An Area Efficient GMSK Demodulator

P&D Electronics and Chip Design - Report

Group 7: Øyvind M. Bjærum, Davide H. Fabbro, Simon L. Pedersen, Jakov Šola

***Abstract*—This report proposes a very low area implementation of a Gaussian Minimum Shift Keying demodulator. The analog front end consists of a variable gain amplifier (VGA) with 8 different gain levels and a 4 bit flash analog-to-digital converter (ADC). The amplified signal is converted to a digital code by the ADC, so it can be processed in the digital domain. The used mixer is a completely digital implementation using only a sorter and interpolator. Phase detection is done using a Coordinate Rotation DIgital Computer (CORDIC) engine. A moving average filter prepares the non-return-to-zero signal for Varicode decoding. VGA control is done by checking for signal clipping and accordingly adjusting the gain. Frequency adjustment is done with a simple calibration flow. All of these steps only use shifting and adding operations that are efficient in the digital domain. The symbol error rate was calculated on the test signals provided to us demodulated on a Spartan 3A FPGA.**

***Index Terms*—GMSK demodulation, VGA, ADC, Digital Mixer, CORDIC, Decoder, DSP**

I. Introduction

GAUSSIAN Minimal Shift Keying (GMSK) is a modulation technique that is very popular due to the fact that it has very low power side-bands and thus prevents interference between adjacent channels. [1]. In this project we were tasked with developing a platform that demodulates a $100bit/s$ bitstream modulated at a $20kHz \pm 0.5kHz$ carrier. This included developing an analog front end that amplifies the signal, an analog-to-digital converter (ADC) and a digital platform that analyses the signal and finally demodulates it.

While examining existing and popular implementations of such a platform we noticed that many perform the mixing stage either in the analog domain or in the digital but still using the same mindset of how an analog mixer works - signal multiplication that can take up a lot of area. Seeing the lack of data about low area implementations, we decided to take up this challenge. Such a platform that we created, with some improvements, can be very valuable in low power and low speed applications.

II. Division of Work

TABLE I
Division of work

Student	Domain	Modules
Øyvind	Analog	ADC, VGA, analog parts integration
Simon	Analog	VGA, advanced clock recovery attempt, analog parts integration
Jakov	Digital	Mixer, CORDIC, $d\theta/dt$, NRZ LPF, Varicode decoder, clock recovery, VGA control, frequency initialization
Davide	Digital	Alternative mixer: Mixer, NCO, CIC; Support with other modules

III. High Level Design

Before starting the project we needed to make some high level decisions that would determine the course of development. The leading idea was to try and develop something that was unique. After all, this is how good management decisions would be made - there is no purpose in spending time developing something that is exactly the same as many other competitors' implementation. Since many of the specifications were set, we needed to make a trade-off between performance and area/power.

Seeing that many other groups were aiming for maximum performance, we decided to go the other way and develop a solution that would be as minimal as possible - probably sacrificing the possibility of decoding signals with low signal-to-noise ratio (SNR).

The cornerstone of such a design was trying to minimize the number of operations that are difficult to perform in the digital domain such as multiplication and division, ideally not using any. One way of doing this was using a fully digital mixer [2], eliminating the need for a mixer, numerically controlled oscillator (NCO), and a cascaded integrator-comb (CIC) low-pass filter (LPF) and downsampler. It would have been a big risk to only try this method as it is not as widely documented as the traditional one. [1] This is why we first started developing both in parallel, but later on abandoning the traditional "analog imitating" approach when the minimal area one showed results. Naturally, both variants were shown to be feasible in our high level Matlab model with $BER = 0$ at $SNR > 50\,\mathrm{dB}$. Details about the "traditional" attempt can be found in Appendix A.

It was determined early that due to the strict time frame of the project that it would not be manageable to make a successive-approximation-register (SAR) ADC. The flash topology was then chosen, with the goal of making it as small as possible.

The final high level block diagram is seen on Fig. 1

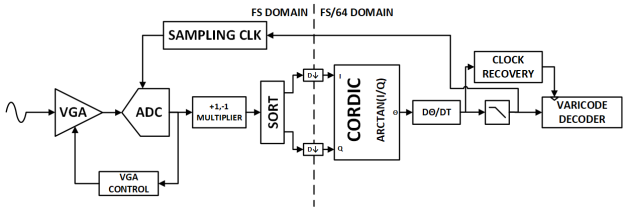

Fig. 1. High level block diagram.

IV. Analog Design

The analog design consists of two parts, a variable gain amplifier (VGA), and an analog to digital converter (ADC).

The topologies of the circuits were determined early with emphasis on making a design work, then trying to integrate the parts and optimize further. The specifications were an output swing on the VGA of $200\,\mathrm{mV} - 900\,\mathrm{mV}$, and a 4 bit ADC. The ADC is running at $82\,\mathrm{kS}/s$, meaning an oversampling rate of 2. The final switch capacitor VGA did unfortunately not integrate well with the ADC. Therefore the resistive feedback implementation was used as the final design. The low carrier frequency of $20 \pm 0.5\,\mathrm{kHz}$ is suited for an accurate and slow ADC, which is perfect for a SAR ADC. The flash ADC topology was chosen due to the time constraints.

A. Variable Gain amplifier

The VGA amplifies a small signal coming from the output of the low pass filter in the receiver. Since this signal can vary in amplitude from $1\,\mathrm{mV}$ to $100\,\mathrm{mV}$ the amplifier must be capable of varying its gain, such that its output signal has a constant amplitude. Higher output swing relaxes the requirements on the comparators in the ADC. The digital VGA control provides a binary code to control the feedback in the VGA, determining the gain of the VGA. The different feedback configurations are selected by a circuit that decodes the 3 bit digital value into a one-hot code that closes the switch on one feedback resistor.

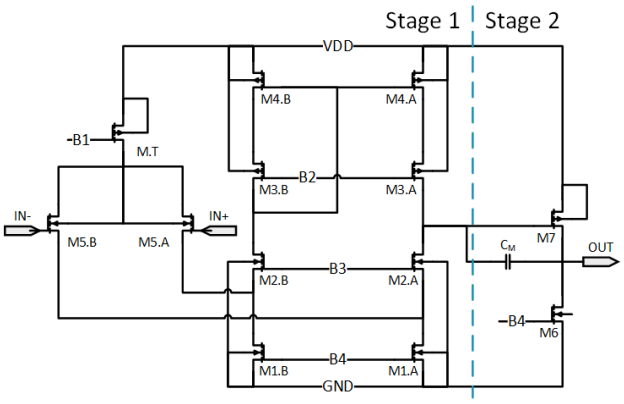

Fig. 2. Two stage Miller OTA Schematic.

The challenge with the VGA is creating a large enough gain and still maintaining a big swing at the output. Topologies such as folded- or symmetrical cascode OTA could give enough gain. However, the output swing would be limited by the cascodes. For the requirements to be fulfilled in one OTA a two stage solution with miller capacitance compensation was chosen, it is shown in Fig. 2. The first stage gives high gain, and the second stage provides the swing as it has no cascodes to lose voltage drop over. The OTA is used as a VGA by a resistor at the input and one in feedback shown in Fig. 3. The digital control signal is 3 bits and chooses which of the 8 switches is used as the feedback resistor. This solution made it possible to fine tune each gain level. The 8 gain levels are spaced linearly in dB. A better implementation using more feedback levels would be to use multiple switches at the same time, i.e more equivalent feedback resistor values. The ideal formula for the gain is given in Eq. 1, where R_{fb} is the feedback resistor and R_{in} is the input resistance. Equation 1 is not linear in our design, but has more of a logarithmic shape as the gain increases. This was adjusted for on each level by increasing feedback resistance.

$$Gain = \frac{R_{fb}}{R_{in}} \tag{1}$$

An alternative design would have a VGA with lower gain requirements followed by a fixed gain amplifier. This solution would reduce the area because it would scale down all the feedback resistors. The advantage of this design is the possibility of receiving a larger input signal, up to $350\,\mathrm{mV}$ amplitude. The resistors can be made on a chip, with the biggest one being $300\,\mu\mathrm{m}$ long, they are however noticeably big compared to the rest of the system. For reference, the biggest transistor in the OTA is $10\,\mu\mathrm{m}$. If the switch capacitor VGA would have worked, only around $30\,\mu\mathrm{m}$ of total feedback area would be required.

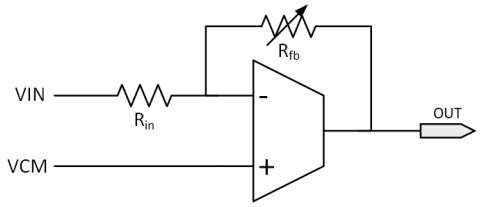

Fig. 3. VGA configuration.

The ADC operates with $700\,\mathrm{mV}$ peak-to-peak voltage swing, making the maximal gain requirement for the VGA in closed loop $50.9\,\mathrm{dB}$. The phase needs to be preserved for the algorithm in the digital part to work, so a bandwidth of $205\,\mathrm{kHz}$ was chosen, which is one decade after the highest input signal frequency. Because the phase starts dropping one decade before the bandwidth. The input capacitance of the ADC was estimated by the gate of the input transistors resulting in around $50\,\mathrm{fF}$ load for the VGA to be designed for.

The second stage transconductance was estimated with Eq. 2, where C_L is the load capacitance, and GBW is the product of gain and bandwidth [3].

$$gm_{out} = 2\pi C_L 4GBW \tag{2}$$

The gm-over-id method was used to find the output transistor size. The input transconductance was found from Eq. 3, where C_M is the miller capacitance between the stages in the OTA [3]. The Miller OTA was designed as at least 3 times the Capacitance at that node. Adjusting it high enough to keep the bandwidth, and low enough to avoid problems with the positive zero, this was determined iteratively.

$$gm_{in} = 2\pi C_M GBW \tag{3}$$

These starting values was enough to make an initial design using the gm-over-id method. A cut feedforward implementation of the miller capacitance was tried, butg higher gain was

achieved by placing the capacitor as shown in Fig. 2. There are two improvements we would have liked to add to the design, the first one being: changing the ideal voltage biases $B1$, $B2$, $B3$ and $B4$ to current mirrors from one ideal current source which would model a physical bandgap current mirror. We did try to do this, but underestimated the amount of adjustments and work to get the same results as with ideal biases. A second change we would make is to turn the resistors into switch capacitor equivalents. This would drastically reduce the chip area. We did manage to do this with the VGA as standalone, but unfortunately, we encountered integration difficulties with the ADC.

We extracted the root mean square error of the output noise and calculated the signal to noise ratio of the VGA with Eq. 4. A_{out} is the output amplitude, which is $350\,\mathrm{mV}$, $v_{n,rms}$ is the output root mean square error voltage.

$$SNR = 20log(\frac{A_{out}}{v_{n,rms}\sqrt{2}}) \tag{4}$$

We further found the total harmonic distortion through transient measurements. At the end of our design we realized we had designed the OTA with too strict requirements using the MC corner on the simulation. After the transient test over corners we realized that designing for the TT corner would give us more headroom with respect to the gain requirements in the design process.

B. Analog to Digital Converter

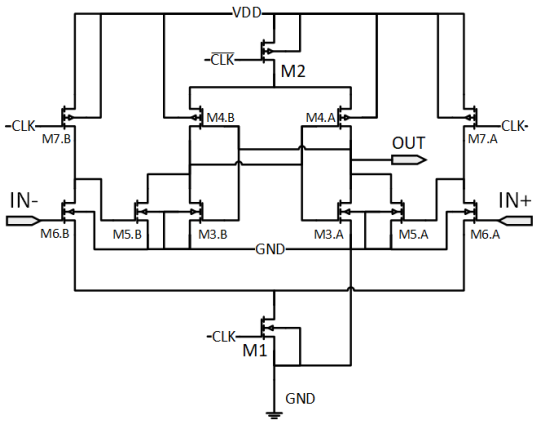

Fig. 4. Schematic of double tail comparator.

The chosen analog to digital converter (ADC) is a 4-bit flash ADC. It consists of a resistor ladder DAC that provides one of the comparison voltages to 15 comparators. The positive and negative reference voltages at the top and bottom of the resistor ladder are ideal. The comparators output the thermometer codes 0-15. The comparator used is the double tail comparator [4] shown in Fig. 4. It has two different phases, the reset phase, and the comparison phase. The clock signals that control these two phases are ideal, and the power consumed by generating these clock signals are not considered. In the reset phase the CLK signal goes low and the $M7.A$ and $M7.B$ transistors pre-charge the gate of the reset transistors, $M5.A$ and $M5.B$, to VDD. The reset transistors discharge the output nodes to zero. When the CLK signal goes high the comparator is in the decision phase. $IN+$ is the varying input coming from the VGA, and $IN-$ is a constant voltage coming from the resistor ladder. If $IN+ > IN-$ the reset transistor's gate on the plus branch is discharged faster, because there is more current going through $M6.A$ than $M6.B$. This makes the $M3.B$ nMOS on the opposite coupled inverter turn on, starting latch regeneration. Finally, the $M4.A$ pMOS turn on charging OUT to VDD and gives logical 1 as the output. The opposite happens in the latch if $IN+ < IN-$, and the comparator outputs a logical 0.

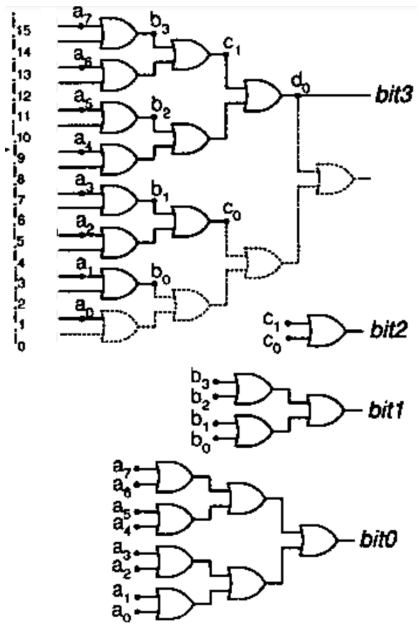

Fig. 5. Circuit schematic of the OR gates in the FAT tree encoder [5].

The thermometer codes coming from the comparators are then converted to binary codes with a FAT tree encoder. A circuit schematic is shown in Fig. 5. The logic in this circuit is asynchronous. The FAT tree encoder first converts the thermometer code to one-hot code with a set of AND gates connected to the comparators, next a set of OR gates convert the one-hot codes to binary. This binary code is then fed to 4 D-flip-flops that act as a register to provide the digital code for the digital circuit to read. Each of the output pins in the register is loaded with an ideal $2\,\mathrm{pF}$ capacitor to simulate the digital circuit.

V. Digital Design

A. Overview

After a signal is amplified and converted to digital data, it is necessary to process it to be able to decode the final message. The mathematical steps that are required to achieve this are: Mixing (I and Q extraction), Phase extraction ($\arctan(\frac{I}{Q})$) and Derivation.

Apart from the mathematical steps, there are other blocks that allow the digital demodulator to be able to correctly perform them as well as decode the final message. These blocks

are: VGA controller, frequency calibration, clock recovery, signal filtering and Varicode decoder.

After doing some initial research and understanding what each block should do and how it should do it, Verilog code was immediately written. Matlab was used to generate test vectors that would check if the block performs as expected, even in corner cases. For example for the Coordinate Rotation DIgital Computer (CORDIC) block was tested with all possible input combinations. The results of these tests were compared with predicted results calculated in Matlab.

After the blocks were tested separately with theoretical inputs, we started putting them together and testing them with input signals extracted from our high level Matlab mode. This way we could imitate real life behaviour of the system at different stages by setting the SNR, frequency and other parameters in the Matlab model. Again, the test vectors were generated by Matlab into hex files imported to a Verilog test bench.

The tools we were supposed to use, namely iSIM 2013, are outdated and lacked some basic functionality that added extra steps to development. For example it does not have an analog plot option for digital buses. To circumvent this problem, the test benches created csv files that we later analysed in Matlab. A benefit of this method is that we could test ideas in Matlab before trying it in Verilog, saving us a lot of time. For example introducing a moving average filter after the derivator was first tested in Matlab and then implemented in Verilog.

B. VGA Control

In the hardware, an R-2R DAC (AD5425) used a digitally configurable resistor as an input resistor of an inverting amplifier. This can control the gain of the amplifier by setting the input code of the DAC, thus forming a VGA.

As we were aiming for a lightweight design, we opted for a very elegant control method. The VGA control block simply checks the maximal value sampled by the ADC in the previous 256 samples. If this value is smaller than the binary code 1110 (14), the gain is incremented because the full range of the ADC is not being used. If the maximal value is 1111 (15), the gain is decremented because the signal is most probably clipping. The downside of this method, in comparison to more complex alternatives, is that it converges to the correct value slowly. However, since the sampling frequency is 80kHz, it takes only 0.82s to converge to the correct value in the worst case scenario (gain is 1 and maximal is needed or vice versa). Due to the low bitrate (100bits/s), only about 80 bits are missed at startup due to the VGA control.

C. Mixer

As previously mentioned, we had chosen the lightweight, purely digital implementation of the mixer [2] during our high level design stage. However, during implementation and testing, it was realized that even further simplifications can be made to the originally planned design. Most notably, the Hilbert filter was replaced with a simple resampling, saving even more area. The structure of the mixer can be seen in Fig. 6

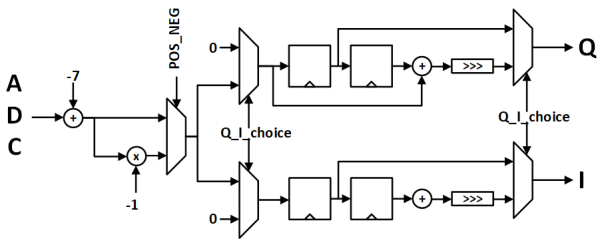

Fig. 6. Mixer structure.

The main idea is that the extraction of Q and I waveforms is done by sampling the input signal at exactly 4 times its input frequency. This means that each sample will be $90°$ phase shifted with respect to the previous one which is what is needed. The samples are then sorted to Q and I waveforms as well as inverted every other sample. This method gives the following sample sequence:

$$I_1, Q_2, I_3, Q_4, ...$$

Due to the sequential sampling, some I and Q samples are missed. In particular in the above sequence they are I_2 and Q_3. To mitigate the effect of missing samples, interpolators are added to both streams to approximate the missing samples. In the previous sequence that would be $I_2 \approx (I_1 + I_3)/2$.

The envelope of the I and Q signals needs to finally be extracted. A possible and often used [2] method is to use a Hilbert filter. However, due to the fact that the streams are essentially a sine wave of frequency f_s that is amplitude modulated by a wave of frequency f_c, the envelope can be extracted by just taking every fourth sample of the stream. This is a **completely original** method that we could not find in any previous work. An example of an extracted envelope can be seen in Fig. 7.

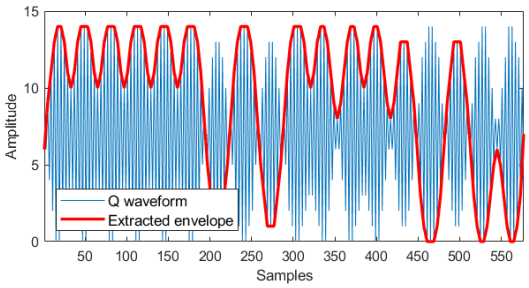

Fig. 7. Envelope extraction example. $Message = Hello; f_s = 4 \times f_c$.

An advantage of this method is that it avoids using any multipliers as well as an NCO, both of which are expensive in area. A downside is that the quality of Q and I waveforms depends heavily on the accuracy of the sampling frequency being $f_s = 4 \times f_c$. How this disadvantage is mitigated is explained in the section V-I.

D. Phase extraction - Vectoring CORDIC

The CORDIC algorithm in vectoring mode [6] was used to find the phase difference between the I and Q waveforms. The

alternative option that was considered was using a lookup table to perform this step. CORDIC was chosen due to the fact that it requires a lookup-table (LUT) of length only as many stages the algorithm performs, in our case 6. This is in contrast with 255 entry long LUT that would be needed for the alternative implementation. During the research phase we also found that many other GMSK demodulator implementations came to the same conclusion as us, further increasing the confidence of our choice. [7] [8] [9]

The first step in the algorithm is to account for the different quadrants the input vector may be in. This is shown on Fig. 8. The block is not pipelined because the clock speed is very slow at this point (1.25kHz) so critical paths are not a consideration.

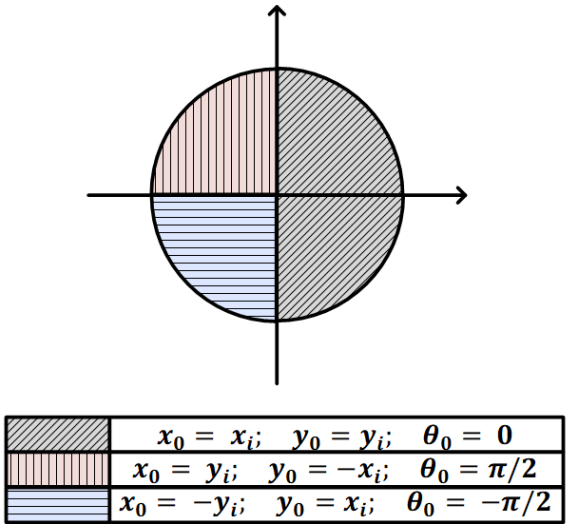

Fig. 8. Input vector quadrant correction

The quadrant correction is followed by 6 vectoring CORDIC stages described by Eq. 5.

$$\begin{cases} x_{i+1} = x_i + \delta_i 2^{-i} y_i \\ y_{i+1} = y_i - \delta_i 2^{-i} x_i \\ \theta_{i+1} = \theta_i + \delta_i \tan^{-1}\left(2^{-i}\right) \\ \delta_i = \text{sign}\left(y_i\right) \end{cases} \tag{5}$$

Due to the fact that CORDIC is basically just a binary search algorithm and the accuracy limited by the number of stages, this block introduces a phase error that is $< 1.42°$.

E. Derivator

The derivator is the simplest block implemented, it simply finds the difference between the current and previous sample outputted by the CORDIC block.

F. NRZ Signal Conditioning

The signal outputted by the derivator block is a non-return to zero signal. This means that the Varicode bits in which the message is encoded are represented only by the sign bit of this signal. Ideally it would be possible to decode the bits at this point, but unfortunately this is not possible and additional processing is needed to condition the signal into a usable form. The an example of the non-return-to-zero (NRZ) signal before and after filtering can be seen in Fig. 9.

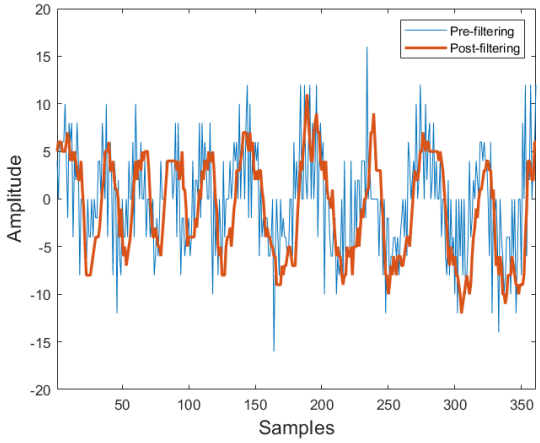

Fig. 9. NRZ signal before and after filtering. Demodulator input signal properties: $Message = "Hello"; SNR = 50dB; f_c = 20kHz$

These results were achieved with a two step approach. Firstly, all input zero values were set to be ignored - after all it is a NRZ signal. The second, more important step was to implement a moving average filter of window size 8 to further filter the signal. Moving average filters of window sizes that are powers of 2 are very cheap to implement in digital as they only include addition and shifting. It is interesting to note that moving average filters are essentially a subset of FIR filters that have all coefficients $k = 1/W$, where W is the window size. [10] The magnitude response of such a filter is seen in Fig. 10.

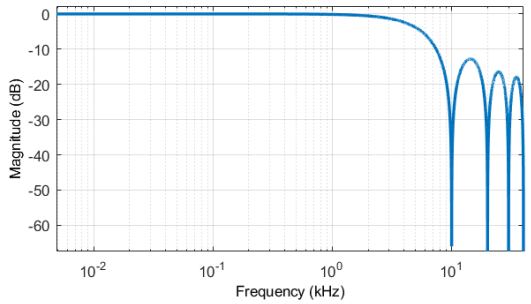

Fig. 10. Moving average filter magnitude response ($W = 8$)

Finally, there is a sort of one-bit filter implemented at the end for the sign bit. We reused the debouncer module for buttons and set its debouncing period to be 3 clock cycles. This makes sense because we know that the bit rate is 100bit/s so no changes in the sign bit that last only a few (we experimentally chose 3) can ever be correct bits.

G. Clock Recovery

The clock recovery block has the purpose of creating a clock signal that is used to sample the NRZ signal. It has a known frequency of 100Hz that is set by the specifications.

The function of the block is to recover the phase of the transmitter clock from the NRZ signal. This is done by a simple zero crossing detection that resets the clock counter, thus synchronising it with the bit transition.

An attempt was made to design a block that could recover the clock from a variable bitrate stream. Although it was promising, the block was abandoned due to time constraints.

H. Varicode Decoder

The Varicode decoder is a rather simple block. It samples the NRZ signal using the clock generated by the clock recovery module. If it detects two consecutive 0s, it pulls a *done* signal to high for 1 clock cycle and writes the ASCII character value to an output register based upon the content of a shift register that was storing previously sampled bits. Varicode to ASCII conversion is done using a lookup table.

I. Sampling frequency calibration and error correction

As mentioned in Section V-C, the mixer architecture that was used is very sensitive to imperfect sampling frequencies. Just as a reminder, the method works by sampling the at exactly $4 \times fc$ and sorting those samples to get two waveforms that are $90°$ offset to each other.

Let us first examine the mathematical consequence of incorrect sampling. A comparison between the ideal case and an extreme sampling frequency error of 20% can be seen in Fig. 11. It is visible that in such a case the phase difference between the I and Q streams is going to increase by a constant on every sample (when compared to the ideal sampling), in this case by $28°$. If the sampling frequency is too slow, the difference is going to be negative.

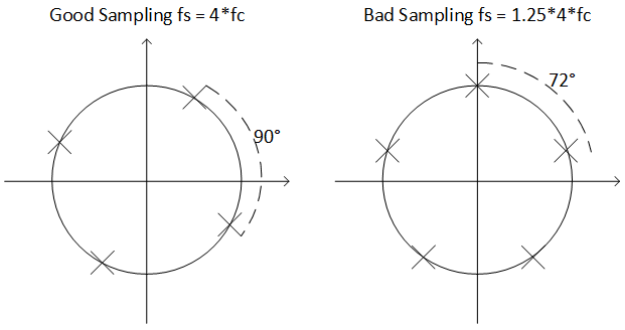

Fig. 11. Sampling error theoretical demonstration

Later on in the demodulation process, only the derivative of the phase angle between the I and Q waveforms is going to be important. Thus, from the previous example it is possible to conclude the derivative signal is just going to have a DC offset imposed onto it due to the incorrect sampling. Additionally this offset is directly proportional to the error of the sampling frequency. Figure 12 shows this effect in reality when the sampling frequency is $fs = 1.005 \times 4t \times fc$

To solve this problem, a calibration flow is executed that first finds the average value of the NRZ waveform and corrects the sampling frequency accordingly using a lookup table based on prior measurements. The average value is measured again. This value is later used to remove the DC offset that still persists. It is worth noting that Varicode encoding has slightly more 1s than 0s when transmitting English language sentences (56% of the bits are 1). The averaging error due to this is removed by subtracting a further experimentally determined constant from the NRZ signal. The full calibration flow is shown in Fig 13. At the time when we devised this approach, we had thought it to be original. However, we later found that this is a known method described in prior research [11] [12].

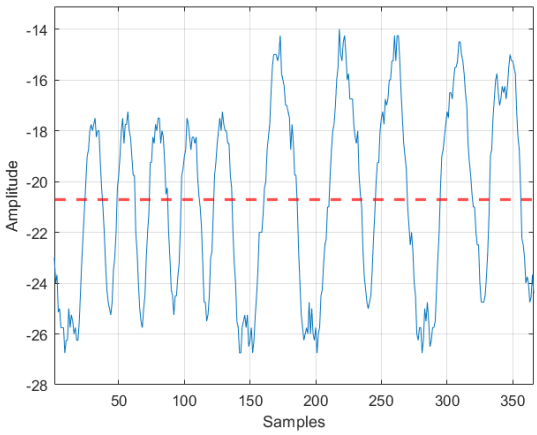

Fig. 12. DC offset due to sampling frequency error, $fc = 19.9kHz; fs = 80kHz$

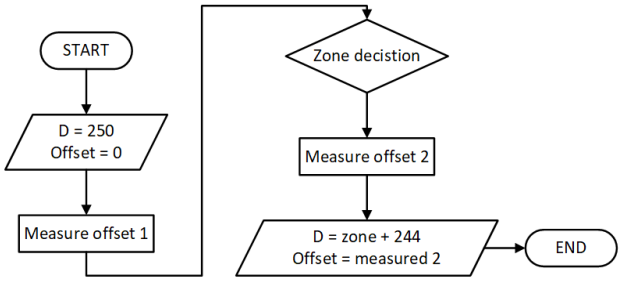

Fig. 13. Initialization process flowchart

VI. Results

A. Analog modules performance

TABLE II
SIMLUATIONS RAN.

Simulations
Transient over corners
Bode plots
Transient for INL/DNL
Transient for VGA gain levels
Noise and harmonic analysis

Table II shows what simulations were done. We were not able to do any Monte Carlo simulations as they would crash Cadence with an I/O disk write error.

The consistency of the VGA output swing for the 8 voltage levels using the ADC as a load, are shown in Fig. 14. The first

subplot shows the input voltages with the same peak to peak levels as shown in Fig. 15, and the second subplot shows the output voltages. We see a lot of overlap for the output of the different gain levels.

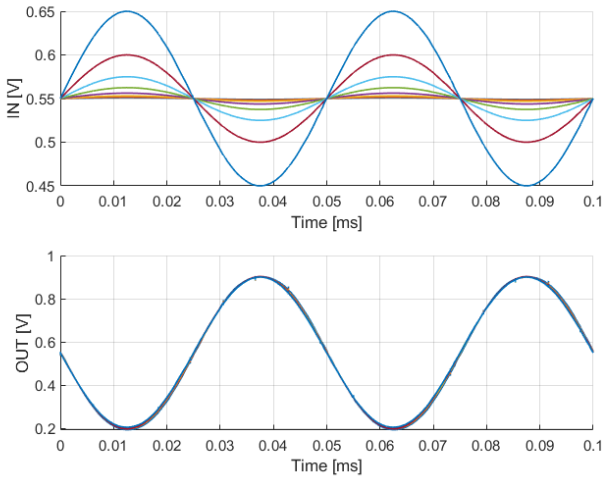

Fig. 14. Output signal of the VGA for the different gain levels.

The gain over frequency for the VGA is presented in Fig. 15. Note that the legend for the input voltages here indicate voltage amplitude.

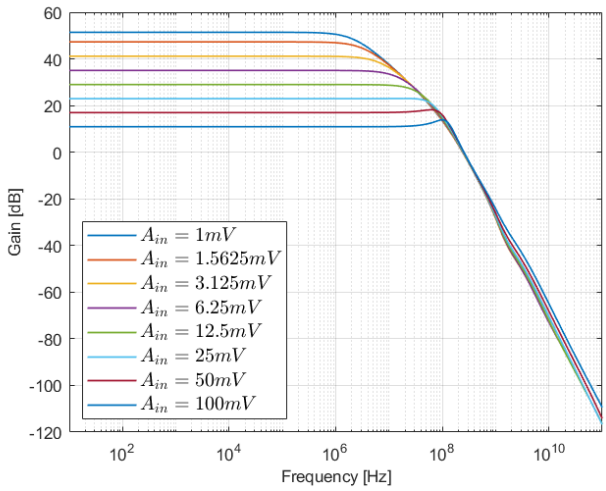

Fig. 15. OTA gain levels as a function of frequency.

The analog system's transient response to a $100\,\mathrm{mV}$ amplitude sine wave at frequency $20\,\mathrm{kHz}$ at the VGA's input is shown in Fig. 16. The nominal corner used in the design was MC. We see more signal swing in the TT corner, meaning we had more margins in the VGA design if it was designed using the TT corner from the start. The corners with the worst output voltage swings are FF and SS. In the SS corner there is some instability at low voltages, and the signal flips. A solution to correct the output swing to be constant like shown in Fig. 14, would be to make temperature sensors to control the feedback resistors in the VGA. For the ADC we see all the same conversions for the corners MC, TT, FS and SF. The corners FF and SS show different conversions, because the signal swing from the VGA is not large enough in the FF corner and because the signal clips for the low voltage in the SS corner.

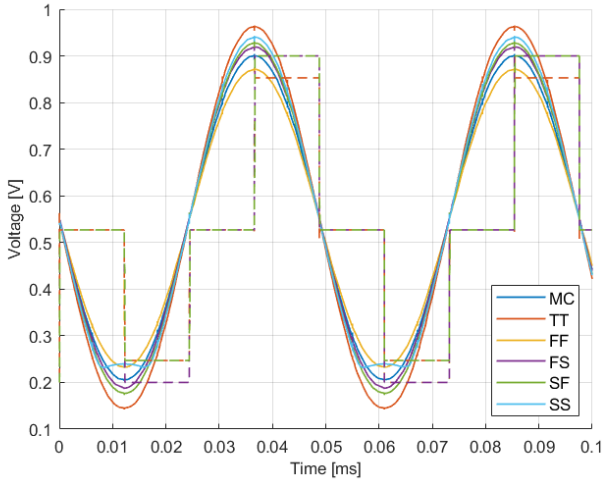

Fig. 16. Corner simulation for the VGA output, and ADC output transformed back to analog levels

TABLE III
VGA SPECIFICATIONS

Specifiation	Unit	Value
Dynamic power	μW	200
Static power	μW	140
# Levels	#	8
Min. Gain	dB	13.9
Max. Gain	dB	51.4
BW	MHz	2.14

In table III we show the gain range of the VGA, the bandwidth (BW), the dynamic power of the VGA in operation, and the static power for no input signal. The static power consummation is lower than the dynamic, although only $30\,\%$ difference.

TABLE IV
VGA NOISE AND DISTORTION

Input amplitude [mV]	SNR [dB]	THD [dB]
1	16.7	-36.2
1.56	20.8	-40.5
3.125	26.9	-46.3
6.25	32.9	-51.8
12.5	38.8	-56.6
25	44.6	-59.9
50	50	-58.7
100	55	-45.8

In Table IV we see that the SNR increases with more gain. This makes sense since the noise of the resistor components are proportional to the square root of the resistor values. As the input signal gets bigger we need a smaller feedback resistor to amplify the signal. We also see the total harmonic distortion (THD) follow the same trend. As the input signal becomes

easier to amplify, the distortion reduces. The THD is negligible compared to the noise introduced by the resistors.

The most important performance metric of the ADC is the integral nonlinearity (INL) and differential nonlinearity (DNL). They show how tall and wide each conversion step is. The test shown in Fig. 17 is used to calculate the INL and DNL for the different digital codes. The slow ramp voltage is generated by an ideal source. Only the time when the clock is low and the result is finished is used for the INL and DNL calculations, this means that the transient spikes down at the start of a clock period does not effect the nonlinearity results. The INL and DNL is shown in Fig. 18. The values are low enough that no digital codes are missing. All of this data was recorded on the nominal corner. Table V summarises the specifications. The power consumption is measured by taking the RMS current drain of the circuit and multiplying it with the supply voltage of 1.1 V.

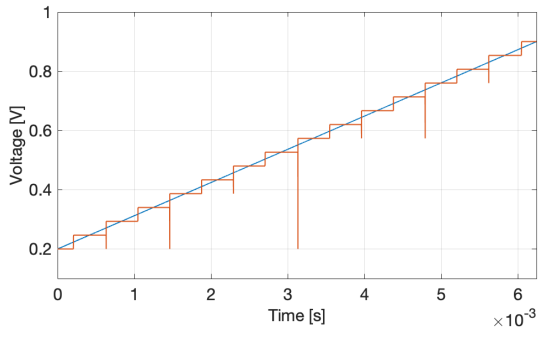

Fig. 17. Transient response of ADC when excited by a very slow analog input ramp, the digital code has been converted back to an analog voltage by an ideal DAC.

Fig. 18. INL (top) and DNL (bottom) of the ADC.

TABLE V
ADC SPECIFICATIONS

Specifiation	Unit	Value
Dynamic power	µW	6.39
Static power	µW	6.83
Bits	#	4
INL	LSB	0.0256
DNL	LSB	0.0256

B. Digital modules area utilization

Table VI shows the FPGA utilization of each block. Table VII shows the total FPGA utilization compared to an implementation that also uses a CORDIC engine but a traditional mixer [7].

TABLE VI
FPGA AREA UTILIZATION BY MODULE

Module	LUT #	Slices #
VGA Control	63	37
Mixer	31	34
CORDIC	131	90
Derivator	8	12
NRZ Smoothing	85	57
Clock recovery	16	10
Frequency init.	43	24
Top level	50	44

TABLE VII
TOTAL FPGA AREA UTILIZATION

Components	#	%	Comparable design [1] [7]
Slice FFs	392	7%	763
4 input LUTs	572	11%	4521
Slices	446	18%	NA
Total # 4 input LUTs	708	14%	NA
RAM16s[2]	8	66%	NA
Multipliers	**0**	**0%**	NA

C. Message decoding results

Table VIII shows the messages that were decoded from the transmitter board at different levels and the symbol error rate at that level. The reason for using symbol error rate instead of BER is that Varicode encoding was created to be self-synchronising and thus more resilient to some single bit flips in the way that the characters that come after the error will not be impacted. Additionally some errors can lead to invisible characters when decoded that cannot be detected when reading the message (NUL, SOH, STX, etc.). The demodulator is designed to decode messages that are human-readable and this is why we believe that the symbol error rate is a better measure of its success.

It is worth noticing that the error rate is not increasing linearly with a degrading SNR. There are several variables that influence the signal quality apart from noise. An important factor is related to the carrier frequency and how close it is to one of the frequencies that can be generated by just

[1] Virtex-4 FPGA, not all parameters listed in referenced paper
[2] All RAM16s are used by the UART transmitter that was provided to us

TABLE VIII
SECRETS AND QUOTES BY LEVEL

L	SER	Secret and quote
0	2.5%	aA6a95cfB46vbtdGm2VMKQsWeNov7s3P "Such is the disposition of men, that we value what is speculative and precarious, more than what is safe and beneficial." -William Playfair
1	0.3%	BcoUdlFqU1WFuA3MdHCy0rlRxKPojawl "On two occasions I have been asked, – 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' In one case a member of the Upper, and in the other a member of the Lower, House put this question. I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question." -Charles Babbage
2	20%	NpaKClYY3DdmIrnytTdzbw8vAv3zLCkw "The aim of science is to make difficult things understandable in a simpler way; the aim of poetry is to state simple things in an incomprehensible way. The two are incompatible." -Paul Dirac

dividing the $20MHz$ onboard clock. Furthermore the SNR is not increasing linearly from level to level but logarithmically.

Even though our implementation is not able to successfully demodulate levels above 2 at this point, we were able to confirm that the frequency calibration flow is able to recover the carrier frequency even up to level 7.

VII. SUMMARY

During the course of this project, we sought out to develop a unique solution to demodulating low frequency GMSK signals. Our strategy was based upon creating an extremely efficient design which we believe we have succeeded in. In the entirety of the digital part, not a single multiplication operation was done, allowing the whole implemented design to use only 7% of available flip-flops and 14% of LUTs of the Spartan 3A FPGA. Furthermore, the solution is unique with with some parts being completely unique or improved versions of existing implementations.

The trade-off of such an approach was relatively low performance in terms of the SER. However, we believe that with some improvements it is possible to turn this design into a viable option of ultra low power and low speed applications such as IOT temperature sensors or communication with health monitoring devices. Additionally some applications might not need to operate in an environment where big carrier frequency changes are an issue. Possible improvements might include:

- Using a low power 8-bit (or more) SAR ADC instead of a 4-bit flash
- Implementing a feedback circuit that is better against mismatch in the resistors, or doing a switch capacitor implementation.
- Additional noise-filtering analog stage to improve SNR
- Increasing the clock frequency to allow for a more precise sampling frequency setting
- Periodical automatic re-calibration of the DC offset removal for the sampling frequency error
- Inclusion of a more advanced specification of the data link layer that includes error detection and correction features regardless of SNR

APPENDIX

ALTERNATIVE DIGITAL DOWN CONVERTER

The aim of the Digital Down Converter (DDC) is to demodulate and down sample the sampled received signal generating two baseband signals, one in phase (I) and one in quadrature (Q) with a $\pi/2$ phase shift. The broadband signal is brought to baseband by multiplying the received modulated signal with the modulating frequency. The modulating frequency is generated by a Numerically Controlled Oscillator (NCO). The received samples are multiplied with the modulating wave samples to generate images at the sum and difference of the Nth multiples of the intermediate frequency. To remove the unwanted broadband images a low pass filter is applied. Furthermore, the demodulated signal is down sampled to reduce sampling rate.

A Numerically controlled oscillator (NCO) generates the modulating frequency of 20 kHz. A quarter wave LUT based oscillator was chosen for its compromise between simplicity and area/power efficiency.

Multiplier

To demodulate the sampled received signal with the carrier wave a multiplier is used. A 4 - bit Vedic multiplier is developed for its low area and low power characteristics since it only employs half adders, full adders and logical AND operations [13]. To further increase power efficiency and speed some full adders and half adders are replaced with 3:2 and 4:2 compressors. The compressors have improved performance because of the reduced internal load capacitances [14]. The hardware architecture of the Vedic Multiplier is shown in Fig. 19.

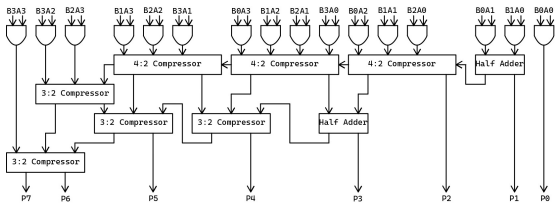

Fig. 19. Hardware architecture of the 4-bit Vedic Multiplier with 3:2 and 4:2 compressors.

Cascaded Integrator-Comb

A Cascaded Integrator-Comb (CIC) decimation filter is used to low pass filter and down sample the demodulated signal. A CIC filter is used since it only utilizes delays and adders, compared to a direct form of an FIR filter which implements also multipliers. A CIC decimation filter is made from three basic building blocks: an integrator section, a down sampling section and a comb section as shown in Fig. 20.

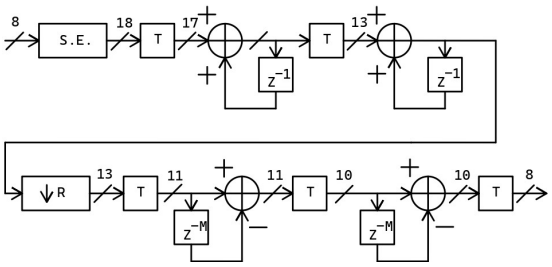

Fig. 20. CIC filter with decimation factor of $R = 4$, with filter order $N = 2$, differential delay $M = 8$, sign extending unit $S.E.$, and truncation sections T.

The frequency response of the CIC filter is fully determined by three design parameters: the order of the filter N, the down sampling rate R, and the differential delay M. To reduce the total number of adders the number of stages is reduced by choosing a small filter order $N = 2$. The down sampling rate of $R = 4$ is chosen. To achieve a bandwidth of $500Hz$, the last design parameter, the differential delay must be $M = 8$. With these design parameters the gain of the CIC is calculated to be $Gain = (MR)^N = 1024 \simeq 60dB$. The gain can be mitigated by dividing the output by the gain magnitude or if the gain magnitude is an integer power of two by right shifting the final result by $\log_2((MR)^N) = \log_2((8 \cdot 4)^2) = 10$ bits.

To achieve accuracy by avoiding overflow the filter must be implemented with registers with a minimum bit width calculated with

$$W_{out} = W_{in} + \lceil \log_2(MR) \rceil \tag{6}$$

where W_{in} is the input bit width, M the differential delay and R the down sampling factor. Nevertheless, not all registers of all stages must have the minimum bit width calculated in Eq. 6. To minimize area consumption and reduce computation complexity register bit pruning is used. Register bit pruning is a distribution of the register truncation at each consecutive stage instead of a single truncation at the final stage while realizing the same quantization error. Register bit pruning is computed with the algorithm from [15] and [16]. CIC filters do not have a flat passband therefore a compensating FIR filter is designed with an inverse frequency response. However, the FIR filter would require multiple multipliers and would increase drastically the computation complexity. Therefore, the compensating FIR filter is omitted as a tradeoff for simplicity.

REFERENCES

[1] T. Turletti, "Gmsk in a nutshell," Oct. 1996.

[2] G. Saulnier, C. Puckette, R. Gaus, R. Dunki-Jacobs, and T. Thiel, "A vlsi demodulator for digital rf network applications: Theory and results," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 8, pp. 1500–1511, 1990. DOI: 10.1109/49.62828.

[3] P. P. Reynaert, "Design and implementation of analog circuits," VGA design, 2022.

[4] S. Babayan-Mashhadi and R. Lotfi, "Analysis and design of a low-voltage low-power double-tail comparator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 343–352, 2014. DOI: 10.1109/TVLSI.2013.2241799.

[5] D. Lee, J. Yoo, K. Choi, and J. Ghaznavi, "Fat tree encoder design for ultra-high speed flash a/d converters," in *The 2002 45th Midwest Symposium on Circuits and Systems, 2002. MWSCAS-2002.*, vol. 2, 2002, pp. II–II. DOI: 10.1109/MWSCAS.2002.1186804.

[6] J. E. Volder, "The cordic trigonometric computing technique," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, 1959. DOI: 10.1109/TEC.1959.5222693.

[7] L. Kumar, D. K. Mittal, and R. Shrestha, "Vlsi-design and fpga-implementation of gmsk-demodulator architecture using cordic engine for low-power application," in *2016 IEEE Annual India Conference (INDICON)*, 2016, pp. 1–6. DOI: 10.1109/INDICON.2016.7838954.

[8] X. Cui, D. Yu, S. Sheng, and X. Cui, "A cordic demodulator platform for digital-if receiver," in *2006 8th International Conference on Solid-State and Integrated Circuit Technology Proceedings*, 2006, pp. 2025–2027. DOI: 10.1109/ICSICT.2006.306582.

[9] O. Kislal, "Gmsk modulator/demodulator design and implementation on fpga for cube satellites," BSc Thesis, Jun. 2016. DOI: 10.13140/RG.2.2.24158.64321.

[10] Matlab Documentation, *How is a moving average filter different from an fir filter?* Available at https://nl.mathworks.com/help/dsp/ug/how-is-moving-average-filter-different-from-an-fir-filter.html (2005/06/12).

[11] Y.-L. Huang, K.-D. Fan, and C.-C. Huang, "A fully digital noncoherent and coherent gmsk receiver architecture with joint symbol timing error and frequency offset estimation," *IEEE Transactions on Vehicular Technology*, vol. 49, no. 3, pp. 863–874, 2000. DOI: 10.1109/25.845105.

[12] D. Han and Y. Zheng, "A mixed-signal gfsk demodulator based on multithreshold linear phase quantization," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 9, pp. 719–723, 2009. DOI: 10.1109/TCSII.2009.2027969.

[13] S. R. Huddar, S. R. Rupanagudi, M. Kalpana, and S. Mohan, "Novel high speed vedic mathematics multiplier using compressors," *Proceedings - 2013 IEEE International Multi Conference on Automation, Computing, Control, Communication and Compressed Sensing, iMac4s 2013*, pp. 465–469, 2013. DOI: 10.1109/IMAC4S.2013.6526456.

[14] S. F. Hsiao, M. R. Jiang, and J. S. Yeh, "Design of high-speed low-power 3-2 counter and 4-2 compressor for fast multipliers," *Electronics Letters*, vol. 34, pp. 341–343, 4 Feb. 1998, ISSN: 00135194. DOI: 10.1049/EL:19980306.

[15] E. B. Hogenauer, "An economical class of digital filters for decimation and interpolation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, pp. 155–162, 2 1981, ISSN: 00963518. DOI: 10.1109/TASSP.1981.1163535.

[16] R. Lyons, *Computing cic filter register pruning using matlab [updated]*, Jan. 2022. [Online]. Available: https://www.dsprelated.com/showcode/269.php.