5 shows the contents of each block.*

Figure 2.5 shows the contents of each cell in the pipeline. Each cell resolves n bits of information. When the input voltage of a cell is bounded by:

$$-A_{max} \leq V_{in} \leq A_{max} \tag{2.11}$$

then the quantization error q is bounded by:

$$-\frac{A_{max}}{2^n} \leq q \leq \frac{A_{max}}{2^n} \tag{2.12}$$

The gain of the amplifier in each cell is chosen such to scale the quantization error back to a full scale input again:

$$\left.\begin{array}{l} -\frac{A_{max}}{2^n} \leq q \leq \frac{A_{max}}{2^n} \\ -A_{max} \leq V_{out} \leq A_{max} \end{array}\right\} \Rightarrow \text{gain} = 2^n \tag{2.13}$$

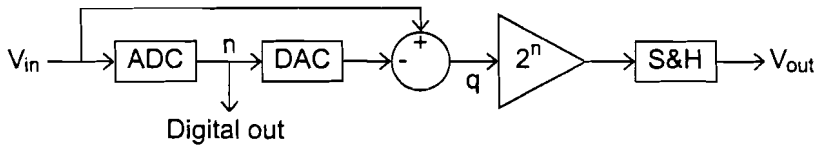


Figure 2.5: *Building block of a pipelined ADC, containing a sub-ADC, sub-DAC, summation node, amplifier and sample-and-hold circuit.*

An important feature of a pipelined converter is that each stage contains a sample-and-hold circuit (S&H). As soon as the stage processed its input voltage, the output is stored into the S&H-circuit. From that moment on, the stage can already start processing the next input value, while its previous output is still available for processing to the subsequent stage. As a result, the maximum sample frequency of a pipelined converter is determined by the time period of a complete conversion cycle of a single stage only. At the same time, the accuracy of the converter can be increased by adding blocks to the back end of the pipeline. Nevertheless, there is still a trade-off between speed and accuracy as increasing the accuracy of the converter implicitly means that each stage will become slower. A more detailed analysis about speed and accuracy is given in chapter 4, where the analog design is dealt with.

One of the critical design issues of a pipelined converter is the number of bits that is produced by each stage. Moreover, the number of bits can vary for each block in the pipeline. By making a correct distribution of the bits over the cells in the pipeline, the overall speed, accuracy, power consumption and chip area can be optimized.

A second design possibility is the application of scaling. It is possible to scale down the physical size of each cell, as a function of its place in the pipeline. (I.e. the first cell in the pipeline is not scaled, the second cell is scaled down with a factor s, the third with a factor s^2, etc.) The accuracy of these down-scaled cells is lower than the accuracy of an unscaled cell, as the mismatch of components increases for smaller devices. However, the *required* accuracy of each cell decreases also as a function of the place in the pipeline (due to the gain of 2^n in each stage), so it is allowed to use less accurate cells. Therefore, proper scaling will decrease both the used chip area and the power consumption, without reducing overall speed or accuracy.

2.2 Correction methods

Normally, the accuracy of each part of the pipelined converter (sub-ADC, sub-DAC, etc.) is chosen such as to fit the overall accuracy of the converter. For high-resolution converters, this means that we have to use physically large components, often resulting in a large chip area, high power consumption and reduced conversion speed. By the use of correction techniques, it is possible to relax the accuracy requirements for the internal components, whereas the overall accuracy remains constant. In this section, two simple correction techniques (code redundancy and digital post-correction) will be explained. In chapter 3, the influence of these techniques on the converter's accuracy is analyzed.

2.2.1 1.5-bit redundancy

The first correction method applied is *code redundancy*, described in [3] and [6]. To explain the reason for applying this correction method, a 1-bit per stage system will be examined first. Then, based on the problems associated with this system, the 1.5-bit system will be introduced.

Basic block with 1-bit per stage

Consider the situation where a pipelined ADC is created, using a 1-bit per stage resolution. Each stage looks like the system in figure 2.5, with $n = 1$. When all components are ideal, the following holds for the sub-ADC and the sub-DAC:

$$D_{out} = \begin{cases} 0 & \text{if } V_{in} \leq V_{ref} \\ 1 & \text{if } V_{in} > V_{ref} \end{cases} \text{, with } V_{ref} = 0 \tag{2.14}$$

$$V_{DAC} = \begin{cases} -\frac{1}{2}A_{max} & \text{if } D_{out} = 0 \\ \frac{1}{2}A_{max} & \text{if } D_{out} = 1 \end{cases} \tag{2.15}$$

The output code of the sub-ADC is indicated with D_{out}, the output voltage of the sub-DAC with V_{DAC}. The output voltage of the stage (equation 2.16), is bounded by $[-A_{max}, A_{max}]$ in the ideal case. This coincides exactly with the allowed input range of the next stage, thus the system works perfectly.

$$V_{out} = 2(V_{in} - V_{DAC}) \tag{2.16}$$

However, suppose that the level of the comparator in the sub-ADC (V_{ref}) is not exactly equal to 0 (as was assumed by equation 2.14). This can be due to static deviation of the reference voltage, or due to dynamic behavior of the comparator. In that case, it is possible that the stage's output voltage exceeds the allowed input range ($[-A_{max}, A_{max}]$) of the next stage, resulting in a large quantization error. Figure 2.6 visualizes the problem. The left picture shows that in the ideal case, the output range of each stage matches the input range of the next stage exactly. The second picture shows that a slight deviation of the reference level V_{ref} immediately results in exceeding the dynamic range of the next stage in the pipeline.

In the ideal case, the digital code, produced by a pipelined converter with k stages, corresponds to the analog equivalent:

$$V_{eq} = \frac{1}{2}A_{max} \sum_{i=0}^{k-1} 2^{-i}(-1)^{1-D_{out}(i)} \tag{2.17}$$

For example, consider the situation where $A_{max} = 1$ V and $k = 10$ stages. An analog input of $V_{in} = 0.1$ V is applied. According to equation 2.14, the first stage in

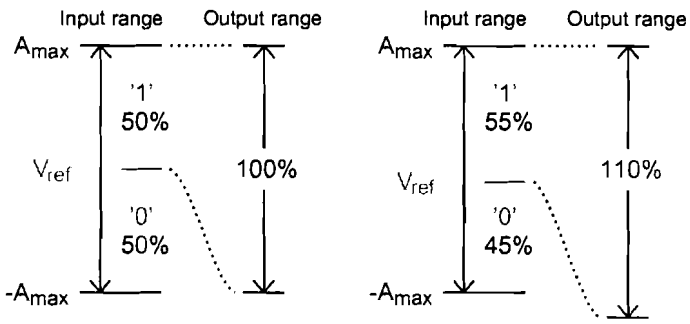


Figure 2.6: *Ideal behavior (left), and behavior in case of deviation of V_{ref} (right) of a 1-bit per stage converter.*

the pipeline produces code 1. The output of the sub-DAC produces 0.5 V (equation 2.15), and the output of the first stage becomes −0.8 V (equation 2.16). Likewise, the signals in the other stages can be calculated. Table 2.1 lists the intermediate values of each stage for this example. The produced digital code, combining the individual outputs of all stages, is 1000110011, thus $V_{eq} = 0.1006$ V.

Stage	0	1	2	3	4	5	6	7	8	9
V_{in}	0.1	-0.8	-0.6	-0.2	0.6	0.2	-0.6	-0.2	0.6	0.2
D_{out}	1	0	0	0	1	1	0	0	1	1
V_{DAC}	0.5	-0.5	-0.5	-0.5	0.5	0.5	-0.5	-0.5	0.5	0.5
V_{out}	-0.8	-0.6	-0.2	0.6	0.2	-0.6	-0.2	0.6	0.2	-0.6

Table 2.1: *Example of a correct analog-to-digital conversion with a 10-stage pipeline.*

Next, consider the situation, where the first stage in the pipeline makes a wrong decision when $V_{in} = 0.1$ V is applied. In that case, code 0111111111 is produced and $V_{eq} = -0.0010$ V. Table 2.2 shows the intermediate values of this conversion. Note that in practice, the analog voltages in the pipeline will saturate to a certain maximum, but this has no influence on the produced digital code. It is clear that the error of the first stage can not be corrected for in the subsequent stages.

Stage	0	1	2	3	4	5	6	7	8	9
V_{in}	0.1	1.2	1.4	1.8	2.6	4.2	7.4	13.8	26.6	52.2
D_{out}	0	1	1	1	1	1	1	1	1	1
V_{DAC}	-0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
V_{out}	1.2	1.4	1.8	2.6	4.2	7.4	13.8	26.6	52.2	103.4

Table 2.2: *Example of an incorrect analog-to-digital conversion with a 10-stage pipeline. The comparator in the first stage made a mistake.*

Basic block with 1.5-bit per stage

The problem of a pipelined converter, using a 1-bit resolution per stage and a gain-of-2 stage, is that a deviation of one of the comparator levels results in exceeding the dynamic range of the next stage in the pipeline. The solution is to use a gain stage with a gain less than 2^n, or to maintain the same gain while increasing the number of bits in the sub-ADC and sub-DAC. The latter solution is used here. Whereas the gain remains equal to 2, the number of bits is increased from 1 to 1.5. The new equations for the sub-ADC and sub-DAC are given below. For convenience, the ADC and DAC levels are described using a ratio relative to the full scale amplitude A_{max}.

$$D_{out} = \begin{cases} 0 & \text{if} \quad V_{in} \leq -R_{ADC}A_{max} \\ 1 & \text{if} \quad -R_{ADC}A_{max} < V_{in} \leq R_{ADC}A_{max} \\ 2 & \text{if} \quad R_{ADC}A_{max} < V_{in} \end{cases} \tag{2.18}$$

$$V_{DAC} = \begin{cases} -R_{DAC}A_{max} & \text{if } D_{out} = 0 \\ 0 & \text{if } D_{out} = 1 \\ R_{DAC}A_{max} & \text{if } D_{out} = 2 \end{cases} \tag{2.19}$$

Ratios R_{ADC} and R_{DAC} can be optimized to allow less or more error correction. The actual value of R_{ADC} is not important for the accuracy: as long as an input voltage in the range $[-A_{max}, A_{max}]$ produces an output value in the same range, a correct digital code can be produced, and no accuracy is lost. Based on equations 2.16, 2.18, 2.19 and the constraint on the in- and output range, the following constraints for the ratios R_{ADC} and R_{DAC} can be derived:

$$R_{ADC} \leq \frac{1}{2} \quad \wedge \quad R_{DAC} \geq \frac{1}{2} \quad \wedge \quad R_{DAC} - R_{ADC} \leq \frac{1}{2} \tag{2.20}$$

The analog equivalent value of a 1.5-bit digital code is given by formula 2.21. This equation is valid under the assumption that R_{DAC} is well known and equal for each stage of the converter. The correction technique described in the next section will also solve deviations of R_{DAC}.

$$V_{eq} = R_{DAC}A_{max} \sum_{i=0}^{k-1} 2^{-i}(D_{out}(i) - 1) \tag{2.21}$$

Figure 2.7 shows an example of the behavior of a basic block with 1.5-bit per stage resolution. In this example, $R_{ADC} = \frac{1}{3}$ and $R_{DAC} = \frac{2}{3}$. In the ideal case, only 67% of the full scale is used. The other 33% is used in case of deviations in the sub-ADC. As long as condition 2.20 is satisfied, errors in the sub-ADC have absolutely no influence on the overall accuracy of the ADC. The price to be paid for is that each 1.5-bit stage effectively resolves less than 1.5 bit. Thus, for an N-bit converter, more than $N/1.5$ stages are required. The exact amount of required stages for a certain accuracy will be discussed in chapter 3.

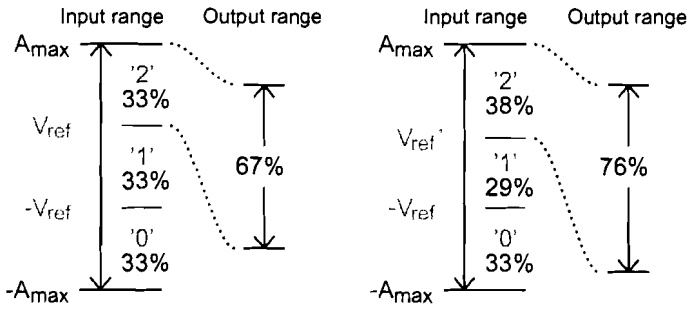


Figure 2.7: *Ideal behavior (left), and behavior in case of deviation of V_{ref} (right) of a 1.5-bit per stage converter. For this example, $R_{ADC} = \frac{1}{3}$ and $R_{DAC} = \frac{2}{3}$.*

To show that subsequent stages are capable of correcting an error of a previous stage, consider a 10-stage pipelined converter with $R_{ADC} = \frac{1}{3}$, $R_{DAC} = \frac{2}{3}$ and $A_{max} = 1.5$ V. When an input voltage $V_{in} = 0.4$ V is applied, and each sub-ADC takes a correct decision, code 1210121012 is produced, yielding $V_{eq} = 0.40004$ V. The intermediate values in the pipeline are listed in table 2.3. When the first sub-ADC produces code 2 instead of code 1, the output of the pipeline will become 2010121012, thus $V_{eq} = 0.4004$ V. For details, see table 2.4. Even though the first decision in the pipeline was wrong, the digital code is an accurate approximation of the input voltage.

Stage	0	1	2	3	4	5	6	7	8	9
V_{in}	0.4	0.8	-0.4	-0.8	0.4	0.8	-0.4	-0.8	0.4	0.8
D_{out}	1	2	1	0	1	2	1	0	1	2
V_{DAC}	0.0	1.0	0.0	-1.0	0.0	1.0	0.0	-1.0	0.0	1.0
V_{out}	0.8	-0.4	-0.8	0.4	0.8	-0.4	-0.8	0.4	0.8	-0.4

Table 2.3: *Example of a correct analog-to-digital conversion with a 10-stage pipeline, using a 1.5-bit resolution per stage.*

Stage	0	1	2	3	4	5	6	7	8	9
V_{in}	0.4	-1.2	-0.4	-0.8	0.4	0.8	-0.4	-0.8	0.4	0.8
D_{out}	2	0	1	0	1	2	1	0	1	2
V_{DAC}	1.0	-1.0	0.0	-1.0	0.0	1.0	0.0	-1.0	0.0	1.0
V_{out}	-1.2	-0.4	-0.8	0.4	0.8	-0.4	-0.8	0.4	0.8	-0.4

Table 2.4: *Example of a corrected analog-to-digital conversion with a 10-stage pipeline, using a 1.5-bit resolution per stage. Although the decision made by the first sub-ADC is wrong, the pipeline corrects the error in the consecutive stages.*

2.2.2 Digital post-correction

The second correction technique applied is based on the method described by [6] and [7]. Its aim is to correct for linear and constant errors in the sub-DAC, summing-node, amplifier and S&H-circuit. Non-linear distortion is not corrected completely, but in section 3.3.2 it is shown that this method reduces the influence of non-linearity on the overall accuracy of the converter. In principle, it is possible to extend the digital post-correction algorithm to correct for non-linearity as well, but this will increase the complexity of the algorithm. For this moment, it is assumed that the linearity of the analog design is such that additional correction is not required.

The basic idea of a *digital post-correction method* (see figure 2.8) is that most errors can be corrected *afterwards (in the digital domain)* as long as the analog output of each block remains in the valid range $[-A_{max}, A_{max}]$. The algorithm performs a simple mapping-function from the uncorrected digital code from the pipeline to the corrected output code of the converter.

In this section, the correction method is described on a system level. Chapter 3 will show which low-level errors can be solved using this technique. First, a system without post-correction will be described, after which the post-correction algorithm is added.

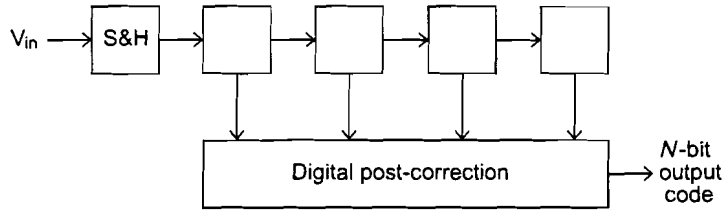


Figure 2.8: *Example of a pipelined AD converter with digital post-correction.*

System without post-correction

When it is assumed that the transfer function of each basic block is ideal, like in figure 2.9, post-correction is not needed. The N-bit output code is generated simply by a summation of constant weights. Each stage adds a single weight, dependent on the produced code of that stage (0, 1 or 2), and the place of the stage in the pipeline. Table 2.5 shows the weights for a 1.5-bit per stage converter with 5 stages.

Stage / Code	0	1	2	3	4
0	$-x$	$-\frac{1}{2}x$	$-\frac{1}{4}x$	$-\frac{1}{8}x$	$-\frac{1}{16}x$
1	0	0	0	0	0
2	x	$\frac{1}{2}x$	$\frac{1}{4}x$	$\frac{1}{8}x$	$\frac{1}{16}x$

Table 2.5: *Example of the weights of each stage in the pipeline, x is an arbitrary constant.*

For example, when the pipeline produces code 01220, the output code becomes:

$$-x + 0 + \frac{1}{4}x + \frac{1}{8}x - \frac{1}{16}x \tag{2.22}$$

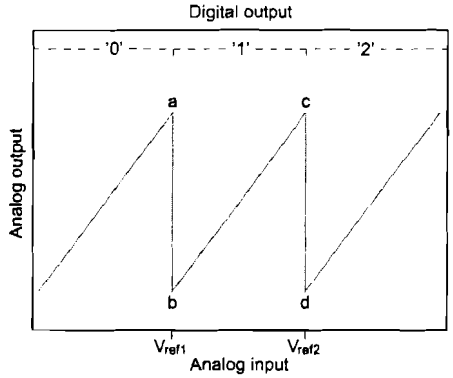


Figure 2.9: *Ideal transfer function of a basic block.*

System with post-correction

Now, consider the situation that the transfer function of each basic block shows a certain deviation from the ideal curve. Moreover, each block can have a different deviation due to mismatch, process spread, etc. In figure 2.10 two examples of non-ideal curves are shown. In the following, it is assumed that the non-ideal curves are always linear. Non linearities are left out of consideration for the moment, but will be included in section 3.3.2.

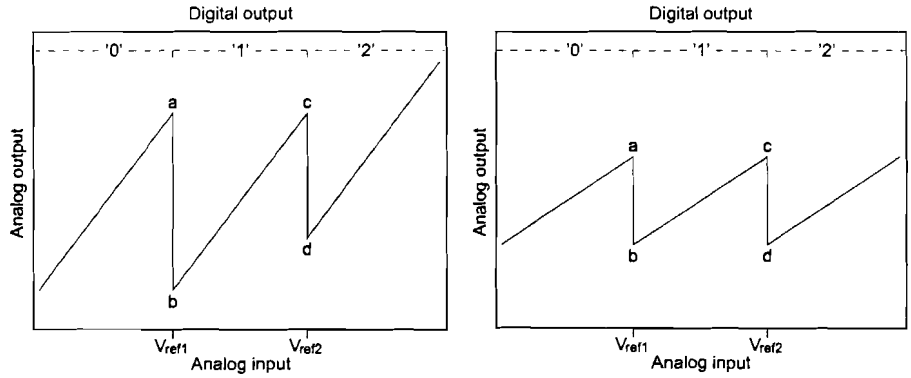


Figure 2.10: *Non-ideal transfer functions of a basic block. Curve with deviation of the DAC-level for code 2 (left), and deviation of the gain stage (right).*

To make clear that it is possible to correct all deviations of the transfer function afterwards, the transfer function of each stage is described with four parameters:

1. The slope of the three linear parts of the curve. (As the same amplifier is used in all regions, the three linear parts always have the same slope.)
2. A constant offset value, equal to the output voltage when the input voltage equals 0 and the digital code equals 1.
3. The transition height of the curve around input level V_{ref1}, the first comparator level.
4. The transition height of the curve around input level V_{ref2}, the second comparator level.

With these parameters, the transfer function can be reconstructed without ambiguity. The first two parameters don't have influence on the overall accuracy of the pipelined converter. The only effect is that they produce a single input referred offset and gain error of the pipelined ADC. Only the deviations of the two transition heights from their ideal value results in loss of overall accuracy. Luckily, the deviations of the transfer heights can be corrected by making use of variable weights, instead of using fixed weights. The correction algorithm measures the transition heights first, and then it derives the optimal values of the weights. It is assumed that the deviations of the transfer function are static, i.e. the deviations are constant during the time of operation. In that situation, it is sufficient to determine the correct weights once at the beginning of operation.

Before explaining the post-correction algorithm, some definitions are introduced:

- the pipelined converter consists of k stages, numbered from 0 (first stage) to $k-1$ (last stage),
- the output code of stage i is indicated with $u(i) \in \{0, 1, 2\}$,
- $U(i)$ is the concatenation of the output codes from stage i up to stage $k-1$:

$$U(i) = u(i)u(i+1)\cdots u(k-2)u(k-1) \tag{2.23}$$

- the variable weights of stage i are indicated with $\omega_0(i)$, $\omega_1(i)$ and $\omega_2(i)$ for code 0, 1 and 2 respectively. As the correction algorithm only needs two degrees of freedom per stage (there are two transition heights that need correction), one weight can be fixed. For simplicity $\omega_1(i) = 0$ will be used,
- $M(\cdot)$ is the mapping function from the output code of the pipeline to the corrected output code of the converter:

$$M(U(i)) = \sum_{j=i}^{k-1} \omega_{u(j)}(j) \tag{2.24}$$

Before the pipelined converter can be used, a measurement procedure is performed to determine the optimal weights for each stage. The measurement procedure measures each stage one by one. Suppose, one would like to measure the weights of stage i. The measurement procedure for $\omega_0(i)$ is as follows:

- the analog input of stage i is set to V_{ref1}, the first reference level of the sub-ADC as indicated in figure 2.9,
- the input of the sub-DAC is forced to code 0. Mark a in figure 2.9 indicates the output voltage of the stage in this situation, named v_a. The output code of the converter from stage i up to $k-1$ now equals:

$$0U_a(i+1) \tag{2.25}$$

  $U_a(i+1)$ is the digital code, produced by the part of the pipelined converter following stage i, when the input voltage of stage $i+1$ equals v_a,
- the input of the sub-DAC is now forced to code 1. Mark b in figure 2.9 indicates the output voltage of stage i. The output code equals:

$$1U_b(i+1) \tag{2.26}$$

This procedure yields two different codes ($0U_a(i+1)$ and $1U_b(i+1)$), both describing the same analog input voltage V_{ref1}. So, after post-correction, the codes should be equal to each other:

$$\begin{aligned} M(0U_a(i+1)) &= M(1U_b(i+1)) \\ \omega_0(i) + M(U_a(i+1)) &= \omega_1(i) + M(U_b(i+1)) \\ \omega_0(i) + M(U_a(i+1)) &= 0 + M(U_b(i+1)) \\ \omega_0(i) &= M(U_b(i+1)) - M(U_a(i+1)) \end{aligned} \tag{2.27}$$

Likewise, the second weight can be measured by applying V_{ref2} in combination with code 1 and 2, yielding:

$$\omega_2(i) = M(U_c(i+1)) - M(U_d(i+1)) \tag{2.28}$$

So, $\omega_0(i)$ and $\omega_2(i)$ can be determined in the digital domain, by making use of the blocks following stage i. Yet, the weights of stage i can be determined only when the weights of the stages following stage i are already known. Therefore, the measurement algorithm starts at the back-end of the pipeline. The last stage $(k-1)$ can not be measured, as there are no consecutive stages anymore, so the values of this stage are fixed to $\omega_0(k-1) = -1$ and $\omega_2(k-1) = 1$. Then, one by one, stages $k-2$, $k-3$, $\cdots$, 0 are measured.

In most calibration systems, the accuracy of the calibration algorithm limits the achievable accuracy of the calibrated converter. However, in this case, the accuracy of the calibration algorithm increases for each additional block that is calibrated. Because of that, the accuracy of the calibration algorithm is not a limit for the achievable accuracy of the pipelined ADC.

The additional requirements for the analog part of the converter, to be able to apply this correction algorithm are:

- the possibility to apply the reference voltages of the sub-ADC to the analog input of each stage,
- the possibility to force the sub-DAC to a desired level, independent on the sub-ADC output.

The additional requirements for the digital system are:

- summation of variable weights instead of fixed weights,
- implementation of the measurement algorithm. This part of the digital system is only needed directly after power-up, and not during normal operation.

Chapter 3 will discuss which errors and to what extent they can be corrected, using this algorithm. In chapter 5, the implementation of the post-correction algorithm is described.

2.3 Design approach

Based on the study on existing high-speed, high-resolution AD converters (section 2.1.1), the pipelined architecture was adopted for our design target: a 12-bit converter. Although the pipelined ADC can be optimized by designing each cell separately (section 2.1.1), our aim is to make the converter flexible and simple. Consequently, our target is to make a pipeline, composed of copies of a single basic block only. This makes it easier to interchange cells, or to use the cells in a different topology without the necessity of re-scaling or redesign. By using a single cell only, the analog design-time and risk are minimized, whilst IP reuse is maximized. A disadvantage of using only one basic block is that the power consumption of the converter will be far from optimum. However, our first target is to create a simple, flexible converter, and power consumption is considered to be of less importance.

The converter will be designed in a standard CMOS technology (UMC 0.25μm-process). Nowadays, CMOS technology is the cheapest, widest available technology. Moreover, it gives the possibility to integrate analog electronics and digital systems on the same chip, without the necessity of a complex, expensive chip technology. Finally, using a CMOS technology without any special features makes it easier to transfer the design to another technology.

To make the basic block of the pipeline as simple as possible, a 1-bit per stage resolution seems to be the most attractive solution. After all, a 1.5-bit resolution per stage is employed. It makes the basic blocks only a little bit more complex, but offers the advantage of code redundancy, which eliminates possible errors in the sub-ADC's (section 2.2.1). Also the digital post-correction algorithm introduced in section 2.2.2 will be employed, reducing the effect of some error sources. In chapter 3, the accuracy of the pipelined ADC in combination with these correction techniques is investigated.

For completeness, figure 2.11 shows the basic cell as it will be used throughout the rest of this thesis. As this is the only building block of the pipelined converter, it will be referred to as **basic block**.

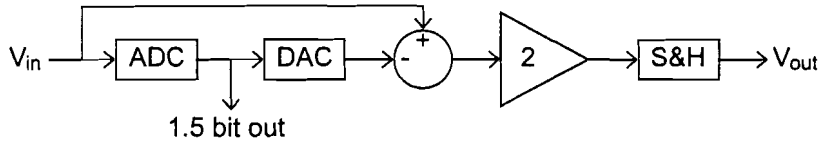


Figure 2.11: *Basic block for a flexible pipelined ADC.*

2.4 Conclusions

In this chapter, an introduction to high-speed, high-resolution AD conversion was given. Based on an investigation of current state-of-the-art converters, it was shown that a pipelined architecture is the best solution for our design goal. To provide a flexible and simple converter, each block in the pipeline resolves 1.5 bits of information. Two correction techniques were discussed and will be implemented to improve the accuracy of the pipelined ADC.

Chapter 3

Influence of errors on the converter's performance

This chapter deals with the influence of non-idealities in the basic blocks on the accuracy of the pipelined converter. First, a short introduction is given about static, quasi-static and dynamic errors. Then, a model of the basic block is given, including error sources. The influence of these error sources on the converter's overall performance is analyzed. Based on a specific performance goal, the constraints for each part of the ADC can be derived. In chapter 4, these constraints are translated to transistor-level requirements of the circuit. Furthermore, the influence of the applied post-correction method (discussed in section 2.2.2) on these constraints is investigated.

3.1 Classification of errors

Each possible error in the converter has a *source* and a certain influence on the *output* of the circuit. A source can be for example mismatch of components or jitter of the clock. The influence on the output is for example harmonic distortion or noise. Both the sources and the results at the output can be divided in three categories: static, quasi-static and dynamic errors. It is important to realize that, for example, a dynamic source can result in a static error at the output. In general, each of the three types of sources can result in each type of output error.

The most important goal of this chapter is to investigate the influence of the correction techniques, described in the previous chapter, on the accuracy requirements of each basic block. As these correction techniques are aimed at the correction of static and quasi-static output errors only, the model presented in this chapter does not take dynamic output errors into account. Even though the actual *source* can be either static, quasi-static or dynamic, the sources are modelled to be static in this chapter.

3.2 Error model of the basic block

To investigate the influence of errors on the converter's accuracy, the model of the basic block (figure 3.1) is extended with several error sources. The model of each component is described below. Only static and quasi-static deviations are taken into account, as it is hard to model dynamic errors without knowing the circuit. Moreover, an important issue of this chapter is to show the influence of the two exploited correction techniques, and they are aimed at (quasi-)static errors only.

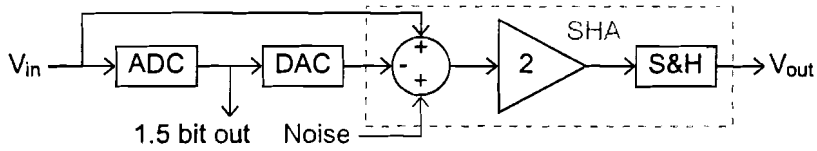


Figure 3.1: *Basic block for a pipelined converter.*

It is assumed that all error sources are *stochastic* and *independent* on other error sources. Systematic errors, as opposed to stochastic errors, can be prevented by proper design and layout. If required, the formulas derived in this chapter can be adapted to describe systematic errors as well.

The effect of clock jitter is not taken into account by the presented model, even though it can be a major limitation for the accuracy of the converter. The reason not to include this effect is that the clock jitter will have by far the most influence on the front-end sample-and-hold stage (placed before the first block of the pipeline) and the design of this stage is beyond the scope of this thesis.

Sub-ADC

The sub-ADC is a 1.5 bit AD converter, having two comparator levels (V_{ADC1} and V_{ADC2}). Due to transistor mismatch, these levels show a stochastic deviation δ_{ADC} from the ideal value:

$$\begin{cases} V_{ADC1}(i) &= -A_{max}\Big(R_{ADC} - \delta_{ADC1}(i)\Big) \quad , \quad \delta_{ADC1}(i) \sim N(0, \sigma^2_{ADC}) \\ V_{ADC2}(i) &= A_{max}\Big(R_{ADC} + \delta_{ADC2}(i)\Big) \quad , \quad \delta_{ADC2}(i) \sim N(0, \sigma^2_{ADC}) \end{cases} \quad [1] \tag{3.1}$$

δ_{ADC1} and δ_{ADC2} represent the stochastic deviation, using a normal distribution with mean 0 and variance σ^2_{ADC}. Due to the symmetry of the two comparator levels, the same variance is used for both levels. As the deviation is stochastic, each basic block will have different reference levels. Thus, an index i is used to indicate each individual basic block. It is assumed that the deviations of the reference levels are static, i.e. during operation of the ADC, the deviations remain constant.

[1] $\delta \sim N(\mu, \sigma^2)$: δ is a normally distributed random value with mean μ and variance σ^2.

Sub-DAC

The 1.5 bit sub-DAC has three reference levels. As was the case with the sub-ADC, these levels show a stochastic deviation due to mismatch of components:

$$\begin{cases} V_{DAC0}(i) &= -A_{max}\Big(R_{DAC} - \delta_{DAC0}(i)\Big) \quad , \quad \delta_{DAC0}(i) \sim N(0, \sigma^2_{DAC}) \\ V_{DAC1}(i) &= A_{max}\delta_{DAC1}(i) \quad , \quad \delta_{DAC1}(i) \sim N(0, \sigma^2_{DAC}) \\ V_{DAC2}(i) &= A_{max}\Big(R_{DAC} + \delta_{DAC2}(i)\Big) \quad , \quad \delta_{DAC2}(i) \sim N(0, \sigma^2_{DAC}) \end{cases} \tag{3.2}$$

The deviation of the ideal values is indicated with δ_{DAC}, and is unique for each reference level of each basic block. It is assumed that the deviations are constant during the time of operation. For simplicity, it is assumed that all DAC levels have the same variance.

SHA

The summing node, amplifier and sample-and-hold stage are combined in the sample-and-hold-amplifier (SHA). A white noise source is added to the SHA to represent thermal noise, produced by the circuit itself. The SHA is modelled as follows:

$$\begin{aligned} V_{out}(i) &= A(i)\Big(V_{in}(i) + V_{off}(i) - V_{DAC}(i) + V_n(i,t)\Big) \\ &- A_3\Big(V_{in}(i) + V_{off}(i) - V_{DAC}(i) + V_n(i,t)\Big)^3 \end{aligned} \tag{3.3}$$

with:

$$A(i) = 2(1 + \delta_A(i)) \quad , \quad \delta_A(i) \sim N(0, \sigma^2_A) \tag{3.4}$$

$$V_{off}(i) = A_{max}\delta_{off}(i) \quad , \quad \delta_{off}(i) \sim N(0, \sigma^2_{off}) \tag{3.5}$$

$$V_n(i,t) = A_{max}\delta_n(i,t) \quad , \quad \delta_n(i,t) \sim N(0, \sigma^2_n) \tag{3.6}$$

The gain A of the SHA (normally 2) and the input-referred offset voltage V_{off} are modelled with a random deviation. The white noise source V_n is modelled as a random value with a normal distribution, and is a function of time t. The non-linearity of the SHA is modelled with a third-order distortion component A_3. As the implemented circuit will be fully differential, it is assumed that the third-order component is the dominant source of non-linear distortion, therefore only this component is taken into account. A_3 is equal for all basic blocks, as our goal is to see the influence of harmonic distortion, and not to see the influence of deviations of the harmonic distortion.

3.3 Relations between error sources and performance

In this section, mathematical expressions are derived, showing the relation between each individual error source and the accuracy of the pipelined converter. This is done for a 1.5-bit per stage pipelined converter both without and with the post-correction algorithm employed. Secondly, simulation results are given to verify the derived relations and to show the difference between a converter without and a converter with post-correction.

3.3.1 Without post-correction

Here, the converter's accuracy as a function of the error sources is investigated, when post-correction is not applied. Though, the 1.5-bit redundancy is used. In order not to make the formulas too complex, the error sources are investigated one by one. The effect of a combination of errors applied at the same time is better to be simulated rather than calculated.

Noise

Suppose, we have a pipelined converter with k identical stages, together resolving N bits of information. According to the model of the previous section, each stage contains an independent noise source $V_n(i)$ with $0 \leq i < k$. All other components are assumed to be ideal.

To derive an upper bound for the amount of noise allowed for a certain accuracy, we use the rule that the average input referred thermal noise power is of the same magnitude as the quantization noise power according to N bits accuracy.

Each stage contains a noise source with standard deviation $A_{max}\sigma_n$ corresponding to an average noise power of $A_{max}^2\sigma_n^2$. The average input referred thermal noise power is the sum of all noise sources, corrected by the intermediate gain stages:

$$P_{tn} = \sum_{i=0}^{k-1} \frac{1}{2^{2i}} A_{max}^2\sigma_n^2 \approx \frac{4}{3} A_{max}^2\sigma_n^2 \tag{3.7}$$

The input range of the first stage is $[-A_{max}, A_{max}]$, for N bits accuracy this means:

$$\frac{1}{2}V_{lsb} = \frac{A_{max}}{2^N} \tag{3.8}$$

When the probability distribution of the quantization error is uniform, the average quantization noise power is:

$$P_{qn} = \int_{-\frac{1}{2}V_{lsb}}^{\frac{1}{2}V_{lsb}} \frac{1}{V_{lsb}} x^2 dx = \frac{1}{3}\left(\frac{A_{max}}{2^N}\right)^2 \tag{3.9}$$

Combining the equations for thermal and quantization noise power yields:

$$\begin{aligned} P_{tn} &\leq P_{qn} \\ \sigma_n &\leq 2^{-(N+1)} \end{aligned} \tag{3.10}$$

Deviations in the sub-ADC

Deviations of the comparator levels in the sub-ADC's have no influence on the converter's accuracy, as 1.5-bit redundancy is employed. The only limitation is that the deviations should not become too severe. According to section 2.2.1, full accuracy is guaranteed as long as condition 3.11 is satisfied.

$$R_{ADC} \leq \frac{1}{2} \quad \wedge \quad R_{DAC} \geq \frac{1}{2} \quad \wedge \quad R_{DAC} - R_{ADC} \leq \frac{1}{2} \tag{3.11}$$

When deviations of the sub-ADC and the sub-DAC are taken into account (equations 3.1 and 3.2) while the SHA is assumed to be ideal, then it can be guaranteed that the sub-ADC is not a limiting factor in the converter's accuracy when the following conditions are satisfied (instead of the conditions from 3.11):

$$\begin{aligned} -\frac{1}{2} + R_{ADC} + \delta_{DAC1} \leq \delta_{ADC1} \leq \frac{1}{2} + R_{ADC} - R_{DAC} + \delta_{DAC0} \\ -\frac{1}{2} - R_{ADC} + R_{DAC} + \delta_{DAC2} \leq \delta_{ADC2} \leq \frac{1}{2} - R_{ADC} + \delta_{DAC1} \end{aligned} \tag{3.12}$$

If σ_{DAC} is known, and one would like to satisfy these conditions as long as the deviations of the sub-ADC and sub-DAC are less than $3\sigma_{ADC}$ and $3\sigma_{DAC}$ respectively, the following constraint for the accuracy of the sub-ADC can be derived:

$$\sigma_{ADC} \leq \frac{1}{3} \min\left\{\frac{1}{2} + R_{ADC} - R_{DAC} - 3\sigma_{DAC}, \frac{1}{2} - R_{ADC} - 3\sigma_{DAC}\right\} \tag{3.13}$$

[2]

Deviations in the sub-DAC

Deviations of the reference voltages in the sub-DAC's are reflected immediately in the converter's accuracy. In the digital domain, the analog value is reconstructed according to equation 3.14, as described in section 2.2.1.

[2]This equation assumes that the SHA is ideal. Equation 3.35 shows the requirements when all possible errors are present at the same time.

$$V_{eq} = R_{DAC} A_{max} \sum_{i=0}^{k-1} 2^{-i} (D_{out}(i) - 1) \tag{3.14}$$

However, this reconstruction assumes that the sub-DAC's are ideal ($\delta_{DAC[c]}(i) = 0$ for $0 \leq c \leq 2$ and $0 \leq i < k$). The actual voltage, represented by the digital code equals:

$$V_{act} = A_{max} \sum_{i=0}^{k-1} 2^{-i} \Big((D_{out}(i) - 1) R_{DAC} + \delta_{DAC[D_{out}(i)]}(i) \Big) \tag{3.15}$$

Thus, the error V_{err} made due to deviations in the sub-DAC's is:

$$V_{err} = V_{act} - V_{eq} = A_{max} \sum_{i=0}^{k-1} 2^{-i} \delta_{DAC[D_{out}(i)]}(i) \tag{3.16}$$

Assuming that the deviations of the sub-DAC's are less than $3\sigma_{DAC}$ results in:

$$V_{err,max} \approx A_{max} 6 \sigma_{DAC} \tag{3.17}$$

The demand that V_{err} should be less than $\frac{1}{2} V_{lsb}$ yields an upper bound for σ_{DAC}:

$$\left.\begin{array}{rcl} \frac{1}{2} V_{lsb} & = & \frac{A_{max}}{2^N} \\ V_{err,max} & \approx & A_{max} 6 \sigma_{DAC} \end{array}\right\} \Rightarrow \sigma_{DAC} \leq \frac{1}{6} 2^{-N} \tag{3.18}$$

Offset in the SHA

As was the case with the deviations of the sub-DAC and the thermal noise source, the offset voltages of all SHA's can be replaced by a single input referred offset voltage $V_{off,in}$ with a normal distribution function according to:

$$V_{off,in} = A_{max} \delta_{off,in} \quad , \quad \delta_{off,in} \sim N(0, (2\sigma_{off})^2) \tag{3.19}$$

As the best-fit curve is used to determine the accuracy of the pipelined ADC, this additional offset has no influence at all on the accuracy. The only limitation is that the offset of each block should not result in exceeding the input range of the next stage. When all other components are ideal, this leads the following constraint:

$$|\delta_{off}| \leq \min \left\{ \frac{1}{2} + R_{ADC} - R_{DAC}, \frac{1}{2} - R_{ADC}, R_{DAC} - \frac{1}{2} \right\} \tag{3.20}$$

Using $3\sigma_{off}$ as a worst case estimation results in:

$$\sigma_{off} \leq \frac{1}{3}\min\left\{\frac{1}{2} + R_{ADC} - R_{DAC}, \frac{1}{2} - R_{ADC}, R_{DAC} - \frac{1}{2}\right\} \quad [3] \qquad (3.21)$$

Gain error in the SHA

The gain error of each SHA is modelled by δ_A in equation 3.3. When all other errors are neglected, this formula can be simplified to:

$$\begin{aligned} V_{out}(i) &= 2\Big(1 + \delta_A(i)\Big)\Big(V_{in}(i) - V_{DAC}(i)\Big) \\ &= 2\Big(V_{in}(i) - V_{DAC}(i)\Big) + 2\delta_A(i)\Big(V_{in}(i) - V_{DAC}(i)\Big) \end{aligned} \qquad (3.22)$$

The first term of 3.22 represents the ideal output, the second term indicates the deviation. The maximum absolute deviation is achieved for the maximum input signal. Ideally, the maximum input signal is bounded by:

$$\begin{aligned} |V_{in}(i) - V_{DAC}(i)| \leq A_{max}R_{max} \text{ , with} \\ R_{max} = \max\Big\{R_{ADC}, R_{DAC} - R_{ADC}, 1 - R_{DAC}\Big\} \end{aligned} \qquad (3.23)$$

which bounds the deviation to:

$$|V_{dev}(i)| \leq 2\delta_A(i)A_{max}R_{max} \qquad (3.24)$$

In theory, the total input referred error is a function of all the gain errors, made in each individual SHA. In practice, the total error can be estimated quite accurate by taking into account the gain error of the first stage only. This proposition can be made plausible by the following example: assume $R_{ADC} = \frac{1}{3}$ and $R_{DAC} = \frac{2}{3}$. The first stage achieves the maximum deviation if the input voltage equals $A_{max}R_{ADC}$ [4]. In that situation, the output voltage of the first stage equals approximately $2A_{max}R_{ADC} = A_{max}R_{DAC}$ (neglecting the influence of the gain-error). In that case, the output voltage of the second stage will be exactly around 0. Because of that, the influence of the gain-error in the second stage is much smaller than the maximum possible deviation according to equation 3.24, and it can be neglected compared to the deviation of the first stage. Likewise, the output voltages of all other stages will be around 0, and their deviations can be neglected as well.

[3] This equation assumes that the sub-ADC, sub-DAC and the gain of the SHA are ideal. Equation 3.35 shows the requirements when all possible errors are present at the same time.

[4] There are more possible input voltages where the maximum deviation occurs (for example $-A_{max}R_{ADC}$), but in all situations the same result is obtained.

To make the deviation of the first stage input-referred, it is necessary to divide by the gain of the stage (approximately 2). For the maximum deviation, the value of $3\sigma_A$ is used:

$$V_{dev,max} \approx 3\sigma_A A_{max} R_{max} \tag{3.25}$$

As before, the constraint that the maximum deviation should be less than $\frac{1}{2}V_{lsb}$ is used:

$$\sigma_A \leq \frac{1}{3R_{max}} 2^{-N} \tag{3.26}$$

Non-linear distortion in the SHA

As was the case with gain error in the SHA, the total error due to harmonic distortion can be estimated quite accurate by the distortion of the first SHA only. Assuming that all other components are ideal, the distortion of the first SHA is given by (derived from equation 3.3):

$$V_{hd}(i) = A_3\Big(V_{in}(i) - V_{DAC}(i)\Big)^3 \tag{3.27}$$

Maximum distortion is achieved when the maximum input signal is applied (equation 3.23). Making this distortion input-referred yields:

$$\begin{aligned} V_{hd,in,max} &= \frac{1}{2}A_3\Big(A_{max}R_{max}\Big)^3 \text{, with} \\ R_{max} &= \max\Big\{R_{ADC}, R_{DAC} - R_{ADC}, 1 - R_{DAC}\Big\} \end{aligned} \tag{3.28}$$

The maximum allowed distortion for N bits accuracy now becomes:

$$A_3 \leq \frac{2}{A_{max}^2 R_{max}^3} 2^{-N} \tag{3.29}$$

3.3.2 With post-correction

Now, the converter's accuracy as a function of the error sources is investigated, when post-correction (according to section 2.2.2) is applied. Also, 1.5-bit redundancy is used in each stage.

Noise

The post-correction algorithm has no influence on the thermal noise, produced by the SHA. Therefore, the same relation as before holds:

$$\sigma_n \leq 2^{-(N+1)} \tag{3.30}$$

Stochastic and linear deviations in the basic block

In contradiction to the system without post-correction, where the influence of each stochastic and linear error source in the basic block was discussed separately, now these errors are analyzed when all of them are present at the same time.

Into account are taken:

- deviations in the sub-ADC,
- deviations in the sub-DAC,
- offset in the SHA,
- gain error in the SHA.

The errors not taken into account are noise and non-linear distortion in the SHA.

According to section 2.2.2 the four types of error listed above have no influence on the converter's accuracy, as long as the output of each basic block remains in the valid range and enough redundant stages are added to the pipeline.

An example of a transfer function of a basic block is given in figure 3.2. Mathematically, this function is given by the following relations:

$$V_{out} = \begin{cases} A(V_{in} + V_{off} - V_{DAC0}) & \text{if} \quad V_{in} \leq V_{ADC1} \\ A(V_{in} + V_{off} - V_{DAC1}) & \text{if} \quad V_{ADC1} < V_{in} \leq V_{ADC2} \\ A(V_{in} + V_{off} - V_{DAC2}) & \text{if} \quad V_{ADC2} < V_{in} \end{cases} \tag{3.31}$$

To prevent that the allowed input range of the next stage is exceeded, the output voltage should be bounded by:

$$-A_{max} \leq V_{out} \leq A_{max} \tag{3.32}$$

Looking at figure 3.2, this means that the six corners (marked with no. 1 up to 6) should be in this range. Using the definitions from equations 3.1, 3.2, 3.4 and 3.5 this leads to the set of constraints:

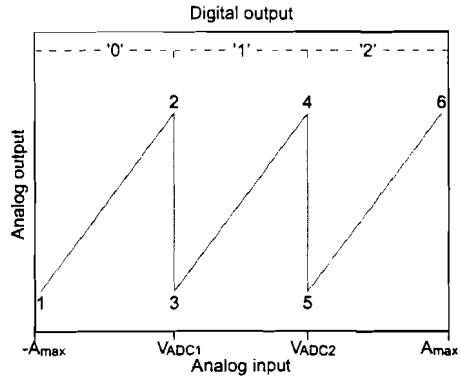


Figure 3.2: *Example of a transfer function of a basic block.*

$$\begin{cases} 2(1+\delta_A)\left(-\quad 1+\delta_{off}+(R_{DAC}-\delta_{DAC0})\right) \geq -1 \\ 2(1+\delta_A)\left(-(R_{ADC}-\delta_{ADC1})+\delta_{off}+(R_{DAC}-\delta_{DAC0})\right) \leq 1 \\ 2(1+\delta_A)\left(-(R_{ADC}-\delta_{ADC1})+\delta_{off}-\delta_{DAC1}\right) \geq -1 \\ 2(1+\delta_A)\left((R_{ADC}+\delta_{ADC2})+\delta_{off}-\delta_{DAC1}\right) \leq 1 \\ 2(1+\delta_A)\left((R_{ADC}+\delta_{ADC2})+\delta_{off}-(R_{DAC}+\delta_{DAC2})\right) \geq -1 \\ 2(1+\delta_A)\left(1+\delta_{off}-(R_{DAC}+\delta_{DAC2})\right) \leq 1 \end{cases} \tag{3.33}$$

The stochastic deviations (δ_{ADC}, δ_{DAC}, δ_{off} and δ_A) are limited by 3σ with great certainty (99.74%), so it is assumed that the following conditions are valid:

$$\begin{aligned} -3\sigma_{ADC} \leq \delta_{ADC} &\leq 3\sigma_{ADC} \\ -3\sigma_{DAC} \leq \delta_{DAC} &\leq 3\sigma_{DAC} \\ -3\sigma_{off} \leq \delta_{off} &\leq 3\sigma_{off} \\ -3\sigma_A \leq \delta_A &\leq 3\sigma_A \end{aligned} \tag{3.34}$$

Using this assumption, the set of constraints can be simplified to:

$$\begin{cases} (1+3\sigma_A)(1 \quad -R_{DAC} \quad +3\sigma_{off}+3\sigma_{DAC}) \leq \frac{1}{2} \\ (1+3\sigma_A)(R_{DAC} \quad -R_{ADC} \quad +3\sigma_{ADC} \quad +3\sigma_{off}+3\sigma_{DAC}) \leq \frac{1}{2} \\ (1+3\sigma_A)(R_{ADC} \quad +3\sigma_{ADC} \quad +3\sigma_{off}+3\sigma_{DAC}) \leq \frac{1}{2} \end{cases} \tag{3.35}$$

A derivation of the number of stages k required to achieve a certain accuracy N is not made. Based on an estimation of the statistical deviations of the circuit and the required bits N, simulations are performed to decide on the number of stages k. An example will be given in section 3.4.

Non-linear distortion in the SHA

Although the post-correction algorithm is not aimed at improving the linearity of the pipelined ADC, it will be shown that the linearity improves after application of post-correction. As before, the distortion of the first stage only is considered.

Figure 3.3 (left) shows an example of an ideal transfer function, and an example of a transfer function including 3^{rd} order harmonic distortion. In a system without post-correction, it is assumed that the transfer function equals the ideal curve, thus the maximum deviation occurs at the highest absolute output-level. In case of a system with post-correction (right picture), the correction algorithm reconstructs the transfer function such that the actual curve and its reconstruction exactly match around the transitions of the transfer curve. The picture shows that in this situation, the maximum distortion occurs not at the maximum absolute output-level, but somewhere in the middle.

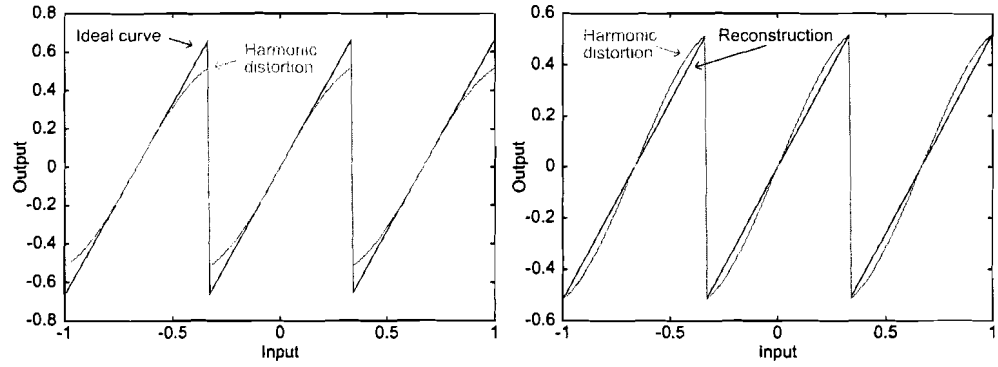


Figure 3.3: *Reconstruction of a transfer function with harmonic distortion in case of a system without post-correction (left) and with post-correction (right).*

Figure 3.4 shows the error due to harmonic distortion in case of a system without and a system with post-correction. Obviously, the same amount of distortion in the SHA has less influence when post-correction is applied.

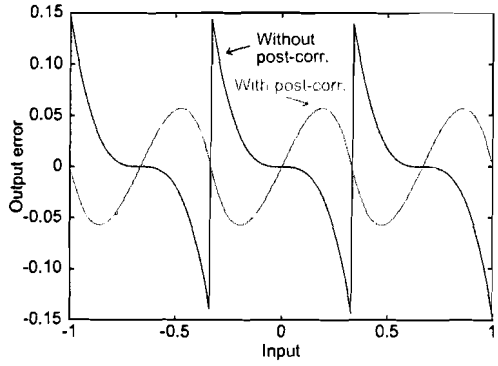


Figure 3.4: *Introduced deviation due to non-linearity in the SHA for systems without and with post-correction applied.*

A simple analysis shows the improvement after application of the post-correction algorithm. First, consider the system without post-correction. The actual curve is given by:

$$y_{act} = 2x - a_3 x^3 \tag{3.36}$$

For simplicity, variables x and y_{act} are used instead of the actual names of the signals. Moreover, only the middle part of the transfer function is dealt with, as the other two parts show exactly the same behavior.

The converter uses the reconstruction function:

$$y_1 = 2x \tag{3.37}$$

Thus, the resulting *input referred* deviation is:

$$y_{dev1} = \frac{1}{2}\left(y_{act} - y_1\right) = -\frac{1}{2}a_3 x^3 \tag{3.38}$$

If the maximum input signal is m, the maximum absolute deviation is:

$$y_{dev1,max} = \frac{1}{2}a_3 m^3 \tag{3.39}$$

In case of a system with post-correction, the reconstruction function equals:

$$y_2 = bx \tag{3.40}$$

b is chosen such that for the maximum input m, y_2 equals y_{act}:

$$y_2(m) = y_{act}(m) \Rightarrow bm = 2m - a_3 m^3 \Rightarrow b = 2 - a_3 m^2 \tag{3.41}$$

The input referred deviation now becomes:

$$y_{dev2} = \frac{1}{2}\left(y_{act} - y_2\right) = \frac{1}{2}a_3 m^2 x - \frac{1}{2}a_3 x^3 \tag{3.42}$$

The maximum deviation is now achieved for:

$$\frac{\partial y_{dev2}}{\partial x} = 0 \Rightarrow x = \frac{m}{\sqrt{3}} \Rightarrow y_{dev2,max} = \frac{1}{9}\sqrt{3}a_3 m^3 \tag{3.43}$$

The improvement of accuracy after application of the post-correction algorithm is determined by the ratio between $y_{dev1,max}$ and $y_{dev2,max}$:

$$\log_2\left(\frac{y_{dev1,max}}{y_{dev2,max}}\right) = \log_2\left(\frac{3}{2}\sqrt{3}\right) \approx 1.38 \text{ bit} \tag{3.44}$$

Combined with equation 3.29, the maximum harmonic distortion for a pipelined converter with post-correction is limited by:

$$A_3 \leq \frac{3\sqrt{3}}{A_{max}^2 R_{max}^3} 2^{-N} \tag{3.45}$$

3.3.3 Simulation results

In this section, simulation results are given, showing the accuracy of a pipelined converter with and without post-correction. Each error source is investigated separately.

The simulations are performed using Matlab. The stage model from section 3.2 (including all error sources) is used to compose a pipelined converter. The errors in each cell are chosen randomly, based on a user specified standard deviation. Due to these random deviations, each simulation will produce different results. The Matlab model includes a converter without and one with post-correction. The accuracy of a converter is estimated by the software by calculating the complete transfer characteristic of the pipelined converter first. Then, the best-fit linear curve is calculated. The maximum deviation between the best-fit curve and the actual transfer function is equated with $\frac{1}{2}V_{lsb}$, yielding an estimation of the accuracy in bits of the converter.

Some parameters were fixed during all simulations, namely:

$$\begin{aligned} A_{max} &= 0.8 \\ R_{ADC} &= \tfrac{1}{3} \\ R_{DAC} &= \tfrac{2}{3} \\ k &= 16 \end{aligned} \tag{3.46}$$

The value for A_{max} equals the value that will be used in the actual analog design. Ratios R_{ADC} and R_{DAC} are chosen such that maximum possible error correction is achieved (the given values maximize the distance to the constraints given in equation 2.20). A converter with 16 stages is simulated, as this is enough to show the influence of the correction algorithms.

Noise

Figure 3.5 shows the achieved accuracy in bits as a function of the amount of noise, produced in each basic block. Shown are the model (from section 3.3.1) and simulation results for a converter without and a converter with post-correction. As expected, the performance of both systems is the same. The reason that the model shows a slightly better performance than the simulations is due to the fact that the

model is based on the average noise power, whereas the simulations take the worst case situation into account. For small amounts of noise (around $\sigma_n = 1 \cdot 10^{-6}$), the performance is limited by the number of stages instead of the thermal noise.

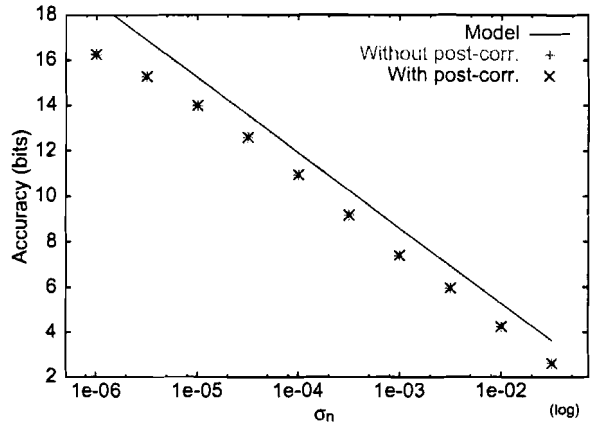


Figure 3.5: *Accuracy as a function of the amount of noise. Both model and simulation results are shown.*

Deviations in the sub-ADC

In figure 3.6, one can see the simulated performance in case of deviations of the comparator levels in the sub-ADC. According to the model (formula 3.35), the accuracy is not affected as long as $\sigma_{ADC} < 0.056$. For larger deviations, the chance of losing accuracy increases. This effect is visible in the figure. Nevertheless, it is hard to draw conclusions as the errors are stochastic: even for $\sigma_{ADC} \gg 0.056$, 16 bits accuracy can be obtained, and even for $\sigma_{ADC} \ll 0.056$ there is a chance of obtaining less than 16 bits accuracy. To be able to express the yield for a certain accuracy as a function of σ_{ADC}, much more simulations should have been performed.

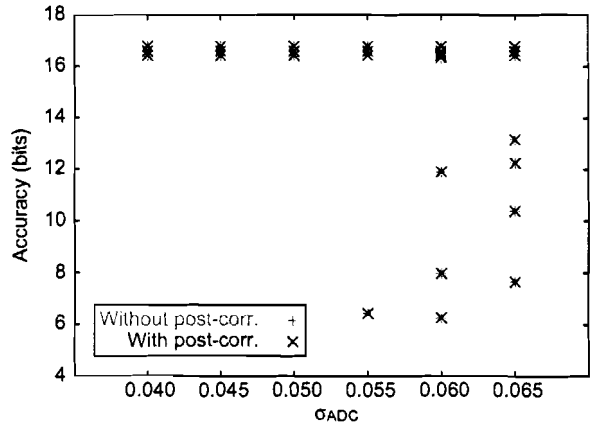


Figure 3.6: *Accuracy as a function of the amount of deviation of the comparator levels in the sub-ADC.*

Deviations in the sub-DAC

Figure 3.7 shows the accuracy of a converter without post-correction as a function of the deviation of the output levels of the sub-DAC. The simulation is in agreement with the trend described by the model (formula 3.18). Also shown are the simulation results for a converter with post-correction. Its performance is independent from the severeness of the deviation. The small drop in performance is because the required amount of redundant stages increases as the deviations increase. When for example 18 stages were used instead of 16, the performance will improve with 2 bits as well.

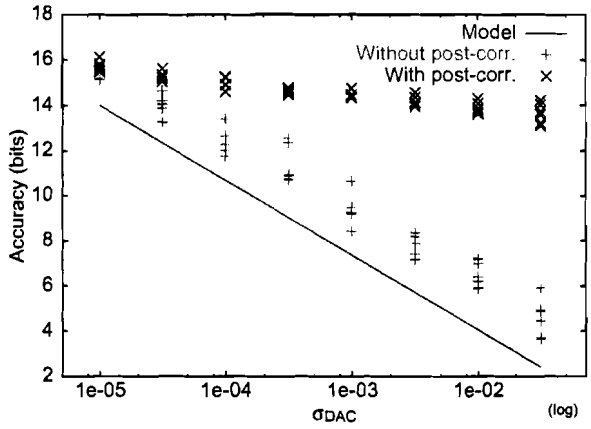


Figure 3.7: *Accuracy as a function of the amount of deviation of the output levels of the sub-DAC.*

Offset in the SHA

In figure 3.8 the accuracy as a function of the offset in the SHA is visible. As expected by the model (formula 3.35) the accuracy is not affected, until the deviations result in exceeding the input range of a basic block (this occurs for $\sigma_{off} > 0.056$).

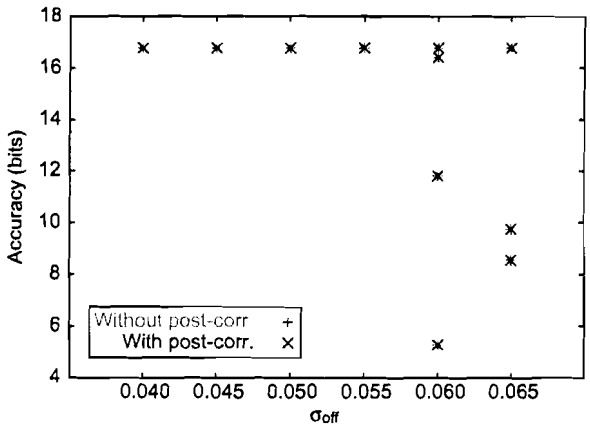


Figure 3.8: *Accuracy as a function of the amount of offset in the SHA.*

Gain error in the SHA

Figure 3.9 shows the accuracy of a converter without post-correction as a function of the deviation of the gain of the SHA. The simulation is in agreement with the model (formula 3.26). Also shown are the simulation results for a converter with post-correction. Its performance is independent from the severeness of the deviation. The small drop in performance is because the required amount of redundant stages increases as the gain error increases.

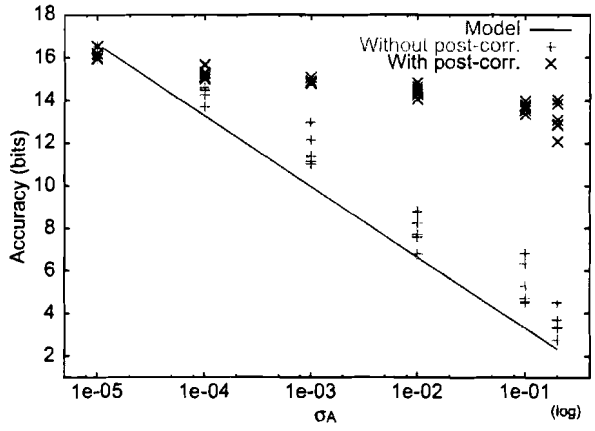


Figure 3.9: *Accuracy as a function of the amount of gain error of the SHA.*

Non-linear distortion in the SHA

Finally, the influence of harmonic distortion in the SHA was simulated. Figure 3.10 shows the models and simulation results for converters without and with post-correction.

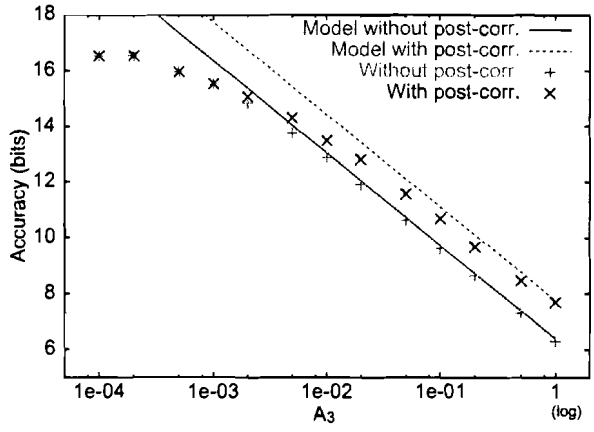


Figure 3.10: *Accuracy as a function of the amount of harmonic distortion in the SHA. Models and simulation results are shown for converters without and with post-correction.*

As expected, the performance after application of the post-correction algorithm improves. However, the expected improvement of 1.38 bit is achieved only when the distortion is rather severe. For small amounts of distortion, the accuracy is limited by the number of stages in the pipeline.

3.4 Design example

In this section, a design example is worked out, based on the theory presented in this chapter. Based on a specific target, the requirements for sub-ADC, sub-DAC and SHA are derived. As before we use:

$$\begin{array}{rcl} A_{max} & = & 0.8 \\ R_{ADC} & = & \frac{1}{3} \\ R_{DAC} & = & \frac{2}{3} \end{array} \tag{3.47}$$

The design target is to achieve 12 bits of accuracy, when all errors discussed before are present at the same time. First an uncorrected converter is used to achieve this target, then a converter with post-correction is used.

3.4.1 Without post-correction

In total, there are six parameters important for the achieved accuracy: σ_n, σ_{ADC}, σ_{DAC}, σ_{off}, σ_A and A_3. Using a system without post-correction, equations 3.10, 3.18, 3.26, 3.29 and 3.35 must be satisfied.

From the first four equations, one can derive easily:

$$\begin{array}{rcl} \sigma_n & = & 9.25 \cdot 10^{-5} \\ \sigma_{DAC} & = & 3.0 \cdot 10^{-5} \\ \sigma_A & = & 1.8 \cdot 10^{-4} \\ A_3 & = & 1.5 \cdot 10^{-2} \end{array} \tag{3.48}$$

With these values, the converter is actually designed for 12.4 bits. However, it is known in advance that the combination of errors has more influence than each separate error.

The last requirement is to satisfy equation 3.35, yielding:

$$\sigma_{ADC} + \sigma_{off} \leq 5.5 \cdot 10^{-2} \tag{3.49}$$

Any basic block, satisfying the last condition, is acceptable. To be able to perform simulations, an arbitrary choice for σ_{ADC} and σ_{off} was made:

$$\begin{aligned} \sigma_{ADC} &= 2.5 \cdot 10^{-2} \\ \sigma_{off} &= 2.5 \cdot 10^{-2} \end{aligned} \tag{3.50}$$

Using Matlab, a pipelined converter without post-correction algorithm was created, using the stochastic deviations stated above. σ_n was excluded from the simulation, otherwise the thermal noise would make the other deviations imperceptible. The number of stages k was varied from 11 up to 15. As the simulation is statistical, each type was simulated 100 times. Figure 3.11 shows the simulation results.

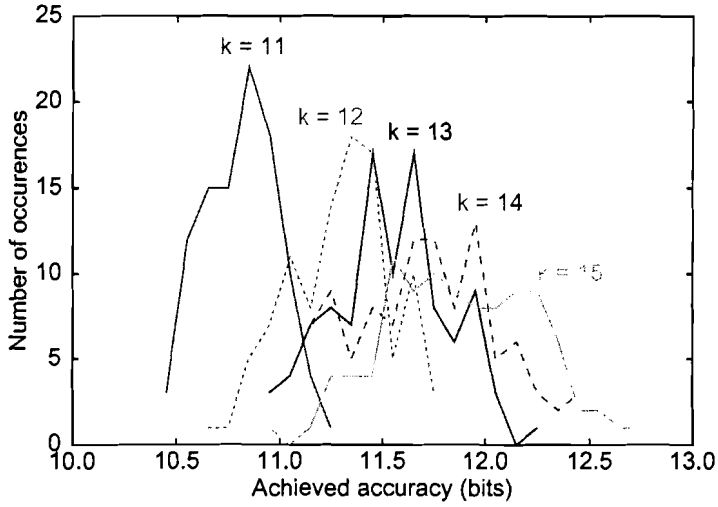


Figure 3.11: *Simulated accuracy of pipelined converters without post-correction.*

For $k < 12$, the accuracy of the converter is limited by the amount of stages, and not by the inaccuracy of the stages. For $k > 12$, the accuracy is almost constant, which means that the accuracy is limited by the deviations in the basic blocks.

The achieved accuracy, according to the simulations, is roughly between 11 and 12 bits, instead of the expected 12 bits. Although each separate error source (DAC deviation, harmonic distortion, etc.) was designed for 12.4 bits, the summation of these errors will reduce the accuracy somewhat. A new iteration with slightly smaller values of the error sources can be done to achieve the desired accuracy of 12 bits. If 11 bits is acceptable, the simulations show that a 13-stage pipeline is good enough.

3.4.2 With post-correction

Now, a converter with correction algorithm is used to achieve the same design target. The conditions to be fulfilled are given by the equations 3.30, 3.45 and 3.35.

Designing for 12.4 bit again, the first two equations yield:

$$\begin{aligned} \sigma_n &= 9.25 \cdot 10^{-5} \\ A_3 &= 3 \cdot 10^{-2} \end{aligned} \tag{3.51}$$

For the other four parameters (σ_{ADC}, σ_{DAC}, σ_{off} and σ_A) only one condition exists (equation 3.35), giving a lot of design freedom. Arbitrarily, the following combination was chosen:

$$\begin{aligned} \sigma_{ADC} &= 3.0 \cdot 10^{-2} \\ \sigma_{DAC} &= 1.0 \cdot 10^{-2} \\ \sigma_{off} &= 1.0 \cdot 10^{-2} \\ \sigma_A &= 1.0 \cdot 10^{-2} \end{aligned} \tag{3.52}$$

Again, simulations were performed. The number of stages was varied from 13 to 17. The results from the statistical simulations are given in figure 3.12. Thermal noise was not included in this simulation as it would conceal the influence of the other deviations.

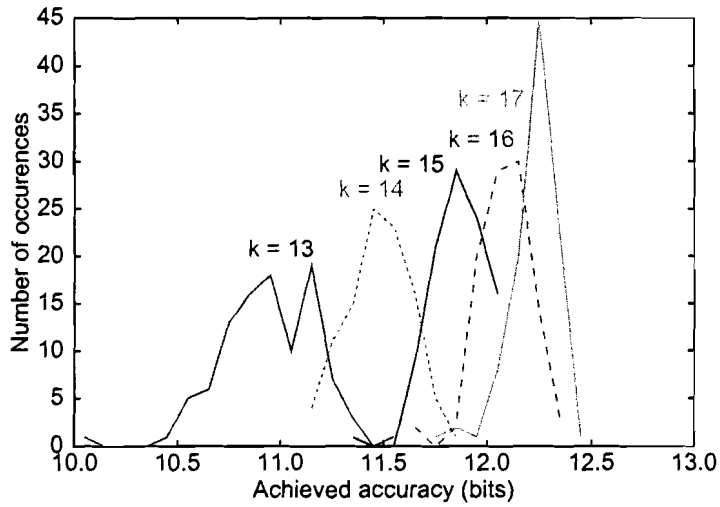


Figure 3.12: *Simulated accuracy of pipelined converters with post-correction.*

In this figure, it can be seen that the amount of stages is important for the achieved accuracy. For $k < 16$ the performance is limited by the amount of stages, for $k \geq 16$ the performance hardly increases, as it becomes limited by the errors in the basic block. From the figure it follows that a pipeline of 16 stages is optimal, as additional stages hardly improve the performance. Also it can be seen that the converter achieves 12 bits accuracy.

3.5 Conclusions

In this chapter, the basic block was extended with static and quasi-static error sources, defined on system level. Formulas were derived to show the relation be-

tween these error sources and the achievable accuracy of the pipelined ADC. Simulations were performed to verify the derived formulas. Finally, it was shown that the accuracy requirements for each component (sub-ADC, sub-DAC, SHA) can be derived easily from the presented theory. A design example aimed at 12 bit accuracy showed that the post-correction algorithm greatly reduces the requirements for the sub-DAC and the SHA. Moreover, additional freedom is given to the designer as the post-correction algorithm reduces the amount of conditions to be fulfilled.

Chapter 4

Analog design

This chapter describes the transistor-level design of the most critical components in the basic block. Starting from the design goal, design constraints are extracted and translated to physical properties of the circuit. Simulations were performed to verify the achieved performance.

4.1 Design goal

As stated before, the analog part of the ADC is composed of a dedicated sample-and-hold stage followed by a pipeline of identical basic blocks. The goal of this project is the design of the basic block; the additional sample-and-hold stage is not dealt with. The design of the basic block has to meet the following criteria:

- accurate enough to achieve 12-bit resolution, under the assumption that 1.5-bit redundancy and post-correction are applied,
- optimize the design for high-speed,
- maximum simplicity of the circuitry,
- design in UMC 0.25μm process,
- functional for power supplies in the range 2.25 to 2.75 V,
- functional for the temperature range 0 to 85 °C.

The basic block contains three major parts: sub-ADC, sub-DAC and SHA. As it is very likely that the SHA is the most difficult part to design, the emphasis is put on this part. The sub-ADC and sub-DAC were modelled but not implemented.

4.2 Derivation of design constraints

Based on the theory presented in chapter 3, accuracy requirements for each component in the basic block can be derived. As the design goal coincides with the design example in section 3.4.2, no new derivations are necessary. For completeness, the results from that example are listed here again:

- to allow maximum correction (derived from equation 2.20):

$$R_{ADC} = \frac{1}{3} \text{ and } R_{DAC} = \frac{2}{3} \tag{4.1}$$

- maximum allowed thermal noise level:

$$\sigma_n \leq 9.25 \cdot 10^{-5} \tag{4.2}$$

- maximum allowed harmonic distortion:

$$A_3 \leq 3 \cdot 10^{-2} \tag{4.3}$$

- to remain in the valid dynamic range:

$$\left(1 + 3\sigma_A\right)\left(\frac{1}{3} + 3(\sigma_{ADC} + \sigma_{DAC} + \sigma_{off})\right) \leq \frac{1}{2} \tag{4.4}$$

- to provide enough redundancy:

$$k = 16 \tag{4.5}$$

During the design of each component of the basic block, these constraints will be used to derive requirements for some critical parts like transistors or capacitors.

4.3 Selection of the analog architecture

Besides the fulfillment of the design constraints described above, an important issue is to select a suitable architecture for the basic block. Based on a literature research, a proper architecture was selected.

4.3.1 Literature research

Important issues for the analog design are: low power supply, simplicity, robustness, high speed and good signal-to-noise ratio. An investigation was performed to search for suitable designs for either a complete SHA or sub-circuits that can be used for a SHA.

The resulting documents can be classified in three groups, and will be discussed one by one:

- open loop, voltage-mode S&H circuits,

- open loop, current-mode S&H circuits,

- operational amplifiers that can be used for closed loop S&H circuits.

Open loop, voltage-mode S&H circuits

References [8] and [9] consider voltage-domain S&H circuits with an open loop structure. Feed-forward compensation is used to compensate for non-linearities. These examples achieve high speed (at least 150 MS/s), but the accuracy level (noise and distortion) is not good enough for our situation. Adapting the design for a higher accuracy is likely to result in a reduced speed. No information is given about mismatch simulations, which can give additional problems in an open-loop structure. Moreover, the circuits do not have a gain-by-two stage, as needed in a pipelined structure. Also, the complexity of the circuit is not attractive as a pipeline contains many SHA's.

Open loop, current-mode S&H circuits

In [10], [11], [12] and [13] open loop, current-mode sample-and-hold structures are considered. The major problem of these designs is the limited speed (20 to 40 MS/s). It is said that the linearity is good enough for 10-bit accuracy, however, no information about the influence of mismatch is given. These circuits do not provide a gain-by-two function either, so the design has to be adapted.

Operational amplifiers

Operational amplifiers (opamps) can be used in combination with a switched capacitor network to create a SHA. Looking at high-speed CMOS opamps, three different architectures were found. In [6], a single stage, telescopic opamp is used, achieving a gain-bandwidth-product (GBP) of 100 MHz. However, using a telescopic opamp at a low power supply reduces the dynamic range and makes it difficult to achieve a good signal-to-noise ratio (SNR). [14] and [15] both use a two-stage opamp. The first stage is a low gain, wide band preamplifier. The second stage is a cascode stage. A high GBP is achieved (450 MHz) and the circuit can operate at a low power supply. Finally, [16], [17], [18] and [19] present comparable two-stage opamps. In all designs, the first stage is a simple differential pair, followed by a folded-cascode output stage. Some designs use a gain boosting technique in the output stage or a pole-zero cancellation technique to increase the bandwidth. Also, the designs are different with respect to the common-mode feedback circuit. The circuits work at low power supplies and achieve GBP's in the order of 400 to 800 MHz.

4.3.2 Selection

Based on the literature investigation, calculations and circuit simulations, it was decided to use a closed-loop SHA based on an operational amplifier. Although it is not possible to justify this choice in a few sentences, we can make it plausible by the following statements about the open-loop systems:

- the available open-loop systems are too slow and don't have a gain-by-two,
- high accuracy, robustness, reliability and high yield are hard to achieve with an open-loop system, as the influences of deviations and mismatch are not corrected by feedback,
- the available circuits for open-loop systems are relatively complex,
- none of the current high-speed AD converters uses an open-loop SHA.

As none of the available open-loop systems comes close to our target, while the closed-loop systems clearly provide a higher speed-accuracy performance, the latter system is selected. To be more exact, the design from [16] is used as a starting point, as this is the most simple circuit, it achieves a high GBP and it has a large dynamic range even at low power supplies.

4.4 Design of the analog components

Before going into design details of each component, a detailed version of the basic block is presented to define all signals between the sub-components.

4.4.1 Basic block

A diagram of the basic block, showing all physical connections, is given in figure 4.1.

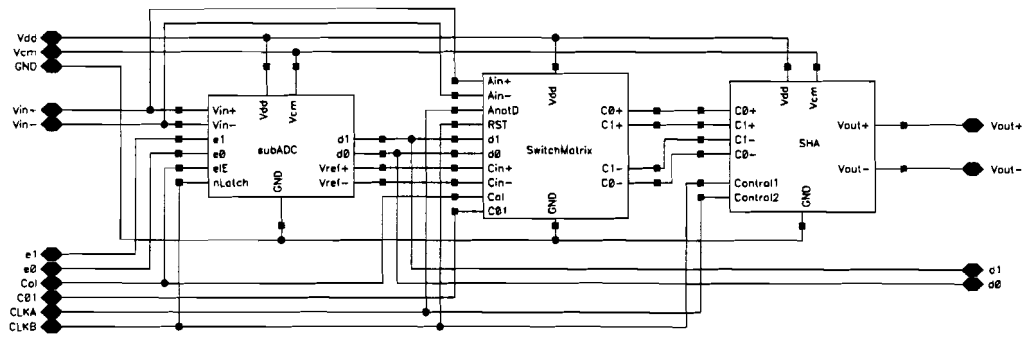


Figure 4.1: *Basic block including all external and internal connections.*

The **sub-ADC** contains the 1.5-bit coarse quantizer, the **SwitchMatrix** contains the 1.5-bit sub-DAC, some digital logic and a matrix of switches to connect the correct signals to the **SHA**. The **SHA** is realized by a switched capacitor network and an opamp. These three sub-items will be discussed in following sections.

The function of the pins of the basic block will be described now. V_{dd} and GND are the power supply connections. As most parts of the circuit are implemented in a differential structure, there are two input terminals (V_{in+} and V_{in-}) and a common mode reference level V_{cm}. The actual input voltage is $V_{in} = V_{in+} - V_{in-}$. V_{out+} and V_{out-} are the differential output voltage of the basic block. The digital output signals d1 and d0 are the sub-ADC output. Two clock signals (CLKA and CLKB) are required for the internal timing. The clock signals determine whether the circuit is in *sample* or in *hold* mode. In the *sample* period, the block samples a new input signal. The output of the cell is not valid. During the *hold* phase, the block is disconnected from the analog input, and the output settles to the new value.

To make digital post-correction according to section 2.2.2 possible, digital inputs e1, e0, Cal and C01 are added to the circuit. With Cal, the basic block can be set to calibration-mode. In that case e1 and e0 are used as input signals for the sub-DAC and C01 determines which of the two reference levels from the sub-ADC is provided to the SHA.

Figure 4.2 shows how basic blocks are connected in a pipelined converter. Note that, according to previous definitions (section 2.1.1), a dedicated sample-and-hold stage is placed in front of the first block of the pipeline (not included in the picture). The odd stages in the pipeline have different clock signals than the even stages: the clock signals of the even stages are delayed by $0.5T_s$. So, while the odd stages are in the *sample* phase, the even stages are in *hold* phase and vice versa.

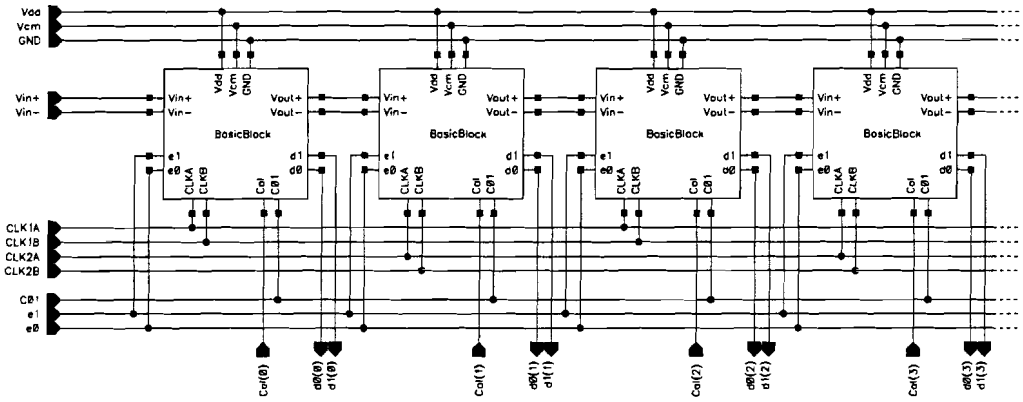


Figure 4.2: *Example of a pipeline of basic blocks.*

Timing of the clock signals

A more detailed view of the clock signals is given in figure 4.3. The odd stages in the pipeline use clock signals CLK1A and CLK1B, the even stages use CLK2A and CLK2B.

The latter two signals have the same shape as the first two signals, but are delayed by $0.5T_s$. In the *sample* phase of CLK1A, the input of the SHA is connected to the analog input of the basic block. The actual sample moment takes place when CLK1B goes low. At that moment, a switch is turned off, and the SHA holds the value of that moment. Also, the code produced by the sub-ADC is latched. After delay τ, the *hold* phase becomes active. From this moment on, the input of the SHA is connected to the sub-DAC. The sub-DAC produces an output voltage according to the digital code that was produced by the sub-ADC in the preceding *sample* phase. The SHA settles to the steady state value (ideally $V_{out} = 2(V_{in} - V_{DAC})$). Just before the end of the *hold* phase, two events take place. First, CLK2B goes low and the subsequent stage samples the output voltage of the SHA. Secondly, CLK1B goes high while CLK1A is still low. This results in resetting the SHA to a predefined state. Due to this reset, each new sample is independent on the previous state of the SHA.

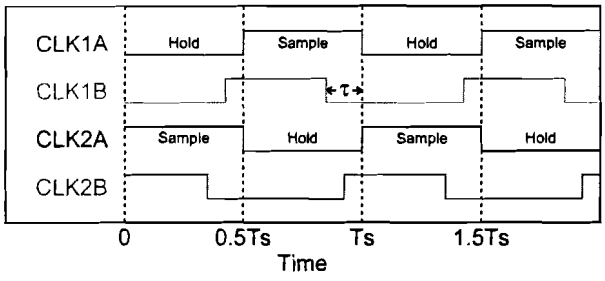


Figure 4.3: *Relative timing of the clock signals, T_s is a complete sample period. For clarity, the length of time interval τ is exaggerated.*

4.4.2 S&H amplifier

The sample-and-hold amplifier is implemented with an operational amplifier and a switched capacitor network (figure 4.4).

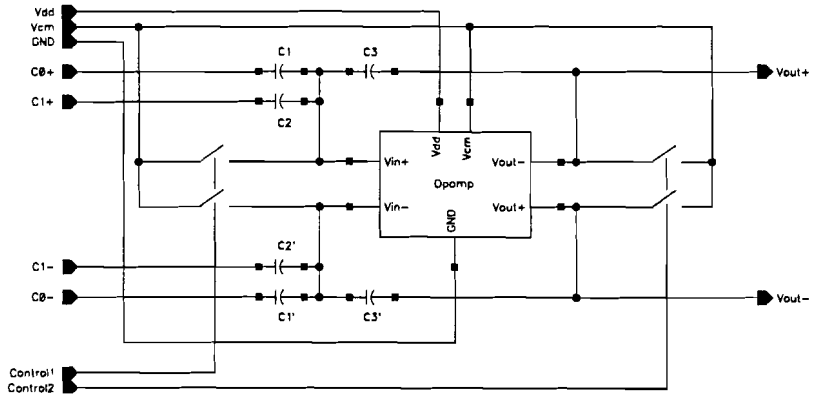


Figure 4.4: *Implementation of the SHA.*

This circuit only contains the switches resetting C_3 and C_3'. The switches selecting

either the analog or the digital input are placed in the **SwitchMatrix**. The size of all capacitors equals C_{std}. Figure 4.5 shows the operation of the SHA in the sample phase before and after the sample moment and in the hold phase.

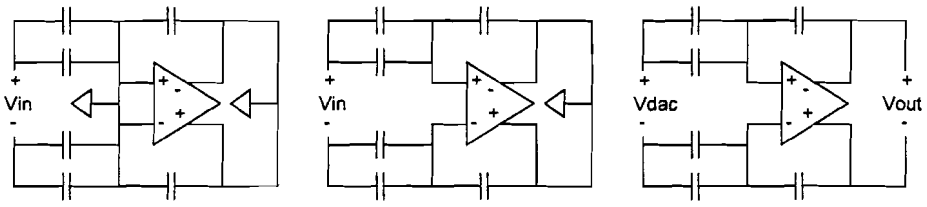


Figure 4.5: *SHA in the sample phase before (left) and after (middle) the actual sample moment, and in the hold phase (right).*

During the first part of the sample phase, the analog input voltage of the basic block is connected to the four input capacitors of the SHA by a set of switches. Also, all switches in the SHA are closed. The total charge stored in the four capacitors C_1, C_1', C_2 and C_2' together equals:

$$Q_1 = 2V_{in}C_{std} \tag{4.6}$$

After the sample moment, the two switches at the input of the opamp are opened by control signal `Control1`. From that moment on, charge can be transferred between the individual capacitors, but the total charge remains constant. In the hold phase, `Control2` also goes low, and the output of the sub-DAC is connected to the SHA. The opamp controls its input to the common-mode level, thus the charge in the four input capacitors settles to:

$$Q_2 = 2V_{DAC}C_{std} \tag{4.7}$$

Neglecting parasitic capacitances, the charge $Q_1 - Q_2$ is transferred completely to capacitors C_3 and C_3', resulting in the output voltage of the SHA:

$$V_{out} = \frac{Q_1 - Q_2}{C_{std}} = 2(V_{in} - V_{DAC}) \tag{4.8}$$

In practise, parasitic capacitances (of the opamp, switches, etc.) have influence on this transfer function. Possible influences are gain error, offset and non-linear distortion. As in our situation digital post-correction is employed, these imperfections are not so critical. Therefore, the parasitic influences are neglected at this moment. The simulation results, presented at the end of this chapter, include the parasitic effects.

Noise

The switches and the opamp produce noise at the output of the SHA. This noise is composed of two components:

- continuous-time noise: during the *hold* phase, noise produced by the switches in the switch matrix is instantaneously visible at the output of the SHA,
- sampled noise: at the *sample moment*, noise is sampled on the sampling capacitors. During the *hold* phase, this noise will become visible at the output of the SHA as well.

In this section, the sampled noise is calculated and compared to the design constraint for noise. The continuous-time noise is dealt with during the discussion of the simulation results in section 4.5.

The amount of sampled noise is determined by the size of the capacitors in the SHA. All switches in the SHA and in the **SwitchMatrix** have a certain on-resistance R_{sw}, and produce thermal noise. At the sample moment, a certain amount of this thermal noise is sampled as charge on the capacitors. From [20], it is known that the noise, stored on each of the six capacitors, has a variance of:

$$\sigma_Q^2 = kTC_{std} \quad [1] \tag{4.9}$$

In the hold phase, the sampled noise is transferred to C_3 and C_3', resulting in an output noise voltage with variance:

$$\sigma_{V_{n,out}}^2 = \frac{6\sigma_Q^2}{C_{std}^2} = \frac{6kT}{C_{std}} \tag{4.10}$$

The design constraint for noise was given in section 4.2:

$$\sigma_n \leq 9.25 \cdot 10^{-5} \tag{4.11}$$

This constraint is valid for input-referred noise, and relative to the maximum signal amplitude A_{max} (equation 3.6). The constraint for output noise power is derived from this equation by multiplying by 2 (the gain of the SHA), taking equation 3.6 into account and squaring the result to obtain the noise *power*. Combining the new constraint with equation 4.10 yields:

$$\sigma_{V_{n,out}}^2 = \frac{6kT}{C_{std}} \leq \left(2A_{max}\sigma_n\right)^2 = 3.4 \cdot 10^{-8} A_{max}^2 \tag{4.12}$$

[1] k is the Boltzmann constant: $1.38 \cdot 10^{-23}$ J/K, T is the temperature in K.

The maximum signal amplitude A_{max} is limited by the opamp. In the following section, the design of the opamp is discussed, and there $A_{max} = 0.8$ is achieved. Using this value and $T = 300$ K yields a lower bound for the capacitor size:

$$C_{std} = 1.2 \text{ pF} \tag{4.13}$$

4.4.3 Operational amplifier

The operational amplifier, used in the SHA, is based on the designs from [16] and [17]. In figure 4.6, the circuit is given. Strictly speaking, this is an operational transconductance amplifier. Nevertheless, in this thesis it will be indicated by *opamp*. For simplicity the common-mode feedback is omitted in this picture.

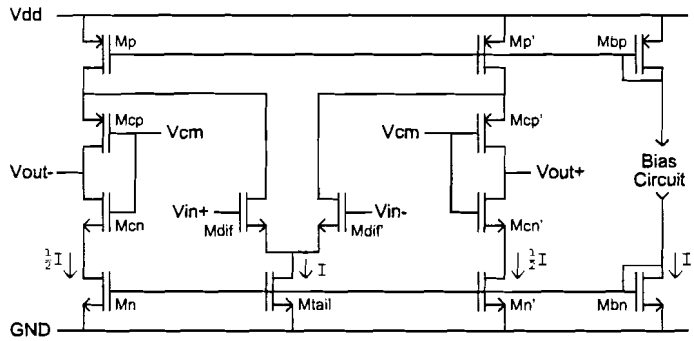


Figure 4.6: *Folded-cascode opamp, used in the SHA. The bias current in each branch is indicated.*

The differential opamp is composed of two stages: a differential pair as input stage and a folded-cascode output stage. The input stage is modelled with its transconductance g_m and an input capacitance C_{in}. The output stage is modelled by an output resistance R_{out} and an output capacitance C_{out}. Figure 4.7 shows the model of the opamp, together with the external feedback from the SHA in hold phase.

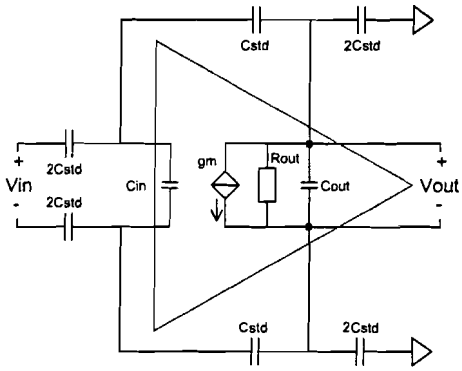


Figure 4.7: *Model of the opamp. The external capacitors represent the switched capacitor network of the SHA.*

Speed considerations

It is known from the references that the output resistance in combination with the capacitive load is responsible for the dominant pole of the circuit. For the model in figure 4.7, one can derive:

$$\begin{aligned}\frac{V_{out}}{V_{in}} &= \frac{sR_{out}C_{std}^2 - 2g_mR_{out}C_{std}}{sR_{out}\Big[4C_{std}^2 + 3C_{std}(C_{in}+C_{out}) + 2C_{in}C_{out}\Big] + \Big[(3+g_mR_{out})C_{std} + 2C_{in}\Big]} \\ &\approx \frac{-2g_mR_{out}C_{std}}{sR_{out}\Big[4C_{std}^2 + 3C_{std}(C_{in}+C_{out}) + 2C_{in}C_{out}\Big] + \Big[g_mR_{out}C_{std}\Big]} \qquad (4.14)\end{aligned}$$

yielding the DC gain $A_{0,cl}$ and the dominant pole frequency f_0:

$$A_{0,cl} = \frac{-2g_mR_{out}C_{std}}{(3+g_mR_{out})C_{std} + 2C_{in}} \approx -2 \qquad (4.15)$$

$$\begin{aligned}f_0 &= \frac{(3+g_mR_{out})C_{std} + 2C_{in}}{2\pi R_{out}\Big[4C_{std}^2 + 3C_{std}(C_{in}+C_{out}) + 2C_{in}C_{out}\Big]} \\ &\approx \frac{g_mC_{std}}{2\pi\Big[4C_{std}^2 + 3C_{std}(C_{in}+C_{out}) + 2C_{in}C_{out}\Big]} \qquad (4.16)\end{aligned}$$

The settling behavior of the SHA is determined by the pole frequency f_0. In the worst case situation, the output voltage starts at $V_{out}(0) = -A_{max}$ and the steady state value is $V_{out}(\infty) = A_{max}$:

$$V_{out}(t) = A_{max}\left(1 - 2e^{-t/\tau_0}\right), \text{ with } \tau_0 = \frac{1}{2\pi f_0} \qquad (4.17)$$

The SHA has to settle within V_{lsb} of the steady state value [2], to achieve the full accuracy. From this constraint follows the time period for the hold phase τ_{set}:

$$\left.\begin{aligned}V_{lsb} &= 2^{1-N}A_{max} \\ V_{out}(\infty) - V_{out}(\tau_{set}) &= 2A_{max}e^{-\tau_{set}/\tau_0}\end{aligned}\right\} \Rightarrow \tau_{set} = \tau_0 N \ln 2 \qquad (4.18)$$

A complete sample period T_s takes twice the settling time τ_{set}. Taking 10% additional time into account for switching overhead and using $N = 12$ yields:

$$f_s = \frac{1}{T_s} \approx \frac{1}{2.2\tau_{set}} = \frac{\pi f_0}{1.1 \cdot 12 \ln 2} \approx \frac{1}{3}f_0 \qquad (4.19)$$

[2] Settling within V_{lsb} at the output of the first SHA coincides with an input-referred error of $\frac{1}{2}V_{lsb}$ due to the gain of 2 in the SHA.

The sample frequency is determined by the dominant pole of the opamp. Equation 4.16 shows that the dominant pole is dependent only on the g_m of the input stage and on the size of (parasitic) capacitors, but not on R_{out}.

Large signal behavior: slew rate

Besides the dominant pole, the slew rate of the opamp is also important for the overall speed. The transient behavior (equation 4.17) shows that the maximum slew rate is required at $t = 0$. Together with equation 4.19 it follows that:

$$\left.\frac{\partial V_{out}(t)}{\partial t}\right|_{t=0} = \frac{2A_{max}}{\tau_0} \approx 12\pi A_{max} f_s \tag{4.20}$$

The maximum slew rate of the opamp is determined by the current setting of the output stage ($\frac{1}{2}I$) and the total capacitive load, connected to the output of the opamp (approximately $\frac{3}{2}C_{std}$, neglecting C_{out}):

$$\frac{1}{2}I = \frac{3}{2}C_{std}\frac{\partial V_{out}}{\partial t} \tag{4.21}$$

Combining equations 4.20 and 4.21 yields for $C_{std} = 1.2$ pF, $A_{max} = 0.8$ V and $f_s = 100$ MHz approximately:

$$I = 36\pi C_{std} A_{max} f_s \approx 10 \text{ mA} \tag{4.22}$$

This current setting is used as a first try, later on it is shown that this is almost the optimal setting for maximum speed.

Design of the input stage

The second step in the design is to determine the size of the differential pair. According to [21], the transconductance of the input stage is given by:

$$g_m = \frac{\partial i_{out}}{\partial v_{in}} = \frac{1}{2}\sqrt{I\mu_n C_{ox}\frac{W_{dif}}{L_{dif}}} \tag{4.23}$$

[3]

It was decided to use $L_{dif} = 240$ nm (the minimum length) and $W_{dif} = 960$ μm. The minimum length minimizes the parasitic input capacitance, the large width is necessary to achieve a high g_m, required for high speed operation. Simulations showed that around $V_{in} = 0$ V, $g_m = 4 \cdot 10^{-2}$ A/V.

[3] i_{out} is the differential output current of the input stage. W_{dif} and L_{dif} are the width and length of the two input transistors. $\mu_n C_{ox}$ is a constant, dependent on the chip process.

Design of the output stage

As explained before, the speed of the SHA is not related to the output resistance of the output stage. Therefore, the design of the folded-cascode stage is aimed at simplicity. The upper PMOS and lower NMOS transistors are biased by a single reference current. The transistors have a large width to be able to conduct the large current setting, without losing too much voltage between drain and source. The cascode devices have the same bias voltage V_{cm} at their gates. From the UMC25 technology documentation, it is known that:

$$0.44 \text{ V} \leq |V_{tn}| \leq 0.64 \text{ V} \quad \text{and} \quad 0.48 \text{ V} \leq |V_{tp}| \leq 0.68 \text{ V} \tag{4.24}$$

As long as the single-ended output voltage remains in the range $[V_{cm}-0.4, V_{cm}+0.4]$ all devices will remain in the saturation region. Because of this, the differential output range becomes $[-0.8, 0.8]$ which clarifies the use of $A_{max} = 0.8$ V throughout this thesis. Finally, the width of the cascode devices is minimized to reduce C_{out}.

Pole-zero cancellation technique

The stability of the opamp was verified in an open loop configuration with an AC simulation. A differential capacitive load of 1.6 pF is connected to the opamp's output, representing the switched capacitor network of the SHA. The achieved gain-bandwidth product (GBP) is 2.6 GHz, with a phase margin (PM) of 50^o. Although this system will be stable in a feedback configuration, a larger phase margin (preferably 65^o) would result in a faster settling behavior.

The phase margin is dependent on the second (non-dominant) pole of the operational amplifier. According to [16] and [17], the PMOS devices in the folded-cascode stage are responsible for this pole. Two separate methods have been investigated to improve the PM to 65^o. First, the external capacitive load was increased. This reduces the GBP (and the overall speed) but improves the phase margin. An external load of 4.8 pF was required to achieve $PM = 65^o$. Because of this load, the GBP is reduced to 1.1 GHz.

The second method is a *pole-zero cancellation* technique, as proposed by [16] and [17]. For this method, two resistors and two capacitors are added to the circuit (see figure 4.8). For low frequencies, the circuit behaves as the original design. For higher frequencies, were the pole of the PMOS devices becomes a bottleneck, the current of the first stage is fed through the capacitors, bypassing the PMOS devices. In the opamp's transfer function it means that a zero is added, compensating for the non-dominant pole. Using this technique, a phase margin of 64^o is obtained with a GBP of 2.36 GHz.

Figure 4.9 shows the transfer functions of the three designs: the original design achieving a phase margin of only 50^o, the design with increased load resulting in a low GBP (1.1 GHz), and the design with pole-zero cancellation, achieving both a

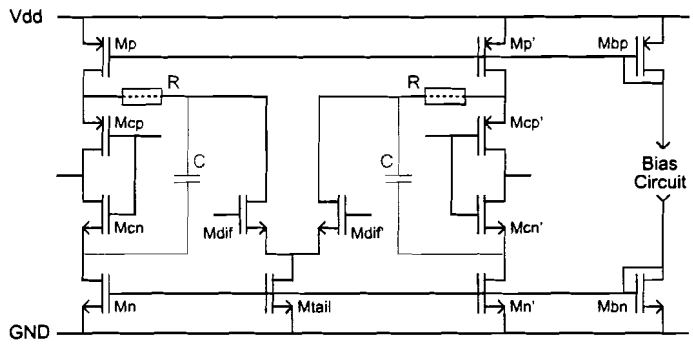


Figure 4.8: *Folded-cascode opamp with pole-zero cancellation technique.*

high GBP (2.4 GHz) and a good phase margin (64°). Because of these results, the design with pole-zero cancellation will be used from now on.

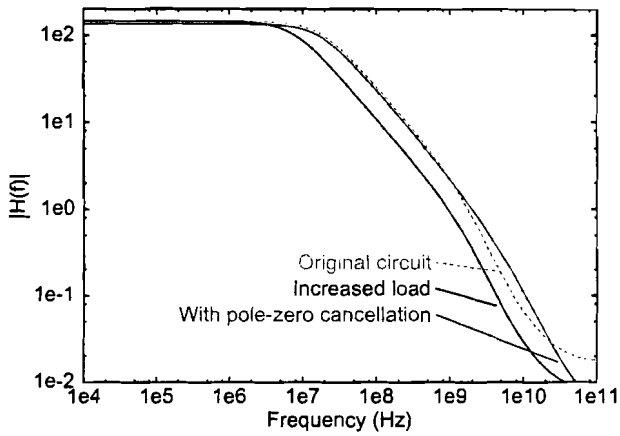


Figure 4.9: *Open loop response $|H(f)|$ of the opamp.*

A well known problem of pole-zero cancellation is that the time response can be relatively slow when the pole-zero pair is not matched very well. However, in our situation the matching is not so critical as the frequency of the pole-zero pair (1.2 GHz) is relatively close to the GBP (2.4 GHz). All simulations performed, including mismatch simulations, show an acceptable settling behavior.

Final result

The opamp was designed according to the procedure described above. The exact circuits including all dimensions of the components are inserted in appendix A. Simulations have been performed to measure the parameters of the model in figure 4.7 and to determine the phase margin and power consumption. Detailed results can be found in appendix B. Here, the most important parameters are summarized in table 4.1. During all simulations, the opamp was used in an open loop configuration.

An external load capacitance C_{ext} was connected between the output terminals of the opamp, representing the load of the SHA during normal operation.

Setting	Value
V_{dd}	2.5 V
V_{cm}	1.25 V
T	300 K
C_{ext}	1.6 pF

Parameter	Value
C_{in}	2.6 pF
C_{out}	0.7 pF
g_m	34 mA/V
R_{out}	4 kΩ
A_0	136
GBP	2.36 GHz
PM	64 ^{o}
P_{total}	90 mW

Table 4.1: *Simulation results of the folded-cascode opamp with pole-zero cancellation.*

Optimizing speed

It is known that the dominant pole frequency of the closed loop SHA (equation 4.16) is determinative for the overall speed. To optimize the opamp for a given external capacitor size C_{std}, a scaling technique can be applied. While C_{std} remains constant, all components in the opamp are scaled with a factor α: transistor widths W are replaced by αW, capacitors C by αC and resistors R by R/α. After scaling, equation 4.16 should be rewritten as:

$$f_0 \approx \frac{\alpha g_m C_{std}}{2\pi\left[4C_{std}^2 + 3\alpha C_{std}(C_{in} + C_{out}) + 2\alpha^2 C_{in} C_{out}\right]} \tag{4.25}$$

Figure 4.10 shows the dominant pole frequency f_0 as a function of the scaling parameter α. Maximum speed is achieved for $\alpha = 1.38$. Nevertheless, it is decided not to apply scaling, as the optimum is only 2% faster than the unscaled circuit but requires almost 40% more chip area and power consumption. Based on table 4.1 and equations 4.19 and 4.25, a maximum sample frequency of 100 MHz is expected.

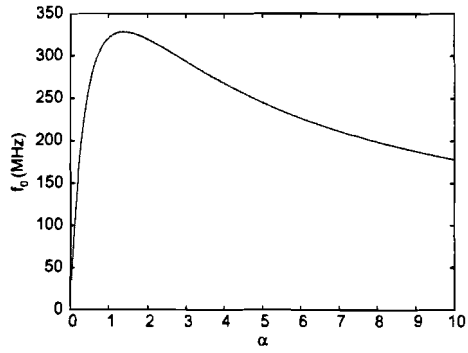


Figure 4.10: *Dominant pole frequency as a function of scaling parameter α.*

4.4.4 Sub-ADC

The sub-ADC has not been implemented yet. To be able to perform simulations on the complete SHA, an ideal model for the sub-ADC was substituted. In short, this model contains two comparators, giving a coarse quantization of the analog input voltage. The model also contains some digital logic to latch the digital code on the sample moment and some additional logic for the calibration algorithm. In appendix A, this model is given together with an explanation of its functionality.

4.4.5 Switch matrix

The switch matrix contains three parts: the sub-DAC, some digital logic and a set of switches. The sub-DAC produces three constant reference voltages. The switches connect the correct signals to the four capacitors in the SHA (C_1, C_1', C_2 and C_2' in figure 4.4). The digital logic turns on the correct switches, dependent on the digital input signals of the switch matrix. During the *sample* phase, the analog input of the basic block is selected. During the *hold* phase, the switches select the sub-DAC. The switches also determine which of the three DAC-levels is present at the output of the switch matrix. At the end of the *hold* phase, the switches connect the common-mode voltage to the output, resetting the capacitors in the SHA. For calibration purposes, the switch matrix can also select one of the two reference voltages of the sub-ADC. Table 4.2 lists all possible states of the switch matrix. In appendix A, detailed information about the switch matrix can be found.

State description	**Output of the switch matrix**
Hold phase, `d1d0` = 00	Sub-DAC level 1: $-V_{DAC}$
Hold phase, `d1d0` = 01	Sub-DAC level 2: 0
Hold phase, `d1d0` = 11	Sub-DAC level 3: $+V_{DAC}$
Sample phase, normal operation	Analog input: V_{in}
Sample phase, calibration mode, `C01` = 0	Sub-ADC level 1: $-V_{ADC}$
Sample phase, calibration mode, `C01` = 1	Sub-ADC level 2: $+V_{ADC}$
Reset phase	0 (V_{cm} on all outputs)

Table 4.2: *Possible states of the switch matrix.*

4.4.6 Switches

The switches in the SHA and the switch matrix have to fulfill several requirements. Because of the large sampling capacitors and the high sample frequency, switches with a low on-resistance are required. As each basic block contains 28 switches, simplicity and required chip area are important aspects as well.

The first idea was to use switches with a clock boosting technique, according to [22]. Here, the switch is a simple NMOS device, but the voltage driving the gate of the switch is boosted above the normal power supply. The gate-driving voltage is

dependent on the signal voltage at the source of the switch, such that the voltage V_{gs} is almost constant, independent on the signal to be switched. Due to this independence, this switch has an improved linearity. Bootstrapping the driving voltage reduces the on-resistance of the switch, thereby increasing the speed. An important drawback of this boosted switch is the complexity and the large chip area required for the circuit driving the switch.

The second idea, used in the final circuit, is to use clock boosting, but independent on the signal to be switched. By doing so, only a single boosting circuit is required, that can be used for all switches in the converter. Figure 4.11 shows the implementation of the switch. Instead of using the positive power supply V_{dd} to drive the switch in the 'on'-state, a higher voltage is used. In our case, 4 V is used to drive the switch, as this is sufficiently below the maximum allowed voltage of the UMC technology, but high enough to control the switches correctly. The circuit producing this voltage was not implemented: it can be made available off-chip or generated internally.

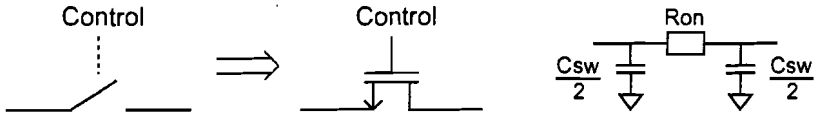


Figure 4.11: *Implementation of the switch (middle) and model during the 'on'-state including the on-resistance and the parasitic capacitance (right).*

The switch has two important parameters: the on-resistance and the parasitic capacitance. Increasing the width W of the transistor reduces the on-resistance, but increases the parasitic capacitance. Dependent on the source and the load connected to both sides of the switch, an optimal width exists, achieving maximum speed. This means that the size of the switches should be optimized, dependent on their location in the basic block. However, to minimize the design time, the same size is used for all switches. Simulations with different sizes were performed to find the optimum. Eventually, the size of the switches was fixed at:

$$\frac{W}{L} = 8\frac{12\ \mu\text{m}}{0.24\ \mu\text{m}} \tag{4.26}$$

Although the on-resistance and the parasitic capacitance of the switch are voltage-dependent, an estimation of these parameters was obtained using a large-signal step-function: $R_{on} \approx 15\ \Omega$ and $C_{sw} \approx 400$ fF.

4.4.7 Capacitors

The SHA contains capacitors at two locations: in the switched-capacitor network (figure 4.4) and in the operational amplifier (figure 4.8). The capacitors are implemented in the UMC technology with MMC capacitors. The non-linearity of this specific capacitor-type was taken into account during all simulations. Also, the parasitic capacitance (between the bottom-plate of the capacitor and the substrate)

was included by adding an additional capacitance between one of the nodes of the capacitor and the ground. The mismatch between capacitors was not taken into account during simulations as it is very small ($\sigma_c \approx 1 \cdot 10^{-2}\%$). Although this mismatch consumes a part of the 'error budget' of the correction technique (equation 4.4), this effect is negligible compared to the influence of mismatch of the transistors in the opamp and hence left out of consideration.

4.5 Simulation results

In this section, the simulations results are discussed shortly. The results are compared to the constraints from section 4.2 to show the correctness of the design.

4.5.1 DC and AC analysis

Both a DC and an AC analysis were performed on the opamp. Results were already given in table 4.1 (page 60). Additional information can be found in appendices B.1.1 and B.1.2. The DC-gain of the opamp is relatively low (136 (= 43 dB)). Because of this, the suppression of deviations in the opamp is not so good. However, as post-correction is applied, this is not a problem. Based on the AC analysis, a sample frequency of 100 MHz is expected to be feasible. A derivation of this value has been given in the end of section 4.4.3 already. The stability of the opamp, determined by the phase margin, has been verified under severe deviations of power supply voltage, temperature and component values. Appendix B.1.3 shows exactly which deviations were taken into account. In the worst case, a phase margin of 60.5 ^{o} was achieved, ensuring good stability.

4.5.2 Transient analysis

A transient analysis has been performed on the SHA. All components of the SHA (opamp, switches, etc) have been implemented as described before. Detailed circuit information can be found in appendix A.

The simulation results (see also appendix B.2.1) show that the settling time of the SHA within $\frac{1}{2}V_{lsb}$ of the final value equals 3.6 ns. It also shows that the SHA is capable of moving from the maximum negative output voltage at one clock cycle to the maximum positive output voltage in the next clock cycle. There is some memory effect (the value of the previous sample has influence on the current sample), but the deviation is always less than $\frac{1}{2}V_{lsb}$ (shown in appendix B.2.2).

Sensitivity and mismatch analyses were performed to investigate the influence of variations in power supply, temperature and components on the settling time. Appendix B.2.4 shows the detailed results. The achieved settling time varies between 3.2 and 4.4 ns. The last value is an extraordinary pessimistic value, measured when all components deviate 3σ from their nominal value in the most harmful direction. In case of the worst case scenario, the subsequent SHA can sample the output 4.4 ns

after the beginning of the *hold* phase. Taking 0.5 ns into account for the subsequent SHA to sample correctly, and another 0.5 ns for the SHA to reset to the zero state, yields a complete period of 2·(4.4+0.5+0.5) = 10.8 ns, thus $f_s = 92.5$ MHz. According to the results in appendix B.2.4, it is likely that in most situations $f_s = 100$ MHz can be achieved.

4.5.3 Accuracy analysis

Using many transient analyses with different input voltages, the transfer function of the SHA has been derived. In appendix B.2.2, the transfer function is inserted. From this picture, the parameters A and A_3 (defined in chapter 3) are derived:

$$\begin{aligned} A &= 1.8647 \\ A_3 &= 3.3 \cdot 10^{-2} \end{aligned}$$

The gain A is less than 2 due to the parasitic capacitances and the gain of the opamp being finite. The values for A and A_3 are valid when all components have their ideal nominal value. In case of mismatch, deviations will occur. Based on the mismatch simulations performed on the opamp (see appendix B.1.3), it is concluded that third order distortion is always the dominant source of non-linearity. Moreover, the third order distortion component is relatively insensitive to deviations. Because of that reason, the values mentioned above are considered to give a correct description of the transfer curve of the SHA. Both A and A_3 satisfy the constraints from section 4.2. The gain, being less than 2, actually improves the available redundancy for solving offset errors and errors in the sub-ADC and sub-DAC.

The input-referred offset of the SHA is determined mainly by the input-referred offset of the opamp, but also by the capacitors and switches. With some simulations, it was found that:

$$V_{off,in,SHA} \approx 3V_{off,in,opamp} \tag{4.27}$$

The offset voltage of the amplifier is caused by mismatch of transistors and resistors. Suppose, we number all parameters influencing the offset voltage with an index i. Such a parameter can for example be the size $(\frac{W}{L})$ of a transistor, or the value (R) of a resistance. Each parameter has a stochastic deviation of its nominal value: σ_i. The input-referred offset voltage of the opamp that arises when parameter i has a stochastic deviation of one σ_i is indicated by $\sigma_{off,i}$. Assuming, as a first order approximation, that all stochastic parameters are mutual independent, the standard deviation of the input-referred offset voltage of the opamp can be estimated by:

$$\sigma_{V_{off,in,opamp}} \approx \sqrt{\sum_i \sigma_{off,i}^2} \tag{4.28}$$

For the simulation results shown in appendix B.1.3, $\sigma_{off,i}$ was determined for all components. With the values from these simulations, the equations above and equation 3.5, the following result is obtained:

$$\begin{aligned} \sigma_{V_{off,in,opamp}} &= 2.7 \text{ mV} \Rightarrow \\ \sigma_{V_{off,in,SHA}} &= 8 \text{ mV} \\ \sigma_{off} = \frac{\sigma_{V_{off,in,SHA}}}{A_{max}} &= 1 \cdot 10^{-2} \end{aligned} \tag{4.29}$$

This result, together with the result that $\delta_A < 0$ (as $A < 2$), can be filled out in the design constraint from equation 4.4. The resulting formula is a design constraint for the accuracy of the sub-ADC and the sub-DAC:

$$\begin{aligned} \frac{1}{3} + 3(\sigma_{ADC} + \sigma_{DAC} + 0.01) &\leq \frac{1}{2} \\ \sigma_{ADC} + \sigma_{DAC} &\leq 4.5 \cdot 10^{-2} \end{aligned} \tag{4.30}$$

4.5.4 Noise analysis

As explained before in section 4.4.2, two contributions are responsible for the noise present at the output of the SHA: sampled noise and time-continuous noise. In section 4.4.2, the contribution of sampled noise was calculated:

$$\sigma^2_{V_{n,out,1}} = 2.07 \cdot 10^{-8} \text{ V}^2 \tag{4.31}$$

The contribution of time-continuous noise was simulated (see appendix B.2.3):

$$\sigma^2_{V_{n,out,2}} = 1.35 \cdot 10^{-8} \text{ V}^2 \tag{4.32}$$

The total noise power at the output of the SHA becomes:

$$\sigma^2_{V_{n,out}} = \sigma^2_{V_{n,out,1}} + \sigma^2_{V_{n,out,2}} = 3.42 \cdot 10^{-8} \text{ V}^2 \tag{4.33}$$

Making this value input-referred and relative to the maximum amplitude A_{max} according to the design model (equation 3.6) yields:

$$\sigma_n = \frac{\sigma_{V_{n,out}}}{2A_{max}} = 1.16 \cdot 10^{-4} \tag{4.34}$$

This value is slightly worse than required by the design constraint ($\sigma_n \leq 9.25 \cdot 10^{-5}$), but the design constraint was aimed at an accuracy of 12.4 bits. With the actual value obtained from simulations, the accuracy is limited to 12.1 bits.

4.6 Conclusions

In this chapter, the transistor-level design of the most critical components of the basic block was described. The theory given in chapter 3 was used to derive design constraints for the sub-ADC, sub-DAC and sampe-and-hold amplifier (SHA). The SHA, composed of an operational amplifier, switches and capacitors was implemented completely. The sub-ADC and sub-DAC were partly implemented and partly modelled.

Simulations were performed and showed that the presented design fulfills the design constraints: the design is accurate enough to achieve 12 bits resolution and fast enough for a sample frequency of 100 MHz. The achieved resolution is limited by thermal noise and non-linearity of the SHA. Figure 4.12 shows the achieved result compared to the reference designs from section 2.1.1. Table 4.3 compares the result of this project to two reference designs ([23] and [24]), also indicated in figure 2.1.1.

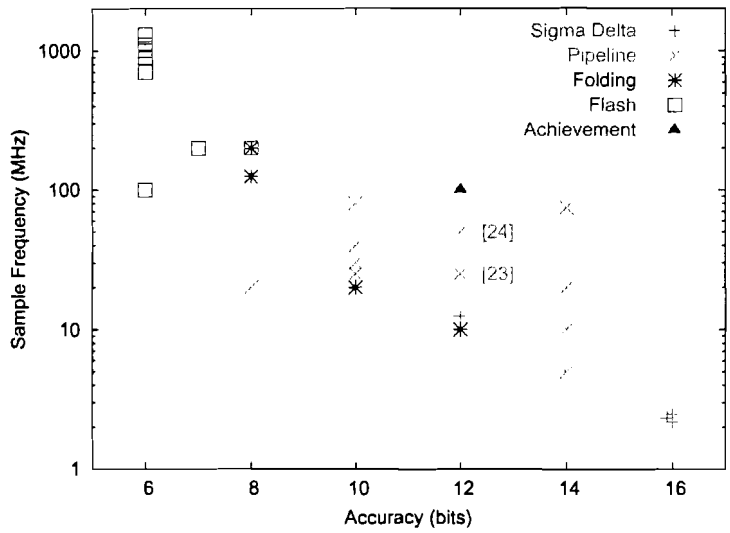


Figure 4.12: *Comparison between available state-of-the-art converters (with respect to speed and accuracy) and the result of this project.*

Design	f_s (MSPS)	SNDR (dB)	ENOB (bits)	Power (W)
Reference [23]	25	70	11.4	0.08
Reference [24]	50	58	9.4	0.20
This design	100	≈70	≈11.4	>1.44

Table 4.3: *Design result compared to two reference designs. The SNDR estimation of this design is based on the results from the accuracy analysis (linearity of 12.7 bits) and the noise analysis (SNR of 12.1 bits). The power consumption of this design includes the opamps only.*

Simulations also showed an excellent robustness and insensitivity to process spreads and environmental conditions. Finally, simulations were used to derive a simple design constraint for the required accuracy of the sub-ADC and sub-DAC in combination with the presented SHA (equation 4.30).

Chapter 5

Implementation of the digital post-correction algorithm

This chapter deals with the implementation of the digital post-correction algorithm (as described in section 2.2.2) in a FPGA. Simulations were performed to verify the functionality of the implementation.

5.1 Introduction

Figure 5.1 shows the structure of the pipelined converter, using post-correction.

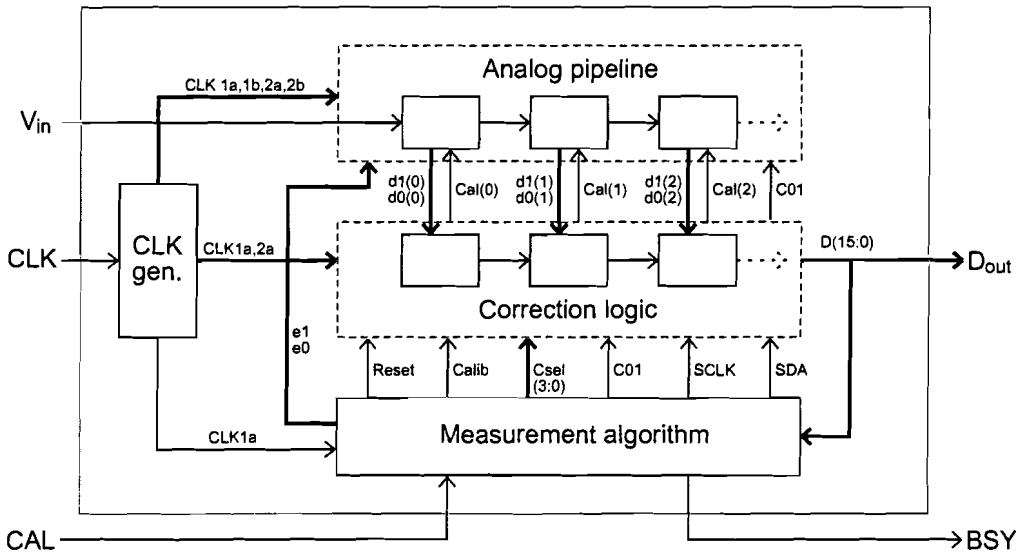


Figure 5.1: *Structure of a pipelined ADC including post-correction.*

The analog pipeline is a concatenation of basic blocks. In the digital domain two parts can be distinguished:

- **Correction logic**
  Digital logic is needed to combine the output codes from the basic blocks to a single output, using the variable weights determined during the measurement phase. This logic remains necessary as long as the converter operates.

- **Measurement algorithm**
  This algorithm determines the optimal weights for each basic block one by one and has to be performed once before the converter can be used.

In the next sections, the implementation of the digital parts is discussed.

5.2 Implementation of the correction logic

One of the two digital parts of the pipelined converter is the correction logic. As visible in figure 5.1, this logic combines the outputs of the basic blocks to a single, corrected output code. In fact, this logic implements the mapping function, previously described in section 2.2.2:

$$M(U(0)) = \sum_{j=0}^{k-1} \omega_{u(j)}(j) \tag{5.1}$$

The correction logic is implemented in a pipelined structure, as indicated by figure 5.1. A single cell of this digital pipeline is drawn in figure 5.2. In each cell, the correct weight $\omega(i)$ is selected, dependent on the input code from the sub-ADC (`d1d0(i)`). The output of each cell is the summation of the output of the preceding stage and weight $\omega(i)$. The output of the last stage thus equals equation 5.1. To be able to apply post-correction, the weights ω_0 and ω_2 are programmable by the external measurement algorithm.

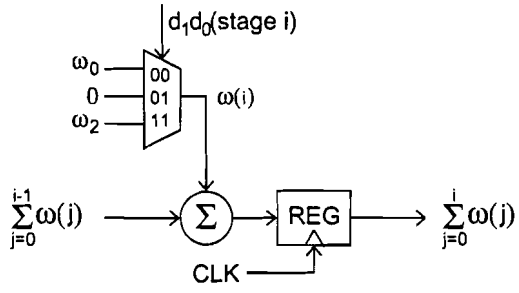


Figure 5.2: *Basic cell of the digital correction logic.*

In appendix C, the circuits implementing the digital basic block (figure 5.2), the pipeline and the programming interface are inserted. The circuit is designed using basic digital cells like registers, multiplexers, etc. Because of the simplicity of the circuit, it will be implemented as dedicated hardware on the same chip as the analog circuit. Moreover, an on-chip implementation reduces the required pins of the chip.

5.3 Implementation of the measurement algorithm

In this section, the implementation of the measurement algorithm in a FPGA is discussed. It was decided to use a FPGA as the measurement algorithm is needed only once. After the measurement phase, the FPGA can be reprogrammed for another task. Moreover, the overall design time is minimized and the FPGA provides the flexibility to make changes to the algorithm in the future.

Before working out the implementation, a short recapitulation of the measurement algorithm is given. Then, the implementation is explained with flowcharts. The actual circuits are inserted in appendix D. Finally, simulation results are given.

5.3.1 Theory

The theory of the measurement algorithm has been explained in section 2.2.2. In short, the measurement algorithm takes the following actions:

1. all weights in all stages are set to zero,
2. the weights of the last stage are set to:

$$\omega_0 = -1 \text{ and } \omega_2 = 1 \ ,$$

3. beginning with the one-but-last stage, the weights of the other stages are measured and stored one by one. The first stage in the pipeline is the last stage to be measured.

For each stage, four different measurements have to be performed. For each of these measurements, a unique combination of input signals is applied to the stage, as indicated by table 5.1. To make the system more reliable, each measurement consists of many identical sub-measurements. The average outcome is the measurement result. The weights are determined by taking the difference of two measurements:

$$\begin{aligned} \omega_0 &= \text{measurement } \mathbf{b} \text{ - measurement } \mathbf{a} \\ \omega_2 &= \text{measurement } \mathbf{c} \text{ - measurement } \mathbf{d} \end{aligned}$$

measurement	analog input	digital input
a	V_{ADC1}	00
b	V_{ADC1}	01
c	V_{ADC2}	01
d	V_{ADC2}	11

Table 5.1: *The combination of input signals applied to a stage for each of the four measurements.*

5.3.2 Implementation

In figure 5.1, all external signal connections of the measurement algorithm were shown. Signals CAL and BSY provide the user interface, the other signals are used to communicate with the ADC. The functionality of these pins is as follows:

- a positive pulse on CAL initiates the calibration measurements,
- during the whole calibration phase, BSY is 1. As soon as the algorithm finishes, BSY becomes 0,
- Reset resets all weights and registers in the correction logic to 0,
- D(15:0) provides the algorithm with the measured data from the pipeline,
- Calib enables the calibration mode of one of the stages,
- Csel(3:0) indicates which of the 16 stages is under measurement,
- C01 determines whether V_{ADC1} or V_{ADC2} is applied to the stage under measurement. Also, it selects the weight to be programmed (ω_0 or ω_2).
- e1e0 determines whether code 00, 01 or 11 is applied to the stage under measurement,
- SCLK and SDA provide a serial data interface to the correction logic. Data is shifted out on the rising edge of SCLK, with the most significant bit first. The interface is used to store the calculated weights in the correction logic. Csel(3:0) and C01 determine to which register the data is written.

Calib main function

In figure 5.3, the main flow diagram of the calibration algorithm is shown. This diagram is implemented in the circuit called **Calib**. The converter starts in the normal operation mode. As soon as the user places a positive pulse on the CAL input, BSY becomes 1, indicating that the converter is busy with the calibration cycle. First, a pulse is given on the Reset output, to reset all weights and internal registers of the correction logic. Then, the Calib signal becomes 1, putting the converter in calibration-mode. With Start1, a sub-system called **LastSt** is triggered to start its task. **LastSt** sets the weights ω_0 and ω_2 of the last stage in the pipeline to -1 and $+1$ respectively, and will be discussed later in this chapter. When **LastSt** finishes its task, it produces an output flag Ready1. Then, with Start2, sub-circuit **CalSt** is triggered to start with the actual calibration algorithm. When **CalSt** is ready, Ready2 becomes 1 and the main state machine returns to the normal operation mode. The BSY indicator goes low, and the user knows that the converter is ready for use. In figure 5.4, a sketch is drawn, showing how these signals change in time.

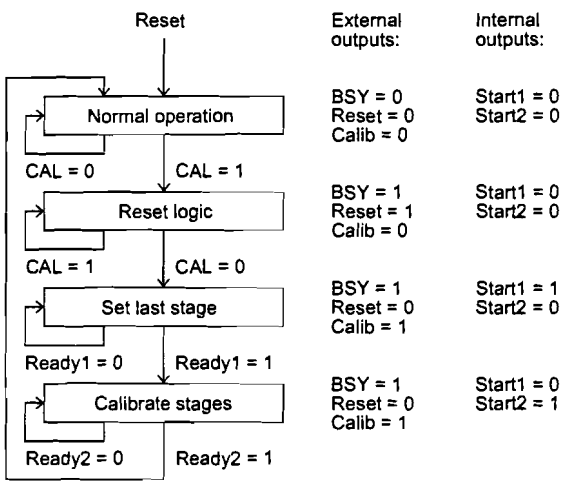


Figure 5.3: *Flowchart of* ***Calib****, the main function of the calibration algorithm.*

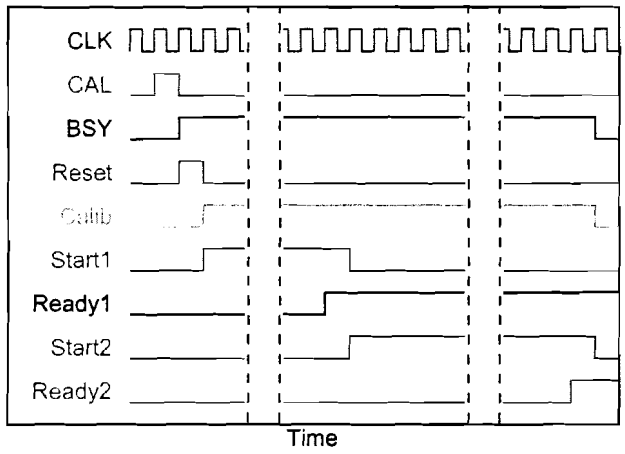


Figure 5.4: *Sketch of the behavior of function* ***Calib*** *in time.*

LastSt sub-function

The function of **LastSt** is to set the weights of the last stage to $\omega_0 = -1 = 0x\text{FFFF}$ and $\omega_2 = 1 = 0x0001$, using the serial interface (`SCLK` and `SDA`) [1]. As all weights are set to zero ($0x0000$) before **LastSt** is started, 16 ones have to be transmitted for ω_0, but for ω_2, a single transmission is enough. The flowchart in figure 5.5 shows the behavior of **LastSt**. A `Reset` input resets the system to state 0. As soon as the main function **Calib** produces a signal on `Start1`, **LastSt** starts to transmit data. States 1 and 2 in the flowchart are repeated 16 times, using an internal `counter`. During this loop, the data to ω_0 is transmitted. In states 3 and 4 a single clock pulse is given to set ω_2 to the correct value. Finally, the algorithm ends up in state 5, where `Ready1` is set. The only way to reset the state machine to state 0 is when

[1] Data is stored in a 16-bit 2's-complement format.

Calib transmits a `reset` signal.

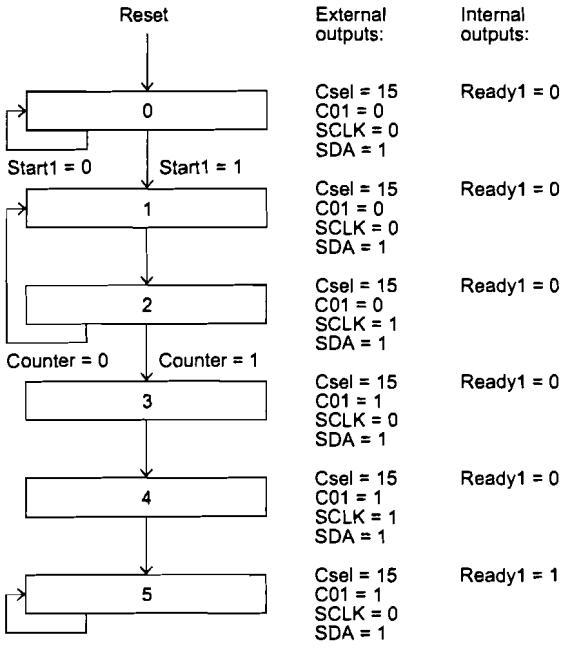


Figure 5.5: *Flowchart of* ***LastSt****, a sub-function of function* ***Calib****.*

Figure 5.5 also shows the values of several outputs, dependent on the state, required to set the correct stage (`Csel`) and the correct weight (`C01`) to be accessed. Figure 5.6 shows the behavior of the produced signals as a function of time.

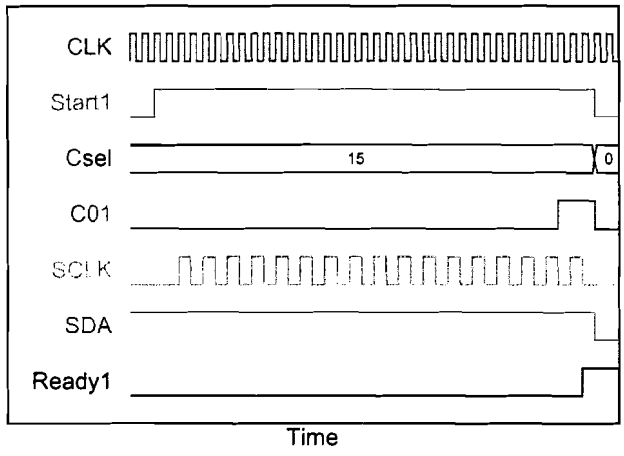


Figure 5.6: *Sketch of the behavior of function* ***LastSt*** *in time.*

CalSt sub-function

The **CalSt** function, a sub-function of **Calib**, implements the actual calibration. It performs the four measurements for each stage, determines the weights, and stores the results in the correction logic. Figure 5.7 (left) shows the flowchart of **CalSt**. A pulse on `Start2` starts the process. The loop (composed of states *Loop*, *Delay* and *Measure*) is executed once for each measurement. A sub-function (**Controller**) sets all required parameters (`Csel`, `C01` and `e1e0`) for each measurement and produces a `Controller ready` signal when all measurements are performed. At the beginning of each loop cycle, a `IncState` signal is transmitted to **Controller** to indicate that a new measurement has to be started. Then, a short delay is introduced before the actual measurement starts. This is necessary as the converter is pipelined, so it takes some time before the first valid data can be seen at the output of the converter. Finally, the measurement is performed. Note that each measurement is composed of many (2^{16}) identical sub-measurements to average out random deviations. The amount of required sub-measurements depends on the required accuracy and the amount of noise obstructing the measurement. The value of 2^{16} was chosen as a first estimation, based on available designs, but might need adjustment in practice.

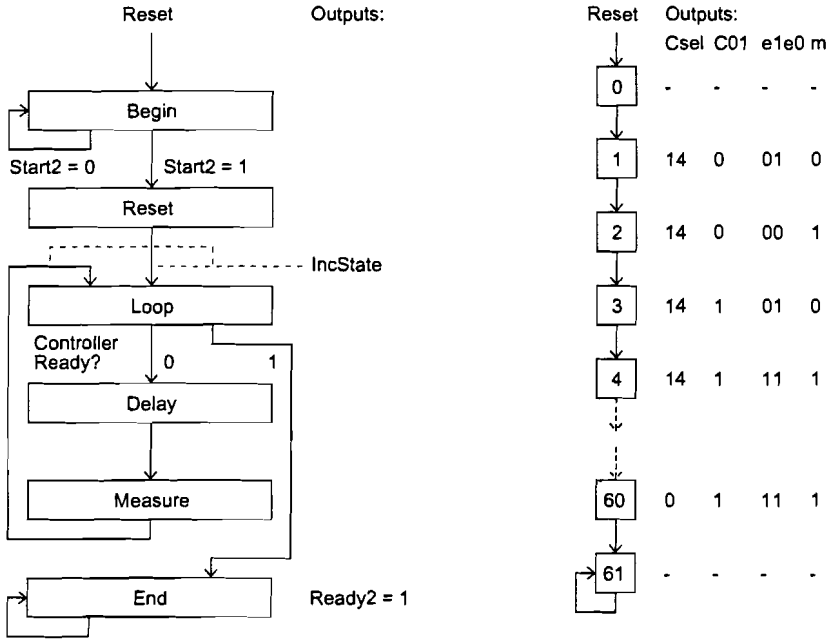


Figure 5.7: *Flowcharts of* ***CalSt*** *(left) and* ***Controller*** *(right).*

Controller sub-function

Figure 5.7 also shows the flowchart of **Controller**. This function arranges all settings, required for each measurement, as indicated in the figure. `Csel` selects the stage to be calibrated, `C01` determines the analog input applied to the stage (either V_{DAC1} or V_{DAC2}) and `e1e0` is the digital code that is applied to the stage. A pulse

on the `IncState` input increments the state by one. Besides the 60 states arranging the measurement setup, a reset state (0) and a ready state (61) are available.

Comparing the flowchart with table 5.1 reveals that the measurements are performed in the order: **b**, **a**, **c** and **d**. Also, it is known that the weights equal $\omega_0 = \mathbf{b} - \mathbf{a}$ and $\omega_2 = \mathbf{c} - \mathbf{d}$. Signal `m`, produced by **Controller**, indicates whether a measurement should be added or subtracted to the weight.

Result sub-function

The measurement data, coming from the digital pipeline is stored in a function called **Result**. This function also calculates the average value of the 2^{16} sub-measurements and it calculates the weights. **Result** contains a 32-bit register `S` [2] where the measurements are stored. As the register is 32-bit, while the data to be measured is only 16-bit, one can simply sum the 2^{16} sub-measurements in this register. The (16-bit) average value is then given by the 16 most significant bits of `S`: `S(31:16)`. Signal `m` (produced by **Controller**) plays an important role in the **Result** function. A high-to-low transition of `m` resets `S` to 0. If `m` = 0, measurement data is added to `S`, otherwise, data is subtracted from `S`.

As an example, consider the flowchart of **Controller**. Before the first measurement starts, `S` is reset to 0. After the first set of measurements (state 1), 2^{16} measurements on stage 14 are performed, using measurement setup **b**, thus:

`S(31:16)` = measurement **b**

After the second measurement, the measurements of type **a** are subtracted from `S`:

`S(31:16)` = measurement **b** - measurement **a**

Thus, `S(31:16)` exactly equals $\omega_0(14)$. After state 2, `S` is reset to zero, and measurements **c** and **d** are performed. After these measurements `S(31:16)` equals $\omega_2(14)$. Likewise, after state 6, `S(31:16)` equals $\omega_0(13)$, after state 8, `S(31:16)` = $\omega_2(13)$ and so on.

Counter sub-function

The **Counter** counts the number of sub-measurements that has been performed. When 2^{16} sub-measurements are performed, a ready signal is produced.

Store sub-function

As soon as the **Counter** produces a ready signal, the **Store** algorithm becomes active. Its function is to transfer the weight, calculated in the **Result** circuit, to the (external) correction logic, using the serial output interface. First of all, the

[2] In practice, this register is split up in two 16-bit registers to minimize the delay time.

value of m is checked. When m = 0, a measurement has been finished, but there is no weight to be transferred. In that case, **Store** immediately produces a ready signal, such that **CalSt** goes on with a new measurement. When m = 1, S(31:16) contains a new weight to be transferred. As the correct stage (Csel) and the correct weight (C01) are already indicated by **Controller**, the **Store** function only has to produce the correct data and clock signals on SDA and SCLK respectively. After the transmission, **Store** produces a ready signal, and **CalSt** starts a new measurement.

5.3.3 Simulation results

The circuits, in this chapter depicted by flowcharts, were designed in Matlab Simulink, using the Xilinx Blockset Library. With Xilinx System Generator, these circuits are translated automatically to VHDL code. Finally, with the Xilinx Foundation package, this VHDL code is translated to a FPGA programming file.

Simulations have been performed at two levels. First, in Simulink the behavior of the circuits was verified. Secondly, the FPGA code was simulated with Xilinx Foundation. This simulation takes the actual delays of the FPGA into account, so the maximum operating frequency can be derived.

Simulink simulations

Simulations in Simulink were performed to verify the behavior of sub-circuits and of the circuit as a whole. Two simulation results of sub-circuits were already shown (figures 5.4 and 5.6). To simulate the circuit as a whole, a model of the converter (according to chapter 3) was connected to the measurement algorithm. Several combinations of errors were used in the converter, to verify the weights produced by the algorithm. All simulations agreed with the theory.

For illustration, figure 5.8 shows a part of the measurement procedure. The stages are calibrated one-by-one according to CSEL. The different combinations of C01 and e1e0 determine the measurement conditions for the stage under calibration. The signal bursts at SCLK and SDA indicate that weights are being stored externally.

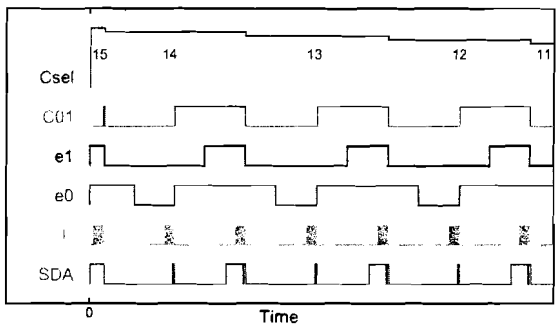


Figure 5.8: *Sketch of the behavior of the measurement algorithm in time.*

Xilinx Foundation simulations

After verification of the behavior in Simulink, the circuit was exported to a FPGA programming file. The target device is a Xilinx Virtex XCV2000E -6 BG560C FGPA, as this device was already available in the research group. The Xilinx Foundation software maps the design on the physical available resources in this chip. Then, a model is extracted from this mapping, including all timing information according to the place and routing on the chip. This model was simulated to verify the behavior of the circuit again, and to check the maximum path delay, determining the maximum operating frequency.

The synthesized circuit, taking place&routing information into account, has a maximum path delay of 9.5 ns. This means that the circuit will operate correctly at clock frequencies of at least 105 MHz. This is good enough, as the analog part of the converter will achieve 100 MHz at most.

5.4 Conclusions

In this chapter, the implementation of the digital logic for the post-correction algorithm was dealt with. The logic is divided in two parts: *correction logic* and *measurement algorithm.*

The correction logic is required continuously and will be implemented as dedicated hardware on-chip. A circuit on logic level was designed, which can be mapped to a transistor-level implementation in the future.

The measurement algorithm is required only once. Therefore, it is implemented off-chip in a FPGA, as the FPGA can be reprogrammed for another task after the measurement phase. Also, the use of a FPGA greatly reduces the design time since the measurement algorithm is a relatively complex system. Simulations proved that the design is fast enough compared to the analog design, so it will not to be a bottleneck for the system.

Chapter 6

Conclusions

Based on an investigation of available high-speed analog-to-digital converters, it was decided to use a *pipelined* architecture to design a 12-bit, high-speed CMOS ADC. To provide a flexible and simple converter, each stage in the pipeline resolves 1.5 bits of information.

A digital post-correction algorithm was introduced and implemented in a FPGA to improve the accuracy of the pipelined ADC. A generic system-level description of a pipelined converter was presented. This model, including static error models, was used to derive design constraints for the analog part of the converter. The model also shows that for the same design target, the constraints for a converter with digital post-correction are reduced significantly compared to a converter without post-correction.

By designing the most critical analog components of the converter, the feasibility of a 12-bit, 100 MS/s converter in standard CMOS technology was demonstrated. Even though the design was restricted to a 1.5-bit resolution for each stage (including the first stage) and a standard CMOS technology, this result is comparable to current state-of-the-art designs.

Chapter 7

Recommendations

This project showed the feasibility of a 12-bit, high-speed ADC in a standard CMOS technology. In this chapter, possible improvements for the current system are mentioned. Also, the parts of the analog design that were not yet implemented are listed.

ADC architecture

Because of maximum simplicity and flexibility it was decided to design a pipelined converter based on 1.5-bits resolution per stage, without scaling the blocks along the pipeline. However, the price to be paid for this approach is an excessive power consumption: the operational amplifiers only consume 1.4 W (16 stages with 90 mW for each opamp). Although at the cost of flexibility and simplicity, reconsidering the decision to use identical and unscaled blocks may be worthwhile. Also, according to [15] and [25], the use of a multi-bit first stage can improve the performance with respect to accuracy and power consumption.

Design of analog and digital components

The design of the ADC is not yet complete. Some parts were not implemented at all, other parts can be improved. Here, an overview of the current state of all blocks is given:

- **sub-ADC**: modelled at high level, to be implemented. Equation 4.30 (page 65) shows the design constraint for the combination of sub-ADC and sub-DAC,
- **sub-DAC**: modelled at high level, to be implemented. Equation 4.30 (page 65) shows the design constraint for the combination of sub-ADC and sub-DAC. Another important design issue is that the sub-DAC needs a low output impedance to be able to drive the large sampling capacitors in the SHA at a high sample frequency,
- **switch logic**: implemented at logic level,

- **opamp**: implemented at transistor level,
- **SHA capacitor network**: implemented in UMC technology, but optimization is possible. As explained in section 4.5.3, the nominal gain of the SHA is less than 2. Although this improves the 'budget' to solve errors in for example the sub-ADC, it also increases the required amount of stages in the pipeline. By skewing the size of the capacitors, the nominal gain can be set to 2,
- **switches**: implemented, but optimization is possible, see section 4.4.6,
- **clock boosting circuit**: modelled with an ideal model,
- **on-chip correction logic**: implemented at logic level,
- **off-chip measurement algorithm**: implemented in a FPGA.

Dynamic errors

The error model of the basic block, presented in chapter 3, takes into account static errors only. A possibility is to extend the model with dynamic error sources. Also, the current digital post-correction algorithm does not consider dynamic errors. The feasibility and usefulness of extending the correction algorithm for dynamic errors can be investigated.

An example of a dynamic error is the memory effect of the SHA: the state of the SHA in the previous *hold* phase has influence on the next *sample* phase. To reduce this effect, the SHA is forced to a predefined state at the end of each *hold* phase. It is also possible to use digital post-correction to alleviate the memory effect. Normally, the correction algorithm selects one out of three weights, dependent only on the *current* code produced by the basic block (0, 1 or 2). Using nine weights in the post-correction algorithm (dependent on both the *previous* and the current *code* produced by the basic block) is sufficient to solve the memory effect. The reason that this is sufficient is because the memory effect is dependent only on the state of the sampling capacitors during the previous *hold* phase. During that phase, the state of the sampling capacitors is determined completely by the code (0, 1 or 2) produced during that clock-period. However, using post-correction to solve the memory effect increases the complexity of the correction logic. Also, the measurement algorithm has to be extended to measure the new weights.

Bibliography

[1] Nyquist, H.
CERTAIN TOPICS IN TELEGRAPH TRANSMISSION THEORY.
Transactions of the AIEE, vol. 47 (1928), p. 617 - 644.

[2] Pelgrom, M.J.M. and A.C.J. Duinmaijer, A.P.G. Welbers
MATCHING PROPERTIES OF MOS TRANSISTORS.
IEEE Journal of Solid-State Circuits, Vol. 24 (1989), No. 5, p. 1433 - 1440.

[3] Jespers, P.G.A.
INTEGRATED CONVERTERS; D TO A AND A TO D ARCHITECTURES, ANALYSIS AND SIMULATION.
New York, Oxford University Press Inc., 2001, ISBN 0-19-856446-5.

[4] Razavi, B.
PRINCIPLES OF DATA CONVERSION SYSTEM DESIGN.
Piscataway, IEEE Press, 1995, ISBN 0-7803-1093-4.

[5] Plassche, R. van de
INTEGRATED ANALOG-TO-DIGITAL AND DIGITAL-TO-ANALOG CONVERTERS.
Dodrecht, Kluwer Academic Publishers, 1994, ISBN 0-7923-9436-4.

[6] Shang-Yuan Chuang and T.L. Scully
A DIGITALLY SELF-CALIBRATING 14-BIT 10-MHZ CMOS PIPELINED A/D CONVERTER.
IEEE Journal of Solid-State Circuits, Vol. 37 (2002), No. 6, p. 674 - 683.

[7] Karanicolas, A.N. and Hae-Seung Lee, K.L. Bacrania
A 15-B 1-MSAMPLE/S DIGITALLY SELF-CALIBRATED PIPELINE ADC.
IEEE Journal of Solid-State Circuits, Vol. 28 (1993), No. 12, p. 1207 - 1214.

[8] Boni, A. and A. Pierazzi
A 150 MHZ TRACK-AND-HOLD AMPLIFIER IN 0.35μm CMOS.
IEE Conference Publication, No. 466 (1999), p. 90 - 93, ISSN 0537-9989.

[9] Boni, A. and A. Pierazzi, C. Morandi
A 10-B 185-MS/S TRACK-AND-HOLD IN 0.35μm CMOS.
IEEE Journal of Solid-State Circuits, Vol. 36 (2001), No. 2, p. 195 - 203.

[10] Sugimoto, Y. and H. Kakitani, H. Tsurumi
LOW-VOLTAGE, HIGH-SPEED, AND HIGH-PRECISION CURRENT-MODE SAMPLE-AND-HOLD CIRCUIT.
Electronics and Communications in Japan, Part 2, Vol. 78 (1995), No. 7, p. 68 - 81.

[11] Sugimoto, Y. and M. Sekiya
DESIGN OF A SUB-1.5V, 20 MHZ, 0.1% MOS CURRENT-MODE SAMPLE-AND-HOLD CIRCUIT.
Analog Integrated Circuits And Signal Processing, Vol. 20 (1999), p. 149 - 153.

[12] Sugimoto, Y. and S. Imai
THE DESIGN OF A 1V, 40MHZ, CURRENT-MODE SAMPLE-AND-HOLD CIRCUIT WITH 10-BIT LINEARITY.
In: Proc. IEEE International Symposium on Circuits And Systems, Vol. 2 (1999), p. 132 - 135.

[13] Sugimoto, Y.
A 1.5V CURRENT-MODE CMOS SAMPLE-AND-HOLD IC WITH 57-dB S/N AT 20 MS/S AND 54-dB S/N AT 30 MS/S.
IEEE Journal of Solid-State Circuits, Vol. 36 (2001), No. 4, p. 696 - 700.

[14] Waltari, M. and K. Halonen
A 220-MSAMPLE/S CMOS SAMPLE-AND-HOLD CIRCUIT USING DOUBLE-SAMPLING.
Analog Integrated Circuits And Signal Processing, Vol. 18 (1999), p. 21 - 31.

[15] Singer, L. and S. Ho, M. Timko, D. Kelly
A 12B 65 MSAMPLE/S CMOS ADC WITH 82 dB SFDR AT 120 MHZ.
Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International, 7-9 Feb. 2000, p. 38 - 39.

[16] Op 't Eynde, F. and W. Sansen
A CMOS WIDEBAND AMPLIFIER WITH 800 MHZ GAIN-BANDWIDTH.
In: Proc. of the IEEE Custom Integrated Circuits Conference 1991, 12-15 May 1991, p. 9.1.1 - 9.1.4.

[17] Steyaert, M. and W. Sansen
OPAMP DESIGN TOWARDS MAXIMUM GAIN-BANDWIDTH.
In: Analog Circuit Design: Operational Amplifiers, Analog to Digital Convertors, Analog Computer Aided Design, p. 63 - 85, Edited by J.H. Huijsing, R.J. van der Plassche and W. Sansen, Dordrecht, Kluwer Academic Publishers, 1993, ISBN 0-7923-9288-4.

[18] Younis, A. and M. Hassoun
A HIGH SPEED FULLY DIFFERENTIAL CMOS OPAMP.
In: Proc. 43^{rd} IEEE Midwest Symp. on Circuits ans Systems, Aug 8-11 2000, p. 780 - 783.

[19] Sumanen, L. and M. Waltari, K.A.I. Halonen
A 10-BIT 200-MS/S CMOS PARALLEL PIPELINE A/D CONVERTOR.
IEEE Journal of Solid-State Circuits, Vol. 36 (2001), No. 7, p. 1048 - 1055.

[20] Furrer, B. and W. Guggenbühl
NOISE ANALYSIS OF SAMPLED-DATA CIRCUITS.
In: Proc. ISCAS, IEEE, April 1981, p. 860 - 863.

[21] Razavi, B.
DESIGN OF ANALOG CMOS INTEGRATED CIRCUITS.
New York, McGraw-Hill, 2001, ISBN 0-07-118815-0.

[22] Abo, A.M. and P.R. Gray
A 1.5-V, 10-BIT, 14.3-MS/S CMOS PIPELINE ANALOG-TO-DIGITAL CONVERTER.
IEEE Journal of Solid-State Circuits, Vol. 34 (1999), No. 5, p. 599 - 606.

[23] Aslanzadeh, H.A. and S. Mehrmanesh, M.B. Vahidfar, A.Q. Safarian
A LOW POWER 25MS/S 12-BIT PIPELINED ANALOG TO DIGITAL CONVERTER FOR WIRELESS APPLICATIONS.
In: IEEE Southwest Symposium on Mixed-Signal Design, 2003, p. 38 - 42.

[24] Young-Deuk Jeon and Byeong-Lyeol Jeon, Seung-Chul Lee, Sang-Min Yoo, Seung-Hoon Lee
A 12B 50 MHZ 3.3 V CMOS ACQUISITION TIME MINIMIZED A/D CONVERTER.
In: Proc. of the ASP-DAC 2000, p. 613 - 616.

[25] Yang, W. and D. Kelly, I. Mehr, M.T. Sayuk, L. Singer
A 3-V 340-mW 14-B 75-MSAMPLE/S CMOS ADC WITH 85-dB SFDR AT NYQUIST INPUT.
IEEE Journal of Solid-State Circuits, Vol. 36 (2001), No. 12, p. 1931 - 1936.

Appendices

Appendix A

Circuit implementations

In this appendix, the implementations of the analog circuits are inserted. Circuits that were not yet implemented are defined by a model.

All transistors are based on a standard transistor with $\frac{W}{L} = \frac{12\ \mu\text{m}}{0.24\ \mu\text{m}}$. In the figures, for each transistor, the multiplier M is given between square brackets. For example, when $M = 4$, the actual transistor size is $W = 4 \cdot 12\ \mu$m and $L = 0.24\ \mu$m.

A.1 Implementation of the basic block

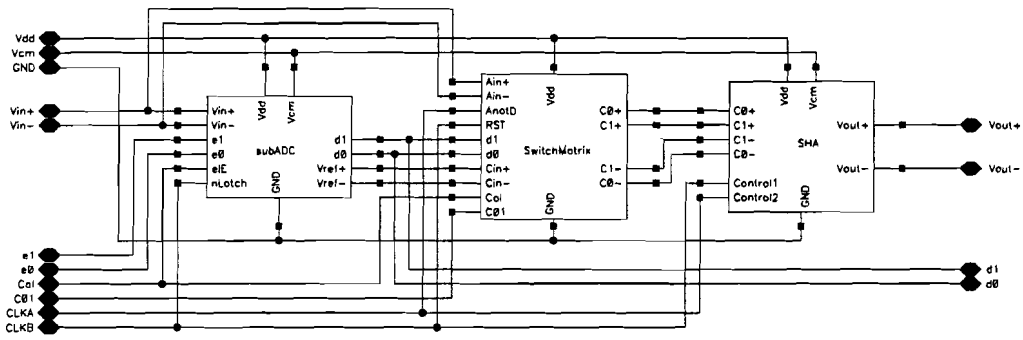


Figure A.1: *Implementation of the basic block.*

A.2 Model of the sub-ADC

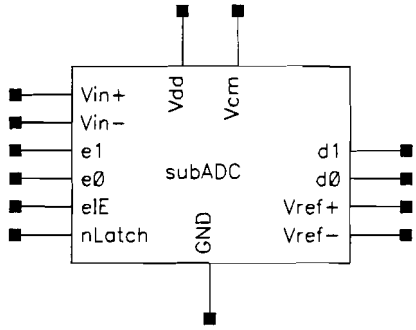


Figure A.2: *External view of the sub-ADC.*

Figure A.3 shows the model used for the sub-ADC. `eIE` selects whether the output `d1d0` equals `c1c0` (normal operation) or `e1e0` (calibration mode). The latch is transparent as long as `nLatch` is high. On the falling edge, data is latched until `nLatch` becomes high again. `Vref` is the reference level of the sub-ADC and equals:

$$\texttt{Vref} = R_{ADC} \cdot A_{max} \tag{A.1}$$

For calibration purposes, the reference voltage is available as an output signal of the sub-ADC as well.

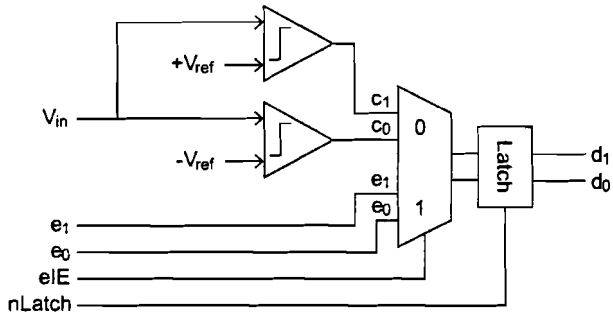


Figure A.3: *Model of the sub-ADC.*

A.3 Implementation of the switchmatrix

Figure A.4 shows the implementation of the switchmatrix. The model of the sub-DAC and the implementation of the switches are given in figure A.5. The functional descriptions of the switchlogic and the switchmatrix are given in table A.1 and A.2 respectively.

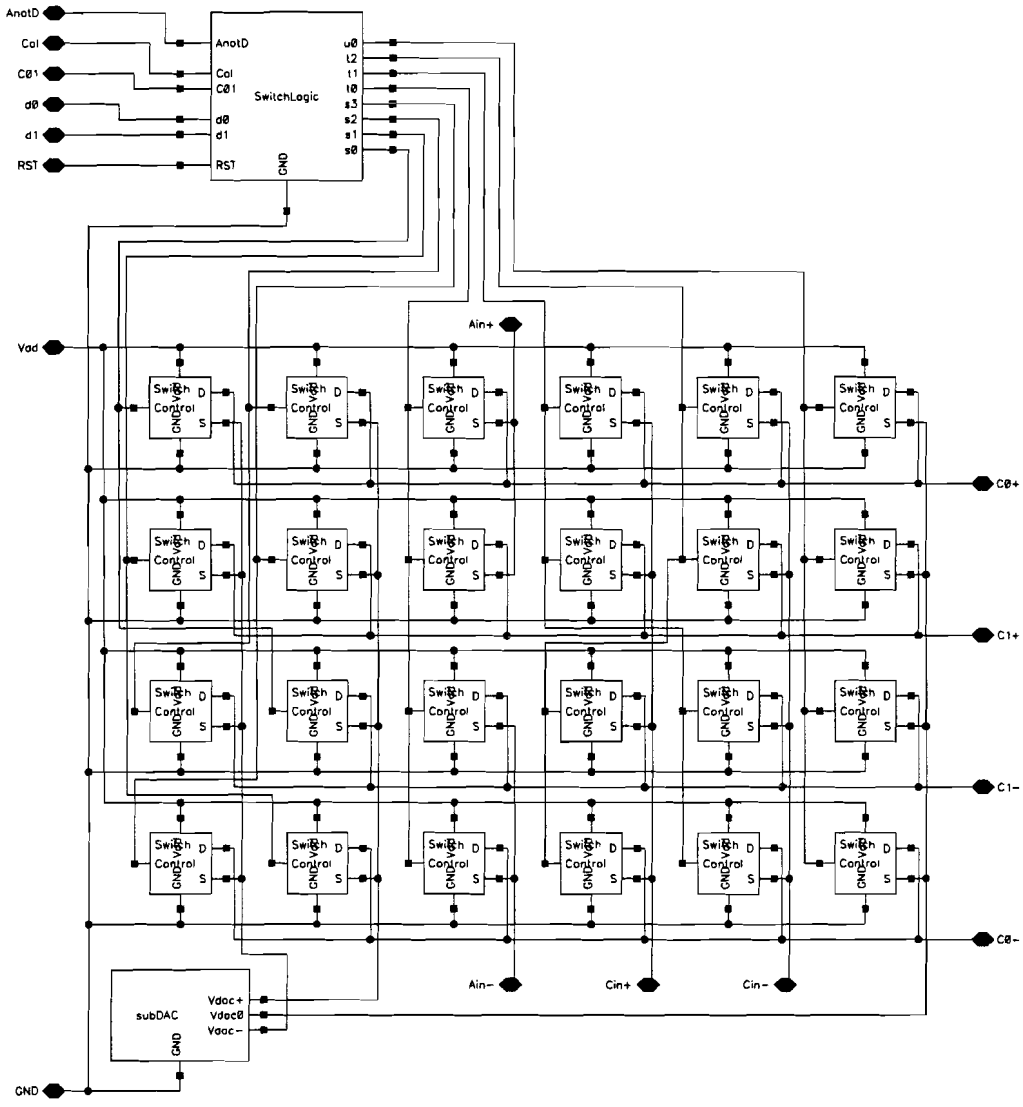


Figure A.4: *Implementation of the switchmatrix.*

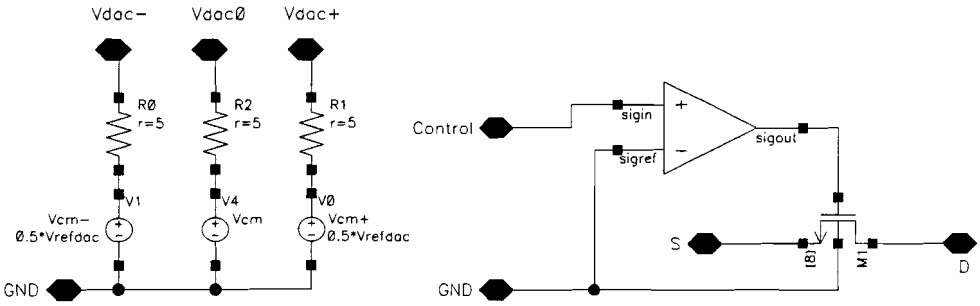


Figure A.5: *Model of the sub-DAC (left) and implementation of the switches (right). The comparator performs the level shift from logical levels to 0 V (low) and 4 V (high level).*

AnotD	RST	d1	d0	Cal	C01		s0	s1	s2	s3	t0	t1	t2	u0
0	0	0	0	x	x		1	1	0	0	0	0	0	0
0	0	0	1	x	x		1	0	0	1	0	0	0	0
0	0	1	1	x	x		0	0	1	1	0	0	0	0
1	x	x	x	0	x		0	0	0	0	1	0	0	0
1	x	x	x	1	0		0	0	0	0	0	1	0	0
1	x	x	x	1	1		0	0	0	0	0	0	1	0
0	1	x	x	x	x		0	0	0	0	0	0	0	1

Table A.1: *Functional description of the switchlogic.*

s0	s1	s2	s3	t0	t1	t2	u0		C0+	C1+	C1-	C0-
1	1	0	0	0	0	0	0		Vdac-	Vdac-	Vdac+	Vdac+
1	0	0	1	0	0	0	0		Vdac-	Vdac+	Vdac+	Vdac-
0	0	1	1	0	0	0	0		Vdac+	Vdac+	Vdac-	Vdac-
0	0	0	0	1	0	0	0		Ain+	Ain+	Ain-	Ain-
0	0	0	0	0	1	0	0		Cin+	Cin+	Cin-	Cin-
0	0	0	0	0	0	1	0		Cin-	Cin-	Cin+	Cin+
0	0	0	0	0	0	0	1		Vcm	Vcm	Vcm	Vcm

Table A.2: *Output of the switchmatrix as selected by the switchlogic.*

A.4 Implementation of the sample-and-hold amplifier

Figure A.6 shows the sample-and-hold amplifier. C_{std} equals 1.2 pF, the block indicated with **FS9** is the operational amplifier.

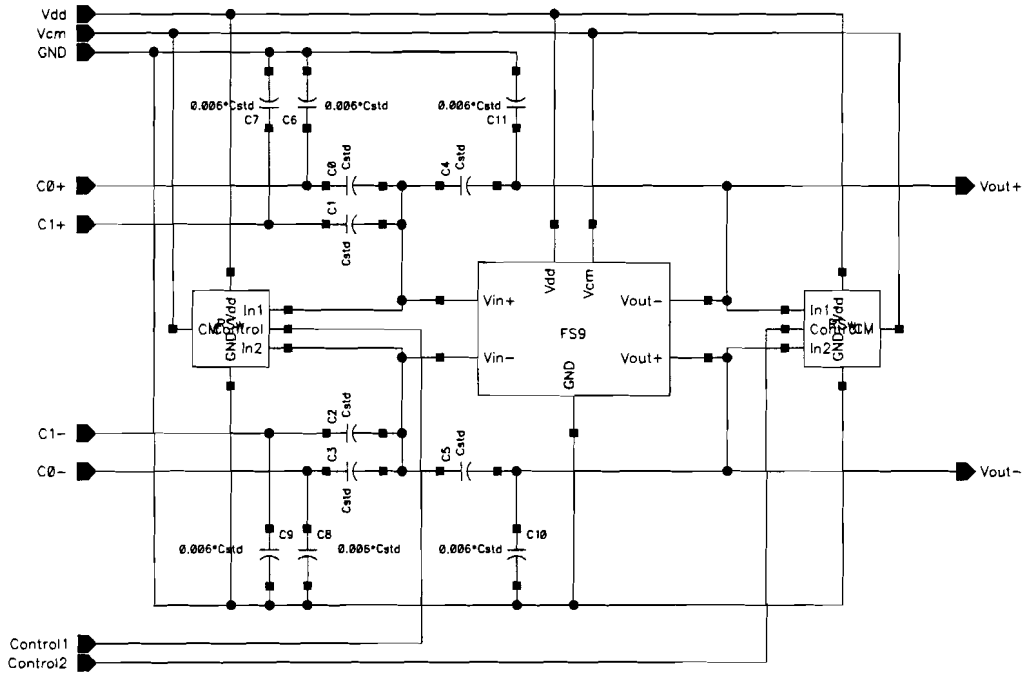


Figure A.6: *Implementation of the SHA.*

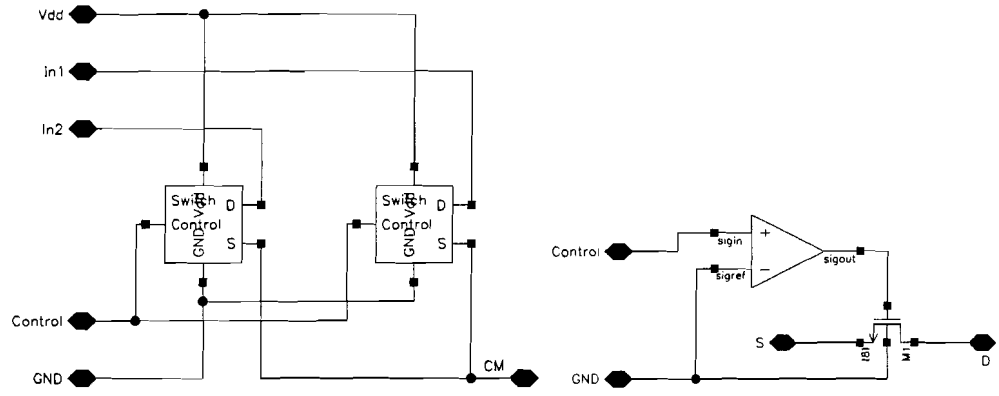


Figure A.7: *The reset switches (left) are composed of two identical switches (right).*

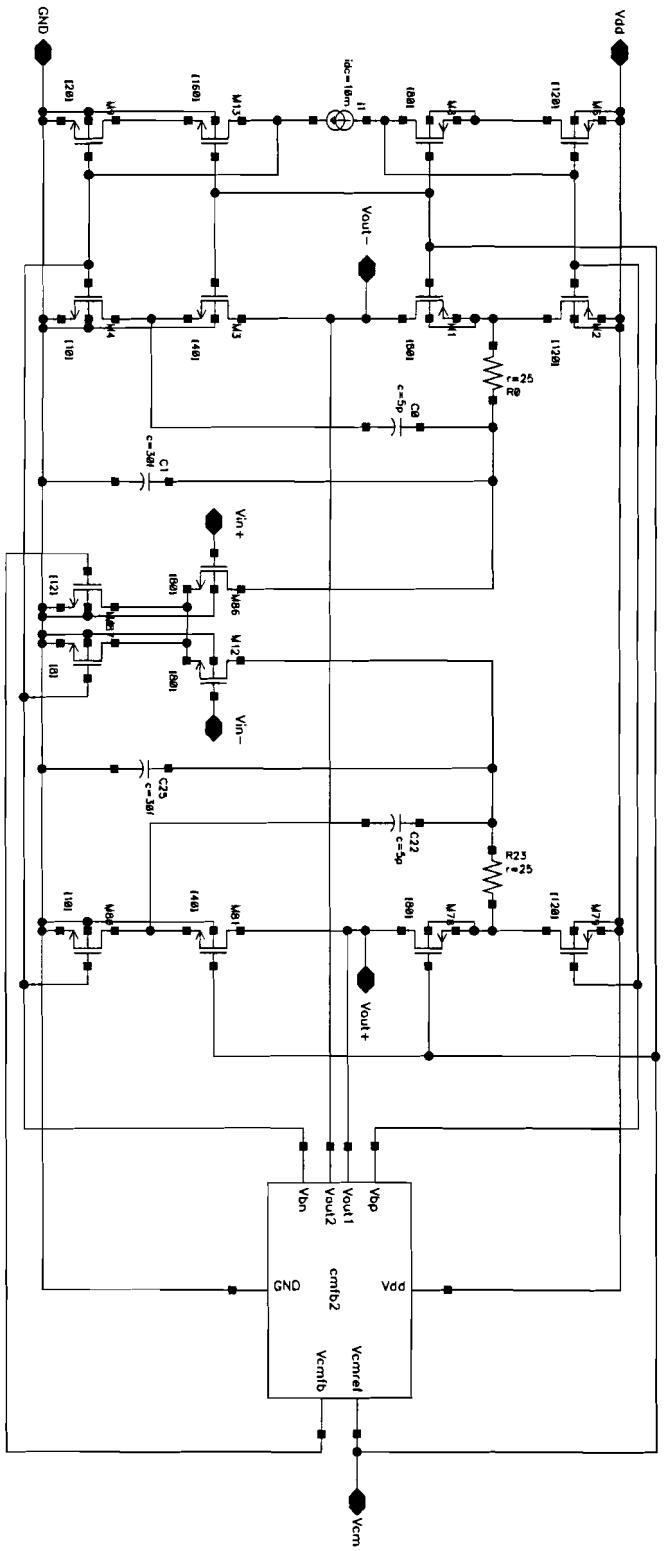


Figure A.8: *Implementation of the opamp.*

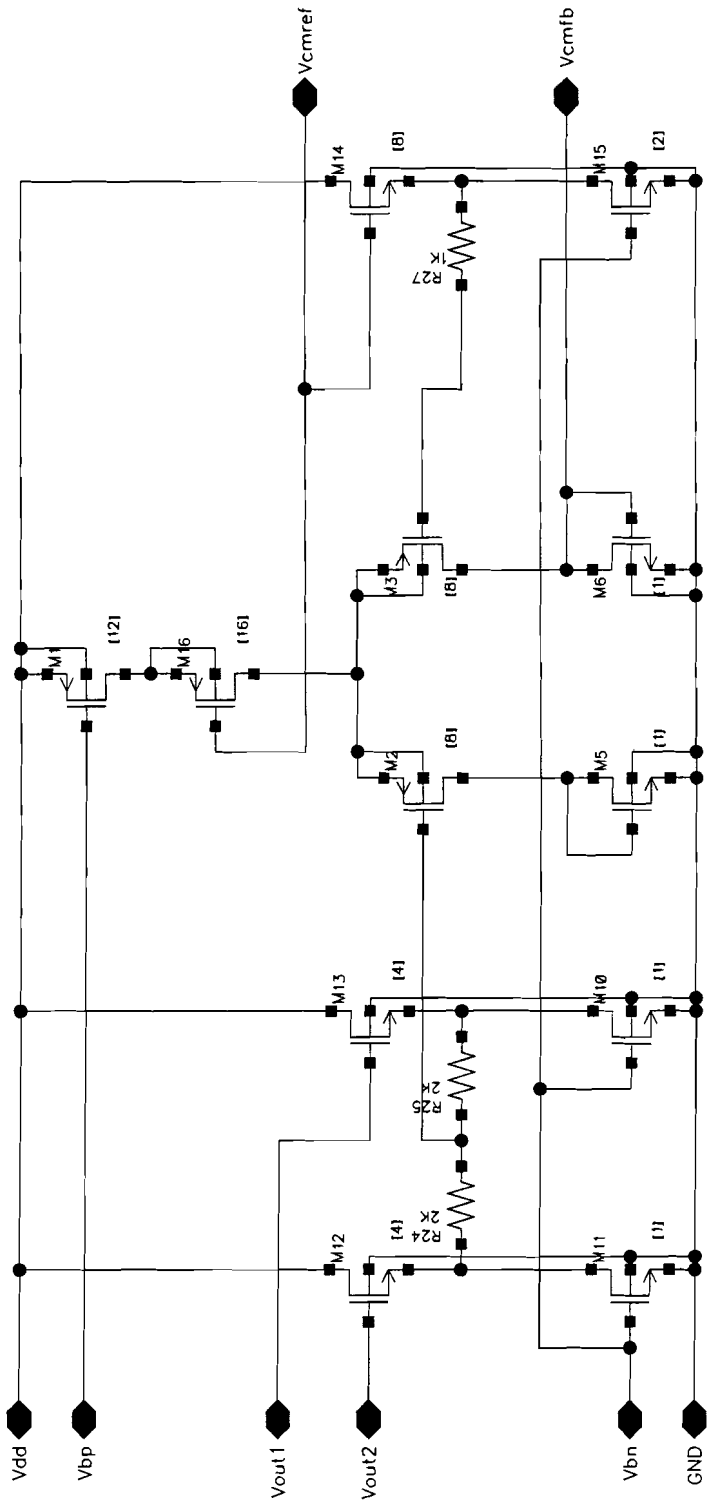


Figure A.9: *Implementation of the common-mode feedback of the opamp.*

Appendix B

Simulation results

In this appendix, simulation results for the opamp and for the SHA are presented. Also, some important parameters are derived from these results.

Unless stated otherwise, simulations are performed with $V_{dd} = 2.5\ V$ *and* $T = 300\ K$.

B.1 Opamp

B.1.1 DC analysis

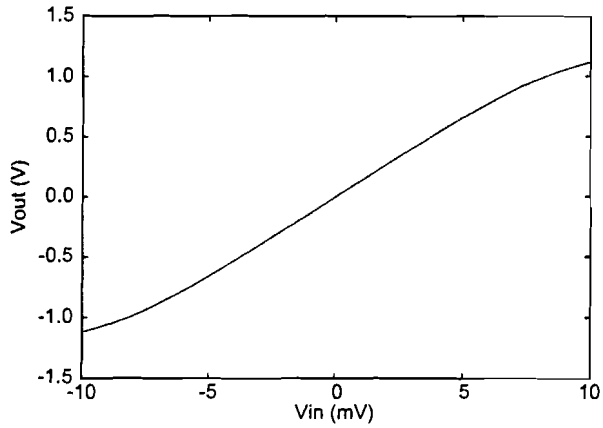


Figure B.1: *Open loop DC response of the opamp.*

Derived parameters

From the DC curve, the non-linear transfer curve of the opamp was determined:

$$V_{out} = A_0(V_{in} - V_{off}) + A_2(V_{in} - V_{off})^2 + A_3(V_{in} - V_{off})^3 \text{ , with} \quad \text{(B.1)}$$

$$A_0 = 135.6 \qquad A_2 = 0\ \text{V}^{-1} \qquad A_3 = -1.6 \cdot 10^4\ \text{V}^{-2} \qquad V_{off} = 0\ \text{V} \quad \text{(B.2)}$$

B.1.2 AC analysis

This analysis was performed with an external load capacitance of 1.6 pF, representing the switched-capacitor network of the SHA.

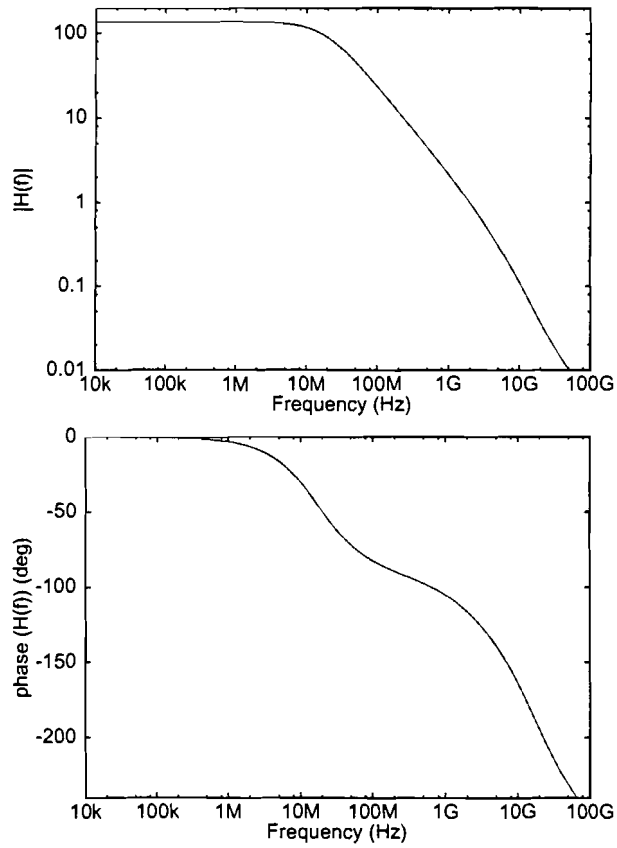


Figure B.2: *Open loop AC response (magnitude and phase) of the opamp.*

Derived parameters

DC gain	A_0	136
3 dB bandwidth	f_{3dB}	17.4 MHz
gain-bandwidth product	GBP	2.36 GHz
phase margin	PM	63.9 ^{o}

B.1.3 Mismatch and sensitivity analysis

The sensitivity of the opamp to deviations of temperature, power supply voltage and mismatch of transistors, resistors and capacitors was analyzed. As neither a Monte Carlo analysis nor a worst case analysis was available, an alternative method was used. First, it is described how this method determines the sensitivity to each

of these parameters (temperature, voltage and process spread). Then, simulation results are shown, which take into account a single deviation of a single component at a certain time only. Finally, the result for a worst case combination of deviations is given.

Sensitivity to power supply and ambient temperature

The sensitivity of the opamp to the power supply voltage was verified by applying either $\pm 10\%$ of the nominal value (2.5 V) to the opamp. The influence of the ambient temperature was determined at the extremes of $T = 0$ °C and $T = 85$ °C.

Sensitivity to deviations in resistors and capacitors

For the resistors and capacitors, two types of deviations have been investigated. First, the influence of an absolute deviation of all components was investigated. Second, the influence of mismatch between individual components was verified. According to the documentation of the UMC process, the absolute deviation of resistors can be as much as $\pm 40\%$ and for capacitors $\pm 20\%$. The mismatch between two nominal identical components is given by its standard deviation $\sigma_R = 0.7\%$ for resistors and $\sigma_C = 0.1\%$ for capacitors.

Sensitivity to mismatch of transistors

The mismatch of transistors is composed of two parts: deviations of the threshold voltage and deviations of the drain-source current. Both types of deviations are modelled by a normal distribution with standard deviations σ_{V_t} (mV) and $\sigma_{I_{ds}}$ (%) respectively. The first type was verified by inserting a voltage source in series with the gate of each transistor, the second type by varying the width W of each transistor individually. Taking into account the UMC process documentation, we obtain $\sigma_{V_t} =$ 3 mV and $\sigma_{I_{ds}} = 1$ % for the standard-size transistor ($\frac{W}{L} = \frac{12\ \mu\text{m}}{0.24\ \mu\text{m}}$). For all other transistors (which have a multiplier of at least 4 in our case), the deviations are estimated by $\sigma_{V_t} = 1.5$ mV and $\sigma_{I_{ds}} = 1$ %, regardless of the exact transistor size. This estimation (which is quite pessimistic) is done as the UMC documentation seems to be unreliable for these large-size transistors.

Simulation results

The DC and AC analyses were repeated, while one by one each parameter (supply voltage, temperature, etc.) was adjusted to one of its extreme values. For the power supply voltage, this means simulating at 2.25 V and 2.75 V. The temperature was adjusted to either 0 °C or 85 °C. The influence of mismatch of resistors, capacitors and transistors was verified using a deviation of plus and minus 3σ. The absolute deviations of resistors and capacitors was verified by scaling all resistors at the same time with either plus or minus 40%. Likewise, all capacitors were scaled with $\pm 20\%$. All simulations consider a single deviation at a certain time only. The results of all simulations are combined in table B.1. It shows for all derived parameters (A_0, GBP, etc) the typical value (without mismatch) and its minimum and maximum value, according to the simulations. Also, the source is indicated for which the derived parameter is the most sensitive.

Parameter	Min	Typ	Max	Critical source
V_{off} (mV)	-4.5	0.0	4.5	transistors M_{dif}, M_p (in the opamp)
A_0	116	136	148	V_{dd}
A_2 (V^{-1})	-45	0	45	mismatch between R_{24}, R_{25} (in the CMFB circuit)
A_3 (V^{-2})	$-2.0 \cdot 10^5$	$-1.6 \cdot 10^5$	$-1.2 \cdot 10^5$	V_{dd}
GBP (GHz)	2.0	2.4	2.6	temperature T
PM (^{o})	63	64	65	absolute deviation of R and C of the pole-zero cancellation circuit (in the opamp)

Table B.1: *Minimum, typical and maximum values of the derived parameters due to process spread and environmental deviations. Also, the source for which the parameter is the most sensitive is indicated.*

To verify the stability of the operational amplifier under worst case circumstances, all parameters of all components described in this section are chosen such that the minimum phase margin is achieved. Table B.2 lists the settings for this analysis. Note that this simulation is extraordinarily pessimistic. Nevertheless, the achieved phase margin is still 60.5^o, ensuring good stability.

V_{dd}	2.75 V
T	85 ^{o}C
Resistors	-40% absolute deviation and $\pm 3\sigma_R$ relative deviation
Capacitors	$+20\%$ absolute deviation and $\pm 3\sigma_C$ relative deviation
Transistors	$\pm 3\sigma_{V_t}$ and $\pm 3\sigma_{I_{ds}}$ relative deviation

Table B.2: *Applied combination of deviations to check the worst-case phase margin.*

B.2 Sample-and-hold amplifier

B.2.1 Transient behavior

An example of the settling behavior of the SHA is given below. The sample frequency is 100 MHz. The applied input voltage follows the sequence -1 V, -1 V, $+1$ V, $+1$ V, $\cdots$. During the *sample* phase, the output resets to zero, during the *hold* phase, the output settles to the new residue value. There is a small difference between values V_1 and V_2, as the previous sample has still some influence on the current sample. However, the difference is always less than $\frac{1}{2}V_{lsb}$, thus acceptable. The settling time of the SHA (within $\frac{1}{2}V_{lsb}$ of the final value) equals 3.6 ns.

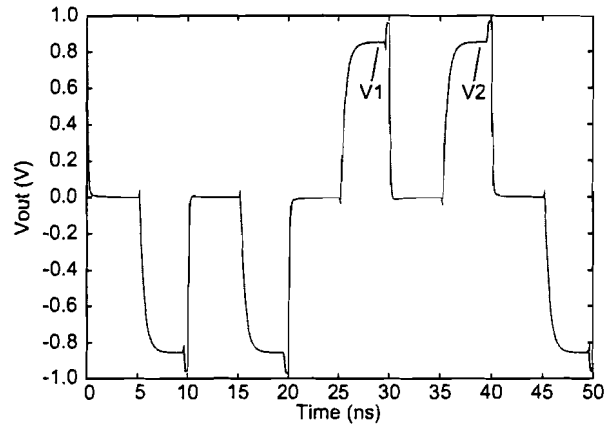


Figure B.3: *Example of the transient behavior of the SHA.*

B.2.2 Transfer function

With a transient simulation the transfer function of the SHA was derived (figure B.4, left). From this picture, the closed loop gain $A_0 = 1.8647$ and the closed loop non-linearity $A_3 = 3.3 \cdot 10^{-2}$ were derived.

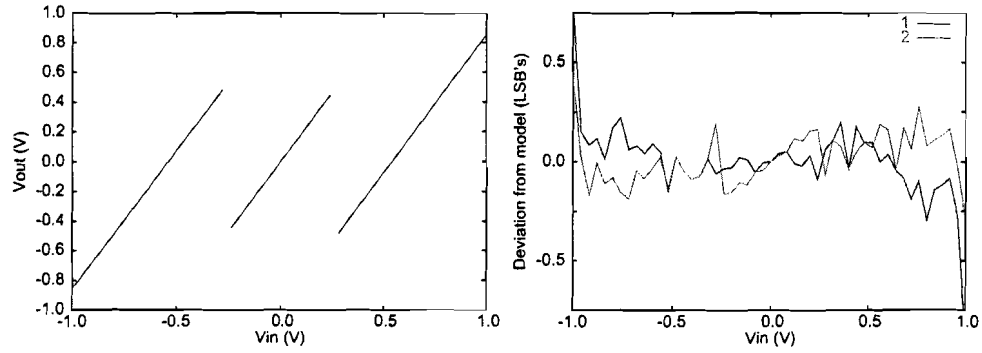


Figure B.4: *Transfer function of the SHA (left) and deviation of the transfer function from the model (right). The difference between '1' and '2' is due to the memory effect of the SHA (see also figure B.3): '1' is measured when the current input signal is V_x while the previous input signal was $-V_x$. '2' is measured when both the current and the previous input signal equal V_x.*

The difference between the model and the actual simulation result is plotted in figure B.4, right side. It shows that within the normal input range (±0.8 V) the deviation is less than $\frac{1}{4}V_{lsb}$, even taking the memory effect (described in the previous section) into account. Therefore, the simple model gives a correct description of the SHA.

B.2.3 Noise analysis

During the hold phase, continuous-time noise is present at the output of the SHA. The noise is produced by both the operational amplifier and the switches in the

switched-capacitor network. Figure B.5 shows the noise power density and the integrated noise power at the output of the SHA. The integrated noise power equals $1.35 \cdot 10^{-8}$ V^2.

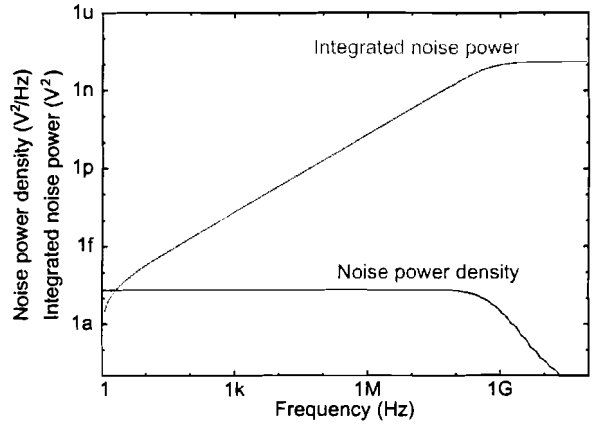


Figure B.5: *Noise power density of the SHA during the hold phase.*

B.2.4 Mismatch and sensitivity analysis

The sensitivity of the SHA to deviations of temperature, power supply voltage and mismatch of components in the opamp was analyzed. The variation of parameters is exactly equal to the variation during the sensitivity analysis of the opamp.

As the settling time of the SHA (within $\frac{1}{2}V_{lsb}$) is the most important parameter of the SHA, the input voltage is fixed to 0.95 V (then $V_{out} \approx A_{max}$) and the settling time is measured. When only a single deviation at a time is considered, the slowest settling time is around 4.1 ns (the nominal value is 3.6 ns). The settling time is most sensitive to deviations of the temperature T and absolute deviations of the resistors and capacitors in the pole-zero cancellation circuit inside the opamp. A worst-case scenario was tested as well. All parameters are adjusted at the same time such as to achieve the slowest settling time. The settings are given table B.3. The achieved settling time is $\tau_{set} = 4.4$ ns.

V_{dd}	2.25 V
T	85 ^{o}C
Resistors	+40% absolute deviation and $\pm 3\sigma_R$ relative deviation
Capacitors	+20% absolute deviation and $\pm 3\sigma_C$ relative deviation
Transistors	$\pm 3\sigma_{V_t}$ and $\pm 3\sigma_{I_{ds}}$ relative deviation

Table B.3: *Applied combination of deviations to check the worst-case settling time.*

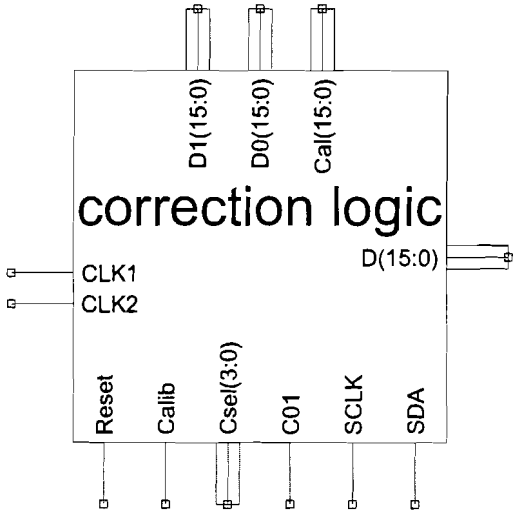
Appendix C

On-chip correction logic circuits

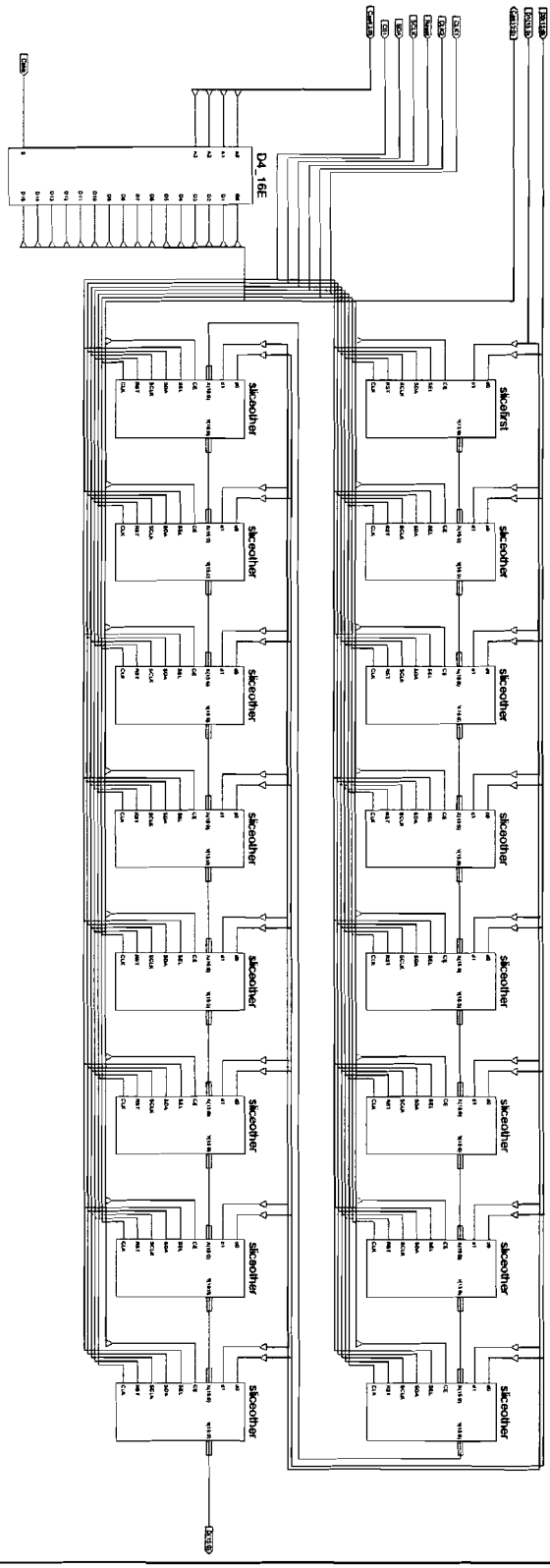
This appendix contains the circuit implementations of the on-chip digital correction logic, according to section 5.2.

C.1 Main circuit

External view



Internal view

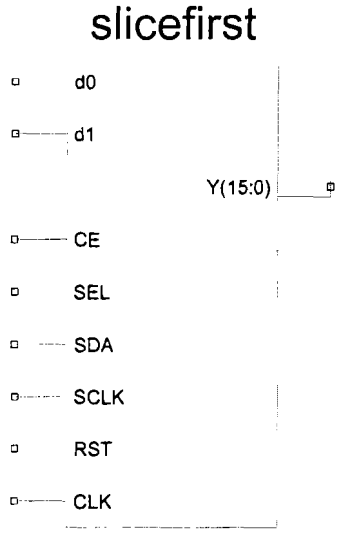


Functional description

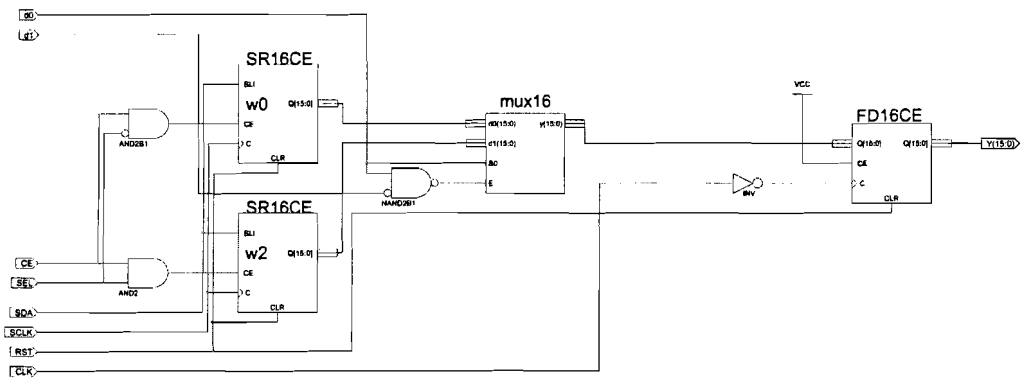
- D_1(15:0) and D_0(15:0): inputs coming from the analog basic blocks. D_1(i) and D_0(i) are coming from the sub-DAC in stage i. The stages are numbered from 0 (first stage in the pipeline) to 15 (last stage in the pipeline),
- Cal(15:0): outputs to the analog blocks. Cal(i) is connected to stage i. If Cal(i) is high, stage i is set in calibration mode. When all Cal(i) signals are low, the converter is in normal operation mode,
- CLK1 and CLK2: clock input signals for synchronization,
- D(15:0) is the corrected 16-bit signed output signal of the pipelined converter,
- Reset: input, resetting all weights and registers in the correction logic to zero,
- Calib: input, enabling calibration. When Calib is low, no calibration takes place, the internal weights are fixed and the Cal(15:0) outputs are disabled,
- C_{sel}(3:0): input, selecting the stage to be calibrated,
- C_{01}: input, selecting the weight (ω_0 or ω_2) to be altered,
- SCLK and SDA: serial clock and data line to transfer data from the measurement algorithm to the active weight in the correction logic. The data SDA is clocked on the rising edge of SCLK with the MSB first. The active weight is set by C_{sel}(3:0) and C_{01}. If Calib is low, the serial data line is disabled.

C.2 Slicefirst sub-circuit

External view



Internal view

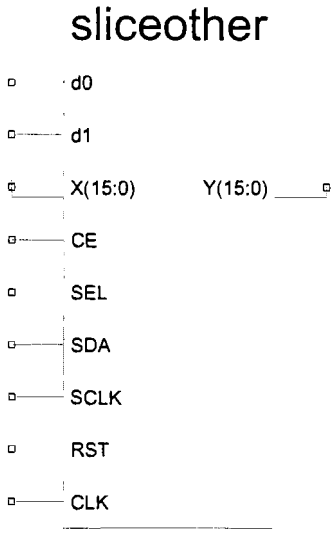


Functional description

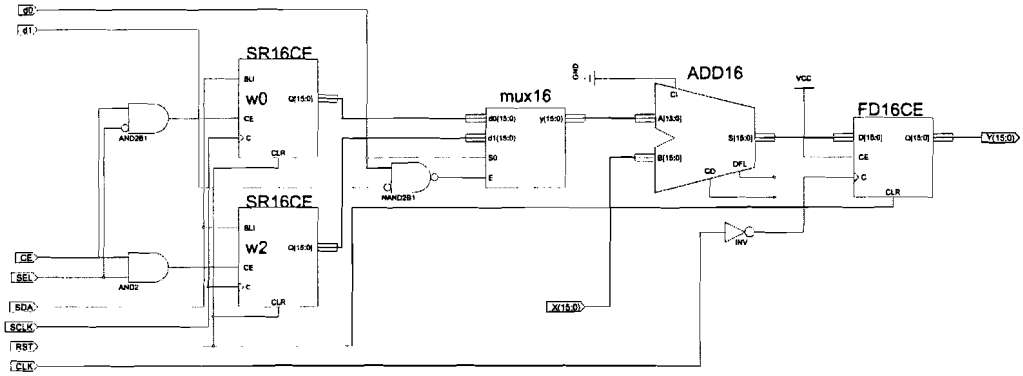
The functional description of the slicefirst block is almost the same as the functional description of the sliceother block (section C.3). The only difference is that for slicefirst, input `X(15:0)` is fixed to 0.

C.3 Sliceother sub-circuit

External view



Internal view



Functional description

This circuit contains two 16-bit registers to store the weights ω_0 and ω_2 for one of the stages in the pipeline. With `CE` (calibration enable) and `SEL` (select), a weight can be selected for programming:

inputs			programming enabled	
CE	SEL		ω_0	ω_2
0	x		No	No
1	0		Yes	No
1	1		No	Yes

[1]

The active weight ω is programmed by the serial interface (`SCLK` and `SDA`), with the MSB first:

SCLK	SDA		$\omega(0)$	$\omega(15:1)$
$\uparrow$	B		B	$\omega(14:0)$

[2]

Signal `S(15:0)` (the output of the ADD16-block), is dependent on inputs `X(15:0)`, d_1 and d_0:

X(15:0)	d_1	d_0		S(15:0)
C	0	0		$C + \omega_0$
C	0	1		C
C	1	1		$C + \omega_2$

The output signal `Y(15:0)` is the synchronized version of `S(15:0)`:

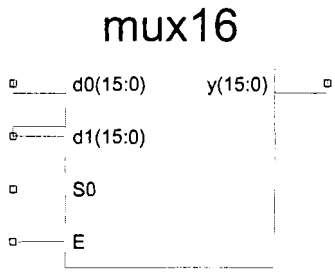
[1] 'x' is a don't care signal.
[2] '$\uparrow$' is a low-to-high transition.

CLK		Y(15:0)
↓		S(15:0)
0, 1, ↑		No change

With the asynchronous reset signal (RST), the internal weights and the output register are set to 0.

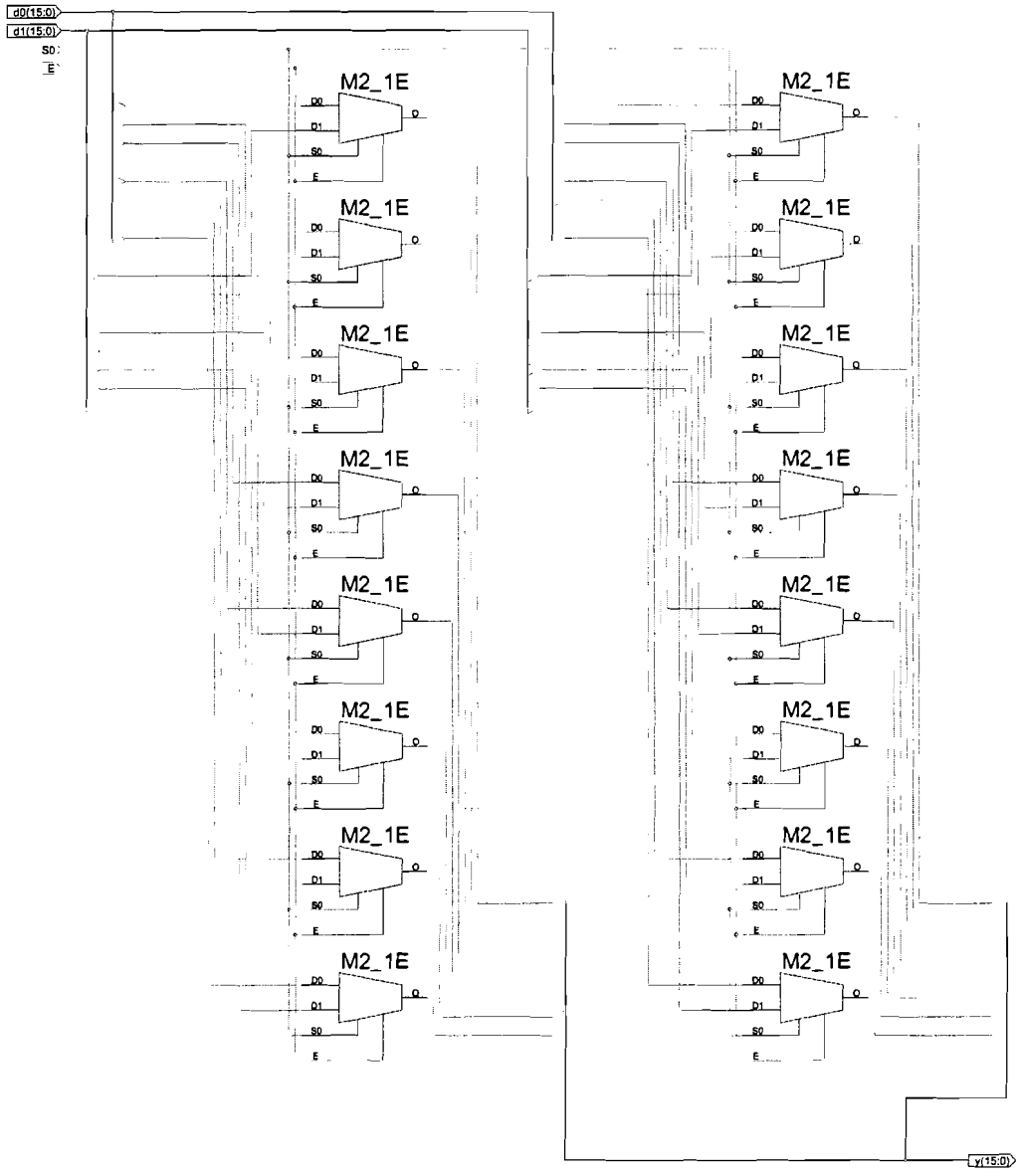
C.4 Mux16 sub-circuit

External view



Internal view

See next page.



Functional description

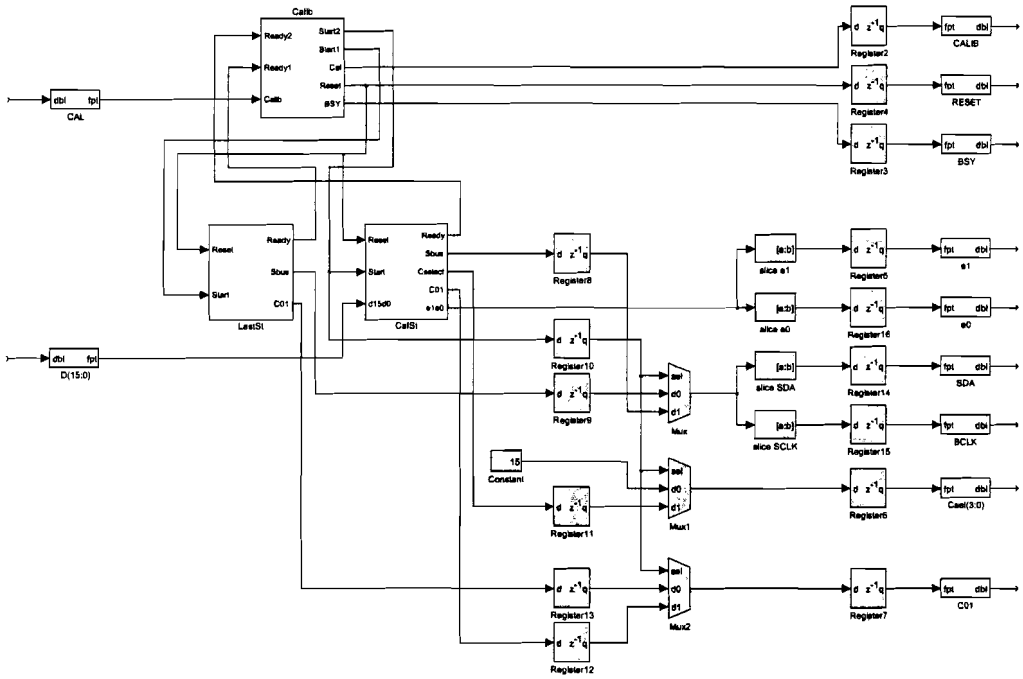
E	S0	D1(15:0)	D0(15:0)		Y(15:0)
0	x	x	x		0
1	0	x	$B(15:0)$		$B(15:0)$
1	1	$C(15:0)$	x		$C(15:0)$

Appendix D

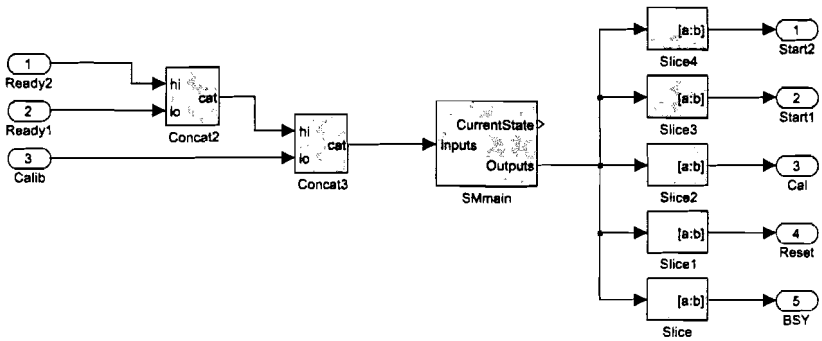
Off-chip measurement algorithm circuits

This appendix contains the circuit implementations of the measurement algorithm, according to section 5.3. The circuit is designed in Matlab Simulink. With Xilinx System Generator, this can be extracted directly to FPGA code.

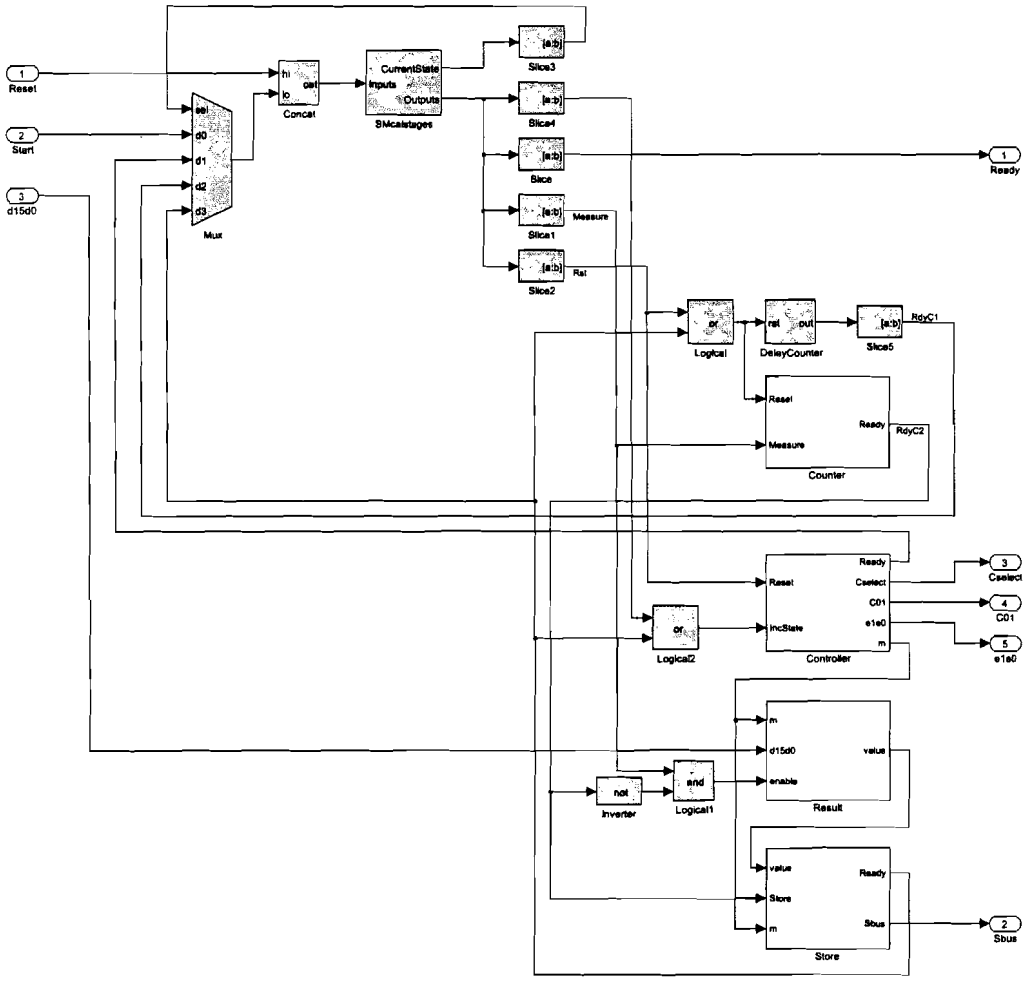
D.1 Main circuit



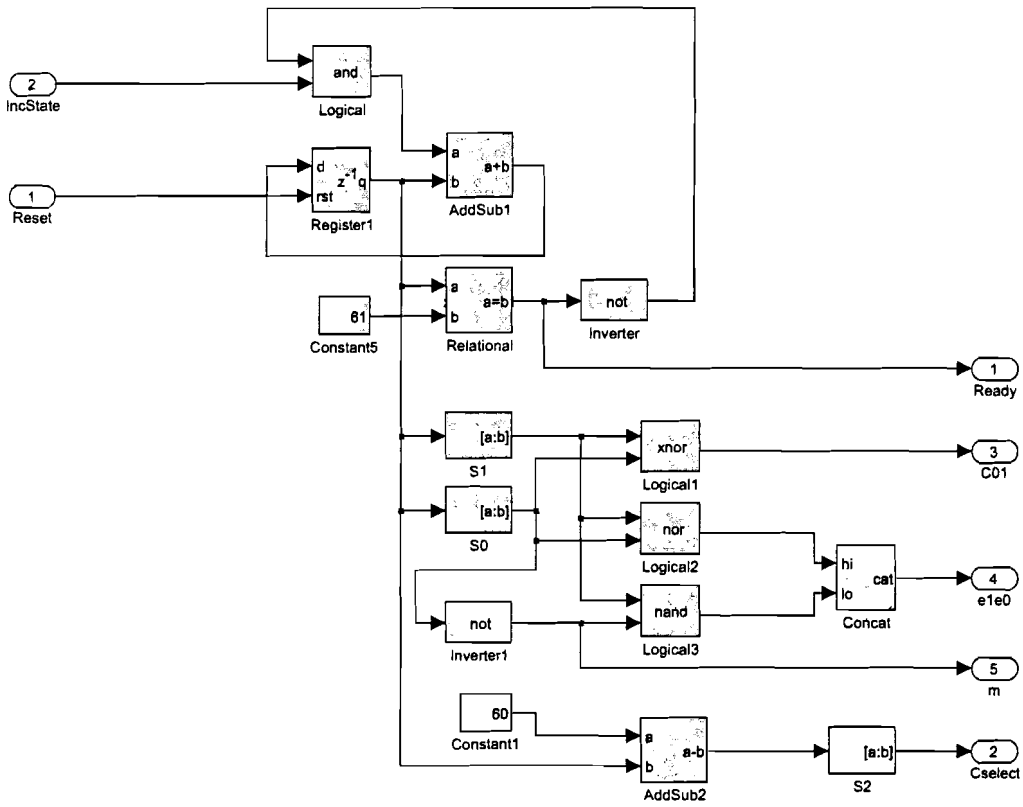
D.2 Calib sub-circuit



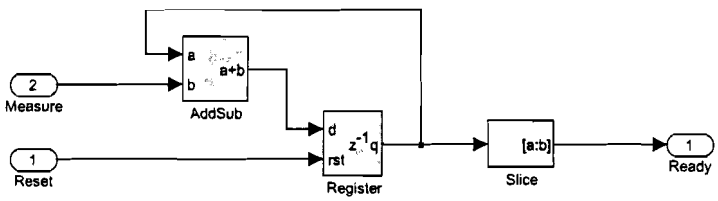
D.3 CalSt sub-circuit



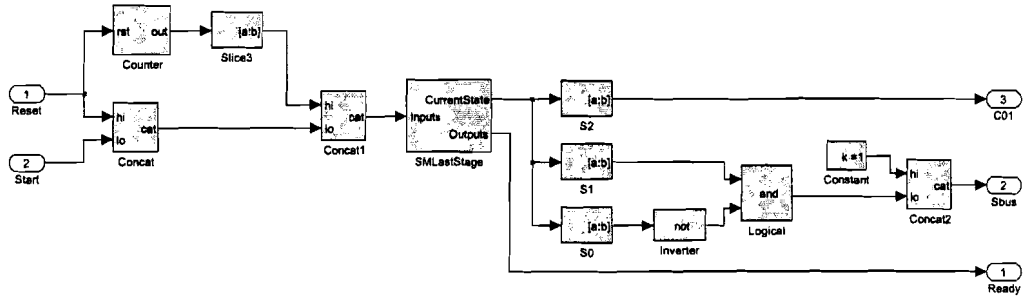
D.4 Controller sub-circuit



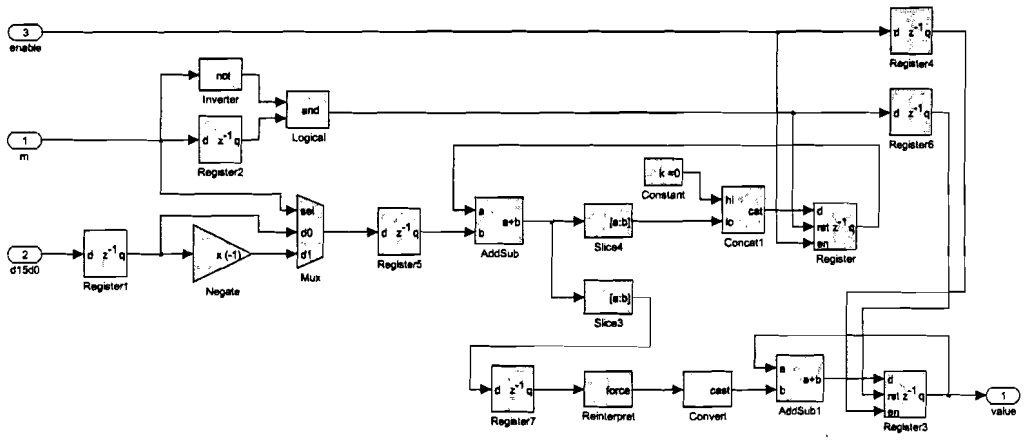
D.5 Counter sub-circuit



D.6 LastSt sub-circuit



D.7 Result sub-circuit



D.8 Store sub-circuit