Low-Latency GFSK Demodulation Architecture Comparison and Design for FPGA

Alexandre Lopez
IC-Design group
University of Twente
Enschede, The Netherlands

***Abstract*—Gaussian Frequency Shift Keying (GFSK) is a widely used type of continuous phase modulation (CPM), where a Gaussian pulse shaping filter is used before frequency modulation. Commercially available GFSK demodulators are mostly optimised to minimise power consumption or bit error rate (BER), but rarely consider the latency of demodulation. However, in situations where the output of the demodulator is used in a feedback loop, the demodulator's latency is a crucial factor regarding stability of the system. In this paper, we primarily investigate which noncoherent GFSK demodulator architecture, suitable for implementation on a field-programmable gate array (FPGA), has the lowest latency of demodulated bits, while considering BER performance secondary. Three demodulation methods were modelled in Simulink: instantaneous frequency estimation using zero-crossing detection, a quadricorrelator, and matched filters with envelope detection. Simulation results show the quadricorrelator demodulator is most suitable, introducing a latency of $1\,\mu\text{s}$ and a BER of 0.1% at $E_\text{b}/N_0 = 13.2\,\text{dB}$. The realistic model of this demodulator for FPGA produces a BER of 0.1% with $E_\text{b}/N_0 = 14.6\,\text{dB}$.**

***Index Terms*—Frequency shift keying, Low latency communication, Field programmable gate arrays, Simulink.**

I. Introduction

Since the first digital communications between devices in the 20th century, countless wireless communication standards have been defined. One such standard is Bluetooth Low Energy (BLE), using the Gaussian Frequency Shift Keying (GFSK) modulation scheme [1]. GFSK is a modulation type that filters the baseband message with a Gaussian filter, after which it is frequency modulated with continuous phase (CPM). This results in less required transmission bandwidth compared to regular frequency shift keying (FSK). Other than in BLE, GFSK is also used in many other standards, including Improved Layer 2 Protocol [2], DECT [3] and IEEE 802.15.4 [4]. This research is done in accordance with the BLE 1M PHY standard.

Currently available GFSK demodulators are primarily concerned with reducing BER and power consumption, since these are the most important attributes of a demodulator in most contexts. Ongoing research in the field of BLE at the Integrated Circuit Design group at the University of Twente, however, makes use of the demodulated message in a control loop, meaning low latency of the demodulator is of high importance regarding stability of the system. In fact, any control system where the demodulated message is used to give feedback on a system, would benefit from low demodulation latency. This research concerns itself with designing such a demodulator for BLE of PHY type 1M. In Bluetooth, receivers are predominantly noncoherent because of the cost efficiency associated with them [5], which is why the focus of this research is only on noncoherent demodulators.

This paper is structured as follows: in Section II-A, the GFSK modulation scheme is explained and modelled, after which the three demodulation architectures are analysed and designed in Section II-B. Simulations are done for the demodulators and results are then compared in Section III using the self-defined metric. In Section IV, a test setup is suggested, as well as a realistic implementation for FPGA. For this realistic implementation, a model is made in Simulink, followed by simulation results of a representative Simulink model in Section IV-D. Finally, the research is concluded in Section V with the implications of the results, as well as areas for further research on the topic.

II. Gaussian Frequency Shift Keying

A. Modulation

GFSK modulation is a modification of FSK, where the baseband message is first pulse-shaped (i.e. filtered) using a Gaussian filter before being frequency modulated (see Fig. 1). The filter removes sharp frequency transitions, significantly reducing spectral width of the transmitted signal, at the cost of ISI [5]. A comparison of their spectra is shown in Fig. 2.

The transmitted signal can be expressed as [6]:

$$s(t) = A\cos(2\pi f_\text{c} t + \phi(t)), \tag{1}$$

where

$$\phi(t) = 2\pi \cdot \frac{hR_\text{s}}{2}\int_0^t m(\tau)\,\mathrm{d}\tau. \tag{2}$$

Parameter f_c is the carrier frequency, h is the modulation index, R_s is the symbol data rate and $m(t)$ is the pulse shaped bit sequence.

The remainder of this research assumes the following parameters based on the BLE 1M PHY specification [1]:

- $R_\text{s} = 1\,\text{Mb/s}$,
- $h = 0.5$,
- The Gaussian filter's bandwidth-time product $\text{BT} = 0.5$.

Furthermore, it is assumed $s(t)$ is already down-mixed to $f_\text{c} = 1\,\text{MHz}$ upon reception at the demodulator.

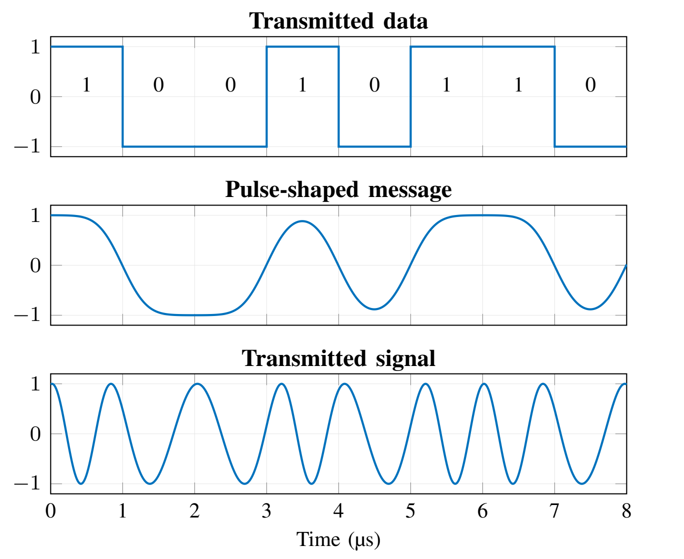

Fig. 1. Modulation stages from 1 Mb/s data to the transmitted GFSK signal with BT = 0.5, f_c = 1 MHz and modulation index $h = 0.5$.

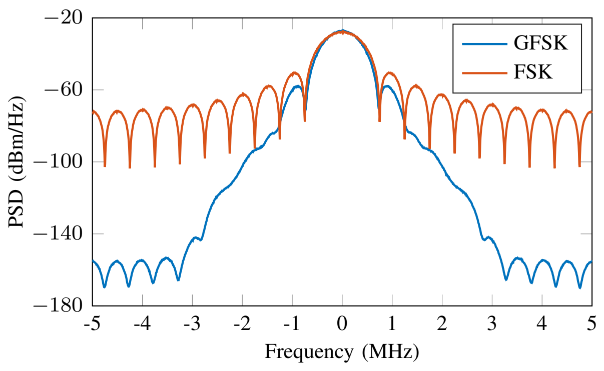

Fig. 2. Baseband frequency spectrum comparison between FSK and GFSK with a bit-rate of 1 Mb/s, a modulation index $h = 0.5$ and BT = 0.5.

A model of a demodulator is made in Simulink R2022b by modulating a Bernoulli bit sequence with the CPM block from the Wireless Communication toolbox (see Fig. 3) with a pulse width of three symbols and oversampling rate sps = 1000. The output of the CPM block is complex baseband, so it is then converted to passband with carrier frequency f_c. Fig. 4 shows that the CPM block also introduces a delay of 1 symbol. This delay is caused by the Gaussian filtering prior to modulation.

Next, the signal is downsampled to the sample frequency f_s of the demodulator, after which additive white Gaussian

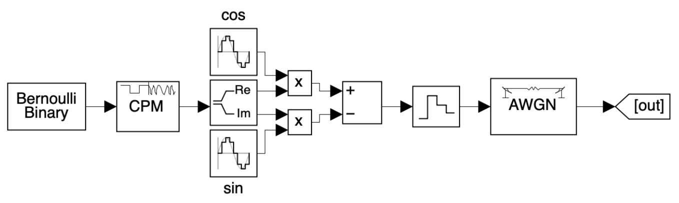

Fig. 3. Simulink model used for GFSK modulation.

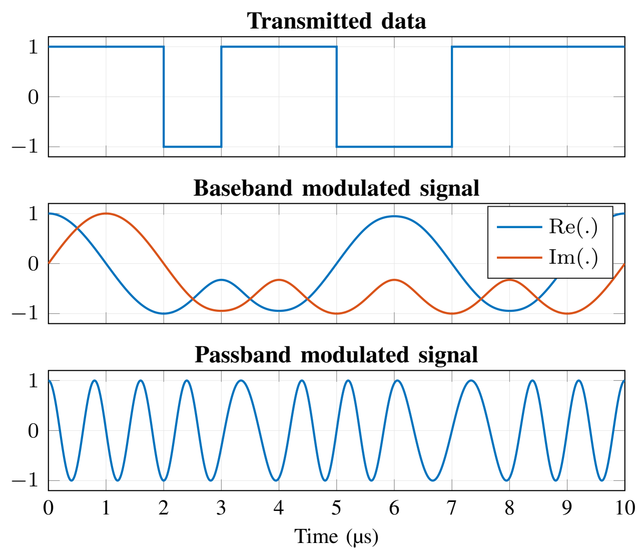

Fig. 4. Plot of the stages of modulation using the CPM block in Simulink.

noise (AWGN) is added to the signal. This way, no additional decimation filter is needed before downsampling to prevent the noise from aliasing. This workaround is needed because of the use of sps = 1000 in the first place, which is necessary to accurately represent a signal coming from an ideal transmitter.

B. Demodulation

For the demodulation of the passband modulated signal provided by the GFSK modulator, three noncoherent demodulator architectures are considered, after which their latencies and BER performances are then compared.

The total time between transmission of a bit and its receival consists of three parts: the 1 µs delay of the CPM block, the delay of the demodulator, and the delay from symbol slicing. The symbol slicer of all three approaches is done by sampling the output of the demodulator exactly in the middle of a symbol, adding a constant $\frac{1}{2R_s} = 0.5$ µs delay. The interested reader is referred to Appendix A for implementation details.

In this paper, we define the latency L of a demodulator as the time delay between the transmitted and detected bit, minus the constant 1.5 µs from the CPM block and symbol slicing.

The presented demodulators all have at least one parameter that trade latency for AWGN resilience. Therefore, multiple BER against E_b/N_0 simulations are done for the demodulators, each with different parameters. Additionally, the BER against E_b/N_0 is plotted for an ideal noncoherent FSK demodulator with the same modulation index, henceforth referred to as the reference demodulator. We use FSK as a lower bound to the BER against E_b/N_0 performance of GFSK, since it does not consider the ISI [5].

The maximum acceptable BER for BLE is 0.1% [1], which is achieved with $E_b/N_0 = 11$ dB for the reference demodulator. As a metric of noise resilience, we define

$$\Delta E_b \equiv \Big[E_b/N_0\Big]_{\mathrm{BER}=0.1\%} - 11\,\mathrm{dB}. \qquad (3)$$

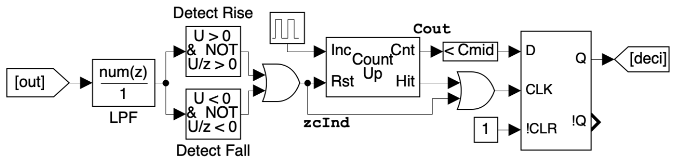

Fig. 5. Simulink model of the zero-crossing detection based GFSK demodulator.

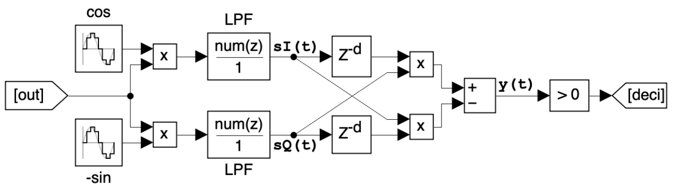

Fig. 6. Simulink model of the quadricorrelator GFSK demodulator.

This number represents how much less noise the demodulator under consideration can handle in the channel, compared to the reference demodulator. To compare the demodulators with their different parameters, we define a figure of merit as

$$\mathrm{FoM} \equiv \frac{1}{L^2 \cdot \Delta E_\mathrm{b}}, \tag{4}$$

representing a better demodulator when FoM is larger. For simplicity, L is in symbol periods (equivalent to 1 µs for $R_\mathrm{s} = 1\,\mathrm{MHz}$) and ΔE_b is in dB.

For fair comparison, each demodulator uses the same sample frequency $f_\mathrm{s} = 20\,\mathrm{MHz}$.

It must be noted this research only makes use of symmetric finite impulse response (FIR) filters because of their ease of construction in Matlab and because they have a constant group delay (of half their order) [7]. Constant group delay, however, is not required in most cases presented here, making this a topic for future research.

1) Zero-Crossing Detection Demodulator: The first demodulator, illustrated in Fig. 5, is based on estimating the instantaneous frequency of the passband signal, suggested by [8].

We do this by constructing a signal `zc_Ind` that indicates zero crossings of the GFSK signal by sending a pulse for one sample when either a rising edge or falling edge is detected from the received signal. This signal is connected to the reset of a counter `Cout` which increments count every $\frac{1}{f_\mathrm{s}}$, essentially timing the zero crossings. When the frequency of the received signal is exactly f_c, the value of `Cout` gets up to $C_\mathrm{mid} = \frac{f_\mathrm{s}}{2f_\mathrm{c}}$. Therefore, if the final value of `Cout` $> C_\mathrm{mid}$, it means a 0 was transmitted, and if `Cout` $< C_\mathrm{mid}$, a 1 was transmitted. Hence, `Cout` is compared to C_mid and the result (the estimated transmitted bit) is stored in a data flip-flop until the next zero crossing. This is achieved by using the zero crossing indication as a clock to the flip flop.

In the case a 0 is sent, however, it is not necessary to wait until the next zero crossing to update the flip flop; the moment the counter reaches C_mid, a hit pulse can be sent to the clock of the flip flop to send through the 0 immediately. This reduces ISI.

The latency of this part of the demodulator is the time between two zero crossings, since this is how long the demodulator needs to conclude if a 0 or 1 was sent. Therefore, the average latency of this part is $\frac{1}{2f_\mathrm{c}}$.

However, this method would be very sensitive to AWGN because this can introduce additional zero crossings, completely throwing off the counter. This is why a lowpass filter (LPF) precedes the zero-crossing detectors. The filter is of order N and has a cutoff frequency of 2 MHz, based on the spectral width seen in Fig. 2. The consequence hereof is that an additional latency of $\frac{N}{2}$ samples is introduced [7], making the total latency of the demodulator $\frac{1}{2} \cdot \left(\frac{1}{f_\mathrm{c}} + \frac{N}{f_\mathrm{s}}\right)$.

2) Quadricorrelator: The second presented method is the discrete-time quadricorrelator shown in Fig. 6, where the conventional differentiation elements are replaced by delay elements [9].

First, the passband GFSK signal is split and down-mixed into its in-phase and quadrature components

$$s_\mathrm{I}(t) = \mathrm{LPF}(\cos(2\pi f_\mathrm{c} t) \cdot s(t)) = \frac{A}{2}\cos(\phi(t)) \tag{5}$$

and

$$s_\mathrm{Q}(t) = \mathrm{LPF}(-\sin(2\pi f_\mathrm{c} t) \cdot s(t)) = \frac{A}{2}\sin(\phi(t)). \tag{6}$$

Next, these signals are delayed by ΔT, cross multiplied and subtracted from each other to produce

$$y(t) = s_\mathrm{Q}(t)s_\mathrm{I}(t-\Delta T) - s_\mathrm{I}(t)s_\mathrm{Q}(t-\Delta T) \tag{7}$$

$$= \frac{A^2}{4}\sin\big(\underbrace{\phi(t) - \phi(t-\Delta T)}_{\Delta\phi(t)}\big). \tag{8}$$

The argument of the sine, denoted $\Delta\phi(t)$ can be expressed as

$$\Delta\phi(t) = 2\pi\frac{hR_\mathrm{s}}{2}\left(\int_0^t m(\tau)\,\mathrm{d}\tau - \int_0^{t-\Delta T} m(\tau)\,\mathrm{d}\tau\right) \tag{9}$$

$$= 2\pi\frac{hR_\mathrm{s}}{2}\int_{t-\Delta T}^{t} m(\tau)\,\mathrm{d}\tau. \tag{10}$$

Ignoring the latency introduced by the image rejection filter of the mixers and scaling factors, this shows the output of the demodulator at time t is the sine of the area under $m(t)$ from $t - \Delta T$ until t. Assuming $-\frac{\pi}{2} \leq \Delta\phi(t) \leq \frac{\pi}{2}$ (which we will soon make sure of), the sine in (8) is a strictly increasing function through 0, meaning the sign of $y(t)$ is the same as the sign of the mentioned area. See Fig. 7 for a graphical relationship between $m(t)$ and $y(t)$.

To ensure $-\frac{\pi}{2} \leq \Delta\phi(t) \leq \frac{\pi}{2}$ we look at the maximum value $\Delta\phi(t)$ can reach and solve for ΔT accordingly. Equation (8) shows the maximum $\Delta\phi$ is achieved when $m(t) = 1$, so

$$2\pi\frac{hR_\mathrm{s}}{2}\int_{t-\Delta T}^{t} 1\,\mathrm{d}\tau \leq \frac{\pi}{2} \implies \Delta T \leq \frac{1}{2hR_\mathrm{s}}. \tag{11}$$

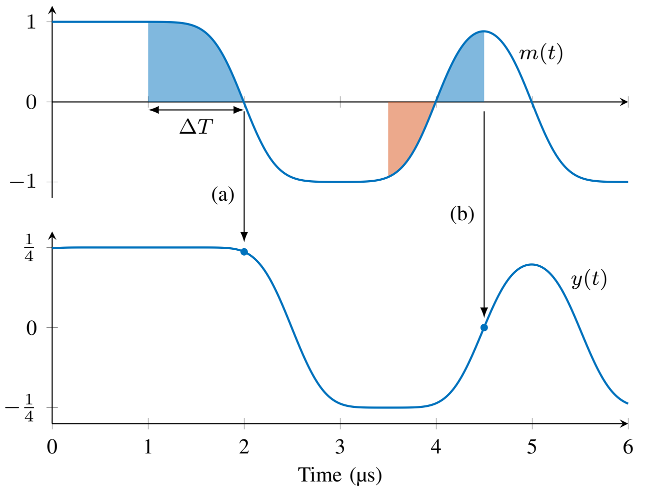

Fig. 7. Illustration of the output $y(t)$ of a quadricorrelator with $\Delta T = 1\,\mu s$. (a) shows that a positive area under $m(t)$ results in $y(t) > 0$. (b) shows that a net zero area under $m(t)$ results in $y(t) = 0$.

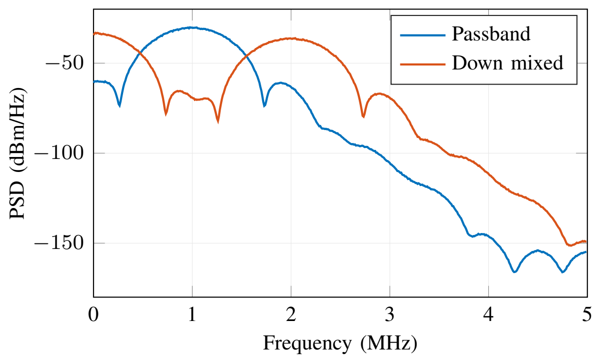

Fig. 8. Spectral analysis of passband GFSK signal before and after down mixing, before the image rejection filter.

When two unequal bits are transmitted, it takes approximately $\frac{\Delta T}{2}$ for the area under the new bit to make up for the area of opposite sign of the old bit due to the close symmetry around the zero crossings of $m(t)$ (see Fig. 7b). This means the latency of this part of the demodulator is $\frac{\Delta T}{2}$. It must be noted this is not exact; there is slight asymmetry around the zero crossing of $m(t)$ caused by the history of the preceding symbol. However, for a BT product of 0.5, this deviation is less than 3 ns, which is insignificant compared to the symbol time of 1 µs.

Another source of delay is due to the image rejection filter. In Fig. 8 we see the cutoff frequency of the filter must be at f_c to reject the unwanted image. Notice no meaningful pass- and stopband frequencies can be set because the unwanted image is already interfering with the baseband image. The LPF is created using the Matlab function `designfilt`, specifying filter order N and normalised cutoff frequency $\frac{f_c}{f_s/2}$.

The total latency of this demodulator therefore is $\frac{\Delta T + N/f_s}{2}$. The effects of N and ΔT on the BER are independently simulated.

3) Matched Filter Demodulator: The last demodulator is shown in Fig. 9. Here, $s(t)$ is duplicated and passed through

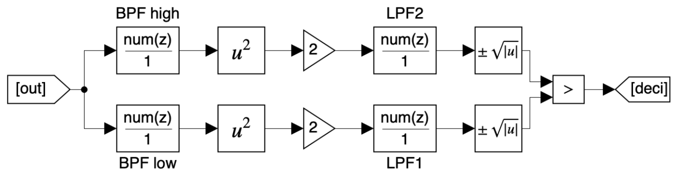

Fig. 9. Simulink model of the matched filter GFSK demodulator.

two different filters, approximating matched filters; one filter has a maximum output when 0 is transmitted and the other one when a 1 is transmitted. Next, envelope detection is applied and the outputs are compared to decide if a 0 or 1 was sent.

The filters used to discriminate the frequencies are bandpass filters centered at $f_c \pm \Delta f$, where $\Delta f = \frac{hR_s}{2}$ is the frequency deviation of modulation. The width of the passband is set to 1 MHz. The result is two signals, roughly centered at $f_c \pm \Delta f$.

After this, we use a square-law AM demodulator as an envelope detector. Squaring the signals results in two terms: the desired envelope, and an unwanted copy, centered at $2(f_c \pm \Delta f)$ [10]. Hence, the squaring blocks are followed by LPFs with cutoff frequencies $f_c \pm \Delta f$ and using a rectangular window for a sharper transition. The order of this filter shall be kept at 14, since results in Appendix B show the performance gets worse with both a higher order or a lower order.

The only parameter of the demodulator therefore is bandpass filter order N, resulting in a latency of $\frac{N+14}{2f_s}$.

III. Simulation Results

A plot of BER against E_b/N_0 plot is made for each demodulator architecture (see Appendix B), from which values for ΔE_b are found and FoM is calculated. The resulting summary is given in Table I. The demodulator with the highest FoM is the quadricorrelator with a delay block of 1 µs and filter order $N = 20$, producing a latency of 1 µs, and requires 2.2 dB more E_b/N_0 than the reference demodulator at a BER of 0.1%.

The fastest demodulator is the zero crossing detector with a second order filter, and the best demodulator regarding BER vs E_b/N_0 performance is the matched filter kind with filters of order 80. We are, however, only interested in the demodulator with the highest FoM.

It must be noted that only a limited number of parameter combinations were simulated due to time constraints, making the optimal solution unlikely to be the one presented here. However, the square in (4) was mostly chosen arbitrarily in the first place anyway, so this is of no serious concern.

IV. FPGA Implementation

This section covers the generation of the GFSK wave, followed by a realistic (quantised) Simulink model of the demodulator. Finally, VHDL suggestions are given for final implementation. The list of components used is given in Table II.

TABLE I
COMPARISON BETWEEN DEMODULATION ARCHITECTURES

Architecture	Parameter(s)	ΔE_{b} [dB]	L [µs]	FoM
Z.C. detector (N)	2	11.2	0.55	0.295
	10	5.6	0.75	0.317
	60	4.8	2.00	0.052
Quadricorr. $(\Delta T, N)$	0.1 50	5.8	1.30	0.102
	0.5 50	4.2	1.50	0.106
	1 50	1.6	1.75	0.204
	1 20	2.2	1.00	0.455
	1 10	4.0	0.75	0.444
M.F. N	30	3.6	1.10	0.230
	50	2.4	1.60	0.163
	80	1.4	2.35	0.129

TABLE II
LIST OF COMPONENTS

Device	Manufacturer	Model	Serial number
Signal gen	Keysight	33622A	MY59003394
FPGA	Terasic	SoCKit rev. C	
ADC	Terasic	THBD-ADA	
Scope	Keysight	DSOX2002A	MY56273381

A. Signal Generation

In order to test the demodulator once implemented on FPGA, a GFSK wave must be generated with a known bit sequence, starting with a flag that the FPGA can easily recognise.

The signal generator can loop an arbitrary signal (from a file) of maximum length 4×10^6 samples and with maximum sample frequency 250 MHz [11]. The amplitude is restricted to 1.5 $\mathrm{V_{pp}}$ because this is the input range of the analog to digital converter (ADC). Creating a GFSK wave with the aforementioned properties results in the signal shown in Fig. 10. The amplitude seems off by a factor 2, but this is because of the high Z of the oscilloscope; the signal generator accounts for the 50 Ω input impedance of the ADC. The exact Matlab script to generate the `.arb` file is given in Appendix C.

Although not done in this research, if BER performance were to be simulated, care must be taken to ensure the AWGN is band-limited to $f_{\mathrm{s}}/2$. The simplest way to ensure this in Matlab is to add the white Gaussian noise, and then filter the signal with a very high order LPF. Alternatively, an analog filter could also be used.

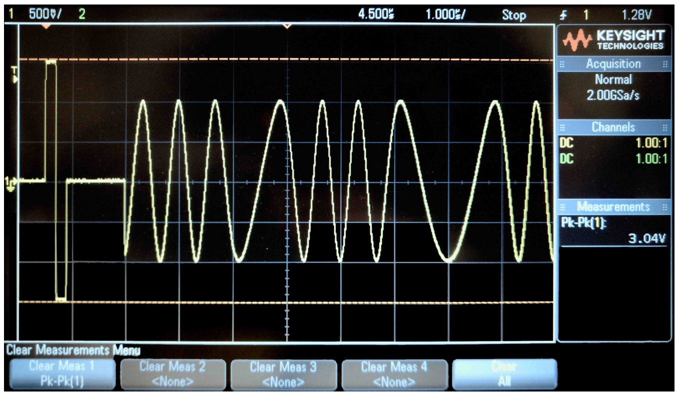

Fig. 10. Oscilloscope aquisition of output from GFSK wave generation.

B. Optimisation for FPGA

A modification that simplifies implementation on FPGA is the reduction of the local oscillator (LO) to a 1 bit resolution instead of a full-scale sine wave. This way, a lookup table can be omitted, at the cost of harmonic distortion. The Fourier series of a 1 bit sine wave of frequency f_{c} is

$$\sin_{\mathrm{sq}}(t) = \frac{4}{\pi} \sum_{\substack{n \text{ odd} \\ n \geq 1}} \frac{1}{n} \sin(n \cdot 2\pi f_{\mathrm{c}} t). \tag{12}$$

Using (12) and (1), the output of the mixer can be expressed as

$$\begin{aligned} s(t) \sin_{\mathrm{sq}}(t) = \sum_{\substack{n \text{ odd} \\ n \geq 1}} \frac{4A}{\pi n} \cdot \frac{1}{2} \Big[&\sin\big((n+1) \cdot 2\pi f_{\mathrm{c}} t + \phi(t)\big) \\ &+ \sin\big((n-1) \cdot 2\pi f_{\mathrm{c}} t - \phi(t)\big)\Big]. \end{aligned} \tag{13}$$

For $n = 1$, this is same as the output of a mixer using a full scale sine wave, scaled by a factor $4/\pi$. The rest of the terms are distortions, centered at frequencies $2f_{\mathrm{c}}$, $4f_{\mathrm{c}}$, $6f_{\mathrm{c}}$, etc. This is, however, insignificant, because the terms in the sum decay by themselves, as well as get filtered by the mixer's image rejection LPF with cutoff frequency f_{c}.

The second simplification regards the LPFs. The Cyclone V on this FPGA board has a dedicated DSP chip which can only be configured to have at most two parallel 18 bit multipliers that run at a maximum frequency of 200 MHz [12]. Using a filter order 20 and a sample frequency of 20 MHz, neither a parallel nor a serial FIR filter can be implemented; a parallel implementation would require $2 \cdot 21 = 40$ multipliers and a serial implementation would require a speed of $21 \cdot 20 = 420$ MHz. Potential solutions include a reduction in sample frequency, as well as using a lower order filter.

However, a different approach is used in this case: all filter coefficients are changed to 1. The result is a scaled moving average filter, which is also a type of LPF. Fig. 12 shows the frequency responses of the original filter compared to the new moving average filter.

C. Implementation Suggestions

To simulate the performance of the realistic demodulator, these two modifications are made in Simulink and the receiving input is quantised. The realistic model of the realisable demodulator is shown in Fig. 11.

The ADC board used with the FPGA has a 14 bit offset binary output [13], so the model's modulator output is mapped and rounded from 0 to $2^{14} - 1$ in the ADC block. For the multiplication and filtering to work, the DC offset is then removed by temporarily turning the bus into 15 bits, subtracting the constant $\left\lfloor \frac{2^{14}-1}{2} \right\rfloor$, and then truncation by 1 bit, while keeping the sign bit.

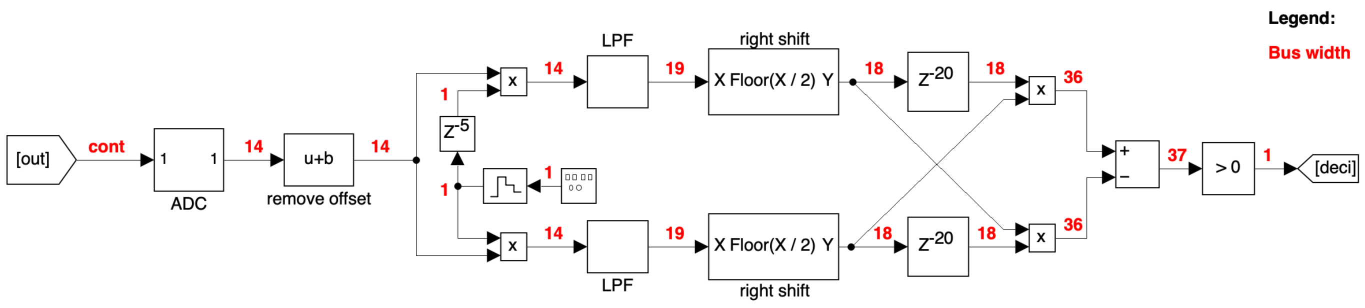

Fig. 11. Simulink model of the realistic quadricorrelator GFSK demodulator, simplified for implementation on FPGA.

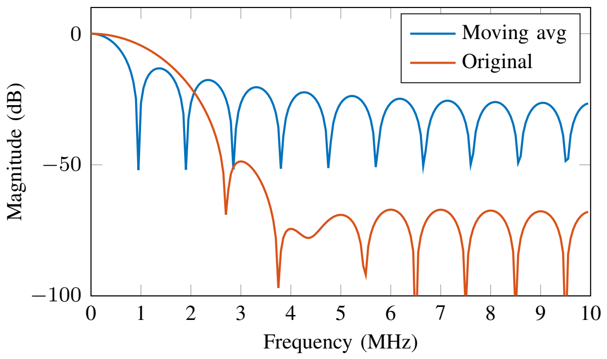

Fig. 12. Frequency response of a 20th order moving average filter compared to the order 20 filter designed by to reject the mixing image.

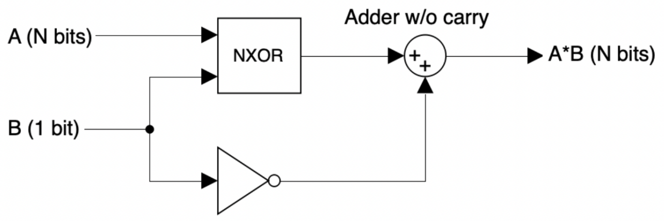

Fig. 13. Diagram of a N by 1 bit multiplier.

The 1 MHz LOs can be made by taking the existing 20 MHz clock used for the ADC, and connecting it to a counter that ranges from 0 to 19. From this, the $-\sin_{\text{sq}}(2\pi f_c t)$ is created by creating a signal that becomes 1 when the counter hits 0, and becomes 0 when the counter hits 10. Similarly, the $\cos_{\text{sq}}(2\pi f_c t)$ is created by creating a signal that becomes 1 when the counter hits 5, and becomes 0 when the counter hits 15.

Mixing the signal with the 1 bit LO is efficiently done by applying the appropriate bit operation rather than using a full multiplier. Signed data is stored in two's complement by the `IEEE.numeric_std` library, so multiplication by -1, is done by inverting every bit and adding one. The operation of inverting the signal in case LO is 0 and leaving it unchanged if LO is 1, is equivalent to a XNOR gate. Next, 1 must be added to the signal in case LO is 0, and 0 must be added in case LO is 1. This is achieved by negating the LO and adding it to the output. The diagram of the multiplier is shown in Fig. 13.

Next, the LPF is made by placing 20 data flipflops in series with adders summing the individual stages. The corresponding block diagram is shown in Fig. 15. The additional bus width required is $\lceil \log_2(21) \rceil = 5$, making the output width of the filter 19 bits.

Next, the signals are delayed by 20 by putting 20 data flipflops in series. Multiplication is done with the dedicated 18 by 18 bit multipliers available. The result is 36 bits wide. Subtraction can be achieved by constant multiplication by -1 and an adder. The output after this stage is 37 bits, from which only the MSB is used, since this is the sign bit. This bit is considered the output of the demodulator. The full model is shown in Fig. 11.

At this stage symbol slicing is performed. As can be seen in Fig. 10, the wave is generated such that the middle of the pulses fall on multiples of 1 µs since the starting flag. This starting flag can be turned into a signal by comparing the ADC output to $\frac{5}{6}(2^{14}-1)$. If a data flipflop samples the output of the demodulator with period $\frac{1}{R_s} = 1\,\mu\text{s}$, starting at the rising edge of the starting flag, then the output of the demodulator is sampled exactly in the middle of each symbol. This bit sequence can then be stored in random access memory (RAM), where also the known bit sequence is stored. A BER calculation can then be made and displayed using the LCD screen on the board.

Note it is very important for the clocks of the FPGA and the signal generator to be synchronised. This can be achieved by either providing the signal generator with a square wave generated from the FPGA by using the DAC, or by providing the FPGA with the clock from the signal generator using the second ADC channel.

D. Realistic Simulation

The transient analysis shown in Fig. 14 gives an overall idea of the stages of demodulation of the quadricorrelator. From it, we see the ripple from the intermediate signal due to the square local oscillator. In Fig. 16, the realistic demodulator does not perform much worse than the original model it was

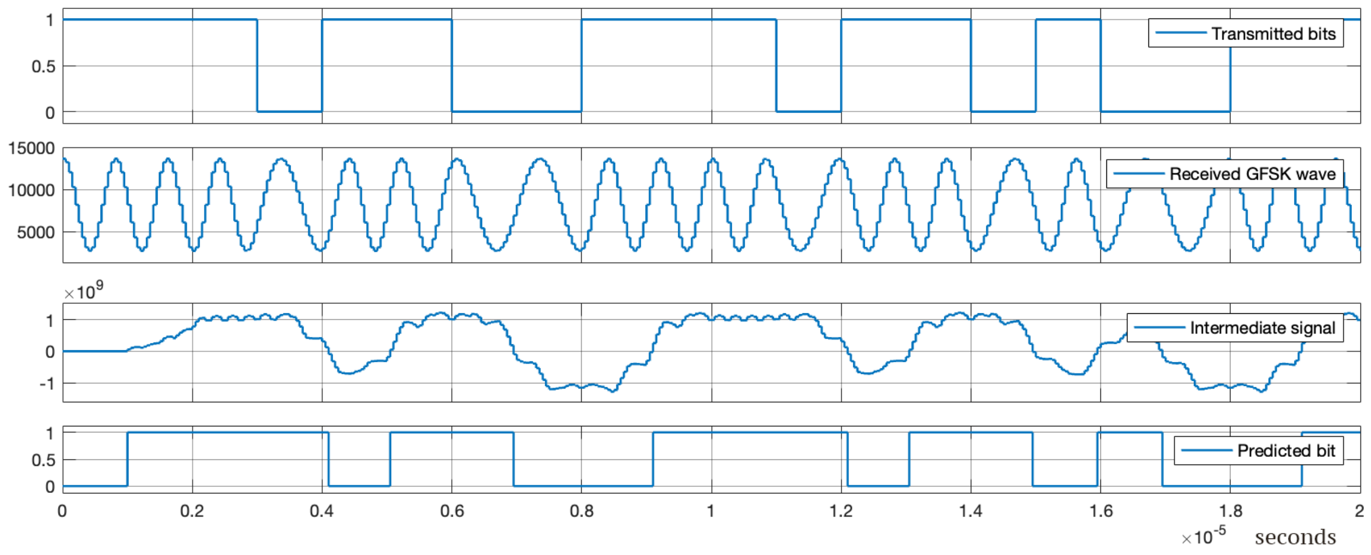

Fig. 14. Simulation result of the Simulink model of the realistic quadricorrelator GFSK demodulator, simplified for implementation on FPGA.

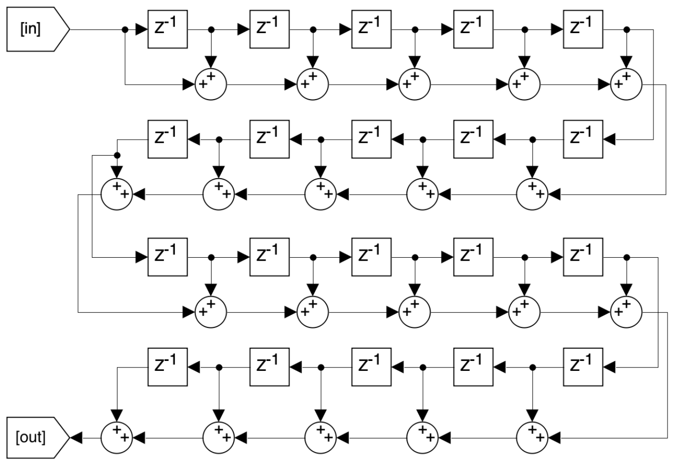

Fig. 15. Scaled moving average filter of order 20.

derived from. The latency remains 1 µs and $E_\mathrm{b}/N_0 = 3.6\,\mathrm{dB}$, resulting in FoM $= 0.28$. This worsening is partially caused by quantisation (modeled by rounding of integers), the introduced harmonics from the square wave, as well as the simplified LPFs.

Although the FoM is lower than the original 0.44, this is still high in comparison to most demodulators listed in Table I.

V. Conclusion

This research presented three non-coherent demodulation architectures for GFSK with parameters in accordance with the BLE 1M PHY standard. The considered architectures were a zero crossing detector, a quadricorrelator and matched filters with envelope detection. Each demodulator was analysed, based on which multiple designs were made for every

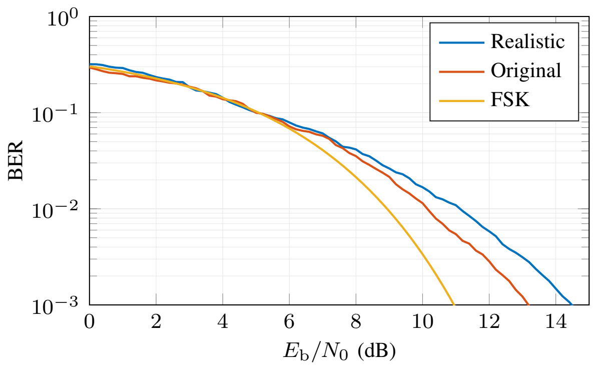

Fig. 16. BER against E_b/N_0 for the realistic Simulink model of the quadricorrelator, compared to the model from which it descended and the reference FSK demodulator.

demodulator and the models were simulated for varying levels of AWGN.

With the primary objective of designing a low-latency demodulator, which still has a good AWGN resilience, a FoM was defined for comparison. Results show the quadricorrelator with a delay $\Delta T = 1\,\mu\mathrm{s}$ and filter order $N = 20$ presents the highest FoM.

Next, a test setup, as well as specific FPGA implementation ideas were suggested, based on which a realistic version of the aforementioned best performing demodulator was made. Realistic implementation did reduce the FoM, but still remained amongst the best ones, suggesting it is a suitable demodulator.

However, further research should be done into the design of more appropriate filters, namely by considering IIR filters, or by constructing FIR filters differently. This research also only considered AWGN as a disturbance, but the demodulators might react differently to different disturbances, such as frequency drift, DC offset, jitter, etc. Additionally, changing the sample frequency might also have effects not observed in this paper. This research also only investigated the architectures for GFSK parameters in accordance to the BLE 1M PHY standard. FoM results may vastly differ for GFSK with different parameters. Finally, only a few versions of each demodulator were simulated, meaning there might still be (combinations of) parameters that produce better results than seen here.

Acknowledgment

I am very grateful for the helpful guidance, invaluable insights, and feedback provided by my supervisors Maryam Dodangeh and Mark Oude Alink; their help has been a major contribution to this research.

I also would like to thank my friends Maurits and Iustin for keeping me company in the ICD lab long after everyone else had left to go home, as well as my housemate Ali, with whom I routinely shared my fluctuating emotions I felt for FPGAs.

References

[1] Bluetooth SIG, *Bluetooth core specification v5.4*, vol. 6: Low energy controller, Jan. 2023. [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification-5-4/.

[2] N. Carrillo, *Improved layer 2 protocol specification draft v0.5*, KK4HEJ, Jun. 2022.

[3] ETSI, *Digital enhanced cordless telecommunications; common interface;* ETSI EN 300 175-2 v2.9.1, Part 2: Physical Layer (PHL), Mar. 2022.

[4] IEEE Computer Society, *IEEE standard for low-rate wireless networks*, 802.15.4, May 2020.

[5] M. Valenti, M. Robert, and J. Reed, "On the throughput of Bluetooth data transmissions," in *2002 IEEE Wireless Communications and Networking Conference Record. WCNC 2002 (Cat. No.02TH8609)*, vol. 1, 2002, 119–123 vol.1. DOI: 10.1109/WCNC.2002.993475.

[6] S. Faruque, "Radio frequency modulation made easy," *SpringerBriefs in Electrical and Computer Engineering*, 2017. DOI: 10.1007/978-3-319-41202-3.

[7] X. Lai, Y. Zuo, Y. Guo, and D. Peng, "A complex-error and phase-error constrained least-squares design of FIR filters with reduced group delay error," in *2009 Chinese Control and Decision Conference*, 2009, pp. 1967–1972. DOI: 10.1109/CCDC.2009.5191660.

[8] K. N. Sankar, A. Srivastava, B. Chatterjee, K. K. Rakesh, and M. S. Baghini, "FSK demodulator and FPGA based BER measurement system for low IF receivers," in *2016 20th International Symposium on VLSI Design and Test (VDAT)*, 2016, pp. 1–2. DOI: 10.1109/ISVDAT.2016.8064901.

[9] T.-C. Lee and C.-C. Chen, "A mixed-signal GFSK demodulator for Bluetooth," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 3, pp. 197–201, 2006. DOI: 10.1109/TCSII.2005.858320.

[10] S. A. Tretter, *Communication System Design Using DSP Algorithms*. Springer New York, NY, Jan. 2008, ISBN: 978-0-387-74885-6. DOI: https://doi.org/10.1007/978-0-387-74886-3.

[11] Keysight Technologies, *Keysight Trueform series waveform generator*, 33500-90901.

[12] Intel, *Cyclone V device datasheet*, 683801, Nov. 2019.

[13] Analog Devices, *Dual AD converter AD9248*, Rev. A.

Appendix A
Simulink models

For every model it is required to run the following code to load the necessary variables into workspace:

```
Rs  = 1e6;  % symbol rate (= bit rate)
BT  = 0.5;  % bandwidth-time product
Tw  = 3;    % FIR filter length/symbol length
h   = 0.5;  % modulation index
fc  = 1e6;  % carrier frequency
sps = 1000; % samples per symbol
fs  = 20e6; % sample frequency of demodulators
```

A. *Modulator*

The exact parameters of the blocks used for modulation are listed below:

- Bernoulli Binary Generator
  - Pr(0): 0.5;
  - Initial seed: 0
  - Sample time: `1/Rs`
  - Samples per frame: 1
  - Output data type: boolean
- CPM Modulator Baseband
  - M-ary number: 2
  - Input type: Bit
  - Symbol set ordering: Binary
  - Modulation index: `h`
  - Frequency pulse shape: Gaussian
  - BT product: `BT`
  - Pulse length: `Tw`
  - Symbol prehistory: 1
  - Phase offset: 0;
  - Samples per symbol: `sps`
  - Rate options: Allow multirate processing
  - Output data type: double
- cos and sin
  - Sine type: time based
  - Time (t): Use simulation time
  - Amplitude: 1
  - Bias: 0
  - Frequency (rad/sec): `2*pi*fc`
  - Phase: `0` for cos, `pi/2` for sin
  - Sample time: `1/(Rs*sps)`
- Zero-Order Hold
  - Sample time: `1/fs`
- AWGN Channel
  - Initial seed: 67
  - Mode: Signal to noise ratio (Eb/No)
  - Eb/No (dB): `EbN0`
  - Number of bits per symbol: 1
  - Input signal power: `1/2`
  - Symbol period: `1/Rs`

B. *Zero-Crossing Demodulator*

The following code needs to be run first, where the value for `N` is not strict.

```
Cmid = fs/(2*fc);
N = 10;
filtZC = designfilt('lowpassfir', ...
    'FilterOrder',N, ...
    'CutoffFrequency',2e6/(fs/2));
```

The blocks in the Simulink model have the following parameters:

- Discrete FIR Filter
  - Coefficients: `filtZC.Coefficients`
- Pulse Generator:
  - Pulse type: Sample based
  - Time: Use simulation time
  - Amplitude: 1
  - Period: 2
  - Pulse width: 1
  - Phase delay: 0
  - Sample time: `1/fs`
- Counter
  - Count direction: Up
  - Count even: Either edge
  - Counter size: User defined
  - Maximum count: `Cmid*2`
  - Initial count: 0
  - Output: Count and Hit
  - Hit values: `Cmid + 1`
  - Reset input: Check
  - Count data type: double
  - Hit data type: Logical

C. *Quadricorrelator*

The following code needs to be run first, where the values for `DT` and `N` are not strict.

```
DT = 2;
N =  50;
filtDM = designfilt('lowpassfir', ...
    'FilterOrder',N, ...
    'CutoffFrequency',1e6/(fs/2));
```

The blocks in the Simulink model have the following parameters:

- Sine Wave
  - Sine type: time based
  - Time (t): Use simulation time
  - Amplitude: 1
  - Bias: 0
  - Frequency (rad/sec): `2*pi*fc`
  - Phase: `pi/2`
  - Sample time: `1/fs`
- Discrete FIR Filter
  - Filter structure: Direct form,
  - Coefficients: `filtDM.Coefficients`
- Delay
  - Delay length: `DT`

D. Matched Filter

The following code needs to be run first, where the values for `DT` and `N` are not strict.

```
N1 = 50;
N2 = 14;
hMF1 = designfilt('bandpassfir', ...
    'FilterOrder',N1, ...
     'CutoffFrequency1',(0.75-0.5)*1e6/(fs/2), ...
     'CutoffFrequency2',(0.75+0.5)*1e6/(fs/2));
hMF2 = designfilt('bandpassfir', ...
    'FilterOrder',N1, ...
     'CutoffFrequency1',(1.25-0.5)*1e6/(fs/2), ...
     'CutoffFrequency2',(1.25+0.5)*1e6/(fs/2));
hLP1 = designfilt('lowpassfir', ...
    'FilterOrder',N2, ...
    'CutoffFrequency',0.75e6/(fs/2), ...
    'Window','rectwin');
hLP2 = designfilt('lowpassfir', ...
    'FilterOrder',N2, ...
    'CutoffFrequency',1.25e6/(fs/2), ...
    'Window','rectwin');
```

The blocks in the Simulink model have the following parameters:

- Discrete FIR Filter
  - Coefficients: `<X>.Coefficients`, where `<X>` is `hMF1` for the low bandpass filter, `hMF2` for the high bandpass, `hLP1` for the low lowpass, and `hLP1` for the high lowpass.

E. Symbol slicing and BER computation

First, the transmitted bits are oversampled with f_s and delayed by the exact amount of samples that correspond to the demodulator's latency. Next, we need to resample exactly in the middle of the bits. This is done in Simulink by delaying the oversampled bitstream such that the middle of a bit falls exactly on an integer multiple of the sample time $\frac{1}{R_s}$. For $f_s = 20\,\text{MHz}$ and $R_s = 1\,\text{MHz}$, the middle of a symbol is after 10 samples, and the entire symbol is 20 samples. So by adding a delay of

$$\texttt{mod(10-<latency in samples>,20)}, \qquad (14)$$

the bit stream is delayed by the smallest amount such that the middle of the bits fall exactly on multiples of $1\,\mu\text{s}$. Then, a zero order hold block samples the bit stream at these instances and the signal is given to a BER calculation block.

APPENDIX B
SIMULATED BER PLOTS

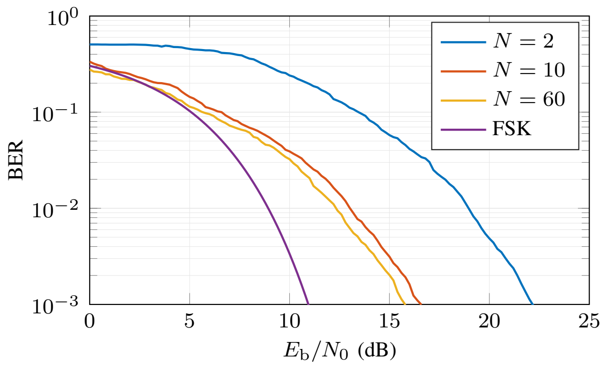

Fig. 17. BER vs E_b/N_0 plot of the zero-crossing based demodulator for varying filter order N.

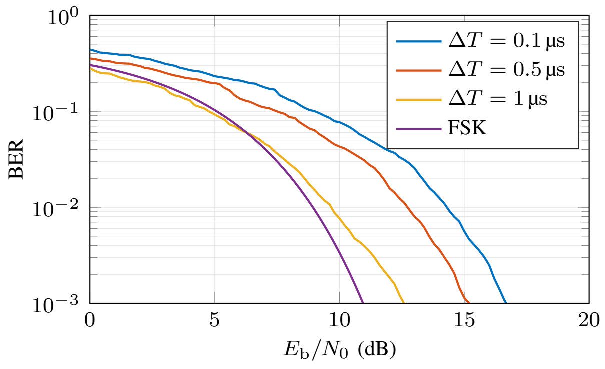

Fig. 18. BER vs E_b/N_0 plot of the quadricorrelator with constant $N = 50$ demodulator for varying ΔT.

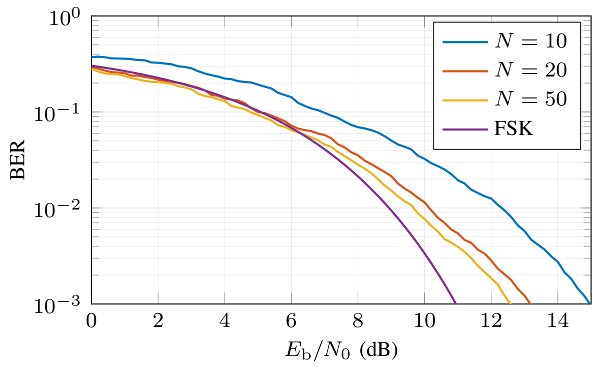

Fig. 19. BER vs E_b/N_0 plot of the quadricorrelator with constant $\Delta T = 1\,\mu\text{s}$ for varying filter order N.

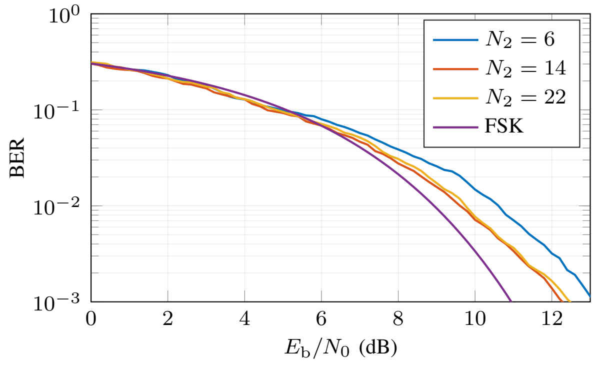

Fig. 20. BER vs E_b/N_0 plot of the matched filter demodulator with constant BPF order $N_1 = 80$ and varying LPF order N_2.

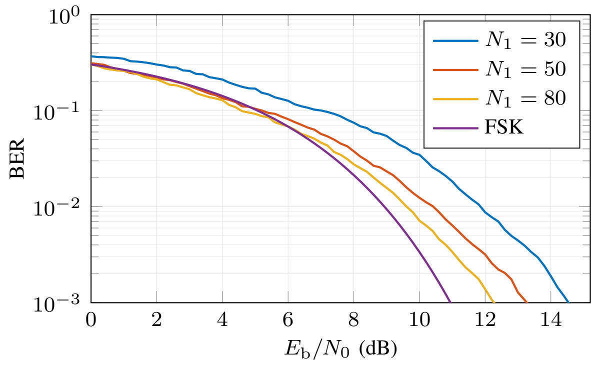

Fig. 21. BER vs E_b/N_0 plot of the matched filter demodulator with constant LPF order $N_2 = 14$ and varying BPF order N_1.

Appendix C
Waveform Generation

The follwing code was used to generate the ARB file that was manually transferred to storage of the wave generator through USB.

```
% %% generate GFSK for KEYSIGHT 33600A
% set params
fs = 250e6;      % max Sa/s for this device
length = 4e6;    % max Sa for this device
Rs   = 1e6;      % symbol rate (= bit rate)
BT   = 0.5;      % bandwidth-time product
Tw   = 3;        % FIR filter length
fc = 1e6;        % Carrier frequecy
Df   = 250e3;    % frequency deviation

% therefore
sps = fs/Rs;                 % samples per symbol
Nsyms = length/sps - Tw;  % -Tw due to filtering

% generate bits
rng(0)
bits = (randi(2,[Nsys,1]) - 1)*2-1;
msg = repelem(bits,sps);

% GFSK
msg_filtered = conv(gaussdesign(BT,Tw,sps),msg);
y = fmmod(msg_filtered,fc,fs,2*Df);

% add beginning mark
y(1:50) = 1.5;
y(51:100) = -1.5;

% remove bleed from filters abruptly
y(101:sps*Tw/2) = 0;
y(end - sps*Tw/2:end) = 0;

% normalise
y = y/1.5;

% make ARB file
header = "File Format:1.10\n"     + ...
         "Channel Count:1\n"      + ...
         "Sample Rate:250E6\n"    + ...
         "High Level:0.75\n"      + ...
         "Low Level:-0.75\n"      + ...
         "Data Type:'short'\n"    + ...
         "Data:\n";
dataspec =  "%d\n";

fileID = fopen("GFSK.arb","w");
fprintf(fileID,header);
fprintf(fileID,dataspec,round(32767*y));
fclose(fileID);

%%% remove last \n manually using text editor,
%%% else insuficient memory prompt!!!
```