UNIVERSITY OF TWENTE

MASTER'S THESIS

Low Power ASIC Design of a DDPSK Demodulator

Author:
P.C. Dijkshoorn

Supervisors:
dr. S. Safapourhajari
dr.ir. A.B.J. Kokkeler
dr.ir. R.A.R. van der Zee

RS
EEMCS

December 3, 2021

Contents

List of Figures

List of Tables

List of Abbreviations

ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
BBC	Baseband Conversion
BER	Bit Error Rate
BLE	Bluetooth Low Energy
CC	Clock Controller
CIC	Cascaded Integrator-Comb
CLK	Clock
DDPSK	Double Differential Phase-Shift Keying
DMD	Demodulator
DSP	Digital Signal Processing
FIR	Finite Impulse Response
FSK	Frequency Shift Keying
HVT	High-Threshold Voltage
IoT	Internet of Things
LPWAN	Low-Power Wide Area Network
LSB	Least Significant Bit
LVT	Low-Threshold Voltage
MSB	Most Significant Bit
NB	Narrowband
PAN	Personal Area Network
RPMA	Random Phase Multiple Access
SDR	Software Defined Radio
SNR	Signal to Noise ratio
SS	Spread Spectrum
SVT	Standard-Threshold Voltage
TLA	Three-Letter Abbreviation
UNB	Ultra-Narrowband
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
WSN	Wireless Sensor Network

Chapter 1

Introduction

After the internet and mobile communications, the Internet of Things (IoT) is considered to be the third wave of information technology [12]. It involves large numbers of devices connected in a wireless sensor network (WSN) able to communicate with each other. These devices have specific requirements such as long range communication, low production costs and long battery life. Generally this means that these applications try to consume as little power as possible while maintaining sufficient performance.

An example application of a WSN can be the measurement and monitoring of appliances and their surroundings, such as the air quality in a building or the features of farmland soil. Many IoT services require a very low data rate in the order of tens of bytes, sent only several times a day [13].

For these applications, Low-Power Wide Area Network (LPWAN) is the best suited technology since it can achieve low power communication over a long range if low datarate is sufficient. Other technologies such as Cellular (e.g. 3G and 4G) offer middle to long range communication but also consume more power as they facilitate much greater data rates. Personal Area Network (PAN) technologies such as Bluetooth/BLE are very power efficient but only work over short distances [10].

Within the field of LPWAN three main techniques exist, Narrowband (NB), Ultra-Narrowband (UNB) and Spread Spectrum (SS). Several companies have already developed and patented technologies based on these techniques such as NB-IoT which is NB-based, Sigfox and Telensa which are UNB-based, and LoRa and RPMA which are SS-based [11]. The main difference between the UNB and SS techniques is the bandwidth, which is very narrow for UNB and much wider for SS. Because of this narrow transmission channel, UNB offers very low in-band noise. A narrow filter in the receiver will filter out most of the noise [1]. However, this also means UNB communication is highly susceptible to frequency offset. A small shift in the signal frequency may shift the signal outside of the filter band causing it to be blocked at the receiver. Such an offset can be caused for instance by a mismatch between the oscillators used in the transmitter and receiver, or by a Doppler shift caused by relative movement between the two.

To overcome the impact of the frequency offset, a frequency offset tolerant demodulation and detection scheme can be used called Double Differential Phase-Shift Keying (DDPSK). It encodes the input data into the double difference of the phase of the transmitted signal. If a double differential autocorrelation based demodulator is used in the receiver, it can extract the transmitted data even in the presence of a frequency offset. This method is invariant to changes in carrier frequency offset as shown in Chapter 2. A downside is that the filter at the receiver still needs to be wide enough for the shifted signal to fit within the filter band. This will inevitably allow more noise to pass through the filter as well.

To prevent deterioration of the Signal to Noise ratio (SNR) from degrading BER performance, a new type of demodulator was proposed in [16]. This demodulator allows for a wide receiver filter to be used while still maintaining a good SNR. It consists of multiple conventional demodulator paths and was first implemented by [14]. This implementation however was not heavily optimized for power consumption. As shown in [14] the main source of the power consumption in this implementation was leakage power. For convenience these values are presented in Table 1.1.

	Switch power (μW)	Int. power (μW)	Leak. power (μW)	Total power (μW)
dmd	0.058	0.662	69.159	69.880
fir	0.005	0.034	1.005	1.044
cic	0.117	2.639	0.161	2.917
bbc	-	-	-	-
Total	0.18	3.335	70.325	73.841

TABLE 1.1: Power consumption values of the implementation presented in [14]

In this thesis the DDPSK demodulator designed in [16] is implemented using VHDL and synthesized towards ASIC. The system components responsible for digital signal conditioning follow the design of [14]. To optimize power consumption, current thesis focuses on the implementation of the components with specifications set via system level design in [14].

1.1 Research questions

The research questions this thesis aims to answer are as follows:

How can the components of the receiver system be optimized for power?

The main trade-off that can be identified in the system is power versus performance. Performing less calculations and with less precision will require less hardware, but this will also increase the bit error rate (BER) of the system. Given the low data-rate of the system, leakage power consumption is dominant in the current implementation, especially in the demodulator component. This leads to the second research question.

How can the component's leakage power consumption be reduced?

Since the major contribution to power consumption comes from leakage, larger improvement can be made in reducing leakage power than dynamic power. In general this means reducing the amount of hardware which can be achieved in various ways. If this requires additional overhead control logic another trade-off arises. The complexity of control circuitry generally increases as more hardware is removed. Control logic that needs to be added should not outweigh the hardware it is able to reduce.

To what extend can the demodulator be power-gated, in a duty cycling fashion, in order to reduce the leakage power consumption?

In reducing leakage power, instead of removing hardware it can also be powered off for a while. In this state it will not be able to leak power, but also unable to perform operations. The main idea behind duty-cycled power-gating is turning a system component off while storing its inputs in memory. When the component is powered up it can then process the data from memory at a much faster rate, such that it can be powered off again as soon as possible when it is finished. The feasibility of this method will be investigated, as there is again a trade-off between the leakage power reduced and the additional power control logic and memory overhead.

1.2 Outline of the thesis

Some background information is given in Chapter 2 as a starting point for the thesis. The mathematics behind DDPSK modulation and demodulation is explained and power-saving techniques that are used in optimizing the design are presented. Additionally, Chapter 2 includes a brief review on related work.

Chapter 3 presents the datapath designs for the system which use the low power techniques to decrease leakage power. Choices made in designing separate system components are discussed. Furthermore, a mathematical model made to evaluate the effectiveness of duty cycling is presented.

The system designed in Chapter 3 is implemented in VHDL in Chapter 4. The working principles of the system are explained in this chapter and choices of word lengths are explained. The scalability of the implementation is also discussed here.

Performance of the system was measured through simulations and the results are shown in Chapter 5. The power consumption of the system as well as the bit error rate (BER) is given and compared to literature as well as the previous implementation of the system.

Finally, Chapter 6 concludes the thesis. In this chapter possible further optimizations are suggested as future research direction which may contribute to further reduction of power consumption.

Chapter 2

Background

This chapter presents background information for the rest of the thesis such as low power techniques and related work. It explains how double-differential phase-shift keying (DDPSK), the detection scheme used in the demodulator, works mathematically and presents an enhanced demodulation scheme for DDPSK. Finally it will present comparable demodulator implementations found in literature in the related work section.

2.1 DDPSK

The receiver uses DDPSK, which is a modulation technique that is invariant to phase and frequency offset. Because phase shift keying encodes information into the phase of a carrier signal, the amplitude of that signal does not contain any information. The mathematics used to modulate and demodulate the signal is presented here.

2.1.1 Modulation

In order to use DDPSK at the receiver's side, the transmitted information should be encoded into the double difference of the transmitted signal phase. Figure 2.1 shows a DDPSK modulator which is used for encoding information [17]. x_n, the input of the modulator, has the information encoded in its phase. u_n denotes the signal after the first modulation stage. The output of the demodulator, y_n, can be described by the following equations:

$$y_n = u_n \cdot y_{n-1} \tag{2.1}$$

$$u_n = x_n \cdot u_{n-1} \tag{2.2}$$

Given that these signals are complex they can be written in polar form, in which case the multiplication adds the phases of the signal. Let α_n, β_n and ϕ_n be the phases of x_n, u_n and y_n, respectively, then:

$$\beta_n = \alpha_n + \beta_{n-1} \tag{2.3}$$

$$\phi_n = \beta_n + \phi_{n-1} \tag{2.4}$$

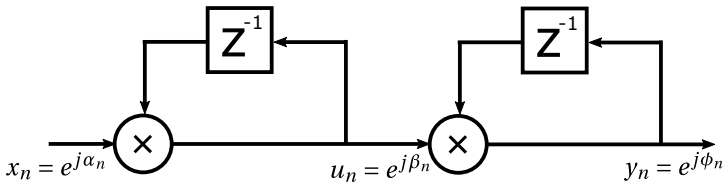

FIGURE 2.1: Block diagram of a double differential modulator

Which can be rewritten to:

$$\alpha_n = \beta_n - \beta_{n-1} \tag{2.5}$$

$$\beta_n = \phi_n - \phi_{n-1} = \Delta\phi_n \tag{2.6}$$

$$\alpha_n = \phi_n - 2\phi_{n-1} + \phi_{n-2} = \Delta\phi_n - \Delta\phi_{n-1} \tag{2.7}$$

And thus the input phase has been successfully encoded into the double difference of the output phase and the signal is ready to be transmitted.

2.1.2 Demodulation

Using a baseband equivalent model, the symbols coming into the demodulator at the receiver side can be described by the following equations:

n^{th} symbol:

$$s_n(t) = e^{j\phi_n} \cdot e^{j\omega_0 t} \tag{2.8}$$

$(n-1)^{th}$ symbol:

$$s_{n-1}(t) = e^{j\phi_{n-1}} \cdot e^{j\omega_0 (t-T)} \tag{2.9}$$

$(n-2)^{th}$ symbol:

$$s_{n-2}(t) = e^{j\phi_{n-2}} \cdot e^{j\omega_0 (t-2T)} \tag{2.10}$$

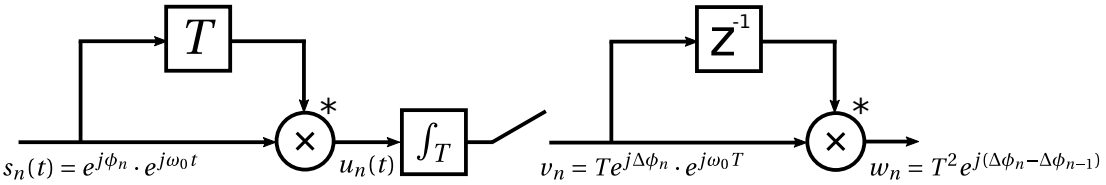

FIGURE 2.2: Block diagram of a double differential demodulator

Where T is the symbol period and $\omega_0 = 2\pi f_{offset}$ is the frequency offset. The output after the first multiplication (the first stage of autocorrelation) is as follows:

$$\begin{aligned} u_n(t) &= s_n(t) \cdot s_{n-1}^*(t) \\ &= e^{j\phi_n} \cdot e^{j\omega_0 t} \cdot e^{-j\phi_{n-1}} \cdot e^{-j\omega_0(t-T)} \\ &= e^{j(\phi_n-\phi_{n-1})} \cdot e^{j\omega_0 t} \cdot e^{-j\omega_0 t} \cdot e^{j\omega_0 T} \\ &= e^{j(\phi_n-\phi_{n-1})} \cdot e^{j\omega_0 T} \end{aligned} \tag{2.11}$$

$$\begin{aligned} u_{n-1}(t) &= s_{n-1}(t) \cdot s_{n-2}^*(t) \\ &= e^{j\phi_{n-1}} \cdot e^{j\omega_0(t-T)} \cdot e^{-j\phi_{n-2}} \cdot e^{-j\omega_0(t-2T)} \\ &= e^{j(\phi_{n-1}-\phi_{n-2})} \cdot e^{j\omega_0 2T} \cdot e^{-j\omega_0 T} \\ &= e^{j(\phi_{n-1}-\phi_{n-2})} \cdot e^{j\omega_0 T} \end{aligned} \tag{2.12}$$

In these expressions we can rename the terms $\phi_n - \phi_{n-1} = \Delta\phi_n$ and $\phi_{n-1} - \phi_{n-2} = \Delta\phi_{n-1}$ so that:

$$u_n(t) = e^{j\Delta\phi_n} \cdot e^{j\omega_0 T} \tag{2.13}$$

$$u_{n-1}(t) = e^{j\Delta\phi_{n-1}} \cdot e^{j\omega_0 T} \tag{2.14}$$

The output of the first stage, $u_n(t)$, is integrated over a symbol period using an integrate and dump block. This results in a sampled signal, v_n, at the output of the integrator. For simplicity it is assumed that the signal amplitude is constant over one symbol:

$$v_n = u_n(t)T = Te^{j\Delta\phi_n} \cdot e^{j\omega_0 T} \tag{2.15}$$

$$v_{n-1} = u_{n-1}(t)T = Te^{j\Delta\phi_{n-1}} \cdot e^{j\omega_0 T} \tag{2.16}$$

There is still a constant phase shift $e^{j\omega_0 T}$ which disappears after calculating the double differential:

$$\begin{aligned} w_n &= v_n \cdot v_{n-1}^* \\ &= Te^{j\Delta\phi_n} \cdot e^{j\omega_0 T} \cdot Te^{-j\Delta\phi_{n-1}} \cdot e^{-j\omega_0 T} \\ &= T^2 e^{j(\Delta\phi_n-\Delta\phi_{n-1})} \end{aligned} \tag{2.17}$$

Now, the phase of output w_n contains the double difference of the input phases $\Delta\phi_n - \Delta\phi_{n-1}$ and the constant phase shift $e^{j\omega_0 T}$ has been removed. The amplitude of the signal has increased by T^2, but its amplitude is not important since it contains no information.

The length of the delays T can be chosen, in this implementation the delay is two symbol periods (T) for the first demodulation stage and one symbol

period for the second one, as suggested by [17]. This provides better performance because it makes the double difference dependent on four samples as opposed to three, reducing noise correlation between the two demodulator stages [17].

The number of samples per symbol N can be increased to allow for a higher tolerable frequency offset. For this reason this thesis uses $N = 16$ and $N = 32$. According to the Nyquist Theorem the frequency offset that can be tolerated equals twice the sample rate. By increasing N the sample rate increases and a larger frequency offset can be tolerated. In practice however a frequency offset will already deteriorate performance, due to the filter shape and bandwidth of the signal.

A visual representation of this is shown in Figure 2.3. It shows how the filter in front of the demodulator does not have a perfectly rectangular frequency response. Instead it has a transition band which limits the highest tolerable frequency offset of the demodulator. If part of the signal falls in this transition band it will be blocked partially. This is a second effect that limits the tolerable frequency offset. The frequency offset shifts the signal's center frequency, but the signal has a bandwidth of B_s, with $\frac{1}{2}B_s$ on either side. This means that in order to prevent part of the signal overlapping with the transition band of the filter, the frequency offset has to be $\frac{1}{2}B_s$ lower than the start of the transition band.

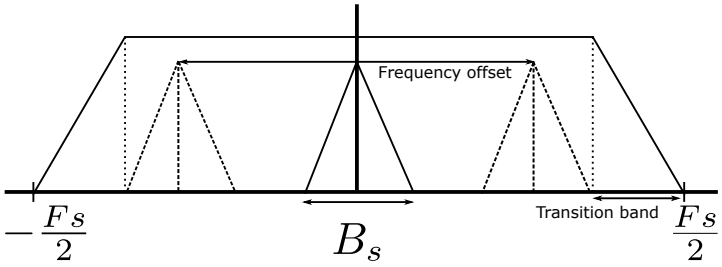

FIGURE 2.3: Visual representation of a frequency offset in the frequency spectrum

2.2 Previous work

2.2.1 The receiver system

Designing the complete receiver digital signal processing (DSP) chain at system level has been investigated in [14]. This system is shown in Figure 2.4.

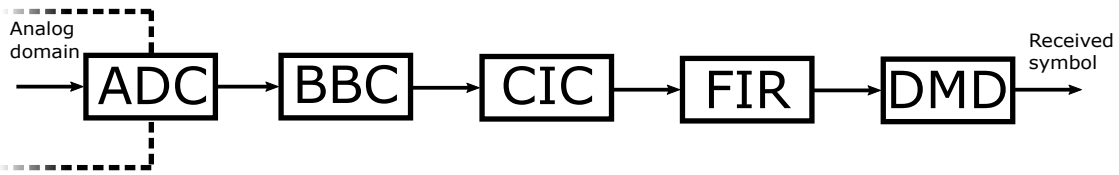

FIGURE 2.4: Block diagram of the full system

It shows the analog front end on the left side, where the analog to digital converter (ADC) converts the signal to a signal in the digital domain. It is followed by a several DSP components. First a baseband conversion block (BBC), which converts the signal from intermediate frequency to baseband. Then a cascaded integrator-comb filter (CIC) followed by a finite impulse response filter (FIR) condition the signal, in preparation for the demodulator, by heavily reducing the sample rate and filtering out-of-band noise. Finally, the conditioned signal is fed to the demodulator (DMD). This thesis looks solely at the digital domain of the receiver, the blocks after the ADC.

2.2.2 Enhanced autocorrelation demodulator

In a conventional autocorrelation demodulator, multiplying two symbols with each other is performed as follows. First, each sample of one symbol is multiplied with the respective sample of the previous symbol. The final result of a multiplication between two symbols (autocorrelation of the input signal) is then calculated as the sum of these sample products.

A method to improve the signal to noise behaviour of the autocorrelation demodulator is presented in [16]. Instead of multiplying each sample of a symbol only with the respective samples of the previous symbol, the samples are also multiplied by the left and right shifted samples of the previous symbol. The amount of shift can increase to up to $N-1$ where N is the number of samples per symbol.

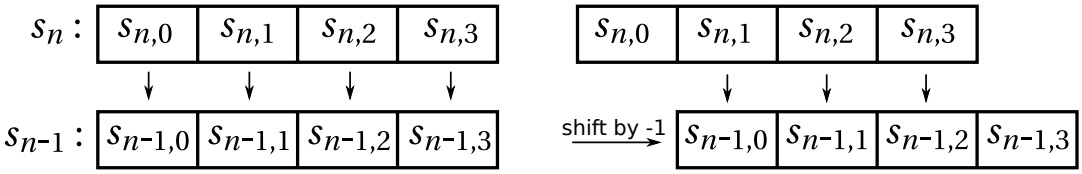

FIGURE 2.5: A representation of two symbols as they are multiplied sample by sample. No shift is applied on the left, a shift of -1 is applied (to the delayed symbol) on the right.

A visual representation of this principle is shown in Figure 2.5. The vertical arrows show which samples are multiplied and thus, in the scenario on the left side of Figure 2.5, each sample of a symbol S_n is multiplied by the respective sample of the delayed symbol S_{n-1}. In the scenario on the right, the delayed symbol is first right-shifted by one sample and then the samples are multiplied. The first sample of S_n and the last sample of S_{n-1} are not multiplied in this case and only three multiplications are performed. In this thesis a positive shift direction is assumed to be to the left. Therefore this right shift corresponds to a shift by -1. Note that the shift refers to the delayed symbol and not the current symbol.

Increasing the shift length in both directions will lead to different pairs of samples being multiplied, until each sample of S_n is multiplied by every sample of S_{n-1}. The magnitude of the shifts is denoted by p so that $p = 0$ corresponds to the conventional demodulation scheme where no shift is applied,

possitive p correspond to the right shifts and negative p to the left shifts. The inter-sample products obtained by a certain shift p are summed, resulting in a single value for each p. A second differential decoder is then applied to this value producing a double differential demodulator for each p.

Therefore, there is a path through the demodulator for each shift amount p, starting with a differential decoder, followed by a summation block, and finally another differential decoder. The values at the output of all the paths are added to obtain the final output of the demodulator. Although this demodulation scheme is more complex than the conventional scheme, it offers a better BER performance.

2.3 Low power techniques

In general, a lower clock frequency leads to lower dynamic power consumption, since the capacitances inside the logic cells are being switched less often. On the other hand, more hardware leads to higher power consumption because of leakage power. Thus lowering the clock frequency while not significantly increasing the amount of hardware is a way to reduce dynamic power consumption. However, when the number of calculations stays the same, the clock frequency cannot simply be reduced without taking longer to perform all necessary calculations. Finding a way to avoid calculations can circumvent this.

When a decimator is present after a digital signal processing block, the decimator can sometimes be moved in front of the DSP block while keeping the system functionally equivalent, depending on the type of DSP block. Decimating the input before processing it reduces the amount of required computation. The DSP block can then operate at a lower clock frequency. A decimator can always be moved to the prior of an adder or multiplier without changing the system functionality [6].

For delay blocks, however, the amount of delay has to be adjusted to ensure correct performance of the system. The following transformation applies as long as the delay length, N, is a multiple of decimation factor, M [18]:

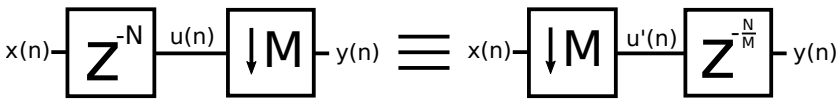

FIGURE 2.6: Moving a decimator with factor M in front of a delay of N

The system on the left hand side can be mathematically described as follows:

$$y_n = u_{Mn} \tag{2.18}$$

$$u_n = x_{n-N} \tag{2.19}$$

$$y_n = x_{Mn-N} \tag{2.20}$$

And for the right hand side:

$$y_n = u'_{n-\frac{N}{M}} \tag{2.21}$$

$$u'_n = x_{Mn} \tag{2.22}$$

$$y_n = x_{M(n-\frac{N}{M})} \tag{2.23}$$

It is clear both sides are identical since:

$$x_{M(n-\frac{N}{M})} = x_{Mn-N} \tag{2.24}$$

Moving the decimator through a delay block is only possible when the ratio $\frac{N}{M}$ is an integer number, and thus the delays in the DSP block should be a multiple of the decimation ratio. This may not always be the case, but in case of an FIR filter, transformations can be applied to enforce this. One way to achieve this is through the use of polyphase decomposition.

The multiplier coefficients in an FIR filter can be divided into a number of groups based on which inputs they are multiplied with. The simplest division would be into two groups, one containing those coefficients that are multiplied by the inputs $\{x_0, x_2, x_4, ...\}$, which one could call the 'even' group, and another group with those coefficients that are multiplied by the inputs $\{x_1, x_3, x_5, ...\}$, then called the 'odd' group.

Note that the inputs in the even group are all delayed by a delay of $2n$ and the delays in the odd group by a delay of $2n + 1$, with $n \in \{0, 1, 2, 3, ...\}$. Therefore the two groups can be implemented as two separate FIR filters with delays of order 2, where the input to the odd part of the filter has one additional delay applied to it. The outputs of these two filters are then added to yield the same output as the original filter. Apart from the initial delay, all delays in this new filter structure are of order two and can therefore have the afore mentioned transformation applied to them with a decimation of factor two. For higher decimation factors the coefficients can be split in a larger number of groups to achieve higher order delays in these groups, so that the decimator may then be moved to the front.

2.3.1 Complex multiplication

Generally a complex conjugate multiplication requires four multiplications and two additions/subtractions, given that $z = (a + bi) \cdot (c - di) = (ac + bd) + (bc - ad)i$. One multiplication can be omitted at the cost of three additions, by rewriting the calculation in the following way [8]:

Let

$$k_1 = a(c - d) = ac - ad \tag{2.25}$$

$$k_2 = c(b - a) = bc - ac \tag{2.26}$$

$$k_3 = d(a + b) = ad + bd \tag{2.27}$$

Then, the magnitudes of the real and imaginary parts of the multiplication result can be written as:

$$Re\{z\} = k_1 + k_3 = ac - ad + ad + bd = ac + bd \tag{2.28}$$

$$Im\{z\} = k_1 + k_2 = ac - ad + bc - ac = bc - ad \tag{2.29}$$

Calculation of the k values and from them deriving the real and imaginary part of the multiplication result requires three multiplications and five additions/subtractions. Since the power consumption of multipliers is much higher than adders [7], implementing the calculation in the way described here is expected to consume less power.

2.3.2 Transistor threshold voltage

There are three kinds of cell libraries with respect to transistor threshold voltage that are used in ASIC design. These are denoted as high-threshold voltage (HVT), standard-threshold voltage (SVT) and low-threshold voltage (LVT) [4].

HTV libraries offer the lowest leakage power consumption of the three but are also the slowest. They are not suitable for high speed designs where timing is critical. LTV libraries on the other hand utilise transistors that waste a lot more power as leakage, but are also much faster and well suited for high speed designs. SVT libraries are in the middle with their average speed and leakage power consumption.

Depending on the design, a more efficient library can be used to reduce power consumption. The system presented in this thesis has a very low clock speed. Timing is not critical and so an HVT library is best suited. This is expected to significantly reduce the leakage power consumption.

2.4 Related work

Several implementations of narrowband receivers for IoT and deep space applications can be found in literature, which are briefly presented here to give the reader an idea of their power characteristics.

A low-power multirate DPSK receiver for space applications was designed and implemented in [19], capable of also decoding DDPSK if necessary, using multiplexers to possibly bypass the second demodulator. It is implemented on $0.35\mu m$ CMOS and its digital part consumes $90\mu W$.

A DBPSK/GFSK tranceiver presented in [9] consumes $14.5mW$ for both the digital and analog parts, processed in $65nm$ CMOS. They target sigfox, which is an application of ultranarrowband, and use a combination of differential detection and constant phase offset estimation to counter frequency offset. It allows them to tolerate around ± 70 Hz of frequency offset. In [5] a low-power FSK receiver was designed and implemented (digital section only) on

$0.25\mu m$ CMOS. Its power consumption is below $100\mu W$ at datarates below 20kb/s.

A software defined radio (SDR) is implemented in [2], which implements a multi-standard digital baseband processor for IoT on a RISC-V core. It is estimated to consume $1.6mW$, when running at 114MHz and implemented on 28nm FDSOI. Another SDR processor was presented in [3] consuming $1.37mW$ when implemented in 28nm technology, realising a BLE baseband processor.

Table 2.1 summarises these implementations found in literature.

Source	**Modulation**	**Power**	**Frequency**	**Technology**
[19]	(D)DPSK	90 µW	4Mhz	SOI CMOS 0.35 µm
[5]	FSK	100 µW	1.2Mhz	CMOS 0.25 µm
[9]	DBPSK/GFSK	14.5 mW	885Mhz	CMOS 65 nm
[2]	BLE GFSK 1-Ms/s (SDR)	1.6 mW	114Mhz	FDSOI 28 nm
[3]	BLE (SDR)	1.37 mW	20Mhz	CMOS 28 nm

TABLE 2.1: Comparison of several narrowband transceivers and receivers found in literature

In this thesis only part of the baseband processing is implemented, and in the works referenced in Table 2.1 the fraction of power consumed by each digital processing task is not specified. Therefore these numbers cannot directly be compared with the power consumption of the system implemented in this thesis, and only a comparison with [14] will be done in Chapter 5.

Chapter 3

Design

This chapter will focus on the steps taken in designing the system components and explain decisions made in their design. As stated in chapter 2, motivation for the choice of system level components is given in [14]. This previous system is used in this thesis where the focus is on a power efficient implementation.

The full system is shown in Figure 3.1. The left input comes from the analog-to-digital converter at a 4Mhz samplerate and is first processed by the baseband conversion block. The converted signal is then processed by a CIC and an FIR filter. These blocks make up the signal conditioning part of the receiver. The conditioned signal is then input into the actual demodulator which outputs the transmitted symbols. The datarate of the system is designed at 100 bits per second, so symbols are output at a rate of 100Hz. Each DSP block is treated in this chapter.

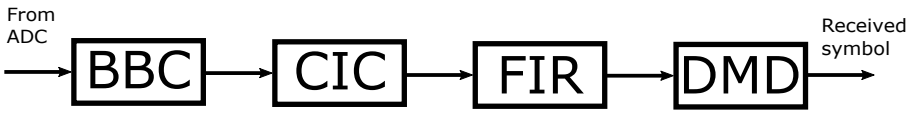

FIGURE 3.1: Block diagram of the full system

3.1 Baseband Conversion

The baseband conversion component down-converts the incoming signal from IF frequency to baseband by multiplying the input with the IF carrier. It is implemented in the same way as in [14] and performs the following operation, where $s(n)e^{j\omega_{IF}nT_s}$ is the input of the baseband conversion block, $s(n)$ the information carrying signal and $T_s = \frac{2\pi}{\omega_s}$ is the sampling period:

$$s(n)e^{j\omega_{IF}nT_s} \cdot e^{-j\omega_{IF}nT_s} = s(n) \tag{3.1}$$

Because calculations are performed digitally, sampling will take place. The expression $e^{-j\omega_{IF}nT_s}$ will therefore have fixed values and does not need to be calculated, depending on the sampling ratio $r_s = \frac{\omega_s}{\omega_{IF}}$. By choosing a sampling ratio of four [14], these values will all be of unity magnitude as shown here.

$$r_s = \frac{\omega_s}{\omega_{IF}} = 4 \implies \omega_s = 4\omega_{IF} \tag{3.2}$$

$$e^{-j\omega_{IF}nT_s} = e^{-j\omega_{IF}n\frac{2\pi}{\omega_s}} = e^{-j\omega_{IF}n\frac{2\pi}{4\omega_{IF}}} = e^{-jn\frac{\pi}{2}} \tag{3.3}$$

When n is equal to $0, 1, 2, 3$ plus $4k$ with $k \in \mathbb{Z}$, the term $e^{-jn\frac{\pi}{2}}$ respectively produces a $1, -j, -1, j$. This removes a lot of complexity because multiplying by these numbers is trivial, which makes the implementation as in [14] possible.

A 2-bit counter will be used to cycle through the 'multiplications'. Since the multiplier value is always of magnitude one, the inputs are simply rerouted to certain outputs as shown in tabel 3.1 and no actual multiplications have to take place.

n	Multiplier	Re out	Im out
$0+4k$	1	Re in	Im in
$1+4k$	$-j$	Im in	-Re in
$2+4k$	-1	-Re in	-Im in
$3+4k$	j	-Im in	Re in

TABLE 3.1: Values the complex input is multiplied with for different $n, k \in \mathbb{Z}$.

3.2 CIC-filter

The CIC filter was implemented similar to [14], as a single integrator stage followed by a single comb stage with a decimator in between them. It has a decimation factor of 1250, to reduce the 4Mhz input samplerate to 3200Hz. This 3200Hz is reduced to 1600Hz by the FIR filter, and given the symbol rate of 100Hz, this leads to a samplerate of 16 samples per symbol. This metric is referred ot as N. By choosing the decimation factor of 1250 differently, the number of samples per symbol N can be changed, of which the implications are discussed in later chapters.

The CIC filter effectively functions as a moving-average filter by continuously adding the input samples and then subtracting the oldest values from the total. The CIC filter implementation was not improved upon given that it consists of only a single integrator comb stage.

A block diagram of the CIC filter is shown in figure 3.2. The decimation factor is achieved by running the comb section on the right hand side at a lower clock rate that the integrator part on the left hand side, the ratio between the two clocks being the decimation factor. This is one of the benefits of a CIC filter; its decimation ratio can be changed simply by altering the relative clock rates.

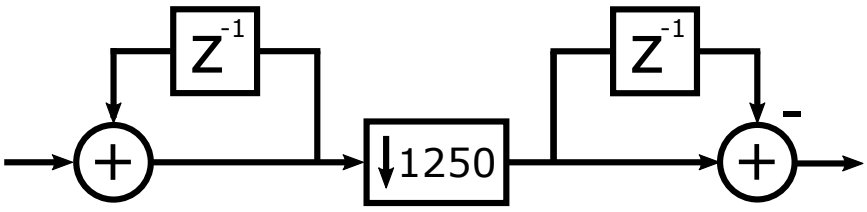

FIGURE 3.2: Block diagram of the CIC filter

3.3 Polyphase FIR-filter

An FIR-filter contains many multiplications with constant coefficients. These multiplications can be converted to shifts and additions in order to prevent using actual multipliers, which consume much more power than adders do. This can be achieved by converting the multiplication by a coefficient to a sum of multiplications by powers of two. For example, a multiplication by ten can be written as the sum of a multiplication by eight and by two; $10x = 2x + 8x$. Since $8 = 2^3$ and $2 = 2^1$, both powers of two, the multiplications only require bit-shifts to be performed, no multipliers are required this way. This principle works for any multiplication, not just a multiplication by ten.

In this specific FIR filter [14], the integer coefficients are values very close to powers of two: {4636, -1137, -7875, 8226, 32767, 32767, 8226, -7875, -1137, 4636}, corresponding with the values of $\{b_(0), b(1) \cdots b(9)\}$ in Figure 3.4. This allows each of the multipliers in the FIR to be replaced by a single bitshift without significantly altering the behavior of the filter. The normalised frequency responses of the filter using the original coefficients from [14] and for the closest power of two coefficients is shown in Figure 3.3. Those powers of two are $\{2^{12}, -2^{10}, -2^{13}, 2^{13}, 2^{15}, 2^{15}, 2^{13}, -2^{13}, -2^{10}, 2^{12}\}$. These coefficients all contain a factor of 2^{10} which can be taken out.

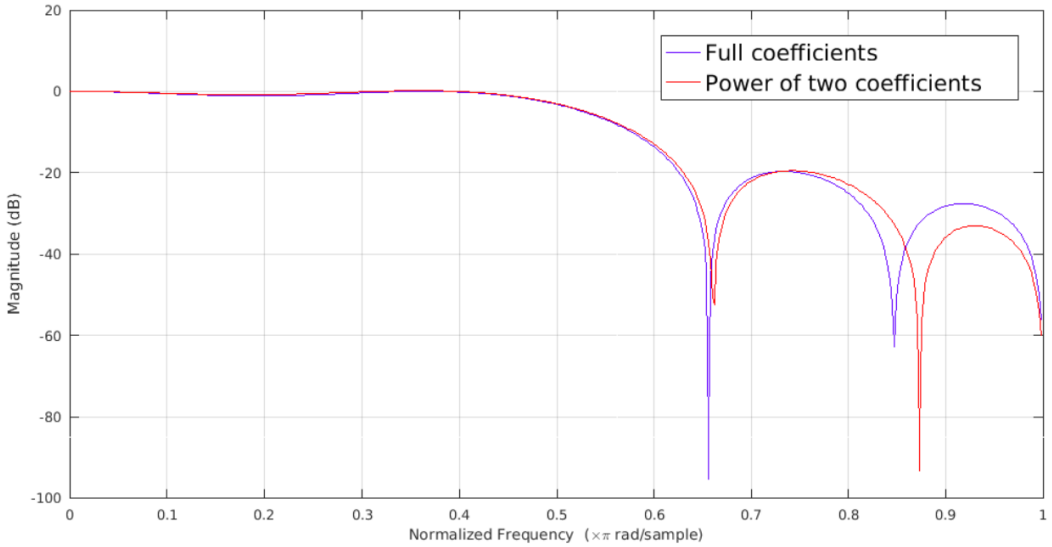

FIGURE 3.3: Comparison of the FIR-filter frequency response using the full and power of two coefficients

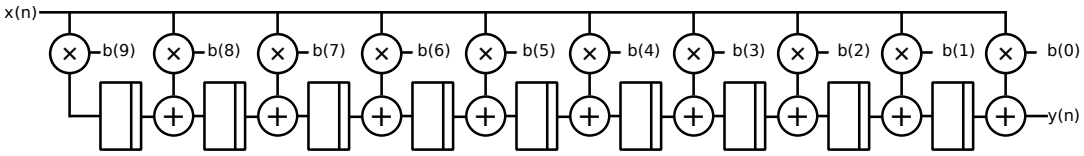

FIGURE 3.4: Dataflow graph of the FIR filter

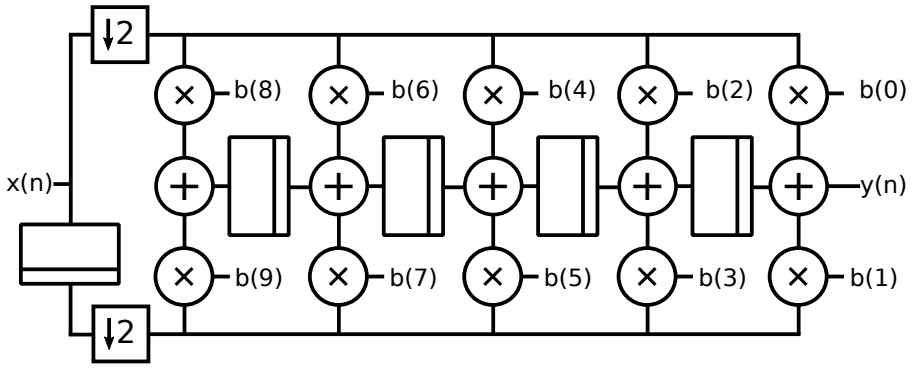

FIGURE 3.5: Dataflow graph of the FIR filter with polyphase decomposition applied to it

After the FIR filter the sample rate is reduced by a factor 2 using a decimator. This allows polyphase decomposition to be applied to the FIR filter. This reduces the clockspeed at which the FIR filter operates at by a factor two. The filter coefficients are split into two groups, equal to the decimation factor, and polyphase decomposition is applied as explained in Chapter 3. The decimator can then be moved in front of the additions and multiplications and before the filter. Instead of calculating all the samples and decimating half of them, only the samples that are not removed by decimation are calculated.

Figure 3.4 shows the filter before polyphase decomposition, decimation happens after the filter as a separate DSP operation. After polyphase decomposition, the filter has the structure shown in Figure 3.5, where decimation is done before filtering. Even though a register was added to delay one of the inputs by one input cycle, overall less registers are required. Even though the polyphase filter structure contains the same number of hardware arithmetic units, it can run at half the speed, proportional to the decimation factor. This structure will therefore consume less power.

3.4 Demodulator

The first stage of the demodulator when $N = 4$ (samples per symbol) can be described by the data flow graph shown in Figure 3.7. The demodulator effectively multiplies each sample of the input symbol a_n with every sample of the double delayed symbol a_{n-2}. Note that the samples are complex valued and the multiplier is a complex conjugate multiplier, so it has been marked with a *. These symbol by symbol multiplications are illustrated in Figure 3.6 and explained in more detail in Chapter 2. The products of the symbol by symbol multiplications are added together as explained in [16]; all the

products that correspond to a certain amount of shift as shown in Figure 3.6 will be summed. This can be easily realised using a grid of multipliers and adders, resulting in the data flow graph in Figure 3.7.

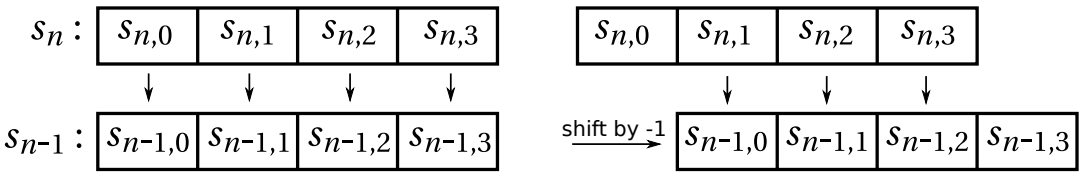

FIGURE 3.6: A representation of two symbols as they are multiplied sample by sample. No shift is applied on the left, a shift of -1 is applied (to the delayed symbol) on the right.

Due to its repetitive nature it can be folded into two directions. 'Folding' in this case refers to collapsing multiple identical operating units or dataflow structures in the dataflow graph onto one another, down to a single unit or structure. For example, the four multipliers on the left hand side of Figure 3.7 are functionally identical. The operations they perform could therefore be performed by a single one of such multiplier. In a way, it is as if the paper of the page these multipliers are printed on could be folded across the grey dotted lines, such that the multipliers end up physically on top of each other, hence the term 'folding'. Of the four multipliers only one remains, and it will perform the operations of the original four multipliers.

When folding vertically, folding the incoming sample inputs over one another, a structure is created that takes a single input sample every cycle. The dataflow graph that is obtained when folding in this way is shown in Figure 3.8. This is a convenient way to fold since the FIR filter in front of the demodulator produces one sample during the same clock cycle. The first stage of the demodulator can then run at the same clock rate as the FIR filter output and process samples as they come out of the FIR filter.

The folded structure reduces the number of used adders and multipliers by a factor of N, with N being the number of samples per symbol. A single value is output in every cycle except for the last. In Figure 3.8, the output of the first cycle, $i = 0$, is denoted by $p = 3$. This is the same output corresponding to $p = 3$ in Figure 3.7 as well. During the last cycle, $N - 1$ additional outputs are produced, so during this cycle a total of N outputs are produced at once. In Figure 3.7 and 3.8 these are the outputs corresponding to $i = N - 1 = 3$. A summary of what happens in which cycle is shown in Table 3.2. Production of $2N - 1$ outputs over N cycles means the average number of outputs per cycle is approximately two.

A dataflow graph describing the second stage of the demodulator can be seen in Figure 3.9. It also contains a lot of repeated hardware which can be reduced through folding, resulting in the dataflow graph shown in Figure 3.10. This is achieved by grouping all the multiplications into a single multiplier unit for which the inputs are selected by multiplexers. Those cycle through the different input pairs depending on the state, which could be a simple counter

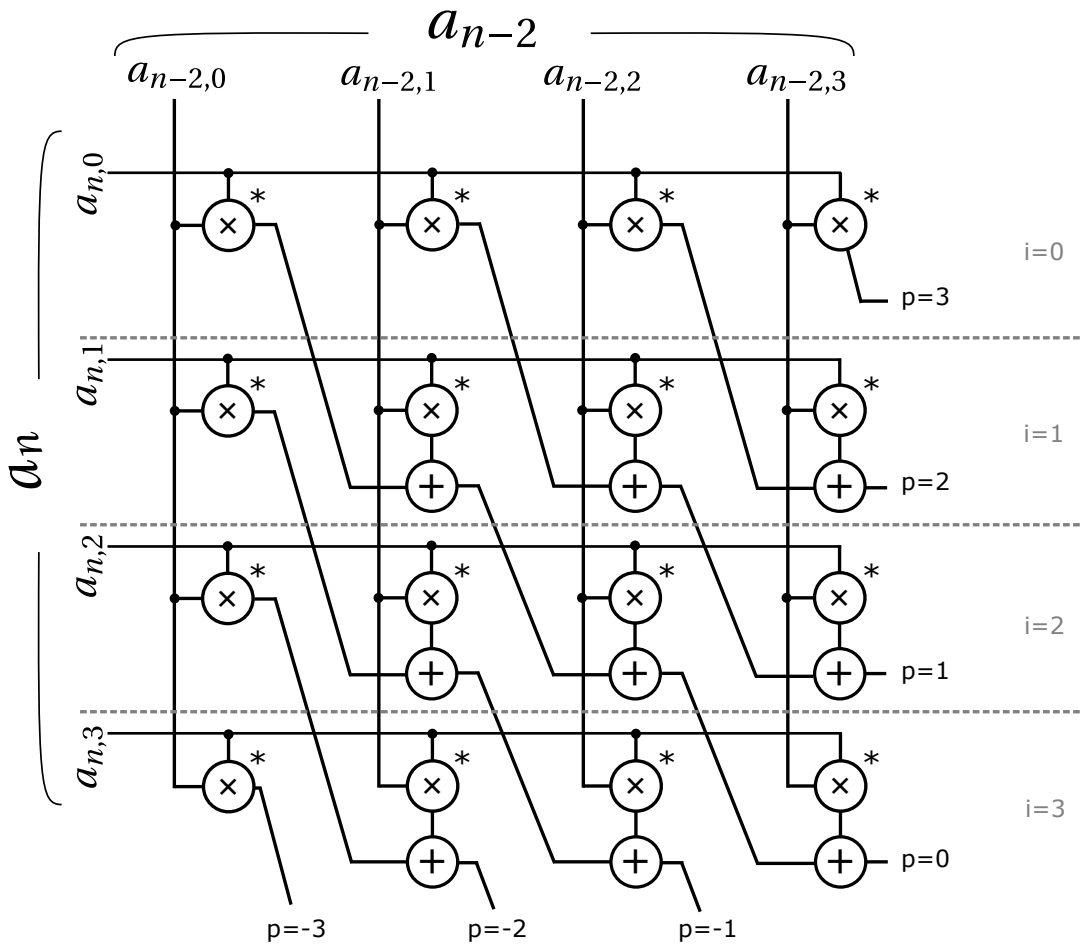

FIGURE 3.7: Dataflow graph of the first demodulator stage for $N = 4$.

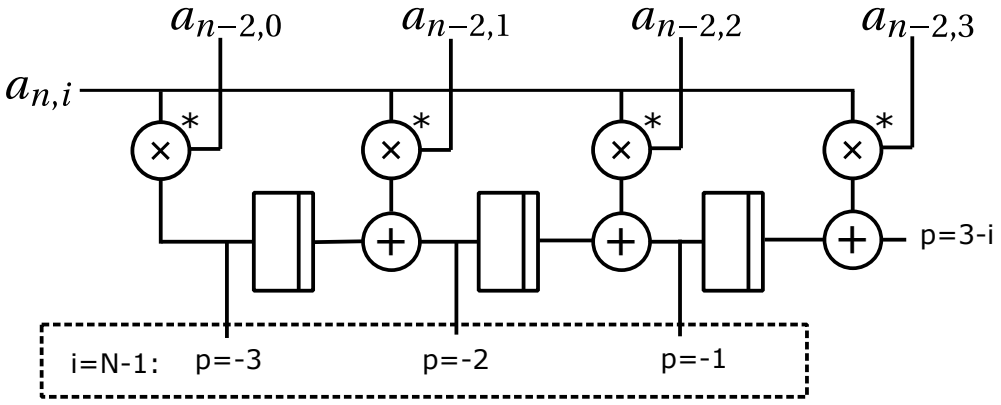

FIGURE 3.8: Folded dataflow graph of the first demodulator stage for $N = 4$.

Cycle	0	1	...	$N-2$	$N-1$
Input	$a_{n,0}$	$a_{n,1}$	...	$a_{n,N-2}$	$a_{n,N-1}$
Registers	Contain zeros	-	...	-	Reset
# of Outputs	1	1	...	1	N

TABLE 3.2: Compact timing diagram showing the input and number of outputs for every cycle. The registers are reset during the final cycle and as a result contain zeros in the first cycle.

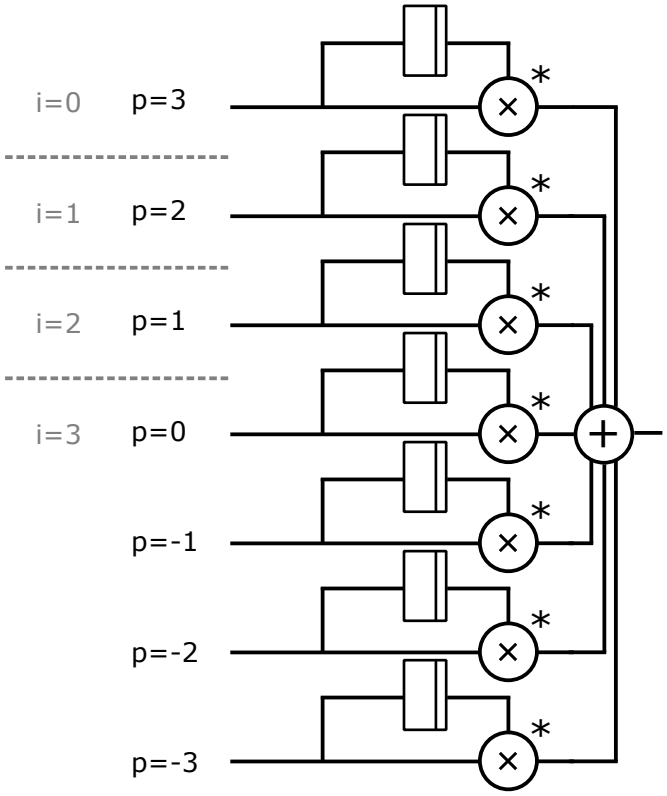

FIGURE 3.9: Dataflow graph of the second demodulator stage for $N = 4$.

running from 0 through $2N-1$. Addition of the multiplier outputs is then performed using an accumulator unit that adds one multiplier output as it is calculated each cycle and resets during the last cycle. This will effectively add up all outputs generated by the multiplier.

In this way the number of multipliers and adders is reduced from $2N-1$ down to just a single multiplier and adder. The clock speed of the folded datapath will have to be increased by a factor of $2N$ compared to when it was unfolded in order to perform all the operations in time, causing the second stage of the demodulator to operate at twice the clock speed of the first stage when both stages are folded. This makes sense because the first demodulator stage on average produces two outputs per cycle while the second stage accepts a single input per cycle. Thus, when the second stage runs twice as fast as the first stage, their input and output rate will match.

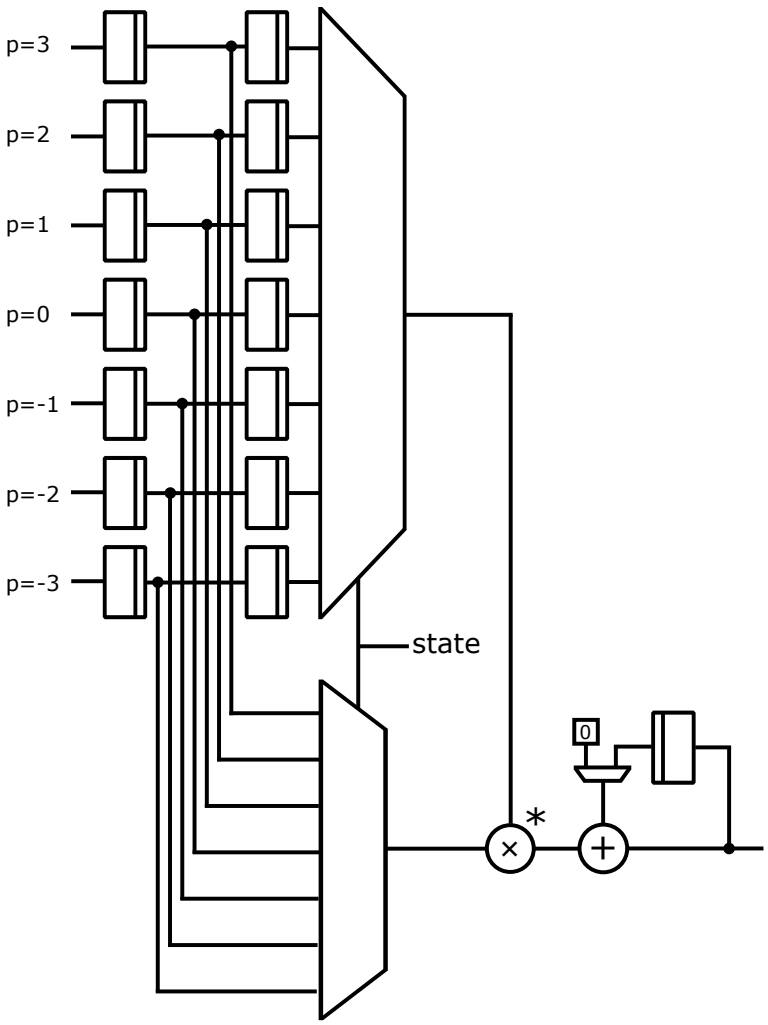

FIGURE 3.10: Folded dataflow graph of the second demodulator stage for $N = 4$.

The values on the input paths $p = -3$ through $p = 3$ of Figure 3.9 are stored in registers to account for the mismatch in data production and consumption between the first and second stage of the demodulator. This mismatch stems from the fact that the first stage does not produce $2N - 1$ outputs evenly spread across the cycles, but rather it produces N outputs during the last cycle, as displayed in the bottom row of Table 3.2. The registers allow the second stage to keep up with the first stage and give it time to process the samples one by one.

3.5 Duty-cycling the demodulator

Hardware can only leak power when it is powered, thus disabling a part of the hardware could help reduce this leakage. Since the hardware will still receive inputs while in the off-state, these inputs will have to be stored in some form of memory. After being powered on, the hardware can then run at a faster rate to process all those inputs, before powering off again. In this way the power to part of the system is being duty-cycled.

Figure 3.11 graphically shows how duty-cycling would work. Figure 3.11a shows normal operation where all the hardware runs all the time. There is some dynamic power consumption consisting of switching and leakage power as well as continuous leakage power. Here, T_{cycle} refers to the duration of one duty-cycle when it would be applied.

Then, when duty cycling is applied, a timing diagram like Figure 3.11b is obtained. Within the cycle time T_{cycle}, for some time T_{off} the hardware will be turned off and consume no power, after which it is turned on for some time $T_{process}$ to do all the work. Since the hardware is going to be turned off again at the end of the T_{cycle}, processing has to occur at a faster rate. In fact the processing speed has to increase by a factor of $\frac{T_{cycle}}{T_{process}}$ to make up for the off-time. Therefore the dynamic power consumption during on-time is expected to increase by the same factor, however the leakage power in this interval will not. Overall, the leakage power will therefore be decreased.

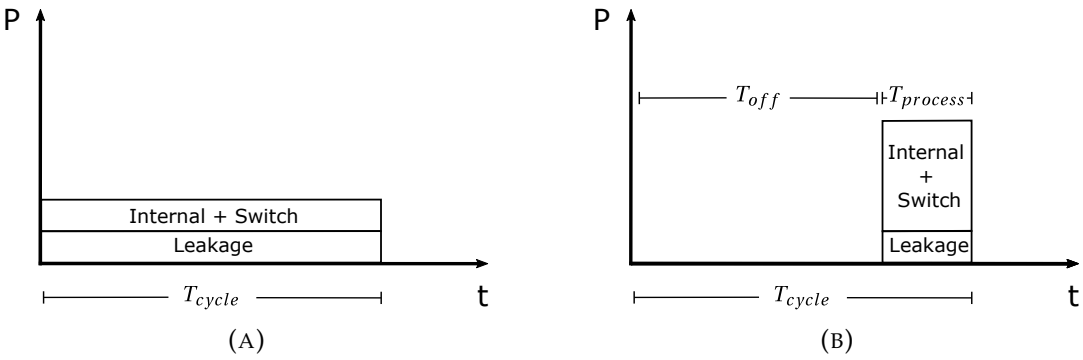

FIGURE 3.11: Timing diagram showing normal operation (left) and duty-cycling (right) of the demodulator.

FIGURE 3.12: A possible location for the memory block

In order to evaluate the viability of using duty-cycling for leakage power reduction, a model was developed. The model is used to compare locations in the system where the memory block can be positioned, as well as different periods and cycle times for the duty cycle. An example position of the memory block is shown in Figure 3.12 where it is placed between the FIR-filter and the demodulator.

Name	**Symbol**
Sample rate*	s_r
Wordlength*	W_L
Cycle time	T_{cycle}
Processing speed	p
Leakage Power*	P_{leak}
Dynamic Power*	P_{dyn}
Memory dynamic energy per bit	$E_{Mb,dyn}$
Memory leakage power per bit	$P_{Mb,leak}$
Samples stored*	s_{stored}
Bits stored*	b_{stored}
Processing time*	$T_{process}$
System off time*	T_{off}
Leakage energy saved*	E_{leak}
Memory total energy consumption*	$E_{M,tot}$

TABLE 3.3: List of symbols used in the model. Parameters dependend on the location of the memory are marked with *.

The following parameters were defined in the model:

Sample rate s_r, the sample rate at which the input samples are being stored in memory. This depends on the location of the memory block in the processing chain, as sample rates differ between components.

Wordlength W_L, the word length of the input samples. This also depends on the location of the memory block in the system.

Cycle time T_{cycle}, the cycle time denotes the duration of one store and process cycle.

Processing speed p, denotes the speed in samples/second at which processing of the stored samples occurs. It is the sample rate at which they are read from memory.

Leakage Power P_{leak}, consumed by the system components that are going to be power-gated. This leakage power will be prevented during shut down time as shown in Figure 3.11.

Dynamic Power P_{dyn}, switching and internal power consumption of the system components that are going to be power-gated, before power-gating is applied. The dynamic power for these components will increase in the power-gated system as shown in Figure 3.11.

Memory dynamic energy per bit $E_{Mb,dyn}$, the energy consumed by the memory to read and write a single bit, excluding leakage.

Memory leakage power per bit $P_{Mb,leak}$, the leakage power consumed by the memory per bit of storage in the memory.

A few of these parameters are dependent on the location of the memory block in the processing chain, namely the sample rate s_r, word length W_L and power consumptions P_{dyn} and P_{leak}. Because the value of these parameters depends on the location of the memory, they have been marked with a * in Table 3.3. As an example, the sample rate at the input of the FIR-filter is twice the sample rate at the output and the word lengths are different as well. If a memory block is placed at the FIR input a different number of samples, and therefore bits, would have to be stored over a certain time interval compared to placing it at the output. The symbols were defined without a fixed memory location in mind and will take on a value when evaluating the final expression formulated at the end of this Chapter (3.11), depending on the memory location.

The dynamic and leakage power parameters depends on the location of the memory block as well since those denote the power consumptions of components behind the memory block. A larger portion of the system can be turned off when the memory block is placed closer to the start of the chain and so the possible power saving would then be higher.

Using the parameters so far presented with the following equations, several more can be calculated.

Samples stored s_{stored}, the amount of samples stored in the memory block during a cycle, can be calculated from the sample rate s_r and cycle time T_{cycle}:

$$s_{stored} = s_r \cdot T_{cycle} \tag{3.4}$$

Bits stored b_{stored}. Each complex sample consists of a real and imaginary part, both W_L bits long. Thus the amount of bits stored in the memory block during one cycle can be calculated as:

$$b_{stored} = s_{stored} \cdot W_L \cdot 2 \tag{3.5}$$

Processing time $T_{process}$, is defined as the time the power-gated hardware is on and processing inputs during each cycle, and can be calculated from the amount of samples stored s_{stored} and the processing speed p:

$$T_{process} = \frac{s_{stored}}{p} \tag{3.6}$$

System off time T_{off}. When processing time $T_{process}$ and cycle time T_{cycle} are known, the time during each cycle for which the power-gated hardware is off can be calculated as:

$$T_{off} = T_{cycle} - T_{process} \tag{3.7}$$

Leakage energy saved E_{leak}. When the system is turned off for time T_{off}, the amount of leakage energy that is saved during each cycle by turning the system off is calculated as:

$$E_{leak} = P_{leak} \cdot T_{off} \tag{3.8}$$

Memory total energy consumption $E_{M,tot}$, the sum of the memory's dynamic and leakage power consumption, can be calculated as:

$$E_{M,tot} = P_{Mb,leak} \cdot T_{cycle} \cdot b_{stored} + E_{Mb,dyn} \cdot b_{stored} \tag{3.9}$$

As illustrated in figure 3.11b the power consumption is going to be concentrated on a small time interval $T_{process}$ and therefore the instantaneous power consumption is increased during processing. It is assumed that the same amount of energy that would be consumed as dynamic power during a single cycle is instead consumed during the processing time and therefore results in a higher power value. The instantaneous power consumption can be an important factor to take into account for practical purposes and is calculated as:

$$P_{inst} = (T_{cycle} \cdot P_{dyn}) / T_{process} + P_{leak} \tag{3.10}$$

Depending on the cycle time, T_{cycle}, and sample rate, s_r, a certain number of samples is stored in the memory block. A longer cycle time and a higher sample rate will lead to larger required memory as more samples will have to be stored. Then, the processing speed p dictates how fast the memory block is emptied again. Subtracting processing time from cycle time yields the time T_{off} during which the power-gated part of the system can be turned off. The leakage energy E_{leak} that is prevented during off-time is the potential energy that can be saved using duty-cycling. It should be noted that the memory block itself cannot be power gated and thus continuously consumes power as it stores samples.

The power per bit consumed by the memory for both the high and low speed variant are used to estimate the dynamic energy consumption and leakage

power per bit of memory, and from that the energy consumption of the memory during a single duty-cycle. By subtracting this energy from the energy saved through power gating, the energy saving that can potentially be gained is calculated.

Shortening the cycle time decreases the amount of data that is stored in the memory block and thus the energy consumed by the memory, at the cost of powering the system on and off more often. Note that the power cost of turning the system off and on has not yet been incorporated into the model. A break-even time can be defined as the cycle time for which the decrease in energy consumption through power gating equals the energy consumed by the memory block:

$$T_{cycle} = \frac{P_{leak} \cdot (1 - \frac{s_r}{p})}{P_{Mb,leak} \cdot 2W_L \cdot s_r} - \frac{E_{Mb,dyn}}{P_{Mb,leak}} \tag{3.11}$$

This equation is derived as follows:

$$E_{leak} = E_{M,tot} \tag{3.12}$$

$$P_{leak} \cdot T_{off} = P_{Mb,leak} \cdot T_{cycle} \cdot b_{stored} + E_{Mb,dyn} \cdot b_{stored} \tag{3.13}$$

$$P_{leak} \cdot (T_{cycle} - T_{process}) = (P_{Mb,leak} \cdot T_{cycle} + E_{Mb,dyn}) \cdot b_{stored} \tag{3.14}$$

$$P_{leak} \cdot T_{cycle} - P_{leak} \cdot T_{process} = (P_{Mb,leak} \cdot T_{cycle} + E_{Mb,dyn}) \cdot 2W_L \cdot s_{stored} \tag{3.15}$$

$$P_{leak} \cdot T_{cycle} - P_{leak} \cdot \frac{s_{stored}}{p} = (P_{Mb,leak} \cdot T_{cycle} + E_{Mb,dyn}) \cdot 2W_L \cdot s_r \cdot T_{cycle} \tag{3.16}$$

$$P_{leak} \cdot T_{cycle} - P_{leak} \cdot \frac{s_r \cdot T_{cycle}}{p} = (P_{Mb,leak} \cdot T_{cycle} + E_{Mb,dyn}) \cdot 2W_L \cdot s_r \cdot T_{cycle} \tag{3.17}$$

$$P_{leak} - P_{leak} \cdot \frac{s_r}{p} = (P_{Mb,leak} \cdot T_{cycle} + E_{Mb,dyn}) \cdot 2W_L \cdot s_r \tag{3.18}$$

$$P_{leak} \cdot (1 - \frac{s_r}{p}) - E_{Mb,dyn} \cdot 2W_L \cdot s_r = P_{Mb,leak} \cdot T_{cycle} \cdot 2W_L \cdot s_r \tag{3.19}$$

$$T_{cycle} = \frac{P_{leak} \cdot (1 - \frac{s_r}{p}) - E_{Mb,dyn} \cdot 2W_L \cdot s_r}{P_{Mb,leak} \cdot 2W_L \cdot s_r} \tag{3.20}$$

$$T_{cycle} = \frac{P_{leak} \cdot (1 - \frac{s_r}{p})}{P_{Mb,leak} \cdot 2W_L \cdot s_r} - \frac{E_{Mb,dyn}}{P_{Mb,leak}} \tag{3.21}$$

Once all parameters are known, this equation can be used to calculate the maximum length of the store and process cycles in order for duty-cycling to be effective. In Chapter 5 this equation is used for analysis based on values achieved through simulations.

Chapter 4

Implementation

This chapter presents how the system components were implemented in VHDL and some of the choices made in the process. Beside, the way the components work is explained in more detail. An overview of the implemented system is shown in Figure 4.1. It shows the different clock domains in the receiver chain. As can be seen, some components run on multiple clock domains. The clocks are generated by the clock controller (CC) and derived from the system input clock, 'clk in'.

4.1 Demodulator

The Demodulator was implemented in VHDL following the design presented in section 3. It's implementation is shown in Figure 4.2. It shows an implementation for $N = 16$ but to increase clarity not all the hardware is drawn. After folding, hardware is being re-used for different purposes, and thus, some control logic is required to manage the flow of data. For this purpose a state machine was implemented that cycles through N states while the samples of the current symbol are processed one by one.

During each state, the incoming sample $a_{n,i}$ is directly multiplied by each of the samples of symbol a_{n-2}. Since every symbol consists of N samples, these multiplications produce N intermediate results per state, specifically:

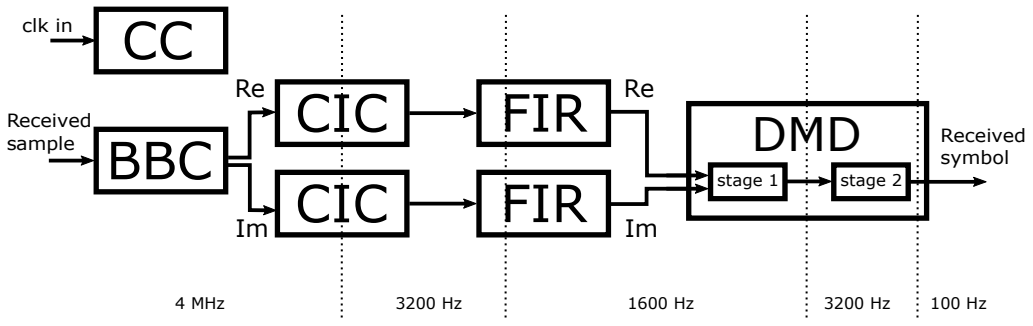

FIGURE 4.1: An overview of the system that is implemented in VHDL, with clock domains indicated by dashed lines.

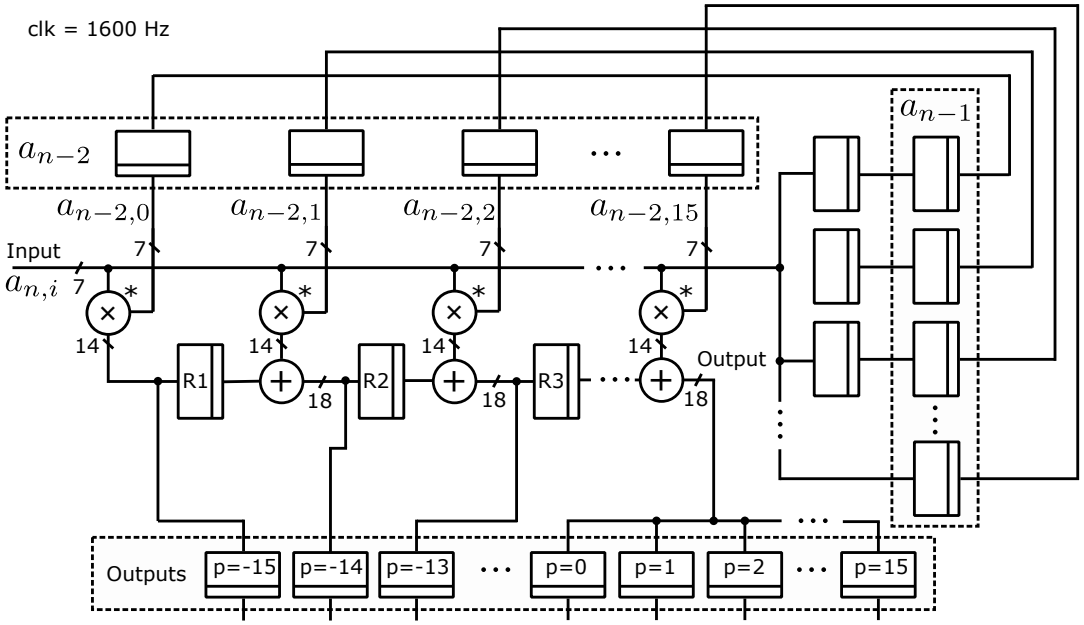

FIGURE 4.2: The implemented first stage of the demodulator for $N = 16$.

$$a_{n,i} \cdot \begin{pmatrix} a_{n-2,0} \\ a_{n-2,1} \\ \vdots \\ a_{n-2,N-1} \end{pmatrix}$$

The results of these multiplications are then added to previously stored sums, which will accumulate all of the multiplications in each path as the samples enter the first stage. In each state, the pipeline outputs a value belonging to path $p = N - 1 - i$ where i is the number of the state.

During the final state, $i = N - 1$, the intermediate values in the pipeline are also output. These values are the sums belonging to paths with shifts $p = -N$ through $p = -1$. This amounts to a total of $2N - 1$ outputs, one for each path in the demodulator. The pipeline registers are reset in the final state so they again contain zeroes as the next symbol comes in.

A visual representation of the workings of Figure 4.2, as described above, is shown in Table 4.1 for $N = 4$. It shows an example run for the first stage of the demodulator. Several cycles are shown for which example inputs have been chosen. Note that time does not start at time 0 when the system has just been turned on, but rather at time $t + 0$ where t is just an arbitrary amount of time after the system first started. This is to indicate the system in this example run has been running for a while and is not in its initial state.

The inputs were chosen such that at time $t + k$ the value at the input is equal to k. Thus for the first symbol that is input, its samples have values $a_n = \{0, 1, 2, 3\}$. This allows the inputs that are not shown in the table to be deduced as well, for example the inputs for the previous symbol that stared

at to time $t-4$. Its samples were valued $a_{n-1} = \{-4, -3, -2, -1\}$ and these values can be found in registers $a_{n-1,i}$ at time $t+0$. At time $t+4$ they have been shifted to the registers $a_{n-2,i}$ and are used for multiplication with the input. For simplicity, the inputs here are assumed to be purely real valued, although in reality they are complex values.

A generic parametrical representation of the values calculated in the first stage of the demodulator is shown in Table 4.2. It shows what the value in each register is calculated from. Some register values are denoted with a $'$ such as R_1'. This is to differentiate between dependency on the current value of registers and a previous value. For example, at time $t+0$ the input value of register R_2, accordingly named R_2, in, is calculated as $A_0 \cdot B_{-2} + R_1$. In the second cycle, the contents of R_2 are updated with this input, and thus R_2 should contain the $A_0 \cdot B_{-2} + R_1$ as it was calculated in the first cycle. However, since the contents of R_1 were also updated, this expression would be incorrect. Therefore the reference to R_1 in the expression is marked with a $'$, to illustrate that the current contents of the register were calculated in the previous cycle.

Going over the main path of the demodulator where no shift is applied, it can be seen exactly how it is calculated. The sum of the sample products for this path appears at the output in state 3: $D_0 \cdot D_{-2} + R_3$, with the contents of R_3 being $C_0 \cdot C_{-2} + R_2'$. R_2' here refers to the value of R_2 in state 2 where its content was $B_0 \cdot B_{-2} + R_1'$. This, in turn, is based on the value of R_1 in state 1 which is just $A_0 \cdot A_{-2}$. Chaining these values together reveals the output to be:

$$A_0 \cdot A_{-2} + B_0 \cdot B_{-2} + C_0 \cdot C_{-2} + D_0 \cdot D_{-2}$$

This indeed corresponds to the sample-by-sample multiplication of the path in the demodulator where no shift is applied; each sample the n^{th} symbol is multiplied by the respective sample of the $(n-2)^{th}$ symbol. Using in the values as they appear in Table 4.1 such that $a_n = \{0, 1, 2, 3\}$ and $a_{n-2} = \{-8, -7, -6, -5\}$, this results in:

$$0 \cdot -8 + 1 \cdot -7 + 2 \cdot -6 + 3 \cdot -5 = -34$$

4.2 Word Lengths

Word lengths affect power consumption because operations involving a higher number of bits require bigger adders and multipliers, as well as larger registers to store results. A smaller word length, on the other hand, will cause a loss of precision and will degrade the BER performance. Optimization lies in keeping the word lengths small to cut power consumption, but large enough to warrant sufficient performance.

t	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11
clk												
state	0	1	2	3	0	1	2	3	0	1	2	3
input	0	1	2	3	4	5	6	7	8	9	10	11
R_1, in	0	-8	-16	-24	-16	-20	-24	-28	0	0	0	0
R_2, in	0	-7	-22	-37	-12	-31	-38	-45	8	9	10	11
R_3, in	0	-6	-19	-40	-8	-22	-43	-52	16	26	29	32
R_1	0	0	-8	-16	0	-16	-20	-24	0	0	0	0
R_2	0	0	-7	-22	0	-12	-31	-38	0	8	9	10
R_3	0	0	-6	-19	0	-8	-22	-43	0	16	26	29
output	0	-5	-16	-34	-4	-13	-28	-50	24	43	56	62
$a_{n,0}$	-4	0	0	0	0	4	4	4	4	8	8	8
$a_{n,1}$	-3	-3	1	1	1	1	5	5	5	5	9	9
$a_{n,2}$	-2	-2	-2	2	2	2	2	6	6	6	6	10
$a_{n-1,0}$	-4	-4	-4	-4	0	0	0	0	4	4	4	4
$a_{n-1,1}$	-3	-3	-3	-3	1	1	1	1	5	5	5	5
$a_{n-1,2}$	-2	-2	-2	-2	2	2	2	2	6	6	6	6
$a_{n-1,3}$	-1	-1	-1	-1	3	3	3	3	7	7	7	7
$a_{n-2,0}$	-8	-8	-8	-8	-4	-4	-4	-4	0	0	0	0
$a_{n-2,1}$	-7	-7	-7	-7	-3	-3	-3	-3	1	1	1	1
$a_{n-2,2}$	-6	-6	-6	-6	-2	-2	-2	-2	2	2	2	2
$a_{n-2,3}$	-5	-5	-5	-5	-1	-1	-1	-1	3	3	3	3

TABLE 4.1: Possible values in an example run for the first stage of the demodulator. The values indicated by the blue boxes are the $2N - 1$ outputs for the first symbol-by-symbol multiplication.

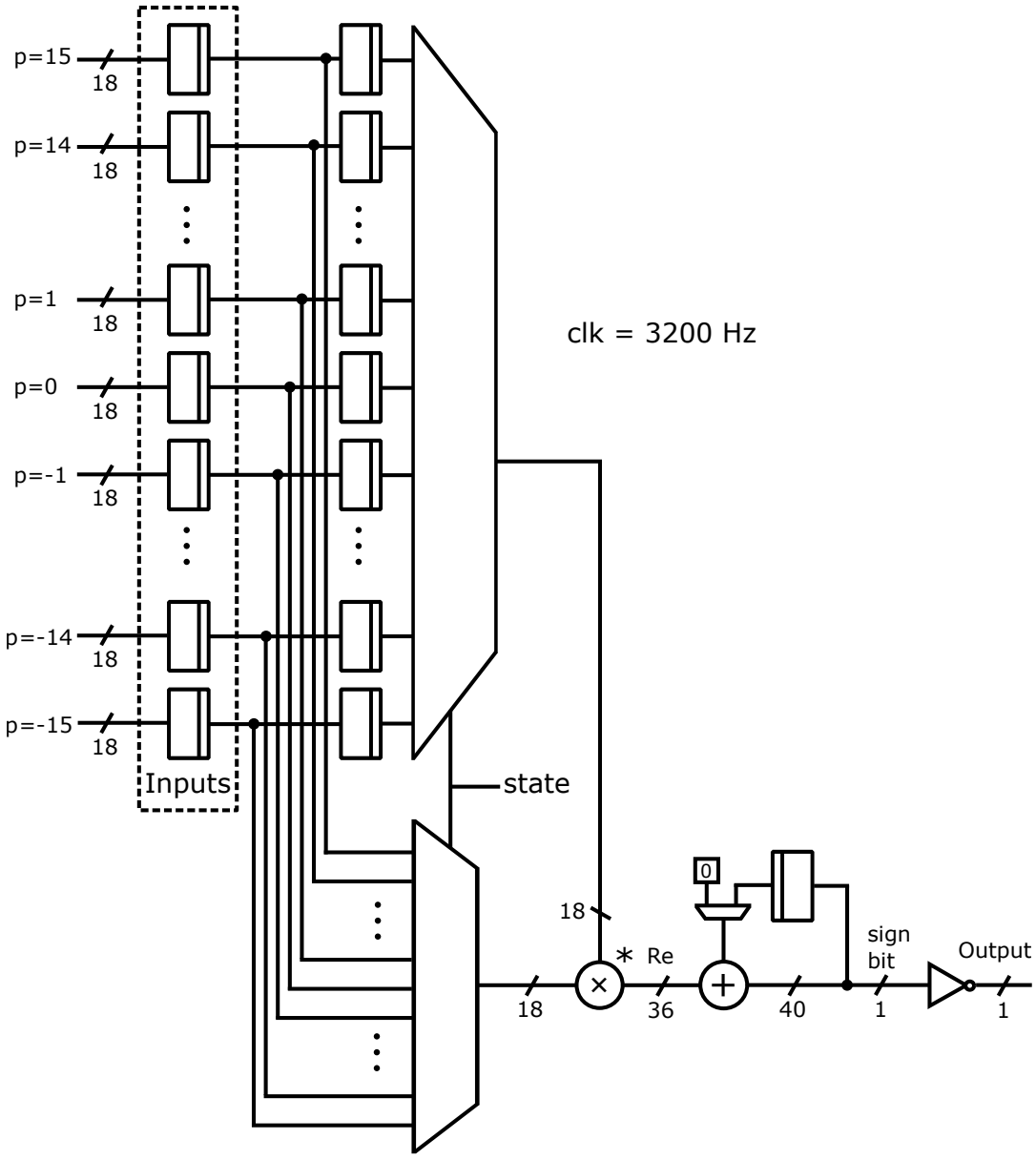

FIGURE 4.3: The implemented second stage of the demodulator for $N = 16$.

t	**0**	**1**	**2**	**3**
state	**0**	**1**	**2**	**3**
input	A_0	B_0	C_0	D_0
R_1, in	$A_0 \cdot A_{-2}$	$B_0 \cdot A_{-2}$	$C_0 \cdot A_{-2}$	$D_0 \cdot A_{-2}$
R_2, in	$A_0 \cdot B_{-2} + R_1$	$B_0 \cdot B_{-2} + R_1$	$C_0 \cdot B_{-2} + R_1$	$D_0 \cdot B_{-2} + R_1$
R_3, in	$A_0 \cdot C_{-2} + R_2$	$B_0 \cdot C_{-2} + R_2$	$C_0 \cdot C_{-2} + R_2$	$D_0 \cdot C_{-2} + R_2$
R_1	**0**	$A_0 \cdot A_{-2}$	$B_0 \cdot A_{-2}$	$C_0 \cdot A_{-2}$
R_2	**0**	$A_0 \cdot B_{-2} + R_1'$	$B_0 \cdot B_{-2} + R_1'$	$C_0 \cdot B_{-2} + R_1'$
R_3	**0**	$A_0 \cdot C_{-2} + R_2'$	$B_0 \cdot C_{-2} + R_2'$	$C_0 \cdot C_{-2} + R_2'$
output	$A_0 \cdot D_{-2} + R_3$	$B_0 \cdot D_{-2} + R_3$	$C_0 \cdot D_{-2} + R_3$	$D_0 \cdot D_{-2} + R_3$
$a_{n,0}$	A_{-1}	A_0	A_0	A_0
$a_{n,1}$	B_{-1}	B_{-1}	B_0	B_0
$a_{n,2}$	C_{-1}	C_{-1}	C_{-1}	C_0
$a_{n-1,0}$	A_{-1}	A_{-1}	A_{-1}	A_{-1}
$a_{n-1,1}$	B_{-1}	B_{-1}	B_{-1}	B_{-1}
$a_{n-1,2}$	C_{-1}	C_{-1}	C_{-1}	C_{-1}
$a_{n-1,3}$	D_{-1}	D_{-1}	D_{-1}	D_{-1}
$a_{n-2,0}$	A_{-2}	A_{-2}	A_{-2}	A_{-2}
$a_{n-2,1}$	B_{-2}	B_{-2}	B_{-2}	B_{-2}
$a_{n-2,2}$	C_{-2}	C_{-2}	C_{-2}	C_{-2}
$a_{n-2,3}$	D_{-2}	D_{-2}	D_{-2}	D_{-2}

TABLE 4.2: Generic representation using parameters of data flow and operations in the first stage of the demodulator.

Word length reduction is achieved through a combination of truncation and saturation. In some cases the least significant bits (LSB) are removed, whereas in other cases the most significant bits (MSB) are rarely used, and can thus be left out. Removing the LSB's can be done without additional hardware through truncation. The wires that transport these bits are simply removed and the rest of data continues with a shorter word length. If the MSB's are rarely used, saturation is used to remove these bits. If the value being saturated does not fit in the new word length, it is clipped to the highest or lowest possible representable value instead. This lowers the magnitude of the value but correctly keeps its sign.

The initial word length was chosen as low as possible while maintaining the BER-curve, determined through MATLAB simulation. The chosen input word length is 5 with 2 fractional bits (and therefore 3 integer bits), which is the input of the baseband conversion (BBC) block. For the other components, the CIC and FIR filters and the demodulator, the input word lengths as they were determined in [14] were kept.

The word lengths do not increase in the base-band conversion component since its inputs are simply rerouted to different outputs without changing their values. In the CIC filter, shown in Figure 4.4, the word length rapidly increases due to the continuous accumulation of input samples. The magnitude of the value inside the accumulator can potentially reach the decimation factor multiplied by the highest magnitude input value. With a decimation

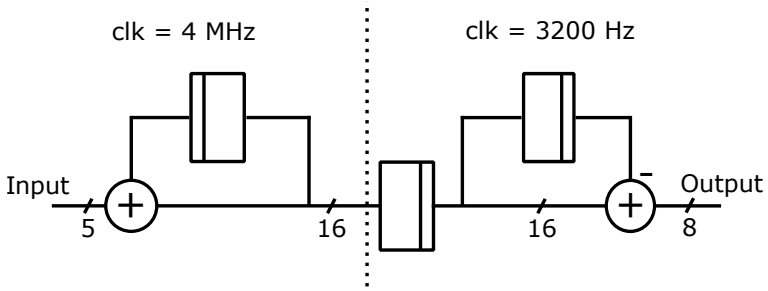

FIGURE 4.4: Datapath of the CIC filter implementation, showing the two clock domains (for $N = 16$).

factor of 1250, this increases the word length required to represent the highest internal value to 16 bits. This comes from the fact that $2^{10} < 1250 < 2^{11}$, adding 11 bits to the input word length allows for a times 2048 higher maximum representable value.

At the output the word length is reduced by truncating the 2 LSB's and removing the 6 MSB's through saturation. This may seem like a lot of MSB's to remove. However, upon inspection of the full precision output of the CIC-filter, it was found that very rarely (in two out of 160 000 samples) the 6 MSB's were actually needed to represent the output. In these cases they were saturated to the nearest representable value, minimising the error made. This leaves the output of the CIC-filter at 8 bits.

Figure 4.5 shows the implementation of the FIR filter using adders and bit shifts. Negative coefficients are subtracted instead of added, which is generally done by inverting the input and setting the adder's carry-in to one, the logic for this is not drawn here as it was generated by the compiler. The coefficients that the shifts represent are $\{4, -1, -8, 8, 32, 32, 8, -8, -1, 4\}$, which are based on the coefficients found in [14]. These coefficients are equal to the powers of two: $\{2^2, -2^0, -2^3, 2^3, 2^5, 2^5, 2^3, -2^3, -2^0, 2^2\}$, and the amount of bit shift seen in Figure 4.5 corresponds to the magnitude of these powers (e.g. a bit-shift of 5 to the left corresponds to a multiplication with $2^5 = 32$).

With the input word length being 8, the largest possible word length in the FIR filter pipeline was calculated to be 15. At the output the word length is then reduced to 7, by removing the single MSB and seven of the LSB's in the same way as for the CIC-filter. This method of downsizing was also used and presented in more detail in [14]. No extensive statistical analysis was applied to the inputs and outputs of the components to find optimal word lengths for them, but an effort was made to narrow down the bits that were most likely to contain the most significant information, again by inspecting the stream of outputs of the FIR filter. For the particular output of 80 000 values inspected, this MSB was in fact never used.

The demodulator was implemented as two stages shown in Figures 4.2 and 4.3. The input word length is equal to 7 and this is increased to 14 after the multiplier. Since these multipliers perform complex conjugate multiplications where the real part is calculated as $ac + bd$, there is a single case where

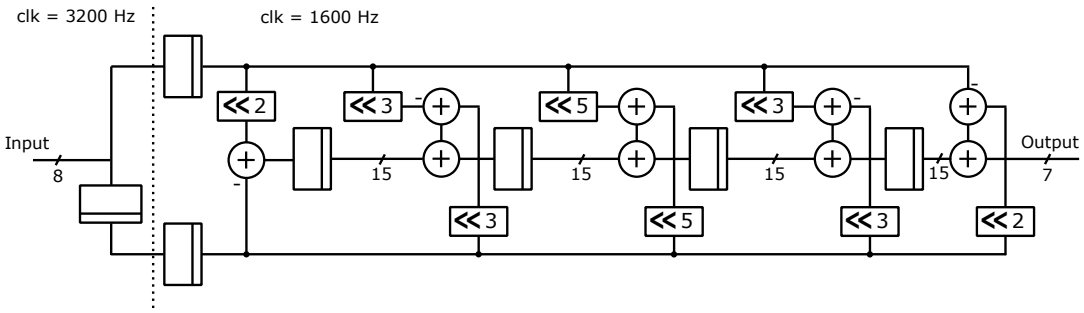

FIGURE 4.5: Datapath of the FIR filter implementation, showing the two clock domains for $N = 16$.

a representation of 14 bits is insufficient. This occurs when $a = b = c = d = -64$, the multiplication result will be 8192, while the maximum two's complement representable number using 14 bits is equal to 8191. In this case, the value is simply assumed to be 8191, introducing a minor error.

The sum of each of the paths is then allowed to grow to 18 bits because if $N = 16$, the values can become up to 16 times larger. For the shorter paths, this is not the case, however to manually set the word lengths for each of the paths would make the implementation in VHDL more difficult and much less generic. Also, the distinction between the paths is not as clear cut due to the folding of the demodulator. In this case the word length optimization was left to the compiler.

The registers marked as 'outputs' in Figure 4.2 are the same as the ones marked 'Inputs' in Figure 4.3. After multiplication the word length can become up to 36 bits long for the same reason as described earlier. The value in the accumulator that follows is then allowed to grow with 4 bits, after which only the sign of the result matters for the output, reducing the output to just a single bit.

Note that the above mentioned word lengths correspond to the system implementation for $N = 16$, even though the datapaths are shown for $N = 4$. The word lengths might need adjustment for other values of N. For example, when $N = 32$, the decimation ratio of the CIC-filter is reduced from 1250 to 625, and the sums it calculates are likely to become only half as large (adding 625 inputs compared to 1250 inputs is expected to yield a sum of half the magnitude). In turn, this means that instead of removing the 6 MSB's at the output of the CIC-filter, the top 7 can be removed. In the demodulator, higher N results in more and longer paths, and so the word lengths need to be increased to accomodate for the higher values that this will yield.

4.3 Scalability

The system can be scaled by increasing the number of samples per symbol. It was implemented in VHDL in a generic way such that the number of samples

per symbol can be set and the required hardware will be generated accordingly. Register arrays are scaled up, the state machine counts more states and additional multipliers and adders are generated in the pipeline for larger N. The word lengths, however, as stated earlier, are not dynamically calculated by the VHDL code and have to me manually adjusted where necessary.

Increasing the sampling rate allows a larger frequency offset to be tolerated. However, since the number of calculations in the demodulator scales with N^2 this will drastically increase the power consumption for larger N. A way to reduce the quadratic scaling behavior is to leave out some paths in the demodulator. The effect this has on the BER performance of the system has been investigated in [15]. Especially for larger N, not all paths are necessary to obtain sufficient BER, and only using a few paths with low shift values p around the non-shifted $p = 0$ path would suffice. For example using the paths that belong to values $-1 < p < 1$ would result in a total of 3 paths, with a maximum shift magnitude of 2.

With M the magnitude of the highest shift (e.g. $M = 1$ implies shifts of 1, 0 and -1, for a total of 3 paths) the total number of complex additions and multiplications required can be described by the following equations:

$$Multiplications = N + \sum_{m=1}^{M} 2(N - m) \tag{4.1}$$

$$Additions = N - 1 + \sum_{m=1}^{M} 2(N - 1 - m) \tag{4.2}$$

Which can be rewritten as:

$$Multiplications = 2MN + N - M^2 - M \tag{4.3}$$

$$Additions = 2MN + N - M^2 - 3M - 1 \tag{4.4}$$

Which scales with N rather than N^2. Therefore the power consumption of a demodulator implementation using a constant number of paths is expected to scale up linearly instead of quadratic with N. As can be seen in the equations, less additions than multiplications are required. This is because when adding up the results of N multiplications, $N - 1$ additions are performed, which is less.

Figure 4.6 shows a visual representation of removing demodulator paths (indicated by the green lines). It shows an implementation for $N = 4$ and $M = 1$ and only shows the complex multipliers. Even though more paths are removed than are used in this example, a relatively small amount of the hardware is removed in this case.

The way in which the first demodulator stage was folded already reduces the amount of hardware required, which limits the effectiveness of reducing the number of paths. After folding the demodulator in the way presented

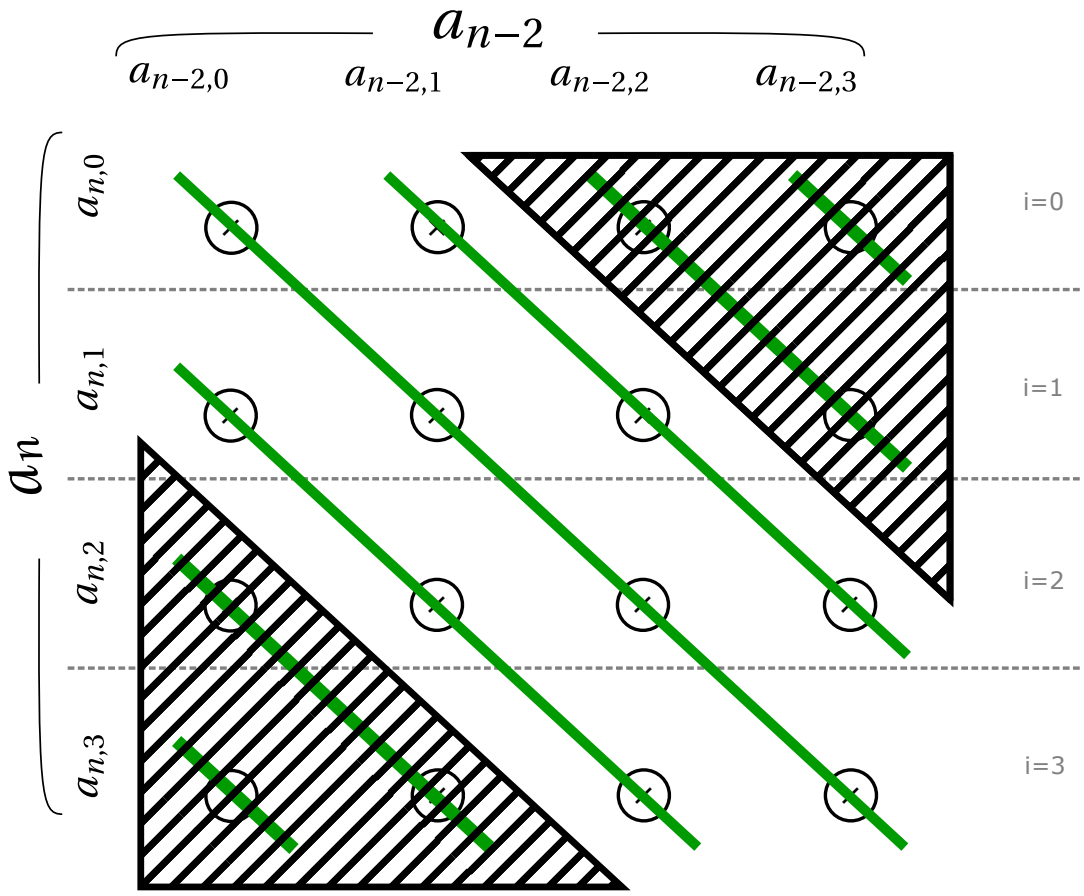

FIGURE 4.6: First demodulator stage for $N = 4$ with only 3 paths being used ($M = 1$).

in Chapter 3, its hardware scales with N already and thus the benefit of removing paths is greatly diminished. Still, some hardware reduction can be achieved in this scenario.

For example, when $N = 4$ and $M = 1$ as in Figure 4.6, it can be seen that only 3 multiplications are ever performed in the same cycle (cycles are indicated by the grey dotted lines). Therefore in a folded implementation, during any given state, a maximum of three complex multipliers would be sufficient. The regular folded structure contains four multipliers, and thus one could be optimized away in this scenario. It is important to note here that the number of multipliers in this case is not dependent on N at all, but purely on the number of paths. As an extreme example, in an $N = 32$ implementation using only 3 paths, three multipliers would still be sufficient to make the folded demodulator work.

Chapter 5

Results and Discussion

This chapter presents the results obtained through simulation of the implemented system. The improvements made are discussed and compared to literature, and some suggestions for improvements are given.

Figure 5.1 shows the setup used to perform measurements on the system. The system VHDL is connected to a testbench that provides the input signals from a file and the clock. Only the main clock signal of 4Mhz is provided, the system has a clock controller that generates the other clocks from it. The testbench also collects the outputs and writes them to a file. Depending on the configuration, different input and output files are selected by the testbench. In this way, input signals with different signal to noise ratios can be simulated. The input files were generated using MATLAB, generating both the signal and the channel noise (additive white Gaussian noise). Modelsim was used to compile the VHDL and run the simulation, and visualize the waveforms for system verification where necessary.

The same setup was used to perform power measurements, but instead of simulating the VHDL files, Modelsim will simulate the hardware synthesised by the Synopsys Design Compiler. During synthesis a cell library is used to implement the VHDL structures in hardware. Both a SVT and HVT cell library were used for power measurements. During post-synthesis simulation in Modelsim the timing information of the system run is stored in a .SAIF file, which is then used by the Synopsys Power compiler to calculate power consumption. For power measurements the outputs of the system are not important, they are only used to verify correct functionality of the post-synthesis system.

Synopsys reports the power consumption split into leakage power, switching power and internal power. Leakage power is the static power consumption that results from leaking currents and is consumed even when the system is inactive. Switching power and intenal power are dynamic power components where switching power is the power lost by charging and discharging the output when making a logical transition (e.g. from 1 to 0 or vice versa) while the internal power consumption is caused by possible short circuits that arise during the switching of transistors inside logic gates. For example, when one transistor opens and another in series closes, momentarily they might both be half open and create a short between the voltage supply and

ground. Both these dynamic power consumptions depend on the number of logic transitions in the circuit.

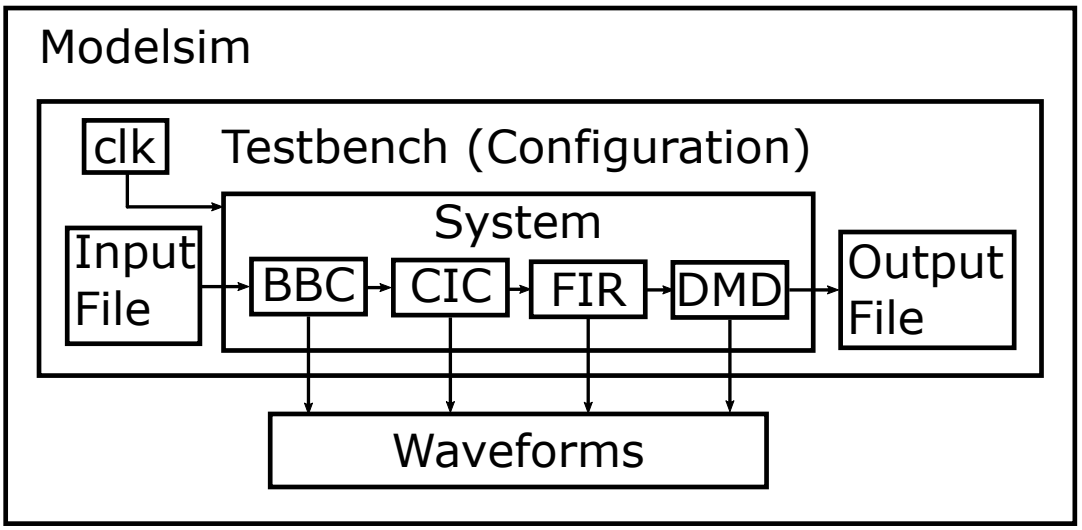

FIGURE 5.1: Visual representation of the simulation platform used in measuring system performance.

5.1 Performance

5.1.1 Bit error rate

The pre-synthesized system was simulated using Modelsim because it simulates faster than a post-synthesis simulation. Functionally the pre and post-synthesis systems were verified to behave identically and thus for the purpose of measuring functional performance, it is sufficient to simulate the pre-synthesis system. Figure 5.2 shows the BER-curve of the system for 16 and 32 samples per symbol, for a sample size of 50.000 symbols.

To test the system's performance with respect to frequency offset, the BER values for inputs with increasing frequency offset and an E_b/N_0 equal to 8dB and 11dB were calculated. These values are plotted in Figure 5.3 for both the 16 and 32 samples per symbol systems. The frequency offset was increased from 100 Hz to 1900 Hz.

It is expected that the performance of the N=16 system starts to deteriorate around 700 Hz of frequency offset and for the N=32 system this happens at an offset of 1400 Hz. This is due to the fact that the frequency offset that can be tolerated equals half of the sample rate f_s, where $f_s = R_s \cdot N$, with R_s being the symbol rate of 100Hz. Therefore, the system can at most tolerate a frequency offset up to $\pm\frac{1}{2}f_s = \pm\frac{1}{2} \cdot 100 \cdot 16 = \pm 800Hz$ when N=16, and $\pm\frac{1}{2} \cdot 100 \cdot 32 = \pm 1600Hz$ that amount for N=32.

In practice, it can be seen that the BER already deteriorates at around 500-600Hz for $N = 16$ and 1200Hz for $N = 32$. This is explained in part by the FIR-filter used, which is not perfectly rectangular and has a transition

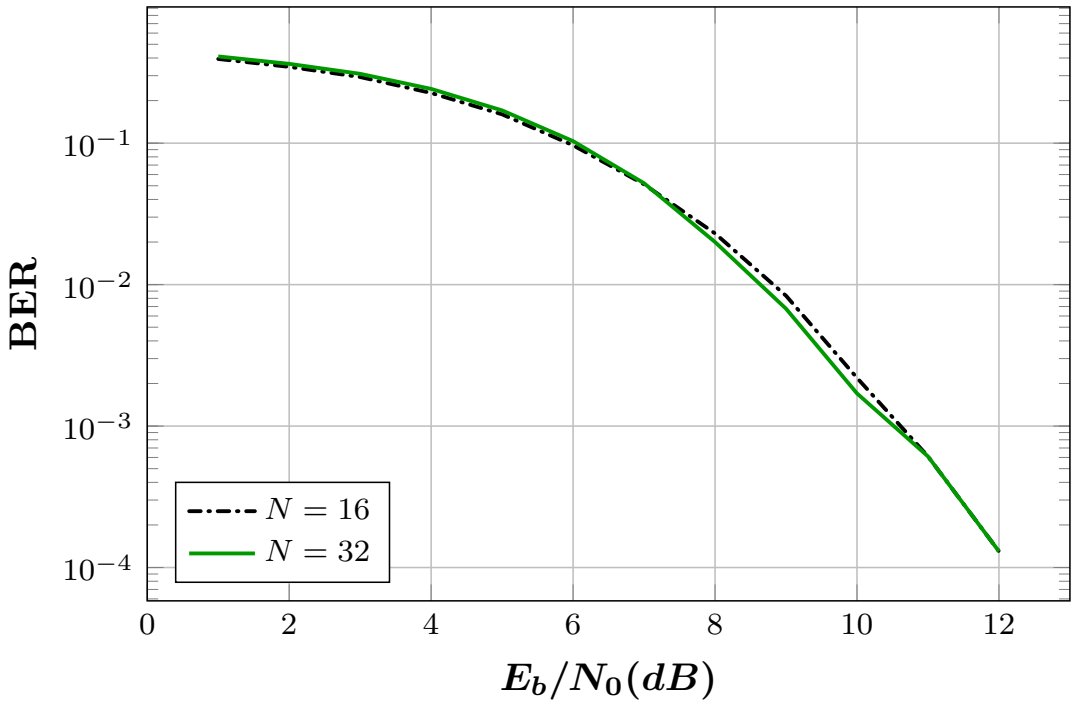

FIGURE 5.2: The BER curve of the system.

band of around 100Hz [14]. Also the fact that the signal has a bandwidth of $2 \cdot 100Hz = 200Hz$ (twice the symbol rate) reduces the maximum tolerable frequency offset, because when the signal's center frequency is offset by for example 800Hz, half of the signal band will fall between 800Hz and 900Hz and would still violate the Nyquist Theorem. For $N = 32$ these values are doubled. The FIR filter has a transition band of around 200Hz in this case, and therefore these implementations can maintain their BER up to around 1100-1200Hz.

5.1.2 Power consumption

To estimate the power consumption of the system, the post-synthesis system was simulated in Modelsim and analysed with Synopsis Design Compiler. Power optimisations used include all those presented in Chapters 3 and 4 but with all the demodulator paths present (e.g. no paths removed). However, while power measurements were performed using HVT cell libraries, to allow a fair comparison with the results obtained in [14], an implementation using an SVT library was also simulated.

The power simulation results using an SVT cell library are shown in Table 5.1. It shows the switching power, internal power and leakage power, as well as the total power consumption for each component. These components are the Baseband Converter (BBC), Cascade-Integrator Comb filter (CIC), Finite Impulse Response filter (FIR) and demodulator (DMD). The demodulator consumes more than $12\mu W$, which is around 80% of the total consumption. This is caused by the fact that it contains most of the hardware, as well as

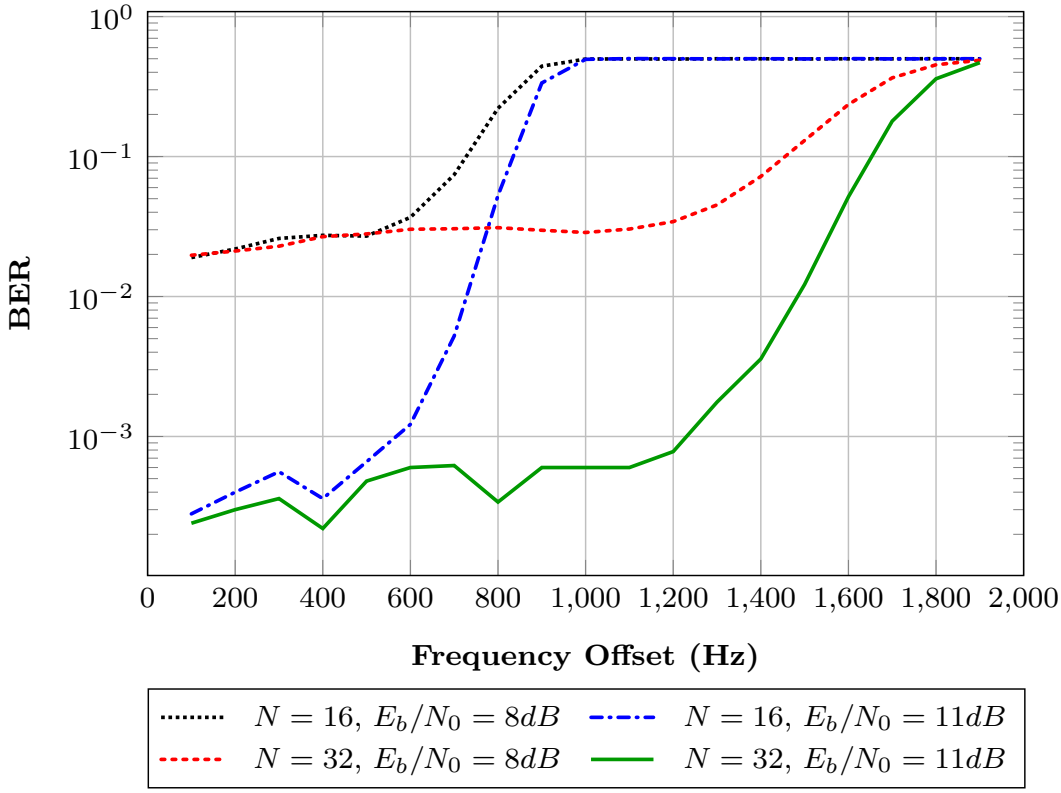

FIGURE 5.3: BER performance of the system at increasing frequency offset

by the long word lengths within the demodulator, which increase to up to 40 bits.

In the BBC and CIC components, power consumption is dominated by switching power and internal power. It is responsible for 98% and 93% of the total power in the BBC and CIC respectively. The reason for this is the fact that the clock rate is relatively high in these components (4MHz, see Figure 4.1). As the clock rate slows down in the FIR and DMD components, the leakage power starts to dominate the total consumption. Therefore within both FIR filter and the demodulator, approximately 98% of the power consumption is caused by leakage. A graphical representation of the power consumption is shown in Figure 5.4.

	Switch power (μW)	Int. power (μW)	Leak. power (μW)	Total power (μW)
dmd	0.0332	0.145	12.0	12.2
fir	0.001033	0.00653	0.471983	0.479
cic	0.1327	1.824	0.166373	2.12
bbc	0.25	0.75	0.021248	1.02
Total	0.417	2.75	12.7	15.8

TABLE 5.1: Power values using the SVT cell library used in [14]

To overcome the leakage power, folding techniques are used to decrease the number of implemented adders and multipliers as explained in previous chapters. Another approach for leakage reduction is using another type

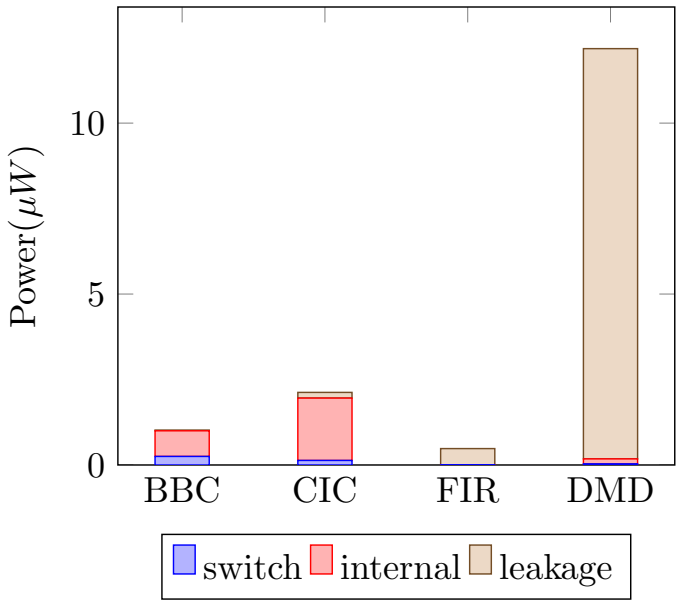

FIGURE 5.4: Power consumption of the system using the SVT cell library used in [14]

of standard cells called HVT with higher threshold voltage [4]. Using HVT cells for synthesis reduces leakage power, however it increases the delay of the cells. Since this system runs at a relatively low clock rate, timing is not critical. Therefore, the HVT cell library can be used to synthesize the system without violating the timing constraints of our design.

The simulated power consumption of the synthesized system using an HVT library has resulted in the values shown in Table 5.2 and visualized in Figure 5.5. When comparing power values of the SVT and the HVT implementation, it can be seen that while the internal and switching power have not changed considerably, the leakage power consumption is greatly reduced as expected for an HVT library, in fact by a factor of four across all components. This reduces the total power consumption from 15.8μW to 6.25μW.

	Switch power (μW)	Int. power (μW)	Leak. power (μW)	Total power (μW)
dmd	0.0308	0.146	2.96	3.14
fir	0.000912	0.00622	0.113739	0.1209
cic	0.119	1.812	0.04025	1.972
bbc	0.232	0.757	0.005501	0.994
Total	0.383	2.75	3.12	6.25

TABLE 5.2: Power values for N=16 using 40nm HVT cells

The power consumption of the system was also measured for the case where $N = 32$, the results of which are presented in Table 5.3 and visualized in Figure 5.6. It shows how the internal and switching power of the demodulator scale quadratically with the number of samples per symbol, however the leakage power scales linearly due to the way the hardware was

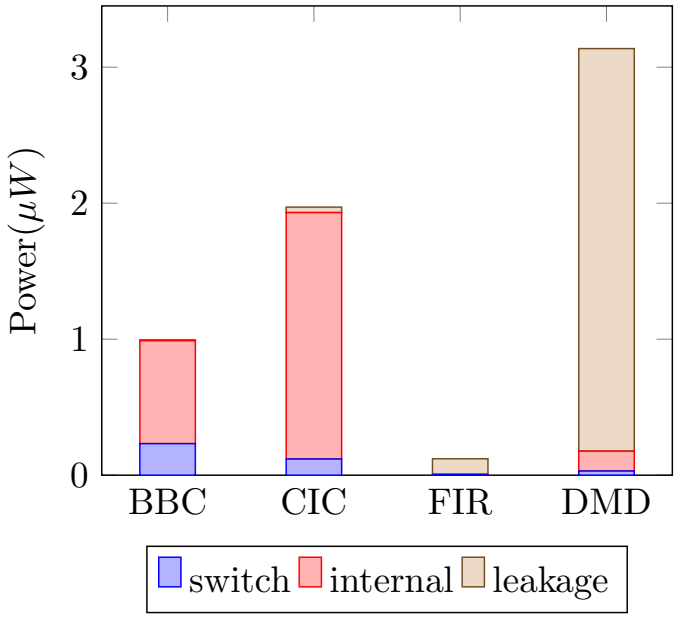

FIGURE 5.5: Power consumption of the system (N=16) using 40nm HVT cells

folded. This results in a much smaller increase in power consumption when increasing N. The full system consumes only 50% more power going from a $N = 16$ to $N = 32$ configuration. The power consumption increased from $6.25\mu W$ to $9.54\mu W$ which originates almost entirely from the demodulator ($\frac{6.424-3.14}{9.54-6.25} = 99.8\%$ of the increase occurs in the demodulator).

The BBC component is identical for both configurations, and so are its power consumptions, as the choice of N is made in the CIC filter which comes after it. For the CIC-filter the same operations happen, except to achieve higher amounts of samples per symbol the decimation ratio is lowered. It consumes roughly the same amount of power for both N. The FIR filter runs at double the clock frequency for $N = 32$ compared to $N = 16$ and therefore consumes double the dynamic power. Its hardware stays the same and so does the leakage power.

	Switch power (μW)	Int. power (μW)	Leak. power (μW)	Total power (μW)
dmd	0.111	0.558	5.75	6.424
fir	0.00181	0.01247	0.113751	0.128
cic	0.1193	1.815	0.040252	1.974
bbc	0.232	0.757	0.005501	0.994
Total	0.464	3.17	5.91	9.54

TABLE 5.3: Power values for N=32 using 40nm HVT cells

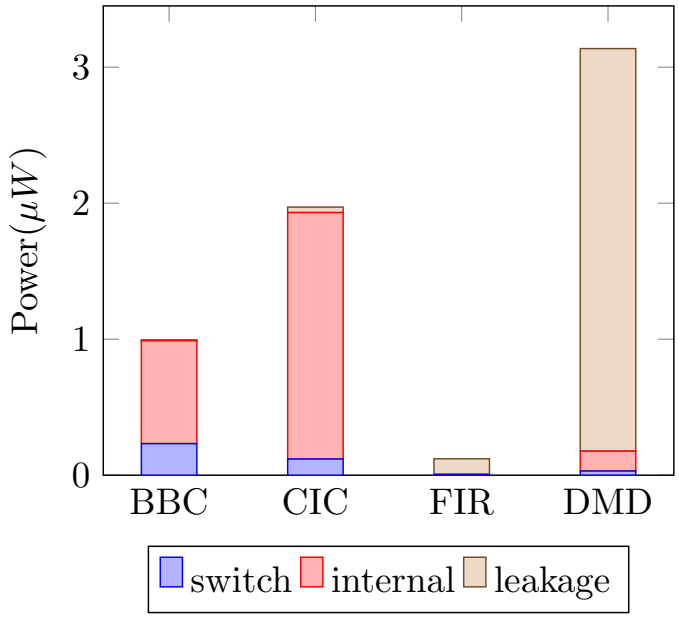

FIGURE 5.6: Power consumption of the system (N=32) using 40nm HVT cells

5.2 Comparison

Table 5.4 shows the power consumption of the system as it was previously implemented in [14]. Since this implementation was simulated using an SVT cell library, to keep the comparison fair, it will be compared to the power values of the improved system simulated using an SVT library as well. Both systems use 16 samples per symbol ($N = 16$).

When comparing Table 5.1 with 5.4, it can be seen that while the total power consumption of the CIC and FIR filters has improved slightly, significant improvements can be seen in the total power consumption of the demodulator. A comparison is shown in table 5.5. The total power consumption has been reduced by 27% in the CIC filter and 54% in the FIR filter. For the FIR filter this is mainly caused by a reduction in leakage power consumption, as this is by far the biggest contributor to its total consumption. The demodulator consumes 83% less power compared to the previous implementation, mainly caused by the reduction in leakage power consumption. The BBC component could not be compared since its consumption in the previous implementation was not reported.

Figure 5.7 shows the demodulator power consumption for the SVT implementation and HVT implementations where $N = 16$ and $N = 32$. It clearly shows the leakage power reduction by a factor four going from a SVT to HVT cell library, and then a doubling of the leakage power and a quadrupling of the dynamic power when increasing N from 16 to 32. The majority of the difference in power consumption is present in the demodulator, the

	Switch power (μW)	Int. power (μW)	Leak. power (μW)	Total power (μW)
dmd	0.058	0.662	69.159	69.880
fir	0.005	0.034	1.005	1.044
cic	0.117	2.639	0.161	2.917
bbc	-	-	-	-
Total	0.18	3.335	70.325	73.841

TABLE 5.4: Power values achieved in [14]

	BBC	CIC	FIR	DMD	Total
This work	1.02	2.12	0.479	12.2	15.8
Prev. work[14]	-	2.917	1.044	69.880	73.841
Improvement	-	0.80 (27%)	0.565 (54%)	57.7 (83%)	58.0 (79%)

TABLE 5.5: Comparison between power consumption (μW) of this system and the previous system implementation[14]

other components consume roughly the same power. When comparing different implementations it therefore makes sense to look at the demodulator individually.

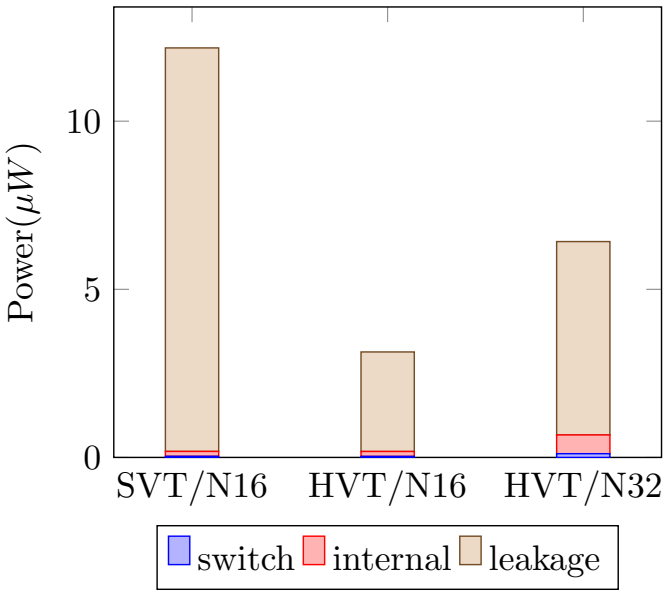

FIGURE 5.7: Power consumption of the demodulator for SVT and HVT implementations

5.3 Duty Cycling

A model for duty-cycled power gating by using a memory block as a sample buffer was presented in Chapter 3. Now that the power consumption of the system is known, these results can be used in the model. For the memory

block the power consumption was estimated by simulating an array of 128 7-bit registers in VHDL, where every register is written to and read from once, at both a high (4MHz) and a low (1600Hz) clock speed. From this simulation the memory's dynamic energy and leakage power consumption per bit were determined.

The memory block is assumed to be placed between the FIR-filter and demodulator which means that the sample rate into the memory equals 1600Hz, being the sample rate from the FIR-filter to the demodulator. The processing speed of the samples from memory is assumed to be 4MHz, equal to the system's main clock. The word length of the samples going from the FIR-filter to the demodulator is 7 bits. We then have the values shown in table 5.6.

Name	**Symbol**	**Value**
Sample rate*	s_r	1600
Wordlength*	W_L	7
Processing speed	p	4000000
Leakage Power*	P_{leak}	$2.96\mu W$
Memory energy consumption per bit	E_{Mb}	$1.04pJ$
Memory leakage power per bit	$P_{Mb,leak}$	$1.37nW$

TABLE 5.6: List of parameters and their values used by the model. Parameters dependend on the location of the memory are marked with *.

These values can be substituted into Equation 3.11 to calculate the maximum amount of time for which power can be switched off, such that the memory block does not consume more power than it saves. The result is that the demodulator has to be power gated at least 10 times per second, in order for the memory block to remain sufficiently small such that it does not consume more power than the leakage power it reduces. Switching less often would result in more samples coming in and thus a larger memory that consumes more power.

This model, however, does not take into account the energy it would cost to drain the demodulator of power and it assumes there is no leakage power at all when the demodulator is switched off. Although the memory block consumes less power if the cycle time is shortened (less samples needing to be stored leads to smaller memory), the power consumed in power gating the system goes up, because it is toggled on and off more often. There will likely be power losses due to overhead with each toggle.

When the idea to duty-cycle part of the system was considered, the leakage power consumption was assumed to be very large. However, with the applied power optimizations and usage of an HVT cell library, the leakage power has been lowered considerably. It therefore appears to be impossible to gain an advantage from duty-cycling a part of the system, as the leakage power saved would not offset the additional overhead power consumption of the memory block and power-gating circuitry.

Chapter 6

Conclusion

The Internet of Things (IoT) involves large numbers of devices connected in a wireless sensor network (WSN) able to communicate with each other. These devices have specific requirements such as long range communication, low production costs and long battery life. Many IoT services require a very low data rate in the order of tens of bytes, sent only several times a day [13].

For these applications, Low-Power Wide Area Network (LPWAN) and more specifically Ultra-Narrowband communication is the best suited technology, however it heavily suffers from frequency offset. To overcome the impact of a frequency offset in these applications, a frequency offset tolerant demodulation and detection scheme can be used called Double Differential Phase-Shift Keying (DDPSK).

In this thesis an implementation of a DDPSK demodulator for ultra-narrow band communication was presented. It is based on a previous implementation of the system in which the main power consumption was contributed by leakage power [14]. For each of the digital system components the power consumption was optimized through use of power saving techniques such as folding, multiplierless design and word length reduction. Additionally the feasibility of duty cycling in part of the system for leakage power reduction was investigated.

The system was simulated using modelsim and analysed with the synopsys design compiler. It was compared to the previous implementation to measure improvement in power consumption. While the previous implementation had a total power consumption of $73.8\mu W$, the improvements presented in the current thesis brought this down to $15.8\mu W$ (almost 75% power saving). This does not include the use of a more efficient cell library to keep the comparison fair. When also applying a HVT cell library, the power consumption is pushed further down to $6.25\mu W$ (more than 90% power saving).

It is possible to increase the number of samples per symbol N to tolerate more frequency offset. To see how the power consumption of the implementation scales with N, a $N = 32$ samples per symbol implementation was also simulated. For $N = 32$ the system consumed $9.54\mu W$, an increase of only 50% compared to the HVT implementation for $N = 16$. For the demodulator specifically, which previously scaled quadratically, going from $N = 16$

to $N = 32$ only doubled the power consumption. This improvement was obtained through using a folded structure which allowed hardware to be reused.

The frequency offset tolerance was measured for implementations with N equal to 16 and 32 samples per symbol. It was shown that the offset tolerance of the implemented demodulator performs as expected. The simulations showed that for N equal to 16 and 32, the BER performance does not change for frequencies up to 500Hz and 1200 Hz, respectively.

Duty-cycling the power in part of the system was not implemented but modeled in order to estimate its effect on the power consumption. It was found that with the improved leakage power consumption there is not as much to save anymore by power gating the system. Since the proposed memory block also consumes power itself it was estimated that when power gating only the demodulator, it has to be turned off and on 10 times per second to break even on the power consumption of the memory. This estimation assumes there are no overhead costs for power-gating such as transition times between the on and off state, that there is no power consumption in the power gating control circuitry and no leakage power consumption by the hardware while it is powered down. When also taking these costs into account, the idea of duty-cycling the power is considered unlikely to improve the power consumption.

The following sections of this chapter will answer and discuss the research questions and suggest future work optimizations to the system that can improve it even further.

6.1 Discussion

This section will answer the research questions that were presented in Chapter 1. Each answer will be motivated by a brief discussion:

How can the components of the receiver system be optimized for power?

Power saving techniques from literature were investigated and presented in Chapter 2. Several power saving techniques were applied in designing the receiver components as described in Chapter 3, in order to optimize for power. The techniques applied include polyphase decomposition, multiplier-less design, hardware folding and usage of an HVT cell library. While all of these techniques reduce the leakage power consumption, polyphase decomposition more specifically targets dynamic power consumption. Hardware folding increases dynamic power consumption slightly, but provides a much larger reduction in leakage power consumption.

The first components in the system, the baseband conversion (BBC) block and CIC filter were shown to be responsible for a small part of the total power consumption. These are already simple components and their optimization (even if any further optimization is possible) will not lead to noticable power saving. Effort made to improve them would therefore have resulted in relatively little overall improvement.

The FIR-filter was improved upon by noting that half of the output samples were not used due to the decimation factor of two after the filter. By applying polyphase decomposition the decimator was moved in front of the filter. This halves the clock speed of the component as well as the dynamic power consumption. It also allowed the implementation to use less registers, slightly reducing the leakage power consumption.

A second improvement made to the FIR-filter was the removal of the multipliers by implementing them as bit-shifts and adders, commonly referred to as multiplier-less design. The filter coefficients were first rounded to the nearest power of two, which had no significant impact on the filter's behaviour. This allowed the coefficient multipliers in the filter to be implemented without adders and a single bit-shift which requires no logic, only wires.

An optimization for dynamic power that was applied to the demodulator was the mathematical rewriting of complex multiplications from four multiplications to three, at the cost of increasing the additions from two to five. Since multipliers are much more expensive than adders in terms of area and power consumption this improves both the dynamic and leakage power consumption. Through all optimisations, the total dynamic power consumption of the system presented in [14] was reduced from $3.515\mu W$ to $3.167\mu W$ (a 5% reduction). However, given the high leakage power consumption, dynamic power reduction was not the main focus.

How can the component's leakage power consumption be reduced?

In the presence of high leakage power consumption, some power saving techniques are more suitable. Hardware folding is one of the techniques that could be used. Folding involves increasing the clock speed while reducing the amount of operation per clock cycle. The total number of operations per time unit remains the same, but it frees up hardware that can then be removed. For example five additions can be performed in one clock cycle by five adders, or in five clock cycles by a single adder. In the second case, the other four adders can be removed. Reducing the hardware in the system naturally reduces the amount of leakage power. The dynamic power consumption remains unchanged because the amount of operations performed does not change.

The demodulator showed most improvement as it is the main contributor to the total power consumption. Leakage power was dominant in this component. It consists of two differential stages which perform the double differential demodulation. These stages were considered separately and both their datapaths were folded as shown in Chapter 3. In the first stage the amount of adders and multipliers was reduced from N^2 to N, with N being the number of samples per symbol. In the second stage the amount of hardware went from $2N-1$ adders and multipliers to a single adder and multiplier. Slightly more control logic and memory was added to manage the dataflow in the folded structure.

Another way of reducing the leakage power consumption is through usage of an HVT cell library. It is very well suited for low speed systems such as the receiver designed in this thesis, where timing is not critical. The benefit of HVT cells is that they leak much less power than SVT technology. The HVT cell library reduced the power consumption of the system by a factor of 2.5.

Finally, leakage power can be reduced by temporarily switching off components through power gating. The main idea is that the hardware is switched off for most of the time and all the processing is done at a higher speed in less time. In this way, the energy dissipated due to leakage decreases. This is referred to as duty cycling and leads to the last research question.

To what extend can the demodulator be power-gated, in a duty cycling fashion, in order to reduce the leakage power consumption?

As explained in Chapter 3 the main goal of power gating is to reduce the leakage power of a component by turning it off for a while. To give a component the time to turn off, a memory block would have to be placed in front of it. The memory then stores all the samples while the component is off such that when the component turns on again, it can process these samples at a faster rate. This increases the instantaneous dynamic power, but not the total energy of the calculations performed. Meanwhile, it lowers the leakage power consumption.

However, the results obtained in Chapter 5 show that the leakage power in the improved system is much lower than it initially was. In fact the leakage power now only contributes $3.12\mu W$, just 50% of the total power consumption. It is therefore unlikely that power gating can be used to reduce this further due to the additional costs of the memory block and power control circuitry.

Ultimately the aim of power gating was to reduce the idle time during which hardware can leak power. There may be another way to achieve this, which is through additional hardware folding. This is elaborated in the next section.

6.2 Further improvements

To decrease the power consumption further, there are other techniques that can be helpful. These are improvements that could have been made to the design but were not implemented due to time constraints.

6.2.1 Folding twice

The first stage of the demodulator was folded in one direction, but there is another additional direction into which it can be folded. This collapses all the multiplications and additions into a single adder and multiplier unit which correspondingly runs at a higher clock-frequency. It will perform all of the additions and multiplications and will therefore require additional control circuitry and registers to manage the dataflow. The demodulator will start to

look more like a processor and might even be able to take over the operations of another component like the FIR-filter, due to the low speed at which these operations are performed.

An additional fold in the demodulator will benefit the scalabilty of the design. Where originally N^2 multipliers and adders were required to compute the first stage, the first folding decreases the number of adders and multipliers to N. This number doubles when doubling the sample rate as it increases the number of samples per symbol. When folding down to a single multiplier and adder, the number of adder and multipliers will become independent of N. Thus, the leakage power contribution is expected to remain constant when N increases. On the other hand, the number of registers, register accesses and the control logic still scale up with N. How much this extra folding can improve overall power consumption needs further investigation.

6.2.2 Compile options

Not a lot of optimizations were made to the compile script used by the Synopsys Design Compiler. Several options exist that allow the compiler to further improve the design. One such option is 'compile_ultra', which was tried but in the end not used for this Thesis. It reduced the total power consumption of the system by 20%, down to $5.02\mu W$, when compiled with a HVT cell library.

The reasons for not including it was that the different components defined in the VHDL code could not be recognised anymore, thus making it unable to obtain results for specific components and compare them. Besides that, only this compile option was tried while there are likely more options available that were not looked at. Therefore this result was not deemed sufficient to include among the results in Chapter 5.

6.2.3 Arithmetic-level optimizations

In this thesis the optimizations on system and hardware implementation level were investigated, but the arithmetic level was not looked at. Looking deeper into the hardware, one may find the arithmetic can be improved upon. This involves implementing adders and multipliers at the bit level, manually specifying the dataflow through them. Certain application specific bit manipulations can sometimes be applied to these operating units that improve performance or power consumption.

One specific example of such a manipulation can be found in the FIR-filter. The first two filter coefficients are 4 and -1. When adding the products of these coefficients with the input, the three least significant bits (LSB) of the addition do not have to be calculated. The product of the input and a coefficient of 4 produces zeroes in these LSB positions. In some cases the Synopsys Design Compiler already found these improvements and applied them.

6.2.4 Word length analysis

As stated earlier, no extensive statistical analysis was applied to the inputs and outputs of the components to find optimal word lengths for them. The optimal word lengths between system components were determined in [14] for that implementation of the system, but this analysis could be extended to the word lengths within the components. It could help to improve the bit-error rate (BER) of the system or quantify the trade-off between performance and power consumption. Additionally, if a mathematical expression or algorithm can be formulated to calculate an optimal value for the various word lengths in the system, it can be used to easily find word lengths for different system configurations. As described in Chapter 4, the system was described in VHDL in such a way that the number of samples per symbol N can be specified and the required hardware will be generated accordingly. This property would be more useful with automatically generated word lengths.

Bibliography

[1] Mehdi Anteur et al. "Ultra Narrow Band Technique for Low Power Wide Area Communications". In: Dec. 2014, pp. 1–6. DOI: 10.1109/GLOCOM.2014.7417420.

[2] Hela Belhadj Amor and Carolynn Bernier. "Software-Hardware Co-Design of Multi-Standard Digital Baseband Processor for IoT". In: *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2019, pp. 646–649. DOI: 10.23919/DATE.2019.8714963.

[3] Yajing Chen et al. "A low power software-defined-radio baseband processor for the Internet of Things". In: *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 2016, pp. 40–51. DOI: 10.1109/HPCA.2016.7446052.

[4] David Flynn et al. *Low Power Methodology Manual*. Springer, Boston, MA, 2007. ISBN: 978-0-387-71819-4.

[5] E. Grayver and B. Daneshrad. "VLSI implementation of a 100-/spl mu/W multirate FSK receiver". In: *IEEE Journal of Solid-State Circuits* 36.11 (2001), pp. 1821–1828. DOI: 10.1109/4.962305.

[6] Fredric J. Harris. *Multirate Signal Processing for Communication Systems*. Prentice Hall PTR, 2004. ISBN: 0131465112.

[7] Mansi Jhamb, Garima, and Himanshu Lohani. "Design, implementation and performance comparison of multiplier topologies in power-delay space". In: *Engineering Science and Technology, an International Journal* 19.1 (2016), pp. 355–363. ISSN: 2215-0986. DOI: https://doi.org/10.1016/j.jestch.2015.08.006. URL: https://www.sciencedirect.com/science/article/pii/S2215098615001287.

[8] Donald E. Knuth. *The Art of Computer Programming volume 2: Seminumerical algorithms*. Addison-Wesley, 1988, pp. 519, 706.

[9] David Lachartre et al. "7.5 A TCXO-less 100Hz-minimum-bandwidth transceiver for ultra-narrow-band sub-GHz IoT cellular networks". In: *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 2017, pp. 134–135. DOI: 10.1109/ISSCC.2017.7870297.

[10] Kais Mekki et al. "A comparative study of LPWAN technologies for large-scale IoT deployment". In: 5 (Mar. 2019), pp. 1–7. DOI: 10.1016/j.icte.2017.12.005.

[11] N. Naik. "LPWAN Technologies for IoT Systems: Choice Between Ultra Narrow Band and Spread Spectrum". In: *2018 IEEE International Systems Engineering Symposium (ISSE)*. 2018, pp. 1–8. DOI: 10.1109/SysEng.2018.8544414.

[12] K. E. Nolan, W. Guibene, and M. Y. Kelly. "An evaluation of low power wide area network technologies for the Internet of Things". In: *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*. 2016, pp. 439–444. DOI: 10.1109/IWCMC.2016.7577098.

[13] Q. M. Qadir et al. "Low Power Wide Area Networks: A Survey of Enabling Technologies, Applications and Interoperability Needs". In: *IEEE Access* 6 (2018), pp. 77454–77473. DOI: 10.1109/ACCESS.2018.2883151.

[14] Victor van Rooij. "Low Power System Design of DDPSK Receiver". In: (2019).

[15] Siavash Safapourhajari. "Frequency Offset Tolerant Demodulators for UNB Communications". English. PhD thesis. Netherlands: University of Twente, Dec. 2020. ISBN: 978-90-365-5086-4. DOI: 10.3990/1.9789036550864.

[16] Siavash Safapourhajari and André Kokkeler. "Demodulation of double differential psk in presence of large frequency offset and wide filter". In: *IEEE 87th Vehicular Technology Conference* (2018).

[17] M. K. Simon and D. Divsalar. "On the implementation and performance of single and double differential detection schemes". In: *IEEE Transactions on Communications* 40.2 (1992), pp. 278–291. DOI: 10.1109/26.129190.

[18] P. P. Vaidyanathan. "Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial". In: *Proceedings of the IEEE* 78.1 (1990), pp. 56–93. DOI: 10.1109/5.52200.

[19] Mehmet Rasit Yuce et al. "SOI CMOS Implementation of a Multirate PSK Demodulator for Space Communications". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 54.2 (2007), pp. 420–431. DOI: 10.1109/TCSI.2006.885988.