Wave Array Design Description

Table of Contents

[1 Introduction 1](#_Toc106540084)

[2 Design 1](#_Toc106540085)

[2.1 Oscillator 1](#_Toc106540086)

[2.1.1 Parameter Table 2](#_Toc106540087)

[2.1.2 Mipmap Tables 3](#_Toc106540088)

[2.1.3 Wave table storage 3](#_Toc106540089)

[3 Implementation 3](#_Toc106540090)

[3.1 Oscillator 4](#_Toc106540091)

[3.1.1 Wave table memory 4](#_Toc106540092)

[3.1.2 Filter coefficient memory 4](#_Toc106540093)

[3.2 UART protocol 5](#_Toc106540094)

[4 References 5](#_Toc106540095)

# Introduction

This document describes the design and implementation of the wave array synthesizer.

# Design

## Oscillator

The oscillator design is based on [1]. Mipmapping is used in combination with 2x oversampling. This avoids not only aliasing during the resampling process, but also prevents the spectral content to change noticeably when switching between mipmap levels.

The interpolation is performed by a polyphase FIR filter. To avoid having to store too many phases of the filter in memory, the filter coefficients are linearly interpolated.

The oscillator also supports crossfading different frames of the wavetable. The samples of two consecutive frames are linearly interpolated before the polyphase filter is applied.

A schematic overview of the memory accesses and operations performed in the oscillator is shown in Figure 1.

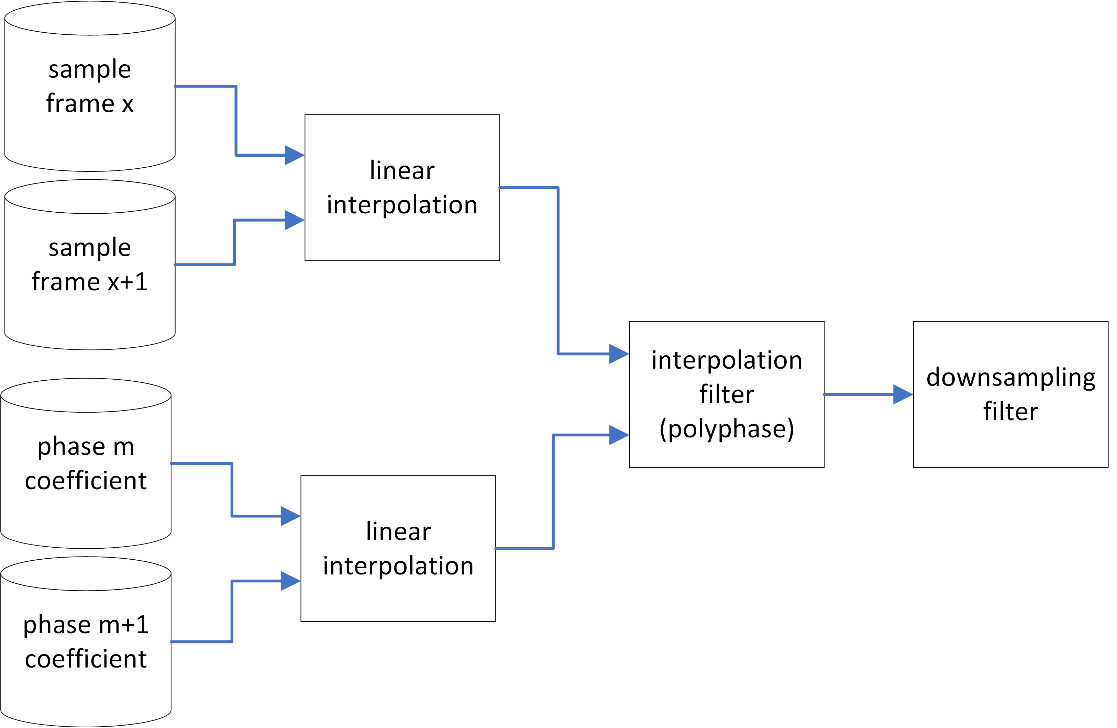


Figure 1: Schematic overview of oscillator.

### Parameter Table

This section gives an overview of the chosen design parameters of the wavetable implementation.

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **symbol** | **Value** | **Comment** |
| Wavetable size | L | 2048 | De-facto standard. |
| Output sample rate | Fso | 48 kHz | Audio output rate. |
| Input sample rate | Fsi | 96 KHz | Table sample rate with oversampling. |
| Table oversampling rate |  | 2 |  |
| Polyphase filter taps | N | 16 |  |
| Polyphase filter phases | M | 128 |  |
| Polyphase phase interpolation resolution | D | 8 bits | Precision used to interpolate filter coefficients. |
| Frame interpolation resolution |  | 8 bits | Precision used to interpolate wave samples. |

### Mipmap Tables

Mipmapping is a technique borrowed from the video graphics world used to prevent aliasing caused by subsampling of textures. A good explanation of the technique in the context of graphics can be found in [2]. This technique can also be applied to wavetable synthesis to the same effect.

dm

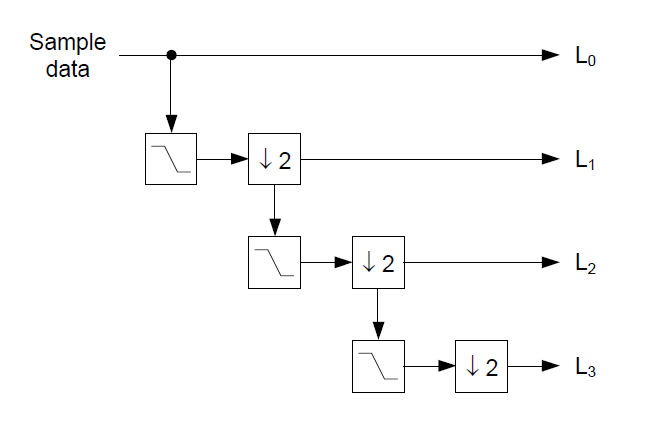


Figure 2: This figure shows how mipmap tables are generated. Each successive level is created by filtering the previous level with a half-band filter and down sampling with a factor of 2 (taken from [1]).

### Wave table storage

There is not enough space in the FPGA to store wavetables when these contain multiple frames. Therefore, the oscillator holds only two frames at once between which it can interpolate. These frames are swapped out dynamically to access the other frames.

Swapping frames takes more time than the oscillator sample period. So swapping frames need to be done in a separate buffer than the oscillator is using to prevent audio glitches.

# Implementation

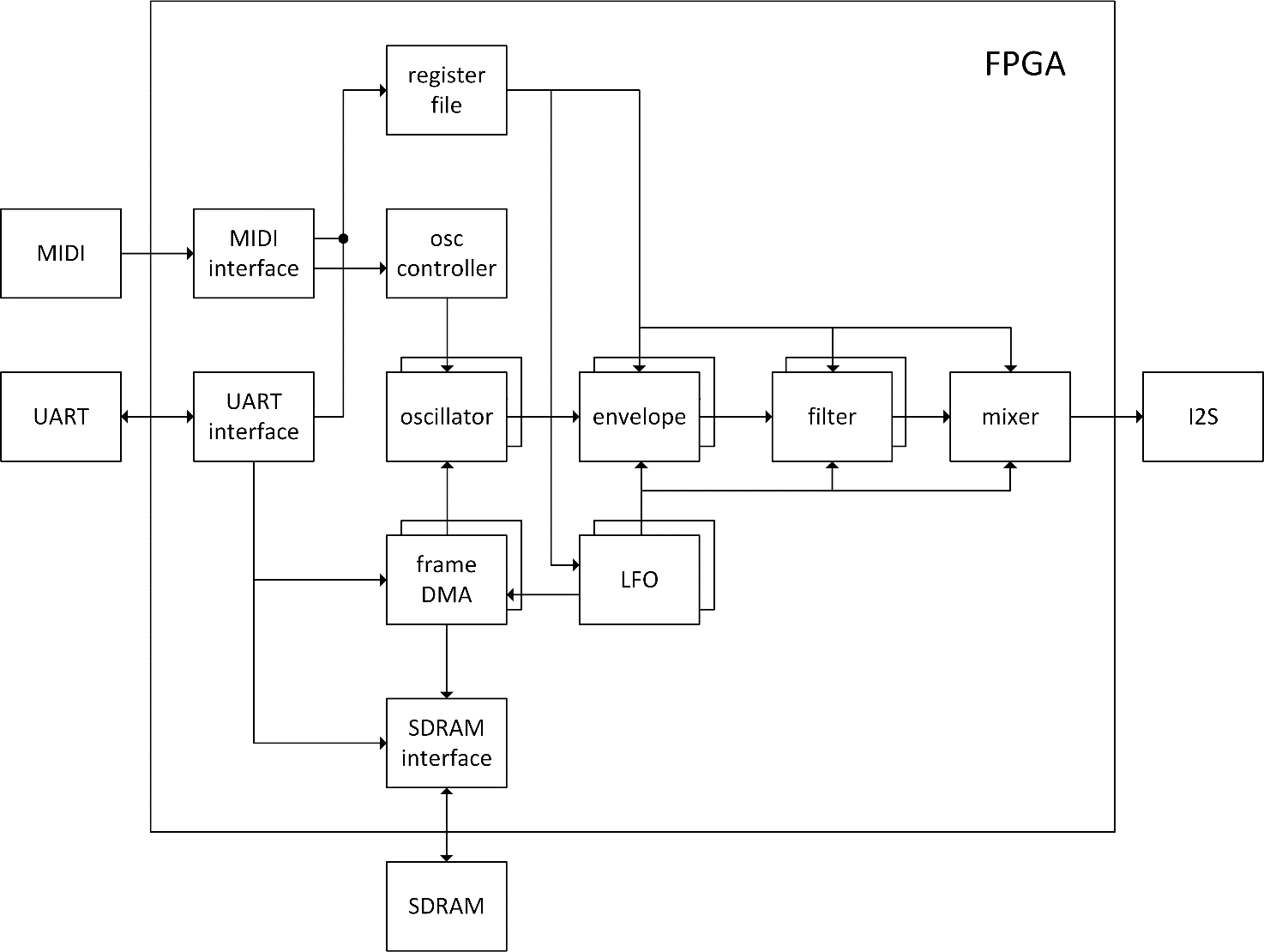


Figure : Simplified system schematic.

## Oscillator

### Wave table memory

The wave table memory needs to hold mipmap tables of two frames of the wavetable at once. Each of these mipmap tables is twice the size of the frame. Dynamic swapping of frames requires an additional set of buffers which doubles the size again. The total required memory size is:

Which takes up seven and a half 36kb BRAMs. The buffers are stored interleaved and the read width is 64 bits wide. This allows reading a sample of each of the 4 mipmap tables/buffers in a single cycle.

### Filter coefficient memory

Two 18kb half BRAMs are used to store the filter phases. One stores the even phases and the other stores the odd ones. This allows to read both coefficients required for interpolation in the same cycle. Both are configured as 16 bits wide and N \* M / 2 coefficients deep. A single BRAM allows storing up to 140 phases for a filter length of 16 taps per phase.

## UART protocol

A simple register-based protocol is used to read from and write to the register file and the SDRAM. The UART interface can be used to read and write single (16 bit) registers or blocks of registers. All messages are acknowledged by either a corresponding reply message or an error reply.

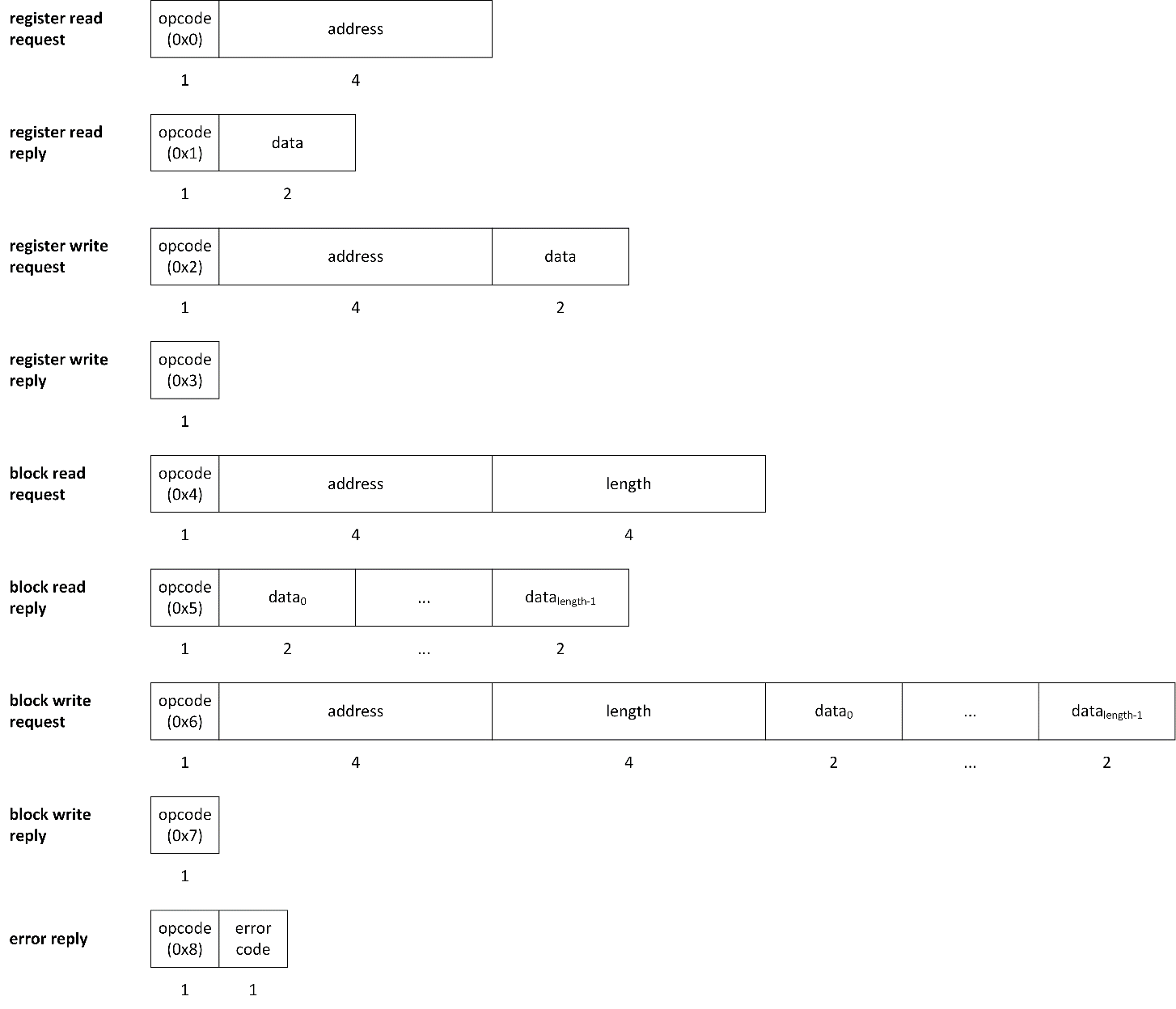


Figure : Overview of all packet types of the UART protocol. The length of each field is given in bytes.

# References

|  |  |
| --- | --- |
| [1] | L. d. Soras, "The Quest For The Perfect Resampler," http://ldesoras.free.fr, 2003. |
| [2] | B. Golus, "Sharper Mipmapping using Shader Based Supersampling," https://bgolus.medium.com/sharper-mipmapping-using-shader-based-supersampling-ed7aadb47bec, 2019. |