Ilya Neganov

ineganov@gmail.com

SUMMARY

Background in hardware, software, and math. Experience in modelling, debugging and bringup.

EXPERIENCE

Apple, Hardware Design Engineer **2018 – Present**

- I'm working in Silicon Engineering Group on the next generation graphics. *Verilog*

Imagination Technologies (MIPS), Leading Hardware Design Engineer **2014 – 2018**

- MIPS I6500, Bus Interface & Load Store units *Verilog*
  - Bus Interface Unit Ownership
  - New MT coherency & interlock features
  - ISO-26262 reliability features in Bus Interface Unit
  - Cache coherency FSM verification and automated test pattern generation
- MIPS I6400, Load Store Unit *Verilog*
  - Store Buffers RTL implementation from the ground up
  - Debugging MT & MC cache coherency protocol
  - Timing closure
  - Linux boot bringup and userland debugging on hardware emulator platform

Tupolev PSC, Leading Engineer **2009 – 2013**

- Airborne Universal Time System *C, Verilog*
  - Digital feedback system design for a GNSS-disciplined oscillator
  - Interface protocol implementation (ARINC429, MIL-STD 1553, IRIG-B, IRIG-106)
  - NTP Slave implementation
- Airborne Data Acquisition System *C++, Verilog*
  - PCB design and bringup from the ground up
  - ADC frontend digital filter design and precision evaluation
  - Analog IO design simulation
  - Embedded software development

EDUCATION

MS, BS in Physics and Technology (Bauman Moscow State Technical University, '04 – '10)

NON-WORK PROJECTS

- Convolutional Neural Network Accelerator (WIP) *https://github.com/ineganov/cnn*
- Alpha Processor on FPGA *https://github.com/ineganov/alpha*
- MIPS Processor on FPGA *https://github.com/ineganov/cpu_4*
- Quadcopter flight stabilization *https://github.com/ineganov/flight_control*
- Oversampling Audio DAC over UDP
- NOAA Satellite APT Imagery receiver & decoder

MISCELLANEOUS

- Languages: Russian mother tongue. Fluent English. Rudimental German. Willing to learn any language
- Work Authorization: UK ILR
- Private Pilot

Ilya Neganov

ineganov@gmail.com

EXPERIENCE DETAILS, MIPS I6400

I have joined MIPS Technologies as a Hardware Design Engineer and have been tasked with implementing Store Buffers under a tight schedule. Store Buffers implementation was up to a microarchitecture spec, with several improvements proposed and implemented by the author. Zero lines of code have been written on the unit when I arrived; both speculative and committed parts were fully functional within a year. I have been working closely with other unit teams over multiple timezones to bring up the core in concert.

The most challenging part of the design was figuring out all the aspects of cache coherency and consistency in a multithreaded, multicore environment; the most engaging – debugging intricate Linux boot failures on the emulator platform.

The next year has been devoted to LSU Control RTL improvements, timing closure, coverage in all its forms and working together with DV team on cache coherency protocol model. Non-LSU RTL had to be improved as well: I had to implement debug and trace features on the core side.

EXPERIENCE DETAILS, MIPS I6500

With I6400 Core finished and released, I've been given a Bus Interface Unit codebase to look after, along with implementing new Load-Store Unit features. This time I got to work on defining the microarchitecture spec, collaborating together with senior members of the team. I took part in speccing up the new inter-thread communications feature as well as Data ScratchPad RAM Store path. Bus Interface unit has been improved with functional safety features. On the DV side, I have written a tool that sanity-checks cache coherency FSM spec in the BIU. The tool generates test scenarios that reach 100% functional coverage according to the spec.

Currently, I am working together with the architecture team evaluating and quantifying features for the future MIPS Cores.

TOOLS

Mostly, I'm working with SystemVerilog code under Linux, using Synopsys toolchain: VCS along with DVE and Design Compiler. Occasionally I'm using Cadence ncsim with simvision. I'm familiar with both major FPGA vendor flows. I've also worked with Mentor Graphics Veloce emulation platform.

Scripting helps a lot in debugging the complex design, so I'm using whatever the environment provides: small shell scripts, in-tool tcl scripts, makefiles, perl & python design flow scripts etc.

On the software side, I have strong C knowledge. My C++ is a bit rusty, but functional enough to use it in tools like gem5 for uarch modelling. Being a CPU designer, I have to work with various assemblers. This is also helpful in embedded software design, where I have a tendency to avoid big libraries and only link in the essentials. And on the opposite side of the spectrum, there are functional languages like Lisp and Haskell, which I use whenever I can for my own entertainment.

Having hand-on experience with DSP design, I am quite familiar with Matlab. But nowadays I prefer Julia, as it is both faster and more expressive.