REAL-TIME FINITE DIFFERENCE PHYSICAL MODELS OF MUSICAL INSTRUMENTS ON A FIELD PROGRAMMABLE GATE ARRAY (FPGA)

Florian Pfeifle
Institute of Musicology
University of Hamburg
Hamburg, DE
`Florian.Pfeifle@haw-hamburg.de`

Rolf Bader
Institute of Musicology
University of Hamburg
Hamburg, DE
`R_Bader@t-online.de`

ABSTRACT

Real-time sound synthesis of musical instruments based on solving differential equations is of great interest in Musical Acoustics especially in terms of linking geometry features of musical instruments to sound features. A major restriction of accurate physical models is the computational effort. One could state that the calculation cost is directly linked to the geometrical and material accuracy of a physical model and so to the validity of the results. This work presents a methodology for implementing real-time models of whole instrument geometries modelled with the Finite Differences Method (FDM) on a Field Programmable Gate Array (FPGA), a device capable of massively parallel computations. Examples of three real-time musical instrument implementations are given, a Banjo, a Violin and a Chinese *Ruan*.

1. INTRODUCTION

Physical modelling and computational synthesis of acoustical instruments are forming the basis of many musicological, acoustical and engineering applications and research [1], [2], [3], [4], [5]. Examples are the understanding of structural features of musical instruments and the estimation of the importance of fine structures, like complex couplings, non-linearities, or orthotropic material properties, all qualities of key importance to instrument builders interested in changing or improving their instruments [3], [6]. Another important field of research is the relation between instruments, perception, composed music, and ethnic musical traditions in terms of the chosen material, sound requirements, and/or tradition. Here the Physical Model can serve in two ways, through the visualisation of the time-dependent vibrational parameters, like displacements or flows, and through the auralization of these vibrations, resulting in musical tones, melodies, articulations, or whole musical pieces.

Many ways of Physical Modelling have been proposed. Even though these methods differ in numerous aspects, they have one conjoining property, namely the fit of the discrete model to the real instrument strongly depends on the accuracy of the modeled geometry and material, as well as on the appropriateness of the chosen mathematical model. In practice this means the computational effort rises with the complexity of the model. Therefore, in Musical Acoustics much effort was put into finding simplified formulations of the vibrational behaviour of musical instruments to compute them in real-time or close to real-time [7]. Most of these works propose simplification and especially linearisations of the physical equations. Most filter based techniques, like delay lines, Waveguides, or modal synthesis use linear approximations and neglect most of the occurring diverse non-linear behaviour. They also neglect most of the geometrical fine structure, discontinuous material or tension and pre-stress distributions, and complex coupling between parts of the specific instrument body. Furthermore they are hard to formulate in higher dimensions. Although some sounds produced by these methods are close to real instrument sounds, still restrictions in terms of articulation and instrument flexibility are present. Moreover, as these models do not consider the real geometry, the link between specific geometry features and sound behaviour often stays unclear and so its use for instrument builders is restricted.

Another approach is Physical Modelling which solves partial differential equations (PDE) governing systems in one, two, or three dimensions. These PDEs may be the wave equation, the membrane or the plate equations, the Helmholtz equation, or flow formulations like the Navier-Stokes or the Euler equation. Due to the fact that analytical solutions of these differential equations are only known for plain geometries, like e.g. rectangles or circles, when solving more complex geometries only discrete formulations of these differential equations can be used to obtain solutions. The most commonly used methods for solving discrete formulations of PDEs include Finite Element Methods (FEM), Boundary Element Methods (BEM), and Finite Differences Methods (FDM). As these approaches are capable of including all mathematical models, arbitrarily shaped geometries, all kinds of nonlinearities, and also all kinds of sound radiation, they seem to be the holy grail of Physical Modelling. Still they have the large trade-off that a real-time computation is far beyond the capabilities of a standard PC. So other hardware solutions have to be used to make these methods real-time capable. At present, two hardware architectures are used for accelerating numerical calculations in many scientific fields, the Graphics Processing Unit (GPU) and the Field-Programmable Gate Array (FPGA). The GPU seems attractive at first glance as most PCs have such a unit installed already and many recent studies have shown the advantages of GPU calculations over CPU implementations. Still, even though a standard GPU accelerates algorithms around 4 - 10 times, this gain in calculation time is still not enough to solve the comprehensive Physical Models discussed above in real-time. The FPGA on the other hand was been shown to be capable of speeding up discrete model FDM solutions to real-time [8], [9], [10], [11], [12], [13], [14].

Because of their massive-parallel and therefore high speed processing capabilities and because of their flexibility, FPGA-devices are used in many fields of Signal Processing. FPGAs are used in the implementation of real-time noise source identification [15], high-speed direction-of-arrival algorithms [16], high-speed cross correlation [17], or delay-and-sum beamforming [18], [19]. Still they are rarely used for the synthesis of sound. Martins et al. [20] focus on a low-cost method to process sound on a FPGA without the need of additional electronics, still omitting musical aspects. A FPGA was also utilised as a function generator for simple signals like a sine wave or a rectangle signal (see [21], [22]). All of these works focus on high-frequency rather than audible signals, e.g. suggesting an amplitude-modulation-demodulation chain for a digital radio receiver [21] but not on musical sound synthesis.

Another conspicuity in typical signal processing applications on a FPGA is that they are mainly realized as a IIR- or FIR-filter design. Madanayake et al. [23] implement 2D/3D plane-wave filters by using IIR/FIR-filters. Shuang at al. [24] focus on analog-to-digital controllers using a similar filter-design. Several papers propose methods of implementing DSP filter designs (IIR/FIR) on a FPGA chip like [25] or [26].

Still there are implementations of solving differential equations on FPGA hardware. Motuk et al. [27], [27] propose the creation of new instruments, similar to a drum machine by simulating plates using a FPGA. Simulating electromagnetic fields by solving the Maxwell equations using a Finite Differences in the Time Domain (FDTD) algorithms was proposed by [28]. The FDTD-Method a widely acknowledged algorithm for the analysis of electromagnetic problems [29].Strzdoka et al. [30] propose the conjugate gradient method implementation as a solution for linear equations which solve differential equation systems. As noted above, its parallel processing capability predestines the FPGA to be used in real-time applications like particle track recognition [31], next to many other applications. All of these works have shown that algorithms could be realized in real-time for the first time or be sped up tremendously using a FPGA.

So indeed a real-time implementation of a FDM algorithm to solve differential equations to simulate a whole musical instrument on a FPGA is highly valuable. Although there are several papers that research the real-time capability of FPGAs for Acoustic Modelling all of these works focus on special cases of the wave equation e.g. strings, membranes, plates or air-volumes [32], [27], [33]. To the best of our knowledge there has been no work that successfully modelled whole instrument geometries with interacting parts or coupled wave equations in varying dimensions next to the authors work [8], [9], [10], [11], [12], [13], [14]. The proposed method in this paper tries to bridge the gap between both worlds, real-time capability and physical accuracy of modelled instruments.

Among other features, the model includes:

- the whole geometry of the musical instrument,
- all material parameters,
- anisotropic distributions of material parameters over the geometry
- a multiple of differential equations governing the single instrument parts
- couplings between these parts,
- non-linearities caused by physical properties, couplings, material, etc...
- solution of the dependent variables, e.g. displacement, velocity, or flow on the geometry
- radiation of sound to a virtual microphone or listener position.

The model is implemeted on a FPGA development board and linked to a standard PC. A user interface gives the possibility to change physical parameters in real-time. These parameter changes include:

- changing the geometry in real-time,
- changing material parameters in real-time,
- changing playing parameters like plucking of a plectrum, bowing-velocity or bowing-pressure.

With these capabilities, the proposed application may be used by:

1. instrument builders who can listen to the sound produced by different geometries or materials
2. researchers studying the behaviour of an instrument when changing parameters
3. musicians playing an instrument and interaction with the model in real-time producing notes and articulation.

2. INTRODUCTION TO FPGA ARCHITECTURE

As shown above, to compute finite differences with sufficient accuracy in real-time a high speed processing unit is needed. In recent years many studies and publications have shown that a FPGA (Field Programmable Gate Array) is the most feasible device to fulfill highly specialized tasks in different fields of signal processing. In this section we give a short overview of the architecure of modern FPGAs, with a focus on the key differences to CPUs.

2.1. FPGA structure

A **F**ield **P**rogrammable **G**ate **A**rray is a logic device that was originally developed to add more flexibility and reconfigurability to gate logic devices like an ASIC[1] or a PLD[2]. In distinction from other logic devices a FPGA is not hard-wired so the user has the ability to to configure the FPGA to his specific needs. The structure of a FPGA can be described as unwired transistor logic blocks that can be freely progammed to yield any desired logical function. Modern FPGA devices have additional functional blocks like DSP- or memory-blocks (called BRAM on a Virtex-6 FPGA) which can be integrated into a design. These special function blocks, included on the device are directly accessible from the wired logic. This means BRAM blocks can be accessed without time costely operation of a communication protocol between a calculation kernel and memory of a PC host or other external hardware. The basic structure of a FPGA is shown in figure 1. The I/O-Blocks depicted in figure 1 connect the logical core of the FPGA with other devices.

Another huge advantage of FPGA's and other programmable hardware devices (ASICs, PLDs) over other classes of hardware is the capability of processing given tasks in parallel and not sequentially like CPUs, DSPs or Micro Controller do. On a FPGA it is possible to compute massive numbers of instructions in parallel, within one clock cycle. To benefit from this advantage it is crucial to implement an algorithm that processes as many parallel instructions as possible. It has been shown that a optimized parallel FPGA algorithm is always superior to a similar sequential algorithm in means of calculation time and maximum clock rates

[1]**A**pplication **S**pecific **I**ntegrated **C**ircuit
[2]**P**rogrammable **L**ogic **D**evice

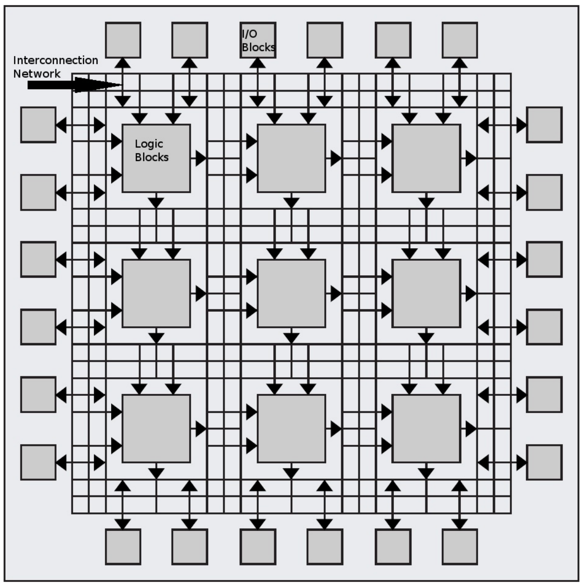

Figure 1: *FPGA structure.*

[34], [35], [36], [37]. Although the FPGA Structure is predestined for parallel processing, sequential statements can also be realized with a Finite State Machine (FSM).

2.2. Programming a FPGA

Another main difference of a FPGA chip compared to other logic devices is the possibility of programming the device from a Personal Computer via a vendor specific programming tool. Program designs that run on a FPGA are written in a special Hardware Description Language (HDL) such as VHDL or Verilog. As the name already implies, this programming language allows the user to model and program hardware components on the device in software, using specially defined constructs. This programming language contains high-level constructs like If-, For-, While-loops and low level aspects like bitwise declaration of signals, variables and constants. The main difference in comparison to other high-level languages is the capability of programming concurrent instructions.

3. METHOD

As mentioned before, in physical modelling it is desireable to to calculate all physical and geometrical properties of a specific musical instrument with the highest accuracy possible. The goal of this work is to implement such accurate formulations of physical models of musical instruments in real-time. To achieve this, a FDM algorithm developed in MATLAB is translated to VHDL and implemented on a FPGA development board. There are several works that exemplify that transient behaviour of musical instruments can effectively be modelled with an FDM formulation in an explicit form [5], [2], [38]. When certain requirements are met, the FDM has a high numerical stability over a large frequency range [39], [5] a property that is crucial when developing FDM-models of musical instruments. In the following section we develop a basic formulation of the implemented algorithm and show some modifications for the final real-time implementation on a FPGA chip.

3.1. FD Formulation

The basic algorithm for all presented models can best be illustrated by a discrete approximation of the 1-dimensional wave equation of the linear string. The analytical wave equation for a ideal string without damping effects and stiffness is given by:

$$u_{tt} = c^2 \cdot u_{xx} \tag{1}$$

with $c = \sqrt{\frac{T}{\sigma}}$ where T = tension and σ the linear density. The subscripts u_{tt} and u_{xx} indicate a differentiation by time and displacement respectively. Using Newtons second law of motion where the force acting on a conservative system can be expressed as

$$\text{Force} = M \cdot a = M \cdot u_{tt} = -\nabla V(u(t)) \tag{2}$$

with M the mass, $u(t) = (\mathbf{u}_1(t), ..., \mathbf{u}_N(t)))$ a position vector and V the potential function depending on the position.

Combining equation 1 and 2 gives a formulation for the acceleration $a = u_{tt}$ depending on u. The discrete solution for equation 1 is calculated by a semi-implicit Euler method also known as Newton-Stormer-Verlet (NSV)[3] algorithm. The NSV algorithm approximates differential equations of the following form

$$\frac{\delta u}{\delta t} = f(t, v) \tag{3}$$
$$\frac{\delta v}{\delta t} = f(t, u) \tag{4}$$

with u = the position and v = the velocity of a system.

The basic iteration of the recursive algorithm is

$$v_{t+\Delta t} = v_t + a_t \cdot \Delta t \tag{5}$$
$$u_{t+\Delta t} = u_t + v_{t+\Delta t} \cdot \Delta t \tag{6}$$

With Δt beeing the discrete time step .

This two step algorithm has several features suiting the calculation of physical models of musical instruments very well. Among its conditional stability and accuracy as shown by Hairer et al.[39] the algorithm gives explicit expressions for the velocity v, the position (deflection) u and the acceleration a at every point of the discretized geometry. In other words, three important physical parameters governing the vibrational behaviour of most musical instruments are calculated and directly accessible at every time step Δ_t. The accuracy of the solution now depends on the discretization steps in space and time and the precision of the physical model. The algorithm in pseudocode[4]:

Listing 1: NSV-Algorithm

```
a[t]=initial potential at t==0
for t=1 to SampleLength
{
  for every discrete point of the string
  {
```

[3]For the later implementation on the FPGA features of the Velocity Verlet [40] algorithm are added

[4]Remark: *The asterisks denote a multiplication and are used for better readability of the pseudo code*

```
    v[t+1]=v[t]+a[t] * delta_t;
    u[t+1]=u[t]+v[t+1] * delta_t;
    a[t+1]=potential function at t+1
  }
}
```

The potential function depends on the geometry and the given equation for the model. In the case of the linear string it can be approximated by discretizing the right hand side of equation 1 with a Finite Difference term:

$$u_{xx} \approx \frac{-2 * u_x + u_{x+1} + u_{x-1}}{\Delta x^2} \tag{7}$$

Summarizing the resources of this algorithm for one discrete point on the ideal string:

Table 1: *Resources of the MATLAB NSV version*

Multiplications/Fractions	Additions
5	4

3.2. FPGA Optimizations

When designing DSP algorithms in Hardware, fundamental design strategies depend on the data type. In most DSP applications based on IIR- or FIR-Filter designs a floating point data type is chosen. This data type has some advantages compared to a fixed point data type like lower resource usage for the typical Multiply and Accumulate (MAC) opperations but on the other hand can lead to stability problems due to a increased sensitivity to rounding or truncation errors [41] especially in IIR-Filter or other designs with a recursive structure [21]. To circumvent these problems a normalised fixed point data type is utilised in this work and used for all proposed instrument models on the FPGA. When translating code from a high level programming language into VHDL the main task is to optimize the code for parallel execution to fully utilize the advantages of the FPGAs hardware structure. Other optimizations can be achieved when data type specific properties are utilised as shown below: In binary logic a multiplication with a number k with the properties

$$k := |k| \in 2^x \text{ with } x \in \mathbb{Z} \tag{8}$$

can be carried out as a leftshift if $k > 0$ or a rightshft if $k < 0$. This property can be applied with the used fixed-point data type of the models, so the 5 multiplications/fractions of the MATLAB/C code are transformed to 4 left-/right-shifts and one multiplication in the optimized FPGA code. The used resources of the algorithm in hardware are:

Table 2: *Resources of the FPGA NSV version*

Multiplications	Additions	Shift Operations
1	4	4

Because of the fact that FPGA shift operations are much less time and resource consuming compared to multiplications and fractions the optimized hardware algorithm performs faster.

3.3. Complete Geometries

The described algorithm can easily be extended to higher dimensions and higher orders of the wave equation as described in section 4. Beside the 1-dimensional wave equation for the string, the models of the three instruments include the 2-dimensional wave equation of the membrane, a 2-dimensional equation for wood plates, a 2-dimensional equation for a wooden bridge and the 3-dimensional wave equation for the air. As mentioned in section 1 besides an accurate discrete model for the singular parts of the instrument, the coupling between these parts is of huge importance. In many cases the manner of the coupling imparts instruments with their specific sound caracteristic, for instance the position and mass of a banjo bridge strongly influences the timbre of the instrument [42]. In conclusion to get a realistic physical model of a complete musical instrument, the coupling parameters must be modelled meticiously. In some cases, like the sound radiation from a membrane and the influence of the air back onto the surface, the coupling can be done via a force coupling of both wave equations [4]. In other cases, like the coupling between a string and a wooden bridge, it has to be done via a reciprocal influence of the respective potentials at the interaction point or a mass coupling [43] respectively. Due to the fact algorithm 5 yields explicit expressions for the deflection, the velocity and the acceleration on every point of the discretized geometry, coupling between different parts can be expressed as functions, in some cases non-linear, of these quantities at the interaction point(s).

3.4. Hardware Implementation

After a complete implementation of the models in MATLAB, the algorithm is converted to VHDL and flashed onto a FPGA development board as described in section 1. The used hardware include a XILINX ML6O5 Virtex-6 development board for the calculation of the implemented models and a standard Personal Computer where a C#-program acts as a user interface for controlling the parameters of the physical model. The block diagram of the Banjo model depicted in figure 2 gives a schematic overview of the core functionality.

3.4.1. Parallel Computation Kernels

To fully utilise the processing capabilities of the FPGA all core elements of the FDM model are calculated in parallel. The calculation kernel of the 1-dimensional wave equation consists of 20 discrete points that are calculated in parallel, in other words if a string is discretized with eighty points the kernel is executed four times. The other entities are modelled similarly. The parallel membrane kernels have a 8x8 grid size, this means for a membrane discretized with 64x64 nodes, the parallel membrane kernels runs 8 times. A similar approach for 3-dimensional structures can be found in [32], [27].

3.4.2. Hardware Resources

The utilisation of hardware resources and clock speed on a FPGA directly depends on the complexity of the implemented design. On a Virtex-6 FPGA one string including a bow model, as described in 4, discretised with 80 points uses approximately 3% of the FPGAs resources and runs with a clock speed of $\approx$ 80 Mhz. More complex designs, like a complete Violin geometry consume nearly 85% of the resources, but still run with a clock speed of $\approx$ 80 Mhz.

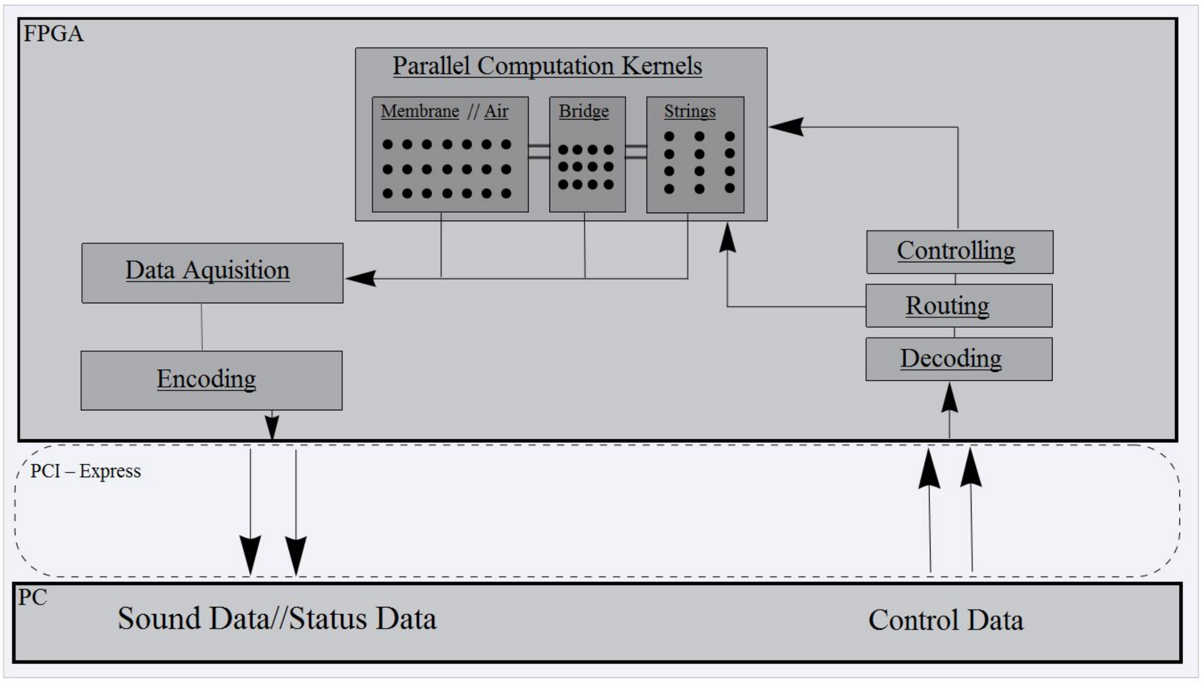

Figure 2: *A schematic overview of the described model. The black dots symbolize the parallel calculation kernels*

4. RESULTS

In the following section a description of three FDM instrument models implemented in real-time on a Virtex-6 ML605 FPGA board is given. Sound examples for all three models can be found on the website of the Institute for Systematic Musicology Hamburg .

4.1. Five string Banjo

The first instrument model that is implemented on the FPGA is a American Five String Banjo. Due to its comparatively plain geometrical features (a string coupled to a membrane), a model of the banjo is a instructive starting point for modelling musical instruments. The current model consists of two strings, a wooden bridge, the membrane and the coupling air underneath the membrane. The FPGA model of the banjo is devided into four entities:

1. The strings (1-dimensional Wave Equation)
2. The membrane (2-dimensional Wave Equation)
3. The air (3-dimensional Wave Equation)
4. The bridge (2-dimensional Equation)

As mentioned in section 3 each entity has a parallel computation structure that can best be illustrated on the string-enity. The string is dicretized into N points (N is dependant on the length of the string and the desired accuracy). In the next step, the string is fragmented into the maximal length of the parallel computation kernel n_p. In the case of the banjo string $N = 80$ and $n_p = 20$. This means that for the calculation of one timestep Δt_s of the whole string the parallel kernel runs 4 times.

4.1.1. Non-Linearities of the Membrane

In a recent work, done by the authors the nonlinear tension distribution of a Banjo membrane was measured and calculated qualitatively [13]. It was shown that these non-linearities arise from the exerted force of the bridge and the non-symmetrical tension distribution at the tension hoop of the Banjo head. The measured tension distribution on the mebrane of the authors banjo is depicted in figure 3. In the current design, the membrane of the Banjo physical model is discretized with 32x32 grid points with the non-linear tension distribution mapped on every point. The modeshapes of a membrane with non-linear tension and the modeshapes of a real banjo can be found in figure 5 in Appendix A.

4.2. Ruan

The Ruan is one of the oldest Chinese string instruments and is often rederred to as the predecessor of the Pi'Pa. The playing style resembles the plucked tremolo playing style of the Italian Mandolin. The front- and back-plate of the Ruans round body is made of *Paulownia tomentosa* wood. Additionally there are two orifices on the front-plate, acting as sound holes. So the model of this instrument extends the Banjo model in several ways, instead of the two-dimensional, second order wave equation of the membrane, the sound radiating front-plate is now modelled as a round wooden plate.

4.2.1. Orthotropic Wood Plate

The front plate is modelled as a modified form of the fourth-order Kirchhoff Plate Equation and is dicretized with 64x64 points. The two frontal orifices are implemented into the formulation of the front plate, coupling the air volume inside the cavity (instrument

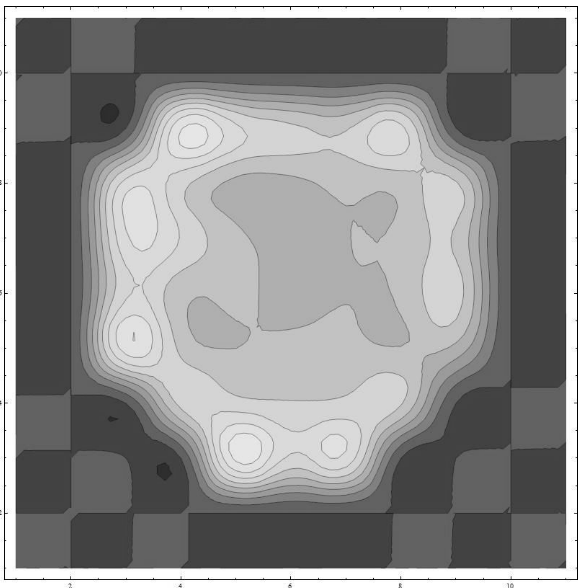

Figure 3: *Non-linear tension distribution on a membrane (lighter colors indicates higher tension)*

body) to the air outside of the instrument. The orthotropic quality of wood can be accounted for with a different Young's modulus in each of the three grain directions [44] which directly influences the transversal wave speed in the respective direction. The specific material property of the wood is considered in the model of the Ruan and the Violin.

4.3. Violin

The building blocks of the physical model of the violin are essentially the same as in the other two instrument models: a wooden resonance body, several metal strings and a wooden bridge. In comparison to the Ruan and the Banjo, two plucked lutes, obviously a major difference of the Violin model is the excitation mechanism of the string. The implemented model of the Violin bow is based on a bow string FDM-model by Bader [38] that has been extended in several ways for the real-time version on the FPGA.

4.3.1. String-Bow Interaction Model

The typical and well known Helmholtz motion of the Violin string is caused by the nonlinear excitation by the Violin bow. The interaction of the bow and the string can be described by a stick-slip model [45] dividing the excitation of the Violin string into two phases:

1. the strings sticks to the bow and is teared in the bowing direction
2. the strings slips off the bow until it sticks to the bow again.

As shown in [38] the stick-slip motion of the bow-string system can be controlled by three physical parameters: the bow velocity, the bow pressure and the bow position on the string. A large expressive range of the violin excitation can be simulated fully with the variation of these three parameters. In addition to these parameters, the real-time model of the bow-string interaction includes a variable for the bow stiction (amount of rosin on the string) and the number of points of contact of the bow on the string [11]. Thereby the amount of the bow-noise in the produced sound and the response of the violin string can be regulated. A diagram of the bow-string model can be found in figure 4 in Appendix A.

5. CONCLUSIONS

In this work we have presented three physical models of musical instruments with non-linear material properties, non-linear coupling or non-linear excitation mechanisms. All three instruments have been implemented on a FPGA development board and can be configured, modified and played in real-time. The intensive work with the three mentioned instrument models has already lead to many interesting insights into the vibrational behaviour and physical properties of each instrument, for instance the importance of non-linearities in the tension of the Banjo membrane. Further work will include the implementation of a MIDI protocol and the development of other instrument models. Eventhough the implementation time for physical models on a FPGA are higher than for C or MATLAB implementations, the capabilities of a real-time physical model highly outweigh the initial work.

6. ACKNOWLEDGMENTS

This work was supported by the Deutsche Forschungs Gesellschaft (DFG).

7. REFERENCES

[1] Rolf Bader, "Physical modeling of the self-sustained oscillation of a labium using molecular dynamic simulation techniques," *Proceedings ISMA (International Symposium of Musical Acoustics)*, 2007.

[2] Rolf Bader, *Computational Mechanics of the Classical Guitar*, Springer, October 2005.

[3] Rolf Bader, "Fine tuning of guitar sounds with changed top plate, back plate and rim geometry using a whole body 3d finite-difference model," *Forum Acusticum joined with American Acoustical Society Paris 08*, pp. 5039 – 5044, 2008.

[4] N. Fletcher and Th. Rossing, *Physics of Musical Instruments*, Springer, 2000.

[5] Stefan Bilbao, *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics.*, John Wiley and Sons, Chichester, UK, 2009.

[6] C. Erkut, M. Karjalainen, P Huang, and V. Valimäki, "Acoustical analysis and model-based sound synthesis of the kantele," *J. Acoust. Soc. Am.*, vol. 112, no. 4, pp. 1681 – 1691, 2002.

[7] Georg Essl, Stefania Serafin, Perry R. Cook, and Julius O. Smith, "Musical applications of banded waveguides," *Comput. Music J.*, vol. 28, no. 1, 2004.

[8] F. Pfeifle and R. Bader, *Musical Acoustics, Neurocognition and Psychology of Music*, chapter Real-Time Physical Modelling of a real Banjo geometry using FPGA hardware technology., pp. 71–86, Rolf Bader, Frankfurt am Main, Germany, 2009.

[9] F. Pfeifle and R. Bader, "Real-time virtual banjo model and measurements using a microphone array.," *J. Acoust. Soc. Am.*, vol. 125, no. 4, pp. 2515 – 2515, 2009.

[10] F. Pfeifle and R. Bader, "Membrane modes and air resonances of the banjo using physical modeling and microphone array measurements.," *J. Acoust. Soc. Am.*, vol. 127, no. 3, pp. 1870 – 1870, 2010.

[11] F. Pfeifle and R. Bader, "Real-time finite-difference string-bow interaction field programmable gate array (fpga) model coupled to a violin body," *J. Acoust. Soc. Am.*, vol. 130, no. 4, pp. 2507 – 2507, 2011.

[12] F. Pfeifle and R. Bader, "Measurement and physical modelling of sound hole radiations of lutes," *J. Acoust. Soc. Am.*, vol. 130, no. 4, pp. 2507 – 2507, 2011.

[13] F. Pfeifle and R. Bader, "Nonlinear coupling and tension effects in a real-time physical model of a banjo," *J. Acoust. Soc. Am.*, vol. 130, no. 4, pp. 2432 – 2432, 2011.

[14] F. Pfeifle and R. Bader, "Measurement and analysis of sound radiation patterns of the chinese ruan and the yueqin," *J. Acoust. Soc. Am.*, vol. 131, no. 4, pp. 3218 – 3218, 2012.

[15] K. Veggeberg and A. Zheng, "Real-time noise source identification using programmable gate array(fpga) technology," *Proceedings of Meetings on Acoustics*, vol. 5, 2009.

[16] C Hao and W. Ping, "The high speed implementation of direction-of-arrival estimation algorithmo," *International Conference on Communication, Circuits and Systems and West Sino Expositions*, vol. 2, pp. 922 – 925, 2002.

[17] B. Von Herzen, "Signal processing at 250 mhz using high-performance fpga's," *IEEE Transactions onvery large scale integration (VLSI) Systems*, vol. 6, no. 2, 1998.

[18] P. Chen, X. Tian, Y. Chen, and X. Yang, "Delay-sum beamforming on fpga," *ICSP 2008 Proceedings*, pp. 2542 – 2545, 2008.

[19] Z. Wang, R. Jin, J. Geng, and Y. Fan, "Fpga implementation of downlink dbf calibration," *Antennas and Propagation Society International Symposium*, 2005.

[20] G. Martins, M. Barata, and L. Gomes, "Low cost method to reproduce sound with fpga," *IEEE International Symposium on Industrial Electronics 2008, ISIE 2008*, 2008.

[21] Uwe Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Springer, Berlin, Heidelberg, 2 edition, 2007.

[22] Juergen Reichardt and Bernd Schwarz, *VHDL-Synthese: Entwurf digitaler Schaltungen und Systeme*, Oldenbourg, Muenchen, 4 edition, 2009.

[23] A. Madanayake, L. Bruton, F. Comis, and C. Comis, "Fpga architectures for real-time 2d/3d fir/iir plane wave filters," *Proceedings of the 2004 International Symposium on Circuits and Systems ISCAS 2004*, vol. 3, 2004.

[24] Kai Shuang, X. Yankai, J. Shan, and Z. Hongwu, "Converting analog controllers to digital controllers with fpga," *ICSP2008 Proceedings*, 2008.

[25] O. Maslennikow and A Sergiyenko, "Mapping dsp algorithms into fpga," *Proceedings of the International Symposium on Parallel Computing in Electrical Engineering*, 2006.

[26] T. Brich, Novacek, K., and A. Khateb, "The digital signal processing using fpga," *ISSE 2006, 29th International Spring Seminar on Electronics Technology*, pp. 322 – 324, 2006.

[27] E. Motuk, R. Woods, S Bilbao, and J. McAllistere, "Design methodology for real-time fpga-based sound synthesis," *IEEE Transactions on signal processing*, vol. 55, no. 12, 2007.

[28] Wang Chen, Panos Kosmas, Miriam Leeser, and Carey Rappaport, "An fpga implementation of the two-dimensional finite-difference time-domain (fdtd) algorithm," in *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, New York, NY, USA, 2004, pp. 213–222, ACM.

[29] K. L. Shlager and B. Schneider, "A selective survey of the finite-diffrence time-domain literature," *IEEE Antennas and Propagation Magazine*, vol. 37, no. No. IEEE 1995, 1995.

[30] R. Strzdoka and D. Göddeke, "Pipelined mixed precision algorithms on fpgas for fast and accurate pde solvers from low precision components," *14th annual IEEE Symposium on Field Programmable Custom Computing Machinges*, 2006.

[31] M. Liu, W Kuehn, Z. Lu, and A. Jantsch, "System-on-an-fpga design for real-time particle track recognition in physics experiments," *11th Euromicro Conference on Digital System Design Architectures, Mthods and Tools*, 2008.

[32] E. Motuk, R. Woods, and S Bilbao, "Implementation of finite-differece schemes for the wave equation on fpga," *IEEE International Acoustics Speech and Signal Processing ICASSP 2005*, vol. 3, 2005.

[33] J.A. Gibbons, D.M. Howard, and A.M. Tyrrell, "Fpga implementation of 1d wave equation for real-time audio synthesis," *IEEE Proceedings, Computers and Digital Techniques*, vol. 152, no. 5, pp. 619 – 631, 2005.

[34] E. Becache, A. Chaigne, G. Derveaux, and P. Joly, "Fpga parallel implementation of cmac type neural network with on chip learning," *IEEE*, pp. 111 – 115, 2007.

[35] V. Subasri, K. Lavanya, and B. Umamaheswari, "Implementation of digital pid controller in field programmable gate array(fpga)," *Proceedings of India International Conference on Power Electronics 2006*, 2006.

[36] J. D. Lis and C. T. Kowalski, "Parallel fixed point fpga implementation of sensorless induction motor torque control," *13th Power Electronics and Motion Control Conference 2008. EPE-PEMC*, 2008.

[37] Z. Zou, W. Hongyuan, and Y. Guowen, "An improved music algorithm implemented with high -speed parallel optimization for fpga," *7th International Symposium on Antennas, Progpagation & EM Theory, ISAPE 2006*, 2006.

[38] Rolf Bader, "Whole geometry finite-difference modeling of the violin," *Proceedings of the Forum Acusticum 2005*, pp. 629 – 634, 2005.

[39] Ernst Hairer, Christian Lubich, and Gerhard Wanner, "Geometric numerical integration illustrated by the störmer/verlet method," *Acta Numerica*, vol. 12, pp. 399–450, 2003.

[40] Loup Verlet, “Computer "experiments" on classical fluids. i. thermodynamical properties of lennard-jones molecules,” *Phys. Rev.*, vol. 159, pp. 98–103, Jul 1967.

[41] David Goldberg, “What every computer scientist should know about floating-point arithmetic,” *ACM Computing Surveys*, vol. 23, pp. 5–48, 1991.

[42] Laurie A. Stephey and Thomas R. Moore, “Experimental investigations of an american five-string banjo,” *J. Acoust. Soc Am.*, vol. 125, no. 5, 2008.

[43] B.Z. Guo and W.D. Zhu, “On the energy decay of two coupled strings through a joint damper,” *Journal of Sound and Vibration*, vol. 203, no. 3, pp. 447 – 455, 1996.

[44] Ulrike G. K. Wegst, “Wood for sound,” *Am. J. Bot.*, vol. 93, no. 10, 2006.

[45] Lothar Cremer, *Physik der Geige*, Hirzel, Stuttgart, Germany, 1981.

8. APPENDIX A

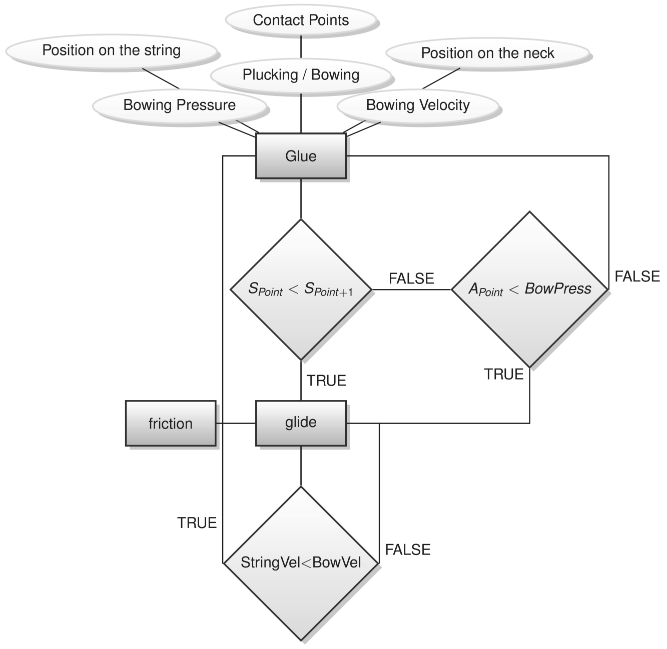

Figure 4: *Bow-string interaction model.*

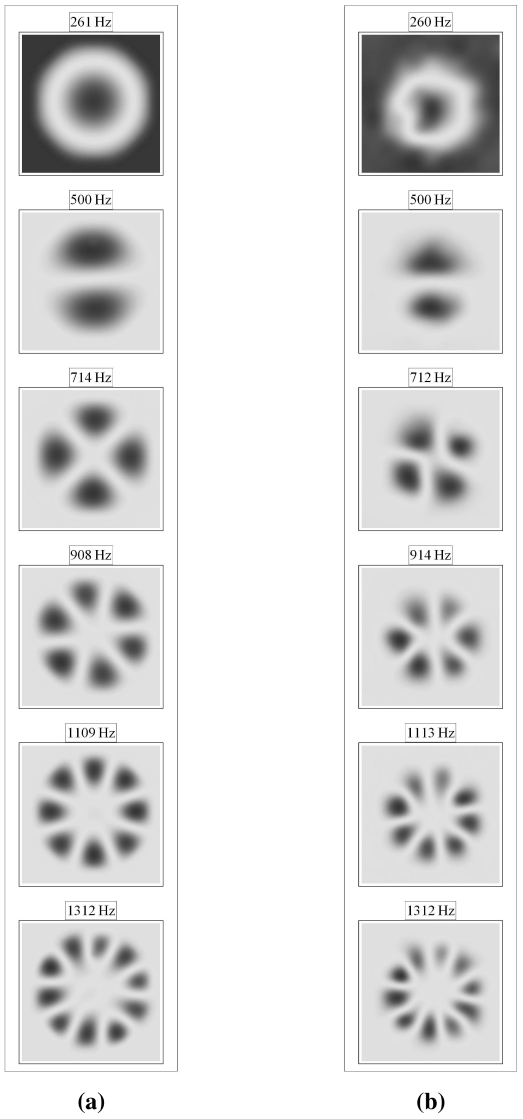

Figure 5: a) Simulated membrane. b)Measured membrane