POWERWAVE - A HIGH PERFORMANCE SINGLE CHIP INTERPOLATING WAVETABLE SYNTHESIZER

Robert Trausmuth

Dept. of Computer Engineering
University of Applied Science
Wiener Neustadt, Austria
trausmuth@fhwn.ac.at

Antti Huovilainen

Laboratory of Acoustics
and Audio Signal Processing
Helsinki University of Technology, Finland
ajhuovil@acoustics.hut.fi

ABSTRACT

In this paper we introduce the single chip implementation of a 16 voices wavetable synthesizer. All digital functions (control and waveform generation) are contained in a single platform FPGA chip (Xilinx Virtex 2 Pro). Only the digital to analog conversion is done by a standard 96 kHz audio DAC (AD 1785). In the first version the synthesizer is controlled via standard RS 232 interface.

1. INTRODUCTION

During the last years the boundaries between hardware and software have become distorted. The technology of FPGAs gives an embedded systems engineer the possibility of designing a full system in one chip. Only the first level drivers for electrical interfaces have to be implemented in discrete electronics. One of the real challenges in modern embedded system design is to find the best solution for dividing the implementation of components in hardware and software. Designing parts of the solution in hardware is no longer the domain of electrical engineers. Our motivation for this project was to create a system on chip design which would exploit the inherent parallelism of FPGAs and also the embedding of the slow control all in one chip. A quite powerful DSP and a lot of glue logic would be the alternative. With this concept we reduce the chip count to FPGA + DAC (and eventually some external memory).

Starting with the PPG Wave synthesizer [4] in mind we tried to avoid the snapping and clicking when switching between waveforms. The way PPG Wave implemented oscillators was that there was a set of 64 short waveforms in memory. Each oscillator had a wavetable position (waveform index) that could be changed during the course of note and that way the spectrum could be modified. There was no interpolation so the waveforms aliased and there was a small click/snap whenever the wavetable position changed.

We planned to improve this by interpolating between adjacent samples in the waveform and by interpolating between adjacent wavetable positions too to remove the snaps. Another improvement was to use different waveform resolutions depending on the frequency, so that the waveforms used at higher frequencies would have less harmonics and aliasing would be reduced.

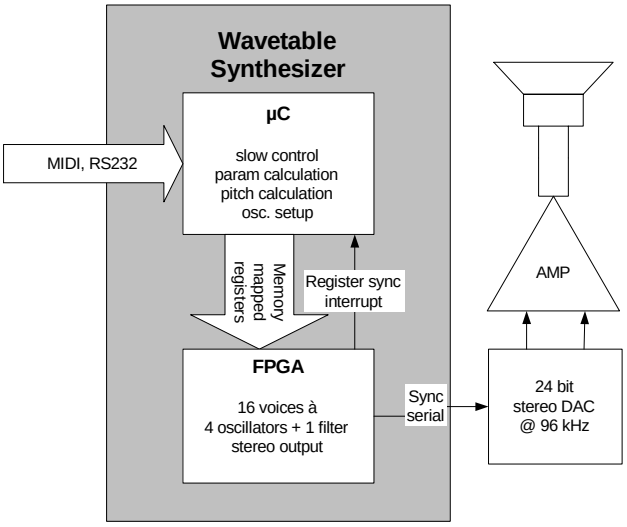

Figure 1: the overall system design

This paper shows the implementation of a 16 voice wavetable synthesizer in one chip. The sound generation and all the calculations of the oscillators are done in hardware, and the slow control is handled by a microcontroller included in the FPGA and a small C program. The standard serial RS 232 interface has been chosen as primary interface to control the synthesizer by an external computer program, although it will be replaced by MIDI or even ethernet in the next version. This will allow for live performance with an external keyboard.

The synthesizer uses 12 wavetables which contain waveform data for different frequencies. The granularity of data is adjusted to the playback frequency, so we need only 12 kB per waveform.

The work of Antti Huovilainen has been supported by the Academy of Finland (project #104934).

2. SYSTEM DESCRIPTION

The PowerWave synthesizer has 16 voices. Each voice is produced by a combination of four wavetable oscillators, which can each be controlled in waveform, amplitude and speed (frequency). Each voice has its own digital nonlinear Moog filter, and the resulting signal is then split to the stereo channels, where all voices are combined to a master left/right signal path. To give further possibilities in sound engineering the main stereo path is equipped with a 6 band parametric EQ and a variable delay feedback loop. The system uses 96 kHz standard sampling rate to produce the stereo audio signal with an AD 1785 DAC. A standard serial interface is used to play the synthesizer. The control of each oscillator and all the filters is handled by a PPC 405 CPU which is part of the Xilinx Virtex II Pro FPGA we used for this implementation. The control program updates the memory mapped control registers every millisecond. We are also working on an alternative (cheaper) design using a Xilinx Spartan 3 FPGA and a Philips UDA1380 CODEC.

The exact timing is derived from the 96 kHz sample frequency. The chip logic master clock is eight times the DAC master clock rate (12.288 MHz, 256 times the sampling rate of 96 kHz) and therefore is 98.304 MHz. The FPGA PPC 405 CPU runs at 300 MHz.

2.1. Data Flow Conceps

When doing a System-On-Chip design the system engineer has the freedom to choose which parts of the solution are done in hardware. In the case of the PowerWave synthesizer all the "fast" logic for sound generation is done this way. The oscillators read the waveform data from tables stored in RAM and generate the desired waveforms. Wave data is processed by filters and amplifiers coded in hardware and sent to the audio DAC. All variables are controlled by a microcontroller program which is executed by a PPC 405 CPU included in the FPGA. The communication between software and hardware is done using memory mapped registers on the OPB chip internal bus. Every millisecond the control program is interrupted by a signal from the oscillator hardware to recalculate the parameters for all the filters, oscillators and voices. Communication with the artist (player of the instrument) is done by the software included in the chip.

2.2. Slow Control

The PowerWave slow control can be distributed into three parts. The first process takes care of the serial interface and implements the communication protocol. A special control protocol is used in the first prototype implementation. The user interface is programmed in C++ and runs on a standard PC. The second process prepares all the parameters for the filters and oscillators. The third process is run every millisecond to update the control registers for the hardware implementation. The hardware synthesizer issues a synchronous interrupt signal every 96 calculations, so the interrupt occurs every ms and a real-time behaviour of the slow control can be achieved.

The first task encapsulates the communication protocol so this can easily be replaced by another protocol. For the prototype an uncomplicated ASCII-based protocol is used as a proof of concept. In the following versions this task will be replaced by a MIDI protocol handler. This task is event driven and works whenever data is available on the interface.

The second task is the main calculator of the PowerWave synthesizer. Since the sampling rate is fixed at 96 kHz all the coefficients for the filters and oscillators have to be calculated here. The calculation results are converted to signed fixed point representation suitable for the FPGA (1 + 17 or 1 + 23 bits) and buffered for transfer to the hardware section. This task is run upon request issued by the communication task.

The third task is responsible for adjusting bit alignment and updating the hardware registers on chip. This task is interrupt driven and runs every millisecond. Since the real synthesizer runs in hardware all the parameters reside in dual port RAM on the chip. This way the communication is easier for both hardware and software implementations.

The parameters include speed (= pitch), position and amplitude values for each oscillator and one mode register per voice. The change in speed and amplitude is done via differential parameters delta_speed and delta_amp to minimize glitching. The parameters for the Moog filter include g and resonance (see section 2.4) as well as four gain parameters per filter stage for alternative responses. Two more registers per voice are used to control the panning. The final parametric EQ has another 18 parameters (gain, cutoff frequency and bandwidth per filter stage). The total number of parameters is 866 and the user interface is therefore implemented in different windows.

The playing of music on this first synthesizer prototype is done by the control program on the PC, since an interactive MIDI protocol is not available now.

2.3. Wavetable Oscillator

The wavetabe oscillator uses data tables which are organized as mip tables [5]. The tables contain interpolation points for the sampled waveform. For low frequencies the table contains 2048 points per period and for frequencies near the Nyquist frequency there are only 16 samples per period. The actual value is fitted according to the current fraction of period. Also the frequency contribution is taken into account: For one sample both nearest mip tables will be evaluated and the real waveform value is fit according to the frequency settings of the oscillator. This method grants smooth transition to higher/lower frequencies without producing glitches in the sound.

In this application the oscillators use fast on chip SRAM

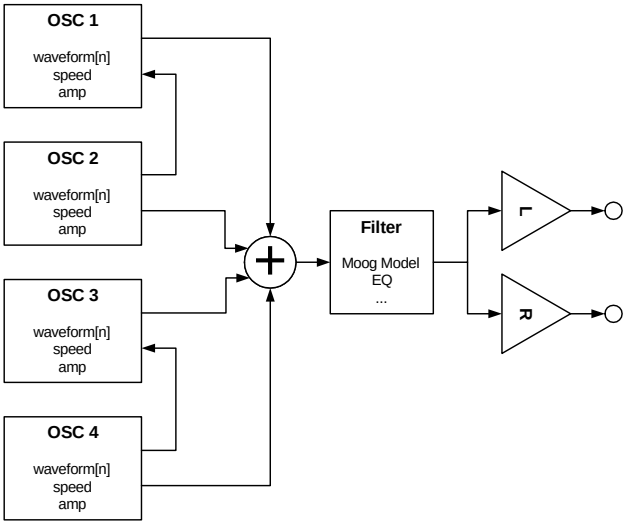

Figure 2: one voice of the synthesizer

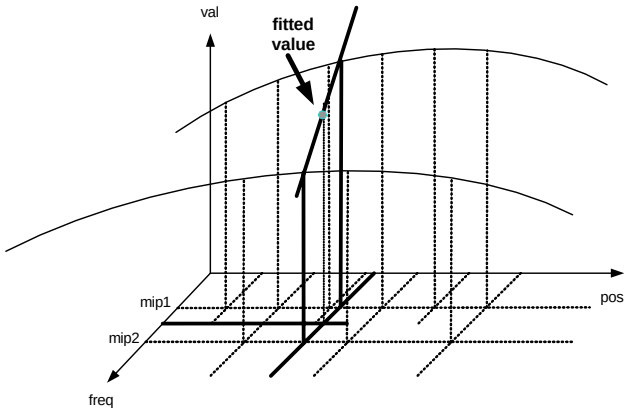

Figure 3: illustration of the fitting procedure used to find the right waveform value for given frequency and oscillator phase

memory. These memory blocks are organized in 16 bit width, and since they have dual port access, two values at a time can be read from one block.

The value for each wave sample is determined by three fits: The ratio of the oscillator speed to the Nyquist speed determines which mip table has to be used. The mip tables contain sample data in ascending resolution. The most detailed table contains 2048 samples per period and will be used for frequencies below 23 Hz. The tables follow mostly in fractions to the power of two relative to the sampling frequency of 96 kHz. The table sizes are larger than dictated by Nyquist criteria to reduce artifacts from linear interpolation. The authors have found that using two to four times longer tables with a limit on minimum and maximum table size gives good tradeoff between quality and memory usage. For a certain phase (= position) the signal value is determined from the nearest base mip table. The same procedure is done for the next octave mip table. Since the oscillator speed normally will not match a mip table frequency exactly, the final value is a linear interpolation between the two mip table values according to the real oscillator frequency. The final value depends on the current oscillator phase and the oscillator frequency, so this interpolation process can be shown in a 3D plot according to figure 3.

To provide more possibilities for sound generation the oscillators 2 and 4 can be used to manipulate the speed of oscillators 1 and 3, respectively. If used as frequency modulators the outputs of oscillator 2 and 4 are fed back to the frequency inputs of the other oscillators. Another possible setting is oscillator synchronisation. In this case the slave oscillators (1 and 3) are reset every time the master oscillators (2 and 4) wrap around. This technique can be used to produce well known sharp synthesizer sounds.

2.4. Moog filter

According to the work of Antti Huovilainen published last year on DAFx'04 [1] each voice has its own nonlinear Moog ladder filter. The original ladder consists of four differential pair transistor stages. each stage is modelled by a **tanh** lookup table with interpolation and a normal one pole IIR lowpass filter. the input of the following stage is the **tanh** of the output of the previous one.

The Moog filter has two parameters: the cutoff frequency and the resonance factor. The cutoff frequency f_c and resonance amount res are used for calculating the coefficients g and $feedback$ according to the formulae

$$\begin{aligned} w_n &= 2f_c/f_s \\ cor_f &= 1.873w_n^3 + 0.496w_n^2 - 0.649w_n + 0.999 \\ g &= 1 - e^{-2\pi w_n cor_f} \\ feedback &= 4res(-3.936w_n^2 + 1.841w_n + 0.997) \end{aligned}$$

Extending the original Moog ladder filter, the PowerWave implementation gives the possibility of using the outputs of each stage and mixing them to produce final filter output signal. Therefore the filter has another four gain values (one for each stage output). By adjusting the mix of invidual stages, various different responses can be obtained [6]. To get the original Moog ladder only the last gain is set to 1, all others are set to 0.

2.5. Output filter

The stereo output of the 16 voice generators can be filtered again before it is sent to the DAC. For this purpose we use a VHDL implementation of the six band EQ presented last

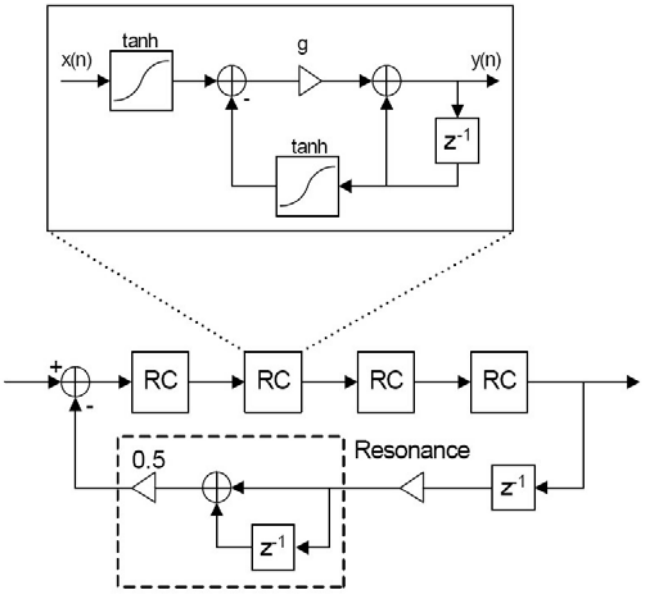

Figure 4: digital implementation of the Moog filter

year on DAFx04 [2]. The filter calculation theory and simulation has been done by Franz Siegmeth [3]. The calculations are implemented in hardware and done in a few time cycles.

The EQ uses six two-pole IIR filters with $\pm 20dB$ each. Since frequency, gain and bandwidth can be adjusted dynamically two or more filter units can be combined to get a sharp attenuation or amplification. The parameter calculations are done by slow control. The real time audio data calculations are done in hardware.

3. ON CHIP RESOURCE USAGE

The XC2VP20 FPGA has enough resources to host all the necessary system components. The PowerWave synthesizer uses roughly half of the on chip resources. To give the reader an overview of those resources a few numbers are mentioned here.

Each oscillator of the PowerWave synthesizer uses 3 18x18-multiplier and 8 36 bit adder. The calculation of one oscillator needs 15 clock cycles. This can get worse if slower memory is used for the wavetables. One voice has 4 oscillators which gives a total of 12 multipliers and 32 adders.

The Moog filter consists of 4 stages with a total of 5 interpolating **tanh** lookup tables (1 multiplier, 2 adders each) and another 1 multiplier and 2 adders per stage. The feedback loop takes 2 multipliers and 2 adders. So the total Moog filter uses 25 multipliers and 38 adders. The output is split to two stereo master channels with 1 multiplier and 1 adder each.

The voices are pipelined in calculation, so we only need the resources for one voice: 39 multipliers and 72 adders.

The EQ filter consists of 6 stages of biquad IIR blocks. One block needs 5 multipliers and 4 adders. So the total filter uses 30 multipliers and 24 adders. The stereo signal has a main gain control which uses another 2 multipliers and 2 adders.

The adders are implemented in chip logic in the required bus witdh. The multipliers are available on chip as hardware components, so they are pretty fast but limited in width. Our design uses 71 out of 88 multipliers availabe on chip. The PowerPC 405 microcontroller system needs roughly 3000 slices of logic, our design needs another 3000. So the 9000 available slices on chip are enough to incorporate the whole system in one chip. Up to 500 kB of BlockRAM (fast dual port memory on chip) can be used for wavetables. Since this SRAM has two independend ports the calculations of the oscillators can be accelerated by getting two values in one clock cycle.

4. CONCLUSIONS

In this paper we presented a fully integrated wavetable synthesizer on one chip. The implementation takes profit from hardware / software codesign. All calculations are done directly in hardware. Only the digital to analog conversion of the final stereo signal has to be done with an external audio CODEC. The proof of concept implementation uses four waveforms.

In the next version there will be more waveforms (up to 64), an additional ring modulator in the stereo output chain and a standard MIDI interface to allow live performace with the PowerWave synthesizer.

5. REFERENCES

[1] **A. Huovilainen, *Non-linear Digital Implementation of the Moog Ladder Filter*, DAFx04 Naples, 2004**

[2] **R. Trausmuth, M. Kollegger, *ADAM - a 64 channel general purpose realtime audio processor*, DAFx04 Naples, 2004**

[3] **Franz Siegmeth, *Realisierung eines digitalen Mischpultes mit DSP*, diploma thesis, 2003**

[4] **Mark Vail, *Vintage Synthesizers*, Miller Freeman Books, 1993, ISBN 0-87930-275-5**

[5] **Ken Greenebaum (ed), Ronen Barzel (ed), *Audio Anecdotes II: Tools, Tips, and Techniques for Digital Audio*, Phil Burk, Band Limited Oscillators Using Wave Table Synthesis, pg 37-54, A K Peters Ltd, 2004, ISBN 1-56881-214-0**

[6] **Oberheim, *XPander service manual*, 1984**