Vivado HLS vs HercuLeS

I've spent these last couple of days to perform head-to-head comparisons of Xilinx Vivado HLS against HercuLeS on HLS-generated digital circuits (from input C code).

I believe that HercuLeS lived up to the challenge; it is competitive to Vivado HLS. The reader should take account that:

- Both tools have been used (almost) out-of-the-box. Vivado HLS was configured with no bufg inclusion, and in "out_of_context" mode. These mean that no clock buffers and I/O pins were routed.
- HercuLeS does not (yet) customize the generated HDL in order to fit better specific architectural features (DSP blocks, embedded SRL units).
- Vivado HLS had some TOTAL FAILURES on some relatively simple codes such as a simple perfect number detector (positive integers equal to the sum of their divisors), a 1D wavelet code, and easter date calculation. It seems that Vivado HLS experiences some hard time with integer modulo/remainder. Codes are provided to anyone interested.

The following table provides a summary of the results:

Benchmark	Description	Vivado HLS (VHLS)			HercuLeS			Comments
		LUTs	Regs	TET	LUTs	Regs	TET	
arraysum	Array sum	102	132	26.5	103	63	73.3	
bitrev	Bit reversal	67	39	72.0	42	40	11.6	
edgedet	Edge detection	246	130	1636.3	680	361	1606.4	1 BRAM for VHLS
fibo	Fibonacci series	138	131	60.2	137	197	102.7	
fir	FIR filter	102	52	833.4	217	140	2729.4	Kintex-7
gcd	Greatest common divisor	210	98	35.2	128	93	75.9	
icbrt	Cubic root approximation	239	207	260.6	365	201	400.5	Virtex-6
popcount	Population count	45	65	19.4	53	102	26.1	
sierpinski	Sierpinski triangle	88	163	11327	230	200	16225	
sieve	Prime sieve of Eratosthenes	525	595	6108.4	565	523	3869.5	1 BRAM for VHLS

NOTES:

- Measurements where obtained for the KC705 development board device: xc7k325t-ffg900-2
- TET is Total Execution Time in ns.
- VHLS is a shortened form for Vivado HLS.
- Vivado HLS 2013.1 was used.
- Bold denotes smaller area and lower execution time.
- Italic denotes an inconclusive comparison.
- For the cases of edgedet and sieve, VHLS identifies a BRAM; HercuLeS does not. In these cases, HercuLeS saves a BRAM while VHLS saves on LUTs and FFs (Registers).

Overall, there are about 30% wins for HercuLeS and ~70% wins for Vivado HLS. Not too bad for a tool like HercuLeS; producing generic, portable, vendor-independent code. I estimate that HercuLeS development effort is around 1-5% to Vivado HLS.

I believe that HercuLeS will do much better in the out-of-the-box experience (which is of high importance in order to draw more software-minded engineers in the game) in the near future.

Both HercuLeS and Vivado HLS have optimization features (e.g. loop unrolling). HercuLeS applies optimizations by using a source-to-source C code optimizer. Vivado HLS mostly resorts to end-user directives. These coding aspects will be taken into account in a followup comparison; they also yield a much more extensive solution space.