Thomas Cemernek, BSc

A Comparative Study of Power Saving Techniques in Digital MASH Delta Sigma Modulators for Fractional N Frequency Synthesis

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Biomedical Engineering

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn., Peter Söser

Institut für Elektronik
Inffeldgasse 12/I, A - 8010 Graz

Infineon Technologies Austria AG: MSc., BEng. Alberto Garcia Izquierdo and
Dipl.-Ing. Franz Pammer

Graz, August 2016

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date

Signature

Ethischer Kodex der TU Graz:

Wir sind eine Gemeinschaft der Forschenden, Lehrenden, Studierenden, Mitarbeiterinnen und Mitarbeiter und Alumnae und Alumni in einer Atmosphäre der intellektuellen Freiheit und Verantwortung. Wir bekennen uns zur Verbindung von Forschung und Lehre auf höchstem Niveau im weltweiten Wettbewerb vergleichbarer Einrichtungen. Auf den Grundlagen der Vision der TU Graz und dem Wissen, dass TechnikerInnen und WissenschafterInnen mit ihrem Denken und Handeln einen überaus starken Einfluss auf Gesellschaft und Umwelt ausüben, ist es der TU Graz wichtig, schon zu Beginn des Studiums bzw. eines Dienstverhältnisses auf die damit verbundene Verantwortung hinzuweisen und ihre Angehörigen zu einem ethisch einwandfreien Handeln zu verpflichten. Als StudentIn und AngehörigeR der TU Graz habe ich die Richtlinie des Rektorates der Technischen Universität Graz zur Sicherung guter wissenschaftlicher Praxis und zur Vermeidung von Fehlverhalten in der Wissenschaft gelesen und verstanden, ich unterstütze und anerkenne diese Richtlinie vorbehaltlos.

Graz, Datum *(Unterschrift)*

Abstract:

A Comparative Study of Power Saving Techniques in Digital MASH DSM

Digital Delta Sigma Modulation is a widely spread tool for fractional-N frequency synthesis. In this Master thesis different techniques for realizing Digital Delta Sigma Modulators with the goal to reduce area and power consumption while keeping the same performance as the currently used structure are analysed. To find new approaches literature is reviewed and according to interesting papers Matlab models are built. After intensive simulation, the best model is chosen and built in System Verilog. The generated System Verilog model is verified and synthesised. Finally the chosen DDSM is compared to the old one with respect to power and area. It can be observed that it is possible to save area and power while improving performance.

Keywords: Digital Delta Sigma Modulator, DDSM, MASH, Power, Area

Zusammenfassung:

Vergleichende Studie von energiesparenden Techniken zur Realisierung von Digitalen MASH DSM

Digitale Delta Sigma Modulatoren werden bei der fractional-N Frequenzsynthese häufig verwendet. In dieser Masterarbeit werden verschiedene Techniken zur Realisierung von Digitalen Delta Sigma Modulatoren mit dem Ziel Fläche und Energie einzusparen bei zumindest gleich bleibender Performance in Bezug auf die derzeit verwendete Struktur betrachtet. Um verschiedene Realisierungsmöglichkeiten zu finden, wurde Literatur herangezogen. Basierend auf interessanten Papers wurden Matlabmodelle erzeugt. Mittels dieser Modelle wurden die Realisierungsmöglichkeiten auf Herz und Nieren getestet und die beste ausgewählt. Anhand des ausgewählten Modells wurde ein System Verilog Module geschrieben, welches verifiziert und synthetisiert wurde. Anschließend wurde die neu erstellte Struktur mit der bestehenden bezüglich Energie und Fläche verglichen. Das Ergebnis ist, dass es durch Verwendung der neuen Struktur möglich ist, Energie und Fläche einzusparen.

Schlüsselwörter: Digitaler Delta Sigma Modulator, DDSM, MASH, Energie, Fläche

Contents

1 List of Figures

2 List of Tables

3 List of Abbreviations

RF Radio Frequency

PLL Phase Locked Loop

DDSM Digital Delta Sigma Modulator

PD Phase Detector

VCO Voltage Controlled Oscillator

LPF Low-Pass Filter

EFM Error Feedback Modulator

MASH Multi Stage Noise Shaping

ECN Error Cancellation Network

LSB least significant bit

PSD Power Spectral Density

GCD greatest common divisor

RC Reduced Complexity

ASK Amplitude Shift Keying

FSK Frequency Shift Keying

SV System Verilog

MSB most significant bit

RTL Register Transfer Level

CDC Clock Domain Crossing

ECG electrocardiograms

$Sp0_2$ saturation of peripheral oxygen

ICD implantable cardioverter defibrillator

4 Introduction

In certain kinds of sensors, like implants in the human body, it is difficult to transmit data. To eliminate these difficulties radio frequency (RF) is used. In so-called RF transmitters phase locked loops convert frequencies up for transmission and down for reception.

4.1 Phase Locked Loop - PLL

Frequency multiplication from crystal sources is a challenging endeavour [1]. The following sections provides an overview of it starting from the basic idea of frequency synthesis up to why DDSMs are needed.

4.1.1 Integer-N PLL

The concept is to generate a frequency out of a given reference frequency. In figure 4.1 a block diagram of the idea is displayed. Due to this model equation 4.1 is satisfied. N is an integer value, the factor the reference frequency is multiplied with to get the output frequency.

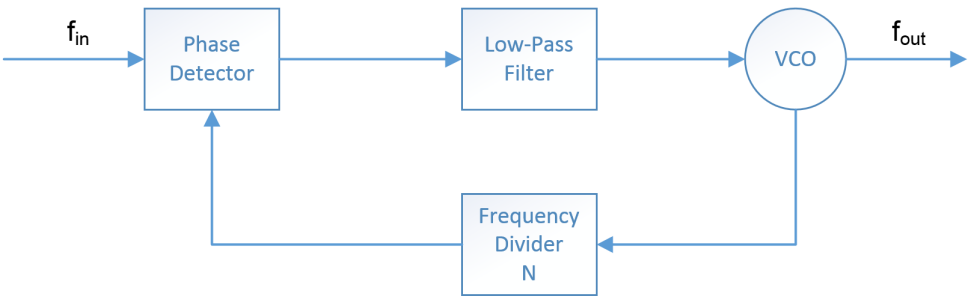

Figure 4.1: Block diagram of an integer-N PLL

$$f_{out} = N * f_{in} \tag{4.1}$$

The PLL consists of a phase detector, a low pass filter and a voltage controlled oscillator.

Phase Detector - PD: The PD compares the reference and the returned output signal, whose frequency is the one generated by the voltage controlled oscillator divided by N. The output is the phase error between the frequencies of these two signals.
Voltage Controlled Oscillator - VCO: The VCO generates a frequency from its input voltage. This frequency is N times the reference frequency when the loop has reached its steady state.
Low-Pass Filter - LPF: The LPF ensures the function of the PLL during the initial power up and when the reference frequency or the factor N change. The second function of the LPF is to reduce the noise in the VCOs input voltage [2] by suppressing high frequencies [1]. Otherwise the VCO generates unwanted side-bands in the output frequency spectrum.

The model displayed in figure 4.1 has a few disadvantages [1]:

- The minimum channel spacing Δf_{out} is the reference frequency, because it is only possible to use integer values for N. In case fine tuning is necessary, the only option is to change the reference frequency.
- The factor N should not be too high, because also the phase noise is multiplied with the factor. That is a problem when a high output frequency with a small resolution is needed.
- Spurs are located at multiples of the reference frequency. Spurs are unwanted frequency modulations, generated by the VCO. To cut them off the loop bandwidth has to be decreased below f_{ref}. Decreasing the bandwidth leads to a larger settling time and large settling times are not allowed by most communications standards.

In the following section a structure to eliminate these disadvantages is presented.

4.1.2 Fractional-N PLL

In figure 4.2 a block diagram of a fractional-N PLL is displayed. The difference to figure 4.1 is that the frequency division factor is variable. The new value N consists of a fixed part N_0 and

a variable part c. The variable part is generated by the comparator displayed in figure 4.2 and can be 0 or 1. The comparator output is periodic and its time average is equal to its input.

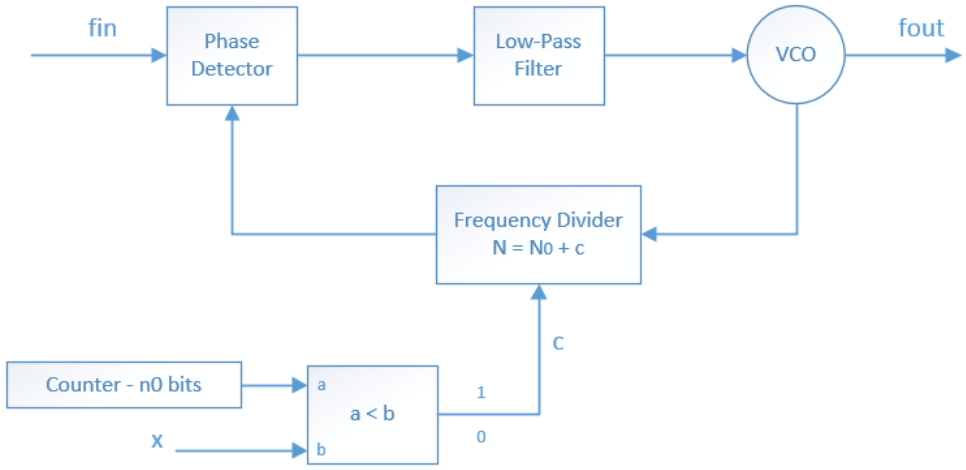

Figure 4.2: Block diagram of a fractional-N PLL

The mathematical relation and an example for better understanding is displayed in following:

Mathematical Relation:

$frac$: fractional input value

n_0: number of bits

x : fractional input value multiplied with M - comparator input

M : period of the output

c_{mean} : average output value

Each period M has x ones and $M - x$ zeros.

$$M = 2^{n_0} \tag{4.2}$$

$$x = frac * M \tag{4.3}$$

$$c_{mean} = \frac{x * 1 + (M - x) * 0}{M} = \frac{x}{M} \tag{4.4}$$

Example:

$frac = 0,25$
$n_0 = 10$

$$M = 2^{10} = 1024 \tag{4.5}$$

$$x = 0,25 * 1024 = 256 \tag{4.6}$$

$$c_{mean} = \frac{256}{1024} = 0,25 \tag{4.7}$$

Through equation 4.7 its confirmed, the time average of the output is equal to x - the comparator input. For the fractional-N PLL this means that the relation between f_{in} and f_{out} is as follows:

$$f_{out} = N * f_{in} \tag{4.8}$$

$$N = N_0 + frac \tag{4.9}$$

The structure in figure 4.2 has the following advantages [1]:

- It is possible to multiply the input frequency with a fractional value.
- The minimum channel spacing (frequency resolution) is given through equation 4.10.
- A higher input frequency for a given channel spacing can be used.

$$\Delta f = \frac{1}{2^{n_0}} f_{ref} \tag{4.10}$$

Beside the advantages one disadvantage is established. The division value switches between two different values. Due to this there is always a periodic phase error between the actual and the average divider value. The difference generates unwanted spurious tones at fractional multiplies of the reference frequency. The first tone is located at $\frac{1}{2^{n_0}} f_{ref}$ and can be in the in-band. Tones outside the bandwidth can be neglected, because the low pass filter cuts them off. One method

to attenuate this problem is to use a sequence whose power is mostly located outside the bandwidth. Such sequences can be generated with Digital Delta Sigma Modulators.

4.2 Digital Delta Sigma Modulator - DDSM

In this section Digital Delta Sigma Modulators are discussed from first principles. Figure 4.3 shows a DDSM in block diagram form. It has an n_0-bit input $x[n]$ and an m-bit output $y[n]$. Throughout this work, its assumed that $n_0 > m$.

Figure 4.3: Block diagram of a DDSM [3]

In figure 4.4 the operation of the DDSM in the frequency domain is depicted: (a) a high resolution, low frequency signal is applied to the input of the DDSM. (b) displays the output signal of the DDSM. Despite the lower resolution of the output, it can be observed that it consists of the whole input signal, and a shaped quantization noise. Through the DDSM the quantization noise is shaped away from the input signal band to higher frequencies, therefore increasing signal-to-noise ratio in the band of interest.

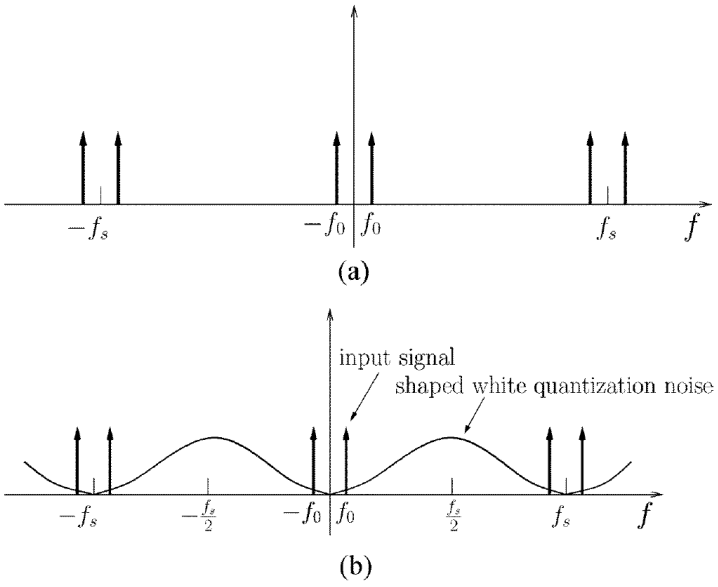

Figure 4.4: Spectrum of a DDSM in the frequency domain [3]

When the input to the DDSM is fixed the output is periodic, because the DDSM is a finite state machine. The cycle length should be as large as possible; otherwise the power of the quantisation noise is distributed among a small number of discrete frequencies and the power in those frequencies is very high. If the cycle length is big the power is distributed among a much bigger number of frequencies and the power in each of those becomes small. In general the cycle length depends on the input value, the initial condition and the architecture of the modulator. In best case the spectrum looks like a smooth shaped curve, like in figure 4.4(b) displayed. The shorter the cycle length gets the more spikes appear in the spectrum.

4.2.1 Error Feedback Modulator - EFM

The simplest implementation of a Digital Delta Sigma Modulator is a digital accumulator also called Error Feedback Modulator. In figure 4.5 an accumulator and its model are displayed. The accumulator has an n_0-bit input signal $x[n]$, a 1-bit carry output signal $c[n]$ and an n_0-bit one clock cycle delayed error signal $s[n]$.

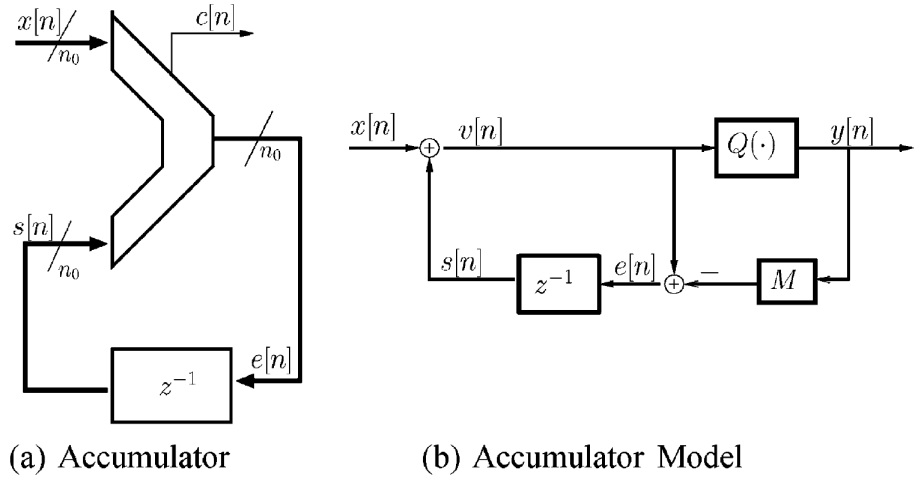

Figure 4.5: Digital accumulator (a) and its model (b) [3]

In figure 4.5 (b) the signal and systems representation of the EFM is shown. The model works as follows: The input signal $x[n]$ is added to the delayed error $s_[n]$ from the last input. The Q(.) block is a one-bit quantizer. Its output is 1 if $v[n]$ is bigger or equal to M otherwise it is 0. Mathematically this can be written:

$$y[n] = \begin{cases} 0, & v[n] < M \\ 1, & v[n] \geqslant M \end{cases} . \tag{4.11}$$

The error $e[n]$ behaves like equation 4.12, gets delayed and feed back to the input.

$$e[n] = v[n] - M * y[n] \tag{4.12}$$

In figure 4.6 an example of the EFM function is displayed. The following values are used [1]:

fractional input: $0,625$

number of bits: $n_0 = 3$

input: $x = M * 2^{n_0} = 5$

initial condition: $s[0] = 0$

clk	x	s	e	v	y
0	5	0	5	5	0
1	5	5	2	10	1
2	5	2	7	7	0
3	5	7	4	12	1
4	5	4	1	9	1
5	5	1	6	6	0
6	5	6	3	11	1
7	5	3	0	8	1
8	5	0	5	5	0
9	5	5	2	10	1
10	5	2	7	7	0

Figure 4.6: Function of a three bit accumulator [1]

Figure 4.6 shows that the period of the output y is eight clock cycles. Hence the average output is $\frac{5}{8} = 0,625$ with a minimum step size from $\frac{1}{8}$.
To prove the noise shaping function of the structure the quantizer is replaced by an additive noise source, like in figure 4.7 and the linear z-domain output $Y(z)$ is calculated. The calculation starts at equation 4.13. The additive noise source is necessary to make a non-linear system linear for calculation.

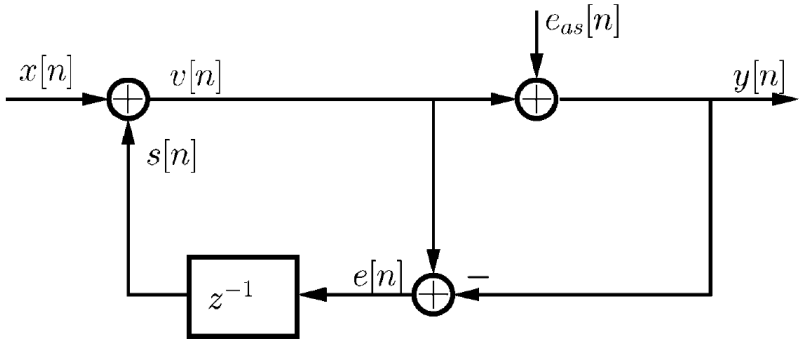

Figure 4.7: Accumulator model with additive quantization noise source [3]

$$V(z) = X(z) + S(z) \tag{4.13}$$

$$V(z) = X(z) + E(z)z^{-1} \tag{4.14}$$

$$E(z) = V(z) - [V(z) + E_{as}(z)] \rightarrow E(z) = -E_{as}(z) \tag{4.15}$$

$$Y(z) = X(z) + S(z) + E_{as}(z) \tag{4.16}$$

$$Y(z) = X(z) + E(z)z^{-1} + E_{as}(z) \tag{4.17}$$

$$Y(z) = X(z) - E_{as}(z)z^{-1} + E_{as}(z) \tag{4.18}$$

$$Y(z) = X(z) + (1 - z^{-1})E_{as}(z) \tag{4.19}$$

From 4.19 it can be observed, that the final output in the z-domain consists of the input signal and the high pass filtered quantization noise.

4.2.2 Multi Stage Noise Shaping - MASH

In order to further increase the noise shaping capabilities, a new structure called MASH DDSM is considered. In figure 4.8 a block diagram of this structure is displayed. It consists of l cascaded error feedback modulators and an error cancellation network (ECN). The negative error from the previous stage is fed forward to the input of the next stage.

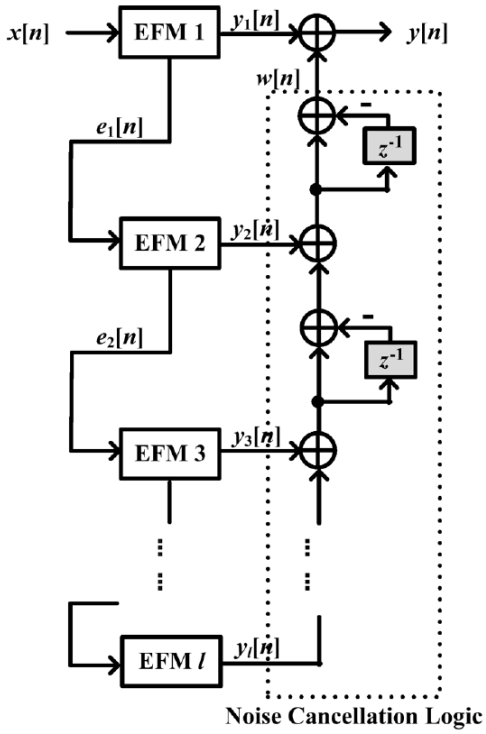

Figure 4.8: MASH DDSM model with l-stages [4]

The inputs to the ECN are the carry from each EFM. The network eliminates the quantisation noise from each stage except the last one. Thus the output in the z-domain consists of the input signal and the l-order shaped quantisation noise. The exact calculation is displayed starting with equation 4.20. For the calculation, the MASH order is selected as three and the quantizer of each stage is replaced by an additive quantization noise source.

$$Y_1(z) = X(z) + (1 - z^{-1})E_1(z) \tag{4.20}$$

$$Y_2(z) = -E_1(z) + (1 - z^{-1})E_2(z) \tag{4.21}$$

$$Y_3(z) = -E_2(z) + (1 - z^{-1})E_3(z) \tag{4.22}$$

$$Y(z) = Y_1(z) + (1 - z^{-1})Y_2(z) + (1 - z^{-1})^2 Y_3(z) \tag{4.23}$$

$$Y(z) = X(z) + E_1(z) - z^{-1}E_1(z) + (1 - z^{-1}) * (-E_1(z) + E_2(z) - z^{-1}E_2(z)) \tag{4.24}$$
$$+(1 - 2z^{-1} + z^{-2}) * (-E_2(z) + E_3(z) - z^{-1}E_3(z))$$

$$Y(z) = X(z) + E_1(z) - z^{-1}E_1(z) - E_1(z) + E_2(z) - z^{-1}E_2(z) + z^{-1}E_1(z) \tag{4.25}$$
$$-z^{-1}E_2(z) + z^{-2}E_2(z) - E_2(z) + E_3(z) - z^{-1}E_3(z) + 2z^{-1}E_2(z)$$
$$-2z^{-1}E_3(z) + 2z^{-2}E_3(z) - z^{-2}E_2(z) + z^{-2}E_3(z) - z^{-3}E_3(z)$$

$$Y(z) = X(z) + E_3(z) - 3z^{-1}E_3(z) + 3z^{-2}E_3(z) - z^{-3}E_3(z) \tag{4.26}$$

$$Y(z) = X(z) + (1 - z^{-1})^3 E_3(z) \tag{4.27}$$

In figure 4.9 the difference in the shaping property between the EFM and the MASH structure is displayed. The MASH structure has a more effective shaping function because of the 3rd order high pass filter but there is still the problem with short cycle lengths. Two different methods are known to eliminate this problem.

Stochastic approach: The idea is to use a random sequence called dither and add it to the LSB of the MASH input. Because of this random sequence the cycles should break up and the cycle length increases.

Deterministic approach: This approach tries to maximise the sequence length by design. One method is to use odd initial conditions. In [5] this idea is discussed and the conclusion is that the sequence length can be maximised by using an odd initial condition in the internal register of the first stage. Another method to increase the sequence length is to increase n_0 [3].

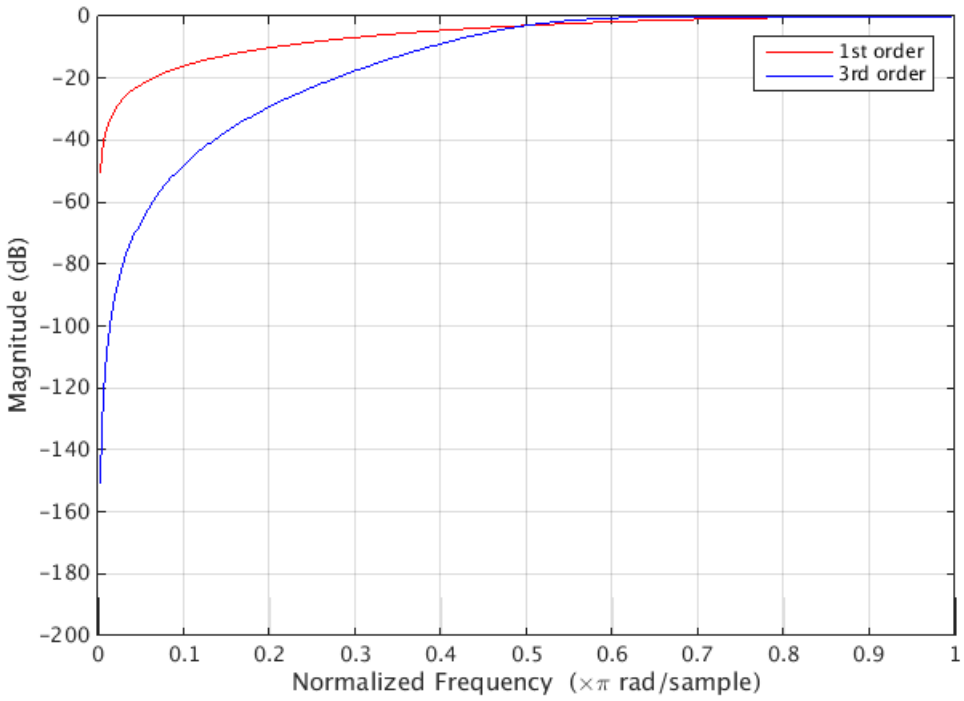

Figure 4.9: 1st order vs. 3rd order high pass filter

In the following section a method to compare different realisations in a graphical way is presented.

4.2.3 Power Spectral Density - PSD

The PSD describes the power distribution of a signal at different frequencies across the whole frequency spectrum. In literature it is the common way to judge about the quality of a DDSM. For the comparison of DDSM structures the DC part of the spectrum is removed because it is the constant input value and only the shaped error is an indicator for the quality of a method. In figure 4.10 an example is plotted. The spectrum should look like a smooth curve and should not have any spikes.

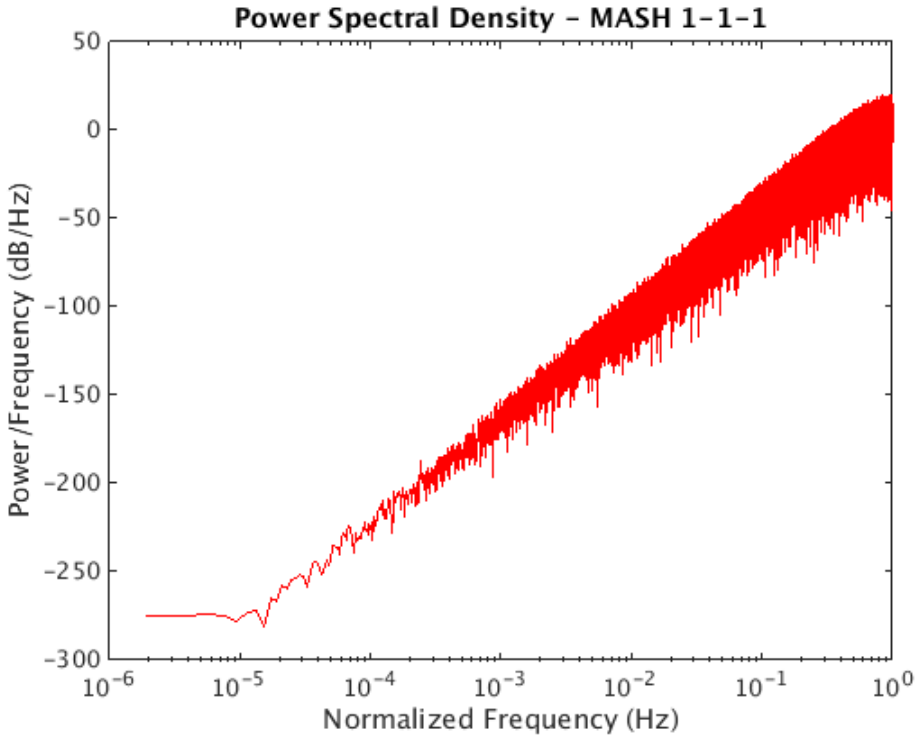

Figure 4.10: PSD example

4.2.4 Calculation of Sequence Lengths

Each DDSM with a constant input produces a periodic output. The shorter this sequence length is the worse the PSD of the output becomes (high power spurious peaks). For MASH structures up to three stages a formula to calculate the sequence length dependent on the input value, the number of bits and the initial conditions is developed in [6] and displayed in figure 4.11. The sequence length for stage one is maximal M when X and M are prime numbers and the

worst case occurs when the input $X = \frac{M}{2}$. For the second stage there are two different formulas depending on the sequence length from the first stage. $GCD(M, X)$ means the greatest common divisor between M and X.

Stage i	Sequence Length Lsi	Condition(s)
1	$\left(\frac{M}{\mathrm{GCD}(M,X)}\right)$	
2	$\left(\frac{M}{\mathrm{GCD}(M,Ls1s_1[0])}\right) \cdot \left(\frac{M}{\mathrm{GCD}(M,X)}\right)$	L_{s1} is odd
2	$\left(\frac{2M}{\mathrm{GCD}(2M,\lvert 2s_1[0]-X \rvert L_{s1})}\right) \cdot \left(\frac{M}{\mathrm{GCD}(M,X)}\right)$	L_{s1} is even
3	$\left(\frac{2M}{\mathrm{GCD}(2M,\lvert 2s_2[0]-s_1[0] \rvert L_{s2})}\right) \cdot \left(\frac{2M}{\mathrm{GCD}(2M,\lvert 2s_1[0]-X \rvert L_{s1})}\right) \cdot \left(\frac{M}{\mathrm{GCD}(M,X)}\right)$	L_{s2} is even,L_{s3} is divisible by 4 but not by 3.

Figure 4.11: Formulas to calculate sequence length [6]

In figure 4.12 an example calculation for the sequence length of the third stage from a MASH 1-1-1 DDSM with different input values from [6] is presented. The used values are: $M = 2^4$, $s_1[0] = 8$ and $s_2[0] = 0$. The maximum sequence length for this example is $2M$ and the worst case occurs when $X = \frac{M}{2}$.

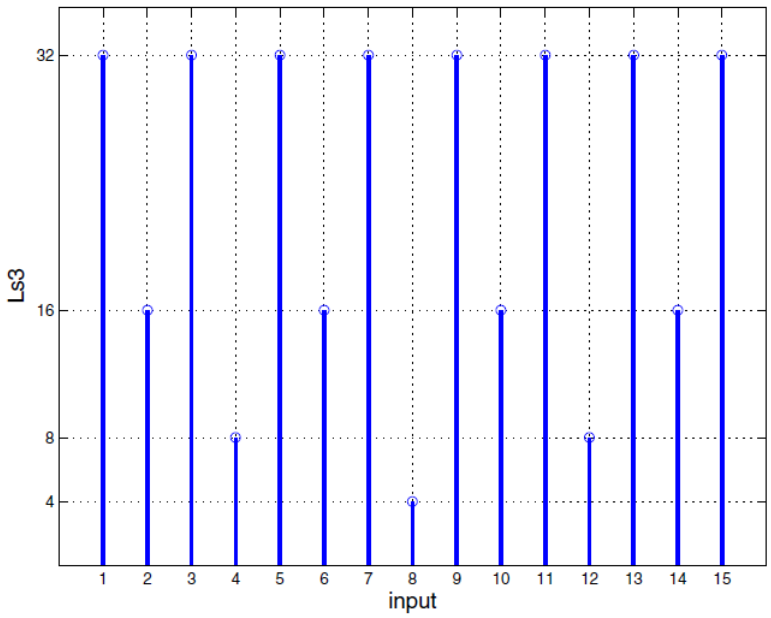

Figure 4.12: Calculated sequence lengths for stage three [6]

4.3 Currently Used Structure

At the moment a MASH 1-1-1 DDSM structure with 21 bit and odd initial condition at the first stage is used. In figure 4.10 the PSD of the structure for an input value $x = 0.4321$ with even initial conditions ($s_1[0] = s_2[0] = s_3[0] = 0$) is plotted. The PSD has a good form, like a smooth shaped curve. In figure 4.13 the blue curve is the PSD of the same structure. The only difference is the input value. It is changed to $x = 0.5$. For this value the PSD spectrum is not the theoretically predicted, showing very high power spurs. One way to improve it is to use an odd initial condition for the register of the first stage ($s_1[0] = 1, s_2[0] = s_3[0] = 0$), then the PSD changes to the red plot in figure 4.13. This behaviour is a big disadvantage of the currently used structure. One goal for the thesis is to find a structure that is independent from the input value and the initial condition.

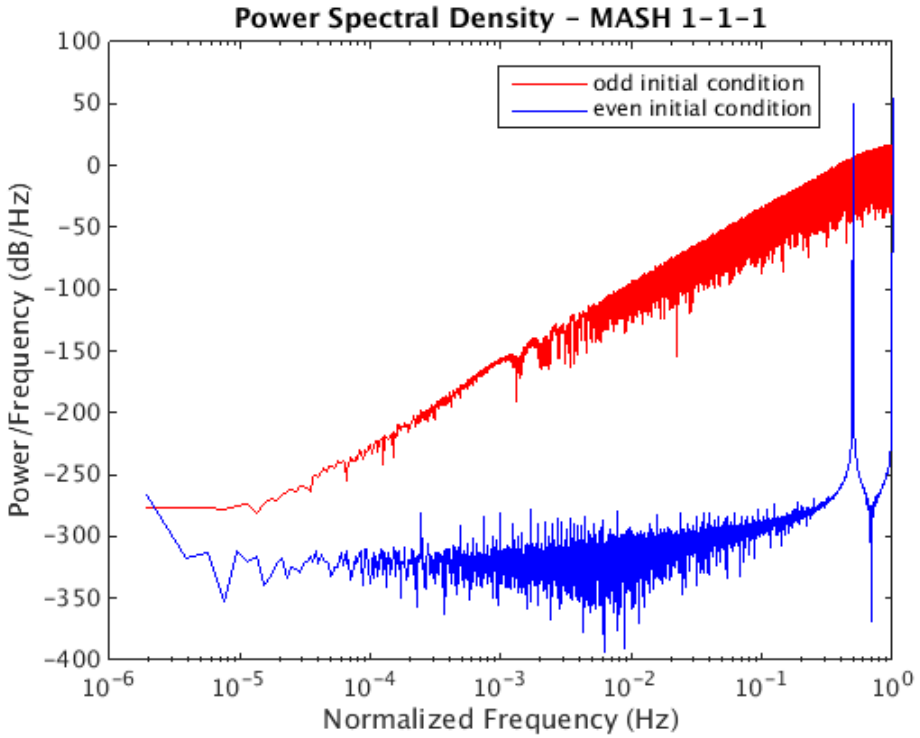

Figure 4.13: Comparison MASH 1-1-1 DDSM even and odd initial condition

4.4 Motivation for the Thesis

The currently used structure is a decade old (not state of the art), and consumes about 80% of the power from the whole digital part. Also the required area is big and it would be advantageous if it is possible to reduce it. Since the first implementation of the DDSM structure a lot

of different papers concerning saving power and area are published. The main goal for this work is to find a structure with reduced power consumption and area with a performance at least as good as the one from the currently used implementation. A secondary goal would be for the desired structure to work independently from the input value and initial conditions.

In the following section a list of all tasks concerning the master thesis is presented.

4.5 Definition of Tasks

- Background reading
- Review literature
- Matlab Models of the current situation and the improvements found in literature
- Evaluation
- Implementation in System Verilog
- Synthesis
- Power vs. Area new / original implementation
- Literature review about telemedicine

5 Summary of the Reviewed Papers

In chapter 4 a short overview about each reviewed paper is provided.

5.1 Hardware reduction in digital delta-sigma modulators via error masking - Part I: MASH DDSM [7]

In the researched paper a method to reduce complexity for maximised cycle length and how error masking can be used to reduce hardware consumption for MASH DDSM is presented. The proposed DDSM has the same cycle length and a bit better power spectrum because of the more effective whitening of the quantization noise due to interstage quantizers compared to the traditional MASH DDSM. Additionally it is possible to save about 20% hardware.

5.1.1 Reduced Complexity - RC

In this approach only the first accumulator has the full bit width N. The input to the second stage has a width M and the input to the third stage has a width L. Equation 5.1 always has to be fulfilled. Through this approach the word length starting from stage two can be reduced without reducing the cycle length. The cycle length of the quantization noise for this structure is 2^N.

$$N > M > L \tag{5.1}$$

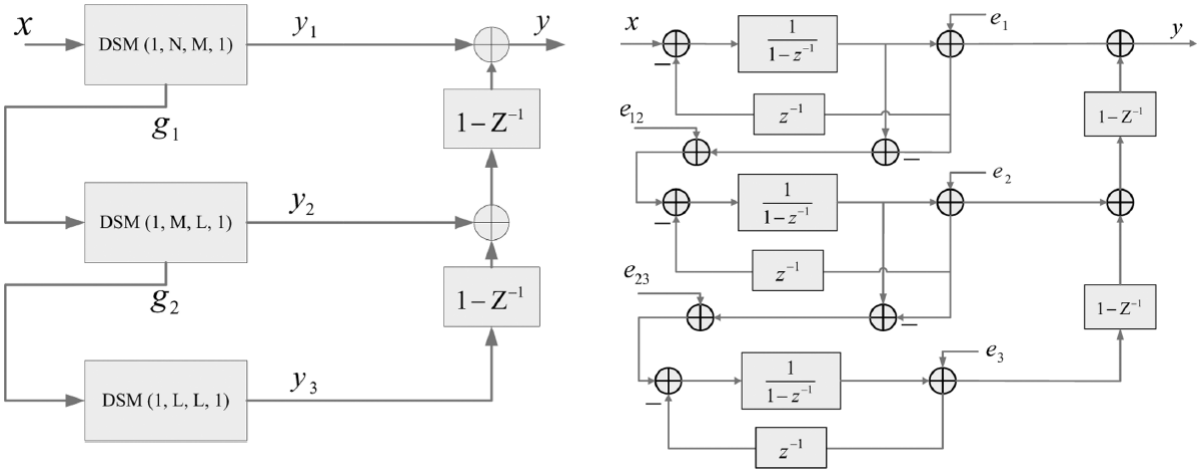

Figure 5.1: RC MASH 1-1-1 DDSM and model of an RC MASH 1-1-1 DDSM [7]

In figure 5.1 a model of the structure is plotted. To get from N to M bit and from M to L bit after each stage an additional interstage quantizer is necessary. One disadvantage of this structure is that through the interstage quantizers additional noise is added to the output. The noise cancellation network is not able to remove it. The output in the z-domain changes to equation 5.2.

$$Y(z) = X(z) + (1 - z^{-1})^3 E_3(z) + z^{-1}(1 - z^{-1}) E_{12}(z) + (1 - z^{-1})^2 E_{23}(z) \tag{5.2}$$

Equation 5.2 can also be written as:

$$Y(z) = X(z) + N_3(z) + N_{12}(z) + N_{23}(z) \tag{5.3}$$

When M and L are chosen conveniently it is possible to mask the additional noise from the interstage quantizers behind the contribution from the third stage. The idea is if the levels of the lowest frequency tones from N_{12} called $f_{12} = \frac{f_s}{2^{N-M}}$ and N_{23} named $f_{23} = \frac{f_s}{2^{N-L}}$ are below N_3 they are always below N_3 because N_{12} and N_{23} are only first and second order shaped. The strategy is displayed in figure 5.2. To ensure error masking the following three points have to be fulfilled:

- Choose $N = N_0 + 1$ to have the same cycle length as in the N_0-bit traditional MASH DDSM.
- Choose $M = ceil((4N - 10, 6)/5))$ - ceil(x) means the smallest integer greater than x
- Choose $L = ceil((2N - 5, 3)/3)$

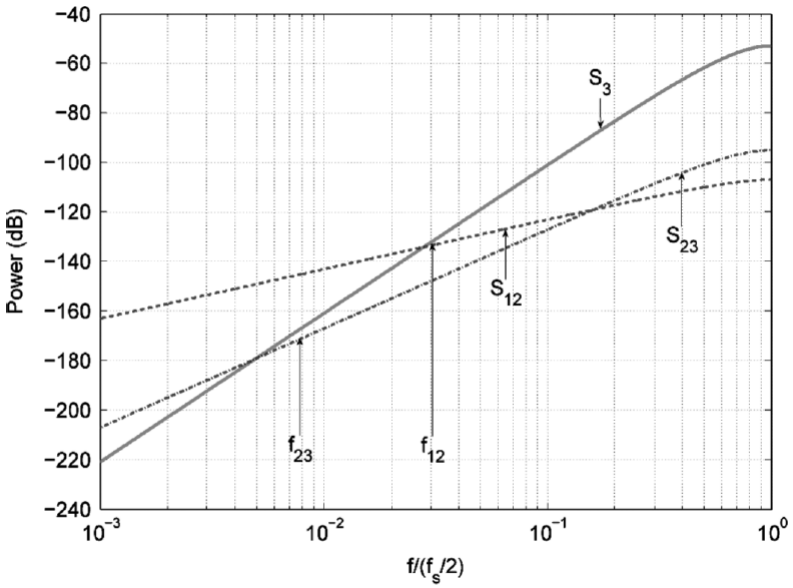

Figure 5.2: Error masking strategy [7]

5.1.2 Adding Dither to the System

In this case a 1-bit pseudo random shaped signal is added to the LSB of the input signal (like displayed in figure 5.3). The minimum cycle length is the cycle length of the dither signal.

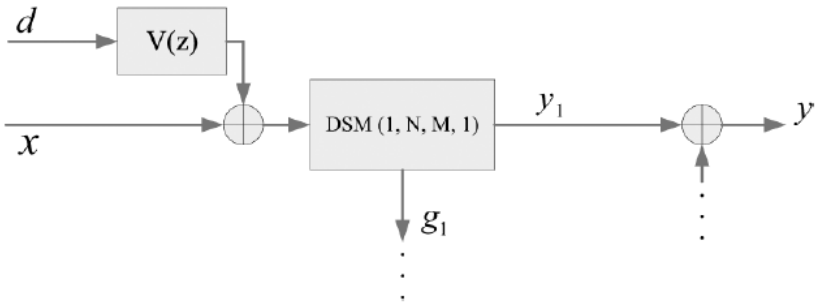

Figure 5.3: Adding dither to MASH DDSM [7]

When dither is used it is hard to find the first tone of e_{12} and e_{23} because the cycle length is very long. In this case another strategy is used. Its has to be distinct between Zeroth-Order and First-Order Dither.

Zeroth-Order Dither:

The low frequency noise floor of a DDSM with dithering is dominated by the dither signal. The largest frequency f_0 at which the PSD of the noise floor is higher than the contribution from N_3 can be calculated. At this point the PSDs from N_{12} and N_{23} have to be lower than the noise floor. For better understanding in figure 5.4 an example is plotted.

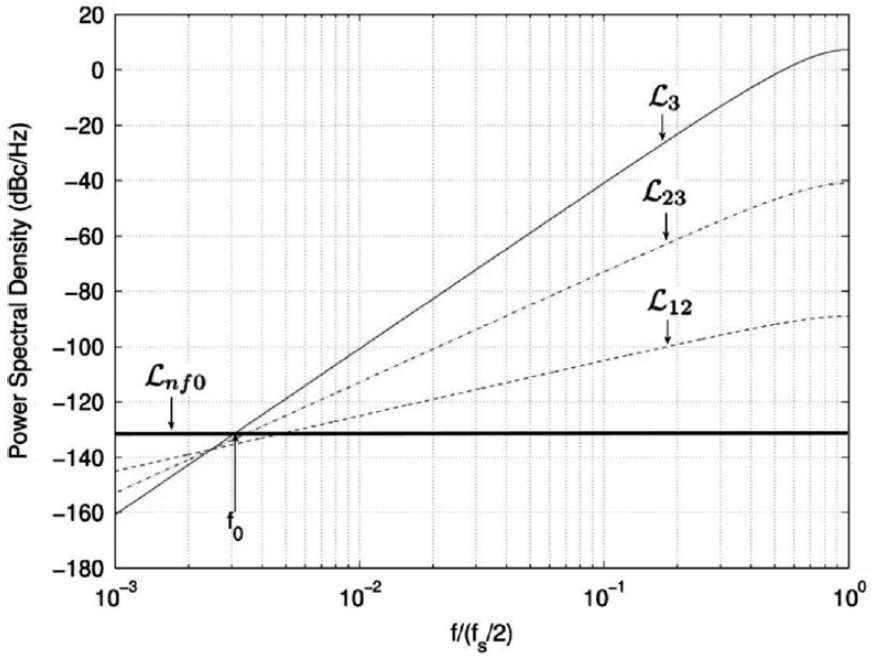

Figure 5.4: Error masking strategy with dither - Zeroth-Order [7]

To realise the error masking strategy equations 5.4 and 5.5 have to be fulfilled. With Zeroth-Order dither it is possible to save about 33% hardware for the DDSM but additional hardware for the dither generator is needed.

$$M = ceil\left(\frac{2N}{3}\right) \tag{5.4}$$

$$L = ceil\left(\frac{N}{3}\right) \tag{5.5}$$

First-Order Dither:

In this case the dither signal is first order high pass filtered. To mask the contribution from N_{12} under the first order shaped noise floor $N = M$ has to be true. The largest frequency at which the PSD from dither is larger than that form e_3 can be calculated. At this frequency the PSD from e_{23} has to be smaller than the PSD from dither to mask the contribution from interstage quantizers. In figure 5.5 an example is plotted.

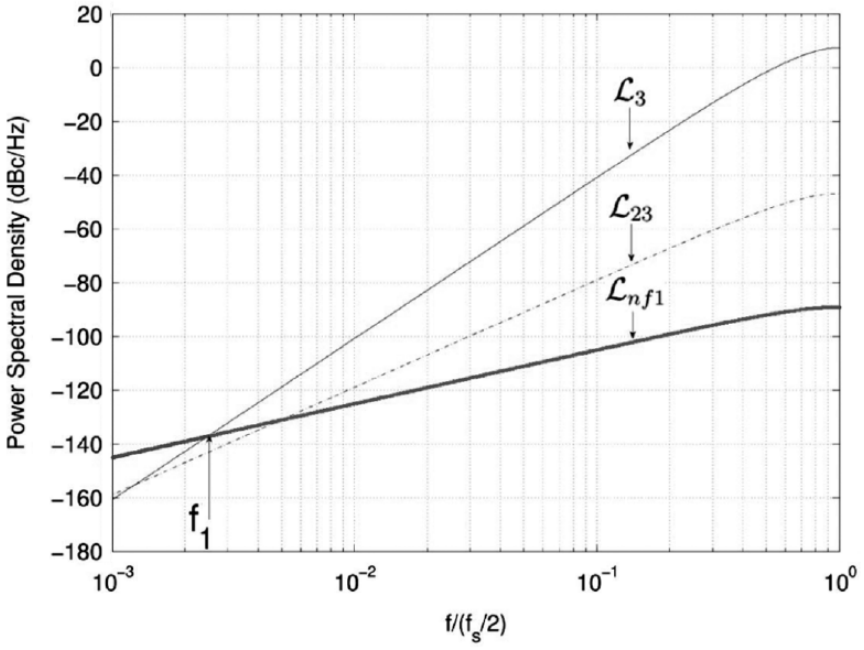

Figure 5.5: Error masking strategy with dither - First-Order [7]

To realise the error masking strategy equation 5.6 has to be fulfilled. With First-Order dither it is possible to save about 17% hardware for the DDSM. In this calculation the necessary hardware for the dither generator is not considered. When comparing the PSD from both dither cases the noise floor for First-Order dither is lower as for Zeroth-Order dither.

$$L > \frac{N+1}{2} \tag{5.6}$$

5.2 A novel implementation of dithered digital delta-sigma modulators via bus-splitting [8]

In this section a method to split up the input word in two or more parts and handle it with different order DDSMs for dithered DDSMs is introduced.

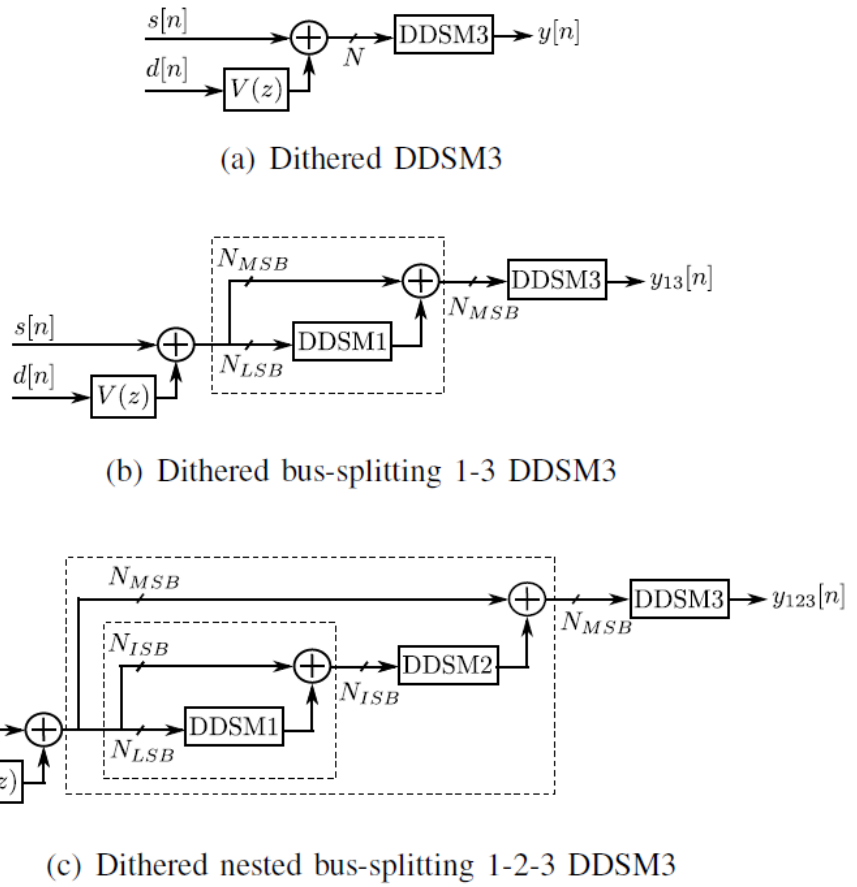

Figure 5.6: Several combinations of dither and bus-splitting [8]

The added dither signal is filtered with a high pass filter $V(z)$. To get a uniformly distributed and input independent quantization noise $V(z) = (1 - z^{-1})^R$ with $R \leqslant l - 2$ where l is the order of the MASH DDSM has to be true. For MASH 1-1-1 DDSM zeroth and first order dither can be used. In figure 5.6 (b) and (c) two different methods to split the input value are displayed. In (b) the input is split into two parts, N_{LSB} and N_{MSB}. N_{LSB} is processed with a first order MASH DDSM, again combined with N_{MSB} and processed with a third order MASH DDSM. In (c) the input is split into three parts, N_{LSB}, N_{ISB} and N_{MSB}. N_{LSB} is processed with a first order MASH DDSM, combined with N_{ISB}, processed with a second order MASH DDSM, combined with N_{MSB} and processed with a third order MASH DDSM.

To realise this method it is also necessary two hide the noise from the additional DDSMs behind the contribution from the third order MASH DDSM. Two different cases have to be distinguished:

Zeroth-Order Dither:

How to calculate the different wordlengths depending on the splitting architecture is displayed in figure 5.7. The values M and L are calculated as follows:

- $M = \left\lceil \frac{2N}{3} \right\rceil$
- $L = \left\lceil \frac{N}{3} \right\rceil$

Nested DDSM	Wordlengths		
	N_{LSB}	N_{ISB}	N_{MSB}
(a) 1-3	N-M	-	M
(b)1-2-3	N-M	M-L	L

Figure 5.7: Wordlengths for bus-splitting [8]

First-Order Dither:

In this case $N_{LSB} = 0$ because dither is first order shaped like the noise floor. As a conclusion the first order MASH DDSM in the 1-2-3 MASH 1-1-1 DDSM is not necessary. The other values have to be chosen as follows:

- $N_{MSB} = \left\lceil \frac{N}{2} \right\rceil$
- $N_{ISB} = N - N_{MSB}$

The conclusion of this approach is to get a PSD similar to that of a traditional MASH 1-1-1 DDSM with possible hardware saving around 20% dependent on the chosen splitting architecture for Zeroth-Order dither. In this calculation the necessary hardware for dither is excluded.

5.3 Hardware simplification to the delta path in a MASH 111 delta-sigma modulator [9]

The idea is to reduce the necessary hardware for the delta path. In this approach the error cancellation network is called delta path, displayed in figure 5.8. To realise the idea the 1-bit carry output signal from each stage should be treated as -1 instead as 1 and twos complement arithmetic should be used. In the normal 1 case w from figure 5.8 has a value set of $-1, 0, 1, 2$ and y has a value set of $-3, -2, -1, 0, 1, 2, 3, 4$. In twos complement three and four bits are needed for the value sets. In the -1 case w has a value set of $-2, -1, 0, 1$ and y has a value set of $-4, -3, -2, -1, 0, 1, 2, 3$. For these value sets only two and three bits are needed. The only difference in the model of this structure compared to the conventional MASH 1-1-1 DDSM is the sign of the output signal. Because of this and the fact the hardware saving with this approach for clock rate equal 26 MHz is only about 2 percent in chapter 5 no model for this idea is presented.

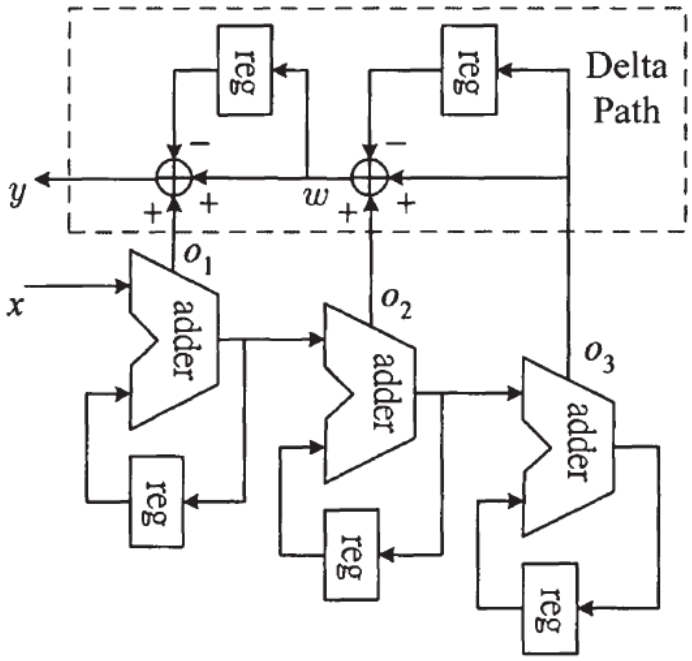

Figure 5.8: MASH 1-1-1 DDSM with marked delta path [9]

In following a short excursion about twos complement is given.

Excursion - Twos Complement:

With twos complement it is possible to get negative binary numbers. In following an example with 4-bits is presented. Normally it is possible to represent values from 0 to 15 with 4-bits. The first position in binary counts 1 in decimal, the second 2, the third 4 and the fourth 8 like in equation 5.7 displayed. In twos complement case with 4-bits values from -8 to 7 can be represented. The decimal value for the last stage counts negative. 8 gets to -8. The rest is the same as for normal binaries.

$$0000 \rightarrow 0_8 0_4 0_2 0_1 \rightarrow 0 - 15 \tag{5.7}$$

$$0000 \rightarrow 0_{-8} 0_4 0_2 0_1 \rightarrow -8 - 7 \tag{5.8}$$

5.4 Maximum Sequence Length MASH Digital Delta-Sigma Modulators [3]

In [3] a method to maximise the cycle length through a modified EFM for all input values and initial conditions is presented. The resulting PSD is comparable to the dithered one with better low frequency performance. The modified structure is displayed in figure 5.9. The difference to the traditional structure is the additional feedback loop with the αz^{-1} block. In 5.9 the calculation of α is displayed. a is a small integer depending on the word length of the accumulator. A list of different values for a is displayed in figure 5.10. When a bigger word length is used a can be calculated as the difference between M and the closest prime number less than M. When $a = 1$ the carry-in input of the accumulator can be used to add the delayed signal to the input. Otherwise hardware for an additional adder with $n_0 + 1$-bit is needed. The inputs to this adder are the summation result from $x[n]$ and $s[n]$ and the returned output times a.

$$\alpha = \frac{a}{2^{n_0}} \tag{5.9}$$

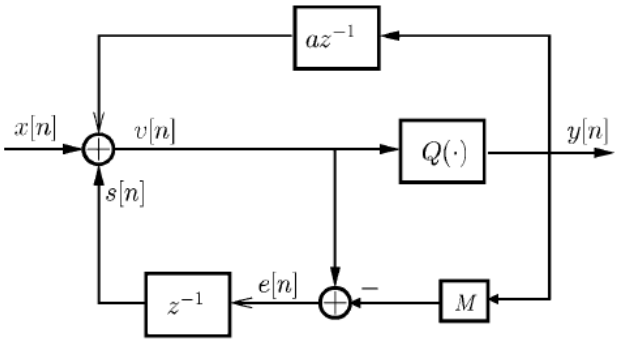

Figure 5.9: Structural changes in the EFM - HK MASH DDSM [3]

n_0	a
5, 7, 13, 17, 19	1
6, 9, 10, 12, 14, 20, 22, 24	3
8, 18, 25	5
11, 21	9
16, 23	15
15	19

Figure 5.10: Different values for a depending on the accumulator word length [3]

In the following equation the z-domain output for the modified EFM is calculated. Therefore all the signals in figure 5.9 are normalized to one and the quantizer is replaced by an additive noise source.

$$Y(z) = V(z) + E_{as}(z) \tag{5.10}$$

$$E(z) = V(z) - Y(z) \tag{5.11}$$

$$E(z) = -E_{as}(z) \tag{5.12}$$

$$V(z) = X(z) + \alpha z^{-1}Y(z) + z^{-1}E(z) \tag{5.13}$$

$$V(z) = X(z) + \alpha z^{-1}Y(z) - z^{-1}E_{as}(z) \tag{5.14}$$

$$Y(z) - E_{as}(z) = X(z) + \alpha z^{-1}Y(z) - z^{-1}E_{as}(z) \tag{5.15}$$

$$Y(z) = \frac{X(z)}{1 - \alpha z^{-1}} + \frac{1 - z^{-1}}{1 - \alpha z^{-1}}E_{as}(z) \tag{5.16}$$

With the new designed EFM a MASH system can be built. The system is called HK-MASH and displayed in figure 5.11.

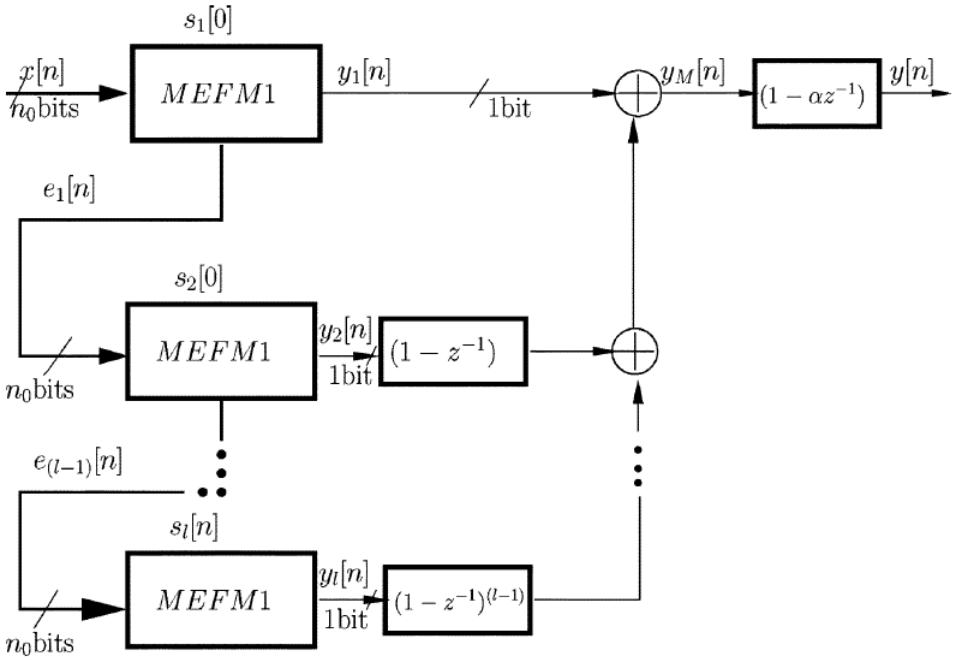

Figure 5.11: HK-MASH DDSM [3]

The z-domain output of this structure is:

$$Y(z) = \frac{X(z)}{1-\alpha z^{-1}} + \frac{(1-z^{-1})^l}{1-\alpha z^{-1}} E_{asl}(z) \tag{5.17}$$

$$Y(z) = X(z) + (1-z^{-1})^l E_{asl}(z) \tag{5.18}$$

Through the additional $(1-\alpha z^{-1})$ filter at the output of the HK-MASH equation 5.17 gets to 5.18. In [3] they came to the conclusion that the introduced error by removing the additional filter is negligible. With the HK-MASH DDSM it is possible to get a cycle length of $(2^{n_0}-a)^l$ for all constant inputs and initial conditions.

5.5 Spur-free MASH delta-sigma modulation [4]

In this paper self made dither is used to break up the periodicity of the output. The MASH structure is changed. The carry output and the error of a stage are added and feed forward to be the input of the next stage. The structure works perfect for all input and initial conditions and is plotted in figure 5.12.

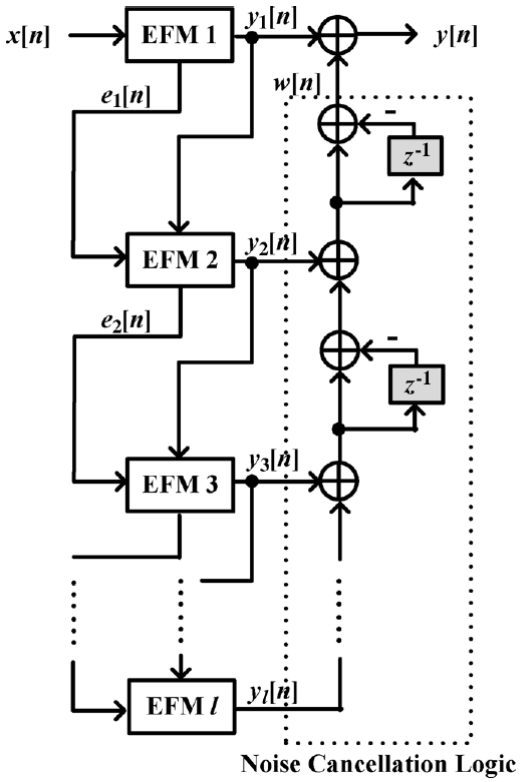

Figure 5.12: Modified MASH structure [4]

The output in the z-domain is calculated in the following equations:

$$Y_1(z) = \frac{1}{M}X(z) + \frac{1}{M}(1-z^{-1})Q_1(z) \tag{5.19}$$

$$Y_2(z) = \frac{1}{M}(Y_1(z) - Q_1(z)) + \frac{1}{M}(1-z^{-1})Q_2(z) \tag{5.20}$$

$$Y_l(z) = \frac{1}{M}(Y_{l-1}(z) - Q_{l-1}(z)) + \frac{1}{M}(1-z^{-1})Q_l(z) \tag{5.21}$$

$$Y(z) = Y_1(z) + (1-z^{-1})Y_2(z) + ... + (1-z^{-1})^{l-1}Y_l(z) \tag{5.22}$$

$$\begin{aligned} Y(z) = \{&\frac{1}{M} + \frac{1}{M^2}(1-z^{-1}) + ... + \frac{1}{M^l}(1-z^{-1})^{l-1}\} \\ &X(z) + (1-z^{-1})^l\{\frac{1}{M}Q_l(z) + \frac{1}{M^2}Q_{l-1}(z) + ... + \frac{1}{M^l}Q_1(z)\} \end{aligned} \tag{5.23}$$

$$Y(z) = \frac{1}{M}X'(z) + \frac{1}{M}(1-z^{-1})^l Q'(z) \tag{5.24}$$

The signal transfer function from equation 5.23 consists of one all pass filter and several high

pass filters. For a constant input the high pass filtered parts do not effect the output of the modulator and can be neglected. The noise transfer function from equation 5.23 is exact a high pass filter with variable order depending on the number of stages of the MASH DDSM like the one in former structures. For the proposed structure no additional hardware is needed, because the additional forwarded signal only has one 1-bit and the carry-in input from the accumulator can be used to add the signals together.

5.6 Hardware Reduction of MASH Delta-Sigma Modulator Based on Partially Folded Architecture [10]

In this section a method to reduce the hardware consumption for MASH DDSM is presented. The method reuses adders and a conventional MASH is combined with a folded MASH to keep the noise shaping capability, like displayed in figure 5.15. The idea is to split the r-bit input into two parts like in equation 5.25, process this parts with the same structure and generate an output carry. If the parts are processed alternating every two clock cycles an output is generated. Thereby the sequence length doubles compared to the traditional MASH DDSM. When the r-bits are split into two parts with equal number of bits hardware consumption can be divided by two. In figure 5.13 the modified EFM and MASH DDSM are plotted. By the reason of every two cycles the EFM generates a carry, additional registers are needed like in figure 5.13 displayed. To realise the every two clock cycles output a finite state machine, like a toggle flip-flop and a multiplexer are needed. If 0 is selected on the multiplexer the $x_{LSB}[n]$ part is processed otherwise (if 1 is selected) the $x_{MSB}[n]$ part is processed.

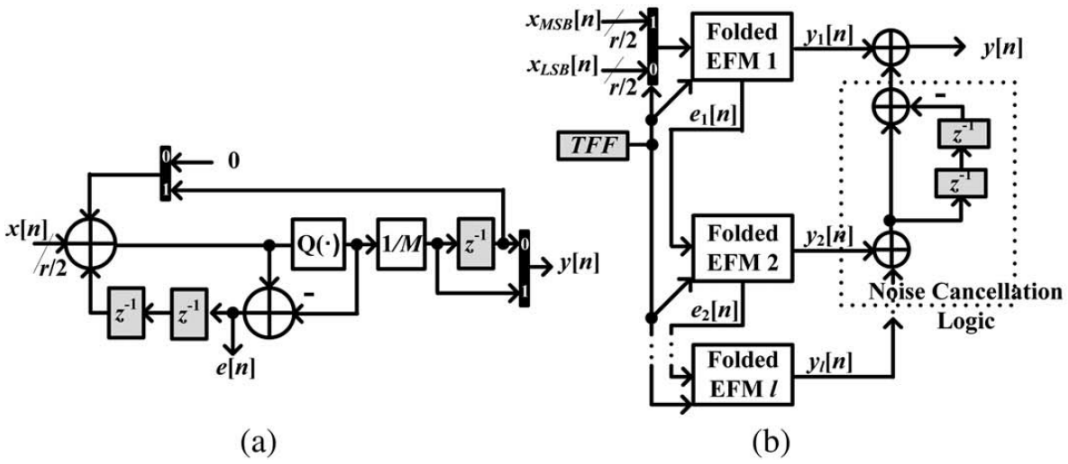

Figure 5.13: Folded EFM and folded MASH structure [10]

$$x_R[n] = x_{MSB}[n]2^i + x_{LSB}[n] \tag{5.25}$$

The z-domain output of the modified EFM is displayed in equation 5.26.

$$Y(z) = \frac{1}{M}X_R(z) + \frac{1}{M}(1 - z^{-2})Q(z) \tag{5.26}$$

In figure 5.13 (b) the folded EFM is combined to a MASH structure. For the whole MASH only one toggle flip-flop is necessary, but the number of registers has to double because the noise of each stage should be eliminated through the noise cancellation network. In equation 5.27 the z-domain output for the MASH is displayed.

$$Y(z) = \frac{1}{M}X_R(z) + \frac{1}{M}(1 - z^{-2})^l Q_l(z) \tag{5.27}$$

When the sequence length is still too short it is possible to combine the folded EFM with the SP - EFM and generate the structure displayed in figure 5.14. The resulting sequence length is double the length of SP-MASH DDSM. The minimum length changes from 2^{2r+1} to 2^{2r+2} for the 3rd order system with r-bit.

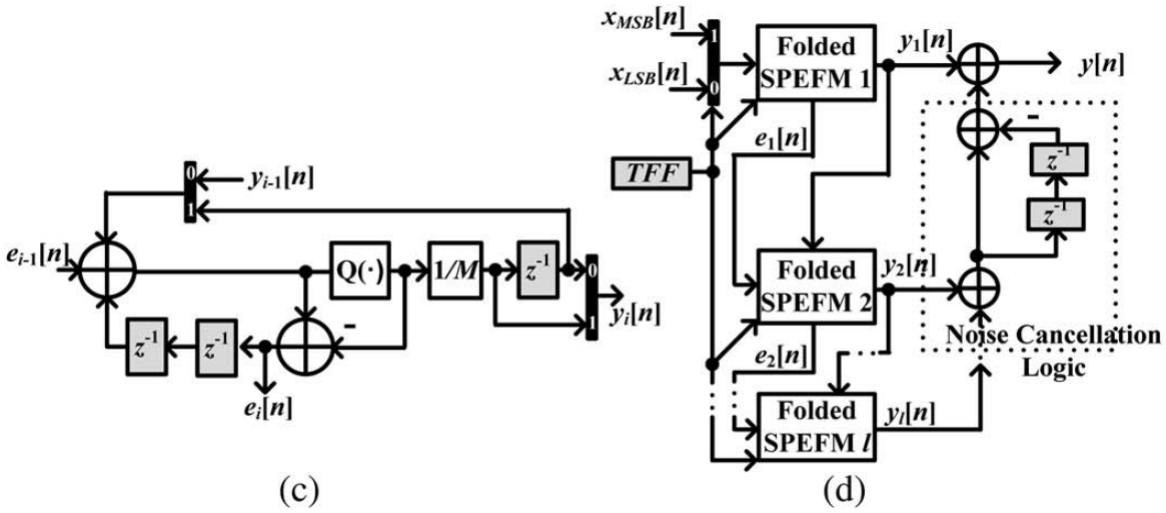

Figure 5.14: Folded SP-EFM and folded SP-MASH structure [10]

The proposed structure can not be used standalone. A combination of the folded and the traditional structure is necessary because the noise transfer function of the folded structure has a larger magnitude at low frequencies. The reason for this is the squaring of z. Through the traditional MASH a down scaling is realized. The resulting power spectrum is comparable to that of the ideal MASH DDSM. The scaling factor is $\frac{1}{2^m}$ where m is the number of bits processed with the traditional MASH displayed in figure 5.15.

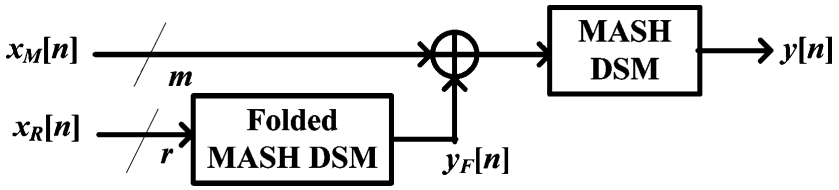

Figure 5.15: Partially folded MASH DDSM [10]

6 Models

In this chapter all realised models based on the papers from chapter 4 are presented and discussed. For the realisation the programs Matlab (The Mathworks Inc., Natick, USA) and Simulink (The Mathworks Inc., Natick, USA) are used.

6.1 First Steps

The first model was build in Matlab. Starting with a simple EFM, a plotting function, an ECN and a main file to combine and control the different functions. The first goal was to build a model of the currently used structure (MASH 1-1-1 with 21-bit and odd initial condition). In the main file control parameters like MASH order, input value and number of bits are defined. Additional error and carry from the EFM are fed forward to the ECN, the output from the ECN is fed forward to the plotting function and the average output is calculated. In following the Matlab code of the model is plotted.

MASH 1-1-1 DDSM:

```
clear;
format long
title = 'Power Spectral Density - conventional MASH 1-1-1'
mashOrder = 3;
sampleSize = 2^21;
busSize = 21;
fraction = 0.4321
fractionIntern = 2^busSize*fraction;
accumulatorBits = 21;
accumulatorSize = 2 ^ accumulatorBits;
e = zeros(mashOrder+1,sampleSize);
c = zeros(mashOrder,sampleSize);
e(1,:) = fractionIntern; %Input des MASH
y(1:sampleSize) = 0;
```

```
for index = 1 : mashOrder
[e(index+1,:),c(index,:)] = DSM1(e(index,:), sampleSize, accumulatorSize);
end

y = errorCancellationNetwork(e, c, sampleSize, mashOrder);
plotPSD(y, sampleSize, title);
meanFrac=mean(y)
```

In the plotting function the PSD is calculated in terms of equation 6.1. The resulting spectrum is cut in the middle because of its symmetry. In [19] an example for calculating the PSD is provided. For plotting the function *semilogx* is used, because it provides a base 10 logarithmic x-axis. As y-value for the *semilog* function 10 times the logarithms instead of 20 times is used because power and not voltage is calculated. In figure 6.1 the generated output is plotted.

N: number of samples
y: output of the DDSM

$$S_{xx}(\omega) = \frac{2}{N^2} \sum_{n=0}^{N-1} |DFT(y)_n|^2 \tag{6.1}$$

Plotting Function:

```
function [ ] = plotPSD( y, sampleSize, name )

figure(1);
N = length(y);
freq = 0:N/N:N/2;

ydft = fft(y);
ydft = ydft(1:N/2+1);

psdy = (1/((sampleSize*N)/4))*(abs(ydft)).^2;
psdy(2:end-1) = 2*psdy(2:end-1);

semilogx(freq/(N/2),10*log10(psdy),'b');
xlim([0.001 1]);
grid on;

title(name);
```

```
xlabel('Normalized Frequency (Hz)')
ylabel('Power/Frequency (dB/Hz)')
end
```

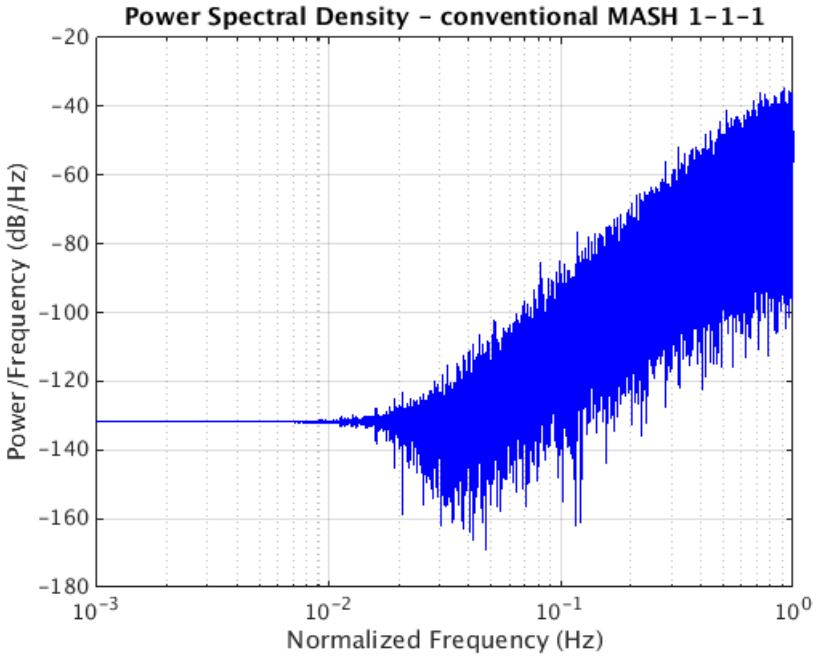

Figure 6.1: PSD - conventional MASH 1-1-1 - 21bit

In the EFM function the error and the carry for n-bit accumulator is calculated. For the first element a catch function is implemented.

EFM:

```
function [s,c] = DSM1( input, sampleSize, accumulatorSize )

s(1:sampleSize) = 0;
c(1:sampleSize) = 0;

for index = 1:sampleSize
    try
        s(index) = input(index) + s(index-1);
    catch ME
        if (strcmp(ME.identifier,'MATLAB:badsubscript'))
            s(index) = input(index);  %first element
        end
    end
```

```
    if s(index) >= accumulatorSize
        c(index) = 1;
        s(index) = s(index) - accumulatorSize;
    end

end
end
```

In the Error Cancellation Network function the carry for each stage and the carry for the whole 3rd order MASH DDSM are calculated.

ECN:

```
function [ y3 ] = errorCancellationNetwork(e, c, sampleSize, mashOrder )

y1(1:sampleSize) = 0;
y2(1:sampleSize) = 0;
y3(1:sampleSize) = 0;

for index = 3:sampleSize
    y1(index) = c(1,index);
    y2(index) = c(1,index) + c(2,index) - c(2,index-1);
    y3(index) = c(1,index) + c(2,index) - c(2,index-1)
    + c(3,index) - 2 * c(3,index-1) + c(3,index-2);
end
end
```

6.2 Method Change

After building the first model and being familiar with Matlab and the necessary attributes the working method changed from only using Matlab to a combination of Matlab and Simulink. In the new method for setting the parameters and calculating/plotting the PSD Matlab and for building and simulating the structure Simulink is used. This method is maintained for the rest of the thesis.

6.2.1 Plotting Function

Matlab has a function called *periodogram* to calculate the Power Spectral Density automatic. In literature using this function is the common way to calculate PSDs [1]. To enable easier

comparison between the built models and literature the plotting function changed a little. In this section the function is described and in the following used for all models to calculate and plot the PSD.

Plotting Function Version 1.1:

```
function [ ] = plotPSD( y, simulationTime, name, input, M )

window = hann(simulationTime);
figure(1);

[Pyy,w]=periodogram(y-input/M,window,simulationTime);
semilogx(w/(pi),10*log10(pi*Pyy),'r')
hold on

title(name);
xlabel('Frequency (Hz)')
ylabel('Power/Frequency (dB/Hz)')

end
```

The inputs to the *periodogram* function are as follows:

- DDSM output minus the input normalized to 1. The input is subtracted because the quality of a structure is only defined by the shaping behaviour of the quantization error.
- window used for calculation
- number of samples used for calculation (number of points for the DFT [[20])

The *hann* function is used for generating a window because by default the *periodogram* function uses a rectangular window which has a 13.3 dB side lobe attenuation and this could be the reason for masking spectral content below this value [20]. In figure 6.2 the difference in the PSD using a *rectangular* and a *hann* window is plotted.

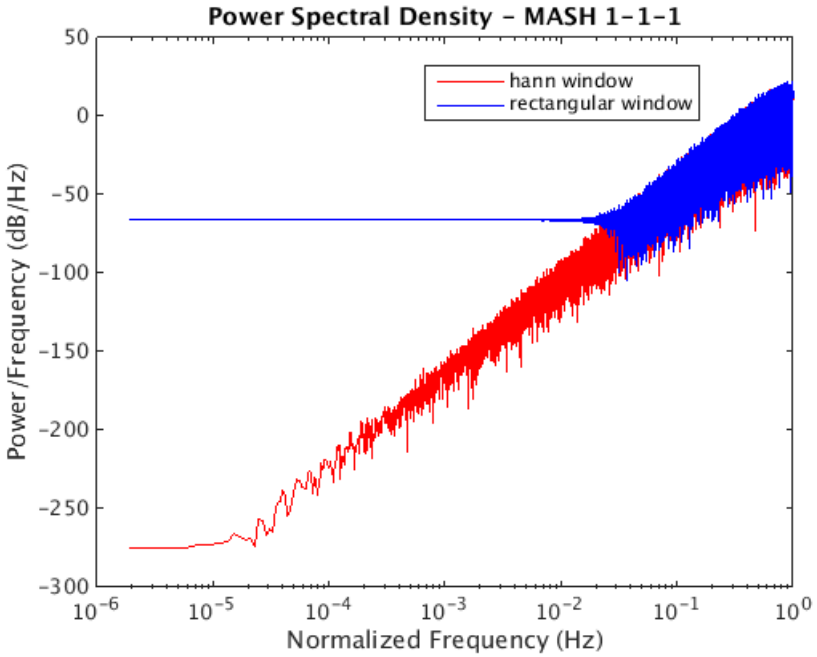

Figure 6.2: Difference between using rectangular and hann window for calculating the PSD

The return values of the *periodogram* function are the PSD and the normalized frequency vector which is used for scaling the plot. The frequency vector spans the interval $[0, \pi]$ [20] and has to be divided by π to get normalized to 1.

In the following section models of the different structures are presented. For simulation and the resulting PSD plots, an input value of 0.5 and an order of three for all MASH structures were chosen. The order is three because with a lower order the shaping properties of the structure are not comparable to the third order structure and with a higher order the structure becomes unstable. The chosen input value is 0.5 because it is a worst case value (see section 3.2.4). For the final decision also other values were used. The time average output value of each structure is always exactly the input value.

6.3 Conventional MASH

In figure 6.3 the Simulink model of the conventional MASH 1-1-1 DDSM is plotted. It consists of three first order EFMs, an input block, an output block, two adders and an ECN. The input block gets its value from the Matlab code and the output block returns its value back to the Matlab code. The content from the *errorCancellation* blocks is displayed in figure 6.4. They consist of a subtractor and a delay block. In figure 6.5 the content from the *DSM* blocks is displayed. It is the same structure as presented in chapter 3 except the 0.5 gain block. Without this

block the quantizer is not working correctly for some input values like $x = 0.25$, because the output of the quantizer is calculated with equation 6.2 [21]. The quantization interval is defined as M. The initial condition for each stage is defined in the delay block of each stage and set to zero.

y: output of the block

u: input of the block

q: quantization interval

$$y = q * round\left(\frac{u}{q}\right) \tag{6.2}$$

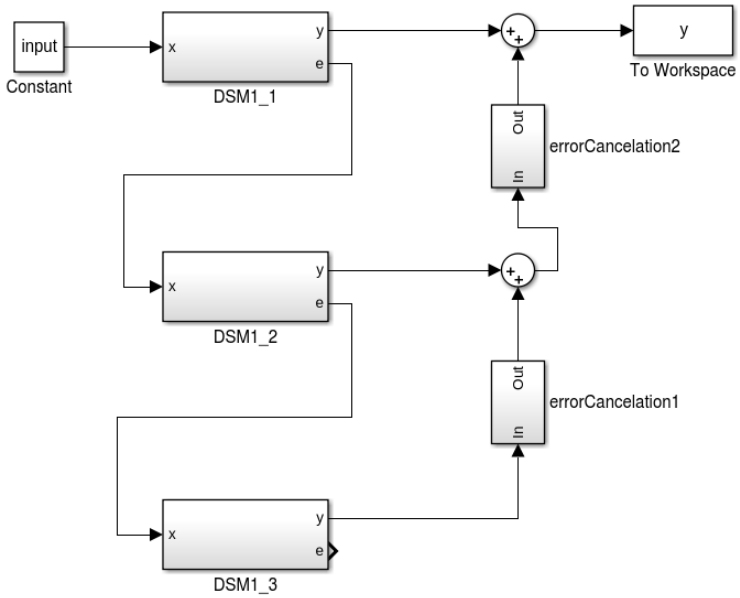

Figure 6.3: Simulink model of the conventional MASH DDSM

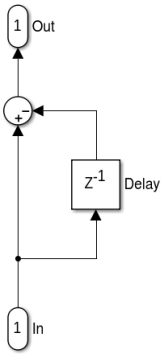

Figure 6.4: Simulink model of the ECN

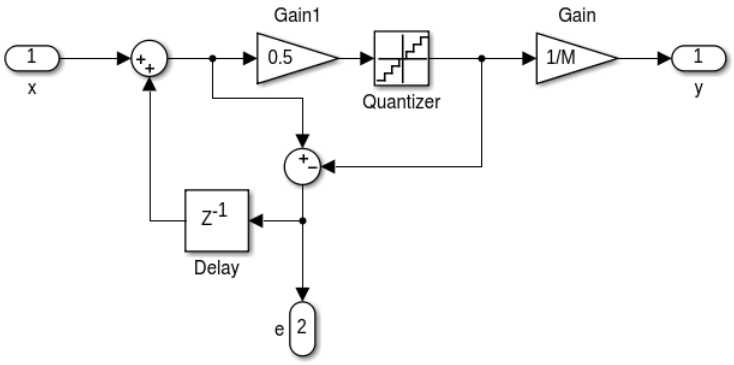

Figure 6.5: Simulink model of the conventional EFM

Following the Matlab code which controls the Simulink model is presented. The code sets the parameters and starts the simulation. Important is to use the *round* function because it scales the input to n_0-bit. At the beginning this was an error in the model, because the model had a value set of all possible integer values and that is not realistic, because in real life only values represented with 21-bit are available in the currently used structure.

Parameters Conventional MASH 1-1-1 DDSM:

```
clear
inputFractional = 0.5
n0 = 21;
M = 2^n0;
input = round(M*inputFractional);
title = 'Power Spectral Density - MASH 1-1-1';

%initial conditions
c1 = 0;
c2 = 0;
c3 = 0;

simulationTime = 2^20;
sim('MASH_111.slx',[1 simulationTime]);
plotPSD(y, simulationTime, title, input, M);
meanFrac = mean(y)
```

In figure 6.6 the resulting PSD for the above presented structure with even initial condition for all stages is presented.

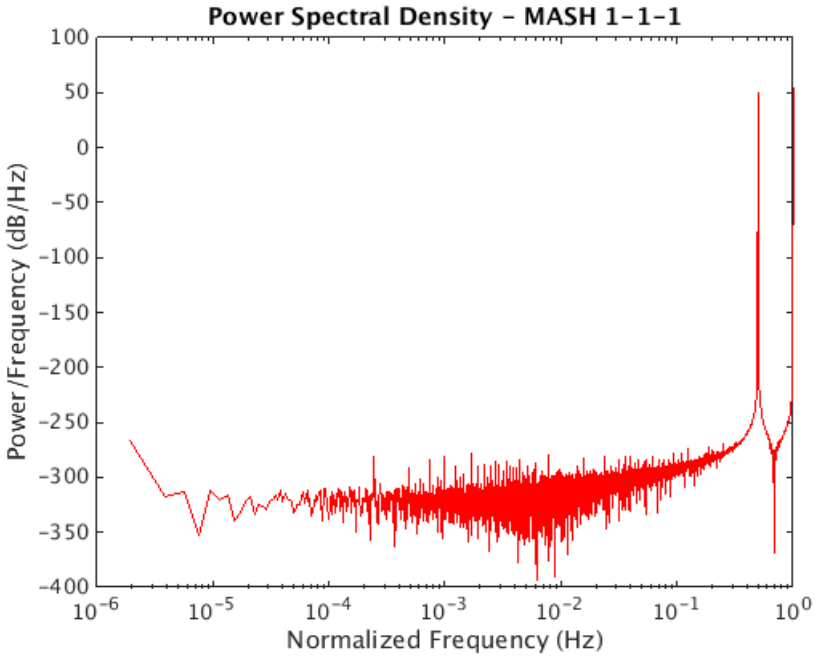

Figure 6.6: PSD from MASH 1-1-1 DDSM with even initial condition - $x = 0.5$

In figure 6.7 the PSD with odd initial condition for the first stage is plotted. In figure 6.8 the setting window for the initial condition for one stage is displayed. The window opens by clicking on the delay block in an EFM.

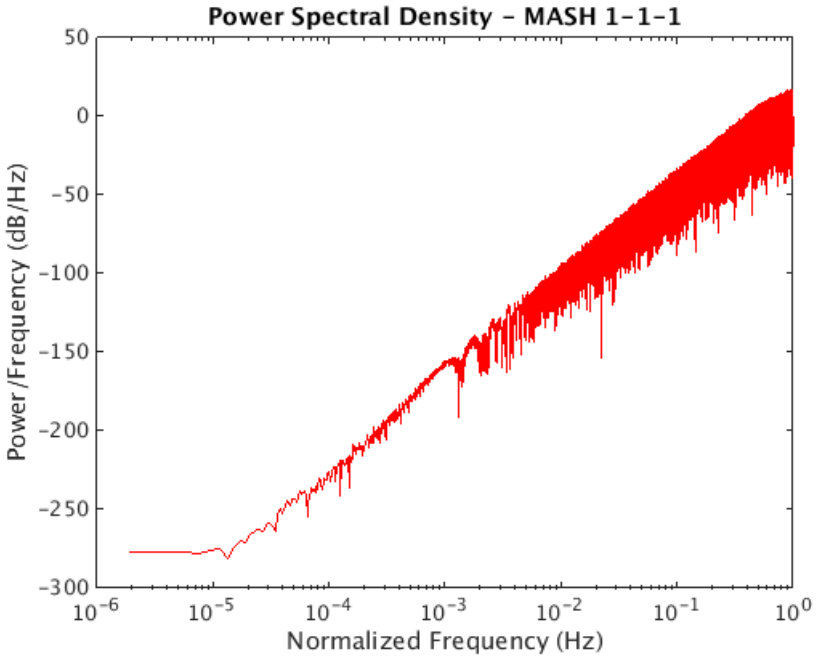

Figure 6.7: PSD from MASH 1-1-1 DDSM with odd initial condition - $x = 0.5$

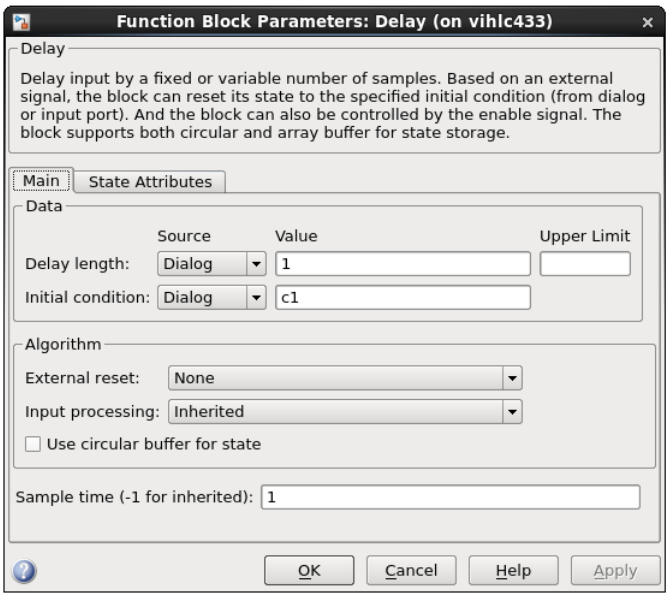

Figure 6.8: Window to define the initial condition for one stage

6.4 Reduced Complexity MASH DDSM with Error Masking

With this approach it is possible to reduce the complexity of the MASH, while maintaining a cycle length comparable to the conventional structure. The variable performance depending on the input value and initial conditions still is an issue, however. Comparing figure 6.6 and 6.7 with 6.9 the same performance can be observed.

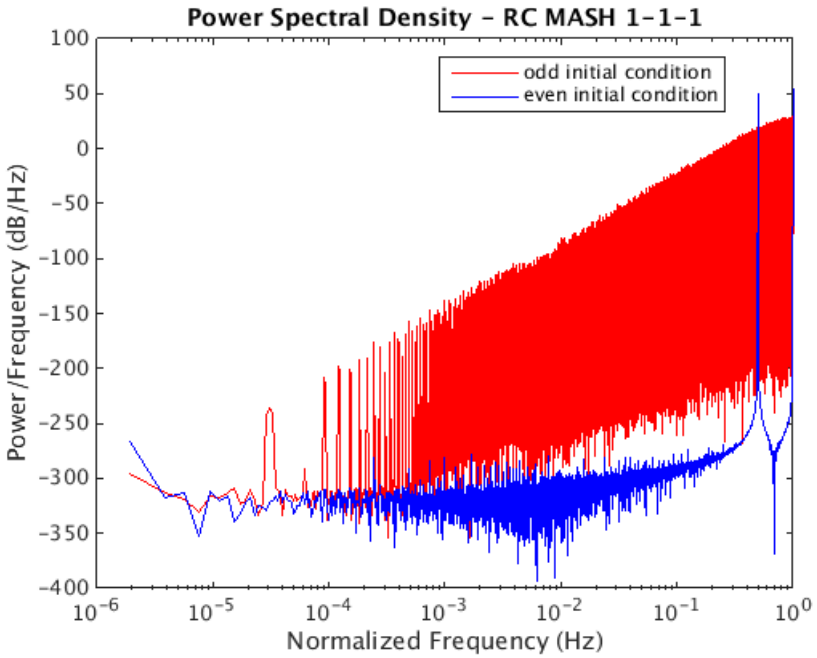

Figure 6.9: PSD of the reduced complexity MASH 1-1-1 DDSM with odd and even initial condition

In figure 6.10 the modified Simulink model is plotted. Compared to the conventional EFM an additional block named *N to M* to reduce the bit width is used. The content of this block is plotted in figure 6.11. The rest of the model, like the ECN and the MASH structure are the same as plotted in figure 6.3 and 6.4.

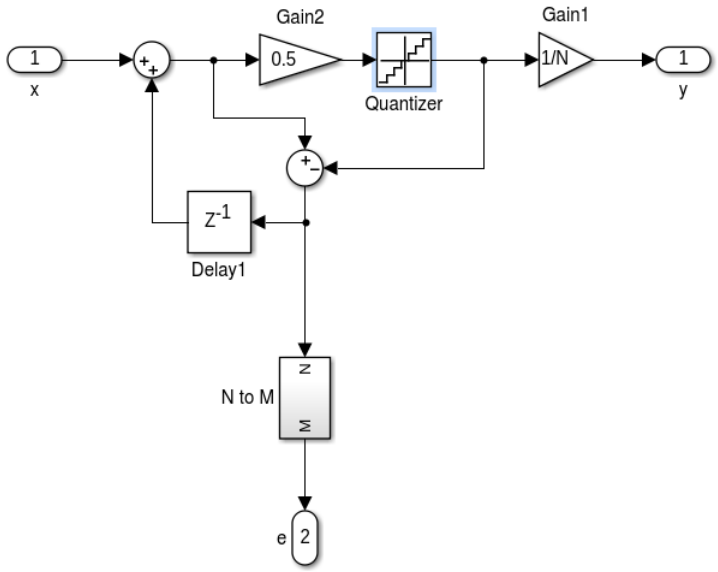

Figure 6.10: Simulink model of the reduced complexity EFM

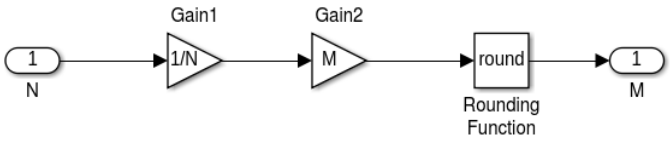

Figure 6.11: Content of the N to M block to reduce the number of bits

To hide the quantisation noise contribution from the interstage quantizers behind the shaped quantisation noise from the third stage values for N, M and L with $N_0 = 21$ are chosen as follows:

- $N = N_0 + 1 = 21 + 1 = 22$
- $M = ceil\left(\frac{4N - 10,6}{5}\right) = ceil\left(\frac{4 * 22 - 10,6}{5}\right) = 16$
- $L = ceil\left(\frac{2N - 5,3}{3}\right) = ceil\left(\frac{2 * 22 - 5,3}{3}\right) = 13$

One important thing to take care of is not to use 1 for the odd initial condition for the first stage because due to the reduced complexity approach from the first to the second stage the

information is cut off and the model behaves like the initial condition is set to zero. One possible initial condition is 65.

6.4.1 Adding Dither

Zeroth-Order

To realise this approach dither is added to the LSB of the input signal, like displayed in figure 6.12. The remaining model is as described in section 5.4. To generate the dither signal the following code is used in the Matlab controlling function:

```
simulationTime = 2^23;
H = comm.PNSequence('SamplesPerFrame',simulationTime);
ditherValue = step(H);
time = [(1:simulationTime)]';
dither = [time, ditherValue];
```

Through the combination of the functions *PNSequence* and *steps* it is possible to generate a sequence of pseudo random binary numbers with 2^{23} samples per frame. [21]

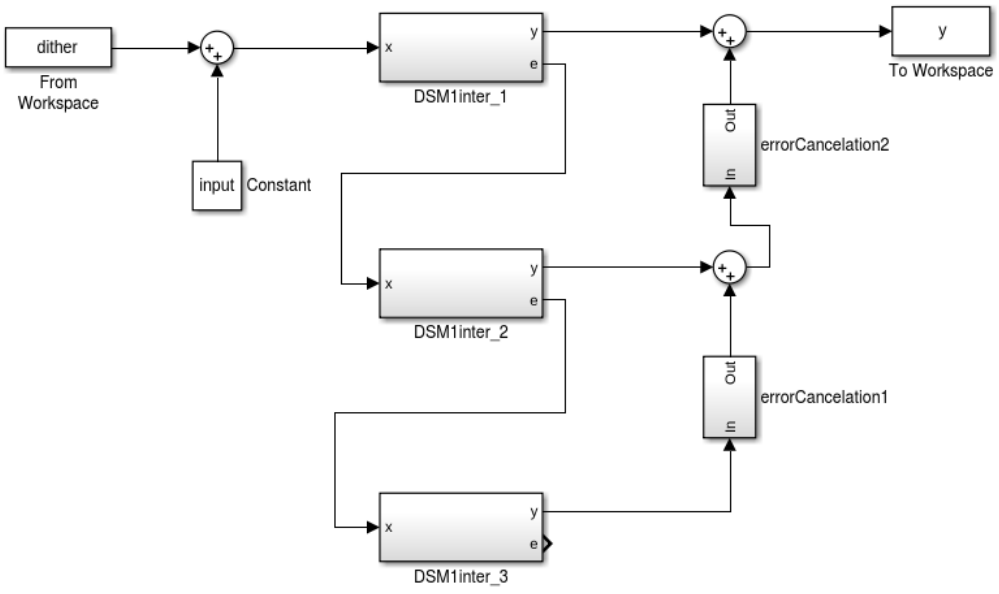

Figure 6.12: Simulink model of the RC zeroth-order dither MASH DDSM

To hide the quantisation noise contribution from the interstage quantizers behind the shaped quantisation noise from the third stage for Zeroth-Order Dither values for N, M and L with

$N_0 = 21$ are chosen as follows:

- $N = N_0 + 1 = 21 + 1 = 22$
- $M = ceil\left(\frac{2N}{3}\right) = ceil\left(\frac{2 * 22}{3}\right) = 15$
- $L = ceil\left(\frac{N}{3}\right) = ceil\left(\frac{22}{3}\right) = 8$

In figure 6.13 the PSD of this structure with odd and even initial condition is presented. It can be observed that the effect of changing the initial condition is not as huge as in the previous models. Additionally, it can be observed that the low frequency noise floor is much higher compared to that of the previously presented models. This is a big disadvantage because the information from the input signal is located in this area and the goal is to shape the noise from low to high frequencies.

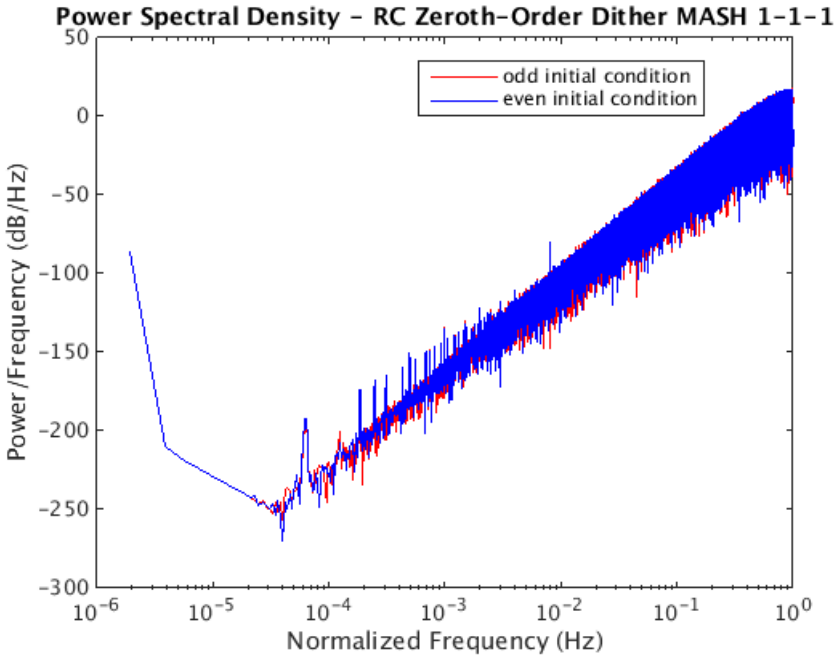

Figure 6.13: PSD of the RC zeroth-order dither MASH 1-1-1 DDSM with odd and even initial condition

First-Order

The bad low frequency behaviour can be removed by using first- instead of zeroth-order dither. To realise this approach the dither signal is first order high pass filtered. In figure 6.14 the modified MASH structure is presented. The content from the filter block is displayed in figure 6.15.

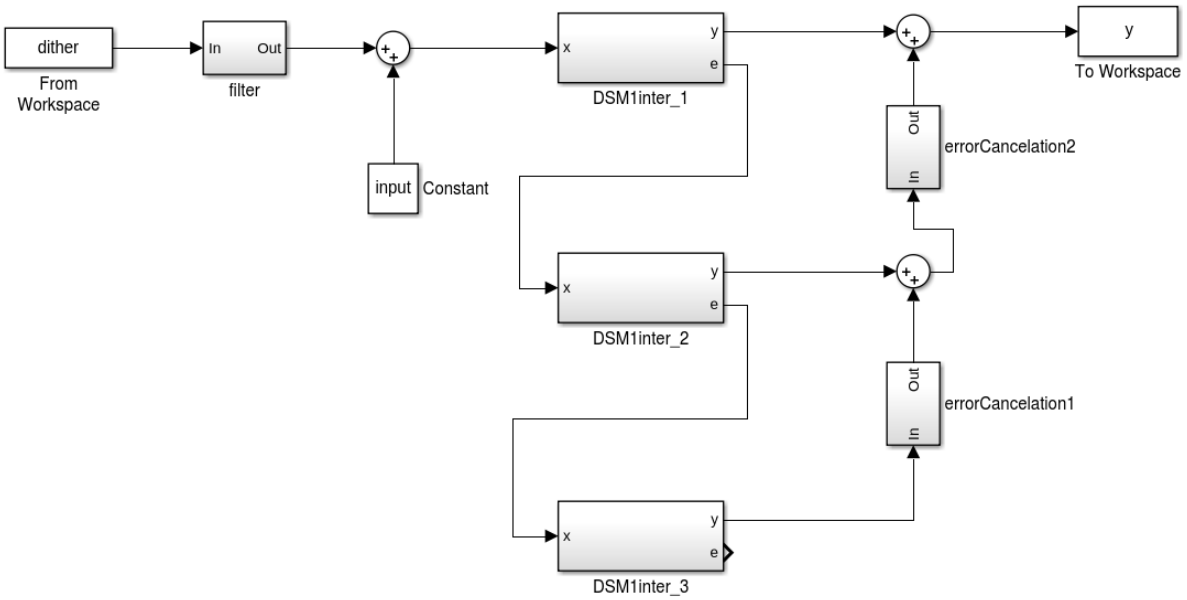

Figure 6.14: Simulink model of the RC first-order dither MASH DDSM

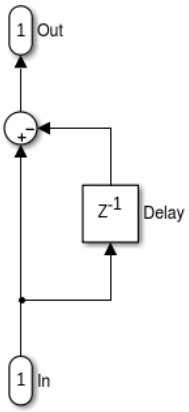

Figure 6.15: Content of the filter block - first order high pass filter

In figure 6.16 the PSD with odd and even initial condition is plotted. It can be observed that the low frequency noise floor is much better compared to figure 6.13 and that with an odd initial condition a very good behaviour for the PSD can be realised.

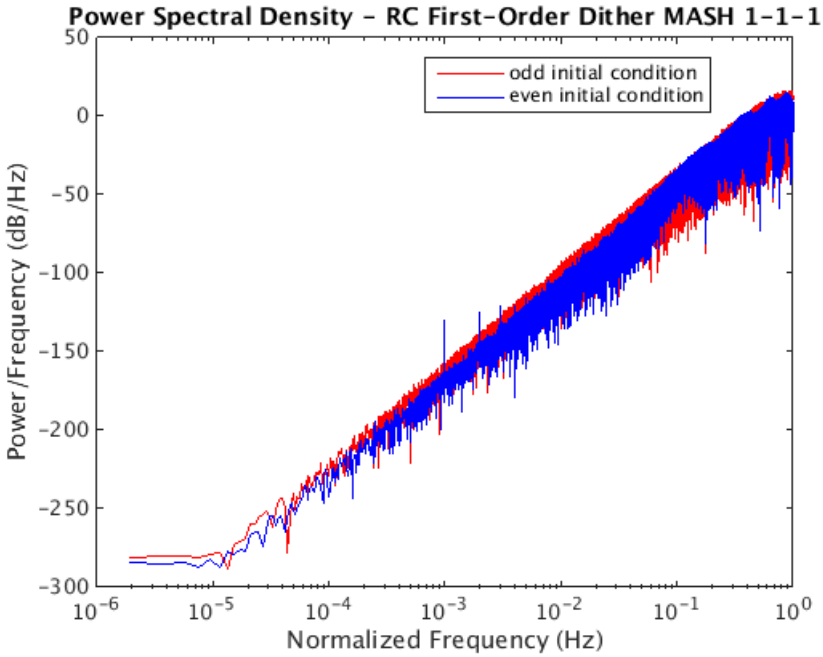

Figure 6.16: PSD of the RC first-order dither MASH 1-1-1 DDSM with odd and even initial condition

To hide the quantisation noise contribution from the interstage quantizers behind the shaped quantisation noise from the third stage for first-order dither values for N, M and L with $N_0 = 21$ are chosen as follows:

- $N = N_0 + 1 = 21 + 1 = 22$
- $M = L = 22$
- $L > \dfrac{N+1}{2} = 12$

Because of the fact that the noise floor for the zeroth-order approach is very high, for the first-order dither more hardware is needed and in general additional hardware is needed to realise the dither generator, structures with dither are not considered in the decision for finding a better structure for the DDSM.

6.5 Bus Splitting DDSM

For this approach plotting the PSD of the output signal only including the shaped quantisation noise is not easy, because several DDSMs are combined to one system. Through the combination it is not possible only to delete the constant input from the output signal, because the output

from the first stage for example includes the input value and the shaped quantisation noise and becomes the input to the next stage. Due to this fact and the reason that structures with dither are not considered in the final decision in the following section only models and some calculations of this approach are displayed.

6.5.1 Zeroth-Order Dither

1-2-3 Nested DDSM

To realise this approach the following calculations using $N = 21$ are needed:

- $M = \left\lceil \frac{2N}{3} \right\rceil = \left\lceil \frac{2 * 21}{3} \right\rceil = 14$
- $L = \left\lceil \frac{N}{3} \right\rceil = \left\lceil \frac{21}{3} \right\rceil = 7$
- $N_{MSB} = L = 7$
- $N_{ISB} = M - L = 14 - 7 = 7$
- $N_{LSB} = N - M = 21 - 14 = 7$

In figure 6.17 the built Simulink model is presented.

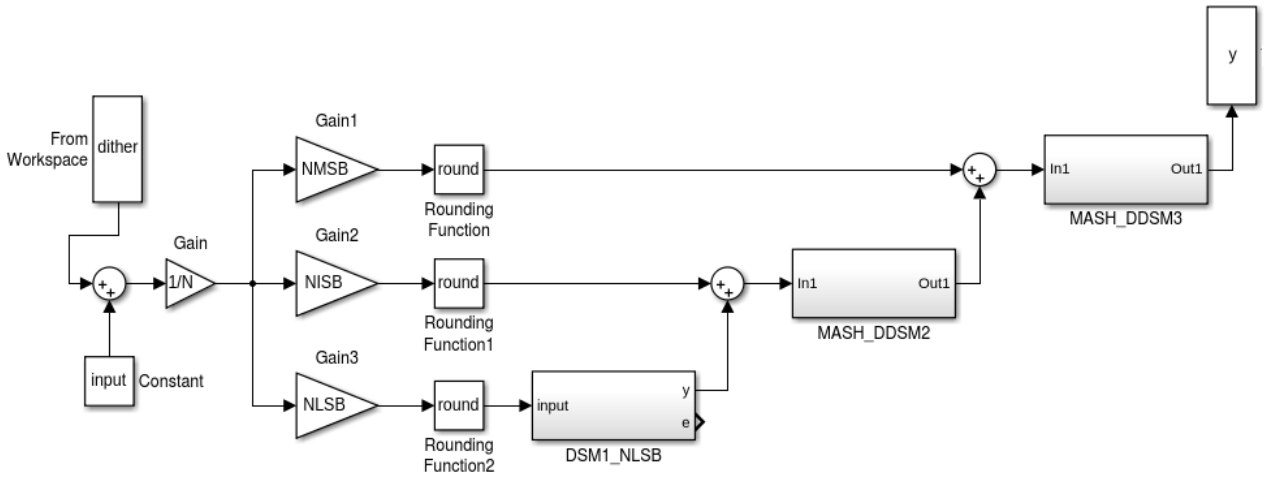

Figure 6.17: Simulink model of the 1-2-3 Nested zeroth-order dither DDSM

1-3 Nested DDSM

To build the model the following calculations using $N = 21$ are needed:

- $N_{MSB} = M = 14$
- $N_{LSB} = N - M = 21 - 14 = 7$

In figure 6.18 the built Simulink model is presented.

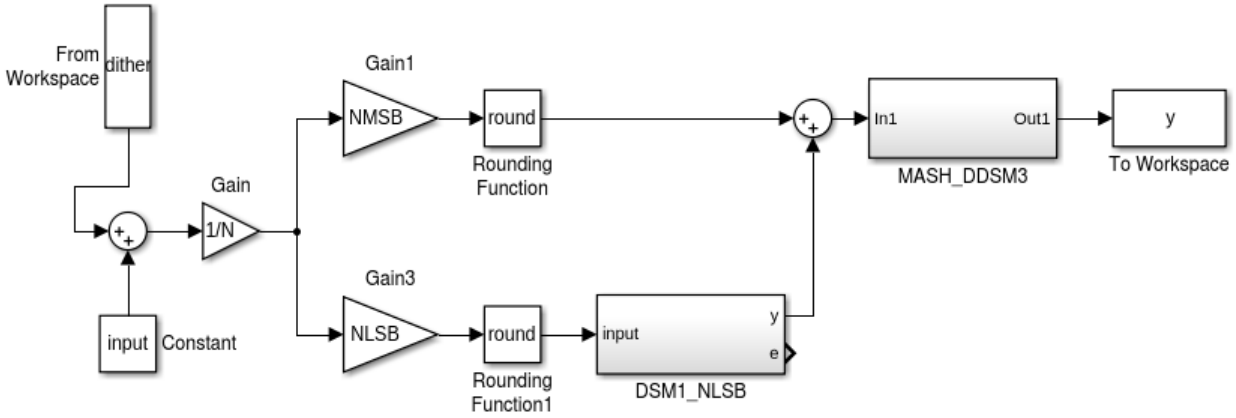

Figure 6.18: Simulink model of the 1-3 Nested zeroth-order dither DDSM

6.5.2 First-Order Dither

To build the model the following calculations using $N = 21$ are needed:

- $N_{MSB} = \left\lceil \frac{N}{2} \right\rceil = \left\lceil \frac{21}{2} \right\rceil = 11$
- $N_{LSB} = N - N_{MSB} = 21 - 11 = 10$

In figure 6.19 the built Simulink model is presented.

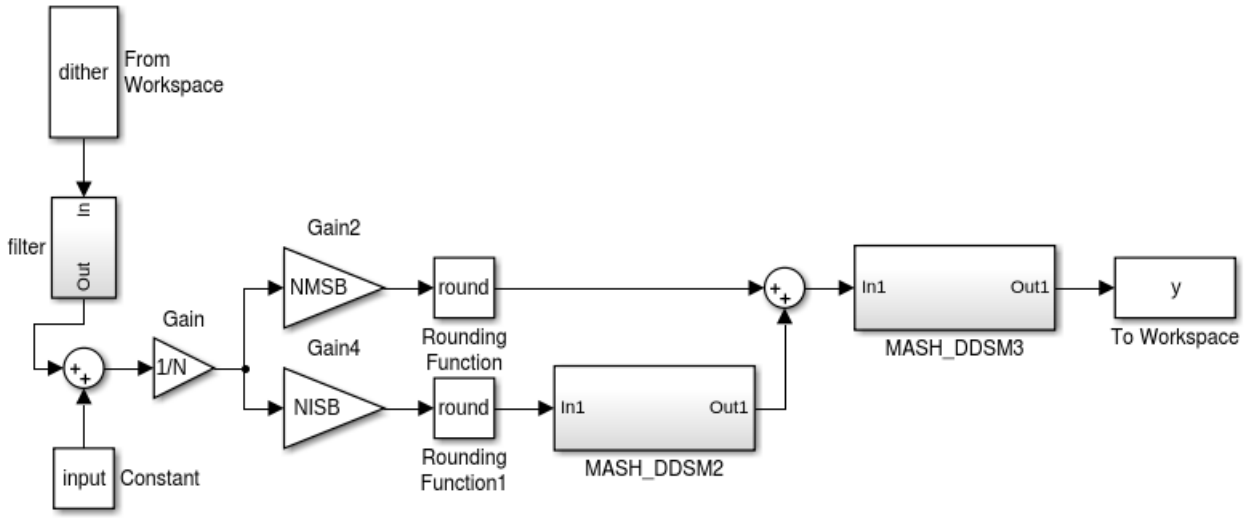

Figure 6.19: Simulink model of the 2-3 Nested first-order dither DDSM

6.6 HK - MASH DDSM

In figure 6.20 the modified MASH 1-1-1 structure with an additional filter is plotted. The content from the filter block is presented in figure 6.22. The MASH structure uses the modified EFM displayed in figure 6.21.

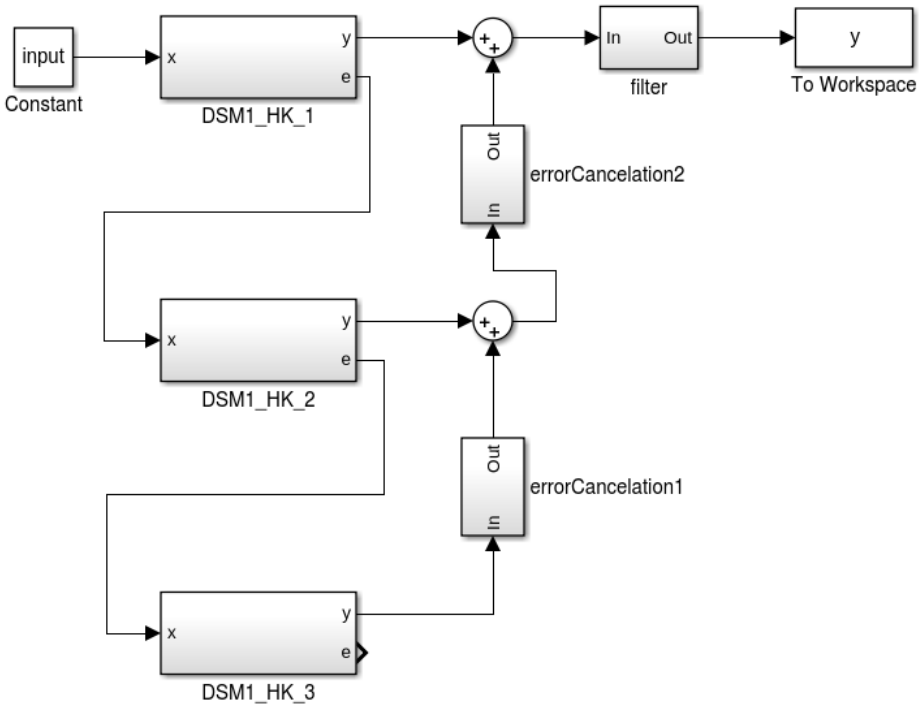

Figure 6.20: Simulink model of the HK MASH 1-1-1 DDSM with filter

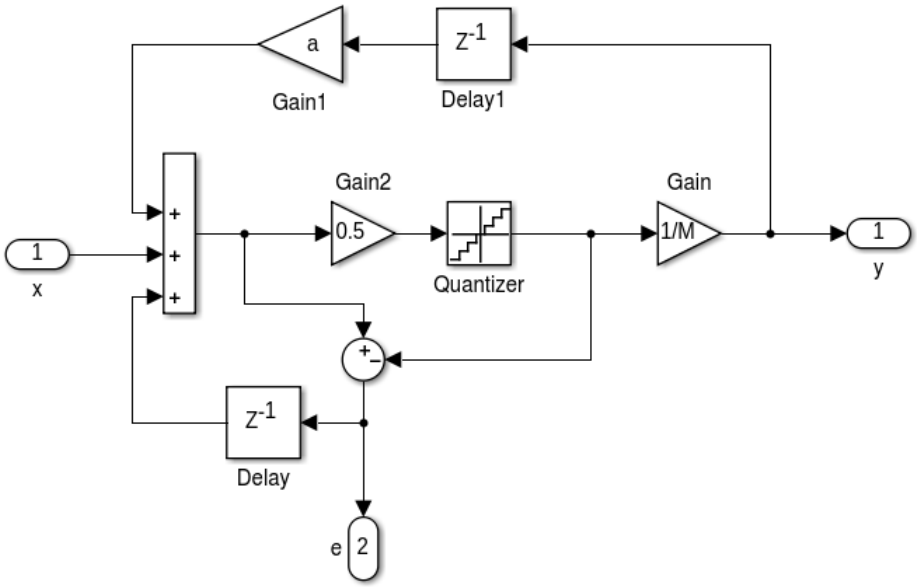

Figure 6.21: Simulink model of the HK EFM

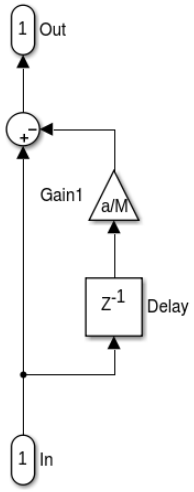

Figure 6.22: Simulink model of the filter used in HK MASH 1-1-1 DDSM

In figure 6.23 the PSD of the output signal for 21-bit with $a = 9$, taken from the last chapter, with filter, odd and even initial condition is plotted. The use of an odd initial condition in the first stage is not necessary, because there is nearly no difference between the two graphs in figure 6.23. The best realisation for the HK MASH DDSM would be using a number of bits for which $a = 1$ because in that case the 1-bit carry-in input from the next stage can be used to add the additional delayed signal to the input. In every other case of a additional hardware for adding the delayed signal to the input is needed. With this approach it is possible to reduce the number of bits from 21 to 13 with $a = 1$ without sacrificing the performance. The resulting PSD for odd

and even initial condition is plotted in figure 6.24. In figure 6.25 the outputs from the 21 and the 13-bit structure are compared.

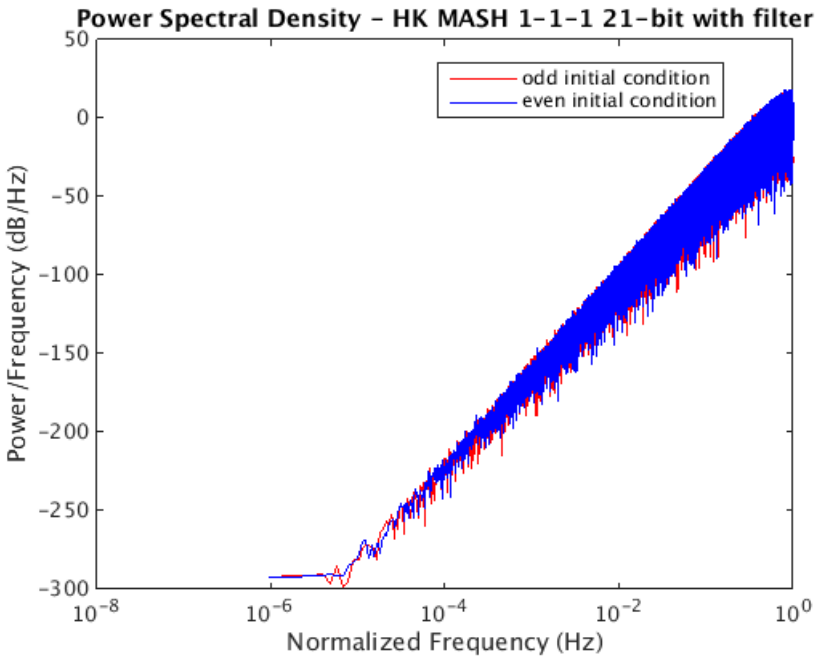

Figure 6.23: PSD of the 21-bit HK MASH 1-1-1 DDSM with filter for odd and even initial condition

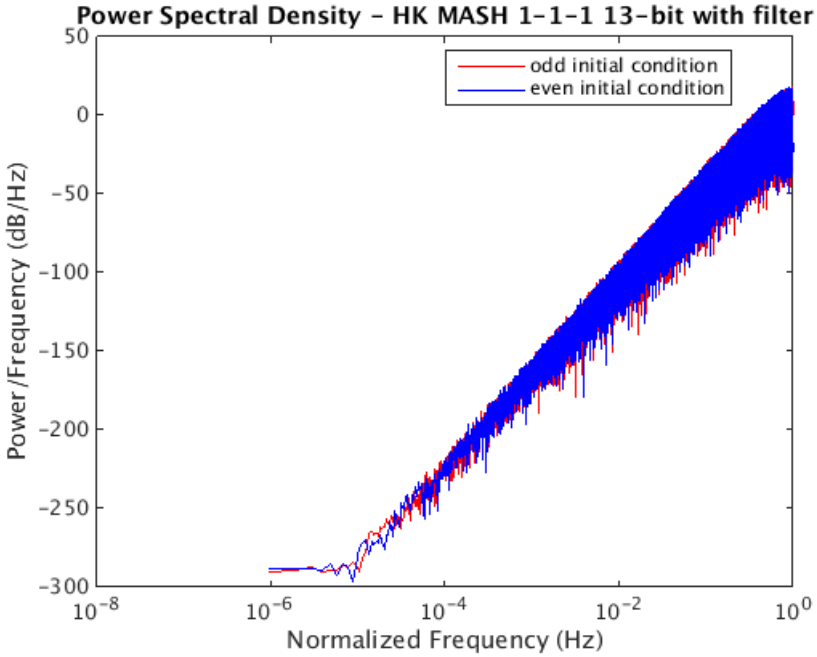

Figure 6.24: PSD of the 13-bit HK MASH 1-1-1 DDSM with filter for odd and even initial condition

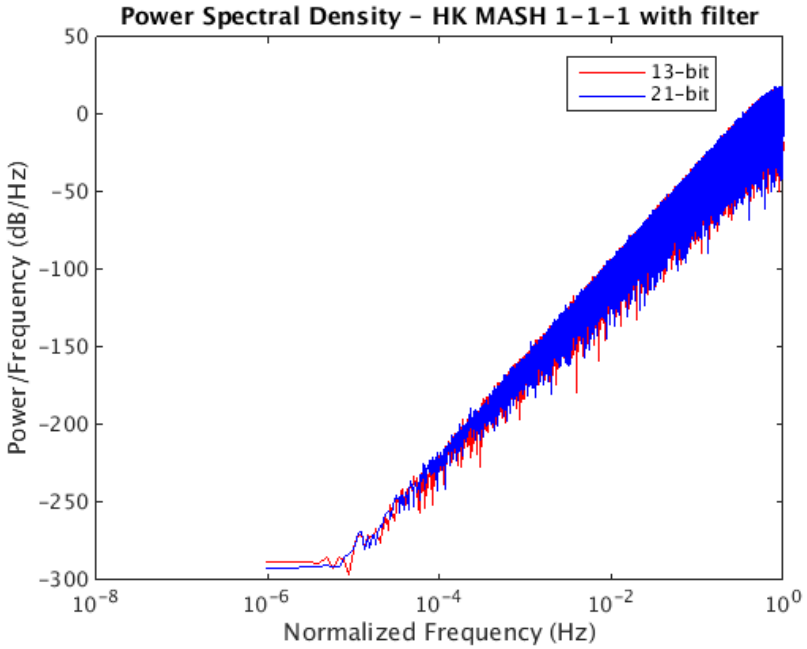

Figure 6.25: PSD of the 13-bit and 21-bit HK MASH 1-1-1 DDSM with filter

In [3] they came to the conclusion that the difference in the PSD by using or not using an additional filter in the HK MASH structure can be neglected. Comparing figure 6.23 and 6.26 the additional error can not be neglected. Because of this comparison in the final decision only the HK MASH structure with filter is considered.

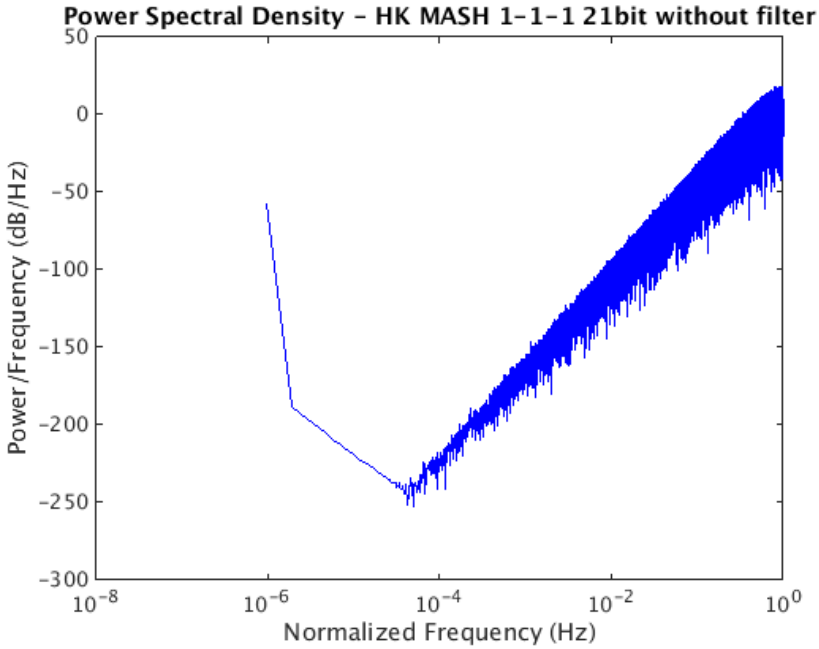

Figure 6.26: PSD of the 21-bit HK MASH 1-1-1 DDSM without filter

6.7 Spur-free MASH DDSM

In figure 6.27 the model of the modified MASH structure is presented. In this approach the carry-out signal is additionally fed forward and added to the input of the next stage. For the addition the carry-in input from the next stages accumulator can be used, because the additional forwarded signal has only 1-bit.

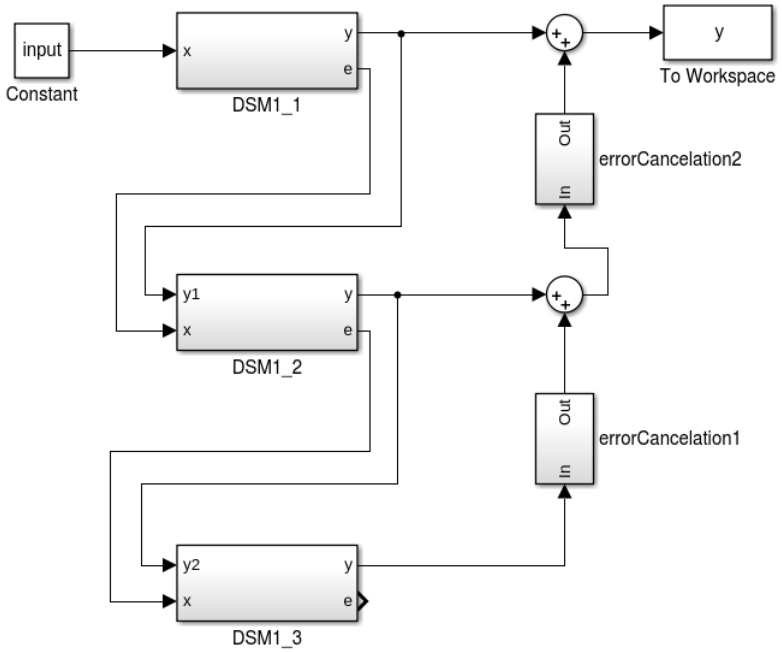

Figure 6.27: Simulink model of the Spur-free MASH 1-1-1 DDSM

In figure 6.28 the resulting PSD for the Spur-free MASH 1-1-1 DDSM with odd and even initial condition using 21-bit is displayed. With this approach it is possible to reduce the number of bits from 21 to 17 without sacrificing the performance. The resulting PSD for odd and even initial condition is plotted in figure 6.29. In figure 6.30 the outputs from the 21- and the 17-bit structure are compared.

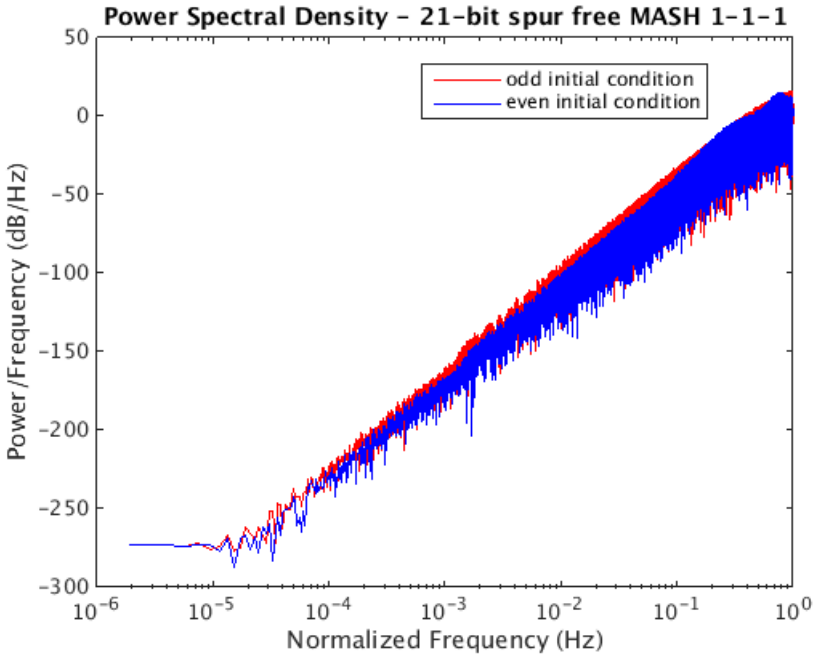

Figure 6.28: PSD of the 21-bit Spur-free MASH 1-1-1 DDSM with odd and even initial condition

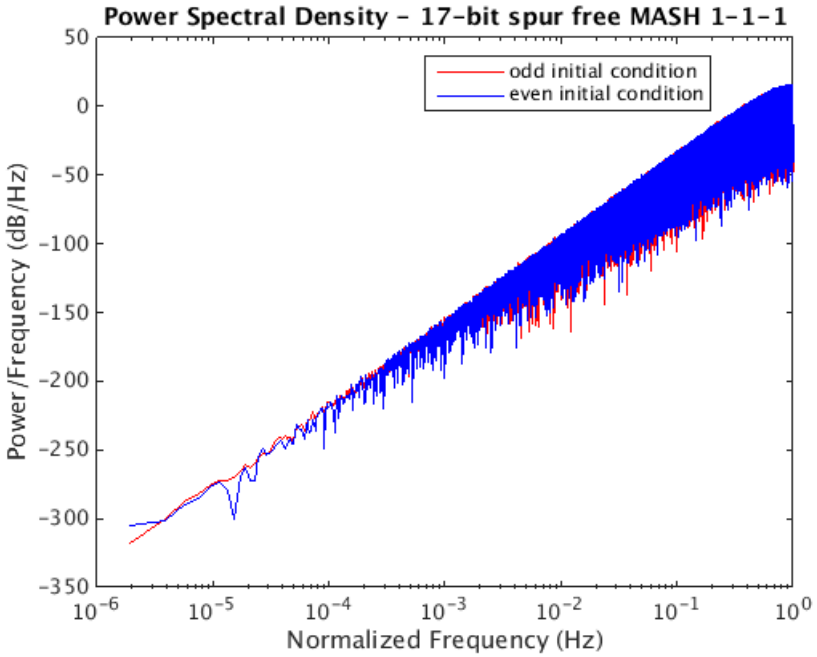

Figure 6.29: PSD of the 17-bit Spur-free MASH 1-1-1 DDSM with odd and even initial condition

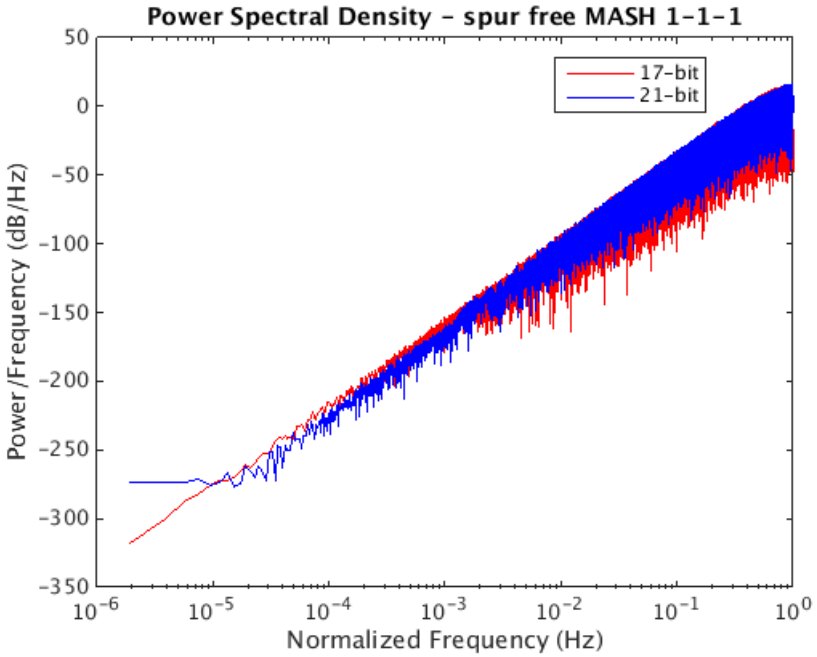

Figure 6.30: PSD of the 17-bit and 21-bit Spur-free MASH 1-1-1 DDSM

6.8 Folded MASH DDSM

In figure 6.31 the Simulink model of the folded MASH 1-1-1 DDSM using the folded EFM and folded ECN displayed in figure 6.32 and 6.33 is presented. The idea is to reduce the hardware consumption by the factor of two by splitting the input into two parts of the same length and processing this parts alternatively with the same structure. To realise this idea a toggle flip-flop and a multiplexer is needed. Additionally every register has to double, because now the structure generates every two clock cycles an output. Another disadvantage is that the folded MASH DDSM has to be combined with a conventional MASH DDSM to keep the noise shaping capability [10]. Summarized the hardware consumption is much bigger compared to the conventional MASH DDSM. Because of this the combined structure of folded MASH DDSM and MASH DDSM was not realised. In following only the Simulink models of the folded approaches are displayed, because of the increased hardware consumption this approach is not considered in the final decision.

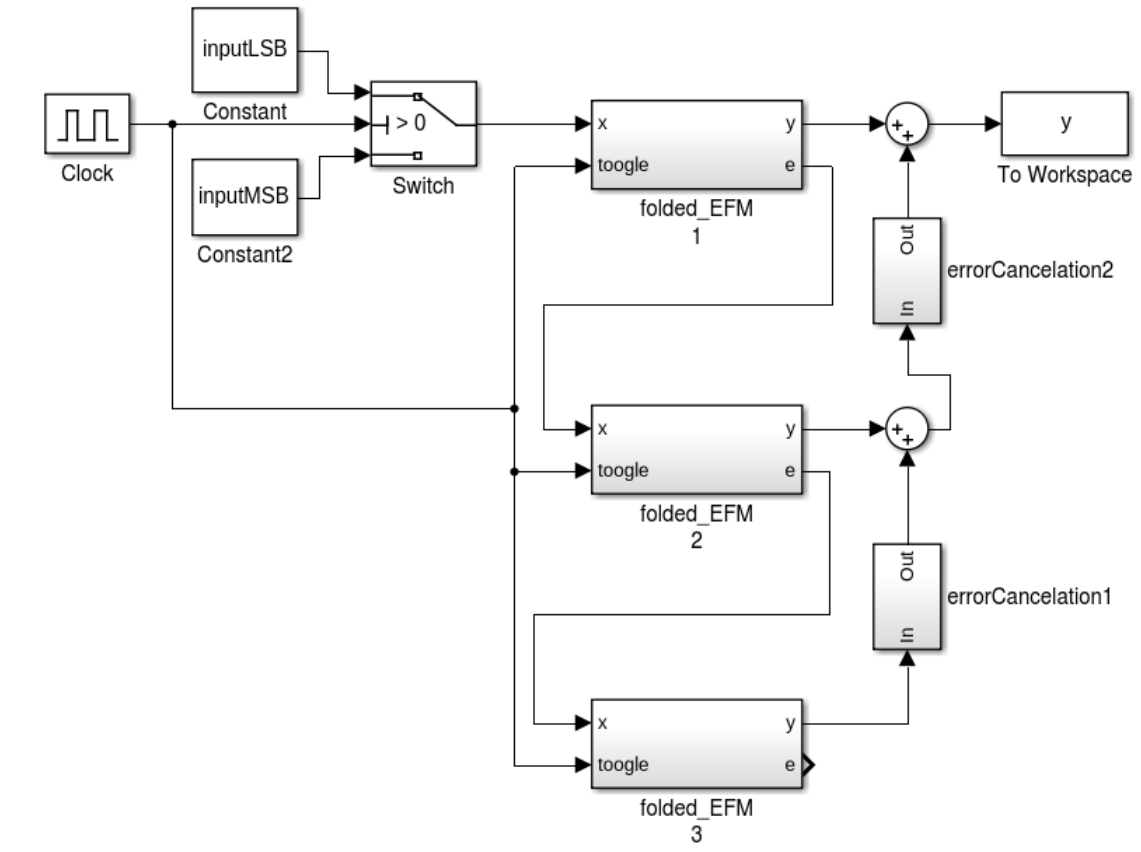

Figure 6.31: Simulink model of the folded MASH 1-1-1 DDSM

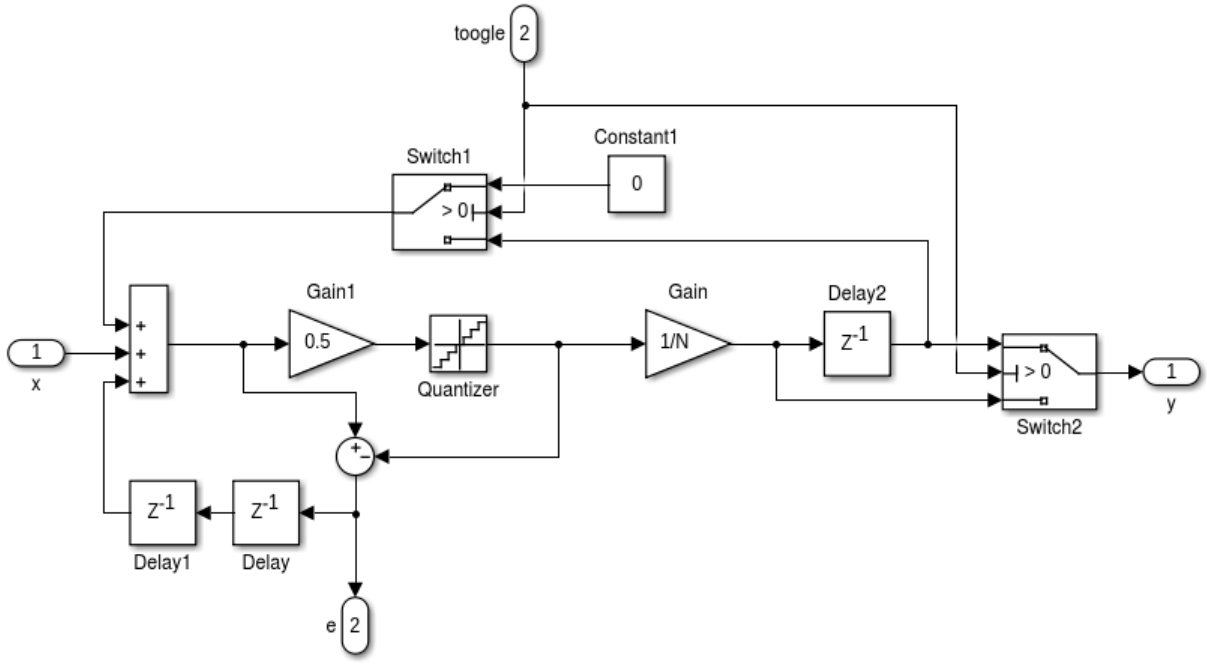

Figure 6.32: Simulink model of the folded EFM

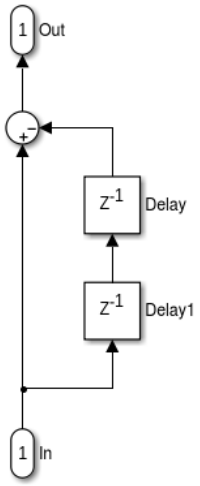

Figure 6.33: Simulink model of the folded ECN

The following Matlab code was used in the main function to split the input into two parts:

```
inputFractional = 0.5
n0 = 20;
N = 2^n0;
input = round(N * inputFractional);
inputBinaer = zeros(1,n0)
inputConv = de2bi(input)
inputBinaer(1:length(inputConv)) = inputConv

inputMSB= bi2de(inputBinaer(n0/2+1:n0))*2^(n0/2)
inputLSB= bi2de(inputBinaer(1:(n0/2)))
```

6.9 Conclusion - Final Discussion

In general with the HK - MASH DDSM presented in this chapter it is possible to reduce the necessary hardware from 21 to 13-bit per stage. Despite the fact that an additional filter is necessary the hardware saving is big compared to the currently used structure. Additionally the performance is constant for all input values and initial conditions.

For this thesis the reduction from 21 to 13-bit per stage is not possible because of the specifications of the project the DDSM is designed for. In the following section this problem is explained in detail.

6.9.1 Limitations According to the Project Specifications

The DDSM is used in the project for Amplitude Shift Keying (ASK) and Frequency Shift Keying (FSK). In figure 6.34 and 6.35 an example for ASK and FSK is presented. Thereby *(a)* is the original digital waveform and *(b)* is the time-domain waveform [11]. For ASK the frequency and due to this the input to the DDSM is constant. For FSK the frequency varies due to equation 6.3. The actual frequency is the center frequency f_c plus/minus the deviation frequency f_d. For the project the center frequency is 434 MHz or 315 MHz and the deviation frequency can be everything from 30 kHz to 80 kHz with a minimum step size of 1 kHz. The sampling frequency for the system is 26 MHz and the minimum resolution for ASK and FSK should be 1 kHz. To realise the 1 kHz resolution due to equation 6.4 15-bit are necessary. To have a little buffer and because of the inaccuracy of the analog part of the chip 17-bit are chosen.

$$f = f_c \pm f_d = 434\,MHz \pm 50\,kHz \tag{6.3}$$

Most of the papers from chapter 4 tested their approaches only with constant inputs. To check if the approach is also working for a variable input a Matlab function for generating a frequency according to equation 6.3 was developed and is presented in the following section.

$$\frac{sampling frequency}{2^{n_0}} = \frac{26\,MHz}{2^{15}} = 793.46\,Hz \tag{6.4}$$

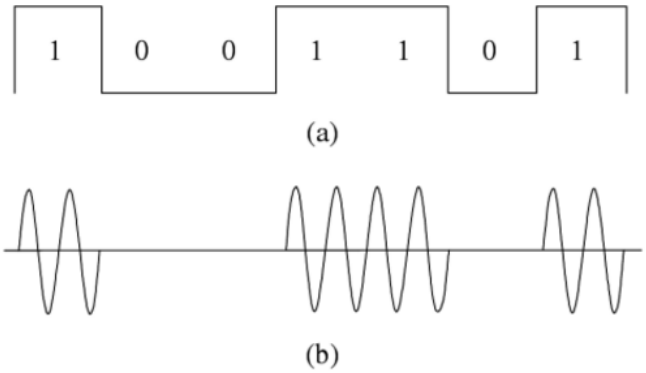

Figure 6.34: Example for ASK [11]

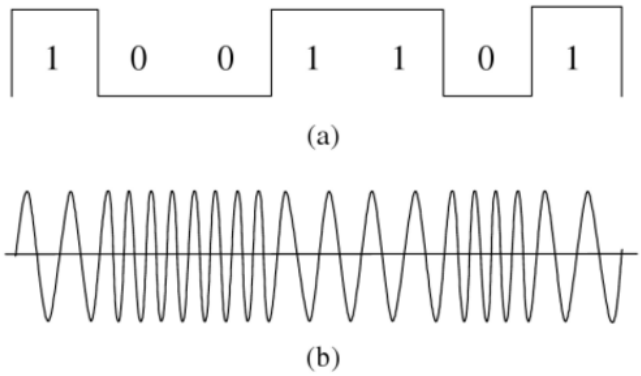

Figure 6.35: Example for FSK [11]

6.9.2 Variable Input

In the following code a function to generate and plot a variable input value is presented. In figure 6.36 the generated output sequence is displayed. The generated signal has a center frequency of 434 MHz and a deviation of 50 kHz. The modulo function from Matlab *mod* is used to generate a sequence of alternating zeros and ones. In the second part the value 434 ± 50 is assigned to the sequence.

```
average = 434000000;
stepSize = 50000;
simulationTime = 2^3;
H = zeros(1,simulationTime);

for i= 1:1:simulationTime
    if mod(i,2) == 0
        H(i) = 1;
    end
end

for i = 1:1:length(H)
   if H(i) == 1
       H(i) = average + stepSize;
   else
       H(i) = average - stepSize;
   end
end

time = [(1:length(H))]';
variablerInput = [time, H'];
```

```
figure(1)
stairs(time, H)
ylim([(average-2*stepSize) (average+2*stepSize)])
ylabel('Inputvalue (Hz)')
xlabel('Time (ns)')
```

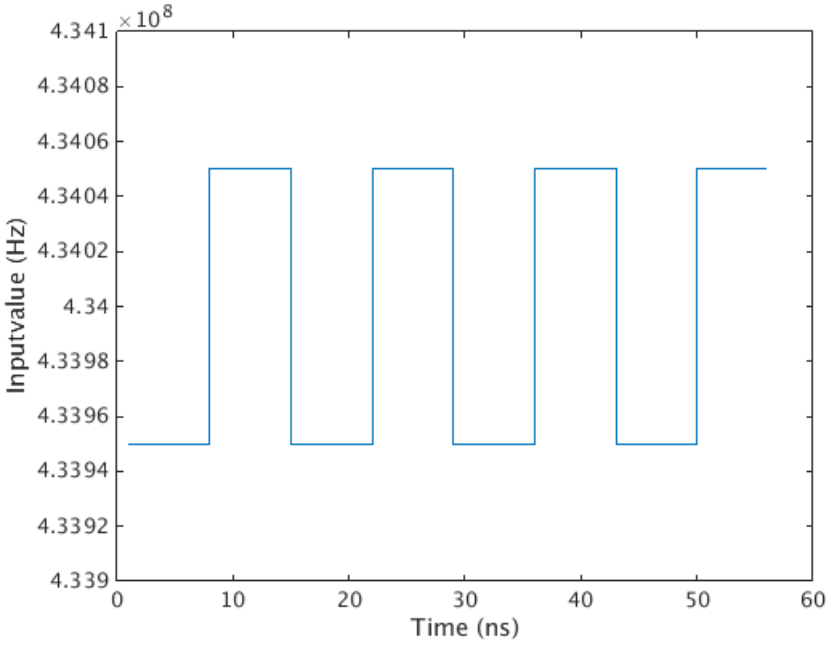

Figure 6.36: Plot of the generated variable Input

6.9.3 Function for Comparing PSDs

Due to the fact that the minimum number of bits has to be 17 and in this chapter different approaches with a comparable PSD for 17-bit are presented a function to evaluate the best solution was developed. The basic idea for the function is to approximate the PSD for each structure with one line according to equation 6.5 and compare the lines. Each line should have the same value for k, because each quantisation error is third order high pass filtered. This means k is 60 dB/dec. Hence it should be possible to evaluate the quality of a structure by the value d. The more negative the value is the smaller the quantization noise becomes the better the structure is. In figure 6.37 the different steps of the function are plotted. Starting point is the original dataset, displayed in blue. For this dataset a lower and upper limit is defined, because the signal above and below these limits would falsify the following approximation. The reduced dataset is plotted in red. Outgoing from the reduced dataset the approximation, plotted in green, is calculated. As final step the approximation has to be shifted up. As far as the line

represents the highest value at each frequency (plotted in black). Additionally to the plot the function generates a .txt file. An example is displayed in table 6.1. It includes the name of the structure, the number of bits, the initial condition, the average output value and a ranking value. The ranking value is the value d as explained above.

$$y = kx + d \tag{6.5}$$

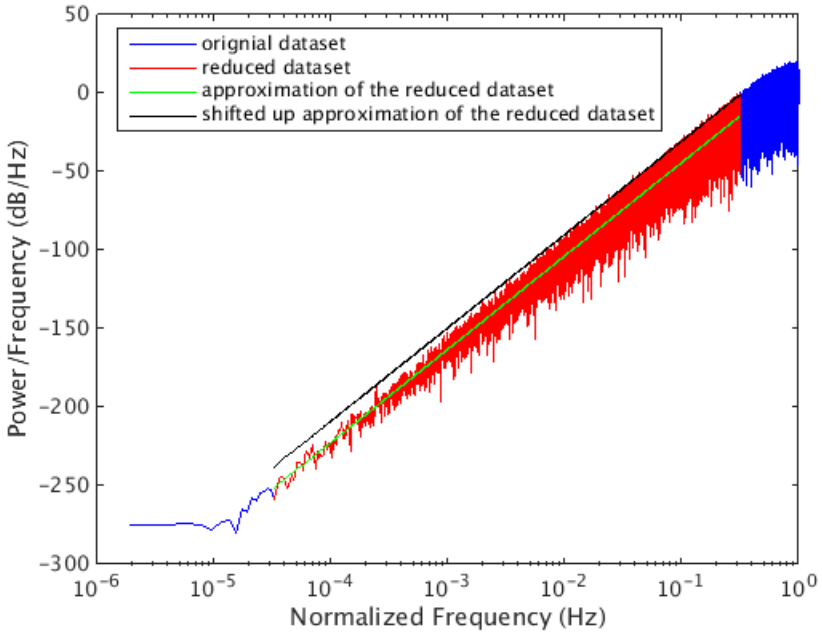

Figure 6.37: Steps to get the approximation

Table 6.1: Example .txt of the comparing function

Name	Bit	Initial Condition	Mean	Ranking
conv	21-21-21	1	0.4321	-26.46
HK	17-17-17	0	0.4321	-28.66
HK	19-17-13	0	0.4321	-28.94
SP	17-17-17	0	0.4321	-29.16

The following code snippets represent the important parts of the function:

Reduce the Dataset:

```
lowerLimit = 0.0001;
upperLimit = 1;
for i = 1 : length(w)
    if w(i) > lowerLimit && w(i) < upperLimit
        first = first + 1;
        w2(i,:) = w(i);

        if first == 1
            j = i;
        end
    end
end
```

The above code snippet generates a new vector containing all the values between the upper and lower limit and saves the index of the first element above the lower limit. The limits are defined by looking at the dataset. Important is to get only the range where the PSD falls 60 dB/dec.

Calculate the Approximation:

```
[p2,~,mu2] = polyfit(log10(w2/(pi)),10*log10(pi*Pyy(j:length(w2)+j-1)),1);
y3 = polyval(p2,log10(w2/(pi)),[],mu2);
```

"*p = polyfit(x,y,n) returns the coefficients for a polynomial p(x) of degree n that is a best fit [...] for the data in y*" [22]. In equation 6.6 the form of the resulting polynomial with $n = 1$ is displayed. The second calculated value is *mu*. "*Which is a two-element vector with centering and scaling values. mu(1) is mean(x), and mu(2) is std(x). Using these values, polyfit centers x at zero and scales it to have unit standard deviation. This centering and scaling transformation improves the numerical properties of both the polynomial and the fitting algorithm*" [22].

$$p(x) = p_1 x^n + p_2 x^{n-1} = p_1 x + p_2 \tag{6.6}$$

"*y = polyval(p,x) returns the value of a polynomial of degree n evaluated at x. [...] polyval evaluates p at each element of x. y = polyval(p,x,[],mu) [...] use* $\hat{x} = (x - \mu 1)/\mu 2$ *in place of x. In this equation,* $\mu 1 = mean(x)$ *and* $\mu 2 = std(x)$*. The centering and scaling parameters* $mu = [\mu 1, \mu 2]$ *are optional output computed by polyfit*" [23].

Shift up the Approximation:

```
maxData = max(10*log10(pi*Pyy(j:length(w2)+j-1));)
maxAppro = max(y3)
diff = abs(maxData -maxAppro)
p2(2) = p2(2) + diff;
y3 = polyval(p2,log10(w2/(pi)),[],mu2);
```

The difference between the maximal value of the reduced dataset and the approximation is calculated. The second coefficient of the polynomial is changed due to that difference and the values of the resulting polynomial are calculated again.

6.9.4 Tested Structures and Results

Constant Input

In table 6.2 and table 6.3 the contents from the generated .txt files for all considered combinations of structures and constant input values are displayed. Due to the explanation in section 5.9.1 the minimum number of bits used for testing is 17. Only the Reduced Complexity approach is an exception. For the Reduced Complexity approach only structures with odd initial condition are tested due to figure 6.9.

Table 6.2: Generated .txt files for constant inputs part 1

Name	Bit	Initial Condition	Mean	Ranking
conv	21-21-21	1	0.5	-30.77
RC 17	18-13-11	65	0.5	-8.56
RC 21	22-16-13	65	0.5	-17.86
HK	17-17-17	0	0.5	-28.46
HK	17-17-17	1	0.5	-29.06
SP	17-17-17	0	0.5	-29.39
SP	17-17-17	1	0.5	-29.40

Table 6.3: Generated .txt files for constant inputs part 2

Name	Bit	Initial Condition	Mean	Ranking
conv	21-21-21	1	0.1	-25.79
RC 17	18-13-11	65	0.1	-23.39
RC 21	22-16-13	65	0.1	-28.46
HK	17-17-17	0	0.1	-28.95
HK	17-17-17	1	0.1	-28.60
SP	17-17-17	0	0.1	-29.33
SP	17-17-17	1	0.1	-28.82
conv	21-21-21	1	0.25	-30.31
RC 17	18-13-11	65	0.25	-9.10
RC 21	22-16-13	65	0.25	-17.19
HK	17-17-17	0	0.25	-28.87
HK	17-17-17	1	0.25	-28.69
SP	17-17-17	0	0.25	-30.17
SP	17-17-17	1	0.25	-30.15
conv	21-21-21	1	0.75	-30.99
RC 17	18-13-11	65	0.75	-9.11
RC 21	22-16-13	65	0.75	-17.20
HK	17-17-17	0	0.75	-28.51
HK	17-17-17	1	0.75	-28.83
SP	17-17-17	0	0.75	-29.96
SP	17-17-17	1	0.75	-30.18
conv	21-21-21	1	0.4321	-26.46
RC 17	18-13-11	65	0.4321	-17.24
RC 21	22-16-13	65	0.4321	-28.14
HK	17-17-17	0	0.4321	-28.66
HK	17-17-17	1	0.4321	-28.03
SP	17-17-17	0	0.4321	-28.87
SP	17-17-17	1	0.4321	-28.01

Also a combination of the RC and the HK approach and a combination of the RC and the SP approach were tested. For the RC SP structure the resulting PSD and the average output value

were not satisfied. The average output value with 0.1 input value was 0.031. For the RC HK structure the resulting PSD was not satisfied. The base for the RC calculation have been 17-bit. In figure 6.38 and 6.39 the PSDs are plotted. Due to the above described facts the structure combinations are not included in table 6.2 and 6.3.

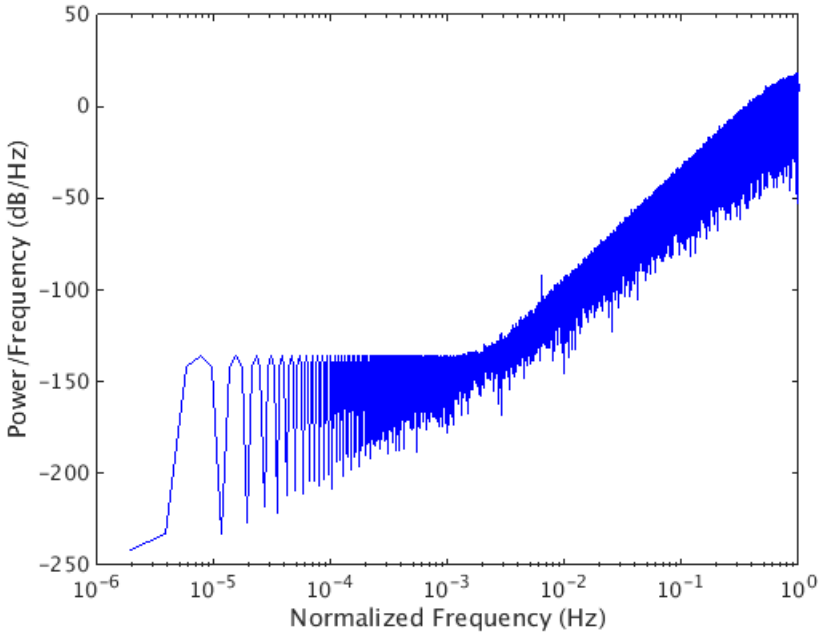

Figure 6.38: PSD of the combination of the RC and HK approach (input = 0.1)

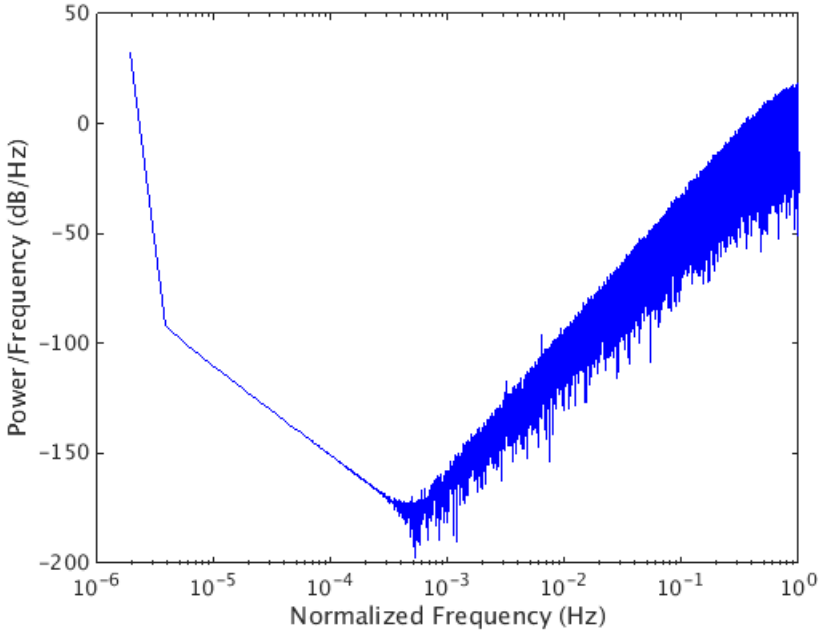

Figure 6.39: PSD of the combination of the RC and SP approach (input = 0.1)

Because of reason that several structures are tested and in the behaviour there is no big difference the generated plots are not clear enough to take a decision. Due to this only the results from table 6.2 and table 6.3 are used to decide about the best approach. Anyway in figure 6.40 an example PSD with all structures for one input value is displayed.

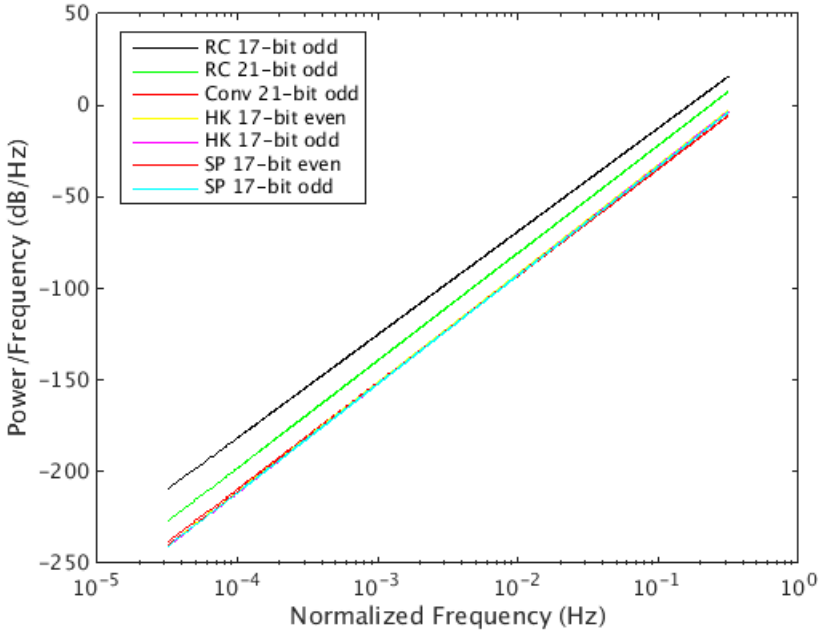

Figure 6.40: Example PSD with all structures for input value 0.5

Looking at table 6.2 and 6.3 it can be observed that the RC MASH 1-1-1 DDSM with 17-bit base and the RC MASH 1-1-1 DDSM with 21-bit base have the worst results for all tested input values. Between the HK MASH 1-1-1 DDSM with 17-bit and the SP MASH 1-1-1 DDSM with 17-bit is no big difference and between using odd or even initial condition for the HK MASH 1-1-1 DDSM or the SP MASH 1-1-1 DDSM is nearly no difference. The advantage of the SP MASH 1-1-1 against the HK MASH 1-1-1 is that the HK MASH 1-1-1 DDSM needs, due to the additional filter at the output, more hardware. Beside this there is nearly no difference because the tested HK MASH 1-1-1 DDSM uses the factor $a = 1$. Comparing the SP MASH 1-1-1 DDSM with even initial and the currently used structure it can be observed that the performance of the SP MASH 1-1-1 DDSM is at least comparable and sometimes even better for constant inputs despite the reduced hardware consumption.

The conclusion of the evaluation is that the SP MASH 1-1-1 DDSM with 17-bit and even initial condition satisfies all set goals and has a better performance than the other approaches. In the next section the favoured structure is tested for variable inputs.

Variable Input

The SP MASH 1-1-1 DDSM is tested for center frequency 434 MHz and 315 MHz with minimum data rate of 8 kchip/sec (4 kbit/sec Manchester encoded), maximum data rate of 40 kchip/second (20 kbit/sec Manchester encoded) and minimal and maximal deviation frequency for FSK of 30 and 80 kHz, because these are the specifications for the project the DDSM is used for. To realise the test parameters the function from section 5.9.2 to generate a variable input is slightly modified. The upper and lower limits are changed from integers to fractional values and the data rate is considered. In the following the new code and the resulting plots are displayed.

```
function [ variableInput, timeValue ] = variablerInput( input_args )

average = 434000000;
stepSize = 50000;
samples = 650;
samplingFrequency = 26000000;
simulationTime = 2^8;
fracUpper = ((average + stepSize)/samplingFrequency) -
floor((average + stepSize)/samplingFrequency);
fracLower = ((average - stepSize)/samplingFrequency) -
floor((average - stepSize)/samplingFrequency);
k = 1;

H = zeros(1, simulationTime);
 for i= 1:1:simulationTime
     if mod(i,2) == 0
         H(i) = 1;
     end
 end

for i = 1:1:simulationTime
   for j=1:1:samples
       value(k) = H(i);
       k = k + 1;
   end
end

for i = 1:1:length(value)
   if value(i) == 1
       value(i) = fracUpper;
   end
```

```
    if value(i) == 0
        value(i) = fracLower;
    end
end

time = [(1:length(value))]';
variableInput = [time, value'];
timeValue = length(value);

end
```

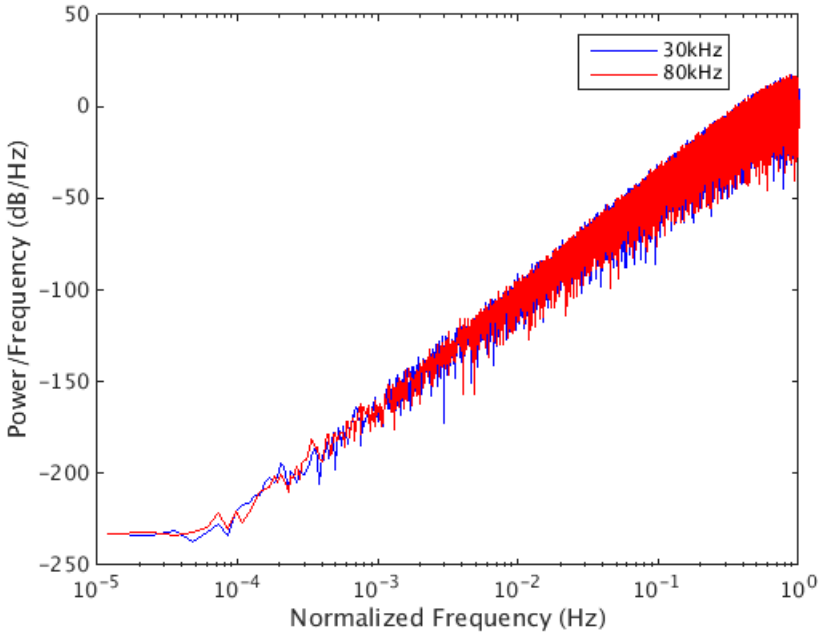

Figure 6.41: PSD for $f_c = 434$ MHz and maximum data rate with 30 and 80 kHz deviation frequency

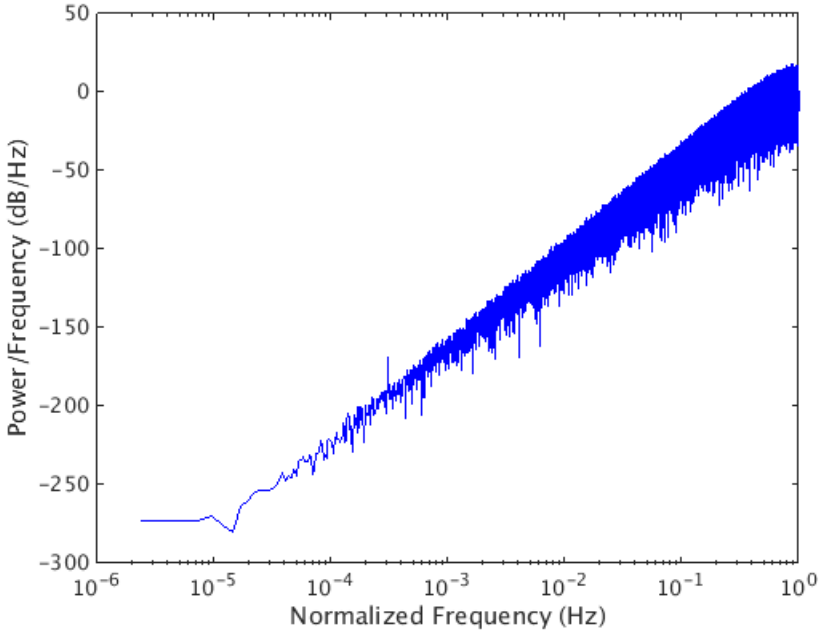

Figure 6.42: PSD for $f_c = 434$ MHz and minimum data rate with 30 kHz deviation frequency

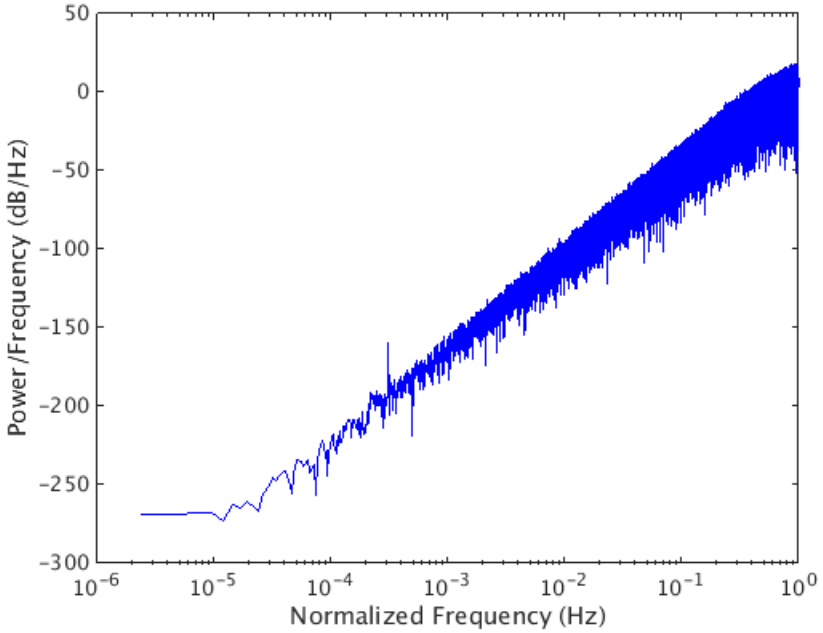

Figure 6.43: PSD for $f_c = 434$ MHz and minimum data rate with 80 kHz deviation frequency

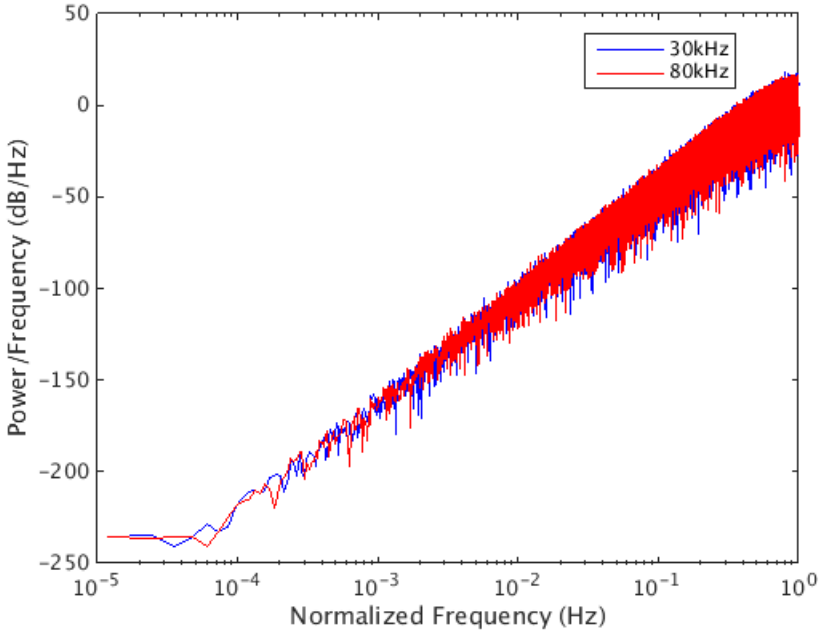

Figure 6.44: PSD for $f_c = 315$ MHz and maximum data rate with 30 and 80 kHz deviation frequency

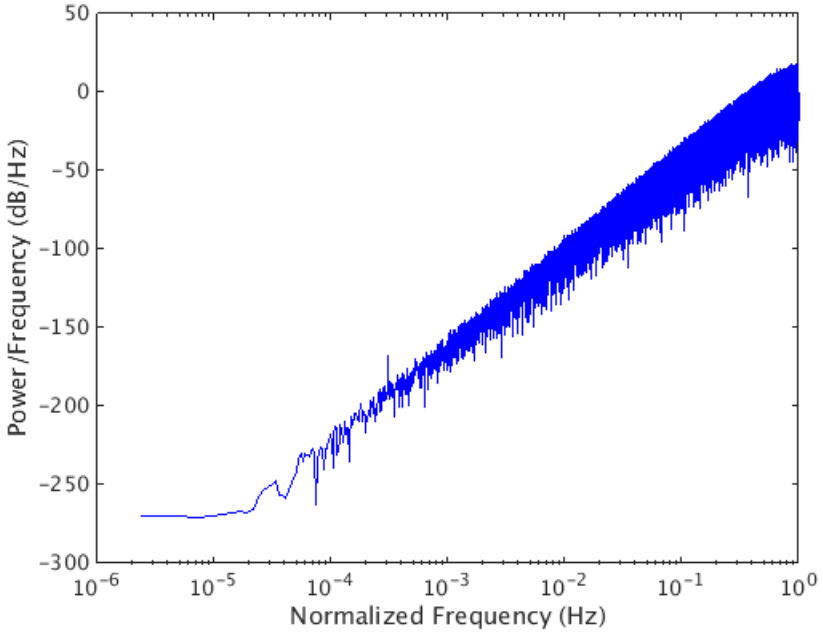

Figure 6.45: PSD for $f_c = 315$ MHz and minimum data rate with 30 kHz deviation frequency

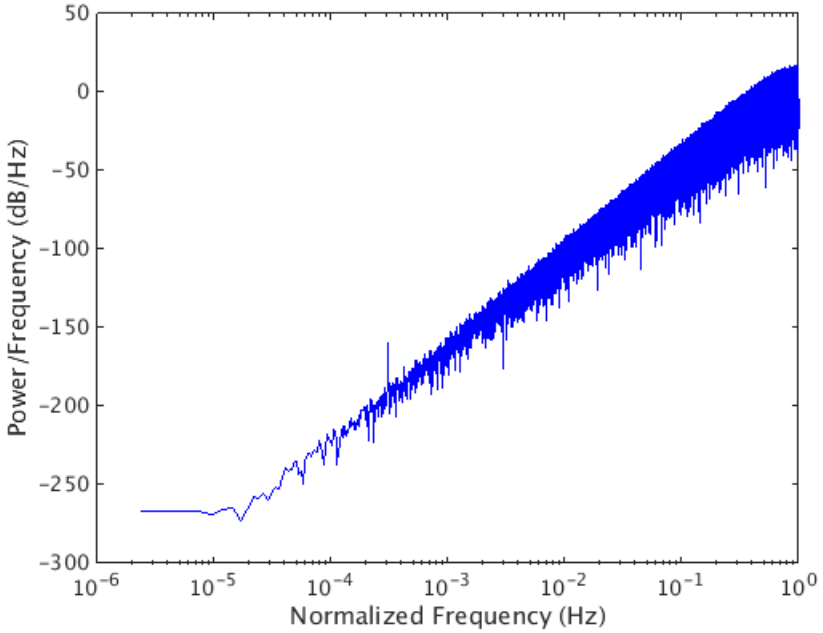

Figure 6.46: PSD for $f_c = 315$ MHz and minimum data rate with 80 kHz deviation frequency

In figure 6.41 and 6.44 everything looks fine and as expected. In all other plots an unwanted spur at low frequencies takes place (figure 6.42, 6.43, 6.45 and 6.46). The reason for it is the fact that for the SP MASH DDSM, the output from the previous stage is fed forward to the next one. As can be seen in section 4.5, for a constant input, the additional components in the z-domain output equation due to output feed-forwarding can be safely neglected, but when a variable input is considered the assumption does not hold. In equation 6.7 the z-domain output is displayed.

$$\begin{aligned} Y(z) = &\{\frac{1}{M} + \frac{1}{M^2}(1 - z^{-1}) + \frac{1}{M^3}(1 - z^{-1})^2\}X(z) \\ &+(1 - z^{-1})^3\{\frac{1}{M}Q_3(z) + \frac{1}{M^2}Q_2(z) + \frac{1}{M^3}Q_1(z)\} \end{aligned} \tag{6.7}$$

In figure 6.47 an example is displayed. The red curve is the shaped quantization noise output of the SP MASH 1-1-1 DDSM model. The blue curve is the whole output of the DDSM including the input and the shaped quantization noise. It can be observed that the error in the red curve is the by the factor $\frac{1}{M^2}$ down scaled first spike of the input. The other spikes do not appear in the PSD because every spike is down scaled by the factor $\frac{1}{M^2}$ and according to the fact that the quantization error rises 60 dB/dec and the input falls 20 dB/dec the other down scaled spikes are hidden by the rising quantization noise. From equation 6.7 it can be observed that also the third stage adds a down scaled input to the output of the DDSM but the contribution is not

visible in the PSD, because it lies below the quantization noise.

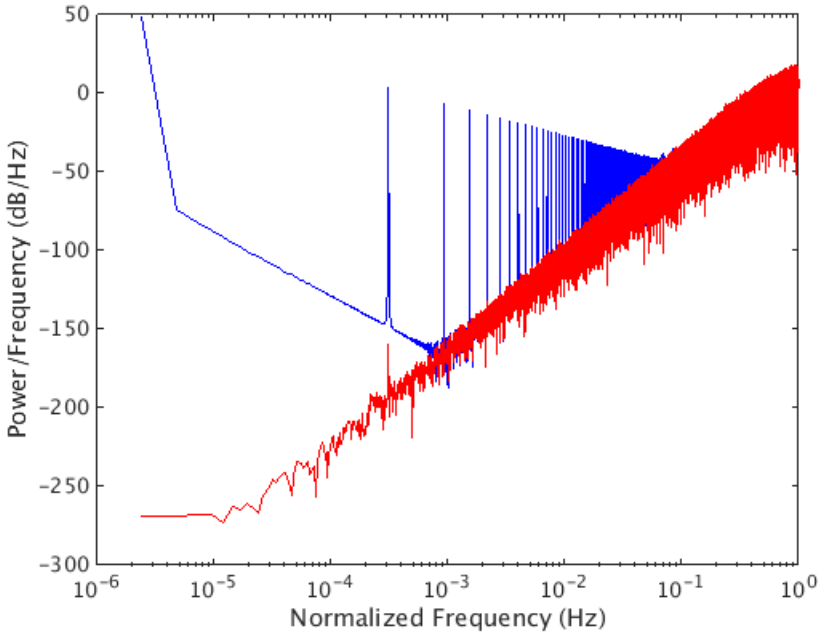

Figure 6.47: PSD for $f_c = 434$ MHz and minimum data rate with 80 kHz deviation frequency - blue: input and shaped quantization noise, red: shaped quantization noise

In figure 6.41 and 6.44 with maximum data rate nothing can be observed because the contribution is to small and due to this not visible in the PSD like displayed in figure 6.48.

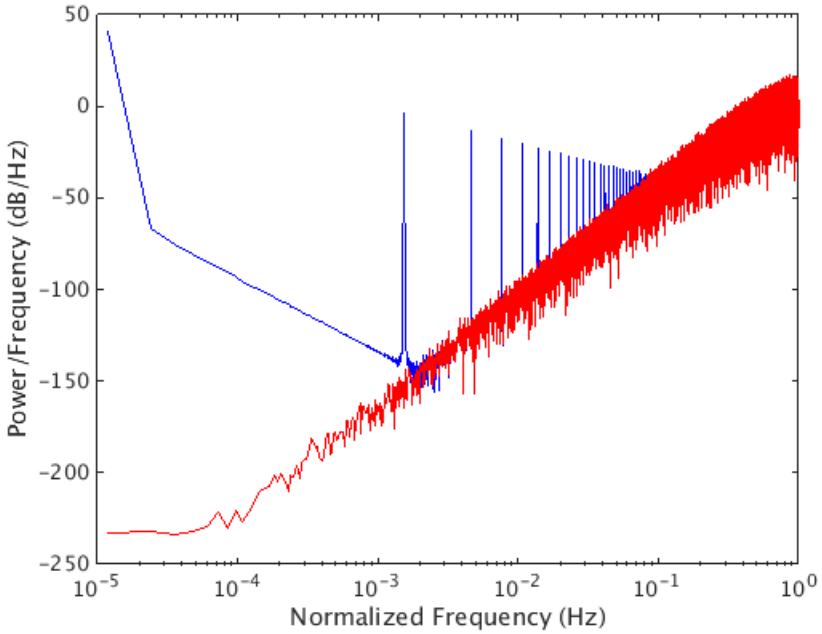

Figure 6.48: PSD for $f_c = 434$ MHz and maximum data rate with 80 kHz deviation frequency - blue: input and shaped quantization noise, red: shaped quantization noise

The conclusion of the structure test with variable input is that there is an additional error in the shaped quantization noise due to the additional forwarded input signal but it is no problem for the function of the structure, because in a real situation, the input and the quantization noise are not separated and thereby the additional error is hidden behind the PSD of the variable input. The conclusion is that the SP MASH 1-1-1 with 17-bit is the selected structure for implementing the DDSM.

7 Implementation of the new Approach

To use the selected structure on a silicon device it is necessary to implement it in a so called Hardware Description Language, such as Verilog, VHDL or System Verilog (SV). The generated code describes the behaviour of the model. Because Infineons common practice is to use SV, the fact that the syntax is not as strict as for VHDL and the author's previous experience, the DDSM is implemented in System Verilog.

The specification for the implementation was to generate a block which is fully parametrizable. The IP block parameters are its input value, the number of bits per stage and the number of stages, like displayed in figure 7.1. The SV code generates the necessary model automatically based on the SP MASH DDSM structure. This parametrizable approach is used because the idea is to reuse the block in different projects with different requirements. Additionally, it makes it possible for the system integrators to reuse the IP block as a "black box".

Figure 7.1: Block diagram of the generated SV model

7.1 SV Code Presentation

In figure 7.2 the flow of the SV model is coarsely represented.

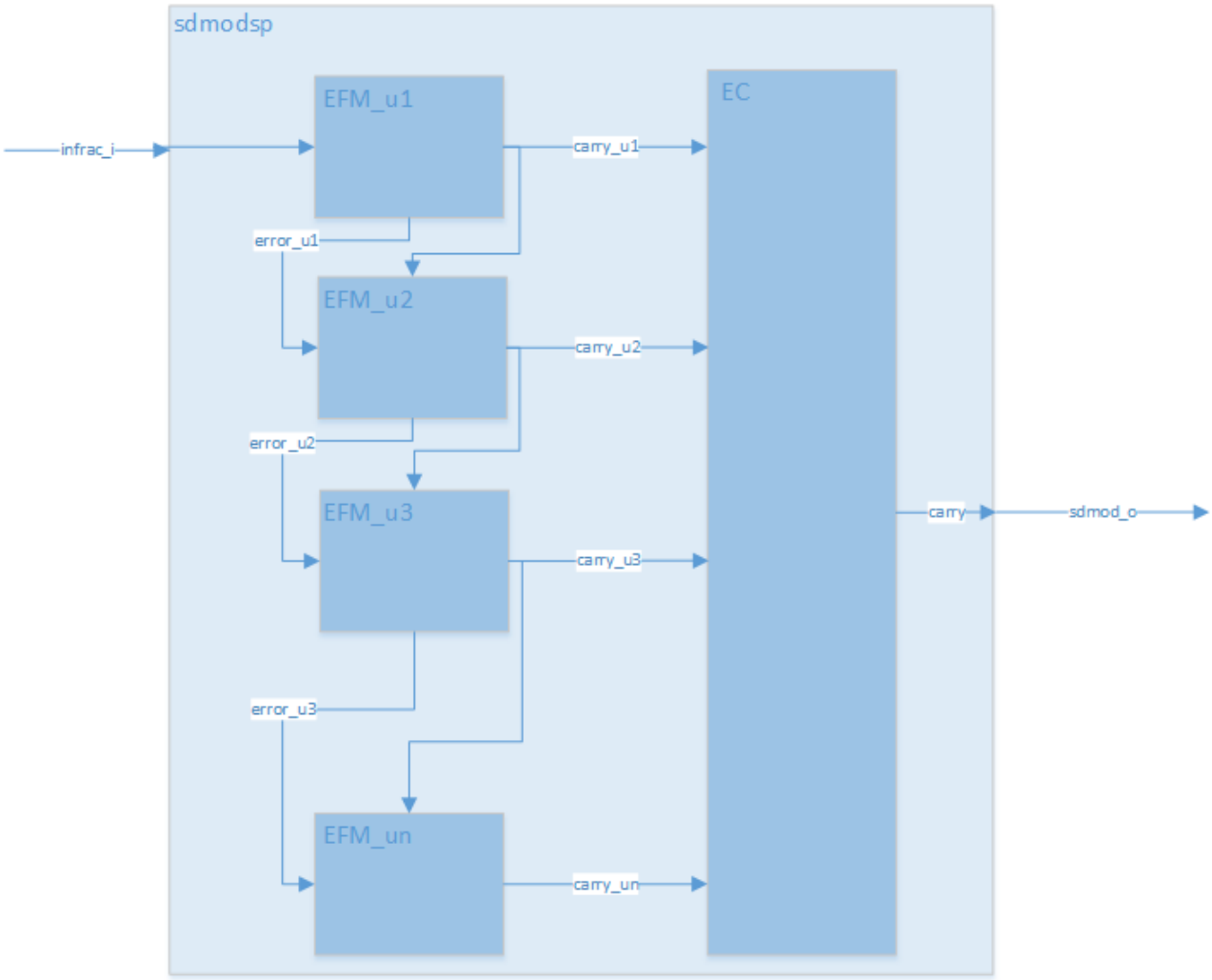

Figure 7.2: Coarse representation of the SV model

The model is split into three parts. The so called main, named *sdmodsp*, where the other parts are instantiated, namely, the error cancellation network (EC) and the error feedback modulators (EFM). According to the input values *number of bits* and *number of stages* the bitwidth of signals inside the block are defined.

```
input logic[numberBits-1:0] infrac_i,
output logic[numberStages:0] sdmod_o,
```

The *sdmodsp* module constitutes the highest level. It is the level where the blocks receive signals from outside and where the generated signal leaves the DDSM. In here, two other types of modules are instantiated. Depending on the input value *number of stages* the *generate* function generates EFM instances. The corresponding code is displayed:

```
genvar i;

generate
  for (i=0; i<numberStages; i=i+1) begin: sigma_path
    EFM #(numberStages, numberBits) EFM_u (
     .clk(clk_i),
        .reset_n(reset_n_i),
        .infrac(input_inf[i]),
        .dither_carry(carry_out[i]),
        .enable(enable),
        .error(input_inf[i+1]),
        .carry(carry_out[i+1]));
  end
endgenerate
```

The EFMs are connected chain-wise. The input to the DDSM block is the input to the first EFM. The negative error and the output from the first EFM are the inputs to the second EFM and so on. The generated carrys from each stage are connected to the *EC* module. With the # sign parameters are fed forward from one module to the next.

```
EFM #(numberStages, numberBits) EFM_u ( ...
```

In each EFM instance the value for EFM_int is calculated which is the result from the addition of the error from the last stage, the carry from the last stage and the returned error from the current stage. EFM_int always has one bit more than defined through the variable *number of bits*, because the MSB is the carry from each EFM. The other bits are the error.

```
logic[bits:0] efm_int;
...
..
.
begin
  if (enable == 1)
  begin
    efm_int = infrac + error + dither_carry;
    carry <= efm_int[bits];
    error <= efm_int[bits-1:0];
  end
end
```

In the first process of the *EC* module the carry from each stage gets n times delayed, where n depends on the stage the carry comes from. For example the carry from the first stage has to be delayed as long as the input needs to be shifted through all EFMs and back up the ECN to the adder next to the output. In the second process the different carry signals are combined, as presented in chapter 3.2.2, to get the final output of the DDSM.

Part of the First Process:

```
assign carry_delayed[0] = carry;

begin
  if (enable == 1)
  begin
    for (index=1; index<stages+1; index=index+1)
    begin
      carry_delayed[index] <= carry_delayed[index-1];
    end
  end
end
```

Part of the Second Process:

```
begin
  if (enable == 1)
  begin
    for (index=0; index<stages-1; index=index+1)
    begin
      co_int [index+1] = carry_delayed[1+index][stages-2-index] +
            co_int [index] - co_D [index];
      co_D [index+1] <= co_int [index+1];
    end
    sdmod <= co_int [stages-1];
  end
end
```

8 Simulation

In this chapter the performed simulation with the program SimVision (Cadence Design Systems Inc., San Jose, USA) and the corresponding results are presented. SimVision is used because it is the tool Infineon is working with. The simulation is necessary to verify the behaviour of the build SV model. The idea is to compare each output value of the SV model with each output value of the Matlab model and if the difference is zero the model is working right. The Matlab model can be used as reference, because in chapter five the correct function of the model was evaluated.

There was only one problem. How can an SV model and a Matlab model be compared? Mathworks offers a solution. Simulink and Matlab have an implemented function to export models. The function builds the model in C code and instances the code in an SV module [24].

In order to verify a testbench is written where the SV model of the new approach and the module controlling the C code are instantiated. In figure 8.1 a block diagram of the testbench is displayed.

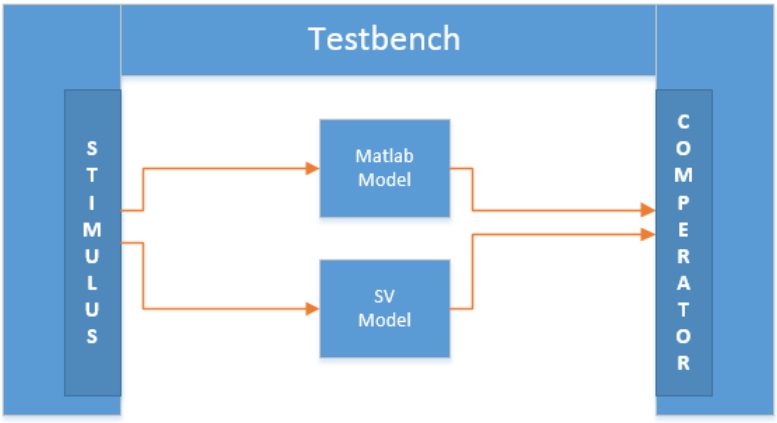

Figure 8.1: Block diagram of the SV testbench

To use the Simulink function the current model has to be modified a bit because it is necessary that the model is a stand alone block with no connection to a Matlab file. In figure 8.2 the

highest level of the new model is displayed. The *Constant* and *ToWorkspace* blocks are replaced by *In1* and *Out1*. This blocks define the input and the output of the model. Additional two converters *Bit to Integer Converter* and *Integer to Bit Converter* are necessary because the Simulink model works with integers and the SV testbench uses binary values. Inside the *DSM_SP* block the well-known SP MASH 1-1-1 DDSM structure is placed.

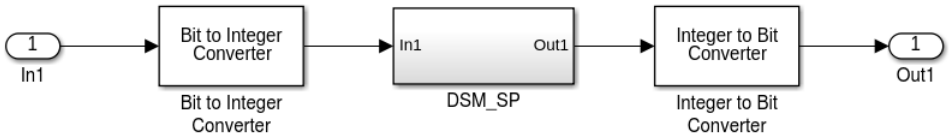

Figure 8.2: Highest level of the exported SV model

In the following important parts of the testbench are displayed.

Clock Generator:

```
timeunit 1ns;
logic clk_i_s;
localparam freq = 26000000;
localparam timeFactor = 1000000000;

real period = timeFactor/freq;

always #(period/2) clk_i_s = ~clk_i_s;
```

The process generates the reference clock (26 MHz).

Process Comparing the Outputs:

```
always@ (posedge clk_i_s)
begin
  if (enable_m_s == 0)
    check = 'b1;
  else
  begin
    for (index=0; index<4; index=index+1)
    begin
      if (sdmod_o_s[index] == m_output_s[3-index])
        check[index] = 'b0;
```

```
      else
        check[index] = 'b1;
    end
  end
end
```

The process checks if each bit of the output signal from the new approach SV model is equal to the output signal of the Matlab model. If the comparison is true the *check* signal is zero, otherwise it is one.

Instantiated Modules:

```
    // SV model
    sdmod_sp #(numberStages, numberBits) u_sdmod_sp (
        .reset_n_i(reset_n_i_s),
        .clk_i(clk_i_s),
        .infrac_i(infrac_i_s),
        .enable_i(enable_i_s),
        .ds_off_i(ds_off_i_s),
        .sdmod_o(sdmod_o_s));

    // Matlab model
    MASH_111_spur_free_dpi u_MASH_111_spur_free_dpi (
        .clk(clk_matlab),
        .reset(reset_n_i_s),
        .enable(enable_m_s),
        .ds_off(ds_off_i_s),
        .MASH_111_spur_free_U_In1(infrac_i_m),
        .MASH_111_spur_free_Y_Out1(m_output_s));

    //old VHDL model
    sdmod_wrapper u_sdmod_wrapper (
        .reset_n(reset_n_i_s),
        .clk(clk_i_s),
        .infrac(infrac_old_s),
.enable_i(enable_i_s),
.ds_off_i(ds_off_i_s),
        .sdmod(sdmod_old_s));
```

Also the model of the currently used structure is instanced in the testbench, because the model has to be simulated and synthesised to get the values for power and area which are discussed in chapter nine. Therefore an SV wrapper is written, because the old model is in VHDL and

not SV. A wrapper is an SV module were the VHDL model is instantiated and the inputs and outputs are connected.

Code for the SV Wrapper

```
module sdmod_wrapper(
    input wire reset_n,
    input wire clk,
    input wire [20:0] infrac,
    input enable_i,
    input ds_off_i,
    output wire [4:0] sdmod
    );

    sdmod_old u_sdmod_old(
        .reset_n_i(reset_n),
        .clk_i(clk),
        .infrac_i(infrac),
.enable_i(enable_i),
.ds_off_i(ds_off_i),
        .sdmod_o(sdmod));

endmodule
```

At first the simulation was started with one fixed input value to check if the testbench and the SV model are working right. For example:

```
//input = 0.5
infrac_i_s = 17'b10000000000000000;
```

When the simulation was successful and the *check* signal was always zero the testbench changed. The next task was to check if the model works correctly for all input values from 0.01 till 0.99 in 0.01 steps. Therefore the following process was used were after a defined time, in this case 10 μs, the input is changed to the next value automatically.

Variable Input:

```
localparam timePerInput = 10000;
infrac_i_s = 17'b00000010100011110;
i = 0;

always@ (posedge clk_i_s)
begin
  if (i == timePerInput)
  begin
    infrac_i_s = infrac_i_s + 17'b00000010100000100;
    for (index=0; index<numberBits; index=index+1)
    begin
      infrac_i_m[numberBits-1-index] = infrac_i_s[index];
    end
    i = 0;
  end
  else
    i = i + 1;
end
```

During the first simulation tries one problem occurred. It was not possible to compare the output values because there is a delay between the two signals. This delay occurred due to the fact that the SV model performs every positive clock edge an operation. This means that it lasts four clock cycles for the SP MASH 1-1-1 DDSM to generate the first output value. Compared to that the Simulink model only needs one clock cycles to generate the first output, because after the model is started the value is calculated in the same clock cycle. To fix the problem a constant delay was implemented applying each time after starting the simulation and changing the input value.

From the simulation only one figure (8.3) as an example is presented because when the model is working correctly not very much relevant for the thesis can be observed. In figure 8.3 the *check* signal comparing the signals *sdmod_o_s* and *m_output_s* is displayed. The *check* signal is always zero. Additionally the input signal (input = 0.5) and two different enable signals realizing the delay as described above are presented.

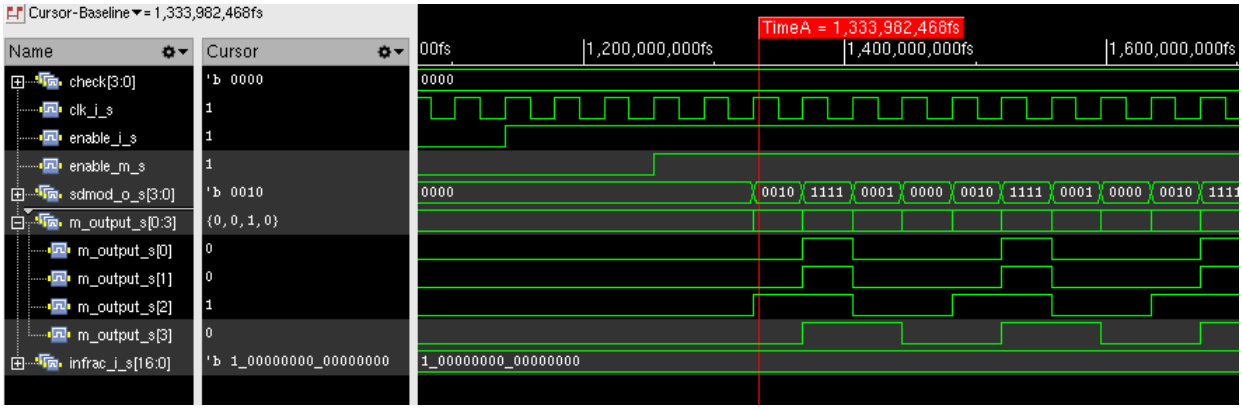

Figure 8.3: Example plot of the simulation

The conclusion of the simulation is that the SV model of the new approach is working correctly for every input value from 0.01 till 0.99 in 0.01 steps.

9 Synthesis

In order to do proper assessment regarding improvements of power and area a synthesis needs to be done before. During synthesis the logic described in System Verilog is mapped to logic gates like AND, OR, XOR e.g.. In general Synopsys Design Compiler is used to create such gate level netlists.

The focus of this work is on getting an early result regarding area/power and a valuation of the quality of the written System Verilog code. For such assessment SpyGlass (Synopsys Inc., California, USA) is the proper tool of choice. SpyGlass also does basic synthesis, linting and clock domain crossing to give advice how to improve the quality of the code. Due to the fact that it is a kind of review tool it also generates the gate level netlist in a well visualized form. So someone can easily identify the corresponding RTL structure in the different hierarchies.

9.1 Lint

Lint is a way of early design analysis. It is required because "*inefficiencies during RTL design usually surface as critical design bugs during the late stages of design implementation.*" [25].

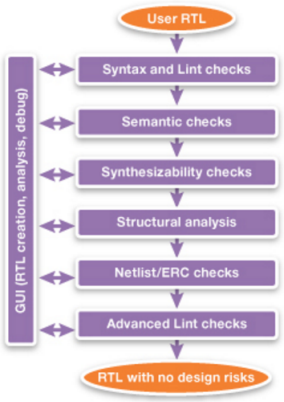

Figure 9.1: SpyGlass lint flow [25]

Figure 9.1 represents an overview about what the SpyGlass tool does respective to lint.

In the following list a short overview about the problems linting finds is given [25].

- basic problems for e.g.: floating input, width mismatch
- simulation problems e.g.: incomplete sensitivity list, incorrect use of blocking and non-blocking assignments, potential functional errors
- finds problems in the structure of the design that influence the post-implementation functionality or the performance of the design e.g.: multiple drivers, synchronous/asynchronous use of reset

9.2 Clock Domain Crossing - CDC

"*CDC issues have become a leading cause of design errors. Such errors can add significant time and expense to the design-and-debug cycle, and may even find their way into silicon, necessitating costly respins*" [26]. The SpyGlass CDC test for example finds all clocks and resets in a design, checks if clocks are synchronous to avoid metastability and proves if all flip-flops get a clock and only one.

9.3 Conclusion

The conclusion of the synthesis is that all lint and CDC checks are passed and the power analysis can begin. In the following figures (9.2, 9.3, 9.4 and 9.5) the produced gate description with its different levels and a list of all used components is presented. In table 9.1 is described which logical functions accord to the cell names from figure 9.5.

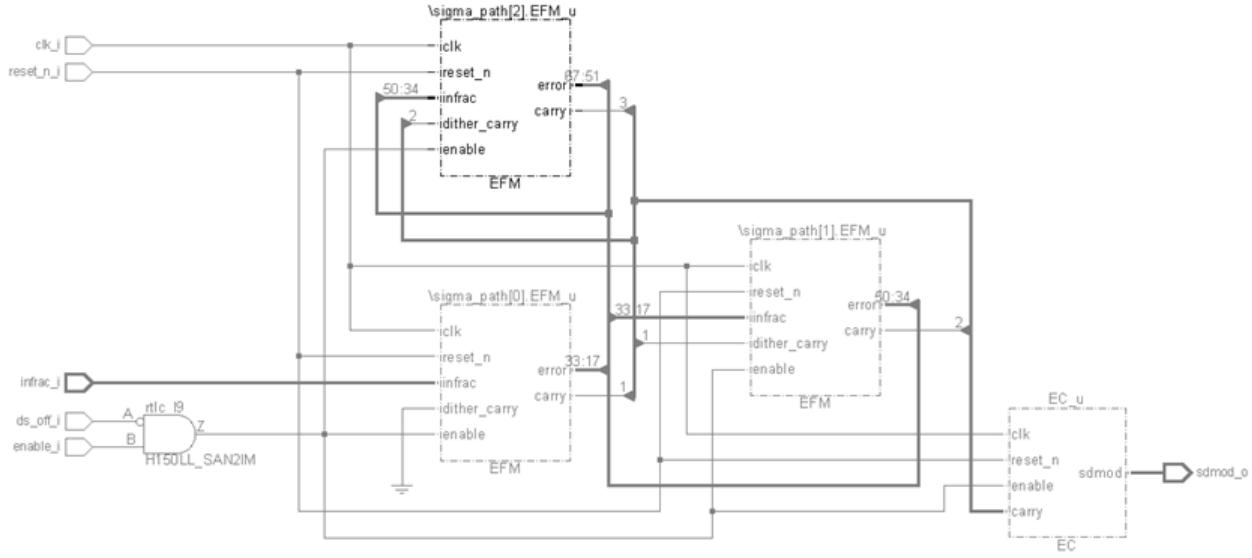

Figure 9.2: Gate description sdmod_sp

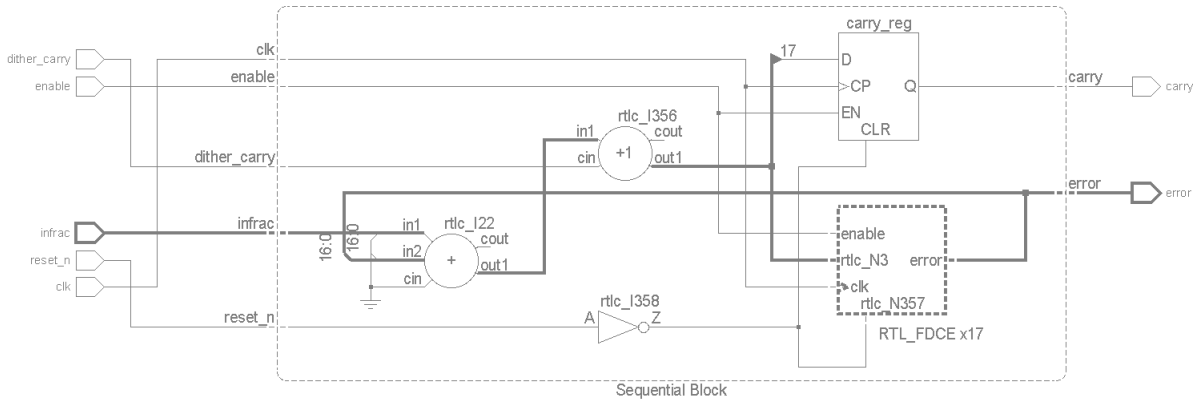

Figure 9.3: Gate description EFM_u block

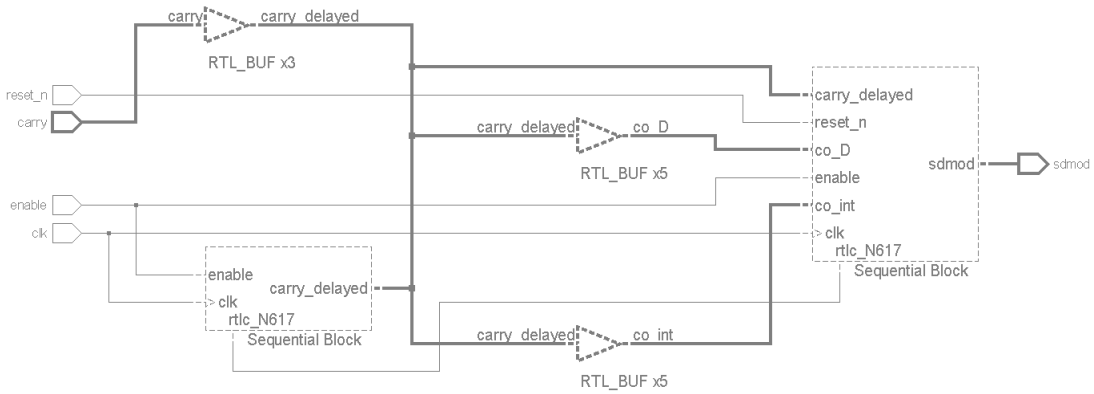

Figure 9.4: Gate description EC_u block

```
Cell Name            View              Total
                                       Cell Count
-------------------------------------------------

H150LL_SAN2IM        Func Available    1
H150LL_SANR1YM       Func Available    1
H150LL_SANR2YM       Func Available    1
H150LL_SAOR1         Func Available    1
H150LL_SBUFX1        Func Available    4
H150LL_SCLKLTOA8     Func Available    4
H150LL_SENAM         Func Available    30
H150LL_SEOAM         Func Available    79
H150LL_SFA1          Func Available    2
H150LL_SFD2QSA       Func Available    66
H150LL_SHA1          Func Available    2
H150LL_SIVYA1        Func Available    6
H150LL_SMX2AM        Func Available    3
H150LL_SMXA12        Func Available    24
H150LL_SMXI2M        Func Available    24
H150LL_SNR2X         Func Available    2
H150LL_SOR2M         Func Available    1
```

Figure 9.5: List of all components used in the gate description

Table 9.1: Explanation of the cell names

Cell Name	Description	Formula Description
SAN2IM	2-input and gate with one inverted input	Z = ((!A)*B)
SANR1YM	2-input AND into 2-input NOR	Z = (((!A)+(!B))*(!C))
SANR2YM	double 2-input AND into 2-input NOR	Z = (((!A)+(!B))*((!C)+(!D)))
SAOR1	2-input AND into 2-input OR	Z = ((A*B)+C)
SBUFX1	Buffer cell	Z = A
SCLKLTOA8	gated clock cell latched by CP with TE input	
SENAM	2 input EXNOR	Z = (!(A*B))
SEOAM	2 input EXOR	Z = (A*B)
SFA1	1 bit FULL-Adder	
SFD2QSA	posedge triggered SCAN-DFF with async. act. low Reset	
SHA1	1 bit HALF-Adder	
SIVYA1	Inverter	Z = (!A)
SMX2AM	2 bit muliplexer	Z = ((S*A1)+((!S)*A0))
SMXA12	2 bit multiplexer, with inverted data input	Z = ((S*(!A1))+((!S)*A0))
SMXI2M	2 bit multiplexer, with inverted data output	Z = ((((!S)+(!A1))*(S+(!A0)))
SNR2X	2 input NOR Gate	Z = ((!A)*(!B))
SOR2M	2 input OR Gate	Z = (A+B)

10 Power analyse

With SpyGlass a power and area analysis for the new approach DDSM is performed. The tool not only measures the power and area consumption but also gives advice how these values could be reduced. For analysis a .fsdb file, which is generated during the simulation step, is necessary. The file contains the signal activity of all wires within the gate level netlist. Such signal information is used to determine the current power consumption of each instantiated cell. The power information regarding leakage and dynamic charge consumption is stored in a lib file. The lib file contains also area information for each cell, which is taken for the area calculation of SpyGlass.

To get the opportunity to compare the new with the old DDSM model respective to power and area also the old approach is synthesised and analysed with SpyGlass. For comparison three different input values X are chosen.

- X = 0.1 - often used in literature
- X = 0.5 - worst case value
- X = 0.6923 - project specific value

In table 10.1 the results for the area calculation are presented. With the new approach 17.4% of area can be saved compared to the old DDSM.
The results for power calculation are displayed in table 10.2. It can be observed that dependent on the input value the possible hardware savings change from about 15% to nearly 30%. One explanation for the different percentages for power saving could be that the improvement in the performance of the DDSM for bad values like 0.5 from the old to the new approach is bigger than for other input values.

Table 10.1: Comparison of area for the new and old DDSM

old approach [μm^2]	6164.8
new approach [μm^2]	5092.8
saving [μm^2]	1072
saving [%]	17.4

Table 10.2: Comparison of power for different input values for the new and old DDSM

input = 0.5	old approach [W]	2.868e-04
	new approach [W]	2.079e-04
	saving [W]	0.789e-04
	saving [%]	27.5
input = 0.1	old approach [W]	3.230e-04
	new approach [W]	2.759e-04
	saving [W]	0.471e-04
	saving [%]	14.6
input = 0.6923	old approach [W]	3.247e-04
	new approach [W]	2.767e-04
	saving [W]	0.48e-04
	saving [%]	14.8

Summing up it can be said that it is possible to save at least 14.6% of power and 17.4% of area with the 17-bit SP MASH 1-1-1 DDSM compared to the 21-bit MASH 1-1-1 DDSM with odd initial condition. In figure 10.1, 10.2, 10.3 and 10.4 the results from table 10.1 and 10.2 are graphical compared.

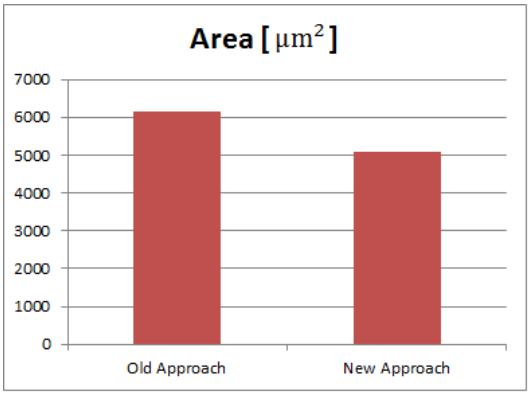

Figure 10.1: Graphical comparison of area for old and new approach

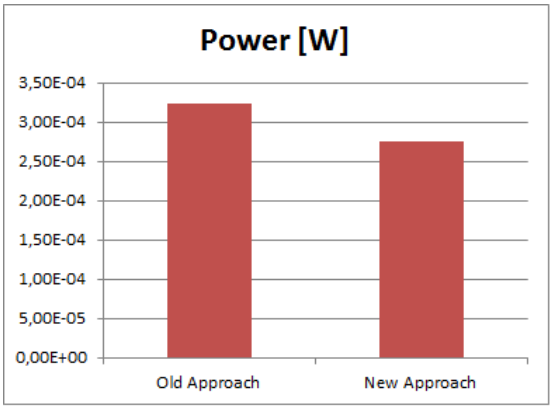

Figure 10.2: Graphical comparison of power with input = 0.1 for old and new approach

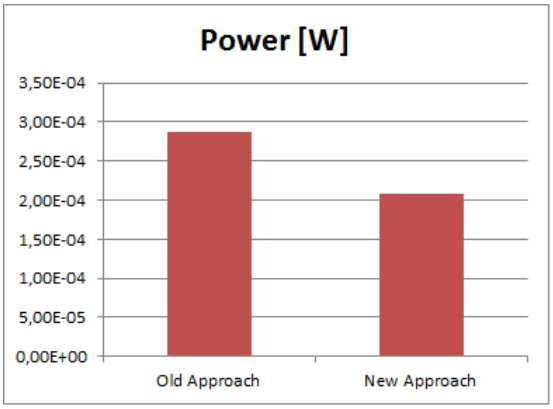

Figure 10.3: Graphical comparison of power with input = 0.5 for old and new approach

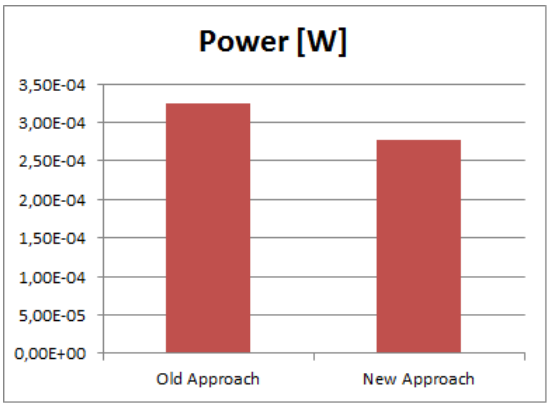

Figure 10.4: Graphical comparison of power with input = 0.6923 for old and new approach

11 Digital Delta Sigma Modulators in Medicine

Since 1949 when Norman J. Holter invented telemetric transmission of electrocardiograms - ECG [12] telemetry in medicine is an ever-growing field. It denotes diagnostic and therapeutic data transmission between the place where the information is measured and the place where it is processed [13]. The possibility to transmit medical data opens a whole new spectrum of applications. Telemetry in medicine can be used for consulting experts anywhere in the world during surgeries, transferring data from patients home to the doctor, distributing informations in hospitals [13], measuring multi channel ECGs, transmitting blood pressure and $Sp0_2$ satiation results [12]. In figure 11.1 an example for consulting an expert during a surgery is presented.

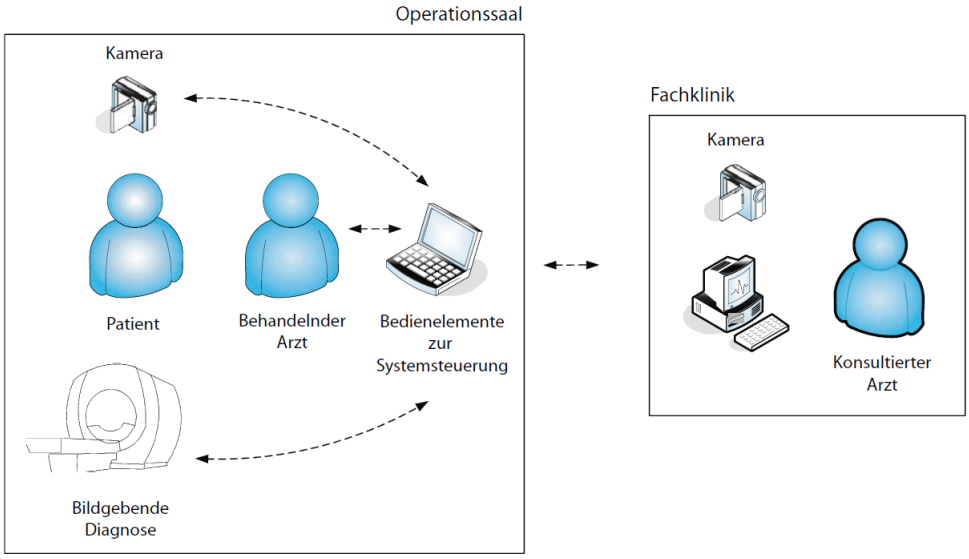

Figure 11.1: Example for consulting an expert during a surgery [13]

In general telemetry can be split in two categories, namely wired and wireless. In this thesis only wireless telemetry is considered.

"*The use of wireless medical telemetry in hospitals is expanding as one way to help meet the ever-rising cost of health care.*"[15].

In order to deliver data by means of wireless transmission, one commonly used method is to employ radio frequency (RF) transmitters. In figure 11.2 a telemedicine system using RF technology is displayed. The system consists of a measuring device, a patient monitor including the RF receiver and a server. The mobile measuring device transmits data via RF to the patient monitor which forwards it to the data server where the information is stored [12].

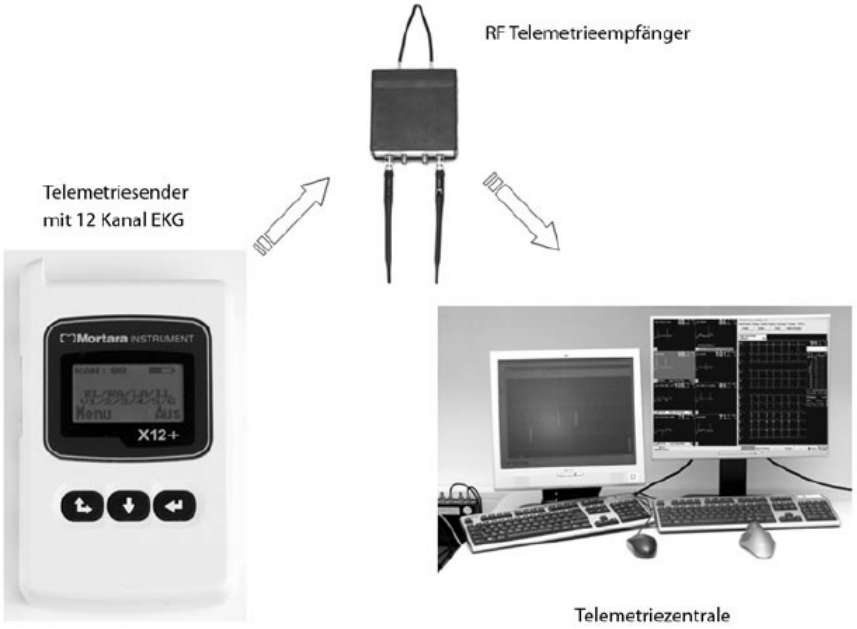

Figure 11.2: Example for a telemetric system using RF technology [12]

Another field of application for RF transmitters is implantable devices like pacemakers, cardioverter-defibrillators, deep brain stimulators and blood sugar monitors [13]. By using these, it is on the one hand possible to permanently monitor the device functions, making its use safer, and on the other hand to provide up-to-date diagnostic information of the patient to modify the medication if it is necessary [12]. In the following section telemedicine at home using implants transmitting data with RF is discussed.

11.1 Telemedicine at Patients Home

The idea is to take care of the patient at home by automated means, as far as possible. The doctor only checks the parameters. The expected advantages using such a system are relief of the patient and the doctor, optimal device settings and precocious problem detection [16]. Several studies came to the conclusion that by using telemedicine at home quality can be improved while reducing costs [12]. In literature different studies are comparing standard follow-up-care and telemetric controls. One point of concern is how to make sure the data transfer is secure. In [14] they point out that many researches checked the technical security for data transfer via remote enquiry and came to the conclusion that the data safety is guaranteed.

The randomised, controlled CONNECT-Study from 2011 points out that the difference between the number of diagnostic hits for standard follow-up-care and telemetric controls is huge. 575 hits by 172 patients for the telemetric controls compared to 391 hits by 145 patients for standard follow-up-care. Compared to standard follow-up-care it was also possible to reduce the average time between problem detection and therapy change from 22 down to 4.6 days [14]. By analysing 1739 follow-up examinations from 169 implantable cardioverter defibrillator (ICD) patients the result was that only in 6 percent a modification of the program was necessary. In 94 percent a telemetric control had been enough [14]. Telemetric controls at home could be a huge time and cost saving for the doctor and the patient which could lead to an increase in life quality of the patient and less stress for the medical professionals.

To date, two different types of systems for transmitting data are used. The company Biotronik provides implants for example that transmit every day a reduced data set which is evaluated by the server. Other systems like the CareLink system from Medtronic only transmit information to the base station if the implant recognizes an alert. This has the advantage that the battery lasts longer because there are less transmissions and the information overload is reduced.

The telemedicine at home systems can be used for telemonitoring and telemetric follow-up care.

11.1.1 Telemetric Follow-up Care

The process looks in detail as displayed in figure 11.3 (a). When the patient gets home from hospital after implanting the ICD the doctor defines control intervals. In the night when the patient sleeps and the control date is reached, the implant measures the defined parameters

and transmits it wirelessly to the base station near the patient. Thereby the maximum range between patient and base station is from 10-25 meters [12]. From the base station the data is transferred via analogue telephone or mobile phone to the server. The next day the doctor logs in to the server and checks the parameters. If the doctor diagnosis is non eventful, the system sends the patient an SMS with this information otherwise the doctor contacts the patient to make a personal appointment. In figure 11.3 (b) the problem scenario is displayed. If the implant recognizes a problem it transmits an alert to the base station. The station sends it to the server which alerts the doctor, who contacts the patient. The system also offers the opportunity to start a measurement manually if the patient believes there is a problem [17].

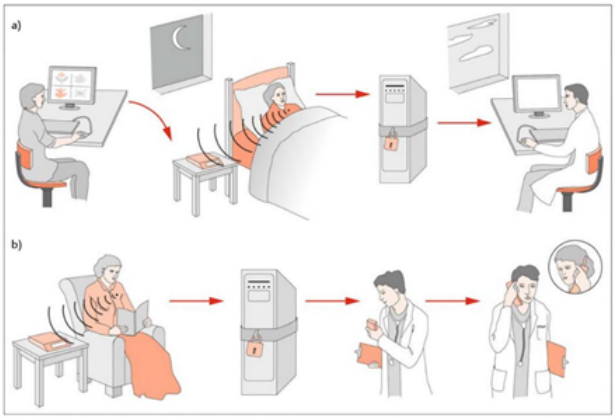

Figure 11.3: Flow of the telemetric control approach [14]

11.1.2 Telemonitoring

The difference to telemetric follow-up care is that the system permanently measures the parameters and sends them to the server. The server checks the data and if there is a problem it informs the doctor, who then calls the patient [14].

11.2 Summary

Telemetry is a growing field in medicine. When talking about mobile devices and implants, patient comfort is a critical topic. In the case of mobile applications, the device should be as small and light as possible, because the patient always has to carry it on his/her body. Implants using RF technology provide a huge advantage compared to inductive coupling because the data transmission is executed automatically and no reading device has to be applied in physical proximity to the implant [7]. When using the newly designed DDSM for RF transmitters it is possible to reduce the area and the power of the device. Which leads to longer lasting batteries

and smaller mobile devices and implants. When the battery lasts longer time between follow-up surgeries extends respectively other functions can use more power and when the implant is smaller less area in the body is necessary respectively more space for other functions is left.

12 Conclusion

In this thesis, different approaches for realising DDSM with the goal of reducing power and area are evaluated. Not all of them are of practical use. A specific modulator architecture has been implemented, considering the requirements and specifications of a real silicon device, where it is to be integrated. By using the new DDSM it is possible to reduce the bits per stage from 21 to 17 while at least maintaining the performance. Additionally, the performance is equal for all input values and initial conditions. Numerically this means it is possible to save 17.4% of area and at least 14.6% of power with the new 17-bit SP MASH 1-1-1 DDSM compared to the 21-bit MASH 1-1-1 DDSM with odd initial condition. In general, the theoretical hardware and power consumption reduction is greater than the implemented design, but practical limits due to the requirements of the device where the modulator is to be integrated, effectively impose a lower bound on the area/power savings achieved.

13 Bibliography

Literature

[1] K. Hosseini and M. P. Kennedy *Minimizing Spurious Tones in Digital Delta-Sigma Modulators. 2011.*

[2] M. Werner *Nachrichten-Übertragungstechnik: Analoge und digitale Verfahren mit modernen Anwendungen. 2006.*

[3] K. Hosseini and M. P. Kennedy *Maximum Sequence Length MASH Digital Delta-Sigma Modulators. 2007.*

[4] J. Song and I.-C. Park *Spur-Free MASH Delta-Sigma Modulation. 2010.*

[5] M. J. Borkowski, T. A. D. Riley, J. Hakkinen and J. Kostamovaara *A practical delta-sigma modulators design method based on periodical behavior analysis. 2005.*

[6] K. Hosseini, M. P. Kennedy and M. Cathal *Calculation of Sequence Lengths in MASH 1-1-1 Digital Delta Sigma Modulators with a Constant Input. 2005.*

[7] Z. Ye and M. P. Kennedy *Hardware reduction in digital delta-sigma modulators via error masking - Part I: MASH DDSM. 2009.*

[8] B. Fitzgibbon, M. P. Kennedy and F. Maloberti *A novel implementation of dithered digital delta-sigma modulators via bus-splitting. 2011.*

[9] C.-Y. Yao and C.-C. Hsieh *Hardware simplification to the delta path in a MASH 111 delta-sigma modulator. 2009.*

[10] J. Song *Hardware Reduction of MASH Delta-Sigma Modulator Based on Partially Folded Architecture. 2015.*

[11] M. J. Ryan and M. Frater *Communications and Information Systems. 2002.*

[12] T. Hilbel, T. M. Helms, G. Mikus, H. A. Katus and C. Zugck *Telemetrie Szenarien im klinischen Umfeld. 2008.*

[13] R. Kramme *Medizintechnik Verfahren - Systeme - Informationsverarbeitung. 2011.*

[14] G. Fröhlig, J. Carlsson, J. Jung, W. Koglek and B. Lemke *Herzschrittmacher- und Defibrillator-Therapie: Indikation – Programmierung. 2013.*

[15] J. F. Dyro *Clinical Engineering Handbook. 2004.*

[16] U. Nixdorff *Check-Up-Medizin: Prävention von Krankheiten - Evidenzbasierte Empfehlungen für die Praxis. 2006.*

[17] F. Goss *Praktische Telemedizin in Kardiologie und Hypertensiologie. 2009.*

[18] D. Morschhäuser, W. Fischer and M. Jakob *Praxis der Herzschrittmacher-Nachsorge: Grundlagen, Funktionen, Kontrolle. 2011.*

Online Sources

[19] http://de.mathworks.com/help/signal/ug/psd-estimate-using-fft.html. Visited on 6 July 2016.

[20] http://de.mathworks.com/help/signal/ref/periodogram.html. Visited on 6 July 2016.

[21] http://de.mathworks.com/help/comm/ref/comm.pnsequence-class.html. Visited on 8 July 2016.

[22] http://de.mathworks.com/help/matlab/ref/polyfit.html. Visited on 8 July 2016.

[23] de.mathworks.com/help/matlab/ref/polyval.html. Visited on 8 July 2016.

[24] http://de.mathworks.com/help/dsp/ug/generate-code-from-simulink.html. Visited on 12 May 2016.

[25] https://www.synopsys.com/Tools/Verification/static-formal-verification/Pages/spyglass-lint.aspx. Visited on 14 July 2016.

[26] https://www.synopsys.com/Tools/Verification/static-formal-verification/Pages/spyglass-cdc.aspx. Visited on 14 July 2016.