# Design of a GMSK 100 bit receiver

Jonas Bertels, Jasper Depuydt, Ruben Heyrman, Ziyu Zhou

## Introduction

THIS document describes the analog and digital design of a GMSK-100 receiver, used in a battery powered Internet-of-Things application. Such an application is low-throughput and should be low-energy. The analog front-end receives a varying amplitude, 100 bps GMSK signal on a $\pm 20$ kHz carrier frequency and passes a series of bits on to the digital hardware at 50 kHz. This requires variable amplification, as well as analog-to-digital conversion. Both modules are implemented in $0.35\mu m$ CMOS technology. The digital hardware, which demodulates the GMSK signal, will be implemented on a FPGA board. It uses a Costas loop to derive the frequency of the frequency modulated signal, then uses this frequency to retrieve the ascii bytes.

## I. Analog Front-end

### A. System overview

The analog front-end should be able to receive a noisy signal with a small variable amplitude and deliver a consistent digital output to the digital decoder. The implementation consists of two building blocks: a variable gain amplifier and an analog-to-digital converter. Because the GMSK demodulation doesn't require a high precision ADC, the proposed building blocks of the analog front-end are a Nyquist rate 3-bit Flash-ADC and a switched-capacitor amplifier with a telescopic OTA.

### B. Variable gain amplifier

The variable gain amplifier consists of a telescopic OTA and a switched-capacitor feedback network. The switched-capacitor amplifier and OTA topology are shown in figures 1 and 2. The use of cascodes offers a high gain, but sacrifices output swing. In the next sections we will discuss the reasoning for the OTA specifications, the design and results.

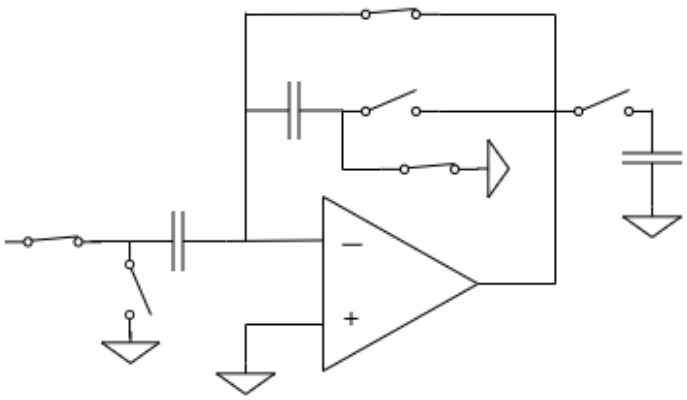

Fig. 1. Switchcap Amplifier

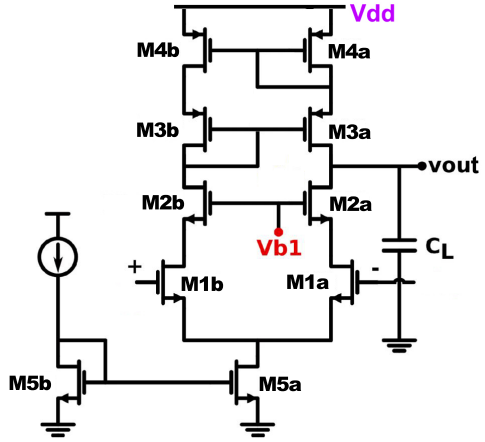


Fig. 2. Telescopic OTA

*1) Calculations:* We first consider the desired amplitude of the amplified signal, keeping in mind the limited output swing of the OTA and the needed ADC input. We decided that 1V would suffice for the input of the ADC. This means that the closed loop gain needs to be minimally 5 (for 100 mV input amplitude) and maximally 500 (for 1 mV input amplitude).

For the bandwidth of the closed-loop system, we choose a settling error of 5%, while sampling at 50 kHz.

$$5\% = e^{\frac{-t}{RC}}$$

where t equals half a sampling period. This results in a closed-loop bandwidth of $\frac{1}{2\pi * RC} = 47.7kHz$ and a gain-bandwidth of $47.7kHz * 500 = 23.8MHz$.

By choosing a gain error of 3%, we can determine the needed DC gain for the OTA.

$$3\% = \frac{A_{ideal} - A_{real}}{A_{ideal}} = \frac{\frac{1}{\beta} - \frac{A}{1+A*\beta}}{\frac{1}{\beta}} = \frac{500 - \frac{A}{1+A*/500}}{500}$$

This results in a required DC gain (A) of 84.2 dB for the OTA. To reach the desired GBW of $23.8MHz$, we need an input transconductance of:

$$g_m = GBW * 2\pi * C_L = 23.8MHz * 2\pi * 1.2pF = 179\mu S$$

where $C_L$ is the total capacitance seen at the output node of the OTA. We also estimate the current needed through the input pair:

$$I_{ds} = \frac{g_m * (V_{gs} - V_t)}{2} = \frac{179\mu S * 0.2V}{2} = 17.9\mu A$$

We can now do a quick check to see if the proposed design is realistic in terms of power consumption. This

OTA would use $2I_{ds} * V_{dd} = 118\mu W$. Assuming that the OTA is the building block that uses the most power of the whole system, this is an acceptable power usage because it can run for almost ten years on an average cell phone battery.

To get estimates of the dimensions of the transistors, we use the following formulas:

$$L_{channel} = r_{ds} * I_{ds}/V_E$$
$$I_{ds} = \mu * \frac{C_{ox}}{2} * \frac{W}{L} * (V_{gs} - V_t)^2$$

The sizing of the transistors is an iterative process, so the final dimensions are reached by testing.

We also need a certain slew rate to obtain distortion-free operation:

$$SR_{min} = 2\pi f A V_{in} \text{ with } A = \text{gain and } V_{in} = \text{ input amplitude}$$
$$= 2\pi * 20kHz * 500 * 1mV = 0.063V/\mu s$$

The calculated specifications for the closed loop system and the OTA itself are listed in the following tables:

| Gain error (%) | Settling error (%) | $f_s(kHz)$ |
|---|---|---|
| 3 | 5 | 50 |

| $BW_{CL}(kHz)$ | $Gain_{CL,max}(/)$ | GBW (MHz) |
|---|---|---|
| 47.7 | 500 | 23.8 |
| $BW_{OL}(kHz)$ | $Gain_{OL}(dB)$ | GBW (MHz) |
| 1.47 | 84.2 | 23.8 |
| $C_L(pF)$ | output swing (V) | $SR_{min}(V/\mu s)$ |
| 1.2 | 1 | 0.063 |

*2) Noise:* The exact values of the input ($C_0$)- and feedback ($C_x$) capacitors can be found using an SNR requirement, which we decide to be 40 dB. This is done by a noise analysis of the capacitors and the OTA during the two phases. The minimal input signal is 2 mV peak-to-peak. The signal power can then be compared with the calculated total integrated noise power, which leads to the SNR and the determination of the exact capacitor values. The following formulas show the derivation of the noise contribution of the switches and the OTA during both phases. The calculation is always the same: the noise source is referred to the output using the transfer function and is then integrated over the whole noise spectrum:

$$\int_0^\infty \overline{dv^2(f)} * H(f)^2 df$$

*Phase 1: switches*

$$\frac{kT}{C_0} + \frac{kT}{C_X}$$

*Phase 1: OTA*

$$\frac{2kT}{3}(\frac{C_0 + C_x}{C_x})^2(\frac{C_x}{C_L(C_0 + C_x) + C_0C_x})$$

*Phase 2: switch*

$$4kTR_{S1}(\frac{C_0}{C_x})^2\frac{1}{4R_{S1}C_0} = \frac{kT}{C_0}(\frac{C_0}{C_x})^2$$

*Phase 2: OTA*

$$\frac{2kT}{3}(\frac{C_0 + C_x}{C_x})^2(\frac{C_x}{C_L(C_0 + C_x) + C_0C_x})$$

The formulas lead to the following output-referred integrated noise power seen over the two phases. This has to be compared to the output-referred signal power, thus leading to an expression for the SNR. Solving for $C_X$ and $C_0$ and keeping into account that the latter is 500 times the former, this leads to a minimal value for the capacitors, which is approximately 56.4 pF for $C_0$ and 113 fF for $C_X$. For comparison, the load capacitance $C_L$ is equal to 1.2 pF.

$$P_{noise} = (2 * \frac{kT}{C_0} * (\frac{C_0}{C_X})^2 + \frac{kT}{C_X} + \frac{2kT}{3}\frac{1}{C_0 + C_x}(1 + \frac{C_0^2}{C_x^2}) + \frac{2kT}{3}(1 + \frac{C_0}{C_x})^2\frac{C_x}{C_L(C_0 + C_x) + C_0C_x})$$

$$P_{signal} = 1^2/2$$
$$SNR = \frac{P_{signal}}{P_{noise}} = 10^{(SNR_{dB}/10)} = 100000$$

*3) Design & results:* After designing and optimizing the OTA, we reached the following results:

| $BW_{OL}(kHz)$ | $Gain_{OL}(dB)$ | GBW (MHz) |
|---|---|---|
| 1.3 | 85.4 | 24.2 |
| $C_L$ (pF) | output swing (V) | Power ($\mu W$) |
| 1.2 | 1.4 | 99 |

The results are also visible in the bodeplot in figure 3. We can conclude that we achieved the required open-loop gain to have a gain error of under 3%. Also the GBW and slew rate are reached. The power consumption of the OTA is $I_B * V_{dd} = 99\mu W$ which is a bit lower than calculated.
Figure 4 shows the result of a monte carlo simulation of the transient response of the switched-capacitor amplifier to a 1 mV 1 kHz input sine. The input offset is relatively large (up to 3.6 mV for 30 monte carlo iterations). This could be reduced by increasing the sizes of the M1 and M4. In order to meet the required specifications, the power consumption would also have to go up. In the following table the dimensions of the transistors in figure 2 are shown.

| | M1 | M2 | M3 | M4 | M5a | m5b |
|---|---|---|---|---|---|---|
| W($\mu m$) | 18 | 18 | 36 | 36 | 18 | 9 |
| L($\mu m$) | 1.75 | 1.75 | 1.75 | 1.75 | 1.05 | 1.05 |

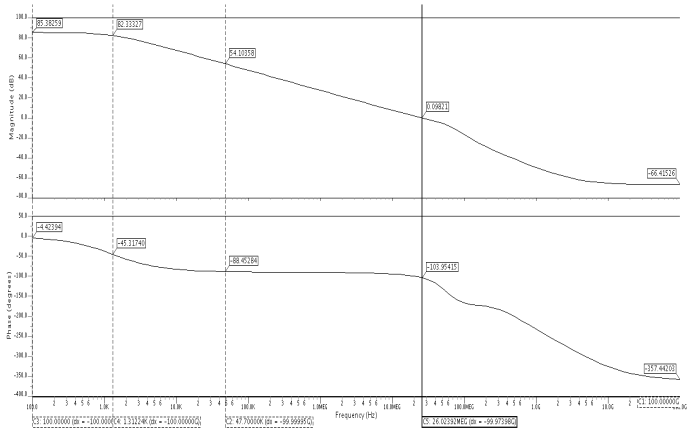

Fig. 3. Bodeplot OTA

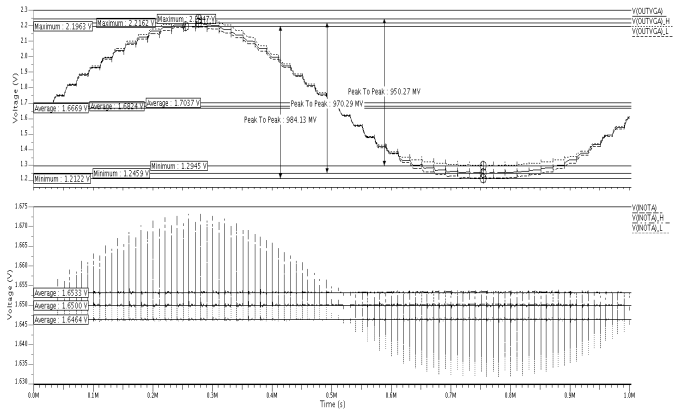

Fig. 4. Transient signal input OTA and output SC amplifier (1 mV 1 kHz input sine)

*4) Feedback:* To make the gain variable, a resistive feedback network was added to the OTA. This resulted in too much output offset due to mismatch at the input, therefore we decided to go with a switched-capacitor implementation. The sample frequency is 50 kHz, so we are sampling a bit faster than Nyquist frequency. To control the gain, a variable gain circuit is implemented as shown in figure 5. The 'gain detector' checks the output bits of the ADC. If the sample is in between a certain range (between 2 and 5), the up signal goes to 1. If the sample is a maximal value (0 or 7), the down signal goes to 1. The 'gain queue' consists of a delayline of registers. The number of registers for the delayline can be tuned in case of different frequency or different waveform signals. If all entries in the delayline are respectively up or down signals, the enable signal or the enable and subtract signal are activated. When the enable signal is activated, the 'gain queue' is reset. The last part is the 'gain regulator', the 3-bit integer stored in the register represents the closed-loop gain of the amplifier circuit. When the adder receives an enable signal, it either adds or subtracts 1 from the current 3-bit value. This 3-bit value will change the value of the input capacitor of the switched-capacitor feedback network and therefore the closed-loop gain of the amplifier. The result of the implemented variable gain amplifier is shown in figure 6 and 7 for a 20 kHz sine input, being adaptively amplified to a peak-to-peak voltage of 1V.

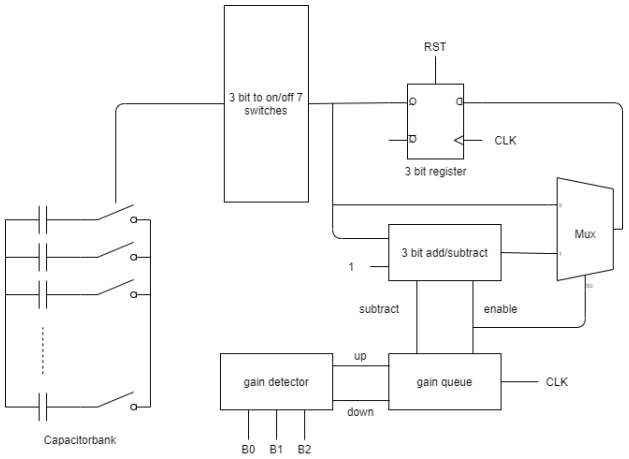


Fig. 5. Variable gain circuit

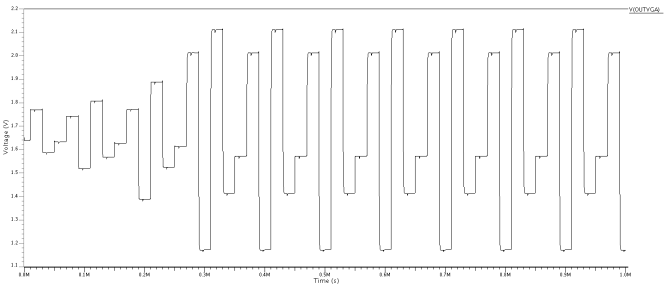

Fig. 6. Variable gain rising (20 kHz sine)

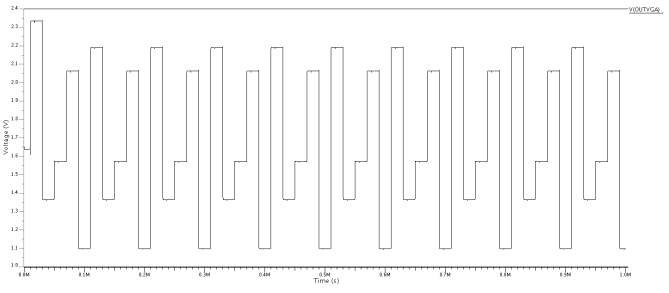

Fig. 7. Variable gain falling (20 kHz sine)

## C. Flash Analog-to-Digital Converter

*1) Calculations:* From MatLab simulations, we decided that 3 bits will suffice. This N=3 provides a maximum quantization SNR of $6.02N + 1.76dB = 19.82dB$ in case of a Nyquist frequency ADC. Just like the switched-capacitor

amplifier, the ADC will work at 50 kHz. With an input voltage range of 1 V, the LSB voltage is $\frac{V_{in}}{2^M} = 0.125V$.
A flash ADC was chosen because of its straightforward implementation (figure 8).

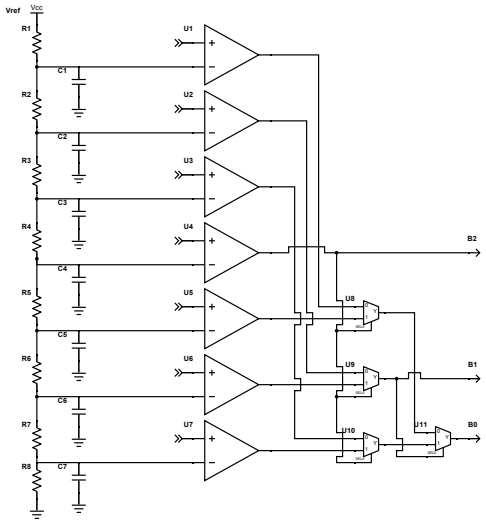


Fig. 8. Flash ADC

*2) Comparators:* There are 7 StrongArm comparators [2], as shown in figure 8 and 9. They consume no static power and produce rail-to-rail outputs. The power consumption of the comparators is mainly due to the charging and discharging of the capacitances and is roughly equal to:

$$\begin{aligned} P &= f_{CK}(2C_{P,Q} + C_{X,Y})V_{DD}^2 \\ &= 50kHz * (70fF + 31fF) * (3.3V)^2 \\ &= 0.055\mu W \end{aligned}$$

This makes the power consumption of all 7 comparators together equal to $0.39\mu W$.

The disadvantage of these comparators is the fact that they introduce kickback on the input signal when several work in parallel. This was remedied by adding an extra capacitor between the OTA and the ADC, which leads to a higher power consumption of the OTA in order to reach the specifications. If we wanted to strongly reduce the kickback effect, we could use a SAR ADC instead of a flash, because there is only one comparator instead of 7 in parallel. This would also have a positive effect on the power consumption, because only 1 comparator is used and the extra capacitor can be smaller or even removed.

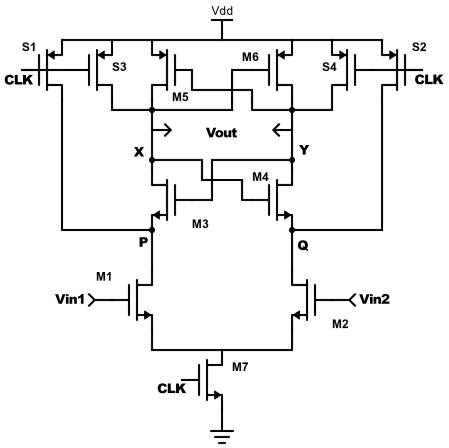


Fig. 9. StrongArm Comparator

*3) Encoder:* The 7 comparators of the flash ADC generate a 7-bit thermometer or unary code. In order to convert this code into a 3-bit binary result, a thermometer-to-binary is used with a multiplexer-based encoder topology as seen in figure 8. This circuit has advantages such as a low power consumption and a low circuit complexity. [1]

### *D. Results*

Figure 10 and 11 show the output of the amplifier with a fixed gain and the output of an ideal DAC placed after the ADC, to visualize and verify the correctness of the 3 output bits. This was done for an input sine of 1 mV and 100 mV, both at 3kHz. In figure 10 we can clearly see the settling time of the switched-capacitor amplifier. The figures also show the results of the Monte Carlo simulation, visualized by stacking the resulting waveforms on top of each other. This mismatch could be made smaller by making the input transistors of the comparators of the ADC and OTA larger, but that would also increase the power consumption. For the comparators, there also is a trade-off between mismatch and kickback. The power consumption of the analog front-end is dominated by the OTA. For the total power consumption, the clock generators, digital circuitry (variable gain circuit and mux-based encoder) and biasing circuits should also be taken into account.

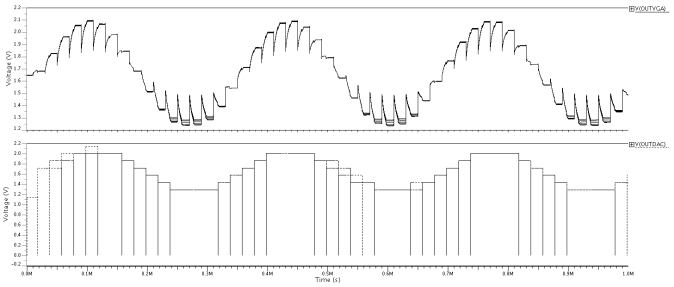

Fig. 10. VGA and ideal DAC output (1 mV 3kHz input sine)

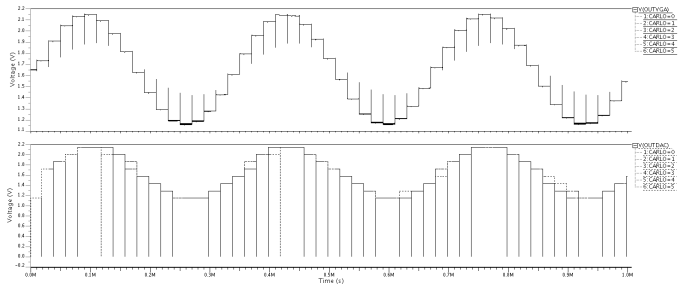

Fig. 11. VGA and ideal DAC output (100 mV 3 kHz input sine)

## II. Digital Design

### A. The full design: receiver based of frequency recovery

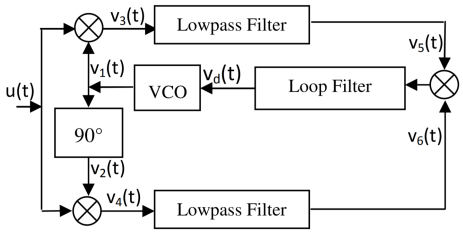


Fig. 12. Frequency recovery part of our design, taken from [3]

Looking at figure 12, we can find the following equations in the locked loop case:

$$u(t) = cos(\omega_c * t + m(t) * t)$$

with $m(t)$ being a positive gmsk pulse for 1, and a negative gmsk pulse for 0.
The signal $v_d$ will converge to this carrier frequency by creating the signals:

$$v_5(t) = 1/2 * cos((\omega_c + m(t) - v_d(t))t)$$

and

$$v_6(t) = 1/2 * sin((\omega_c + m(t) - v_d(t))t)$$

The product of these signals is:

$$v_5(t) * v_6(t) = 1/8 * sin(2 * (\omega_c + m(t) - v_d(t))t)$$

The loop filter is given by figure 13, with $C_1 = 1$ and $C_2 = \frac{1}{64}$. At first, $v_5(t) * v_6(t)$ will oscillate over the possible frequencies. As it reaches the wanted frequency, it will slow down in its oscillation and will stop completely (some overshoot is possible) at the desired frequency, where the product will become (approximately) zero. This is shown in figure 14 and 15.

Once we have locked to the desired frequency, we have:

$$v_d(t) = \omega_c + m(t)$$

This signal is mixed with some high frequency components, because our loop filters are not perfect. To filter this signal and retrieve our ascii bytes, we apply the circuit given in figure 16.

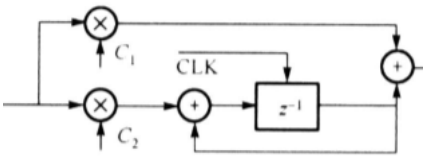


Fig. 13. Loop filter design

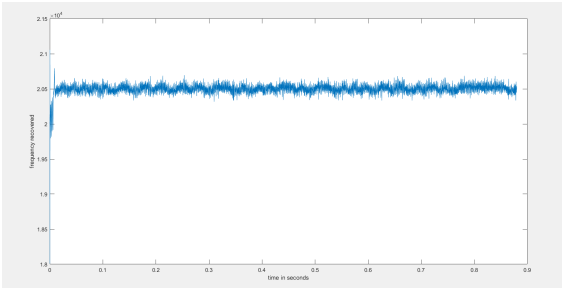

Fig. 14. Frequency recovery for f = 20500 Hz

It is important to note that the ”normal way” of retrieving the signal would be to use the recovered frequency to demodulate the in-phase and quadrature components, take the angle between both components and then differentiate. We originally designed our circuit this way and implemented all the required components to achieve this in hardware. However, once the VCO input is filtered, we receive the waveform of figure 17. It is clear that the message ”Hello World!” can be easily retrieved from this signal, without the need for a circuit to recover the angle and find the differential of this angle.
The disadvantage of the ”straight-from-VCO” method is that it requires a better low-pass filter to remove the noise seen in figure 14. In our design we decided to use the 22nd order FIR filter described in the next section, and whose hardware implementation is given by figure 18. We required 2 of these to filter the VCO output, whereas the inphase and quadrature signals would already have been filtered by the costas loop.
However, because the VCO frequency closely tracks the input signal, what actually came out of the quadrature and in-phase outputs was not useful. A separate demodulator would have to be built using an averaged VCO output. This would require even more hardware, making the ”straight-from-VCO” method the simplest and most area efficient method.

### B. Design of the filter

*1) Matched filter:* The original design of the filter called for a matched filter that would have the same shape as the pulse given by figure 19.

*2) Hann Filter:* Because a GMSK pulse would not give the frequency response that we were looking for with the low-pass filter, we decided to switch to a Hann filter, as pictured in figure20. This filter was originally designed for an ADC working at 5 KHz, not 50 KHz, but it was later realized that this would make frequency recovery very difficult.
We designed the filter for an order of 8. This filter was fully implemented in hardware in simulation and gave the

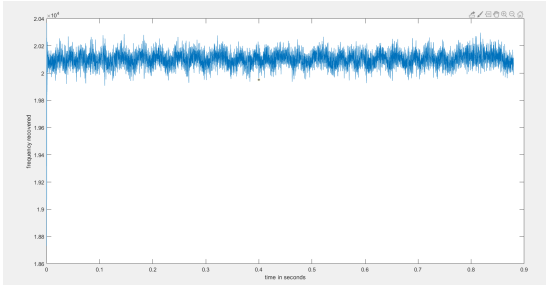


Fig. 15. Frequency recovery for f = 20100 Hz

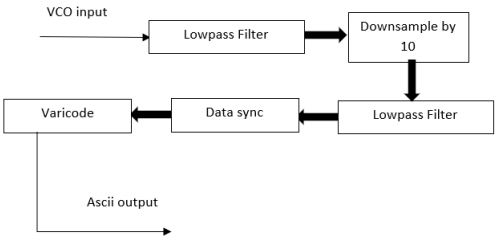


Fig. 16. VCO to ascii circuit

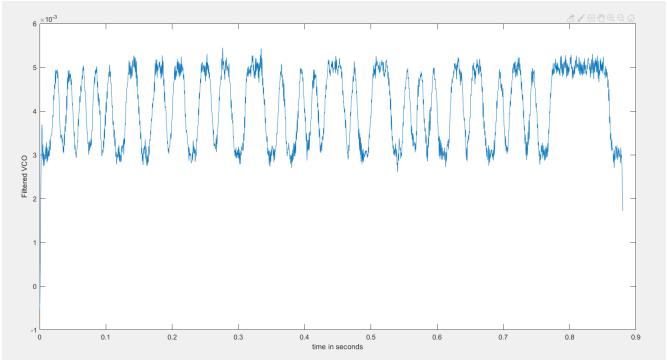


Fig. 17. Filtered VCO for $f = 20500Hz$

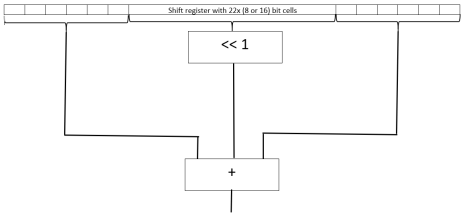


Fig. 18. Diagram of our 22nd order filter

same results as it gave in matlab, so was considered ”done”. However, while functional, it was actually a rather poor filter.

*3) Problems with the Hann filter:* First we plot the time domain in-phase wave function at 200 KHz in orange and the samples taken on this wave function with a purple scatter plot in figure 21. At this point the in-phase is still very accurate, but it does have its own problems with the fact that the data it is taken from has a high frequency component. When we applied our digital filter to the data in the simulation, the result we received was the one given in figure 22. Even worse, to save area on our chip, we decided not to implement a real FIR filter. Instead it simply added the last 8 values together, weighted by the coefficients of the Hann filter. This allowed us to filter as well as demodulate. The frequency plot of our input signal is given by figure 23.

*4) A very poor window:* The solution to the problem with the Hann filter was to find a filter that would suppress the frequencies at 350 Hz, as shown by figure 23. Originally, a new filter which still had an order of 8 was tried to achieve the required specifications. We did this because we were reluctant to redesign working hardware. Although it was possible to design such a filter that would heavily suppress the areas around 400 Hz, it was not possible to do both of these without very high side lobes in other parts of the frequency spectrum. If the demodulation resulted in a unwanted frequency component at 800 Hz, this would only have been suppressed by approximately 5 dB. Our ”window” filter was also almost inversely shaped to achieve the resolution to be able to suppress 350 Hz (which in a signal sampled at 5000 Hz is close to zero). The window had its lowest values in the middle and its highest values at the edges, effectively being a ”worse” filter than a unity filter would be when it came to side lobes.

*5) Solution to the problems: a twenty-second order filter:* We then accepted that we would have to redesign our filter and went for a fifty order filter that would meet the specifications without overly large side lobes, using least squares design. Because when rounded down, 14 values on the left and right side of the filter were zero, we were able to remove those values in the hardware, and the end result was a very simple shift register of $22 \cdot 8$ bits, combined with an addition for all values and a shift left for the inner values. The filter frequency response is given by figure 24 and the effect on the signal is given by figure 25. The hardware implementation of the filter is shown in figure 18.

### C. Design of carrier recovery

*1) Cordic algorithm:* CORDIC is a hardware-efficient iterative method which uses rotations to calculate the sine and cosine of a given angle [4]. There are two ideas to achieve rotation. The first idea is that rotating the input vector by an angle $\theta$ is equal to rotating the vector by several smaller angles, $\theta$i (i=0,1, ...n) provided $\theta = \sum_{i=1}^{n} S_i\theta_i$. The second idea is that we can choose the small elementary angles $\approx tan(\theta_n) = 2^{-n}$. In this way, multiplication by $tan(\theta_n)$ can be achieved by a bitwise shift which is a much faster operation. A single iteration is defined by the following equation:

$$\begin{bmatrix} x_{(n+1)} \\ y_{(n+1)} \end{bmatrix} = \cos(\theta_n) \begin{bmatrix} 1 & -S_n tan(\theta_n) \\ S_n tan(\theta_n) & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix}$$

At the same time, we introduce a new variable Z, which represents the part of the angle $\theta$ that has not been rotated

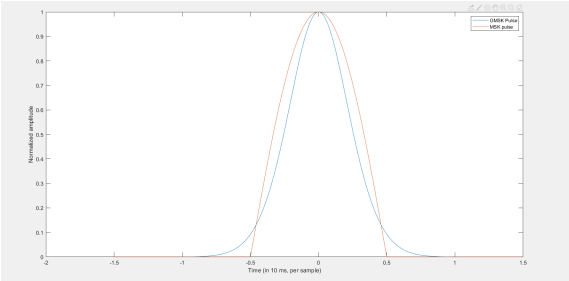


Fig. 19. GMSK and MSK pulse plotted over time

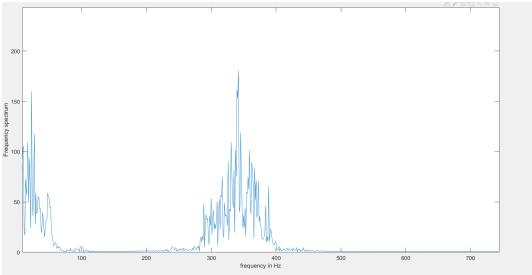


Fig. 23. Frequency of the input In-phase Response

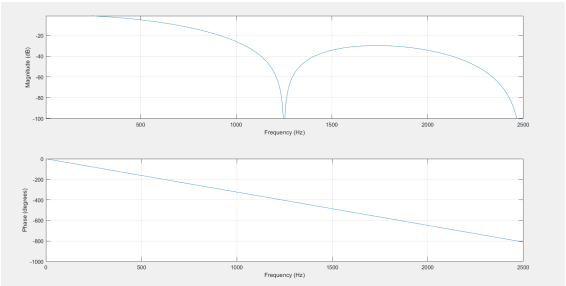


Fig. 20. Hann filter frequency response

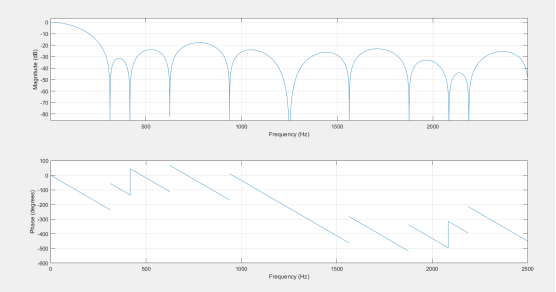


Fig. 24. Frequency plot of the filter

yet. If Z is larger than 0, then S=1, otherwise S=-1. The $cos(\theta_n)$ coefficients can be pre-calculated for all values of n. In hardware, we give input $X_i = K$, $Y_i = 0$ and $Z_i = \theta$ and then calculate $cos(\theta_n)$ and $sin(\theta_n)$.

Since the bit width is 16, the most efficient amount of pipeline stages is 15. To be able to process all angles, we use the code that we implemented to return the angle of the first quadrant, and rotate the results as required. This is illustrated in the previous table. To reduce the quantization noise, we increased the bit width of internal values to 20 bits.

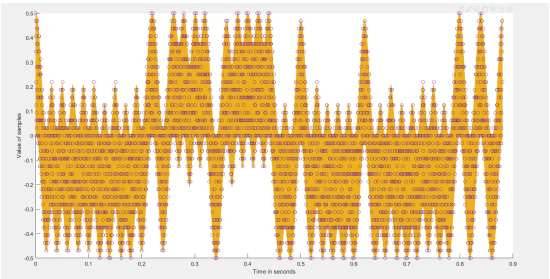


Fig. 21. Sampled In-phase Response

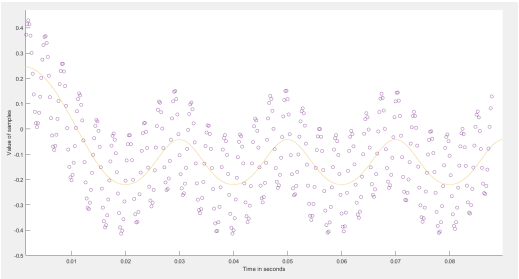


Fig. 22. Filtered zoomed-in In-phase Response

In hindsight, this was too much as we really only needed 8 bits to do our frequency recovery.
Another error occurred when the angle was close to 0 or 90 degrees. In this case, sin($\theta$) or cos($\theta$) should be 7FFF but the result was 8000, which is -1.000 instead of 1.000. The solution is setting the actual output to 7FFF when the initial output is 8000.

### *D. Design of clock recovery*

Recovering the VCO signal was not enough: for every 10 ms we would have to determine whether the signal was positive or negative. This is why a data synchronization module was required. The structure of our original design [5] is shown by figure 26.

The original design was slow to converge and rather complicated, so the final design (see figure 27 and 28) was much simpler and consisted of 3 counters:

1) cnt: increments on new value, returns done @ 10ms
2) count0: increments on a positive value
3) count1: increments on a negative value

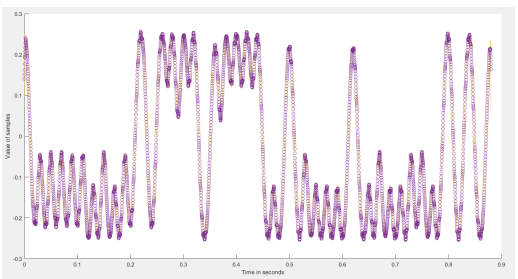


Fig. 25. Time response of filter in-phase

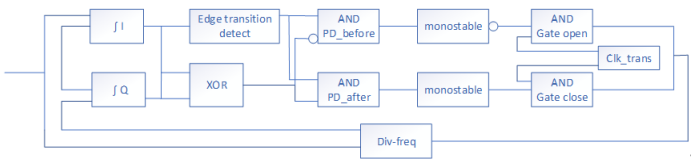


Fig. 26. Structure of Digital phase lock loop

By counting the number of positive and negatives values we could determine which one ”dominated” in a certain period of 10 ms. If the ”non-dominant” counter started ticking up during the period of a ”dominant” counter, we would speed up or slow down cnt.

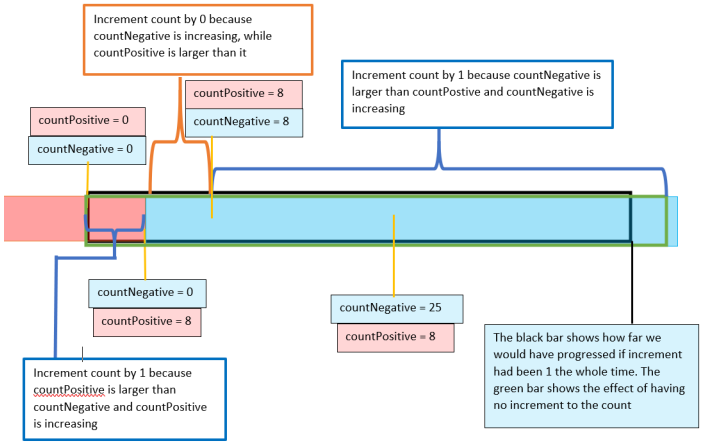


Fig. 27. Data sync

## III. Results and Resources

### A. Resources

Our full design, including demodulation from 20 MHz to 50 KHz used the following resources:

| Logic | Used | Available | Utilization |
|---|---|---|---|
| Slice Registers | 2968 | 4896 | 60% |
| total 4 input LUT’s | 3813 | 4896 | 77 % |
| RAMB16’s | 1 | 12 | 8 % |
| MULT18X18SIOs | 3 | 12 | 24 % |

Excluding the demodulation reduced the resources to the following:

| Logic | Used | Available | Utilization |
|---|---|---|---|
| Slice Registers | 2570 | 4896 | 52% |
| total 4 input LUT’s | 3661 | 4896 | 74 % |
| RAMB16’s | 1 | 12 | 8 % |
| MULT18X18SIOs | 3 | 12 | 24 % |

### B. Results

We edited the UART to plot the ascii as it came in. We placed the ascii outputs at the input of the uart in the hardware and modified the uart plotter script to convert the uart hex values into ascii text. The result can be seen in figure 29. It appears that some ascii values are duplicated. This might be caused by a done signal at the very end that occasionally triggers twice. We were still able to see the signals with an SNR of 30 dB in matlab, but we were not able to test this in hardware.

```
LEEVEL: 2
SECRET: NpaaKKZllYYY3DdmIrnytTddzbww8vAv33zLCkw


"The aim of science is to make difficult things u
nderstandable in a simpler wayy; the aim of poetr
y is to state simple things in an incomprehensibl
e way. The two are incompatible.."
- Paul Dirac

----------
```

Fig. 29. Decoded uart message

## IV. Conclusion

In conclusion we managed to achieve a low power analog circuit with adaptive gain and accurate analog-to-digital conversion. With our digital circuit we are able to decode a digital message with an SNR of 42 dB using not much more than a Costas loop (3 multipliers + 2 low-pass filter blocks with 22 adders each) and a look up table for varicode.

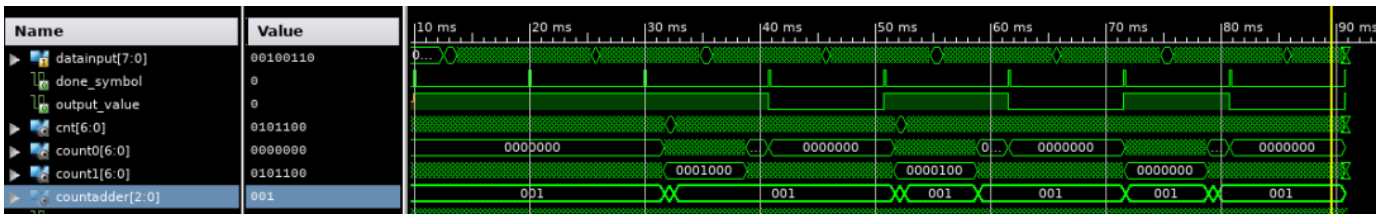


Fig. 28. Data sync in simulation. To slow down we change the countadder to ”000” which results in ”cnt” not being incremented.

## REFERENCES


[1] E. Sail, M. Vesterbacka, *"Thermometer-to-binary Decoders for Flash Analog-to-digital Converters"*, in Proc. Of ECCTD 2007, pp. 240-243, 2007

[2] B. Razavi, *"The StrongARM Latch",* IEEE J. Solid-State Circuits, pp. 12–17, Jun. 2015.

[3] R, Roshna and Rajendra, Nivin and Joy, Sherly and J, Apren and V., Alex *"Design and implementation of digital Costas loop and Bit synchronizer in FPGA for BPSK demodulation",* in ICCC 2013, pp 39-44, Dec. 2013

[4] Richard Herveille, *"Cordic Core Specification",* Dec. 2001

[5] Huimin Duan, Hui Huang,and Cuihua Li, *"Improved Design of Bit Synchronization Clock Extraction in Digital Communication System",* July. 2018