Ghosts of HLS past, present and future

This is mostly an adaptation of my position statement as requested by Brian Bailey Consulting.

- The current state of HLS compared to the original expectations for it 10 years ago

I think that these past few years, a lot of interesting developments occurred in the HLS field, especially in the programmable/FPGA realm. Essentially, 3rd generation high-level synthesis tools and environments made a successful, yet belated, entry in the FPGA market. Here, I'm arbitrarily making a distinction among 1st generation HLS tools (academic endeavors of the 80's), 2nd generation HLS tools that made the ASIC market in the 90's (e.g. Behavioral Compiler), and the current generation with usable high-level language frontends, rich optimization portfolios, IP integration and verification facilities. For these tools, the entry bar has lowered significantly from tens of thousand USD to about 2-5k USD and this really helps broader adoption. Technology vendors are not interested in selling their HLS tools but the entire platforms instead.

On the other side, it is much more difficult for tools of this grade to penetrate the ASIC market, where design failures are much more costly. There exist both software and hardware infrastructure issues. Most HLS offers lack the design space exploration and analysis tools that would allow a safer and faster assessment of QoR on multiple design points.

- Technological changes during the last 10 years. Where does the most potential lie.

I don't think that 3rd gen HLS tools encompass significant theoretical advances compared to what was achievable 10 or 15 years ago in core HLS; most of the theory (scheduling, resource sharing, retiming) was already there. Changes and improvements are incremental in effect. However, it is this new bunch of tools, that have usable C/C++/SystemC frontends and target accessible FPGA platforms that start to make a difference. There is potential for increased competition around MATLAB or Python to hardware. This market will be increasingly more important. Computationally-wise, bioinformatics will be big, big data of course, as well as non-Von Neumann computing, particularly neuromorphic computing for instance to map the mammal brain. Von Neumann will still be in use for emulating neurocomputers basically as a convenience.

- Is HLS a disruptive innovation that will shake the EDA industry? What has changed?

I think that HLS starts to find its place within the ESL flow. I don't believe that the prevailing view is of expecting HLS to be disruptive. It seems that we never really lacked HLS, but the flow was not there (in this sense I agree with Gabe on EDA). There

were a lot of things missing (interfaces, integration, frontends, competitive processes to ASIC) for HLS to be disruptive in the past.

- Estimating market size.

I would say that the estimated market has rised from a few million USD to maybe 30 to 50 million USD. In order to expand the pie, software-oriented engineers must ride the wagon of HLS, for instance algorithm developers. The majority of these engineers work with MATLAB, Python (or CUDA or OpenCL), so the corresponding language frontends have to be implemented. FPGA/SoC system engineers are either already heavily using HLS or considering extending their use of HLS technology. The easier converts are DSP engineers who clearly see the benefits of HLS in their day-by-day work, e.g. implementing matrix algebra. However, trusting HLS up to tapeout is a different story; I still see lots of manual interaction layers following the initial HLS outcome, primarily for interface modifications, old school optimizations and late adaptations (which should be back-propagated effectively by the current HLS tools in the first place).

A question rises here: what will ultimately replace RTL. If HLS is the answer, then the market size will first grow to the limits of the current RTL-powered market, and will then contract since it will have become a commodity. This transition will take about 12-15 years to complete; highly-customized functions such as device controllers will be the last stand.

- Who benefits from HLS?

Semiconductor companies from the Far East are known to be early, faithful adopters of HLS. I think that HLS has played a small part in their success, primarily in reducing time-to-market. A number of IP vendors use either third-party or homebrew, partial, HLS tools for streamlining the IP. Whenever an IP vendor releases a new non-trivial IP every one or two weeks, this is a typical sign for heavy HLS use :) Apart these, HLS should find each way to high-performance computing applications. HPC applications are most of the cases stencil codes, and the most troublesome part is to exploit and map task-level or processor-level parallelism. I think that there is much available room in HLS for HPC scientific computing. So companies offering HPC (either large or small form factor) design/programming services have lots to benefit from HLS.

- How has HLS progressed over the past 10 years?

In the most part, academic ideas are continuously transforming HLS. The key algorithms are established but (for instance) polyhedral frameworks are just starting out to be used in hardware compilers. Further, putting an intermediate representation at the heart of HLS is the right idea; frontends, backends, analyses and optimizations are naturally more easy to extend and maintain. Exposing this representation might also be of benefit to all parties.

- HLS in 10 years from now.

There are still lots of interesting things to happen in HLS:

1. Support very high-level, functional, dynamic, and concurrency-oriented specifications to HLS (there are multiple attempts towards these ends already).

2. Transparent preoptimization through intelligent code refactoring.

3. The high-quality, extensible, open HLS toolflow: an LLVM for HLS. A key part is missing at the backend side of the flow; it is very difficult for open-source projects like VPR/VTR and Torc to keep up with process advances.

4. HLS as a service: a superoptimizing hardware backend reusing its acquired knowledge for aggressive state optimizations running on the cloud.

5. Better tools at all levels: early assessment, design-space exploration, and analysis tools in the HLS environment.

6. Transparent development environments: ideally it will not be necessary to even know that we are using HLS, especially for hybrid, heterogeneous systems.

In 10-20 years from now, at or past the end of Moore, HLS will be the preferred choice for squizzing all performance potential out of 5nm or 8nm silicon. These last processes will be around for a decade or so. Universities will not offer advanced courses for ASIC/FPGA (no academic interest there); RTL will only be hobbyist, fun nonetheless. Still, most of the theory will be usable on graphene, carbon nanotube, organic or bio- processes and in general to whatever else will come in prominence.