

# LOOKING AT PROCESSORS FOR THE SEWING SMART SENSOR

V. PARISI I BARADAD, S. BERMEJO, J. CABESTANY

UNIVERSITAT POLITECNICA DE CATALUNYA (UPC), SPAIN

**KEYWORDS: Hardware architectures, Blind Source Separation, Smart Sensors.**

**ABSTRACT:** The *System for European Water Monitoring* (SEWING) project requires a hardware design which satisfies the computational requirements of Blind Source Separation (BSS) techniques, runs on low power and has both analog and digital input/output capabilities. In this paper we explore two processor solutions, which represent the most suitable candidates in terms of Digital Signal Processor (DSP) and Microcontroller Unit (MCU).

## INTRODUCTION

Smart means intelligent, sharp, effective, accurate... an entire row of adjectives that affect the choice of a processor, for electronic sensors, features that must be translated into measurable characteristics when we specify our design.

The most preferable situation would be not to have to choose, but this would only happen if only one device fulfilled our needs, which, due to competence is not the normal case. Thus, we will explore two different alternatives, DSP and MCU, for the SEWING smart sensor, which come from worlds which were unassociated several years before, but now tend to appear sharing common resources.

The SEWING project looks for a transducer demonstrating intelligence, the criteria being; adaptive algorithms and learning competences, the ability to allow interaction with the exterior by reading Ion Sensitive Field Effect Transistor (ISFET) measures and transmitting alarms, the cheapest price possible, and the minimum wastage of energy, since it must be portable and expendable, if possible.

Starting from the processing perspective, in the last paragraph, one tends to settle on DSPs as a common solution for adaptive algorithms. These processors were initially conceived to satisfy the numerical demands of signal treatment, based on Harvard architecture which allows efficient computations, and have been evolving to offer additional input/output capabilities as a response to market demands. Nowadays several devices are available, announced as *low power*, at very attractive prices.

On the other hand, if we choose power as a primary characteristic, we'll surf above the MCUs market, finding devices really ecologic at a cost of several cents of Euro. They were designed initially to integrate versatile input/output peripherals and lately they have been

incorporating more complex Central Processing Units, in this way they are an alternative to DSPs.

An initial exploration of the market leads us to the c5000 as the most preferred DSP and to the MSP430 as the most appropriate MCU, both families of devices come from Texas Instruments (TI).

We begin by presenting briefly the adaptive BSS algorithms, which at this moment, seem the most appropriate for our purposes. Then we introduce two suitable processors (the TMS320c55xx DSPs and the MSP340x1xx MCUs), showing their characteristics, as they appear in the TI data sheets, with the intention of clarifying which one best suits our needs. We conclude with the computational requirements of BSS, which in fact, open the discussion to choose the processor..

## ADAPTIVE BSS ALGORITHMS

In the classical blind source separation (BSS) model [1], we have discrete-time signals  $x_1[n], \dots, x_m[n]$  that have been generated using a linear mixture of  $p$  source signals  $s_1[n], \dots, s_p[n]$ , i.e.

$$\mathbf{x}[n] = (x_1[n] \dots x_m[n])^t = \mathbf{A} \mathbf{s}[n] \quad (1)$$

where  $\mathbf{A}$  is known as the  $m \times p$  "mixing matrix" and  $t$  denotes transpose. According to the above model and given a temporal window of the observable vector  $\mathbf{x}$ , i.e.  $\mathbf{D}_1 = \{x_i[n], n=1, \dots, T, i=1, \dots, m\}$ , we must compute a  $p \times m$  separating matrix  $\mathbf{B}$  which allows a estimation of the source signals  $\{s_i[n], i=1, \dots, p\}$  using the following reconstruction algorithm,

$$\mathbf{y}[n] = (y_1[n] \dots y_p[n])^t = \mathbf{B} \mathbf{x}[n] \approx \mathbf{s}[n] \quad (2)$$

Clearly, the solution to the BSS reconstruction problem is  $\mathbf{B} = \mathbf{A}^{-1}$  if no noise is assumed.

The point of departure for computing  $\mathbf{B}$  is the assumption the source signals  $\mathbf{s}=(s_1 \dots s_n)^T$  are independent, which is often true in an array of sensors since unrelated physical information can be detected. If  $\mathbf{s}$  is formed by independent random variables, then its pdf can be expressed as the product of the marginal distributions ([2] p.132),

$$f_{\mathbf{s}}(\mathbf{s}) = \prod_{i=1}^p f_{s_i}(s_i) \quad (3)$$

In order to compute  $\mathbf{B}$  using  $\mathbf{D}$ , an objective (or risk) function must be defined and minimized by the BSS learning algorithm. The Kullback-Leibler (KL) divergence is a natural candidate for this purpose since it measures the divergence between two probability distributions  $p_y(\mathbf{y})$  and  $q(\mathbf{y})$  as follows,

$$\text{KL}[p_y(\mathbf{y})||q(\mathbf{y})] = \int p_y(\mathbf{y}) \log \frac{p_y(\mathbf{y})}{q(\mathbf{y})} d\mathbf{y} \quad (4)$$

Note that  $\text{KL}=0$  if and only if  $p=q$  and  $>0$  otherwise. Hence, if  $p_y(\mathbf{y})$  is the pdf of the reconstructed signal and  $q(\mathbf{y})$  is the probability of the source signals  $\prod_{i=1}^p f_{s_i}$ , the KL divergence will measure how close  $\mathbf{y}=\mathbf{B}\mathbf{x}$  is to the original source  $\mathbf{s}$ . It can be shown [3] that Equation (4) can be estimated using the set of samples  $\mathbf{D}_T$  as

$$\begin{aligned} R(\mathbf{B}) &= \hat{\text{KL}}[p_y(\mathbf{y})||q(\mathbf{y})] = -\frac{1}{2} \log[\det(\mathbf{B}\mathbf{B}^t)] - \\ & - \frac{1}{T} \sum_{n=1}^T \sum_{i=1}^p \log q_i(y_i[n]) \end{aligned} \quad (5)$$

where  $\det \mathbf{B}$  is the determinant of  $\mathbf{B}$  and  $R(\mathbf{B}) \rightarrow \text{KL}[p_y(\mathbf{y})||q(\mathbf{y})]$  as  $T \rightarrow \infty$ . However, in order to derive an on-line learning algorithm [4], we must use the instantaneous empirical estimate of Equation (4), which only uses one training sample,

$$R(\mathbf{B}[n]) = -\frac{1}{2} \log[\det(\mathbf{B}[n]\mathbf{B}[n]^t)] - \sum_{i=1}^p \log q_i(y_i[n]) \quad (6)$$

Thus, we can apply the stochastic gradient descent method to compute  $\mathbf{B}$ ,

$$\mathbf{B}[n+1] = \mathbf{B}[n] - \eta[n] \frac{\partial R(\mathbf{B}[n])}{\partial \mathbf{B}[n]} \quad (7)$$

where  $\eta[n]$  is the step size function of the learning algorithm. It is worth noting that using the stochastic approach storing the entire set  $\mathbf{D}_T$  in the memory is avoided and only sample  $\mathbf{x}[n]$  is stored. However, in order to ensure a good convergence of the algorithm, it is

desirable to store as much training samples as possible when performing the on-line approach using a sample procedure (e.g. a cyclic sampling) over  $\mathbf{D}_T$ .

As it is shown in [3], Equation (7) gives

$$\mathbf{B}[n+1] = \mathbf{B}[n] - \eta[n] \left\{ \left( \mathbf{B}^t[n] \right)^{-1} - \mathbf{f}(\mathbf{y}[n]) \mathbf{x}^t[n] \right\} \quad (8)$$

where  $\mathbf{f}(\mathbf{y}[n]) = (f_1(y_1[n]) \dots f_p(y_p[n]))^t$  is obtained from  $q_i(y_i)$  as

$$f_i(y_i) = -\frac{d \log(q_i(y_i))}{dy_i} = -\frac{dq_i(y_i)/y_i}{q_i(y_i)} \quad (9)$$

Observe that Equation (8) involves the computation of the inverse of the matrix  $\mathbf{B}^t[n]$ , which can be time-consuming since we usually apply an iterative algorithm in order to compute  $\{\mathbf{B}^t[n]\}^{-1}$  numerically. On the other hand, it has been observed [5] that the ordinary gradient descent does not work for non-Euclidean spaces since the descent direction in such a situation is represented by the usual gradient direction multiplied by the inverse of the Riemannian metric  $\mathbf{G}(\mathbf{B})$ . In BSS,  $\mathbf{G}^{-1}(\mathbf{B})$  can be easily computed and then Equation (7) can be modified as

$$\mathbf{B}[n+1] = \mathbf{B}[n] - \eta[n] \frac{\partial R(\mathbf{B}[n])}{\partial \mathbf{B}[n]} \mathbf{B}^t[n] \mathbf{B}[n] \quad (10)$$

Equation (10) is known as the natural gradient descent learning algorithm for BSS [3], which gives

$$\mathbf{B}[n+1] = \mathbf{B}[n] - \eta[n] \left\{ \mathbf{I} - \mathbf{f}(\mathbf{y}[n]) \mathbf{y}^t[n] \right\} \mathbf{B}[n] \quad (11)$$

where  $\mathbf{I}$  denotes the identity matrix.

## THE TMS320c55xx DSPs

The C55x<sup>TM</sup> DSP architecture [6] achieves high performance and low power through increased parallelism and total focus on reduction in power dissipation. The CPU supports an internal bus structure that is composed of one program bus, three data read buses, two data write buses, and additional buses dedicated to peripheral and DMA activity. These buses provide the ability to perform up to three data reads and two data writes in a single cycle. In parallel, the DMA controller can perform up to two data transfers per cycle independent of the CPU activity.

The C55x CPU provides two multiply-accumulate (MAC) units, each capable of 17-bit x 17-bit multiplication in a single cycle. A central 40-bit arithmetic/logic unit (ALU) is supported by an additional 16-bit ALU. Use of the ALUs is under instruction set control, providing the ability to optimize parallel activity

and power consumption. These resources are managed in the Address Unit (AU) and Data Unit (DU) of the C55x CPU. The Instruction Unit (IU) performs 32-bit program fetches from internal or external memory and queues instructions for the Program Unit (PU). The Program Unit decodes the instructions, directs tasks to AU and DU resources, and manages the fully protected pipeline.

Its general-purpose input and output functions and the 10-bit A/D provide sufficient pins for status, interrupts, and bit I/O for LCDs, keyboards, and media interfaces. The 5509 peripheral set includes an external memory interface (EMIF) that provides glueless access to asynchronous memories like EPROM and SRAM, as well as to high-speed, high-density memories such as synchronous DRAM. Additional peripherals include Universal Serial Bus (USB), real-time clock, watchdog timer, I2C multi-master and slave interface, and a unique device ID. Three full-duplex multichannel buffered serial ports (McBSPs) provide glueless interface to a variety of industry-standard serial devices, and multichannel communication with up to 128 separately enabled channels. The enhanced host-port interface (EHPI) is a 16-bit parallel interface used to provide host processor access to 32K words of internal memory on the 5509. The DMA controller provides data movement for six independent channel contexts without CPU intervention, providing DMA throughput of up to two 16-bit words per cycle. Two general-purpose timers, up to eight dedicated general-purpose I/O (GPIO) pins, and digital phase-locked loop (DPLL) clock generation are also included.

As a candidate, to compete against the MCU solution, we have chosen the TMS320vc5509 device, which has a very low consumption taking into account its impressive capabilities; its main features are shown below.

- 144-/200-MHz Clock Rate at 1.5 V
- One/Two Instruction(s) Executed per Cycle
- Dual Multipliers (Up to 400 MMACS)
- 128K x 16-Bit On-Chip RAM, Composed
- 8M x 16-Bit Addressable External Memory
- Programmable Low-Power Control
- On-Chip Scan-Based Emulation Logic
- On-Chip Peripherals
- Two 20-Bit Timers and a Watchdog Timer
- 64-Bit Unique Device ID
- Up to 3 Multichannel Buffered Serial Ports (McBSPs)
- Programmable Digital Phase-Locked Loop
- Seven (LQFP) or Eight (BGA) General-Purpose I/O
- USB Full-speed (12 Mbps) Slave Port With Isochronous Transfer Support
- Inter-Integrated Circuit (I2C) Multi-Master and Slave Interface
- Real-Time Clock (RTC)

- 2-Channel (LQFP) or 4-Channel (BGA) 10-bit Successive Approximation A/D
- IEEE Std 1149.1 (JTAG) Boundary Scan Logic
- 2.5-V – 3.6-V I/O Supply Voltage
- 1.5-V Core Supply Voltage

## THE MSP340x1xx MCUs

The Texas Instruments MSP430 series [7] is an ultralow-power microcontroller family consisting of several devices featuring different sets of modules targeted to various applications. The microcontroller is designed to be battery operated for use in extended-time applications, it consumes less than 400  $\mu$ A in active mode operating at 1 MHz in a typical 3-V system.

The MSP430 achieves maximum code efficiency with its 16-bit RISC architecture, 16-bit CPU-integrated registers, and a constant generator. The digitally-controlled oscillator provides wake-up from low-power mode to active mode in less than 6  $\mu$ s. The MSP430x13x and the MSP430x14x series are microcontroller configurations with two built-in 16-bit timers, a fast 12-bit A/D converter, one or two universal serial synchronous/asynchronous communication interfaces (USART), and 48 I/O pins.

Additionally, the MSP430x1xx family has an abundant mix of peripherals and memory sizes enabling true system-on-a-chip designs. The peripherals include, multiple timers (some with capture/compare registers and PWM output capability), on-chip clock generation, H/W multiplier, USART(s), Watchdog Timer, GPIO, and others.

The MSP430 employs a von-Neumann architecture, therefore, the entire memory and peripherals are in one address space, and a reduced instruction set is applicable to all functional blocks.

The CPU consists of a 16-bit arithmetic logic unit (ALU), 16 registers, and instruction control logic. Instructions fetched from the program memory are always 16-bit accesses, whereas data memory can be accessed using word (16-bit) or byte (8-bit) instructions. Any access uses the 16-bit memory data bus (MDB) and as many of the least-significant address lines of the memory address bus (MAB) as required to access the memory locations. The data memory is connected to the CPU through the same two buses as the program memory (ROM): the memory address bus (MAB) and the memory data bus (MDB). The data memory can be accessed with full (word) data width or with reduced (byte) data width. Peripheral modules are connected to the CPU through the MAB, MDB, and interrupt service and request lines. The MAB is usually a 5-bit bus for most of the peripherals. The MDB is an 8-bit or 16-bit bus. Most of the peripherals operate in byte format. Modules with an 8-bit

data bus are connected by bus-conversion circuitry to the 16-bit CPU.

As a candidate, to compete against the DSP solution, we have chosen the MSP430F1xx devices, which offer a valuable Flash memory; its main features are shown below.

- Low Supply-Voltage Range, 1.8 V...3.6 V
- Ultralow-Power Consumption:
- Standby Mode: 1.6 uA
- RAM Retention Off Mode: 0.1 uA
- Low Operating Current:
  - 2.5 uA at 4 kHz, 2.2 V
  - 280 uA at 1 MHz, 2.2 V
- Five Power-Saving Modes
- Wake-Up From Standby Mode in 6 us
- 16-Bit RISC Architecture,
- 125-ns Instruction Cycle Time
- 12-Bit A/D Converter With Internal Reference, Sample-and-Hold and Autoscan Feature
- 16-Bit Timer With Seven Capture / Compare-With-Shadow Registers, Timer\_B
- 16-Bit Timer With Three Capture/Compare Registers, Timer\_A
- On-Chip Comparator
- Serial Onboard Programming,
- No External Programming Voltage Needed
- Programmable Code Protection by Security Fuse

## CONCLUSION

At first glance, we see that both, the DSP and the MCU, offer impressive computational strength, and what's more, they add analog and digital I/O capabilities which draw a possible complete solution in a chip.

Based on the SEWING technical specifications, we could also have chosen two processors without A/D converters, and we would be able to find devices without moving from these DSP and MCU families (C5000 and MSP430), but the main CPU characteristics would remain basically the same.

Taking into account, the computational requirements, equation (11) involves performing  $p[2m+p(1+m)]$  multiplications,  $pm(1+p)$  additions,  $p(p+m)$  subtractions and  $p$  non-linear transforms  $f_i(y_i)$ . For instance, if the true pdf of the sources is unknown, we can select  $f_i(y_i) = \alpha y_i + y_i |y_i|^2$  for sub-Gaussian source signals with negative kurtosis ([3], p.2034), which implies that the  $p$  non-linear operations are in fact  $3p$  multiplications and  $p$  additions. However other more complex functions can be employed, e.g.  $f_i(y_i) = \alpha y_i + \tanh(\gamma y_i)$  for super-

Gaussian sources and consequently additional computation will be needed.

The final choice will be based on the algorithm that will be implemented which is yet to be decided, but we can already point out the more feasible ways. The direction to follow will be clearer when we work further on the actual open questions.

## ACKNOWLEDGEMENTS

This work is being supported by the IST Programme No. 2000-28084 (SEWING) of the EU.

## THE AUTHORS

Vicenç Parisi i Baradad, Sergio Bermejo and Joan Cabestany, are with the Department of Electronic Engineering, Universitat Politècnica de Catalunya, C4 building, Jordi Girona 1-3, 08034 Barcelona.

E-mail: parisii@eel.upc.es

## REFERENCES

- [1] J.-F. Cardoso, "Blind Signal Separation: Statistical Principles", Proceedings of the IEEE, Vol. 86, No. 10, 1998, pp. 2009-2025
- [2] A. Papoulis, "Probability, Random Variables and Stochastic Processes", 3rd Edition, Singapore: McGraw-Hill, 1991.
- [3] S. Amari, A. Cichocki, "Adaptive Blind Signal Processing- Neural Network Approaches", Proceedings of the IEEE, Vol. 86, No. 10, 1998, pp. 2026-2048.
- [4] A. Benveniste, M. Métivier, P. Priouret, "Adaptive Algorithms and Stochastic Approximations", Berlin: Springer-Verlag, 1990.
- [5] S. Amari, "Natural gradient works efficiently in learning", Neural Computation, Vol. 10, 1998, pp. 251-276.
- [6] TMS325C55xx Technical Overview, SPRU393, Texas Instruments, 2000.
- [7] MSP430x1xx Family, SLAU049, Texas Instruments, 2002.