UNIVERSITÀ DI PISA
FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

TESI DOTTORATO

DESIGN OF ELECTRONIC SYSTEMS FOR AUTOMOTIVE SENSOR CONDITIONING

IX CICLO

FRANCESCO IOZZI

CURRICULUM

TECNOLOGIE, DISPOSITIVI E SISTEMI MICRO E NANOELETTRONICI

UNIVERSITÀ DI PISA
FACOLTÀ DI INGEGNERIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

TESI DOTTORATO

DESIGN OF ELECTRONIC SYSTEMS FOR AUTOMOTIVE SENSOR CONDITIONING

IX CICLO

CURRICULUM

TECNOLOGIE, DISPOSITIVI E SISTEMI MICRO E NANOELETTRONICI

Tutori:

Prof. L. Fanucci _____

Prof. R. Saletti _____

Allievo:

Francesco Iozzi _____

TABLE OF CONTENTS

INDEX OF FIGURES

INDEX OF TABLES

INTRODUCTION

Electronic systems for vehicles are constantly gathering more commercial importance in the extremely competitive automotive market, as they represent a factor of differentiation for car manufacturers and have proved to be very attractive for customers. Nonetheless, the integration of electronics into most mechanical components inside modern cars has considerably contributed to the increase of vehicles performances, comfort and, above all, travel safety.

Such systems have reached a noteworthy complexity and one or more functionalities are handled by each Electronic Control Unit, which generally acquire the needed information through a set of sensors and process the data in order to properly command the actuators. Dozens of these ECUs are commonly present inside a medium-segment car: this implies that hundreds of sensors are working at the same time under the hood or hidden anywhere in the body or chassis.

This thesis deals with the development of sensor systems for automotive, mainly targeting the exploitation of the new generation of Micro Electro-Mechanical Sensors (MEMS), which achieve a dramatic reduction of area and power consumption (thus making possible the coexistence of so many sensing elements) but at the same time require more complexity in the sensor conditioning interface.

The design of sensor systems for automotive applications is subject to tighter constraints with respect to, for example, consumer electronics, as superior performances as required (especially for applications related to safety systems) and they must be guaranteed over the [-40°C,+125°C] temperature range and in harsh operating environment (high vibrations, liquid and moist exposure). Also self detection of failures is a common automotive demand, together with 'classic' requirements such as low power, low area and short time-to-market.

Several issues concerning the development of automotive ASICs are presented in Chapter 1, together with an overview of automotive electronics market and its main sensor applications. Chapter 2 introduces the state of the art for sensor interfaces design (the generic sensor interface concept), which consists in sharing the same electronics among similar sensor applications, thus saving cost and time-to-market but also implementing a sub-optimal system with area and power overheads. A Platform Based Design methodology is proposed to overcome the limitations of generic sensor interfaces, by keeping the platform generality at the highest design layers and pursuing the maximum optimization and performances in the platform customization for a specific sensor. A complete design flow is presented (up to the ASIC implementation for gyro sensor conditioning), together with examples regarding IP development for reuse and low power optimization of third party designs.
Chapter 3 describes a further evolution of Platform Based Design achieved by means of implementation into silicon of the ISIF (Intelligent Sensor InterFace) platform. ISIF is a highly programmable mixed-signal chip which allows a substantial reduction of design space exploration time, as it can implement in a short time a wide class of sensor conditioning architectures. Thus it lets the designers evaluate directly on silicon the impact of different architectural choices, as well as perform feasibility studies, sensor evaluations and accurate estimation of the resulting dedicated ASIC performances.
Several case studies regarding fast prototyping possibilities with ISIF are presented: a magneto-resistive position sensor, a biosensor (which produces pA currents in presence of surface chemical reactions) and two capacitive inertial sensors, a gyro and a low-g YZ accelerometer. The accelerometer interface has also been implemented in miniboards of about 3 cm^2 (with ISIF and sensor dies bonded together) and a series of automatic trimming and characterization procedures have been developed in order to evaluate sensor and interface behaviour over the automotive temperature range, providing a valuable feedback for the implementation of a dedicated accelerometer interface.

Chapter 1
SENSOR SYSTEMS IN AUTOMOTIVE APPLICATIONS

1.1 Automotive electronics market overview

Automotive electronics are major criteria of differentiation in the automotive market. Car manufacturers use chips in increasing numbers to develop powerful electronic systems for driver information and communication, in-car entertainment electronics, power train and body control electronics as well as automotive safety and convenience electronics. In 2005, about one quarter of the value of the average car is comprised of electrical and electronics components [1].

- The $16-plus billion worldwide automotive semiconductor business in 2004 will rise at an average annual growth rate (AAGR) of 9% through 2009 to nearly $25 billion.
- Integrated circuits will show the highest use growth-trend in safety (10.4%), followed by body and chassis, each with AAGRs of 9.8%. In these areas, there is need for accurate, closed-loop, real-time control, necessitating the processing of large volumes of data from multiple sensors.
- Automotive safety represents a growing, and the most stable market for semiconductors due to influence from governments, consumers and the automakers themselves.

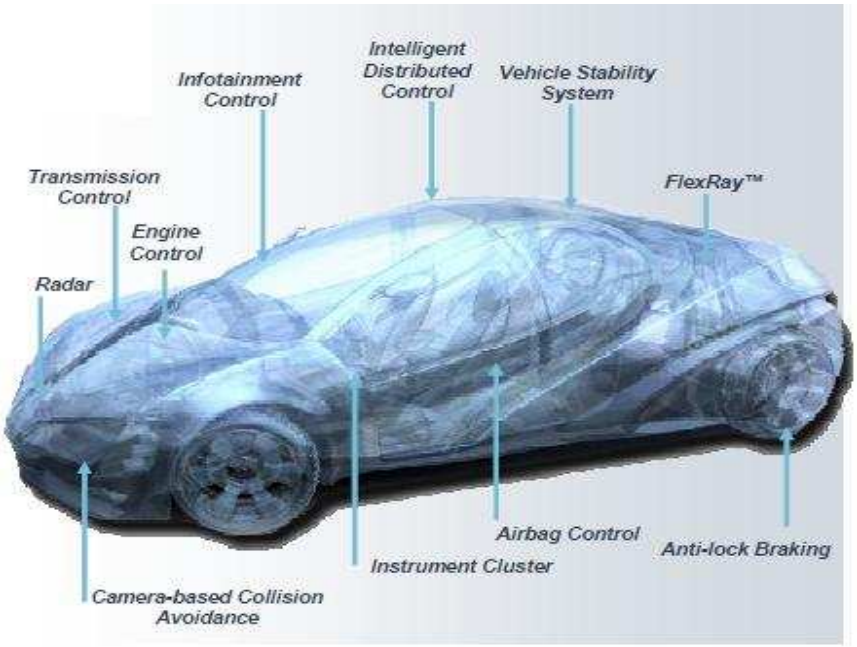

Figure 1: Electronic Units inside a car [source: Freescale Corp.]

With innovative automobile electronics serving as considerable value additions, customers are increasingly demanding features that enhance overall travelling pressure. Such features range from the basic control of climate, lighting, and humidity to the automatic setting of seat, rear view mirrors and steering wheel position according to the driver's physique. The mixed signal system on a chip (SoC) in the very last years has appeared to be the single-source solution that can manage the entire spectrum of such features.

System on a chip applications are expected to further extend to a range of new intelligent and compact systems, prompting semiconductor companies to heighten their focus on the automobile industry by developing a wider range of solutions for the industry. There is now a widely shared vision among chip designers, suppliers, as well as original equipment manufacturers (OEMs) and this promises a clear commitment to improvising the use of the technology in the automobile end-user segment. In addition to the growing popularity of novel comfort features, the increasing demand for advanced safety systems as well as driver infotainment systems is likely to spur the use of system on chips in automobiles.

For what concerns safety, one of the main purposes of electronic systems is to assist the driver to control the vehicle through functions related to the steering, traction (i.e., control of the driving torque) or braking such as the antilock braking system (ABS), electronic stability program (ESP), electric power steering

(EPS), active suspensions, or engine control. Such functions are enjoying greater demand and employ mixed signal chipsets to alert drivers ahead of any possible mishaps.
With respect to infotainment features (e.g. entertainment and communication equipment such as radio, DVD, hands-free phones, navigation systems), more and more consumers are opting for wireless systems, onboard computers, information display, as well as voice processing capabilities in their vehicles and this has lead to tremendous technology improvements from companies developing this class of mixed signal system on chips.
In today's luxury cars, up to 2500 signals (i.e., elementary information such as the speed of the vehicle) are exchanged by up to 70 Electronic Control Units [2].

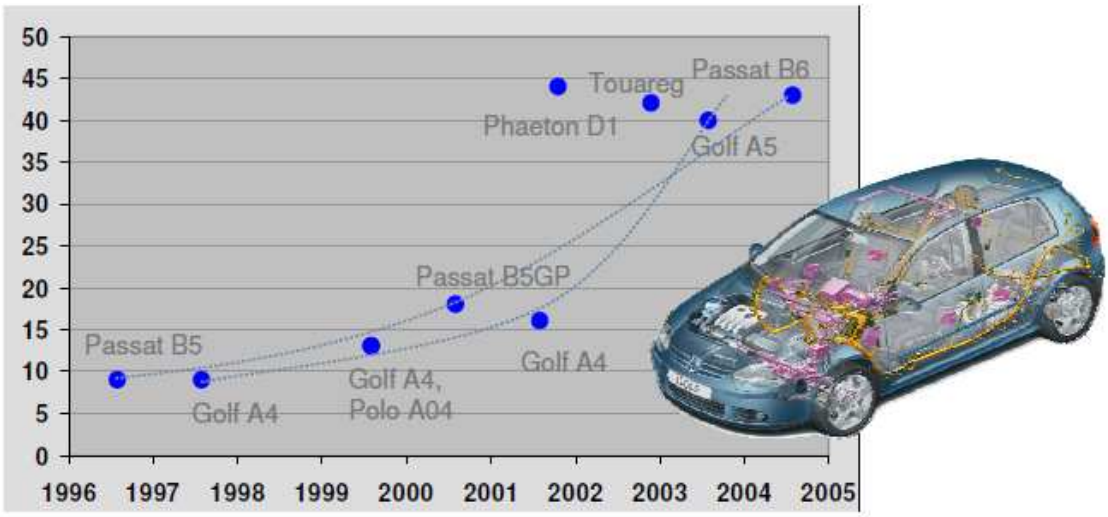

Figure 2: Number of ECUs inside Volkswagen cars in the last 10 years [source: Volkswagen AG]

In terms of geographic regions, Europe is the predominant revenue generator for SoCs in the automobiles market and is expected to garner a market share of around 58.0 percent in 2006. Among the others, the Americas (including the United States, Canada, Mexico, and Latin America) as well as Asia Pacific are likely to account for an almost equal share of the remaining market revenues between 2006 and 2009. Overall, in the next ten years, mixed signal system on chips in automobiles are likely to offer tremendous market opportunities for vendors in the segment. SoCs are expected to have an incredible impact on the market and are posed to be among the best emerging opportunities in the automotive segment [3].

1.2 MEMS sensors markets

The market for MEMS inertial sensors (accelerometers and gyroscopes) is set to grow from $835 million in 2004 to over $1360 million in 2009 — a CAGR of 10%. Currently, the main applications are in the automotive industry. These markets are well established and growth rates range from a stagnant 1% for airbag acceleration sensors up to 8% for gyroscopes used in ESP units and GPS navigation assistance.
A new exciting opportunity for MEMS inertial sensors is the market of mobile applications and consumer electronics (Figure 3). Over the next few years, market reviews predict annual growth rates exceeding 30% for accelerometers. Mobile phones in particular will provide multi-axis accelerometers with interesting opportunities in menu navigation, gaming, image rotation, pedometers, GPS navigation and the like. Gyroscopes are largely servicing markets for image stabilization and HDD protection in camcorders.

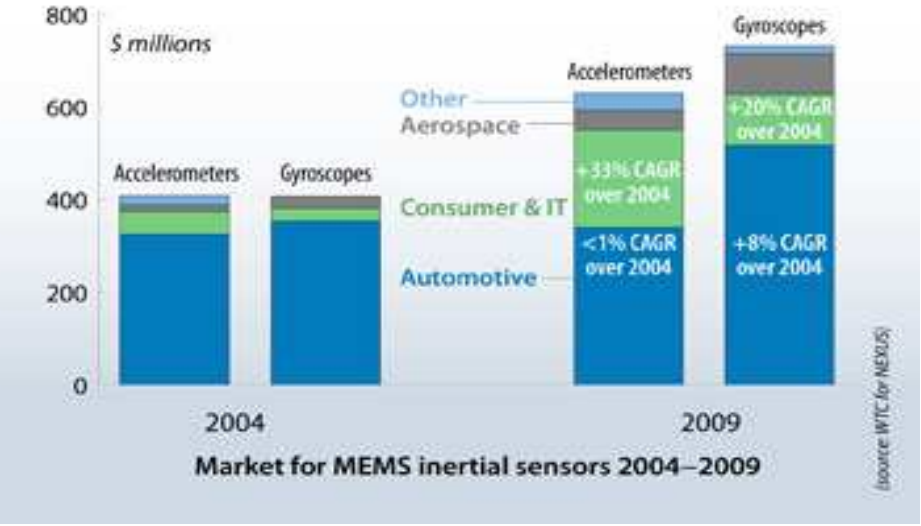

Figure 3: Growth forecast for MEMS gyros and accelerometers up to year 2009 [4].

In contrast to the automotive sector, consumer applications feature relaxed specifications. Failure rates for automotive electronic control units (ECU) that house airbag accelerometers must be less than 50 ppm, and down to a few ppm for ASICs. Car manufacturers deploy reliable, high performance accelerometers that are relatively expensive (up to $5 to measure lateral acceleration in ESP units, for example). Mobile phones manufacturers on the other hand tolerate failure rates of 5000

ppm. Active control of the car motion is critical to safety whereas failure in consumer applications result more in an inconvenience. Sensor manufacturers can therefore sacrifice some accuracy and reliability to lower the price [4].
The auto industry was among the first to embrace micro-electro-mechanical sensor (MEMS) technologies on a wide scale, as it substituted MEMS technology for more expensive conventional manifold absolute pressure sensors in engine management systems and electro-mechanical accelerometers in airbag deployment applications. The industry continues to be a leader in finding ever more MEMS applications in the automobile platform, including new inertia, pressure and position sensing roles. Other sensor technologies of interest include object and headway detection systems, which can utilize a mix of sonic, laser, radar and vision technologies to avoid accidents and increase vehicle safety. Engine and drivetrain applications represent the largest and most well established category in sensor use, with much of the value in this category residing in oxygen sensors, which continue to both multiply and become more sophisticated. However, safety and security applications promise the greatest growth potential for OEM automotive sensors, propelled by new mandated and market-driven products such as tire pressure sensors in the US, pedestrian warning systems in Europe, and airbags and automatic seatbelts in many emerging markets [5]. A high-end car may contain between 50 and 100 sensors, most of which are MEMS-based and they are distributed all over the car as shown in Figure 4.

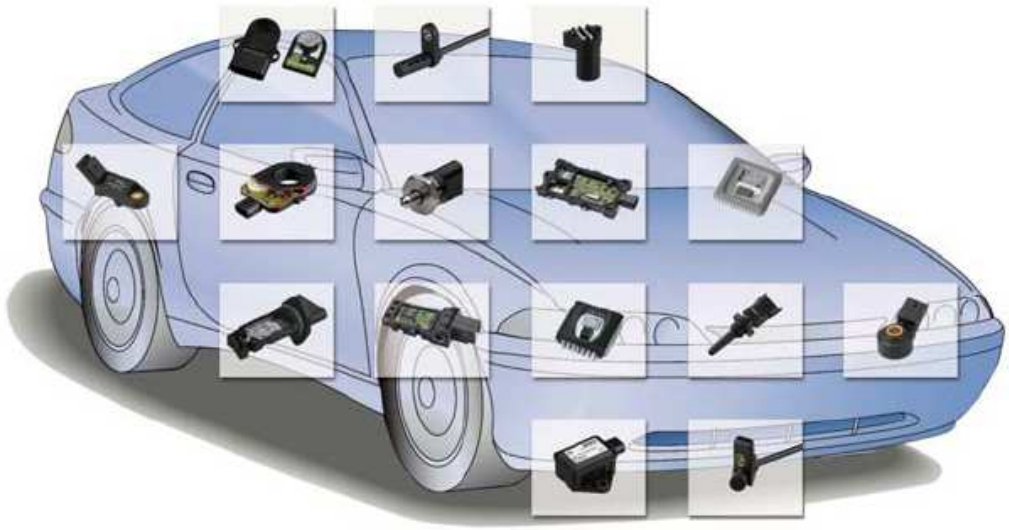

Figure 4: Sensors are the main application field for microsystems in the car. [Source: Robert Bosch Corp.]

1.3 MEMS technology

Micro-Electro-Mechanical-Systems (MEMS) are three-dimensional structures manufactured through silicon micromachining technologies. They made their first appearance in semiconductor foundries in the sixties. Our daily life is full of micromachined products.
MEMS compete with non-semiconductor based solutions in price and performances. But miniaturization is definitely another big advantage they bring to the consumer market.
MEMS are micron-sized devices that interact with the physical world. They are manufactured via a process called micromachining, which shares the same processing steps derived from basic integrated circuit techniques. The end result, however, is typically a 3-Dimensional mechanical structure, most often on a silicon substrate. Nonetheless, other materials can be micro-machined or micro-formed. Among these materials are quartz, glass, plastic, and ceramic [6]. Quartz is used for crystal resonators and for Coriolis-based gyroscopes. Still, silicon is becoming increasingly popular thanks to its excellent electrical, mechanical and thermal properties. In addition to its excellent physical properties, silicon is extremely attractive because manufacturers can realize thousands of micro-machined components at a time on silicon wafers using proven manufacturing techniques developed for silicon chip production.

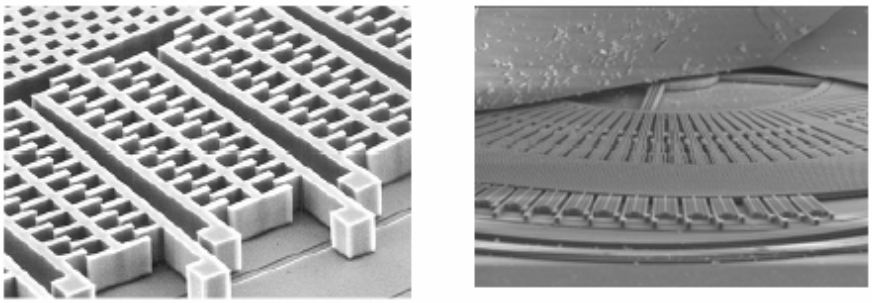

Figure 5: microscope views of MEMS structures on silicon [6].

The scope and use of MEMS is primarily due to extremely small size, terrific reliability, and low power consumption, which, in many instances, allows MEMS to be capable of faster and more precise operations than their macroscopic equivalents. But the cost advantage for the customer cannot be ignored.
Although each company uses a specific micro –machining process, all of the processes can be classified into two broad classes:
A. *Bulk Micromachining*: it is a subtractive process because a large portion of the substrate is removed to form whatever structure is

desired. This technique requires less precision than surface micromachining. Thicker structures are easier to fabricate because the substrate thickness can be chosen quite freely, but the shape of the micro-machined structure is quite limited by the crystal planes of the silicon substrate.
B. *Surface Micromachining*: it is an additive process requiring the building up of various layers of materials that are selectively left behind or removed by subsequent processing. The bulk of the substrate remains essentially untouched. This technique was initially limited to thin devices (~2 micron), since only thin films could be deposited or grown on the substrate.

1.4 MEMS automotive applications

In this paragraph an overview of the possible automotive applications for MEMS sensors is presented [7].

i. Compensated Compass

In addition to user interfaces the use of MEMS sensors are allowing more precise and user-friendly compasses on handhelds. With an inertial module, containing magnetometers and accelerometers, a user is able to read a compass while handling the device in every position and inclination. In fact, the accelerometer compensates the inclination of the device enabling a correct and precise reading of the compass.

ii. Intelligent Airbags

During car crashes the safety of the occupants are also devoted to airbags deployment, and it must occur at the perfect time and also with the correct power toward the passengers.
While the detection of the passengers and their motion can be detected by a seats-based MEMS accelerometer system, the airbag system can be positioned differently in a car allowing multiple information determination such as identification of type of collision, its direction, g-force impact; in this way a specific and dedicated airbag system reaction can be guaranteed.
Using the MEMS accelerometer, exploiting their high integration capability and accuracy, can be defined more complex system architecture will allow replacing multiple electromechanical crash sensor system guaranteeing an advanced passenger safety condition.

iii. Vibration Monitoring

An accelerometer may be used to measure the frequency, amplitude (strength) and spectrum (signature) of vibrations, enabling the ability to perform active monitoring of home appliances and industrial applications.
It is well known that machine vibrations lead to excessive power consumption and impose additional wear on bearings, seals and foundations. Vibrations, which are typically caused by machine misalignment and unbalance, are detectable through the spectral analysis (FFT) of the measured acceleration signal.
If left uncorrected, machine vibration results in degraded performances, shorter tool life, unpleasant noise and increased maintenance costs. In addition, vibration monitoring permits the identification of potential machine problems prior to equipment failure and allows predictive maintenance to be implemented.

iv. Antitheft

The accelerometer also finds application in another important area especially in the automotive field: antitheft. In this case the accelerometer is used as an inclinometer which senses the inclination of the car or motorbike versus the ground. When a tow truck is used to steal a vehicle, the accelerometer will detect the change in inclination and sound the security system.
A 3-axis accelerometer also enables easy installation: the security system can be installed in any position inside the vehicle.

v. Head Light Positioning

Driving cars safely in a darkness condition it depends also on accurate and precise head light positioning of our cars.
A lot of different conditions may vary the alignment of head light respect to road we are driving; some of them depend by road track such as asphalt, curves, uphill, downhill, driver skill, experiences, speed and so on, other ones depend on car-system condition like type pressure, suspensions, number of occupants, weights and balancing.
Driver must not take care to continuously align head light to those conditions but the head lights themselves should take care of it.
With the MEMS sensors like accelerometer and gyroscope a new era of head lights will appear enhancing road and obstacles illumination guaranteeing safer driving conditions.

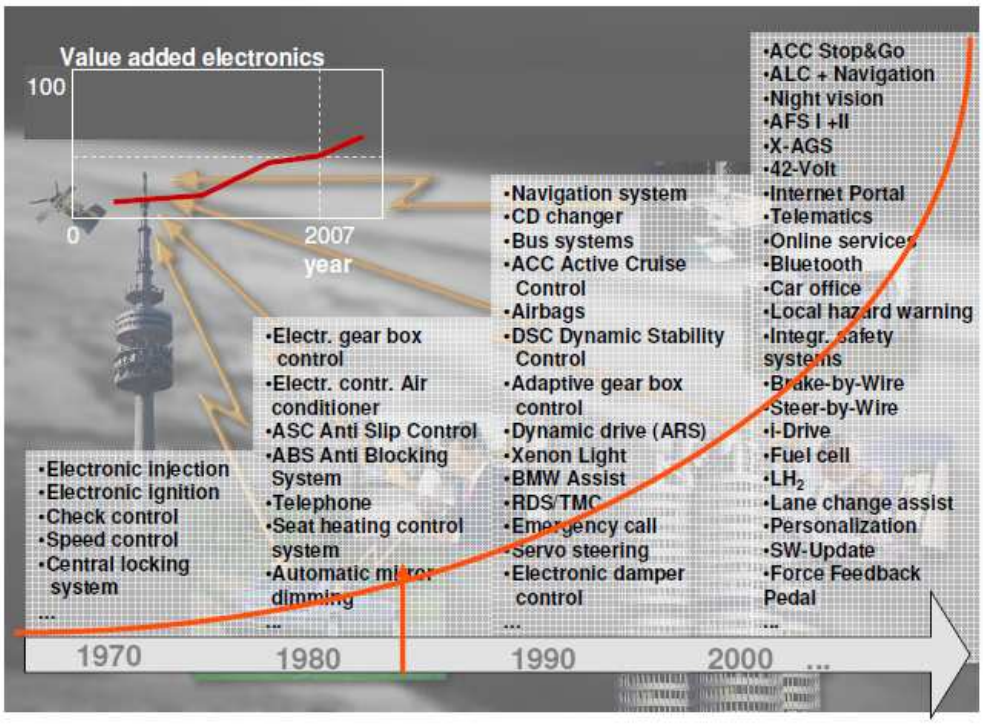

Figure 6: Value added electronics in cars. [source: BMW Group]

vi. Occupant Detection

In case of impact between cars and obstacles to safely allow life of passengers a complete system of airbags are distributed along the seats.

Because the instant of the impact cannot be foreseen in advance, the car-system shall determine who and/or what is sitting in passenger seats. This will allow not only which airbags should be deployed but even more important the force it should be deployed accordingly some factors depend from each passenger and their position inside the car.

The MEMS accelerometer enhances this function – detects when a person is lifted from the seat because of the force of impact.

vii. Navigation Systems

Satellite navigation systems used in vehicles use radio signals from Global Positioning System (GPS) satellites to determine the position of the receiver anywhere on the globe.

In practice, however, satellite position data is not sufficient because the satellite signals are shadowed by buildings and blocked by bridges and overpasses especially in dense urban areas.

In this context, the MEMS-based inertial sensors can assist and substitute for the GPS in case there is signal loss: a dead reckoning system will continue tracking movements during the time when satellite signals are not visible or where they are not sufficiently accurate. To implement dead reckoning, it is necessary to know the distance and direction travelled.
To provide continuous coverage vehicle navigation equipment also includes an acceleration sensor attached to the transmission and a MEMS-based Coriolis Effect gyroscope and a magnetometer to determine the direction of motion. Together the acceleration sensor, the magnetometer and gyroscope allow "deduced reckoning" independent of the visibility of satellites. The GPS satellite position fixes eliminate the accumulation of errors over time.
It is important to note that because battery-operated GPS devices consume a lot of power, dead reckoning is a vital feature enabled by MEMS-based inertial sensors in portable devices.

viii. Active Safety

The MEMS accelerometers and gyroscopes are both sensors which can perfectly address active safety systems in the automotive domain.
Control of car roll-over, vehicle stability for skidding and antilock braking, parking brake energy, activation of wheel pressure monitoring, suspension adaptation to car and road condition, and other systems embedded in cars represent always more attractive features for customers and are becoming a synonym for high quality standard of the vehicle. Hereafter a brief explanation [8] of the most in vogue acronyms is reported:
ABS (Anti-lock Braking System): prevents the wheels from locking while braking. Since it came into widespread use in production cars (1978), ABS has made considerable progress. Recent versions not only handle the ABS function itself but also *Emergency Braking Assistance* (EBA), which interprets the speed and pressure at which the brake pedal is pushed to detect a critical braking situation, and is often coupled with *Electronic Brake force Distribution* (EBD), which automatically varies the amount of force applied to each of a vehicle's brakes, based on road conditions, speed, loading, etc.
Cornering Brake Control (CBC): stabilisation during partial braking whilst cornering. With sensitive regulation of the braking force on individual wheels, this ABS function compensates for any destabilizing yawing moments around the vertical axis when braking on a curve.

Traction control system (TCS): an electro-hydraulic system designed to prevent loss of traction (and therefore the control of the vehicle) when excessive throttle or steering is applied by the driver.
Electronic (Dynamic) Stability Control (ESC, ESP, DSC): such systems compare the driver's intended direction in steering and braking inputs, to the vehicle's response, via lateral acceleration, rotation (yaw) and individual wheel speeds. ESC then brakes individual front or rear wheels and/or reduces excess engine power as needed to help correct understeer (plowing) or oversteer (fishtailing). ESC also integrates all-speed traction control, which senses drive-wheel slip under acceleration and individually brakes the slipping wheel or wheels, and/or reduces excess engine power, until control is regained.
Adaptive (Active, Intelligent) Cruise Control (ACC, ICC): use either a radar or laser setup to allow the vehicle to slow when approaching another vehicle and accelerate again to the preset speed when traffic allows. Some systems also feature forward collision warning systems, which warns the driver if a vehicle in front - given the speed of both vehicles - gets too close (within the preset headway or breaking distance).
Dynamic Steering Response (DSR): corrects the rate of hydraulic or electric power steering system to adapt it to vehicle's speed and road conditions.
Lane Departure Warning system (LDW): warns a car driver when the vehicle begins to move out of its lane (unless a turn signal is on in that direction) on freeways and arterial roads.
AWAKE: monitors the driver and detects hypovigilance in real time, based on multiple measuring parameters. It is composed of HDM (Hypovigilance Diagnosis Module), TRE (Traffic Risk Estimation Module), DWS (Driver Warning System, using acoustic, visual and haptic means in various levels of warnings) and HM (Hierarchical Manager, to perform self-diagnosis and co-ordinate the other system components) [9].

Microsystem	Application	Technologies
Acceleration Sensor	Frontairbag, Sideairbag, Upfontsensors, Vehicle Dynamic Control, Active Suspension	Si Surface Micromachining, Si-Bulk Micromachining, ceramic piezoelectric
Angular Rate Sensors (Gyro, Yaw Rate)	Vehicle Dynamics Control, Rollover Sensing, GPS Navigation Systems	Si Surface Micromachining, Si-Bulk Micromachining, Quartz Micromachining
Pressure Sensors	Air intake, Atmospheric pressure, Fuel Vapour pressure, Turbocharger pressure, tire pressure, HVAC, Fuel injection	Si Bulk micromachining Si Surface micromachining
Flow sensors	Air intake	Si Bulk micromachining Hot wire
Chemical Sensors (electronic nose only)	HVAC	Si Bulk micromachining + Metaloxide-Thickfilm, Thinfilm
Nozzles	Fuel injection	Stamping, Laser, Micro-erosion in metal substrates
Fluid condition sensors*	Oil condition	Mostly conventional
Magnetic Sensors* for angle, position and speed.	Wheelspeed, Camshaft position and speed, pedal position, steering wheel angle	MR, Hall, inductive
Radar Sensors*	Parking aid, blind spot surveying, Cruise Control	Microwave sensor

Table 1 : resume of MEMS sensors, applications and technologies in automotive field.

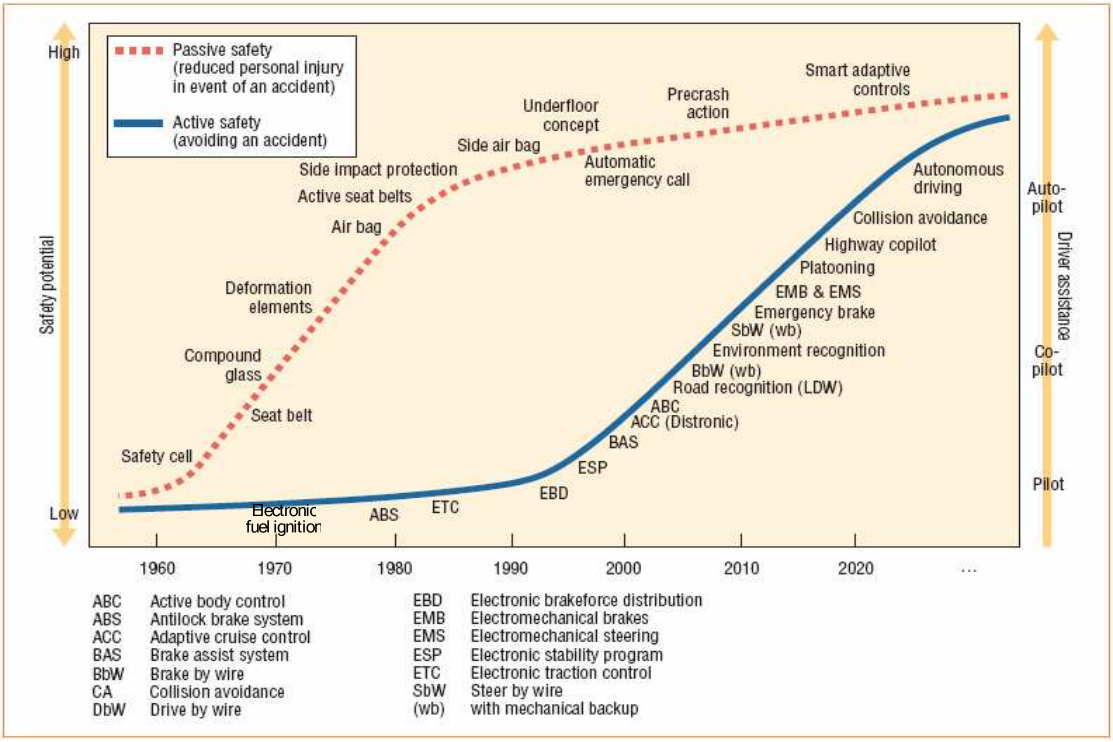

Figure 7: Evolution of passive and active safety features in cars [10].

1.5 Specifications for automotive electronics

The design of electronics for automotive applications must necessarily comply with the severe specifications of this field. A strict requirement of such devices concerns performances, first of all in terms of power consumption, area and cost. The high numbers of ECUs simultaneously working inside a car and the growing impact of electronics on the vehicle cost are pushing towards a reduction of available power for each electronic system, together with the need to combine multiple functionalities inside a smaller number of modules in order to optimize power consumption and decrease the expensive demands of interconnection resources. Regarding the edges of such complex units, i.e. the sensing elements, the related specifications may add further requirements with respect to the ECU functionality they belong to. While sensors working in the infotainment sphere share specifications with most of consumer electronics, those being part of active safety systems or body electronics have often stricter requirements in terms of noise, sensitivity and stability over temperature.

A maybe even bigger challenge for automotive electronics designers concerns the reliability issues. Car manufacturers are competing in a difficult and almost zero-growth market. Robustness is a powerful mean to attract customers and claim superior quality with respect to competitors. For this reason they tend to raise as much as possible the expectations on the operative life of the vehicle (the so called "mission profile"), resulting in always tighter specifications for the related electronics. BMW for example, assumes a car service life of 15 years, with 300000 Km distance and 6000 hours of operation, 12000 engine starts and environment temperature between -40°C and +85°C [11]. Moreover OEM customer quality level requirements increased from defects over 100 units produced to parts per million and are expected to raise to parts per billion in the near future in the quest for absolute zero defects.

Electronics, as a part of the vehicle, has to guarantee the compliance to such standards, with the added difficulty of operating in harsh environment. The electronics products in vehicles, especially under-hood components, work in severe conditions, including petroleum vapours, vibration (up to 10g rms near the engine), moisture, various fluids, dirt, chemicals and electro-magnetic interference. Besides all these, automotive

microelectronics must sustain temperatures well above the traditional maximum operating temperatures for consumer electronics, as in Table 2.

Electronics	Operating Temperature
Consumer	0 °C to +70 °C
Industry	-40 °C to +85 °C
Automotive	-40 °C to +125 °C
Military	-55 °C to +125 °C

Table 2: Thermal operating environments [12].

Protract exposure to high temperature can affect badly the behaviour of an integrated circuit. The main consequence of high operating temperature is a faster wearing of the device: as a matter of fact, during testing the average life expectancy of a component is calculated statistically from the faults detected in a set of samples working continuously at high T or subject to a number of temperature cycles. According to experimental laws [13], the medium junction temperature shortens the device life time exponentially. This implies that when the specified operating temperature gets too high (also considering the physical limits of working T for the silicon substrate, around 125 °C) it becomes critical (or too time consuming) to evaluate correctly the life expectancy of the electronic system. Such issues obviously conflict with the long term reliability required from the car manufacturers. Table 3 resumes the worst conditions to be faced by automotive electronic devices according to their position inside the vehicle.

Location	Typical Continuous Max Temperature	Vibration Level	Fluid Exposure
On engine On transmission	140°C	Up to 10 g rms	Harsh
At the engine (intake manifold)	125°C	Up to 10 g rms	Harsh
Under hood (near engine)	120°C	3 – 5 g rms	Harsh
Undrer hood (remote location)	105°C	3 – 5 g rms	Harsh
Exterior	70°C	3 – 5 g rms	Harsh
Passenger compartment	70-80°C	3 – 5 g rms	Benign

Table 3: Automotive temperature extremes (Delphi Delco Electronic Systems) [14].

1.6 Technologies and electronics design issues

Increasing time-to-market pressures and the availability of shrinking process technologies (as the Moore's law is showing no sign of outdating) are the two fundamental forces driving designers, design methodologies, and EDA tools and flows today.

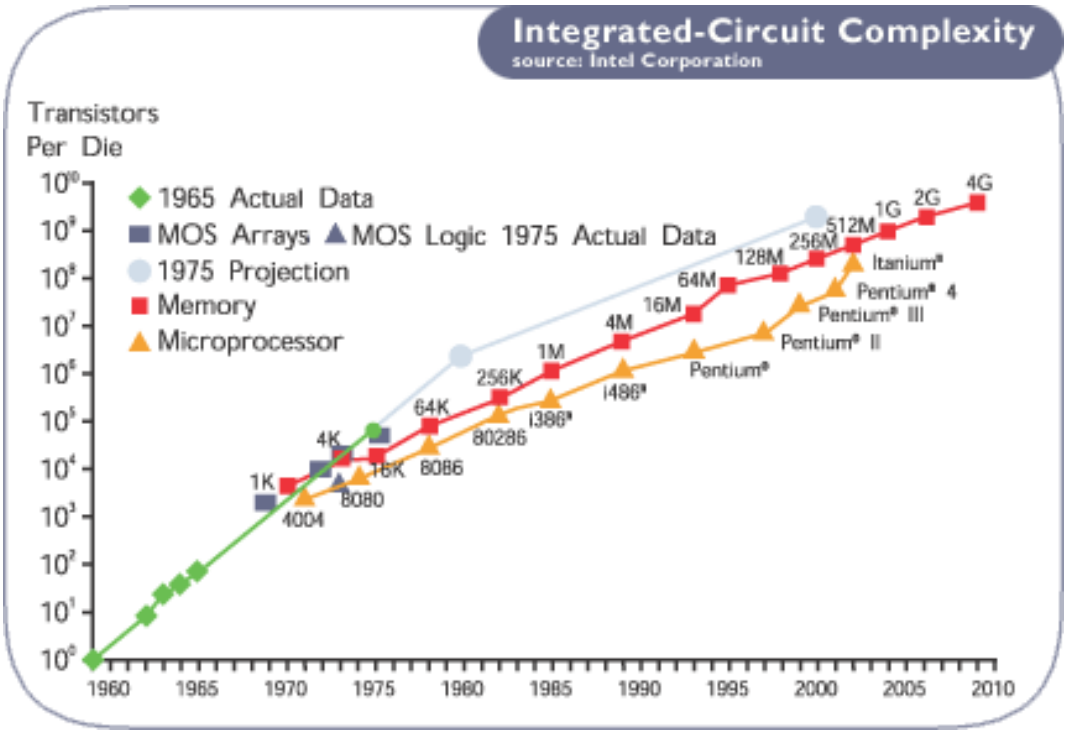

Figure 8: Moore's law: projections and integration capability over the last 30 years [source: Intel Corporation].

On one hand, market pressures together with the added integration afforded by the latest technologies, have forced a move to higher levels of abstraction to cope with the added complexity in design. On the other hand, shrinking processes have also caused a move in the opposite direction: because of the increasing significance of physical effects, there has been a need to observe lower levels of detail. Signal integrity, electro-migration, and power analysis are adding severe complications to design methodologies already stressed by the increasing device count.

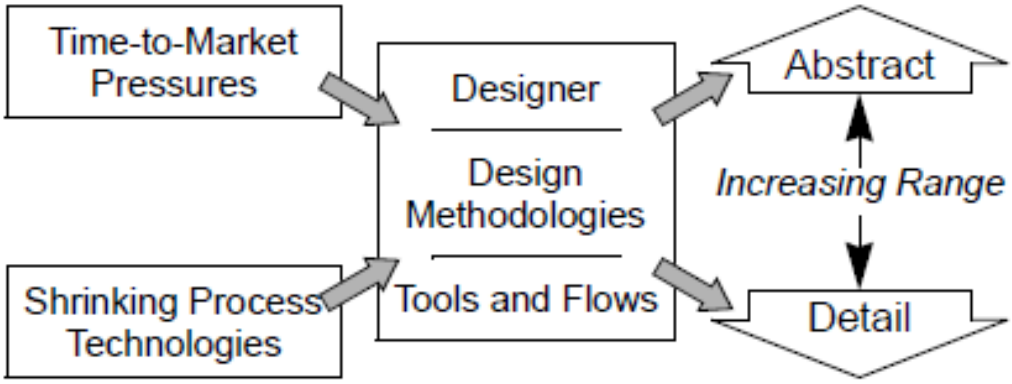

Figure 9: Design drivers and design methodology gaps [15].

These stresses uncover significant methodology gaps, which occur both between abstraction layers as well as within them. Design methodologies, tools, and flows, evolve to try to hold the design "system" together. Designers are driven to take advantage of the smaller process technologies, putting entire systems on chips: analog blocks, digital hardware, microcontrollers with memories and so on. In this way they succeed in reducing the product cost and fulfil the market demands, but also have to deal with a complexity difficult to be handled. In this scenario, top-down design methodologies have become a must [16]. In a basic top-down approach, the chip architecture is defined as a block diagram and simulated and optimized using either a Mixed Signal HDL simulator or a system simulator. From the high-level simulation, requirements for the individual circuit blocks are derived. Circuits are then designed individually to meet these specifications. Finally, the entire chip is laid out and verified against the original requirements.

Designers and EDA tools have to keep under control an increasing number (and various categories) of constraints when implementing mixed signal SoCs. Physical constraints apply to the physical entities used to implement the layout. Examples are distances, area and aspect ratio, alignment between instances. Electrical constraints apply to specific signals in the circuit. Examples are timing, parasitics, IR drop, crosstalk noise, substrate coupling noise and electro-migration. Design constraints are used to characterize the behavior of individual components in terms of their I/O signals and performance. Examples are throughput, slew rate, bandwidth, gain, phase margin, power dissipation, jitter, etc. With complex AMS chips, design constraints might include specifications on sophisticated measurements such as distortion, noise and frequency response.

Due to the high complexity of modern SoCs and the large design effort thus required, it is hoped that the mixed-signal blocks can

be designed in advance as relatively generic components and incorporated into many designs. To support this, the mixed-signal blocks must be designed for reuse: at a minimum this implies that should be available certain documentation that describes the block. Standards that specify what type of documentation is required have been set by the Virtual Socket Interface Alliance (VSIA) [17]. In addition, if the block is large it may be required to be embedded in special interface collars to make it easier to import them into an ASIC-SOC. Design for reuse is effective only in case that developing a module for future reuse and customizing it takes significantly less effort than redesigning the block for a new application. An important task when preparing a block for reuse is generating a high-level model of the block that captures its essential behavior in order to be able to evaluate the suitability of the block for use in follow-on projects.
For what concerns more specifically the technologies for automotive applications, Figure 10 explains the trend over the last ten years. While pure digital devices can follow the pace of consumer and industry electronics, affording the front-end CMOS processes, mixed signal SoCs often employ derivate technologies (such as HVCMOS or BCD), which fit better the implementation of high performance analog cells and also allow the integration of power devices.

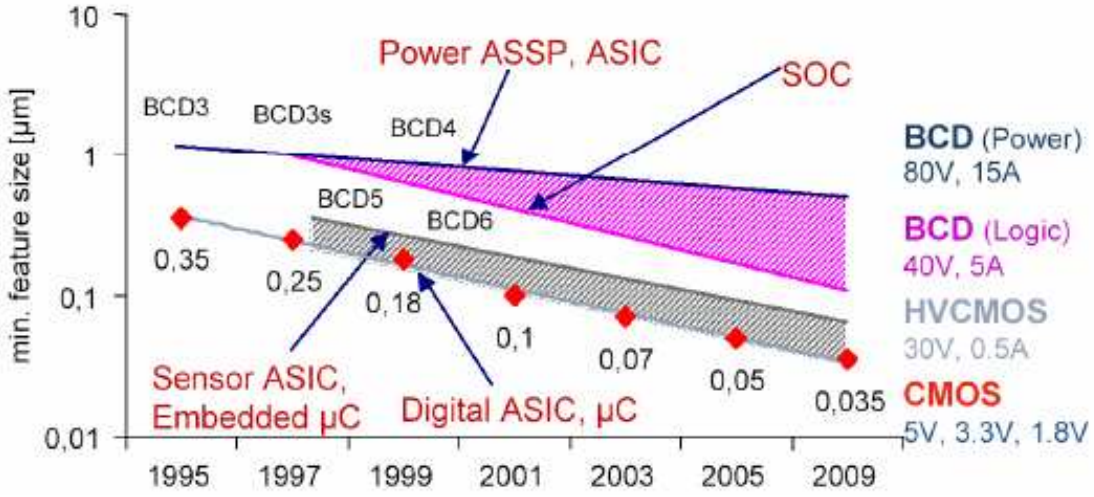

Figure 10: Technologies in automotive ICs [source: Robert Bosch GmbH]

In terms of dimensioning the automotive technologies have the same shrinking trend as front-end CMOS, simply with a few years gap that the latest BCD processes are trying to reduce (for example 0.35 µm BCD6 comes almost 10 years after the corresponding CMOS size appearance). This gap originates from the need to provide sufficient analog characterization of the

featured devices, after the CMOS process modification for the integration of bipolar transistors and DMOS. Moreover, the strict requirements of automotive standards (as mentioned in paragraph 1.5) impose accurate process verification before release.

Bibliography

[1] 'World Automotive Semiconductor Markets', http://www.electronics.ca/reports/ic/semiconductor_markets.html

[2] Navet, N.; Song, Y.; Simonot-Lion, F.; Wilwert, C.; "Trends in Automotive Communication Systems", Proceedings of the IEEE Volume 93, Issue 6, June 2005 Page(s):1204 - 1223

[3] 'World Markets for Mixed Signal SoC in Automobiles' , www.marketresearch.com

[4] 'Inertial MEMS sensors for consumer applications' by Richard Dixon and Jérémie Bouchaud, Wicht Technologie Consulting http://www.memsinvestorjournal.com/2006/05/inertial_mems_s.html

[5] 'World Automotive Sensors', http://www.electronics.ca/reports/instruments-sensors/automotive_sensors.html

[6] Vigna, B.; "Future of MEMS: An industry point of view", Thermal, Mechanical and Multiphysics Simulation and Experiments in Micro-Electronics and Micro-Systems, 2006. EuroSime 2006. 7th International Conference on, 24-26 April 2006 Page(s):1 - 8

[7] 'MEMS Applications Automotive', http://www.st.com/stonline/products/technologies/mems/applications/auto.htm

[8] 'Automobile Safety', Wikipedia

[9] AWAKE, System for Effective Assessment of Driver Vigilance and Warning According to Traffic Risk Estimation, http://www.awake-eu.org/

[10] G. Leen, D. Heffernan, "Expanding automotive electronic systems," Computer, Vol. 35, no. 1, January 2002, pp. 88-93.

[11] 'Targets and Requirements for Robust Automotive Electronics', BMW group

[12] Ohadi, M.; Jianwei Qi; "Thermal Management of Harsh-Environment Electronics", Semiconductor Thermal Measurement and Management Symposium, 2004. Twentieth Annual IEEE 9-11 Mar 2004 Page(s):231 - 240

[13] R. Amro; J. Lutz; A. Lindemann; "Power Cycling with High Temperature Swing of Discrete Components based on Different Technologies" Proc. Of 35th annual IEEE power electronics specialists conference, pp.2593-2598, Aachen, Germany,2004

[14] M. R. Fairchild, R. B. Snyder, C. W. Berlin, and D. H. R. Sarma, "Emerging substrate technologies for harsh-environment automotive electronics applications,", SAE Technical Paper Series 2002-01-1052.

[15] Ken Kundert, Henry Chang, Dan Jefferies, Gilles Lamant, Enrico Malavasi, Fred Sendig, "Design of Mixed-Signal Systems on Chip", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 19, no. 12, pp. 1561-1571, December 2000.

[16] H. Chang, E. Charbon, U. Choudhury, A. Demir, E. Felt, E. Liu, E. Malavasi, A. Sangiovanni-Vincentelli, I. Vassiliou, “A Top-Down, Constraint-Driven Design Methodology for Analog Integrated Circuits”, Kluwer Academic Publishers, 1997.
[17] Virtual Socket Interface Alliance Official Web Page, www.vsi.org.

Chapter 2
DESIGN OF SENSOR INTERFACES

2.1 State of the art

The designer of electronic devices for automotive applications has to make his way in a fast growing but very competitive market. The key to success resides in the ability to handle the always increasing technology options (from MEMS advances to shrinking BCD processes), developing high performance products and fulfilling all the safety and reliability demands with short time-to-market. This leads to an inevitable trade-off between the need for the highest degree of optimization (in term of area, power and overall performances) and the market pressures for low cost and fast developed devices. While a full optimized ASIC requires noteworthy design efforts and often results in higher development costs (both time and money), a more generic and flexible chip can meet the market demands by addressing at once several similar applications and reduce its costs by having a longer production life. From these basic considerations comes the Universal Sensor Interface (USI) approach, a first answer to the issues concerning the design of sensor interfaces.
The USI implementation proposed in [18] integrates a number of high performance optimized analog front-ends, in order to address the different conditioning needs of the various sensor classes (for example capacitive, resistive or inductive sensors). Using a voltage-to-period converter, the signal coming from the analog front-end is digitalized and a microcontroller or DSP handles the data and performs further digital processing.

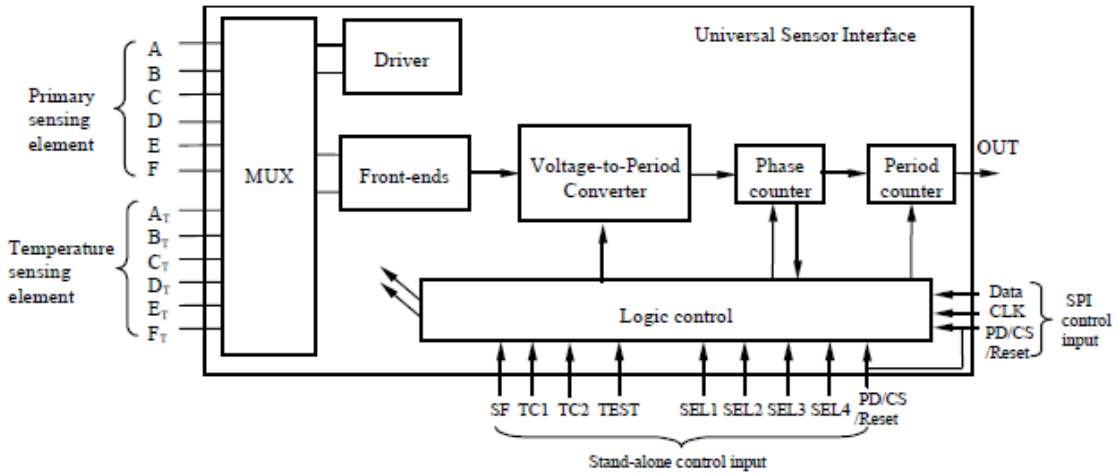

Figure 11: Block diagram of the USI [18].

This approach aims to reduce costs by merging and re-evaluating the functions of the sensors, actuators, analog interfacing circuits and digital processors in overall design [19]. In practice, under the assumption that a complex system needs to process at the same time the information of many different sensors, a lot of conditioning electronics can be shared between several functionalities and thus a global area and cost saving is achieved. Unfortunately, such system does not foresee any customization possibility for specific sensor needs, therefore resulting in good performances but not up to the strict requirements of demanding fields like the automotive one.

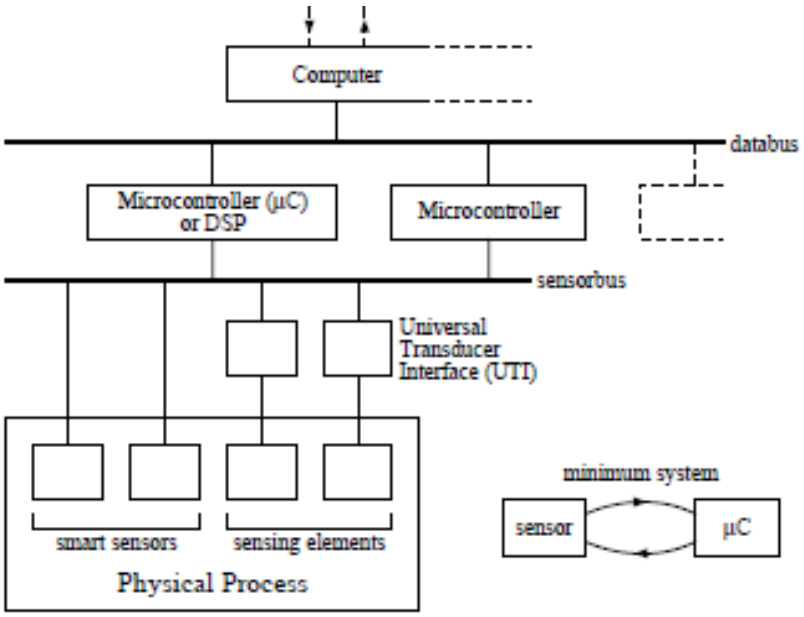

Figure 12: Example of complex sensor system architecture with USI [20].

A similar approach more specifically oriented towards MEMS and called Universal Micro-Sensor Interface (UMSI) [21] expands the capability of the USI circuits. It implements a network-capable sensor bus, integrates programmable readout circuits for multiple signal modes, and includes a standard interface for application-specific peripherals. Up to 255 UMSI chips can be connected in a complex sensor system and each UMSI can interface with up to 8 capacitive, resistive, and/or voltage output devices.
Such implementation eases the construction of a sensor network with numerous nodes (which is indeed a need in several applications, among which also the automotive stands), yet the featured basic and generic analog front-end does not fit the performance demands of modern automotive sensor interfaces.

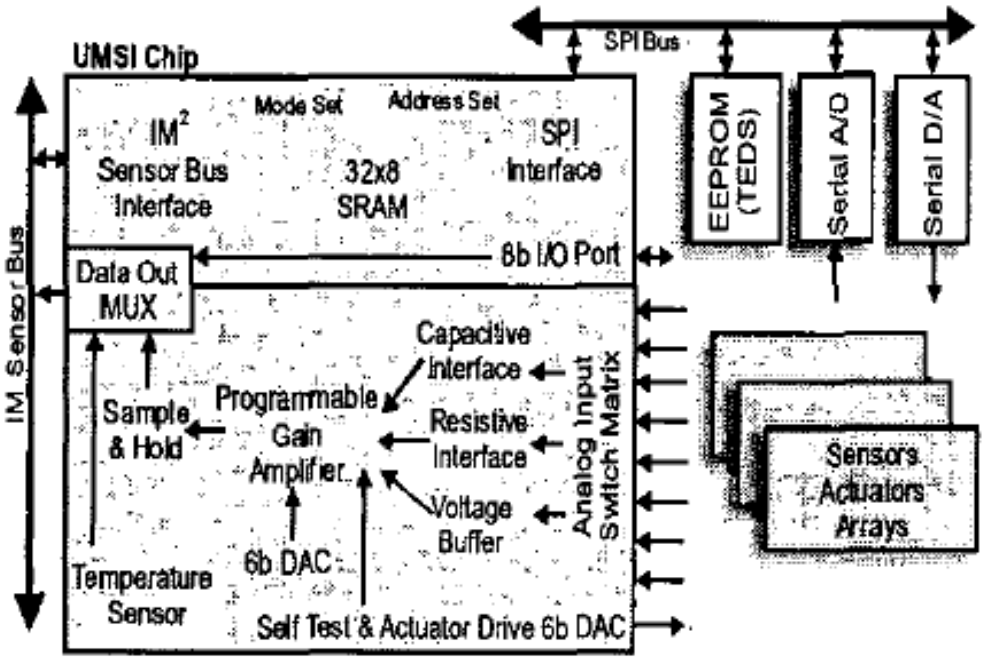

Figure 13: UMSI chip and optional external components.

2.2 Platform Based Design

The technical difficulties exposed in Paragraph 1.6 have not been overcome by approaches like Universal Sensor Interface. In the last years, the progress of silicon technology has been followed by the definition and the increasing application of Platform Based Design [22], a methodology conceived to cope with the most significant issues of modern ASIC development:

1. *Manufacturing cost* depends mainly on the hardware components of the product. Minimizing the size of the chip implies tailoring the hardware architecture to the functionality of the product. However, the cost of a state-of-the-art fabrication facility continues to rise.
2. *NRE (Non-Recurrent Engineering) costs* associated with the design and toolings of complex chips are growing rapidly. The cost of masks will grow even more rapidly for the most shrunk geometries, adding even more to the up-front NRE for a new design. Furthermore, the cost of developing and implementing a comprehensive test for such complex designs will continue to represent an increasing fraction of a total design cost unless new approaches are developed.
3. *Design costs* are exponentially rising due to the increased complexity of the products, the challenges posed by physical effects for deep sub-micron and the limited human resources. Design productivity is falling behind exponentially with respect to the technology advances. Time-to-market constraints are also growing at such a fast pace that even if costs were not an issue, it is becoming plainly impossible to develop complex parts within the constraints. An additional problem is the lack of skilled work force that could implement future IC's considering the system aspects of the design and all second order physical effects that will be of primary importance in deep sub micron.

A *platform* [23] represents a layer in the design flow for which the underlying, subsequent design flow steps are abstracted. Thus a design from conception to implementation can be seen as a set of platforms (abstraction layers) and of methods to transform the design from one platform to the next. Platform based design provides a rigorous foundation to design re-use, correct-by-construction assembly of pre-designed and pre-characterized components (versus full-custom design methods), design flexibility (through an extended use of reconfigurable and programmable modules) and efficient compilation from specifications to implementations. At the same time, it allows to trade-off various components of manufacturing, NRE and design costs while sacrificing as little as possible potential design performance.

In general, a platform is made of a library of components that can be assembled to generate a design at that level of abstraction. The library may contain both computational blocks and communication components and each element of the library has a characterization in terms of performance parameters and supported functionality.

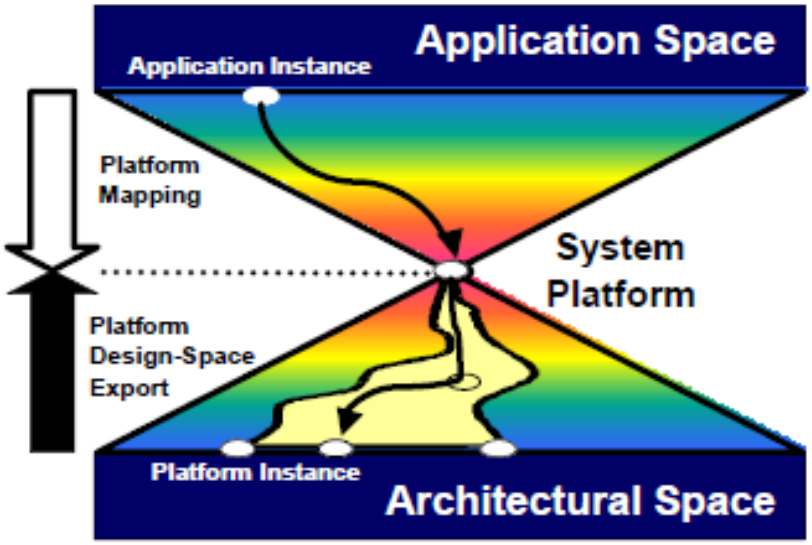

Figure 14: System platform layer and design flow. The system platform effectively decouples the application development process (the upper triangle) from the architecture implementation process (the lower triangle).

For every platform level, there is a set of methods used to map the upper layers of abstraction into the platform and a set of methods used to estimate performances of lower level abstractions.
A platform instance is a set of architecture components that are selected from the library and whose parameters are set, according to needs imposed by specifications.
Platforms should be defined to eliminate large loop designs iterations: they should restrict design space via new forms of regularity and structure that surrender some design potential for lower cost and first-pass success.
A critical step of the platform-based design process is the definition of intermediate platforms to support predictability, which enables the abstraction of implementation detail to facilitate higher-level optimization, and verifiability, i.e. the ability to formally ensure correctness.

2.3 Generic platform for inertial sensors

The theory of Platform Based Design expressed in previous paragraph has been employed in the development of a platform for inertial sensors conditioning in automotive applications. With respect to the USI approach, the aim is not to implement a single chip which can address a wide range of inertial sensors in MEMS technology (as it would result in a lower level of performances and an overhead of resources that the automotive field cannot afford),

but to keep the generality only at the system platform level (thus only as a design concept instead of a silicon implementation), while the highest level of optimization can be pursued in the platform implementations (each one dedicated to a target sensor). Figure 15 basically resumes the development of a Universal Interface: according to the blocks needed for conditioning each sensor class, they are integrated into the USI, which can save some hardware as few blocks can be shared by more than one sensor application. Of course some of them can't be the optimal choice for specific conditioning purposes, and others won't be used for certain application, therefore resulting in area and power overhead and increased parasites.

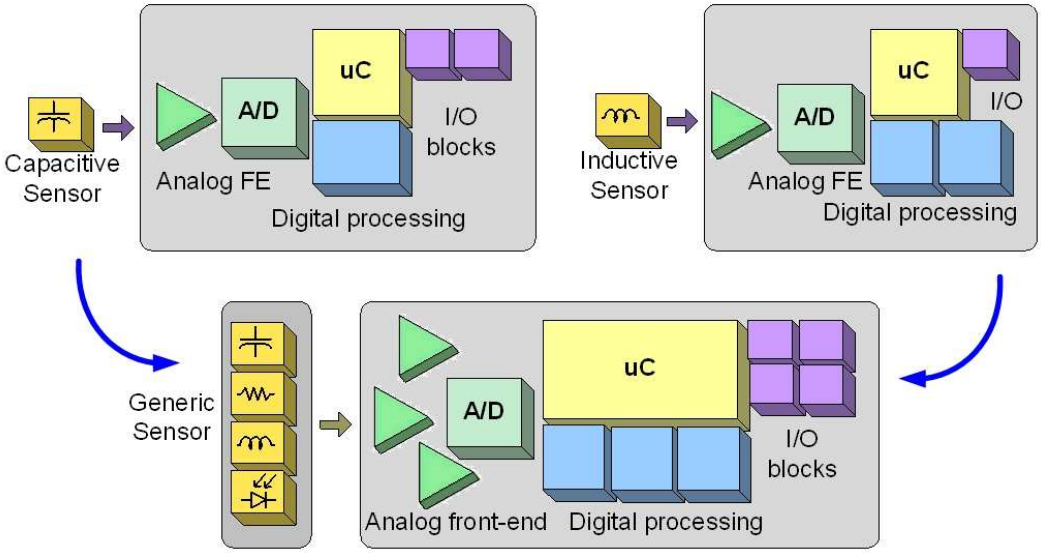

Figure 15: Building a Universal Sensor Interface

In the platform based approach instead [24], all the required blocks for different applications are collected inside the generic platform as a library of building blocks, which aim to be part of the signal processing path for inertial sensors. This platform represent the highest level of abstraction and thus its components are described as behavioural blocks in a proper language (for example MATLAB™), without any specification about their physical implementation (for instance whether they will be hardware or software, analog or digital). The platform itself doesn't even contain any indication of the processing architecture, as this choice will be made at a further step considering the target application.

2.4 Design flow

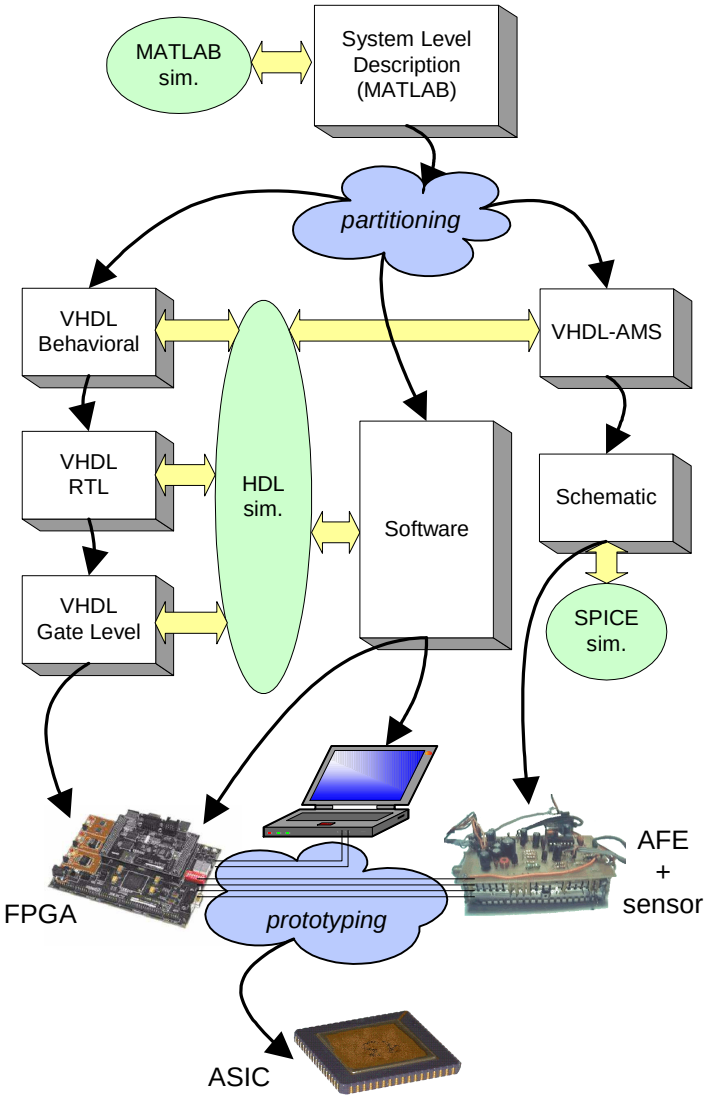

Figure 16: Design flow.

In the sensor conditioning field, the use of MATLAB™ language for the system platform eases the design space exploration phase and the subsequent tightening of each component specification before passing to the next platform at a lower abstraction level. In fact, the target sensor itself can be modelled and employed in simulations to evaluate the best conditioning architecture. In this way the computational blocks can be first generically modelled in their behaviour and then better defined through a set of specifications: for example, a basic multiplier block during the architecture evaluation can be provided step by step with more specification (for example output saturation limit, input dynamic range, etc…) in order to gather enough information for deciding

the best implementation for the block (analog or digital) and setting the final specifications (in the digital case for instance, the proper bit true model, clock frequency and data-paths width).
The aim of the MATLAB™ simulation at system level is to find a suitable conditioning architecture for the target sensor and perform a proper partitioning of the used blocks between analog cells, digital hardware and software routines. If the MATLAB™ evaluation has been comprehensive enough, the partitioning entails:

- the matching of in-detail-specified (but implementation independent) blocks with the nearest (for functionality and requirements) already available IP.
- the design from scratch of the remaining IPs (in the case none of those available fulfils the block requirements).

Except for the first run, a platform based design flow addresses a set of similar applications which are supposed to share several functionalities. For this reason, in this phase the reuse of existing IPs prevails on the development of new blocks and the more this design technique is exploited (i.e. the more applications are addressed), the less is needed to create new IPs.
In both cases anyway, at the HDL abstraction level, each block has to provide different views in order to guarantee the possibility to verify step by step the functionality of the block itself together with the overall system architecture. For example, digital hardware IPs need first of all a behavioural description (in VHDL) which resumes the block functionality hiding the details of the internal architecture and the possible implementation choices. The corresponding descriptions for the analog domain are VHDL-AMS models [25][26], which allow a first system level simulation (at the HDL abstraction level) that must match the results of the MATLAB™ investigation, thus validating the architectural and partitioning choices.
The use of VHDL-AMS has two main purposes. First, it summarizes the block specifications in a top-down approach before the design of the analog cell, constituting a reference for the analog designer and allowing a preliminary system validation during the development of the block (so the system level verification environment can be set up before the completion of each single block implementation). Once the schematic is ready, the same model can be updated with more detailed and realistic information coming from SPICE simulations, in a bottom-up data flow. The model can be then completed at the highest level of detail after layout, allowing the platform verification with even higher accuracy.

The main benefit of the VHDL-AMS approach is the possibility to check, at each stage of the design progress, the coherency of the whole system with respect to the initial behavioural model. In fact, though mixed signal HDL/SPICE simulation are supported by most tools, they result to be extremely slow (and may have convergence troubles) as the analog complexity raises over hundreds of transistors. VHDL-AMS instead, allows the extraction from the huge complexity of the schematic of only few parameters which depict the cell behaviour, in a way that the system simulation is dramatically speeded up but with no substantial loss of global accuracy [26].
Another important feature of the VHDL-AMS modelling is the possibility to detect bugs, shorts and system faults that can't be seen by simply simulating cells or macro-blocks. A mixed-signal sensor interface parts itself almost equally between analog and digital sections. Each analog cell has tens of configuration switches, added to the routing configurability of input and test signals, which lead to a huge number of configuration strings, some of which may cause dangerous (or even killing) shorts or misbehaviours. Being as a matter of fact impossible to check all the possible configurations manually, VHDL-AMS allows the insertion of safety checks within each model (for example current or voltage over boundaries on a wire or node) that signal the potential bug during simulation, helping to implement a first time success silicon.
After the platform verification by means of VHDL/VHDL-AMS/C++ system simulations, the HDL abstraction level can be mapped to last developing stage before silicon, the prototype. In this phase, the needs of verification may depend strongly on the application and the implementation choices made at the higher levels. For instance, in case the design results to be mostly digital oriented (i.e. analog section much smaller than the digital one) the verification of the digital part assumes a predominant role and a suitable strategy can be the emulation of the analog functionality with a discrete-components front end, connected with a FPGA implementing all the ASIC digital blocks plus few add-ins for further test, evaluation and debug. This approach reduces development time and cost, but does not cover in a comprehensive way all the possible interactions between analog and digital domains. Moreover, it is not fit for half-digital-half-analog (or mostly analog) designs and the prototype can provide little information on final ASIC performances, because of the excessive level of parasites and disturbs in the discrete components setup with respect to the corresponding integrated analog front-end. In order to have a realistic and reliable prototype (for what concerns

performance estimation and feedback information for further improvements) the only possibility is to integrate the analog section into a test chip, at the cost of a substantial increase of design expenses and stretching of time-to-market. An effective solution to this drawback will be described in Chapter 3.
The prototyping phase concludes the design iteration with a bottom-up propagation of all the data collected by the prototype working with the target sensor, in order to update the HDL description and VHDL-AMS analog models and, in the end, the original MATLAB™ blocks, allowing a system evaluation with coherent results at each abstraction level. If the system specifications haven't been met, improvements are to be evaluated and carried out at the top level and then propagated down to the physical implementation.

2.5 Case study: gyro sensor

In this paragraph a case study for the above mentioned platform based design flow is presented. The generic platform architecture has been customized for conditioning a gyroscopic angular rate sensor, and implemented in collaboration with Sensordynamics AG for a commercial chip.

i. Gyro sensor

Vibrating ring gyroscopes consist of a circular ring provided with drive, sense and control electrodes [27]. Even though they are also available as discrete components, automotive applications push towards MEMS implementations [28], requiring stand-alone devices able to provide accurate yaw rate measurement.

Gyro sensors base their working on the Coriolis force acting on a vibrating mass: while the driving electrodes keep the ring vibrating along the primary direction (with fixed amplitude, around the z-axis in Figure 17 [28]), the rotation of the device (Ω_x) causes an energy transfer to the second vibrating mode, which is located at 90° from the primary mode (on the y-axis). The amplitude of this vibration is proportional to the angular rate, and in open loop mode it can be detected through a differential capacitance measure on the sense electrodes placed on the rectangular structure. A closed loop configuration exploits the control electrodes, by means of which the secondary vibration can be compensated, in order to let the sensor work around its rest point, thus achieving more linearity and accuracy.

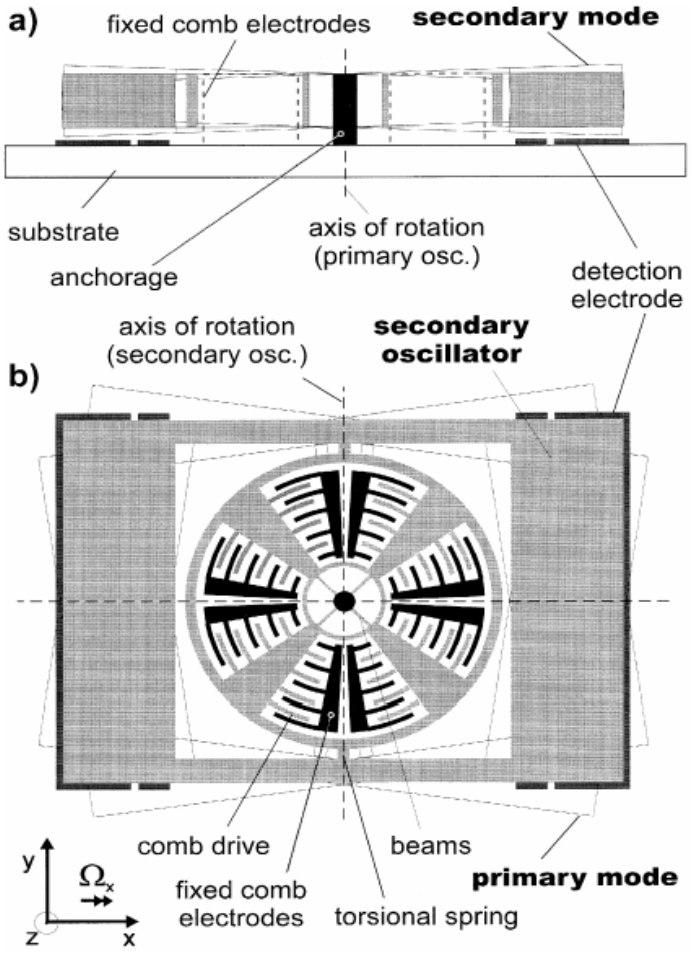

Figure 17: Gyro sensor: Cross-section (a) and top view (b).

The conditioning electronics for such sensors basically requires a Phase Locked Loop (PLL, for primary drive), which has to keep the ring in resonance (at a frequency of approximately 6 KHz), an Automatic Gain Control (AGC, to control the amplitude of this vibration) and a chain including demodulators, filters, temperature/offset compensation and modulators for secondary drive and rate sensing.

ii. System architecture

The generic platform architecture depicted in Figure 18 has been customized to the gyro sensor conditioning requirements, in order to achieve an almost optimal implementation for interfacing this kind of sensor, together with several communication resources for easier debugging, development and performance improvement.
For what concerns the analog front end, it can be critical in automotive applications to meet the required noise, area and power consumption constraints within a wide temperature range (-40°C÷125°C). These limitations do not apply to the digital

counterpart: for this reason it's preferable to use as little analog signal processing as possible and to perform most of the conditioning in the digital domain. Thus the analog section only absolves functions of driving sensor's electrodes (through couples of DACs for each loop) and performing signal acquisition (by means of SAR ADCs, amplifiers and low pass filters). It also provides stable voltage and current references and clock to the digital section.

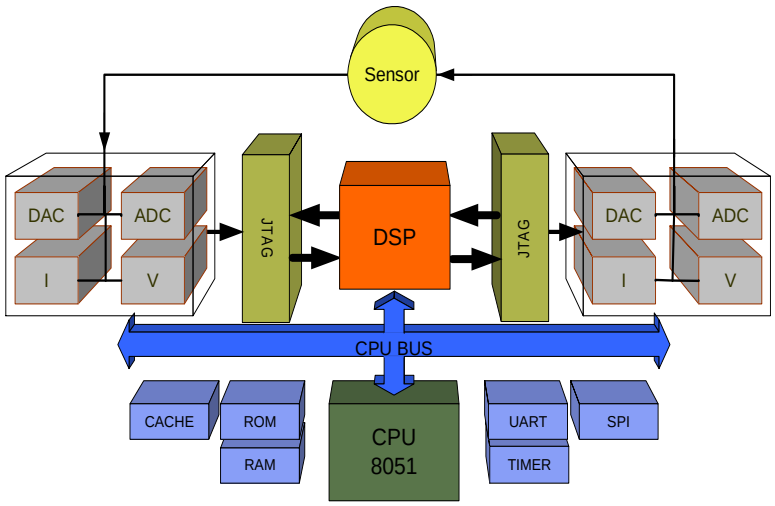

Figure 18: Platform architecture

Each analog cell in the front-end is digitally controlled, and this programmability can be of paramount importance for the whole system functioning. Particular effort has been placed in building a reliable and efficient interface between the analog and digital world. A JTAG-like interface has been selected for the following reasons:

- The standard protocol has largely been studied and tested for several applications, so it guarantees high reliability since first implementation.
- JTAG bases on asynchronous communication, which limits clock skew issues that may lead to undetectable faults.
- It employs a short number of wires (only 4 per chain), thus resulting easy to route also on very complex chips.
- It allows for full read-back capability (for fast verification or debugging).

The digital section performs almost all the signal processing (within the DSP block), and manages the communication resources (through the CPU core). The DSP contains a chain of IPs for signal elaboration, featuring the blocks schematically depicted in Figure 19: its architecture and sub-blocks dimensioning are derived from the MATLAB™ model set up during the design space exploration phase.

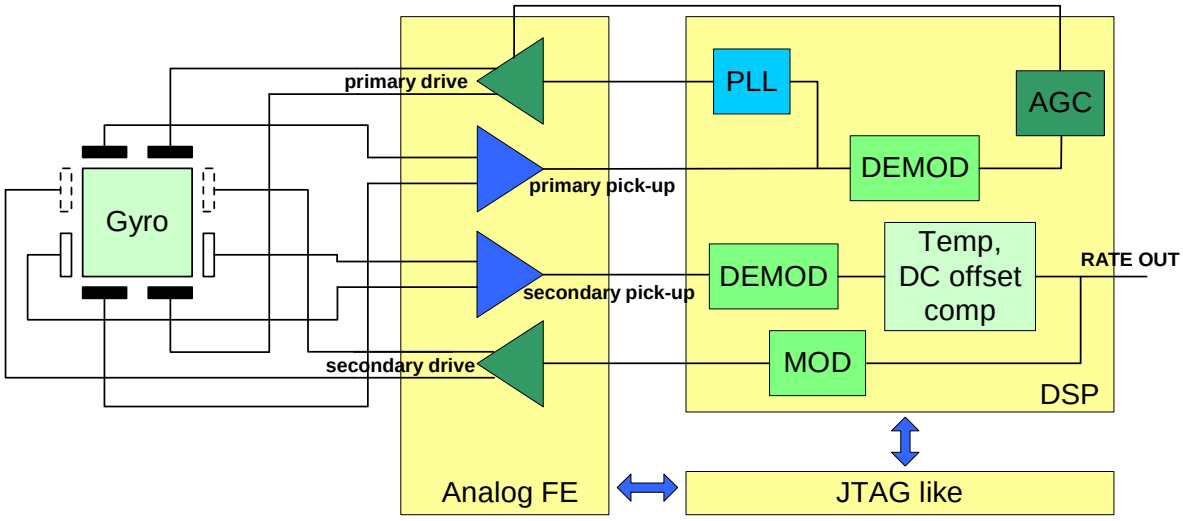

Figure 19: DSP chain for gyro sensor conditioning.

Though all the required processing can be performed independently by analog front end and DSP block, a CPU core is also present to fulfil control, monitoring and communication tasks. These functionalities naturally find their best implementation via software routines, as they may vary through system updates and new system requirements.

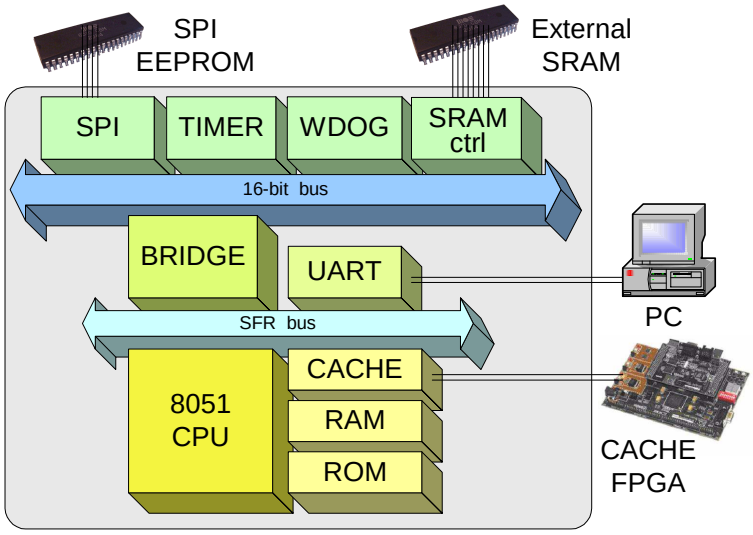

Figure 20: CPU core architecture with communication resources.

Control and monitoring are performed real-time by the processor on both DSP and analog front end: a routine constantly checks the system status by accessing the several readable registers spread along the processing chain (for example makes sure that the PLL is locked). Meanwhile other routines handle communication services,

providing status and output data to the user: during prototyping phase, the system can be linked to a PC and through a graphical interface manual trimming can be performed and all intermediate data of the chain can be accessed. The same communication resources are used during the chip normal working conditions to provide external devices (typically the car's electronic control system) the required angular rate measure and status information.
As shown in Figure 20, CPU core architecture comprises the Oregano 8051 processor [31] (freely distributed under LGPL licence), which provides a good compromise between performance and area occupation, and fits well the mentioned microcontroller applications; it is provided with ROM and RAM memories and cache controller, all configurable (both with hardware generics and at run-time) in order to get the maximum flexibility for software download, development and update. Just to give a few examples, an 'ASIC' version could include a big ROM (16 Kb) with all the needed software (the latest available at the moment) and through the cache (which is conceived to access big external RAM with a custom 2-wire protocol) newer software versions could be downloaded and tested; in a 'prototype' version, a big RAM would be instantiated and used as Program Storage (while the boot placed in a small 1 Kb ROM would perform software download via UART) and cache would not be instantiated. Software download is also possible by means of RS485 (in place of simple RS232 protocol implemented by the UART) and SPI (at start-up all the communication devices look for a response on their channel, in a way that the connected peripheral is automatically detected); moreover it's possible to store the downloaded software into an external SPI EEPROM, and so reboot directly from EEPROM instead of downloading each time after reset. This high configurability gives the designers the chance of developing software with maximum simplicity and efficiency, through the whole system prototyping phase and even after the first ASIC releases, in order to achieve the utmost confidence on the final product success. Cache controller and UART are located on the 8051 Special Function Register (SFR) Bus (8-bit), while the other peripherals (SPI, timer, watchdog, and SRAM controller) are accessed via a custom bridge by means of a 16-bit bus. SRAM controller is used during the prototyping phase, to store at real-time (into a 512 Kb SRAM) digital data coming from any node of the DSP chain, with chance of later read-back for analysis purposes.

iii. Results

The platform based design flow has been carried out step by step as described in paragraph 2.4: simulation of the entire system (in particular sensor locking) has been first performed with MATLAB™ model (Figure 21 depicts main PLL signals), and then the same result has been reached in a HDL simulation environment, thanks to the full VHDL-AMS modelling of the analog section and the sensor itself. After this important achievement, designers have been able to get to the final mixed-signal platform implementation in a short time, being already defined block partitioning, dimensioning and connections.

The prototyping phase has then proved the validity of this approach, and an emulation environment has brought the target sensor to locking (see Figure 22 for measured PLL data) and output yaw rate data. The digital part of roughly 200 Kgates complexity has been implemented in a Xilinx X2S600E running a 20 MHz clock frequency, while the analog front-end has been integrated into a 12 mm^2 custom chip in 0.35 µm CMOS technology.

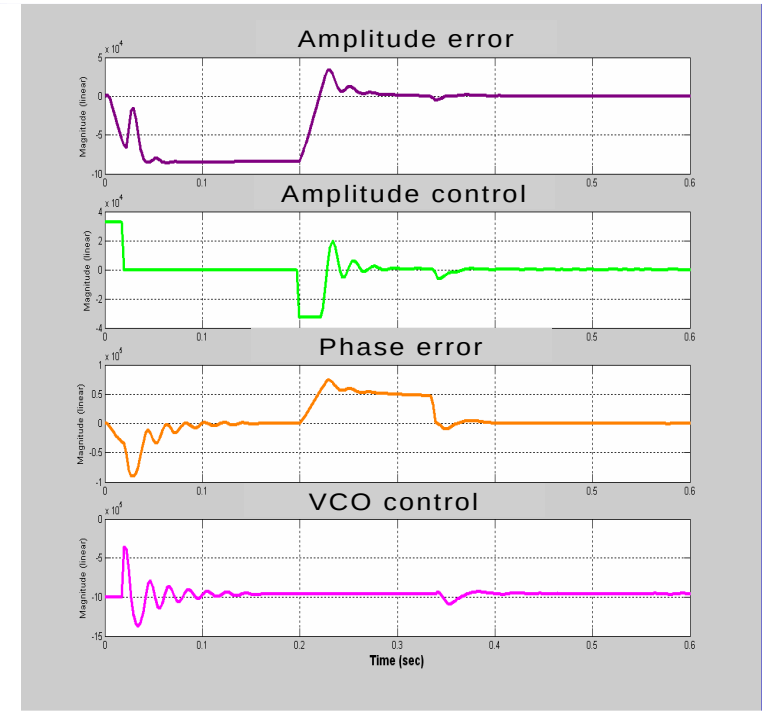

Figure 21: Waveforms of PLL locking (MATLAB).

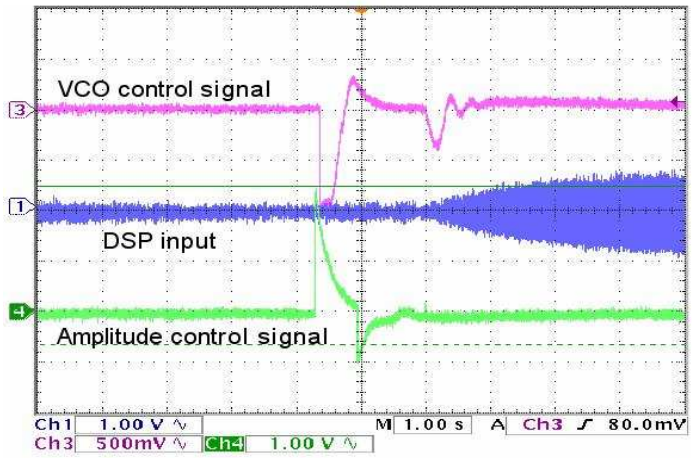

Figure 22: Measured waveforms (AC probe).

Performance of higher level (as reported in Table 4, in comparison with other commercial gyro sensors of Table 5 and Table 6) have been achieved with the integration of the whole system in a single mixed-signal chip in 0.35 µm CMOS technology.

Parameter	Sensordynamics			Units
	Min.	Typ.	Max.	
Sensitivity				
Dynamic Range	+/- 75		+/- 300	°/s
Initial	4.85	5.00	5.15	mV/°/s
Over Temperature	4.80	5.00	5.20	mV/°/s
Non Linearity	0.07	0.10	0.20	% of FS
Null				
Initial	2.70	2.50	2.53	V
Over Temperature	2.70		2.53	V
Turn On Time		500.00		ms
Noise				
Rate Noise Dens.	0.02	0.027	0.035	°/s / √Hz
Freq. Response				
3 dB Bandwidth	25.00		75.00	Hz
Temp. Ranges				
Operating Temp.	- 40		+ 85	°C

Table 4: Performance of Sensordynamics implementation.

Parameter	Analog Devices			Units
	Min.	Typ.	Max.	
Sensitivity				
Dynamic Range	+/- 300			°/s
Initial	4.60	5.00	5.40	mV/°/s
Over Temp.	4.60	5.00	5.40	mV/°/s
Non Linearity		0.10		% of FS
Null				
Initial	2.30	2.50	2.70	V
Over Temp.	2.30		2.70	V
Turn On Time		35.00		ms

Noise Rate Noise Dens.		0.1		°/s / √Hz
Freq. Response 3 dB Bandwidth		40.00		Hz
Temp. Ranges Operating Temp.	- 40		+ 85	°C

Table 5: Performance of ADXRS300 [29].

Parameter	Murata			Units
	Min.	Typ.	Max.	
Sensitivity Dynamic Range Initial Over Temperature Non Linearity	 0.54 - 5.00	 0.67 	 +/- 300 0.80 + 5.00	 °/s mV/°/s mV/°/s % of FS
Null Initial Over Temp. Turn On Time		1.35 - -		 V V ms
Noise Rate Noise Dens.		-		°/s / √Hz
Freq. Response 3 dB Bandwidth			< 50	Hz
Temp. Ranges Operating Temp.	- 5		+ 75	°C

Table 6: Performance of Murata's Gyrostar® [30].

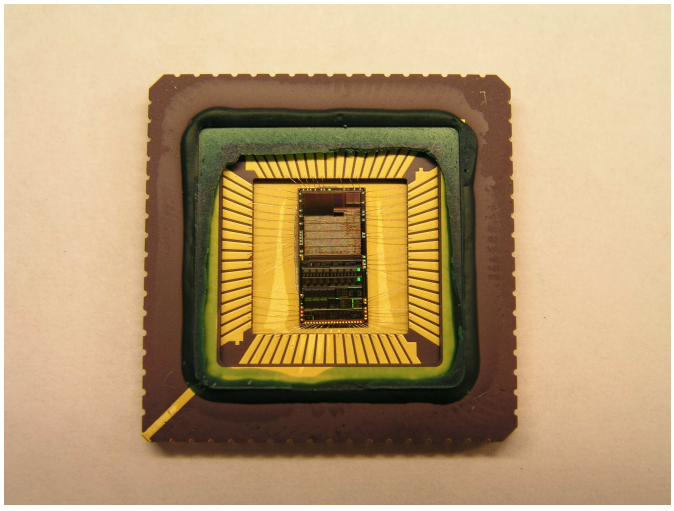

Figure 23: Macro photo of the single chip ASIC implementation.

2.6 IP development for sensor interfaces

In this paragraph a few IPs are presented. They have been developed as reusable designs with the aim of being easily

integrated in the Platform Based Flow described in Paragraph 2.4. One of the keys for success of the Platform Based approach resides in the availability of a wide portfolio of IPs, among which the fittest for each particular application can be chosen (without the need to design them each time from scratch) and in the possibility to easily and quickly configure the existing blocks in order to allow their employment in as many applications as possible.
These digital IPs can be integrated in the system either for implementing the required ASIC functionalities or for test and debug purposes during the customized platform development.

i. sd_8051_cache

The 8051 Cache is an IP designed to replace a big code memory (PSRAM or PSROM) with a smaller on-chip SRAM and an off-chip EEPROM (or SRAM) accessed serially. The purpose of such cache memory is to extend the capabilities of firmware development for embedded systems, which usually cannot afford big code memories because of area and power constraints. This cache for 8051 processors gives the possibility to extend the Program Storage memory up to the full 8051 code address space (64 Kb), by physically implementing on chip only a small SRAM and accessing the external code memory through a custom serial protocol which uses only 2 wires (thus not affecting badly also the critical constraint on ASIC's max number of pins).
The 8051 CPU works in the same way as in presence of the PSRAM, issuing addresses and enable signal, and expecting data before the following rising clock edge. Cache looks for the data in its memory block and forwards it to the processor (if valid), otherwise freezes the processor and loads data serially from the external device.
To let Cache and CPU work properly, the processor must have a freeze pin that halts CPU operations when active. If this pin is not present, the freeze signal must gate the CPU clock, to prevent any possible malfunction.
Serial communication with the EEPROM is intended to minimize the number of wires and to keep a data rate as high as possible: this is done with a 2-wire synchronous custom protocol (clock and bidirectional data), that is much faster than standard SPI or I²C, but requires an external interface (*sram_interface*) to latch data and drive the second level memory (like EEPROM or SRAM).
The external device signals its presence after reset (according to a simple protocol); in case it is not present, cache works in memory mode, allowing the processor to use its SRAM as another PSRAM bank, located in the address space left free by standard PSRAM.

Main features:

- Compliant to 8051 CPUs.
- Fully parametric VHDL description, allows quick architectural changes and dimensioning.
- Custom serial protocol (with open collector driving of the bidirectional data line): easy to implement and gaining high performances.
- SFR reduced interface allows to choose polarity of serial data sampling (negative or positive edge).
- Includes power saving features (CPU and its memories are not clocked during miss time).
- Memory mode: can be seen by CPU as PSRAM when no off-chip memory is present.

Cache has to deal with two main functionalities:

- Interface the CPU, providing data for each address issued by the CPU.
- Serially download new blocks from the external device in case of cache miss.

In case of cache miss the CPU keeps receiving the last valid code byte, while the assertion of signal *CPU_freeze* prevents all the blocks (except *cache*) from being clocked. The absence of clock and the constancy of inputs (from both ROM and RAM) guarantee the processor to resume its functionality in the correct way after the block replacement (and subsequent cache hit).

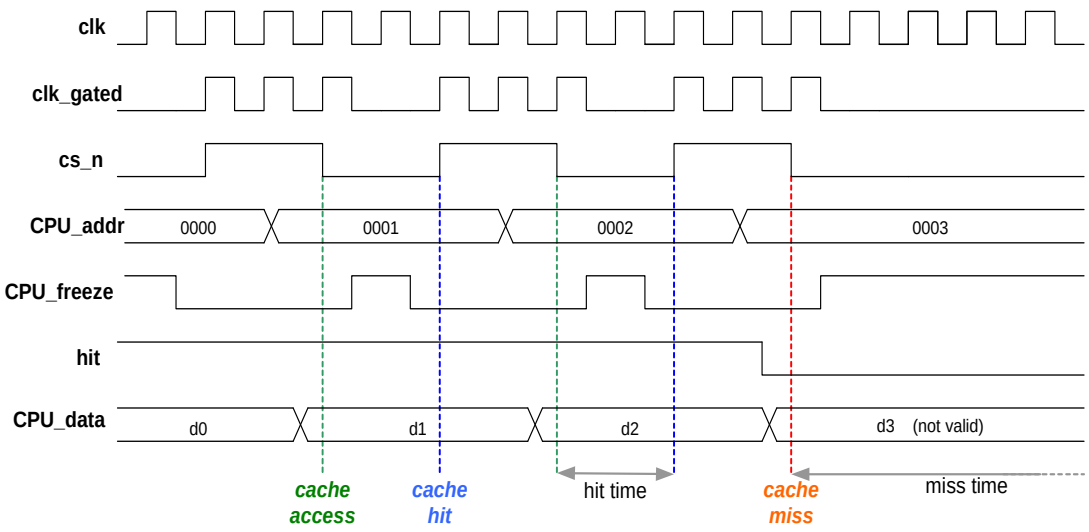

Figure 24: cache hit and cache miss timing diagrams

The time diagram in Figure 24 shows how CPU data is managed by the cache, during hits and misses. In this example addresses

“0000”, “0001” and “0003” are supposed to generate cache hits, while “0004” generates a cache miss and freezes the CPU: notice that hits and misses are evaluated on the negative edge of *clk* (as cache SRAM is clocked on the negative edge) allowing the issue of signal *CPU_freeze* before the following rising clock (in which the CPU would read invalid data). Notice also that a wait cycle (with *CPU_freeze* active) is inserted in every cache access (resulting in a hit), in order to simplify clock gating structure: for this reason cache hit time is 2 clock cycles.
Miss time, instead, is controlled by the user (within certain limits) through the cache configuration: by specifying a value for the constant *N_bytexblock* is not only modified the cache memory layout, but also the length of the serial stream sent to the cache after each miss.
The following diagram illustrates the serial custom protocol implemented by entities sd_8051_cache and sram_interface (the part of sd_cache_fpga IP that handles the serial communication protocol as a slave). It works on two wires only, a serial clock (sclk, equal to system clock) and a bidirectional data line, driven (as open collector) by master and slave alternatively. 8051 Cache is the master, with the exclusive right to start a transaction by driving to zero the data line. After the start bits, it transmits the (16-*N_bit_offset*) MSB of the address (relative to the first location of the EEPROM to be read). No information relative to the number of bytes to be read is issued, as the slave inherits this parameter from the cache package. The off-chip interface latches the address and reads sequentially *N_bytexblock* bytes; then transmits the whole new block serially (still starting with 2 start bits) through *sdata* line. The first start bit is always a zero and is used to signal the beginning of a stream. The second bit is a ‘1’ if the previous stream has been received correctly (i.e. has passed the parity check) and communication can go on normally. If it’s a ‘0’ (parity fault), the talker stops driving the line, and waits for the last stream to be retransmitted.

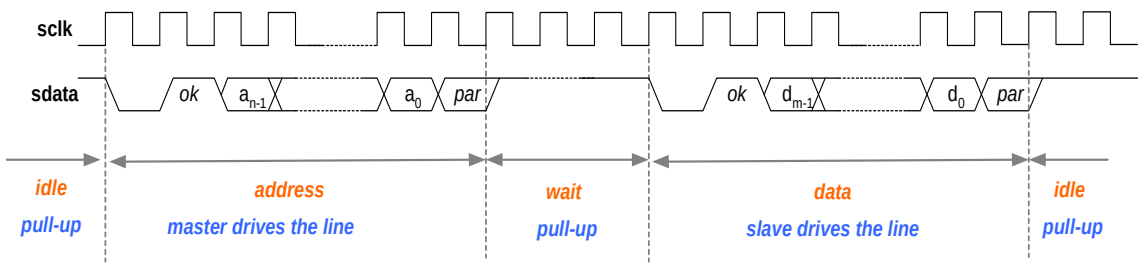

Figure 25: example of serial protocol.

From the above diagram we can calculate miss time (in clock cycles) as:

$$T_{miss} = 2 + 16 - \log_2 N + 1 + \alpha \cdot N + 2 + 8 \cdot N + 1 = 22 + (\alpha + 8) \cdot N - \log_2 N$$

With $N = N\_bytexblock$ and $\alpha =$ EEPROM/SRAM read access time (in clock cycles, usually 1).
For instance, in the cache implementation of the customized platform for gyro sensor, total miss time is 95 clock cycles (91 for the above mentioned stream, plus states for miss detection and data storage/providing).
Internal cache can be configured with two different architectures: direct mapped or 2-ways set-associative. The other configuration parameters (cache dimension and number of bytes per block) can be set independently from the chosen architecture, while these three parameters together affect the derived dimensioning constants, like tag and index bits and SRAM blocks width.

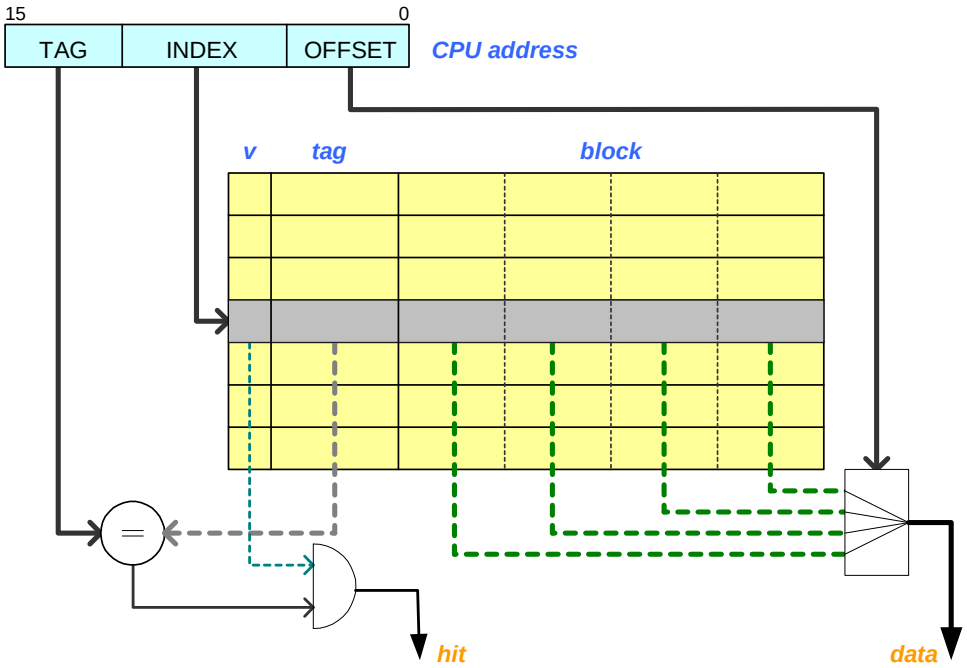

Figure 26: direct mapped architecture.

The direct mapped is the simplest cache architecture, featuring minimum complexity and SRAM employment. A schematic representation of this architecture is given in Figure 26.
CPU address is split into three components:
- OFFSET selects one byte within a block. It's expressed on log2(*N_bytexblock*) bits (starting from LSB).

- INDEX selects one cache row (v + tag + block). It takes log2(*N_block*) bits, starting from the first not used as OFFSET. *N_block* (number of blocks) is calculated as cache dimension / number of bytes per block.
- TAG bits are all the other address bits not used as OFFSET or INDEX.

Since cache dimension is always minor than address space (64 Kb for 8051 CPU), a direct mapped architecture maps many memory blocks to the same cache row (i.e. all those with the same INDEX). The number of these blocks is just the ratio between memory and cache dimensions. For this reason the TAG bits of the CPU address must be compared to those of the block stored in cache: if they're equal (and if valid bit is set) the cache access results in a hit. Valid bit is initially cleared for all rows (during flush state, after reset): then every new block loaded into cache has a valid bit set to '1', being it valid data. By this moment it's not possible to invalid data present in cache (flush operation), neither via software nor hardware. The above shown logic (SRAM, AND gate and comparator, without multiplexer) is included in component called 'cache', which is instantiated both in direct mapped and associative architectures.

Once set cache dimension and number of bytes per block, a 2-way set-associative cache can be seen as two sets, each containing half of the blocks that would be in a direct mapped cache.

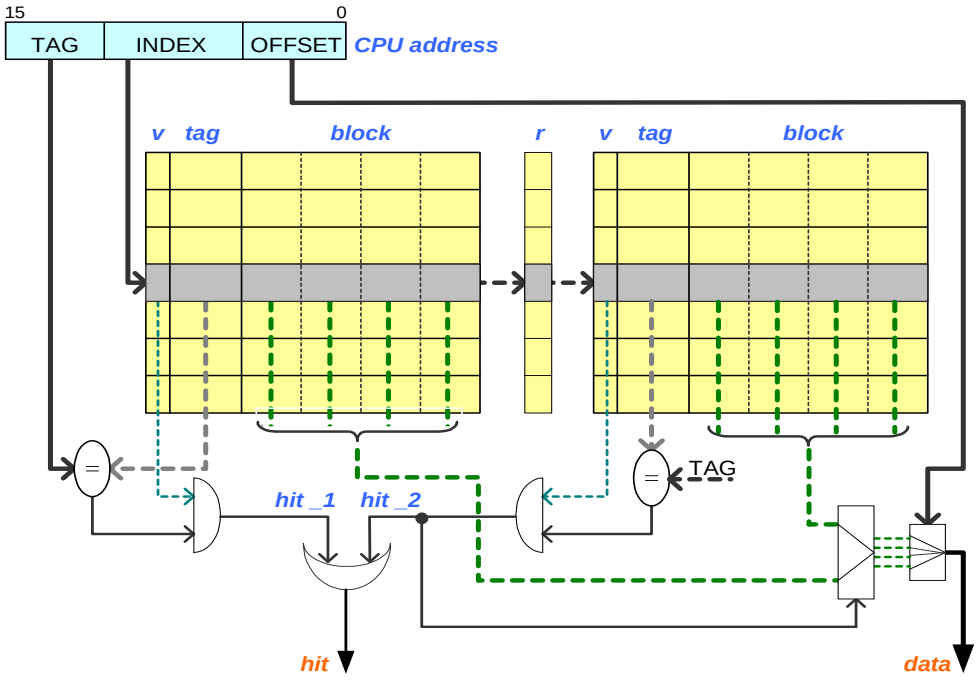

Figure 27: 2-way set-associative cache architecture.

This means that two 'cache' components are present: they receive the same index, and both send out their data and partial hit result. Global hit is the OR of the two partial hits, while global output data is selected in a bigger multiplexer between partial outputs, depending on OFFSET and partial hits.
This architecture employs more resources than direct mapped: with the same cache dimension (i.e. total number of blocks) it has 1 bit less of INDEX and thus 1 bit more of TAG; so it requires bigger SRAM, two bigger comparators (instead of one) and a bigger output multiplexer. SRAM has to be wider not only for the increased length of tags, but also for the presence of replacement bit (r), necessary to indicate which is the last set used (for each row) and so what is the block to be replaced in case of cache miss.
The higher area occupied by this architecture is rewarded with an increase of performances, identified by a lower miss rate and subsequent lower mean access time (TA). This is due to the higher flexibility of cache storage (with respect to Direct Mapped architecture): two memory blocks with the same INDEX can be both stored in cache (and reused many times by processor) without having to be replaced every time. This advantage is more significant for smaller cache dimensions, as there are in this case many memory blocks mapped to the same cache row.

ii. sd_cache_fpga

This IP is intended for being implemented on FPGA and provides a 32kb (or bigger, depending on the FPGA available resources) off-chip SRAM for the 8051 cache. This SRAM has to be filled up with the code that will be executed by the ASIC: for this reason it is provided with CPU, UART, and embedded boot firmware which handles the download of software at power up through the standard 232 or 485 protocol. A memory wrapper lets CPU write access the memory after reset (for code storage); when download is finished, CPU disables itself by writing in a proper SFR, and the wrapper lets the *sram_interface* (a serial/parallel converter, implementing a slave in the cache custom protocol) read-access the memory. The wrapper has also the task of driving the reset pin of the ASIC (releasing it after download is completed) and issuing the identification stream (defined in the package) through the serial data line, so letting *sd_8051_cache* detect the presence of this FPGA. To keep *sd_sram_interface* and *sd_8051_cache* completely synchronous, this FPGA uses as system clock the serial clock output pin of the ASIC.
Main features:

- Compliant to *sd_8051_cache*: same serial custom protocol and same package for dimensioning.
- Fully parametric VHDL description.
- SFR reduced interface allows to choose polarity of serial data sampling (negative or positive edge).
- Power saving feature: CPU and its memories (PSROM, IDATA) are not clocked after software download.

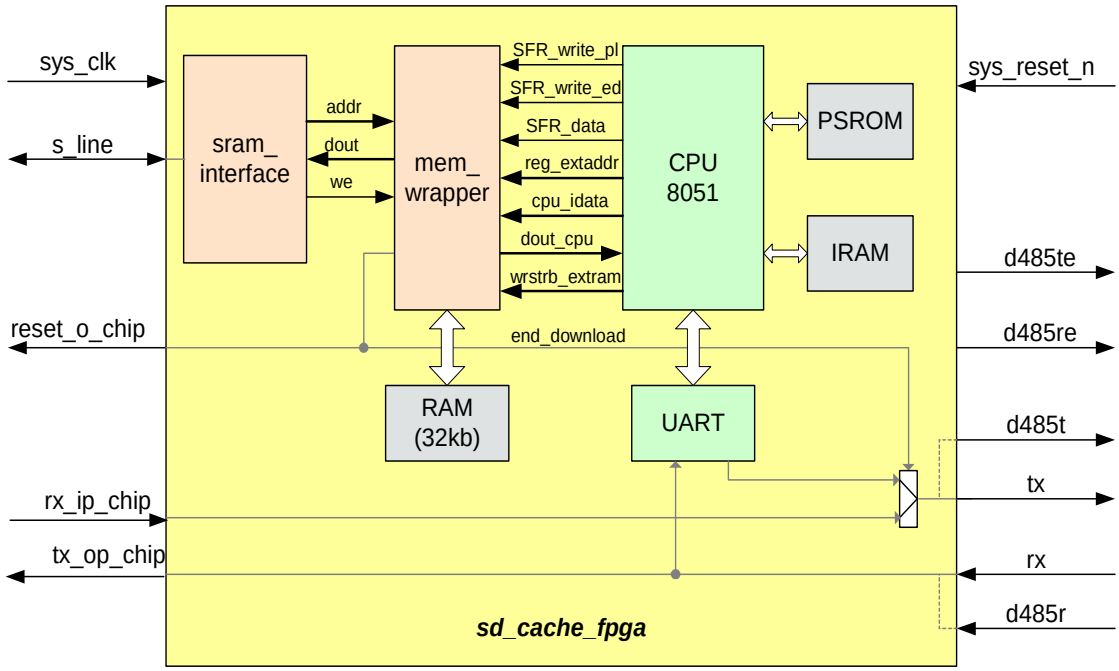

Figure 28: sd_cache_fpga block diagram.

The CPU inside sd_cache_fpga is provided with a minimum number of peripherals (UART, 256 bytes IDATA and 512 bytes PSROM containing boot firmware), just those needed to correctly perform serial code download (either through 232 or 485, selection made by default value of signal *pio_switch*). The wrapper implements two SFRs: *polarity*, used to choose between negative and positive clock edge for sampling (and issuing) serial data on pin *s_line*, and *end_download*, which is set by CPU after download is completed. The signal *end_download*, output of this SFR, controls who can access the 32kb memory (CPU or *sram_interface*), which UART drives the *tx* pin (the one from *sd_cache_fpga* or the one from ASIC) and is connected to the freeze pin of CPU 8051 (gates CPU clock and disables IDATA and PSROM when active).
Figure 29 depicts the activation procedure of the cache modules at power on. The user PC (containing the software to be run by the ASIC) is connected via UART to the *sd_cache_fpga*. The FPGA receives the CLOCKOUT signal from the ASIC but drives to '0' its RESETN pin keeping the chip idle. After the FPGA memory has

been filled with the software, the *end_download* signal switches the access control to the memory and connects the UART from the ASIC to the PC serial interface. At the same time the RESETN pin is driven to '1' and thus the ASIC starts its own power up routines (mostly hardware Built In Self Tests on ROMs and RAMs). If BISTs are ok, the bidirectional CACHE line is released (pull-up) and the *sd_cache_fpga*, after sampling the '1' level, issues an identification stream that enables in *sd_8051_cache* the active mode, thus masking the ASIC PSROM or PSRAM and providing the code to the CPU after serial communication with the FPGA slave.

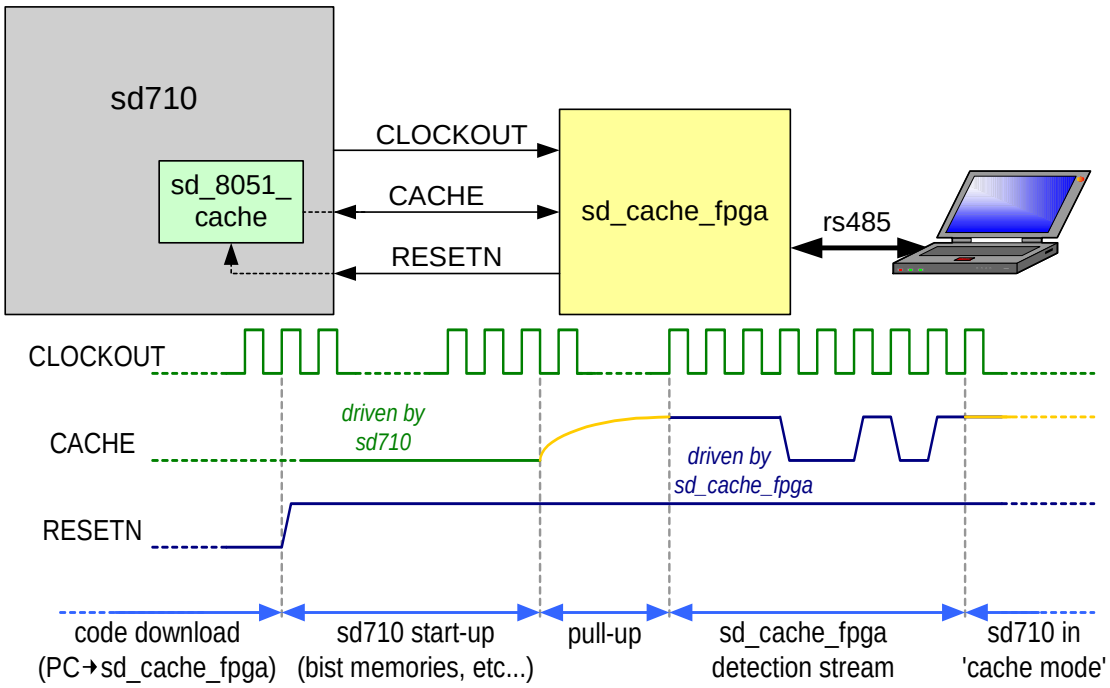

Figure 29: Handshake at start-up between *sd_8051_cache* and *sd_cache_fpga*.

This handshake has been conceived to guarantee the maximum flexibility and the minimum overhead due to the presence of the cache. In this way, when the cache is not needed it automatically deactivates and its memory is available as additional PSRAM, while the two pins for serial communication (CACHE and CLOCKOUT) are free to be driven by other IPs.

iii. sd_SRAM_controller

This memory controller is an interface designed to drive an external SRAM memory, with two main features: write (and read back) 16-bit words from CPU and store into memory digital data probes. The purpose of this IP is to be instantiated into a design

with a multi-stage DSP chain allowing a software controlled probe of the monitored nodes, by means of real-time signal acquisition and subsequent data read back by the CPU, for off-line processing (generally the CPU forwards the data stored in memory to a host PC which performs signal elaboration and visualization. In this way the designer can easily debug the hardware-implemented DSP chain being able to trace up to 16 signals with the only limitation of external SRAM dimension.

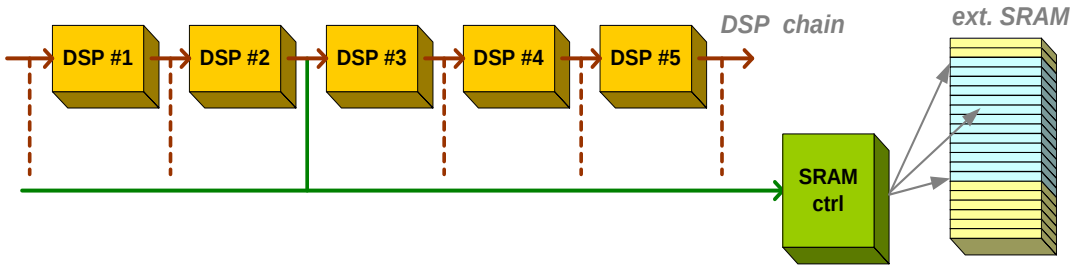

Figure 30: SRAM controller connections.

The controller is located on the AMBA APB bus, where several registers can be accessed by the CPU to issue commands and settings, such as max number of samples, timing of the write cycles, start address of the stored data and probe signal selection.
Through the APB interface the CPU can set these parameters and also store into memory test words to verify that write (and read) accesses to SRAM work properly, before starting to probe a signal.
Once in probing mode the selected probe words are forwarded to memory, with variable latency but fixed throughput (depending on the chosen write timing, max is 1 sample every 4 clock cycles). Probing stops automatically after reaching the max number of samples (or it's forced to stop with the appropriate command).
Memory data can be read back by the CPU issuing first a 'start reading' command, and then through read accesses to DATA register (each access invokes a new memory read). Oldest stored data are read first.
A STATUS register provides to CPU three flags: Probing Mode, Data Ready (to be read) and All Memory Read (set when all the stored samples have been read by the CPU).
Main features:

- Suitable to probe up to 16 digital signals (of 16 bits each).
- Programmable probe signal, write timing, start address and number of samples.
- Write test word (and read back) feature allows implementation of a SRAM access operation monitor.
- Fast memory access.
- Hardware management of probing operations (CPU is completely free during probing mode).

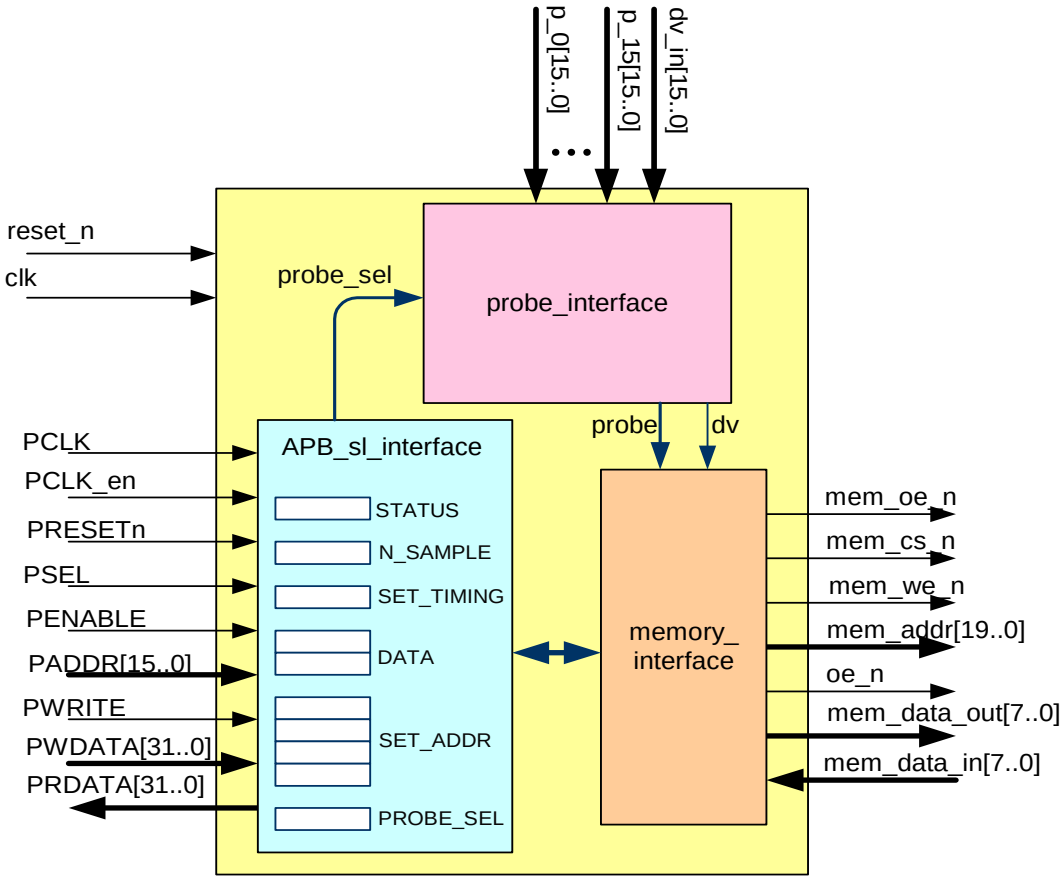

Figure 31: SRAM controller block diagram.

The SRAM controller is a set of three interfaces, each one described in a separate entity.
The *probe_interface* has the duty to work as a multiplexer (selecting one of the sixteen probe inputs and the related Data Valid) without being a multiplexer (it is indeed implemented as a kind of shift register). This implies a limited use of routing resources (that would be critical for a 256 to 16 mux) at the cost of a higher employment of registers and a latency (up to 16 clock cycles) applied to the probe signals.
The *APB_sl_interface* physically implements the APB registers (through which the controller is programmed and accessed by CPU) and forwards all the appropriate commands and information to the *memory_interface*.
The *memory_interface* implements a simple state machine designed to access memory on read and write, depending on the operating mode set by the CPU.
The memory controller is by default in an idle state, where all the APB registers can be accessed by the CPU, so that the controller can be properly configured. It is possible to choose one of four possible write timings, the number of samples that must be stored

in memory during the probing mode, which one of the 16 probes has to be stored and the start address in memory where they will be located (this also allows to probe in sequence several signals and then read them back all together, but this operation needs a careful software handling). The idle state can be used to store test words directly form CPU to memory, just writing the word in the DATA register. Several test words can be written before reading them back (but they mustn't be more than the selected max number of samples). Issuing the command 'start reading' the CPU can then (read) access DATA register to read back all the test words (starting from the first written), to verify the correctness of memory accesses before starting to probe.
The memory controller enters the probing mode by issuing the command 'start probing'. It's highly recommended to set all the parameters (write to N_SAMPLE, SET_TIMING, SET_ADDR and PROBE_SEL) before issuing this command, in order to avoid unexpected behaviors. During the Probing Mode the PM flag is set and all the accesses to the APB registers are masked (except the STATUS register, where it's possible to write the 'stop probing' command to abort the current probing and get back to idle state). Otherwise the probing continues until all the N_SAMPLEs are stored into memory; then the controller returns to idle state (resets the PM bit).
It's reasonable to have a read session after the signal probing: this is done issuing first the 'start reading' command and then read accessing DATA register. Each read automatically invokes a new memory access, so that a new data is ready on the following access to DATA. Memory read accesses are usually faster than APB register reads; anyway a Data Ready (DR) flag is provided, in case a handshake was needed (DR is reset after a DATA read and is set when a new word is loaded into the register). The All Memory Read (AMR) flag is set only when N_SAMPLE memory reads have been carried out (after having issued a command 'start reading') and is reset on a new command issued.
Note that no hardware control is provided on the read data kind (probes or test words) and on the effective presence in memory of valid data (for example, when a 'stop probing' command is issued, less then N_SAMPLE valid words are stored in memory: in this case AMR is not set when all the probed words have been read, but just when N_SAMPLE reads have occurred).
Figure 32 depicts a typical probing/reading session: starting from the address specified in the register SET_ADDR, data are stored into subsequent memory locations until 2^{N_SAMPLE} = 8 words have been written. When the 'start reading' command is issued, words are read starting from the same starting address, and the AMR flag

is set when the last location (the 8th) is read from memory (not when it is withdrawn by CPU through DATA register).

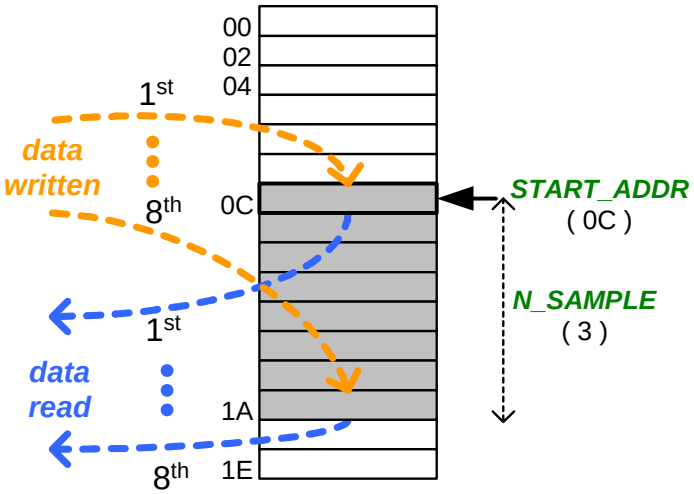

Figure 32: example of a probing session with subsequent read back.

iv. sd_freq_meter

The sd_freq_meter IP is a simple period calculator for square wave frequency monitoring. It can receive a programmable number of square wave inputs (within a programmable frequency range) and returns a period measurement according to the working mode: full period, half period high or half period low. This IP is accessed through the JTAG-like chain (JLCC).

Main features:

- JTAG-like access for programming and data output.
- Fully parametric VHDL description.
- Low power design (uses enable Flip-Flops that can be automatically converted to gated clock FF).

The sd_freq_meter IP is period calculator accessed via JLCC chain. One out of 4 possible working modes can be chosen by issuing a 2 bit command: NOP (No OPeration), FP (Full Period measurement), HP (High half Period measurement), LP (Low half Period measurement). As well the target input waveform must be chosen (total number of inputs is programmable via generic). Issuing any command implies a JLCC read back of the command previously written and output data (relative to the old command): the period measurement plus a valid bit (stating if the measurement has been successfully accomplished or not).

In order to perform a period measurement the following procedure should be followed:

- Send a command to select which operation to do on which input signal.
- Wait enough time to let *sd_freq_meter* complete the measurement. Up to as twice as the measurement effective length can be needed, because the first edge of input waveform is cut off (it could be a spurious transition due to a change of input channel).
- Issue another command in order to setup next measure or a NOP command.
- With the third JLCC access another measure can be started, and first meaningful data can be read back on the chain (this data refers to the first command issued). Each command read back is consistent with period data and valid bit.
- Measures can go on according to the described scheme. Software must take into consideration the data pipeline (typical of JLCC operations) and should not access the peripheral violating the timing in point 2). If a valid bit = '0' is read back, it can be because of a measure not yet terminated or a counter overflow (input frequency out of specification).

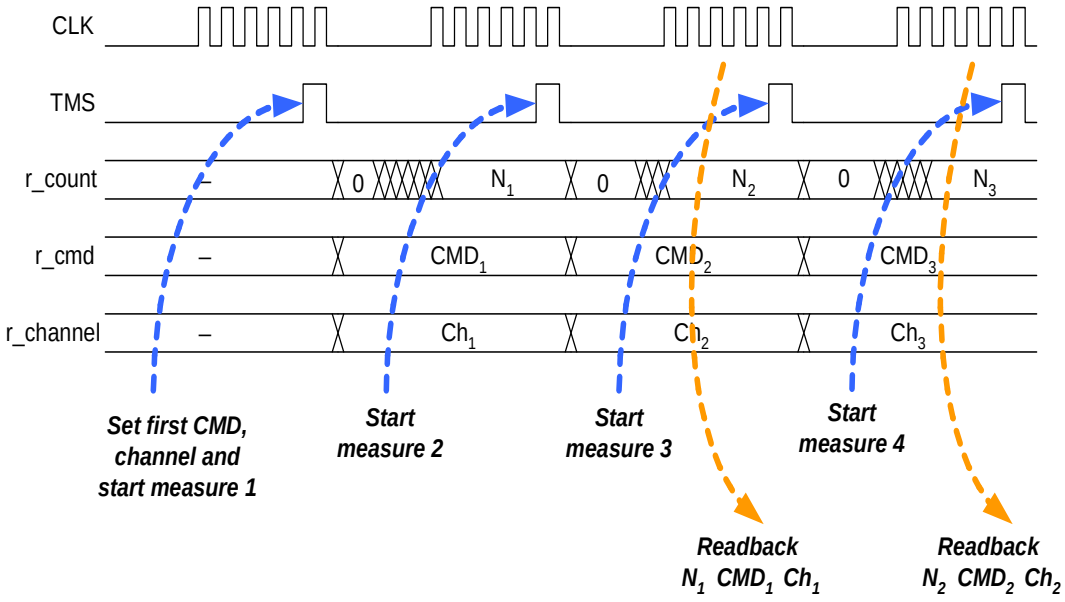

Figure 33: sd_freq_meter control via JLCC.

The basic block diagram of the *sd_freq_meter* is sketched in the figure below. Synchronization is performed on paths crossed by the dashed lines.

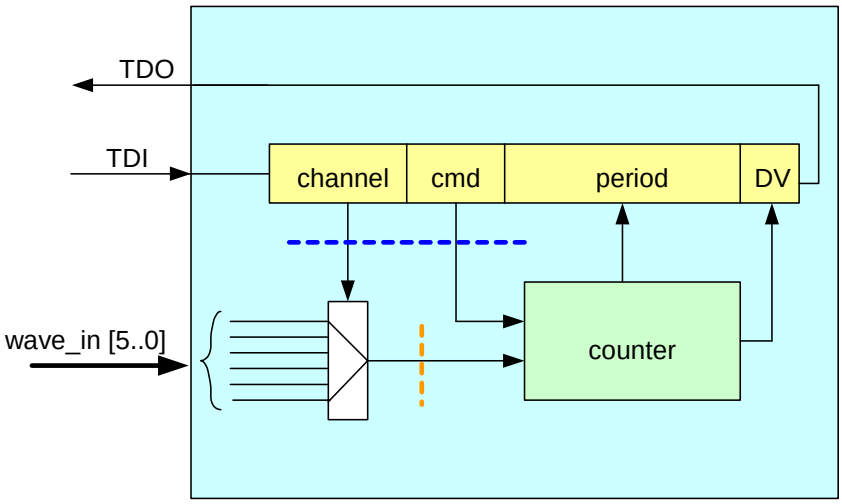

Figure 34: sd_freq_meter block diagram.

2.7 Low power optimization of IPs for SoCs

Automotive applications and in particular the sensor interfaces field is not exempt from tight constraints regarding power consumption. Even though most of automotive system are powered by the car's battery (with several Amperes available) and the main focuses of automotive electronics are elevated performance and high reliability, low power optimization still plays an important role due to several factors:

- The number of sensors inside a car (and in general of electronic devices) is rapidly growing and for this reason electric power budgeting is becoming a issue, especially regarding power peaks which may compromise the functionality of vital safety systems handled by electronic units.
- Sensor systems must guarantee their high performances over the full automotive range (-40°C to +125°C). Stability over temperature is often a critical issue and a high power consuming device is likely to suffer an excessive silicon heating and the subsequent biasing (in particular in the analog cells) which may spoil the overall system performances.
- Also the system reliability, which is a binding requirement for every automotive electronic device, may be badly affected by the silicon heating due to the high power consumption. Elevated silicon temperature means faster aging and precocious wearing which inevitably shorten the device life.

The power consumption of CMOS digital circuits can be expressed as follows:

$$P = (\alpha C_L V_{dd}^2 f_{clk} + I_{SC} V_{dd}) + I_{leakage} V_{dd}$$

where f_{clk} is the clock frequency, α is the average switching activity, V_{dd} is the supply voltage. The term within parenthesis in the above formula is the dynamic power. It is given by the switching power (due to charging and discharging of the capacitive load C_L of the circuit), and the short circuit component due to the current I_{SC} that arises when NMOS and PMOS transistors are simultaneously on. The last component is around 10% of the whole dynamic power [32]. The second term is the static power. It is mainly due to reverse-bias diode leakage on the transistor drains and to the sub-threshold current of off-devices. The dominant contribution to power is currently the dynamic component although, with latest technologies scaling below the 100 nm barrier, the contribution of leakage power can be no more negligible [32].

The minimization of power consumption, for a CMOS digital system, involves optimizations at different design levels: from the technology used, to the custom sizing of transistors and clock tree at gate level, up to architecture level techniques and, at a highest level, to the algorithm to be implemented. All these approaches require a trade-off between power, area, speed and flexibility.

In this paragraph are presented few examples of low power optimization for IPs which were not originally designed with low power features.

The optimization focuses on architecture level, being implemented directly on the RTL description of the IP cell and hence portable in different technologies and CAD environments. At the architectural level the dynamic power can be significantly reduced by avoiding superfluous switching activity, while static power can be kept low by choosing a low-leakage target technology for the CMOS implementation. The proposed optimizations are technology and CAD independent, thus they can be combined with the other low-power techniques proposed in literature.

i. 8051 CPU core by Oregano Systems

The considered 8051 core is freely distributed by Oregano Systems under LGPL licence [31]. It features a fully synchronous design, with an 8051 compatible instruction set, and optimized architecture to execute most of instructions in one clock cycle.

The Oregano core is up to 10 times faster than conventional 8051 architecture making it suitable for integration in a large class of embedded systems. A key feature, to let this IP highly reusable, is

the chance to customize it, for example by implementing or not the most demanding instructions (MUL, DIV) and by instantiating a user defined number of timers and UARTs serial interfaces, with relevant Special Function Registers. The VHDL source code has been synthesized within Synopsys™ environment, using a configuration with one timer, one UART and implementing MUL/DIV instructions, targeting a 0.18µm CMOS low-leakage technology. It resulted in a maximum clock speed of 48 MHz with a circuit complexity of about 10.5 Kgates, distributed among the four main blocks: ALU, control unit, UART and timer (Figure 35).

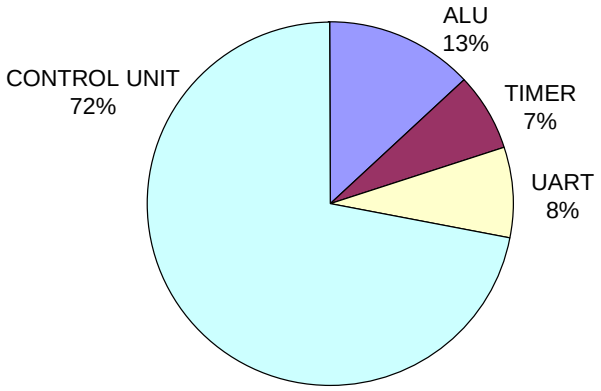

Figure 35: Area complexity of IP main blocks.

In order to identify the most power consuming block (on which concentrating the efforts for power optimization), and to evaluate power reduction on the modified architecture, the Dhrystone test code from Keil™ Software [33] has been chosen. This code is conceived to compare different CPU architectures (or compiler's efficiency), by providing a balanced instruction mix: 53% of assignments, 32% of control statements and 15% of function and procedure calls. This mix has been set up for simulating the behaviour a generic microcontroller application: for this reason it fits perfectly the role of reference code, as the results provided by the optimization are guaranteed to be not application specific, and an easy check of no performance loss due to the introduced architectural changes is immediately given at the end of the Dhrystone execution.

Power consumption of the four blocks has been estimated with Synopsys™ Power Compiler, using a 0.18µm CMOS library at 1.95 V supply voltage and 10 MHz clock. Percentage results are shown in Figure 36. This graph proves that control unit is definitely the most power hungry within the four blocks: for this reason it has been the object of the optimization. The 'other' entry in Figure 36

is mainly related to the switching activity of the clock distribution net.

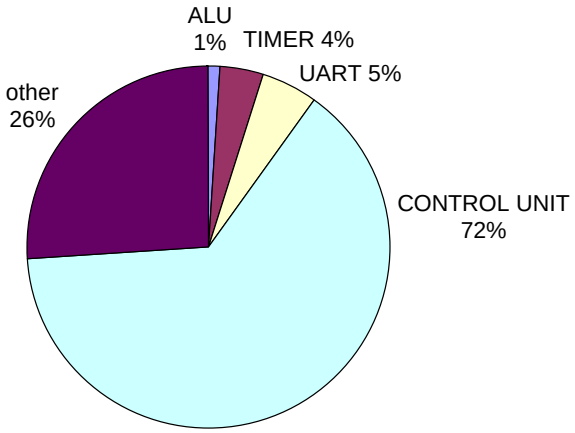

Figure 36: Power consumption of IP main blocks.

Three different techniques have been adopted to reduce power consumption at architectural level:
a) *State encoding*: this simple optimization [34] works on the state register, and aims to reduce its switching activity by minimizing Hamming distance between subsequent states (i.e. the number of bits which toggle during the transition). With an only 3-bit state register, this optimization did not lead to remarkable power improvements for the 8051 core.
b) *Clock gating*: this technique is one the most effective and thus widely employed in low power designs [32][35][36]. It is based on the principle that substantial power savings can be obtained if registers are not clocked when they don't have to produce valid outputs.

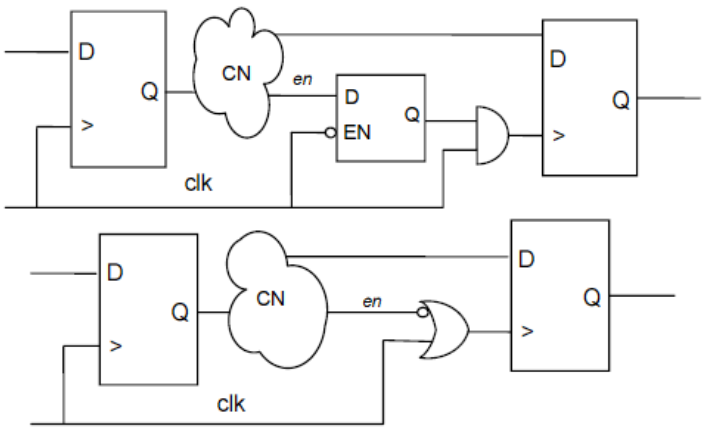

Figure 37: Different styles for clock gating.

Two kinds of solutions [36] are generally adopted to prevent clock from reaching a register: as shown in Figure 37, one employs a transparent latch (with negative enable), while the other simply ORs clock line with the enable logic. The latch-less technique introduces a smaller area overhead, but can lead to misbehaviour due to glitch propagation through the control logic, which is directly connected to the clock pin of the register (signal *en* must be held constant for all the clock negative half period).
To prevent timing errors, the latch technique has been chosen, as it allows the employment of this IP core in safe critical applications too (e.g. automotive, space [38]). The optimization methodology bases on scanning the VHDL code to find descriptions of enable registers, which are replaced by clock-gating latch-based structures.

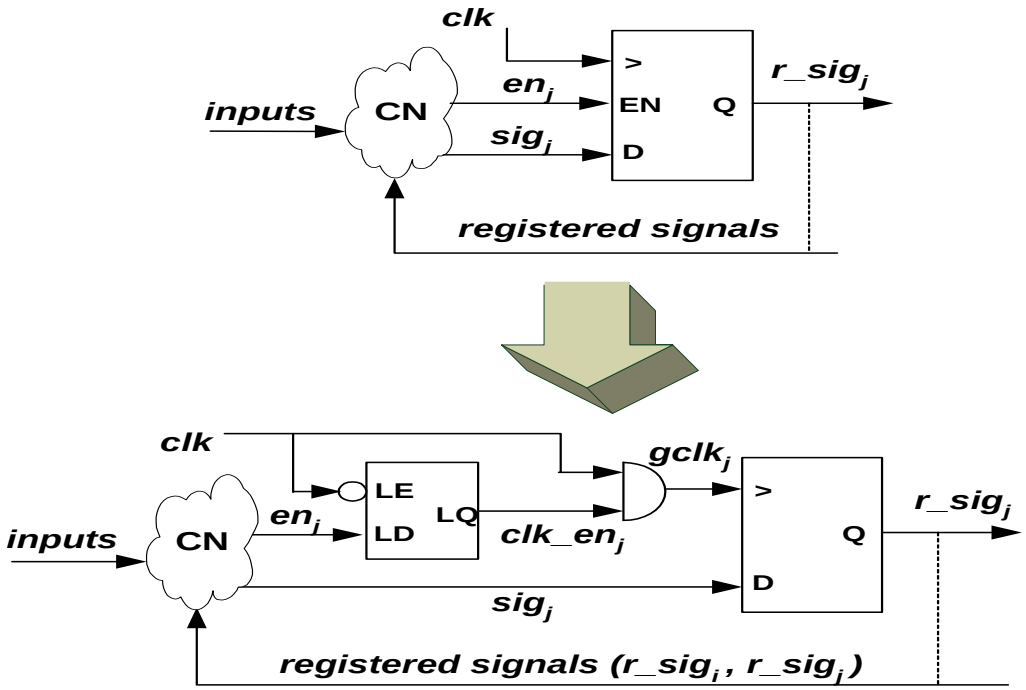

Figure 38: Clock gating insertion.

If we use the naming convention presented in Figure 38, a generic synchronization process can be described like this:

```
sync: process(clk)
...
if  Rising_Edge(clk) then
   r_sig_i <= sig_i ; -- system clocking
   case (ctrl_sig_i) is
     when (cond_k) => r_sig_k <= sig_k ; -- gating
     end case;
   if (en_j = '1') then r_sig_j <= sig_j ;-- gating
```

```
    end if;
end if;
end process;
```

We leave in process *sync* (sensitive to system clock) only those signals which are always enabled (*r_sigi*), while we create two new processes and a combinatorial statement for each register *r_sigj* that has to be gated. Combinatorial logic that generates inputs and enables for the registers doesn't need to be changed.

```
gclkj <= clk and clk_enj; -- clock gating
latch: process(clk, enj) -- latch modeling
begin
  if (clk = '0') then clk_enj <= enj;
  end if;
end process;
gsync: process(gclkj) -- new sync
...
  if  Rising_Edge(gclkj) then
       r_sigj <= sigj ;
  end if;
end process;
```

This new description allows synthesizer to instantiate the proper gating logic, but at the same time may be wrongly interpreted by the RTL simulator. Indeed, in case some of the registered signals sketched in Figure 38 are edge-triggered by the system clock (e.g. signals *r_sigi* in the above description) the following problem arises. A clock event causes the execution of the *sync* process, updating signals *r_sigi* which are inputs to enabled (now gated) registers; the relevant processes are executed after some delta cycles, as they now originate from a *gclkj* event, instead of being concurrent. This implies that some of the *r_sigj* signals are updated with the new values, coming from *r_sigi* registers, that they should have got on the following clock event according to the original description. The solution is to assign these critical signals outside the *sync* process to some aliases called *sigi_aux*, so that they are updated at the same time as *gclkj*, and the *gsync* process works correctly. Example:

```
sync: process(clk)
...
  if  Rising_Edge(clk) then
       r_sigi <= sigi ;
  end if;
end process;
```

```
sig_i_aux <= CN(r_sig_i,…); -- updated together
gclk_j <= clk and clk_en_j; -- updated together
gsync: process(gclk_j) -- new sync
...
  if  Rising_Edge(gclk_j) then
       r_sig_j <= sig_i_aux;
  end if;
end process;
```

Clustering of gated registers: An important issue concerning the clock gating of a set of registers is whether to gate clock once for all the registers, or to generate gating signals dedicated to each one. The more clock-gating are inserted, the more switching activity is reduced, since each register has different clock enable conditions. However, each gating structure introduces also an area and power overhead. Therefore, it's required to find an optimal clustering. That is to say, dividing the total amount of registers in M clusters of K registers, where each cluster is clocked by its own *gclk* signal, we have to find K optimal.

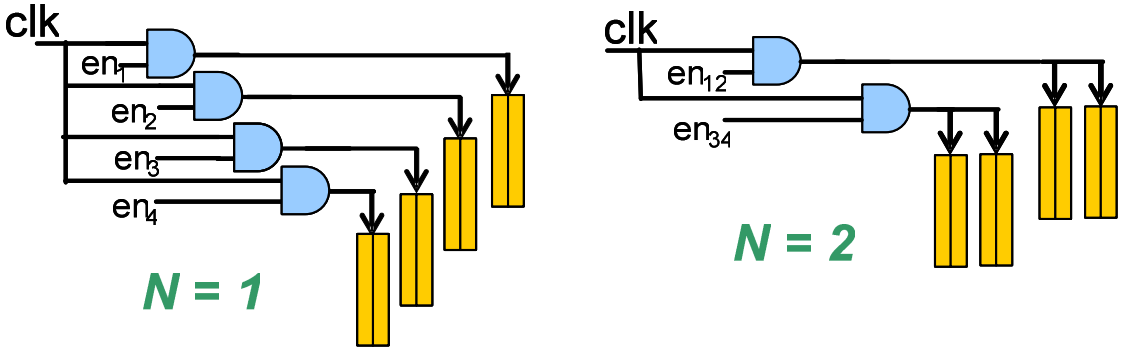

Figure 39: example of clock gating clustering.

For a generic n-bit register, the power saving due to clock gating insertion can be expressed as follows [36]:

$$P_{saved} = f_{clk} V_{dd}^{2} [C_{gated}(1-\alpha) - C_{add}]$$

where C_{gated} is the total input capacitance of the gated n-bit register and, with reference to Fig. 4, C_{add} is the capacitance of the AND and LATCH inputs, a is the switching activity of the *gclk* signal. In case of M subsets of K registers the total power saving is (all the K registers belonging to the same i_{th} cluster share the same *gclki* signal and hence the same a_i) :

$$P_{saved} = f_{clk} V_{dd}^{2} \sum_{i=1}^{M} [KC_{gated}(1-\alpha_i) - C_{add}]$$

Logic synthesis results prove that C_{add} is roughly equal to the capacitance of a 1-bit register clock pin. In particular, for the considered IP core most of the registers are at 8-bit and hence $C_{gated} \approx 8C_{add}$

Therefore, the total power saving is:

$$P_{saved} = f_{clk} {V_{dd}}^2 C_{add} \sum_{i=1}^{M} [8K(1-\alpha_i)-1] = f_{clk} {V_{dd}}^2 C_{add} \left[8K\left(M - \sum_{i=1}^{M} \alpha_i \right) - M \right]$$

where C_{add} is the capacitance of the AND and LATCH inputs, a_i is the switching activity of the *gclki* signal. Given that the number N = 8KM is known a-priori from the CPU architecture and under the assumption, verified by simulations, that the average IP switching activity does not depend on the organization of gating clusters for a given set of input stimuli, the summation

$$\sum_{i=1}^{M} \alpha_i = c$$

is independent from the particular value of K.

The saved power can be maximized using clusters of

$$K_{opt} = \frac{1}{8} \sqrt{\frac{N}{c}}$$

registers. Through VHDL simulations with a special counting process, we have evaluated that c = 0.48 for the Dhrystone test code, with N = 328, leading to the result of K_{opt} = 3.26, that is to say the optimal solution is using clusters of 3 registers (selecting such value for K, the solution will be optimal for applications with average switching activity in the range from 0.42 to 0.82).

While the clock gating insertion can also be performed automatically by the synthesis tool (for example Synopsys™), the clustering optimization cannot be performed automatically by the tools (c is not known a priori), allowing this method to achieve better results than the automatic clock gating insertion.

c) *Operand isolation*: this technique consists in 'freezing' the inputs to big combinatorial nets when their outputs are not needed, for example by ANDing them with a valid signal [38], and it can prove very useful in DSP applications where large adders, shifters and multipliers are employed. For the target 8051 IP core, operand isolation has been implemented on the program counter register, where adders that calculate pc (program counter), pc+2 and pc+3 have a high switching activity at their inputs and not always provide significant data at their outputs, while has not been applied to ALU's adder, comparator and multiplier, as the whole

energy consumed by this block is negligible if compared to that of the control unit.

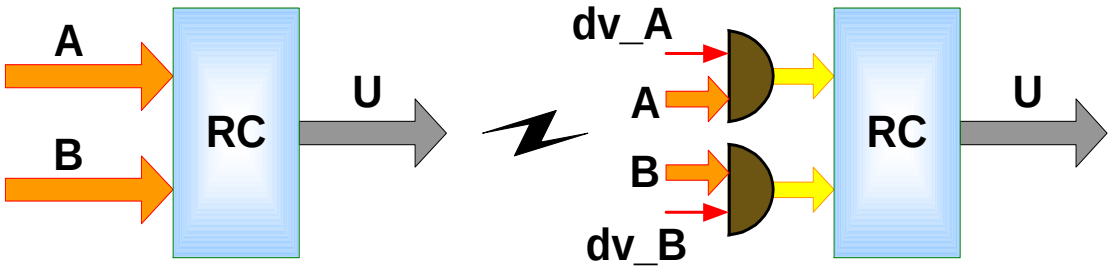

Figure 40: operand isolation.

Results: The 8051 IP was first modified inserting state encoding: this optimization did not affect significantly area and maximum frequency, and resulted in a small power reduction (1%). Better results came from clock gating, which brought to a remarkable decrease in power consumption (≈ 37%), while limiting area overhead at 3%. Compared to automatic clock gating insertion, a further 7% of dynamic power was saved. As a counterpart, this technique lead to a slow down of maximum operating frequency, as the enable signals entering the latches must settle before the falling edge of clock. In the end, implementation of operand isolation introduced with respect to the original IP a 3% power reduction, with few effects on area and maximum frequency. The final low-power version of the Oregano 8051 core, with all the three techniques implemented, features a total power reduction of about 40% with respect to the original IP, a small area overhead of roughly 3% and a maximum clock frequency of 44 MHz. Yet the modified 8051 core keeps the same timing and functionalities (verified through the Dhrystone execution, it results in the same performances), being still a technology independent, highly reusable IP.

ii. Flow for low power synthesis

The manual modification of the IP's VHDL code presented in previous paragraph achieves the best power saving results. Yet it implies two main drawbacks:

- The VHDL code modification has to be performed by a designer (no scripting or automatic procedure can efficiently accomplish this task) and often results to be rather time consuming (and consequently expensive).

- Such activity may introduce bugs into the IP, thus requires a thorough verification before integrating the modified IP into a project.

A costs/benefits analysis must be carried out in order to evaluate whether this technique can be profitable for a wide set of outsourced IPs, also considering that low power optimization is undoubtedly helpful in automotive applications but no killing requirement is set for power consumption constraints.
An alternative approach consists in waiving part of the low power optimization and letting the synthesis tools handle this issue. By setting up a low power synthesis flow (basically made of a series of scripts and procedures) the tool automatically inserts clock gating and sleep logic, achieving a substantial reduction of design costs (those related to the engineering needed for the IP modifications) and guaranteeing a bug free implementation.
This paragraph addresses the implementation of low power synthesis flows for the main commercial synthesis tools (Cadence™ and Synopsys™) and the results achieved for comparison with the manual optimization of previous paragraph.

a) *Synopsys*™: as exemplified in Figure 41 [39], the HDL Compiler perform a first synthesis and generates a forward-annotation file (SAIF format) which contains the information on the nodes whose switching activity has to be monitored and that will be used in a further RTL simulation for the back-annotation SAIF file creation. Then clock gating and operand isolation structures are inserted and the modified design is synthesized by Design Compiler. At the end of the flow, Power Compiler uses the switching activity information from the back-annotation SAIF file for deciding which power-saving structures are effective and which simply constitute an overhead, thus outputting the power optimized netlist. A gate-level (besides the RTL) simulation gives Power Compiler further information on parasites, thus improving the optimization process.
The depicted flow was tested on the 8051 core by Oregano Systems, using as a reference the same Dhrystone as in paragraph 2.7. The automating insertion of clock gating achieved a power saving of 30% with respect to standard synthesis flows, thus proving to be a valuable optimization method even if manual IP modifications showed a better overall power reduction.

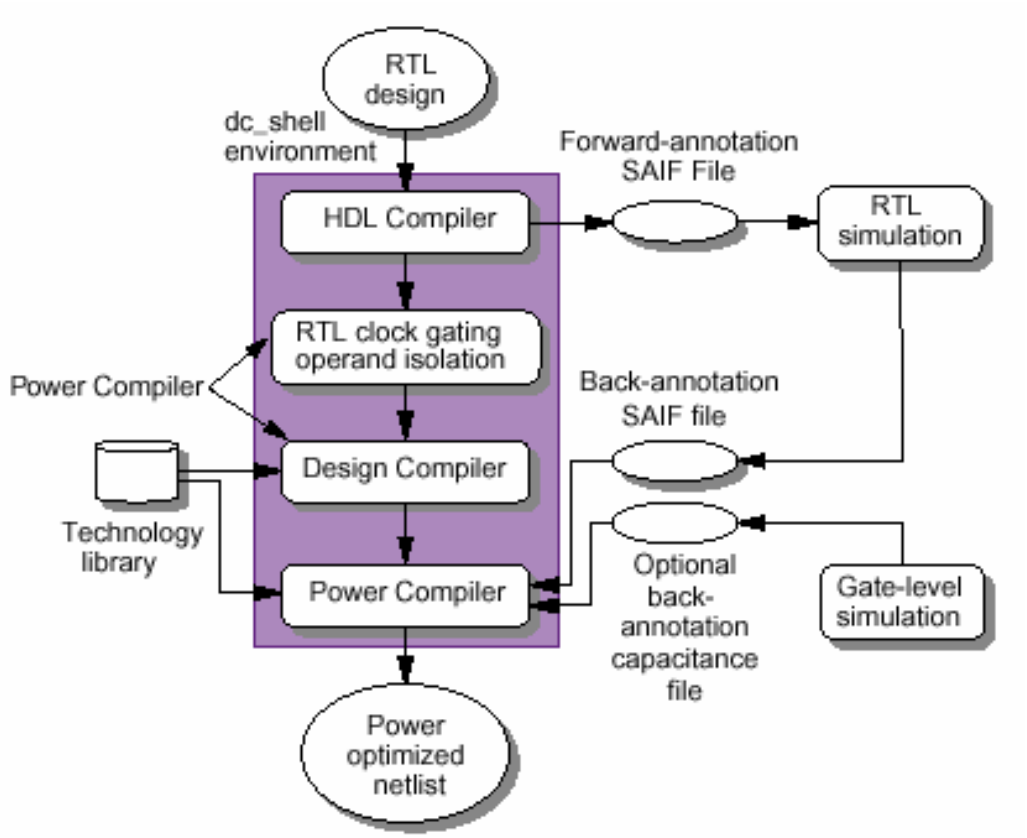

Figure 41: Low power synthesis flow with Synopsys™ [39].

b) *Cadence*™: this tool [40] handles the synthesis for low power in a similar way of Synopsys™. It basically consists of three steps:

- The initial synthesis (with low power switch selected) searches for all the possible candidates of clock gating and sleep mode, inserting the low power structure in every case.
- A gate level simulation on the generated netlist collects the switching activity data needed by the last synthesis step.
- Low power optimization, according to the switching activity information, decides whether to commit or remove the clock gating and sleep mode structures inserted at the first step.

This tool allows the choice of which low power technique to be used (only clock gating, only sleep mode or both) and thus the impact of each possibility has been evaluated. In order to extend the validity of the results, besides the Oregano core another 8051 CPU has been employed for this study: the CPU 8051 developed by Research Institute for Integrated Circuits (RIIC) [41]. The RTL VHDL code of such IP is freely distributed under LGPL licence and the IP is compliant to the MCS-51 specification (from Intel Corp.)

for 8051 CPUs. The RIIC features a fully synchronous design with code, external and internal data memories, SFRs, interrupt controller, UART and timer, and optional second data pointer (DPTR) and selectable hardware implementation of MUL, DIV and DA instructions.
The performed analyses have shown global power saving of about 25% for both microcontrollers, which has been achieved mostly by the automatic insertion of clock gating, while the sleep mode (both when evaluated alone or added to the clock gating) did not bring substantial improvements for this kind of IPs. This approach has been further verified by extending the analysis to the back-end flow: in this way the power saving results have been confirmed also after place & route and interconnection parasites extraction.

Bibliography

[18] Xiujun Li; Meijer, G.C.M.; de Boer, R.; van der Lee, M.; A high-performance Universal Sensor Interface Sensor for Industry, 2001, Proceedings of the First ISA/IEEE Conference, Pages:19 – 22, 5-7 Nov. 2001

[19] Meijer, G.C.M.; Xiujun LI; "Smart Sensor Interface Electronics", Microelectronics, 2002. MIEL 2002. 23rd International Conference on, Volume 1, 12-15 May 2002 Page(s):67 - 74 vol.1

[20] Guan Chao; Li Xiujun; Meijer, G.C.M.; "A System-Level Approach for the Design of Smart Sensor Interfaces", Sensors, 2004. Proceedings of IEEE 24-27 Oct. 2004 Page(s):210 - 214 vol.1

[21] Jichun Zhang; Junwei Zhou; Balasundaram, P.; Mason, A.; "A highly programmable sensor network interface with multiple sensor readout circuits" Sensors, 2003. Proceedings of IEEE , Volume: 2, Pages:748 – 752, 22-24 Oct. 2003

[22] Sangiovanni-Vincentelli, A.; Martin, G.; "Platform-based design and software design methodology for embedded systems" Design & Test of Computers, IEEE Volume 18, Issue 6, Nov.-Dec. 2001 Page(s):23 - 33

[23] Carloni, L.; De Bernardinis, F.; Sangiovanni Vincencentelli, A.; Sgroi, M.; "The Art and Science of Integrated Systems Design", Solid-State Device Research Conference, 2002. Proceeding of the 32nd European, 24-26 September 2002 Page(s):19 - 30

[24] Fanucci, L.; Giambastiani, A.; Iozzi, F.; Marino, C.; Rocchi, A.; "Platform based design for automotive sensor conditioning" Design, Automation and Test in Europe, 2005. Proceedings 2005 Page(s):186 - 191 Vol. 3.

[25] Marino, C.; Forliti, M.; Rocchi, A.; Giambastiani, A.; Iozzi, F.; De Marinis, M.; Fanucci, L.; "Mixed signal behavioral verification using VHDL-AMS", Research in Microelectronics and Electronics, 2005 PhD Volume 2, 25-28 July 2005 Page(s):115 - 118

[26] C. Marino, M. Forliti, A. Rocchi, F. Iozzi, A. Giambastiani. "VHDL-AMS Modelling and System Verification Flow for Mixed-Signal System-On-Chip", Forum on specification & Design Languages Sept. 27-30 Lausanne 2005

[27] Farrokh Ayazi, and Khalil Najafi; "Design and Fabrication of A High-Performance Polysilicon Vibrating Ring Gyroscope", Eleventh IEEE/ASME International Workshop on Micro Electro Mechanical Systems Heidelberg, Germany, January 25-29 (1998)

[28] W. Geiger, W. U. Butt, A. Gaißer, J. Frech, M. Braxmaier, T. Link, A. Kohne, P. Nommensen, H. Sandmaier, W. Lang and H. Sandmaier; "The silicon angular rate sensor system DAVED©" Sensors and Actuators A: Physical, Volume 95, Issues 2-3, Pages 239-249; (2002)

[29] ADXRS300 Datasheet www.analog.com

[30] Gyrostar® Datasheet www.murata.com

[31] Oregano Systems, Design & Consulting GesmbH, www.oregano.at

[32] Benini, L.; De Micheli, G.; Macii, E.; "Designing low-power circuits: practical recipes", IEEE Circuits and Systems Magazine, 1 (1) 2001, pp. 6-25

[33] Keil™ Software Embedded Development Tools, www.keil.com

[34] De-Sheng Chen; Sarrafzadeh, M.; Yeap, G.K.H.; "State encoding of finite state machines for low power design", Proc. IEEE ISCAS, pp. 2309 – 2312, May 1995

[35] Raghavan, N.; Akella, V.; Bakshi, S.; "Automatic insertion of gated clocks at register transfer level", Proc. IEEE VLSI Design, pp. 48–54, 1999

[36] F. Emnett, M. Biegel, “Power Reduction Through RTL Clock Gating”, Synopsys User Group, San Jose, March 2000
[37] Alidina, M.; Monteiro, J.; Devadas, S.; Ghosh, A.; Papaefthymiou, M.; “Precomputation-based sequential logic optimization for low power”, IEEE Trans. on VLSI Systems, 2 (4), 1994 pp. 426 – 436
[38] J.C. Lyke, “Design of a power-optimized micro miniature advanced instrument controller for sensor craft applications”, AIAA/IEEE Digital Avionics Syst. Conf., pp. 145–150, 1996
[39] Synopsys™ Design Compiler User Manual
[40] Cadence™ Buildgates User Manual
[41] Research Institute for Integrated Circuits, www.riic.at

Chapter 3
FAST PROTOTYPING FLOW

3.1 Limitations of Platform Based Design flow

The Platform Based methodology presented in Chapter 2 constitutes a paramount support to the designers for handling the raising chip complexity and the tightening of design constraints. Nevertheless, regarding the development of sensor interfaces in automotive applications, this approach features few drawbacks and limitations that should be overcome in order to keep the pace of demands coming from the market and at the same time fulfil all the typical requirements of such applications.
The most critical aspect of a Platform Based flow (which also recurs in every top-down design approach) resides in the architectural choices which have to be performed at the highest level of abstraction. These decisions concern not only the system overall architecture (which blocks are to be used) but also rough specifications for the selected blocks and partitioning choices at least between analog and digital implementation domain. Of course, being at the highest design level these decisions will affect at the utmost degree final system performances, and thus they need to be taken with maximum care. Yet at such design stage the designer can easily lack comprehensive information that would be necessary to make the correct choices: for example, the architecture of a sensor interface is evaluated through MATLAB™ simulations using a model of the target sensor, but little inaccuracies on that model may easily lead to wrong processing blocks employment or insufficient specification, thus spoiling system performances and outing only by the end of the design flow, after the prototyping phase. This means that little and unavoidable design mistakes have the possibility to cause further

design iterations with the subsequent increase in development costs and time-to-market delay.
In a similar way, the lack of critical information at the top design level can be the reason (if not of a defective chip) of missing some of the original chip specification. This may happen because the simulations at high abstraction level often provide low accuracy when used for estimating performances of complex systems. What perfectly matches the project requirements after system level simulations, does not give enough guarantees to be so performing after its implementation.
At the extreme consequences, the top-down approach results to be critical for all those platform customizations which would require a dedicated feasibility study. Obviously the platform for inertial sensors conditioning presented in Chapter 2 addresses a wide range of capacitive sensors (like gyros and accelerometers), as it was conceived for this goal, but it's quite hard to foresee to what extent it can be adapted to (for example) inductive hall sensors or magneto-resistive position sensors, unless an implementation for such application is designed and prototyped.

3.2 ISIF platform

Intelligent Sensor InterFace (ISIF) is a platform implementation [42] which has been conceived to overcome the issues with top-down design methodologies described in previous paragraph. ISIF provides a set of programmable analog and digital IPs directly on silicon, with the possibility to change their interconnections and integrate them with DSP software routines emulating hardware blocks. In this way a number of conditioning architectures can be implemented and evaluated by simply connecting ISIF to the target sensor and thus easing and speeding up the design space exploration phase. A design flow with the employment of ISIF does not necessarily preclude some MATLAB™ modelling and system level simulations: yet it makes these analyses a further investigation possibility instead of the only source of information for taking critical architectural decisions.
In practice the main enhancement brought by ISIF resides in the drastic reduction of development time of the interface. In place of long time consuming simulations and potential inaccuracies in the used model that may lead to several design iteration, ISIF allows the system prototyping before its actual design, with accurate feedback information coming from IPs already on silicon, thus it eliminates the big parasitic overheads of discrete components

setups and lets the designer free to explore all the possible processing solutions just by selecting the desired blocks, their characteristics and the related interconnections.

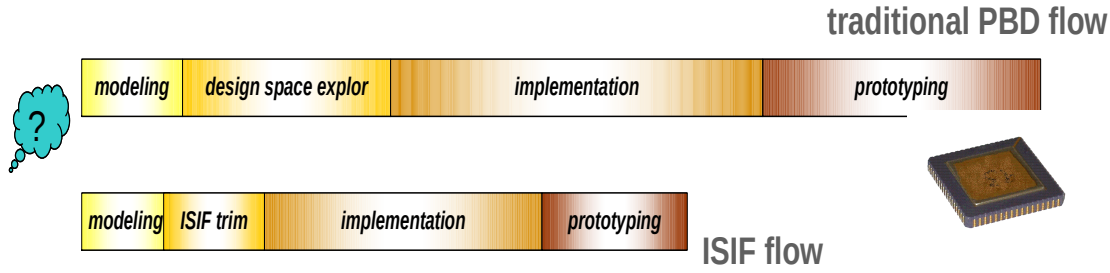

Figure 42: comparison between design time for a traditional Platform Based and the ISIF flow.

The key to achieve the goal of ISIF concept is the maximum flexibility of each hardware-implemented IP (both in analog and digital domains) and the extremely high routing possibilities, also including the chance of moving from hardware to software at any stage of the signal processing chain. Another paramount ISIF feature is the *emulation* nature of the chip: although it presents commonalities with the Universal Sensor Interfaces (as it aims to conditioning a wide class of sensors) it is not intended to be a product for any particular application, instead it constitutes a hardware/software implementation of a generic platform for sensor conditioning from which customizations and optimizations can be derived for target applications, in order to achieve the highest performances and the lowest overheads. For this reason most of the IPs that are generally available for running a Platform Based design flow have been integrated into ISIF keeping as wide as possible the configurability options, while the remaining IPs (mostly digital signal processing blocks) which haven't been integrated can be anyway inserted via their software emulations (routines which perform the same functionalities of hardware IPs and emulate their timing and data-paths limitations).
In this way a sensor conditioning architecture can be fast prototyped (as shown in the example in Figure 43) by inserting the signal coming from the sensor in any of the input channel stages (it is possible to skip those not needed in the rest of the chain), then the preferred ADC converter can be chosen and then the signal is routed to one of the digital processing blocks, each one can be accessed at his input or output by software for emulated IPs insertion. Through the wide possibilities to route out the signal via several DAC types, signal can be feedback to the sensor implementing closed loop architectures.

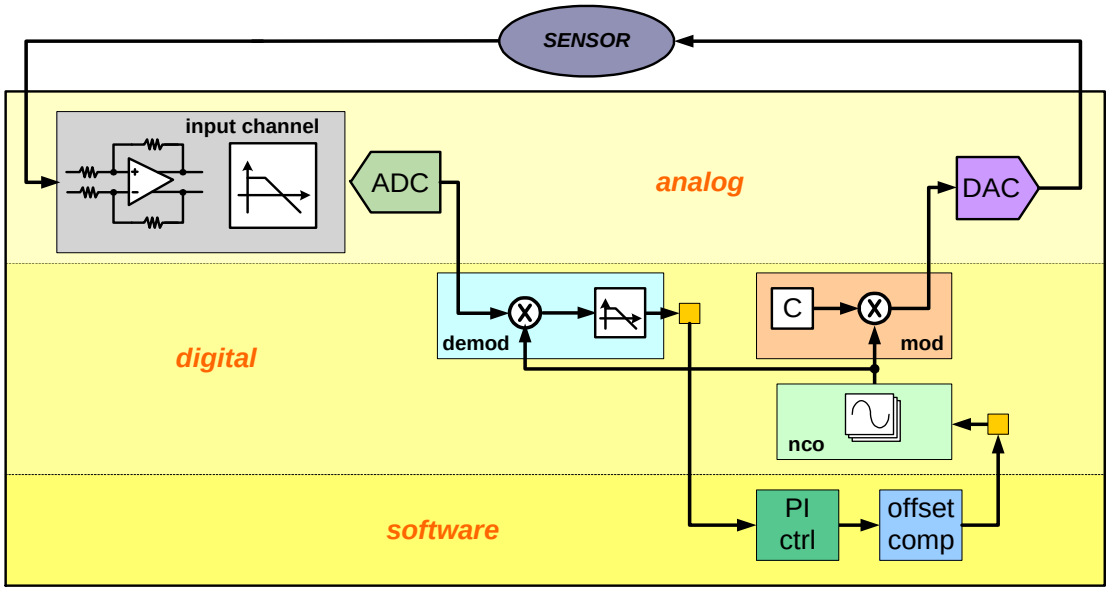

Figure 43: example of partitioning of a DSP chain within ISIF platform between analog, digital and software processing blocks.

For what concerns ISIF hardware, the analog section features a wide range of IPs for sensor signal acquisition, driving and basic analog conditioning such as DACs, ADCs, amplifiers, filters, and current/voltage sources, while the digital section is composed by the LEON CPU core, together with standard peripherals for communication with external devices, memories and buses (AMBA APB/AHB) and few blocks for digital signal processing (modulator, demodulator and sine wave generator. ISIF platform has been implemented by Sensordynamics AG in 0.35µm BCD6 technology, on a single chip with area occupancy of about 72mm^2.

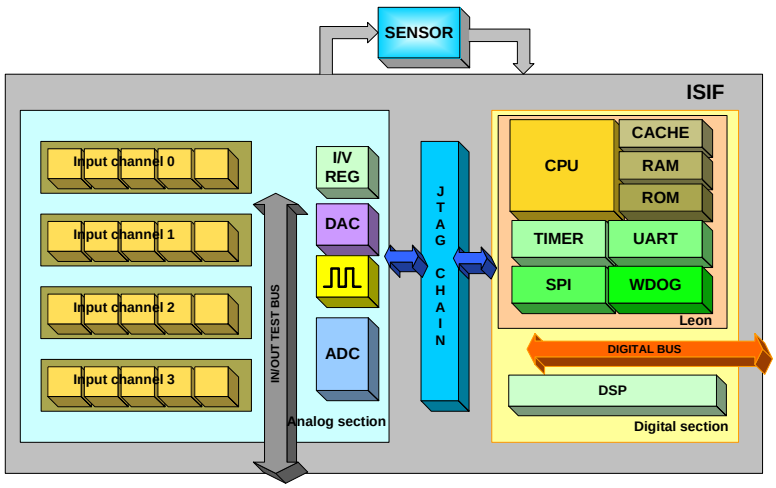

Figure 44: ISIF block diagram.

i. Analog section

ISIF analog section is based on 4 input channels for signal acquisition, and each channel is composed of different stages as we can see in Figure 45. First an input charge amplifier is able to detect voltage, current or capacitance (thus covering the most of sensor typologies). This is possible thanks to the number of different feedback capacitances and resistances which can be selected and the input matrix switches that can make the input stage operate as a charge amplifier, I/V converter or instrument amplifier.
After signal acquisition, differential amplifiers, low pass filters and level shifter provide proper analog conditioning with a high degree of configurability. Then the signal properly filtered and adjusted in gain and dynamic is converted by the Sigma Delta (or SAR) ADCs. Additional analog blocks provide voltage/current references, oscillators for clock generation and DACs converters.

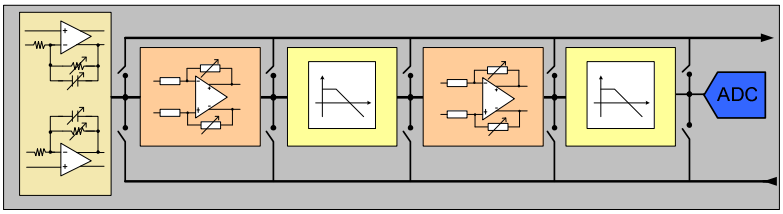

Figure 45: ISIF input channel.

An additional feature is the input/output test bus, provided to supply stimuli and probe output signals for each block. It represents a key issue for an effective and quick debug of the signal conditioning path, besides allowing the injection of input signal at a different stage of the channel than the charge amplifier.
A peculiarity of ISIF analog section is the design accurateness for improving the noise margin (for example analog and digital supplies are kept separated in order to minimize noise couplings) and that the digital bits for analog blocks configurations are handled by a JTAG-like approach: programming is performed by shifting the configuration bits through shift registers, in order to overcome clock skew issues and thus guaranteeing a safe communication between digital and analog structures.

ii. Digital section

The digital hardware section is composed of a CPU core with related peripherals and dedicated IPs for digital signal processing (see Figure 46).

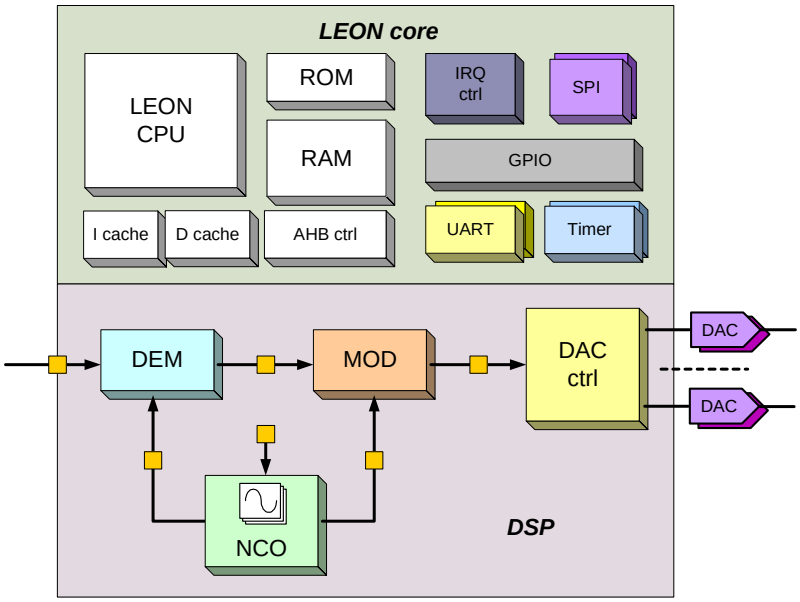

Figure 46: ISIF digital section.

LEON core is freely distributed under LGPL licence and includes a 32-bit RISC general purpose processor compliant to SPARC-V8 standard which features hardware multiplier and divider plus memories (ROM, RAM and data/instruction caches) and a set of peripherals like interrupt controller, UARTs, SPIs, timers and watchdog. The hardware digital signal processing block is made up of dedicated IPs optimized for low power consumption such as modulator, four-channel demodulator, controller for driving up to 6 DACs, low pass filters (FIR and IIR) and sine wave generator which can provide up to 16 waves with 3 different frequencies and programmable phases. The interconnection among these IPs can be implemented by hardware or they can be directly accessed at their input/output by software. The high flexibility of DSP section and CPU potentiality allows designers to implement complex and ad hoc algorithms for the target sensor conditioning, for example a digital PLL has been fully implemented and tested on a fast prototyping board [42].

LEON processor performs system monitoring, controls signal processing chain and handles the communication resources. Part of the software is included in ROM (boot and few utility functions), while the rest can to be downloaded at startup via UART, or can be stored in external SPI EEPROM and so directly reboot from EEPROM (which can hold different software and data to speed up time in trimming and test procedures). Firmware utilities can change interconnections among digital IPs, manage the communication with external devices (for debug, monitoring) and configure the whole analog front-end section (adjusting

parameters such gain, bandwidth e.g.) to match different sensors requirements.

iii. DSP software

The requirements of automotive applications are pushing towards the use of hardware solutions (especially concerning safety issues), on the other hand presence the elevated number of variables (for example regarding block and data-paths dimensioning) make hardware implementation really difficult to be successful at first time. Furthermore several digital IPs require detailed analysis for proper parameters setting, as automotive applications often requires both high performances and reduced area, thing not compatible with the usage of a large number of configuration bits for trimming or over-dimensioned paths. To allow comprehensive investigations on the purpose of meeting such requirements, ISIF platform includes software peripherals (filters, controllers e.g.) with an exact matching with hardware devices. The LEON processor offers good signal processing features (hardware multipliers and accumulation) and guarantees the flexibility and the computational power for real-time software IPs implementation. It is worth noting that the aim of this platform is not tended towards achieving the best performances (thus fully exploiting LEON computational potential), but in the maximum accurate emulation of a complete hardware optimized sensor interface (which could hardly afford such an area and power consuming processor). So the most common functionalities, which are not fulfilled by ISIF digital section, are modeled by software routines keeping the same behavior of the original DSP library IPs (concerning bits width, saturation, linearity e.g.). A DSP software environment allows input/output data from hardware digital IPs to be acquired by routines, afterwards elaborated and passed back to physical blocks: just as the data processing was completely hardware-implemented. The monitoring, control and communication functionalities are also handled by software, when the CPU is free from data processing tasks (which have the highest priority).
This platform flexibility helps the designer to find out the most proper DSP solutions for a target sensor. With a Personal Computer connected via UART to the ISIF board the designer can quickly explore a wide project design space, changing analog settings, interconnecting digital IPs and even instantiating new ones in order to optimize architecture both in terms of area and performances.

In the ASIC derived from ISIF prototyping, software routines will be replaced by corresponding hardware IPs with minimum risk of redesign necessity and subsequent time-to-market improvement.

3.3 Case studies

In this paragraph several examples of sensor interface prototyping with ISIF are presented.

i. Magneto-resistive position sensor

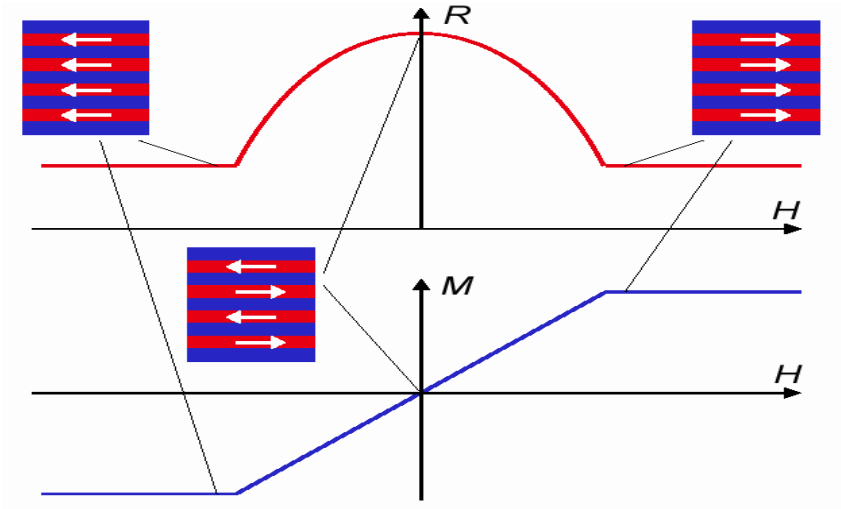

Figure 47: Variation of resistance versus ferromagnetic field (R) and magnetization of sensor versus applied magnetic field (M).

Magneto-resistive sensors have a wide range of applications such as proximity detection, displacement sensing, rotational reference detection and current sensing. Such sensors are based on the Giant Magneto-Resistance (GMR) effect [43], a quantum mechanical effect observed in thin film structures composed of alternating ferromagnetic and nonmagnetic metal layers.
The effect manifests itself as a significant decrease in resistance from the zero-field state, when the magnetization of adjacent ferromagnetic layers are antiparallel due to a weak anti-ferromagnetic coupling between layers, to a lower level of resistance when the magnetization of the adjacent layers align due to an applied external field. The spin of the electrons of the nonmagnetic metal align parallel or antiparallel with an applied magnetic field in equal numbers, and therefore suffer less

magnetic scattering when the magnetizations of the ferromagnetic layers are parallel [44].
The electronic interface must detect resistance variation, also providing some digital signal processing like noise filtering and proper output signal bandwidth.
For the GMR sensor application two ISIF input channels have been configured as instrument amplifier for voltage detection; data coming from input channel have been converted with the sigma-delta and the low-pass filtered first trough hardware LPFs inside demodulator and then further processed by a software routine performing a time average of samples.
The conditioning of this sensor was set up in about 2 days, exploring a number of possible analog chain configurations in order to achieve the minimum noise level for the desired sensitivity of 0.1°. Output signals bandwidth can reach up to 10 KHz.

ii. Biosensor

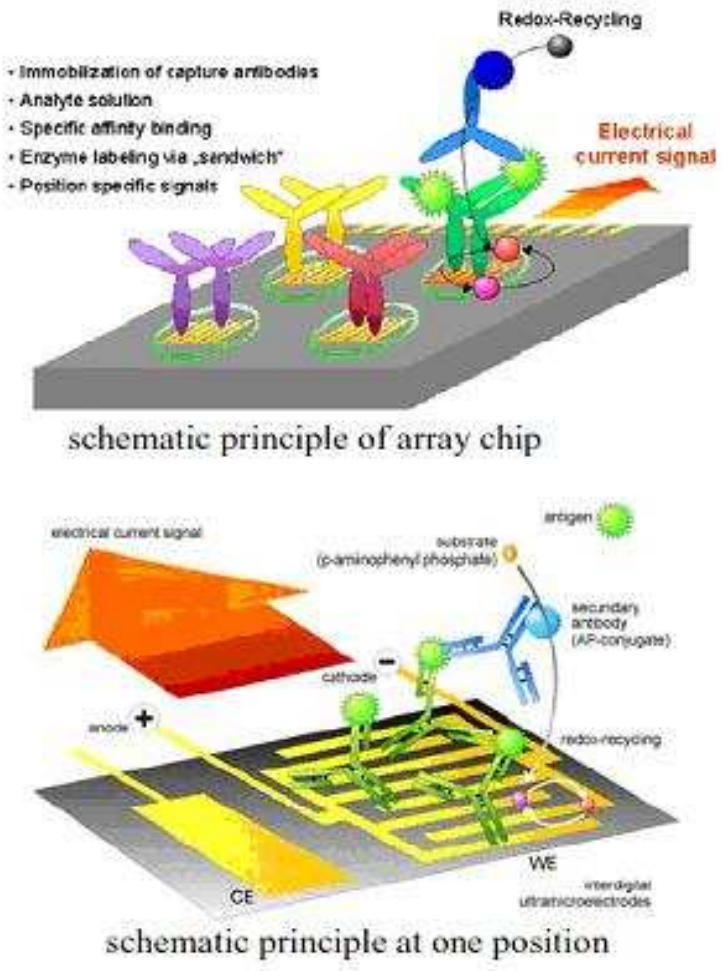

Figure 48: schematization of biochip working principle [45].

A biosensor is an array of microelectrodes able to detect proteins, acids and small molecules with many different applications such as disease/virus diagnostic [46], toxins discovery [47] or environment pollution monitoring [48]. It bases its working on a chemical reaction on the sensor surface between the target molecules and a proper reagent, which generates different current time profiles on the electrodes for different target substances. The electronic interface must detect these small currents (in the range of nA) with extremely high sensitivity (up to few pA) from the electrodes array, also providing some digital signal processing like noise filtering and time average. The samples have to be transmitted periodically to a host PC for current profiles displaying and substance detection (made by specific software).

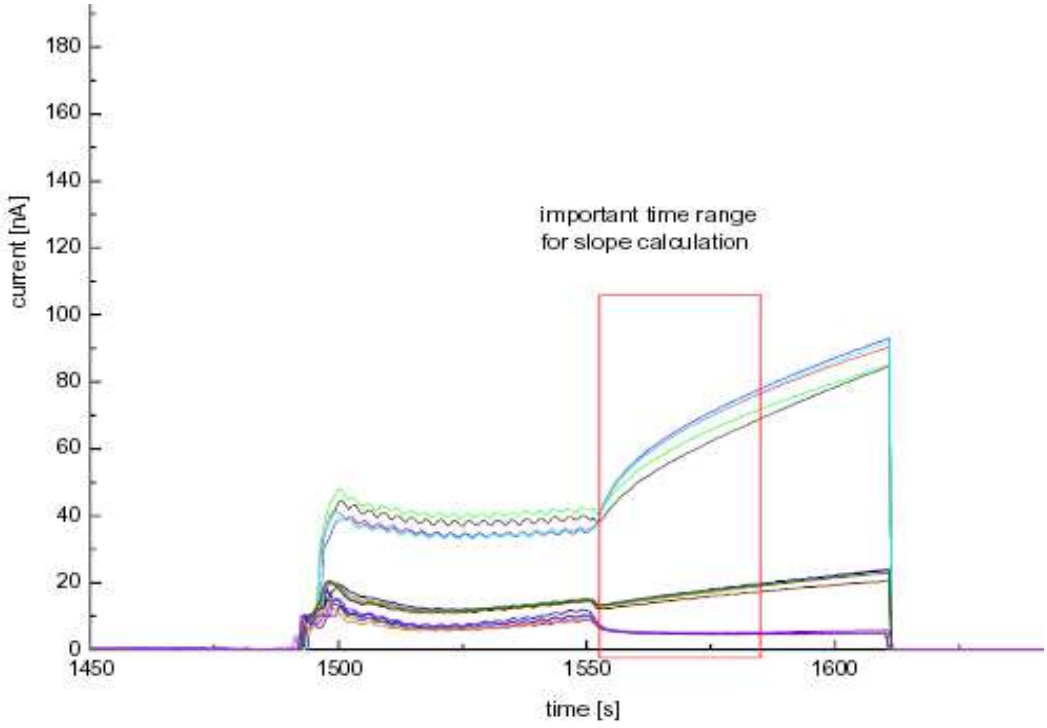

Figure 49: Example of biosensor measure.

In our implementation ISIF input channels have been configured for high sensitivity current detection; analog data is then converted and digitally processed by DSP software routines, while LEON forwards output data (via UART) to a LabVIEW™ interface for screen displaying and post-processing. The current profiles of all the electrodes over a long time scale must be provided in order to discriminate between the manifold chemical agents which can interact with the biosensor. Firmware must also handle the input channel multiplexing as 16 electrodes have to be monitored at the same time with only 4 input channels available within ISIF: a

timed loop performs in a sequence the input switch matrix re-configuration (connecting ISIF to 4 out of 16 electrodes), data acquisition and related processing (such as time average) and then forwards outputs to the UART for displaying on the LabVIEW™ interface.
The setup of the high-sensitivity acquisition procedure and the firmware development was completed in few weeks, achieving a sensitivity of 5 pA (noise standard deviation of 3.4 pA/√Hz on a 10 Hz output bandwidth) and fulfilling all the biosensor conditioning requirements.

iii. Gyro sensor

Gyro sensors provide yaw rate measurement by working accordingly to the Coriolis Effect [28]. While two sensor electrodes are driven with a sine wave (in order to keep it oscillating at its resonating frequency), an angular rotation of the device causes a perpendicular oscillation, which can be detected by measuring the capacitance variation at the sense electrodes (a gyro sensor schematization is shown in Figure 50).

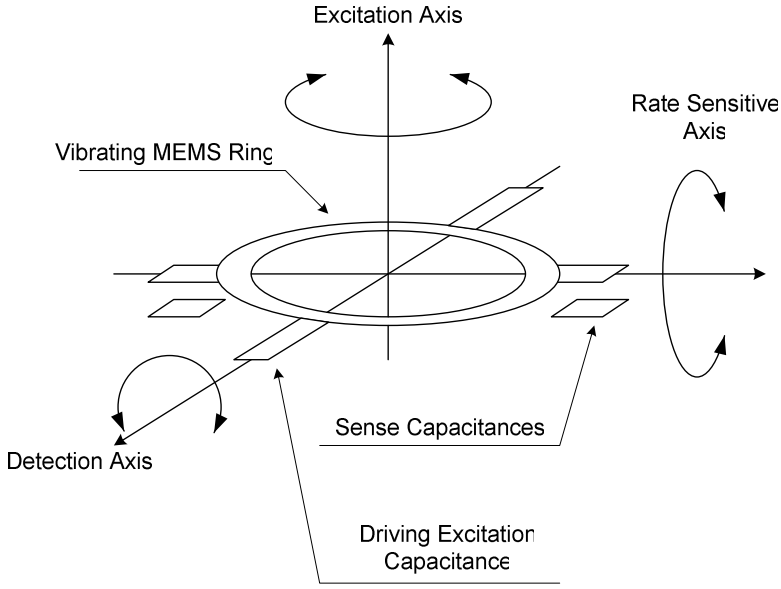

Figure 50: Gyro sensor vibrating, sense and rate axis.

For such application the electronic interface has to provide a sine driving by a PLL (Phase-Locked Loop) with the capability of locking at the gyro resonating frequency (around 5-10 KHz), meanwhile an Automatic Gain Control (AGC) keeps its amplitude at a safe level: more driving implies better SNR but excessive amplitude makes the moving structure hit the fixed frame and causes sensor malfunctioning.

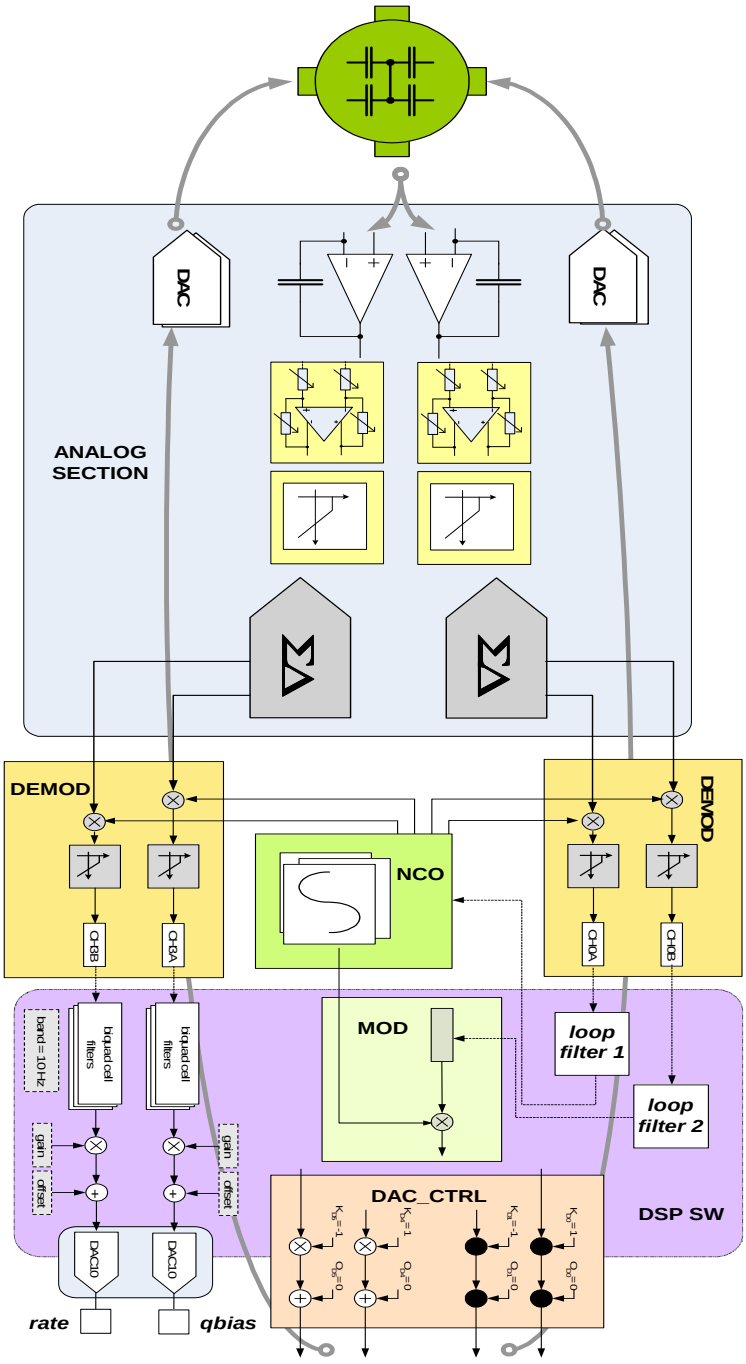

Figure 51: signal processing architecture for gyro sensor.

A filtering path (with gain and offset compensation) is required after demodulation for the detected rate signal. ISIF platform implements this circuitry by using two input channels with acquisition stage in charge amplifier configuration, sigma-delta converters feed the hardware digital demodulator, which combines the signals from the sensor with the sine waves coming from the NCO (Numeric Controlled Oscillator). The DSP software routines add biquadratic cells implementing generic transfer functions for loop filtering, such as Proportional Integral Derivative (PID)

controller, which are needed in PLL and AGC loops, plus gain adjustments and offset compensations, as depicted in Figure 7. Another sine wave from NCO (whose frequency is set by PLL control and whose amplitude comes from AGC loop) is forwarded to the DAC controller which provides the needed 4 driving signals.
The customization of the whole system for gyro sensor conditioning, including optimal input channel gains and bandwidth setting, hardware IPs configuration and software modules setup has been performed in about one week, and resulted in a working system with a sensitivity of 20 mV/°/s and a rate noise of 0.4 °/s on a 10 Hz output bandwidth.
The results achieved by ISIF fast prototyping (despite the short time required to develop this implementation) can be compared with those of the interface dedicated to gyro described in Chapter 2. Rate noise stays at higher levels because of the slightly higher parasites of the signal acquisition stage (input switch matrix and programmable R and C for charge amplifier), yet it still reaches the levels of the commercial chips from Analog Devices [29] or Murata [30] reported as a reference. The optimized interface for gyro sensor can be designed using the ISIF prototype a starting point, reusing most of its blocks (simply cutting unneeded configuration options) and thus minimizing area, power consumption and parasitics, with a strong confidence in achieving a first time successful silicon with superior performances.

iv. YZ low-g accelerometer

MEMS accelerometers are capacitive inertial sensors which consist of a proof mass suspended by beams anchored to a fixed frame [49]. The device acceleration causes a displacement of the proof mass with respect to the support frame, resulting in a capacitance variation on the electrode between the moving and the fixed structure.
Most of the accelerometers available today in literature [50][51] and on the market [52] are called ‘in plain’ as they feature a lateral structure similar to that depicted in Figure 52. The sense fingers attached to the proof mass are interdigitated with the fingers on the fixed frame, forming parallel differential capacitor elements. In these devices the proof mass moves along the silicon plane (XY plane).

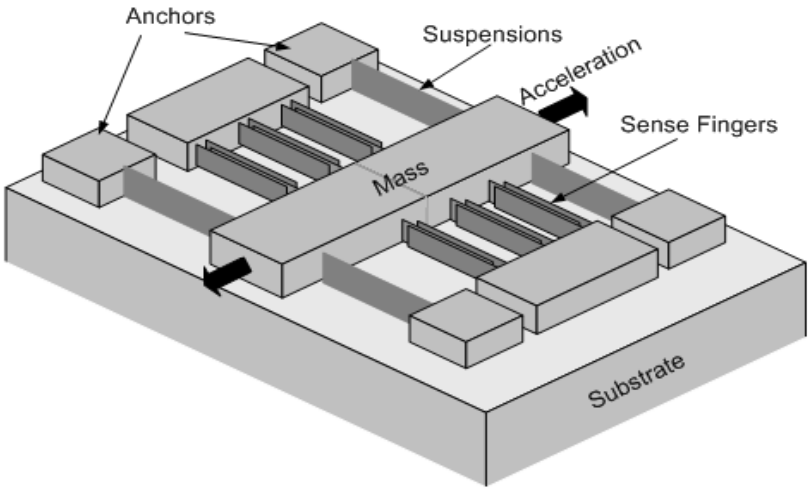

Figure 52: lateral structure for MEMS accelerometers.

A vertical structure instead, as depicted in Figure 53, consists in a suspended mass without fingers which is attached by beams to the fixed anchor and changes its position with respect to the ground plate electrode according to the external acceleration [53]. These accelerometers detect an out-of-plane acceleration and are less common as they feature intrinsic non linearity, thus requiring a more complex signal conditioning (for example a closed loop control makes the system work always around the rest position and is needed to improve linearity). Vertical accelerometers find their best application when combined on the same chip with a lateral one, in order to implement a dual axis accelerometer with out-of-plane acceleration detection, as they result much cheaper than two single axis devices mechanically mounted onto perpendicular planes.

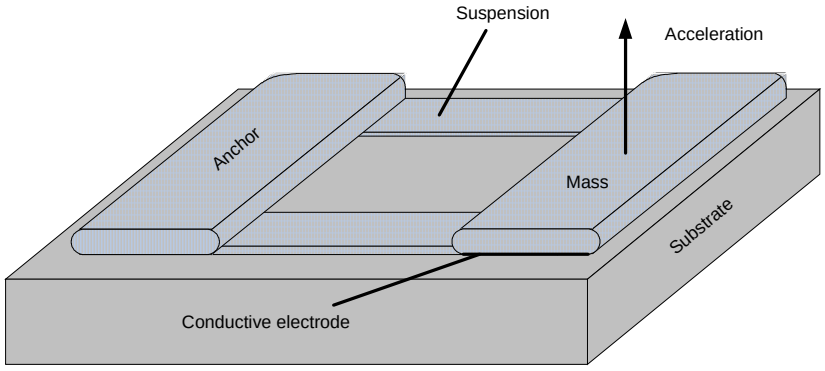

Figure 53: vertical structure for MEMS accelerometers.

Among the described accelerometers, the low-g category includes those with full scale within few g (3 to 10) and fits automotive applications as acceleration detection for mechanical assistance (ABS to ACC) and tilt detection in anti-theft systems.

Typical values of sensing capacitances in low-g accelerometers are 1-100 pF. Readout circuits for accelerometers often use capacitance-to-frequency converters [54][55] capacitive AC-bridges [56] or switched capacitor circuits [57]. By means of the correlated double-sampling technique the switched capacitor circuits can reduce the effect of 1/f noise [58][59].
The targeted dual axis YZ accelerometer features an in-plain moving structure which shifts on Y axis causing a ΔC variation of about 10 fF/g in the sense capacitances, while its rest value is about 10 pF. The Z moving structure, as mentioned above, presents poor linearity, but thanks to its feedback electrodes a closed loop control can be implemented. A sweep on the feedback electrodes has been performed in order to evaluate the extent of acceleration that could be compensated in closed loop (and thus the maximum full scale of the application). Results shown in Figure 54 prove that Y sensor can compensate less than +/- 1g and therefore no closed loop is possible (but as well it is not needed being the sensor structure intrinsic linear), while Z axis has excellent feedback capabilities up to +/- 4g, suitable for the target low-g FS of +/- 2g.

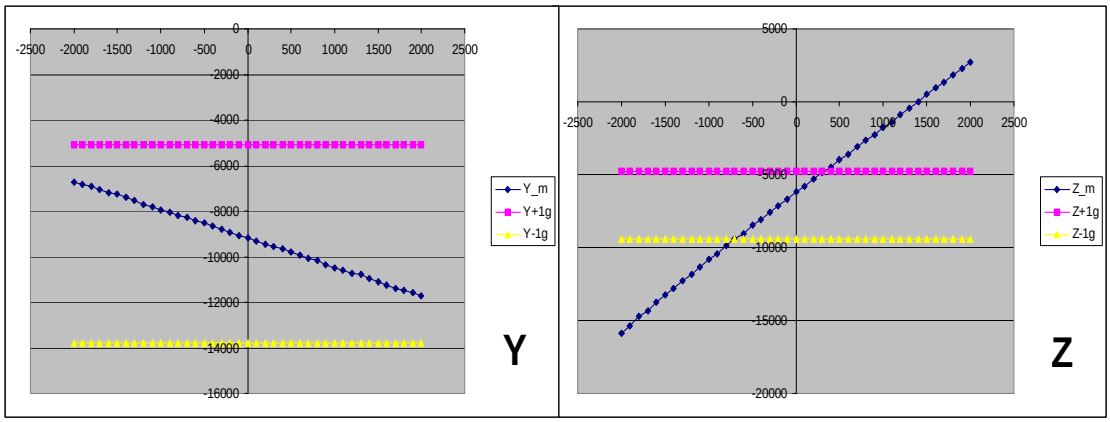

Figure 54: Feedback sweep on Y and Z sensors.

In the interface implementation with ISIF, charge amplifier differentially reads the slight ΔC variation at the sense electrodes, while the other input channel stages perform further low pass filtering and gain adjustment, so that the input dynamic of the 12 bit ADC converter can be fully exploited.
As shown in Figure 519, closed loop on Z axis is implemented by means of software-emulated IPs for reference subtraction, PI controller and feedback actuation, while the Z acceleration value is calculated as the difference between feedback driver inputs. Such signal requires further low pass filtering (at the target bandwidth of 10 Hz), offset compensation and gain correction in order to have

the sensitivity in compliance with specifications. Y axis acceleration signal coming from the analog section is directly low pass filtered and sent to the offset and gain correction stages.
A series of measurements on several samples (three of them are reported in Figure 56 and Figure 57) showed quite strong drift over temperature for both Y and Z acceleration, because of a combination of effects on the sensor itself and in the ISIF analog conditioning stages. The overall drift showed an approximately linear trend over T, so software T compensation has been set up by means of further offset and sensitivity correction coefficients calculated as a linear function of T (the approximation employ two trend lines for each parameter to be corrected, as shown in Figure 55).

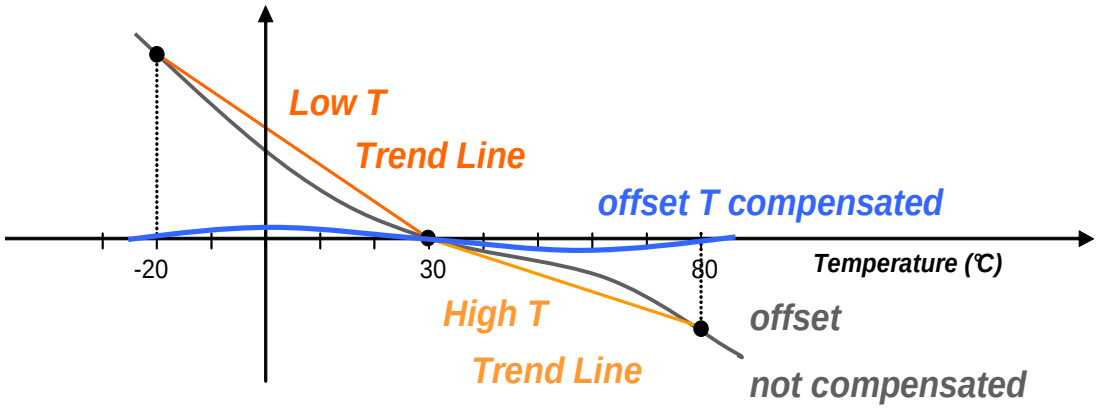

Figure 55: T compensation procedure.

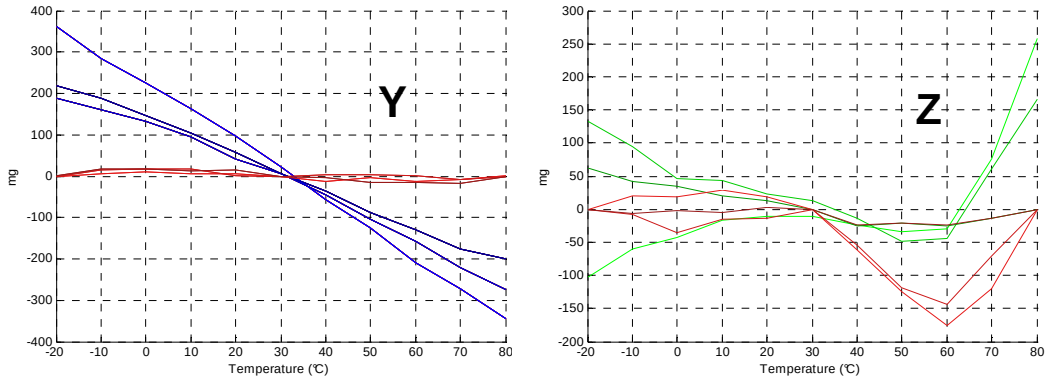

Figure 56: offset over T characterization and compensation (red lines) with 2 linear trend line approximation.

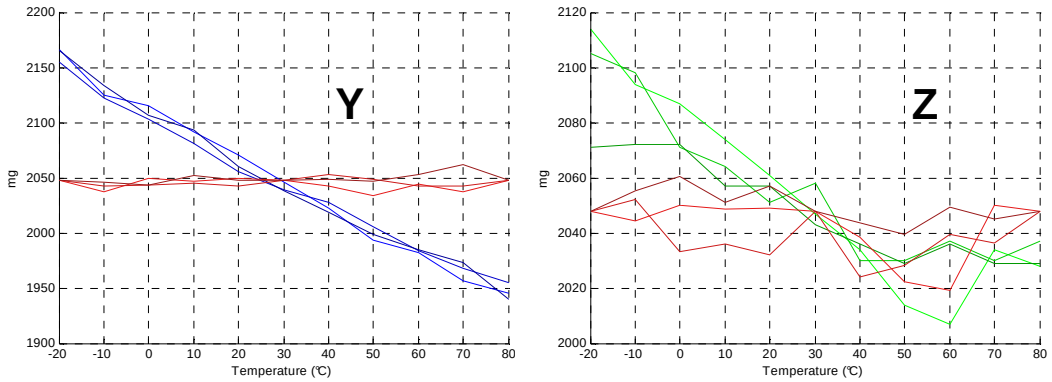

Figure 57: sensitivity over T characterization and compensation (red lines) with 2 linear trend line approximation.

In addition a set of trimming procedures have been developed in order to automatically set few important system parameters, such as Z reference for closed loop (the value read by the digital part with the sensor in open loop and position Z = 0 g), the sensitivity and offset correction coefficients and the T dependant factors. First, Z reference has to be set with an average value read at the end of analog and digital signal conditioning when the sensor is in Z = 0g position, then loop can be closed with confidence on the fact that the moving structure will stay in its rest position (thus improving linearity). Another procedure, to be started when the sensor is rotating along the sensitive axis at about 25 °/s, detects maximum and minimum acceleration values on both axes in order to compensate offset and set gain coefficients for the target sensitivity. The same routine, run with the sensor inside the climate chamber at three different temperatures, allows calculation of 2 linear trend lines both for offset and sensitivity, so that with proper conversion factors T compensation coefficients can be calculated at runtime (trend lines are estimated within the ranges $T_{low} - T_{env}$ and $T_{env} - T_{high}$). The described implementation has been first evaluated with setups made by a PCB hosting two CLCC84 sockets for ISIF and accelerometer. A smarter setup is depicted in Figure 58: it consists in a very small PCB (about 3 cm^2 area) with ISIF and sensor dies glued on and bonded together, then covered by a cap. Such setup minimizes interconnection parasitics and is fit for commercialization. Thanks to its reduced size it also allowed parallel trimming and characterization of several modules in the turning table, with noteworthy time savings.

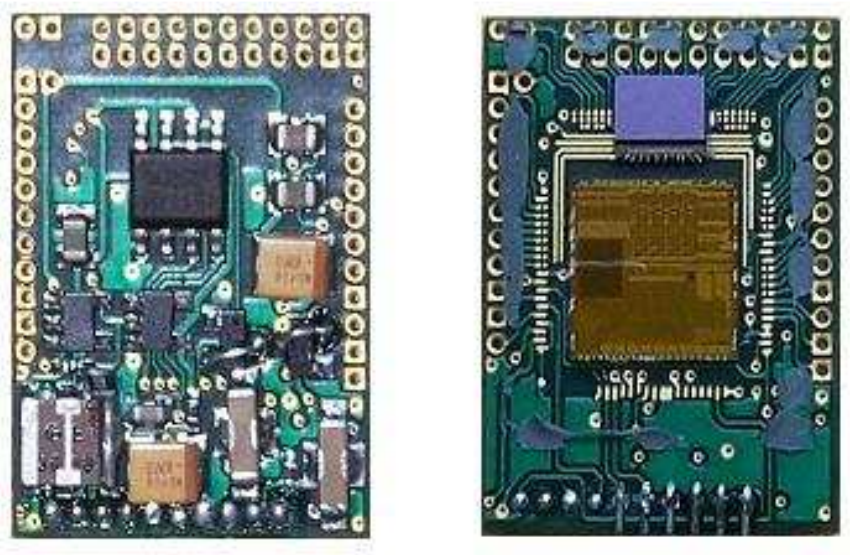

Figure 58: miniboard with ISIF and sensor dies bonded together.

The system provides digital output via UART or SPI, with sensitivity of 1024 LSB/g and FS of +/- 2g. The noise, which can also be evaluated by firmware, stays within 3 mg for both axes (with 10 Hz bandwidth).
The linear procedure for compensation over T resulted in a max offset drift of 20 mg on Y and 40 mg on Z, and with sensitivity error within 1% of FS on Y and 3% of FS on Z.
With respect to commercial devices, as for example ADXL322 [52], this implementation features a slightly higher noise but better 0g offset and stability over T.

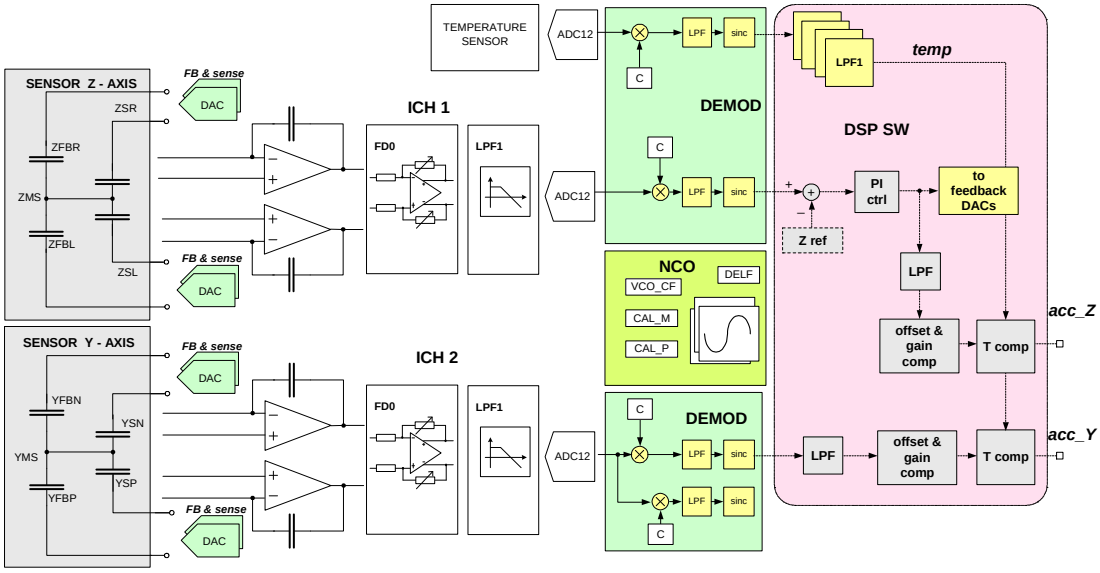

Figure 59: processing architecture for YZ accelerometer.

The ISIF prototyping for dual-axis YZ accelerometer had a twofold aim. As a first goal, the architectural space was explored in order to find the best conditioning path as depicted in Figure 59. This task is a step of the enhanced platform based design flow and was needed to start the development of a optimized interface for automotive applications, which implements the equivalent architecture fully in hardware (for safety reasons) without the ISIF configuration capabilities (in this way gaining immediately better performances because of reduced parasites and area/power optimization). The second goal was to extend the prototyping phase (which typically requires short time and is performed on few samples only) to a comprehensive characterization of the sensor and the ISIF modules, investigating the issues concerning the used blocks and the possible improvements to be implemented in the dedicated ASIC. This phase exploits the firmware trimming functions described above, which automatically set the modules in working conditions with compensated offset and desired sensitivity, plus a set of procedures for sensor and miniboard characterizations.

a) *Sensor mechanical transfer function*: a different ISIF configuration with respect to the standard accelerometer working mode of Figure 60 allows the measurement of the sensor mechanical transfer function. Keeping the device in 0g position for both axes, the NCO generates a sine wave which is directly fed to Y and Z feedbacks, thus stimulating the sensor at the given frequency. The input channel detects the sense capacitance variation as normal and the acquired waveform (representing the moving mass displacement only due to the feedback driving) is demodulated in phase and quadrature and filtered in order to calculate by firmware its amplitude $A = \sqrt{I^2 + Q^2}$.

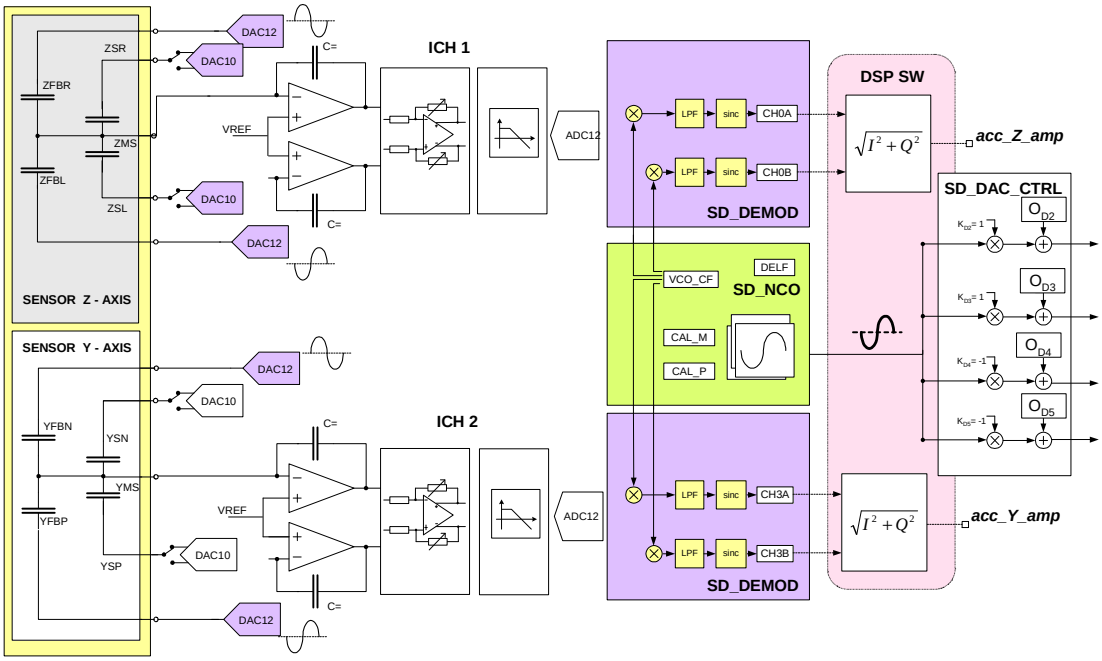

Figure 60: architecture for evaluating mechanical transfer function of the sensor.

The evaluation is performed at frequencies from 20 to 3000 Hz and it has included about 100 sensors in different packages and with different internal pressures, disclosing a low pass mechanical behaviour for Z sensors (with 200 to 500 Hz bandwidth) while Y features a resonating peak around 1.7 KHz with Q factors ranging from 3 to 6 (of course mechanical damping acts a higher frequencies). The investigation on sensors mechanical behaviour provides important feedback to the sensor engineers (who can validate related data coming from simulations) and helps them to better understand how certain sensor parameter (among which sizing and internal pressure) affect this behaviour. The aim in automotive applications is to provide sensors which mechanically feature low pass filter behaviour with as low bandwidth as possible (theoretically matching the specification on acceleration bandwidth), in order to minimize noise at the very source and also prevent any malfunctioning due to environmental mechanical stresses (like electrostatic sticking or beam damages).
The results of this investigation are shown in Figure 61 and Figure 62.

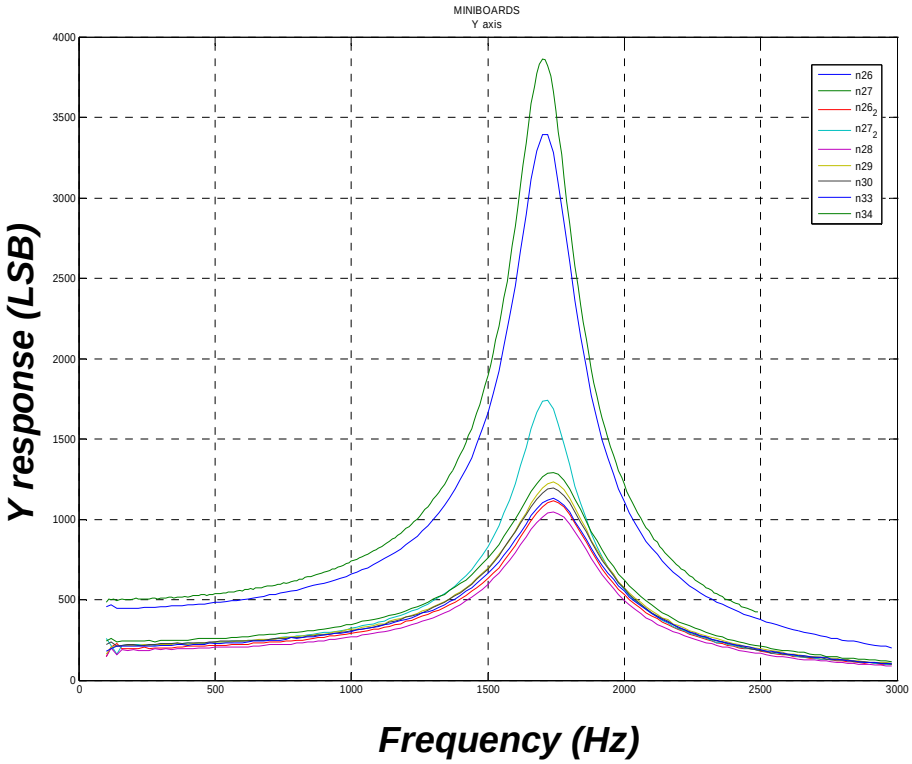

Figure 61: mechanical transfer functions of Y sensors.

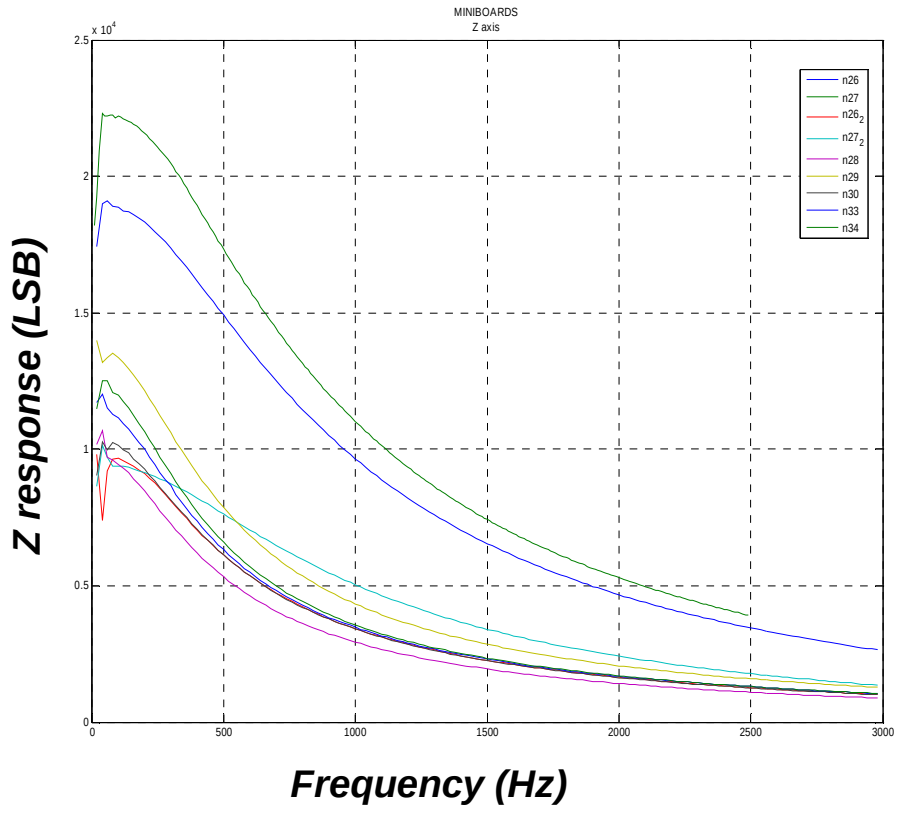

Figure 62: mechanical transfer functions of Z sensors.

b) *Offset and sensitivity drift over temperature*: this study has mostly involved samples in the miniboard setup shown in Figure 58. Once a module is working and trimmed, it is important to verify the success of trimming procedures and also have a statistical database of sensor behaviours over the chosen temperature range. Stability over T is a crucial factor for automotive sensors and such investigation aims to providing all the needed information for the improvement of the critical blocks which affect predominantly the drift over temperature (from analog acquisition stage to T sensor and T compensation strategy). For this reason a series of procedures analogue to the trimming ones have been set up in order to characterize a wide lot of samples in the shortest time and with lowest human employment possible. The characterization environment includes a parallel board setup for up to 6 modules concurrent evaluation (the same used for parallel trimming, see Figure 63) with a LabVIEW™ interface controlling turning table, climate chamber and handling SPI communication with the modules.

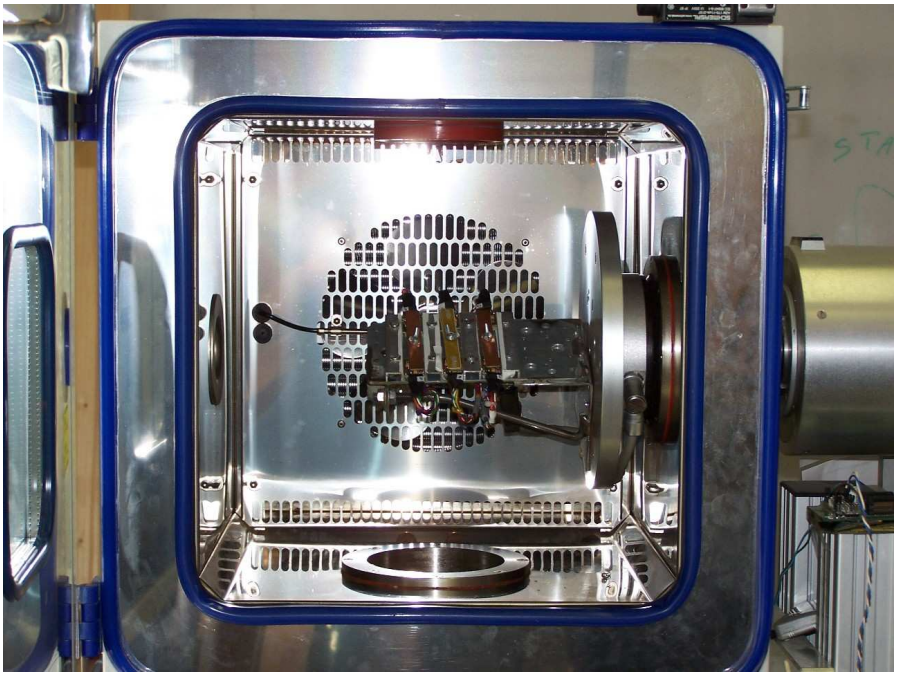

Figure 63: parallel board inside turning table for simultaneous trimming and characterization of up to 6 modules.

The characterization procedure is depicted in Figure 64. The LabVIEW™ interface acts as a master repeating a number of times the characterization cycle, which consists in setting a temperature in the climate chamber, waiting 3 minutes after it has been reached (for thermal equilibrium of all components) and then

sending a command to each module via SPI and reading back the results (temperature read by ASIC, offset and sensitivity for Y and Z axis). The turning table is rotating at 25°/s during the whole procedure to let both sensors cross all the positions between +/- 1g and allow the firmware calculation of sensitivities and offsets. This characterization has been typically performed between -20°C and 80°C with 10°C steps and the results for a 20-samples lot is displayed in Figure 65 and Figure 66.

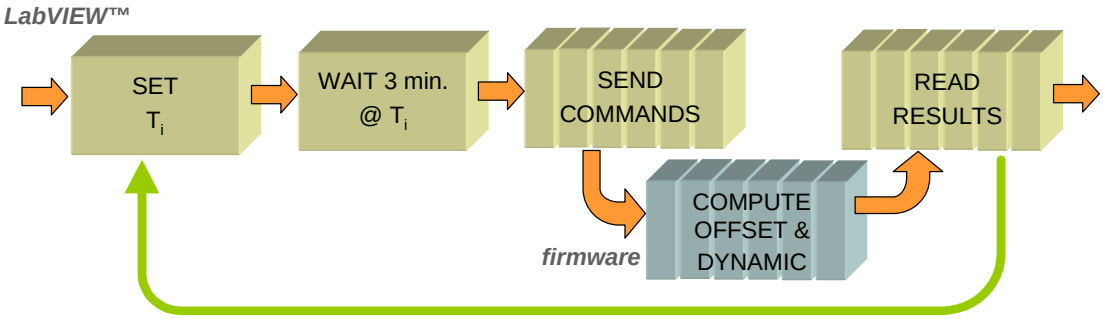

Figure 64: Characterization procedure.

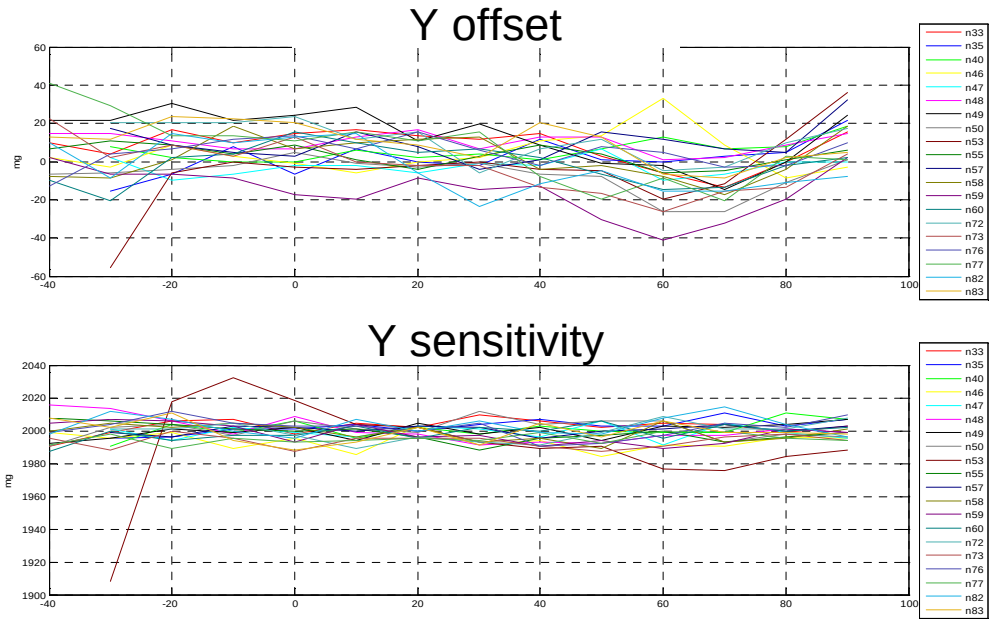

Figure 65: characterization of Y offset and sensitivity (20 samples).

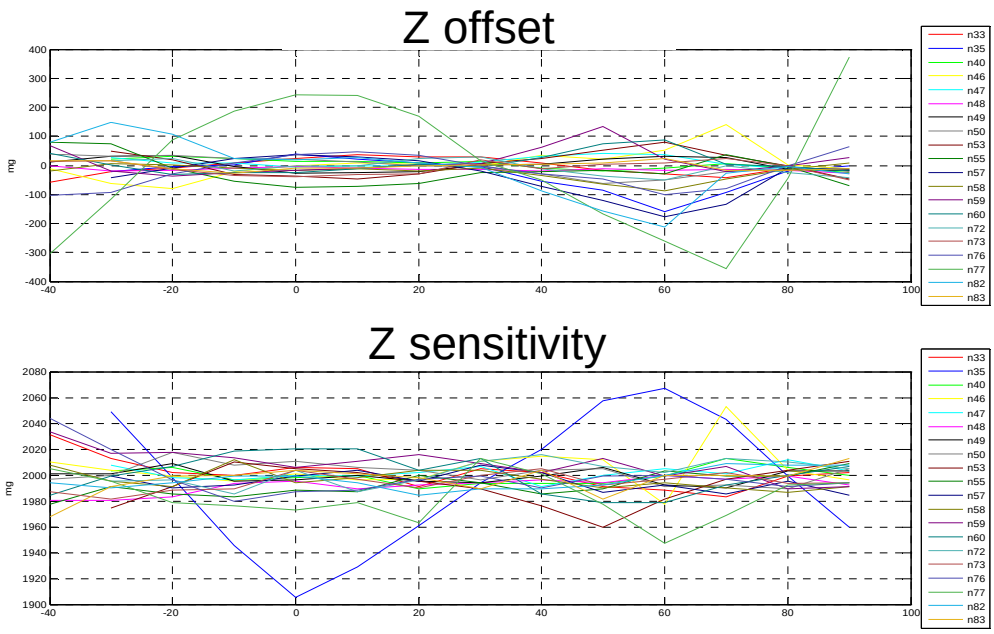

Figure 66: characterization of Z offset and sensitivity (20 samples).

Bibliography

[42] F. D'Ascoli, M. Tonarelli, M. Melani, M. De Marinis, a. Giambastiani, L. Fanucci, "Intelligent sensor interface for automotive applications" ICECS2005 Gammarth Tunisia Dec 11-14

[43] 'Giant magnetoresistive effect', Wikipedia, http://en.wikipedia.org/

[44] J. Daughton, "GMR and SDT sensor applications", Magnetics, IEEE Transactions on, Volume 36, Issue 5, Part 1, Sept 2000 Page(s):2773 – 2778

[45] eBiochip Systems GmbH, www.ebiochip.com

[46] Yobas, L.; Wing Hui; Hongmiao Ji; Yu Chen; Liw, S.S.I.; Jing Li; Choong Ser Chong; Xie Ling; Chew Kiat Heng; Lye, H.J.; Siti Rafeah Bte; Lee, K.; Sanjay Swarup; Sek Man Wong; Tit Meng Lim; "Microfluidic chips for viral RNA extraction & detection", Sensors, 2005 IEEE, 30 Oct.-3 Nov. 2005 Page(s):4 pp.

[47] Radke, S.M.; Alocilja, E.C.; "Design and fabrication of a microimpedance biosensor for bacterial detection", Sensors Journal, IEEE, Volume 4, Issue 4, Aug. 2004 Page(s):434 – 440

[48] Starodub, N.F.; Katzev, A.M.; Levkovetz, I.A.; Goncharuk, V.V.; Klimenko, N.A.; Vakulenko, V.F.; "Biosensor based on the photoluminescent bacteria and its use for express control of water contamination by some surface active substances", TRANSDUCERS, Solid-State Sensors, Actuators and Microsystems, 12th International Conference on, 2003, Volume 2, 8-12 June 2003 Page(s):1197 - 1200 vol.2

[49] N. Yazadi, F. Ayazi and K. Najafi, "Micromachined Inertial Sensors", Proceedings of the IEEE, Volume 86, Issue 8, Aug. 1998 Page(s):1640 – 1659

[50] Ang, W.T.; Khoo, S.Y.; Khosla, P.K.; Riviere, C.N.; "Physical model of a MEMS accelerometer for low-g motion tracking applications", Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, Volume 2, Apr 26-May 1, 2004 Page(s):1345 - 1351 Vol.2

[51] Bais, B.; Majlis, B.Y.; "Suspension design analysis on the performance of MEMS area-changed lateral capacitive accelerometer", Semiconductor Electronics, 2004. ICSE 2004. IEEE International Conference on, 7-9 Dec. 2004 Page(s):5 pp.

[52] ADXL322 Datasheet. www.analog.com

[53] Pakula, L.S.; Yang, H.; French, P.J.; "A CMOS compatible SiC accelerometer" Sensors, 2003. Proceedings of IEEE, Volume 2, 22-24 Oct. 2003 Page(s):761 - 764 Vol.2

[54] M. S. Smith, L. Bowman, and J. D. Meindl, "Analysis, design, and performance of micropower circuits for a capacitive pressure sensor IC," IEEE J. Solid-State Circuits, vol. SC-21, pp. 1045–1056, Dec. 1986.

[55] Y. Matsumoto, M. Esashi, "Low drift integrated capacitive accelerometer with PLL servo techniques," in Tech. Dig. 7th Int. Conf. Solid-State Sensors and Actuators (Transducers'93), Yokohama, Japan, June 1993, pp. 826–829.

[56] Jiangfeng Wu; Fedder, G.K.; Carley, L.R., "A low-noise low-offset capacitive sensing amplifier for a 50-μg/√Hz monolithic CMOS MEMS accelerometer"; Solid-State Circuits, IEEE Journal of Volume 39, Issue 5, May 2004 Page(s):722 - 730

[57] Amini, B.V.; Ayazi, F., "A 2.5-V 14-bit ΣΔ CMOS SOI capacitive accelerometer"; Solid-State Circuits, IEEE Journal of Volume 39, Issue 12, Dec. 2004 Page(s):2467 - 2476

[58] Junseok Chae; Kulah, H.; Najafi, K., “An in-plane high-sensitivity, low-noise micro-g silicon accelerometer with CMOS readout circuitry”; Microelectromechanical Systems, Journal of Volume 13, Issue 4, Aug. 2004 Page(s):628 - 635

[59] Seungbae Lee; Gi-Joon Nam; Junseok Chae; Kim, H.; Drake, A.J.; “Two-dimensional position detection system with MEMS accelerometers, readout circuitry, and microprocessor for padless mouse applications”; Very Large Scale Integration (VLSI) Systems, IEEE Transactions on; Volume 13, Issue 10, Oct. 2005 Page(s):1167 - 1178

CONCLUSIONS

In this work a Platform Based Design Flow for automotive sensor interfaces has been presented. This design approach achieves the development of a sensor interface with optimized performances, as required by the automotive standards, and yet keeps a short time-to-market and low development costs thanks to the deep reuse of design parts and the generic features of the highest platform layers. The generic platform for inertial sensor conditioning has been implemented on ASIC for a gyro sensor application (in collaboration with Sensordynamics AG), featuring lower noise and better offset and sensitivity stability over temperature with respect to commercial state-of-the-art devices.

Moreover, the Platform Based Design flow has been substantially enhanced with the implementation of ISIF platform, a mixed-signal System On Chip which allows a fast prototyping of the sensor interface, letting the designer find the best conditioning architecture in a short time, thus improving time-to-market and final ASIC performances.

Several case studies have been address, with the aim of proving ISIF capabilities of conditioning different sensor classes, like magneto-resistive sensors, biochips and capacitive inertial sensors (accelerometers and gyros) with outstanding performances despite the short time needed for prototype setup.

As a matter of fact, the ease and time savings in architectural space exploration is the main achievement of ISIF interface employment: for the presented case studies, working conditioning architectures have been setup in times ranging from 1 day to 1 week, with huge benefits with respect to traditional investigations performed through system level simulations. These benefits include mainly development cost savings and accuracy of results, but they also extend to the possibility of carrying out quick feasibility studies and in depth analysis of the chosen architecture for detecting eventual weak points or critical blocks, in order to

further improve the performance in a dedicated ASIC implementation derived from ISIF prototype.
Despite the ISIF lack of optimization for any specific sensor application, valuable results have been obtained by ISIF case studies. For example, the magneto-resistive sensor application featured angular position sensitivity of only 0.1°, while the biosensor conditioning has showed ISIF extremely high capability of low current measurement (up to few pA) together with the needed flexibility for time multiplexing of the input channels.
The setup of high performance interfaces for capacitive inertial sensors has been a more challenging test for ISIF. The gyro sensor case study has proved that ISIF is fit also for implementing very complex conditioning architectures, resulting in a rate noise of 0.4°/sec which is still competitive with most commercial devices, though higher than the one obtained by the dedicated ASIC implementation of the generic inertial platform.
Also the YZ low-g accelerometer prototype has achieved excellent performances for what concerns noise (3 mg on 10 Hz output bandwidth) and stability over temperature: max offset drift of 20 mg on Y and 40 mg on Z, and sensitivity error within 1% of FS on Y and 3% of FS on Z. This has been possible thanks to the full exploitation of ISIF resources in a proper miniboard setup, including the development of automatic trimming and characterization firmware procedures and the evaluation of the sensor mechanical transfer function. Such investigation has involved about 100 modules, providing valuable information on all the critical aspects of the sensor-interface couple, in order to achieve superior performances in the optimized ASIC developed for this application after ISIF prototyping phase.