Parallel Wave Digital Filter Implementations of Audio Circuits with Multiple Nonlinearities

RICCARDO GIAMPICCOLO,* ANTONINO NATOLI, ALBERTO BERNARDINI, AND AUGUSTO SARTI
(riccardo.giampiccolo@polimi.it) (antonino1.natoli@mail.polimi.it) (alberto.bernardini@polimi.it) (augusto.sarti@polimi.it)

Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)
Politecnico di Milano, Piazza L. Da Vinci 32, 20133 Milano, Italy

Modern audio systems and musical effects feature multi-core processing units. The development of parallel audio processing algorithms capable of exploiting the architecture of such hardware is thus in order. In this paper, we present a parallel version of the Hierarchical Scattering Iterative Method (HSIM), a technique based on Wave Digital Filter principles recently proposed for the emulation of multiphysics audio circuits containing multiple nonlinear one-ports and nonlinear transformers. HSIM operates in a modular fashion, and it is characterized by a high number of embarrassingly parallelizable operations, making it a good candidate for parallel execution. After analyzing HSIM from the parallel computing perspective, we propose three different strategies for the distribution of HSIM workload among threads of execution, and we show how to compute the maximum achievable speedup. The emulation of a possible output stage of a vacuum-tube guitar amplifier is considered, and a performance comparison between parallel and serial implementations of HSIM is presented, pointing out a speedup of nearly 30%. The proposed method proves thus to be promising for Virtual Analog modeling applications, leading the way towards the parallel digital emulation of increasingly complex audio circuits.

0 INTRODUCTION

In recent years, with the advent of processors characterized by enhanced computing capabilities, Digital Audio Effects (DAFx) have become a standard for music production [1]. In particular, Digital Audio Workstations (DAWs) and Virtual Studio Technologies (VSTs) represent a major turning point in the market, since they own the potential to overcome the issues related to both physical room and excessive cost of analog equipment. However, many audio professionals argue that common DAFx lack warmth and emotional content with respect to their analog counterparts, and thus they still prefer to rely on analog devices. Among the different typologies of DAFx, Virtual Analog (VA) modeling techniques [2] are gaining resounding success among musicians since they pursue the digital emulation of existing analog audio gear, sparing them the drawbacks of buying heavy and bulky analog equipment.

VA modeling techniques can be divided into two main categories: (i) *black-box* methods that aim at solving a system identification task starting from input/output data, using, for example, neural networks [3], Volterra series [4], dynamic convolution [5], or block-oriented Wiener-Hammerstein models [6]; (ii) *white-box* methods that aim at emulating the reference circuit by solving the corresponding system of ordinary differential equations (or differential algebraic equations), using, for example, the State-space approach [7], the Port-Hamiltonian approach [8], or Wave Digital Filters (WDFs) [9]. White-box methods are, in general, more accurate than black-box methods [10]. However, if the reference circuit features a high number of nonlinear elements, white-box techniques might be characterized by a high computational cost which prevents them from running in real-time. Hence, the development of efficient white-box methodologies is in order.

The latest findings in the WDF literature prove useful in achieving this goal. WDFs were introduced by A. Fettweis in the 1970s for deriving digital implementations of reference passive analog filters [9]. WDFs are realized describing the reference analog circuit as an interconnection of WD blocks. This is accomplished by substituting Kirchhoff port variables (port voltages and port currents) with linear combinations of *wave variables* (incident waves and reflected waves) including a free parameter per port called *port resistance*. Circuit elements and connection networks are described as one-port or multi-port blocks characterized by input/output scattering relations, which express the reflected waves as functions of the incident waves. Moreover, the descriptions of elements and topological connec-

*Corresponding author.

tion networks are separately managed, enhancing the modularity of the circuit representation. Employing stable discretization methods (e.g., trapezoidal rule), circuits with up to one nonlinear element and characterized by an explicit scattering relation can be implemented in the WD domain without the need of any iterative solver [11, 12]. In this case, in fact, the implicit relations governing wave variables, called *delay-free-loops* in the literature on WDFs, can be eliminated by properly setting the free parameters [9, 13]. This is not valid, in general, in mainstream circuit simulation methods implemented in the Kirchhoff domain, e.g., the Modified Nodal Analysis methods used in Spice-like software [14].

If multiple nonlinear elements are present in the circuit, however, the use of iterative solvers is required also in the WD domain [15, 16]. As an example, in [17] a hybrid iterative scheme, which discusses as particular cases fixed-point, Newton-Raphson (NR), and Newton-Jacobi methods, is presented, and the flexibility of the WD approach is highlighted. An alternative NR approach is proposed in [18], where the multidimensional WDF formalism [13] is integrated with the concept of automatic differentiation in order to accommodate multiple nonlinearities. However, there were no in-depth studies on the optimal selection of the free parameters in the WD domain (port resistances) until the publication of [19], where a WD fixed-point method, called Scattering Iterative Method (SIM), was proposed. Introduced for the analysis of large photovoltaic arrays under partial shading conditions, SIM is able to solve generic circuits containing multiple nonlinear one-ports using independent one-dimensional solvers. SIM was extended for the discrete-time domain emulation of diode-based ring modulators in the context of VA modeling [12, 20], and then employed for the emulation of many other audio circuits [21–24]. It is worth adding that, in the WD domain, a proper choice of the free parameters has a direct effect not only on the speed of convergence of fixed-point methods, but also of WD Newton-Raphson methods, as discussed in [25].

In [26, 27], a generalized version of SIM, called Hierarchical SIM (HSIM), is provided, whose enhanced modularity is exploited for the multiphysics simulation of nonlinear transformers. In fact, contrary to SIM that solves WD structures characterized by one single topological junction, HSIM is able to solve WD structures characterized by an arbitrary number of junctions (topological or Magnetic/Electric in the case of multiphysics modeling of nonlinear transformers [26]). This opens up new possibilities for handling the circuit topology, which can be arranged following different strategies.

Apart from being efficient and robust, both SIM and HSIM are characterized by operations that are *embarrassingly parallelizable* [19]; this is a unique property that, until now, has never been exploited for enhancing their efficiency. A workload is referred to as *embarrassingly parallelizable* if it can be subdivided into tasks that can be concurrently executed [28]. As a consequence, SIM and HSIM allow us to distribute part of their operations on different cores or threads, which do not need to exchange data with one another, leading the way towards the real-time implementation of increasingly complex audio circuits. From this point of view, SIM and HSIM are able to exploit the new possibilities offered by modern audio systems that are often characterized by multiple Digital Signal Processors (DSPs) or Field Programmable Gate Arrays (FPGAs) and allow the parallel execution of code.

In the light of these considerations, with the aim of developing a new class of WD algorithms suitable to fully exploit the capability of modern audio processing systems, we analyze and develop a parallel version of HSIM (parHSIM). HSIM is chosen as main framework since, thanks to its enhanced modularity w.r.t. SIM, it offers more options as far as the distribution of tasks among cores/threads is concerned. In particular, we show that HSIM can be parallelized following three main strategies, characterized in turn by different scheduling policies. Moreover, a precise method for assessing the number of threads is discussed. Finally, the proposed methodology is applied to the simulation of a possible output stage of a vacuum-tube guitar amplifier (with a nonlinear transformer), implementing both HSIM and parHSIM (i.e., the parallelized version of HSIM) in C++ language. The accuracy of the WD simulations is verified via a comparison with MathWorks Simscape. Finally, the speedup obtained by parHSIM w.r.t. HSIM is highlighted, pointing out the efficiency of the new approach.

The paper is structured as follows. Section 1 provides background knowledge on the theory of parallel computing. The design procedure of WDFs is then presented in Section 2, whereas HSIM and its parallel implementation parHSIM are discussed in Section 3 and in Section 4, respectively. After presenting in Section 5 an example of application of the proposed parallel algorithm, concluding remarks are drawn in Section 6.

1 BACKGROUND ON PARALLEL COMPUTING

Parallel computing theory aims at designing and analyzing parallel algorithms, deriving optimal scheduling policies (both offline and online), and predicting the *speedup*, which is defined as the increment in execution speed introduced by the parallel version of an algorithm w.r.t. its sequential version [28]. In the literature, several mathematical models of speedup have been proposed [29–33]. In the following, assuming the number of threads to be fixed in time, we present some theoretical tools useful to predict the performance of parallel algorithms. Moreover, the terms *cores*, *processes*, *threads*, or *workers* will be used interchangeably.

1.1 Amdahl's Law

Amdahl's law is the simplest model describing the speedup achievable by a parallel program [29]. It is based on the following assumptions:

- the workload can be divided into a fully sequential part and a fully parallel part. The latter is assumed to be *in-*

finitely parallelizable, meaning that it can be executed on an infinite number of cores;
- the use of multiple CPU cores leads to an improvement in the execution time of the parallel part and not of the sequential part.

Naming N_c the number of cores, the speedup $S(N_c)$ is expressed as

$$S(N_c) = \frac{T_1}{T_{Nc}} = \frac{T_s + T_p}{T_s + \frac{T_p}{N_c}}, \quad (1)$$

where T_1 and T_{Nc} are the execution times of the whole workload considering only one core and N_c cores, respectively, T_s is the execution time of the sequential part, and T_p is the execution time of the parallel part. Once we have defined the *p*-fraction $f \in [0,1]$ as $f = T_p/(T_s + T_p)$ and the *s*-fraction $s \in [0,1]$ as $s = 1 - f$, we can rewrite Eq. (1) as follows [29]

$$S(N_c) = \frac{1}{(1-f) + \frac{f}{N_c}}, \quad (2)$$

whose upper bound is given by

$$S_{max} = \lim_{N_c \to \infty} S(N_c) = \frac{1}{s}. \quad (3)$$

According to Amdahl's law, the maximum speedup is thus always bounded by its *s*-fraction. It is worth noting that if $f = 1$ (i.e., the execution time of the serial part is considered to be negligible) the speedup reduces to $S(N_c) = N_c$, i.e., the number of cores. In this case, the workload is said to be *embarrassingly parallelizable*, since it can be completely divided among cores.

Although very simple, Amdahl's law does not describe a reliable speedup, which is limited, instead, by many other factors. Gustafson's law, for example, [30] does not consider a fixed workload but a scalable one: the higher N_c, the higher the number of tasks that can be assigned at runtime. Communication and synchronization overheads are, instead, taken into account by the Li-Malek model [31] or the Sun-Ni model [32]. However, from a practical stand point, it is worth considering the concept of *parallelism* which will be presented in the next subsection.

1.2 Parallelism and Direct Acyclic Graph

The following discussion is related to workloads characterized by both serial and parallelizable tasks. In order to overcome the limitations introduced by Amdahl's law, the concept of *parallelism* can be considered [32]. The parallelism p of a workload is defined as the ratio between the execution time T_1 with $N_c = 1$ and the execution time T_∞ with $N_c = \infty$, i.e., $p = T_1/T_\infty$. It follows that the quantity p represents the maximum speedup achievable by a parallel implementation, as we will discuss further ahead. Moreover, p represents also a good starting point for setting the number of threads, since, given that the maximum speedup is bounded to p, setting $N_c > p$ does not necessarily imply further advantages [34]. In fact, such a choice may cause *starvation*, slowing down the overall execution [28].

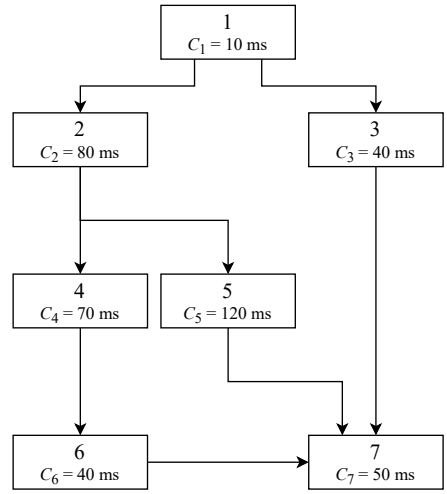


Fig. 1: Example of Direct Acyclic Graph (DAG). The *work* T_1 is equal to the sum of the costs $(C_1,\ldots,C_7)$ of all the strands, i.e., $T_1 = 410$ ms, whereas the *span* T_∞ is the most expensive path, i.e., the path with nodes 1,2,5,7, and is $T_\infty = 260$ ms.

Let us consider a workload composed of a fixed number of atomic instructions. A practical way to compute p is to consider the workload in the form of a *Direct Acyclic Graph* (DAG) [35,36]. A DAG is a computation task graph arranged as a tree structure, where all the nodes (also called *strands*) represent the atomic instructions of the workload, and the edges connecting the nodes denote the order of execution [35]. Each strand is associated to a cost in terms of execution time; moreover, if a set of instructions does not contain any parallel code, it is considered on a par with atomic instructions, and, therefore, it is represented as a single node [35].

Given a DAG, we can define [36]:

- the *work* as the time T_1 needed to execute the whole workload. It can be seen as the time required by a serial program to complete the same workload;
- the *span* as the most expensive path, from the first strand to the last strand, in terms of execution time. As a particular case, if each strand has unitary cost, the span is also the *critical path*, and it is equal to its number of nodes. Moreover, the span is interpreted as the time required to execute the whole workload on a machine with infinite number of cores, and, thus, it is denoted with T_∞.

Figure 1 presents an example of DAG together with the relative *work* and *span*. In order to derive an upper bound for the speedup $S(N_c)$, we consider the *span law*, which states that a machine with N_c cores cannot complete the workload before a machine with infinite number of cores [35]. In formula, the span law is written as

$$T_{Nc} \geq T_\infty, \quad (4)$$

which, combined with the definition of parallelism, leads to the following expression

$$S(N_c) = \frac{T_1}{T_{Nc}} \leq \frac{T_1}{T_\infty} = p. \quad (5)$$

The parallelism p represents a more reliable upper bound for $S(N_c)$ than Eq. (3), since it is not based on the assumption of *infinitely parallelizable* operations. Moreover, in the case of fixed workload, work and span can be easily computed (see Fig. 1), making p a very useful metric for the design of parallel algorithms. Finally, it is worth pointing out that, in real conditions, the equality in Eq. (5) cannot be reached due to communication and synchronization overheads.

1.3 Scheduling

The *scheduling* consists in mapping jobs/tasks/instructions into available cores, or, more generally, into available workers [36]. Given a workload represented via a DAG, a scheduler generates a timing diagram in the form of a *Gantt chart* [36] and can either operate *online* or *offline*. In the latter case, the entire schedule is determined before run-time; in the former case, it is dynamically changed at run-time. A requirement for offline scheduling is, however, to know the whole DAG *a priori*.

When addressing scheduling, *load balancing* should be guaranteed, meaning that the workload should be assigned to threads such that each of them is characterized by a similar computational load. Moreover, offline scheduling policies implement *static threading*, i.e., each thread executes its tasks independently of the others. Offline scheduling enables a higher control at the hands of the programmer who can precisely assign tasks to threads, but, on the other hand, may be characterized by lower performance w.r.t. online scheduling.

The HSIM algorithm discussed in this manuscript is indeed characterized by a known DAG, and, therefore, we opted for considering only offline scheduling policies in the following. In the next Section, we will present general principles for modeling reference audio circuits in the WD domain. The theory on parallel computing will be then applied in Section 4 for deriving a parallel version of HSIM.

2 DESIGN OF WDFS

In the WD domain, elements and topological connection networks (junctions) are realized through input-output blocks described by scattering equations. Each port voltage v and port current i is substituted by a linear combination of waves, whose most used definition is [9, 13]

$$a = v + Zi, \qquad b = v - Zi, \quad (6)$$

where a and b are the incident and reflected waves, respectively, whereas Z is a port free parameters called *port resistance*. A proper choice of free parameters is fundamental for the efficient solution of WD structures [9, 13]. It is worth adding that when two WD blocks are connected together, the wave incident to the port of one is set equal to the wave reflected by the port of the other and vice versa; moreover, the two port resistances must match as well.

2.1 Modeling Topological Junctions

Naming $\mathbf{v} = [v_1, \ldots, v_N]^T$ the vector of port voltages and $\mathbf{i} = [i_1, \ldots, i_N]^T$ the vector of port currents, wire connections among elements can be described by the sets of Kirchhoff equations [37, 38]

$$\mathbf{v} = \mathbf{Q}^T \mathbf{v}_t, \quad \mathbf{i} = \mathbf{B}^T \mathbf{i}_l, \quad (7)$$

where $\mathbf{v}_t$ is the $q \times 1$ vector of independent port voltages, $\mathbf{i}_l$ is the $l \times 1$ vector of independent port currents, whereas $\mathbf{Q}$ and $\mathbf{B}$ are the $q \times N$ fundamental cut-set and the $l \times N$ fundamental loop matrices [19, 38], respectively. In the WD domain, wire connections turn into one or more reciprocal lossless topological junctions that can be described by means of the scattering relation $\mathbf{a} = \mathbf{S}\mathbf{b}$, where here $\mathbf{b} = [b_1, \ldots, b_N]^T$ is the vector of waves incident to the junction (and reflected by the WD blocks connected to the junction), while $\mathbf{a} = [a_1, \ldots, a_N]^T$ is the vector of waves reflected by the junction (and incident to the WD blocks connected to the junction), while the $N \times N$ scattering matrix $\mathbf{S}$ can be computed employing one of the two equivalent formulas [19, 37, 38]

$$\mathbf{S} = 2\mathbf{Q}^T(\mathbf{Q}\mathbf{Z}^{-1}\mathbf{Q}^T)^{-1}\mathbf{Q}\mathbf{Z}^{-1} - \mathbf{I}, \quad (8)$$

$$\mathbf{S} = \mathbf{I} - 2\mathbf{Z}\mathbf{B}^T(\mathbf{B}\mathbf{Z}\mathbf{B}^T)^{-1}\mathbf{B}, \quad (9)$$

where $\mathbf{I}$ is a properly sized identity matrix and $\mathbf{Z} = \mathrm{diag}[Z_1, \ldots, Z_N]^T$ is a diagonal matrix having port resistances as non-zero entries.

2.2 Modeling Magnetic/Electric Junctions

Starting from the magneto-motive force/voltage analogy [39], the coupling between magnetic and electric domains is realized by means of the Magnetic/Electric (ME) junction shown in Fig. 2(a), whose constitutive equations in the continuous-time domain are

$$\begin{cases} v(t) = -n_t \frac{d\phi(t)}{dt} \\ \mathscr{F}(t) = n_t i(t) \end{cases}, \quad (10)$$

where $\mathscr{F}$ is the magneto-motive force (m.m.f.), ϕ is the magnetic flux, and n_t is the number of winding turns. We call "magnetic port" the port facing the magnetic subcircuit, and "electric port" the port facing the electrical subcircuit. Employing Backward Euler for discretizing the time derivative in Eq. (10) and considering the mapping

$$\begin{aligned} v[k] &= \tfrac{a_e[k]+b_e[k]}{2}, & i[k] &= \tfrac{a_e[k]-b_e[k]}{2Z_e[k]}, \\ \mathscr{F}[k] &= \tfrac{a_m[k]+b_m[k]}{2}, & \phi[k] &= \tfrac{a_m[k]-b_m[k]}{2Z_m[k]}, \end{aligned} \quad (11)$$

the ME junction can be described in the WD domain (see Fig. 2(b)) by the following system of equations

$$\begin{bmatrix} b_e[k] \\ b_m[k] \end{bmatrix} = \mathbf{S}_{ME} \begin{bmatrix} a_e[k] \\ a_m[k] \end{bmatrix} + \mathbf{M}_{ME} \begin{bmatrix} a_m[k-1] \\ b_m[k-1] \end{bmatrix}, \quad (12)$$

where k is the sample index, a_m and b_m are the waves incident to and reflected from the magnetic port, a_e and b_e are the waves incident to and reflected from the electric port,

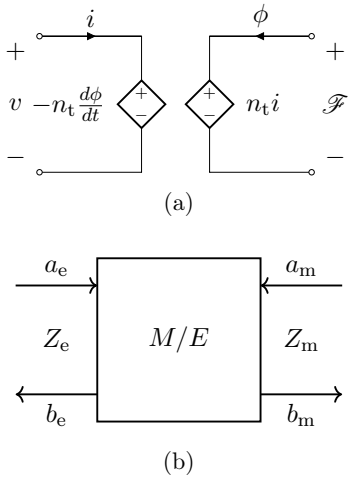


Fig. 2: (a) Magnetic/Electric (ME) junction in the continuous-time domain; (b) ME junction in the WD domain.

and

$$\mathbf{S}_{\mathrm{ME}} = \begin{bmatrix} -\frac{\frac{2Z_\mathrm{e}[k]}{n_\mathrm{t}}-\beta[k]}{\beta[k]} & \frac{1}{2}\left[\frac{\left(\frac{2Z_\mathrm{e}[k]}{n_\mathrm{t}}-\beta[k]\right)^2}{\beta[k]} - \beta[k]\right] \\ \frac{2}{\beta[k]} & -\frac{\frac{2Z_\mathrm{e}[k]}{n_\mathrm{t}}-\beta[k]}{\beta[k]} \end{bmatrix}, \quad (13)$$

$$\mathbf{M}_{\mathrm{ME}} = \begin{bmatrix} \frac{Z_\mathrm{e}[k]}{T_s Z_\mathrm{m}[k-1]}\frac{1}{\beta[k]} & -\frac{Z_\mathrm{e}[k]}{T_s Z_\mathrm{m}[k-1]}\frac{1}{\beta[k]} \\ -\frac{n_\mathrm{t}}{T_s Z_\mathrm{m}[k-1]}\frac{1}{\beta[k]} & \frac{n_\mathrm{t}}{T_s Z_\mathrm{m}[k-1]}\frac{1}{\beta[k]} \end{bmatrix}. \quad (14)$$

Moreover, in Eqs. (11), (13), and (14) Z_m and Z_e are the free parameters of the magnetic and the electric ports, respectively, while $\beta[k] = \frac{Z_\mathrm{e}[k]}{n_\mathrm{t}} + \frac{n_\mathrm{t}}{T_s Z_\mathrm{m}[k]}$.

As shown in [26], since the two diagonal entries of $\mathbf{S}_{\mathrm{ME}}$ are identical, it is possible to remove, at the same time, the implicit relations involving the two ports. This can be achieved by setting either

$$Z_\mathrm{e}[k] = \frac{n_\mathrm{t}^2}{T_s Z_\mathrm{m}[k]}, \quad (15)$$

or, equivalently,

$$Z_\mathrm{m}[k] = \frac{n_\mathrm{t}^2}{T_s Z_\mathrm{e}[k]}. \quad (16)$$

Finally, it is worth mentioning that Backward Euler was selected in [26] as discretization method for its stability properties, but other implementations of the ME junction can be obtained using the trapezoidal rule or Backward Discretization Formulas (BDFs) as discussed in [27].

2.3 Modeling Linear Elements

A wide class of linear one-port circuit elements, including resistive sources, resistors, capacitors, and inductors [12], can be represented in the discrete-time domain by the following Thévenin equivalent

$$v[k] = R_\mathrm{g}[k] i[k] + V_\mathrm{g}[k], \quad (17)$$

where k is the sampling index, $R_\mathrm{g}[k]$ is the Thévenin equivalent resistance, and $V_\mathrm{g}[k]$ is the Thévenin equivalent voltage. Applying to (17) the transformation of variables (6), after some algebra, we can express the wave reflected by the element $b[k]$ as a function of the wave incident to the element $a[k]$ as

$$b[k] = \frac{R_\mathrm{g}[k] - Z[k]}{R_\mathrm{g}[k] + Z[k]} a[k] + \frac{2Z[k]}{R_\mathrm{g}[k] + Z[k]} V_\mathrm{g}[k]. \quad (18)$$

It is possibile to eliminate the instantaneous dependence of $b[k]$ on $a[k]$ by setting the port resistance $Z[k] = R_\mathrm{g}[k]$ [12]. In this case, the element is said to be *adapted* and the wave $b[k]$ can be easily assessed since no delay-free-loop involves the WD block [9, 13].

2.4 Modeling Nonlinear Elements

Contrary to linear circuit elements, nonlinear one-ports cannot be adapted [19, 25, 40–42]. Nonetheless, their port resistance can be set in such a way that the reflection coefficient relating $b[k]$ to $a[k]$, defined here as the derivative $f'(a[k])$ with respect to $a[k]$ of $b[k] = f(a[k])$, is kept small [25]. This can be achieved by setting $Z[k] = v'(i[k-1])$, where $v(i)$ represents the nonlinear characteristic of the element in the Kirchhoff domain, and $v'(i[k-1])$ is the tangent slope at the previous operating point [19, 25]. A nonlinear element can be modeled in the WD domain making use of different methods, such as scalar Newton-Raphson solvers [19], Canonical Piecewise-Linear (CPWL) representations [43], Lambert or Wright ω functions [44, 45].

The nonlinear elements considered in this work are magnetic reluctances that can be efficiently implemented in the WD domain using CPWL functions [46], also enabling the possibility of deriving WD models of commercial transformers [26, 27]. A CPWL representation of the wave mapping $b = f(a)$ can be written as

$$b = \lambda_0 + \lambda_1 a + \sum_{j=1}^{J} \eta_j |a - a_j|. \quad (19)$$

Equation (19) is valid for functions with no jump discontinuities, such as the nonlinear $B-H$ curves characterizing magnetic materials (where B is the magnetic flux density and H is the magnetic field). Moreover, in Eq. (19), $J+1$ is the number of segments of the CPWL curve, and J is the number of vertexes with coordinates (a_j, b_j). The remaining parameters are, instead, computed as follows

$$\begin{aligned} &\lambda_1 = 0.5(m_0 + m_J), \\ &\eta_j = 0.5(m_j - m_{j-1}), \qquad j = 1, \ldots, J, \\ &\lambda_0 = f(0) - \textstyle\sum_{j=1}^{J} \eta_j |a_j|, \end{aligned} \quad (20)$$

where m_j is the slope of the jth segment defined as

$$m_j = \frac{b_{j+1} - b_j}{a_{j+1} - a_j}, \qquad j = 1, \ldots, J-1. \quad (21)$$

A more in-depth discussion on the use of CPWL functions for the modeling of nonlinear magnetic reluctances in the WD domain can be found in [26, 27].

3 Hierarchical Scattering Iterative Method

The Hierarchical Scattering Iterative Method (HSIM) was proposed in [26] for the multiphysics simulation of nonlinear transformers in the context of VA modeling, and, later on, it was extended in [27] for the simulation of electromagnetic circuits containing nonlinear transformers characterized by more complex topologies. HSIM inherits the characteristics of the Scattering Iterative Method (SIM) (proposed in [19]) but with an enhanced modularity, being capable of solving WD structures characterized, at the same time, by multiple nonlinearities and multiple junctions. Moreover, as far as the emulation of electromagnetic circuits containing nonlinear transformers is concerned, HSIM is able to maintain the modularity of the two physical domains, i.e., the electric domain and the magnetic domain, opening up different possibilities for modeling the topology. In addition, the method has proved to be more efficient than standard simulation software [27], such as LTspice or MathWorks Simscape, when dealing with large and complex circuits, and, therefore, it is promising for VA applications.

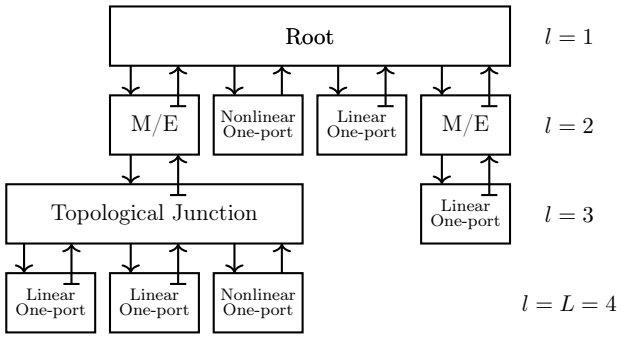


Fig. 3: Example of a WD structure organized as a rooted connection tree. The structure is composed of four levels. The T-shaped stubs indicate port adaptation.

3.1 HSIM Algorithm

Let us consider a WD structure composed of multiple scattering junctions. It is possible to arrange it in the form of a rooted connection tree made of many levels, which go from level $l = 1$ (the root of the tree) to level $l = L$ consisting only of leaves (one-port circuit elements). The topological junction with the biggest scattering matrix is selected to be the *root* of the tree. Although the algorithm can be employed for the simulation of circuits where no ME junction is needed (i.e., no magnetic equivalent circuit is present), hereafter we present it in the context of multiphysics electromagnetic simulation. The WD connection tree entails as many *nodes* as the number of the other topological junctions and ME junctions, and as many *leaves* as the number of the linear and nonlinear one-port elements. An example of the sort is shown in Fig. 3, where four root subtrees are present. A root subtree is defined as the tree composed of a node/leaf branching out from the root and all of the possible node descendants. Looking at Fig. 3, two root subtrees are degenerate since composed of just one leaf (linear/nonlinear one-port), one is composed of two nodes (ME and Topological Junction) and three leaves (two linear one-ports and a nonlinear one-port), whereas the last is composed of a node (ME junction) and a leaf (linear one-port).

In the following, we employ different subscripts to wholly identify the leaves (one-ports) and the nodes (junctions) of the connection tree. In order to denote the uth leaf/node of the lth level, scalar wave variables referred to leaves, as well as vector wave variables and scattering matrices referred to nodes, have a subscript in the form l,u. The subscript l,u,n, instead, indicates a variable referred to the nth port of the uth node in level l. Vectors with no subscripts are a concatenation of the vectors of port variables referred to the whole circuit. Moreover, the total number of ports of the uth node in level l is indicated with $N_{l,u}$. Finally, it is important to underline that the waves incident to a certain level are the waves reflected by an adjacent level and vice versa.

The algorithm is composed of the following six stages performed at each sampling step k:

1. *Initialization and Update*: the port resistance $Z_{l,u}[k]$ of each one-port element is set as close as possible to $v'_{l,n}(i_{l,n}[k])$ (or $\mathscr{F}'_{l,n}(\phi_{l,n}[k])$). For linear one-ports, this corresponds to setting $Z_{l,u}[k] = R_{\mathrm{g}}[k]$ [12]. For nonlinear elements, instead, it can only be estimated, e.g., by considering $Z_{l,u}[k] = v'_{l,n}(i_{l,n}[k-1])$ (or $Z_{l,u}[k] = \mathscr{F}'_{l,n}(\phi_{l,n}[k-1])$) [12, 19, 27]. As far as topological junctions are concerned, the port facing the previous level is made *reflection-free* employing the corresponding adaptation condition. For ME junctions, the same action is achieved employing either Eq. (15) or Eq. (16). Moreover, whenever the port resistances are changed, the matrices of the WD junctions need to be updated.
2. *Leaves Scattering Stage*: the waves $b_{l,u}[k]$ reflected by one-port elements (leaves) in level l are computed according to their WD scattering equations. In particular, for adapted linear one-ports we use Eq. (18), while for nonlinear elements we consider a nonlinear mapping in the form (see Eq. (19))
$$b^{(\gamma)}_{l,u}[k] = f_{l,u}\big(a^{(\gamma-1)}_{l,u}[k]\big)\,, \tag{22}$$
where γ is the HSIM iteration index.
3. *Forward Scattering Stage*: the waves $b_{l,u,n}[k]$ reflected by junctions in level l (starting from level $L-1$) are computed and propagated to level $l-1$. As far as topological junctions are concerned, we employ
$$b^{(\gamma)}_{l,u,n}[k] = \mathbf{s}_{l,u,n}[k]\mathbf{a}^{(\gamma)}_{l,u}[k]\,, \tag{23}$$
where $\mathbf{s}_{l,u,n}[k]$ is the row vector of the scattering matrix $\mathbf{S}_{l,u}[k]$ corresponding to the nth port facing level $l-1$. The nth entry of $\mathbf{s}_{l,u,n}[k]$ is zeroed according to the junction adaptation condition for port n. The wave reflected by a ME junction is, instead, scalar and is computed using one of the two relations in Eq. (12), depend-

ing on which port (electric or magnetic) is connected to level $l-1$.

4. *Root Scattering Stage*: once level $l=1$ is reached, the vector of waves reflected by the root $\mathbf{b}_{1,1}[k]$ is computed as

$$\mathbf{b}_{1,1}^{(\gamma)}[k] = \mathbf{S}_{1,1}[k]\mathbf{a}_{1,1}^{(\gamma)}[k]\,, \tag{24}$$

where $\mathbf{a}_{1,1}[k]$ is the vector of waves incident to the root, and $\mathbf{S}_{1,1}[k]$ is the root scattering matrix.

5. *Backward Scattering Stage*: the $N_{l,u}-1$ waves reflected by the junctions in level l are computed and propagated towards level $l+1$. The computation starts at level $l=2$ and stops when level $L-1$ is reached. If the uth WD block of the lth level is a topological junction we have that

$$\mathbf{b}_{l,u}^{(\gamma)}[k] = \mathbf{S}_{l,u}[k]\mathbf{a}_{l,u}^{(\gamma)}[k]\,, \tag{25}$$

whereas if the WD block is a ME junction, $b_{l,u}[k]$ is scalar and it is computed according to one of the two scattering relations in Eq. (12), depending on which domain (magnetic or electric) the port facing level $l+1$ belongs to.

6. *Convergence Check*: Leaves, Forward, Root, and Backward Scattering Stages are iterated until the inequality

$$||\boldsymbol{\upsilon}^{(\gamma)}[k] - \boldsymbol{\upsilon}^{(\gamma-1)}[k]||_2 < \varepsilon_{\text{HSIM}}\,, \tag{26}$$

holds true, where $\boldsymbol{\upsilon}^{(\gamma)}[k]$ is the vector collecting all the port voltages and port magneto-motive forces of the circuit at iteration γ, while $\varepsilon_{\text{HSIM}}$ is a small threshold (e.g., $\varepsilon_{\text{HSIM}} = 10^{-5}$).

It is worth pointing out that HSIM computational flow follows a *forward-backward* approach, where Leaves Scattering Stage and Forward Scattering Stage belong to the *forward scan*, and Backward Scattering Stage belong to the *backward scan*. Moreover, some of the scattering operations previously described can be arranged following different orders. In the next Section, we will further investigate such concepts. Finally, more insights on the implementation of HSIM can be found in [27].

4 Parallel HSIM

Figure 4 represents a possible generic DAG of the whole HSIM algorithm at sample k, while a zoom on a possible generic DAG of the γth HSIM iteration is shown in Fig. 5. From Figures 4 and 5 it is clear that a certain sequentiality, determined by specific *synchronization barriers*, must be observed. During the forward scan, considering each root subtree in a separate fashion, the scattering operations of a WD block in level $l-1$ can be executed only after those in level l directly connected to that block; conversely, during the backward scan, the scattering operations of a WD block in level $l+1$ can be executed only after those in level l directly connected to it. It is worth pointing out that, even though in Figures 4 and 5 each root subtree is characterized by L levels, some of them might feature a number of effective levels lower than L, meaning that the last ones are empty (e.g., see Fig. 3).

As far as the "Initialization and Update Stage" is concerned, we decide to update the free parameters of the elements belonging to level l in a parallel fashion, but they can be also updated sequentially since such an operation is done only once per sample and it is characterized by a negligible cost. Moreover, given that the update of the scattering matrices is the most computationally expensive operation [21], we decide to assign these tasks to different workers.

In the current formalization of the HSIM algorithm, both nonlinear and linear elements are involved in the iterative procedure (see Leaves Scattering Stage in Subsection 3.1). However, given that linear elements are always adapted, their waves do not vary during iteration and, therefore, they can be computed just once per sample k. Hence, we decide to keep these operations outside of the iterative loop. In accordance with what was stated for the SIM algorithm in [19], we can argue that the Leaves Scattering Stage of HSIM is *embarrassingly parallelizable*, since the computation of the reflected waves referred to the elements can be assigned to different threads. However, contrary to SIM, it is not the only stage that is suitable for a parallel implementation. As a matter of fact, considering a general HSIM connection tree, we can think of at least three different strategies for implementing a parallel version of HSIM:

- *Parallelization by Levels*: consists in assigning to different workers the scattering operations to be performed at each level l. Hence, during the forward scan, we split all the scattering operations in level l among the threads. Once the workload of level l is completed, we split the operations in level $l-1$, and so forth. This procedure is repeated until the root (in level $l=1$) is reached. Then, after the Root Scattering Stage has taken place, we apply the same procedure for the backward scan, but levels are solved in inverse order, from level 2 down to level $L-1$. A synchronization barrier must be observed at each level, meaning that all the operations in a certain level l must be completed before solving those in the next level. Considering at least two topological junctions, the total number of synchronization barriers per iteration is thus ≥ 2.
- *Parallelization by Stages*: consists in assigning the workload of each HSIM stage to different workers. Hence, we first split the workload of the Leaves Scattering Stage, then we address the Forward Scattering Stage and so on. A synchronization barrier must be always observed before addressing the next stage. Therefore, the total number of synchronization barriers per iteration is 6.
- *Parallelization by Root Subtrees*: consists in assigning each root subtree to a different worker (as shown in Fig. 4 and Fig. 5). It follows that the Root Scattering Stage and the Convergence Check are the only synchronization barriers, making this strategy the most promising for the parallel implementation of HSIM (the total number of synchronization barriers per iteration is 2).

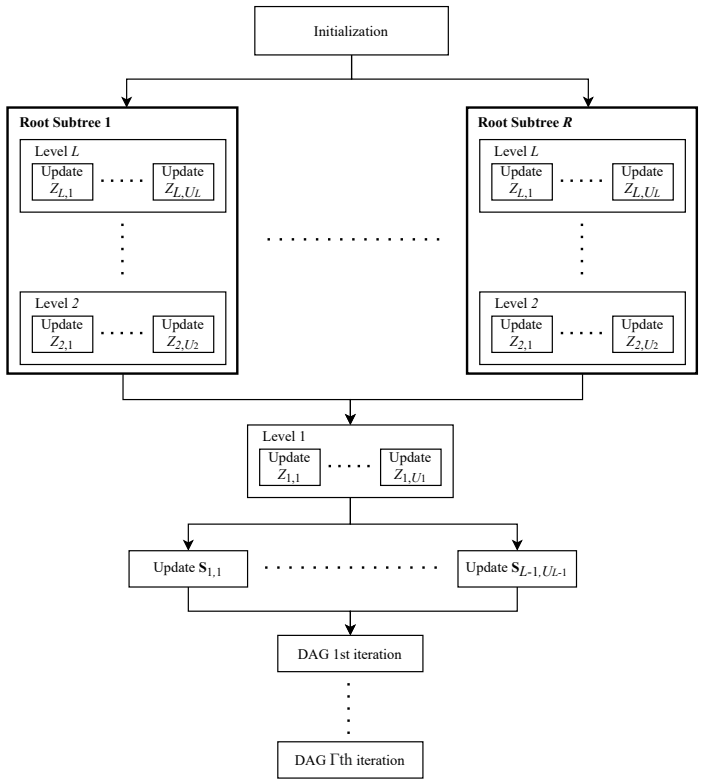


Fig. 4: Parallelization by Root Subtrees: DAG of HSIM at the kth sample, where R is total number of root subtrees, U is the total number of leaves/nodes in level l, and Γ is the total number of iterations referred to sample k. The generic block describing the DAG at iteration γ is shown in Fig. 5.

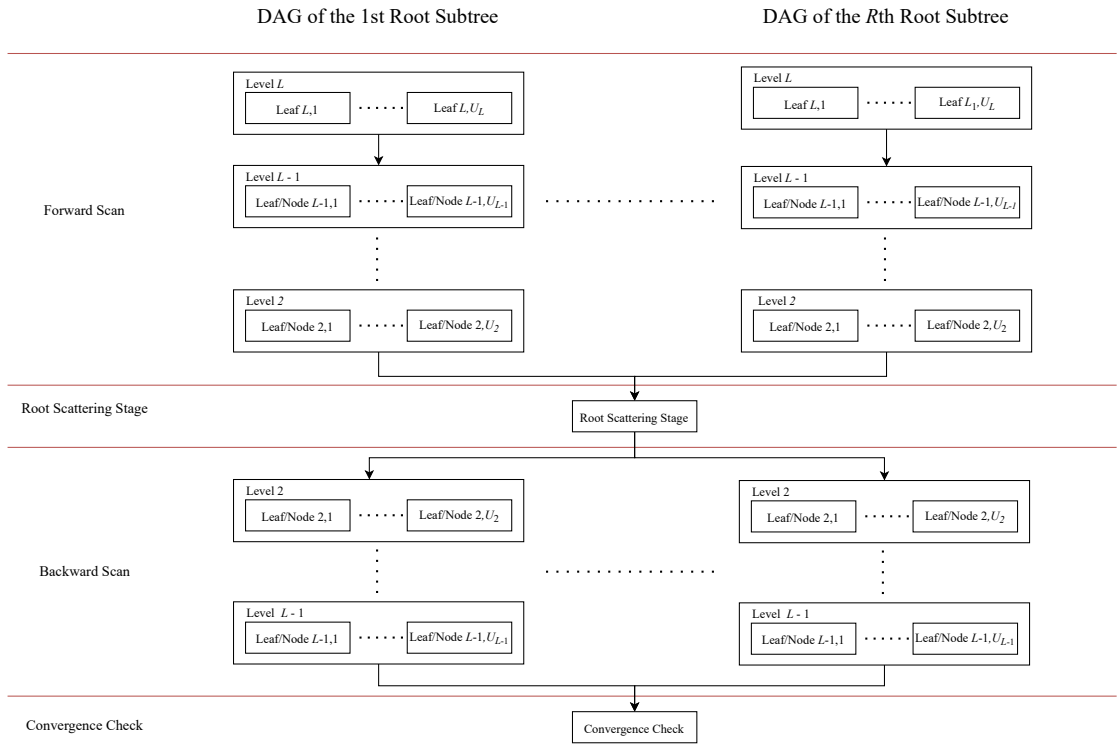


Fig. 5: Parallelization by Root Subtrees: DAG of HSIM at the γth iteration, where R is total number of root subtrees, U is the total number of leaves/nodes in level l, and Γ is the total number of iterations referred to sample k.

It is evident that HSIM levels, stages, and root subtrees are all embarrassingly parallelizable. It is worth stressing that, in order to design an efficient algorithm, *load balancing* must be always satisfied. The "Parallelization by Root Subtrees" not only is characterized by the lowest number of synchronization barriers per iteration, but it offers more flexibility w.r.t. the other strategies as far as the distribution of the computational load among threads is concerned. It is thus the most suited to obtain *load balancing* and, therefore, it has been chosen in the following analysis. Finally, it is worth adding that, although the foregoing discussion refers to HSIM, some of the concepts introduced in this Section can be applied to other types of WD algorithms as long as their WD structure can be arranged in the form of a rooted tree and some of their operations are embarrassingly parallelizable.

4.1 Speedup Analysis

As already pointed out in Section 1, Amdahl's law provides a speedup estimate that is far from the actual workload processing behavior. For this reason, here we employ the concept of parallelism to derive a more reliable maximum speedup achievable by HSIM.

The span T_∞ of a generic HSIM workload can be computed as follows

$$T_\infty = T_{\text{init}} + T_{\text{Z,max}} + T_{\text{S,max}} + T_{\text{fwd,max}} + \\ + T_{\text{root}} + T_{\text{bwd,max}} + T_{\text{check}}\,, \tag{27}$$

where T_{init} is the execution time of the initialization strand, $T_{\text{Z,max}}$ is the highest execution time as for the update of the free parameters $Z_{l,u}$, $T_{\text{S,max}}$ is the highest execution time as for the update of the scattering matrices, $T_{\text{fwd,max}}$ and $T_{\text{bwd,max}}$ are the highest execution times of forward and backward scans, whereas T_{root} and T_{check} are the execution times of Root Scattering Stage and Convergence Check, respectively. Given that the scattering matrix of the root is the biggest scattering matrix, $T_{\text{S,max}}$ is the time required to update the root scattering matrix. According to what is explained in Subsection 1.2, the parallelism p is obtained as $p = T_1/T_\infty$, where now

$$T_1 = T_{\text{init}} + T_{\text{Z}} + T_{\text{S}} + T_{\text{fwd}} + T_{\text{root}} + T_{\text{bwd}} + T_{\text{check}}\,, \tag{28}$$

and T_{Z} and T_{S} are the times required to update all the free parameters and the scattering matrices, respectively, whereas T_{fwd} and T_{bwd} are the times required to perform the whole forward and backward scans. By dividing both the numerator and the denominator of p by T_1, we can express it as a function of the workload fractions (indicated with capital letter C) referred to the different tasks, which we call *costs*. We thus obtain

$$p = 1/\big(C_{\text{init}} + C_{\text{Z,max}} + C_{\text{S,max}} + C_{\text{fwd,max}} + \\ + C_{\text{root}} + C_{\text{bwd,max}} + C_{\text{check}}\big)\,, \tag{29}$$

where the subscripts have same meaning of those in Eq. (27).

As shown in Subsection 1.2, the parallelism p provides a good starting point for choosing the number of threads [34]. However, given that p is typically non-integer, the number of threads can be chosen between $\lceil p \rceil$ and $\lfloor p \rfloor$; hence, p can be best determined experimentally. Nevertheless, a reasonable initial guess is the nearest integer number, i.e., $\lfloor p + 0.5 \rfloor$. Once p is determined, the programmer can follow the strategy "Parallelization by Root Subtrees" by assigning the workload of the root subtrees to the different threads, bearing in mind that *load balancing* must always be met.

5 EXAMPLE OF APPLICATION

In this Section, we provide an example of application of the proposed parallel implementation of HSIM, called parHSIM. We implement both HSIM and parHSIM in C++ language, since it is the main language employed for the design of audio plug-ins, and it offers the best possibilities as far as generation and control of threads are concerned. In particular, we use the *thread support library*, where the *thread* class is employed to manage separate threads of execution, whereas *mutex*, *unique_lock*, and *condition_variable* classes are employed to implement the synchronization barriers.

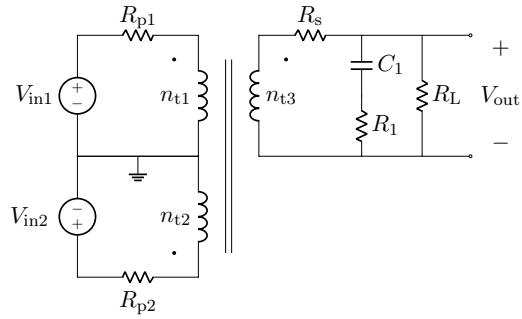


Fig. 6: Circuit schematic of an output stage of a vacuum-tube guitar amplifier.

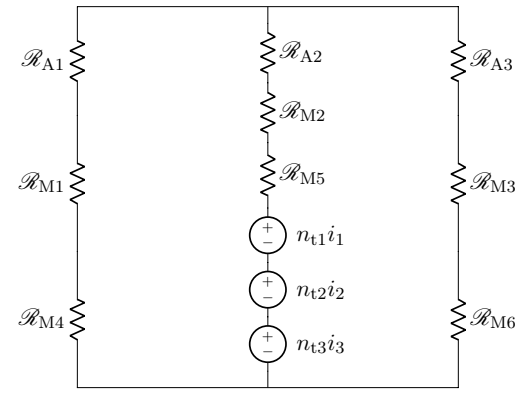


Fig. 7: Magnetic Equivalent Circuit (MEC) of the transformer present in Fig. 6.

Let us consider the circuit shown in Fig. 6 representing an output stage of a vacuum-tube guitar amplifier.

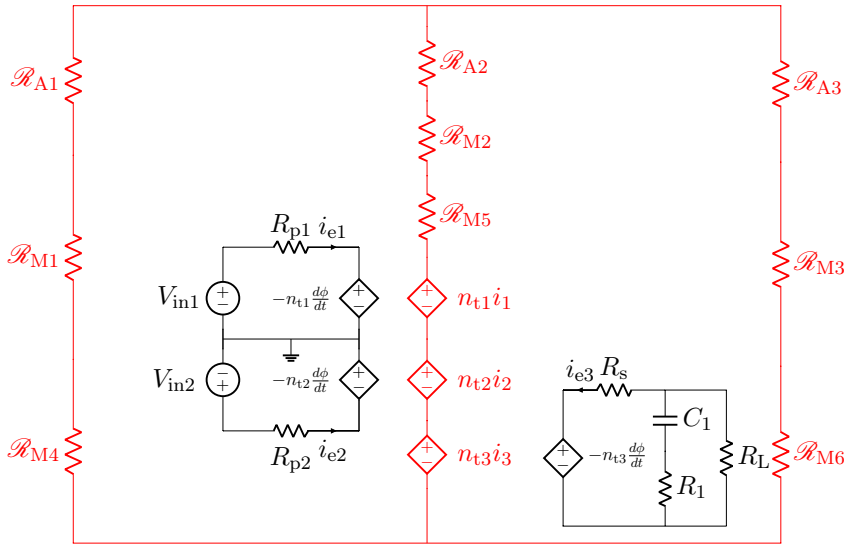


Fig. 8: Multiphysics model of the circuit shown in Fig. 6 encompassing the transformer MEC of Fig. 7. The magnetic domain (in red) and the electrical domain (in black) are coupled by means of ME junctions.

The input signals V_{in1} and V_{in2} represent the two push-pull outputs of a vacuum-tube amplifier. In particular, we consider V_{in1} as the positive half-wave rectified version of the sinusoid $A\sin(2\pi k f_0/f_s)$, where $A = 300$ V is the amplitude, $f_0 = 50$ Hz is the fundamental frequency, and $f_s = 44.1$ kHz is the sampling frequency, whereas V_{in2} as the negative half-wave rectified version of the same signal. Moreover, R_{p1} and R_{p2} are the resistances of the primary windings, R_s is the resistance of the secondary winding, whereas C_1, R_1, and R_L represent a low-pass filter. The transformer is nonlinear and it is modeled in the magnetic domain by means of the Magnetic Equivalent Circuit (MEC) shown in Fig. 7, where $\mathscr{R}_{A1}$, $\mathscr{R}_{A2}$, and $\mathscr{R}_{A3}$ are linear reluctances, whereas $\mathscr{R}_{M1}$, $\mathscr{R}_{M2}$, $\mathscr{R}_{M3}$, $\mathscr{R}_{M4}$, $\mathscr{R}_{M5}$, and $\mathscr{R}_{M6}$ are nonlinear reluctances. With the purpose of modeling transformers available on the market, the nonlinear $B-H$ curve is taken from the Voestalpine isovac 235-35 A datasheet [47]. Reluctances $\mathscr{R}_{A1}$, $\mathscr{R}_{A2}$, and $\mathscr{R}_{A3}$ are realized as in [26]. Instead, $\mathscr{R}_{M1} = \mathscr{R}_{M3} = \mathscr{R}_{M4} = \mathscr{R}_{M6}$ and $\mathscr{R}_{M2} = \mathscr{R}_{M5}$ are implemented as in [26] but with half of the magnetic path length. In fact, in certain cases, it is convenient to split nonlinear reluctances into two or more elements in order to account for local differences in the magnetic material (e.g., due to anisotropy of particular magnetic materials [48]) or different flux distributions [49]. However, for the sake of simplicity, in this work we considered just a single nonlinear $B-H$ curve. In addition, increasing the number of nonlinear elements is also interesting from the view point of our work, which is focused on the parallelizability of processing operations. The coupling between magnetic domain and electric domain is obtained employing ME junctions. The other parameters are set as follows: $n_{t1} = n_{t2} = 100$, $n_{t3} = 50$, $R_{p1} = R_{p2} = 100\ \Omega$, $R_s = 50\ \Omega$, $R_1 = 8.2\ \Omega$, $R_L = 8\ \Omega$, and $C_1 = 10\ \mu$F. For both HSIM and parHSIM, the accuracy of the results is confirmed via a comparison to MathWorks Simscape results. This is done by arranging the Kirchhoff circuit shown in Fig. 8 in Simscape and implementing *ad hoc* blocks for the modeling of nonlinear reluctances. Finally, Backward Euler method with same fixed-time step is selected for the discretization of the time derivatives.

A WD implementation of the circuit in Fig. 8 is shown in Fig. 9. The scattering matrix $\mathbf{S}_M$ of the WD 12-port junction entails the topological information of the magnetic domain, and it is computed substituting into Eq. 9 the following fundamental loop matrix

$$\mathbf{B}_M = \begin{bmatrix} 1 & 1 & -1 & -1 & 1 & 1 & 1 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 0 & -1 & 1 \end{bmatrix}, \tag{30}$$

whereas the scattering matrix $\mathbf{S}_E$ of the WD 5-port junction entails the topological information of the electrical domain, and it is obtained substituting into Eq. 9 the matrix

$$\mathbf{B}_E = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}. \tag{31}$$

The magnetic and the electrical domains are then coupled by means of ME junctions. The port resistances are set according to the "Initialization and Update Stage," and the WD structure is solved employing both HSIM and parHSIM. Further details concerning the HSIM implementation, iterative procedure, and convergence can be found in [27].

As far as parHSIM is concerned, the first operation to be carried out is computing the number of threads, which we consider to be fixed in time. As explained in Subsection 1.2, the parallelism p provides this information, since having a number of threads greater than p (which is also the maximum speedup) may introduce overheads and limit the performance of the parallel algorithm. In order to de-

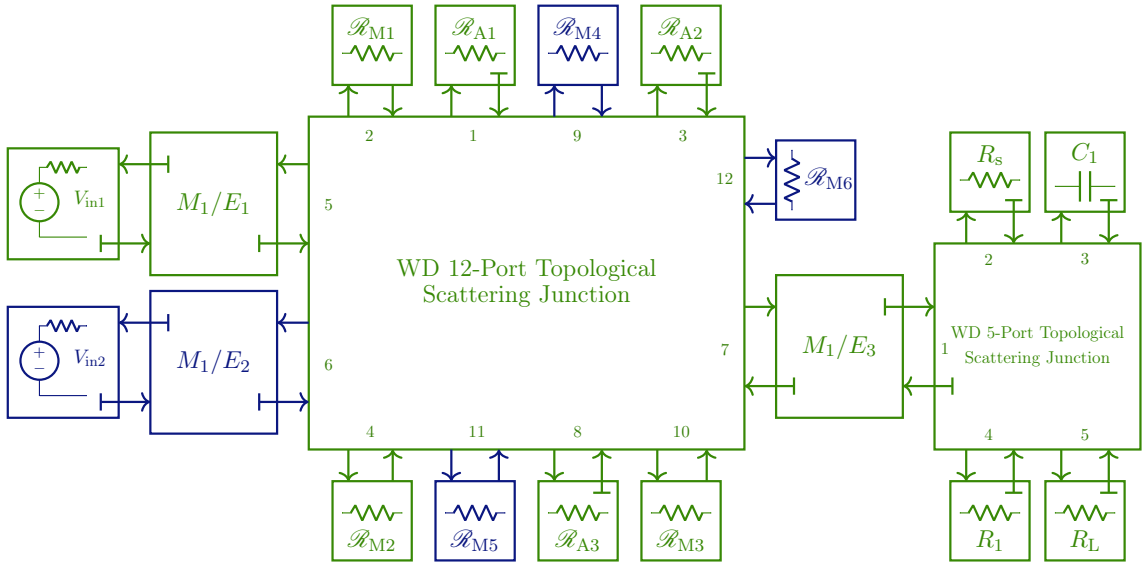


Fig. 9: A possible WD realization of the circuit shown in Fig. 8. The two topological junctions, encompassing the topological information of the two physical domains, are coupled by means of ME junctions. The T-shaped stubs indicate port adaptation. The WD blocks $\mathscr{R}_{M4}$, $\mathscr{R}_{M5}$, $\mathscr{R}_{M6}$, V_{in2}, and M_1/E_2 (depicted in blue) are assigned to the child thread, while all the others are assigned to the main thread.

termine the execution times required to compute p according to Eq. (29), we performed 100 identical runs of the HSIM algorithm (on an Intel Core i7 processor) and we averaged the results. Table 1 shows the measured costs of all the tasks involved in the algorithm, which, once substituted into Eq. (29) yield

$$p \approx 2.26\,. \tag{32}$$

The number of threads is thus set equal to 2.

Once the number of threads is determined, *load balancing* must be considered while assigning tasks to the two workers. We do this by taking into account the execution times of each single WD block in Fig. 9 and the "Parallelization by Subtrees" strategy presented in Section 4. We choose to set up the two threads in a *main/child* configuration, where the *main* thread is also in charge of the synchronization task, whereas the *child* thread executes only computational tasks. Moreover, the child thread notifies the main thread when its task queue is empty (i.e., when it has completed the execution). Looking at Fig. 9, apart from the reluctances directly connected to the root (i.e., the 12-port topological junction), we can identify three root subtrees: (i) a first one composed of blocks M_1/E_1 and V_{in1}; (ii) a second composed of blocks M_1/E_2 and V_{in2}; (iii) a third composed of blocks M_1/E_3, R_s, R_1, R_L, C_1, and the 5-port topological junction. Bearing in mind which blocks can be executed in parallel according to the "Parallelization by Subtrees" strategy, a possible realization of *load balancing* is shown in Fig. 9: the blocks $\mathscr{R}_1$, $\mathscr{R}_{A2}$, $\mathscr{R}_{A3}$, $\mathscr{R}_{M1}$, $\mathscr{R}_{M2}$, $\mathscr{R}_{M3}$, R_s, R_L, R_1, C_1, V_{in1}, M_1/E_1, M_1/E_3, and the 12-port and 5-port topological junctions (depicted in green) are assigned to the main thread, while the blocks $\mathscr{R}_{M4}$, $\mathscr{R}_{M5}$, $\mathscr{R}_{M6}$, V_{in2}, and M_1/E_2 (depicted in blue) are assigned to the child thread. Having all those blocks assigned to a single thread might sound unfair and sub-optimal, but it should be kept in mind that not all of them are concurrently executed. In fact, the scheduling policy cannot dis-

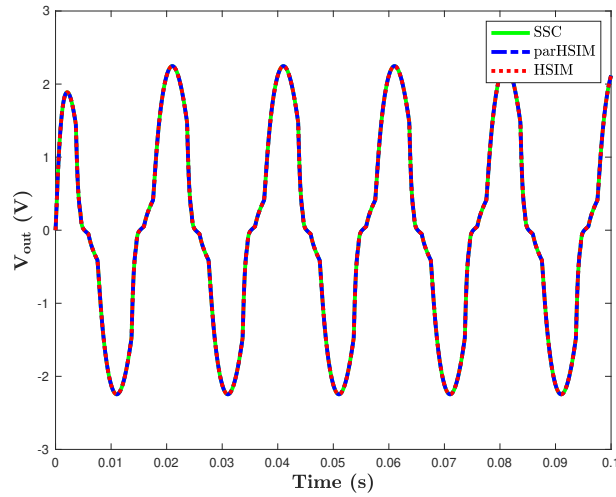


Fig. 10: Voltage V_{out} at the output of the circuit. The continuous green line represents the result of the Simscape (SSC) implementation, whereas the dashed blue line and the dotted red line represent the results of the parHSIM and HSIM implementations, respectively. The accuracy of the proposed method is confirmed by the overlap of the three curves.

Table 1: Costs of HSIM tasks averaged over 100 identical runs of the algorithm.

C_{init}	$C_{\text{Z,max}}$	$C_{\text{S,max}}$	$C_{\text{fwd,max}}$	C_{root}	$C_{\text{bwd,max}}$	C_{check}
0.03	0.02	0.29	0.031	0.02	0.028	0.024

Table 2: Mean execution time $\bar{t}_{\text{sim}}$, standard deviation σ, and mean Real-Time Ratio $\overline{RTR}$ of HSIM and parHSIM.

	$\bar{t}_{\text{sim}}$	σ	$\overline{RTR}$
HSIM	15.05 s	0.24 s	0.92
parHSIM	11.78 s	0.2 s	0.72

regard the constraints imposed by the "Parallelization by Subtrees" strategy.

It is worth pointing out that the nonlinear elements are split equally between the two threads in order to achieve a better *load balancing*. Moreover, we decide to execute the Backward Scattering Stage only on the main thread such that just the synchronization barrier represented by the Root Scattering Stage remains. In fact, the overhead introduced by the parallel execution of the Backward Scattering Stage was greater than the execution time itself. Finally, *load balancing* and parallel implementation are realized also for the "Initialization and Update Stage": the main thread is in charge of the update of $\mathbf{S}_{\text{M}}$ and of the free parameters of three nonlinear reluctances; the child thread is, instead, in charge of the update of $\mathbf{S}_{\text{E}}$, of the matrices of all the ME junctions, and of the free parameters of the other three nonlinear reluctances.

Figure 10 shows the comparison between HSIM (dotted red curve), parHSIM (dashed blue curve), and Simscape (continuous green curve) implementations. The accuracy of the proposed method is confirmed by the overlap of the three curves. Moreover, the accuracy of parHSIM remains unaltered w.r.t. its serial version, since the parallel implementation is realized without introducing any approximation. In order to verify the speedup introduced by parHSIM w.r.t. HSIM, we perform a further test considering now as input signal an audio signal of an electric guitar with duration 16 seconds. We launch 20 identical runs of HSIM and parHSIM (one after the other) with the purpose of measuring a fair execution time t_{sim}. The Real-Time Ratio (RTR) [20, 21], defined as $RTR = (t_{\text{sim}} f_{\text{s}})/K$ where K is the total number of samples, is taken into account for determining whether the algorithms are capable of running in real-time. In fact, a real-time execution is possible only if $RTR < 1$. Table 2 shows the average execution time $\bar{t}_{\text{sim}}$, the standard deviation σ, and the average RTR for both HSIM and parHSIM. Although both the algorithms are able to be executed in real-time (they are both characterized by $RTR < 1$), the proposed parallel implementation results to be more efficient than its serial version. We can, in fact, compute the attained speedup considering the ratio between the average execution time of HSIM and the average execution time of parHSIM obtaining

$$S = \frac{\bar{t}_{\text{sim}}^{\text{HSIM}}}{\bar{t}_{\text{sim}}^{\text{parHSIM}}} = \frac{15.05 \text{ s}}{11.78 \text{ s}} \approx 1.28 . \tag{33}$$

The proposed parallel implementation is thus capable of running nearly 30% faster w.r.t. to its serial implementation. The efficiency of the approach opens up thus new possibilities as far as the simulation of nonlinear audio circuits is concerned, allowing the real-time execution of increasingly complex circuits.

6 CONCLUSIONS

In this paper, we addressed for the first time the parallel implementation of HSIM, i.e., a WD iterative method able to emulate circuits with multiple nonlinearities and characterized by a high number of *embarrassingly parallelizable* operations. Such a property allows us to distribute the workload on different threads of execution. Hence, the proposed method, which we called parHSIM, matches well with modern audio hardware characterized by multiple CPU cores, which, if correctly exploited, can lead to highly-efficient implementations. We showed how it is possible to apply simple concepts borrowed from the parallel computing theory to the WDF theory. In particular, in order to determine the maximum speedup achievable by a parallel implementation, we considered the parallelism p, defined as the ratio between work and span. We then showed how it is possible to compute such quantities from the Direct Acyclic Graph (DAG), i.e., a graph representation of the whole workload, and we discussed three possible strategies for deriving DAGs associated to the HSIM algorithm. The "Parallelization by Subtrees" proved to be the most effective strategy, since it entails the lowest number of synchronization barriers and it allows the highest degree of flexibility in terms of *load balancing*. Moreover, we showed how it is possible to determine the number of threads to be used for the implementation. The method has been employed for the emulation of an output stage of a vacuum-tube guitar amplifier characterized by a nonlinear audio transformer. Then, we drew a performance comparison with HSIM in order to assess the attained speedup. The proposed parallel approach was able to run in real-time and turned out to be nearly 30% faster w.r.t. the serial implementation, keeping unaltered the overall accuracy. It is worth pointing out that such a speedup was reached considering a circuit with just six nonlinear elements, which is a rather restrained number of nonlinearities. Hence, we are confident that it is possible to achieve greater speedups when the number of nonlinear elements becomes higher since, given that their computation is al-

ways *embarrassingly parallelizable*, the parallelism p (and thus the maximum speedup) would increase accordingly. Moreover, higher speedups can be also obtained considering structures with more root subtrees or with a deeper tree. In the context of VA modeling, parHSIM proves thus to be a promising algorithm, since it enables the real-time emulation of increasingly complex nonlinear audio circuits

Future work may concern the analysis and implementation of different and more refined scheduling policies. Moreover, the Dynamic Scattering matrix Recomputation (DSR) method [21] can be considered for further enhancing the efficiency of parHSIM. Another interesting development of the method, in the context of simulations of very large nonlinear circuits, might be the implementation of networks of solvers, exploiting at its highest the embarrassingly parallelizable workload characterizing HSIM.

7 REFERENCES

[1] U. Zölzer, *DAFX: Digital Audio Effects* (John Wiley and Sons, Ltd, Chichester, UK) (2011 mar), doi:10.1002/9781119991298.

[2] J. Pekonen, V. Välimäki, "The Brief History of Virtual Analog Synthesis," presented at the *The 6th Forum Acusticum*, pp. 461–466 (2011 jun).

[3] A. Wright, E. P. Damskägg, V. Välimäki, "Real-Time Black-Box Modelling With Recurrent Neural Networks," presented at the *22nd International Conference on Digital Audio Effects (DAFx-19)*, pp. 1–8 (2019 sep).

[4] D. Bouvier, T. Hélie, D. Roze, "Phase-based Order Separation for Volterra Series Identification," *International Journal of Control*, vol. 94, no. 8, pp. 2104–2114 (2021 aug), doi:10.1080/00207179.2019.1694175.

[5] A. Primavera, S. Cecchi, L. Romoli, M. Gasparini, F. Piazza, "Approximation of Dynamic Convolution Exploiting Principal Component Analysis: Objective and Subjective Quality Evaluation," presented at the *133rd Audio Engineering Society Convention* (2012 oct).

[6] F. Eichas, S. Möller, U. Zölzer, "Block-Oriented Modeling of Distortion Audio Effects Using Iterative Minimization," presented at the *18th International Conference on Digital Audio Effects (DAFx-15)* (2015 nov).

[7] G. Borin, G. De Poli, D. Rocchesso, "Elimination of Delay-Free Loops in Discrete-Time Models of Nonlinear Acoustic Systems," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 5, pp. 597–604 (2000), doi:10.1109/89.861380.

[8] A. Falaize, T. Hélie, "Simulation of an Analog Circuit of a Wah Pedal: A Port-Hamiltonian Approach," presented at the *135th Audio Engineering Society Convention*, pp. 548–556 (2013 may).

[9] A. Fettweis, "Wave Digital Filters: Theory and Practice," *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327 (1986), doi:10.1109/PROC.1986.13458.

[10] A. Bernardini, A. Sarti, "Towards Inverse Virtual Analog Modeling," presented at the *22nd International Conference on Digital Audio Effects (DAFx-19)* (2019 sep).

[11] K. Meerkötter, R. Scholz, "Digital Simulation of Nonlinear Circuits by Wave Digital Filter Principles," presented at the *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 1, pp. 720–723 (1989 aug), doi:10.1109/iscas.1989.100452.

[12] A. Bernardini, P. Maffezzoni, A. Sarti, "Linear Multistep Discretization Methods with Variable Step-Size in Nonlinear Wave Digital Structures for Virtual Analog Modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 1763–1776 (2019), doi:10.1109/TASLP.2019.2931759.

[13] S. Bilbao, *Wave and Scattering Methods for Numerical Simulation* (John Wiley and Sons, Ltd, Chichester, UK) (2004 may), doi:10.1002/0470870192.

[14] C. W. Ho, A. E. Ruehli, P. A. Brennan, "The Modified Nodal Approach to Network Analysis," *IEEE Transactions on Circuits and Systems*, vol. 22, no. 6, pp. 504–509 (1975), doi:10.1109/TCS.1975.1084079.

[15] S. Petrausch, R. Rabenstein, "Wave Digital Filters with Multiple Nonlinearities," presented at the *12th European Signal Processing Conference*, vol. 06-10-Sept, pp. 77–80 (2015 sep).

[16] M. J. Olsen, K. J. Werner, J. O. Smith, "Resolving Grouped Nonlinearities in Wave Digital Filters Using Iterative Techniques," presented at the *19th International Conference on Digital Audio Effects (DAFx-16)*, pp. 279–286 (2016).

[17] C. Christoffersen, "Transient Analysis of Nonlinear Circuits Based on Waves," in *Scientific Computing in Electrical Engineering SCEE 2008*, pp. 159–166 (Springer, Berlin, Heidelberg) (2010), doi:10.1007/978-3-642-12294-1_21.

[18] L. Kolonko, J. Velten, A. Kummert, "Automatic Differentiating Wave Digital Filters with Multiple Nonlinearities," presented at the *European Signal Processing Conference*, pp. 146–150 (2021 jan), doi:10.23919/EUSIPCO47968.2020.9287674.

[19] A. Bernardini, P. Maffezzoni, L. Daniel, A. Sarti, "Wave-Based Analysis of Large Nonlinear Photovoltaic Arrays," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1363–1376 (2018), doi:10.1109/TCSI.2017.2756917.

[20] A. Bernardini, K. J. Werner, P. Maffezzoni, A. Sarti, "Wave Digital Modeling of the Diode-Based Ring Modulator," presented at the *144th Audio Engineering Society Convention* (2018 may).

[21] A. Proverbio, A. Bernardini, A. Sarti, "Toward the Wave Digital Real-Time Emulation of Audio Circuits with Multiple Nonlinearities," presented at the *Proceedings of 28th European Signal Processing Conference (EUSIPCO)*, pp. 151–155 (2021 jan), doi:10.23919/Eusipco47968.2020.9287449.

[22] A. Bernardini, P. Maffezzoni, A. Sarti, "Vector Wave Digital Filters and Their Application to Circuits With Two-Port Elements," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 3, pp. 1269–1282 (2021), doi:10.1109/TCSI.2020.3044002.

[23] D. Albertini, A. Bernardini, A. Sarti, "Antiderivative Antialiasing Techniques in Nonlinear Wave Digital

Structures," *Journal of the Audio Engineering Society*, vol. 69, no. 7/8, pp. 448–464 (2021), doi:10.17743/jaes.2021.0017.

[24] R. Giampiccolo, M. G. de Bari, A. Bernardini, A. Sarti, "Wave Digital Modeling and Implementation of Nonlinear Audio Circuits with Nullors," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3267–3279 (2021), doi:10.1109/TASLP.2021.3120627.

[25] A. Bernardini, E. Bozzo, F. Fontana, A. Sarti, "A Wave Digital Newton-Raphson Method for Virtual Analog Modeling of Audio Circuits with Multiple One-Port Nonlinearities," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2162 – 2173 (2021), doi:10.1109/TASLP.2021.3084337.

[26] R. Giampiccolo, A. Bernardini, G. Gruosso, P. Maffezzoni, A. Sarti, "Multiphysics Modeling of Audio Circuits with Nonlinear Transformers," *Journal of the Audio Engineering Society*, vol. 69, no. 6, pp. 374–388 (2021 jun), doi:10.17743/jaes.2021.0008.

[27] R. Giampiccolo, A. Bernardini, G. Gruosso, P. Maffezzoni, A. Sarti, "Multidomain Modeling of Nonlinear Electromagnetic Circuits using Wave Digital Filters," *International Journal of Circuit Theory and Applications*, vol. 50, no. 2, pp. 539–561 (2022), doi:10.1002/cta.3146.

[28] M. Herlihy, N. Shavit, *The Art of Multiprocessor Programming* (Morgan Kaufmann Publishers Inc., San Francisco, USA) (2012 jun).

[29] M. A. N. Al-hayanni, F. Xia, A. Rafiev, A. Romanovsky, R. Shafik, A. Yakovlev, "Amdahl's Law in the Context of Heterogeneous Many-Core Systems – A Survey," *IET Computers and Digital Techniques*, vol. 14, no. 4, pp. 133–148 (2020 jul), doi:10.1049/iet-cdt.2018.5220.

[30] J. Gustafson, "Reevaluating Amdahl's Law," *Communications of the ACM*, vol. 31, no. 5, pp. 532–533 (1988 may), doi:10.1145/42411.42415.

[31] X. Li, M. Malek, "Analysis of Speedup and Communication/Computation Ratio in Multiprocessor Systems," presented at the *Proceedings. Real-Time Systems Symposium*, vol. 35, pp. 282–288 (1988), doi:10.1109/REAL.1988.51123.

[32] X.-H. Sun, L. Ni, "Another View on Parallel Speedup," presented at the *1990 ACM/IEEE Conference on Supercomputing*, pp. 324–333 (1990 nov), doi:10.1109/SUPERC.1990.130037.

[33] A. S. Cassidy, A. G. Andreou, "Beyond Amdahl's law: An objective Function that Links Multiprocessor Performance Gains to Delay and Energy," *IEEE Transactions on Computers*, vol. 61, no. 8, pp. 1110–1126 (2012), doi:10.1109/TC.2011.169.

[34] C. E. Leiserson, T. B. Schardl, "A Work-Efficient Parallel Breadth-First Search Algorithm (or How to Cope with the Nondeterminism of Reducers)," presented at the *22nd ACM symposium on Parallelism in algorithms and architectures - SPAA '10*, pp. 303–314 (2010), doi:10.1145/1810479.

[35] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms* (MIT Press) (2009).

[36] H. El-Rewini, H. Ali, T. Lewis, "Task Scheduling in Multiprocessing Systems," *Computer*, vol. 28, no. 12, pp. 27–37 (1995), doi:10.1109/2.476197.

[37] G. O. Martens, K. Meerkötter, "On N-Port adaptors for Wave Digital Filters with Application to a Bridged-Tee Filter," presented at the *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 514–517 (1976).

[38] A. Bernardini, K. J. Werner, J. O. Smith, A. Sarti, "Generalized Wave Digital Filter Realizations of Arbitrary Reciprocal Connection Networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 2, pp. 694–707 (2019), doi:10.1109/TCSI.2018.2867508.

[39] E. Laithwaite, "Magnetic Equivalent Circuits for Electrical Machines," *Proceedings of the Institution of Electrical Engineers*, vol. 114, no. 11, p. 1805 (1967), doi:10.1049/piee.1967.0344.

[40] S. D'Angelo, J. Pakarinen, V. Valimaki, "New Family of Wave-Digital Triode Models," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 2, pp. 313–321 (2013 feb), doi:10.1109/TASL.2012.2224340.

[41] K. J. Werner, V. Nangia, J. O. Smith, J. S. Abel, "Resolving Wave Digital Filters with Multiple/Multiport Nonlinearities," presented at the *18th International Conference on Digital Audio Effects* (2015 dec).

[42] T. Schwerdtfeger, A. Kummert, "Nonlinear Circuit Simulation by Means of Alfred Fettweis' Wave Digital Principles," *IEEE Circuits and Systems Magazine*, vol. 19, no. 1, pp. 55–C3 (2019), doi:10.1109/MCAS.2018.2872666.

[43] L. O. Chua, S. M. Kang, "Section-Wise Piecewise-Linear Functions: Canonical Representation, Properties, and Applications," *Proceedings of the IEEE*, vol. 65, no. 6, pp. 915–929 (1977), doi:10.1109/PROC.1977.10589.

[44] R. M. Corless, G. H. Gonnet, D. E. Hare, D. J. Jeffrey, D. E. Knuth, "On the Lambert W function," *Advances in Computational Mathematics*, vol. 5, no. 1, pp. 329–359 (1996), doi:10.1007/bf02124750.

[45] R. M. Corless, D. J. Jeffrey, "The Wright ω function," in *Artificial Intelligence, Automated Reasoning, and Symbolic Computation*, pp. 76–89 (2002), doi:10.1007/3-540-45470-5_10.

[46] A. Bernardini, A. Sarti, "Canonical Piecewise-Linear Representation of Curves in the Wave Digital Domain," presented at the *25th European Signal Processing Conference (EUSIPCO)*, pp. 1125–1129 (2017 aug), doi:10.23919/EUSIPCO.2017.8081383.

[47] Voestalpine, "Voestalpine isovac Electrical Steel 235-35 A - Datasheet," URL `https://www.voestalpine.com/isovac/en/Product-overview/Data-sheets`.

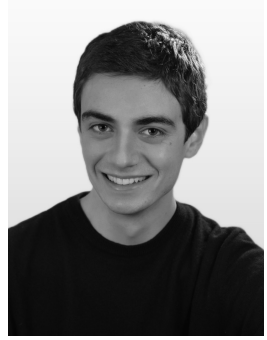
[48] E. Brück, *Handboook of Magnetic Materials* (North Holland) (2019 nov).

[49] G. Shilyashki, H. Pfützner, "Magnetic Circuit Modelling of Transformer Core Induction – Reso-

lution and Accuracy," *IET Electric Power Applications*, vol. 11, no. 7, pp. 1341–1346 (2017 aug), doi: 10.1049/iet-epa.2016.0812.

THE AUTHORS

Riccardo Giampiccolo



Antonino Natoli

Alberto Bernardini

Augusto Sarti

Riccardo Giampiccolo received both the B.S. and the M.S. degrees in Electronics Engineering from the Politecnico di Milano, Italy, in 2017 and 2020, respectively. He is currently a Ph.D. Candidate in Information Technology with the Dipartimento di Elettronica, Informazione e Bioingegneria of the Politecnico di Milano, Italy. His main research interests include signal processing methodologies for small-size transducers in consumer electronics.

●

Antonino Natoli received the B.S. degree in Computer Engineering in 2019 and the M.S. degree in Music and Acoustic Engineering in 2021, both from the Politecnico di Milano, Italy. He is currently working as software engineer at INVENTVM Semiconductor.

●

Alberto Bernardini received the B.S. degree from the University of Bologna, Italy, and M.S. degree (cum laude) from the Politecnico di Milano, Italy, both in computer engineering, in 2012 and 2015, respectively. In 2019, he received his Ph.D. degree (cum laude) in information engineering from Politecnico di Milano, Italy, where he is currently an Assistant Professor. His main research interests are audio signal processing and modeling of nonlinear systems. He authored more than 30 publications in international journals and proceedings of international conferences. He is also the first author of an international patent.

●

Augusto Sarti received his Ph.D. in Information Engineering from the University of Padova, Italy, in 1993, with a joint graduate program with the University of California, Berkeley. In 1993, he joined the Politecnico di Milano (PoliMI), Italy, where he is currently a Full Professor. From 2013 to 2017 he held a professorship at the University of California, Davis. At PoliMI he currently coordinates the research activities of the Musical Acoustics Lab and the Sound and Music Computing Lab, and the M.Sci. program in "Music and Acoustic Engineering". He has coauthored well over 300 scientific publications on international journals and congresses and numerous patents in the multimedia signal processing area. His current research interests are in the area of audio and acoustic signal processing, with particular focus on audio and acoustic signal processing; music information retrieval; and musical acoustics. He served two terms with the IEEE Technical Committee on Audio and Acoustics Signal Processing. He also served as Associate Editor of IEEE/ACM Tr. on Audio Speech and Language Processing, and as Senior Area Editor of IEEE Signal Processing Letters. He is currently serving in the EURASIP board of directors.