CASCADED BUFFER MODULATION FOR SOUND SYNTHESIS ON EMBEDDED SYSTEMS

TSCHAVOLL Jakob

j.tschavoll@campus.tu-berlin.de

ABSTRACT

Much greater processing power allows for more creative ways for digital signal processing or more specifically, sound synthesis. Even embedded systems today are well suited to audio manipulation thanks to direct hardware access with read and write pointer manipulations. One of these manipulations is *buffer modulation* by which the reading pointer's position is not merely an incremental operation, but a function of itself, causing the digital equivalent of „turntable-scratching". This paper explores the possibility for unique sound synthesis by chaining such functions in series, creating very obscured but mathematically simple representations of a buffer of audio dubbed *cascaded buffer modulation* without ever having to manipulate the original sound source itself.

Index Terms— sound synthesis, DSP, embedded systems, buffer modulation

1. INTRODUCTION

The more artistic side of digital signal processing is in constant search for new and unique ways to create sound. This sound synthesis is often seen as trivial: A sound stored in digital form and then retransformed via analog-to-digital converters and loudspeakers for the human ear either stems from an audio recording or is based on a simple waveform which can additionally be filtered in the time and frequency domain.

There are however more interesting approaches which don't involve „classical" DSP, but physically nonsensical yet interesting methods. One of these methods is called *buffer modulation* by which a buffer full of arbitrary audio is not read incrementally from the first sample to the last, but instead the reading pointer, a.k.a. *index*, follows a non linear function of its own. This is, to some extent, the digital form of turntable-scratching. The constant turning of the vinyl record can be seen as an incremental procedure of analog samples while moving the record back puts already occurred samples „back in time".

To expand on this idea, it is not far fetched to imagine an index function which computes the output of another function instead of the true index. If enough of these *index functions* are chained in series, the original buffer contents will be transformed in an unexpected way, even if the mathematical procedures behind it are simple. As well as being a unique method for sound synthesis, this method is also very efficient, since only indexes have to be calculated. This makes the method very suitable to run on cheap and hardware-centered embedded systems.

2. METHOD

A buffer represents a block of digital memory read- and writable contents. It has N samples which each have a size, usually called *bit depth* or *resolution*. Each sample can be addressed with an index ranging from 0 to $N-1$. To accurately reproduce the digitally stored sound, the index has to read the contents of the sample and then address the next sample within equal time steps. This increment can be viewed as a function, here called *base function* f_B , visible on the top left in figure 1.

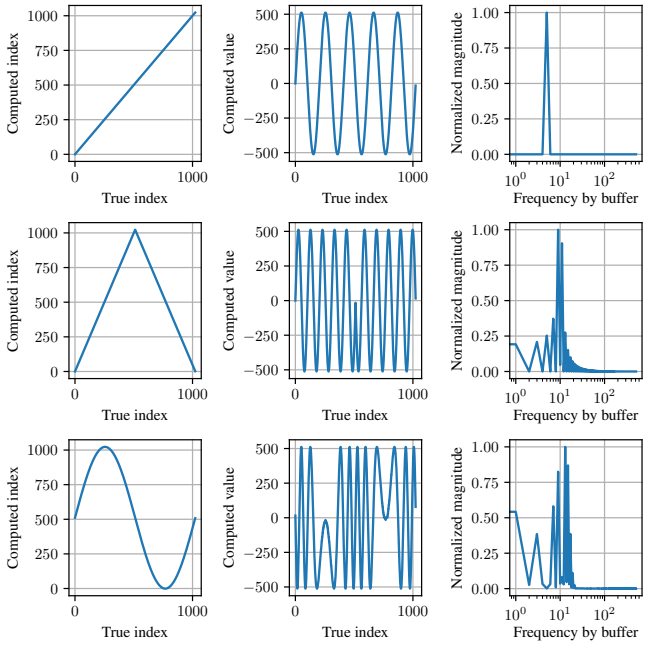

Fig. 1. Indexer functions (left) with resulting signals (center) and frequency responses (right).

By instead reading the buffer with a triangle shaped indexing function f_T, the content of the buffer (second column of figure 1) is read twice as fast and then backwards. Naturally, a frequency shift due to twice the reading speed and spectral leakage due to a jump in the buffer occur as results. Alternatively, the indexing function can also be a sine wave f_S, visible for a single period in the third row in figure 1.

If such functions are now dynamically chained after each other, the effects on the original sound become physically nonsensical but interesting from a sound synthesis perspective. For example, a cascade of two triangle indexing functions causes a frequency duplication for periodic signals, while two sine indexing functions create a signal with greater bandwidth, visible in figure 2. To make such a system fast and dynamic on embedded systems, a *linked list* containing one indexing function and an in- and outgoing temporary index per node are especially well suited. With this approach, a static buffer is able to produce an infinite number of unique sounds just by being read by different cascaded indexing methods. It should be noted that cascaded base functions do not create change in the indexers, as their inputs and outputs are equivalent. The implementation is visible in the C-code below:

```
typedef struct Indexer {
   void (*method)(uint16_t* i, uint16_t* o);
   uint16_t i;
   uint16_t o;
   struct Indexer* next;
} indexer_t;

void nextSaw(uint16_t* i, uint16_t* o){
   *o = *i;
}

void nextTriangle(uint16_t* i, uint16_t* o){
   if(*i < (f_end_pos-f_start_pos)/2){
      *o = 2*(*i);
   }
   else{
      *o = 2*((f_end_pos-f_start_pos)-*i);
   }
}

void nextSine(uint16_t* i, uint16_t* o){
   *o = (uint16_t)(sin(omega * (*i)) *
       (double)(((f_end_pos-f_start_pos)+1)/2
       - 1)) + ((f_end_pos-f_start_pos)+1)/2;
}
```

A practical implementation was made possible on an ESP32 micro-controller by Espressif[1]. It features an external SD-card reader to obtain the contents of a static buffer and a I²S-DAC with 3,5 mm headphone jack for analog audio. Interfacing is possible via a command line interface (CLI) connected to a serial terminal. A dynamically allocated

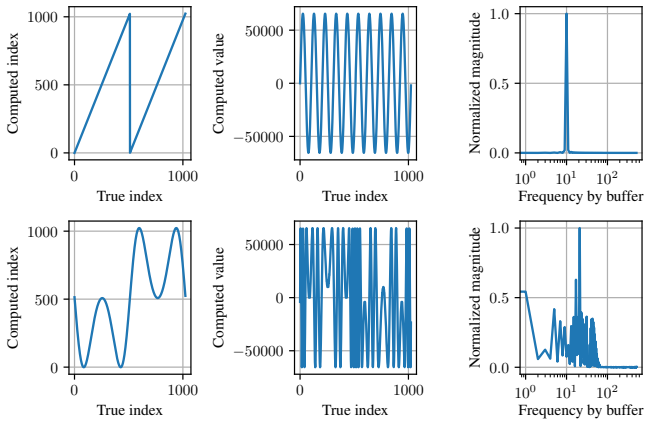

Fig. 2. Cascaded indexer functions (left) with resulting signals (center) and frequency responses (right).

linked list is able to chain an arbitrary number of triangle and sine indexing functions after each other. Additionally, the start and end points of the buffer containing one second of 48 kHz audio can be adjusted to shorter durations.

3. RESULTS

By applying various indexing functions and changing the buffer ranges, a multitude of sounds can be created. Naturally, shortening the buffer and applying the triangle indexing function results in frequency shifting towards higher values. The modulated sound files consist of one second of audio and have the following contents:

- water droplet
- tom transient from a drum rack
- pad synthesizer
- funk hit

The tom transient is visible as original (top) and cascaded buffer modulation (bottom) in figure 3. The indexer structure is $f_S(f_T(f_S(f_B)))$, visible on the lower left of figure 3. For transient sounds, noticeable repetition occurs in case of large buffer sizes. If the buffer gets reduced or enough indexer functions are applied, the original sound becomes unrecognizable, leading to a unique synthetic sound. The original sounds and a short track composed of their cascaded buffer modulations can be found in the project's repository[2].

4. DISCUSSION

The method *cascaded buffer modulation* demonstrates a unique and unusual way to obtain new sounds from prerecorded sound samples. By moving the signal processing to

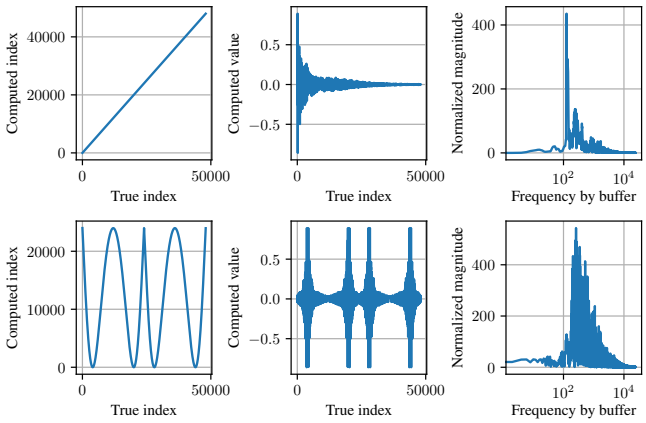

Fig. 3. (Cascaded) indexer functions (left) with resulting tom transient (center) and frequency responses (right).

the index instead of the waveform itself, the process becomes efficient and fast. Due to the limited amount of indexing functions, the parameter control stays somewhat limited and non-deterministic to the human ear and is therefore more of a randomizer with a seed strongly correlated to the original sound source. This sound source can be of synthesized nature as well, as cascaded buffer modulation is not dependent on the waveform of such a source. The only limiting factor is buffer length, which can be used for additional parameter control.

As of now, the indexing functions create their results always based on buffer length. This causes the same shape to appear for every buffer size. Adding an independent index might be a way to create even more unique sounds which feature indexer dependent modulations even after a full buffer has been read. This concept can also be expanded into the dimension of the amount of indexing functions. By mapping the amount of these functions to a linear control, a dynamic amount of them can be changed during runtime, resulting in wildly unexpected behaviour. This gives artists the tools to quickly explore a wide range of sounds without having to load additional samples.

5. REFERENCES

[1] Espressif, "ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems," .

[2] Jakob Tschavoll, "cascaded-buffer-modulation," Apr. 2023, original-date: 2023-04-11T08:22:18Z.