University of California
Santa Barbara

Modular Understanding: A Taxonomy and Toolkit for Designing Modularity in Audio Software and Hardware

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Media Arts and Technology

by

Michael Lynn Saverio Hetrick

Committee in charge:

Professor Curtis Roads, Chair
Professor Clarence Barlow
Professor Andrés Cabrera
Professor Matthew J. Wright

March 2017

The dissertation of Michael Lynn Saverio Hetrick is approved.

Professor Clarence Barlow

Professor Andrés Cabrera

Professor Matthew J. Wright

Professor Curtis Roads, Committee Chair

December 2016

Modular Understanding: A Taxonomy and Toolkit for Designing Modularity in Audio Software and Hardware

by

Michael Lynn Saverio Hetrick

In memory of Don Buchla (1937 - 2016)

Acknowledgments

This dissertation would not have been possible without the support of the people around me.

First and foremost, I would like to thank my academic committee for being friends and mentors for the last six years. Without their support, guidance, and friendship, you would not be reading this document. Curtis, you've completely changed the way that I listen to music and think about our field. I cannot plainly describe the impact that you've had on my life and career. Matt, you made me remember the importance of the human element in music through your community percussion performances and insistence on better electronic control schemes. Andrés, your DSP know-how and generosity in maintaining the open-source QtCSound were critical for the creation of Euro Reakt. Clarence, your analytic lectures and freewheeling small-group discussions reminded me to always balance work and fun, levity and gravity, humor and gloom, experience and experiments.

Next, I would like to thank my business partners Joshua Dickinson and Ryan McGee for helping me to build Unfiltered Audio as an amazing company and turning my dream job into a reality. Here's to an exciting 2017!

I would like to thank all of the students and faculty at the Media Arts and Technology department at UC Santa Barbara. The diverse backgrounds and experiences present have lead to an incredible community full of exciting projects and research. Charlie Roberts, Graham Wakefield, Lance Putnam, Angus Forbes, and Karl Yerkes are a few of the colleagues whose patience and knowledge I am grateful for. Thank you for answering every DSP or programming question.

I would like to thank every module manufacturer that supported me throughout this long dissertation. Many companies provided me with academic discounts on

gear, including 4ms, The Harvestman, Intellijel, Make Noise, Malekko, Mannequins, Monome, Mutable Instruments, SSF, Tiptop Audio, and WMD. I would especially like to thank Walker Farrell (at Make Noise) and Olivier Gillet (at Mutable Instruments) for their frequent correspondence and in-depth conversations regarding modular design and performance. The MW, Lines, and Electro-Music communities are irreplaceable centers of the modular community that I am very grateful for.

I would like to thank the Reaktor User Library community for their support and feedback through the many iterations of Euro Reakt over the past year. Of course, I would also like to thank Native Instruments not only for their excellent Reaktor 6, but also for believing in my work and giving me the platform to present my work at their Native Sessions event.

Finally, I would like to thank my family. My wonderful wife, Rebecca Hetrick, has provided me with endless love and support (not to mention patience as I holed myself up in my office for endless periods of time). My mother and mother-in-law, Mimi Geihsler and Allison Snyder, were critical for their weekly question:

"Is your dissertation done yet?"

Curriculum Vitae

Michael Lynn Saverio Hetrick

Education

- Bachelor of Arts in Digital Media and Distribution, Vanderbilt University, June 2010
- Masters of Arts in Media Arts and Technology, University of California Santa Barbara, December 2011

Professional Employment

2012-Present: Owner/Developer at Unfiltered Audio LLC.

2014-2015: Windows Programmer at Slate Digital.

2014-2015: Teaching Associate at University of California, Santa Barbara.

2012-2014: Teaching Assistant at University of California, Santa Barbara.

2013-2014: C++/CUDA Developer at Mayachitra, Inc.

2012-2013: Developer at Biobeats.

2011-2013: CREATE Technical Coordinator at University of California, Santa Barbara.

Awards

2016: Innovation Award - Computer Music Magazine. Awarded for Unfiltered Audio Fault.

2016: Performance Award - Computer Music Magazine. Awarded for Unfiltered Audio Fault.

2015: Performance Award - Computer Music Magazine. Awarded for Unfiltered Audio Sandman.

2015: Value Award - Computer Music Magazine. Awarded for Unfiltered Audio Sandman.

2014: Gold Award - Graduate Design Competition. Audio Engineering Society. Awarded for Unfiltered Audio G8 Dynamic Gate.

Abstract

Modular Understanding: A Taxonomy and Toolkit for Designing Modularity in Audio Software and Hardware

by

Michael Lynn Saverio Hetrick

Modular synthesis is a continually evolving practice. Currently, an effective taxonomy for analyzing modular synthesizer design does not exist, which is a significant barrier for pedagogy and documentation. In this dissertation, I will define new taxonomies for modular control, patching strategies, and panel design. I will also analyze how these taxonomies can be used to influence the design of musical applications outside of hardware, such as my company Unfiltered Audio's software products. Finally, I will present Euro Reakt, my collection of over 140 module designs for the Reaktor Blocks format and walk through the design process of each.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

In the past decade, hardware modular synthesizers have seen a massive resurgence. Electronic musicians and composers are discovering the flexibility and hands-on nature of this creative equipment at an unprecedented rate. New Eurorack module designs are reaching this market on a weekly basis, and computer musicians have an expanding variety of software modular platforms to choose from.

It is a continuously and quickly evolving practice. When Dieter Doepfer established the "Eurorack" modular format in 1995 [2], individual module designs were simple, all-analog circuits. Now, many modules feature complex digital algorithms featuring cutting-edge DSP techniques [3]. With case space at a costly premium, modules of both analog and digital design have attempted to pack more functionality into smaller spaces.

As hardware modules take on more polymorphic design strategies, they are breaking further away from software modulars which usually rely on simpler, single-function designs. In a software modular system, there's one primary constraint:

processor power. A user can add as many copies of a single module as they'd like and never worry about the power, space, or monetary costs of each. However, this processor constraint influences modules to do only the bare minimum and not create computational overhead for secondary functionality.

The single-function design paradigm necessitates more intent from a user. If the user does not know what they wish to create, they won't overcome the initial intimidating hurdle of the blank canvas presented by Max, Reaktor, and like-minded environments. A well-designed hardware system, though, is less of a programming environment and more of a finished instrument waiting to be played.

This difference in design perspectives has created a difficult situation for modular pedagogy and documentation. For a student interested in learning how a hardware modular works, a software modular may be an affordable alternative, but it doesn't quite capture the patching techniques, the physical immediacy, or the multi-level control strategies of the hardware environments. For students and educators who are able to afford a hardware modular, they may find themselves overwhelmed by poor documentation, a constant influx of new designs, and a lack of guidance when putting together a new system.

This dissertation aims to resolve a number of these issues. My primary contribution is a set of three taxonomies to analyze modular design, modular control strategies, and modular patching techniques. Each taxonomy is a step toward a more complete pedagogical framework for modular synthesis, along with a useful analysis as to exactly how software modulars differ from their hardware counterparts.

As a result of these taxonomies, I am also presenting two large software projects. The first is Unfiltered Audio, a plug-in company that I created in 2012 with two other MAT students. Our software designs are heavily influenced by hardware modular

synthesizers. I will present some of our most notable designs, along with how they relate to the new taxonomies.

I will also present Euro Reakt, a collection of over 140 modules for the affordable Reaktor software modular environment. This collection is perhaps the closest a software modular system has come to hardware design, as it focuses on polymorphic module behavior along with intuitive control sets and layouts. Each module is thoroughly documented, and the Euro Reakt package comes with many pre-built demonstration systems. This makes it an ideal choice for students, educators, musicians, and composers.

1.2 Statement of Research

This dissertation outlines five contributions to the fields of modular synthesis design and pedagogy:

1. **A Taxonomy of Modular Design Strategies**. This taxonomy considers how modules are designed and how they interact with other modules in a system. I will tackle this from both a historical perspective (i.e. a more rigorous definition of the difference between Moog and Buchla design strategies instead of "East Coast" vs. "West Coast") and a modern Eurorack perspective. This taxonomy will later be applied to software modulars in Chapter 6.

2. **A Taxonomy of Modular Control Strategies**. This taxonomy looks at the wide range of control methods available to modular musicians and composers, from sensors and translators to keyboards and joysticks.

3. **A Taxonomy of Modular Patching Strategies**. This taxonomy presents ways of breaking down large, complex modular patches into smaller subpatches

and concepts known as "meta-modules". These meta-modules are useful to designers, as they present synergistic interactions between more basic building blocks, encouraging designs that combine and extend these ideas. They are also useful to composers, as they outline strategies for composing more complex behaviors through the use of simple ideas.

4. **Unfiltered Audio: Polymorphism in Plug-ins**. This section outlines ways of analyzing and extending existing signal processing algorithms through the use of the above taxonomies. My primary project for this section is Unfiltered Audio (my plug-in company) and its designs.

5. **Euro Reakt**. This last section focuses on Euro Reakt, my collection of over 140 modular designs for Native Instruments' Reaktor 6. This section will present each module and investigate the design process of each by using the above taxonomies. Euro Reakt is an attempt to bridge the design gap between hardware and software modular synthesis. Most software modulars use monosemous design strategies, while Euro Reakt embraces the polymorphic and rhizomatic strategies present in modern hardware. It also includes a number of demonstrative instruments and per-knob documentation, making it an excellent choice for modular pedagogy.

Chapter 2

Taxonomy #1: Module Designs

2.1 Classification of Module Designs

In this section, I will outline the seven most common design strategies for modules. In the hardware realm, a single module is more easily defined as a physical unit that connects to a larger system. In the software realm, the concept of a "module" is defined based on the platform. In Reaktor, this would be a "Block". In Max or PureData, this would be an "object". In SuperCollider or CSound, this would be an "opcode" or "uGen".

2.1.1 Monosemous

A ***monosemous*** module is one that does not welcome alternative patching methods. Its usage is singular, static, and (generally) easy to comprehend. These modules hold pedagogical value and are great choices for student and teaching systems.

This is the most common design strategy for software modules, where CPU usage is the primary design concern.

2.1.2 Rhizomatic

A ***rhizomatic*** module is a module that serves one distinct purpose, but welcomes experimental manipulation and alternative patching strategies. In general, it is a module that is designed to be more aware of the system around it.

2.1.3 Expandable

An ***expandable*** module is a module that is designed to link up to another module (typically through the use of a behind-the-panel ribbon cable) to extend or even completely change functionality. The "expander" can be a standalone module that can function on its own or a dedicated module whose only function is to work as the expander.

2.1.4 Polymorphic

A ***polymorphic*** module is a module that can serve multiple, distinct functions. There are four major categories of polymorphism:

1. Modal
2. Independent
3. Linked
4. Simultaneous

A module can combine multiple forms of polymorphism into one design. Most polymorphic modules fit the definition of "rhizomatic" as well. The only exception is that the individual modes of a modally polymorphic module can be monosemous. Examples will be given in sections 2.2.3 and 2.2.4.

2.1.4.1 Modal Polymorphism

A ***Modally Polymorphic*** module is a module that can serve multiple purposes, but only one purpose can be served at a time. These modules typically have switches or menus to access various modes of operation. However, smooth parameters like Frequency can create modal polymorphism if an oscillator is able to move between audible and sub-audible frequencies.

It is important to note, though, that a module is modally polymorphic if and only if the module can change its purpose on its own. A simple sample-and-hold module that relies on external clocking, for instance, is not modally polymorphic. You could argue that the rate of the external clock would change the behavior of the module from a stepped modulation sequencer to an audio-rate sample rate reduction effect. However, these two functions are determined by an external module. If the wide-range clock is part of the same module, though, then it could be considered to be modally polymorphic.

There are two ways to further analyze modally polymorphic modules: functional and terminal uniformity.

FUNCTIONAL UNIFORMITY A modally polymorphic module displays ***Functional Uniformity*** if all available modes fit into the same category of signal processing (i.e. all modes are oscillators, all modes are echo effects, etc.).

TERMINAL UNIFORMITY A modally polymorphic module displays ***Terminal Uniformity*** if all available modes use identical input and output configurations. For instance, if one of the modes requires an additional gate that the other modes do not use, then the module does not display terminal uniformity.

Modally polymorphic modules that are non-uniform are typically useful for small

systems where very versatile modules are needed. As an example, the Expert Sleepers Disting is a small, 4HP module that covers dozens of functions including envelopes, oscillators, delays, distortions, and more [4].

Uniform modules can be more useful for quickly exploring compositional ideas without the need for repatching. For instance, a composer could create a melody for a functionally uniform, modally polymorphic oscillator. After finding a melody that they like, they could then switch between the available modes to audition various timbres.

2.1.4.2 Independent Polymorphism

A module that exhibits ***Independent Polymorphism*** contains multiple, unlinked functions. These are typically smaller "utility" functions that don't have complex circuits or layouts.

As a general design strategy, this is more viable for hardware. Panels and case space are expensive, so using one panel to host a number of useful functions is a good value proposition for a user. This design strategy does not hold up well for software, where every active function uses CPU.

2.1.4.3 Linked Polymorphism

A module that exhibits ***Linked Polymorphism*** contains sections that can work completely independently, but are chained together by default.

2.1.4.4 Simultaneous Polymorphism

A module that exhibits ***Simultaneous Polymorphism*** is one that can be used for multiple purposes at a time. The distinguishing factor of Simultaneous Polymor-

Figure 2.1: Synthrotek Echo. This is a simple, user-friendly delay.

phism is that these various functions share common parameters. For instance, an envelope with an "End of Envelope" trigger could be considered to be a simultaneous envelope generator and trigger delay. Manipulating the length of one stage of the envelope would affect both the length of the envelope and the length of the gate delay.

2.2 Examples of Classifications

2.2.1 Effect Example: Delay

2.2.1.1 Monosemous: Synthrotek Echo

The Synthrotek Echo [5] is a simple delay unit primarily meant for DIY builders. There are three knobs on the Echo: Rate, Feedback, and Mix. There are jacks for Input, Output, and CV control over Rate. This module is defined as "Monosemous", as there is only one possible patching strategy for it. A dry audio signal goes into the input, and a wet audio signal comes out of the output. There is voltage control over one parameter, but it doesn't change the functionality of the unit.

2.2.1.2 Rhizomatic: Audio Damage Dub Jr. Mk2

The Audio Damage Dub Jr. Mk2 [6] is an example of a "Rhizomatic" design. The three knobs are identical to the Echo: Time, Feedback, and Mix. There are two major improvements that make this more flexible in a modular environment. The first is a Clock jack. This allows a composer to ensure that the length of the echoes will always be some metric division of their master clock (When synced, the Time knob becomes a Div knob). The second is a tapped feedback loop. The feedback from this delay can be processed using other modules, such as frequency shifters and filters to provide timbral interest or VCAs to control the intensity of the feedback. Compared to the Synthrotek Echo, this is a delay design that is more "aware" of communication and interaction with other modules. It is "rhizomatic" and not "polymorphic" because it can only be used as a delay. There's no alternative functionality.

Figure 2.2: Audio Damage Dub Jr. MK2

Figure 2.3: Mungo d0, shown with Mungo Zoom and Macro Machines Storage Strip expanders.

2.2.1.3 Expandable: Mungo d0 + Mungo Zoom or Macro Machines Storage Strip

The Mungo d0 [7] is a stereo delay with tempo tracking. It is an example of expandable design as it can be upgraded with one of two expander modules. The first is the Mungo Zoom, a helper module that only works by connecting to Mungo modules. It is a switch that allows a user to "zoom" in on a knob's parameter range, giving finer control over a smaller parameter range.

The other possible expander is the Macro Machines Storage Strip [8]. The Storage Strip is a module that replicates the functionality of the Mungo Zoom while also adding the ability to save and recall presets on the connected module. These presents can be recalled manually. To build on this, the Storage Strip is also an expandable design. Macro Machines also make a module called the Dynamic Destiny. On its own, the Dynamic Destiny can be used as a dual switch. However, when connected to the Storage Strip, it can be used to recall Mungo module presets under voltage control (instead of only manual control).

2.2.1.4 Modal Polymorphism: Tiptop Audio Z-DSP

The Tiptop Audio Z-DSP [9] is an example of Modal Polymorphism. This module has a cartridge reader on the front that allows for the loading of various algorithms. It comes with one cartridge called "Dragonfly Delays". This cartridge contains eight

Figure 2.4: Tiptop Audio Z-DSP. Note the cartridge reader on the front for loading different algorithms.

different delay algorithms. The cartridge can be swapped, turning the Z-DSP into a multi-algorithm filter, an oscillator, a bitcrusher, and more. The primary reason that this is modally polymorphic is that only one algorithm can be loaded at a time.

Analyzing the uniformity of the Z-DSP is on a per-cartridge basis. For instance, the aforementioned Dragonfly Delays cartridge is terminally and functionally uniform. Every mode on the cartridge is an echo effect, and each mode shares the same input/output configuration. However, other cartridges like the "Broken Silicon Error Codes #1" [10] combine disparate modes like bit-crushing, noise generation, additive synthesis, and burst generation onto one cartridge. These modes are not terminally uniform, either, as some of the modes do not use the Z-DSP's inputs, while other modes depend on them.

2.2.1.5 Independent Polymorphism: Sputnik Modular Four-Tap Delay and Dual Crossfader

The Sputnik Modular Four-Tap Delay and Dual Crossfader [11] is an example of Independent Polymorphism. This single module has two major sections. The top

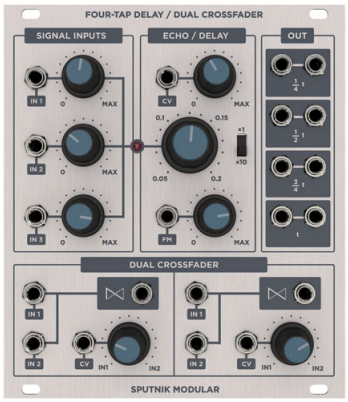

Figure 2.5: Sputnik Modular Four-Tap Delay/Dual Crossfader. The delay's outputs and crossfader's inputs are not connected.

section is a four-tap delay unit, while the bottom section contains two independent crossfaders.

The design of this unit forgoes a dedicated Feedback control and instead uses three-channel mixer as the delay's input. The intent is that the individual delay outputs can be plugged into the extra mixer channels, allowing the user to explicitly design the feedback path. The crossfaders on the bottom can create submixes of any of the two outputs, and can be plugged back in to the input mixer to shift feedback emphasis between taps.

This design is independently polymorphic, as the crossfaders are completely unconnected from the delay's outputs. The delay can be used while ignoring the crossfaders, and the crossfaders can be used separately from the delay.

2.2.1.6 Linked Polymorphism: Folktek Conduit

The Folktek Conduit [12] is an example of Linked Polymorphism. The Conduit is a complicated design that combines a filter, a delay, and a number of oscillators that are a byproduct of the delay process (thus, this module can also be considered

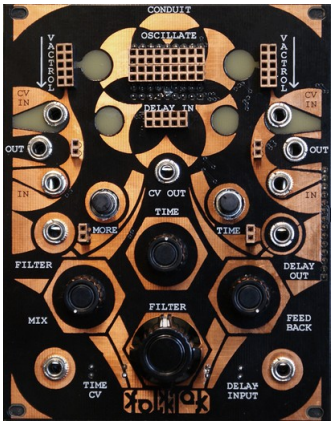

Figure 2.6: Folktek Conduit. Note the separate Filter and Delay sections, along with the dedicated Delay output.

simultaneously polymorphic). The important part of this design is that the delay's output is normalled to the filter's input, thus creating a more complex signal path. However, the delay has a dedicated output, and the filter has its own input and output. Thus, the two sections can operate separately. Going back to the example of the Make Noise Echophon, the Echophon has a pitch shifter and a delay, but does not exhibit Linked Polymorphism. If the shifter and the delay could be used independently of each other, then it would qualify here.

2.2.1.7 Simultaneous Polymorphism: Make Noise Echophon

The Make Noise Echophon [13] is an example of Simultaneous Polymorphism. The delay half has an almost identical feature set compared to the Dub Jr., as it has tempo sync and tapped feedback. Even though the Echophon has a pitch shifter, this is not what makes it polymorphic (The pitch shifter is part of the algorithm and is not accessible separately). The feature of the Echophon that demonstrates polymorphism is the "CLK OUT" jack. Like the Dub Jr., the Echo knob becomes a "Div" knob when a signal is present in the Tempo Sync jack. This sets the delay's

Figure 2.7: Make Noise Echophon. Note the CLK OUT jack in the top right.

length to some metric division of the tempo present on the TEMPO jack. The CLK OUT jack outputs a clock that has the same period as the delay and the same phase as the Tempo jack. Because of this one simple feature, the Echophon becomes simultaneously polymorphic. It can be used as a clock generator, divider, or multiplier while ignoring the delay features entirely.

2.2.2 Analyzing Generators: Oscillators

2.2.2.1 Monosemous: Snazzy FX Dronebank

The Snazzy FX Dronebank [14] is a module that houses five Monosemous oscillators. These oscillators have a fixed timbre. The pitch of each oscillator can be controlled only via a front-panel knob. The oscillators can be output separately or via a MIX output (each oscillator has a constant amplitude at the MIX output).

Figure 2.8: Snazzy FX Dronebank.

Figure 2.9: Make Noise STO.

Figure 2.10: WMD Synchrodyne with Expander.

2.2.2.2 Rhizomatic: Make Noise STO

The Make Noise STO [15] is a Rhizomatic oscillator. Many oscillators could be considered Modally Polymorphic depending on their operating range. The STO operates from about 8 Hz to 4 kHz without external voltages. Since it cannot function as an effective LFO on its own, I am not including it in the Modal Polymorphic category.

However, it has a number of features that make it Rhizomatic. First, it has two varieties of FM (linear and exponential) in addition to a standardized 1v/oct pitch input. It also has the ability to modulate waveshapes. Finally, it has multiple waveform outputs including a gated sub-oscillator output. These features encourage alternative patching strategies including plugging the STO into itself to generate unpredictable waveforms, or linking the S-Gate to a clock to create a synced, staccato bassline.

Figure 2.11: Mutable Instruments Braids.

2.2.2.3 Expandable: WMD Synchrodyne

The WMD Synchrodyne [16] is a complicated oscillator that is also an example of Linked Polymorphism. In its base form, it is an oscillator that drives a Phase Locked Loop (PLL). The PLL, in turn, drives a switched capacitor filter that is used to filter the oscillator. Each of these three sections can be broken out and used on their own. The Synchrodyne has one of the most unusual expanders in Eurorack, as the expander is larger than the Synchrodyne itself [17]. It adds a large amount of functionality, including a second VCO, filter, and PLL, along with a compressor, a wavefolder, and more control over the original Synchrodyne circuit. The expander cannot be used on its own.

2.2.2.4 Modal Polymorphism: Mutable Instruments Braids

The Mutable Instruments Braids [18] is a self-described "macro oscillator," a perfect example of Modal Polymorphism. It features a prominent encoder that allows a user to choose between over 40 modes of sound generation. It includes many synthesis methods (FM, wavetable, microsound) and options for generating noise or percussion.

Braids is functionally uniform, as every mode is an audio oscillator. Braids is

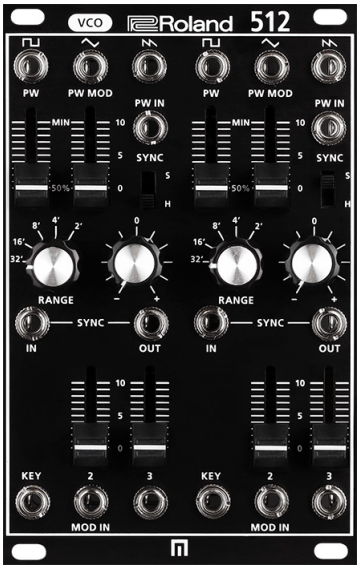

Figure 2.12: Roland System-500 512 VCO

nearly terminally uniform. Almost every mode generates continuously without the need for excitation. However, a few of the percussion modes require the use of an additional trigger input to generate sound.

2.2.2.5 Independent Polymorphism: Roland System-500 512 VCO

The Roland System-500 512 VCO [19] is a rare example of Independent Polymorphism in oscillators. I say "rare" because almost every other hardware modular with two or more oscillators has at least some form of normalization between the oscillators. This module contains two completely independent wide-range oscillators. The polymorphism stems from the fact that one oscillator can be used as an audible-range oscillator while the other could be an LFO.

2.2.2.6 Linked Polymorphism: Intellijel Atlantis

The Intellijel Atlantis [20] is a self-contained subtractive synthesizer. It is a great example of Linked Polymorphism, as every section can be used completely inde-

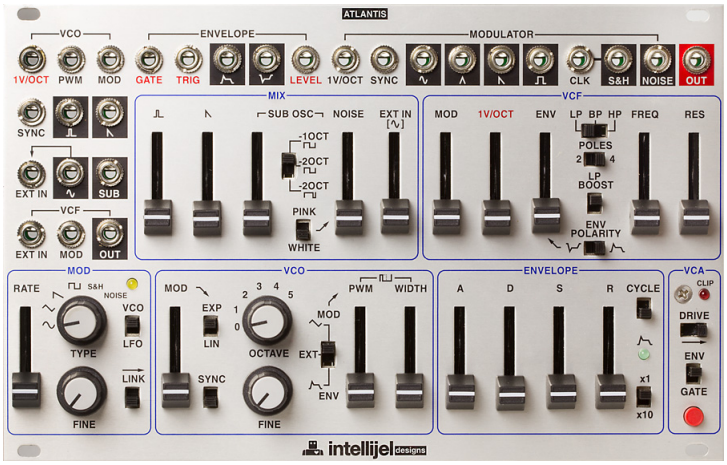

Figure 2.13: Intellijel Atlantis.

pendently. For instance, a composer can choose to use only the envelope generator or filter when the oscillator is not needed. However, without any patching, it will behave like a more traditional standalone synthesizer.

2.2.2.7 Simultaneous Polymorphism: The Harvestman Piston Honda MK II

The Harvestman Piston Honda MK II [21] is a wavetable oscillator that exhibits Simultaneous Polymorphism. There are two outputs: "External" and "Internal". The "Internal" output uses an internal phasor, meaning that the Piston Honda can act as a standalone wavetable oscillator. It also as an "Ext. In" section that acts as a separate input for wavetable lookup, the output of which appears at the "External" jack. This is considered Simultaneous Polymorphism (and not Independent or Linked) because this external waveshaper and internal oscillator share the same wavetable selection controls.

Figure 2.14: Piston Honda Mk. 2

2.2.3 Monosemous Polymorphism Example: Mutable Instruments Peaks

Peaks is a multi-mode trigger processor from Mutable Instruments [22]. It contains two channels, each of which have a single trigger input and a single output. There is no CV control. Peaks has four primary modes: AD envelope, LFO with reset, tap-tempo LFO, and drum synthesizer. There are a number of alternate modes and easter eggs that are not displayed on the panel graphics. Despite the flexibility introduced by modal polymorphism, the individual channels of Peaks are monosemous in nature.

Since every mode uses one trigger input and one output, Peaks displays terminal uniformity. However, since each mode has a different purpose (especially the drum synthesizer modes vs. the modulation generators), Peaks does not display functional uniformity.

Figure 2.15: Mutable Instruments Peaks.

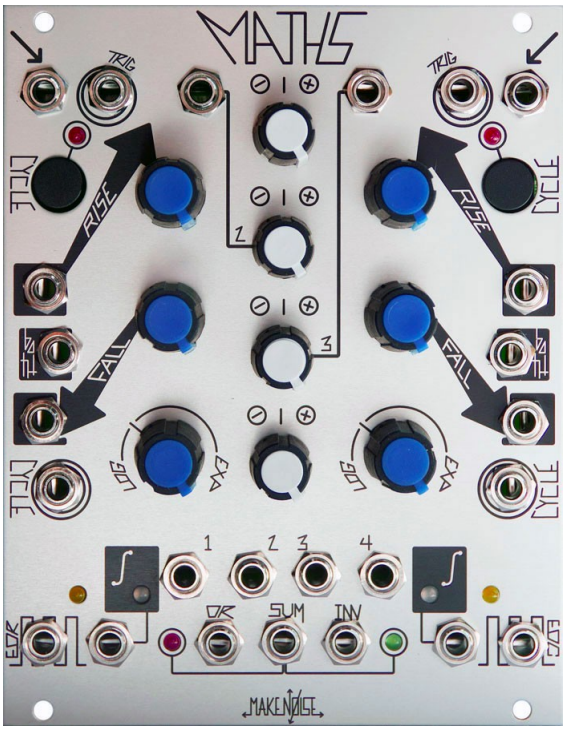

Figure 2.16: Make Noise Maths (2013 Revision).

2.2.4 Combination Polymorphism Example: Make Noise Maths

The Make Noise Maths is a highly flexible Eurorack design, combining multiple forms of polymorphism into one very dense design [23]. Maths is an update of the Serge Dual Universal Slope Generator (DUSG) [24], a classic design where each section can behave either as an AD Envelope or a Bipolar Slew Generator. Each section of the DUSG produces a trigger when finishing a "fall". This can be patched back into the Trig In input to produce cycling behavior, thus turning each section into an oscillator.

Maths builds upon this design by adding in a four-channel bipolar mixer (with optional offset generation and logical outputs) and toggled cycling modes. By the standards outlined by my above taxonomy, I would classify Maths as Modally, Linked, and Simultaneously polymorphic.

- Modal: Pushing the "Cycle" button sets the corresponding channel to oscillate. Each channel can function as a standalone oscillator, LFO, envelope, or slew generator depending on the state of CYCLE and the attack/decay speeds.

- Linked: The outputs of channels 1 and 4 (the envelope/slew/oscillator channels) are normalled into the four-channel mixer.. Offset generators are normalled to the inputs of channels 2 and 3.

- Simultaneous: One channel has an "End of Rise" gate output, while the other has an "End of Cycle" gate output. These can be used for a number of simultaneous functions, including a master clock, a square wave LFO with variable PW (width can be set by manipulating the relationship between rise and fall times), and a gate delay (End of Rise provides a delay from the moment that an incoming gate goes high, End of Cycle provides a delay from the moment

that an incoming gate goes low).

2.3 Design Limitations

There are three major platforms for modular design: hardware analog, hardware digital, and software digital. Each of these platforms introduces important design constraints, which will be outlined below.

It is important to note that these platforms are not representative of the entire module, but rather the generation and/or processing sections. For instance, a hardware module built around a digital platform still needs analog sections for acquiring input voltages and restricting them to levels that are safe for the digital processor.

Multiple platforms can be used on a single module for processing and/or generation as well. As an example, the Intellijel Shapeshifter uses a digital platform for generating signals and an analog section for wavefolding the signals [25].

2.3.1 Hardware Analog

Precision is Expensive For sections where precision is a priority, expensive parts are required. Examples of precision sections include stable tracking of 1V/Oct signals across multiple octaves, sampling and holding voltages with low drift, stable clocks without timing jitter, and more. Precision-matched transistors, temperature compensated resistors, and other parts can quickly drive up the cost of a module's BOM.

Physical part counts Analog modules typically require more components than digital modules. This increased parts count creates a number of issues. First, having more parts will lead to a more complicated layout, resulting in more time spent by

the designer placing parts on the PCB, and more space taken up by parts (leading to either larger PCBs or stacked PCBs). Second, having more parts creates the potential for more places for a circuit to fail, along with the potential for stock of a specific part to disappear. This can force a designer to either redo or cancel a design.

2.3.2 Hardware Digital

Input Resolution Most affordable processors and microcontrollers have ADCs (Analog-to-Digital converters) with sampling rates around 3000 Hz. This leads to aliasing for sections like FM inputs on oscillators and filters. Higher sampling rates can be problematic for two reasons. First, the parts cost can increase rapidly. Second, many high sampling rate ADCs are AC-coupled, meaning that DC signals (including slow modulation) will be filtered out.

CPU/Storage Limitations When designing an algorithm, a programmer can run up against two major limitations on the microcontroller itself: the processor's speed and the amount of storage available for instructions. Storage space can run out quickly if the algorithm depends on sets of supplemental data, like wavetables.

DACs Digital-to-Analog converters are expensive, especially if a designer needs more than two high resolution audio outputs. This can make it difficult for simultaneously polymorphic digital hardware. The primary exception to this rule is that it is cheap to add binary outputs, like gates and triggers.

Power Consumption Digital modules typically have higher power requirements than analog modules. For many users, their cases are not able to provide enough

power to run these modules. Furthermore, some digital Eurorack modules are designed to require +5V on the power bus, which many cases do not have.

2.3.3 Software Digital

System Compatibility A software modular needs a stable system to run on. A software modular can fail from a hardware incompatibility, an operating system update, a bad audio driver, etc.

CPU Limitations Unlike hardware digital modules (where the CPU requirement of a single algorithm is known), in a software modular each module additively affects the CPU. As patch complexity increases, so do the CPU demands. The design of individual modules can often come down to a balancing act of features and CPU demand.

User Control For a hardware module, a static interface is provided by the designer. Software modulars can vary greatly in this aspect. In software modulars like Max/MSP, PureData, SuperCollider, and CSound, an interface has to be created by the patch designer instead of the module designer (i.e. each object or uGen doesn't come with standard sliders, knobs, buttons, etc.; they are separate objects). Other software modulars like Reaktor Blocks, WREN, and OSCiLLOT provide per-module software interfaces. In both instances, if a user wants external hardware control (via MIDI or OSC), the user must create this mapping themselves.

2.4 Project: Hardware Module Designs

As part of my research, I've ended up creating a number of proposed hardware module designs, some of which are currently in active development. Due to the public nature of this document and the commercial nature of these designs, I will be presenting these designs only during my dissertation defense.

Chapter 3

Taxonomy #2: Methods of Control

3.1 Classification of Control Methods

Simply outlining a patch's connections is not a sufficient enough explanation for how the patch is actually performed. A modular can be controlled in many ways. In this section, I will outline a number of control strategies and describe currently existing examples from Eurorack.

3.1.1 Control Modules

There are many modules that bring useful control methods to a modular system, including joysticks, keyboards, pressure-sensitive pads, contact microphones, motion sensors, gesture recorders, and more. These are modules that exist within a modular case, are powered by the modular case, and are meant explicitly to control other modules.

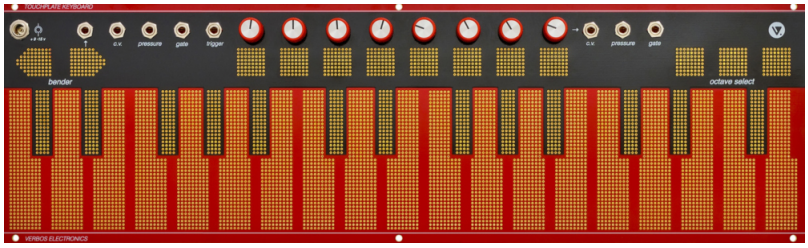

Figure 3.1: Verbos Electronics Touchplate Keyboard. The "keys" are capacitive and flat. In addition to the standard keyboard interface, a number of manual voltages are available on the top row.

3.1.1.1 Modular Keyboards and Pressure Sensitive Pads

Module-based keyboards exist in many forms. Some, like the Verbos Controller keyboard [26] or Sputnik keyboard [27] use a traditional piano layout (albeit with capacitive keys instead of physical keys). This traditional layout is welcoming to keyboardists, but also adds new capabilities like continuous pressure sensitivity, fine pitch adjustment, easy pitch slides, built-in arpeggiators, and more. As a drawback, they tend to take up a lot of space in a system. Many of these modules can be powered outside of the modular as well, placing them in the same category as Section 3.1.3.1 (External Modular-Compatible Controllers: Dedicated Controllers). One great side effect of having a traditional piano layout is that it enables a composer to use traditional music notation to explain to a user how to perform a patch.

In addition to the traditional piano layout, there exist alternative "keyboard" formats, like WMD's upcoming Poly Pressure Array [28], which uses an unusual layout of square, rubber keys. This interface allows users that are not well-versed in piano to still quickly experiment with chords and scales by using geometrical shapes. There also exist simplified layouts, like Make Noise's Pressure Points [29]. Each Pressure Points contains four "keys", and each key outputs five separate voltages, three of which are user-selectable (the fourth is a pressure-based voltage, while the

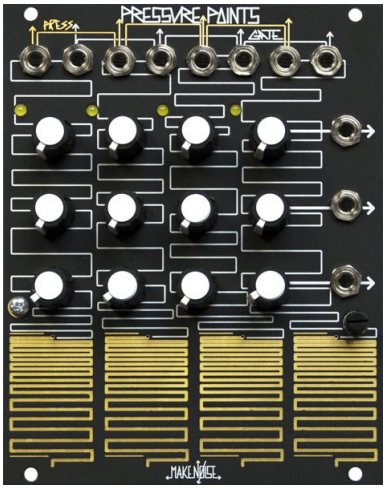

Figure 3.2: Make Noise Pressure Points. This has pressure-sensitive "keys" on the bottom. Instead of using a traditional keyboard layout, each key outputs a pressure voltage, a gate, and three manually set voltages. Only one key stage can be active at a time.

fifth is a binary gate). This design splits the difference between flexibility and size. A user who needs more keys can buy multiple Pressure Points and chain them together. Due to the abstract nature of the module, it is harder to share patch notes via common music notation.

Some modules, like the Synthwerks FSR series [30] use force-sensitive resistors to output a voltage based on how much the user presses on it, but remove the concept of the keyboard and stored voltages.

3.1.1.2 Joysticks and Gesture Recorders

A number of modules provide a joystick or pressure grid interface for interacting with patches in multiple dimensions. The Intellijel Planar [31] and Flight of Harmony Choices [32] are examples of physical joysticks that output X and Y positions as voltages. The Soundmachines LP1 Lightplane [33] is an example of an X-Y grid that records a users touch gestures. In addition to reading where the user's finger

Figure 3.3: Intellijel Planar, here demonstrated with two different faceplates. The module can be inverted to help avoid cables from physically interfering with the user's range of motion.

appears on an X-Y grid, the module also detects pressure. This gives the user three simultaneous dimensions of control. Furthermore, these three dimensions can be recorded and played back as looping gestures, freeing the user to interact with other components of a patch.

3.1.2 External Modular-Compatible Controllers

These are controllers that exist outside of the modular system. They can still connect directly to a modular via CV outputs and do not need intermediate devices for translation.

3.1.2.1 Dedicated Controllers

These are controllers that exist outside of the modular, but contain control voltage outputs that are calibrated specifically to work with a modular system. These include keyboards (like the Keith McMillen QuNexus), trigger sequencers (Arturia Beatstep Pro), note sequencers (Korg SQ-1), motion controllers (Koma Kommander), and more. These devices can not generate audio on their own.

Figure 3.4: Keith McMillen QuNexus, here interfacing directly with a Eurorack system via CV outputs. The QuNexus can connect to a computer via USB and provide MIDI-over-USB.

3.1.2.2 Instruments with CV Capabilities

In addition to dedicated controllers, there are many standalone instruments that have control sections that can interface with a modular. As an example, the Elektron Analog Four [34] is a self-contained synthesizer with four complete analog voices. It has two separate stereo CV outputs that can be used for pitch, gate, and two modulation signals. The Analog Four's internal sequencer can be used to sequence the Analog Four by itself, the Analog Four and modular simultaneously, or the modular by itself. Note sequences from the Analog Four are converted to the 1v/oct format on the pitch output.

Other instruments, such as the Teenage Engineering Pocket Operators [35] or Korg Volcas [36] have analog trigger inputs and outputs for syncing a modular to their internal clock, or syncing to a modular via a clock input.

Figure 3.5: Mutable Instruments Yarns. This module converts MIDI messages into control voltages and gates.

3.1.3 Translator Modules for External Controllers

These are modules that exist within a modular system (i.e. screwed into the modular rack and powered from the modular's busboard). However, they plug into external controllers that are otherwise independent from the modular.

3.1.3.1 MIDI and OSC

A number of modules are capable of converting various signals to modular-compatible voltages. The most obvious instance of this is MIDI, as there exist numerous modules capable of converting MIDI note messages to control voltages and gates. Some of these MIDI converters will also translate CC messages to CV. Mutable Instruments' Yarns [37] is a notably complex example, as it features many different methods for interpreting MIDI inputs, including 4 voices of pitch and gate or 1 voice with many parameters. Simpler modules, like Doepfer's A-190-2 [38] or Pittsburgh Modular's MIDI 3 [39], simply read one or two channels of note and gate data.

Modules like the Synthtech e620 [40] or Expert Sleepers FH-1 [41] act as hosts for any MIDI-over-USB compatible hardware device. These modules break out specific MIDI CC values and messages to jacks. A newer Monome module, Ansible, connects

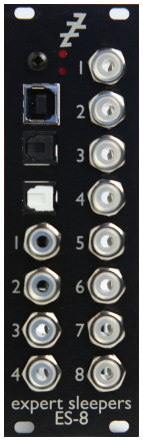

Figure 3.6: Expert Sleepers ES-8 USB Interface. Note the USB port in the top-left. This module provides DC-coupled inputs and outputs.

to Monome's proprietary Grid and Arc hardware devices, along with any MIDI-over-USB device [42]. This is the only module currently out that can change between OSC and MIDI protocols.

3.1.3.2 Software to CV

Many modules provide deep integration with a computer, going beyond the low resolution that MIDI offers. In general, a computer that has an audio interface with DC-coupled outputs is capable of sending CV to a modular. However, this is a fairly rare feature for audio interfaces. As a workaround, a company called Expert Sleepers makes a series of modules designed to connect to computers or audio interfaces in various ways, depending on what the user has available [43]. For instance, the ES-3 [44] uses an audio interface's optical output to produce 8 DC-coupled 1/8" outputs. The ES-4 [45], meanwhile, uses S/PDIF. His newest, the ES-8 [46] connects directly to a computer using USB, providing DC-coupled inputs and outputs.

If a user has only analog, AC-coupled 1/4" jacks, Expert Sleepers also makes a software plug-in suite called Silent Way [47]. This suite has an AC Encoder plug-

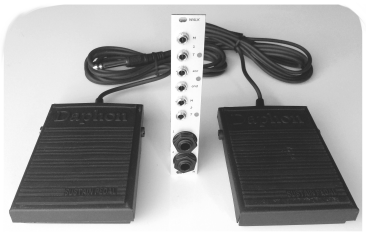

Figure 3.7: Monome Walk. This module connects to two sustain pedals and creates six logical outputs.

in that adds rapid modulation to a CV signal to allow it to be output from an AC-coupled interface. A partner module, the ES-1 [48], demodulates this signal to produce a clean CV signal.

Furthermore, the rest of the Silent Way package is designed for generating and processing CV's, including plug-ins for creating LFOs, envelopes, trigger sequences, and more. However, Silent Way is not required for these purposes. An experienced user can use Max/MSP, Reaktor, Supercollider, or a number of other software packages to generate control signals on a computer. My Reaktor 6 package, Euro Reakt, is fully capable of interacting with a Eurorack system.

3.1.3.3 Non-proprietary Controllers and Sensors

Some modules are designed to hook up to non-proprietary controllers and sensors. For example, the Monome Walk [49] is a module that is designed to connect to two sustain pedals to create six logical outputs. Many keyboard manufacturers create sustain pedals, so a user can select their favorite.

The Eowave EO-310 Sensor Signal Processor is a module designed to connect with any sensor with a voltage range between 0 and 5 volts [50]. Eowave manufactures a few sensors to easily connect to this module, but an experienced user can solder a 1/8"

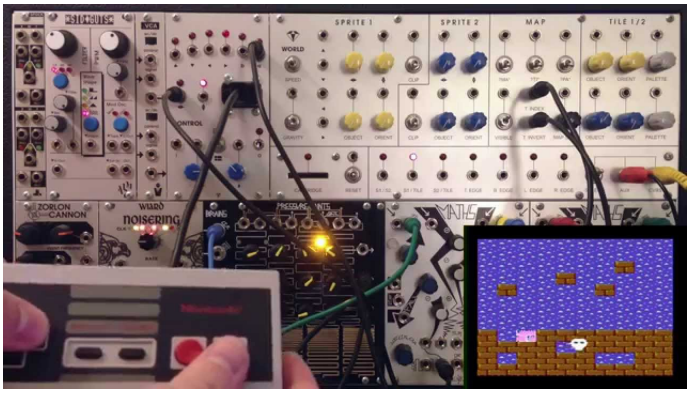

Figure 3.8: Ming Mecca Control Core (top middle) being controlled by a Nintendo NES controller.

jack onto most standard sensors (such as FSRs, photosensitive resistors, moisture sensors, etc.).

The Monome Teletype is a module that connects to any QWERTY keyboard or numpad [51]. The keyboard is used to program control scripts on the module that generate trigger/gate and CV sequences. A user can manually trigger scripts directly from the keyboard or numpad.

3.1.3.4 Proprietary Controllers

Other modules receive control from proprietary formats such as video game controllers. The ADDAC302 "NCHUNK *(sic)* CONTROL" [52] receives messages from a Nintendo Wii Nunchuk joystick, while the Special Stage Systems Ming Mecca Control Core [53] connects to a Nintendo NES controller. These modules use the proprietary jacks that these controllers require instead of a more open format like USB.

The Monome series of modules is an interesting case. They use open-source firmware and communication protocols, but are programmed to host one specific piece of hardware. In the case of the Monome modules, these modules use bi-

Figure 3.9: The Harvestman English Tear. This module acts as a translator between the 1 Volt Per Octave and Hz per Volt control voltage standards.

directional OSC to receive button presses from a Monome Grid or encoder information from a Monome Arc [1], which will then receive lighting information from the module [55].

3.1.3.5 Voltage Translators

A few modules are able to convert between various voltage standards. As an example, The Harvestman's English Tear [56] is a Eurorack module that converts 1v/oct pitch signals to the Hz/V standard used by the Korg MS series of synthesizers (and vice versa). This allows a user to use the keyboard on a Korg MS-20 (for example) to control a Eurorack modular in a predictable manner.

The discontinued Format Jumbler by Make Noise [57] was a panel that combined banana jacks, 1/8" jacks, and 1/4" jacks as a passive format translator. It did

[1] It is worth noting that the Monome modules are open-source, and users have created alternate firmwares capable of using other MIDI-over-USB controllers. The Monome Earthsea was originally designed only to interact with the Monome Grid. A user later added in the ability for it to interact with any MIDI-over-USB keyboard. Another notable example, the Orca [54], replaces Monome's White Whale module's firmware with a completely different program that is compatible with the Monome Arc or Grid, presenting alternate interfaces for each.

not provide voltage scaling, but it allowed for an easy connection between various modular formats.

3.1.4 The Modular as a Controller

Finally, the modular system can be used to control other devices, whether directly with control voltage or through the use of controllers. As previously mentioned, there are many modules that receive a MIDI input and convert it to various voltages. The inverse is true as well, as there exist modules and external devices that convert voltages to MIDI signals. This can be useful to many users, as modular synthesizers are capable of generating extremely complex sequences that would be difficult to produce using other means.

The main difficulty is figuring out how to convert the modular's signals into another format. Many of the previously mentioned devices allow for bi-directional communication. In the case of the Keith McMillen QuNexus, it receives CV signals and outputs MIDI-over-USB [58]. The Harvestman's English Tear can be used to convert 1v/oct signals to Hz/V, allowing a Korg MS-series synthesizer to be controlled by a modular [56].

The BEMI Buchla LEM3 Spider is one of the only complete modular systems sold with the intent of being used solely as a controller [59]. It consists of a Buchla 252e Polyphonic Rhythm Generator, a 226h CV-to-MIDI Interface, and a 225h MIDI-to-CV Interface. It can connect to a computer over USB or to other devices using a 5-pin MIDI output.

In addition to the previously mentioned Expert Sleepers output modules, he also makes a series of modules designed to take voltages from a hardware modular and send them to a computer. The ES-6 [60] is a partner module to the ES-3 and provides

Figure 3.10: The Buchla LEM3 Spider

six DC-coupled inputs that connect to an audio interface using an optical cable. The previously mentioned ES-8 connects to a computer directly over USB, skipping the need for an existing audio interface. It provides 8 outputs from the computer and 4 inputs to the computer.

3.2 Project: Simple MIDI for Max

Simple MIDI is a free package that I have published for Max 7. Max is extremely flexible in regards to external input, as it can communicate with MIDI, OSC, and a number of other communication protocols (DMX, HID, etc.).

Unfortunately, this flexibility and power leads to long setup times for every patch that you wish to use OSC and/or MIDI with. The typical Max way of doing things is to use the "route" object to take large input messages and route them based on input matching. For instance, if an OSC device is communicating with Max and sending messages to "/maxosc/knob1/25" you could create an object that says "route

/maxosc/knob1/" to listen specifically to the value of that address.

To create a simple MIDI patch, you need to setup a large number of routing objects. Let's say you want to listen to the value of a single knob on an external MIDI controller. First, you would need to look up the CC address of the knob in your controller's documentation (or experiment with printing every incoming MIDI message to the Max Console Window). After you know your MIDI knob's CC value (for this example, it will be CC #33), you'll need to setup a listener. The "midiin" object specifies which MIDI device to listen to. This object is then connected to a "midiparse" object, which selectively routes all MIDI messages from a device based on whether the message is a CC value, a Note value, velocity, pitch bend, etc. This "midiparse" object has 8 outputs, including a separate output for the message's channel number. In the simplest case scenario (where you would want to listen for CC #33 on *all* MIDI channels), you would now connect a "route 33" object to the third output of "midiparse". Only now will you have the value of one knob. If you are listening on multiple channels, this becomes even more complicated.

This amount of complication impedes an artist or composer's flow. A more experienced Max user could create a Max "clipping" to automatically paste a number of route messages based off of their favorite MIDI controller. However, this solution would only work for one specific controller and would need to be rewritten if the user wants to use a different controller or different mapping for the same controller.

My solution for this is Simple MIDI. Simple MIDI is organized as a Max "package," a new method of organizing files for easy installation and management introduced in Max 7. When a user installs Simple MIDI, a number of "clippings" are installed. These clippings show up under a "Simple MIDI" menu when a user right-clicks on an unlocked patch. The user can then add a Simple MIDI object. If a user doesn't know

how to interact with the object, there is an interactive help patch that is installed to the Max "Extras" menu.

The method of interaction is extremely simple. The user adds Simple MIDI to a patch. To add MIDI control to a Slider, Dial, or Button widget, the user simply clicks the "X" above the object that they wish to control. After a MIDI control is moved, the Max widget is then mapped to that external control. This control remembers the CC# and channel, so multiple controllers and channels can be used. The channel and CC# mappings are then saved with the patch, allowing for control sets to be quickly recalled.

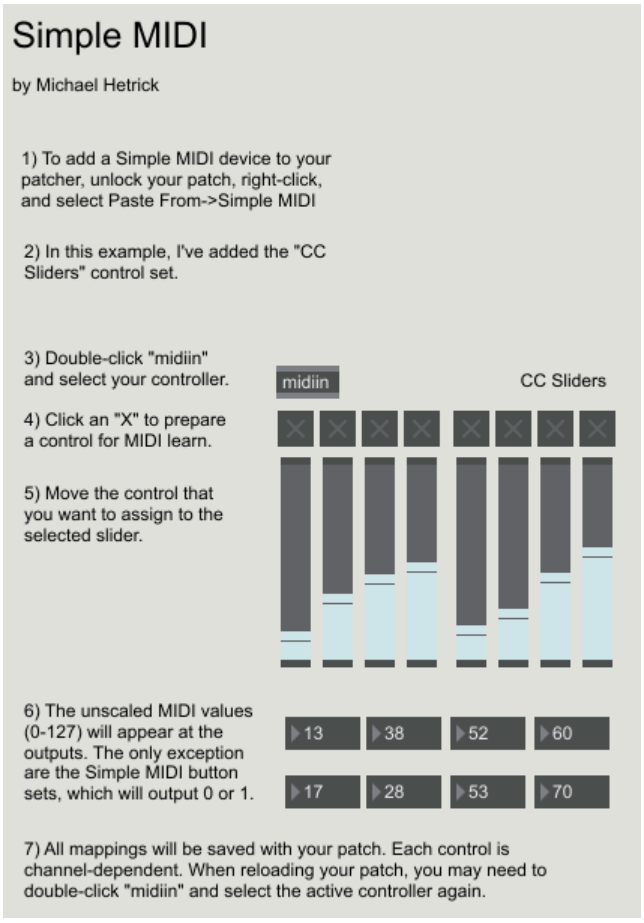

Figure 3.11: The Simple MIDI help patch. This interactive patch is quickly available under the Max "Extras" menu after installing the package.

Chapter 4

Taxonomy #3: Patching Strategies

4.1 Classification of Patch Types

One of the most difficult aspects of modular synthesis is managing complexity. As a patch grows larger, it becomes more difficult to remember which aspects of the patch are responsible for each interaction. It also becomes more difficult to document the patch for preservation. In this chapter, I will present two tools for managing complexity. The first is a taxonomy to categorize the primary elements of a patch. The second is the concept of "Meta-Modules", a tool for examining synergies between common low-level functions that exist in most modular environments.

4.1.1 Primary Elements

I've decided to place all modules in a patch into three simple categories: Voice, Modulation, and Timing. These groupings are more of a guideline than a hard rule, as individual modules can serve in all three categories in a single patch (especially if the module is polymorphic).

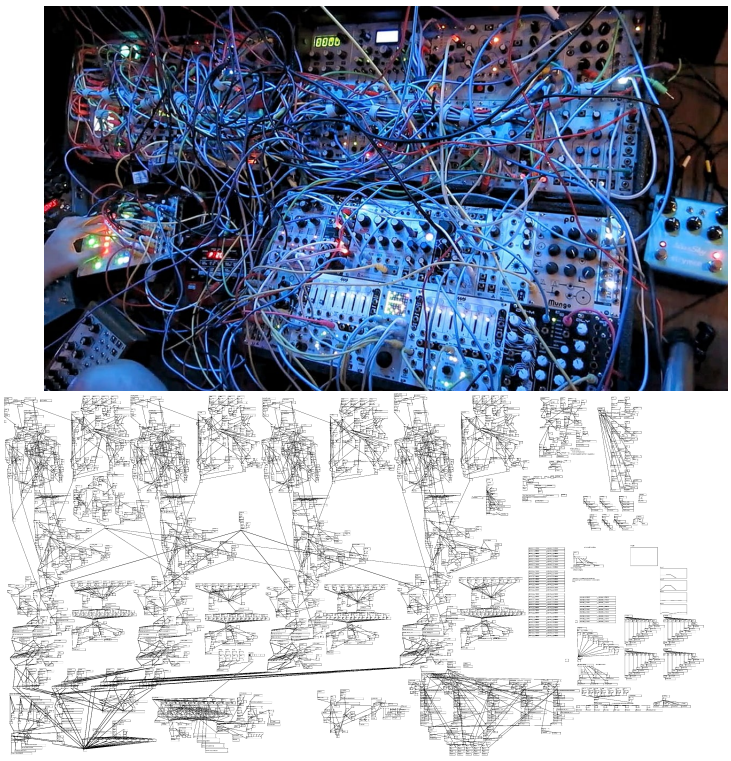

Figure 4.1: Complex patches present issues in hardware and software environments.

4.1.1.1 Voice

This grouping includes all modules that are audible in a patch. Typically, this would include oscillators, filters, and other effects. This does not include audible devices that a modular system could interact with, like guitar pedals or other synthesizers.

4.1.1.2 Modulation

This grouping includes all modules that are used to modify the parameters of other modules (or themselves). Typically, this would include LFOs, envelopes, step sequencers, and controllers. Audio-rate modulation (like FM and AM) would be included here. An audible oscillator with an active internal modulation bus would count as Voice and Modulation.

4.1.1.3 Timing

This grouping includes all modules that are used to generate gates and triggers. Typically, this would include clock generators, clock dividers, boolean logic, and other timing modules. If a slow timing signal is derived from an audible oscillator via a clock divider, then both the oscillator and the clock divider would be in this category.

4.1.2 Classifications

The three categories of primary elements are not required in every modular patch. As a result, there are seven possible combinations of the basic elements. In this section, I will list the seven combinations along with basic example patches. These patches do not cover every possible strategy for each category.

4.1.2.1 Voice Only: Static Drone, Tone Cluster

A Voice Only patch consists of an arbitrary number of unchanging sounds. A single, unmodulated voice would be a static drone. With at least two voices, perceived modulations like frequency beating could occur. A cluster of voices could be combined to create chords or complex timbres.

4.1.2.2 Modulation Only: Controller for External Source

A Modulation Only patch would likely only be a Timing-free patch that is modulating another device. For instance, an LFO could be used to control the depth of a guitar pedal.

4.1.2.3 Timing Only: Clock for External Source

A Timing Only patch would consist of a modular synthesizer providing a clock for other devices. This could be a single, stable, metronomic clock. With one clock and one clock divider, a modular synthesizer could distribute multiple clocks among devices. Even without modulation, a multiple stable clocks, dividers, and boolean logic modules could be combined to create extremely complex gate and trigger patterns.

4.1.2.4 Voice and Timing: Drums/Triggered Voices, Metronome

A Voice and Timing patch could consist of a complex Timing section being used to trigger a Voice section made up of multiple, unmodulated drum modules (effectively a modular drum machine). Alternatively, the simplest Timing section (a single, stable clock) could be used to create an audible metronome Voice.

4.1.2.5 Timing and Modulation: Variable Clock Source, Synced Modulation

A Modulation section could drive a clock's frequency, stepping between multiple tempi. Alternatively, a Timing section could be used to sync a Modulation section to some rhythm or steady tempo. Either way, these sections would then interact with an external device.

4.1.2.6 Voice and Modulation: Animated Drone, Manually Performed Patch

A Modulation sections could animate various parameters on a Voice, such as the index of an FM voice or the active wavetable of a digital oscillator. This category also includes Voice sections manipulated by non-Timing-based modular controllers (keyboards, joysticks, etc., but not sequencers).

4.1.2.7 Voice, Modulation, and Timing

The majority of patches will fall within this category.

4.2 Meta-Modules

One particular difficulty exists in hardware modulars that doesn't in software: sharing patches. In a software modular environment, it is typically very easy to save a patch and send the file(s) to other users to open using the same environment. In a software environment, these patches can be used to train users on how to use the environment. As an example, Euro Reakt (see Chapter 6) contains over 40 interactive examples that teach users helpful interactions between the system's modules. In

Cycling 74's Max 7, every single object comes with an interactive help patch showing useful synergies between other Max objects [61].

Hardware lacks the uniformity necessary for this pedagogical nicety. Since a user is free to choose whatever modules make up his or her system, it becomes very difficult to create documentation that covers every possible configuration. Many hardware module manuals include simple descriptions of a module's inputs, outputs, and controls, but fail to include suggestions on how to use the module with others. The only exceptions are pre-made modular systems like the Make Noise Shared System [62] and Pittsburgh Modular Foundation [63], both of which come with detailed manuals outlining a number of starter patches.

A "Meta-Module" is a term that I coined to describe a useful synergy between at least two basic, low-level functions. I created this concept to assist with both the design of modules and the pedagogy on how to use them.

These low-level functions are functions that are present in practically every modular system. Examples include: sample & hold, slew, mixer, noise, comparator, clock, VCA, oscillator/LFO (basic waveforms like sine, saw, triangle, square), filter, AD envelope, delay, attenuator, boolean logic, analog logic, step sequencer, clock divider, rectifier, switch, and more. As an example Meta-Module, a random source could be created by combining noise, sample and hold, and a clock.

The purpose of a Meta-Module is to separate the concepts of the patch from their specific implementations. By breaking a patch down into Meta-modules, a composer can teach other composers how to create similar gestures or sounds without requiring the purchase of specific equipment.

Meta-Modules benefit module designers in two ways:

- Meta-Modules are the foundation of good polymorphic design. Almost ev-

ery linked, simultaneously, or independently polymorphic module is designed around a Meta-Module (modally polymorphic modules being the exception).

- A designer could use Meta-Modules to improve the documentation of their product by suggesting synergistic module types.

My original goal for this dissertation was to write a book that served as a large catalog of Meta-Modules. However, this proved to be too difficult of a task. Instead, I have focused on spreading the idea of Meta-Modules through Euro Reakt's documentation and my open-source compositions.

4.3 Project: Open-Source Compositions

Starting in 2014, I released a series of compositions for Make Noise Records [64]. These were the first digital releases in their "Shared System Series," a collection of recordings intended to show off the power and flexibility of their Shared System[1]. One track, "Late Bloomer," was included in *Shared System Series Side A*, the first compilation of Shared System recordings [65].

With each released recording, I included a patch diagram created using the Modular Grid website [1]. These diagrams were publicly available to all listeners directly from the Soundcloud page. In addition to the diagrams, I created a forum topic [66] to assist any users interested in recreating the sounds found in each composition.

[1]The Shared System is a specific collection of modules curated by Make Noise founder Tony Rolando. The original concept was one system that was sent to various artists. Their only addition could be an external reverb. Since the original concept, the Shared System Series now focuses on recordings made with Make Noise equipment.

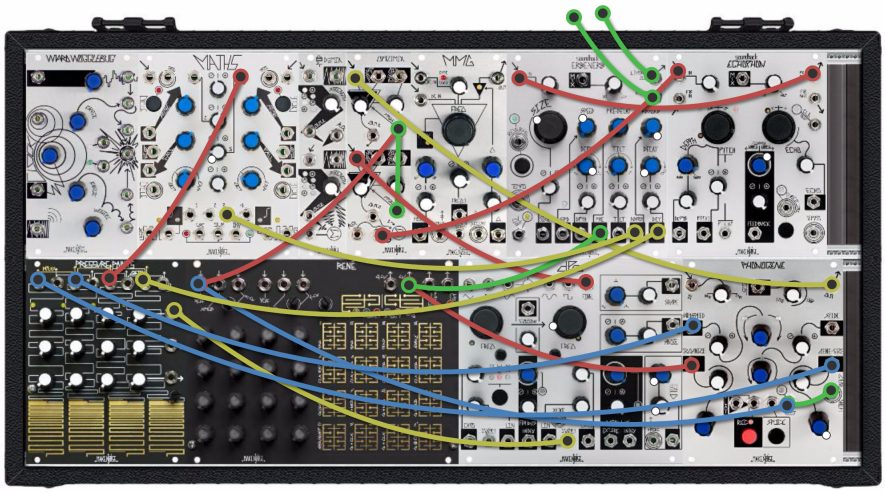

Figure 4.2: "Dependence" Shared System patch diagram.

4.4 An Example Analysis: "New Leaf"

On April 15th, 2015, I put on a live modular performance as part of UCSB's CREATE concert series [67]. My piece, "New Leaf," was fully documented through patch diagrams and meta-module analysis, which I have copied below. Here, you can see how the above categories can be used to break a very complicated patch down into simple, reusable elements. A full recording of the live performance of this patch is available on my Soundcloud page [68].

4.4.1 Module Breakdown

Timing:

- Intellijel Shapeshifter (Pulse output)
- Make Noise Wogglebug (Pulse output)

Voices:

- Make Noise Mysteron

Figure 4.3: "New Leaf" Patch diagram. Modular Grid [1], a popular community website, is used for producing these images.

- Mutable Instruments Elements (Generator and effect)
- Music Thing Modular Radio Music
- Intellijel Shapeshifter (Oscillator output)
- Make Noise RxMx + FxdF (Mixer and filter)

Modulation:

- Mutable Instruments Frames
- Intellijel Shapeshifter (LFO output)
- Intellijel Planar
- Make Noise Wogglebug (CV output)
- Make Noise Brains/Pressure Points

4.4.2 Getting a Pulse

In this patch, I decided to use the Intellijel Shapeshifter as the primary clock source. This is a continuation of my patch "Intellijel Shapeshifter as Self-patched Granulator", in which I used feedback and built-in options to create rapid percussion and grains.

The Shapeshifter has a "Pulse" output, which goes high or low depending on a selected rule. For instance, the Pulse output can be high whenever the voltage of Osc 2 is negative. With simple waveforms (sine, saw, triangle, etc.), this output simply produces a secondary, unipolar square wave output that runs at the same frequency as one of the two primary oscillators. With more complex waveforms (noise or string wavetables from the Shapeshifter's many banks), the Pulse output

becomes more irregular. By modulating or morphing the primary oscillator, the output can provide complex, unpredictable rhythm patterns or chaos.

For this specific patch, I am deriving the Pulse output from Osc 2's polarity. I've introduced complexity into the patch by having Osc 2 modulate its own shape and harmonic ratio (relative to Osc 1). Osc 2 is running at LFO rates, and set to a fairly complex wavetable set. Without touching any controls, this system ends up settling into relative periodicity, providing a fairly stable clock pattern. However, a minor adjustment to Osc 2's Shape parameter can have a drastic, sudden effect on the entire pattern.

4.4.2.1 Similar Meta-Module: LFO + Comparator

You can create a similar system by combining an LFO with a comparator. A comparator is a device that provides a positive gate or trigger whenever an input voltage exceeds a specified threshold.

You can use one to derive steady clocks from simple waveforms. If your comparator provides a gate output, try deriving a unipolar square wave from a triangle wave. The pulse width and phase of the square wave will be determined by the comparator's threshold.

If you have a wavetable LFO, this is an excellent recipe for creating a pattern generator. Changing the waveform of the LFO will provide a different pattern, while changing the comparator's threshold can act as a density control.

4.4.3 Designing an Unpredictable Sequence, Part 1

The primary pitch sequencer is the Make Noise Brains + Pressure Points combo. The Brains receives the Shapeshifter's Pulse out as a clock, and the Pressure Points

provide the selection of eight possible voltages. Because of how the Shapeshifter's sequence has been programmed there are slow, audible pulses mixed with sudden, frantic, yet extremely brief groups of pulses.

The tight, brief groupings have a very interesting (and musically useful) side effect. They act almost as a randomizer for the sequence by forcing the Brains + Pressure Points to jump forward multiple steps without producing audible intermediary pitch changes.

4.4.3.1 Similar Meta-Module: Noise + S&H + Voltage Controlled Clock

These three modules work together to create unpredictable impulses via a feedback network. Essentially, the clock generator controls the Sample and Hold module. The Sample and Hold module samples the noise generator's voltage, and the sampled voltage is used to modify the speed of the clock. By using a "high-frequency" noise source (like white noise), the clock's output will be very random and difficult to predict. By using a "low-frequency" noise source (or even a stable, periodic waveform), the clock's output will start to conform to shapes and groupings.

4.4.4 Designing an Unpredictable Sequence, Part 2

The secondary pitch and gate sequencer in this patch is the Make Noise Wogglebug, a module that specializes in producing a range of random outputs. It has three separate sections: random audio, random CV, and random clocks. In this patch, it is receiving the Pulse output from the Shapeshifter, and using that as its primary clock source. With every clock that it receives, it is producing a new, random stepped voltage, along with a related slewed voltage. The stepped voltage is controlling the Elements' "Space" parameter, effectively providing a different amount of reverb

for each triggered note. The smooth/slewed voltage is controlling the "Position" parameter, adding a subtle amount of timbral modulation to the composition at all times.

The Wogglebug also has a burst output, which is a fairly unpredictable burst generator that produces groups of triggers.

4.4.4.1 Similar Meta-Module #1: Noise, S+H, and Slew

This is a continuation of the Noise + S&H Meta-Module. By adding a slew generator to the stepped random output, a smooth, continuous random modulation can be created. With the slew generator being separated from the noise source, you have more control over the "shape" of the random modulation. At quick slew settings, you can create rapid slides to values that are held for a period of time. At slower slew settings, the generated signal will never reach a value where it "holds". At extremely slow slew settings, the signal will fluctuate gently around its initial voltage.

4.4.4.2 Similar Metamodule #2: Random Source + Comparator

This is a similar Meta-Module to the LFO + Comparator above. By swapping out the periodic waveform of the LFO with a random source (i.e. sampled-and-held noise), a random trigger source can be created. A more intriguing Meta-module can be created by using a stable clock source to sample-and-hold the noise source going into the comparator. By doing this, the output of the comparator is a random gate that only changes state on new triggers. This can be useful for taking a stable, static clock source and using probability to remove triggers from it..

4.4.5 Combining the Voices

The primary voice of this patch is the Mutable Instruments Elements, which acts as both an instrument and an effect. As an instrument, it is a modal synthesizer that is being triggered by the primary patch pulse (the Pulse output from Shapeshifter). As an effect, it is receiving the output of the Make Noise RxMx, which mixes together the Make Noise Mysteron, the Intellijel Shapeshifter, the Music Thing Modular Radio Music (set to play birdsong samples) and the Elements itself.

The output of the Elements is plugged into the Make Noise FxdF, which is a fixed filterbank that breaks an input into six bandpass filters and sends it to the RxMx. Three of the channels of the RxMx are used in this patch to feed these bands back into the Elements, creating a frequency-sensitive feedback path.

The RxMx has a primary "Level" control, which targets the volume of all six channels, along with a "Strike" input, which receives a trigger and applies a vactrol envelope to the Level parameter. This "Strike" input is being triggered by the Wogglebug's burst output. This creates an intermittent mix, where components of the other voices suddenly appear inside of the Elements' resonating body for brief periods of time. In a way, it sounds a lot like rapid tape edits, not entirely unlike John Cage's "Williams Mix". By running everything through Elements, it unifies all of the disparate sounds by applying the same reverb and resonating pitch to each component.

Finally, the RxMx has a "Radiate" parameter, which controls what channels are currently active (i.e. which channels are modified by the "Level" parameter). The Mysteron is an always-active element, ensuring that it appears very frequently throughout the piece. The Radio Music's birdsongs and the Shapeshifter's Osc 1 output are plugged into the outermost channels, make them appear least frequently

in the mix.

4.4.5.1 Similar Metamodule #1: Voltage-Controlled Mixing (VCAs + Mixer)

Most Eurorack mixers lack voltage control, meaning that the user has to interact directly with the mixer to change levels. This can be changed by using a VCA per-channel before the mixer. The mixer's control thus acts as a maximum gain setting. In addition to his, the RxMx's "Strike" control acts like a VCA that affects all channels simultaneously. To achieve this same affect, put a VCA after the mixer instead of before it. With that setup, the complete mix can be affected simultaneously instead of on a per-channel basis.

4.4.5.2 Similar Metamodule #2: Voltage-Controlled Feedback (Effect + VCA + Mixer)

You can add feedback to any effect module by using a mixer (of at least two channels) after the effect. Plug the effect's input into one channel of the mixer, and the effect's output into the other channel. Monitor the effect's output as usual. To add feedback, turn up the output channel. It is important to keep this level low initially to avoid massive feedback swells. To animate this feedback, insert a VCA between the effect's output and the mixer's input. Now, the amount of feedback can be controlled using CV.

4.4.5.3 Similar Metamodule #3: Multi-Band Feedback (Effect + Filters + Mixer)

This is a more complicated Meta-Module that leads to more intricate feedback sculpting. You will need one filter for every band that you want to use. Use the same mixer and effect setup as above. Instead of plugging the effect's output directly into the mixer, add a filter between it. High-pass, low-pass, and band-pass filters all work here, depending on the frequency range that you are targeting. To add another band, add another filter and plug the filter's output into another channel on the mixer. You can now use the mixer as a feedback equalizer. More even more timbral options, you can modulate the filters' cutoffs.

Chapter 5

Unfiltered Audio: Polymorphism in Plug-ins

Unfiltered Audio is a company that I founded with fellow MAT students Joshua Dickinson and Ryan McGee in 2012. Our software makes strong use of modular design strategies. In this section, I will outline how the ideas in this dissertation have influenced our products.

5.0.1 Yoko: Adding a Mixer to a Band-Splitter

Yoko is a "band-splitter," or a crossover filter used for splitting a signal into three frequency bands. Joshua Dickinson and I designed and implemented Yoko as a Rack Extension for Reason. The Rack Extension format is particularly good for modular designs, as the user can switch between a front panel control set and a rear panel terminal set. Rack Extensions have standard in/outputs that are connected automatically when one is added, but the user can switch to the rear view and repatch as desired.

Yoko splits a signal into three frequency bands using Linkwitz-Riley filters. Each band has a Gain control, along with a stereo "Send" output. The most notable design element on Yoko is that we added three stereo sets of "Return" inputs on the back, along with a stereo "Sum" output. The Sum output adds together all three Send outputs. If a signal is preset on the Return inputs, it replaces the corresponding Send output on the Sum mix. This simple feature makes Yoko more convenient to use and introduces polymorphism.

Without using the individual band outputs, Yoko can be used as a simple 2- or 3-band EQ. A user can plug in a signal, manipulate the Gain knobs and cutoff ranges, and get an equalized signal on the Sum output. When used as a band-splitter, this design removes the need to add a mixer at the end of the signal path to sum the individually manipulated bands.

This design has been praised by many users, and Yoko currently has a 5-star average with 285 reviews on the Propellerhead shop.

5.0.2 G8 Gate: Manipulating the Noise Gate Envelope

G8 Gate was our first VST/AU plug-in (and later AAX and VST3). It is a noise gate, or an effect that tracks a signals amplitude and reduces it when it drops below a specific threshold. This effect is typically used to remove the noise floor from a recording. With G8, we analyzed every last section of a noise gate's signal path and created polymorphism wherever possible. Because of this, we often describe G8 as a comprehensive "amplitude workstation".

5.0.2.1 Reject Outputs and "Amplitude Splitting"

G8 features two sets of stereo outputs. The primary outputs are the expected gate outputs, while the other outputs are known as "Reject Outputs". These outputs contain the audio that does not make it through the gate. When summed together, the main outputs and the Reject Outputs provide the original signal with no change to amplitude or phase. When "Flip" mode is active, the outputs are swapped, allowing a user to explore the Reject signal on hosts that do not support multi-out VST2 plug-ins.

This feature allows users to create amplitude-sensitive effect chains, where louder signals are processed differently than softer signals. This feature was inspired by Tony Visconti's production techniques on David Bowie's "Heroes," in which three microphones were placed at various distances from the singer. Each microphone was processed differently, allowing for a wide dynamic range and the effect of an expanding room during louder sections [69]. We like to refer to this technique as "Amplitude Splitting".

A great use for this is to use G8 as an amplitude-sensitive auto-panner. To do this, a user can take the main outputs and pan them hard left, while the Reject Outputs would be panned hard right. In this scenario, a sound would begin on the right channel and pan to the left after it becomes loud enough before panning back to the right channel. The rate of panning would be controlled by the Attack and Release settings, while changing the "Reduction" parameter would affect the severity of the pan. By using G8's alternative behaviors (see section III.E), the auto-panner could be set to cycle at regular intervals.

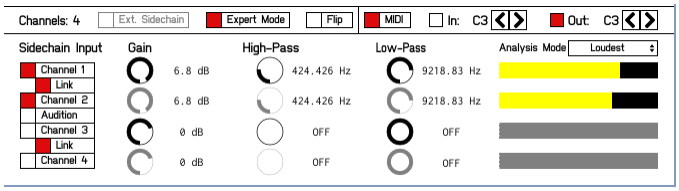

Figure 5.1: G8's Expert Mode Panel

5.0.2.2 Expert Mode

G8 features an "Expert Mode" panel, which contains controls pertinent to manipulating the analysis input. Without Expert Mode enabled, G8 features a toggle switch to choose between whether the gated sound runs through the analysis stage, or whether an external sidechain is analyzed. This is a fairly common control found on most other noise gates.

We wanted to give the user a lot more control, along with the possibility for opening up more creative mixing strategies. With Expert Mode, we've created something that we call an "Analysis Matrix". Here, the user has complete control over the toggle state, gain level and filtering of all four input channels. The channels can be "linked" together in stereo pairs, meaning that each pair can share gain and filter controls instead of requiring separate tweaking. The user can also choose whether the loudest sample from all four channels is analyzed, or whether the analysis sample comes from an average of the four. Each channel has a meter showing the channel's amplitude. Each meter also has an indicator that shows the gate's Threshold setting.

With Expert Mode enabled, the user also has access to "Audition Mode". With Audition Mode enabled, the user can listen to the signal that is being analyzed. This allows the user to hear exactly how the signal is being filtered.

As an example of the utility of Expert Mode, imagine that a user wants to gate

a stereo signal with huge variability between the individual channels. Let's also say that the user only wishes to use the left channel's loudness for gating analysis. If they were to use a gate that only toggles between the main input and the external sidechain, they would need to bus their stereo track to another track, use a utility to change that track to dual mono, and then route that track into the gate's external input. With G8, the user simply needs to enable Expert Mode and disable Channel 2. Another idea is to create gating polyrhythms by using two different rhythm tracks.

5.0.2.3 MIDI Functionality

For increased functionality, G8 contains MIDI input and output support. With MIDI input, G8 can listen for a specific Note On message. It will open the gate upon receiving a Note On, and close the gate with the Note Off. As an example, this can be used to rhythmically gate a synthesizer drone based off of some MIDI rhythm (this effect is frequently referred to as a "trance gate").

G8 can also have a MIDI output that sends a Note On when the gate is open and a Note Off when the gate is closed. This can be used to extract rhythms from audio tracks, or for drum replacement.

Another benefit of the MIDI functionality is for hosts that do not support VST2 plug-ins with more than two inputs and two outputs. A user can setup a "MIDI Sidechain," in which one instance of G8 can send MIDI output to the MIDI input of another instance. In this usage scenario, the G8 that sends MIDI output is similar in functionality to a sidechain input. The G8 instance that receives MIDI is on the track that is being gated. More creative uses will be explored in the next section.

5.0.2.4 Alternate Behavior Modes

G8's gate can use one of three different behavior modes: Regular Gating, One-Shot, or Cycle. For Regular Gating, the gate envelope behaves as expected. The envelope opens up when the signal's amplitude exceeds the threshold and closes when the signal drops below the hysteresis level.

One-Shot is a mode designed for transient shaping of more percussive tracks, or for applying percussive envelopes to other sounds. In this mode, once the signal's amplitude exceeds the threshold, the gate envelope immediately opens, runs through its Hold duration, and closes without waiting for the signal to drop below the hysteresis level. The envelope will not open again until the signal drops below the hysteresis level and exceeds the threshold again. With this behavior, an envelope with a specified attack and release time can be applied to individual drum hits.

Cycle mode is a creative mode inspired by our interaction with hardware modular synthesizers. In particular, we were inspired by Make Noise's Maths module, which in simplest terms is a dual AD envelope that can be set to cycle automatically [23]. In Cycle mode, when the incoming signal's amplitude exceeds the threshold, the envelope immediately fires and completes in the same style as One-Shot mode. However, after completion, the envelope will fire again provided that the signal is still above the threshold. In this mode, a "Delay" control becomes active, allowing the user to specify a required time between envelope triggers.

This mode allows the user to turn any sustained sound into a rhythmic effect. For example, a sustained synthesizer drone can be turned into a rhythmic eighth-note percussion track. If the user modulates the Delay control, complicated "bouncing ball" rhythmic effects can be achieved. Combined with its MIDI output feature, G8 can effectively be used as a standalone rhythm generator without requiring any audio

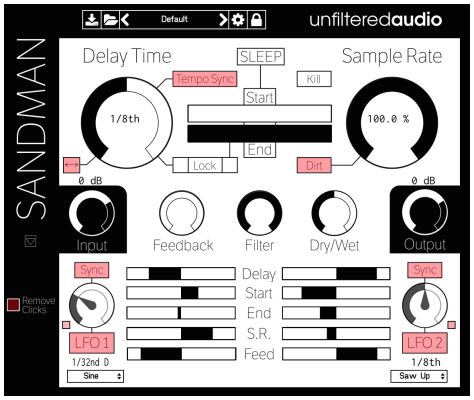

Figure 5.2: Sandman interface, with fixed modulation system visible.

input.

In addition to Cycle's rhythmic utility, its wide-ranging envelope times allow G8 to be used as a tremolo, an AM synthesizer, or a granulator. For tremolo, slower envelopes can be combined with G8's variable reduction for flexibility. For AM synthesis, G8's envelope time can go down to about 2 milliseconds, or 500 Hz. The Delay parameter can then space out these small envelopes, allowing for a single-stream granulator.

5.0.3 Creating an Expandable, Patchable Modulation System

At the start of 2016, we set out to create a expandable, patchable modulation system for all of our future plug-ins. When we wrote the first iteration of Sandman in 2014, a significant percentage of the code was wasted handling a fixed modulation system. It was difficult adding more parameters after release due to both programming complexity and interface requirements.

Our goal was to create a generic, expandable system that would work with all of our plug-ins and use a common codebase. It would also need to be implemented in a way to accommodate future upgrades (more modules, refined controls, etc.).

Figure 5.3: Fault interface, with expandable modulation system visible.

In April 2016, We released Fault as our first plug-in with the new modulation system. The design of the modulation system is visually simple and generic enough to work with all of our plug-ins. A MODULATION tab sits on the bottom of the plug-in interface. When the tab is active, a bottom section appears on the plug-in with a number of modules.

There is extensive visual feedback for the user. The modules are color coded in order from left to right. Whenever the user drags a cable from the module, the cable's color will match the color of its source module. The modulated parameter has a white bar showing its center value and a red modulation indicator to show its current value as set by the modulator(s). Multiple cables plugged into the same input will be summed (instead of requiring another module for mixing). Every output of every module has an attenuverter. Every time a user connects a cable from a module's output, a new output with a new attenuverter will appear. This makes every module expandable.

The initial list of modules is as follows:

- Sine LFO
- Saw/Tri LFO. This LFO has a SHAPE parameter that sets the LFO's wave-

form. At 12 o'clock, it's a triangle. Changing this value will morph the waveform toward sawtooth or ramp waveforms.

- Square LFO: This has a WIDTH control to set the waveform's pulse width.
- S+H Noise: This is a noise source that is sampled and held at a given interval. It has a SLEW control. At full CCW, no slew is applied, providing a random stepped signal. At full CW, the slew length will be equal to the rate of generation. This creates a smooth random signal that never holds a value.
- Input Follower: This module generates an envelope based on the amplitude of the plug-in's input signal. A SMOOTH control determines how rapidly the envelope will track the amplitude.
- Macro Control: This module allows a user to generate offsets using one knob. Since this module has an expandable number of outputs, turning this one knob can affect an arbitrary number of parameters in various amounts.

5.0.4 Dent: Making a Modular Distortion

Dent is a distortion and multi-mode bitcrusher released in August 2016. Our central design focus was making a distortion that would center around modulation. We ended up with a tool that takes design cues from Linked Polymorphism. It's not technically a linked design, as the separate elements can't be independently used, but the concept of chaining together simple sections to create more complex behavior is used. Dent does use Modal Polymorphism, which I will describe later in this section.

Our first design inspiration is the Doepfer A-136 Waveform Processor [70]. This module takes in an input and provides five amplitude controls: positive amplitude,

negative amplitude, total amplitude, positive clipping boundary, and negative clipping boundary. Only the clipping boundaries can be modulated.

I ported this module to Euro Reakt in late 2015 as the Waveform Processor 6.2.20. While working with that, I split the total amplitude control into Pre-Gain and Post-Gain controls, added a symmetrical DC bias control called "Split," and made everything a modulation target.

The next design inspiration came from the popularity of wavefolders in Eurorack. Wavefolders serve the same basic purpose as distortion: add harmonics to a simple signal. Most wavefolders have the same two controls. These controls are Fold (Gain) and Symmetry (DC Bias). At this point, a great way to combine distortion and wavefolding became clear. Every basic distortion algorithm has a gain control, and most have a bias control. The only difference between hard clipping and wavefolding is how to deal with audio that clears a given threshold. In hard clipping, the signal gets lopped off at the boundary. In wavefolding, the signal gets reflected.

The first draft of the Dent control set effectively took the control set from Waveform Processor, added in a Bias control, and replaced the asymmetrical clipping controls with a boundary-handling Mode selector. The Mode selector chose between hard clipping, soft clipping, and wavefolding. It also added in a toggled DC filter at the end of the signal path to handle offset introduced by the Bias and asymmetrical Gain controls.

After that, we looked at how most digital distortions are implemented. Most common distortion algorithms involve tanh and related sigmoid curves being applied to the signal, due to how well they model they behavior of Operational Transconductance Amplifiers [71]. A lot of the character of a distortion can be determined by the intensity of the sigmoid curve and the boundary-handling. Since Dent already

had variable boundary handling, this sigmoid shaping was the last basic element of distortion that hadn't been implemented.

For this, I had already implemented a Euro Reakt module called Waveshaper 6.2.22. This used one knob to skew a signal using either a hyperbolic or parabolic shaper. This behavior was ported to Dent using a hyperbolic shaper with a more limited control range (to avoid unwanted extreme behavior).

The output of the top-row shaping section is fed into a multi-mode "bitcrusher". It features a traditional bitcrushing mode, but it also includes two bitwise logic and three other digital degradation methods. This runs through a low-pass filter before being processed through Waveset analysis. The Waveset section selectively drops wavecycles, replacing them with either silence or the dry signal.

The modal polymorphism comes from the Bias control and the optional DC filter at the tail end of the signal path. By disabling the DC filter, the Bias control can be used to manually create a modulation signal at Dent's output. For even more complicated behavior, the Bias control can be manipulated through the expandable modulation section. This means that Dent can be used as a complex LFO or random source. With a DC-coupled output, Dent can directly interface to a hardware modular synthesizer.

5.0.5 Sandman Pro: Polymorphic Delay

Sandman Pro is a modally polymorphic delay with an expandable modulation system. Its predecessor, Sandman, only featured one mode of operation and had a fixed modulation system. The basic signal path of Sandman is a delay buffer that feeds a "frozen" buffer. Whenever the delay line is frozen, playback switches to the windowed frozen buffer, where the user then has the ability to manipulate the start and end

times of the buffer. In essence, it is an echo that can quickly turn into a looper or granulator. Sandman Pro builds upon this by adding six additional echo modes, an all-pass diffusor that can act upon the frozen buffer, and more.

The seven modes of operation are:

- Classic Tape: The original Sandman behavior, with added DSP techniques to simulate tape playback (wow and flutter, tape saturation, etc.). When changing delay times, pitch artifacts are heard.
- Modern Instant: A granular delay mode. In this mode, changing the delay time does not produce any audible artifacts. It achieves this by using two windowed delay taps.
- Pitch Shifter: A granular pitch shifter mode. Like Modern Instant, this mode is created through two windowed delay taps.
- Glitch Shifter: This mode was implemented by purposefully adding mistakes to the pitch shifter code.
- Reverse: A reverse echo effect. The reverse echo is achieved through two windowed delay taps.
- Multi-Tap: A complex echo effect with up to sixteen active delay taps. Changing the delay length or the spacing of the taps does not produce audible artifacts.
- No Echo: The plug-in's input is fed directly to the frozen buffer. This can be used to create stutter and glitch effects.

From a DSP standpoint, all seven modes are constructed through the use of a single 32-tap delay. Because of this, a user is able to quickly switch between modes without

experiencing clicks or pops. This design encourages creative uses of the various modes instead of forcing the user to think of each mode as a static entity.

Chapter 6

Euro Reakt

"Euro Reakt" is a free expansion for Native Instruments' Reaktor 6 [72]. Reaktor is a software platform for modular synthesis by Native Instruments [73]. With Reaktor 6, Native Instruments introduced the "Blocks" standard [74]. A Reaktor "Block" is very similar to a hardware module. It has an interface with user controls that manipulate a non-visible algorithm that generates or processes a signal. This algorithm is written in Reaktor Core [75], a JIT-compiled visual programming language that is similar to Graham Wakefield and Wesley Smith's Gen language [76] for Cycling 74's Max 6+ [61].

In the Block standard, all Blocks process signals within a restricted -/+ 1.0 floating point range [77]. This means that any output can be plugged into any input without worrying about whether the signal is scaled correctly. This is different from CSound and SuperCollider, where the various uGens and OpCodes expect differently scaled signals. As an example, filters and oscillators in SuperCollider expect signals to be scaled to Hz range (typically 20 to 22,000). Envelopes require signals to be scaled to values representing seconds. In one particularly bad example, SuperCollider's "Crackle" uGen expects signals in the range of 1 to "just above 2.0"

Figure 6.1: Reaktor Blocks: "MIDI & OSC Learn"

[78]. Values outside of this range lead to massive, unbounded signal explosions that can be frightening to users (and damaging to speakers). In my opinion, this sort of behavior discourages experimentation for novice users.

In the Reaktor 6 Blocks standard, there is no distinction between a timing signal, a control signal, or an audio signal. The programmer of the Block can choose whether aspects of a signal are processed at a standard control rate or sampling rate (In Euro Reakt, I have chosen to use sampling rate for every aspect).

The standard Block widgets (knobs, buttons, etc.) feature an easy-to-use MIDI and OSC learn system. To use it, a user right-clicks on any control to bring up a button that says "MIDI and OSC Learn". Clicking this will set that control into learn mode. Reaktor will then map that control to the next MIDI or OSC message. This is a very immediate and intuitive way of linking hardware control to the software environment.

Because of these features, I feel that Reaktor 6 is the software modular that most closely resembles the hardware experience.

The Euro Reakt project was started immediately after the release of Reaktor 6 in mid-September 2015. The first Block (Logic Mix) was posted only a day after Reaktor 6's public release. By November 2015, the project had grown to over 100 Blocks (95 of which were released publicly)[1]. The project was started as a response to what I felt

[1]For a complete list of Euro Reakt Blocks, please see Appendix A.

was an excellent workflow and a short-sighted built-in library. The standard Reaktor 6 Block library focuses much more on sound design and performance instruments than composition or generative strategies. The standard Block library is as follows: 4 oscillators, 4 filters, 4 effects (comb filter, reverb, delay, distortion/overdrive), 3 mixers, 2 envelopes, 1 LFO, 1 VCA, 1 8-step sequencer, 1 clock divider, 1 quantizer, 1 CV Processor (attenuverter, offset, and slew), 1 sample and hold, 1 master clock, 2 MIDI control inputs (trigger or note), 1 oscilloscope, and 2 amplitude sliders. In terms of sequencing-focused Blocks, only the sequencer, master clock, clock divider, quantizer, and sample and hold Blocks really qualify. In my opinion, this does not leave a lot of opportunity for generative compositions.

The Euro Reakt series has the following goals:

1. **Expanding the number of sound design strategies present in Reaktor 6**. The standard Block library focuses only on Subtractive and FM synthesis, with light additive synthesis provided by one oscillator. Euro Reakt adds in granular synthesis, Karplus-Strong synthesis, physical modeling synthesis (with help from Chet Singer's Ampere Modular series), wavetable synthesis (with help from Sandy Small's Reaktor Blocks), sampling, VOSIM, Multi-Phasor synthesis (a novel approach to complex synthesis described later in this paper), Wave Terrain synthesis, Vector synthesis, and more.

2. **Expanding the number of compositional strategies present in Reaktor 6**. As mentioned above, the built-in sequencing Blocks do not expand past the most common sequencing paradigms. Euro Reakt has boolean logic, flip-flops, probabilistic switches, sequential switches, a comparator, a burst generator, an analog shift register, and more Blocks that are focused on advanced generative ideas. This allows a composer to create emergent behavior using

only a handful of Blocks.

3. **Providing a platform for modular synthesis pedagogy**. Combinations of Blocks are called "Ensembles" in Reaktor. Euro Reakt includes a large variety of ensembles to help teach users how to harness the power of these Blocks by showing them in musical contexts. Each ensemble has a number of "Snapshots" (presets) that show variations of each patch with different control settings. Each ensemble has text-based documentation that appears when the ensemble is loaded. Furthermore, each Block has built-in documentation. There is a general description of each Block that is shown when the Block is loaded. Whenever the user clicks on a control, a detailed description of that control is then shown. Reaktor costs less than a typical Eurorack module and less than almost every Eurorack case on the market. With this toolkit, students and educators have access to a low-cost, high-performance system that works well as a platform for learning the fundamentals of modular synthesis.

4. **Prototyping and exploring advanced DSP ideas**. Throughout the creation of Euro Reakt, the visual modular language of the Core environment has encouraged me to explore many ideas. As a result, I have described novel effects and synthesis methods that I have discovered during this process (including Multi-Phasor Synthesi and Interleaved Modulation). This has also been a very useful platform when prototyping ideas for Unfiltered Audio plug-ins. Dent, for example, was created while exploring the Waveform Processor Block and the various modes of the Bitcrusher Block.

5. **Demonstrating Rhizomatic and Synchronously Polymorphic design, when possible**. The Euro Reakt library covers nearly every fundamental of

modular synthesis. Perhaps my largest goal was to emphasize the flexibility of modular synthesis through polymorphic design. Without a number of hardware constraints (price, panel sizes), I've been able to create designs that tap into many possible inputs and outputs within the DSP chain. I will describe this process under the "Design Notes" section of each Block.

In this chapter, I will list every Block in the library and analyze their designs using the taxonomies and ideas described in this dissertation. I will also look at other software modular systems and analyze them through the taxonomies detailed in this dissertation.

6.1 Other Software Modular Systems

In this section, I will use my three taxonomies to analyze many existing software modular synthesis environments. I will analyze them primarily on two qualities: how similar the software is to a hardware modular, and how useful the software is for education. I will not analyze text languages like CSound, SuperCollider, and ChucK or visual programming languages like Max or PureData. Instead, I will focus on software that is intended exclusively for modular synthesis.

6.1.1 Packages for Visual Programming Environments

In this section, I will look at modular synthesis packages for visual programming environments. Max and Pure Data offer a lot of functionality aside from modular synthesis. These packages provide a more focused set of tools with consistent interfaces and patching standards.

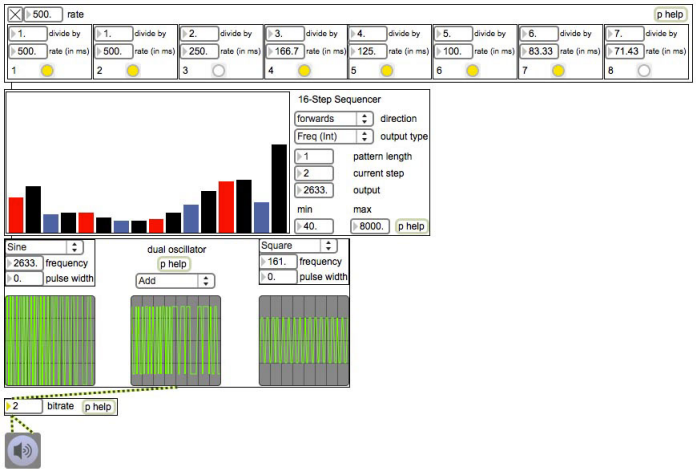

Figure 6.2: Euromax for Max 5+.

Previous Work: Euromax for Max 5

"Euromax" is a free package for Cycling 74's Max 5+ [79]. I originally started the project for MAT 201A, taught by Matthew Wright. The project was turned in as my final project for the class.

Euromax predates the similar BEAP package, which was released with Max 6. BEAP, in turn, predates the OSCiLLOT package, a similar environment that focuses more on Max4Live.

I made this is 2010, early into both my programming and modular synthesis studies. Surprisingly, many of the modules display simultaneous polymorphism. The Gate Sequencer has per-stage gate outputs (along with a sum output), the AD generator has a Loop switch and a End of Cycle trigger, the noise source has a simultaneous S+H output, etc. There are some very unusual designs, such as an "4x4 Envelope Matrix," a combo of 4 AD envelopes with a matrix modulation panel for routing the trigger outputs of each envelope into the trigger inputs of any of the envelopes.

There are some major drawbacks. The first is the visual design. Each input and output is only labeled through tooltip functionality, meaning that the user has to hover their cursor above each terminal to find out what it does. This greatly slows down the patching process.

The second drawback (which is common to many of the modular packages in this section) is difficult control. Euromax actually includes a number of quick MIDI Learn modules (the foundation of Simple MIDI, described in Taxonomy #2) for interfacing with hardware controllers. However, external controllers can only interact with each module through CV inputs. If a control is missing a CV input, it needs to be manipulated by the mouse.

The third drawback is that there are a number of DSP issues. Since this was made during my first year of studies, there are a number of areas where the modules display a lack of understanding. For example, the Trainlet Generator requires an external clock. The clock input is set to only receive "bangs," which are control rate timing messages in Max. This means that it's impossible to create a stable clock for the generator. It should have an internal audio-rate clock and the ability to clock itself for stable oscillations.

Finally, every module loads with poor default settings. Most modules load with 0.0 for all controls, meaning that each module doesn't produce useful results by default. This was due to my lack of understanding of the Max "pattr" system for storing settings with clippings.

Overall, this was a decent first effort for personal use but a poor choice for a teaching platform.

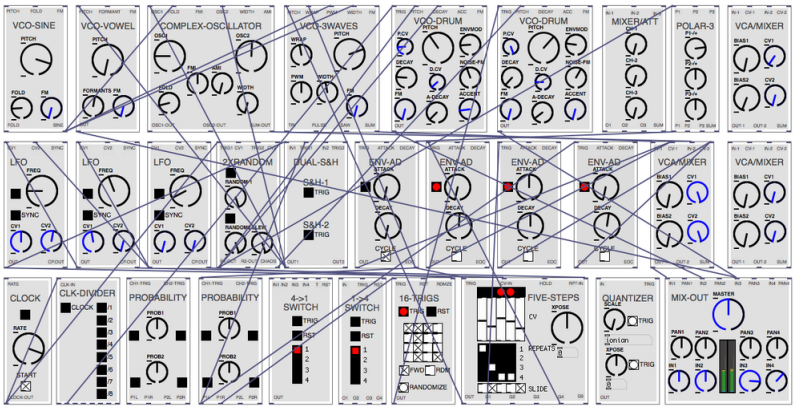

Figure 6.3: XODULAR for Pure Data Extended.

XODULAR

XODULAR is a package of modules for the Pure Data Extended environment [80]. It consists of twenty "foundational modular building blocks". The panel designs focus on simplicity, but many of the modules exhibit simultaneous polymorphism. As examples, the "Five-Steps" sequencer has gates for every stage, the Dual VCA offers a mix output, and the AD Envelope has an End of Cycle gate. It comes with a detailed manual with per-module documentation along with descriptions of every input and output.

There are a few drawbacks, the most significant of which is control. There is not an easy way to connect a hardware controller to the various knobs, meaning that the primary mode of interaction is the mouse. A less significant drawback is the requirement of Pure Data Extended (instead of Pure Data "vanilla"). This is a minor issue as PD Extended is no longer actively maintained, so future compatibility is questionable. Since Pure Data is free, a user can maintain PD and PD Extended installations side-by-side. Regardless, this is still a great system for students with its versatile designs and open-source nature.

Figure 6.4: Example BEAP modules.

BEAP

BEAP (Berklee Electro Acoustic Pedagogy) is a package of bPatcher modules originally developed by Matthew Davidson for Max 6 [81], but later included as standard in Max 7 [82].

BEAP's modules cover a lot of ground. The "Oscillators" category includes Karplus-Strong, Wavetable, Granular, FM, Additive, and more. One major issue with the designs is the lack of CV inputs on many modules, a few examples of which are visible in Figure 6.4. For instance, the Sync Delay, Decay, Feedback Delay, Pulse Designer, ADSR, and Wavefolder modules have no CV inputs. There are also a lot of missed opportunities for polymorphism in the more complex designs. For instance, the Pulse Designer features two square wave LFOs interacting through boolean logic. However, it only has one output. The individual LFOs do not have dedicated outputs.

The lack of CV inputs also affects external control. BEAP has a surprisingly deep controller module collection, including modules dedicated to interfacing with Monome controllers (ROLI has a separate package available in the Max Package Manager that adds BEAP modules for interfacing with their hardware controllers). Like Euromax, XODULAR, and OSCiLLOT, if a control does not have a CV input, it can only be manipulated via the mouse. One workaround is MIRA, a method of

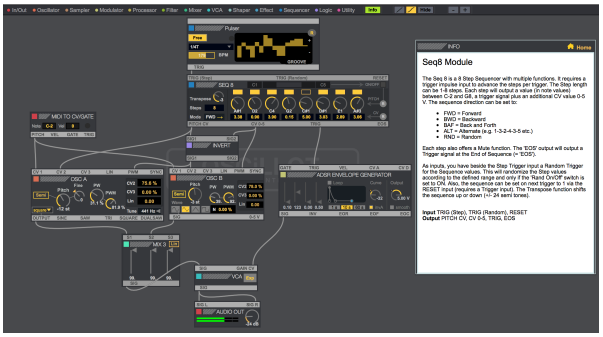

Figure 6.5: Max for Cats OSCiLLOT.

mirroring a Max interface onto an iPad or web browser. With this workaround, a user can at least use multi-touch to interact with multiple controls simultaneously.

BEAP has thorough documentation. Right-clicking on a module brings up the option to open an example help patcher with an interactive patch. There are many additional official YouTube videos and blog posts from Cycling '74. For classrooms that already have Max, BEAP makes for a great, educational synthesizer. The lack of deep control holds it back from being useful in performance situations.

OSCiLLOT

OSCiLLOT is a Max4Live modular synthesizer package developed by Max for Cats [83]. It differs from BEAP in two major ways:

- OSCiLLOT only runs in the Max4Live environment. It does not run in standalone Max.
- OSCiLLOT does not have the ability to interface with the rest of Max. It exists only in its own specialized environment.

OSCiLLOT is a polished environment. There are over 100 modules. Every module is fully documented (via a floating help system), and there are numerous example

patches. There is a notable amount of polymorphism present. As an example, the ADSR Envelope generator has outputs for Envelope, Inverted Envelope, End of Rise, End of Fall, and End of Cycle. However, the amount of polymorphism is not consistent from module to module. As another example, the AD Envelope only has a single output for the envelope with no additional triggers or gates.

There are two primary drawbacks to OSCiLLOT. The first is the upfront cost. To use OSCiLLOT, a user needs to own Ableton Live and Max4Live plus the additional purchase cost of OSCiLLOT. For students, this can be a large barrier to entry.

More importantly, OSCiLLOT is not suited to live control. While there are modules for interacting with controllers (including MIDI CC control and a module to directly interface with Ableton through 8 control macros), there isn't a way to directly map a controller to any knob. Modules can only be automated through their CV inputs. As an example, an OSCiLLOT step sequencer does not have CV control over each stage. This means that a user cannot map the sequencer's stages to a MIDI controller and must instead use their mouse to change the value of a stage.

Ampere Modular

Ampere Modular is a collection of "macros" for Reaktor 5+ [84]. Ampere is not designed for rapid patch creation. The Ampere modules are distributed as monolithic Reaktor "ensembles". A user needs to open these ensembles, copy the modules that they wish to use, and paste them into a new Reaktor ensemble.

The module design is monosemous with a small handful of exceptions. There's a rhizomatic 8-tap delay with per-tap outputs along with a stereo Mix output with per-tap level and pan. There's also a 4-pole filter with dedicated outputs for various filter types and one output with a Mode control (thus demonstrating simultaneous

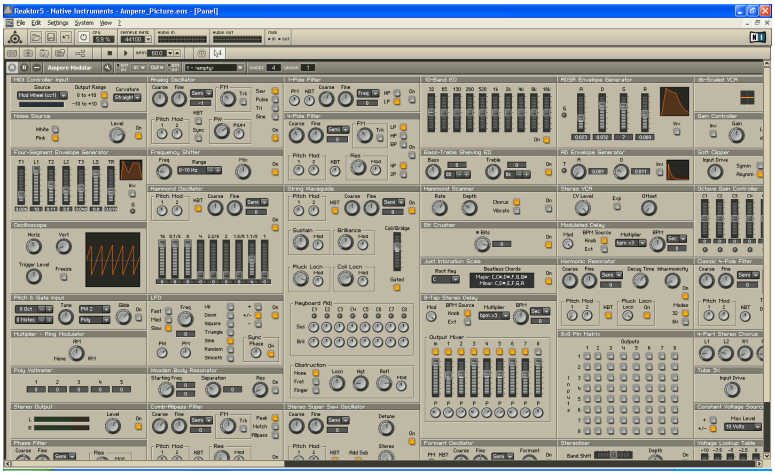

Figure 6.6: Ampere Modular for Reaktor.

and modal polymorphism).

Since the Ampere modules are built using standard Reaktor controls, every control has MIDI & OSC Learn.

Documentation on the modules is thin. Most modules have a one sentence description. Each control has a basic description as well.

This is an excellent resource for DSP techniques, and a few of the modules here have been ported to Euro Reakt (namely Resonating Wood and Resonating Bar). It is not a great choice for modular synthesis pedagogy, as using these modules requires a lot of Reaktor patching knowledge.

Other Reaktor Blocks Packages

There are other collections of Reaktor 6 Blocks released after Euro Reakt, including The Infinite Phi Collection by Sandy Small [85], The Nouveau Collection by Matthew Friedrichs [86], Brett Blocks by Brett Lavallee [87], and The Synite Collection by Jonathan Tremblay [88]. Every one of these collections is fully compatible with Euro Reakt as they follow the Reaktor Blocks standards. These collections also share a

Figure 6.7: The Infinite Phi Collection by Sandy Small.

lot of development work. Euro Reakt's wavetable Blocks are based on Sandy Small's Microwave Oscillator, Matthew's samplers were originally branches of Euro Reakt's Stereo Sample Looper, Jonathan extended Sandy's Pendulum Block, Sandy used macros from Chet Singer's Ampere collection and some of my noise generators, and so on.

These collections are not competing with each other so much as turning Reaktor 6 into an extremely versatile environment and deep educational resource for DSP and composition. Every collection is open-source and free. The shared ideas and differing design strategies mirror the current Eurorack environment where each manufacturer has a unique style.

6.1.2 Dedicated Software Modular Synthesizers

In this section, I will analyze a set of software programs dedicated to modular synthesis. These programs do not offer alternative functionality.

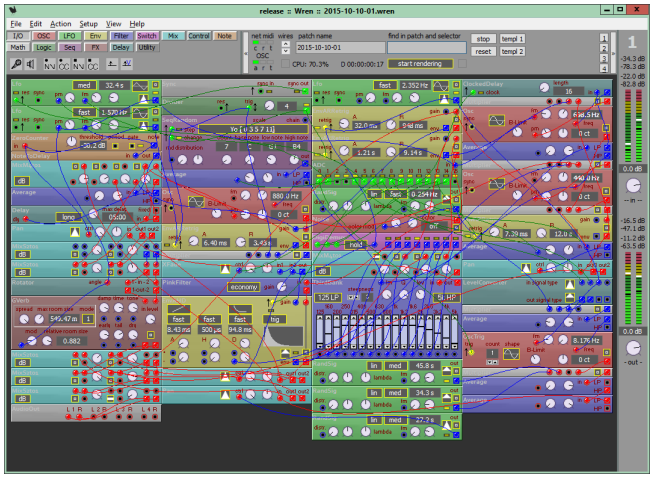

Figure 6.8: WREN Modular

WREN

WREN is a free, open-source modular environment for Windows only [89]. It is a large environment consisting of over 200 modules covering a wide range of synthesis methods, effects, and sequencing strategies. The interface is quite similar to the Nord Modular, but this is a software-only environment.

WREN's modules cover a wide range of design strategies. Most of the envelopes and filters are synchronously polymorphic. The envelopes have gate outputs and built-in VCAs, while the filters have multiple responses with individual outputs. Overall, though, most modules tend toward monosemous or rhizomatic design.

Every control can be mapped to MIDI messages. There isn't an easy auto-learn function. Instead, right-clicking a control brings up a menu for choosing which CC to map the control to. This requires that a user is familiar with the CC number of each control on their chosen controller.

WREN is thoroughly documented. Right-clicking on a module can bring up the associated documentation. In addition, the WREN website has every module documented on a single page. While WREN uses separate control and audio sam-

Figure 6.9: Sonigen Modular

pling rates, most modules are "RateSmart". This means that they will automatically change their sampling rate based on their inputs. The primary barrier to education is that the interfaces are small and many of the labels are abbreviated.

For experienced users on Windows, WREN is a flexible, free environment. With it's free, open-source nature, it is also a great choice for DSP and composition students.

Sonigen

Sonigen is a free modular environment for Windows only [90]. Like WREN, it features an interface that is very similar to the Nord Modular.

The module designs on Sonigen are extremely simple. One of the more unusual aspects is that the oscillators do not have pitch or FM inputs. All oscillators track Sonigen's MIDI note input and can only be set to offsets of this input. This means that none of the oscillators can be set to drone at an arbitrary frequency.

Each control can be assigned to a MIDI CC via an automation assignment mode. Sonigen comes with a PDF manual with documentation on each module, along with a large library of presets.

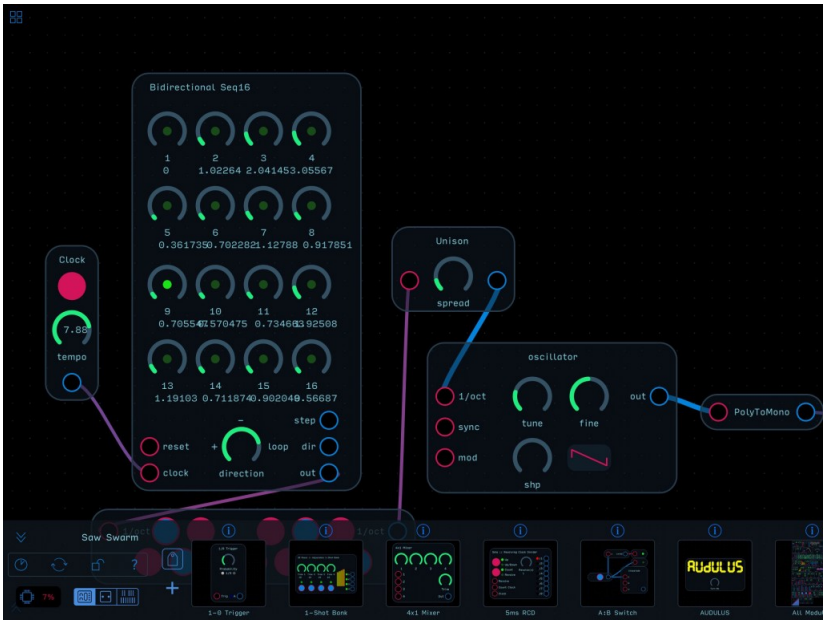

Figure 6.10: Audulus Modular. This screenshot shows both low-level "Nodes" like PolyToMono and higher level "Modules" like Bidirectional Seq16.

The strange pitch design choice makes this an interesting "transitional" synthesizer. It is a good first modular synthesizer for users who are familiar with plug-in instruments and want a similar workflow with more flexibility.

Audulus

Audulus is a modular synthesizer for iOS, OS X, Windows, and Linux [91]. It is a multi-scale modular in the vein of Reaktor. Like Reaktor's "Blocks" system, Audulus features a number of high-level "Modules" for easier patching. There is a lower level, referred to as "Nodes", featuring objects that serve more basic DSP functions.

The design of both the Modules and Nodes is almost exclusively monosemous. A user can use Nodes to design their own modules, though. Audulus can connect to external MIDI controllers very easily. Like Reaktor Blocks, every control has a quick MIDI learn system.

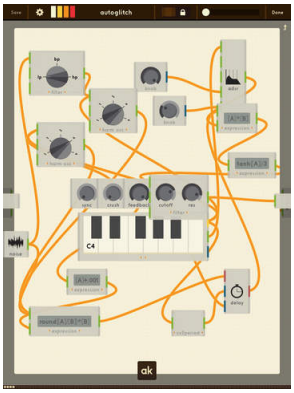

Figure 6.11: AnalogKit. This image shows the internals of a higher level module.

Every module is documented thoroughly with a description of inputs and outputs. There are many example patches as well. With the simple designs, good documentation, low cost, easy MIDI learn, and wide platform availability, this is a great choice for education.

AnalogKit

AnalogKit is a modular synthesizer for iPads only [92]. Like Audulus, it features a low-level set of objects that can be used to create higher level modules and interfaces. Unlike Audulus, there is also a lesser focus on the ability to create visuals.

AnalogKit has an easy MIDI learn system on every control. Instead of having a per-control menu like Audulus, it has a MIDI learn mode. When the mode is active, every available control is highlighted. The user touches the control they want to link and then sends a message from the MIDI controller that they wish to link. This method can be quicker for learning many controls at once.

Documentation is unfortunately poor. The application has a quick slideshow tutorial, but per-module documentation doesn't exist. As a result of this, it is not recommended as much as Audulus for education. For users who are already

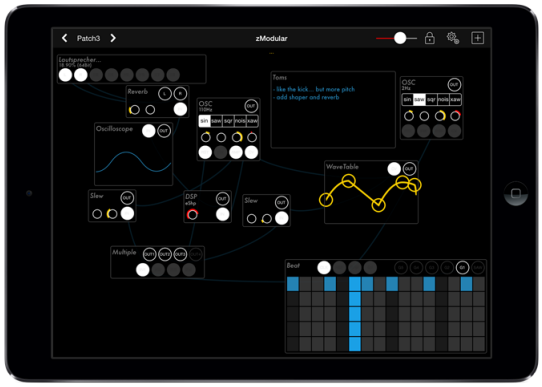

Figure 6.12: zMors Modular running on an iPad.

experienced with modular synthesis, though, it is a useful mobile platform.

zMors Modular

zMors Modular is a modular synthesizer for iPads only (with upcoming iPhone support) [93]. Unlike Audulus and AnalogKit, zMors Modular has only one level of modules. A user can create a "macro module" that hides a more complex patch behind a simplified interface.

The design of the modules is almost exclusively monosemous. Nearly every module has a single output.

Unlike Audulus and AnalogKit, zMors does not have quick MIDI learn. Instead, it has a number of MIDI modules used for routing incoming messages via channel, note, and CC.

A detailed manual is available is available from the website, featuring a patching walkthrough and per-module documentation. This is potentially a less flexible system than AnalogKit, but it is a good choice for education.

Figure 6.13: Arturia Modular V

6.1.3 Hardware Emulators

In this section, I will list software modulars that are direct emulations of modular hardware. These applications feature recreations of hardware interfaces and models of analog circuits.

Arturia Modular V and Moog Model 15

Arturia's Modular V is a software plug-in emulation of a Moog Modular for Windows and OS X [94]. Moog Model 15 is an emulation of the Moog Modular Model 15 for iPads only [95].

Both applications emulate Moog Modulars. The Arturia Modular represents a non-specific, large collection of modules while the Model 15 represents the same

Figure 6.14: Moog Model 15

module configuration as the hardware Model 15. The Modular V was endorsed by Bob Moog, while the Model 15 is an official Moog product. The Model 15 has a fixed layout, meaning that a user cannot add or reposition modules. The Modular V has a fixed layout except for the top row. On the top row, a user can choose their own configuration of filters, effects, and basic modulators.

Both applications map easily to external controllers. The Modular V has an easy MIDI Learn mode where every control is highlighted. Touching a control prepares it to map to the next incoming message. On the Model 15, any control can be manually mapped to a MIDI CC. The Model 15 comes with a number of multi-touch optimized keyboards that a user can switch between.

Documentation is great for both applications. The Model 15 has over 160 presets, while the Modular V has over 500. In each instance, every module is fully documented.

One of the only drawbacks is the inherent simplicity of the Moog modules. These are systems that are most comfortable for subtractive and FM synthesis with basic sequencing. That issue aside, these are great choices for learning modular synthesis.

Figure 6.15: Softube Modular. In this image, Doepfer and Intellijel emulations are visible.

Softube Modular

Softube Modular is a software plug-in emulation of a Eurorack system for Windows and OS X [96]. It is the first piece of software to feature licensed models of Eurorack hardware, including designs from Doepfer and Intellijel.

The base purchase of Softube Modular gives you access to a set of 33 modules. The Intellijel modules are available individually at additional cost. Most of the standard designs are monosemous, although there are simultaneously polymorphic modules: Logic (multiple logic types), Sequencer (voltage, gate, and trigger outputs), Noise (regular and S+H noise), and Signal Tool (multiple 2-input analog logic functions).

It is easy to control. In addition to mapping knobs to MIDI controllers, there are a number of MIDI-to-CV modules and "Performance Modules". The Performance Modules allow a user to map an arbitrary number of knobs, switches, or sliders to a single knob for macro control. One module connects directly to the ROLI Seaboard and offers over 30 CV outputs.

Documentation is great as well. The manual contains descriptions of every module

along with a patching tutorial. Over 200 presets are included.

There are a number of drawbacks. Like the Moog Modular emulations, the module selection is fairly basic. In the manual, it is described as a "more traditional subtractive system" with the Intellijel modules being "a step towards more west-coast thinking". Softube Modular also uses iLok for copy protection. While iLok is fine for a single studio computer, it can be difficult to manage in a shared studio or classroom environment. It is also a very demanding piece of software requiring a fast, up-to-date computer. Large, generative patches are unfortunately difficult to pull off with the CPU constraints.

6.1.4 Software-Hardware Hybrids

This section will focus on hardware devices that are programmed using modular software interfaces. These are typically hardware devices that are programmed via a computer and then used independently. These types of devices are useful for performing musicians who wish to remove a laptop from their stage equipment or use a piece of hardware with a dedicated interface for a patch.

Nord Modular

The Nord Modular was a digital hardware platform manufactured by Clavia from 1998 through 2009 [97]. The Nord Modular hardware consisted of common non-modular form factors, such as a keyboard or rack mount device. These devices were programmed using a proprietary software toolset, known as the Nord Modular Editor.

Despite being discontinued in 2009, the Nord Modular series still has an active community around it, centered primarily around the Electro-Music forum [98]. There

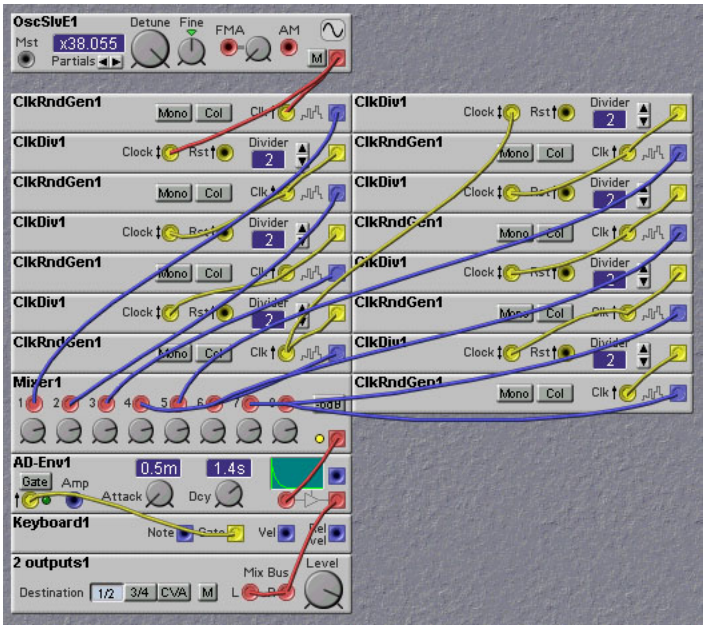

Figure 6.16: A Pink Noise generator patch from Jim Clark's Nord Modular book. Note the large number of low-level modules required for a single noise source.

are a significant number of pedagogically useful documents written using the Nord Modulars, including books by Professor James Clark [99], Rob Hordijk [100], and Roland Kuit [101]. The included manual is excellent as well.

There are a number of drawbacks to the Nord Modular system, the largest of which is its dependency on hardware. Many users use the Nord Modular G2 demo, which is a software-only demo with limited features. There are a number of hacked solutions that "unlock" these limitations, but the demos still contain an artificial limitation: the RAM and CPU of the hardware that they run on. Even on modern machines, a user cannot create patches that would exceed the hardware capabilities of a G2.

This is compounded by the Nord's focus on lower-level, monosemous modules. Many simple techniques require the use of multiple modules. For example, the delay modules do not feature dedicated feedback or mix controls. To create a typical

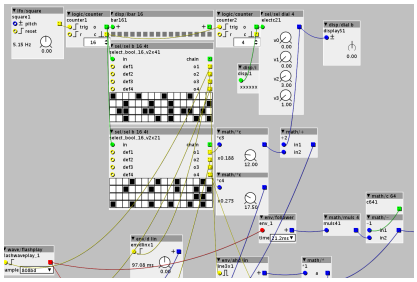

Figure 6.17: Axoloti software patcher.

echo patch, a user would need a mixer for feedback and a crossfader for dry/wet balance. This not only slows down the composer, but it also exacerbates the hardware limitations.

Axoloti

The Axoloti is an open-source, standalone hardware platform by Johannes Taelman [102]. It comes as a naked PCB with a number of audio, MIDI, and USB terminals along with a microSD card slot. Sketches for the Axoloti are programmed using a visual patching environment available on OS X, Windows, and Linux. Once a sketch is uploaded to the Axoloti, it can be used without a computer.

Axoloti has a very active community around it. New modules are written in C using a built-in programming interface inside of the environment. Every user-submitted module is downloaded when booting the software environment. This means that there are hundreds of available modules, many of which are variations of the same ideas but with different levels of complexity, fidelity, and polymorphism (One user, SirSickSik, has submitted over 400 modules alone). One huge drawback is that the documentation for each module can be poor or non-existent, and a significant percentage of the modules have cryptic names and interfaces. For instance, a 4-channel 16-step gate sequencer is called "sel/sel_b_16_4t". There are also variations of many

Figure 6.18: Monome Aleph with "Bees" patcher interface.

low-level modules. For instance, a simple two channel mixer has separate modules for adding control or audio signals (though the authors have mentioned eliminating this in a future update with modules that can detect what type of input signals they have).

The Axoloti has plenty of inputs for controllers, including a large number of GPIO pins and a dedicated USB port to act as a MIDI host for any MIDI-over-USB controller. One major issue is that controls can only be manipulated through CV inputs. This means that a MIDI controller cannot be mapped to an interface knob. Instead, a user needs to add a MIDI CC module, map the module to the knob's CC value, and then route the CC message to the module's CV input. if the desired control doesn't have a CV input, it cannot be manipulated from the controller and can only be set statically. Again, the authors are planning on fixing this in a future update.

Overall, this is an extremely promising environment. At the moment, though, it is not recommended for education until some of these issues are addressed. However, for experienced users this is an excellent choice for creating dedicated hardware devices.

Monome Aleph

The Monome Aleph is an open-source "soundcomputer" designed by Brian Crabtree and Ezra Buchla [103]. It is intended to be an all-in-one sound device that can inter-

Figure 6.19: Two Shbobo Shnths.

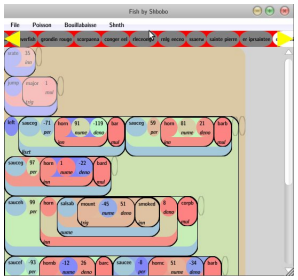

Figure 6.20: Fish patching environment for Shnth.

act with modulars, MIDI controllers, Monome's controllers, and other computers, as well as working on its own. The Aleph comes with a number of programs, but new programs can be written using a modular environment called "Bees".

Bees is a barebones, text-based interface with a list of sources on the left column. A user selects a source via an encoder and turns another encoder to select that source's destination or parameter value. A visual patching environment was planned but has not been released.

Shbobo Shnth

The Shnth is an open-source "handtop" by Shbobo, a digital offshoot of Peter Blasser's Ciat-Lonbarde company [104]. It is a small, battery-powered device designed around "squish data" interaction (buttons and pressure).

It can be programmed in two ways. The first is a Lisp-like text language called "Shlisp". A user can write text files using this language and then compile and upload

them to the Shnth via a command line executable. The second option is "Fish," an unusual visual patching environment. Instead of using patch cables, modules are nested inside of each other.

Every module (called an "opcode") has one output and one function.

Every parameter (including the device's sampling rate) can be linked to one of the device's controls. It cannot be controlled by another device, though.

The Shnth is thoroughly documented. There is a complete tutorial with per-module documentation available from the Shbobo website. There are also a number of official tutorial videos. One barrier is that the opcodes all have unusual names. "Wave" is a low-pass filter while "Salt" is a high-pass filter. There are three granular oscillators: "Fog," "Swamp" and "Haze".

The Shnth is a very unique modular synthesizer that cannot be compared to any other device. It should not be considered as an option for teaching modular synthesis. Still, it is an intriguing design that explores the idea of modular synthesis beyond the traditional module-and-cable concept.

Roland AIRA "Effectors"

The Roland "AIRA Effector" lineup consists of four effects: Demora (Delay), Scooper (Looper/Glitcher), Torcido (Distortion), and Bitrazer (Bitcrusher) [105]. These are very unusual "modular within a module" designs. They can act as standalone tabletop effects, Eurorack modules, or USB audio interfaces.

By default, each module is a monosemous stereo effect. Every module has an identical interface with four controls, two attenuators, two buttons, and a master VOLUME knob (Scooper has an additional RECORD button).

However, each module can interface with a free companion app called the "AIRA

Figure 6.21: Roland Scooper, shown next to the Roland Modular Customizer. Here, the base effect is combined with an animated filter.

Modular Customizer," available for iOS, Android, Windows, and OS X. This application is a complete modular synthesis environment featuring 32 "sub-modules" (31 modules are shared between each effect with 1 additional module representing the given piece of hardware). A user can create a full patch which can then be uploaded to the module either over USB or through a front-panel audio jack. Each module thus becomes a standalone modular synthesizer. Internally, each "sub-module" can be controlled by any of the hardware controls.

This is a very interesting choice for composers wishing to learn more about modular synthesis. While each effect is fairly expensive ($300), they are each standalone modular synthesizers that can later be integrated into a larger Eurorack system.

6.2 Effects

This category is primarily used to process audible signals. Some of these effects can also double as useful CV processors (including the Bitcrusher and Bitshifter Blocks).

Many of these Blocks share the following controls, which will not be detailed in

each Block's description unless notable:

Panel Controls:

- GAIN IN/OUT: Controls the level of the signal before and after processing. These can be unipolar or bipolar, indicated by the knob's graphics.
- x1/x2: Determines whether the Gain knobs can provide amplification.
- OFFSET: Adds a DC offset to the signal. This is usually a bipolar control that can add DC offset in either direction.
- DRY/WET: Controls the mix between the dry input and wet output.
- AC/DC: This enables an optional DC filter to remove bias from effects that can introduce it. I've included this switch especially for effects that double as CV processors.
- S. RATE: Controls the sampling rate of the entire Block. This is a common control on many effects that I write (including my plug-ins with Unfiltered Audio). Changing the sampling rate will affect the sound quality, increase maximum delay length, pitch shift signals trapped in feedback loops, and generally affect the signal in dramatic ways.

Inputs

- In: Primary signal input.

Outputs:

- Out: Primary signal output.

Figure 6.22: Bitcrusher Panel

6.2.1 Bitcrusher

6.2.1.1 Description

This is a multi-mode 16-bit bitcrusher. Its primary purpose is to take a signal and reduce its sampling rate (time resolution) and/or its bit rate (amplitude resolution).

The bit reduction section has six different modes:

- FLOAT - Smooth, between-integer bit-crushing. With this mode, there are smooth transitions between the "bits". This is not technically true bit reduction, as it uses floating point rounding errors instead of the direct manipulation of bits.

- INT - Integer bit crushing. This is true bit reduction. With this mode, an integer number of bits is chosen to represent the signal. Unlike FLOAT mode, this is not a smooth transition. Instead, transitions occur abruptly at 15 bits, 14 bits, 13 bits, etc.

- AND - Bitwise AND operation between the input signal and a 16-bit integer generated by the BITS knob. This is a harsh operation for audio. It works well on modulation signals, presenting jagged, fractal-like structures on smooth waveforms.

- FOLD - Bitwise AND operation between the input signal and a wavefolded version of the input signal. Reducing the bit knob increases the wavefolding amount. This mode is an original creation for Euro Reakt. It works better than AND mode on audio signals.

- XOR - Bitwise XOR operation. Sounds very similar to AND, but has an interesting mirroring effect around bits 8 and 9. At the lowest settings, you simply get a phase-inverted version of the input signal.

- TOY - Taken from Mutable Instrument's TOY mode on the Braids oscillator [106]. This chains together various bitwise operations to simulate "circuit bending" a CMOS system.

6.2.1.2 Controls and Terminals

Panel Controls:

- TYPE: Selects the mode of bitcrushing.

- BITS: For FLOAT and INT, this controls the number of bits used to represent the signal. For other modes, it generates a 16-bit integer to compare the input against.

Figure 6.23: Bitshifter Panel

6.2.1.3 Design Notes

The idea for this Block was to create a Modal bitcrusher that would cover a range of digital bitwise destruction algorithms. It was inspired by the Doepfer A-189-1 [107]. The A-189-1 included a number of bitshifting operations, but I chose to move those into a dedicated Bitshifting Block. The primary reason I did this is that the bitshifting algorithms are aesthetically much different from standard crushing modes. Minor parameter changes are also very severe and can lead to noise. The second reason is that I find that bitshifting works well in series with bitcrushing. With the separate Blocks, the user can choose whether shifting occurs before or after crushing.

6.2.2 Bitshifter

6.2.2.1 Description

This is an aggressive bit-manipulation effect. It converts and incoming signal to a 32-bit integer representation. From there, it performs a bitshift operation, shifting the representative bits left or right in their binary representation.

6.2.2.2 Controls and Terminals

Panel Controls:

- SHIFT: Bipolar control. 12 o'clock represents no bitshift. Turn this to the right to shift the bits right, or turn it left for a left shift.

6.2.2.3 Design Notes

See Bitcrusher notes.

6.2.3 Chebyshev + Chebyshev Scanner

6.2.3.1 Description

These two Blocks use Chebyshev polynomials to either distort a signal or generate higher-order harmonics from sine waves.

There are two Blocks: Chebyshev and Chebyshev Scanner. Both Blocks allow the selection of Chebyshev polynomial orders 1 through 8. The Chebyshev Block uses a switch to accomplish this. Only one polynomial order is active at a time. The Chebyshev Scanner smoothly interpolates between the eight orders.

Figure 6.24: Chebyshev Scanner Panel. The standard Chebyshev Block has an identical interface.

6.2.3.2 Controls and Terminals

Panel Controls:

- ORDER: Determines the order of the active Chebyshev polynomial, from first to eighth. On the Scanner, this is a smooth control.

Outputs:

- OUT1-8: On the Scanner Block, all eight orders of the polynomial are available simultaneously.

6.2.3.3 Design Notes

The standard Chebyshev Block is a straightforward, monosemous design. There is a modulatable switch to choose the active algorithm. The Chebyshev Scanner Block is an original design and a much more interesting way of dealing with the polynomials. With the Chebyshev Scanner, I took the wavescanning algorithm from the Scanner

Figure 6.25: Circle Delay Panel

Block and use it to smoothly crossfade between all eight available polynomials. This allows a user to provide gentle, smooth modulation for morphing distortion effects or harmonic crossfades. It also means that all eight polynomials needs to be calculated at once. Because of this, all eight polynomials have individual outputs. This is a more rhizomatic design, as the various harmonics or distortion flavors can run through different effects in unusual combinations. The two reasons to keep the standard Block are precision (easy to select exactly which polynomial is active) and CPU usage.

6.2.4 Circle Delay

6.2.4.1 Description

This Block provides two related delay lines.

6.2.4.2 Controls and Terminals

Panel Controls:

- TIME: Sets the delay length for Delay 1.
- SPREAD: Sets the delay length for Delay 2. This is an offset from Delay 1.
- FB1/2: Controls the amount of feedback each delay line has upon itself.
- 1->2/2->1: Controls the amount of feedback each delay line sends to the other.
- XFADE: Controls the balance of the two delay lines present on the Mix output.

Outputs:

- Mix: This output contains a mix of the dry signal and both delay lines. It is affected by the OUT, XFADE, and DRY/WET controls.
- Delay 1/2: These outputs contain the raw output of each delay line, unaffected by the OUT and DRY/WET controls.

6.2.4.3 Design Notes

The design was originally inspired by the Delay No More Eurorack module by Non-linear Circuits [108]. However, that module is implemented using PT2399 delay chips and focuses on exploiting those for noisy, glitchy purposes (The chips are known for producing a large amount of noise past 150 ms delay times). In the Delay No More, the delays are in series, and only the output of the second delay line is available. This Block is much cleaner and is intended for the creation of complex echoes instead of noise.

Figure 6.26: Clipper Panel

The Delay No More is a monosemous design, as it only has an input, a CV for the length of the second delay, and an output. This one is more rhizomatic, as both delays can be accessed independently. There is also a modulated crossfader output for more complex effects. One patch idea is to use this as a mono-to-stereo widener by treating the independent delay outputs as stereo outputs.

6.2.5 Clipper

6.2.5.1 Description

This Block sets an amplitude boundary that a signal cannot cross. When the signal exceeds that boundary, it will either be hard clipped or sent through one of three saturators. It is important to note that this Block only provides symmetrical clipping. For asymmetrical clipping, the Waveform Processor Block can be used instead.

The four modes are:

- HARD - Hard clipping. Signals that exceed the threshold are replaced by the

threshold value. This is a very harsh, digital sound.

- PARA - Parabolic saturation.
- HYPER - Hyperbolic saturator.
- TANH - Hyperbolic Tangent saturator.

This can be used to add harmonics to a signal. Alternatively, it can be used as a very unusual coloring VCA by modulating the clip level. The front panel has a clipping indicator as a visual cue that the signal has exceeded the given threshold.

6.2.5.2 Controls and Terminals

Panel Controls:

- CLIP: Sets the amplitude threshold for the signal. This is a unipolar control that sets up a bipolar threshold.
- TYPE: Selects one of four clipping methods.

Outputs

- Hard, Para, Hyper, Tanh: Dedicated outputs for each clipping method.
- Clipped: A gate that is positive when the input signal exceeds the clipping boundary.

6.2.5.3 Design Notes

This is a simultaneously polymorphic design. It can be used as a signal clipper and a comparator (with a true gate whenever the signal is clipped). The gate could be used to quickly duck the signal or route it to a different place when it clips.

Figure 6.27: Comb Filter Panel

6.2.6 Comb Filter

6.2.6.1 Description

This is a simple comb filter. It differs from the standard library's Modern Comb in that it uses the Bento Box Osc's frequency control. This makes it a lot easier to tune the Comb to useful values. In addition to this, it adds the ability to choose between two FM modes, a dry/wet control, in/out gain controls, and an inverted mode.

6.2.6.2 Controls and Terminals

Panel Controls:

- PITCH: Big blue knob. This determines the "pitch" of the comb delay line.
- KEYBOARD: When active, the PITCH control will be modified by signals at the "Pitch" input. The PITCH knob then functions as a semitone offset.
- FM: Sets the depth of frequency modulation over the comb filter. The FM

modulator is the signal present on the "FM" input terminal.

- FM TYPE: Selects an FM algorithm. These are EXPonential, LINear, and LINear Thru-Zero.
- REG/INV: REG is regular comb behavior. INV swaps the notch and peak positions.
- FEEDBK: Sets the intensity of regeneration in the delay line. This is a bipolar control, so negative feedback can be used for a different effect.

Outputs:

- Wet: Dedicated wet output, unaffected by OUT gain.

6.2.6.3 Design Notes

This is modally polymorphic. It has an extremely wide frequency range, so it can be used to create the distinctive comb effect or much longer echoes.

6.2.7 Dattorro Verb

6.2.7.1 Description

This is a reverb based on a paper by Jon Dattorro [109]. It is apparently a similar algorithm to the one used by classic Lexicon reverbs. This is the same reverb found in the Clouds Eurorack module by Mutable Instruments [110].

6.2.7.2 Controls and Terminals

Panel Controls:

- SIZE: Controls the apparent size of the reverb.

Figure 6.28: Dattorro Verb Panel

- PRE: Determines the length of pre-delay. This is an amount of time between when the signal is received on the input and when it is sent to the reverb. This can make the apparent space larger, or it can be used to create unusual, diffuse echoes.
- DAMP: Increases the amount of lowpass filtering on the delays' feedback. This will dampen the high frequency reflections.
- BRIGHT: Changes the amount of low-pass filtering on the input.
- TUNE: Change the frequency of the first-stage allpass filters. 12 o'clock is the standard Dattorro tuning.

There is a mono audio input and a stereo output.

6.2.7.3 Design Notes

The Dattorro, Schroeder, and Freeverb reverbs are monosemous designs with basic inputs and outputs. The nature of their DSP algorithms makes it difficult to consider

Figure 6.29: Entropy Filter Panel

alternative input and output strategies. For instance, you could consider tapping the individual combs/delays/all-pass filters, but taken on their own their outputs would not provide much variety as the delay lengths are so short. On the Erbe-Verb [111], Make Noise added a envelope follower output to provide simultaneous polymorphism. On a software design, though, this would be a superfluous addition. The Erbe-Verb also features a Tempo Sync input and a Reverse gate input, both of which are incompatible with the structure of these reverbs.

6.2.8 Entropy Filter

6.2.8.1 Description

8-Bit Probabilistic Destruction with Sample Rate reduction and a variable SR filter.

This takes the A/D and D/A conversion Blocks and stuffs eight probability filters between them. Each probability feature determines the chance that each bit will be turned to "0" when its "1". This effectively leads to amplitude-sensitive destruction.

The result of this runs through a low-pass filter to reduce harshness.

6.2.8.2 Controls and Terminals

Panel Controls:

- OFFSET - Adds a DC offset to the signal. In A/D conversion, this happens post-scaling. In D/A conversion, this happens pre-scaling.
- SCALE - Attenuates and/or inverts the signal. Turn on X2 to amplify the signal as well.
- BIT1-8: Determines the probability that each bit will be forced to "0" each sample. At full CW, this bit will always be 0, thus creating a bitmask.
- CUTOFF - Determines the cutoff of the low-pass filter.
- STBLE/VAR - Changes whether the filter receives a stable sampling rate clock, or is crushed.

Encoding/Decoding MODEs:

- UNI8 - 8-bit unsigned representation. Expects a unipolar input signal of 0-1, but don't let that stop you from using bipolar signals!
- BI OFF - Scales and offsets a +/- 1.0 signal to 0-1 before using the UNI8 encoder. Naturally, detail is lost.
- BI SIG - First 7 bits are used to represent your signal. The 8th bit carries the sign of the signal (positive or negative)

Three modes of RECTification.

- NONE - No rectification. Normal signal.
- HALF - Negative component of signal is silenced.
- FULL - Negative component of signal is flipped (Takes absolute value of signal).

Outputs:

- B1-8: Individual outputs for all eight bits, post-probability.

6.2.8.3 Design Notes

This is an original effect of my design. While designing it, I was experimenting with a Eurorack DSP program called "Bit Rot" [112]. It is a multi-mode bitwise destruction device built around the interaction of two bitmasked signals. I really liked the bitmasking sound and wanted to build upon that.

This design effectively takes the A/D and D/A conversion Blocks and places 8 Probability Blocks between them. By setting an individual bit's probability to 0%, you can mask it out entirely. Any non-extreme value produces a noisier bit. For less significant bits, this adds a pleasant noise floor. For more significant bits, this creates amplitude-sensitive noise generation.

Because all eight bits are available as outputs, this Block is simultaneously polymorphic. It can be used, for instance, as the Entropy Filter effect along with an above-zero comparator (depending on the encoding and decoding mode, Bit 8 can act as a comparator or square wave extractor).

Figure 6.30: Frequency Shifter Panel

6.2.9 Frequency Shifter

6.2.9.1 Description

6.2.9.2 Controls and Terminals

Panel Controls:

- PITCH: Sets the frequency of the internal complex oscillator.
- KEYBOARD: When active, the frequency of the internal complex oscillator will be controlled by the signal present at the "Pitch" input. The blue PITCH knob turns into a semitone offset control.
- FM: Sets the amount of frequency modulation over the complex oscillator. The modulator is the signal present on the FM terminal.
- FM MODE: Selects an FM algorithm. These are EXPonential, LINear, and LINear Thru-Zero.

- UPPER/LOWER: Sets the gain and polarity of the upper and lower sidebands.
- FEED: Controls the amount of feedback sent from the wet output back into the input. This is a bipolar control, so unusual phasing effects can be created with negative feedback.

Outputs:

- Out Mix: Both sidebands summed together. Affected by UPPER and LOWER controls.
- Out Up: Upper sideband. Affected by the UPPER control.
- Out Down: Lower sideband. Affected by the LOWER control.
- Sin/Cos Osc: Oscillator outputs for the internal quadrature oscillator. The frequency of these oscillators is equal to the chosen shifter value.
- Pitch: Outputs a signal based on the signal present at the Pitch input combined with the offset generated by the PITCH control.

6.2.9.3 Design Notes

This is a simultaneously polymorphic design. It can be used as an effect and/or a quadrature oscillator. The oscillator and frequency shifter outputs are both affected by the primary pitch/frequency control.

6.2.10 Low-Pass Gate

6.2.10.1 Description

A Low-Pass Gate is an effect that combines a low-pass filter with a voltage controlled amplifier (VCA). This effect is frequently used to make physical sounding percussion

Figure 6.31: Low-Pass Gate Panel

tones.

Most analog low-pass gates are based on opto-isolators, also known as "vactrols". A vactrol is an LED and a light sensor wrapped in a light-proof case. The vactrol effectively acts as an interpolator for modulation inputs, imparting a fast attack time and slow, rubbery decay. In typical usage, a low-pass gate will be modulated using a gate or trigger.

The vactrol's output controls the amplitude of the VCA and/or the cutoff of the low-pass filter (determined by setting the LPG's mode to VCA, LP, or BOTH).

6.2.10.2 Controls and Terminals

Panel Controls:

- OFFSET - Adds bias to the vactrol, leaving it partially open.
- DAMP - Controls the decay time of the vactrol.
- RES - The resonance of the low-pass filter. Only audible in LP or BOTH mode.

- MODE - Choose between VCA (amplifier only), LP (filter only), or BOTH.
- LIN/EXP - Controls the modulation curve for the VCA and filter cutoff.
- ENV - Controls the depth of modulation that the vactrol imparts on the VCA and filter.

Inputs:

- Vac - Vactrol Input. Typically, you would hit this with a gate or trigger, but you can place any modulation source in here. The source will be modified to have a rapid attack and slow, rubbery decay.
- Ping - Converts a modulation source to a trigger by use of a comparator.
- Direct - Modulation input. Skips the vactrol and modifies the VCA/Filter directly without interpolation.

Outputs:

- Env - Output's the vactrol's control signal.

6.2.10.3 Design Notes

This Block existed before Native Instruments released their own "West Coast LPG" Block in the Reaktor 1.1 update. Their LPG sounds excellent, but it's a monosemous design. My LPG Block adds a number of features, making it rhizomatic and simultaneously polymorphic. First, the wide variety of inputs means that there are more ways of interacting with the vactrol simulation. The user can even skip the vactrol and use it as a direct VCF and/or VCA. To create polymorphism, there is a separate "Env" output derived from the vactrol, allowing this to be used as an envelope follower or a vactrol-like modulation smoother.

Figure 6.32: Schroeder Reverbs

6.2.11 Schroeder Reverbs: JCREV, JCREV FF, SATREV, Freeverb

6.2.11.1 Description

These Blocks are implementations of various Schroeder Reverberators described by Julius O. Smith [113]. A Schroeder Reverberator is a primitive reverb algorithm the uses a combination of all-pass filtering to diffuse a signal and comb filters to create echoes. They are often described as sounding “metallic”.

6.2.11.2 Controls and Terminals

Panel Controls:

- SIZE: Controls the apparent size of the reverb by changing the lengths of the delay lines. This can result in delay lengths that are not co-prime, so resonance issues can appear.

Figure 6.33: Quad Delay Panel

- S. RATE: Controls the sampling rate of the entire Block. If audio is already present in the delay lines, it will be pitch shifted as a result.

6.2.11.3 Design Notes

See Dattorro Verb design notes.

6.2.12 Quad Delay

6.2.12.1 Description

This Block contains four completely separate delay units. They are referred to as "Taps" on the interface, but that is inaccurate. They are actually four separate delay buffers.

Each delay has its own dedicated output (not affect by gain controls). A quad bi-polar mixer at the top of the Block controls the level of each delay line at the Mix output. This mixed output is further controlled by Dry/Wet and Out Gain

parameters.

6.2.12.2 Controls and Terminals

Panel Controls:

- MIX 1-4: Controls the level and polarity of each delay line at the Mix output.
- TIME: Sets the master delay time. Delay line 1 will be equal to this length.
- SP. 2-4: (Spread). These controls set the length of delay lines 2-4 as offsets of delay line 1's length.
- FEEDBK: Sets the amount of feedback. This is a bipolar control, so negative feedback can be created for phasing effects.
- TAP/MIX: Controls the feedback mode. In TAP mode, each delay receives only its own feedback. This is a cleaner type of feedback, and sounds especially good with negative feedback amounts. "Mix" takes the entire wet output and sends it back to the input.

Outputs:

- Mix: All four delay lines summed together. Affected by MIX 1-4, DRY/WET, and OUT parameters.
- Main: Delay line 1. Unaffected by MIX 1, DRY/WET, and OUT parameters.
- Tap 2-4: Delay lines 2-4. Unaffected by MIX 2-4, DRY/WET, and OUT parameters.

Figure 6.34: Ring Modulator Panel

6.2.12.3 Design Notes

This delay was inspired by the design of the Sputnik Four-Tap Delay and Dual Crossfader [11]. This Block features four delay lines that are fed by a single audio input. Unlike the Sputnik design, the spacing of the non-primary delays can be any length between a single sample and the length of the main delay line. The Sputnik design is independently polymorphic, as the crossfaders are separate from the delays. I would consider this design to be rhizomatic, as the individual delay line outputs all serve the same behavior.

6.2.13 Ring Modulator

6.2.13.1 Description

A Ring Modulator is a classic timbral effect. In its digital variation, it multiplies two signals together directly. In its original analog incarnation, it uses a "ring" of four diodes for a similar effect.

This Block contains a basic digital implementation, along with an analog simulation taken from Mutable Instruments' Warps module, which re-implements a model published by Julian Parker ("A Simple Digital Model of the Diode-based Ring Modulator", 2011). The ANALOG knob crossfades smoothly between the two models.

For ease of use, this contains an internal sine oscillator with FM. A toggle switch allows you to choose whether the internal oscillator or IN 2 is used to modulate IN 1.

6.2.13.2 Controls and Terminals

Panel Controls:

- PITCH: Controls the frequency of the internal oscillator.
- KEYBOARD: When active, the frequency of the internal oscillator will be controlled by the signal present at the "Pitch" input. The blue PITCH knob turns into a semitone offset control.
- OSC/IN 2: Sets the modulator source. OSC uses the internal oscillator, while IN 2 uses the signal present at the "In 2" terminal.
- FM DEPTH: Sets the amount of frequency modulation over the internal oscillator. The modulator is the signal present on the "FM" terminal.
- FM MODE: Selects an FM algorithm. These are EXPonential, LINear, and LINear Thru-Zero.
- ANALOG: Crossfades between a digital multiplication algorithm (CCW) and an analog diode ring simulator (CW).

Inputs:

- In 1/2: In 1 is always the carrier. In 2 is an optional modulator if the internal oscillator is not desired.
- Pitch: Sets the pitch of the internal oscillator when Keyboard mode is active.
- FM: FM modulator input for the internal oscillator.

Outputs:

- Out: Modulated output. Affected by DRY/WET and ANALOG controls.
- Digital: Digital multiplication output. Unaffected by DRY/WET and ANALOG controls.
- Analog: Analog diode simulation output. Unaffected by DRY/WET and ANALOG controls.
- Int. Osc: Internal oscillator output. Unaffected by the OSC/IN 2 switch.
- Pitch:

6.2.13.3 Design Notes

This is a sophisticated ring modulator with a linked polymorphic design. In its default configuration, only one input is needed for the ring modulation effect to occur, as the input acts as a carrier while the internal oscillator acts as the modulator. The user can instead provide two inputs and use the oscillator independently of the ring modulator. The crossfading between "digital" and "analog" modes gives this more timbral variation than other ring modulators.

Figure 6.35: Saw Multiplier Panel

6.2.14 Saw Multiplier

6.2.14.1 Description

This Block is based off of the Doepfer A-137-2 Wave Multiplier [114], and a similar paper by Bernie Hutchins [115]. Effectively, it takes in a single waveform (except for square waves), and produces four "phase-shifted" copies via a simple comparator method. "Phase-shifted" is in quotes, as it only properly phase-shifts sawtooth waveforms. It is still useful for other signals, but the results are much more unpredictable.

6.2.14.2 Controls and Terminals

Panel Controls:

- MIX 1-4: Controls the level and polarity of each phase shifter.
- SP. 1-4: (Spread) Sets the amount of phase shift on each channel. At full CCW, no shift will be present.

Inputs:

- IN: In Gain. This is the most critical control to get the desired sound from this effect. Full CW provides the proper effect for the Bento Box Oscillator (a standard Reaktor Block by Native Instruments) and all oscillators with +/- 0.66 range output.

Outputs

- Out 1-4: Individual outputs for each phase-shifter saw.

6.2.14.3 Design Notes

This is a rhizomatic take on the monosemous A-137-2 design. The A-137-2 only has one output where all of the saws are summed together. Also, all saws have the same amplitude at the output. In this Block, each saw is output separately. At the Mix output, each saw's amplitude is determined by the associated MIX control. While this isn't a polymorphic Block, it provides a lot more functionality and flexibility than the Doepfer counterpart.

6.2.15 Spectral Compressor

6.2.15.1 Description

This is a port of the Spectral Compressor from Native Instruments' ezFFT bundle for Reaktor 5 [116]. This uses an FFT to break a signal into 256 frequency bins. Each bin is then treated by a separate compressor before an iFFT is performed.

6.2.15.2 Controls and Terminals

Panel Controls:

Figure 6.36: Spectral Compressor Panel

- THRESH: Sets the amplitude threshold at which the compression behavior will become active.
- RATIO: Maximum amount of compression that is applied to loud signals. The Ratio applies only to levels higher than the Threshold.
- ATTACK: Adjusts how fast the compression rises when the input level goes above the threshold.
- DECAY: Adjusts how fast the compression returns to zero when the input level drops below the threshold.
- KNEE: Range above the threshold in which the compression rises to its maximum setting (set by RATIO).
- TILT: Acts as a basic EQ by emphasizing compression on high or low frequency bins. At 12 o'clock, normal compression will occur.
- MASK: Sets how much spectral leakage occurs between loud bins.

Figure 6.37: Timbre Panel

- AUTO/OFF: In AUTO mode, a gain compensation algorithm will be used to offset the influence of the THRESH and RATIO controls.

NOTE: Some of these control descriptions come from the original ezFFT compressor.

6.2.15.3 Design Notes

Despite the complexity of the control set, this is a monosemous design with a simple stereo input and output configuration. I did not contribute any design flourishes of my own other than the interface. Elements of the DSP algorithm were rewritten to accommodate the Blocks format. This holds true for the other ezFFT port in Euro Reakt, the Vocoder.

6.2.16 Tape Delay

6.2.16.1 Description

This is a Block based off of the Reaktor Core Library's "Tape-ish Delay" Macro. In addition to breaking out all of the controls to a panel, this Block also features a variable sampling rate. This feature was inspired by my company Unfiltered Audio's delay plug-in, Sandman.

6.2.16.2 Controls and Terminals

Panel Controls:

- TIME - Delay time, from 10ms to 1 second.
- FEEDBK - Feedback amount. Bipolar.
- PRE/POST - Controls where feedback gain occurs. With "Post", at 0% Feedback, you will hear no wet signal.
- OFF/FLUTTER - Adds a slight warble to the delay line. Turn FLUTTER on for a more faithful tape delay effect.
- SAT - Saturation. At low levels, the tape will almost never saturate. At high levels, it saturates very quickly.
- LPF - Low Pass Filter. Acts on the feedback line.
- HPF - High Pass Filter. Acts on the feedback line.

Output:

- Wet: 100% Wet output, not affected by the D/WET control.

Figure 6.38: Timbre Panel

6.2.16.3 Design Notes

This is a mostly monosemous design with the rhizomatic embellishment of the Wet output. The Wet output allows a user to create alternative feedback paths and mixes independent of the main output and the D/WET control.

6.2.17 Timbre

6.2.17.1 Description

This Block extracts the Timbre circuit from Native Instruments' DWG Complex Oscillator Block.

That Timbre algorithm depends on the DWG's triangle out, square out, and current frequency. This Block accepts any waveform. To create a square waveform, it uses a comparator. To create a triangle, it slews that square. As such, it is not a 1:1 copy, and can have some fairly unpleasant results with very complex waveforms. Still, this has proven to be a very useful Block for west-coast sounds.

6.2.17.2 Controls and Terminals

Panel Controls:

- SHAPE: Applies a waveshaper to the incoming signal. For a triangle wave input, this will fade between sine, impulse, and triangle + impulse. Native Instruments chose these waveforms to match the Shape control on the Make Noise DPO [117].
- MOD SH.: Controls the amount that the signal at the "Mod" input will modulate the SHAPE parameter.
- SYMM: (Symmetry) Applies DC bias to the waveform before it is folded.
- MOD SY.: Controls the amount that the signal at the "Mod" input will modulate the SYMM parameter.
- FOLD: Applies gain to the input signal to increase the depth of wavefolding.
- MOD F.: Controls the amount that the signal at the "Mod" input will modulate the FOLD parameter.

Inputs:

- Mod: Modulation signal input. This signal is used by the orange MOD controls.

6.2.17.3 Design Notes

This Block breaks from the regular Euro Reakt design in that it features a dedicated Modulation input. In every other Block, there are two generic modulation inputs. For this Block, the modulator is so important that it features a direct input along with attenuverters on every destination. This design is consistent with how the modulating oscillator is treated on both the NI DWG and Make Noise DPO interfaces.

Figure 6.39: Vocoder Panel

6.2.18 Vocoder

6.2.18.1 Description

This is a port of the Vocoder from Native Instruments' ezFFT bundle for Reaktor 5 [116].

6.2.18.2 Controls and Terminals

Panel Controls:

- ATTACK: Attack time of the analyzer for the Modulator input.
- DECAY: Decay time of the analyzer for the Modulator input.
- TILT: Cuts or boost high frequency bins. At 12 o'clock, this knob provides no effect.
- CAR/MOD IN: Set the gain and polarity of the the Carrier and Modulator inputs.

Figure 6.40: Wavefolder Panel

Inputs:

- In Car: Carrier input.
- In Mod: Modulator input.

6.2.18.3 Design Notes

See Spectral Compressor design notes.

6.2.19 Wavefolder

6.2.19.1 Description

Wavefolders are one of the critical building blocks of "West-coast" style synthesizers. In a way, they act like reverse low-pass filters. Instead of taking a complex signal and removing higher frequencies, a wavefolder takes a simple signal (typically a sine or triangle wave), and "folds" its peaks to introduce harmonic content. A "Symmetry" control adds a DC offset to the input, leading to even more shapes.

6.2.19.2 Controls and Terminals

Panel Controls:

- FOLD: Controls the intensity of the folding operation by increasing the amount of gain applied to the input signal.
- ALGO: Chooses which wavefolding algorithm is used.
- SYMM: Adds DC bias to the signal, greatly affecting the timbre of the wavefold.

Modes:

- DIST - Foldover Distortion. This uses an algorithm found in the MusicDSP archives [118]. This is the harshest mode.
- SIN - Feeds the signal through a Sin function. This method has a linear gain multiplier of the input signal, so it folds evenly across the knob.
- SIN2 - Same as SIN, but uses an exponential function to scale the input signal. It increases slowly for part of the Fold knob, and then increases much more rapidly into very aggressive territory.
- OD - Idea taken from Madrona Labs' Max Patch prototype of the Aalto Timbre knob [119]. Same as SIN2 mode, but adds a variable Overdrive post-fold.
- OD 2 - Same as OD, but also added a second Overdrive post-gain, pre-bias.

6.2.19.3 Design Notes

Like the Bitcrusher, this is a modal design that is ultimately monosemous. The various modes of wavefolding are fairly similar, and there's only one output. I've considered adding simultaneous polymorphism to this Block by adding a "Fold Gate,"

Figure 6.41: Waveform Processor Panel

or a gate output is positive whenever the signal crosses the threshold required to fold. Alternatively, a similar output could be a square wave comparator output, similar to the one found on the Toppobrillo Triple Wavefolder [120].

6.2.20 Waveform Processor

6.2.20.1 Description

This is a waveform processor based off of the Doepfer A-136 [70]. This takes in a waveform and breaks it into its positive and negative components. The individual components can then be modified via gain and clipping controls. In addition to the A-136 control set, I have added input and output attenuverters, a "Split" control, and modulation on everything.

This can be used for a variety of distortions, like half-wave and full-wave rectification.

6.2.20.2 Controls and Terminals

Panel Controls:

- GAIN +/-: Determines the amplitude and polarity of the negative and positive components of the signal. As an example, "GAIN -" at 12 effectively mutes the negative component, leading to half-wave rectification. Full CCW inverts the negative component, creating full-wave rectification.
- CLIP +/-: Sets a hard-clip absolute value threshold for the negative and positive components.
- SPLIT: Adds a symmetrical amount of DC bias to the two components.

Outputs:

- Pos: Only the positive components of the input signal (half-wave rectification).
- Neg: Only the negative components of the input signal.

6.2.20.3 Design Notes

This Block (along with the A-136) is the inspiration for Dent, a modular distortion plug-in by my company Unfiltered Audio. I would consider this to be monosemous, despite the extra Positive and Negative outputs. Those outputs aren't polymorphic in my mind, as they do not serve a different enough function from the distortion itself.

6.2.21 Waveset

This is an implementation of Trevor Wishart's waveset processing concept [121]. A waveset is a collection of 3 zero crossings, or one full cycle of a waveform (i.e. a sine

Figure 6.42: Waveset Panel

wave).

At the start of each waveset, the probability setting is used to determine whether the waveset will be sent out of one of two outputs.

With audio signals, this can be used as distortion. It will sound like a faulty connection, but it doesn't have pops or clicks. With modulation signals, this is especially useful. You can use probability to turn modulations on and off after each cycle. You can also use the Trig outputs for triggers that are synced to modulations.

6.2.21.1 Controls and Terminals

Panel Controls:

- PROB: Determines the per-waveset probability that the waveset will be sent out of output 2.
- OUT 2: Sets the amplitude for Out 2.

Outputs:

Figure 6.43: Waveshaper Panel

- Out 1/2: Waveset outputs. The active output is determined per-waveset by the probability control.
- Mix: Both outputs summed together. This is useful when the OUT 2 control is not set to 100%.
- Trig 1/2: Fires a trigger whenever the associated output is active.

6.2.21.2 Design Notes

Despite the simplicity of this Block, it exhibits polymorphism. It can be used as a probabilistic switch, an intermittency effect (on the Mix output), or a trigger extractor.

6.2.22 Waveshaper

This Block contains two waveshapers from the Reaktor Core library: a Hyperbolic shaper and a Parabolic shaper. The Hyperbolic shaper provides a more dramatic

shaping effect.

6.2.22.1 Controls and Terminals

Panel Controls:

- SHAPE: Sets the shape of the signal. At 12 o'clock, the input signal is returned unchanged. Towards the negative side, the waveshaper provides a "needle" effect as signals are shaped towards 0.0. Towards the positive side, the waveshaper provides a "square" effect as signals are shaped towards +/-1.0.
- HYP/PAR: Selects the active shaping algorithm for the main output. HYPerbolic or PARabolic.

Outputs:

- Hyp: Dedicated Hyperbolic output. Unaffected by the GAIN OUT control.
- Par: Dedicated Parabolic output. Unaffected by the GAIN OUT control.
- Out: Switchable output. Affected by the GAIN OUT control.

6.2.22.2 Design Notes

This is a straightforward, monosemous design. This Block is intended as nothing more than a wrapper around the default NI macros. These shapers show up in more complex designs in Euro Reakt, including the Triple Ring Oscillator Block.

6.2.23 Wavetable Distortion

Wavetable distortion based on Sandy Small's excellent "Microwave Oscillator" Block [85].

Figure 6.44: Wavetable Distortion Panel

This is an effect version of a wavetable oscillator. Essentially, a wavetable oscillator has an internal sawtooth generator (called a phasor) that "looks up" values inside of a pre-defined wavetable. A wavetable distortion unit does away with the internal oscillator, and allows a composer to use any arbitrary input.

6.2.23.1 Controls and Terminals

Panel Controls:

- TABLE: Selects the active wavetable.
- WAVE: Sweeps through the selected table.
- INTRP/LIMIT: When INTRP is enable, the last three waves in each table are skipped. These waves are interpolated between waves 60 and 0 in the table--enabling LIMIT results in a sharper discontinuity between the last and first waves in the table.

- LIMIT/GLITCH: Affects how loud signals (in excess of +/- 1.0) are dealt with. In LIMIT mode, the signal is hard clipped before being used as a lookup. In GLITCH mode, loud lookup signals will bleed over into neighboring waves and tables. This can lead to really unusual wave splice combinations and other unusual artifacts.

- BRILLIANCE: Changes the type of interpolation used. At its highest setting, no interpolation is used. This leads to a bright, fizzy sound with lots of aliasing.

- IN LP: Sets the amount of low-pass filtering added to the input.

Three modes of rectification are available. Any signal can be used as an input, but only unipolar values in the range [0.0, 1.0] can be used to scan through the sample.

- UNI - Converts a bipolar signal into a unipolar signal. This adds 1.0 to the signal and then halves the amplitude.

- HALF - Negative component of signal is silenced.

- FULL - Negative component of signal is flipped (Takes absolute value of signal).

6.2.23.2 Design Notes

This is an extremely complex effect with a monosemous design. Wavetable distortion is a fairly expensive effect CPU-wise, so there aren't a lot of options for polymorphism. A polymorphic Eurorack wavetable distortion is The Harvestman's Piston Honda Mk. 2, which also acts as a wavetable oscillator. This design only makes sense in hardware due to the CPU demands.

6.3 Mixing

These Blocks are generally used to combine two or more signals. This category also features Blocks that assist with the mixing process, like panners, DC filters, and stereo field modifiers.

Many of these Blocks share the following controls, which will not be detailed in each Block's description unless notable:

Panel Controls:

- GAIN IN/OUT: Controls the level of the signal before and after processing. These can be unipolar or bipolar, indicated by the knob's graphics.
- x1/x2: Determines whether the Gain knobs can provide amplification.
- OFFSET: Adds a DC offset to the signal. This is usually a bipolar control that can add DC offset in either direction.
- DRY/WET: Controls the mix between the dry input and wet output.
- AC/DC: This enables an optional DC filter to remove bias from effects that can introduce it. I've included this switch especially for effects that double as CV processors.

6.3.1 2-to-4 and 4-to-4 Mix Matrices

6.3.1.1 Description

This Block implements a 4-input, 4-output mixing matrix developed by John Chowning and described by Julius O. Smith [122]. It was originally implemented as a primitive spatializer for the JCREV reverb. It is a novel way of taking in four inputs and

Figure 6.45: Mix Matrices

creating four mix variations. It was originally intended for quadriphonic spatialization. This Block implementation allows the matrix to process CV inputs, meaning that a composer could create four variations of four modulation sources. As an audio spatializer, its weakness is that two of the outputs are simply phase inversions of the other two outputs. This creates an interesting effect on specific setups, but can lead to major phase-cancellation issues in mixdowns.

6.3.1.2 Controls and Terminals

Panel Controls:

- In Gain 1/2/3/4: These determine the amplitude and polarity of the signals at the input terminals.

Inputs:

- In 1/2/3/4: Inputs to the mix matrix.

Output:

- Out 1: In 1 + In 2 + In 3 + In 4. In the 2-to-4 Mix Matrix, this output is (In 1 + In 2).
- Out 2: -(Out 1)
- Out 3: -(Out 4)
- Out 4: (In 1 + In 3) - (In 2 + In 4). In the 2-to-4 Mix Matrix, this output is (In 1 - In 2).

6.3.1.3 Design Notes

These are rhizomatic mixers that can be used either for the spatialization of audio or the distribution of variations of modulation inputs. They are not polymorphic, as their functionality does not change. In the future, it may be worth eliminating the 2-to-4 mixer, as using the first two inputs of the 4-to-4 mixer provides identical outputs.

6.3.2 8-Way Scanner

6.3.2.1 Description

This Block is a 1 to 8, 8 to 1, or 8 to 8 scanner/multiplexer. It is inspired by the Make Noise RxMx and the Toppobrillo Mixiplexer. This Block will smoothly crossfade 8 inputs to 1 output, 1 input to 8 outputs, or 8 inputs to 8 outputs. It can also output the amplitude of each channel, making it an 8-way modulation generator.

6.3.2.2 Controls and Terminals

Panel Controls:

Figure 6.46: 8-Way Scanner Panel

- INS (Switch): This switch determines whether one input will be sent to all eight outputs (1->8), or eight inputs will be sent individually to the outputs (8 INS).
- Scan: Sets the position of the wavescanner.
- Steps: Sets the number of channels that the wavescanner can address.
- OUTS/AMPS: This switch determines the behavior of the eight individual outputs. In OUTS mode, each output is a VCA. In AMPS mode, each output is a signal representing the amplitude of the channel's VCA.
- Width: Determines the width of the wavescanner. At a minimum setting, only one channel is active at a time. At its maximum setting, multiple channels will be active.
- Slope: Determines the shape of the wavescanner. At a minimum setting, the scanner has a rapid cutoff. At its maximum setting, there is a gentler rolloff.

Outputs:

- Main Out: All eight VCAs summed together. Unaffected by the OUTS/AMPS control.
- Out1-8: Individual outputs for each channel. Affected by the OUTS/AMPS control.

6.3.2.3 Design Notes

This is a modally and simultaneously polymorphic design. It is modally polymorphic as the eight individual outputs can be switched between VCA or amplitude curve outputs. It is also modally polymorphic with the various input and output configurations, as the user can decide whether to use it for 1->x, x->1, or x->x channel mixing, where "x" is determined by the STEPS control. It is simultaneously polymorphic in AMPS

6.3.3 Bit Mix

6.3.3.1 Description

Takes in two signals, turns them into 8-bit signals, and processes them in a bitwise fashion.

Each channel has a MODE control. This determines the bit encoding type for that input (See the A/D or D/A Blocks for more info on encoding/decoding types). A MODE switch on the bottom-right determines the decoding type. Make all three match for more predictable sounds.

Each channel has a MIX control. This determines the amplitude of the channel. Turn on the "X2" mode to use it as an amplifier.

Figure 6.47: Bit Mix Panel

Each channel has an OFFSET control. This occurs *after* the gain control. To help deal with DC offset, an AC/DC switch is on the bottom. In AC mode, DC offsets will be filtered out. Use AC mode for audio. Use DC for LFOs and modulation signals.

6.3.3.2 Controls and Terminals

Panel Controls:

- MIX 1/2: Set the gain and polarity of the signals present at each input.

TYPES:

- OR: Bitwise OR of both signals.
- AND: Bitwise AND of both signals.
- XOR: Bitwise XOR of both signals.

- INTER: Interleave bits. Most significant bit for output is taken from input 1. Next significant bit is taken from input 2. Next significant is taken from input 1...
- C-ELE: C-Element bitwise operation.
- SUM: Simple summing of both channels. Use for standard mixing.

6.3.3.3 Design Notes

Bit Mix, Bit Mix 32, and Logic Mix are all modally and simultaneously polymorphic, two-input mixer Blocks. Logic Mix was actually the first Block that I created for Reaktor 6 and was released the day after Reaktor 6 came out. It uses "analog" logic to mix two signals, using functions like analog AND (the minimum of two signals), analog OR (the maximum of two signals), sum, difference, and more. Bit Mix came later, using a similar design strategy of having two signals interact through logic functions. Bit Mix, however, uses 8-bit digital bitwise operations. Bit Mix 32 was a further refinement, eliminating the various 8-bit ADC/DAC modes and replacing them with one 32-bit signal path.

On all three Blocks, all functions are available simultaneously at separate outputs and are affected by the gain controls. The primary output on all three Blocks is modally polymorphic, as it follows the MODE knob's position.

6.3.4 Bit Mix 32

6.3.4.1 Description

This is a higher resolution version of the Bit Mix Block.

Figure 6.48: Bit Mix 32 Panel

It takes in two signals, converts them into 32-bit integers, and processes them in a bitwise fashion.

Each channel has a MIX control. This determines the amplitude and polarity of each channel. Turn on the "X2" mode to use it as an amplifier.

Each channel has an OFFSET control. This occurs *after* the gain control. To help deal with DC offset, an AC/DC switch is on the bottom. In AC mode, DC offsets will be filtered out. Use AC mode for audio. Use DC for LFOs and modulation signals.

6.3.4.2 Controls and Terminals

Panel Controls:

- MIX 1/2: Set the gain and polarity of the signals present at each input.

TYPES:

- OR: Bitwise OR of both signals.

Figure 6.49: Contrast Panel

- AND: Bitwise AND of both signals.
- XOR: Bitwise XOR of both signals.
- JUNK: A bunch of random logic operations cobbled together until it sounded good.
- FLIP: If In 1 is greater than In 2, flip the bits of In 1. Only outputs In 1.
- SUM: Simple summing of both channels. Use for standard mixing.

6.3.4.3 Design Notes

See Bit Mix Design Notes.

6.3.5 Contrast

6.3.5.1 Description

This effect is also referred to as "Audio MSG". It uses a bit of phase manipulation and waveshaping to brighten and boost a signal. This makes the signal stand out more in a mix. This effect also appears on the “Final Output” Block.

6.3.5.2 Controls and Terminals

Panel Controls:

- CONTRAST: Changes the amount of shaping present to increase brightness.

6.3.5.3 Design Notes

This is a monosemous design. A better, updated implementation of this effect is available in the Final Output Block, which combines this with a number of other useful, last-stage mixing utilities.

6.3.6 Crossfader

6.3.6.1 Description

This Block takes in two mono signals and creates a single, mono output that is a variable mix of the inputs. If XFADE is at 12 o’clock, a 50/50 mix of the signals will be present at the output. This crossfader does not use a smoothing interpolator, so the fade can be modulated at audio rates.

Figure 6.50: Crossfader Panel

6.3.6.2 Controls and Terminals

- XFADE: Controls the balance of the two inputs. At 12 o'clock, both inputs will appear at the output in a 50/50 mix.
- LIN/EXP: Determines the control curve used when crossfading. An EXPonential curve provides a more natural response for audible signals, while LINear is more predictable for fading between modulation signals.

6.3.6.3 Design Notes

This was one of the first Euro Reakt Blocks. It is a simple, monosemous design meant to add an essential function that was strangely absent from the first release of Reaktor 6. Native Instruments later added an XFade Block as part of their Bento Box series. My Flip Pan Block is intended as a polymorphic replacement for this.

Figure 6.51: DC Blocker Panel

6.3.7 DC Blocker

6.3.7.1 Description

This is a simple, control-free Block that removes DC offset from a signal by way of high-pass filtering. DC offset is a significant problem in modular synthesis. Many modules are capable of creating unipolar waveforms that may sound correct but present issues at the mixing stage (for instance, a unipolar envelope oscillating at audio rates).

6.3.7.2 Controls and Terminals

This Block only contains a simple input and output for the signal being processed.

6.3.7.3 Design Notes

This is perhaps the simplest Block in Euro Reakt. Almost every Euro Reakt Block that can produce DC offset has an optional DC filter on it. This Block is meant

Figure 6.52: Feedback Panel

to remove DC offset from signals that modulate between audible and extremely low frequencies. For a more interesting design, the Final Out Block combines a DC filter with many other useful last-stage mixing functions.

6.3.8 Feedback

6.3.8.1 Description

This Block was created to solve a Reaktor patching problem: a Block's outputs can not be plugged into its own inputs. Feedback can be created, but at least one other Block has to exist in the path.

6.3.8.2 Controls and Terminals

Four channels of feedback are available per Block, each with a simple input, output, and Mode switch. The modes are:

- T: "Instant" feedback.

Figure 6.53: Final Output Panel

- T-1: Use 1 sample back in the history.

"Instant" is in quotes because there will always be one sample of delay. The mode difference will be very subtle with low-frequency signals and/or low levels of feedback. At high levels of feedback, the difference can be striking.

6.3.8.3 Design Notes

As mentioned above, this Block was created simply to fix a major issue with Reaktor patching. Feedback is an essential tool in modular synthesis. Since this Block is so critical (and inexpensive, CPU-wise), I opted to add four instances of feedback, along with the ability to choose between 1 or 2 samples of feedback.

6.3.9 Final Output

6.3.9.1 Description

This Block combines a number of other Blocks into an easy, last-stage signal conditioner. It combines a master stereo level attenuator/amplifier, a switched DC filter, a multi-mode clipper (with indicator), a stereo width control, and a contrast control.

6.3.9.2 Controls and Terminals

- LEVEL: Attenuates or amplifies the signal.
- x1/x2: Determines whether the LEVEL knob can provide amplification.
- WIDTH: Affects the stereo image. At 12 o'clock, the stereo image is not affected. At full CCW, the stereo signal is averaged down to dual mono. At full CW, M/S processing is used to make the signal appear wider.
- CONTR: Sets the level of Contrast.
- SOFT/HARD: Determines how signals are handled when they exceed +/- 1.0. In hard clipping mode, the signals are simply lopped off beyond the threshold. In soft clipping mode, the signal is saturated using a gentle sigmoid curve.

6.3.9.3 Design Notes

Conceptually, the design of this Block is similar to Linked Polymorphism, but it's technically monosemous since the individual effects cannot be accessed separately. This is intentional, as this Block is specifically designed to be the last Block in a patch before the output, and not a highly-connected rhizomatic or polymorphic

Figure 6.54: Flip Pan Panel

Block. This is one of the most used Blocks in the User Library, as I have seen it show up in a large number of other people's ensembles.

6.3.10 Flip Pan

6.3.10.1 Description

This is a novel panning Block based on the Nonlinear Circuits Segue module for Eurorack [123]. The Segue uses a vactrol to shape the Pan voltage, which this Block does not model.

Simply put, this Block takes two mono inputs and pans them in different directions on the stereo field. This can be used in a number of ways:

- With one input and two outputs, this is a regular panner.
- With two inputs and one output, this is a regular crossfader.
- With one input and one output, this is a regular VCA.

- With two inputs and two outputs, this is an unusual panner that flips the inputs back-and-forth in the stereo field.

6.3.10.2 Controls and Terminals

Panel Controls:

- PAN: Sets the stereo position of the left input, and the inverse stereo position of the right input.

6.3.10.3 Design Notes

This is a modally polymorphic module where the mode is determined by which inputs are used. This Block is intended as a replacement for the monosemous, generic Panner and Crossfader Blocks. I've left the old designs in the library simple because they're such standard designs.

6.3.11 Logic Mix

6.3.11.1 Description

This Block takes in two signals and processes them using various logical operations. The individual modes are as follows:

- MAX - Takes the greater of the two inputs.
- MIN - Takes the lesser of the two inputs.
- RING - Multiplies the inputs together.
- PONG - Based off of the excellent Shapeshifter by Intellijel and Jim Clark [25]. If input 1 is positive, take that. Else, if input 2 is negative, take that. Else, 0.

Figure 6.55: Logic Mix Panel

- REKT - The positive portion of In 1 is summed with the negative portion of In 2.
- SUM - Basic mixing. In 1 + In 2.
- DIFF - Inverse mixing. In 1 - In 2.
- PING - Incorrectly implemented version of the Pong algorithm.
- TERRAIN - Wave Terrain Synthesis function, taken from Jim Clark's Nord Modular book [124], in turn taken from *The Computer Music Tutorial* [125].
- DIVFOLD - In 1 divided by In 2 (zeroes replaced by the previous value) and sent through a wavefolder. Extremely aggressive.

6.3.11.2 Controls and Terminals

Panel Controls:

- MIX 1/2: Determines the level and amplitude of each input.

Figure 6.56: Mono Widener Panel

- OFF. 1/2: Adds a positive or negative offset to either input.
- TYPE: Determines the active mode for the main output.

6.3.11.3 Design Notes

See Bit Mix design notes.

6.3.12 Mono Widener

6.3.12.1 Description

Takes a mono signal and enhances its image. It achieves this effect by adding a micro-delay to the right output.

6.3.12.2 Controls and Terminals

Panel Controls:

Figure 6.57: M/S Decoder Panel

- WIDTH: Manipulates the length of the delay lines to change the perceived width of the signal.

6.3.12.3 Design Notes

Unlike the Stereo Widener, this is a simple, monosemous design. It does not have alternative outputs like the Stereo Widener, as the algorithm for widening a mono signal is different.

6.3.13 M/S Decoder

6.3.13.1 Description

Takes a MID and SIDE input pair, and produces a standard LEFT/RIGHT output pair. For Mid-Side encoding, use the Stereo Widener Block.

Figure 6.58: Panner Panel

6.3.13.2 Controls and Terminals

Panel Controls:

- MID/SIDE - Gain amounts for the input channels. Can be used to invert one or both channels.

6.3.13.3 Design Notes

This is a partner Block to the Stereo Widener and MS Encoder Block. It is a monosemous design, as it is not intended for any other usage.

6.3.14 Panner

6.3.14.1 Description

Takes in a mono signal and pans it across a stereo field.

Figure 6.59: Stereo Widener Panel

6.3.14.2 Controls and Terminals

Panel Controls:

- PAN: Sets the stereo position of the input signal.

6.3.14.3 Design Notes

Like the Crossfader Block, this was one of the first Euro Reakt Blocks. It is a simple, monosemous design meant to add an essential function that was strangely absent from the first release of Reaktor 6. Unlike the Crossfader Block, Native Instruments has not added an adequate, voltage-controlled panner to Reaktor. My Flip Pan Block is intended as a polymorphic replacement for this.

6.3.15 Stereo Widener + MS Encoder

6.3.15.1 Description

This Block takes a stereo signal and enhances its image. It can be used to average a stereo signal down to dual mono, leave the stereo field unchanged, or produce a stereo image that seems wider and brighter.

This Block can also be used for M/S encoding. The MID and SIDE channels are available as outputs on the Block. For Mid/Side Decoding, an MS Decoder Block is available in Euro Reakt.

6.3.15.2 Controls and Terminals

Panel Controls:

- WIDTH - Changes the stereo image. At 12 o'clock, the image is unchanged. At full counter-clockwise, the stereo signal moves to dual mono. At full clockwise, a wider image is produced.
- LEFT/RIGHT - Gain amounts for the input channels. Can be used to invert one or both channels.

6.3.15.3 Design Notes

This is a simple mixing utility with simultaneous polymorphism. The stereo widener uses Mid/Side encoding as part of its algorithm, so the M/S signal is available at all times along with the processed stereo image.

Figure 6.60: Vector Mix Panel

6.3.16 Vector Mix

6.3.16.1 Description

This Block serves many purposes: 1) X-Y Mixing of four inputs (essentially a four-way linear crossfader). 2) Quad Panner 3) Quad Linear VCA 4) Complex CV/Waveform Generator

For X-Y Mixing, use four different inputs. Changing the X and Y values will morph between the four inputs. If you are using four oscillators, this is known as Vector Synthesis. For especially interesting waveforms, modulate X and Y with an audio-rate oscillator.

For Quad Panning/Distribution, take one signal and plug it into all four inputs. Monitor the individual outputs. If you want to do perfect circular panning, use the Quadrature LFO Block.

For a Quad VCA, use the Quad Panning recipe, but use four different inputs. The four outputs will be the four VCAs.

For Complex CV and Waveform Generation, modulate X and Y and monitor the various CV outputs. A UNI/BI switch (at the top of the Block) will change the range of this CV from 0-1 or -.5/+.5.

6.3.16.2 Controls and Terminals

Panel Controls:

- X/Y: Coordinate pair to determine which quadrant is active.
- Uni/Bi: This switch determines whether the six CV outputs are unipolar or bipolar.

Inputs:

- In A-D: Signal inputs for the four quadrants.

Outputs:

- Mix: Outputs A-D, summed.
- Out A-D: Individual outputs for each quadrant.
- CV A-D: Outputs for the current level of each quadrant. Useful for modulation. Affected by the Uni/Bi switch.
- CV X/Y: Outputs the modulated value of the X-Y knobs, affected by the Uni/Bi switch.

6.3.16.3 Design Notes

This is a modally and simultaneously polymorphic design. It is modal based off of which inputs and outputs are used. As listed in the description, the various input

Figure 6.61: AD Envelope Panel

and output configurations can lead to very different behaviors. In addition, the CV outputs can be used simultaneously with the audio outputs. A future design consideration would be to add an "In All" input for easily panning one source to all four outputs.

6.4 Modulation

6.4.1 AD Envelope and VCA

6.4.1.1 Description

This is a traditional, West-coast style AD Envelope and VCA. An AD envelope is a simple two-segment envelope, useful primarily for percussive tones and modulations. Like many West-coast AD envelopes, this one has three modes:

AD - Triggered Attack-Decay envelope. No matter the length of the incoming gate, the full attack phase will complete, followed by the full decay phase. This is

often called a "one-shot" envelope.

AHD - Gated Attack-Hold-Decay envelope. This envelope's length is determined by the length of the incoming gate. If the attack phase finishes while the gate is still high, the envelope will enter its Hold phase, and will hold until the gate goes low. At that point, it will move to its decay phase.

CYC - Looping AD Envelope/LFO. In this mode, the envelope will loop automatically. If it receives a new gate, it will return to the beginning of its attack phase.

6.4.1.2 Controls and Terminals

Inputs:

- GATE - A positive signal here will activate the envelope.
- VCA IN - Input to the voltage controlled amplifier. The amplitude of this signal will be controlled by the envelope.
- FREEZE - A positive signal here will hold the envelope at its current value.

Panel Controls:

- A.SHAPE - Controls the shape of the attack segment. At 12 o'clock, it's linear. To the left is logarithmic (fast initial rise then slower towards the top). To the right is exponential (slow intial rise, faster towards the top). NOTE: Changing the shape of the segment will not affect the length of the segment.
- D. SHAPE - Controls the shape of the decay segment. Works like the A. Shape knob, but in the opposite direction. Exponential is towards the left.

- ATTACK - Controls the length of the envelope's attack segment, from less than 1 ms to 1 second.
- DECAY - Controls the length of the envelope's decay segment, from less than 1 ms to 1 second.
- IN - Controls the amplitude of the input to the internal
- VCA. OUT - Controls the amplitude of the envelope on the OUT and VCA outputs.
- RES X/RES 0 - Controls the reset behavior of the envelope when a new trigger/gate is received. RES X means that the attack phase will start from the current envelope value. RES 0 means that the envelope will hard reset to 0 before the attack phase begins. RES X is generally smoother, but RES 0 can be useful for glitchy sounds or rapid modulations.
- x1/x10/x100/x1000 - Multiplies the length of the envelope segments. Yes, you can get 2000 second (over 30 minute) envelopes.

Outputs:

- OUT - Envelope output. Amplitude determined by OUT control.
- VCA - VCA output.
- EOA - End of Attack. This output is HIGH (+1) when the envelope is in its decay phase.
- EOC - End of Cycle. This output is HIGH when the envelope is not active. A trigger will appear here at the start/end of every cycle in CYCLE mode.

- -EOA - Opposite End of Attack. This output is HIGH when the envelope is not in its decay phase.
- +ENV - Positive copy of the envelope, not affected by the OUT panel control.
- -ENV - Negative copy of the envelope, not affected by the OUT panel control.

6.4.1.3 Design Notes

The AD and Trapezoid envelope generators are versatile Blocks that exhibit modal and simultaneous polymorphism. Both envelope generators take design cues from the Serge DUSG [24] and Make Noise Maths [23]. In both Blocks, modal polymorphism is created with the AD/AHD/CYC switch, which determines the behavior of the envelope. In CYC mode, the envelope oscillates, turning it into a useful LFO. At low ATTACK and DECAY settings, this will be an audible oscillator. They are simultaneously polymorphic as they have a gate outputs representing the current state of the envelope. I have expanded upon the Maths/DUSG design by adding a VCA input and output to both Blocks. This allows a user to quickly modulate the amplitude of a signal without needing another VCA Block and more connections.

6.4.2 Difference Rectifier

6.4.2.1 Description

This is based on a design by Nonlinear Circuits [126]. This circuit appears on a few of his modules, including the Neuron [127], the 1050 Mix Sequencer [128], and the Dual LFO [129]. It takes in a number of signals, finds the voltage difference between two sections, and outputs the rectified differences. A good meta-module is created by hooking up all four envelopes from NI's West Coast CFG Block, set them to cycle,

Figure 6.62: Difference Rectifier Panel

and monitor the outputs from this Block.

IN+ 1 and IN+2 are summed together to make up V+.

IN- 1 and IN- 2 are summed together to make up V-.

Diff = V+ - (V-)

If Diff is positive, it goes out the Out+ output. If Diff is negative, it goes out the Out- output.

6.4.2.2 Controls and Terminals

Panel Controls:

- IN 1/2 +/-: Set the gain and polarity of each input.

6.4.2.3 Design Notes

See the Neuron Design Notes.

Figure 6.63: Neuron Panel

6.4.3 Neuron

6.4.3.1 Description

This is a design from Nonlinear Circuits, typically built with a Difference Rectifier on the same panel [127]. It is based off of an analog model of a neuron [130]. Three signals are mixed together, manipulated, and run through a comparator. The comparator's output is summed with a variation of the input signal.

Use this to jumble and shred modulation signals. It works well on audio signals, but be sure to filter out DC offset after.

This Block has the following modifications over the hardware Neuron:

- All three inputs have attenuverters.
- Sense is bipolar.
- Response is bipolar. The hardware Neuron "leaks" a bit, so no Response on the hardware Neuron is about 0.1 on this Block.

- Output attenuverter.
- Separate comparator output.
- Separate mix output.

6.4.3.2 Controls and Terminals

Panel Controls:

- IN 1-3: Attenuversion for all three inputs, pre-comparator.
- SENSE: Controls the threshold above which the comparator will fire.
- RESPONSE: Sets the amplitude and polarity of the comparator's output.

Outputs:

- Out: Input mix summed with the comparator's output. Affected by OUT gain.
- Comp: Comparator output. Unaffected by OUT gain. Affected by RESPONSE.
- Mix: Sum of inputs. Affected by IN gains. Unaffected by OUT gain.

6.4.3.3 Design Notes

The original Nonlinear Circuits Neuron + Diff/Rect module is an independently polymorphic design, as the Neuron and Difference-Rectifier circuits are not normalled to each other. In Euro Reakt, I've broken out both circuits into their own Blocks and added a number of features.

Neuron was upgraded from a monosemous design to a simultaneously polymorphic design. In the original module, the only available output is the comparator

Figure 6.64: Quad Min-Max Panel

summed with the input mix, represented by the main output on this Block. However, on this Block, the comparator and input mix are available separately as well. The mixer has been improved. On the original module, there are only bare input jacks. Here, there are per-channel attenuverters. The Sense and Response controls have been expanded as well. They use a larger, bipolar range.

The Difference-Rectifier has also been upgraded. The Block is based on the Diff/Rect present on the Neuron module. In this configuration, there are four inputs (two positive, two negative). Each input now has an attenuverter. With the increased focus on mix precision, there's also a dedicated Mix output. This makes this version of the Difference-Rectifier simultaneously polymorphic.

6.4.4 Quad MinMax

6.4.4.1 Description

This Block takes in up to four signals and outputs the maximum and minimum values. This is identical to the Min/Max modes in Logic Mix, but with more inputs.

If you have unused inputs, those inputs will contribute "0.0" to the Min/Max equation. This means that Max will always be positive (since a negative signal is less than 0.0) and Min will always be negative. If you do not want these zeroes to be preset, simply fill each unused channel with duplicates of other inputs.

6.4.4.2 Controls and Terminals

Panel Controls:

- IN 1-4: Set the gain and polarity of each input.

6.4.4.3 Design Notes

This is a simultaneously polymorphic mixer. It takes in up to four signals and produces two opposing functions. Both functions are affected by the gain settings on every channel.

6.4.5 Quad Ranger

6.4.5.1 Description

Four separate unipolar-to-bipolar and bipolar-to-unipolar converters. This tool is mainly useful on modulation signals, but UNI2BI can be used as an interesting audio distortion.

Figure 6.65: Quad Ranger Panel

6.4.5.2 Controls and Terminals

Panel Controls:

- UNI2BI: Takes a unipolar signal and converts it to a bipolar signal (0.0 to 1.0 becomes -1.0 to 1.0). Note that the input will be hard clipped to 0.0-1.0.
- BI2UNI: Takes a bipolar signal and converts it to a unipolar signal.

6.4.5.3 Design Notes

This is an independently polymorphic design. Each of the four channels can be used for different purposes, but they do not affect each other in any way.

6.4.6 Quad Rectifier

6.4.6.1 Description

Four separate half- and full-wave rectifiers.

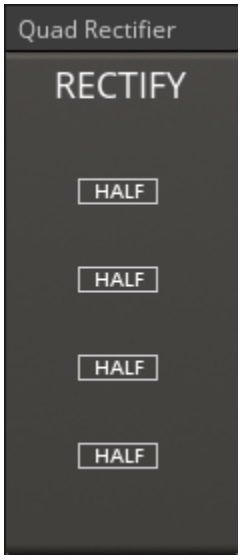

Figure 6.66: Quad Rectifier Panel

This tool is mainly useful on modulation signals, but can be used as an interesting audio distortion. If used for audio, a DC filter should be used after to center the signal.

6.4.6.2 Controls and Terminals

Panel Controls:

- HALF: Half-wave rectification removes negative components from a signal.
- FULL: Full-wave rectification takes the absolute value of a signal, flipping negative components into positive components.

6.4.6.3 Design Notes

Like the Quad Ranger, this is an independently polymorphic design. Each of the four channels can be used for different purposes, but they do not affect each other in any way.

Figure 6.67: Quadrature LFO Panel

6.4.7 Quadrature LFO

6.4.7.1 Description

A "Quadrature" oscillator is a sine oscillator that provides at least two outputs: sine and cosine. Many quadrature oscillators, like this LFO, provide 4 outputs: sine, cosine, -sine, and -cos. All of these outputs are frequency-locked. They are simply phase-shifted copies of the primary core.

This Quadrature LFO is based off of the Bento LFO. For this implementation, the saw output is used to address a sine table and a cosine table, providing the 0 and 90 degree outputs. Those outputs are inverted, providing the 180 and 270 degree outputs.

Quadrature LFOs are useful for creating modulations that chase each other, or for mirror modulations.

This Quadrature LFO is unique in that it retains the Bento LFO's "shape" parameter, allowing you to warp the lookup phasor. This creates really interesting

push-pull modulations.

6.4.7.2 Controls and Terminals

Panel Controls:

- SHAPE: Applies a phase distortion to the internal phasor.
- PHASE: Sets the phase that the LFO resets to.

Inputs:

- Freeze: When a positive signal is present here, the internal phasor will hold its value until the signal is no longer positive.
- Reset: When the signal present here transitions to a positive state, the internal phasor will immediately reset to phase 0. It will not reset again until the signal goes negative and then positive again.

6.4.7.3 Design Notes

This is a rhizomatic design that can interact with a patch in many ways, especially with the gate inputs. I would not consider it to be polymorphic, as the four outputs share identical functions but at different phases.

6.4.8 Trapezoid Envelope and VCA

6.4.8.1 Description

This is a complex envelope based on the envelope found in the classic Synthi. It is very similar to the AD Envelope Block, but it has two additional stages. The HOLD ON Stage adds a held stage to the top of the envelope, while the HOLD OFF Stage

Figure 6.68: Trapezoid Envelope Panel

adds spacing to every cycle in CYC mode. This can be useful for creating extremely short envelopes with long spaces between them.

Like many West-coast envelopes, this one has three modes:

AHD - Triggered Attack-Hold-Decay envelope. No matter the length of the incoming gate, the full Attack phase will complete, followed by the full Hold On and Decay phases. This is often called a "one-shot" envelope.

ASHD - Gated Attack-Sustain-Hold-Decay envelope. This envelope's length is determined by the length of the incoming gate. If the Attack phase finishes while the gate is still high, the envelope will enter its Sustain phase, and will hold until the gate goes low. At that point, it will move to its Hold On phase before finishing with its Decay phase.

CYC - Looping AHDH Envelope/LFO. In this mode, the envelope will loop automatically. If it receives a new gate, it will return to the beginning of its attack phase. Hold Off will determine the space between cycles.

6.4.8.2 Controls and Terminals

INPUTS:

- GATE - A positive signal here will activate the envelope.
- VCA IN - Input to the voltage controlled amplifier. The amplitude of this signal will be controlled by the envelope.
- VELOCITY - Determines the amplitude of the envelope when VEL ON is enabled.
- FREEZE - A positive signal here will hold the envelope at its current value.

OUTPUTS:

- OUT - Envelope output. Amplitude determined by OUT control.
- VCA - VCA output.
- EOA - End of Attack. This output is HIGH (+1) when the envelope is in its decay phase.
- EOC - End of Cycle. This output is HIGH when the envelope is not active. A trigger will appear here at the start/end of every cycle in CYCLE mode.
- -EOA - Opposite End of Attack. This output is HIGH when the envelope is not in its decay phase.
- +ENV - Positive copy of the envelope, not affected by the OUT panel control.
- -ENV - Negative copy of the envelope, not affected by the OUT panel control.

Figure 6.69: Trigonometric Shaper Panel

6.4.8.3 Design Notes

See the Design Notes for the AD Envelope and VCA Block.

6.4.9 Trigonometric Shaper

6.4.9.1 Description

This multi-mode waveshaper contains a number of classic trig functions: Sine, Cosine, and Tangent. In addition, all three functions have Regular, Arc, and Hyperbolic modes.

This is mainly useful for modulation signals, but it can also be very useful for audio. For example, Hyperbolic Tangent is frequently used as a digital overdrive algorithm.

6.4.9.2 Controls and Terminals

Panel Controls:

- FUNC: Choose the active trig function on the main output. This can be Sine, Cosine, or Tangent.
- MODE: Choose the active function mode on every outputs. This can be Regular, Arc, or Hyperbolic.

Outputs:

- Sin: Dedicated output for Sine function. Affected by the MODE control.
- Cos: Dedicated output for Cosine function. Affected by the MODE control.
- Tan: Dedicated output for Tangent function. Affected by the MODE control.

6.4.9.3 Design Notes

This is a modally and simultaneously polymorphic voltage processor. Using one input provides three very different outputs. Every output is affected by the MODE control. In the 4.0 update, an optional DC filter was added to make this more useful as an audio effect as well.

6.4.10 Voltage Mirror

6.4.10.1 Description

A "Voltage Mirror" is a unipolar inverter. If you have an ascending unipolar envelope, this will provide a descending unipolar envelope.

This differs from typical inversion. Most inverters flip a signal around the x-axis, meaning that positive signals become negative, and negative become positive.

The Voltage Mirror's output is equal to (1.0 - input). The input is rectified before this occurs. Thus, the output of the Voltage Mirror is always positive. This positive, unipolar output can be made negative by using the bipolar Out Gain control.

Figure 6.70: Voltage Mirror Panel

6.4.10.2 Controls and Terminals

Panel Controls:

- HALF/FULL: Determines whether the input goes through HALF-wave or FULL-wave rectification.

6.4.10.3 Design Notes

This is a monosemous design with one input, one output, and no alternative uses. This may seem like a strange design to include in the library, but a Voltage Mirror is a common patching strategy that takes some effort to set up and calibrate correctly. This Block takes the guesswork out of this technique.

Figure 6.71: Wavetable LFO Panel

6.4.11 Wavetable LFO

6.4.11.1 Description

This is a wavetable LFO based on Sandy Small's excellent "Microwave Oscillator" Block [131], which is a wavetable oscillator that emulates the Waldorf Microwave [132].

This replaces the frequency controls on the Microwave Oscillator with the lower frequency controls from the Bento Box LFO. To emphasize low-frequency operation, RESET and FREEZE controls and inputs have been added. A PHASE knob sets the phase of the waveform upon receiving a reset signal. The BRILLIANCE knob has been replaced by two simple modes of interpolation.

Additions over both the Bento Box LFO and Microwave Oscillator:

- Out knob for easy amplitude control or phase inverting.
- Dedicated unity-gain UNIpolar output.

- Dedicated unity-gain BIpolar output.
- Dedicated Freeze input.
- Dedicated Phase output.
- Reset output.

6.4.11.2 Controls and Terminals

Panel Controls:

- PHASE: Determines the phase of the LFO when it receives a reset signal.
- TABLE: Determines the active wavetable.
- WAVE: Selects the active wave from the current wavetable.
- INTRP/LIMIT: When INTRP is enabled, the last three waves in each table are skipped. These waves are interpolated between waves 60 and 0 in the table. Enabling LIMIT results in a sharper discontinuity between the last and first waves in the table.
- SMOOTH/LO-FI: SMOOTH enables linear interpolation. LO-FI turns off interpolation, leading to a heavily stepped signal.
- INSTANT/PHASE 0: Determines the behavior when a user changes the WAVE or TABLE. INSTANT means that the new waveform is selected immediately. PHASE 0 means that it waits until the LFO reaches phase 0, usually a zero crossing.
- UNI/BI: Sets whether the main output is unipolar or bipolar.

Inputs:

- Reset: A positive gate here will reset the LFO to the phase set by the PHASE control.
- Freeze: A positive gate here will hold the LFO in its current phase. It will not advance until the gate is zero or negative.

Outputs:

- Uni/Bi: Dedicated unipolar and bipolar outputs for the LFO. Unaffected by the OUT gain control.
- Phase: Unipolar sawtooth waveform representing the current phase of the LFO.
- Reset: Outputs a trigger whenever the LFO cycles or whenever a positive gate is received on the Reset input.

6.4.11.3 Design Notes

This is a rhizomatic, simultaneously polymorphic design. The Reset output adds a timing trigger that can be used at the same time as the primary modulation signal. There are four output variations of the LFO itself, including a variable amplitude main output, two unity-gain outputs, and one phase output. There are also multiple inputs that allow the LFO to interact with other timing signals.

6.4.12 XY to Polar

6.4.12.1 Description

This Block uses a Cartesian-to-Polar equation to manipulate input voltages. The input voltages are treated as (X, Y) coordinates on a Cartesian plane. The R and A

Figure 6.72: XY-to-Polar Panel

outputs are these coordinates translated to Polar Radius and Azimuth values.

This is not meant to be used for accuracy (Like, say, cartopol~ in Max). Out R is hard clipped, and Out A is reduced in amplitude by 75%. This Block is intended more to bend modulations into unusual shapes.

6.4.12.2 Controls and Terminals

Outputs:

- R: Radius output. Hard clipped.
- A: Azimuth output. Reduced in amplitude by 75% to keep signals within bounds.

6.4.12.3 Design Notes

Like cartopol~ in Max, this is a straightforward, monosemous design. Its only function is to receive an (X, Y) pair and convert them to polar coordinates. A potentially

interesting module design could be created by combining a quadrature source with this type of coordinate conversion.

6.5 Oscillators and Sound Sources

This category of Blocks is primarily used to generate audible signals. They include various oscillators and triggered percussion sources.

Many of these Blocks share the following controls, which will not be detailed in each Block's description unless notable:

Panel Controls:

- PITCH: This is a big blue knob on most oscillators. It either chooses the pitch of the oscillator (or some aspect of the signal chain described on each Block) as an offset (in semitones) of the voltage present at the Pitch input, or an exact frequency (in Hz) that ignores signals present at the Pitch input.

- KEYBOARD: This is a keyboard icon that sits next to the big blue PITCH knob. It determines whether the PITCH control acts in semitone offsets or exact frequencies.

- FM: Sets the depth of frequency modulation over the oscillator (or some aspect of the signal chain described on each Block). The FM modulator is the signal present on the "FM" input terminal.

- FM TYPE: Selects an FM algorithm. These are EXPonential, LINear, and LINear Thru-Zero.

- WAVEFORM: This is a big orange knob on some oscillators. It smoothly crossfades between Sine, Triangle, Saw, and Square waves.

- PW: Most oscillators that have the WAVEFORM control will also have this. This knob selects the Pulse Width of the Square waveform. 12 o'clock gives a symmetrical pulse width.

- OUT: Controls the amplitude of the oscillator at its primary output.

Inputs

- Pitch: When the KEYBOARD icon is active, this input sets the base pitch of an oscillator (or some aspect of the signal chain described on each Block).

- FM: The signal present here will be used to modulate the frequency of an oscillator (or some aspect of the signal chain described on each Block). The depth of this modulation is controlled by the FM knob.

A significant number of these oscillators are ports of oscillator modes from Braids, an open-source "macro oscillator" by Mutable Instruments. These were ported with Olivier Gillet's approval.

6.5.1 Clap

6.5.1.1 Description

Triggerable drum/noise source.

This Block is based on synth patches found in Jim Clark's Nord Modular Book [124]. It attempts to roughly model the clap synthesis method on a TR-808 or TR-909. It's not meant to be a 100% accurate model, but it is a great way of quickly reaching a usable clap sound.

A noise source runs through an amplitude envelope that is rapidly triggered four times in a row. On the last trigger, a slower "reverb" envelope is opened up. The

Figure 6.73: Clap Panel

multiple triggers give the effect of multiple people clapping simultaneously, while the final envelope simulates the dissipation of sound throughout a space.

6.5.1.2 Controls and Terminals

Panel Controls:

- FREQ: Determines the rate at which new noise samples are generated. If an external input is used as the noise source, this acts as a sample rate reducer.
- STUTTER: Determines the spacing of the four noise triggers. At low settings, it will provide a tight, loud clap. At high settings, you will hear the individual noise bursts.
- REVERB: Determines the length of the envelope opened at the end of the stutter burst. This envelope provides a low-quality reverb effect.
- SHAPE: Controls the shape of the noise envelope's decay stage.

- CUTOFF: Controls the cutoff of the two filters acting upon the noise. The filters are a band-pass filter and a low-pass filter setup simultaneously in parallel and serial configurations. The band-pass filter's output is the primary noise source, while the low-pass filter's output is summed with the band-pass to create the reverb source.
- DECAY: Controls the decay length of the primary noise envelope.
- RES: Controls the resonance of the two noise filters. At high values, this will impart a distinct pitch to the usually noisy sound.
- WHITE/LFSR/EXT. IN: Chooses the noise source. WHITE is white noise, LFSR is a low frequency shift register (which provides a more lo-fi digital sound). EXT. IN uses the signal present at the "Ext. In" input and runs it through a sample rate reducer.
- GATE: Manually trigger the Clap via a click.

Inputs:

- Gate: Triggers the Clap's envelopes.
- Ext. In: If a signal is present here, it can be used in place of the internal noise generators.

Outputs:

- Out: Main Clap output.
- Noise: Outputs the band-pass filtered noise source only without any amplitude modification.

- Reverb: Outputs the reverb tail only.
- Amp. Env.: Outputs the amplitude envelope.
- Rev. Env.: Outputs the reverb envelope.

6.5.1.3 Design Notes

This Block makes heavy use of Simultaneous Polymorphism. It simultaneously outputs a percussive clap sound, a constant noise source, a filtered "reverb" tail, and two envelopes. The envelopes appear as a rapid burst, followed by a slow, last-stage envelope. Finally, it can also be used as a sample rate crusher via the external input and dedicated noise output. I wanted to add a "Burst" output for the gate trigger burst, but it proved to be too difficult with the design. To prevent unpleasant retriggers, the envelopes have a different type of internal gating than what the rest of Euro Reakt typically uses. In a later update, I plan on redoing that section.

6.5.2 Comb Oscillator

6.5.2.1 Description

This is based off of one of the modes ("/|/|_|_|_") from Mutable Instruments' excellent open-source Braids Macro Oscillator. In this generator, a sawtooth wave runs through a comb filter.

The saw oscillator follows the tuning of the keyboard input and frequency knob, while the comb filter's frequency is offset from the saw's frequency via the SPREAD control (this mimics the TIMBRE knob on the Braids model). The FEEDBK control determines the amplitude and polarity of the comb filter's feedback (this follows the COLOR knob on the Braids model). In addition to the standard Braids features, I

Figure 6.74: Comb Oscillator Panel

added an additional MIX control to balance between the dry sawtooth oscillator and the wet, post-comb signal.

6.5.2.2 Controls and Terminals

Panel Controls:

- SPREAD: Controls the tuning relation between the sawtooth oscillator and the comb filter. At 12 o'clock, they run at the same frequency.
- MIX: Controls the mix between the dry sawtooth oscillator and the wet, post-comb signal.
- FEEDBK: Controls the amplitude and polarity of the comb filter's feedback.

Outputs:

- Out: Main output.

Figure 6.75: Complex Oscillator Panel

- Saw: Outputs the sawtooth waveform, unaffected by the comb filter or the OUT control.

6.5.2.3 Design Notes

This is a fairly simple design that can lead to great timbral results. Aside from working as an unusual LFO, there's no polymorphism or flexibility present. An idea for a future update would be to add a VCA between the oscillator and the comb filter, thus allowing for the creation of resonant percussion and sustained comb echoes.

6.5.3 Complex Oscillator

6.5.3.1 Description

One of the foundations of West-Coast synthesis. A complex oscillator is generally a dual oscillator with many internal modulation busses between the two. This type of design can be traced back to early Buchla systems, and is present in many modern

Eurorack oscillators, including the Make Noise DPO, The Harvestman's Hertz Donut, Intellijel's Shapeshifter, Sputnik Modular's Complex Oscillator, and many, many more.

This Complex Oscillator is based on two Bento Box oscillators. Internally, they are connected via Frequency Modulation, Amplitude Modulation, and variable-strength Sync. Both oscillators also have last-stage wavefolding for even more timbral flexibility.

6.5.3.2 Controls and Terminals

Panel Controls:

- FOLD: Adjusts the intensity of the wavefolder for each oscillator.
- SYNC: Adjusts the intensity of syncing from the opposing oscillator. At full CW, this is a traditional "hard sync".
- AM: Adjusts the amount of amplitude modulation applied to each oscillator.
- AM/AM R/RING: Determines the type of amplitude modulation applied to each oscillator. AM is standard amplitude modulation (where negative modulations are thrown away). AM R is rectified AM (negative modulations are full-wave rectified). RING is ring modulation.
- SLEW: Sets the amount of time it takes for Oscillator 2 to reach its target frequency.
- P. 1/2: Sets the Pitch input that Oscillator 2 listens to.
- OSC/EXT: Determines whether each oscillator receives AM and FM from the opposing oscillator or the dedicated FM and AM inputs.

Outputs:

- Mix: 50/50 blend of both oscillators.
- Osc1/2: Individual oscillator outputs.

Inputs:

- AM In: Dedicated input for AM signals. These signals will only modulate the amplitudes of oscillators set to EXT for their AM source.

6.5.3.3 Design Notes

This was an excellent early release for Euro Reakt, as it came out before Native Instruments released DWG, their take on a complex oscillator Block. My Block has a few advantages over the Native Instruments design. First, neither oscillator in my design is considered "Primary". The two oscillators are of equal complexity, and both have a wavefolder (in the NI design, only the Carrier oscillator runs through the Timbre circuit). Second, the bidirectional multi-mode AM is unique to my design. It provides a great timbral alternative, and can also create interesting modulation sources when used as LFOs. Finally, I prefer the way that I implemented the FM and AM buses. Instead of having a dedicated input for FM Carrier and FM Modulator, one FM input and one AM input can target both oscillators. A mode switch allows you to choose on each oscillator whether they're receiving internal modulation from the opposite oscillator, or external modulation.

That being said, the NI design is still excellent, especially the Timbre circuit. I ended up creating a standalone effect Block based on the Timbre section, meaning that you can use my oscillator design with their waveshaper. One of my other preferred meta-modules is to connect the two outputs from this oscillator into my Logic Mix Block, which provides a number of interesting two-operator instructions.

Figure 6.76: Drum Panel

6.5.4 Drum

6.5.4.1 Description

Triggerable drum/noise source, based on drum synthesis recipes from Gordon Reid's Synth Secrets column for Sound on Sound [133].

One trigger activates three envelopes: Oscillator amplitude, Noise amplitude, and Oscillator pitch. All three are simple decay envelopes with nearly instant attack.

The left half is the Oscillator. This uses the Bento Box core. The right half is a dual-mode noise source and multi-mode resonant filter. In the center are mixing controls.

6.5.4.2 Controls and Terminals

INPUTS:

- Gate - Triggers the Drum.
- Pitch - Standard pitch input when Keyboard mode is on.

- FM - External FM
- Ext. In - Optional replacement for internal oscillator
- Noise - Optional replacement for internal noise source (pre-filter)

OUTPUTS:

- OUT - The final drum sound will appear here.
- NOISE - Noise source, unfiltered.
- OSC - Internal oscillator
- OSC ENV - Amplitude envelope for internal oscillator
- NOISE ENV - Amplitude envelope for noise source
- PITCH ENV - Pitch envelope for internal oscillator

OSCILLATOR SIDE CONTROLS:

- OSC SIDE FM - FM Amount for internal oscillator. Has three modes: Exponential, Linear, and Linear thru-zero.
- INT/EXT - Replaces the internal oscillator with the signal at the EXT. IN input.
- DECAY (Bottom Left) - Envelope length for the oscillator's pitch envelope. The depth of this envelope is controlled by P. ENV.
- PITCH - Big blue knob. Controls coarse and fine tuning. Switch to enable Keyboard Mode is on the top-right. When using an external input, this controls the sampling rate of the input.

- WAVEFORM - Internal oscillator's waveform. When using an external input, this will wavefold the input.
- DECAY - Envelope length for the oscillator's amplitude.
- LOG - Logarithmic envelope. Very long initial sustain.
- LIN - Linear envelope. Even rolloff from 1 to 0.
- EXP - Exponential envelope. Very rapid die-off from maximum value.

MIX CONTROLS:

- XFADE - Crossfade between the noise source and the oscillator.
- OUT - Main output level. Does not affect NOISE or OSC outputs. x1/x2 - Chooses how loud the output can get.

NOISE SIDE CONTROLS:

- DECAY - Envelope length for the noise source's amplitude
- COLOR - Changes the tone of the noise source. When using an external noise source, this will affect the sampling rate.
- LFSR - Low frequency shift register noise source. Useful for raw digital sounds.
- WHITE - White noise source. Much brighter sounding.
- INT/EXT - Replace the internal noise source with the signal present at NOISE input.
- CUTOFF - Controls the cutoff of the filter.
- LP/HP/BP - Choose between Low-Pass, Band-Pass, and High-Pass filters.

Figure 6.77: FM Oscillator Panel

- RES - Resonance of the filter F.ENV - Controls how much the cutoff of the filter is affected by the noise amplitude envelope.

6.5.4.3 Design Notes

This started off as an 808 bass drum emulator and ended up turning into one of the more polymorphic sound sources in Euro Reakt. It works well for all sorts of synthetic percussion, including kicks, snares, hi-hats, toms, claves, and more. This acts as a good focal point of a patch, as every trigger generates three independent envelopes that can be used to modulate other Blocks. A common strategy is to plug the modulation envelope into a different oscillator Block and use that to replace the internal oscillator. This means that Drum can take practically every generator in Euro Reakt and turn it into a percussion voice.

6.5.5 FM Oscillator

6.5.5.1 Description

This is a dual oscillator inspired by the "FM", "FBFM", and "WTFM" modes on the Mutable Instruments Braids Oscillator.

Inside of this Block, there are two oscillators: Carrier and Modulator. The Carrier oscillator receives frequency modulation from the Modulator. The depth of this FM is controlled by the "INT FM" knob. The Modulator's frequency is an offset of the Carrier's base frequency. There are many ways to make these oscillators interact.

6.5.5.2 Controls and Terminals

Panel Controls:

- SPREAD: Controls the Modulator's frequency as an offset of the Carrier's base frequency. At 12 o'clock, both oscillators will have the same base frequency.
- EXT FM: Sets the depth of frequency modulation applied from the FM input to both the Carrier and Modulator oscillators.
- INT FM: Sets the depth of frequency modulation applied from the Modulator to the Carrier.
- C->C: Sets the depth of frequency modulation the Carrier imparts on itself.
- C->M: Sets the depth of frequency modulation the applied from the Carrier to the Modulator.

Outputs:

- Carrier: Outputs the Carrier oscillator, affected by the OUT gain parameter.

Figure 6.78: Fold Oscillator Panel

- Mod: Outputs the Modulator oscillator, affected by the OUT gain parameter.
- Mix: Outputs a 50/50 blend of the Carrier and Modulator, affected by the OUT gain parameter.

6.5.5.3 Design Notes

This is a simultaneously polymorphic design. At its heart are two oscillators. Turning the main frequency control will affect all three outputs. Turning the SPREAD control affects the Mod and Mix outputs. A user could ignore the FM functionality entirely and use the Mix output as a two partial additive voice. Unlike the monosemous Fold Oscillator, wiring up something comparable to this Block would take a lot more effort. This design is actually quite close to a "complex oscillator" (like Make Noise DPO or Harvestman Hertz Donut) but without variable waveform selection.

6.5.6 Fold Oscillator

6.5.6.1 Description

This is based off of the "FOLD" mode from Mutable Instruments' Braids Macro Oscillator. In this generator, a blend of a sine and/or a triangle oscillator runs through a wavefolder.

6.5.6.2 Controls and Terminals

Panel Controls:

- FOLD: Controls the strength of the wavefolding operation by increasing the amplitude of the oscillator.
- BIAS: Changes the symmetry of the wavefolding operation by adding DC offset to the wavefolder's input. This is a bipolar control, so 12 o'clock adds no bias.
- SIN/TRI: Crossfade between sine a triangle shapes for the oscillator.

6.5.6.3 Design Notes

This is a monosemous design and is less flexible then using an oscillator of your choice with the dedicated Wavefolder Block. There are dedicated, unfolded outputs so that you can sum the base waveform with the folded output for a more layered sound. I've opted to exclude an external input, as that's the purpose of the Wavefolder Block. Still, this can be a useful Block for users who want the specific wavefolder sound quickly.

Figure 6.79: Harmonic Oscillator Panel

6.5.7 Harmonic Oscillator

6.5.7.1 Description

This is an additive oscillator that combines 8 sine waves. With a simplified set of controls, you can set the spacing of the 8 waves, choose the central harmonic, and scan between all 8 harmonics. This is partially based off of the "HARM" mode in Mutable Instruments' Braids Oscillator, but has a number of major differences (In fact, some may point out that due to the nature of the SPREAD control, it's not necessarily a "Harmonic" Oscillator, but a versatile Additive Oscillator Bank). This oscillator features individual outputs for each oscillator, allowing you to create whatever mix you want, instead of relying only on the internal wavescanner.

6.5.7.2 Controls and Terminals

Panel Controls:

- SPREAD: Sets the frequency relationship between the eight oscillators. At 12

o'clock, the oscillators are harmonics of the base frequency (x1, x2, x3... x8). At full CW, each oscillator doubles from its 12 o'clock frequency.

- HARM: Sets the most prominent oscillator in the wavescanner.
- WIDTH: Controls the width of the wavescanner. At full CCW, only one oscillator is audible at a time. At full CW, nearly the entire bank will be audible.
- OSCS/SCAN: In OSCS mode, each individual oscillator output is constant amplitude. In SCAN mode, the amplitude of each output is equal to the amplitude of each oscillator at the main output.

Outputs:

- Osc 1-8: Each oscillator is available independently here. The amplitude of this output is determined by the OSCS/SCAN mode switch.

6.5.7.3 Design Notes

Aside from Braids, this also takes design cues from the Verbos Harmonic Oscillator and the Make Noise RxMx mixer. The RxMx is a six-channel mixer with "Channel" and "Radiate" parameters that behave very similarly to the HARM and WIDTH controls here. A more generic Block with this functionality is the 8-Way Scanner Block in the Mixing category. Since the highest oscillator can have up to 16 times the frequency of the bottom oscillator, this can cover a large simultaneous frequency range, making this a simultaneously polymorphic Block under certain conditions.

Figure 6.80: Hi-Hats Panel

6.5.8 Hi-Hats

6.5.8.1 Description

This is a triggerable drum/noise source that takes in two triggers (HH GATE and OH GATE) and produces two "hi-hat" drum sounds (HH - Closed Hi-Hat, OH - Open Hi-Hat). To produce a hi-hat sound, two sound sources are used. The first is a Low-Frequency Shift Register. This produces a random 8-bit noise tone. The second is a stack of tuned square waves. These are mixed together and then multipled by a simple decay envelope.

6.5.8.2 Controls and Terminals

Inputs:

- HH Gate: A positive signal on here will trigger the closed hi-hat sound.
- OH Gate: A positive signal on here will trigger the open hi-hat sound.

OUTPUTS:

- Out: Both drum sounds will appear here.
- HH: Closed hi-hat only
- OH: Open hi-hat only
- Noise: LFSR noise, post-filter
- Square: Stacked squares, post-filter
- Mix: Noise + Square, post-crossfade.

CONTROLS:

- FREQ - Determines the frequency of the shift register, changing the overall timbre of the noise. Also determines the pitch of the stacked oscillators.
- CUTOFF - Determines the cutoff of the high-pass filter on the noise source, and the band-pass filter on the stacked oscillators.
- XFADE - Crossfade between the noise source and the stacked oscillators.
- DECAY - Determines the decay time of the exponential decay envelope for the open hi-hat sound only. At minimal settings, this can be used to produce hits that are even shorter in length than the closed hi-hat.
- OUT - Output level for the MIX output.
- 606/808/110 - Chooses the frequency spacing for the stacked oscillators.
- x1/x2 - Chooses how loud the output can get.

Figure 6.81: Impulse and Sinc Train Panels

6.5.8.3 Design Notes

This Block exhibits simultaneous polymorphism. It generates a hi-hat sound on the main output while generating a stable noise source and stacked square oscillator at the same time. A planned future update would allow for a user to select a noise source. The choices would include LFSR, white noise, or an external input. This behavior would match the behavior on the Clap and Drum Blocks.

6.5.9 Impulse Train + Sinc Train

6.5.9.1 Description

Generates an impulse train (or "Dirac Comb"). Each impulse lasts for only one sample. This is useful for pinging filters (try it on the Comb Filter for string-like tones) or activating triggers on certain Blocks.

The RAND knob controls how much variation appears between trigger frequencies. The internal random generator appears at the "Rand CV" output and can be

used to affect other Blocks on a per-impulse basis.

The Sinc Train is very similar to the Impulse Train, but it generates sinc-shaped impulses. These are very similar to band-limited impulses and sound great as standalone grains or as generators for both grain envelopes or grain oscillators.

6.5.9.2 Controls and Terminals

Panel Controls:

- RAND: Adds an amount of random gaussian disturbance to the frequency of the generator. This random disturbance is calculated only once at the beginning of each cycle.
- HARM (Sinc only): Decreases the width of the sinc impulse, increasing the perceived frequency.
- UNI/BI (Impulse only): In UNI mode, all impulses are positive only. In BI mode, the polarity of each impulse alternates.

Outputs:

- Phase: Outputs the current phase of the oscillator, from 0-1.0.
- Rand CV: If the RAND knob is above 0, a random CV will appear here after each impulse is generated.
- Filtered (Impulse only): A filtered version of the main output. The impulse runs through a gentle, one-pole filter with a cutoff of 1000 Hz.

6.5.9.3 Design Notes

A simple impulse generator is a very useful building block for microsound. I went for a simultaneously polymorphic design here. The gaussian FM is inspired by a

Figure 6.82: Karplus Panel

SuperCollider plug-in uGen called GaussTrig. I decided to take the random FM and break it out onto its own output, making this a good simultaneous master clock and random source. It also has a simultaneous phase output which can be used for synced modulations. The phase output is especially useful with an upcoming Window Generator Block, as it can be used to create grain envelopes equal in length to the impulse generator's cycle period.

6.5.10 Karplus

6.5.10.1 Description

Classic String-tone generator. A quick-noise burst is run through a tuned delay line with heavy feedback.

This Block features the Bento Oscillator FM Core for tuning the delay line, giving it the ability to create wild granular patterns or spacey detuned strings. You can choose between two noise types (White or LFSR), and can change the decay and

cutoff of the noise. In addition to this, you can use external signals of your own design.

6.5.10.2 Controls and Terminals

Inputs:

- Pitch: Frequency of the delay line.
- Gate: Triggers the internal amplitude envelope
- FM: Frequency modulation source for the delay line.
- Ext. In: External Input

Outputs:

- Burst: Post-envelope, pre-delay signal (i.e. the input to the delay line).
- Noise: Noise source, post filter, pre-envelope.
- Noise Env: Unipolar decay envelope, activated whenever a positive signal is received on the Gate input.

Controls:

- FREQ - Big blue knob. Controls the frequency of the delay line.
- FM - Controls the amount of FM upon the delay line.
- OFST. - Amount of signal that bleeds through the internal decay envelope. Modulate this if you wish to use an envelope of your own design. Raise it to 1 if you're using your own percussive input.
- DECAY - Decay time of the internal amplitude envelope.

- COLOR - When using an internal noise source, this acts as a high-pass filter. When using an external source, this acts as a sample-rate reducer.
- WHITE/LFSR - Choose the internal noise type
- INT/EXT - Choose between internal noise or an external signal.
- OUT - Output level.

6.5.10.3 Design Notes

This is a simultaneously and modally polymorphic design with a lot of functionality. It is modally polymorphic as it can be used as a standalone, triggered oscillator. Alternatively, it can be used to process external audio. Like the Comb Filter effect, it is also modally polymorphic due to its extremely large frequency range. It can be used for the classic Karplus string effect, or it can be used as a straightforward, filtered echo generator. Finally, it demonstrates simultaneous polymorphism as it is capable of creating the main output, a noise output, and an envelope at the same time.

6.5.11 Pulsar Oscillator

6.5.11.1 Description

This is a simple implementation of Pulsar Synthesis, a form of Microsound described by Curtis Roads.

In this simplified version, a variable waveform oscillator is windowed by a gaussian oscillator. Instead of a FORMANT control, this implementation has a SPREAD parameter to determine the pitch offset of the Gaussian window generator vs the main oscillator.

Figure 6.83: Pulsar Oscillator Panel

6.5.11.2 Controls and Terminals

Panel Controls:

- PROB: Changes the probability of the Gaussian window being generated each cycle.
- SPREAD: Controls the pitch offset of the Gaussian window oscillator from the main oscillator.
- WIDTH: Controls the shape of the Gaussian window oscillator. Counter-clockwise provides needle shapes, while clockwise widens the window and ultimately produces offset.

Outputs:

- Osc: Oscillator output. Affected by OUT.
- Window: Gaussian window output. Unipolar. Affected by PROB and OUT.

Figure 6.84: Resonating Bar Panel

6.5.11.3 Design Notes

This is a first attempt at a Pulsar synthesizer for Euro Reakt. I am planning on creating a more in-depth Block with more parameters and better control over the windowing. Still, this Block displays elements of simultaneous polymorphism, as the oscillator and envelope generator are output separately and can generate at very different frequencies. Changing the speed of either affects the main Out.

6.5.12 Resonating Bar

6.5.12.1 Description

This is a port of the "Bar Resonator" module from Chet Singer's classic Ampere Modular for Reaktor 5 [84]. This Block replicates the sound of metallic percussion. It can be used as either a triggered drum source, or as a sound processor.

6.5.12.2 Controls and Terminals

Exciter:

- INT: Use an internal decay envelope and noise source to excite the resonator.
- EXT: Use the input at EXT. IN to excite the resonator.

Noise types:

- LFSR: Low Frequency Shift Register. A white-like noise generator with a more digital flavor.
- EXT. IN: Replaces the noise generator with an external input. Changing COLOR downsamples this input.
- WHITE: Standard white noise generator.

Panel Controls

- CUTOFF: Sets the cutoff frequency for the filter. This filter is between the input and the resonator. Use it to emphasize high frequencies, or to remove them entirely.
- LP/BP/HP: Changes the type of filtering applied to the noise source (Low Pass, Band Pass, High Pass).
- F. ENV: Determines if the filter's cutoff point changes when the internal envelope is triggered. This has no effect when "EXT" mode is selected.
- HARM: Changes the harmonic profile of the resonator. Counter-clockwise dampens the bar, while higher values open it up and add more content.

- COLOR: Changes the generation rate of the internal noise generator. Downsamples the external input if it is selected.
- DECAY: Sets the length of the noise source's amplitude envelope.
- LIN/EXP/LOG: This button cycles through the available envelope shapes for the noise amplitude envelope.

Outputs:

- Noise: Noise generator. Unaffected by filtering or gain controls.
- Filt. Noise: Enveloped noise, post-filter.
- Noise Env.: Noise generator's amplitude envelope.

Inputs:

- Ext. In: Used to replace the noise generator when EXT IN mode is selected.

6.5.12.3 Design Notes

Resonating Bar and Wood are two of the most complex generators in Euro Reakt. They can be used as generators or effects. The basic design was inspired by Mutable Instruments' Rings [134] and Elements [135] physical modelling modules. Those modules can act as triggered percussion sources or as effect processors (where the input is fed directly to the resonant filter bank instead of an enveloped noise source). These Blocks are modally and simultaneously polymorphic. They are modally polymorphic as the INT/EXT switch can be used to change between percussion and effect modes. They are simultaneously polymorphic as either mode can be used simultaneously with noise and envelope generation on the various outputs.

Figure 6.85: Resonating Wood Panel

6.5.13 Resonating Wood

6.5.13.1 Description

This is a port of the "Wooden Body Resonator" module from Chet Singer's classic Ampere Modular for Reaktor 5 [84]. It is a close relative of the "Resonating Bar" Block. This Block, however, is strongly inharmonic. The algorithm isn't necessarily good for only wood tones. It can produce bells, metal, and glass tones as well.

6.5.13.2 Controls and Terminals

Exciter:

- INT: Use an internal decay envelope and noise source to excite the resonator.
- EXT: Use the input at EXT. IN to excite the resonator.

Noise types:

- LFSR: Low Frequency Shift Register. A white-like noise generator with a more digital flavor.

- EXT. IN: Replaces the noise generator with an external input. Changing COLOR downsamples this input.

- WHITE: Standard white noise generator.

Panel Controls

- CUTOFF: Sets the cutoff frequency for the filter. This filter is between the input and the resonator. Use it to emphasize high frequencies, or to remove them entirely.

- LP/BP/HP: Changes the type of filtering applied to the noise source (Low Pass, Band Pass, High Pass).

- F. ENV: Determines if the filter's cutoff point changes when the internal envelope is triggered. This has no effect when "EXT" mode is selected.

- RES: Sets the resonance of the 32 internal resonators. Lower values will start to make the internal noise source more audible.

- SPREAD: Changes the spacing of the 32 internal resonators. Lower values mean that the resonators are clustered together closer to the fundamental frequency. Higher values produce aggressively inharmonic tones.

- COLOR: Changes the generation rate of the internal noise generator. Downsamples the external input if it is selected.

- DECAY: Sets the length of the noise source's amplitude envelope.

- LIN/EXP/LOG: This button cycles through the available envelope shapes for the noise amplitude envelope.

Figure 6.86: Rungler Oscillator Panel

Outputs:

- Noise: Noise generator. Unaffected by filtering or gain controls.
- Filt. Noise: Enveloped noise, post-filter.
- Noise Env.: Noise generator's amplitude envelope.

Inputs:

- Ext. In: Used to replace the noise generator when EXT IN mode is selected.

6.5.13.3 Design Notes

See the design notes for Resonating Bar above.

6.5.14 Rungler Oscillator

6.5.14.1 Description

This Block is built around Rob Hordijk's "Rungler" circuit [136]. For more information, look at the standalone Rungler Block. More specifically, this Block is loosely based on Hordijk's "Benjolin" circuit [137], but it features a few changes (namely, two Runglers).

In this implementation, the Block consists of two Bento-core Oscillators and two Runglers. Oscillator 1 (top) provides the clock for Rungler 1, and the data for Rungler 2. Oscillator 2 (bottom) provides the clock for Rungler 2, and the data for Rungler 1.

Each oscillator has two FM inputs. The FM knob controls the depth of modulation from either the opposing oscillator or the FM IN (controlled by the OSC/EXT switch under the FM knob). The RUNGLER knob controls the amount of modulation provided by the internal Runglers (selectable via the RUNG1/RUNG2 switch below the RUNGLER knob). Like the Bento Box oscillator, there are three modes of FM (EXPonential, LINear, and THRU-ZERO LINear).

Like the standalone Rungler Block, the sensitivity of the DATA input for each Rungler is controlled by the COMP knob. Each Rungler can also be set to LOOP.

6.5.14.2 Controls and Terminals

Panel Controls:

- RUNGLER: Controls the amount of FM modulation applied to the oscillator via one of two internal Runglers.
- EXT/OSC 1/2: Select the FM source for the FM knob. Choose between the

opposing oscillator or the FM IN input.

- RUNG 1/2: Select which internal Rungler is used for FM.
- COMP 1/2: Control the internal comparator on the DATA input (in this case, the opposing oscillator). Essentially, this will control how sensitive the DATA input is. Below 12 o'clock, you'll receive a lot of positive bits. Above 12 o'clock, you will find fewer.
- WRITE/LOOP: LOOP will lock the shift register's contents. It will still advance with a positive GATE input, but Bit 8 will be passed to Bit 1 instead of new DATA being read. WRITE continously reads new data.

Outputs:

- Mix: Both oscillators summed together in a 50/50 blend.
- Osc1/2: Independent oscillator outputs.
- Rung1/2: Direct, stepped outputs from the Runglers.

6.5.14.3 Design Notes

This is a very complex generator capable of modal and simultaneous polymorphism. It is simultaneously polymorphic as it can be used as an oscillator and a random modulation generator at the same time. It is modally polymorphic, as either oscillator can be dropped down to LFO rates via the frequency control. When dropped to LFO rates, the associated Rungler performs better as a step sequencer.

Figure 6.87: Snare Panel

6.5.15 Snare

6.5.15.1 Description

Based on synth patches found in Jim Clark's Nord Modular Book [124] and Sound on Sound's Synth Secrets column [133]. It attempts to roughly model the snare synthesis method on a TR-808 or TR-909. It's not meant to be a 100% accurate model, but it is a great way of quickly reaching a usable snare sound.

The left half is composed of two triangle oscillators. These are meant to emulate the tone of the drum heads being struck. The right half is composed of a noise source and two filters. These are meant to emulate the tone of the snares on the bottom of the drum. By using only the oscillator half, you could use this as a primitive Tom-Tom generator. By using only the noise half, you could use this as a lo-fi cymbal source.

6.5.15.2 Controls and Terminals

Panel Controls:

- FREQ: Determines the frequency of the two triangle oscillators. Their harmonic ratio remains stable. Using the 12 o'clock setting produces a value closest to the intended sound.
- RATIO: Determines the decay ratio of the oscillators vs. the decay of the noise source. At full clockwise, the oscillators will decay at the same rate as the noise source. Using a setting around 12 o'clock yields the most typical results.
- P. ENV: Controls the amount that the oscillators will detune over the course of the amplitude envelope. Leave at low values for a more natural sound.
- XFADE: Crossfade between the triangle oscillators and the noise source.
- CUTOFF: Determines the cutoff of the serial low-pass and high-pass filters for the noise source. Full clockwise gives the most natural sound.
- DECAY: Determines the decay time of the exponential decay envelope for the noise source only.
- RES: Controls the resonance of the serial filters on the noise side. Can be used to produce very unusual snares.

Noise Types:

- LFSR: Low Frequency Shift Register. More digital, lo-fi noise source.
- WHITE: White noise.

- EXT IN: Use the signal present at the Ext. In input instead of an internal noise generator.

Inputs:

- Ext. In: Signal used when EXT IN is selected as the noise source.

Outputs:

- Noise: Noise generator, post-filtering. Unaffected by OUT.
- Oscs: Two triangle oscillators. Unaffected by OUT.
- Noise Env.: The noise half's decay envelope. Unipolar, unaffected by OUT.
- Osc. Env. The oscillators' decay envelope. Unipolar, unaffected by OUT.

6.5.15.3 Design Notes

Like Clap and Drum, this is a simultaneously polymorphic design. It can be used as a triggered drum generator, a filtered noise source, a stacked triangle oscillator pair, and a dual envelope generator at the same time.

6.5.16 SumSyn Oscillator

6.5.16.1 Description

This is a summation synthesis Block, based on an algorithm found in Noise Engineering's "Loquelic Iteratis" [138]. It differs from his implementation in that it doesn't have internal Phase Modulation or a more sophisticated waveshaper. However, it adds multi-mode FM and a switchable Fold/Hard-clip output.

In this method of synthesis, three sine wave oscillators are used. Sine 1 (Main Freq) and Sine 2 (Main Freq - Spread Freq) are summed together in a way based on

Figure 6.88: SumSyn Oscillator Panel

the BRIGHT setting. They are divided by Sine 3 (Spread Freq, and 90 degrees out of phase, so technically a cosine wave), which runs through a complicated shaping algorithm first. At the final stage, the entire mix runs through either a hard clipper or a wavefolder (The wavefolder is used in the original implementation).

6.5.16.2 Controls and Terminals

Panel Controls:

- BRIGHT: This control introduces higher harmonics into the output, making the main output sound brighter.
- SHAPE: Each sine wave runs through a hyperbolic waveshaper (like the one present in the Waveshaper Block). This control affects the shape of all three internal oscillators.
- SPREAD: Controls the frequency relationship between Sine 1 and 2. Directly sets the frequency of Sine 3.

- FOLD/CLIP: Select whether the final mix runs through a wavefolder or hard clipper.

Outputs:

- Sin1-3: Dedicated outputs for all three oscillators. These outputs are tapped post-waveshaping.
- Sin Mix: All three sine waves mixed together and averaged.

6.5.16.3 Design Notes

This is a rhizomatic and polymorphic design with a large number of linked outputs. In addition to the complex timbres available from the main output, the three oscillators can be mixed independently or together at the Sin Mix output. Thus, this could be used as a basic additive oscillator with three sines while also being used as a Summation Synthesis voice.

6.5.17 Sync Oscillator

6.5.17.1 Description

This is based off of the “SYNC” mode from Mutable Instruments’ Braids Macro Oscillator. In this generator, two oscillators are hooked up in the classic "hard-sync" patch style, where a master oscillator phase resets a second oscillator whenever it completes a cycle.

The main oscillator follows the tuning of the keyboard input and frequency knob, while the synced oscillator’s frequency is offset from the main oscillator’s frequency via the SPREAD control.

Figure 6.89: Sync Oscillator Panel

6.5.17.2 Controls and Terminals

Panel Controls:

- SAW/SQR: Crossfades between Saw and Square waveforms at the main Out.
- SPREAD: Controls the frequency relationship between the main and synced oscillators.
- MIX: Controls the balance between the Main oscillator and the Syncing oscillator.

Outs:

- Main Saw: Dedicated output for the main oscillator's sawtooth wave. Unaffected by the OUT control.
- Sync Saw: Dedicated output for the syncing oscillator's sawtooth wave. Unaffected by the OUT control.

Figure 6.90: Toy Oscillator Panel

- Main Sqr: Dedicated output for the main oscillator's square wave. Unaffected by the OUT control.
- Sync Sqr: Dedicated output for the syncing oscillator's square wave. Unaffected by the OUT control.

6.5.17.3 Design Notes

This is a rhizomatic design with a lot of outputs. It could possibly be considered simultaneously polymorphic in that it could generate a synced LFO (main saw out) and gate (synced out) at different or linked frequencies.

6.5.18 Toy Oscillator

6.5.18.1 Description

This is based off of the "TOY*" mode from Mutable Instruments' Braids Macro Oscillator. In this generator, a variable waveform runs through various bit manipulating

operations to sound like a low-quality or circuit-bent toy.

6.5.18.2 Controls and Terminals

Panel Controls:

- PW: Controls the pulse width of the square shape of the oscillator.
- S. RATE: Controls the sampling rate of the oscillator. Does not affect the oscillator's audible frequency.
- GLITCH: Determines the strength of the bitshifting operations.

6.5.18.3 Design Notes

Aside from being able to operate at LFO rates, this is a monosemous design intended only to emulate the mode from Braids. I expanded on the Braids design by adding a variable waveform with PW control, but that's about it. The most intriguing part of the original design is the "Glitch" control, which combines a number of bitshift and logic operations to simulate circuit bending sounds. To make that more reusable, I took that part of the algorithm and added it as a mode to the Bircrusher effect Block.

6.5.19 Triple Bento

6.5.19.1 Description

This is inspired by a few of the modes ("/|/|x3", "-_-_x3", "/\x3", and "SIx3") from Mutable Instruments' Braids Macro Oscillator. In this generator, three waves are summed together. Unlike Braids, instead of using a static waveform stack, this

Figure 6.91: Triple Bento Panel

oscillator is built around the variable-shape Bento oscillator from the Reaktor 6 standard Blocks Library.

Osc 1 follows the tuning of the keyboard input and frequency knob, while Osc 2 and Osc 3 are offset from Osc 1's frequency via the SPREAD controls.

6.5.19.2 Controls and Terminals

Panel Controls:

- SPREAD1/2: Sets the frequency of the secondary oscillators as a relationship to the main oscillator. At 12 o'clock, the frequencies are equal.

Outputs:

- Osc 1-3: Individual oscillator outputs, affected by the OUT control.

Figure 6.92: Triple Ring Panel

6.5.19.3 Design Notes

This is a rhizomatic design. It combines three extremely simple oscillators into one quick interface. The SPREAD controls (which can be independently modulated) are useful for making either chords or a detuned stack of oscillators. The main oscillator can drop down to .1 Hz, and the secondary oscillators can go even lower (via SPREAD). Because of this, it can double as an interesting multi-LFO.

6.5.20 Triple Ring

6.5.20.1 Description

This is based off of the "RING" mode from Mutable Instruments' open-source Braids Macro Oscillator. In this generator, three sine waves are multipled (ring modulated) with each other before running through a waveshaper.

Sine 1 follows the tuning of the keyboard input and frequency knob, while Sine 2 and Sine 3 are offset from Sine 1's frequency via the SPREAD controls.

6.5.20.2 Controls and Terminals

Panel Controls:

- SPREAD1/2: Sets the frequency of the secondary oscillators as a relationship to the main oscillator. At 12 o'clock, the frequencies are equal.
- SHAPE: Controls a hyperbolic waveshaper that occurs after the cascaded ring modulation. At 12 o'clock, the sine timbre is preserved. At full CW, it turns the output into square waves only. At full CCW, the output is closer to impulses.

Outputs:

- Sin1-3: Individual sine wave outputs, unaffected by the OUT or SHAPE controls.
- 1 x 2: Oscillators 1 and 2 ring modulated, unaffected by the OUT or SHAPE controls.

6.5.20.3 Design Notes

Like the SumSyn oscillator, this is a rhizomatic design with a large number of linked outputs. In addition to the complex timbres available from the main output, the three oscillators can be mixed independently. Thus, this could be used as a basic additive oscillator with three sines.

6.5.21 Twin Peaks

6.5.21.1 Description

This is based off of one of the modes from Mutable Instruments' excellent open-source Braids Macro Oscillator. In this generator, white noise runs through parallel

Figure 6.93: Twin Peaks Panel

BP filters. Both of them share the same resonance.

BP1 follows the tuning of the keyboard input and frequency knob, while BP2 is offset from BP1's frequency via the SPREAD control.

6.5.21.2 Controls and Terminals

Panel Controls:

- SPREAD: Sets the frequency of the second band-pass filter in relation to the main frequency.
- RES: Sets the resonance of both band-pass filters.
- BP1/BP2: Controls the mix of the filters present at the main output.

Outputs:

- White: Dedicated white noise generator.
- BP1/2: Outputs for both filters, affected by the OUT gain control.

Figure 6.94: VOSIM Oscillator Panel

6.5.21.3 Design Notes

This is a simultaneously polymorphic design. It can be used to generate white noise along with multiple "pitched" outputs (depending on how resonant the filters are).

6.5.22 VOSIM Oscillator

6.5.22.1 Description

This Block implements a synthesis method that is very similar to VOSIM (but not exactly accurate). This alternative method is described by Rob Hordijk [139].

"Real" VOSIM depends on an impulse train with uneven spacing. This uses a sawtooth oscillator as an envelope generator with synced sine waves to act as the formants. This oscillator includes further modifications from Hordijk's implementation.

VOSIM is a type of synthesis that excels in producing vocal and vowel tones. It is often considered to be a form of granular synthesis. In this implementation, a

sawtooth wave creates a grain envelope for a two sine generators. The sine generators operate at a frequency multiple of the sawtooth wave (determined by the FORMANT knobs). This grain is then multiplied by itself (in SIN^2 mode) or by a rectified version of itself (DIODE mode).

One modification that I made is the addition of a hyperbolic waveshaper. At 12 o'clock, normal sine waves are produced. Towards full clockwise, you will get something closer to square waves (giving you something like a lo-fi VOSIM). Towards full counter-clockwise, glitchy impulses are produced.

6.5.22.2 Controls and Terminals

Panel Controls:

- FORM. 1/2: Determines the frequency of the sine generator in relation to the frequency of the saw envelope.
- SHAPE: Hyperbolic waveshaper. Modifies the internal sine waves. 12 o'clock produces a normal tone. Full clockwise gives lo-fi squares, full anti-clockwise gives glitchy impulses.
- 1/2: Controls the balance of the two sine generators.
- SIN^2/DIODE: In SIN^2 mode, the grain ring modulates itself. In DIODE mode, the grain is multiplied by a rectified version of itself. DIODE mode gives a more symmetrical waveform with a lower perceived frequency.

Outputs:

- Osc 1/2: Dedicated oscillator outputs. These are tapped after waveshaping occurs.

- Saw Env: Unipolar sawtooth oscillator (labelled "Env" because it is the grain envelope).

6.5.22.3 Design Notes

This is a simultaneously polymorphic design capable of generating two bipolar sine oscillators and one unipolar saw oscillator at the same time. It has a very wide frequency range, so this can be changed to two sine LFOs and one saw envelope. The frequency of all three outputs can be separate.

6.6 Noise and Chaos

This category of Blocks is focused on the generation of unpredictable audio and/or modulation signals. A significant number of these Blocks were ported directly from SuperCollider and expanded with more controls.

Many of these Blocks share the following controls, which will not be detailed in each Block's description unless notable:

Panel Controls:

- OUT: Controls the level of the signal present at the Block's output. This can be unipolar or bipolar, indicated by the knob's graphics.

- FREQ: "Frequency" is a bit of a misnomer for this control, as most of these Blocks do not produce cyclical signals. However, this control determines the frequency of the Block's internal clock that is used to calculate new samples.

- FAST/SLOW: Dramatically changes the frequency range of the FREQ control. In SLOW mode, the lowest speed is 0.04 seconds *per sample*.

Figure 6.95: 1-Op Chaos Panel

- STEPPED/LINEAR: Turns on linear interpolation between samples. At audio rates, this will sound like a low-pass filter. At modulation rates, new values will appear in a smooth, sliding fashion. This is equivalent to the "L" variation of each SuperCollider chaos uGen.

- AC/DC: This enables an optional DC filter to remove bias from effects that can introduce it. I've included this switch especially for effects that double as CV processors.

- CHAOS x: Controls a named coefficient within a given chaotic equation.

Outputs:

- Out: Primary signal output.

6.6.1 1-Op Chaos

6.6.1.1 Description

This Block combines a number of 1-operator chaos Blocks into an easy, switch-based interface.

6.6.1.2 Controls and Terminals

The following chaotic equations are found using the MODE switch:

- CRACKLE: A re-implementation of SuperCollider's Crackle uGen. 0-0.5 on the CHAOS knob will produce a pleasant hiss with color variations. Higher values introduce pops and clicks.
- BAD CRACKLE: A failed re-implementation of SuperCollider's Crackle uGen (while I was first learning Reaktor Core). 0-0.5 on the CHAOS knob will produce a pleasant hiss with color variations. Higher values introduce screams and machine noise.
- IKEDA: An implementation of the Ikeda Chaotic Map [140]. At high chaos values, this becomes a chaotic attractor. At lower chaos values, it provides stable oscillations.
- LOGISTIC: A re-implementation of SuperCollider's Logistic uGen. At low frequencies, this provides random stepped modulation. At high frequencies, this will produce wild modem sounds.
- STANDARD: A re-implementation of SuperCollider's StandardN and StandardL uGens. These uGens are based off "an area preserving map of a cylinder discovered by the plasma physicist Boris Chirikov". In musical terms, it's a

chaotic oscillator that is good at maintaining a stable state before breaking up into unusual behavior. It works well as both a modulation source and as an audio-rate noise source.

- TENT: Chaos generator based on the Tent Map equation [141]. It provides fairly stable triangle-shaped oscillations.

Outputs:

- Out X/Y: Most of the above chaotic equations have two dimensional outputs. For the one-dimensional equations, the Y output is simply -X.

6.6.1.3 Design Notes

Before writing this dissertation, there were eleven single-equation chaos Blocks. Considering that they all had roughly the same functionality, this was wasteful design. If a user wanted to experiment with various forms of chaos, they would need to delete their current chaos Block, add in a different one, and rewire it. This design uses modal polymorphism in two ways. First, the various chaos modes have largely different purposes. Crackle is great as a pleasant audio source, while Standard works well for stable oscillations. Second, the frequency range and interpolation switches have huge impacts on the output, selecting between smooth and stepped modulation or full audio functionality. After creating these Blocks (1-Op, 2-Op, and 3-Op Chaos), they are now my favorite Blocks for unpredictable modulations.

Figure 6.96: 2-Op Chaos Panel

6.6.2 2-Op Chaos

6.6.2.1 Description

This Block combines a number of 2-operator chaos Blocks into an easy, switch-based interface.

6.6.2.2 Controls and Terminals

The following chaotic equations are found using the MODE switch:

CUSP A re-implementation of SuperCollider's CuspN and CuspL uGens. This creates a chaotic map based on the following equation:

$x[n+1] = a - b * \sqrt{|x[n]|}$

At low frequencies, this produces a useful stepped or smoothed semi-random sequence. (Why "semi-random"? Well, it's chaotic, meaning that it's a deterministic sequence. It's not a typical pseudo-random generator. Its output tends towards repetition with only minor variation.) At high frequency, it provides stable tones

with intermittent dropouts.

This Block is recommended much more for modulation than for audio. The two knobs have a lot of "dead spots" where a DC offset or 0 value will be produced. These occur pretty suddenly. It makes for an interesting modulation that occasionally holds its value, but it will produce random silence if being used as a heavily modulated oscillator.

An optional DC Blocking filter is available at the output.

GAUSS CHAOS A chaos oscillator of my own design, based in part on Supercollider's LFGauss uGen. That uGen creates a oscillation or envelope with the cycle shape of a Gaussian window. This implements that equation, but uses feedback instead of a phasor for the equation's input.

The LFGauss equation is:

$f(x) = \exp((x - iphase)^2/(-2.0 * width^2))$

In this Block, "x" is the previous output value of f(x). Chaos A is iphase, and Chaos B is width.

At certain settings, this can create very stable oscillations.

HENON A re-implementation of SuperCollider's HenonN and HenonL uGens. This creates a chaotic map based on the following equation:

$x(n + 2) = 1 - a * x(n + 1)^2 + b * x(n)$

The Y output is equal to the b*x(n) portion of the equation, so the scaling of that output is heavily dependent on B.

HETRICK A variation on Henon created after a programming error.

Figure 6.97: 3-Op Chaos Panel

MOUSE This is an implementation of the Mouse Map, also known as the Gauss Iterated Map [142].

There are a lot of regions that provide stable oscillations in this map. At higher chaos values, you will see repeating waveforms with interesting interruptions.

6.6.2.3 Design Notes

See 1-Op Chaos Design Notes.

6.6.3 3-Op Chaos

6.6.3.1 Description

This Block combines a number of 3-operator chaos Blocks into an easy, switch-based interface.

6.6.3.2 Controls and Terminals

The following chaotic equations are found using the MODE switch:

LCC (Linear Congruent Chaos) A re-implementation of SuperCollider's LinCongN and LinCongL uGens. These uGens are based on the following chaotic equation:

$x[n+1] = (a * x[n] + c)\%$

This is one of the pickier chaos generators. Some settings provide extremely stable and rich oscillations. Other settings are pure noise. Sometimes the controls feel like they are doing nothing. Othertimes, a tiny adjustment will have a massive effect!

QUADRATIC A re-implementation of SuperCollider's QuadL and QuadN uGens. These uGens are based on the following difference equation:

$x(n+1) = a * x(n)^2 + b * x(n) + c$

6.6.3.3 Design Notes

See 1-Op Chaos Design Notes.

6.6.4 Brusselator

6.6.4.1 Description

This is a chaotic grain generator, partially based on the Brusselator SLUGen for SuperCollider [143]. It simulates a chemical reaction that typically dies off quickly. With extreme parameter settings, it can oscillate.

Figure 6.98: Brusselator Panel

It has an internal impulse generator that rapidly retriggers its initial conditions. The reason for this is because it frequently "dies off" to zero values. In a way, it's a biologically inspired grain generator.

6.6.4.2 Controls and Terminals

Panel Controls:

- FREQ: This behaves differently from the other FREQ knobs on the Euro Reakt chaos generators. This control will affect the perceived frequency of the system.
- DELTA: This behaves more like the other FREQ controls. It determines how quickly new values are calculated.
- GAMMA & MU: These two controls work in concert to determine the timbre of the system and whether or not it will self-oscillate.
- REGEN: Sets the frequency of the internal impulse generator. At 0, it will not auto-generate and will need to be manually triggered.

- DISTURB: Adds an amount of gaussian randomization to the frequency of the impulse generator.
- SEED: Manually resets the system with new initial conditions.

Inputs:

- Reset: A positive zero-crossing transition here will reset the system with new initial conditions. This can be combined with the internal impulse generator.

Outputs:

- Out X/Y: Chaotic outputs.
- Trig: Outputs a trigger whenever the system is reset. This is equal to the sum of the internal impulse generator, the SEED panel control, and triggers received at the Reset input.

6.6.4.3 Design Notes

This is a simultaneously polymorphic design, as it generates an impulse train and a chaotic signal. It could be considered modally polymorphic at certain settings, as the chaotic signal can be either a steady oscillator or triggered percussion source.

6.6.5 Chaotic 2D/3D Attractors

6.6.5.1 Description

These two Blocks contain a number of 2D and 3D chaotic attractors, most of which use 4 operators.

Figure 6.99: Chaotic Attractor Panels

6.6.5.2 2D Modes

The TYPE knob selects the following attractors on the 2D Block:

DE JONG This is a chaotic attractor described by Peter de Jong [144].

CLIFFORD This is a chaotic attractor described by Clifford Pickover [144].

MODIFIED LATOOCARFIAN This is a chaotic attractor described by Clifford Pickover and implemented as a SuperCollider uGen [145]. I have made my own modification to the attractor. I replaced a sine function with a cosine function to prevent the attractor from getting stuck at 0 values and not regenerating.

TINKERBELL The Tinkerbell map is described by Alligood, Sauer, and Yorke [146].

6.6.5.3 3D Modes

The TYPE knob selects the following attractors on the 3D Block:

LORENZ This attractor was originally developed by Edward Lorenz as a way to model atmospheric convection [147].

ROSSLER This attractor was developed by Otto Rossler as as a simplified alternative to the Lorenz attractor [148].

MODIFIED PICKOVER 3D This attractor is described in the appendix of Clifford Pickover's book *Chaos in Wonderland* [145]. The original equation uses five variables. I have modified it to use only four variables to work with DrawJong and this Block's interface.

6.6.5.4 Design Notes

These Blocks are a continuation of my Masters research on chaotic oscillators. My Masters project was an iOS application called DrawJong. DrawJong used all of the above attractors except for Tinkerbell and Latoocarfian. It was a way to visualize and sonify these chaotic systems by the use of wavetable oscillators. These Blocks eliminate the use of wavetables and simply continuously generate the attractors. However, they have a greatly expanded frequency range and independent output for each dimension. It could be considered to be modally polymorphic, as the FREQ knob covers a massive frequency range. These can be used as a slow stepped/smooth modulation sources or an audio-rate oscillators.

Figure 6.100: Dust Generator Panel

6.6.6 Dust Generator

6.6.6.1 Description

This Block is a re-implementation of SuperCollider's Dust and Dust2 opcodes. Dust generates irregular impulses with random heights.

6.6.6.2 Controls and Terminals

Panel Controls:

- DENSITY: Selects how frequently random impulses will appear. At low densities, it will sound like vinyl crackle or small artifacts. At higher densities, it turns into a white noise source.
- UNI/BI: Selects between unipolar impulses, which are useful for triggers, or bipolar impulses, which are more useful for audio noise. In SuperCollider, Dust is unipolar, while Dust2 is bipolar.

Figure 6.101: Feedback Sine Panel

Outputs:

- Trig: Generates triggers of a uniform amplitude (instead of random amplitude).
- Noise: Dedicated white noise source.

6.6.6.3 Design Notes

This is one of my favorite SuperCollider uGens, as it is a versatile generator that can work as a pure noise source, an irregular trigger generator, or as a pleasant click generator. I wanted to expand on the original Dust algorithm by tapping more outputs. This Block is simultaneously polymorphic, as it creates the standard Dust output along with a constant white noise output. I've also added a switch to change between unipolar or bipolar impulses, making it easier to switch between the Dust and Dust2 behaviors.

6.6.7 Feedback Sine Chaos

6.6.7.1 Description

This Block implements two Supercollider Chaos uGens: FBSineN and FBSineL. A sine wave is fed back on itself using Phase Modulation. Controls are given to modify the index and phase behavior of the sine oscillator directly. This can generate a stable sine wave or chaos.

6.6.7.2 Controls and Terminals

Panel Controls:

- INDEX X: Multiples the sine lookup index. At around 10 o'clock, you can use this to get a perfect sine.
- PHASE X: Multiplies the phase of the lookup section. At full counter-clockwise, the multiplier is x1, giving you a perfect sine.
- FEEDBACK: Controls the amount that the sine feeds back into its phase lookup section. This is an attenuverter, so it can flip the polarity of the feedback.
- PHASE +: Determines how quickly the phase increments or decrements. 12 o'clock freezes the oscillator. Clockwise acts like a frequency control (but can introduce lots of aliasing at high speeds, as the FREQ control effectively sets a sampling rate). Counter-clockwise gives more chaotic behavior.

6.6.7.3 Design Notes

Like many of the chaos and noise generators, this is modally polymorphic. The frequency and interpolation settings mean that this can operate as an audio oscillator,

Figure 6.102: Fitzhugh-Nagumo Panel

an LFO, or a stepped modulation source. The various chaos settings move this between a pure sine tone and an aggressive, chaotic generator.

6.6.8 FitzHugh-Nagumo Chaos

6.6.8.1 Description

This is a chaotic oscillator based on the model of a neuron firing. This Block is based on the FitzHughNagumo uGen for SuperCollider (part of the SLUGens plug-in collection).

It tends to be much more stable than many of the other chaotic oscillators. As such, it works very well as a semi-predictable modulation source.

6.6.8.2 Design Notes

Like the other chaotic generators, this is modally polymorphic. A future design improvement would be to combine this with the 2D Attractors Block. Unlike the 2D

Figure 6.103: Gingerbread Chaos Panel

Attractors Block, this Block has a SEED gate and a frequency range switch. Both of those could be added to the Attractors.

6.6.9 Gingerbread Chaos

6.6.9.1 Description

A re-implementation of SuperCollider's GbmanL and GbmanN uGens. These uGens create a chaotic map based on the "Gingerbreadman" algorithm [149].

This algorithm is heavily dependent on initial parameters. When you first load this Block, it loads standard (non-random) initial conditions. Clicking "SEED" will clear the current memory and reseed the Block with new, random initial conditions. It will occasionally achieve stable oscillation.

6.6.9.2 Controls and Terminals

Panel Controls:

Figure 6.104: Low Frequency Noise Panel

- SELF FM: Sets the amount of feedback used to modulate the FREQ setting.

6.6.9.3 Design Notes

Like the other chaos Blocks, this is a modally polymorphic design. This algorithm was not rolled into the 1- or 2-op Chaos Blocks since it used a different control set. Instead of having a CHAOS control, it has a SELF FM feedback control that did not exist in the original SuperCollider implementation.

6.6.10 Low Frequency Noise

6.6.10.1 Description

This is a switchable noise source designed for extra-slow modulations. It can generate at a steady sampling rate or with random fluctuations.

6.6.10.2 Controls and Terminals

Panel Controls:

- FLUX: Controls the intensity of random frequency fluctuations. At 0, the noise will be generated at a constant rate. As you increase FLUX, this rate will become more random.

Modes:

- White: Classic noise. Very rapid changes and large amplitude jumps.
- LFSR: Low Frequency Shift Register Noise. Also pretty rapid, but has a more "digital" feel at faster frequencies.
- Gray: "Gray Noise" as defined by Supercollider (I've seen multiple definitions of Gray, which is why I've specified SC here). This is an extremely harsh digital distortion that randomly flips bits in a word-length variable. It is characterized by very extreme value jumps.
- Pink: Less harsh than White noise. More gradual amplitude changes. Tends to cluster in regions.
- Brown: "Low-frequency" noise. Very useful as a random modulation source. Extremely slow rate of amplitude change. Closer to a "drunk" random source.

Outputs:

- Stepped: Whenever a new sample is generated, this output holds that value until the next sample is generated.
- Smooth: This output linearly interpolates each sample at a rate equal to the generation rate (i.e. if a new sample is generated every second, it will take one second to reach the new value).

Figure 6.105: Multi-Noise Panel

6.6.10.3 Design Notes

Yes, "low-frequency" is a bit of a misnomer for a noise generator. However, I feel that it is an apt musical description. This is modally polymorphic as the choice of noise sources can vary greatly. It could be consider simultaneously polymorphic, as the Smooth and Stepped outputs have very different purposes. There's an alternative version of this Block called "Spectral Noise". It uses the generators in this Block to populate spectral bins. It creates extremely harsh, digital-sounding noise.

6.6.11 Multi-Noise

6.6.11.1 Description

This is a multi-out simple noise source. There are no controls. All noise sources are generated simultaneously.

6.6.11.2 Controls and Terminals

Outputs:

- White - Classic noise. Fairly bright and harsh.
- Pink - Less harsh than White noise.
- Brownian - “Low-frequency” noise. Very useful as a random modulation source. Artificial gain has been added to make it more audible.
- SC Gray - "Gray Noise" as defined by Supercollider (I’ve seen multiple definitions of Gray, which is why I’ve specified SC here). This is an extremely harsh digital distortion that randomly flips bits in a word-length variable.

6.6.11.3 Design Notes

This is an unusual design for Euro Reakt, and one that I intend to replace in a future update (I plan on combining it with Low Frequency Noise and adding a frequency range switch). This design is based off of the Steady State Fate Quantum Rainbow [150], a Eurorack module that has a number of simultaneous noise outputs with no controls. There isn’t much of a reason to have access to all of the outputs at once in a software environment, so combining it with the LFNoise Block and making it modally polymorphic would be a better all-around design strategy.

6.6.12 Probability Noise

6.6.12.1 Description

This is a binary noise generator. It randomly generates samples with two possible values.

Figure 6.106: Probability Noise Panel

6.6.12.2 Controls and Terminals

Panel Controls:

- BI/UNI: In bipolar mode, the output will be -1.0 or 1.0, which is more useful for audio. In UNI mode, the output will be 1.0 or 0.0. This is useful for generating random gates.
- PROB: This determines the likelihood that the output will be high each cycle. At full clockwise, no minimum value will appear at the output.

6.6.12.3 Design Notes

This is an original design that isn't based on an existing Eurorack module (at least, to my knowledge). This is modally polymorphic, as it works well as both an extreme audible noise generator or a slow, random gate source. In SMOOTH mode, it generates unpredictable triangle modulations.

Figure 6.107: Spectral Noise Panel

6.6.13 Spectral Noise

6.6.13.1 Description

This is a remix of the Low Frequency Noise Block. The noise generator is used to populate spectral bins, resulting in a lot of harsh, digital-sounding noise.

6.6.13.2 Design Notes

See Low Frequency Noise design notes.

6.6.14 Squid Axon

6.6.14.1 Description

A very unusual design from Nonlinear Circuits [151]. It combines a three-voice mixer, an Analog Shift Register, and two kinds of feedback (linear and diode-clipped nonlinear). It is "based on the Hodgkin-Huxley equation describing the chaotic behavior observed in giant squid axons" [152]. Essentially, this is an Analog Shift Register

Figure 6.108: Squid Axon Panel

with an alternative counting method, a mixer, and two types of feedback.

6.6.14.2 Controls and Terminals

Panel Controls:

- LIN: Linear feedback. Feedback is taken from the mix output and is directly added to the mixed input.
- NONLIN: Nonlinear feedback. Feedback is taken from the mix output and is sent through a diode emulation, rectifying the feedback and affecting the amplitude curve.
- GATE: Manual, clickable button that duplicates the functionality of the Gate input.

Outputs:

- Out 1-4: Individual outputs for each stage of the shift register.

Figure 6.109: Triggered Noise Panel

- Gate: Gate output. Outputs the Gate input OR the manual GATE panel button.
- Mix: Simple sum of all three inputs.

6.6.14.3 Design Notes

See Analog Shift Register design notes.

6.6.15 Triggered Noise

6.6.15.1 Description

Manually triggered random source. It only uses CPU when triggered, so it's much more efficient to use this than a Noise Block with a Sample & Hold Block.

6.6.15.2 Controls and Terminals

Modes:

- White: Classic noise. Very rapid changes and large amplitude jumps.
- LFSR: Low Frequency Shift Register Noise. Also pretty rapid, but has a more "digital" feel at faster frequencies.
- Gray: "Gray Noise" as defined by Supercollider (I've seen multiple definitions of Gray, which is why I've specified SC here). This is an extremely harsh digital distortion that randomly flips bits in a word-length variable. It is characterized by very extreme value jumps.
- Pink: Less harsh than White noise. More gradual amplitude changes. Tends to cluster in regions.
- Brown: "Low-frequency" noise. Very useful as a random modulation source. Extremely slow rate of amplitude change. Closer to a "drunk" random source.
- Gauss: Gaussian distribution. This means that most values will cluster toward the center value of 0.0 instead of the extremes of +/- 1.0.

6.6.15.3 Design Notes

This is a monosemous design. Even though it has different "modes", the various modes do not greatly affect the behavior of the Block. It is a quick, efficient replacement for the classic Noise and Sample & Hold Meta-Module. A future update idea would be to add a "Ext. In" input and mode, allowing this to work as a more generic Sample & Hold Block.

Figure 6.110: Tuned Noise Panel

6.6.16 Tuned Noise

6.6.16.1 Description

This is a filtered noise source with extra resonance. A white noise generator runs through a tuned 4-Pole Ladder Filter.

6.6.16.2 Controls and Terminals

Panel Controls:

- RES: Controls the resonance of the filter.
- NOISE: Controls the gain of the white noise to the filter.
- LP/HP: Crossfade between low-pass and high-pass filter responses on the main output.

Outputs:

- White: Dedicated white noise output, unaffected by the OUT gain control.

- LP/HP/BP: Dedicated high-pass, low-pass, and band-pass outputs, affected by the OUT gain control.

6.6.16.3 Design Notes

This is very similar to Twin Peaks, also a simultaneously polymorphic design. Twin Peaks is based on a mode from Braids that emphasizes tonality more than noise, as the only available filters are two highly resonant band-pass filters. This Block, meanwhile, emphasizes the shaping of noise through the use of a multimode filter. The noise is available independently of the various filtered outputs, making this simultaneously polymorphic.

6.7 Samplers

This category consists of two Blocks dedicated to sample playback and manipulation.

These two Blocks share the following controls:

Panel Controls:

- OUT: Controls the output level.
- START/END: These two controls determine the start and end points for sample playback.
- AC/DC: Enables or disables DC filtering. DC filtering is useful for removing low-frequency artifacts that appear when using extremely slow playback rates. Turning off DC filtering is useful if you want to play a recording of a modulation waveform.
- SMOOTH/LO-FI: "Smooth" turns on cubic interpolation for sample playback.

Figure 6.111: Stereo Sample Looper Panel

The difference in interpolation strategies is very apparent when playing back samples at lower rates than 1x.

6.7.1 Stereo Sample Looper

6.7.1.1 Description

This Block allows you to load a mono or stereo audio file and play it back in a looped manner. The START and END points of the loop can be modulated, along with the SPEED of playback.

The sampler will output its PHASE so that you have a synced modulation source (It ramps from 0-1, 0 being the START point and 1 being the END point). The sampler will also output a trigger when the loop reaches its END point. You can use this to trigger other sounds or sequences in sync with your loop.

6.7.1.2 Controls and Terminals

Panel Controls:

- SPEED: Controls the speed of the internal phasor, in effect controlling the playback rate of the sample. 12 o'clock is no playback, and anything CCW from there is reverse playback.
- X1/2: Controls the maximum speed of playback for the SPEED knob.
- FREE/KEYBD: In FREE mode, the playback speed follows the SPEED knob. In KEYBD mode, the playback speed is determined by the signal present at the Pitch input, with C3 being 1x playback, C4 being 2x, C2 being 0.5x, etc.
- RUN/STOP: Enable or disable playback.
- FM: Frequency modulation depth for the internal phasor.
- RESET: Immediately restarts the sample at its start index.
- FREEZE: Pauses the internal phasor, holding the sample playback at its current value.

There are two modes for how the START and END controls behave:

- CLEAN - After adjusting the Start and End points, the new points will take effect after the current loop completes or a manual reset is triggered.
- SMEAR - After adjusting the Start and End points, the internal oscillator immediately adjusts its speed, causing all sorts of strange sounds.

Inputs:

- Pitch: When using KEYBD mode, the signal present here will control the playback speed of the sample.
- FM: FM input for modulating the speed of the read head.
- Reset: A positive gate here will reset the read head to the START point.
- Freeze: A positive gate here will hold the read head in its current position.

Outputs:

- End Trig: Whenever the sample resets, a trigger will appear here. The red lamp on the panel indicates the state of this output.
- Phase: This is a unipolar output with a sawtooth wave corresponding to the current position of the read head. The phase is given as the phase between START and END, not the absolute start and end points of the sample.

6.7.1.3 Design Notes

This is a simultaneously polymorphic design. In addition to playing a sample, this will also produce a timing trigger along with a phase signal. Both of these outputs are useful for syncing other patch elements to the sample.

6.7.2 Stereo Sample Scanner

6.7.2.1 Description

This Block allows you to load a mono or stereo audio file and scan through it using another waveform. This has been calibrated to work best with the Standard Library's Bento Box LFO.

Figure 6.112: Stereo Sample Scanner Panel

An ascending Ramp wave will give you your sample played back forwards. A descending Sawtooth wave will play your sample backwards. A Triangle will play the sample forwards then backwards. Experiment with other waveforms to find very unusual sounds.

6.7.2.2 Controls and Terminals

Panel Controls:

- IN: Controls the amplitude of the signal used to address the sample's table data.
- X1/2: Used to amplify both the input scanner signal and the output data.

Three modes of rectification are available. Any signal can be used as an input, but only unipolar values in the range [0.0, 1.0] can be used to scan through the sample.

- UNI - Converts a bipolar signal into a unipolar signal. This adds 1.0 to the signal and then halves the amplitude.

- HALF - Negative component of signal is silenced.
- FULL - Negative component of signal is flipped (Takes absolute value of signal).

Inputs:

- Scan: The signal present here is used to address the sample's table data.

6.7.2.3 Design Notes

Unlike the Sample Looper, this is a monosemous design. It is a monosemous design that is almost identical to the Wavetable Distortion Block, except in this Block the user manually defines the table by loading a sample. There is a lot of complexity present in the control set, but ultimately it is an effect with a single mono input and stereo output.

6.8 Sequencing and Logic

This category of Blocks is primarily used to generate or process gates, triggers, and other timing signals. Some of these Blocks are capable of generating stepped modulation sequences as well.

Many of these Blocks share the following controls, which will not be detailed in each Block's description unless notable:

Panel Controls:

- GATE: This is a clickable button on the front panel that matches the functionality of the Block's Gate input. The Gate is high for as long as the user's mouse click is held down.

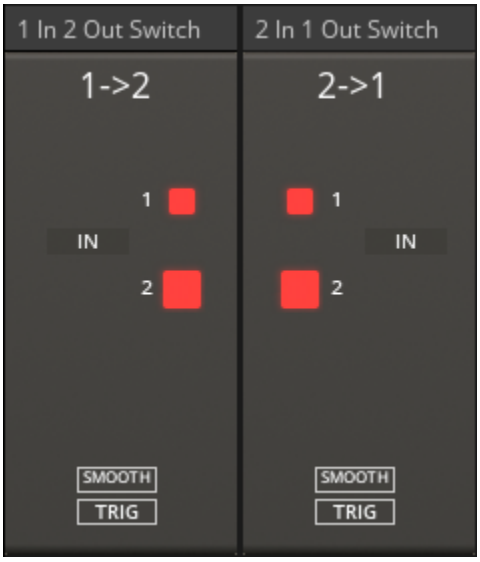

Figure 6.113: Simple Switch Panels

- RESET: This is a clickable button on the front panel that matches the functionality of the Block's Reset input. The Reset message is triggered immediately on mouse down.

- Indicators: These are red squares that show the user the status of the Block's gate inputs and/or outputs. The user can assign new colors to the indicators by clicking and dragging on them.

Inputs

- Gate: This is the primary timing input for most sequencing Blocks. Typically, a true/false signal is used here to advance the sequencer Blocks.

- Reset: On sequencing Blocks with multiple stages/steps, a positive signal received here will reset the sequencer back to its first step.

6.8.1 1->2 and 2->1 Switches

6.8.1.1 Description

These two Blocks provide two simple switching strategies. The 1 In 2 Out Switch takes one input and toggles between two outputs for it. It can be thought of as a binary panner. Likewise, the 2 In 1 Out Switch takes two inputs and toggles which one appears at the output. In can be thought of as a binary crossfader.

6.8.1.2 Controls and Terminals

Panel Controls:

- IN: Manual, clickable button that behaves the same as the Gate input.
- SMOOTH/INSTANT: Determines how quickly the switch changes states. In SMOOTH mode, the switching is smoothly interpolated. This is useful for switching audible signals, as it eliminates clicks and discontinuities. INSTANT mode toggles the switch instantaneously. This is more useful for sequencing and timing signals, where instant changes are critical for proper patch behavior.
- TRIG/GATE: In TRIG mode, the active channel will change and hold on the reception of a positive gate signal. That channel will remain active until a new, separate, positive gate signal is received. In GATE mode, the second channel is selected if (and only if) the gate input is high. In other words, TRIG mode acts like a flip-flop (toggle) switch, while GATE mode behaves like a momentary switch.

Outputs:

- Gate 1/2: Outputs a gate that is true when the corresponding stage is active.

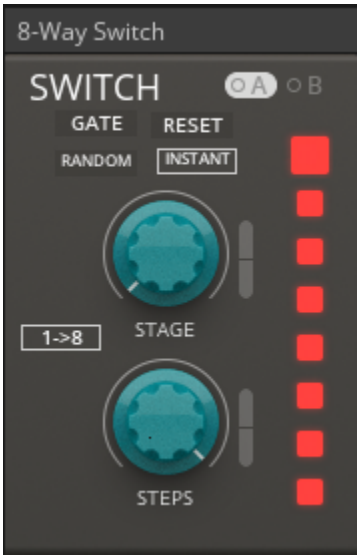

Figure 6.114: 8-Way Switch Panel

6.8.1.3 Design Notes

See 8-Way Switch design notes.

6.8.2 8-Way Switch

6.8.2.1 Description

This is a modulation-addressable 8-way switch. It will take in 1 input, and send it to 1 of 8 outputs. It will take in 8 inputs, and select one for the MIX output. It will also take 8 inputs, and determine which one is active at its respective output.

The "STAGE" knob will select the currently active in/output, while "STEPS" determines the length of the sequence.

This Block is a swiss-army knife of utility. You can use it for creating gate sequences, routing audio or modulations, and more. When creating snapshots, you can use the STAGE setting to store a static routing.

6.8.2.2 Controls and Terminals

Panel Controls:

- STAGE: Manually select the currently active output. If STAGE is higher than STEPS, STEPS will be selected. Changing the STAGE knob automatically triggers a reset.
- STEPS: Determines the highest active output. If set to 6, for example, Outputs 7 and 8 will be skipped.
- RANDOM: Selects a random gate (within range) to activate.
- INSTANT/SMOOTH: Determines how quickly the switch changes states. In SMOOTH mode, the switching is smoothly interpolated. This is useful for switching audible signals, as it eliminates clicks and discontinuities. INSTANT mode toggles the switch instantaneously. This is more useful for sequencing and timing signals, where instant changes are critical for proper patch behavior.
- 1->8/8 IN: In 1->8 mode, In 1 will be used for inputs 2-8 as well. If you want to do 1->8 switching, use this, and then monitor OUT 1-8. In 8 IN mode, all 8 inputs are independent. If you want to do 8->1 switching, use this mode, and monitor the MIX output.

Inputs:

- GATE: Advances the counter by one.
- RESET: Resets the counter to "STAGE" or "STEPS," whichever is lower.
- RANDOM: Chooses a stage to activate at random (within range).

Outputs:

- OUT1-8: Switched outputs
- G1-G8: Gate outputs for individual steps. Remain high while the step is selected.
- Trigs: Generates an impulse whenever a new output is selected.
- Reset: Generates an impulse whenever a reset event occurs (whether from the panel, externally, or from the counter reaching max).

6.8.2.3 Design Notes

This is one of the most complex designs in Euro Reakt, and it is highly polymorphic. One of my favorite uses for it is as a variable patch manager for Reaktor. In Reaktor, the routing between Blocks is static within an ensemble and cannot be changed from preset to preset. Since this can route 1 input to 8 outputs, 8 inputs to 1 output, or 8 inputs to 8 normally-closed outputs, this Block can act as a per-preset patch manager.

It also acts as the foundation for a step sequencer. The outputs can be addressed sequentially (or randomly) via gates, and the dedicated gate outputs can then be used to trigger other sources. In fact, this Block is the foundation for the Voltage Controlled Gates Block, which is almost identical in design but ultimately less flexible. The VC Gate Block is not polymorphic as it only acts as a sequential gate Block. Still, it is useful when only the counting abilities of the 8-Way Switch are needed.

Another non-polymorphic, rhizomatic offshoot is the Random Gates Block, which has eight gates but no sequential counter. The gate selection is only random.

This Block is much more useful than the unidirectional 2->1 and 1->2 Blocks,

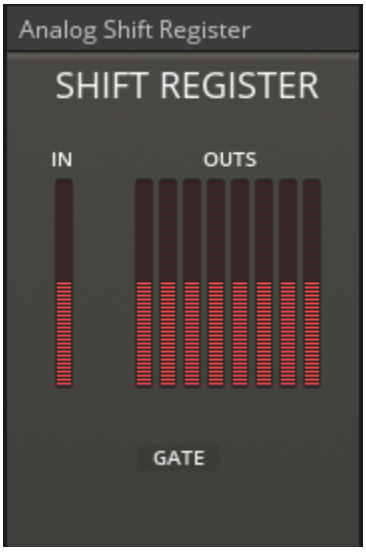

Figure 6.115: Analog Shift Register Panel

which do not save their states with presets. I've only kept those Blocks in Euro Reakt due to their lower CPU requirements. They are intended for patches which need simple binary switching. Despite their simplicity, the 2->1 and 1->2 Blocks are simultaneously polymorphic as they also generate gates corresponding to the active stage.

6.8.3 Analog Shift Register

6.8.3.1 Description

This is an eight-stage clocked memory device, typically used to create rounds or canons.

Essentially, OUT 1 acts like a simple Sample-and-Hold. Whenever the ASR receives a positive GATE, the state of IN will be frozen and sent to OUT 1.

After another GATE, the previous state of OUT 1 will be passed to OUT 2, and OUT 1 will acquire a new input. For each successive gate, the previous voltage will

"shift" by one output.

6.8.3.2 Controls and Terminals

Panel Controls:

- GATE: Manually trigger the Gate input from the panel.

6.8.3.3 Design Notes

This is a rhizomatic design, as it is intended to connect to many other modules. There aren't any improvements upon a typical ASR design. A more unusual ASR design is the Squid Axon Block, based on a Eurorack design by Nonlinear Circuits [151]. It features a three channel input mixer and two feedback paths. Even then, the Squid Axon is not polymorphic in its hardware configuration. I created simultaneous polymorphism in the Block version by adding three input attenuverters and an additional Mix output.

One future design consideration is that the Analog Shift Register and Squid Axon use different counting methods. The ASR uses a more traditional cascading method where each stage immediately passes its value to the next. In the Squid Axon, all four stages receive the same value one-by-one before the first stage receives a new value. It would be worth adding a switch to change the counting method on Squid Axon.

6.8.4 Analog-to-Digital/Digital-to-Analog Converters

6.8.4.1 Description

These two Blocks are used to convert signals to and from 8-bit representations.

Figure 6.116: ADC and DAC Panels

6.8.4.2 Controls and Terminals

Panel Controls:

- OFFSET - Adds a DC offset to the signal. In A/D conversion, this happens post-scaling. In D/A conversion, this happens pre-scaling.

Encoding/Decoding MODEs:

- UNI8 - 8-bit unsigned representation. Expects a unipolar input signal of 0-1, but don't let that stop you from using bipolar signals!
- BI OFF - Scales and offsets a +/- 1.0 signal to 0-1 before using the UNI8 encoder. Naturally, detail is lost.
- BI SIG - First 7 bits are used to represent your signal. The 8th bit carries the sign of the signal (positive or negative)

Three modes of RECTification.

Figure 6.117: Binary Gate Panel

- NONE - No rectification. Normal signal.
- HALF - Negative component of signal is silenced.
- FULL - Negative component of signal is flipped (Takes absolute value of signal).

6.8.4.3 Design Notes

These were inspired by the ADC and DAC modules in the Nord Modular [97]. These are interesting designs to analyze with the taxonomy! I would say that the ADC is modally polymorphic, as BI SIG mode can be used to find the sign of an incoming signal and output it on the eighth bit. The DAC is rhizomatic, as it can connect to up to eight different Blocks. However, its function never changes.

6.8.5 Binary Gate

6.8.5.1 Description

This Block is an original idea that I haven't seen in hardware before. It is a boolean gate with separate triggers for activating the gate's on and off states.

6.8.5.2 Controls and Terminals

Panel Controls:

- ON: Sets the gate output to 1.0.
- OFF: Sets the gate output to 0.0 or -1.0, depending on the mode.
- UNI/BI: In UNI mode, the gate is 0.0 when off. In BI mode, the gate is -1.0 when off.

Inputs:

- Gate On: Sets the gate output to 1.0 upon reception of a positive signal.
- Gate Off: Sets the gate output to 0.0 or -1.0 upon reception of a positive signal.

6.8.5.3 Design Notes

This is a monosemous design that I have wanted in a hardware module. I've discovered that this is known as a Set-Reset Flip Flop (S-Dominated type, as the ON/Set gate takes precedence over the OFF/Reset gate). In a future update, I will move this Block to Legacy and roll it into the main Flip Flop Block.

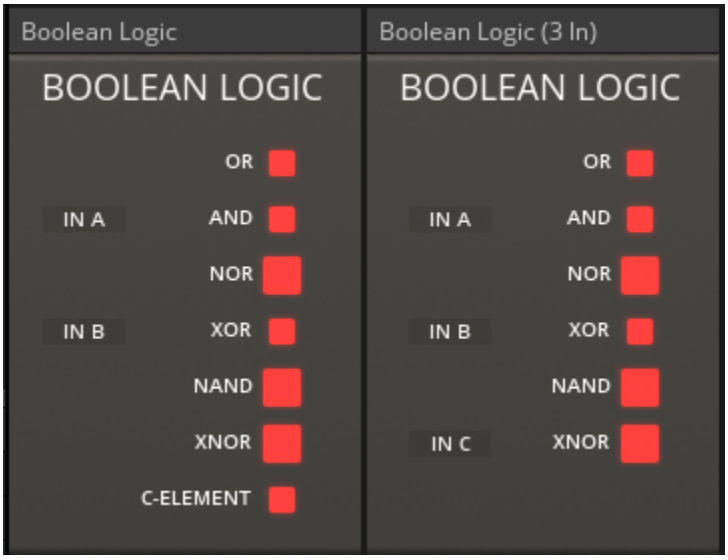

Figure 6.118: Boolean Logic Panels

6.8.6 Boolean Logic (2 or 3 Input)

6.8.6.1 Description

These Blocks take in two or three inputs and produce multiple outputs based on logical boolean operations. The inputs are considered to be true if they are above 0.5.

6.8.6.2 Controls and Terminals

Panel Controls:

- IN A/B/C: Manual, clickable gate controls for each input.

Modes:

- OR: Produces a gate when either input is high.
- AND: Produces a gate when both inputs are high.
- NOR: Produces a gate when neither input is high.

- XOR: Produces a gate when only one input is high.
- NAND: Produces a gate as long as both gates aren't high simultaneously.
- XNOR: Produces a gate when either both gates are low or both gates are high.
- CELE (2-Input Block Only): This is the only included logic operation that uses history. This is a Muller C-Element output. It is true when both elements are true, and false when both elements are false. However, when only one input element is true, it will hold its previous state.
- NOT A/B/C: Outputs that are always the logical inverse of their associated inputs.

6.8.6.3 Design Notes

This is a simultaneously polymorphic design where 2-3 inputs provide many different output behaviors, each of which has its own output. The current designs could be improved in a number of ways. First, instead of having two Blocks, it would be better to have one Block with the ability to switch between 2 and 3 input behaviors. Second, it could have a modally polymorphic output with a modulated MODE switch. This would replace the need to combine this Block with the 8-Way Switch Block if a user wants to use multiple types of logic. Finally, I think it would be an improvement to move the C-ELEMENT output to the Flip Flop Block instead, as that section has behavior that is more similar to the various Flip Flop modes.

Figure 6.119: Burst Generator Panel

6.8.7 Burst Generator

6.8.7.1 Description

This is a useful tool that takes in one trigger and turns it into many evenly-spaced 1 ms triggers. The number of triggers is determined by the REPEATS knob.

6.8.7.2 Controls and Terminals

Panel Controls:

- SPACE: Determines the length of time between each trigger in the burst.
- REPEATS: Determines the number of triggers that appear inside of the burst.
- x1/x10: Increase the range of the SPACE or REPEATS knob tenfold.
- GATE: Manually trigger a burst from the panel.
- KILL: Manually stop a burst immediately from the panel.

Outputs:

- Out: The initial trigger followed by all repeats.
- Burst: The repeats only.
- First: The initial trigger only.
- Last: The last repeat only.
- Phase: Unipolar envelope output. Outputs the phase between the last impulse and the next.

6.8.7.3 Design Notes

This is a great rhizomatic and simultaneously polymorphic design. There are a lot of variations on the trigger outputs, meaning that this can provide a lot of variation within a patch. The phase output adds polymorphism. It may seem like a strange feature, but it's perfect for creating bursts of grains. The phase can be used to create a grain envelope.

6.8.8 Comparator

6.8.8.1 Description

A comparator listens to a signal (audio or control), and determines whether the signal crosses a threshold (set by the COMPARE knob). If the signal exceeds this threshold, the comparator outputs a positive signal. Otherwise, the comparator outputs 0.0 (if set to UNI mode) or -1.0 (if set to BI mode).

Figure 6.120: Comparator Panel

6.8.8.2 Controls and Terminals

Panel Controls:

- COMP: Sets the threshold to compare the input signal against. At 12 o'clock, the threshold is 0.0.
- UNI/BI: In UNI mode, the gate outputs are 0.0 when low. In BI mode, the gate outputs are -1.0 when low.

Outputs:

- > Out: Gate output, true when the input is greater than the COMP threshold.
- > Trig: Trigger output, fired when the input is greater than the COMP threshold.
- < Out: Gate output, true when the input is less than the COMP threshold.
- < Trig: Trigger output, fired when the input is less than the COMP threshold.

Figure 6.121: Delta Panel

- Crossing: Trigger output, fired when the input crosses the threshold in either direction.

6.8.8.3 Design Notes

This is a flexible, audio-rate comparator with simultaneous polymorphism. The Crossing output adds polymorphism, as the comparator can simultaneously act as a sign detector and sample-accurate zero-crossing detector.

6.8.9 Delta

6.8.9.1 Description

This Block outputs a voltage based on the rate of change of its input. For instance, if the input is a triangle waveform, Delta's output will be positive while the triangle is rising, and negative while the triangle is falling.

The amplitude of the Delta output will be highly dependent on the frequency of

the input signal. Because of this, a "D-Boost" knob controls a massive gain range (over 8000x amplitude). The Delta output is hard-clipped, so it won't go louder than +/- 1.0.

The Delta output also runs through a bi-directional comparator. The comparator will output a gate and a trigger based on the current direction. The triggers, for instance, can be useful for synchronizing timing signals to a sawtooth wave. When the sawtooth wave resets, a directional trigger will fire.

6.8.9.2 Controls and Terminals

Panel Controls:

- D-BOOST: Delta Boost. Multiplies the Delta output by up to 8000x.
- UNI/BI: In UNI mode, the gate outputs are 0.0 when low. In BI mode, the gate outputs are -1.0 when low.

Outputs:

- > Out: Gate output, true when the delta is positive.
- > Trig: Trigger output, fired when the delta transitions to positive.
- < Out: Gate output, true when the delta is negative.
- < Trig: Trigger output, fired when the delta transitions to negative.
- Dir. Change: Trigger output, fired when the signal changes direction.

6.8.9.3 Design Notes

This Block idea came about while prototyping a Sandman Pro mode in Max and realizing the importance of the "delta" object in many situations. The Max delta

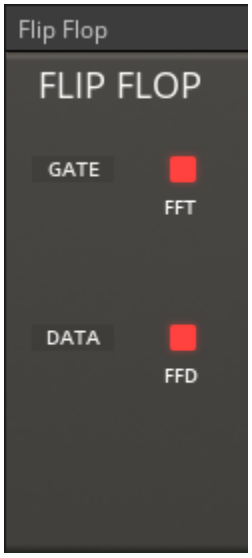

Figure 6.122: Flip Flop Panel

object is monosemous, simply taking in an input and outputting the rate of change per sample. For this Block, I created simultaneous polymorphism by adding separate gate and trigger outputs for positive and negative signals, along with a trigger output that fires whenever the signal changes direction.

6.8.10 Flip Flop

6.8.10.1 Description

A multi-purpose clock and gate utility. Flip-Flops are frequently combined with Boolean Logic gates to create complex, generative patterns and events. This Block contains both a Flip-Flop T and a Flip-Flop D.

6.8.10.2 Controls and Terminals

There are two inputs, GATE and DATA. GATE is important for both Flip-Flops. DATA is important only for the bottom Flip-Flop (D).

FFT acts like a simple toggle switch. Every time it receives a clock, it flips its current state (on or off). If it receives a steady clock, it can be thought of as a simple clock divider. It has many other excellent use cases, though! For instance, it can react to a button or key press and act like a latch (so that a user does not need to continue holding a key press).

FFD acts like a true-or-false Sample and Hold. Whenever the GATE input goes high, the current state of the DATA input will be written to FF2. It will continue holding this state until another GATE is received.

6.8.10.3 Design Notes

Like the Boolean Logic Blocks, this is a simultanously polymorphic design where two inputs interact with different output systems. This Block is due for an upgrade, as there are many more types of Flip-Flop circuits that can be emulated here. First, the Binary Gate Block is an SR NOR Latch Flip-Flop, so it could be eliminated and combined with this Block. Second, the C-ELEMENT output on the Boolean Logic (2-input) Block is closer in behavior to a Flip-Flop, meaning that it would fit better here. Finally, the JK Flip-Flop type would be a useful addition. In addition to these, it could be advantageous to add a modally polymorphic output with a MODE control, allowing a user to use one output and experiment with the various behaviors.

6.8.11 Gate Combiner

6.8.11.1 Description

This Block adds up to 8 gates together to form one gate, which is present at the OR output. The NOR output is always the inverse of the OR output. Finally, the TRIGS output takes all input gates and converts them to triggers of .001 seconds in

Figure 6.123: Gate Combiner Panel

length.

6.8.11.2 Controls and Terminals

Outputs:

- OR: True when any of the gate inputs are true.
- NOR: True when none of the gate inputs are true.
- Trigs: Outputs a trigger when any of the inputs become true.

6.8.11.3 Design Notes

This is a simple, useful design that exhibits simultaneous polymorphism. It has two logic outputs that are useful for determining whether there is an active timing event between many streams. It is also used for extracting a single trigger stream from the union of multiple timing sources.

Figure 6.124: Gate Delay Panel

6.8.12 Gate Delay

6.8.12.1 Description

This is a very useful gate and trigger manipulation tool. This receives a gate or trigger as input, and outputs a variable length gate after a specified time delay. For instance, it could receive a trigger, wait half a second, and then output a .25 second gate.

At low delay times, you could use this for things like drum flams. At higher delay times, you can program ghost notes and sub-rhythms.

6.8.12.2 Controls and Terminals

Panel Controls:

- DELAY: Sets the amount of time it takes for the delayed gate to occur. This is from 0 to 1 second. With the x10 switch enabled, this is from 0 to 10 seconds.
- WIDTH: Sets the width of the delayed gate. This is from .001 to 1 second.

With the x10 switch enabled, this is from .001 to 10 seconds (Note that the bottom limit does not change).

Outputs:

- Out: Variable-width delayed gate
- Del. Trig: Delayed trigger (constant .001 second width)
- In Trig: Input signal, converted to a .001 second width trigger.
- In + Del.: Input and delayed triggers combined.

6.8.12.3 Design Notes

This is a simultaneously polymorphic design. It acts as a standard gate delay and a gate-to-trigger converter. One of my favorite Meta-Modules is to combine this with the Burst Generator. The Burst Generator creates a series of events before triggering the Gate Delay on its last event. The Gate Delay determines a length of silence that occurs before triggering the Burst Generator again.

6.8.13 Gate Matrix

6.8.13.1 Description

This complicated gate processor is based on Numberwang by Nonlinear Circuits [153].

It takes in four gates and outputs 16 multiplexed variations. The ONE mode emulates the Numberwang by only outputting one gate at a time. The separate ALL mode can trigger up to fifteen simultaneous outputs, based on the inputs.

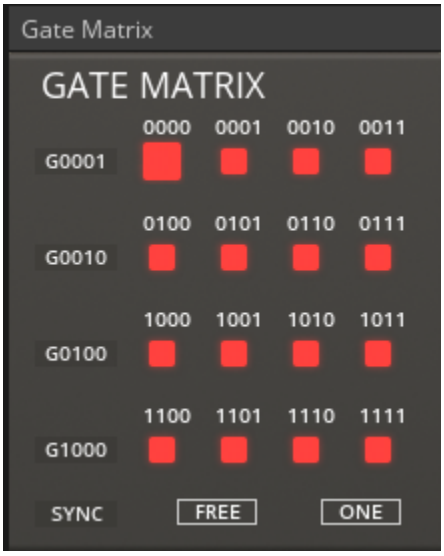

Figure 6.125: Gate Matrix Panel

Essentially, the Block acts as a 4-bit decoder where the input gates are the bits and the output gates are integers 0-15. An easy way to look at it is to add the input gates' binary values.

For instance, if gates 0001 and 0100 are triggered, output gate 0101 is true.

6.8.13.2 Controls and Terminals

Panel Controls:

- FREE/SYNC: In FREE mode, output gates go high immediately in response to incoming gates. In SYNC mode, output gates only go high if the Sync gate is high.
- SYNC: Manual trigger for the Sync input. When SYNC mode is active, the gate outputs will only be active when the Sync gate is high.
- ONE/ALL: In ONE mode, the Block acts as a 4-bit decoder and only activates one output gate at a time. In ALL mode, the output rule is that an output is

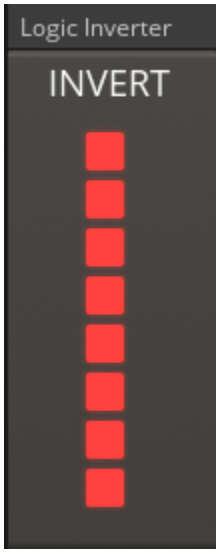

Figure 6.126: Logic Inverter Panel

true if (output OR input) is true. For instance, if the input bits are 0101, then outputs 0101, 0100, and 0001 will be true.

Outputs:

- 4-Bit: This output acts as a simple 4-bit DAC. It generates a stepped value based on the state of the input bits.

6.8.13.3 Design Notes

The original Numberwang is a wild, rhizomatic design that combines a large number of inputs and outputs. However, the behavior of the module is fairly static. I've expanded upon the original design by adding the ONE/ALL mode switch and the 4-Bit output. The 4-Bit output adds simultaneous polymorphism as the Block generates both gates and a stepped voltage based on its inputs.

6.8.14 Logic Inverter

6.8.14.1 Description

This is a simple logic-inverting circuit, multiplied 8 times. This simply flips any true or false signal (gates, triggers, logic, bits, etc.). The inverse of 1 is 0, and the inverse of 0 is 1. This cannot be achieved with a polarizing VCA/attenuator, as the inverse of 1 is -1 (and 0 is 0) when done that way.

6.8.14.2 Design Notes

This is a simple design with eight copies of a monosemous circuit with no additional functionality or interaction between channels. This Block is intended as a quick, cheap problem solver. If a user wants inversion and complexity, both Boolean Logic Blocks provide inverted outputs of each input channel along with their various logical interactions.

6.8.15 Probability Gates

6.8.15.1 Description

This Block takes in one gate or trigger and passes it to up to 8 outputs with per-output probability.

6.8.15.2 Controls and Terminals

Panel Controls:

- PR. 1-8 (Probability 1-8): Determines the probability that the assigned output will be true on the reception of an incoming gate. At full clockwise, the output will always go true. At full counter-clockwise, the output will never go true.

Figure 6.127: Probability Gates Panel

- GATE/TRIG: In GATE mode, the active true outputs will be true for as long as the input gate is true. In TRIG mode, each active true output will only be active for the length of a .001 second trigger.

6.8.15.3 Design Notes

See Probability design notes below.

6.8.16 Probability

6.8.16.1 Description

This Block takes in one gate and routes it to one of two outputs, depending on probability determined by the PROB knob.

6.8.16.2 Controls and Terminals

Panel Controls:

Figure 6.128: Probability Panel

- PROB: Determines the probability that an incoming gate will be routed to OUT 2. Full counter-clockwise means that OUT 1 will always be active, while full clockwise means that OUT 2 will always be active. In the middle, the GATE will have about an equal chance of going to either output (Never both simultaneously). Please be aware that 50/50 does not mean that the gates will alternate! It simply means that they will *roughly* be active for equal amounts of time.

- GATE/HOLD: In GATE mode, the length of time that OUT 1 and OUT 2 will be held high is the same as the length of the incoming gate. In HOLD mode, each output is held high until the other goes high.

Outputs:

- OUT 1/2: Gate output with behavior determined by GATE/HOLD.

- Trig 1/2: Trigger outputs, fired whenever the corresponding stage goes active (if the same stage is selected twice in a row, two triggers will be fired). These

Figure 6.129: Random Gate Panel

outputs are not affected by the GATE/HOLD switch.

6.8.16.3 Design Notes

This Block was inspired by Branches by Mutable Instruments [154]. It is a very useful Block for adding chance to compositions by breaking up steady timing streams. This is a rhizomatic design as it has four outputs that all derive from the same basic algorithm. A more rhizomatic design is available in the Probability Gates Block, which breaks one gate into eight outputs with per-output probability. By combining Probability Gates with the Logic Inverter, you can create a Meta-Module equivalent to eight Probability Blocks (minus Trigger outputs).

6.8.17 Random Gates

6.8.17.1 Description

This Block takes in one gate or trigger, and passes it randomly to one of eight outputs.

6.8.17.2 Controls and Terminals

Panel Controls:

- MIN: Determines the minimum gate stage that will be selected.
- MAX: Determines the maximum gate stage that will be selected.

6.8.17.3 Design Notes

See 8-Way Switch design notes.

6.8.18 Rungler

6.8.18.1 Description

This Block is based on Rob Hordijk's "Rungler" circuit [136]. A Rungler is based on a Shift Register (see also: Turing Machine and Analog Shift Register Blocks). One input is the GATE, which steps the Shift Register forward. The other input is DATA, which determines whether the current Bit will be high or low.

The last three bits of the Shift Register (Bits 6, 7, and 8, where 8 is the most significant bit) are then run through a digital-to-analog converter. This produces a random, stepped, unipolar signal at the OUT output. This signal has many uses, namely random modulation.

For a complete Rungler circuit, check out the Rungler Oscillator Block.

Figure 6.130: Rungler Panel

6.8.18.2 Controls and Terminals

Panel Controls:

- WRITE/LOOP: In WRITE mode, the contents of the shift register are affected by the signal present on the Data input. In LOOP mode, the register is locked. This creates a looping sequence.
- IN COMP: Sets a threshold for the Data input. When the signal present at the Data input exceeds this threshold, the active bit will be set to true.
- SCALE: Sets the amplitude and polarity of the main output.

Outputs

- B 1-8: Gate outputs. The state of the gate is equal to the state of each bit in the register.

Figure 6.131: Turing Machine Panel

6.8.18.3 Design Notes

See Turing Machine design notes.

6.8.19 Turing Machine

6.8.19.1 Description

This is a Block based on Music Thing Modular's excellent Turing Machine [155]. The Turing Machine is an open-source DIY module that produces random sequences. It uses a shift register to store 8 binary bits, which are converted into an analog voltage.

The red squares on the right side indicate the state of the 8 internal bits. Bit 1 (at the top) is the least significant bit, meaning it barely affects the sequence at all. Bit 8 (the bottom bit) is the most significant bit, meaning that it has a huge influence on the sequence. The bits in between are in increasing order of significance.

Essentially, whenever a bit is true (indicated by its square being larger), it will increase the amplitude of the output sequence. When all bits are false, the sequence

output is simply "0". The sequence output is unipolar, so it goes from a minimum of 0 (all bits false) to 1.0 (all bits true).

Whenever a gate is received, the shift register advances. The state of BIT 1 is passed to BIT 2, BIT 2 is passed to BIT 3 (and so on). Finally, BIT 8 is passed back to BIT 1 via feedback. The PROB knob changes the probability that this bit will flip.

When positive, WRITE 0 forces BIT 1 to be 0 (False) on the next incoming gate. WRITE 1 forces BIT 1 to be 1 (True) on the next incoming gate.

All 8 bits have their own G OUT, turning each bit into a gate. This replicates the functionality of the Turing Machine's PULSES expander.

6.8.19.2 Controls and Terminals

Panel Controls:

- WRITE0/1: When clicked this will write the chosen value to the currently active stage.
- GATE: When clicked, this will immediately advance the shift register by one step.
- PROB: Changes the probability that the currently active bit will flip. It is functionally identical to the hardware Turing Machine's knob: At full-clockwise (100%), the bits will *never flip*, meaning that a looping 8-step sequence will appear at the output. At full-counter-clockwise (-100%), the bit will *always flip*, meaning that a looping 16-step sequence will appear at the output. At center-detent (0%, or 12 o'clock), the sequence will be as random as possible.

- SCALE: Changes the amplitudes of the sequences on the Main and Alt Out outputs.

Inputs:

- Write 0/1: A positive signal here will write the chosen value to the active bit.

Outputs:

- Out: Main output with 256 possible voltages. On this output, each bit has different significance.
- Alt Out: Alternative output with 9 possible voltages. On this output, each bit has equal significance.
- G 1-8: Gate outputs. The state of the gate is equal to the state of each bit in the register.

6.8.19.3 Design Notes

The Turing Machine and Rungler are sister modules with similar functionality. They are random sequencers built upon the interaction of a shift register with a DAC. The Turing Machine uses all eight bits of a shift register along with a simple feedback circuit to provide sequencing behavior without the use of other Blocks (aside from a clock). The Rungler, meanwhile, requires both a clock and a data source. It uses only three of the eight bits in the shift register to produce its random output.

For simultaneous polymorphism, all 8 bits of the shift registers are available on both Blocks. This means that you can create your own path for converting the bits to a voltage (primarily by using the Digital-To-Analog Block, but any mixer can be used to assign value to each bit). Alternatively, if you clock either Block with

Figure 6.132: Voltage Controlled Gates Panel

slower signals, you can use them as semi-random 8-step gate sequencers. At high frequencies, you can use each bit output as a random unipolar square wave noise source.

6.8.20 Voltage Controlled Gates

6.8.20.1 Description

This is a modulation-addressable step sequencer, based on the "Dig. Out" section of the Doepfer A-152 [156]. It provides a bank of eight exclusive, sequential gates. The sequence can be advanced via gates or scanned via modulation.

6.8.20.2 Controls and Terminals

Panel Controls:

- GATE - Manual Trigger and indicator.

- RESET - Manual Trigger and indicator. Resets clock to "STAGE" or "STEPS," whichever is lower.
- STAGE - Manually select the currently active gate. If STAGE is higher than STEPS, STEPS will be selected. Changing the STAGE knob automatically triggers a reset.
- STEPS - Determines the highest active gate. If set to 6, for example, Gates 7 and 8 will be skipped.
- M Column - Mutes the selected gate. This gate will not be skipped. Rather, its G output will not go high, nor will it contribute to the "Trigs" output.
- Indicators - Largest square indicates the currently active step.

Inputs:

- GATE: Advances the counter by one.
- RESET: Resets the counter to "STAGE" or "STEPS," whichever is lower.

Outputs:

- G1-G8: Gate outputs for individual steps. Remain high while the step is selected.
- Trigs: Generates an impulse whenever a non-muted step is selected.
- Reset- Generates an impulse whenever a reset event occurs (whether from the panel, externally, or from the counter reaching max).

6.8.20.3 Design Notes

See 8-Way Switch design notes.

Figure 6.133: Voltage Storage Panel

6.8.21 Voltage Storage

6.8.21.1 Description

This Block allows for the manual selection of voltages. The voltages are organized into three rows, each with eight voltages. The GATE row at the bottom shows what stage is currently active. The knobs above the indicated stage determine the level present at that row's output (OUTS 1-3).

A conceptual analog is a sequencer without a built-in clock. To advance the sequencer, a user must select the stage manually (either via the panel GATE buttons or the GATE 1-8 inputs). A RANDOM button and input will select a random stage.

6.8.21.2 Controls and Terminals

Panel Controls:

- STAGE: This knob will select the currently active stage. This can be modulated, so a unipolar phasor will produce behavior similar to a traditional step

sequencer.

- RANDOM: When clicked, a new random stage will be immediately selected.
- UNI/BI: Determines the behavior of Out1-3. UNI selects unipolar behavior, meaning that the outputs will be within the range of 0.0-1.0. BI selects bipolar behavior, meaning that the outputs will be within the range of +/- 1.0.
- GATE 1-8: When clicked, the associated stage will be immediately selected.

Inputs:

- Gate 1-8: A positive gate here will cause the sequencer to jump to the associated stage. If two gates are received simultaneously, the higher numbered stage takes precedence.
- Random: A positive gate here will cause the sequencer to jump to a random stage. This input takes precedence over Gates 1-8.

Outputs:

- Out 1-3: Outputs the voltage associated with the row on the active stage.
- G 1-8: Gate outputs. Each output will be true when the associated stage is active.

6.8.21.3 Design Notes

This is a simultaneously polymorphic design inspired by the Make Noise Pressure Points [29]. The most typical use for this is preset storage. By storing three voltages per stage, a user can "save" three values to jump to during a patch. Alternatively, with the Random input gate and the individual output gates, this can be used as

Figure 6.134: Lissajous Panel

a random gate sequencer. When combined with Voltage Controlled Gates, this can act as a very flexible 8-step sequencer. When combined with a Switch, the number of steps in the sequence can be expanded to 16 or 24 steps.

6.9 Utilities

These Blocks deal mostly with controlling Euro Reakt or using Euro Reakt to control other devices.

6.9.1 Lissajous Display

6.9.1.1 Description

This Block is a two-dimensional oscilloscope. It has X and Y inputs, which are then graphed on a Cartesian plane (instead of the traditional amplitude-over-time oscilloscope displays). The primary reason for its inclusion in Euro Reakt is to have a helpful visualizer for the multi-dimensional chaos Blocks. By graphing the chaos

Figure 6.135: Manual Gates Panel

outputs simultaneously, it is easier to see how the outputs are related. It can also be used to graph any two modulation sources to see how closely they interact. Finally, it can be used as a tuner. Two oscillators can be used as inputs. If the oscillators are "in tune" with each other, stable visual forms will appear.

6.9.2 Manual Gates

6.9.2.1 Description

This Block was designed to assist with the connection of MIDI controllers to Blocks that depend on Gates or Triggers, but lack their own MIDI controls. It is also useful as a centerpiece in large patches, where all important gate controls can be reduced to one panel.

There are three different types of gate controls on here:

- GATE - Only stays positive as long as the button is held down.
- TOG - Toggle. Goes positive on the first positive event, and stays positive

Figure 6.136: Meta Control Panel

until the next positive event.

- TRIG - Trigger. Reduces positive events of any lengths to positive events with lengths of only 1 ms.

This Block is very easy to attach to MIDI or OSC controllers. The Gate buttons can be right-clicked, bringing up a MIDI/OSC learn menu.

6.9.3 Meta Control

6.9.3.1 Description

This Block acts as a Macro Generator and/or a 5-way signal copier with per-out amplitude control. Like the Manual Gates Block, it was designed to be a centerpiece control Block for maintaining large patches.

The general idea is that this Block creates four copies of a signal, each with separate amplitude and polarity (using an attenuverter for each output). There are two modes of operation, INT and EXT.

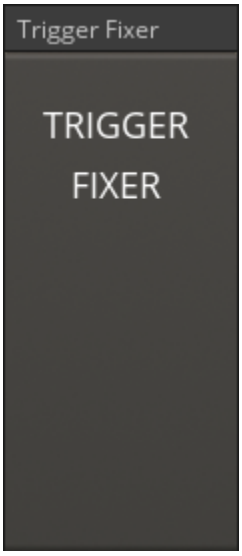

Figure 6.137: Trigger Fixer Panel

In INT mode, an internal offset generator is used as the MAIN input. The MAIN knob is an attenuverter that changes the amplitude and polarity of the offset signal.

In EXT mode, an external signal is used instead of the internal offset generator. Like INT mode, the MAIN knob attenuverts the signal preset on the EXT. IN input.

6.9.3.2 Design Notes

With the INT/EXT switch, this is a modally polymorphic control Block. It can generate signals on its own in INT mode or process external signals in EXT mode.

6.9.4 Trigger Fixer

6.9.4.1 Description

This Block is meant as an interface between Euro Reakt's timing Blocks (many of which use single sample impulses) and external Eurorack hardware (or other digital hardware with clock inputs). This takes triggers and impulses as inputs, and pro-

duces 5 ms gates as outputs. These gates are long enough to be detected by nearly all Eurorack modules, even those with low sampling rates.

Chapter 7

Conclusion

7.1 Evaluation of Work

7.1.1 Euro Reakt

Euro Reakt is the ultimate byproduct of the three taxonomies that I have created. The Blocks make heavy use of rhizomatic and simultaneously polymorphic design. I find these Blocks inspiring to use, as the many inputs and outputs encourage creative patching ideas that I would not have thought of with more monosemous designs.

For the control taxonomy, connecting a MIDI or OSC controller to a Reaktor Block is as simple as right-clicking on any panel control and instantly mapping it to hardware. I have created multiple Blocks that facilitate further control through macro splits (Meta Control), button routers (Manual Gates), and interacting with the modular (Trigger Fixer). Furthermore, outside of the Euro Reakt, the Reaktor environment integrates with hardware modulars extremely well. Native Instruments has released a Block that will tune and quantize any hardware oscillator automatically, along with a Block that translates Block signals to MIDI output [157]. These

Blocks are fully compatible with Euro Reakt's sequencers, allowing Euro Reakt's many complex sequencing paradigms to interact with external gear (or other software synthesizers).

Finally, the exhaustive documentation and large set of example ensembles come from my experience with the third taxonomy. I believe that this library is a valuable addition to any classroom.

Euro Reakt has proven to be successful among Reaktor 6 users. It is currently the highest rated and most downloaded item on the "User Blocks" section of the Reaktor User Library. As of December 2016, it is the 11th most downloaded item of all time in the Reaktor User Library (out of nearly 5,000 available downloadable items). It is also the 12th highest rated item in the library, holding a five star average after 246 reviews. Of the top fifteen most downloaded items in the library, it is the only one with a five star average review. It is also the only item in the top 15 that was updated in the last nine years.

Many popular modular musicians have used Euro Reakt in their Reaktor ensembles, including Richard Devine [158] and Shiro Fujioka [159]. In November 2015, I was invited by Native Instruments to give a presentation in Los Angeles on Euro Reakt as part of their Native Sessions series. A recording of the presentation was posted on YouTube [160]. It currently has over 12,000 views, with 74 "likes" and zero "dislikes". Tutorials for Euro Reakt have been posted by other users, including educators like Point Blank Music School in London [161]. I have received a number of donations from happy users, many of whom sent me personal messages that they had sold portions of their Eurorack systems after using Euro Reakt.

7.1.2 Unfiltered Audio

The patchable modulation system has been a critical and commercial success for our company. In the first professional review available so far, Fault won the "Performance" and "Innovation" awards from Computer Music Magazine [162]. In their review, they say that the modulation system is "a brilliant, intuitive system and one that greatly rewards experimentation". Sandman Pro received a near-perfect review from the same magazine, along with similar praise for the modulation system and many available modes [163].

7.1.3 Taxonomies

On the basis of personal evaluation, the taxonomies have already proven to be useful tools. The design taxonomy was written before the Euro Reakt analysis chapter of this dissertation and lead directly to the Euro Reakt 3.1 and upcoming 4.0 updates. In the 3.1 update, I identified a large number of design redundancies in the chaos Blocks, replacing 11 Blocks with 3 better, more flexible designs. The 4.0 update not only continues this trend of combining similar modules, it also focuses on improving existing designs by identifying new output patch points to create simultaneous polymorphism.

For an example, the Clap Block was one of the first to receive a major upgrade. The Clap Block already had a number of audio outputs, but its design was less versatile than the more generic Drum Block, which had outputs for every internally generated envelope. It made sense to add these envelope outputs to Clap. The amplitude envelope is interesting since it provides a rapid, multi-peak modulation source, while the reverb tail envelope is a delayed modulation with a more pronounced decay. Since these envelopes were already being calculated, it added no CPU overhead to

add these useful outputs to the Block.

Overall, the 4.0 update is the largest expansion yet for Euro Reakt and features over 20 updated Blocks with expanded polymorphism. 4.1 will be released shortly with another batch of design improvements.

So far, these taxonomies have held up well through the analysis of hundreds of available Eurorack modules. So far, there haven't been any modules that evade categorization.

7.2 Future Work

7.2.1 Euro Reakt Updates

After this dissertation is filed, I will release a large microsound update for Euro Reakt. This update will focus primarily on grain generation. These Blocks include a Window Generator (featuring multiple window types, variable exponentiation, and a linked phase output) and an FOF oscillator. Many other Blocks will receive mindful upgrades, including a phase output on the Burst Generator (allowing for precise windows to be generated in complex trigger streams).

In addition to this microsound-focused update, there is also the constant push for more polymorphic and rhizomatic design in Euro Reakt. During the writing of the design section of this dissertation, many Blocks (including Clap and Comb Oscillator) had their DSP rewritten to accommodate more simultaneous outputs. The Clap Block, for instance, received additional outputs for its internal amplitude and reverb envelopes, along with an output that provided only the simulated reverb tail. In total, twenty Blocks were rewritten to add simultaneous polymorphism for the 4.0 update.

More Blocks are on the to-do list, including Gendy (a stochastic oscillator based on a design by Iannis Xenakis), a rotating switch (similar to RYO Paths to create rotating outputs like the 4ms Rotating Clock Divider), a Complex Random in the vein of the Buchla Source of Uncertainty or Make Noise Richter/Wiard Wogglebug, and ports of many more SuperCollider uGens, including Formlet, Klank, GravityGrid, Ringz, and DoubleWell.

Aside from new and upgraded Blocks, there's also a focus on removing redundancies from the library. While writing the design section of this dissertation, it became very clear that many of the Chaos Blocks shared identical interfaces with different (but similar) algorithms. Eleven of these Blocks were combined to create three much simpler (and ultimately more versatile) Blocks. One specific redundancy that will be eliminated is the inclusion of many different Schroeder reverbs with identical control sets (SATREV, JCREV, and JCREV FF). It would make sense to combine these in an effort to make the library less difficult to navigate.

7.2.2 Unfiltered Audio

7.2.2.1 Modulation System

We are currently hard at work adding many improvements to the patchable modulation system at the heart of all of our new plug-ins. Our first major task is to improve the existing modules, including the LFOs and Envelope Follower.

Right now, the LFOs only reset when the DAW's transport restarts, and every LFO resets to phase 0. To improve this, we are adding a phase reset input to each LFO. This can be manually triggered via mouse click or automatically triggered from another modulation source. We are also working on two designs for phase control. Our first design is to simply add a Phase knob to each LFO. This knob would set the

phase that the LFO would reset to. This is a simple, easy-to-implement concept. Our more interesting concept is to add variable phase to each output on the LFO (right below the per-output amplitude controls). This is much more difficult to implement in a CPU-efficient manner, but would allow for a lot more variety in patches.

For the Envelope Follower, we want to add one option: the ability to follow a sidechain input. This would greatly expand the capabilities of all existing plug-ins, as they become more aware of other signals in the DAW (thus making each plug-in more "rhizomatic").

With those upgrades in place, we want to improve the user experience for managing more complicated systems. There is currently a limit of six modules available at a time, but unlimited outputs. This can quickly lead to modules and cables going off screen, requiring the user to scroll to see all current modules. We want to add the option to collapse modules, making it easier to hide sections that are not being worked on. Furthermore, we want to add per-module mute switches to better listen to each modules effect on the system.

After we tackle all of these refinements, we are going to add new modules. The first two that we'd like to implement are A(H)D and ADSR envelopes. These envelopes will use the same trigger system as the upgraded LFOs. These modules are not only nice improvements for the effect plug-ins, but critical modules for future instrument plug-ins. In addition to the envelopes, we will eventually add a step sequencer, a drawable LFO, and a MIDI note converter.

7.2.2.2 Plug-ins

Early next year we will start releasing plug-ins based on original research, including a commercial implementation of Ryan McGee's Spatial Modulation Synthesis [164]

and potentially an implementation of Multi Phasor Synthesis, described earlier in this dissertation.

Most relevant to the design ideas present in this dissertation is SpecOps, our upcoming multi-mode spectral processor. SpecOps was designed and developed as a response to most of the spectral plug-in packages currently available, including GRM Tools, Soundmagic Spectral, and Soundhack's Spectral Shapers. Each of those packages break a number of spectral utilities (pitch shifting, brickwall filtering, etc.) down into separate plug-ins. Each plug-in thus incurs the computational overhead and latency penalty of an FFT and iFFT.

SpecOps is a design that combines linked and modal polymorphism to overcome this problem. In SpecOps, the incoming signal only goes through one FFT and iFFT. The spectral bin data is then shared between a number of sub-processors. The most common processors are always available, including pitch shifting and freezing. More esoteric filters are available in modal sections. There are two modal sections with smooth controls (filters with variable cutoff, variable noise reduction/focus, etc.) and two modal sections with toggled controls (binary effects such as neighbor filtering).

By taking advantage of these modular design principals, we've created a plug-in that has many advantages over the single-function plug-in packages:

- Only incur the latency and computational overhead of one FFT, usually the most expensive part of a spectral effect.
- Allow a user to quickly experiment with swapping spectral algorithms instead of deleting and loading plug-ins.
- Share a single modulation system between various spectral processors.

7.2.2.3 Hardware

We are currently developing firmware for commercial Eurorack modules, on track for release next year. For business purposes, I will leave these designs out of this public dissertation and present these at my dissertation defense.

7.2.3 Modular Recordings

At the end of this dissertation, I am excited to return to my role as a modular composer. In 2017, I plan on finally releasing an album of modular recordings with the hours of material that I have.

Bibliography

[1] K. Schade, "Modular Grid," 2015. [Online]. Available: https://www.modulargrid.net

[2] "Doepfer-Story - Der Modul-Mogul," 1997. [Online]. Available: http://www.doepfer.de/home{_}e.htm

[3] K. Schade, "Modular Grid: Modules," 2015. [Online]. Available: https://www.modulargrid.net/e/modules

[4] A. Ostler, "Expert Sleepers Disting," p. 1, 2016. [Online]. Available: http://www.expert-sleepers.co.uk/disting.html

[5] Synthrotek, "ECHO - Voltage Controlled Echo," 2014. [Online]. Available: http://www.synthrotek.com/products/modular-circuits/echo/

[6] C. Randall and A. Schabtach, "DubJr Mk2," p. 1, 2015. [Online]. Available: http://www.audiodamage.com/hardware/product.php?pid=ADM16

[7] J. Mungo, "Mungo Eurorack Overview," p. 15, 2016. [Online]. Available: http://mungo.com.au/Doc/MungoEuro.pdf

[8] MacroMachines, "Macro Machines Products," p. 1, 2016. [Online]. Available: http://macromachines.net/shop/

[9] M. Pulver and G. Milstein, "Tiptop Audio Z-DSP User Manual," p. 19, 2009. [Online]. Available: http://www.tiptopaudio.com/manuals/z-dsp.pdf

[10] BrokenSilicon, "Error Codes #1," p. 2, 2014. [Online]. Available: http://analoguehaven.com/brokensilicon/errorcodes1zdspcard/manual.pdf

[11] R. Filippov, "Four-tap Delay/Dual Crossfader Manual," p. 7, 2014. [Online]. Available: http://sputnik-modular.com/four-tap-delay-dual-crossfader/

[12] A. Blaze, "Folktek Conduit," p. 1, 2016. [Online]. Available: http://folktek.com/instruments/modular/conduit-modular

[13] T. Rolando, "Make Noise Echophon Manual," Asheville, 2013. [Online]. Available: http://www.makenoisemusic.com/content/manuals/EchophonManual.pdf

[14] D. Snazelle, "Snazzy FX Dronebank Manual," p. 13, 2014. [Online]. Available: http://www.analoguehaven.com/snazzyfx/dronebank/manual.pdf

[15] T. Rolando, "Make Noise STO Manual," Asheville, p. 9, 2013. [Online]. Available: http://www.makenoisemusic.com/content/manuals/STOManual.pdf

[16] W. Mathewson, "WMD Synchrodyne," p. 1, 2012. [Online]. Available: https://www.wmdevices.com/collections/eurorack-modules/products/synchrodyne

[17] ——, "WMD Synchrodyne Expand," p. 1, 2013. [Online]. Available: https://www.wmdevices.com/collections/eurorack-modules/products/synchrodyne-expand

[18] O. Gillet, "Braids Quick Start Guide," Paris, p. 2, 2013. [Online]. Available: http://mutable-instruments.net/static/manual/braids{_}quickstart.pdf

[19] RolandCorp, "Roland System-500 512 Parameters," p. 1, 2015. [Online]. Available: https://static.roland.com/assets/media/pdf/SYS-512{_}parameter{_}guide{_}e01.pdf

[20] D. van Tijn, "Intellijel Atlantis," p. 1, 2013. [Online]. Available: https://intellijel.com/eurorack-modules/atlantis/

[21] S. Jaeger, "Piston Honda Mk. 2 Manual," p. 1, 2012. [Online]. Available: http://www.theharvestman.org/pistonmk2.pdf

[22] O. Gillet, "Peaks Quick Start Guide," Paris, p. 2, 2013. [Online]. Available: http://mutable-instruments.net/static/manual/peaks{_}quickstart.pdf

[23] T. Rolando, "Make Noise Maths," p. 1, 2013. [Online]. Available: http://www.makenoisemusic.com/maths.shtml

[24] SergeFans, "Serge DUSG," p. 1. [Online]. Available: http://www.serge-fans.com/m-class-quad-slope.cfm

[25] J. Clark and D. van Tijn, *Intellijel Cylonix Shapeshifter Manual*, 1st ed. Intellijel, 2013. [Online]. Available: https://dl.dropboxusercontent.com/u/84988338/shapeshifter{_}manual{_}1.4.pdf

[26] M. Verbos, "Verbos Electronics: Module List," p. 1, 2016. [Online]. Available: http://www.verboselectronics.com/modules/

[27] R. Filippov, "Multitouch Keyboard/Controller Manual," Portland, p. 9, 2015.

[Online]. Available: http://sputnik-modular.com/wp-content/uploads/2015/11/SPUTNIK-KEYBOARDCONTROLLER-MANUAL-v1.pdf

[28] W. Mathewson, "WMD Poly Pressure Key-Array Teaser," p. 1, 2014. [Online]. Available: https://www.youtube.com/watch?v=3WBbAWo9pW8

[29] T. Rolando, "Pressure Points Manual," Asheville, p. 10, 2010. [Online]. Available: www.makenoisemusic.com/content/manuals/PressurePointsManual.pdf

[30] Synthwerks, "analoguehaven: Synthwerks FSR Series." [Online]. Available: http://www.analoguehaven.com/synthwerks/

[31] D. van Tijn, "Intellijel Planar," p. 1, 2016. [Online]. Available: https://intellijel.com/eurorack-modules/planar/

[32] FlightOfHarmony, "Flight of Harmony Choice Manual," p. 6, 2009. [Online]. Available: http://www.analoguehaven.com/flightofharmony/choices/manual.pdf

[33] Soundmachines, "LP1 Lightplane Manual," Fabriano, p. 7, 2013. [Online]. Available: http://www.sound-machines.it/cms/wp-content/uploads/2013/11/LP1lightplane{_}Manual{_}Sep2013.pdf

[34] Elektron, "Analog Four," 2012. [Online]. Available: https://www.elektron.se/products/analog-four/

[35] TeenageEngineering, "Pocket Operators," p. 1, 2016. [Online]. Available: https://teenage.engineering/products/po

[36] Korg, "Volca Keys," p. 1, 2015. [Online]. Available: http://www.korg.com/us/products/dj/volca{_}keys/

[37] O. Gillet, "Mutable Instruments Yarns Quick Start Guide," Paris, p. 2, 2014. [Online]. Available: http://mutable-instruments.net/static/manual/yarns{_}quickstart.pdf

[38] D. Doepfer, "A-190-2 Low Cost MIDI-to-CV/Gate Interface," p. 1. [Online]. Available: http://www.doepfer.de/a1902.htm

[39] R. Nicol, "Midi 3," p. 1. [Online]. Available: http://pittsburghmodular.com/midi3/

[40] P. Schreiber, "E620 USB Host/Device," p. 1, 2014. [Online]. Available: https://www.muffwiggler.com/forum/viewtopic.php?t=122583

[41] A. Ostler, "Expert Sleepers - FH-1 USB MIDI Host," p. 1, 2016. [Online]. Available: http://www.expert-sleepers.co.uk/fh1.html

[42] B. Crabtree and K. Cain, "monome Ansible," p. 1, 2016. [Online]. Available: http://monome.org/docs/modular/ansible/

[43] A. Ostler, "Expert Sleepers Module Overview," p. 1, 2016. [Online]. Available: http://www.expert-sleepers.co.uk/moduleoverview.html

[44] ——, "Expert Sleepers - ES-3 Lightpipe/CV Interface," p. 1, 2016. [Online]. Available: http://www.expert-sleepers.co.uk/es3.html

[45] ——, "Expert Sleepers - ES-4 SPDIF/CV Interface," p. 1, 2016. [Online]. Available: http://www.expert-sleepers.co.uk/es4.html

[46] ——, "Expert Sleepers - ES-8 USB Audio Interface," p. 1, 2016. [Online]. Available: http://www.expert-sleepers.co.uk/es8.html

[47] ——, "Expert Sleepers - Silent Way," p. 1, 2016. [Online]. Available: http://www.expert-sleepers.co.uk/silentway.html

[48] ——, "Expert Sleepers - ES-1 Audio/CV Interface," p. 1, 2016. [Online]. Available: http://www.expert-sleepers.co.uk/es1.html

[49] B. Crabtree and K. Cain, "monome Walk," p. 1, 2015. [Online]. Available: http://monome.org/docs/modular/walk/

[50] Eowave, "EO-310 Sensor Signal Processor," p. 1, 2016. [Online]. Available: http://www.eowave.com/modules/eo-310-sensor-signal-processor/

[51] B. Crabtree and K. Cain, "monome Teletype," p. 1, 2015. [Online]. Available: http://monome.org/docs/modular/teletype/

[52] ADDAC, "ADDAC302 Nchunk," p. 1, 2011. [Online]. Available: http://www.addacsystem.com/product/addac300-series/addac302-nchunk

[53] J. Bartee, C. Novello, M. Roberts, and C. Bartee, "Control Core," p. 1, 2013. [Online]. Available: http://www.specialstagesystems.com/control/

[54] Scanner_darkly, "Orca - alternative firmware for white whale and arc," p. 1, 2015. [Online]. Available: http://llllllll.co/t/orca-alternative-firmware-for-white-whale-and-arc-and-now-grid/351

[55] B. Crabtree and K. Cain, "monome Module List," p. 1, 2016. [Online]. Available: http://monome.org/modular/

[56] S. Jaeger, "English Tear Manual," Seattle, p. 1, 2014. [Online]. Available: http://www.theharvestman.org/manuals/ENGLISHTEARmanual.pdf

[57] T. Rolando, "Make Noise Format Jumbler," p. 1, 2009. [Online]. Available: https://www.modulargrid.net/e/make-noise-format-jumbler

[58] K. McMillen, "QuNexus: The keyboard that connects to everything," p. 1, 2016. [Online]. Available: https://www.keithmcmillen.com/products/qunexus/

[59] BEMI, "Buchla LEM3 Spider," p. 1, 2015. [Online]. Available: https://buchla.com/product/buchla-lem3-spider/

[60] A. Ostler, "Expert Sleepers - ES-6 CV/Lightpipe Interface," p. 1, 2016. [Online]. Available: http://www.expert-sleepers.co.uk/es6.html

[61] Cycling 74, "Max 7." [Online]. Available: https://cycling74.com/max7/

[62] W. Farrell, *Black & Gold Shared System*, 1st ed. Asheville: Make Noise, 2016. [Online]. Available: http://www.makenoisemusic.com/content/manuals/sharedsystemmanual{_}1.10.pdf

[63] R. Nicol, "Foundation 3.1+," p. 1, 2016. [Online]. Available: http://pittsburghmodular.com/foundation-3-1-1/

[64] M. Hetrick, "Shared System Patches by The February Thaw," p. 1, 2014. [Online]. Available: https://soundcloud.com/the-february-thaw/sets/shared-system-patches

[65] Make Noise, "Shared System Series Side A," p. 1, 2014. [Online]. Available: https://soundcloud.com/shared-system-series/sets/shared-system-series-side-a

[66] M. Hetrick, "All of my Shared System patches, including diagrams," p. 1, 2014. [Online]. Available: https://www.muffwiggler.com/forum/viewtopic.php?t=119006{&}highlight=

[67] CREATE, "CREATE Concert: ELECTRO ACOUSTIC," p. 1, 2015. [Online]. Available: http://www.create.ucsb.edu/event/create-concert-2/

[68] M. Hetrick, "New Leaf (Live Modular 04/15/15)," p. 1, 2015. [Online]. Available: https://soundcloud.com/the-february-thaw/new-leaf-live-modular-041515

[69] R. Buskin, "CLASSIC TRACKS: Heroes," 2004. [Online]. Available: http://www.soundonsound.com/sos/Oct04/articles/classictracks.htm

[70] D. Doepfer, "Doepfer A-136 Manual." [Online]. Available: http://www.doepfer.de/a100{_}man/A136{_}man.pdf

[71] A. Gratz, "Operational Transconductance Amplifiers." [Online]. Available: http://synth.stromeko.net/diy/OTA.pdf

[72] M. Hetrick, "Euro Reakt," p. 1, 2015. [Online]. Available: https://www.native-instruments.com/en/reaktor-community/reaktor-user-library/entry/show/9093/

[73] Native Instruments, "Reaktor 6," p. 1, 2015. [Online]. Available: https://www.native-instruments.com/en/products/komplete/synths/reaktor-6/

[74] A. Hanley, *Reaktor 6: What is New.* Berlin, Germany: Native Instruments, 2008.

[75] V. Zavalishin, *Building in Core.* Berlin, Germany: Native Instruments, 2015.

[76] G. Wakefield, "Real-Time Meta-Programming for Interactive Computational Arts," Ph.D. dissertation, 2012.

[77] D. Forrester, *Reaktor Blocks Framework Manual*. Native Instruments, 2015.

[78] SuperCollider, "SuperCollider Classes: Crackle," p. 1. [Online]. Available: http://doc.sccode.org/Classes/Crackle.html

[79] M. Hetrick, "Euromax for Max/MSP," Santa Barbara, p. 1, 2010. [Online]. Available: http://mhetrick.com/121800/1219320/software/euromax-for-maxmsp

[80] J. Eriksson, "XODULAR," p. 1, 2015. [Online]. Available: http://www.monologx.com/xodular/

[81] M. Davidson, "BEAP - Powerful Modules for Max for Live," p. 1, 2013. [Online]. Available: https://www.ableton.com/en/blog/beap-powerful-modules-max-live/

[82] D. Grosse, "A Few Minutes with BEAP, Part 1," p. 1, 2015. [Online]. Available: https://cycling74.com/2015/09/15/a-few-minutes-with-beap-tutorial-series/

[83] MaxForCats, "OSCiLLOT," p. 1, 2015. [Online]. Available: https://www.ableton.com/en/packs/oscillot/

[84] C. Singer, "Ampere Modular for Reaktor," p. 1, 2009. [Online]. Available: https://www.native-instruments.com/en/reaktor-community/reaktor-user-library/entry/show/6019/

[85] S. Small, "The Infinite Phi Collection," p. 1, 2016. [On-

line]. Available: https://www.native-instruments.com/en/reaktor-community/reaktor-user-library/entry/show/9149/

[86] M. Friedrichs, "Nouveau Collection," p. 1, 2016. [Online]. Available: https://www.native-instruments.com/en/reaktor-community/reaktor-user-library/entry/show/9997/

[87] B. Lavallee, "Brett Blocks," p. 1, 2016. [Online]. Available: https://www.native-instruments.com/en/reaktor-community/reaktor-user-library/entry/show/10403/

[88] J. Tremblay, "The Synite Collection," p. 1, 2016. [Online]. Available: https://www.native-instruments.com/en/reaktor-community/reaktor-user-library/entry/show/10401/

[89] J. Punter, "WREN," p. 1, 2016. [Online]. Available: http://bluehell.electro-music.com/modules/{#}license

[90] C. Jones, "Sonigen Modular," p. 1, 2014. [Online]. Available: http://www.sonigen.com/

[91] M. Boyd, "Audulus 3.0 Documentation," p. 1, 2016. [Online]. Available: http://docs.audulus.com/

[92] Bitcount, "AnalogKit on the App Store," p. 1, 2015. [Online]. Available: https://itunes.apple.com/us/app/analogkit/id984597969?mt=8

[93] N. Trass, "zMors Modular Manual," p. 57, 2016. [Online]. Available: http://www.zmors.de/zMors{_}Modular{_}Manual.pdf

[94] Arturia, "Modular V," p. 1, 2016. [Online]. Available: https://www.arturia.com/products/analog-classics/modular-v

[95] Moog Music Inc., "Model 15 App," p. 1, 2016. [Online]. Available: https://www.moogmusic.com/products/apps/model-15-app/{#}info-tab

[96] Softube, "Softube - Modular," p. 1, 2016. [Online]. Available: https://www.softube.com/modular

[97] Clavia, "Nord Modular V3.0 User Manual," 2003. [Online]. Available: Clavia

[98] Electro-music.com, "Clavia Nord Modular Subforum," p. 1, 2016. [Online]. Available: http://www.electro-music.com/forum/forum-43.html

[99] J. Clark, *Advanced Programming Techniques for Modular Synthesizers*, 2003. [Online]. Available: http://www.cim.mcgill.ca/{~}clark/nordmodularbook/nm{_}book{_}toc.html

[100] R. Hordijk, "G2 Workshops and tutorials." [Online]. Available: http://rhordijk.home.xs4all.nl/G2Pages/index.htm

[101] R. Kuit, "E-Books Modular Synthesis," p. 1, 2016. [Online]. Available: http://rolandkuit.com/e-Books.html

[102] J. Taelman, "Axoloti," p. 1, 2015. [Online]. Available: http://www.axoloti.com/

[103] B. Crabtree, K. Cain, and E. Buchla, "Aleph: Bees," p. 1, 2014. [Online]. Available: http://monome.org/docs/aleph/bees/

[104] P. Blasser, "Shbobo Shnth," p. 1, 2013. [Online]. Available: http://shbobo.net/

[105] Roland, "AIRA Effector Product List," p. 1, 2015. [Online]. Available: http://www.roland.com/global/aira/categories/effector.html

[106] O. Gillet, "Mutable Instruments Braids Source Code," 2013.

[107] D. Doepfer, "A-189-1 Voltage Controlled Bit Modifier / Bit Cruncher," p. 1, 2008. [Online]. Available: http://www.doepfer.de/a1891.htm

[108] A. Fitch, "DelayNoMore Build Notes," Perth, Australia, p. 3, 2014. [Online]. Available: http://www.sdiy.org/pinky/data/DelayNoMorebuildnotes.pdf

[109] J. Dattorro, "Effect Design. Part 1: Reverberator and Other Filters," *AES*, vol. 45, no. 9, p. 25, 1997. [Online]. Available: https://ccrma.stanford.edu/{~}dattorro/EffectDesignPart1.pdf

[110] O. Gillet, "Mutable Instruments Clouds Quick Start Guide," Paris, p. 2, 2014. [Online]. Available: http://mutable-instruments.net/static/manual/clouds{_}quickstart.pdf

[111] T. Rolando, "Soundhack Erbe-Verb," Asheville, p. 14, 2014. [Online]. Available: http://www.makenoisemusic.com/content/manuals/Erbe-VerbManual.pdf

[112] S. Gutsfeld, "Segoh Bit Rot," p. 1, 2014. [Online]. Available: http://www.analoguehaven.com/segoh/bitrotzdspcard/

[113] J. O. Smith, "Schroeder Reverberators." [Online]. Available: https://ccrma.stanford.edu/{~}jos/pasp/Schroeder{_}Reverberators.html

[114] D. Doepfer, "Doepfer A-137-2 Wave Multiplier 2." [Online]. Available: http://www.doepfer.de/a1372.htm

[115] B. Hutchins, "A SAWTOOTH-DRIVEN MULTI-PHASE WAVEFORM ANIMATOR : THE SYNTHESIS OF " ANIMATED " SOUNDS - PART 1 ;," *Electronotes*, vol. 87, no. 3, pp. 3–11.

[116] S. Schmitt, "Ez FFT Christmas Collection," 2006. [Online]. Available: https://www.native-instruments.com/en/reaktor-community/reaktor-user-library/entry/show/4453/

[117] T. Rolando, "Make Noise DPO: Official Manual," Asheville, p. 14, 2012. [Online]. Available: http://www.makenoisemusic.com/content/manuals/dpo-manual.pdf

[118] Unknown, "Fold Back Distortion." [Online]. Available: http://musicdsp.org/archive.php?classid=4{#}203

[119] R. Jones, "max/pd versions of serge and buchla wavemultipliers/folders?" p. 2, 2014. [Online]. Available: https://www.muffwiggler.com/forum/viewtopic.php?t=125642

[120] Toppobrillo, "Toppobrillo Triple Wave Folder," p. 1, 2012. [Online]. Available: https://www.modulargrid.net/e/toppobrillo-triple-wave-folder

[121] C. Roads, *Microsound*. MIT Press, 2004.

[122] J. O. Smith, "Example Schroeder Reverberators." [Online]. Available: https://ccrma.stanford.edu/{~}jos/pasp/Example{_}Schroeder{_}Reverberators.html

[123] A. Fitch, "Segue Build & BOM," Perth, Australia, p. 2, 2015. [Online]. Available: http://www.sdiy.org/pinky/data/SeguebuildBOM.pdf

[124] J. J. Clark, *Advanced Programming Techniques for Modular Synthesizers*, 2003. [Online]. Available: http://www.cim.mcgill.ca/{~}clark/nordmodularbook/nm{_}book{_}toc.html

[125] C. Roads, *The Computer Music Tutorial*. MIT Press, 1996.

[126] A. Fitch, "difference rectifier," p. 1, 2012. [Online]. Available: http://www.sdiy.org/pinky/data/dif.html

[127] ——, "Difference Rectifier/Neuron Build Doc," Perth, Australia, p. 9, 2013. [Online]. Available: http://www.sdiy.org/pinky/data/diffrectneuronbuilddocv1.pdf

[128] ——, "seqmix1050 Build and BOM Guide," Perth, Australia, p. 4, 2015. [Online]. Available: http://www.sdiy.org/pinky/data/mixseq1050BuildBOMvers1.pdf

[129] ——, "WAMOD #10 - dual LFO," Perth, Australia, p. 2, 2015. [Online]. Available: http://www.sdiy.org/pinky/data/WAMOD10dualLFO.pdf

[130] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, dec 1943. [Online]. Available: http://link.springer.com/10.1007/BF02478259

[131] S. Small, "Microwave Oscillator," p. 1, 2015. [Online]. Available: https://www.native-instruments.com/en/reaktor-community/reaktor-user-library/entry/show/9149/

[132] Vintage Synth Explorer, "Waldorf Microwave," p. 1, 2016. [Online]. Available: http://www.vintagesynth.com/waldorf/microwave.php

[133] G. Reid, "Synth Secrets | Sound On Sound," p. 64, 2004. [Online]. Available: http://www.soundonsound.com/techniques/synth-secrets

[134] O. Gillet, "Rings Resonator Quick Start Guide," Paris, p. 2, 2015. [Online]. Available: http://mutable-instruments.net/static/manual/rings{_}quickstart.pdf

[135] ——, "Elements Quick Start Guide," Paris, p. 2, 2014. [Online]. Available: http://mutable-instruments.net/static/manual/elements{_}quickstart.pdf

[136] R. Hordijk, "The Rungler," 2015. [Online]. Available: http://hordijk-synths.info/synth/2015/03/19/the-rungler.html

[137] ——, "electro-music.com : Benjolin schematics," 2009. [Online]. Available: http://www.electro-music.com/forum/topic-38081.html

[138] S. McCaul, "Loquelic Iteritas Complex Digital Oscillator," Los Angeles, p. 14, 2015. [Online]. Available: https://static1.squarespace.com/static/5572aedde4b0080836614a7b/t/5785a22fb8a79b55397469ef/1468375924429/LI{_}manual.pdf

[139] R. Hordijk, "VOSIM, an advanced application of oscillator sync," p. 1, 2000. [Online]. Available: http://electro-music.com/nm{_}classic/015{_}workshops/Clavia/NordModularWorkshops{&}Threads/WerkMap/WorkShops/Hordijk1999-2000/VOSIM.html

[140] K. Ikeda and K. Matsumoto, "High-dimensional chaotic behavior in systems with time-delayed feedback," *Physica D: Nonlinear Phenomena*, vol. 29, no. 1-2, pp. 223–235, nov 1987. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/0167278987900583

[141] E. W. Weisstein, "Tent Map," *Wolfram Mathworld*, 2016. [Online]. Available: http://mathworld.wolfram.com/TentMap.html

[142] R. C. Hilborn, *Chaos and Nonlinear Dynamics*. Oxford University Press, sep 2000. [Online]. Available: http://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780198507239.001.0001/acprof-9780198507239

[143] E. E. Hairer, S. P. S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations. I, Nonstiff problems*, 1987.

[144] C. A. Pickover and C. A., *The pattern book : fractals, art, and nature*. World Scientific, 1995.

[145] ——, *Chaos in wonderland : visual adventures in a fractal world*. St. Martin's Griffin, 1995.

[146] K. T. Alligood, T. Sauer, and J. A. Yorke, *Chaos : an introduction to dynamical systems*. Springer, 1996.

[147] W. Tucker and W. Tucker, "The Lorenz attractor exists," *C. R. ACAD. SCI. PARIS*, vol. 328, pp. 1197—-1202, 1999. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.34.9324

[148] O. Rössler, "An equation for continuous chaos," *Physics Letters A*, vol. 57, no. 5, pp. 397–398, 1976.

[149] R. L. Devaney, "Fractal patterns arising in chaotic dynamical systems," in *The Science of Fractal Images*. New York, NY: Springer New York, 1988, pp. 137–168. [Online]. Available: http://link.springer.com/10.1007/978-1-4612-3784-6{_}3

[150] A. Morelli, "Quantum Rainbow 2," p. 1, 2014. [Online]. Available: http://www.steadystatefate.com/quantum-rainbow-2

[151] A. Fitch, "Squid Axon build & BOM," Perth, Australia, p. 3, 2016. [Online]. Available: http://www.sdiy.org/pinky/data/SquidAxonbuildandBOM.pdf

[152] T. Matsuzaki and M. Nakagawa, "A bipolar logistic chaos neuron and its hardware implementation," *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, vol. 86, no. 11, pp. 46–56, nov 2003. [Online]. Available: http://doi.wiley.com/10.1002/ecjc.10063

[153] A. Fitch, "Numberwang Build Guide and BOM," Perth, Australia, p. 3, 2016. [Online]. Available: http://www.sdiy.org/pinky/data/NumberwangBOMguide.pdf

[154] O. Gillet, "Branches Dual Bernoulli Gate," p. 1, 2014. [Online]. Available: http://mutable-instruments.net/modules/branches

[155] T. Whitwell, "Music Thing Modular: Turing Machine Random Looping Sequencer Documentation," 2016. [Online]. Available: http://musicthing.co.uk/modular/?page{_}id=21

[156] D. Doepfer, "Doepfer A-152 Manual." [Online]. Available: http://www.doepfer.de/a100{_}man/A152{_}man.pdf

[157] Native Instruments, "Reaktor 6 1.2 Update," p. 1, 2016. [Online]. Available: https://www.native-instruments.com/en/products/komplete/synths/reaktor-6/connect/

[158] R. Devine, "DevineAcid," p. 1, 2016. [Online].

Available: https://www.native-instruments.com/en/reaktor-community/reaktor-user-library/entry/show/9825/

[159] S. Fujioka, "VoltageCtrlr-We Rise," p. 1, 2016. [Online]. Available: https://www.native-instruments.com/en/reaktor-community/reaktor-user-library/entry/show/10297/

[160] Native Instruments, "Native Sessions: Play. Patch. Build. - EuroReakt Blocks with Michael Hetrick," p. 1, 2016. [Online]. Available: https://www.youtube.com/watch?v=nL4TNaJm-3M

[161] Point Blank Music School, "Plugin of the Week: Euro Reakt for Reaktor Blocks - YouTube," p. 1, 2016. [Online]. Available: https://www.youtube.com/watch?v=YAb2S0ZNDl0

[162] R. Macdonald, "Fault Review," *Computer Music*, p. 97, nov 2016.

[163] ——, "Sandman Pro Review," *Computer Music*, dec 2016.

[164] R. McGee, "Spatial Modulation Synthesis," *ICMC*, p. 4, 2015. [Online]. Available: https://www.mat.ucsb.edu/Publications/SpatialModulationSynthesis{_}ICMC2015.pdf