RISC-V based sound classifier intended for acoustic surveillance in protected natural environments

Carlos Salazar-García*, Reinaldo Castro-González*, Alfonso Chacón-Rodríguez*
*Escuela de Ingeniería Electrónica, Instituto Tecnológico de Costa Rica
URL: http://www.ie.tec.ac.cr, Email: {csalazar, alchacon}@tec.ac.cr

***Abstract*—This paper presents results on FPGA of a RISC-V based Application Specific Processor (ASP), used as the main classification unit for acoustic pattern recognition of firearms and chainsaws in environmentally protected areas. The classifier is based on the Hidden Markov Models (HMM) technique, giving a probabilistic estimation of the current state of an acoustic environment, thereby identifying particular sounds such as gunshots or the hum from a chainsaw engine. The ASP was modeled in Verilog, and uses Berkeley's RISC-V's open Instruction Set Architecture (ISA). The code is written in C and ported to the ASP using RISC-V's toolchain. Verified results on a commercial FPGA, and preliminary synthesis results in a commercial CMOS 130nm process, point at the feasibility of a low power ASIC implementation, integrated into a wireless network of similar nodes.**

***Index Terms*—Acoustic signals recognition, ASP, ASIC, environmental protection, FPGA, HMM, low-power CMOS, RISC-V.**

I. Introduction

An electronic system called SiRPA (Spanish acronym for "Recognition System of Acoustic Patterns") has been devised for the detection and classification of acoustic patterns of gunshots and chainsaws in protected forests (see [1]–[3]). The system should be integrated on a node connected to a Wireless Sensor Network (WSN) in order to cover a wide geographic area such as a forest; considering the high likelihood of not having access to external power supplies, a restriction has been established for the whole node to run on a primary battery such as a 3LR12, which constrains energy to no more than 0.5mAh on average, for a year of uninterrupted service (see [3] for details on the sizing of the node's energy requirements). As a typical pattern recognition systems, SiRPA is composed of an identification stage (ID-stage) and a classification processor (the ASP). Figure 1, shows the block diagram of the system.

There have been many approaches in the past at the problem of detecting guns and chainsaws (see [1], [3], [5] for an evaluation of some of them, and their comparison with this work's approach). More recently, there have been recent implementations of WSNs for environmental protection, such as [6], intended for the detection of tree logging, wildfires and unauthorized access. In the latter work however, power needs are not mentioned, nor details are given about the algorithms used, claiming that their major contribution is in the architecture of the network topology used. Regarding HMMs, several authors have recently integrated them in hardware as an efficient tool for either sound or speech recognition, such as [7], [8], implementing the forward algorithm with a mix of hardware and software running on FPGAs. However, in these two cases authors have used vendor specific IPs for their solutions, which prevent them from being ported directly to a low power ASIC.

This paper shows the implementation details of SiRPA's sound classification processor. The ID-stage details are given in [2]. The acoustic pattern recognition proposed is based on the HMM technique. Training is performed offline and the classification is done by the forward algorithm (see [1], [2], [4] for details on all the training and classification strategies used).

Now, due to the highly iterative nature of the forward algorithm, the low power requirements of the intended application, its floating point processing needs, and the convenience of having re-programmable Markov chains depending on the nature of the sounds and the acoustic environment, an ASP with an open ISA seemed a balanced approach between a specific hardware based solution and the use of a commercial DSP chip. With an ASP is thus possible to reduce the hardware area and consequently the power consumption of the system, without greatly compromising the data processing performance [9]. If changes on the application are needed, they can be easily ported without the need of a re-spin (as in the case of an ASIC) or a full reprogramming (in case of a FPGA implementing a hardware coded algorithm).

This paper is structured as follows: Section II gives a complete description of the chosen processor architecture. Section III details the programming of the processor per se, including the training of the models loaded into it. Section IV gives the results obtained with the ASP running along the ID-stage on a Digilent Nexys-4 board, with recorded data from a real setting used as test signals. Section V shows the conclusions.

II. Implementation of the RISC-V based ASP

Before implementing a microprocessor one needs choosing an ISA that efficiently fits the computational problem to be solved. The underlying microarchitecture is a result both of the ISA and the processing efficiency expected. An open RISC architecture seemed an attractive path from the beginning as this usually means no licensing restrictions and the possibility of exchanging results and experiences with other designers.

978-1-5090-5859-4/17/$31.00 ©2017 IEEE

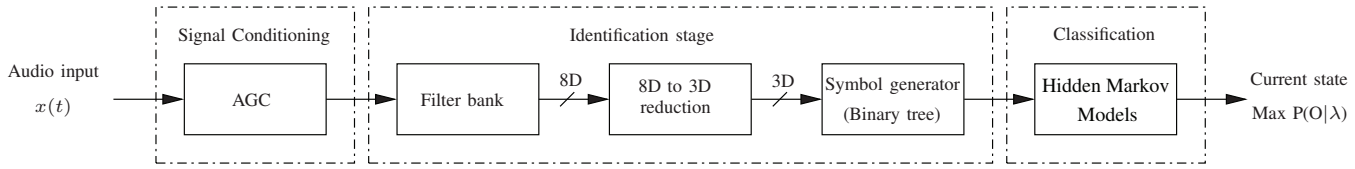

Figure 1. Functional block diagram of SiRPA. The architecture allows for its total or partial accommodation into an ASIC. A complete mathematical description of each processing stage is given in [1], [4]. A complete implementation description of the ID-stage is given in [2].

Thus, the first steps of this work naturally required a selection of the open processing platform to be used.

A. ISA Selection for the ASP

Two open RISC ISA alternatives were analyzed as potential platforms: Open RISC and RISC-V. Both ISAs had all the required features for the development of the intended ASP, but RISC-V was finally selected due to the extensive support on tools and documentation offered by the University of California at Berkeley, and the growing community of users, both academic and commercial, behind this initiative.

B. Assembler instructions implemented

RISC-V provides a full set of typical instructions within its ISA; however, many instructions might not be necessary for a particular case of an ASP. For this reason, an extensive profiling of the intended algorithm becomes mandatory in order to figure out the minimum set of instructions needed, and the resulting hardware to be implemented.

In this case, the RISC-V standard base RV32I provided both with the flexibility and compactness required. The list of implemented instructions can be found in [10].

C. Microarchitecture design

A multicycle pipelined structure was chosen, because it provides acceptable performance at a power and area cost suitable for the application intended. Yet, provisions were made for the transformation of the microarchitecture into an unicycle pipelined one, in case that a processing speed improvement is required.

The ASP was described in synthesizable Verilog, avoiding vendor specific directives that could prevent its portability to an ASIC or another FPGA platform. Pipelining was used to reduce critical routes and glitching, at the expense of a few extra clocks in certain operations, but allowing for a full 100MHz clock speed in the tested FPGA. Figure 2 shows a block diagram of the ASP core. The ASP bootstraps from a simple FSM and loads via an SPI interface the compiled C code to be run from a 2kB RAM program memory block. A 4kB block of RAM is available for data and stack storage. Data describing the HMM chains is also stored there, loaded via SPI as well. In order to fulfill timing requirements, an open core FPU is added to the ASP (see [11] for details). The logarithmic functions needed are implemented via C code.

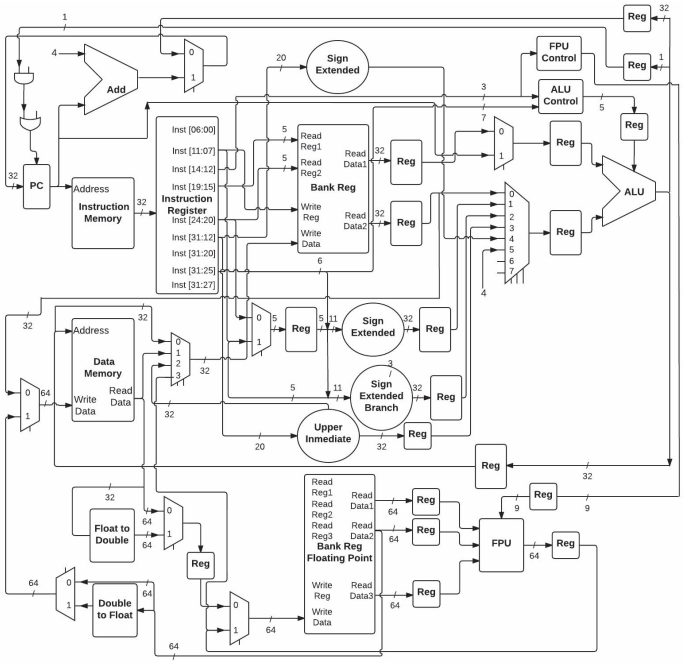

Figure 2. RISC-V based ASP core plus FPU. A multicycle pipeline structure with separate data and program memory is used. Access to the FPU is also registered.

D. ASP intefacing with the ID stage

The ASP connect through several parallel interfaces (PPIs) with the previous ID-stage, which provides the symbols for their classification (see [2] for details). A binary tree at the ID-stage generates an alphabet of 32 symbols, which represent the input observations of the forward algorithm. Symbols are stored into a shift register, and after a certain number of observations (that depend on the length of the trained Markov chain), a start signal is generated for the forward algorithm.

The output PPI simply stores the probabilistic results of the forward algorithm running on the three Markov models available in the ASP. Some extra signals give the system's state and flag when the system finishes an estimation.

III. Implementation of the classification algorithm to run on the ASP

The forward algorithm is used to compute the logarithm of the probability $P(O|\lambda)$ of the observation chain O, given a Hidden Markov model $\lambda = \langle \mathbf{A}, \mathbf{B}, \boldsymbol{\pi} \rangle$, characterized by the transition matrix $\mathbf{A}$, the symbol emission matrix $\mathbf{B}$, and the start probability vector $\boldsymbol{\pi}$. The standard algorithm has three

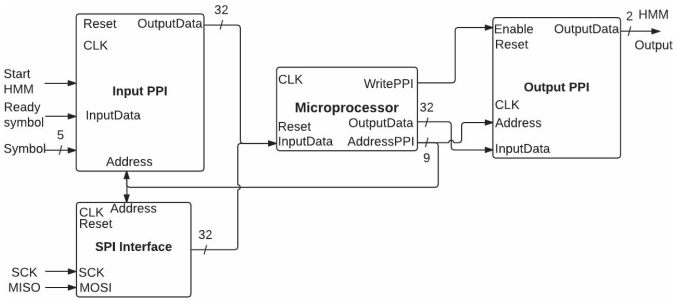

Figure 3. Blocks Diagram of the ASP implemented, interconnected with the ID-stage developed in [2].

steps, with an scaling after the first two steps that modifies the last one (this algorithm is extensively described in the literature; see [1], [4] for the case at hand). First, some parameters are initialized, using the following rule:

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N \tag{1}$$

where N is the number of states in the model.

Second, an induction rule is given, using

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \quad \begin{matrix} 1 \leq t \leq T-1 \\ 1 \leq j \leq N \end{matrix} \tag{2}$$

with

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)} \tag{3}$$

and

$$\alpha_t^{'}(i) = c_t \alpha_t(i), \quad 1 \leq i \leq N \tag{4}$$

where T is the number of observations in the sequence.

And third, a termination step computes the desired probability as

$$\log(P(O|\lambda)) = -\sum_{t=1}^{T} \log(c_t) \tag{5}$$

The results from all three HMMs are compared, and the one closer to zero is chosen as the winner class. Since the FPU had no support for non linear arithmetic, and the ASP has no access to C's math.h library (as no OS is provided), an extra logarithmic function was thus coded in C and loaded into the ASP. All the constants for each one of the λ models, are loaded after the ASP boots. After initializing the chains, the system starts receiving data from the ID-stage. Models can be re-loaded after booting via the same SPI interface.

IV. Results

The C coded implementation described in [3] provided with the reference model for functional verification of the ASP. Both the ID stage and the ASP were instantiated on a Artix 7 based Digilent's Nexys-4 platform. Recorded data from a tropical rainforest setting was used as stimuli, using the testing circuit already described in [2]. Classification output data was sent back to a PC via UART for analysis against the reference model. Three acoustic situations were tested: firearms being shot at different distances, chainsaws being started and accelerated at different distances, and several recordings of the forest without man-made acoustic disturbances (see [1], [5] for details on the data set used). Table I gives a summary of the algorithm's confusion matrices, showing recognition rates for the three models tested. Here, the sensitivity gives the probability that a pattern of a given class will be correctly classified. The PPV indicates the probability that a pattern assigned by the classifier to a given class is correctly classified (see [12] for details on these definitions). Table I shows positive rates up to 90.33% at identifying chainsaws and 90.43% for gunshots, depending on the λ model used. Figure 4 plots the percentage error between results from the reference model and data from the system on the FPGA (error arises mainly from the ID-Stage, running on fixed point arithmetic). Table II, gives the percentage error's average (well under 1%) and its standard deviation, between results from the reference model against results from the FPGA.

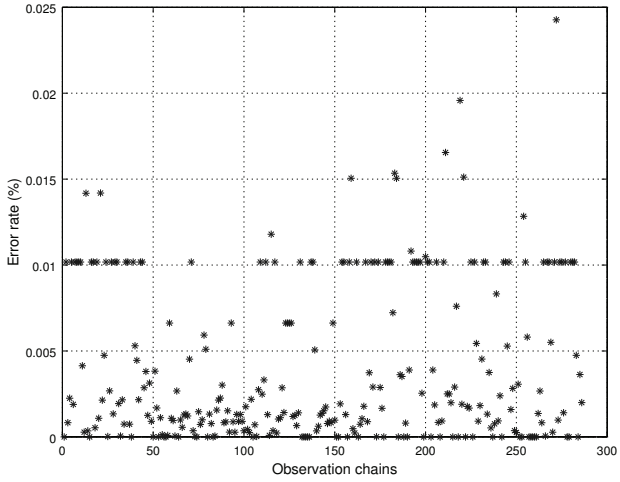

Figure 4. Error (as a percentage) between reference model data and results from SiRPA running on a FPGA, for each observation chain tested.

Table II
Percentage error average and standard deviation between results from the reference model against results from the system running on the ASP.

Error	Forest	Chainsaw	Gunshot
AVG	0.01	0.015	5.0271×10^{-4}
STD	2.8896×10^{-5}	1.4336×10^{-5}	1.0271×10^{-5}

The system takes on average 23.01ms for a complete classification (average estimated after more than 20000 tests), well under the original 58.14ms restriction imposed in [3], [10].

Table III gives a summary of the Artix 7 resources usage. Xilinx's Xpower Analyzer reports an average 47mW dynamic power consumption at a 100MHz clock. This makes the FPGA implementation unsuitable for the system's final deployment. Verified code of the whole system (the proposed ASP with

Table I
RECOGNITION RATE AGAINST HMM' $\lambda = (\mathbf{A}, \mathbf{B}, \underline{\boldsymbol{\pi}})$ MODELS OF A GUNSHOT, A CHAINSAW AND AN UNDISTURBED RAINFOREST. SYSTEM'S EXECUTION TIMES FOR A VALID ESTIMATION ARE ALSO GIVEN, WITH AND WITHOUT FPU (ASP CLOCK @100MHZ).

Model Features			Sensitivity (%)			PPV (%)			Execution time (ms)	
Sub-sampling	Chain Length (T)	State (N)	Forest	Chainsaw	Gunshot	Forest	Chainsaw	Gunshot	With FPU	Without FPU
1	20	10	91.37	90.33	85.43	82.33	86	100	80.43	23.01
1	10	10	81.32	73.68	90.43	65.89	63.44	100	39.80	10.12
5	20	5	92.34	90.23	0.00	91.67	0	78.23	18.16	0.92
1	5	3	57.34	54.78	88.52	62.78	97.01	53.29	4.79	0.78
10	25	15	96.37	83.34	0	90.23	88.23	0	123.34	45.97

FPU included, plus ID-Stage depicted in [2]) was thus ported to a 130nm CMOS commercial process, using a commercial standard cell library and hard coded macros for RAM memories. Preliminary synthesis results from Design Compiler show a total power consumption under 2mW at a 100MHz clock, with a switching activity factor of 0.3. Though this power figure is still well over the bounds established in [3], it nonetheless points at the feasibility of a low energy ASIC implementation if power gating strategies are used, considering that the ASP need only work if an event is previously detected. This might be done by adding a simpler detection unit capable of awakening SiRPA, as proposed in [5]. This might also increase the system's data recognition ratio.

Table III
RESOURCES USED BY THE ASP WITH INCORPORATED FPU, ON AN ARTIX 7 FPGA. XILINX ISE DESIGN 14.04'S POST PLACE & ROUTE REPORT.

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers as FFs	6174	126800	4%
Number of Slice LUTs	5505	63400	8%
Number of occupied Slices	2322	15850	14%
Number with an unused Flip Flop	2440	8137	29%
Number with an unused LUT	2632	8137	32%
Number of fully used LUT-FF pairs	3065	8137	29%
Number of RAMB36E1/FIFO36E1s	24	135	17%
Number of DSP48E1s	23	240	9%

V. CONCLUSIONS

Correct FPGA implementation of a HMM based classification stage, running the forward algorithm on A RISC-V based ASP, has been shown. The ASP, integrated with previous signal conditioning and identification stages, works as an efficient acoustic pattern recognition system for firearms and chainsaws activity. Preliminary synthesis results on a 130nm commercial CMOS process indicate that system may be ported to a low power ASIC, as part of a WSN intended for environmental protection in wildlife reserves, but still providing with the flexibility of algorithmic reprogramming and trimming through the RISC-V development toolchain.

ACKNOWLEDGMENTS

The authors would like to thank Synopsys and Xilinx for their support through their university programs. The authors would also like to thank the MOSIS Educational and Research Program for its support with the libraries for IC prototyping.

REFERENCES

[1] C. Gomez-Viquez, L. Li-Huang, O. Villalta-Gutierrez, A. Chacón-Rodriguez, and P. Alvarado-Moya, "An Embedded Test System for Acoustic Pattern Recognition Intended for Environmental Monitoring and Protection in Tropical rain forest reserves," in *Proceedings of the ETC2012 Third Embedded Technology Conference*, San Jose, Costa Rica, January 2012.

[2] C. Salazar-García, L. Alfaro-Hidalgo, M. Carvajal-Delgado, J. Montero-Aragón, R. Castro-Gonzalez, J. A. Rodríguez, A. Chacón-Rodríguez, and P. Alvarado-Moya, "Digital integrated circuit implementation of an identification stage for the detection of illegal hunting and logging," in *Circuits Systems (LASCAS), 2015 IEEE 6th Latin American Symposium on*, Feb 2015.

[3] J. Montero-Aragon, C. Salazar-Garcia, R. Castro-Gonzales, J. Cardenas-Reyes, A. Chacon-Rodriguez, and P. Alvarado-Moya, "An embedded system for the detection of illegal hunting and logging," in *2016 IEEE XXXVI Convencion de Centroamerica y Panama (Concapan)*, November 2016.

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

[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] R. Mina and Y. Ziade, "Towards an optimal solution for environmental protection using wireless sensor networks," in *2016 Third International Conference on Electrical, Electronics, Computer Engineering and their Applications (EECEA)*, April 2016, pp. 120–125.

[7] K. Sujuan, Y. Hou, Z. Huang, and H. Li, "A HMM Speech Recognition System Based on FPGA," in *Congress on Image and Signal Processing*, China, Mayo 2008.

[8] I. Abbas, "Isolated Uttered Words Recognition Based on GMM/HMM Algorithms Using SoPC/Nios II Processor Build on Altera Cyclone II FPGA Chip," in *National Conference for Engineering Sciences*, Iraq, Noviembre 2012.

[9] A. C. Cheng and G. S. Tyson, "An energy efficient instruction set synthesis framework for low power embedded system designs," *IEEE Transactions on Computers*, vol. 54, no. 6, pp. 698–713, Jun 2005.

[10] C. Salazar-García, "Implementación de un microprocesador de aplicación específica para la ejecución del algoritmo de modelos ocultos de Markov en el reconocimiento de patrones acústicos," Tesis de Maestría, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Diciembre 2015.

[11] A. Cervantes, J. Quiros, D. Rodriguez-Valverde, C. Salazar-Garcia, and A. Chacon-Rodriguez, "Implementation of an open core IEEE 754-based FPU with non-linear arithmetic support," in *2016 IEEE XXXVI Convencion de Centroamerica y Panama (Concapan)*, November 2016.

[12] Q. Wu, F. A. Merchant, and K. R. Castleman, *Microscope image processing*. Amsterdam, Boston, Paris: Elsevier/Academic Press, 2008.