Journal of Systems Architecture 76 (2017) 89–101

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Real-Time neural signal decoding on heterogeneous MPSocs based on VLIW ASIPs

Paolo Meloni[a,*], Claudio Rubattu[a], Giuseppe Tuveri[a], Danilo Pani[a], Luigi Raffo[a], Francesca Palumbo[b]

[a] *Dipartimento Ingegneria Elettrica ed Elettronica, Università degli Studi di Cagliari, Cagliari, 09123, Italy*
[b] *PolComIng – Gruppo Ingegneria dell'Informazione, Università degli Studi di Sassari, Sassari, 07100, Italy*

ARTICLE INFO

Article history:
Received 5 February 2016
Revised 31 August 2016
Accepted 15 November 2016
Available online 19 November 2016

Keywords:
Neural signal decoding
MPSoCS
ASIPs
Design space exploration
Low power

ABSTRACT

An important research problem, at the basis of the development of embedded systems for neuroprosthetic applications, is the development of algorithms and platforms able to extract the patient's motion intention by decoding the information encoded in neural signals. At the state of the art, no portable and reliable integrated solutions implementing such a decoding task have been identified. To this aim, in this paper, we investigate the possibility of using the MPSoC paradigm in this application domain. We perform a design space exploration that compares different custom MPSoC embedded architectures, implementing two versions of a on-line neural signal decoding algorithm, respectively targeting decoding of single and multiple acquisition channels. Each considered design points features a different application configuration, with a specific partitioning and mapping of parallel software tasks, executed on customized VLIW ASIP processing cores. Experimental results, obtained by means of FPGA-based prototyping and post-floorplanning power evaluation on a 40nm technology library, assess the performance and hardware-related costs of the considered configurations. The reported power figures demonstrate the usability of the MPSoC paradigm within the processing of bio-electrical signals and show the benefits achievable by the exploitation of the instruction-level parallelism within tasks.

1. Introduction

A key research topic in bioengineering relates with the development of neuro-controlled prosthetic systems. Within such systems, a robotic active prosthesis is controlled to respond to the patient's motion intention, identified by means of an adequate decoding of the information carried by electrophysiological neural signals. Among different approaches, a solution that appears very promising and effective is represented by neuroprostheses that rely on the sensing of Peripheral Nervous System (PNS) signals [1]. In this kind of devices, the signal to be decoded is acquired by means of electrodes implanted in the residual nerves near the amputation region, in order to allow the amputee to exploit the natural pathways of motor control. Thus, prostheses relying on PNS sensing can be more easily perceived by the patient, compared to electromyographic ones.

The identification of motion intention is performed by means of complex decoding algorithms, that analyse neural *spikes*, temporal sequences of *action potentials* that show a typical impulsive waveform, fired by excited motor neurons. The starting point of decoding is usually a *Spike sorting* phase [2], that identifies spikes emitted by different neurons looking at their morphology.

Decoding algorithms are computationally intensive, since sampling frequency required by neural signals [3] are relatively high and usually signal acquisition from multiple channels has to be managed [4]. Moreover, application requires execution on highly portable embedded processing system, wearable or even implantable, with very limited budget in terms of power and energy consumption, and, at the same time, must comply with real-time constraints.

This work proposes a solution that exploits the intrinsic parallelism in the considered application, executing it on a custom Multi-Processor System-on-Chip (MPSoC). In this way, we combine the processing power provided by the MPSoC paradigm with the power/energy efficiency that can be obtained by means of macro-architectural and micro-architectural customization. We consider a state-of-the-art PNS signal decoding algorithm [5], that achieves good performance even in case of low Signal-to-Noise Ratio (SNR)

* Corresponding author.
E-mail addresses: Paolo.Meloni@diee.unica.it, paolo.meloni@diee.unica.it (P. Meloni), Claudio.Rubattu@diee.unica.it (C. Rubattu), Giuseppe.Tuveri@diee.unica.it (G. Tuveri), Danilo.Pani@diee.unica.it (D. Pani), Luigi.Raffo@diee.unica.it (L. Raffo), fpalumbo@uniss.it (F. Palumbo).

http://dx.doi.org/10.1016/j.sysarc.2016.11.005
1383-7621/© 2016 Elsevier B.V. All rights reserved.

conditions, both on animals [5] and humans [6]. Our work considers such algorithm as a target and takes as reference an architectural template and a library of configurable building elements that expose a wide scope of customization knobs. We propose a design space exploration (DSE), comparing multiple hardware/software MPSoC configurations in terms of throughput and hardware costs.

The remainder of this paper is organized as follows. Section 2 presents previous works. Section 3 describes in general the system to be implemented and the adopted design strategy. The target state-of-the-art algorithm [5] is presented in Section 4. Section 5 describes the reference MPSoC architectural template, while Section 6 defines the programming model. Section 7 discusses in detail the DSE results. Section 11 concludes with some final remarks.

2. Related works

As already explained in Section 1, spike sorting analyzes the morphology of spikes in order to identify the firing activity of the different neurons [2]. Neural signal processing has been addressed in several previous works, mostly using FPGAs as target implementation technology [7,8], with the aim of improving flexibility, with respect to ASICs [9,10], and performance, by means of parallelism exploitation, with respect to general-purpose processors.

In [7], a low-power nano-FPGA is used to implement a fully implantable programmable neuroprocessor. The system takes care of data acquisition and reduces the output bit-rate using a compression algorithm that exploits the sparse representation of the neural signals. In this way, it overcomes the bandwidth limitation of wireless telemetry, transmitting only the samples associated to the detected spikes to an external computing platform that implements a cortically-controlled Brain-Machine Interfaces. The system functionality has been assessed using a dataset of raw extracellular signals recorded through microelectrode arrays chronically implanted in the brain of sedated rats.

The feasibility of this approach in terms of power has been investigated on standard CMOS VLSI [11]. In this approach, the computational complexity is shifted at downstream of the implantable device in order to perform the decoding which can be performed on many-core platforms [12] or FPGA-accelerated solutions [13].

Other works have targeted energy-efficient implementation of multi-channel spike sorting [14], dealing in general with signals coming from the Central Nervous System (CNS), focusing on the analysis of requirements, in terms of hardware resources and accuracy, of some processing steps of spike sorting algorithms [15,16]. Such implementations exploit massive parallelization and cannot comply with the low-power requirements posed by wearable or implantable approaches.

Coarse-grained reconfigurable approaches, accelerating some computational intensive kernels using a reduced set of hardware resources [17,18], have also been presented. In this case, hardware reuse maximization is combined with latency minimization to satisfy the relatively strict timing constraints. However, even if providing a certain degree of flexibility, such approach is not programmable and does not provide adequate support to prospective improvements of the device, that could be obtained by means of updated versions of the decoding algorithm.

In [19], a homogeneous MPSoC architecture has been used to preliminary test the porting of a neural signal decoding algorithm on parallel processing platforms. Results have shown that real-constraints can be satisfied by clocking the system at a reasonable frequency and taking profit from the parallelism to reduce power consumption using a clock-gating programmable manager. The application code has been parallelized effectively using an approach based on software pipelined.

Our work is in line with [19], since it presents a multi-processor architecture, but it enhances the fine tuning of the system with respect to the considered application, exploiting micro-architecture customization by adopting Application Specific Instruction-set Processors (ASIPs) as building blocks. In [20] and in [21] a similar analysis is performed, building a final system that consists in a custom heterogeneous MPSoC structure. However such works do not consider the exploitation of instruction-level parallelism as a method to reduce computation latency, thus a main point of novelty of our work is the extension of the design space exploration to the evaluation of internal parallelism in multiple issue processor architectures. This paper is an extension of [20], that makes a step forward in the evaluation of the MPSoC paradigm in neural signal processing by considering the execution of the decoding algorithm on multiple acquisition channel.

3. General application and design strategy overview

This paper aims at evaluating the possibility of designing a custom ASIP-based MPSoC implementing an integrated neural signal decoding algorithm. The main objective, as represented in Fig. 1, is to target a digital processing platform capable of processing inputs acquired by invasive electrodes applied on PNS terminations, to ease the control of a prosthetic bionic hand. In the prospective system, the electric signals are firstly treated by an analog front-end that applies a first denoising stage and converts them to digital format. Several channels are acquired in parallel, the number of channels can vary, but 8 is a typical value for state-of-the-art electrodes. The signal samples corresponding to the different channels are produced in input to the Digital Processing Platform, which is the subject of this paper. As shown in Fig. 1, the platform executes three main actions: it cleans the signal from the noise, detects spikes and identifies their similarity with a set of known templates stored in memory. The execution of these steps allows to propagate information to a classifier, implemented in a non implantable platform with more relaxed power constraints, that actually recognizes the motion intention and controls the mechatronic part of the system. The overall design strategy that we have chosen in this paper has two main key points:

- we choose to use ASIP technology, to allow implementation of a power- and performance-effective solution, without giving up on programmability. After the design, a user, prospectively a biomedical engineer or a clinician, can tune the parameters of the algorithm using only software programming. This operation does not require a hardware expert to manage a refinement of the algorithm. Moreover, in case of implanted solution, new software releases can be downloaded to the device simply as new instruction bitstreams.
- An integrated solution has to be developed to reduce the communication bitrate required to send information to the external environment. Moving the spike sorting to the non-implantable processing platform working as classifier would require all the samples acquired by the electrode to be transmitted in output through the communication mean between the two platforms. Especially in case of an implanted device, this link is expected to be a low-power wireless communication module, thus with unsufficient bitrate to transmit such a data rate. In an integrated solution, after the spike sorting, a simple ID and a time stamp information are associated to each detected event, thus the required channel capacity is significantly decreased.

4. Target application and constraints

As mentioned, three subsequent phases, described in more details in the following sections, are addressed in this paper:

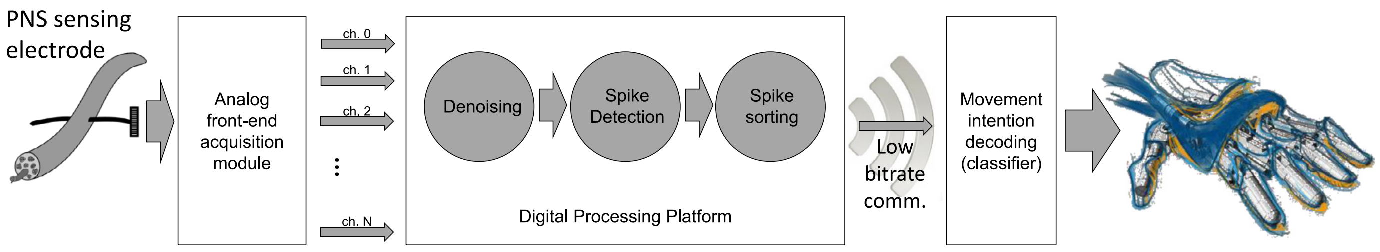
PNS sensing electrode
Analog front-end acquisition module
ch. 0
ch. 1
ch. 2
ch. N
Denoising
Spike Detection
Spike sorting
Digital Processing Platform
Low bitrate comm.
Movement intention decoding (classifier)

Fig. 1. Target application overview.

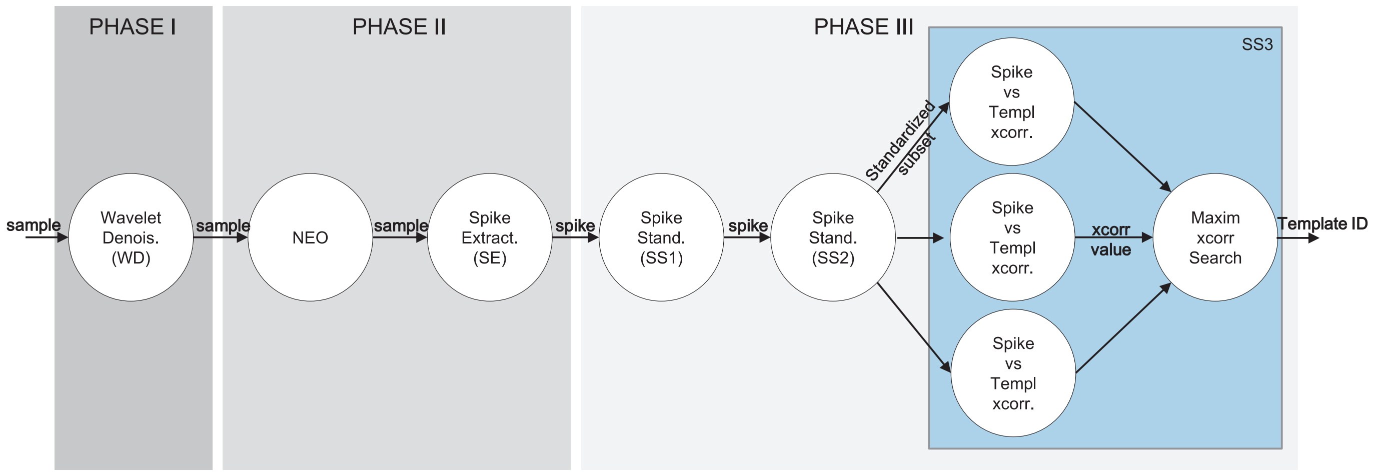

Fig. 2. Application task graph.

- *Wavelet Denoising* – It removes the noise that affects the signal to be processed.
- *Spike Detection* – It extracts the spikes from the filtered input signal.
- *Spike Sorting* – It identifies which motor neuron has generated the detected spike. The algorithm calculates for each spike its cross-correlation with all the elements in a set of known reference spike templates and outputs the spike-template couple with the highest similarity.

The results of the *Spike Sorting* phase are sent as output, through a wireless network interface. The bandwidth required to communicate with the external environment is minimal. While sending in output all the *raw* acquired signal would require a communication bandwidth of around 2 Mbit/s, embedding the *Spike Sorting* phase in the integrated processing device allows to send only the ID of the most similar template. Taking into account that the firing rate of single motor neurons at the level of the hand is typically between 8 and 25 spikes per second [22], the low number of neurons per active electrode site (which is from 8 to 10 [23]) and the number of active sites of an intraneural electrode (usually 8–12, [24]), such communication requires a much reduced bandwidth, around 35 Kbit/s.

The *Movement Intention Decoding*, achieved by means of classification, is not addressed in this paper. It can be performed on the prosthetic hand, by the non-implantable controller. This will allow having more relaxed requirements in terms of power consumption.

In [19], the digital processing section of the targeted application, integrating the above-mentioned phases that are going to be prospectively performed on the implantable device, has been executed on a general purpose processor in order to characterize the computation. As a result of such a profiling and of the evaluation of the execution latency of each phase, the computation has been partitioned into the tasks depicted in Fig. 2. This task graph, depicting the processing flow, highlights data dependency and available parallelism. In this paper, since we are referring to the same algorithm, the same partitioning will be also adopted. The rest of this section provides some insights of each single processing phase, with special emphasis on the constraints characterizing them.

4.1. Phase I – Wavelet Denoising

The *Wavelet Denoising* phase is composed of a unique task, the *Wavelet Denoising (WD)* one. This task processes the input samples to remove the additive noise affecting the signal of interest. An example of the signal pre and post *Wavelet Denoising* is depicted in Fig. 3. In this phase a three-step non-linear filtering scheme, based on multi-level decomposition, thresholding and reconstruction of the input signal is implemented.

Algorithmically, the choice of the mother wavelet determines the structure of the filters composing the processing trellis. In case of orthogonal wavelets, such as the Haar one chosen in this work, they are Quadrature Mirror Filters (QMF). QMF, coupled to a decimated schema, leads to very efficient solutions. Since such approaches cannot ensure shift invariance, which is important to preserve the shape of short spiky signals, an undecimated *à trous* implementation has been selected here as in [25]. This solution is more expensive both computationally and in terms of memory usage.

With an undecimated schema, the processing trellis is simply a cascaded filter bank, with the thresholding step halfway, so the dominant processing operation is the convolution. In a streaming processing model (computation of one sample out for each sample in), this reduces to several dot products, in turn composed of multiply-and-accumulate operations. According to [5], the sam-

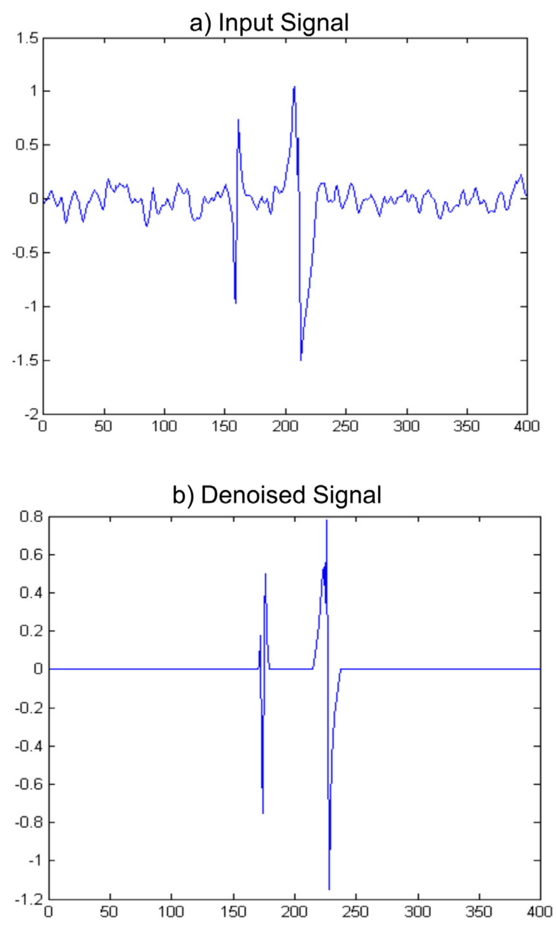

Fig. 3. Phase I: effect of Wavelet Denoising.

pling rate of the incoming neural signal is 12 ksamples per second. The thresholds applied within the thresholding phase are four and must be computed offline and pre-loaded at the bootstrap of the prosthetic device.

4.2. Phase II – Spike Detection

The *Spike Detection* phase is composed of two tasks, namely *NEO* and *Spike Extraction (SE)*. *NEO* stands for *non-linear energy operator*. This task computes the NEO feature signal, defined as:

$$NEO\{x[n]\} = x^2[n] - x[n+1] \cdot \; x[n-1] \tag{1}$$

on the *Wavelet Denoising* output *x*. The spike detection threshold is computed as a scaled version with an empirically chosen constant C of the mean value of the NEO in a predefined window of N samples:

$$Th = C\frac{1}{N}\sum_{n=1}^{N} NEO\{x[n]\} \tag{2}$$

Spike identification is then left to the *SE* task, which selects the relevant windows in the NEO signal: those above the computed threshold are recognized as windows containing spike activity. The identified samples, which contain a spike, are stored in a buffer to be passed to the *Spike Sorting* phase.

An example of the signal pre and post *Spike Detection* is depicted in Fig. 4 where, starting from the input denoised signal, a clearly identified spike is extracted.

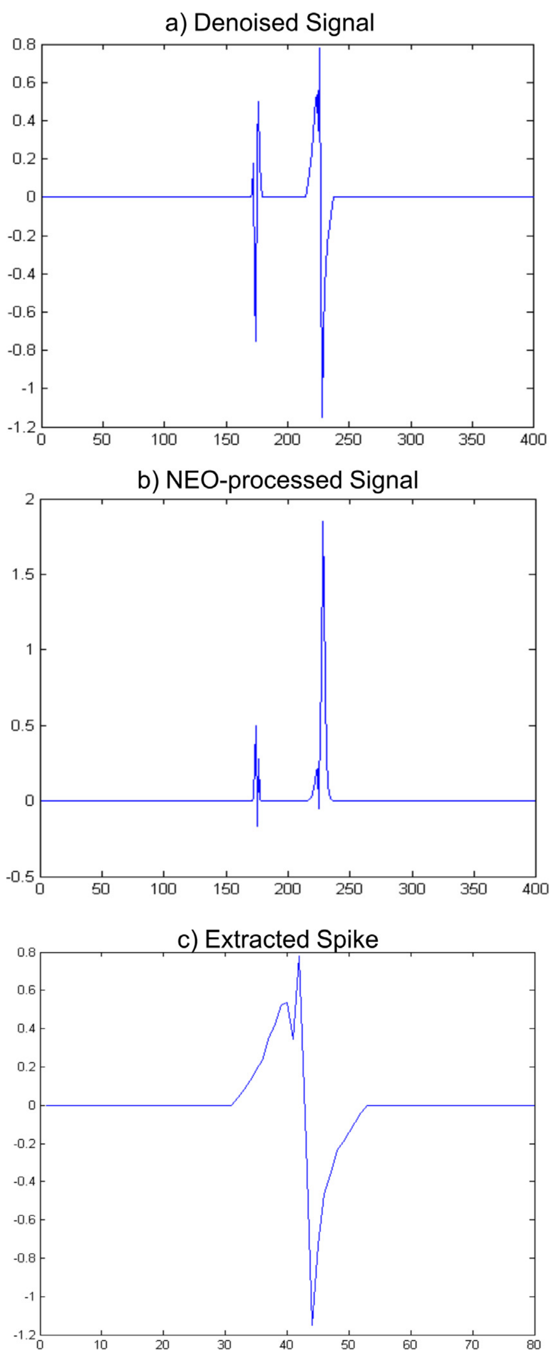

Fig. 4. Phase II: effect of Spike Detection.

4.3. Phase III – Spike Sorting

The *Spike Sorting* phase is by far the most computationally-intensive one. It is composed of several different tasks. It is responsible for identifying which motor neuron has produced the considered spike. This identification process is performed by determining the similarity of an incoming spike, produced as an output by the *Spike Detection* phase, with the *i*th reference template. To do that, the maximum Pearson's correlation coefficient, descending from the cross-correlation analysis, is associated to the spike-template couple.

In the proposed implementation, both the maximum number of samples in a spike and in a template (*Len*) has been set to 40 (3.3 ms). To evaluate the best alignment, for each detected spike, the relative samples are stored in a buffer with $2 \cdot \mathit{Len} = 80$ locations, placing the element with the maximum amplitude value in the center of the buffer. Then the algorithm starts calculating the Pearson's correlation coefficient for each different overlapping between the buffer and the reference template, and the maximum value is taken as result. Each coefficient is obtained considering a window of *Len* samples out of $2 \cdot \mathit{Len}$, then shifting the window by one position to evaluate a new overlapping.

In [19], as shown in Fig. 2, the *Spike Sorting* phase is partitioned in three steps. The first two steps perform some pre-processing, while the third one is in charge of the actual coefficients calculation and template identification.

- The *Spike Standardization (SS)* tasks: the reference algorithm adopts a normalized cross-correlation method, thus samples are pre-processed in order to have zero mean and unit variance (we assume the same processing to be performed offline for the reference templates). These tasks take place only once per spike, for each different overlapping out of *Len*.
- The *Spike vs template cross-correlation* tasks: different templates have to be cross-checked to determine which neuron produced the spike, thus cross-correlation has to be computed for each of them. Nevertheless, parallel computations can be performed and the corresponding node in the task graph can be replicated to evaluate, in a concurrent manner, the similarity with different sets of templates on multiple processors.
- The *maximum cross-correlation search* task: produces the *Spike Sorting* step output. It extracts the *template ID*, which is the index of the template with the highest cross-correlation coefficient. This index is the only information that will be signalled outside to the non-implantable controller for the final classification of the movement intention.

The *spike vs template cross-correlation* tasks along with the *maximum cross-correlation search* task are referred, later on in this paper, to a unique one identified as *SS3*.

4.4. Real-time constraints

As already stated, the sampling frequency considered during the development of the system is 12 kHz. To be functional, the system has to comply with two real-time constraints:

a) the *wavelet denoising, NEO calculation* and *spike extraction* tasks have to be repeated for each input sample;
b) the *spike sorting* has to be repeated for every spike. We assume a maximum spike rate corresponding to one spike every 23 samples. This assumption is quite conservative, according to information obtained from in-vivo experiments. This means that, in the worst case condition, nearly 522 spikes/s are present. Such a number is largely higher than the one that could be computed considering the typical firing rate of the motor neurons and the typical number of such neurons recorded by the same active site of an intraneural electrode.

5. Target architectural template

5.1. Macro-architectural template

We have considered a general template for heterogeneous MPSoCs, depicted in Fig. 5. The proposed system necessarily embeds a host processor, in charge of interfacing the system with the external environment, performing the more "control-like" functionalities and dispatching the samples to be processed to the other processors in the system for "data crunching". Such other processors are customized instances of an Application-Specific Instruction Set template. The features of the micro-architectural template will be described more in detail in Section 5.2. The processors are connected using a multi-layer system bus and by a set of point-to-point FIFO connections. The FIFOs are two 32-bit words wide and are used for signalling within the handshake needed to implement the application level flow-control, as discussed in Section 6.

Similar templates, exposing in general the same optimization knobs to the designer, can be constructed using different industrial and academic solutions available for the community, such as [26–30]. All these toolsets can generate simulation models, HDL description for implementation, and provide a re-targetable compiler that adapts compilation to the selected processor and system configuration.

5.2. Micro-architectural processor template

The processor IPs that are used as building blocks in the architectural template are design-time configurable VLIW processors. Each processor can be composed of parallel datapath slices (*issue slots*) capable of executing a configurable set of operations. These operations may be selected from a library, which can be enriched by the designer to implement frequently used non-standard operations effectively exploitable by the target algorithm.

Each issue slot has a register file and a pass unit that can be used to move data from one register file to the other. The first issue slot acts as a *sequencer*. It manages the program counter and a status register to control the program flow.

Each processor has a program memory and a data memory and can access an arbitrary number of FIFO-based connections through dedicated ports. Given these general assumptions, the micro-architectural parameters that we explore are:

- set of operations inside each issue slot. The operations that have been used in this work were selected from a standard list including arithmetic and logic, program flow management, compare, shifting and memory access operations. Standard multiplication and multiply-and-accumulate operations have also been exploited. However, to achieve better efficiency, a custom instruction combining multiplication, shifting and accumulation was defined. Such instruction has been very useful to implement convolutions of fixed point values, where the multiplication result has to be shifted before accumulation to avoid overflow. The partitioning of the operations between the issue slots is tailored to improve the utilization factor, according to the operation scheduling required by the execution of target algorithm microcode.
- register file size
- memory size
- number of FIFO ports

6. Reference programming model and communication primitives

The target algorithm is a typical streaming application. We can speed up the execution implementing a software pipeline, exploiting a model of computation (MoC) based on process networks, basically derived from Kahn Process Networks (KPN) [31]. Such MoC is based on parallel tasks that communicate through FIFOs.

Each task has to be represented as an iteratively repeated functional actor. At each iteration a task receives a given amount of input data, called *token*, from one (set of) FIFO(s); then, the task elaborates it (executing a *compute* function, which represents the actual computation workload of the parallel task) and writes the output to the downstream FIFOs.

Provided that enough buffering resources are instantiated between different processes, the elaboration step envisioned in each process step can run in parallel on subsequent tokens, thus the

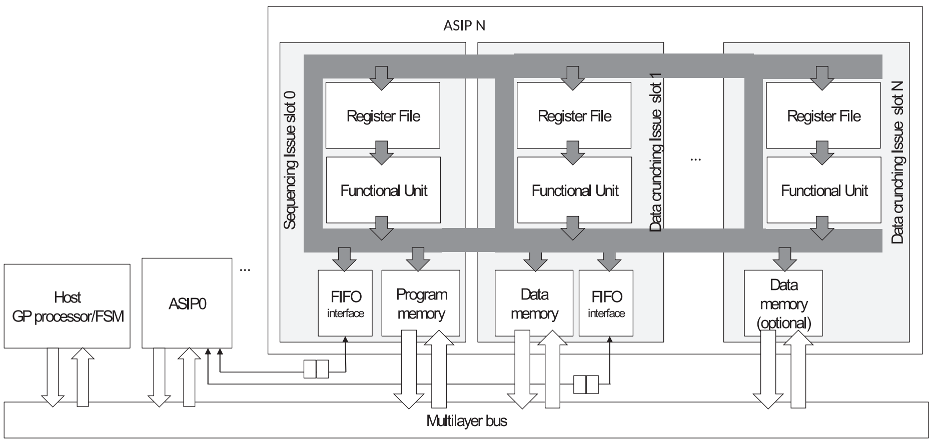

Fig. 5. Architectural template overview. *ASIP n* shows also the internal micro-architecture of the ASIPs.

throughput is limited by the execution time of the slowest task in the network. Performance can be pre-estimated using an analytical model of the dependence of the throughput on the node workloads [32]. presents an approach for modelling the throughput of a KPN network, by calculating the throughput τ_{P_i} of every KPN process. Each process P_i of the KPN can be annotated with a workload number W_{P_i}:

$$W_{P_i} = C^{P_i} + x \cdot C^{R_d} + y \cdot C^{W_r},$$

where C^{P_i} denotes the number of time units (i.e. clock cycles) required to execute the process function once, x and y denote how many FIFOs are read and written per process firing, and C^{R_d} and C^{W_r} denote the communication costs. The throughput of each process is, hence, $\tau_{P_i} = \frac{1}{W_{P_i}}$. The overall KPN throughput is denoted by τ_{out} and is defined as the average number of tokens produced by the network per time unit. Since the slowest process determines the system throughput [32], then $\tau_{out} = \tau_{P_{slowest}}$.

To find an effective mapping solution, the pipeline stages have to be as balanced as possible. To be practically implemented, the KPN nodes, theoretically communicating through unbounded FIFOs, have to use blocking *read()* and *write()* primitives.

In this work, we have implemented an efficient set of primitives that exploits processor internal memories for data exchange and uses FIFOs only for synchronization. This approach has two advantages:

- the communication is copy-free. Within each communication channel between a *producer* and a *consumer*, the token data structure is mapped at a memory address that corresponds to the consumer local memory. This way, the token is directly produced in place and can be consumed without further transfers. Exploiting *ping-pong buffering* production and consumption of tokens can be overlapped [33];
- As the communication happens through software FIFOs implemented in local memories, big token data structures can be exchanged, avoiding the instantiation of power and area-hungry hardware FIFO structures.

7. Design space exploration results: single channel

In order to assess the exploitability of the MPSoC paradigm in this application domain, we performed a design space exploration. We considered some design points that feature different clusterings of the application tasks on the processing elements, implementing the algorithm on a single train of sample coming from a single acquisition channel. For each clustering, a micro-architectural DSE was performed in order to find a customized ISA for each processor.

Table 1
Workload (expressed in clock cycles) associated with the computation and communication tasks mapped on each ASIP for different mapping configurations.

Config	Host (cycles/sample)	ASIP1 (cycles/spike)	ASIP2 (cycles/spike)	ASIP3 (cycles/spike)
Mapping 1	1.8K	25.67K	16.6K	33.28K
Mapping 2	1.8K	42.27K	33.28K	-
Mapping 3	1.8K	75.55K	-	-

The latency related with the execution of the *wavelet denoising, NEO calculation* and *spike extraction* tasks, when mapped on a general purpose host processor, is known from [19] (about 1.8K cycles). We decided to consider such latency acceptable, since the target constraint *a)* sets a minimum required working frequency of about 22 MHz, thus we kept such three tasks grouped on the host. This means that in all the explored mappings, presented in Fig. 6, the host executes the *WD, NEO* and *SE* tasks. Fig. 6 highlights also the different choices that have been explored with respect to the clustering of the *spike sorting* tasks. Three solutions have been evaluated, hereafter referred as Mapping 1, Mapping 2 and Mapping 3.

Table 1 shows the latency associated with the execution of each pipeline stage in the chosen mappings. Figs. 7, 8 and 9 show the execution traces obtained from the FPGA prototyping of the evaluated system configurations. In such traces, for the ASIPs, the black bars represent the time slots dedicated to task execution. All the traces are obtained stimulating the system with a worst case synthetic input signal exposing the maximum spike rate (one spike every 23 samples). Here follows the analysis of the chosen configurations.

7.1. Mapping 1

As a first solution, we tried to exploit the maximum degree of task-level parallelism, instantiating one ASIP for each parallel task (see *Mapping 1* in Fig. 6). In this way, it is possible to minimize the latency and to customize the processor architecture for each task, in order to achieve maximum efficiency. The design results in a solution with three ASIPs. As may be noticed by the corresponding execution trace, the limiting pipeline stage is the process executed by the host. Denoising and spike detection (considering 23 iterations, to be capable of decoding a spike composed of 23 samples) take about 42 K cycles. The minimum frequency is set by constraint *a)*, thus is about 22 MHz.

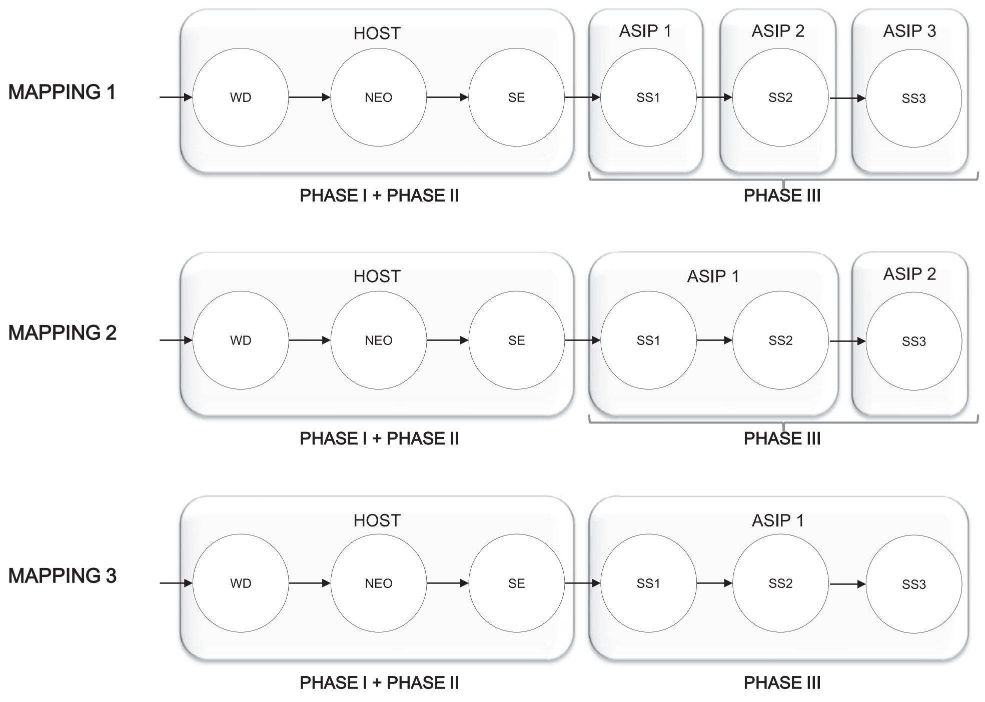

Fig. 6. Adopted mapping choices.

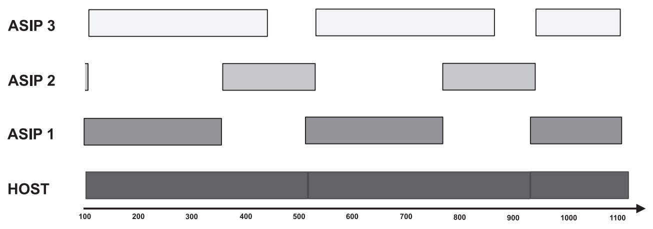

Fig. 7. Execution traces associated to Mapping 1.

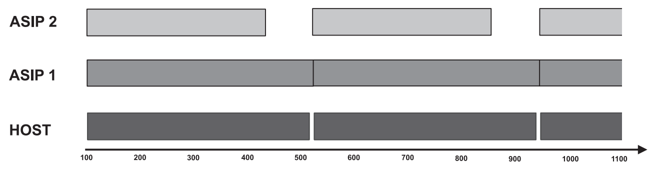

Fig. 8. Execution traces associated to Mapping 2.

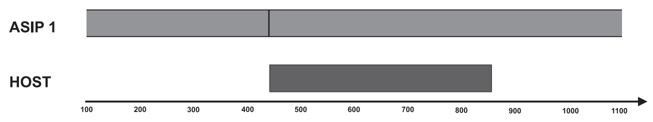

Fig. 9. Execution traces associated to Mapping 3.

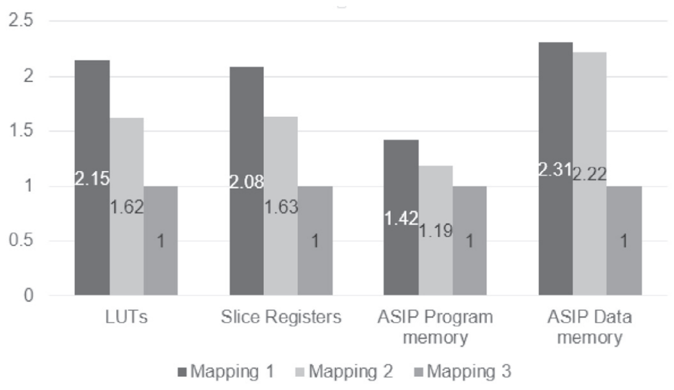

Fig. 10. Hardware resources required by the system configuration for the three mappings, normalized with respect to *mapping 3*.

7.2. Mapping 2

Inthe second solution, we merged the two standardization steps in one single ASIP (see *Mapping 2* in Fig. 6). The host processor is no longer the limiting node in the pipeline. The function with highest execution latency is the standardization. The minimum required clock frequency has thus to be fixed, to respect constraint *b)*, to about 23 MHz.

7.3. Mapping 3

In the third solution, we merged the entire spike sorting, corresponding to Phase III in the application task graph, in one single ASIP (see *Mapping 3* in Fig. 6). The limiting pipeline stage is obviously executed by ASIP 1. To respect constraint *b)*, in this case the frequency has to be increased to about 40 MHz.

8. Hardware architecture evaluation

8.1. Evaluation on FPGA

Throughput optimization through parallelism allows reducing the working frequency required to process the signal in real-time. On the other hand, instantiating more processors on the die obviously costs in terms of area and power.

In Fig. 10 we show the hardware resources occupied by the implementation of the chosen system configurations on a Xilinx Artix-7 FPGA. The target device is a XC7A35T-1CPG236C FPGA, specifically designed to reduce development costs (the FPGA price

Table 2
Actual ASIP utilization (% of the total pipeline cycles).

ASIP processor	% of active cycles
Mapping 1 – ASIP1	62%
Mapping 1 – ASIP2	40%
Mapping 1 – ASIP3	80%
Mapping 2 – ASIP1	100%
Mapping 2 – ASIP2	79%
Mapping 3 – ASIP1	100%

is around 40 USD) and power consumption, being realized on TSMC's 28 nm high performance, low power (HPL) process. This makes the device a promising technology target for a wearable implementation of the neural signal decoding system. It may be noticed that, as it is logical to expect, the utilization of registers and slices is more than doubled going from one to three ASIPs. Moreover, even if the application code and data segments are partitioned between the ASIPs, the size of the memory modules that have to be instantiated in the system also increases significantly. Such increase is basically due to the overhead related to the data memory space reserved for the communication tokens (using ping-pong double buffering the application has to reserve space for two token data structures for each communication channel) and to the program memory space needed to store the communication routines and the write and read APIs.

Nevertheless, exploiting parallelism can also bring advantages with respect to power and energy consumptions, for three main reasons:

- Target working frequency is reduced, thus more relaxed implementation constraints can be used within the implementation flow, prospectively leading to the use of more low power cells in the design and to less power-hungry netlists.
- As may be noticed by the scheduling traces in Figs. 7 and 8, the ASIPs are stalled during part of the pipeline cycle waiting on blocked communication channels. Their power consumption can be reduced or switched off when not active, with clock- or even power-gating techniques. The actual percentage of activity for each processor in the different mappings is reported in Table 2.
- As presented in [19], spikes may be very sparsely detected in the input signal. Thus having the host processor taking care of all the execution steps needed for spike detection enables the ASIPs to be activated only when meaningful neural information has to be decoded (around 10% of the time in physiologically meaningful signals), further reducing their contribution to the overall power consumption.

In order to provide an estimation of the actual power consumption of the FPGA implementation of the system, in Fig. 11 we present the activity-based estimation of the power consumption for the three considered mappings. As may be noticed, due to the reduction of the working frequency, both Mapping 1 and Mapping 2 dissipate less power than Mapping 3, even if the hardware architecture corresponding to Mapping 3 uses fewer resources. However, Mapping 2 is the more power-efficient, thus the additional parallelism exploited in Mapping 1 is not optimal from a power-consumption point of view. The overall power consumption of the computing system for Mapping 2, assuming a continuous (unrealistic) neural activity, is below 200 mW. Considering a standard Li-Ion single cell battery (3.7 V, 2500 mAh capacity), such power consumption corresponds to a battery lifetime of nearly 48 h, thus ensuring a comfortable use of the device in real-life prostheses.

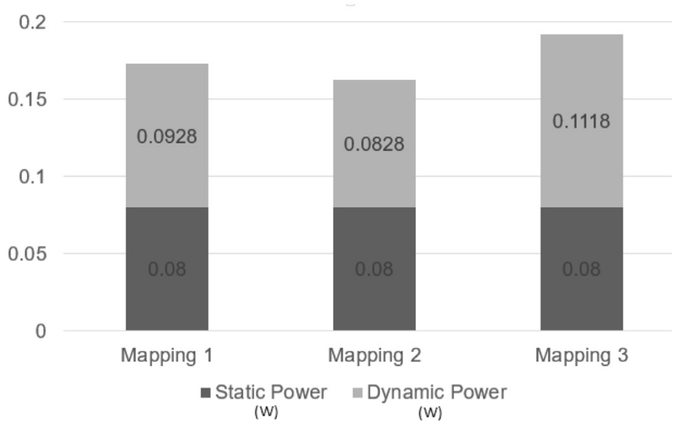

Fig. 11. Power consumption of the system configuration for the three mappings. Static and dynamic contributions to the overall power consumption is shown.

8.2. Evaluation on ASIC 40 nm technology

To evaluate power consumption of the ASIPs in prospective ASIC implementation, we have used as reference technology a 40 nm high-VT industrial standard cell library. The power figures reported in the following are obtained after post-routing simulation-based experiments. Within the power evaluation flow, as a first step, we performed physical synthesis, floorplanning and place and route using Cadence RTL-Compiler and Cadence Encounter. During the implementation flow, the floorplanning phase required specific manual effort. We initially executed the floorplanning exploiting Encounter's dedicated utility. This required manual specification of the directives for the floorplanning tool, describing the hierarchy in the synthesized netlist and listing the modules for which the tool has to define area region constraints (i.e. in the considered use case the ASIP cores). We had later to manually refine the automatically generated floorplanning to remove overlapping violations.

After the floorplanning, we performed placement and routing, enabling in-place optimization to fix the timing violations evaluated in the post-floorplanning phase. The resulting netlist was used to perform a simulation with Mentor Questasim, using the HDL models of the standard cells provided with the library. During the post-routing simulation, the program memories were loaded with the actual code of the target application, thus the obtained switching activity represents the actual workload to be supported by the system. Such an activity is extracted from the simulation in the form of a SAIF (or a VCD file) that can be given in input to the Cadence tools to estimate power consumption.

An average power consumption of approximately 0.05 mW/MHz for each ASIP was achieved in Mapping 1. This number refers to effective simulation periods, during which the ASIPs were active in actual computation and not stalled on FIFO accesses. This result sets to 5 mW the upper bound for the power consumption of the overall system (considering 4 processors running in parallel at 25 MHz). This level of power consumption makes the system implementation prospectively implantable, since, if adequately packaged, it complies with the constraint related to maximum peak power consumption (80 mW/cm^2) [34], that has to be respected in order to avoid organic tissues damages. Moreover, considering the capacity of commercial implantable batteries [35] (3.6 V, 200 mAh), the expected battery lifetime is around 150 h. The system can be clocked up to 200 MHz using the low-power technology node in the library, without timing violations. Therefore, multi-channel neural signal decoding can be prospectively implemented, increasing the frequency to respect real-time constraints. Table 3 summarizes the prospective ASIC implementation features.

Table 3
Summary of the hardware-related features of a prospective ASIC implementation of the neural decoding system.

Maximum working frequency (High Vt technology node)	200 MHz
Power consumption of a single ASIP	0.05 mW/MHz
Power consumption of a 4-processor system configuration (Mapping 1)	5 mW
Considered battery features	3.7 V, 200 mAh capacity
Battery lifetime (active use)	150 h

Table 4
Computation latency reduction by means of instruction-level parallelization.

Number of issue slots	Latency SS1	Latency SS2	Latency SS3	Total
One data memory per issue slot				
1 (1IS-IDMEM)	25672	16600	33278	75550
2 (2IS-2DMEM)	18031 (−29.76%)	10065 (−39.37%)	19724 (−40.73%)	47819 (−36.70%)
4 (4IS-4DMEM)	16052 (−37.47%)	6561 (−60.48%)	13262 (−60.15%)	35874 (−52.52%)
Single data memory				
1 (1IS-1DMEM)	25672	16600	33278	75550
2 (2IS-1DMEM)	19706 (−23.24%)	10065 (−39.37%)	21298 (−36.00%)	51069 (−32.40%)
4 (4IS-1DMEM)	18751 (−26.96%)	6913 (−58.35%)	21055 (−36.73%)	46719 (−38.16%)
[20] (2 ISs)	33K	34K	34K	103K
[21] (1 IS)	32K	19K	76K	129K

9. Design exploration results: multi-channel

According to [21], a multi-channel decoder could be implemented instantiating multiple parallel datapaths, each one dedicated to handle a subset of the recording channels.

In [21] such a solution has been tested, using an ASIC 40 nm technology library as a reference, even if considering slightly different latency numbers, Mapping 3 has been used. Phase I and Phase II are mapped on a host processor. In this paper, we compare that solution with a different approach that exploits instruction-level parallelism. We evaluate the effects on the execution latency provided by means of the instantiation of multiple issue slots inside the ASIP micro architectural template, and the corresponding quality of implementation for both FPGA and ASIC target technology. Basically we exploit the VLIW approach to reduce latencies and, as a consequence, the number of processors that are required to respect real-time constraints. As a first trial, we consider a set of processor configurations, in charge of executing, as envisioned in Mapping 3, *SS1, SS2* and *SS3*. Considered configurations feature one *sequencing* issue slot and 1, 2 or 4 issue slots dedicated to data crunching, each one endowed with a private data memory. In a second experiment, we only provide one data memory, connected to the first *crunching* issue slot. In Table 4 we report the computation latency of the application tasks associated with the use of the different configurations.

As may be noticed, with respect to the implementation presented in [20], both standardization steps have been optimized. A significant reduction of the execution latency of SS1 and SS2 has been achieved basically applying compilation-based techniques (inline functions and loop unrolling) and inserting inside the ASIP ISA an enhanced load-store unit that enables memory access and pointer increment to happen simultaneously. Moreover, comparing the numbers considered in this paper with those presented in [21], again there is a noticeable latency reduction. Readers should consider that in [21] only ASIPs with one single issue slot have been used, while the micro-architectural template considered in this paper is composed by two in its minimal configuration. Further significant optimizations can be obtained, as shown in Table 4, by increasing the instruction level parallelism in ASIPs, especially when moving from a single issue slot to two (36.70% improvement). This is basically due to the possibility, in all tasks, to parallelize actual arithmetic and logic operations with memory accesses. Further increase in the number of issue slots results in more limited latency reduction, that also require significant manual intervention to be applied in order to transform the code so that the parallelism in the ASIP can be effectively exploited. More in detail:

- input data (detected spikes) must be adequately dispatched to the different data memories (if available)
- instrumental data structures, such as array pointers, loop iterators and accumulators must be replicated and distributed over the different data memories
- compiler-based optimization must be limited, in order to improve predictability of the instruction scheduling

As an example, in order to achieve an effective utilization of the 4 issue slots, in Fig. 12 we show the strategy adopted to allow parallel processing in task *SS1*. We have modified the code so that different sections of input spike are processed by different issue slots.

The most computation intensive part of the tasks calculates mean and standard deviation values over 40 input samples. The processing has to be repeated 40 times, over all the different subsets of 40 samples within the 80 samples in the input token. After manual modification, the workload is partitioned among the 4 issue slots. For example issue slot IS1 performs the first 10 iterations starting from the section 0–39, as highlighted in the figure. Code modification has required manual intervention, in Table 5 we show that the latency numbers obtained without adequate manual intervention do not benefit from increased parallelism in the ASIP datapath. Finally, Table 4 shows that latency reduction can be achieved

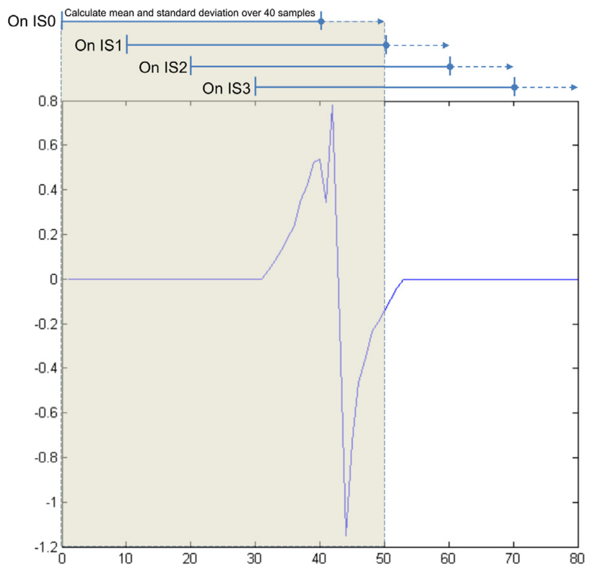

Fig. 12. Parallelization scheme of SS1 task. Partitioning of the workload among four issue slots (IS0, IS1, IS2 and IS3). The solid line represents the single iteration over the considered issue slot, while the dotted one all the subsequent ones.

Table 5
Computation latency without manual intervention.

Number of issue slots	Latency SS1	Latency SS2	Latency SS3	Total
2	24370	17042	22185	63597
4	24288	17041	21932	63261

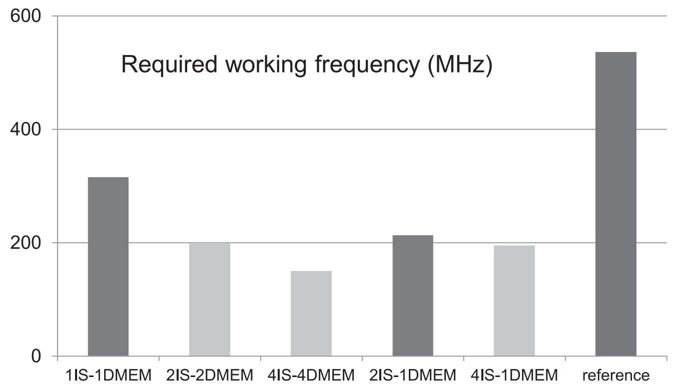

Fig. 13. Working frequency required to respect real-time constraints for different ASIP configurations. The light gray bars indicates required frequency below 200 MHz.

without increasing the number of data memories, thus limiting the cost related with the instantiation of multiple issue slots. Such a solution, on the one hand clearly implies less latency benefits, but on the other is affected by a slightly more limited area and power consumption increase. Therefore it will allow a partial parallelism exploitation in highly constraint power and area scenarios.

10. Hardware architecture evaluation: multi-channel

10.1. Evaluation on ASIC 40 nm

Once the latency associated with the computation tasks, corresponding to different ASIP architectures, has been identified, it is possible to evaluate hardware-related characteristics of prospective system architectures integrating the ASIPs. To evaluate power consumption of the different considered architectures we have used the power evaluation flow described in Section 8.2. As a first step we have evaluated the possibility of using one single (VLIW) ASIP to process all the decoder input channels. This poses a required working frequency derived by the previously mentioned real-time constraints. As may be noticed in Fig. 13, only three configurations, in light gray, can respect constraints when clocked at a frequency lower than 200 MHz. This means that only in these cases, we can implement the multi-channel decoder exploiting only one ASIP to execute Phase III.

In all the other cases, two different ASIPs have to be used, each one assigned with the workload related with 4 out of the 8 input channels. The bar named as *reference* represents the configuration presented in [21], that needs three parallel datapaths to respect real time constraints. Fig. 14 shows the power consumption of all the considered ASIP configurations. The numbers reported in the table account for the whole ASIP population inside the system, that is summarized in Table 6.

The most power-efficient configuration is the one with two IS and two data memories, both with respect to peak and to average power consumption (average power consumption takes into account that in the considered configuration the computation terminates before the deadline posed by real-time constraint, so the processor goes in the idle state for the remaining time, dissipating around 25% of the *active* power consumption [21]). The higher power efficiency of this ASIP with respect to more parallel processors is mainly due to the smaller width of its program memory. Increasing the number of issue slots, the instruction becomes longer to accomodate the additional opcode and control fields, thus the power dissipation of the program memory counterbalances the benefits obtained with the increased parallelism. Fig. 14 shows also average power consumption when power-gating is applied to idle cores. Also in this case, configurations with two *crunching* issue slots are more power-efficient than the others for the considered workload. This highlights the application- and technology-dependent nature of the benefits achievable with the exploitation of instruction-level parallelism.

The system architecture with minimal power consumption can be used to build a device with the characteristics highlighted in Table 7. If we consider continuous neural signal activity on all the channels, the system can work for around 24 h with the energy supplied by an implantable battery.

It is worth to point out that the reported power consumption assumes an unrealistic spike rate and that at each iteration the spike sorting task will have to be executed only for those channels which have detected a spike. This may cause power consumption to be much lower than what is reported in Table 7. The

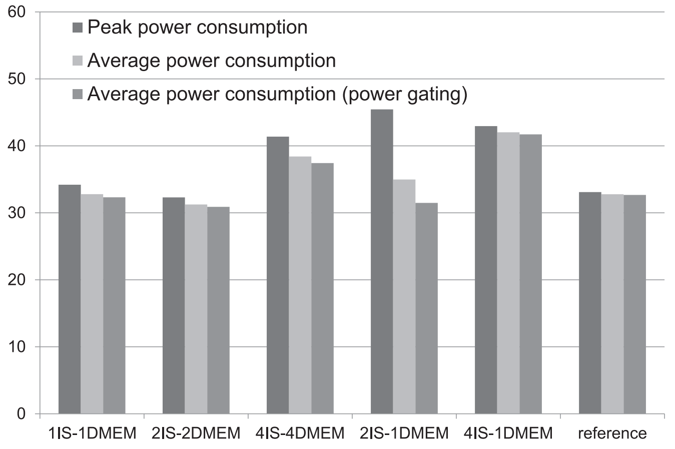

Fig. 14. Power consumption (peak, average and average with power gating) of different ASIP-based system configurations.

Table 6
ASIP population and working frequency in the explored system configurations .

Considered ASIP architecture	ASIPs executing WD and SD	ASIPs executing spike sorting	System working clock frequency (MHz)
1IS-1DMEM	1× single-IS	2 × 1IS-1DMEM	170
2IS-2DMEM	1× single-IS	1 × 2IS-2DMEM	200
4IS-4DMEM	1× single-IS	1 × 4IS-4DMEM	170
2IS-1DMEM	1× single-IS	2 × 2IS-1DMEM	170
4IS-1DMEM	1× single-IS	1 × 4IS-1DMEM	170
reference	1× single-IS	3× single-IS	196

Table 7
Features of the prospective implantable ASIC implementation of the multi-channel neural decoding system. Battery features: 3.7 V, 200 mAh capacity.

	Multi channel use-case
# of channels	8
Average power consumption @ 200 MHz	31.2 mW
Peak power consumption @ 200 MHz	32.3 mW
Battery lifetime (active use)	~ 24 h

approach exploiting instruction-level parallelism presented in this paper, if compared with the strategy used in [21], where one very simple processor is instantiated for each channel, improves worst-case power consumption figures but loses predictability, since the workload in a given iteration depends on the activity of multiple channels. The optimization of power consumption with power-gating techniques is more difficult to implement when more channels are clustered on one single processor, thus a more partitioned approach may be preferable when shifting to more leakage-dominated technology nodes.

10.2. Evaluation on FPGA

Building a convenient on-FPGA implementation of the decoding algorithm is not really feasible. Considering XC7A35T-1CPG236C as a target FPGA device, the maximum working frequency of the ASIP configurations on the target FPGA device is around 55 MHz, thus only two channels can be processed by the host processor.

11. Conclusion

In this work we have evaluated the MPSoC paradigm within the development of power efficient neural signal decoders. Starting from a state-of-the-art algorithm, as reference target application, we have performed a comparison of different MPSoC-based system configurations. Considered solutions feature a differently partitioned application, executed on a custom heterogeneous architecture, realized using instruction-level parallel ASIPs. We have firstly studied the decoder considering a single channel use-case, then we have explored the use of multiple issue-slots in the ASIPs as a method to implement a multi-channel use case. Considering a prospective ASIC implementation on a 40 nm technology, evaluated using post-synthesis power evaluation, we have demonstrated that the multi-channel implementation can guarantee peak and average power consumption savings. The estimated power consumption for the system is around 32 mW (24 h lifetime of an implantable battery) and is compliant with the constraints posed by a prospective implantable neural decoding device. However, we have also shown that this multi-channel approach requires significant manual intervention on the software application, thus posing for the users the requirement of advanced skills in parallel programming.

Acknowledgments

The research leading to these results has received funding by the Region of Sardinia in the ELoRA project (Fundamental Research Programme, L.R. 7/2007, grant agreement CRP-60544), by the European Commission in the NEBIAS project (FP7, FET Proactive, grant agreement 611687) and by the Italian Government in the HANDBOT project (PRIN2010/11, prot. 2010YF2RY_003).

References

[1] M. Tombini, J. Rigosa, F. Zappasodi, C. Porcaro, L. Citi, J. Carpaneto, P. Rossini, S. Micera, Combined analysis of cortical (EEG) and nerve stump signals improves robotic hand control, Neurorehabil. Neural. Repair. 26 (3) (2012) 275–281.

[2] M. Lewicki, A review of methods for spike sorting: the detection and classification of neural action potentials, Netw. Comput. Neural Syst. 9 (4) (1998) 53–78.

[3] A. Diedrich, W. Charoensuk, R. Brychta, A. Ertl, R. Shiavi, Analysis of raw microneurographic recordings based on wavelet de-noising technique and classification algorithm: wavelet analysis in microneurography, IEEE Trans. Biomedl Eng. 50 (1) (2003) 41–50.

[4] R. Normann, A. Branner, A multichannel, neural interface for the peripheral nervous system, in: Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings, 4, 1999, pp. 370–375.

[5] L. Citi, J. Carpaneto, K. Yoshida, K.-P. Hoffmann, K.P. Koch, P. Dario, S. Micera, On the use of wavelet denoising and spike sorting techniques to process electroneurographic signals recorded using intraneural electrodes, J. Neurosci. Methods 172 (2008) 294–302.

[6] P.M. Rossini, S. Micera, A. Benvenuto, J. Carpaneto, G. Cavallo, L. Citi, C. Cipriani, L. Denaro, V. Denaro, G.D. Pino, et al., Double nerve intraneural interface implant on a human amputee for robotic hand control, Clin. Neurophysiol. 121 (5) (2010) 777–783.

[7] F. Zhang, M. Aghagolzadeh, K. Oweiss, A fully implantable, programmable and multimodal neuroprocessor for wireless, cortically controlled brain-machine interface applications, J. Signal Process. Syst. 69 (3) (2012) 351–361, doi:10.1007/s11265-012-0670-x.

[8] B. Yu, T. Mak, X. Li, F. Xia, A. Yakovlev, Y. Sun, C.-S. Poon, Real-time fpga-based multichannel spike sorting using hebbian eigenfilters, Emerg. Sel. Top. Circuits Syst., IEEE J. 1 (4) (2011) 502–515, doi:10.1109/JETCAS.2012.2183430.

[9] T.-C. Chen, W. Liu, L.-G. Chen, 128-channel spike sorting processor with a parallel-folding structure in 90nm process, in: Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on, 2009, pp. 1253–1256, doi:10.1109/ISCAS.2009.5117990.

[10] Y. Perelman, R. Ginosar, An integrated system for multichannel neuronal recording with spike/lfp separation, integrated a/d conversion and threshold detection, Biomed. Eng., IEEE Trans. 54 (1) (2007) 130–137.

[11] Z.S. Zumsteg, C. Kemere, S. O'Driscoll, G. Santhanam, R.E. Ahmed, K.V. Shenoy, T.H. Meng, Power feasibility of implantable digital spike sorting circuits for neural prosthetic systems, Neural Syst. Rehabil. Eng., IEEE Trans. 13 (3) (2005) 272–279.

[12] D. Chen, L. Wang, G. Ouyang, X. Li, Massively parallel neural signal processing on a many-core platform, Comput. Sci. Eng. 13 (6) (2011) 42–51.

[13] S. Gibson, J.W. Judy, D. Marković, An fpga-based platform for accelerated offline spike sorting, J. Neurosci. Methods 215 (1) (2013) 1–11.

[14] V. Karkare, S. Gibson, D. Markovic, A 130-μw, 64-channel neural spike-sorting dsp chip, Solid-State Circuits, IEEE J. 46 (5) (2011) 1214–1222, doi:10.1109/JSSC.2011.2116410.

[15] M. Montani, L. De Marchi, A. Marcianesi, N. Speciale, Comparison of a programmable dsp and fpga implementation for a wavelet-based denoising algorithm, in: Circuits and Systems, 2003 IEEE 46th Midwest Symposium on, 2, IEEE, 2003, pp. 602–605.

[16] S. Gibson, J.W. Judy, D. Markovic, Technology-aware algorithm design for neural spike detection, feature extraction, and dimensionality reduction, Neural Syst. Rehabil. Eng., IEEE Trans. 18 (5) (2010) 469–478.

[17] N. Carta, C. Sau, F. Palumbo, D. Pani, L. Raffo, A coarse-grained reconfigurable wavelet denoiser exploiting the multi-dataflow composer tool, in: 2013 Conference on Design and Architectures for Signal and Image Processing, Cagliari, Italy, October 8–10, 2013, 2013a, pp. 141–148.

[18] N. Carta, C. Sau, D. Pani, F. Palumbo, L. Raffo, A coarse-grained reconfigurable approach for low-power spike sorting architectures, in: Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on, 2013b, pp. 439–442, doi:10.1109/NER.2013.6695966.

[19] N. Carta, P. Meloni, G. Tuveri, D. Pani, L. Raffo, A custom mpsoc architecture with integrated power management for real-time neural signal decoding, IEEE J. Emerg. Sel. Top. Circuits Syst. 4 (2) (2014) 230–241, doi:10.1109/JETCAS.2014.2315881.

[20] P. Meloni, G. Tuveri, D. Pani, L. Raffo, F. Palumbo, Exploring custom heterogeneous mpsocs for real-time neural signal decoding, in: E.E.E. Chips, S. design Initiative (Eds.), Design and Architectures for Signal and Image Processing (DASIP), 2015 Conference on, DASIP, ECSI – European Electronic Chips and Systems design Initiative, 2015, pp. 1–8, doi:10.1109/DASIP.2015.7367243.

[21] P. Meloni, C. Rubattu, G. Tuveri, F. Palumbo, D. Pani, L. Raffo, MPSoCs for real–time neural signal decoding: a low-power ASIP-based implementation, Microprocess. Microsyst. (2016).

[22] D. Purves, G. Augustine, D.e.a. Fitzpatrick, Neuroscience, 2nd edition, Sinauer Associates, Sunderland (MA), 2001.

[23] M. Djilas, C. Azevedo-Coste, D. Guiraud, K. Yoshida, Spike sorting of muscle spindle afferent nerve activity recorded with thin-film intrafascicular electrodes, Comput. Intell. Neurosci. 2010 (2010) 13. Article ID 836346

[24] A. Kundu, K. Harreby, K. Yoshida, S.T. Boretius, W.T. Jensen, Stimulation selectivity of the "thin-film longitudinal intrafascicular electrode" (tflife) and the "transverse intrafascicular multi-channel electrode" (time) in the large nerve animal model, Neural Syst. Rehabil. Eng., IEEE Trans. 22 (2) (2014) 400–410.

[25] D. Pani, F. Usai, L. Citi, L. Raffo, Real-time processing of tflife neural signals on embedded dsp platforms: a case study, in: Neural Engineering (NER), 2011 5th International IEEE/EMBS Conference on, 2011, pp. 44–47, doi:10.1109/NER.2011.5910485.

[26] L. Jozwiak, M. Lindwer, R. Corvino, P. Meloni, L. Micconi, J. Madsen, E. Diken, D. Gangadharan, R. Jordans, S. Pomata, P. Pop, G. Tuveri, L. Raffo, G. Notarangelo, Asam: automatic architecture synthesis and application mapping, Microprocess. Microsyst. 37 (8, Part C) (2013) 1002–1019. http://dx.doi.org/10.1016/j.micpro.2013.08.006. Special Issue on European Projects in Embedded System Design: {EPESD2012}

[27] Synopsys Inc., Enabling the Design of Multicore SoCs with Application-Specific Processors, [Online]. Available: http://www.synopsys.com/IP/ProcessorIP/asip/Pages/default.aspx.

[28] Cadence Design Systems Inc., Tensilica Customizable Processor IP. [Online]. Available: http://ip.cadence.com/ipportfolio/tensilica-ip

[29] O. Esko, P. Jääskeläinen, P. Huerta, C.S. de La Lama, J. Takala, J.I. Martinez, Customized exposed datapath soft-core design flow with compiler support, in: Proceedings of the 2010 International Conference on Field Programmable Logic and Applications, in: FPL '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 217–222, doi:10.1109/FPL.2010.51.

[30] O. Derin, E. Cannella, G. Tuveri, P. Meloni, T. Stefanov, L. Fiorin, L. Raffo, M. Sami, A system-level approach to adaptivity and fault-tolerance in noc-based mpsocs: the madness project, Microprocess. Microsyst. 37 (6–7) (2013) 515–529, doi:10.1016/j.micpro.2013.07.007.

[31] G. Kahn, The semantics of a simple language for parallel programming, in: J.L. Rosenfeld (Ed.), Information processing, North Holland, Amsterdam, Stockholm, Sweden, 1974, pp. 471–475.

[32] S. Meijer, H. Nikolov, T. Stefanov, Throughput modeling to evaluate process merging transformations in polyhedral process networks, in: Design, Automation Test in Europe Conference Exhibition (DATE), 2010, 2010, pp. 747–752.

[33] T. Kluter, P. Brisk, E. Charbon, P. Ienne, MPSoC Design Using Application-Specific Architecturally Visible Communication, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 183–197. 10.1007/978-3-540-92990-1_15

[34] T.M. Seese, H. Harasaki, G.M. Saidel, C.R. Davies, Characterization of tissue morphology, angiogenesis, and temperature in the adaptive response of muscle tissue to chronic heating., Lab. Invest. 78 (12) (1998) 1553–1562.

[35] Q. LLC, Ql0200i-a, (http://www.quallion.com/new-pdf/QL0200IA.pdf). Original document from Quallion LLC.

Paolo Meloni is currently assistant professor at the Department of Electrical and Electronic Engineering (DIEE) in the University of Cagliari. In October 2007 he received a Ph.D. in Electronic Engineering and Computer Science, presenting the thesis "Design and optimization techniques for VLSI network on chip architectures." His research activity is mainly focused on the development of advanced digital systems, with special emphasis on the application-driven design of multi-core on-chip architectures. He is author of a significant record of international research papers and tutor of many bachelor and master students'thesis in Electronic Engineering. He is teaching the course of Embedded Systems at University of Cagliari and is currently part of the technical board and acting as work-package leader in the research projects ASAM (http://www.asam-project.org) and MADNESS (http://www.madnessproject.org).

Claudio Rubattu received the B.Sc. and M.Sc. degrees in Electronic Engineering from University of Cagliari, Italy, in 2011 and 2015 respectively. He is part of EOLAB since June 2015, when he joined the Department of Electrical and Electronic Engineering of University of Cagliari, as a research assistant. His research interests include embedded system design in bio-medical applications.

Giuseppe Tuveri received the B.Sc. and M.Sc. degrees in Electronic Engineering from University of Cagliari, Italy, in 2006 and 2009 respectively. He is part of EOLAB since March 2010, when he joined the Department of Electrical and Electronic Engineering of University of Cagliari, as a Ph.D. student. His main research interests include embedded operating systems, system adaptivity in embedded platforms, and FPGA-based multiprocessor prototyping.

Danilo Pani received the university degree (Laurea, magna cum laude) in electronic engineering and the Ph.D. degree in electronic and computer engineering from the University of Cagliari, Cagliari, Italy, in 2002 and 2006, respectively. He is nontenure Assistant Professor in Biomedical Engineering at the Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy, since 2011. Current main research topics are embedded real-time processing of peripheral nervous system signals for neuroprostheses, real-time noninvasive fetal ECG extraction, bioinspired integrated architectures for parallel biomedical signal processing, telemedicine systems. He is an author of more than 40 international publications. Dr. Pani is a member of the national Medical Informatics committee of UNINFO. He is PI of the Regional Project L.R.7/2007 "ELoRAûLow-power realtime processing of neural signals for prosthetic aids" and in involved in many national and European Projects in the field of neuroengineering. He is co-Chair of the CoHeB International Workshop on Collaboration Technologies and Systems in Healthcare and Biomedical Fields.

Luigi Raffo (M.Sc. Electronic Engineering (magna cum laude) in 1989, Ph.D. in Electronics and Computer Science in 1994, University of Genova, Italy) is full professor of Electronics at the Dept. Electrical and Electronic Engineering of the University of Cagliari, Italy. He is a teacher of electronics and system design courses. His main research field is the design of digital/analog devices and systems. In this field he has authored more than 80 international publications, and patents. He has been coordinator of EU, Italian Research Ministry, Italian Space Agency, industrial projects.

Francesca Palumbo is currently an assistant professor at PolComIng Department of the University of Sassari, within the Information Engineering unit. She received her "laurea degree" in Electronic Engineering in 2005 at the University of Cagliari, the Master Advanced in Embedded System Design in 2006 at the Advanced Learning and Research Institute of the University of Lugano and the Ph.D. in Electronic at the University of Cagliari. Her research focus is related to reconfigurable system design and development of code generation tools for advanced reconfigurable hardware architectures. In the field of dataflow-based programming and dataflow-based tools she has received a Best Paper Award at the Conference on Design and Architectures for Signal and Image Processing 2011 for her work entitled "The Multi-Dataflow Composer tool: A runtime reconfigurable HDL platform composer".