



Milkymist

The architecture for the open hardware world

Sébastien Bourdeauducq

December 2009

How it all started

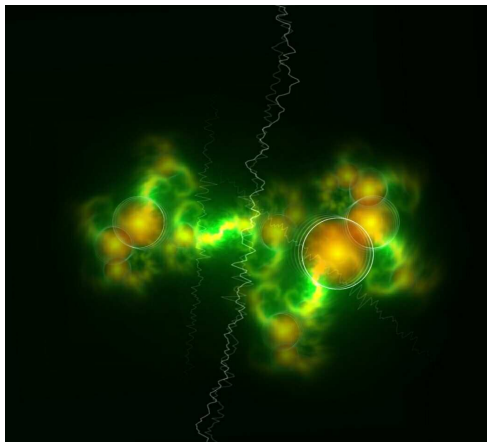
A device for video performance artists (VJs)...

- ▶ inspired by the popular MilkDrop program for PCs
- ▶ with many interfaces: MIDI, DMX, can also do video mixing
- ▶ highly integrated

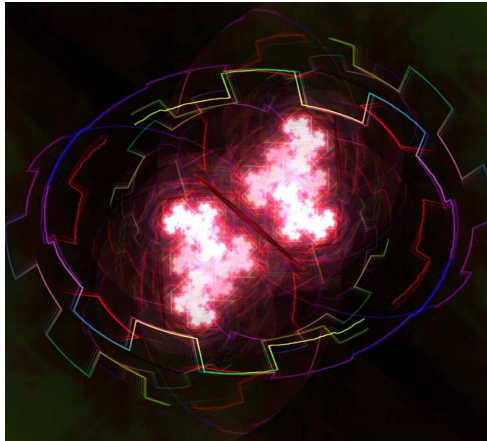
At the frontier between...

- ▶ big computers with software to render visual effects
- ▶ and small, handy microcontroller boards you connect to anything
(“Arduino” is today’s buzzword for those)

What is that MilkDrop thing?



What is that MilkDrop thing?



What is that MilkDrop thing?



Open Hardware

- ▶ Arduino = AVR + power supply + connectors.
- ▶ The AVR chip does all the magic! And it's a black box in every sense.
- ▶ OTOH, you could describe all the logic of the chip...
- ▶ ...and open source it!
- ▶ Ask Atmel for the same!
- ▶ Let's make Milkymist truly open hardware...

Truly Open Hardware projects

- ▶ GRLIB/LEON3 (Aeroflex Gaisler, ESA)
- ▶ OpenSPARC (Sun Microsystems)
- ▶ Open Graphics
- ▶ Project VGA
- ▶ ...and others
- ▶ Let's do the same and build our own SoC design for the project!

How is MilkDrop working?

Iterative rendering:

- ▶ Take the current image, and distort it
 - ▶ zoom
 - ▶ rotation
 - ▶ scaling
 - ▶ others...
- ▶ Draw waves and shapes
- ▶ (In Milkymist: overlay some live video)
- ▶ Display the result
- ▶ Repeat the process!

(This is very simplified)

How is MilkDrop working?

- ▶ Distortion and waves are controlled by fully customizable equations
- ▶ The set of those equations is called a “preset”
- ▶ Similar to a “patch” in PureData
- ▶ Interaction of the visuals with sound is defined by those equations
- ▶ ...and also with DMX and MIDI in Milkymist

Challenges

- ▶ The need for a CPU
 - ▶ flexibility
 - ▶ ease of reprogramming
 - ▶ ease of patching software bugs
 - ▶ software-friendly tasks: GUI, filesystems, protocols, ...
- ▶ FPGA speed, size, and cost
 - ▶ careful design
 - ▶ balance between hardware and software
 - ▶ software is cheap and slow, hardware is expensive and fast
- ▶ Memory problems: bandwidth, size
- ▶ Compute-intensive operations
 - ▶ distorting the image
 - ▶ evaluating the equations

System architecture

“System-on-a-Programmable-Chip” (SoPC):

- ▶ A microcontroller is implemented using the FPGA fabric
- ▶ The software is written and compiled (GCC)
- ▶ The compiled firmware is flashed/loaded on the board
- ▶ The microprocessor made from the FPGA fabric executes it
- ▶ Acceleration units and specialty peripherals are added to the system and driven by the software

Case studies — The memory problem

The memory problem

- ▶ A tough one.
- ▶ The application requires memory to be big, fast, and cheap at the same time.
- ▶ The required memory size prohibits the use of SRAM
- ▶ ...then we have to use DRAM and face all its problems.

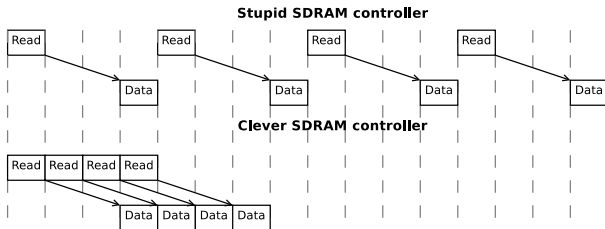
How bad is it?

Task	Bandwidth	Capacity
VGA framebuffer, 1024x768, 75Hz, 16bpp	900Mbps	3MB
Distortion, 1024x768, 30fps, 16bpp	720Mbps	–
2xNTSC input, 720x576, 30fps, 16bpp	380Mbps	3MB
Software and misc.	250Mbps	16MB
Total	2.2Gbps	22MB

- ▶ One DDR SDRAM chip running at 100MHz:
(DDR-200 in marketingspeak)
 - ▶ 3Gbps peak bandwidth
 - ▶ 32MB capacity
 - ▶ a few dollars
- ▶ This is still manageable!

Peak bandwidth?

Performance of SDRAM depends a lot on the cleverness of its controller. Example:



In Milkymist, memory transfers are always done using bursts of 4 consecutive words (FML bus). The bus master caches or discards the data it does not want.

Bursts of 4 consecutive words: a good heuristics?

- ▶ Good for the VGA framebuffer (a big bandwidth consumer):
 - ▶ when it gets a burst of 4 consecutive chunks from memory...
 - ▶ those chunks also represent consecutive pixels (in scan order)
 - ▶ ...so it can just put them in its output FIFO and easily achieve 100% utilization!
- ▶ It is the same for PAL/NTSC video inputs.
- ▶ For image distortion: yes; more on this later.
- ▶ For software: principle of temporal/spatial locality, caches.
- ▶ And simplifies a lot (compared to a variable burst length):
 - ▶ SDRAM controller design
 - ▶ Bus signaling
 - ▶ Bus arbitration
 - ▶ Cache controllers

Case studies — Distorting the image

What is “distortion”?



More precisely...

- ▶ “Texture mapping” in OpenGL terminology
- ▶ The MilkDrop preset equations define a mapping $f : (x, y) \rightarrow (X, Y)$
- ▶ The point at (x, y) in the source image should be at (X, Y) in the destination image
- ▶ Example: zoom by a factor of 2: $f(x, y) = (2 \cdot x, 2 \cdot y)$
- ▶ Problem with pixels: how to fill the “holes”?!
- ▶ Tessellation, triangle strip

Performance constraints

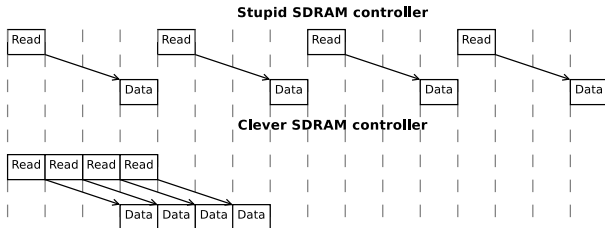
- ▶ To achieve 30fps at 1024x768, the Texture Mapping Unit (TMU) must process 24 million pixels per second.
- ▶ With a 100MHz clock, we have 4 cycles to process a pixel.
- ▶ With a naive implementation, each pixel requires (approx.):
 - ▶ math: ≈ 100 cycles
 - ▶ one memory read: ≈ 12 cycles
 - ▶ one memory write: ≈ 12 cycles
- ▶ Total: ≈ 124 cycles!!! Solutions for a $\approx 3000\%$ speedup?!?

Solutions

- ▶ Improved algorithm
 - ▶ Bresenham's linear interpolation algorithm
- ▶ SIMD parallelism
 - ▶ the same operation on independent data can be done in parallel
 - ▶ example: computing X and Y in the source image
- ▶ Pipelined parallelism
 - ▶ Milkymist's TMU has 14 pipeline stages
 - ▶ will grow as more features are added...
 - ▶ commercial GPUs can have hundreds
- ▶ Smart memory access
 - ▶ cache
 - ▶ write buffer

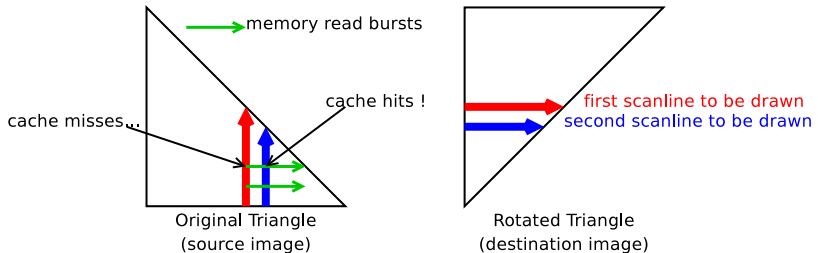
Solutions

- ▶ Let's focus on the memory read problem.
 - ▶ others would need deep explanations of the algorithms.
- ▶ Currently takes ≈ 12 cycles, and we have at most 4.
- ▶ Use the bursts, and memorize them in a cache.



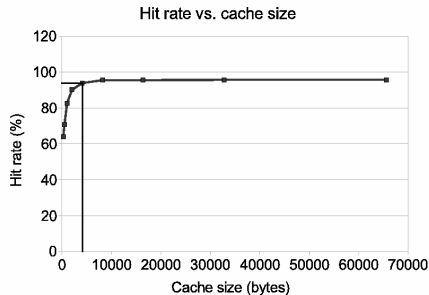
Are bursts and cache a good idea?

Example: rotation of a triangle.



It can work !

How big should the cache be?



- ▶ Cache size: 4KB
- ▶ Hit rate: 94%
- ▶ Average memory access time: 1.4 cycles
- ▶ Problem solved!

Case studies — Evaluating MilkDrop equations

- ▶ Some performance tricks are also used here
- ▶ Floating-point hardware coprocessor (PFPU)
- ▶ The main CPU compiles code on-the-fly for the coprocessor

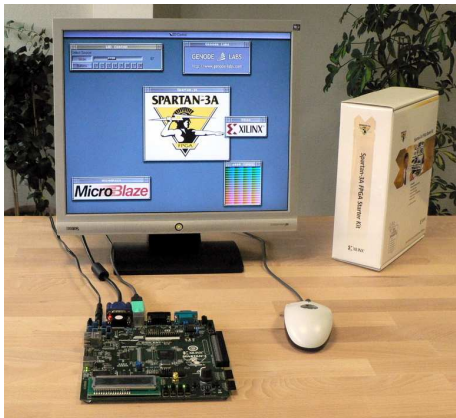
Linux

- ▶ noMMU for now. Who wants to contribute a LM32 MMU?



Genode FX

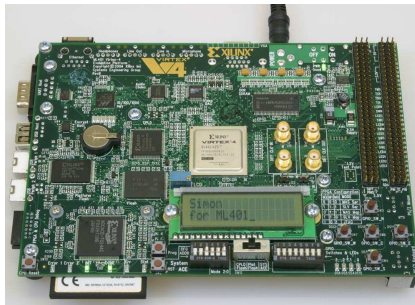
- ▶ Embedded GUI toolkit



In practice

Development board

- ▶ Most developments were done on a Xilinx ML401 board.
- ▶ Feature-rich.
- ▶ Affordable: \approx USD 500.



Verilog simulations

- ▶ Free simulators
 - ▶ Icarus Verilog
 - ▶ GPL Cver
 - ▶ Verilator, innovative and amazingly fast, but currently buggy
- ▶ Light-weight, run without a hitch on ultra-mobile laptops (EeePC)
 - ▶ debug your Verilog code anytime and anywhere
 - ▶ try to do the same with Modelsim and FLEXnet licensing!
- ▶ Expensive proprietary simulators from big EDA companies were not needed to develop Milkymist.

Logic synthesis

- ▶ ISE Webpack from Xilinx.
- ▶ Proprietary but free of charge.
- ▶ Linux version.
- ▶ Pretty good quality if you stick to command line.

Conclusion

- ▶ Wide availability of FPGAs opens new fields for hobbyists.
- ▶ High-speed embedded computing, and applications.
- ▶ This talk has presented a couple of high-performance logic design techniques.
- ▶ I did not invent them. They are ubiquitous in the computer industry, but seldom heard of.

Thank you for your attention

- ▶ Web: <http://www.milkymist.org>
 - ▶ documented source code (GPLv3 licensing)
 - ▶ binary kits (to get started fast)
 - ▶ mailing list
 - ▶ wiki (with suggested contributions)
 - ▶ blog
 - ▶ these slides are online (GNU FDL licensing)
- ▶ Mail: [sebastien.bourdeauducq \[AT\] lekernel DOT net](mailto:sebastien.bourdeauducq@lekernel.net)

Questions?