Anatomy of contemporary GSM cellphone hardware

Harald Welte <laforge@gnumonks.org>

April 14, 2010

Abstract

Billions of cell phones are being used every day by an almost equally large number of users. The majority of those phones are built according to the GSM protocol and interoperate with GSM networks of hundreds of carriers.

Despite being an openly published international standard, the architecture of the GSM network and its associated protocols are only known to a relatively small group of R&D engineers.

Even less public information exists about the hardware architecture of the actual mobile phones themselves, at least as far as it relates to that part of the phone implementing the GSM protocols and facilitating access to the public GSM networks.

This paper is an attempt to serve as an introductory text into the hardware architecture of contemporary GSM mobile phone hardware anatomy. It is intended to widen the technical background on mobile phones within the IT community.

1 Foreword

This document is the result of my personal research on mobile phone hardware and system-level software throughout the last 6+ years.

Despite my past work for Openmoko Inc., I have never been professionally involved in any aspect of the actual GSM related hardware of any phone. Nevertheless I have the feeling that in the wider information technology industry, I am part of a very, very small group of people who actually understand mobile phones down to the lowest layer.

I hope it is useful for any systems level engineer with an interest in understanding more about how mobile phone hardware actually works.

There are no guarantees for accuracy or correctness of any part of the document. I happily receive your feedback and corrections.

2 Is your phone smart or does it have features?

Initially, for the first couple of years, GSM cell phones were actual phones with very little additional functionality. They provided everything that was required for voice calls, as well as SIM phone book editing features. The only additional non-features were simple improvements like the ability to use them as an alarm clock.

In the mid-1990s, a certain new type of devices became popular: The PDA (personal digital assistant). They pioneered handheld computing by introducing touch screen user interfaces and a wide range of application programs, ranging from calendar/scheduling applications, dictionaries, exchange rate and tip calculators, scientific calculators, accounting / finance software, etc.

While in mobile phones the actual cellphone aspect was becoming more and more commoditized, at some point the PDA features and functionalities were added to phones, coining the term *smartphone*. At that point there was a need to differentiate from those phones that were not-so-smart. Those phones were then called *feature phones*.

There has never been an industry-wide accepted definition of those terms, and especially in the late 2000s, feature phones started to inherit a lot of the functionality that was formerly only present in smartphones.

This document will define the terms (only for the purpose of this document) along a very clear border in hardware architecture, as will be described in the following sections:

2.1 Feature Phone

A feature phone is a phone that runs the GSM protocol stack (the software implementing the GSM protocol) as well as the user interface and all applications on a single processor. For historic reasons, this processor is known as the so-called *baseband processor* (BP).

The baseband processor often exposes a serial port (or today USB) over which the phone can be used as a terminal adapter, similar to old wireline modems. The industry standard protocol for this interface is an AT command set - extended and modified from how computers interfaced old wireline modems. The AT-command interface can be connected to a computer. The computer can then use the phone to establish data calls, send/receive short messages via SMS, and generally remote-control the phone.

2.2 Smartphone

A smartphone is a phone that has a dedicated processor for the GSM protocol stack, and another (potentially multi-core) general purpose processor for the user interface and applications. This processor is known as the *application processor* (AP).

The first hardware generations of smartphones did nothing else but to put the feature phone and a PDA into one case. The keypad and display of the baseband processor is removed. What remains of the feature phone is a *GSM modem*, controlled by AT commands sent from the AP.

Each processor has its own memory (RAM and Flash), peripherals, clocking, etc. So this setup is not to be confused with the symmetric multi-processor setups like seen in the personal computer industry.

Later generations of smartphones have exchanged the AT command interface by various proprietary protocols. Also, the serial line was replaced by a higher-bandwidth hardware connection such as USB, SPI or a shared memory interface.

Due to market pressure for ever smaller phones with ever more functions, the industry has produced highly integrated products, uniting the AP and BP inside one physical package. Further pressure on reducing cost and PCB footprint has led to products where there is no need to have independent RAM and Flash chips for AP and BP. Rather, a single RAM and Flash chip is divided by assigning portions of the RAM and Flash to each of the two processors.

However, the fundamental separation between the AP and BP, each with their own memory address space and software, remains present in all smartphones until today.

3 GSM modem architecture

Every GSM phone, feature phone and smartphone alike, has a GSM modem interfacing with the GSM network.

This GSM modem consists of several parts:

- RF Frontend, responsible for receiving and transmitting at GSM frequency
- Analog Baseband, responsible for modulation and demodulation
- Digital Baseband, responsible for digital signal processing and the GSM protocol stack

3.1 The RF Frontend

The RF Frontend is tasked with the physical receive and transmit interface with the GSM air interface (sometimes called Um interface).

It minimally consists of an antenna switch, GSM band filters, low-noise amplifier (LNA) for the receive path, power amplifier for the transmit path, a local oscillator (LO) and a mixer.

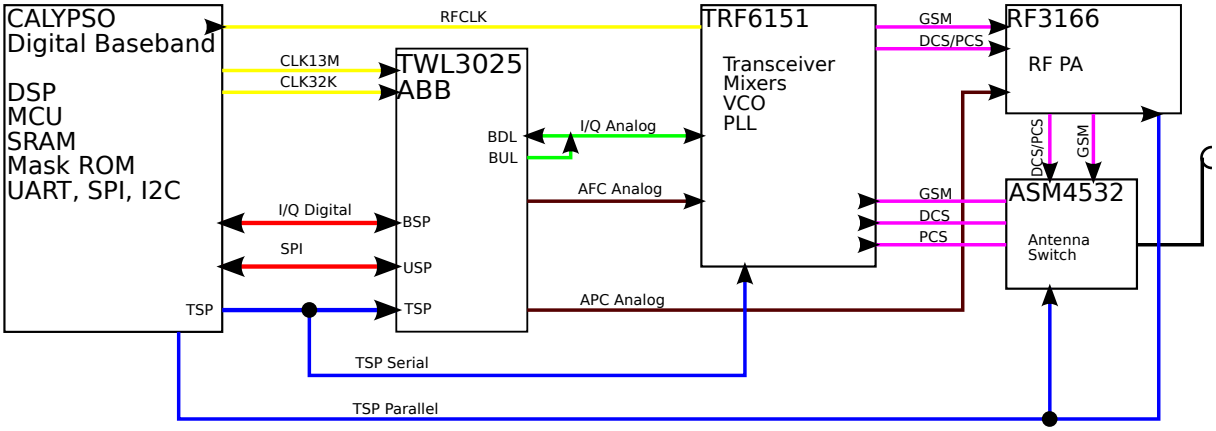

Figure 1: Block schematics of a TI Calypso/Iota/Rita GSM modem

By mixing the LO frequency with the received RF signal, it generates an analog baseband signal that is passed to the Analog Baseband (ABB) part of the modem. By mixing the output of the ABB with the LO frequency, it generates a RF signal that is to be transmitted in the GSM frequency band.

As the receive and transmit framing has an offset of 3 TDMA frames, there is no need for a frequency duplexer. Instead, an antenna switch is used. The switch typically is implemented using a MEMS or diode switch. For a quad-band phone, typically a SP

3.1.1 RF Frontend receive path

The antenna picks up the GSM radio signal as it is sent from a GSM cell (called Base Transceiver Station, BTS). The antenna signal first hits the antenna switch, which connects the antenna with the Rx path for the GSM band of the to-be-received radio frequency. It is then filtered by a bandpass to block out-of-band signals before entering a low-noise amplifier for increasing signal amplitude.

After passing the LNA, the RF signal is mixed with a frequency generated by the LO. Depending on the LO signal, either an intermediate frequency (IF) or a direct baseband signal is produced. In modern GSM modems, zero-IF designs with immediate down-conversion to analog baseband signals are most common.

The baseband signal is then filtered to remove unwanted images and sent as analog I/Q signals (representing amplitude and phase) to the ABB.

3.1.2 RF Frontend transmit path

The ABB generates analog I/Q signals, which are filtered and passed into the mixer, where they are mixed with the LO frequency and thus up-converted to the GSM RF band. From there, they are sent to the transmit amplifier (RF PA) for amplification. After amplification, they traverse the antenna switch and are transmitted by the antenna.

3.1.3 Local Oscillator

The LO of a GSM modem has to be synchronized very closely to that of the cell (BTS). To achieve the required precision, a Voltage-Controlled, Temperature-Compensated Crystal Oscillator (VCTCXO) is used.

Common frequencies for this VCTCXO are 26MHz or 13MHz, as the GSM bit clock (270,833 Hz) is an integral division (/96 or /48, respectively) of those frequencies. The tuning range of the VCTCXO is several kHz to compensate for temperature drift.

3.2 The Analog Baseband (ABB)

The ABB part of a GSM modem is responsible to interface between the digital domain and the analog domain of the GSM modem.

3.2.1 ABB Receive path

The analog baseband I/Q signals are potentially filtered again and digitized by and Analog-Digital converter (ADC). The sample clocks used are typically integral multiples of the GSM bit-clock. The sample clock itself is derived by dividing the VCTCXO of the RF frontend.

The digital I/Q samples are passed to the Digital Signal Processor (DSP) in the Digital Baseband (DBB). To reduce the number of traces to be routed on the PCB, the samples are typically sent over some kind of synchronous serial link.

3.2.2 ABB Transmit path

There are multiple architectures found in the ABB transmit path.

The obvious architecture is to do the inverse of the receive operation: Transmit digital I/Q samples from the DSP to the ABB and convert them into an analog signal that is then to be sent to the mixer of the RF Frontend.

However, sending a GSM signal with its GMSK modulation is much simpler than receiving. So in order to reduce computational complexity (and thus cost as well as power consumption) inside the DSP, the modulation of the bits is often performed in hardware inside the ABB.

In this design, the unmodulated GSM burst bits are sent from the DBB to a burst buffer inside the ABB. From there, based on ROM tables and a Digital-to-Analog converter (DAC), an analog GMSK modulated signal is generated.

3.3 The Digital Baseband (DBB)

The digital baseband implements the actual GSM protocols from Layer1 up to Layer3 as well as higher layers such as a user interface in the case of the feature phone. In a smartphone, the DBB only implements a machine interface to be used by the AP.

A typical DBB design includes a Digital Signal Processor (DSP) for the lower half of Layer1, and a general-purpose processor (MCU) for the upper part of Layer1, as well as anything above.

3.3.1 Digital Signal Processor

The choice of DSP architecture largely depends on the DBB chipset vendor. Often they already have a line of DSP cores in-house and will of course want to reuse that in their DBB chip designs. Every major DSP architecture can be found (TI, Analog Devices, ...).

The DSP performs the primary tasks such as Viterbi equalization, demodulation, decoding, forward error correction, error detection, burst (de)interleaving.

Of course, if actual speech data is to be communicated over the GSM network, the DSP will also have the auxiliary task to perform the computation of the lossy speech codec used to compress the speech.

Communication between the DSP and MCU happens most commonly by a shared memory interface. The shared memory contains both actual data that is to be processed, as well as control information and parameters describing what to be done with the respective data.

For the receive side, the MCU will instruct the DSP to perform decoding for a particular GSM burst type, after which the DSP will receive I/Q samples from the ABB, perform detection/demodulation/decoding and report the result of the operation (including any decoded data) back to the MCU.

For the transmit path, the MCU will present the to-be-transmitted data and auxiliary information to the DSP, which then takes care of encoding and sends the corresponding burst bits to the ABB (remember, most ABB take care of the modulation to reduce DSP load).

The detailed programming information (API) of the DSP shared memory interface is a closely-guarded secret of the baseband chip maker and is not commonly disclosed even to their customers (the actual phone making companies).

In doing so, the baseband chip makers create a close dependency between the GSM Layer1 software

(running on the MCU) driving/implementing this API and the actual baseband chip. Whoever buys their chip will also have to license their GSM protocol stack software.

It is thus almost impossible for an independent software vendor to get access to the DSP API documentation, which the author of this paper finds extremely anti-competitive.

3.3.2 DSP Peripherals

The specifications of the GSM proprietary On-air encryption A5/1 and A5/2 are only made available to GSM baseband chip makers who declare their confidentiality. Implementing the algorithm in software is apparently considered as breach of that confidentiality. Thus, the encryption algorithms are only implemented in hardware - despite them being reverse-engineered and publicly disclosed by cryptographers as early as 1996.

Thus, the DSP in a DBB commonly has a integrated peripheral implementing the A5 encryption.

Further integrated DSP peripherals may include a viterbi hardware accelerator, a DMA capable serial interface to the ABB and others.

3.4 Baseband Processor (MCU)

The MCU of almost all modern GSM DBBs is a System-on-a-Chip (SoC) utilizing a 32bit ARM7TDMI core. The only notable exception are low-cost Infineon chips like PM7870, who still use a version of their 16bit C166 core.

Baseband chips that support 3G cellular networks often use a more powerful ARM926 or ARM975 core as MCU.

3.5 MCU peripherals

The MCU cores have the typical set of peripherals of any ARM7 based microcontroller, such as RTC, UARTs for RS232 and IRDA, SPI, I2C, SD/MMC card controller, keypad scan controller, USB device, ...

However, there are some additional peripherals that are very GSM specific:

- A GPRS crypto unit for the proprietary GEA family of ciphers
- Extended power management facilities, including a timer that can calibrate the RTC clock based on the synchronized VCTCXO in order to wake-up the MCU ahead of pre-programmed events in the GSM time multiplex
- GSM TDMA timers that can synchronize to the on-air time frames and generate interrupts to MCU and DSP
- Software-programmable hardware state machines for sequencing GSM burst Rx or Tx in ABB and RF Frontend
- An ISO7816 compatible smart card reader interface for the SIM card

The programming of those peripherals is highly device-specific and there are no industry standards. Every DBB architecture of every supplier has its own custom register set and programming interface.

The register-level documentation for those proprietary peripherals is (like all documentation on DBB chipsets) closely guarded by NDAs, effectively preventing the development of Free Software / Open Source drivers for them, unless such documents are leaked by third parties.

However, as opposed to the DSP API documentation, the register-level documentation to the MCU peripherals is normally provided to the cellphone manufacturers.

4 Digital Baseband Software Architecture

This section provides an introductory reading in the typical software architecture as it is found on contemporary GSM digital baseband designs.

4.1 GSM Layer 1

The Layer1 (L1) software is highly device-specific, as it closely interacts with the DSP using the shared memory DSP API, as well as the proprietary integrated peripherals controlling the ABB and RF Frontend.

However, there are some general observations that can be made about the L1:

4.1.1 L1 Synchronous part

The synchronous part is executed synchronously to the GSM TDMA frame clock. Both CPU and DSP are interrupted by some hardware GSM timer every TDMA frame.

The L1 synchronous part typically runs inside IRQ or FIQ context of the MCU, taking care of retrieving data from and providing data to the DSP API.

4.1.2 L1 Asynchronous part

The asynchronous part is scheduled as a normal task, potentially with high or even real-time priority. It picks up the information provided by the L1 Sync and schedules its next actions.

The L1 async typically communicates via a message queue with the Layer2 above. Common primitives for L1 control are described (as non-normative parts) of the GSM specifications.

4.2 GSM Layer 2

As opposed to L1, the GSM Layer 2 (L2) is already fully hardware independent. It implements the LAPDm protocol as specified in GSM TS 04.06. LAPDm is a derivative of the ISDN Layer 2 called LAPD, which in turn is a descendent of the HDLC family of protocols.

LAPDm takes care of providing communication channels for Layer3. Those channels are protected from frame loss by the use of sequence numbers and retransmissions.

The interface to Layer3 is typically implemented by means of a message queue.

Primitives (but no detailed protocol) for use of the Layer2 / Layer3 interface are provided in the GSM specifications.

4.3 GSM Layer 3

GSM Layer 3 (L3) consist of sublayers for Radio Resource (RR), Mobility Management (MM) and Call Control (CC).

There is sufficient treatment of the GSM L3 and its sublayers in existing texts, so there is no point in making a futile attempt repeating that here.

5 Synchronization and Clocking

The author of this paper has been quoted saying *GSM is a synchronous TDMA nightmare*. This is by no means intended as an insult to the technology itself or to its inventors. It merely serves as evidence how hard it is to get into the synchronous TDMA mindset, especially for engineers who have spent most of their career in the world of packet switched networks.

GSM is synchronous in multiple ways between cell (BTS) and phone (MS):

- Synchronization of the carrier clock to tune the receiver and transmitter to the correct frequency
- Synchronization of the bit clock in order to perform sampling at the most optimal sample intervals
- Synchronization of the frame clock (and thus timeslots) to know when a TDMA frame and its 8 timeslots start

- Synchronization of the TDMA multiplex to correctly (de)multiplex the logical channels that are sent over each timeslots

As all those clocks are related to each other, they can (and should) all be derived from the same master clock: The VCTCXO in our phone.

5.1 How to synchronize the VCTCXO

Every cell sends frequency correction bursts as part of the Frequency Correction CHannel (FCCH), which is itself part of the BCCH, which in turn is constantly transmitted by the BTS.

To acquire initial synchronization ot the GSM network, the LO is tuned to the desired GSM RF channel (ARFCN) frequency. However, at this point, the LO frequency is a multiple of the VCTCXO frequency which in turn still has an undetermined error. This initial frequency error is as large as that of a regular crystal oscillator, potentially already with temperature compensation.

The resulting baseband signal thus can be shifted by a fairly large amount in our baseband spectrum. A specific DSP code is now using correlation and other techniques to identify the frequency correction burst. The DSP can then further calculate the actual frequency error of the LO by comparing the received FCCH burst with the FCCH burst as specified.

This computed frequency error can be fed into a (software) frequency control loop filter. The loop filter output is applied to an auxiliary DAC, which generates the control voltage for the VCTCXO.

After a number of FCCH bursts and corresponding frequency control loop iterations, the VCTCXO generated clock has only a residual error. Whenever the phone is receiving, the frequency control loop is continuously exercised in order to maintain synchronization.

5.2 How to synchronize the frame clock

When the DSP performs FCCH burst detection as described above, it identifies the exact position in the incoming sample stream when the FCCH burst was happening. By knowing from the specification that the FCCH burst is part of the BCCH, and that the BCCH is sent on timeslot 0, the Layer1 software can then synchronize the phone to the TDMA frame start.

Commonly, a hardware timer unit is clocked by a (divided) VCTCXO clock and thus counts in multiples of the GSM bit clock, wrapping/resetting at the TDMA duration of 1250 bits.

By scheduling events synchronously to this GSM bit-clock timer, the L1 can now trigger events (such as asking the DSP to demodulate incoming data) or instructing the LO to retune synchronously to every TDMA frame. From this timer the DBB typically also generates interrupts to the DSP and MCU.

5.3 How to synchronize the GSM TDMA multiplex

As part of the BCCH, the BTS not only sends the FCCH but also the Synchronization CHannel (SCH). The Synchronization channel indicates the current GSM time / frame number (skipping the 3 least significant bits). By using this received GSM time and incrementing it every time the GSM bit-clock timer wraps at the beginning of a new TDMA frame, the GSM time is synchronized.

Understanding the multiple layers of time multiplex such as the 26/51 multiframe, superframe and hyperframe, the L1 can multiplex and demultiplex all the logical channels of GSM.

6 Miscellaneous Topics

6.1 GPRS

GPRS was the first packet switched extension to GSM. In fact, it is much more its entirely own mobile network, independent of GSM. The only parts shared are the GSM modulation scheme (GMSK) and time multiplex, in order to ensure peaceful coexistence between them.

The L1 and L2 protocols are very different (and much more complex) than GSM.

So while the phone baseband hardware did not need any modifications for a basic GPRS enabled phone, the software needed to be extended quite a lot.

6.2 EDGE

EDGE is a very small incremental step to GPRS. It reuses all of the time multiplex and protocol stack, but introduces a new modulation: Offset 8-PSK instead of GMSK to increase the bandwidth that can be transmitted. Offset 8-PSK is used (as opposed to simple 8-PSK) to avoid zero-crossings in the modulator output.

So while the software modifications from GPRS to EDGE are minimal, the 8PSK modulation scheme has a significant impact on the DSP, ABB and even RF PA design.

6.3 UMTS

UMTS (sometimes called WCDMA) is an entirely separate cellular network technology. Its physical layer, modulation schemes, encoding, frequency bands, channel spacing are entirely different, as is the Layer1.

UMTS Layer2 has some resemblance to the GPRS Layer2.

UMTS Layer3 for Mobility Management and Call Control are very similar to GSM.

Given the vast physical layer and L1 differences, a UMTS phone hardware design significantly differs from what has been described in this document.

Notwithstanding, all known commercial UMTS phone chipsets as of today still include a full GSM modem in hardware and software to remain backwards-compatible.

6.4 Dual-SIM and Triple-SIM phones

In recent years, a large number of so-called *Dual-SIM* or even *Triple-SIM* phones have entered the market, particularly in China and other parts of East Asia.

Those phones come in various flavours. Some of them simply have a multiplexer that allows electrical switching between multiple SIM card slots. This is similar to replacing the SIM card in a phone, just without the manual process of mechanically removing/inserting the card. As a result, you can only use one of the two SIMs at any time.

The more sophisticated Dual-SIM phones have two complete phones in one case. Yes, that's right! They contain two full GSM phone chipsets, i.e. 2 antennas, 2 rf frontends, 2 analog basebands, 2 digital basebands, ...

However, they use the same trick as smartphones: One of the two basebands does not have keypad or display and is simply a GSM modem connected via serial line to the other baseband processor.

So if a smartphone (as defined in this document) is a GSM modem connected to a PDA in one case, a Dual-SIM phone is a GSM modem connected to a feature phone in one case.

Triple-SIM phones often combine the two approaches, i.e. they contain two complete GSM baseband chips, but three SIM slots that can be switched among the base bands. Only two SIMs can be active at the same time.

6.5 Powerful feature phones

Feature phones are becoming more and more powerful. However, their comparatively lower market price cannot afford a full-blown smartphone design with its two independent processors and the associated design complexity.

Thus, more and more hardware peripherals are added to the only processor left in the phone: The baseband processor. Such peripherals include sophisticated camera interfaces, high-resolution color display controllers, TV output, touchscreen controllers, audio and video codecs and even interfaces for mobile TV reception.

However, all of those features are still implemented on a fairly weak ARM7 or ARM9 CPU core (compared to ARM11 and Cortex-A8 in the smartphone market). They also lack a real operating system and still run on top of a real-time microkernel intended for much less complex systems. They almost always lack any form of memory protection or multiple address spaces. This makes them more prone to security issues as there is no privilege separation between the GSM protocol stack and the applications, or between the applications themselves.

7 Personal rant on the closedness of the GSM industry

The GSM industry is one of the most closed areas of computing that I've encountered so far. It is very hard to get any hard technical information out of them. All they like to spread is high-level marketing information, but they're very reluctant when it comes down to hard technical facts on their products.

If you want to build a phone, you need to buy a GSM chipset for your product. There are only very few companies that offer such chipsets. The classic suppliers are Infineon, Texas Instruments, ST/Ericsson, ADI (now MediaTek) and Freescale.

The GSM handset products they sell are not generally available and distributed like other electronic component they manufacture. If you need a Microcontroller/SoC, a power management IC, a Wifi or Bluetooth chip, RFID reader ASIC, you simply approach the respective distributors and order them. You get your samples directly from Digikey.

This is impossible for GSM (or other cellphone) chipsets. For some reason those chips are sold only to hand-picked manufacturers. If you want to qualify, you have to subscribe to at least six-digit annual purchasing quantities. And in order for them to believe you, you have to cough up a significant NRE (non-refundable engineering fee). This has been reported as high as a seven-digit US$ amount and is to make sure that even if you end up buying less chips than you indicate, the chipset maker will still have your upfront NRE money and keep it.

And if you buy your way into that select club of cellphone makers, what you get from the chipset maker is typically not all too impressive either. The documentation you get is incomplete, i.e. it alone would not enable you as a cellphone maker to make any use of the hardware, unless you license the software (drivers, GSM protocol stack, ...) from the chipset maker, too.

On the software side, most of the technologically interesting bits (like the protocol stack) are provided as binary-only libraries, you only get source code to some parts of the systems, i.e. some hardware drivers that might need modification for your particular phone electrical design.

That GSM protocol stack was not written by the chipset maker either. They simply license a stack from one of the estimated 4 or 5 organizations who have ever implemented a commercial GSM protocol stack.

It is not like the GSM protocols were some kind of military secret. They are a published international standard, freely accessible for anyone. So why does everybody in that industry think that there is a need to be so secretive?

Having spent a significant part of the last 6 years with reverse engineering of various aspects of mobile phones in order to understand them better and do write software tools for security analysis, I still don't understand this secrecy.

All the various vendors do more or less the same. The fundamental architecture of a GSM baseband chip is the same, whether you buy it from TI, Infineon or from MediaTek. *They all cook with water*, like we Germans tend to say. The details like the particular DSP vendor or whether you use a traditional IF, zero-IF or low-IF analog baseband differ. But from whom do they want to hide it? If people like myself with a personal interest in the technical aspects of mobile phones can figure it out in a relatively short time, then I'm sure the competiton of those chipset makers can, too. In much less time, if they actually care.

This closedness of the cellular industry is one of the reasons why there has been very little innovation in the baseband firmware throughout the last decades. Innovation can only happen by very few players. Source code bugs can only be found and fixed by very few developers at even fewer large corporations. No chance for a small start-up to innovate, like they can in the sphere of the internet.

It is fundamentally also the reason why the traditional phone makers have been losing market share to newcomers to the mobile sphere like Apple with its iPhone or Google with its Android platform.

Those innovations really only happened on the application processor on high-end smartphones. The closed GSM baseband processor had to be accompanied by an independent application processor running a real operating system, with real processes, memory management, shared libraries, memory protection, virtual memory spaces, user-installable applications, etc.

They still don't happen on the baseband processor, which is as closed as it was 15 years ago.