

Name: Maxime Burgener

Student Number: 1904587

Module Title: Development
(ADAD7005_LON)

Assignment Title: Element 2b

Module Lecturer: Ciaran Robinson

Introduction

When I came to know that we had to build our own instrument, I felt intimidated by the task, never having gotten into sound design and modular synthesis before. However, after doing some research, I started to get a better idea of the kind of project I wanted to lead for the assignment. I decided to build my own digital synthesiser into the graphical modular software Reaktor developed by Native Instruments. From all the different options we had for the assignment, building my own software instrument from the ground up would be the most beneficial for me as a music producer and the process would give me valuable knowledge on the basic principles of synthesis. From assembling the components that make a synthesiser to understanding their role and function in the machine, my research would give me the knowledge I need to create my own pre-sets on soft - synths, to understand the basic functionality of any synthesiser I come across and to go as far as to create my own synthesisers in the future, which will take my sound design and production skills to the next level.

In this essay, I will firstly explain in detail every step I have taken to create my software synthesiser; from building the chore of the synths with different oscillators, to adding filters and effects in order to create interactive and exciting sounds that can be used in professional productions. Secondly, I will explain the issues I have encountered while creating my instrument and how I managed to overcome or adapt to them. Finally, I will talk about the design and visual of the product, which is a really important aspect of a soft-synth's commercial success if it happens to be sold on the plug-ins' market in the future (Stephenson, 2019).

The Core

Reaktor is a software that allows producers and engineers to access the different modules necessary to build a synthesizer in a virtual environment. The assembly of different modules to create a coherent unit is what is commonly called modular synthesis. The modules are electronic circuit that are designed to take on specific music - related tasks in the synth's circuit. There are thousands of modules available on the market, made by hundreds of companies but usually they fall into one of these categories (Makes, 2019):

- Oscillators
- Filters
- Amplifiers
- LFO's
- Envelope Generator
- Effects
- General Utility

For my oscillators' section, I have decided to model the soft-synth Massive by Native Instruments and create three different oscillators so I can get a lot of different options in terms of the primary sound selection. Each of my three oscillators allows the user to choose between four different waveforms. The *Waveform* modules are designed to create a repeated and continuous signal resulting in the generation of a sound (Electrical4U, 2019). For my synthesiser, I have decided to use the triangle, the pulse, the sawtooth and the sine waveforms, which are considered the most common periodic waveforms (Comeau, 2020). My first two oscillators, called A and B, are identical to each other (Appendix; Fig. 1). However, for my third oscillator I have removed the sine waveform to replace it with a *Noise* module, which is a common module found in all kinds of synthesisers (Appendix; Fig. 2). The *Noise* modules are designed to create a random electric signal that will typically output a

white or pink noise and therefore add a distinguishable texture to the sound when routed through different oscillators, filters and voltage control amplifiers (Sweetwater, 2006). Noise oscillators cannot be pitched since their primary role is to generate noise (Rise, 2020). In order to control my three oscillators with a midi keyboard, I added a *Note Pitch* module to pick up the midi signal, as well as a *Gate* module to manage the signal flow when a note is being triggered or released. The *Gate* module also has the ability to detect the velocity of the midi signal and trigger the oscillators accordingly. Furthermore, I have added a *Selector/Scanner* module to crossfade between the four different waveforms as well as one *Multiply* module at the end of the signal to create a volume knob.

After creating my three oscillators, I figured that it would be important to create some form of controls for each of them in order to enable its' user to alter the sound of the oscillators to their liking. The first control I made was the P-Width, which is an option available exclusively for the *Pulse* modules (Corman, 2015). It allows the user to control the pulse width for the square shape, thus emulating the amplitude modulation of an analogue input signal (Christ & Wernli, 2014). Being able to control the tuning of my oscillators is another crucial element for sound design. In order to put into place tuning controls, I created an *Add* module with two control knobs attached. The *Add* module has the ability to alter pitch by subtracting or adding a value to the data coming from the *Note Pitch* module. In order to control the pitch, I routed the *Note Pitch* module into the first input of the *Add* module. I labelled the first knob Coarse and changed its maximum and minimum value to 24 and -24 in order to allow altering the pitch by two octaves above or below. The second knob is used as a fine tuner with a default range that varies between -1 and 1, which is perfect for fine-tuning (Appendix; Fig. 1&2).

At this point, I had my three oscillators working well together with a few important controls at my disposal. However, because the *Gate* module was responsible for triggering the amplitude, the oscillators' amplitude was jumping to its maximum every time a MIDI note was triggered. This was inconvenient since I was aiming to get more versatility from the synthesiser. In order to control the amplitude, I used one of the most common envelopes called the ADSR envelope (Mantione, 2017). The *ADSR* module gives the option to tweak the attack, decay, sustain and release of pretty much anything in the signal path (Crute, 2019). To work effectively, the ADSR had to be placed in between the *Gate* module and each oscillator. At this point, I had a solid foundation featuring three functioning oscillators with one ADSR envelope to control their amplitude (Appendix; Fig. 3).

Filters

As the name suggests, filters play a massive role in subtractive synthesis; they give the user a wide array of possibility when shaping sounds and playing with modulations (Crute, 2019). For my synthesiser, I have decided to create a low pass filter that will feature cut off frequency and resonance's control as well as an LFO and filter envelope's modulation for the cut off.

To create the low pass filter, I used a *Ladder Filter* module with an LP4 outputting a 24db/oct slope, which is the most common filter curve in subtractive synthesis (Rishabh, 2016). I created two controls from the module so I could tweak the cut off and the resonance of the ladder filter to taste. Like I did for the amplitude envelope, I enabled the graph option so I could graphically see the function of the low pass on the signal flow. Now that my low pass filter was set up, I have created an LFO to modulate my ladder filter's cut off parameter. From my *LFO* module, I have

created two knobs so I can control the frequencies and amplitude of my LFO. In order to connect this module to my ladder filter, I needed to connect my LFO's sine output to the input of my ladder filter. But the ladder filter was already connected to the *Cut Off* module and I did not want to loose that connection. The way around the issue was to create an *Add* module and connect the ladder filter's cut off to the first input and the sine output of my LFO to the other input. Then both of the signals were summed in my *Add* module and could be connected to the input of my ladder filter without loosing any of the connections. At this point my LFO modulation was working well but I did not have any filter envelope yet.

To create my filter envelope, I used the ADSR envelope, just like I had done earlier to control my amplitude when dealing with my oscillator section (Crute, 2019). After loading my *ADSR* envelope module, I created controls for the attack, decay, sustain and release. For the module's gate input, I created a midi trigger to ensure that the filter envelope would receive a midi signal. To create the connection between my filter envelope and my ladder filter I had to load an *A to E* module that would convert audio-rate signals into control rate (Milly March, 2014). However, I was still missing a control for the filter modulation's depth. Therefore, I connected my *A to E* module to a *Multiply* module with a depth control on one of the inputs before making the connection to the *Add* module, resulting in the completion of the signal path going into the ladder filter (Appendix; Fig. 4). After finishing my low pass filter, I decided to create a simple high pass filter with the aim of making my filtering section more versatile.

To build the high pass filter, I had to create a *HP/LP 1-Pole* module, which functions as a dual mode 6db/octave filter. I added a control to the *HP/LP 1-Pole* module to operate as the filter's control knob on the visual interface of the synths

(Appendix; Fig. 5). Inspired by the method used in the Juno's 60 and 106, I routed the high pass ahead of the low pass filter in the signal chain (Rishabh, 2016).

As a last feature, I thought it would be convenient to get a master fader which would control the overall output signal of the instrument. This is why I added a *Multiply* module at the end of the signal chain and attached a control to it, named master. The control initially appeared as a knob control but I changed it to display as a fader on the visual interface of the synthesiser.

At this point, my synthesiser as an instrument was finalised (Appendix; Fig. 6), however, I was disappointed with the result. The sounds I was able to create with it, were neither inspiring nor exciting to me. This is when I remembered that the dry signals coming out of synthesisers were never really appealing to me and that I got inspired by synthesizers once I started to route them through different effects. Consequently, I decided to create basic distortion, phaser, delay and chorus units to improve the sound palette of my synth.

Effects

Distortion

In order to fabricate a basic distortion unit, I had to create the core of the effect with a *Clipper* module. The clipper acts like a threshold and will make the signal distort if it reaches a certain value. To simplify the unit, I decided to set the threshold values to a maximum of 0.4 and a minimum of -0.2. For my distortion unit to be usable, I needed to create a dry/wet control knob so the user could mix the dry signal of the synth with the wet signal of the distortion unit. For this purpose, I created a *Crossfade* module where the 00 input was getting the dry signal from the unit's input while the 01 input of the module was receiving the wet signal from the Clipper's output (Appendix; Fig. 7). In order to make the unit more practical, I created a *Switch*

module and named the first input “On” and the second one “Off”. To make the switch work properly, I needed to make sure the signal could still go through the effect when the switch was turned off. In order to do so, I routed the signal from the unit’s input straight into the “Off” input of the switch and linked the crossfade output to the other input. Since the switch was getting both signals, the unit was able to turn on and off smoothly (Rishabh, 2016).

Stereo Delay

To set up my stereo delay, I created a *Single Delay* module with a control to tweak the time of the delay. Then I routed the output signal of the delay into a *Saturator* module, which would act as a limiter to prevent the delay from getting out of control. In order to set up a feedback loop for the delay, I routed the output of the Saturator into the first multiply input of a *Multiply/Add* module and routed the output back into the input of the Delay. I then created a knob on the second multiply input of the module to control the feedback of the delay and routed the input of the delay unit into the add input of the *Multiply/Add* module. Since I wanted a stereo delay, I decided to fake the stereo effect by sending the unit’s right signal into a duplicate of my mono delay. However, I wanted my knobs to control the left and right side altogether, therefore I routed the delay time, dry/wet and feedback controls of my left delay into the appropriate inputs of my right delay (Appendix; Fig. 8). When it came to creating the dry/wet control, I simply repeated the same steps I had done for my distortion unit (Dubspot, 2013).

Flanger

Matt Piper from Strymon effect wrote (2018): “The flanging effect is created by mixing two identical audio signals, applying a slight delay to one signal and slightly varying the delay time”. In other words, the core of a flanger is based upon an actual delay effect. Therefore, to set up my flanger unit I had to load a *Single Delay* module

and create two audio signals, a feedback and a feedforward. To create the feedback signal, I multiplied the delay output by a value smaller than 1 and then added back this signal into the input of the delay. To create the feedforward signal, I simply routed the output of my *Multiply* module into an *Add* module. However, to create the bulk of the flanger I needed to create a dry signal in parallel of the delayed signal and addition both of these signals at the end of the chain. Consequently, I routed my input signal directly into a *Multiply* module with the same set value as the wet signal's module and then routed both signal into an *Add* module. At this point most of the flanger was set up but I still needed to create a modulation.

I decided to use a sine wave oscillator to modulate the delay time. However, to make a sine wave behave like an LFO, it needs to have a much lower frequency range than it normally would (March, 2014). Consequently, I had to convert my frequency value into a pitch value with a *Log F* module and then, create a frequency control to modify its range. Finally, I routed the signal of my modulation unit into the input of the delay.

However, since a sine wave oscillator outputs an audio signal in the range of -1 and +1, the modulation could not work properly because negative values do not work with delays. The way around that issue was to route an *Add* module, with a constant's set value of 1, between the oscillator and the delay. As a consequence, the delay time got modified with a range between 0 and 2 milliseconds (Appendix; Fig. 9). As a final step, I added a dry/wet control for the unit (ADSR Music Production Tutorilas, 2013).

Stereo Chorus

A chorus effect is created when an audio signal is mixed with multiple copies of itself, slightly delayed and pitch shifted, giving the sound source a wider and fuller sound (March, 2014). To make the effect, I had to pitch shift the copies of the audio

source with an LFO modulating the delay time up and down. To create my LFO modulation, I used a sine wave oscillator, which is the exact same method used earlier to achieve the flanger effect. The only difference is that in this case, my sine wave oscillator's amplitude was controlled by a knob and not by a constant. Moreover, to make the modulated delay time more interesting, I decided to add to each sine wave an inverted version of itself by using an *Invert* module routed into an *Add* module. In other words, the *Add* module receives the signal of an inverted sine wave as well as a dry sine wave. In order to correct the negative value that the sine waves were outputting, I added a constant on my *Add* module and routed the module into a diffuser delay with a control knob attached to control the diffusion. I chose to use a *Diffuser Delay* module because it adds more depth to the sound than a standard delay. Since I wanted to have a really big sounding chorus effect, I duplicated my chorus four times so I had four delays controlled by four sine waves. In order to make my chorus stereo, I added the top two choruses into one *Add* module going into the left output of the unit and the bottom two choruses into another *Add* module going into the right output of the unit. As a consequence, the left and the right signal were different from each other, which is actually the result I was looking for (Appendix; Fig. 10). The last feature I incorporated was the dry/wet control knob that I created by using a *Crossfade* module (March, 2014).

Issues and Improvements

The main issue I never managed to resolve was the creation of a step sequencer for my synthesiser. I created a couple of different step sequencers but for some reason, I never managed to route them properly to my instrument (Appendix;

Fig. 11a&b). All of them are still into the instrument structure but are deliberately absent from the instrument's panel (ADSR Music Production Tutorials, 2013).

One of the main improvements I made to the overall synthesiser interface was to add switches to each oscillators and effects present on the panel so the user could engage or disengage anything as quick as possible. At first, my synthesiser did not have any switches for the different parts of the synth, which is not very practical in terms of workflow when, for example, the user wants to create a sound with only one or two oscillators.

The wave shape selector present on each oscillator is another improvement I made on the interface. On the first version of my synth, each oscillator featured a shape knob that had the function of mixing the signal of the different waveforms inside the oscillator. The system was impractical since the knob did not give the user any information about which waveforms they were using. Moreover, I preferred selecting a set waveform for oscillators and mix their signals together with volume knobs, just like the synthesiser Massive from Native Instruments works (Monalichi, 2011).

The last improvement was made on my delay unit. The first delay I set up did not let the dry signal of my synthesiser go through the effect. Consequently, when the delay was engaged, only the wet signal was outputted, which was compromising the playability of the instrument. The way to fix this issue was really straight forward since I only had to route the input of the delay unit directly into the "Off" input of my switch so when the effect would be turned off, the signal could still go through.

Design and Visuals

When creating a synthesiser, it is indeed important to take care of the technical aspect in order to ensure that the product is highly functional. However, having a functional product is not the only reason why producers choose one synth over another for their production. The interface, the ease of use and the visuals are equally important to make a synth popular on the market (Stephenson, 2019). Since I had already improved my synth in terms of interface, I had to focus on the visuals to make it appealing to users. After making some research on the subject, I found out through scientific studies that the human brain retains information much more easily if the information is visual. That is the reason why successful commercial products have beautiful visuals displayed on their packaging; the goal is to hook potential customers/users instantly without giving any explanation or saying any words (Stephenson, 2019). In the world of software, the packaging is the actual shelf of the product, which in this case is the control panel of the synthesiser.

I personally always loved plug-ins companies, such as Dada Life (2020), designing funky visuals for their products. As a consequence, I decided to base the design of my synthesizer around one of my favourite childhood game: Super Mario. However, since in Reaktor the designs options are limited to background pictures and colour choices for buttons/graphs, I had to think creatively. Therefore, I chose pictures in PNG format because they have the ability to merge themselves into any background, creating a nice effect of superposition. However, I quickly realised that in Reaktor it is impossible to reframe pictures to the user's desire and as a consequence, it is extremely difficult to position the pictures in a coherent manner on the panel. In order to realise my vision, I had to firstly find a background picture that fitted more or less the size of the synthesiser panel. Secondly, I had to download

really small pictures with a dimension of 250 by 250 to fit inside my oscillators and then adapt the controls' placement in relation to my pictures. In other words, the design options were so limited that I had to adjust the placement of my controls to fit the picture instead of using the pictures to fill the space between my controls. However, after some experimentation, I found out that I could use buttons created in my instrument structure to frame my pictures even if they are not routed to anything in the unit. This finding helped me enormously to get my design right. In terms of picture selection, I decided to use iconic objects of the Mario series such as the star or the fire flower but made sure to avoid pictures of the main character. I believe that this is a more subtle choice and consequently gives the product a stronger identity (Appendix; Fig. 12a). Since my effects unit was created specifically for The Super Mario synth, I decided to keep the same approach for their design (Appendix; Fig. 12b).

Conclusion

In this essay I walked you through my process of building The Super Mario synthesiser from scratch. I started with a core of three oscillators controlled by an envelope and went on to develop a fully functional filtering section with four different effects. During the process, I stumbled upon a lot of issues and even if I could not find all the answers, looking actively for a way around each problem made the process a valuable learning experience. I learned about the different waveforms, the role of the diverse modules in the synthesiser, the principle of signal routing and most importantly, I understood the basic logic behind modular synthesis. This means that I can now use all this knowledge to create my own soft-synth pre sets, get my hands around analogue synthesizers more easily and experiment with modules to create

other units. These are valuable skills that will make me feel more at ease with sound design and as a consequence will strengthen my identity as a producer.

An interesting future study would be to replicate classic analogue synthesisers in the Reaktor digital's environment to see how close we can emulate analogue units with digital tools. It would also reveal the characteristics of these synthesisers that marked the history of music making since the 60's.

References

ADSR Music Production Tutorials (2013), “*Reaktor – How to Create a Flanger – How to Tutorial*”, available at: <https://www.youtube.com/watch?v=Dy7LMlgKFd0> (Accessed: 6th April 2020)

ADSR Music Production Tutorials (2013), “*Reaktor – Creating an Arpeggiator Device from the Ground Up – How To Tutorial*”, available at: <https://www.youtube.com/watch?v=mORR7WuuVZU> (Accessed: 6th April 2020)

Christ, R. & Wernli, R. (2014), “Pulse – Width Modulation”, *The ROV Manual*, 2nd edition, available at: <https://www.sciencedirect.com/topics/computer-science/pulse-width-modulation> (Accessed: 6th April 2020)

Comeau, J. (2020), “Let’s Learn About Waveforms”, *The Pudding*, available at: <https://pudding.cool/2018/02/waveforms/> (Accessed: 6th April 2020)

Corman, E. (2015), “Simple Synthesis: Part 6, Pulse Width Modulation”, *keithmcmillen.com*, 22 May, available at: <https://www.keithmcmillen.com/blog/simple-synthesis-part-6-pulse-width-modulation/> (Accessed: 6th April 2020)

Crute, A. (2019), “All You Need To Know About Subtractive Synthesis”, *MusicTech*, 4 June, available at: <https://www.musictech.net/guides/essential-guide/what-is-subtractive-synthesis/> (Accessed: 6th April 2020)

Dada Life (2020), “Our Plugins “, *dadalife.com*, available at: <http://www.dadalife.com/sausage-fattener/> (Accessed: 6th April 2020)

Dubspot (2013), “*Sound Design Tutorial: How To Create A Delay Effect w/ Reaktor - Native Instruments*”, available at: <https://www.youtube.com/watch?v=sMKGSaO20zY> (Accessed: 6th April 2020)

Electrical4U (2019), "Oscillators: What Are They (Definition, Types & Applications)" *electrical4u.com*, available at: <https://www.electrical4u.com/what-is-an-oscillator/> (Accessed: 6th April 2020)

Makes, L. (2019), "What Exactly Is A Modular Synth?", *AttackMagazine*, available at: <https://www.attackmagazine.com/technique/technique-modular-synthesis/what-exactly-is-a-modular-synth/> (Accessed: 6th April 2020)

Mantione, P. (2017), "The Basics of Synth Envelope Parameters, Functions and Uses", *Sonarworks*, 16 July, available at: <https://theproaudiofiles.com/synthesis-101-envelope-parameters-uses/> (Accessed: 6th April 2020)

Milly March (2014), "*How to build a stereo chorus effect in Reaktor - Synthesis and Sampling Coursework*", available at: <https://www.youtube.com/watch?v=uEFYZPUU6HY> (Accessed: 6th April 2020)

Monalichi (2011), "*Reaktor 5 Tutorials - Tutorial 2 Part 2, Selecting Different Waveforms*", available at: <https://www.youtube.com/watch?v=F531Lr-O9fl> (Accessed: 6th April 2020)

Piper, M. (2018), "What is a Flanger", *Strymon*, 1 February, available at: <https://www.strymon.net/flanger/> (Accessed: 6th April 2020)

Rise, S. (2020), "Noise", *synthesizeracademy.com*, available at: <http://synthesizeracademy.com/noise/> (Accessed: 6th April 2020)

Rise, S. (2020), "Voltage-Controlled Amplifier (VCA)", *synthesizeracademy.com*, available at: <http://synthesizeracademy.com/voltage-controlled-amplifier-vca/> (Accessed: 6th April 2020)

Rishabh, R. (2016), "A Guide To Building Synths in Native Instruments Reaktor", *Ask.Audio*, 19 April, available at: <https://ask.audio/articles/a-guide-to-building-synths-in-native-instruments-reaktor-part-1> (Accessed: 6th April 2020)

Sweetwater (2006), "Noise Generator", *sweetwater.com*, 14 June, available at: <https://www.sweetwater.com/insync/noise-generator/> (Accessed: 6th April 2020)

Stephenson, A. (2019), "The Power Of Packaging Visuals", *explorerresearch.com*, 8 May, available at: <https://explorerresearch.com/the-power-of-packaging-visuals/> (Accessed: 6th April 2020)

Appendix

Fig. 1: Oscillators A & B

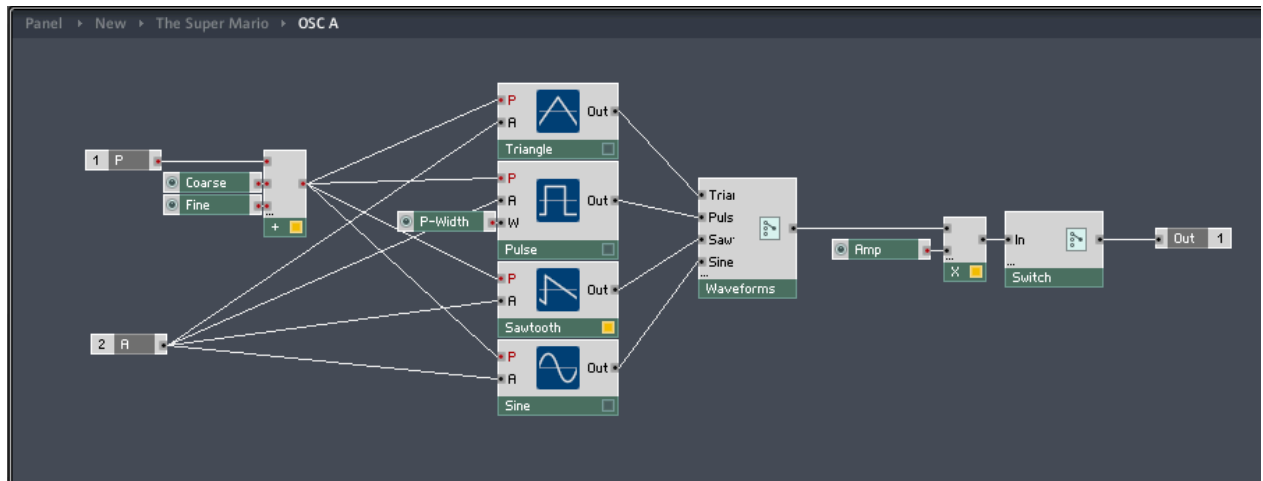


Fig. 2: Oscillator C

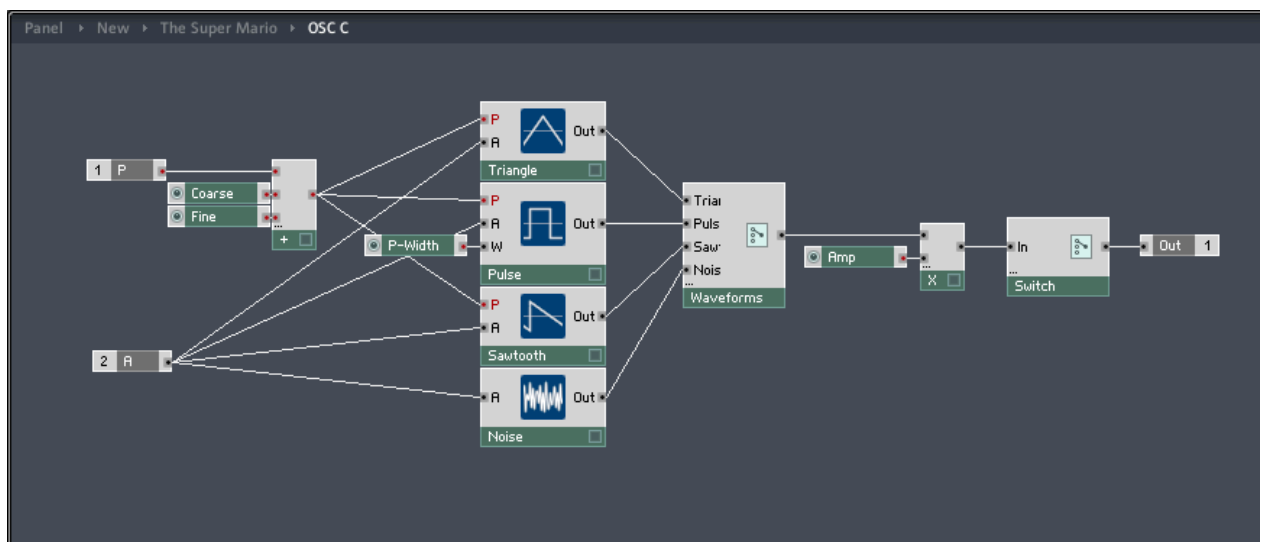


Fig. 3: Oscillators with ADSR Envelope

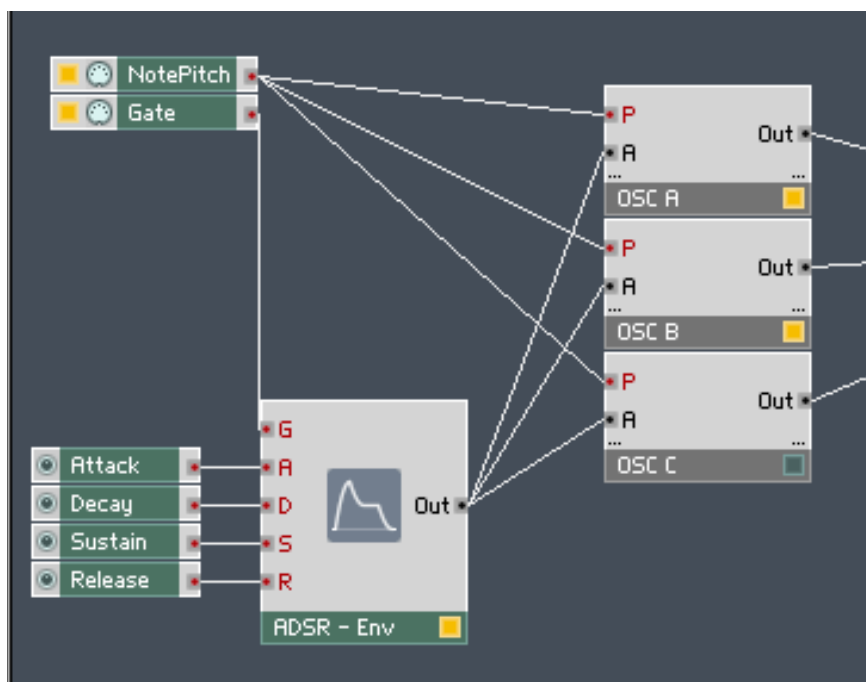


Fig. 4: Low Pass Filter with LFO and Filter Envelop

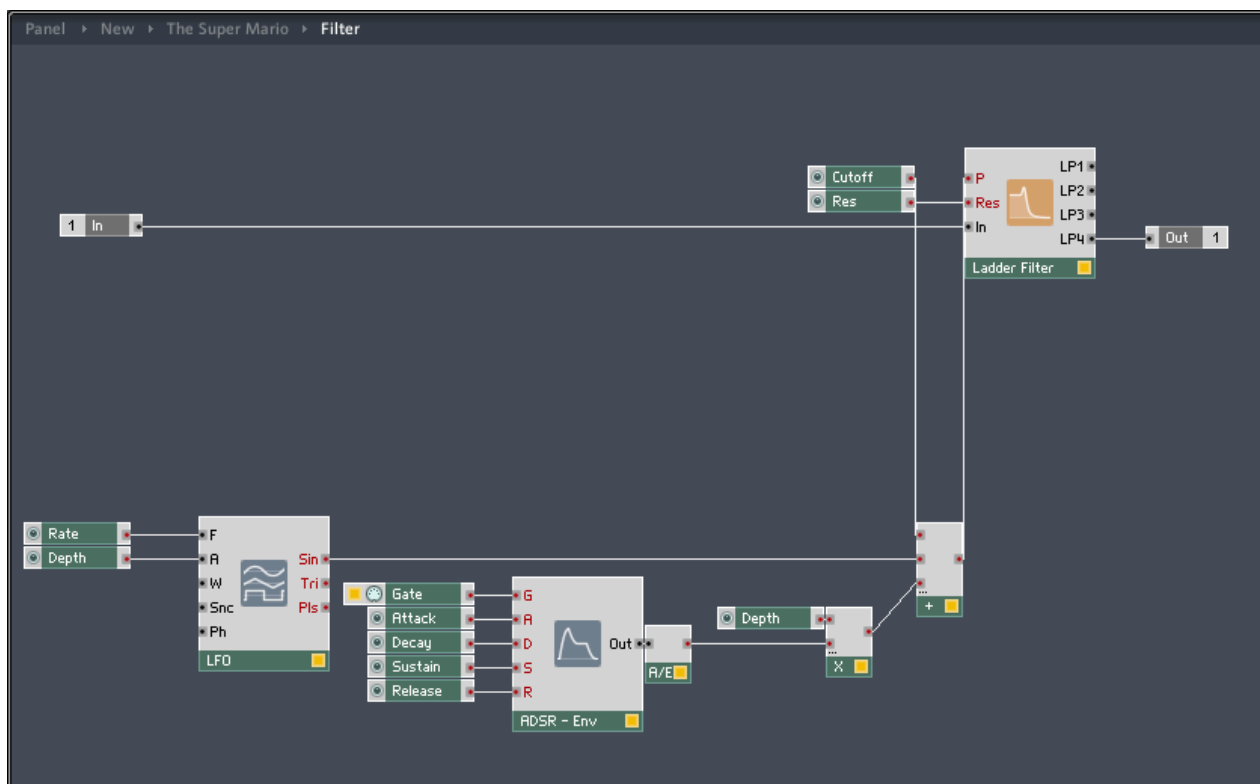


Fig. 5: High Pass Filter

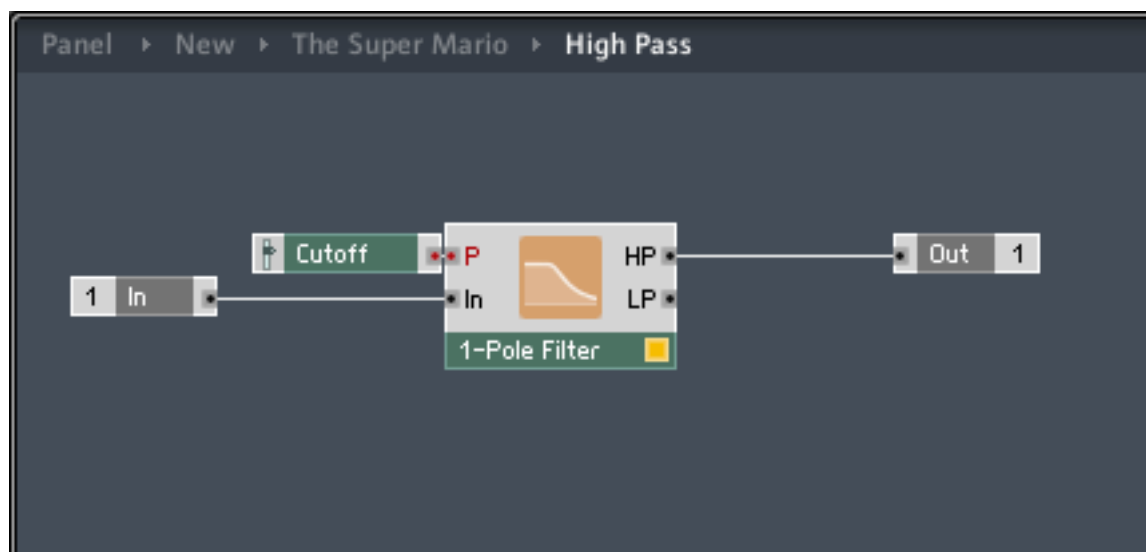


Fig. 6: Synthesizer without effects

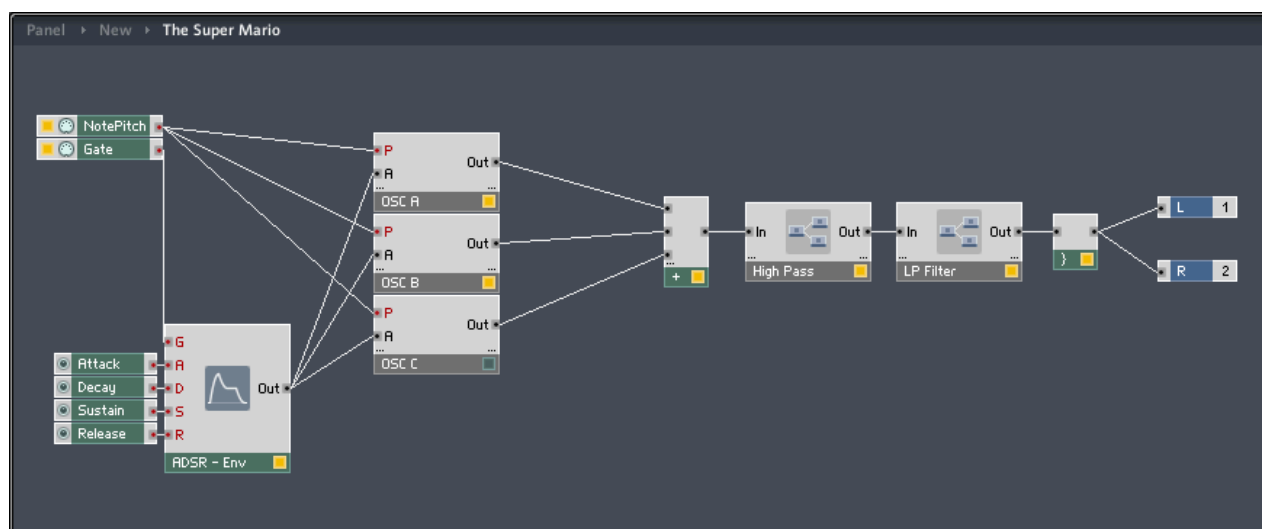


Fig.7: Distortion Unit

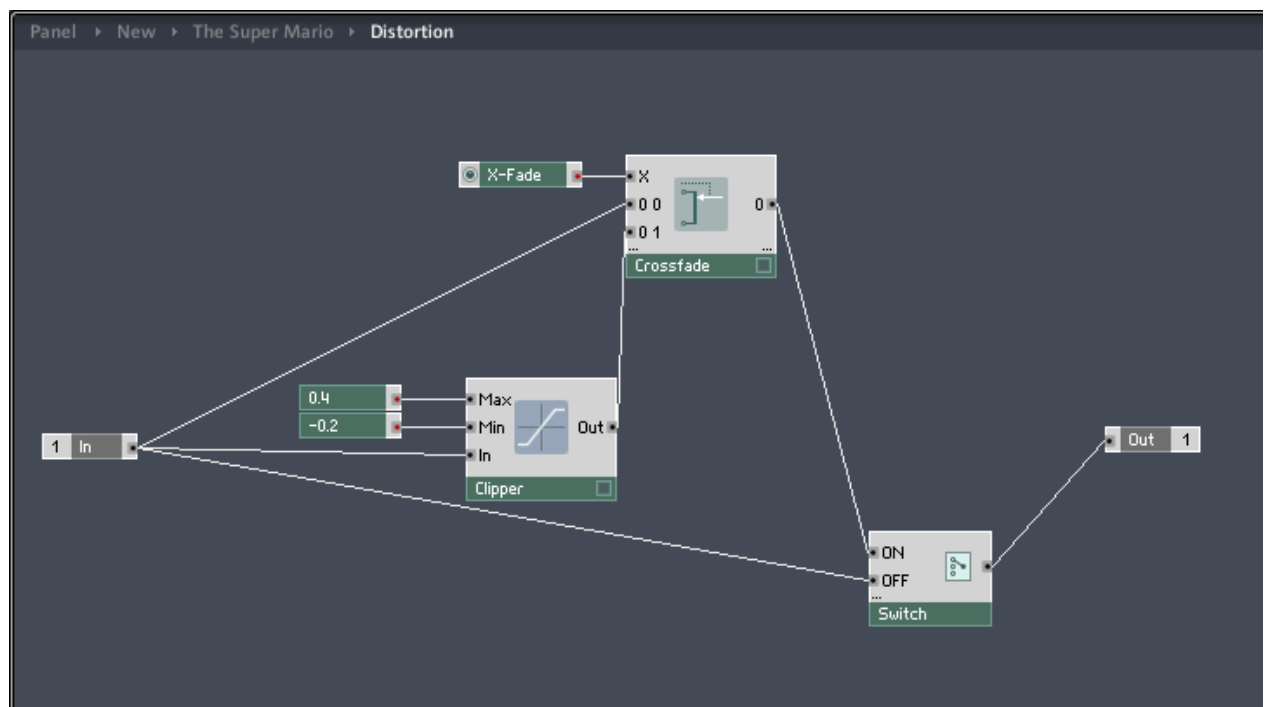


Fig. 8: Stereo Delay Unit

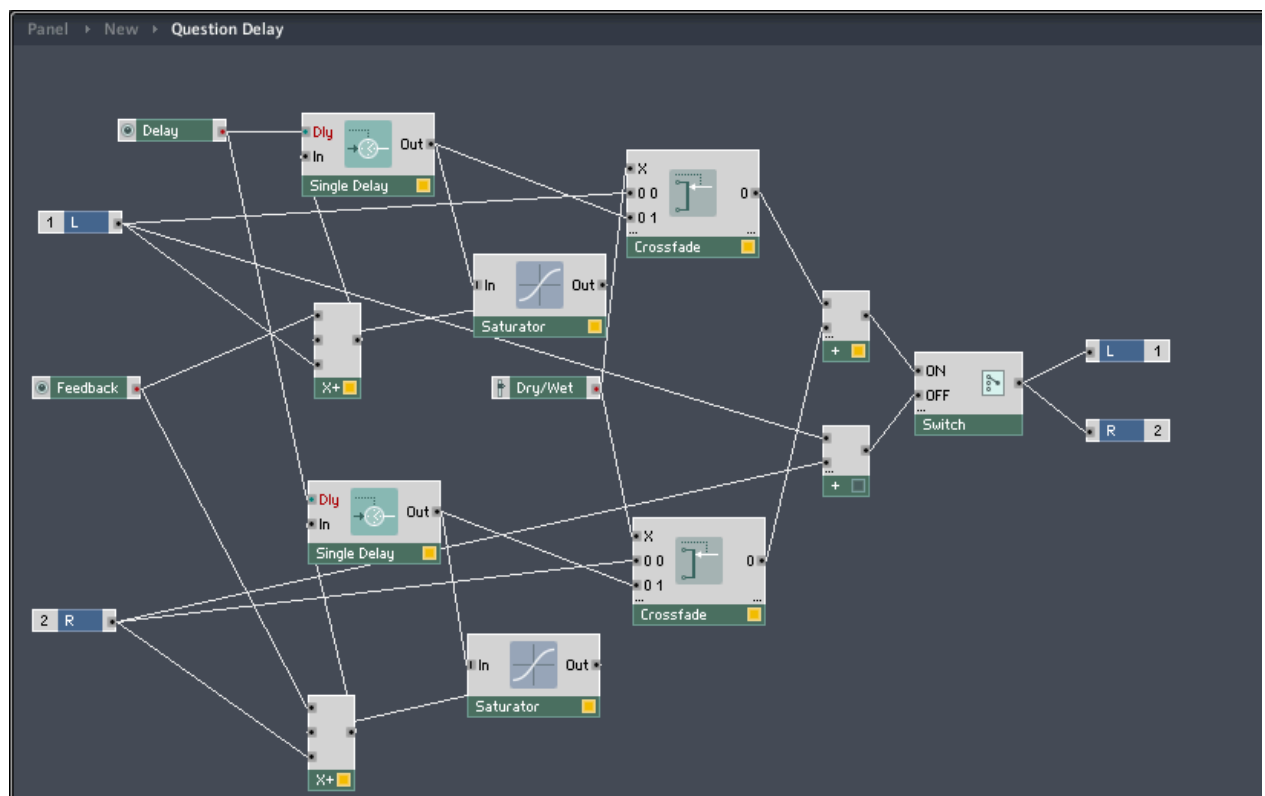


Fig. 9: Flanger Unit

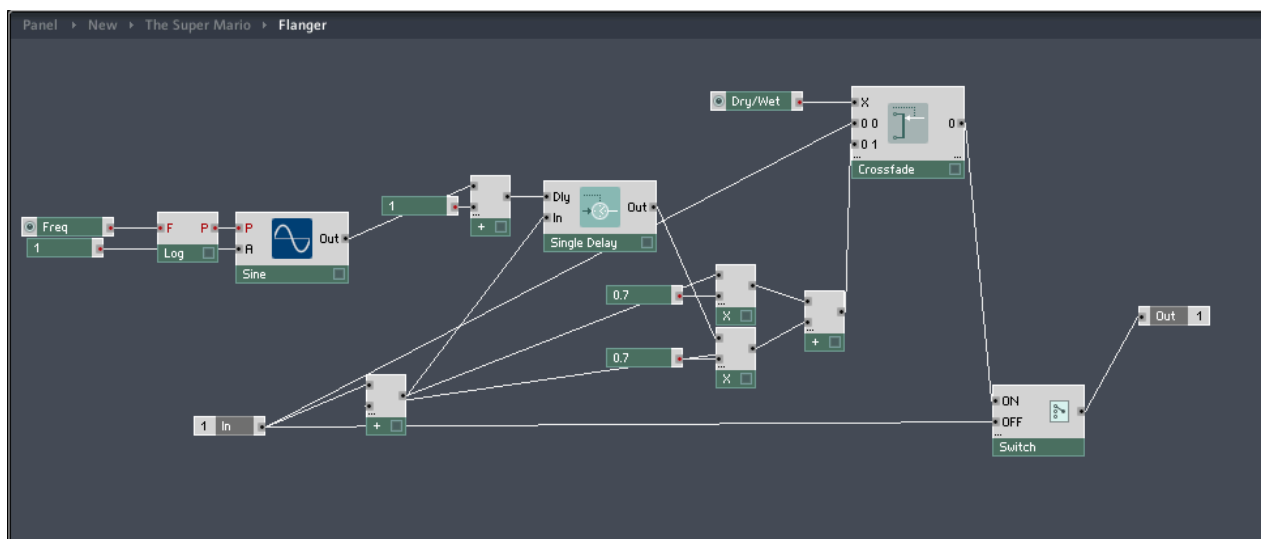


Fig. 10: Stereo Chorus

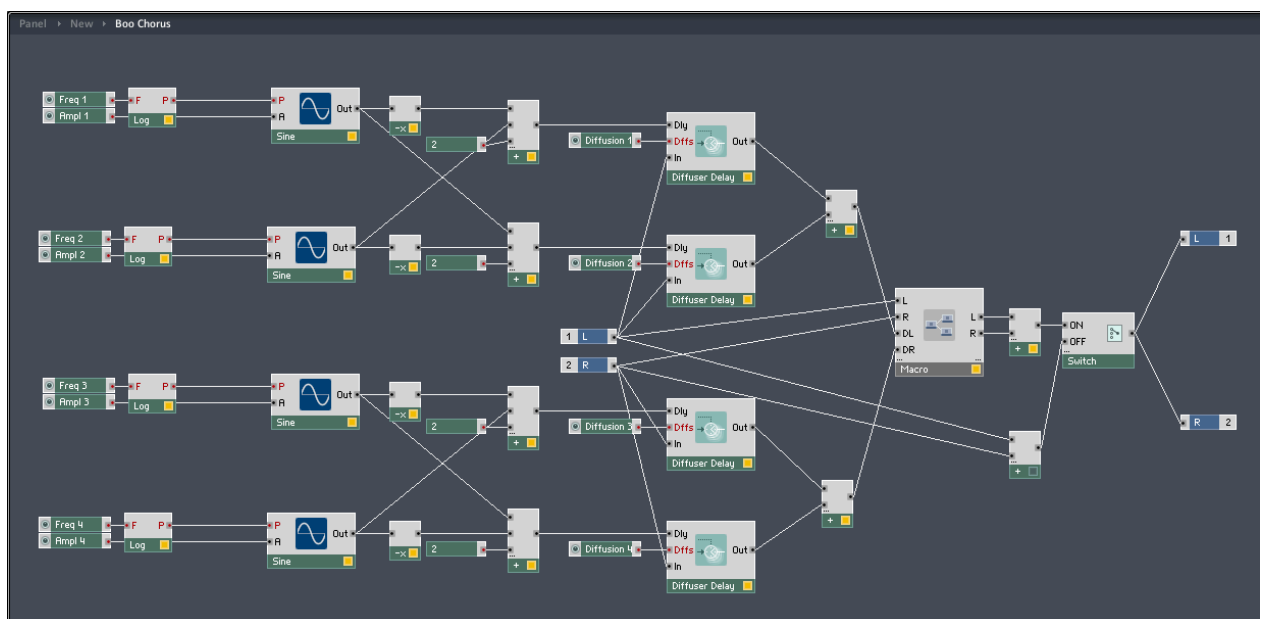


Fig. 11a: Arpegiattor Structure

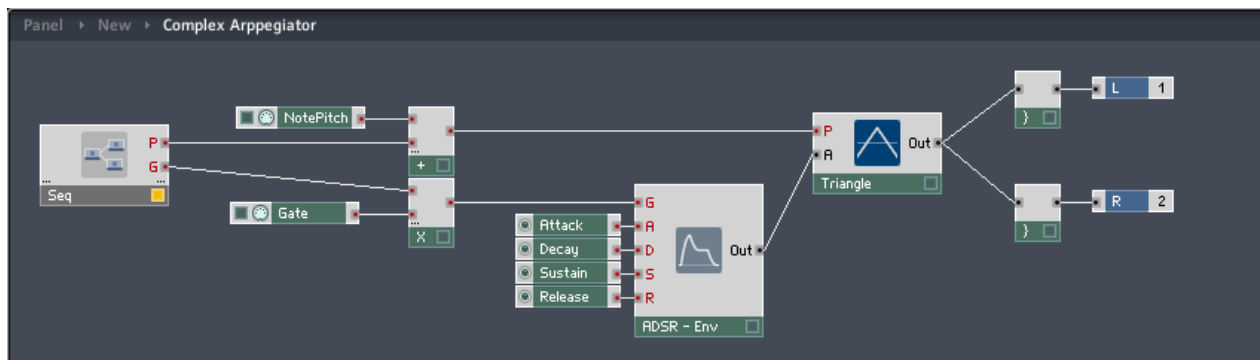


Fig. 11b: Sequencer Structure

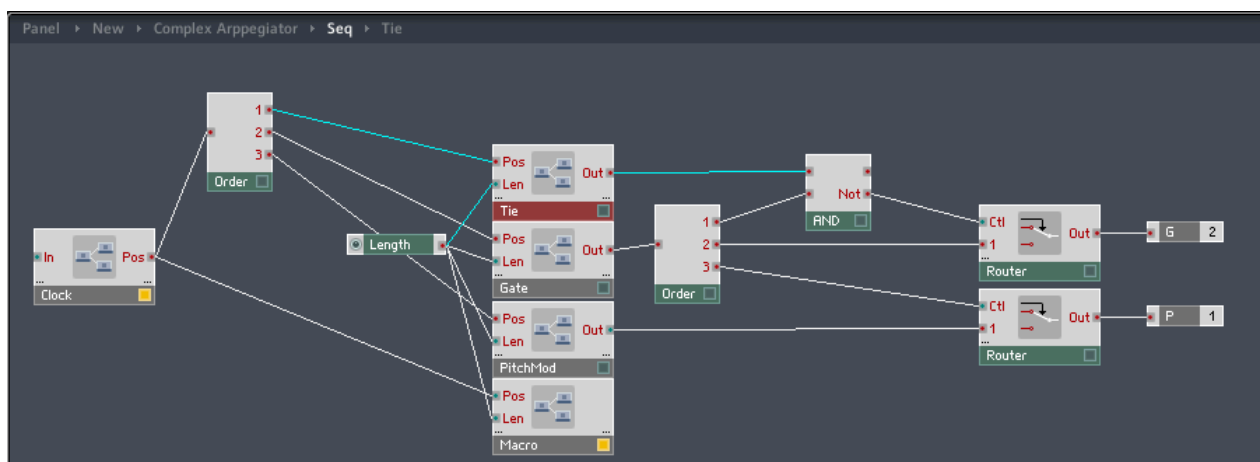


Fig. 12a: The Super Mario Synth Design



Fig. 12b: Effects Design (Boo Chorus & Question Delay)

