UNIVERSITÀ DI PISA

Scuola di Dottorato in Ingegneria "Leonardo da Vinci"

Corso di Dottorato di Ricerca in
INGEGNERIA DELL'INFORMAZIONE

Tesi di Dottorato di Ricerca

FLEXIBLE LOW-COST HW/SW ARCHITECTURES FOR TEST, CALIBRATION AND CONDITIONING OF MEMS SENSOR SYSTEMS

Autore:

Francesco Sechi _____

Relatori:

Prof. Luca Fanucci _____

Prof. Roberto Saletti _____

Anno 2011
SSD ING-INF/01

A Simona
Alla mia famiglia

ACKNOWLEDGMENTS

A lot of people attend me for the whole period of my PhD course, but some in particular deserve a special thanks.
I would like to say thank you to my parents and my sisters for their essential support: I would have never reached this goal without your protection and the warmth of your words.
Another very important person I would like to thank you is Simona, because her love gave me the strength to get over all the difficulties I met during this period.
Then my gratitude goes to my tutors Prof. Luca Fanucci and Prof. Roberto Saletti both for their precious technical help and for the several chances of professional improving they provided to me.
A special thanks goes to the SensorDynamics' design center of Navacchio, particularly to Alessandro Rocchi and Adolfo Giambastiani for their precious technical and professional teaching, and for giving me the chance to explore the amazing world of electronic!
Last but not least, a great thanks goes to all my friends, especially to Sergio, Alberto, Chiara, Marcolino, Weiwor, Claudioca', Agostino, Paolo and Fabio&Enrica, and to all my cousins and relatives.

SOMMARIO

Negli ultimi anni gli smart sensor basati su sistemi Micro-Elettro-Meccanici (MEMS) si sono largamente diffusi in diversi settori quali l'Automotive, il settore biomedico, quello ottico e il consumer, e oggigiorno rappresentano il più avanzato stato dell'arte. Le ragioni della loro diffusione sono legate alla loro capacità di misurare quantità fisiche e chimiche usando componenti miniaturizzati.
Lo sviluppo di questo tipo di architetture, per via della eterogeneità delle loro componenti, richiede un flusso di progetto molto più complesso per via della presenza sia di parti meccaniche proprie del sensore MEMS, sia di componenti elettroniche per l'interfacciamento e il condizionamento.
In questo tipo di sistemi acquista una notevole rilevanza l'attività di testing, che interessa varie fasi del ciclo di vita di un sistema basato su sensori MEMS. Infatti, sin dalla fase di design del sensore è importante validare il progetto estraendone i parametri caratteristici, che saranno utili durante la fase di design del circuito di interfaccia e condizionamento. Inoltre, un'architettura di questo tipo richiede, oltre ai tradizionali metodi di test che riguardano la circuiteria di controllo, anche tecniche per la calibrazione e la valutazione dell'intero sistema.
La prima parte di questo lavoro di ricerca affronta il tema dell'ottimizzazione del testing mediante lo sviluppo di differenti architetture hardware/software per le diverse fasi di test che fanno parte del flusso di sviluppo di un sistema basato su sensori MEMS. E' stata sviluppata una piattaforma flessibile e a basso costo per la caratterizzazione e la prototipazione di sensori MEMS, così da offrire un ambiente che permetta inoltre di supportare la progettazione dell'interfaccia per sensore. Per ridurre il tempo di ingegnerizzazione richiesto durante la fase di "verification testing" è stata progettata un'architettura client-server universale che offre un unico framework per il test di diversi tipi di dispositivi, permettendo l'utilizzo di differenti tipi di ambienti di sviluppo e linguaggi di programmazione. Dato che l'uso di macchine ATE durante lo sviluppo dell'algoritmo di calibrazione è costoso in termini di tempo di occupazione della macchina stessa, poiché un suo utilizzo durante questa fase richiederebbe la sospensione del processo di produzione, si è proposta un'architettura hardware/software a basso costo e facilmente adattabile per la calibrazione e la valutazione delle performance che consente lo sviluppo dell'algoritmo di test in un ambiente user-friendly e permette inoltre di realizzare una produzione su piccola e media scala.
Nella seconda parte del lavoro di ricerca si è approfondito un argomento che sta acquisendo una sempre maggiore importanza nell'ambito dello sviluppo di applicazioni per sensori MEMS, e riguarda la possibilità di combinare le informazioni estratte da diversi tipologie di sensori (tipicamente accelerometri, giroscopi e magnetometri) per ottenere informazioni più complesse. In questo contesto sono stati sviluppati e analizzati due differenti algoritmi di sensor fusion: il primo è un algoritmo puramente software che è stato utilizzato come strumento per valutare quanto un errore nella misura del MEMS può inficiare la stima del parametro calcolato usando un algoritmo di sensor fusion; il secondo, invece, è un algoritmo di sensor fusion basato su un filtro di Kalman semplificato. Di questo algoritmo è stato creato anche un modello bit-true in Mathworks Simulink™ che verrà usato come studio di sistema per l'implementazione dell'algoritmo su chip.

ABSTRACT

During the last years smart sensors based on Micro-Electro-Mechanical systems (MEMS) are widely spreading over various fields as automotive, biomedical, optical and consumer, and nowadays they represent the outstanding state of the art.
The reasons of their diffusion is related to the capability to measure physical and chemical information using miniaturized components.
The developing of this kind of architectures, due to the heterogeneities of their components, requires a very complex design flow, due to the utilization of both mechanical parts typical of the MEMS sensor and electronic components for the interfacing and the conditioning.
In these kind of systems testing activities gain a considerable importance, and they concern various phases of the life-cycle of a MEMS based system. Indeed, since the design phase of the sensor, the validation of the design by the extraction of characteristic parameters is important, because they are necessary to design the sensor interface circuit. Moreover, this kind of architecture requires techniques for the calibration and the evaluation of the whole system in addition to the traditional methods for the testing of the control circuitry.
The first part of this research work addresses the testing optimization by the developing of different hardware/software architecture for the different testing stages of the developing flow of a MEMS based system. A flexible and low-cost platform for the characterization and the prototyping of MEMS sensors has been developed in order to provide an environment that allows also to support the design of the sensor interface. To reduce the reengineering time requested during the verification testing a universal client-server architecture has been designed to provide a unique framework to test different kind of devices, using different development environment and programming languages. Because the use of ATE during the engineering phase of the calibration algorithm is expensive in terms of ATE's occupation time, since it requires the interruption of the production process, a flexible and easily adaptable low-cost hardware/software architecture for the calibration and the evaluation of the performance has been developed in order to allow the developing of the calibration algorithm in a user-friendly environment that permits also to realize a small and medium volume production.
The second part of the research work deals with a topic that is becoming ever more important in the field of applications for MEMS sensors, and concerns the capability to combine information extracted from different typologies of sensors (typically accelerometers, gyroscopes and magnetometers) to obtain more complex information. In this context two different algorithm for the sensor fusion has been analyzed and developed: the first one is a fully software algorithm that has been used as a means to estimate how much the errors in MEMS sensor data affect the estimation of the parameter computed using a sensor fusion algorithm; the second one, instead, is a sensor fusion algorithm based on a simplified Kalman filter. Starting from this algorithm, a bit-true model in Mathworks Simulink™ has been created as a system study for the implementation of the algorithm on chip.

TABLE OF CONTENTS

INDEX OF FIGURES

INDEX OF TABLES

INTRODUCTION

Micro-electro-mechanical-systems (MEMS) technology is widely spreading during the last years, also thanks to the employment in the consumer applications.
MEMS devices can be defined as micrometers embedded systems whose purpose is to collect physical and chemical information of various kinds about their environment and to make this information available in a form more suitable for the human senses and technical systems.
The design of this kind of devices is very challenging, but also testing activity has a key role in the development of MEMS based device, because of the complexity of a MEMS system architecture that is composed by heterogeneous elements, and each one of these components requires a different approach to verify its correctness.
A MEMS device, in fact, is composed by electrical and mechanical parts, so it requires different test equipment to perform electrical testing and mechanical testing. MEMS testing and reliability assurance are critical processes to achieve high yields and profitability as these processes account for 40 to 70% of the total device cost.
In order to reduce this cost, the main key factor can be identified as the standardization of the test equipment. Indeed, most of the MEMS testing is performed separately for electrical and mechanical elements. This market scenario is changing and there are several test vendors developing key solutions to perform both the electrical and mechanical testing in a single test equipment. Another challenge prevailing in the market today is the non-availability of standardized equipment to test the MEMS devices cost effectively. In fact, current test equipment is priced as high as $1 million, which can deter most MEMS device manufacturers from testing their products. Since customized solutions require vendors to conduct research every time they receive an order, they tend to be expensive. The costs are expected to drop once standard solutions become more prevalent. Most semiconductor industry engineers have a mechanical background or an electrical background, which is sufficient to test any type of semiconductor product. However, testing combined components is more complex than testing them separately, and not many have the technical training required to perform the tests.
To overcome the target of cost effective and time-to-market reduction, the standardization of the test equipment is not enough. Indeed, the life-cycle of a MEMS device consists on several steps, and almost all of these requires different approach to verify and validate its output product.
During the design of the MEMS, simulation and prototyping of the device are required. High volume, high-cost, and accurate measuring systems are necessary to characterize and test the designed device, especially to examine motions, deflections and resonance frequencies of the mechanical structures that are the distinguishing characteristics of these systems. A variety of custom systems relying on interferometry have been developed for deflection measurement but they require a significant amount of development time. Alternatively, there are a variety of commercial deflection measurement systems based on scanning Laser-Doppler

vibrometers. However, even though these systems feature very accurate results, they are often very expensive.
The design of the sensor interface circuit requires an high accurate simulation and validation, and often the use of CAD tools are not enough to verify the correctness of the design, because it is based on approximated models of the MEMS device, produced during the design of the MEMS itself. A solution to perform also a prototyping of the sensor interface will be appreciated, but, for the moment, no solution are commercially available to supply for this lack.
Once the sensor interface is developed, it must be tested. Different typologies of testing have done, depending on the phase of the life-cycle of a chip. The verification testing regards the certification of the correctness of the design and of the test procedure in laboratory, and it is done the first time the chip is designed and fabricated. The manufacturing testing, instead, regards the testing of the fabricated chip in the factory during the production phase. Each device has its own pinout, its interface, and its peculiar characteristic, so the verification testing requires to set up a custom device each time a new device is developed. This reengineering activity entails a growth of costs and time-to-market. So, the reengineering time must be reduced as much as possible, standardizing the applications that interact with the device to provide a common interface that allows to reuse the same tests developed for other products. Some applications are available in the marketplace to achieve this goal, but all of these suffer from different drawbacks, as incompatibility, non concurrent access to the device, absence of libraries to manage laboratory instruments, and so on. The use of this approach, moreover, permits to define the test program directly in the laboratory, reducing the time a test machine is used to debug the test procedure.
The last step of the production flow is the design of the calibration algorithm. The calibration is the procedure of correcting the transfer function of a sensor, using a reference measurement system in order to guarantee a specified input-output relationship with a certain accuracy and under certain conditions. To calibrate a sensor, it is necessary to excite it with an appropriate physical stimulus. To do so, a test machine able to produce this stimulus is necessary. To calibrate a sensor, very expensive test machines are used. These machines allow to calibrate different pieces contemporarily and to handle the socketing of each piece automatically, in order to calibrate pieces continuously. The cost of this machine can be quantified in terms of cost for the equipment and cost for the usage time. While the first one depends on the machine vendor, the second one is dependent on the allocation time disposed by the MEMS device developer. So, in order to minimize the costs, a solution to avoid the use of the production machine for the debug of the calibration algorithm is desirable.
Besides the problematic concerning the development of these smart sensors, the diffusion of this kind of technology in consumer electronic application is causing a continuous demand for new functionalities together with the one offered by the sensors themselves. In fact, differently to other fields the MEMS were traditionally used, the consumer field is very dynamic and requires a continuous innovation in the offered functionalities in order to keep up with the competitors. In the last few years, sensor fusion algorithms, used to combine the outputs of different kind of sensor in order to produce new information, are identified as the future of

microsensor technology, and different companies are equipping their devices with components that implement functionalities on the basis of these algorithms.
This research work deals with two main topics: the first is the development of hardware and software architecture for the optimization of the verification and validation stages in terms of cost effectiveness and time-to-market reduction; the second concerns the analysis and the developing of sensor fusion algorithms.
Chapter 1 presents the MEMS technology by describing its historical development and the fields of applications, then the fundamental steps that characterize the life-cycle of a MEMS are illustrated. The chapter is then concluded by showing the trend of this technology for the next future.
The first part of chapter 2 illustrates a flexible and low-cost platform for the characterization and the prototyping of MEMS sensors. This architecture permits to reduce the time for simulating complex characteristic parameters combining the simulation data with measurements. Moreover, its high reconfigurability permits also to use it for the evaluation of the required sensor interface. The second part describes a pin limited frequency converter bridge for the communication between an ASIC and a FPGA. This architecture is very important during the prototyping phase because it permits to design new functionalities without the need to redesign the ASIC.
The first part of chapter 3 describes a solution for the optimization of the validation phase of the packaged device. The proposed architecture offers a standard framework for the developing of the test environment in laboratory, permitting to reuse the same setup for different systems. The second part shows a low-cost architecture for the calibration of MEMS sensor systems that permits the debugging of a calibration algorithm without using a complex ATE.
Chapter 4 deals with the analysis of algorithm for sensor fusion. In particular, after a brief analysis of the advantages and disadvantages of a purely software development, a solution for an integrated angular position estimation is illustrated. A simplified Kalman filter has been used in order to permit the integration in the sensor interface integrated circuit.

1 TESTING OF MIXED-SIGNAL MEMS SENSOR SYSTEMS

1.1 Introduction

Micro electromechanical systems (MEMS) are probably the smallest functional machines that are currently engineered by humans. They can be defined as embedded systems involving both electronic and mechanical components with a dimensional scale of the order of micrometers and that employ both an electronic movement of charge and mechanical movement for operation. The purpose of microsensors is to collect physical and chemical information of various kinds about their environment and to make this information available in a form more suitable for the human senses and technical systems.

MEMS device development can be traced back to the 1970's, but the first microsensor was developed in 1956, thanks to the discovery of the piezoresistive effect in silicon and germanium. However, it is in the period since 1995 that this technology caught on, because a wide range of new materials and bulk micromachining technologies have become available. Business and scientific world were very interested in MEMS technology, and also many governments heavily funded the research in this field. The reasons for this attention were numerous: first of all, the cost of this technology is reduced, because it scales with its size; second, MEMS devices are characterized by excellent mechanical properties, due to their pure crystalline structure; moreover, they can be fabricated using the same technology infrastructure of the integrate circuit (IC) industry; finally, MEMS devices can be integrated with IC circuitry to create low cost systems on a chip. All these reasons contribute to the proliferation of this technology, particularly the possibility to integrate different components together to perform more complex functionalities. For instance, a single device can integrate multiple sensors and combine the information they produce to extract other kind of information, like position, combining a 3-axis gyroscope and a 3-axis accelerometer.

MEMS is a potent technology and its powerful expresses especially in sensor application. There is now the possibility of designing complex MEMS based systems able to sense and react to their environment and respond and adapt to it. However, even though MEMS technology has been very promising, commercialization efforts have encountered multiple stumbling blocks that have significantly delayed the availability of commercial devices. A first stumbling block was the fact that the IC industry has not provided all process modules required to fabricate mechanical devices (such as wafer deep etching, double side wafer alignment and multiple wafer bonding) and the development of such MEMS processes was long and expansive. Secondly, the mechanical properties of IC materials were unknown and dependent on the fabrication process as well as electrical stability of silicon structures. The third point is that in the IC industry the structure thickness was not an issue, IC industry required an accuracy on thickness of about ±10%, but in MEMS industry the thickness becomes a key point and a better control for mechanical thickness is required. Moreover high volume, low cost MEMS dedicated testing had to be developed entirely by the MEMS industry. And finally the compatibility with the real world was the major task to be developed by MEMS industry.

Despite of this problems, MEMS technology importance in the electronic marketplace is increasing rapidly, particularly during the last years. Figure 1-1 shows the trend of the market for this kind of technology.

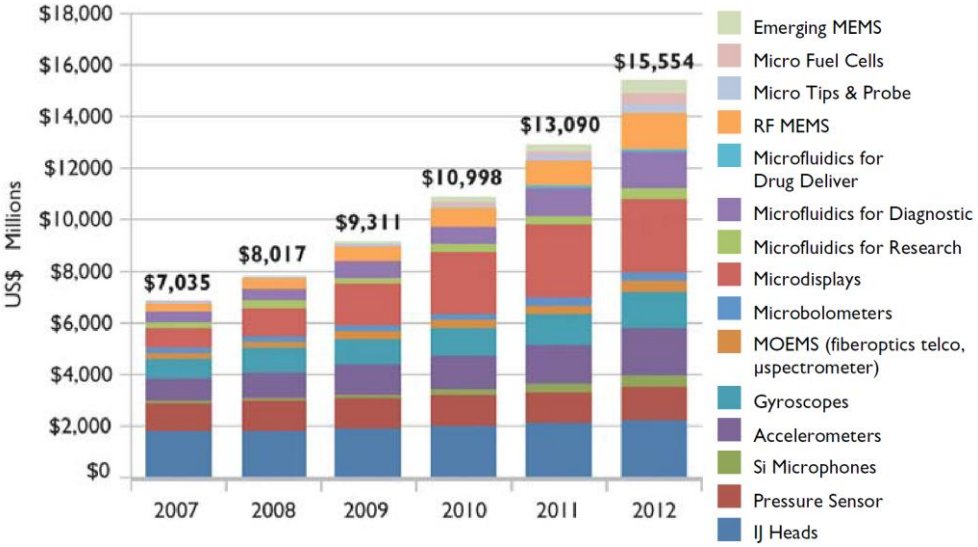


Figure 1-1. *MEMS market forecast [1]*

The sector will likely see a healthy increase this year despite any general semiconductor slowdown, and will remain on track to maintain its 17% average growth for the next years. This charge in volume production is owing to the wide spreading of these devices also in consumer market. By 2012, MEMS makers will be shipping 8.1 billion units a year, worth some $15.5 billion, and nearly half that market will be consumer devices. Major market drivers will include silicon microphones, micro displays, RF MEMS and even microfluidics for biomedical applications. RF MEMS and silicon microphones alone will account for more than 45% of unit demand from 2011.

1.1.1 Applications

With the consolidation of the MEMS technology, it spread in various fields, becoming the outstanding state of the art for a broad range of mechanical, chemical, optical, and biotech products (sensors, microstructures and actuators) fabricated as integrated circuits on silicon wafers in a batch mode. Commercial MEMS products comprise flow sensors, pressure sensors, acceleration sensors, gyros, ink-jet nozzles, head locators in hard drives, and digital light processors (DLPs) in projectors and screen sets. In the following an overview of the major field of applications for MEMS sensor is presented.

Life science applications
The requirement of reliable and very small implantable devices, and the need of handheld devices for patient monitoring and diagnostics at home brought to the utilize of this technology in medicine applications, creating a separated branch of devices called bioMEMS. Biocompatibility, reliability, miniaturization, low power are some of the advantages of this technology. Moreover, bioMEMS feature non invasive/painful procedures and allow to reduce the patient recovery time because they permit to monitor continuously the patient also outside the hospital. One of the most important segment for medical applications of MEMS is the so called Lab-on-a-chip (LOC) which is a typology of devices that integrate multiple laboratory functions on a single small device and that are able to manage extremely small fluid volumes.

Communications
Radiofrequency and microwave MEMS applications are introducing a important surplus value in wireless communications. Low loss, high Q-factor, high linearity and a good power handling are all the MEMS features that make this technology more suitable for this kind of applications with respect to traditional implementations. The use of MEMS in this field permits to produce wireless handsets, base stations and satellites with the key properties of low-power consumption, low area occupation and enhanced re-configurability, which can result in superior functionalities and performances.

Automotive
Electronic technologies in the automotive industry have been introduced starting from the 1960s, gradually replacing mechanical systems and hydraulic actuators. Today, high-end vehicles feature up to 100 different sensors among which about 30 these are now MEMS, and the percentage is forecasted to grow. Some of the main sensing applications in the automotive sector concern the active safety (ABS, ESP, TCS, and so on), intelligent light positioning, intelligent airbags, vehicle monitoring, satellite navigation systems and enhanced antitheft systems. To implement these automatic controls, gyroscopes and accelerometers are the most used MEMS devices.

Commercial applications
MEMS application are well established in specialized markets, primarily automotive, industrial and medical, and the conditions are mature for impacting the consumer market for a wide spectrum of applications. Some of these are ink-jet print heads, silicon microphones, hard disk drive free fall protection, gaming interfaces and digital still camera image stabilization. The consumer market is looking for tiny, low cost, low-power consumption devices. Consumer devices are all battery-operated and are becoming smaller and thinner. Furthermore, the product life cycle of consumer devices is shorter than the one in automotive markets; therefore, this make MEMS system suppliers to face an increasing reduction in time to market for developing new products.

Industrial applications
Requirements for safety in workplaces requires ah high accuracy of the sensing solution used to detect the dangerous operating conditions. MEMS technology supply these requirements, so also in industrial applications like safety systems, process control, equipment position, impact detection and environmental monitoring this technology has been widely used in the later years. Recently, another industrial segment grown significantly: the so called micro-drives. These micromechatronic systems consist of elements capable of processing information and energy and can be employed for fine refinements and advanced operations in which extreme accuracy is required. Moreover, MEMS devices are also used in instruments for synthesizing and analyzing (bio)chemical materials of higher quality, at higher throughputs and against reduced costs. Finally, MEMS devices can be effectively employed for industrial robots, as the technology can apply to tactile sensors, navigation, or proximity sensors.

Military and aerospace applications
Military and aerospace fields were the first major users of the MEMS technology. MEMS technology has enabled smaller, low power, and low cost micro-instruments currently including pressure, flow rate, acceleration, angular velocity, and MEMS actuators such as valves, synthetic jets, boundary layer trip devices, micro jet engines and microthrusters that have supported a variety of aerospace missions. Some of the major military and aerospace applications are microsatellites, micro-propulsion systems, navigation assistance and equipment monitoring,

Optical applications
A very important branch of MEMS family are the micro-opto-electro-mechanical systems (MOEMS). They were used for the first time in 1990 for fiber optic telecommunication applications, but, after a brief crisis, they started to be used in other kind of applications. The most important fields were MOEMS are employed are Digital Micromirror Devices (DMD) for the projection of images, spectrometers used to measure properties of light over a specific portion of the electromagnetic spectrum, barcode scanners, Infrared Focal Plane Array (IRFPA) for the image sensing, and picoprojectors.

1.1.2 Design and testing of a MEMS based devices

The design of a MEMS device is a very challenging activity, because requires a multidisciplinary approach due to the heterogeneity of its components. In fact, these components have mechanical, electrical, thermal, electrostatic, and fluidic interactions. This complexity requires structured design techniques to manage each activity involved in the design of this kind of devices. Although the design of the MEMS sensor(s) and of the sensor interface integrated circuit are crucial tasks for the success of the device, testing activities, that includes prototyping, characterization and calibration play a key role for the production of a winning product.
Figure 1-2 shows the fundamental steps that characterize the life-cycle of a MEMS based device. The design starts from an idea, arisen from market analysis, needs, lacks, and so on, an high level description is defined. It can include a block diagram of the components the device suppose to have and the interaction among them,

focusing on the target application the device will be designed to. Once the requirements are defined, both the MEMS and the sensor interface are developed. These steps are characterized by different activities that includes both design and testing tasks. In fact, the MEMS component, once designed, must be simulated and prototyped before starting to produce it. Likewise, the sensor interface, during the design and once the design is accomplished, must be simulated to verify its correctness. These two steps are strictly coupled, because the design of the sensor interface requires knowledge about the structure of the MEMS, particularly during the MEMS system modeling level (see paragraph 1.2.1). When both the sensor interface and the MEMS component(s) are developed, these components are packaged together. The type of packaging process depends on the adopted approach for the implementation of the microsensor system (see paragraph 1.3.1).

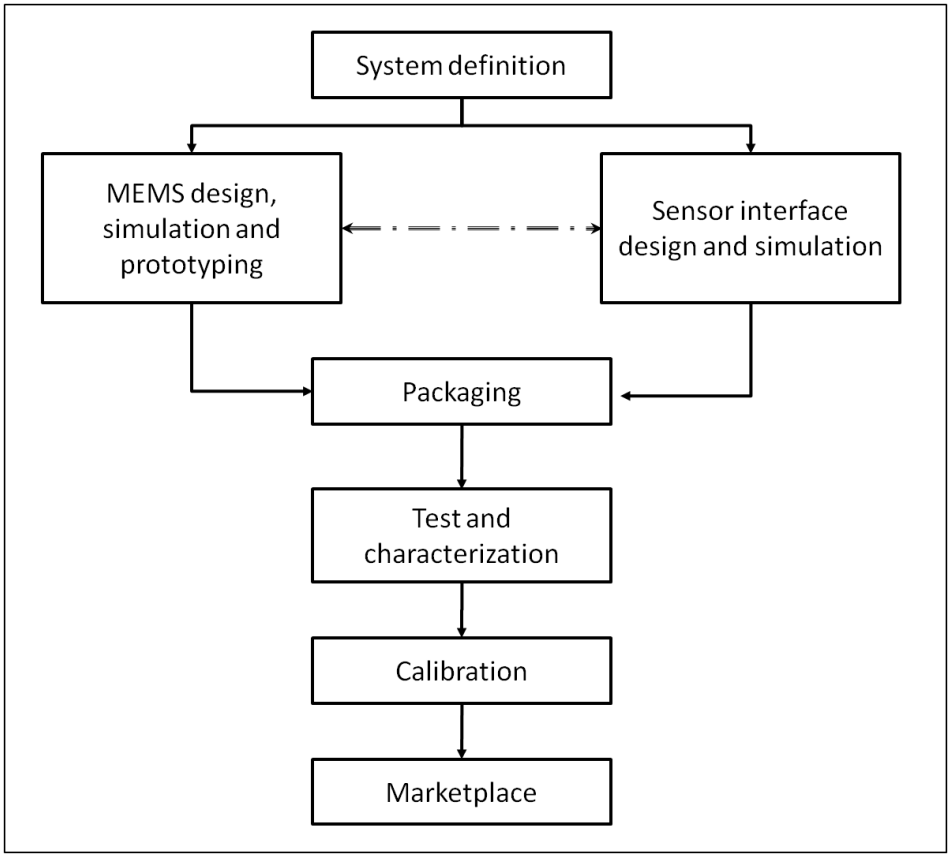


Figure 1-2, *Flowchart of the life cycle of a MEMS system*

The following step consists on the testing of all the functionalities of the device and its characterization. For a sake of simplicity in the representation of the flowchart

the characterization has been inserted after the packaging. Actually, the characterization involves different stages of the life cycle of the device. The purpose of the characterization is to verify that the design is correct and the device will meet all specifications, execute a so called test program to control an automatic test machine (see paragraph 1.4).
The last step before the production is the calibration. It consists on a set of measurements of the characteristic parameters when some appropriate stimuli are applied on the device and, on the basis of the measured value with respect to the data produced by a reference measurement unit, the characteristic transfer function is corrected in order to obtain a certain accuracy and under certain conditions.
The final step is the delivery to the customers. Also this step can be considered a testing activity, because the customer itself will test the device.
The next paragraphs describes all this steps in detail, focusing particularly on testing operation connected to the described activity.

1.2 Design of MEMS sensors

The design of MEMS sensors, as any complex engineering process, requires the definition of a formal design flow, in order to provide a structured and manageable design environment, to permit to spread a technology over a large audience and to formalize a detailed interface of each step and the interaction among them.
A good definition of a design flow allows to enable complex engineering design in the shortest time and through few fabrication iterations. For the definition of a MEMS sensor design flow the first approach was to embed the required design tools into a traditional analogical VLSI design flow. This solution, however, does not address the significant differences between the two technologies. VLSI design flow, in fact, almost exclusively involve a single physical domain: the electric domain, whereas MEMS design requires multiple domains. Moreover, while in VLSI modeling the layout is generated after modeling the functionality of the circuit, in MEMS design the layout itself (intended as geometry, orientation and position of the basic elements) is an essential component for the description of the functionality of a MEMS. For these reasons, the design flow for MEMS sensors requires an ad-hoc design flow.
The MEMS design flow block diagram is shown in Figure 1-3. Starting from the combination of the idea and a feasibility study based on a general architecture, the MEMS design flow begins with the product definition. During this phase, the device specifications, intended as sensitivity, noise, temperature stability and die size are defined. Costs are evaluated in this phase too. The next step consists on the developing of a Finite Element Method (FEM) model to define the MEMS geometry. This phase permits also to extract the characteristic device parameters by simulation, and this parameters are used to develop a high level MEMS model, used to start designing the sensor interface circuit (see paragraph 1.3). Once the MEMS geometry is completely defined and respects the device specifications, the device layout must be generated. To do so, usually the same tools used for the electronics layout are used. Before the production step is very important to perform the layout verification in order to discover and correct errors. This step consists of two actions: the verification of the match between the process specifications and

the layout realization through a Design Rule Check (DRC) and the verification of the compliance between the realized layout and the sensor schematic through a Layout Versus Schematic (LVS) check. Another optional step before production is the layout simulation (also called post layout simulations). This phase permits to verify the design correctness, and to tune the mechanical device properties used in the system level description to enhance the MEMS high level model. Moreover, simulating the geometry allows to find errors that are missed in previous models due to high approximations.

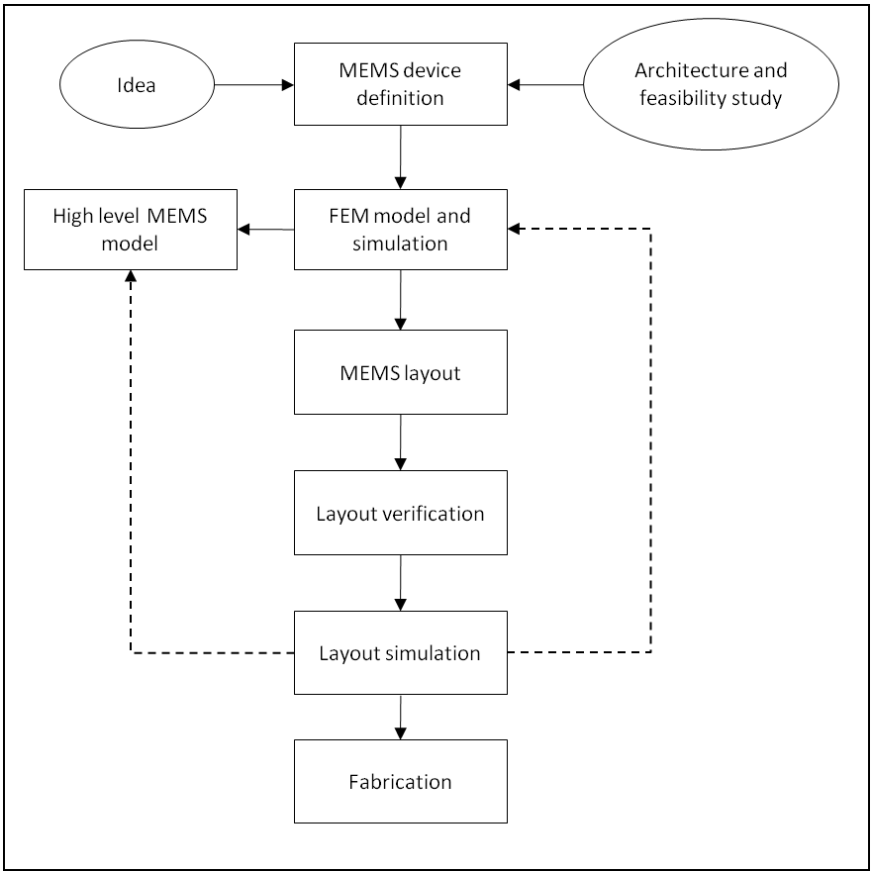


Figure 1-3. *MEMS design flow block diagram*

1.2.1 MEMS modeling

Modeling and analysis of devices and systems are complex subjects. Modeling occurs at many levels and uses a variety of modeling paradigms. Four modeling levels can be identified: System, Device, Physical and Process. All the levels are strictly coupled among them.

The top level is the system, in which the MEMS is modeled together with the sensor interface. In this level either of block diagram descriptions or lumped-element circuit models can be used. Both lead to a coupled set of ordinary differential equations (ODE) to describe the dynamical behavior of the system. Often these equations are written in a specific form as a coupled set of first-order ordinary differential equations for the state variable of the system. In this form the equations are referred as state equations for the system. The tool usually employed for system level models is Simulink™.
The bottom level is the process, in which the process sequence and photomask designs for device manufacturing are created. Process modeling at this level is a highly sophisticated numerical activity, so a number of commercial CAD tools have been developed: they are generically referred as technology CAD or TCAD. These CAD tools allow to predict device geometry from the mask and process sequence. However, the role of the designer is fundamental during this phase, because the correct material properties must be selected, and these properties depend on the detailed process sequence.
The physical level models the behavior of the real device in three dimensional continuum. The governing equations are partial differential equations (PDE), and various analytical models can be used to find closed-form solutions in ideal geometries; however, to model a realistic device it is necessary to use either approximate analytical solutions to the PDE's or highly meshed numerical solutions. Different numerical modeling tools permits to simulate at the physical level, using either finite-element, boundary-element or finite-difference methods. However, to model electrostatic forces, mechanical behavior, coupled electro mechanics and damping effects commercial finite-element and boundary-element tools (for instance, Ansys™ and Comsol Multiphysics™) are necessary.
The device level is used to create a simplified model able to interact with the entire device and their associated circuitry. In fact, while the physical model is adapt to simulate the physical behavior, to interact with the system level without introducing excessive overload it is necessary to reduce a reduced order model. This model must be developed in a form that permits rapid calculation and insertion in a system level simulator. However, it must be energetically correct, conserving and dissipating energy when it should, have the correct dependence on material properties and device geometry, represent both static and dynamic device behavior for both small (linear) and large (non linear) amplitude excitation, and agree with the results of the 3D simulation of the physical level and with the results of experiments on suitable test structure.
There is no order in the developing of the different modeling levels: the designer can create models directly at the system level, or starting from one of the other levels. For example, the designer could start from a physical device description with all the device dimensions and the material properties and then use physical simulations tools to calculate device behavior, capture this behavior in a reduced order model and finally insert it into a system-level block diagram. Or alternatively the designer could simply use a parameterized reduced order model to represent a particular device and defer until later the specification on device dimensions to achieve the desired performance.
In conclusion, the operation of MEMS modeling is a fundamental step in MEMS design, because it permits to analyze the behavior of the sensor the designer is

going to build, evaluating also the interaction with the system it will be part of. This modeling phase allows to develop a mathematical representation of the MEMS sensor before building experimental prototypes, avoiding the costs of a wrong actual prototype. For this reason, modeling is also called “numerical prototype”.

1.2.2 MEMS layout

The MEMS layout is the last step before the device production. A MEMS process is based on the different steps of deposition, pattern and etching. A schematic representation is shown in Figure 1-4.

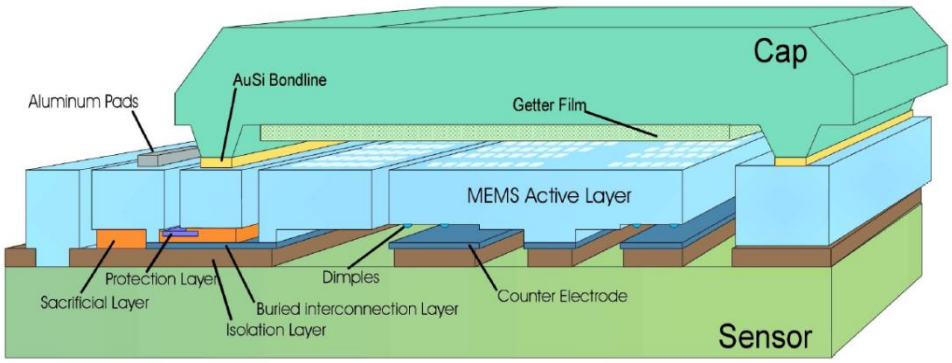


Figure 1-4. *Schematic representation of a MEMS physical process*

A structural layer of polysilicon (called epi poly and characterized by typical thickness of 11 um) is used to build the moving structure of the MEMS (light blue in figure). This moving structure is suspended over an etched pit and is anchored to the substrate with well defined anchor points. The position of these anchors define the MEMS motion which can be characterized by a rotating motion or/and a tilting motion. If the device is a capacitive structure, as the device shown in figure above, electrodes positioned under the moving structure are used to detect the motion through the capacitance variation between the moving structure and the electrodes themselves. The electrodes are made up of thin polysilicon (typical 100nm thick) defined as buried poly (blue layer in figure). The different signals are applied to the different mechanical structures by using appropriate connections between the mechanical parts and the MEMS pads. These connections are realized with both epi poly and buried poly paths which are vertically connected through “vias”.
Since all MEMS processes are based on the same phases, it is possible to abstract them in a layered structure, where each layer represents a process step. Each process can be defined in terms of the manufacturing steps needed for each layer together with the characteristic properties of the material used and the geometrical dimensions of the layers. In the MEMS design methodology this information is

captured in a layout technology file and a layout design rules file which customize an appropriate layout tool. The tools used for this purpose were not originally intended for MEMS (e.g. Virtuoso tool of Cadence™ environment), so MEMS layout designers still face some common issues and roadblocks while attempting to create lithographic masks that correspond with the original device design, specifically in the areas of drawing, design rule checking (DRC) and output. In fact, these tools provide only rectangles as typical geometries, whereas MEMS structures involves arbitrary geometries like arcs, curves, and so on. In addition, in MEMS layout understanding the three-dimensionality of the topography is required. Moreover, while typical transistor blocks may cover 20x20 micrometers areas, MEMS geometries may have 5 micrometers features, and an overall dimension of 1mm. This wide range in size can result in constant zooming in and out during the design process. Thus, some MEMS designs require the ability to snap to corners, midpoints or user specified relative distances without zooming in. Finally, also the MEMS process requires DRC (Design rules check) and LVS (layout versus schematic check) to find errors before tape-out, but the same tools used for IC cannot be used for MEMS process because of the free-form nature of that, which requires to vary design rules depending on the MEMS fab and associated tooling, and many DRC tools are not able to perform operations on all-angle polygon geometries. So, the implementation of these tools also for MEMS is required to have a more reliable design flow.

1.3 Design of the conditioning system

The creation of the MEMS sensor is only the first phase of the develop of a sensor chip. In fact, the information produced by the MEMS must be converted into a form that may be easily manipulated by every components that could interact with it. So, it is necessary to design an electronic circuit able to do it, called sensor interface.
The functions implemented by a sensor interface can range from simple amplification or filtering to A/D conversion, calibration, digital signal processing, interfacing with other electronic devices or displays, and data transmission (that can be handled by wired or, recently, wireless connection). Since the appearance of the first integrated circuit (IC), Very-large-scale integration (VLSI) technologies has been used to make sensor interface circuits, combining discrete sensors with application-specific integrate circuits (ASICs) or circuits on a printed circuit boards (PCB). With the birth of miniaturized sensors (microsensors), realized using IC technologies and materials, it has been possible to integrate the interface circuit and several sensors on the same chip or in the same packages, reducing the cost of measurement systems, the size of the whole system and its reliability.
However, the design of the sensor interface for microsensor systems is very complex for different reasons. This kind of systems, indeed, show worse performance due to weak signals and to offset and nonlinear transfer characteristics. Moreover, most of the traditional circuit techniques cannot be used because they rely on accurate component matching and complex analog functions that will make the device weak with respect to aging and degradation. In addition, it requires sensor specific design techniques to increase accuracy and to reduce the power consumption.

For these reasons, microsensor interface features are strictly coupled with the microsensor the interface must control and the package, because they depend heavily on the quantity to be measured, the physical effect used, the system architecture and the application.

1.3.1 Microsystem vs. Micromodule

There are two possible approaches for implementing microsensor systems: the microsystem approach and the micromodule approach [4]. In the microsystem approach, the sensor and the interface circuitry are integrated on the same chip, whereas, in the micromodule approach, the sensor and the interface circuitry are integrated on different chips, and they are included in the same package (SoP) or mounted on the same substrate.

Both the approaches have advantages and disadvantages. In the first approach, the microsensor must be designed taking into account the material characteristics given by the standard IC process used and any specific processing step has to be performed after the completion of the standard IC fabrication flow, limiting the possibility to improve the sensor performances. Moreover, there are also disadvantages about the cost and the yield. In fact, when the feature size of the technology reduces, while the cost per unit area of the IC is compensated by the area reduction, this is not true for the microsensor, because its size does not depend only on the technology, but also on physical constraints. In addition, the yield for a microsensor is generally lower than for ICs; so, having both elements in the same chip, force to discard a chip when a defect on the microsensor is discovered, even if the IC is working. However, this approach also has a lot of advantages. In fact, the system assembly is simple, inexpensive, and independent of the number of interconnection needed. Moreover, the parasitic due to the interconnections between the sensors and the interface are minimized, and also other parasitic effects are compensated by the use of the same technology that allows to achieve good matching between elements of the two components of the system.

In the second approach, it is possible to use different technology for the sensors and the interface, allowing to choose expensive submicron technologies for the IC and low cost technology for the sensor. Using two different materials, it is possible to optimize the performance separately, reducing the cost and increasing the yield. This approach, however, has a number of drawback. First of all, the assembling is quite expensive and it is a source of possible failures; moreover, the number of interconnections allowed between the sensors and the IC is limited, and these interconnections produces high and unpredictable parasitic. Finally, matching between elements of the sensor and elements of the interface circuitry cannot be guaranteed.

1.3.2 Functions of a microsensor system

As we already said in the introduction, the components of a sensor interface strictly depend on the quantity to be measured, the physical effect used, the system architecture and the application. So, it is more suitable to talk about functions instead of components [5, 8].

Figure 1-5 reports the set of function that constitutes the typical structure of a sensor interface. Each element represented in the figure can be implemented as

single component or subsystems or can be distributed in different components or sub-components. Moreover, some parts can be implemented also outside the sensor interface by the application that uses the system. The decisions as to which functions are implemented where form an important discussion in modern sensor systems design, and depend strictly on the designer choices and the target application. The scheme in figure, however, is not constraining, so some functions can miss and other functionalities can be implemented in the system. For example, while the forward chain is mandatory to extract the information from the MEMS sensor, not all the systems implement the feedback chain that is used only in the "closed loop" sensor category.

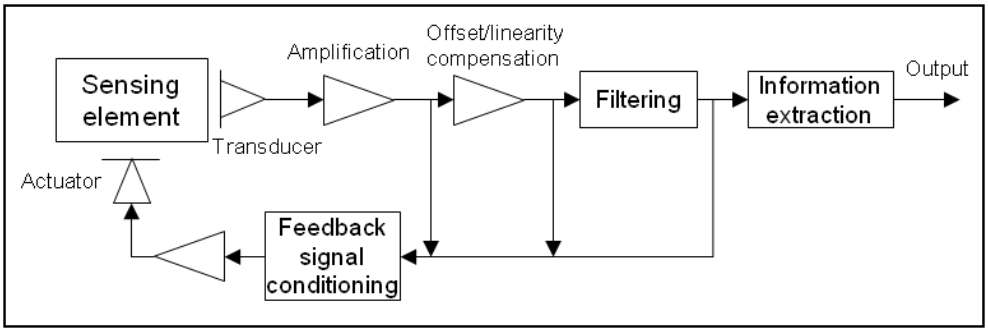


Figure 1-5. *Functions of a sensor system*

A generic sensor system can be described starting from the sensing element, that can be defined as a component that receives the energy from the physical entity being sensed. This form of energy is converted to electrical energy by the transducer. As we already said in the introduction, microsensors produce weak signals, so it is necessary to increase the amplitude of the signals. The role of the amplification function is to translate an electrical signal to one of a different amplitude, that may be expressed as a voltage, a current or a charge. The amplification factor is the gain. Also the conversion between two signal forms (for instance from current to voltage) and the level translation within digital systems can be considered as part of the amplification function. The Offset/linearity compensation is the process of suppress the offset and nonlinearity, which can be originated from imperfection or the characteristic of the sensor itself, from the sensor characteristic. More details on the compensation process are described in paragraph 1.5. The filtering function identifies all the operations that acts on a AC signal to modify its phase or frequency response. So, other than traditional filter such as low-pass filters, it includes also processes such as time integration and differentiation. The last function of the forward chain is the information extraction. It represents the activity of conversion from the data provided by the sensor interface to information. Different action can be part of this category: in fact, the conversion of sensor data in a wired communication protocol is a part of it, but also the manipulation and the processing of the data to produce other kind of information

can fall within it (see Chapter 4). So, all the operation concerning the processing of the data produced by the system can be considered as a part of this function, and can be placed in different locations of the processing chain (inside the chip, in a PC outside the chip, in a Field Programmable Gate Array (FPGA), and so on).
The last two functions regards only a subset of the available typology of sensors. These kind of sensors use a “closed loop” topology in order to control its condition on the basis of the output signal. With the Feedback signal conditioning is intended the process of producing the required feedback signal from the forward signal. To achieve this task, all the functions described for the forward chain can be used. The Actuator function is the process of converting the electrical signal provided by the feedback chain in the appropriate form of energy, depending on the sensor requirements. Usually, the form of energy required for the feedback input is the same of the output to the transducer, but it is not mandatory.
From a functional viewpoint it is not possible to describe some of the most used components that characterize a typical sensor interface, because their position and implementation depends on how the designer choose to distribute the functions described above. To make up for this lack, the next paragraph illustrates a typical microsensor interface circuit.

1.3.3 Typical microsensor interface circuit

A typical microsensor interface circuit is composed by a set of components used to implement most of the functionalities described in the previous paragraph. Figure 1-6 shows a block diagram of a generic sensor interface, for a three sensing elements system [4]. This architecture is valid for both microsystem and micromodule sensors.

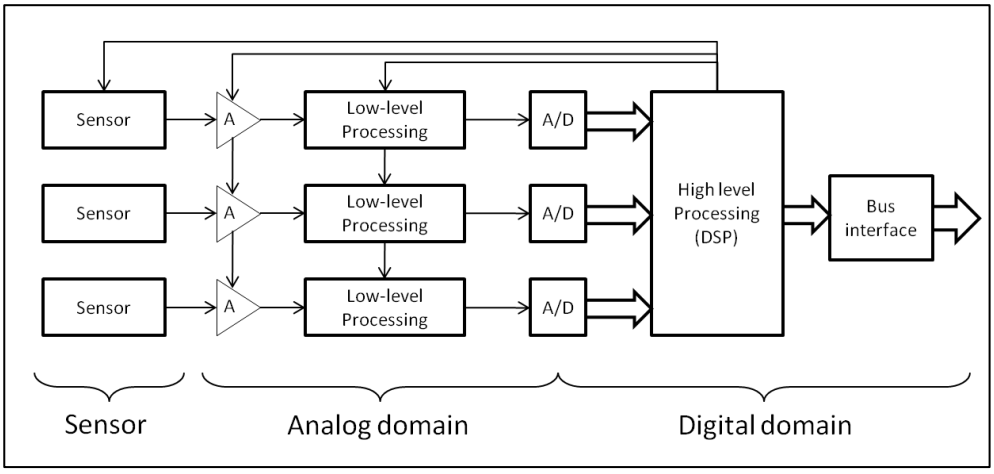


Figure 1-6. *Block diagram of a generic sensor interface*

A microsensor system interface is typically composed by some analog front-end circuit (amplification and low-level processing), one or more analog-to-digital (A/D) converters, a digital signal processor, and an output interface. Although processing is performed more efficiently with analog techniques, in the presence of harsh environmental conditions, the trend is to minimize the analog section, moving the A/D converter toward the input and leaving complex processing to the digital section, because signal processing in the digital domain is more robust than in the analog domain thanks to the larger noise margin. So, the design of the A/D converter turn out to be a critical task, because its bandwidth and dynamics range specifications become more severe than in a almost fully analog design. As we can see in Figure 1-6 a feedback chain is reported, although it is not mandatory as we already explained in the previous paragraph. But in this case, using the backward chain is not useful only for “closed loop” sensor category, but it is used also to adjust system parameters in the analog front-end to optimize its performance, depending on the output signal.
The role of the analog front-end is to transform the raw sensor signal into something suitable for the subsequent A/D converter. The functions implemented in the analog front-end are typically limited to amplification and filtering, in order to keep the analog side as small as possible. Since the analog-front end is directly connected to the sensor, its features depend strongly on the kind of sensor considered.
After the analog front-end, one or more A/D converters connect the analog side to the digital side. In order to supply the reduction on the analog domain, it is necessary to design these components accurately. Its design depends hardly from the signal produced by the analog side. For example, consider a sensor providing a maximum output signal of 10 mV on top of an offset voltage of ±100 mV. If we want to resolve 0.1% step by connecting an A/D converter directly to the sensor and performing the offset cancellation in the digital domain, we need 14-bit resolution. But if we implement some sort of offset cancellation in the analog domain in front of the A/D converter, the required resolution drops to 10 bits. So, designing an A/D converter requires to identify the signal we have to “translate” and requires to reach a compromise among its size (that depends on the number of bits), the complexity of the analog front-end, the sampling time and the sampling noise we can tolerate. On the basis of the method used to reduce the quantization error, two families of A/D converter are identifiable: Nyquist rate and oversampled A/D converters. The first family increases the resolution of the quantizer, thus making the step size smaller; the second family, instead, increases the sampling rate above the Nyquist rate. The first family is imperative in high-frequency applications, simply because the use of oversampling would lead to an impractical speed of operation.
Connected to the A/D barrier, there is the core of the signal processing in modern microsensor system: the digital processing. The most important signal processing functions required for sensor applications are filtering, calibration, and control. Filtering is used to limit signal bandwidth and remove out-of-band spurs or to decimate the output signal of oversampled A/D converters. The response of integrated sensors is often nonlinear. In many cases, therefore, interface circuits have to include a calibration section to linearize the transfer characteristic of the sensor, avoiding the undesirable and unpredictable effects due to nonlinear terms.

Moreover, since aging often modifies the response of the sensor during the lifetime of the device, the programmability of the calibration function is also important. Linearization and calibration are typically implemented in the digital domain to exploit the flexibility of digital signal processing. The most common techniques for sensor calibration are based on lookup tables or polynomial correction. Another important function typically implemented digitally in microsensor systems is the control of the system operation. This includes the timing generation, the selection of the mode of operation (for example, acquisition, calibration, transmission, and self-test), and the generation of the feedback signal for adjusting the sensor or analog front-end characteristics.
The last part of the chain is the output interface, whose role is to convert the sensor output signals into a standard communication protocol, keeping the number of wires limited to avoid cost and reliability problems. Serial bus systems are the best candidates to solve these problems, since they require a minimum number of wires and allow simple transmission protocols to be implemented. One of the most used bus is the Philips I^2C bus system that has been specially developed to interconnect integrated circuits. This system allows relatively small distance data transmission through a serial connection using only four lines. The I2C bus is a multimaster bus, since more than one device can initiate and terminate a data transmission. However, to avoid degradation of the message, only one device at a time can be the master. Finally, depending on the application, specific bus interfaces can be used and possibly be compatible with standard computer systems,like the SPI bus and the Ethernet. Another approach to solve the wiring problem is to use wireless interface. The most promising approach for wireless interfaces, especially when short-range interconnections are required, is the Bluetooth standard. Several fully integrated Bluetooth transceivers are available on the market, either as commercial parts or as IP blocks to be included in custom integrated circuits without considerable design effort. Other solutions for wireless interconnections, especially for applications operating over longer ranges, are based on cellular phone standards or on wireless LANs.
Designing of a sensor interface must keep in mind also some other kind of requirements that are outside the implementation of a single component of the system, but regards the design approach of the whole system and the application itself. Some applications, indeed, may require real-time response: in the case of human-like sensing, real-time means that a few milliseconds, while for control or recognition of fast-moving objects such as cars or planes, real-time can imply several megahertz of bandwidth. Electronic equipment is becoming more and more portable, leading to battery-operated sensor systems with a small volume and weight. These features imply microsensor technologies, special packaging and assembling, low-voltage and low-power design methodologies, robustness, and shock resistance.

1.4 Testing and characterization of the SoP

Testing activity involves several different figures, depending on the phase of the life-cycle of a chip. When a new chip is designed and fabricated for the first time, testing should verify correctness of the design and the developing of the test procedures. During this first phase of testing, also the designers of the chip are

involved and it may ever take place in the design laboratory rather than in a factory. Based on the result, both the design and the test procedures may be changed. This first phase is called *verification testing*.
A successful verification signals the beginning of production, that means large scale manufacturing. In this phase, called *manufacturing testing*, fabricated chips are tested in the factory.
The last phase of testing of a chip is done by the customer itself to ensure quality. This testing is known as *acceptance testing* and is conducted either by the user or for the user by some independent testing house.
The basic principle of digital testing is the application of binary patterns (called also test vectors) to the input of the circuit and the comparison of the response with the expected one, as shown in Figure 1-7. This test is done automatically by the use of *automatic test equipment* (ATE) that is a powerful computer operating under the control of a test program written in a high level language [6, 7].

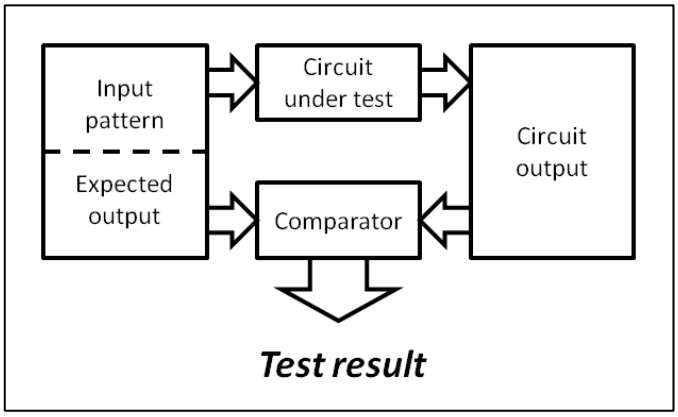


Figure 1-7. *Basic principle of digital testing*

1.4.1 Type of testing

Chip testing can be classified, depending upon the specific purpose it accomplishes, as characterization, production, burn-in and incoming inspection.
The characterization testing is performed on a new design before it is sent to production. The purpose is to verify that the design is correct and the device will meet all specifications. Functional tests are run and comprehensive AC and DC measurements are made. Probing of internal nodes of the chip, commonly not done in production testing, may also be required during characterization. A characterization test determines the exact limits of device operating values, testing the worst case, because devices passing this test will work for any other intermediate conditions. This kind of tests produce a pass/fail decision. Each test is done on a statistically significant sample of devices and it is repeated for every combination of two or more environmental variables. During this kind of testing, design errors are diagnosed and corrected, chip characteristics are measured for

setting final specifications and the test program is developed. Some of the tests done during the characterization phase is repeated also throughout the production life of the device in order to verify if improvements in the design and the process yield can be done. Moreover, characterization can be used to verify the failure causes of pieces discarded during the production test.
The production testing comprehends less tests than characterization because its aim is to determine whether the device meets specifications. So, the vector may not cover all possible functions and data patterns but must have a high coverage of modeled faults. The preparation of the production testing procedures must keep the duration as low as possible, because every devices must be tested. So, each test must determine if a piece passes or not; if a fault diagnosis is necessary, the piece must be examined by the characterization testing. This kind of testing is not repetitive, but its aim is to simply verify if a piece respects all relevant specifications.
Burn-in testing is a special type of testing: its aim is to verify the robustness of a device over a long period of time. Correlation studies show that the occurrence of potential failures can be accelerated at elevated temperatures. Two types of failures are detected by burn-in: infant mortality failures, often caused by a combination of sensitive design and process variation, may be screened out by a short-term burn-in in a normal or slightly accelerated working environment, and freak failures, that require long burn-in time in an accelerated environment. During burn-in, we subject the chips to a combination of production tests, high temperature, and over-voltage power supply.
The incoming inspection testing perform incoming inspection on the purchased devices before integrating them into the system. The test procedures used during this testing phase can be the same of the production, more comprehensive than that or ad-hoc for the target application. It is done for a set of random samples, whose size depends on the quality and the system requirements.
A single test can be classified in one of the following two types: parametric or functional. Parametric tests are those tests that regards electrical failures; for instance short test, open test, leakage test, and so on. These tests are usually technology-dependent. Functional tests, indeed, determine whether the internal digital logic and analog sub-systems in the chip behave as intended. They check for proper operation by testing the internal chip nodes. These tests may be applied at an elevated temperature, at several voltages and at varying timing conditions (e.g. clock frequency) to guarantee specifications.

1.4.2 Automatic test equipment

The tests described in the previous paragraph are executed by a tester, whose purpose is to drive the inputs and to monitor the output of a device under test (DUT); testers are also called ATE (automatic test equipment). So, the ATE is an instrument used to apply test patterns to a DUT, analyze the responses from the DUT and mark the DUT as passed or not. These machines are controlled by a workstation or a PC where the test program is executed. The test program contains the set of operation that the ATE must execute to conduct testing. Because its syntax depends on the machine it is used, the tools used to generate patterns (test pattern generator, or TPG) commonly generate a tester-independent program

which can be customized for the used machine. Most of these machines provide a choice of input signal waveforms, permit to mask output signals, are able to sense impedance state and offers a variety of sophisticated capabilities.
The tester has one or more test heads, which contain buffering electronics local to the DUT, but one mainframe with common instrumentation. To automate the loading of DUT(s) in the test head(s), usually the ATE is connected to a handler. Thus, while one chip is being tested in one test head, another chip can be loaded into a second test head, so the tester overlaps mechanical handling of parts with electrical testing of parts. Some ATEs can be designed to test several devices at the same time: this testing approach is called multi-site testing. In multi-site testing, single test heads have been designed to handle multiple packages simultaneously, so it contains more than one socket. Using this approach it is possible to test multiple DUT at the same time with a reduced cost, because most ATE instruments can be replicated in the tester and most operating systems allow to execute the program designed for one site in multiple sites using duplicate resources in the tester, simply modifying it to address different resources. The limits of this approach are the number of instruments installed in the ATE to handle all of the required pins and the available type of handling equipment.

1.4.3 Testing of sensor systems

Testing of MEMS sensor systems are quite different from testing of CMOS devices and packages. Indeed, the techniques used for traditional devices cannot test specific MEMS-related issues such as moving parts, temperature, humidity, pressure, and so on. Special chambers, probes, sample holders, test structures, detection systems, sample preparation techniques and electronics are required. Some points must be considered when testing MEMS systems: first of all, the tester must be able to test influences such as temperature, high-G vibrations, pressure and vacuum, because the environment at the sensor ring is often very harsh. Second, the device often has openings through which the medium carrying the sensor/actuator signals is exposed directly to the microsystem chip inside the package. This means that the device is exposed to unwanted environmental influences and must be tested in a completely shielded environment such as a vacuum chamber. Moreover, the packaging techniques used for microsystems devices are normally device-specific or application-specific and tend to have failure mechanisms, which differ substantially from those of other electronic components and systems. This requires a high level of equipment customization. In addition, many MEMS are used in safety-critical applications where long-term reliability is critical. Finally, substrate handling is often difficult because common pick and place systems and pin drives could damage the micromechanical parts.
However, different companies already provides ATE machines able to manage testing of MEMS systems.

1.5 Calibration and performances evaluation

One of the most important phase of the design of a sensor system is the calibration and performance evaluation. Obtaining the maximum in terms of performance, intended as minimum error in the measure with respect to the physical variable, resolution, sensitivity, and so on, at the lowest possible cost is a challenging

activity. In order to accommodate the effects of interfering and modifying inputs, non-ideal sensing devices, process variations and time variations it is necessary to modify the system design or to add some new elements to it. Filters on the conditioning chain, negative feedback and compensation in one or both the analog or digital domains are some of the processing functions applied to the sensor read-out with the scope of improving sensor signal quality, following the process of calibration or measurement of the sensor's characteristics. The easiness in the adoption of these techniques in MEMS sensor, with respect to the old approach that uses discrete sensors, is one of the reasons of the success of this technology.

1.5.1 Performance parameters and sensor errors

To achieve the goal of obtaining a “good” sensor some quality parameters must be defined. A list of some of the most important parameters is reported in Table I [8]. Obviously it is impossible to obtain a sensor system that satisfies the optimum for all of these parameters; a trade-off among them must be reached, depending on the application, the cost and the design constraints.

Table I – Performance parameters

Quality parameter	**Definition**	**Ideal value**
Full scale output (FSO)	Algebraic difference between upper and lower endpoints of output	Whatever is required by the downstream electronics, usually in the volt range, for voltage signals
Error	Difference between measured physical variable and true value of the physical variable (usually expressed in percent of FSO)	0%
Offset	Sensor output for zero applied input	0 (assuming zero referenced voltage output, although some systems, for instance 4-20 mA voltage loop would require an offset of exactly 4 mA)
Hysteresis	Maximum difference in the sensor output when the value is approached first with increasing input and second with decreasing input, expressed in percent of FSO	0%

Quality parameter	Definition	Ideal value
Linearity	Closeness of calibration curve to a specified straight line (usually measured as the maximum deviation of calibration point from straight line as percent of FSO)	0% error
Sensitivity	Magnitude of change in the sensor output with respect to change in the physical variable	Whatever is required to allow measurement of the minimum input required to be detected
Accuracy	Ratio of error to FSO expressed in percent	0% error
Repeatability	Agreement between independent measurements made under identical conditions (maximum difference in output readings given as % of FSO)	0% difference
Resolution	Smallest change in the physical variable that results in a detectable change in the sensor output	Infinitesimal
Frequency response	Change with frequency of output/input magnitude ratio and phase difference for sinusoidal varying input	Flat to infinity
Cross-sensitivity	Sensitivity of sensor to another variable than the physical quantity under measurement	0%
Stability	Ability of sensor to reproduce output for identical input and condition over time (expressed in percent of FSO)	0% error

These parameters are also valid to characterize the sensing element itself (see paragraph 1.2) and, in this case, they are useful during the design of the sensor

interface to develop the conditioning system (see paragraph 1.3). In this phase, however, they are used to evaluate the performance of the whole system to obtain information about its characteristics and select the best pieces during the production phase.
The set of errors the sensor system may fall can be classified in two categories: systematic errors and random errors. The systematic errors category such as inaccuracy of system parameters and parasitic effects, streaming from the sensor design, its fabrication processes and/or the read out electronics groups all the errors that are deterministic, and in most cases measurable and sensor type specific. Systematic error batch calibration or compensation is generally applied to alleviate such undesirable effects if the errors are large enough as to take the sensor's accuracy outside the desired range. An automated calibration process could then be performed by placing many sensors in a controlled environment and, via a bus, measuring each sensor output and programming the integrated calibration function. The random errors category, instead, contains all the errors that arise either from random variation in the production process or from the environment. These kind of errors are device dependents, so it is possible to compensate them through individual calibration.
The most common errors are offset, gain, range or full-scale error, non-linearity, cross-sensitivity, hysteresis and drift. Whilst hysteresis and drift are common in some sensor types, all other errors are present to a higher or lesser degree in all sensor types. These are, moreover, very difficult to compensate, even if some techniques to compensate these kind of errors are available in literature. For the other typology, starting from the assumption that all the error can be approximated to a linear function, can be corrected during the calibration phase. In fact, usually sensor systems are equipped with programmable offsets and gains, and the integration of complex signal processing elements permits to compensate this kind of errors also applying complex techniques; in some cases, it is possible to apply also non-linear correction techniques.

1.5.2 Sensor calibration

ISO defines the calibration as “the set of operation which establish, under special conditions, the relationship between values indicated by a measuring instrument or a measuring system, or values, represented by a material measure, and the corresponding known value of a measurand”. This definition identifies the calibration process as a measurement activity. The calibration, however, can include also the correction of the sensor characteristics; so, it can be defined as the procedure of correcting the transfer function of a sensor, using a reference measurement system in order to guarantee a specified input-output relationship with a certain accuracy and under certain conditions. In practical, to calibrate a sensor it is necessary a reference sensor and, on the basis of the comparison between the measure obtained under some stimuli (the source of the stimulus depends on the physical quantity measured by the sensor itself) by the reference with the one obtained by the sensor under calibration, it is possible to determine the error with respect to the desired sensor transfer characteristic. After this information is extracted, adjustment or compensation of the sensor characteristic can be applied in order to obtain a more accurate transfer function. During the calibration phase, however, other useful properties can be extracted; complex

systems, indeed, requires to measure different system parameters to evaluate the optimal setup of the conditioning system. The number of measurements required mostly depends on two factors: the way in which the sensor user will be accounting for the sensor non-ideal behavior and the time and cost investment in the process of calibration, which reflects in the sensor cost. In fact, the process of calibration requires time to be executed, because measurements throughout their range have to be taken. If a measure must be done for each device, the calibration can take a lot of time, increasing the cost of the MEMS sensor. In devices with a slow response time, the cost increases even further.
After a calibration session is terminated, the result of a calibration is stored in a document or a database. These data are used to verify if the measurements fall between set limits: the sensors which do not calibrate within the set limits are simply rejected. So, also the cost of the discarded piece and the cost of their calibration in terms of time are virtually added to the cost of the successful sensors, when assessing the overall cost of the sensing application. To reduce the calibration's cost, a first optimization to the process is to calibrate several sensors in the same run; this solution requires a more complex equipment and the calibration in this kind of setups can take more time, but the total time taken is significantly reduced.
The activity discussed in this thesis regards the calibration of inertial sensors (see Chapter 3), so the next paragraphs illustrates in detail the peculiarities of the calibration process for this kind of sensors.

1.5.3 Calibration of inertial sensors

Inertial sensors, differently to other non mechanical sensing devices, require a more complex equipment to perform a calibration session. In fact, mechanical stimuli must be applied to the device in order to compare the measured value with the reference.
To calibrate an inertial sensor, it must be mounted in a mechanical machine able to stimulate the sensor with a proper physical stimulus. Some of these test equipments are: Accelerometer turntables, rate table, vibration and shock machine and centrifuge. The mounting fixture (i.e. the board and the mass where the piece is mounted on the machine) might contain one inertial sensor or contain two or more sensors in order to increase test productivity by testing several sensor simultaneously; simultaneous sensor testing can also help distinguish between noise inherent to the individual sensors and noise from common seismic and other input. But, in this case, Care must be taken to avoid sensor-to-sensor crosstalk, such as mechanical coupling through the mounting fixture, electronic coupling through a common power supply system, or electromagnetic coupling between closely spaced sensors or signal lines. When mechanical measurements are involved, it is necessary to take care of external noise due to vibration of the floor of a building, local cultural activities (such as automobile and other traffic, rotary equipment in the building in which the test station is housed, fans in test equipment, and even people walking by the test station), seismic activity and so on. Moreover, the mounting fixture should be designed to not have any structural resonance near the resonance frequencies of the sensor under test.

Accelerometer turntable

This machine is used to perform various gravity field laboratory tests on an accelerometer. Figure 1-8(A) shows a scheme of a typical accelerometer turntable configuration. As we can see, the accelerometer is placed in the machine so that its sensing axis (IA) is perpendicular to the rotation axis; this is necessary to calibrate bias and scale factor using the local gravity as a reference and IA misalignment using the local vertical. However, to measure cross-sensitivity, it is necessary to align the IA to the rotation axis. Rotation of the machine between IA up and down is used to determine whether there is any shift or transient in an accelerometer's output across a change in input between plus and minus the local value of gravity. Up-down testing is also used to calibrate accelerometer scale factor and bias, with results that are insensitive to misalignments although the results could be aliased by other model parameters, such as nonlinearities, if they are not known from other tests. A set of subsequent up-down movements permits also to measure the stability of scale factor and bias. If the fixture is equipped with slip rings it is possible to measure the angular velocity and acceleration sensitivity by rotating the mounting fixture continuously.

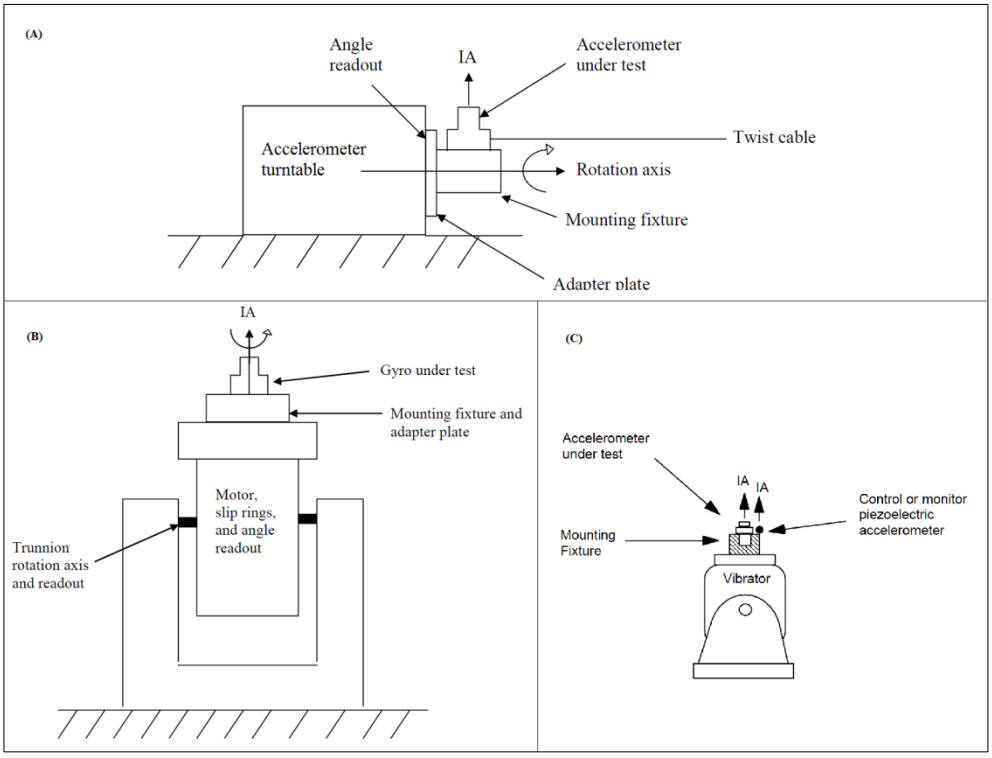


Figure 1-8. *Test machines: (A) Accelerometer turntable; (B) Two-axis rate table; (C) Vibration machine [3]*

Rate table
Various tests can be done on a gyroscope using a rate table: Long-term drift and noise tests are done with the rotary table in a fixed position, such as with IA vertical or with IA parallel or perpendicular to the earth's rotation vector. Continuous rotations of the rate table at different rates and directions about the axis parallel to the gyro IA are used to calibrate gyroscope bias and scale factor, although gyro bias is more accurately calibrated in a drift test, after compensation for sensed earth rotation rate. A gyroscope's dead zone is evaluated as follows: First, place the gyro IA perpendicular to the table rotation axis. Second, rotate the table slowly through the position where its IA is perpendicular to the earth's rotation vector. The dead zone, if any, typically occurs where the bias is cancelled by the input rate. Figure 1-8(B) shows a two-axis rate table. Rate table are usually equipped with slip rings in order to allow continuous rotation. Another technique to avoid the use of slip rings (only for data lines) is to send data by a radio link; for power lines, in any case, slip rings are necessary.

Vibration machine
This machine is used for both accelerometer and gyroscopes in order to perform various laboratory tests (Figure 1-8(C)). A frequency sweep on a electrodynamics vibrator can look for any structural resonances in the sensor that would be revealed by shifts in sensor output at vibration frequencies where there are resonances. Shifts and transients are sought in the sensor output across exposure to vibration and shock levels encountered in applications. Performance through random vibration and shock can also be determined, such as whether the signal processing used for the sensor adequately compensates for any vibration rectification effects. Tumble and rotation rate calibrations before and after exposure to vibration and shock can determine the repeatability of bias, scale factor, and other parameters across vibration and shock. Sine vibrations at various levels and along various sensor axes are used to calibrate sensor model nonlinearities. Sine vibrations along an accelerometer's IA are used to measure the accelerometer's transfer function.

Centrifuge
The use of a centrifuge allows to determine the sensitivity of gyroscopes and accelerometers to high-level sustained acceleration. It usually rotates in the horizontal plane about a vertical rotation axis so that the inertial sensor under test senses an acceleration of $r\omega^2$ along the centrifuge arm and the local value of gravity in the vertical up direction, where r is the arm radius and ω the centrifuge rotation rate. If the centrifuge angular rate exceeds a gyroscope's angular rate capability, then either the gyroscope has to have its IA perpendicular to the centrifuge rotation axis, or the gyroscope must be mounted on a counter-rotating table at the end of the centrifuge arm. Calibration rotation rate and tumble tests before and after the centrifuge exposure can determine whether there are any shifts in bias, scale factor, and other gyroscope or accelerometer model parameters across the centrifuge exposure. One of the main uses of a precision centrifuge is to calibrate accelerometer nonlinear model coefficients.

1.6 The future of MEMS based systems

The MEMS business is moving very fast, but it is still a very fragmented market [2]. Some points has been put in evidence by the CEO of Yole Développement:

- A limited number of applications have a market size above $200M;
- Simplification of manufacturing is still an objective;
- MEMS packaging and software development are more and more adding value: sensor integration (in silicon or in SoP) and sensor fusion are key challenges for the industry;
- The development of new MEMS applications is taking years to be commercialized: In average, 4 years from first developments to first commercial product and $45M of investment.

On the other hand, the diffusion of this kind of technology in consumer electronic application is changing the business.
Several large companies are working on the optimization of their process, but a lot more work must be done to be after the dynamic market of consumer applications. In fact, time-to-market and cost reduction are increasingly assuming a key role in the developing of a MEMS sensor system.
Moreover, the necessity of new functionalities like sensor fusion requires a redesign of the structure of electronic companies by introducing strong software engineering teams, in addition to the set of multidisciplinary entities required for the design of MEMS devices. Figure 1-9 shows the strategies of integration of different typologies of sensors to obtain additional features.

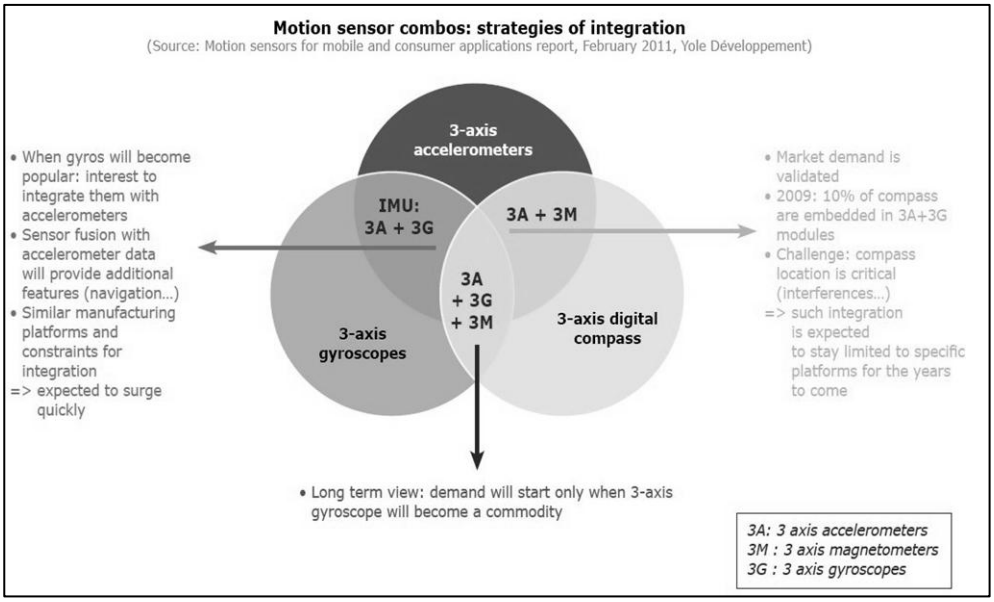


Figure 1-9. *Motion sensor combos: strategies of integration [2]*

As we can see, different combinations of sensors can be adopted to produce new functionalities by fusing their information. In a long term view, all these three types of sensors can be combined together to obtain a stand-alone multi-sensor device able to trace the position of the device itself with a high degree of reliability. Not only the consumer business is interested on these combined devices, but also all part of the industry: medical, defense, telecommunication will all benefit from the high scale production of this combo-sensors.

2 CHARACTERIZATION OF MEMS AND PROTOTYPING OF THE SENSOR INTERFACE

Nowadays MEMS devices are employed in many areas, such as communications, automotive, signal sensing and space technology. The continuous progress in MEMS fabrication technologies led to the development of MOEMS that represent the outstanding state of art for projection technology. The increasing success of these systems stems from their low-power consumption, low manufacturing cost, miniaturization and their capacity for integration with electronic circuits. On the other hand, testing and characterization of MEMS and MOEMS systems, whether in the development or production phase, can be very challenging compared to pure electrical tests due to the intrinsic multi-domain nature of such systems [9]-[14]. Moreover, the design of the sensor interface itself is a very difficult process, because traditionally it is based on a simplified model of the MEMS sensor to simulate the behavior by the use of CAD tools during the developing of the ASIC.
High volume, high-cost, and accurate measuring systems are necessary to characterize and test MEMS and MOEMS, especially to examine motions, deflections and resonance frequencies of the mechanical structures that are the distinguishing characteristics of these systems [15]-[16]. A variety of custom systems relying on interferometry have been developed for deflection measurement but they require a significant amount of development time. Alternatively, there are a variety of commercial deflection measurement systems based on scanning Laser-Doppler vibrometers. However, even though these systems feature very accurate results, they are often very expensive [17]-[22].
This chapter illustrates two architectures to achieve these tasks effectively and efficiently.
In the first paragraph a fast-developing and low-cost characterization and testing environment for MEMS and MOEMS is presented [23]. The environment is based on a platform for sensor interface developing called Intelligent Sensor InterFace (ISIF). ISIF platform can be used as a complete solution for sensor signal conditioning as well as an actuation electronic driver for MEMS and MOEMS [24]-[25]. The test environment can be customized thanks to the large programmability of the ISIF platform (input channels, signal conditioning blocks). All these customizations can be easily achieved using a simple graphical interface developed with LabVIEW^{TM}. The paragraph is organized as follows: the first part describes the ISIF platform, then the test environment and the scanning micromirror used as case study, and how the proposed test and characterization environment has been used to characterize the micromirror are described. Finally, the results of the tests are shown and the conclusions are drawn.
In the second paragraph a bridge solution based on bi-synchronous First-In First-Out (FIFO) structures for frequency conversion, using a custom protocol with priority paths managing is described [26]-[28]. The IP bridge is configurable both during the implementation phase and at run-time execution. The size and number of locations for the FIFOs can be chosen at synthesis time, while the output frequency can be modified at run-time. The on-chip communication between the main CPU and our module can be configured for AMBA AHB or APB protocols, which are de-facto standards in embedded systems [29]. The design internally

requires a protocol change to be able to provide clock frequency conversion and information transmission as fast as possible, using the lowest number of pads. To convert back our custom protocol in a standard one an IP must be foreseen in the external FPGA (in case AHB or APB are used in the FPGA the proposed IP bridge can be replicated). This solution allows the use of an FPGA to design new functionalities that may be included in the final MEMS device. It has been chosen not to integrate the bridge in the ISIF platform, but in a final sensor interface platform based on ARM9 microprocessor, in order to evaluate its performance in the 0.18 µm Bipolar C-MOS D-MOS (BCD8) technology. However, the next step is to integrate it in the ISIF platform in order to provide a complete MEMS prototyping environment. In this paragraph, after a brief analysis of the state of the art, a detailed description of the IP design and architecture is reported. Then, test results and CMOS synthesis data are exposed. Finally, the results of the integration of the IP bridge in a real sensor interfacing platform implemented in BCD8 technology are presented and the conclusions are drawn.

2.1 ISIF: a low-cost and flexible platform for the characterization and the conditioning of MEMS sensors

2.1.1 ISIF platform

The ISIF chip was developed and implemented in a 0.35 µm BCD (Bipolar C-MOS D-MOS) technology. Figure 2-1 shows the ISIF block diagram. The platform provides a set of programmable analog and digital IPs (Intellectual Properties) directly on silicon. The IPs are configurable together with relevant interconnections and interfaces with DSP software routines running on the embedded processor. These routines emulate hardware blocks and/or are used to perform calibrations and compensations on measured data.

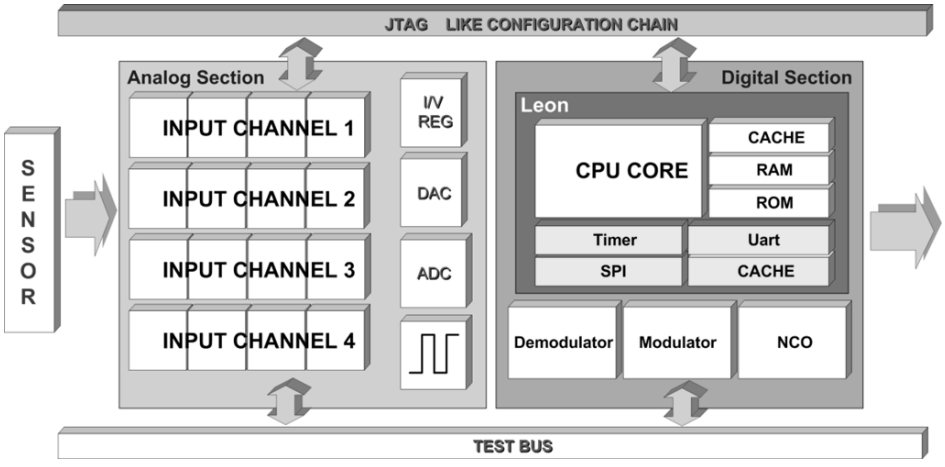


Figure 2-1. *ISIF block diagram*

The ISIF platform features four different input channels (Figure 2-1) that allow the direct interconnection of the system with the sensor. Moreover all the analog and digital IPs are fully programmable by a set of status registers linked together in a serial JTAG-like chain. For these reasons, different interfacing architectures, data paths and signal processing chains can be quickly implemented and evaluated on the field thus allowing a fast and a flexible interfacing, characterization and test of the sensor.

2.1.1.1 Analog section

The analog section of the ISIF is mainly composed by four input channels for signal acquisition and conditioning (Figure 2-1). The input stage (Figure 2-2) is made up of a charge amplifier, which can sense voltage, current, resistance and capacitance, thus covering MEMS needs in terms of signal acquisition and feedback information. Then, the signal is filtered and properly shifted and amplified by the subsequent blocks in order to fit the input dynamics of the Sigma-Delta ADC. The analog section features several DACs that provide the driving signals for the sensor.

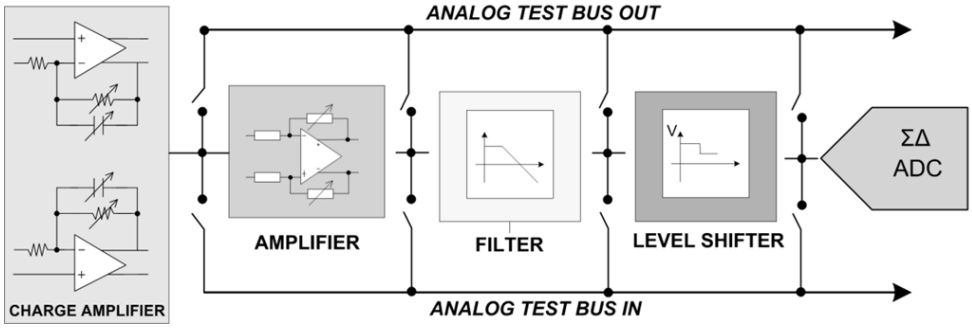


Figure 2-2. *Input channel block diagram*

2.1.1.2 Digital section

The digital section includes several peripherals such as CACHE, ROM, RAM and EEPROM memories and some IPs for digital signal processing. The core of digital section is based on a LEON processor [30]. The LEON is a general purpose processor based on a 32-bit RISC SPARC-V8 compliant architecture which features hardware multiplier and divider, interrupt controller, memories busses and standard peripherals like timers, watchdog, UART (Universal Asynchronous Receiver Transmitter) and SPI (Serial Peripheral Interface). The set of DSP IPs includes a modulator, a demodulator, a 6 DACs controller, a sine wave generator, which can provide up to 16 waves with 3 different frequencies and programmable phases. All these IPs can be accessed at their inputs and outputs via hardware and via software allowing a higher degree of flexibility in the hardware/software partitioning. The CPU can perform some DSP routine as well, like digital filtering. These routines can be easily integrated with the real DSP blocks creating a flexible and an ad-hoc DSP chain for different kinds of MEMS. As an example, a digital

hardware/software PLL (Phase Locked Loop) has been implemented on a fast prototyping board (Figure 2-3).

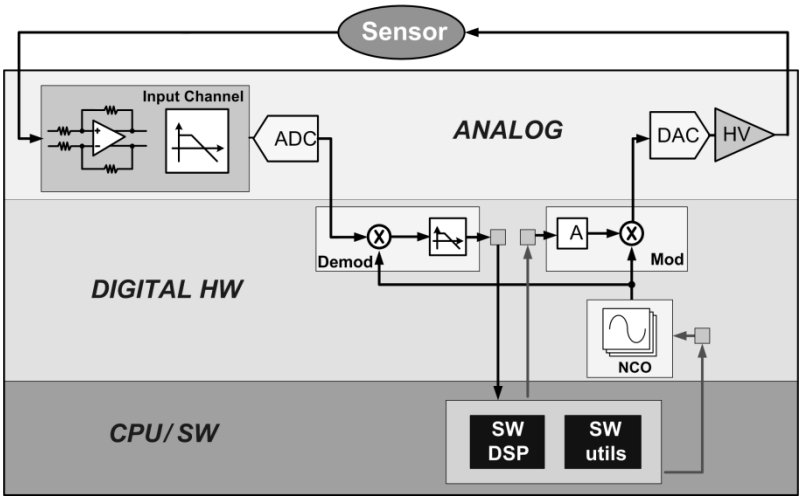


Figure 2-3. *Signal chain*

2.1.1.3 Software

A ROM memory is used to store the boot and some firmware utilities. The dedicated software runs in the RAM and can be downloaded via UART from a PC or stored on an external SPI EEPROM and then loaded by the boot utilities at the start-up via SPI interface. Developed firmware utilities are used: (i) to configure the whole analog front-end, (ii) to change the digital blocks configurations and interconnections, (iii) to perform useful routines like temperature and non-linearities compensation, (iv) to handle the output communication for debug monitoring and data logging for post processing.

Many safety market requirements are pushing towards the use of hardware-only solutions. Unfortunately, these types of solutions are not suitable for the earlier stages of a system development where maximum flexibility is required to explore quickly and inexpensively the wide system design space. Moreover, a hardware-only approach would require too many parameters and bits of configuration for trimming, which are not compatible with our area and power consumption requirements. For all these reasons the ISIF platform features a LEON embedded processor. The LEON provides a library of signal processing software modules, which are designed with the aim of matching real hardware devices so that a future hardware-only implementation (which is necessary for the mentioned reasons) will be costless. The LEON processor offers (i) good signal processing features, (ii) guarantees high flexibility, (iii) the required computational power for these real-time software IPs implementation, and (iv) an easy system updating as well, due to possible system modification and new requirements. It is worth noticing that the aim of the ISIF platform is not achieving the best performances, especially in terms

of area and power consumption, rather is providing an as wide as possible set of IPs (real and/or software emulated) which can offer the largest number of possible solutions for MEMS characterization. In the final ASIC device, software routines can be quickly replaced by corresponding hardware IPs with a low risk and low cost for redesign minimizing time to market as well.

2.1.1.4 High Voltage Board

In order to satisfy the requirements in term of driving of some types of MEMS or MOEMS (like micromirrors), a high voltage analog driving board has been developed. It basically includes high voltage amplification stages, which can shift and/or amplify the output voltage of ISIF DACs (Figure 2-14).

2.1.2 Fast-developing and low-cost characterization and test environment

A fast-developing and low-cost test environment for MEMS and MOEMS has been developed. The approach is based on the use of the ISIF and on the following flow:

- design space exploration;
- preliminary configuration of the ISIF and connection to the MEMS;
- first results, cross-check with expected results, run-time minor changes of the configuration and settings of the IPs;
- tests on final configuration;
- comparison and validation with FEM simulations;
- analysis of data and results.

A test environment has been set up in order to configure the interface system. A LabVIEW™ Graphical User Interface (GUI) has been developed in order to easily configure the ISIF. The GUI is able to apply minor changes to the configuration of all the IPs' settings at run-time as well. Once the configuration of ISIF has been set by the user-friendly GUI, the LabVIEW™ software generates a text file with the settings of all the analog and digital IPs. The text file is used by the LEON firmware compiler and the settings are taken as parameters for the configuration chains of the different IPs.

A run-time configuration mode is possible as well. This type of configuration is used mainly when minor run-time changes of the configuration are needed. A firmware module running in the LEON firmware implements a communication protocol with the LabVIEW™ software running in the PC. Following this protocol, the GUI is able to set every IP's configuration register.

Resuming, the easily configurable interface architecture of ISIF, combined with this flexible configuration software, allow us to fast set up a test and characterization environment for different MOEMS, in contrast with the usual big amount of time needed to set up traditional (and often more expensive, even if more precise) characterization setups. Moreover, the run-time reconfigurability of the architecture speed-up also the eventual minor architectural changes, that in other setup could require a complete revision of the overall test architecture.

The GUI is depicted in Figure 2-4 where the configuration of a DAC is taken as an example.

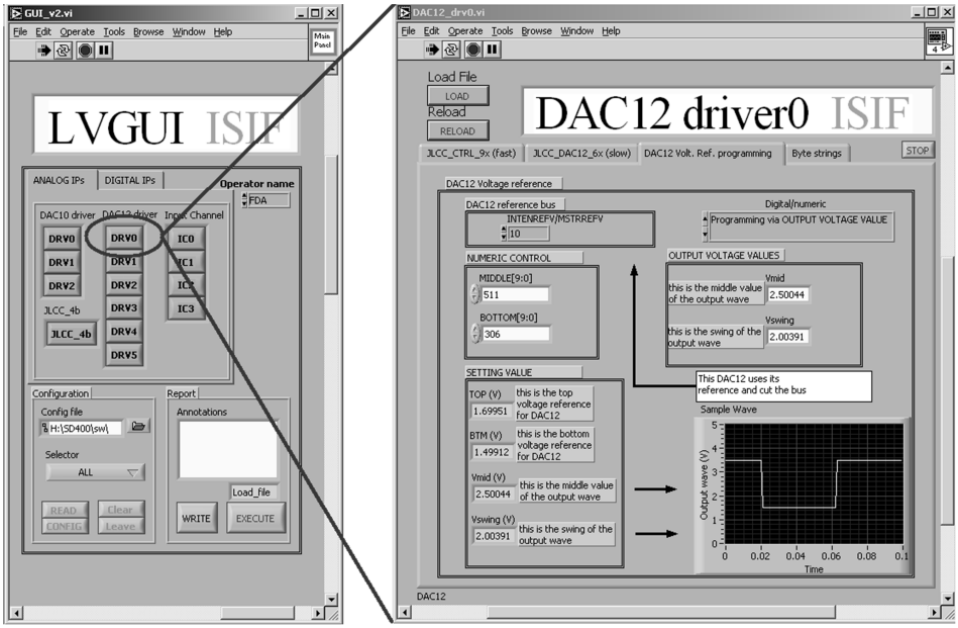


Figure 2-4. *LabVIEW™ GUI*

2.1.3 Test and characterization of a double axis resonating micromirror

2.1.3.1 Physical description and principle of operation of the micromirror

The structure under test is a double axis scanning micromirror used for image projection purposes together with laser sources [31]. It basically consists of a circular polysilicon mirror plate covered with aluminium and connected to a gimbal frame by a pair of polysilicon torsion springs (Figure 2-5). The micromirror is a dual axis structure: the slow axis has a resonance frequency of about 700 Hz while the resonance frequency of fast axis is about 30 KHz. The working principle of the device is the following: the fast axis enables the micromirror tilting around the y direction while the slow axis enables the micromirror tilting around x direction. Both axes are electrostatically actuated by means of vertical comb drives. Each vertical comb drive consists of a set of rigid electrodes bound to the substrate, and a set of moving electrodes linked to the axis and suspended over an etched pit.

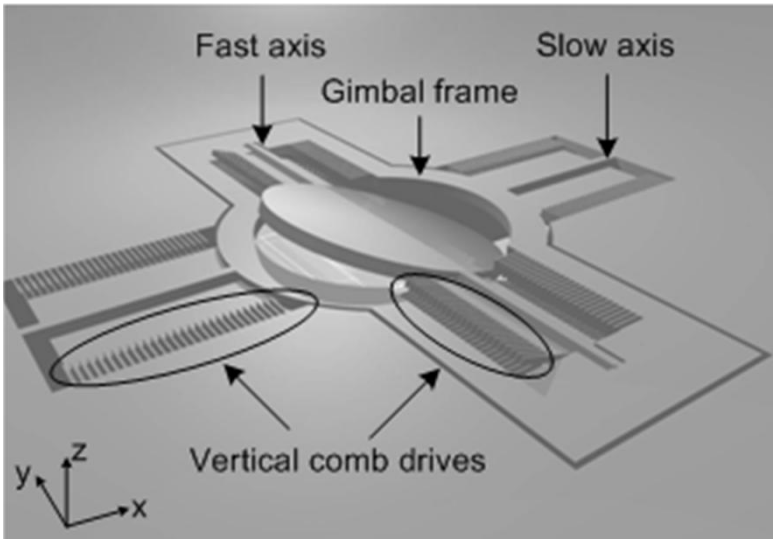


Figure 2-5. *Micromirror structure*

When a voltage is applied between the fixed and the movable electrodes, an electrostatic torque arises and induces the motion of the axis. The movable fingers rotate around the torsional axis until the electrostatic torque (T_e) and the mechanical restoring torque (T_m) of the springs reach the equilibrium condition [32]. The equations that describe the micromirror motion are:

$$T_e = T_m \tag{1}$$

$$\frac{1}{2} N_f V^2 \frac{dC}{d\vartheta} = K \cdot \vartheta \tag{2}$$

where N_f is the number of the fingers of each comb drive, V is the applied voltage, *C* is the capacitance between a fixed and a movable finger, *K* is the torsional spring constant and finally ϑ is the rotation angle. The previous expression highlights that the capacitance versus angle relationship is a fundamental parameter to model each micromirror axis at the resonance frequency as well as the torsional spring constant. These parameters have been extracted from the results of electrostatic simulations performed with a FEM tool (COMSOL Multiphysics™) as will be described in the next paragraph. This device has an actuation voltage up to 90 V.

2.1.3.2 Electrostatic FEM simulations

The relationship between the capacitance and the rotation angle has been extracted performing several electrostatic simulations with COMSOL Multiphysics™. In order to speed up the simulations without losing in generality and accuracy, we

adopted a simplified model of the structure which consists of only one movable finger between two fixed fingers (Figure 2-6).

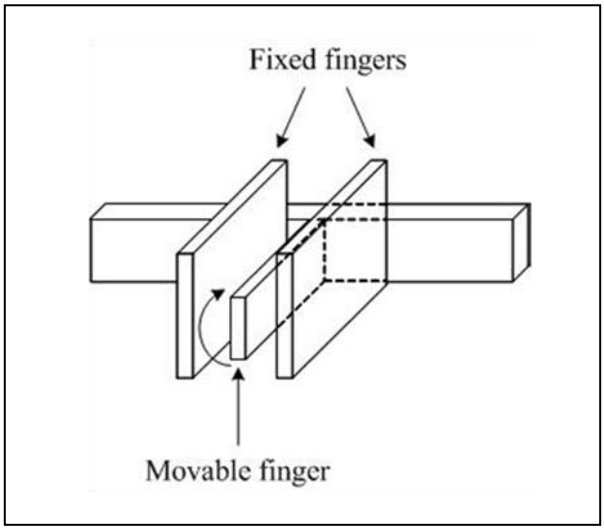


Figure 2-6. *Simplified structure used for electrostatic simulations*

As previously mention, the micromirror requires an actuation voltage up to 90 V, thus in our electrostatic simulations the upper half part of the fixed fingers is biased at 90 V while the movable fingers are kept at 0 V. From simulation results the electrostatic energy (W_{es}) stored between the fingers can be extracted and consequently the capacitance can be calculated as follows:

$$C = \frac{2W_{es}}{V^2} \tag{3}$$

where *V* is the bias voltage of the fixed fingers.
In order to automatically extract several capacitance values for several rotation angles, a MATLAB™ routine has been developed. This routine performs a rotation of 20° (from −10° to +10°) of the movable finger with steps of 1° and calculates the capacitance value for each step. Since COMSOL Multiphysics™ is MATLAB™ compliant, the routine has been directly imported in the COMSOL environment and used to obtain the capacitance versus rotation angle waveform (Figure 2-7). The capacitance versus angle relationship (Figure 2-7) for the micromirror fast axis is not symmetric in respect to the deflection angle, since both the fixed and the movable fingers are staggered in the vertical direction. This displacement is necessary to generate the electrostatic force that enables the structure deflection.

The analytical expression for the relationship between the capacitance versus angle relationship is the fifth order polynomial expression (4) and has been extracted fitting the curve of Figure 2-7 in MATLAB™ environment.

$$C(\vartheta) = 6523 \cdot \vartheta^5 - 1353 \cdot \vartheta^4 - 554 \cdot \vartheta^3 + 92.43 \cdot \vartheta^2 + 36.79 \cdot \vartheta + 7.084 \quad (4)$$

where *C* is the capacitance expressed in fF and ϑ is angle expressed in radiant.

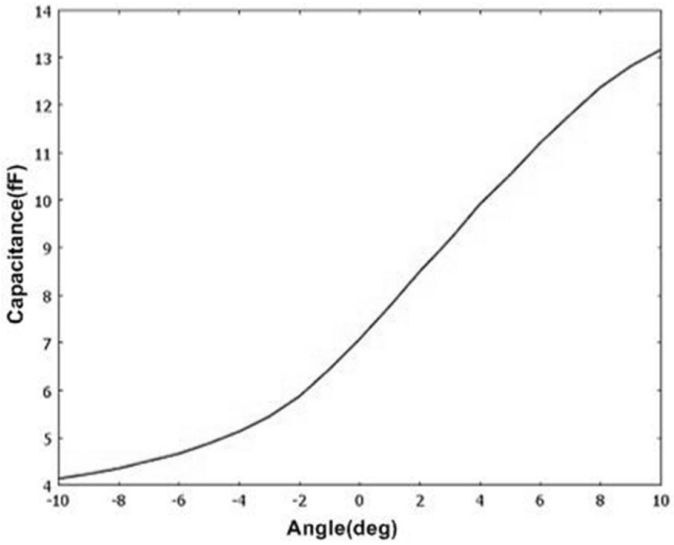


Figure 2-7. *Capacitance versus angle relationship for the micromirror fast axis*

2.1.3.3 Mechanical Simulations for the resonance frequency extraction

Each micromirror axis must be driven at the resonance frequency in order to reach the maximum rotation angle for a given voltage amplitude. Thus the study of the micromirror resonance frequencies is a key issue to perform a correct driving of the device.

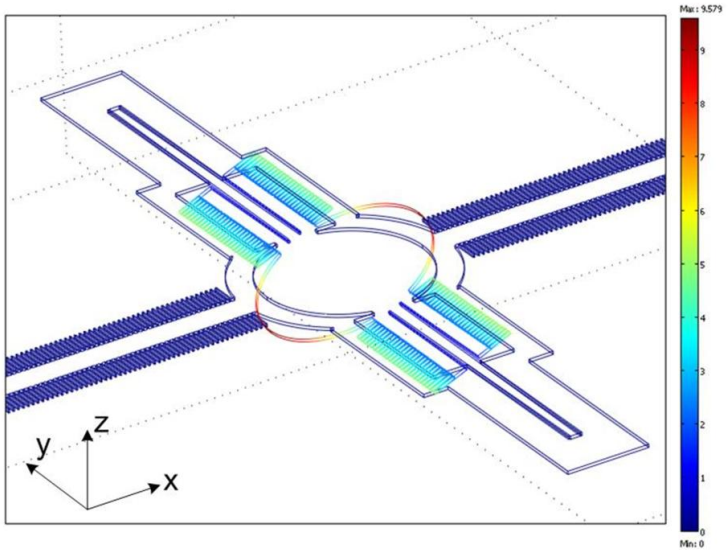


Figure 2-8. *Fast axis motion at its resonance frequency*

The eigenfrequency analysis has been carried out performing a mechanical FEM simulation in which no loads have been applied to the structure and the fixed boundaries have been set with appropriate constrains.
Table I shows the six lowest fast axis resonance frequencies obtained from the eigenfrequency analysis.

Table II – Resonance frequencies for the micromirror fast axis

1. f1(Hz)	2. f2(Hz)	3. f3(Hz)	4. f4(Hz)	5. f5(Hz)	6. f6(Hz)
7. 10938	8. 18597	9. 29945	10. 36889	11. 55249	12. 65730

The f3 resonance frequency (Table II) is related to a torsional motion of the micromirror axis in and out the x-y plane, as shown in Figure 2-8, and the value of this resonance frequency is confirmed by experimental results.

2.1.3.4 Mechanical Simulations for the K extraction

In order to have a complete characterization of the micromirror, the torsional constant of each micromirror axis is needed. The purpose of this section is to show how these constants have been calculated for the micromirror under study by performing FEM mechanical simulations.
For simplicity in this section only how to calculate the torsional constant of the micromirror fast axis will be shown, the procedure to calculate the torsional constant of the slow axis is the same.

The strategy adopted is the following: two opposite forces ($\pm F_Z$) have been applied at the two opposite sides of the mirror plate in the direction of the z-axis (Figure 2-9) and then a parametric simulation has been performed by setting the F_Z module as parameter.

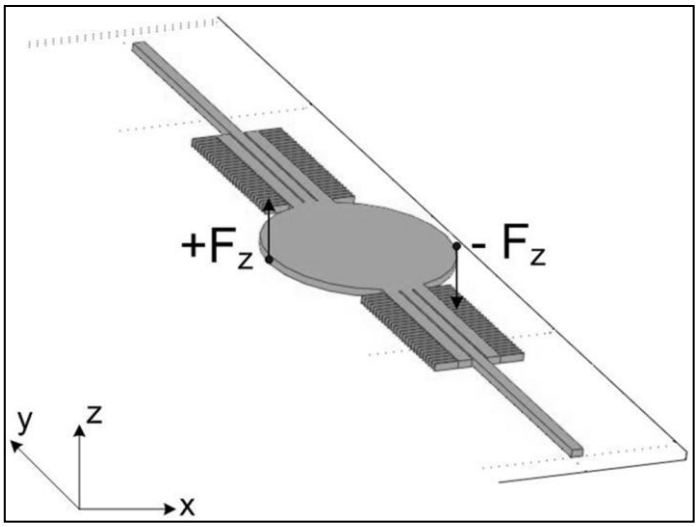


Figure 2-9. *Forces applied to the micromirror plate*

From these mechanical simulations different values of the z-displacement (z_{displ}) have been extracted for different value of the applied force and this data has been used to find the mechanical torque (T_m) and rotation angle, as shown in (5) and (6)

$$T_m = r \times F_z \tag{5}$$

$$\vartheta = \sin^{-1}\left(\frac{Z_{displ}}{r}\right) \tag{6}$$

where r is the micromirror plate radius.
The force module has been swept between 0 and 12 mN, thus obtaining the curve Torque versus angle shown in Figure 2-10. Fitting this curve in Matlab™ environment the relationship between torsional torque and rotation angle has been extracted:

$$T_m(\vartheta) = K_{nonlin} \cdot \vartheta^3 + K_{lin} \cdot \vartheta + T_0 \tag{7}$$

Where K_{nonlin}=7.458e-9 N*m/rad^3, K_{lin}=1.358e-5 N*m/rad and T_0=-9.903e-14 N*m. From the comparison between the magnitude of the K values and the constant term T_0 we can gather that T_0 can be neglected without losing in accuracy.

Moreover, even though the relationship extracted from Matlab™ fitting shows a non linear behavior of the fast axis due to the cubic term, the latter is not appreciable (as shown also in Figure 2-10) and consequently can be neglected.

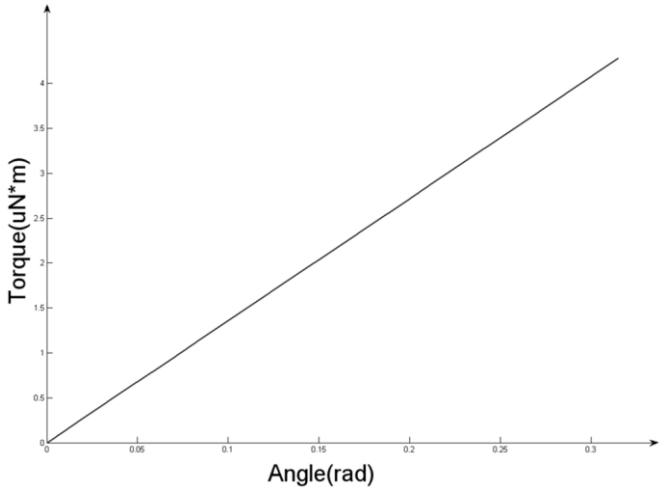
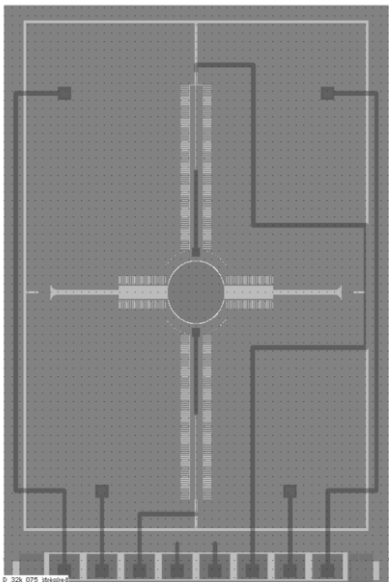


Figure 2-10. *Mechanical Torque versus rotation angle for the fast axis*

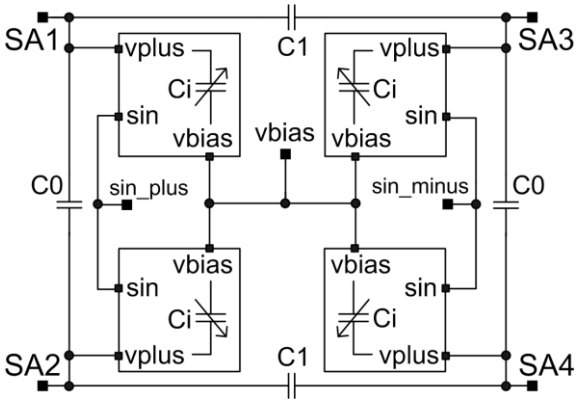
2.1.3.5 Electrical Model

The results of COMSOL electrostatic simulations and MATLAB post-processing have been used for the development of an equivalent electrical model in CADENCE™ environment for each micromirror axis.

In this section the model of the micromirror slow axis is shown. The slow axis can be divided in four quadrants (SA1,SA2, SA3 and SA4 as shown in Figure 2-11(a)) and each quadrant can be described with its equivalent capacitance.



(a)



(b)

Figure 2-11. *(a) Micromirror layout; (b) Slow axis model*

Figure 2-11(b) shows the Cadence model of the micromirror slow axis, the blocks on the left side represent the equivalent capacitance of SA1 and SA2 quarter slow axis, while the blocks on the right side represent the equivalent capacitance of SA3 and SA4 quarter slow axis.

The model describes the micromirror from a capacitive point of view and has shown its effectiveness in the design of the electronic circuitry for the micromirror driving and conditioning.
As already said each block represents the capacitance versus time relationship of a quarter axis at the resonance frequency. The capacitance versus time relationship C(t) has been extracted starting from C(ϑ) (4) and assuming that the rotation angle ϑ is a sinusoidal time dependent waveform at the resonance frequency.
In order to have an expression that can be implemented in HDL language the C(t) expression has been approximated by performing its Fourier transform with the use of the MATLAB Fourier function. Thus obtaining a C(t) function expressed as sum of sine e cosine functions multiplied for appropriate coefficients, as shown in (8).

$$C(t) = C_p + N_f \left[C_0 \cdot \sum_{n=1}^{4} (C_n \cdot \cos(n \cdot \omega_r \cdot t) + S_n \cdot \sin(n \cdot \omega_r \cdot t)) \right] \quad (8)$$

Where C_P is the parasitic capacitance of the moving structure versus the substrate, N_f is the number of finger correspondent to a quarter axis, C_0 is the static capacitance, C_n and S_n are the coefficient of the Fourier Transformer and ω_r is the slow axis resonance frequency.
Each block of Figure 2-11(b) is the description in HDL language of the Fourirer transform of a quarter axis plus the parasitic capacitance between the moving structure and the substrate extracted from laboratory measurements.
The signals *sin_minus* and *sin_plus* represent the micromirror nodes that are connected to the driving stages of the ISIF in the real device and *vbias* is the constant low voltage applied to the fixed mirror electrodes. In our simulations and laboratory tests *vbias* has been fixed to 0V. C0 and C1 are the parasitic capacitances between different comb drives and were evaluated with the ISIF platform as described in the following section. SA terminals are used for sensing purposes. A similar model has been developed for the fast axis following the same procedure.

2.1.3.6 Simulink™ model

The characteristic micromirror parameters, extracted as shown in previous sections, have been used to build up a Simulink™ model for each micromirror axis (Figure 2-12) which has been successfully verified via experimental measurements. The model has proved his effectiveness in the high level study of the micromirror conditioning system.
The Simulink™ model represents the micromirror behavior from both a mechanical and an electrostatic point of view. In this section only the model of the fast axis will be shown, but the model of the slow axis has been developed by following the same procedure.
The micromirror fast axis is driven by applying to the fixed fingers of the opposite comb drive the two voltage signals (9) and (10)

$$V_{drive\_right} = V_{BIAS} + V_A \sin(2\pi f_R t) \quad (9)$$

$$V_{drive\_left} = V_{BIAS} - V_A \sin(2\pi f_R t) \quad (10)$$

where V_{BIAS} is the DC polarization voltage, V_A is the amplitude of the sinusoidal actuation voltage signals and f_R is the resonance frequency of the axis.
The torque momentum responsible for the micromirror rotation is the difference between the two electrostatic torques that arise from the application of voltage (9) and (10) and is expressed by (11).

$$T_e = \frac{1}{2}\left(\frac{dC_{dr}}{d\vartheta} V^2_{drive\_right} - \frac{dC_{dl}}{d\vartheta} V^2_{drive\_left}\right) \quad (11)$$

In the model of Figure 2-12 the driving voltages V_{drive_left} and V_{drive_rigth} generate an electrostatic torque which is the input of the block that implements the mechanical transfer function of the micromirror axis. The output of this block is the rotation angle of the fast axis and it becomes the input of the two blocks which represent the capacitive behavior of the micromirror fast axis. Indeed the first one implements the derivative of the capacitance versus angle relationship while the second one implements the capacitance versus angle curve. The first block is used to generate the $\frac{dC}{d\vartheta}$ terms which appear in the electrostatic torque expression (11) while the second block is used to extract the capacitive signals (C_{sr} and C_{sl} of Figure 2-12) that will be used in the read out electronic circuit.

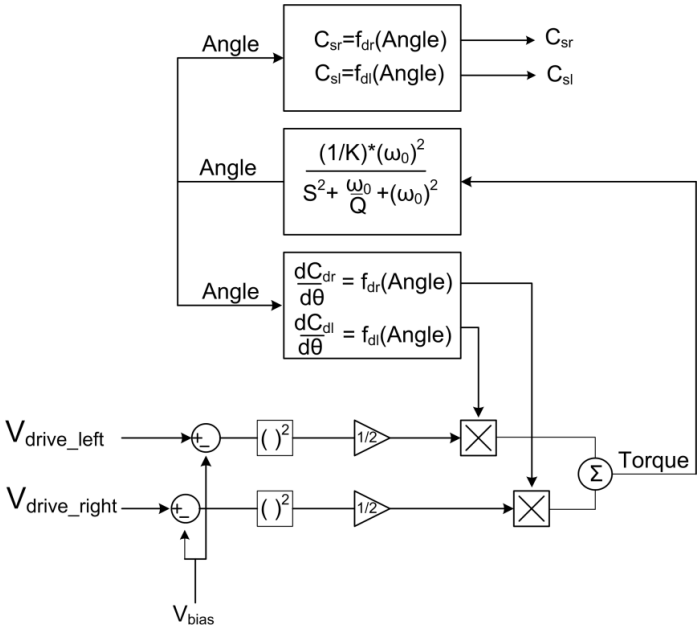


Figure 2-12. *Block diagram of the Simulink™ model for the micromirror fast axis*

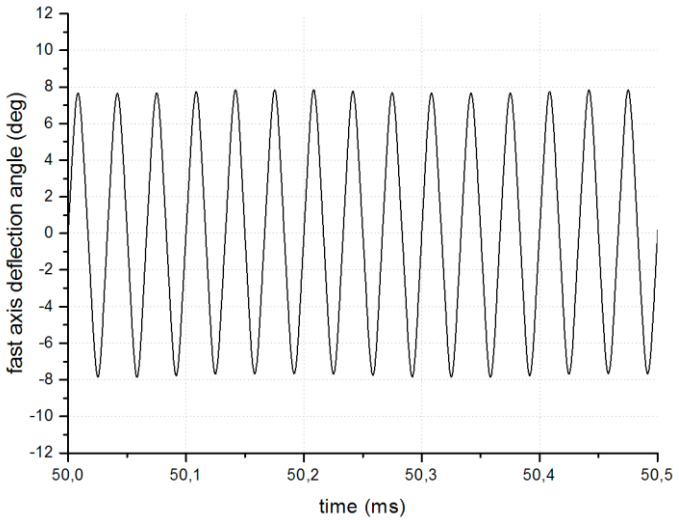


Figure 2-13. *Rotation angle versus time curve for the micromirror fast axis*

Simulation results show the effectiveness of the developed model. In fact the angle extracted from simulations is a sinusoidal rotation angle characterized by amplitude equal to 7.8 degrees as shown in Figure 2-13. This result is in accordance with experimental measurements performed in laboratory as will be shown in the following section.

2.1.4 Test and characterization with ISIF flow

The aim of the previously described simulations and of our test and characterization environment is to obtain a model of the device as close as possible to the real device.
Electrostatic simulations do not allow for example the evaluation of the parasitic capacitance values and effects in a simple way (i.e. the measurements of parasitic capacitances is mandatory for the evaluation of the interaction between the different comb drives and between the two axes of the mirror). A solution would be using a complete device geometry, perform 3D FEM electrostatic and dynamic simulations and extract the capacitance values in a similar way as described in the previous section. The two main drawbacks of this approach are the high computation power and long time required to perform such kind of simulations. The strategy we followed is to use the correlations between simulation results and test results performed exploiting the high flexibility of the ISIF platform. In this way it is possible to reduce the test and characterization costs and time without losing too much in accuracy.
The testing set-up used in laboratory is set equal to the voltage source configuration used in the CADENCE electrical simulations. The results of the tests are used to validate and improve the model accuracy and the electrical simulations are performed again. Summarizing, the two important goals achieved are: the development of an electrical model as close as possible to the real micromirror that

can be used for a more accurate design of the final driving and sensing blocks and a complete characterization of the device.

2.1.4.1 Setup and static capacitance measurement

The micromirror has been tested and characterized using the development board of Figure 2-14 following the approach previously described. Firstly, the MEMS has been connected to the ISIF and the static capacitances (parasitic capacitances included) of the structure have been measured. The capacitances were measured detecting the gain of the input channel charge amplifier with the capacitance under test used as part of it and a known feedback capacitance. The driving sine wave was provided by the numeric controlled oscillator of the ISIF linked to a DAC with the High Voltage section (HV section) bypassed. The most important results were the evaluation of several capacitance values: (i) the static capacitances between a slow axis driving electrode and the Moving Structure (MS) are about 100 pF, (ii) the capacitances between a fast axis driving electrode and the MS are about 30 pF, (iii) the capacitance between the MS and the Substrate is about 660 pF.

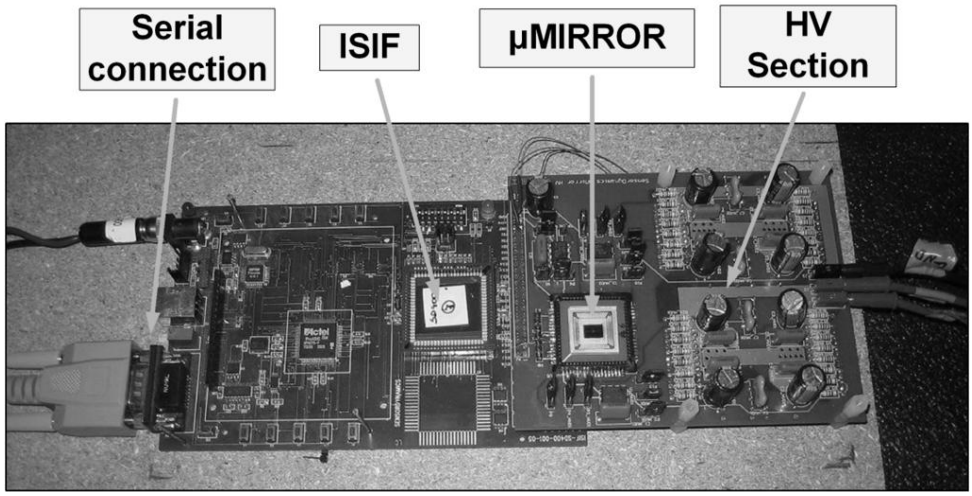


Figure 2-14. *Test and Characterization setup*

2.1.4.2 Dynamic ΔC measurements

A key point of the test and characterization of the micromirror has been the measure of the dynamic capacitances (ΔC) from the sensing electrodes. The capacitance variation is directly related to the position of the moving structure's fingers in respect to the fixed sensing electrodes; thus it is possible to detect the mirror position measuring the ΔC variation. The sensing signal is acquired by the input channel, amplified, filtered, and converted in digital.

Firstly, an open loop analysis has been performed in order to detect the resonating frequencies of the micromirror axis. In Figure 2-15 the results of a measure performed on the slow axis are shown: the y-axis represents the digitally converted value of sensing signal, the x-axis represents the driving signal frequency. The

driving signal is a 90 V peak-to-peak sine wave with a 45 V DC component. As we can see from the graph, the resonant peak of the slow axis is at about 735 Hz; the fast decreasing behavior of the amplitude is due to the fact that the signal is the composition of two contributes: the ΔC signal and the coupling signal due to parasitic capacitances. The ΔC signal is affected by a phase inversion due to the phase delay of the mechanical structure response which appears just after the resonant peak, while the coupling signal has no phase inversion; the sum of these two contributions causes the behavior represented in Figure 2-15. The mechanical deflection angle of the slow axis has been evaluated in ±8° using a laser source. The maximum deflection angle of the slow axis that can be achieved for different driving frequencies is shown in Figure 2-16. The same analysis has been performed on the fast axis. The maximum deflection angle of the fast axis that can be achieved for different driving frequencies is shown in Figure 2-17.

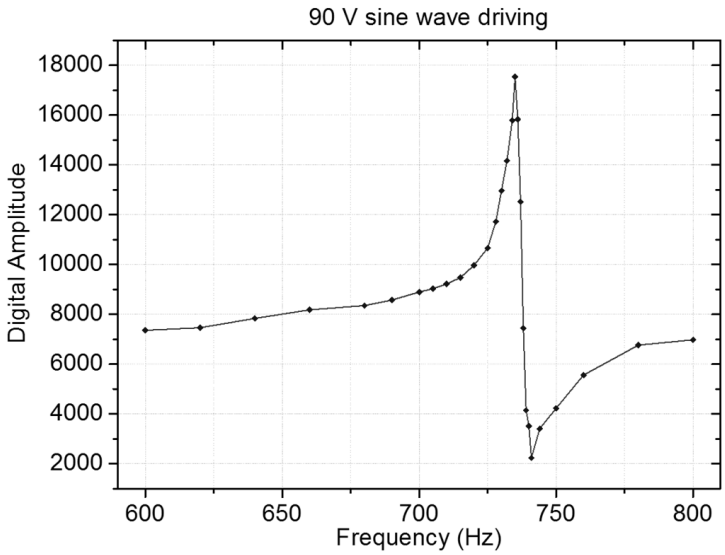


Figure 2-15. *Digital amplitude of the sensing signal versus frequency extracted from the open loop analysis on slow axis*

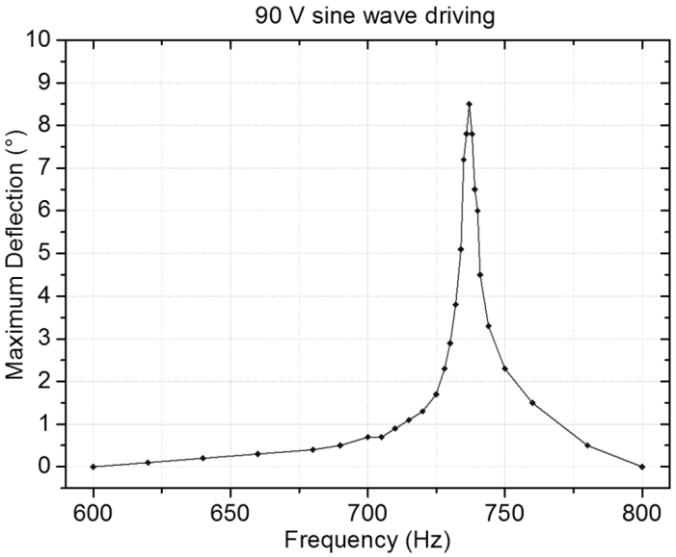


Figure 2-16. *Maximum deflection angle versus driving frequency extracted from the open loop analysis on slow axis*

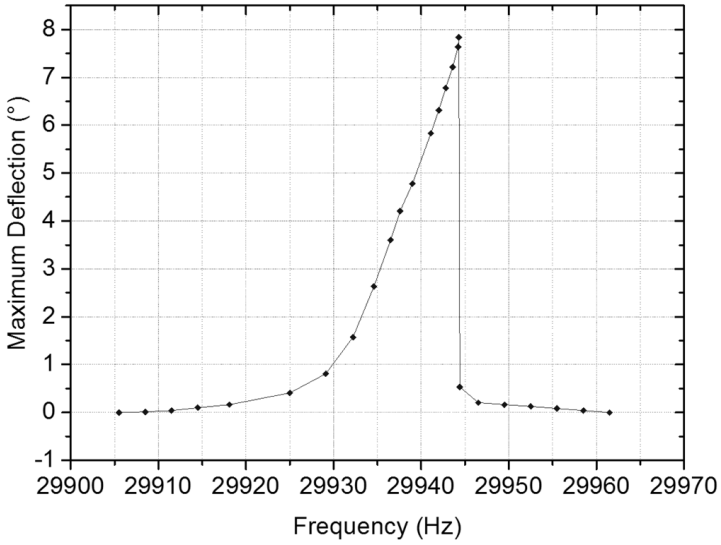


Figure 2-17. *Maximum deflection angle versus driving frequency extracted from the open loop analysis on fast axis*

Secondly, a closed loop driving on the slow axis has been performed. Figure 2-18 shows the functional block scheme of the loop system. An Automatic Gain Control

(AGC) has been implemented to control the driving amplitude. A second order PLL fixes the frequency of the sine wave generated by the NCO so that the micromirror driving signal and the signal sensed by the ISIF input channel are 90° out of phase. The mechanical response of the micromirror introduces a 90° phase delay when the axis is driven at its resonant frequency.
The sensing signal is detected by the input channel, filtered, digitally converted and sent to the demodulator, which generates two control signals, the first one for PLL frequency locking and the second one for gain control. These two signals are processed by the software DSP, which implements a PI (Proportional Integral) control, whose outputs are connected respectively to the NCO and the modulator for frequency and amplitude driving.

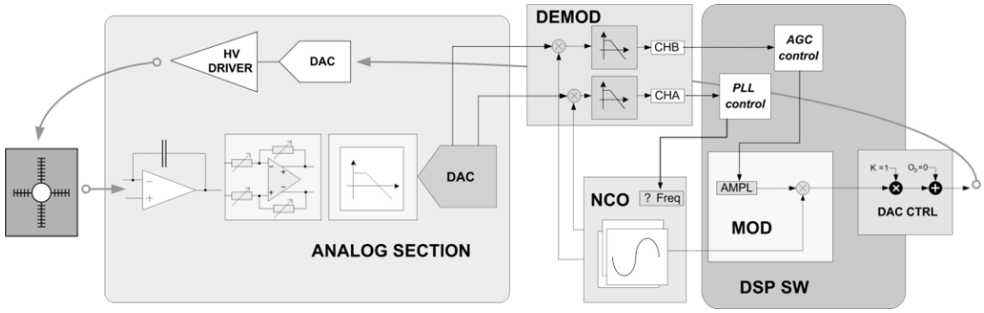


Figure 2-18. *Closed loop driving block diagram*

2.2 Pin-limited frequency converter bridge for fast prototyping of custom functionalities in platform-based sensor interfaces

2.2.1 State of the Art Review

During the last years, the use of platform-based systems is widely spreading, especially in the automotive field, for sensor interfacing and conditioning [24][33][34]. The adoption of configurable/programmable platforms, adaptable to a large set of sensors, is particularly useful during the first steps of a new design to speed up the comprehension of the system before developing ad-hoc architectures [25][35].
Several solutions have been developed to allow the communication between an ASIC and a programmable chip, but none of them allows pin-limited connections via external plugs. An example should be represented by the incorporation of FPGA cores into the ASIC [36]. This strategy offers high performances, but reduces system flexibility, because FPGA's number of gates can be embedded is a relative small percentage of the total ASIC gates. Instead, using an external solution, it is possible to choose the suitable technology (FPGA or DSP or Microcontroller) for the out-of-chip IPs.

For the IP bridge design presented in this paragraph we inherited the experience in downscaling technique from the GALS (Global Asynchronous, Local Synchronous) architectures, whose main characteristic is the partition of the SoC into isolated synchronous islands that have independent frequency and phase clock [37]. Differently from them, we have not used bisynchronous FIFOs to synchronize different clock domains [38], but to downscale the clock to reduce interferences generated by high frequency transmission in mixed signal systems.

2.2.2 Project Design Flow

The bridge is basically intended to be implemented on ASIC solutions, including bridge IPs in the final project. During project flow we had to consider the main advantages of various technologic solutions:

- ASIC: high performances and better integration for analog and digital circuits in mixed signal design;
- FPGA: best in terms of versatility and reconfigurability;
- FPGA to ASIC interface: needed during prototyping and testing phases and for particular applications not included into chip design (external System on Programmable chip – SoPC).

We made a first implementation of key blocks on FPGA platform to verify timing and area constrains in terms of slacks and standard cell number. After this phase, we have implemented all the design on FPGA. In this way we have been able to perform a preliminary test of the whole bridge, communicating with different IPs on the same FPGA that have been initially discarded from the first ASIC implementation. We have been able to test both the bridge and a side part of the final ASIC project to refine and optimize critical parameters to evaluate the implementation of new IPs on ASIC project. The IP bridge is finally integrated on ASIC technology (see case study in Section 5).

2.2.3 IP Bridge Architectural Design

The frequency conversion takes place using bi-synchronous FIFOs (Figure 2-19). Each FIFO manages two different totally uncorrelated clock domains (one for writing and one for reading operations) basically maintaining a strict division between them, comparing only the two read and write location indexes.
For example, if we consider a single FIFO that has been written at the fastest clock and has been read at the slowest one, we perform the domain conversion with the only passage of the index indicating the last written location to the slowest domain. In this way a read operation can be performed in the right location. The reading index is incremented every read operation until empty condition is verified. The dual passage is operated by sending the last read location to repeat the previous described procedure in the fastest domain to handle writing operations correctly, checking if the condition of FIFO full is not verified. Both indexes are gray-coded before the frequency conversion to increase the tolerance to errors. The indexes are sent to the other domain by a triple registration through D flip-flops barrier (the first one clocked with the starting clock frequency and the second couple with the arrival clock frequency) to avoid meta-stability conditions. The comparison on each couple of indexes (last written and read locations) in the same domain for each side of the FIFO presents two main advantages: (i) the system could guarantee a

correct synchronization having at its disposal all the information about the other domain at its own clock frequency, (ii) empty and full conditions can be detected by index comparison.
The write pointer always points to the next word to be written and, similarly, the read pointer always points to the current FIFO word to be read; on reset, both pointers are set to zero, which also happens to be the next FIFO word location to be written. On a FIFO-write operation, the memory location that is pointed to by the write pointer is written, and then the write pointer is incremented to point to the next location to be written, the empty flag is cleared and a valid read operation is now possible.
Referring to Figure 2-19, full flag is managed by system clock domain. We need this condition to avoid mismatches with the master writer and with the whole system on the ASIC. All the design is synchronized with system clock, so we preferred to keep this condition for every communication and interrupt at the ASIC side. In normal operation conditions (no full), write pointer and its gray coded version into downscaled clock domain are incremented every write operation. In case of full state, gray coded write pointer is not incremented until a valid write operation (out of the full condition, so after at least a read) is done. In this way the gray coded pointer is equal to the write pointer minus one. After a read operation, the two write pointers are re-synchronized. For empty state in downscaled clock domain we have a dual situation. In this way, full and empty states can be separately managed simply by index equality comparison for each side without having the two flags set at the same time.

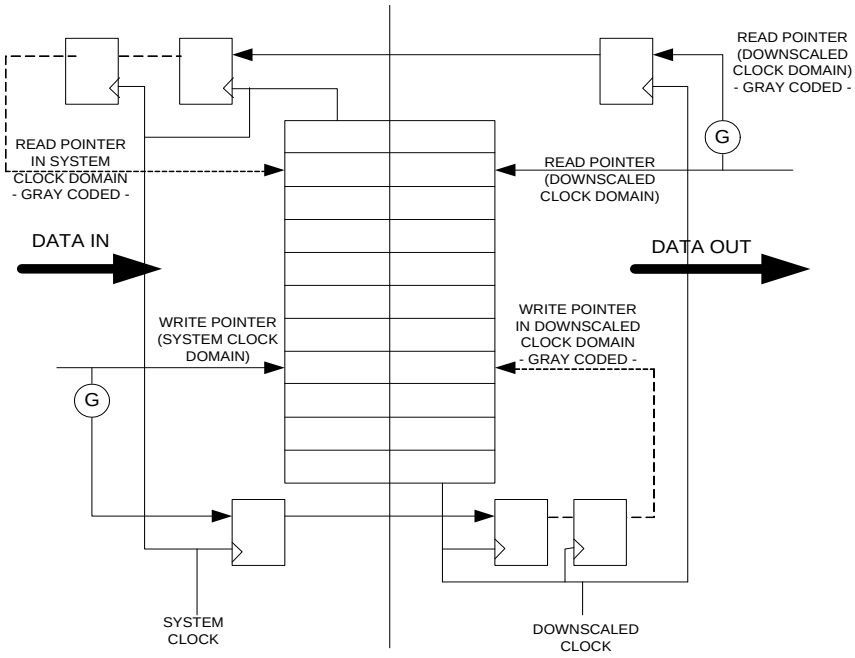


Figure 2-19. *Bi-synchronous FIFO*

In Figure 2-20 the pinout of the IP bridge is presented. The whole module, described as a parametric VHDL IP cell, is an AHB lite standard slave; its completed architecture is shown in Figure 2-21.
The main manager (e.g. an AHB master) of the ASIC cannot access directly to the slaves on the off-chip FPGA. It must write (read) only into (from) the transmission (reception) FIFOs; the access to the desired slave is made by an interface that operates the final protocol change on the external module.
The bridge is a critical bottle-neck for the whole system in terms of performances, and the main limit is represented by the lowest frequency clock domain. The proposed system operates in single transfer mode, so the interface does not provide signals to manage burst operations. This feature is not necessary, and hence is not supported, because of the low frequency needed by pads.
Three FIFOs are necessary to implement the write/read transaction between the ASIC and the external device. A first write-only FIFO, FIFO1 in Figure 2-21, operates the frequency conversion on data representing the address of the desired out-of-chip device. The second write-only FIFO, FIFO2 in Figure 2-21, operates in the same way on data to be written at the previously specified location. For a writing operation on off-chip slave devices the master must operate two transfers: the first to FIFO1, sending the address of the desired device as data, and the second to FIFO2, providing the effective data to be sent to the off-chip device. In case of reading, the second operation does not take place. The read operation terminates with the storing of the data read in FIFO3. This executes the frequency upscale on data read from out-of-chip IPs and appears as a read only module for the master side. When a new data is ready, an interrupt is asserted. In this way, the bus is not locked during the read operation that usually is slow, due to the off-chip clock frequency lower than the on-chip one.

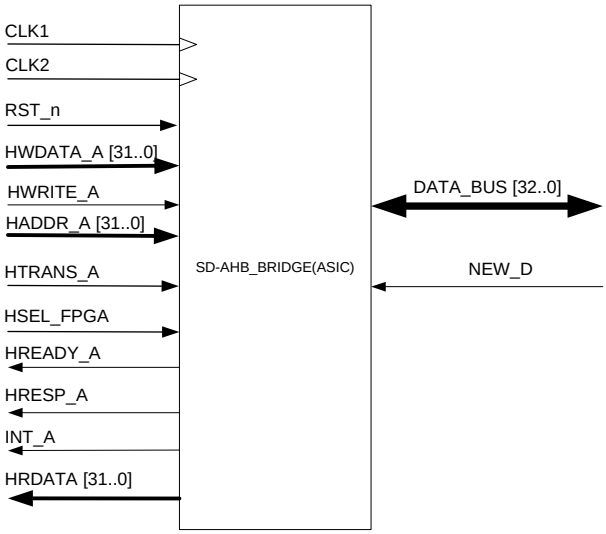


Figure 2-20. *IP bridge pinout*

We chose to represent each FIFO as an address inside a single slave, masking the external module to the bus' master, to avoid bus locking for long time. In fact, during a read operation, if all the slaves on the FPGA could be directly accessed by the master, the bus should be locked for all the time needed to transmit the address to the external unit and to receive the data. Considering the downscaled domain, this period of bus inactivity could delay CPU activity too much. Writing the FIFO as a standard slave, the CPU can perform other operations during the communication with the external device that is fully independent from the system core.
During the master design, the number of IPs needed in the prototyping phase could not be known. So, address sending as data of an AHB write operation on bridge slave location, can enhance the flexibility of the system, allowing IP addiction on FPGA without master modifications.

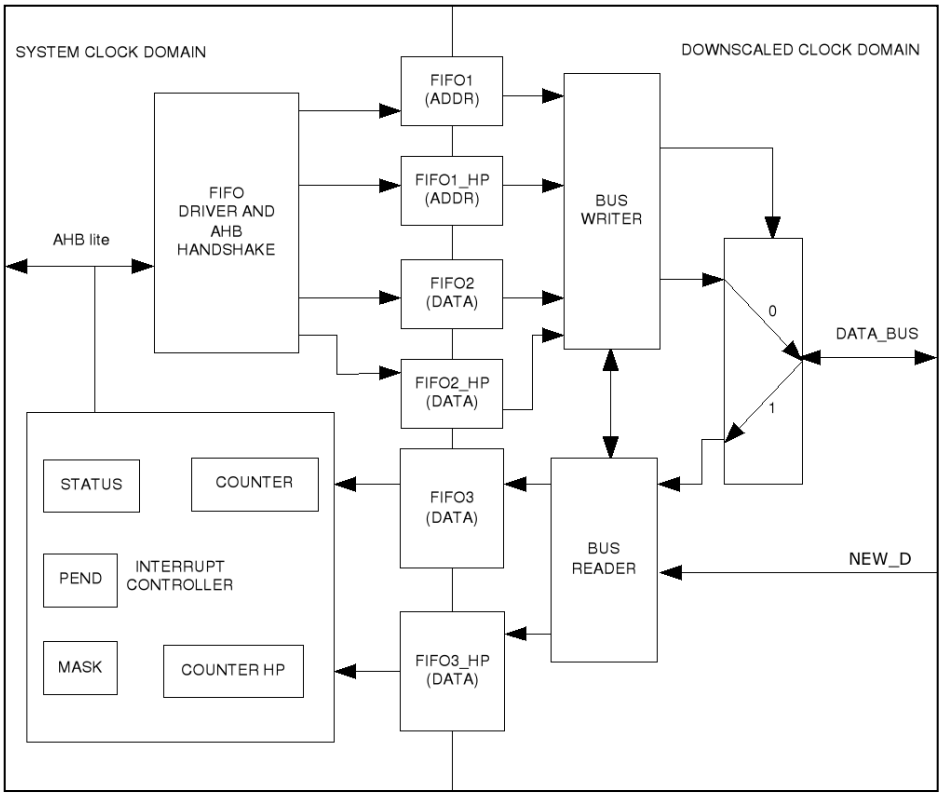


Figure 2-21. *IP bridge architecture*

To better understand the adopted strategy, in the following a simple C code extract is shown.

```
// Address of fifo designed to keep addresses
#define BR_FIFOADDR (*((volatile unsigned long*) 0xE000AB00))

// Address of fifo designed to keep addresses for high priority
transmissions
#define BR_FIFOADDRHP (*((volatile unsigned long*) 0xE000AB04))

// Address of fifo designed to keep data to write
#define BR_FIFODATAW (*((volatile unsigned long*) 0xE000AB08))

// Address of fifo designed to keep data to write for high priority
transmissions
#define BR_FIFODATAWHP (*((volatile unsigned long*) 0xE000AB0C))

// Address of fifo designed to keep data to be read
#define BR_FIFODATAR (*((volatile unsigned long*) 0xE000AB10))

// Address of fifo designed to keep data to be read for high
// priority transmissions
#define BR_FIFODATARHP (*((volatile unsigned long*) 0xE000AB14))

// Address of status register
#define BR_STATUS (*((volatile unsigned long*) 0xE000AB18))

// Address of pend register
#define BR_PEND (*((volatile unsigned long*) 0xE000AB1C))

// Address of counter register
#define BR_COUNTER (*((volatile unsigned long*) 0xE000AB20))

// Address of counter HP register
#define BR_COUNTERHP (*((volatile unsigned long*) 0xE000AB24))

// Address of mask register
#define BR_MASK (*((volatile unsigned long*) 0xE000AB28))

/*****************************************************
 * Example of low priority write procedure
 *****************************************************/
int write_data(uint addr_fpga_slave, int data)
{
    // full flag for fifoaddr or fifodataw is set
    if((BR_STATUS & 0x4 ) == 1 ||
       (BR_STATUS & 0x10) == 1)
        return -1; // Error

    // Write fpga IP address in fifoaddr with Write flag set
    BR_FIFOADDR = addr_fpga_slave | 0x1;
    BR_FIFODATAW = data;
    return 0; // OK
}

/*****************************************************
 * Example of low priority read procedure using polling
```

```
 ******************************************************/
int read_data(uint addr_fpga_slave, int& data)
{
    // full flag for fifoaddr is set
    if((BR_STATUS & 0x4) == 1)
        return -1; // Error

    // Write fpga IP address in fifoaddr
    // with Write flag clear (read operation)
    BR_FIFOADDR = addr_fpga_slave & 0xFFFFFFFE;

// wait until new data flag is set
while(BR_STATUS & 0x1);
    data = BR_FIFODATAR;
    return 0; // OK
}
```

In the proposed IP bridge (see Figure 2-21) the paths have been duplicated with different addresses to support the management of different priority transfers. So, a high priority (HP) transfer request by the master CPU will be served immediately at the end of the current transaction.
The entire writing structure is managed by a finite state machine, called “bus writer” in Figure 2-21, that performs the empty checks on address FIFOs (respecting their priority) and then transmits address and data. Note that the target device that manages AHB transmissions on external side receives the information about the kind of operation to perform (read or write) through the LSB of the address, exactly as happens into the ASIC side. Figure 2-22 shows the meaning of pinout in transaction from the ASIC to the FPGA. To minimize the number of pins, the information about address, data and type of operation are multiplexed on the same bus. Particularly, during the addressing phase the meaning of the pinout is the following: the external unit is addressed by the AHB master with a 30 bits string word (DATA_BUS[31..2]), bit 1 of the DATA_BUS is not meaningful and bit 0 indicates if the transaction is a read or a write operation. Note that during the data communications phase DATA_BUS [31..0] contains the 32 bits word to be transferred. Communications are synchronized through a bit inside each transmission (NEW_D represented by DATA_BUS[32], see Figure 2-22) that toggles every time a new data is ready on the bus.
The other manager for reading operations, called “bus reader” in Figure 2-21, operates in a similar way, waiting for a new data signal from the FPGA and performing reading FIFOs storage. The notification of a new data from an external device is represented by the toggling of the NEW_D pin in Figure 2-21, which causes a read interrupt request to the ASIC’s CPU. The NEW_D pin from FPGA to ASIC is unidirectional and isolated from the DATA_BUS because it contains the information about multiplexer commutation. Figure 2-23 shows the meaning of pinout in transaction from the FPGA to the ASIC. The 32 bits read data is transferred through the bits DATA_BUS[31..0].
In a transmission to the bridge (from the off-chip FPGA), an error bit is provided to inform the ASIC if current read data is not valid due to an error occurred during off-chip operations. To minimize the pinout we choose to change the meaning of the

bit 32 of the DATA_BUS with respect to Figure 2-22, using it as an error flag (see Figure 2-23). Summarizing the total number of pads results equal to data width plus two.

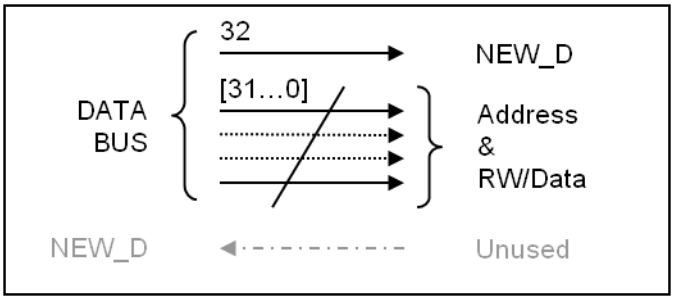


Figure 2-22. *Meaning of pinout in transaction from ASIC to FPGA*

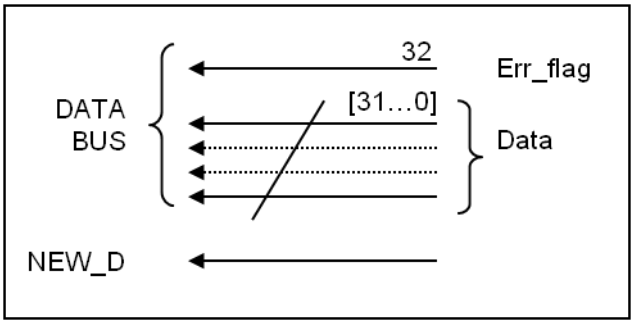


Figure 2-23. *Meaning of pinout in transaction from FPGA to ASIC*

The number of locations for each single FIFO can be defined by the IP's user during the synthesis phase, setting a VHDL generic's value. Because of the way the FIFO is designed, the number of locations must be greater than two: this constraint is introduced to allow the distinction of full and empty state conditions. To keep a priority path sense and to minimize area occupation, it is always recommended to implement a small priority queue.
An AHB interface block implements AHB lite protocol and FIFO (or internal registers) control, checking the full or empty conditions for write and read FIFOs respectively. For data writing, only one clock cycle is needed. The HRESP signal is always set, except in case of illegal operations (write on a read only register or vice versa) or non-existent internal addresses requests.
The data latency, corresponding to time between AHB information sampling and first data available on bidirectional bus structure, can be expressed as follow:

$$2 \cdot T_{down\_clk} < T_{lat} < 3 \cdot T_{down\_clk} \tag{12}$$

where T_{down_clock} is the period of the downscaled clock. To explain equation (12) we have to consider two components: one cycle for double flip-flops barrier and another needed to FIFO to make data available since enable signal reception. Latency is greater or equal to this quantity, depending on relative skew between the two clocks that cannot be greater than a downscaled clock cycle.
To monitor each path, the user can access some registers such as a control register that provides reset signals for each single FIFO. Due to the different clock frequencies, the reset operation requires several clock cycles to be executed, as shown in Equation (13), where T_{sys_clk} and T_{down_clock} are the periods of the AHB clock and of the downscaled clock, respectively:

$$4 \cdot T_{sys\_clk} + 2 \cdot T_{down\_clk} < t < 5 \cdot T_{sys\_clk} + 3 \cdot T_{down\_clk} \quad (13)$$

Two faster and two slower cycles are needed to pass through synchronization flip-flops and come back. Two additional faster clock cycles are required to set the reset flag at the beginning of the operation and to automatically remove the reset request into the dedicated register. As seen from previous equation (12), a clock cycle for each domain must be added to consider worst case on relative skew between the two clocks.
The reset requests into the control register are automatically removed by a dedicated hardware and the CPU must check that all requests have been removed before starting further operations.
There are several other registers the user can access in read, write or read/write mode (depending on the specific register selected) for real-time transfer monitoring. All accesses to registers are synchronized by the system clock signal. These registers allow to monitor the status of each FIFO and to read all the interrupt events that can be also selectively masked. Two counters give the instantaneous value of the location that will be involved in the upcoming operation for the correspondent reception FIFO.

- Control register allows to send the reset signal to each FIFO selectively.
- Status register is used to read the presence of a new data, the full and empty signals and error on data for each FIFO; this register is read only.
- Pend register holds the information about the source of interrupt request, after being masked by means of the mask register, with the following convention for each bit; this register is read only.
- Mask register is used to set or not the possibility to generate an interrupt request when a protection is activated as a result of an external fault.

2.2.4 Implementation and test results

Several tests have been performed on the bridge to identify the working domain in terms of temperature and supply voltage, see Figure 2-24 and Figure 2-25.
For the synthesis, we have used a new mixed signal technology, called BCD8, developed by ST Microelectronics for industrial and automotive applications [39]. The BCD8 features high-density 0.18 μm logic with 1.8 V supply voltage plus higher-voltage devices at 5 V, 12 V and with DMOS up to 70 V. Tests have been performed on post-place and route models, using tech libraries by

STMicroelectronics, including six different operating conditions: T=-40°C and V=1.95V, T=25°C and 1.8V, T=85°C and V=1.6V, T=150°C and V=1.95V, T=150°C and V=1.55V, T=150°C and V=1.08V. Indeed for automotive IP macrocells is essential the characterization in all different conditions considering temperature ranges from -40 to 150 °C.
The target speed is 100 MHz for the AHB system clock while the downscaled frequency for output pads is 10 MHz. The off-chip frequency has been evaluated as the best trade-off between pads' efficiency and global communication speed. A much higher downscaling factor will represent a bottleneck in our case study where the off-chip FPGA is used to perform signal processing tasks.
Figure 2-24 shows the bridge area vs. the size of the FIFOs, considering 32 bits data (so with 34 pads), nominal operating conditions and the adoption of two different priority paths. Figure 2-25 presents the area occupation and the worst time slack or a target configuration of 8 FIFO locations and 4 FIFO_HP locations at different operating conditions.

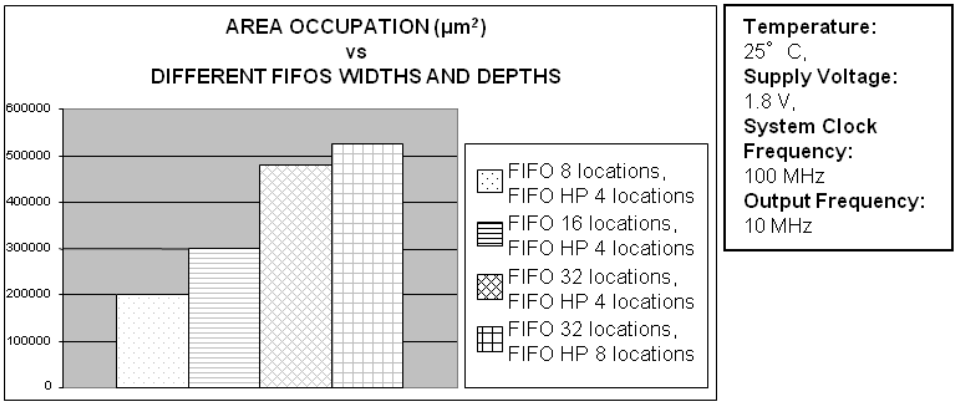


Figure 2-24. *Bridge area vs. different size of the FIFOs*

2.2.5 Case Study of an Automotive Smart IC Sensor

We have integrated the IP bridge described above in an ARM926 (a 32 bits 100 MHz CPU) based architecture, synthesized in BCD8 technology. The aim of the architecture is the implementation of a flexible framework for closed loop control of MOEMS (Micro Opto-Electro-Mechanical System) sensors; flexibility is obtained through real-time software management of the whole control system.
The specific case study refers to MOEMS sensors used to project images on car's glass by a single micromirror, in case of bad visibility conditions and for driver's assistance, identifying the control and monitoring requirements before final SoP (System on Package) integration. A side target is also the performance test and the constraints identification of the new mixed signal BCD8 technology.

The proposed architecture is characterized by an analog part for sensors' driving control and sensing reading, described in [31][40]. A digital acquisition and processing part implements the control loop and the monitoring system. We use the bridge described in 2.2.3 to communicate with an off-chip FPGA used to realize a simplified generic FPGA. Figure 2-26 shows the ASIC communicating with external FPGA through the bridge and the block diagram of our test framework for the digital part respectively.

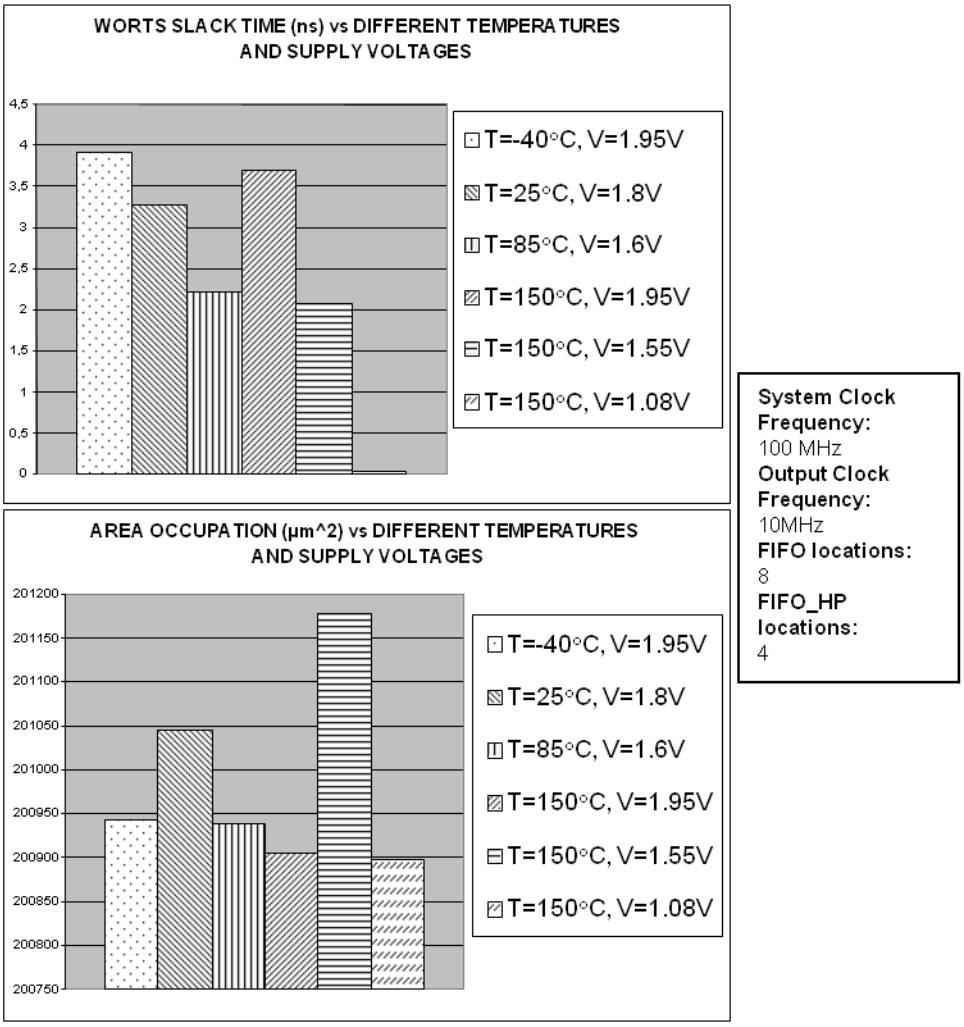


Figure 2-25. *Bridge slack time and area vs. temperatures and supply voltages*

The design foresees three different clock domains: 100 MHz for IPs on AHB bus (ARM9, memory controller, interrupt controller…), 50 MHz for APB IPs (demodulator, UART, timers…) and 10 MHz for output pads dedicated to FPGA connections. Memory resources are an on-chip SRAM for image data (76800 byte for VGA resolution) and two off-chip memories (8Mx16 Flash and 256Kx16x16 SRAM). The custom bridge is connected to the AHB bus matrix needed to separate instructions and data flows (multi-layer AHB architecture). Total area occupation is roughly 31 mm2. The integration of the bridge in this architecture does not increase the complexity of the system, because it has a simple AHB slave interface. Moreover, thanks to its transparency due to FIFO capability, it does not introduce overhead in system performances. The area occupation, as we have seen before, depends on FIFO width, so, during the synthesis phase, it is necessary to strike a balance between area occupation and efficiency, on the basis of estimated traffic on the external bus. The area overhead of the bridge, configured as in Figure 2-25, is less than 0.21 mm2. A little complexity increase is introduced into the microprocessor software since a routine is needed to periodically send the required information from the external device (FPGA) through the bridge bus.
Summarizing, the integration of this module in AMBA based embedded architectures is easy and costless, but requires a preliminary study based on sensor's target.

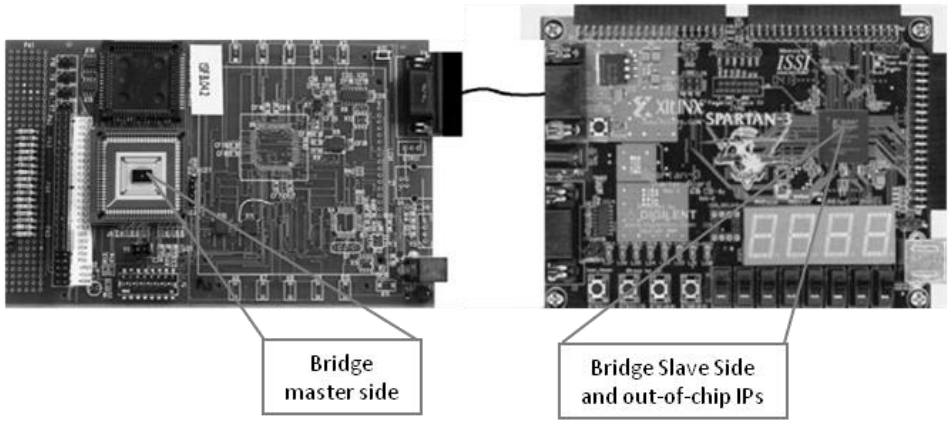


Figure 2-26. *Micromirror ASIC and FPGA connection through bridge*

2.3 Conclusions

In the first part of this chapter, a fast-developing and low-cost test and characterization environment for MEMS and MOEMS has been presented. The system exploits the ISIF a new platform for sensor interface development. The ISIF together with a High Voltage hardware section and a graphical user interface has allowed us to create a laboratory test environment which can easily and rapidly explore different types of architecture for MEMS and MOEMS characterization and test.

As a case of study for our environment, the characterization and test of a double axis resonating micromirror has been performed. After the static and dynamic capacitance measurements performed with an open loop driving approach, a closed loop control driving has been developed. In order to validate this type of approach and to develop an electrical model of the micromirror, the results of the measurements have been compared with results of several electrical simulations. The comparison between the simulations results and the test result are reported in Table III.

On the other hand, the second part of the chapter presented a solution for interfacing AMBA-based architectures of smart MEMS sensors with an external device, solving the problems concerning low-frequency bus constraints, mixed signal substrate noise vulnerability and limited number of pins. The proposed bridge exploits bi-synchronous FIFOs to scale the clock and re-maps the AMBA AHB protocol on a reduced off-chip bus. In this way, it is possible to communicate with all the IPs synthesized on the external FPGA side implementing a symmetric bridge on it. The modularity of the architecture allows the use of this IP also with external FPGAs containing different kinds of buses, not only AHB.

In general, this bridge could be used into every application that needs clock domain conversion such as switches running at different rates or devices requiring slow peripherals. Test results demonstrate that this module is suitable also for worst operational conditions, so the IP could be used in critical situations typical for sensor applications. The adoption of Gray coding and double FF re-sampling increases the system tolerance to errors and avoids metastability conditions.

The integration of this solution in the ISIF platform completes the lack of this architecture concerning the design of new functionalities that are not provided natively by the platform itself, simply connecting an FPGA to the bridge interface and implementing the new functionality in the external device.

This IP, as part of a full custom analog and digital architecture for a smart automotive sensor based on ARM9 processor, has been implemented also to test the performances and the constraints of the new mixed signal BCD8 technology.

Table III – Comparison between simulation results and test results (slow axis)

Frequency (Hz)	**Measurements 82 fingers (mV)**	**Simulations 82 fingers (mV)**
736	117	128
1472	81	78
2208	28	32
2944	20	17

NOTE: The number of fingers of each comb drive is 82.

3 VERIFICATION TESTING AND CALIBRATION

The verification testing and the calibration of MEMS device are probably two of the most important steps in the developing process of this kind of devices. In fact, the aim of the verification testing is to check the correctness of the design and of the test procedure in laboratory before starting the production phase, and the aim of the second is to improve the performance of the device, correcting the transfer function of the sensor exciting it with an appropriate physical stimulus.
The verification testing is a very challenging activity, because each device has its own pinout, its interface, and its peculiar characteristic, so it requires to set up a custom test environment each time a new device is developed. This reengineering activity entails a growth of costs and time-to-market. So, the reengineering time should be reduced as much as possible, standardizing the applications that interact with the device to provide a common interface that allows to reuse the same tests developed for other products. Some applications are available in the marketplace to achieve this goal, but all of these suffer from different drawbacks, as incompatibility, non concurrent access to the device, absence of libraries to manage laboratory instruments, and so on. The use of this approach, moreover, permits to define the test program directly in the laboratory, reducing the time a test machine is used to debug the test procedure.
About the calibration of the sensor, to achieve this goal very expensive test machines are used. These machines allow to calibrate different pieces contemporarily and to handle the socketing of each piece automatically, in order to calibrate pieces continuously. The cost of this machine can be quantified in terms of cost for the equipment and cost for the usage time. While the first one depends on the machine vendor, the second one is dependent on the allocation time disposed by the MEMS device developer. So, in order to minimize the costs, a solution to avoid the use of the production machine for the debug of the calibration algorithm is desirable.
In this chapter, two solutions to achieve the goal of optimize the verification testing and the calibration in terms of reengineering time and costs are illustrated.
The first paragraph presents DevCom, a framework whose aim is to solve the issues concerning the verification phase. DevCom is a client-server architecture that allows the communication between one or more test machine (e.g. a PC) with a MEMS device that communicates using a digital communication protocol. This architecture provides a common interface that permits to implement application with any kind of programming languages and tools under platform that support the Microsoft .NET framework. Moreover, it permits to access the device also from a different machine than the one physically connected to the MEMS, using the client API (Application Programming Interface). The server is split into three layers to allow the extension of the framework in a second time with add-on concerning the low level communication layer (hardware abstraction layer, HAL) and the layer that implements the rule about the protocol itself. The paragraph starts with an overview of the state of the art, then it describes the architecture, starting from an overall description that explains the adopted solution for the communication between the client and the server, and then it is described in detail the characteristic and the solution adopted to implement both the client and the server. Finally, the hardware

and low level software layer used for the communication with SensorDynamics' inertial sensors is described.
The second paragraph describes CaLVal, a complete low-cost and flexible calibration environment composed by a hardware structure to stimulate the piece under test (PUT) with movements and to connect the PUT with the PC, and a software architecture, developed using NI LabView 8.2 to control all the hardware components, to communicate with the PUT and to elaborate sampled data in order to obtain useful information. In this paragraphs each part of the architecture is described, and in the last part it is illustrated a case study concerning how we use the base architecture to develop a complete calibration and evaluation environment for a 3D gyroscope produced by SensorDynamics AG.

3.1 Universal communication framework: DevCom

3.1.1 State of the art

Data acquisition for testing of inertial sensors is a challenging procedure, because it requires complex setups [3][41]. With the introduction of mixed-signal MEMS sensor systems, splitting the acquisition phase and the elaboration of data is not necessary anymore, because this kind of systems provides a digital interface to communicate with the outer world. This new kind of devices, however, introduces other issues concerning the availability of different digital protocols and different languages and applications for the implementation of the test procedures. Indeed, depending on the communication protocol implemented in the MEMS device, it is necessary to develop a set of drivers that permits the interaction between the test machine and the MEMS itself. Moreover, different software applications for supporting test activities and programming languages for the developing of custom test procedures are available in the marketplace, but they are usually not compatible each other. For example, a typical application used for the implementation of test procedures is LabView by National Instruments, because it provides a set of library to control different kind of instruments. The problem of this development environment is that it is not fully compatible with other programming languages, (i.e. python), so customizing the test procedure with external features is almost impossible. Moreover, LabView does not permit to access to the device from different applications concurrently, and this is a strong drawback during the evaluation of a MEMS, because it is very useful to change some internal parameters during data acquisition to check how the system reacts to changes in real-time. On the other hand, programming languages allow to manage concurrent access to the device and other advanced features, but they do not offer common library for the implementation of communication protocols and for the control of laboratory instruments.
Other than issues concerning the testing and evaluation of a MEMS device, another critical aspect of the life-cycle of a chip is customer support. Recall a piece to verify the reasons of a possible malfunctioning is an expensive procedure in terms of costs, time and reputation. So, the possibility to remotely investigate the malfunctioning is really interesting.
Some solutions are available in literature. Reference [42] illustrates a multi-sensor acquisition system, capable of acquiring data and decoding digital protocols from many different electronic controlled systems, but it is an ad-hoc solution that cannot

be easily adapted to different environments and scenarios; moreover, the provided software allows only the collection of data, whereas their elaboration is demanded to other software. Reference [43] describes an Automated Test Station that allows an accurate measurement of gyroscope characteristic, but, as the previous, it requires a well-defined hardware to work, so it is expensive and it can be used only for limited applications. Reference [44] presents a data acquisition system developed using NI LabView for BioMEMS, but it is also adaptable for generic MEMS. The limitation of this architecture is the adoption of a proprietary development environment that increases costs and limits its flexibility and adaptability.

3.1.2 Architecture

3.1.2.1 Overall description

DevCom is a client-server architecture based on Microsoft .NET framework 3.5; particularly it uses the Windows Communication Foundation (WCF) API to manage the communication between the server and the set of clients that desire to interact with it. Figure 3-1 shows a block diagram of the whole architecture.

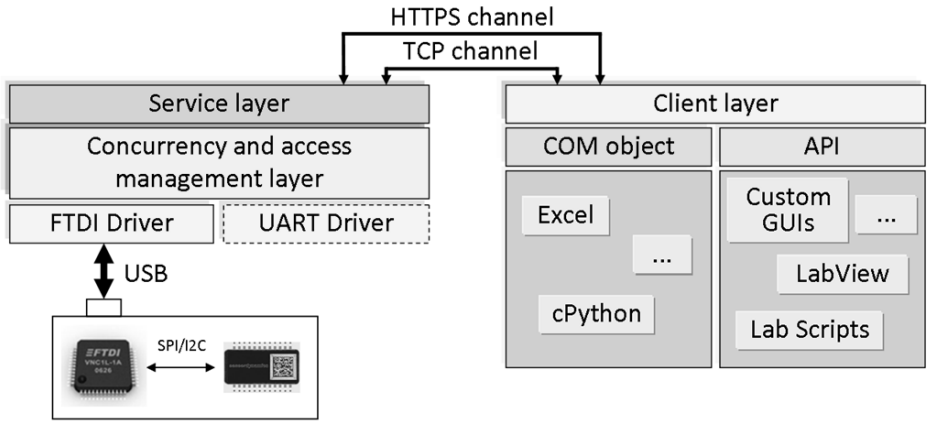


Figure 3-1. *Block diagram of DevCom architecture*

DevCom has been developed to allow the communication between a generic device and a generic application. To do so, it has been split in layers to permit the introduction of new add-in (concerning the driver layer) for the addition of new communication protocols. In this paper it will be described only the FTDI driver (described in section III), because it is used to implement the SPI and I2C protocols used for the communication with the SensorDynamics (SD) devices. However, as an add-in, it has been also developed a SW partition for the implementation of a custom UART protocol.

About the communication between server and client, the WCF API offers different solutions [45]. Among them, we select to provide two different bindings (a binding is a consistent, canned set of choices regarding the transport protocol, message

encoding, communication pattern, reliability, security, transaction propagation, and interoperability): the NetTcpBinding and the WSHttpBinding. We choose to support two protocols because they offer complementary functionality that will cover all possible scenarios. In fact, the first one is a binary-encoded protocol, developed over TCP, so it is faster than other HTTP based protocol, but it is not a standard protocol, so it is almost impossible to implement a non-WCF version of this one. The second one, instead, is a text-encoded protocol, developed over HTTP, so it is slower that the previous, but it is based on the W3C Web Service standard [46], so it can interoperate also with non-WCF clients.
Apparently the second binding seems to be useless in our context, because DevCom itself provides a client API for the interconnection. Actually it can be used to communicate with the server without installing the client API, but simply using the browser. Moreover, using the TCP binding it is impossible to access the server outside an intranet.
The communication using the WSHttpBinding provides also an authentication phase based on the exchange of certificates, and uses the HTTPS secure communication protocol to encrypt all communication on the channel and to provide for integrity and privacy. The NetTcpBinding, instead, does not provide any kind of security and authentication protocol, because we suppose that, inside an intranet, every terminal is authorized to access to the server, and no encryption is provided for the same reason and, what's more, this procedure slows down the transmission.

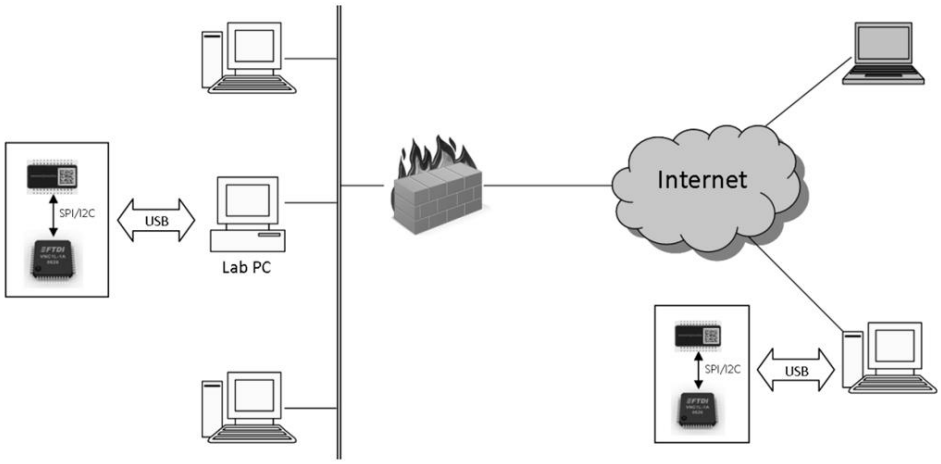


Figure 3-2. *Hypothetical scenario of DevCom architecture*

Figure 3-2 shows a hypothetical scenario in which DevCom could work. For instance, a terminal in an office can access to a lab PC using the TCP binding to execute a test in a machine inside the lab. Another use case could be that a seller wants to show a customer how the device works, and so he can connect his

terminal to the lab PC using the web service protocol. Another case could be the connection to a customer computer to provide support in case of malfunctioning. Finally, a generic case is when a user want to access to a generic PC where the DevCom server is running.

3.1.2.2 Server

The server application is the real core of the framework, because it manages the concurrency among different clients, it incorporates the rule to access to the device and implements the server-side endpoints to expose the application to the net.
It is split in three layers: the first layer deals with the communication between the hardware and the framework; the second layer provides the concurrency and access management; the last layer implements the endpoints that listen to requests for connection from the clients and, moreover, offers a configuration interface to the user of the machine where the server is running.
About the first layer (called Driver layer), it is a library (or a set of libraries, if more than one are installed) that interprets the requests from the upper layer and convert them to commands for the hardware and, if necessary, returns data. Moreover, this layer deals with all the configuration operations for the connected device. Concerning this layer, it is possible to develop new libraries to support different kinds of hardware and protocol, and install them as a plug-in of the application. In fact, when the server application is started, it searches for installed libraries and make them available for interaction.
The driver layer is composed by a 32 bit addressable register bank for the configuration of the driver and a set of methods to write and read the register bank and to send/receive data to/from the device. The use of the register bank has been necessary because we do not have an a priori knowledge of which parameters must be configured for the reasons explained before. So, the meaning of each register is assigned by the developer during the implementation of the driver library. For example, in an implementation of a UART driver, register 0 could be the baud rate, register 1 the parity, and so on. Speaking about the communication, two kinds of approach are provided: immediate and queued. An immediate operation simply executes the requested operation as well as the request arrives; on the other hand, when a queued operation is requested, it is not executed immediately, but the information it contains is stored in a queue: when the *executeQueue* operation is requested, all the operation in the queue are executed sequentially. Obviously, also commands to manage the queue are provided. The queued approach has been introduced to reduce the delay among different requests for a command, to support atomic set of operation from a single client. In fact, during a sequence of immediate operations, a second client could request for an operation that could be executed between two of the operations requested by the first client.
The aim of the concurrency and access management layer is to guarantee the coherence among different call from different clients and to manage the access to the device. For example, if a client is running a long operation, this layer queues requests from other clients until the running operation is terminated. Moreover, it introduces a simple access control to permit a client to have the exclusive control of the writing operation. The access rights are shown in the following table:

Table IV – Device access right

	Read	Write	WrEx
Read	X	X	X
Write	X	X	
WrEx	X		

To implement the access right control engine, an access right descriptor is provided for both the clients and the device. The device has the more restrictive right among the connected device. When a client opens a device, the requested right is compared with the current access right of the device (if already opened): if the check fails an exception is thrown, otherwise the client is connected to the device. Keep track of the access right of the clients is necessary because, if a client with the more restrictive access right disconnects, the device right must be updated with the new restrictive access right.
The last layer provides all communication features and an interface for the configuration of the server. It is the running application and it appears as an icon in the tray bar. Right clicking on it, it is possible to disable the server, to allow local connection only, to execute the server at the startup and to see which clients are connected to it and which devices are connected to the machine.

3.1.2.3 Client

The client is actually a library that is installed using two different mechanisms for interoperability: the Global Assembly Cache (GAC) that is a .NET assemblies cache for Microsoft's CLR platform and the COM interface that is the old approach adopted by Microsoft for the interoperability.
The first approach has been introduced because it is faster than the other, but it has the drawback that only .NET application can use the application stored on it. So, it has been introduced also the second approach, because it is supported by almost all applications and programming languages developed for Microsoft Windows operative system.
The aim of the client library is, at first, the managing of the connection handshake, i.e. the protocol selection (TCP or HTTPS), address and port selection and the creation of a secure connection (if necessary). Moreover, it manages server exceptions and connection problems and errors. About the interface, it provides the same as the server, so it is possible to trigger commands about the configuration and the communication with the device. In addition, methods to check the status of the connection and to create or close it are also provided.

3.1.3 DevCom for SD sensors

3.1.3.1 SD74x series inertial sensors

DevCom has been initially developed to interact with the SD74x series of 2D/3D/6D inertial sensors produced by SensorDynamics AG. These chips are QFN SoP

composed by a MEMS sensor and a mixed-signal ASIC for the conditioning of the sensor itself and for the communication with the outside world. The supported communication protocol are SPI and I2C, and it has been implemented a custom data-link protocol over them. The configuration of the ASIC is done through a 128 byte register bank: writing appropriate data on it, it is possible to configure gains, offsets, phases, and so on. The output of the system is also transmitted using the same approach: setting an opportune flag it is possible to mask a portion of the register bank, replacing that part with a set of 16 bit register that contains the output.

3.1.3.2 FTDI driver

The driver developed for the communication with the SD74x sensors are based on the FT2232H Dual High Speed USB to Multipurpose UART/FIFO IC by FTDI Ltd. This device has the capability of being configured in a variety of industry standard serial or parallel interfaces, and for some synchronous serial protocol (JTAG, SPI and I2C) it provides a Dual Multi-Protocol Synchronous Serial Engine (MPSSE) to simplify the design. Moreover, it can be used to control GPIO pins. Figure 3-3 shows the board it has been developed to connect the FT2232H device and the sensor. As we can see, two sockets are provided, because the prototype was packaged in CLCC packages.

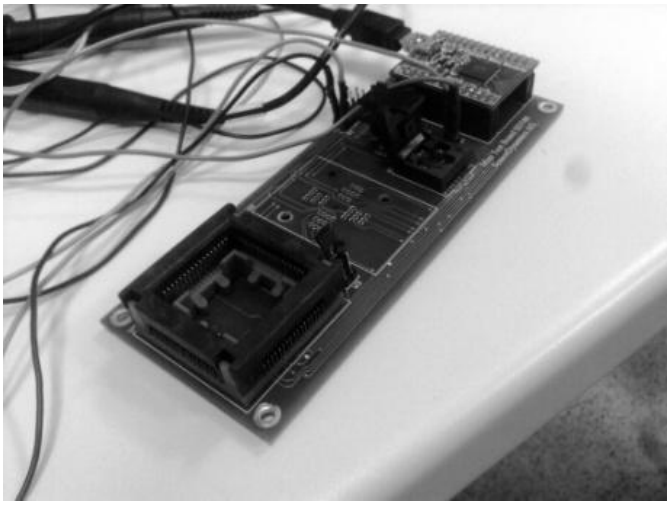


Figure 3-3. *Board for SD sensor communication*

The interface used to control the FT2232H chip is based on buffered commands constructed as a combination of CODE/VALUE(s) ASCII strings. Each command is stored in an internal buffer and then they are executed sequentially.
The DevCom FTDI driver implements the data-link protocol, constructing the frame on the basis of the request it receives and then it converts the frame in commands for the FT2232H chip. The develop of queued commands has been simplified by the buffered approach of the FTDI chip: in fact, when an operation is requested by

the upper layer, the driver layer simply convert it in the appropriate set of commands for the FT2232H and then it saves them in a character buffer. When the *executeQueue* operation is requested, the whole character buffer is sent to the FT2232H internal buffer.

3.1.3.3 Applications

As an integration of the DevCom framework, a set of additions has been designed to facilitate the developing with some of the most common applications and programming languages:

- **LabView 7.1.1 library of Virtual Instruments**: using the .NET support provided by LabView starting from the version 7.1.1, we develop a set of VIs whose aim is to create an instance of the client class, manage the connection and supply the configuration and communication commands to LabView users;
- **Excel 2003 spreadsheet**: a Visual Basic for Application (VBA) example has been created to demonstrate how to use the COM object provided by DevCom inside the Microsoft Office suite;
- **Python 2.6 wrapper**: a wrapper class that uses the COM object has been developed to provide an easy-to-use interface for this programming language;
- **.NET applications suite**: a set of application has been also developed to support the most usual operations (collect and plot data, configure the regbank, and so on).

3.2 Low-cost architecture for the calibration and evaluation of IMSS for small and medium volumes production: CaLVal

3.2.1 Overall description

CaLVal (Calibration and Valuation using LabView™) is composed by a hardware structure to stimulate the piece under test (PUT) with movements and to connect the PUT with the PC, and a software architecture to control all the hardware components, to communicate with the PUT and to elaborate sampled data in order to obtain useful information.

The aim of this architecture is to offer a low-cost and flexible system for the calibration and evaluation of inertial measurement units, providing a tool usable also in small and medium production processes.

Moreover, another target was to design a portable hardware, so that it could be possible to insert it in other instruments like a oven.

Speaking about low-cost, obviously it is intended as “as lower as possible”, and in its estimation it is included also costs in trainings. So, considering that LabView™ is the most popular IDE for electronic system control, we choose to use it for the implementation of the software. About flexibility, the developer must be able to introduce new components without being forced to modify parts of the base software architecture. LabView™ is not the best environment for this kind of purposes, but we have organized the software in layers, so only minor modifications are required to integrate new elements in that.

3.2.2 Hardware design

The hardware must be able to provide the required power to the chip, to stimulate the sensor along all three axes and to connect the communication bus to the PC.
The core of the hardware architecture is a pair of Megatorque® Motor System by NSK, positioned orthogonally as shown in Figure 3-4. Each motor can rotate up to 10 s^{-1} and accelerate up to 800s^{-2}.
Motors are controlled by a UART protocol through a EDC driver unit. Several commands are available, so it is possible to set the angular velocity, the acceleration, the target position and so on. Moreover, it is possible to generate movements also using predefined or custom patterns (sinusoids, saw tooth, and so on). As shown in Figure 3-4, aligning motors orthogonally permits to generate movements to stimulate all three axes.

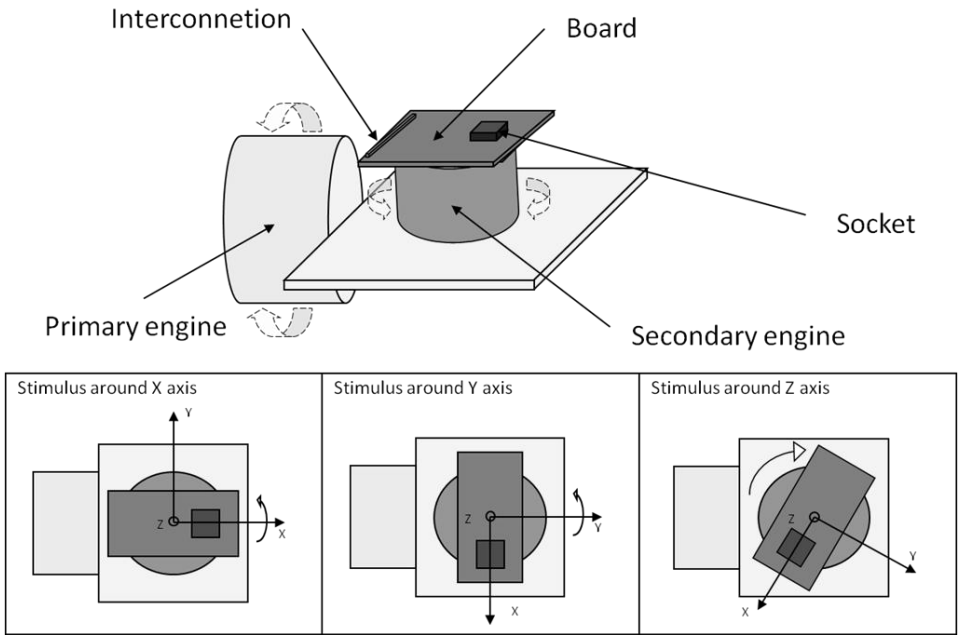


Figure 3-4. *Motors movements in CaLVal*

To provide the power, a Agilent E3646A has been used. It is a cheap DC power supply able to provide up to 20 Volts and 60 Watts, and it is controllable using GPIB interface.
To implement the digital communication protocol, it has been used the NI PCI-6220, a PCI board by National Instruments that is able to generate and capture analog and digital signals, with frequency up to 1MHz for synchronized signals. We choose this board because it is a good trade-off between cost and capability; moreover, it is controllable using the NI-DAQmx driver software provided together with LabView™.

To connect the bus and other useful signals to the PUT it has been designed a PCB that can be easily fasten over the secondary engine base. A block diagram of the board is shown in Figure 3-5.
Because the PCI-6220 produces 5 Volts signals and most of the chips works at 3.3 Volts, we equipped the board with a set of level translator to generate the signal to/from one of the digital buses. We provided the board also with a power regulator to generate the power; it was not really necessary, because the required power is generated by a power supply, but it permits to suppress any noises from the power line due to long lines or electromagnetic fields in the lab.
Because it is possible to calibrate also temperature sensor that might be included in the chip, we equipped the board with the Analog Devices ADT7301, a 13-bit temperature sensor connected to the SPI bus.
Four lines are dedicated for the implementation of the SPI bus, but a group of other ten bi-directional lines are available for the implementation of other kinds of digital buses (I2C, UART, and so on). By means of a socket in the common board, it is possible to connect a custom board, designed to map the signals to the chip's socket correctly and to insert custom components, if necessary. Figure 3-5 shows the block diagram of the custom board for the SD74x sensor family (see paragraph 3.2.4) that included a step-up regulator used to generate the required voltage for OTP programming.
Finally, the common board is equipped with a 32-pin connector; through a flat cable it is connected to another PCB whose purpose is to connect each signal to the right instrument.

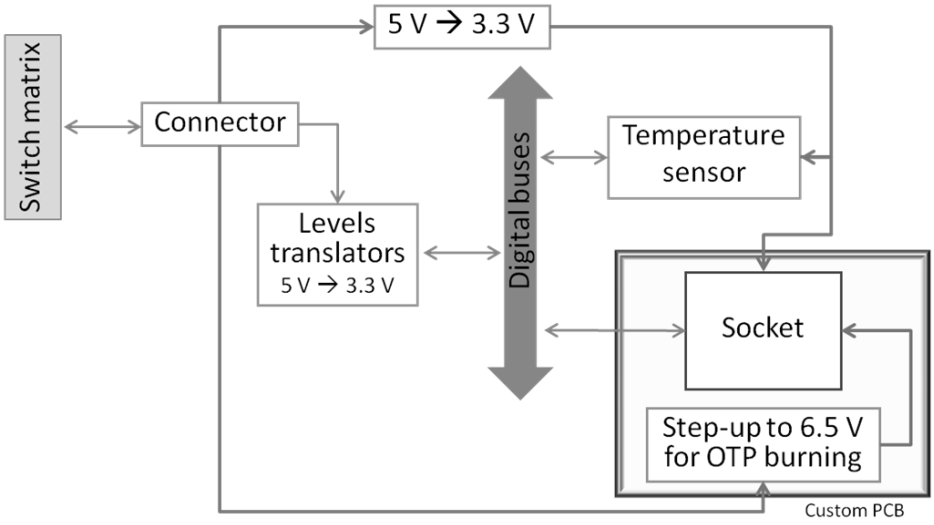


Figure 3-5. *Board block diagram*

3.2.3 Software design

The developing of a software that permits to control all instruments and machine, to communicate with different digital protocols, including custom implementations of standard ones, and that is enough flexible to integrate custom components and implement ad-hoc test procedures was very challenging. Moreover, the software should be easy to extend by the most part of people that work in the electronic field. So, NI LabView™ was an inevitable choice, because we suppose that most of the people know this IDE or, however, implementing a test procedure as blocks and signals is easier than using procedural languages for most of them.
This IDE, however, introduced a lot of difficulties concerning the requirements of flexibility and expandability.
To supply this kind of difficulties, we chose to split the architecture in layers, so that it is possible to insert new components without being compelled to change other parts of the system. Figure 3-6 shows the block diagram of the whole software.
In next paragraphs each layer will be described in detail.

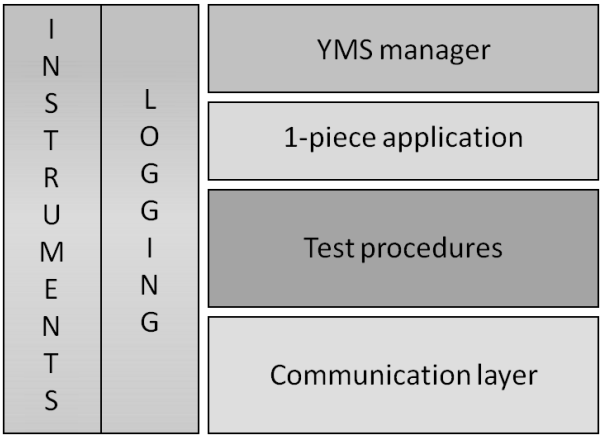


Figure 3-6. *Layered software composition*

3.2.3.1 Instruments

This common component is a set of libraries that can be used by all layers. It comprises a group of VIs, one for each instrument that is used, or will be used by the architecture. In the base version of the architecture, it includes a VI for the control of the motors, one for the power supply and one for the temperature sensor.
The VI for the motors permits to modify the status of each motor separately. It includes command to initialize the device, to set the angular speed and acceleration and to modify the current position; moreover, it includes a command to detect the current position, in order to verify if the target position has been reached (it is useful when data are sampled during the movement).
The VI for the power supply permits to control the E3646A. It provides command to initialize the device, to set current and voltage for both channels and to turn on/off

the power. By now, only a subset of the commands offered by the Agilent power supply are implemented, but it is possible to modify this VI, if necessary, without compromise the correct working of the rest of the application.
The VI for the temperature sensor uses the implementation of the SPI protocol in the communication layer (see paragraph 3.2.3.2). It simply permits to read the current temperature from the sensor, in order to use it as a reference value during the calibration of parts of the chip that are temperature-dependent.

3.2.3.2 Communication layer

The role of this layer is to provide the low level drivers for the communication with the PCI-6220 board. It implements different digital protocols (SPI, I2C, UART, and so on), and it is designed to be expanded with other custom protocols. To permit the insertion of new protocols, it itself has been split in two sub-layers: a lower layer that implements the frames in terms of bits to be sent in the digital lines and a higher layer that provides an interface to the upper layer in the stack of the entire software. Concerning the lower layer, for example, in the SPI protocol it describes the data should be sent in MOSI, CSN and SCL lines. Moreover, it captures the data from the MISO lines and returns the read data. The upper layer, instead, provides some mechanisms to abstract the physical communication. Two interfaces are provided: normal and queued request.
About the normal request, it is possible to query a single read or write operation, passing the payload (that is chip dependent) to the lower layer, and getting back the data read.
The queued request, instead, permits to insert different operations (read, write) in a queue, so that they can be sent to the physical layer one behind the other, without the delay due to the software elaboration (preparation of the payload, construction of the frame, sending to the bus, reading back the data, construction of the result). So, this layer implements three commands to manage the communication: *Initialize queue* that empties the queue and reinitialize internal signals; *insert command* that enqueue the specified operation; *send queue* that prepare the data to be sent to the lower layer, send them to it, receives the results and reconstructs the data depending on the queue it has been sent.
The queued requests is useful when it is necessary to sample data as fast as possible: for example, to collect data from the sensor channels to compute the Fourier transform.
The normal request, on the other hand, is useful when a single operation must be done, because it requires a single operation instead of three.

3.2.3.3 Test procedures

This layer is product-dependent: here the procedures necessary to calibrate and evaluate the chip, depending on the product, are implemented. However, there are defined some standard practice to implement a procedure. In fact, for each product there is the possibility (and it is suggested) to define some configuration files containing parameterized information (for example, the desired sensitivity) and YMS limits (it is mandatory, see paragraph 3.2.3.5). These files are stored in a well-defined path. To access to the main configuration file, a VI has been created: passing the project name, the procedure label and the parameter name it returns the written value.

It is a good practice to keep the output produced by every test procedure standard. To achieve it, the test procedure finite state machine (FSM) must be boxed in a standard template, as shown in Figure 3-7.

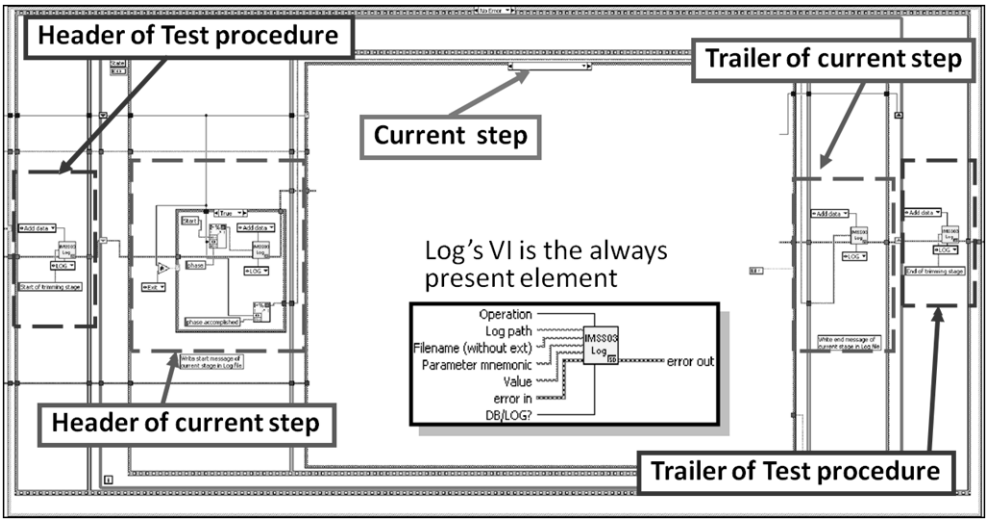


Figure 3-7. *Test procedure template*

The template uses the logging VI to indicate in the log file (see paragraph 3.2.3.5) information about the progress of the procedure itself. So, it has been defined a pair of header/trailer sections, each one indicating the test procedure it is running and the current step.
After creating the new test procedure, it is necessary to specify the project it is related to and a test code in the application configuration file (see paragraph 3.2.3.4).

3.2.3.4 Single piece application

This layer implements a GUI to execute a procedure in a single piece. This GUI has been implemented to avoid the generation of log files related to the YMS (paragraph 3.2.3.5), so it is used during a debug session of the test procedure itself, or during the engineering phase, when the chip behavior is analyzed and the default values is being defined.
The GUI allows to select the project name, the test code (each test procedure has a code associated to it) and other information related to the location and chip identification. Starting from these information, when the application is executed it runs the selected test procedure, using the "Call By Reference Node" internal VI, passing the right path as argument. The path associated to a test code is stored in a application configuration file. This file is a set of tuple *<testcode, path>*, grouped by project codes. So, when the application is launched, it search for the testcode specified in the GUI by the user, in the group of paths related to the selected

project, and then it opens the extracted VI and executes it. Obviously, the insertion of the right path in this configuration file, when a new test procedure is implemented, is a test procedure developer's job.
Concluding, also the GUI's controls of the front panel are filled dynamically when the application is launched, on the basis of the keys in the configuration file.
The appearance of the GUI is similar to the user interface provided by the YMS, shown in Figure 3-8.

3.2.3.5 Logging and Yield Management System

These are two separated portions of the software, but they are included in the same paragraph because the scope of both of them is to create files to store information about the PUT.
The logging component is simply a VI, but due to its importance it has been classified as a separate component. This VI permits to create two kinds of file: a DB file and a LOG file. Each of them is composed by a header section and a content section. The header file is the same for both files, and contains information about the current test procedure, as the project name, the code of the test procedure, the current date and time and so on. It is filled by the single piece application (paragraph 3.2.3.4), but the test procedure itself can add other entries in that portion of the file (i.e. ID information, see paragraph 3.2.4).
The content section of the DB file is composed by a set of <*parameter : value*> tuples: each tuple reports the computed value for each parameter it has been calibrated/evaluated. The LOG file content section, instead, is a superset of the content section of the DB file. In fact, it contains both the tuples and other information the developer of the test procedure desired to log on file.
So, the VI receives as inputs the file type (DB or LOG), the two field data (parameter and value) for DB file, or the value only for LOG file, and the section (header or content). The VI keeps the inserted value during the entire execution of a test procedure, so it is possible to add elements in every part of the application. When the test procedure starts, it is necessary to specify the file name and the path, and this task is done by the single piece application. Likewise, the application executes the finalization of the logging VI, saving the produced data at the end of the test procedure.
The Yield Management System is the last layer of the stack: its role is to collect data from different pieces, estimating the percentage of good pieces and discarding the failed ones. The front panel provided by this VI is shown in Figure 3-8. It is similar to the front panel of the single piece application: in fact, it extends the single piece application, producing other information on the basis of the DB file.
The aim of the YMS is to compare the parameters extracted during the test procedure with some thresholds defined by the test engineer. In details, for each parameter four thresholds are defined: one range is used to detect if a parameter is in spec or not, and the other range is used to distinguish parameters that are not in spec but are acceptable for other uses. To do so, a limit file for each project is defined. The limit file is a XML file containing a table for each test procedure; each table contains the four thresholds for each parameter, and for each range a BIN number that identifies the tray where to place the piece if the range is not satisfied (this information is useful when a pick and place handler is used). In fact, when a parameter fails, the YMS interrupts the execution without parsing other parameters

and the piece is located in the specified tray. If all parameters are in spec, the piece is located in the default tray (zero). An example of limit table is shown in Table V.

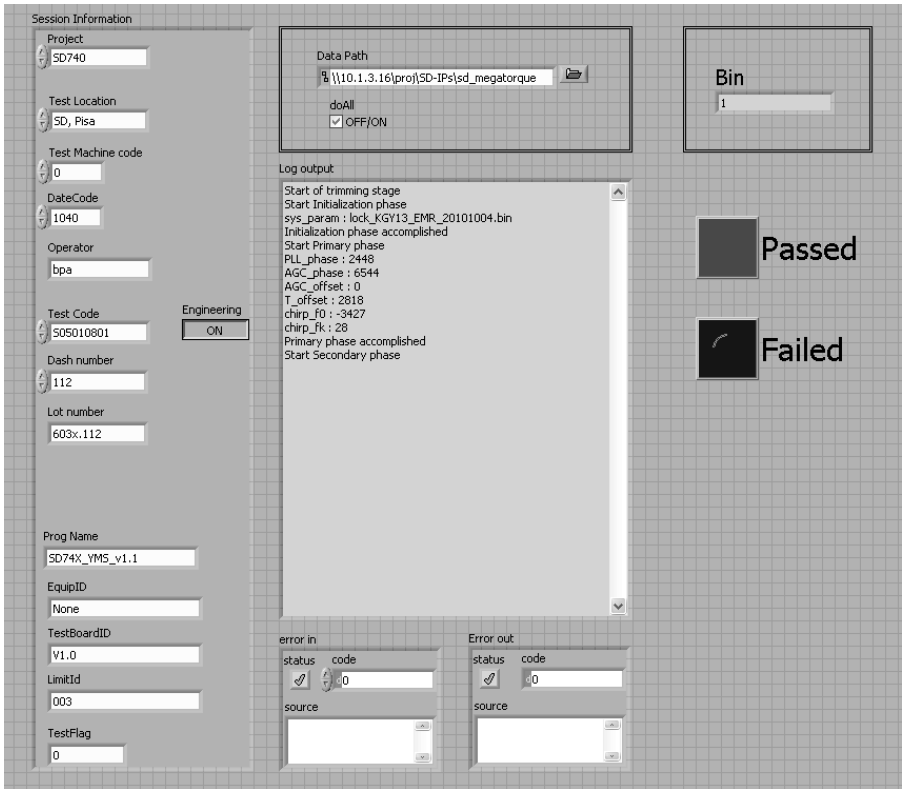


Figure 3-8. *Front-end of the YMS manager*

The generated DLOG file is composed by three section: a global header section, a limit section and a piece section. The global header section contains information about the YMS session, i.e. date, time, project, test code, and so on. The limit section quotes all defined limits for the specified project and test procedure. The piece section contains information about the single piece has been tested; it is split in two parts too: a header part containing information about the piece itself, and a content part containing, for each defined parameter's limit, if it is failed or passed, and a overall result that indicated if a parameter has failed.

3.2.4 Case study: SD740

CaLVal architecture has been used to calibrate and evaluate the SD74x series inertial sensors already described in paragraph 3.1.3.1. These products communicate using both SPI and I2C protocols, so we did not need to create a

new low level VI for the communication. However, another VI has been create to hide the creation of the payload: so, it constructs the payload using the address and data specified by the caller, sends the payload to the SPI communication layer, receives data and returns the read value (if any). Moreover, because these sensors use a register bank of 128 byte registers for the configuration, but the logical registers are not byte-aligned, another VI has been implemented to mask it. This VI receives the mnemonic name of the register as parameter, the operation (read/write) and the data (if a write operation is requested), and extracts, from a text file that has been specified during the initialization phase, where the register is located (start bit) and the length of the register; once these information are available, the VI executes the communication layer's VI one or more times depending on the position of the logical register. For example, if a write operation has been requested in the *regname* register, the start position and the length of the register are extracted (suppose start = 20, length = 5); then the start and the end physical addresses are calculated (start = 2, end = 3); after that, "partial" registers are read, where with partial is intended registers whose bit are only partially written; then the read data are masked, shifted and pasted together to create the new values of the physical registers; finally, new data are physically written. Actually, to avoid the execution of a lot of accesses to the bus by the software that would have introduced delays, an image of the register bank has been created: so, multiple read/write operations does not really act to the bus, but simply modify the image inside the VI. To transfer the new register bank, an upload command must be explicitly called. Obviously, also a download command is provided to align the clone with the original register bank.
The new stack for the SD74x product family is shown in Figure 3-9.

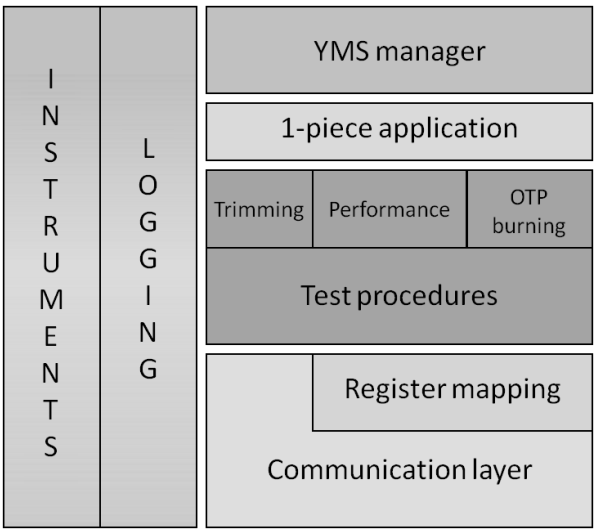


Figure 3-9. *Software layers including SD74x family components*

Using the new communication VI, three test procedure has been created: trimming, performance and OTP burning. The latter test procedure really does not measure anything, but it has been inserted as test procedure because it is an integral part of the production process. The flow charts of the other test procedures have been reported in Figure 3-10.
By now, a detailed description of the algorithms have been implemented is not allowed, but we can describe the secondary step of the trimming test procedure as an example (regarding the 3D gyroscope).

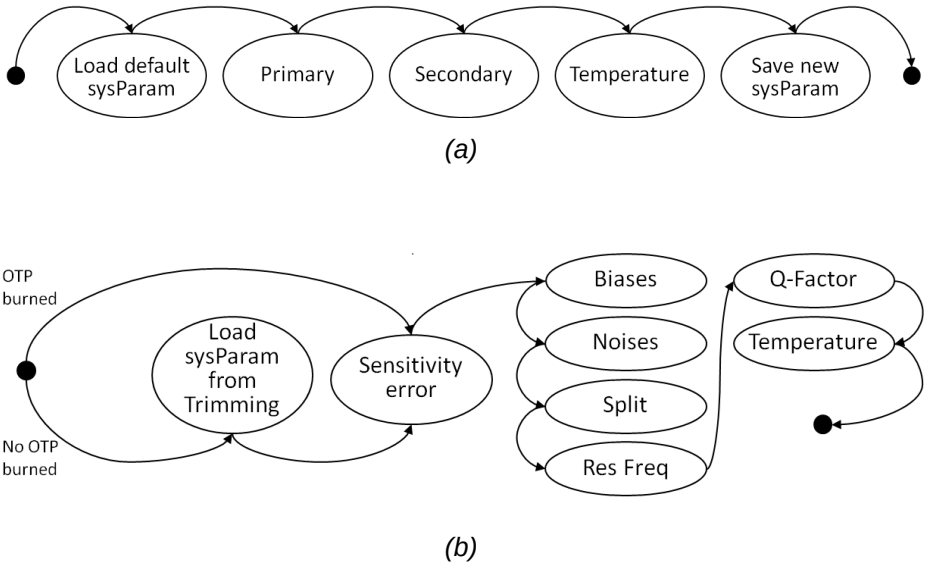


Figure 3-10. *Flow chart of (a) trimming (b) performance*

The aim of this step is to compute the gain of the secondary chain, in order to rotate and scale the data obtained by the MEMS on the basis of the specs. To explain how to obtain it, we need to point out that a cross compensation matrix has been inserted in the chip's secondary chain, in order to allow the combination of the three channels to rotate and scale the gyroscope's output.
The algorithm starts applying positive and negative movements with a well-defined speed (it is a parameter in the project's configuration file) along all the sensible axes and, during the movements, data from all channels are sampled and the mean of the samples extracted for each movement is calculated. A sensitivity matrix is created, using the following formula:

$$Sens\_matrix_i = \frac{\mathrm{mean}(\mathrm{S_i^{pos}}) - \mathrm{mean}(\mathrm{S_i^{neg}})}{2 \cdot \mathrm{speed}}, \mathrm{i} = \mathrm{x,y,z} \tag{14}$$

Where S_i^{pos} and S_i^{neg} are the collection of samples obtained during positive or negative movement along *i* axis, for all channels (the result of the mean is a row vector). For row ≠ column, Sens_matrix[row, column] is the projection of one axis over the others, and it is called *Cross axis.* Once computed the Sens_matrix, to obtain the gain matrix it is necessary to invert it and multiply the Sens_matrix by the desired sensitivity (it is a parameter in the project's configuration file too). Starting from the gain matrix, only a part of the resulting gains are written in the cross-compensation matrix, the remaining part is distributed along the secondary chain's gain stages.
Other than the algorithms themselves, another mechanism has been added in the test procedures for the SD74x family: the ID evaluation and auto-generation. It is a VI that permits to verify if a piece has an ID: if not, a new unique ID is assigned and written in the OTP, otherwise the ID is read from the register bank. This operation is done during the first stage of each test procedure, so each of these also adds the computed ID in the header section of the logging files.
Concluding, also the modification of the configuration files and the creation of limit file has been done: so, the new projects and test procedures are added to the application configuration file, the folder containing the project files is created and a limit table for each test procedure has been defined (an example of this kind of table is shown in the following).

Table V – Example of limit file for trimming stage

ID	Param Name	Meas Unit	Trimming					
			Min1	***Max1***	***Bin2***	***Min2***	***Max2***	***Bin3***
1	Bias	deg/s	-1	1	2	-2	2	3
2	Noise	deg/s	0.1	1	2			
...	...							

3.3 Conclusions

Laboratory activities, both concerning the characterization of inertial mixed-signal sensors and their calibration are critical tasks, because requires a complex engineering work. In fact, it is necessary to develop the low level layer for the communication with them, the test procedures for the calibration and the characterization, a set of software to verify malfunctioning, and so on. These operations, in general, are required every time a new kind of device is produced, since the used technologies could be changed. Moreover, the develop of the calibration algorithm usually requires the use of production machine, entailing high costs connected to the occupation of a complex machine for activities not directly connected to the production.

DevCom has been developed to reduce the engineering time, because it increases the reusability of the produced software, providing a common interface for every devices. This can be possible allowing the addition of new low level drivers to communicate with the ASIC under analysis. So, when a new device is produced, it is necessary only to develop a single portion of the framework and all the software that uses DevCom become compatible with this one. Another characteristic that improves the efficiency of the test and maintenance phases is the possibility to access to the device remotely. This allows to check the status of a device remotely and to change its configuration, both during a test procedure or when the device has been already delivered to customers.
To develop DevCom we choose to use .NET framework 3.5, because it already provides the support for distributed application (WCF API). This has been a risky choice, because it worked only under Microsoft operative systems at the time the architecture has been developed. During this time, however, the team of MONO project [47] has extended the port of .NET framework for Linux systems until version 3.5, so DevCom is now fully portable also on this operative system.
CaLVal, on the other hand, overcomes the problematic of using expensive production machines for the debug of the calibration algorithm proposing a flexible low-cost calibration environment that proposes itself as an alternative equipment for the design of the algorithm. By the use of a low-cost dual axis engine to stimulate the MEMS sensor, and through the use of a flexible software framework developed in LabView, a software development environment typically used by electronic engineer, it is possible to create a simplified automatic test equipment.
In this context, an example of application for the calibration and the evaluation is presented. The target of this application is a 3D MEMS gyroscope developed by SensorDynamics AG. This application uses the CaLVal environment to set up also a minimal yield management system that can be used to estimate the yield of this product.

4 DATA PROCESSING

MEMS sensors are widely used in many consumer and automotive applications, due to their low cost, small size and low power consumption. However, there are also some disadvantages: MEMS sensors suffer from higher errors than other expensive sensors. In particular, gyro drift is a problem in many applications that require a high precision. In an orientation tracking system the problem is that, with the simple integration of the gyro data to obtain the angular position, the drift error is always increasing, so that after some time the information of the angular position is meaningless.
To avoid this problem, some other kind of sensors must be used in addition to gyros, such as an accelerometer or a magnetic compass, so as to obtain an external reference for the angular positioning system.
The introduction of new functionalities [48-51] by the use of sensor fusion algorithms is becoming the new target for the next years in MEMS sensor systems design, as explained in paragraph 1.6. For this reason, in this chapter two different algorithm for the sensor fusion are approached. The first algorithm [52] is actually a preliminary evaluation, based on a fully software algorithm for the sensor fusion that has been used to estimate how much the errors in MEMS sensor data affect the estimation, and if it is possible to suppress them in some manner. The second, instead, is an algorithm that has been designed so that it can be implemented in the sensor interface itself [53]. It is based on a simplified Kalman filter and uses quaternions to represent the angular position.

4.1 A preliminary evaluation using a fully software algorithm

4.1.1 Overview

This paragraph describes a simplified algorithm that combines the information provided by a 3D axis gyroscope and a 3D axis accelerometer. It is realized as a graphical demo for the micro machined integrated gyro and acceleration module produced by SensorDynamics AG [54].
This SoP integrates two high performance sensors (gyroscope and accelerometer), oriented according to two perpendicular axis, and one ASIC. The gyro sensor works on the principle of the *Coriolis Effect* and on a capacity-based sensing system: rotation of the sensor causes a shift in response to an oscillating silicon structure resulting in a capacitance change. The accelerometer is based on a classical spring-mass structure with one degree of freedom: the acceleration on the given direction causes a linear capacitance change. The role of the ASIC is to detect changes in capacitance and transform them into a digital output, which is proportional on angular rate or linear acceleration. Both sensors are continuously monitored by an independent safety survey system.
This demo allows to point out the features of this SoP and to demonstrate how it is possible to elaborate the information produced by this sensor to obtain further complex ones, which could be applicable in several fields. However, as it is shown in the following, it suffers for the influence of noise in the measured data, so it has been impossible to detect the linear position.
The demonstrator consists of a board equipped with three SD modules and a microprocessor connected to them: the running on firmware collects information

from the modules and sends them to a PC, which is running the graphical demo software. Then, this GUI processes the received data for moving a virtual 3D car in according to the board's movement.

4.1.2 The architecture

4.1.2.1 Hardware and Firmware

The main hardware device which characterizes the system is a board equipped with three accelerometer/gyroscope modules and a microcontroller to control them and to manage the information they produce.
The core of the board is the ATMega128 microcontroller by Atmel. The sensors are placed along the three axis as showed in Figure 4-1.
Figure 4-2 shows the block diagram of this board.

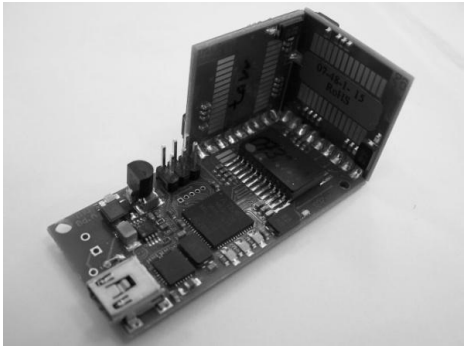


Figure 4-1. *3D view of the board*

They communicate with the microcontroller through a shared SPI bus, and the CS and reset signals are generated using GPIO pins. The core communicates with other devices through a USB port. Since the microcontroller is not equipped with a USB controller, we use an UART to USB Bridge controller by *FTDI chip Ltd* that is able to send data faster than simple RS232. The board is also equipped with an RS485 interface for testing.
The communication between SD sensors and the microcontroller is managed according to a master-slave paradigm where the host always acts as the master and the sensor as a slave. The communication can be thought as based on a per-session mode: the core starts a session by sending a fixed size message to the sensors in turn. The requests are driven periodically using a timer: the microcontroller alternatively requests the values of the rate and of the acceleration to all three sensors.
The communication between the microcontroller and the external device (i.e. PC) is obtained making polling on the UART's *char received register*; when a character is received, the main routine read which data is requested decoding the received char and then sends the requested data.

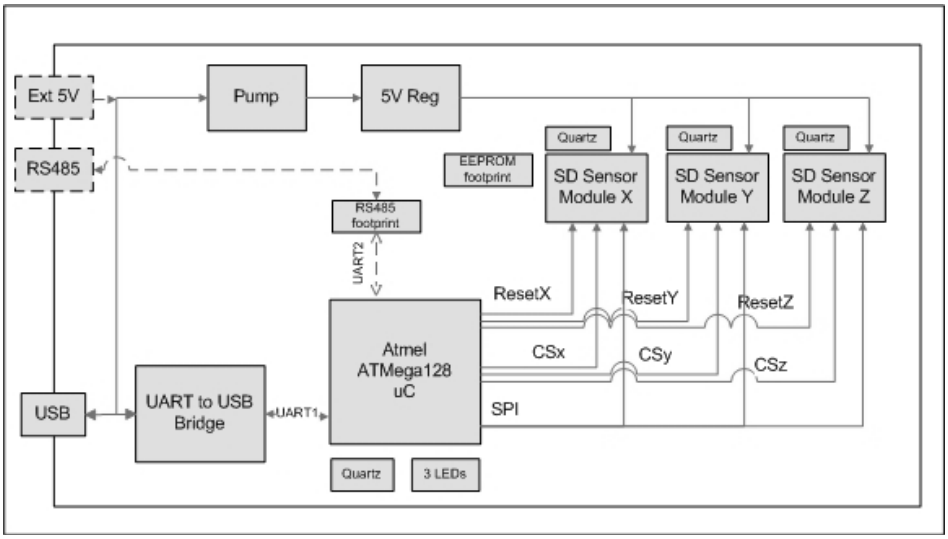


Figure 4-2. *Block diagram of the board*

4.1.2.2 Software

This section describes the algorithms adopted to simulate the rotational and transactional movement of the car in the GUI. For 3D motion representation we have chosen to use the Irrlicht Engine, an open source high performance real-time 3D engine written and usable in C++ [55].
For a simpler explanation we are going to describe the algorithms as sequential; actually, the graphical management and the elaboration block are two different processes.

Rotation algorithm

The first problem to deal with is the representation of the car position. To represent the motion of the model, we use a rotation matrix for keeping track of the position of the car's axis with respect of absolute axis (the horizon axis). This matrix is constructed as follow:

$$R = \begin{bmatrix} ix & iy & iz \\ jx & jy & jz \\ kx & ky & kz \end{bmatrix} \tag{15}$$

where $u = [x, y, z]$ is the position of the absolute reference system, whereas $v = [i, j, k]$ is the position of the car's relative reference system. Each column of the matrix represents the component of one of the three relative axis in respect of each absolute axis.
Starting on a known position of the car at the software's reset (the car model lean on its wheel in a horizontal surface), the algorithm computes the initial offset between the car model and the board mounted in: during the running phase this offset will be removed from the angle's value received by the board. Always at the

reset phase, the software sets the rotation matrix to identity matrix, because the car's relative axis' position agrees with the absolute axis' position.
The algorithm for updating the rotational position is composed by two sections: one of them elaborates the values received by the board; the other one computes the new values for the rotation matrix and updates it.
In detail, the first section removes the offset to compensate for the initial displacement as showed above. Then, it applies a threshold to avoid infinitesimal change of the angular velocity that could cause integration's divergence. Finally, it applies a simple mobile mean filter to integrate the angular velocity obtaining the rotation angles.
The second section, at first, convert the elaborated values from degree to radiant; then it updates the rotation matrix using the following formulas:

$$\begin{aligned} ax &= ax \cdot \cos\theta + bx \cdot \sin\theta \\ ay &= ay \cdot \cos\theta + by \cdot \sin\theta \\ az &= az \cdot \cos\theta + bz \cdot \sin\theta \end{aligned} \tag{16}$$

$$\begin{aligned} bx &= bx \cdot \cos\theta - ax \cdot \sin\theta \\ by &= by \cdot \cos\theta - ay \cdot \sin\theta \\ bz &= bz \cdot \cos\theta - az \cdot \sin\theta \end{aligned} \tag{17}$$

where:

- a = j, b = k for rotation around x axis;
- a = i, b = k for rotation around y axis;
- a = i, b = j for rotation around z axis.

After computing the new values for the elements of the rotation matrix, it is necessary to orthonormalize the rotation matrix, because of approximation errors; in fact, it is possible to have values of the rotation angles which do not satisfy the conditions of orthogonal rotation.
The current position is obtained using the following formulas:

$$\begin{aligned} \theta_x &= \tan^{-1}\frac{-180 \cdot kx}{\pi \cdot ix} \\ \theta_y &= \sin^{-1}\frac{180 \cdot jx}{\pi} \\ \theta_z &= -\tan^{-1}\frac{-180 \cdot jz}{\pi \cdot jy} \end{aligned} \tag{18}$$

Figure 4-3 shows the flowchart concerning the rotation algorithm.
As we described above, when the application starts or the *reset button* is pressed, the offset is estimated and the 3D car is oriented into default position; after that, the application begins the elaboration explained above. The result of the elaboration is

then plotted in the GUI. Then, at the end of the angular position computation, the 3D car is rotated according to the rotation matrix using the graphical engine interface.

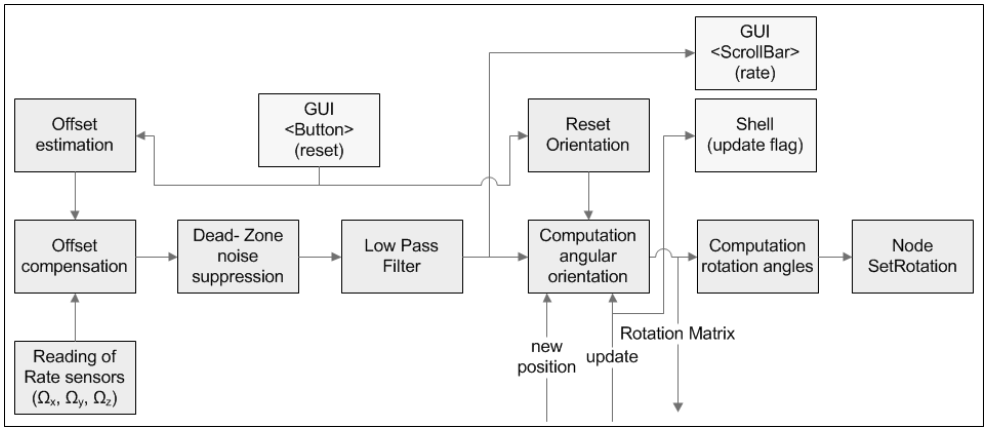


Figure 4-3. *Rotation algorithm flowchart*

Translation algorithm

The translation algorithm is more challenging than rotation algorithm for two reasons: first of all, the accelerometers measure also the gravity, so it is necessary to eliminate this unwanted acceleration to avoid a "fall effect" in the graphical model; secondly, computing the values of the angles from the angular velocity requires single integration, whereas computing position from the acceleration requires a double integration. So, it was necessary to find a trick to eliminate this error.

The solution for the elimination of the gravity effect utilizes the rotation matrix. The gravity, in the absolute reference system, is represented by a vector equal to the opposite of z-versor. This vector is decomposed into relative board's coordinates and the resulting components are subtracted from the measured acceleration's values.

For eliminating the double integration problem is necessary to introduce a constraint in the movement or to add another reference of measurement (i.e. an optical sensor or a GPS device). To avoid this, it was chosen to modify the position of the 3D model following the velocity. So, the model's position vector is equal to the measured velocity vector.

This solution avoids the implementation of a cam that follows the model because, if the graphical model would follow the translation movement, it gets out of the screen almost immediately. The velocity, instead, returns to zero when there is no movements, so the algorithm keeps the model always centered on the screen.

Summarizing, after subtracting the gravity, the algorithm applies a threshold to avoid infinitesimal change of the acceleration that could cause integration's divergence and integrate its output values using a mobile mean low pass filter;

then, it converts the acceleration, oriented according to the angular position of the car, to absolute reference system. Finally, another low pass filter is applied for finely calibration.
Figure 4-4 shows the flowchart concerning the acceleration algorithm.
When the application starts, it subtracts the estimated gravity value and evaluates the right movement according to the algorithm described above, and then it plots the filtered values on the GUI and updates the 3D car position. If the *autocalibration ON/OFF button* is pressed, the cross-compensation is enabled: it allows correcting the angular position periodically, based on gravity orientation estimation, as it is described in the next paragraph.

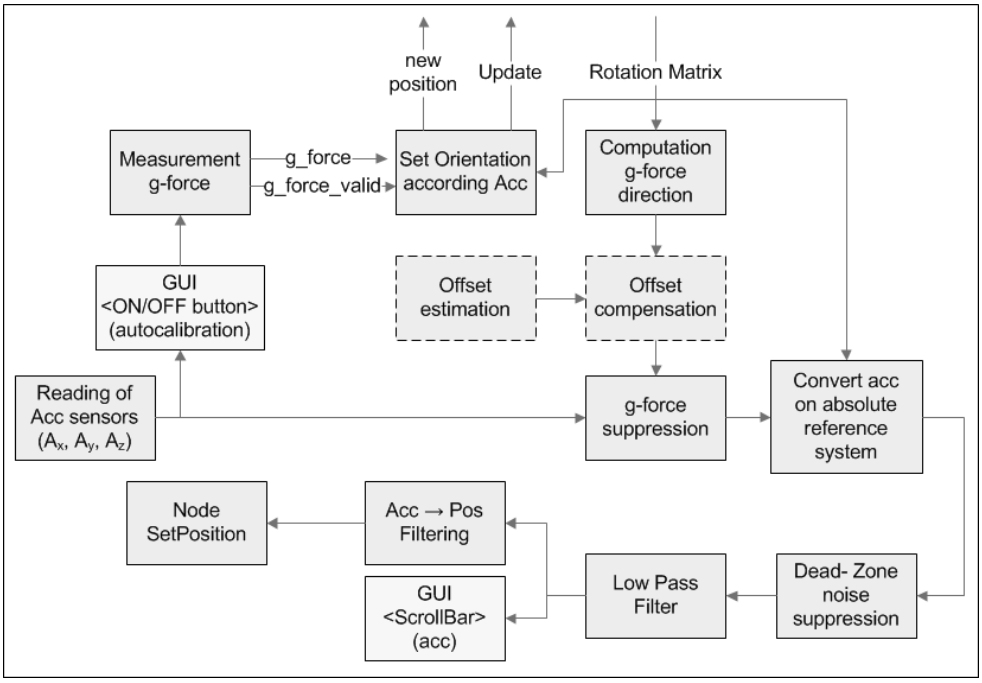


Figure 4-4. *Acceleration algorithm and cross-compensation flowchart*

Cross-compensation algorithm
This algorithm allows correcting angular position using both acceleration and angular velocity information.
It runs periodically, avoiding too much pervasive modification that could compromise the real sampled movement representation. So, when the user doesn't move the car model, the algorithm starts correcting the angular orientation according to gravity estimation. Figure 4-4 shows how this algorithm appears on the block diagram.
The algorithm computes the mean and the variance of the acceleration without subtracting the gravity; then, if the sum of every accelerations variance is lower

than a threshold it means that the car is not moving so it can start updating the rotation matrix. To do this, it normalizes the mean's values according to these formulas to obtain the *z* column of the inverse of the new rotation matrix:

$$R_{new}^{-1}(q,3) = \frac{-G_{mean_q}}{\sqrt{\sum {G_{mean_q}}^2}}, q = 1,2,3 \tag{19}$$

Since the choice of the start vector is unconstrained, it identifies which axis is nearer to the correspondent in the rotation matrix to make a softer update of the matrix:

$$\begin{aligned} scal_i &= \left|\sum R_{old}^{-1}(q,1) \cdot R_{new}^{-1}(q,3)\right| \\ scal_j &= \left|\sum R_{old}^{-1}(q,2) \cdot R_{new}^{-1}(q,3)\right| \end{aligned}, q = 1,2,3 \tag{20}$$

The greater of them determinates which axis will be calibrated next, according to Gram-Schmidt's orthonormalization (*l* in the following formulas).

$$\begin{aligned} R_{new}^{-1}(q,l) &= \sum R_{old}^{-1}(q,l) - scal_q \cdot R_{new}^{-1}(q,3) \\ R_{new}^{-1}(q,l) &= \frac{R_{new}^{-1}(q,l)}{\sqrt{\sum R_{new}^{-1}(q,l)^2}} \end{aligned} \tag{21}$$

Finally it orthonormalize the last axis, completing the matrix, and compute its inverse to obtain the new rotation matrix.

Critical parameters

Controlling and time critical applications, such the one described above, are characterized by several critical parameters. In this demo the parameters that it was necessary to calibrate are the frequency of the timer in the firmware and the ones concerning the dead zone and low pass filters.

The first one is a function of the speed of the SPI bus, because in one timer period the firmware could either process the last received message and prepare the new message to transmission or sending the data on the SPI bus. So, the minimum timer's period must be:

$$T_{timer} > \max((T_{elabRx} + T_{propTx}), (8T_{sendSPI} + 8T_{recvSPI})) \tag{22}$$

The other parameters, instead, are chosen according to visual empirical simulations.

4.2 An integrated sensor fusion algorithm for the orientation tracking

4.2.1 State of the art

Many papers propose sensor fusion algorithms to estimate the angular position. Almost all the papers refer to quaternions to represent it, because the use of quaternions instead of the Euler angles eliminates the problems related with the singularities and the gimbal lock. Different types of Kalman filters and state equations are used to describe the system.
In [56], an Extended Kalman filter is used with a three rate gyro and three accelerometers. The state equation is composed by the quaternion and also by the angular velocity and the gyro drift. The system has a good estimation of roll and pitch angles, and there is also a low correction on yaw angle.
For a better estimation of the yaw angle, a magnetic compass, in addition to the gyro and the accelerometers, is used in [57]. The process is described with the quaternion and the angular velocities, and a linear Kalman filter is developed. In particular, to avoid the use of an Extended Kalman filter, the read equation of the filter is not composed by the data of the MARG sensor (magnetic, angular rate and gravity), but is composed by a quaternion, which is estimated with a Gauss-Newton method from the sensor data. This simplifies the filter design, but implies a quite complicated algorithm of estimation of the quaternion.
In [58] and [59] an adaptive filter is developed for a MARG sensor unit for automotive use. When the sensor is in high acceleration mode, the angular position is calculated mainly with the data of the gyro, updating the quaternion. The accelerometers are used to estimate the roll and pitch angles, and the magnetic compass to estimate the yaw angle. The angle estimation is more influential in the state correction when the system is in non-acceleration mode.
In [60] an Unscented Kalman filter is used instead of the traditional Extended Kalman filter, because the Unscented filter is deemed to be more accurate and less costly to implement. In [61] there is a comparison between the Extended Kalman filter and the Unscented Kalman filter: the resulting precision is found comparable, but the Unscented filter requires much more computation time.
It is noteworthy that all these works refer to discrete sensor systems, where the filter algorithm is processed on a microcontroller or a PC. This work, instead, proposes the design of a simplified Kalman filter and relevant fixed point architecture to be integrated together with the 6D Inertial Measurement Unit (IMU) sensor.

4.2.2 System modelling

Matlab Simulink was used to build a model of our system and to verify the correct operation of the filter. The state equation is composed by the quaternion only, because the addition of other variables does not increases significantly the precision of the angular estimation, but requires a lot more of hardware requirements. The state equation is the following:

$$\dot{q}_n^b = \frac{1}{2}\Omega_{\mathrm{nb}}^{\mathrm{n}} q_n^b \tag{23}$$

Where q_n^b is the quaternion representing the rotation of the body-frame, united to the IMU sensor, respect to the inertial n-frame, and $\Omega_{\mathrm{nb}}^{\mathrm{n}}$ is the rotational matrix, derived from the quaternions properties.

$$\Omega_{\mathrm{nb}}^{\mathrm{n}} = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{24}$$

This matrix is formed by the angular velocities ω_x, ω_y and ω_z measured by the gyro. Equation (14) is a time-continuous equation that can be easily transformed in the time-discrete equation to be used in our system, obtaining A_k matrix that relates the state evolution of the system using only the gyro data. To ensure data integrity, the quaternion must be normalized with a unit norm. A correction equation that uses the data from the accelerometers has to be introduced. From the estimated angular position an estimated gravity vector $\hat{g}$ can be calculated using the direction cosine matrix R_n^b, assuming constant the g-force acceleration |g|:

$$\hat{g} = R_n^b \begin{bmatrix} 0 \\ 0 \\ |g| \end{bmatrix} = |g| \begin{bmatrix} 2q_1q_3 - 2q_0q_2 \\ 2q_0q_1 + 2q_2q_3 \\ q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{25}$$

This estimated gravity vector will be compared with the measured gravity a, which obviously suffers of high errors, because the accelerometers do not measure the gravity only, but also the external accelerations of the system. For this reason, the correction factor must be weighted with the Kalman gain K_k, a coefficient that is calculated from the statistics of the noise covariance matrices of the system.

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1} \tag{26}$$

P_k^- and P_k are the a priori and a posteriori error covariance matrices, R_k and Q_k are the error covariance matrices of the read and the state equation of the filter, assumed constant at each filter iteration, H_k and V_k are the Jacobean matrices of the partial derivatives respect to the quaternion and to the noise, of the nonlinear equation (25), which relates the quaternion to the estimated gravity.

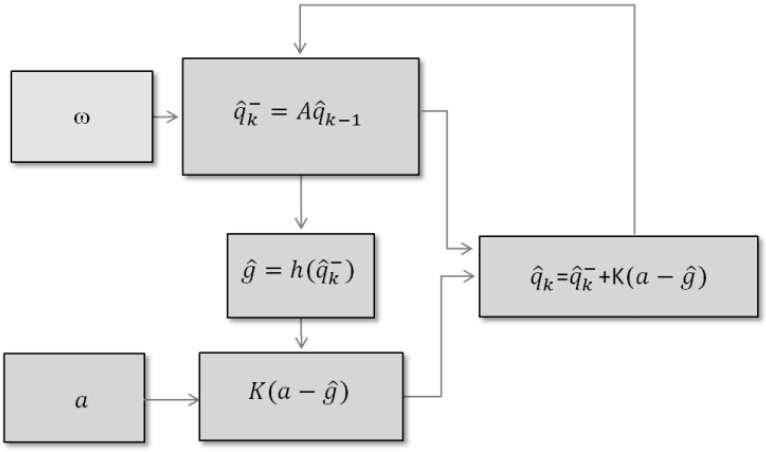


Figure 4-5. *Kalman filter algorithm.*

Before calculating the gain K_k, the a priori error covariance P_k^-, evaluating the error in the state estimation with the state equation only, needs to be calculated:

$$P_k^- = A_k P_{k-1} A_k^T + Q_{k-1} \tag{27}$$

Finally, the a posteriori error covariance matrix P_k that is needed for the following step of the filter is obtained.

$$P_k = (I - K_k H_k) P_k^- \tag{28}$$

In this work, considering that the system may be subject to high external acceleration, the covariance matrix of the measurement noise R has a quite high estimation:

$$\mathrm{R} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \tag{29}$$

After some tests, the following value was found as the best fit for the process noise covariance matrix Q:

$$\mathrm{Q} = 10^{-6} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{30}$$

and the following value was chosen to initialize the starting covariance matrix:

$$P0 = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix} \tag{31}$$

In order to have a better angular estimation, [63] uses a complementary filter on the gyro and accelerometers data. In this work, a dead zone on the gyro data is applied with a limit of 0.01 radiant for second, while a low-pass filter on the accelerometer data will be evaluated in a future work.

4.2.3 Algorithm simplifications

There is no problem to run this kind of filter on a PC, but the goal is to obtain an integrated system, with minimal area complexity and power consumption. For each filter iteration about 600 multiplications, 400 additions, a square root and 13 divisions are required and so some algorithm simplifications with negligible performance degradation were devised.
First, the a priori error covariance P_k^- is approximated with the a posteriori error covariance P_k. This is based on the assumption that the angular position estimation before the correction is approximately the same as after the angular position correction. This is always the case when the frequency of the filter iteration is greater w.r.t. the gyro drift.
Second, a pre-computed value of P_k is used. This is a tuning parameter of the filter that has been calculated offline, from an average of the values assumed by P_k in some runs of the filter. Then, the following value was calculated:

$$\mathrm{P} = \begin{bmatrix} 0.001 & 0.0002 & 0.0002 & 0.0002 \\ 0.0002 & 0.001 & 0.0002 & 0.0002 \\ 0.0002 & 0.0002 & 0.001 & 0.0002 \\ 0.0002 & 0.0002 & 0.0002 & 0.001 \end{bmatrix} \tag{32}$$

To allow a faster evaluation of the initial position, a higher value of P_k is used in the first 50 iterations of the filter.

$$\mathrm{P} = \begin{bmatrix} 0.5 & 0.0003 & 0.0003 & 0.0003 \\ 0.0003 & 0.5 & 0.0003 & 0.0003 \\ 0.0003 & 0.0003 & 0.5 & 0.0003 \\ 0.0003 & 0.0003 & 0.0003 & 0.5 \end{bmatrix} \tag{33}$$

A higher value of P_k implies a higher value of K_k, so that the accelerometers are especially used to calculate the initial position of the system.
Third, because the matrix P_k is pre-computed, the determinant used to invert a matrix in the K_k calculation was considered a constant, so that 9 division operations were eliminated.
Fourth, it was checked that the normalization operation on the quaternion, that involves the calculus of a square root and 4 divisions, can be eliminated when the Kalman filter is working, because the norm of the quaternion is controlled by the filter itself.
Following this simplification, the final amount of operations is estimated as 400 multiplications and 300 additions.

4.2.4 Simulations

The Simulink model was experimentally tested with SensorDynamics Cube Demo sensor, described in [52]. It is composed by three orthogonal SD755 sensors, each integrating a gyro and an accelerometer (see Figure 4-6). Thus, the Cube Demo sensor simulates a 6D integrated sensor, not yet available. The maximum declared gyro bias is ±0.5°/s at 25°C and ±1°/s within the entire temperature range.
The test were carried out with a test equipment specially designed for gyro calibration, which allows to rotate the sensor on two axes with an accuracy of 0.01°; the data were imported in the Matlab workspace and processed with the Simulink model. For clarity, the following results are presented for the *x* axis angle, or roll angle, but it was verified that the results obtained are valid for both roll and pitch angle, whereas for the yaw angle there is no filter correction.

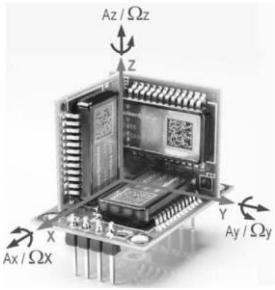


Figure 4-6. *SensorDynamics Cube Demo sensor*

A first test was carried out with the still sensor to verify the influence of the drift on the angular position estimation. With the filter inactive, the drift is quite high, about 30 degree per minute, as shown in Figure 4-7.

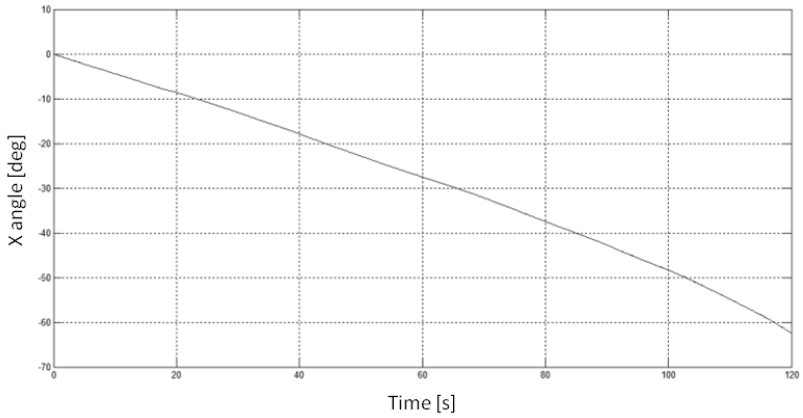


Figure 4-7. *Drift on the x axis*

With the Kalman filter active, the drift is eliminated: Figure 4-8 shows that, after the initial position correction caused by the high starting value of P_k, the angular position is stabilized with a maximum error of 0.7° in a simulation of 120 seconds.

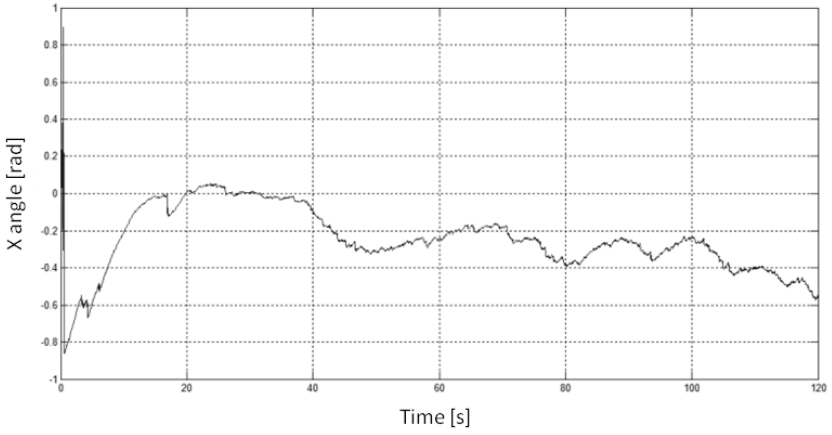


Figure 4-8. *Drift correction on the x axis with the Kalman filter*

We performed a test also to verify the capacity of the system to correctly estimate the initial position of the sensor, even if it is upside down at the startup. The result is very encouraging, because the estimation of the angular position is corrected after 0.5 seconds only, as shown in Figure 4-9.

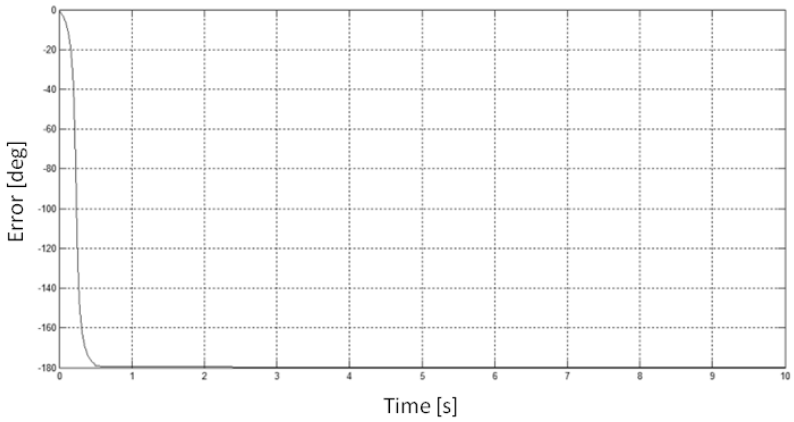


Figure 4-9. *Initial angular position estimation on the x axis*

Subsequently, the path in the second column of Table VI was set in the test equipment and the mean square error of the angular position estimated by our model was calculated. Table VI shows that, when the filter is not active, the drift is quite high, but with the simplified Kalman filter the mean square error is very low.

Table VI – Rotation sequence

Time (s)	**Angular positions for the x axis (deg)**		
	Position expected (deg)	***Kalman filter off***	***Kalman filter on***
11.25	20.00	18.70	19.73
14.25	0.00	-1.70	-0.25
17.25	45.00	42.90	44.94
20.50	0.00	-2.55	-0.17
23.50	45.00	42.00	45.01
26.50	135.00	131.55	134.88
29.50	45.00	40.25	44.46
32.50	135.00	129.40	134.29
35.50	180.00	173.30	179.80

Time (s)	Angular positions for the x axis (deg)		
	Position expected (deg)	*Kalman filter off*	*Kalman filter on*
38.50	135.00	128.10	135.75
41.50	180.00	171.75	180.61
50.00	0.00	-9.40	-0.26
Mean square error		28.27	0.17

4.2.5 Bit true model

A quantized Simulink model was built and tested to simulate fixed point architecture. The rotation sequence of the previous test was utilized to calculate the mean square error, varying the number of bits used to represent each single parameter. Table VII shows the results obtained by varying the number of bits for the quaternion *q*, the A matrix and the P and K matrices.
For the quaternion, the optimum number of bits to obtain the best precision without increasing the logic complexity is 20 bits. For the A matrix representation, even if there is a small increase of precision with the use of 24 bits, 20 bits are recommended to avoid an explosion of the system complexity. Finally, the use of 16 bits is adequate to represent P and K matrices. With the use of the above mentioned arithmetic precision, a final RMS value of 0.19 was obtained, with only a slight degradation w.r.t. the simplified floating point model, whose RMS is 0.17.

Table VII – Mean square error related to the number of bits used for data representation

	Mean square error				
Number of bits used for data	***10 bit***	***12 bit***	***16 bit***	***20 bit***	***24 bit***
quaternion	-	8.86	0.24	0.19	0.19
A matrix	2.36	0.75	0.29	0.19	0.18
P and K matrices	-	0.23	0.19	0.19	-

4.3 Conclusions

In this chapter two sensor fusion algorithms for the estimation of the position has been presented.
The first algorithm, that can be considered as a preliminary evaluation of the possibility to combine the information of different typologies of sensor to obtain other functionalities, show that it is possible to develop algorithms for the approximation of movements with a high degree of precision for what concerns rotational movement. For translational movement, instead, it is not possible to obtain the instantaneous position using only an accelerometer yet; in fact, another measure reference is necessary for compensate errors due to double integration.
The proposed algorithm for the modeling of the translational movement fit our specific issue well, but it is not a general solution. In fact, it is not applicable in applications that need the knowledge of the linear position time after time. So, for this kind of problems, it is still necessary another reference or the imposition of constraints.
Starting from this evaluation, it has been chosen to implement a more reliable and efficient algorithm that could be easily implemented on a chip. At first, a Simulink model of a Kalman filter for an orientation tracking system with 6D IMU sensors was experimentally tested with SensorDynamics Cube Demo sensor. Then, some simplifications of the Kalman filter algorithm has been proposed and it has been verified experimentally that the filter is able to correct the gyro sensor drift and also the initial position at system startup, even if the sensor is upside down. Finally, Simulink model was quantized and the precision loss in fixed point architecture has been evaluated by varying the number of bits used to represent each data. These results are fundamental to finalize the digital design of an integrated system for integrating the orientation tracking in a future 6D IMU sensor.

5 CONCLUSIONS

In the first part of this research a complete analysis of the design flow of a smart sensor based on MEMS technology has been carried out proposing different solutions to minimize the costs and the time-to-market optimizing all the stages concerning the testing.
The whole design flow of a MEMS based architecture has been analyzed, and three critical stages in terms of costs and time-to-market has been identified: the MEMS device characterization and the sensor interface prototyping, the verification testing and the calibration. These stages has been analyzed to find out alternative solutions with respect to the state of the art.
To reach the goal of optimizing the first stage, an environment for the characterization and testing of MEMS and MOEMS devices has been developed. This highly customizable environment permits to achieve the goal of characterizing MEMS and MOEMS sensors reducing the time for the setup of a testing environment and the costs for the adoption of more expensive solutions. As a case study, the environment has been used to characterize a micromirror, showing how it is possible to compute parasitic capacitance values and effects by the use of the correlations between simulation results and test results obtained using this platform, instead of simulating them that requires a high computation power and a long simulation time. The high reconfigurability of this platform permits also to use it for the prototyping of the sensor interface, but not to design new functionalities. To make up for this lack, a bridge for the communication between an ASIC and an FPGA using a reduced number of pin has been designed, and its integration in a sensor interface platform based on ARM9 microprocessor has been illustrated as a case study. Future work is to integrate the bridge in the ISIF platform in order to provide a complete MEMS prototyping environment.
The verification testing concerns the evaluation of the correctness of the design and the developing of the test procedures. It takes place in the laboratory by the collaboration of the designers and the test engineers. This stage requires the setup of a new test environment each time a new product must be tested. In order to reduce the reengineering time, a client-server architecture for the communication between one or more test machine and the MEMS device has been developed. It provides several digital communication protocols and offers a common interface that permits to implement test procedures using different IDEs and any kind of programming language. Providing a common framework for all the sensor devices that use digital protocols, it allows to reduce the setup time and to reuse the same testing applications used by other devices. As a case study, the framework has been used for the communication with SensorDynamics' inertial sensors.
The calibration stage is the most critical in terms of costs, because it requires the use of expensive ATE machines. Although non recurrent costs cannot be reduced, because these machine are fundamental for the calibration and the evaluation of a piece during the production, minimizing the usage time reduces recurrent costs considerably. For this reason, a low-cost hardware/software architecture for the calibration and the evaluation of a MEMS chip has been developed. By the use of a pair of motor system placed orthogonally it is able to stimulate the PUT with movements, and thanks to the flexibility of the software, the architecture can be

easily adaptable to different kind of devices. Although its main goal is to reduce the usage time of an ATE machine during the developing of the calibration algorithm, it is also possible to exploit the architecture also in small and medium production processes.
The focus of the second part of this research is the evaluation of sensor fusion algorithms to combine the outputs of different kind of sensor in order to provide new and more complex information together with the one offered by the sensors themselves. In this context, a preliminary analysis of the behavior of this kind of algorithms in the elaboration of noisy data from the MEMS sensor has been done by the use of a fully software algorithm for the sensor fusion. Then, an algorithm for the sensor fusion based on a simplified Kalman filter has been developed. For this algorithm, it has been also built and tested a quantized Simulink model to simulate a fixed point architecture in the perspective of its synthesizing together with the conditioning chip. In the future, artificial intelligence techniques will be evaluated in order to compare the performance of our solution with other algorithms.

REFERENCES

1. "Yole market research", *http://www.yole.fr* [Online]
2. Eloy J.C., Yole Développement. "What can we expect in 2011 in the MEMS business...?", *http://www.i-micronews.com/* [Online]
3. IEEE Std 1554™-2005, **IEEE Recommended Practice for Inertial Sensor Test Equipment, Instrumentation, Data Acquisition, and Analysis**, Institute of Electrical and Electronics Engineers Inc., 2005.
4. Korvink J. G., Paul O.**, MEMS: A Practical Guide to Design, Analysis, and Applications**, William Andrew Publishing and Springer, 2006.
5. Pallaá S-Areny R., Webster J.G., **Sensors and signal conditioning**, John Wiley & Sons Inc., 2001.
6. Wang L.T., Stroud C.E., Touba N.A., **System-on-chip test architectures,** Elsevier Inc., 2008.
7. Bushnell M.L., Agrawal V.D., **Essential of Electronic Testing for digital, memory and mixed-signal VLSI circuits.,** Kluwer Academic Publishers, 2002.
8. Gaura E., Newman R., **Smart MEMS and Sensors Systems**, Imperial College Press, 2006.
9. Kerkhoff H. G., "Testing of MEMS-Base Microsystems", European Test Symposium, Tallinn May 22-25 2005, pp. 223 - 228
10. Technical Digests of the Annual IEEE International Conference on MEMS.
11. Proceedings of SPIE, The international society for optical engineering.
12. Islam Md. F., Mohd M.A., "On the use of a mixed-mode approach for MEMS Testing", International Conference on Semiconductor Electronics, Kuala Lumpur Dec. 2006, pp.62-65
13. Deb N., Blanton R. D., "Built-in-self-test of MEMS Accelerometers", *Journal of Micro-Electro-Mechanical Systems*, Vol.15, n.1, 2006
14. Courtois B., Karam J. M., Mir S., Lubaszewski M., Székely V., Rencz M., Hofmann K., Glesner M., "Design and Test of MEMS", VLSI Design, Goa 07-10 Jan. 1999, pp.270-275
15. Chen L., Huang Y., Fan K., "A Dynamic 3-D Surface Profilometer With Nanoscale Measurement Resolution and MHz Bandwidth for MEMS Characterization", *IEEE Trans. On mechatronics*, vol.12, no.3, 2007
16. Bosseboeuf A., Petitgrand S., "Characterization of the static and dynamic behaviour of M(O)EMS by optical techniques: Status and trends", *Journal of Micromechanics and Microengineering*, vol.12, pp.S23-S33, 2003
17. MSV 300—Micro Scanning Vibrometer, "Polytec PI product information", Tustin, CA
18. Lawton R. A. et al., "MEMS characterization using scanning laser vibrometer", SPIE 1999 Symp. Microelectronic Manufacturing, 1999
19. "Reliability, Testing and Characterization of MEMS/MOEMS", Proceedings of SPIE, http://www.spie.org/
20. Lawrence E. M., Speller K. E., Yu D., "Laser Doppler vibrometry for optical MEMS," SPIE, Ancona Jun. 18–21 2002, vol.4827, pp.4827–4834

21. Ong A.O., Hock F.E., “Motion Characterizations of Lateral Micromachined Sensor Based on Stroboscopic Measurements”, *IEEE Sensors Journal*, vol.7, n.2, 2007
22. Dong Z., Huang D., Zhang D., Wu W., “An Automatic MEMS Testing System based on Computer Microvision”, IEEE ICMA, Luoyang Jun. 2006.
23. Battini F., Volpi E., Marchetti E., Cecchini T., Sechi F., Fanucci L., Hofmann U., “A fast-developing and low-cost characterization and test environment for a double axis resonating micromirror”, *Microelectronics Journal*, Vol.41, n.1, 2010, pp. 778-788
24. D’Ascoli F., Tonarelli M., Melani M., De Marinis M., Giambastiani A., Fanucci L., “Intelligent sensor interface for automotive applications”, ICECS 2005, Gammarth Dec. 2005
25. Iozzi F., Fanucci L., Giambastiani A., “Fast prototyping flow for sensor interface”, IEEE Ph.D. Research in Microelectronics and Electronics, Otranto Jun. 2006
26. Cecchini T., Sechi F., Saponara S., Fanucci L., “AHB-Compliant Bridge with Programmable Frequency Downscaling for Efficient off Chip Digital Communication of Pin-limited Automotive Smart IC Sensors”, *Sensors and transducers*, Vol.8, n.1, 2010
27. Cecchini T., Sechi F., Saponara S., Fanucci L., “Pin-limited Frequency Converter IP Bridge for Efficient Communication of Automotive IC Sensors with Off-chip ECUs”, IEEE IDAACS 2009, Rende Sep. 21-23 2009, pp.167-171
28. Cecchini T., Sechi F., Bacciarelli L., Mostardini L., Battini F., Fanucci L., De Marinis M., “Pin-limited frequency downscaler AHB bridge for ASIC to FPGA communication”, DSD 2008, Parma Sep. 3-5 2008, pp. 367-372
29. ARM IHI 0033A, “AMBA 3 AHB-lite protocol specification”, v1.0 2008
30. “Gaisler research home page”, http://www.gaisler.com [Online]
31. Hofmann U., Oldsen M., Quenzer H., et al., “Wafer-level vacuum packaged micro-scanning mirrors for compact laser projection displays”, SPIE, San Jose Jan. 22-23 2008 , vol.6887, pp.100−114, 2008
32. Patterson P.R., Hah D., Nguyen H., Toshiyoshi H., Chao R., Wu M.C., “A Scanning Micromirror With Angular Comb Drive Actuation”, 15th IEEE International Conference on Micro Electro Mechanical System, Las Vegas Jan. 20-24 2002, pp. 544-546
33. Serafini L., Carrai F., Ramacciotti T., Zolesi V., “Multi sensor configurable platform for automotive applications”, IEEE DATE 2006, Munich, pp 219-220
34. Vincentelli A.S., Di Natale M., “Embedded systems design for automotive applications”, *IEEE Computer*, vol. 40, issue 10, pp.42-51, 2007
35. Battini F., Tonarelli M., Fanucci L., Giambastiani A., De Marinis M., “Experiencing with AMR sensor conditioning in automotive field”, IEEE PRIME 2007, Bordeaux Jul. 2–5 2007, pp. 1-5
36. Zuchowski P.S., Reynolds C.B., Grupp R.J., Davis S.G., Cremen B., Troxel B., “A hybrid ASIC and FPGA architecture”, IEEE ICCAD 2002, San Jose Nov. 10-14 2002, pp.187-194
37. Miro Panades I., Greiner A., Sheibanyrad A., “A Low Cost Network-on-Chip with Guaranteed Service Well Suited to the GALS Approach”, IEEE Nano-net 2006, Lausanne Sep. 14-16 2006, pp.1-5

38. Miro Panades I., Greiner A., “Bi-Synchronous FIFO for Synchronous Circuit Communication Well Suited for Network-on-Chip in GALS Architectures”, IEEE Networks-on-Chip, Princeton May 6-7 2007, pp.83–94
39. Riccardi D., Causio A., Filippi I., Pregnolato L.V.A., Galbiati P., Contiero C., “BCD8 from 7V to 70V: a new 0.l8um Technology Platform to Address the Evolution of Applications towards Smart Power ICs with High Logic Contents”, IEEE ISPSD '07, Jeju May 27-31 2007, pp.73 – 76
40. D’Ascoli F., Melani M., Fanucci L., Bacciarelli L., Ricotti G., Pardi E., Vincis F., Forliti M., De Marinis M., “A Programmable and Low-EMI Integrated Half-Bridge Driver in BCD Technology”, IEEE DATE 2008, Munich Mar. 10-14 2008, pp. 879 – 884
41. Graebe R.H., Poe C.C., Conductron-Missouri, “Digital Techniques to Test Analog Systems”, *IEEE Transactions on Aerospace and Electronic Systems*, AES-2 Issue 4, 1966, pp. 791-797
42. Baronti F., Marraccini E., Roncella R., Saletti R., Manni G., Palama G., “Multi-sensor multi-protocol acquisition system for luxury-yacht production test and characterization”, Signals, Circuits and Systems (SCS), Jerba Nov. 6-8 2009, pp.1-6
43. Qureshi M.A., Ahmed I., Hassan A., “Automation of testing and qualification of inertial rate sensors”, Computer, Control and Communication, Karachi Feb. 17-18 2009, pp.1-6
44. Rokkam M., Chatni M.R., ul Haque A., De Carlo A.R., Robinson B.F., Irazoqui P.P., Porterfield D.M., “High-density data acquisition system and signal preprocessor for interfacing with microelectromechanical system-based biosensor arrays”, *Review of Scientific Instruments*, volume 78, 2007
45. Löwy J., **Programming WCF Services**, 2nd ed., O'Reilly Media Inc., 2009
46. Booth D., Haas H., McCabe F., Newcomer E., Champion M., Ferris C., Orchard D., “Web Services Architecture”, W3C Working Group Note, 2004
47. “Novell, Mono project website”, http://mono-project.com/ [Online]
48. Seongbae L., Gi-Joon N., Junseok C., Hanseup K., Drake A.J., “Two dimensional position detection system with MEMS accelerometer for mouse applications”, Design Automation Conference, 2001. Proceedings Volume , Issue , 2001 pp.852-857
49. Young K., Li D., “A Motion Capture System Using Accelerometers” [Online]
50. Mizell D., “Using Gravity to Estimate Accelerometer Orientation”, Seventh IEEE International Symposium on Wearable Computers (ISWC’03), White Plains Oct. 21-23 2003
51. Algrain M.C., Saniie J., “Estimation of 3D angular motion using gyroscopes and linear accelerometers”, *IEEE Transactions on Aerospace and Electronic Systems*, Vol.27, Issue 6, pp.910–920, 1991
52. Sechi F., Fanucci L., Maci E., Giambastiani A., Rocchi A., De Marinis M. “Demonstrator of 3 axis gyroscope/accelerometer using embedded 3D demo architecture”, DSD 2008 (WIP session), Sep. 3-5 2008
53. Sabatelli S., Sechi F., Fanucci L., Rocchi A., “A sensor fusion algorithm for an integrated angular position estimation with inertial measurement units”, DATE 2011, Grenoble Mar. 14-18 2011
54. “SD755 - Micromachined Integrated Gyro and Acceleration Module Specification”, Sensordynamics AG, 2007

55. “Irrlicht Engine website”, http://irrlicht.sourceforge.net/ [Online]
56. Kim A., Golnaraghi M. F., “A Quaternion-Based Orientation Estimation Algorithm Using an Inertial Measurement Unit”, PLANS 2004, Apr. 26-29 2004, pp.268-272
57. Yun X., Lizarraga M., Bachmann E.R., McGhee R.B., “An Improved Quaternion-Based Kalman Filter for Real-Time Tracking of Rigid Body Orientation”, IROS 2003, Las Vegas Oct. 27-31 2003, Vol.2, pp.1074-1079
58. Marins J.L., Yun X., Bachmann E.R., McGhee R.B., Zyda M.J., “An extended Kalman Filter for Quaternion-Based Orientation Estimation Using MARG Sensors”, IROS 2001, Oct. 29 - Nov. 3 2001, Vol.4, pp.2003-2011
59. Wang M., Yang Y., Hatch R.R., Zhang Y., “Adaptive Filter for a Miniature MEMS Based Attitude and Heading Reference System”, PLANS 2004, Apr. 26-29 2004, pp.193-200
60. Kraft E., “A Quaternion-Based Unscented Kalman Filter for Orientation Tracking”, IEEE International Conference of Information Fusion, 2003, pp.47-54
61. LaViola Jr. J.J., “A Comparison of Unscented and Extended Kalman Filtering for Estimating Quaternion Motion”, IEEE American Control Conference, Jun. 4-6 2003, Vol.3, pp.2435-2440
62. Khoder W., Fassinut-Mombot B., Benjelloun M., “Inertial Navigation Attitude Velocity and Position Algorithms using Quaternion Scaled Unscented Kalman Filtering”, IECON 2008, Orlando Nov. 10-13 2008, pp.1754-1759
63. Lee H.J., Jung S., “Gyro Sensor Drift Compensation by Kalman Filter to Control a Mobile Inverted Pendulum Robot System”, ICIT 2009, Gippsland Feb. 10-13 2009, pp.1-6
64. Vicci L., “Quaternions and Rotations in 3-Space: The Algebra and its Geometric Interpretation”, Microelectronic Systems Laboratory, Department of Computer Science, University of North Carolina at Chapel Hill, 2001
65. Kuipers J.B., **Quaternions and Rotations Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality,** Princeton University Press, 1999
66. Kalman R.E., “A New Approach to Linear Filtering and Prediction Problems”. *Transactions of the ASME–Journal of Basic Engineering*, 1960
67. Welch G., Bishop G., “An Introduction to the Kalman Filter”. Department of Computer Science, University of North Carolina at Chapel Hill, 2001
68. Grewal M.S., Andrews A.P., **Kalman Filtering: Theory and Practice Using Matlab, Second Edition**, John Wiley & Sons, 2001
69. Chui C.K., Chen G., **Kalman Filtering: with Real-Time Applications,** Springer, 1991