SoC Architecture for Real-Time Interactive Painting based on Lattice-Boltzmann

Domien Nowicki, Luc Claesen
Hasselt University
3590 Diepenbeek, Belgium
firstname.lastname@student.uhasselt.be, firstname.lastname@uhasselt.be

***Abstract*—A new SoC-based hardware architecture for interactive watercolor paint simulation based on the lattice Boltzmann fluid dynamics simulation model is presented. A scalable proof-of-concept FPGA based hardware prototype achieves a VGA framerate of 60 fps with a resolution of 160x120.**

***Index Terms*—Digital Painting, Lattice Boltzmann, Fluid Simulation, FPGA, VLSI, SoC, Active Canvas, Real-Time, Interactive.**

I. Introduction

In recent years, advanced models for digital painting have been developed that try to realistically mimic the traditional painting process [1], [2], [3], [4]. The complex interaction between the paint medium, canvas and brush is simulated with realistic physically based rules to provide a digital equivalent. Recently, real brushes can be used to provide even more realistic input [5], [6]. Advantages over the traditional painting process include a clear digital representation, undo support, saving and loading intermediate images, experimenting with different paint media, and more control over paint flow.

When adding paint to a digital canvas, the paint fluid can start to flow, following physically based fluid dynamics such as diffusion and advection. A digital canvas is usually divided into a regular grid of cells containing water and pigment concentrations. Eventually the water will evaporate, and leave dried colored strokes. This process is depicted in Fig. 1 and is called “active canvas”.

An active canvas will try to adhere to the macroscopic Navier-Stokes equations for fluid dynamics [3], [4]. When implicit integration methods like backward Euler are used to solve the systems of differential equations, it requires several Newton-Raphson iterations [7] over the whole fluid field. Thus they are computationally intensive and require a lot of memory access, limiting real-time simulation.

To achieve acceptable simulation speed, existing active canvas methods utilize GPU processing and a coarser simulation grid size than the actual canvas.

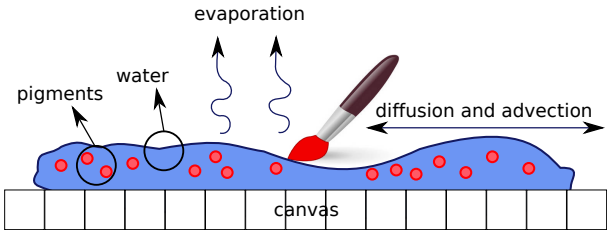

Fig. 1. Active canvas approach modelling physical fluid dynamics.

A new SoC based hardware architecture is presented that can achieve interactive simulation speed at larger grid sizes using a specialized parallel pipeline. The solution is based on the lattice Boltzmann model [8], [9], that solves the incompressible fluid dynamic system equations on a mesoscopic level in a single iteration, making use of basic laws of physics such as the preservation of momentum, energy and mass. The model was first introduced and extended for Eastern ink painting by Chu et al. [2]. The floating point implementation has been redesigned using fixed point arithmetic to enable an efficient solution in hardware. A satisfactory accuracy can be achieved using a word length of 12 bits.

Summary of Contributions: A fixed point lattice Boltzmann simulation pipeline for fluid dynamics and a real-time interactive active paint canvas hardware architecture for watery paint is presented.

In section II related work is discussed, and in section III the lattice Boltzmann hardware approach is described. The watercolor paint system hardware architecture is described in section IV and the prototype implementation is discussed in section V. Finally, conclusions are presented in section VI.

II. Related work

Sano et al. [10] present an FPGA based hardware accelerator for simulating the lattice Boltzmann model. They plan on using their accelerator for computing 3D thermal flows. Their architecture continuously streams simulation data from a host PC over a PCI-Express bus and uses floating point operations. Unfortunately this limits the maximum possible throughput. This is addressed in this paper by using fixed point arithmetic and maintaining the entire grid in on-board external memory, directly accessible by the SoC.

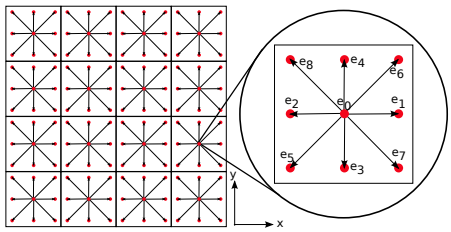

Fig. 2. The D2Q9 lattice Boltzmann model. Left: Regular lattice. Right: Close-up of a cell.

III. Lattice Boltzmann Hardware Fluid Simulation

The lattice Boltzmann idea is to model fluid flow using a simplified particle model arranged in a regular lattice of cells. Fluid particles moving with arbitrary velocities are modelled by a set of distribution functions in specific directions for each cell. At each cell x and time t, the distribution function $f_i(x,t)$ describes the expected number of particles moving in the direction of vector $\vec{e_i}$. In Fig. 2, the arbitrary particle velocities are divided into a discrete set of 9 directions, $\vec{e_0}$ through $\vec{e_8}$, with $\vec{e_0}$ as the stationary particle direction. This particular 2D lattice Boltzmann configuration is called D2Q9.

At each timestep Δt, two operations are performed for each cell: the f_i's are propagated to their neighbour cells following their direction $\vec{e_i}$, and the arriving f_i's at the same cell are subjected to the collision operator. The collision operation redistributes the colliding f_i's in a cell to their equilibrium distribution function $f_i^{(eq)}$. The two operations of propagation and collision are formulated mathematically:

$$f_i(x+\vec{e_i}\Delta t, t+\Delta t) = (1-\omega)f_i(x,t) + \omega f_i^{(eq)}(x,t) \qquad (1)$$

Here, ω is the relaxation parameter to modulate between the new and old fluid state. For simplicity, ω was chosen to be 1 and therefore the left term was ignored. Like Chu et al. the incompressible lattice Boltzmann variant of He and Luo [9] is used, which minimizes the compressible effect. They present an equilibrium distribution function with fluid diffusion in the left term and advection in the right term:

$$f_i^{(eq)} = w_i\left[\rho + \rho_0\left(\frac{3}{c^2}\vec{e_i}\cdot\vec{u} + \frac{9}{2c^4}(\vec{e_i}\cdot\vec{u})^2 - \frac{3}{2c^2}\vec{u}\cdot\vec{u}\right)\right] \qquad (2)$$

Here, $c = \Delta x/\Delta t$, with Δx the lattice spacing, w_i are weights determined by the lattice geometry, ρ is the fluid density, ρ_0 is a predefined average fluid density and $\vec{u}$ is the fluid velocity. For simplicity, the parameters are defined as follows: $c = \Delta t = \rho_0 = 1$. The weights w_i are $4/9$ for $i = 0$, $1/9$ for $i = 1...4$ and $1/36$ for $i = 5...8$. The fluid density ρ and velocity $\vec{u}$ for a cell is:

$$\rho = \sum_{i=0}^{8} f_i \qquad (3) \qquad\qquad \vec{u} = \frac{1}{\rho_0}\sum_{i=1}^{8} f_i \cdot e_i \qquad (4)$$

For simulating the collision operation in hardware, cells need to be updated with their new equilibrium particle density functions $f_i^{(eq)}$:

$$\begin{aligned}
f_1^{(eq)} &= w_1*[\rho+3ux+4.5uxx-1.5uu] && (5)\\
f_2^{(eq)} &= w_2*[\rho-3ux+4.5uxx-1.5uu] && (6)\\
f_3^{(eq)} &= w_3*[\rho-3uy+4.5uyy-1.5uu] && (7)\\
f_4^{(eq)} &= w_4*[\rho+3uy+4.5uyy-1.5uu] && (8)\\
f_5^{(eq)} &= w_5*[\rho-3(ux+uy)+4.5(uxx+uyy+uxy)-1.5uu] && (9)\\
f_6^{(eq)} &= w_6*[\rho+3(ux+uy)+4.5(uxx+uyy+uxy)-1.5uu] && (10)\\
f_7^{(eq)} &= w_7*[\rho+3(ux-uy)+4.5(uxx+uyy-uxy)-1.5uu] && (11)\\
f_8^{(eq)} &= w_8*[\rho-3(ux-uy)+4.5(uxx+uyy-uxy)-1.5uu] && (12)\\
f_0^{(eq)} &= \rho - \sum_{i=1}^{8}\hat{f}_i && (13)
\end{aligned}$$

Here, ux and uy are the x and y component of the fluid velocity vector $\vec{u}$, and:

$$\begin{aligned}
uxx &= ux\cdot ux \qquad & uxy &= 2\cdot ux\cdot uy\\
uyy &= uy\cdot uy \qquad & uu &= uxx+uyy
\end{aligned}$$

The $f_i^{(eq)}$ equations are calculated each time step using fixed point arithmetic. A distribution function f_i is represented using 12 fractional bits and provides satisfactory results for the paint system.

Note that these equations share common products which only need to be calculated once. The constants used for the weights w_i, and for the other terms can be recoded using canonical signed digits (CSD) [11], reducing their multiplication to an efficient combination of a few additions, subtractions. Shifts can be done at no cost in hardware. For example, the constant w_1 ($= 1/9 \simeq 455$ in 12-bit fixed point) recoded in CSD needs only 3 operations:

$$x\cdot 455 = x\cdot 2^9 - x\cdot 2^6 + x\cdot 2^3 - x\cdot 2^0$$

Since each f_i incurs a small inaccuracy due to fixed point arithmetic, f_0 is based on the other f_i's to ensure the total fluid mass is conserved. Only 3 real multiplications are required in total, together with a few additions and subtractions.

The collision operation presented here only depends on the new fluid density ρ and the velocity vector $\vec{u}$, that are calculated from the propagated f_i's.

For simulating the propagation operation in hardware, information from the neighbour cells is needed. Only the direct neighbours of a cell are needed, so a sliding window approach was chosen. The sliding window approach can be seen in Fig. 3. The window is represented by a 3x3 matrix of registers to allow simultaneous access in hardware to each direct neighbour cell. The sliding is performed by a long FIFO shiftregister, shifting in a cell at a time, with 3 taps for feeding the window. Each shift moves the window horizontally one column ahead. In this approach, each cell of the simulation grid only needs to be looked up once from memory since the shift register acts as a cache for future cell lookups. This

allows an efficient use of memory bandwidth. The shiftregister contains $3N$ cells, with N the width of the simulation grid. Using the window W, the particle distribution functions from the neighbour cells can be propagated as follows:

$$\bar{f}_i \quad = \quad W(-e_i)_{f_i} \tag{14}$$

Here, $\bar{f}_i$ is the propagated particle distribution function and $W(-e_i)_{f_i}$ is the particle distribution f_i of the cell $W(-e_i)$ of the window. This completes the discussion for simulating lattice Boltzmann in hardware.

IV. Paint System Hardware Architecture

The lattice Boltzmann water simulation is extended to a watercolor paint simulation by incorperating a pigment simulation, similar to Chu et al.

The paint simulation is currently based on a single layer paper model, that mostly resembles the flow layer of the paper model used by Chu et al. Water can be added to the canvas by a virtual brush and will flow and spread by the water simulation. In addition, the virtual brush can carry pigments that will be deposited on the canvas. These pigments will move with the flow of water and is simulated by the pigment simulation. The output of the pigments is rendered to a VGA output. This architecture is depicted in Fig. 4.

A. Water simulation

At each timestep, if a cell is affected with additional water by the virtual brush then the fluid density in the cell is increased by the amount of water on the virtual brush. This step is performed when calculating the fluid density. The lattice Boltzmann approach is also extended with variable permeability, boundary pinning and evaporation.

Variable permeability allows the creation of interesting water flow patterns, and can be realized by dynamically blocking the lattice Boltzmann propagation step. This reduces the flow of some particle distribution functions f_i, that bounce back in the opposite direction [8]:

$$K_i \quad = \quad \frac{W(0,0)_K + W(-e_i)_K}{2} \tag{15}$$

$$\bar{f}_i \quad = \quad K_i \cdot W(0,0)_{f_v} + (1 - K_i) \cdot W(-e_i)_{f_i} \tag{16}$$

$$= \quad \frac{K_i \cdot W(0,0)_{f_v}}{F} + W(-e_i)_{f_i} - \frac{K_i \cdot W(-e_i)_{f_i}}{F} \tag{17}$$

Here, f_v is the particle distribution in the opposite direction of e_i, K is the blocking factor of a cell and F is the fixed point accuracy of K. A cell is augmented with a dynamic blocking factor that changes over time. By averaging the blocking factors of the participating cells an equal propagation is ensured in both directions. The blocking mechanism is also used to prevent water from flowing off the simulation grid. Equation (16) can lead to loss of mass ρ when using fixed point arithmetic due to truncation errors, and is corrected in (17). Currently the prototype uses a small set of constant blocking factors K to avoid using embedded multipliers. In the prototype these blocking factors K are calculated based on the boundary pins. Boundary pinning is used to model the rough

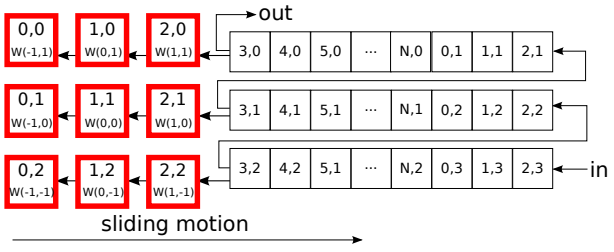

Fig. 3. Sliding window approach. Left: 3x3 matrix window. Right: Long FIFO shiftregister with 3 taps.

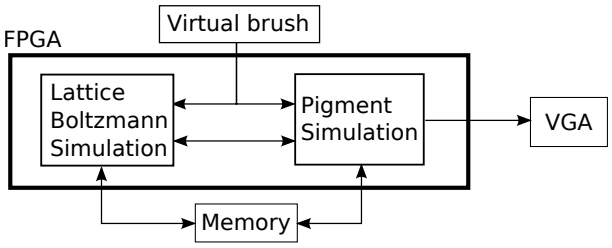

Fig. 4. Architecture of the watercolor paint system.

spread of water, and uses pinning sites to obstruct movement of water. Pinning sites are gradually depinned to allow the water to spread. Simple rules are used to determine pinning sites: a cell is a pinning site if it is dry and the fluid density of its direct neighbours are below a threshold. To enable pinning, the blocking factor of that cell is set to a very high value, preventing water to go through that cell.

Water evaporation is modelled by reducing the fluid density ρ of each cell by a small amount each timestep. An extra loss of water occurs at pinned sites, to induce migration of pigments towards the boundary resulting in a subtle darkened edge. This can be realized by reducing the particle distribution functions f_i that bounce back.

B. Pigment simulation

For the pigment simulation, cells are augmented with pigment information. The CMY color model with cyan, magenta and yellow pigments is used for simplicity. Each pigment uses 10 bits, matching the word length of the prototype VGA DAC. Each timestep, a cell is checked if it is affected by new pigments coming from the virtual brush, and those pigments are added to the cell. In the current prototype, brush pigments simply overwrite the current pigment in a cell.

Pigments on the canvas are mainly affected by the flow of water and a simple advection scheme is used to simulate the movement of pigments. For simulating pigment advection, pigments are propagated to their direct neighbours, similar to the lattice Boltzmann propagation step. The amount of pigment propagated to the neighbours is relative to the particle distribution functions f_i of a cell. This is expressed as follows:

$$\hat{g} \quad = \quad W(0,0)_g \left(1 - \sum_{i=1}^{8} W(0,0)_{f_i}\right) + \sum_{i=1}^{8} W(-e_i)_{f_i} \cdot W(-e_i)_g \tag{18}$$

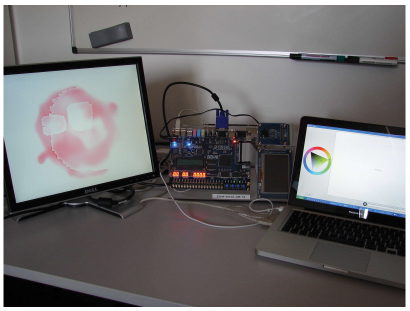

Fig. 5. Prototype setup.

Here, $\hat{g}$ is the resulting pigment after advection and W_g represents the pigment of a cell in the window. The left term represents the pigment that remains after propagation, and the right term represents the incoming pigment from the neighbour cells. This equation is calculated in fixed point arithmetic for each of the CMY pigments. Since a subtractive color scheme is used, darkened colors will appear when different pigments clump together. More experimentation is being done to get a more expected linear blend when advecting pigments.

V. Prototype Implementation

A scalable proof-of-concept hardware prototype for interactive watercolor paint simulation based on the lattice Boltzmann fluid dynamics simulation was developed on an Altera TerasIC DE2-70 FPGA board. The board contains a Cyclone II FPGA, and the onboard 2MB SSRAM was used for storing the 256-bit cells of the 160x120 simulation grid. The setup can be seen in Fig. 5. Fig. 6 shows the contents of a cell. And Fig. 7 provides an overview of the implemented HDL modules. The DSP elements are used as embedded multipliers.

The double scanline buffer needs to store two scanlines for the VGA output screen, and the sliding window needs to store three horizontal lines of the simulation grid. Thus the used blockram scales linearly with the width of the simulation grid and width of the VGA output screen. The combinationals and registers are mostly unaffected by the used resolution or framerate. The VGA output was scaled to 640x480 and achieves full VGA framerate at 60 fps using an internal clock of 25MHz. In the current implementation the memory size and bandwidth is the bottleneck, not the clock frequency.

VI. Conclusions

Detailed fluid dynamics and physical model based active canvas paint simulation systems require huge amounts of computation. A new and dedicated hardware SoC architecture is presented based on the lattice Boltzmann model for the real-time paint simulation. A prototype has been realized on a memory bandwidth limited educational FPGA board. Future work will focus on the development of a paint system based on new high-performance and high-bandwidth FPGA systems. The SoC architecture presented could also be extended for other fluid-dynamics based simulations, such as air-flow simulations for virtual wind-tunnels, simulations of water, smoke and fire in advanced graphics and more.

Description	Bits	Amount	Total
Particle distribution functions f_i	12	9	108
Pigment concentrations g	10	3	30
Boundary pin	1	1	1
Grid boundary cell	1	1	1
			140

Fig. 6. Bit information stored in a cell.

Module name	C	R	BRAM	DSP
dblscanline	26	0	9660	0
vgacontroller	86	20	0	0
ssram256controller	121	177	19904	0
simulator	5230	1860	44800	60
calculatecell	4910	248	0	60
windowshift	12	8	44800	0

Fig. 7. Logic elements usage. From left to right: Module name, used combinationals, registers, blockram bits, and DSP elements.

References

[1] W. Baxter, J. Wendt, and M. C. Lin, "Impasto: a realistic, interactive model for paint," in *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*. New York, NY, USA: ACM, 2004, pp. 45–148.

[2] N. S.-H. Chu and C.-L. Tai, "Moxi: real-time ink dispersion in absorbent paper," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*. New York, NY, USA: ACM, 2005, pp. 504–511.

[3] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin, "Computer-generated watercolor," in *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 421–430.

[4] T. Van Laerhoven and F. Van Reeth, "Real-time simulation of watery paint: Natural phenomena and special effects," *Comput. Animat. Virtual Worlds*, vol. 16, no. 3-4, pp. 429–439, 2005.

[5] P. Vandoren, L. Claesen, T. Van Laerhoven, J. Taelman, C. Raymaekers, E. Flerackers, and F. Van Reeth, "Fluidpaint: an interactive digital painting system using real wet brushes," in *ITS '09: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. New York, NY, USA: ACM, 2009, pp. 53–56.

[6] P. Vandoren, T. Van Laerhoven, L. Claesen, J. Taelman, C. Raymaekers, and F. Van Reeth, "Intupaint: Bridging the gap between physical and digital painting," in *TABLETOP-2008 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems*. IEEE, October 2008, pp. 65–72. [Online]. Available: http://dx.doi.org/10.1109/TABLETOP.2008.4660185

[7] J. Stam, "Stable fluids," in *Proceedings of SIGGRAPH 99*, ser. Computer Graphics Proceedings, Annual Conference Series, Aug. 1999, pp. 121–128.

[8] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University Press, USA, August 2001. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0198503989

[9] X. He and L.-S. Luo, "Lattice boltzmann model for the incompressible navier-stokes equation," *Journal of Statistical Physics*, vol. 88, no. 3, pp. 927–944, Aug 1997. [Online]. Available: http://dx.doi.org/10.1023/B:JOSS.0000015179.12689.e4

[10] K. Sano, O. Pell, W. Luk, and S. Yamamoto, "Fpga-based streaming computation for lattice boltzmann method," in *Proc. Field Programmable Technology*. IEEE, December 2007.

[11] H. De Man, L. Claesen, J. Van Ginderdeuren, and L. Darcis, "A structured multiplier-free digital filter building block for lsi implementation," in *ECCTD'80: Proceedings European Conference on Circuit Theory and Design*. Warsaw, 1980, pp. 527–532.