An Approximate Quantizer for High Speed ΣΔ Modulators with Carry Save Arithmetic

by

Dan Zupcau

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

Ottawa-Carleton Institute for Electrical Engineering
Department of Electronics
Carleton University
Ottawa, Ontario
Canada K1S 5B6

October 1998

0-612-36915-3

The undersigned recommend to the Faculty of Graduate Studies

and Research acceptance of the thesis

An Approximate Quantizer for High Speed $\Sigma\Delta$ Modulators with Carry Save Arithmetic

submitted by Dan Zupcau (B.Eng.)

in partial fulfillment of the requirements

for the degree of Master of Engineering

Chairman,
Department of Electronics
Professor Jim Wight

Thesis Co-Supervisor
Professor Martin Snelgrove

Thesis Co-Supervisor
Professor Ralph Mason

Ottawa-Carleton Institute for Electrical Engineering
Department of Electronics
Faculty of Engineering
Carleton University
October 1998

Abstract

There is a growing market demand for wideband D/A conversion at resolutions in excess of 12 bits, especially for video and radio applications. ΣΔ D/A conversion is known for providing high resolution in a certain bandwidth. The maximum resolution bandwidth of a ΣΔ DAC is dictated by the maximum speed of its digital ΣΔ modulator, which is in turn limited by the speed of its accumulators.

This thesis demonstrates that by using carry save arithmetic and approximate quantization in a digital ΣΔ modulator, the achievable increase in speed compared to the state of the art is roughly a factor of two.

The method was verified in a particular design case, a ΣΔ DAC using a second order carry save ΣΔ modulator, implemented in 0.5μ CMOS. The measured performance of the carry save ΣΔ modulator is 73.53 dB peak SNR (12 bits of resolution) in a 4.68 MHz bandwidth at a 150 MHz sampling frequency.

Acknowledgments

Financial support for this work has been provided by Micronet, a National Center for Excellence program of the National Science and Engineering Research Council of Canada (NSERC). The integrated circuit was fabricated through MOSIS at the University of Southern California. The 0.5μ CMOS technology cell library and tools were provided by CMC (Canadian Microelectronics Corporation) and the test equipment was supplied by NSERC.

I had the great opportunity to work under the supervision of Dr. Martin Snelgrove. I would like to thank him for his guidance, patience and support throughout my thesis.

I would like to thank Dr. Ralph Mason for his advice on thesis writing.

Many thanks to Peter Nyasulu for his always friendly and modest way of explaining to me the details of Synopsys tools, to Philip Lauzon for the help with the board design, to Alana Wirta for help with the paperwork and to all the other people in the department. Not in the last place, I would like to thank Cristian Cojocaru from Philsar for his advice on chip work.

Also, I would like to acknowledge the help of Walter Fergusson with the CAE tools and the UNIX system and of Nagui Michail for technical support.

In the end, I would like to thank my family for their love and moral support.

Glossary of Terms

A/D	Analog to Digital
ADC	Analog to Digital Converter
AC	Alternating Current
AWGN	Additive White Gaussian Noise
APR	Automatic Place and Route
BJT	Bipolar Junction Transistor
CD	Compact Disc
CLA	Carry Look Ahead Adder
CPA	Carry Propagate Arithmetic
CT	Continuous Time
CS	Carry Save

MOS	Metal Oxide Semiconductor
MSB	Most Significant Bit
NTF	Noise Transfer Function
PCB	Printed Circuit Board
PSRR	Power Supply Rejection Ratio
RMS	Root Mean Square
STF	Signal Transfer Function
SSB	Single Side Bandwidth
$\Sigma\Delta$	Sigma Delta

List of Figures

CHAPTER 1 Introduction

1.1. Problem description

Digital to analog converters are critical elements in data conversion systems, making the interface between the digital computational world and the real analog world. High speed, high resolution DACs are increasingly finding their way into high frequency applications, such as spread spectrum communications or waveform synthesis. A trade-off is to be made when choosing a DAC, depending on the specific application field. For example, CD players use relatively low sampling frequency DACs with 16 to 24 bits of resolution, whereas higher sampling frequency DACs used in arbitrary waveform synthesis or DDS have typically 8 to 12 bits of resolution.

The technical problem addressed in this thesis is the feasibility of increasing the maximum sampling frequency of a particular high resolution DAC architecture, while maintaining its in-band resolution in the order of 16 bits. In other words, is it possible to convert, with high resolution, wider bandwidths of digitized signals to analog form?

1.2. Proposed solution

ΣΔ techniques, used in ΣΔ data converters, are known to provide high resolution in a certain frequency bandwidth by pushing the quantization noise to ranges of frequencies from which it can be easily filtered out. Practical implementation of such data converters trade a higher oversampling rate for increased in-band resolution. Computations are done mainly in the digital domain and the requirements on the precision of analog components are considerably reduced, compared to other types of data converters. On the other hand, this type of data converter is limited by the range of sampling frequencies. On the ΣΔ A/D side, however, it has been proven that GHz range sampling is feasible [Wein95].

It will be shown that the method proposed in this thesis allows for a substantial increase in maximum sampling frequency of digital ΣΔ modulators, the fundamental building blocks of ΣΔ DACs.

1.3. Contributions

The thesis demonstrates:

a) A new digital ΣΔ modulator architecture which employs carry save arithmetic. This was the main research focus. In this type of arithmetic computations are done in parallel with the data flow, as opposed to classical carry propagate arithmetic, in which computations are done orthogonal to data flow. The key innovation is the use of CSA approximate quantization in the proposed circuit, instead of exact quantization, as in CPA digital ΣΔ modulators. For small enough approximate CSA quantization errors, both approximate quantization error energy and quantization error energy are shaped out of the band of interest by the same ΣΔ negative feedback effect. The method roughly doubles the maximum sampling frequency of digital ΣΔ modulators, for a given technology, compared with the state of the art.

b) To prove the technique, a particular second order ΣΔ DAC design case was chosen. Since the application is targeted for wideband and high resolution applications, the digital ΣΔ modulator uses a three bit CSA approximate quantizer. The CS ΣΔ modulator, part of the 0.5μ CMOS implementation of the ΣΔ DAC, works correctly up to 145 MHz sampling frequency. The measured digital performance is limited by I/O pad speed.

The mismatch noise issue, linked to the use of a multibit ΣΔ modulator, is addressed by using a ΣΔ-based bit shuffling network cascaded with the digital ΣΔ modulator.

c) The circuit implementation addresses specific layout problems of mixed signal design, such as substrate coupling, component matching, etc. The author followed the whole design flow, from problem formulation to circuit testing.

1.4. Thesis organization

The thesis is divided into seven chapters.

Chapter 2 provides general D/A background useful for high speed DACs: digital representation formats, current steering DACs and factors limiting their linearity.

Chapter 3 introduces the reader to ΣΔ D/A conversion and carry save arithmetic. Fundamental ΣΔ conversion issues, such as quantization, oversampling and noise shaping are presented. First order and second order digital ΣΔ modulator architectures are then introduced, highlighting the advantages of the latter. In this context, a state of the art circuit implementation of a second order ΣΔ modulator is presented. An example of a bit shuffling algorithm using dynamic element matching is briefly discussed. At the end of the chapter, a brief section gives an overview of carry save arithmetic.

Chapter 4 is the core of the thesis. This chapter starts by laying out the mathematical foundation for the use of CSA in digital ΣΔ modulators. The CSA implementations of the three building blocks of a digital ΣΔ modulator (integrator, approximate quantizer and approximate clipper) are shown. The theoretical arguments are enforced by simulations done on a particular design case, a second order digital ΣΔ modulator. The speed advantage of the circuit is demonstrated by comparing it to that of the state of the art circuit discussed in chapter 3.

Chapter 5 describes the ΣΔ-based bit shuffling network. A theoretical section introduces the reader to the method. The circuit implementation and simulation results are presented at the end of the chapter.

Chapter 6 is dedicated to experimental results. Both a Xilinx 4010E prototype and the 0.5μ CMOS implementation are discussed.

Chapter 7 summarizes the research and discusses future work that can be done, with a focus on digital radio applications.

CHAPTER 2 D/A conversion background

2.1. Introduction

A/D and D/A converters provide the interface between the analog signal domain and the binary digital world. The outputs of digital circuits must often be converted by DACs to drive analog actuators such as a speakers, video displays, final stages of radio transmitter chains (power amplifier and antenna), etc. A Nyquist rate DAC (with output updating frequency close to, but higher than the Nyquist frequency of the analog signal to be reproduced) produces an analog output O which is proportional to the input digital word D:

$$O = K \cdot D \tag{2.1}$$

where K is the proportionality factor which can be voltage, electrical charge or current. K sets both the dimension and the full scale value of the output O. If the digital input D is in binary format, it is common practice to normalize D with respect to its full scale value, 2^m-1, where m is the number of bits of D. For example, if K is a voltage reference

$$O = \frac{D \cdot V_{ref}}{2^m - 1} \tag{2.2}$$

The input word D can be expressed in any digital form easily convertible to analog. A positive integer N can be represented with $m = 1 + \lfloor \log_2(N) \rfloor$ bits in binary code as:

$$N = \sum_{i=0}^{m-1} D_i \cdot 2^i \qquad (2.3)$$

where $\lfloor \; \rfloor$ is the round to floor integer operation. A signed integer N can be represented in two's complement form using $r = 2 + \lfloor \log_2(|N|) \rfloor$ bits, with MSB the sign bit:

```
If N >= 0 then
  $D_{r-1}$ = 0; sign bit of N is 0;
  $D_{r-2}$ $D_{r-3}$..... $D_0$ = binary representation of N on
                         $m = 1 + \lfloor \log_2(|N|) \rfloor$ bits;
else
  N = -N;   compute -N;
  $P_{r-1}$ = 0; sign bit of -N is 0;
  $P_{r-2}$ $P_{r-3}$..... $P_0$ = binary representation of -N on
                         $m = 1 + \lfloor \log_2(|N|) \rfloor$ bits;
  $D_{r-1}$ $D_{r-3}$..... $D_0$ =!($P_{r-1}P_{r-2}$....$P_0$) + 1 LSB;
end;
```

Fig. 2.1. 2's complement definition

where ! is the bitwise negation operation. Inverting the sign bit of an m bit two's complement number, one gets a binary representation of that number's integer value plus 2^{m-1}. For example, inverting the sign bit of 100 (-4 in two's complement form using three bits) one gets 000 (0 in three bit binary code), thus a translation of $4 = 2^{3-1}$. This observation will be used later in the thesis.

i

An integer N can be represented by a thermometer code of length n >= N (n >=1 for N = 0).

```
If i < N then
        $T_i$ = 1;
else
        $T_i$ = 0;
end;                  i= 0......n;                          (2.4)
```

The output of the bit shuffling network used to implement the ΣΔ DAC proposed in this work is expressed using an m out of n code. In such a form of representation, a positive integer N is represented on n >= N bits (m >= 1 for N = 0) in such a way as m = N bits have a logic ONE value and the rest have a logic ZERO value. The following relation holds:

$$N = \sum_{i=0}^{n-1} R_i$$

2.2. Important DAC specifications

DAC performance specifications can be divided into static and dynamic specifications. There is a wealth of information in the literature about DAC static performance specifications such as resolution, DNL, INL, PSSR, etc. ([Hoes94], [Raza95]). More relevant for a ΣΔ DAC are the dynamic specifications presented below:

1) The most important dynamic specification of a DAC is the signal to noise ratio [SNR], the ratio between the signal power and the total quantization noise power (see Section 3.2) in the band of interest. It depends on the resolution of the DAC and automatically includes specifications such as linearity, distortion, glitches, noise, etc. Over half the sampling

frequency, the maximum SNR should ideally follow the theoretical formula [Plas94]:

$$SNR_{max}(dB) = 6.02 \cdot N + 1.76 \qquad (2.5)$$

Figure 2.2. shows the dependence of a DAC output as a function of normalized sinusoidal input level. While the dependence is linear for small input levels, it rolls-off sharply for large input levels. Linear quantizers, discussed in Section 3.2., exhibit the same behavior.

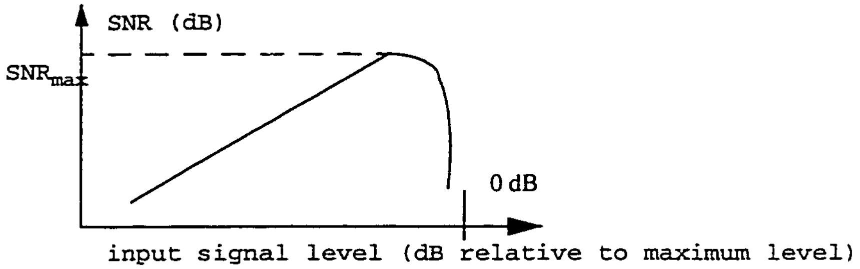


Fig.2.2 . Representative curve of SNR versus signal level

An even more stringent dynamic performance measure is obtained if the total harmonic distortion power is added to the quantization error power in the band of interest, thus computing the signal to noise and harmonic distortion ratio [SNHR].

2)When DACs are used with large oversampling ratios or the spectral purity is important, the spurious free dynamic range [SFDR] is used to define the ratio between the maximum signal component and the largest distortion component. SNR and SFDR in a 200 KHz bandwidth are shown in Figure 2.3. for a Nyquist DAC.Additional DAC dynamic performance specifications can be found in the literature [Plas94], [Raza95].

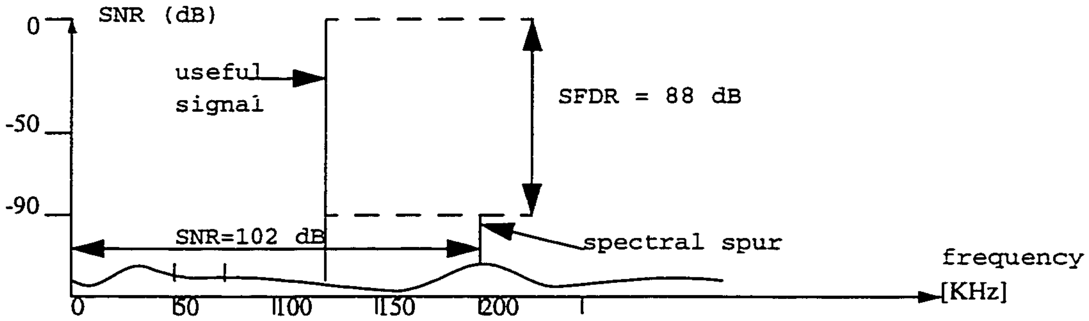


Fig.2.3. Definitions of SNR and SFDR for a Nyquist DAC

2.3. Current division DAC

Current division reconstruction DAC stages are commonly used in high speed DACs, such as the ΣΔ DAC proposed in this thesis. Current switching is preferred in high speed applications over other reconstruction DAC architectures (eg. charge division) for its good high frequency performance [Raza95]. Since high D/A conversion resolution is the other important issue addressed in this work, knowing the sources of error in practical current steering DAC implementations is necessary. The D/A reconstruction stage shown in Figure 2.4. has two blocks: a two's complement (or binary) to thermometer decoder and a current steering array with nominally identical current sources (segmented current steering array).

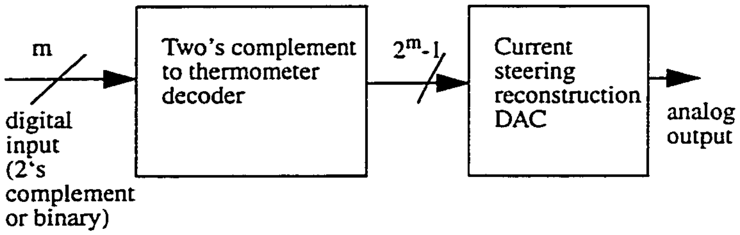

m
Two's complement to thermometer decoder
2^m-1
Current steering reconstruction DAC
digital input (2's complement or binary)
analog output

Fig.2.4. Current steering D/A reconstruction stage

The use of thermometer code ensures reconstruction DAC monotonicity since digital input increments simply cause additive increments in the analog output [Raza95]. The use of m out of n code and segmented current steering arrays has the same advantage, and it was chosen in the ΣΔ DAC implementation proposed in this work. Figure 2.5.a. shows a segmented current steering array: N current mirrors that generate N nominally identical output currents, the sum of which is equal to I_{out}. The thermometer code value (D_N.....D_1) controls which current sources are switched to the output.

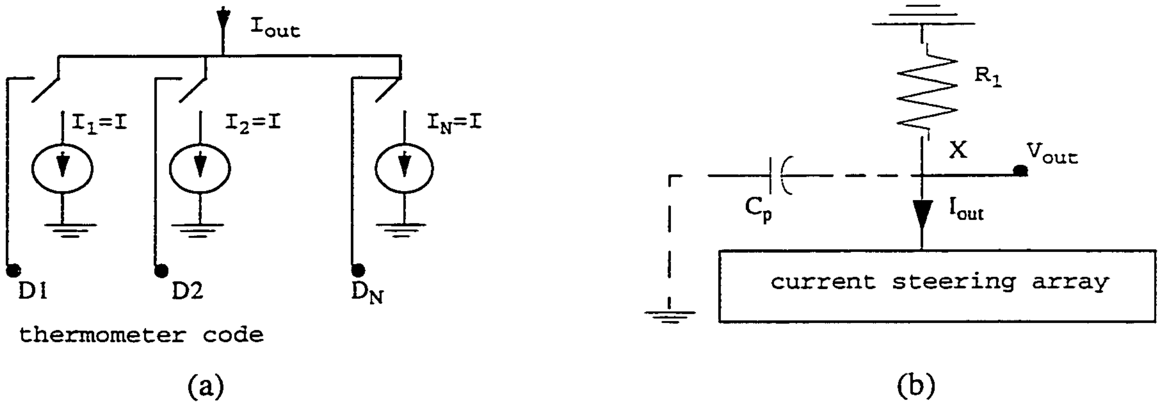


Fig.2.5. (a) Segmented current steering array; (b) conversion of the output current to voltage using a resistor.

In Figure 2.5.b, resistor R_l is used to convert to voltage the output current of the current steering array. The output settling time is limited by the total parasitic capacitance C_p at node X, because this node experiences the total output swing. This approach is common in high speed applications, and it is used in the proposed ΣΔ DAC.

The major sources of error in segmented current steering arrays are as follows:

a) Current source mismatch. In Figure 2.6, the two current sources are implemented using MOS transistors M_l and M_2, which ideally would be matched.

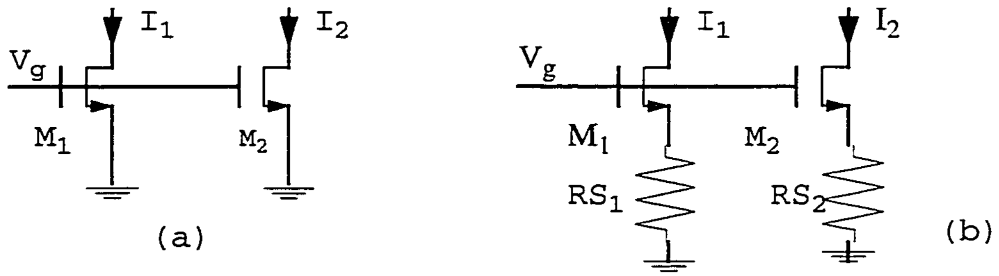


Fig.2.6. Nominally identical MOS current sources

In constant current operation, the output currents can be approximated as:

$$I_D = \frac{1}{2} \cdot \mu \cdot C_{ox} \cdot \frac{W}{L} \cdot (V_{GS} - V_{TH})^2$$

Since μ, C_{ox} and V_{TH} are constant for a given process, W, L and V_{GS} are the only parameters under designer's control and they can be increased in order to reduce the absolute variation of I_D. .ΔI process variations in the D/A reconstruction stage can have a severe impact on the in-band resolution of a ΣΔ DAC. Mismatches typically limit in-band DAC resolution to 8 - 10 bits.

b) Finite current source output impedance. If the output current summing node of a current steering array experiences large voltage excursion, then INL (maximum deviation of the input/output characteristic from a straight line passed through its end points [Raza95]) is affected by the finite output impedances of the current sources.

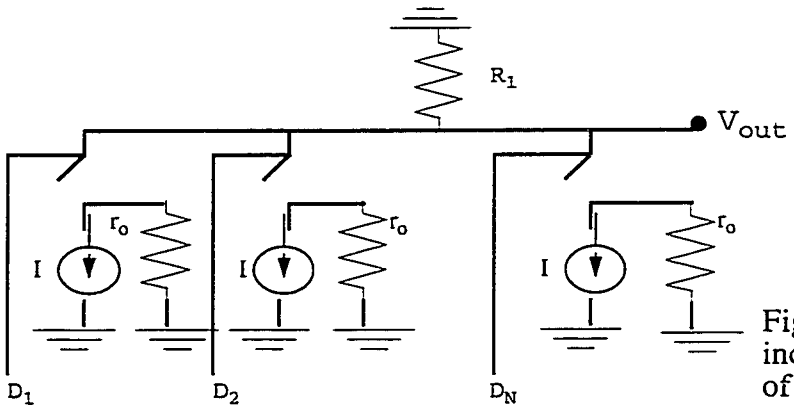


Fig.2.7. Current steering array including output impedance of each current source

The maximum value of INL of the reconstruction DAC using the segmented current steering array shown in Figure 2.7. is [Raza95]:

$$INL_{max} = I \cdot R_1^{\ 2} \cdot \frac{N^2}{4 \cdot r_o} \qquad (2.8)$$

Let us consider, for example that the MOSFET used to implement such a DAC have an Early voltage of 5V. The output impedance of such a MOSFET is

$r_0 = V_A / I$. Substituting this relation in equation (2.8), the $\mathrm{INL}_{\mathrm{max}}$ value becomes:

$$INL_{max} = \frac{(N \cdot I \cdot R)^2}{4 \cdot V_A} \qquad (2.9)$$

where the maximum output voltage is $V_{max} = N \cdot I \cdot R$. The relative error is

$$\frac{INL_{max}}{V_{max}} = \frac{V_{max}}{4 \cdot V_A} \qquad (2.10)$$

For for a maximum DAC output voltage of 500 mV, the relative error is 1/40, which is significant even for a 5 bit DAC.

Process variations and finite current source impedance effects on ΣΔ DAC linearity can be limited by using bit shuffling, as described in Chapter3 (Section 3.6) and Chapter 5.

2.4. Summary

This chapter introduced the reader to basic D/A concepts which are to be used later in the thesis. A section on digital representation formats was followed by a brief introduction to important DAC specifications. The last section showed that current switching D/A reconstruction stages, while fast, are difficult to design for resolutions beyond about eight bits. The high speed ΣΔ DAC proposed in this work uses such an analog D/A reconstruction stage.

CHAPTER 3 ΣΔ D/A and carry save arithmetic background

3.1. Introduction

For a Nyquist rate DAC (Figure 3.1.below), the output smoothing filter's characteristics are essential to the overall accuracy of the D/A conversion. If the sampling frequencies are relatively close to the Nyquist frequency (twice the maximum frequency of the signal to be converted, i.e. the minimum sampling frequency which guarantees complete signal recovery), the smoothing filter's characteristics must have high roll-off and high out-of-band attenuation. These filters therefore require high accuracy analog components.

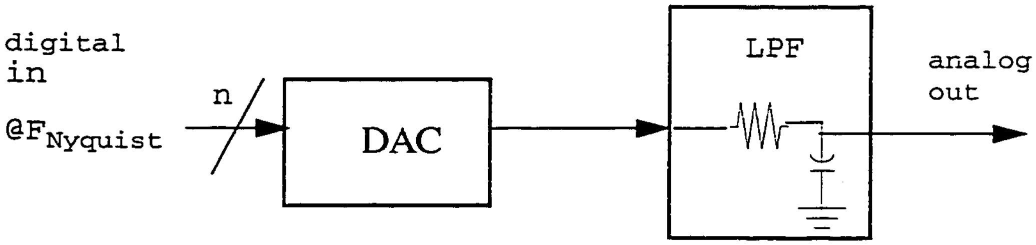


Fig.3.1. Nyquist rate DAC

The above drawback of Nyquist rate DACs is not present in oversampling ΣΔ DACs. A block diagram of a ΣΔ DAC is shown in Figure 3.2. below.

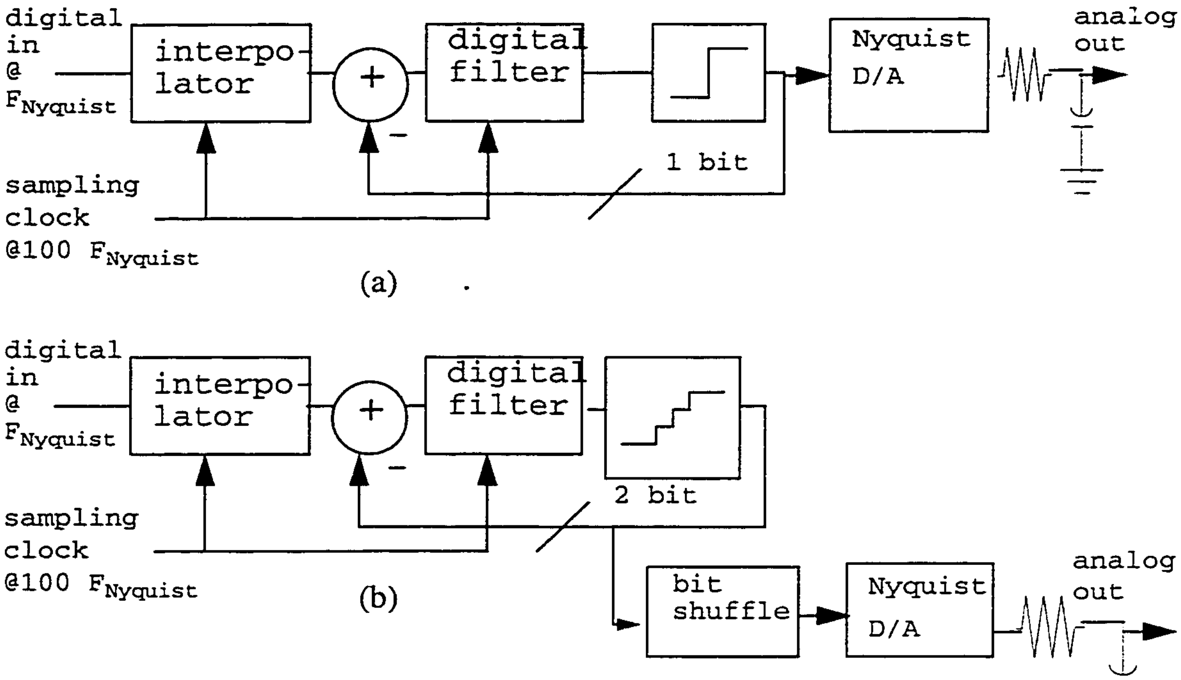


Fig.3.2. (a) ΣΔ DAC using a 1 bit digital ΣΔ modulator (b) ΣΔ DAC using a 2 bit digital ΣΔ modulator

The incoming digital data is sampled at Nyquist rate. The interpolator stage (see Section 3.4.) increases the sampling rate of the signal (chosen 100 times greater, for example). The next stage is an interpolating circuit called a digital ΣΔ modulator, working at the same speed as the interpolator, which truncates the input words into shorter output words. These shorter words change their value in such a way that the truncation noise that lies in the frequency band of interest is significantly reduced. A ΣΔ digital modulator is essentially built by cascading a digital filter and a truncation element (a 1 bit quantizer, in the case shown in Figure 3.2.a.) inside a negative feedback loop. The digital ΣΔ modulator is cascaded with a D/A reconstruction stage (see Chapter 2). The ΣΔ DAC output is taken from a LPF cascaded with the D/A reconstruction stage.The overall resolution performance of the ΣΔ DAC can be increased by increasing the amount of information fed back (i.e. increasing the number of bits quantized (Figure 3.2.b.)). The advantages are increased stability and in-band resolution. The disadvantage is the need to use an array of unit D/A elements (current sources for the circuit proposed in this thesis) in the analog stage, with their associated mismatch problem (see Chapter 2). Fortunately, the effect of unit D/A element mismatch on in-band DAC resolution can be minimized by using a bit shuffling network (see Section 3.6.).

ΣΔ techniques in D/A conversion trade accuracy and complexity in the analog stage for increases in sampling speed. By using a sampling frequency much higher than the Nyquist frequency of the input signal, the requirements for the roll-off of the analog smoothing filter at the output of such a DAC are considerably reduced. ΣΔ DACs normally require more digital circuitry than Nyquist rate DACs, but the relative increase in active silicon area is minor (a 3 K gates ΣΔ DAC in a 100 K gates chip, for example). The next two sections will present quantization, oversampling and noise shaping, fundamental issues linked to ΣΔ conversion, with an accent on ΣΔ D/A conversion.

3.2. Quantization and oversampling

Figure 3.3. shows an uniform analog quantizer that rounds-off a continuous signal to odd integers in the range -5 to +5.

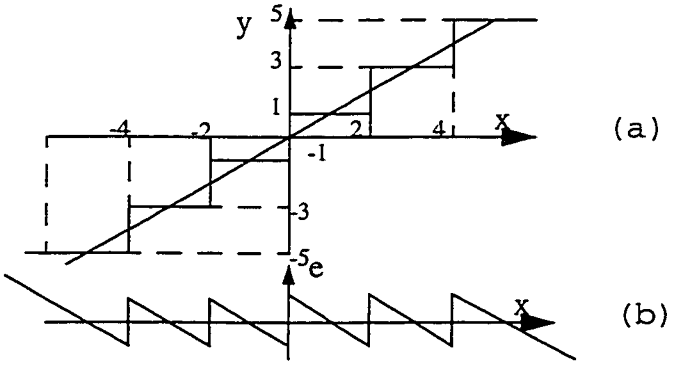


Fig.3.3. (a) Uniform analog quantization;
(b) quantization error

The quantizer level spacing is $\delta = 2$ and the output is

$$y = G \cdot x + e \tag{3.1}$$

where x is the quantizer input, G is the slope of the straight line connecting the centers in the quantization characteristic and e is the quantization error (see Figure 3.3.b.).

The k bit quantization of a binary number of length n is done by taking its k most significant bits. The input/output characteristic of a three bit quantizer with four bit two's complement number input is given in Figure 3.4.

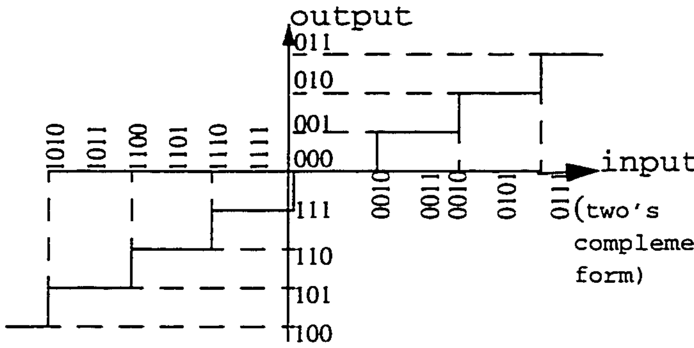


Fig.3.4. Digital quantization

An observation can be made regarding Figure 3.4. Tthe output of the digital quantizer is offset down by 1/2 LSB. The digital ΣΔ modulator used in the ΣΔ DAC proposed in this work uses such a quantizer, and therefore its output has a residual DC offset.

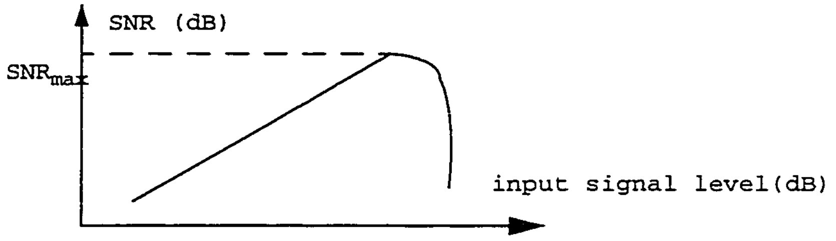


Fig.3.5. SNR versus signal level for a linear quantizer

Figure 3.5. shows the SNR at the output of a linear quantizer with sinusoidal input as a function of input signal level. At small input levels, the dependence is linear. At large input levels, when the quantizer is clipped (the output spends large amounts of time at the extremes of the quantizer's characteristic), the SNR has a fast roll-off. Any deviation from the ideal quantizer behavior (nonlinearity, step size impairment, dynamic error, harmonic distortion, physical noise, interfering signals, etc.) will impair such an SNR characteristic.

The Nyquist theorem states that the information in a signal can be completely recovered if the signal is sampled at least twice as fast as the signal bandwidth.

Oversampling is defined as deliberately sampling a signal at a frequency much higher than its Nyquist frequency. For a signal in the frequency band $0 < f < F_0$ sampled at a frequency $F_s = 1/T$, the oversampling ratio (OSR) can be defined as the ratio of the sampling frequency F_s to twice the signal bandwidth.:

$$OSR = \frac{F_s}{2 \cdot F_0} = \frac{1}{2 \cdot F_0 \cdot T} \qquad (3.2)$$

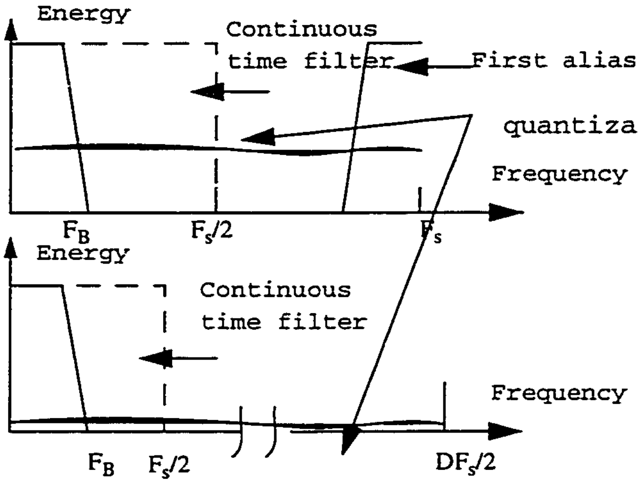
It can be demonstrated [Haus91] that the in-band quantization noise power is:

$$n_0^2 = e_{rms}^2 \cdot (2 \cdot F_0 \cdot T) = \frac{e_{rms}^2}{OSR} \qquad (3.3)$$

where e_{rms} is the RMS of the quantization noise. Oversampling reduces the in-band RMS quantization noise power n_0 by the square root of the OSR. Therefore, each doubling of the sampling frequency decreases the in-band quantization noise power by 3 dB. This is equivalent to increasing the resolution by half a bit (6 db of SNR corresponds roughly to one resolution bit). For consistency reasons, formula (2.5) and Figure 2.2. will be reproduced again, this time referring to a linear quantizer. Formula (3.4) gives the maximum theoretical SNR value for a linear N bit Nyquist rate quantizer

$$SNR_{max}(dB) = 6.02 \cdot N + 1.76 \qquad (3.4)$$

It is useful to discuss now some frequency domain aspects of oversampling.



(a) Sampling at Nyquist rate

(b) Oversampling

Fig.3.6. Effect of oversampling on the in-band quantization noise

The analog band of interest is $F_s/2$ in both cases in Figure 3.6. In case (a) quantization was done using a Nyquist rate quantizer. In case (b), the sampling frequency is DF, corresponding to an OSR of D. The same quantization error power as in the first case is spread in Figure 3.6.b. over a much wider range of frequencies. If one can separate with a continuous time (CT) filter the range of frequencies from 0 to $F_s/2$, an SNR improvement of D times is achieved, since the quantization noise power in the second case is D times lower in the band of interest because of oversampling. The maximum SNR is then given in [Haus91] as:

$$SNR_{max}(dB) = 6.02 \cdot N + 1.76 + 10 \cdot \log(D) \qquad (3.5)$$

It is convenient to write $D = 2^L$, where L is the number of octaves of oversampling. The SNR at the output of the quantizer is then given by the formula:

$$SNR_{max}(dB) = 6.02 \cdot (N + 0.5 \cdot L) + 1.76 \qquad (3.6)$$

Equation (3.6) shows directly that the oversampling quantizer yields SNR in the bandwidth of interest equivalent with that of a nonoversampling (Nyquist) quantizer with a higher number of bits. In this case, the gain in resolution is 0.5 bits per octave of oversampling.

3.3. Noise shaping

Improvements in resolution larger than half a bit per octave of oversampling may be achieved by arranging for the quantization noise power of the data converter to be spectrally shaped. The quantization noise power can be pushed out of the frequency band of interest, thereby increasing the in-band resolution of the data converter. A filter can then eliminate the out-of-band quantization error. The method, commonly referred to as noise shaping, is illustrated in Figure 3.7.

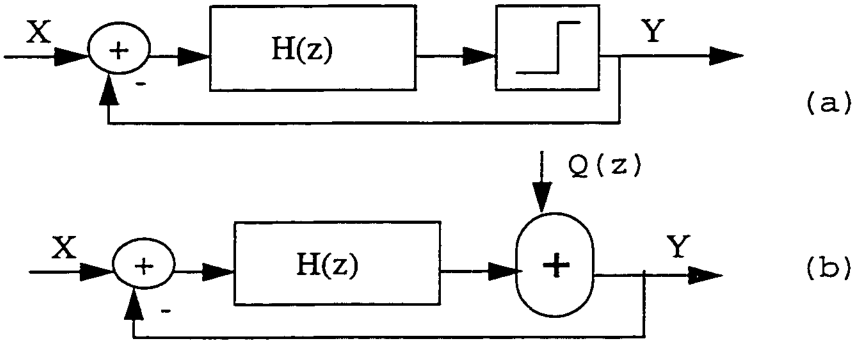


Fig.3.7. Noise shaping principle (a) z domain diagram
(b) linearized model

In Figure 3.7., H(z) is the z domain transfer function of a filter that is cascaded with a quantizer whose output Y is fed back and substracted from the input X. Figure 3.7.b. shows a linearized model of the loop. The quantization process can be modeled as an additive linear process whose inputs are the quantization noise, with z domain representation Q(z), and the output of the discrete time filter H(z). The signal transfer function (STF) and the noise transfer function (NTF) are given by formulae (3.7) and (3.8) respectively.

$$\frac{Y(z)}{X(z))} = \frac{H(z)}{1+H(z)} = STF \tag{3.7}$$

$$\frac{Y(z)}{Q(z)} = \frac{1}{1+H(z)} = NTF \tag{3.8}$$

By choosing H(z) appropriately and with the condition that the loop has to be stable (no value of z for which the denominator is zero, i.e. 1+H(z) = 0), it is possible to control the NTF. For example, if first order z domain differentiation is needed for the NTF, then

$$\frac{1}{1+H(z)} = 1-z^{-1} \Rightarrow H(z) = \frac{1}{z-1} \tag{3.9}$$

with $z = e^{j\omega T}$, where $\omega = 2\pi f$ is the signal pulsation and T the sampling period, and then

$$1-z^{-1} = 1-e^{-j \cdot \omega \cdot T} \approx j \cdot \omega \tag{3.10}$$

with absolute value increasing 20 dB per decade of frequency. If a second order differentiation is needed for the NTF, then

$$(1-z^{-1})^2 = \frac{1}{1+H(z)} \Rightarrow H(z) = \frac{2 \cdot z - 1}{(z-1)^2} \tag{3.11}$$

with absolute value increasing 40 dB per decade of frequency. Using formulae (3.9) and (3.11), respectively, two noise shaping ΣΔ modulator architectures can be built. They also use oversampling, and are called first order ΣΔ modulator and second order ΣΔ modulator respectively. These two architectures will be discussed next, with accent on the second order ΣΔ modulator, the type used in the ΣΔ DAC proposed in this work

3.4. First order ΣΔ modulator

The block diagram of a first order ΣΔ modulator is shown in Figure 3.8.a. This circuit is the simplest of all ΣΔ modulator architectures. The input into the circuit is fed to the quantizer via an integrator and the quantized output is fed back and substracted from the input. This feedback forces the average value of the quantized signal to track the average input value. Any difference between them accumulates in the integrator and eventually corrects itself.

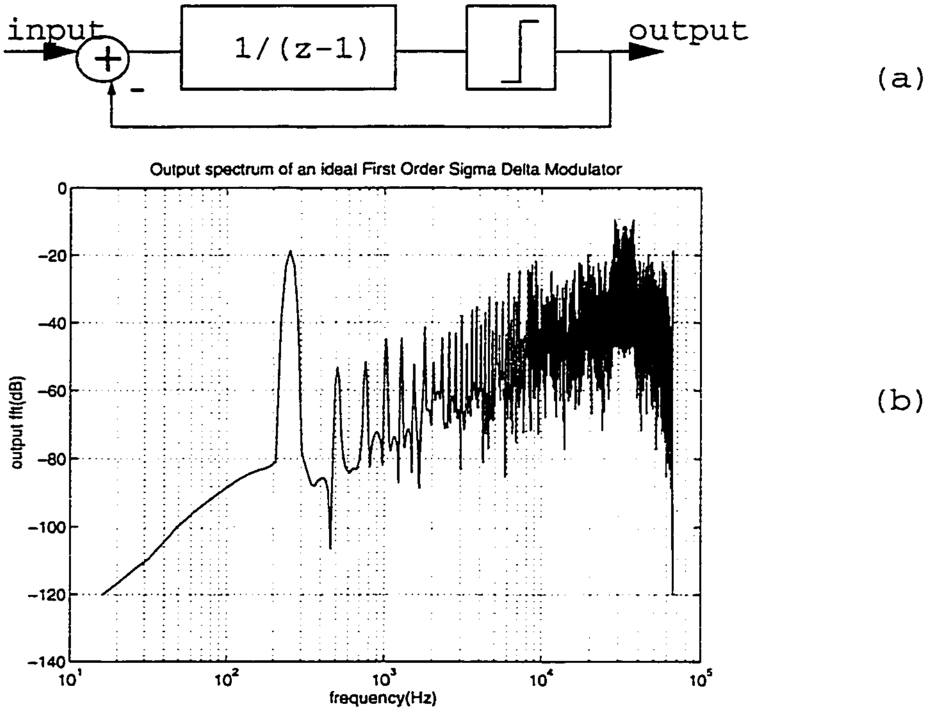


Fig.3.8. First order ΣΔ modulator
(a) z domain block diagram; (b) output spectrum for sinusoidal input

Figure 3.8.b. shows the output spectrum of a first order ΣΔ modulator using a one bit quantizer. The input is a sinusoidal signal with amplitude -18 dB (0 dB is the quantizer step). The 20 dB/decade NTF is evident when using a logarithmic frequency scale.

The circuit implementation of a first order digital ΣΔ modulator is shown in Figure 3.9. The input word, held in register R_0, feeds one port of a binary adder, the output of which separates in two paths. The more significant component, y_1,feeds to the output, while the less significant component, e_1, feeds back to the second port of the adder via register R_1. The LSBs accumulate until they overflow into the MSBs and thus contribute to the output. The incoming word rate is F_0, while the feedback register R_1 is clocked at a much higher frequency, $F_1 = KF_0$. The output of this circuit, expressed in the z domain, is:

$$Y_1(z) = X(z) + E_1(z) \cdot z^{-1} - E_1(z) = X(z) - \left((1 - z^{-1}) \cdot E_1(z)\right) \quad (3.12)$$

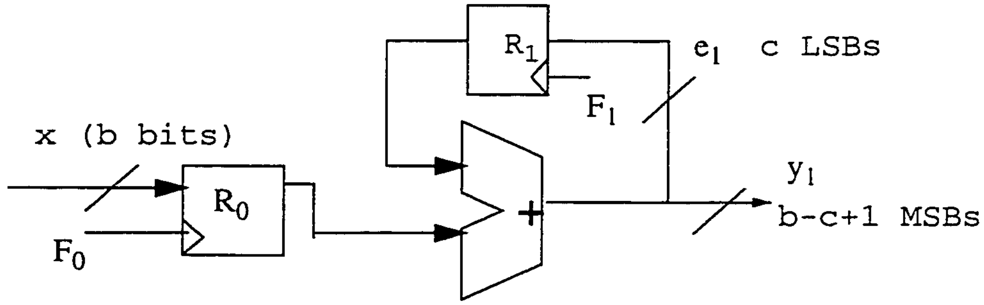


Fig.3.9. Circuit implementation of a first order digital ΣΔ modulator

where $z = e^{j\omega T}$, with $T=1/F_1$. From (3.12), the error E_1 is first order differentiated, which corresponds to 20 dB per decade in NTF, as demonstrated in Section 3.3. $Y_1(z)$ represents the input contaminated by a truncation noise $E_1(z)$ which is filtered by the high pass function $(1\text{-}z^{-1})$. Lets consider a binary representation of numbers in the digital ΣΔ modulator, with the incoming word width b and the error e_1 comprising the c least significant bits of the sum. Then y_l is (b-c+1) bits wide, the extra bit being the carry out bit from the top of the adder. Input codes can assume integer values from 0 to 2^b-1, the error integer values from 0 to 2^c-1 and the output assumes integer multiples of 2^c in the range 0 to 2^b. The number of levels needed to represent the output is:

$$l_1 = 2^{b-c} + 1 \tag{3.13}$$

but the switching between values must occur fast enough to suppress the truncation noise that enters in the signal band. A ΣΔ DAC using a first order digital ΣΔ modulator is shown in Figure 3.10; it was used in early Phillips CD players [Snel95]. The digital ΣΔ modulator is cascaded with an interpolation stage which increases the sampling frequency by a factor of four. The output of the digital ΣΔ modulator feeds a D/A reconstruction stage and the DAC output is taken from a CT smoothing filter. Using ΣΔ techniques required a cheaper D/A reconstruction stage (less analog output levels).

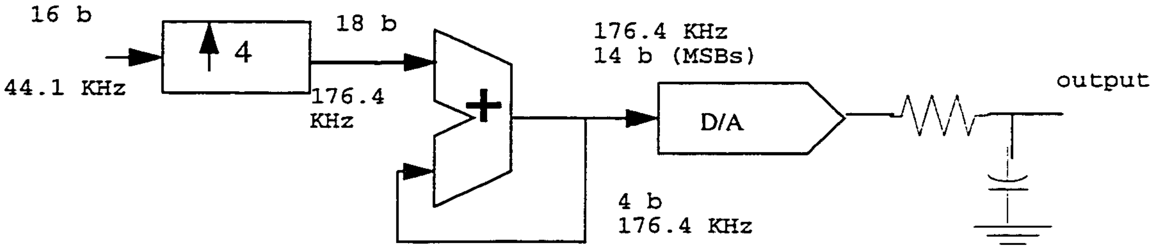


Fig.3.10. Phillips ΣΔ DAC using a first order digital ΣΔ modulator

At this stage it is useful to briefly discuss the interpolator stage, a necessary part of any ΣΔ DAC. Its function is to deliver the incoming digital information at the sampling rate of the digital ΣΔ modulator. The circuit proposed in the present work does not include such a stage.

Interpolation is the process of digitally converting the sampling rate of a signal from a given rate F=1/T to a higher rate $F_I = 1/T_I$. The process of interpolating a signal x(n) by an integer ratio L is depicted in Figure 3.11. Figure 3.12. illustrates the interpolation process in the time and frequency domains, respectively.

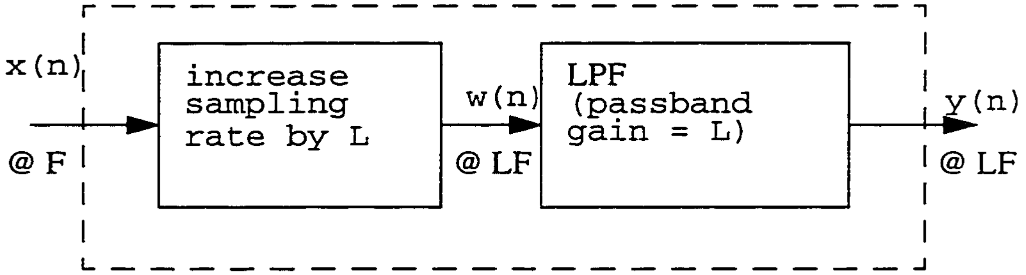


Fig.3.11. Interpolator stage

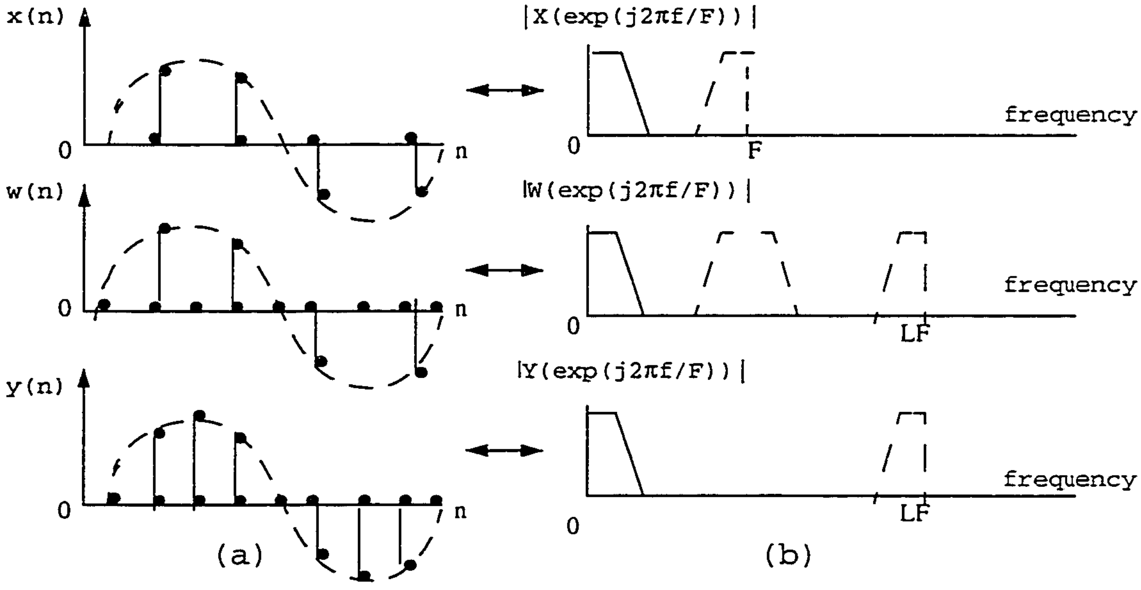


Fig.3.12. Interpolation process
(a) time domain; (b) frequency domain

The sampling rate of input x(n) is increased by a factor of L (in this case L = 2) by inserting L-1 zero-valued samples between each two samples of x(n). This creates a signal w(n) whose frequency components are periodic with period equal to the original sampling frequency F. To eliminate these periodic components and retain only the baseband frequencies, it is necessary to filter the signal w(n) with an appropriate low pass filter. The resulting signal y(n), with sampling rate LF, is then the desired interpolated signal. In order to ensure that the amplitude of the interpolated signal is the correct one, the gain of the filter has to be L. A detailed mathematical treatment of the interpolation process can be found in [Croc80]. [Hoge81] introduces an elegant interpolator circuit implementation.

Returning to the first order ΣΔ digital modulator subject, it is to be noted that it has a series of drawbacks which make it unsuitable for industrial applications. Nevertheless, its behavior is well understood at the mathematical level [Snel95] and that knowledge base helps understanding the more complicated (and not always guaranteed to be stable) higher order ΣΔ modulator architectures. Stability in ΣΔ sense refers to the dynamic behavior of a ΣΔ modulator whose output does not change for long periods of time, compared to the sampling period. Alternatively, a ΣΔ modulator is stable as long as its output SNR does not drop dramatically.

The most important drawback of a first order ΣΔ modulator is its limit cycle behavior. Figure 3.13. shows the output spectrum of a one bit output first order ΣΔ modulator with a DC input of 0.05 (the 1 bit quantizer step is the reference, $\delta = 1$). The in-band frequency components are clearly visible and they depend on the DC input value.

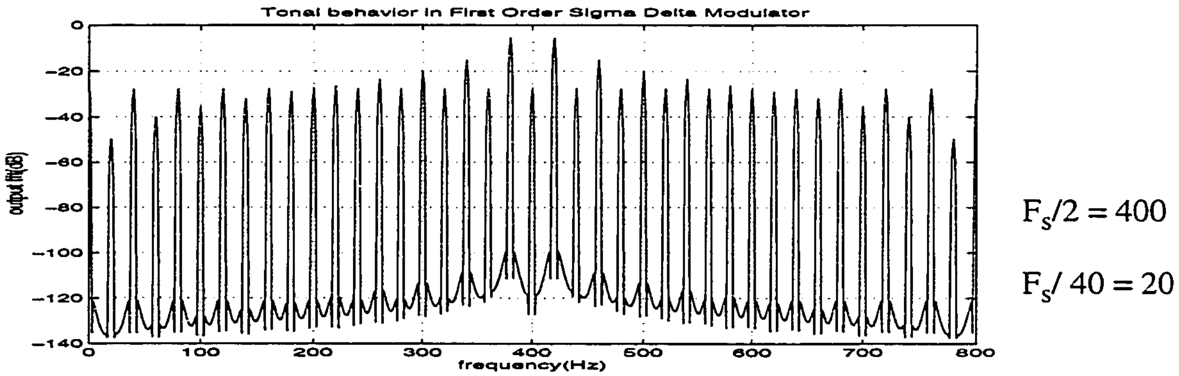


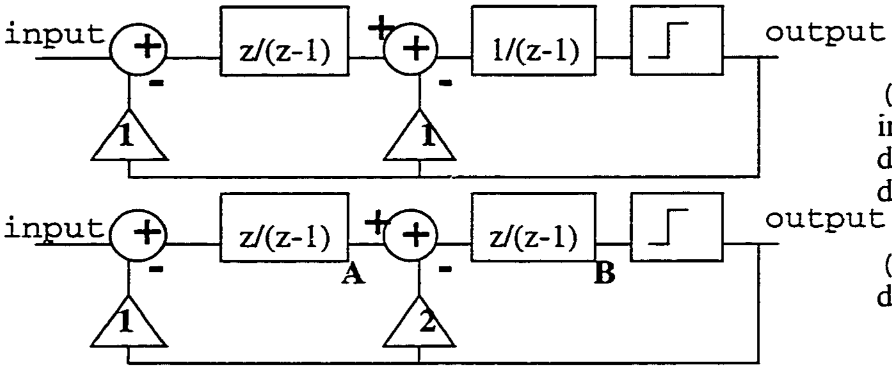
Fig.3.13. Limit cycle behavior in first order ΣΔ modulator; output spectrum for 0.05 DC input; limit cycle at Fs/40.

The limit cycle behavior of a first order ΣΔ modulator can also be seen in Figure 3.8.b. for a sinusoidal input. The in-band resolution of a ΣΔ DAC built with such a digital ΣΔ modulator will be severely affected. [Snel95], [Cand86], [Haus91] give more detailed information about the first order ΣΔ modulator drawbacks.

One solution to the limit cycle problem in ΣΔ modulators is the use of dithering. Dithering is the process of disrupting the long deterministic idling patterns at the output of ΣΔ modulators by adding some AC signal at different points of the circuit. In reference [Haus81], Hauser mentions dithering methods by adding AC signal at the loop input for ΣΔ A/D converters. Adding a square wave signal at powers of two of ΣΔ A/D converter sampling rate is one of the above methods. Dither can also be added at the quantizer input or, equivalently, randomly translating the quantizer transfer function from midthread to midrise. An implicit dithering method similar to this one is employed in the digital ΣΔ modulator proposed in the present work. It should be noted that there is a shortage of information in the literature regarding dithering in digital ΣΔ modulators

3.5. Second order ΣΔ modulator

The second order ΣΔ modulator, while more complicated to understand or describe mathematically, is proven to be stable, has higher in-band resolution for the same OSR and less limit cycle problems compared to the first order ΣΔ modulator architecture. These advantages make it suitable for practical implementations in telecommunications. The z domain transfer function of the filter H(z) used to implement a second order ΣΔ modulator architecture was given in equation (3.11). There are two corresponding z domain block diagrams, as shown in Figure 3.14.



(a) cascade of two integrators: one with delay, one without delay

(b) cascade of two delaying integrators

Fig.13.14.Second order ΣΔ modulator architectures

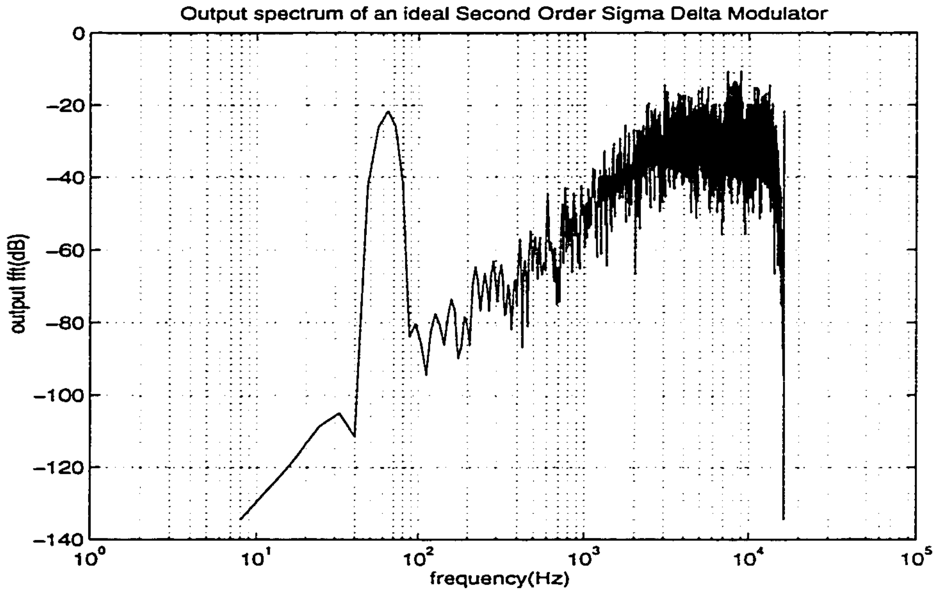


Fig.3.15. Output spectrum of an ideal second order ΣΔ modulator with sinusoidal input

The 40 dB/decade NTF of a second order ΣΔ modulator is clearly evident on a logarithmic frequency scale, as well as its relative lack of in-band harmonics when compared with the spectrum of the first order ΣΔ modulator architecture shown in Figure 3.8.b. An empirical formula for peak SNR in a bandwidth BW for a given OSR = Fs/(2*BW) is [Cand92]:

$$SNR[dB] = 15 \cdot \log_2(OSR) - 23 \qquad (3.17)$$

The resolution in bits is given by:

$$Resolution[bits] = 2.5 \cdot \log_2(OSR) - 4 \tag{3.18}$$

One problem with digital implementations of the second order ΣΔ modulator architectures shown in Figure 3.14. is that the maximum signal level at the second integrator's output can reach very high values as the DC code input level in the modulator closes to the rails (for a one bit quantizer, the rails are the quantizer levels)

.

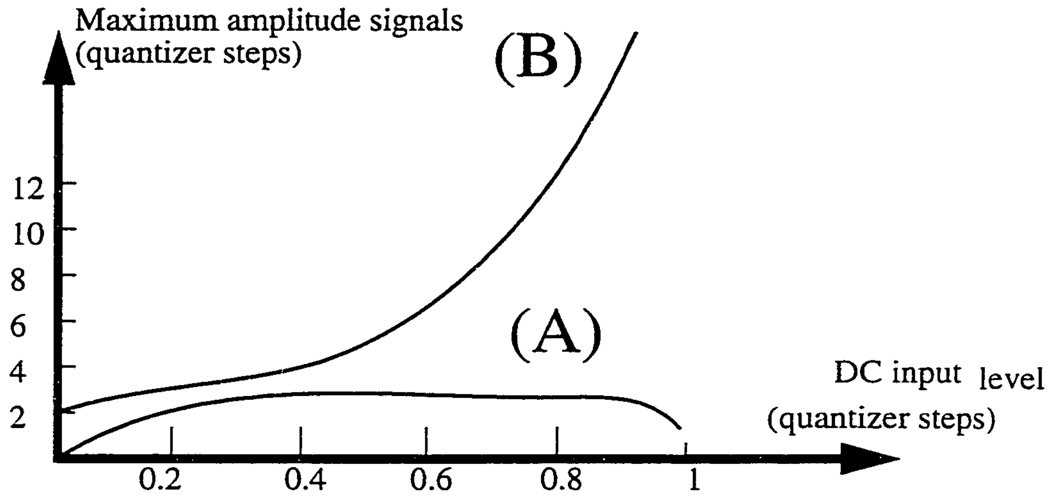


Fig.3.16. Maximum integrator output levels for a one bit second order digital ΣΔ modulator as a function of DC code input level;

The DC sweep is a standard test for ΣΔ modulators [Cand92]. AC sweep test results (in-band SNR, behavior of maximum amplitudes at various points in the circuit) depend on the DC bias [Cand86].The problem of two's complement numbers wrap-around is well known. For example, when incrementing the positive number 01111111 by one LSB, the result is the negative number 10000000 , which is at the other end of the representable range. (see Figure 3.17.c.). In a digital ΣΔ modulator, the signal level at the output of the second integrator can become larger than the representable range of two's complement numbers at that point, thereby causing arithmetic overflow.

As a result, the quantized value will have a wrong value. This will create the conditions for low frequency limit cycles in the loop, which in turn will negatively affect the in-band resolution of a ΣΔ DAC using such a digital ΣΔ modulator. For example, let us consider a second order ΣΔ modulator, one bit output implementation for the circuit shown in Figure 13.14.b. The corresponding circuit (with a clipper added to prevent arithmetic overflow) is shown in Figure 3.17.a. below. The maximum signal level at point (A), the output of the first integrator, reaches a maximum of 2δ (δ is the quantizer step), as implied from Figure 3.16. There are two ways to avoid two's complement wrap-around at the output of the second integrator (see Figure 3.17.b.). One approach is to use sign bit extension at the input of the ΣΔ modulator in such a way as the representable two's complement signal range at the second integrator output to cover all possible signal values at that point. The sign bit extension can also be used locally, as seen for the circuit shown in Figure 3.18, for the input in the second integrator (a one bit sign extension).

Sign bit extension is a widely used circuit technique in digital hardware. For example, the number 10111 can also be expressed as 1110111 by replicating twice the value of the sign bit of the first representation. For the circuit discussed at the moment, an input sign bit extension of three will result in an increase in the representable range of eight times throughout circuit. As seen in the graph in Figure 3.16., increased wordlength alone is not enough to guard against two's complement wrap-around at the second accumulator output, as the DC input code approaches the rails. The maximum value of the signal at the output of the second integrator will require too wide a word (for example 5 or 6 bits extension), which will slow down the second accumulator (part of the critical path in the circuit), and therefore the whole ΣΔ modulator. The maximum achievable resolution in a certain bandwidth will therefore be affected. A second method, used in conjunction with the first, is to keep the signal level at point (B) inside a certain region by using a clipper, as seen in Figure 3.17.a.

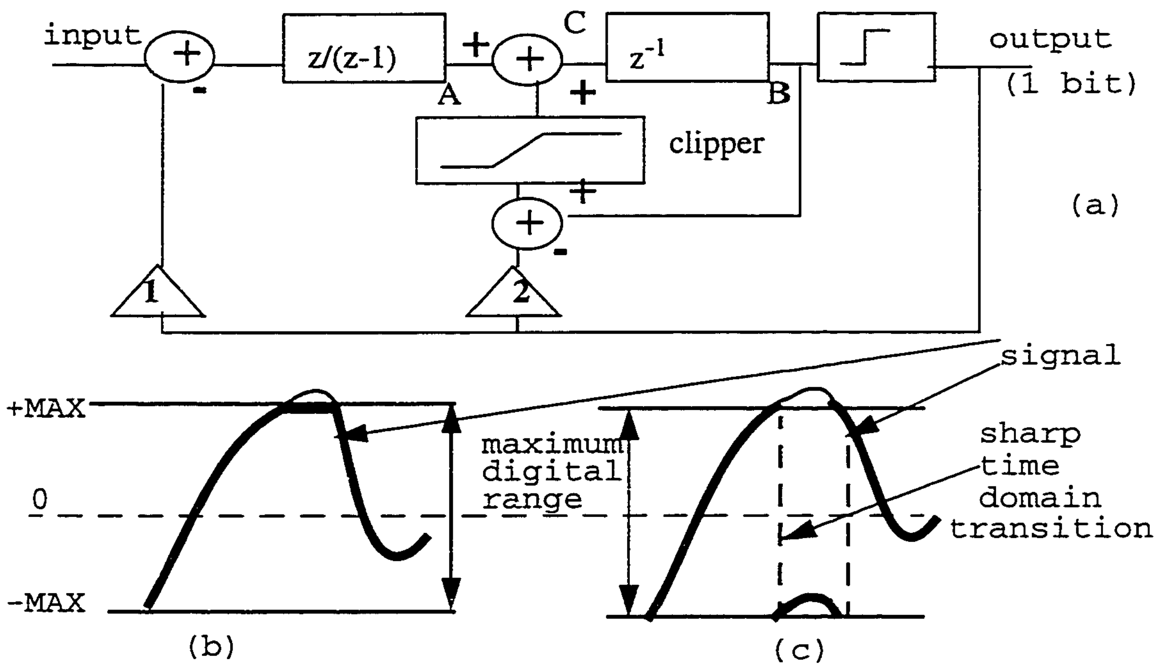


Fig. 3.17. (a) Second order digital ΣΔ modulator using a clipper (b) saturation arithmetic (c) two's complement wrap-around

If the maximum signal level at point (A) is 2δ (i.e. two quantizer steps for a one bit output modulator) and the clipper limits the feedback signal to a maximum of 4δ, for example, the maximum signal level at point (C) (and at the quantizer input, which is just one sampling period delayed) is 6δ. An input sign bit extension of three bits corresponds to increasing the representable range in every point of the circuit by a factor of $2^3 = 8$ and thus will accommodate the resulting maximum signal level at point (B).

The digital clipper can be implemented using saturation arithmetic, which emulates the "full scale" or "pegged" behavior of analog circuits. Figure 3.17.b. shows the behavior of a circuit using saturation arithmetic, as opposed to one allowing for two's complement wrap-around (Figure 3.17.c.). When the signal level is more than the digital representable range, the output is represented by the maximum representable value. No big jumps are allowed in the waveform, so the output spectrum is not as rich in harmonics as the one generated by the sharp time domain transitions in the wrap-around case.

A practical second order digital ΣΔ modulator will be discussed next. This implementation is the model against which the second order carry save ΣΔ modulator architecture proposed in this work is to be compared in terms of achievable speed. The circuit was proposed by Su *et al.* in [Su93]. The digital ΣΔ modulator is part of a CMOS oversampling ΣΔ DAC and has the architecture shown in Figure 3.17.a. The corresponding circuit level implementation is shown in Figure 3.18. Because two's complement arithmetic is used, the one bit quantizer is implemented by taking the sign bit of the second integrator output. The input is 16 bits wide and sign bit extension in the first stage is two bits and in the second stage is three bits. The positive and negative full scale levels at the output of the quantizer are chosen to be 08000_{HEX} and 18000_{HEX}. In both cases, if the lower 15 bits are zero, only the three MSBs in the two subtractors need to be implemented. In binary form, the quantizer outputs are respectively:

$$HIGH = 01000000000000000$$

and

$$LOW = 11000000000000000$$

Then inverting the quantizer's output is done using a simple one bit inverter, as shown in the circuit implementation in Figure 3.18., whose z domain block schematic was given in back in Figure 3.17.a.

The same technique will be used in the digital ΣΔ modulator proposed in this thesis. Two three bit adders, each with one input complemented, implement the subtraction operations. The gain of two is wired as a left shift by one bit at the input of the second subtractor. To demonstrate this technique, consider the two's complement number 1110100110, with a two bit sign extension. The wired multiplication by two is implemented by shifting left one position all bits and assigning 0 to the LSB of the new number: (1101001100). The clipper is implemented using combinational logic. A block diagram for the clipper and its transfer function are shown in Figure 3.19.

The accumulators are implemented with ripple carry adders (slow, but not power intensive) and the registers are using master slave logic. The one bit second order digital ΣΔ modulator is part of a complete oversampling DAC which also includes: an interpolator stage at the input, and a current mode semidigital reconstruction filter cascaded with a continuous time filter at the output

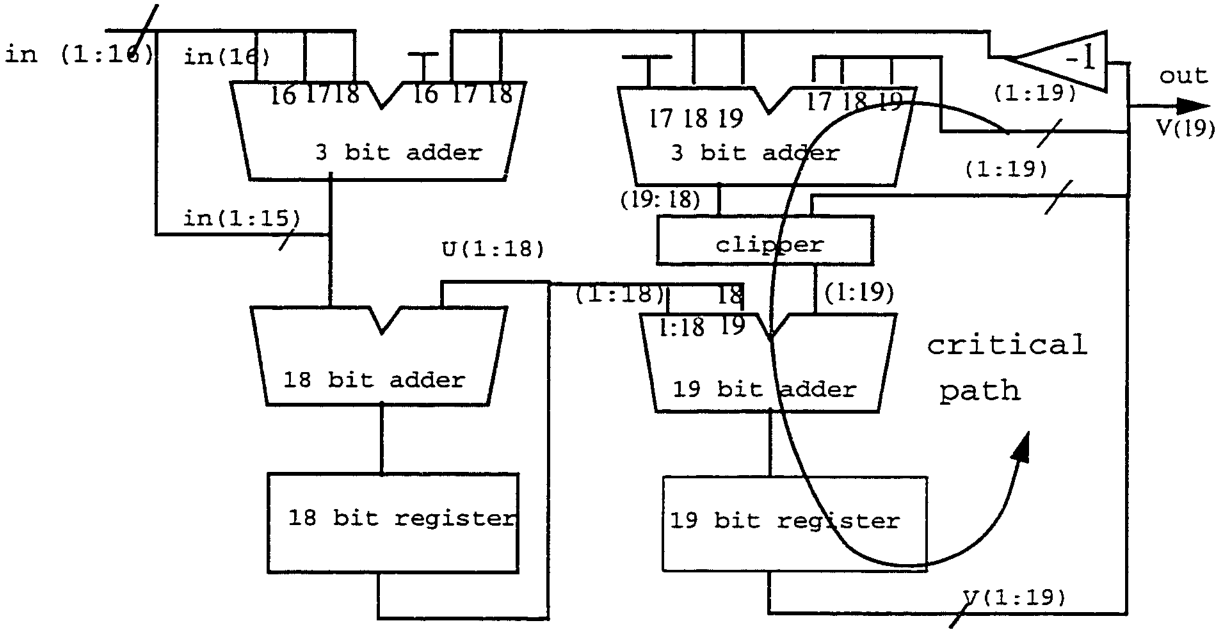


Fig.3.18. State of the art second order digital ΣΔ modulator [Su93]

The circuit reported by Su *et al.* takes 3 mm^2 of active die area in a 1.2μ CMOS implementation. The maximum sampling frequency is 20 MHz. For an OSR of 176, the dynamic range of the ΣΔ modulator is 94 dB, the peak SNDR in 20 KHz bandwidth is 88 dB and the power dissipation at 5V supply is 59 mW [Su93].

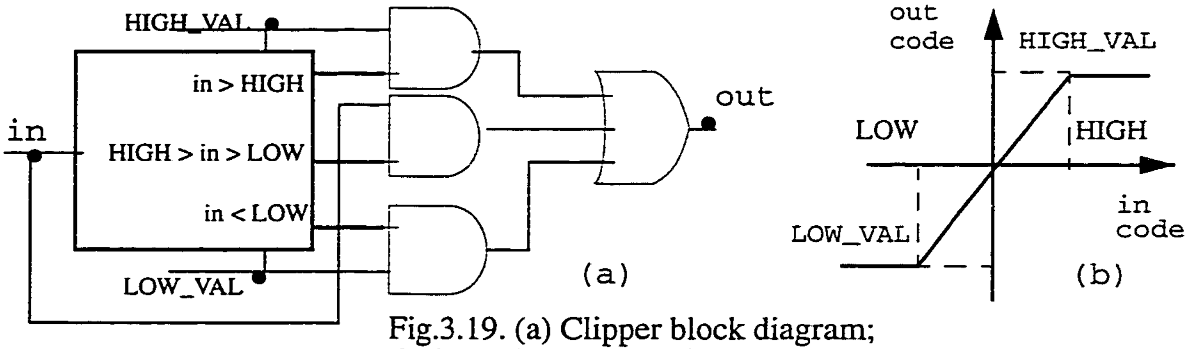


Fig.3.19. (a) Clipper block diagram;
(b) input /output transfer function.

3.6. Bit shuffling network

The greatest strength of the one bit quantizers used in ΣΔ modulators is their inherent linearity. Since there are only two levels of decision, the quantizer gain can have any value. One bit ΣΔ modulators require high OSR to achieve a certain resolution, which puts heavy demands on circuit operating speed and power dissipation. In addition, one bit ΣΔ modulators suffer from idle tone problems that require dithering (see Section 3.4.)

On the other hand, ΣΔ modulators using multibit quantizers exhibit more robust stability characteristics, with wider signal range, better immunity to in-band limit cycle problems, lower out-of- band noise and higher achievable resolution for the same OSR, compared to the single bit architectures. Figure 3.20. qualitatively illustrates the in-band quantization noise power versus input signal range for a typical one bit ΣΔ modulator versus an n bit ΣΔ modulator.

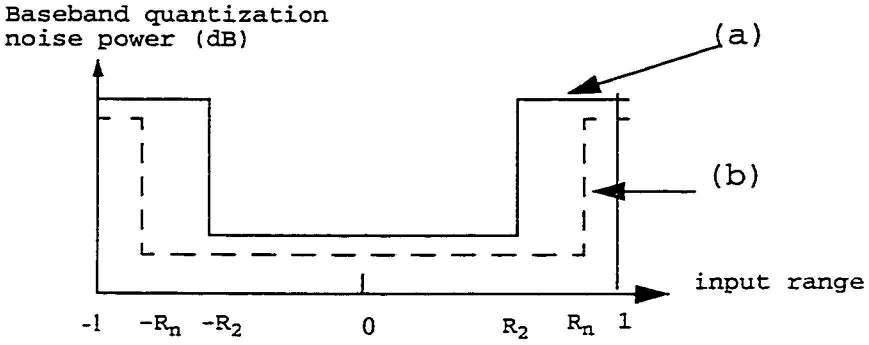


Fig.3.20. In-band quantization noise power versus input signal range for a ΣΔ modulator with (a) one bit quantizer; (b) n-level quantizer.

The input level is normalized to the one bit quantizer treshold. R_2 (<1) is the maximum normalized input signal level for which a stable ΣΔ modulation (indicated by a good SNR) is still obtainable. The sudden increase in in-band quantization noise for signal levels beyond R_2 indicates that the one bit quantizer is clipped and the ΣΔ modulator is saturated. When a uniform n-level quantizer using the same normalized levels -1 and 1 is inserted in place of the one bit quantizer, the quantization step size is reduced from 2 to 2/(n-1). The baseband quantization power is reduced accordingly by $6\log_2(n-1)$ dB, or an increase in resolution of the modulator by 6 dB for each supplementary bit of the quantizer [Peic95]. For example, a three bit output digital ΣΔ modulator can provide 12 dB more SNR than a one bit output digital ΣΔ modulator measured under the same conditions (same OSR and sampling frequency). This important result will be used later in the thesis.

Let us visit again the problem of a DAC using thermometer code (Figure 3.21). Chapter two presented the sources of error in current steering arrays. Figure 3.22. shows the ideal and real transfer functions, respectively, of such a D/A reconstruction stage.

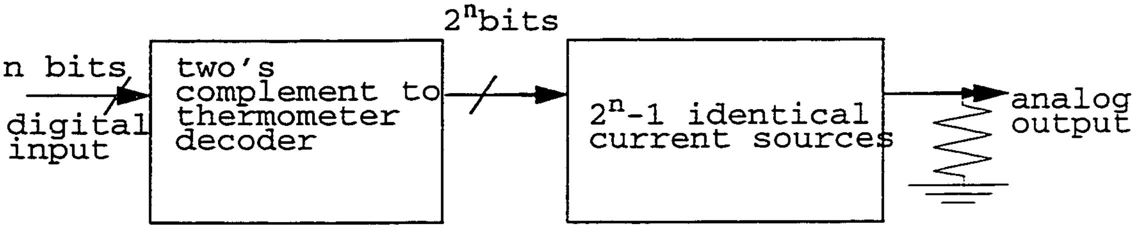

n bits
digital input
two's complement to thermometer decoder
2^n bits
2^n-1 identical current sources
analog output

Fig.3.21. DAC using thermometer code and current steering array D/A stage

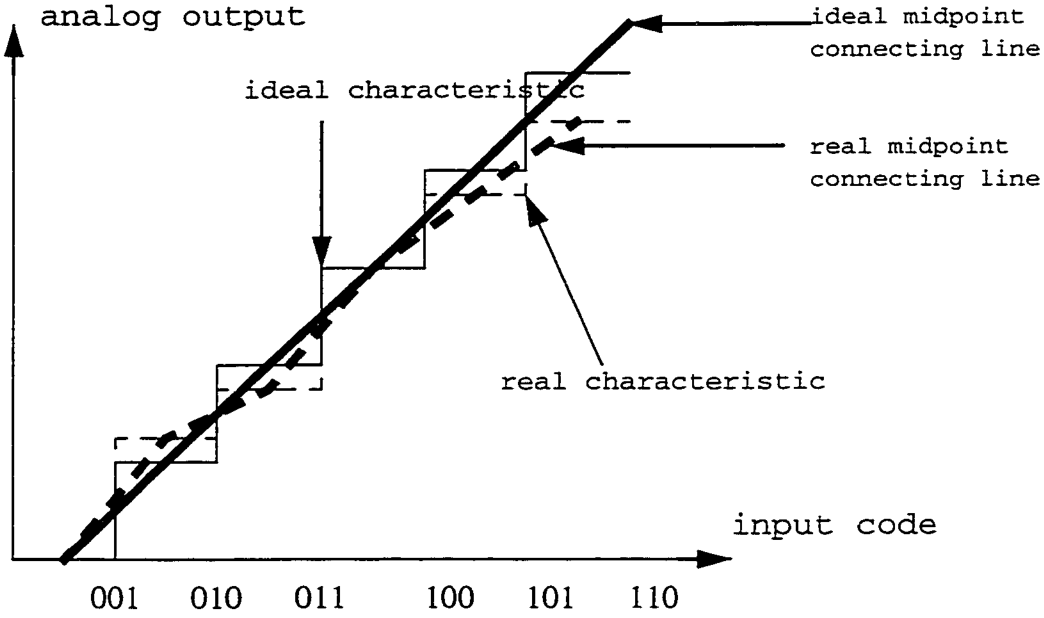


Fig.3.22. Ideal and real DAC input/output transfer function

When using a real segmented current steering array as the D/A reconstruction stage of a ΣΔ DAC (Figure 3.23), the in-band resolution is affected by the current source mismatch, as shown in Figure 3.24.

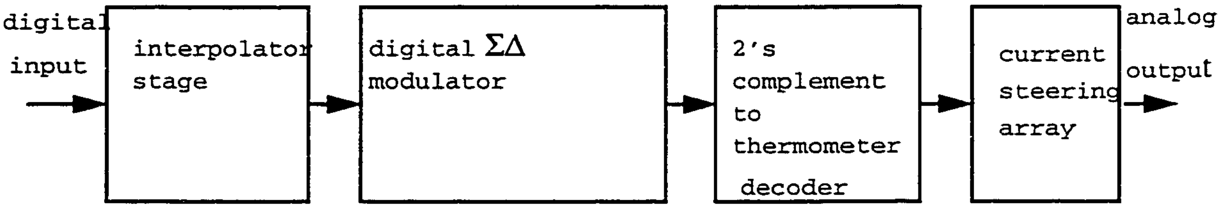


Fig.3.23. ΣΔ DAC architecture

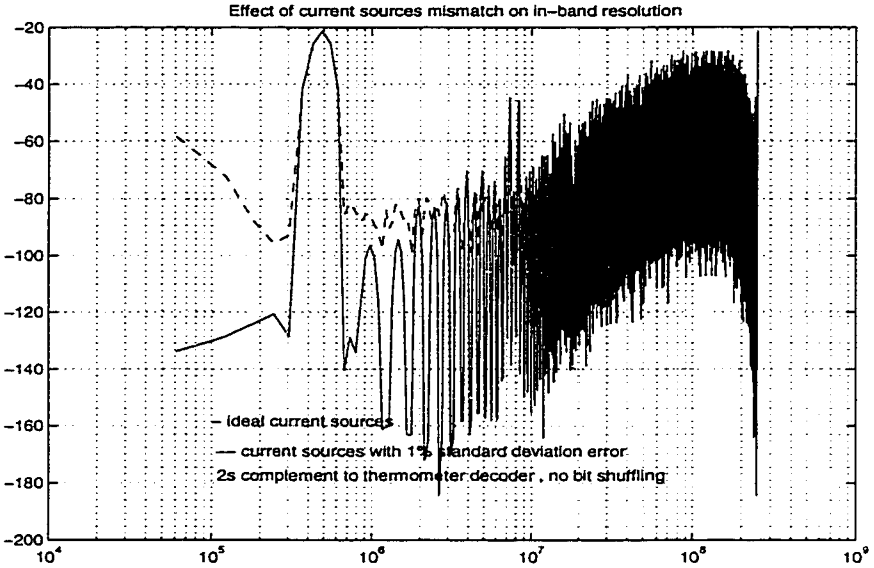


Fig.3.24. Effect of current sources mismatch on the output spectrum of a ΣΔ DAC (no bit shuffling)

To avoid the severe in-band resolution loss due to current source mismatch, one can use a technique called dynamic element matching (DEM). One type of DEM is called clocked averaging DEM, proposed in [Leun92]. Let us onsider the 2 bit, 3 level output DAC shown in Figure 3.25.a., with 2 unit elements. The two unit elements (capacitors or current sources) have ρ and -ρ normalized errors ($0 < \rho << 1$). Evidently, the output corresponding to the input code 2 (binary 10) will be the correct one. On the other hand, level 1 (binary

01) will be implemented differently, depending on which unit element is employed. The DAC can possibly have two input/output transfer characteristics, as shown in Figure 3.25.b. By alternately choosing each unit D/A element to implement the level 1, the average DAC output reaches the correct value after two clock periods (the output is updated every clock period, T). This process is illustrated in Figure 3.26.

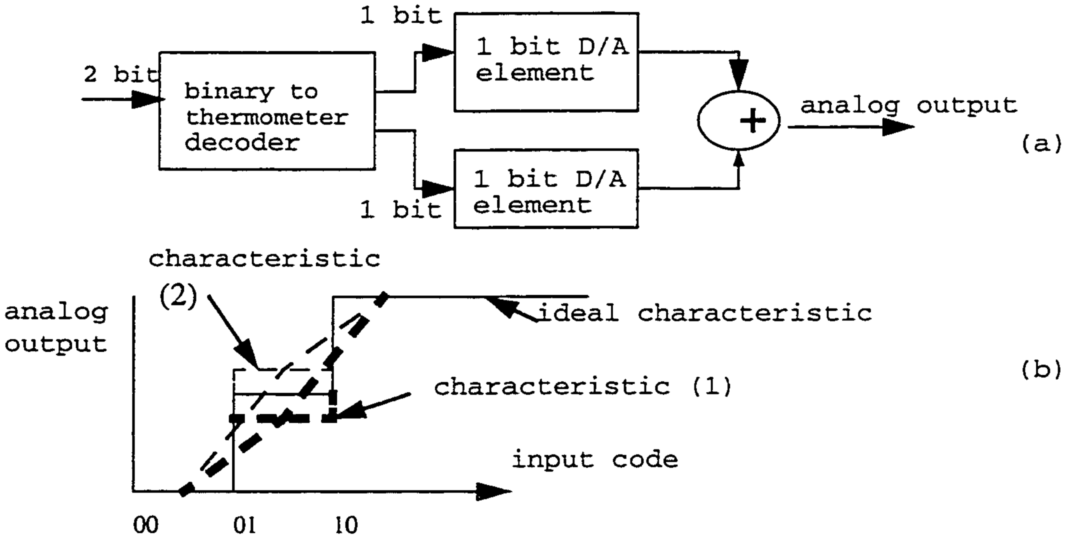


Fig.3.25(a) 2 bit, 3 level DAC;
(b) two possible input /output characteristics

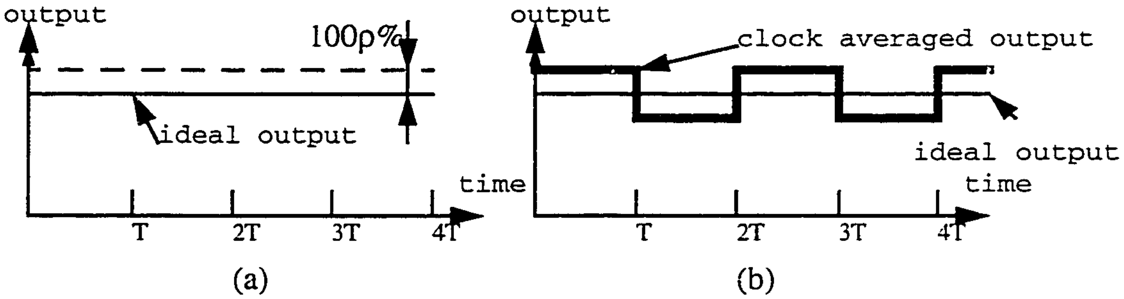


Fig.3.26. Effect in time of the clock averaged DEM on DAC output
(a) output of DAC which does not use the method
(b) output of DAC using the clock averaged DEM

For an n-element DAC with mismatches, the structure implementing DEM is shown in Figure 3.27. The circuit includes a parallel unit-element DAC structure and a digital randomizer which controls the connections between the input and the parallel DAC. The circuit has been proposed by Carley in [Carl89]. The randomizer connects the 2^n outputs from the decoder to the 2^n switching elements (eg. current sources) in a time varying fashion.

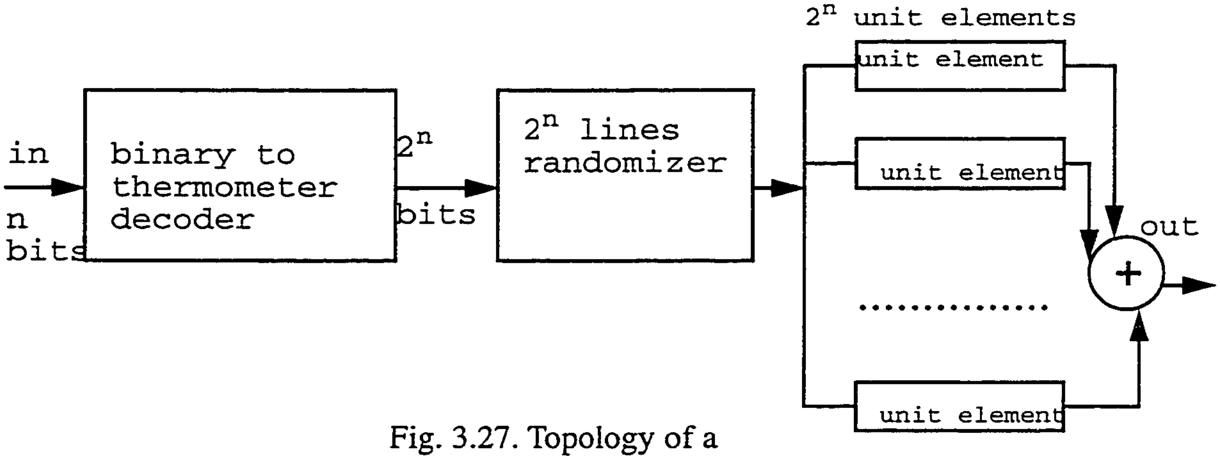


Fig. 3.27. Topology of a DAC using DEM

The number of possible connections is $2^n!$. When DAC input word width n is small, it is possible to randomly select between all possible combinations. When n is large (e.g. larger than 8) the large number of possible connections makes it necessary to select only a subset of them in order to conserve die area (eg. for an 8 bit randomizer, the number of possible connections is 40320).

The implementation of a clock averaged DEM randomizer needs rather complex digital circuitry. In contrast, the bit shuffling network used in this work uses a ΣΔ-based method, with an elegant and simple hardware implementation.

3.7. Carry save arithmetic

The ΣΔ modulator speed-up technique proposed in this work is based on the use of carry save arithmetic. This section is dedicated to introducing the reader to basic notions of this type of arithmetic.

Lets take a look at the way the summation of two one bit numbers is done. The sum of two one bit numbers is a two bit number: the most significant bit is the carry, and the least significant bit is the logic sum. The summation process of two one bit numbers, a and b, can then be considered an encoding process:

$$a + b = (co, so) \tag{3.19}$$

where the carry bit is computed as the logical and of the two one bit numbers:

$$co = a \cdot b \tag{3.20}$$

and the sum bit is computed as their logical exclusive or:

$$so = a \otimes b \tag{3.21}$$

The sum of three one bit numbers a, b and c, can also be considered an encoding process and the equations for the carry and sum bits are given below. The corresponding circuit is a full adder (FA), as shown in Figure 3.28.

$$co = (a \cdot b) + (b \cdot c) + (c \cdot a) \tag{3.22}$$

$$so = a \otimes b \otimes c \tag{3.23}$$

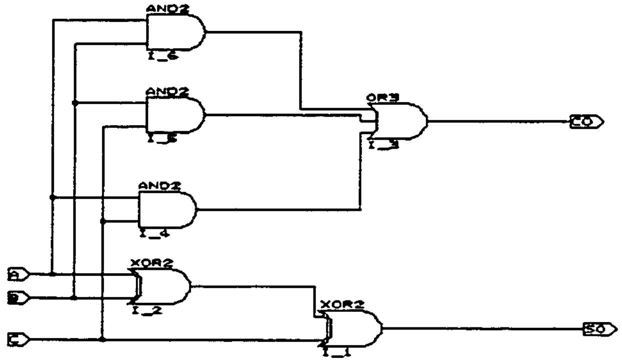


Fig.3.28. Full adder circuit

For two or three n-bit two's complement numbers, carry save summation can be done by coding the individual bits of the numbers using the method described above. The result is a n-bit carry number and a n-bit sum number. The real sum can be retrieved by adding the sum number with the carry number shifted left one position. For example:

$$\begin{array}{lllllllllllllll}
A = 1 & 0 & 1 & 1 & 0 & 1 & \quad & A+B = & 1 & 0 & 1 & 1 & 0 & 1 & + \\
B = 1 & 0 & 0 & 1 & 1 & 0 & & & 1 & 0 & 0 & 1 & 1 & 0 & \\
\cline{9-14}
 & & & & & & & & s_5 & s_4 & s_3 & s_2 & s_1 & s_0 & \\
 & & & & & & & & c_5 & c_4 & c_3 & c_2 & c_1 & c_0 &
\end{array}$$

$$s_5\ s_4\ s_3\ s_2\ s_1\ s_0 \quad = 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1$$

$$c_5\ c_4\ c_3\ c_2\ c_1\ c_0 \quad = 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0$$

The sum number is 001011 and the carry number is 100100. The real sum can then be computed directly:

$$\begin{array}{rl} A + B = & 0\ 0\ 0\ 1\ 0\ 1\ 1\ + \\ & 1\ 0\ 0\ 1\ 0\ 1\ \boxed{0} \longleftarrow \text{shift left one position} \\ \hline & 1\ 0\ 1\ 0\ 1\ 0\ 1 \end{array}$$

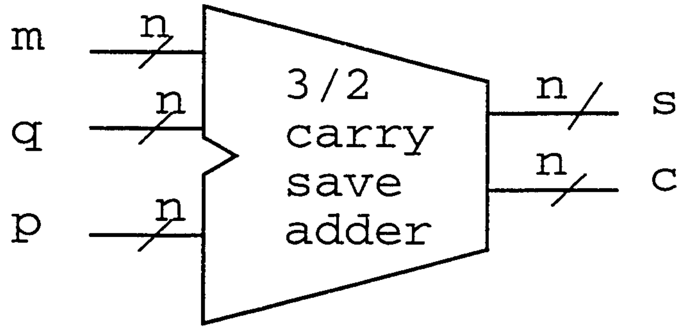
Similarly, for the summation of three n-bit two's complement numbers, same order bits can be summed independently using full adders (FA). The resulting n-bit carry and sum numbers can be used to retrieve the real sum exactly as before: the sum number has to be added to the carry number shifted left one position (see Figure 3.29.a.).

$$\begin{array}{ll} m_{n-1}\ m_{n-2} \cdots\cdots m_1\ m_0\ + & \\ & 0\ \ s_{n-1}\ s_{n-2} \cdots\cdots s_1\ s_0\ + \\ q_{n-1}\ q_{n-2} \cdots\cdots q_1\ q_0\ + \ \Leftrightarrow & \\ & c_{n-1} c_{n-2} \cdots\cdots c_1\ c_0 \quad 0 \\ p_{n-1}\ p_{n-2} \cdots\cdots p_1\ p_0 & \end{array}$$

c_i = (m_i and q_i) or (q_i and p_i) or (m_i and p_i)

s_i = $\underline{m_i \text{ xor}}$ q_i xor p_i

i = 0: n-1 (a)



(b)

Fig.3.29 (a) carry save summation of three n-bit binary numbers (b) 3/2 carry save adder symbol

Since the sum of three n-bit numbers is represented by two n-bit numbers (a carry and a sum) the process can be called 3/2 carry save (3/2 CS) summation and the circuit implementing it a 3/2 carry cave adder (Figure 3.29.). The circuit used to do the 3/2 CS summation is the full adder, shown in Figure 3.28. implemented in a technology which allows maximum three-input gates.

It is evident that accumulators implemented with this kind of adders are very fast: the critical path of the FA in Figure 3.28. is only two gates. The main problems in adapting the carry save arithmetic for use in ΣΔ digital modulators are the quantizer and clipper implementations, which are explained in the next chapter. In brief, the problem is this: quantization and clipping are nonlinear operations that would force us to convert from the carry save form of representation to a nonredundant format, and this conversion requires us to use slower conventional adders.

3.8.Summary

This chapter presented a high precision D/A method, called sigma delta, which increases the D/A conversion resolution in the band of interest. A particular implementation of a second order ΣΔ modulator was discussed, highlighting the hardware implementation difficult points. The price to be paid for the benefits of using multibit ΣΔ modulators (increased stability, higher resolution) is the necessity of using bit shuffling. The last section of the chapter introduced the reader to carry save arithmetic: very fast, but not easy to adapt for use in ΣΔ D/As.

CHAPTER 4 New ΣΔ modulator architecture

4.1. 7/3 carry save summation

The research work described in this thesis focused mainly on finding a way to speed-up the operation of digital ΣΔ modulators, essential building blocks of ΣΔ DACs. Figure 4.1. shows a z domain block diagram of a second order ΣΔ modulator. As explained in Chapter 3, there are two possible architectures which implement the second order differentiation of quantization noise. The architecture shown below has two delaying integrators.

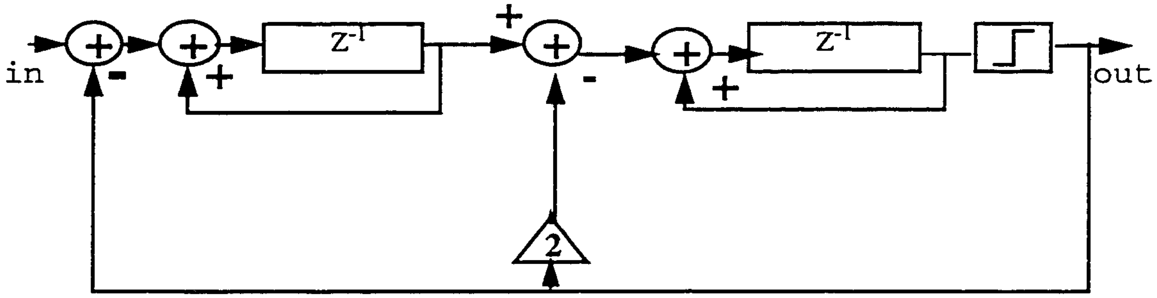


Fig.4.1. Second order ΣΔ modulator
z domain diagram

This architecture was chosen to demonstrate the CS technique because of its demonstrated stability. By using a stable ΣΔ modulator architecture, the risk of producing non-working silicon had been reduced (there was a risk that the method could not be stable in the long run). In a digital ΣΔ modulator, the integrators are the elements which take most of the computation time. A digital integrator (accumulator) is built using an adder and a register (see Figure 4.2.).

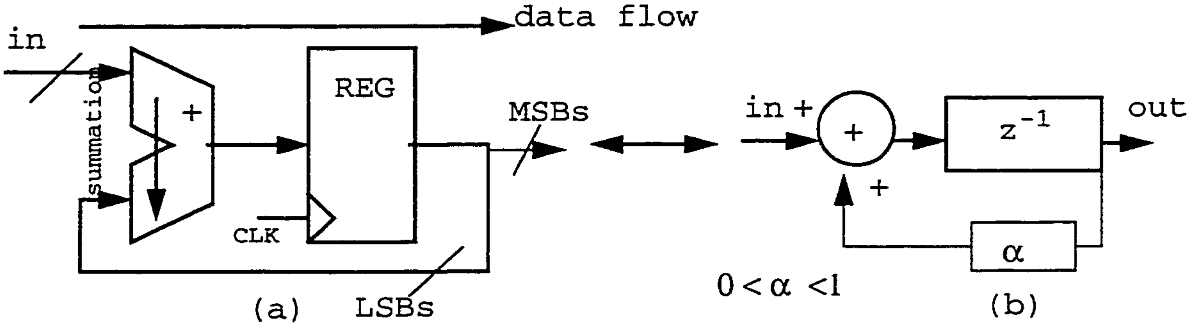


Fig.4.2. (a) Digital accumulator; (b) z domain equivalent

The summation takes most of the computation time in an accumulator. Very high speed for a digital ΣΔ modulator means high achievable OSR, which translates into high in-band resolution for a given frequency bandwidth. Equivalently, a much higher sampling frequency for a digital ΣΔ modulator translates into an extended same-resolution bandwidth. For high speed, carry look ahead adders (CLA, the fastest CPA available) are commonly used (see Figure 4.7.) and slow ripple carry adders (RPL) are avoided. Not considering detailed implementation aspects, such as the delays through wiring (which are important in modern submicron technologies), the carry propagate time through adders ultimately dictates the maximum sampling speed of a digital ΣΔ modulator. Classically, summations are done orthogonal to data flow (Figure 4.2.a.). Explicit computations of a CPA output requires a certain time to compute the carry out bit. The computations are done from the LSBs to MSBs, orthogonal to the propagation direction of data, which is from CPA input to its output. The method proposed in this work solves the carry propagate problem in adders in a negative feedback environment, such as ΣΔ modulators, by using a carry save (CS) form of representation for the two's complement numbers.

An extension of the concept of 3/2 carry save summation discussed in Section 3.7. is the 7/3 CS summation. When computing the sum of four, five, six or seven n-bit two's complement numbers, the same order bits can be added independently and their integer sum value can be encoded as a three bit binary number (see Figure 4.3).

$$
\begin{array}{lcl}
m_{n-1}m_{n-2}\ldots m_1m_0 \quad + & & \\
q_{n-1}q_{n-2}\ldots\ldots q_1q_0 & & 0 \quad 0 \quad s_{n-1}\ s_{n-2}\ldots\ldots\ldots s_1\ s_0\ + \\
p_{n-1}p_{n-2}\ldots\ldots p_1p_0 & \Leftrightarrow & 0 \quad c_{n-1}\ c_{n-2}\ \ldots\ldots\ldots\ldots c_1\ c_0\ 0 \\
r_{n-1}r_{n-2}\ldots\ldots r_1r_0 & & d_{n-1}\ d_{n-2}\ \ldots\ldots\ldots\ldots\ldots d_0 . 0\ \ 0 \\
v_{n-1}v_{n-2}\ldots\ldots v_1v_0 & & \\
u_{n-1}u_{n-2}\ldots\ldots u_1u_0 & & \\
t_{n-1}t_{n-2}\ldots\ldots t_1t_0 & &
\end{array}
$$

Equivalence:

$$(d_i\ c_i\ s_i)_2 \quad \Leftrightarrow \quad (m_i + q_i + p_i + r_i + v_i + u_i + t_i)$$

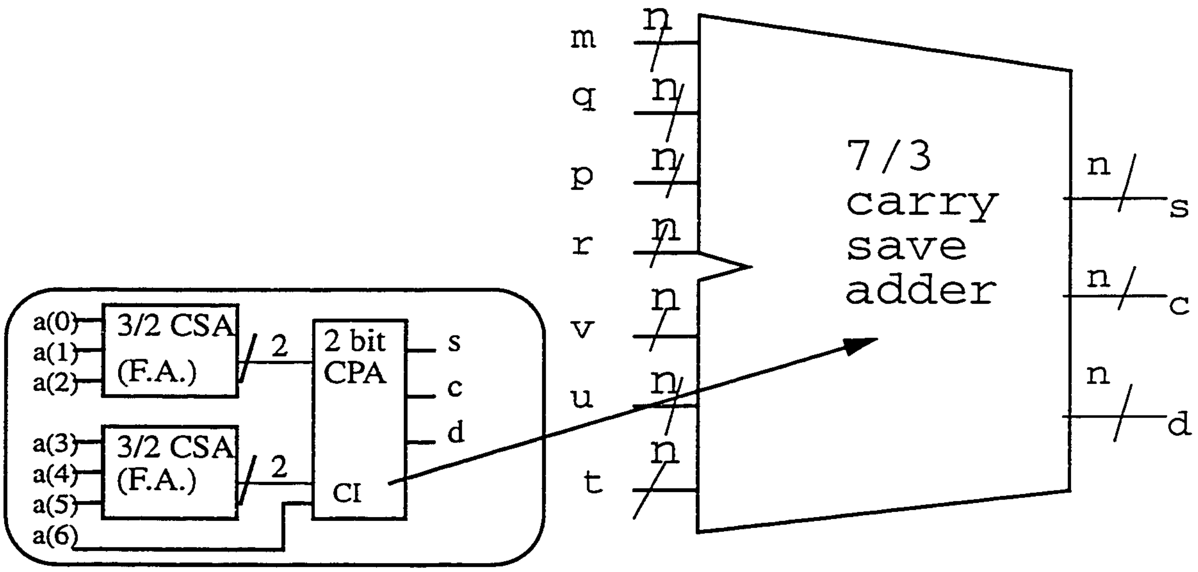


Fig.4.3. 7/3 carry save summation and circuit implementation

The corresponding bits of the triplet can be called, in increasing order of importance, the sum bit (s), the carry bit (c) and the supercarry bit (d). The result of the summation is then a triplet of n-bit numbers: sum, carry and supercarry. The real sum can be retrieved by adding the sum number, the carry number shifted left one position and the supercarry number shifted left two positions. The 7/3 CS summation process is shown in Figure 4.3. Also shown in Figure 4.3. is an implementation of a 7/3 CSA cell. Two FAs and a two bit carry propagate adder (CPA) can encode the integer sum of the seven one bit inputs into a three bit binary number $(d, c, s)_2$.

The real binary number computed by adding the sum number, the carry number shifted left one position and the supercarry number shifted left two positions has a maximum length of n+3. There is then at least one 7/3 CS representation of a binary number of length n+3 (Figure 4.4.).

$$b_{n+2}b_{n+1}b_nb_{n-1}\cdots\cdots b_1b_0 \iff \begin{bmatrix} 0 & 0 & s_{n-1} & s_{n-2} & \cdots\cdots\cdots & s_0 \\ 0 & c_{n-1} & c_{n-2} & \cdots\cdots\cdots & c_0 & 0 \\ d_{n-1} & d_{n-2} & \cdots\cdots\cdots & d_0 & 0 & 0 \end{bmatrix}$$

Fig.4.4. 7/3 CS representation
of a two's complement number

If the value of the final sum is much less than the digital representable range, a few of its most significant bits are identical (the reverse situation of sign bit extension). Let b be the integer value of this number. If:

$$b < \text{maximum range}/8 \iff b_{n+2} = b_{n+1} = b_n \qquad (4.1)$$

then the representations on n+3 bits and n+1 bits are equivalent and it is true that

$$b_{n+2}b_{n+1}b_n\cdots\cdots b_1b_0 \iff b_nb_{n-1}\cdots\cdots\cdots b_1b_0$$

Since eliminating two most significant bits of the number $b_{n+2}b_{n+1}....b_0$ is equivalent to truncating the two leftmost columns in the matrix representation in Figure 4.4., the relation shown below holds:

$$b_{n+2}b_{n+1}b_n.....b_1b_0 \Leftrightarrow \begin{bmatrix} s_{n-1} & s_{n-2} & \cdots\cdots\cdots & & s_0 \\ c_{n-2} & c_{n-3} & \cdots\cdots & c_0 & 0 \\ d_{n-3} & \cdots\cdots & d_0 & 0 & 0 \end{bmatrix}$$

Fig.4.5. Truncated 7/3 CS representation
of two's complement numbers

If (4.1) is true, the matrix representations shown in Figure 4.4. and Figure 4.5. are equivalent. The condition (4.1) can easily be satisfied by providing enough sign bit extension at the circuit input (see also Figure 4.7.). The equivalence between the two 7/3 CS forms of representation for two's complement numbers forms the basis for the implementation of the proposed digital ΣΔ modulator.

4.2. Carry save accumulator

The implementation of a CS accumulator is straight forward, given that condition (4.1) is satisfied. One computes the full 7/3 CS representation of the sum (Figure 4.4.) and feeds back the truncated 7/3 CS representation (Figure 4.5.).

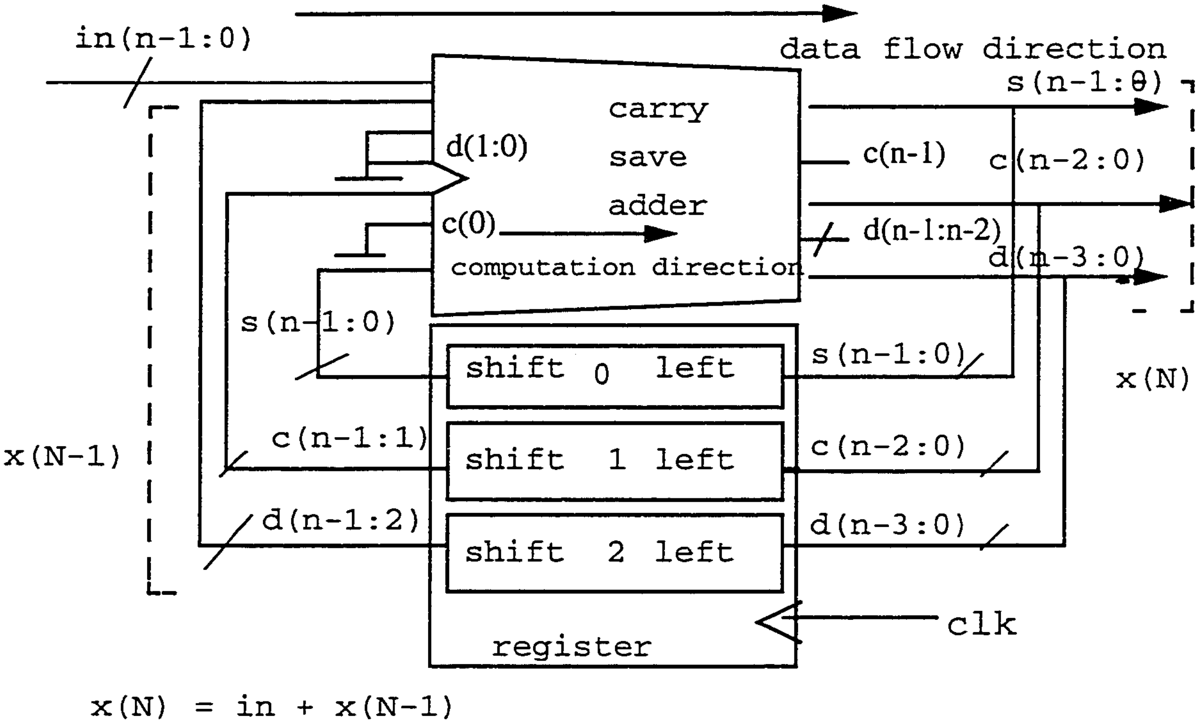


Fig.4.6. CS accumulator

Two observations can be made about the CS accumulator implementation. One is that normal two's complement representation of numbers is a particular case of the 7/3 CS form of representation, where the carry and supercarry numbers are equal to zero. This implies that two's complement numbers and numbers in 7/3 CS form can be added directly (Figure 4.6).

A second observation is about the use of sign bit extension in CSA. In an adder using normal carry propagate arithmetic (where carry out bits are explicitly computed; a CLA for example), the word width of the output has to be increased by one in order to provide space for the carry out bit, while the two inputs do not need a sign bit extension if they have the same word widths. By contrast, in a CSA with output word width n, all inputs have to have the same word width as the output. If one of the inputs has a word width n1 <

n and is expressed in normal two's complement form, sign bit extension has to be employed. The use of sign bit extension in CPA and in CSA is exemplified in Figure 4.7.

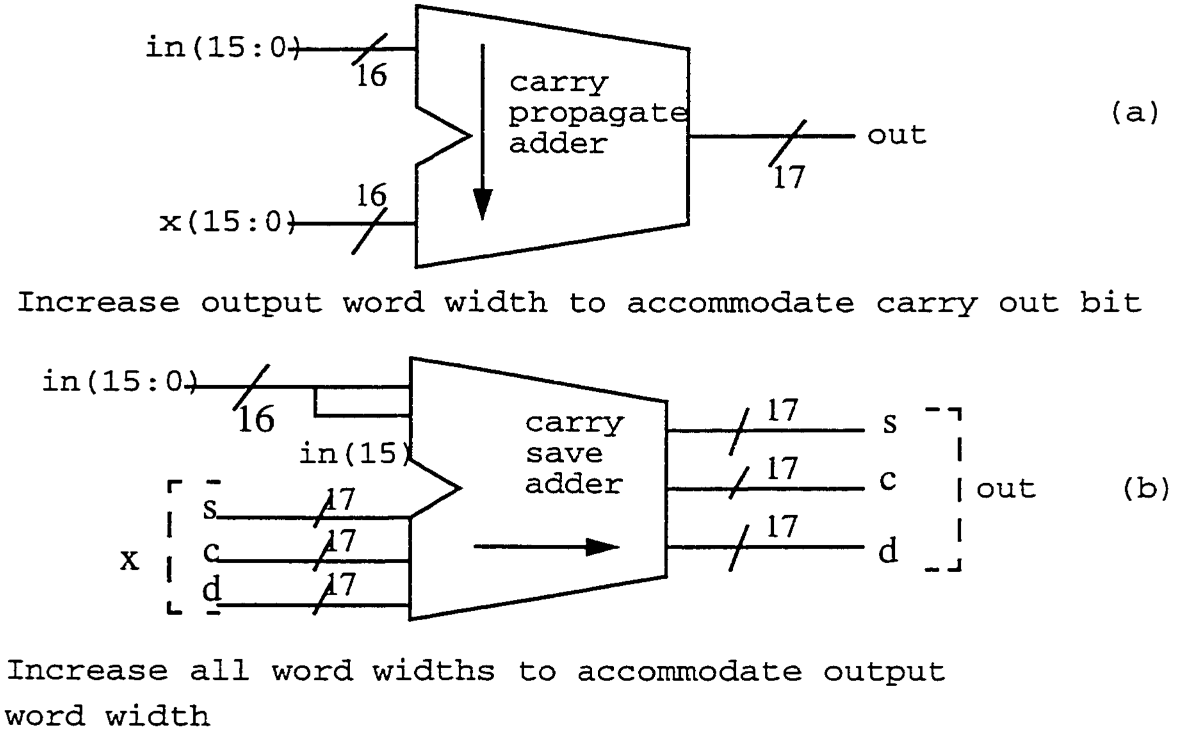


Fig.4.7. Sign bit extension (a) CPA; (b) CSA.

4.3. Approximate quantizer

Quantization is not an easy task when using carry save arithmetic. For example, computing the exact sign bit value of the 7/3 CS number in Figure 4.5. (1 bit quantization) implies adding the three matrix lines using CPA. This is a time consuming process, not suited for high speed ΣΔ modulators. In turn, an approximate CS quantizer is proposed in this work. Let us start from the equivalence demonstrated in Section 4.1.:

:

$$b_{n+2}b_{n+1}b_n\cdot\cdot\cdot\cdot\cdot.b_1b_0 \iff \left[\begin{array}{lllll} s_{n-1} & s_{n-2} & \cdots\cdots\cdots & & s_0 \\ c_{n-2} & c_{n-3} & \cdots\cdots & c_0 & 0 \\ d_{n-3} & \cdots\cdots\cdots & d_0 & 0 & 0 \end{array}\right]$$

which is true when $b_{n+2} = b_{n+1} = b_n$. One bit quantization of two's complement numbers is implemented by taking the sign bit (either b_{n+2}, b_{n+1} or b_n). The exact value of the sign bit can be computed by adding the lines in the 7/3 CS matrix form of representation shown above. A fast and easy way to perform the same summation is to make a 3/2 CS summation of the matrix lines, to compute a carry number and sum number and then to add them using normal carry propagate arithmetic (add the sum number and the carry number shifted left one position). The result will be a maximum n+1 bit two's complement number, identical in value with $b_{n+2}b_{n+1}.....b_0$.

An approximate quantization starting with the CS form of representation is to take the K << n most significant columns in the above matrix, do a 3/2 CS summation and then add the sum and carry K-bit numbers using normal CPA. If two bit sign extension has been provided at the modulator input (all word widths increased by two), the result will be a K+2 bit number with the property that bits K+2, K+1 and K are identical and represent the actual sign bit of the quantized number with a certain probability. The probability that the leftmost bit (i.e. the quantized bit in a one bit quantizer) is the true value of the sign bit of the real two's complement equivalent number can be increased by using two methods:

a) allowing for more input sign bit extension, or

b) increasing the number of columns, K, used to compute the approximate sign bit.

The mathematical process of approximate CS quantization is shown as a flow in Figure 4.8. for a 1 bit output ΣΔ modulator with 2 bit input sign extension.

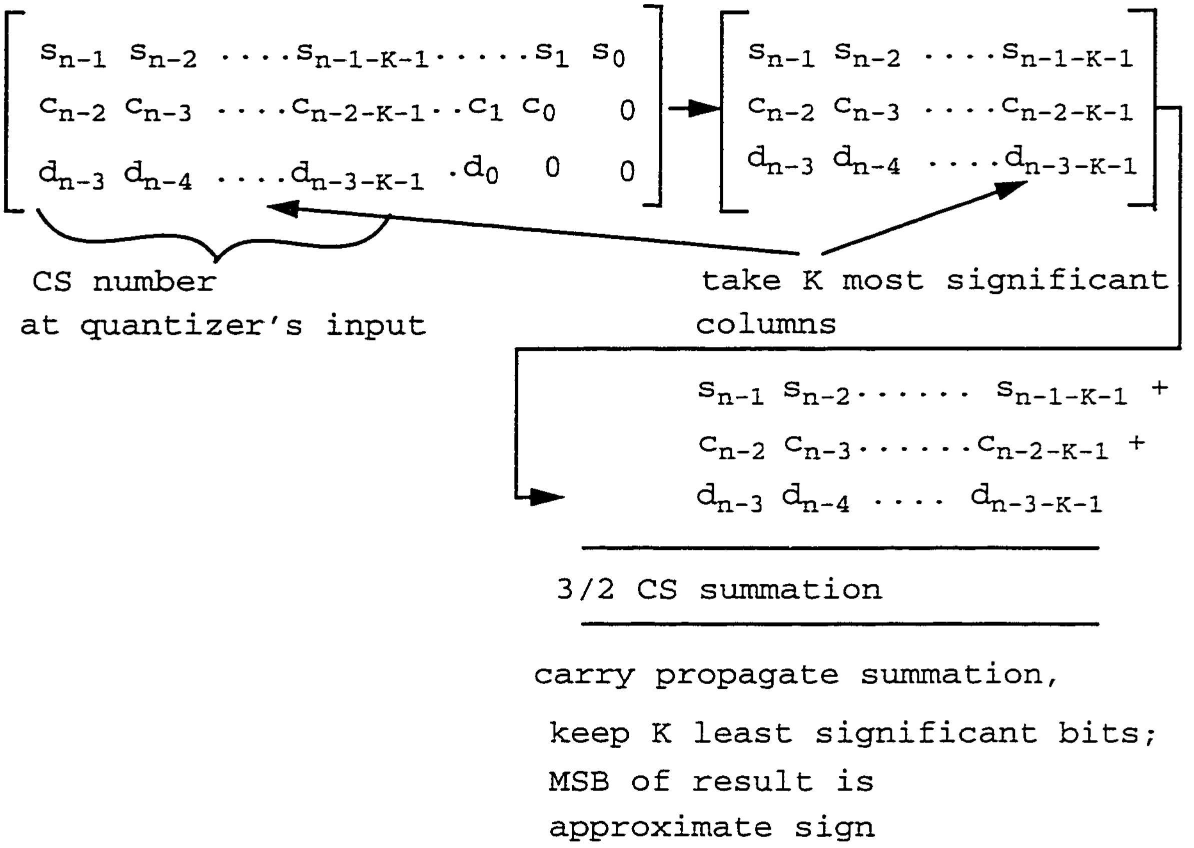
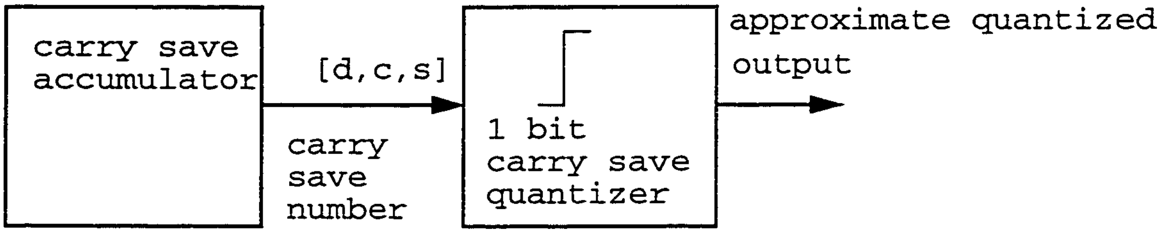

s_{n-1} s_{n-2} ....$s_{n-1-K-1}$.....s_1 s_0
c_{n-2} c_{n-3} ....$c_{n-2-K-1}$..c_1 c_0 0
d_{n-3} d_{n-4} ....$d_{n-3-K-1}$ .d_0 0 0
s_{n-1} s_{n-2} ....$s_{n-1-K-1}$
c_{n-2} c_{n-3} ....$c_{n-2-K-1}$
d_{n-3} d_{n-4} ....$d_{n-3-K-1}$
CS number
at quantizer's input
take K most significant
columns
s_{n-1} s_{n-2}...... $s_{n-1-K-1}$ +
c_{n-2} c_{n-3}......$c_{n-2-K-1}$ +
d_{n-3} d_{n-4} .... $d_{n-3-K-1}$
3/2 CS summation
carry propagate summation,
keep K least significant bits;
MSB of result is
approximate sign

Fig.4.8. CS approximate quantization
2 bit sign extension throughout the ΣΔ modulator

The final K+2 bit number at the output of the approximate CS quantizer has this property: the top three bits are usually the same for moderate input signal levels. The MSB (bit K+2) is the most accurate estimate of the sign, but the second bit (bit K+1) is usually the same and can be computed more quickly. Figure 4.9 illustrates the approximate CS quantization process as a sequence of pseudo instructions, while Figure 4.10. gives the reader a practical example.



1)take K (~4) MSB's of [d,c,s]; [d,c,s]=>$[d,c,s]_K$;

2)do 3/2 CS reduction: $[d,c,s]_K$=>(carry, sum) ;

3)compute (2*carry+ sum) = b'; b" = take K LSBs;

4)take MSB of b";

If sign bit extension in ΣΔ modulator is M,
out of max K+2 bits of b', M+1 bits are "sign bits"

Fig.4.9. One bit approximate CS quantization algorithm
using the four most significant columns
in the 7/3 CS representation of quantizers's input.
(two bit sign extension in ΣΔ modulator).

Note that the quantizer output used is not the MSB of the computed sum, but the next MSB. The CPA needed in this case is shorter and the loss in probability when computing the real value of the sign bit is not significant when the quantizer is used in a ΣΔ modulator. This argument will be enforced by simulations shortly. In turn, this choice allows for a significantly shorter critical path in the digital ΣΔ modulator, speeding-up its operation.

In general, if the sign bit extension in the ΣΔ modulator is M, M+1 bits out of K+2 bits of b' are considered sign bits (with a certain probability, as explained before). This aspect is of crucial importance for understanding the method.

Figure 4.11. shows a z domain block diagram of a second order CS ΣΔ modulator. Both quantization error and the error due to approximate CS quantization have been modeled as uncorrelated additive white gaussian noise processes (AWGN).

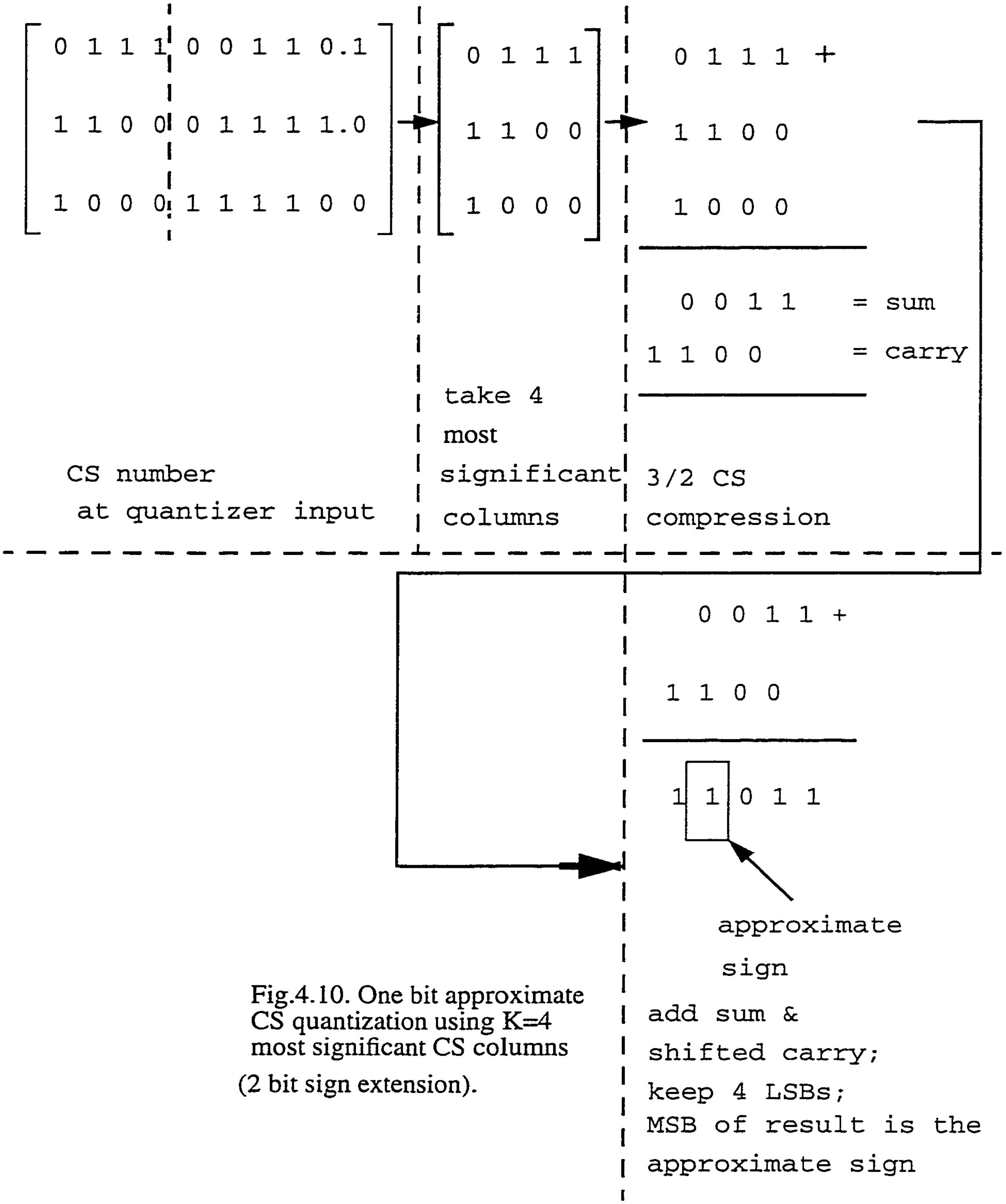


Fig.4.10. One bit approximate CS quantization using K=4 most significant CS columns (2 bit sign extension).

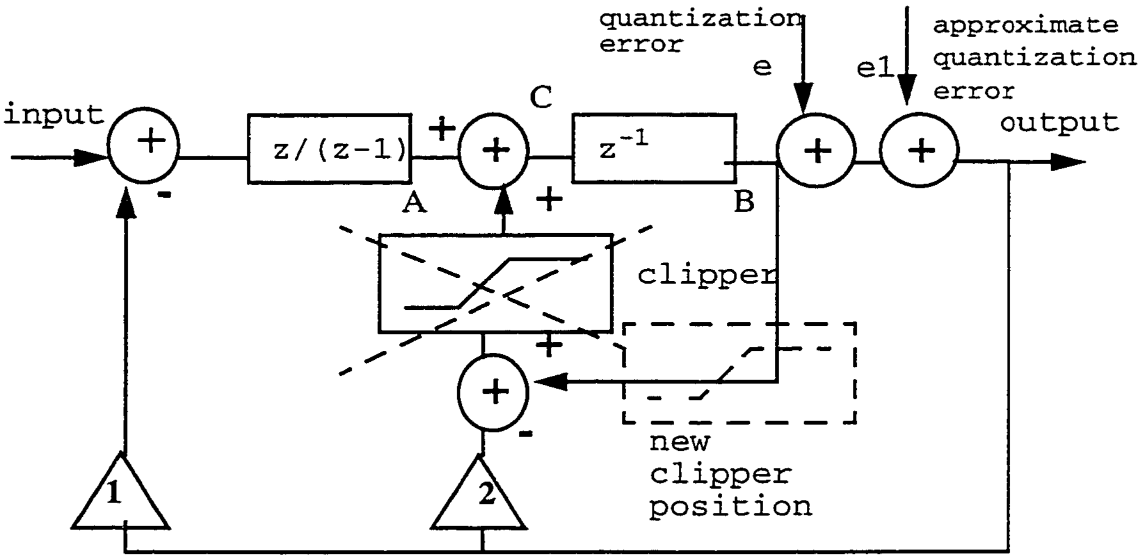


Fig.4.11. z domain diagram of a second order CS ΣΔ modulator

To illustrate the additional error resulting from approximate quantization, let us consider a number represented by its sum and carry components: sum = 1111111, carry = 0000011. An exact one bit quantization will imply adding the sum and shifted carry and taking the sign bit (a logic ONE). If CS truncation of the 5 most significant columns is performed and then a one bit quantization is done on the result (after CPA summation), the sign bit is a logic ZERO. Repeating the process with 6 most significant CS columns gives the correct result (a logic ONE).

Simulations will show that the approximate quantization error is indeed shaped out of the band of interest and the overall second order differentiation of the noise (due to both quantization and approximate CS quantization) is present. This is based on a careful choice of K, the number of most significant columns used to do the approximate CS quantization, and a reasonable sign bit extension throughout the digital ΣΔ modulator.

A useful subproduct of the approximate CS quantization process is the approximate overflow bit. The approximate overflow bit signals that the approximate sign (the quantized output, for one bit quantization) has a "wrong" value, so that its inverted value is the "true" one. Since we are discussing the whole process probabilistically, one should not expect probabilities of 1 for the "wrong" or "true" values. The approximate overflow bit is computed using the carry in and carry out bits from the computational cell whose output is the designated approximate quantized value (bit K in Figure 4.8.). Figure 4.12. shows the circuit level implementation of a approximate CS quantizer using K = 4 most significant CS columns. The sign bit extension in the CS ΣΔ modulator is two in this case.

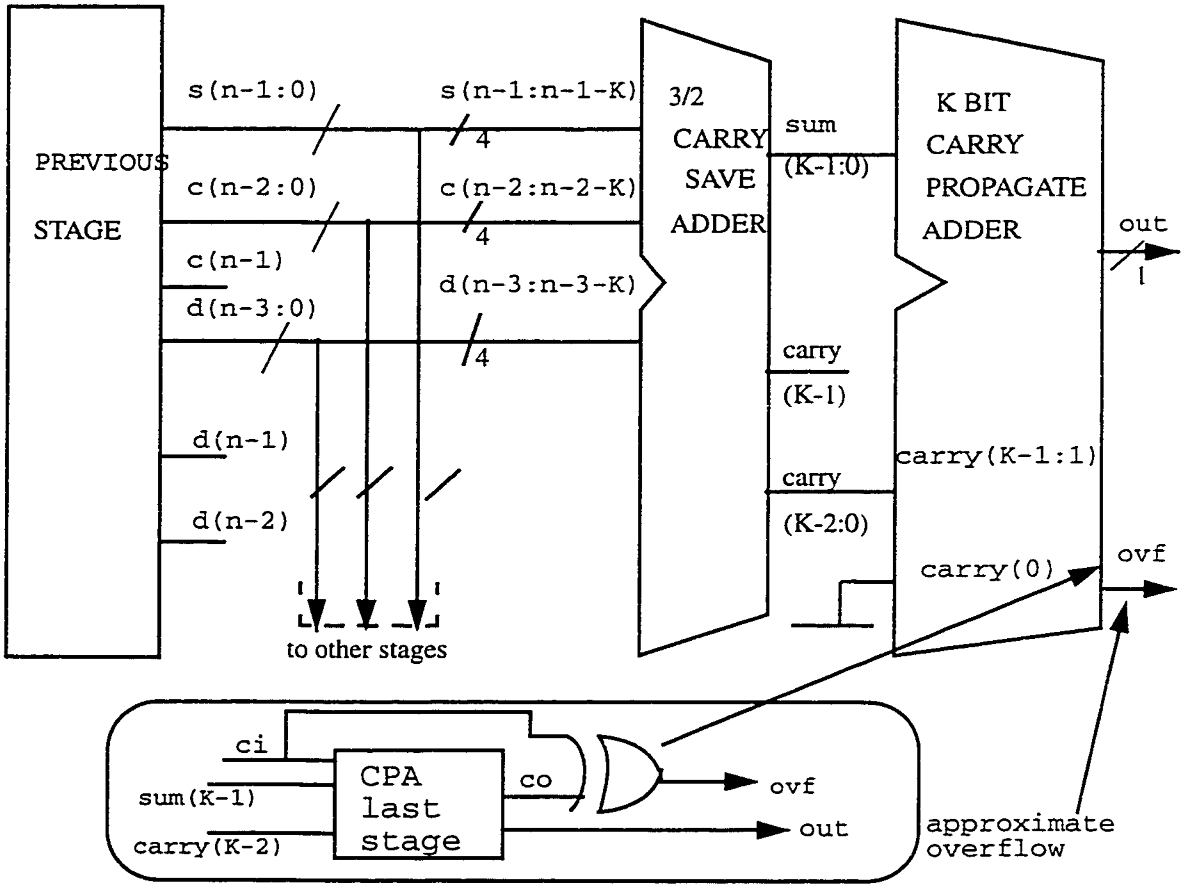


Fig.4.12. One bit approximate CS quantizer using K = 4 most significant CS columns; (two bit sign extension).

For example, Figure 4.13. shows two 4-bit two's complement summations: a correct one and one with overflow. In our circuit, the approximate overflow bit is computed in the same manner.

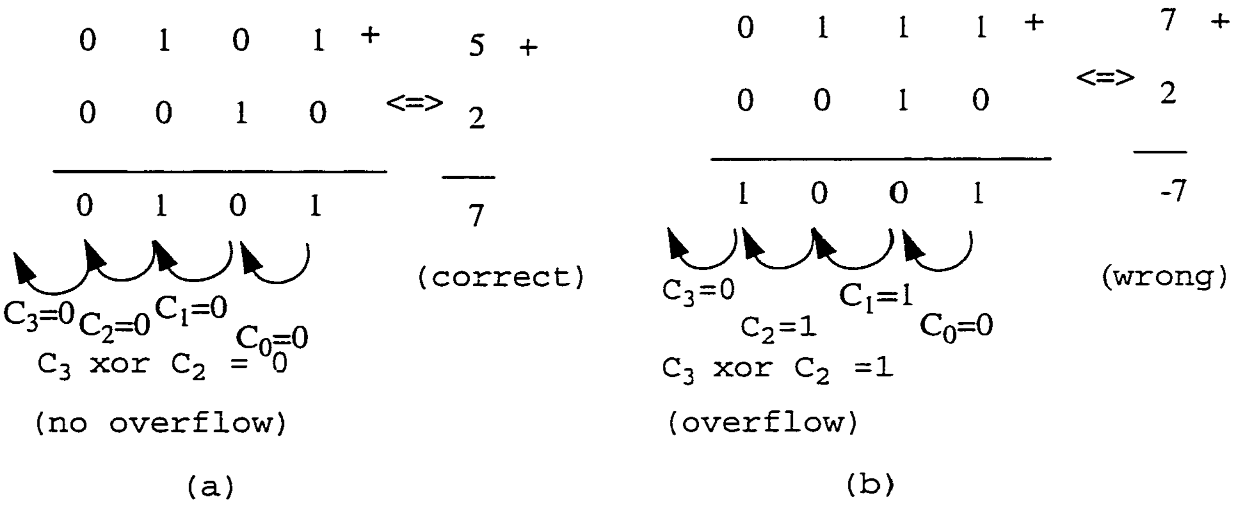


Fig.4.13. Summation of four bit two's complement numbers (a) correct result; (b) overflow situation.

In Section 3.6., the advantages of using multibit quantization in ΣΔ modulators were briefly described. They are repeated below:

a) for an n-bit quantizer, the in-band resolution is increased by 6(n-1) dB;

b) more stability, increased input range for which the modulator is stable;

c) same resolution for lower speed, thus less power dissipation;

In the case of a multibit approximate CS quantizer, more quantized bits require a wider CPA (the adder used is a CLA for speed reasons (see Figure 4.2.)).

For example, for a three bit quantizer and three bit word width extension throughout the modulator (corresponding to three bit sign extension), a minimum 6 bit CPA is need. Indeed, since the adder output is of the form $u_5u_4u_3u_2u_1u_0$, bits u_4, u_3 and u_2 are usually the same for moderate input levels (see also Fig.3.16. for peak integrator output levels as a function of input DC code). Bits u_1 and u_0 are the middle bit and the least significant bit of the quantizer output, respectively. The quantized 3 bit output is then $u_5u_1u_0$ (shown in Figure 4.14.).

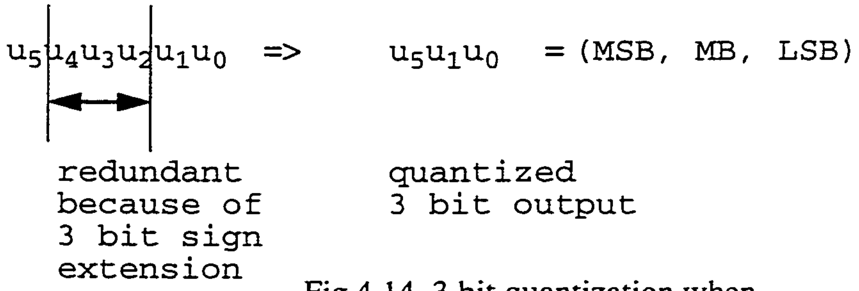

$u_5u_4u_3u_2u_1u_0$ => $u_5u_1u_0$ = (MSB, MB, LSB)
redundant because of 3 bit sign extension
quantized 3 bit output

Fig.4.14. 3 bit quantization when the sign bit extension is 3

4.4. Approximate clipper

In Chapter 3, where the second order digital ΣΔ architecture was introduced to the reader, we discussed the problem with increased signal levels at the output of the second integrator as modulator DC code input approaches the rails (for a one bit modulator). The risk of two's complement wrap-around at the quantizer input was prevented by using a combinational clipper, as in Figure 4.11. The maximum signal level in point (B) is controlled by adequately choosing the sign bit extension throughout the modulator, as well as the clipping levels. One modification to Figure 4.11. allows the clipping stage to be implemented in the second order CS ΣΔ modulator (see Figure 4.16.).

The clipper can be moved to the new position shown with dots in Figure 4.11. Let δ be the one bit quantizer step. If the clipping levels are chosen to be +/- 4δ and remembering that the maximum signal levels after the first integrator are +/- 2δ (see Figure 3.17.) then the maximum signal level at point (C) is

$$Max(C) = \pm 4\delta \pm 2\delta \pm 2\delta = \pm 8\delta$$

Thus using three bit sign extension throughout the modulator, the maximum signal level after the second integrator (point B) in Figure 4.11. is kept under control and no two's complement wrap-around will take place.

The clipping can be implemented using saturation arithmetic, as explained in Section 3.5. For coherence purposes, Figure 4.16. reproduces the situation described in Figure 3.17 (b, c): a comparison between the case when saturation arithmetic is used, as opposed to normal two's complement wrap-around.

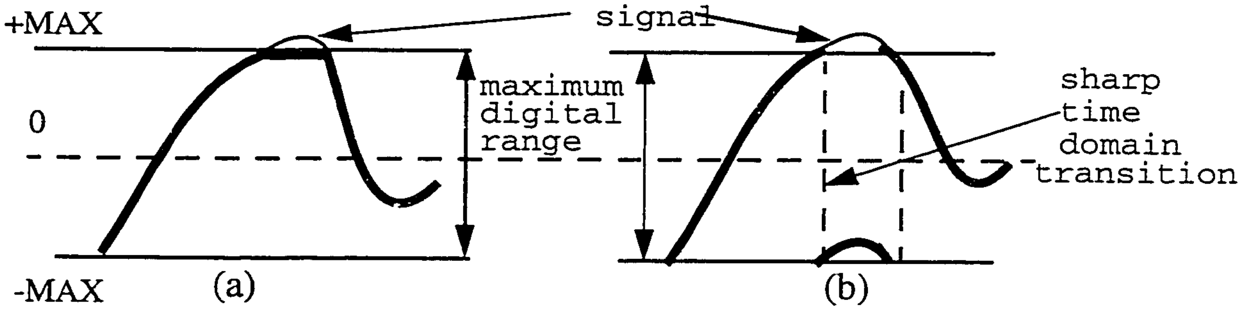


Fig.4.15. (a)saturation arithmetic
(b) two's complement wrap-around

Figure 4.16. shows a block diagram for the second stage of the second order ΣΔ modulator using approximate clipping.

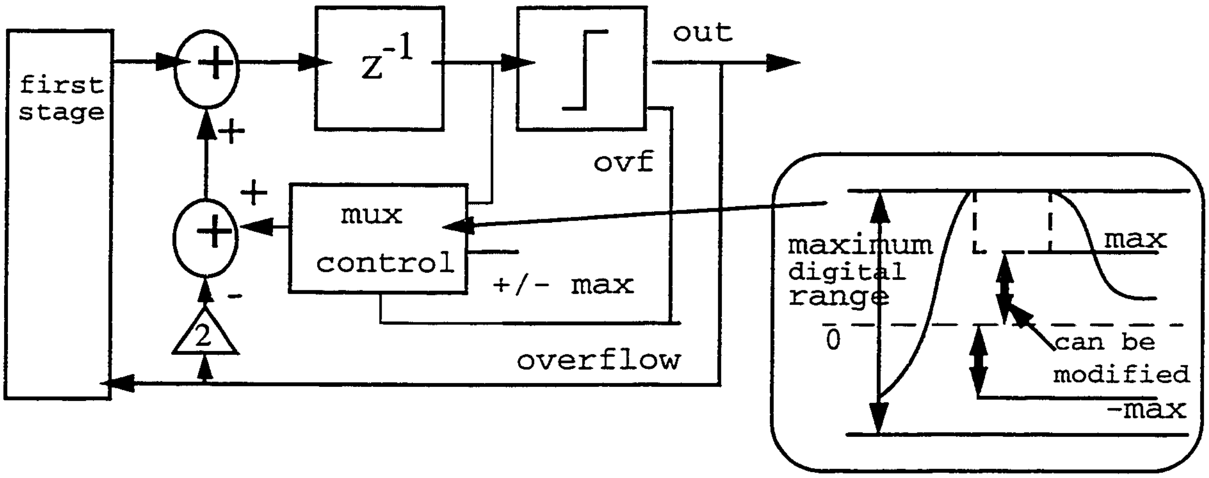


Fig.4.16. Second stage of the second order CS ΣΔ modulator using saturation arithmetic and approximate clipping

One essential element of the approximate clipper using saturation arithmetic is the approximate overflow bit, the subproduct of the approximate CS quantization. A piece of code describing the behavior of the approximate clipper can be:

```
If (approximate overflow == 1)
          multiplex predetermined value,
          invert sign;
else
          multiplex result of
          carry save summation;
end;
```

In the case an approximate overflow is flagged, a predetermined value is to be multiplexed and fed back to the input of the second integrator, instead of the result of the CS summation (Figure 4.15.). The approximate overflow can be exactly the opposite of the true value (one bit, can be either a ONE or a ZERO), so the approximate clipper can make big mistakes, translated into sharp time domain transitions and spectrally rich, much like those in Figure 4.15.b. The prescribed maximum and minimum values can be chosen in such a way as to minimize this effect (see Figure 4.15.). In the circuit proposed in this work, these values are chosen to be at half the representable range.

For example, for a 16 bit modulator input with three sign bit extension, the prescribed values are:

```
if approximate overflow ==1 (an "error")
      multiplex the opposite prescribed value,
      half representable range
        [~sgn ~sgn 11111111111111111]
else
      multiplex the result of
      carry save summation
end;

sgn = approximate quantized sign bit
```

Choosing optimized values for the prescribed quantities in case of approximate overflow can increase the DC code input range for which the CS ΣΔ modulator is stable.

An observation: errors due to approximate CS quantization and approximate clipping are not statistically independent. However, simulations have shown that energy corresponding to both types of errors is shaped out of the band of interest in the CS ΣΔ modulator.

4.5. Feedbacks

A one bit quantizer uses the same method as the implementation discussed in Section3.5 [Su93]. The quantizer outputs are chosen to be at half the positive and negative representable ranges respectively. For example, for a 16 bit modulator input and a three bit sign extension, the one bit quantizer outputs are:

$$HIGH = 0000100000000000000 \qquad (4.2)$$

$$LOW = 1111100000000000000 \qquad (4.3)$$

The two numbers are each other's two's complement. If sgn is the approximate quantized sign bit, the outer feedback branch is implemented by inverting the quantized sign, thus adding only an inverter delay in the critical path:

Outer_Feedback = [~sgn ~sgn ~sgn ~sgn 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

The inner feedback needs a multiplication of two, implemented as a one position left shift:

Inner_Feedback = [~sgn ~sgn ~sgn 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

The feedback two's complement numbers can be represented in CS form, with the carry and supercarry lines containing null elements:

$$[\sim sgn\ \sim sgn\ 1\ 1..1] \quad <=> \quad \begin{bmatrix} \sim sgn & \sim sgn & 1\ ......1 \\ 0 & 0 & 0\ ....0 \\ 0 & 0 & 0\ ....0 \end{bmatrix}$$

which allows direct CS summations between the feedbacks in two's complement form and numbers in CS form.

In case of a three bit quantizer and three sign bit extension throughout the CS ΣΔ modulator, for the same 16 bit input, one can choose the maximum and minimum tresholds to be the same as (4.2) and (4.3) respectively. The quantized output is then the 19 bit number

$$\text{Quantizer\_Output} = [u_5\ u_5\ u_5\ u_5\ u_1\ u_0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

and the two feedbacks are implemented using only inverters:

$$\text{Outer\_Feedback}= [\sim u_5\ \sim u_5\ \sim u_5\ \sim u_5\ \sim u_1\ \sim u_0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

$$\text{Inner\_Feedback}= [\sim u_5\ \sim u_5\ \sim u_5\ \sim u_1\ \sim u_0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$$

The ONE and ZERO levels are to be hardwired to the inputs of the respective CS adders.

4.6. Proposed implementation

Figure 4.16. shows the second stage of a 3 bit output second order ΣΔ modulator using CS arithmetic. The outputs from the first stage, the feedback from the quantizer and the feedback from the register output are summed using a 7/3 CSA

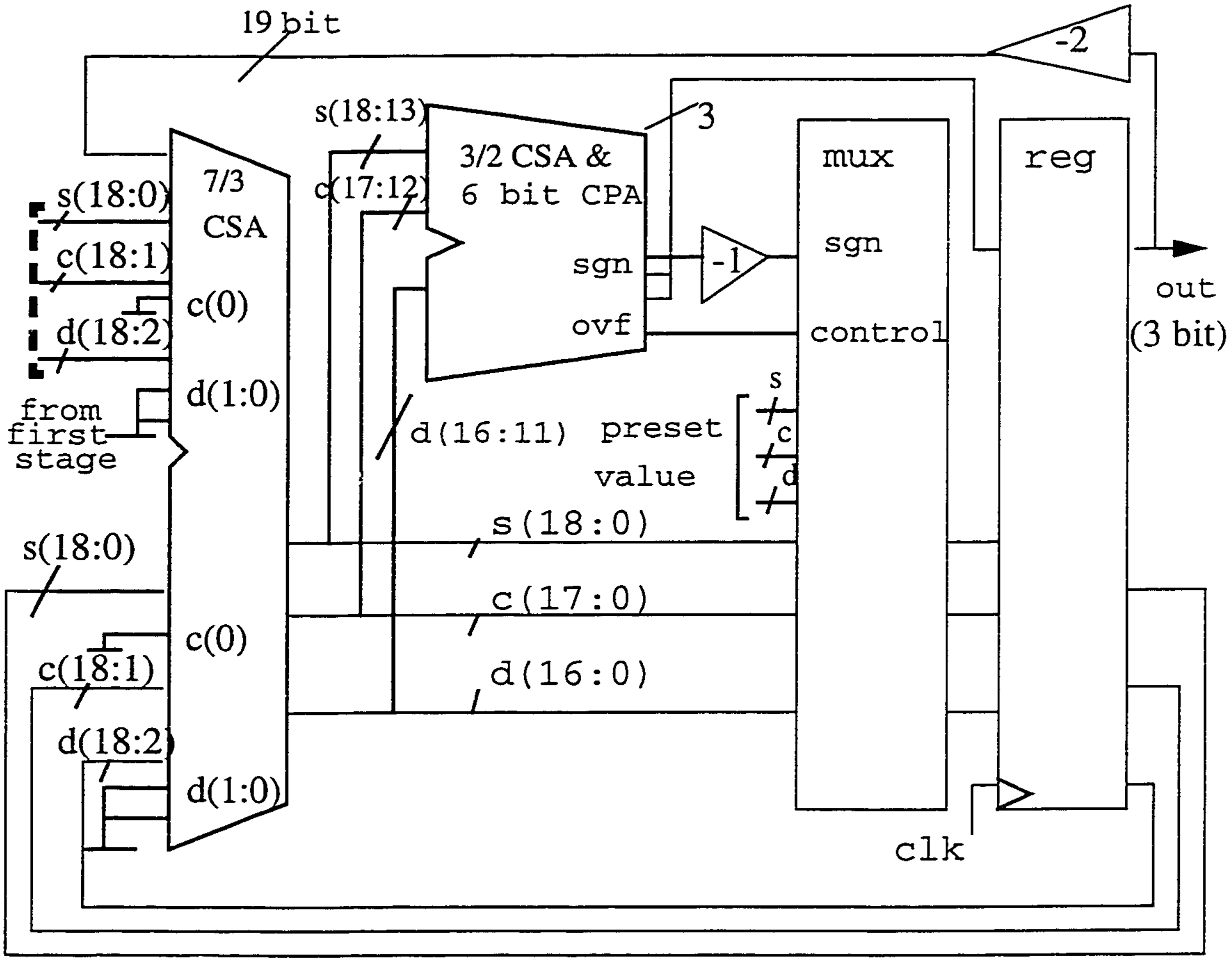


Fig.4.16.Second stage of a 3 bit output second order CS ΣΔ modulator

The CS accumulator is not different from the one in Figure 4.6. The outputs from the first stage are of the same form as the feedback of the CS accumulator in that Figure. The only difference is the truncated CS matrix of the first stage output is fed to the second stage.

Approximate CS quantization is done by taking the six most significant columns of the 7/3 CS output, doing a 3/2 CS compression and then a carry propagate summation using a six bit CLA adder. The approximate quantization is done by taking the most significant, the fifth and the sixth bit of the CLA output, as explained in Section 4.3. The approximate overflow, computed as in Section 4.4., is used to do an approximate clipping. Both the output of the approximate CS quantizer and that of the multiplexer are passed through a register, whose job is to provide one sample delay. The sign bit extension at the modulator input is chosen to be three (the wordwidths throughout the modulator are increased by the same quantity).

Some of the early simulations have been done for a one bit output second order CS ΣΔ modulator. The differences are:

a) the number of most significant columns used to do approximate CS quantization is four, as in the example shown in Figure 4.10.

b) the modulator output is the computed approximate sign bit

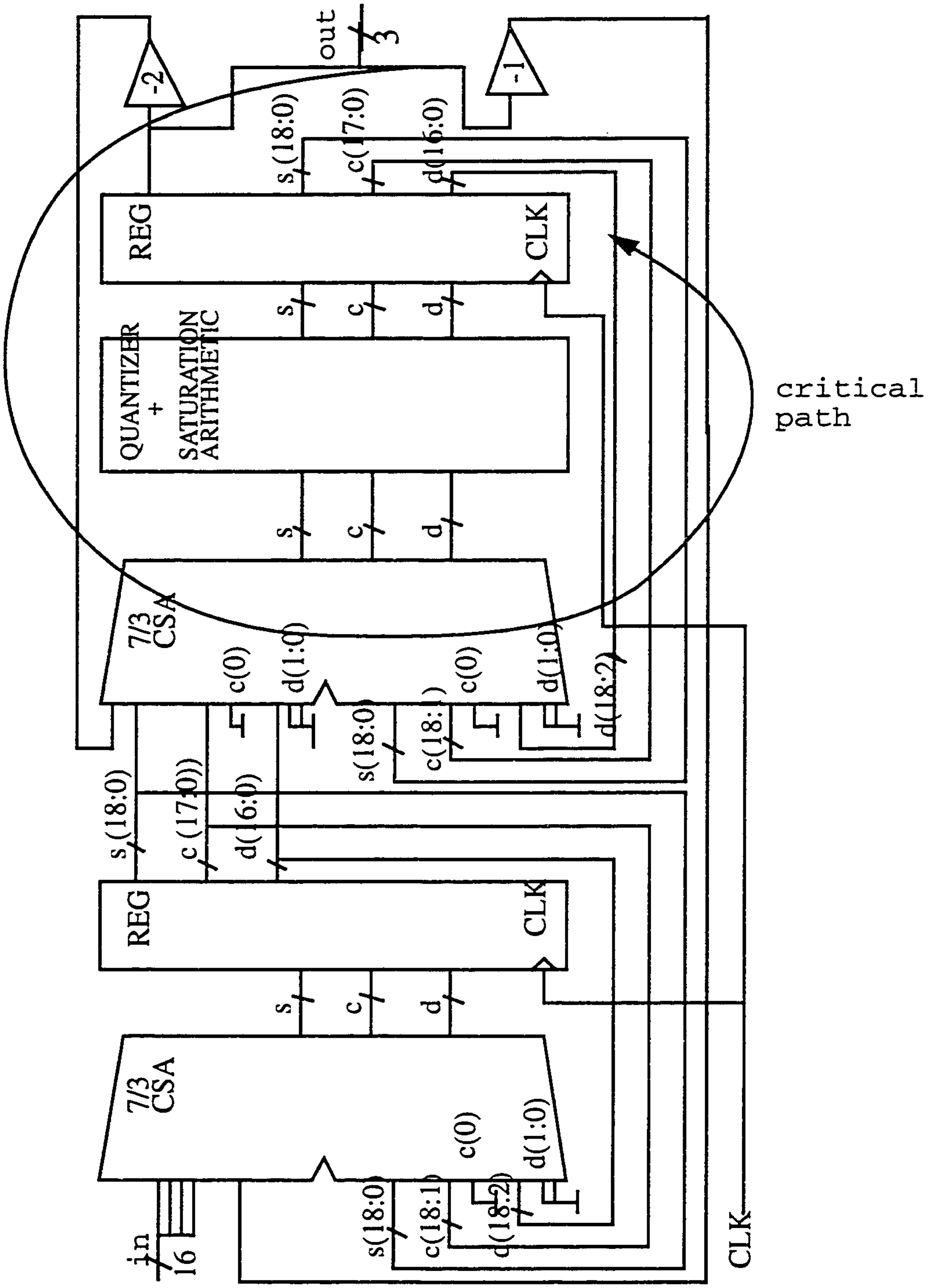


Fig.4.17.3 bit output second order CS ΣΔ modulator (3 bit input sign extension).

4.7. Simulation results

First, a one bit second order CS ΣΔ modulator was simulated at the mathematical level using Matlab. The Matlab code describing a CS ΣΔ modulator with 2 sign bit extension is shown in Appendix A. Figures 4.18 and 4.19. show the output spectrum of this ΣΔ modulator in two cases: when the approximate quantizer uses two and four of the most significant ΣΔ columns, respectively, when computing the output. The output spectrum is correctly noise shaped only in the second case.

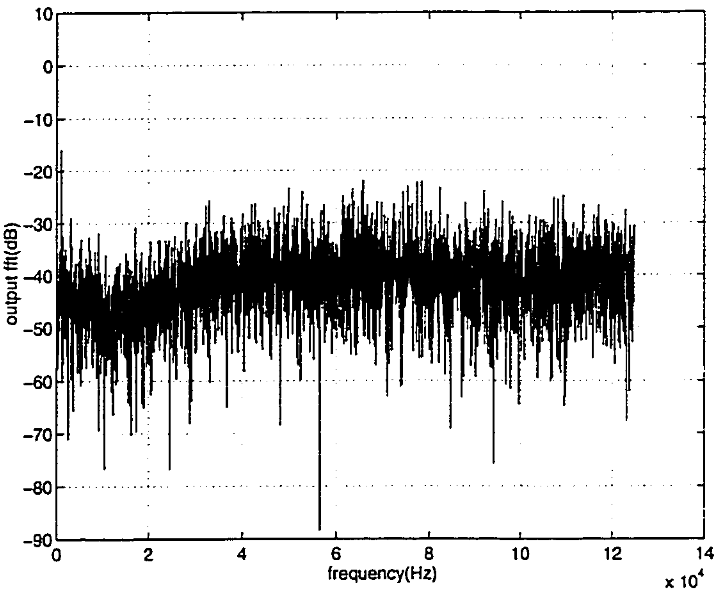


Fig.4.18. Output spectrum of a second order
CS ΣΔ modulator
with approximate CS quantizer using
K=2 most significant columns (two bit sign extension)

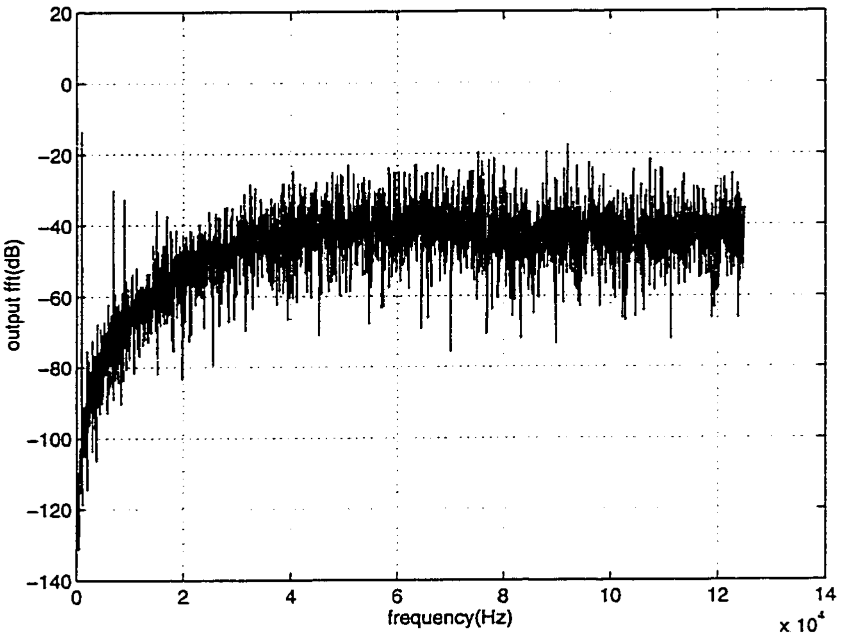


Fig.4.19. Output spectrum of a second order
CS ΣΔ modulator
with approximate CS quantizer using
K = 4 most significant columns (two bit sign extension)

In Figure 4.19., the spectral lines other than the baseband are a result of the fact that the incoming signal is not interpolated. Section 3.4. explained briefly the interpolation process and its role in ΣΔ DACs. The output of a digital ΣΔ modulator using CS arithmetic can be computed using only a few most significant CS columns. This argument is reinforced by the Matlab simulation results shown in Figure 4.20. The signal to noise and distortion ratio (SNDR) was computed for on OSR of 128 for three sinusoidal input levels, normalized to quantizer step. There is a 6 dB spacing between the consecutive input levels. The SNDR was computed using 2,3,..., 18 most significant CS columns of quantizer input.

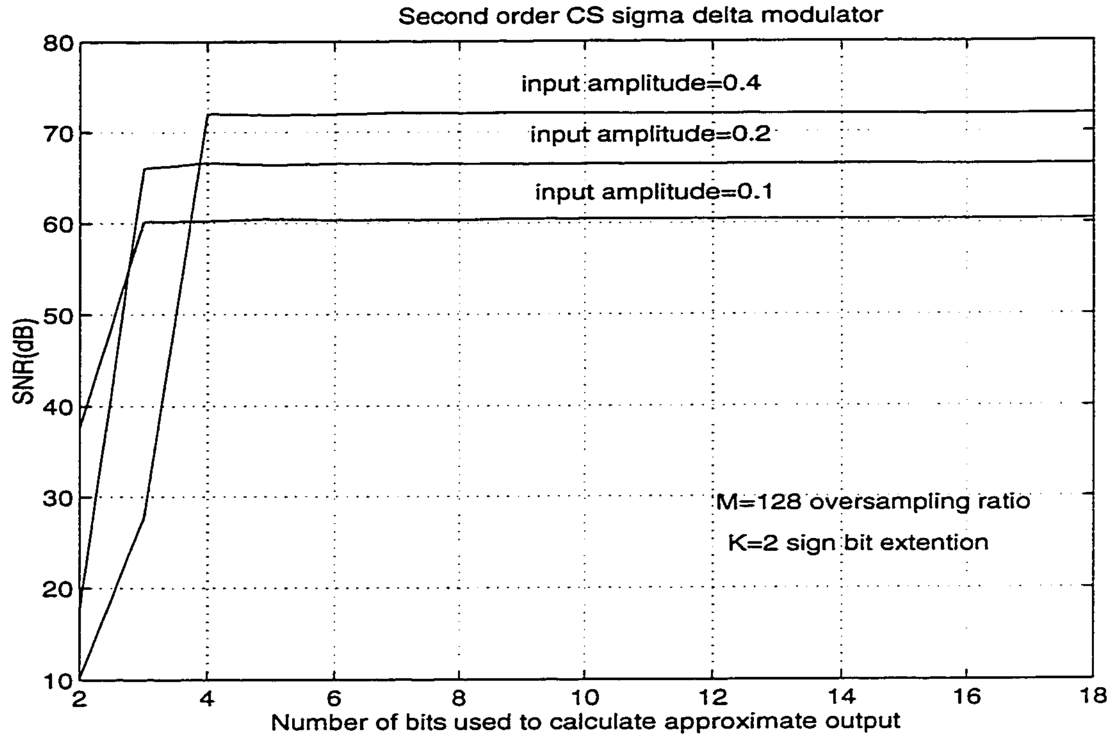


Fig.4.20. Output SNDR of a one bit
second order CS ΣΔ modulator
as a function of the number of most significant CS columns
used to compute the output

Two observations can be made regarding Figure 4.20.:

a) The SNDR is the same when CS ΣΔ modulator output is exactly computed and when the quantizer uses just a few of the most significant CS columns (for example four).

b) Linearity range is affected by the number of most significant CS columns used for approximate CS quantization. For a normalized input level of 0.4, four most significant CS columns are enough to compute the output. For an input level of 0.1 (12 dB less), only three most significant CS columns are necessary. The 0 dB reference is the quantizer step.

More word width extension bits throughout the modulator ensure higher dynamic range and the maximum value of sinusoidal baseband input for which the modulator is linear is increased. Unfortunately, the quantization process then requires a wider CPA, slowing down the CS ΣΔ modulator operation. There is a trade-off to be made between the required dynamic range and the maximum achievable ΣΔ modulator speed. Figure 4.21. below shows the SNDR variation as a function of normalized input level for a one bit second order CS ΣΔ modulator. The Matlab simulations were done for an OSR of 128.

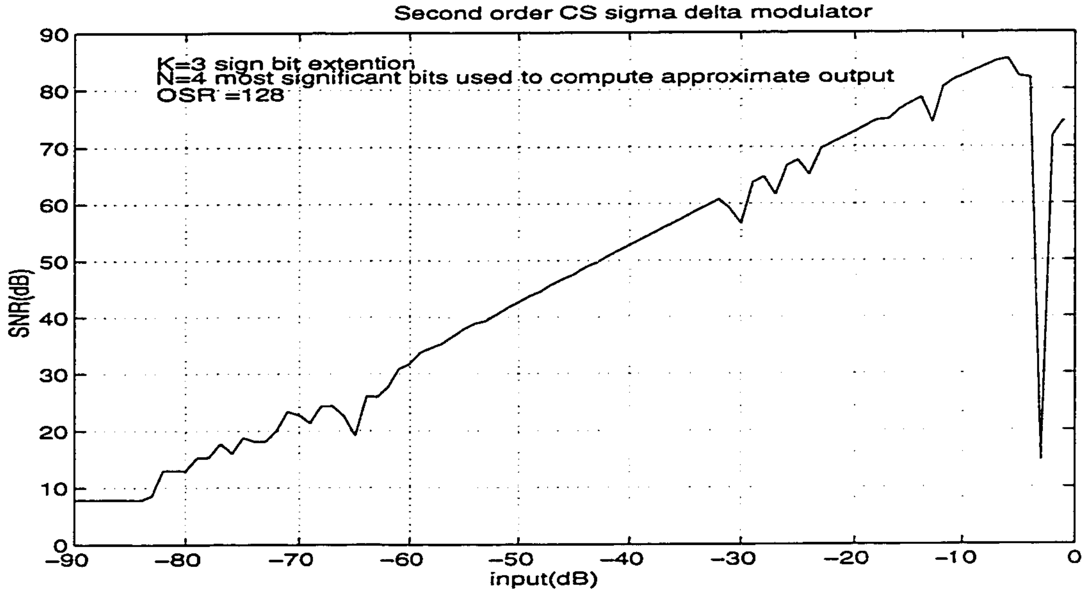


Fig.4.21. SNDR as a function of normalized input level for a one bit second order CS ΣΔ modulator (three bit sign extension, K = 4 most significant CS columns used to compute the output.); 0 dB = quantizer step

The ΣΔ modulator becomes unstable for input levels higher than -6 dB due to the fact that the one bit quantizer is overloaded most of the time (stays ONE or ZERO for long periods of time), which explains the roll-off of the characteristic (see also Section 3.2.). The CS ΣΔ modulator is not able to detect signals below -83 dB. It had reached its resolution treshold, determined by the OSR (see Section 3.5.). The quantization error energy, as well as the approximate quantization error energy, are not totally eliminated from the band of interest. The SNR can be characterized as a function of input for a given word width extension and number of truncated bits. Allowing for a wider word width extension, for example, the linearity range of the modulator will be increased at the expense of decreasing the speed, i.e. the maximum achievable SBR.

The next step was to verify the mathematical model in a circuit implementation. The circuit was implemented using ideal standard IEEE Synopsys gates (zero rise/fall times, no input capacitance, infinite fan-out, etc.). Figure 4.26. shows the block diagram of the Synopsys test set-up.

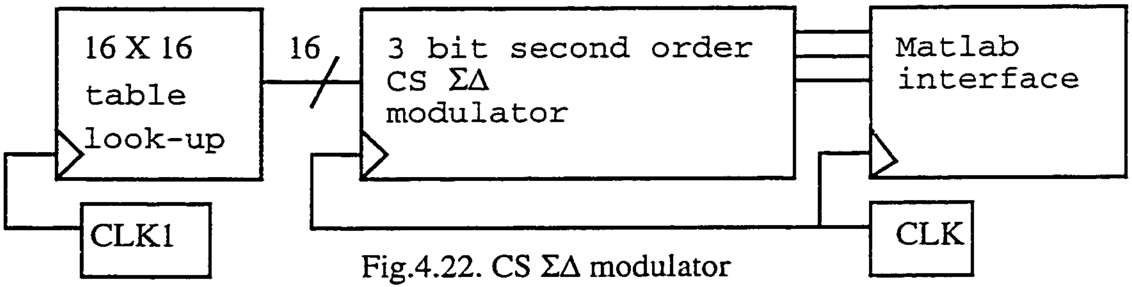


Fig.4.22. CS ΣΔ modulator Synopsys test set-up

The 16 X 16 table look-up, clock generators and Matlab interfaces are VHDL codes. The incoming 16 bit data is delivered at the sampling frequency F_{S1} generated by the clock generator CLK1. The clock generator CLK, synchronous with CLK1, generates the sampling frequency F_S, much higher than F_{S1}, used in the CS ΣΔ modulator. The three bit output of the CS ΣΔ modulator is sampled on the rising edge of CLK and the sample values are written as ASCII data to Matlab files. Each 3 bit word is translated to its equivalent level (between -4, 100, and +3, 011) and then the output spectrum is computed using a Matlab routine.The output spectrum from Synopsys simulations, shown in Figure 4.23.a., displays the 40 dB/decade NTF characteristic for a second order ΣΔ modulator. Figure 4.23.b. displays the output time domain waveform showing the two dimensional density (amplitude versus time) with a maximum following the baseband sinusoid

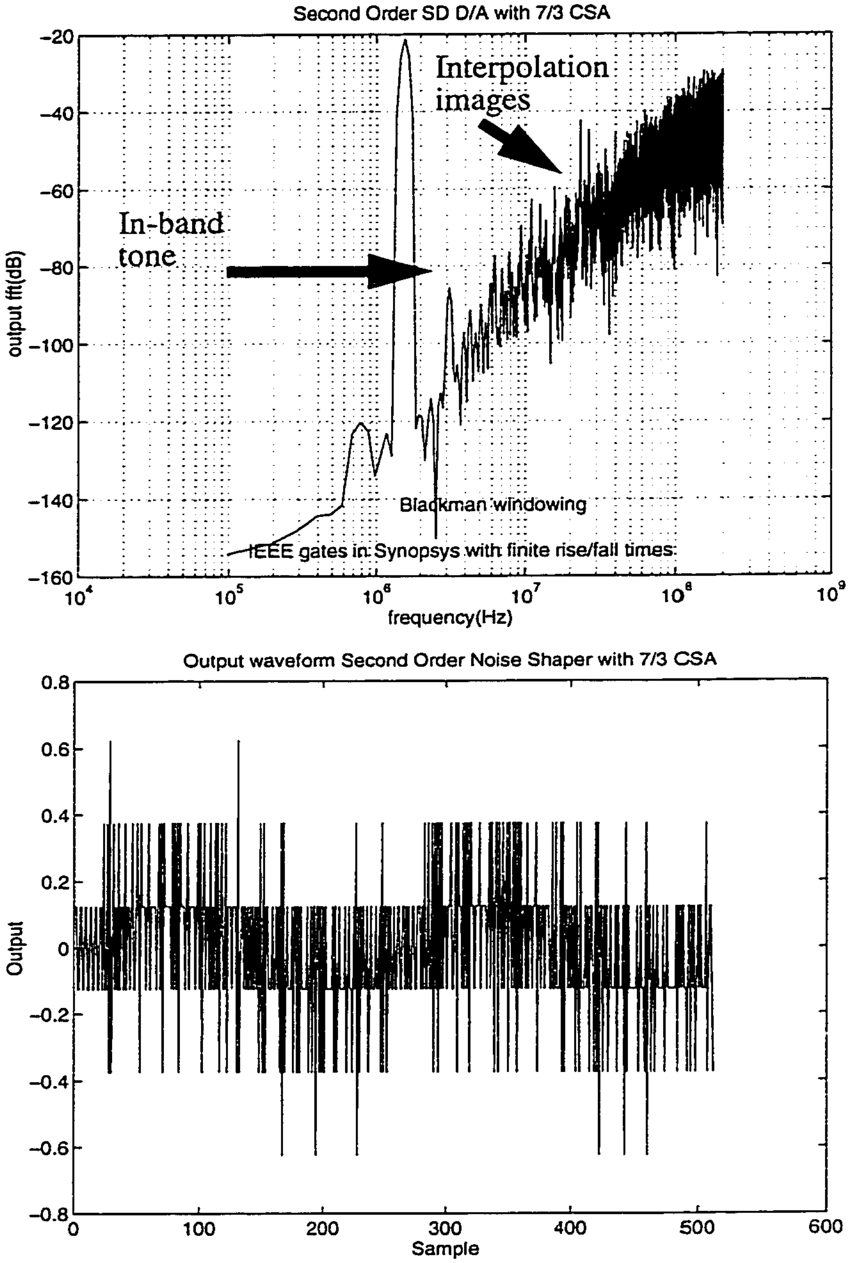


Fig.4.23.(a) Second order CS ΣΔ modulator output spectrum (Synopsys implementation); (b) output waveform.

In Section 3.2. was discussed the 1/2 LSB offset inherent to digital quantization in ΣΔ modulators. This offset was compensated for in Matlab when the output spectrum shown in Figure 4.27 was computed. The three bit output samples were replaced by four bit samples, with LSB equal to ONE.

From the Synopsys simulation results, for Fs = 250 MHz, the peak SNDR in Fs/32 = 7.81 MHz (OSR = 16) is 77.67 dB, corresponding to 13 bits of resolution.

4.8. Speed advantage over state of art

The state of the art implementation against which the proposed CS ΣΔ modulator architecture is to be compared in terms of speed was presented in Figure 3.18. The basic building block used in adders is the four bit CLA, shown in Figure 4.28. in a technology which allows maximum three input gates. The reason is that 0.5μ CMOS library used to implement the proposed ΣΔ DAC has maximum three input gates. The maximum flip-flop toggle frequency is 1 GHz, the delay through an inverter is 150 ps and through a NAND gate is 250 ps. The critical path through the clipper (Figure 3.16.) is considered to be six gates (optimistic estimate).

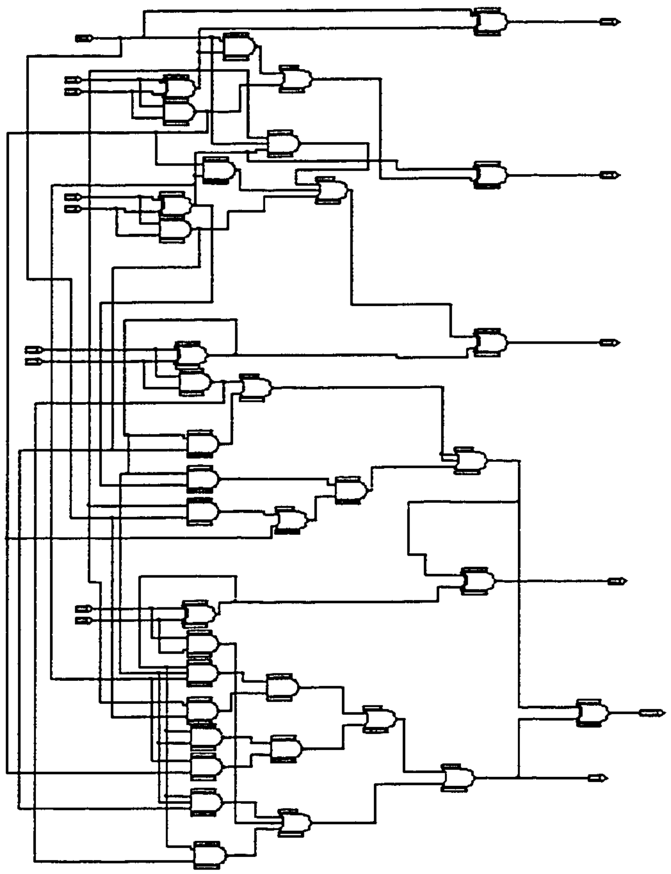


Fig.4.24. 4 bit CLA used to implement both the state of art and the second order CS ΣΔ modulator

The state of the art implementation of a digital second order ΣΔ modulator was discussed in Section 3.5. The critical path in the circuit is highlighted in Figure 3.18.

```
maximum_path = CPA_adder_3_bit + clipper +

                CPA_adder_19_bit    + dff +

                set_up_time_dff + inv =

                = 37 gates + dff = 8.17 ns

maximum speed = 122 MHz
```

The proposed second order CS ΣΔ modulator architecture has the critical path highlighted in Figure 4.17.

```
maximum_path = 3/2_carry_save_adder+
                + 7/3_carry_save_adder +
               + CPA_adder_6_bit + inv+ dff +
               +  set_up_time_dff + inv      =
               = 18 gates + dff = 4.49 ns

maximum speed = 222 MHz
```

The speed improvement is:

```
 speed improvement= ((8.17 - 4.49)/4.49) * 100 = 82%
```

In the above calculations , the multiplexer required for testing was not taken into consideration. The critical path through a multiplexer is three gate delays, which will impact severely on the achievable speed (15% reduction in speed). This was the reason testing was not included in the silicon implementation. An eventual industrial implementation will have to include such a feature.

Changing the number of bits in the second stage does not affect the speed-up factor, since computations are done in parallel to data flow and approximate quantization uses only the most significant CS columns. On the contrary, it will introduce unnecessary truncation noise prior to quantization. It should be noted, however, that word width truncation prior to quantization is a valid method to speed-up CPA-based ΣΔ modulators.

The speed improvement estimate has to be taken with caution at this stage because:

a) The state of the art implementation is a one bit output circuit and the proposed CS ΣΔ modulator architecture is a three bit circuit. A corresponding three bit implementation of the state of the art will be slower (will require a six bit adder instead of a three bit adder in the critical path in the inner loop in Figure 3.18.). The delay through the clipper was considered to be an optimistic six gate delays. From this point of view, the speed improvement figure is pessimistic.

b) The delay through wires was not taken into consideration. For a deep submicron technology, such as 0.5μ CMOS, these delays are significant. The physical layout has a large influence on the actual speed performance of the final circuit. From this second point of view, the speed improvement factor computed above is optimistic.

Even with these correction factors, the speed improvement provided by the use of CS arithmetic in ΣΔ modulators is substantial. The experimental results shown in Chapter 6 confirm up to a certain limit (there was an I/O pad speed limitation) the above theoretical results

An implementation of a digital ΣΔ modulator using signed redundant binary representation of numbers (SBNR) and approximate quantization was presented in [Sewe94]. Our calculations show that for the chosen target technology, the critical path through a three input SBNR adder is 15 gates, compared with 6 gates for a 7/3 CSA adder. For a one bit output, the delay through a 4 bit approximate CSA quantizer is 8 gates (not including approximate overflow), compared to 11 gates for a SBNR approximate quantizer (not including approximate overflow).

4.9. Summary

We have proposed a method for making fast CSA compatible with ΣΔ. Approximate CS quantization and clipping are shown to be faster than the exact computation and have little effect on the modulator performance. A speed increase of roughly a factor of two over state of the art is predicted. Speed increase translates into resolution bandwidth increase.

CHAPTER 6 *Implementations and experimental results*

The $\Sigma\Delta$ DAC was implemented both in a Xilinx 4010E FPGA and a 0.5μ CMOS technology provided through MOSIS.

6.1. Xilinx 4010E implementation

The main purpose of this implementation was to demonstrate that the CS $\Sigma\Delta$ modulator inside the $\Sigma\Delta$ DAC is stable in the long run. The digital part of the $\Sigma\Delta$ DAC (i.e. the CS $\Sigma\Delta$ modulator and the bit shuffling network) was modeled in VHDL and synthesized in the Synopsys environment. In parallel, the schematic was captured and synthesized in the Viewlogic environment. The Synopsys synthesis is more flexible, allowing the user to do synthesis under timing and area constraints, generate reports, etc.

Figure 6.1. shows the test set-up for the digital part of the $\Sigma\Delta$ DAC. Input data, the digitized samples of the baseband sinewave (16 samples, 16 bits wide each), is generated using the 200 MB pattern generator capability of the HP 16500C logic analyzer system. The FPGA is configured by downloading the synthesized.bit file of the circuit from an IBM PC. The logic analyzer clock and the $\Sigma\Delta$ and bit shuffler clock are generated synchronously using an HP 80000 data generator system. The circuit outputs are connected to the 2 MB data acquisition capability of the logic analyzer system. The digital output samples are written in ASCII format to an IBM PC through the HPIB interface of the logic analyzer system.

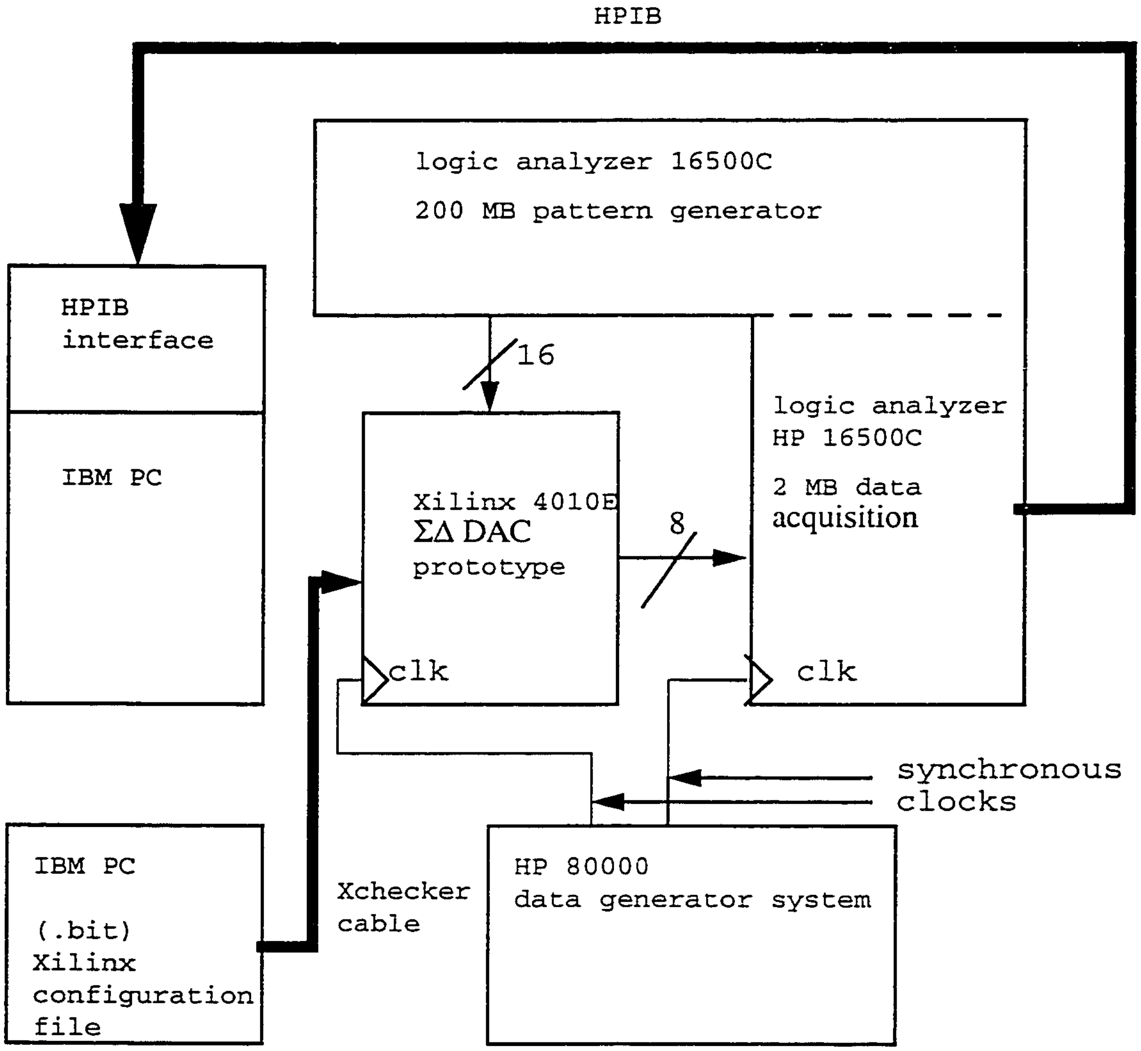


Fig.6.1. Test set-up for the Xilinx 4010E implementation of the digital part of the ΣΔ DAC

The ASCII file was decoded using a simple C routine. Current source mismatch was modeled in Matlab by using non-unity coefficients when computing the output samples from the m out of eight output of the digital part. The VHDL code describing the CS ΣΔ modulator and the bit shuffling network from which the FPGA configuration file (.bit file) was computed is given in Appendix A.

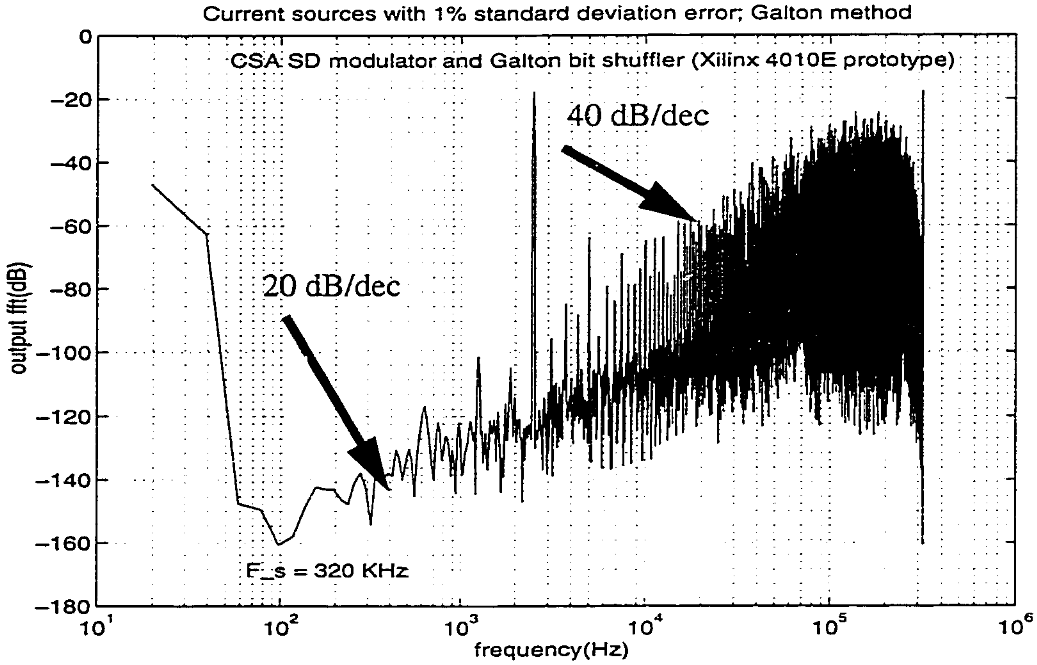


Fig.6.2. Output spectrum from digital part of ΣΔ DAC, Xilinx 4010E implementation, (1% current source standard deviation error)

The analog stage with mismatches was modeled in Matlab. The spectrum computed for a 1% standard deviation error in current source values is shown in Figure 6.2., with the Galton effect clearly evident. The Xilinx prototype clocks correctly up to 25 MHz.

The test was run two days in a row: SNR was computed at the begining and at the end and no difference was noticed.

6.2. 0.5μ CMOS implementation

The main objective of the 0.5μ CMOS implementation was to demonstrate the ΣΔ DAC speed improvement predicted by theory. The silicon level design involved dealing with mixed signal issues such as substrate coupling, special analog layout techniques to minimize current source mismatch, on-chip buffering, clock distribution, clock skew control. For speed reasons, a manual layout was chosen over Synopsys synthesis and automatic place and route flow (APR) in the Cadence Cell Ensemble environment. Choosing the second implementation method would have had defeated the very purpose of the proposed speed-up method. The delay around the critical path is a few times (four, five times) larger when choosing synthesis. and APR. Also, delays in on-chip wiring are hard to control tightly in an APR flow. This last aspect is of crucial importance, since for deep submicron processes, such as 0.5μ CMOS, delays in wiring can be important. Another way to implement the circuit is to do a manual Verilog/VHDL gate level netlist for the circuit in the target technology, read it in Synopsys and then do an APR.

In a CMOS process, drain current asymmetry occurs when drain-source direction of a transistor is interchanged. Figure 6.7. helps to understand why this happens and what can be done to avoid this situation.

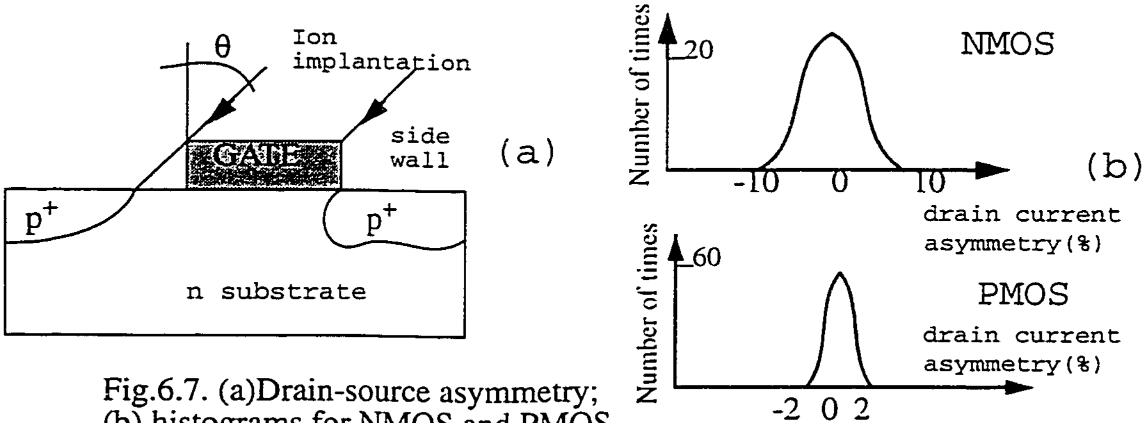


Fig.6.7. (a)Drain-source asymmetry; (b) histograms for NMOS and PMOS

When the source/drain regions are created, the gate shadowing effect determines an imbalance between the electrical properties of the two regions. The effect is modeled by a parasitic resistance in series with the source of the MOS transistor, which affects the value of the drain current. Figure 6.7.b. shows the drain current asymmetry for NMOS and PMOS transistors respectively. The standard deviation error in the case of using PMOS transistors is much less than in the other case. This is the reason why the current sources in the analog reconstruction stage of the proposed ΣΔ DAC use PMOS transistors.

To regulate drain current asymmetry, the solution proposed in [Naka91] and shown in Figure 6.8.b. was employed in the layout of the ΣΔ DAC. A transistor is divided into two mirrored subtransistors connected in parallel. An output of one current source is given by the addition of drain currents of the two mirrored subtransistors. In this layout, neither mirroring nor 180 degrees rotation can change the output current.

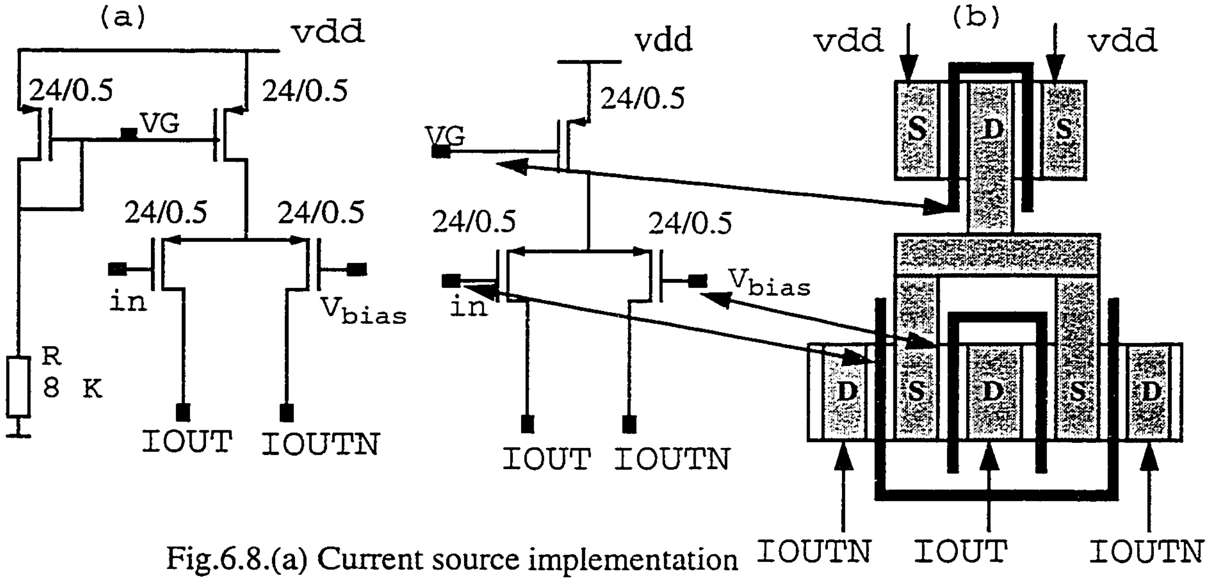


Fig.6.8.(a) Current source implementation (b) symmetrical layout technique

Another important factor to be taken into consideration when implementing mixed signal integrated circuits is reduction of coupling through the power supplies. A noisy digital power supply can disturb the analog on-chip stage using the same supply. Figure 6.10. gives the picture of the chip layout, illustrating the techniques used. One technique is to keep separate on-chip power supplies for the digital part of the chip (the ΣΔ modulator and the bit shuffler), the analog reconstruction stage and the clock circuitry. The digital, analog and clock supplies have a common off-chip ground, on the PCB. Close attention is to be paid both to on-chip and off-chip power supply coupling. Poor PCB design can seriously affect the measured resolution performance of the ΣΔ DAC. Also, the n-wells of analog current sources are separated from those of the digital circuits [Ing97].

Even an optimally decoupled analog circuit can be disturbed by substrate noise injected in some other place on chip. Guard rings can limit this effect when used correctly. Figure 6.9. shows the effect of using guard rings attached to VSS and VDD respectively. They provide low impedance return paths for the substrate currents by acting as traps for the injected minority carriers [West93].

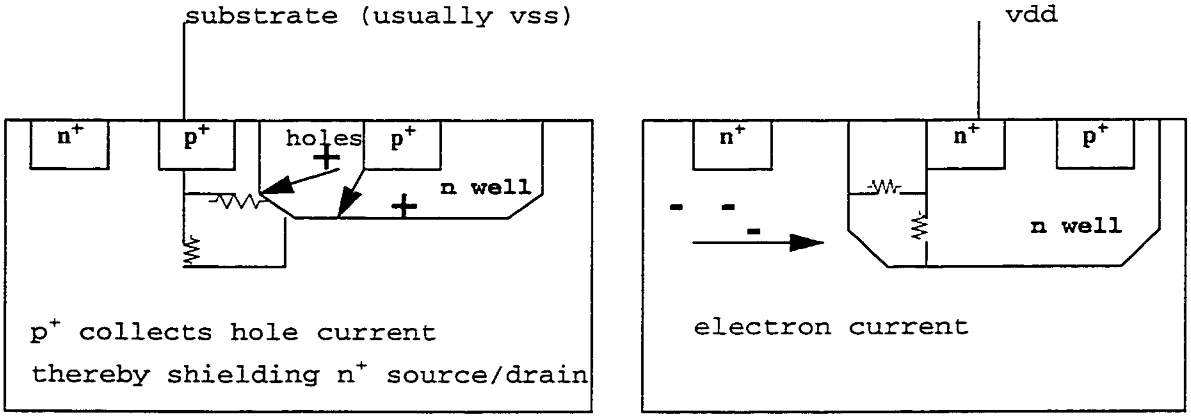


Fig. 6.9. Use of guard rings
to reduce substrate coupling

Attention must be paid to proper guard ring biasing. For example, biasing the substrate contact of the respective guard ring (Figure 4.9.a.) with the digital ground will inject extra noise into the substrate [Ing97]. The proposed circuit layout uses two guard rings around the digital part: a p^+ guard ring tied to the substrate pad (SUB) and a n^+ guard ring, in an independent n well, tied to the digital VDD (see Figure 6.10.).

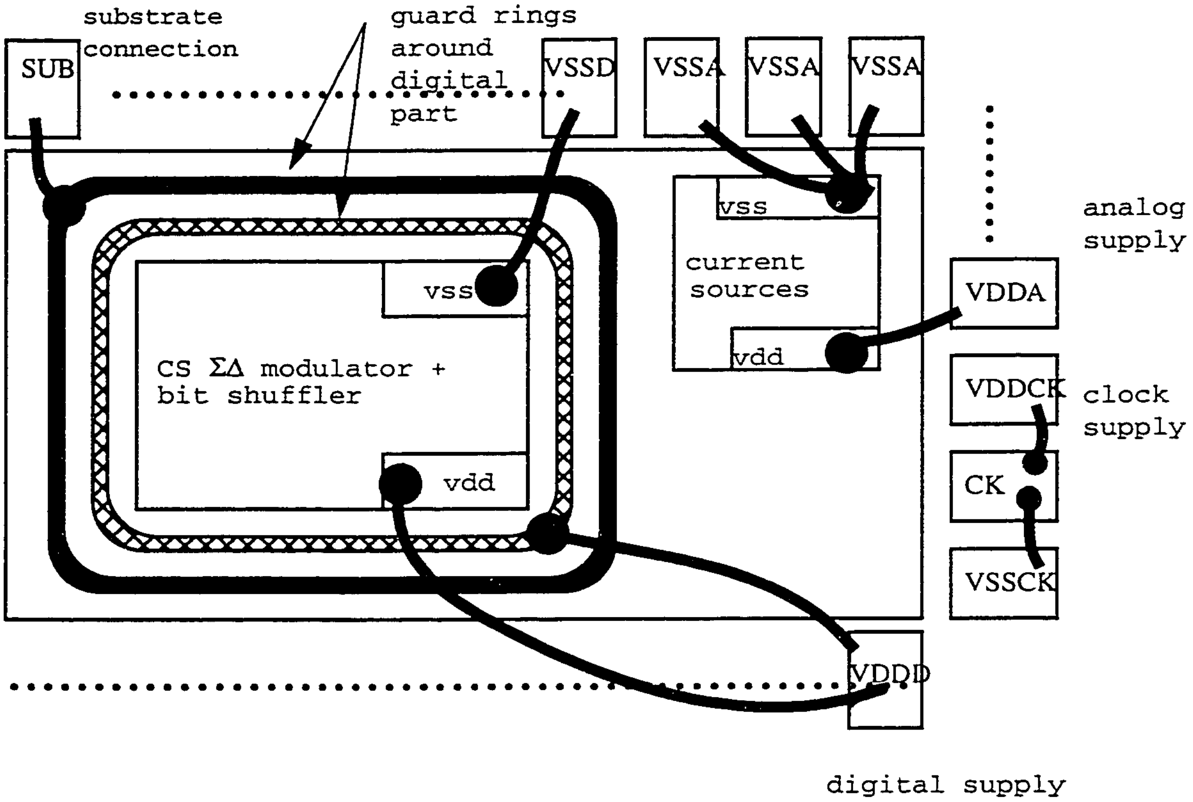


Fig.6.10. Chip layout sketch illustrating the use of guard rings and power supply decoupling

Figure 6.11. shows the block diagram digital section (CS ΣΔ modulator and bit shuffler) layout. The clock tree has nine branches, driving roughly the same load in order to minimize the clock skew. The flip-flop sections are driven by 4X buffers, driven in turn by an on-chip 4X clock buffer.

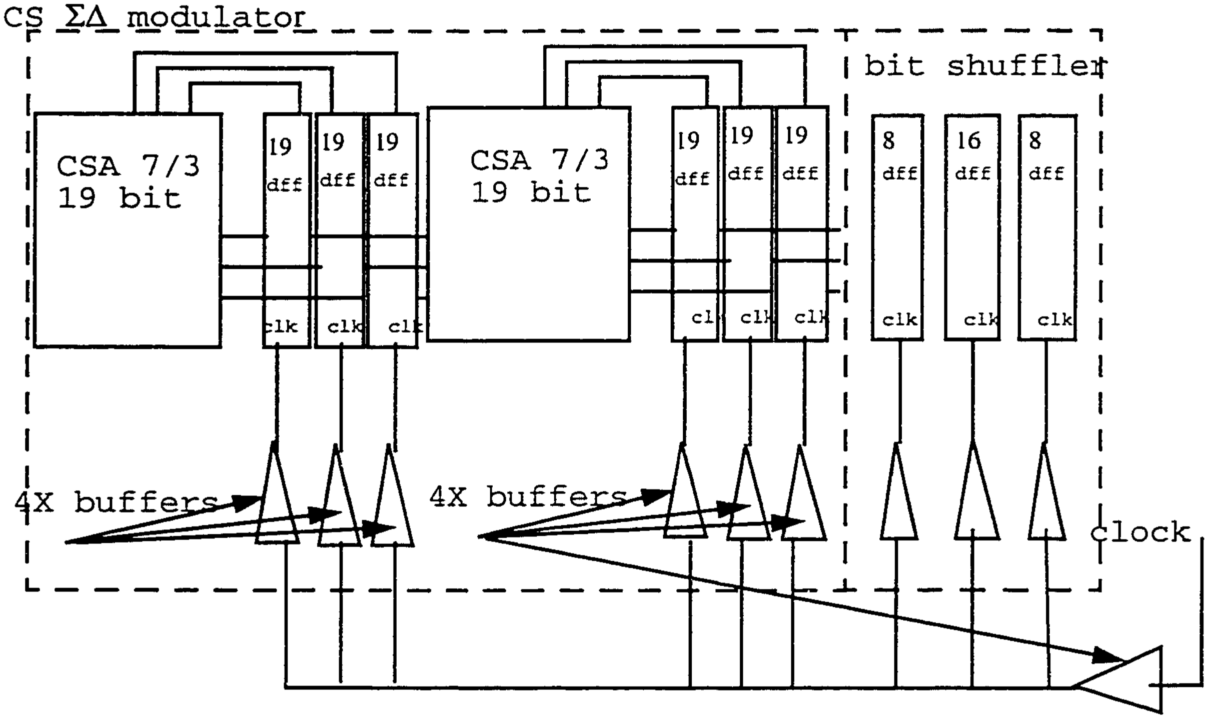


Fig.6.11.Chip layout sketch
illustrating clock tree and buffering

Total chip area, including pads and scribe line, is 7.063 mm^2, of which 2.56 mm^2 is core area. The CS ΣΔ modulator takes an area of 1.63 mm^2 and has an equivalent of 2854 gates.

The 0.5μ CMOS chip test set-up is sketched in Fig.6.12. The second order CS ΣΔ modulator's three bit output can be fed to a logic analyzer system and digital samples written using a HPIB interface to a PC. The ΣΔ DAC is clocked using an HP 8131A signal generator. The chip has two analog outputs, corresponding to IOUT and IOUTN (see Figure 6.8.). The analog output can be taken single ended or differentially. The analog output is connected to the 50 ohm input of a HP8591E spectrum analyzer. The two synchronous clocks are generated using the same HP 80000 data generator system in the same way as in the Xilinx prototype test set-up. The stability test was done in the same manner as for the Xilinx prototype.

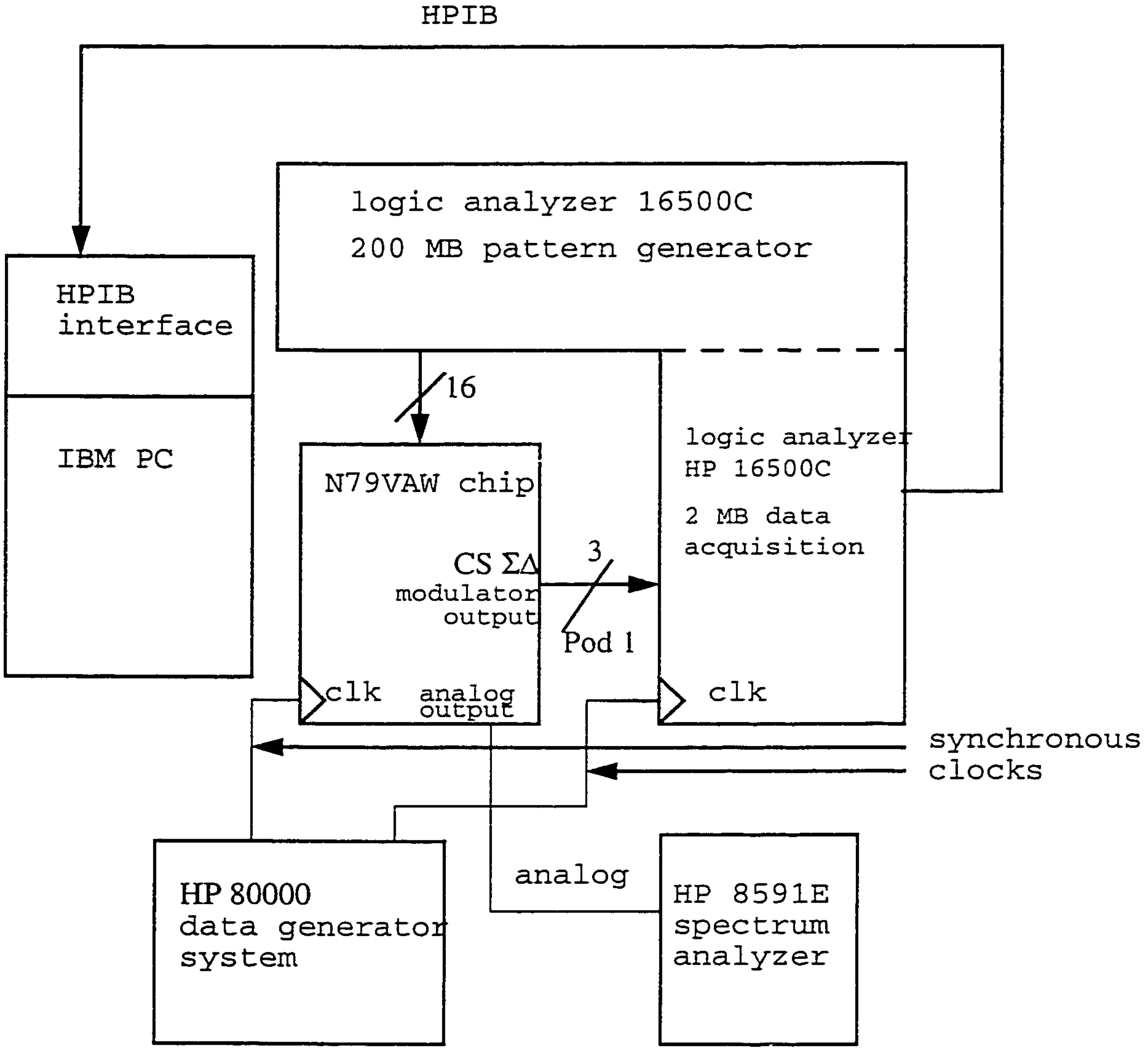


Fig.6.12. Test set-up for the N79VAW chip

The measured 3 bit digital output spectrum is shown in Figure 6.13. The sampling frequency is $F_S = 5.12$ MHz, the OSR=F_S/ (2*BW)= 68.27 and the baseband signal has a frequency of 2.5 KHz. The spectral images seen are due to the fact that the baseband sinusoid is not interpolated. The baseband signal is modulated then by the incoming sampling frequency, F_{s1}= 40 KHz from the 200 MB pattern generator. The maximum measured SNR in [0 37.5 KHz] bandwidth is 92 dB, corresponding to 15 bits of resolution (SNR measured in the same way as in Chapter 4). Maximum SNR for a given OSR for a second order ΣΔ modulator is given by the following formula (see Chapter 3):

$$SNR(dB) = 15 \cdot \log_2(OSR) - 23 + 12 = 80.39 \qquad (6.1)$$

The third term in the above equation is due to the use of a three bit quantizer. Every added bit to the quantizer increases the resolution of the modulator by 6 dB (see Section 3.5.). The measurement results and theoretical formula give close results only if OSR has double value in (6.1). The ideal peak SNR value is in this case 95.39 dB. The DC component was eliminated in Matlab by adding 1/2 LSB to each sample (see also Section 4.7).

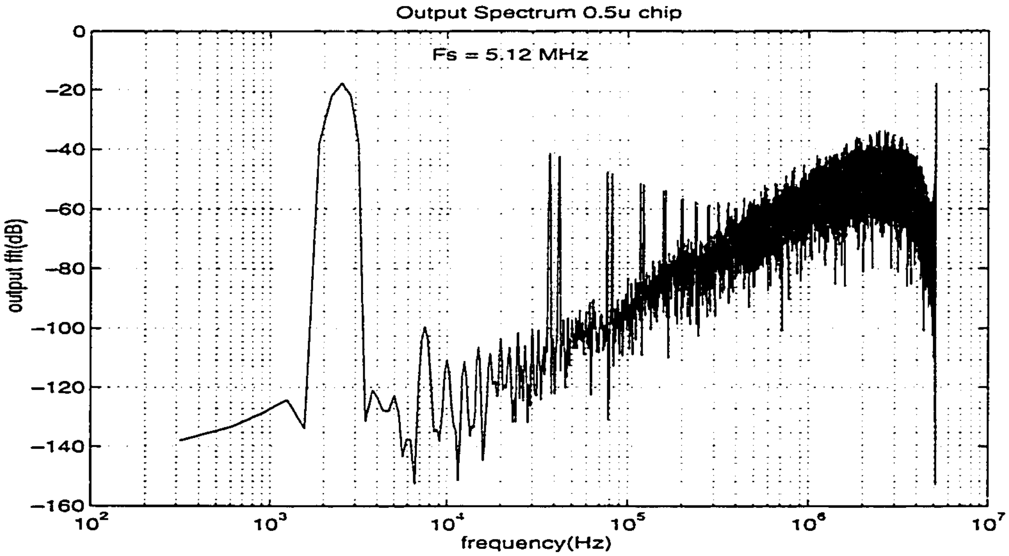


Fig.6.13. Output spectrum from CS ΣΔ modulator output, 0.5μ CMS chip (F_s = 5.12 MHz)

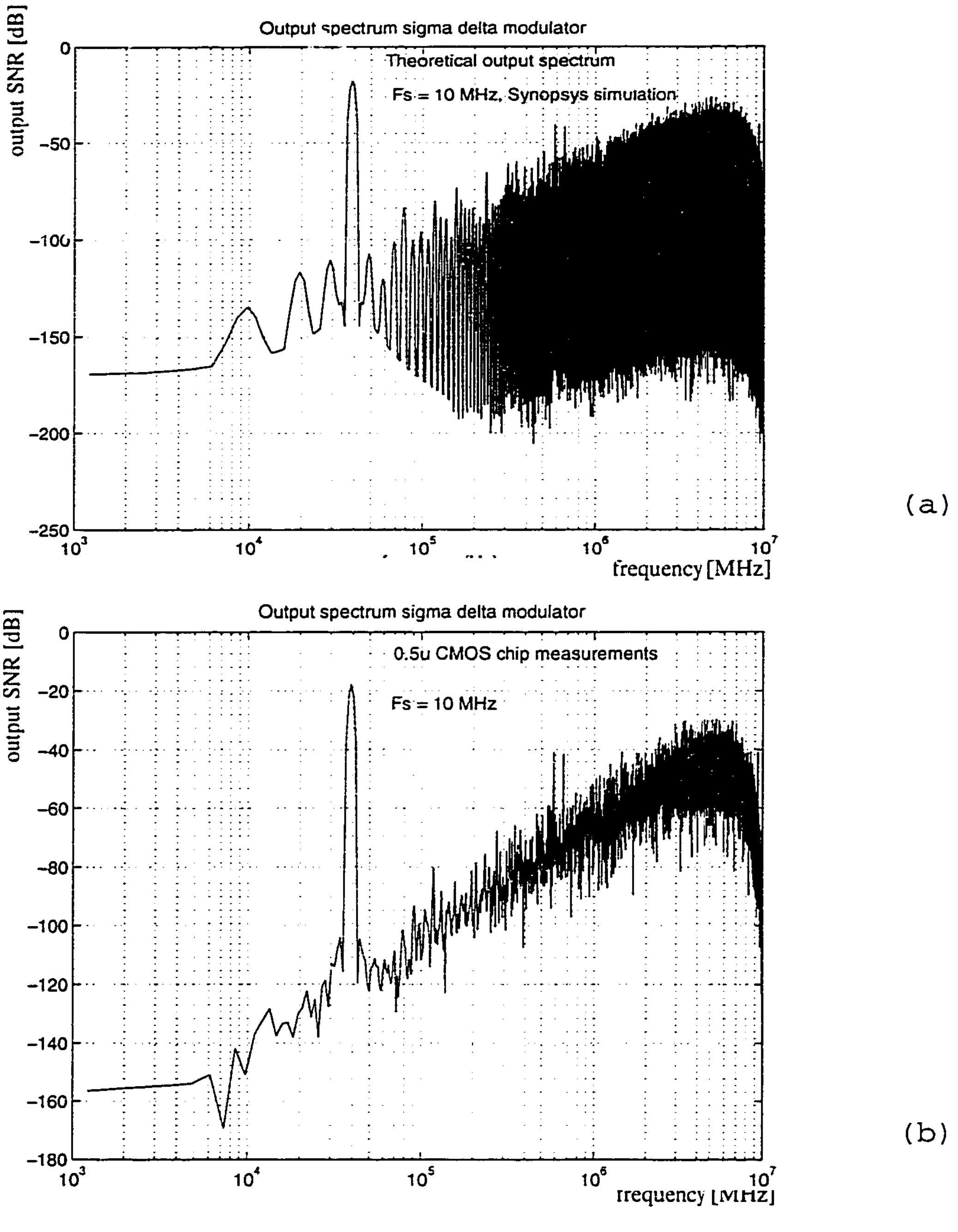


Fig.6.14. Comparison between simulated and measured CS ΣΔ modulator output spectrums: a) Synopsys simulation b) 0.5μ chip measurement

Figure 6.14. above shows a comparison between the theoretical and measured second order CS ΣΔ modulator output spectrums. For a sampling frequency of 10 MHz and the peak SNR in $F_S/32$ (OSR = 16) bandwidth (0 to 312.5 KHz) is 77.67 dB for the Synopsys simulation and 76.34 dB measured from the chip.

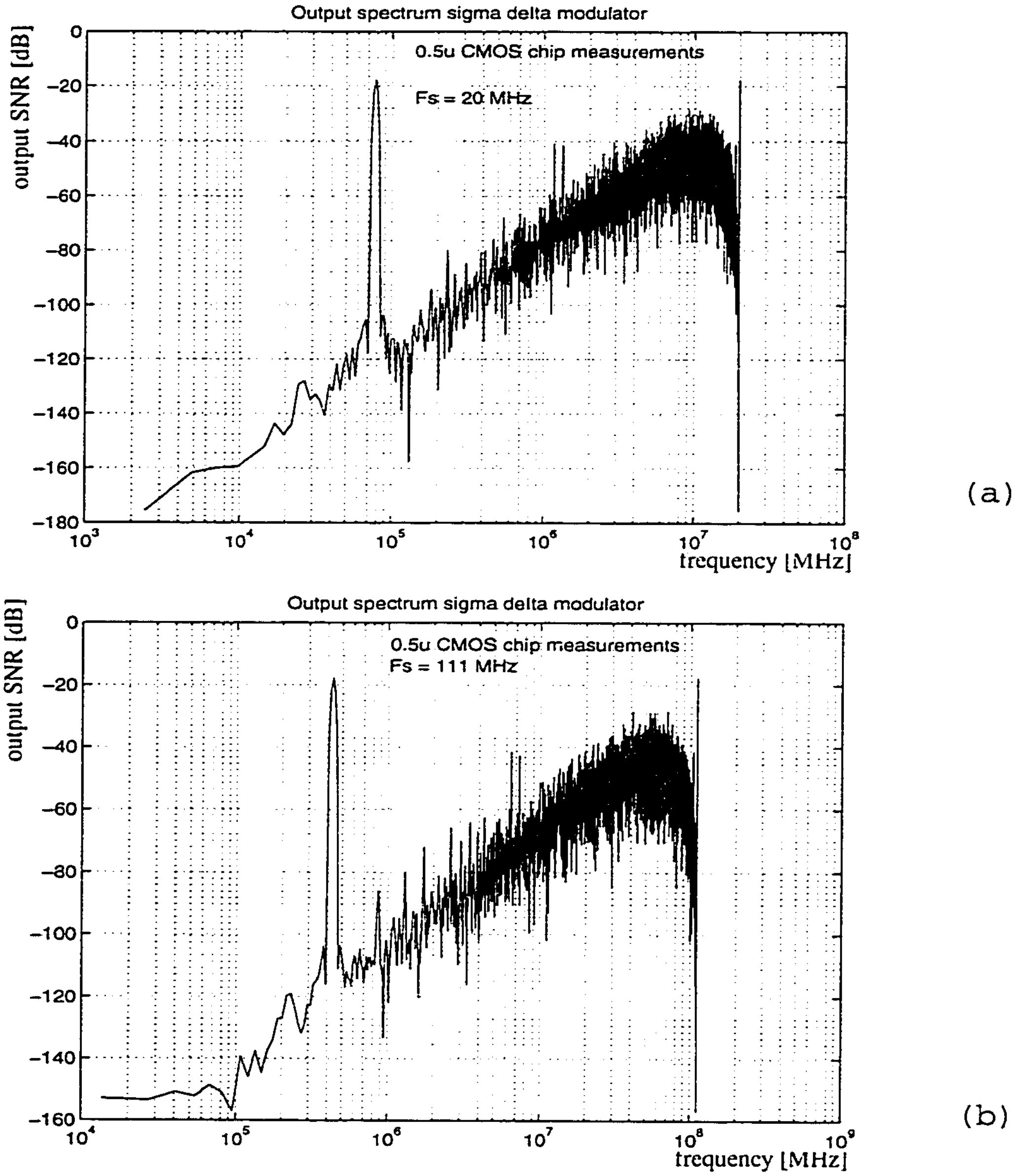


Fig.6.15. Measured CS ΣΔ modulator output spectrums
(a) F_S = 20 MHz; (b) F_S = 111 MHz.

Figure 6.15. shows that the measured CS ΣΔ modulator output spectrum is practically the same at 20 MHz and 111 MHz. The two corresponding maximum SNRs for the same relative bandwidth (Fs/32) are 76.17 dB and 75.84 dB respectively.

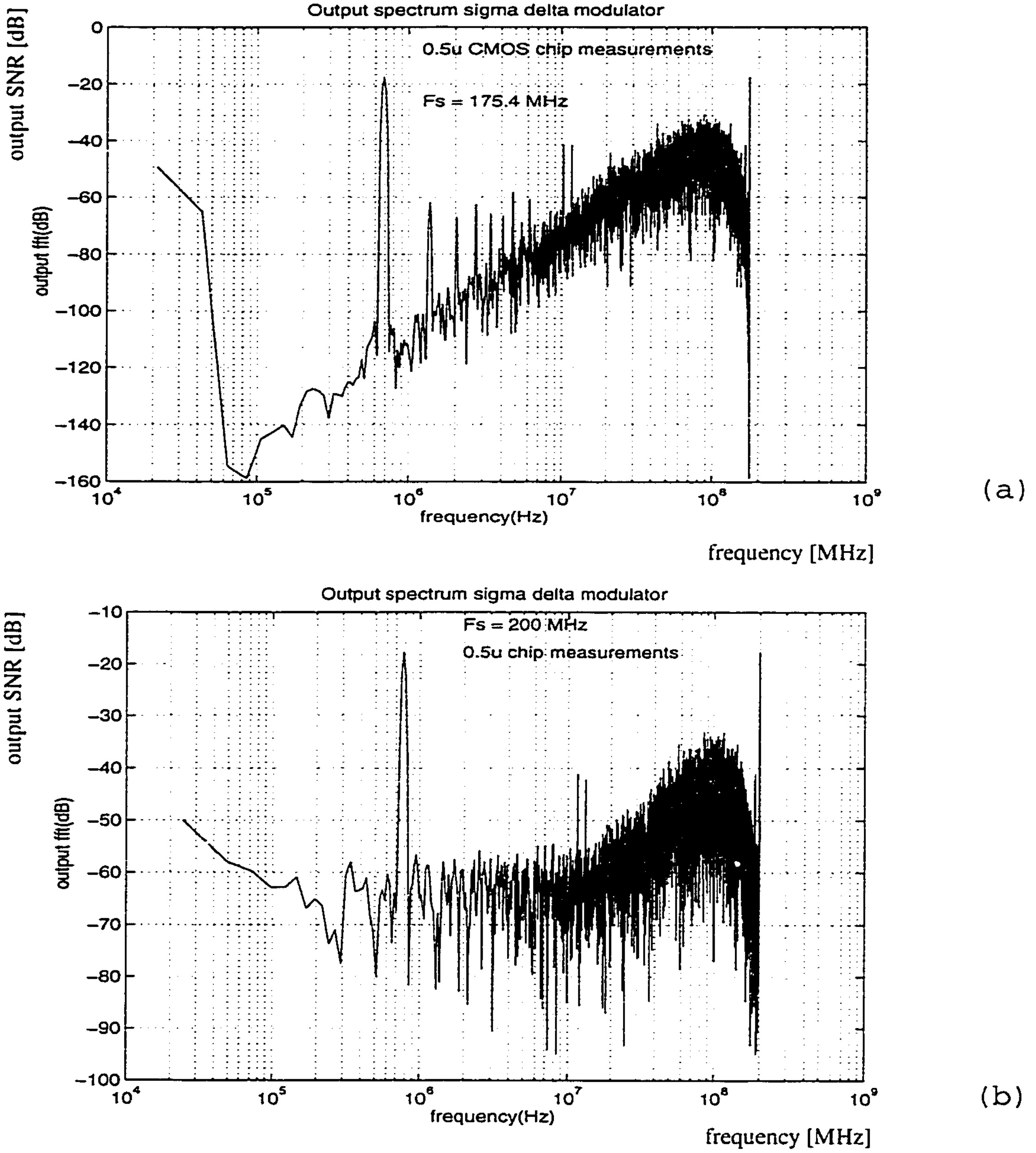
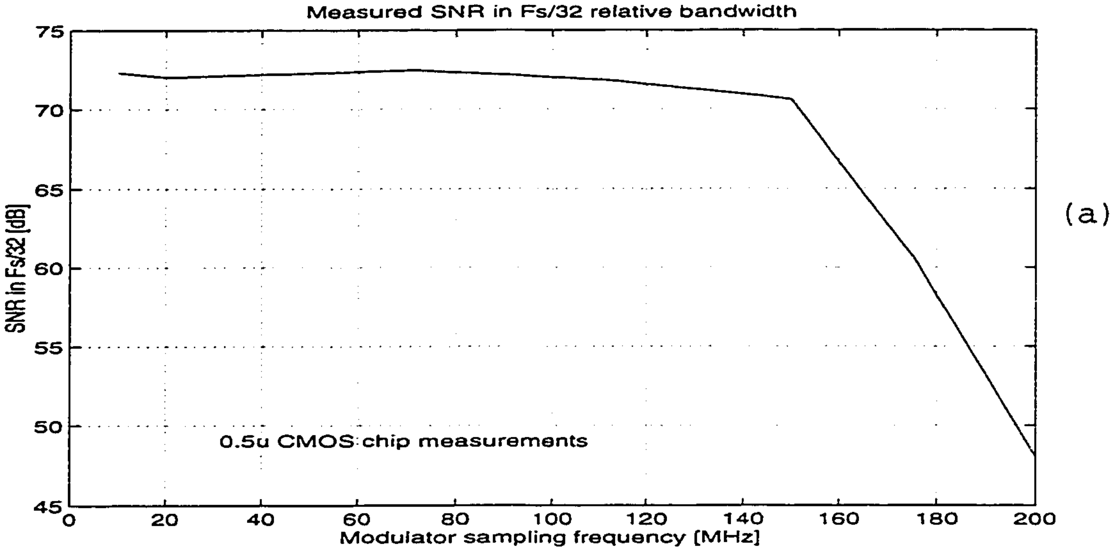


Fig.6.16. Measured CS ΣΔ modulator output spectrums (a) F_S = 175.4 MHz; (b) F_S = 200 MHz.

Figure 6.17. below shows the variation of the peak SNR of the CS ΣΔ modulator in the same relative bandwidth (Fs/32) as a function of sampling frequency.



frequency [MHz]	10	20	71.4	111	150	175.4	200
max SNR in Fs/32 relative bandwidth [dB]	76.34	76.17	76.02	75.84	73.53	64.2	58.13

(b)

Fig.6.17. Maximum CS ΣΔ modulator output SNR in Fs/32 relative bandwidth as a function of sampling frequency: (a) graph; (b) table.

It can be seen the above Figure that the maximum measured CS ΣΔ modulator output SNR in a $F_S/32$ relative bandwidth has a smooth roll-off after 145 MHz sampling frequency. The explanation is that the output pads and the clock pad (bidirectional pads) are specified to work properly only up to 120 MHz. This factor makes it difficult to measure the real maximum sampling frequency of the CS ΣΔ modulator. The SNR in Fs/32 bandwidth

(OSR = 16) shown in Figure 6.17. is relatively constant up to 145 MHz, a thing to be expected since the measured circuit is digital.

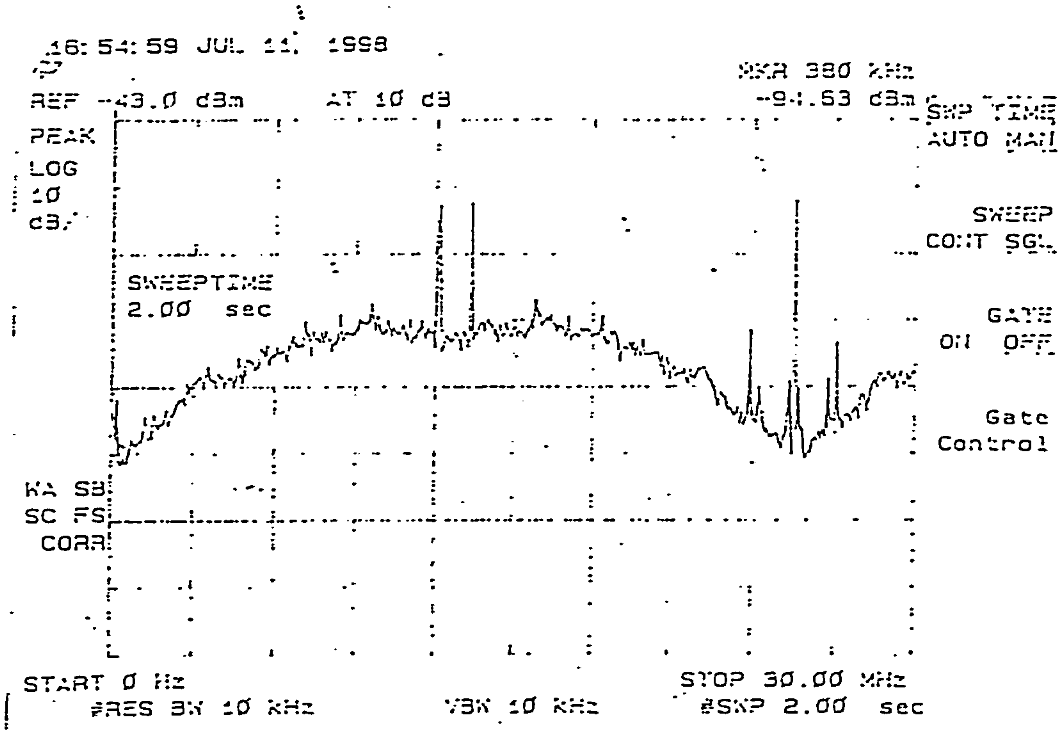


Fig. 6.18. Analog output spectrum at Fs = 25 MHz.

Figure 6.18. above shows the analog output spectrum of the 0.5µ CMOS chip at 25 MHz sampling frequency. The spectrum appears noise shaped, with significant in-band harmonics and clock feedthrough at 25MHz.

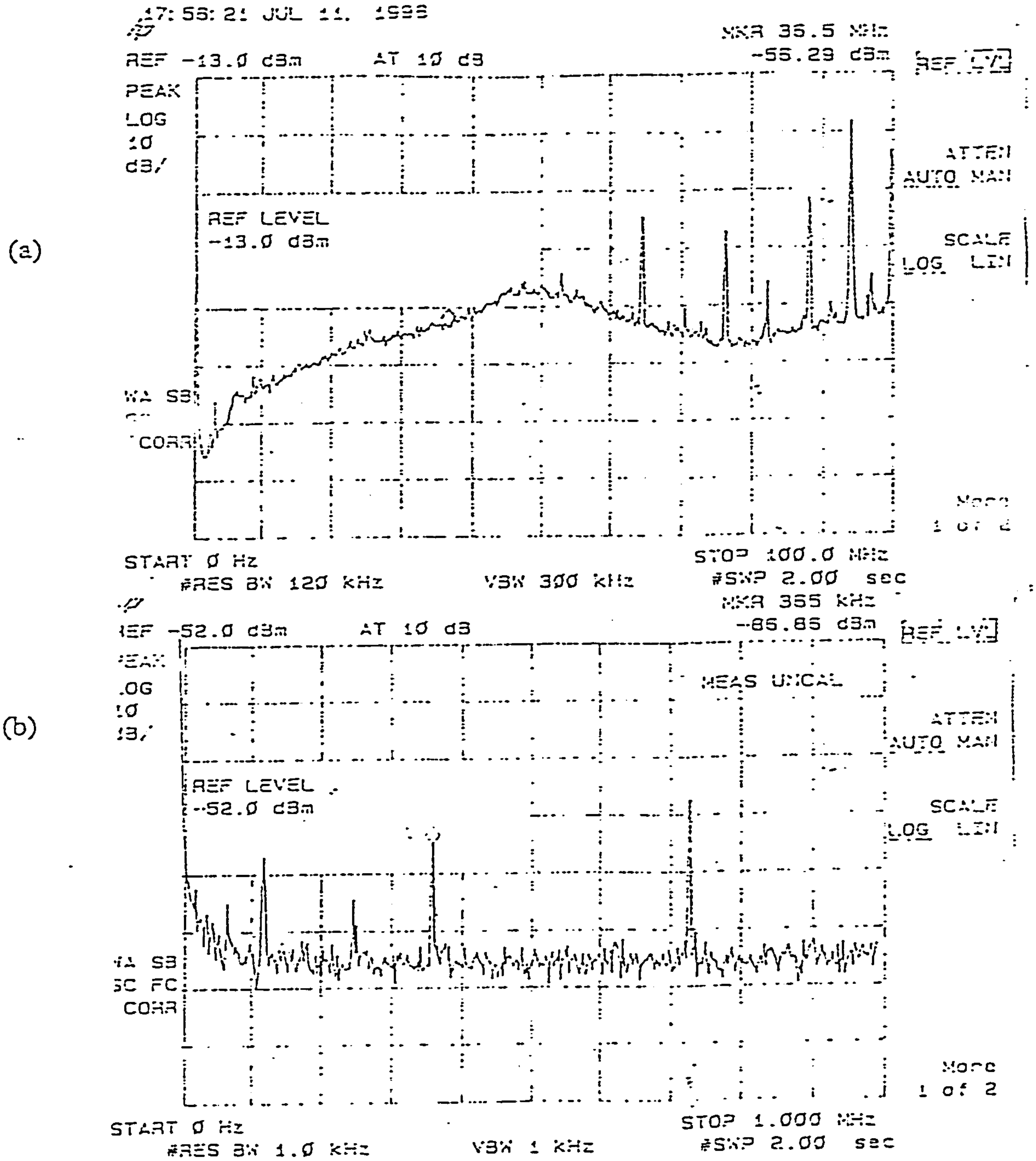
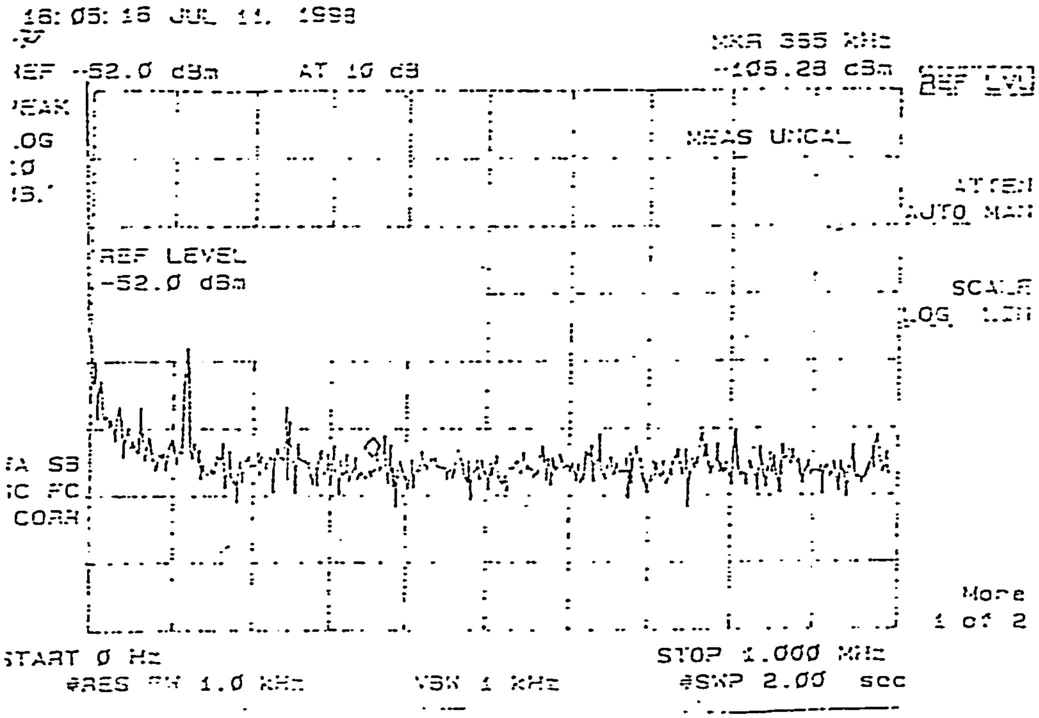


Fig. 6.19. Output spectrum at Fs = 93.44 MHz
(a) overall view; (b) in-band



·Fig.6.20. In-band spectrum at Fs = 93.44 MHz
with DAC input turned off.

Figure 6.19. shows the analog output spectrum of the ΣΔ DAC at Fs = 93.44 MHz. The spectrum displays a rough noise shaping, it is not symmetrical, and has a lot of out of band harmonics. There is a signal component of -86.88 dBc/Hz at the expected frequency of 365 KHz (256 times less than the sampling frequency (see Figure 6.19.b.). The spectral line corresponding to the baseband signal disappears from the in-band spectrum when the 200 MB pattern generator is turned off, (see Figure 6.20. above). The 365 KHz component amplitude does not change with the amplitude of the incoming baseband digital sinusoid

The analog measurements demonstrate a rough noise shaping at frequencies up to 250 MHz (the frequency limit of the testing equipment). The in-band spectrum demonstrates small level signal components at the baseband frequency and its harmonics. There is no relation between the input code and the amplitude of the component at the baseband frequency.

There was a pad problem in the original circuit: a digital I/O pad was used as an analog pad, which overrode the analog output. Microsurgery was used in a repair attempt, which seems to have slightly improved the results, but the SNR is still very poor.

Possible reasons for the poor behavior of the analog output can be:

a) The analog output stage is damaged or has a fabrication error. The output has a DC component of 2.35 V for 1 KOhm load and practically no AC component. The most likely supposition is that one or more of the differential pairs have drain to source short-circuited transistors (see Figure 6.21.). For this type of defect, all the current of T2 flows through T3 and its load resistor.

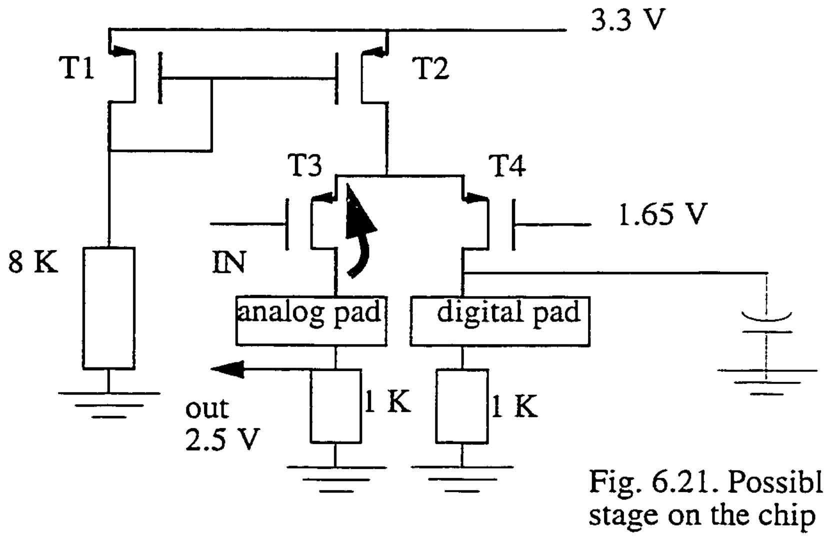


Fig. 6.21. Possible short circuit in the analog stage on the chip

Since there is no current switching, the output stays at a constant DC voltage, with almost no AC component.

b) If the bit shuffling network had any circuit or fabrication error (it has a potential glitch problem, explained in Chapter 5), noise shaping at the output would be difficult to detect, so the problem cannot be there. This possibility it is most likely to be discarded.

It is difficult to make any real conclusions about the analog performance of the chip.

In conclusion, the only quantitative results are the digital measurement results, limited also by the I/O pad performance. The 145 MHz clock for full digital functionality is close to the value predicted by theory, but it is apparent from the measurement results at 175.4 KHz (correct second order ΣΔ noise shaping, along with a DC component and baseband harmonics) that the core of the chip can function properly at higher speeds. A difficult measurement to be done is to probe internal nodes on the die directly.

Future chip iterations have to ensure proper analog stage functionality. The use of ECL pads is a necessity, since the pad speed limits the performance of the present chip, along with on-chip clock generation. Off-chip synchronization problems between the two clocks have to be avoided by using on-chip clock generation. Chip thermal protection is also to be addressed in a higher performance technology in order to achieve high speed functionality for reliable amounts of time.

6.3.Summary

This chapter presented the experimental results which support the theory shown in Chapter 4. The Xilinx 4010E implementation was a proof of concept. The 0.5μ CMOS part runs correctly up to 145 MHz, with measurable performance limited by pads.

CHAPTER 7 Conclusions and future work

7.1. Conclusions

High resolution D/A conversion for a wide range of frequencies is a major bottleneck for modern communication systems. The work presented in this thesis demonstrated (in the limits allowed by the technology) that it is possible to increase the performance of state of the art ΣΔ DACs, by using an improved digital ΣΔ modulator. The modulator is based on a simple extension of the concept of two's complement representation of numbers. The main accomplishments are outlined below.

1) A digital ΣΔ modulator architecture using CS arithmetic was proposed. The theoretical considerations were backed by extensive simulations at the mathematical and circuit levels. A significant improvement in the maximum speed of the digital ΣΔ modulator is achieved by doing computations in parallel with data using a carry save structure. The increase in speed implies wider bandwidths for high resolution D/A conversion.

2) The ΣΔ DAC resolution is further increased by using a three bit quantizer in the digital ΣΔ modulator. The mismatch noise problem, an inherent drawback associated with the use of a multibit ΣΔ modulator, is minimized in the band of interest by using a hardware efficient ΣΔ-based bit shuffling network.

3) The ΣΔ DAC was implemented as a proof of concept using a Xilinx 4010E FPGA (the analog stage modeled in Matlab). The FPGA implementation demonstrated the stability of the second order CS ΣΔ modulator inside the ΣΔ DAC.

4) The ΣΔ DAC was implemented in 0.5μ CMOS. The CS ΣΔ modulator runs at 145 MHz. The maximum speed increase predicted by theory was difficult to detect due to I/O pad speed limitations. An important part of the design was laying out the chip by hand. High speed mixed signal issued were addressed, including: short metal runs in order to reduce RC delays, clock distribution, buffering, substrate coupling.

Some observations have to be made regarding filtering. The ΣΔ DAC does not have an interpolation stage or an analog smoothing filter. At this moment, the CS ΣΔ modulator is faster than the available CPA-based interpolators [Hoge81]. The speed bottleneck of the whole ΣΔ DAC is in this stage. Analog filtering was made easier by the use of a multibit ΣΔ modulator, which has reduced out-of-band quantization noise.

7.2. Future work

The algorithm used to speed-up the operation of ΣΔ DACs is directed for use in the cell basestation transmitter path for cellular telephony (see Figure 7.1.).

Two lowpass digital ΣΔ modulators can be used to implement a ΣΔ bandpass characteristic.The structure is based on the common SSB modulator architecture used in radio. The output of the overall ΣΔ DAC feeds directly the power amplifier of the radio transmitter path. The notch frequency is at half the sampling frequency of the lowpass digital ΣΔ modulators. A mixer and a power amplifier stage are cascaded with the ΣΔ stage. With the present implementation, the feasible notch frequency is:

145 MHz/2 = 72.5 MHz.

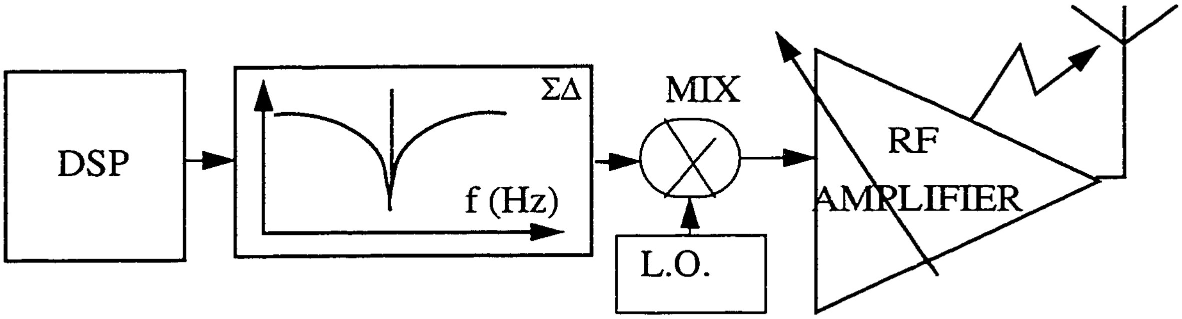


Fig.7.1.Cell basestation transmitter path using a bandpass ΣΔ DAC

There are several ways to increase the above notch frequency, including:

1) Improved algorithm (undetermined).

2) Use a faster technology (200% - 300% increase).

3) Better layout, less RC delays (20% - 30% increase).

4) Less CS ΣΔ modulator output bits (50% increase).

5) Custom cells (50% - 100% increase). The present implementation used standard cells.

Multiplication of the above factors gives an approximate increase in speed between 540% and 1170%. Considering a factor of 5 in speed increase, the achievable notch frequency is

72.5 MHz X 5 = 362.5 MHz.

No claim is to made regarding the accuracy of these figures. For example, the 2X - 3X speed improvement depends on the technology : a SiGe implementation, for example will be much faster. Figure 7.2 below shows speed-up figures for CMOS technologies , as predicted in the National Technology Roadmap for Semiconductors (http://www.semantech.org).

Year	1997	1999	2003
Technology [μ]	0.25	0.18	0.13
On-chip local clock, high performance ASIC [MHz]	750	1250	2100
On-chip, across chip clock, high performance ASIC [MHz]	300	500	700

Fig. 7.2. CMOS technology speed roadmap

A fair amount of effort in these future implementations has to be invested in the full custom layout design. Another field of application of this idea is direct digital synthesis.

The application of this method to IIR filters is under study.

Bibliography

[Cand86] James C. Candy and An-Ni Huynh, "Double Interpolation for Digital-to-Analog Conversion", IEEE Transactions on Communications, January 1986, pp.. 77-81.

[Cand92] James C. Candy and Gabor C. Themes, "Oversampling Methods for A/D and D/A Conversion", IEEE Press, 1992.

[Haus91] Max W. Hauser, "Principles of Oversampling A/D conversion", Journal of Audio Engineering Society, January/February 1991, pp.3-23.

[Su93] David K. Su and Bruce A. Wooley, "A CMOS Oversampling D/A Converter with a Current-Mode Semidigital Reconstruction Filter", IEEE Journal of Solid State Circuits, December 1993, pp.1224-1233.

[Gold96] Bar-Giona Goldberg, "Digital Techniques in Frequency Synthesis", McGraw Hill, 1996.

[Raza95] Behzad Razavi, "Principles of Data Conversion System Design", IEEE Press, 1995.

[Hoes94] David F. Hoeschelle, "Analog-to-Digital and Digital-to-Analog Conversion Techniques", John Wiley & Sons Inc., 1994.

[Plas94] Rudy van de Plassche, “Integrated Analog-to-Digital and Digital-to-Analog Converters”, Kluwer Academic Publishers, 1994.

[Sewe94] J.I. Sewell and David M. Hossack, “The Application of redundant Number Systems to Digital Sigma-Delta Modulators”, IEEE Press, 1994.

[Carl89] L. Richard Carley, “A Noise Shaping Coder Topology for 15+ Bit Converters”, IEEE Journal of Solid State Circuits, April 1989, pp.267-273.

[Ju95] Peicheng Ju & al., “A 22 KHz multibit Switched Capacitor Sigma Delta D/A Converter with 92 dB Dynamic Range”, IEEE Journal of Solid State Circuits, December 1995, pp. 1316-1324.

[Bair95] Rex T. Baird and Terry S. Fiez, “Linearity Enhancement of Multibit Sigma Delta A/D and D/A Converters using Data Weighted Averaging”, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, December 1995, pp. 753-762.

[Leun92] Bosco H. Leung and Sehat Sutarja, “Multibit Sigma Delta A/D Converter Incorporating A Novel Class of Dynamic Element Matching Techniques”, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, January 1992, pp. 35-51.

[Croc81] Ronal F. Crochiere and Lawrence R. Rabiner, “Interpolation and Decimation of Digital Signals - A Tutorial Review”, IEEE Transactions on Accoustics, Speech and Signal Processing, March 1981, pp. 417-488.

[[Hoge81] Eugene B. Hogenauer, “An Economical Class of Digital Filters for Decimation and Interpolation”, IEEE Transactions on Accoustics, Speech and Signal Processing, April 1981, pp. 155-162.

[Naka91] Yasuyuki Nakamura *et. al.*, "A 10-bit 70-Ms/S CMOS D/A Converter", IEEE Journal of Solid State Circuits, April 1991, pp. 637-642.

[Galt96] Ian Galton, "Noise Shaping D/A Converters for Sigma Delta Modulation", IEEE International Symposium on Circuits and Systems, 1996, pp. 441-444.

[Hig90] Richard J. Higgins, "Digital Signal Processing in VLSI", Analog Devices, 1990.

[West93] Neil H. E. Weste and Kamran Esraghian, "Principles of CMOS VLSI Design, A Systems Perspective", Second Edition, Addison-Wesley Publishing Company, 1993.

[Gray89] Robert M. Gray *et. al.*, "Quantization Noise in Single-Loop Sigma Delta Modulation with Sinusoidal Inputs", IEEE Transactions on Communications, September 1989, pp. 956-967.

[He90] Ning He *et al.*, "Double Loop Sigma Delta Modulation with DC Input", IEEE Transactions in Communications, April 1990, pp. 487-495.

[Cand81] James C. Candy and O'Connel J. Benjamin, "The Structure of Quantization Noise from Sigma Delta Modulation", IEEE Transactions on Communications, September 1981, pp.. 1316-1323.

[Snel95] W.M.Snelgrove, "Sigma Delta Course Notes", Carleton University, 1995.

[Kaya88] C. Kaya *et al.*, Polycide Metal Capacitors for high precision A/D Converters", EDM Digital Technology Papers, December 1988, pp. 782-785.

[Pelg89] M. J. M. Pelgrom *et. al.*, "Matching Properties of MOS Transistors", IEEE Journal of Solid State Circuits, October 1989, pp.. 1433-1439.

[Lu81] N. C. C. Lu, “Modeling and optimization of Monolithic Polychristaline Silicon Resistors”, IEEE Transactions on Electronic Devices, July 1981, pp. 818-830.

[Ing97] Mark Ingels and Michiel S. J. Steyaert, “Design Strategies and Decoupling Techniques for Reducing the Effects of Electrical Interference in Mixed-Mode IC’s”, IEEE Journal of Solid State Circuits, July 1997, pp. 1136 - 1141

Appendix A

ΣΔ DAC Matlab codes

```
function[]=saess2nd1(Fin,s,Z,U,M,V,N,K)
%3 bit extention K=4 *************************************************************
% noise shaper and shuffle network modeled in Matlab
%******************************************************************************
q=zeros(V+K-1);
caz=q(1,:);
w=zeros(V-1);
w1=w(1,:);
sm=caz;cm=caz;sm1=caz;sm2=caz;sm3=caz;
t=0:1./U*M:(Z*U*M-1)./U*M;
for i=1:Z*U
 ll(i)=round((2^(V-1))*s*sin(2*pi*(i-1)./U));
  if ll(i) < 0
    z1=dec2bin(abs(ll(i)),V);
    z=~z1;
    w2=[1 w1];
    m=sum1(z,w2,V);
  else
    m=dec2bin(ll(i),V);
  end;
 [smo,cmo,sm1o,sm2o,sm3o,coa(1+M*(i-1):M*i , : )]=...
scycle_2nd1(m,sm,cm,sm1,sm2,sm3,V,M,N,K);
 sm=smo;cm=cmo;sm1=sm1o;sm2=sm2o;sm3=sm3o;

end;
ll=length(coa(:,1));
%initial conditions
pc1(1)=0;pc2(1)=0;pc3(1)=0;pc4(1)=0;pc5(1)=0;pc6(1)=0;pc7(1)=0;
p1(1)=0;p2(1)=0;p3(1)=0;p4(1)=0;p5(1)=0;p6(1)=0;p7(1)=0;
for i=1:ll
%[co(i),t1,t2,tt1,tt2,tt3,tt4]=shuffle1(coa(i,:),p1(i),p2(i),p3(i),p4(i)...
%,p5(i),p6(i),p7(i));
[co1(i),t1,t2,tt1,tt2,tt3,tt4]=shuffle2(coa(i,:),p1(i),p2(i),p3(i),p4(i)...
,p5(i),p6(i),p7(i));
in=coa(i,:);
%johnson counters : first layer
[pc1(i+1),p1(i+1)]=johnson1(pc1(i),p1(i),in(1));
% second layer
[pc2(i+1),p2(i+1)]=johnson1(pc2(i),p2(i),t1(1));
```

```
[pc3(i+1),p3(i+1)]=johnson1(pc3(i),p3(i),t2(1));
% third layer
[pc4(i+1),p4(i+1)]=johnson1(pc4(i),p4(i),tt1(1));
[pc5(i+1),p5(i+1)]=johnson1(pc5(i),p5(i),tt2(1));
[pc6(i+1),p6(i+1)]=johnson1(pc6(i),p6(i),tt3(1));
[pc7(i+1),p7(i+1)]=johnson1(pc7(i),p7(i),tt4(1));
end;
%for i=1:length(coa)
%co(i)=d2a1(coa(i,:),4)/8;
%end;
%x=2*fft(co,length(co))./length(co);
%x=2*fft(co.*hanning(length(co))’)./(length(co)./2);
x1=2*fft(co1.*hanning(length(co1))’)./(length(co1)./2);
n=length(co1);
%nfft=length(co);
%window=hanning(nfft);
%noverlap=nfft./2;
%dflag=’none’;
Fs=U*M*Fin;
%Pxx=psd(co,nfft,Fs,window,noverlap,dflag);
%for i=2:Z+3
% P(i)=(x(i)*conj(x(i)))./length(co);
%end;
f=(0:length(co1)-1)*Fs./length(co1);
%NOISE=sum(Pxx(3:(Z-1)));
%SNR=10*log10(Pxx(Z+1)./NOISE);
%plot(co);grid
%pause
semilogx(f(1:n/2),log(20*log10(abs(x(1:n/2)))),’-’,f(1:n/2)...
,20*log10(abs(x1(1:n/2))),’--’);grid
%semilogx(log10(f(1:n/2)),log10(20*log10(abs(x1(1:n/2)))+200));grid
%title(‘Case N=15 bits retained’);
%xlabel(‘frequency (MHz)’);
%ylabel(‘spectrum output modulator (dB)’);
%pause

%plot(f,20*log10(abs(x)));grid
%xlabel(‘frequency (MHz)’);


%ylabel(‘spectrum output modulator (dB)’);
%pause
%plot(10*log10(Pxx(1:100)*norm(window)^2./sum(window)^2));
end;
```

```
function[smo,cmo,sm1o,sm2o,sm3o,co1]=scycle_2nd1(a,sm,cm,sm1,sm2,sm3,V,M,N,K)
%***************************************************************************
% model a 2nd order SD with CSA && 3 bit output && approximate clipping
% a input;cm,sm carry and sum vectors describing the
% first accumulator; sm1,sm2,sm3 vectors describing the state of the second
% accumulator;V word length of the input ; M sampling factor for the input
% in the modulator;the second accumulator is a 6/3 CSA.
%representation of quantizer step has same sign bit extention as input******
%K bit sign extention , 2's complement;V+K-1 word length inside the modulator
% cm1=zeros(1:V+K-1);cm=cm1(1,:);sm=cm;sm1=sm;sm2=sm;sm3=sm;
%**************************************************************************
 cu=cm;su=sm;sxm1=sm1;
sxm2=sm2;sxm3=sm3;
% length of c1 and a1 is K(=4)
for k=1:M
saa2=[ 0 sxm2(1:V+K-2)];
saa3=[0 0 sxm3(1:V+K-3)];
%quantize using 6 MSB
[co,qq1]=sum32(sxm1,saa2,saa3,V+K-1,N);
co1(k,:)=[ co(1) co(2) co(6)];
%q2=[ 1 ~r ~r ~r ];
 c1=[0  1  ~co(1) ~co(2)  ~co(6) ~co(6) ~co(6) ~co(6) ];
 a1=[a(V-4) a(V-3) a(V-2) a(V-1)  a(V) a(V) a(V) a(V)];
 a2=sum1(a1,c1,K+4);
%3 bit sign extention K=4
 a3=[a(1:V-5) a2];
 cmx=[ 0 cu(1:V+K-2)];
 smx=su;
%for i=1:V+K-1
 sa=xxor(a3,cmx,smx);
 ca=(cmx&(a3|smx))|(a3&smx);
%end;
% the second part of the modulator
w1=zeros(V+K-4);w2=w1(1,:);
d1=zeros(V-2);d=d1(1,:);
cm1=zeros(V+K-1);cmv=cm1(1,:);
c1=ones(V+K-1);c=c1(1,:);
%coming from the output

%coming from the first stage
cax=[0 cu(1:V+K-2)];
sax=su;

% 6/3 CSA
% N=5 used
s(k)=xor(qq1(5),qq1(6));
if (s(k)==0)
  w=[cmv(1:13)  1  ~co(1) ~co(2)  ~co(6) ~co(6) ~co(6)    ];
  [sa1,sa2,sa3]=CSA63(sax,cax,w,sxm1,saa2,saa3,V+K-1);
else
  w=[ cmv(1:13)  1  ~co(1) ~co(2)  ~co(6) ~co(6) ~co(6) ];
  saa2=cmv;saa3=cmv;

   sxm1=[ c(1:V+K-2)  ~co(6) ];

  [sa1,sa2,sa3]=CSA63(sax,cax,w,sxm1,saa2,saa3,V+K-1);

end;
cu=ca;su=sa;sxm1=sa1;
sxm2=sa2;sxm3=sa3;
end;
smo=sa;cmo=ca;sm1o=sa1;sm2o=sa2;sm3o=sa3;
%plot(co);grid
end;


function[a1,a2,a3]=CSA63(b1,b2,b3,b4,b5,b6,V)
for i=1:V
  [m] = sum2( b1(i) ,b2(i),b3(i) ,b4(i),b5(i) ,b6(i));
  a1(i)=m(1);a2(i)=m(2);a3(i)=m(3);
end;
end;

function[m]=sum21(a1,a2,a3,a4,a5,a6,a7)
V=a1+a2+a3+a4+a5+a6+a7;
m=dec2bin(V,3);
end;

function [c1,p1]=johnson1(c,p,e)

if e==1
  c1=~p;
  p1=c;
else
  c1=c;
  p1=p;
end;
end;
```

```
function[yout,t1,t2,tt1,tt2,tt3,tt4]=shuffle2(in,p1,p2,p3,p4,p5,p6,p7)

%pc1(1)=0;pc2(1)=0;pc3(1)=0;pc4(1)=0;pc5(1)=0;pc6(1)=0;pc7(1)=0;

%p1(1)=0;p2(1)=0;p3(1)=0;p4(1)=0;p5(1)=0;p6(1)=0;p7(1)=0;
% first layer
%for i=1:n
in1=[in(1) in(2) ~in(3) 0];

%johnson counter
%[pc1(i+1),p1(i+1)]=johnson1(pc1(i),p1(i),in(1));
s1=[p1 0 0 0];
s2=[~p1 0 0 0];

[o1,c1]=sum1(in1,s1,4);
[o2,c2]=sum1(in1,s2,4);
%second layer
t1=[o1(2) o1(3) o1(4)];
t2=[o2(2) o2(3) o2(4)];

%johnson counters
%[pc2(i+1),p2(i+1)]=johnson1(pc2(i),p2(i),t1(1));
%[pc3(i+1),p3(i+1)]=johnson1(pc3(i),p3(i),t2(1));
ss1=[p2 0 0 ];
ss2=[~p2 0 0 ];
ss3=[p3 0 0 ];
ss4=[~p3 0 0 ];

[o3,c3]=sum1(t1,ss1,3);
[o4,c4]=sum1(t1,ss2,3);
[o5,c5]=sum1(t2,ss3,3);
[o6,c6]=sum1(t2,ss4,3);

tt1=[o3(2) o3(3) 0];
tt2=[o4(2) o4(3) 0];
tt3=[o5(2) o5(3) 0];
tt4=[o6(2) o6(3) 0];

%third layer

%johnson counters
%[pc4(i+1),p4(i+1)]=johnson1(pc4(i),p4(i),tt1(1));
%[pc5(i+1),p5(i+1)]=johnson1(pc5(i),p5(i),tt2(1));
%[pc6(i+1),p6(i+1)]=johnson1(pc6(i),p6(i),tt3(1));
%[pc7(i+1),p7(i+1)]=johnson1(pc7(i),p7(i),tt4(1));
```

```
sss1=[p4 0 0];
sss2=[~p4 0 0];
sss3=[p5 0 0];
sss4=[~p5 0 0];
sss5=[p6 0 0];
sss6=[~p6 0 0];
sss7=[p7 0 0];
sss8=[~p7 0 0];

[oo1,cc1]=sum1(tt1,sss1,3);
[oo2,cc2]=sum1(tt1,sss2,3);
[oo3,cc3]=sum1(tt2,sss3,3);
[oo4,cc4]=sum1(tt2,sss4,3);
[oo5,cc5]=sum1(tt3,sss5,3);
[oo6,cc6]=sum1(tt3,sss6,3);
[oo7,cc7]=sum1(tt4,sss7,3);
[oo8,cc8]=sum1(tt4,sss8,3);

%y(i,:)=[oo1(2)  oo2(2) oo3(2) oo4(2) oo5(2)  oo6(2) oo7(2) oo8(2)];
yout=(oo1(2) + 1.035*oo2(2)+ 0.965*oo3(2)+ 1.005*oo4(2)+ 0.985*oo5(2) + ...
1.015*oo6(2)+ oo7(2)+ 0.995*oo8(2)-3.5)/4;
end;
%plot(yout);grid
end;
```

Appendix B

CS ΣΔ modulator VHDL description

```
-- VHDL Model Created from SGE Schematic mod.sch -- Mar  7 14:36:22 1997

library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_misc.all;
  use IEEE.std_logic_arith.all;
  use IEEE.std_logic_components.all;
use work.HANDY2.all;
entity moddl1 is
    Port (
            CLK : In   std_logic;
            RES : In   std_logic;
             O : Out  std_logic_vector (3 downto 0) );
end moddl1;
architecture BEHAVIORAL of moddl1 is
  signal Q : std_logic_vector(3 downto 0 );
 signal Q1 :  std_logic_vector(3 downto 0 );
 signal a :  std_logic_vector (15 downto 0);
  signal QV : std_logic;
--outputs from first adder
 signal SO0 :  std_logic_vector( 18 downto 0);
 signal SO1 : std_logic_vector(18 downto 0 );
 signal SO2 : std_logic_vector( 18 downto 0);
--outputs from second adder
 signal SOK0 : std_logic_vector(18 downto 0);
 signal SOK1 : std_logic_vector( 18 downto 0);
 signal SOK2 : std_logic_vector(18 downto 0 );
--intermediate values,outputs from first process
 signal VA0 : std_logic_vector( 18 downto 0);
 signal VA1 : std_logic_vector( 18 downto 0);
 signal VA2 : std_logic_vector(18 downto 0 );
 signal VB0 : std_logic_vector( 18 downto 0);
 signal VB1 : std_logic_vector(18 downto 0);
 signal VB2 : std_logic_vector(18 downto 0);
--feedbacks
  signal COPP : std_logic_vector(18 downto 0 );
  signal COP : std_logic_vector( 18 downto 0);

 signal  sm1 : std_logic_vector( 2 downto 0);
```

```
begin
 DIV: process(CLK,RES)
   variable N : integer := 0;
   variable m : integer;
   begin
    if RES = '1' then
       Q <= "0000";
    else
    if CLK'event and CLK = '1' then
     N := N+1;
     m := N mod 16;
     case m is
      when 0 => Q <= "0000";
      when 1 => Q <= "1000";
      when 2 => Q <= "0100";
      when 3 => Q <= "1100";
      when 4 => Q <= "0010";
      when 5 => Q <= "1010";
      when 6 => Q <= "0110";
      when 7 => Q <= "1110";
      when 8 => Q <= "0001";
      when 9 => Q <= "1001";
      when 10 => Q <= "0101";
      when 11 => Q <= "1101";
      when 12 => Q <= "0011";
      when 13 => Q <= "1011";
      when 14 => Q <= "0111";
      when 15 => Q <= "1111";
      when others => Q <= "XXXX";
  end case;
  end if;
  end if;
  QV <= Q(0);
  end process DIV;
DEC: process(QV,RES)
   variable N : integer := 0;
   variable m : integer;
   begin
    if RES = '1' then
       Q1 <= "0000";
    else
    if QV'event and QV= '1'  then
     N := N+1;
     m := N mod 16;
     case m is
      when 0 => Q1 <= "0000";
      when 1 => Q1 <= "1000";
      when 2 => Q1 <= "0100";
      when 3 => Q1 <= "1100";
      when 4 => Q1 <= "0010";
      when 5 => Q1 <= "1010";
      when 6 => Q1 <= "0110";
      when 7 => Q1 <= "1110";
      when 8 => Q1 <= "0001";
      when 9 => Q1 <= "1001";
      when 10 => Q1 <= "0101";
      when 11 => Q1 <= "1101";
      when 12 => Q1 <= "0011";
      when 13 => Q1 <= "1011";
      when 14 => Q1 <= "0111";
      when 15 => Q1 <= "1111";
      when others => Q1 <= "XXXX";
  end case;
  end if;
  end if;
  end process DEC;

ROM1 : process(Q1)
  begin
   case Q1 is
     when "0000" => a <= "0000000000000000";
     when "1000" => a <= "0000011110110010";
     when "0100" => a <= "0000111000111000";
     when "1100" => a <= "0001001010010011";
     when "0010" => a <= "0001010000011011";
     when "1010" => a <= "0001001010010011";
     when "0110" => a <= "0000111000111000";
     when "1110" => a <= "0000011110110010";
     when "0001" => a <= "0000000000000000";
     when "1001" => a <= "1111100001001110";
     when "0101" => a <= "1111000111001000";
     when "1101" => a <= "1110110101101101";
     when "0011" => a <= "1110101111100101";
     when "1011" => a <= "1110110101101101";
     when "0111" => a <= "1111000111001000";
     when "1111" => a <= "1111100001001110";
    when others => a <=  "XXXXXXXXXXXXXXXX";
  end case;
 end process ROM1;

 CSA1:process(SO0,SO1,SO2,SOK0,SOK1,SOK2,a,COPP,COP)
 -- CSA1:process
  variable A0 : std_logic_vector( 18 downto 0);
  variable A1 : std_logic_vector( 18 downto 0);
  variable  A2 : std_logic_vector(18 downto 0 );
  variable  aa1 : std_logic_vector(18 downto 0 );

variable  k0: std_logic_vector( 4 downto 0);


variable  kx0 : std_logic_vector( 2 downto 0);

  __************


   -- intermediate variables,second adder
  variable  B0 : std_logic_vector( 18 downto 0);
  variable  B1 : std_logic_vector( 18 downto 0);
  variable B2 : std_logic_vector(18 downto 0);

--
variable  k10 : std_logic_vector( 6 downto 0);


variable  k1x0 : std_logic_vector( 2 downto 0);

--
  variable S1 : std_logic_vector( 7 downto 0);
     variable  S2 : std_logic_vector(7 downto 0);
     variable S3 : std_logic_vector(7 downto 0 );
     variable  s : std_logic_vector( 7 downto 0);
     variable  c : std_logic_vector( 7 downto 0);
     variable  sx : std_logic_vector(7 downto 0);
     variable cx : std_logic_vector(7 downto 0 );
   variable cm : std_logic_vector( 8 downto 0);
variable  sm : std_logic_vector( 7 downto 0);
 variable VB10 : std_logic_vector(18 downto 0 );
 variable VB20 : std_logic_vector(18 downto 0 );
 variable VB00 : std_logic_vector(18 downto 0 );
 variable VA00 : std_logic_vector(18 downto 0 );
 variable VA10 : std_logic_vector(18 downto 0 );
 variable VA20 : std_logic_vector(18 downto 0 );
 variable ovf : std_logic;
 begin


   A0 := SO0;
   A1(0) := '0';
   A1(18 downto 1) := SO1(17 downto 0);
   A2( 1 downto 0) := "00";
   A2(18 downto 2) := SO2(16 downto 0);
   B0 := SOK0;
   B1(0) := '0';
   B1(18 downto 1) := SOK1(17 downto 0);
   B2( 1 downto 0) := "00";
   B2(18 downto 2) := SOK2(16 downto 0);
--input sign bit extention
aa1(18) := a(15);
aa1(17) := a(15);
aa1(16) := a(15);
aa1(15 downto 0) := a;
--compute CSA 7/3 for the two adders
 for I in 0 to 18 loop
k0(0) := aa1(I);
k0(1) := COPP(I);
k0(2) := A0(I);
k0(3) := A1(I);
k0(4) := A2(I);
kx0  := PS22(k0);
VA00(I) := kx0(0);
VA10(I) := kx0(1);
VA20(I) := kx0(2);
 end loop;


---- 888888888888888888888888888888888888888
 for I in 0 to 18 loop
k10(0) := A0(I);
k10(1) := A1(I);
k10(2) := A2(I);
k10(3) := B0(I);
k10(4) := B1(I);
k10(5) := B2(I);
k10(6) := COP(I);
k1x0 := PS1(k10);
--
VB00(I) := k1x0(0);
VB10(I) := k1x0(1);
VB20(I) := k1x0(2);
```

```
 end loop;


       S1(7 downto 0) := VB00(18 downto 11);
       S2(7 downto 0) := VB10(17 downto 10);
       S3(7 downto 0) := VB20(16 downto 9);
--transform in normal 2’s complement with carry propagate
-- 3/2 compression and shifted sumation
  for I in 0 to 7 loop
    s(I) := (S1(I) xor S2(I)) xor S3(I);
    c(I) :=not((not (S1(I) and S2(I))) and ( not (S2(I) and S3(I))) and
            (not (S1(I) and S3(I)))) ;
  end loop;
   sx := s;
   cx(7 downto 1) := c(6 downto 0);
   cx(0) := ‘0’;
  cm(0) :=’0’;
--Compute approximate output ; cm(0) = carry in is 0;

-- for I in 0 to 7 loop
--sm(I) := (sx(I) xor cx(I)) xor cm(I);
--cm(I+1) := (sx(I) and cx(I)) or (sx(I) and cm(I)) or (cx(I) and cm(I));
--  end loop;
  PP3 ( sx,cx,sm,ovf);

  sm1(2) <= sm(7);sm1(1) <= sm(3);sm1(0) <= sm(2);
-- multiplexing - approximate limiting if approximate overflow
-- if (cm(7) xor cm(6)) = ‘0’ then
   if ovf = ‘0’ then
   VB0 <= VB00;
   VB1 <= VB10;
   VB2 <= VB20;
 else
   VB0(18) <= not sm(7);
  VB0(17) <= not sm(7);
   VB0(16 downto 0) <= “11111111111111111”;
   VB1 <= “0000000000000000000”;
   VB2 <= “0000000000000000000”;
 end if;
 VA0 <= VA00;
 VA1 <= VA10;
 VA2 <= VA20;
 end process CSA1;
```

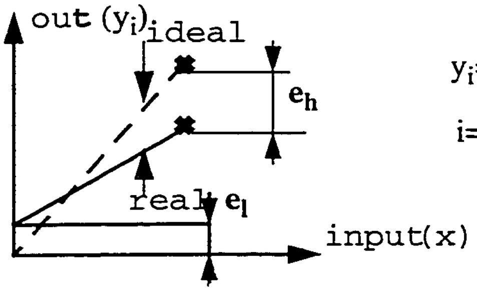
```
--Registering processes


----------

REG1: process(CLK,RES,VA0,VA1,VA2,VB0,VB1,VB2,sm1)

      begin
        if RES ='1'  then

          SO0(18 downto 0) <= "0000000000000000000";
          SO1(18 downto 0) <= "0000000000000000000";
          SO2(18 downto 0) <= "0000000000000000000";
          SOK0(18 downto 0) <= "0000000000000000000";
          SOK1(18 downto 0) <= "0000000000000000000";
          SOK2 (18 downto 0)<= "0000000000000000000";
          O <= "0000";
          COPP <= "1111111000000000000";
          COP  <= "1111110000000000000";
        else
         if CLK'EVENT and CLK ='1' then

          SO0(18 downto 0) <= VA0(18 downto 0);
          SO1(18 downto 0) <= VA1(18 downto 0);
          SO2(18 downto 0) <= VA2(18 downto 0);
          SOK0(18 downto 0) <= VB0(18 downto 0);
          SOK1(18 downto 0) <= VB1(18 downto 0);
          SOK2(18 downto 0) <= VB2(18 downto 0);
--feedbacks
          COPP(11 downto 0 ) <="000000000000";
          COPP(18) <= not sm1(2);
          COPP(17) <= not sm1(2);
          COPP(16) <= not sm1(2);
          COPP(15) <= not sm1(2);
          COPP(14) <= not sm1(1);
          COPP(13) <= not sm1(0);
          COPP(12) <='1';
--
          COP(12 downto 0) <= "0000000000000";
          COP(18) <= not sm1(2);
          COP(17) <= not sm1(2);
          COP(16) <= not sm1(2);
          COP(15) <= not sm1(1);
          COP(14) <= not sm1(0);
```

```
           COP(13) <= '1';
--output

           O(3 downto 1) <= sm1( 2 downto 0);
           O(0) <= '1';

         end if;
         end if;

 end process REG1;

 end BEHAVIORAL;



library IEEE;
  use IEEE.std_logic_1164.all;
package HANDY is

procedure PP1( AA,BB : in std_logic_vector;SS: out  std_logic_vector;
          OVF1 : out std_logic);
procedure PP2( MM,NN : in std_logic_vector;QQ: out  std_logic_vector);

function PS1(val1 : std_logic_vector(6 downto 0)) return std_logic_vector;
function PS2(val2 : std_logic_vector(4 downto 0)) return std_logic_vector;
end HANDY;
package body HANDY is

--second CSA adder

  procedure  PP1 ( AA,BB : in std_logic_vector;
          SS : out  std_logic_vector; OVF1 :out std_logic) is
  variable P0,P1,P2,P3,P4,P5,G0,G1,G2,G3,G4,G5 : std_logic;
  variable AAV,BBV,SSV :std_logic_vector(AA'LENGTH-1 downto 0);
  variable CCV : std_logic_vector(AA'LENGTH downto 0);
  begin
  AAV := AA; BBV := BB;
  P0 := AA(0) xor BB(0); G0 := AA(0) and BB(0);
  P1 := AA(1) xor BB(1); G1 := AA(1) and BB(1);
  P2 := AA(2) xor BB(2); G2 := AA(2) and BB(2);
  P3 := AA(3) xor BB(3); G3 := AA(3) and BB(3);
  P4 := AA(4) xor BB(4); G4 := AA(4) and BB(4);
  P5 := AA(5) xor BB(5); G5 := AA(5) and BB(5);

  CCV(0) := '0';
 CCV(1) := G0 or (P0 and CCV(0));

 CCV(2) := G1 or (P1 and G0) or (P1 and P0 and CCV(0));

 CCV(3) := G2 or ( P2 and G1) or (P2 and P1 and G0) or
         (P2 and P1 and P0 and CCV(0));

 CCV(4) :=  G3 or (P3 and G2) or (P3 and P2 and  G1) or
          (P3 and  P2 and  P1 and G0) or
          (P3 and  P2 and P1  and P0 and CCV(0));


 CCV(5) :=  G4 or (P4 and G3) or (P4 and P3 and G2) or
          (P4 and P3 and P2 and G1 ) or
          (P4 and P3 and P2 and P1 and G0 ) or
         (P4 and P3 and P2 and P1  and P0 and CCV(0));

 CCV(6) :=  G5 or ( P5 and G4) or (P5 and P4 and G3) or
         ( P5 and P4 and P3 and G2 ) or
         ( P5 and P4 and P3 and P2 and G1) or
         ( ( P5 and P4 and P3 and P2 and P1) and
         ( G0 or( P0 and CCV(0))));

 SSV(0) := P0 xor CCV(0);

 SSV(1) := P1 xor CCV(1);

 SSV(2) := P2 xor CCV(2);

 SSV(3) := P3 xor CCV(3);

 SSV(4) := P4 xor CCV(4);

SSV(5) := P5 xor CCV(5) ;

 OVF1 := CCV(6) xor CCV(5) ;

 SS := SSV;

end PP1;

procedure  PP2 ( MM,NN : in std_logic_vector;
            QQ : out  std_logic_vector) is
 variable PR0,PR1,PR2,PR3,GR0,GR1,GR2,GR3 : std_logic;
 variable CRV,ARV,BRV,QQS : std_logic_vector(MM'LENGTH downto 0);
 begin
 ARV := MM; BRV := NN;
 PR0 := MM(0) xor NN(0); GR0 := MM(0) and NN(0);
 PR1 := MM(1) xor NN(1); GR1 := MM(1) and NN(1);
 PR2 := MM(2) xor NN(2); GR2 := MM(2) and NN(2);
 PR3 := MM(3) xor NN(3); GR3 := MM(3) and NN(3);


 CRV(0) := '0';

 CRV(1) := GR0 or (PR0 and CRV(0));

 CRV(2) := GR1 or (PR1 and GR0) or (PR1 and PR0 and CRV(0));

 CRV(3) := GR2 or ( PR2 and GR1) or (PR2 and PR1 and GR0) or
        (PR2 and PR1 and PR0 and CRV(0));


 QQS(0) := PR0 xor CRV(0);

 QQS(1) := PR1 xor CRV(1);

 QQS(2) := PR2 xor CRV(2);

 QQS(3) := PR3 xor CRV(3);

 QQ := QQS;

end PP2;

function PS1(val1 : std_logic_vector(6 downto 0)) return std_logic_vector is
 variable oq : std_logic_vector(2 downto 0);
 variable va : std_logic_vector( 6 downto 0);
 variable S0,C0,S1,C1,PP0, PP1,GG0,GG1,CC0,CC1,oqx : std_logic;
  begin
   va := val1;
   S0 := (va(0) xor va(1)) xor va(2);
   C0 := not((not(va(0) and va(1))) and (not (va(1) and va(2))) and
        (not (va(0) and va(2)))) ;
   S1 := (va(3) xor va(4)) xor va(5);
   C1 := not((not(va(3) and va(4))) and (not (va(4) and va(5))) and
        (not (va(3) and va(5)))) ;
   PP0 := S0 xor S1;
   GG0 := S0 and S1;
   PP1 := C0 xor C1;
   GG1 := C0 and C1;
   CC0 := GG0 or ( PP0 and va(6));
   CC1 := GG1 or ( PP1 and GG0) or ( PP1 and PP0 and va(6));
   oq(0) := PP0 xor va(6);
   oq(1) := CC0 xor PP1;
   oq(2) := CC1;

   return oq;
  end PS1;
--first CSA adder
function PS2(val2 : std_logic_vector(4 downto 0)) return std_logic_vector is      variable oq1 :
std_logic_vector(2 downto 0);
  variable va1 : std_logic_vector( 4 downto 0);
  variable S01,C01,S11,C11,PP01, PP11,GG01,GG11,CC01,CC11 : std_logic;
   begin
    va1 := val2;
   S01 := (va1(0) xor va1(1)) xor va1(2);
   C01 := not((not(va1(0) and va1(1))) and(not (va1(1) and va1(2))) and
          (not (va1(0) and va1(2)))) ;
   S11 := va1(3) xor va1(4);
   C11 :=not((not( va1(3))) or (not( va1(4)))) ;
   PP01 := S01 xor S11;
   GG01 := S01 and S11;
   PP11 := C01 xor C11;
   GG11 := C01 and C11;
   CC01 := GG01 ;
   CC11 := GG11 or ( PP11 and GG01) ;
   oq1(0) :=  PP01 ;
   oq1(1) := CC01 xor PP11;
   oq1(2) := CC11;

   return oq1;
  end PS2;

 end HANDY;
```

Appendix C

Three layer ΣΔ-based bit shuffling network (Galton method)

Figure 1 shows the transfer function of a unit D/A element used in the Galton bit shuffling network. Figure 2 shows the switching cell used in the method.



$y_i=(1+e_{hi}-e_{li})x + e_{li}$,

$i=1,..,8$

Fig.5.2. Unit D/A element input/output transfer function

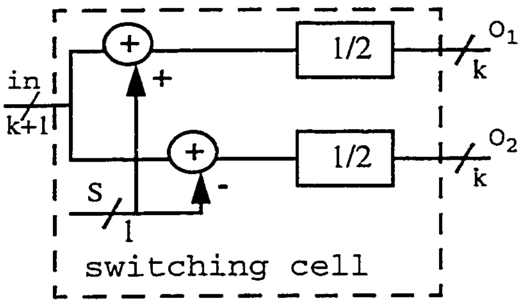


Fig.5.3. Bit shuffling network switching cell

The e_{hi} and e_{li} are the errors in the analog output level for a ONE and respectively a ZERO input..In Figure 2, s is a one bit sequence generated inside the switching cell. The following relations hold for every sample time n :

$$o1(n) + o2(n) = in(n); \tag{1}$$

$$o1(n) - o2(n) = s(n); \tag{2}$$

It follows that

$$o_1(n) = \frac{in(n) + s(n)}{2} \tag{3}$$

$$o_2(n) = \frac{in(n) - s(n)}{2} \tag{4}$$

For a one bit D/A element it is true that

$$y(n) = \alpha \cdot x + \varepsilon_h \tag{5}$$

$$\alpha = 1 + \varepsilon_h - \varepsilon_l \tag{6}$$

The block diagram of a three level Galton bit shuffling network is shown in Figure 3.

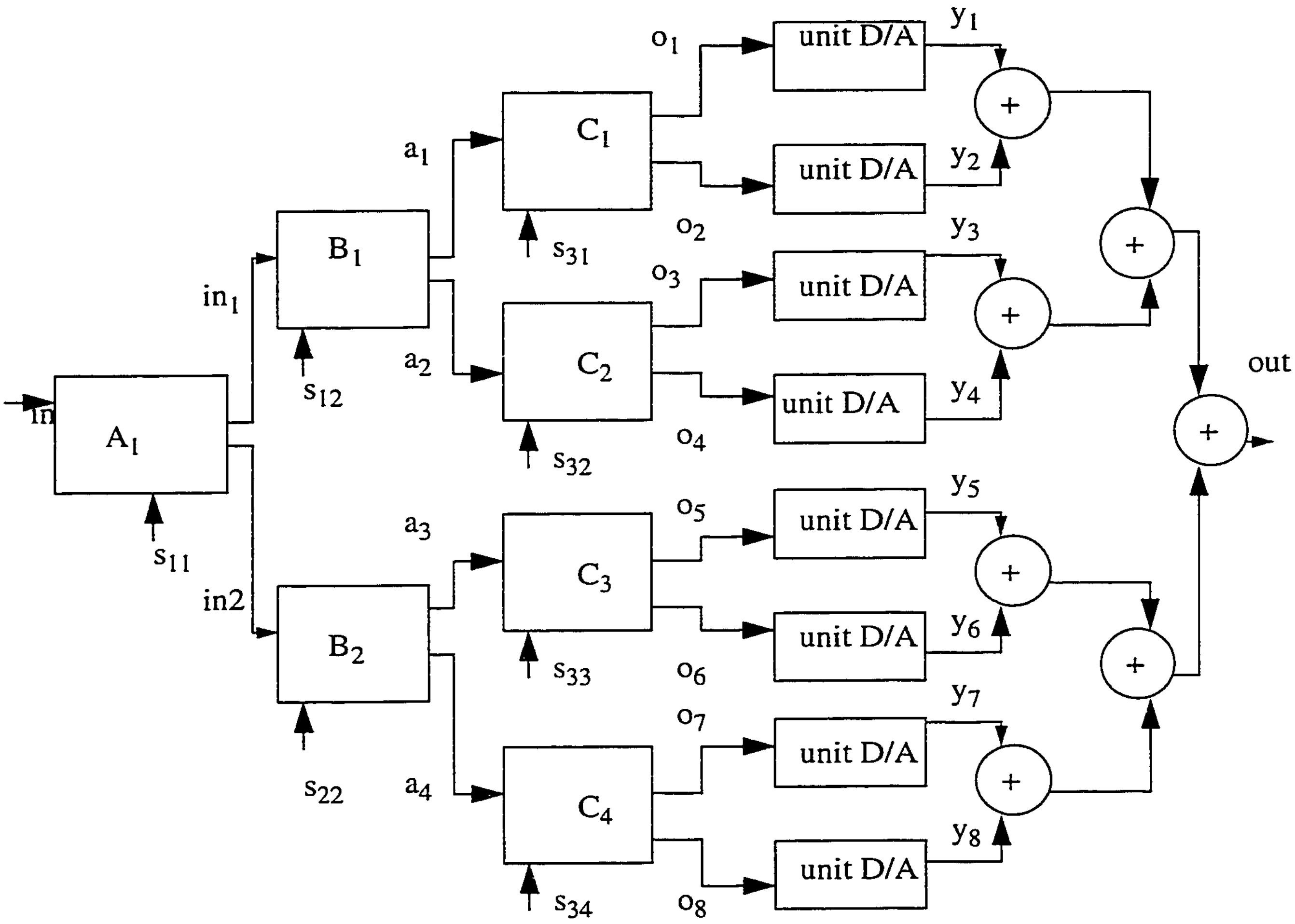


Fig.3. Three layer ΣΔ-based bit shuffling network (Galton network)

In the above Figure, the switching cells are placed in three layers : layer 1 (A_1), layer 2 (B_1, B_2) and layer 3 (C_1, C_2, C_3, C_4). The one bit sequences s_{ij} are internally generated. Following the general relations (3) and (4), one can write:

layer 1 :

$$in_1(n) = \frac{s_{11}(n) + in(n)}{2} \tag{7}$$

$$in_2(n) = \frac{-s_{11}(n) + in(n)}{2} \tag{8}$$

layer 2 :

$$a_1(n) = \frac{s_{21}(n) + in_1(n)}{2} \tag{9}$$

$$a_2(n) = \frac{-s_{21}(n) + in_1(n)}{2} \tag{10}$$

$$a_3(n) = \frac{s_{22}(n) + in_2(n)}{2} \tag{11}$$

$$a_4(n) = \frac{-s_{22}(n) + in_2(n)}{2} \tag{12}$$

layer 3 :

$$o_1(n) = \frac{s_{31}(n) + a_1(n)}{2} \tag{13}$$

$$o_2(n) = \frac{-s_{31}(n) + a_1(n)}{2} \tag{14}$$

$$o_3(n) = \frac{s_{32}(n) + a_2(n)}{2} \tag{15}$$

$$o_4(n) = \frac{-s_{32}(n) + a_2(n)}{2} \tag{16}$$

$$o_5(n) = \frac{s_{33}(n) + a_3(n)}{2} \tag{17}$$

$$o_6(n) = \frac{-s_{33}(n) + a_3(n)}{2} \tag{18}$$

$$o_7(n) = \frac{s_{34}(n) + a_4(n)}{2} \tag{19}$$

$$o_8(n) = \frac{-s_{34}(n) + a_4(n)}{2} \tag{20}$$

The output of the bit shuffling network is given by :

$$out(n) = \sum_{i=1}^{8} y_i(n) = \sum_{i=1}^{8} \alpha_i \cdot o_i + \varepsilon_{hi} \tag{21}$$

Using the relations (7) to (20) , the output value becomes:

$$out(n) = A \cdot in(n) + B \cdot s_{11}(n) + C \cdot s_{12}(n) + D \cdot s_{21}(n) + E \cdot s_{22}(n) + F \cdot s_{31}(n) + G \cdot s_{32}(n) + H \cdot s_{33}(n) + I \cdot s_{34}(n) + J \tag{22}$$

where :

$$A = \frac{\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 + \alpha_7 + \alpha_8}{2^3} \tag{23}$$

$$B = \frac{\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \alpha_5 - \alpha_6 - \alpha_7 - \alpha_8}{2^3} \tag{24}$$

$$C = \frac{\alpha_1 + \alpha_2 - \alpha_3 - \alpha_4}{2^2} \tag{25}$$

$$D = \frac{\alpha_5 + \alpha_6 - \alpha_7 - \alpha_8}{2^2} \tag{26}$$

$$E = \alpha_1 - \alpha_2 \tag{27}$$

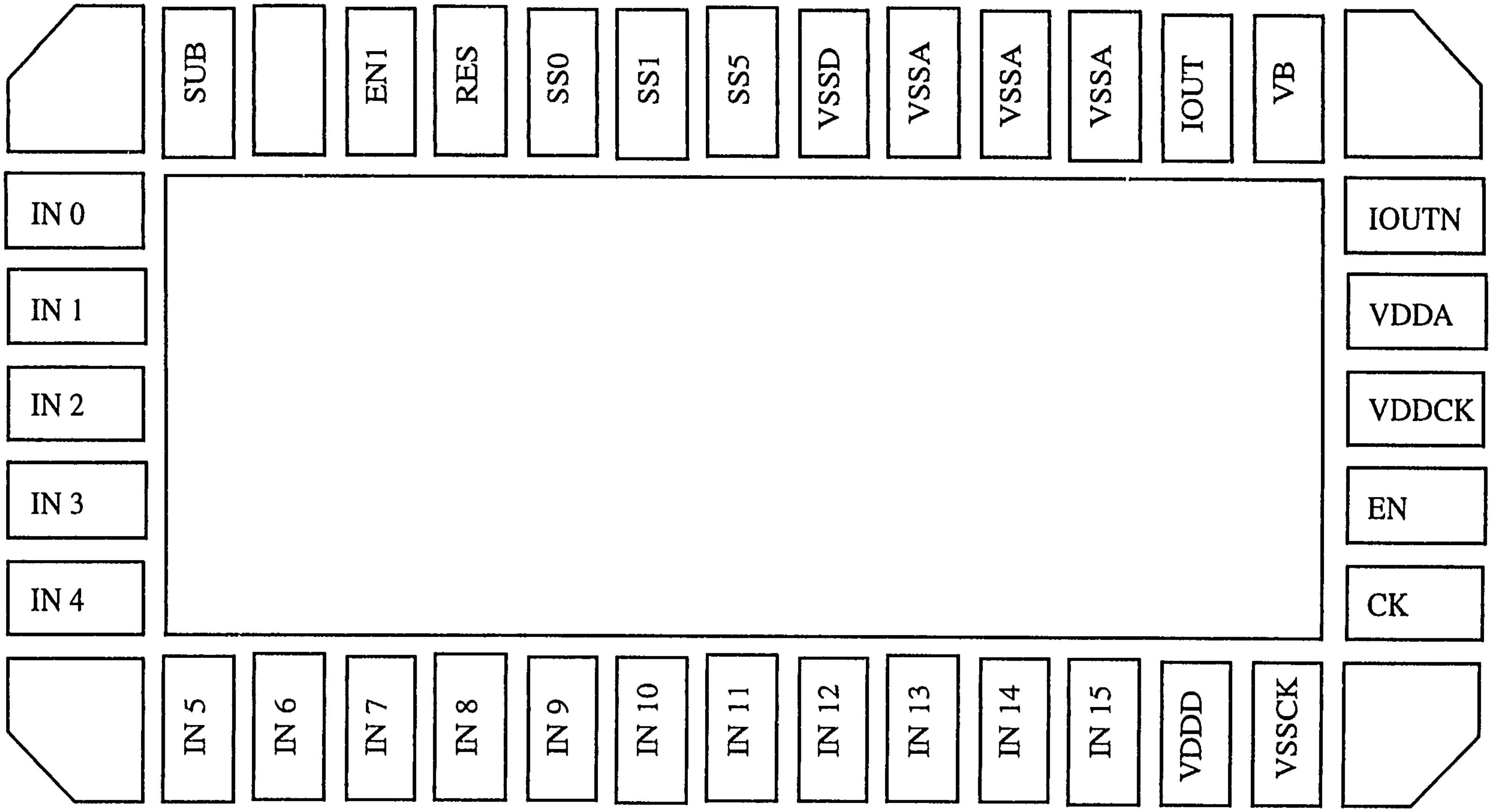
$$F = \alpha_3 - \alpha_4 \tag{28}$$

$$G = \alpha_5 - \alpha_6 \tag{29}$$

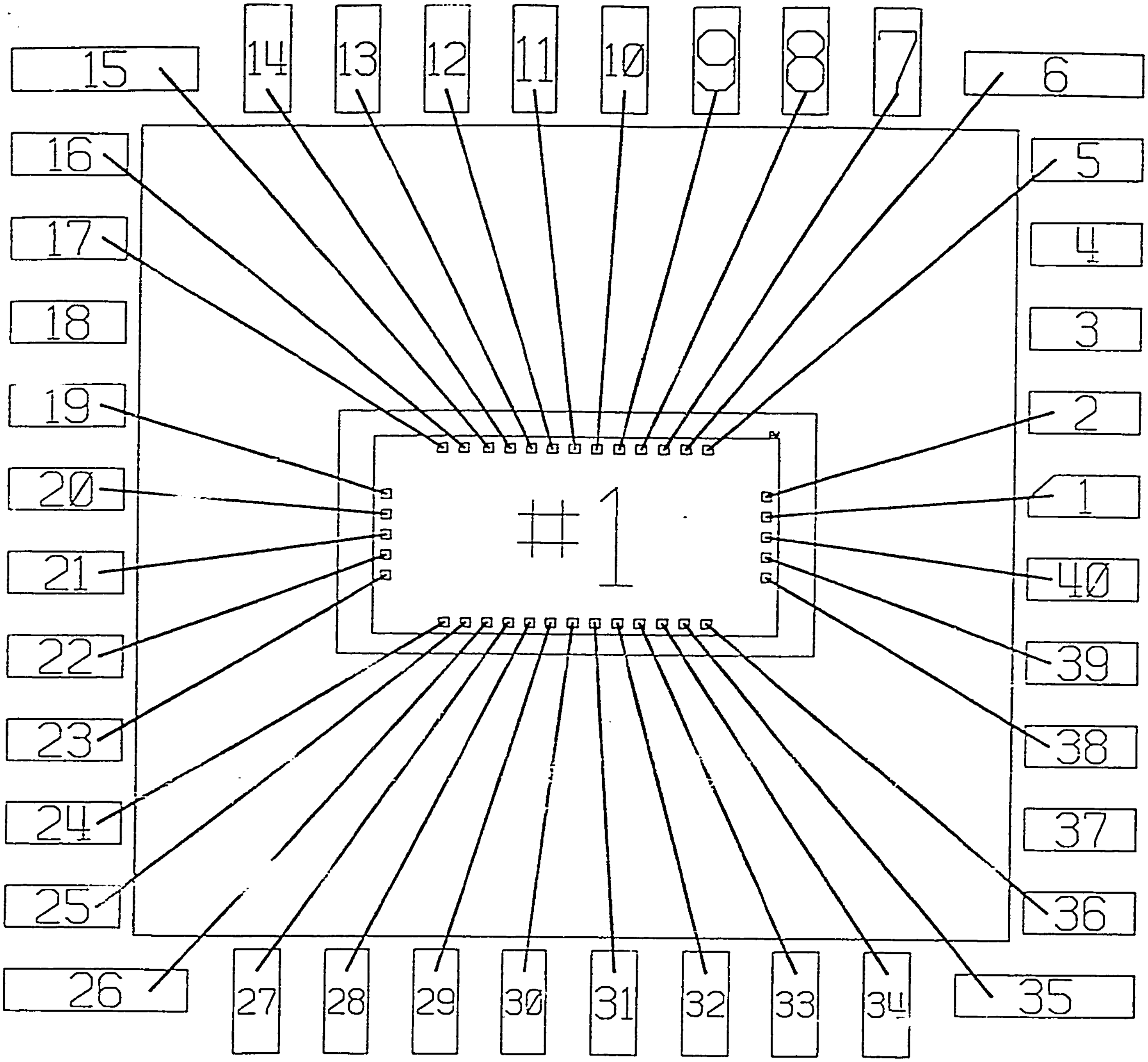
$$H = \alpha_7 - \alpha_8 \tag{30}$$

$$I = \varepsilon_1(n) + \varepsilon_2(n) + \varepsilon_3(n) + \varepsilon_4(n) + \varepsilon_5(n) + \varepsilon_6(n) + \varepsilon_7(n) + \varepsilon_8(n) \tag{31}$$

If the 1 bit sequences s_{ij} in equation (22) are chosen to be n-th order $\Sigma\Delta$ and are uncorrelated, the mismatch noise is n-th order $\Sigma\Delta$ noise shaped out of the band of interest.


SUB
EN1
RES
SS0
SS1
SS5
VSSD
VSSA
VSSA
VSSA
IOUT
VB
IN 0
IN 1
IN 2
IN 3
IN 4
IOUTN
VDDA
VDDCK
EN
CK
IN 5
IN 6
IN 7
IN 8
IN 9
IN 10
IN 11
IN 12
IN 13
IN 14
IN 15
VDDD
VSSCK

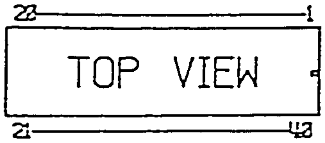
CHIP PINS



N79VAW

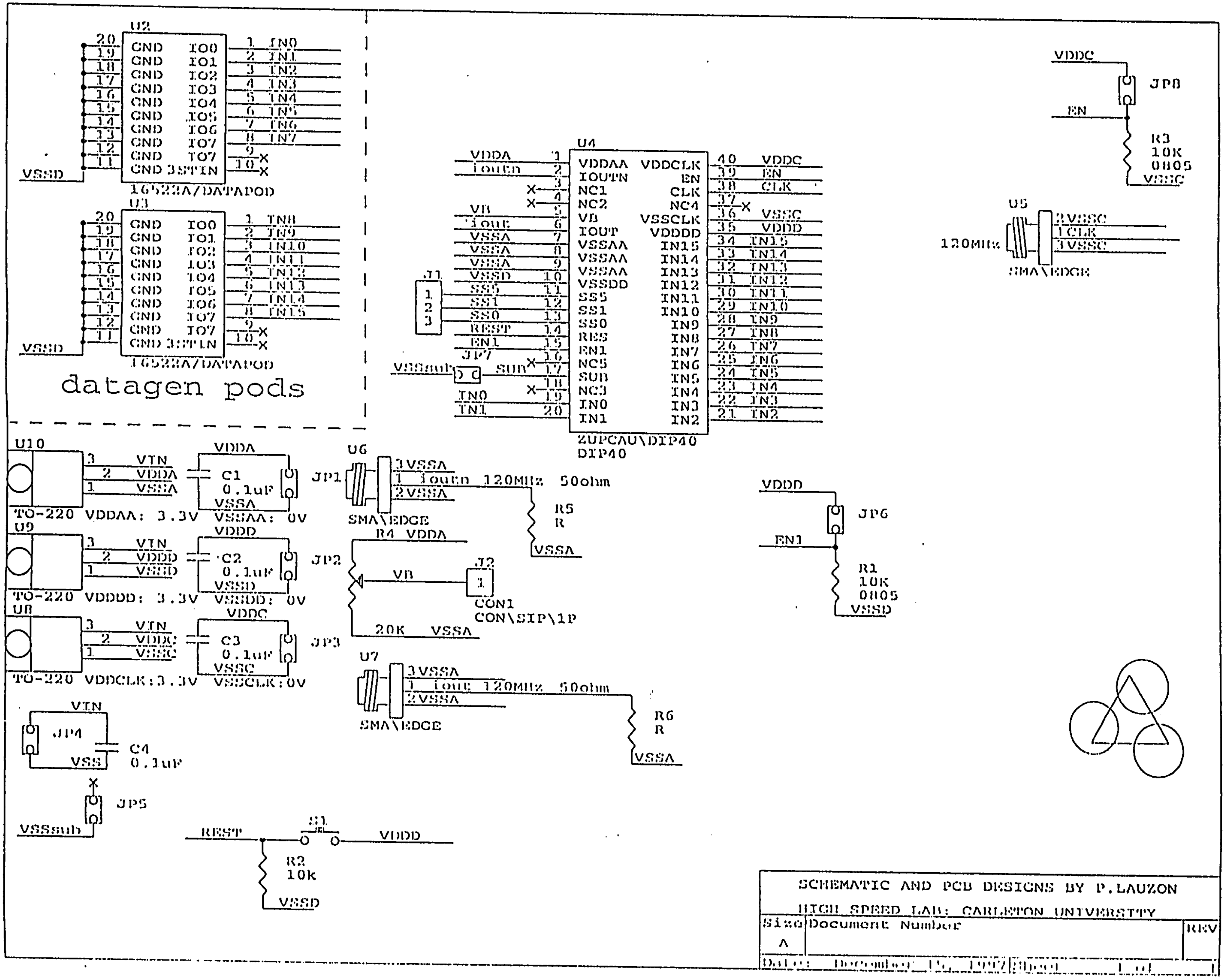
1777-COM/CARLETONU

#1: 52940/APSIMON/SDDAC



DIP40: 25 PARTS

19-SEP-1997



SCHEMATIC AND PCB DESIGNS BY P.LAUZON

HIGH SPEED LAB: CARLETON UNIVERSTTY

Size	Document Number	REV
A		
Date: December 15, 1997	Sheet 1 of	1