Bottom trawl fishing is a relevant extractive economic activity. However, the limitation of resources and the impact on the environment demand for a more rational exploitation in order to render the activity sustainable and productive in a near future. To this purpose it is important to understand and to analyze the behaviour of fishing gears. DISPAR is a collaboration project between the UPC-CTVG and the CSIC aimed to move towards this direction. DISPAR includes a simulation module and a user-friendly interface software. The module of simulation solves the governing equations for different components of a bottom trawl gear (warps, otterboards, sweeps and net) and uses an iterative staggered strategy to account for coupling effects. Relevant outcomes are the configuration of the gear and the distribution of tensions under different fishing conditions such as fishing depth or towing speed. The former includes, for instance, otterboard opening, angle of attack, or warp geometry. The later allows to analyse the contribution of each component to the total gear drag. This simulation module is interfaced in order to facilitate data inputs and visualization of results to the final user. The resulting product is a multifunctional aid tool that complements data measurements, helps fishermen and sector companies to optimize bottom trawl gears, and deserves to pedagogic purposes.

Design Flow for DSP&FPGA algorithms with Matlab

J. del Río, M. López, A. Mànuel, E. Molino (1), A. Carlosena (2)

Rambla Exposición s/n, 08800 Vilanova i la Geltrú, Spain
Telf: +34 938 967 200, joaquin.del.rio@upc.edu
(2)Dpt. Electrical and Electronic Engineering
Universidad Pública de Navarra, E-31006 Pamplona, Spain

Keywords: C Intrinsics, Matlab, DSP Software, Smart Sensor, Simulation, Verification

1. Introduction
The tangible aim of this Project is the design of a smart accelerometer, having a very low cost, making use of a piezoelectric element as basic sensing material, and adding a mixed mode conditioning circuit. This synthetic goal, consist of a number of combined objectives as:

- Development of techniques for the calibration, compensation and frequency range extension of basic sensing elements made typically of piezoelectric ceramics
- Adaptation or development of novel techniques of calibration, to make possible the above mentioned methods
- Development and implementation of novel signal processing techniques as they apply to the above mentioned sensors for the estimation (parametric, non-parametric, blind,..) of mechanical systems, and excitation signals (impact). Both multiple and single sensor applications will be considered.

This paper presents a proposed methodology in order to evaluate the execution over programable hardware platforms (DSPs, FPGAs) of different algorithms developed and tested in Matlab. In a more general view about the global project this paper is related about Digital Signal Processing block in figure 1. Analog Signal Conditioning block is related in references[7][9]. The algorithms developed are explained in[3][4][5][6] and are related to process impulsive signals from impact sensors. The aim of this work is to implement and synthetize this algorithms over a specific hardware or a programmable hardware like FPGA or DSPs in order to evaluate the algorithm complexity and the viability of its implementation about computational cost, processing time, necessary memory, etc... in order to integrate the signal processing algorithm in an integrated circuit with the sensor (accelerometer), the analog signal conditioning, and the analog to digital conversion. The work is framed inside the project DIATRIBA, “Design a low cost smart accelerometer” CICYT DPI 2003-08637-C3-03.

2. State of the art
At present the simulation environments as Matlab make possible the implementation and simulation of algorithms using PCs or workstations for execute algorithms. Often these signal processing algorithms finally will be implemented in a specific hardware platform.
that is not going to be a PC. It is necessary to keep in mind on what hardware goes to be executed the algorithm, therefore this is going to establish and to delimit the velocity of processed, the resolution in the mathematical operations, the amount of memory available, etc... in final: the computational price of the algorithm. Matlab permits the Code generation for different platforms (DSPs and uC) from Simulink using the Toolbox Real-Time Workshop Embedder Coder although certain limitations exist about the code that can be compiled, for example: the functions of Matlab, Matlab Functions, they are not borne for Real-Time Workshop “block type ”MATLABFcn”: Not yet supported by Real-Time Workshop”.

The compilation on hardware of external functions of Matlab written in ANSI-C can be carried out previous compilation to S-Functions for to be utilized in Simulink or compiled to a MEX function (Matlab Executable dll) (mex funcion.c) for to be utilized in scripts of Matlab (*.m) or in a Matlab Function in Simulink.

Also it is possible the stand-alone applications generation (*.exe) or C code from scripts (*.m) of Matlab by means of Matlab Compiler, mcc (mcc – m function.m) although this solution is only valid for the compilation of applications for windows operating system, and is not usefull for the compilation on hardware.

These and other limitations do the generation of C code from Matlab, using the Matlab functions be not the sufficiently versatile for our purpose:

1- Simulation of algorithms by means of Matlab.
2- C code generation.
3- Algorithm evaluation on DSP or FPGA hardware.

For that reason exist publications related to that problem and propose the development of software to solve said problem, and they present systematic of software design for environments DSP or FPGA [1]. At present different companies work in the software tools implementation to synthetize algorithms directly from Matlab [10].

3. Proposed methodology to evaluate the algorithms over hardware

Refreshing the final objective that is the implementation on hardware of algorithms simulated in Matlab have been carried out the following phases:

3.1 Initial tests

The objective of this first phase is the evaluation of the algorithm, and to verify that the processed carried out extracts the information desired [3][4][5][6]. The first phase has consisted of the development and the simulation of the algorithms integrally in Matlab. In this first phase, themselves has not kept in mind at no time which is going to be the final platform on the algorithm will be executed and they have developed without carrying out any type of optimization by memory, utilizing numerical values in floating point, complex values, etc.

In the second Phase we waste a lot of time studying the best way of executing these algorithms on hardware. In our case we have a DSP platform of Texas Instruments, the DSK6711 for which Matlab is able to generate C code toward Code Composer Studio. The problems that were arising were various, but the most important are:

- algorithm are developed as scripts of matlab (*.m). Valid for the simulation from Matlab.
- The generation of C code toward the DSP using the Real-Time Workshop Embedded Coder (RTW) is carried out from Simulink.
- The simulation of *.m from Simulink is possible by means of the block “Matlab Functions” but is not possible its compilation by the RTW.

At this point we seek another alternative to the direct compilation using RTW and Simulink on the DSP, just as appears in the figure 2, wrong way, the generation of code directly is not possible in our case because we are using different Matlab Functions that are not in the DSP library for RTW.

In the third phase we utilize the property of Matlab to generate C code using the Matlab compiler, the MCC. Thanks to MCC is possible to generate stand-alone files for windows *.exe, dynamic libraries *.dll and C code with his prototipe functions *.h from a Matlab script *.m.

Once it generated this C code we can import it since another compiler as could be Borland C++ or Visual C++ and to integrate this functions in another project. The problem is that C code generated from Matlab, uses a series of libraries *.lib own of Matlab and this does that this project in C only can be compiled toward an Win32 application and not toward a specific hardware.

We carry out different tests importing the C code to Code Composer Studio, but was impossible to compile the project for to be executed on the DSP. Después de ininidad de pruebas decidimos seguir el siguiente procedimiento:

After infinity of tests we decide to continue the following procedure:

3.2 Proposed Methodology

The process to evaluate and execute our algorithms on hardware is summarized in the figure 2, good way:

Evaluated the algorithms using Matlab without keep in mind the hardware, the necessary functions are implemented directly in ANSI-C that the algorithm will utilize, keeping in mind the characteristics of the hardware: fixed point, floating point, blocks multipliers, etc... Once the library functions is generated by us manually in C we can import them to Matlab to
carry out again the simulations of the algorithm. This process we carry out it by means of the creation of “MEX functions” using Mex Compiler (mex function.c). It is important to keep in mind that to utilize these functions in C from Matlab it should have a specific syntax for the call to the functions that comes described in the documentation of Matlab [8]. Imported the functions of C as MEX files in Matlab can be utilized directly in scripts of Matlab *.m and in calls from Simulink using the block “Matlab Fcn”.  

At this time we are ready to carry out the simulation of our algorithm in Matlab utilizing the functions that create initially in C. Once it validated the algorithm on Matlab we have two possible options:  

Option 1:  
For the execution on the DSP (of TI) we would use directly the CCS (Code Composer Studio) and we would create a new project where we will utilize the libraries of C functions that we have developed. In this case, Matlab is used for the simulation, and the implementation of the project for the DSP is created by hand manually, no using the RealTime Workshop. Of analogous form, if the execution is going to carry in a FPGA we will utilize the C functions and we will integrate them in our VHDL code in the environment that we are using (ISE, MaxPlus...).  

Option 2:  
Another possible solution that at present are evaluating is the creation of s-functions from our functions in C for to be simulated in Simulink and subsequently and thanks al RTW to generate automatically the project and the necessary code for the DSP platform, DSK6711.  

3.3 Comercial Alternatives  
Actually firm Accelchip has developped a software platform and libraries to help the programmer to translate automatically from Matlab routines to RTL (register transfer level) language in order to minimize the time spent to translate manually (figure 3) the algorithms as we have been proposed in this paper.  

Figure 3: AccelChip representation about barriers between Matlab and the hardware language depending (RTL, register transfer level).  

4. Conclusions  
This paper has presented a design flow to simulate, develop and to verificate of DSP software within Matlab combining C code with intrinsic Matlab functions. Right now we are evaluating this algorithms in hardware and measuring differents execution parameters in order to improve its efficiency. Now we are evaluating AccelChip software in order to test this alternative to our proposed traditional methodology.  

5. References  
[10] www.accelchip.com