

SHARC WDF Library and R-Solver – Tools for Prototyping Wave
Digital Filter Models of Circuits

by

Samuel Schachter

Submitted in Partial Fulfillment of the
Requirements for the Degree
Master of Science

Supervised by Professor Mark Bocko

Department of Electrical Engineering
Arts, Sciences and Engineering

Edmund A. Hajim School of Engineering and Applied Sciences

University of Rochester

Rochester, New York

2021

Contents

Biographical Sketch	iv
Acknowledgments	v
Abstract.....	vi
Contributors and Funding Sources.....	vii
List of Tables	viii
List of Figures	ix
List of Code Listings	x
Chapter 1 – Introduction to Wave Digital Filters	1
1.1 - Motivation	1
1.2 - Definitions.....	3
1.2.1 - Ports.....	3
1.2.2 - Waves	4
1.3 - Code Realization.....	5
1.4 - Linear One-Port Derivations	7
1.4.1 - Resistor	7
1.4.2 - Ideal and Resistive Voltage Sources	8
1.5 - Reactive One-Port Derivations.....	10
1.5.1 - Discretization Methods.....	11
1.5.2 - Capacitor	13
1.6 - Two-Port Derivations.....	15
1.6.1 - Two-Port Series Adaptor (Inverter)	16
1.7 - Three-Port Derivations.....	17
1.7.1 - Series Adaptor.....	18
1.7.2 - Parallel Adaptor	20
1.8 - Nonlinear One-Port Derivation	21
1.8.1 – Diode.....	22
1.8.2 - Diode Pair	23
1.8.3 – Antiderivative Antialiasing of Diode Models	24
1.9 - SPQR Trees and R-type Adaptors	26

1.9.1 – SPQR Tree Derivation for Butterworth Low Pass Filter	27
1.9.2 – SPQR Tree Derivation for Single-Pot Tone Control Circuit	29
1.9.3 - SPQR Tree Derivation of MXR Distortion+ Pedal.....	31
1.9.4 – Scattering Behavior of R-type Adaptors.....	36
1.9.5 – MXR Distortion+ R-Type Adaptor Derivation	40
1.10 – Chapter Summary	41
Chapter 2 – SHARC Audio Module WDF Library	42
2.1 – Previous Work	42
2.2 – SHARC Audio Module Bare Metal Framework	43
2.3 - Library Overview	45
2.4 – Butterworth Low Pass Filter Implementation Example.....	50
2.5 – MXR Distortion+ Implementation Example.....	52
2.6 – Performance and Limitations of SHARC WDF Library.....	55
2.7 – Chapter Summary	56
Chapter 3 – R-solver.....	57
3.1 – Previous Work	57
3.2 – R-Solver Overview	58
3.3 – MXR Distortion+ Scattering Matrix Derivation using R-Solver.....	61
3.3 - Chapter Summary	62
Bibliography	64

Biographical Sketch

Samuel Schachter was born and raised in Westborough, Massachusetts. He attended the University of Rochester and graduated with a Bachelor of Science in Audio and Music Engineering in May 2020. He is currently a candidate for the degree of Master of Science in Electrical Engineering from the University of Rochester. His central research was in Virtual Analog modeling under the direction of advisor Professor Mark Bocko.

Acknowledgments

I cannot thank my mentors, friends, or family enough for their guidance and support while completing this thesis. First and foremost, I would like to thank my advisor Dr. Mark Bocko, whose teaching in AME 140 introduced me to the world of electrical engineering and inspired me to pursue this research. I would also like to thank my professors in the electrical engineering department for providing me with resources, guidance, and confidence to consider academic research of this depth.

This thesis would not exist without the extensive guidance of Kurt Werner. The hours we spent deep-diving into the theory behind Wave Digital Filters and working through problems together were instrumental to my progress and instilled me with a deep appreciation for Virtual Analog modeling. I would be equally lost without the collaboration of Jatin Chowdhury, whose expertise and willingness to debug code with me were keys to my success. I'd also like to thank Alberto Bernardini, Chad Wentworth, and Murray Anderson for their help in answering my theoretical and implementational questions.

Finally, for their love and support I'd like to thank my parents Bernie and Benita, my siblings Mike, Abby, Allie, and Rishi, and my partner Alexa, without whom I would certainly not be where I am today.

Abstract

Wave Digital Filters (WDF) represent a burgeoning approach to digitally modeling analog circuitry. WDFs simulate the wave scattering relationship between discrete circuit elements, allowing for a computationally efficient simulation of a circuit's behavior in real-time [1]. In recent years, researchers have made great strides in expanding the class of circuits that can be modeled using WDFs. One such development is the introduction of *R*-type adaptors, which are used to represent circuit topologies that cannot be broken down into simple serial or parallel connections [2].

In this thesis, I present two tools to aid in the prototyping and derivation of WDF models. The first is a C library of Wave Digital Filter objects which is optimized for use with the SHARC Audio Module, a floating-point DSP processor designed for audio software prototyping [3]. The second is *R-solver*, a MATLAB live script that algorithmically derives the scattering parameters for any *R*-type adaptor [1].

Chapter 1 presents the background information necessary to form WDF structures and implement them as a real-time traversable connection tree. Chapter 2 discusses the SHARC WDF library and the specific challenges of creating Wave Digital Filter models for the SHARC. Finally, Chapter 3 presents *R-solver* and how it may be used to simplify the WDF model derivation process.

Contributors and Funding Sources

This work was supported by a dissertation committee consisting of Professor Mark Bocko (advisor), Professor Michael Heilemann of the Department of Electrical Engineering, and Professor Steve McAleavey of the Department of Biomedical Engineering. The code described in Chapter 2 was based on code provided with permission by Jatin Chowdhury (EE, Stanford 2018).

List of Tables

Table 1	List of Relevant MNA Element Stamps	39
Table 2	Performance of SHARC WDF Library on Cores 1 and 2 for the Derived Circuit Models	55

List of Figures

1.1	Kirchhoff and Wave Digital Representation of a One-port	3
1.2	Unadapted and Adapted Resistor One-ports	7
1.3	(a) Unadapted Voltage Source One-port, (b) Adapted Resistive Voltage Source One-port	8
1.4	Unadapted and Adapted Capacitor One-Ports	13
1.5	Unadapted and Adapted Inverter Two-ports	16
1.6	Unadapted and Adapted Series Two-ports	18
1.7	Unadapted and Adapted Series Two-ports	20
1.8	Unadapted Diode One-port	22
1.9	Unadapted Diode Pair One-port	23
1.10	WDF Derivation of Butterworth Low Pass Filter Circuit	28
1.11	WDF Derivation of Passive Tone Control Circuit	29
1.12	MXR Distortion+ Circuit	31
1.13	Ideal Op-Amps Modeled by (a) a Nullor and (b) an Ideal VCVS	32
1.14	Resistive VCVS Macromodel of an Op-Amp	32
1.15	MXR Distortion+ Wave Digital Filter Derivation	35
1.16	Thévenin Port Equivalent	36
1.17	R -type Adaptor with Thévenin Port Equivalents	40
1.18	Figure 1.18: MNA Matrix \mathbf{X} for MXR Distortion+ R -type Adaptor	41
2.1	Audio Flow through SHARC Cores 1 and 2	44
2.2	SHARC WDF Library Class Structure	46
2.3	Cross-Controlled Model of MXR Distortion+ Circuit: (a-b) Sub-circuits 1 and 2 and (c-d) Corresponding SPQR Trees	53
3.1	LTSpice Representation of MXR Distortion+ R -type Adaptor	60
3.2	Scattering Matrices Derived with (a) Nullor, (b) Ideal VCVS, and (c) Resistive VCVS Macromodel	62

List of Code Listings

Listing 1	SHARC WDF Library Resistor Implementation	47
Listing 2	SHARC WDF Library Series Adaptor Implementation	49
Listing 3	SHARC Butterworth Low Pass Filter Class Definition	51
Listing 4	SHARC Butterworth Low Pass Filter Implementation	52
Listing 5	SHARC MXR Distortion+ Implementation	54

Chapter 1 – Introduction to Wave Digital Filters

1.1 – Motivation

Audio software plugins grant music producers more flexibility and precision for the control of audio signals than analog circuitry. Despite this, many of the most popular plugins on the market are those that faithfully emulate analog sound-effects processors. Digital audio signal processing methods simply don't seem to capture the 'magic' of analog hardware. Try as they might, audio plugins struggle to replicate the subtle interactions, distortion, and noise characteristics of electrons streaming through traditional analog circuit components.

Practitioners of Virtual Analog modeling have derived numerous techniques to digitally simulate the performance of analog hardware. One such method involves the derivation of a system transfer function that describes the circuit's voltage-current relationship, which may then be implemented using biquad or state-space filters [4]. However, depending on the complexity of the circuit its transfer function can be tedious to derive. Even more difficult is to parameterize the transfer function to allow control over the sound of the filter.

Wave Digital Filters (WDFs) were first described by Alfred Fettweis in 1970 as a method of designing digital filters with the same structure as classical filters in lattice or ladder configurations [5]. WDFs are characterized by the use of wave variables as signal inputs and outputs as opposed to the more traditional Kirchhoff variables, voltage and

current. By discretizing each circuit element as well as the connections between elements, the resulting filters capture the behavior of their passive analog counterparts.

In recent years, WDFs have been employed as a method of Virtual Analog modeling, and research in the field has shifted towards expanding the class of circuits and circuit elements that can be modeled. One such method that will be explored in greater detail in this thesis is the use of Modified Nodal Analysis (MNA) to derive the so-called *R*-type adaptor, a general adaptor that can be used to represent any circuit topology that cannot be decomposed into series or parallel connections. Other methods of handling such circuit topologies have been introduced ([6], [7]), but none are as robust as MNA.

In this chapter, an abbreviated overview of Wave Digital theory is presented. In sections 1.2-3, preliminary definitions are provided and the requirements for creating a computationally realizable WDF structure are outlined. Next, sections 1.4-8 provide examples of the adaptor derivation process for a variety of linear and nonlinear circuit elements. Finally, section 1.9 explains the process of deriving a WDF's SPQR connection tree and *R*-type adaptors from a circuit by deriving three example circuits, including a model of the MXR Distortion+ guitar pedal.

1.2 - Definitions

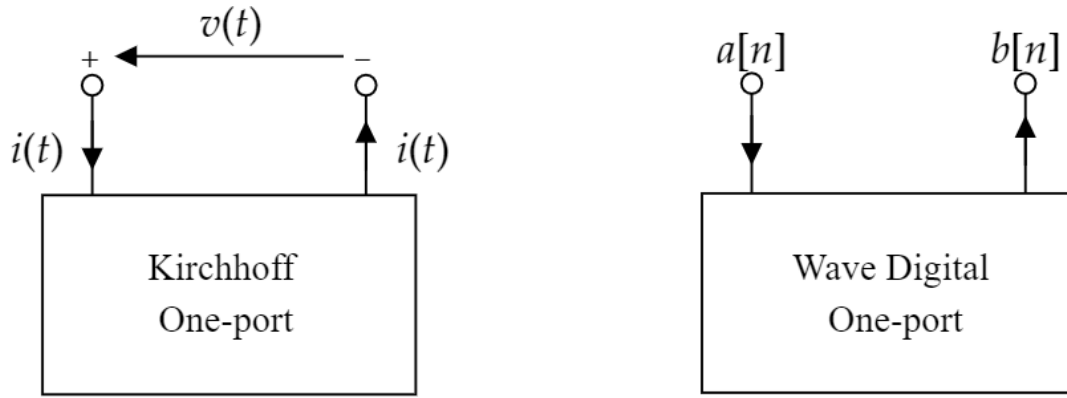


Figure 1.1: Kirchhoff and Wave Digital Representation of a One-port

1.2.1 - Ports

Elements in any lumped physical system, be they electrical or acoustical, can be characterized by the behavior at their ports using dual variable pairs. In an electrical context, an element is described by its Kirchhoff-domain variables, namely the voltage $v(t)$ between its negative and positive terminal as well as the current $i(t)$ flowing through its terminals [1], [8]. An electrical element with two terminals can be described as a one-port element, as shown in Figure 1.1a. Virtual Analog designers are concerned with the emulation of analog filters, which can be described as an interconnected network of N -port elements. An analog network consisting only of passive elements will behave passively and stably, so a digital filter emulating its analog counterpart should reflect similar energetic properties [8].

Wave Digital Filters emulate analog filters in part by characterizing one-ports in terms of wave variables. Wave variables result from a linear, invertible transformation of

Kirchhoff variables which allows for the network to be defined in terms of the energy incident to, and reflected from, a circuit element [5], [9]. Figure 1.1b shows a one-port element defined by its incident wave $a[n]$, reflected wave $b[n]$, and free parameter R_p , commonly referred to as the port resistance.

1.2.2 - Waves

Wave variables are commonly defined in terms of either voltage waves, current waves, or power waves. Each definition is based on a physical interpretation of the linear transformation, and each has its advantages and disadvantages. For a one-port with voltage v and current i , *voltage waves* are defined as

$$a = v + iR_p \quad (1.1)$$

$$b = v - iR_p \quad (1.2)$$

Assuming $R_p \neq 0$, this variable definition is invertible, and the inverse definitions are as follows:

$$v = \frac{a + b}{2} \quad (1.3)$$

$$i = \frac{a - b}{2R_p}. \quad (1.4)$$

Voltage waves are the most used wave variable transformation because they are efficiently computable and physically analogous to how analog circuits are typically analyzed.

Power waves are the next most common wave variable transformation. They are typically preferred for use in time-varying circuits or nonlinear circuits [8]. The power wave variable definition is defined as

$$a = \frac{v + iR_p}{2\sqrt{R_p}} \quad (1.5)$$

$$b = \frac{v - iR_p}{2\sqrt{R_p}} \quad (1.6)$$

and its inverse is defined as

$$v = \frac{\sqrt{R_p}(a + b)}{2} \quad (1.7)$$

$$i = \frac{a - b}{2\sqrt{R_p}}. \quad (1.8)$$

Current waves, the dual of voltage waves, are less commonly used. Their transformation and inverse are defined as follows:

$$a = \frac{V}{R_p} + I \quad (1.9)$$

$$b = \frac{V}{R_p} - I \quad (1.10)$$

$$v = \frac{R_p(a + b)}{2} \quad (1.11)$$

$$i = \frac{a - b}{2}. \quad (1.12)$$

1.3 - Code Realization

Generally, a digital filter is realizable at a sampling rate F_s under two conditions. First, its signal flow diagram must be computable using a combination of adds, multiplies, and delays. Second, there must not be an instantaneous relationship between the input and output of the filter; this would correspond to a signal-flow diagram containing a branch

whose transfer function equals zero [1], [5]. To ensure this condition, all such branches (commonly referred to as *delay-free loops*) in the WDF structure must be removed. The process of removing delay-free loops is referred to as *adaptation*.

Given that each element of a Wave Digital Filter is represented by a discrete N -port, WDF structures are best organized into *connection tree* structures. Connection trees evolved from the concept of the systematized Binary Connection Tree (BCT) [10], [11]. A BCT is fully described by its *root*, which has only downward-facing ports, *leaves*, which have only one upward-facing port and no downward-facing ports, and *adaptors*, which have one upward-facing port and one or more downward-facing port.

The computation of a Wave Digital Filter characterized by a connection tree can be described by a cycle of “forward scans” and “backward scans”. During a forward scan, the states of the WDF elements are computed starting at the leaves, traversing up the tree via the adaptors towards the root. At the root, the relationship between the input wave a and the output b is computed. Finally, the backward scan propagates the computation back down towards the leaves, where all filter states are updated and the cycle begins anew [10]. To remove delay-free loops in this process, it is thus necessary to *adapt* the upward-facing port of each element in the tree other than the root. To adapt a port is as simple as choosing a value for the free parameter R_p which eliminates the instantaneous relationship between $a[n]$ and $b[n]$ at a given sample n .

In the following sections, it will be shown that some elements cannot be adapted. Given that only the root of a connection tree can be left unadapted, circuits with multiple unadaptable elements must be emulated using multiple *subtrees*, with each subtree

representing a section of the circuit [12] The output of one subtree can then be fed into the input of another without introducing delay-free loops. This strategy works particularly well when emulating large linear circuits. Other strategies can be invoked when deriving WDF structures for circuits with multiple nonlinearities, but this goes beyond the scope of this thesis ([1], [6], [7], [13]–[15]).

1.4 - Linear One-Port Derivations

Linear one-port electronic elements are characterized by the relationship between the voltage and current at their lone port. In this section the wave-domain equations for two pertinent linear one-port elements will be derived. This is accomplished by substituting the wave definition directly into the device's Kirchhoff-domain equation to solve for $b[n]$. For all port derivations, the voltage wave definition shall be assumed. For a complete list of element derivations, refer to [1].

1.4.1 - Resistor

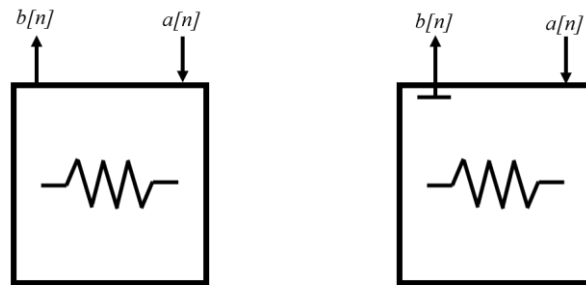


Figure 1.2: Unadapted and Adapted Resistor One-ports

Ohm's law fully characterizes the continuous-time relationship between the port voltage $v(t)$ and port current $i(t)$ of a resistor with resistance R

$$v(t) = i(t)R. \quad (1.13)$$

Plugging (1.3-4) into (1.13) and replacing t with n

$$\frac{a[n] + b[n]}{2} = \frac{R(a[n] - b[n])}{2R_p} \quad (1.14)$$

then solving for $b[n]$ returns

$$b[n] = \frac{R - R_p}{R + R_p} a[n], \quad (1.15)$$

the resistor's unadapted wave equation. Since $b[n]$ instantaneously depends on $a[n]$, this equation can be only used if the resistor is at the root of the connection tree. It should be noted that equation (1.15) holds for voltage, power, and current waves.

To adapt the WDF resistor, we must select a value for the port resistance R_p such that this dependence is eliminated. This is accomplished by setting $R_p = R$, rendering the scattering coefficient $\frac{R-R_p}{R+R_p}$ equal to zero and returning the adapted equation [1]

$$b[n] = 0. \quad (1.16)$$

Using the adapted wave equation (1.16) allows the resistor to be used as a leaf in a WDF connection tree.

1.4.2 - Ideal and Resistive Voltage Sources

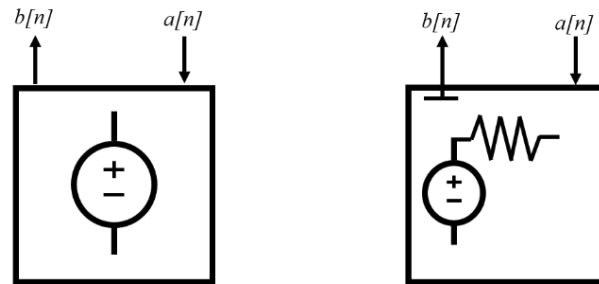


Figure 1.3: (a) Unadapted Voltage Source One-port, (b) Adapted Resistive Voltage Source One-port

An ideal voltage source is characterized by the voltage $e(t)$ across its port

$$v_0(t) = e(t) \quad (1.17)$$

and draws sufficient current $i(t)$ across the port to attain voltage $e(t)$. Plugging in (1.3-4), replacing index t with n , and solving for $b[n]$ yields the discrete-time unadapted wave-domain equation

$$b[n] = 2e[n] - a[n]. \quad (1.18)$$

Unlike (1.15), this equation differs depending on the choice of either voltage, power, or current waves. The power and current wave-domain equations are defined as:

$$\text{Power waves:} \quad b[n] = 2\sqrt{R_p} e[n] - a[n] \quad (1.19)$$

$$\text{Current waves:} \quad b[n] = 2R_p e[n] - a[n]. \quad (1.20)$$

Unlike the resistor, the ideal voltage source cannot be adapted to be used at the leaf of a connection tree since the delay-free directed path between $b[n]$ and $a[n]$ is not parameterized by R_p . The ideal voltage source, therefore, can only be used at the root of a connection tree. Since WDF connection trees can only support one unadapted element, it is advantageous to define the resistive voltage source. The resistive voltage source is comprised of an ideal voltage source $e(t)$ in series with a resistor R . Its characteristic equation can be found using Kirchhoff's voltage law

$$v(t) = e(t) + i(t)R. \quad (1.21)$$

Plugging in (1.3-4), replacing index t with n , and solving for $b[n]$ yields the discrete-time unadapted wave-domain equation

$$b[n] = \frac{R - R_p}{R + R_p} a[n] + \frac{2R_p}{R + R_p} e[n]. \quad (1.22)$$

This equation is now adaptable by setting $R_p = R$; this removes the delay-free directed path from the signal flow diagram and yields the resistive voltage source's adaptive wave equation

$$b[n] = e[n]. \quad (1.23)$$

Like the ideal voltage source, the adapted wave equation for the resistive voltage source changes based on the choice of wave variable

$$\text{Power waves:} \quad b[n] = \sqrt{R_p} e[n] \quad (1.24)$$

$$\text{Current waves:} \quad b[n] = R_p e[n]. \quad (1.25)$$

When creating a WDF structure from a reference circuit that contains an ideal voltage source which must be adapted, it is simple to reformulate it as a voltage source in series with a fictitious 1Ω resistance. The fictitious resistor will not fundamentally change the energetic behavior of the circuit since the resistor is a passive element. Additionally, if a voltage source or battery in the reference circuit is already in series with a resistor, combining the two elements into a resistive voltage source one-port will reduce the number of computations necessary to traverse the tree.

1.5 - Reactive One-Port Derivations

Reactive one-port electronic elements are elements that have a continuous-time derivative in their characteristic equations. The two reactive elements of importance to Virtual Analog modeling are the capacitor and the inductor. To discretize these elements, it is necessary to approximate the derivative in discrete time. One commonly employed

method is to use the trapezoidal rule for integration, which is identical to the Bilinear Transform in the Laplace domain [5]. The Bilinear Transform represents a frequency mapping from the s -plane to the z -plane, which necessarily results in a warping of the continuous frequency range. Other transforms can be chosen to discretize the reactive one-port, including the Backward Euler transform or the Alpha Transform, a generalized transform that allows for a compromise between the Bilinear Transform and Backward Euler method [16].

To derive the wave-domain equation for a reactive one-port, it is necessary to convert its continuous-time Kirchhoff equation in the Laplace domain to the wave-domain, discretize the resulting expression using one of the aforementioned transforms, and then use the inverse z -transform to find the discrete-time wave-domain equation. In the following section, the Bilinear Transform, Backward Euler transform, and Alpha Transform will be formally defined. The process of discretizing a reactive one-port will then be demonstrated with a capacitor using both the bilinear transform and the Alpha Transform.

1.5.1 - Discretization Methods

Each of the discretization methods described in the following section map the $j\Omega$ axis of the s -plane to the z -plane, except for a pre-determined frequency warping. This warping occurs as a result of mapping the infinite s -plane frequency range $\Omega \in [-\infty, \infty]$ to the finite z -plane frequency range $\omega \in [-\pi, \pi]$.

Given a transfer function $H(s)$, the Bilinear Transform (BLT) maps each s value in $H(s)$ to an equivalent discrete-time transfer function $H(z^{-1})$ with

$$s = \frac{2}{T} \frac{1 - z^{-1}}{1 + z^{-1}} \quad (1.26)$$

for sampling period $T = \frac{1}{F_s}$. The warping of the bilinear transform is thus described by

$$\omega = \frac{2}{T} \arctan\left(\Omega \frac{T}{2}\right). \quad (1.27)$$

This indicates that the DC point is mapped from ($s = 0$) to ($z = 1$) and that high frequencies extending to infinity are mapped to $z = -1$.

Backwards Euler discretization can be similarly described by the mapping

$$s = \frac{1 - z^{-1}}{T}. \quad (1.28)$$

This indicates that the DC point is mapped from ($s = 0$) to ($z = 1$) and that high frequencies extending to infinity are mapped to $z = 0$. While less commonly used than the BLT, this method results in better performance with circuits containing nonlinearities or switches [16].

The Alpha Transform makes use of a parameter α to parameterize the warping characteristics of the transform [16]. The mapping is defined as

$$s = \frac{\frac{1 + \alpha}{T} - \frac{1 + \alpha}{T} z^{-1}}{1 + \alpha z^{-1}} \quad (1.29)$$

and maps dc from ($s = 0$) to ($z = 1$) and high frequencies extending to infinity are mapped to ($z = -\alpha$). α is meant to operate in the range $\alpha \in [0,1]$, with special cases at $\alpha = 1$ (BLT) and $\alpha = 0$ (Backward Euler).

1.5.2 - Capacitor

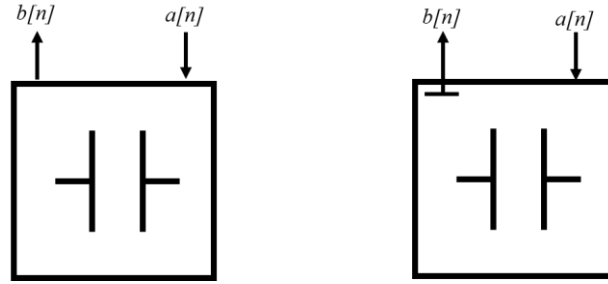


Figure 1.3: Unadapted and Adapted Capacitor One-Ports

A capacitor with capacitance C can be characterized by the differential equation

$$C \frac{dv(t)}{dt} = i(t) \quad (1.30)$$

which can be represented in the Laplace domain as

$$CsV(s) = I(s). \quad (1.31)$$

Plugging (1.3-4) into (1.26) and solving for $B(s)$ yields the wave-domain transfer function

$$H(s) = \frac{B(s)}{A(s)} = \frac{1 - R_p Cs}{1 + R_p Cs}. \quad (1.32)$$

First, choosing the Bilinear Transform to discretize (1.32) results in the discrete-time transfer function

$$H(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{T - 2R_p C + (T + 2R_p C)z^{-1}}{T + 2R_p C + (T - 2R_p C)z^{-1}}. \quad (1.33)$$

Subsequently taking the inverse z -transform of (1.33) and solving for $b[n]$ reveals the unadapted wave-domain equation

$$b[n] = -\frac{T - 2R_p C}{T + 2R_p C} b[n - 1] + \frac{T - 2R_p C}{T + 2R_p C} a[n] + a[n - 1]. \quad (1.34)$$

This equation must be adapted by removing the delay-free path between $b[n]$ and $a[n]$, which is accomplished by setting $R_p = \frac{T}{2C}$. Making this substitution yields the adapted wave-domain equation for a capacitor

$$b[n] = a[n - 1]. \quad (1.35)$$

The adapted equation reveals that a capacitor discretized by the BLT can be represented in a WDF structure by a unit delay. The inductor, as a complement to the capacitor, can be similarly represented by a unit delay and a gain of -1 [1]

$$b[n] = -a[n - 1]. \quad (1.35)$$

If the capacitor is instead discretized using the Alpha Transform, plugging (1.29) into (1.32) yields

$$H(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{(T - 2R_p C(1 + \alpha)) + (T\alpha + 2R_p C(1 + \alpha))z^{-1}}{(T + 2R_p C(1 + \alpha)) + (T\alpha - 2R_p C(1 - \alpha))z^{-1}}. \quad (1.36)$$

Taking the inverse z -transform and solving for $b[n]$ then gives the unadapted wave-domain equation

$$b[n] = -\frac{R_p C(1 + \alpha) - T\alpha}{R_p C(1 + \alpha) + T} b[n - 1] + \frac{-R_p C(1 + \alpha) + T}{R_p C(1 + \alpha) + T} a[n] + \frac{R_p C(1 + \alpha) + T\alpha}{R_p C(1 + \alpha) + T} a[n - 1], \quad (1.36)$$

and adapting (1.34) with $R_p = \frac{T}{C(1 + \alpha)}$ finally yields the adapted wave-domain equation

$$b[n] = \frac{1 - \alpha}{2} b[n - 1] + \frac{1 + \alpha}{2} a[n - 1]. \quad (1.37)$$

Unlike the BLT discretization, the Alpha Transform discretization of a capacitor requires two state updates. The Alpha Transform discretization allows for greater flexibility in tuning the response of the circuit at the cost of added computational complexity.

Special considerations must be made in the case where a time-varying capacitance is modeled. In this case, it is necessary to discretize the behavior of the capacitance as well as the time derivative of the voltage. In [17], Bogason and Werner propose a generalized time-varying model of a capacitor

$$i(t) = C^{1-\lambda}(t) \frac{d}{dt} (C^\lambda(t) v(t)) \quad (1.38)$$

where the parameter λ allows for the simulation of different time-varying models, with ($\lambda = 1$) representing the circuit-theoretic model, ($\lambda = 0.5$) a Fettweis model [18], and ($\lambda = 0$) the traditional model. Discretizing (1.38) with the Alpha Transform and solving for $b[n]$ as before results in unadapted and adapted equations (32) and (33) from [17].

1.6 - Two-Port Derivations

Two-port electronic elements are characterized by a system of two equations relating voltage and current in the Kirchhoff domain. It is necessary to derive the scattering relationship between wave vectors \mathbf{b} and \mathbf{a} given port voltages \mathbf{v} and port currents \mathbf{i} related by

$$\mathbf{X}\mathbf{v} + \mathbf{Y}\mathbf{i} = \mathbf{0} \quad (1.39)$$

for gains \mathbf{X} and \mathbf{Y} . A two-port's scattering matrix \mathbf{S} is thus defined as the square matrix that resolves the wave-domain relationship

$$\mathbf{b} = \mathbf{S}\mathbf{a}, \quad (1.39)$$

and can be found by plugging in the voltage wave definition and solving as before.

In the following section, the wave-domain equations for the two-port series adaptor are derived. For a full list of two-port derivations, refer to [1].

1.6.1 - Two-Port Series Adaptor (Inverter)

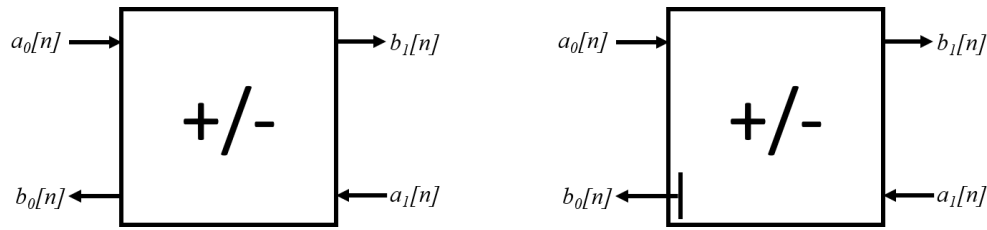


Figure 1.5: Unadapted and Adapted Inverter Two-ports

When two ports are connected in series with one another, the currents across their ports are equal and the voltages across their ports are exactly opposite

$$v_0(t) = -v_1(t), \quad i_0(t) = i_1(t) \quad (1.40)$$

The two-port series adaptor is mainly used as a voltage polarity inverter, hence its alternate naming convention [19].

Plugging (1.3-4) into (1.40), converting continuous-time indices t to discrete-time indices n , and solving for $\mathbf{b} = [b_0, b_1]^T$ reveals the unadapted wave-domain equation

$$\begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} -\frac{R_0 - R_1}{R_0 + R_1} & \frac{-2R_0}{R_0 + R_1} \\ \frac{-2R_1}{R_0 + R_1} & \frac{R_0 - R_1}{R_0 + R_1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}. \quad (1.41)$$

Either port 0 or port 1 can be adapted, depending on which port is chosen to face the root of the tree. Generally, adapting a port p of an N -port described by scattering matrix \mathbf{S} of size $(p \times p)$ corresponds to choosing a value for R_p which sets matrix entry $s_{pp} = 0$. To adapt port 0 of the inverter, a value must be chosen for R_0 such that $-\frac{R_0 - R_1}{R_0 + R_1} = 0$. Setting $R_0 = R_1$ accomplishes this and yields the inverter's adapted wave-domain equation

$$\begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}. \quad (1.42)$$

For implementation in software, it is more efficient to convert (1.42) to the form of a system of linear equations than to solve the matrix multiplication described by (1.39) directly. It is simple to rewrite the adapted wave equation as

$$b_0 = -a_1, \quad b_1 = -a_0. \quad (1.43)$$

1.7 - Three-Port Derivations

A three-port adaptor is characterized by a system of equations relating the voltages and currents of its three constituent ports. The most important three-port adaptors are those which represent series and parallel connections between elements. Generally, the wave-domain equations of a three-port adaptor can be derived using the same process as a two-port adaptor. However, in [5], Fettweis derives wave-domain equations for the series and parallel adaptors which minimize the number of required adds and multiplies. This realization is more useful for software implementation, especially on embedded systems

where computational efficiency is a greater concern. In this section, the wave-domain equations for both the series and parallel adaptor will be derived using Fettweis's derivation method.

1.7.1 - Series Adaptor

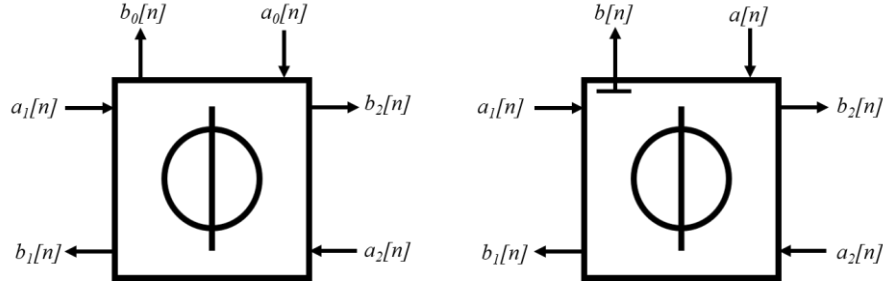


Figure 1.6: Unadapted and Adapted Series Three-ports

Like the two-port case, three-ports connected in series are characterized by equal currents and voltages which sum to zero

$$i_0(t) = i_1(t) = i_2(t), \quad v_0(t) + v_1(t) + v_2(t) = 0 \quad (1.44)$$

Substituting (1.3-4) into (1.44), exchanging t for n , and solving for $b_p[n]$ for a given port $p = 0, 1, 2$ yields the unadapted wave-domain equations

$$b_p[n] = a_p[n] - \gamma_p(a_0[n] + a_1[n] + a_2[n]) \quad (1.45)$$

where

$$\gamma_p = \frac{2R_p}{R_0 + R_1 + R_2} \quad (1.46)$$

and

$$\gamma_0 + \gamma_1 + \gamma_2 = 2. \quad (1.47)$$

Fettweis notes that the number of computations can be reduced by choosing one port to be the *dependent* port and eliminating its corresponding coefficient. This process is not to be confused with adapting the port; the dependency of a port is not indicative of its relationship to any other element or port in a connection tree. If port 2 is chosen as the dependent port, (1.45) may be rewritten as

$$a_d = a_0[n] + a_1[n] + a_2[n] \quad (1.48a)$$

$$b_p[n] = a_p[n] - \gamma_p a_d, \quad p = 0, 1 \quad (1.48b)$$

$$b_2[n] = -(b_0[n] + b_1[n] + a_d) \quad (1.48c)$$

which requires only two multiplies and six additions, a significant improvement over the typical n^2 multiplications it would cost to solve the linear relationship $\mathbf{b} = \mathbf{S}\mathbf{a}$.

To adapt port 0 of the series adaptor it is sufficient to set $\gamma_0 = 1$; this renders the port reflection-free and guarantees that there is no delay-free path between $b_0[n]$ and $a_0[n]$. Setting $\gamma_0 = 1$ additionally yields

$$R_0 = R_1 + R_2 \quad (1.49a)$$

$$\gamma_p = \frac{R_p}{R_0}, \quad p = 1, 2. \quad (1.49b)$$

Intuitively, equation (1.49a) confirms that the impedance of the adapted port matches the series combination of the other two ports. Keeping port 2 as the dependent port and choosing to adapt port 0 produces

$$b_0[n] = -(a_1[n] + a_2[n]), \quad a_d = a_0[n] - b_0[n] \quad (1.50a)$$

$$b_1[n] = a_1[n] - \gamma_1 a_d \quad (1.50b)$$

$$b_2[n] = -(b_0[n] + b_1[n] + a_0[n]). \quad (1.50c)$$

This substitution reduces the complexity to one multiply and four additions.

1.7.2 - Parallel Adaptor

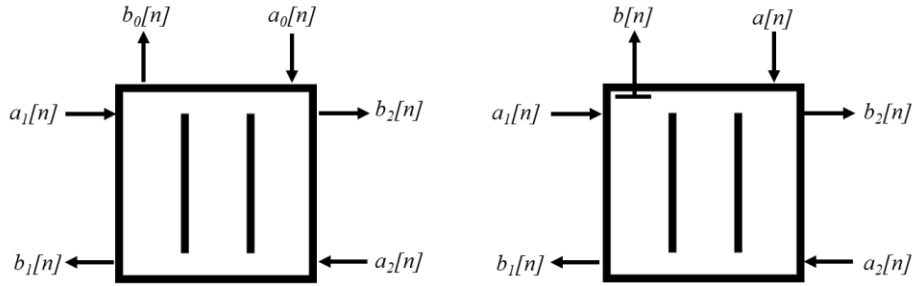


Figure 1.7: Unadapted and Adapted Parallel Three-ports

Three ports connected in parallel have equal voltages across their ports and currents that add to zero

$$v_0 = v_1 = v_2, \quad i_0 + i_1 + i_2 = 0 \quad (1.51)$$

Substituting (1.3-4) into (1.51), exchanging t for n , and solving for $b_p[n]$ for a given port $p = 0, 1, 2$ yields the unadapted wave-domain equations

$$b_p[n] = (\gamma_0 a_0[n] + \gamma_1 a_1[n] + \gamma_2 a_2[n]) - a_p[n] \quad (1.52)$$

where

$$\gamma_p = \frac{2G_p}{G_0 + G_1 + G_2}, \quad (1.53)$$

$G = \frac{1}{R}$ represents the conductance at a given port, and

$$\gamma_0 + \gamma_1 + \gamma_2 = 2. \quad (1.54)$$

Just as with the series adaptor, choosing port 2 as the dependent port allows for (1.52) to be rewritten to reduce the computation down to two multiplies and six additions.

$$b_2[n] = (1 - \gamma_0 - \gamma_1)a_2[n] - \gamma_0 a_0[n] - \gamma_1 a_1[n] \quad (1.55a)$$

$$b_p[n] = b_2[n] + (a_2[n] - a_p[n]), \quad p = 0, 1 \quad (1.55b)$$

To adapt port 0 it is again sufficient to set $\gamma_0 = 1$, yielding

$$G_0 = G_1 + G_2 = \frac{R_1 R_2}{R_1 + R_2} \quad (1.56a)$$

$$\gamma_p = \frac{G_p}{G_0}, \quad p = 1, 2. \quad (1.56b)$$

Equation (1.56a) confirms that the adapted port matches the impedance of the parallel combination of ports 1 and 2. Now, designating port 2 as dependent and adapting port 0 reveals the final adapted wave-domain equations for a parallel adaptor [5]

$$b_d = \gamma_1(a_1[n] + a_2[n]) \quad (1.57a)$$

$$b_0[n] = b_d + a_2[n] \quad (1.57b)$$

$$b_2[n] = b_d + a_0[n] \quad (1.57c)$$

$$b_1[n] = b_2[n] + a_2[n] - a_1[n]. \quad (1.57d)$$

1.8 - Nonlinear One-Port Derivation

Nonlinear electrical elements are characterized by a single constituent equation that defines its port currents as a function of its voltages or vice versa. Generally, nonlinear elements cannot be adapted and must act as the root of a WDF connection tree. Per [1], this is due to the necessarily instantaneous relationship between the nonlinearity's port

resistance and incident wave. This renders the relationship between $b[n]$ and $a[n]$ unsolvable using the techniques described in the previous sections.

The most important one-port nonlinearity in audio is the diode. Diodes play a key role in countless musical effects processors including distortion pedals [20] and envelope followers [12]. Several methods have been devised to model diodes in the Wave Digital Filter context ([1], [21]–[24]) including piecewise linear and iterative models. For the sake of brevity, this thesis will only consider the implemented model, which analytically solves the nonlinearity. In the following sections, the wave-domain equations for a single diode as well as two anti-parallel diodes are derived.

1.8.1 – Diode

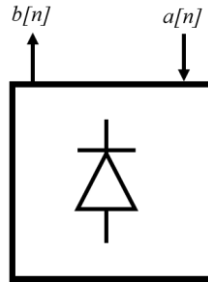


Figure 1.8: Unadapted Diode One-port

A diode can be characterized by the Shockley large-signal model, which relates the current through a PN junction diode to its port voltage

$$i(t) = I_s \left(e^{\frac{v(t)}{n_D V_T}} - 1 \right) \quad (1.58)$$

where I_s represents the diode's saturation current (typically 2.52 nA), V_T represents the thermal voltage (typically 25.86 mV) and n_D represents the diode ideality factor. Plugging (1.3-4) into (1.58) and substituting t for n yields

$$e^{\frac{a[n]+b[n]}{2n_D V_T}} = \frac{a[n] - b[n]}{2R_p I_s} + 1. \quad (1.59)$$

This equation can be solved for $b[n]$ using the Lambert W function [21]. The Lambert W function can be used to solve exponential equations of the form

$$(A + Bx)e^{Cx} = D \quad (1.60)$$

for x with

$$x = \frac{1}{C} \mathcal{W} \left(\frac{CD}{B} e^{\frac{AC}{B}} \right) - \frac{A}{B}. \quad (1.61)$$

Substituting the proper coefficients from (1.59) into (1.61) reveals the diode's wave-domain equation

$$b[n] = a[n] + 2R_p I_s - 2n_D V_T W \left(\frac{R_p I_s}{n_D V_T} e^{\frac{R_p I_s + a[n]}{n_D V_T}} \right). \quad (1.62)$$

1.8.2 - Diode Pair

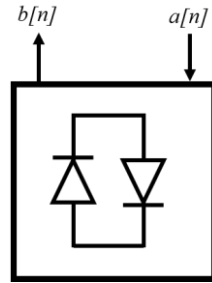


Figure 1.9: Unadapted Diode Pair One-port

The solution for a single diode can be used to approximate the behavior of a pair of anti-parallel diodes as a nonlinear one-port element [21]. The voltage-current behavior of

a diode pair can be characterized as the application of two instances of Shockley's large-signal model

$$i(t) = I_s \left(e^{\frac{v(t)}{n_D V_T}} - e^{\frac{-v(t)}{n_D V_T}} \right). \quad (1.63)$$

Plugging (1.3-4) into (1.63), substituting t for n , and solving for $b[n]$ using the Lambert W equation yields the wave-domain equation for a diode pair

$$b[n] = \text{sgn}(a[n]) \left(|a[n]| + 2R_p I_s - 2n_D V_T W \left(\frac{R_p I_s}{n_D V_T} e^{\frac{R_p I_s + |a[n]|}{n_D V_T}} \right) \right) \quad (1.64)$$

where $\text{sgn}(x)$ represents the signum function [20]. A computationally efficient approximation of the Lambert function as described in [23] was implemented for use on the SHARC Audio module.

1.8.3 – Antiderivative Antialiasing of Diode Models

When a band-limited signal is processed by a nonlinear function such as (1.58), the signal's bandwidth is expanded and frequency components above the Nyquist frequency are mirrored into the signal baseband. This causes aliasing distortion, which is often inharmonic and displeasing to the ear [25], [26]. Aliasing can be reduced by oversampling the input with a high oversampling factor O . This would increase the number of operations needed to process the tree by a factor of O , however, and given the SHARC's limited processing power this would be undesirable. A method that reduces aliasing with minimal oversampling is preferred for embedded processors.

Antiderivative Antialiasing (ADAA) reduces aliasing in memoryless nonlinearities by approximating the nonlinear function in terms of its antiderivatives. This is accomplished by approximating the input signal as a continuous-time piecewise linear function, applying the nonlinearity to the input signal, then convolving the result with the continuous-time impulse response of a lowpass filter. By approximating the nonlinearity with the p th antiderivative, a p th-order ADAA function can be derived; higher order ADAA results in fewer aliasing artifacts at low oversampling factors at the cost of $p/2$ samples of additional delay added to the system [25]–[27].

Given a continuous nonlinear function f with first-order antiderivative F_1 , a nonlinear scattering equation $b = f(a)$ can be approximated by first-order ADAA with

$$f_1(a[n], a[n-1]) = \begin{cases} \frac{F_1(a[n]) - F_1(a[n-1])}{a[n] - a[n-1]} & \text{if } a[n] \neq a[n-1] \\ f\left(\frac{a[n] + a[n-1]}{2}\right) & \text{if } a[n] \approx a[n-1] \end{cases} \quad (1.65)$$

where f_1 introduces a half sample delay. It is necessary to synchronize the samples delayed by f_1 with the rest of the samples in the connection tree by applying a half-sample delay filter to the remaining outputs in the connection tree using the implementation procedure described in [25].

Explicit WDF ADAA models of a diode and a diode pair are found by finding the first-order antiderivatives of equations (1.62) and (1.64) respectively. Solving the antiderivative of (1.62) for a WDF diode yields

$$F_1(a) = \frac{a^2}{2} + 2R_p I_s a - n_D^2 V_t^2 \omega(\phi(a)) (2 + \omega(\phi(a))) \quad (1.66)$$

with

$$\phi(a) = \frac{a + R_p I_s}{n_D V_t} + \log \left(\frac{R_p I_s}{n_D V_t} \right). \quad (1.67)$$

Similarly, the WDF ADAA model of a diode pair is found by solving the antiderivative of (1.64), yielding

$$F_1(a) = \frac{a^2}{2} + 2R_p I_s |a| - n_D^2 V_t^2 \omega(\phi(|a|)) (2 + \omega(\phi(|a|))) \quad (1.68)$$

with $\phi(a)$ as above.

1.9 - SPQR Trees and R-type Adaptors

The BCT formulation of a WDF structure will always be realizable for circuits comprised solely of serial and parallel connections; this is not the case for circuits with complex topologies, such as lattice or Twin-T networks. In rare cases, it is possible to reformulate certain networks, such as the Jaumann structure, into a specialized two-port adaptor ([1], [28]), but these structures do not occur very frequently in audio circuits and are thus outside the scope of this thesis. In keeping with the modular nature of Wave Digital Filters, a generalized method of representing complex topologies such that they can be included in a connection tree is desired.

Fränken et al. proposed a method to accomplish this task based on a generalized formulation of the *SPQR* tree concept from graph theory [11], [29]. An SPQR tree is derived from the decomposition of a biconnected graph with respect to its triconnected

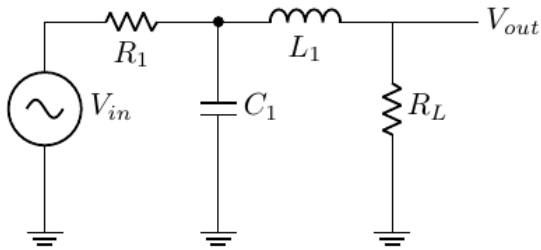
components [30]. The tree can be derived algorithmically from the connected graph by finding its *split components*, each of which corresponds to an \mathcal{S} , \mathcal{P} , or \mathcal{R} node. These nodes in turn correspond to series adaptors (\mathcal{S}), parallel adaptors (\mathcal{P}), and *Rigid* or *R-type* adaptors (\mathcal{R}). In the graph theory context, *R-type* adaptors represent a skeleton of a graph that is triconnected and cannot be further decomposed into parallel connections (three or more edges connecting two nodes) or series connections (two nodes connected in a triangular formation) [11]. In the Wave Digital Filter context, this translates to any complex topology that cannot be represented by series or parallel adaptors.

In sections 1.9.1-3, the process of deriving a circuit's SPQR tree will be described by way of three motivational examples: a two-pole low pass filter, a tone control circuit, and the MXR Distortion+ guitar distortion pedal. Next, section 1.9.4 will discuss the process of systematically deriving an *R-type* adaptor's scattering behavior. Section 1.9.5 will outline how *R-type* adaptors can be used to represent circuits with operational amplifiers, and a complete derivation of the MXR Distortion+ WDF structure will be provided as a summarizing example.

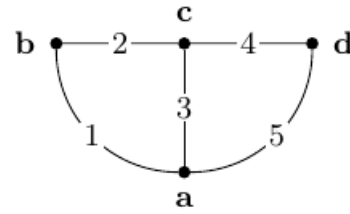
1.9.1 – SPQR Tree Derivation for Butterworth Low Pass Filter

Consider the passive two-pole Butterworth low pass filter circuit shown in Figure 1.10a. While this circuit could be emulated more simply by deriving its transfer function, deriving its SPQR tree serves as a good intuition-building example since its topology is solely comprised of series and parallel connections. To derive the WDF adaptor structure for the low pass circuit, it is first necessary to derive the connected graph representing its

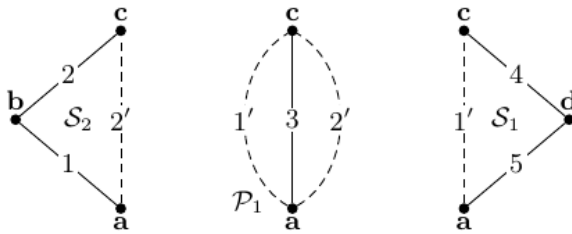
topology as shown in Figure 1.10b. Each lettered node in the graph corresponds to a circuit node, while each numbered edge corresponds to a port. The graph is then decomposed into its split components, shown in Figure 1.10c, using the graph separation algorithm from [11]. This process reveals two series connections and one parallel connection, which can be verified by inspection of the circuit. Finally, one element is designated as the root of the tree and the rest of the tree can be formed beneath it (Figure 1.10d). The ideal voltage source is chosen as the root since it is the only element of this circuit that cannot be adapted. The corresponding WDF adaptor structure is equivalent to the SPQR tree and is shown in Figure 1.10e.



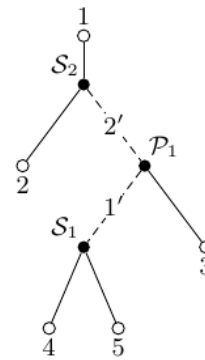
(a) Butterworth Lowpass Circuit



(b) Connected Graph



(c) Split Component Derivation



(d) SPQR Tree

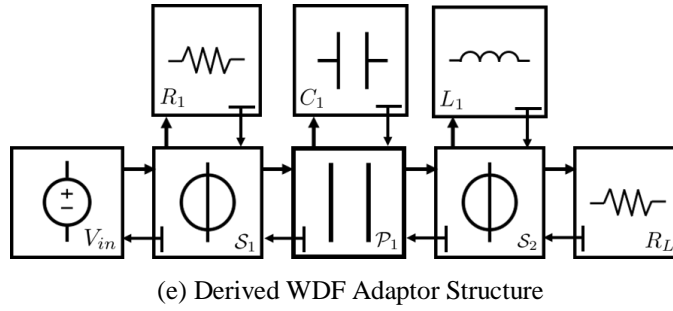
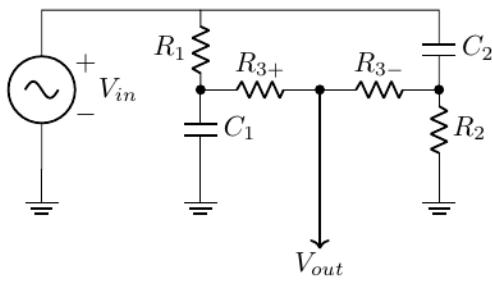


Figure 1.10: WDF Derivation of Butterworth Low Pass Filter Circuit

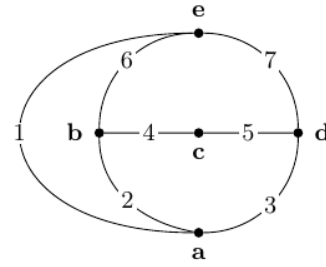
1.9.2 – SPQR Tree Derivation for Single-Pot Tone Control Circuit

Next, consider the single-pot tone control circuit (Figure 1.11a) as described in [43]. Unlike the low pass filter, this circuit is a more obvious candidate for emulation via Wave Digital Filter because its transfer function is not simple to derive using Kirchhoff's laws. Note that resistors R_{3+} and R_{3-} represent two sides of a center-tapped potentiometer.

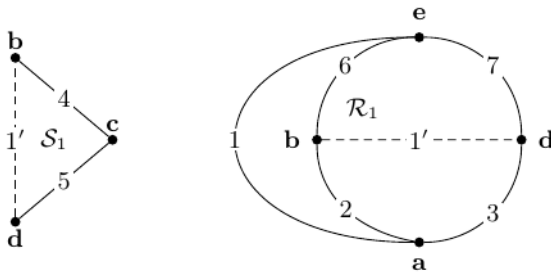
To derive its WDF adaptor structure, the graph shown in Figure 1.11b is derived by inspection. The graph is then decomposed into its split components as shown in Figure 1.11c. This circuit only has one series connection; the rest of the components are connected in such a way that they cannot be further decomposed and are therefore relegated for conclusion inside an R -type adaptor. The circuit's SPQR tree is finally formed (Figure 1.11d), again choosing the ideal voltage source as the root of the tree. The corresponding WDF adaptor structure is shown in Figure 1.11e.



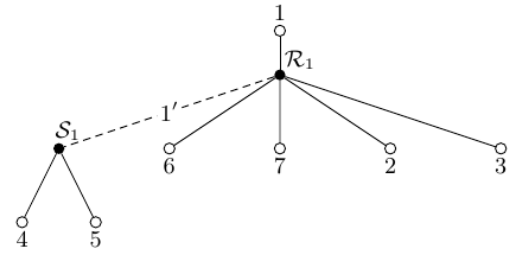
(a): Passive Tone Control Circuit



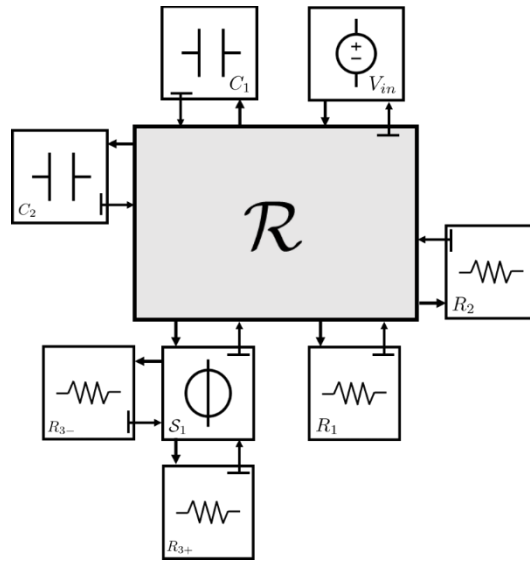
(b) Connected Graph



(c) Split Components



(d) SPQR Tree



(e) Derived WDF Adaptor Structure

Figure 1.11: WDF Derivation of Passive Tone Control Circuit

1.9.3 - SPQR Tree Derivation of MXR Distortion+ Pedal

As a final motivating example, consider the MXR Distortion+ guitar effects circuit shown in Figure 1.12 [20]. This circuit consists of an op-amp amplification and filtering stage followed by a diode clipper in parallel with a filtering capacitor. Potentiometer R_{dist} controls the voltage gain of the amplifier, while R_{out} controls the circuit's output volume.

Outside of the SPQR context, Paiva et al. showed that op-amps in differential amplifiers could be modeled in the Wave Digital context without the use of R -type adaptors, but only under the assumption of ideality [21]. Their model will not be considered in this thesis because it cannot account for more complex circuit topologies, such as state-variable or Sallen-Key filters which are common in audio devices [1].

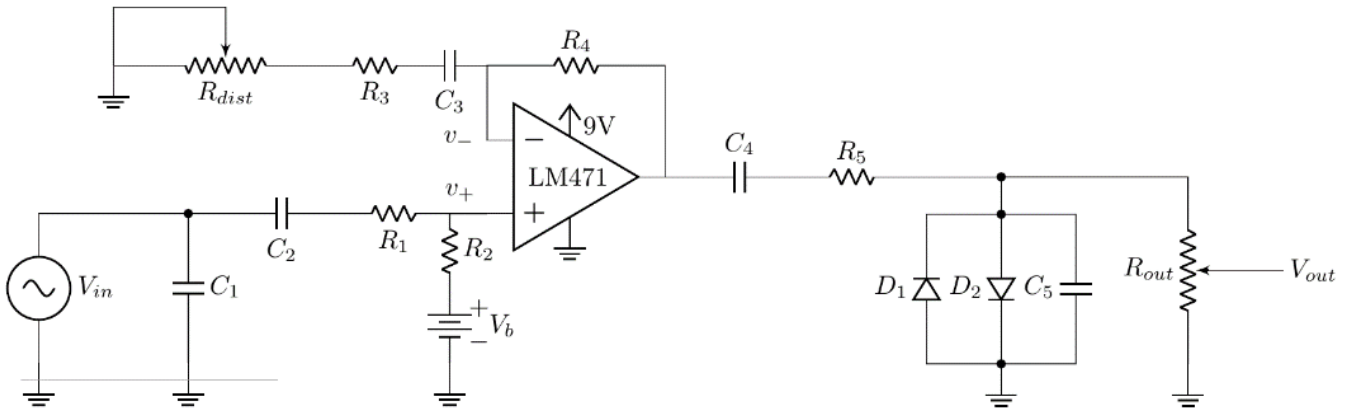
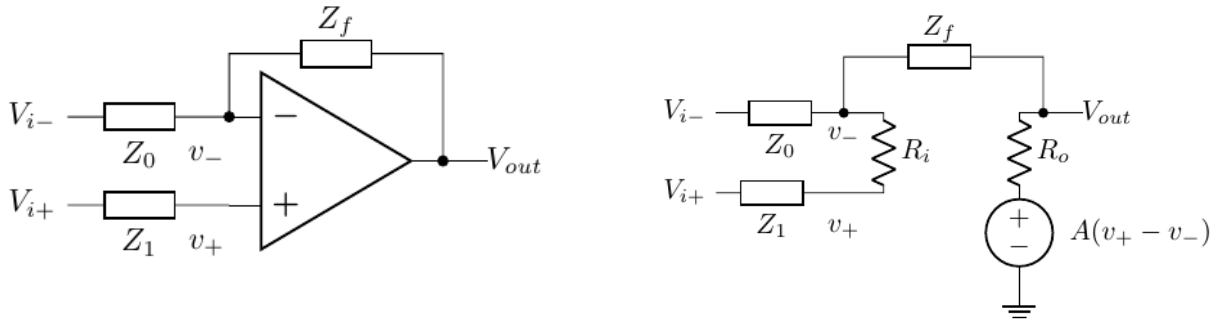
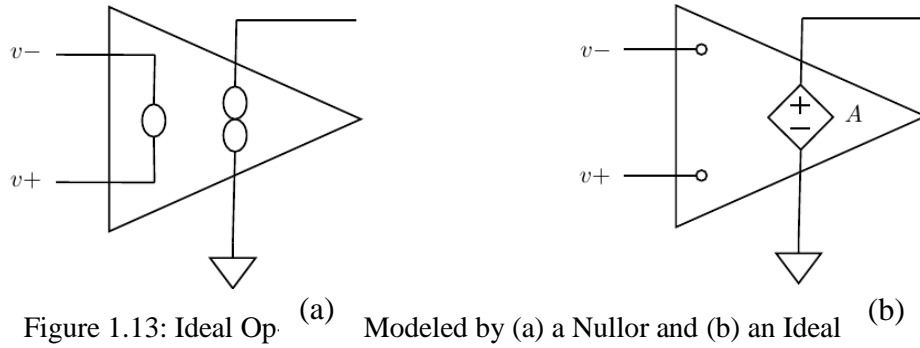


Figure 1.12: MXR Distortion+ Circuit

To emulate a circuit with an operational amplifier using R -type adaptors, the op-amp can be modeled either as ideal or with a macromodel. An ideal op-amp can be represented by a *nullor* (Figure 1.13a), a purely theoretical element that always has zero input voltage and current as well as infinite output current, voltage, and transconductance

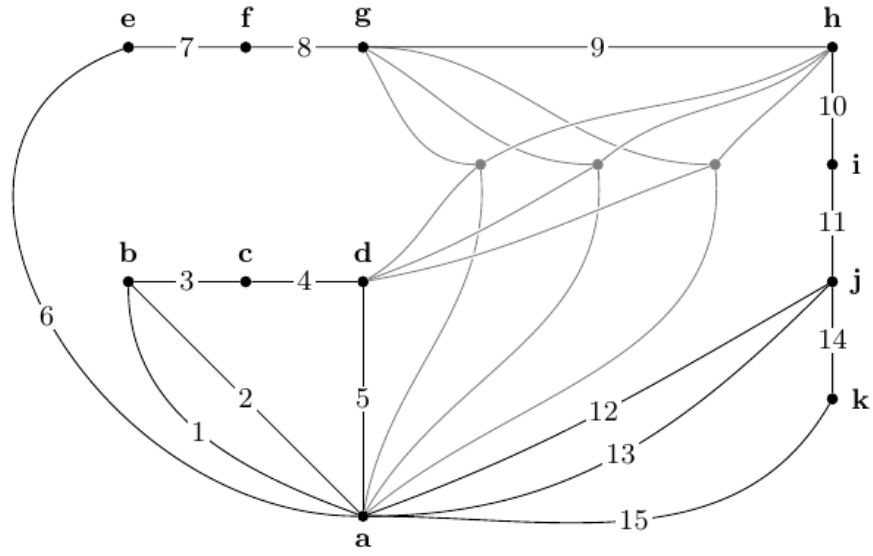
gain [1]. The nullor is a combination of two other theoretical elements: the nullator, which is characterized by zero port voltage and current and represents the input to the op-amp, and the norator, which is characterized by no restrictions on either port voltage or current and represents the output. It is also possible to model an ideal op-amp using a voltage-controlled voltage source (VCVS) with a large open-loop gain (Figure 1.13b).



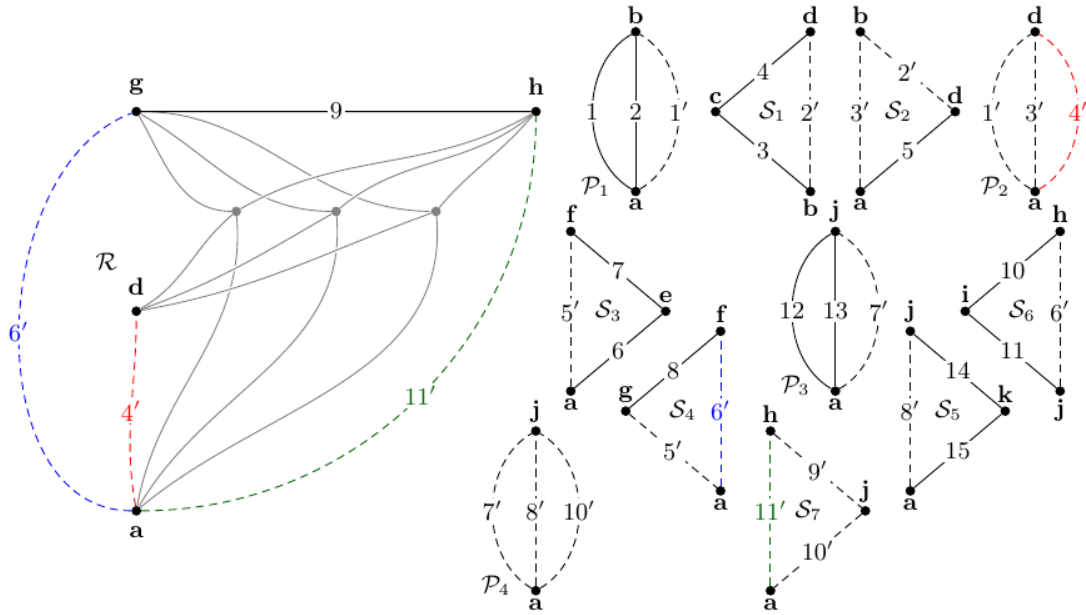
Given that the non-ideal characteristics of an op-amp contribute strongly to the behavior of a circuit, it is often preferred to use a macromodel to model an op-amp. Op-amp macromodels allow for the emulation of complex behaviors, including gain-bandwidth effects, slew rate limiting, and transfer function nonlinearities ([31], [32]). For the sake of simplicity, a simple macromodel consisting of a VCVS with large open-loop gain and input and output resistances will be considered (Figure 1.14).

When creating a graph for circuits containing op-amps or other multiport elements, such as transformers and controlled sources, it is necessary to pre-process the graph with *replacement graphs*. Replacement graphs ensure that the ports of the multiport element will not be split or misconstrued as a series or parallel connection in the separation process. For a three-terminal element such as an op-amp, Fränken et al. proposed a replacement graph consisting of three internal nodes which connect to each node adjacent to the op-amp via fictitious edges [11]. This structure sufficiently ensures that the graph separation algorithm does not separate the ports of the op-amp. A multiport element wholly represented within an R -type adaptor is called *absorbed*.

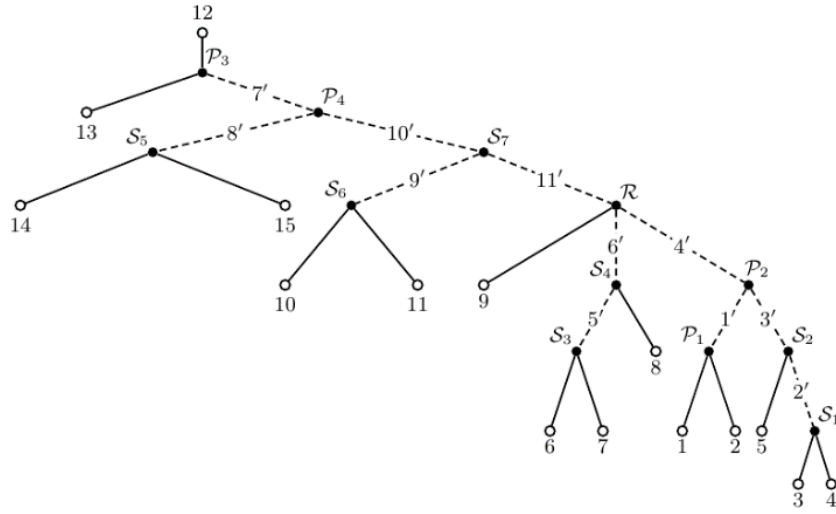
The graph representing the MXR Distortion+ is shown in Figure 1.15a. The gray unlabeled nodes and edges represent the internal nodes and edges of the replacement graph representing the op-amp. The graph separation process, the resulting SPQR tree, and the corresponding WDF adaptor structure are shown in Figures 1.15b-d. The diode pair must be chosen as the root of this tree since it is the sole nonlinear element in the circuit.



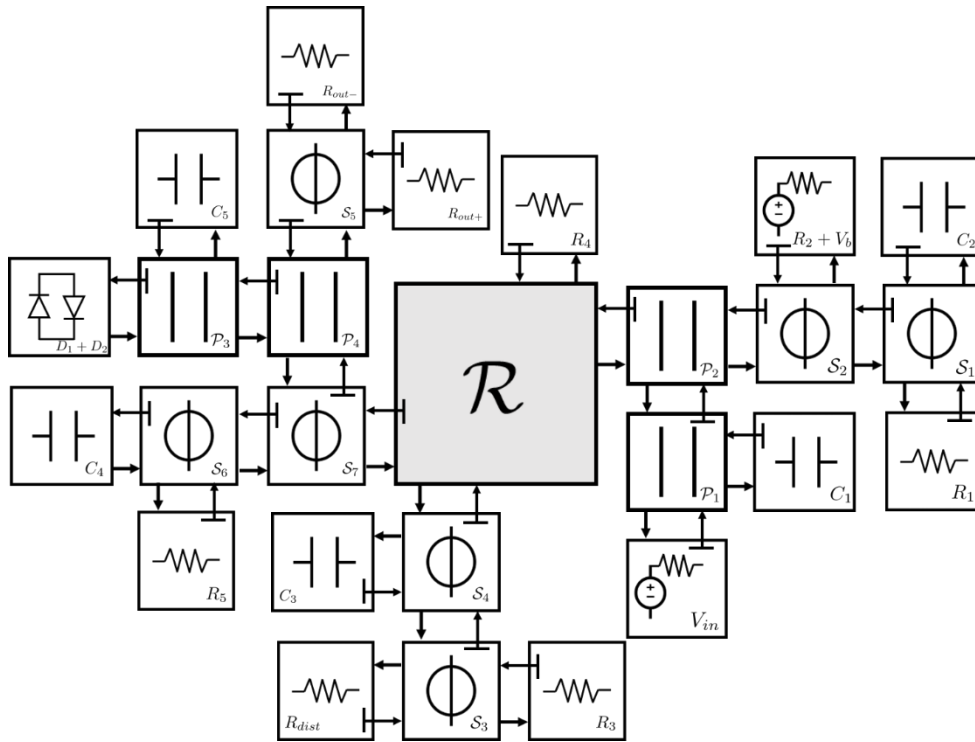
(a) MXR Distortion+ Connected Graph



(b) MXR Distortion+ Graph Separation Process



(c) MXR Distortion+ SPQR Tree



(d) MXR Distortion+ WDF Adaptor Structure

Figure 1.15: MXR Distortion+ Wave Digital Filter Derivation

1.9.4 – Scattering Behavior of R-type Adaptors

For an R -type adaptor to be realizable in a connection tree, it needs to solve the scattering relationship between its incident and reflected waves as per equation (1.39). Unlike the adaptors investigated in sections 1.6-7, the internal topology of an R -type adaptor is unique to the circuit, which means the scattering matrix \mathbf{S} needs to be derived on a case-by-case basis.

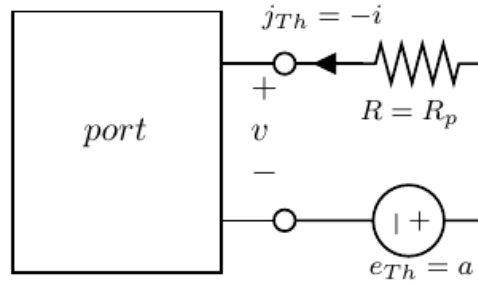


Figure 1.16: Thévenin Port Equivalent

In [2], Werner et al. derive a method to systematically define the scattering behavior for any R -port which may include absorbed multiport elements. The approach uses Modified Nodal Analysis (MNA) techniques to represent a circuit equivalent to the complex topology that is included within the adaptor. They start by representing the voltage and current across each port attached to the R -type adaptor with an equivalent Thévenin source (Figure 1.16) with vectors of voltage source voltages \mathbf{e}_{Th} , currents \mathbf{j}_{Th} , and resistances \mathbf{R}_{Th} arranged in a diagonal matrix. To find the relationship between the Thévenin values and the port's wave variables, equations (1.3-4) are plugged into the Thévenin port's constituent equation

$$\mathbf{R}_{Th}^{-1}(\mathbf{e}_{Th} - \mathbf{v}) = \mathbf{i} \quad (1.69)$$

yielding

$$\mathbf{e}_{Th} = \frac{1}{2}(\mathbf{R}_{Th}\mathbf{R}_p^{-1})\mathbf{a} + \frac{1}{2}(\mathbf{R}_{Th}\mathbf{R}_p^{-1})\mathbf{b}. \quad (1.70)$$

Setting $\mathbf{R}_{Th} = \mathbf{R}_p$ removes \mathbf{b} from (1.66), revealing the relationship

$$\mathbf{e}_{Th} = \mathbf{a}. \quad (1.71)$$

It follows from inspection that

$$\mathbf{j}_{Th} = -\mathbf{i}. \quad (1.72)$$

Werner then combines equations (1.1-4) to derive a generalized scattering relationship between \mathbf{b} and \mathbf{a} . This process yields

$$\mathbf{b} = \mathbf{a} - 2\mathbf{R}_p\mathbf{i} \quad (1.73)$$

which is not a function of \mathbf{a} . It is thus necessary to systematically find a matrix such that

$$\mathbf{i} = \mathbf{Q}\mathbf{a} \quad (1.74)$$

for any R -type adaptor.

Modified Nodal Analysis allows for the systematic representation of a system of circuit equations via MNA matrix \mathbf{X}

$$\underbrace{\begin{bmatrix} \mathbf{Y} & \mathbf{A} \\ \mathbf{B} & \mathbf{D} \end{bmatrix}}_{\mathbf{X}} \begin{bmatrix} \mathbf{v}_n \\ \mathbf{j} \end{bmatrix} = \begin{bmatrix} \mathbf{i}_s \\ \mathbf{e} \end{bmatrix} \quad (1.75)$$

which defines the relationship between node voltages \mathbf{v}_n , node currents \mathbf{j} , voltage source values \mathbf{e} , and current source values \mathbf{i}_s using partitions \mathbf{Y} , \mathbf{A} , \mathbf{B} , and \mathbf{D} . [33]. For adaptors with absorbed linear multiport elements, (1.75) must be adjusted to separate Thévenin voltages and currents from any extraneous voltages and currents in the circuit topology

$$\underbrace{\begin{bmatrix} \mathbf{Y} & \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{B}_1 & \mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{B}_2 & \mathbf{D}_{21} & \mathbf{D}_{22} \end{bmatrix}}_{\mathbf{X}} \begin{bmatrix} \mathbf{v}_n \\ -\mathbf{j}_{Th} \\ \mathbf{j}_{etc} \end{bmatrix} = \begin{bmatrix} \mathbf{i}_s \\ \mathbf{e}_{Th} \\ \mathbf{0} \end{bmatrix}. \quad (1.76)$$

Substituting (1.71-72) into (1.76) and solving for \mathbf{i} yields a linear relationship between \mathbf{i} and \mathbf{a}

$$\mathbf{i} = \left(-[\mathbf{0} \quad \mathbf{I} \quad \mathbf{0}] \mathbf{X}^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \\ \mathbf{0} \end{bmatrix} \right) \mathbf{a}. \quad (1.77)$$

Clearly, the expression inside the parentheses corresponds to matrix \mathbf{Q} , thus plugging (1.77) into (1.74) reveals the expression for \mathbf{S} that solves the $\mathbf{b} = \mathbf{S}\mathbf{a}$

$$\mathbf{S} = \mathbf{I} + 2\mathbf{R}_p[\mathbf{0} \quad \mathbf{I} \quad \mathbf{0}] \mathbf{X}^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \\ \mathbf{0} \end{bmatrix}. \quad (1.78)$$

The last step of this method is to populate \mathbf{X} with the proper node voltages, currents, and admittance values. This is easily accomplished using the *element stamp* method [33], [34]. Each node in the circuit is assigned a numbered index, and the contribution of each element is added to \mathbf{X} according to the matrix specified by each element stamp [2]. Relevant element stamps are shown in Table 1, while a more complete list of stamps is given in [35].

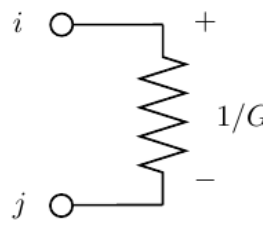
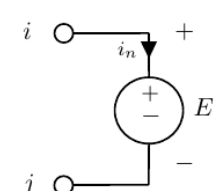
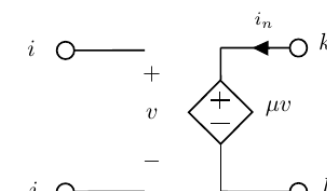
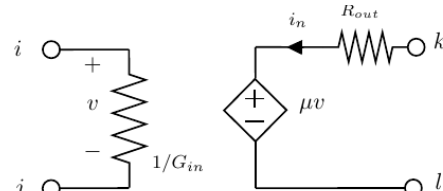
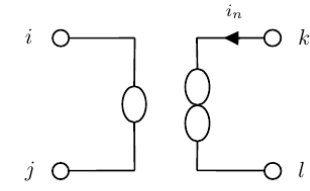
Resistor	Voltage Source	VCVS
 $\begin{matrix} i & j \\ \begin{bmatrix} G & -G \\ -G & G \end{bmatrix} \end{matrix}$	 $\begin{matrix} i & j & n & \text{source} \\ \begin{bmatrix} & & 1 \\ 1 & -1 & -1 \end{bmatrix} & \begin{bmatrix} E \end{bmatrix} \end{matrix}$	 $\begin{matrix} i & j & k & l & n \\ \begin{bmatrix} & & & 1 \\ & & & -1 \\ -\mu & \mu & 1 & -1 \end{bmatrix} \end{matrix}$
Resistive VCVS Macromodel		Nullor
 $\begin{matrix} i & j & k & l & n \\ \begin{bmatrix} -G_{in} & G_{in} & & & \\ G_{in} & -G_{in} & & & \\ & & & 1 & \\ & & & -1 & \\ -\mu & \mu & 1 & -1 & R_{out} \end{bmatrix} \end{matrix}$	 $\begin{matrix} i & j & k & l & n \\ \begin{bmatrix} & & & & \\ & & & & \\ & & & 1 & \\ & & & -1 & \\ 1 & -1 & & & \end{bmatrix} \end{matrix}$	

Table 1: List of Relevant MNA Element Stamps

Once \mathbf{X} is populated, it is necessary to remove one node from the circuit since \mathbf{X} represents a series of KCL equations and there is always one less KCL equation than the number of nodes in a circuit [2]. Removing the so-called *datum* node is equivalent to removing the row and column corresponding to that node from \mathbf{X} . Once the datum node is removed, \mathbf{X} is suitable for inclusion in (1.78).

It should be noted that the unadapted and adapted wave equations for an R -type adaptor are identical. To adapt port n of an R -type adaptor, an expression for the port resistance R_n is derived such that its corresponding scattering matrix entry $s_{nn} = 0$.

In Chapter 3, a novel MATLAB script that can derive the scattering matrix of any R -type adaptor is presented. *R-Solver* streamlines the derivation process by filling in the MNA matrix \mathbf{X} from an LTSpice style circuit netlist.

1.9.5 – MXR Distortion+ R-Type Adaptor Derivation

To find the scattering behavior of the R -type adaptor derived in Figure 1.15b, each attached port is replaced with its Thévenin equivalent port, and each node is labeled as shown in Figure 1.17. Nodes 1-3 are absorbed within the R -type adaptor, while nodes 4-8 are added externally due to the Thévenin equivalent port representation.

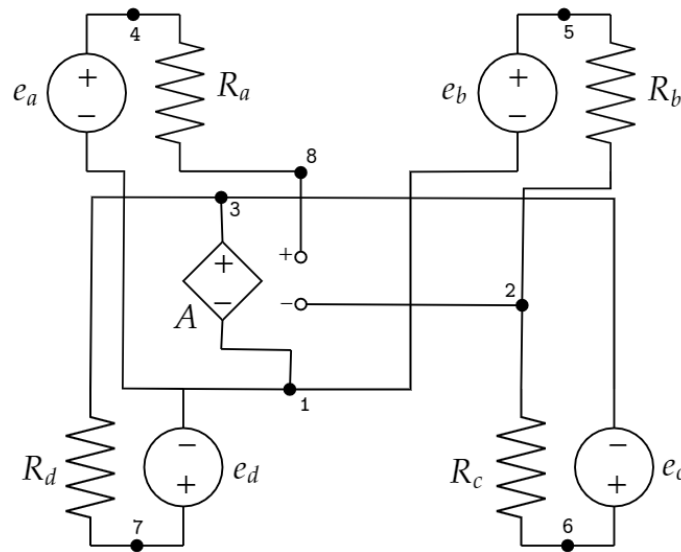


Figure 1.17: R -type Adaptor with Thévenin Port Equivalents

The MNA matrix \mathbf{X} is then formed by using the element stamps from Table 1 and is shown in Figure 1.18. Node 3 was chosen as the datum node; hence it is not shown in the figure. Plugging \mathbf{X} into (1.78) yields the scattering matrix for the R -type adaptor.

$$\underbrace{\begin{bmatrix} \mathbf{Y} & \mathbf{A}_1 & \mathbf{A}_2 \\ \mathbf{B}_1 & \mathbf{D}_{11} & \mathbf{D}_{12} \\ \mathbf{B}_2 & \mathbf{D}_{21} & \mathbf{D}_{22} \end{bmatrix}}_{\mathbf{X}} = \left(\begin{array}{cccccc|cccc|c} \frac{1}{R_d} & 0 & 0 & 0 & 0 & -\frac{1}{R_d} & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & \frac{1}{R_b} + \frac{1}{R_c} & 0 & -\frac{1}{R_b} & -\frac{1}{R_c} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{R_a} & 0 & 0 & 0 & -\frac{1}{R_a} & 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{R_b} & 0 & \frac{1}{R_b} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -\frac{1}{R_c} & 0 & 0 & \frac{1}{R_c} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -\frac{1}{R_d} & 0 & 0 & 0 & 0 & \frac{1}{R_d} & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{R_a} & 0 & 0 & 0 & \frac{1}{R_a} & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 100 & 0 & 0 & 0 & 0 & -100 & 0 & 0 & 0 & 0 & 0 \end{array} \right)$$

Figure 1.18: MNA Matrix \mathbf{X} for MXR Distortion+ R -type Adaptor

1.10 – Chapter Summary

This chapter introduced the theory and concepts of Wave Digital Filter Modeling for circuits with linear elements and nonlinear elements, as well as those with complicated nodal topologies. The Wave Digital Filter models of several one-port, two-port, and three-port electrical elements were derived in order to demonstrate a general process that can be used to derive any such element. Special attention was given to demonstrating how the presented methods could be most efficiently implemented in code in anticipation of Chapter 2, which will discuss the implementation of Wave Digital Filters on the SHARC Audio Module.

Chapter 2 – SHARC Audio Module WDF Library

In his seminal paper on Wave Digital Filter theory, Alfred Fettweis speculated that WDFs could be implemented with optimal efficiency on digital signal processors (DSPs) [5]. He argued that DSPs which use two's complement arithmetic and those which realize multiplications via bit-shift and addition operations would allow WDFs to be solved more simply than on other hardware units. While his paper was written before the advent of personal computers with high-speed processors, his point that WDFs could be efficiently implemented on embedded processors stands.

In this chapter, a C library for Wave Digital Filter modeling on the SHARC Audio Module embedded processor is presented. The library allows for a WDF structure to be specified with an object-oriented connection tree that easily interfaces with the SHARC's Bare Metal Framework. In section 2.1, an overview of existing WDF code implementations is given to contextualize the need for a library made for embedded systems. In section 2.2 a structural overview of the SHARC's Bare Metal Framework is provided. Sections 2.3-5 give an architectural overview of the library, followed by example implementations of the circuits described in sections 1.9.1 and 1.9.3. Finally, commentary on the limitations of implementing WDFs on embedded hardware is given in section 2.6.

2.1 – Previous Work

While recent research has greatly expanded the types of circuits that can be implemented in real-time using WDFs, few software packages that implement these advances in an easy-to-understand way exist. Libraries that implement the latest research,

such as WDF++ [36] and RT-WDF [37], rely on the JUCE C++ audio framework. JUCE makes productizing audio processing algorithms simple, but it comes with its own learning curve and does not offer simple prototyping features. The motivation for implementing a code library on the SHARC Audio Module is to provide an easy-to-use framework for implementing and prototyping WDF connection trees without having to implement a complex GUI.

MATLAB code that allowed for prototyping of basic Wave Digital Filter structures was provided by Udo Zölzer in his seminal textbook DAFX [38]. While MATLAB is a superior prototyping environment to JUCE, the library came out before much of the research described in Chapter 1 and has not since been updated. It also did not provide a systematic method to implement WDF connection trees. An overview of this package as well as WDF++ is provided in [39].

The C library presented in the following sections is based on an open-source library provided by Jatin Chowdhury [40], with several adjustments to make it easy to use on the SHARC platform. This framework was chosen since it does not rely heavily on C++11 conventions nor external libraries which cannot be implemented on the SHARC.

2.2 – SHARC Audio Module Bare Metal Framework

The SHARC's Bare Metal Framework is a simple C/C++ interface to its ADSP-SC589 processor that allows for efficient audio signal processing and algorithm prototyping [3]. The framework uses frame-based callback processing and processes all audio using 32-bit floating-point. It also supports the use of the Audio Project Fin, an

expansion connector for the SHARC that allows for the parametric control of variables via buttons and potentiometer, as well as MIDI and ¼ inch input/outputs.

The SHARC allows for multicore audio processing; each project contains three project folders, one of which interfaces with the ARM processor while the other two handle all the audio processing. Parameter data is shared between the two processing cores via the `multicore_data` struct. In the context of Wave Digital Filters, this feature is useful when a circuit must be broken up into multiple connection trees; different sections of the circuit can be run on each core to improve processing speed and reduce latency.

Audio flows through the SHARC cores as shown in Figure 2.1. Audio is sent from the receive buffer to SHARC Core 1, followed by SHARC Core 2. From Core 2, audio is sent back to Core 1 where it is routed to the proper peripheral via the `processaudio_output_routing()` function.

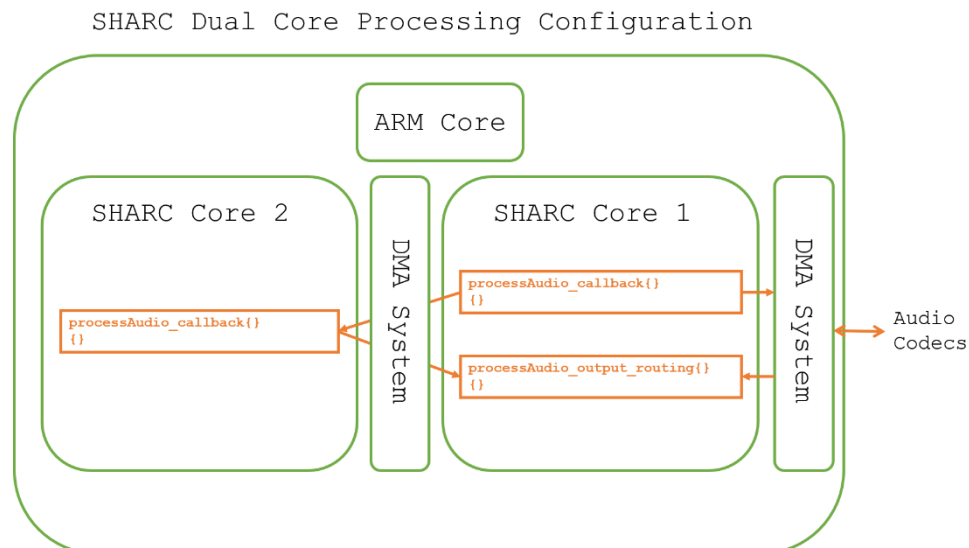


Figure 2.1: Audio Flow through SHARC Cores 1 and 2

SHARC Cores 1 and 2 process audio within their respective `callback_audio_processing.cpp` files. Each file contains several key C functions that the user must edit to implement their custom audio processing algorithms. The bulk of the audio processing occurs in `processaudio_callback()`; this function is called every time an audio frame is passed into the core and allows for samples to be read from and written to the input and output audio buffers respectively. `processaudio_setup()` is called once before the first frame is processed; this function is useful for initializing memory or generating coefficients. Any code which does not need to be run every frame, such as any functionality that only occurs when a button is pressed, can be assigned within `processaudio_background_loop()`.

2.3 - Library Overview

Figure 2.2 gives an overview of the class structure of the SHARC WDF library. The library is built off three essential classes: `wdfPort`, `wdfNode`, and `wdfTree`. The `wdfPort` class represents a generalized port as described in Section 1.2.1; each `wdfPort` instance tracks the ports incident and reflected waves a and b as well as its port resistance R_p . Additionally, each port's voltage and current are accessible via the `getPortVoltage()` and `getPortCurrent()` functions.

Connections between ports are managed using the `wdfNode` class. `wdfNode` implements two key functions. The first, `connectToNode()`, is called upon each node's initialization and enables the node to keep track of the node to which it connects. The second, `propagateImpedance()`, recursively calculates the port impedance of each node from the leaves to the root. This general interface enables each circuit element and adaptor in the library to be represented as an extension of `wdfNode`. One-port adapted and

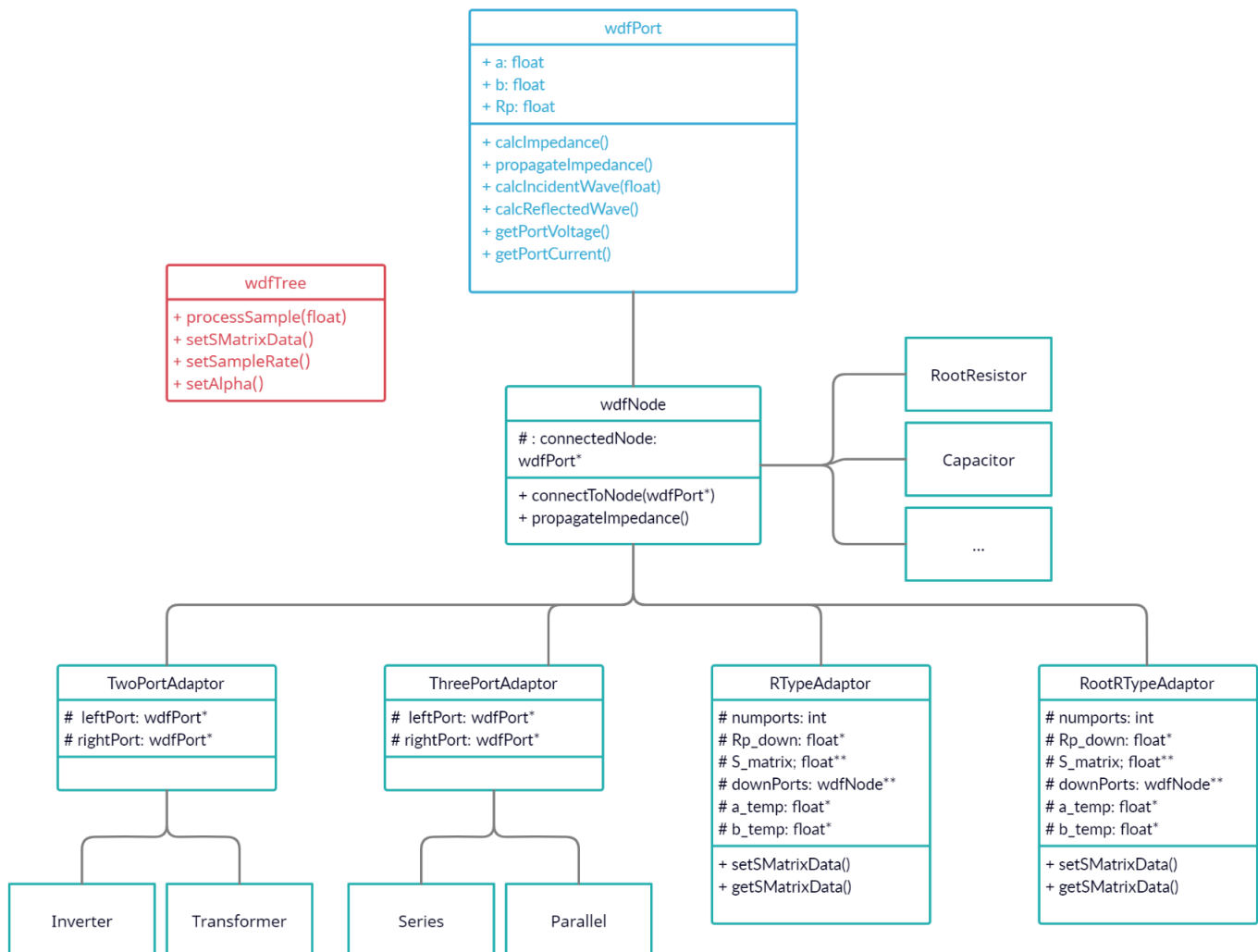


Figure 2.2: SHARC WDF Library Class Structure

unadapted elements are derived directly from `wdfNode`, while two-port, three-port, and R-type adaptors are represented by separate classes derived from `wdfNode` (e.g. `TwoPortAdaptor`, `RTypeAdaptor`).

```

1  class Resistor : public wdfNode
2  {
3  public:
4      Resistor(float R) : wdfNode("Resistor"), R_val(R)
5      {
6          calcImpedance();
7      }
8
9      ~Resistor() {}
10
11     void setResistance(float R)
12     {
13         if (R == R_val)
14             return;
15         R_val = R;
16         propagateImpedance();
17     }
18
19     float getResistance() {return R_val;}
20
21     void calcImpedance()
22     {
23         Rp = R_val;
24     }
25
26     void calcIncidentWave(float downWave)
27     {
28         a = downWave;
29     }
30
31     float calcReflectedWave()
32     {
33         b = 0.0;
34         return b;
35     }
36
37 protected:
38     float R_val;
39 };

```

Listing 1: SHARC WDF Library Resistor Implementation

Each one-port implementation must define the element's scattering behavior, port impedance, and a method that adjusts its physical characteristics. Consider Listing 1 which shows the implementation of an unadapted resistor. The functions `calcIncidentWave()` and `calcReflectedWave()` set the resistor's incident wave and retrieve its reflected wave respectively, thus implementing its scattering behavior. `calcImpedance()` sets the resistor's port impedance, while `setResistance()` adjusts its physical resistance. Any adjustment to the resistance triggers `propagateImpedance()`, which updates the port impedances of every adaptor in the connection tree.

Listing 2 shows the implementation of the three-port series adaptor. The series adaptor is initialized with the two 'child' ports connected beneath it on the connection tree, labeled `leftPort` and `rightPort`. `calcIncidentWave()` and `calcReflectedWave()` implement the adaptor's scattering behavior just as before, except in this case each function calls its children's scattering functions as well. This enables the incident and reflected waves at every port of the connection tree to be calculated recursively by leveraging the implicit connections between ports.

To implement a custom circuit, an instance of `wdfTree` must be created which creates the circuit's elements and adaptors, implements the circuit's processing functions, and provides a method to edit circuit parameters. Each element in the circuit is first initialized in the custom `wdfTree` constructor. The element designated as the root is then attached to the topmost adaptor in the tree using the `connectToNode()` function.

Finally, the user must implement the wave processing equations in `processSample()`.

Any parameterization is handled by a user-created function rather than by an overridable function.

```

1  class Series : public ThreePortAdaptor
2  {
3  public:
4      Series(wdfNode* leftPort, wdfNode* rightPort) :
5          ThreePortAdaptor(leftPort, rightPort, "Series")
6      {
7          calcImpedance();
8      }
9      ~Series() {}
10
11     void calcImpedance()
12     {
13         Rp = leftPort->Rp + rightPort->Rp;
14         gammaLeft = leftPort->Rp / Rp;
15         gammaRight = rightPort->Rp / Rp;
16     }
17
18     void calcIncidentWave(float downWave)
19     {
20         //port->incident = port->b
21         leftPort->calcIncidentWave(leftPort->b - gammaLeft
22             * (downWave + leftPort->b + rightPort->b));
23         rightPort->calcIncidentWave(rightPort->b - gammaRight
24             * (downWave + leftPort->b + rightPort->b));
25
26         a = downWave;
27     }
28
29     float calcReflectedWave()
30     {
31         //port->calcreflectedwave = port->a
32         b = -1.0 * (leftPort->calcReflectedWave()
33             + rightPort->calcReflectedWave());
34         return b;
35     }
36
37 private:
38     float gammaLeft, gammaRight;
39 };

```

Listing 2: SHARC WDF Library Series Adaptor Implementation

2.4 – Butterworth Low Pass Filter Implementation Example

Listing 3 shows the `wdfTree` class implementation of the WDF representation of a Butterworth low pass filter as derived in section 1.9.2. To start, pointers to all one-port elements and adaptors are declared as private members of the class. The user-controlled parameter `fc`, which controls the cutoff frequency of the filter, is declared here as well. The declared elements are then created and initialized in the class constructor. One-port elements are initialized with values corresponding to their physical parameters, while the adaptors are initialized with references to their two child ports. Initializing the adaptors in this manner automatically builds the connection tree up from the leaves, while line 33 connects the root node to the rest of the tree.

After initialization, `processSample()` must be overridden to implement the low pass filter tree's wave traversing behavior. First, the input sample must be set; in this case, the input sample is set equal to the voltage of the ideal voltage source. Next, the “forward scan” is implemented by calculating the reflected waves from the leaves and setting them equal to the root's incident waves. The “backward scan” is computed oppositely; the leaves' incident waves are calculated using the root's reflected wave. The output sample is computed as the voltage across the load resistor R_L . The final method, `setParams()`, adjusts the filter's frequency response by tuning the inductance of L_1 according to the desired cutoff frequency.

```

1  class TwoPoleButterworthLowPass : public wdfTree
2  {
3  private:
4      Resistor* R1;
5      Resistor* RL;
6      Capacitor* C1;
7      Inductor* L1;
8
9      Series* S1;
10     Series* S2;
11     Parallel* P1;
12
13     IdealVoltageSource* Vin;
14
15     float fc; //cutoff frequency
16
17 public:
18     TwoPoleButterworthLowPass() : fc(0.0)
19     {
20         R1 = new Resistor(50.0);
21         RL = new Resistor(1000000.0);
22         C1 = new Capacitor(47e-6, sr, alpha);
23         L1 = new Inductor(110e-6, sr, alpha);
24
25         Vin = new IdealVoltageSource;
26
27         S1 = new Series(RL, L1);
28         P1 = new Parallel(S1, C1);
29         S2 = new Series(R1, P1);
30
31         S2->connectToNode(Vin);
32     }
33
34     float processSample(float inSamp)
35     {
36         Vin->setVoltage(inSamp);
37
38         Vin->calcIncidentWave(S2->calcReflectedWave());
39         float output = RL->getPortVoltage();
40         S2->calcIncidentWave(Vin->calcReflectedWave());
41
42         return output;
43     }
44
45     void setParams(float cutoffFreq)
46     {
47         if(cutoffFreq != fc)
48         {
49             fc = cutoffFreq;
50             L1->setInductance(R1->getResistance()*RL->getResistance()*C1->getCapacitance()*fc*fc*FOUR_PI_SQ);
51         }
52     }
53
54 };

```

Listing 3: SHARC Butterworth Low Pass Filter Class Definition

Listing 2.4 briefly demonstrates how the WDF implementation is processed inside the SHARC framework. Two instances of `TwoPoleButterworthLowPass` are implemented at the top of `callback_audio_processing.cpp`, as well as the parameter `cutoffFreq` which is assigned to potentiometer 0. Inside `processaudio_callback()`, the parameter `cutoffFreq` is updated every audio block by `setParams()`, while the audio processing occurs inside the frame-based sample loop.

```

1  TwoPoleButterworthLowPass filt[2];
2  static float cutoffFreq = 0.0;
3
4  #pragma optimize_for_speed
5  void processaudio_callback(void) {
6
7      cutoffFreq = multicore_data->audioproj_fin_pot_hadc0;
8      filt[0].setParams(cutoffFreq);
9      filt[1].setParams(cutoffFreq);
10
11     for (int i = 0; i < AUDIO_BLOCK_SIZE; i++) {
12         audiochannel_0_left_out[i] =
13             filt[0].processSample(audiochannel_0_left_in);
14         audiochannel_0_right_out[i] =
15             filt[1].processSample(audiochannel_0_right_in);
16     }
17 }

```

Listing 4: SHARC Butterworth Low Pass Filter Implementation

2.5 – MXR Distortion+ Implementation Example

Given that the WDF representation of the MXR Distortion+ pedal has a large connection tree with two computationally expensive elements (the R-type adaptor and the diode pair), it is an excellent candidate to be divided into two connection trees and processed on SHARC cores 1 and 2 simultaneously. In order to achieve this, the circuit

shown in Figure 1.11a is divided into two sub-circuits, shown in Figures 2.3a and 2.3b, and the corresponding connection tree in Figure 1.14c is split up into the two subtrees shown in Figure 2.3c and 2.3d. In this formulation, the input signal travels through the first sub-circuit, yielding V_1 , the voltage across capacitor C_4 . V_1 is then used as the voltage input to

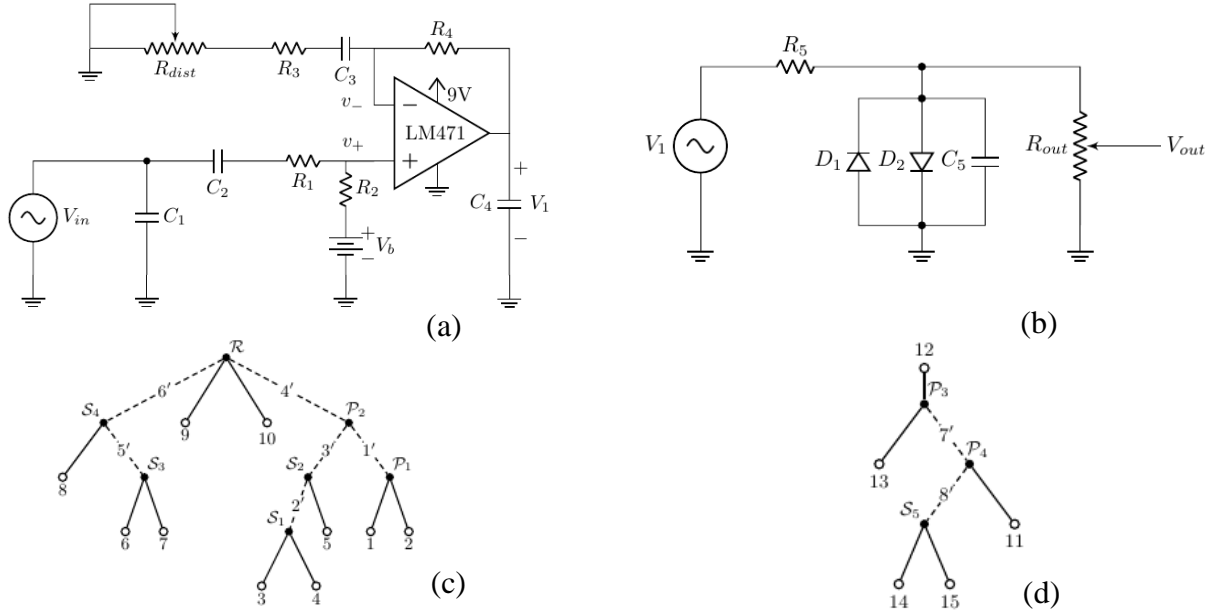


Figure 2.3: Cross-Controlled Model of MXR Distortion+ Circuit: (a-b) Sub-circuits 1 and 2 and (c-d) Corresponding SPQR Trees

the second sub-circuit which returns the final output voltage V_{out} . This method of splitting a circuit into sub-circuits and using the output voltage/current of one circuit as the input of another is referred to as *cross-controlling* the circuits [12].

Listing 2.5 shows the relevant excerpt from the implementation of the sub-circuit shown in Figure 2.3a. As before, the circuit elements are initialized and created in each tree's constructors, and the scattering behavior is implemented in `processSample()`. The scattering matrix for the R-type adaptor in `mxrDistTree1` is calculated by *R-solver*

(Chapter 3) and implemented inside the `setSMatrixData()` method. Since the values inside the scattering matrix depend on parameters Ra and Rb, `setSMatrixData()` is called within `setParams()` each time a knob is turned to update the scattering matrix values accordingly. Processing the MXR Distortion+ circuit on both cores works the same way as shown in section 2.4, except `mxrDistTree1` is initialized on core 1 while `mxrDistTree2` is initialized on core 2.

```

1  class mxrDistPlus1 : public wdfTree
2  {
3  private:
4      ...
5      // initialize wdfNode instances
6      ...
7
8      RootRtypeAdaptor* R;
9      wdfNode** R_subTreeNodes;
10
11 public:
12
13     mxrDistPlus1()
14     {
15         ...
16         // create wdfNode instances
17         ...
18
19         //Add ports to subtree nodes
20         R_subTreeNodes = new wdfNode*[4];
21         R_subTreeNodes[0] = P2;
22         R_subTreeNodes[1] = S2;
23         R_subTreeNodes[2] = R4;
24         R_subTreeNodes[3] = C4;
25
26         R = new RootRtypeAdaptor(4,R_subTreeNodes);
27
28         setSMatrixData();
29     }

```

```

53     float processSample(float inSamp)
54     {
55         Vres->setVoltage(inSamp);
56
57         R->calcIncidentWave(0.0);
58         R->calcReflectedWave();
59
60         return C4->getPortVoltage();
61     }
62
63     void setParams(float rDist)
64     {
65         if(rDist != ResDistPrev)
66         {
67             ResDist->setResistance(rDist);
68             ResDistPrev = rDist;
69             setSMatrixData();
70         }
71     }
72
73 };

```

Listing 5: SHARC MXR Distortion+ Implementation

2.6 – Performance and Limitations of SHARC WDF Library

The performance of the SHARC WDF Library was measured by examining the average CPU load on each core used to process the Butterworth low pass filter, the MXR Distortion+, and the passive tone control circuit (section 1.9.2). CPU load was measured both in MHz and as a percentage of the core's total available processing. The results are shown in Table 2.1.

Circuit	Average CPU Load (MHz)	Average CPU Load (%)
Butterworth Filter	104.5	23.2%
Passive Tone Control	381.7	84%
MXR Distortion+ (Core 1)	324.8	72.1%
MXR Distortion+ (Core 2)	159.8	35.5%

Table 2: Performance of SHARC WDF Library on Cores 1 and 2 for the Derived Circuit Models

The results indicate that while the SHARC processes Wave Digital Filters as expected, there are limitations as to the size of circuits that can be emulated. The MXR Distortion+ circuit, for example, can only be processed as two trees on separate cores since the combined average CPU load of each tree on separate cores equates to 107.6% of the available processing on a single core. Users must be mindful of CPU requirements when prototyping circuits with R-type adaptors or nonlinearities for this reason.

Another limitation of the SHARC is the lack of oversampling functionality. Oversampling can be approximated using built-in interpolation and decimation filters `fir_interp()` and `fir_decima()`, but these functions introduce additional latency (depending on the filter design) and require additional processing load. As a result, it is challenging to implement ADAA for circuits involving diodes without overloading the SHARC cores.

2.7 – Chapter Summary

In this chapter, a C library is presented which enables users to prototype Wave Digital Filter models on the SHARC Audio Module. Any circuit which consists of linear and one-port nonlinear elements in simple or complicated topologies can be modeled on the SHARC, with the caveat that special consideration must be given based on the size of the circuit in question.

The MIT Licensed version of the SHARC WDF library, as well as several example circuits, can be found at GitHub at:

https://github.com/schachtersam32/WaveDigitalFilters_Sharc

Chapter 3 – R-solver

In this chapter, a novel program that automatically finds the scattering matrix for a given *R*-type adaptor is introduced. *R-solver* can find the S-matrix for any *R*-type adaptor with or without absorbed linear multiport elements by automatically performing Modified Nodal Analysis on any given circuit. The program also allows users prototyping Wave Digital Filter models to easily switch between different op-amp macromodels to determine which model best suits their design. *R-solver* returns the scattering matrix given in a form that can be simply copied and pasted into the SHARC Audio Module `wdfTree` implementation of a circuit or any other WDF library which implements *R*-type adaptors.

In section 3.1, a brief overview of existing software which solves circuit equations via Modified Nodal Analysis is given. In section 3.2, *R-solver* is introduced and an example derivation of the MXR Distortion+ *R*-type adaptor is shown.

3.1 – Previous Work

While several programs which use Modified Nodal Analysis to solve for node voltages and currents exist, *R-solver* is the first which is explicitly designed in the context of analog modeling with Wave Digital Filters. Symbolic Circuit Analysis with MATLAB (SCAM) by Swarthmore College, for example, allows users to input a netlist from LTSpice and returns the corresponding MNA matrices [41]. While it would be trivial for users to use the information generated by SCAM to derive the appropriate scattering matrix, SCAM

does not account for certain element stamps, such as nullors and resistive VCVS, which are common to WDFs.

Another approach that is not tailored to solving WDF scattering equations can be found in the program Circuit-Solver [42]. This open-source program includes a drag-and-drop interface that allows users to build circuit diagrams graphically before solving their voltage equations. This program is more user-friendly than SCAM but does not allow access to the underlying \mathbf{X} matrix and can be prone to error.

3.2 – R-Solver Overview

R-solver comes in the form of a MATLAB live script which outputs a text file containing the derived *R*-type adaptor's scattering matrix. This format was chosen because MATLAB features a simple interface for solving symbolic equations, while live scripts allow for script parameters to be adjusted easily using controls.

To use *R-Solver*, the user must first generate a netlist which details the connections between each element in the *R*-type adaptor subcircuit. This can be easily accomplished using a program such as SPICE. When recreating the *R*-type adaptor's subcircuit in SPICE, special labeling conventions are required to simplify porting to the SHARC Audio Module:

1. A port n 's Thévenin voltage and resistance must have the same letter-name identifier, e.g. V_n and R_n .
2. If port n 's port resistance R_n is to be left as a parameterizable variable, the resistor should be given a value R_n .

3. Any two-port element absorbed within the *R*-type adaptor must be represented by a voltage-dependent voltage source in spice. This includes ideal VCVS, resistive VCVS, and nullors. A VCVS gain value only needs to be specified in SPICE if the two-port is meant to represent a VCVS.

These labeling conventions are necessary to properly differentiate external ports and absorbed two-port elements from one another inside the program.

Once the netlist has been generated, the user must set the parameters of *R-solver*.

These parameters include:

1. `fileName`: the name of the netlist file from SPICE. This file is generated automatically in SPICE and typically includes the extension `.net`.
2. `outputFileName`: the name of the file the user wishes to output the scattering matrix to. The user may leave this blank if they do not wish to return a file.
3. `adaptPort`: a Boolean which is selected if the user wishes to adapt one of the ports of the *R*-type adaptor.
4. `portToAdapt`: the port being adapted, specified as a value between 1 and N for an *R*-type adaptor with N ports.
5. `datumNode`: the node which is removed from the MNA matrix \mathbf{X} . It is recommended that the node corresponding to ground is chosen as the datum node, but if left unspecified node 1 is chosen by default.
6. `VCVSType`: if the *R*-type adaptor contains an absorbed op-amp model, this parameter allows the user to replace the ideal VCVS element stamp with the

resistive VCVS macromodel stamp or nullor stamp as shown in Table 1.1. If required, the input and output impedances can be set by the user to customize the macromodel.

Once the parameters are set, *R-solver* uses the element stamp method to systematically fill in the MNA matrix as described in section 1.9.4.

Once \mathbf{X} is populated, the adaptor's scattering matrix \mathbf{S} is calculated using (1.78). If port n is selected to be adapted, *R-solver* calculates a symbolic expression for port resistance R_n such that scattering matrix entry $s_{nn} = 0$. The final scattering matrix is then printed to a text file so that it may easily be used with the SHARC WDF library. If no such solution for R_n exists, an error message informs the user that the port cannot be adapted, and the unadapted scattering matrix will be output to the text file.

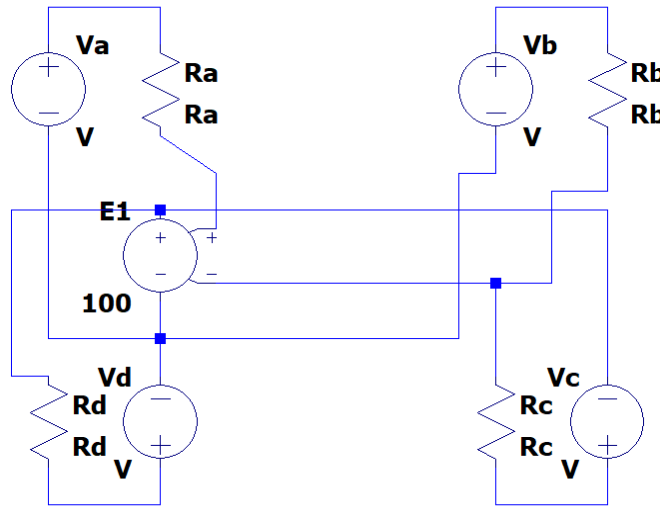


Figure 3.1: LTSpice Representation of MXR Distortion+ *R*-type Adaptor

3.3– MXR Distortion+ Scattering Matrix Derivation using R-Solver

Figure 3.1 shows the Thévenin equivalent circuit for the MXR Distortion+ as displayed in SPICE. As before, port A represents the elements connected to the op-amp's positive input, port B represents the elements connected to the negative input, port C represents the feedback resistor R_4 , and port D represents the elements connected after the output. E1 represents a general two-port device that can be replaced by a nullor, an ideal VCVS, or a resistive VCVS macromodel.

The corresponding netlist was input into *R-solver* and the unadapted scattering matrices were derived using all three op-amp models. The input impedance to the VCVS in the resistive case was set to $3\text{M}\Omega$, while the output impedance was set to 75Ω . The unadapted scattering matrices are shown in Figure 3.2. The scattering matrix derived with a nullor is the simplest to implement but the least accurate; the opposite is true of the resistive VCVS macromodel.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & -1 & 0 & 0 \\ -\frac{2 R_c}{R_b} & \frac{2 R_c}{R_b} & 1 & 0 \\ \frac{2 (R_b+R_c)}{R_b} & -\frac{2 R_c}{R_b} & -2 & -1 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \frac{200 R_b}{101 R_b+R_c} & 1 - \frac{202 R_b}{101 R_b+R_c} & \frac{2 R_b}{101 R_b+R_c} & 0 \\ -\frac{200 R_c}{101 R_b+R_c} & \frac{101 R_b+R_c}{200 R_c} & 1 - \frac{2 R_c}{101 R_b+R_c} & 0 \\ \frac{200 R_d (R_b+R_c)}{\sigma_1} & -\frac{101 R_b+R_c}{200 R_c} & -\frac{101 R_b+R_c}{200 R_c} & -1 \end{bmatrix}$$

where

$$\sigma_1 = 101 R_b R_d + R_c R_d$$

(b)

$$\begin{bmatrix} 1 - \frac{2 R_a (75 R_b+75 R_c+75 R_d-R_b R_d-R_c R_d)}{\sigma_1} & \frac{2 R_a (75 R_c+75 R_d-R_c R_d)}{\sigma_1} \\ -\frac{2 R_b (99999925 R_d-75 R_c+R_c R_d)}{\sigma_1} & 1 - \frac{2 R_b (75 R_a+75 R_c-100999925 R_d-R_a R_d-R_c R_d+75000000)}{\sigma_1} \\ \frac{2 R_c (75 R_b+100000000 R_d-R_b R_d)}{\sigma_1} & \frac{2 R_c (75 R_a-101000000 R_d-R_a R_d+75000000)}{\sigma_1} \\ -\frac{50 R_d (3999997 R_b+4000000 R_c)}{\sigma_1} & \frac{50 R_d (3 R_a+4000000 R_c+3000000)}{\sigma_1} \\ \frac{2 R_a (75 R_b-R_b R_d)}{\sigma_1} & \frac{150 R_a R_b}{\sigma_1} \\ \frac{2 R_b (75 R_a-1000000 R_d-R_a R_d+75000000)}{\sigma_1} & \frac{150 R_b (R_a+1000000)}{\sigma_1} \\ 1 - \frac{2 R_c (75 R_a+75 R_b-1000000 R_d-R_a R_d-R_b R_d+75000000)}{\sigma_1} & -\frac{150 R_c (R_a+R_b+1000000)}{\sigma_1} \\ -\frac{50 R_d (3 R_a-3999997 R_b+3000000)}{\sigma_1} & \frac{(100999925 R_b-75 R_a+1000000 R_c+R_a R_b+R_a R_c+R_b R_c-75000000)}{\sigma_1} + 1 \end{bmatrix}$$

where

$$\sigma_1 = 75000000 R_b+75000000 R_c+75000000 R_d+75 R_a R_b+75 R_a R_c+75 R_a R_d+75 R_b R_c-100999925 R_b R_d-1000000 R_c R_d - R_a R_b R_d - R_a R_c R_d - R_b R_c R_d$$

(c)

Figure 3.2: Scattering Matrices Derived with (a) Nullor, (b) Ideal VCVS, and (c) Resistive VCVS Macromodel

3.3 - Chapter Summary

In this chapter, a novel MATLAB Live Script that can derive the scattering behavior of any R-type adaptor is presented. This program greatly simplifies this process by algorithmically generating the Modified Nodal Analysis equations for a given R-type adaptor's Thévenin equivalent circuit and solving it symbolically in terms of its port resistances. The aim of creating R-solver was to provide Virtual Analog developers with a helpful tool that streamlines the derivation of WDF models.

R-solver opens the door for other utility software which can aid in algorithmically deriving Wave Digital Filter structures. Examples include a utility that can derive the scattering behavior of R-type adaptors with multiple attached nonlinearities which need to be treated as described in [14], [15], or one which could algorithmically derive a circuit's tree structure.

The R-solver live script can be found on GitHub alongside the SHARC WDF Library at

https://github.com/schachtersam32/WaveDigitalFilters_Sharc.

Bibliography

- [1] K. J. Werner, “Virtual Analog Modeling of Audio Circuitry © 2016 by Kurt James Werner . All Rights Reserved . Re-distributed by Stanford University under license with the author . This dissertation is online at : <http://purl.stanford.edu/jy057cz8322>,” vol. Ph.D. in C, no. December, p. 261, 2016.
- [2] K. J. Werner, J. O. S. III, and J. S. Abel, “Wave Digital Filter Adaptors for Arbitrary Topologies and Multiport Linear Elements.” Accessed: Jan. 25, 2021. [Online].
- [3] “SHARC Audio Module [Analog Devices Wiki].” <https://wiki.analog.com/resources/tools-software/sharc-audio-module> (accessed Apr. 26, 2021).
- [4] “State Space Filters.” https://ccrma.stanford.edu/~jos/filters/State_Space_Filters.html (accessed Apr. 29, 2021).
- [5] 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.
- [6] A. Bernardini, P. Maffezzoni, and 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, 2020, doi: 10.1109/TCSI.2020.3044002.
- [7] A. Proverbio, A. Bernardini, and A. Sarti, “Toward the wave digital real-time emulation of audio circuits with multiple nonlinearities,” *European Signal Processing Conference*, vol. 2021-Janua, pp. 151–155, 2021, doi: 10.23919/Eusipco47968.2020.9287449.
- [8] S. Bilbao, *Wave and Scattering Methods for Numerical Simulation*. 2005.
- [9] R. W. Anderson, “S-Parameter Design for Faster, More Accurate Network Design,” no. February, 1997, [Online]. Available: <http://cp.literature.agilent.com/litweb/pdf/5989-9273EN.pdf>
<http://cp.literature.agilent.com/litweb/pdf/5952-1130.pdf>.
- [10] A. Sarti and G. de Sanctis, “Systematic Methods for the Implementation of Nonlinear Wave-Digital Structures,” *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: REGULAR PAPERS*, vol. 56, no. 2, 2009, doi: 10.1109/TCSI.2008.2001801.

- [11] D. Fränken, J. Ochs, and K. Ochs, "Generation of Wave Digital Structures for Networks Containing Multiport Elements," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 3, pp. 586–596, 2005, doi: 10.1109/TCSI.2004.843056.
- [12] G. de Sanctis and A. Sarti, "Virtual Analog Modeling in the Wave-Digital Domain," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 4, pp. 715–727, 2010, doi: 10.1109/TASL.2009.2033637.
- [13] A. Bernardini, A. E. Vergani, and A. Sarti, *Wave Digital Modeling of Nonlinear 3-terminal Devices for Virtual Analog Applications*, vol. 39, no. 7. Springer US, 2020.
- [14] K. J. Werner, V. Nangia, J. O. Smith, and J. S. Abel, "Resolving wave digital filters with multiple/multiport nonlinearities," *DAFx 2015 - Proceedings of the 18th International Conference on Digital Audio Effects*, pp. 31–33, 2015.
- [15] M. J. Olsen, K. J. Werner, and J. O. S. III, "Resolving grouped nonlinearities in wave digital filters using iterative techniques," *DAFx 2016 - Proceedings of the 19th International Conference on Digital Audio Effects*, pp. 1–8, 2016.
- [16] D. A. Effects, "Design Principles for Lumped Model Discretisation Using Möbius Transforms," François G. Germain, Kurt J. Werner Center for Computer Research in Music and Acoustics (CCRMA), pp. 1–8, 2015.
- [17] Ó. Bogason and K. J. Werner, "Modeling Time-Varying Reactances Using Wave Digital Filters." Accessed: Feb. 08, 2021. [Online].
- [18] A. Fettweis, "Robust numerical integration using wave-digital concepts," in *Multidimensional Systems and Signal Processing*, Jan. 2006, vol. 17, no. 1, pp. 7–25, doi: 10.1007/s11045-005-6236-3.
- [19] S. D'Angelo and V. Välimäki, "Wave-Digital Polarity and Current Inverters and Their Application to Virtual Analog Audio Processing," 2012. Accessed: May 11, 2021. [Online]. Available: http://www.ieee.org/publications_standards/publications/rights/rights_link.html.
- [20] "ElectroSmash - MXR Distortion + Circuit Analysis." <https://www.electrosmash.com/mxr-distortion-plus-analysis> (accessed May 15, 2021).
- [21] R. C. D. Paiva, S. D'Angelo, J. Pakarinen, and V. Välimäki, "Emulation of operational amplifiers and diodes in audio distortion circuits," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 10, pp. 688–692, 2012, doi: 10.1109/TCSII.2012.2213358.

- [22] K. J. Werner, V. Nangia, A. Bernardini, J. O. Smith, and A. Sarti, "An Improved and generalized diode clipper model for wave digital filters," *139th Audio Engineering Society International Convention, AES 2015*, no. October, 2015.
- [23] S. D. ' Angelo, L. Gabrielli, and L. Turchet, "Fast Approximation of the Lambert W Function for Virtual Analog Modelling." Accessed: Apr. 11, 2021. [Online].
- [24] A. Bernardini, K. J. Werner, P. Maffezzoni, and A. Sarti, "Wave digital modeling of the diode-based ring modulator," *144th Audio Engineering Society Convention 2018*, no. May, 2018.
- [25] D. Albertini, A. Bernardini, and A. Sarti, "Antiderivative Antialiasing in Nonlinear Wave Digital Filters." Accessed: Feb. 18, 2021. [Online].
- [26] S. Bilbao, F. Esqueda, J. D. Parker, V. Välimäki, and V. V. Antiderivative, "Antialiasing for Memoryless Nonlinearities," *IEEE Signal Processing Letters*, vol. 24, no. 7, pp. 1049–1053, 2017, doi: 10.1109/LSP.2017.2675541.
- [27] J. D. Parker, V. Zavalishin, and E. le Bivic, "Reducing the Aliasing of Nonlinear Waveshaping Using Continuous-Time Convolution." Accessed: Apr. 19, 2021. [Online].
- [28] R. Nouta, "The Jaumann Structure in Wave-Digital Filters," *International Journal of Circuit Theory and Applications*, vol. 2, no. 2, pp. 163–174, Jun. 1974, doi: 10.1002/cta.4490020205.
- [29] D. Fränken, J. Ochs, and K. Ochs, "Generation of wave digital structures for connection networks containing ideal transformers," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 3, no. June 2003, 2003, doi: 10.1109/iscas.2003.1205000.
- [30] C. Gutwenger and P. Mutzel, "A Linear Time Implementation of SPQR-Trees." Accessed: May 14, 2021. [Online].
- [31] B. Pérez-Verdú, J. L. Huertas, and A. Rodríguez-Vézquez, "A New Nonlinear Time-Domain Op-Amp Macromodel Using Threshold Functions and Digitally Controlled Network Elements," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 4, pp. 959–971, 1988, doi: 10.1109/4.347.
- [32] K. J. Werner, W. Ross Dunkel, M. Rest, M. J. Olsen, and J. O. Smith, "Wave Digital Filter modeling of circuits with operational amplifiers," *European Signal Processing Conference*, vol. 2016-Novem, pp. 1033–1037, 2016, doi: 10.1109/EUSIPCO.2016.7760405.

- [33] A. E. Ruehli, “The Modified Nodal Approach to Network Analysis,” no. October, 2018, doi: 10.1109/TCS.1975.1084079.
- [34] A. E. Ruehli, G. Antonini, and L. Jiang, “Modified Nodal Analysis Stamps,” in *Circuit Oriented Electromagnetic Modeling Using the Peec Techniques*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2017, pp. 373–382.
- [35] K. J. Werner, A. Bernardini, J. O. Smith, and A. Sarti, “Modeling Circuits with Arbitrary Topologies and Active Linear Multiports Using Wave Digital Filters,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 12, pp. 4233–4246, 2018, doi: 10.1109/TCSI.2018.2837912.
- [36] “WDF++ - new restructuration & project (audio processor) - Useful Tools and Components - JUCE.” <https://forum.juce.com/t/wdf-new-restructuration-project-audio-processor/13104> (accessed May 23, 2021).
- [37] M. Rest, W. R. Dunkel, K. J. Werner, and J. O. Smith, “RT-WDF-A modular wave digital filter library with support for arbitrary topologies and multiple nonlinearities,” *DAFx 2016 - Proceedings of the 19th International Conference on Digital Audio Effects*, pp. 287–294, 2016.
- [38] S. Edition, *DAFX: Digital Audio Effects*. 2011.
- [39] J. O. Smith and K. J. Werner, “Recent Progress in Wave Digital Audio Effects WDF Software Overview and Demo.” Accessed: May 23, 2021. [Online].
- [40] “GitHub - jatinchowdhury18/WaveDigitalFilters: Circuit Modelling with Wave Digital Filters.” <https://github.com/jatinchowdhury18/WaveDigitalFilters> (accessed May 23, 2021).
- [41] “SCAM: Symbolic Circuit Analysis in MatLab.” <https://lpsa.swarthmore.edu/Systems/Electrical/mna/MNA6.html> (accessed Jan. 21, 2021).
- [42] D. Bala, “GitHub - circuit-solver/circuit-solver.github.io: This is Circuit Solver which can solve any kind of circuit for you. Do check it out!!,” Dec. 27, 2020. <https://github.com/circuit-solver/circuit-solver.github.io> (accessed May 30, 2021).