COMPSYS701 Advanced Digital Design 2021
Individual Research Project

Nikhil Kumar - Nkmu576

1 Introduction

Signal processing is the application of various signal processing techniques or algorithms to analyse, modify or synthesize a signal. These signals can pertain to a wide variety of fields and applications, and are done to extract information or provide clarity to incoming signals or data.

In the case of the application specific processor for the Heterogeneous Multiprocessor System on a Chip, the signals which have been sampled are within the frequency in the audio spectrum. These signals can be represented in multiple ways and can be a single tone or an entire piece of music.

This has led my research into implementing various Signal processing algorithms which pertain to digital audio processing techniques. This is the most common form of digital signal processing and is very familiar to nearly everyone, such as filtering signals using an EQ. The techniques and the outcomes from processing can also be said to be more human understandable as audio signals are something that humans interact with everyday.

Some algorithms for Digital Audio signal processing and audio effects include:

- Filtering and equalisation
- Artificial Reverberation and Echo
- Dynamics control
- Audio Synthesization
- Amplitude Modulation
- Stereo imaging

From these select effects I have chosen to implement Artificial Reverberation and Echo.

2 Research

2.1 Reverb and Echo

Reverberation or Reverb is the persistence of a sound after the sound has since been created. This is often in the form of repetitions of the sound, often with decreased volume, being heard long since the original sound. The phenomenon is due to the sound waves bouncing off the environment and travelling back to the listener, often with some delay due to the time taken to travel through the

environment. [1] An Echo is a similar phenomenon, however, describes only one repetition of the original sound, rather than multiple over a period of time. As such, Reverb and Echo have been mainly associated with modelling of room acoustics.

Reverb and echo audio effects are in very common usage in many audio effects packages. This ubiquity has led to a multitude of various algorithms and implementations of the effect. The general approach of these algorithms is to try and model how audio will interact in a certain environment.

2.1.1 Implementations of Reverb

2.1.1 Complex Reverb and Echo implementations

As the goal of reverb is to model certain room acoustics, one approach is to convolve the signal with a known impulse response of a room. This known impulse response is often an audio recording of an audio impulse in an environment. This convolution of the two signals can be realised using a Finite impulse response filter. Each sample of the impulse response of the target room can be used as a coefficient of an FIR filter. [2]

However, often this is very impractical, as an impulse response of a room can be lengthy, depending on the environment, and thus recordings of impulse responses can last seconds. With a sample rate of 48.8Khz, this would lead to 48800 samples for 1 second. Any reverberation or echo effect may last several seconds, and thus is not feasible due to the high memory and arithmetic unit cost, as well as having a recording of a room impulse to use for convolution.

Another implementation is to model a room mathematically by using Finite Element analysis to mathematically model a room and how sound waves interact inside the room. However, this is far too computational intensive, and thus is out of scope for the project.

With these implementations requiring far more resources as well as time and expertise to implement, the choice was made to use Delay based Reverberation techniques.

2.1.2 Delay based Reverb

As Reverb is the persistence of a sound, after the sound has been produced, and with a delay since that sound has been created, a classic method of creating reverb or echo was to simply use a delay on the incoming signal, to be outputted later with the current incoming audio. These techniques are realised with the use of memory blocks, which hold onto a piece of incoming audio, to then be output into the incoming signal later.

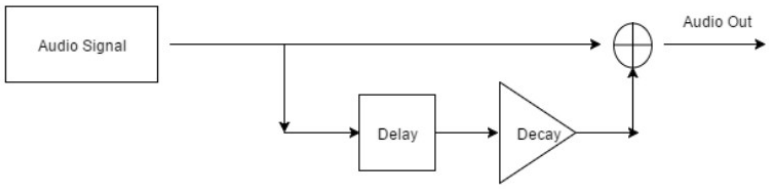


Fig 1. Simple Representation of Delay based Reverb

Delay based reverb can come in many forms, however is often built on a Recursive Comb Filter [5], which is a form of infinite impulse response filter. These filters have a delay line, which takes in a

sample, stores it into memory and then outputs the sample after a certain amount of clock cycles or seconds. The output is often fed back to the input signal.

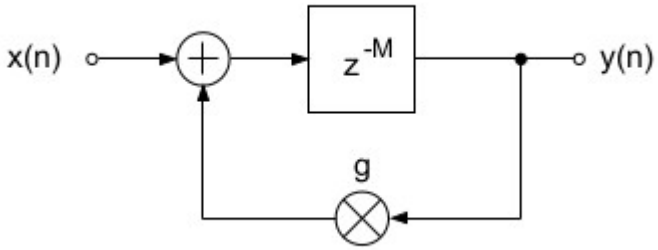


Fig 2. Block diagram of Recursive Comb Filter

Due to the positive feedback of the system, the stability of the system is reliant on the g factor, which relates to the gain of the feedback signal. The gain of the feedback must not exceed 1, as the system can easily become unstable due to increased gain and summation per sample[3].

This filter, as opposed to a feedforward version of the comb filter, provides the exponential decay required to have reverberation. A sample will be delayed before being outputted, thus satisfying the delay criteria of a reverb, and then the sample is gained down and added back into the input signal, thus subsequently going through the process again, to be gained further and further down. This satisfies the exponential decay often associated with reverberated sounds. Comb Filters often have a frequency response which resembles a comb, antennuating and gaining various frequencies with notches. This means that various frequencies are close to resonant frequencies and thus, if the system parameters are not chosen correctly, the overall system gain can reach more than 1. Thus causing instability. Another issue with Comb Filters, is that due to the gain on certain frequencies, high frequencies tend to also be accentuated, and thus the resulting reverberated output tends to sound more “metallic” as higher pitch frequencies are present in the signal. Generally, high frequencies tend to decay quicker in environments, as the subsequent bounces of the signal off the environment absorb these frequencies. This is why Recursive comb filters sound more “metallic” and unnatural to real reverberation [2].

2.1.3 All Pass Filter

All-pass filters are characterised by the flat frequency response produced by the filter. Similar in design to the recursive comb filter, an All-Pass filter provides a feedforward path that bypasses the delay, and sums input samples with the delayed samples, as well as providing the feedback and decay necessary for exponential decay of the reverb sample. All-pass filters, as opposed to comb filters, also have less latency between inputting a sample and hearing the output of the sample, due to the feedforward loop of the all-pass filter. The flat frequency response of the filter also means that reverberations sound less “metallic” than the recursive comb filter [2].

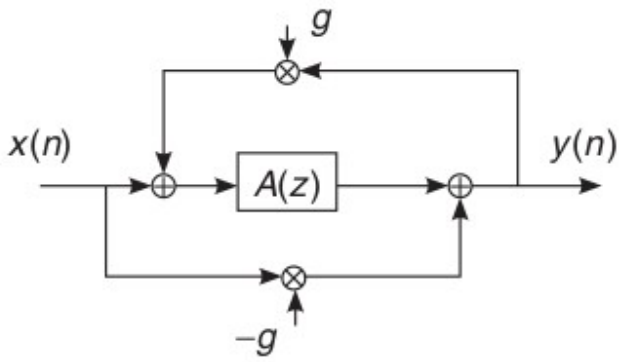


Fig 3. Block diagram of All Pass Filter

2.1.4 Higher Order Reverberation Unit Designs:

The filters discussed previously are two key filters in Reverberation and Echo units. The filters are used in combination with each other to build a larger system, which can provide realistic simulations of reverberations in an environment. These designs are often set filters in cascading or parallel configurations to produce a higher density of delayed input signals in the system, with a subsequent higher number of feedback loops. This translates to more reverberations in the output signal, which can be desirable depending on the desired effect[6]. Many reverberation unit designs make use of multiple filters in different topologies, often with other forms of filtering placed between stages such as high pass filtering of the reverberated signal, as to mimic the higher decay associated with high frequency signals in a closed environment [7].

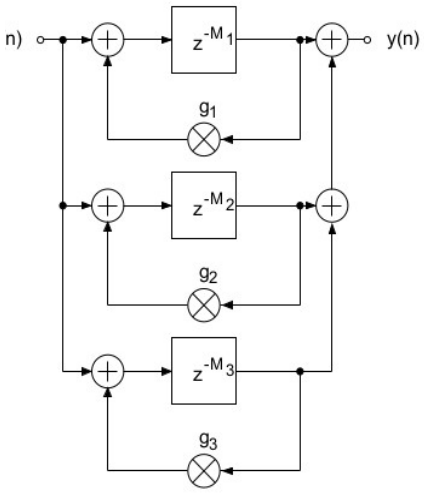


Fig 4. Block diagram of combinational filter reverb design

Generally, Reverberation units with more complex topologies, using multiple filtering stages, sound closer to real reverb/echo found in various environments. This is due to the higher density of signals that are in feedback in the system, which increases the density of reverberations in the output signal. This more closely mimics the behavior of reverberated signals in an environment.

Room simulation reverb units such as the embedded all-pass system [8], use multiple All-Pass filters.

The filters are cascaded with one another, and the outputs from each stage multiplied with a gain input and then summed together for the reverberated output and feedback loop.
The system allows for more precise control over the echo density and the reverberation time, as changing the gain for any number of the stages, changes the strength of the reverberated signal, and thus the reverberation time and density can be controlled.

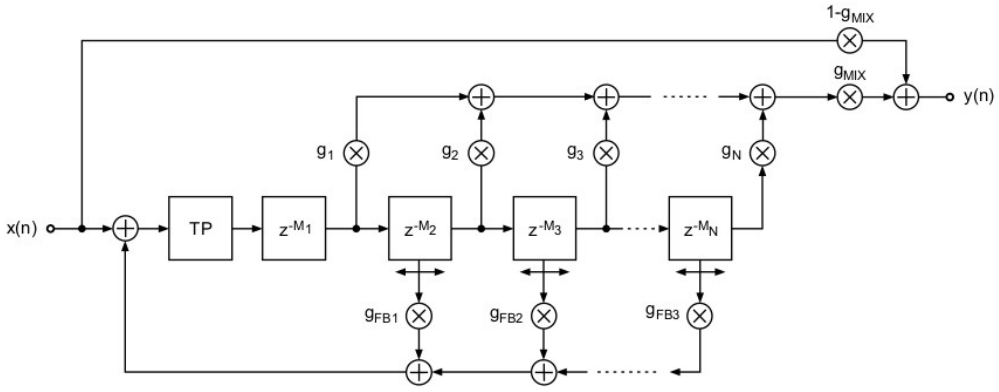


Fig 5. Diagram of Embedded All-Pass Reverb System

2.2 Reverberation designs chosen

The decision was made to implement a low pass Recursive Comb filter, an All-Pass filter, and a room simulator reverb unit which uses cascaded All-Pass filters. These three filters were chosen as the different filters provide contrasting and varied reverberations from different types of filters. The Low Pass Comb Filter makes use of a parallel delay line which implements a low pass filter. The subsequent feedback contains less high frequency elements, which in turn reduces the “metallic” sounding reverberation common in recursive comb filters. An All-Pass filter was chosen for its simple design as well as being a building block for more complex reverberation units, such as the third filter. The Room simulator reverb unit is based on the embedded all-pass system [8], which makes use of multiple All-Pass filters. The simulator system was chosen to provide a high quality reverb effect, which sounds closer to realistic reverberation without the use of any other additional complex filter designs.

3 Data Processor Design

3.1 Design

The ASP contains a control unit and a datapath unit. The control unit contains an FSM for decoding incoming network packets from the TDMA network, and sets the various control signals and parameters which are required to operate the datapath. The datapath contains the signal processing chain, which implements effects discussed previously. The datapath currently contains the three filters, units for decoding and reencoding left and right signal channels, and a mux for the output of each unit.

3.1.1 Control Unit

The Control unit decodes incoming packets from the TDMA network to configure the sending and receiving addresses, the parameter inputs for the various reverb effects and setting the control signals for operation of the datapath. The incoming packets and packet checking design are the same as the network protocol which has been implemented in the reference design for this project.

	Bits																					...	0
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	
Data-Audio	1	0	0	0	Dest				Reserved							Ch	16-bit signed integer						
Conf-DP	1	0	0	1	Dest				Next				Mode				Unused						
Conf-ADC	1	0	1	0	Dest				Next				SR		En	Ch	Unused						
Conf-DAC	1	0	1	1	Dest				Reserved				SR		En	Ch	Unused						

Fig 6. Network Protocol for reference design

The control unit contains an FSM which decodes the incoming instruction command. The incoming received data packet is checked for it is a data or configuration packet, by checking the first four bits of the incoming packet. If a configuration packet is received, bits 19 to 16 are checked for the mode input, which informs the ASP as to the operating mode that should be running. A case statement is used to check these bits to the related mode. Once the operating mode has been determined, the DPASP_State variable is set to the correct mode, which is used in another case statement. In this case statement, the DPASP_State is checked. Depending on the state of DPASP_State, the relevant control signals related to each operating mode are set.

The Conf-DP packets have been slightly modified to make use of the previously unused bits 15 to 0 to set the various parameters which are required to be set for the delay line size and gain input.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Gain 1 in		Gain 2 In		Delay Line Size Parameter											

Fig 7. 16 bit Configuration-DP packet modification

With these inputs, the configuration of the various filters can be modified by the command packet input.

The control unit also provides a valid data signal for the datapath and the various left/right filters. This signal is paramount to the operation of the datapath, as the system clock is far faster than the sampling rate of the ADC, and thus the incoming data must be validated to be a proper data packet which has been inputted into the system, as well as allow the correct delay line data input.

The control unit is responsible for sending the output of the datapath to the network. The current implementation is such that as soon as a packet is received by the network, the output from the datapath is then sent out to the network, with the address information that was present in the configuration packet. This implementation was chosen as it allows for simple synchronization between the input data and output data packets. As the sampling rate for an audio channel is 48khz, thus 96khz for both channel inputs, by sending out a data packet as soon as one is received, this adheres to the roughly 96khz sampling rate output of the Analog to Digital converter.

3.1.2 Datapath:

Left / Right Decoding
A unit inside the datapath decodes the incoming input samples as to whether the input is a left or right channel sample. This unit was devised as the filters require a signed sample input which contains no other data associated with it, such as audio channel information. This simplifies the rest of the datapath, as any subsequent operations conducted on the inputs can be done without having to check the channel tag bit, and exclude it from the operation.

Left/Right Reencoding
Present inside every effect unit block, is a unit which attaches a channel tag bit depending on which filter channel the sample was outputted from. This is then outputted to the output mux and then the rest of the network through the Control unit.

Reverberation Units
All implementations of the filters make use of two of the same unit, in parallel, which filters left and right channel audio respectively. Each generally consists of a delay line unit, as well as units which conduct arithmetic right bit shifts, and addition. The gains associated with each feedback or feedforward signal must be less than 1, to retain system stability, which in turn translates to division of the signal.

The delay line size and gain inputs are parameterisable, and thus the delay and gain for each filter can be adjusted based on the command input decoded from the Control unit. This in turn changes the response of the filters, and can provide a range of various reverberated signal times and density of reverberated signals in the output signal.

The delay line is a circular buffer, with synchronized read and write. The unit contains 16 bit memory of length 4096. The input signal of size into the unit is used to determine length of the delay line. In implementation, this means that the amount of the 16 bit memory used for the circular buffer is based on this size parameter. This means that for a given input size that is less than 4096, not all the memory elements are used. The Delay line is also clocked by the valid signal input which is produced by the Control Unit, to ensure that the incoming signal is a valid data sample, and not garbage values produced from previous operations. The ramifications of these design decisions are discussed later.

As bitwise division can easily translate into right bit shifts, the use of units which make use of shifting was opted for, rather than proper fixed point division. This choice is later discussed in the design justifications section.

3.1.2 Low Pass Recursive Comb

The implementation of the low pass recursive comb is a modified recursive comb with the use of a delay in parallel to the feedback loop of the filter.

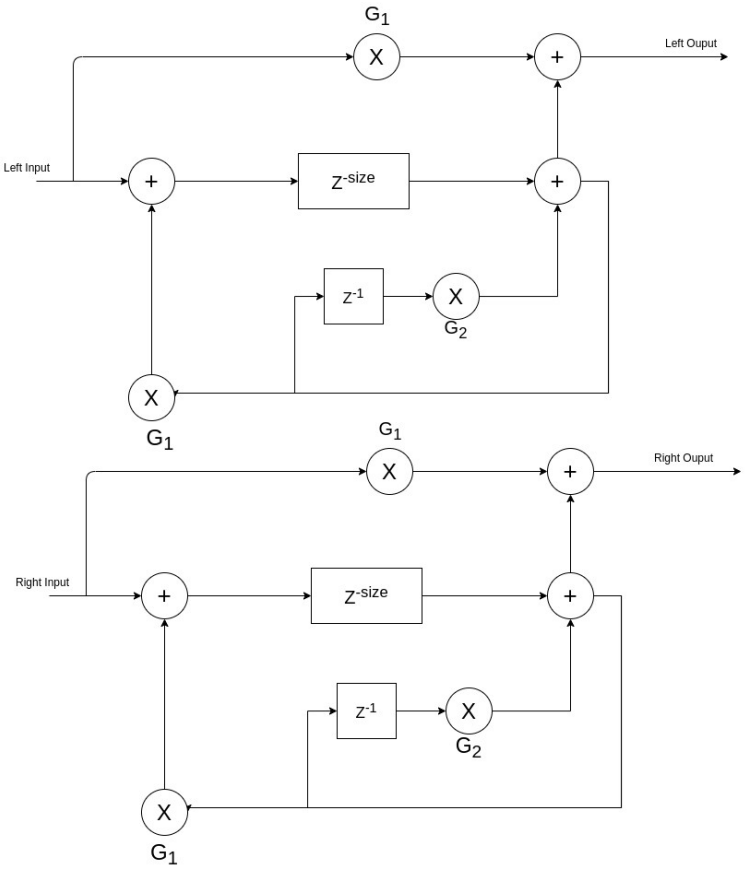


Fig 8. Implementation of Low Pass Recursive Comb

The use of the 1 sample delay line and gain G2 implements a low pass filter in the feedback loop of the comb filter. The resulting feedback signal contains less of the high frequencies that would otherwise be present in a normal recursive comb filter. The resulting output of the filter sounds more natural than what would otherwise be present in a non-modified filter.

3.1.3 All Pass

The implementation of the All-Pass filter is the same as the filter previously discussed, with the use of feedforward and feedback loops around a delay line. The design of the unit is simpler than the other two reverberation units present.

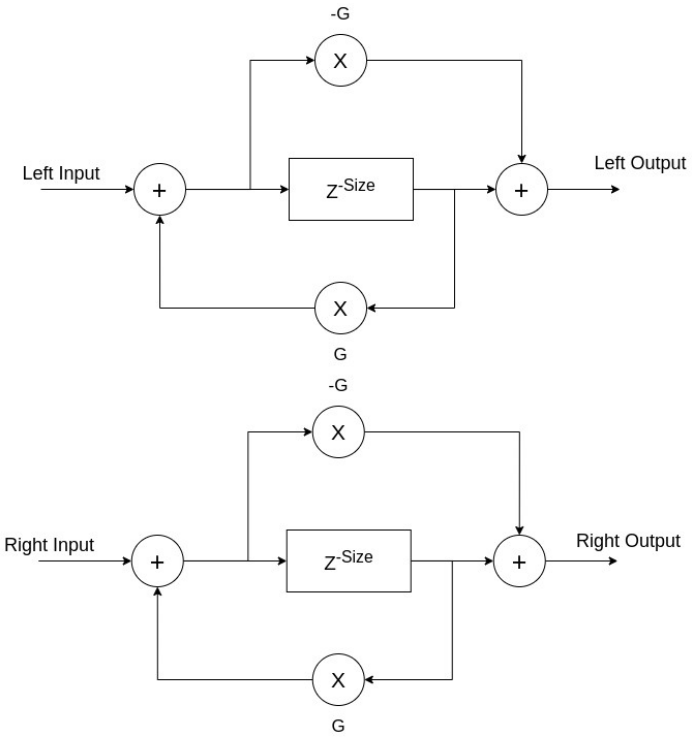


Fig 9. Implementation of All Pass Filter

As discussed the gain and delay can be modified by the command packets, and thus can provide a wide range of reverberation times and densities.

3.1.4 Room Simulation

The room simulation unit is the most complex unit present in the ASP. However, it's design is relatively simple to understand. The use of multiple All-Pass filters, which have been cascaded with one another is used. The resulting output from each stage of the All-Pass filtering is gained down, and then summated, before being summed with the input signal and outputted. This implementation allows for multiple reverberated signals, with the same sample being present in the output signal multiple times, not only through the feedback of the output, but also through the feedforward operation of the system. Combined with smaller gain values, the further in the process the sample is, the resultant output signal contains a high density of reverberated signals, which also contain early reflections of signals, which are present in real life reverberation of sound in an environment. The feedback present in the system allows for the previous iterations of reverberation which were present in the feedforward part of the system to be represented to the input of feedforward, thus providing further decaying reverberations.

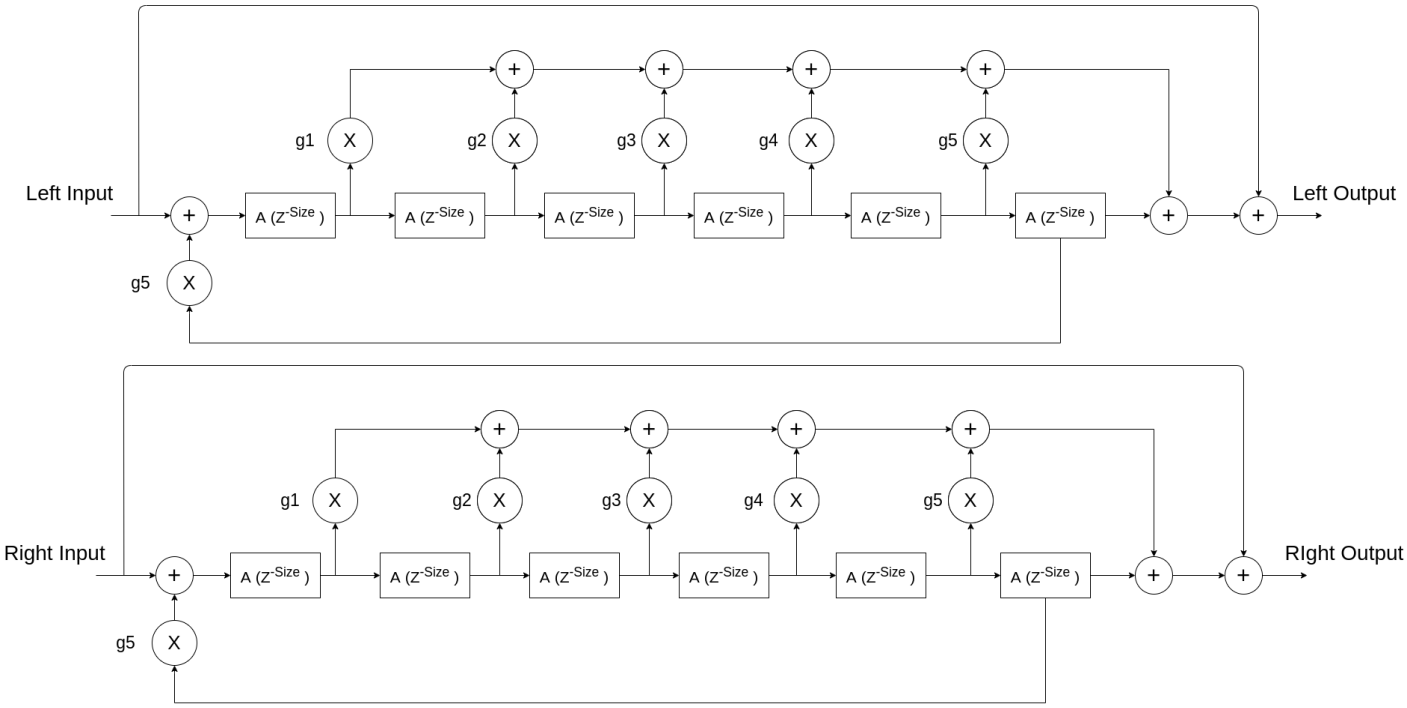


Fig 10. Implementation of Embedded All-Pass Room Simulator

The gains for each stage of filtering are chosen based on the gainset unit present in the room simulation block. The unit takes in a single gain input from the control unit, and then provides various gain values for each stage of the filtering process. These gains are multiples of the input gain, with the gains intended for the earlier stages of the process only being multiplied by one or two, which translates to samples being arithmetically right shifted by the input or double the input. Gains intended for the later stages of processing are multiplied by values of three or four, thus right shifts of three or four times the input. As these operations are shifts, the gains can be translated to dividing by multiples of two. The output gains can be generalised as:

Gain Set operation	Translated Gain
InputGainShiftValue * 1	InputGain
InputGainShiftValue * 2	InputGain * 1/4
InputGainShiftValue * 3	InputGain * 1/8
InputGainShiftValue * 4	InputGain * 1/16

This gain selection was chosen as it provides a realistic simulation of how sound waves tend to propagate and reverberate inside an environment. Later reverberations tend to decay faster than those which reverberated earlier, which is why this scheme for the gain inputs to the simulator was chosen.

3.2 Design Choices

Separate Left / right filtering

The choice to use two of the same filter for left and right filtering was done as to not introduce each filter to a mono input, meaning that both left and right samples are combined and treated as a single audio source. As many pieces of audio contain left and right channels, the audio often has stereo separation between the two inputs. Stereo separation is a concept in audio reproduction where slight differences in the left and right channels for a piece of audio leads to the impression that audio being played back has some three dimensional element, such as an instrument being played towards the right side of the listener. This effect is due to the difference in timing between the signal being heard in the right ear before the left ear. This effect would be removed if the two channels were to be combined. As the goal of reverberation units is to provide the effect that the piece of audio being processed is inside a physical environment, the use of separate left and right filtering must be used to preserve this effect. This does, however, double the amount of resources used for a single reverberation unit, which is especially apparent in systems such as the room simulator, which takes a large number of memory resources. The decision was made that without the separate filtering, then the reverb unit would not be able to fully provide the intended goal for the individual research project, to provide an accurate reverberation unit.

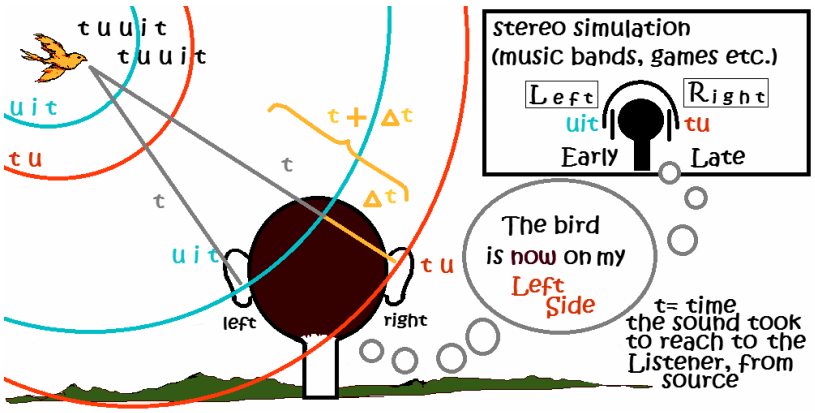


Fig 11. Diagram of Stereo Separation

Decoding / recoding channel data inside the Datapath

The decoding and reencoding of channel data was chosen to be implemented inside the datapath itself, rather than inside the control unit. This was done so as to simplify the control unit and ensure that the correct output data had the correct channel associated with it. Because the datapath makes heavy use of delays, it is unfeasible to track each sample, especially with samples recurring due to feedback. As the filters have separate left and right channels, reencoding the channel information by simply concatenating a channel bit at the end of the operation is trivial, in comparison to having a 17 bit wide, channel bit aware, reverb unit.

Parameterized inputs for Gain and Delay

The choice to use inputs from the command packets into the filters was chosen over hardcoding values for the various gains and delay line sizes. This choice was made to provide flexibility and precise control over each reverberation unit. As the reverberation time and density of echos present in a signal is a function of the delay line size, and the gain at which each reverberated signal decays, there is a vast range of configurations that can be implemented inside each reverberation unit. This allows for various kinds of reverberation settings, such as mimicking a concert hall setting or cathedral to a small room. Thus, the utility of the reverb units is expanded greatly, and can be used for a wide number of scenarios, which would otherwise not be possible with hardcoded values for the delay line size and gain values. However, this comes at a cost of memory consumption, especially with consuming more memory than would be required to operate at a certain setting. A setting which may require a small delay line, such as a small room, would have a large amount of memory space not used. The room simulator currently makes use of six All-Pass filters, if the size of the delay line only requires a length of 512 words, then there is 21504 words of memory space unused, resulting in roughly 43Kb of unused memory, for one channel only. Roughly 86Kb will be unused in total. However, the inverse is true, if the reverb setting requires a large delay line, such as a cathedral setting, then the memory space will be well utilized in comparison. Overall, the flexibility of the system far outweighs the potential resource underutilization inherent in the design. The differences in resource usage, especially for memory, can be considered a non-issue, depending on the other factors such as other ASP units in the system.

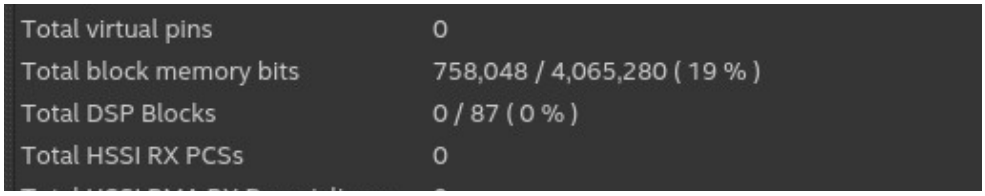


Fig 12. Memory Utilization for Non Parameterized Design

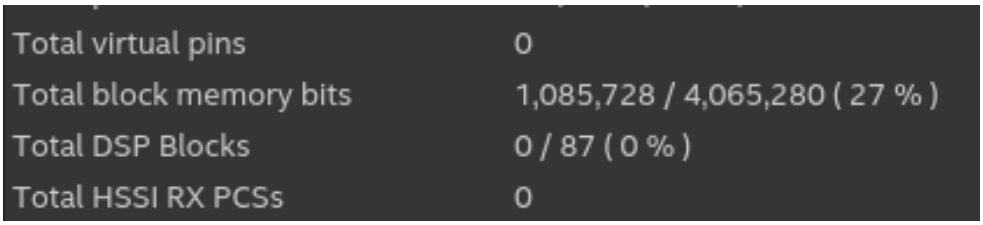


Fig 13. Memory Utilization for Parameterized Design

Shifting over proper division

The use of arithmetic shifting the samples for division over having proper fixed point division was chosen to further simplify the design of the system as well as allowing division operations to be done far quicker than implementing a proper divider. The design is slightly limited by this approach, as any disiver that is not a multiple of 2 cannot be used. However, due to the use of signed integer inputs outputted by the ADC, with no support for floating point inputs to the network, the overall design is not inherently hindered by the use of bit shifting, and is still able to operate as intended.

The use of multiples of Gain input for Room Simulation

As previously stated, the room simulator makes use of a gain set unit, which takes in an input gain and provides multiples of that input to the various stages of the simulator. As the configuration packet is somewhat limited due to the 16 bits used for parameters, with 12 bits used for delay line size, it was considered a better option to simply have the gain values be multiples of the input, as it is the relationship between the various gains between the filtering stages which determines the decay, and as such, having larger gains at near the end of the filtering stage produces optimal results. With multiple reverberations, the subsequent reverberations should begin to decay. With the design of gain set, later reverberations decay at a faster rate than earlier reflections. Whilst this design decision limits the Room simulator somewhat, the resulting outputs still conform to this reverb decay principle, and thus, fulfills the requirements of the Room Simulator.

4 Future work:

Acknowledgement packets

Currently the ASP control unit does not send out any form of acknowledgement of completion of tasks. This is partially due to the fact that functions implemented in this IRP are free running, and thus any acknowledgement that an action has been completed is somewhat void. However, acknowledgement that a mode has changed would be an important feature for extending the functionality for the rest of the system, as it allows for processors such as the ReCOP or any other processor on the network to receive acknowledgement that something has changed inside the system, and can conduct other operations which may be related to the acknowledgement packet. As discussed further on, when switching the control unit to allow synchronised operation of the reverb functions, then the use of acknowledgement packets will be necessary to facilitate this functionality.

Making use of Memory Banks

Currently, the reverberation units are free running, and do not operate on memory banks which may contain a select sampling of the input channels. For synchronised operation between the ASP and another processor, the use of memory banks which store sampled data to operate on would be required. This would require the acknowledgement packets as previously discussed, but also the ability to load memory with the incoming samples, and to have the reverberation functions operate over these saved inputs, before storing to another memory bank to be sent out. This would allow the ASP to operate synchronously with other ASP units over a single set of samples, rather than a free running stream of samples.

Memory Banks for Delay lines

Currently the implementation of delay lines inside the system contain memory specified specifically for delay line operations. However, due to the high memory usage of the system, this may turn out to be unfeasible if integrated with many processors, which also require memory for various operations. With the use of shared memory banks, which can be used for a number of operations, the delay line function can be delegated to these shared memory banks. This would in turn, shift the memory burden over to the memory banks, which can be used for a number of operations, not just reverberation. Thus, the overall memory usage of the system can be reduced, or mitigated, with the addition of memory

banks. However, this does increase the complexity of the system, as samples will have to be carefully managed whilst inside the memory bank, and the functionality of each reverberation unit will have to change to fetch and operate over memory banks, rather than from within the unit itself. The implementations of all three reverb units will have to be drastically changed to allow operation of circular buffer semantics over memory banks. This will also increase the complexity of the control unit considerably, as incoming samples and existing samples in memory will have to be managed carefully. One solution is to have shared FIFO head and tail pointers over the memory banks, and to change to multistage operation of the reverberation units. As stated previously, this will inherently increase the complexity of the units, which with units such as the Room Simulator, will require a large amount of work.

5 Testing

Testing on individual units was conducted inside ModelSim, and overall integration testing and refinement was conducted using the DE1-SoC development board. Due to the nature of reverberation units requiring a large amount of streaming data, and difficulties with testing configuration packets alongside having all three units present inside the system, testing with ModelSim was not considered. Instead, the output from the integration testing with the various units present being synthesized and programmed onto the DE1-SoC development board was used for testing the outputs for all three units.

Testing of the Low Pass Recursive Comb Filter was done inside ModelSim as well as on the development board. From Figure 14, the signal input is filling the delay line buffer before being output to the signal.

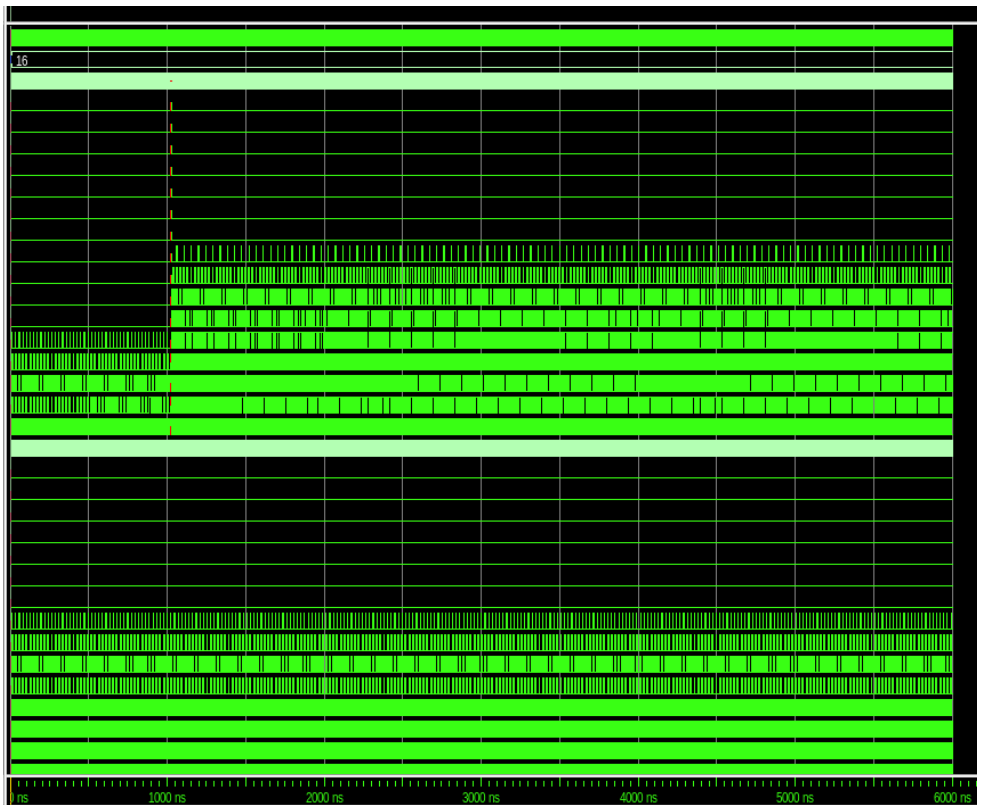


Fig 14. ModelSim output of Low Pass Recursive Comb Filter

Testing of the All-Pass filter was conducted on ModelSim and on the development board. A similar

effect as the Recursive comb can be found, with subsequent reverberations input into the output signal from the filter.

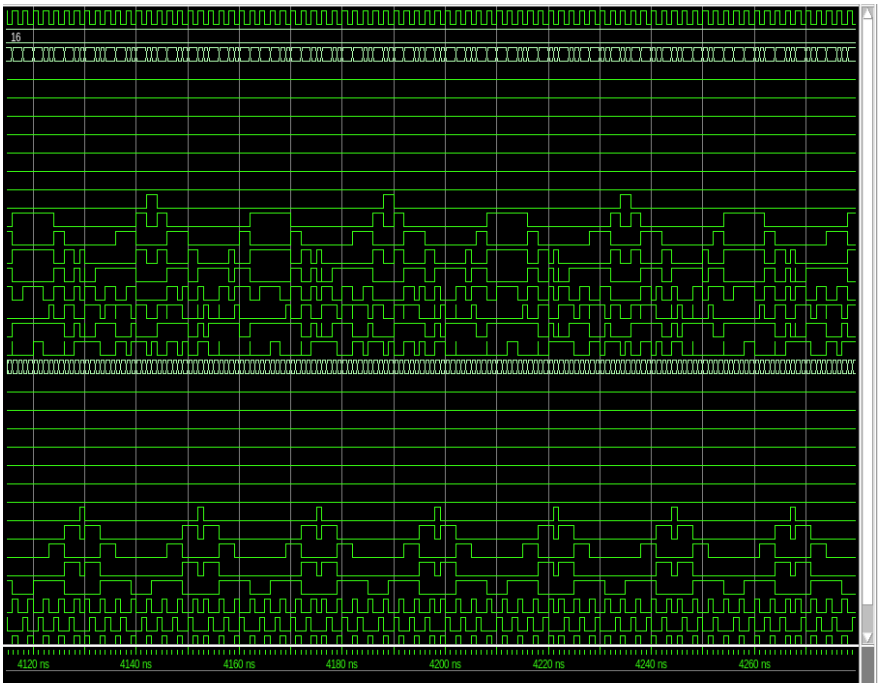


Fig 15. ModelSim output of All-Pass Filter

Compilation of the units both individually and integrated together was done. The use of both hardcoding the various delay line and gain values was done to appreciate the difference between using the parameterized and non parameterized version of delay line.

Compilation Report - CS701

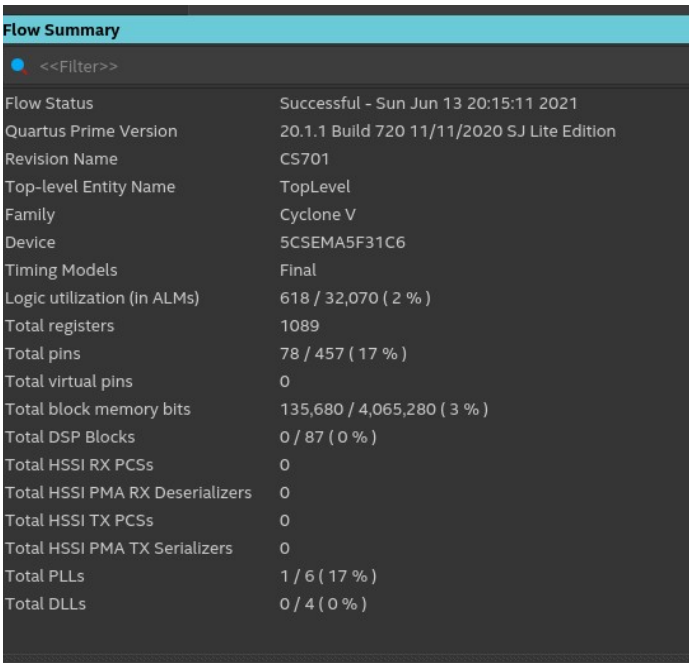
Flow Summary

<<Filter>>

Flow Status	Successful - Sun Jun 13 20:29:53 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	CS701
Top-level Entity Name	TopLevel
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	619 / 32,070 (2 %)
Total registers	1160
Total pins	78 / 457 (17 %)
Total virtual pins	0
Total block memory bits	70,144 / 4,065,280 (2 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1 / 6 (17 %)
Total DLLs	0 / 4 (0 %)

Find Next

Fig 16. Compilation report of Individual Comb Filter with hardcoded delay line size

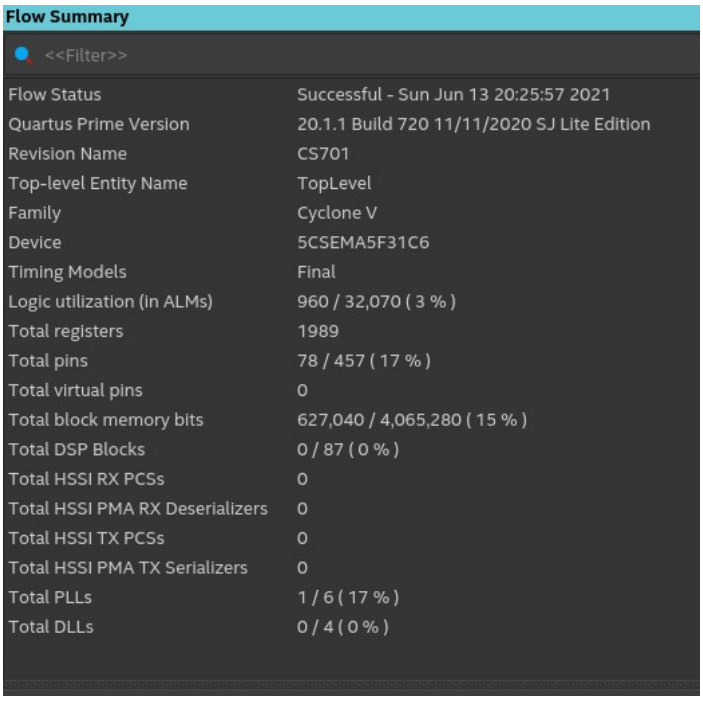


Flow Summary

<<Filter>>

Flow Status	Successful - Sun Jun 13 20:15:11 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	CS701
Top-level Entity Name	TopLevel
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	618 / 32,070 (2 %)
Total registers	1089
Total pins	78 / 457 (17 %)
Total virtual pins	0
Total block memory bits	135,680 / 4,065,280 (3 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1 / 6 (17 %)
Total DLLs	0 / 4 (0 %)

Fig 17. Compilation report of Individual All-Pass Filter with hardcoded delay line size



Flow Summary

<<Filter>>

Flow Status	Successful - Sun Jun 13 20:25:57 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	CS701
Top-level Entity Name	TopLevel
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	960 / 32,070 (3 %)
Total registers	1989
Total pins	78 / 457 (17 %)
Total virtual pins	0
Total block memory bits	627,040 / 4,065,280 (15 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1 / 6 (17 %)
Total DLLs	0 / 4 (0 %)

Fig 18. Compilation report of Individual Room Simulator with hardcoded delay line size

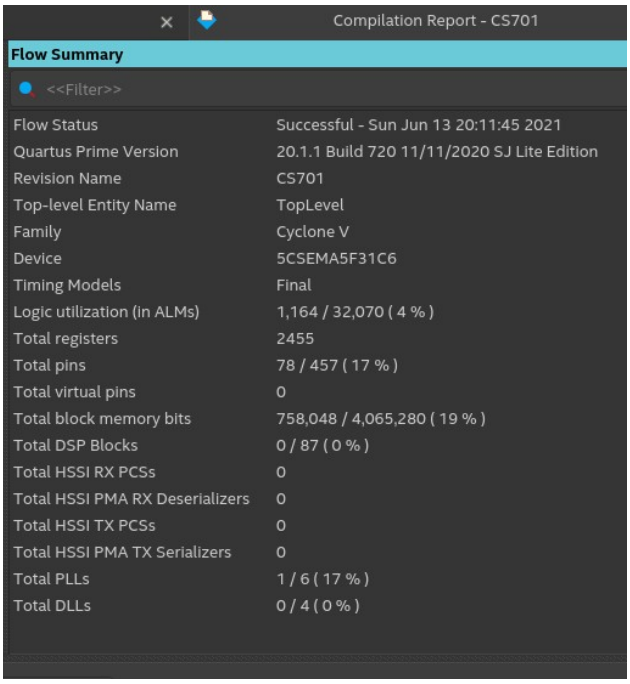


Fig 19. Compilation report of all functions implemented inside system, delay size hardcoded

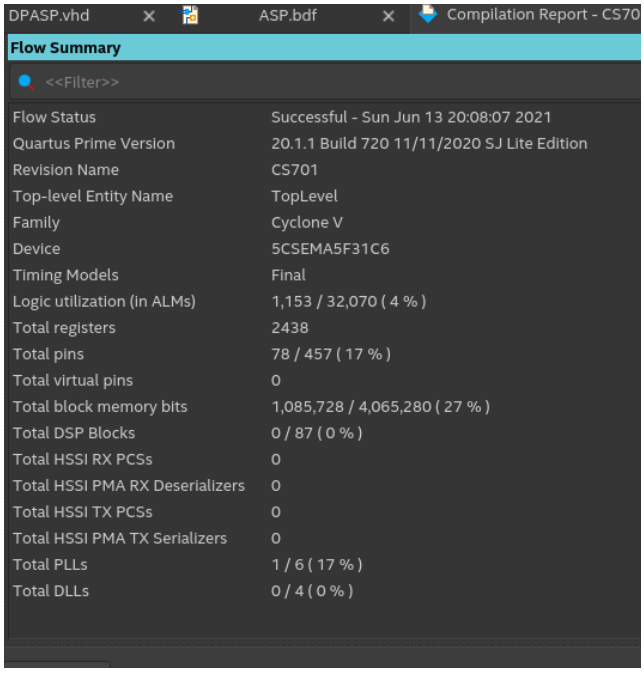


Fig 20. Compilation report of all functions implemented, delay size parameterized

The implementation of parameterized Delay lines drastically increases the amount of memory used in the overall system.

6 References

[1]
Anon: Reverberation
https://en.wikipedia.org/wiki/Reverberation

[2]
U.Zölzer: Digital Audio Signal Processing, Second Edition, ISBN 978-0-470-99785-7
pp. 200-212, 2008

[3]
G. P. M. Egelmeers, P. C. W. Sommen: A New Method for Efficient Convolution in Frequency Domain by Nonuniform Partitioning for Adaptive Filtering, IEEE Trans. Signal Processing, Vol. 44, pp. 3123–3192, December 1996.

[5]
J. Dattorro: Effect Design – Part 1: Reverberator and Other Filters, J. Audio Eng. Soc., Vol. 45, pp. 660–684, September 1997.

[6]
M. Joho, G. S. Moschytz: Connecting Partitioned Frequency-Domain Filters in Parallel or in Cascade, IEEE Trans. CAS-II: Analog and Digital Signal Processing, Vol. 47, No. 8, pp. 685–698, August 2000.

[7]
Udo Zölzer, Ed., Digital Audio Effects, John Wiley & Sons, Inc., New York, 1 edition, 2002, ISBN-13 978-0471490784.

[8]
W. G. Gardner: Reverberation Algorithms, in M. Kahrs and K. Brandenburg (Ed.), Applications of Digital Signal Processing to Audio and Acoustics, Kluwer Academic Publishers, Boston, pp. 85–131, 1998.

Figures: