NTNU – Trondheim
Norwegian University of
Science and Technology

Modeling and Design of a Dual-Residue Pipelined ADC in 130nm CMOS

Eirik Steen-Hansen

Master of Science in Electronics
Submission date: June 2012
Supervisor: Trond Ytterdal, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Master project description

Modeling and Design of a Dual-Residue Pipelined ADC in 130nm CMOS

The objective of this project is to model and design a 50MS/s, 9-bit dual-residue pipelined ADC in a 130nm CMOS technology.

The project consists of the following tasks:

- Perform a literature survey of pipelined ADCs to establish current-state-of-the-art
- Analyze and compare published work
- Develop a behavioral model of the ADC and use the model for deriving specifications of the individual blocks
- Design key blocks at transistor level and study the effects of non-idealities by analytical methods and simulation

Acknowledgements

I would like to thank my supervisor Trond Ytterdal for his support, and valuable suggestions throughout the work on this thesis. I would also like to thank Stig Viste for feedback on the report.

Abstract

A 9-bit $50MS/s$ dual-residue pipelined ADC is modeled and analyzed. The first stage is a modified pipelined ADC stage, while the other stages uses an interpolator to resolve the signals, the focus is on designing these stages. The dual-residue architecture is insensitive to the gain of the residue amplifiers, and only a matching between two amplifiers is necessary. Limiting parameters of the ADC is the offset in the residue amplifiers, as well as gain mismatch between the amplifiers. The maximum allowed offset voltage of the residue amplifier is $\frac{V_{lsb}}{2}$, and maximum allowable mismatch between the two residue amplifiers is $\frac{1}{256}$ for a 9-bit ADC. Multiple amplifier topologies were discussed and the best candidate for residue amplification is found to be a zero-crossing based amplifier. With this type of amplifier the last 8 stages of the ADC has an estimated power consumption of $2.1m$W.

Sammendrag

En 9-bit $50MS/s$ dual-residue pipelined ADC har blitt modelert og analysert. Det første trinnet er et modifisert pipeline trinn mens de andre bruker en interpolator for å detektere signalet. Dual-residue arkitekturen er uavhenge av gain i residue forsterkeren og bare en matching av forsterkningen i to forsterkere er nødvendig. Begrensende parametere for ADCen er offset i residue forsterkerene og mismatch mellom de to forsterkerene i hvert trinn. Den største offset spenningen forsterkeren kan ha er $\frac{V_{lsb}}{2}$, og den maksimale mismatchen er $\frac{1}{256}$ for en 9-bits ADC. Flere forsterker topologier ble diskutert og den beste kanditaden for residue forsterkningen er en zero-crossing bassert forsterker. Med denne forsterkeren brukte de siste 8 trinnene i ADCen $2.1m$W.

Contents

1 Introduction

The increasing demand for low power analog to digital converters push the existing topologies to the limits. With the new deep-submicron technologies the supply voltage and intrinsic gain of a transistor decreases. Pipelined converters are a good choice in order to decrease the power consumption and total area for high-speed, low power ADCs. The traditional approach for pipelined converters uses a high gain OTA with feedback to generate an accurate amplification between the stages. These amplifiers has a high power consumption, and becomes increasingly harder to design as the intrinsic gain decreases with newer technologies. In [Mangelsdorf et al., 1993] another approach for solving this problem where introduced. By adding another residue voltage the necessity of an accurate gain is reduced to a demand of matching the gain of two amplifiers, which is a easier goal to reach.

In this thesis such a converter is modeled and the advantages and disadvantages are discussed. The ADC will be modeled in a $130n$m CMOS process and all high level models are written in Verilog-A. The report is divided into 5 parts, first this short introduction before continuing with the background theory for the converter. In section 3 the different design choices are presented and are discussed in section 4.

2 Theory

2.1 Pipeline ADC

A single 1-bit stage of a conventional pipelined ADC is shown in Figure 1. These stages are placed in series, where each stage resolve 1-bit, which is combined to the final output code. A flash-stage compares the input voltage to the reference and resolves the output bit of the stage. This bit is further sent into a DAC which again is fed into an adder. This adder subtracts the DAC voltage from the input, and generates a residue voltage. This voltage is amplified and sent to the next stage.

In a conventional pipelined ADC all components in a single stage needs to be as accurate as the total resolution of the remaining stages. The only exception is the comparators s which can contain an error of $\frac{V_{lsb}}{2}$ if a 1.5-bit architecture and error correction is used. [Lewis et al., 1992]

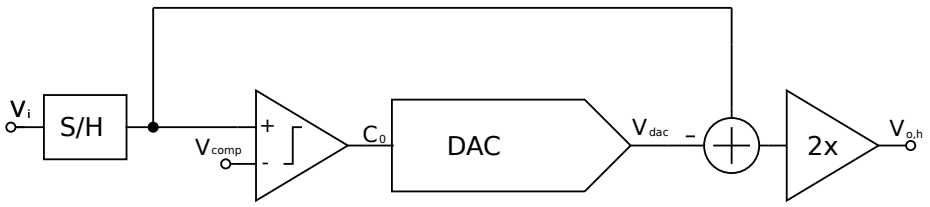


Figure 1: Single 1-bit pipeline stage

The amplifier is usually implemented as a high gain OTA with feedback to produce an accurate gain of 2.[Shen and Kinget, 2008] The advantages of a pipelined architecture compared to a normal flash ADC are reduced size and power consumption, while still attaining the same throughput, but with a small delay. Disadvantages is that the amplifier consumes a lot of power and in newer processes with low intrinsic gain may be hard to implement.

Typical error sources in a pipelined ADC are inaccurate gain between the stages, incomplete settling, input offset in the amplifiers and charge injections from switches in switched capacitor circuits[Lee and Geiger, 1999, Taherzadeh-Sani and Hamoui, 2006]

2.2 Dual-residue

By introducing an extra residue voltage to the conventional pipelined ADC, a few of the dominant error sources can be completely removed, most importantly the need for an accurate gain. [Mangelsdorf et al., 1993] This is achieved by changing the representation of the signal from an exact voltage, to the position of the zero-crossing between a positive and negative voltage. The conversion between these representations is illustrated in Figure 2, a 1-bit per stage architecture is assumed. In Figure 2a an input voltage of approximately $\frac{7}{10}V_{ref}$ is shown. The first output voltage, $V_{o,high}$ is identical to the residue voltage in a conventional pipelined ADC, namely the difference between the input voltage and the nearest lower quantization level, in this case $\frac{V_{ref}}{2}$. The second residue voltage,$V_{o,low}$ is the difference between the input signal and the higher quantization level, in this case V_{ref}. These two voltages are shown in Figure 2b, this is the dual-residue representation which will be transmitted to the next stage in the pipeline.

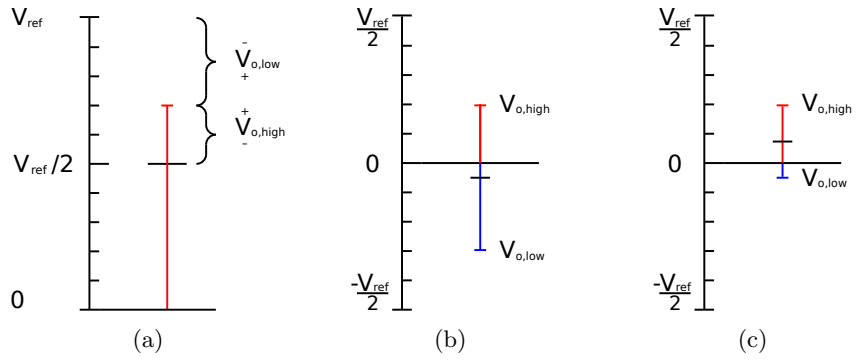


Figure 2: Dual-residue representation and operation, the outputs are not amplified. (a) Input voltage of appr 7/10Vref. (b)Dual-residue representation after the first bit is removed. (c) The second bit is resolved and zoomed in on the zero-crossing.

The way these voltages are generated it's easy to show that $V_{o,high}$ always will be positive, while $V_{o,low}$ always will be negative. This representation is dimensionless and independent of any reference voltage and therefore only the

first stage in a dual-residue ADC will have reference voltages. In the ADC the first stage will resolve the first bit as well as perform the transformation, hence this stage will differ from all the following stages, which will use interpolation to zoom in on the zero-crossing. [Vecchi et al., 2011]

2.2.1 Converter stage

The converter stage will be the first stage in the ADC and is shown in Figure 3. This stage will transform the input signal to the dual-residue representation, as well as resolve the first bit. This stage is quite similar to a conventional pipeline stage, but with a few important modifications. An extra output amplifier and subtracter are added as well as an extra output voltage from the DAC. This is used to generate the second residue voltage. The possible residue voltages for both outputs are shown mathematically in Table 1. The reference level of the comparator is assumed to be $\frac{V_{ref}}{2}$. If the input voltage is above the comparator level, as is the case in Figure 2a, the output code of the comparator is 1. The high output voltage will be the input voltage subtracted the lower quantization level, which is $\frac{V_{ref}}{2}$ here, while the low output voltage will be the input subtracted the high quantization level which is V_{ref} here. An amplifier is used at the output to charge the sampling capacitors in the next stage, even though no exact amplification is needed, a gain of 2 would keep the difference in the two output voltages constant between the stages and is the recommended value.

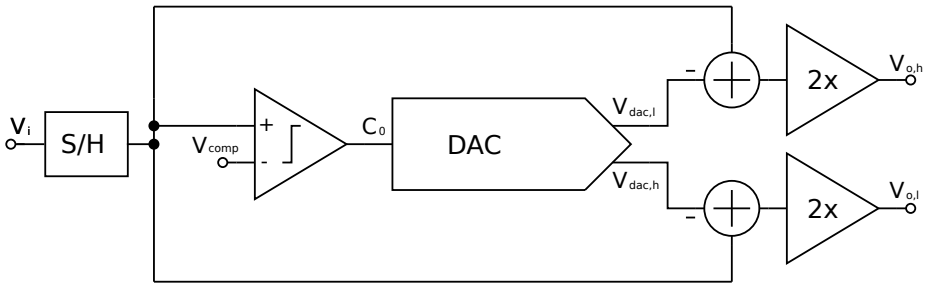


Figure 3: Converter stage topology,1-bit case

Table 1: Output voltages of the Converter stage, 1-bit case

Code	$\mathbf{V_{o,low}}$	$\mathbf{V_{o,high}}$
0	$2 \cdot (V_i - \frac{1}{2}V_{ref})$	$2 \cdot (V_i - \frac{0}{2}V_{ref})$
1	$2 \cdot (V_i - \frac{2}{2}V_{ref})$	$2 \cdot (V_i - \frac{1}{2}V_{ref})$

2.2.2 Dual-residue stage

The rest of the stages in the ADC will be dual-residue stages. The basic topology is shown in Figure 4. It differs from the conversion stage by using an interpolator instead of the DAC and subtraction circuitry. Also the input voltages should be a dual-residue representation. The interpolator is used to zoom in on the zero-crossing to generate the output voltage, as well as generate the input voltage for the comparator. The input voltage for the comparator should be the mean value of the two input voltages, and is compared to zero to check if the zero-crossing is higher or lower than the mean value.

The output voltages of the stage are selected from the comparator result as shown in Table 2. In Figure 2 the zooming operation is shown, the illustration is made without amplification of the output to better show the operation. In 2b the middle point of the input voltages is shown as a black line. Since this point is negative the zero-crossing have to be in the upper half of the signal, hence the interpolator will zoom in on this half. The output voltages will therefore be the high input voltage, and the middle point as shown in Figure 2c. The same argument is valid if the middle point is positive as is the case in 2c, here the output voltages will be the low input voltage and the middle point.

Table 2: Output Voltages of the Dual-residue stage, 1-bit case

Code	$\mathbf{V_{o,high}}$	$\mathbf{V_{o,low}}$
0	$2\left[\frac{2}{2}(V_{i,high} - V_{i,low}) + V_{i,low}\right]$	$2\left[\frac{1}{2}(V_{i,high} - V_{i,low}) + V_{i,low}\right]$
1	$2\left[\frac{1}{2}(V_{i,high} - V_{i,low}) + V_{i,low}\right]$	$2\left[\frac{0}{2}(V_{i,high} - V_{i,low}) + V_{i,low}\right]$

2.3 Proof of operation

Let V_i be the input voltage, $V_{Q,high}$ the maximum input voltage, and $V_{Q,low}$ the minimum input voltage. The dual-residue representation is given by (1) and (2)

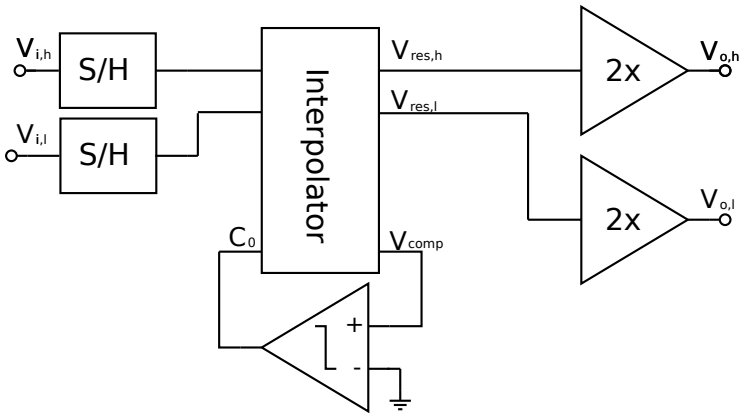


Figure 4: Topology of the Dual-residue stage

$$V_{o,high} = V_i - V_{Q,low} \tag{1}$$

$$V_{o,low} = V_i - V_{Q,high} \tag{2}$$

The position of the zero crossing is given by (3). Here $V_i - V_{Q,low}$ is the distance from the minimum voltage to the input voltage, and $V_{Q,high} - V_{Q,low}$ is the total range, hence the zero crossing is a representation of the input voltage.

$$Z = \frac{V_h}{V_h - V_l} = \frac{V_i - V_{Q,low}}{(V_i - V_{Q,low}) - (V_i - V_{Q,high})} = \frac{V_i - V_{Q,low}}{V_{Q,high} - V_{Q,low}} \tag{3}$$

2.4 Modifications for 1.5-bit stages

By using a 1.5-bit per stage architecture instead of 1-bit it is possible to take advantage of the added redundancy for digital error correction.[Vecchi et al., 2011] In order to resolve the extra bit a extra comparator is added to each stage. When this is done the output voltages and comparator voltages of each stage is changed, such that the output voltages overlap. The input voltages for the comparators should be placed in the middle of this overlap area to remove

small offset errors. This is illustrated in Figure 5 for the dual-residue stage. The voltages V_{c0} and V_{c1} are the input voltages for the two comparators, and are placed in the middle of the overlaps to make room for errors which can be digitally corrected later. This principle also applies for the converter stage. In Table 3 and Table 4 the output voltages and comparator voltages for the converter and dual residue stage are shown.

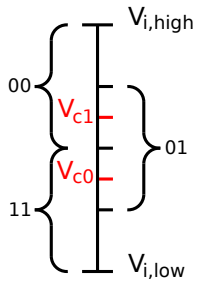


Figure 5: Output voltages for an 1.5-bit architecture, V_{c0} and V_{c1} are the input voltages for the comparators

Table 3: Output and comparator voltages for the Converter stage, 1.5-bit case

Code	$\mathbf{V_{o,low}}$	$\mathbf{V_{o,high}}$
00	$2 \cdot (V_i - \frac{2}{4}V_{ref})$	$2 \cdot (V_i - \frac{0}{4}V_{ref})$
01	$2 \cdot (V_i - \frac{3}{4}V_{ref})$	$2 \cdot (V_i - \frac{1}{4}V_{ref})$
11	$2 \cdot (V_i - \frac{4}{4}V_{ref})$	$2 \cdot (V_i - \frac{2}{4}V_{ref})$
	$\mathbf{V_{comp,low}}$	$\mathbf{V_{comp,high}}$
	$\frac{3}{8}V_{ref}$	$\frac{5}{8}V_{ref}$

2.5 Error analysis

2.5.1 Amplifier offset

Because the dual-residue ADC does not require an accurate gain, the offset voltage in the amplifiers becomes one of the dominant error sources.[Vecchi et al., 2011] The position of the zero-crossing can be written as (4). By introducing the offset errors, $V_{off,high}$ and $V_{off,low}$,on the amplifiers, the equation is modified to

Table 4: Output and comparator voltages for the dual-residue stage, 1.5-bit case

Code	$\mathbf{V_{o,high}}$	$\mathbf{V_{o,low}}$
00	$2\left\lceil\frac{4}{4}(V_{i,high}-V_{i,low})+V_{i,low}\right\rceil$	$2\left\lceil\frac{2}{4}(V_{i,high}-V_{i,low})+V_{i,low}\right\rceil$
01	$2\left\lceil\frac{3}{4}(V_{i,high}-V_{i,low})+V_{i,low}\right\rceil$	$2\left\lceil\frac{1}{4}(V_{i,high}-V_{i,low})+V_{i,low}\right\rceil$
11	$2\left\lceil\frac{2}{4}(V_{i,high}-V_{i,low})+V_{i,low}\right\rceil$	$2\left\lceil\frac{0}{4}(V_{i,high}-V_{i,low})+V_{i,low}\right\rceil$
	$\mathbf{V_{comp,low}}$	$\mathbf{V_{comp,high}}$
	$\frac{3}{8}(V_{i,high}-V_{i,low})+V_{i,low}$	$\frac{5}{8}(V_{i,high}-V_{i,low})+V_{i,low}$

(5). To get the expression for the error in the zero-crossing caused by an offset error, (5) is subtracted from (4). By assuming $V_{off,low}-V_{off,high}$ is significantly smaller than $V_{i,low}-V_{i,high}$ the the expression can be written as (6)

$$Z=\frac{V_{i,low}}{V_{i,low}-V_{i,high}} \tag{4}$$

$$Z'=\frac{V_{i,low}+V_{off,low}}{V_{i,low}+V_{off,low}-V_{i,high}-V_{off,high}} \tag{5}$$

$$\Delta Z=Z-Z_1\approx\frac{V_{i,low}\cdot(V_{off,low}-V_{off,high})-V_{off,low}(V_{i,low}-V_{i,high})}{(V_{i,low}-V_{i,high})^2} \tag{6}$$

If both offset voltages are identical there will be a constant offset in the zero-crossing of $\frac{V_{off}}{V_{i,low}-V_{high}}$. However if the offset is differential there will be a signal dependent error as shown in Figure 6. The error will be between $\pm\frac{V_{off}}{V_{i,low}-V_{high}}$. Since $V_{i,low}-V_{high}$ is equal to the total distance between the two input voltages, offset voltages for the amplifiers should be lower than $\frac{V_{lsb}}{2}$.

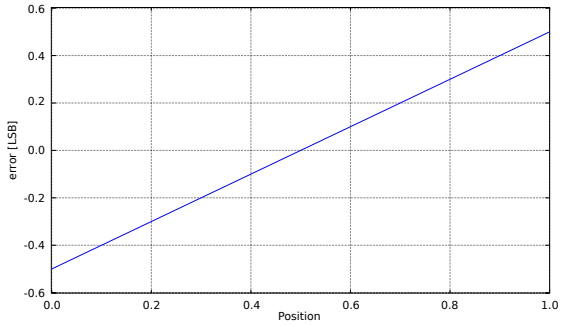


Figure 6: Error in the zero-crossing due to an offset error of $\pm\frac{1}{1024}$ of full scale in the two amplifiers.

2.5.2 Gain mismatch

The second most important error source is due to gain mismatch between the two amplifiers.[Vecchi et al., 2011] The equation for the zero-crossing can be modified to include the gain, and if there is no mismatch in the gain the expression becomes (7). By introducing a mismatch in the gain the equation is modified to (8), where a_1 and a_2 is the errors of the two amplifiers. By subtracting (8) from (7) we have the error in the zero-crossing due to mismatch in the two amplifiers as given in (9). The assumption that $V_{off,low} - V_{off,high}$ is significantly smaller than $V_{i,low} - V_{i,high}$ is made to simplify the expression. The error is shown in Figure 7 with the mismatch $a_1 - a_2$ equal to $\frac{1}{2^8}$ and the resolution is 9 bits. The error in the zero-crossing is signal dependent with a maximum error of $\frac{LSB}{2}$ when $V_{i,high} = -V_{i,low}$. This indicates that the amplifiers should be matched to a gain within $N - 1$ bits.

$$Z = \frac{A \cdot V_{i,low}}{A \cdot V_{i,low} - A \cdot V_{i,high}} \tag{7}$$

$$Z' = \frac{A \cdot a_1 \cdot V_{i,low}}{A \cdot a_1 \cdot V_{i,low} - A \cdot a_2 \cdot V_{i,high}} \tag{8}$$

$$\Delta Z = Z - Z_1 \approx \frac{V_{i,low} \cdot V_{i,high}(a_1 - a2)}{(V_{i,low} - V_{i,high})^2} \tag{9}$$

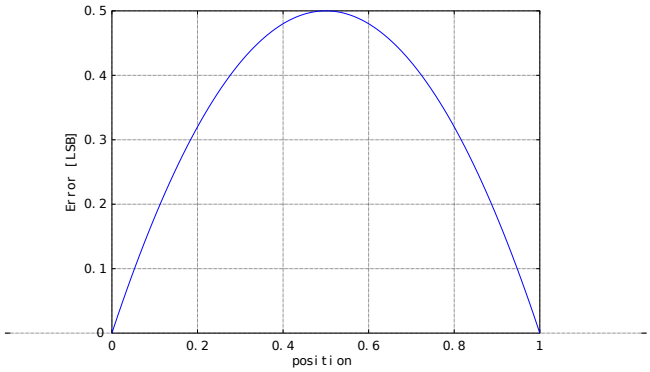


Figure 7: Error in the zero-crossing due to gain mismatch in the two amplifiers, $a_1 - a_2 = \frac{1}{256}$

3 Design

The ADC is divided into two types of stages, a converter stage and a dual-residue stage. These stages are connected as shown in Figure 8 where the first stage is a converter stage and is followed by 8 of the dual residue stages to complete the ADC. There's also a block which performs the digital error correction. All analog blocks are differential or pseudo differential with a common mode of 1.1V to make room for NMOS input transistors. The common-mode could be moved if needed.

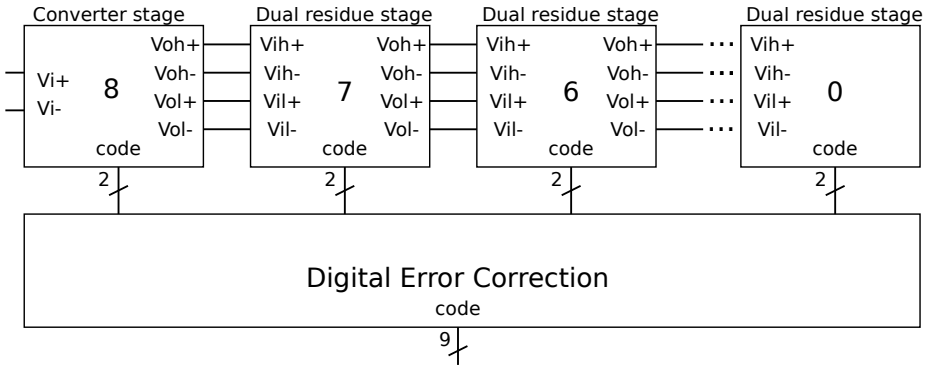


Figure 8: Topology of the Dual-residue pipelined ADC

Since the converter stage is basically a conventional pipeline stage with an extra output, the main focus was on designing the dual residue stage. Therefore only a high level model was created for the converter stage. The topology of the model is shown in Figure 9. Differential signals are left out to simplify the figure. All blocks are created in Verilog-A and the code can be viewed in Appendix A. Care should be taken when the first stage ultimately is designed as it is the stage which is expected to contribute most to both accuracy and power consumption. Not shown in the figure is the reference generator needed to supply the reference voltages to the DAC.

The second stage consists of an interpolator and residue amplifiers as described in Section 2.2.2, the basic topology is shown in Figure 10. Differential signals are left out. The interpolation is performed as described in 3.1.2 and the amplification is done by amplifier models described in Section 3.2. The comparator

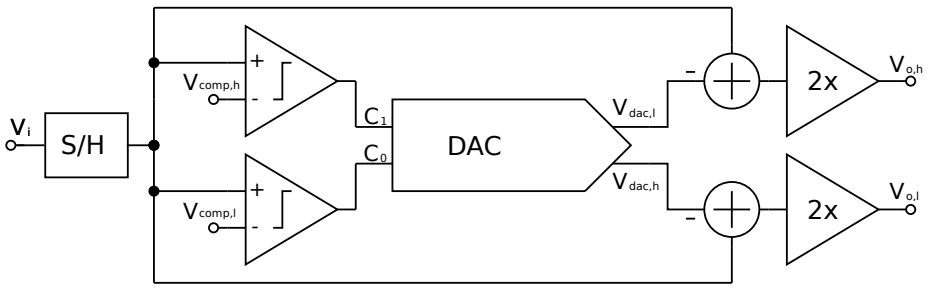


Figure 9: Topology of the Converter stage

is a normal single ended comparator and uses the differential voltage to check if the input is positive or negative and does not need any external reference.

3.1 Interpolator

The interpolator needs to generate the voltages from Table 4. There are multiple ways to perform this interpolation by using resistors, capacitors or active circuits. [Mangelsdorf et al., 1993, Vecchi et al., 2011] Three methods were considered to be used, a resistive ladder, a capacitive ladder and a charge distribution array.

3.1.1 Resistive ladder

A resistive ladder is the simplest method and consists of voltage dividers with multiple resistors in series between the input voltages as shown in Figure 11. The correct nodes are selected as the output and comparator voltages. To reduce the thermal noise enough to support 9 bits the resistance would have to be small, this would again consume a lot of power and the resistive ladder was discarded as a option.

3.1.2 Capacitive ladder

A capacitive ladder is shown in Figure 12. As with the resistive ladder multiple identical capacitors are placed in series and the correct nodes are selected as the

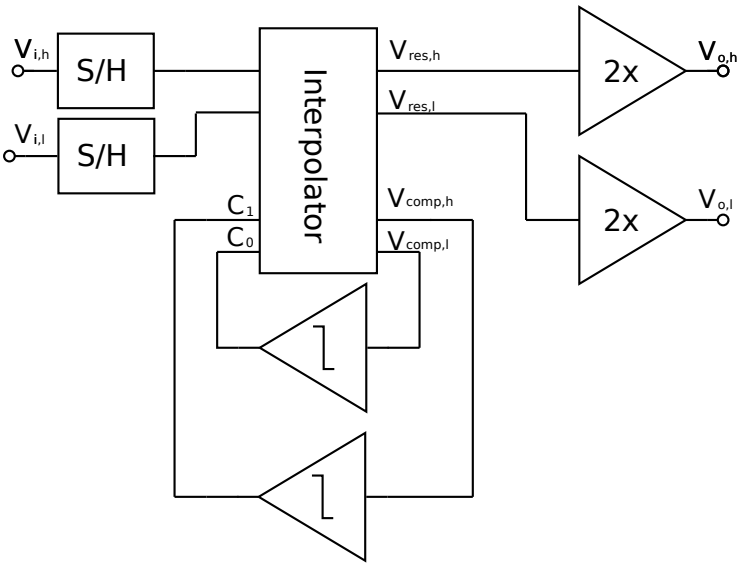


Figure 10: Topology of the Dual-residue stage

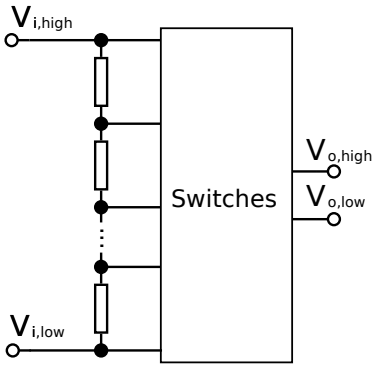


Figure 11: Resistive ladder as interpolator

output. All nodes in the ladder are reset to common mode periodically, before connecting the inputs to the top and bottom node to generate the voltages. This method would generate all needed voltages at same time, as well as function as the sample and hold of the stage.

A problem with the capacitive ladder is parasitic capacitance. All the nodes in the capacitive ladder will have a parasitic capacitance to ground. This capacitance will be charged to common mode during reset, hence the output voltages will move towards common mode compared to their ideal voltage. To compensate for this error the capacitors in the ladder can be of different sizes to match them to the parasitic capacitance. However due to the dual-residue architecture the mean voltage between the two input voltages will vary, and not be constant at common-mode. This will introduce a new error where the output voltages still won't be correct, except when the input voltages are differential such that the mean value is the common mode voltage. Again this can be compensated by adding the difference in the charge over the parasitics when charged to common mode, and if they were reset to the mean value of the input. To do this either large capacitors or an extra buffer is needed, and both the current consumption and the complexity would increase. Another drawback by a capacitive ladder is that by placing capacitors in series the effective capacitance would decrease, hence the total capacitance would need to be increased to reduce the thermal noise. Because of these disadvantages the capacitive ladder was discarded as an option.

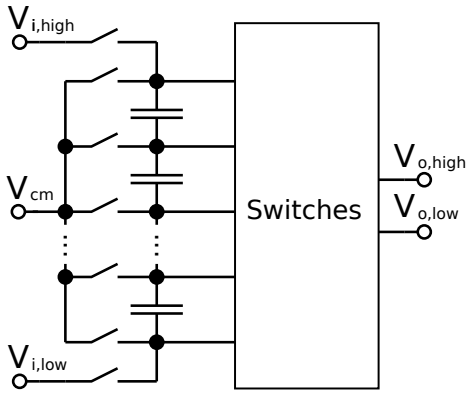


Figure 12: Capacitive ladder used as interpolator

3.1.3 Charge redistribution

A differential charge redistribution array as shown in Figure 13 may be used as the interpolator. The positive and negative halves are symmetrical, therefore only the positive half is explained. The output voltage is generated by connecting the output node to common mode while the capacitors are connected to either V_{h+} or V_{l+}, depending on the desired output voltage. In the next phase the output node is disconnected from common mode and all the capacitors are switched to common mode. This generates the output voltage in (10) where C_1 and C_2 is the total capacitance connected to V_h and V_l respectively. Since there are two residue voltages in the ADC, two differential arrays are used in parallel, with different number of capacitors connected to each input voltage, to generate both the high and the low residue voltage. In (11) the position of the zero crossing is calculated. Here C_3 and C_4 is the capacitances in the second array. Since the arrays are identical, $C_1 + C_2 = C_3 + C_4$, hence the parasitic capacitance C_p can be removed from the equation and does not affect the zero crossing. The parasitic capacitances will however reduce the output voltage.

$$V_o = \frac{(V_{h+} - V_{h-})C_1 + (V_{l+} - V_{h-})C_2}{C_1 + C_2 + C_p} \tag{10}$$

$$\begin{aligned} Z &= \frac{V_{o,l}}{V_{o,l} - V_{o,h}} \\ &= \frac{\frac{V_{l,h}C_1 + V_{l,l}C_2}{C_1+C_2+C_p}}{\frac{V_{l,h}C_1 + V_{l,l}C_2}{C_1+C_2+C_p} - \frac{V_{h,h}C_3 + V_{h,l}C_4}{C_3+C_4+C_p}} \\ &= \frac{V_{l,h}C_1 + V_{l,l}C_2}{V_{l,h}C_1 + V_{l,l}C_2 - V_{h,h}C_3 - V_{h,l}C_4} \end{aligned} \tag{11}$$

The interpolator has to generate both the voltages for the comparators as well as the output voltages. The timing diagram to do this for two following stages is shown in Figure 14. First the comparator voltages are generated, one is generated by each array before the digital control circuitry reads the comparator outputs, and uses this to set the appropriate output voltage. Table 5 summarizes the number of unit capacitors connected to each input voltage to generate any residue voltage as well as the voltages for the comparators. The arrays is also used as the sample and hold of the stage and the clock is stopped to hold the

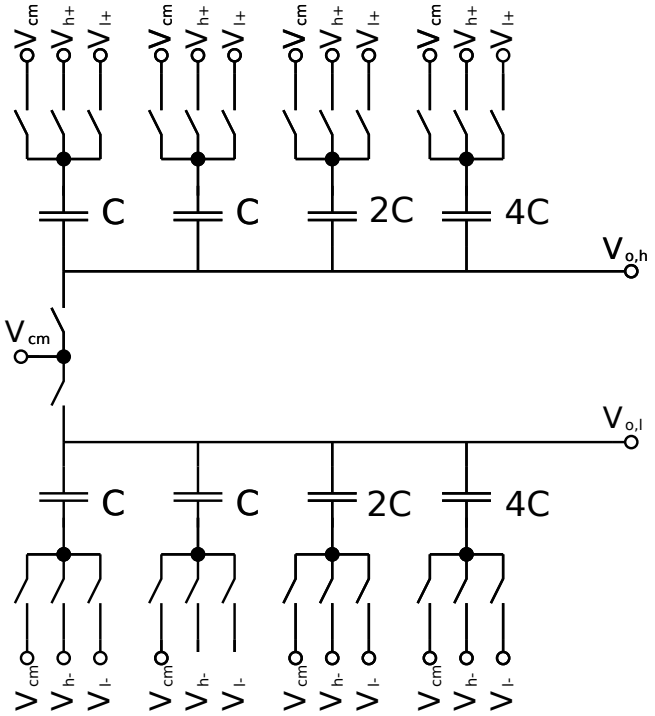


Figure 13: Differential charge redistribution array used for interpolation

output while the next stage is sampling. The digital control was written in Verilog-A and can be viewed in Appendix B.

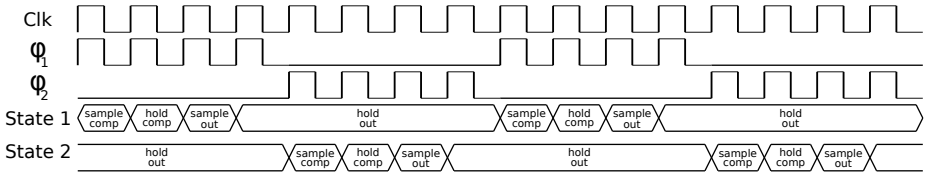


Figure 14: Clock scheme of SAR when a time continues amplifier is used

Table 5: Capacitance connected to each input voltage for both charge redistribution arrays to generate any output

	Array H		**Array L**	
code	$\mathbf{C_{Vi,h}}$	$\mathbf{C_{Vi,l}}$	$\mathbf{C_{Vi,h}}$	$\mathbf{C_{Vi,l}}$
00	8	0	4	4
01	6	2	2	6
11	4	4	0	8
comparator	5	3	3	5

Capacitor sizes

To ensure low enough thermal noise the capacitor sizes should be selected such that the noise is below $\frac{V_{lsb}}{2}$. This is calculated in (12) for a 9 bit converter with $\pm 100m$V swing. This capacitance is calculated for the first stage and can be reduced for each stage given there is a gain of over 1 in each stage. In the simulations a unit capacitance of $20f$F is used giving a total capacitance of $320f$F in each array.

$$\begin{aligned} \sqrt{\frac{kT}{C}} &= \left(\frac{V_{swing}}{2 \cdot 2^9}\right)^2 \\ C &= \frac{kT \cdot (2 \cdot 2^9)^2}{0.2^2} = 108fF \end{aligned} \tag{12}$$

3.2 Residue amplifier

The amplifier in the ADC should have a gain close to 2 to reduce the effects of thermal noise in later stages, an exact gain is not important but it needs to be possible to match the two amplifiers in each stage. Because of the SAR operation of the interpolator the amplifier will have to settle within $\frac{V_{LSB}}{2}$ within $\frac{1}{4}$ of the clock period of each stage or in the case of a 50MS ADC within $2.5ns$. There are numerous possible architectures and approaches.

3.2.1 Comparator based switched capacitor (CBSC) amplifier

A comparator based switched capacitor amplifier can be used in a pseudo differential configuration as shown in Figure 15 to get a accurate gain for the residue voltage. The CBSC works like a switched capacitor amplifier based on an opamp, but with an current source controlled by a comparator instead. In phase 1 the input voltage is sampled over two capacitors, and the output is reset to a given voltage, as common mode or rail. In the second phase the the capacitors are connected in a feedback loop from the output to the comparator. When the current source charges the output capacitance the voltage in the feedback will move towards the reference of the comparator and when the correct output voltage is reached the current source will switch off.

The output voltage of a CBSC can be written as (13) where C_{tot} is the equivalent capacitance from the output and the feedback and V_{reset} is the reset voltage of the output. This can be rewritten to calculate the minimum current as in (14).

Since a real comparator will have both delay and offset, the output voltage will contain an error based on the time the comparator uses to turn of the current source compared to the ideal timing. This will generate the output error as shown in (15), however since a pseudo differential configuration is used this error will cancel out if the two circuits are identical. Instead the common mode voltage will be change by the same amount. To keep this common mode error as low as possible, the current should be kept to a minimum. If this error gets to big it might be necessary to introduce a level-shifter in order to move the common-mode back.

$$V_o = \frac{I(t) \cdot t}{C_{tot}} + V_{reset} \tag{13}$$

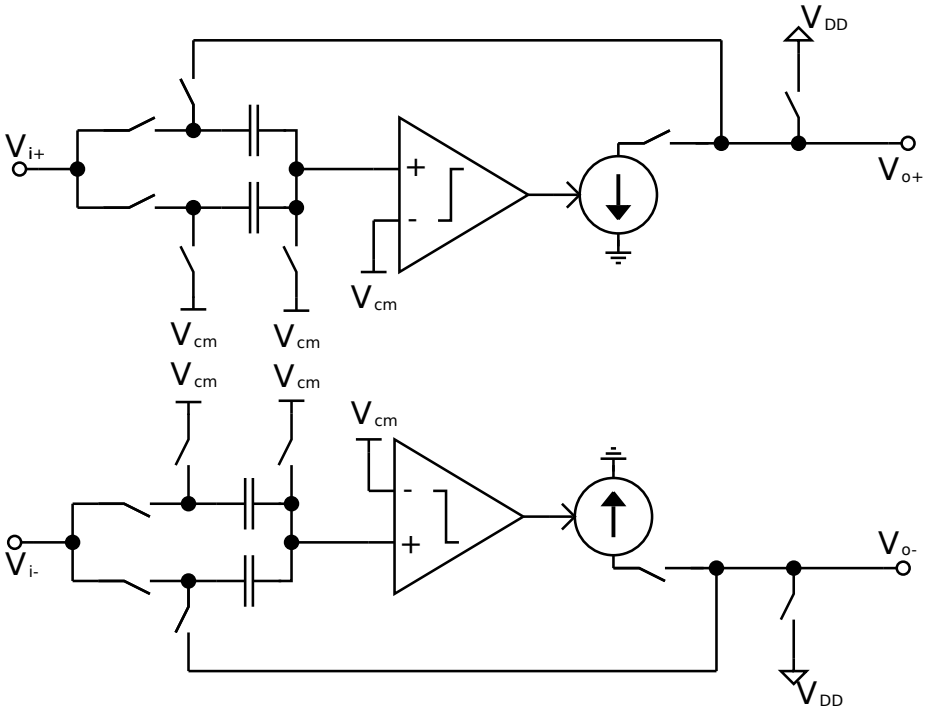


Figure 15: Topology of the CBSC amplifier

$$I = \frac{(V_{swing})C_{tot}}{t} \tag{14}$$

$$V_{err} = \frac{I(t) \cdot \Delta t}{C_{tot}} \tag{15}$$

The cancellation of the delay error will only occur when the current from the current source is constant during the active period. Since the current source will be implemented by a current mirror with a final output impedance this assumption is not valid.[Wulff and Ytterdal, 2008b] The two amplifiers in the pseudo differential pair will have different output voltages, hence the voltage over the current mirrors will be different. In (16) this error is presented, I_{avg} is the average difference in the current between the positive and negative amplifier during the delay of the comparator.

$$V_{err} = \frac{\Delta I_{avg} \cdot \Delta t}{C_{tot}} \tag{16}$$

Since the error due to the current source and the delay of the comparator are strongly connected the error can be reduced by one or both of the comparator and the current source. However since the common mode offset error is only due to the non-idealities in the comparator this is the component where it's most room for improvement. All errors described above can be modeled as offset errors in the dual residue ADC and should be designed accordingly to (6) in Section 2.5.

The CBSC needs needs to work synchronous with both an charge redistribution array on the input and output. The digital control was modified to work with the CBSC and can be found in Appendix C.

3.2.2 Open loop buffer and charge pump

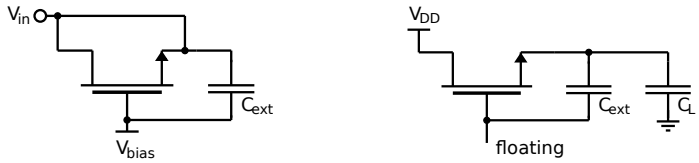
One method to get the gain between the stages is to use a charge pump to get close to 2 in gain[Ahmed et al., 2009] and an open loop buffer. This was attempted by connecting a charge pump at the output of the charge redistribution array followed by an open loop buffer, but due to the input capacitance of the buffer combined with the small capacitances used in the charge pump the

gain was reduced to less than 1. Also the input capacitance in the buffer will be signal dependent and will be a problem with small capacitors. Larger capacitors in the charge pump would decrease the problem with the input capacitance in the buffer, however this would also decrease the output voltage of the charge redistribution array and the overall gain would still be to low. A possibility is to add another buffer between the charge redistribution array and the charge pump at the cost of extra power consumption.

3.2.3 Dynamic source follower

A dynamic source follower as shown in Figure 16 has been proven an efficient method for residue amplification[Hu et al., 2009]. The parasitic capacitances in a MOS combined with a small external capacitor is used to set the gain, the expression for the gain is given in (17) where C'_{gd} is the gate drain capacitance in the amplification phase. This method will give a inaccurate gain, and when used in a conventional pipeline calibration is required. However it should still be possible to match two separate transistors and thereby get the matching required to work in a dual residue architecture without calibration.

$$A \cong \frac{C_{gs,ext} + C_{gs} + C_{gd} + C_{gb}}{C'_{gd}} \tag{17}$$



(a) Sample (b) Amplify

Figure 16: Dynamic source follower

3.3 Error correction

To get the final output code of the ADC the outputs of each stage needs to be combined. Since it's a pipelined ADC the output from the stages needs to be delayed such that the output of all stages appears at the same time, this can

be done with shift registers of different length. Further the two bits from each stage needs to be combined to one output word. This is done digitally by using Algorithm 1. The basic operation of the error correction algorithm is that each stage is either added, subtracted, or not used in the final output. If both bits is one the the stage contributes $+2^N$ where N is the significance of the stage. If both bits is 0 the stage contributes -2^N, and finally if one bit is 1 and the other one 0 the the stage does not contribute to the final output code.

Algorithm 1 Error correction algorithm

Input: c[N][2]
Output: code

```
for i = N − 1 → 0 do
  if c[i][1] == 1 AND c[i][0] == 1 then
    operation ← 1
  else
    if c[i][1] == 0 AND c[i][0] == 0 then
      operation ← −1
    else
      operation ← 0
    end if
  end if
  code ← code + operation · pow(2, i)
end for
code ← (code + 512)/2
return code
```

3.4 Summary of block demands

3.4.1 Flash Comparators

Because of the digital error correction the offset voltage of the flash comparators can be $\pm\frac{V_{lsb}}{8}$. The comparator needs to resolve the input in one clock cycle of the charge redistribution array and the delay has to be lower than $2.5ns$. The input voltages for the comparator will be $V_{cm} \pm V_{swing}$

3.4.2 Amplifier

The gain of the residue amplifiers should be close to 2, but no exact value is needed. The two amplifiers should however be matched to $\frac{1}{256}$, and the offset should be lower than $\frac{V_{lsb}}{2}$. The amplifiers also need to be linear to 9 bits within the input range. Because of the SAR operation of the interpolator the amplifier will have to settle within 2.5ns.

3.4.3 Switches

Due to the dual-residue architecture the voltages don't have to settle completely[Vecchi et al., 2011], this relaxes the demands on the switches as they only needs to be matched. The switches should still be designed for full settling to reduce problems caused by thermal noise.

More important is to keep the charge injection down. Charge injection will introduce an offset error and should cause an voltage error lower than $\frac{V_{lsb}}{2}$. Depending on the size of the switches it may be necessary to introduce techniques to reduce the charge injection like dummy switches.

4 Results and Discussion

4.1 Accuracy

Table 6 shows a summary of the results simulated with different parameters and amplifier models. The simulations were run with an ideal converter stage as described in Section 3, followed by 8 identical dual-residue stages. The blocks used in the dual-residue stage consists both of Verilog-A models and transistor level models. All simulations used the charge redistribution array with a unit capacitance of $20fF$ as the interpolator. The results are calculated from 2048 samples using the Matlab-script in Appendix D. The sampling frequency was $49.152MS/s$ and the input frequency was $4.008MHz$ with an input swing of $\pm 200mV$. The offset and mismatch errors were applied to all 9 stages in the ADC.

Table 6: Summary of simulation results

TYPE	ol	ol	ol	ol	ol	CBSC
Gain mismatch	0	$\frac{1}{256}$	$\pm\frac{1}{128}$	0	0	0
offset	0	0	0	$\pm\frac{1}{1024}$	$\pm\frac{1}{512}$	0
SNDR	55.9	55.0	50.7	53.6	50.3	55.
SNR	56.0	55.8	52.4	55.4	54.0	56
ENOB	8.99	8.85	8.13	8.61	8.13	8.9

The first simulation was run with zero offset and gain mismatch, using an ideal amplifier model with a gain of 2. The resulting ENOB is 8.99 bits and proves that the overall topology of a dual-residue pipelined ADC with a charge redistribution array as the interpolator works. The simulation with a gain mismatch of $\frac{1}{256}$ shows a ENOB of 8.85 bits. This is the calculated maximum gain mismatch for a 9-bits ADC and 8.85-bits is and expected result. When the gain mismatch was increased to $\frac{1}{512}$ the ENOB was reduced to 8.13 bits. This is a bit higher than expected from the equations in Section 2.5, but taking into account that the error is signal dependent, and will be at its maximum at just a single point, the average error during a dynamic test would be lower. If a static test is performed this error should show up as a linearity error.

When an input offset of $\pm\frac{1}{1024}$ of the full scale range is added to the amplifier the ENOB was 8.61-bits, and correlates with the equations in Section 2.5. By increasing the offset error to $\pm\frac{1}{512}$ the ENOB drops to 8.13bits. Once again this

is a bit higher than expected, but since the offsets are differential for the two amplifiers, the error is signal dependent and the average error is lower than the maximum. Also here it's likely to show up as a linearity error in a static test.

4.2 Amplifier choice

The circuit was simulated with models of different types of amplifiers. An open-loop buffer with a charge-pump was implemented, but a large input capacitance in the buffer reduced the overall gain below 1 and further simulations were discarded. A pseudo differential CBSC amplifier was designed and simulated with ideal switches and comparators, the current source used delivered $70u$A. From Table 6 the effective number of bits was 8.9. Keeping in mind that most of the components in the simulation were ideal, this is an expected result, and proves that a CBSC amplifier is possible to use as the residue amplifier. The power estimate for the CBSC circuit is 140μW per stage, but the capacitance in each stage is higher than required and it should be possible to decrease the power consumption further. However this only includes the charging and discharging of the capacitors, while the main contribution to the current consumption in addition to the capacitors is the comparator. In [Wulff and Ytterdal, 2005] a comparator with similar specs is reported to use 222μW, and with two for each pseudo differential CBSC pair, in total 4 in each stage, the power consumption will be dominated by the comparators. The estimated power consumption of the CBSC would be $500u$W. An possible approach to reduce the power consumption is by exchanging the CBSC with a zero-crossing based switched capacitor circuit(ZCBC)[Brooks and Lee, 2007]. A zero-crossing based switched capacitor amplifier will be faster than a CBSC, and will not consume any static power, hence it is likely to reduce the power consumption for the amplifiers.

The performance of the CBSC amplifier will degrade with comparator delay and offset voltage. A comparator designed for this purpose may have a delay of 400ps[Wulff and Ytterdal, 2005]. With the currents and capacitances used here this would move the common mode voltage at the output $175m$V. This is not necessary a problem, and can easily be corrected in the charge redistribution array, but will increase power consumption as the capacitors will have a larger voltage swing than needed. The current source will also be subject to an error due to the finite output resistance, and a delay from the comparator will introduce an error due to different voltages over the positive and negative output mirror, hence the current in the two mirrors will not be matched. In

[Wulff and Ytterdal, 2008a] it was proposed to introduce an offset voltage to the comparators such that the delay is corrected. This is possible since the feedback voltage will change linearly with time as the current source charges the capacitance, and thereby an voltage offset can remove a time delay. If a ZCBC is used instead, it will have a lower delay and thereby reduce this error, and might be better suited.

Another possible amplifier choice is an open loop amplifier. A dynamic source follower was proposed in [Hu et al., 2009], and used in a 9.4bit 50MS/s ADC. The total power consumption from the amplifiers were $0.49m$W for all 14 stages. This amplifier depends on capacitances in MOS devices combined with an external capacitor to set the gain, hence the gain is inaccurate, and calibration had to be used to compensate. In a dual-residue ADC it should be possible to match two devices, and thereby remove the need for calibration. Another advantage is that this amplifier depend on high intrinsic gain and is well suited for technology down scaling. The low power consumption of this amplifier makes it an interesting option for implementation.

4.3 Power consumption estimate for the different amplifiers

To get a quantitative comparison foundation of the power consumption, an estimate is calculated for the different amplifiers. The estimate only accounts for the current needed to charge the output capacitances and the power consumed by the comparators. The converter stage is left out from the estimation and only the 8 dual-residue stages are accounted for, but the results should still be valuable to chose amplifiers to be used in the converter stage. Since the last stage does not need an output voltage the power for the amplifiers are multiplied by 7 as show in (18). In [Hu et al., 2009] the comparators used for 14 stages consumed less than 950μW or 68μW for each comparator and this is used as the estimated current consumption.

$$P = P_{amp} \cdot 7 + P_{comparators} \cdot 8 \tag{18}$$

The ADC with CBSC amplifiers is calculated to consume $5.2m$W, where the main part is consumed by the comparators in the amplifiers. By using a ZCBC amplifier instead the estimated power consumption is $2.1m$W, a significant improvement. It's important to keep in mind that this is only an estimate under

the assumption that no significant power is consumed by the amplifier itself. The advantages by moving from a CBSC to a ZCBC is verified by looking at[Murmann, 2012] where the best CBSC has a FOM of $764\frac{fJ}{conv}$ while the best ZCBC has a FOM of $87.5\frac{fJ}{conv}$. The dynamic source follower is estimated to consume the same power as the ZCBC, this is because the output capacitors will be reset and charged in a similar matter and the total swing over the capacitors are expected to be identical. From the estimates of power consumption the ZCBC and dynamic source follower seems to be the best choice as amplifiers. Because the reliability of the ZCBC is better proven[Brooks and Lee, 2007, Hershberg et al., 2010, Brooks and Lee, 2009] it might be the safest choice.

$$P_{CBSC} = (144\mu W + 2 \cdot 220\mu W) \cdot 7 + 2 \cdot 68\mu W \cdot 8 = 5.2mW \tag{19}$$

$$P_{ZCBC} = 144\mu W \cdot 7 + 2 \cdot 68\mu W \cdot 8 = 2.1mW \tag{20}$$

$$P_{openloop} = (144\mu W) \cdot 7 + 2 \cdot 68\mu W \cdot 8 = 2.1mW \tag{21}$$

4.4 1-bit architecture

The comparators are expected to be the main contributor to the power consumption of the ADC. Because of this it might be an advantage to reduce the number of comparators by using a 1-bit pr stage architecture instead. This would remove 1 comparator from each stage, but at the same time increase the demands of the comparators that is left. If this is done fewer voltages are needed to be generated by the interpolator, and only the two input voltages and the mean value between them is needed, hence a simpler interpolator can be designed.

5 Conclusion and Further work

A 9-bit $50MS/s$ dual-residue pipelined ADC using a charge redistribution array as the interpolator is presented and analyzed. When simulated with idealized components the ADC had an ENOB of 8.99-bits and the topology is proven to work. The maximum non-idealities for the different blocks were found and verified by simulations. The maximum allowed offset voltage of the residue amplifier is $\frac{V_{lsb}}{2}$, and with this offset voltage for all the amplifiers in the ADC the ENOB dropped to 8.61-bits. The maximum allowable mismatch between the two residue amplifiers is $\frac{1}{256}$, with this mismatch the ENOB is 8.85-bits. Both these demands should be possible to reach without the use of calibration. The best candidate for residue amplification is a zero-crossing based amplifier, and with this type of amplifier the last 8 stages of the ADC has a estimated power consumption of $2.1m$W. Both scaling of the stages and reducing the unit capacitance can decrease the power consumption.

5.1 Further work

In this thesis error sources in the amplifiers and power consumption has been estimated and simulated. A deeper noise analysis should be performed to get better design equation for noise parameters, and also include noise generated by the amplifiers. Also no mismatch simulations were performed and should be performed to ensure correct operation, especially when the circuit is implemented with transistor-level blocks .

References

I. Ahmed, J. Mulder, and D.A. Johns. A 50ms/s 9.9mw pipelined adc with 58db sndr in 0.18μm cmos using capacitive charge-pumps. In *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pages 164 –165,165a, feb. 2009. doi: 10.1109/ISSCC.2009.4977359.

L. Brooks and Hae-Seung Lee. A zero-crossing-based 8-bit 200 ms/s pipelined adc. *Solid-State Circuits, IEEE Journal of*, 42(12):2677 –2687, dec. 2007. ISSN 0018-9200. doi: 10.1109/JSSC.2007.908770.

L. Brooks and Hae-Seung Lee. A 12b 50ms/s fully differential zero-crossing-based adc without cmfb. In *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pages 166 –167,167a, feb. 2009. doi: 10.1109/ISSCC.2009.4977360.

B.P. Hershberg, S.T. Weaver, and Un-Ku Moon. A 1.4v signal swing hybrid cls-opamp/zcbc pipelined adc using a 300mv output swing opamp. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 302 –303, feb. 2010. doi: 10.1109/ISSCC.2010.5433894.

J. Hu, N. Dolev, and B. Murmann. A 9.4-bit, 50-ms/s, 1.44-mw pipelined adc using dynamic source follower residue amplification. *Solid-State Circuits, IEEE Journal of*, 44(4):1057 –1066, april 2009. ISSN 0018-9200. doi: 10.1109/JSSC.2009.2014705.

Y.P. Lee and R.L. Geiger. Gain error correction scheme for multiply-by-two gain amplifier in pipelined adc. In *Circuits and Systems, 2000. 42nd Midwest Symposium on*, volume 1, pages 190 –193 vol. 1, 1999. doi: 10.1109/MWSCAS.1999.867240.

S.H. Lewis, H.S. Fetterman, Jr. Gross, G.F., R. Ramachandran, and T.R. Viswanathan. A 10-b 20-msample/s analog-to-digital converter. *Solid-State Circuits, IEEE Journal of*, 27(3):351 –358, mar 1992. ISSN 0018-9200. doi: 10.1109/4.121557.

C. Mangelsdorf, H. Malik, S.-H. Lee, S. Hisano, and M. Martin. A two-residue architecture for multistage adcs. In *Solid-State Circuits Conference, 1993. Digest of Technical Papers. 40th ISSCC., 1993 IEEE International*, pages 64 –65, feb 1993. doi: 10.1109/ISSCC.1993.280082.

B. Murmann. Adc performance survey 1997-2012. [Online] Available: `http://www.stanford.edu/~murmann/adcsurvey.html`, 2012.

Junhua Shen and P.R. Kinget. A 0.5-v 8-bit 10-ms/s pipelined adc in 90-nm cmos. *Solid-State Circuits, IEEE Journal of*, 43(4):787 –795, april 2008. ISSN 0018-9200. doi: 10.1109/JSSC.2008.917470.

M. Taherzadeh-Sani and A.A. Hamoui. Analysis of dynamic element matching (dem) in pipelined adcs. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4 pp. –5266, may 2006. doi: 10.1109/ISCAS.2006.1693820.

D. Vecchi, J. Mulder, F.M.L. van der Goes, J.R. Westra, E. Ayranci, C.M. Ward, Jiansong Wan, and K. Bult. An 800 ms/s dual-residue pipeline adc in 40 nm cmos. *Solid-State Circuits, IEEE Journal of*, 46(12):2834 –2844, dec. 2011. ISSN 0018-9200. doi: 10.1109/JSSC.2011.2164301.

C. Wulff and T. Ytterdal. 0.8v 1ghz dynamic comparator in digital 90nm cmos technology. In *NORCHIP Conference, 2005. 23rd*, pages 237 – 240, nov. 2005. doi: 10.1109/NORCHP.2005.1597033.

C. Wulff and T. Ytterdal. Cbsc pipelined adc with comparator preset, and comparator delay compensation. dec. 2008a.

C. Wulff and T. Ytterdal. Design and behavioral simulation of comparator-based switched-capacitor circuits. In *NORCHIP, 2008.*, pages 246 –249, nov. 2008b. doi: 10.1109/NORCHP.2008.4738320.

List of Figures

List of Tables

Listings

List of Algorithms

A Verilog-A code for the converter stage

Listing 1: Ideal Adder Verilog-A code

```
// VerilogA for highlevel_adc, adder_diff, veriloga

`include
`include

module adder_diff(viap,vian,visp,visn,vop,von);
  input viap, vian, visp,visn;
  output vop,von;
  electrical viap,vian,visp,visn,vop,von;
  parameter real vcm=1.1;
  analog begin
    V(vop)<+ vcm + (V(viap,vian)-V(visp,visn))/2;
    V(von)<+ vcm - (V(viap,vian)-V(visp,visn))/2;
    //V(vop,von)<+ V(viap,vian)-V(visp,visn);
  end

endmodule
```

Listing 2: Residue amplifier Verilog-A code

```
// VerilogA for highlevel_adc, 2xamp_diff, veriloga

`include
`include

module amp2x_diff(vip,vin,vdd,vss,vop,von);
  input vdd,vss,vip,vin;
  output vop,von;
  electrical vdd,vss, vin,vip,von,vop;
  analog begin
    //V(vop,von)<+2*V(vip,vin);
    V(vop)<+2*(V(vip)) -(V(vip)+V(vin))/2;
    V(von)<+2*(V(vin)) -(V(vip)+V(vin))/2;
  end



endmodule
```

Listing 3: Comparator Verilog-A code

```
// VerilogA for highlevel_adc, comparator_diff, veriloga

`include
`include

module comparator_diff(vdd,vss,vin,vip,vrefn,vrefp,vo);
  input vdd,vss,vin,vip,vrefn,vrefp;
  output vo;
  electrical vdd,vss,vin,vip,vrefn,vrefp,vo;

  analog begin
  if(V(vip,vin)>V(vrefp,vrefn))
    V(vo)<+1.5;
  else
    V(vo)<+0;
  end

endmodule
```

Listing 4: Dac Verilog-A code

```
// VerilogA for highlevel_adc, dac_dualresidue_diff, veriloga

`include
`include

module dac_dualresidue_diff(vdd,vss,vref4d4p,vref4d4n,vref3d4p,
    vref3d4n,vref2d4p,vref2d4n,vref0d4p,vref1d4n,vref1d4p,
    vref0d4n,vohp,vohn,volp,voln,c1,c0);
  input vdd,vss,vref4d4p,vref4d4n,vref3d4p,vref3d4n,vref2d4p,
      vref2d4n,vref0d4p,vref1d4n,vref1d4p,vref0d4n,c1,c0;
  output vohp,vohn,volp,voln;

  electrical vdd,vss,vref4d4p,vref4d4n,vref3d4p,vref3d4n,vref2d4p
      ,vref2d4n,vref0d4p,vref1d4n,vref1d4p,vref0d4n,vohp,vohn,
      volp,voln,c1,c0;

  real vohpv,vohnv,volpv,volnv;
  parameter real tdelay=0.0;
  parameter real trise=1n;
  parameter real tfall=1n;

  analog begin
    if(V(c1)<(V(vdd)/2) && V(c0)<(V(vdd)/2))
    begin
      vohpv= V(vref2d4p);
```

```
21          vohnv= V(vref2d4n);
22          volpv= V(vref0d4p);
23          volnv= V(vref0d4n);
24        end
25        else if (V(c1)<(V(vdd)/2) && V(c0)>(V(vdd)/2))
26        begin
27          vohpv= V(vref3d4p);
28          vohnv= V(vref3d4n);
29          volpv= V(vref1d4p);
30          volnv= V(vref1d4n);
31        end
32        else if (V(c1)>(V(vdd)/2) && V(c0)>(V(vdd)/2))
33        begin
34          vohpv= V(vref4d4p);
35          vohnv= V(vref4d4n);
36          volpv= V(vref2d4p);
37          volnv= V(vref2d4n);
38        end
39
40     V(vohp)<+transition( vohpv, tdelay, trise, tfall);
41     V(vohn)<+transition( vohnv, tdelay, trise, tfall);
42     V(volp)<+transition( volpv, tdelay, trise, tfall);
43     V(voln)<+transition( volnv, tdelay, trise, tfall);
44
45   end
46 endmodule
```

Listing 5: Reference generator Verilog-A code

```
1  // VerilogA for highlevel_adc, refgen_diff, veriloga
2
3  `include
4  `include
5
6  module refgen_diff(vdd,vss,vref8d8p,vref8d8n,vref6d8p,vref6d8n,
       vref5d8p,vref5d8n, vref4d8p,vref4d8n, vref3d8p,vref3d8n,
       vref2d8p,vref2d8n,vref0d8p,vref0d8n);
7
8    parameter VMAX=1.2;
9    parameter VMIN=1.0;
10
11   input vdd,vss;
12   output vref8d8p,vref8d8n,vref6d8p,vref6d8n, vref5d8p,vref5d8n,
         vref4d8p,vref4d8n, vref3d8p,vref3d8n, vref2d8p,vref2d8n,
         vref0d8p,vref0d8n;
13   electrical vdd,vss,vref8d8p,vref8d8n,vref6d8p,vref6d8n,
         vref5d8p,vref5d8n, vref4d8p,vref4d8n, vref3d8p,vref3d8n,
```

```
        vref2d8p, vref2d8n, vref0d8p, vref0d8n;
  real vcm;
  analog begin
  vcm=(VMAX+VMIN)/2;
  V(vref8d8p)<+VMAX;
  V(vref8d8n)<+VMIN;
  V(vref6d8p)<+vcm+(VMAX-VMIN)*6/8/2;
  V(vref6d8n)<+vcm-(VMAX-VMIN)*6/8/2;
  V(vref5d8p)<+vcm+(VMAX-VMIN)*5/8/2;
  V(vref5d8n)<+vcm-(VMAX-VMIN)*5/8/2;
  V(vref4d8p)<+vcm+(VMAX-VMIN)*4/8/2;
  V(vref4d8n)<+vcm-(VMAX-VMIN)*4/8/2;
  V(vref3d8p)<+vcm+(VMAX-VMIN)*3/8/2;
  V(vref3d8n)<+vcm-(VMAX-VMIN)*3/8/2;
  V(vref2d8p)<+vcm+(VMAX-VMIN)*2/8/2;
  V(vref2d8n)<+vcm-(VMAX-VMIN)*2/8/2;
  V(vref0d8p)<+vcm;
  V(vref0d8n)<+vcm;
  end
endmodule


/*   V(vref8d8p)<+VMAX;
  V(vref8d8n)</2;
  V(vref6d8p,vref6d8n)<+(VMAX-VMIN)*6/8/2;
  V(vref6d8p,vref6d8n)<+(VMAX-VMIN)*6/8/2;
  V(vref5d8p,vref5d8n)<+(VMAX-VMIN)*5/8/2;
  V(vref5d8p,vref5d8n)<+(VMAX-VMIN)*5/8/2;
  V(vref4d8p,vref4d8n)<+(VMAX-VMIN)*4/8/2;
  V(vref4d8p,vref4d8n)<+(VMAX-VMIN)*4/8/2;
  V(vref3d8p,vref3d8n)<+(VMAX-VMIN)*3/8/2;
  V(vref3d8p,vref3d8n)<+(VMAX-VMIN)*3/8/2;
  V(vref2d8p,vref2d8n)<+(VMAX-VMIN)*2/8/2;
  V(vref2d8p,vref2d8n)<+(VMAX-VMIN)*2/8/2;
  V(vref0d8p,vref0d8n)</2;
  V(vref0d8p,vref0d8n)</2;
  end */
```

Listing 6: Sample and hold Verilog-A code

```
1  // VerilogA for highlevel_adc, samplehold_diff, veriloga
2
3  `include
4  `include
5
6  module samplehold_diff(clk,vip,vin,vop,von);
7    input clk,vip,vin;
8    output vop,von;
9    electrical clk,vip,vin,vop,von;
10   real sampp,sampn;
11    parameter real tdelay=0n;
12    parameter real trise=1n;
13    parameter real tfall=1n;
14   analog begin
15     // on Rise  edges of clk, sample vin
16       @( cross(V(clk)-0.75, 1)) begin
17             sampp=V(vip);
18             sampn=V(vin);
19     end
20
21
22
23
24
25     //  assign output
26     V(vop) <+ transition( sampp, tdelay, trise, tfall);
27     V(von) <+ transition( sampn, tdelay, trise, tfall);
28
29     end
30
31
32 endmodule
```

B Charge redistribution control Verilog-A code

Listing 7: Charge redistribution control Verilog-A code

```
// VerilogA fosr highlevel_adc, sar_ctrl, veriloga

`include
`include
`define   RISING   +1
`define   FALLING  -1

module sar_ctrl(cnt_o,c0,c1,clk, vc4h_h,vc4l_h,vc2h_h,vc2l_h,
    vc1ah_h,vc1al_h,vc1bh_h,vc1bl_h,vc4h_l,vc4l_l,vc2h_l,vc2l_l,
    vc1ah_l,vc1al_l,vc1bh_l,vc1bl_l, vcsample,vccm, c0_out,
    c1_out);
  input c0,c1,clk;
  output cnt_o,vc4h_h,vc4l_h,vc2h_h,vc2l_h,vc1ah_h,vc1al_h,
      vc1bh_h,vc1bl_h,vc4h_l,vc4l_l,vc2h_l,vc2l_l,vc1ah_l,vc1al_l
      ,vc1bh_l,vc1bl_l,vcsample,vccm,c0_out, c1_out;
  electrical cnt_o, c0,c1,clk, vc4h_h,vc4l_h,vc2h_h,vc2l_h,
      vc1ah_h,vc1al_h,vc1bh_h,vc1bl_h,vc4h_l,vc4l_l,vc2h_l,vc2l_l
      ,vc1ah_l,vc1al_l,vc1bh_l,vc1bl_l, vcsample,vccm,c0_out,
      c1_out;
  parameter real vhi=1.5;
  parameter real vlo=0.0;
  parameter real tdelay=0.0;
  parameter real trise=1n;
  parameter real tfall=1n;
  integer cnt;
  real c0_reg,c1_reg;
  real cnt_ov,vc4h_hv,vc4l_hv,vc2h_hv,vc2l_hv,vc1ah_hv,vc1al_hv,
      vc1bh_hv,vc1bl_hv,vc4h_lv,vc4l_lv,vc2h_lv,vc2l_lv,vc1ah_lv,
      vc1al_lv,vc1bh_lv,vc1bl_lv,vcsamplev,vccmv,c0_outv, c1_outv
      ;

  analog begin
      @(initial_step(    ,    ,      ,    )) begin
      cnt=3;
    end
   @( cross(V(clk)-vhi/2, `RISING)) begin
    if (cnt==1) begin
      c1_reg=V(c1);
      c0_reg=V(c0);
    end
        cnt=(cnt+1)%4;

  end
    cnt_ov=cnt;
```

```
34          c0_outv=c0_reg;
35        c1_outv=c1_reg;
36
37        if(cnt==0)
38        begin //generate comparator voltages
39              vc4h_hv=vhi;
40          vc4l_hv=vlo;
41          vc2h_hv=vlo;
42          vc2l_hv=vhi;
43          vc1ah_hv=vhi;
44          vc1al_hv=vlo;
45          vc1bh_hv=vlo;
46          vc1bl_hv=vhi;
47
48          vc4h_lv=vlo;
49          vc4l_lv=vhi;
50          vc2h_lv=vhi;
51          vc2l_lv=vlo;
52          vc1ah_lv=vhi;
53          vc1al_lv=vlo;
54          vc1bh_lv=vlo;
55          vc1bl_lv=vhi;
56
57          vcsamplev=vhi;
58          vccmv=vlo;
59        end
60        else if (cnt==1) begin //compare
61          vc4h_hv=vlo;
62          vc4l_hv=vlo;
63          vc2h_hv=vlo;
64          vc2l_hv=vlo;
65          vc1ah_hv=vlo;
66          vc1al_hv=vlo;
67          vc1bh_hv=vlo;
68          vc1bl_hv=vlo;
69
70          vc4h_lv=vlo;
71          vc4l_lv=vlo;
72          vc2h_lv=vlo;
73          vc2l_lv=vlo;
74          vc1ah_lv=vlo;
75          vc1al_lv=vlo;
76          vc1bh_lv=vlo;
77          vc1bl_lv=vlo;
78
79          vcsamplev=vlo;
80          vccmv=vhi;
81
82        end
```

```
83      else if (cnt==2) begin // generate output voltages
84        if (c0_reg<vhi/2 && c1_reg < vhi/2) begin //0
85          vc4h_hv=vhi;
86          vc4l_hv=vlo;
87          vc2h_hv=vhi;
88          vc2l_hv=vlo;
89          vc1ah_hv=vhi;
90          vc1al_hv=vlo;
91          vc1bh_hv=vhi;
92          vc1bl_hv=vlo;
93
94          vc4h_lv=vlo;
95          vc4l_lv=vhi;
96          vc2h_lv=vhi;
97          vc2l_lv=vlo;
98          vc1ah_lv=vhi;
99          vc1al_lv=vlo;
100         vc1bh_lv=vhi;
101         vc1bl_lv=vlo;
102
103         vcsamplev=vhi;
104         vccmv=vlo;
105       end
106       else if (c0_reg>vhi/2 && c1_reg < vhi/2) begin //2
107         vc4h_hv=vhi;
108         vc4l_hv=vlo;
109         vc2h_hv=vhi;
110         vc2l_hv=vlo;
111         vc1ah_hv=vlo;
112         vc1al_hv=vhi;
113         vc1bh_hv=vlo;
114         vc1bl_hv=vhi;
115
116         vc4h_lv=vlo;
117         vc4l_lv=vhi;
118         vc2h_lv=vhi;
119         vc2l_lv=vlo;
120         vc1ah_lv=vlo;
121         vc1al_lv=vhi;
122         vc1bh_lv=vlo;
123         vc1bl_lv=vhi;
124
125         vcsamplev=vhi;
126         vccmv=vlo;
127       end
128       else  if (c0_reg>vhi/2 && c1_reg > vhi/2) begin //4
129         vc4h_hv=vhi;
130         vc4l_hv=vlo;
131         vc2h_hv=vlo;
```

```
132           vc2l_hv=vhi;
133           vc1ah_hv=vlo;
134           vc1al_hv=vhi;
135           vc1bh_hv=vlo;
136           vc1bl_hv=vhi;
137
138           vc4h_lv=vlo;
139           vc4l_lv=vhi;
140           vc2h_lv=vlo;
141           vc2l_lv=vhi;
142           vc1ah_lv=vlo;
143           vc1al_lv=vhi;
144           vc1bh_lv=vlo;
145           vc1bl_lv=vhi;
146
147           vcsamplev=vhi;
148           vccmv=vlo;
149         end
150
151       end
152
153       else if (cnt==3) begin //set output
154         vc4h_hv=vlo;
155         vc4l_hv=vlo;
156         vc2h_hv=vlo;
157         vc2l_hv=vlo;
158         vc1ah_hv=vlo;
159         vc1al_hv=vlo;
160         vc1bh_hv=vlo;
161         vc1bl_hv=vlo;
162
163         vc4h_lv=vlo;
164         vc4l_lv=vlo;
165         vc2h_lv=vlo;
166         vc2l_lv=vlo;
167         vc1ah_lv=vlo;
168         vc1al_lv=vlo;
169         vc1bh_lv=vlo;
170         vc1bl_lv=vlo;
171
172         vcsamplev=vlo;
173         vccmv=vhi;
174       end
175       //set aoll outputs
176
177       V(vc4h_h) <+ transition(vc4h_hv   , tdelay, trise, tfall);
178       V(vc4l_h) <+ transition(vc4l_hv   , tdelay, trise, tfall);
179       V(vc2h_h) <+ transition(vc2h_hv   , tdelay, trise, tfall);
180       V(vc2l_h) <+ transition(vc2l_hv   ,tdelay, trise, tfall);
```

```
181     V(vc1ah_h)  <+ transition(vc1ah_hv , tdelay, trise, tfall);
182     V(vc1al_h)  <+ transition(vc1al_hv , tdelay, trise, tfall);
183     V(vc1bh_h)  <+ transition(vc1bh_hv , tdelay, trise, tfall);
184     V(vc1bl_h)  <+ transition(vc1bl_hv , tdelay, trise, tfall);
185
186     V(vc4h_l) <+ transition(vc4h_lv   , tdelay, trise, tfall);
187     V(vc4l_l) <+ transition(vc4l_lv   , tdelay, trise, tfall);
188     V(vc2h_l) <+ transition(vc2h_lv   , tdelay, trise, tfall);
189     V(vc2l_l) <+ transition(vc2l_lv   , tdelay, trise, tfall);
190     V(vc1ah_l)  <+ transition(vc1ah_lv , tdelay, trise, tfall);
191     V(vc1al_l)  <+ transition(vc1al_lv , tdelay, trise, tfall);
192     V(vc1bh_l)  <+ transition(vc1bh_lv , tdelay, trise, tfall);
193     V(vc1bl_l)  <+ transition(vc1bl_lv , tdelay, trise, tfall);
194
195     V(vcsample) <+ transition(vcsamplev , tdelay, trise, tfall);
196     V(vccm)   <+ transition(vccmv      , tdelay, trise, tfall);
197
198     V(c1_out)<+ transition(c1_outv, tdelay, trise, tfall);
199     V(c0_out)<+ transition(c0_outv, tdelay, trise, tfall);
200     V(cnt_o)<+ transition(cnt_ov , tdelay, trise, tfall);
201
202
203   end
204
205 endmodule
```

C Charge redistribution and CBSC control Verilog-A code

Listing 8: Charge redistribution and CBSC control Verilog-A code

```
1  // VerilogA fosr highlevel_adc, sar_ctrl, veriloga
2
3  `include
4  `include
5  `define   RISING   +1
6  `define   FALLING  -1
7
8  module sar_ctrl(cnt_o,c0,c1,clk, vc4h_h,vc4l_h,vc2h_h,vc2l_h,
       vc1ah_h,vc1al_h,vc1bh_h,vc1bl_h,vc4h_l,vc4l_l,vc2h_l,vc2l_l,
       vc1ah_l,vc1al_l,vc1bh_l,vc1bl_l, vcsample,vccm, c0_out,
       c1_out, sync, cbsc_samp,cbsc_rst,cbsc_feedback,cbsc_current);
9    input c0,c1,clk,sync;
10   output cnt_o,vc4h_h,vc4l_h,vc2h_h,vc2l_h,vc1ah_h,vc1al_h,
         vc1bh_h,vc1bl_h,vc4h_l,vc4l_l,vc2h_l,vc2l_l,vc1ah_l,vc1al_l
         ,vc1bh_l,vc1bl_l,vcsample,vccm,c0_out, c1_out, cbsc_samp,
         cbsc_rst,cbsc_feedback,cbsc_current;
11   electrical cnt_o, c0,c1,clk, vc4h_h,vc4l_h,vc2h_h,vc2l_h,
         vc1ah_h,vc1al_h,vc1bh_h,vc1bl_h,vc4h_l,vc4l_l,vc2h_l,vc2l_l
         ,vc1ah_l,vc1al_l,vc1bh_l,vc1bl_l, vcsample,vccm,c0_out,
         c1_out,sync, cbsc_samp,cbsc_rst,cbsc_feedback,cbsc_current;
12   parameter real vhi=1.5;
13   parameter real vlo=0.0;
14   parameter real tdelay=0.0;
15   parameter real trise=1n;
16   parameter real tfall=1n;
17   integer cnt;
18   real c0_reg,c1_reg;
19   real cnt_ov,vc4h_hv,vc4l_hv,vc2h_hv,vc2l_hv,vc1ah_hv,vc1al_hv,
         vc1bh_hv,vc1bl_hv,vc4h_lv,vc4l_lv,vc2h_lv,vc2l_lv,vc1ah_lv,
         vc1al_lv,vc1bh_lv,vc1bl_lv,vcsamplev,vccmv,c0_outv, c1_outv
         , cbsc_sampv,cbsc_rstv,cbsc_feedbackv,cbsc_currentv;
20
21   analog begin
22         @(initial_step(     ,     ,     ,     )) begin
23         cnt=0;
24       end
25     @( cross(V(clk)-vhi/2, `RISING)) begin
26       if (cnt==1) begin
27         c1_reg=V(c1);
28         c0_reg=V(c0);
29       end
30       if (V(sync)>vhi/2) begin
```

```
31          cnt=0;
32        end
33        else begin
34          cnt=(cnt+1);
35        end
36
37      end
38        cnt_ov=cnt;
39        c0_outv=c0_reg;
40      c1_outv=c1_reg;
41
42      if(cnt==0)
43      begin //generate comparator voltages
44            vc4h_hv=vhi;
45        vc4l_hv=vlo;
46        vc2h_hv=vlo;
47        vc2l_hv=vhi;
48        vc1ah_hv=vhi;
49        vc1al_hv=vlo;
50        vc1bh_hv=vlo;
51        vc1bl_hv=vhi;
52
53        vc4h_lv=vlo;
54        vc4l_lv=vhi;
55        vc2h_lv=vhi;
56        vc2l_lv=vlo;
57        vc1ah_lv=vhi;
58        vc1al_lv=vlo;
59        vc1bh_lv=vlo;
60        vc1bl_lv=vhi;
61
62        vcsamplev=vhi;
63        vccmv=vlo;
64
65        // CBSC controll signals
66        cbsc_sampv=vhi;
67        cbsc_rstv=vhi;
68        cbsc_feedbackv=vlo;
69        cbsc_currentv=vlo;
70
71      end
72      else if (cnt==1) begin //compare
73        vc4h_hv=vlo;
74        vc4l_hv=vlo;
75        vc2h_hv=vlo;
76        vc2l_hv=vlo;
77        vc1ah_hv=vlo;
78        vc1al_hv=vlo;
79        vc1bh_hv=vlo;
```

```
80          vc1bl_hv=vlo;
81
82          vc4h_lv=vlo;
83          vc4l_lv=vlo;
84          vc2h_lv=vlo;
85          vc2l_lv=vlo;
86          vc1ah_lv=vlo;
87          vc1al_lv=vlo;
88          vc1bh_lv=vlo;
89          vc1bl_lv=vlo;
90
91          vcsamplev=vlo;
92          vccmv=vhi;
93
94
95          // CBSC controll signals
96          cbsc_sampv=vhi;
97          cbsc_rstv=vhi;
98          cbsc_feedbackv=vlo;
99          cbsc_currentv=vlo;
100
101       end
102       else if (cnt==2||cnt==3) begin // generate output voltages
103         if (c0_reg<vhi/2 && c1_reg < vhi/2) begin //0
104           vc4h_hv=vhi;
105           vc4l_hv=vlo;
106           vc2h_hv=vhi;
107           vc2l_hv=vlo;
108           vc1ah_hv=vhi;
109           vc1al_hv=vlo;
110           vc1bh_hv=vhi;
111           vc1bl_hv=vlo;
112
113           vc4h_lv=vlo;
114           vc4l_lv=vhi;
115           vc2h_lv=vhi;
116           vc2l_lv=vlo;
117           vc1ah_lv=vhi;
118           vc1al_lv=vlo;
119           vc1bh_lv=vhi;
120           vc1bl_lv=vlo;
121
122           vcsamplev=vhi;
123           vccmv=vlo;
124         end
125         else if (c0_reg>vhi/2 && c1_reg < vhi/2) begin //2
126           vc4h_hv=vhi;
127           vc4l_hv=vlo;
128           vc2h_hv=vhi;
```

```
129           vc2l_hv=vlo;
130           vc1ah_hv=vlo;
131           vc1al_hv=vhi;
132           vc1bh_hv=vlo;
133           vc1bl_hv=vhi;
134
135           vc4h_lv=vlo;
136           vc4l_lv=vhi;
137           vc2h_lv=vhi;
138           vc2l_lv=vlo;
139           vc1ah_lv=vlo;
140           vc1al_lv=vhi;
141           vc1bh_lv=vlo;
142           vc1bl_lv=vhi;
143
144           vcsamplev=vhi;
145           vccmv=vlo;
146         end
147         else  if (c0_reg>vhi/2 && c1_reg > vhi/2) begin //4
148           vc4h_hv=vhi;
149           vc4l_hv=vlo;
150           vc2h_hv=vlo;
151           vc2l_hv=vhi;
152           vc1ah_hv=vlo;
153           vc1al_hv=vhi;
154           vc1bh_hv=vlo;
155           vc1bl_hv=vhi;
156
157           vc4h_lv=vlo;
158           vc4l_lv=vhi;
159           vc2h_lv=vlo;
160           vc2l_lv=vhi;
161           vc1ah_lv=vlo;
162           vc1al_lv=vhi;
163           vc1bh_lv=vlo;
164           vc1bl_lv=vhi;
165
166           vcsamplev=vhi;
167           vccmv=vlo;
168         end
169
170         // CBSC controll signals
171         cbsc_sampv=vhi;
172         cbsc_rstv=vhi;
173         cbsc_feedbackv=vlo;
174         cbsc_currentv=vlo;
175
176
177       end
```

```
178
179     else if (cnt==4) begin //set output
180       vc4h_hv=vlo;
181       vc4l_hv=vlo;
182       vc2h_hv=vlo;
183       vc2l_hv=vlo;
184       vc1ah_hv=vlo;
185       vc1al_hv=vlo;
186       vc1bh_hv=vlo;
187       vc1bl_hv=vlo;
188
189       vc4h_lv=vlo;
190       vc4l_lv=vlo;
191       vc2h_lv=vlo;
192       vc2l_lv=vlo;
193       vc1ah_lv=vlo;
194       vc1al_lv=vlo;
195       vc1bh_lv=vlo;
196       vc1bl_lv=vlo;
197
198       vcsamplev=vlo;
199       vccmv=vhi;
200
201
202       // CBSC controll signals
203       cbsc_sampv=vhi;
204       cbsc_rstv=vhi;
205       cbsc_feedbackv=vlo;
206       cbsc_currentv=vlo;
207
208     end
209
210     else if(cnt==5)begin
211         vc4h_hv=vhi;
212       vc4l_hv=vlo;
213       vc2h_hv=vlo;
214       vc2l_hv=vhi;
215       vc1ah_hv=vhi;
216       vc1al_hv=vlo;
217       vc1bh_hv=vlo;
218       vc1bl_hv=vhi;
219
220       vc4h_lv=vlo;
221       vc4l_lv=vhi;
222       vc2h_lv=vhi;
223       vc2l_lv=vlo;
224       vc1ah_lv=vhi;
225       vc1al_lv=vlo;
226       vc1bh_lv=vlo;
```

```
227         vc1bl_lv=vhi;
228
229         vcsamplev=vhi;
230         vccmv=vlo;
231
232         // CBSC controll signals
233         cbsc_sampv=vlo;
234         cbsc_rstv=vlo;
235         cbsc_feedbackv=vhi;
236         cbsc_currentv=vhi;
237
238       end
239       else if(cnt==6)begin
240           vc4h_hv=vhi;
241         vc4l_hv=vlo;
242         vc2h_hv=vlo;
243         vc2l_hv=vhi;
244         vc1ah_hv=vhi;
245         vc1al_hv=vlo;
246         vc1bh_hv=vlo;
247         vc1bl_hv=vhi;
248
249         vc4h_lv=vlo;
250         vc4l_lv=vhi;
251         vc2h_lv=vhi;
252         vc2l_lv=vlo;
253         vc1ah_lv=vhi;
254         vc1al_lv=vlo;
255         vc1bh_lv=vlo;
256         vc1bl_lv=vhi;
257
258         vcsamplev=vhi;
259         vccmv=vlo;
260
261         // CBSC controll signals
262         cbsc_sampv=vlo;
263         cbsc_rstv=vlo;//hi?
264         cbsc_feedbackv=vhi;
265         cbsc_currentv=vlo;
266
267       end
268       else if(cnt==7)begin
269           vc4h_hv=vhi;
270         vc4l_hv=vlo;
271         vc2h_hv=vlo;
272         vc2l_hv=vhi;
273         vc1ah_hv=vhi;
274         vc1al_hv=vlo;
275         vc1bh_hv=vlo;
```

```
276         vc1bl_hv=vhi;
277
278         vc4h_lv=vlo;
279         vc4l_lv=vhi;
280         vc2h_lv=vhi;
281         vc2l_lv=vlo;
282         vc1ah_lv=vhi;
283         vc1al_lv=vlo;
284         vc1bh_lv=vlo;
285         vc1bl_lv=vhi;
286
287         vcsamplev=vhi;
288         vccmv=vlo;
289         // CBSC controll signals
290         cbsc_sampv=vlo;
291         cbsc_rstv=vhi;
292         cbsc_feedbackv=vhi;
293         cbsc_currentv=vlo;
294
295       end
296       else if(cnt==8)begin
297           vc4h_hv=vhi;
298         vc4l_hv=vlo;
299         vc2h_hv=vlo;
300         vc2l_hv=vhi;
301         vc1ah_hv=vhi;
302         vc1al_hv=vlo;
303         vc1bh_hv=vlo;
304         vc1bl_hv=vhi;
305
306         vc4h_lv=vlo;
307         vc4l_lv=vhi;
308         vc2h_lv=vhi;
309         vc2l_lv=vlo;
310         vc1ah_lv=vhi;
311         vc1al_lv=vlo;
312         vc1bh_lv=vlo;
313         vc1bl_lv=vhi;
314
315         vcsamplev=vhi;
316         vccmv=vlo;
317
318         // CBSC controll signals
319         cbsc_sampv=vlo;
320         cbsc_rstv=vlo;
321         cbsc_feedbackv=vhi;
322         cbsc_currentv=vhi;
323
324       end
```

```
325     else if(cnt==9)begin
326         vc4h_hv=vhi;
327       vc4l_hv=vlo;
328       vc2h_hv=vlo;
329       vc2l_hv=vhi;
330       vc1ah_hv=vhi;
331       vc1al_hv=vlo;
332       vc1bh_hv=vlo;
333       vc1bl_hv=vhi;
334
335       vc4h_lv=vlo;
336       vc4l_lv=vhi;
337       vc2h_lv=vhi;
338       vc2l_lv=vlo;
339       vc1ah_lv=vhi;
340       vc1al_lv=vlo;
341       vc1bh_lv=vlo;
342       vc1bl_lv=vhi;
343
344       vcsamplev=vhi;
345       vccmv=vlo;
346
347       // CBSC controll signals
348       cbsc_sampv=vlo;
349       cbsc_rstv=vlo;
350       cbsc_feedbackv=vhi;
351       cbsc_currentv=vlo;
352
353     end
354     //set aoll outputs
355
356     V(vc4h_h) <+ transition(vc4h_hv   , tdelay, trise, tfall);
357     V(vc4l_h) <+ transition(vc4l_hv   , tdelay, trise, tfall);
358     V(vc2h_h) <+ transition(vc2h_hv   , tdelay, trise, tfall);
359     V(vc2l_h) <+ transition(vc2l_hv   ,tdelay, trise, tfall);
360     V(vc1ah_h)  <+ transition(vc1ah_hv , tdelay, trise, tfall);
361     V(vc1al_h)  <+ transition(vc1al_hv , tdelay, trise, tfall);
362     V(vc1bh_h)  <+ transition(vc1bh_hv , tdelay, trise, tfall);
363     V(vc1bl_h)  <+ transition(vc1bl_hv , tdelay, trise, tfall);
364
365     V(vc4h_l) <+ transition(vc4h_lv   , tdelay, trise, tfall);
366     V(vc4l_l) <+ transition(vc4l_lv   , tdelay, trise, tfall);
367     V(vc2h_l) <+ transition(vc2h_lv   , tdelay, trise, tfall);
368     V(vc2l_l) <+ transition(vc2l_lv   , tdelay, trise, tfall);
369     V(vc1ah_l)  <+ transition(vc1ah_lv , tdelay, trise, tfall);
370     V(vc1al_l)  <+ transition(vc1al_lv , tdelay, trise, tfall);
371     V(vc1bh_l)  <+ transition(vc1bh_lv , tdelay, trise, tfall);
372     V(vc1bl_l)  <+ transition(vc1bl_lv , tdelay, trise, tfall);
373
```

```
374     V(vcsample) <+ transition(vcsamplev , tdelay, trise, tfall);
375     V(vccm)   <+ transition(vccmv      , tdelay, trise, tfall);
376
377     V(c1_out)<+ transition(c1_outv, tdelay, trise, tfall);
378     V(c0_out)<+ transition(c0_outv, tdelay, trise, tfall);
379     V(cnt_o)<+ transition(cnt_ov , tdelay, trise, tfall);
380
381
382     V(cbsc_samp)  <+ transition(cbsc_sampv     , tdelay, trise,
            tfall);
383     V(cbsc_rst)   <+ transition(cbsc_rstv      , tdelay, trise,
            tfall);
384     V(cbsc_feedback)<+ transition(cbsc_feedbackv, tdelay, trise,
            tfall);
385     V(cbsc_current) <+ transition(cbsc_currentv , tdelay, trise,
            tfall);
386
387
388   end
389
390 endmodule
```

D Matlab code for calculations of dynamic parameters

Listing 9: Matlab code for calculations of dynamic parameters

```
function [ENOB,SINAD]=dyn_calc(x,Fs,bit)


  L=length(x);
  t=(0:L-1)/Fs;

  figure(1)
  plot(Fs*t(1:50),x(1:50))
  title(                                                    )
  xlabel(                          )
  NFFT = 2^(nextpow2(L)-1); % Next power of 2 from length of y
  x=x(L-NFFT:L);
  Y = fft(x,NFFT)/NFFT/2^bit;
  P=abs(Y.*Y);
  f = Fs/2*linspace(0,1,NFFT/2+1);
  fin=0;
  spanh=2;
  [devnull,index_fin]=max(P(1+spanh:NFFT))
  index_fin=index_fin+spanh
  fin=f(index_fin)
  P=abs(Y.*Y);
  Pdc=sum(P(1:1+spanh))
  Ps=sum(P(index_fin-spanh:index_fin+spanh))
  Pharm=zeros(1,7);
  for i=2:7
    if (index_fin*i -spanh< NFFT/2)
      if(index_fin*i+spanh<NFFT/2)
        Pharm(i)=sum(P(index_fin*i-spanh:index_fin*i+spanh));
      else
        Pharm(i)=sum(P(index_fin*i-spanh:NFFT/2));
      end
    end
  end

  Pharm
  Ph=sum(Pharm)
  Pn=sum(P(1:NFFT/2+1))-Ps-Pdc-Ph
  SNR=10*log10(Ps/Pn)
  THD=10*log10(Ph/Ps)
  SNDR=10*log10(Ps/(Pn+Ph))
  SINAD=10*log10((Ps+Pn+Ph)/(Pn+Ph))
  ENOB=(SNDR-1.76)/6.02
```

```
43    SFDR=10*log10(Ps)-10*log10(max(Pharm))
44    % Plot single-sided amplitude spectrum.
45
46    length(Y)
47
48    for i=1:length(Y)
49
50       if Y(i)==0 && i>1
51          Y(i)=Y(i-1);
52       elseif Y(i)==0 &&i==1
53          Y(i)=1e-9;
54       end
55
56
57
58
59
60    end
61    P=abs(Y.*Y);
62    figure(2)
63    plot(f,10*log10((2*abs(P(1:NFFT/2+1))))-10*log10(Ps))
64    title({       ;[          ,num2str(SNR),          ,num2str(SNDR),
            ,num2str(ENOB)];[          ,num2str(fin,      ),          ,num2str
          (THD),          ,num2str(SFDR)]})
65    xlabel(                    )
66    ylabel(          )
67    return
68
69
70 end
```

E Schematic of the Converter stage

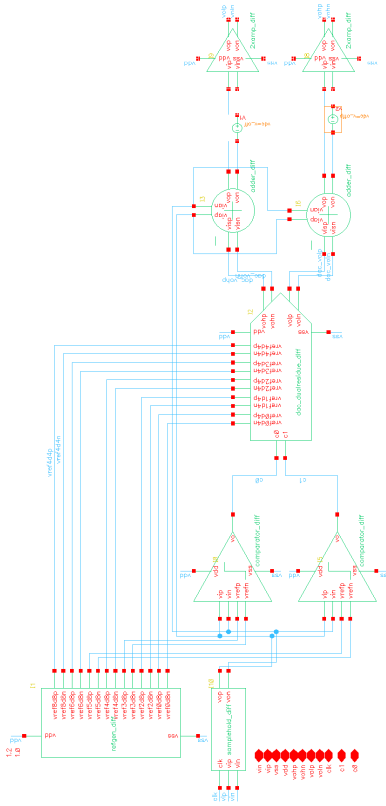


Figure 17: Schematic of the Converter stage

F Schematic of the Dual-residue stage

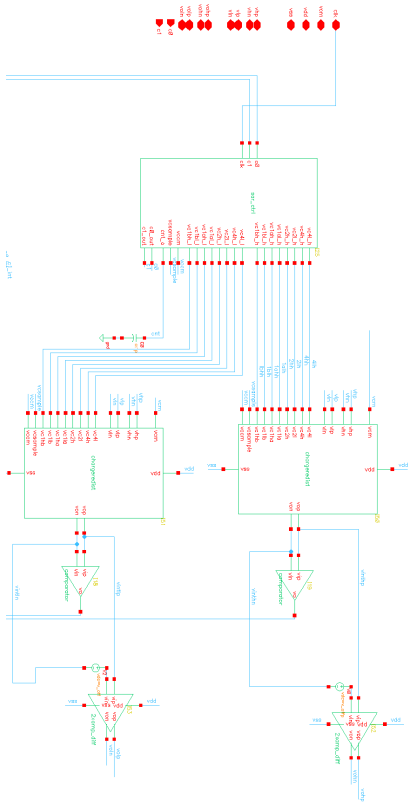


Figure 18: Schematic of the Dual-residue stage