Real-Time Circuit Simulation with Wave Digital Filters in C++

CppCon 2024, Aurora, CO, USA, September 15-20, 2024

Author: Jatin Chowdhury
Affiliation: Chowdhury DSP
E-Mail: jatin@chowdsp.com
Website: chowdsp.com

Introduction

Analog audio effects have unique sound characteristics that are desirable to musicians and recording engineers. However, musicians are increasingly using software-based audio effects, which are typically cheaper and more convenient. As a result, many audio software effects utilize simulations of analog circuits as part of their sound-processing algorithms. Given that audio effects are typically required to run in "real-time", traditional circuit modelling softwares (e.g. LTSpice) are typically not suitable for this purpose.

Wave Digital Filters are a paradigm for modelling circuits that has become popular over the past decade in the audio research community. This poster presents header-only C++ library that implements commonly used circuit components (e.g. Resistors, Capacitors, Diodes, etc) as Wave Digital Filters, allowing the user to quickly and easily construct circuit simulations that are suitable for real-time applications.

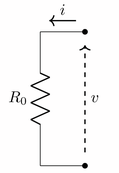
Wave Digital Filters

Wave Variables

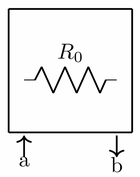
Wave Digital Filters (WDFs) use "wave variables" instead of Kirchoff variables (voltage/current)

- "Incident" wave: $a = v + R_0 i$
- "Reflected" wave: $b = v - R_0 i$

Kirchoff Domain



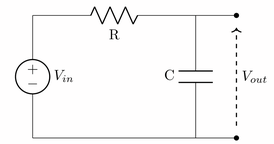
Wave Domain



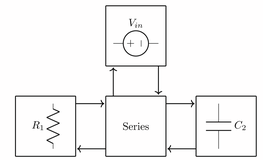
Wave Digital Filter Rules

- Each "port" has three variables: incident wave, reflected wave, port impedance
- "Leaf" nodes are 1-port: resistor, capacitor, inductor
- "Adaptor" nodes are 2-port, 3-port, or more: series, parallel, inverter
- Special "unadaptable" leaf nodes need to be at the "root" of the tree
- Unadaptable nodes include: ideal voltage/current source, diodes, switches
- Each port can be "probed" to determined the current or voltage across the port

Example:



We can construct a model of this circuit using a wave digital voltage source, resistor, and capacitor, all connected with a series adaptor.



References:

- Alfred Fettweis, "Wave Digital Filters: Theory and Practice", 1986, IEEE Invited Paper
- David Yeh and Julius Smith, "Simulating Guitar Distortion Circuits Using Wave Digital And Nonlinear State Space Formulations", Proc. of DAFX 11, 2008
- Kurt Werner, "Virtual Analog Modeling of Audio Circuitry Using Wave Digital Filters", PhD. Dissertation, Stanford University, 2016

C++ Implementation

WDF Basics

```cpp
/** WDF Base Variables */
struct WDFMembers
{
    float R0 = 0.0f; // port impedance
    float a = 0.0f;  // incident wave
    float b = 0.0f;  // reflected wave
};

/** WDF Voltage Probe */
template <typename WDFElementType>
float voltage (WDFElementType& element)
{
    return (element.wdf.a + element.wdf.b) / 2.0f;
}
```

Constraints

- The compiler can "see through" the entire call tree (no indirect function calls).
- Circuit topology is fixed at compile-time.

WDF Resistor

```cpp
class WDFResistorT
{
public:
    ResistorT (float new_resistance_ohms) : R (new_resistance_ohms) {}

    void set_resistance (float new_resistance_ohms) { R = new_resistance_ohms; }

    void calc_impedance()
    {
        wdf.R0 = R; // port impedance := resistance
    }

    inline void incident(float x) { wdf.a = x; }
    inline float reflected() { return 0.0f; }

    WDFMembers wdf;

private:
    float R = 0.0f; // resistance value (Ohms)
};
```

"Leaf" nodes have a common interface: *incident()*, *reflected()*, and *calc_impedance()*.

Example WDF Circuit Model

```cpp
struct RCLowpass
{
    ResistorT R1 { 1.0e3f }; // 1 kOhm
    CapacitorT C2 { 1.0e-6f }; // 1 uFarad
    WDFSeriesT<decltype (R1), decltype (C2)> S1 { R1, C2 };
    VoltageSourceT<decltype (S1)> Vin { S1 };

    void setup (float sample_rate)
    {
        C2.set_sample_rate (sample_rate);
        C2.reset();
    }

    float process_sample (float input_voltage)
    {
        Vin.set_voltage (input_voltage);
        Vin.incident (S1.reflected());
        S1.incident (Vin.reflected());
        return voltage (C2);
    }
};
```

An RC Lowpass Filter can be implemented by combining multiple WDFs. Note that *decltype* can be very useful for expressing complicated type hierarchies.

WDF Series Adaptor

```cpp
template <typename Port1Type, typename Port2Type>
class WDFSeriesT
{
public:
    WDFSeriesT (Port1Type& _port1, Port2Type& _port2) : port1 (_port1), port2 (_port2) {}

    void calc_impedance()
    {
        wdf.R0 = port1.wdf.R0 + port2.wdf.R0;
        port1Reflect = port1.wdf.R / wdf.R;
    }

    void incident(float x)
    {
        const auto b1 = port1.wdf.b - port1_reflect * (x + port1.wdf.b + port2.wdf.b);
        port1.incident (b1);
        port2.incident (-(x + b1));
        wdf.a = x;
    }

    float reflected()
    {
        wdf.b = -(port1.reflected() + port2.reflected());
        return wdf.b;
    }

    WDFMembers wdf;

private:
    Port1Type& port1;
    Port2Type& port2;

    float port1_reflect;
};
```

"Adaptor" nodes can be implemented as template classes, with the types of the down-stream components as the template arguments.

chowdsp_wdf Library

chowdsp_wdf is a header-only C++ libraries containing Wave Digital Filter implementations of commonly used circuit components, including resistors, capacitors, inductors, diodes, and more. The implementations can be used with both scalar and SIMD data types, and the library can be extended to support custom circuit elements. The library is licensed under the BSD 3-clause license.

The table below compares the run-time performance of *chowdsp_wdf* with two other open-source WDF libraries: *RT-WDF* (C++) and *wd_models* (Faust). For all tested circuits *chowdsp_wdf* achieved the best or 2nd-best score.

Code Repository

https://github.com/Chowdhury-DSP/chowdsp_wdf

Reference Paper

https://arxiv.org/pdf/2210.12554

Performance Comparison:

How much time (seconds) is required to process 1000 seconds of audio?

	chowdsp_wdf	***wd_models***	***RT-WDF***
LPF-2	**0.3319**	0.7685	3.141
FF-2	**2.035**	2.083	11.538
Diode Clipper	**1.905**	5.756	N/A
Bassman Tone Stack	**0.6576**	0.7411	7.92
Baxandall EQ	1.184	**1.021**	14.87

References:

- "A Wave-Digital Modeling Library for the Faust Programming Language", D. Roosenburg et. al, SMC, 2021
- "RT-WDF—A Modular Wave Digital Filter Library with Support for Arbitrary Topologies and Multiple Nonlinearities", M. Rest, et. al, DAFx-19, 2016