Application Specific Instruction Set Processor for Sensor Conditioning in Automotive Applications

A. Sisto[a,*], L. Pilato[a], R. Serventi[b], S. Saponara[a], L. Fanucci[a]

[a] *DII, University of Pisa, via G. Caruso 16, 56122 Pisa, Italy*
[b] *ams Italy S.r.l., Via Giuntini 13, 56023 Navacchio (Pisa), Italy*

Abstract

In the automotive electronic market, sensor conditioning is one of the driving applications. Sensor solutions are pervasive in the vehicle, while signal processing in such application is getting more and more complex. Currently the design strategy is often the standard ASIC flow, but the design effort can be reduced by automatic or platform-aided design strategies, or by using software-based solutions. In this paper SensASIP platform is presented. It is a design platform targeting a microprocessor architecture enhanced by dedicated instructions for computing intensive sensor signal processing tasks. SensASIP allows a seamless design flow from MATLAB-based algorithm definition and instruction set design and simulation, down to hardware macrocell HDL description and implementation in CMOS technology. SensASIP features are described through two automotive sensor conditioning examples. Special focus is put on its increased flexibility and reduced design-effort vs. standard ASIC design approach and on its low complexity overhead vs. other state-of-art software-based solutions.

Keywords: ASIC (Application Specific Integrated Circuit), ASIP (Application Specific Instruction set Processor), digital design, sensor conditioning, automotive, signal processing, design platform

1. Introduction

Sensor conditioning in automotive application is one of the greatest shares of the market. The number of sensor-based solutions for each vehicle is growing, together with the requested features. As observed in [1], the number of sensor systems in each vehicle was around 60-100 in year 2013, and it is expected to reach 200 in next years. Automotive sensor technologies have usually peculiar features and require different processing capabilities. However, a big part of the market is made up of sensor families, which feature quite low complexity and data rate. Some examples of such sensor systems are capacitive sensors, for pressure, acceleration or angular rate measurements, inductive sensors for linear and angular position measurement, and magnetic sensors for position measurements. In addition to processing specification, in automotive electronic market also functional safety features are important specifications. Sensor conditioning electronic in automotive application must be functioning and safe. Signal processing growing complexity and more strict safety features lead to an increasing area for automotive sensor ASIC devices, and hence increasing cost and static power consumption. Moreover, in this field the ASIC design flow is traditionally custom. It is possible to assert that each of the additional features to be implemented on silicon requires an additional design effort. Of course, blocks reuse and designer experience play a key role in custom, i.e. hardwired design, in a scenario where the hardwired complexity is getting larger, and the too high design effort puts the design at risk of not to meet the time-to-market. The emerging features growth is leading to new design strategies (e.g. [2, 3, 4]), and to novel design tools ([5]). The trend is to move the design effort to higher level. However, the complexity overhead due to high-level design strategies, and to their lack of optimization capability, is a cost that companies are typically reluctant to pay.

In order to address the above issues, in this paper a novel design platform, called SensASIP, for sensor conditioning devices is proposed. In Section 2 the state of the art of possible design paradigms

*Corresponding author
Email addresses: arcangelo.sisto@for.unipi.it (A. Sisto), luca.pilato@for.unipi.it (L. Pilato), riccardo.serventi@ams.com (R. Serventi), sergio.saponara@unipi.it (S. Saponara), luca.fanucci@unipi.it (L. Fanucci)

is presented for the same class of applications. Section 3 describes the new SensASIP platform. In Section 4 and Section 5 two target applications are presented and the design with state of the art methodologies are compared with the SensASIP design steps. Its implementation results in CMOS technology are shown in details in Section 4.5 and Section 5.3. Conclusions are drawn in Section 6.

2. State of the art review on automotive IC design for sensor signal processing

2.1. Hardwired DSP design

Sensor conditioning algorithms commonly implemented in automotive applications are relatively simple, with respect to other applications such as e.g. telecommunications, imaging and so on. For this reason, the automotive field has been often less exposed to technology and architecture innovations. Devices are often manufactured in mature CMOS technologies such as 0.35 μm or 0.18 μm, also to face harsh environment: ESD up-to 4 kV , large temperature (from $-40°C$ to $150°C$) and voltage range requirements.

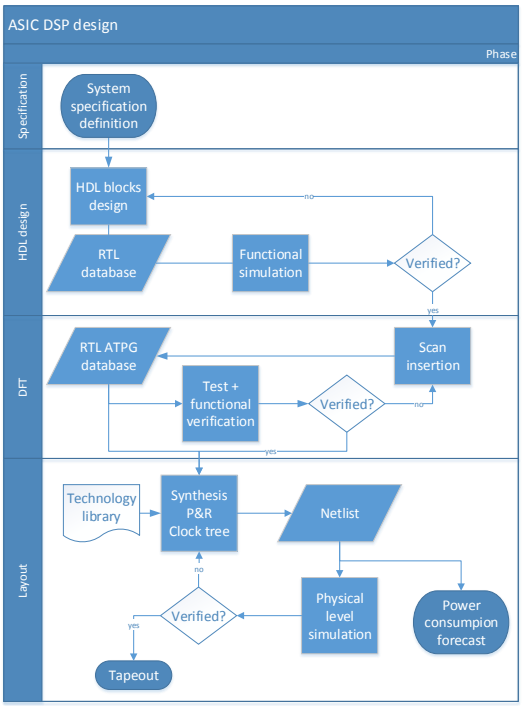

Figure 1: Hardwired design flow.

The hardwired design flow has been used in the last decades and it is often still used; for instance, it is explained in a paper from the middle nineties [6]. Since than, the design methodology hasn't been changed too much, but many innovative tools have been deployed to make the design database management easier. A standard hardwired design flow is depicted in Figure 1. This approach allows an effort-dependent optimization of the area constraint, but as shown in picture, the flow is likely to loop many times the verification and the bug correction stages, for each design phase. Furthermore, in many cases a verification fail in latest phases can only be solved by HDL block fix. In this case, the successive steps must be repeated. Of course, well-experienced digital designer allows a faster design, and usually less loops are necessary if many known blocks are re-used. However, increasing implemented features require increasing design effort and increasing silicon area occupation. Both these factors are cost elements for electronic companies. For these reasons, in latest years different approaches have been attempted.

2.2. Microprocessors IP

The software-based solution is a good option when many features are required, if the timing specification can be met. While a hardwired DSP can ideally perform the whole processing in a single clock cycle, or in few clock cycles in case of timing violations, a programmable processor is intrinsically requiring more cycle to perform an algorithm coded into an instruction-based software. With

equal clock frequency, a hardwired solution is likely to gain better output timing performances than a software-based solution. However, the option of embedding a microcontroller unit within a system has been widely adopted, even in FPGA-based SoC, e.g. [7].

The software-based solution can be started by instantiating a microprocessor IP in current design. Core vendors provide these IPs for embedded purposes. One of the most used core IP for applications in the same complexity range of the automotive sensor systems is the ARM Cortex™ family. In latest years many researches have been performed for different applications with embedded Cortex-M0 processors ([8],[9] and [10]), including automotive applications, [11]. The M0 results suitable for applications not computationally critical, such as sensor conditioning circuits for sensor classes introduced in Section 1. From design effort point of view, once a microprocessor IP is instantiated, many RTL verification iterations can be avoided, since the HDL design problems move to software design field. The microprocessor hardware is supposed to be bug-free, only its firmware must be designed and verified.

A great feature of this solution is that a software-based solution is more flexible than a hardwired one: electronic devices often require to be redesigned in case of issues found in prototype evaluation, production test or application. If a bug has to be fixed, or a specification is changed, the modifications on a hardwired design, however simple, make necessary the verification flow to be fully re-run. Even in case of issue solvable by metal change, the layout checks and the netlist simulations must be repeated. For software-based solutions, most of these issues can be solved simply by a program modification.

Microprocessor IPs are favorable from design and redesign points of view. However, the main drawback of this strategy is the area overhead. Given the same processing algorithm, the hardwired solution is usually better optimized from complexity point of view. When an additional feature must be implemented on hardwired architectures, the designer must quote an additional silicon area, depending on the complexity of the feature to be added. For microprocessors instead, given a program memory size, the additional feature can either be coded with no need to increase the program memory size; or the additional code can make necessary a larger memory: in this case the total complexity will increase, too. This leads to a stepped area vs features trend; steps are due to software code maximum memory size reaching. By the way, it is possible to assert that for a certain amount of features, the microprocessor architecture could be more suitable than the hardwired one: the microprocessor core keeps its complexity, and for large program memory sizes it may be more affordable a code extension than a new hardwired block.

From cost point of view, the IP instantiation must be paid. One must take into account this cost, beside design effort factor. I/O interfaces from ARM are very flexible, but with respect to hardwired solution some timing requirements may be not met: an additional application-specific I/O layer between the IP and the analog-to-digital interface may be needed. Generally, to meet application specific specifications, a microprocessor IP must be often instantiated beside other simple hardwired blocks (i.e. custom hardware accelerators).

2.3. Custom processors and ASIPs

When a third-part microprocessor IP cannot meet all the performance specifications, other solutions must be explored. In work [12], Takaya tried a multi-core solution to boost performances. However, another option to gain better performances with a microprocessor architecture is to enable custom instructions. EDA tool suite providers moved to this approach in latest years. Cadence™ provides Xtensa® IP [13], which is a customizable processor with a great ease of insertion in ASIC standard design flow; but it includes many blocks for highly complex computations: the resulting design would be oversized for relatively simple applications such as signal conditioning.

The same concept has led to Application Specific Instruction set Processor (ASIP) automatic design methodology development ([14]). The LISA (Language for Instruction Set Architectures) is now part of Synopsys™ tool suite. It is suitable for applications requiring only few instructions, targeting applications not very computationally complex, too. The LISA language and the Processor Designer tool make available the HDL description of an ASIP and its custom compiler starting from high-level specifications. The user must define some key features of the target ASIP, such as pipeline width, instruction names and their custom operations. This design methodology has been also used for automotive applications, e.g. a MEMS IMU sensor conditioning in [15].

Even if ASIP design with LISA gains good area results, it is an automatic design procedure, where complexity optimization is not a key feature. From performances point of view, this approach can achieve good results through custom instruction definition and implementation. The trend in this

design methodology branch is to develop multicore and parallel processing units (e.g. [16], [17], or in automotive application [18]). Design effort is a positive aspect of ASIP, too. Once LISA language is getting familiar with the designer, the flow is automated. The Processor Designer© tool cost must be considered.

3. SensASIP platform description

SensASIP (acronym for Sensor conditioning ASIP) is a platform aiming at supporting development of sensor conditioning systems. It has some of the main advantages of the software-based solution, to overtake the standard hardwired design flow of sensor conditioning electronic in automotive. The design target of the SensASIP platform is a custom microprocessor with the following main features: three pipeline stages; hazard detection circuits; customizable RISC-based Instruction Set; Multi-Cycle Instruction (MCI) capability; modular extended-ALU; parametric register file size; RAM and OTP (One-Time Programmable) memory interfaces capability; interrupt management control logic; power-save mode `sleep`. The extended-ALU is a modular block implementing two's complement fixed-point operations. In the automotive application field, fixed-point arithmetic results suitable for signal processing unit architecture.

SensASIP is a hardware platform [19], being the micro-architecture of the target macrocell (the ASIP) substantially fixed. The target hardware of the platform is abstracted to model levels for achieving a design aid from early design steps.

SensASIP is a set of development environments: a MATLAB™ platform for a design space exploration through the definition of the current IS (ISAEP, Instruction Set Architecture Emulator Platform), another MATLAB-based platform to run the assembly code on a one-to-one model of the microprocessor, and the block-based VHDL description of the microprocessor for the hardware configuration verification and physical implementation of the sensor conditioning unit (synthesis, place&route and clock tree).

The design flow (Figure 2) starts from the ideal algorithm to get proper results from the sensor signal. ISAEP can be used in early steps of the design to define a good Instruction Set to remap the ideal function on the microprocessor architecture. The starting language at this stage is a RISC-based assembly; it makes available some basic instructions, e.g. signed and unsigned sum and subtraction, shifts, memory accesses and program jumps. ISAEP development was aiming at ease of adding custom instructions; if the processing flow requires them, it is possible to add an instruction only by defining its name and the operation to be performed in MATLAB language. The designer must take into account that a custom operation often requires an ad-hoc module inside the extended ALU of the processor. The trade-off between the area constraint and the processing advantages must be evaluated at this stage. ISAEP is a bit-true Instruction Set simulation platform; this feature enables the evaluation of a core system parameter: the bit-width of the processing registers and buses. It is possible to get performances analysis already at this step.

Once the Instruction Set is defined, the software design can start. The development of the firmware also takes into account the other application specification, e.g. control and features, communication interfaces and functional safety strategy. The software design is aided by the bit-true/cycle-true (BT/CT) model. It emulates the three-stage pipeline architecture of the RTL microprocessor. The software code can be reliably verified in MATLAB environment. The top-down design aided by the two high-level platforms in MATLAB is strongly supported by the bottom-up approach of the BT/CT model: the RTL behavior of the microprocessor is one-to-one mirrored in the model, allowing reliable simulations.

The verified firmware can be moved into RTL verification environment through a conversion to binary format. A parser tool for this conversion is part of the platform. Another design step is to customize the hardware instances in the RTL design (e.g. select the current application RAM size, choose the parameters for register file size, addresses bus-width etc.).

A SensASIP platform-based design can target many different sensor-conditioning applications. Inductive, capacitive and magnetic position sensor systems are worth the mention in this context. In next sections, two of these application implementations will be presented.

4. Inductive position sensor case study

The main focus of this paper is the design of a inductive position sensor conditioning unit through the employment of SensASIP platform.

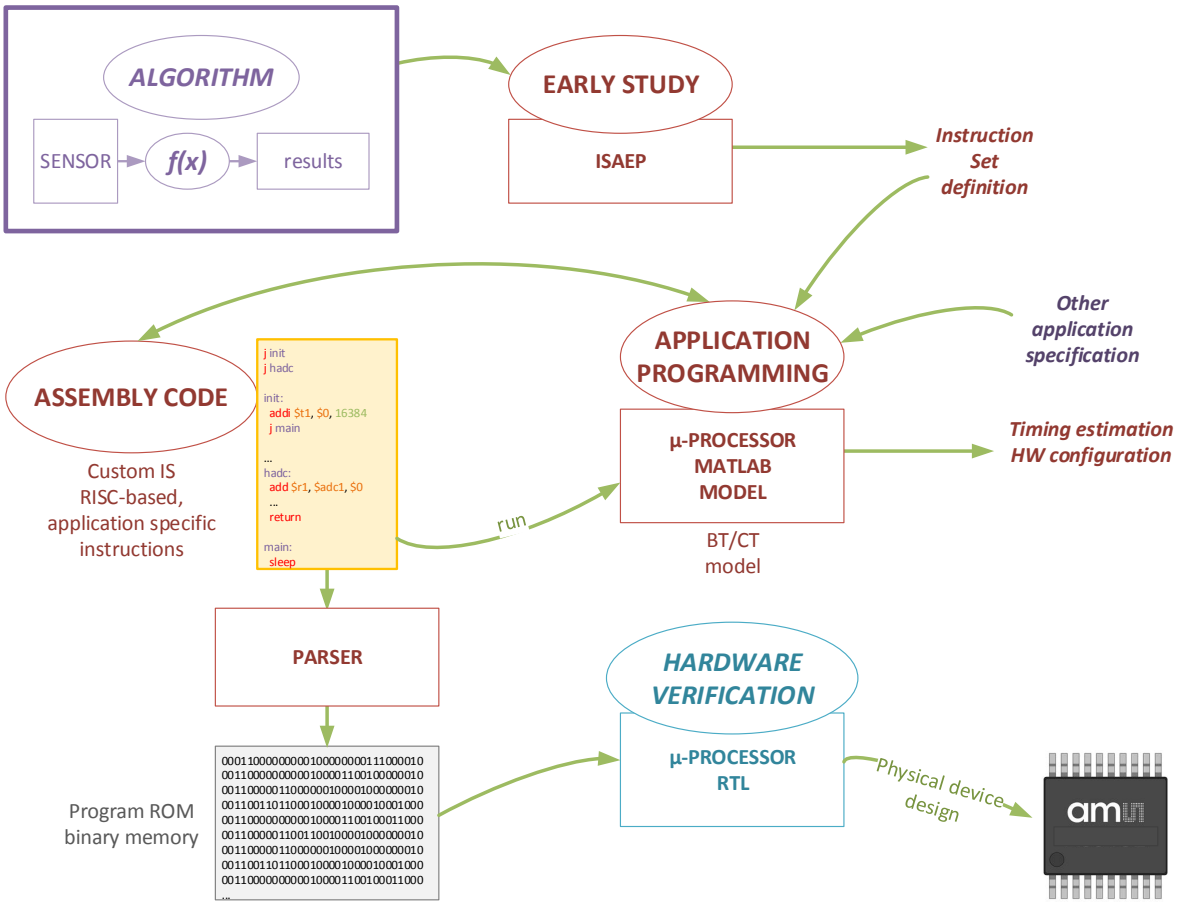

Figure 2: SensASIP platform design flow.

4.1. Target algorithm

The inductive position sensors measures the angular displacement of a coil placed on a movable part (e.g. throttle, or accelerator pedals). The basic functionality of this application is presented in [20, p. 1902] and [21].

The sensor system has a transmission stage, which drives an excitation coil at RF. The transmitted signal has a carrier and a modulated signals. The magnetic field of the excitation coil inductively couples to the movable coil. The current on the receive coil is sensed; the phase shift between the modulated transmitted and received signals is processed to calculate the pedal displacement.

Most part of the processing chain is a series of conditional modulo, sum / subtraction, shift extension and clamping operations. The conditions are often depending on the processed quantity current value and stored past values, or sometimes on processing parameters stored in OTP memory.

For some scaling steps the multiplication operation is necessary. A simplified scheme for the signal processing operations is shown in Figure 3.

The ratio between RF transmission and the modulated signal frequencies, which was around 1000, and the clock frequency for the processing, which was 4 times greater than the RF carrier frequency, make the maximum phase shift signal representable on 12 bits.

4.2. Hardwired design

A standard design flow has been followed to implement the algorithm described in 4.1. The device has been produced by ams AG in 0.35 μm HV CMOS technology.

Beside the signal processing features already discussed, that have been implemented into a DSP block, a Tx (transmitter) and a Rx (receiver) units, other blocks have been implemented: communication block (PBD interface); OTP interface and parameters storing; test management block; diagnostic management unit; main control block.

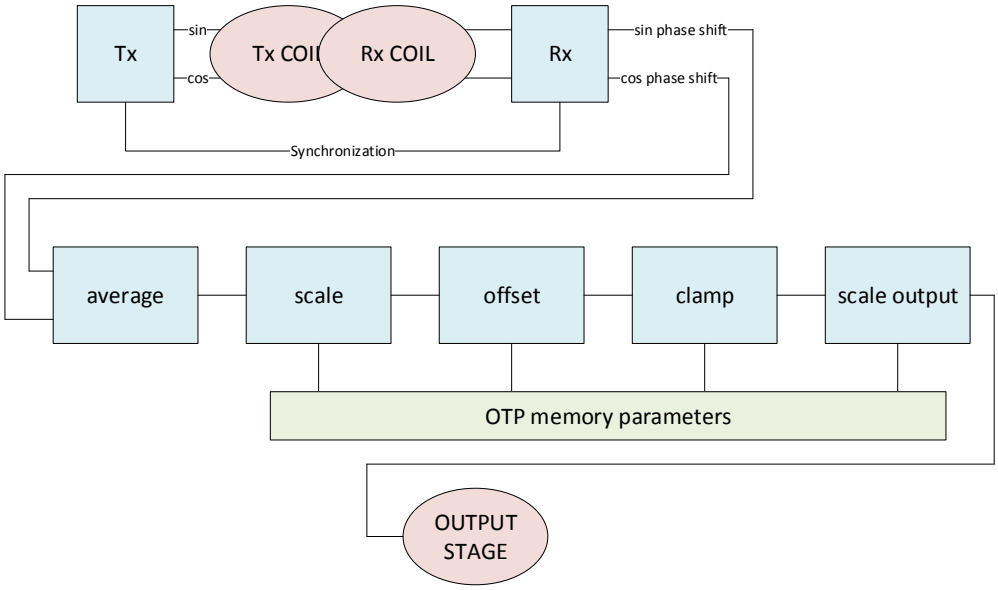

Figure 3: Inductive position sensor signal processing scheme.

The DSP block has a resource sharing architecture, with an ALU for the arithmetic operations. The block is clocked at 2 MHz. The number of necessary clock cycles for the entire signal processing, once an ADC data is available, is around 130 (DSP elaboration time around 65 μs). The silicon area for the implementation of the whole digital architecture is 1.09 mm^2. In terms of complexity it is equivalent to around 25000 gates. These data include a 128-bit OTP memory, too. More detailed data will be discussed in Section 4.5.

4.3. Microprocessor IP implementation

In order to implement the same algorithm presented in Section 4.1, a Cortex-M0 IP can target the inductive application.

Cortex-M0 microprocessor implements the ARMv6-M Thumb instruction set [22]. The microprocessor has a 32-bit architecture. A multiplication instruction is included in the instruction set. The IP makes available two different implementation of the multiplier, one aiming at good performances and the other at area saving; the designer can choose which implementation better matches the requirements. The fast multiplication instruction lasts only one clock cycle, while the slow one lasts 32 cycles. Of course for this algorithm the better option is the small (i.e. slow) one.

The core of the Cortex-M0 also needs a flash memory for the program storing. A $1kB$ flash memory is for this application largely sufficient. No RAM memory for data storage is needed.

The chosen clock frequency for the Cortex-M0 core was 8 MHz. Not every feature could be easily implemented in code, especially because of timing constraints. This issue could be solved by instantiating in the design ad-hoc hardwired blocks for the functions not mapped on software; more details will be presented in Section 4.4, since this issue is in common with the SensASIP platform-based design approach.

The core of the Cortex-M0 synthesized at 8 MHz in 0.35μm HV CMOS technology reached a complexity of around 48000 equivalent gates. In terms of silicon area, the above digital macrocell can be estimated to about 2.1 mm^2. This result, compared with the one of the hardwired design, is coherent with the area overhead due to the software-based approach.

4.4. SensASIP platform-based design of inductive position sensor

4.4.1. Instruction set definition

Starting from the DSP algorithm described in Section 4.1, ISAEP platform has been employed to map the signal processing operations in an assembly code. The system required an OTP memory to store the processing parameters: the parameter memory interface had to be implemented and two ad-hoc instructions (i.e. `otprd` and `otpwr`) had to be included in the Instruction Set.

From a preliminary performance analysis the minimum bit-width of 16 for the microprocessor architecture was chosen to prevent the signal processing to have overflow cases. The 16-bit architecture

can be obtained also from signal range and other signal processing parameters weight considerations (ADC data are 12-bit wide, as stated in Section 4.1, and some OTP data are 15-bit unsigned); in a N-bit two's complement fixed-point algorithm the MSB is used for the sign information.

The processing chain required three multiplication stages; writing a procedure of sum and shifting for each one of them would have been too much time and code wasting (if N is the bit-width of the current implementation, at least $2N$ sums and $2N$ shifts would be necessary for a multiplication software implementation). For this reason a `mul` instruction was included.

The preliminary algorithm code run in ISAEP required 60 rows. At this stage it was not significant to precisely map algorithm in terms of parameter-dependent program flow and number of cycles for extended instructions. The Instruction set was defined and we got root architecture information. A more precise simulation had anyway to be run in BT/CT platform.

4.4.2. Firmware design

The complete firmware design and the final system configuration are design steps to be completed in BT/CT environment.

System specification. The starting points were the preliminary ISAEP code and the system specifications. The device redesign should finally have had the same functionality of the hardwired design, presented in Section 4.2. It is not always easy to remap features thought for an hardwired design in a low level assembly code. Bit-wise conditional operations resulted in particularly long code portions, with respect to a MUX-based HDL code. By the way the code could sequentially perform all these operations. For instance, an OTP safety CRC (Cyclic Redundancy Check) was implemented in only 23 code rows, by employing conditional jump instructions (i.e. `beq`, `bne`, `bgt`) to save program size. The same feature in hardwired design occupies an area of around 24000 μm^2.

The `mul` instruction has been implemented as MCI; the multiplication was executed in two clock cycles, for gate level netlist timing constraints and area constraints in target technology.

Reused function blocks. The following features were chosen to be not implemented in software. Their hardwired implementations were kept in the SensASIP-based design:

- transmission block, because many events at the carrier frequency would have kept the processor busy too much time; the transmission and the DSP functions must run simultaneously;
- receiver block, for performances issues: the counting operation could have been allocated to on-board timer modules, but the clock frequency chosen was not enough for an accurate counting;
- communication block for the packet building in transmission and receiving;
- test block for hardware test configuration.

I/O registers. Each of the blocks in the previous paragraph had to be interfaced to the microprocessor. The registers defined for this purpose were called `commo1`, `commo2`, `commi1`, `commi2` for communication I/O (the communication protocol required 20 bits for each packet, so only one 16-bits register was not enough), `txcnf` for the transmission block configuration, `adc1` and `adc2` for the receiving counter block. Other application specific registers were called `ana1`, `ana2` and `ana3` for the analog I/O interface, and `otprg` for OTP special functions addressing.

Clock frequency. From preliminary code, system specifications and technology considerations it was possible to choose the microprocessor clock frequency. The most important timings were the DSP throughput constraint and the real time diagnostic requirements.

The BT/CT model environment has enabled the extraction of key parameters such as the maximum and minimum duration of a DSP interrupt handler performing. For this application the maximum handler duration was measured to be 156 clock cycles. Since a system requirement is a DSP elaboration time around 65 μs, the minimum clock frequency to meet it is around 2.5 MHz. Since the most critical functions were maintained in hardwired version, and in order to keep a safe margin for other functional safety requirements, the clock frequency was chosen to be 8 MHz.

Pushing the clock frequency up to 16 MHz would have been a critical choice, for power consumption issues, and for timing constraint too, being the technology forced to be 0.35 μm HV CMOS for benchmarking the hardwired design in the same technology conditions.

Firmware code size. The final firmware code had the length of 300 rows. The parsing operation in ROM binary content was done in $512x32$ ROM, in order to preserve the possibility to change the program in case of bug found or system re-specification.

The DSP core algorithm was coded in a 165-row software. It was possible to measure the side features software overhead in this case: 82%.

The entire firmware design required 3 man-weeks effort; the software design can be easily parceled, since the software architecture is based on interrupt handlers, for the different events occurring, while the `main` function can be easily coded as a sleep instruction, once the initialization of the register is finished.

4.4.3. RTL hardware configuration and verification

The HDL code of the microprocessor was a ready IP, only the pipelined multiplication block had to be added. RAM blocks had to be removed and the ROM and OTP had to be correctly sized. The register file size was set 31 16-bit registers.

The multiplier was designed with a matrix sign-detection architecture in two pipeline stages.

Once the design is completed, the simulations had only to verify that the parameters had been correctly chosen and no macroscopic errors had been made. The correctness of the software had already been verified in BT/CT model simulations.

For the final RTL release also the hardwired blocks from the original design had to be instantiated.

This RTL design step required 2.5 man-weeks effort. The verification can be completed in few hours, once a benchmark simulation environment had been identified. A suitable setup stimulates all the instances of the current design.

4.4.4. Synthesis

The synthesis scripts were part of the RTL design: many configurations have been tried and a stable synthesis flow of the microprocessor was available. The only customization of the scripts are the addition of the application specific hardwired blocks, the removal of the not used HDL blocks and relative libraries (e.g. for the RAM).

The synthesis for this redesign was made in 0.35 μm high voltage CMOS technology by ams AG. The company also provided models and libraries for the memories.

The synthesis flow included the design for testability (DFT): a single scan chain at the same microprocessor clock frequency was instantiated.

4.5. Implementation results

4.5.1. Performance, area and power consumption results

As already mentioned, in the worst case a DSP algorithm code run lasted in the worst case 156 8 MHz-clock cycles. The worst case was taking place when all the branches jumped to the longer code portions. The output results were case-by-case equal to the results of the hardwired DSP output. This demonstrates that the register- and bus-width choice had been made correctly.

The synthesis of the RTL design led to a standard-cell based netlist. The area report for the 0.35 μm HV CMOS technology, in worst case conditions (2.73 V supply voltage) and a clock frequency margin of 2 MHz (the design can work up-to 10 MHz), showed a microprocessor core complexity of around 20 equivalent kGates. The area of the entire microprocessor, including ROM for program storing and OTP memory, resulted 1.244 mm^2. The scan chain insertion and scan constraints setting caused a 10% area overhead. This data don't include the hardwired blocks complexity, which leads in total to around 0.19 mm^2. The same design synthesized in 0.18 μm CMOS technology showed a core area of around 245000 μm^2.

A power consumption estimation has been extracted from gate-level netlist simulation. Two switching activity files have been generated: one for a program for which the microprocessor never entered sleep mode, and one for a case in which the microprocessor had been kept in sleep mode for 98% of the time: it had been waked up only in case of receiving data interrupt occurring, for a signal processing handler run.

The core of the microprocessor showed a power consumption in worst conditions (3.3 V power supply) as in Table 1. The sleep mode allows a power save of 42.12%.

Sleep mode	Wake mode
3.543 mW	6.121 mW

Table 1: Power consumption, 0.35 μm HV CMOS technology, power supply at 3.3 V.

4.5.2. Results comparison

In this section we will summarize a comparison in terms of area, performances and power consumption for the same inductive sensor conditioning application, designed with different methods.

Table 2 summarizes complexity results, obtained for 0.35 HV CMOS technology libraries. The Cortex-M0 implementation result is a low estimation: the flash memory for the program was not taken into account.

Hardwired	Cortex-M0	SensASIP
1.09 mm^2	$>$ 2.3 mm^2	1.43 mm^2

Table 2: Implementation area detail of inductive sensor application.

SensASIP complexity overhead is limited to 31.2%, while a software-based solutions with a 32-bit Cortex-MO has an overhead higher than 100%.

Figure 4: Hardwired vs SensASIP area comparison in details. On vertical axis, design complexity in equivalent kGates is shown.

Performances comparison can be conducted in terms of speed in returning a DSP result and accuracy of result with respect to the hardwired design output.

The original design processing lasts 65.5 μs with DSP clock frequency of 2 MHz. The most critical operations were the multiplications, lasting 22 clock cycles each one. A microprocessor algorithm run needs around 160 clock cycles. Being the clock frequency 8 MHz, the time to perform the signal processing is around 20μs. This timing improvement is a good feature to have a margin in a sequential architecture, for diagnostic purposes. In terms of number of clock cycles for a complete data processing, the microprocessor architecture overhead has been measured to be around 20%.

The SensASIP implementation guarantees DSP results with no error with respect to the hardwired design, thanks to the ease of verification in software design phase. The same is true for the Cortex-M0 implementation, being a 32-bits logic architecture.

Power consumption for the hardwired design was estimated around 2.184 mW. The power consumption of the SensASIP implementation, with a cautious use of the sleep mode, was estimated 34.2% larger than the original. The Cortex-M0, being a considerably more complex design, was estimated to have a power consumption overhead of around 75%.

In figure 4 a block-by-block complexity chart is shown; it is possible to visualize the area overhead of the SensASIP-based architecture.

5. 3D Hall position sensor case study

SensASIP platform has been successfully adopted for remapping a 3D Hall's effect sensor conditioning algorithm, which is one of the most complex applications in the field [23]. The 3D magnetic sensor signal processing needs some correction stages for an accurate magnetic field calculation (correction of offset, sensitivity and their drift over temperature) which require sum, shift and multiplication; a CORDIC block for a spherical coordinates position calculation; memory accesses to get processing parameter values; some linearizations to map the position values in the correct ranges.

5.1. Hardwired design

Figure 5 shows the block diagram of the complete system. The new tasks, implemented from scratch in the SensASIP firmware, are the following: signal processing, FSM, result storing and sensor control blocks; they are depicted in red in Figure 5. The other blocks, such as delta-sigma ADC filtering, communication, analog switching, main control and power management have been reused from original design. The DSP block was clocked at 8 MHz. It is an FSM-programmable unit: its behavior depends on current processing configuration; its architecture is based on resources sharing: CORDIC, normalization, multiplication blocks are used many times during the same processing event. The design included an E^{2}PROM memory for the processing parameters storing.

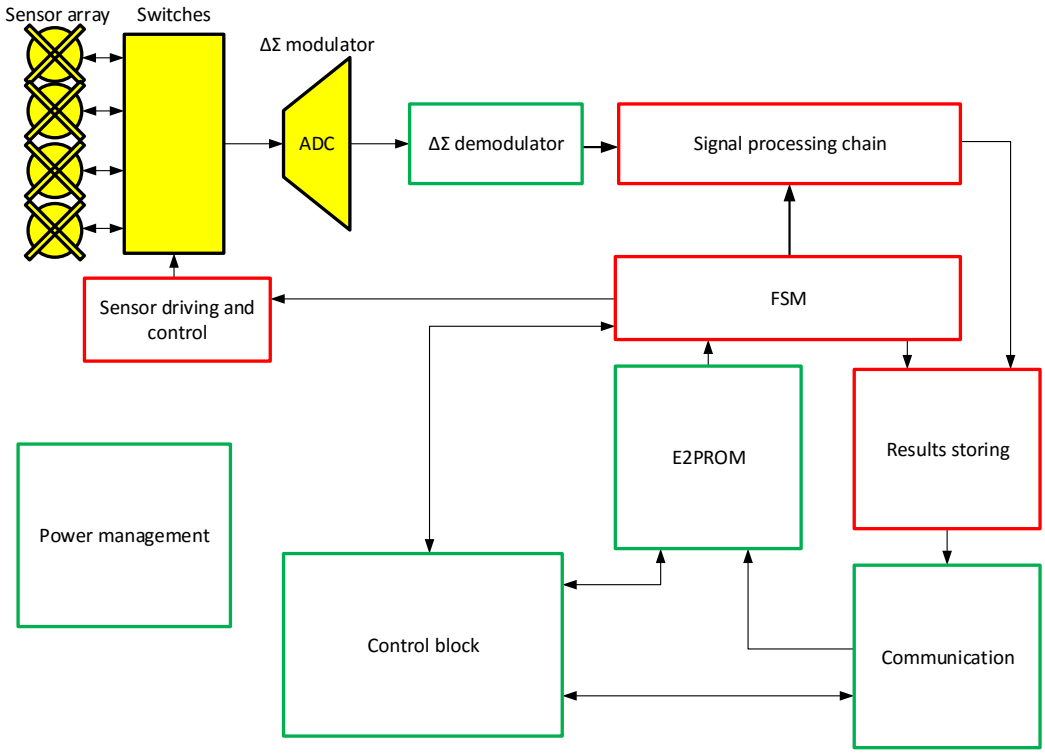

Figure 5: Magnetic position sensor system block scheme.

The area of the hardwired implementation was reported to be around 3 mm^2, including E^{2}PROM memory, in 0.35 μm HV CMOS technology. With respect to the application presented in Section 4, the 3D magnetic sensor requires a signal processing unit much more complex.

5.2. SensASIP platform-based design of 3D magnetic position sensor

For this application, the instruction set definition led to the implementation of a CORDIC custom operation, whose instruction has been called `atan`; multiplication operation has been targeted by a `mul` instruction, as in Section 4 application. The E^{2}PROM memory interfacing instruction had to be defined, too.

Firmware design, hardware definition and timing evaluation were performed through BT/CT platform. The complete firmware resulted in a 750-row code. Greatest part of it was the signal processing algorithm implementation in software (650 rows). The SensASIP microprocessor resulted to need a data memory (RAM, size $128x16$), a program ROM of 1024 words (to have some margin in case of firmware extension) and the same E^2PROM of the original hardwired design. I/O specific registers were defined. The processing timing overhead was 10 μs in worst case (total 47.5 μs for SensASIP implementation at 8 MHz clock); being the output update frequency 4 kHz, the processing time was largely suitable for the application.

The RTL configuration required some effort for the CORDIC unit design. The `atan` instruction has been implemented as a 7-cycle MCI. The CORDIC operation reused some of the modular ALU sum/subtraction sub-blocks already present, in order to preserve a low complexity as much as possible. The multiplication blocks were reused from the design in 4.4.3.

5.3. Results for 3D magnetic sensor

For 3D magnetic sensor application we obtained a SensASIP core area of around 0.86 mm^2: with respect to Section 4.5 a complexity increase of the microprocessor core (+10%) was reported. This is due to the CORDIC block instantiation inside the extended-ALU. The ROM element occupied an area of 0.37 mm^2; the ROM area increment (+28%), with respect to inductive sensor application, has been caused by a more complex algorithm code and consequently a larger program memory. Also the RAM instantiation must be taken into account (0.24 mm^2). Figure 6 shows a block-by-block complexity chart for hardwired vs SensASIP designs. The area overhead of the SensASIP microprocessor approach, with respect to hardwired design, is in this case 2.41%.

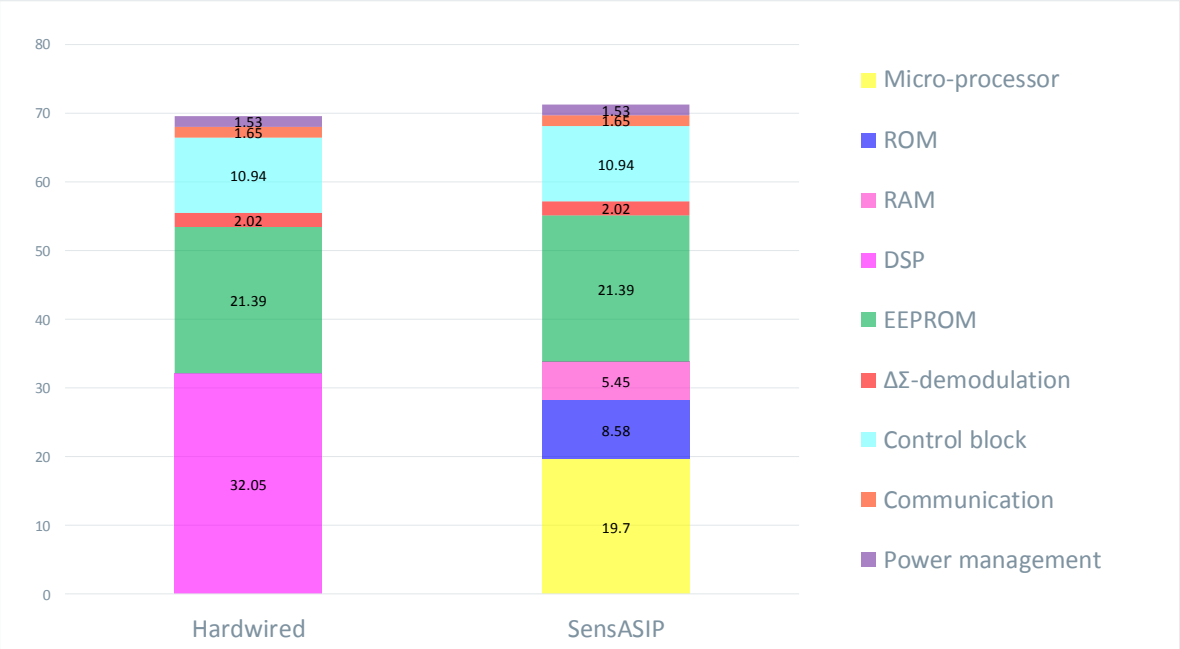

Figure 6: Magnetic sensor solutions area comparison in details. On vertical axis, design complexity in equivalent kGates is shown.

6. Conclusions

In this paper SensASIP, a design-aiding platform for sensor conditioning devices has been presented. The target IP for signal processing and system control is a three-pipeline stage microprocessor; its development had been carried out with strict area and flexibility constraints, significant cost parameters in automotive market.

The platform enables a three-step design flow (Section 3): the Instruction Set definition and a design space exploration with ISAEP; a firmware design through the bit-true and cycle-true model, which returns accurate timing and resources allocation results, too; an hardware configuration on parametric and modular RTL design for final verification and back-end implementation.

The SensASIP design methodology can target families of sensor conditioning devices for automotive applications, such as inductive, capacitive and magnetic sensor families. SensASIP has been employed to redesign an inductive position sensor conditioning device. The design effort was around 2 man-month. The same design in hardwired development strategy can require six man-month. With respect to hardwired design SensASIP presented an area overhead of 31.2% (Section 4.5 for detailed results). Performances were comparable in terms of processing results, and the clock cycles overhead was less than 20%. It has successfully been employed to target a 3D magnetic sensor conditioning application, too (Section 5). In this case the area overhead was 2.41%. By the results of other applications SensASIP-based design, not shown in this paper, we observed that in general with system complexity increasing, the area overhead decreases.

The same application can be targeted with third-part microprocessor IP, such as ARM Cortex-M0. The problem in this case is a large area overhead and an architecture which is possibly oversized and fixed. SensASIP allows a full customization of the I/O register, as shown in Section 4; when third-part IP must be embedded in application specific circuits, it may be an hard task to interface the standard I/O of the microprocessor. Another mentioned design method is the automatic ASIP generation. This strategy enables a microprocessor architecture customization, and custom instruction definition, but a deep optimization in terms of area is not possible.

The comparative analysis in Section 4 and Section 5 is fair, since the results of the original designs, and that of the new SensASIP approach, are compared at the same level of the design flow.

The proposed SensASIP platform, with respect to other design paradigms (hardwired, ASIP- and microprocessor-based) shows a suitable flexibility/complexity overhead trade-off in sensor conditioning for automotive field.

Acknowledgments. The authors would like to thank the ams team in Pisa for their support in development of the design platform. The TeTraCom project (Technology Transfer in Computing Systems), funded by the European Union Commission FP 7 - Contract No. 609491, made the entire work possible, through a financial support.

References

[1] Mike Pinelis. Automotive sensors and electronics: trends and developments in 2013. Automotive Sensors and Electronics Expo, Detroit, MI, USA, 2013.

[2] A. Qamar, C. Passerone, L. Lavagno, and F. Gregoretti. Design space exploration of a stereo vision system using high-level synthesis. In *Mediterranean Electrotechnical Conference (MELECON), 2014 17th IEEE*, pages 500–504, April 2014. doi: 10.1109/MELCON.2014.6820585.

[3] S.K. Kumaran and K.R. Santha. DSP control of DC/DC boost converter for fuel cell based hybrid electric vehicle. In *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, pages 409–414, March 2012.

[4] L. Sarti, A. Sisto, L. Pilato, L. Di Piro, and L. Fanucci. Platform based design of 3D Hall sensor conditioning electronics. In *Ph.D. Research in Microelectronics and Electronics (PRIME), 2015 11th Conference on*, pages 200–203, June 2015. doi: 10.1109/PRIME.2015.7251369.

[5] W. Ecker, M. Velten, L. Zafari, and A. Goyal. Metasynthesis for designing automotive SoCs. In *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pages 1–6, June 2014.

[6] J. Andersson. A DSP ASIC design flow based on VHDL and ASIC-emulation. In *Design Automation Conference, 1995, with EURO-VHDL, Proceedings EURO-DAC '95., European*, pages 562–567, Sep 1995. doi: 10.1109/EURDAC.1995.527460.

[7] Zbigniew Hajduk. An FPGA embedded microcontroller. *Microprocessors and Microsystems*, 38(1):1 – 8, 2014. ISSN 0141-9331. doi: http://dx.doi.org/10.1016/j.micpro.2013.10.004.

[8] Michal Chovanec and Peter Sarafin. Real-time schedule for mobile robotics and WSN aplications. In *Computer Science and Information Systems (FedCSIS), 2015 Federated Conference on*, pages 1199–1202, Sept 2015. doi: 10.15439/2015F146.

[9] Liu Xin, Sun Ling, Shi Quan, Chen Changzhu, and Cao Xiaoqiang. Temperature controlling system for LED lighting based on ARM. In *Consumer Electronics, Communications and Networks (CECNet), 2013 3rd International Conference on*, pages 649–652, Nov 2013. doi: 10.1109/CECNet.2013.6703414.

[10] P. Pannuto, Yoonmyung Lee, B. Kempke, D. Sylvester, D. Blaauw, and P. Dutta. Demo: Ultra-constrained sensor platform interfacing. In *Information Processing in Sensor Networks (IPSN), 2012 ACM/IEEE 11th International Conference on*, pages 147–148, April 2012. doi: 10.1109/IPSN.2012.6920923.

[11] P.S. Kedareswar and V. Krishnamoorthy. A CAN protocol based embedded system to avoid rear-end collision of vehicles. In *Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2015 IEEE International Conference on*, pages 1–5, Feb 2015. doi: 10.1109/SPICES.2015.7091439.

[12] K. Takaya. Transputer-like Multicore Digital Signal Processing on the Array of ARM Cortex-M0 Microprocessors. In *Embedded Multicore Socs (MCSoC), 2012 IEEE 6th International Symposium on*, pages 45–50, Sept 2012. doi: 10.1109/MCSoC.2012.14.

[13] R.E. Gonzalez. Xtensa: a configurable and extensible processor. *Micro, IEEE*, 20(2):60–70, Mar 2000. ISSN 0272-1732. doi: 10.1109/40.848473.

[14] A. Hoffmann, O. Schliebusch, A. Nohl, G. Braun, O. Wahlen, and H. Meyr. A methodology for the design of application specific instruction set processors (asip) using the machine description language lisa. In *Computer Aided Design, 2001. ICCAD 2001. IEEE/ACM International Conference on*, pages 625–630, Nov 2001. doi: 10.1109/ICCAD.2001.968726.

[15] S. Sabatelli, M. Galgani, L. Fanucci, and A. Rocchi. A double-stage Kalman filter for orientation tracking with an integrated processor in 9-D IMU. *Instrumentation and Measurement, IEEE Transactions on*, 62(3):590–598, March 2013. ISSN 0018-9456. doi: 10.1109/TIM.2012.2218692.

[16] G. Goossens, D. Lanneer, W. Geurts, and J. Van Praet. Design of asips in multi-processor socs using the chess/checkers retargetable tool suite. In *System-on-Chip, 2006. International Symposium on*, pages 1–4, Nov 2006. doi: 10.1109/ISSOC.2006.321968.

[17] Hector Posadas, Alejandro Nicols, Pablo Peil, Eugenio Villar, Florian Broekaert, Michel Bourdelles, Albert Cohen, Mihai T. Lazarescu, Luciano Lavagno, Andrei Terechko, Miguel Glassee, and Manuel Prieto. Improving the design flow for parallel and heterogeneous architectures running real-time applications: The {PHARAON} {FP7} project. *Microprocessors and Microsystems*, 38(8, Part B):960 – 975, 2014. ISSN 0141-9331. doi: http://dx.doi.org/10.1016/j.micpro.2014.05.003.

[18] Yao-Hua Chen, Chia-Pin Chen, Pei-Wei Hsu, Chunfan Wei, Wei-Min Cheng, Hsun-Lun Huang, Tai-Yuan Cheng, and A.Y.P. Chen. A novel multicore sdr architecture for smart vehicle systems. In *ITS Telecommunications (ITST), 2012 12th International Conference on*, pages 275–279, Nov 2012. doi: 10.1109/ITST.2012.6425181.

[19] K. Keutzer, A. R. Newton, J. M. Rabaey, and A. Sangiovanni-Vincentelli. System-level design: orthogonalization of concerns and platform-based design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(12):1523–1543, Dec 2000. ISSN 0278-0070. doi: 10.1109/43.898830.

[20] W.J. Fleming. New automotive sensors - a review. *Sensors Journal, IEEE*, 8(11):1900–1921, Nov 2008. ISSN 1530-437X. doi: 10.1109/JSEN.2008.2006452.

[21] A. Sisto, L. Pilato, R. Serventi, and L. Fanucci. A design platform for flexible programmable DSP for automotive sensor conditioning. In Jim Torresen, Snorre Aunet, and Tor Sverre Lande, editors, *Nordic Circuits and Systems (NORCAS), 2015 IEEE 1st Conference*, pages 12–16, October 2015. ISBN 978-1-4673-6575-8. doi: CPF15828-USB.

[22] ARM Limited . ARM Infocenter. `http://infocenter.arm.com/help/index.jsp`, 2015. Accessed: 2015-09-07.

[23] J. Bretschneider, A. Wilde, P. Schneider, H.-P. Hohe, and U. Koehler. Design of multi-dimensional magnetic position sensor systems based on hallinone technology. In *Industrial Electronics (ISIE), 2010 IEEE International Symposium on*, pages 422–427, July 2010. doi: 10.1109/ISIE.2010.5637864.