# Digital integrated circuit implementation of an identification stage for the detection of illegal hunting and logging

Carlos Salazar-García*, Luis Alfaro-Hidalgo*, Mauricio Carvajal-Delgado*, Jordan Montero-Aragón*, Reinaldo Castro-Gonzalez*, Juan Agustín Rodríguez†, Alfonso Chacon-Rodríguez*, and Pablo Alvarado-Moya*
*Escuela de Ingeniería Electrónica, Tecnológico de Costa Rica
URL: http://www.ie.itcr.ac.cr, Email: {csalazar, palvarado, alchacon}@itcr.ac.cr
†Instituto de Investigaciones en Ingeniería Eléctrica, IIIE (UNS-CONICET)
Departamento de Ingeniería Eléctrica y de Computadoras
Universidad Nacional del Sur, Bahía Blanca, Argentina
Email: rodrijuana@gmail.com

***Abstract*—Results of the VLSI implementation of an acoustic classification system's identification stage, intended for the detection of gunshots and chainsaws in a protected tropical area, are shown, with the idea of later building a surveillance wireless sensor network with similar nodes. The system performs from signal preprocessing to feature extraction, with results of the HDL description of the system tested on a FPGA against a golden reference, using real data taken from a protected rain forest area. Final classification of signals, using HMM, is in the final stages of testing. Some post-place-and-route results of the code ported to a commercial 130nm CMOS technology are also given.**

***Index Terms*—Acoustic signals recognition, embedded systems, dimensionality reduction, $kd$-tree, HMM, environmental protection, FPGA, ASIC, low-power CMOS.**

## I. Introduction

Theoretical results of a system for the detection of illegal activities in environmentally protected areas such as logging and hunting have been shown in [1]–[4]. The system, named SiRPA (Spanish acronym of "Acoustic Pattern Recognition System") is a hardware oriented pattern recognition pipeline designed to reduce the global false-positive detection rates of an acoustic sensor, while keeping the flexibility of its configuration for varying environments, and adjusting its architecture for the lowest energy consumption possible.

Partial results of SiRPA's implementation on a BeagleBoard-xM embedded platform have been shown in [2].These results demonstrate the feasibility of SiRPA's hardware structure. But the power requirements of this solution make it unsuitable to be integrated on a low power Wireless Sensor Network supposed to operate on battery power for long periods of time, in places where battery replacement may be next to impossible. The integration of SiRPA on a low power ASIC may circumvent such power limitations. A first step is then to translate the system's architecture to a HDL description and thoroughly validate its results, against the already tested sections of the architecture. The advantage of testing on a FPGA is that it allows for more exhaustive testing using vectors of data already tested on the embedded implementation of SiRPA, without the unavoidable long delays of post-place-and-route simulations. As the different sections of the architecture are ported to the intended CMOS technology, just a few validation simulations should be necessary to verify the correct translation of the code to silicon.

This paper is structured as follows: Section II exposes the steps taken to translate the C code description of SiRPA to Verilog, considering issues such as word format and parallelization of the sequential algorithms for their hardware implementation. Results from each sub-stage compared to output results from embedded system implementation – using a BeagleBoard-xM – are also given in this Section II. Hardware resources and power needs for the identification stage – both on a commercial FPGA and from place and route results on a standard cells library for a commercial CMOS 130nm process are given in Section III. Conclusions are offered in Section IV.

## II. Porting of SiRPA to HDL

The first step was to translate SiRPA's code to an HDL description, using Verilog with some open-source VHDL libraries (the architecture of SiRPA is given in Fig. 1). But testing the ported code using regular simulations resulted too time intensive, considering the complexities of the arithmetic operations involved, and the need to verify results against those of the prior software implementations. Therefore, it was decided to test each section of SiRPA on a FPGA, loading input vector data either from prior testings on the embedded implementation of the system, or from real gunshot and chainsaw audio signals taken from measurements on a tropical rain forest (see [5] for more details on the sound data base), and then verifying results of the output vectors on Matlab. A Digilent Inc.'s Nexys 4, with an Artix$^{TM}$ XC7A100T-CSG324 FPGA from Xilinx, was used as the testing platform. Serial and SPI interfaces were implemented on the board to connect with Matlab and the needed ADC. Input data was then fed either through the ADC (using the AGC and antialias conditioning depicted on the first block of Fig. 1), or as blocks of data coming from the serial interface. Results were stored on the board's RAM, and transmitted back to the PC for evaluation.

Regarding the Verilog implementation, several issues were taken into consideration, especially regarding data resolution (as floating point resolution would be extremely expensive if used in all SiRPA's sections), programmability (the ability to change parameters on the system after its fabrication), and the modification of the essentially sequential or recursive algorithms of each section, into their concurrent hardware versions.

## A. Word format

SiRPA was originally implemented on a BeagleBoard-xM, using C, where standard floating point data representation was used. Translating the system to a parallel implementation required readjusting numeric precision in order to used fixed point representation, but trying to keep resolution error low. After several tests on the bank filter, and checking for overflow and underflow conditions using either typical input signals from the sound database or a impulsive function, a signed fixed point representation in two's complement was decided, using 5 bits for the integer part and 19 bits for the fractional part. The fractional 19-bit precision gives a $1.9073466x10^{-6}$ LSB precision, which meant an error under 4% at SiRPA's filter bank output when contrasting results against its floating point implementation in C (see next subsection).

## B. Identification stage implementation

The identification (ID) stage comprises an 8 band dyadic filter, a dimensional reduction stage and a symbol generator (more details of each in [1]–[4].

*1) Filter bank:* The identification stage generates the observation chain by first passing the conditioned signal through an 8 band filter bank, using Quadrature Mirror Third Order IIR Cauer Filters on a typical cascaded dyadic structure [1]. Multirate sampling reduces computational demands. The bank's output is updated at the same frequency as the last filter pair: 344.53125 Hz, reducing the workload on the system and its dynamic energy requirements. The energy per band is estimated at each output of the filter bank, using an averaging rule (see [1]). Since in cascaded dyadic structures, the QMFs (acronym of Quadrature Mirror Filter) are the same for each dyad, being that, is the sampling that determines the particular band, the low pass and high pass sections of the filter are implemented using the same recursive structure (see Fig. 2), with a transfer function given by

$$H = G\left(\frac{b_{01}+b_{11}z^{-1}}{a_{01}+a_{11}z^{-1}}\right)\left(\frac{b_{02}+b_{12}z^{-1}+b_{22}z^{-1}}{a_{02}+a_{12}z^{-1}+a_{22}z^{-1}}\right) \quad (1)$$

where coefficients (as given in [1]) determine the filter's passband (coefficients are programmable, via the SPI interface).

This recursive structure, though ideal for software, becomes too expensive if directly translated to hardware (each stage of

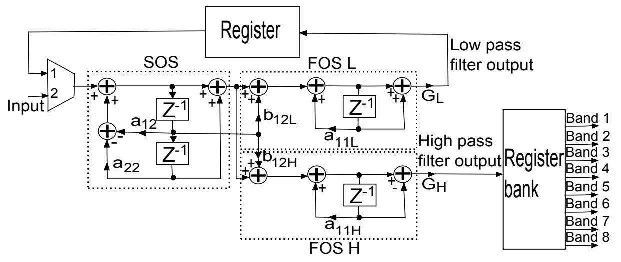

Figure 2. QMF Cauer IIR Filter used for each dyad. Taken from [1]

the filter uses 9 adders and 8 multipliers, each with 24-bit resolution). Thus, only one instance of the IIR filter (being its coefficients equal for each band), is used sequentially for each band after the appropriate sub-sampling. A finite state machine was added in order to control which band is being executed. A 24 register bank is added to provide the previous outputs of each band, to account for the IIR operation. Regarding the operating frequency of the filter bank, every 44.1 kHz a sample must reach the filter's input, and every 128 samples there will be a value at the output.Given its recursive implementation, the filter must operate then at a minimal frequency of 705.6 kHz. Figure 3 shows the results of the Verilog implementation of the filter bank, tested on the FPGA board and compared to data from the BeagleBoard's filter version. Table I shows the filter bank's output's STD error per band for an impulsive input signal, tested against BeagleBoard's implementation output for the same input signal. Worst case is band 5, close to a 4% STD error, acceptable considering the translation from floating to fixed point representation.

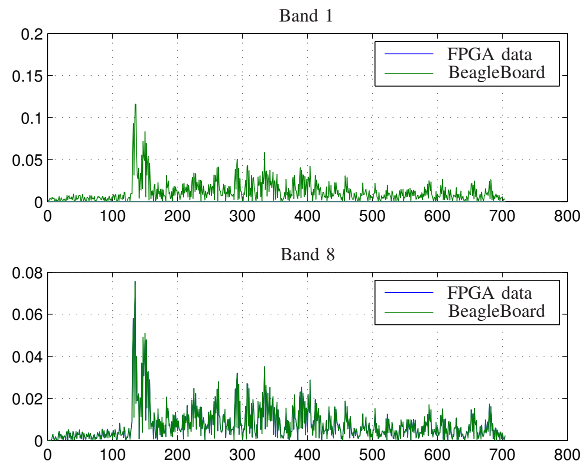

Figure 3. Sample of results from bands 1 and 8 from the Verilog version of the filter bank, compared against data from the BeagleBoard's filter version.

*2) Dimensional reduction:* The dimensional reduction process projects the 8D space out of the filter bank into a 3D space, by means of a linear transformation. This takes care of the *curse of dimensionality* [6] that predicts problems for a classification task using descriptors in a high-dimensional

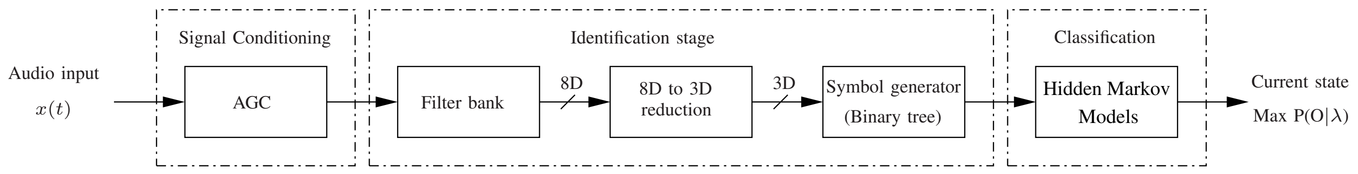

Figure 1. Block diagram for SiRPA. The modularity of the architecture allows for its total or partial accommodation into an ASIC. A complete mathematical description of each processing stage is given in [1]–[4].

Table I
FILTER BANK'S OUTPUT'S STD ERROR PER BAND FOR AN IMPULSIVE INPUT SIGNAL, TESTED AGAINST BEAGLEBOARD'S OUTPUT FOR THE SAME INPUT SIGNAL. WORST CASE IS BAND 5, CLOSE TO A 4% STD ERROR.

| Band | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| STD | 0.0090 | 0.0097 | 0.0166 | 0.0291 | 0.0389 | 0.0116 | 0.0039 | 0.0082 |

space. Though this reduction impacts with the extra area and energy consumption needed for the 8D→3D linear projection, this extra energy that should be compensated with the lesser energy needs of the following stages, proportional to their reduced processing (see [1]–[4]). Figure 4 shows the block diagram of this transformation, basically a mean centered matrix-vector product $\mathbf{y} = \mathbf{W}\mathbf{x}$.

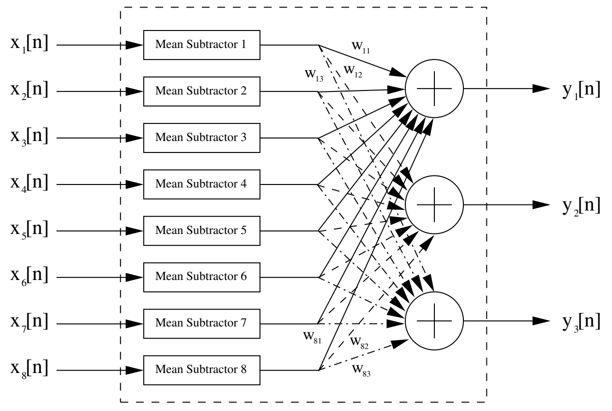

Figure 4. Block diagram for the dimensional reduction stage.

Direct implementation of this stage requires at least 32 adders and 24 multipliers if realized in combinational form. A sequential implementation was thus used in our design, with only a subtracter, an adder and a multiplier, that performed the operation recursively, with a FSM providing recursion control and the appropriate coefficients for each projection (coefficients are again programmable, via an SPI interface). This way, a new value is available every 34 clock cycles, the time required by system to map the 8D input vector to an 3D output. Since the filter has a value every 2.9ms, and the reduction stage takes only 45$\mu$s to have a value ready at a clock frequency of 749 kHz, there is not a significant impact in latency using the aforementioned sequential approach. On the other side, the use of just two adders instead of 32 adders, and two multipliers instead of 24 multipliers, entail a significant reduction in area and power. Figure 5 compares results between the Verilog described 8D to 3D reduction stage against the BeagleBoard's results. STD error is 0.0344, 0.0084 and 0.0113 for each dimension, almost negligible, considering the use of fixed point resolution.

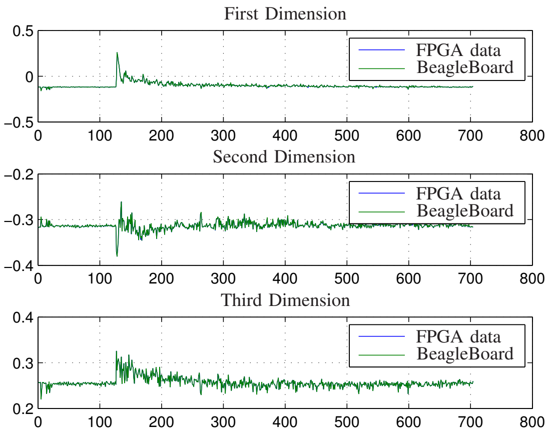

Figure 5. Output results of the HDL described 8D to 3D reduction stage compared with data from the BeagleBoard's implementation. Error is negligible, considering that fixed point resolution is used.

*3) Symbol generator or tree classifier stage:* The tree classifier is a combinational binary tree, also called symbol generator, used to find the closest centroid to a 3D input pattern (see [1]–[4]); L1 or Manhattan distance is used. This way, discrete symbols are generated, associated with the continuous paths in the input vector 3D space that describe the acoustic signal. This structure replaces the original $kd$-tree search algorithm described in [1], [2] and used in the BeagleBoard, unpractical in hardware due to its strongly recursive nature. Centroids are also programmable via an SPI interface.

## C. Classification stage

The final classification module of SiRPA (see Fig. 6 for a block diagram) is based on three HMM machines. A Hidden Markov Model $\lambda = \langle \mathbf{A}, \mathbf{B}, \boldsymbol{\pi} \rangle$ is characterized by the transition matrix $\mathbf{A}$, the symbol emission matrix $\mathbf{B}$, and the start probability vector $\boldsymbol{\pi}$ [7].

A model is used for each of the classes to be recognized. Details on the training and implementation of the HMM algorithms used are given in [1]–[3]. The forward algorithm is used to compute the logarithm of the probability $P(O|\lambda)$ of the

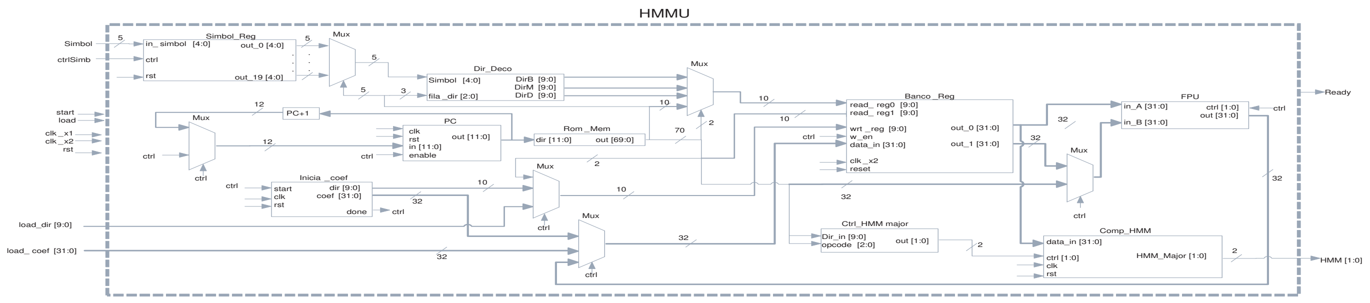

Figure 7. RTL diagram of the classification stage. Floating point unit is taken from [8]. Control is microcoded.

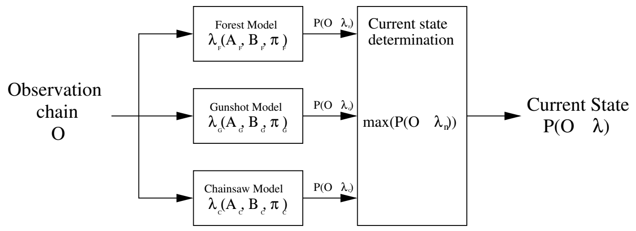
Observation chain O
Forest Model
$\lambda_F(A_F, B_F, \pi_F)$
Gunshot Model
$\lambda_G(A_G, B_G, \pi_G)$
Chainsaw Model
$\lambda_C(A_C, B_C, \pi_C)$
P(O|λ)
P(O|λ)
P(O|λ)
Current state determination
max(P(O|λ_n))
Current State
P(O|λ)

Figure 6. Block diagram for the classification module, using a HMM for each class to be recognized.

observation chain $O$, given the model $\lambda$. Since this stage includes probability and logarithmical sequential computations, a microcoded FSM is implemented to control execution, and a VHDL open source floating point unit is used [8]. The main code is nonetheless implemented on Verilog and is still under test as of the writing of this paper (see Fig. 7 for a RTL diagram of the unit).

## III. Resource consumption

Table II, shows total FPGA resources needed for the ID stage of SiRPA (results from the classification unit are not available yet). Xilinx Xpower Analyzer tool estimated 44 mW of total power consumption at 750 kHz (1 mW of dynamic power and 43 mW due to the FPGA's own static consumption), still very high in terms of energy needs.

Table II
Device Utilization Summary for the identification stage, taken from Xilinx's ISE post-synthesis report.

| Hardware resources | Used |
|---|---|
| Number of Slice Registers | 8592 out of 126800 |
| Number of Slice LUTs | 13910 out of 63400 |
| Number of fully used LUT-FF pairs | 2928 out of 18360 |
| Number of DSP48E1s | 12 out of 240 |

Table III, shows total area and power consumption estimated for ID stage of SiRPA, ported to a low-power standard cell library, for a 130nm CMOS commercial process. Results are promising in terms of energy needs: only 139$\mu$W of dynamic power consumption.

## IV. Conclusions

An acoustic pattern recognition system (SiRPA) has been partially implemented on a HDL, with only the classification stage being still under verification. Partial results of the identification stage, verified against a BeagleBoard's implementation of the system have been shown. These results attest that achieving similar recognition rates as the embedded system's implementation should be feasible on an ASIC. Post-place and route power and area estimation of the tested ID stage code, ported to a standard cells library for a 130nm CMOS commercial process, show a 7.19 times dynamic power improvement over the FPGA implementation, and decidedly several orders of magnitude less than the power required by a BeagleBoard-xM platform.

Table III
Power and area estimation (reported from Synopsys' post-place and route results on a low-power standard cell library, 130nm CMOS process, at 750 kHz; IOs/Pad cells are included.

| Parameter | Post-place and route data |
|---|---|
| Cell Leakage Power | 851.3693 nW |
| Total Dynamic Power | 139.8817 $\mu$W |
| Total cell area | 634,754 $\mu$m$^2$ |

## References

[1] E. Salas-Chaverri and P. Alvarado-Moya, "Implementación de un banco de filtros digitales multitasa para la estimación energética espectral en una aplicación de protección ambiental," in *XXX IEEE Convención de Centroamérica y Panamá*, November 2010.

[2] M. Sequeira and P. Alvarado-Moya, "Módulo de reducción de dimensiones espectrales en un sistema empotrado nodal de una red inalámbrica de sensores," in *Proceedings of the Embedded Technology Conference 2011*, San José, Costa Rica, 2011.

[3] J. Cárdenas-Reyes, "Training strategies for HMM in the acoustic pattern recognition," Master's thesis, Escuela de Ingeniería en Computación, Instituto Tecnológico de Costa Rica, May 2012.

[4] E. Salas Chaverri and P. Alvarado-Moya, "Implementation of an automatic gain control for audio signals in an application for environmental protection," in *Proceedings of the Conference on Technologies for Sustainable Development TSD2011*, Cartago, Costa Rica, 2011.

[5] A. Chacon-Rodriguez, P. Julián, L. Castro, P. Alvarado, and N. Hernández, "Evaluation of gunshot detection algorithms," *Circuits and Systems Part I: Regular Papers, IEEE Transactions on*, vol. 58, no. 2, February 2011.

[6] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[7] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, vol. 77(2), February 1989, pp. 257–286.

[8] R. Usselmann, "Open floating point unit," 2000. [Online]. Available: http://opencores.org/project,fpuvhdl