Low-Power Analogue Receiver ASIC for Space Telecommand Applications

S. Chicca[a], F. Bigongiari[a], G. Piscopiello[a], G. Tuccio[a]

[a]SITAEL, 56122 San Piero a Grado (Pisa), Italy

amicsa2014@cern.ch

Abstract

In this paper an ASIC implementation of an analogue receiver chain for telecommand applications for Category A missions (Return-to-Earth, lunar and even Lagrangian missions) will be presented. More specifically, in addition to the Low-Density Parity Check (LDPC) 128 bit analogue decoder component, the ASIC receiver will also include other important blocks of the telecommand reception chain normally accomplished inside an FPGA device, such as IF coherent demodulation stage front end, carrier recovery, baseband clock recovery, data conversion from input SP-L signal to NRZ codify, Start Frame pattern recognition, analogue memory for input codeword storing. The ASIC is now in detail design phase and it will be manufactured in XFAB 0.18um technology.

I. Introduction

Nowadays, on-board telecommand receivers for space applications consume an important percentage of the overall offered power of the satellite, especially due to their always ON need of operations.

Decoders, in charge of elaborating received data and of providing error correction according to redundancy introduced by related encoding protocol are one of the fundamental components of a satellite receiver. They currently follow a digital development approach based on a large FPGA. Even though, initially, the power consumption of digital decoders was not a factor of concern, the increasing communication and data storage complexity and capacity makes the applicability of error correction codes in a digital domain more and more expensive in terms of hardware resources and power consumption.

Therefore, in the last ten years, being analogue decoding recognised for its potential to efficiently decrease the overall power consumption of a receiver, an important growth in analogue decoding research programs is registered, although only a few VLSI integrated circuits have been developed satisfying a given communication standard. Analogue domain implementation of error correction codes, despite its lower power consumption potential with respect to its digital counterpart, seems to also provide some additional advantages: it takes benefit from the similarity between the mathematical operations required by the algorithms and the physical laws governing the circuit; it improves the total system efficiency, because the analogue decoder is much smaller than its digital counterpart and consumes about one order of magnitude less power at the same frequency; it offers high modularity design more immune to noise, by means of differential operation; it offers the capability of providing a finer estimation of the logic state of a single information unit with respect to digital implementations (no quantization); it needs a lower signal to noise ratio to properly correct a wrong input sequence of information unit.

Thus, the proposed paper will address such benefits of an on-board analogue receiver chain implementation for telecommand applications for Category A missions (Return-to-Earth, lunar and even Lagrangian missions.

In particular, Low-Density Parity Check (LDPC) 128 bit analogue decoder have been chosen as the design basis of the analogue decoder, since it showed in preliminary investigations a big potential for increased power saving when short length codes are concerned as of telecommand communication for Category A missions. Moreover, the analogue receiver will be compliant with the communication protocol described in ECSS-E-ST-50-04C “Telecommand protocols synchronization and channel coding”.

The paper is organized as follow: in the Section II the ASIC architecture description has been reported whereas details about the receiver, symbol synchronization, codify block and start frame recognizer have been reported respectively in sections III, IV, V and VI. The decoder core organization has been described in section VII and, finally, the conclusion has been reported on Section VIII.

SITAEL S.p.A. has produced all relevant work in the frame of “RLP_AD: Receiver Low-Power Analogue Decoder” activity (ESA TRP), developed in the context of ESA ITT AO/1-6722/11/NL/GLC with the aim of investigating feasibility of analogue decoding for space applications.

II. Architecture Description

The ASIC Block Diagram is reported in Figure 11.

The Serial Programming Interface (SPI) accepts external customized commands for proper internal bias and for test modes configuration.

The analogue input to the ASIC is the intermediate frequency DC component and the sidelobes of the modulated signal: the in-phase and the in-quadrature components of baseband signal are produced internally to the ASIC, after the final down conversion step, and they are filtered and amplified inside the AFE (Analog Front End) block.

The CR (Carrier Recovery) section provides to the local mixer a suitable frequency (same as carrier frequency) and phase (such that maximizes the In-Phase baseband demodulated component) for down conversion purposes: carrier tracking proceeds by extracting the carrier frequency to be tracked by the PLL from the received signal, and by

aligning the tracking-carrier phase according to a signal-energy-maximization principle.

In details, carrier phase tracking is achieved by correcting the PLL phase based on the measured amplitude inside the demodulated channels: the aim of phase correction is to re-phase the oscillation output by the PLL and used for local mixing (demodulation) in order to maximize the in-phase signal energy with respect to the in-quadrature signal energy. Downstream circuitry accepting the demodulated signal operates only on the in-phase demodulated baseband signal.

The SSU unit (Symbol Synchronization Unit) operates the clock extraction from the baseband PCM/PM/BI-PHASE signal: a local PLL tracks the data transition synchronism (since SP-L codify guarantees always at least one transition per clock cycle). Clock recovery is operated in the digital domain, that simplifies the PLL configuration. Due to the digital configuration adopted for clock recovery, ad hoc matched filtering must necessarily be provided inside the Codify Conversion block.

The CC (Codify Conversion) section converts the data codify from PCM/PM/BI-PHASE to NRZ codify. The PCM/PM/BI-PHASE to NRZ conversion task is performed by differentiating the left and right symbol half integrations with respect to the useful clock edge of data transition. Integration provides matched filtering and noise averaging.

Finally, the decoder block presides to data decoding and provides output NRZ digital decoded data and clock for sampling: it is able to trigger when a Start Frame pattern is recognized and it is able to stop decoding once an End Frame command is detected, waiting for next Start Frame pattern.

Being the signal processing chain of ASIC a fully concatenated-cascaded chain, several test modes are foreseen in order to allow verification of single functional sections.

III. Receiver Section Description

The Receiver Section is composed by AFE and CR block. The block diagram is reported in Figure 1.

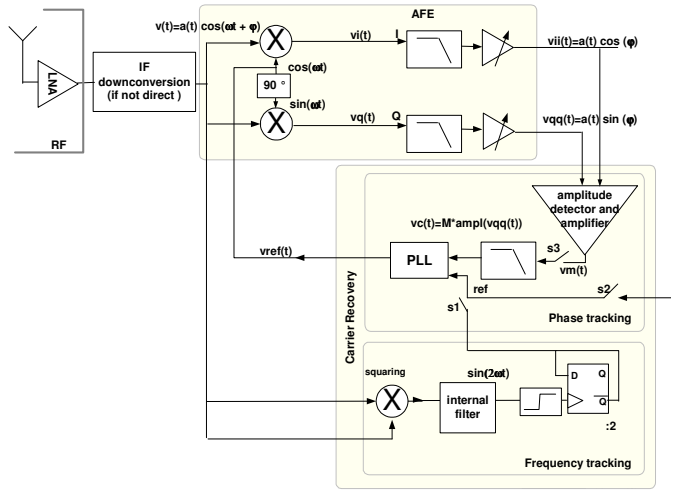

Figure 1: Receiver Section Block Diagram.

The carrier frequency recovery (which mixer works at 8MHz of IF) can be operated in two (2) different ways:

a. by a squaring and frequency divider by two, which sends a reference reconstructed frequency to the PLL (with switches s1 closed, s2 and s3 open). This configuration is a standard frequency recovery approach for BPSK suppressed-carrier signals;
b. by using the carrier-only transmission phase CMM1 of PLOP2 procedure [1] to drive the PLL output frequency in the neighbours of carrier frequency (s1 initially closed, s3 open, s2 open), and, once the PLL is in lock, by leaving the internal VCO being controlled by the amplitude detector and amplifier block (s3 closed, s1 and s2 open). The training frequency during CMM1 phase can be optionally provided by a local oscillator (s2 closed, s1 and s3 open). s3 and s1/s2 switch activity is mutually exclusive in time, and it is governed by the PLL lock signal. A dedicated internal configuration selects between s1 and s2 training options.

Carrier phase tracking is achieved by correcting the PLL phase based on the measured amplitude inside the demodulation channels: the aim of phase correction is to re-phase the oscillation output by the PLL and used for local mixing (demodulation) in order to maximize the in-phase signal energy with respect to the in-quadrature signal energy. Downstream circuitry accepting the demodulated signal operates only on the in-phase demodulated baseband signal.

The amplifier and filter inside the AFE chain condition the down-converted signal and filter the out-of-band signal components.

The AFE block has twofold functionality: to produce the final signal down-conversion for demodulation purposes on both I and Q signal components by using properly re-phased local reconstructed carrier and to amplify and filter the baseband demodulated I and Q signals. For this purpose, continuous time bi-quad filters are used in order to reject the image frequency components. The DC-rejecting amplifiers are used in order to reject the DC component at amplifier input.

IV. Symbol Synchronization Unit Description

The Symbol Synchronization Unit is composed by a Data Transition Detector and a Clock Recovery Unit.

The Data Transition Detector Unit is a simple comparator whereas the Clock Recovery Unit is a PLL like the one shown in Figure 2 which receives the 1-bit quantized data information and locks its oscillating frequency to the data transition frequency.

The clock recovery function extracts the clock information from the demodulated symbol data stream in order to allow correct signal sampling inside the matched filter and decoder downstream sections.

The main drawback of employing linear phase detectors in the PLL loop is that the phase detector response strongly depends on transition density, that has an heavy impact on clock phase jitter.

In PCM/PM/BI-PHASE signals (Figure 3), transitions are allowed to happen or synchronously with the data clock (in a string of all “0” or in a string of all “1” logic signal) or every

clock rising edge (in a string of alternating “0” and “1” logic signal).

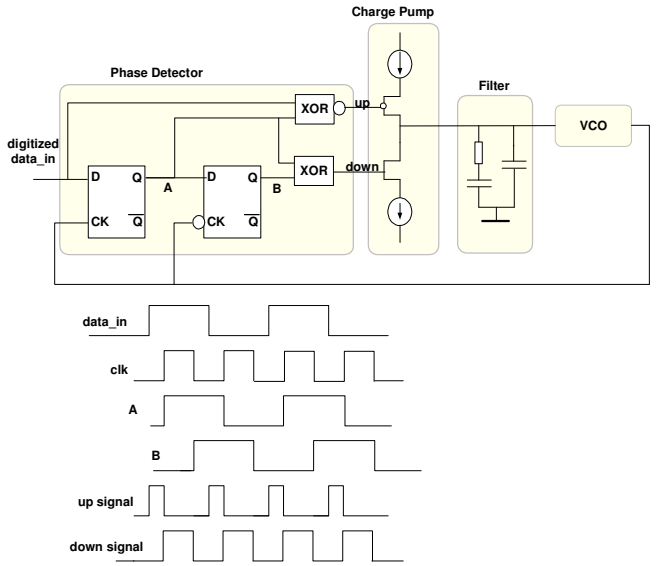

Figure 2: signal activity and timings for the Hogge’s phase detector. Retimed data are present at the output of lower flip flop.

Since there is a factor two between the transition frequency corresponding to the two above different cases, the transition density dis-uniformity is important: as an effect, the equivalent phase detector gain walking causes an important dispersion in the reconstructed clock phase.

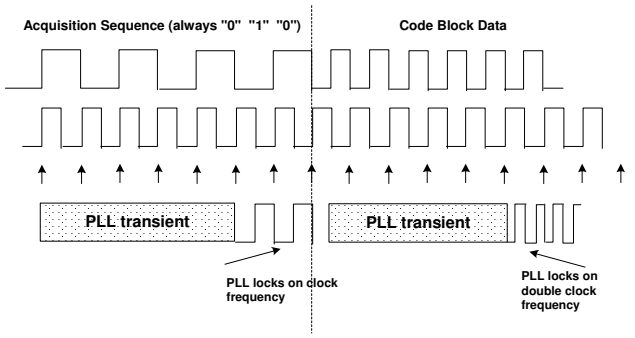

Figure 3: PLL acquisition sequence.

To overcome this issue, a principle has been applied, which equalizes the transition density and takes benefit of the property of PCM/PM/BI-PHASE signals of presenting always a data transition at the significant lock edge.

In practice, a PLL employing a standard linear phase detector is adopted using a certain (programmable) frequency divider in the feedback chain: the high-frequency clock at the input of frequency divider (VCO output) is used as a synchronism in order to mask the undesired transitions from the input data sequence, and to equalize data transition density over time.

After the PLL locks during the data Acquisition Sequence, the introduced synchronism allows to keep the transition density constant over time after the Acquisition Sequence ends and code block reception starts (with related data transition density discontinuities).

This mechanism allows the phase detector to operate with a fixed gain, greatly improving the performances of the PLL itself and reducing clock frequency/phase jitter.

The synchronism signal used to equalize the data transition density practically masks the extra-transitions not synchronous with the clock rising edge in the data stream, according to the principle reported in Figure 4.

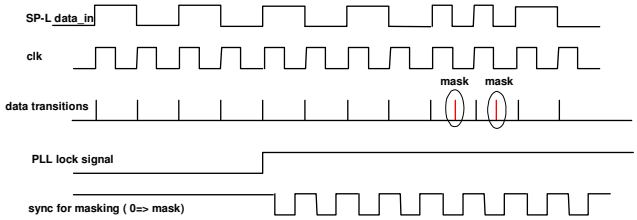

Figure 4: Masking data transitions operation to equalize data transition density over time.

As a result of masking operation, the data transitions are only the useful data transitions at symbol middle point individuated during the Acquisition Sequence, and any other data transition (used in PCM/PM/BI-PHASE codify for properly set upping data level before the symbol middle transition) is discarded and not used for PLL phasing.

In details, the symbol duration is divided into 4 equally spaced quadrants during the Acquisition Sequence: once the PLL locks and after Acquisition Sequence ends, this quadrant-spacing is kept for each symbol and data transitions happening within the second and the third quadrant are systematically masked.

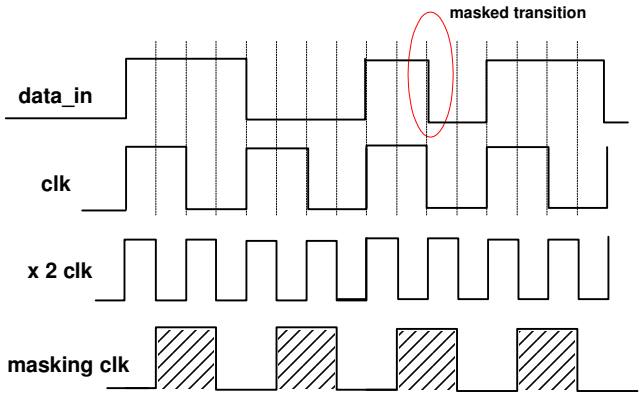

Figure 5: Clock multiplying internal to PLL allows an easy data transition masking principle.

Masking operation is enabled only if the PLL is locked during the Acquisition Sequence and the locking status is detected during the Acquisition Sequence when the transitions passible of masking are not detected for a certain number of clock cycles.

The masking mechanism provides a systematic solution to the transition density uncertainty proper of PCM/PM/BI-PHASE signals. Another mechanism is part of the PLL design to improve the PLL noise tolerance. This different mechanism detects the missing data transitions and inserts transitions into the data stream fed to the PLL in order to maintain the PLL in lock condition.

By summarizing, the strategy for the clock recovery is:

a. during the Acquisition Sequence to achieve the lock starting from the first data transition, a timeout corresponding to the expected symbol rate inserts artificial transitions when missing transitions are detected by the timeout in the data stream;

b. once the lock is achieved, the masking mechanism is enabled: in this way, data transition corresponding to the clock edge not representing the SP-L are systematically masked to the PLL, in order to avoid PLL frequency walking according to the data transition density.

Both mechanisms are aimed to provide to the PLL a constant data transition density in order to maintain the lock.

The lock status of clock recovery PLL is used to enable downstream signal processing chain, in details the Start Frame Recognizer

The data rate change is achieved internally to the clock recovery circuit by properly programming (through SPI) the frequency divider that is used inside the PLL loop. This frequency divider allows obtaining six (6) binary progressively increasing data rates from 8Kbit/s to 256Kbit/s.

V. Codify Conversion Description

Once the data clock is extracted, correct data sampling is possible. However, soft values are required to be passed to the analogue decoder by avoiding any squaring operation on analogue levels before decoding: on the contrary, an analogue level has to be stored inside the decoder memory representative of the log-likelihood probability of corresponding bit. Hence, this representative level has to be constructed and sampled. Sampling analogue could be affected by superimposed noise; in fact considering at the moment NRZ signals, if at sampling instant a large noise spike is superimposed to the signal, the resulting sampling is affected by that noisy sample. If, instead, the symbol level is subjected to a continuous integration operation during the symbol period, the sampling of the result at symbol end will take advantage from the integration, allowing an effective noise filtering and a signal-to-noise ratio maximization.

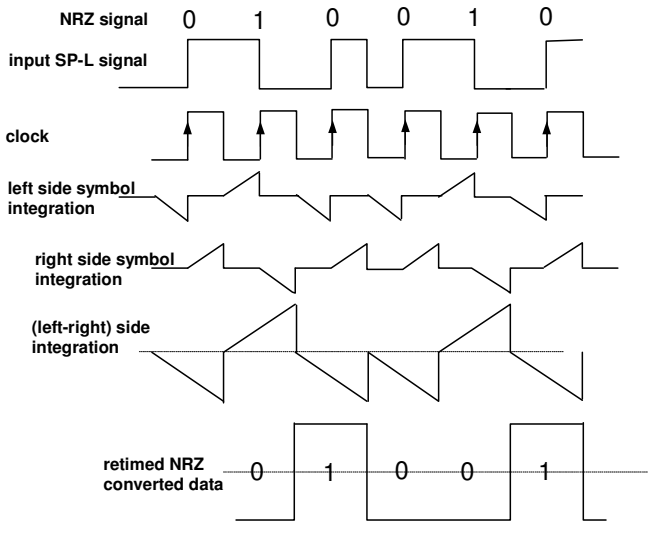

Figure 6: Conversion of PCM/PM/BI-PHASE signal into NRZ signal by averaging noise on half symbol periods and by measuring the level jump related to received logic state at clock edge.

In PCM/PM/BI-PHASE signals, the logic state information resides in the sign of level jump experienced at clock rising edge: if the data stream presents a positive jump, it is associated with the reception of a logic "0" whereas if the data stream presents a negative jump, this is associated with the reception of a logic "1".

As a conclusion, averaging noise on logic states when codified according to the PCM/PM/BI-PHASE convention means producing the differentiation between the integration results of right-symbol side and left-symbol side (Figure 6): the differentiation result is sampled on the opposite clock edge with respect to data transition in order to provide the NRZ data conversion.

VI. Start Frame Recognizer and Data Memory

The input data stream pattern, to be recognized in order to allow decoding operation starts, is composed by 16-bit logic pattern 1110101110010000; however the programmability of the target pattern can easily be achieved through SPI interface. For pattern recognize a sliding window approach has been used. The incoming level is stored bit-by-bit into a 16-bit analogue First-In First-Out chain and each value is compared with the 16 values corresponding to the pattern to be recognized; the 16-bit correlations between received and expected bit are summed too (Figure 7).

If the correlation result exceeds a certain (externally programmable) correlation threshold, the Start Frame pattern is recognized, and the START flag is asserted for decoding operation and it will remain asserted until an End Sequence will be recognized. Multiple triggers caused by possible recognition of a Start Frame sequence during normal decoding operation has been carefully avoided.

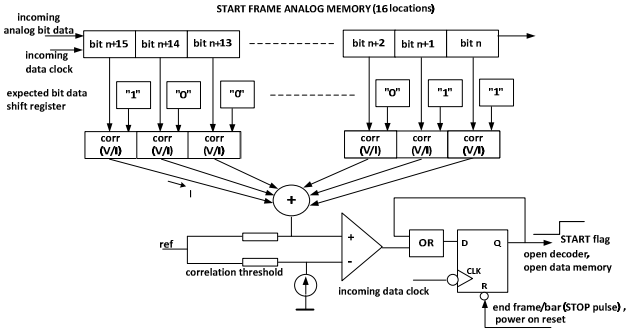

Figure 7: Acquisition Sequence Recognizer functional description.

The input memory is in charge of storing the incoming serial input analogue values upon each clock cycle into a corresponding memory cell, before parallel feeding the values to the decoder core for elaboration.

In principle, after the input values are stored inside the memory, they are fed all simultaneously to the decoder core for parallel decoding operation. With this approach, a "dead time" in input data throughput should be observed between the reception of two consecutive words caused by the decoding time. This constraint, although tolerated by a demonstrator, becomes unacceptable in real applications, where data throughput is in general continuous and a double buffer strategy seems better responding to the requirements.

Double buffering allows to fill one memory with incoming analogue values while the second memory is being redirected in parallel to the decoder inputs: in front of the cost of an additional memory, data throughput is allowed to be continuous, and the period available for the decoder core to converge into the decoded sequence is the whole period needed for receiving a single (next) encoded word, with a single word latency. This last property of double buffering relaxes the decoder speed requirements and hence the power consumption requirements.

The memory organization is reported in Figure 8.

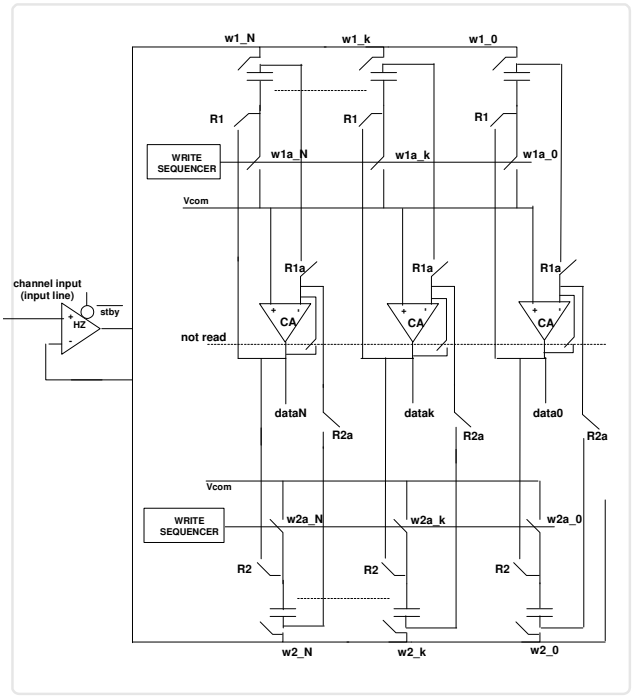

Figure 8: Double buffer memory structure.

Two arrays of capacitors, building the memory storing elements, are alternatively charged to store the incoming word values. A single input buffer provides the current capability for charging the quite high-capacitive writing line (due to the long tracks stray capacitances): the capacitors are charged in sequence, according to the switches *"w1_k/w1a_k"*, whose timings are synchronous with the input signal sampling operation.

While one of the memories is written sequentially with the incoming analogue values, the other one is read-out in a block and its values are fed in parallel to the analogue decoder core; during the read operation, each memory capacitor is connected as the feedback element of the column amplifier (CA), which provides an unitary replica of the input voltage stored in the capacitor element, referred to the common mode voltage. It is noticed that a single column amplifier is used which serves two different memory cells, read and written in different times: while one of the two memory cells connected to the unique amplifier is written (by storing a charge into the capacitor actually connected between the input line and the common mode reference), the column amplifier is in closed loop connection because it serves the other memory cell, so that its negative input terminal is never floating.

Finally, and for correctly resetting the decoder after each decoding operation, each column amplifier is connected as a buffer to produce an output level corresponding to the same common mode voltage for memory and decoder when no memory read operation occurs that is when a decoding operation is just finished and a new decoding operation is not yet started. It is noticed that, thanks to switches *"R1a"* and *"R2a"* in Figure 8, the resetting operation does not affect memory cells write operation.

With the purpose of using the above described reset phase also to discharge the memory capacitors before next memory bank write operation, the deactivation of read switches *"R"* and *"Ra"* of Figure 8 is delayed of half clock cycle with respect to the logic signals read. During this half clock cycle, the memory bank is still in its read phase, but the reset switch is closed, allowing the memory capacitor discharges (Figure 9).

In the following, some details about the optimum timing sequence for switches closing/opening are discussed with respect to charge injection issues.

The charge in steady state condition for each capacitance when the writing switches are both closed is determined basically by the signal-to-common-mode level, and both the input channel buffers and the common mode buffer driving the line *"Vcom"* in Figure 8 are low impedance nodes.

When the write operation ends for a single capacitor, the switch *"wa"* are open, whereas switch *"w"* remain in the low impedance state. Since the switch *"wa"* has a null voltage at its terminals, the charge injection expected by its turning OFF is a constant amount, not signal depending and this means that a systematic offset is added to the useful signals but no signal distortion is introduced by charge injection phenomena. Selecting a common mode voltage close to half the power supply rail and by implementing the switch as a carefully-sized complementary-transistors pass gate, the positive charge injection related with the pass-gate PMOS opening (which causes the bottom plate voltage of memory capacitor raises) is compensated by the equivalent negative pass-gate NMOS opening charge injection, thus leading in compensating effects.

During the read operation, still the closing sequence of switches *"R"* and *"Ra"* has poor relevance, because, until both switches are closed, no charge injection can occur: on the other hand, in steady state conditions, the memory capacitor terminals are respectively the CA (virtually to ground forced to *"Vcom"* level by the amplifier gain) and the CA low-impedance output node. When the read operation ends, first *"Ra"* switches open and as previously noticed for the write operation, the charge injection related with *"Ra"* switches opening is a constant amount of charge, not signal depending, which can be compensated for by adopting careful pass-gate design for the switches; of course, once *"Ra"* switches are open, no charging of capacitor can happen by the subsequent opening of switches *"R"*.

The most expensive power consumption inside the memory block is imputed to the line buffers because they have to provide the memory-capacitors charging/discharging currents within a single clock period, whereas the column

amplifiers settling time is drowned in the longer decoding time.

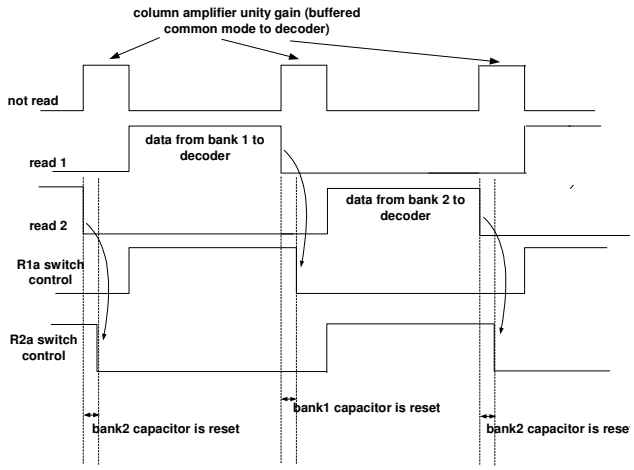

Figure 9: Timings diagram for decoder reset and memory bank capacitors reset on delayed read phase end.

VII. DECODER CORE ORGANIZATION

The last element of the receiver chain is the Decoder. According to ESA interest in LDPC short length code block codes for telecommand applications ([1] and [2]), the decoder implements the min-sum algorithm on the code LDPC (128, 64) referred in [2], which uses 64 check nodes, 8-bit-complexity each.

The mathematical steps of min-sum approximation approach to LDPC decoding and the main architectural results have been summarized in the general schema of Figure 12 and Figure 13.

The whole decoder configures as an asynchronous analogue network, built basically by two types of macro-cells (variable and check nodes), in which the decoding law is established by the interconnections between the cells at routing level, virtually programmable by metal mask option.

The iteration principle of digital implementation for solving the check law equations by progressively converging estimations is substituted by a continuous back-connection of each estimation-process-output at the estimation-process-input that forces the network to find its equilibrium "final" point estimations that satisfy the decoding law. Iterations and related overclocking with respect to input data throughput are hence suppressed, together with the need of checking the parity check matrix at each step in order to verify convergence.

By referring Figure 13, each variable node ("*qij*" level in the picture) calculates its own estimation, based on input log-likelihood data and based on data provided by all check nodes which are providing an estimation for that variable node, except the check node to which the variable node is just sending its own estimation.

Each parity check node ("*rji*" level in the picture) provides an estimation of each variable nodes afferent to it, by applying the parity check law (min-sum) to all other variable nodes (other than the one under estimation) afferent to it.

Both message forming processes (from the variable nodes to the check nodes and in the contrary direction) take place by adopting an *extrinsic information principle*: the information produced by a node (at its output) is never looped at its input to confirm itself but on the contrary, the information building process inside each node (variable or check node) always happens on the base of the information passed by different nodes.

The above observation is at the base of correct decoding principle: it may be noticed that each estimation of a single log-likelihood variable $\hat{c}i$ passed to check node *j* in Figure 13 is built by summing all estimations available for that variable (included the input log-likelihood level) except its estimation produced just by node *j*.

In the log min-sum approximation of the sum-product algorithm, the functionality of each variable node is log-likelihood probabilities summing (current sum), whereas the functionality of each check node is to select the minimum confidence (absolute value) of input log-likelihood probabilities and to assign it the expected sign for the check node output message.

Consequently, the basic processing analogue cells required for implementation are as follows:

- at variable node level, a voltage-to-current conversion must be operated to convert the input log-likelihood voltage levels into currents (the current sum can be easily performed afterwards by wiring currents together)
- at check node level, the minimum absolute value of input currents (disregarding their sign) has to be produced. To this purpose, each input variable to the check node is treated and de-composed in amplitude and sign (Figure 10): a looser takes all circuit coupled with a multiplexer (block "*select min (|u|)*") and it is used to select the minimum amplitude and the sign computation is performed by propagating the sign through XOR digital gates; finally, a reconstruction block is used to assign the proper sign to the minimum individuated for the output amplitude.

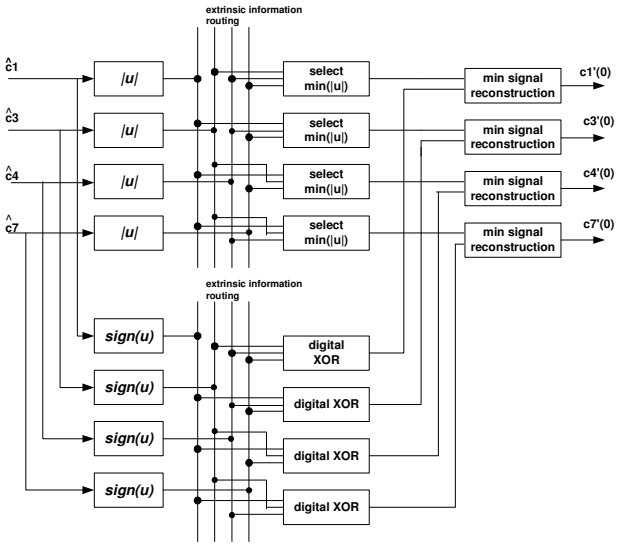

Figure 10: Details of the min-sum check node functionality.

The above operation performed by the check node finds its implementation in Figure 14, which reports the organization

of the basic 8-bit check node. The sub-block VN implements the functionalities of absolute value extraction and sign extraction of Figure 10, but it presides also to extract the a priori information and to combine this information with check nodes messages ($\sum$ operator of Figure 13). The LTA sub-block implements the minimum absolute value extraction and selection and the XOR sub-block implements the sign attribution to the check node estimation. Hence, for each input variable *i*, the schematic of Figure 12 selects minimum confidence and assigns logic sign to force parity-check law. In performing this operation, it applies an *extrinsic information principle*, basing its estimation about variable node *i* by relying only on other variable nodes (other than *i*) afferent to it.

Figure 15 shows the LDPC 128 bit decoder hardware organization.

VIII. CONCLUSIONS

In this paper have been presented a complete analogue receiver chain ASIC for telecommand applications for Category A missions (Return-to-Earth, lunar and even Lagrangian missions). Advantage and disadvantage of the analogue implementation respect to the traditional digital implementation based on FPGA have been presented. The blocks component the receiver have been described in detail and their functionalities have been analysed.

The receiver is now in detail design phase and it will be manufactured in XFAB 0.18um CMOS process.

IX. REFERENCES

[1] "European Cooperation for Space Standardization – Engineering – Space data links - Telecommand protocols, synchronization and channel coding", ECSS-E-ST-50-04C, Revision 2, 31/07/2008

[2] "Short Blocklength LDPC Codes For TC Synchronization and Channel Coding - Draft Recommendation for Space Data System Standards", (ref. CCSDS 231.0-O-x.x, Orange Book, April 2012)

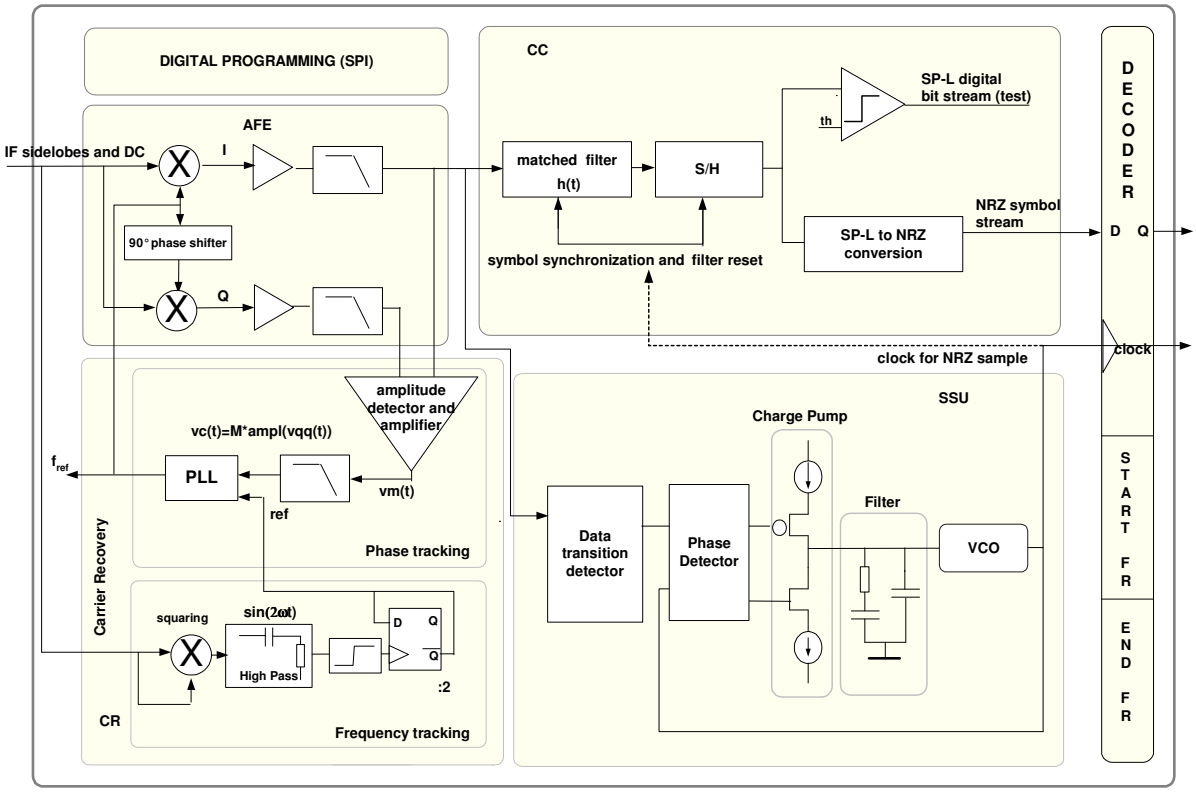

Figure 11: ASIC Block Diagram.

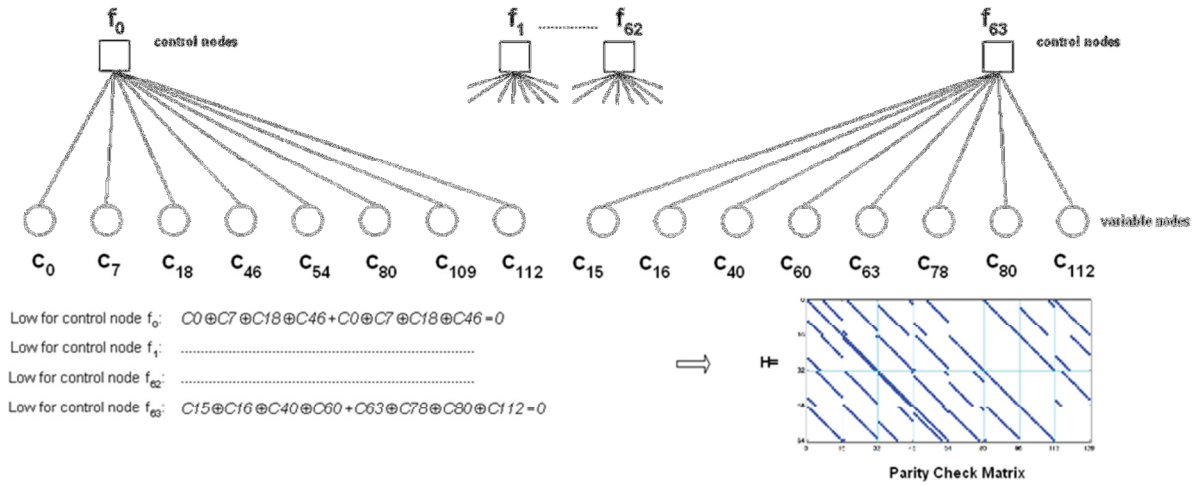

Figure 12: Check nodes for the referred (128, 64) code comprises 64 check nodes, each accepting systematically 8 bit from the variable node sequence.

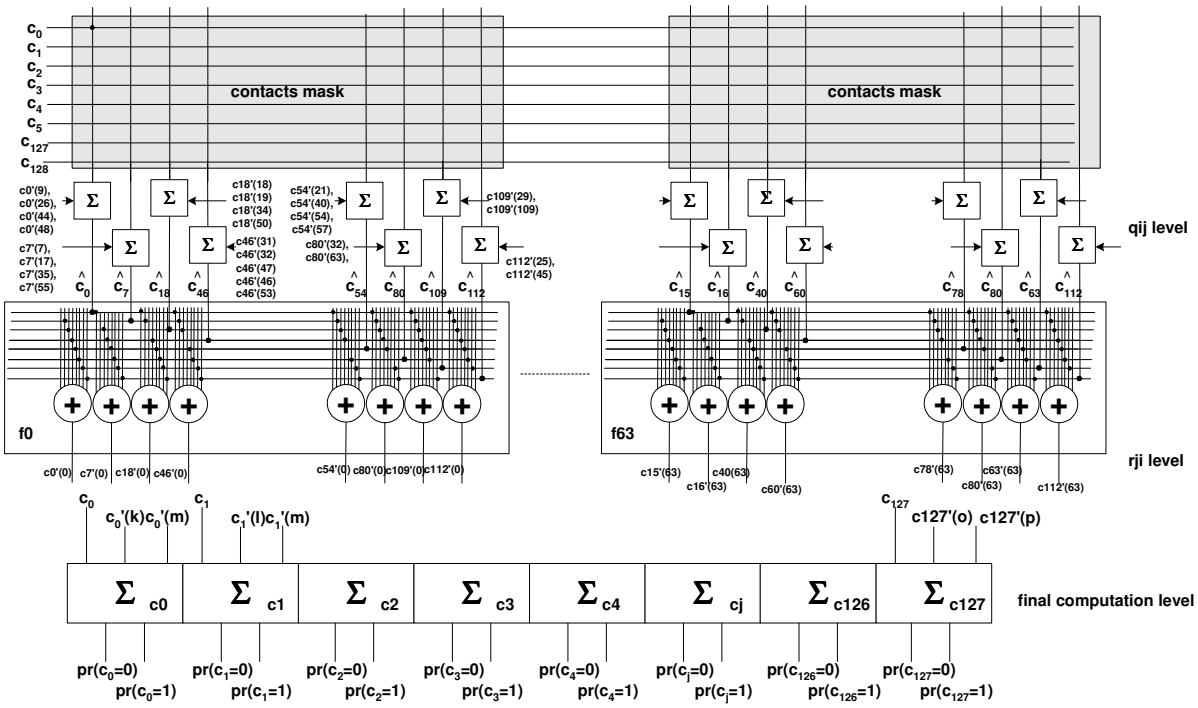

Figure 13: Low-level description of the min-sum algorithm applied to the parity check matrix of Figure 12

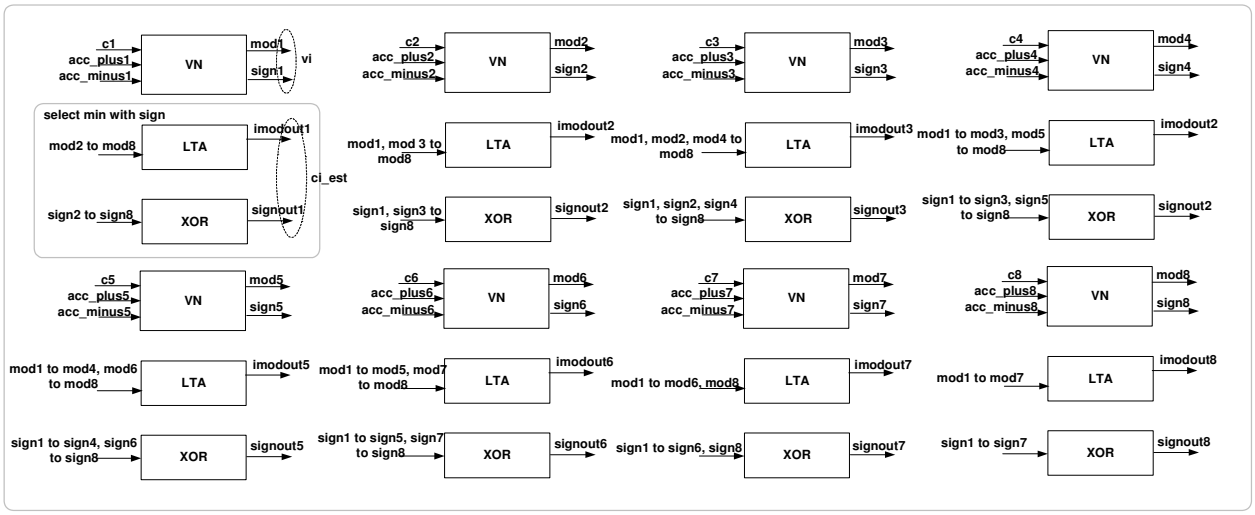

Figure 14: Basic 8-inputs check node cell.

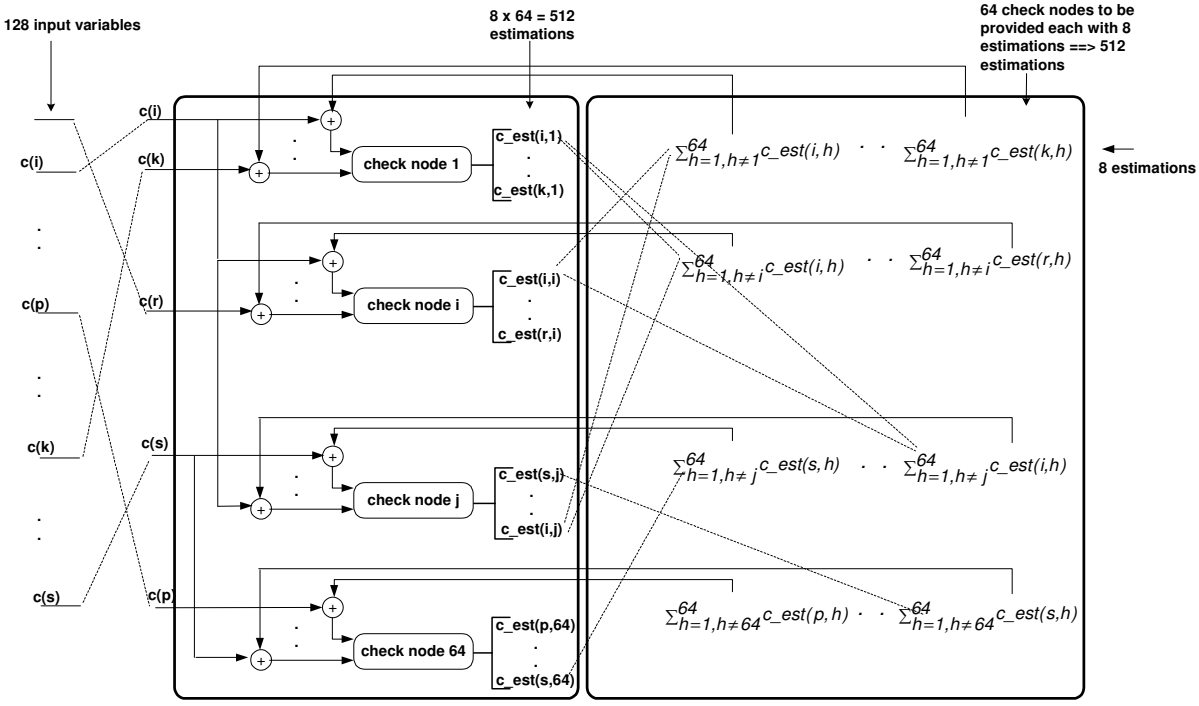

Figure 15: Decoder hardware organization: the left block is composed by the 64x8 inputs nodes and the right block is composed by 64x8 estimation adders.