Top 12 HercuLeS HLS user feedback patterns

As you might already know, HercuLeS by Ajax Compilers, Inc. (http://www.nkavvadias.com/hercules/) is a high-level synthesis (HLS) environment for the automatic translation of algorithms to hardware.

Since November 2013, the FREE version of HercuLeS has been made free to download, install and use for Windows 7 (64-bit) or Linux (32-bit and 64-bit): http://www.nkavvadias.com/temp/index.php

Well, this free/demo version has generated substantial user feedback. Dear users, wherever you are located (US, Canada, Japan, P.R. China, Sweden, Germany, UK, India, Brazil, the list is not conclusive by any means) thank you-thank you-thank you!!!

I have been compiling a short list of (let's say) the "Top 12" of user feedback patterns. Focusing on the more generic points, I am disclosing the list as food-for-thought and to generate even more feedback!

Top-12 user feedback patterns and concerns (not in any particular order)

1) Development time minimization (algorithm, early verification, RTL generation, simulation, implementation, late verification). HercuLeS simulation is blazing-fast compared to the SystemC-HDL cosimulation used by a competition tool. If you are simulating with this particular competition tool, then good luck and don't forget that you have enough time to prepare dinner for ~8 (yes, for all your folks, in-laws included)!

2) High result quality, reducing runtime requirements, chip area and power consumption (QoR). (Latest head-to-head out-of-the-box to Vivado HLS is a tie: http://nkavvadias.com/blog/2014/10/14/vivado-hls-vs-hercules-2/)

3) Readability of the generated HDL code. (HercuLeS code is much more readable than code generated by competition)

4) Source to IR to RTL to netlist cross-referencing (via means of "intelligent" cross-tagging). (It looks that an intelligent IDE is a whole new project.)

5) Theoretically provable correct-by-design approach. (I have been looking into automatic proving of code transformations.)

6) Transparent interface to the logic synthesis tool and up to downloading the bitstream. (The prototype version works and has been checked for my development boards.)

7) Plug-in approach to interconnecting legacy reusable designs with HLS-generated designs ("I have this piece of code in idiosyncratic C flavor or

assembly or FORTRAN..."). Looks like a call for implementing point solutions for select (aka paying) users.

8) Optimize HDL descriptions for specific implementation processes (e.g. FPGA devices); users don't actually seek portability!!!

9) Pthread or OpenMP frontend support. Explicit parallelism if you please! (My bet is with parallelism extraction but I get this important point)

10) VHDL frontend support (!) for behavioral VHDL to netlist HDL end-to-end flow. (There exist hard-to-the-core developers -- and greatness is with them -- that do their algorithmic exploration in behavioral VHDL. I usually do my own in C or VHDL and only lately I have been increasingly using MATLAB and Processing.)

11) Can you synthesize malloc and free? (Yes I can, and I am improving it, since until now I had been *intercepting* malloc and free and mapping them to a hardware-managed stack.)

12) Can you show me the automatic IP integration feature? (Yes I can, check this blog post as well: http://nkavvadias.com/blog/2014/10/13/hercules-overview/)

NOTE: quoted text is not a factual reply of an individual but rather artistic rendition thereof.