Inner-Hair Cells Parameterized-Hardware Implementation for Personalized Auditory Nerve Stimulation

Miguel A. Sacristán-Martínez[1], José M. Ferrández-Vicente[1], Vicente Garcerán-Hernández[1], Victoria Rodellar-Biarge[2], and Pedro Gómez-Vilda[2]

[1] Universidad Politécnica de Cartagena, Dpto. Electrónica, Tecnología de Computadoras y Proyectos, Cartagena, 30202, Spain
miguel.sacristan@upct.es, jm.ferrandez@upct.es, vicente.garceran@upct.es

[2] Universidad Politécnica de Madrid, Dpto. Arquitectura y Tecnología de Sistemas Informáticos, Boadilla, 28660, Spain
victoria@pino.datsi.fi.upm.es, pedro@datsi.fi.upm.es

Abstract. In this paper the hardware implementation of an inner hair cell model is presented. Main features of the design are the use of Meddis' transduction structure and the methodology for Design with Reusability. Which allows future migration to new hardware and design refinements for speech processing and custom-made hearing aids.

1 Introduction

Many people suffering deafness would find relief to their disabilities by receiving cochlear implants. They deliver electric stimuli proportional to the sound or speech captured from the environment to the auditory nerve, [1]. Artificial bio-inspired systems behave adequately in general, but demand very high computational costs, which render them inadequate for many real-time applications.

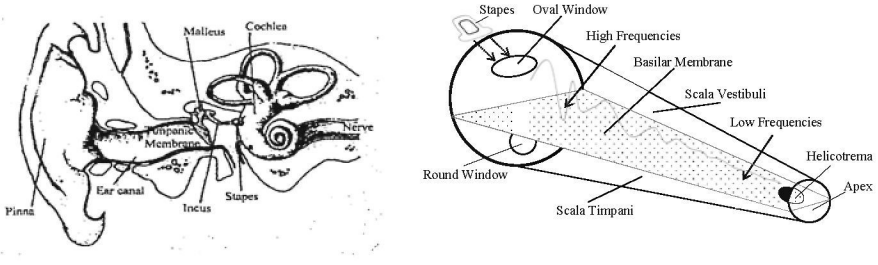
In addition, Speech processing has advanced tremendously in recent years, however, still fall short for performance and noise tolerance level when compared with biological recognition systems [2]. For this reason, biologically inspired algorithms are not only important in the design field of hearing aids, also to improve the interfaces between humans and machines.

The work herein described is substantially focussed to the implementation of a first prototype in a FPGA showing the behaviour of the inner auditory system, with the function of converting the movements of the basilar membrane in trains of pulses to be carried to the brain for its processing. The proposed system may serve as a front-end processor in bio-inspired speech processing and recognition applications. The utilization of Design with Reusability methodology

allows not only accelerating the cycle of prototyping, but also adapting the design to the specific needs of a particular problem, for example, varying the amount and frequency of transduction channels as a function of the patient's needs or recognition system.

1.1 Auditory System Description

Auditory periphery can be described as the concatenation of three stages: outer, middle and inner ear, as shown in Fig. 1a. Sound is the result of the vibrations of the air surrounding the ear. They reach the outer ear, consisting of the pinea and the ear canal, which adapts the sound pressure wave and is partly responsible for the location of the source. Vibrations reach the middle ear through the auditory canal and cause displacements in the tympanic membrane where the middle ear begins. These movements are transmitted through a chain of three small bones (malleus, incus and stapes) and reach to the cochlea which is the main part of the inner ear. Cochlea can be considered as divided into two scales by the basilar membrane, the tympanic and the vestibular scales, fig. 1b. The organ of Corti, located in the basilar membrane, transduces the mechanical vibrations into electrical pulses transmitted to the brain by the auditive nerve [3]. High frequency waves stimulate hair cells near the organ of Corti base and low frequency ones act close to the apex.



(a) Auditory Periphery (b) Cochlea

Fig. 1. Structure of the Auditory Periphery, (a) extracted from [4]

1.2 Bio-inspired Models

Figure 2 shows the typical block structure based on the auditory periphery. Outer and Middle ears capture the sound waves and perform an initial filtering of the signal in order to increase the pressure of the sound in the region from 2 to 7 kHz. The cochlea is typically modelled by two stages. The first one models the mechanical selectivity taken place in the cochlear structure. The vibration velocity of several points along the basilar membrane may be simulated by a set of filters like the dual resonance nonlinear filter (DRNL) proposed in [5,6]. The second one simulates the behaviour of the inner hair cell transducing the vibrations into electrical signals.

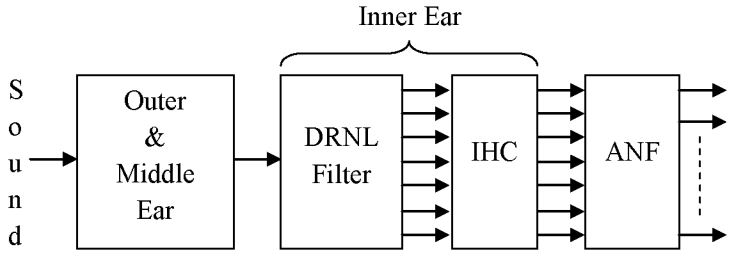


Fig. 2. Bio-inspired block diagram of the processor

2 Inner Hair Cells (IHC)

The IHC perceives basilar membrane movements and stimulates the afferent neural fibbers (ANF) by neurotransmitters libration in the synaptic cleft. Each IHC rectifies and compresses signals coming from the basilar membrane reducing heavily the frequencies above 1 kHz and the occurrence of the non-instantaneous compressions in the synaptic cleft.The behaviour of the IHC can be modelled as a function of the neurotransmitters flux through three reservoirs, fig. 3.

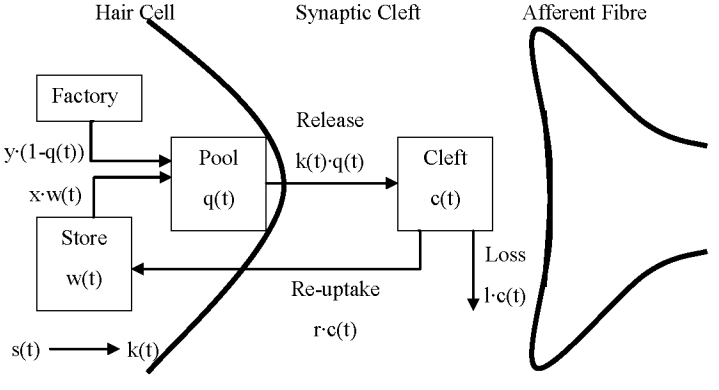


Fig. 3. IHC function characterized by the Meddis model B

The first one $q(t)$ represents the amount of neurotransmitters ready to be released as a function of the membrane displacement $s(t)$. The second reservoir represents the amount of neurotransmitters released in the synaptic cleft $c(t)$, which determines the impulse rate in the postsynaptic afferent fibbers. The third reservoir $w(t)$, represents the amount of neurotransmitters recovered from the synaptic cleft contributing to the net amount of free neurotransmitters.

The first step is to calculate the membrane permeability $k(t)$ as function of the acoustic stimulus $s(t)$. The following relations hold:

$$k(t) = \begin{cases} [g(s(t)dt + A)]/[s(t) + A + B]; & s(t) + A > 0 \\ 0; & s(t) + A \leq 0 \end{cases} \quad (1)$$

where A and B are positive constants and $B >> A$. It must be noticed that when $s(t) = 0$ some spontaneous activity is allowed.

The total amount of free neurotransmitters will depend on the presence of the generated, released and re-processed neurotransmitters within a given interval dt. The product between the free transmitters available $q(t)$ and the membrane permeability $k(t)$, gives the amount of transmitters released into the cleft. On the other hand, the cell has the capability for generating transmitters at the rate of $y\ dt[M - q(t)]$, y represents the replenishing rate and M, the upper limit of free transmitters, for $M > q(t)$; otherwise this value will be zero. The recovered transmitters amount is the input to a re-processing store $w(t)$ and it will contribute in a proportion x to the free transmitters total amount $q(t)$.

$$q(t + 1) = q(t) + y[(M - q(t)]dt - q(t)k(t)dt + xw(t)dt \quad (2)$$

The re-processing reservoir stores the difference between the returned and re-processed transmitters within dt:

$$w(t + 1) = w(t) + rc(t)dt - xw(t)dt \quad (3)$$

The total amount of transmitters presents into the cleft is denoted by $c(t)$. Part of these transmitters will return into the hair cell in a proportion of $r\ c(t)$ according to a rate r and part of it will be lost: $l\ r(t)$, l representing the loss rate. And finally the amount of neurotransmitters in the cleft will depend on the released, returned and lost amounts:

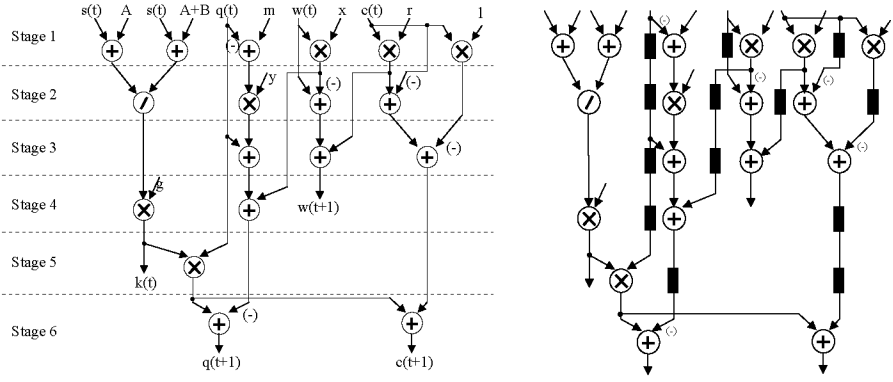
$$c(t + 1) = c(t) + q(t)k(t)dt - lc(t)dt - rc(t)dt \quad (4)$$

The parameters, q, A, B, r, l, x, y and M are constants.

3 Implementation

Dataflow shown in figure 4a is the transformation of the previous equations into an implementable structure that improves the solution proposed in [7]. The main difference corresponds with the first left branch where the adders have been reordered to provide the operands to the divider as soon as possible. This improvement minimizes the delay to start the division operation.

Parallel hardware implementation, shown in fig. 4b, includes several buffers, represented by the black boxes. These delaying elements have been included in order to adjust the latency of all paths. So, the structure has been transformed into a pipeline.



(a) Data-flow representation. (b) Hardware implementation.

Fig. 4. Parallel implementation

Also a serial implementation has been explored to evaluate its performance in terms of area, which could be critical in the design of low-power nerve stimulators with a big number of channels. This implementation consists in a single arithmetical block for each operation involved in the algorithm and the controller for sequencing the operations execution, fig. 5.

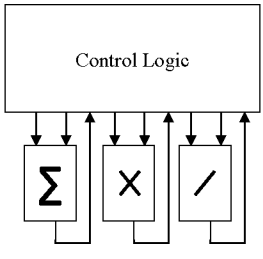


Fig. 5. Serial implementation

As the arithmetic blocks are internally pipelined, the controller sequentially programs same type operations that haven't dependencies among them. Operations scheduling for the serial implementation is shown in table 1. The complete algorithm is computed in six stages. By comparing this information with the parallel implementation shown in fig. 4b we can notice that the number of stages is the same in both cases, but the parallel implementation is fully pipelined meanwhile the serial one needs to completely cover one stage before starting the computation of the following one. It means that the results of the first stage needs 2 more clock tics in the serial implementation. For stage 2 and 3 the results availability is the same in both implementations because the division operation is

the bottleneck. Since there is only one addition and one multiplication in stages 4 and 5, the delay is the same. And finally stage 6 needs one clock cycle more in the serial implementation.

Table 1. Operations scheduling for the serial hardware

	ADDER	MULT.	DIVIDER
Stage 1	**[1]** $s(t) + A$ **[2]** $s(t) + (A + B)$ **[3]** $q(t) + M$	**[4]** $x \cdot w(t)$ **[5]** $r \cdot c(t)$ **[6]** $l \cdot c(t)$	
Stage 2	**[7]** $w(t) - [4]$ **[8]** $c(t) - [5]$	**[9]** $y \cdot [3]$	**[10]** [1]/[2]
Stage 3	**[11]** $[9] + q(t)$ **[12]** $[7] + [5]$ **[13]** $[8] - [6]$		
Stage 4	**[14]** $[11] + [4]$	**[15]** $g \cdot [10]$	
Stage 5		**[16]** $[15] \cdot q(t)$	
Stage 6	**[17]** $[16] + [14]$ **[18]** $[16] + [13]$		

4 Results

The design was implemented using IEEE 754 double precision arithmetical blocks available at the CoreGen tool of $Xilinx^{TM}$. These blocks can be configured to perform addition, multiplication and division operations. Subtraction will be calculated by inverting the sign bit of the subtrahend. Overflow control is accomplished inside the blocks according to the mentioned IEEE standard.

Each block generated by the CoreGen tool is encapsulated within a VHDL *entity*, which will contain only the input and output ports and control signals. In this way, just changing the description of the components we can easily evaluate the performance of different particular structures and implementations of the basic components.

The structures were implemented over VIRTEX-5 devices using ISE 10.1 SP3 synthesis software and simulated for the 17 channels as proposed in [8]. The design was tested under the same conditions than [8], the simulation results were fully coincident which validates the functional behaviour of the structures.

Related with the hardware implementation, all the arithmetical blocks were generated by the CoreGen utility in two different ways: **Full logic**, using only LUTs; and **DSP48E**, exploding the internal arithmetic structures inside the

Virtex5 family. As DSP48E slices are mainly intended for multiplication and inner product, multipliers benefits from them doubling their working frequency and reducing to one fourth the amount of needed LUTs. Adders have 10% of improvement in speed and divider doesn't benefit from these slices.

The two structures have almost the same working frequencies regardless if they use full logic or DSP48E slices as can be seen in table 2. The use of DSP48E slices don't present a significative impact on the working frequency due to the critical path is inside the divider. The differences in the frequencies are due to the routing of the signals, bigger circuits let less free routes. DSP48E slices increase the latency of the arithmetic blocks because the control logic for normalizing the result or checking overflows can not be integrated within the arithmetical logic. The second line in table 2 shows that the working frequency for calculating 17 IHCs is in the range 1.5 to 2 MHz. Both structures are able to work in real time considering a data sampling frequency of 16 kHz for the input sound.

Table 2. Working frequency

	Parallel		Serial	
	Full logic	DSP48E	Full logic	DSP48E
Clock cycle (MHz)	203	211	250	252
17 IHC channels (MHz)	1.46	1.45	2	1.92

Table 3 shows the synthesis results for both structures using either LUTs or DSP48E blocks. The parallel structure needs the same resources to calculate just one IHC or the 17 used in [8] due to its pipeline operation.

Table 3. Hardware allocation

	Parallel		Serial	
	Full logic	DSP48E	Full logic	DSP48E
1 IHC channel	23870 LUTs	$14803 LUTs$ $81 DSP48E$	7037 LUTs	$4973 LUTs$ $16 DSP48E$
17 IHC channels	23870 LUTs	$14803 LUTs$ $81 DSP48E$	119629 LUTs	$84541 LUTs$ $272 DSP48E$

If only LUTs are used, the design fits in the second smallest Virtex5 device, XC5VLX50. But if DSP48E blocks are used, the high number of them require jumping to the second largest Virtex5 although only a 10% of the LUTs will be used. The number of LUTs in the full logic implementation of the parallel structure is smaller than expected because the constant operands cause that the inner structure of the arithmetical can be simplified.

Serial structure demonstrates being the smaller of the two for just one IHC. But with only three IHCs serial structure equals parallel results and it's impracticable for 17 IHCs.

5 Conclusions

The parallel structure can be synthesized on the smallest devices of the Virtex5 family. And its hardware implementation cost is independent of the number of stimulation channels. The results of the serial implementation explodes in physical resources demand as the number of channels increases. But serial implementation has the facility of easily customizing the constants of each ICH involved in the complete model.

DSP48E blocks utilization has no advantage in any case because they can not be harnessed to optimize the critical path. Also the use of DSP48E slices in the parallel structure avoids its internal simplification although there are many arithmetical blocks with constant operands.

The design methodology allows the design modification in order to introduce more or less transduction channels in different basilar membrane sections. Thus the design can be tailored for a specific nerve stimulation problem.

Acknowledgment

This work has been funded by grant TEC2009-14123-C04-03 from Plan Nacional de I+D+i, Ministry of Science and Technology, by grant CCG06-UPM/TIC-0028 from CAM/UPM, and by project HESPERIA (http.//www.proyectohesperia .org) from the Programme CENIT, Centro para el Desarrollo Tecnológico Industrial, Ministry of Industry, Spain.

References

1. Bruce, I.C., White, M.W., Irlicht, L.S., O'Leary, S.J., Dynes, S., Javel, E., Clark, G.M.: A stochastic model of the electrically stimulated auditory nerve: single-pulse response. IEEE Transactions on Biomedical Engineering 46(6), 617–629 (1999)
2. Martínez–Rams, E.A., Garcerán–Hernández, V.: ANF stochastic low rate stimulation. In: Mira, J., Álvarez, J.R. (eds.) IWINAC 2007. LNCS, vol. 4527, pp. 103–112. Springer, Heidelberg (2007)
3. Zigmond, M.J., Bloom, F.E., Landis, S.C., Roberts, J.L., Squire, L.R.: Fundamental Neuroscience. Academic Press, London (1999)
4. Rodellar, V., Gómez, P., Sacristán, M.A., Ferrández, J.M.: An inner ear hair cell parametrizable implementation. In: 42th Midwest Symposium on Circuits and Systems (Las Cruces), vol. 2, pp. 652–655 (1999)
5. Lopez-Poveda, E.A., Meddis, R.: A human nonlinear cochlear filter bank. The Journal of the Acoustical Society of America 110(6), 3107–3118 (2001)
6. Meddis, R., O'Mard, L.P., Lopez-Poveda, E.A.: A computational algorithm for computing nonlinear auditory frequency selectivity. The Journal of the Acoustical Society of America 109(6), 2852–2861 (2001)
7. Ferrández-Vicente, J.M., Sacristán-Martínez, M.A., Rodellar-Biarge, V., Gómez-Vilda, P.: A High Level Synthesis of an Auditory Mechanical to Neural Transduction Circuit. In: Mira, J., Álvarez, J.R. (eds.) IWANN 2003. LNCS, vol. 2686, pp. 678–685. Springer, Heidelberg (2003)
8. Martinez-Rams, E., Garceran-Hernandez, V., Ferrandez-Vicente, J.M.: Low rate stochastic strategy for cochlear implants. Neurocomputing, 936–943 (2009)